POZNAN UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTING AND TELECOMMUNICATIONS

PHD THESIS

Alexandre QUEMY

End-to-End Approach to Classification in Unstructured Spaces

Application to Judicial Decisions

Thesis Advisor: Robert WREMBEL

Poznan, Poland, 2020

End-to-End Approach to Classification in Unstructured Spaces with Application to Judicial Decisions

Abstract: The main objective of this thesis is to develop a fully end-to-end machine learning workflow for the problem of classification, that is to say, without human interaction from the data transformation through a data pipeline, to the final prediction. In this doctoral thesis, we contribute to each step of the usual workflow, namely the data collection, the data pipeline creation and the model selection. To guide the development of our work, we chose to focus on the judicial domain as main field of application for all the constraints the field offer: grey areas of interpretation, non-monotonic reasoning, etc.
The first contribution concerns the data collection. We create the largest open access repository of documents related to the European Court of Human Rights and showed the quality of this repository by achieving 15pp gain in prediction accuracy compared to similar approaches on other datasets. Also, for the first time, we built efficient multilabel models capable of robust predictions using only descriptive features available before any judgment, opening the road to practical applications.
The second contribution concerns the model phase. We proposed a new Case-Based Reasoner based on hypergraphs. This new classification framework allows to work in unstructured spaces, i.e. space without a meaningful metric. It has been shown to be the most robust accross 11 datasets, has very few hyperparameters and does not require to transform the data to work which lowers the expertise required to obtain predictions. We studied an extension of its decision rule under the form of another classification problem and used a pure physical approach via heat propagation to solve the problem.

Last, we use bayesian optimization to automatically build a data pipeline in order to maximize the performances of the final model. For this purpose, we reused the tool usually dedicated to select and tune a model. We proposed an architecture to allocate the computation time between building the pipeline and tuning the algorithm and studied several time allocation policies. We proposed a new indicator to determine whether a data pipeline is algorithm-specific or rather universal to a dataset such that we can propose it to the user on a similar dataset.

Keywords: Classification, AutoML, End-to-End Machine Learning, Legal Analytics, Hypergraph

Streszczenie rozprawy

Remark: The author of this abstract is NOT a native Polish speaker. For this reason, the Polish text may have some flaws.

I. Wprowadzenie i motywacja

Uczenie maszynowe - machine learning (**ML**) do początku XXI wieku przeżywa gwałtowny rozwój, stymulowany nie tylko zastosowaniami praktycznie w każdej dziedzinie naszego życia, ale także stale wzrastającą i taniejącą dostępną mocą obliczeniową (m.in., obliczenia na procesorach graficznych, obliczenia rozproszone w chmurze, superserwery z ogromną pamięcią RAM).

Typowy *przepływ zadań uczenia maszynowego* (przedstawiony na rys. 1) składa się z dwóch następujących części: (1) potoku danych (ang. data pipeline, data processing pipeline) i (2) budowania modelu (ang. model building). *Potok danych* obejmuje zadania znalezienia prawidłowej sekwencji transformacji wejściowego zbioru danych tak, aby wyjściowy zbiór danych nadawał się do przetworzenia algorytmem uczenia maszynowego. *Budowanie modelu* obejmuje zadania wyboru właściwego algorytmu uczenia maszynowego i jego hiper-parametrów tak, aby zbudowany model zapewniał dobrą generalizację w odniesieniu do danej miary wydajności.

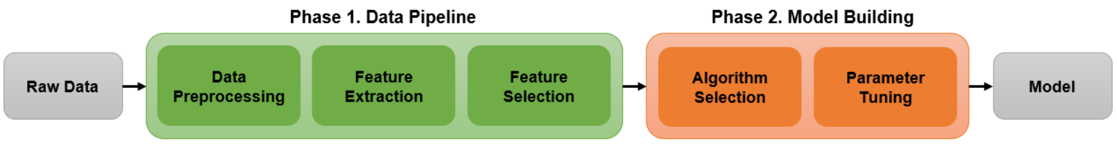


Figure 1: Typowy przepływ zadań uczenia maszynowego.

Ważnym trendem badawczym w zakresie uczenia maszynowego jest tzw. *kompleksowe uczenie maszynowe - end-to-end machine learning* (**E2EML**). Odnosi się ono do systemów, które są zdolne do budowania modeli z surowych danych bez ingerencji człowieka. Zwykle proces ten obejmuje czyszczenie i wstępne przetwarzanie danych, wybór odpowiedniego algorytmu i dostrojenie jego hiper-parametrów, podobnie jak przedstawiono na rysunku 1. Hiper-parametry algorytmu to parametry, które sterują procesem uczenia maszynowego, tj. wspomagają algorytm w znalezieniu dobrego modelu. Hyper-parametrów algorytm nie uczy się na podstawie danych ale są one dobierane ręcznie, są to np.: liczba drzew w lesie losowym (ang. random forrest), współczynnik uczenia (ang. learning rate) - w przypadku algorytmu gradientu prostego (ang. gradient descent) lub współczynnik mutacji (ang. mutation rate) - w przypadku algorytmu genetycznego (ang. genetic algorithm).

Zbudowanie wysokiej jakości modelu uczenia maszynowego dla zastosowań w przemyśle jest trudnym, czasochłonnym i złożonym obliczeniowo zadaniem, wymagającym wiedzy eksperckiej. Jest to główną przeszkodą dla powszechnego zastosowania rozwiązań ML i E2EML w firmach. Mimo, że większość firm gromadzi ogromne wolumeny danych w trakcie swojej działalności, możliwości analizy tych danych są ograniczone brakiem wystarczającej liczby pracowników posiadających odpowiednią wiedzę z zakresu przetwarzania danych i uczenia maszynowego.

Z raportów ekspertów z dziedziny wynika, że od 50% do 80% czasu i zasobów w projektach data science jest poświęcane na budowanie przepływów zadań dla uczenia maszynowego i analizy danych [Chessell 2014, Pa1 2018, Cog 2019]. W tym procesie, równie ważnym jak budowanie samego potoku danych (por. rys. 1) jest wybór i strojenie algorytmu ML budującego model. Najnowsze rozwiązania częściowo umożliwiają budowanie wydajnych potoków danych i wybór dobrego modelu. Na etapie wyboru modelu, meta-optymalizatory są w stanie automatycznie wybrać algorytm i dostroić model bez pomocy użytkownika, kosztem dużego narzutu czasu obliczeniowego. Jednakże należy zauważyć, że rozwiązania te nie są szeroko stosowane ze względu na wymagania obliczeniowe i konieczną wiedzę ekspercką. W tym kontekście, dostrajanie hiper-parametrów często w ogóle nie jest realizowane [Couronné 2018]!

Jedną a technik ML najczęściej wykorzystywanych w praktyce jest klasyfikacja (and. classification). Polega ona na zbudowaniu modelu umożliwiającego przewidzenie na podstawie pewnych cech obiektu, do jakiej klasy należy ten obiekt. Wiele rzeczywistych scenariuszy może być modelowanych jako problem klasyfikacyjny. Przykładowo, techniki klasyfikacji są z powodzeniem stosowane w medycynie - do celów diagnostycznych, w bankowości - do oceny zdolności kredytowej, przemyśle spożywczym - do klasyfikowania produktów według jakości, w handlu - do budowania profili zakupowych klientów, czy w telekomunikacji - do podziału klientów ze względu na ich profile zachowania się.

Na drugim końcu spektrum zastosowań technik ML jest dziedzina prawna, która jest jedną z najsłabiej wspieranych przez te techniki. Dziedzina ta ma bardzo wysoki potencjał aplikacyjny technik ML. Pomoc systemowi prawnemu w podejmowaniu lepszych decyzji zmniejszyłaby koszty i stronniczość, tym samym dając każdemu obywatelowi szerszy dostęp do obiektywnego wymiaru sprawiedliwości. W ostatnich latach obserwuje się rosnące zainteresowanie firm informatycznych dostarczaniem nowoczesnych narzędzi dla dziedziny prawa. Firmy te, znane jako LegalTech, rosną w imponującym tempie (25% wzrostu rocznie na rynku szacowanym na ponad 1 miliard USD rocznie[1].

Prawo jest złożoną dziedziną (ang. messy concept) [Rissland 2006], która z natury rzeczy stwarza szereg trudności dla algorytmów uczenia maszynowego. Trudności te to przede wszystkim: możliwość wieloznacznej interpretacji, pojawiające się wyjątki od reguł, niestacjonarność obserwacji, rozumowanie dedukcyjne i indukcyjne, logika nieklasyczna. Co więcej, modele statystyczne często działają na zasadzie czarnej skrzynki, co znacznie ogranicza ich praktyczne zastosowania. Podobnie, uczenie maszynowe oparte o sztuczne sieci neuronowe nie jest możliwe do zastosowania w dziedzinie prawnej, ze względu na trudność związaną z objaśnianiem zbudowanego modelu orzekania o winie, a tym samym trudność uzasadniania wyroków otrzymanych przez taki model. Innymi słowy, dziedzina prawna łączy niektóre z najtrudniejszych wyzwań dzisiejszego uczenia maszynowego.

Podsumowując, konieczne jest zatem opracowanie technik ograniczających wymaganą wiedzę ekspercką i udział człowieka w budowaniu kompleksowych przepływów danych w uczeniu maszynowym bez drastycznego wydłużania czasu obliczeniowego, we wszystkich dziedzinach zastosowania technik ML. W przeciwnym razie, koszt przyjęcia rozwiązań w zakresie ML pozostanie wyższy niż utrzymanie obecnie istniejących, mniej wydajnych pro-

[1] `https://prismlegal.com/legal-tech-market-sizing-and-opportunities/`

cesów, szczególnie w dziedzinach tradycyjnie dalekich od dziedzin technicznych, takich jak dziedzina prawna.

II. Cel i zakres rozprawy

W tym kontekście, głównym celem niniejszej rozprawy doktorskiej jest opracowanie w pełni zautomatyzowanego (kompleksowego) rozwiązania wspierającego budowanie *modelu klasyfikacji* bez udziału człowieka. Rozwiązanie to ma pracować z dowolnymi typami danych, co implikuje konieczność budowania modeli dla przestrzeni bez żadnych metryk (ang. non-metric space). Jako dziedzinę aplikacyjną wybrano przewidywanie decyzji sądowych ze względu na wyzwania, jakie stwarza ta dziedzina oraz ze względu na to jak niewiele rozwiązań zastało zaproponowanych do tej pory w literaturze naukowej i dostępnym oprogramowaniu komercyjnym i niekomercyjnym.

Niniejsza rozprawa doktorska stara się odpowiedzieć na trzy następujące *pytania*:

- Czy algorytm klasyfikacji może nauczyć się modelu w przestrzeni bez żadnych metryk?
- W jakim stopniu przygotowanie danych wpływa na jakość modelu predykcji, tj. czy ważniejsze jest przygotowanie danych dla algorytmu, czy strojenie tego algorytmu?
- Jak skutecznie zautomatyzować fazę przygotowania danych (potok danych)?

W niniejszej rozprawie stawiamy dwie *hipotezy*.

- Po pierwsze, algorytm uczenia maszynowego może nauczyć się metryki na samych danych w oparciu o informacje zwrotne dostarczone przez zbiory uczące (ang. learning set).
- Po drugie, aby zbudować model, który będzie poprawnie działał także na nowych (nieznanych) danych, do budowy tego modelu należy dostarczyć danych wysokiej jakości, tj. jakość danych jest ważniejsza niż sam algorytm. Mówiąc dokładniej, jeżeli dany jest algorytm, który zachowuje się jak uniwersalny aproksymator [Csáji 2001], czyli jest w stanie nauczyć się prawie każdej funkcji ciągłej na zwartym podzbiorze R^m, wtedy głównym praktycznym ograniczeniem dla procesu uczenia jest jakość danych. Dla określonego budżetu czasu zakładamy, że ważniejsze może być poświęcenie większej jego części na wstępne przetwarzanie danych, niż na wybór algorytmu i dokładne dostrojenie jego hiper-parametrów.

III. Aktualny stan wiedzy

Przedstawiona w tym punkcie analiza stanu wiedzy dotyczy dwóch dziedzin objętych zakresem niniejszej rozprawy, tj. uczenia maszynowego (w szczególności zautomatyzowanego) i zastosowania technik ML w domenie prawnej.

Zautomatyzowane uczenie maszynowe i kompleksowe uczenie maszynowe

Techniki tradycyjnie nazywane jako *zautomatyzowane uczenie maszynowe* (**AutoML**) lub wspomniane wcześniej kompleksowe uczenie maszynowe, koncentrują się w praktyce na problemie łączenia algorytmów i optymalizacji hiper-parametrów - nazywanym dalej CASH (ang. combined algorithm selection and hyperparameter optimization - CASH) [Kotthoff 2017, Feurer 2015]. Podejście takie całkowicie pomija znaczenie potoków danych dla jakości modelu [Crone 2006], koncentrując się na wyborze algorytmu i dostrajaniu hiper-parametrów. Metoda sekwencyjnej optymalizacji w oparciu o model [Hutter 2011] (ang. Sequential Model-Based Optimization) może być zrealizowana na różne sposoby, między innymi przy użyciu Lasu Losowego [Hutter 2011], tzw. Estymatora Tree-Parzen [Bergstra 2015], lub Regresji Gaussa [Martinez-Cantin 2014].

W przypadku potoku danych i wstępnego przetwarzania, większość rozwiązań wykorzystuje półautomatyczne narzędzia wspierające naukowców danych (ang. data scientists). W [Polyzotis 2017] stosuje się wytyczne do weryfikacji jakości wstępnie przetworzonych danych w ciągłym uczeniu maszynowym, tj. modelach uczenia maszynowego w produkcji i otrzymywaniu w sposób ciągły nowych danych treningowych. Ostatnio zaproponowano metodę wykorzystującą meta-atrybuty do oszacowania wpływu operatorów przetwarzania wstępnego na dokładność modelu [Bilalli 2017]. Podejście to tworzy ukrytą przestrzeń wykorzystując meta-atrybuty (np. liczbę klas lub atrybutów, entropię, stosunek sygnału do szumu), w których można przedstawić dowolny zbiór danych. Moduł zwany meta-learner jest uczony na kilku różnych zbiorach danych. Meta-model jest zatem w stanie przewidzieć wpływ zastosowania różnych technik transformacji danych w potoku danych na jakość budowanego modelu predykcji, bez konieczności uczenia modelu i jego oceny za pomocą np. walidacji krzyżowej. Wreszcie, w innym podejściu użytkownik przekazuje do systemu informację zwrotną na temat jakości danych w celu optymalizacji przepływów [**?**].

Uczenie metryki polega na wyborze właściwej metryki, która umożliwia prawidłowe porównanie lub klasyfikację danych [Bellet 2013, Wang 2015]. Wybór odpowiedniej metryki do pomiaru odległości między dwoma punktami jest kluczowy dla jakości algorytmów klasyfikacji [Davis 2007]. Uczenie metryki polega na znalezieniu rzutu f z przestrzeni początkowej na przestrzeń euklidesową, tak że dla dowolnych elementów $\mathbf{x}$ i $\mathbf{x}'$, $d(\mathbf{x},\mathbf{x}') = ||f(\mathbf{x}) - f(\mathbf{x}')||$. Metryka powinna odzwierciedlać różnicę semantyczną między obiektami. Zaskakująco, większość metrycznych metod uczenia zakłada, że dane są początkowo reprezentowane w przestrzeni wektorowej, co może nie być właściwe dla wielu problemów, w których mogą pojawiać się dane ustrukturalizowane, częściowo ustrukturalizowane, lub nieustrukturalizowane.

Domena prawna

Jako dziedzina aplikacyjna rozwiązań opracowanych w ramach niniejszej rozprawy został wybrany wymiar sprawiedliwości. Nieliczne opublikowane wcześniej badania naukowe w zakresie stosowania uczenia maszynowego do wspomagania decyzji sądowych pokazały, że domena prawna jest szczególnie interesująca i trudna dla algorytmów uczenia maszynowego. Po pierwsze, ze względu na wielość i złożoność reguł prawnych oraz złożoność semantyczną aktów prawnych. Po drugie, ze względu na brak jednolitego repozytorium aktów prawnych i orzeczeń sądowych. Po trzecie, decyzje sądowe zmieniają się w czasie dla podobnych przypadków (tj. nie występuje stacjonarność obserwacji) i obserwuje się wielość odstępstw od reguł w wydawaniu orzeczeń. Zatem opracowanie całościowego (zautomatyzowanego) podejścia do budowania modeli klasyfikacji dla wspomagania decyzji sądowych ma ogromny potencjał praktyczny (wdrożeniowy).

Przewidywanie decyzji sądowych stanowi wyzwanie samo w sobie, nawet dla najlepszych ekspertów prawnych: w przypadku *Supreme Court of the United States* (SCOTUS) osiągnięto 58% dokładność [Ruger 2004]. Natomiast projekt Fantasy SCOUTS[2], w którym mamy odczynienia z ogromną grupą wolontariuszy przewidujących jak dany członek Sądu Najwyższego Stanów Zjednoczonych będzie orzekał w danej sprawie, osiągnął 84,85% poprawnych prognoz. Brak jest podobnych wyników dla orzecznictwa europejskiego, za wyjątkiem badań na małych zbiorach danych [Aletras 2016].

Dotychczas zaproponowane podejścia do przewidywania decyzji sądowych można podzielić na trzy grupy: (1) modele statystyczne, (2) wnioskowanie na podstawie przypadków (ang. Case Based Reasoning - CBR) i (3) abstrakcyjną argumentację (ang. Abstract Argumentation – AA).

Modele statystyczne wykorzystano do przewidywania werdyktów sądu amerykańskiego - *Supreme Court of the United States* [Katz 2017a, Martin 2004b, Guimerà 2011]. Zgodnie z naszą najlepszą wiedzą, w odniesieniu do *European Court of Human Rights* istnieje niewiele modeli predykcji [Aletras 2016, Medvedeva 2020, Chalkidis 2019]. Zbiór danych użyty w [Aletras 2016] obejmuje wyłącznie kilka artykułów prawnych, z których każdy zawiera od 80 do 254 przypadków. Wykorzystane w pracach [Aletras 2016, Medvedeva 2020] modele predykcyjne wykorzystują liniowy klasyfikator SVM osiągając od 75% do 79% dokładności predykcji (accuracy). W [Chalkidis 2019] wykorzystano sztuczne sieci neuronowe, uzyskując wartość miary F1 maksymalnie 82%

Podejście CBR wykorzystuje podobieństwa pomiędzy cechami i rozwiązaniami poprzednich obserwacji w celu zbudowania nowego rozwiązania dla nowego przypadku (w kontekście niniejszej rozprawy - nowej sprawy sądowej). Metody CBR nie uwzględniają czynników pozaprawnych, a zatem nie są w stanie poradzić sobie z problemem prognozowania. Metody te dostarczają natomiast uzasadnienia dla swoich decyzji [Aleven 1997].

Podejście AA polega na modelowaniu informacji jako graf argumentów i wyciąganiu wniosków poprzez rozwiązywanie konfliktów za pomocą logiki lub ważenia argumentów. Mimo, że metody statystyczne dostarczają interesujących wyników dla problemu prognozowania [Guimerà 2011, Martin 2004a, Ruger 2004, Katz 2017b], nie są one w stanie dostarczyć prawnego uzasadnienia swoich prognoz. W AA pojawiły się dwa rodzaje przeciwstawnych podejść: pozytywne, które mają na celu modelowanie rzeczywistych procesów decyzyjnych [Baroni 2015] i normatywne, które próbują opracować metody wyboru spośród najlepszych alternatyw i argumentów [Dung 2006]. Pierwsze podejście może dobrze wspierać rozwiązanie problemu prognozowania, a drugie - problemu uzasadnienia. Oba podejścia w dużej mierze polegają na wiedzy eksperckiej koniecznej do konstruowania tzw. argumentów, co ogranicza zastosowanie AA.

[2]https://fantasyscotus.lexpredict.com/

IV. Kontrybucja rozprawy

W niniejszej rozprawie proponujemy alternatywne podejście do konstruowania potoku danych z uczeniem maszynowym, które zaprezentowano na rys. 2. Proponowany potok danych zakłada, że typy i formaty danych przetwarzanych przez potok danych nie są z góry znane i mogą ewoluować podczas przetwarzania danych. Stanowi to problem, ponieważ nie wszystkie algorytmy uczenia maszynowego mogą obsługiwać dowolny typ danych. W szczególności niektóre algorytmy działają tylko z danymi liczbowymi lub wartościami ciągłymi, niektóre nie mogą działać, gdy pojawiają się wartości puste, lub są wrażliwe na wartości odstające.

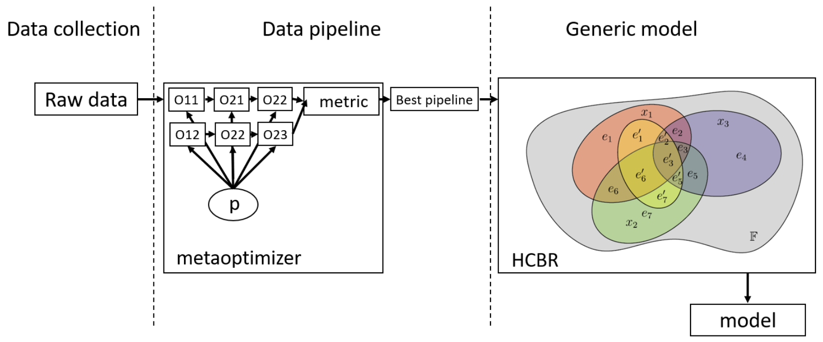


Figure 2: Zmodyfikowany przepływ zadań uczenia maszynowego zaproponowany jako rozwiązanie problemu kompleksowej klasyfikacji.

W szczególności, podejście zaproponowane w niniejszej rozprawie doktorskiej bazuje na trzech następujących rozwiązaniach.

- Po pierwsze, proponujemy **ogólny sposób automatycznego budowania i konfigurowania potoku** danych w celu przygotowania danych dla dowolnego algorytmu uczenia maszynowego. Konstrukcja potoku danych może być sformułowana jako problem optymalizacji, można go zatem rozwiązać automatycznie, w oparciu o istniejące meta-optymalizatory, przy wykorzystaniu minimalnej wiedzy specjalistycznej. Według naszej najlepszej wiedzy, dotychczas nie zaproponowano podobnego rozwiązania.
- Po drugie, proponujemy **zastosowanie metody Hypergraph Case-Based Reasoning** (HCBR), wykorzystującej zalety metod statystycznych, CBR i systemu argumentacji, jednocześnie unikając ich wad. W HCBR proponujemy zastosowanie generycznego algorytmu, który może przetwarzać dane dowolnego typu i uczyć się złożonych modeli, wykorzystujący przy tym niewiele hiper-parametrów lub nie wykorzystujący ich w ogóle. Dzięki temu, można zredukować czas potrzebny na budowanie modelu, bez konieczności udziału użytkownika.
- Po trzecie, opracowaliśmy **otwarte repozytorium danych prawnych**, zawierające sprawy sądowe i orzeczenia z Europejskiego Trybunału Praw Człowieka. Dane w repozytorium zostały wcześniej oczyszczone, uspójnione i przetransformowane (przez

zadania potoku danych) do postaci wymaganej przez algorytmy klasyfikacji. Repozytorium zostało upublicznione w postaci portalu (https://echr-opendata.eu/). Dzięki temu naukowcy z całego świata mogą korzystać ze zgromadzonych w nim danych i uruchamiać algorytmy uczenia maszynowego na danych przygotowanych do tego typu przetwarzania. Repozytorium stanowi tym samym benchmark dla algorytmów ML działających w domenie prawnej.

Budowa i optymalizacja zautomatyzowanego potoku danych

W prezentowanej rozprawie doktorskiej proponujemy, zgodnie z naszą najlepszą wiedzą, pierwszy ogólny sposób automatycznego budowania i konfigurowania potoku danych w celu przygotowania danych dla dowolnego algorytmu uczenia maszynowego. Zaproponowaliśmy zmodyfikowany przepływ zadań [Quemy 2020a, Quemy 2019b], zaprezentowany na rys. 3.

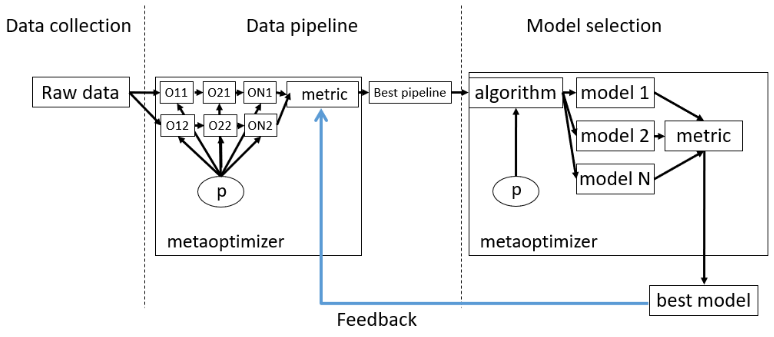
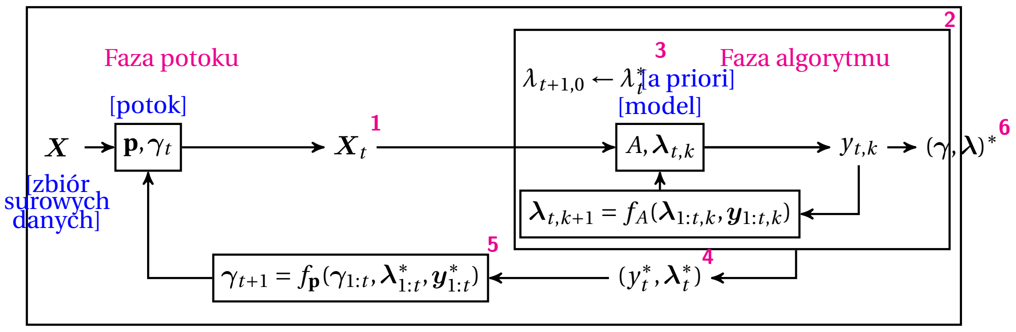


Figure 3: Przepływ zadań dla automatyzacji konstrukcji potoku danych. Główną ideą jest ponowne wykorzystanie meta-optymalizatora na podstawie danych zwrotnych dostarczonych przez działający model.

W celu wykazania potencjału takiego podejścia, zdefiniowaliśmy gramatykę, która umożliwia definiowanie potoków na wyższym poziomie abstrakcji, reprezentowanych jako grafy. Taką reprezentację nazwaliśmy prototypem potoku. Każdy węzeł może być utworzony za pomocą kilku operatorów (np. PCA), a każdy z nich ma swój własny zestaw parametrów (np. liczbę składników w PCA). Użytkownik końcowy nie musi posiadać żadnej wiedzy na temat tych operatorów.

W [Quemy 2020a, Quemy 2019b] zaproponowaliśmy dwu-etapowy proces optymalizacji budowania potoków dla uczenia maszynowego. Proces ten zilustrowano na rys. 4. W szczególności, zdefiniowaliśmy polityki alokacji czasu pomiędzy potok danych a algorytm budowania modelu. Pokazaliśmy, że często korzystniejsze jest przeznaczenie większej części czasu na konstruowanie potoku, niż na sam algorytm, oraz że polityki adaptacyjne podziału czasu pomiędzy potok danych a algorytm budowania modelu są lepsze niż polityki statycznego podziału.

Zaproponowane podejście zostało ocenione eksperymentalnie na wielu zbiorach danych i wielu potoków. Średnio, dla wszystkich testowanych zbiorów danych i metod, dla 20 potoków (0,42% przestrzeni przeszukiwania), zautomatyzowany proces był w stanie zmniejszyć błąd o 58,16% w porównaniu z podejściem, w którym cały dostępny czas został przeznaczony wyłącznie na strojenie hiper-parametrów.


Faza potoku
[potok]
X
$\mathbf{p}, \gamma_t$
X_t
1
[zbiór surowych danych]
Faza algorytmu
3
$\lambda_{t+1,0} \leftarrow \lambda_t^*$ [a priori]
[model]
$A, \boldsymbol{\lambda}_{t,k}$
$y_{t,k}$
$(\gamma, \boldsymbol{\lambda})^*$
6
2
$\boldsymbol{\lambda}_{t,k+1} = f_A(\boldsymbol{\lambda}_{1:t,k}, \boldsymbol{y}_{1:t,k})$
5
$\gamma_{t+1} = f_{\mathbf{p}}(\gamma_{1:t}, \boldsymbol{\lambda}^*_{1:t}, \boldsymbol{y}^*_{1:t})$
4
$(y_t^*, \boldsymbol{\lambda}_t^*)$

1 Pojedynczy potok przekształca cały zbiór danych podczas każdej iteracji.
2 Wyjście $y_{t,k}$ pętli wewnętrznej jest miarą poprawności (np. walidacja krzyżowa).
3 Pętla wewnętrzna jest inicjowana z poprzednią najlepszą konfiguracją (a priori).
4 W t iteracji, wewnętrzna pętla zwraca najlepszą predykcję i konfigurację.
5 f_M zwraca najbardziej korzystną konfigurację w odniesieniu do najlepszej osiągalnej metryki.
6 Cały proces zwraca najlepszą konfigurację do wykorzystania w praktyce.

Figure 4: Dwu-etapowy proces optymalizacji budowania potoków uczenia maszynowego.

Przykładowo, wyniki dla lasu losowego na zbiorze danych Breast[3] pokazano na rys. 5. Widoczny z lewej strony żółty rozkład konfiguracji badanych przez algorytm jest przekrzywiony w kierunku większej dokładności, wskazując, że statystycznie nasze podejście tworzy dobre potoki. Wykres po prawej pokazuje, że algorytm jest efektywny w znajdowaniu potoku, który działa prawie najlepiej w przestrzeni przeszukiwania.

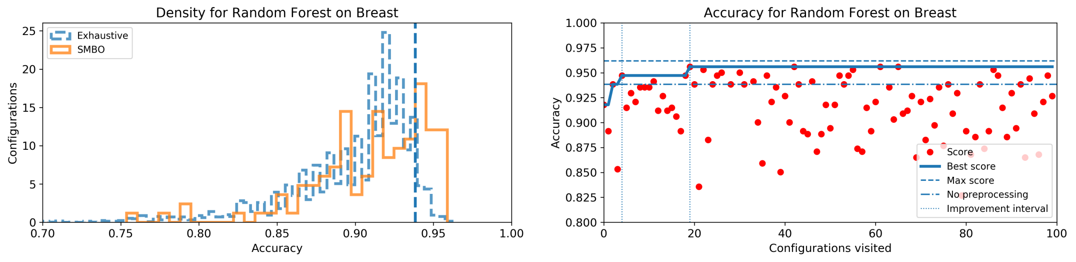


Figure 5: Przykład wyników dla lasu losowego na zbiorze danych Breast. Meta-optymalizator z większym prawdopodobieństwem próbkuje konfiguracje o wyższej dokładności.

Podsumowując, rozwiązując problem budowy i optymalizacji zautomatyzowanego potoku danych:

1. Wykazaliśmy, że wpływ konfiguracji potoku danych na dokładność klasyfikacji jest ogromny w porównaniu z wpływem wyboru hiper-parametrów i modelu.
2. Wykazaliśmy, że potoki danych można budować i konfigurować automatycznie przy użyciu istniejących meta-optymalizatorów, nawet przy ograniczonym budżecie obliczeniowym lub czasowym.

[3] `https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original)`

Hypergraph Case-Based Reasoning

Jako drugą kontrybucję w procesie budowy przepływu zadań (rys. 3) proponujemy zastosowanie generycznego algorytmu klasyfikacji - Hypergraph Case-Based Reasoning (HCBR) [Quemy 2019a, Quemy 2018b]. Może on przetwarzać dane dowolnego typu i uczyć się złożonych modeli. HCBR wymaga do pracy niewiele hiper-parametrów lub nie korzysta z nich w ogóle. Jak sugeruje nazwa, w HCBR, zbiór treningowy jest reprezentowany jako hiper-graf. HCBR do oszacowania stopnia przypisania danego podzbioru atrybutów do klasy jest wykorzystywana partycja utworzoną przez tzw. sub-hiper-grafy.

Zaproponowana w rozprawie kontrybucja posiada kilka interesujących własności, użytecznych nie tylko w zastosowaniu w dziedzinie prawnej. W szczególności, przestrzeń modelu i reprezentacja danych jako hiper-graf zapewnia wygodny sposób wyjaśnienia każdej decyzji osobno, w oparciu o interakcje z decyzjami z przeszłości (np. postrzeganymi jako „kontrprzykłady" lub „analogie" w przypadku procesu sądowego, podobnie jak dla wnioskowania opartego na przypadkach). Ponadto, wrażliwość HCBR na hiper-parametry jest znikoma, dzięki czemu czasochłonne dostrajanie nie jest wymagane dla użytkownika końcowego. Hiper-parametry można natomiast wykorzystać do kontroli ryzyka związanego z prognozą, lepiej dostosowaną do potrzeb reprezentowanych przez konkretną dziedzinę (np. sędzia woli podejmować mniejsze ryzyko decyzji fałszywie pozytywnych, tj. wysyłania niewinnych do więzienia, podczas gdy lekarz woli podejmować mniejsze ryzyko fałszywie negatywnych, tj. niewykrycia nowotworu). Wreszcie, HCBR nie zakłada żadnej metryki w przestrzeni atrybutów (ang. feature space). Na działanie algorytmu nie wpływa reprezentacja atrybutu i może on pracować z niekompletnymi lub nieustrukturyzowanymi zbiorami danych.

HCBR został zaimplementowany w różnych wariantach (użyto do tego C++) [4] i oceniony eksperymentalnie. Eksperymenty pokazały że:

- zaproponowany przepływ zadań działa tak samo dobrze jak zbudowany w oparciu o standardowe metody, dla kilku referencyjnych zbiorów nieustrukturyzowanych danych;
- zaproponowane rozwiązanie sprawdza się lepiej niż rozwiązania konkurencyjne [Aletras 2016] w zakresie przewidywania decyzji Europejskiego Trybunału Praw Człowieka;
- HCBR średnio osiąga lepsze wyniki przy braku wiedzy specjalistycznej w porównaniu z 9 innymi uznanymi metodami: AdaBoost, k-Nearest Neighbors, Linear SVM, Radius-Based Function (RBF) SVM, Decision Tree, Random Forest, Neural Network i Quadratic Discriminant Analysis (QDA).

Otwarte repozytorium Europejskiego Trybunału Praw Człowieka

Jak wspomniano, w ramach rozprawy opracowano kompleksowe podejście do budowania przepływów danych, dla zastosowań w dziedzinie prawnej. W celu oceny jego działania niezbędnym było zbudowanie repozytorium danych sądowniczych. Repozytorium to integruje dane z Europejskiego Trybunału Praw Człowieka. Trybunał publikuje dokumenty

[4]https://github.com/aquemy/HCBR

związane ze sprawami sądowymi w języku naturalnym. Aktualnie dostępnych jest ponad 50 000 decyzji, gromadzonych od czasu utworzenia Trybunału. Oryginalne dane są dostępne w kilku formatach, min., tabelarycznym, JSON bez elementów zagnieżdżonych, CSV. W ramach projektu, z dostępnych dokumentów wyroków wyodrębniliśmy i zunifikowaliśmy standardowe atrybuty opisowe (ang. descriptive features), tworząc: (1) relacyjną bazę danych zawierającą sprawy sądowe i meta-dane o tych sprawach i (2) złożoną reprezentację *bag of words* z wyroków sądowych (uporządkowaną według paragrafów). Wstępne przetworzenie oryginalnych dokumentów (potok przygotowania danych) zostało przeprowadzone za pomocą algorytmu *entity matching* dostępnego w IBM Watson Services.

W celu zapewnienia powtarzalności zaprojektowanego przepływu danych i umożliwienia oceny jakości powstałych danych:

- każda wersja zbiorów danych jest wersjowana i publicznie dostępna, w tym także pliki pośrednie w celu zapewnienia tzw. data lineage;
- integralność procesu i wytworzonych danych jest dokładnie dokumentowana;
- skrypty do pobierania nieprzetworzonych dokumentów i tworzenia zbiorów danych są wersjowane i ogólnie dostępne;
- żadne dane nie są przetwarzane ręcznie na żadnym etapie konstruowania przepływu danych.

W celu przetestowania mocy predykcyjnej zbudowanego repozytorium, przeprowadziliśmy wiele eksperymentów, m.in., porównując 13 standardowych algorytmów uczenia maszynowego do klasyfikacji pod względem kilku wskaźników wydajności. Otrzymane wyniki dla zbiorów danych binarnych cechują się **dokładnością** (ang. accuracy) w zakresie od 75,86% do 98,32% i średnią 96,45%. Ponadto, eksperymenty pokazały, że niektóre atrybuty nadają się lepiej do predykcji decyzji sądowniczych niż inne. W szczególności stwierdziliśmy, że atrybuty tekstowe (ang. textual features) dobrze nadają się do przewidywania (binarnego) wyniku. Jednak po raz pierwszy pokazaliśmy, że nie są one tak dobre jak atrybuty czysto opisowe (ang. descriptive features) do określenia jakiego artykułu dotyczy dany przypadek sądowy.

V. Podsumowanie

Głównym celem niniejszej rozprawy było zapewnienie nowego, wydajnego podejścia do procesu budowy kompleksowego przepływu zadań dla problemu klasyfikacji oraz weryfikacja opracowanego podejścia w zastosowaniach klasyfikacji dokumentów prawnych. Główne kontrybucje rozprawy obejmują:

- ogólne podejście do automatycznej budowy i optymalizacji potoków danych przy użyciu standardowych technik meta-optymalizacji [Quemy 2020a, Quemy 2019b];
- nowy ogólny model matematyczny do klasyfikacji w przestrzeni nieustrukturyzowanych, zwany Hypergraph Case-Based Reasoning [Quemy 2019a, Quemy 2018b];

- zbudowanie wysokiej jakości repozytorium danych prawnych, dostępnego dla społeczności zajmującej się analizą danych i uczeniem maszynowym w dziedzinie prawnej [Quemy 2020c, Quemy 2020b].

Acknowledgments

I would like first to warmly thanks Robert Wrembel for accepting to follow my doctoral project. Of course, I would never have met Pr. Wrembel without the help of Michał Bodziony from IBM. Additionally, I feel grateful for the plain support I received from IBM and my colleagues.

I would never have finished this dissertation if it was not for Natalia's daily support for years. Thank you! Kevin, you are bad at fishing but still deserve a spot here, in the acknowledgments, so thank you!

I dedicate this dissertation to Roger Goglu, my former algebra teacher. When I was just 19 years old, you helped me like no one, securing my spot at the Institute making sure I could continue my education. You were the first to take some news after the operation. When I came back after my convalescence, you were gone. I never had the time to say how much I was owe you and how grateful I am.

Contents

Notations

$<\mathbf{x},\mathbf{y}>$	Scalar product between $\mathbf{x}$ and $\mathbf{y}$
$\mathbf{X}$	Collection of vectors
$\mathbf{x}$	Vector
$\mathbf{a}_{:j}$	jth column vector of matrix A
$\mathbf{a}_{i:}$	ith row vector of matrix A
$A = (a_{ij})$	Matrix of components a_{ij}
x_i	i-th component of vector $\mathbf{x}$

Abbreviations

HCBR Hypergraph Case-Based Reasoning. viii, xi, 4, 5, 7, 33, 37, 38, 41, 42, 78, 81–83, 85, 87, 95, 97, 98, 100–107, 109, 111, 113–117, 151, 161, 162, 164, 166

AA Abstract Argumentation. 29–32

AA-CBR Abstract Argumentation-Case-Based Reasoning. 32

AF Abstract Framework. 30

AutoML Automated Machine Learning. viii, 7, 28, 33, 152, 167

BoW Bag-of-Words. 19, 38, 40, 56, 57, 61, 63, 67, 69–71

CBR Case-Based Reasoning. 7, 15, 27–33

DL Deep Learning. 1

DPSH Data Pipeline Selection and Hyperparamater Optimization. 136

ECHR European Court of Humain Rights. x, 4, 7, 18, 19, 49, 146, 147, 151

ECHR-DB European Court of Humain Rights Database. 146

ECHR-OD European Court of Humain Rights OpenData. 4, 7, 151

KNN K-Nearest Neighbors. 26

LDA Latent Dirichlet Allocation. 24, 25

LSTM Long Short-Term Memory. 1, 57

MAR Missing At Random. 41

MCAR Missing Completely At Random. 41

MCC Matthews Correlation Coefficient. 60–63, 66, 67

ML Machine Learning. 15, 27, 31, 33, 35, 50, 51

NLP Natural Language Processing. 18, 21, 25, 28, 31, 32

NMAD Normalized Mean Absolute Deviation. 145–149

PSD Positive Semi-Definite. 39, 40

SCOTUS Supreme Court of the United States. 3, 11, 15, 19

SVM Support Vector Machine. v, 19, 35–37, 41, 60, 66, 67

TF-IDF Term Frequency-Inverse Document Frequency. 52, 60, 74

TPE Tree-structured Parzen Estimator. 44, 45

CHAPTER 1

Introduction

Listen – I say that justice is nothing other than the advantage of the stronger.

THRASYMACHUS IN THE REPUBLIC, PLATO

Contents

1.1 Research Questions

End-to-end machine learning refers to systems that can build models from raw data without human intervention. It usually covers cleaning and preprocessing data, selecting an adequate algorithm and tuning its hyperparameters. Such systems require little expertise, which is currently the main barrier for a wider adoption of AI solutions in many fields.

There are many obstacles in order to achieve a fully end-to-end solution. For instance, the data needs to be preprocessed not only to fit the algorithm's input space but also to be certain that the algorithm learns properly, without bias and other traps implied by poor data quality. But even with high quality data, it is well known that *there is no free lunch*, i.e., there is no algorithm or algorithm configuration that is superior to any other for all possible tasks, problems and instances. Therefore, there is always a need to select the proper algorithm for the given problem.

Currently, end-to-end machine learning usually evokes Deep Learning (DL) [LeCun 2015]. DL represents the state-of-the-art in classification (and other AI tasks) in multiple domains [Schmidhuber 2015]: vision, audio and natural language processing to name a few, and is often presented as an end-to-end solution. DL in theory does not require to preprocess data, but in practice, each type of neural network has specific applications and require a specific type of data as an input. For instance, convolutional networks will work on images, while Long Short-Term Memory (LSTM) deals with text or more generally time series. The main drawback of DL is that, as soon as multiple data sources are involved, or the data structure does not match the input requirements of a given network, it is necessary to either transform data or to develop a new network architecture. Therefore, in practice, DL is rarely end-to-end from the user perspective. **The core problem is the need for a metric in the space of features.** On top of that, finding the proper neural architecture requires a lot of expertise, the results are highly dependent on hyperparameters and is computationally expensive. Thus, DL is the perfect candidate for specific applications, in an environment with highly trained people that have a lot of time and computational power to find the proper architecture, train and fine-tune the neural network. However, in many, if not most practical situations, people are not AI specialists or data scientists, and do not have the resources large corporations have. In this case, a DL-based solution might not be possible or suitable.

The main problem of this doctoral study is then to develop a full end-to-end solution capable of using multiple sources of arbitrary data and build an efficient machine learning pipeline without any human intervention. To narrow down the problem, we will consider the specific **problem of classification**. The requirement to work with arbitrary data types implies spaces without any metric and thus, our hypothesis is that a machine learning algorithm needs to learn a metric over the data itself, based on the feedback provided by training examples.

Another hypothesis is that to build a model that generalizes well, the quality of data is more important than the algorithm. More specifically, once we have an algorithm that behaves as a universal approximator [Csáji 2001], i.e., it is capable to learn any continuous function on compact subset of $\mathbb{R}^n$ almost surely, the main practical bottleneck is the quality of data. For a given time budget, we assume that it might be more important to spend more time on preprocessing the data rather than on the selecting the algorithm and fine-tuning its hyperparameters.

Therefore, this doctoral project is articulated around two **research questions**:

1. **can we learn a classification model in a space without any metric?**
2. **should we shift our focus from algorithm to data preparation? If yes, how to efficiently automate this preparation phase?**

As a field of applications, we chose the **legal domain**, and in particular, the prediction of judicial decisions. The goal is not to develop solutions specifically and solely for the legal domain but to use this domain to guide our choices by its specific constraints and difficulties. Indeed, the legal environment is a *messy concept* [Rissland 2006] that intrinsically poses a certain number of difficulties to analyze: grey areas of interpretation, many exceptions, non-stationarity, deductive and inductive reasoning, non-classical logic. Statistical models often act as a black-box which is redhibitory for practical applications. In other words, the legal domain combines some of the most challenging elements of today's machine learning. Therefore, by imposing ourselves the constraints of this specific field, we hope to design better machine learning systems.

On top of that, the legal domain is an important part of our society. Helping the justice system to make better decisions could lead to reduce cost and bias and give a better access to a fair justice to every citizen.

1.2 Research Questions Importance

The main obstacle for a wider adoption of machine learning solutions by companies and institutions is the expertise required to obtain satisfying results: many companies have large amounts of data but they lack employees with adequate knowledge of data science and machine learning. In fact, building a high quality machine learning model to be deployed in production is a challenging task that is time consuming and computationally demanding. The usual machine learning workflow, described by Figure 1.1, is broken down into two parts:

1. finding the correct sequence of data transformations such that the dataset is consumable by a machine learning algorithm,
2. selecting a proper machine learning algorithm and its hyperparameters, such that the model provides a good generalization w.r.t. a given performance metric.

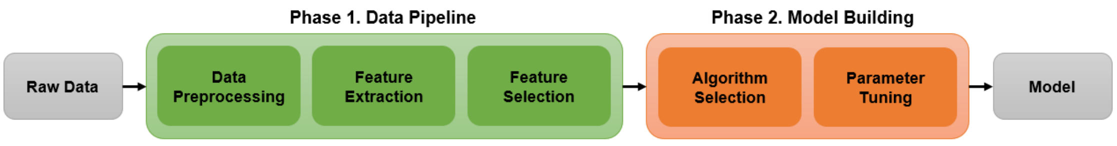


Figure 1.1: Typical machine learning workflow.

Usually, most of computational time and resources are spent on selecting, tuning the algorithm and training the model, while data scientists spend up to 80% of their time on setting up the data pipeline [Chessell 2014, Pa1 2018, Cog 2019]. The state-of-the-art provides data scientists with semi-automated tools to help them to setup a *good* data pipeline. For the model selection phase, meta-optimizers are capable of automatically selecting an algorithm and tuning the model without human intervention but at the price of a large computational time overhead. Notice that these state-of-the-art techniques are far from being widely adopted due to the computational and expertise requirement. In this context, hyperparameter tuning is sometimes not even done [Couronné 2018]!

It is then necessary to develop techniques to lower the expertise and human intervention required to setup a full end-to-end machine learning workflow, without increasing drastically the computational time. Otherwise, the cost of adopting machine learning solutions will remain higher than maintaining less efficient processes already in place. This is particularly true, in domains traditionally far from technical environments such as the legal domain.

We now detail why we chose the specific problem of classification, and in particular, applied to the legal domain. Classification consists in predicting whether a given element belongs to a particular class. It is one of the most common problems in machine learning due to the large amount of situations that can be modeled as such. For instance, classification techniques have been successfully applied to medecine to make diagnostic, finance to assess credit attribution, food industry to classify products by quality. Among the fields of applications, **the legal domain is probably the least researched**, while having a considerable impact on every citizen. On top of that, the industry recently gained interest in providing modern tools for the legal domain. As a results companies specialized in the legal domain, known as

LegalTech, grow at an impressive pace (25% growth a year for a market estimated at over 1 billion USD per year[1]).

In addition, predicting the outcome of legal cases is challenging, even for the best legal experts. Indeed, in [Ruger 2004] the authors report a success rate of prediction not more than 67.4% for the judges and 58% for the global decisions of the Supreme Court of the United States (SCOTUS). We will detail the state of the art on legal prediction in Chapter 2.

To summarize, this doctoral project focuses on creating a **fully automated** (also known as *end-to-end*) **approach to classification** in order to ease the adoption of machine learning solutions. While the domain of application is potentially unlimited, we chose to focus on the **prediction of judicial decisions** for the challenges the field offers and little research interest the field received so far. The main challenge, as mentioned earlier, is to answer the first resarch question: to be fully end-to-end, the machine learning workflow needs to accept any type of data, thus need to work with input spaces that potentially have no metric. Once such workflow is created, we will be able to answer the second question by studying the impact of each phase of the workflow on the final model performances.

1.3 Concept, Workplan and Contributions

To explain the concept and workplan, we detail a bit further the state of the art of the usual machine learning workflow of Figure 1.1. Figure 1.2 illustrates this state of the art in machine learning workflow. The data pipelines is a sequence of operations defined manually by a data scientist, potentially assisted by semi-automated tools. A meta-optimizer then generates several models, possibly through different algorithms, with different hyperparameter configurations, to select the best w.r.t. a certain metric.

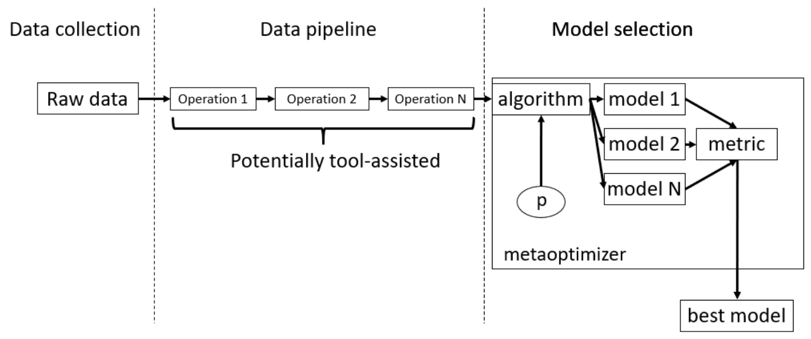


Figure 1.2: The usual machine learning workflow.

In this doctoral project, we proposed some contributions to each step of this machine learning workflow. Each of the contribution can be used independently although they have been conceived as a whole to answer the main research questions of this dissertation.

Data Collection: First, from a practical point of view, we created the European Court of Humain Rights OpenData (ECHR-OD) project to offer the largest database related to the Euro-

[1] https://prismlegal.com/legal-tech-market-sizing-and-opportunities/

pean Court of Humain Rights (ECHR). This database is composed of various types of data that includes natural language judgments, structured and semi-structured data with various representations (tabular, adjacency matrices, tree, etc.) and serves several purposes:

1. attracting Machine Learning practitioners to the field of legal analysis by providing an extended qualitative database suited for many problems,
2. being able to reproduce and extend existing work on the ECHR that have been done on small datasets,
3. evaluating the machine learning workflow proposed in this doctoral thesis on real and large datasets.

This contribution can be found in Chapter 5.

Model Selection: Second, we developed a new machine learning model for classification called Hypergraph Case-Based Reasoning (HCBR), as shown in Figure 1.3. During the development of the model, we kept in mind the following constraints:

- The model must be able to work with any type of data to ease its usage by non-data-scientists end-users.
- For the same reason, the model must be robust to hyperameters and ideally have few of them.
- The model must be able to justify each decision, which is motivated by the fact that, maybe more than any other field, the legal domain requires explainability.

The method itself can be used as any other machine learning method, and is not specifically dedicated to the legal domain. This contribution can be found in Chapter 6.

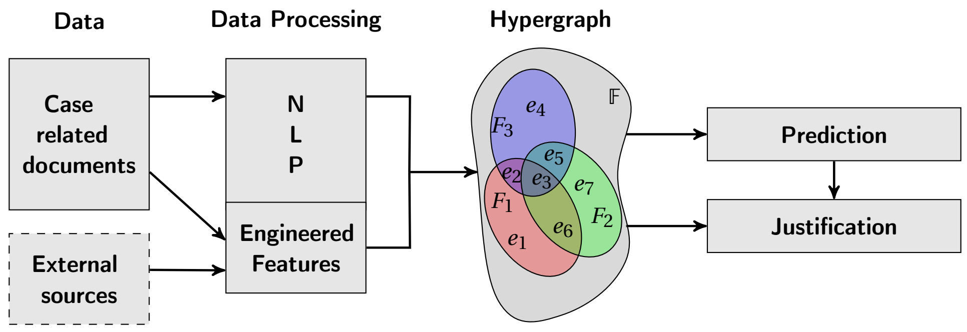


Figure 1.3: High level view of the Hyperbase Case Base Reasoning inserted in a Machine Learning workflow, where data come from multiple sources and is preprocessed.

Data Pipeline: Third, we focused on the automated data pipeline construction. We proposed to use meta-optimizer usually used for hyperparameter tuning and algorithm selection in order to automatically select the proper data transformation operators. This leads to switch from the workflow shown in Figure 1.2 to the two-stage optimization process shown in Figure 1.4. Additionally, we studied the time allocation policy between the data pipeline phase and the model selection phase, to reach the conclusion that spending time on automating the data pipeline is more beneficial than spending time on the algorithm selection and tuning. This contribution can be found in Chapter 8.1.

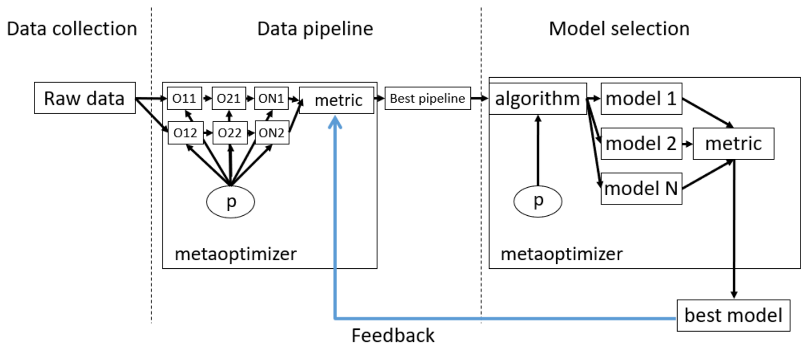


Figure 1.4: The modified machine learning workflow proposed as a solution to the end-to-end classification problem.

A limitation of the workflow presented in Figure 1.4 is that the representation of the data processed by the data pipeline is not know a priori and might evolve during the process. This is a problem because not all machine learning algorithms can handle every type of data. In particular, some algorithms work only with numerical data or continuous values, some cannot work with missing values or are sensitive to outliers.

Therefore, we can combine HCBR to the workflow of Figure 1.4 to obtain the workflow presented in Figure 1.5. The rationale behind the proposed workflow is that if an algorithm is generic enough to work with any type of data and learn fairly complex functions while having few or no hyperparameters, we could reduce the computational time needed during the Model Building phase without increasing the need for human expertise. This time could be spent on the Data Pipelines contruction phase which we showed to be the most important part to focus on. As a result, the human expertise needed to transform data and create a classification model would be minimal while the required computational time would not change or even decrease, resulting in a fully end-to-end machine learning workflow.

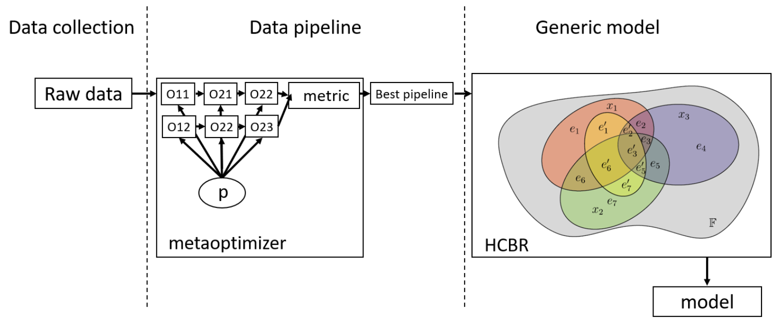


Figure 1.5: The modified machine learning workflow that replaces the usual Model Selection phase by the generic Hypergraph Case-Based Reasoning model.

1.4 Publications

The content of this doctoral thesis is based on the following publications:

- [Quemy 2021] A. Quemy. *A Physical Approach to Classification.* In To be submitted to International Conference on Machine Learning (ICML), 2021
- [Mróz 2020] P. Mróz, A. Quemy, M. Ślażyński, K Kluza and P. Jemioło. *GBEx, towards Graph-Based Explainations.* International Conference Tools with Artificial Intelligence (ICTAI), 2020
- [Quemy 2020b] A. Quemy and R. Wrembel. *ECHR-DB: On Building an Integrated Open Repository of Legal Documents for Machine Learning Applications.* Information Systems (submitted), 2020
- [Quemy 2020c] A. Quemy and R. Wrembel. *On Integrating and Classifying Legal Text Documents.* International Conference on Database and Expert Systems Applications (DEXA), vol. 12391, 2020
- [Quemy 2020a] A. Quemy. *Two-stage optimization for machine learning workflow.* Information Systems, vol. 92, page 101483, 2020
- [Quemy 2019b] A. Quemy. *Data Pipeline Selection and Optimization.* In International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP) @ International Conference on Extending Database Technology/International Conference on Database Theory (EDBT/ICDT) Joint Conference, 2019
- [Quemy 2019a] A. Quemy. *Binary classification in unstructured space with hypergraph case-based reasoning.* Information Systems, vol. 85, pages 92–113, 2019
- [Quemy 2018a] A. Quemy. *AI for the legal domain: an explainability challenge.* In PhD Student Research Competition, IFIP World Computer Congress, 2018
- [Quemy 2018b] A. Quemy. *Binary Classification With Hypergraph Case-Based Reasoning.* In International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP) @ International Conference on Extending Database Technology/International Conference on Database Theory (EDBT/ICDT), 2018

- [Quemy 2017] A. Quemy. *Data Science Techniques for Law and Justice: Current State of Research and Open Problems.* In Advances in Databases and Information Systems (ADBIS) Workshops and Short papers, pages 302–312. Springer, 2017

1.5 Thesis Organization

This thesis is organized in two parts. The **first part** presents the state of the art in the different fields that cover our workplan. In Chapter 2, we explore the different approaches to Computational Law. Chapter 3, is dedicated to the classification problem, as well as the state of the art in metric learning. Finally, in Chapter 4, we present the field of Automated Machine Learning (AutoML).

The **second part** presents our contributions. In Chapter 5, we present ECHR-OD project and a study on the predictibility of the ECHR. In Chapter 6, we introduce a new Case-Based Reasoning (CBR) system named HCBR dedicated to classification in non-metric space. Chapter 7 proposes to extend the discriminative decision function of HCBR to a generative model. This work offers a new binary classification algorithm based on heat diffusion. Finally, in Chapter 8, we study the automated construction of complex data pipelines, as well as the tradeoff between spending time on hyperparameter tuning and on data preprocessing.

Part I

State of the Art

CHAPTER 2

Computational Law

The principle of utility judges any action to be right by the tendency it appears to have to augment or diminish the happiness of the party whose interests are in question...

AN INTRODUCTION TO THE PRINCIPLES OF MORALS AND LEGISLATION, JEREMY BENTHAM

Contents

Law is a set of more or less simple rules or principles, but at the same time most of these rules are ambiguous when it comes to their application. To illustrate this duality, let us take an example from the Cour de Cassation[1] about neighborhood disturbance:

« nul ne doit causer à autrui un trouble anormal du voisinage ou encore excédant les inconvénients normaux du voisinage. »

— **M. TRÉBULLE, 19 NOVEMBRE 1986**[2]

"one cannot cause abnormal neighborhood disturbances or disturbances above the normal drawbacks of living in a neighborhood."

— **M. TRÉBULLE, 19 NOVEMBER 1986**

[1] Highest jurisdiction in France, that can be compared on many aspects to SCOTUS.
[2] Original document: Intervention de M. Trébulle, 2005, Cour de Cassation.

Despite the simplicity of the statement, one may agree that the *normality* it refers to by two times is an ambiguous concept, even once the concept of neighborhood is defined. First of all, how does someone can clearly evaluate the normal drawbacks? It appears that the law text is not self-sufficient to its application. Secondly, it appears that if the neighborhood definition evolves in time with the society, habits of the inhabitants, and many other factors, there is no absolute way of applying the law by itself and thus it arises two concerns:

1. How do simple citizens can be precisely aware that their behavior is against the law?
2. How are we sure the law is correctly applied, i.e. the ambiguity can be fully resolved by the context and the available information?

For many economical agents such as companies, these two questions are crucial and might have a significant impact on their business decisions or daily actions. To answer these questions we might study how the law is actually applied, i.e. what is the nature and qualitative properties of the link between the law text and the persons that apply or make it, namely the judges (and to a certain extent every legal experts).

The current approach consists in manual analysis of situations and decisions by experts hired solely for this purpose: verifying the conpliance of the company processes and business practices with regulations and laws, anticipating complaints or trials from competitors to avoid them, analyzing the past cases to infer by analogy, etc. However, the complexity of our modern societies comes with the price of larger amounts of incompressible procedures and laws to protect the interests of everyone, and on top of that, as the decisions to be taken are more and more complex, the different fields they cover also require a wider expertise. As a result, the risks related to the legal domain dramatically increased the past years, as well as the cost to analyze and reduce these risks.

During the past decades, scientists started to leverage mathematics and artificial intelligence to provide some answers to these concerns which will present in this chapter. In Section 2.1, we will introduce the *realism vs legalism* debate and its different developments during the second part of the 20th century. This will allow us to put the light on some important legal and philosophical elements that must be taken into account to make reasonable choices for our work on predicting justice decisions. In Section 2.2 we will introduce the different approaches to Computational Law. In particular this Section will be broken down into three major parts. While the first one will present some statistical models to predict justice decisions, the second part will focus on specific methods to model preferences and ideologies, with in mind the hypothesis extracted from the previous section about the relation between the judge and the law text (and more generally the act of judging). Last but not least, we will present another branch of Computational Law to complete our landscape: the expert systems with a main focus on rule-based and case-based reasoning systems.

2.1 Elements of Philosophy of Law

To be able to properly address the problems and needs faced by practitioners, we need to understand not only how they work but also how their field evolves and the internal debates. For this reason, we discuss in this Section few elements of the XXth century philosophy of law that inspired most of the practical applications that we present in the rest of this chapter. Although this section may seem to deviate from the research questions formulated in the introduction of this manuscript, we argue that a lot of AI related projects fail because they do not understand the field. Without the goal of being a Subject Matter Expert, understanding the latest developments of our field of application highly decrease the chance to develop solutions that do not have real practical interests.

Hermeneutics, from old greek hermeneutikè – the art of interpretation –, is at the heart of debate between jurists since at least the 11th century. The main debate around interpration consists of knowing if judges interprete the law texts using a rational and objective method (*legalism*) or if the law texts are used to justify their own interpretations (*realism* or *attitudialism*).

During the XXth century, we observe the emergence of two major shifts in the way to handle this debate: the Law and Economics approach and the Hermeutic revolution. The first one totally avoids the debate by stating that the law should strictly be the transcription of economical principles. The second is a more systematic rejection of the postulate according to which the interpretation of the law texts made by judges is the application of a rational and objective method. The legal interpretation is now such a problem that it became in few years one of the main axes of research in legal theory, if not its heart.

2.1.1 Law and Economics

The Law and Economics approach consists in using the tools and criteria coming from the economists to study questions related to the law sphere. Even if the interest for the legal domain by the economist is not recent (for instance, Adam Smith was teaching jurisprudence [Smith 1759] and David Hume was interested in justice and property notions [Hume 1739]), the main rise of the Law and Economics started from the early 1950 before a renewal the last two decade due to the computational power and the availability of data.

Traditional Law and Economics is mostly descriptive rather than normative, that is, interested in describing the justice system at the light of economic tools. One we can mention the work of George Stigler on the analysis of regulations and the impact of lobbies in [Stigler 1971] or those of James Buchanan on the political decision process [Stigler 1974].

On the contrary, modern Law and Economics is rather normative, that is, tries to shape the justice system based on economic tools and principles. The common ideas and fundamentals are probably established by Posner in [Posner 2011] initially published in 1972. For Posner the law must be strictly considered as an incitation mechanism to modify human behavior to tend to the economic efficiency. According to Posner, the Common Law system, and more generally the jurisprudence system, is a tool to maximize wealth in the society. Coarse shown in the 60s [Coase 1960] that some jurisprudence decisions were made in conformity with the optimal solution to an associated optimization problem. For Calabresi, "the principal function of accident law is to reduce the sum of the cost of accident and the cost of avoiding accidents" [Calabresi 1970].

This new approach can be illustrated by the work of Deffains and Fluet [Deffains 2009], among others, on the interaction between legislative incitations due to sanctions and the normative incitations produced by social disapproval. They showed that the social disapproval of a verdict is function of the respect of the law and that the behavior of an agent is more likely to be influenced by social disapproval than by the respect of the law itself. Samuel Ferey enlightened the civil liability problem by leveraging cooperative game theory to provide hints on how to determine the compensation between parties in case of complex situations (e.g. corporal damaged caused by a driver, worsen by a medical error) where traditional law arguments failed to tackle the issue [Dehez 2012]. Régis Blazy studied the legal mechanisms to handle excessive debt cases from households through many past decisions in order to extract objective and quantitative criteria to resolve the ambiguity of the expression "irremediably compromised situation" that appears in the legal procedure [Blazy 2013b, Blazy 2013a, Blazy 2011, Blazy 2012]. From a more behavioral approach, Mathieu Lefebvre studied the attitude toward fiscal fraud and social fraud among the population of different countries to determine the non-economical factors that model this attitude. This is helpful if not necessary in order to conceive efficient public policies and also, before, to decide how to allocate public resources to fight those problems.

2.1.2 The Hermeneutic Revolution

The hermeneutic revolution is a more and more systematic and radical rejection of the "formalist myth" that consider that judges applies a formal and objective method [Kelsen 1967, Hart 2012, Ross 2004].

For Frydman the interpretation is an intellectual operation to determine the sense of a text in order to delimit the extent of the rule in the given context of its application. In [Frydman 2005], he describes a (near) consensual approach of judging as the choice of the best interpretation among the set of projections of the law on a given cases, according to some criteria.

For Troper, the interpretation is iterative because the law or the Constitution itself are often formulated in generic terms, and in the process of interpretation lie some discretionary elements [Troper 2001].

Another intrinsic problem of the act of interpretation is the innovation or disruption. As mentioned by Sunstein in [Sunstein 1998], even if law is written in such a way it can be applied in practical situations, law makers cannot predict the future and *de facto* new situations will emerge, as the result of technological evolutions, creating by then grey zones of interpretation than be either be filled by a new law or by jurisprudence, and thus the act of interpretation.

This problematic takes dramatic proportions in the United States for example because of the institutional combination between the Common Law and the Supreme Court that can create polemic. Indeed in the United States a Justice is appointed until his death and new ones are chosen by the President, turning the selection into a long term political strategy. Rosenfeld goes even further by stating that the act of interpretation by a Judge is always suspicious because it is always an act of pure will of its interprete [Rosenfeld 1998]. In the Section 2.2 we will present some statistical studies that try to quantifiy the subjective part in the act of judging.

2.2 The different approaches of Computational Law

This Section is an extension of [Quemy 2017] and [Quemy 2018a]. We can distinguish two categories among the approaches, namely rule-based and data-driven methods as illustrated in Figure 2.1. While the former encodes legal knowledge to automatically infer and create reasonings, the second relies only on data, non-necesarily purely legal. Historically, the first methods were preminent until the begining of this century but the progress of Machine Learning (ML) and the availabity of data inverted the trend the last years.

In general, there are two major objectives to Computational Law: prediction and explanation. In the first case, the model is used to predict the outcome of a decision, while in the second case, research focuses on building an argumentation based on legal reasonings (to take a decision or to assist or replace human in some tasks).

A major classification factor appears to be the nature and scope of the information. Some methods rely on legal knowledge and reasoning techniques while others exploit the available data: past decisions records, non-legal features, etc. The border is far from being clearly drawn since to apply legal reasonings and this taxonomy should not be taken as strict. However, it seems important to notice that some models do not exploit legal knowledge at all and solely rely on statistical evidences to predict. A model that does not exploit legal knowledge and relies only on statistical evidences cannot produce an explanation to the forecasts it produces. This is the reason why a predictive model with explanation must integrate legal knowledge at some point. However, to be broadly adopted by legal experts and institutions, a model needs to provide a explanation, and if possible an explicit model of decisions, and thus must take into account legal knowledge. This is a challenge in particular for pure ML techniques that are well-known to propose non-analytical and hard-to-interpret models, especially the most powerful techniques such as the Random Forests or DL.

It is important to notice that the literature mostly focused on SCOTUS. For this reason, unless stated otherwise, we assume that the studied Court is SCOTUS.

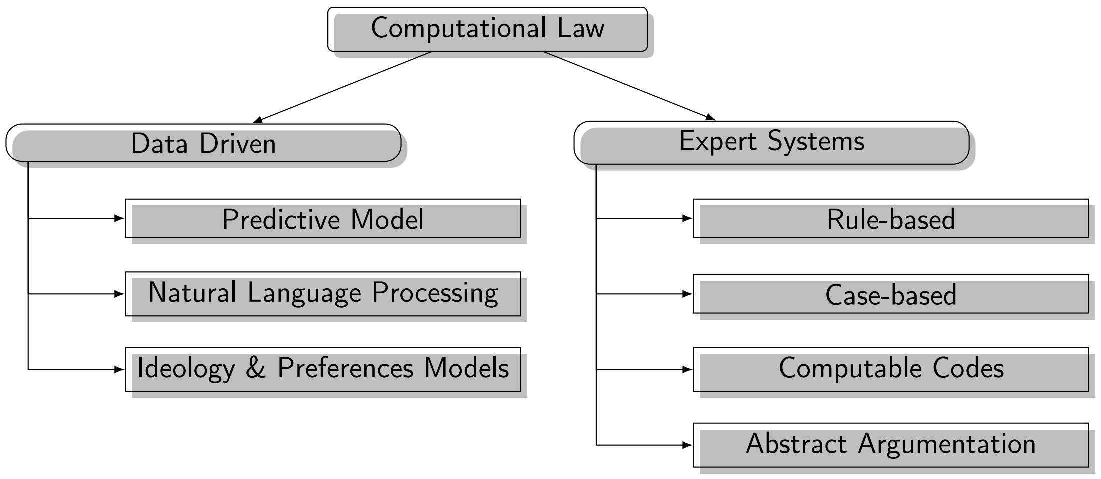


Figure 2.1: Taxonomy of Computational Law.

In this section, we review some of the majors methods of Computational Law and Legal Analysis. We excluded from our review Rule-Based systems since Kowalski showed that CBR systems are more efficient and reliable than rule-based systems when it comes to the legal domain [Kowalski 1991]. We also excluded Computable Codes that focuses on computational models for laws, as well as domain-specific applications since it is either out of the scope of this thesis, or not applicable to our research questions.

2.2.1 Predictive Models

One axiom (often implicit) of all quantitative approaches is based on the concept of **omniscient court** where the judges are perfectly rational, free of all bias and preferences, and have an unlimited access to knowledge[3]. The axiom states that in this ideal configuration all judges must reach the same decisions, and two exactly similar cases would result in the same decision.

The assumption in ideal court is thus that two judges that agree (or disagree) do not carry information about their potential agreement in the future. In other words, the decisions are not correlated to past cases. As a result, it is impossible to predict a judgement from an ideal court using the information from the past cases. On the contrary, statistical methods try to detect hidden patterns between the sequence of decisions and therefore, the less a court is predictible, the closer it is from an ideal court.

On the quality of predictive models In [Katz 2017b], Katz and al. formulate three criteria for the quality of a predictive model in legal domain:

- **General**: A good model needs to be stable over a long period of time and not only for few years. The legal environment can drastically change over time, including invalidating some previous laws.
- **Robustness**: A predictive model should not be sensitive to dogma, political or doctrinal shifts, macro-economic dynamic or economic and political environment.
- **Fully predictive**: The lowest acceptable level of prediction granularity is the case final decision level. To illustrate, let us consider a case starting at time t which the final decision is available at time $t+1$. A model is fully predictive if it does not require partial information available at time $t+1$ to predict the final decision.

Notice that the general characteristic of a model is strongly correlated to its robustness since the stability over years is reduced in probability.

The Block Model Using social network and affiliation network techniques [Guimerà 2011] proposes to predict the vote of a judge given the votes of the other judges composing the U.S. Supreme Court and the history of decisions. By achieving a better prediction score than the legal experts (77% versus 75.4% in [Ruger 2004]), without using legal based arguments or

[3] In general, the literature constrains the knowledge to the information about the case, without to specify if it includes or not the information about past cases also. This is an important point because in legal system where jurisprudence plays even a small role, the temporal order and comparison with past cases is important to take a decision in the present.

case-based information but hidden associations between social actors, the authors clearly exhibit empirical proofs to support the attitudinalism paradigm[4].

Unfortunately, this approach suffers from two drawbacks. Firstly, it is not a fully predictive model as it can only predict one judge's decision per case. Secondly, this model cannot be used to build a legal explanation that can be associated to the prediction. Indeed, even in a very pessimistic view where the decisions would be taken solely on the will and cognitive bias of the judges, these judges have to provide a decision motivated by legal arguments.

The Decision Tree Approach In [Martin 2004a, Ruger 2004], several hundred of cases were described using 6 visible[5] features such as the lower court circuit or the type of petitionner. After training a classification tree per judge appointed to the Court, the authors generated a forecast for 68 new cases and compared the results to the predictions made by a set of 83 legal experts.

They found out that on case outcome, the statistical model performs significantly better than the experts, while the difference of prediction at the judge level is not significant. However, using a majority rule to predict the global outcome and not individual votes, based on the expert forecast on judges decisions, the model performs better. An important result is also that the experts where better at drawing good conclusions on the vote of the most extremely ideologically oriented judges[6].

The authors exhibit that the model takes into account all the hundred of cases. On the contrary, the experts uses only few important and representative cases. As the experts were not constrained to purely legal factors, the comparison between the experts and model predictions is not about behavior versus legal arguments but rather about the capacity to exhibit patterns between cases and combine the most useful information.

"Although lawyers can and do make predictions about both outcomes and reasoning, the model is incapable of generating predictions about the content of the Court's opinions. By design, the statistical model is blind to legal doctrine in its inputs-and thus, it is correspondingly mute as to doctrine in its predictive outputs."

— **RUGER AND AL.**[7]

The Extra-Tree Approach In [Katz 2017b] improved the previous Decision Tree approach using Extra-Tree. The general, robust and fully-predictive model proposed in [Katz 2017b] successfully identified 69.7% of the Court decision over 60 years, using the data available prior to a date of a case. This is slightly less than the experts and then less than with the Block

[4]On top of the predictions, they show the existence of different predictability between judges, implying a difference of attitude toward the law, as well as a decrease in the U.S. Supreme Court predictability during some periods or depending on the political party at the presidence.

[5]i.e. also available to legal experts

[6]Where the measure is calculated as given by Martin and Quinn, explained later in this Chapter

[7]In [Ruger 2004].

Model [Guimerà 2011], but the characteristic of this method provide a better starting point to build the foundation of quantitative legal prediction.

But once again, the drawback of such an approach is the lack of nuances in the possible judgements since the decisions are binary only[8] and do not construct a legal justification.

The model is build up on more than 300 features that are believed to have an impact on the decisions, divided into 3 categories:

- Court and Justice level information.
- Case information.
- Historical Justice and Court information.

The last category is composed of unidimensional aggregated statistics rather than the whole history of past case description using the two previous categories of features. The ideological shift is expressed as a mean and standard deviation of these indicators.

Using an Extra-Tree algorithm [Geurts 2006], the authors store a time-ordered subset of the features for all cases before the one to predict at time t. They trained an extremely randomized tree on this cases history, and generated predictions using only information available at time t. The classifier provides a prediction for each judge and the overall decision is deduced from a majority rule. They trained their algorithm on cases from 1946 to 1953 and them make predictions from 1953 to 2013.

An interesting characteristic of the general approach of [Katz 2017b] is that the classifier returns the weights aka the importance of every feature in the prediction. Of course, those weights evolves in time giving additional comprehensive hints on the Court dynamic.

Despite the enormous advantage of having the weights compared to other models, the interpretation is complex due to the correlation between the features [Tolosi 2011]. A very interesting characteristic is the predictive power of the different category of features. The case features account for 23% while the Court background only for less than 5%. In other words, most of the predictive power holds in the behavioral trend including ideological shifts. More than the exact percentage for this particular model and Court, this tends once again to reinforce the realism hypothesis.

Last, a major drawback is the computational time. Indeed, they had to retrain the model from scratch after some period of time.

2.2.2 Natural Language Processing

Few attempts to leverage Natural Language Processing (NLP) techniques have been recently proposed. In [Aletras 2016] the authors achieve 79% accuracy to predict the decisions of the ECHR. They make the hypothesis that the textual content of the European Convention of the

[8]Despite it suits the Supreme Court decisions, we can argue that there are various degrees in the way a case is affirm or reverse and using this information may be useful for the prediction in a non-ideal court setting. On top of that, in lower court, a decision is along with other features such as fine, jail duration or other compensating measures.

Human Rights and the case elements holds hints that will influence the decision of the Judge. They model the decisions as binary classification rules and train a Support Vector Machine with N-grams and topics extracted from the past cased labeled with the outcome of the case.

The cases of the ECHR are particularly structured: procedure, non-legal facts, relevant law (other than the Convention articles), summary of submissions including the alleged violation of some articles of the Convention, and of course, the final verdict.

The authors focused only on three articles of the Convention[9] for a total of 584 only cases. Using a Bag-of-Words (BoW) representation they computed the top 2000 N-grams for $N \in 1,2,3,4$ per case. Given the matrix of features representing the N-grams of all the cases, they extracted the features related to some groups (Procedure, Circumstances, Facts, Relevant Law, Law and Full). After partitionning each article into topics, they compute a N-gram Similarity Matrix with a cosine metric and finally apply a Spectral Clustering on the N-gram Similarity Matrix. The classification is done using a SVM with a linear kernel which allows the identification of weights for each feature.

Additionally, in [Medvedeva 2020] a SVM is also to reach an overall of 75% accuracy on judgment documents up to September 2017. The protocole is roughly similar to the one in [Aletras 2016] except that it uses few thousands of cases rather than few hundreds. Finally, in [Chalkidis 2019], the authors used DL to classify about 11 thousands judgment texts and reached 80% F1-score.

In the discussion section of [Aletras 2016], the authors state in favor of the realism hypothesis because the case contains non-neutral statement of facts. They also invite to take into account the *selection effect* [Klerman 2012], i.e. if a case is accepted in appellate court, especially international ones, it means that the normal legal framework was proven not to be determinate enough. Due to this effect, the authors sauggest to investigate lower courts, in particular domestic ones.

2.2.3 Ideology & Preferences Models

As enlighted by [Guimerà 2011, Katz 2017b], one preminent factor in the predictive power of statistical methods is the ideology and preferences of the judge or justice. In order to capture as much as possible this reality, several indicators and models to describe the preferences and ideology have been developed since 1950 [Tanenhaus 1967].

A primary taxonomy of methods to model preferences and ideologies can be established according to the type of information they rely on. Three types of methods emerge: those relying on the party affiliation [Spitzer 1994], on experts judgements [Segal 1989, Segal 1995] and finally, on votes [Quinn 2002, Quinn 2006], possibly including decision-related material [Lauderdale 2014, Islam 2016, Sim 2014]. In general, all studies focus on two objects: the ideal points and the median voter.

Ideal Point The Ideal point is a latent variable to position ideology or preferences of a decision maker in a continuous space, often 1 dimensional.

[9]Article 3. Prohibits torture and inhuman and degrading treatment.
Article 6. Protect the right of a fair trial.
Article 8. Provides the right for one's private and family life, his home and his correspondance

A discussion about the dimension of the ideal point space can be found in [Lauderdale 2014]. Most studies model the Ideal point in one dimension, some in two, but quantiatives studies showed that preferences are highly multidimensional. Estimating the right value for the dimension is a challenge, but on top of that, higher dimensional states are challenging by itself since it is often harder to interpret the dimensions and provide useful insight about the political cleavage as firstly intented by the study.

The median voter theorem and the median Justice A majority rule voting system select alternatives which has a majority. The rule is not used only for election in democratic countries but also in many Courts, such as the Cour d'Assises in France for crimes, the SCOTUS in the USA, or the ECHR in the European Union. The median voter theorem states that such system will select the outcome most preferred by the median voter. For this reason, it is often enough to study the preferences of the median voter.

2.2.3.1 Segal-Cover Score

The Segal-Cover Score [Segal 1989, Segal 1995] is constructed out of editor's assessments published before the nomination of a given judge. Extracted from four newspapers including two with liberal stance (Washington Post, New York Times) and two with conservative views (Los Angeles Times and Chicago Tribune), the papers enable the authors to give a note from -1 (unanimously conservative) to 1 (unnimously liberal) to a judge. To do so, the authors established a coding scheme where for instance the identification of a Justice to a conservative or the Republican party account as an occurrence for the conservative side while a support for the New Deal economic policies as liberal support. One might notice that since it is very likely that the party affiliation will be mentioned in at least one text this method encompass the methods using solely the affiliation to give a score.

One main advantage of the Segal-Cover score is the independence of the sources used from the votes preventing from circular reasoning to validate predictive model using it.

To test the predictive power of their criterion they created dependent variables by manually labeling some cases on the same unidimensional scale as previously (one variable for civil liberties cases and another for economics cases). The coding was arbitrarely made such that liberal points are counted as a decision in favor of the ciminally accused, civil liberties claimant, indigents, Native Americans, against the government on private property litigation, etc. for the civil liberties and as a decision pro-union against the government or individuals, pro-government against challenges to its regulatory authority, pro-competition, antibusiness, etc. for economics.

From a simple linear regression $\hat{Y} = aX + b$ between the dependent variables Y and the score X, they showed several results. First of all, the coefficient of determination is higher that one might expect (0.69 and 0.56 respectively for the Civil Liberties and Economics cases for the terms for the appointees from Roosevelt to Bush). Second of all, signifiant differences exist between the area of law: as mentioned Civil Liberties seems to be more prone to Judges preferences and ideal point than Economics cases. Last but not least, the predictibility strongly depends on the court composition: the court under the Roosevelt-Truman era is far less correlated than the one under Eisenhover-Bush. Notice that this is not necessarily in contradiction with the observations made in [Guimerà 2011] statuing a decrease in predictability of the court for at least three reasons. First of all, while the former article treats specifics area of Law, the latter uses the first 150 for each court composition. The period studied is also not the same: starting from 1933 with Roosevelt to end with Bush in 1993 for the first article, from the first Warren court in 1953 (appointed by Eisenhower) until 2004. Finally, if the trend of predictability from 1953 to 2004 goes down, a regression on the restricted data from 1953 to 1993 (the intersection of both time ranges) might not give the same result as the clear fall of predictability occurs from 1990.

Nonetheless, despite a clear match between the score and the reality, the Segal-Score method suffers from numerous drawbacks. A lot of manual work has to be done from labeling the cases to extract the information from the editor's assessments. On top of that, some concepts are not formally or universally defined such as the coding of what accounts for liberal or conservative. Not accounting for measure mistake in the ideal point estimation also results in incorrect standard error for the linear model employed further.

To overcome these drawbacks, the use of modern NLP would allow us to automatically extract the information and keep it updated to refine the estimation of the model parameters over time.

2.2.3.2 Martin-Quinn Score

The Martin-Quinn Score [Quinn 2002, Quinn 2006] is one of the most accepted and discussed measure of Ideal Point of the SCOTUS. The ideal point is modeled as a unidimensional indicator, retrieved using the votes records. One specifity of Martin-Quinn approach is the hypothesis that ideal points evolve in time. This said, the model makes the assumption by design that there are only two alternatives only which is reasonable for apellate court but not suitable for lower court or purposes. Nevertheless, we made the choice to present this model as most models are can be viewed as an extensions.

The spatial model The matrice of the observations representing the votes for K temporally ordered cases judged by J judges is denoted by $V = \{0, 1, \emptyset\}^{K,J}$.

As all the judges did not participate to vote for every cases we introduce the following notations:

- $K_t \subset \{1, ..., K\}$, the set of cases decided à time t with $|K|$ its cardinality.
- $J_k \subset \{1, ..., J\}$, the set of judges voting for case k with $|J_k|$ its cardinality.
- When the data is not missing, $v_{t,k,j}$ is the value of the vote at time t for the case k by judge j coded as 1 if the vote reverses the jugement of the lower court.

Each judge is assigned two utility functions, $u^{(a)}_{t,k,j}$ and $u^{(r)}_{t,k,j}$ respectively the utility to affirm the case or to reverse it. The vote reverses the case if $z_{t,k,j} = u^{(r)}_{t,k,j} - u^{(a)}_{t,k,j} > 0$.

Dynamic Item Response Model Using a spatial voting model, $z_{t,k,j}$ can be expressed as follow:

$$z_{t,k,j} = -|\theta_{t,j} - x_k^r|^2 + \varepsilon_{t,k,j}^r + |\theta_{t,j} - x_k^a|^2 - \varepsilon_{t,k,j}^a$$

with

- $\theta_{t,j}$ the ideal point of the judge j at time t.
- x_k^r, x_k^a the location of the revert and support policy.
- $\varepsilon_{t,k,j}^r, \varepsilon_{t,k,j}^a$ a gaussian noise, centered and with a fixed variance.

The voting model is isomorphic to a 2-parameters Item Response Model [Clinton 2004] such that $z_{t,k,j} = \alpha_k + \beta_k \theta_{t,j} + \varepsilon_{t,k,j}$ where the parameters $\alpha_k = [x_k^{a2} - x_k^{r2}]$ and $\beta_k = 2[x_k^a - x_k^r]$ by identification. In the original terminology, α_k and β_k are called *difficulty* and *discrimination* but in this context they are called *popularity* (of the alternative) and *polarity* which ease their interpretation. Indeed, popularity describes how appreciated or natural regarding the context an alternative is, while polarity describes how much this popularity is disturbed by the ideal point and in which direction (in favor of affirmation or reversal).

A posterior estimation is used to estimate the parameters. α (resp. β) denotes the vectors $(\alpha_i)_{1 \le i \le K}$ (resp. $(\beta_i)_{1 \le i \le K}$) and θ the matrix of ideal point for each judge j at each time t. Using Bayes' formula, the joint distribution probability of parameters is expressed by:

$$p(\alpha, \beta, \theta \mid V) \propto p(V \mid \alpha, \beta, \theta) p(\alpha, \beta, \theta)$$

with $p(\alpha, \beta, \theta)$ a prior and $p(V \mid \alpha, \beta, \theta)$ expressed in function of the gaussian cumulative distribution function ϕ by:

$$p(V \mid \alpha, \beta, \theta) \propto \prod_{t=1}^{T} \prod_{k \in K_t} \prod_{j \in J_k} \Phi(\alpha_k + \beta_k \theta_{t,j})^{v_{t,k,j}} (1 - \Phi(\alpha_k + \beta_k \theta_{t,j}))^{1 - v_{t,k,j}}$$

The author used a standard approach to define the prior by using a multivariate gaussian distribution:

$$\begin{bmatrix} \alpha_k \\ \beta_k \end{bmatrix} \sim \mathcal{N}(b_0, B_0), \ \forall k \in \{1, ..., K\}$$

For the ideal point however, they model it as a random walk:

$$\theta_{t,j} \sim \mathcal{N}(\theta_{t-1}, \Delta_{\theta_{t,j}}), \ \forall t \in \{\underline{T}_j, ..., \overline{T}_j\}$$

where $\underline{T}_j$ (resp. $\overline{T}_j$) are the first (resp. last) decision of judge j and $\Delta_{\theta_{t,j}}$ a constant variance fixed *à priori*.

Interestingly, $\Delta_{\theta_{t,j}} = 0$ corresponds to a static ideal point over time, i.e. the judges never change their preferences over time, while with $\Delta_{\theta_{t,j}} \to \infty$ the preferences are independent between each decision, which is not realistic.

Finally, the author used a Metropolis-Hasting algorithm with data augmentation to sample the matrix of latent utility differences Z, a Dynamic Linear Models for the parameters sample and a conjunction of foreward(-filtering) backward(-sampling) algorithm to simulate $(\theta_{t,j})_{\underline{T} \le t \le \overline{T}_j}$. For more details on the sampling procedure, we refer the reader to the orignal article.

After computational expensive calculations, the authors exhibit results coherent with previous models, in particular with the Segal-Cover, and an outstanding correlation factor for both Civil Liberties and Civil Rights cases. Despite a sligthly lower correlation for the Economics cases, the average correlation over time around 0.8 is sufficient to dismise the hypothesis drawn from [Segal 1989] according to which the preferences of Justice are less preponderant in the economic area or should be captured by a higher dimension space.

In [Andrew 2004], the authors give a computational approach to determine the median Justice of each term based on their model which is helpful for cheaper predictions.

2.2.3.3 Random Utility Model

The authors of [Sim 2014] extended the Martin-Quinn model to take into account the amicus briefs[10]. To do so, they considered that authoring a text is equivalent to maximize a certain utility.

Amici as agent and Random Utility Model As the amici briefs represent an attempt to shift the position of the case and thus considered to belong to one of the two classes representing the parties. They rewrote the votes probability distribution as follows:

$$p(v_{i,j}|\theta_i,\phi_i,\Delta_i,\alpha_i,\beta_i,\gamma_i) = \sigma(\alpha_i + \theta_i^T(\beta_i\phi_i + \gamma_i^a\Delta_i^a + \gamma_i^r\Delta_i^r))$$

with

- ϕ_i the legal arguments in merits briefs
- θ_i justice ideal point.
- Δ_i^a, Δ_i^r the mean issue proportions of the amicus briefs supporting or not the revertal
- $\sigma(x) = \frac{e^x}{1+e^x}$, the logistic function

For the sake of readability, we denote $\kappa = (\alpha_i, \beta_i, c_i^a, c_j^r)$.

The amici are viewed as an agent with an utility function for a given case k: $u((v_{k,j})_{j\in J_k}) = \sum_{j\in J} \mathbb{1}_{(v_{k,j}=s)}(j)$ with s the side of the agent.

To model the cost of writing, a regularization term is defined such that $C(\Delta,\phi) = \frac{\epsilon}{2}||\Delta - \theta||_2^2$ with ϵ an hyperparameter to control the importance of the cost. Under the assumption that an agent ignores any other agent[11] the expected utility for an amicus on side s can be expressed as an optimization problem:

$$\max_{\Delta} \mathbb{E}_\Delta[u((v_{k,j})_{j\in J_k})] - \frac{\epsilon}{2}||\Delta-\theta||_2^2 = \max_{\Delta} \sum_{j\in J} \sigma(\alpha + \theta_j^T(\beta\phi + \gamma^s\Delta)) - \frac{\epsilon}{2}||\Delta-\theta||_2^2$$

They introduced a prior on Δ coming from the Random Utility Model [McFadden 1973]: $p_{\text{util}}(\Delta) \propto \mathbb{E}_\Delta[u((v_{k,j})_{j\in J_k})] + \epsilon(1 - \frac{1}{2}||\Delta-\theta||_2^2)$

[10]Amicus brief are texts written by the *amici curiae* (lit. friend of the court) to influence the Justice to a side or another. The amici curiae are not directly connected to the given case they want to influence.

[11]The authors justify this assumption to avoid to to deal with complex game-theory models.

Finally they express the quantity to estimate, taking into account all the amici:

$$\mathcal{L}(w, v, \theta_i, \phi_i, \Delta_i, \alpha_i, \beta_i, \gamma_i)[\prod_{k\in\mathcal{A}} p_{\text{util}}(\Delta_k)]^{\eta}$$

where $\mathcal{L}(.)$ is the likelihood function associated to the votes probability distribution and η a hyperparameter[12].

Learning and Infering As it is not easy to automatically recognize the side of an amicus brief, the author labeled some to train a binary classifier to automate the process. For the different parameters they use a gaussian prior.

To infer the topics mixture for ϕ_i and Δ_k an Latent Dirichlet Allocation (LDA) is used where the topic-word distribution is build over a huge document obtained as the concatenation of all amicus briefs.

For the parameters estimation, they proceed in two steps:

1. Infer ϕ_i and Δ using LDA.
2. Fixe ϕ and Δ to their posterior mean and then solve for θ and κ using an hybrid Markov Chain Monte Carlo algorithm.

The data is comming from LexisNexis to cover 2074 cases over 23 terms (1990-2012).

Experiments and results As for the Martin-Quinn score, the vote prediction is issued from a bayesian approach with posteriori estimation. The non-dynamic aspect makes easier the final probability distribution but the integration of external documents offer several interesting features. It is possible to decompose the estimation into the different influences (see Figure 2.2) and perform counterfactual analysis by removing some amici to simulate what whould have been the vote result and the outcome of a case (see Figure 2.3).

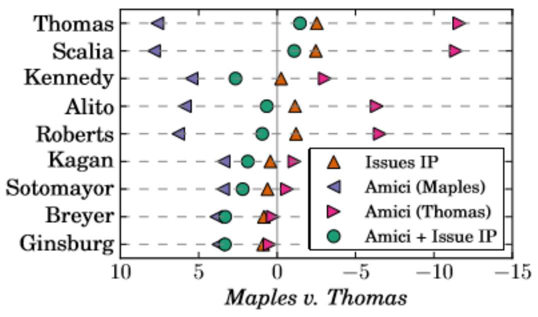


Figure 2.2: Decomposition of a decision per judge and factors (ideal point, amici for both sides, and combined).

In this specific case, the briefs shift the ideal point toward Maple side and for three Justices, enough to change the initial side. The scale represent a log-odd of vote.

[12]For a justification, refer to [Hinton 2002] on the product of Expert Models

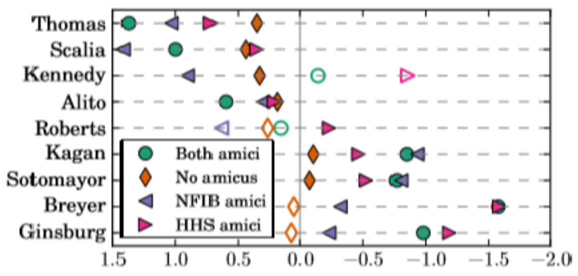


Figure 2.3: Counterfactual analysis answering the question "What whould have been the probabilities of vote if one or both amicus were not filled?".
The scale represent a log-odd of vote.

However, the approach suffers from some drawback and offer rooms of improvement. Of course, integrating the dynamic from Martin-Quinn would represent a first step. Also, the cost function is rather simple and free from legal intricacies. Finally, with the recent progresses of DL techniques, it is possible to automatically infer the side of the amicus such that, a much larger dataset could be easily obtained.

A quite similar approach, based on Item Response Model combined to an LDA used on the court opinions instead of the amicus brief can be found in [Lauderdale 2014]. The major difference is that the LDA is used to determine the the ideal point in a higher dimensional space contrarely to [Sim 2014].

2.2.3.4 A Natural Language Process approach

The approach taken in [Islam 2016] is to consider that the court opinions can be expressed in the same space as the Justices. Both the Justice and the opinion has an ideal point (notice that an opinion is authored by several Justices and there exist several type of court opinion depending on the agreement or disagreement with the final vote). While in [Sim 2014], a LDA was used to infer the influence power of an amicus brief, the ideal points were still modeled as a predefined unidimensional space. In comparison, the approach proposed here consists in building a K dimensional space were the dimensions are learned using NLP techniques.

Each Justice has a topic distribution and for a given opinion, a topic mixture is deduced from its authors. Those topic distributions are created using an LDA. For a given distribution the ideal points of the Justice and opinions are estimated using a gradient descent. Conversely, the topic mixture for an opinion is deduced from the Justice ideal point. The author justified this approach by the fact that the opinion is written after the case is argued and then, after the ideal points has been shifted. The topic distribution of a Justice is also deduced by its ideal point. The global algorithm turns to be a two-step iterative process: (a) optimizing the ideal points estimation using a gradient descent and (b) learning topic opinion using a Gibbs sampling coupled to a LDA. We refer the reader to [Islam 2016] and the Supplementary Material [13] provided by the authors for more details about the method, the dataset and results.

[13]https://dl.dropboxusercontent.com/ u/8921131/scotus/scipm-suppl.pdf

The authors compared the model prediction to the other predictive models including [Lauderdale 2014] on a dataset from 2010 to 2014 removing unanimous decisions for a total of 185 cases with 467 opinions. The average recall value obtain is 79.46% which outperformed the 48.13% obtained using the method described by [Lauderdale 2014].

The ideal points live in a 10 dimensional space were each dimension has to be manually labeled depending on the top word. If labeling might be sometimes difficult, as each opinion can be located in the same space as the Justices, there is no problem to interprete the distance from an opinion to the Justice. However, this is a problem when it comes to predict a case currently judges and which there is no opinion. In particular, the model does not use external information about a given case. However, it allows to create a better *profile* of Justices depending on several area of law to cover the drawbacks of Segal-Cover or Martin-Quinn score and would gain to be integrated in a more general model as described in [Katz 2017b].

Last but not least, the author defined the median Justice as the Justice with the highest average deviance of ideal points over the issues. The results are similar to Segal-Cover and Martin-Quinn's ones but should be confirmed on a larger time range.

2.2.4 Case-Based Reasoning

Case-Based reasoning postulates that humans reason by features analogy between past cases or situations.

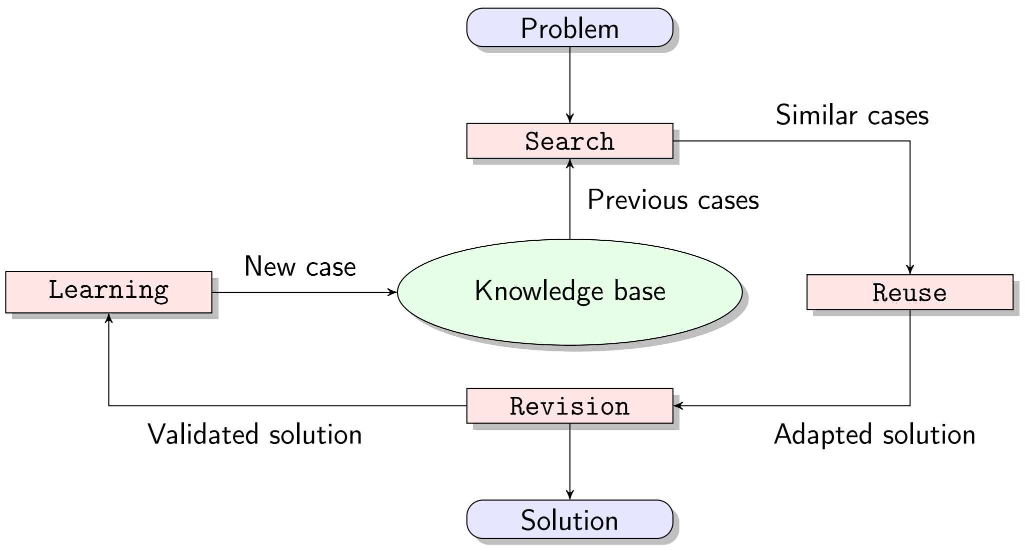


Figure 2.4: Illustration of the cycle carried out by a Case-based system to solve a problem as illustrated by [Aamodt 1994].

The solve a given problem, the case-based system performs the following cycle illustrated by Figure 2.4:

1. Search for the most related past cases either by filtering the irrelevant cases or selecting the closest ones depending on a metric and a K-Nearest Neighbors (KNN) algorithm.

2. Reuse of case with adaptation to produce a new solution that fit to the case. The two main methods to do so are the *transformational adaptation* [Kolodner 1993] where the solution of a past case is taken as it is and modified to be used for the new case, and the *derivative adaptation* [Fuchs 2014] where the solution is built from scratch by applying the same reasoning or inference process used by the previous cases.
3. Evaluate and revise the proposed solution, including at least why the solution is not satisfying.
4. Integrate the solution to the database.

One of the main challenging part to conceive a CBR system is the representation and structure of cases. In the best case, a case should be structured including a description of the problem, the steps to find the solution, the result of the evaluation step and the reason of the failures or drawbacks of the adopted solution. More than the difficulty to collect this knowledge, there is a tradeoff in the level of abstraction that is hard to reach: too abstract will result in poor analogy (the cycle search, adapt, evaluate, integrate), and too concrete will make it hard to build an anology or create analogy based on anecdotical evidence. With the growing popularity of CBR, statistical frameworks have been developped to provide confidence level on the prediction [Hüllermeier 2007]. Another possibility to improve the adaptation and evaluation is to rely on an iterative user feedback to learn an underlying utility function as described in [Akrour 2014].

Originally the cases were compared by relying purely on the knowledge encoding of cases using a weighting feature model assisted by ML [Stahl 2003, Gabel 2004], in a similar fashion as in [Katz 2017b]. However, the search step tends to be merged into others steps to rely on other criteria than solely the case features. For instance, in [Smyth 1998] the closed cases are picked depending on how easy the solutions can be adopted to solve the problem. For a general review on CBR systems, we refer the reader to [De Mantaras 2005, Jian 2015].

Finally, one common pitfall of CBR systems is the impact of the database growth on retrieval cost aka the *case-based maintenance*. Minton shown that the rules learned to filter and select carefully the past cases might degrade the overall performances on the long term [Minton 1990].

2.2.4.1 Case-based Reasoning and Legal Reasoning

The utility of CBR systems applied to legal reasoning is bounded by the following assumption: legal practitioners reason by analogy, i.e. evaluate arguments related to a situation depending on similar situations encountered in the past. While this question is still ardently discussed within the law community (for instance the discussion in [Weinreb 2005, Posner 2005, Kaye 2005] or the more general work on the subject [Becker 1973]) in this thesis we do stand for the analogy importance but with more pragmatic reasons:

1. In practice legal analogies are use, and broadly speaking, an rule is made to be applied when a situation meets some requirements. As a law will never fully describe a situation, a law applies to cases that share this set of requirements as minimal common denominator that we can call grounded analogy. There, to model the decisions we need to understand and take into account the underlying process of taking a decision and

build a legal reasonning, independently of the potential flaws regarding the purpose of the legal environment.

2. We do believe that even if legal practioners do not use analogy or change not to use them in the future, it can be considered as a tool for computability without a mimetic justification.

3. As mentioned in Section 2.2.3, many studies pointed out the importance of the judge's preferences in the decision. Those preferences are built over time by a trajectory unique to a judge. As a result, other approaches than those based on intelligible rules should be considered.

Under the analogy assumption, Legal Reasoning still arises several questions when it comes to computationality:

1. Are computers able to reasons in terms of analogy?

2. Is Law and its intricacies computationally representable?

Not only case-based reasoning is somehow natural in trials based on legal precedence as it is the case in Common Law systems, but seems to be a natural way of thinking for legal experts when asked to predict the outcome of a case [Ruger 2004]. This can probably be explained by the fact that case-based teaching became a standard in legal teaching since the method has been introduced by Christopher Columbus Langdell [Kimball 2004]. Therefore CBR can also be considered for roman traditions.

Kannai proposes an alternative to standard CBR using rule abductions [Kannai 2014]. He argues that another valid approach to reason by analogy is to learn some rules from a set of similar cases and then to use those rules to reason and predict a new case, by opposition to standard CBR that directly compaire a new case to a set of past cases. The rules represents the prevailing norm in a legal environment at a given moment and thus can evolves while the law corpus remains the same. Such an approach would allow to study and detect concept drifts. Indeed, in its survey of CBR applied to legal domain [Delgado 2007], Delgado clearly mentions the current problems with CBR: similarity and relevance of precedent cases are dynamic, non stationnary as social and governmental laws evolves.

In parallel to the emergence of those considerations, many hybrid case-based reasoning systems have been developed, often coupled with rule-based systems [Branting 1991, Rissland 1991, Stranieri 1999].

In addition to these drawbacks, [Prakken 1998] demonstrates how specific formal structures must be modified from traditional case-based reasoning to legal reasoning before a proper model can be formulated. In a way, they were already pointing out the tremundous effort to transform data into a suitable format. Finally, [Brüninghaus 2005] also pointed the difficulty in information retrieval and transformation, and proposed to use NLP to extract information from cases and generate predictions and explanations. In other words, the main problem with CBR is the same problem that currently faces the AutoML community, namely the automation of data preparation.

2.2.4.2 CATO

Rather than presenting the multiple CBR systems that have been proposed for different areas of the law, we chose to detail CATO because it is probably the most well known CBR applied to the legal environment. Introduced in [Aleven 1997], CATO is a case-based environment based on HYPO system [Ashley 1988, Ashley 2002], to teach student argumentation It selects strategically alternative interpretations of cases to elaborate a more abstract parallel between cases. The database is composed of a textual summary and a factor set for cases in the specific domain of trade secret law. The factors are static and extracted by humans.

The factor hierarchy is also pre-established by human and divided into 16 abstract factors and 26 base level factors connected by 50 links. The links are annoted with a plus (+) or minus (−) to indicate if a factor attacks or defend another one. The links can also be weak or strong.

CATO uses 8 basic *reasoning moves* to build a reasoning:

- Analogizing a problem to a past case with favorable outcome
- Analogizing a problem to a past with with an unfavorable outcome
- Downplaying the significance of a distinction
- Emphasizing the significance of a distinction
- Citing a favorable case to highlight strenghts
- Citing a favorage case to argue that weaknesses are not fatal
- Citing a more on point counterexample to a case city by an oponent
- Citing an as on point counterexample.

The process to justify a favorable decision on an issue is as follow:

1. Process to justify a favorable decision on an issue
2. Point to strengths related to an issue and why it matters
3. Show favorable cases
4. Discuss weaknesses and compensating factors
5. Show cases with favorable outcome but with the same weaknesses

Everything in CATO is created by humans and does not evolve in time: knowledge extracted from the cases, the process of argumentating, the elementary bricks of reasoning. If this lack of flexibility is not a limitation in the context of teaching students, the interest is limited for legal practitioners. Despites these drawbacks, inherent to expert systems before the era of machine learning, the authors showed that the usage of CATO led to a statistically significant improvement in student' argumentation skill compared to traditional teaching mechanisms.

2.2.5 Abstract Argumentation

Abstract Argumentation (AA) [Dung 1995] is a hot topic in non-monotonic reasoning, that is to say where there is a need to reason with pieces of information that can turn to be in contradiction with classical logic. The versatility of AA is thus that it can be used as a descriptive tool, for instance by including agent's beliefs in order to study and understand a given situation, as well as a normative tool where, given a knowledge base, we want to infer the best action or decision to take [Amgoud 2005]. In [Cayrol 2005] the author defines the usage of AA with the following three steps:

1. Defining the arguments and the relation(s) between them.
2. Valuating the arguments using their relations, a strength, etc.
3. Selecting some arguments using some criteria (a *semantic*).

From this very succinct summary, one may catch a glimpse on the interest of AA applied to the legal domain, either as a modeling or decision aid tool: the judgment of a case in both Common Law or Civil Law countries follow this exact procedure of collecting the evidence and legal facts, evaluating their suitability in the context of the case, take a decision based on those facts.

In a similar fashion of CATO , an Abstract Framework (AF) where the arguments are taken manually from a set of cases and the attack deduced from the opinions has been proposed [Bench-Capon 2002, Bench-Capon 2003]. Given a particular case, the arguments that do not apply are removed to obtain a new AF, a subset of the initial one. The key arguments are calculated to determine if the outcome is admissible, i.e. in favor, of the plaintiff.

Oren and al. built a concrete AF by instantiating arguments as inference rules and use Subjectic Logic [Jøsang 2016] to value the strength of the arguments [Oren 2007, Modgil 2009]. After giving an algorithm to propagate the strength of the arguments, they defined a dialogue game protocol for several agents to argue about the state of the environment. Using Assumption-Based Argumentation [Dung 2006], Dung and Thang model a pool of agents arguing about some arguments in front of judges with a final jury taking a decision [Dung 2010]. If the jury can also introduce new arguments, they are limited to considering the probabilities of causal arguments, while the judges are the only one to determine the admissibility of an argument. Other Assumption-Based Argumentation applications to the the legal domain can be found in [Dung 2008, Kowalski 1996]. Meanwhile, several quantitative methods to calculate and propagate the strengths of arguments have been developed [Baroni 2015, Rago 2016] with e.g. Social AA [Leite 2011] that intends to model debates and decision-making in social networks using a voting system, with possible reuse in legal AA.

Recently, CBR based on AA formalism have been proposed. Contrary to previous approaches where arguments are elements of a case, here the arguments are the cases, and the attacks relations between cases. The cases are defined as a set of features and an outcome. In [Čyras 2016a], the outcome of a new case is given depending on the *grounded extension* and a justification[14] to support the decision is built using a Dispute Tree [Dung 2006] while in [Athakravi 2015] the outcome is deduced from rules learnt from past cases. In [Ontañón 2008], the outcome is given after a deliberation between several agents and a fully adaptive and dynamic approach such that the agents learn from each other and are able to resolve conflict through a specific game protocol. In parallel, finding a justification to a case rather than simply a prediction started to be actively explored within the AA community [Fan 2014, Čyras 2016b, GarcíA 2013].

[14]To be precise, some previous cases and some of their specific features. Thus, this is not a legal justification for Roman Law.

2.3 Summary and Limitations

As analyzed in this Chapter, the approaches to Computational Law can be roughly divided between the **data centric** approaches and the **expert systems** methods depending on what the type of information is used.

Among the **statistical models**, we distinguish two categories. First, some methods use ML [Martin 2004a, Ruger 2004, Guimerà 2011, Katz 2017b] or NLP [Aletras 2016] to predict the outcome of a case given the past cases or votes. Papers in the second category focus on modeling and estimating the ideal point of judges to predict votes solely using this estimation [Segal 1989, Segal 1995, Quinn 2002, Quinn 2006, Andrew 2004, Lauderdale 2014, Sim 2014, Islam 2016]. These methods aim at solving the problem of prediction but cannot provide a justification. Finally, all models adopt a binary outcome and must integrate sentencing elements for more complex and detailed outcomes.

	Information	LK	General	Robust	Fully Pred.	Extra Data	Just.
Pred. Models (inc. NLP)							
[Martin 2004a, Ruger 2004]	Data-driven	no	no	no	yes	past cases	no
[Guimerà 2011]	Data-driven	no	yes	yes	no	past votes	no
[Katz 2017b]	Data-driven	no	yes	yes	yes	past cases	no
[Aletras 2016]	Data-driven	no	yes	no	yes	past cases	no
Ideal Point							
[Segal 1989, Segal 1995]	Data-driven	no	yes	no	yes	non-legal	no
[Quinn 2002, Quinn 2006, Andrew 2004]	Data-driven	no	yes	yes	yes	past votes	no
[Lauderdale 2014, Sim 2014]	Data-driven	no	yes	no	yes	amicus	no
[Islam 2016]	Data-driven	no	yes	no	yes	opinions	no
CBR							
[Aleven 1997]	Rule-based	yes	no	no	yes	past cases & legal factors	yes
AA							
[Bench-Capon 2002, Bench-Capon 2003]	Both	yes	no	no	yes	past cases & legal factors	yes
[Oren 2007, Modgil 2009]	Both	yes	yes	yes	yes	norm	yes
[Dung 2006]	Rule-Based	no	yes	yes	yes	-	yes
[Baroni 2015, Rago 2016]	Both	no	yes	no	yes	-	-
[Leite 2011]	Rule-based	no	yes	no	yes	-	-
AA-CBR							
[Čyras 2016a]	Both	no	no	no	yes	past cases	partly
[Athakravi 2015]	Both	no	yes	yes	yes	past cases	partly
[Ontañón 2008]	Both	no	yes	yes	yes	past cases	partly

Table 2.1: Comparison of Computational Law approaches.
The features to compare include: (1) whether the method relies on pre-defined rules or exploit the data (Information), (2) if it uses Legal Knowledge (LK), (3) the capacity to be applied to any case over years (General), (4) to adapt to the environment shifts (Robust), (5) to make a prediction solely based on past information (Fully Pred.), (6) the extra-data used on top of the information about the current case (Extra Data) and (7) the capacity to justify a decision (Just.).

Among the **Expert Systems**, we distinguish two categories: the Case-Based Reasoners and the Abstract Argumentation. **Case-based reasoning** systems evolved in their formalism from ad-hoc structures to the AA structure. However, the most important factor to distinguish between CBR systems is the way they build the justification. [Aleven 1997, Bench-Capon 2002, Bench-Capon 2003] focus on building a justified prediction based on pre-determined legal factor hierarchy and past cases description, while in [Athakravi 2015, Ontañón 2008, Čyras 2016a] the factor hierarchy is deduced from past cases relations. Most CBR systems focus on providing a legal reasoning and thus, neglect the importance of non-legal factors, that is, they implicitly work in an ideal court setting. The non-legal factors are not taken into account in the CBR systems case representation, and thus not taken into account in the justification process. By doing so, they ruin their capacity to handle properly the problem of prediction. Future CBR could separate the prediction problem from the justification problem. First the prediction is made using not only the past cases, but also non-legal indicators that are valid at the moment a prediction is made, then a compatible justification is built.

The law text cannot be reduced to a rule, but a support to express its *essence*. Understanding this substance is the real challenge from a computational point of view. Also, the essence is more than the rule, its power and range of application ; it is also the legitimacy of its application. The legitimacy of a justice decision deeply resides in the reasoning and justification from which the conclusion is drawn.

"The reduction of legal norms to the commands of a political sovereign would mean that law, in the course of modernity, had been dissolved into politics. [...] Under this premise political power could no longer be understood as legal authority, since a law, which as become completely at the disposal of politics would lose its legitimating force" [Habermas 1988].

This might explain the fall of the rule-based systems [Leith 2010] that cannot grab this essence while the CBRs tend to get around the problem by deriving knowledge and rules from the case features. However, this comes with the price of a lack of legal justification and thus the absence of legitimacy of the decision. If a case-based justification might be satisfying up to a certain degree in Common law since it bring hints on the jurisprudence, this is not the case for Roman law.

Finally, CBR neglect the temporal dimension of the legal environment. To overcome this drawback, future work must focus on integrating *dynamic features* into (AA)-CBR systems. Currently, the case features are static. If a feature is e.g. the judge, two cases might be considered close because they are judged by the same person. However, we know that the preferences of the judges change in time and are influenced by local documents such as case amicus. Thus this cannot directly be used as a feature. Conversely, using the value of the ideal point estimation as a feature is not a good idea: the ideal point is strongly dependent on many underlying features (the judge, the area of law, other case feature). Another possibility is to leverage the work on dynamic in AA to add a temporal dimension to Abstract Argumentation-Case-Based Reasoning (AA-CBR) (the relevance between two cases evolves in time and this dimension is not covered).

In **Abstract Argumentation**, apart from AA-CBR, two kinds of approaches emerge. The first one is *positive* and intends to model real-life decision processes or environment [Oren 2007, Modgil 2009, Dung 2006]. The second one is *normative* and tries to elaborate methods to select among the best alternatives and discuss arguments [Baroni 2015, Rago 2016, Leite 2011].

A main pitfall of all the approaches, except the statistical models up to a certain degree, is the lack of automation: the Segal-Cover score relies on manually extracted information, the features hierarchy in CATO are manually constructed, the AF are manually constructed, etc. Future works should focus on the automation using the recent progresses in NLP and the availability of data.

From this Chapter, we conclude that AA provide the best way to explain decisions within a given legal framework. Statistical methods provide the best predictive results but cannot explain their decisions. CBR systems suffer from a lack of evolutivity and automation while being a natural reasoning framework for the legal domain. In this thesis, we propose a new

CBR system named HCBR working in abstract spaces such that the case representation is not a limiting factor, along with statistical concepts such as metric learning to overcome the lack of evolutivity. In a way, our work aligns with [Bench-Capon] in which the author argues that traditional AI is still needed in combination with data-centric method, in the field of legal analytics. A consequence of these choices is that HCBR can be fully integrated to traditional ML workflow and embeded in AutoML systems.

CHAPTER 3

Classification and Metric Learning

Whoever has fully understood the doctrine of absolute irresponsibility can no longer include the so called rewarding and punishing justice in the idea of justice, if the latter be taken to mean that to each be given his due.

HUMAN TOO HUMAN, NIETSZCHE

Contents

In this chapter, we present the problem of classification and the different approaches to solve it. In this thesis, we decided to focus on the classification problem as it is one of the most widely studied problem in ML due to the large amount of situations that can be modeled as such problem. Specifically, we focus on Binary Classification because more complex versions of the problem such as multiclass or multilabel can be decomposed into multiple binary classification problems, and the formal framework is the same. Additionally, we present the state of the art in Metric Learning that is the usual way to work in spaces without metric.

3.1 Binary classification

In machine learning, the problem of classification consists in finding a mapping from an input vector space $\mathcal{X}$ to a discrete decision space $\mathcal{Y}$ using a set of examples. The binary classification problem is a special case such that $\mathcal{Y}$ has only two elements. It is often viewed as an approximation problem s.t. we want to find an estimator $\bar{J}$ of an unknown mapping J available only through a sample called *training set*. A training set $(\mathbf{X},\mathbf{y})$ consists of N input vectors $\mathbf{X} = \{\mathbf{x}_1,...,\mathbf{x}_N\}$ and their associated correct class $\mathbf{y} = \{y_i = J(\mathbf{x}_i)\}_{i=1}^N$.

Let $\mathcal{J}(\mathcal{X},\mathcal{Y})$ be the class of mappings from $\mathcal{X}$ to $\{-1,1\}$, or simply $\mathcal{J}$ if there is no ambiguity. A machine learning algorithm to solve binary classification is an application $\mathcal{A} : \mathcal{X}^N \times \mathcal{Y}^N \to \mathcal{J}$ capable of providing a good approximation for any $J \in \mathcal{J}$ under some assumptions on the *quality* of the training set. In practice, it is not reasonable to search directly in $\mathcal{J}$ and some assumptions on the "shape" of J are made s.t. $\bar{J} = \mathcal{A}(\mathbf{X}, \mathbf{y})$ belongs to a *hypothesis* space or model space $\mathcal{H} \subset \mathcal{J}$. This restriction implies not only that the exact mapping J is not always reachable but might also not be approximated correctly by any element of $\mathcal{H}$. The choice of the model space is thus crucial as it should be large enough to represent fairly complex functions and small enough to easily find the best available approximation.

In general, a robust classification algorithm must be able to approximate correctly any possible mapping. The problem of finding such algorithm consists in minimizing the generalization error for all possible mappings. Formally, it consists in solving:

$$\min_{\mathcal{A}} \sum_{J \in \mathcal{J}} \int_{\mathcal{X}} ||J(\mathbf{x}) - \bar{J}(\mathbf{x})|| \mu(\mathbf{x})\, d\mathbf{x} \tag{3.1}$$

where μ is a probability measure over $\mathcal{X}$.

In practice, the generalization error cannot be computed, and the set of possible mappings is unreasonably large. For these reasons, we aim at minimizing the empirical classification error on a reasonably large set of datasets $\mathcal{D} = \{(\mathbf{X}_1, \mathbf{y}_1), ..., (\mathbf{X}_K, \mathbf{y}_K)\}$, i.e.

$$\min_{\mathcal{A}} \sum_{(\mathbf{X},\mathbf{y}) \in \mathcal{D}} \sum_{(\mathbf{x},y) \in (\mathbf{X},\mathbf{y})} \mathbb{1}_{\{y \neq \bar{J}(\mathbf{x})\}}. \tag{3.2}$$

To highlight the differences between standard classification problem and the approach developped in this thesis, we briefly present linear models for classification followed by the challenges of classification in unstructured spaces. For an overview of theoretical results on classification, we refer the reader to [Boucheron 2005].

3.1.1 Linear binary classification

The problem of binary classification is commonly studied with $\mathcal{X} = \mathbb{R}^M$. Many popular classification approaches such as SVM [Vapnik 2013], perceptron [Rosenblatt 1958] or logistic regression [Cox 1958] define the model space as the set of M-hyperplanes. A M-hyperplane is uniquely defined by a vector $\mathbf{w} \in \mathbb{R}^M$ and a bias $w_0 \in \mathbb{R}$, and is formulated by

$$h_{\mathbf{w}}(\mathbf{x}) = <\mathbf{w}, \mathbf{x}> + w_0 = 0 \tag{3.3}$$

The *homogeneous* notation consists in adding w_0 to $\mathbf{w}$ by rewriting $\mathbf{x}$ such that $\mathbf{x} = (1, x_1, ..., x_M)$. The hyperplane equation (3.3) is then expressed by $h_{\mathbf{w}}(\mathbf{x}) = <\mathbf{w}, \mathbf{x}>$. A hyperplane separates $\mathbb{R}^M$ into two regions, and thus, can be used as a discriminative rule s.t.

$$\bar{J}_{\mathbf{w}}(x) = \begin{cases} 1 & h_{\mathbf{w}}(\mathbf{x}) > 0 \\ -1 & h_{\mathbf{w}}(\mathbf{x}) \leq 0 \end{cases} \tag{3.4}$$

Then, given a training set $(\mathbf{X}, \mathbf{y})$, the classification problem is equivalent to finding the best hyperplane s.t. it minimizes a certain *loss* function over the training set [Lin 2004]:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{N} \ell(\mathbf{w}, \mathbf{x}_i) + \lambda R(\mathbf{w}) \tag{3.5}$$

where $R(\mathbf{w})$ is *regularization* term to prevent overfitting (usually $||\mathbf{w}||_2^2$ or $||\mathbf{w}||_1$) and $\lambda > 0$ a hyperparameter controlling the effect of regularization. Note that (3.5) is a parametric problem due to the choice of model space. Several losses functions exists and are generally based on the *margin* of $\mathbf{x}_i$ which is defined by $m(\mathbf{w},\mathbf{x}_i) = J(\mathbf{x}_i)h_{\mathbf{w}}(\mathbf{x}_i)$. The margin represents the distance of a vector $\mathbf{x}_i$ to the hyperplane defined by $h_{\mathbf{w}}$. It is positive if $\mathbf{x}_i$ is correctly classified, negative otherwise. For the most known, the 0-1 loss, hinge loss and log loss are defined by

$$\begin{aligned} \ell_{01}(\mathbf{w},\mathbf{x}) &= \mathbb{1}_{\{m(\mathbf{w},\mathbf{x})\leq 0\}} \\ \ell_{\text{hinge}}(\mathbf{w},\mathbf{x}) &= \max(0, 1 - m(\mathbf{w},\mathbf{x})) \\ \ell_{\log}(\mathbf{w},\mathbf{x}) &= \ln(1 + e^{-m(\mathbf{w},\mathbf{x})}) \end{aligned} \tag{3.6}$$

The Perceptron algorithm uses the 0-1 loss, while SVM minimizes the hinge loss and the Logistic Regression the log loss.

3.1.2 Non-linear SVM and Gaussian kernel

The Equation (3.5) can solve only linearly separable problems. To handle non-linearly separable problems, the main idea is to use the kernel *trick* to map the training set into a higher dimensional space s.t. it becomes linearly separable [Hofmann 2008]. Let denote by $\phi : \Omega \to \mathcal{V}$ this feature map.

A kernel is a symmetric continuous function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ s.t. $K(\mathbf{x},\mathbf{x}) = K(\mathbf{x}',\mathbf{x})$. The kernel is positive semi-definite iff $\sum_i^n \sum_j^n K(\mathbf{x}_i,\mathbf{x}_j)c_i c_j \geq 0$ for any finite sequence of $\{x_i\}_{i=1}^n$, $\{x_j\}_{i=1}^n$ and any choice of real for c_i and c_j. A linear classifier computes a prediction for an unlabeled input $\mathbf{x}'$ as $\bar{y} = \text{sgn} \sum_{i=0}^n w_i y_i < \mathbf{x}_i, \mathbf{x}' >$ where $\mathbf{x}_i$ is the i-th training example. ϕ image being in a high dimensional space, it makes the scalar product in $\mathcal{V}$ impossible to compute in practice. The kernel trick thus consists in replacing the costly scalar product by a kernel function such that $K(\mathbf{x},\mathbf{x}'') =< \phi(\mathbf{x}), \phi(\mathbf{x}'') >$. More precisely, Mercer's Theorem states that for any positive semi-definite kernel on $\Omega \times \Omega$ there exists a well defined scalar product in $\mathcal{V}$ such that $K(\mathbf{x},\mathbf{x}') =< \phi(\mathbf{x}), \phi(\mathbf{x}'') >$. This means that in practice, we select K a priori rather than determining it depending on a certain ϕ. A typical kernel-based classifier then computes $\bar{y} = \text{sgn} \sum_{i=0}^n w_i y_i K(\mathbf{x}_i, \mathbf{x}')$ where the kernel is used as a similarity measure.

In summary, the kernel trick allows linear classifier to learn non-linear boundaries by replacing a scalar product in a high dimensional space by an easy-to-compute kernel function. For a more formal and exhaustive presentation of kernel-based methods, we refer the reader to [Mohri 2018, Cucker 2002, Hofmann 2008]

One popular kernel for SVM is the Gaussian kernel [Williams 2006], also known as Radial basis function, defined by $K_G(\mathbf{x},\mathbf{x}') = \exp(\frac{||\mathbf{x}-\mathbf{x}'||^2}{2\sigma^2})$ where σ how far is the influence of examples around them.

3.1.3 Other Approaches to Classification

The method called HCBR developed in Chapter 6 is discriminative and fits the mathematical framework introduced above. Therefore, it models $p(y|\mathbf{x})$. There exists other frameworks to handle binary classification. We briefly introduce them as they will be compared to HCBR in Section 6.3.

Generative techniques such as Naive Bayes or Bayesian Networks [Friedman 1997] estimate the joint probability $p(\mathbf{x}, y)$. They make assumptions on the data distribution (e.g. a mixture of gaussians) and estimate the unknown parameters to generate predictions. We extend HCBR to a generative model in Chapter 7

Another popular family of classification techniques are based on trees [Venables 2002, Breiman 2017]. For instance, Random Forest [Ho 1995, Breiman 2001] is one of the most widely used method due to its good performances [Couronné 2018] and several extensions have been developed [Geurts 2006, Friedman 2001]

Last but not least, DL is a family of techniques based on stacking layers of neurons whose weights are adjusted using gradient back-propagation [LeCun 2015]. For the most basic network, each layer can be seen as performing a logistic regression. It represents the state-of-the-art in classification in multiple domains [Schmidhuber 2015]: vision, audio and natural language processing, to name few.

3.2 Metric learning

Choosing an appropriate pairwise metric to measure distance between two points is crucial in the success of classification algorithms [Davis 2007]. Learning a metric consists in finding a projection f from an initial space to a Euclidian space s.t. for any elements $\mathbf{x}$ and $\mathbf{x}'$, $d(\mathbf{x},\mathbf{x}') = ||f(\mathbf{x}) - f(\mathbf{x}')||$. The metric should reflect the semantic difference between objects. If the elements are non-numerical, a metric can be viewed as a proxy to represent the elements in a vector space s.t. it becomes possible to apply standard classification methods. However, as reported by [Bellet 2013], the literature mostly focused on numerical data, i.e. when the elements already belong to a vector space.

The most common setting for metric learning is to find a parametrization M of the Mahalanobis pseudometric $d_M(\mathbf{x},\mathbf{x}') = \sqrt{(\mathbf{x}-\mathbf{x}')^T M(\mathbf{x}-\mathbf{x}')}$ using the training set $\mathbf{X}$ and under the constraint that M is Positive Semi-Definite (PSD). Using M as the identity recovers the usual Euclidian metric, whereas setting M as the covariance matrix of $\mathbf{X}$ gives the original Mahalanobis distance [Mahalanobis 1936]. Most methods aim at finding the parameters that best agree with the following three sets of relations on the examples:

- $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{x}_j) \,:\, \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are similar}\}$, (must-link)
- $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{x}_j) \,:\, \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are dissimilar}\}$, (cannot-link)
- $\mathcal{R} = \{(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) \,:\, \mathbf{x}_i \text{ more similar to } \mathbf{x}_j \text{ than to } \mathbf{x}_k\}$. (relative)

However, supervised methods usually derived these sets (implicitly or explicitly) from the examples and a proper notion of neighborhood.

We now present the approaches with similarities to the one introduced in this Chapter 6. For more detailed surveys on metric learning, we refer the reader to [Bellet 2013, Wang 2015].

LMNN [Weinberger 2006, Weinberger 2009] is one the most popular Mahalanobis metric learning techniques and many other linear methods are based on it. The constraint sets $\mathcal{S}$ and $\mathcal{R}$ are defined by a notion of neighborhood:

- $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{x}_j) \,:\, y_i = y_j \text{ and } \mathbf{x}_j \text{ belongs to the } k\text{-nearest neighbors of } \mathbf{x}_i\}$,

- $\mathcal{R} = \{(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) \,:\, (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S},\ y_i \neq y_k\}$.

In the original work, the neighborhood is defined using the Euclidian distance, and thus, assumes the elements live in a vector space. To avoid this, the method presented in this thesis defines the neighborhood based on set intersections to derive $\mathcal{D}$ and $\mathcal{R}$. The PSD matrix W is found by solving the following convex optimization problem:

$$\begin{aligned} M^* = \operatorname*{arg\,min}_{M \in \mathbb{S}^n_+} (1-\mu) &\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} d_M^2(\mathbf{x}_i,\mathbf{x}_j) + \mu \sum_{i,j,k} \varepsilon_{i,j,k} \\ \text{s.t. } & d_M^2(\mathbf{x}_i,\mathbf{x}_k) - d_M^2(\mathbf{x}_k,\mathbf{x}_j) \geq 1 - \varepsilon_{i,j,k}, \forall (\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k) \in \mathcal{R} \\ & \varepsilon_{i,j,k} > 0 \end{aligned} \tag{3.7}$$

with $\mu \in [0,1]$ a tradeoff parameter, and $\varepsilon_{i,j,k}$ some slack variables.

OASIS [Chechik 2010] learns a bilinear similarity metric of the form $d_M(\mathbf{x},\mathbf{x}') = \mathbf{x}^T M \mathbf{x}'$ without the PSD constraint on M. It can define similarity between instances of different dimensions such as BoW. Relaxing the PSD constraint allows the author to use an efficient online Passive-Aggressive algorithm [Crammer 2006] to solve for M:

$$\begin{aligned} M^t = \operatorname*{arg\,min}_{M,\,\varepsilon} & \frac{1}{2}||M - M^{t-1}||^2_{\mathcal{F}} + C\varepsilon \\ \text{s.t. } & 1 - d_M^2(\mathbf{x}_i,\mathbf{x}_j) + d_M^2(\mathbf{x}_i,\mathbf{x}_k) \leq \varepsilon \\ & \varepsilon > 0 \end{aligned} \tag{3.8}$$

with C a regularization factor and $||.||_{\mathcal{F}}$ the Frobenius norm.

SLLC [Bellet 2012] focuses on learning a bilinear similarity matrix as a quadratic constraint program defined by

$$\min_{M \in \mathbb{R}^{n \times n}} \frac{1}{n} \sum_{i=1}^{n} \ell_{\text{hinge}}\Big(1 - y_i \frac{1}{\gamma |S|} \sum_{\mathbf{x}_j \in \mathcal{D}} y_j d_M(\mathbf{x}_i,\mathbf{x}_j)\Big) + \beta ||M||^2_{\mathcal{F}} \tag{3.9}$$

where $\mathcal{D}$ is a sample from the training set, γ a margin parameter and β a regularization parameter. The objective of SLLS is to find a metric s.t. the elements of one class are in average more similar than the elements of the other class by a margin γ.

MMDA [Kocsor 2004] learns the Euclidian distance between $W^T \mathbf{x}_i$ and $W^T \mathbf{x}_j$ and thus is a linear method. To do so, it learns k projection hyperplanes $\{\mathbf{w}_r\}_{r=1}^k$ by solving the following optimization program:

$$\begin{aligned} \min_{W,\mathbf{b},\varepsilon_r} & \frac{1}{2} \sum_{r=1}^{k} ||\mathbf{w}_r||^2_2 + \frac{C}{n} \sum_{r=1}^{k} \sum_{i=1}^{n} \varepsilon_{ri} \\ \text{s.t. } & \ell_{\text{hinge}}(\mathbf{w}_r^T \mathbf{x}_i + b_i) \leq 1 - \varepsilon_{ri} \\ & \varepsilon_{ri} > 0 \\ & W^T W = Id \end{aligned} \tag{3.10}$$

where ε_r are slack variables to penalize margins and C a regularization parameter.

LSMD [Chopra 2005] learns a non-linear projection G_W parametrized by W s.t. $d(\mathbf{x},\mathbf{x}') = ||G_W(\mathbf{x}) - G_W(\mathbf{x}')||_1$ is small when the $\mathbf{x}$ and $\mathbf{x}'$ are in the same class and large otherwise. The parameter W represents the weights of a neural network used to learn the projections.

3.3 Data Wrangling and Missing Data

Data wrangling consists in selecting, transforming and curating data that are unstructured or not suitable for a given selected algorithm. It is commonly accepted that up to 80% of data scientists time mau be spent on data wrangling [Chessell 2014, Pal 2018, Cog 2019]. On top of the time consumption, the whole preprocessing step has a huge impact on the final model quality. For instance, in [Crone 2006], the authors showed that the accuracy obtained by Neural Network, SVM and Decision Trees are significantly impacted by data scaling, sampling and encoding. For a more comprehensive view on data processing impact, we refer the reader to [Dasu 2003].

Data wrangling addresses several challenges notably dealing with missing data and outliers or combining data from multiple sources in an automated way [Furche 2016, Kandel 2011]. In this section, we present the work related to these challenges. However, we believe that another way to tackle data wrangling challenges is to design models that are less sensitive to data preprocessing, missing data or data representation. Not only less sensitive models decrease the time allocation needed to create efficient data pipelines, but it also decreases the CPU time by shrinking the data pipeline itself. For instance, the method proposed thoroughly this thesis does not need to remove outliers because they will be marginalized by the model selection itself. As HCBR works with any unstructured space, it becomes easy to combine data from multiple sources.

Modeling datasets in a latent space using meta-features allows to predict the impact of preprocessing operators on the model accuracy [Bilalli 2017]. To merge multiple data sources, semi-automated tools were developed such a Clio [Haas 2005] or Schema Mapper [Robertson 2005].

The most common technique to deal with missing numerical values is replacing the value (imputation) using the mode, average or a value obtained by a model, over the available data. The imputation might be done using a k-nearest neighbors algorithm [Batista 2002], local regression [Preda 2010] or Random Forest [Stekhoven 2011]. In general, machine learning approaches tend to outperform traditional statistical methods [Kuhn 2013]. To handle values under Missing At Random (MAR) settings, one can use multiple imputations [Royston 2004, Van Buuren 2018]. However, this technique requires to generate several variations of the training set and build several models that are then averaged s.t. once again, it requires a large computational effort.

Even when values are Missing Completely At Random (MCAR), there are scalability issues with practical machine learning. Assume some values are MCAR (e.g. due to a sensor malfunctioning for some time), or that new pieces of information are available after an initial model was built (e.g. adding new sensors). In the case of linear classification, e.g. in $\mathbb{R}^3$, if an element has only 2 components it does not describe a point but a plane and as a result, there is no way to separate it with a (hyper)plane.

Conversely, consider a model built in $\mathbb{R}^2$ with some additional information available afterward, s.t. the classification instance now lives in $\mathbb{R}^3$. Mathematically, there is no guarantee that the model in two dimensions is even close to the model built from scratch in three dimensions. For instance, if the points are not linearly separable in two dimensions but are in three dimensions, there is no chance for the projection of the plane into the subspace of two dimensions to be the model found while working only in this subspace. Despite multiple algorithms have an *online* counterpart, as far as we know, there is no significant work on having this *horizontal* scalability (as opposed to vertical scalability, i.e. adding more training examples). One step toward horizontal scalability is to work with unstructured spaces as proposed in this doctoral thesis.

3.4 Classification and Hypergraph

Hypergraphs generalize graphs and can be used to represent higher order relations while graphs are limited to binary relations. A hypergraph is defined by a set of vertices and a collection of hyperedges where each hyperedge is a subset of this set of vertices. Therefore, a graph is a special case of hypergraph for which each hyperedge contains only two vertices. For additional results on hypergraphs, we refer the reader to [Berge 1984].

Definition 3.4.1 (Hypergraph)**.** *A hypergraph is defined by* $H = (V, \mathbf{X})$ *with* V *a set of vertices,* $\mathbf{X}$ *the hyperedges s.t.* $\forall \mathbf{x} \in \mathbf{X},\ \mathbf{x} \subseteq V$.

A hypergraph can be viewed as a collection of subsets $\mathbf{X}$ of a given set of vertices V.

Recently hypergraphs have been used as data representation, and some classification algorithms on hypergraph have been proposed. A vast majority of approaches models the objects to classify as the set of vertices and constructs the hyperedges as the representation of a metric. This conventional approach is known as *neighborhood-based* hypergraph. The metric relies on some assumptions on the data or a specific representation (e.g. Zernike moment and histogram of oriented gradient to measure the distance between images in [Lifan 2017]) and for each vertex, a hyperedge is created to represent its k-nearest neighbors [Huang 2010].

The problem of classification on hypergraph consists in labeling some unlabeled vertices given a training set such that all vertices in a same hyperedge have the same label. As all the vertices are known a priori, the problem is part of transductive learning. To learn the labels, the standard approach is to minimize a cost function based on a hypergraph equivalent of a graph Laplacian [Lifan 2017, Zhou 2007] with a structural risk:

$$C(\mathbf{x}) = \mathbf{x}^t \Delta \mathbf{x} + \gamma ||\mathbf{x} - \mathbf{y}||^2 \tag{3.11}$$

where Δ is the hypergraph Laplacian, $\gamma > 0$ a regularization factor and $||.||$ a norm. The vector $\mathbf{y}$ represents the initial labels for all vertices with $y_i = 0$ for unlabeled vertices, a negative (resp. positive) value for label -1 or 1.

On the contrary, the **algorithm HCBR proposed in this thesis** models the elements to classify as the hyperedges and the vertices as the different components of those elements. As far as we know, there is no previous work that uses this modeling choice. In addition, it does not require knowing all the elements before building the model: our approach is inductive. More important, as most previous work consists in building metrics based on the feature representation, it obviously conflicts with our goal of agnosticity described in the previous section.

CHAPTER 4

End-to-End Machine Learning and AutoML

There can be no doubt that the knowledge of logic is of considerable practical importance for everyone who desires to think and to infer correctly.

ALFRED TARSKI

Contents

In this chapter, we introduce the main problem that the AUTOML community focuses on, namely Combined Algorithm Selection and Hyperparameter-tuning (CASH). After presenting the different approaches to solve it in Section 4.1, we discuss in Section 4.5, why CASH is a difficult problem, why its formulation might not be the most suitable and why current tools are not satisfying w.r.t. to the way machine learning practitioners work.

For a broader review on AUTOML, we refer the reader to [Elshawi 2019, Hutter 2019].

4.1 CASH problem

The learning problem consists in finding or constructing an approximation of an unknown function $f\colon \mathcal{X} \to \mathcal{Y}$. A *learning* algorithm A maps a set of training points $\{d_i\}_{i=1}^n$ with $d_i = (\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ to $\mathcal{Y}^{\mathcal{X}}$. A learning algorithm A is parametrized by some hyperparameters $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ that modify the way the algorithm $A_{\boldsymbol{\lambda}}$ learns. Each hyperparameter λ_i belongs to a space Λ_i and $\boldsymbol{\Lambda}$ is a subset of the cross-product of each domain, i.e. $\boldsymbol{\Lambda} \subset \Lambda_1 \times \ldots \times \Lambda_n$. In general, Λ can be more structured (conditional tree, directed acycle graph,...).

The AUTOML community focuses on the Combined Algorithm Selection and Hyperparameter-tuning problem (CASH) formulated as an optimization problem.

Definition 4.1.1 (Combined Algorithm Selection and Hyperparameter-tuning)**.** *Given a portfolio of algorithms* $\mathcal{A} = \{A^{(1)}, \ldots, A^{(m)}\}$ *with associated hyperparameter spaces* $\boldsymbol{\Lambda}^{(1)}, \ldots, \boldsymbol{\Lambda}^{(m)}$, *the Combined Algorithm Selection and Hyperparameter Optimization (*CASH*) problem is defined by:*

$$A^*_{\boldsymbol{\lambda}^*} = \operatorname*{arg\,min}_{A^{(j)}\in\mathcal{A},\boldsymbol{\lambda}\in\boldsymbol{\Lambda}^{(j)}} \frac{1}{k}\sum_{i=1}^{k}\mathcal{L}(A^{(j)}_{\boldsymbol{\lambda}},\mathcal{D}^{(i)}_{train},\mathcal{D}^{(i)}_{test}), \tag{CASH}$$

where $\mathcal{L}$ is a loss function (e.g. error rate) obtained on the test set by the model learned by algorithm A parametrized by $\boldsymbol{\lambda}$ over the training set.

4.2 Black-box optimization and surrogate learning

The most basic technique for hyperparameter tuning is a grid search or factorial design [Montgomery 2017], which consists in exhaustively testing parameter configurations on a grid. In practice, this approach is computationally intractable. An efficient alternative, called Randomized search [Bergstra 2012], consists in testing configurations (pseudo)randomly until a certain budget is exhausted.

Beyond those naive approaches, the most prominent approach to tackle CASH consists in iteratively building an approximation, called *surrogate model*, of the optimization objective.

CASH can be seen as special instance of the following black-box optimization problem:

$$\text{Find } \boldsymbol{\lambda}^* \in \operatorname*{arg\,max}_{\boldsymbol{\lambda}\in\boldsymbol{\Lambda}} \mathcal{F}(\boldsymbol{\lambda}), \tag{4.1}$$

where $\boldsymbol{\Lambda}$ is the space of machine learning onfiguration, and $\mathcal{F}(\boldsymbol{\lambda})$ the performance of the model learned over the dataset using the configuration $\boldsymbol{\lambda}$. At step t, surrogate model $\hat{\mathcal{F}}$ is learned from the history $\{\boldsymbol{\lambda}_k, \mathcal{F}(\boldsymbol{\lambda}_k)\}$ for $k = 1...t$ of previously selected configurations and associated performances. An acquisition function is used to determined the most promising configuration for $\boldsymbol{\lambda}^*_{t+1}$.

The difference between the various surrogate approaches lies into the model space assumption and the acquisition function. Sequential Model-based Algorithm Configuration (SMAC) [Hutter 2011, Bergstra 2011] is based on Random Forest, so are frameworks based on SMAC such as AutoWeka [Thornton 2013, Kotthoff 2017] or AutoSklearn [Feurer 2015]. Hyperopt [Bergstra 2015] uses a Tree-structured Parzen Estimator (TPE) while Spearmint [Snoek 2012] is based on Gaussian processes (GP).

An acquisition function is used to determined the next configuration to be sampled. Most of those functions are based on Bayesian optimization [Wilson 2018, Frazier 2018]. One popular strategy is to select $\boldsymbol{\lambda}_{k+1}$ such that it maximizes the expected improvement [Močkus 1975].

As an alternative to Bayesian optimization, [Rakotoarison 2018] proposes to use Monte-Carlo Tree Search to iteratively explore a tree-structured search space while pruning the less promising configurations.

4.3 Multi-fidelity optimization

The black-box optimization problem defined above is expensive: an iteration means preprocessing the whole dataset through the pipeline and then training a model, often many times as cross-validation is preferred to validate the result. Multi-fidelity optimization focuses on decreasing the computational cost by using large number of *cheap* low-fidelity evaluations. For this, several approaches have been considered.

Extrapolating the model learning curve from the available training data provides an early stop criterion that reduces the computation time [Domhan 2015, Swersky 2014].

Bandit-based algorithms such as Successive halving [Jamieson 2016] or HYPERBAND [Li 2018] find configurations by greedily allocating more budget to promising configurations. For instance, HYPERBAND randomly selects configurations and iteratively removes unpromising candidates while increasing the budget for the promising ones.

In [Nalepa 2018], the authors uses a genetic algorithm to sample a subset of representative input vectors in order to speed-up the model training while increasing the model performances. Genetic algorithms are also used to search for the whole pipeline as in TPOT[Olson 2016] or AUTOSTACKER[Chen 2018].

Last, despite a small configuration space, REINBO[Sun 2019], a reinforcement learning approach, has been shown to outperform TPE, AUTOSKLEARN and TPOT on many datasets.

4.4 Meta-learning and warm-start

Another research direction consist predicting and recommending good pipelines or operators for a given dataset or task. Referred to as meta-learning, it is particularly important for Bayesian optimization, extensively used to solve CASH, since the quality of the results is conditioned by the surrogate initialization. To solve this problem, known as coldstart, several solutions have been investigated.

In AUTOSKLEARN [Feurer 2015], about 140 datasets are represented as vectors made of 38 meta-features, and associated to the best pipeline ever found. When a new dataset is used, AUTOSKLEARN initializes the search process with the best configuration found for the closest dataset w.r.t. Euclidian distance in the meta-feature space. A more generic approach consists in learning the metric between datasets, using, e.g. a Siamese Network [Kim 2017].

At a lower level, [Bilalli 2017] proposes to predict the impact of individual preprocessing operators rather than the whole pipeline. The authors considered discretization, data imputation, PCA, normalization, standardization and an encoding method for categorical variables.

For more extensive surveys on meta-learning, we refer the reader to [Elshawi 2019, Vanschoren 2018]

4.5 Limits of current approaches

The intrinsic difficulty of building a machine learning pipeline lies in the nature of the search space:

- the objective is non-separable i.e., the marginal performance of an operator a depends on all the operators in all the paths leading to a from the source,

- within the configuration space of a specific operator a, there might be some dependencies between the hyperparameters (e.g. for Neural Networks, the coefficients α, β_1 and β_2 make sense only for Adam solver [Kingma 2014]).

Therefore, building a machine learning pipeline is a mix between selecting a proper sequence of operations and, for each operation, selecting the proper configuration in a structured and conditional space. On the contrary, most AutoML systems handle the problem by aggregating the whole search space, losing the sequential aspect of it. A notable exception is Mosaic [Rakotoarison 2018], inspired by Alpha3Dm, that explores the search space in terms of actions on operators (insertion, deletion, etc.).

A second limitation is that most AutoML frameworks propose a static search space and, even more constraining, a fixed *pipeline prototype* i.e., a high-level structure defining an ordered sequence of operator types (a precise definition is given in Section 8.1). For instance, AutoSklearn has a fixed pipeline made of one feature selection operator among 13 operators, and one up to three data preprocessing operator among only four. Those data preprocessing operators are of various nature: one-hot encoder, a specific imputation, balancing and rescaling method. There is no possibility to add custom operators nor to specify additional constraints in case some additional knowledge is available (e.g. no need for imputation since there is no missing values).

To the best of our knowledge, the only approach that uses a non-predetermined sequence of operators is Tpot [Olson 2016], but it is not possible to add additional constraints.

This lack of flexibility is a problem in practice because the user might have different understanding of the problem and thus, would like to define a search space depending on the problem she tries to solve. This motivates us to reformulate the CASH problem in a more general and flexible way, without adding computational overhead to its resolution. This reformulation is given in Chapter 8.

Part II

Contributions

CHAPTER 5

The European Court of Human Rights OpenData

Who would have thought, indeed, in the immediate aftermath of the war, that citizens would one day be able to obtain a judgment against a State from an international court? [...] An acquired right that we have a duty to preserve.

PRESIDENT DEAN SPIELMANN, ABOUT THE ECHR

Contents

This chapter presents an exhaustive and unified repository of judgments documents, called *ECHR-DB*, based on the European Court of Human Rights. The need of such a repository is explained through the prism of the researcher, the data scientist, the citizen, and the legal practitioner. Contrarily to many open data repositories, the full creation process of *ECHR-DB*, from the collection of raw data to the feature transformation, is provided by means of a collection of fully automated and open-source scripts. It ensures reproducibility and a high level of confidence in the processed data, which is one of the most important issues in data governance nowadays. The experimental evaluation was performed to study the problem of predicting the outcome of a case, and to establish baseline results of popular machine learning algorithms. The obtained results are consistently good across the binary datasets with an accuracy comprised between 75.86% and 98.32%, having the average accuracy equals to 96.45%, which is 14pp higher than the best known result with similar methods. We achieved a F1-Score of 82% which is aligned with the recent result using BERT. We show that in a multilabel setting, the features available prior to a judgment are good predictors of the outcome, opening the road to practical applications. This chapter is based on [Quemy 2020c] and [Quemy 2020b].

We observe a shift from reasoning techniques and expert systems to data-centric approaches [Conrad 2018, Antoniou 2018], which use ML algorithms. However, to provide satisfactory prediction ML models need to be trained on large datasets. The availability of such real datasets is limited in practice, and it remains a major problem for researchers and practitioners. There exist few initiatives to provide unified repositories of clean data. Moreover, they are limited to specific courts, rather incomplete or behind paywalls. These observations motivated us to build an open repository of judgment documents. For the European Court of Human Rights, there exists database HUDOC [1] that contains all judgments since its creation. However, it is impossible to access multiple documents at once and case documents are not unified in the way that they offer data in a tabular format and free unstructured texts. There exists two significant databases [Chalkidis 2019, Cichowski 2017]. However, they are both static in the sense they are not updated by the authors. Additionally, they provide only textual information, and no meta-data or additionnal information available about each case. Finally, [Cichowski 2017] is not easily available since it requires contacting the authors to access the data. In other words, despite their public availability, it is difficult to access the data and work with them.

For these reasons, the overall **goal** of this project is to provide an exhaustive and unified set of data, along with metadata, about one of the main European legal institution, namely the European Court of Human Rights. The importance of such work is as follows:

- to draw the attention of researchers on this domain that has important consequences on the society;
- to allow researchers and practitioners to easily study the European Court of Human Rights;
- to provide a unified benchmark to compare ML techniques dedicated to the legal domain;
- to provide a similar and more complete repository for Europe as it already exists for the United States judicial system, notably because the law systems are different in both sides of the Atlantic.

[1] https://hudoc.echr.coe.int

The **contributions** of this chapter can be summarized as follows.

- First, we provide a benchmark set for ML algorithms. It is composed of (almost) all cases judged by the European Court of Human Rights since its creation. The data is cleaned and transformed to ease the exploration and usage of ML algorithms.
- Second, we provide the whole data extraction, transformation, integration, and loading (ETL) pipeline used to generate the benchmark data repository, as the open-source software. This technical contribution aims at increasing the trust in the processed data and ease future iterations of the benchmark, to integrate new cases and data per case.
- Third, we provide exhaustive and high-quality repository, called *ECHR-DB*, of judicial documents for diverse ML problems in the legal domain, based on the European Court of Human Rights documents.
- Fourth, we present the first analysis of the benchmark with standard classification algorithms, in order to predict the outcome of cases and establish baseline models for comparison with future studies. The experiments study binary classification, like previous studies, but also multiclass and multilabel classification, which represent more complex situations.

This chapter comes with *supplementary material* available on GitHub[2]. It contains additional examples about the data format, as well as all secondary results of the experiments that we omitted due to space constraints.

The plan of this chapter is as follows. The whole ETL pipeline is presented in details in Section 5.2. Section 5.3 presents the datasets for our experiments. Sections 5.4, 5.5, and 5.6 discuss the results of experiments on the quality of prediction models built by binary, multiclass, and multilabel classification algorithms, respectively. Finally, Section 7.5 concludes the chapter by discussing the remaining challenges and future work. Appendix A outlines the functionality of *ECHR-DB* and its user interface.

5.1 ECHR-DB Overview

The *ECHR-DB* repository aims at providing exhaustive and high-quality database for diverse problems, based on the European Court of Human Rights documents from HUDOC . The main objectives of this project are as follows: (1) to draw the attention of researchers on this domain that has important consequences on the society and (2) to provide a similar and more complete database for Europe as it already exists in the United States, notably because the law systems are different in both sides of the Atlantic.

The final data is available for direct download through a portal available at https://echr-opendata.eu under the **Open Database Licence (ODbL)**. To ensure availability, a miror is available at the Open Science Framework[3] at the following address: https://osf.io/52rhg/.

[2]https://echr-od.github.io/ECHR-OD_project_supplementary_material/
[3]https://osf.io

The creation scripts and website sources are provided under **MIT Licence** and they are available on GitHub [Quemy]. The project offers data in several format:

- The unstructured format is a JSON file containing a list of all the information available about each case, including a tree-based representation of the judgment document (cf., Section 5.2).
- Structured information files are provided in JSON and CSV and are meant to be directly readable by popular data manipulation libraries, such as PANDAS or NUMPY . Thus, they are easy to use with machine learning libraries such as SCIKIT-LEARN. It includes the description of cases in a flat JSON and the adjacency matrix for some important variables such as the body members, the cross-references between cases or the representatives. Additionally, the ready-to-use Bag-of-Words and TF-IDF representations of judgments are also available.

We also provide a normalized SQL database for more advanced queries. Finally, the portal allows to explore online the data or to interface it with external applications through a well documented REST API. More details about its implementation and user interface is provided in **??**.

ECHR-DB is guided by three core values: **reusability**, **quality** and **availability**. To reach those objectives:

- each version of the database is carefully versioned and publicly available, including the intermediate files,
- the integrality of the process and files produced are documented in details,
- the scripts to retrieve raw documents and to build the database from scratch are open-source, versioned and containerized to maximize reproducibility and trust,
- no data is manipulated by hand at any stage of the creation process to make it fully automated,
- *ECHR-DB* is augmented with rich metadata that allow to understand and use its content more easily.

5.2 Data Processing Pipeline of ECHR-DB

In this section, we discuss a full data processing pipeline used to build the integrated repository (database) of judicial cases - *ECHR-DB*.

The processing pipeline that we have used to build *ECHR-DB* is shown in Figure 5.1. The process of ingesting data is broken down into the following five steps discussed in this section, i.e.,: (1) ingesting judgment documents and basic metadata, (2) cleaning cases, (3) pre-processing documents, (4) normalizing documents, and (5) generating the repository.

5.2.1 Retreiving Judgment Documents and Basic Metadata

Using web scrapping, we retrieved all entries from HUDOC . The available data consist of basic metadata and the judgment document in natural language. Metadata include: case name,

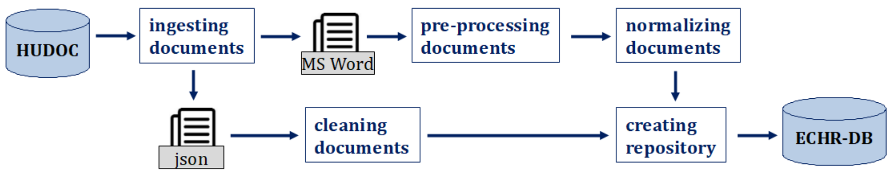


Figure 5.1: The processing pipeline for building *ECHR*.

the application number, the language used, the conclusion in a natural language, plaintiff representative, the parties, the decision body members and their role, the other cases cited in a given case, the Strasbourg Case Law mentioned in a given case, the decision date, the introduction date, the doctype branch, the external sources cites in a given case (e.g. national laws), the importance of the case, the originating body, the respondant, if there are separate opinion for a given case, the articles a given case is about (which are defined during the application). We also retrieved the judgments documents in Microsoft Word format. MS Word is a proprietary format. However, its structure in XML is easier to parse than a PDF which is the reason why we used it.

5.2.2 Cleaning Metadata

HUDOC includes cases in various languages, cases without judgments, cases without or with vague conclusions. For this reason, its content needs to be cleaned before making it available for further processing. To clean the content of HUDOC we applied a standard extract-transform-load (ETL) process [Ali 2017]. As part of the ETL process, we also parsed and formatted some raw data: parties are extracted from a case title and many raw strings are broken down into lists. In particular, a string listing articles discussed in a case are transformed into a list and a conclusion string is transformed into a slightly more complex JSON object. For instance, string *Violation of Art. 6-1; No violation of P1-1; Pecuniary damage - claim dismissed; Non-pecuniary damage - financial award* becomes the following list of elements:

```
{
   "conclusion":[
      {
         "article":"6",
         "element":"Violation of Art. 6-1",
         "type":"violation"
      },
      {
         "article":"p1",
         "element":"No violation of P1-1",
         "type":"no-violation"
      },
      {
         "element":"Pecuniary damage - claim dismissed",
         "type":"other"
      },
      {
         "element":"Non-pecuniary damage - financial award",
         "type":"other"
      }
   ]
}
```

In general, each item in the conclusion can have the following elements: (1) *article*: a number of the concerned articles, if applicable, (2) *details*: a list of additional information (a paragraph or aspect of the article), (3) *element*: a part of a raw string describing the item, (4) *mentions*: diverse mentions (quantifier, e.g., 'moderate', 'substantial aspect' or 'conditional', geographic precision, e.g., 'Kyrgyzstan' for a case filed against Russia for an extradition to Kyrgyzstan...), (5) *type*: of value *violation*, *no violation*, or *other*.

To ensure a high quality and usability of the data, the process cleaned and filtered out the cases. As a consequence, *ECHR-DB* includes: (1) only cases in English, and (2) only cases with a clear conclusion, i.e., containing at least one occurrence of *violation* or *no violation*. Therefore, we removed cases with conclusion 'Struck out of the list', 'Revision rejected' or 'Inadmissible' which represent 1847 cases. Additionally, three cases has been removed because of a broken judgment file that could not be parsed.

Finally, on top of saving the case information in a JSON file, we output a JSON file for each unique article with at least 100 associated cases[4]. Additionally, some basic statistics about the attributes are generated, e.g. the cardinality of the domain and the density (i.e. the cardinality over the total number of cases). For instance, the attribute `itemid` is unique and thus, as expected, its density is 1, as shown in the listing below.

```
"itemid":{
   "cardinal":12075,
   "density":1.0
}
```

In comparison, the field `article_` (raw string containing a list of articles discussed in a case - not kept in the final database -) and `article` (its parsed and formatted counterpart) have a density of respectively 0.26 and 0.01. This illustrates the interest of our processing method: using the raw string, the article attribute is far more unique than it should be. In reality, there are about 130 different values that are really used across the datasets and that represents all the possible combinations of articles discussed accross cases.

```
"article_":{
   "cardinal":3104,
   "density":0.26
}

"article":{
   "cardinal":131,
   "density":0.01
}
```

We denote by *descriptive features* the data and metadata that are not the judgement document.

5.2.3 Pre-processing Judgment Documents

The pre-processing task consists in parsing judgment documents in MS Word format to extract additional information and create a tree structure of a judgment file. For each case, the metadata is extended with some additional information such as the decision body, i.e. with

[4]This constant is a parameter of the script and can thus be modified for additional experimentations.

the list of persons involved in a decision, including their roles. The most important extension of a case description is the tree representation of the whole judgment document, under the field *content*. The content is described in an ordered list where each element has two fields: (1) *content* that describes the element (paragraph text or title) and (2) *elements* that represents a list of sub-elements. This tree representation eases the identification of some specific sections or paragraphs (e.g., facts or conclusion) or explore judgments with a lower granularity.

```
{
   "content":{
      "001-155097.docx":[
         {
            "content":"PROCEDURE",
            "elements":[
               {
                  "content":"1. The case originated in an application [...].",
                  "elements":[
                  ]
               },
               "..."
            ]
         },
         {
            "content":"THE FACTS",
            "elements":[
               "content": "I. THE CIRCUMSTANCES OF THE CASE",
               "elements": [
                  "..."
               ]
            ]
         },
         "...",
         {
            "content": "FOR THESE REASONS, THE COURT, UNANIMOUSLY,",
            "elements":[
               "..."
            ],
            "section_name": "conclusion"
         }
      ]
   }
}
```

Each judgment has the same structure, which includes the following sections: (1) *Procedure*,(2) *Law* and (3) *Facts*, that is further composed of: *Circumstances of the Case* and *Relevant Law*, as well as (4) *Operative Provisions*. In [Medvedeva 2020] and [Aletras 2016] it has been shown that each section has a different predictive power. The representation that we propose allows to go further to identify each individual paragraph.

5.2.4 Normalizing Documents

In this task, judgment documents (without the conclusion) are normalized by means of: (1) part-of-speech tagging, (2) tokenization, (3) stopwords removal, followed by a lemmatization, and (4) n-gram generation for $n \in \{1, 2, 3, 4\}$. The list of stopwords is provided by NLTK [**?**].

Part-of-speech tagging. This step consists in associated words with their category depending on the context. For instance, "fire" can be either a noun or a verb depending on the sentence. This is particularily useful to extract entities but also to properly clean the documents from words that carry no semantic.

Tokenization. This step consists in breaking down sentences into words. It is greatly helped by the part-of-speech tagging.

Stopwords removal. Once the part-of-speech and tokenization is done, we removed words that carry no semantic information (stopwords) such as "a", "the", etc.

***n*-gram generation**. A n-gram is a contiguous sequence of n words from a text. For intance, the sequence "new york city" provides three unigram, two 2-grams ("new york" and "york city") and one single 3-gram. For probabilistic models which cover most machine learning algorithms, n-grams are important to statistically address such ambiguous statements. The true semantic of "new york city" is lost when considering only the three separate words. In addition, the 2-grams cannot distinguish between the city of New-York and the state of New-York.

To construct the final dictionary of tokens, we use an open-source library for unsupervised topic modeling and natural language processing - GENSIM [Řehůřek 2010]. The dictionary includes the 5000 most common tokens, based on normalized documents. The number of tokens to use in the dictionary is a parameter of the script. The judgment documents are thus represented as a Bag-of-Words and TD-IDF matrices on top of the tree representation.

5.2.5 Creating Repository

Once data and metadata have been generated, to ease data exploration, notably the connections between cases, we generated adjacency matrices for the following variables: decision body, extracted application, representatives and Strasbourg case law citations. Finally, using all the generated data, we created a normalized SQL database to allow for more complex queries then what is possible to achieve on JSON or CSV.

The SQL database is composed of one main table `case`. The table `case` has several n-to-n relationships: `representative`, `party`, `decisionbodymember`, `scl` (Strasbourg Case Law) and `conclusion`. The table `conclusion` is itself composed of two n-to-n relationships: `mention` and `detail`. Finally, The table `case` has many 1-to-n relationships: `article`, `issue`, `externalapp`, `documentcollectionid`, `kpthesaurus` and `externalsources`. For each case, the tree representation of the judgment document is provided in a JSON field. The relational schema of the normalized SQL database is available in B.

5.3 Datasets For Classification

In this section, we describe the datasets extracted from *ECHR-DB* for the purpose of our experiments. The **goal of the experiments** is twofold. First, to study the predictability offered by the database. Second, to provide the additional baselines by testing the most popular machine learning algorithms for classification as previous studies used only SVM. In this chapter, we have focused the experiments on determining the outcome of new cases. The problem can be seen as a classification problem: is a law article being violated or not?

In these experimental evaluations, we are interested in answering the following four questions, in particular:

- what is the predictive power of the data in *ECHR-DB*,
- are all the articles equal w.r.t. predictability,
- are some methods performing significantly better than others, and
- are all data types (textual or descriptive) equal w.r.t. predictability?

To answer these questions, we studied three variations of the classification problem, namely: binary (described in Section 5.4, multiclass (cf. Section 5.5), and multilabel classification (cf. Section 5.6).

All the experiments are implemented using Scikit-Learn [Pedregosa 2011]. All the experiments and scripts to analyze the results as well as to generate the plots and tables are open-source and are available on a separated GitHub repository [Quemy] for repeatability and reusability.

From *ECHR-DB*, we created 11 datasets for the *binary* classification problem, one for the multiclass problem and one for the multilabel problem. Each dataset comes in different flavors, based on the descriptive features and bag-of-words representations. These different representations (listed below) allow to study the respective importance of descriptive and textual features in the predictive models build upon the datasets:

1. *descriptive features*: structured features and metadata retrieved from HUDOC or deduced from the judgment document,
2. *bag-of-words* BoW representation: based on the top 5000 tokens (normalized n-grams for $n \in \{1,2,3,4\}$),
3. *descriptive features* + *BoW*: combination of both sets of features.

For each dataset, we removed the conclusion, as well as the following metadata: the articles discussed in the case (field `articles`) and the conclusion (field `conclusion`). We discarded the articles for which there are less than 100 cases. For binary classification, each dataset corresponds to a specific article. Notice that the same case can appear in several datasets if it has in its conclusion several elements about different articles. A *label* corresponds to a violation or no violation of a specific article. The descriptive features have been one-hot encoded for non-numeric variables. A basic description of these datasets is given in Table 5.1.

Bag-of-Words is a rather naive representation that loses a substantial amount of information. However, we justify this choice by two reasons. First, so far, the studies on predicting the violation of articles for the *ECHR* cases use only the BoW representation[5]. To be able to compare the interest of the proposed data with the previous studies, we need to use the same semantic representation. Second, from a scientific point of view, it is important to provide baseline results using the most common and established methods in order to be able to

[5]At the moment the experiments have been conducted, [Chalkidis 2019] was not published. However, the results we obtained with classic Machine Learning methods are better the state of the art Deep Learning methods of [Chalkidis 2019].

Table 5.1: Datasets description for binary classification.

	# cases	min #features	max #features	avg #features	prevalence
Article 1	951	131	2834	1183.47	0.93
Article 2	1124	44	3501	2103.45	0.90
Article 3	2573	160	3871	1490.75	0.89
Article 5	2292	200	3656	1479.60	0.91
Article 6	6891	46	3168	1117.66	0.89
Article 8	1289	179	3685	1466.52	0.73
Article 10	560	49	3440	1657.22	0.75
Article 11	213	293	3758	1607.96	0.85
Article 13	1090	44	2908	1309.33	0.91
Article 34	136	490	3168	1726.78	0.64
Article p1	1301	266	2692	1187.96	0.86

Columns min, max, and avg #features indicate the minimal, maximal, and average number of features, respectively, in the cases for the representation *descriptive features and bag-of-words*.

quantify the gain of more advanced techniques. Future work will consist of investigating advanced embedding techniques that are context aware such as LSTM or BERT-like networks in a similar fashion as in [Chalkidis 2019]. In particular, we hope not only to improve the prediction accuracy by a richer semantic, but also being able to justify a decision in natural language.

For multiclass classification, there exists 18 different classes in total (the number of different articles multiplied by two possible decisions: violation or no violation). To create the multiclass dataset, we aggregated different binary classification datasets by removing the cases present in several datasets. When a case has several article in its conclusion, we kept as label the first one given by HUDOC. A description of these datasets is given in Table 5.2. Notice that this multiclass setting differs from [Chalkidis 2019] as a non-violation of a given article is also a label to be identified. Also, after applying our merging method, articles 13 and 34 had less than 100 cases and have been discarded.

Multiclass and multilabel datasets are not obtained by simply merging the binary datasets. As the corpus of judgment documents differs between the datasets, the most frequent n-grams also change. It implies that for a given case, its BoW representation is different in the binary, multiclass and multilabel datasets. The descriptive features are, however, not modified.

For multilabel classification, there exists 22 different labels and the main difference with the multiclass is that there is no need to remove cases that appear in multiple binary classification datasets. The labels are simply stacked. Table 5.3 summarizes the dataset composition. Given the class imbalance for all datasets, the reader might wonder why they are not rebalanced. We justify this choice a posteriori by the analysis of the confusion matrices and learning curve (Section 5.4.2) and discuss it further in Section 7.3.

Figure 5.2 shows the labels repartition among the multiclass and multilabel datasets. Fig-

Table 5.2: Dataset description for the multiclass dataset.

	# cases	violation	no-violation	prevalence
Article 1	310	280 (0.039)	30 (0.004)	0.90
Article 2	267	230 (0.032)	37 (0.005)	0.86
Article 3	775	676 (0.095)	99 (0.014)	0.87
Article 5	791	689 (0.097)	102 (0.014)	0.87
Article 6	3491	3143 (0.441)	348 (0.049)	0.90
Article 8	623	457 (0.064)	166 (0.023)	0.73
Article 10	413	315 (0.044)	98 (0.014)	0.76
Article 11	110	92 (0.013)	18 (0.003)	0.84
Article p1	353	294 (0.041)	59 (0.008)	0.83

For each article the following features are indicated: (1) the number of cases, (2) the number of cases labeled as violated and not violated (in parenthesis, the prevalence w.r.t. the whole dataset), and (3) the prevalence per article.

Table 5.3: Dataset description for the multilabel dataset.

	# cases	violation	no-violation
Article 1	951	882 (0.082)	69 (0.006)
Article 2	1124	1017 (0.095)	107 (0.010)
Article 3	2573	2295 (0.214)	278 (0.026)
Article 5	2292	2081 (0.194)	211 (0.020)
Article 6	6891	6152 (0.574)	739 (0.069)
Article 8	1289	940 (0.088)	349 (0.033)
Article 10	560	418 (0.039)	142 (0.013)
Article 11	213	180 (0.017)	33 (0.003)
Article 13	1090	997 (0.093)	93 (0.009)
Article 34	136	87 (0.008)	49 (0.005)
Article p1	1301	1120 (0.105)	181 (0.017)

For each article the following features are indicated: (1) the number of cases, (2) the number of cases labeled as violated and not violated. In parenthesis, the label prevalence w.r.t. the whole dataset.

ure 5.3 shows the histogram of label numbers and cases per label.

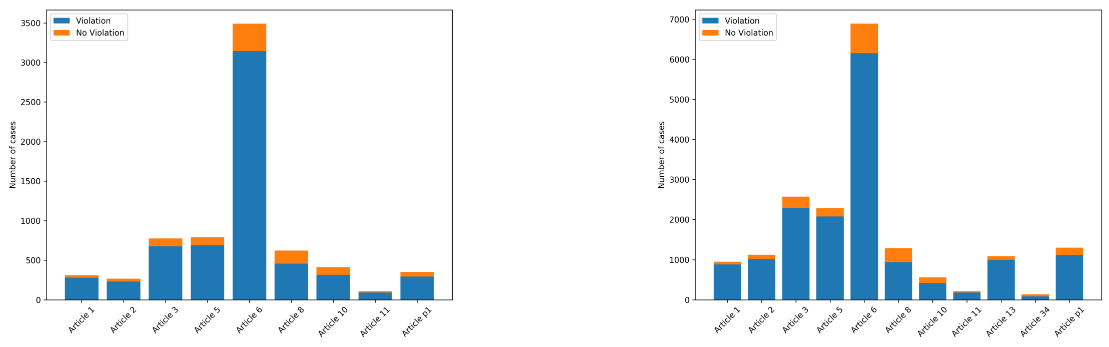


Figure 5.2: The number of cases depending on the article and the outcome for the multiclass dataset (left) and multilabel dataset (right).

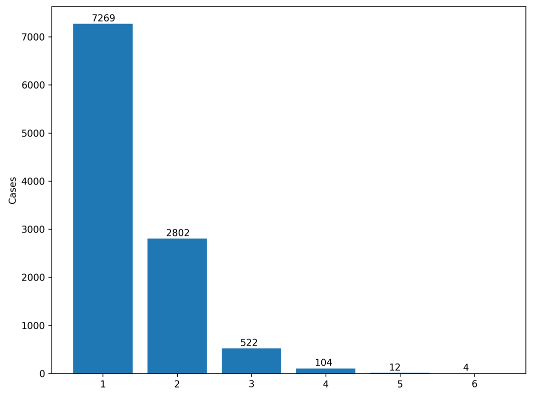


Figure 5.3: The number of cases depending on the number of labels for the multilabel dataset.

5.4 Experiments: Binary Classification

5.4.1 Protocol

We compared 13 standard classification algorithms provided by SCIKIT-LEARN, namely: AdaBoost with Decision Tree, Bagging with Decision Tree, Naive Bayes (Bernoulli and Multinomial), Decision Tree, Ensemble Extra Tree, Extra Tree, Gradient Boosting, K-Neighbors, SVM (Linear SVM, RBF SVM), Neural Network (Multilayer Perceptron), and Random Forest.

For each article, we used the following three flavors: (1) descriptive features only, (2) bag-of-words only, and (3) descriptive features combined with bag-of-words. For each method, each article, and each flavor, we performed a 10-fold cross-validation with stratified sample, for a total of 429 validation procedures. Due to this important amount of experimental settings, we discarded the TF-IDF representation. For the same reason, we did not perform any hyperparameter tuning at this stage and used the default parameters of the SCIKIT-LEARN implementation.

To evaluate the performances, we reported some standard performance metrics, namely: (1) accuracy, (2) F_1-score, and (3) MCC. We recall the definition of these metrics. Denoting by: TP - the number of true positives, TN - the number of true negatives, FP - the number of false positives, and FN - the number of false negative. For cthe sake of ompleteness, we include the standard definitions of these metrics.

$$\mathrm{ACC} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}} \tag{5.1}$$

$$\mathrm{F}_1 = \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}} \tag{5.2}$$

$$\mathrm{MCC} = \frac{\mathrm{TP}\times\mathrm{TN}-\mathrm{FP}\times\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}} \tag{5.3}$$

The values of accuracy, F_1-score, and MCC are within ranges $[0,1]$, $[0,1]$ and $[-1,1]$, respectively (values closer to 1 are better). MCC has been shown to be more informative than other metrics derived from the confusion matrix [Chicco 2017], in particular with imbalanced datasets.

Additionally, we report the learning curves to study the limit of the model space. The learning curves are obtained by plotting the accuracy as a function of a training set size, for both the training and the test sets. The learning curves help to understand if a model underfits or overfits and thus, shape future axis of improvements to build better classifiers.

To find out what type of features are the most important w.r.t. predictability, we used a *Wilcoxon signed-rank*, i.e., a non-parametric paired difference test at 5%, in order to compare the accuracy obtained on bag-of-words representation to the one obtained on the bag-of-words combined with the descriptive features. Given two paired samples, the null hypothesis assumes that the difference between the pairs follows a symmetric distribution around zero. The test is used to determine if the changes in the accuracy are significant when the descriptive features are added to the textual features.

5.4.2 Results

Table 5.4 shows the best accuracy obtained for each article as well as the method and the flavor of the dataset. For all articles, the best accuracy obtained is higher than the prevalence. Linear SVC offers the best results on 4, out of 11 articles. Gradient Boosting accounts for 3, out 11 articles and Ensemble Extra Tree accounts for 2 articles.

The standard deviation is rather low and ranges from 1% up to 4%, at the exception of article 34, for which it is equal to 9%. This indicates a low variance for the best models. The accuracy ranges from 75.86% to 98.32%, with the average of 94.43%. The micro-average that ponders each result by the dataset size is 96.44%. In general, the datasets with higher accuracy are larger and more imbalanced. For the datasets being highly imbalanced, with a prevalence from 0.64 to 0.93, other metrics may be more suitable to appreciate the quality of the results. In particular, the micro-average could simply be higher due to the class imbalance rather than the availability of data.

Regarding the flavor, 8 out 10 best results are obtained on descriptive features combined to bag-of-words. BoW only is the best flavor for article 10, whereas descriptive features are the

Table 5.4: The best accuracy obtained for each article. Standard deviation is reported between brackets.

Article	Accuracy	Method	Flavor
Article 1	0.9832 (0.01)	Linear SVC	Descriptive features and Bag-of-Words
Article 2	0.9760 (0.02)	Linear SVC	Descriptive features and Bag-of-Words
Article 3	0.9588 (0.01)	BaggingClassifier	Descriptive features and Bag-of-Words
Article 5	0.9651 (0.01)	Gradient Boosting	Descriptive features and Bag-of-Words
Article 6	0.9721 (0.01)	Linear SVC	Descriptive features and Bag-of-Words
Article 8	0.9542 (0.03)	Gradient Boosting	Descriptive features and Bag-of-Words
Article 10	0.9392 (0.04)	Ensemble Extra Tree	Bag-of-Words only
Article 11	0.9671 (0.03)	Ensemble Extra Tree	Descriptive features and Bag-of-Words
Article 13	0.9450 (0.02)	Linear SVC	Descriptive features only
Article 34	0.7586 (0.09)	AdaBoost	Descriptive features only
Article p1	0.9685 (0.02)	Gradient Boosting	Descriptive features and Bag-of-Words
Average	0.9443		
Micro average	0.9644		

Legend: *desc* - descriptive features; *BoW* - bag of words.

best for articles 13 and 34. This seems to indicate that combining information from different sources improves the overall results.

Figure 5.4 displays the normalized confusion matrix for the best methods on article 1 and 13. Similar results are observed for all the other articles. The normalization is done per line and it allows to quickly figure out how the true predictions are balanced for both classes. As expected due to the prevalence, true negatives are extremely high, ranging from 0.82 to 1.00, with an average of 97.18. On the contrary, the true positive rate is lower, ranging from 0.47 to 0.91. For most articles, the true positive rate is higher than 80% and it is lower than 50% only for article 34. This indicates that despite the classes being highly imbalanced, the algorithms are capable of producing models that are fairly balanced.

Additionally, we provide the values of the MCC in Table 5.5 and F1-score in Table 5.6. As results are similar between both indicators, we analyze only the MCC. The MCC is generally superior to the accuracy because it takes into account the class prevalence. Therefore, it is a far better metric to estimate model quality than accuracy. The MCC ranges from 0.4918 - on article 34 to 0.8829 - on article 10. The best score is not obtained by the same article as for the accuracy (article 10 achieved 93% accuracy, below the average). Interestingly, the MCC reveals that the performances on article 34 are rather poor in comparison to the other articles and close to the performance on article 13. Surprisingly, the best method is not Linear SVC anymore (best on 3 articles) but Gradient Boosting (best on 4 articles). While the descriptive features were returning the best results for two articles, according the MCC, it reaches the best score only for article 34.

Once again, the micro-average is higher than the macro-average. As the MCC takes into account class imbalance, it supports the idea that adding more cases to the training set could still improve the result of these classifiers. This will be confirmed by looking at the learning curves.

Tables 5.7 and 5.8 rank the methods according to the average accuracy, F1-score and MCC

Table 5.5: The best MCC obtained for each article.

Article	MCC	Method	Flavor
Article 1	0.8654	Linear SVC	Descriptive features and Bag-of-Words
Article 2	0.8609	Linear SVC	Descriptive features and Bag-of-Words
Article 3	0.7714	BaggingClassifier	Descriptive features and Bag-of-Words
Article 5	0.7824	Gradient Boosting	Descriptive features and Bag-of-Words
Article 6	0.8488	Linear SVC	Descriptive features and Bag-of-Words
Article 8	0.8829	Gradient Boosting	Descriptive features and Bag-of-Words
Article 10	0.8411	Gradient Boosting	Bag-of-Words only
Article 11	0.8801	Ensemble Extra Tree	Descriptive features and Bag-of-Words
Article 13	0.5770	Ensemble Extra Tree	Bag-of-Words only
Article 34	0.4918	AdaBoost	Descriptive features only
Article p1	0.8656	Gradient Boosting	Descriptive features and Bag-of-Words
Average	0.7879		
Micro average	0.8163		

The flavor and method achieving the best score for both metrics are the same for every article. Legend: *desc* - descriptive features; *BoW* - bag of words.

Table 5.6: The best F1-Score obtained for each article.

Article	F1 score	Method	Flavor
Article 1	0.9819	Linear SVC	Descriptive features and Bag-of-Words
Article 2	0.9757	Linear SVC	Descriptive features and Bag-of-Words
Article 3	0.9558	BaggingClassifier	Descriptive features and Bag-of-Words
Article 5	0.9637	Gradient Boosting	Descriptive features and Bag-of-Words
Article 6	0.9713	Linear SVC	Descriptive features and Bag-of-Words
Article 8	0.9536	Gradient Boosting	Descriptive features and Bag-of-Words
Article 10	0.9372	Ensemble Extra Tree	Bag-of-Words only
Article 11	0.9672	Ensemble Extra Tree	Descriptive features and Bag-of-Words
Article 13	0.9360	Linear SVC	Descriptive features only
Article 34	0.7526	AdaBoost	Descriptive features only
Article p1	0.9671	Gradient Boosting	Descriptive features and Bag-of-Words
Average	0.9420		
Micro average	0.9627		

The flavor and method achieving the best score for both metrics are the same for every article. Legend: *desc* - descriptive features; *BoW* - bag of words.

Table 5.7: The overall ranking of methods according to the average accuracy obtained for every article.

Method	Accuracy	Micro Accuracy	Rank
Ensemble Extra Tree	0.9420	0.9627	1
Linear SVC	0.9390	0.9618	2
Random Forest	0.9376	0.9618	3
BaggingClassifier	0.9319	0.9599	4
Gradient Boosting	0.9309	0.9609	5
AdaBoost	0.9284	0.9488	6
Neural Net	0.9273	0.9535	7
Decision Tree	0.9181	0.9419	8
Extra Tree	0.8995	0.9275	9
Multinomial Naive Bayes	0.8743	0.8907	10
Bernoulli Naive Bayes	0.8734	0.8891	11
K-Neighbors	0.8670	0.8997	12
RBF SVC	0.8419	0.8778	13
Average	0.9086	0.9335	

performed on all articles. For each article and method, we kept only the best accuracy among the three dataset flavors.

Surprisingly, the best method is neither Linear SVC nor Gradient Boosting, but Ensemble Extra Tree. Random Forest and Bagging with Decision Tree are the second and third ones, respectively, and they never achieved the best result on any article. It simply indicates that these methods are more consistent across the datasets than Linear SVC and Gradient Boosting.

In Table 5.9 is reported the F1-Score on article 2 for all methods. In general, all methods performs above the prevalence except for Naive Bayes methods, KNN and RBF SVM. The score for descriptive features only is always lower than the Bag-of-Words representation, but it is not clear if descriptive features improve the results. For instance, the result on Random Forest is higher for Bag-of-Words only and does not appear to be significant for Neural Network. Similar results are observed on all articles (see Supplementary Material).

Figure 5.5 displays the learning curves obtained for the best methods on articles 10 and 11. The training error becomes (near) zero on every instance after the model has been trained with only few examples, except for article 13 and 34. The test error converges rather fast and remains relatively far from the training error, which is synonym of high bias. The two above-mentioned observations indicate underfitting. Similar results are observed for all methods. Usually, more training examples would help, but since the datasets are exhaustive w.r.t. the European Court of Human Rights cases, this is not possible. As a consequence, we recommend using a more complex model space and hyperparameter tuning. In particular, as mentioned above, the usage of more advanced embedding techniques is an obvious way to explore. Finally, an exploratory analysis of the datasets may also help in removing some noise and finding the best predictors.

On article 13 and 34, the bias is also high, and variance relatively higher than for the

Table 5.8: The overall ranking of methods according to the average F1-Score obtained for every article.

Method	F1 score	Rank
Ensemble Extra Tree	0.9364	1
Linear SVC	0.9321	2
Random Forest	0.9297	3
BaggingClassifier	0.9285	4
Gradient Boosting	0.9260	5
AdaBoost	0.9259	6
Neural Net	0.9189	7
Decision Tree	0.9177	8
Extra Tree	0.8970	9
Multinomial Naive Bayes	0.8750	10
Bernoulli Naive Bayes	0.8716	11
K-Neighbors	0.8329	12
RBF SVC	0.7725	13
Average	0.8972	0.9266

Table 5.9: F1-Score for all methods on article 2.

Prev=0.9048	F1 score - Article 2		
	desc	BoW	both
AdaBoost	0.9253	0.9607	0.9521
BaggingClassifier	0.9272	0.9642	0.9689
Bernoulli Naive Bayes	0.8785	0.8126	0.8427
Decision Tree	0.9169	0.9590	0.9488
Ensemble Extra Tree	0.9274	**0.9748**	0.9736
Extra Tree	0.9123	0.9400	0.9436
Gradient Boosting	0.9114	0.9728	0.9692
K-Neighbors	0.9057	0.8670	0.8670
Linear SVC	**0.9391**	0.9707	**0.9757**
Multinomial Naive Bayes	0.8619	0.9067	0.9040
Neural Net	0.9380	0.9585	0.9591
RBF SVC	0.8596	0.8596	0.8596
Random Forest	0.9103	0.9738	0.9684

.

other articles, clearly indicating the worst possible case. Again, adding more examples is not an option. However, if we assume that the process of deciding if there is a violation or not is the same, independently of the article, a solution might be a transfer learning, to leverage what can be learned from the other articles. We let this research axis for future work.

Finally, we used a Wilcoxon signed-rank test at 5% to compare the accuracy obtained on the BoW representation to the one obtained on the BoW combined with descriptive features. The difference between the samples has been found to be significant only for article 6 and article 8. The best result obtained on BoW is improved by adding descriptive features for every article. However, statistically, for a given method, adding descriptive features does not improve the result. Additionally, we performed the test per method. The results are significant for every method.

In conclusion, the datasets for binary classification demonstrated a strong predictability power. Apart from article 13 and 34, each article seems to provide similar results, independently of the relatively different prevalence. The accuracy is rather high and a more informative metric, such as MCC, shows that there are still margins of improvements. Hyperparameter tuning [Quemy 2020a] is an obvious way to go, and this preliminary work has shown that good candidates for fine tuning are Ensemble Extra Tree, Linear SVC, and Gradient Boosting as shown by Tables 5.5 and 5.4.

5.4.3 Discussion

To sum up, we achieved an average accuracy of 94% which is respectively 15pp and 19pp higher than best results reported in [Aletras 2016] and [Medvedeva 2020]. The average F1-Score obtained is 81.6 which is similar to the 82.0 obtained in **??**. However, it worth to notice that we used traditional Machine Learning that requires less computational effort and time than state of the art Deep Learning method such as BERT. The size of the dataset does matter since we showed that the model underfits thanks to Figure 5.5. Also, we showed that SVM is far from being the best method for all articles. However, such a huge gap cannot be explained only by those two factors.

In our opinion, the main problem with the previous studies is that the authors rebalanced their datasets. As those datasets were highly imbalanced, they used undersampling, which resulted in a very small training dataset. Most likely, the training dataset was not representative enough of the feature space which leads to underfitting (even more than in our experiments). They justified that rebalancing was necessary to ensure that the classifier was not biased towards a certain class. For this reason, we argue that they modified the label distribution. As some classification methods rely on the label distribution to learn, they introduce themselves a prior shift [Lemberger 2020]. In general, rebalancing is necessary only when, indeed, the estimator is badly biased. It is true that the accuracy is meaningless on imbalanced datasets but we can still control the quality of the model using a collection of more robust indicators, including among others: F1-score, MCC, and normalized confusion matrices.

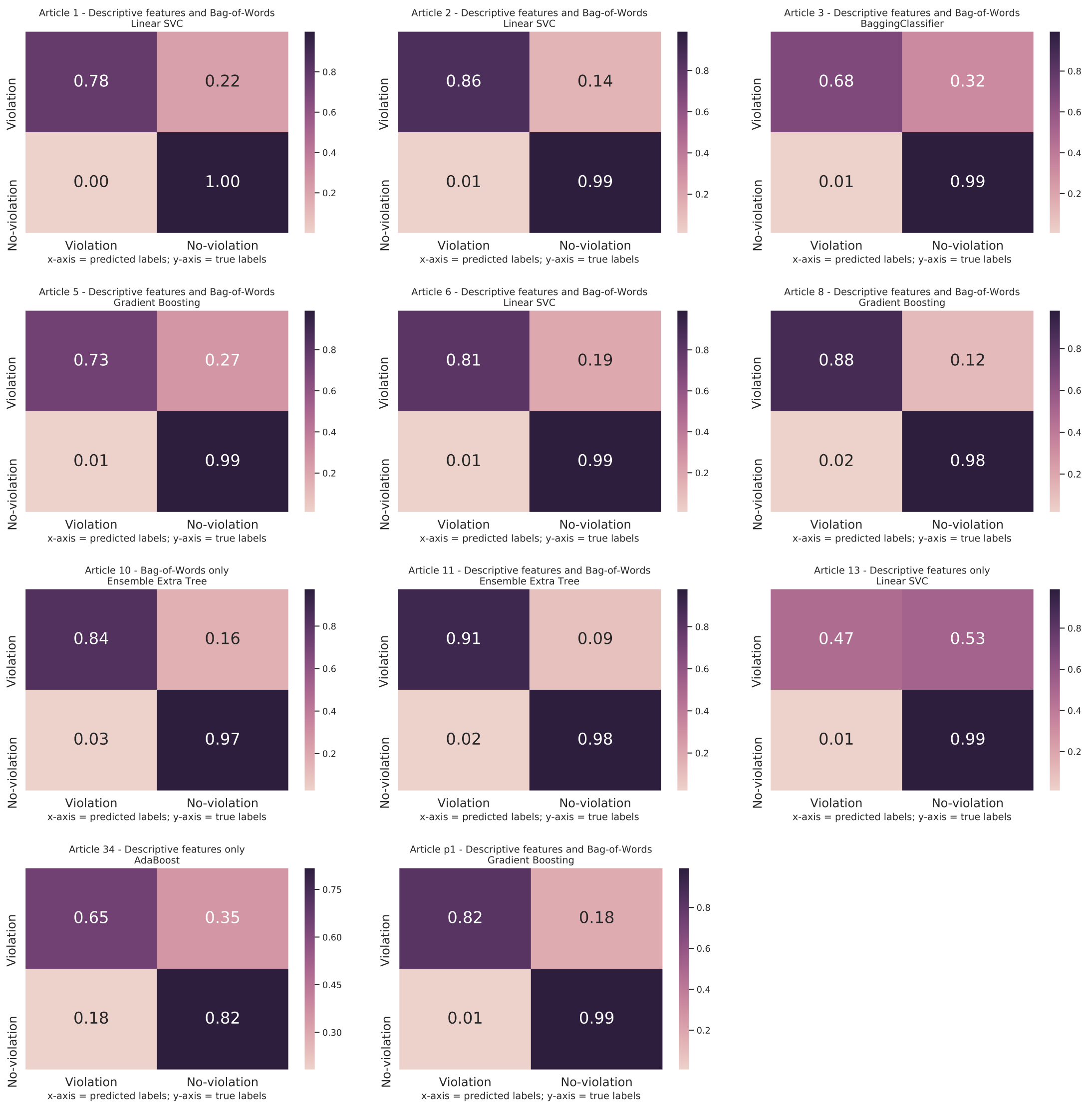


Figure 5.4: Normalized Confusion Matrices for the best methods as described by Table 5.4.

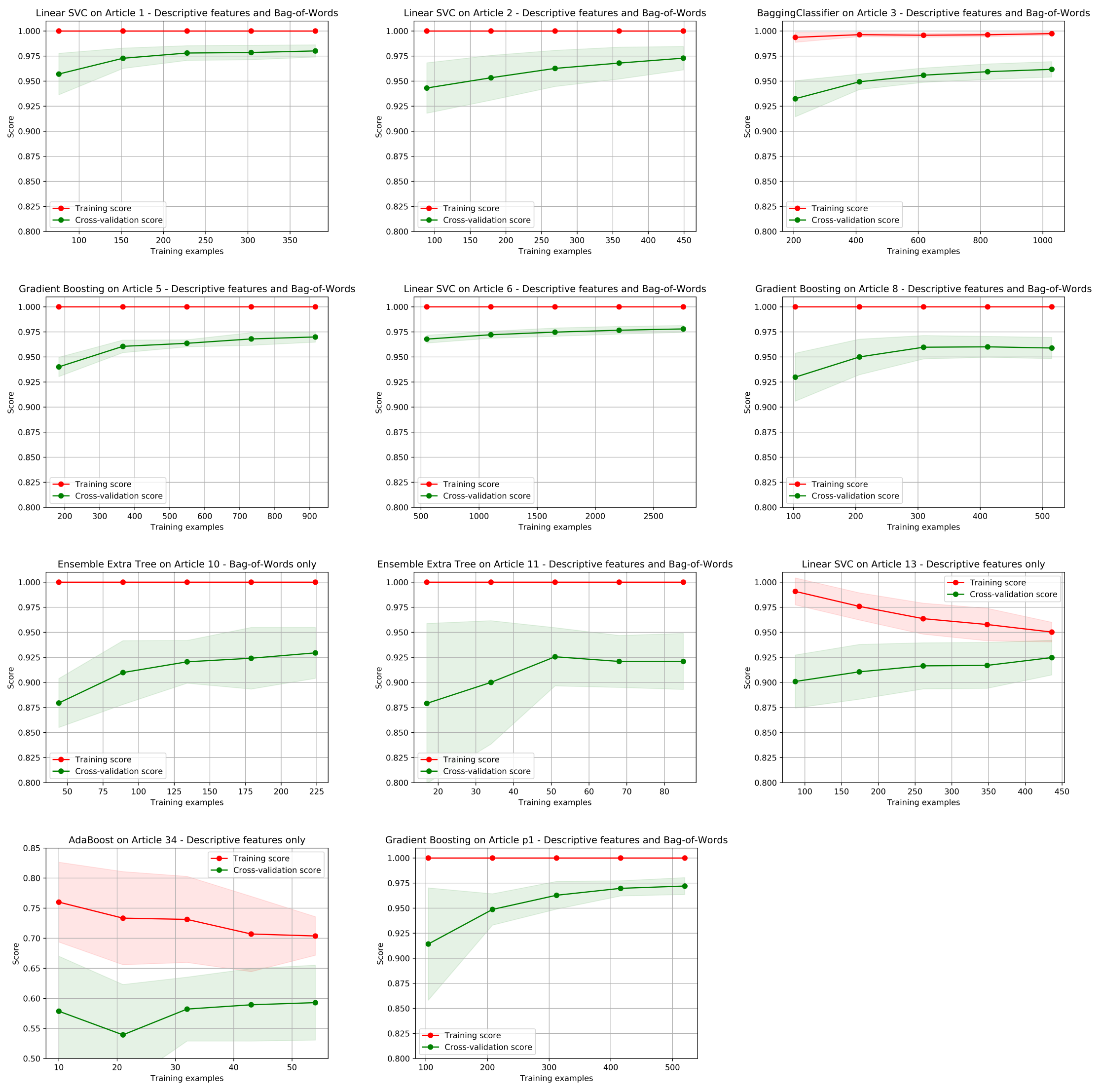


Figure 5.5: Learning Curves for the best methods as described by Table 5.4.

To sum up, the approach discussed here is more neutral in the sense we do not change the label distribution, and it still offers a robust classifier. This is confirmed in Section 5.6 on multilabel classification.

Our in-depth experimental evaluation has demonstrated that the textual information provides better results than descriptive features alone, but the addition of the descriptive feature improved in general the result of the best method (obtained among all methods). We emphasize the best method because for a given method adding the descriptive feature are not significantly improving the results.

Another way of improving the results is to tune the different phases of the dataset generations. In particular, our preliminary work reported in [Quemy 2020a] has shown that 5000 tokens and 4-grams might not be enough to take the best out of the documents. It might seem surprising, but the justice language is codified and standardized in a way that n-grams for large n might contain better predictors for the outcome.

5.5 Experiments: Multiclass Classification

In the previous section, we showed that most methods could obtain an accuracy higher than the dataset prevalence, and more generally, good performance metrics. Usually algorithms for binary classification adapt relatively well to multiclass problems, however, in the case of *ECHR-DB*, the labels come by pair (violation or no-violation of a given article), which may confuse the classifiers.

The experimental protocol for these experiments is identical to the one of the previous section. For computational purposes, we dropped the two worst classifiers from the the binary datasets, namely RBF SVM and KNN.

5.5.1 Results

Table 5.10 presents the accuracy obtained on the multiclass dataset. The best accuracy for descriptive features only and BoW only is linear SVM with 91.41% and 91.36% correctly labeled cases, respectively. This is aligned with the results obtained on binary datasets. However, the top score of 94.99% is obtained by Bagging Classifier that only ranked 4th on binary datasets. In other words, SVM ranked first on two types of features individually, but the improvement of combining the features is lower than the one obtained by Bagging Classifier. The same can be observed for Gradient Boosting that outperforms SVM. Except from Ada Boost, the standard deviation is mostly lower than 1%.

For most methods, BoW only scores better than descriptive features. This observation is reversed by looking at the MCC provided by Table 5.11. However, for both indicators, combining both types of features increases performances at the notable exceptions of Extra Tree, Multinomial Naive Bayes, and Ada Boost with Decision Tree. This highly contrasts with the binary setting for which descriptive features were quantitatively far below textual features, in particular w.r.t. MCC. For binary datasets, *descriptive features only* was mostly scoring below the *BoW only*, for any article and any method (cf. GitHub[6]). On top of that, taking only the

[6]https://echr-od.github.io/ECHR-OD_project_supplementary_material/

Table 5.10: The accuracy obtained for each method on the multiclass dataset.

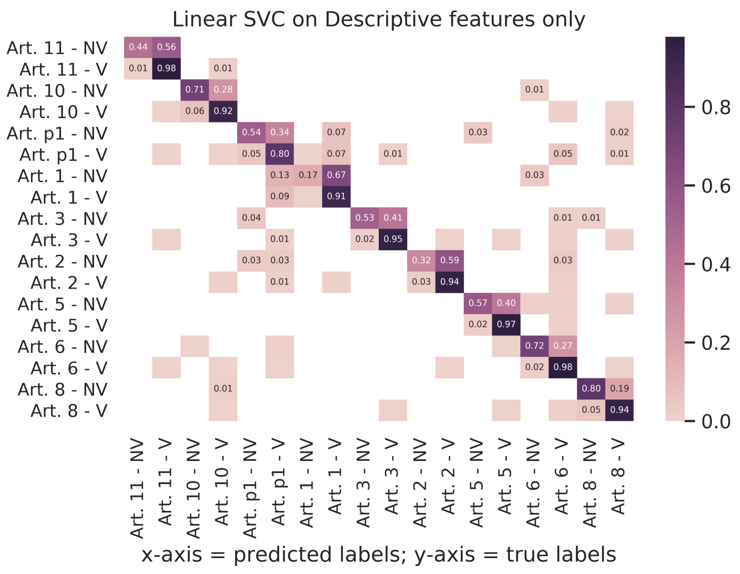
	Accuracy - Multiclass		
	desc	BoW	both
AdaBoost	0.7789 (0.04)	0.5451 (0.15)	0.5720 (0.04)
BaggingClassifier	0.8998 (0.01)	0.8794 (0.01)	**0.9499** (0.01)
Bernoulli Naive Bayes	0.5096 (0.01)	0.7516 (0.01)	0.7464 (0.01)
Decision Tree	0.8897 (0.02)	0.8457 (0.01)	0.9434 (0.01)
Ensemble Extra Tree	0.8788 (0.01)	0.8904 (0.01)	0.9195 (0.01)
Extra Tree	0.7776 (0.03)	0.7164 (0.03)	0.7458 (0.02)
Gradient Boosting	0.8911 (0.01)	0.8904 (0.01)	0.9494 (0.01)
Linear SVC	**0.9141** (0.01)	**0.9136** (0.01)	0.9420 (0.01)
Multinomial Naive Bayes	0.7980 (0.01)	0.7784 (0.01)	0.7829 (0.01)
Neural Net	0.8813 (0.01)	0.9072 (0.01)	0.9231 (0.01)
Random Forest	0.8669 (0.01)	0.8825 (0.01)	0.9125 (0.01)

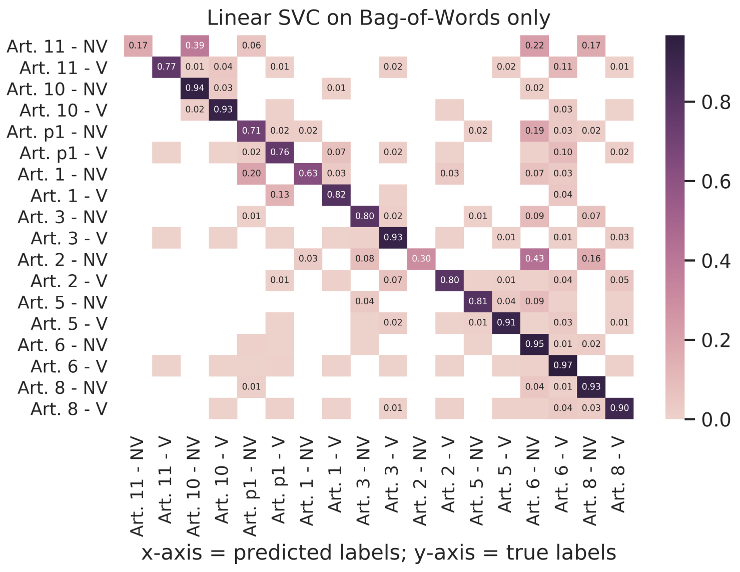
best result per flavor, the *descriptive features* score better than purely textual features. The explanation can be found by studying the confusion matrix.

Figure 5.6 shows the normalized confusion matrix for Linear SVM. The normalization has been done per line, i.e. each line represents the distribution of cases according to their ground truth. For instance, on *descriptive features only*, for class *Article 11, no-violation*, 44% only were correctly classified and 56% assigned to a violation of article 11. The perfect classifier should thus have a diagonal of 1. The diagonal is equivalent to the recall for the corresponding class and the average the diagonal terms is the balanced accuracy [Brodersen 2010].

Flavor *descriptive features only* has a sparser normalized confusion matrix than the counterpart with BoW. The fact that the flavor *descriptive features only* returns a lower accuracy is explained by looking at the 2x2 blocks on the diagonal. These blocks are the normalized confusion matrix of the subproblem restricted to find a specific article. For instance, 100% of non-violation of article 10 has been labeled in one of the two classes related to article 10 (99% for a violation). In general, the classifiers on *descriptive features only* are good at identifying the article but generates a lot of false negatives, most likely due to the imbalance between violation and non-violation labels for a given article. Adding BoW to the case representation slightly lowers the accuracy on average but largely rebalances the 2x2 blocks on the diagonal. On the other hand, it seems that the textual information does not hold enough information to identify the article, which explain why classifiers perform in general lower on BoW only.

We performed two Wilcoxon signed-rank tests: first between the samples of results on BoW and BoW + descriptive features, then between descriptive features and BoW + descriptive features. The first result is clearly significant while the second is not significant, comforting us in the idea that for the multiclass domain, Bag-of-Words flavor alone is unlikely to give good results compared to descriptive features.





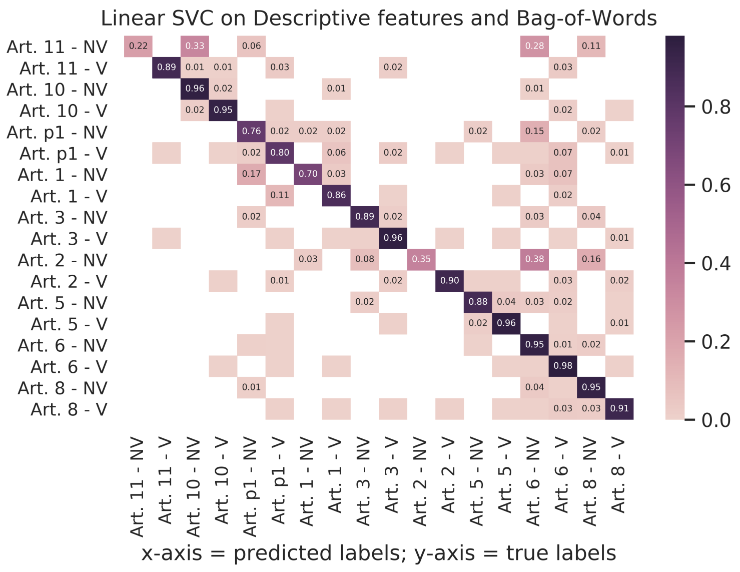


Figure 5.6: Normalized Confusion Matrix for multiclass dataset.
The normalization is performed per line. A white block indicates that no element has been predicted for the corresponding label. Percentages are reported only if above 1%.

Table 5.11: Matthew Correlation Coefficient obtained for each method on the multiclass dataset.

	MCC - Multiclass		
	desc	BoW	both
AdaBoost	0.7171 (0.04)	0.4416 (0.14)	0.4580 (0.05)
BaggingClassifier	0.8700 (0.02)	0.8435 (0.02)	**0.9353** (0.01)
Bernoulli Naive Bayes	0.2750 (0.02)	0.6845 (0.01)	0.6664 (0.01)
Decision Tree	0.8572 (0.02)	0.8004 (0.02)	0.9268 (0.01)
Ensemble Extra Tree	0.8419 (0.01)	0.8576 (0.01)	0.8956 (0.01)
Extra Tree	0.7103 (0.04)	0.6343 (0.04)	0.6700 (0.02)
Gradient Boosting	0.8580 (0.01)	0.8568 (0.01)	0.9346 (0.01)
Linear SVC	**0.8886** (0.01)	**0.8883** (0.01)	0.9251 (0.01)
Multinomial Naive Bayes	0.7323 (0.01)	0.7193 (0.01)	0.7190 (0.02)
Neural Net	0.8452 (0.01)	0.8795 (0.01)	0.9001 (0.01)
Random Forest	0.8262 (0.01)	0.8472 (0.01)	0.8864 (0.01)

Legend: *desc* - descriptive features; *BoW* - bag of words

5.5.2 Discussion

The main conclusion to draw from the multiclass experiment is that the descriptive features are excellent at identifying the article while the text offers more elements to predict the outcome. It is not surprising since descriptive features are available before the judgment while the judgment itself discuss specifically the violation or not. However, the fact that descriptive features hold the best predictive power supports the realism theory which considers that judges do not simply apply objective and neutral legal reasonings. Indeed, descriptive features has little to do with legal arguments and facts but more about judges and parties. This can be tempered by the fact that a judge might be specialized in cases related to a specific article and country, which would explain why judges and parties are strong predictors. To rule for one or the other hypothesis, we plan to explore the network of decision body members, representatives and countries provided with *ECHR-DB* in future work.

Using only BoW leads to the worst possible results, while adding textual information to the descriptive features slightly increases the accuracy and has a strong beneficial effect on discriminating between violations and non-violations. This is quite in opposition with the conclusion drawn from the experiments on the binary datasets where the textual representation clearly overperformed while the descriptive features had only a marginal effect.

This indicates that it might be more interesting to create a two-stage classifier, namely: a multiclass classifier - for determining an article based on descriptive features, followed by an article-specific classifier - for determining if the article is violated or not. Over-sampling techniques to deal with imbalanced classes constitute another axis of improvement to explore in future work.

Finally, we conclude that the benefit of combining sources of information is not monotonic: the best scoring method on individual types of features might not be the best method overall.

5.6 Experiments: Multilabel Classification

A multilabel dataset generalizes the multiclass one in a way there is not only one article to identify before predicting the outcome, but an unknown number. In the ECHR, application must specified the articles that are to be discussed. This articles are taking into consideration to judge if a case is admissible or not. The multilabel dataset reflects a real-life situations in which a lawyer might advise the paintiff on the articles to be added in her application, as well as the probability of violation or not. In such a case, on top of analyzing the usual performance metrics, we would like to quantify how good are the methods to identify all the articles in each case. From the multiclass results, it is expected that the textual information alone will provide the lowest results among all flavors which would be an interesting results since the judgment is obviously not known at the moment of filling the application.

5.6.1 Protocol

Appreciating the results of a multilabel classifier is not as easy as in the binary or multiclass case. For instance, having wrongly added one label to 100 cases is not exactly the same as adding 100 wrong labels to a single case. Similarly, being able to correctly predict at least one correct label per case, is not the same as predicting all good labels for a fraction of the cases, even if the total amount of correct labels is the same in both scenarios. The distributions of ground truth and predicted labels among the dataset are important for evaluating the quality of a model.

For this reason, we reported the following multilabel-specific metrics: subset accuracy, precision, recall, F_1-score, Hamming loss, and the Jaccard index. The subset accuracy is the strictest metric since it measures the percentage of samples such that all the labels are correctly predicted. It does not account for partly correctly labeled vectors. The Hamming loss calculates the percentage of wrong labels in the total number of labels. The Jaccard index measures the number of correctly predicted labels divided by the union of prediction and true labels. Let T (resp. P) denotes the true (resp. predicted) labels, n - the size of the sample, and l - the number of possible labels and Y_i the set of correct labels for case i. Then the standard definitions of these metrics are defined as follows:

$$\text{ACC} = \frac{1}{n}\sum_{i=1}^{n} I(Y_i = \bar{Y}_i) \tag{5.4}$$

$$\text{RECALL} = \frac{\text{T} \cap \text{P}}{\text{T}} \tag{5.5}$$

$$\text{PRECISION} = \frac{\text{T} \cap \text{P}}{\text{P}} \tag{5.6}$$

$$\text{F}_1 = \frac{\text{RECALL} \times \text{PRECISION}}{\text{RECALL} + \text{PRECISION}} \tag{5.7}$$

$$\text{HAMMING} = \frac{1}{nl}\sum_{i=1}^{n} \text{xor}(y_{i,j}, \bar{y}_{i,j}) \tag{5.8}$$

$$\text{JACCARD} = \frac{\text{T} \cap \text{P}}{\text{T} \cup \text{P}} \tag{5.9}$$

Table 5.12: The accuracy of each method on the multilabel dataset.

	Multilabel		
	desc	BoW	both
Decision Tree	**0.7837** (0.02)	0.6612 (0.02)	**0.7966** (0.02)
Ensemble Extra Tree	0.7252 (0.03)	0.6643 (0.02)	0.6954 (0.02)
Extra Tree	0.5978 (0.03)	0.5410 (0.02)	0.5407 (0.03)
Neural Net	0.6914 (0.03)	**0.6745** (0.02)	0.7159 (0.02)
Random Forest	0.7061 (0.03)	0.6438 (0.02)	0.6674 (0.02)

Table 5.13: The precision of each method on the multilabel dataset.

	Multilabel		
	desc	BoW	both
Decision Tree	0.8470	0.7780	0.8705
Ensemble Extra Tree	**0.8808**	**0.8957**	**0.9147**
Extra Tree	0.7202	0.6786	0.6780
Neural Net	0.8674	0.8719	0.8995
Random Forest	0.8698	0.8929	0.9120

We are interested in quantifying how much a specific article was properly identified, as well as how many cases with a given number of labels are correctly labeled, taking into account their respective prevalence in the dataset reported in Figure 5.3. Indeed, about 70% of cases in the multilabel dataset have only one label such that a classifier assigning only one label to each case could reach about 70% of subset accuracy.

Not all binary classification algorithms can be extended for the multilabel problem. Therefore, in our experiments we used the following five algorithms: Extra Tree, Decision Tree, Random Forest, Ensemble Extra Tree, and Neural Network. As previously, a 10-fold cross-validation has been performed on each flavor.

5.6.2 Results

The accuracy is reported in Table 5.12 and it shows that Decision Tree outperforms with 79.66% of cases that have been totally correctly labeled. Similarly to the multiclass setting, descriptive features provide a better result than BoW. Decision Tree scores also the best for the F_1-score (Table 5.15) and recall (Table 5.14). However, Ensemble Extra Tree overperforms Decision Tree w.r.t. precision (Table 5.13). Decision Tree provides the best results for the *strict* metrics (highest accuracy and lowest Hamming loss) but also on more permissive metrics (best F_1-score and Jaccard index). Therefore, Decision Tree is clearly the top classifier for multilabel which is a bit surprising, since it ranked 8th over the binary datasets and 3rd on the multiclass one.

As expected, the BoW flavor provides the worst possible results. Similarly to the experiments on the multiclass dataset, the textual information is inefficient for identifying the article.

Figure 5.7 shows the accuracy, recall, and precision depending on the number of labels

Table 5.14: The recall of each method on the multilabel dataset.

	Multilabel		
	desc	BoW	both
Decision Tree	**0.8482**	**0.7688**	**0.8611**
Ensemble Extra Tree	0.7635	0.7049	0.7261
Extra Tree	0.6999	0.6575	0.6564
Neural Net	0.7615	0.7671	0.7792
Random Forest	0.7440	0.6821	0.6996

Table 5.15: The F_1-score of each method on the multilabel dataset.

	Multilabel		
	desc	BoW	both
Decision Tree	**0.8446**	0.7711	**0.8639**
Ensemble Extra Tree	0.7917	0.7720	0.7923
Extra Tree	0.7066	0.6651	0.6642
Neural Net	0.7938	**0.7991**	0.8174
Random Forest	0.7733	0.7533	0.7720

Table 5.16: The Hamming loss of each method on the multilabel dataset.

	Multilabel		
	desc	BoW	both
Decision Tree	**0.0188**	0.0286	**0.0169**
Ensemble Extra Tree	0.0195	0.0226	0.0203
Extra Tree	0.0350	0.0412	0.0412
Neural Net	0.0209	**0.0210**	0.0183
Random Forest	0.0208	0.0239	0.0219

Table 5.17: The Jaccard Similarity Score of each method on the multilabel dataset.

	Multilabel		
	desc	BoW	both
Decision Tree	**0.8424**	0.7591	**0.8672**
Ensemble Extra Tree	0.7877	0.7376	0.7652
Extra Tree	0.6911	0.6431	0.6452
Neural Net	0.7673	**0.7596**	0.7888
Random Forest	0.7700	0.7160	0.7394

assigned by Decision Tree on the test dataset. It also indicates the number of cases for each label count. It is striking to observe how the distribution of cases depending on the labels is close to the real distribution shown in Figure 5.3. Therefore, we can reasonably assume that the model correctly identifies the articles of a given case. The subset accuracy for cases with a single label is consistent with the score on the multiclass dataset.

The subset accuracy decreases linearly with the number of labels, which is not surprising, since the metric becomes stricter with the number of labels. However, the recall and precision remain stable, above 80% on average, indicating that not only the algorithm carefully identifies the labels (recall) but also identify a large portion of labels (precision). Thus, from these observations, we can clearly discard the possibility that the algorithm mostly focuses on cases with a single label.

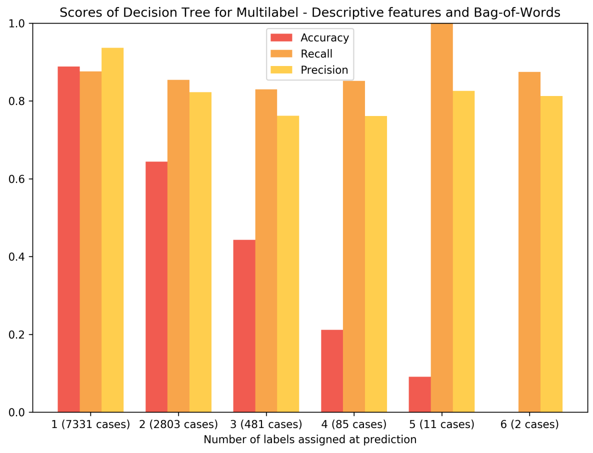


Figure 5.7: Multilabel scores depending on the number of labels assigned.

5.6.3 Discussion

The multilabel experiment is, as far as we know, the first experiment in the legal domain to predict a more structured outcome than a binary outcome. On top of that, it showed that, contrarily to all past studies, textual features are not necessarily holding the most adequate information for prediction. In particular, the multilabel experiment showed that *descriptive features only* are enough to obtain results that are quantitatively close to the combination of both sources of information. We are confident that this conclusion is valid despite being in opposition with all previous conclusions. Indeed, not only the corpus of documents we used is larger but also our models provide better results.

This opens the road to practical applications, while previous studies used only data known a posteriori. For instance, knowing a basic description of a case, a citizen might quickly determine the part of the law that applies to her case, and a reasonable estimation of the outcome. This might help her to find an advisor or representative specialized in this area of the law.

5.7 Conclusion

In this chapter, we presented an open repository, called *ECHR-DB*, of legal cases and judicial decision justifications. The main purposes of constructing the repository are as follows. First, to provide cleaned and transformed content from the repository of the European Court of Human Rights, that is ready to be used by researchers and practitioners. Second, to augment original legal documents with metadata, which will ease the process of analyzing these documents. Third, to provide a benchmark with baseline results for classification models in the legal domain, for other researchers. The repository will be iteratively corrected and updated along with the European Court of Human Rights new judgments.

Currently, *ECHR-DB* is the largest and most exhaustive repository of legal documents from the European Court of Human Rights. It includes several types of data that can be easily used to reproduce various experiments, which have been done so far by other researchers. We argue that providing the final data is not enough to ensure quality and trust. In addition, there are always some alternative choices in the representation, such as the number of tokens, the value of n for the n-grams calculation, or the weighting schema in the TF-IDF transformation. As a remedy, we provide the whole pipeline of dataset construction from scratch. The pipeline was implemented by means of Python scripts and available on GitHub [7].

The experiments on *ECHR-DB* provide a 15pp improvement compared to the previous studies on binary classification and similar results than the state of the art Deep Learning. They also allow us to draw the following conclusions.

1. The models for binary classification clearly underfit while the data is already exhaustive. Therefore, to provide a larger training set, we need to consider the more complex multiclass or multilabel problem. Despite this additional complexity, the multiclass and multilabel models provide as good results as the binary counterparts thanks to this larger training set.
2. Textual features are good at finding if there is a violation or not for a given article, while the descriptive features alone are good at identifying the article. Descriptive features surprisingly hold reasonable predictive power.
3. For the most complex problem that is the multilabel setting, using *descriptive features only* provides equivalent results as textual features. This is particularily important, since descriptive features are available mostly before a verdict contrarily to the judgment document, by definition available after. This opens the road to more practical applications, especially if the results are reproducible with any type of judgments, beyond the European Court of Human Rights.

The experiments indicated several axes of improvements, e.g., better embedding with state of the art encoders, hyperparameter tuning, multi-stage classifiers, and transfer learning. From the obtained results, it seems clear that predicting if an article has been violated or not can be handled with the current state of the art in artificial intelligence. However, other interesting research questions and problems arise from the proposed repository, e.g. *can we*

[7] `https://github.com/aquemy/ECHR-OD_predictions`

provide legal justification in natural language to a prediction?, which will be addressed in the future work.

Supplementary Material

https://echr-od.github.io/ECHR-OD_project_supplementary_material

CHAPTER 6

Hypergraph Case Based Reasoning

Freedom is not the possibility of realizing all its whims; It is the opportunity to participate in defining the constraints that will be imposed on all.

PETITE PHILOSOPHIE À L'USAGE DES NON-PHILOSOPHES, ALBERT JACQUARD

Contents

In many real-life situations, one tries to take a decision based on previous *similar* situations. Each situation is described by a certain amount of relevant information, either collected by an expert, or automatically, e.g. by some sensors. Those situations share similarities on which to make analogies or counter-examples in order to take new decisions. Conversely, in general, if two situations do not share any common characteristic, then they are totally independent, i.e. it is impossible to infer one's outcome from the other one. The purpose of supervised machine learning algorithms is to exploit the available information and interactions between past cases or examples in order to build a model or infer the key rules to take correct decisions.

In this chapter, we investigate the problem of binary prediction under a supervised setting. In particular, this chapter **contributes** to binary classification with a new algorithm called Hypergraph Case-Based Reasoning (HCBR). The method is agnostic to data representation, can work with multiple data sources or in non-metric spaces, and accommodates with missing values. As a result, it drastically reduces the need for data preprocessing or feature engineering. Each element to be classified is partitioned according to its interactions with the training set. For each class, a seminorm over the training set partition is learnt to represent the distribution of evidence supporting this class.

Empirical validation demonstrates its high potential on a wide range of well-known datasets and the results are compared to the state-of-the-art. The time complexity is given and empirically validated. Its robustness with regard to hyperparameter sensitivity is studied and compared to standard classification methods. Finally, the limitation of the model space is discussed, and some potential solutions proposed.

The idea behind HCBR is to create a hypergraph where each element of the training set is a hyperedge and vertices are represented by the features describing the elements. The intersections between edges create a partition, unique to a hypergraph. For each case, we model the support as a convex combination of the elements of this partition. Each of those elements is valued according to its importance w.r.t. the set of all the hyperedges it belongs to and their respective labels.

The plan of the chapter is as follows. First, we insist on the necessity to work in unstructured spaces in Section 6.1.1. A formulation of binary classification in an abstract space of information is proposed in Section 6.1. The main contribution is Section 6.2 which defines HCBR framework. The rest of the chapter focuses on empirical validation: Section 6.3 presents empirical results on seven structured datasets from the UC Irvine Machine Learning Repository (UCI)[1] and the LIBSVM[2], while Section 6.4 is dedicated to unstructured datasets for text classification. Section 6.5 studies important properties, namely the computational

[1]http://archive.ics.uci.edu/ml/index.php
[2]https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

time, learning curves and hyperparameters usage. In Section 7.3, we discuss the current limitations and possible model space extensions. This chapter ends in Section 7.5 with a discussion about the results, future work, and possible improvements

This chapter is based on [Quemy 2018b, Quemy 2019a] with some unpublished work (Sections 6.1 and 6.2.6).

6.1 Binary classification in unstructured spaces

With all the considerations of Section 6.1.1 in mind, we reformulate the problem of binary classification for unstructured space. The only *ad-hoc* operation we allow is checking if a particular feature belongs to the element to classify, and by extension we allow elementary set operations.

6.1.1 On the necessity of classification in unstructured spaces

In machine learning, and broadly speaking in mathematics, the term structured refers to objects that are not living in $\mathbb{R}^n$. In particular, there is no obvious metric between objects. Graphs are an example of structured object and the right notion of distance between two graphs depends on the considered problem.

In the database and data science community, the usage is different. A dataset is structured if it can be easily mapped into pre-defined fields or represented by a relational database or tabular-like schema. By opposition, any other dataset is unstructured or semi-structured (raw text, BoW, JSON, XML, graph, etc.). A structured dataset does not guarantee a norm on the rows because some columns might not be expressed in a space with an obvious metric (e.g. categorical variables).

A key element to apply classification methods to datasets is a (meaningful) metric, independently of whether or not the data are stored in a structured format. **Thus, we define as unstructured, a space without obvious or informative metric[3].**

A tremendous amount of data is collected but never used. On top of that, the natural form of data is *messy*: not sanitized, in different formats, without informative metric, etc. Most of the time, solving optimally a concrete problem requires using data from multiple sources, thus multiplying the above-mentioned problems. This highlights the necessity to develop algorithms capable to work in any unstructured space.

In this thesis, we develop a classification method working with any unstructured space, allowing to relax the usual constraints on the input space:

1. **non-metric space:** there is no metric embedded with the data space,
2. **non-homogeneous cardinality:** the number of features per element is not fixed a priori,
3. **representation agnostic:** the concrete *representation* of features does not influence the model.

[3]A discrete set can always be indexed, however, in this case, the usual distance in $\mathbb{N}$ is uninformative, therefore not defining a structure.

To handle the problem of **non-metric spaces**, one can use *metric learning* methods. We present these methods in Section 3.2 and discuss how they do not handle the two other points.

A special yet very common case of **non-homogeneous cardinality** is some missing values. As the data may be missing for different reasons, there exists different ways of handling them that we briefly present in Section 3.3. However, none of them is as straightforward as having a classification method that works for non-homogeneous cardinality.

Regarding **representation agnosticity**, the representation is very often imposed by the classification method itself and when the dataset does not respect those requirements, it has to be transformed. Of course, exploiting the specificities of features often leads to better results but also requires manual expertise that blocks a wider adoption by end-users.

6.1.2 The binary classification problem in Average Hamming Distance

Consider the abstract set of information $\mathbb{F}$ containing all the available information. For instance, in the case of the legal domain, it represents all possible and relevant information to judge the set of all possible cases at anytime. This includes formal legal knowledge as well as non-legal elements such as the ideology of the judges, external elements or evidences. We assume the information set is measurable and associated to the σ-algebra $\sigma(\mathbb{F})$ generated by its elements. Notably, if $\mathbb{F}$ is discrete, the generated σ-algebra is the powerset of $\mathbb{F}$, i.e. $\sigma(\mathbb{F}) = \mathcal{P}(\mathbb{F})$. For clarity, we use $\mathcal{X}$ for $\sigma(\mathbb{F})$. Each possible element to classify is a measurable element of $\mathbb{F}$, i.e. an element of $\mathcal{X}$.

By abuse of notation, we call $\mathbf{x} \in \mathcal{X}$ an "vector" despite $\mathcal{X}$ is not a vector space. We also refer to $\mathbf{x}$ as a *case* as a reference to the legal domain.

In practice, it is very likely that only a subset of $\mathcal{X}$ may appear (for instance if two features encode two contradictory propositions or if every case has the same number of features). The real class for any input vector $\mathbf{x}$ of $\mathcal{X}$ is given by the decision mapping:

$$\begin{aligned} J\colon \mathcal{X} &\to \{-1,1\} \\ \mathbf{x} &\mapsto J(\mathbf{x}) \end{aligned} \tag{6.1}$$

As the cases to be judged arrives randomly in time, the sequence $\{\mathbf{x}_i\}_i$ is generated by an underlying unknown probabilistic measure μ defined on $\Omega = (\mathcal{X}^\infty, \mathcal{A})$ with $\mathcal{X}^\infty$ the set of one-way infinite sequence in $\mathcal{X}$ and $\mathcal{A}$ its generated σ-algebra. In other words, $\{\mathbf{x}_i\}_i$ is a sequence of elements of the σ-algebra $\mathcal{X}$ randomely generated by μ. Estimating μ is equivalent to understand what type of cases are more likely to be generated and as it includes all the information about a case, it would provides interesting insights about the legal domain and the society in general. In this doctoral thesis, we make the assumption that μ is stationnary and it is **not** our main objective to estimate it.

We denote by $\mathcal{J}(\mathbb{F})$ the class of decision mappings from $\sigma(\mathbb{F})$ to $\{-1,1\}$, or simply $\mathcal{J}$ if there is no ambiguity.

Definition 6.1.1 (Average Hamming Distance)**.** *Given two sequences of* $\mathbf{y} = \{\mathbf{y}_i\}_{1\leq i\leq n}$ *and* $\bar{\mathbf{y}} = \{\bar{\mathbf{y}}_i\}_{1\leq i\leq n}$, *their average Hamming distance is defined by*

$$d_H(\mathbf{y}, \bar{\mathbf{y}}) = \frac{1}{n}\sum_{i=1}^{n} |\mathbf{y}_i - \bar{\mathbf{y}}_i|$$

.

Definition 6.1.2 (Convergence in Average Hamming Distance). *Given two binary sequences* $\mathbf{x}$ *and* $\bar{\mathbf{x}}$, $\bar{\mathbf{x}}$ *converges to* $\mathbf{x}$ *in average Hamming distance, noted* $\bar{\mathbf{x}} \overset{d_H}{\to} \mathbf{x}$, *iif* $\lim\limits_{n\to+\infty} d_H(\bar{\mathbf{x}}, \mathbf{x}`) = 0$

Definition 6.1.3 (Classification problem (in average Hamming distance)). *The classification problem in unstructured space consists in approximating J as solution to the following program:*

$$J^* = \underset{\bar{J}\in\mathcal{J}}{\arg\inf}\{d_H(\bar{J}(\mathbf{x}), J(\mathbf{x}))\},\ \forall \mathbf{x} \in \mathcal{X}^\infty$$

This problem is a functional problem. In particular, if $\exists\{J_k\}_k \in \mathcal{J}^\infty$, $\exists K > 0$, $\forall k \geq K$, $d_H(J_k(F), J(F)) = 0$, $\forall F \in \mathcal{F}^\infty$ then $J_k \to J$ μ-almost surely.

The whole purpose of this chaper is to develop an iterative algorithm $J_k \in \mathcal{J}^\infty$ that converges μ-almost surely to the real decision mapping J for any decision mapping J.

6.1.3 Refining the problem with assumptions

As mentioned above, the binary classification problem in unstructured space is a functional problem. In this doctoral thesis, we make additional assumptions to restrict the search space and make it easier to develop a solution.

Assumption 1 (Outcomes coherence). *The outcomes are coherent iff*

$$\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X} \times \mathcal{X},\ \mathbf{x}_1 = \mathbf{x}_2 \implies J(\mathbf{x}_1) = J(\mathbf{x}_2)$$

This assumption consists in ignoring *uncertainty*: if two situations are described the same in $\mathbb{F}$, then they have the same label.

Assumption 2 (Omniscience). *An information space is omniscient w.r.t. a Decision Mapping J iff the cases $\mathbf{x}_i$ contains only the necessary information to the verdict and all the information is used in a way to reach the decision.*

This assumptions consists in stating that there is no non-necessary information in $\mathbb{F}$ nor that any information is missing. Containing only the nece ssary information can be translated by $\forall \mathbf{a} \not\in \mathcal{X}, \forall \mathbf{x} \in \mathcal{X}, J(\mathbf{a} \cup \mathbf{x}) = J(\mathbf{x})$. All the information being used, means that the mapping J is not monotonic, i.e. $\forall \mathbf{x}_1 \in \mathcal{X}, \forall \mathbf{x}_2 \in \mathcal{X}, \mathbf{x}_1 \subset \mathbf{x}_2, J(\mathbf{x}_1) = y_1 \not\Longrightarrow J(\mathbf{x}_2) = y_1$ and $J(\mathbf{x}_2) = y_2 \not\Longrightarrow J(\mathbf{x}_1) = y_2$.

Definition 6.1.4 (Well-defined Classification Problem). *A well-defined Classification Problem is a pair $(\{\mathbf{x}_i\}_i, J)$, with $\{\mathbf{x}_i\}_i \in \mathcal{X}^\infty$ and $J \in \mathcal{J}$, for which the outcomes coherence and omniscience assumption hold.*

From now, we consider a well-defined classification problem.

Finally, in this doctoral thesis, we consider only a countable space $\mathbb{F}$ and define $\mathcal{X} = 2^{\mathbb{F}}$. Concretely, the elements of $\mathbb{F}$ can take different forms depending on the data: a pair "variable=value", a word, a proposition that is true or false, etc. The only minimal requirement is that there exists a way to index the data elements. In the experimental validation performed in Section 6.3, the datasets are structured (with some missing values) and thus, the features are all represented by a pair “variable=value". In Section 6.4, the features are represented by “word=occurrences" if a word appears at least once. However, as $\mathbb{F}$ can be seen as a subset of $\mathbb{N}$, the method HCBR presented below would work on datasets mixing representations (descriptive features, Bag-of-Words, etc.).

6.1.4 Some results when $\mathbb{F}$ is finite

Let M denotes the cardinality of $\mathbb{F}$. The σ-algebra $\mathcal{X}$ is thus the powerset of $\mathbb{F}$ and contains a maximum number of possible elements equals to $N = 2^M$. We assume that $\forall \mathbf{x} \in \mathcal{X},\ \mu(\mathbf{x}) > 0$, that is to say, each possible case a non-null probability to appear. In such configuration, the existence of a convergence sequence is trivial and an obvious method to build an estimator $\bar{J}_k$ of J that converges in average Hamming distance is to wait long enough. At each timestep k, if the new case $\mathbf{x}_k$ already has been judged once, by the Coherence assumption, we can reuse the past outcome. In other words, we build $\bar{J}_k$ such that

$$\forall \mathbf{x}_k \in \mathcal{F},\ \bar{J}_k(\mathbf{x}_k) = \begin{cases} y_j = J(\mathbf{x}_j) & \text{if } \exists j < k,\ \mathbf{x}_k = \mathbf{x}_j \\ b_k & \text{otherwise} \end{cases}$$

where b_k is drawn according to a Bernoulli law with parameter $\frac{1}{2}$.

In this case, the minimum of distance is reached and is 0, i.e. $\bar{J}_k(\mathbf{x}_k) \overset{d_H}{\to} J(\mathbf{x}_k),\ \forall \{\mathbf{x}_k\}_k \in \mathcal{F}^\infty$. This minimum is reached for any value b_k because for any $\bar{J}$ constructed as mentioned above, there exists a K above which, we already visited every possible case. This implies that the average Hamming distance of the sequence of outcomes will necessarely start to decrease with $k > K$ since $\forall k > K,\ \bar{J}_k = J$.

We can calculate the probability to be able to reconstruct perfectly J after k timesteps depending of M and μ. Indeed, at each timestep the new case $\mathbf{x}_i$ is drawn from a categorical distribution with parameters (N, μ). To avoid confusion, we use the superscript to index the cases in $\mathcal{F}$. Let denote by $\#_k\mathbf{x}_i$ the number of times $\mathbf{x}_i$ appeared after k cases.

Lemma 3. $\forall k \in \mathbb{N}, \mathbb{P}(\#_k\mathbf{x}_i \geq 1,\ \forall i \leq k) = \begin{cases} k \sum\limits_{a \in K_m^N} \frac{1}{\beta(a+1)} \prod\limits_{i=1}^{N} \frac{\mu(\mathbf{x}_i)^{a_i+1}}{(a_i+1)} & \text{if } k \geq N \\ 0 & \text{otherwise} \end{cases}$ *with* $m =$ $k - N$, K_m^N *is the set of* m*-multicombinations (or multisets of size* m*) defined by* $K_m^N = \{a \in \mathbb{N}^N \mid ||a||_1 = m\}$ *and with* β *the multivariate beta function defined by* $\beta(a) = \frac{\prod_i^N \Gamma(a_i)}{\Gamma(\sum_i^N a_i)}$.

Proof: We reconstruct perfectly J if each case has been visited at least once.

$$\mathbb{P}(\{\#_k\mathbf{x}_i\}_i \geq 1,\ \forall i \leq k) = \sum_{a=(a_1,\dots,a_N) \in K_m^N} \mathbb{P}(\{\#_k\mathbf{x}_i\}_i = 1 + a_i,\ \forall i \leq k)$$

with $\forall a,\ \sum\limits_i a_i = m$ where $\forall i,\ a_i \in \{0, \dots, m\}$.

For a given $a \in K_m^N$, $\{\#_k\mathbf{x}_i\}_i$ can be seen as a random variable following a multinomial distribution with parameters k and μ, i.e. $\mathbb{P}(\{\#_N\mathbf{x}_i\}_i = 1 + a_i) = \frac{k!}{\prod\limits_{i=1}^{N}(1+a_i)!} \prod\limits_{i=1}^{N} \mu(\mathbf{x}_i)^{1+a_i}$

$$
\begin{aligned}
\mathbb{P}(\{\#_k \mathbf{x}_i\}_i \geq 1) &= \sum_{a \in K_m^N} \frac{k!}{\prod\limits_{i=1}^{N} (1+a_i)!} \prod_i \mu(\mathbf{x}_i)^{1+a_i} \\
&= \sum_{a \in K_m^N} \frac{\Gamma(\sum_i (a_i+1)+1)}{\prod_i \Gamma(a_i+2)} \prod_{i=1}^{N} \mu(\mathbf{x}_i)^{1+a_i} \\
&= \sum_{a \in K_m^N} k \frac{\Gamma(\sum_i a_i + 1)}{\prod_i (a_i+1)\Gamma(a_i+1)} \prod_{i=1}^{N} \mu(\mathbf{x}_i)^{1+a_i} \\
&= k \sum_{a \in K_m^N} \frac{1}{\beta(a+1)} \prod_{i=1}^{N} \frac{\mu(\mathbf{x}_i)^{1+a_i}}{(a_i+1)}
\end{aligned}
$$

□

If the convergence is certain, the question we want to answer is: can we create an estimator with a faster convergence. As we do not have any control on μ and thus the sequence $\mathbf{x} \in \mathcal{X}^\infty$, the only margin of maneuvre is to improve the "prediction" b_k we make at time k. To do so, we will use the information contained in the structure generated by the $\{\mathbf{x}_i\}_{1 \leq i \leq k}$ that we will study in Section 6.2.

We can rewrite this problem as finding $\bar{J}_k$ such that $\forall k \in \mathbb{N}, \mathbf{x}_k \not\in \{\mathbf{x}_i\}_{1\leq i \leq k-1}$, $\mathbb{E}_\mu(\tilde{J}_k(\mathbf{x}_k) = J(\mathbf{x}_k)) < \frac{1}{2}$, that is to say, doing better than randomly picking an outcome with a uniform probability.

Before presenting HCBR, we are interested in the following quantity. Given a set X, how many sequences of k steps $\{\mathbf{x}_i\} \in \mathcal{X}^k$ that form a cover of X exist?

Lemma 4.

$$
|\{\{\mathbf{x}_i\} \in \mathcal{X}^k \mid \bigcup_i \mathbf{x}_i = X\}| = \sum_{p \in P(k,m)} \frac{m!}{[\prod\limits_{i=0}^{k-1} (p_{i+1}!)](p_k)!} 2^{\sum\limits_{i=0}^{k-1} (k-1)p_i}
$$

Proof: A valid sequence $\{\mathbf{x}_i\} \in \mathcal{X}^k$ is described by the fact there is at least one element in each $\mathbf{x}_i$ and the constraint on the cover of $\mathbf{x}_k$. Let denote by $P(k,m)$ the set of partitions of m into at most k subsets.

For each $p \in P(k,m)$, the number of possible sequence is $A(p) = \prod\limits_{i=1}^{k-1} \binom{m-\alpha_i}{p_{i+1}} \sum\limits_{j=0}^{\alpha_i} \binom{\alpha_i}{p_{i+1}} = \prod\limits_{i=1}^{k-1} \binom{m-\alpha_i}{p_{i+1}} 2^{\alpha_i}$ with $\alpha_i = \sum\limits_{j=0}^{i} p_j$ and $p_0 = 0$ by convention.

Let illustrate the formula for $m = 5$, $k = 3$ and $p = (2,2,1)$. In this case, we firstly select 2 elements in the 5 to be covered. Then, the second element should have 2 new elements, so among the 3 remaining elements. However, as the elements of the sequence are not disjoint, we can chose any number of features already covered, that is to say the sum from 0 to 2 elements in 2.

$$\prod_{i=1}^{k-1}\binom{m-\alpha_i}{p_{i+1}} = \frac{(m-\alpha_0)!}{[\prod_{i=0}^{k-1}(p_{i+1}!)](m-\alpha_{k-1})!} = \frac{m!}{[\prod_{i=0}^{k-1}(p_{i+1}!)](p_k)!}$$

$$\prod_{i=1}^{k-1} 2^{\alpha_i} = 2^{\sum_{i=0}^{k-1}(k-1)p_i}$$

which lead to

$$A(p) = \frac{m!}{[\prod_{i=0}^{k-1}(p_{i+1}!)](p_k)!} 2^{\sum_{i=0}^{k-1}(k-1)p_i}$$

□

Corollary 5.

$$\forall p \in P(k,m),\ A(p) \leq \frac{m!}{kp_k!(\min_i p_i)!} 2^{k(m-p_k)}$$

6.2 Hypergraph Case-Based Reasoning

In this section, we introduce our main contribution with **a new framework for binary classification in unstructured spaces** called Hypergraph Case-Based Reasoning (HCBR). The presentation is broken down into five steps. First, we present in Section 6.2.1 how to represent the training set as a hypergraph and how to project any element of $2^{\mathbb{F}}$ onto it. These elements allow us to formally define the model space in Section 6.2.2. Section 6.2.3 is dedicated to parameter estimations while Section 6.2.4 focuses on the training. Finally, Section 6.2.5 is dedicated to the decision rule refinement and hyperparameters. Section 6.2.7 provides the time complexity of the main phases of the algorithm.

6.2.1 Representation and projection

Before defining the projection operator used by HCBR to make predictions, we recall the definition of a hypergraph.

Definition 6.2.1 (Hypergraph)**.** *A hypergraph is defined by $H = (V, \mathbf{X})$ with V a set of vertices, $\mathbf{X}$ the hyperedges s.t. $\forall \mathbf{x} \in \mathbf{X}$, $\mathbf{x} \subseteq V$.*

A training set $\mathbf{X}$ can be seen as a hypergraph $H = (\mathbb{F}, \mathbf{X})$, i.e. such that each example is a hyperedge (Figure 6.1). In practice, we do not need to know $\mathbb{F}$ as we can always use the implicit

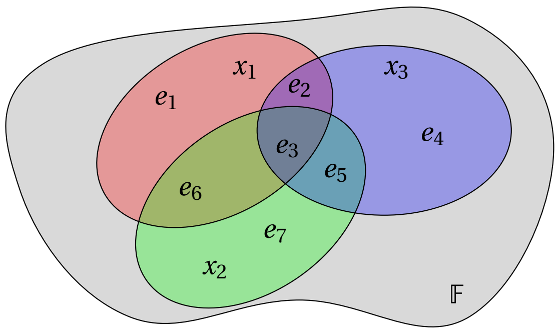


Figure 6.1: The family $\mathcal{E} = \{\mathbf{e}_i\}_i$ forms the partition obtained by the union of the projection of cases and represents how the three cases share information.

hypergraph $H = (\mathbb{F}_\mathbf{X}, \mathbf{X})$ where $\mathbb{F}_\mathbf{X}$ is the restriction of $\mathbb{F}$ to the features present in the sample $\mathbf{X}$. For structured datasets, the elements of $\mathbb{F}_\mathbf{X}$ are all the distinct pairs "variable=value" in $\mathbf{X}$. For any hypergraph $H = (\mathbb{F}_\mathbf{X}, \mathbf{X})$, there exists a unique partition $\mathcal{E}_H = \{\mathbf{e_i}\}_{i=1}^M$, $\forall 1 \leq i \leq M$, $\mathbf{e_i} \subseteq \mathbb{F}_\mathbf{X}$ defined by the intersections of its edges as illustrated by Figure 6.1.

The projection of a case over a hypergraph returns the elements of $\mathcal{E}_H$ it intersects with.

Definition 6.2.2 (Projection over a hypergraph)**.** *The projection operator π_H over a hypergraph $H = (V, \mathbf{X})$ for any $A \subseteq V$ is defined by $\pi_H(A) = \{\mathbf{e} \in \mathcal{E}_H \mid \mathbf{e} \cap \mathbf{x} \neq \emptyset\}$.*

Example 1: Each element of $\pi_H(\mathbf{x})$ is a (sub)set of features. For instance, in Figure 6.1, $\pi_H(\mathbf{x_1}) = \{\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}, \mathbf{e_6}\}$ and in Figure 6.2, the projection of $\mathbf{x}$ (in yellow) on H is the whole partition $\mathcal{E}_H$ as $\mathbf{x}$ intersects with every $\mathbf{e} \in \mathcal{E}_H$.

We call *discretionary features* of $\mathbf{x}$ (w.r.t. H) the (possibly empty) set of features that are not in $V_\mathbf{X}$, denoted by $D_\mathbf{x}$. It can be interpreted as the features of $\mathbf{x}$ that do not belong to any hyperedge. When adding hyperedges, the discretionary features of $\mathbf{x}$ are new pieces of information that were never encountered before.

Example: Considering the hypergraph composed of $\mathbf{x_1}$ and $\mathbf{x_2}$ as illustrated by Figure 6.1, the set of discretionary features of $\mathbf{x_3}$ is $\mathbf{e_4}$. In Figure 6.2, the yellow case $\mathbf{x}$ has no discretionary feature: all its features are present at least in one example.

For any set of features $\mathbf{x} \subseteq \mathbb{F}$, we define $d_H(\mathbf{x}) = \{\mathbf{x}' \in \mathbf{X} \mid \mathbf{x} \cap \mathbf{x}' \neq \emptyset\}$ the set of edges sharing some features with $\mathbf{x}$. In particular, the set d_H can be split into $d_H^{(1)}$ and $d_H^{(-1)}$ depending on the label of the case, and defined by $d_H^{(l)}(\mathbf{x}) = \{\mathbf{x}' \in d_H(\mathbf{x}) \mid J(\mathbf{x}') = l\}$. Note that if $\mathbf{x} \not\in \mathbf{X}$ and $|d_H(\mathbf{x})| = 0$, then $\mathbf{x} = D_\mathbf{x}$, i.e. the case $\mathbf{x}$ is in relation with no case in $\mathbf{X}$. In the hypergraph literature, $|d(\mathbf{x})|$ is called the *degree* of a hyperedge and its domain of definition is restricted to $\mathbf{X}$ while here, it is extended to $2^\mathbb{F}$. Finally, to use matrix notations, we define the vectors $\mathbf{d}^l = (|d^{(l)}(\mathbf{x}_i)|)_{i=1}^N$.

From now, we consider only the specific hypergraph generated by the set of examples $\mathbf{X}$. For the sake of readability, we remove the subscript H.

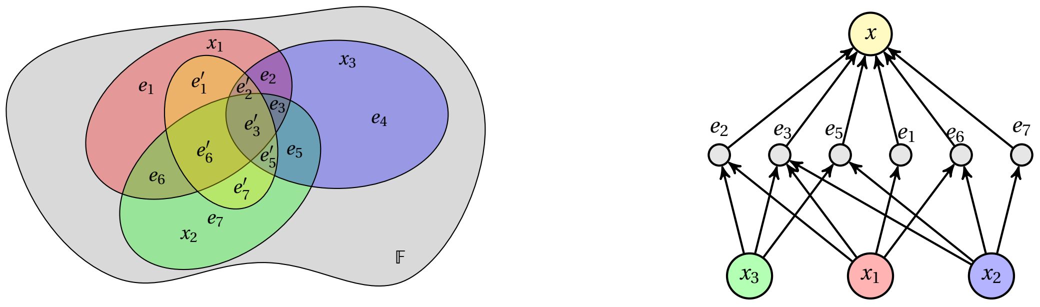


Figure 6.2: The projection of $\mathbf{x}$ (in yellow) on H is the set of $\mathbf{e}_i$ intersecting with it. On the right, the graph represents the projection elements $\{\mathbf{e}_i\}_i$ and their respective connections to the cases $\{\mathbf{x}_i\}_i$, in particular, $D_{\mathbf{x}} = \emptyset$.

6.2.2 Model space

As discussed in Section 6.1.1, we relaxed some implicit constraints on the input vector space. As a counterpart, HCBR relies on the assumption that if two input vectors $\mathbf{x}$ and $\mathbf{x}'$ do not share any feature, they are *independent* i.e. $\mathbf{x}$ cannot help to understand the correct class of $\mathbf{x}'$ and vice versa. This limitation comes from the fact there is no metric on $\mathbb{F}$ to determine a distance between two elements s.t. we can rely only on intersections. As a result, HCBR produces only *local* models because if a new input vector is independent of all examples, it is impossible to generate a prediction. On the contrary, a hyperplane model is *global* in a sense that it can produce a prediction for any element of $\mathbb{R}^M$. We discuss and propose a solution to this issue in Section 6.6.3.

An example of concrete situation for which such assumption is natural is a justice trial. Cases are composed of some elements, and the correct label is the result of a reasoning that can possibly use analogies or counter-examples with a set of past cases on top of the legal texts. However, if a judge or lawyer wants to use $\mathbf{x}$ to justify the outcome of $\mathbf{x}'$, $\mathbf{x}'$ must have similarities with $\mathbf{x}$.

First, we give the **intuition** behind our model space. For each element $\mathbf{e}_i$ in the projection, we value 1) its importance in the case $\mathbf{x}$ and 2) its support toward a specific class given the whole hypergraph. Intuitively, 1) consists in answering the question: *how important is* $\mathbf{e}_i$ *w.r.t.* $\mathbf{x}$*?* or *what is the potential for analogies or counter-examples between* $\mathbf{x}$ *and the other examples also containing* $\mathbf{e}_i$*?* The purpose of 2) is to value how important is $\mathbf{e}_i$ with regards to the other elements of $\mathcal{E}$. For instance, to explain the case $\mathbf{x}_1$ in Figure 6.1, we will use only the elements of the projection $\pi_H(\mathbf{x_1}) = \{\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}, \mathbf{e_6}\}$. Without prior information, we will use the size of $\mathbf{e}_i$ in $\mathbf{x}$ to measure their importance $\mathbf{x}$. The way of measuring the support with regard to the whole hypergraph as well as an intuitive interpretation will be given in Section 6.2.3.

Let us now **formally** define the model space. Given the hypergraph $H = (\mathbb{F}, \mathbf{X})$ defined by a training set $\mathbf{X}$ and its associated partition $\mathcal{E} = \{\mathbf{e_i}\}_{i=1}^{m}$, the relation between an example $\mathbf{x}$ and its class is modeled by

$$\begin{cases} s_{w,\mu}(\mathbf{x}_j) & = & \sum\limits_{i=1}^{M} w(\mathbf{e}_i,\mathbf{x}_j)\mu(\mathbf{e}_i,\mathbf{X},\mathbf{y}) \\ \sum\limits_{i=1}^{M} w(\mathbf{e}_i,\mathbf{x}_j) & = & 1 \qquad \forall 1 \le j \le N \\ \sum\limits_{i=1}^{M} \mu(\mathbf{e}_i,\mathbf{X},\mathbf{y}) & = & 1 \end{cases} \tag{6.2}$$

where $w(\mathbf{e}_i,\mathbf{x}_j) \geq 0$ models the importance of $\mathbf{e}_i$ in $\mathbf{x}_j$ and $\mu(\mathbf{e}_i,\mathbf{X},\mathbf{y})$ the support of $\mathbf{e}_i$ for class 1 w.r.t. whole training set. For this reason, we call μ the *intrinsic strength* of $\mathbf{e}_i$. The above-mentioned assumption implies that if $\mathbf{e}_i \cap \mathbf{x} = \emptyset$ then $w(\mathbf{e}_i,\mathbf{x}) = 0$. For readability, we write s in place of $s_{w,\mu}$.

The classification rule consists in selecting the class with the total highest support:

$$\bar{J}_{w,\mu}(x) = \begin{cases} 1 & s_{w,\mu}(\mathbf{x}) > 0 \\ -1 & s_{w,\mu}(\mathbf{x}) \leq 0 \end{cases} \tag{R1}$$

The classification problem consists then in finding the couple (w,μ) that minimizes the classification error:

$$(w^*,\mu^*) = \underset{w,\mu}{\arg\min} \sum_{(\mathbf{x},y)\in(\mathbf{X},\mathbf{y})} \mathbb{1}_{\{y\neq \bar{J}(\mathbf{x})\}} \tag{6.3}$$

The problem described by (6.3) is a functional problem, in general much harder than parametric ones such as (3.5), and without restrictions on the search space it may not be reasonable to look for a solution. For this reason, and for the rest of this chapter, we will fix w a priori. As we do not formulate any particular assumption on the feature space and allow only basic set operations, a natural yet trivial way to model the importance of $\mathbf{e}_i$ in $\mathbf{x}_j$ is to set $w(\mathbf{e}_i,\mathbf{x}_j) = \frac{|\mathbf{x}_j \cap \mathbf{e}_i|}{|\mathbf{x}_j|}$. However, if one has some information about the impact of some features in some particular cases, w might be redefined to integrate this prior knowledge.

Once the method to calculate w and μ is specified, the model can be expressed matricially by

$$\begin{cases} \mathbf{s} & = & W\mu \quad W \in \mathcal{M}(\mathbb{R})_{N\times M},\ \mu \in \mathbb{R}^M \\ ||\mathbf{w}_{j:}||_1 & = & 1 \qquad \forall 1 \leq j \leq N \\ ||\mu||_1 & = & 1 \end{cases} \tag{6.4}$$

By setting $d_\mu(\mathbf{x},\mathbf{x}') = |s(x) - s(x')|$, we can see the problem as learning a metric parametrized by μ s.t. the training set elements are properly classified. Rather than learning directly μ, we split it into $\mu^{(1)}$ and $\mu^{(-1)}$, representing the distribution of the support toward class -1 and 1 over $\mathcal{E}$.

To fit a model capable of generalization, it is not enough to simply select μ s.t. it minimizes the classification error. Assuming W is fixed and $\mathbf{s}$ is set to some arbitrary values s.t. each example is correctly classified, μ can be obtained by using the Moore-Penrose pseudo-inverse of W. The Moore-Penrose pseudo-inverse generalizes the inverse for non-square matrix. It always exists and is unique. Depending on $\mathbf{s}$ and W, it might be impossible to classify correctly all the elements using $\mu = W^+\mathbf{s}$. However, by construction of the Moore-Penrose pseudo-inverse, $W^+\mathbf{s}$ is a least-square minimizer of $\mathbf{s} = W\mu$. If this system has many solutions, $W^+\mathbf{s}$ is the solution with the lowest norm $||.||_2$.

By artificially setting **s** to obtain μ with the Moore-Penrose inverse, the constructed model holds no discriminative information and there is no chance it can provide good results on new cases. For this reason, μ has to be designed such that **s** is *calibrated*, i.e. the higher **s** is, the more confident the model is about the prediction. Ideally, among all the elements with, e.g. a (normalized) strength close to 0.8, 80% of them should be correctly classified.

To summarize, after defining a general model space to solve the binary classification problem in unstructured space, we made additional assumptions to reduce the problem of model selection to find a vector μ s.t. the classification rule (R1) provides a good solution to the original problem (3.1). For this purpose, we will proceed in three steps:

- **Step 1:** Define μ to capture as much information as possible from $\mathcal{E}$ for any $\mathbf{X}$ (Section 6.2.3).
- **Step 2:** Train the model to adjust μ on a specific $\mathbf{X}$ using the classification rule (R1) (Section 6.2.4).
- **Step 3:** Refine the classification rule (R1) to take into account the local nature of the model (Section 6.2.5). The refined classification rule (R2) is presented in Section 6.2.5.

A summary and high level view of HCBR is given by Algorithm 1.

Algorithm 1 HCBR (High level view).

```
1: Build $H$ and $\mathcal{E}$ from $\mathbf{X}$.
2: Calculate $w$ and $\mu$ on $\mathcal{E}$.
3: Adjust $\mu$ with training algorithm 2 on $\mathbf{X}$ using rule (R1)
4: for each $\mathbf{x}$ in the test set do
5:     Calculate the projection $\pi(\mathbf{x})$.
6:     Calculate the support $s(\mathbf{x})$ using the projection.
7:     Predict using the refined rule (R2).
8: end for
```

While SVM aims at separating the data using a single hyperplane in the original input vector space, HCBR tries to explain the decision for each case by a convex combination expressed in a lower dimensional space $\mathcal{E}$.

6.2.3 Step 1 - Model selection

In this section, we define how to concretely select μ for a given hypergraph. Ultimately, μ is a measure over $\mathbb{F}_\mathbf{X}$ directly, and cannot be interpreted as a probability: $\mathbb{F}_\mathbf{X} \in 2^{\mathbb{F}_\mathbf{X}}$ but by definition of equation (6.2), $\mu(\mathbb{F}_\mathbf{X}) = 1$. However, there is absolutely no reason to think that $J(\mathbb{F}_\mathbf{X}) = 1$. Our definition of μ tries to answer the following question: *knowing a certain amount of information materialized by the intersection family $\mathcal{E}$, where are located the features holding more discriminative information than the others?* Once again, with further information about the features, a prior might be introduced to select μ.

For $\mathbf{x}$ and $\mathbf{x}'$ in $2^{\mathbb{F}}$, a basic quantitative measure on the importance of $\mathbf{x}'$ w.r.t. $\mathbf{x}$ can be expressed by $\frac{|\mathbf{x}\cap\mathbf{x}'|}{|\mathbf{x}|}$, i.e. how much $\mathbf{x}'$ overlaps with $\mathbf{x}$. This measure is a sort of *potential* for an analogy with $\mathbf{x}$. Potential because it does not account for the individual importance of the features and simply holds the idea that the larger is a subset of features in a case, the higher is the chance it holds important features to decide the outcome.

Let us consider $\mathcal{E}$ and an example $\mathbf{x} \in \mathbf{X}$.

Definition 6.2.3 (Intrinsic strength w.r.t. $\mathbf{x}$). *The intrinsic strength of* $\mathbf{e} \in \mathcal{E}$ *w.r.t.* $\mathbf{x} \in \mathbf{X}$ *is defined by*

$$\forall l \in \{-1,1\},\ \bar{S}^{(l)}(\mathbf{e},\mathbf{x}) = \frac{|d^{(l)}(\mathbf{e})|\frac{|\mathbf{x}\cap\mathbf{e}|}{|\mathbf{x}|}}{\sum\limits_{\mathbf{e}_j\in\mathcal{E}} |d^{(l)}(\mathbf{e}_j)|\frac{|\mathbf{x}\cap\mathbf{e}_j|}{|\mathbf{x}|}} = \frac{|d^{(l)}(\mathbf{e})||\mathbf{x}\cap\mathbf{e}|}{\sum\limits_{\mathbf{e}_j\in\mathcal{E}} |d^{(l)}(\mathbf{e}_j)||\mathbf{x}\cap\mathbf{e}_j|} \tag{6.5}$$

Matricially, for any $\mathbf{e}_i$ *and* $\mathbf{x}_j$,

$$\bar{S}^{(l)}_{i,j} = \frac{d^{(l)}_i w_{ji}}{<\mathbf{d}^{(l)},\mathbf{w}_{:i}>} \tag{6.6}$$

In particular, for a given $\mathbf{x} \in \mathbf{X}$, $\bar{S}^{(l)}(\mathbf{e}_i,\mathbf{x}) = 0$ if $\mathbf{e}_i$ is not part of the projection of $\mathbf{x}$ on H.

The more $\mathbf{e}_i$ is shared by many cases with the same class l, the higher $\bar{S}^{(l)}(\mathbf{e}_i,\mathbf{x})$ is. Conversely, for a fixed number of cases, the more $\mathbf{e}_i$ describes $\mathbf{x}$, the higher $\bar{S}^{(l)}(\mathbf{e}_i,\mathbf{x})$ is. As $\forall \mathbf{e}_i \in \mathcal{E}$, $|d(\mathbf{e}_i)| > 0$, either we have $\bar{S}^{(1)}(\mathbf{e}_i,\mathbf{x}) \neq 0$ or $\bar{S}^{(-1)}(\mathbf{e}_i,\mathbf{x}) \neq 0$. We have $\bar{S}^{(l)}(\mathbf{e}_i,\mathbf{x}) = 0$ only when all the cases in which $\mathbf{e}_i$ results of are labeled with the opposite class $\bar{l}$. For $\bar{S}^{(l)}(\mathbf{e}_i,\mathbf{x}) = 1$, it needs both the unanimity of labels for the cases in which $\mathbf{e}_i$ belongs to and that $\mathbf{e}_i = \mathbf{x}$. The relation $\mathbf{e}_i = \mathbf{x}$ implies that $\mathbf{x}$ does not share any feature with any other example or that $\mathbf{x}$ is included into another example.

Definition 6.2.4 (Intrinsic strength w.r.t. a hypergraph H). *The intrinsic strength of* $\mathbf{e} \in \mathcal{E}$ *w.r.t.* $H = (\mathbb{F}_{\mathbf{X}},\mathbf{X})$ *is defined by*

$$\forall l \in \{-1,1\},\ S^{(l)}(\mathbf{e}) = \frac{|\mathbf{e}|}{\sum\limits_{\mathbf{e}'\in\mathcal{E}}|\mathbf{e}'|} \sum_{\mathbf{x}\in d^{(l)}(\mathbf{e})} \bar{S}^{(l)}(\mathbf{e},\mathbf{x}) = \frac{|\mathbf{e}|}{|\mathbb{F}_{\mathbf{X}}|} \sum_{\mathbf{x}\in d^{(l)}(\mathbf{e})} \bar{S}^{(l)}(\mathbf{e},\mathbf{x}) \tag{6.7}$$

Matricially, for any $\mathbf{e}_i$,

$$S^{(l)}_i = \frac{|\mathbf{e}|}{|\mathbb{F}_{\mathbf{X}}|}||\mathbf{S}^{(l)}_{i:}||_1 \tag{6.8}$$

The more $\mathbf{e}$ belongs to several cases, the more information it represents to support one or another class. As $\mathcal{E}$ represents the sets of features that appear all the time together, we favor the larger $\mathbf{e} \in \mathcal{E}$ as they hold more information to explain a decision. The normalized version is defined by:

$$\forall l \in \{-1,1\},\ \mu^{(l)}(\mathbf{e}) = \frac{S^{(l)}(\mathbf{e})}{\sum\limits_{\mathbf{e}'\in\mathcal{E}} S^{(l)}(\mathbf{e}')} \tag{6.9}$$

Finally, the measure μ is simply defined by the difference between the strength of both classes:

$$\mu(\mathbf{e}) = \mu^{(1)}(\mathbf{e}) - \mu^{(-1)}(\mathbf{e}) \tag{6.10}$$

which can be expressed matricially for a given $\mathbf{e}_i$ by

$$\mu_i = \frac{|\mathbf{e}_i|}{|\mathbb{F}_{\mathbf{X}}|}\left[\frac{S_i^{(1)}}{||S^{(1)}||_1} - \frac{S_i^{(-1)}}{||S^{(-1)}||_1}\right] \tag{6.11}$$

Example (Numerical example): Consider the hypergraph in Figure 6.1 made of $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ arbitrarily labeled with resp. 1, -1 and 1. Arbitrarily, we assume the cardinal of the elements of $\mathcal{E}$ to be $\#\mathbf{e} = (2,1,2,3,1,2,3)$ s.t. the cardinal of cases are $\#\mathbf{x} = (7,8,7)$ and $|\mathbb{F}_{\mathbf{X}}| = 14$. The values of $|d^{(l)}(\mathbf{e})|$ can be summarized by the vectors $\mathbf{d}^{(-1)} = (0,0,1,0,1,1,1)$ and $\mathbf{d}^{(1)} = (1,2,2,1,1,1,0)$. Let us calculate $S^{(-1)}(\mathbf{e}_3)$:

$$\begin{aligned}
\bar{S}^{(1)}(\mathbf{e}_3,\mathbf{x}_1) &= \frac{2\times 2}{2\times 1+1\times 2+2\times 2+1\times 2} = \frac{4}{10}\\
\bar{S}^{(1)}(\mathbf{e}_3,\mathbf{x}_2) &= \frac{2\times 2}{2\times 2+1\times 1+1\times 2+0\times 3} = \frac{4}{7}\\
\bar{S}^{(1)}(\mathbf{e}_3,\mathbf{x}_3) &= \frac{2\times 2}{2\times 1+2\times 2+3\times 1+1\times 1} = \frac{4}{10}
\end{aligned}$$

which we interpret as $\mathbf{e}_3$ being responsible for $\frac{4}{10}$ of the support toward class 1 in $\mathbf{x}_1$ and $\mathbf{x}_3$, while $\frac{4}{7}$ for $\mathbf{x}_2$. This leads to

$$S^{(1)}(\mathbf{e}_3) = \frac{2}{14}\left[\frac{4}{10} + \frac{4}{7} + \frac{4}{10}\right] \simeq 0.1959$$

Similarly, we compute the support for each $\mathbf{e}$ and both labels. We summarize this into the following vectors:

$$\begin{aligned}
\mathbf{S}^{(1)} &\simeq (0.0286, 0.0286, 0.1959, 0.0643, 0.0173, 0.0694, 0.0000)\\
\mathbf{S}^{(-1)} &\simeq (0.0000, 0.0000, 0.2024, 0.0000, 0.0327, 0.1071, 0.0000)
\end{aligned}$$

After normalization, we obtain the intrinsic strength:

$$\mu \simeq (0.0707, 0.0707, 0.0060, 0.1591, -0.0345, -0.0818, -0.1901)^T$$

Let us evaluate the model on the three examples:

$$\begin{aligned}
s(\mathbf{x}_1) &= \frac{2}{7}\mu(\mathbf{e}_1) + \frac{1}{7}\mu(\mathbf{e}_2) + \frac{2}{7}\mu(\mathbf{e}_3) + \frac{2}{7}\mu(\mathbf{e}_6) \simeq 0.0086\\
s(\mathbf{x}_2) &= \frac{2}{8}\mu(\mathbf{e}_3) + \frac{1}{8}\mu(\mathbf{e}_5) + \frac{2}{8}\mu(\mathbf{e}_6) + \frac{3}{8}\mu(\mathbf{e}_7) \simeq -0.0946\\
s(\mathbf{x}_3) &= \frac{1}{7}\mu(\mathbf{e}_2) + \frac{2}{7}\mu(\mathbf{e}_3) + \frac{3}{7}\mu(\mathbf{e}_4) + \frac{1}{7}\mu(\mathbf{e}_5) \simeq 0.0751
\end{aligned}$$

As a result, $\mathbf{x}_1$ and $\mathbf{x}_3$ are labeled 1 and $\mathbf{x}_2$ is labeled -1. All three cases are correctly labeled. The highest support is given for case $\mathbf{x}_2$ and $\mathbf{x}_3$ while the support for $\mathbf{x}_1$ is one order of magnitude lower than for $\mathbf{x}_3$. This is because the discretionary features of $\mathbf{x}_3$ are larger while the intersection with $\mathbf{x}_2$ is lower than for $\mathbf{x}_1$ ($\frac{3}{8}$ of $\mathbf{x}_3$ against $\frac{4}{7}$ of $\mathbf{x}_1$).

Consider a new case $\mathbf{x}$ as described in Figure 6.2. Its support is given by $s(\mathbf{x}) = \sum_{\mathbf{e}\in\pi(\mathbf{x})} w(\mathbf{e},\mathbf{x})\mu(\mathbf{e})$ with $\pi(\mathbf{x}) = \{\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3,\mathbf{e}_5,\mathbf{e}_6,\mathbf{e}_7\}$. It is impossible for $\mathbf{x}$ to be classified as 1 because the highest support would be for a maximal intersection with $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ and minimal for $\mathbf{e}_5$, $\mathbf{e}_6$ and $\mathbf{e}_7$ s.t. $s(\mathbf{x}) = \frac{1}{8}(2\mu(\mathbf{e}_1) + \mu(\mathbf{e}_2) + 2\mu(\mathbf{e}_3) + \mu(\mathbf{e}_5) + \mu(\mathbf{e}_6) + \mu(\mathbf{e}_7)) \simeq -0.0103 < 0$. It can be explained by the fact that the support for 1 is provided by a larger set of features (11 features versus 8). On top of that, the intersections between positive cases ($\mathbf{e}_2$ and $\mathbf{e}_3$) are too small (1 for $\mathbf{e}_2$ compared to, e.g. 3 for $\mathbf{e}_7$) or include also negative cases ($\mathbf{e}_3$).

Example (Car accident): We are interested in knowing if a driver is responsible for an accident involving a pedestrian. From the past cases and the considered case, we collected seven facts: 1) the driver was not speeding, 2) the driver was speeding, 3) the pedestrian was outside the crosswalk, 4) the driver was drunk, 5) the accident happened at night, 6) the driver was a young driver, 7) the accident happened on the highway, 8) the crosswalk light was red.

To simplify, we assume two past cases: $\mathbf{x}_1 = \{2,4,5,7\}$, $\mathbf{x}_2 = \{1,3,5,6\}$ with $y_1 = 1$, $y_2 = -1$. We are interested in y_3 knowing $\mathbf{x}_3 = \{2,5,6,8\}$. The partition is given by $\mathcal{E} = \{\mathbf{e}_1 = \{2,4,7\}, \mathbf{e}_2 = \{5,6\}, \mathbf{e}_3 = \{1,3\}\}$. In particular, $\pi(\mathbf{x}_3) = \{\mathbf{e}_1,\mathbf{e}_2\}$.

We assume $\mu^{(1)} = (0.75, 0.25, 0)$ and $\mu^{(-1)} = (0, 0.3, 0.7)$ s.t. after normalization $\mu = (0.5, -0.03, -0.46)$.

$$\begin{aligned} s(\mathbf{x}_3) &= w(\mathbf{e}_1,\mathbf{x}_3)\mu_1 + w(\mathbf{e}_2,\mathbf{x}_3)\mu_2 \\ &= \frac{1}{4}0.5 - \frac{1}{2}0.03 = 0.11 > 0 \end{aligned}$$

The model concludes that the driver is guilty. The fact the crosswalk light was red is not taken into account. The decision can be explained mostly by the features of $\mathbf{e}_1$. The elements of $\mathbf{e}_2$ are not discriminative enough to reverse the judgement. If the driver's lawyer would highlight the fact his client is not responsible because of the outcome of $\mathbf{x}_2$ and the fact it shares many similarities, the defense could argue that the main reason $y_2 = -1$ holds in $\mathbf{e}_3$. By building a chain of analogies and counter-examples, each decision can be explained w.r.t. past cases.

6.2.4 Step 2 - Training

At this stage, it is already possible to generate predictions for new cases. However, the intrinsic strength vector calculated on the hypergraph might not be perfectly accurate on the training set because of the lack of information contained in the training set (or some limitations on the model space itself that we will discuss in Section 6.5). In this section, we give an algorithm to adjust the intrinsic strength in order to correct the initial estimation.

Once the model is built, it can be evaluated on the training set. Analogously to SVM, we define the margin as the distance to the correct class, i.e. $m(w,\mu,\mathbf{x}) = J(\mathbf{x})s_{w,\mu}(x)$. To improve the pertinence of the strength of the elements of $\mathcal{E}$, we use the iterative algorithm described by Algorithm 2 to minimize the total margin over the training set $\mathbf{X}$.

Algorithm 2 Model training.

Input:
- **X**: training set
- **y**: correct labels for **X**
- k: number of training iterations
- $\mu^{(1)}, \mu^{(-1)}$: weights calculated with (6.2.4)

Output:
- Modified vectors $\mu^{(1)}, \mu^{(-1)}$

```
1: for $k$ iterations do
2:   for $\mathbf{x}_i \in \mathbf{X}$ do
3:     $\bar{y}_i \leftarrow \bar{J}(\mathbf{x}_i)$
4:     if $\bar{y}_i \neq y_i$ then
5:       for $\mathbf{e} \in \pi(\mathbf{x}_i)$ do
6:         $\mu^{(y_i)}(\mathbf{e}) \leftarrow \mu^{(y_i)}(\mathbf{e}) + w(\mathbf{e}, \mathbf{x}_i)|\mu(\mathbf{e})|$
7:         $\mu^{(\bar{y}_i)}(\mathbf{e}) \leftarrow \mu^{(\bar{y}_i)}(\mathbf{e}) - w(\mathbf{e}, \mathbf{x}_i)|\mu(\mathbf{e})|$
8:       end for
9:     end if
10:  end for
11: end for
```

The order in which points are considered is fixed in the current implementation. When a decision is incorrect for **x**, the algorithm modifies each element of the projection by lowering its strength for the wrong class and increasing it for the proper class. The margin is split between the element of the projection w.r.t. their respective weight in **x** i.e. $w(\mathbf{e}, \mathbf{x})$. If a case **x** is wrongly classified, it is due to the cases intersecting with it. Indeed, if **x** was not intersecting with any other example, its projection would be itself, and its support toward the wrong class would be 0 and positive for the real class. In other words, **x** would be correctly classified. Thus, the idea is not to directly bring the support of x to the correct class but to gradually adjust the weights s.t. the neighbors are modified enough for **x** to be correctly classified. In particular, it is sensitive to the order in which the cases are considered: a modification in the strength of any $\mathbf{e} \in \mathcal{E}$ impacts all cases in which it appears and potentially changes the predicted class for those cases.

Proposition 6. *Algorithm 2 is guaranteed to converge.*

Proof: For a training iteration k, let assume a case $\mathbf{x}_i$ with a strength $S_k(\mathbf{x}_i)$ that is wrongly classified. The update rule on μ implies that $S_{k+1}(\mathbf{x}_i) \in [-|S_k(\mathbf{x}_i)|, S_k(\mathbf{x}_i)]$. Any case $\mathbf{x}_j$ such that $\mathbf{x}_i \cap \mathbf{x}_j \neq \emptyset$ is modified, and in the same direction (toward the same class). Let us assume that $S_k(\mathbf{x}_i) < 0$ such that after the update rule $S_{k+1}(\mathbf{x}_i) > S_k(\mathbf{x}_i)$. Then, $S_{k+1}(\mathbf{x}_j) > S_k(\mathbf{x}_j)$.

The only problematic case is when $y_j = 0$, $S_k(\mathbf{x}_j) < 0$ but $S_{k+1}(\mathbf{x}_j) > 0$ (that is to say that $\mathbf{x}_j$ become wrongly classified due to the modification of μ involved by $\mathbf{x}_i$).

Let us consider any μ_l for a e_l that is included in $\mathbf{x}_i$ and $\mathbf{x}_j$. When μ_l is modified, then $|S(x_j)|$ is smaller because by definition $|w(e_l, x_j)\mu_l| \leq |S(x_j)|$.

As a result, both the case that *triggers* the modification of μ and the cases that are consequently modified have a strength that is closer to 0 than before the modification.

Therefore, there are only two possible cases:

- All cases become correctly classified and the process stops.
- Some cases cannot be properly classified within the model space and switch iterations after iterations between classes. Their strength converges toward 0. It does not imply that the process converges toward the best possible accuracy. □

The update rule being a contracting mapping because $|\mu(\mathbf{e})| < 1$ at the initial step and $w(\mathbf{e},\mathbf{x}_i) < 1$ unconditionally, Algorithm 2 is guaranteed to converge. However, too many iterations may lead to overfitting μ to the training set. Empirical experiments suggest that the result after one to five training iterations is (near-)optimal (see Section 6.5).

6.2.5 Step 3 - Decision rule refinement

This step is not mandatory in a sense that the model can be built and already generates predictions. The hyperparameters introduced in this section can help either to increase prediction accuracy (see Section 6.3.3.1) or control the risk associated to a prediction (see Section 6.5.3).

The measure μ is defined as the difference of support for both classes. Thus, by linearity we can rewrite

$$\begin{aligned} s(\mathbf{x}) &= \sum_{i=1}^{M} w(\mathbf{e}_i,\mathbf{x})\mu^{(1)}(\mathbf{e}_i) - \sum_{i=1}^{M} w(\mathbf{e}_i,\mathbf{x})\mu^{(-1)}(\mathbf{e}_i) \\ &= s^{(1)}(\mathbf{x}) - s^{(-1)}(\mathbf{x}) \end{aligned} \tag{6.12}$$

This form is convenient because we can control how much evidence we need to support a specific class using the following constraints and a family η of four hyperparameters:

$$s^{(-1)}(\mathbf{x}) > \max(\frac{\bar{\eta}_{-1}}{1-\bar{\eta}_{-1}} s^{(-1)}(\mathbf{x}), \eta_{-1}) \geq 0 \tag{C_0}$$

$$s^{(1)}(\mathbf{x}) > \max(\frac{\bar{\eta}_{1}}{1-\bar{\eta}_{1}} s^{(-1)}(\mathbf{x}), \eta_{1}) \geq 0 \tag{C_1}$$

with $\eta_{-1},\eta_1 \in \mathbb{R}^+$ and $\bar{\eta}_{-1},\bar{\eta}_1 \in [0,1]$. The constraints on η_{-1} and η_1 define a minimal amount of support respectively toward class -1 and 1 while $\bar{\eta}_{-1}$ and $\bar{\eta}_1$ requires the support toward a class to be significantly higher than the support for the other class. As μ is normalized over $\mathcal{E}$, the value of η_{-1} and η_1 must be set w.r.t. the hypergraph. On the contrary, $\bar{\eta}_1$ and $\bar{\eta}_0$ can be set independently of the hypergraph.

Those constraints may be used to design a decision rule for new cases depending on the application or the dataset. The most generic decision rule is as follows:

$$\bar{J}(\mathbf{x}) = \begin{cases} 1 & s(\mathbf{x}) > 0 \quad \text{and } C_1 \\ -1 & s(\mathbf{x}) \leq 0 \quad \text{and } C_0 \\ l_1 & s(\mathbf{x}) > 0 \quad \text{and not } C_1 \\ l_{-1} & s(\mathbf{x}) \leq 0 \quad \text{and not } C_0 \end{cases} \tag{R2}$$

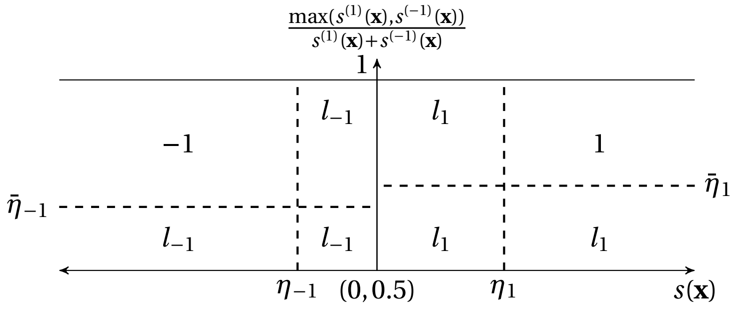


Figure 6.3: Representation of the updated decision rule (R2) in the extended decision space.

where l_{-1}, l_0 are two labels. A representation is given by Figure 6.3. Those hyperparameters are intended to model the "burden of proof". For instance, in a trial, one is assumed innocent until proven guilty which implies the support for the class "guilty" must be *beyond a reasonable doubt* (where the term reasonable is defined by the jurisprudence of the applicable country). In case $\eta_{-1} = \eta_1 = \bar{\eta}_{-1} = \bar{\eta}_1$ (and $l_{-1} = 0$ and $l_1 = 1$), then the decision rule is equivalent to the original one defined by (R1).

The whole chapter 7 is dedicated to extend these decisions rules to obtain continuous decision surfaces,

6.2.6 Justifying the model space using a mixture model

In this section, we provide a more theoretical justification to the model space and in particular that the model space will converge in law toward a certain random variable with the size of the training set. For each $\mathbf{x}_i \in \mathbf{X}$, we associate a random variable Y_i for which we have only a single observation given by its outcome y_i. The idea behind is that, despite the fact we can observe only one realization of each random variables, they are not independent and we will estimate the distribution of probability using elements shared by their respective σ-algebra. First, we give some generalities about σ-algebras.

Definition 6.2.5 (σ-algebra on a set X)**.** *$\mathcal{A}$ is a σ-algebra on a non-empty set X is a element of the powerset of X with the following properties:*

- **non-empty:** $\mathcal{A} \neq \emptyset$
- **closed under complementation:** $\forall A \in \mathcal{A},\ X \setminus A \in \mathcal{A}$
- **closed under countable unions:** $\forall n \in \mathbb{N},\ A_n \in \mathcal{A},\ \bigcup\limits_{n\in\mathbf{N}} A_n \in \mathcal{A}$

In particular, $\mathcal{P}(X)$ is a σ-algebra on X. Given $x \in \mathcal{P}(X)$, the σ-algebra generated by x, noted $\sigma(x)$, is the smallest σ-algebra on X containing all the elements of x.

Properties 7 (σ-algebra generated by a partition)**.** *Given a (countable) partition $\varepsilon = \{e_n \mid n \in \mathbb{N}\}$ of X, $\mathcal{X} = \{\bigcup\limits_{i\in I} e_i \mid I \subseteq \mathbb{N}\}$ is a σ-algebra. In particular $\mathcal{X} = \sigma(X)$.*

Properties 8. *The set of σ-algebras on X countable is exactly the set of σ-algebras generated by the partition of X.*

Given a hypergraph $H = (V, \mathbf{X})$, we have ε, the partition obtained by the projection operator π_H. Therefore, not only $\sigma(\varepsilon)$ is a σ-algebra on $\mathbf{X}$, but $\forall \mathbf{x}_i \in \mathbf{X}$, $\sigma(\pi(\mathbf{x}))$ defines a σ-algebra on $\mathbf{x}$.

We consider the vector of outcomes $(y_1, ..., y_n)$ as an observation of the vector $(Y_1, ..., Y_n)$ where the Y_i are not supposed to be identically distributed and certainly not independent. Each Y_i is defined over a probability space $(\Omega_i, \sigma(\pi(\mathbf{x}_i)), \mathbb{P}_i)$ to $(\{0,1\}, \mathcal{A})$, the measurable set of outcomes. The Y_i are random variables modeling the decision of a given case.

Given a case $\mathbf{x}_i$, we consider the random variable Y as mixture model such that the density of Y_i can be defined by $f_{Y_i} = \sum\limits_{e \in \mathcal{P}(\mathbf{x}_i)} \omega_e p_i(e)$ such that $\sum_e w_e = 1$. The rational behind is that all the possible combinaisons of features participate in the final outcome.

In practice, we do not know $p_i(e)$ for all the elements of $\mathcal{P}(\mathbf{x}_i)$ but we can rely on realizations that are materialized by the intersection family represented by the hypergraph to estimate some $p_i(.)$.

Also, if a case can be seen as a random variable whose density is a mixture model of its generated σ-algebra, we can also consider that every element of this algebra $e \in \mathcal{P}(\mathbf{x}_i)$ (more generally $e \in \mathcal{P}(\bigcup\limits_i \mathbf{x}_i)$) is theoretically a case and can be also associated to an underlying random variable Y_i^e. In other words, the familly p_i is the probability density associated to a random variable Y_i^e defined for each element of $\mathcal{P}(\mathbf{x}_i)$. Contrarely to $\mathbf{x}_i$, we do not have a single observation of X_i^e. However, our goal is to deduce some information about the X_i from the intersection of the several realizations.

Let $\pi_k(\mathbf{x})$ denotes the projection operator on a hypergraph when there are exactly k hyperedges, i.e. $|\mathbf{X}| = k$.

Properties 9. $\forall m \leq +\infty$, *s.t.* $\forall i \in \mathbb{N}, 1 \leq |\mathbf{x}_i| \leq m$, $\forall j \leq i$, $\lim\limits_{k \to \infty} \sigma(\pi_k(\mathbf{x}_i)) = \mathcal{P}(\mathbf{x}_i)$

Proof: We suppose here that m is a bound on the number of features. An hypothesis is that each feature has a non-null probability to appear after a certain number of cases (otherwise it is useless information and it is not part of $\mathbb{F}$). In particular, each feature will appear in the intersection of a certain amount of cases if we add enough cases, including the features of a particular case $\mathbf{x}_i$. $\square$

We define $f_{\tilde{Y}_{i,k}} = \sum\limits_{e \in \sigma(\pi_k(\mathbf{x}_i))} \tilde{w}_e^k p_j(x)$ with $\sum_e \tilde{w}_e^k = 1$ avec $\forall k$, $\forall j \in \sigma(\pi_k(\mathbf{x}_j))$, $\tilde{w}_j^k \xrightarrow{k} w_j$. The density $f_{\tilde{Y}_{i,k}}$ is an approximation of the real density $f_{Y_{i,k}}$ that can be made given a hypergraph $\{\mathbf{x}_i\}_{1 \leq i \leq k}$.

Properties 10. $\forall i, \tilde{Y}_{i,k} \xrightarrow{\mathcal{L}} Y_i$

Proof: Consider the difference of density $A_k(x) = |f_{Y_i}(x) - f_{\tilde{Y}_{i,k}}(x)|$. $A_k(x) = |\sum\limits_{j \in \sigma(\pi_k(\mathbf{x}_j)} (w_j - \tilde{w}_j^k) p_j(x) + \sum\limits_{j \in \mathcal{P}(\mathbf{x}_i) \setminus \sigma(\pi_k(\mathbf{x}_i)} w_j p_j(x) \quad |$ but $\mathcal{P}(\mathbf{x}_i) \setminus \sigma(\pi_k(C_i) \underset{k}{\rightarrow} \emptyset$ and $\tilde{w}_j^k \xrightarrow{k} w_j$ due to the Property 9. Thus $A_k(x) \rightarrow 0$. The Scheffé's lemma ensures the convergence in law $\tilde{Y}_{i,k} \xrightarrow{\mathcal{L}} Y_i$ $\square$

Intuitively, it can be understood as the fact that, by adding more cases, we decrease the coarsity of the partition ε and converges toward the largest possible σ-algebra, i.e. the power set of $\mathbb{F}$. As a result, if the real density of Y_i behaves like a convex combination of the elements of the powerset of $\mathbf{x}_i$, then the approximation obtained by the convex combination defined by the elements of the partition induced by the hypergraph will converge in law, **no matter the unknown measures μ and $\mathbf{P}_i$** and thus, no matter the exact unknown mapping and the domain.

There is, however, a problem with space complexity since the model space would grow exponentially due to the powerset size, and with time complexity in the estimation as well, for the exact same reason. For this reason, in HCBR, we restricted the model space to the convex combination of the elements of the partitions defined by the projection and not by its powerset (which is described by Equation (6.2)). As demonstrated by the experiments, this approximation might be too loose and lead to to simple models. Future work will focus in addressing this problem and quantifying the impact of the approximation on the convergence.

6.2.7 Time complexity

Model Building: Given $\mathbf{X} \in (2^{\mathbb{F}})^N$, constructing $\mathcal{E}_H$ can be done in $\mathcal{O}(\sum_{\mathbf{x} \in \mathbf{X}} |\mathbf{x}|)$ by using a Partition Refinement data structure [Paige 1987]. Given $\mathbf{x} \in \mathbf{X}$, calculating the family $\{\bar{S}(\mathbf{e}, \mathbf{x})\}_{\mathbf{e} \in \mathcal{E}_H}$ can be done in $\mathcal{O}(|\mathbf{x}|)$ by asking for each feature of $\mathbf{x}$ the $\mathbf{e}$ it belongs to and maintaining the size of each $\mathbf{e}$ during the construction of $\mathcal{E}_H$. Thus, calculating $\{\bar{S}(\mathbf{e}, \mathbf{x})\}_{\mathbf{e} \in \mathcal{E}_H}$ for all $\mathbf{x} \in \mathbf{X}$ can be done in $\mathcal{O}(\sum_{\mathbf{x} \in \mathbf{X}} |\mathbf{x}|)$. On m-uniform hypergraphs (when all cases are described with m features), it becomes $\mathcal{O}(mN)$.

Calculating $\{S(\mathbf{e})\}_{\mathbf{e} \in \mathcal{E}_H}$ and μ can be done in $\mathcal{O}(|\mathcal{E}_H|)$ because it requires to iterate over $\mathcal{E}_H$. An obvious upper bound on $|\mathcal{E}_H|$ is $|\mathbb{F}_\mathbf{X}|$ i.e. the number of vertices in the hypergraph. The worst-case cardinal of $\mathcal{E}_H$ is when each $\mathbf{x} \in \mathbf{X}$ intersects with all the others and none of them is strictly a subset of any other. Thus, $|\mathcal{E}_H| \leq \min(2^N - 1, |\mathbb{F}_\mathbf{X}|)$.

Learning Phase: For each wrongly classified $\mathbf{x} \in \mathbf{X}$, a training iteration requires at most $\mathcal{O}(|\mathbf{x}|)$ steps (maximal cardinal for $\pi(\mathbf{x})$). The worst-case scenario is when the model wrongly classifies every $\mathbf{x} \in \mathbf{X}$. Thus, the learning phase worst-case complexity is $\mathcal{O}(k \sum_{\mathbf{x} \in \mathbf{X}} |\mathbf{x}|)$ and on m-uniform hypergraphs it becomes $\mathcal{O}(kmN)$.

Model Query: For a case $\mathbf{x} \in 2^{\mathbb{F}}$, the projection can be done in $\mathcal{O}(|\mathbf{x}|)$. Calculating the classification rule also requires at most $\mathcal{O}(|\mathbf{x}|)$ (maximal cardinal for $\pi(\mathbf{x})$).

6.3 Experiments on structured datasets

In this section, we validate HCBR on well-known structured datasets. Two series of experiments are performed. The first one compares HCBR to the best results from the literature and includes hyperparameter tuning. The second one focuses on the robustness, i.e. the capacity to deliver good performances on a wide range of datasets without spending time on feature engineering, data preprocessing or hyperparameter tuning.

Table 6.1: Datasets description.

	Cases	Total features	Unique	Min. size	Max. size	Avg. size	Real	Prev.
adult	32561	418913	118	10	13	12.87	No	0.7586
breasts	699	5512	80	8	8	8	No	0.3338
heart	270	3165	344	12	13	12.99	Yes	0.5107
mushrooms	8124	162374	106	20	20	20	No	0.4804
phishing	11055	319787	808	29	29	29	No	0.5562
skin	245057	734403	768	3	3	3	Yes	0.2075
splice	3175	190263	237	60	60	60	No	0.5164

Some specific elements such as the learning curves or computing times are discussed in Section 6.5 dedicated to intrinsic performances and properties.

6.3.1 Data and method

We used seven structured datasets for binary classification. All of them are available either from the UCI Machine Learning Repository[4] or provided with LIBSVM[5] : `adult`, `breasts`, `heart`, `mushrooms`, `phishing`, `skin` and `splice`. For each dataset, the original features (`name=value`) are converted into a unique identifier and the union of all such identifiers constitutes the information set $\mathbb{F}$.

The datasets are described by Table 6.1. The minimal, maximal and average size give information about the case sizes (notice `adult`, `heart` and `mushrooms` datasets are missing values). The column unique reports the cardinal of $\mathbb{F}$. Two datasets have at least one real-valued attribute as indicated by the column "Real". Three datasets (`adult`, `breasts` and `skin`) are highly imbalanced.

For both experiments, we saved the confusion matrix obtained over all runs and after each prediction. From this confusion matrix, we calculated standard performance indicators: accuracy, recall, specificity, precision, negative prediction value, F_1-score and Matthews correlation coefficient. Denoting by TP the number of true positives, TN the true negatives, FP the false positives and FN the false negative, the accuracy, the F_1-score and Matthews correlation coefficient (MCC) are defined by:

$$\mathrm{ACC} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$$

$$\mathrm{F}_1 = \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$$

$$\mathrm{MCC} = \frac{\mathrm{TP}\times\mathrm{TN}-\mathrm{FP}\times\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$$

The accuracy, F_1-score and MCC respectively belongs to $[0,1]$, $[0,1]$ and $[-1,1]$. The closer to 1, the better it is. F_1-score and MCC take into account false positive and false negatives.

[4]https://archive.ics.uci.edu/ml/index.php

[5]`https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`

Furthermore, MCC has been shown to be more informative than other metrics derived from the confusion matrix [Chicco 2017], in particular with imbalanced datasets.

6.3.1.1 Experiment 1 – Literature comparison

The objective is to compare HCBR performance to the best results from the literature. As most studies do not report F_1-score nor MCC, we will base our comparison on the accuracy.

We used a 10-fold cross-validation with stratified sample to preserve the original dataset prevalence. The number of training steps k was adjusted with a manual trial-and-error approach.

To measure the impact of hyperparameters on HCBR, the runs have been performed twice (with the same seed). The first campaign was done with the family η sets to $\eta_{-1} = \eta_1 = \bar{\eta}_{-1} = \bar{\eta}_1 = 0$. The second one was done with a nested 10-fold cross-validation in order to tune hyperparameters. The training set was itself divided into 10 folds. The first nine folds were used to build and train the model, the last one used as validation set to find the optimal values for the family η. On the test set, we used for η the average values found over the 10 validation sets.

It is difficult to know in advance the range of values for the components of μ such that setting a grid is not convenient. Instead, we define a hyperparameter space that depends on the strength expressed in the decision defined by Figure 6.3. Consider the upper half-plane of this space, i.e. the points with positive support. The hyperparameters can define a vertical split, horizontal split or another (open) half-plane with a certain margin from the axes. We performed an exhaustive search for the family η. We describe here the procedure for cases with a positive support i.e. to find η_1 and $\bar{\eta}_1$. The procedure is symmetric for negative support.

1. For each point (x, y) from the validation set, we considered three points: $p_1 = (x+\epsilon, y+\bar{\epsilon})$, $p_2 = (x+\epsilon, 0)$ and $p_3 = (0, y+\bar{\epsilon})$ where ϵ and $\bar{\epsilon}$ are positive or negative depending on whether (x, y) is correctly classified or not and set to half the distance to the immediate neighbor of (x, y).
2. For each point of those three points, we recalculated the accuracy when $(\eta_1, \bar{\eta}_1)$ is sets to p_i
3. We choose the values that return the highest accuracy.

6.3.1.2 Experiment 2 – Robustness comparison

The state-of-the-art results obtained per dataset required a lot of efforts in terms of feature engineering, as well as model selection or hyperparameter tuning. As discussed in Section 6.1.1, it represents nowadays most data scientists work and CPU time. If well-calibrated models with high performances on their domain of application is undeniably useful for practical usage, being able to solve a problem on a large variety of instances without effort is also of a vital importance, especially in the industry where the end-user might not be specialized in data science and machine learning. One main-challenge of large scale machine learning systems is thus to achieve the compromise between ready-to-deploy models and good performance metrics.

There exist tools such as auto-sklearn [Feurer 2015] to automatically tune hyperparameters and select the best algorithm in a given portfolio. However, this approach still requires an extensive CPU time which might be prohibitive in case end-users need to build classification models over numerous datasets. For instance, one may think to cloud offers such as IBM Cloud[6] or Amazon AWS[7] that create for their customers thousands of models from scratch every day. Having models that give good results with limited CPU time dedicated to feature engineering and hyperparameter is thus an important competitive advantage. On top of that, Automated Machine Learning is far from being widely adopted in the industry where hyperparameter tuning might not even be done for several reasons, as noted by [Couronné 2018] and confirmed by our experience.

Therefore, we would like to quantify how good HCBR can perform on different datasets without feature engineering or hyperparameter tuning. To measure this robustness, we performed a 10-fold cross-validation for each of the seven selected datasets using nine standard classification methods: AdaBoost, k-Nearest Neighbors, Linear SVM, Radius-Based Function (RBF) SVM, Decision Tree, Random Forest, Neural Network and Quadratic Discriminant Analysis (QDA). The implementation is provided by Scikit-Learn [Pedregosa 2011]. No hyperparameter tuning was performed and the default values of parameters were used.

6.3.2 Previous work on the datasets

To compare the results of the proposed method, we explored for each dataset the best results from the literature. The results are summarized in Table 6.2. In [Datta 2016], 5 rule-based classification techniques dedicated to medical databases are compared and achieve at best 95.85% and 82.96% accuracy resp. on `breast`, and `heart` datasets. Comparing bayesian approaches, [Jiang 2012] demonstrated 97.35% (`breast`) and 83.00% (`heart`) accuracy. A 5 layers neural network with fuzzy inference rules achieved 87.78% on `heart` [Sagir 2017] while a k-NN algorithm reached 99.96% on `mushrooms` [Das 2001]. The best alternative among 6 rules-based classification methods achieved 95.84% on `breast` and 100.00% on `mushroom` [Hadi 2017]. Using 80% of `phishing` as training set, an adaptative neural network achieved an average accuracy of 93.76% (among 6 alternatives) with the best run at 94.90% [Thabtah 2016]. Still on `phishing`, [Tahir 2016] proposes to combine several classifiers and reaches 97.75% accuracy for the best hybrid model (and demonstrates 97.58% for Random Forest classifier). On `adult`, the comparison of several classifiers (naive bayes, decision tree, ...) demonstrated at most 86.25% accuracy [Kou 2012] while a Support Vector Machine approach reached 85.35% [Lee 2001]. On `splice`, a method using Fuzzy Decision Trees [Bhatt 2009] reaches 94.10% accuracy and a neural network combined to boosting [Çatak 2017] 97.54%. On `breast`, Support Vector Machine approaches reached resp. 96.87%, 98.53%, 99.51% accuracy [Chen 2011, Polat 2007, Akay 2009], 99.26% and 97.36% for neural network based techniques [Marcano-Cedeño 2011, Übeyli 2007], 98.1% for a bayesian network method [Fallahi 2011], or 94.74% using Decision Trees [Quinlan 1996]. On `skin`, [Çatak 2017] reports 98.94% accuracy against 99.68% for Decision Tree based method [Cazzolato 2013]. The best result, as far as we know, is 99.92%, obtained by a Gener-

[6] `https://www.ibm.com/cloud/`
[7] `https://aws.amazon.com/`

alized Linear Model [Basterrech 2015].

Table 6.2: Previous literature results measured as the highest accuracy obtained by the authors.

Dataset	Ref.	Type	Accuracy
adult	[Kou 2012]	Many classifiers	86.25%
	[Lee 2001]	SVM	85.35%
		HCBR(tuned)	**82.90%**
		HCBR	**82.06%**
breast	[Akay 2009]	SVM	99.51%
	[Marcano-Cedeño 2011]	Neural Net	99.26%
	[Polat 2007]	SVM	98.53%
	[Fallahi 2011]	Bayes	98.1%
		HCBR(tuned)	**97.83%**
	[Übeyli 2007]	Neural Net	97.36%
	[Jiang 2012]	Bayes	97.35%
		HCBR	**96.96%**
	[Chen 2011]	SVM	96.87%
	[Datta 2016]	Rule-based	95.85%
	[Hadi 2017]	Rule-based	95.84%
	[Quinlan 1996]	Decision Tree	94.74%
heart		**HCBR(tuned)**	**90.77%**
	[Sagir 2017]	Neural Network + Rule-based	87.78%
		HCBR	**85.77%**
	[Jiang 2012]	Bayes	83.00%
	[Datta 2016]	Rule-based	82.96%
mushrooms	[Hadi 2017]	Rule-Based	100.00%
		HCBR	**100.00%**
	[Das 2001]	k-NN	99.96%
phishing	[Tahir 2016]	Ensemble	97.75%
	[Tahir 2016]	Random-Forest	97.58%
		HCBR(tuned)	**96.82%**
		HCBR	**96.05%**
	[Thabtah 2016]	Neural Net	94.90%
skin	[Basterrech 2015]	Generalized Linear Model	99.92%
	[Cazzolato 2013]	Decision Tree	99.68%
	[Çatak 2017]	Neural Network + Boosting	98.94%
		HCBR(tuned)	**98.68%**
		HCBR	**98.65%**
splice	[Çatak 2017]	Neural Network + Boosting	97.54%
		HCBR(tuned)	**95.09%**
		HCBR	**94.43%**
	[Bhatt 2009]	(fuzzy) Decision Tree	94.10%

6.3.3 Results

The entirety of the data used for the experiments, as well as the scripts to transform them and analyze the results are available within the HCBR Github repository[8] s.t. the whole experimental campaign starting from the raw data can be easily be reproduced.

6.3.3.1 Literature comparison

The average confusion matrix obtained for each dataset is showed in the Appendix C.1. The performance indicators are reported in Table 6.3. The proposed algorithm performs very well on a wide range of datasets as reported by the Appendix C.1, Appendix C.2, C.3, and Table 6.3.

Without hyperparameter tuning: The accuracy is contained in a range from 0.8206 (`adult`) to 1 (`mushrooms`) while the F_1-score is bounded by 0.8653 (`heart`) and 1 (`mushrooms`). On `adult`, the accuracy is only 6% higher than the prevalence, i.e. a baseline model consisting in returning 1 for any point would be only 6% worse. This relatively poor performance in learning the underlying decision mapping is better reflected by the Matthews correlation coefficient of 0.51.

Table 6.3: Average performances obtained with a 10-fold cross-validation.

	Accuracy (std dev.)	Recall	Specificity	Precision	Neg. Pred. Value	F_1 score	MCC
`adult`	0.8206 (0.0094)	0.8832	0.6233	0.8808	0.6290	0.8820	0.5081
	0.8290 (0.0063)	0.9008	0.6029	0.8773	0.6029	0.8889	0.5194
`breasts`	0.9696 (0.0345)	0.9691	0.9676	0.9479	0.9844	0.9575	0.9344
	0.9783 (0.0204)	0.9553	0.9910	0.9833	0.9910	0.9691	0.9526
`heart`	0.8577 (0.0943)	0.8695	0.8437	0.8699	0.8531	0.8653	0.7178
	0.9077 (0.0659)	0.9310	0.8783	0.9060	0.8783	0.9184	0.8126
`mushrooms`	1.0000 (0.0000)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
`phishing`	0.9605 (0.0081)	0.9680	0.9514	0.9615	0.9590	0.9647	0.9199
	0.9682 (0.0067)	0.9689	0.9672	0.9741	0.9672	0.9715	0.9355
`skin`	0.9865 (0.0069)	0.9608	0.9932	0.9736	0.9898	0.9672	0.9587
	0.9868 (0.0062)	0.9740	0.9900	0.9618	0.9900	0.9679	0.9596
`splice`	0.9443 (0.0124)	0.9478	0.9398	0.9450	0.9441	0.9463	0.8884
	0.9509 (0.0108)	0.9478	0.9544	0.9577	0.9544	0.9527	0.9018

The false positives and false negatives are equilibrated for each dataset, despite a huge variation in the prevalence (between 20% and 64%, cf. Table 6.1) which is a desirable property as it is known to be a problem for many machine learning algorithms [He 2009].

The support is a metric of confidence for the prediction as illustrated in Figure 6.4. In general, the wrongly classified cases have a smaller difference between the evidence for each class. This can also be observed in Figure 6.5.

HCBR performs the best on `mushrooms` and `heart` datasets. For the rest, the accuracy is slightly lower than the best results from the literature (`adult` 82.90% against 86.25%, `breast` 97.83% against 99.51%, `phishing` 96.82% against 97.75%, `splice` 95.09% against 97.54%,

[8]https://github.com/aquemy/HCBR

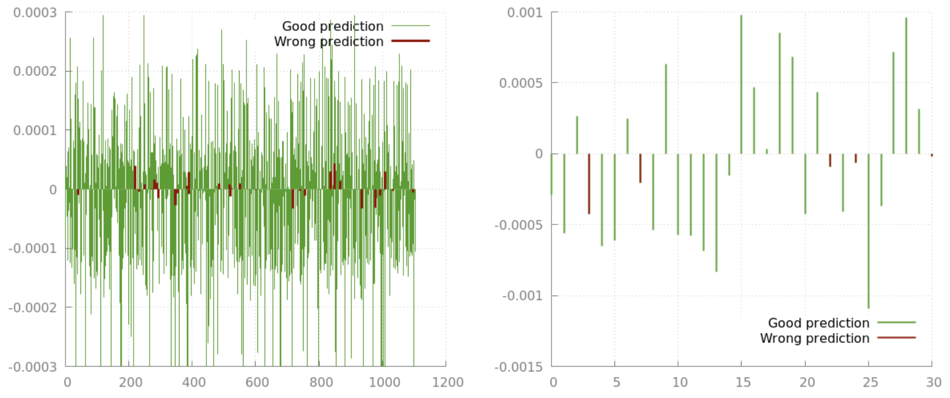


Figure 6.4: Difference between the weight assigned to both classes for each decision on `phishing` and `splice` (average). Similar results are observed for all datasets.

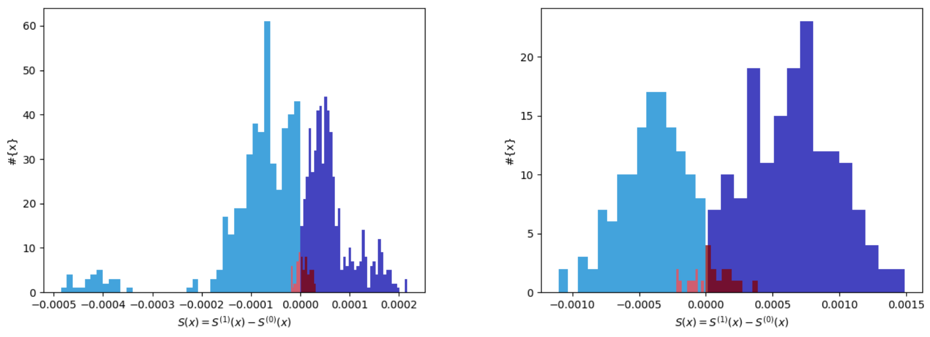


Figure 6.5: Histogram of decisions depending on the strength for `phishing` and `splice`. In blue the correctly classified elements, in red the wrongly classified ones. The false positives and false negatives are concentrated around 0. Similar results are observed for all datasets.

`skin` 98.68% against 99.92%). We explain this by at least two factors. First, the best methods on a given dataset are often dedicated to this dataset with *ad-hoc* or engineered parts which is not the case of HCBR. Secondly, the learning curves study in Section 6.5.2.1 reveals that the model space is not complex enough. HCBR performed better than Bayes classifier in two thirds of cases. Bayes classifier performs better on `breast` by approximately 1% which represents less than one case wrongly classified. Similar results are observed with Decision Trees. However, the 1% difference on `skin` represents an average of 7 cases misclassified in comparison in favor of Bayes. It performs better than Rule-based approaches (or gives similar results on `mushrooms` with an accuracy of 1) in the four considered references on three different datasets. Except for `heart` and `phishing`, Neural Network returns better results (0.46 more cases with correct classification in average for `breast`, 71 for skin and almost 10 for `splice`). Last, SVM gives better results in all three cases, but appear only as best results in two datasets.

With hyperparameter tuning: An illustration of parameters tuning on a real instance is depicted by Figure 6.6. With hyperparameter tuning, the accuracy lower bound is obtained by the same dataset (0.8290 with `adult`). The lower bound on F_1-score is now obtained by `adult` with 0.8889. On `heart`, the accuracy gains 5 percent point (pp) which represents a 35% error reduction. This allowed HCBR to rank first. On `breast`, the gain represents 0.87pp which is about 29% error improvement and a higher accuracy than [Jiang 2012] and [Übeyli 2007]. For the other datasets, the accuracy variation does not result in a better rank and the error improvement lies between 2% for `skin` and 19% for `phishing`.

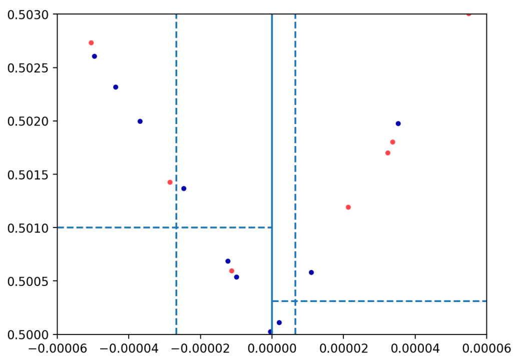


Figure 6.6: Illustration of the hyperparameter optimization on `splice`.
One specific run, zoomed in the origin. The dashed lines are defined by averaging the best parameters obtained on the test set (10 folds). For negative support, it turned 4 FN into FP and 1 TN into FP compared to the version without hyperparameter tuning. For positive support, it turned one FP into TN. The hyperparameter values are not optimal as a better result could have been achieved.

We explain the limited effect of hyperparameters by the fact that for small support values, there can be several cases with the exact same support but different outcomes. Two different cases have the same support if the elements of their projections are weighted the same despite being composed of different features. In order to be more discriminative, one can increase the model space complexity s.t. the support is obtained by more variables which decreases the probability of collisions. See Section 6.5.2 for a study of the model space limitations and possible remedies.

Discussion: Even if HCBR does not rank first in most cases, we see at least three reasons to use it in practice. First, it provides consistently good results on all datasets without a need to tune any hyperparameter, without correcting the imbalanced datasets or feature transformation that would require expert knowledge. In other words, it requires absolutely no domain knowledge or data science expertise to deploy and use. Second, HCBR works directly in unstructured spaces allowing combining data from multiple sources without tedious transformations. This has a considerable advantage in practice where the information per case is rarely structured by default. Last but not least, HCBR provides local explanation: for each case and each group of features $\mathbf{e}_i$ in this case, HCBR provides not only the support for $\mathbf{e}_i$ but also all the cases that participated in increasing the support. In the medical domain, the practitioner may check, for instance, the top two groups of features, and for each, the five most influencing past cases to find a reasonable justification to the prediction. We explore this possibility in some preliminary work [Mróz 2020].

6.3.3.2 Robustness comparison

The average MCC achieved per method over the datasets is reported in Table 6.4. The details per dataset can be found in the Appendix C.2. The counterpart for the accuracy is provided by Table 6.5 and the Appendix C.3. Additionally, we displayed the evolution of MCC with the training set size in Figure C.1.

Table 6.4: Average MCC and rank obtained with several methods (Scikit-Learn implementation).

Method	MCC	Rank	Δ_{HCBR}
Neural Network	0.8914	1	5.68%
HCBR	0.8435	2	-
RBF SVM	0.8267	3	2.00%
Decision Tree	0.8066	4	4.37%
AdaBoost	0.8063	5	4.41%
k-NN	0.7859	6	6.82%
Linear SVM	0.7858	7	6.84%
QDA	0.7358	8	12.76%
Random Forest	0.7237	9	14.20%
Naive Bayes	0.6953	10	17.56%

The column Δ_{HCBR} represents the relative difference in MCC w.r.t. HCBR.

Table 6.5: Average accuracy and rank obtained with several methods (Scikit-Learn implementation).

Method	Accuracy	Rank	Δ_{HCBR}
HCBR	0.9360	1	-
Neural Network	0.9354	2	0.06%
RBF SVM	0.9247	3	1.16%
AdaBoost	0.9207	4	1.60%
Linear SVM	0.9115	5	2.62%
Decision Tree	0.9056	6	3.25%
k-NN	0.9011	7	3.81%
Random Forest	0.8903	8	4.88%
QDA	0.8802	9	6.27%
Naive Bayes	0.8235	10	13.59%

The column Δ_{HCBR} represents the relative difference in accuracy w.r.t. HCBR.

The highest MCC value is obtained by Neural Network with 0.8914 followed by HCBR with 0.8435. Neural Network improves HCBR result by 5.68% while HCBR improves the result of all other methods from 2% to 17.56%. The lowest MCC score is obtained by Naive Bayes with 0.6953, mostly because it performs poorly on some datasets (0.2493 on `adult`, 0.5292 on `phishing` or 0.7600 on `skin`). In general, Neural Network and HCBR are the only two methods whose ranks remain consistent across all datasets, close to the first and second rank. On the contrary, methods like k-Nearest Neighbors or Decision Tree performed

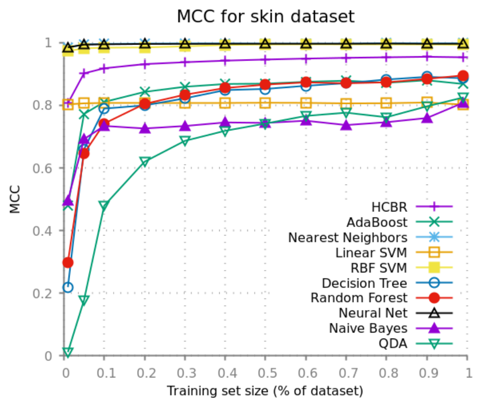


Figure 6.7: Matthew Correlation Coefficient depending on the training set size for `skin`. See the Appendix C.4 for other datasets.

very well on one dataset (`skin`, resp. `splice`). Surprisingly, Random Forest not only perform poorer than Decision Tree but also performs worse than most methods. Naive Bayes and QDA perform very poorly in general which is less surprising knowing the assumptions behind those methods that are likely to be unrealistic on real datasets.

Regarding accuracy, HCBR is the best performing algorithm followed by Neural Network. In general, the ranking is consistent with the one obtained for MCC.

In other words, HCBR and Neural Network have shown to be more versatile than the other approaches on this selected set of classification instances and without parameter tuning. We have no doubt that the results obtained with other methods can be highly improved by a proper hyperparameter selection, in particular for Random Forest or even k-Nearest Neighbors. However, this time-consuming operation can be avoided with HCBR that provides the best compromise between ready-to-use and good performances.

6.4 Experiments on unstructured datasets for text classification

In this section we study the performance of HCBR on unstructured datasets and show it performs better in most cases compared to the reference study that uses SVM.

6.4.1 Data and method

We used the European Court of Human Rights dataset provided by the authors of [Aletras 2016]. This dataset for binary classification is broken down into three articles, namely Article 3, Article 6 and Article 8. For each article, the authors collected some judgments (250, 80 and 254 respectively for Article 3, 6 and 8) s.t. the prevalence is 50%. For each document, they isolated different sections (Procedure, Circumstances, Relevant Law, Law) and excluded the verdict section. The authors do not provide the raw text but the Bag-of-Words representation obtained by keeping the 2000 most common N-grams for $N \in \{1,2,3,4\}$. The authors used a linear SVM after applying a TF-IDF schema on the dataset to predict whether an article has been violated.

On top of those Bag-of-Words representations, the authors provide the representation of a judgement in a topic space. This topic space is obtained by calculating the matrix of cosine distance between the documents and keeping the 30 top-components after applying a spectral clustering. Finally, the authors include two additional parts defined as the mean vector of other sections **after** applying the TF-IDF transformation. Facts is the mean vector of Circumstances and Relevant Law, and Full as mean of all the other parts. Thus, they are continuous contrarily to the other sections. In this work, we stacked the representations to obtain Facts and Full as HCBR does not handle properly continuous variables.

It has to be noted that for some parts, there are empty cases, i.e., without any feature. For the Law part with Article 3, there are 162 empty cases (64.8% of the total casebase) with a prevalence of 17%, for the Article 6, 52 cases are empty with 17% prevalence and for the Article 8, 146 cases are empty for a 21% prevalence. This is not compatible with the hypothesis s.t. if two cases are described by the same features, they must have the same outcome used both by HCBR and SVM. Therefore, the results are biased for both methods and could be improved by a better preprocessing step[9].

For each of the seven sections (Full, Procedure, Circumstances, Relevant law, Facts, Law, Topics), we performed a 10-fold cross-validation and reported the accuracy.

6.4.2 Results

Table 6.6 summarizes the results. The accuracy is improved in 14 cases out of 21, deteriorated in 5 and remained unchanged in 2 cases. The improvement ranges from 1 to 20pp while the deterioration ranges from 1pp to 8pp.

All sections have seen their average accuracy unchanged or increased except for Topics. This is not surprising knowing Topics is made out of 30 continuous variables which are not correctly handled by HCBR. Surprisingly, the conclusions we can draw from this experiment are quite opposed to those of the reference study [Aletras 2016]. The full text was outperformed by using only the section Circumstances while here, taking the full text returns a much higher accuracy. The section with the best predictive power was Circumstances. It is now Relevant Law. The section Law was the worst predictor, it now one of the best, notably thanks to a gain of 20pp on Article 3.

In the original study, the best performances are obtained on Sections Topics and Topics and Circumstances with an accuracy of 0.78, 0.84 and 0.78 respectively for Article 3, 6 and 8. Thus, HCBR performed 2% lower on Article 3 and 6 and 1% better on Article 8. Knowing continuous variables are not properly handled yet, it is encouraging. Also, it turns out the best section for HCBR is the full text, which implies that in practice there is no need for feature engineering or time-consuming operations such as splitting the text into subsections. We are confident that HCBR could perform better with a larger number of tokens in the bag-of-word representation and let this for future work.

[9] Ironically, this supports our claim that shortening the data preprocessing steps is necessary in order to avoid such hard-to-notice problems.

Table 6.6: Accuracy obtained by HCBR on the European Court of Human Rights dataset, depending on the article and the judgement section.

	Article 3	Δacc	Article 6	Δacc	Article 8	Δacc	Average	Δacc
Full	.76	+.06	.83	+.01	.77	+.05	.79	+.04
Procedure	.67	-	.81	-	.72	+.01	.73	-
Circumstances	.67	-.01	.81	-.01	.72	-.05	.73	-
Relevant law	.71	+.02	.86	+.08	.76	+.04	.78	+.05
Facts	.73	+.03	.76	-.04	.72	+.04	.74	+.01
Law	.76	+.20	.80	+.12	.73	+.11	.76	+.14
Topics	.70	-.08	.83	+.02	.74	-.02	.76	-.02

The columns Δacc represent the difference with the reference study [Aletras 2016]. The color indicates if the result has been improved (green), deteriorated (red) or unchanged (white).

6.5 Intrinsic performances and properties

In this section, we validate the time complexity (Section 6.5.1) and discuss the model space limitations of HCBR (Section 6.5.2). We show how the hyperparameters can be used to control the confusion matrix in Section 6.5.3.

6.5.1 Computation time

We generated a casebase of N cases of size m s.t. case i is described by $\{i, ..., i+m\}$ i.e., each case is partitioned into m elements (one discretionary feature). This is the worst-case scenario in terms of the size of $\mathcal{E}$ if $m < N$ because the family grows exponentially in function of m or N. We split the computation time into constructing the hypergraph (and determining the intersection family) and calculating the strength of the partition. The results are illustrated in Figure 6.8. By increasing N with a fixed m, the partition grows exponentially and thus, it is expected to have an exponential curve for the strength computation. On the contrary, building the hypergraph can be done in linear time when m fixed. When N is fixed and m increases, constructing the hypergraph is still doable in linear time as expected. Interestingly, calculating the strength has two phases: (1) if $m \leq N$, increasing m exponentially increases the time (because $\mathcal{E}$ exponentially increases) but (2) if $m > N$, increasing m cannot results in an exponential growth in the computation time (because $\mathcal{E}$ grows linearly).

6.5.2 Model space limitations

We now study the learning curves to show the model space limitations and propose some extensions left for future work. We also discuss the tradeoff between model locality and generalization.

6.5.2.1 Learning curves

The learning curves are useful to study the limit of the model space on different datasets. It consists in plotting the accuracy in function of the training set size for both the training and the test sets. For the training set, it is expected to observe an accuracy starting close to 1 and

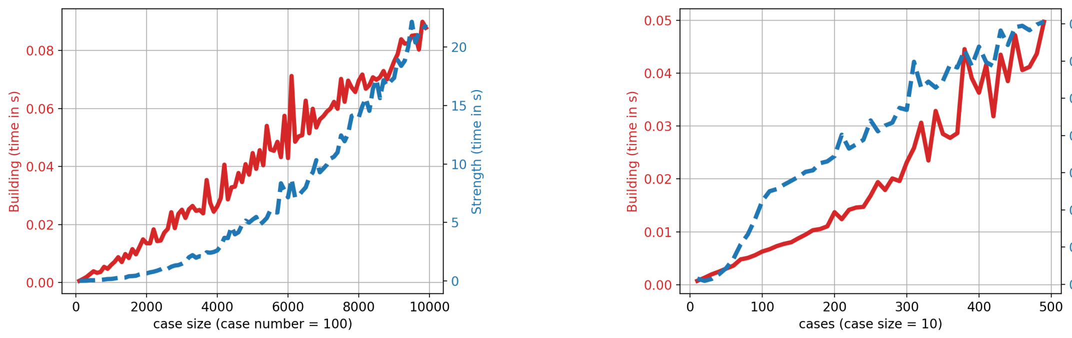


Figure 6.8: On the left, computation time to build the model (hypergraph construction + strength calculation) depending on N ($m = 10$), and on the right, depending on m ($N = 100$). The case i is described by $\{i, ..., i + m\}$ s.t. each case is partitioned into m elements (one discretionary feature).

decreasing fast to reach a plateau. A low stationary accuracy value indicates a high bias in the model or/and an irreducible error contained in the dataset, such as noisy or uninformative features. On the contrary, the accuracy on the test sets starts close to 0 as the training set is small and should increase until a plateau which is very often expected to be lower than the accuarcy of the training set. If the training set curve converges toward a much higher value than the test set curve, then the model has a large variance. In other words, to achieve a good bias-variance tradeoff, the accuracy of both curves should converge toward more or less the same value, expected as high as possible.

The learning curves are calculated as the average over 10 runs with random splits and are shown in Figures 6.9 and 6.10. The first remark concerns the variance. On all datasets, the variance is very low and the accuracy on the test set extremely close to accuracy of the training set. Despite a relatively low accuracy, HCBR seems to reach a good bias-variance compromise on `heart` suggesting a more complex model space might help. In general, the remark applies on the four datasets displayed in Figure 6.9 as the observed error rate results from bias and not only from irreducible error. Indeed, the literature comparison provided by Table 6.2 proves that the accuracy could be improved.

For `phishing` and `skin`, we arrive at the same conclusion. However, for those datasets, a heuristic is used during the prediction phase. We discuss its implication on the learning curves in the Appendix C.5. Finally, the learning curve of `adult` in Figure 6.10 shows once again a small variance but high bias.

The analysis of the learning curves indicates that the main limitation of HCBR lies in a large bias. This may come either from the model space complexity or the model selection method that does not properly fit the parameters. The number of parameters in the model is equal to the cardinality of μ, itself proportional to the number of atoms in the training set. However, a closer look at how a decision is taken reveals that each case has its own *small model* as a convex combination of its features. In other words, the real number of parameters to model the decision for a given case never exceed its number of features. This might not be enough to represent fairly complex functions. To discard the second hypothesis, we

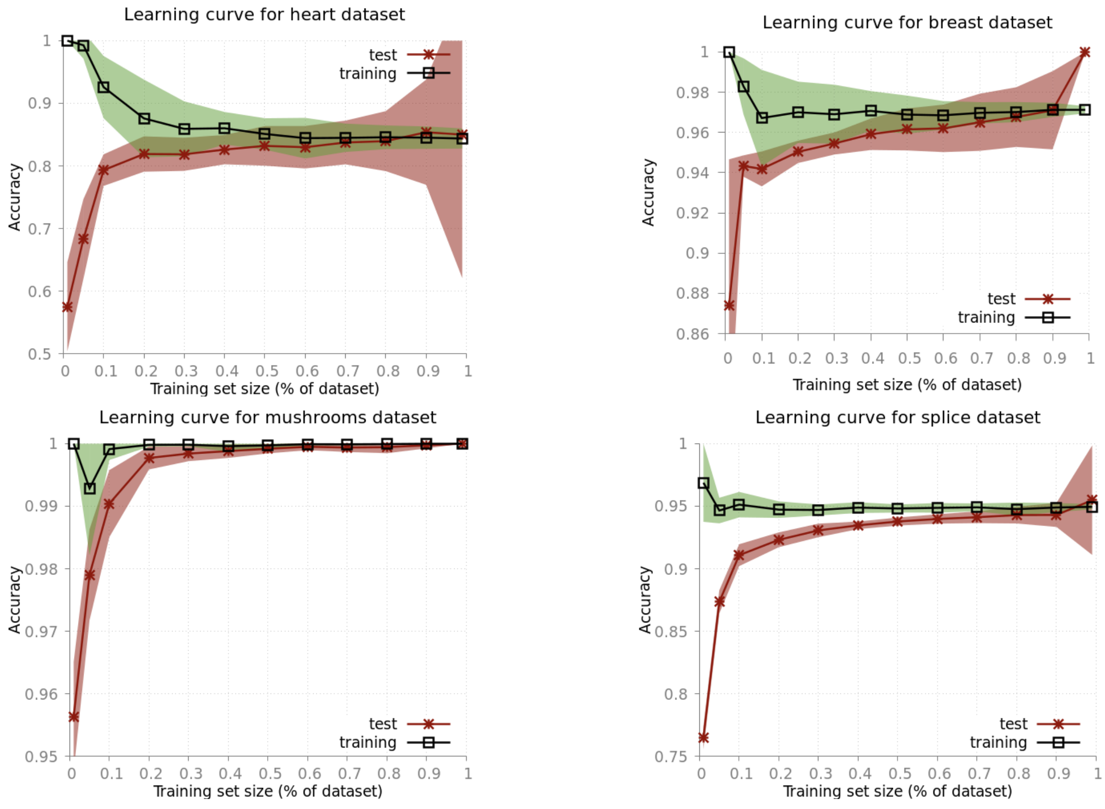


Figure 6.9: Learning curves for `heart`, `breast`, `mushrooms` and `splice` datasets.
Despite a bias and/or irreducible error observed on `heart`, `breast` and `splice`. The model variance appears very low on all datasets. Conversely, the bias and/or irreducible error ranges from very low on `mushrooms`, relatively low on `breast` and `splice`, to high for `heart`.

conducted additional investigations on the model selection in the next section.

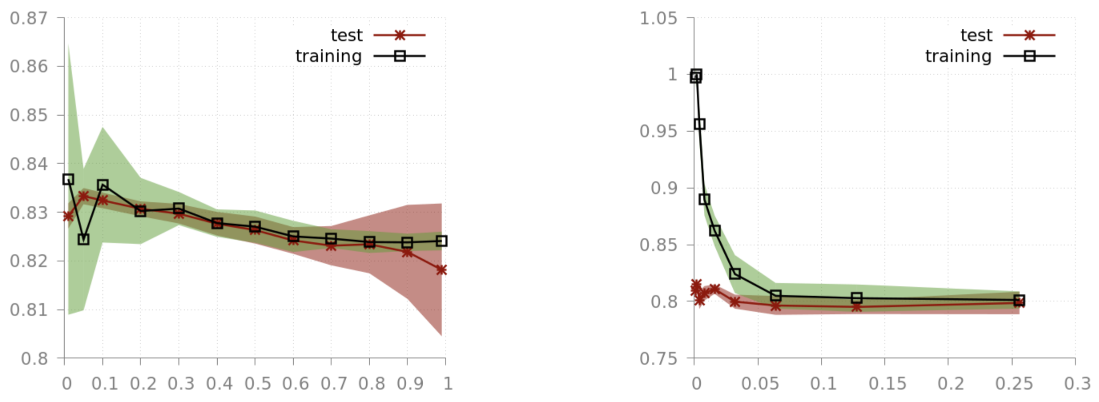


Figure 6.10: Learning curve for `adult`.
On the right, the training set size is restricted from 0 to 30% of the dataset.

6.5.2.2 Assessing model space limitations

We are interested in understanding whether it is possible for HCBR to overfit the training set or at least improve significantly the accuracy or MCC on the training set. We consider the matrix formulation of the support,

$$\mathbf{s} = W\mu \tag{6.14}$$

and we are interested in knowing if there exists a μ s.t. $\mathbf{s}$ leads to classify correctly all the examples. As we already verified that the initial μ provides better results than baseline models, we would like to perturbate as little as possible μ. For this, we calculate $\mathbf{k}$ s.t. $\mathbf{s}+\mathbf{k}$ would correctly classify all the examples. We are looking for δ s.t.,

$$\mathbf{s}+\mathbf{k} = W(\mu+\delta) \tag{6.15}$$

$$\Leftrightarrow \mathbf{s}+\mathbf{k} = W\mu + W\delta \tag{6.16}$$

$$\Leftrightarrow \mathbf{k} = W\delta \tag{6.17}$$

$$\Longrightarrow \delta = W^{+}\mathbf{k} + [I - W^{+}W]\mathbf{w},\ \forall \mathbf{w} \in \mathbb{R}^N \tag{6.18}$$

with W^+ the Moore-Penrose pseudo-inverse of W. When W is not of full rank, the solutions of the undetermined system are given for any vector $\mathbf{w} \in \mathbb{R}^N$, however, it can be shown that $\delta = W^{+}\mathbf{k}$ is a least-square minimizer, i.e. $\forall \mathbf{x} \in \mathbb{R}^M\ ||W\mathbf{x} - \mathbf{k}||_2 \geq ||W\delta - \mathbf{k}||_2$. In particular, if $||W\delta - \mathbf{k}||_2 = 0$, then all the elements would be correctly classified.

Of course, it might not be possible to obtain δ s.t. $||W\delta - \mathbf{k}||_2 = 0$, but it does imply that there exist no couple $(\mathbf{k}, \delta)$ s.t. the accuracy is 1. Solving directly for all such couples seem to be a difficult problem without additional assumptions and thus, we adopted a slightly different approach: instead of fixing $\mathbf{k}$ a priori, we formulated the problem as optimizing the Matthew Correlation Coefficient:

$$\delta^* = \max_{\delta \in \mathbb{R}^M} \mathrm{MCC}(\delta, \mu, W) - c||\delta||_2^2 \tag{6.19}$$

where MCC is the Matthew Correlation Coefficient associated to $\mathbf{s} = W\delta$ and c a regularization factor. The regularization factor translates the idea that a smaller perturbation to obtain the same MCC is better than a larger one. We arbitrarily set c to 0.1. Despite further work could be needed to determine a more tailored value, the conclusions drawn from this section would remain valid.

To solve (6.19), we used a $(\mu + \lambda)$ genetic algorithm with an evolution strategy. Each individual is made of two vectors: $\delta \in \mathbb{R}^M$ and a $\nu \in \mathbb{R}^M$ representing the mutation probability of each corresponding component of δ. The mutation operator is a centered gaussian perturbation and the crossover a 2-points crossover. The implementation is provided by DEAP [Fortin 2012].

We performed 10 runs of a 10-fold cross-validation with random split and compared to the result obtained without the optimization process. The dataset `mushrooms` has been discarded as HCBR reached an accuracy of 1. We set the population to 100, and for each dataset, we adjusted the number of generations and the standard deviation of the gaussian mutation manually (see Appendix C.6 for the details). To set the standard deviation of the mutation, we used $\sigma = \frac{\mu^-}{\alpha}$ where α is an factor determined empirically, and μ^- defined by $\min_{i,j} \mu_i - \mu_j$ i.e. the minimal difference between two components of μ. The rationale behind is that, once again, we would like to slightly perturbate μ and it is reasonable to think that a perturbation should be small enough not to directly switch the estimation of two elements of μ. To conclude, we used a Wilcoxon signed-rank test at 5% and 1% on the test MCC obtained with and without the optimization process. There are three possible scenarios: 1) if the MCC can be improved on the training set and on the test set, it means the problem comes from the model selection method (estimation of μ and training) and results might be improved without changing the model space, 2) if the MCC can be improved on the training set but remains the same or deteriorate on the test set, the model space can represent the training set but overfits, and thus, should be extended 3) if the MCC cannot be improved even on the training set, then the model space is definitely not capable of representing properly the underlying decision mapping and should be extended.

A summary of the results are provided in Table 6.7 and the evolution of the fitness by Figure 6.11. A look at Figure 6.11 confirms that the genetic algorithm converged or was closed to converged and was able to optimize the cost function. In all cases except `skin`, the optimization procedure succeeded to find a vector δ that yields a better MCC on the test sets. However, the improvement is quantitatively different from a dataset to another. For instance, on `heart` the absolute difference in MCC is 16% (relative difference: 23.64%) while on `phishing` the difference is barely 1%. In general, the higher is the initial MCC and the less the improvement is visible.

The variations of MCC on the test set are mitigated. The procedure returns significant changes in only two cases (`adult`) and (`skin`) for which one is improved (`adult`) and one deteriorated (`skin`). It indicates that the model selection and training phase described in Section 6.2.3 and 6.2.4 return one of the best MCC achievable within the model space. More precisely, the vector μ represents one of the best support approximation in its neighborhood. Notice that for `skin`, the optimization process deteriorated the MCC on the training set. A smaller mutation factor might help, however we believe this would barely change the results and thus entail the conclusion on the model space limitation.

Table 6.7: Results obtained on solving (6.19).

Dataset	initial MCC	Δ MCC training	Δ MCC test	WSR 5%	WSR 1%
`adult`	0.5190	0.0400	0.0084	yes	yes
`breasts`	0.9360	0.0312	-0.0025	no	no
`heart`	0.6912	0.1634	-0.0023	no	no
`phishing`	0.8690	0.0093	0.0002	no	no
`skin`	0.8432	-0.0242	-0.0118	yes	yes
`splice`	0.8866	0.0208	0.0006	no	no

6.5.3 Hyperparameters η to control the accuracy

We showed in Section 6.3.3.1 that the hyperparameters can increase the overall performances while the impact is limited by the model space. In this experiment, we show how η can be

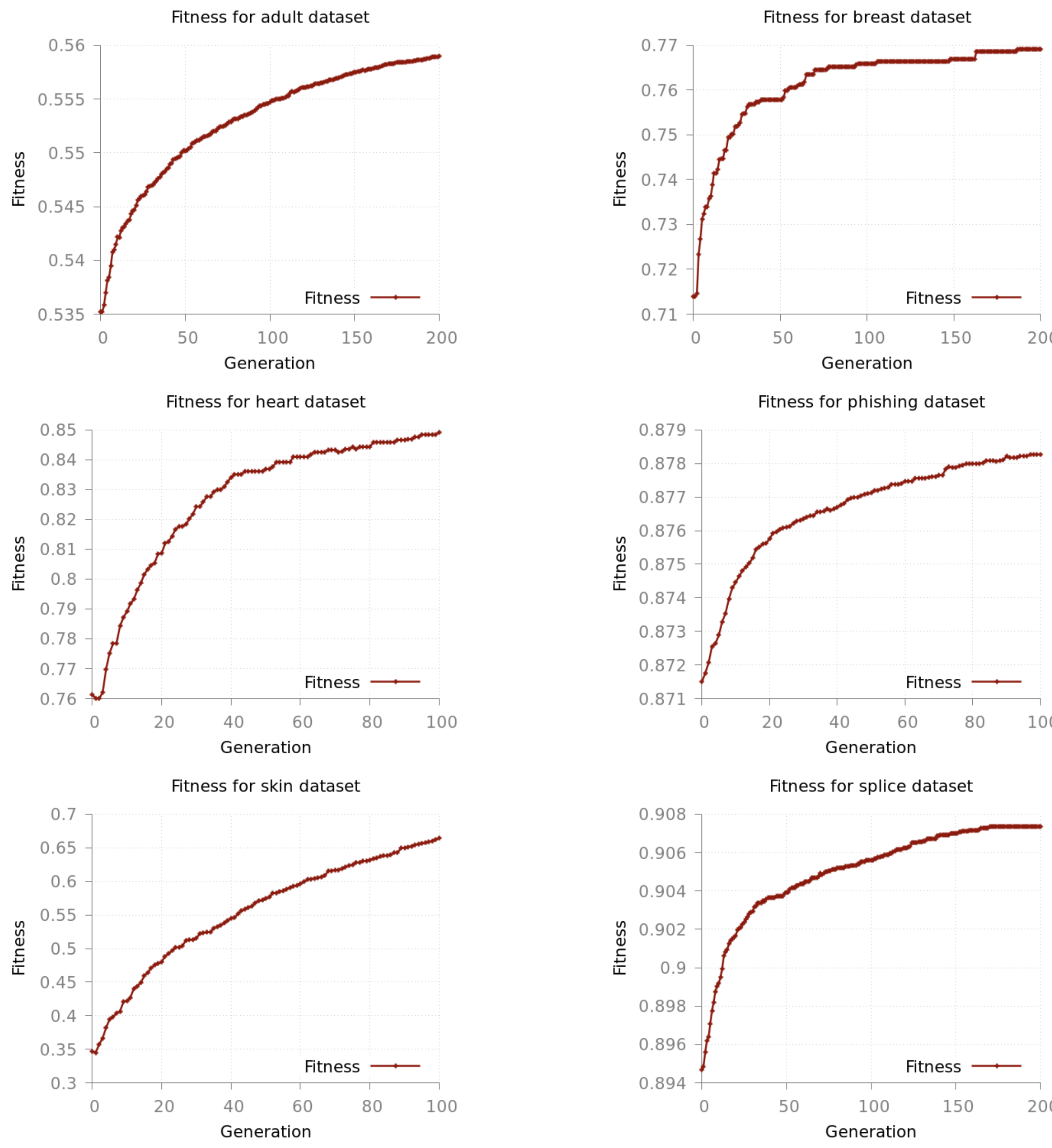


Figure 6.11: Maximal fitness value in the population in function of generations for `adult`, `breast`, `heart`, `phishing`, `skin` and `splice`.

The fitness is lower than the MCC by definition of (6.19). The columns *initial MCC*, Δ *MCC training* and Δ *MCC test* represent respectively the initial MCC on the training set, the difference of MCC obtain on the training with and without the genetic algorithm, the difference of MCC obtain on the test sets with and without the genetic algorithm. *WSR r%* indicates the result of the Wilcoxon signed-rank test at r% risk (yes for significant difference in the sample median, no otherwise).

used to control the accuracy by specifying a threshold on the risk associated to a prediction.

We used a 90-10 split and set $\eta_{-1} = \eta_1$ to ease the visualization. Instead of using the decision function defined by (R2), we did not produce a prediction if the constraints C_1 or C_0

were not respected. It can be viewed as creating a third class *unknown* for which we consider HCBR cannot produce a decision. We measured the accuracy and the test set size ratio for which a prediction has been produced for different values of $\eta := \eta_{-1} = \eta_1$. If the model correctly approximates the underlying mapping function J, increasing η should increase the accuracy while the test set ratio should remain high. Additionally, we plot the test set ratio in function of the accuracy and calculate the Pareto frontier[10], which represents the best compromises accuracy/ratio. The closer the points are to $(1,1)$ the better it is. A Pareto frontier consisting of $(1,1)$ represents the perfect model (e.g. reached on `mushroom` dataset). Figures 6.12, 6.13, 6.14 and 6.15 provide the result for the best and worst two datasets. Figure 6.16 shows all of the four Pareto frontiers. As expected, the results are better on `phishing` and

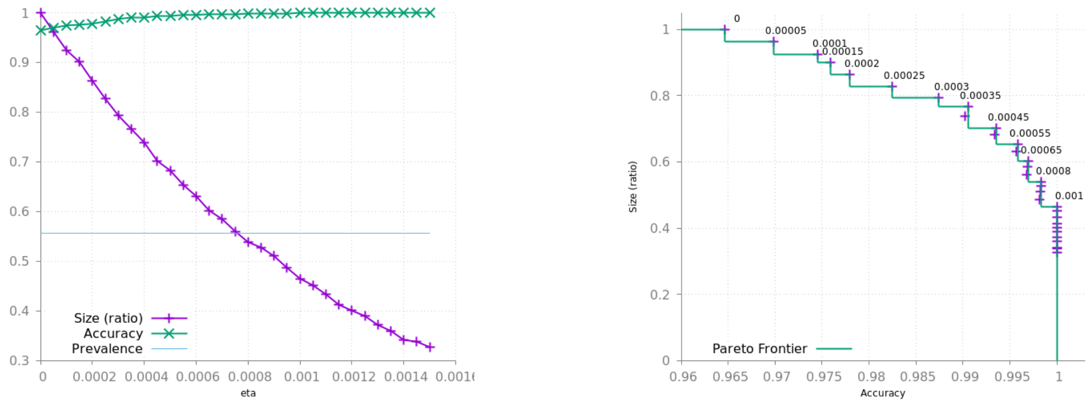


Figure 6.12: Influence of η on `phishing` dataset.

`breast`. On `phishing`, `breast` and `heart`, the accuracy globally increases with η while on `heart` the accuracy slightly decreases indicating poor influence of the hyperparameters and model.

Notice that for certain values of η it is possible to reach 100% accuracy with `heart` (sacrificing over 70% of the dataset) while it is not with `breast`. Also, for high values of η, we observe a fall in accuracy for `breast`. We suspect those two phenomena to appear because we used the same value for η_0 and η_1.

6.6 Discussion

In this section, we discuss several aspects of the proposed algorithm. In particular, we justify the hypergraph representation and propose several improvements to extend the model space and solve the model locality problem.

6.6.1 On the hypergraph representation

The reader may have observed that the model space has a purely set-theoretic interpretation, and no hypergraph-specific property is used so far. Another possibility would be to use a bipartite graph with the first class being the cases of $\mathbf{X}$ and the second the elements of $\mathcal{E}$.

[10]Points s.t. improving one component would deteriorate the other one.

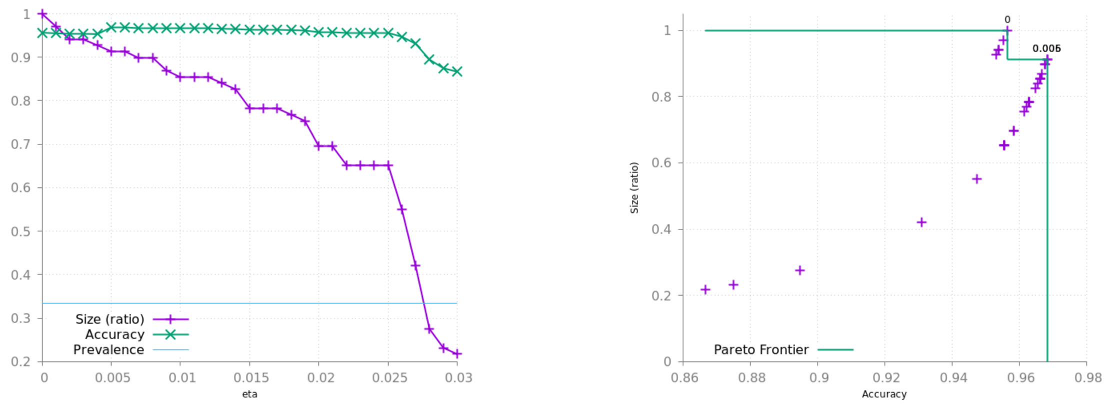


Figure 6.13: Influence of η on `breast` dataset.

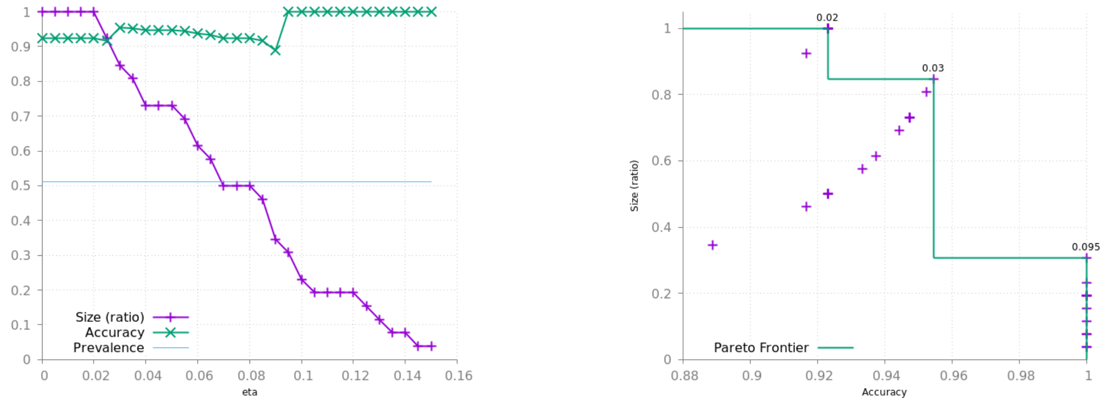


Figure 6.14: Influence of η on `heart` dataset.

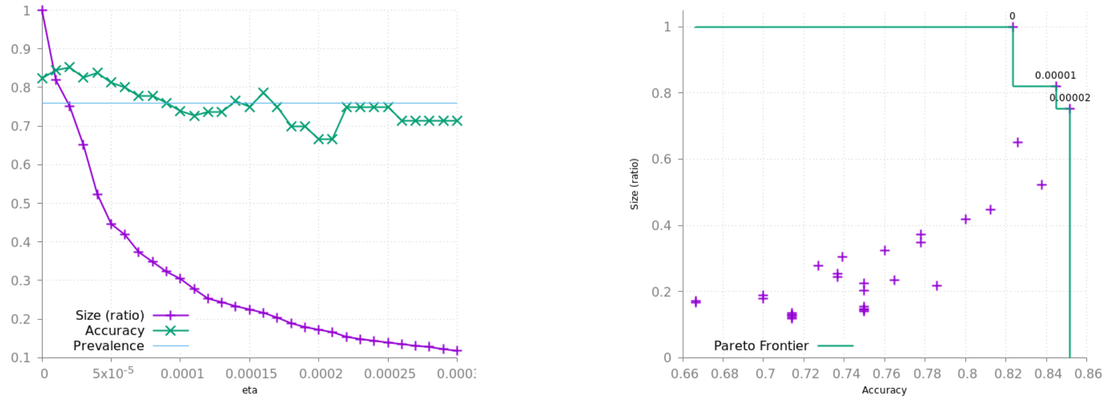


Figure 6.15: Influence of η on `adult` dataset.

We justify viewing the method from a hypergraph perspective by three axes currently being investigated:

- **Model space extension.** The current model space is not complex enough (see Section 6.5.2). An extension using hyperpaths is proposed in Section 6.6.2. Sets or graphs are

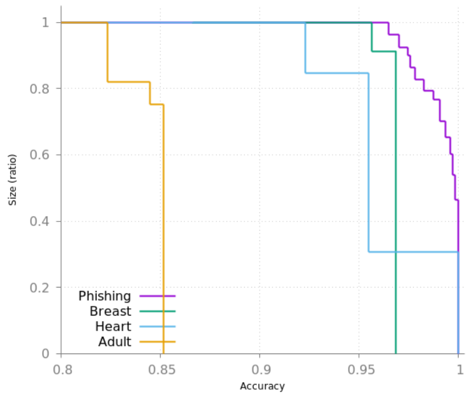


Figure 6.16: Pareto Frontiers comparison.

not suitable to manipulate such space.

- **Justification.** Most justification techniques provide a justification about the model itself (e.g. a decision tree[11]) or provide hints about each decisions under the form of weights for each variable [Ribeiro 2016]. In particular, [Lundberg 2017] unifies the literature and formulates the justification as learning a simple model. With HCBR we are exploring the possibility to generate explanations tailored for an element depending on its neighbors, in a case-based reasoning fashion. For this, we need a notion of neighborhood given by the hyperpaths. This is not possible with a set approach, feasible with graphs but less natural than with hypergraphs.
- **Performances.** Hypergraphs have computational advantages over graphs. Indeed, hypergraphs can be represented as graphs using clique expansion technique. However, increasing hyperdeges cardinality leads to a larger increase in the graph counterpart [Pu 2012].

6.6.2 On the model space extension

It is now clear that the model space is too limited. The number of parameters to describe a case is lower than m, the cardinal of the partition $\mathcal{E}$, and at most $|x|$. The number of parameters to describe the whole datasets is m.

To increase the model space complexity, there are two paths to explore in future work:

1. **Increasing the number of parameters per case.** Currently, the support of $\mathbf{x}$ is modeled by a linear combination over the partition of its features s.t. the interactions between the elements of the partition are not taken into account. A reasonable way to go is to use some combinations of those elements. For instance, if $\mathbf{x}$ is partitioned into e_1, e_2 instead of modeling the support as $s(\mathbf{x}) = w(e_1, \mathbf{x})\mu(e_1) + w(e_2, \mathbf{x})\mu(e_2)$, we would have $s(\mathbf{x}) = w(e_1, \mathbf{x})\mu(e_1) + w(e_2, \mathbf{x})\mu(e_2) + w(e_{1\cup 2}, \mathbf{x})\mu(e_{1\cup 2})$. This would allow to be more discriminative and solve the problem of cases having the same support.

[11] https://github.com/andosa/treeinterpreter

2. **Increasing the number of parameters of the whole model.** A hyperpath of length k is a sequence of hyperedges $(\mathbf{x}_1, ..., \mathbf{x}_k)$ such that $\forall i \in \{1, ...k-1\}$, $\mathbf{x}_i \cap \mathbf{x}_{i+1} \neq \emptyset$. Instead of modeling the support as a combination of the elements $\mathbf{e}_i$ belonging to a specific case $\mathbf{x}_i$, it could be extended to include the elements $\mathbf{e} \in \mathcal{E}$ belonging to the neighbor cases where a neighbor is a case that can be reached by a hyperpath of length k. Thus, the model space proposed here would be the particular case with $k = 0$.

Naturally, it raises many questions, notably how to extend w and μ. The second point seems to be the most interesting because it really requires the hypergraph representation and cannot be interpreted using set theory.

6.6.3 On the model locality

The generalization capacity of HCBR depends on the number of intersections between the examples. Conversely, if a new case does not intersect with the examples, it is impossible to generate a prediction. Therefore, the locality property depends on how much the examples cover the feature space.

Consider that it is possible to choose n examples of k features in a space with K features and $K \gg nk$. There is a tradeoff between covering as much space as possible and having enough intersections to construct a meaningful model. The first extreme configuration consists in maximizing the space cover: there is no intersection between cases, the accuracy during the training phase is 1 but the generalization capacity is null. The probability that a new case will intersect with some examples is maximized. The second extreme configuration maximizes the intersections between the examples: all cases intersect with each other, therefore minimizing the space cover and thus the probability that a new case will intersect with the training set. The accuracy on the training set might not be 1 but the generalization capacity is high compared to the previous case.

For configurations close to the first extreme, it is possible to generate more intersections by using clustering or discretization in the original feature space. For instance, "variable_1=v_1" and "variable_1=v_2" could be encoded the same if both values belong to the same cluster. This requires slightly more feature engineering that previously stated. For the second case, the problem lies in the training set itself. It is not specific to HCBR, but to the fact the training set is not representative of the underlying distribution that generates the cases to classify. In both cases, acquiring more data can help.

In case $\mathbf{x}$ has too many discretionary features, the classification rule is likely to be irrelevant. Indeed, the intersection between $\mathbf{x}$ and $\mathbb{F}_\mathbf{X}$ is to small to hold enough information and make strong analogies with $\mathbf{x}$. To overcome this drawback, $2^\mathbb{F}$ is split into two subsets:

- $\mathcal{F}_1 = \{\mathbf{x} \in 2^\mathbb{F} \mid |\mathbf{x} \cap \mathbb{F}_\mathbf{X}| \geq \delta\}$, $\forall \delta \in \mathbb{N}$
- $\mathcal{F}_2 = 2^\mathbb{F} \setminus \mathcal{F}_1$

$\mathcal{F}_1$ corresponds to the elements s.t. they share some features with the examples. An alternative may be considered by using $\mathcal{F}_1 = \{\mathbf{x} \in 2^\mathbb{F} \mid \frac{D_\mathbf{x}}{|\mathbf{x}|} \leq \delta\}$, $\forall \delta \in [0,1]$. In this case, $\mathcal{F}_1$ contains the elements for which we have enough information provided by the examples. From our preliminary tests, the choice depends on the dataset structure.

Finally, the decision rule for new cases is built as follows:

$$\bar{J}(\mathbf{x}) = \begin{cases} \tilde{J}(\mathbf{x}) & \text{if } \mathbf{x} \in \mathcal{F}_1 \\ o_{\mathbf{x}} & \text{if } \mathbf{x} \in \mathcal{F}_2 \end{cases} \tag{R2}$$

where $o_{\mathbf{x}}$ is one draw from a random variable that has Bernoulli law with parameter $p = \frac{|\{\mathbf{x} \in \mathbf{X} | J(\mathbf{x})=1\}|}{|\mathbf{X}|}$, i.e. the prevalence of class 1 in **X**. It assumes that the prevalence of **X** is close to the prevalence over $2^{\mathbb{F}}$ (or that the prevalence does not change in time for sequential problems in which the new cases are generated by an unknown random measure). The rationale behind is that if for a case **x**, it is not possible to exploit the model built on the hypergraph, then we can still model J as a Bernoulli random variable and use a maximum likelihood estimation for p. In a sense, it is extending the *local* model to the entire input space $2^{\mathbb{F}}$.

6.6.4 Relation with Metric Learning

Similarily to OASIS, HCBR works with input vectors of different dimensions. The iterative form of the optimization program is close to the training phase introduced in Section 6.2.4.

The idea behind HCBR is similar to SLLC except that it learns one seminorm per class, such that we can impose to calibrate the distance (i.e. the distance is a confidence measure) over the training set rather than having an average distance parameter. Also, the seminorms are parametrized by a vector and not by a matrix.

HCBR, as LSMD, learns a non-linear projection but such that the sign indicates the class and the distance to 0, the confidence in this class. Therefore, a small distance between two elements does not indicate the same class but the same level of confidence.

HCBR adopts an opposite direction compared to MMDA: it gives explicitly a projection per $\mathbf{x}_i$ and tries to find a vector μ such that the sum of hinge loss $\ell_{\text{hinge}}(\mathbf{w}_i^T \mu)$ is minimized over the training set.

HCBR is not a pairwise metric learning method. However, many similarities exist with such methods as HCBR needs a proxy to express non-numerical data in a numerical decision space s.t. a simple decision rule performs well. What is learnt is a seminorm per class, representing the distribution of discriminative information to support this class w.r.t. the interactions between the training set examples. The pairwise metric can be artificially defined using the pseudo-norms but is never directly used in the current state of the method.

According to the usual taxonomy of metric learning methods [Bellet 2013, Wang 2015], HCBR can be viewed as a fully-supervised, non-linear and global[12] metric learner algorithm dedicated to classification on non-vector space. The dimensionality reduction is implicitly defined by the hypergraph representation. The literature work on non-numerical data mostly focused on distance between strings [Oncina 2006] and trees [Dalvi 2009, Bernard 2006], with a problem of combinatorial explosion. The scalability study in Section 6.2.7 shows this is not the case with HCBR.

[12]In this thesis, we use the notion of locality in a different meaning than in the metric learning community. The locality of our model expresses the fact the model cannot be used outside of a certain neighborhood, but the metric has been defined using the entire training set.

6.7 Conclusion

This chapter introduced HCBR, a method for binary classification in unstructured space. The method can be seen as learning a metric that optimizes the classification score on the training set. Contrarily to most classification methods or metric learning methods, HCBR does not require to work in vector space and is agnostic to data representation. Therefore, it allows combining information from multiple sources by simply *stacking* the information. It does not require transforming the data to obtain satisfactory results.

The general framework introduced in Section 6.2.2 is instantiated in Section 6.2.3 and 6.2.4 where the support is determined using all the interactions between the hyperedges. Beyond this generic implementation, one can imagine different model selection methods using some assumptions or prior on the data.

To validate HCBR, we performed extensive validation:

1. HCBR has been tested on seven well-known structured datasets and demonstrated similar accuracy when compared to the best results from the literature, with and without hyperparameter tuning. We showed that the model is properly calibrated. Additionally, we performed a comparison with nine alternative methods to find out HCBR, along with Neural Network, outperforms in average with a constant good result. Those experiments showed that HCBR can easily be used and deployed in practice, as it lowers the requirement for feature engineering, data preprocessing and hyperparameter tuning, i.e. the most consuming operations in practical machine learning nowadays.
2. In Section 6.4, we tested HCBR on unstructured datasets and showed it improves the accuracy in most cases compared to reference study.
3. We empirically validated the worst-case complexity. Finally, we studied the properties and limitations of the model space. We showed that the model selection procedure provides one of the best possible performance within the model space. Hence, further work will focus on extending the model space as proposed in Section 6.6.2.

This proof of concept raises many questions and offer many improvement axes. First, it seems relatively easy to extend the method to several classes, with a linear increase of the computation time. As calculating the class support represents most of the computational effort, working on an approximation of the main measure should be investigated. The solution may come from exploring the feature selection capacity of HCBR. It may be possible to remove from the partition some elements that are not discriminative enough, reducing the computation time.

Additionally, we plan to investigate explanation generation about each prediction, using the link between cases in a similar way a lawyer may use past cases to make analogies or counter-examples. We also work on an online version of HCBR where the hypergraph is constructed case after case, including forgetting some old cases (which would allow handling non-stationary environment). It seems possible not only to add new examples dynamically, but also some vertices (i.e. adding some pertinent information to some cases) without generating the whole model from scratch.

Last but not least, we would like to answer some questions: can we provide some quality bounds depending on the initial hypergraph configuration w.r.t. the number of intersections and space cover? How to handle continuous values without discretization?

CHAPTER 7

Continuous Surface Decision using Heat Propagation

Contents

The extended decision space described by Equation (R2) and illustrated by Figure 6.3 is a two dimensional continuous space. However, the decisions are taken in a discriminative fashion, by several linear rules, in the same fashion a SVM would do. In this section, we are interested in finding a generative model in the decision space. To reach this objective, we adopt a different approach by modeling the classification problem using a purely physical point of view and validating the model pertinence a posteriori, by numerical experiments. No direct optimization problem is used, and the model provides an explicit analytic form [Borwein 2013]. We justify the usage of the model by few reasonable assumptions that are, in general, implicit in all machine learning algorithms. We would like to draw the attention on the fact that, by construction, the new classification algorithm proposed in this section does not scale in terms of dimensions, which is tolerable in our two dimensional use-case, but scale in terms of training set size.

The contributions of this section are twofold. First, we introduce in Section 7.2 a new classification algorithm based on the direct resolution of the diffusion equation. In particular, we discuss in Section 7.3 the difference with kernel-based SVM methods, and in particular, with Radial basis function (RBF) SVM [Chang 2010]. Last, we provide numerical experiments to validate the approach in Section 7.4.

The plan of this Chapter is as follows: in Section 7.1, we introduce the heat diffusion problem and provide the key steps to prove the existence of solutions using Lions-Lax-Milgram theorem, and find an explicit form with a spectral decomposition that can be applied to arbitrarily large spaces. In Section 7.2, we present the application of these solutions to the binary classification problem, including the initial hypothesis and an iterative training algorithm to adjust the heat source, such that it improves the decision boundary over iterations. Section 7.4 provides a numerical validation, including a comparison with other standard machine learning methods and a study on the influence of the main model parameters. This Chapter is based on [Quemy 2021], to be published.

7.1 Heat equation

The heat equation is trivially derived from the continuity equation that describes the continuous transport of conserved quantities. In this case, the quantity is heat and the equation refers to the diffusion of heat in a solid. Fourier was the first to understand that the rate of heat through a material is proportional to the negative gradient of temperature [Grattan-Guinness 2005]. He also was the first to provide a solution to this equation for the simple cases.

7.1.1 The problem

The heat equation is defined by

$$\begin{cases} \frac{\partial u}{\partial t} - \Delta u = f(x,y,t) \in L^2(\Omega), \Omega =]0,1[^2 \times]0,T[\\ u(x,0,t) = 0 \\ u(x,1,t) = 0 \\ \frac{\partial u}{\partial n}(0,y,t) = 0 \\ \frac{\partial u}{\partial n}(1,y,t) = 0 \\ u(x,y,0) = f(x,y,0) \end{cases} \tag{HE}$$

This standard equation models heat diffusion in a two dimensional surface of an homogeneous material, heated according to a source f of finite energy. Dirichlet and Neumann conditions specify respectively the behavior of heat on the vertical edges and the heat direction on the horizontal edges. Finally, the initial condition provides the initial heat distribution over the spatial space. We assume the problem to be homogeneous, i.e. the boundary conditions are equalized to zero.

The general technique to solve the problem would work for a N dimensional hypercube and is easy to adapt to any regular domain, such as a sphere or torus.

7.1.2 Solving the problem

In this section, we provide the main steps to solve the problem (HE). For a formal demonstration of Lions-Lax-Milgram theorem and further reading on functional analysis, we refer the reader to [Brezis 2010, Allaire 2007].

First, we considering only the spectral part of the problem:

$$-\Delta w = \lambda w \tag{SP}$$

Thanks to Lions-Lax-Milgram theorem [Lax 1955], we know that the solution exists almost everywhere in $L^2(\Omega)$. Using a variational approach coupled to the spectral decomposition of self-adjoint compact operators, we can find explicit solutions.

Theorem (Spectral theorem): *Let us assume V and H two real Hilbert spaces of infinite dimensions. We suppose $V \subset H$ with compact injection (i.e. the inclusion operator is continuous and compact), and that V is dense in H. Let $a(.,.)$ be a bilinear symmetric continuous form satisfying V-ellipticity. Then, the eigenvalues of a s.t. $\forall v \in V, a(u,v) = \lambda < u,v >_H$ form an increasing sequence $(\lambda_k)_{k\geq 1}$ of positive reals that goes to infinity, and there exists a Hilbert base of H, $(u_k)_{k\geq 1}$ of eigenvectors:*

$$u_k \in V,\ et\ a(u_k,v) = \lambda_k < u_k, v_k >_H, \quad \forall v \in V \tag{7.1}$$

In addition, $(\frac{u_k}{\sqrt{\lambda_k}})_{k\geq 1}$ is an Hilbert base of V embeded with the scalar product $a(.,.)$.

Let $H^1_0(\Omega)$ be the closure of $C^1_c(\Omega)$ in the Sobolev space $H^1(\Omega)$, and $\mathcal{D}(\Omega)$ the space of infinitely differentiable functions with compact support, also known as test functions. To take into account the Dirichlet conditions, we define $V = H^1_0(\Omega)$, and H as the Lebesgue space $L^2(\Omega)$. Using a variational approach on the spectral problem (SP), we deduce that $a(u,v) = \int_\Omega \nabla u.\nabla v dx = \lambda \int_\Omega uv dx = \lambda < u,v >_{L^2(\Omega)}$. According to Rellich theorem $H^1_0(\Omega)$ is compactly included in $L^2(\Omega)$. As $\mathcal{D}(\Omega)$ is dense in both spaces, we can deduce that $H^1_0(\Omega)$ is dense in $L^2(\Omega)$. Therefore, we can apply the spectral theorem above which demonstrates the existence of solutions to (SP) in $H^1_0(\Omega)$.

To come back to the original problem, the Green identity and the trace operator Sobolev spaces are sufficient to conclude [Brezis 1999]. A simpler alternative is to use the space of distributions:

$$\begin{aligned} &\int_\Omega \nabla u.\nabla v dx = \lambda \int_\Omega uv dx \\ &\Leftrightarrow < \nabla u, \nabla v > = \lambda < u,v > \\ &\Leftrightarrow - < \Delta u, v > = \lambda < u,v > \end{aligned} \tag{7.2}$$

thus $-\Delta u = \lambda u$ in the sense of distribution. Thanks to the density of $\mathcal{D}$ in V and H, and using the compact canonic injection, we deduce the equivalence in L^2.

With the existence of solution proven, we look for explicit solutions. A remarkable properties of the heat equation is that the solution is separable. In this work, we consider only the problem with a two dimensional spatial space, but the technique generalizes to any dimension. In other words, we assume that the solution is of the following form:

$$w(x,t) = X(x)Y(y) \tag{7.3}$$

Using the initial conditions, the solution can be expressed as follows:

$$\begin{cases} w_{k,l}(x,y) = Csin(l\pi y)cos(k\pi x) \\ \lambda_k = \pi^2(l^2 + k^2) \end{cases} , \forall k \geq 0 \tag{7.4}$$

The normalization constant is easily obtained by the following:

$$||w_{l,k}||_{L^2(\Omega)} = 1 \Leftrightarrow \begin{cases} C = \sqrt{2} & \text{if } k = 0 \\ C = 2 & \text{otherwise} \end{cases} \tag{7.5}$$

The solution of (SP) can be writen as follows:

$$w(x, y) = \sum_{l,k} \xi_{l,k}(t)\, w_{l,k}(x, y) \tag{7.6}$$

As f is assumed to be in $L^2(\Omega)$, it can also be expressed in the Hilbert base by $f(x, y) = \sum_{l,k} f_{l,k}(t)\, w_{l,k}(x, y)$ with $f_{l,k}(t) =< f(x, y, t), w_{l,k}(x, y) >_{L^2(\Omega)}$. We can finally rewrite the problem in the Hilbert base:

$$\begin{aligned} \frac{\partial}{\partial t} \sum_{l,k} \xi_{l,k}(t)\, w_{l,k}(x, y) - \Delta \sum_{l,k} \xi_{l,k}(t)\, w_{l,k}(x, y) \\ = \sum_{l,k} f_{l,k}(t)\, w_{l,k}(x, y) \end{aligned} \tag{7.7}$$

By linearity,

$$\sum_{l,k} [\xi'_{l,k}(t) - \lambda_{l,k} \xi_{l,k}(t)]\, w_{l,k}(x, y) = \sum_{l,k} f_{l,k}(t)\, w_{l,k}(x, y) \tag{7.8}$$

which is equivalent to solve the following Ordinary Differential Equation (ODE):

$$\xi'_{l,k}(t) - \lambda_{l,k} \xi_{l,k}(t) = f_{l,k}(t), \quad \forall k, l \geq 0 \tag{7.9}$$

7.2 Classification using explicit heat equation

In this work, we simplify the problem by removing the time dimension. This removes the need to compute the solutions of the ODEs to obtain $\xi_{l,k}$:

$$\xi_{l,k} = \frac{f_{l,k}}{\lambda_{l,k}}, \quad \forall k, l > 0 \tag{7.10}$$

The hypothesis behind using the heat equation for binary classification is that for a point of a given class, the confidence level for this class around this point evolves continuously as described by the heat equation. The continuity hypothesis is common to most machine learning techniques.

Consider Ω as the unit hypercube of dimension N. With the continuity hypothesis in mind, we define the source f as a certain prior with the following constraint: $\forall \mathbf{x} \in X,\ f(\mathbf{x}) \in [-1, 1]$. Given a training set $(\mathbf{X}, \mathbf{y})$ with $\forall y \in Y, y \in \{0, 1\}$, we define $\forall \mathbf{x}, y \in \mathbf{X} \times Y, f(\mathbf{x}, t) = y$ and $\forall \mathbf{x} \in \Omega \setminus \mathbf{X}, f(\mathbf{x}, t) = 0$.

To compute the solution, we use a finite difference method with an homogeneous grid in every dimension with a step h. We denote by Ω_h the discretized space. The projection on the Hilbert base is done using Simpson numerical integration [Süli 2003] using the M first eigenvectors. Once the solution $u \in L(\Omega_h)$ is obtained and normalized within $[-1, 1]$, the decision boundary is defined by a level set defined by $\beta \in [-1, 1]$. In practice, β is set to 0 and can be automatically adjusted as hyperparameter using, e.g. a grid search or more advanced hyperparameter tuning techniques.

$$\forall \mathbf{x} \in \Omega,\ \bar{y} = \begin{cases} 1 & \text{if } u(\mathbf{x}) \geq \beta \\ -1 & \text{otherwise} \end{cases} \tag{7.11}$$

The solution is then evaluated on the training set. For each misclassified input vector, its corresponding heat value in the source is updated depending on its margin, and a regularization factor, as described by Algorithm 3. The problem is then solved again for u with the updated source. This procedure is performed K times.

Algorithm 3 Model generation for hypercubes.

Input:
- $(\mathbf{X}, y)$: training set
- N: hypercube dimensions
- M: number of eigenvalues
- h: grid step size
- $K \geq 1$: number of source adjustment iterations
- $\alpha \geq 1$: regularization factor
- $\beta \in [0,1]$: level set to use as decision boundary

Output:
- $u_H \in L^2(\Omega_H)$

```
1: $(\lambda_i, w_i)_i \leftarrow$ precompute_eigenvalues()
2: $f^{(0)} \leftarrow$ generates_source(X, y)
3: for $k \in \{1..K\}$ do
4:     $(f_i^{(k)})_i \leftarrow \int_{\Omega_H} f^k(\mathbf{x}) w_i d\mathbf{x}$
5:     $(\xi_i^{(k)})_i \leftarrow \frac{f_i^{(k)}}{\lambda_i}$
6:     for $(\mathbf{x}, y) \in \mathbf{X} \times Y$ do
7:         $u(\mathbf{x}) \leftarrow \sum_{i=0}^{\frac{N}{H}} (\xi_i^{(k)})^T w_i$
8:     end for
9:     for $(\mathbf{x}, y) \in \mathbf{X} \times Y$ do
10:        $\bar{y} = 1$ if $u(\mathbf{x}) \geq \beta$ else $-1$
11:        if $\bar{y} \neq y$ then
12:            $f^{k+1} \leftarrow f^k + y\alpha(1 - |u(\mathbf{x}) - \beta|)$
13:        end if
14:    end for
15: end for
```

7.3 Discussion on the Relation with Gaussian Kernel

In this section, we clarify the similarities and differences between non-linear SVM using gaussian kernel and the approach presented in this chapter, namely solving the heat equation, considering the heat as class probability for a given point. cd The heat kernel of $\mathbb{R}^N$ is defined as follows:

$$K_H(t,x,y) = \frac{1}{(4\pi t)^{\frac{N}{2}}} \exp(-\frac{|x-y|^2}{4t}) \tag{7.12}$$

This time-varying function is solution of $\frac{\partial K}{\partial t}(t,x,y) = \Delta K(t,xy)$ for all $t > 0;\ x,y \in \mathbb{R}^N$ under the following condition:

$$\lim_{t\to 0} \int_{\mathbb{R}^N} K(t,x,y)\phi(y) = \phi(y) \tag{7.13}$$

for any test function $\phi \in \mathcal{D}(\mathbb{R}^N)$. This is the limit in the sens of distribution s.t. the the value of the heat kernel next to zero tends toward the dirac distribution $\delta(x-y)$. Therefore, K is the Green's function of the heat equation [Berline 2003].

In comparison, the gaussian kernel does not depend on time and is defined by:

$$K_G(x,y) = \exp(\frac{||x-y||^2}{2\sigma^2}) \tag{7.14}$$

To allow the kernel trick in RBF SVM, Mercer's therem [Mercer 1909] is used. In particular, it maps the kernel to a linear operator of the following form:

$$T_K\phi(x) \int_a^b K_G(x,y)\phi(y)dy \tag{7.15}$$

for any function $\phi \in L^2([a,b])$. This is this operator on which a spectral decomposition is applied and justified by Mercer's theorem [Ferreira 2009].

In the physical approach of Section 7.1, the model describing heat diffusion is defined, then Lions-Lax-Milgram theorem used to prove the existence and uniqueness of solution under restrictive conditions. To find the explicit the solution and being able to numerically approach it, the spectral theorem is applied because the Laplacian is a compact operator. We are never interested in finding a solution in another space. The space V in the variational approach is an auxiliary space to solve the problem. Density relations and canonic injection allows then to make sure the solution makes sense in the original space.

In the machine learning approach, the shape of the solution is determined in advance and an optimization problem is defined. The solution is assumed to live in a finite dimensional space. The existence of a feature map ϕ from the original space to a higher dimensional space is assumed to exist such that the solution in the higher dimensional space is better than in the original space. The kernel is then selected to fulfill Mercer's theorem that gives a spectral decomposition of the kernel seen as a compact self-adjoint integral operator. This decomposition provides a proper inner product on the the higher dimensional space V such that the explicit representation of ϕ is not necessary. In other words, the space V plays a preponderant role in the resolution, and the spectral decomposition is used only to allow the *kernel trick*.

We mentioned that we do not need to assume or solve any optmization problem contrarily to the machine learning approach. In fact, there is an underlying optimization program that is solved by a direct corrolary of Lions-Milgram-Lax theorem. Indeed, the theorem states that if $a(.,.)$ is symmetric, as is the Laplacian, then $\exists! y \in H,\ J(u) = \min_{v \in H} \frac{1}{2} a(v,v) - L(v)$. As $a(.,.)$ is a bilinear symmetric form, and assuming any basis, there exists a matrice A s.t. it is equivalent to $\exists! y \in H,\ J(u) = \min_{v \in H} \frac{1}{2} v^T A v - L v$ [Mohri 2018].

Kernel-based methods solve a similar optimization program. Using the same notations, the program is defined by $\min_{v \in H} \frac{1}{2} v^T Q v - e^T v$ with $e = [1, ..., 1]^T$ and $Q_{ij} = y_i y_j K(x_i, x_j)$. Although its nature is the same, the dimension of A depends only on the number of eigenvectors kept in the basis while the dimension of Q depends directly on the training set size. This formulation illustrates one major difference between the two kernels: the heat kernel depends on the dimension N which is the computational bottleneck of the physical approach. On the contrary, kernel-based SVM computational time is primarily influenced by the training size set.

7.4 Numerical experiments

In this section, we study the property of the parameters of Algorithm 3 and compare it to some standard classification algorithms: linear SVM, RBF SVM, Random Forest, Extra Tree, Gaussian Process for classification, Multilayer Perceptron, Gaussian Naive Bayes, and Gradient Boosting. We use a 10-folds cross-validation with a stratified shuffle sampling method. The value of β is selected by an exaustive search for values in {-0.2, -0.1, 0, 0.1, 0.2 } on a validation set.

We use three dataset generators with various difficulties. The datasets used to study the model parameters are shown in Figure 7.1. The first dataset generates two clusters per class whose barycenters are located on a randomly generated polytope. The second dataset is made out of two interleaving half circles with 20% noise (points that are flipped to belong to the opposite class). The third one is made out of two imbricated circular clusters, with 20% noise. Those datasets are non-linearly separable and present some difficulties that a modern machine learning algorithm must be able to tackle.

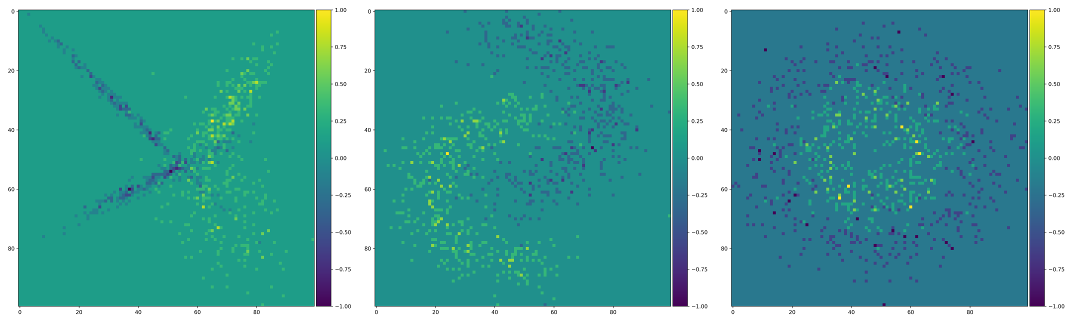


Figure 7.1: Synthetic datasets made out of 1000 points with two clusters per class on a 100x100 grid.

In the following sections, the solutions are displayed with their isolines from -1 to 0 by 0.1 step. The bold line represents the boundary for level 0 and the dashed line, the best boundary found by the algorithm. In case the best boundary is 0, only the dashed line is displayed. The points displayed on the solution are the test set.

7.4.1 Influence of model parameters

Eigenvalues number: Figures 7.3 and 7.2 show the impact of the number of eigenvectors kept in the decomposition, for dataset 1 and 2. Low values allows only very regular and symmetric solutions in terms of level sets. Increasing the value allows the level sets and thus the decision boundary to be less regular as it increases the numerical approximation of the exact solution to the heat problem. In particular, in Figure 7.3, for five eigenvectors, it is impossible to distinguish between the two blue clusters, while for 20 and 50 there exists an isoline that clearly allows to dinstinguish between them. Above 20 eigenvalues, the gain in precision does not seem to increase the quality of the solution. The number of eigenvalues could even be lower depending on the datasets. For instance, on dataset 3, because of the axial symmetry through the center by any axis, less eigenvectors are requires to properly describe the solution. Conversely, more complex datasets, with many clusters, various cluster shapes may need even more eigenvectors to describe the solution.

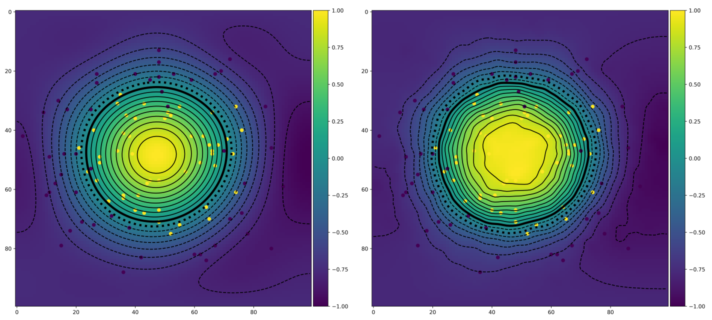


Figure 7.2: Solution for 5 and 50 eigenvalues on dataset 2.

Iterations number: Figures 7.4, 7.5 and 7.6 display the evolution of the solution depending on the number of iterations using 20 eigenvalues and α set to 1. For dataset 1 in Figure 7.4, the boundary in the area where the clusters intersect is pushed by decreasing the temperature of misclassified points of class -1. Two small isolated components appear after seven iterations in the *hot* area. After 10 iterations, those components are merged with the main cold area. However, as it leads to massively misclassifying hot points, the boundary is almost brought back to the initial one after 13 iterations. Notice however, that the isolines clearly display the shape of the clusters. Finally, after 18 iterations, the solution clearly displays the four clusters, including the parts accross both classes. This oscillating behavior comes from the noise in the center: modifying the heat of some points leads some other to be misclassified, themselves being misclassified after the next iteration, and so forth. The value α is probably too high and in the future, we could use a decaying schema such that the total variation of heat on the source between iterations decreases iteration after iteration (similarily to the simulated annealing algorithm). Another possibility would be to solve the non-homogeneous problem such that the diffusion coefficient is not constant: increased if a point is correctly classified, decreased otherwise such that even if the heat is drastically incrased and the point correctly classified at the next iteration, this heat will not be propagated to far around.

In Figure 7.5, after three iterations the solutions is still very close to the initial one. Con-

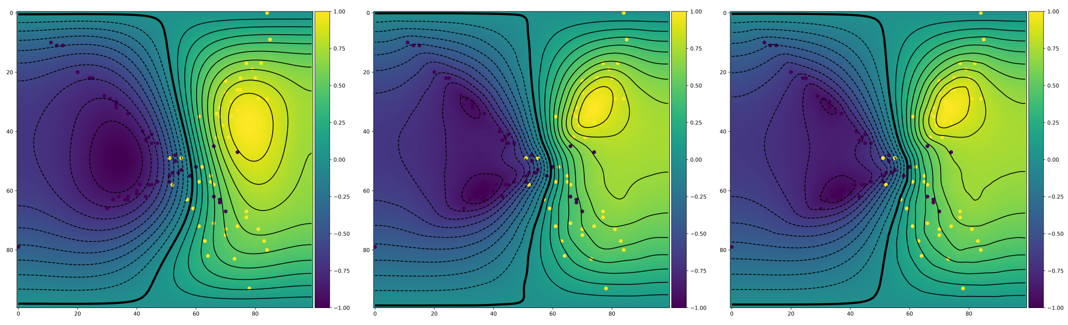


Figure 7.3: Solution for 5, 20 and 50 eigenvalues on dataset 1.

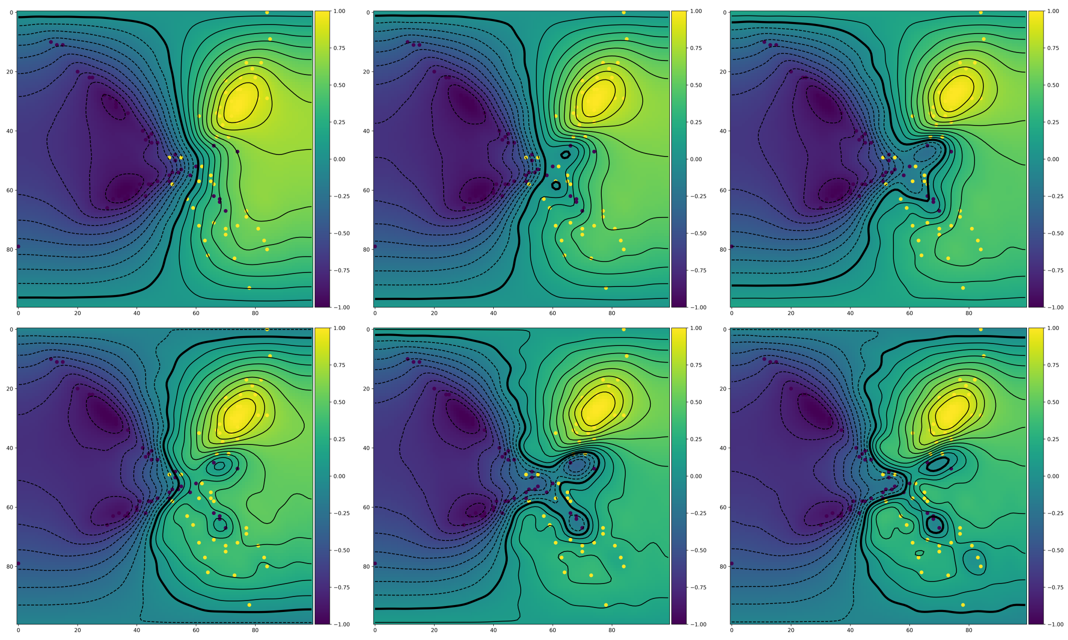


Figure 7.4: Solution for 3, 7, 10, 13, 18 and 25 iterations with 20 eigenvectors and $\alpha = 1$ on dataset 1.

trarily to the first dataset, there exists a rather smooth boundary to separate the clusters and thus, there is no oscillation. Iteration after iteration, the isolines are torn to fit the clusters. The process is rather slow, most likely due to a low value for α.

Dataset 3 is probably the easiest one for the method due to the Laplacian properties. In Figure 7.6, we can observe that the final solution conserves its circular shape while adjusting the local specifities of the training set to the point it clearly overfits. For instance, after 18 iterations, there is an irrelevant isolated component above the main isoline.

α **value:** Figures 7.7 and 7.8 display the solution depending on α after 10 iterations with 20 eignenvectors for dataset 1 and 2 (dataset is omitted for space reason). As expected, the solutions highly differ depending on the value of α. In particular, despite the solution provides in general good accuracy, the solution feels less fitting the training set topology.

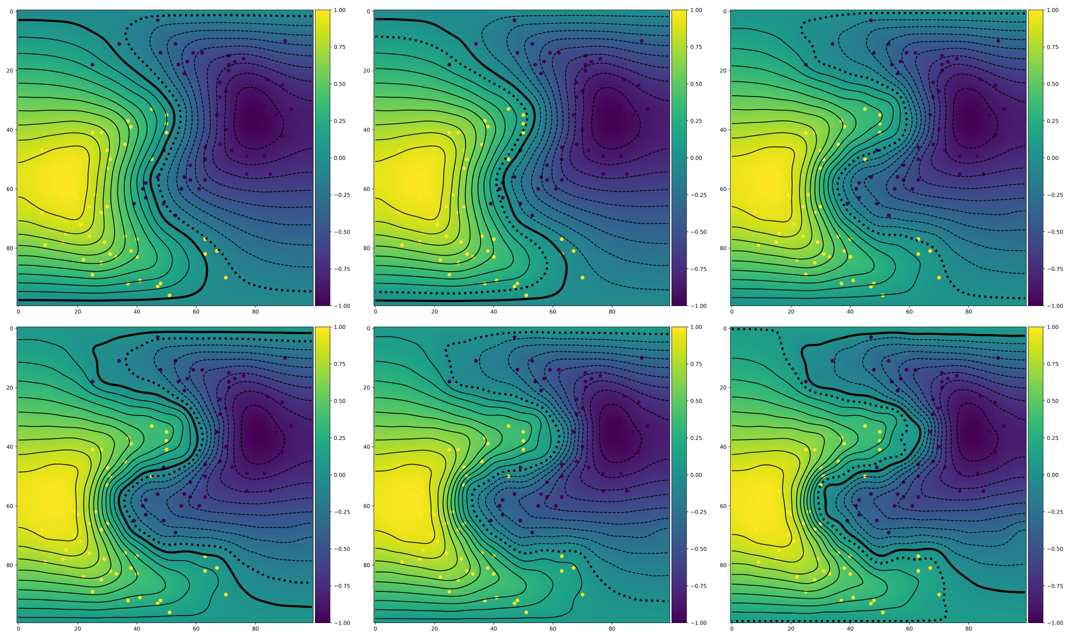


Figure 7.5: Solution for 3, 7, 10, 13, 18 and 25 iterations with 20 eigenvectors and $\alpha = 1$ on dataset 2.

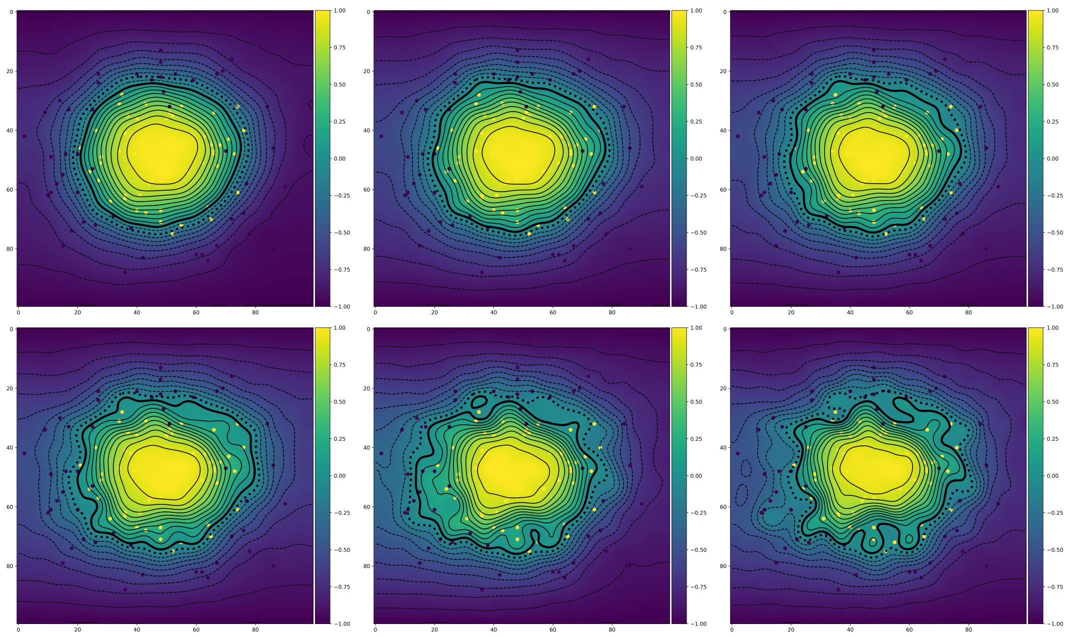


Figure 7.6: Solution for 3, 7, 10, 13, 18 and 25 iterations with 20 eigenvectors and $\alpha = 1$ on dataset 3.

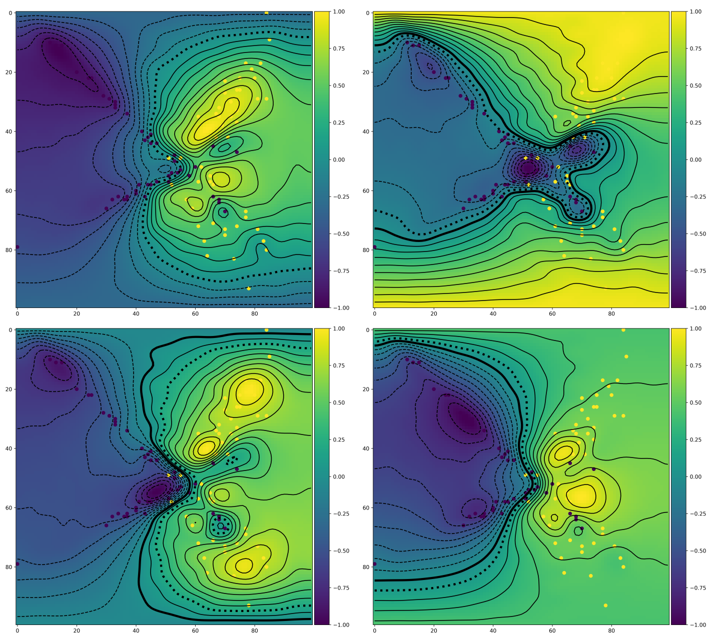


Figure 7.7: Solution after 10 iterations for values of α in $\{3,5,10,25\}$ with 20 eigenvectors on dataset 1.

On dataset 2, Figure 7.8, a higher value for α than 1 speeds up the training process, as displayed for $\alpha = 3$ and $\alpha = 5$. However, above a certain threshold, the solution will oscillate with the training iterations without guarantee that the solution will provide good results. Actually, α controls the radius of the neighborhood around a misclassified point that will be affected by the heat modification of this specific point. A too high value implies that a lot of neighbors will be misclassified, themselves misclassifying their neighbors in a sort of chain reaction. Empirically, $\alpha - 3$ or $\alpha = 5$ seem to provide the best results. Notice that α is different from the parameter σ of the Gaussian kernel. A given point might see its temperature changed several time in the process, with possible different values of α in case of adaptive schema. These temperature changes implies a local change in the radius of influence. On the contrary, the latter fixes a common radius of influence for all points on the domain. The value is not update during the optimization process.

7.4.2 Comparison to other methods

We compared our method to eight other methods provided by SCIKIT-LEARN [Pedregosa 2011]: linear SVM, RBF SVM, Random Forest, Extra Tree, Gaussian Process for classification, Multilayer Perceptron, Gaussian Naive Bayes, and Gradient Boosting. We used three dataset generators and a 10-fold cross-validation as mentioned above. However, to obtain really statistically robust results, we generated five datasets per generator, and then averaged the cross-validation scores to obtain the final result. As the datasets and the split for the cross-validation are balanced, the accuracy is a proper indicator of model performance. We also report the average standard deviation.

We used the default parameters for the algorithm implementation. For our method, we

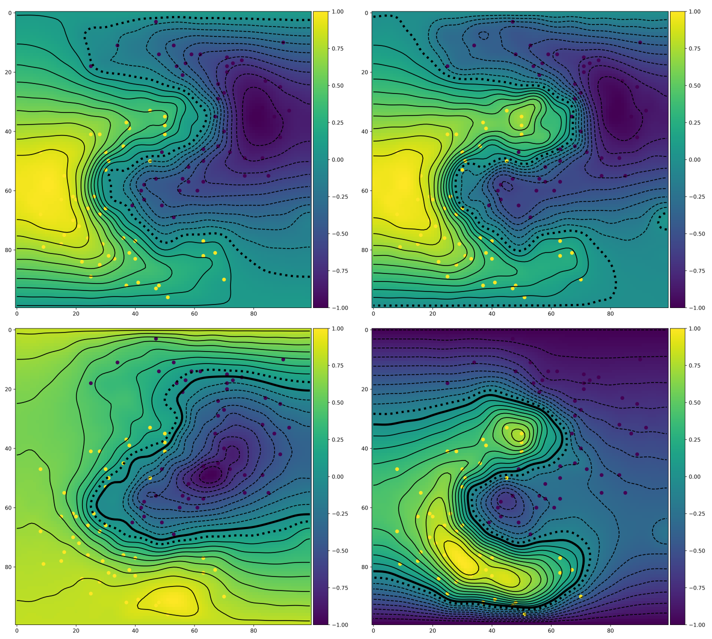


Figure 7.8: Solution after 10 iterations for values of α in $\{3, 5, 10, 25\}$ with 20 eigenvectors on dataset 2.

used 20 eigenvalues, 10 iterations and $\alpha = 1$ which we consider being the default configuration.

The results are provided in Table 8.1. As expected, Linear SVM provides the worst results as none of the dataset are separable. In general, the Laplacian approach presented here performs well on all datasets. It ranks first on dataset 3 which is not surprising, knowing the properties of the Laplacian operator. On dataset 2, it ranks second, only 0.20% behind Gradient Boosting. The worst results are achieved on the first dataset, where it ranks 4th behind Gradient Boosting (-1.46%), Gaussian Process (-1.32%) and slightly after Extra Tree (-0.12%). In addition, the standard deviation belongs to the same range as the other good performing methods.

Those results are encouraging since the model performed well against some state-of-the-art methods s.a. Gradient Boosting or Extra Tree. Even more interesting, it provides better results than Random Forest or RBF SVM. Although the selected datasets propose some common difficulties in classification, additional validation is still needed. In particular, while the size of the training set has little impact on the execution time, the number of dimensions is the main bottleneck as computing the solution requires computing as many integral as there are dimensions.

7.5 Conclusion

We can summarized the contributions of this chapter as follows:

1. We assumed that the confidence in a label is continous accross the space and behave as a diffusion model. A training dataset can thus be viewed as an initial distribution

Table 7.1: Average accuracy obtained over five datasets for each dataset generator.

	Dataset gen. 1	Dataset gen. 2	Dataset gen. 3
Laplacian	0.8472 (0.035797)	0.9628 (0.020220)	**0.9020** (0.024278)
Linear SVM	0.3982 (0.433634)	0.4338 (0.480578)	0.2577 (0.449427)
RBF SVM	0.8338 (0.034809)	0.9292 (0.026755)	0.8526 (0.034984)
Random Forest	0.8140 (0.034044)	0.9074 (0.025235)	0.8648 (0.040679)
Extra Tree	0.8484 (0.030553)	0.9638 (0.019310)	0.8720 (0.030403)
Gaussian Process	0.8604 (0.030100)	0.9004 (0.016497)	0.9001 (0.032232)
Multilayer Perceptron	0.7730 (0.038215)	0.8794 (0.032538)	0.7428 (0.034757)
Gaussian Naive Bayes	0.8116 (0.034721)	0.8964 (0.032232)	0.8978 (0.030916)
Gradient Boosting	**0.8618** (0.026415)	**0.9648** (0.019082)	0.8920 (0.029064)

of heat such that solving the heat equation propagates the belief or information in the whole space.

2. We showed the existence and unicity of the solution, as well as determined the time complexity of the algorithm.
3. Numerical validation over three different types of non-separable datasets has shown the pertinence of such approach.
4. The diffusion model managed to be as accurate as the state-of-the-art machine learning techniques, such as Gradient Boosting or Extra Tree.
5. We showed the connections and differences between the proposed approach and kernel-based methods.

One advantage of adopting this approach is the well understood mathematics behind, in particular the proven existence and uniqueness of solution. Notice also that the method is purely deterministic. We do believe that leveraging robust mathematical models rather than traditional "learned" black-box models is a key to provide more transparent and explanable solutions to problems.

Beyond this proof of concept, several axis of improvement appear. First, we mentioned several ideas to select the correct parameter to speed-up the algorithm while preventing overfitting and solution oscilations between iterations: adaptive parameters and heat equation with non-homogeneous diffusion coefficient.

More theoretically, we do not provide a proof of convergence nor under which condition the solution converges to a perfectly classified training set. The intuition tells us the algorithm converges towards a cycle of distributions that depends on α. But it remains to be formally proven.

In this preliminary work, we studied only two dimensional problems with a hypercube as space domain, although the theory adapts immediatly to any dimension and pretty easily to other type of regular domains. Future work should study the scalability of the method in high dimensions. It is clear that for a given hypercube, the training set size does not affect the computation time, however, more dimensions implies more costly numerical integration. We hope to improve the scalability by using the different symmetries available.

However, as mentioned in the introduction, our priliminary use-case was to replace our discriminative and linear decision rules by a generative model. As this problem lies in a two dimensional space, there is no practical limitations. More interesting, replacing the initial rule by this model can be done without taking into account the size of the training set since it does not affect the computation time.

CHAPTER 8

Two-Stage Optimization for Machine Learning Workflow

Contents

Machine learning techniques play a preponderant role in dealing with massive amount of data and are employed in almost every possible domain. Building a high quality machine learning model to be deployed in production is a challenging task, from both, the subject matter experts and the machine learning practitioners.

For a broader adoption and scalability of machine learning systems, the construction and configuration of machine learning workflow need to gain in automation. In the last few years, several techniques have been developed in this direction, known as AUTOML.

In practical machine learning, data are as important as algorithms. Algorithms received a lot of interest in hyperparameter tuning methods [Elshawi 2019, Hutter 2019], that is to say, the art of adjusting parameters that are not dependent on the instance data. Contrarily, data pipeline construction and configuration received little if any interest. For instance, [Crone 2006] notices that algorithm hyperparameter tuning is performed in 16 out of 19 selected publications while only 2 publications study the impact of data preprocessing. More recent work suggest this observation is still valid [Couronné 2018]. This can probably be explained by the fact that the research community mainly uses ready-to-consume datasets, hence occulting de facto this problematic. In practice however, raw data are rarely ready to be consumed and must be transformed by a carefully selected sequence of preprocessing operations.

On one hand, there are plenty of reasons that can explain why a data source cannot be used directly and require preprocessing: too many variables, imbalanced dataset, missing values, outliers, noise, specific domain restriction of the algorithms, etc. On the other hand, data preprocessing has a huge impact on the model performances [Crone 2006, Dasu 2003, Nawi 2013].

The data pipeline depends both on the data source and the algorithm such that there is no universal pipeline that can work for every data source and every algorithm [Wolpert 1996]. The data pipeline is usually defined by trial and error, using the experience of data scientists and the expert knowledge about the data. This step is so crucial it may represent up to 80% of data scientist time [Chessell 2014].

We recall that the main AUTOML problem consists in solving the following black-box optimization problem: given a dataset,

$$\text{Find } \boldsymbol{\lambda}^* \in \operatorname*{arg\,max}_{\boldsymbol{\lambda}\in\boldsymbol{\Lambda}} \mathcal{F}(\boldsymbol{\lambda}), \tag{8.1}$$

where $\boldsymbol{\Lambda}$ is the space of machine learning configuration, and $\mathcal{F}(\boldsymbol{\lambda})$ the performance of the model learned over the dataset using the configuration $\boldsymbol{\lambda}$.

This chapter presents a two-stage optimization approach to solve the AUTOML problem. Specifically, we are interested in understanding the respective impact of the data pipelines and the algorithm configurations on the model performances. Additionally, we study the best allocation of time between the two phases of the machine learning workflow.

To the best of our knowledge, most of AUTOML systems tackle the problem by aggregating the data pipeline operators, the algorithm portfolio and their respective configuration space into a single gigantic search space (corresponding to $\boldsymbol{\Lambda}$ in Eq. (8.1)). In the approach presented in this thesis, we divided the search for a solution into two steps, namely the data pipeline construction and configuration, and the algorithm selection and configuration.

Under the assumption that the two steps are roughly independent, the two-stage optimization brings the two following advantages:

1. by splitting the search space into two smaller ones, it speeds up the overall optimization process, as showed in [Quemy 2020a]
2. it is possible to statistically assess if a data pipeline is specific to an algorithm or rather *universal* w.r.t. the dataset, enabling meta-learning at a lower granularity level (see Section 8.5), as showed in [Quemy 2019b].

The rest of the chapter is organized as follows: a reformulation of the CASH problem is proposed in Section 8.1. The two-stage optimization process is described in Section 8.2, followed by an experimental validation presented in Section 8.3 and discussed in Section 8.4. Finally, Section 8.5 presents an indicator to evaluate the dependency of a data pipeline to the algorithm.

8.1 Data Pipeline Selection and Hyperparameter Optimization

8.1.1 Operators and pipelines

As mentioned before, most approaches model pipelines as fixed ordered sequences of m algorithms and define for each step a specific set of algorithms that can be used. We define a more general version of pipelines, and then constrain this definition for a tradeoff between practical usage and flexibility.

Definition 8.1.1 (Worflow Operator). *An operator parametrized by $\gamma \in \Gamma$ is a function defined by*

$$\begin{aligned} a_\gamma : \mathcal{X}_1^{n_1} &\to \mathcal{X}_2^{n_2} \times \mathcal{X}_2^{\mathcal{X}_1} \\ \mathcal{D}_1 &\mapsto \mathcal{D}_2, T_{\mathcal{D}_1}. \end{aligned}$$

The function operates over a dataset expressed in a certain representation space $\mathcal{X}_1$ and expresses it into another representation space $\mathcal{X}_2$. The size of the dataset may be modified (e.g. rebalancing operation), the input and output space may be the same (e.g. removing outliers) and the dimension of the input and output space may differ (e.g. Principal Component Analysis). The operator initially operates using the whole dataset (e.g. PCA or missing values imputation using a value derived from the available data such as the mean or median). It also returns a functor from $\mathcal{X}_1$ to $\mathcal{X}_2$ that is used to project a single element during the prediction phase. For instance, a PCA on the training set expresses this dataset in a k dimensional space. The associated functor is the projector from the original space to the new space. For a rebalancing operator, as it operates only on the training set, the functor is the identity. Note that the functor is implicitly parametrized by γ (e.g. the parameter k in a PCA is forwarded to the functor).

Definition 8.1.2 (Compatible Workflow Operators). *Two operators are* **compatible** *if the representation output space $\mathcal{X}_2$ of the first operator is the same as the representation input space $\mathcal{X}_1$ of the second one.*

Definition 8.1.3 (Machine Learning Pipeline). *A machine learning pipeline* **p** *as a Directed Acyclic Graph (*DAG*) with two distinguished nodes: a source, from which all paths start, and a sink, from which all paths end. The other nodes are Worflow Operators.*

The words *source* and *sink* come from Flow Network vocabulary. The source is the node by which the data enter, and the sink produces the solution to the machine learning problem. Generally, the nodes represent operators or algorithms. Note that the sink is not necessarily a machine learning algorithm but might also be an operator that aggregates the result of several algorithms (or paths) in an ensemble fashion.

Definition 8.1.4 (Compatible Machine Learning Pipeline). *A pipeline is said* **compatible** *if all the connected nodes are compatible.*

8.1.2 DPSH **problem**

We propose a reformulation of the CASH that better takes into account the nature of machine learning pipelines and the way practitioners work.

Definition 8.1.5 (Data Pipeline Selection and Hyperparamater Optimization)**.** *Given a collection of operators $\mathcal{A}$, the Data Pipeline Selection and Hyperparamater Optimization (DPSH) problem is defined by:*

$$\mathbf{p}^*_{\gamma^*} = \underset{m\in\mathbb{N},\mathbf{p}\in\mathcal{G}_m(\mathcal{A}),\gamma\in\boldsymbol{\Gamma}(\mathbf{p})}{\arg\min} \frac{1}{k}\sum_{i=1}^{k}\mathcal{L}(\mathbf{p}_\gamma, \mathcal{D}^{(i)}_{train}, \mathcal{D}^{(i)}_{test}), \tag{DPSH}$$

where $\mathcal{G}_m(\mathcal{A})$ is the set of compatible pipelines with m vertices selected in $\mathcal{A}$ (with possible replacement), and $\boldsymbol{\Gamma}(\mathbf{p})$ the cartesian product of the configuration space of each operator in the pipeline $\mathbf{p}$. The size m of the sequence is unknown a priori.

8.1.3 Pipeline prototypes

Current approaches that intend to handle the pipeline construction uses a fixed sequence of few steps with a strict order, without possibility to change this topology. While this is useful for end-user with zero knowledge, it is also too restrictive for more complex workflows or end-user with slightly more advanced expertise. On the contrary, the problem depicted by DPSH is far too general to be directly handled because of the enormous number of operators one can imagine, and the number of possible compatible pipelines built over them.

Figure 8.1 depicts two real-life pipelines created by two different pipeline modelers. They are more complex than the usual small pipelines studied in the AutoML literature. Also, the network topology is highly different from one to another, notably because each pipeline is designed to answer a specific question on specific data. While the data scientist might not know exactly what concrete operations will perform the best, she has a general idea of the topology and some orders on the operations. This knowledge can be translated into constraints on the set of compatible pipelines.

Definition 8.1.6 (Pipeline Prototype)**.** *A Pipeline Prototype is a particular graph topology organized in* **layers** *such that all nodes in a given layer are instanciated from a same predifined set of operators.*

In practice, each layer groups specific operations by their purpose, with a level of granularity to be decided by the user. For instance, a layer could be specifically dedicated to imputation if the user knows some values are missing, or generically labelled as *feature engineering* with all possible sorts of preprocessing operators. An overly specialized prototype with little choice for each node or layer is easy to optimize but restricts the search space and thus the possibility to find outstanding configurations. On the contrary, too broadly defined prototype makes the search very difficult as a lot of combinations are either non-feasible or lead to poor results. For instance, in Figure 8.1, left part, a possible prototype could be made of four layers: data preprocessing (orange), model building (green), ensembling (dark blue) and model aggregation (light blue).

To control the pipeline compatibility, two combined approaches are used at the *software* level:

- for each operator, we defined the input and output space in terms of types (e.g `int`, `float`, dictionary, vector of mixed types, etc.); when a pipeline is selected, a fast preliminary check can be done by graph traversal without having to feed the pipeline,

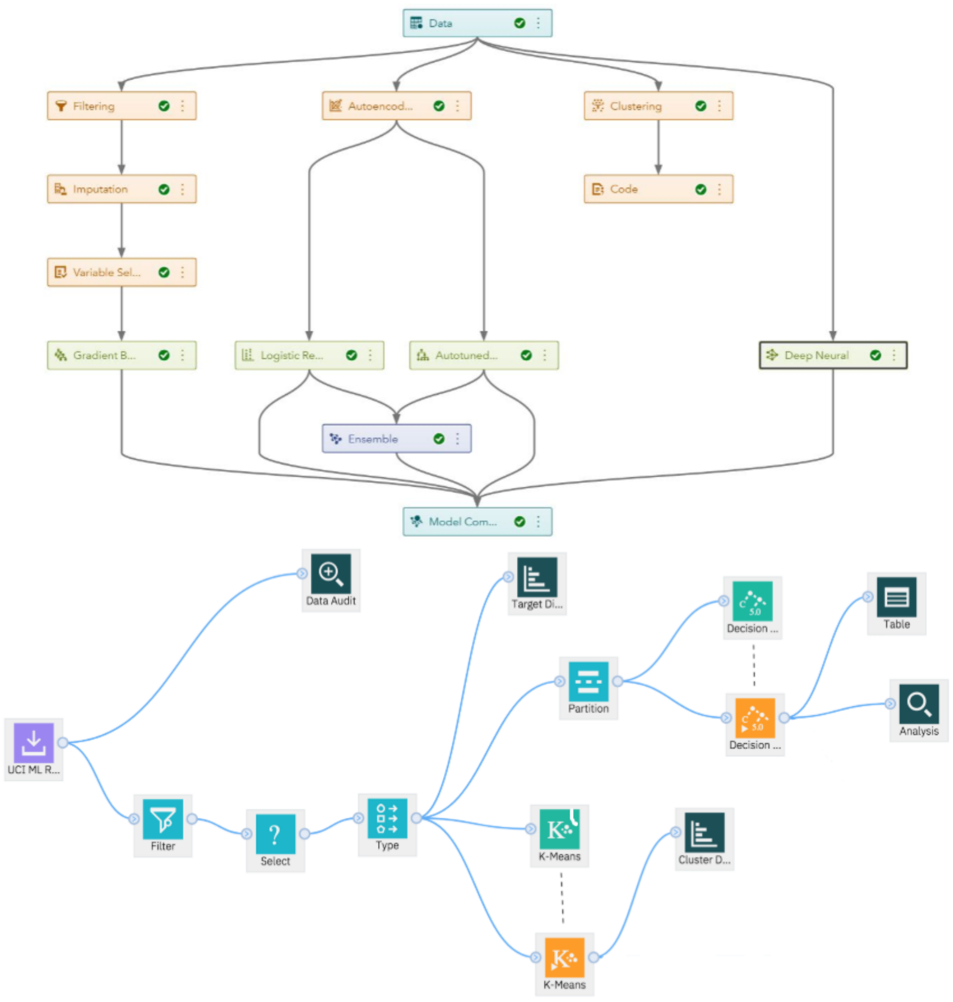


Figure 8.1: Example of real-life pipelines designed with SAS (top) and IBM Watson Studio (bottom).
The flow can be complex, and the topology totally depends on the problem to solve.

- during the execution, non-compatible pipelines throw an exception that can be caught as early as possible.

In both cases, the pipeline is declared *incompatible* by setting the loss function to $+\infty$, with the benefit of preventing the metaoptimizer to explore close areas of the search space, most likely incompatible as well.

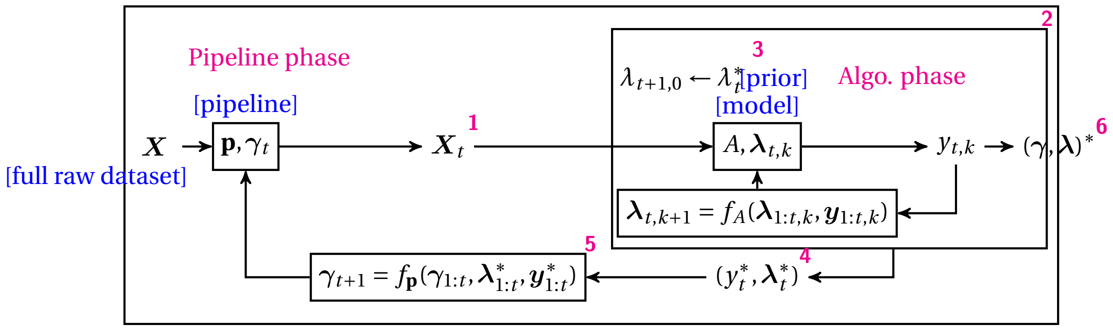
Working with **pipeline prototypes** allows to drastically reduce the search space of DPSH while aligning on real-life practices.

8.2 Two-stage optimization under time-budget

To solve this problem, rather than considering one large search space, we break the optimization process into two smaller problems: searching for a good pipeline, and searching for a good algorithm selection and configuration.

The rationale behind breaking down the AutoML optimization process lies into the distinct nature of the two steps described by Figure 1.1. *Feature engineering* deals for most with improving the quality of the data, like a craftsman that transforms raw material into remarkable and consumable objects. In general, it has little to do with the person that will buy the object, or to speak less metaphorically, with the algorithm that consumes the preprocessed data. Obviously, this assumption is true up to a certain point since algorithms might have different structural requirements on their input space (e.g. categorical features for tree-based methods, and the influence of encoders on their performances), and some algorithms might be more sensitive to some preprocessing steps due to their mathematical properties. For instance, for a given time budget, a dimensionality reduction grants, theoretically at least, a better advantage to an algorithm with a $\mathcal{O}(N^2)$ time complexity rather than a $\mathcal{O}(\log N)$ algorithm where N is the initial input space dimension. Indeed, assuming an initial dimension N and a reduction to $K << N$ dimensions, the time reduction is $N^2 - M^2$ for the first one, and $\log(\frac{N}{l}M)$ for the second one. Of course, this rough estimatation hides the lower order terms in the Landau notation, and represents the asymptotic best case time reduction.

8.2.1 Architecture


Pipeline phase
[pipeline]
X
[full raw dataset]
$\mathbf{p}, \gamma_t$
X_t
1
2
3
$\lambda_{t+1,0} \leftarrow \lambda_t^*$ [prior]
Algo. phase
[model]
$A, \boldsymbol{\lambda}_{t,k}$
$y_{t,k}$
$(\gamma, \boldsymbol{\lambda})^*$
6
$\boldsymbol{\lambda}_{t,k+1} = f_A(\boldsymbol{\lambda}_{1:t,k}, \boldsymbol{y}_{1:t,k})$
4
$(y_t^*, \boldsymbol{\lambda}_t^*)$
5
$\gamma_{t+1} = f_{\mathbf{p}}(\gamma_{1:t}, \boldsymbol{\lambda}_{1:t}^*, \boldsymbol{y}_{1:t}^*)$

[1] A single pipeline transforms the whole dataset at each iteration.
[2] The output $y_{t,k}$ in the inner loop is a validation measure (e.g. cross-validation).
[3] The inner loop is initialized with the previous best configuration as prior.
[4] The inner outputs the best prediction and configuration at iteration t.
[5] f_M returns the best promising configuration w.r.t. the best achievable metric.
[6] The whole process returns the best configuration to be used in production.

Figure 8.2: Two-stage optimization process.

The proposed two-stage optimization process is illustrated by Figure 8.2. Due to the sequential nature of the machine learning workflow, it consists in two imbricated loops. The inner loop performs a cross-validation on $\mathbf{X}_t$ for a given time budget or until a Cauchy criterion is verified (e.g. the difference in the cross-validation score between two iterations of the inner loop is lower than a threshold ε).

The rationale behind using the previous optimal algorithm configuration $\boldsymbol{\theta}_t^*$ as prior for the next inner loop is that if the "distance" between the processed data $\mathbf{X}_t$ at iteration t and $\mathbf{X}_{t+1}$ is small, we can expect the new optimal configuration for the algorithm to be rather close, shortening the inner loop computation time if a Cauchy criterion is used.

Notice that this architecture is independent of the metaoptimizer. Even more, different metaoptimizers can be used for the two stages, and different optimization criteria might be used for each step. For instance, one can imagine optimizing the data pipeline for a fairness criterion and a standard performance metric such as the (cross-validation) accuracy for the algorithm.

8.2.2 Policies

In practical situations, the limiting factor to construct a machine learning workflow is time. Therefore, we are interested in optimization under time constraint, and thus finding how to allocate time between both steps and through iterations. Given a time budget of T, we define different policies of time allocation.

Split policy: The budget is split between T_1 and T_2, allocated respectively for the data pipeline configuration search and the algorithm hyperparameter tuning. In the first phase, the metaoptimizer is used during T_1 to build the data pipeline. During the second phase, the metaoptimizer is used to configure the algorithm during T_2.

Considering the convex combination $T = (1-\omega)T_1 + \omega T_2$, for $\omega = 0$, no hyperparameter tuning is done because the whole budget is spent on building the data pipeline. Conversely, for $\omega = 1.0$, the process is fully dedicated to tune hyperparameters of the algorithm.

Iterative policy: Each step alternates during a short runtime of t seconds. The best configuration found during a step is reused during the next step, iteratively until the total budget is expired. It is relatively fast for the metaoptimizer to find a better pipeline configuration than the baseline, but then, stagnates. Therefore, the time would be better allocated to the search for a better algorithm configuration. If the data pipeline was modified, the data used to train the algorithm changes. Those variations might help during the hyperparameter tuning to explore a new region of the search space compared to the previous iteration.

Adaptive policy: This policy reuses the iterative policy. However, the time allocation is not fixed per iteration. If during an iteration the cross-validation score improved, the time allocated to the next iteration of the same type (data pipeline or algorithm configuration) is multiplied by two. Conversely, if after two iterations of the same type, the score is not improved, the allocated time for this type of iteration is divided by two.

Joint policy: This policy simply uses the union of both search spaces. In other words, it is equivalent to what is usually done in practice by current meta-optimizers, i.e., searching over the whole space of machine learning pipelines.

8.3 Experimental Setting

In this section, we present the experiments to valide the approach described above. For our purpose, we consider a simplified version of the problem where the algorithm is given such that we are left with its configuration.

8.3.1 Experimental setting

Datasets and algorithms. We performed the experiments on three datasets: `wine`[1], `iris`[2] and `breast`[3][4]. We used four classification algorithms: SVM, Random Forest, Neural Network, and Decision Tree. The implementation is provided by SCIKIT-LEARN[Pedregosa 2011].

Data pipeline search space. We created a *pipeline prototype* made of three layers: "rebalance" (handling imbalanced dataset), "normalize" (scaling features), "features" (feature selection or dimension reduction). For each step, we selected few possible concrete operators with a specific configuration space summarized in Table D.1. The pipeline topology is defined by Figure 8.3. Each node can be instantiated with an operator or left empty. When both "features selection" slots are instantiated, the final vector is obtained by stacking the output of both operators. For instance, for "features", there is the choice between a PCA with keeping 1 to 4 axes, selecting the $k \in \{1, ..., 4\}$ best features according to an ANOVA, or a combination of both. There is a total of 4750 possible pipeline configurations. This is roughly the same as in AUTOSKLEARN or MOSAIC, and about 5 times less than AUTOWEKA.

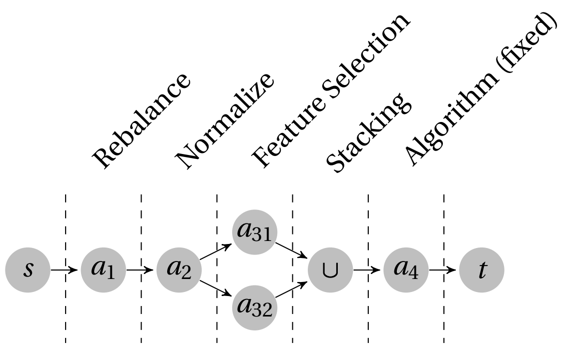


Figure 8.3: Pipeline topology created for the experiments.

Algorithm hyperparameter search space. For each algorithm, we defined a reasonable hyperparameter space with approximately the same size as the data pipeline space. The search space size contains 4800 elements for Random Forest and Decision Tree, 1944 for Neural Network and 768 for SVM.

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html`

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html`

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html`

[4] The choice of small datasets is justified by the need to know the optimal score in the search space to effectively evaluate the metaoptimizer and policies results.

Metaoptimizer. We selected HYPEROPT [Bergstra 2015] as metaoptimizer as it has been shown to perform better than alternatives for high dimensional problems [Eggensperger 2013]. A 10-fold cross-validation is used to assess the pipeline performances.

Resources. The machine used for the experiments is equipped with an Intel i7-6820HQ and 32GB RAM.

8.3.2 Protocol and goals

We decompose the experiments in two parts. First, given a restricted budget, we want to assess the respective influence of searching for a data pipeline and configuring the algorithm. The second part focuses on the optimization with a time budget and the evaluation of two-stage optimization process, and in particular the different policies. All experiments are reproducible and the source-code is documented and available in a dedicated GitHub repository[5].

Experiment 1. We would like to quantify the impact of each phase on the final result. To this end, we proceed in two steps. First, we perform an exhaustive search in the data pipeline search space, followed by a search using HYPEROPT with a budget of 100 configurations to explore (about 2% of the configuration space). The algorithm uses the default configuration of its implementation. In a second step, we perform the same for hyperparameter tuning with the baseline data pipeline. Those two steps have been repeated for each algorithm and each dataset. We report the density of configurations depending on the accuracy, both for the exhaustive search and for the restricted budget.

Experiment 2. We ran the two-stage optimization process for $T = 300$ seconds, for each dataset, each method and each policy. For the **split policy**, we performed the experiment for each $\omega \in \{0, 0.1, ..., 0.9, 1\}$ to show the effect of different allocations between the two stages. For the **iterative policy**, we setup the iteration runtime to 15 seconds. Similarly, the default iteration time for the **adaptive policy** has been fixed to 15 seconds.

8.4 Results

8.4.1 Experiment 1

Figures 8.4 and 8.5 provide the result obtained with Random Forest on `breast` and `wine`. A summary of the results is provided by Table 8.1. All results are qualitatively similar.

Figure 8.4, on `breast`, shows that the baseline score is 0.9384 and the best score 0.9619 i.e. an error reduction of 38% is achievable in the pipeline search space. Similarly, Figure 8.5 for `wine`, the best accuracy is 0.9906, i.e. a reduction of 25% of the error rate. Most configurations deteriorate the baseline score. However, HYPEROPT is skewed towards better configuration compared to the exhaustive search. It indicates HYPEROPT has a higher probability to find a good configuration than random search. The right parts show that HYPEROPT starts to

[5] `https://github.com/aquemy/DPSO_experiments`

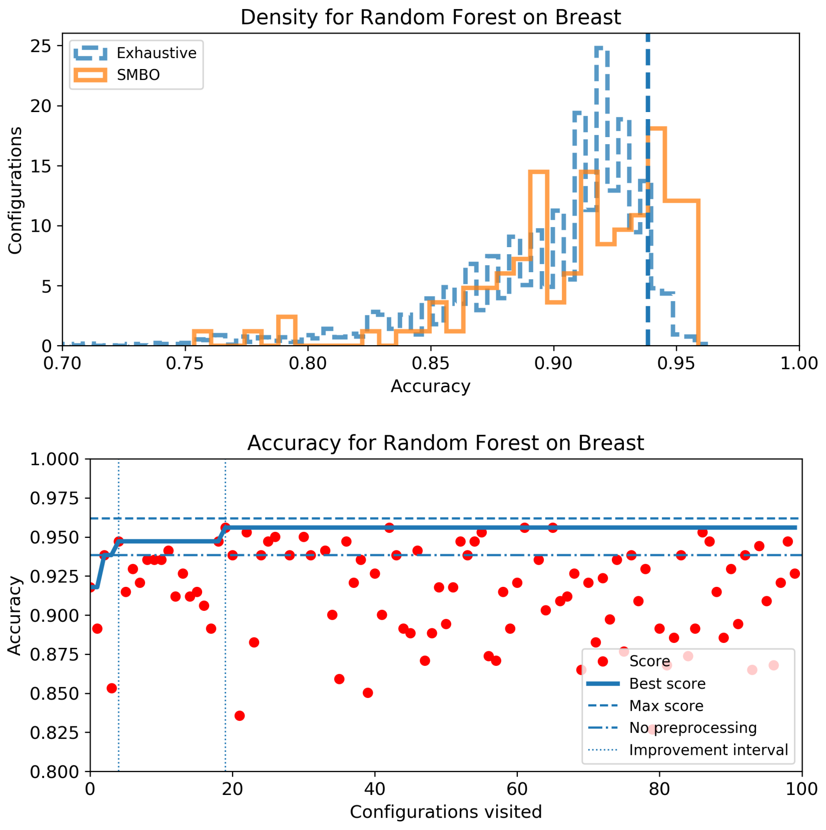


Figure 8.4: Density of pipeline configurations (top) for `breast`. The vertical line represents the baseline score. Evolution of the accuracy iteration after iteration (bottom).

improve the baseline score after only 4 iterations and reached its best configuration after 19 iterations on `breast` (resp. 5 and 20 for `wine`).

Table 8.1 shows that similar results are obtained for all methods on all datasets. HYPEROPT always found a better pipeline than the baseline, in at most 17 iterations. In average, the best score is achieved around 20 iterations (excluding Decision Tree on `iris` and `breast`). Decision Tree was able to reach the optimal configuration on `iris` (resp. `wine`) after one (resp. five) iteration. In general, the score in the normalized score space belongs to $[0.9780, 1.000]$. To summarize, in average, with 20 iterations (0.42% of the search space) HYPEROPT is able to decrease the error by 58.16% compared to the baseline score and found configurations that reach 98.92% in the normalized score space.

Exhaustive results for the algorithm step are better for Random Forest and Decision Tree on `breast` only and similar to the data pipeline one only for Random Forest on `wine`. In all other nine cases, the optimal score reachable in the search space is better for the data pipeline phase.

Regarding the score after 100 iterations, HYPEROPT found a higher accuracy for the data pipeline construction in all cases, except for Random Forest on all datasets, and SVM on `iris` dataset.

In conclusion, for a similarly large and realistic search space, and a given number of con-

Table 8.1: Optimization results.

		Exhaustive		Cross-validation Score				Imp. Interval	
	Baseline	DP	Algo.	DP	DP (norm.)	Algo.	Algo. (norm.)	DP	Algo.
				`iris`					
SVM	0.9667	**0.9889**	-	0.9778	0.9831	**0.9866**	-	[11, 11]	[2, 10]
Random Forest	0.9222	**0.9778**	0.9667	**0.9667**	0.9828	**0.9667**	1.0000	[8, 27]	[1, 70]
Neural Net	0.9667	**0.9889**	0.9778	**0.9778**	0.9831	0.9667	0.9820	[17, 17]	[1, 11]
Decision Tree	0.9222	**0.9889**	0.9667	**0.9889**	1.0000	0.9667	1.0000	[1, 83]	[12, 36]
				`breast`					
SVM	0.9501	**0.9765**	-	**0.9765**	1.0000	0.9474	-	[12, 20]	-
Random Forest	0.9384	0.9619	**0.9765**	0.9560	0.9780	**0.9685**	0.8982	[4, 19]	[4, 46]
Neural Net	0.9326	**0.9765**	0.9472	**0.9707**	0.9903	0.9175	0.9600	[1, 7]	[3, 13]
Decision Tree	0.9296	0.9619	**0.9648**	**0.9589**	0.9900	0.9527	0.9605	[0, 67]	[13, 23]
				`wine`					
SVM	0.9151	**1.0000**	0.9728	**0.9906**	0.9811	0.9728	1.0000	[3, 13]	[1, 3]
Random Forest	0.9623	**0.9906**	**0.9906**	0.9811	0.9818	**0.9906**	1.0000	[5, 20]	[1, 23]
Neural Net	0.9057	**0.9906**	0.9434	**0.9906**	1.0000	0.9245	0.9778	[1, 25]	[1, 13]
Decision Tree	0.9057	**0.9811**	0.9528	**0.9811**	1.0000	0.9339	0.9671	[5, 35]	[11, 75]

DP and *Algo.* represents respectively the Data Pipeline phase and the algorithm phase. The column *norm.* is the cross-validation score normalized within the search space. The last column is the interval where the left bound is the number of configurations required for HYPEROPT to improve the baseline score, and the right, the number of configurations before reaching the best score.

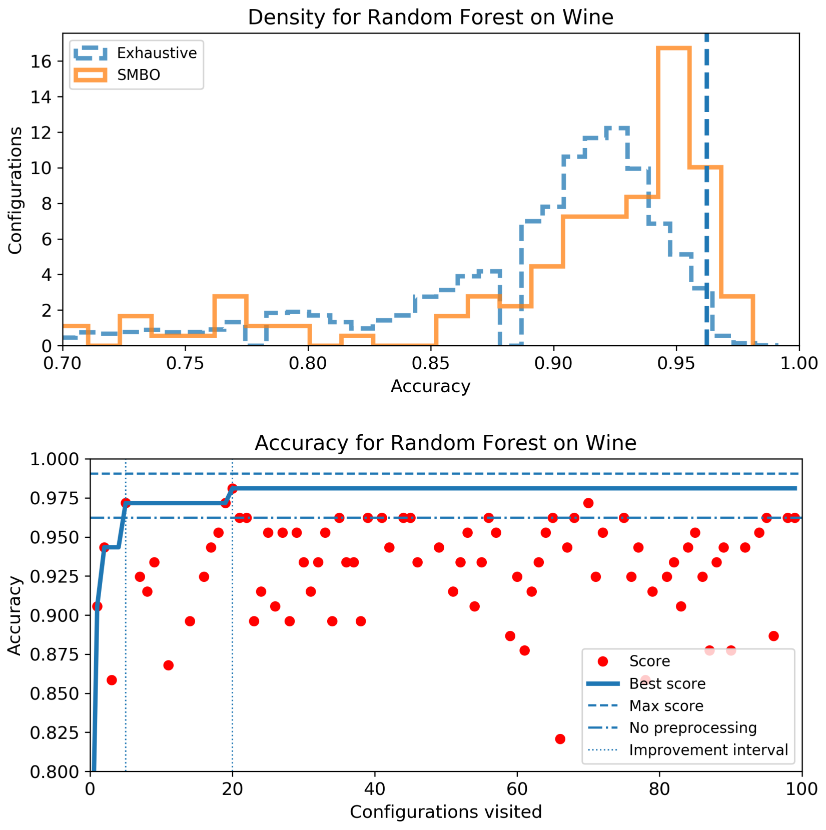


Figure 8.5: Density of pipeline configurations (top) for `wine`. The vertical line represents the baseline score. Evolution of the accuracy iteration after iteration (bottom).

figurations to explore, a good data pipeline with a default algorithm configuration provides better results than a tuned algorithm with no pre-processing operations. A notable exception is Random Forest that largely benefits from hyperparameter tuning.

8.4.2 Experiment 2

Figure 8.6 shows the accuracy depending on different time allocations between each phase. By construction, the accuracy reached on the second phase can be only equal or higher to the best accuracy reached by the first phase. For the same method, depending on the dataset it might be better to allocate the whole budget on a phase or another. This is the case for Decision Tree: allocating the whole budget to the algorithm configuration on `breast` returns better results than spending the whole budget on the data pipeline construction. The exact opposite is observed on `wine`. In general, the best accuracy is obtained by a tradeoff between the two phases which seems to depend both on the algorithm and the dataset. Therefore, more advanced time allocation policies such as iterative or adaptive may help. Learning to predict optimal time allocation for a new dataset and algorithm based on previous runs is left for future work.

In Figure 8.7, we reported the evolution of the best accuracy through time, for each policy

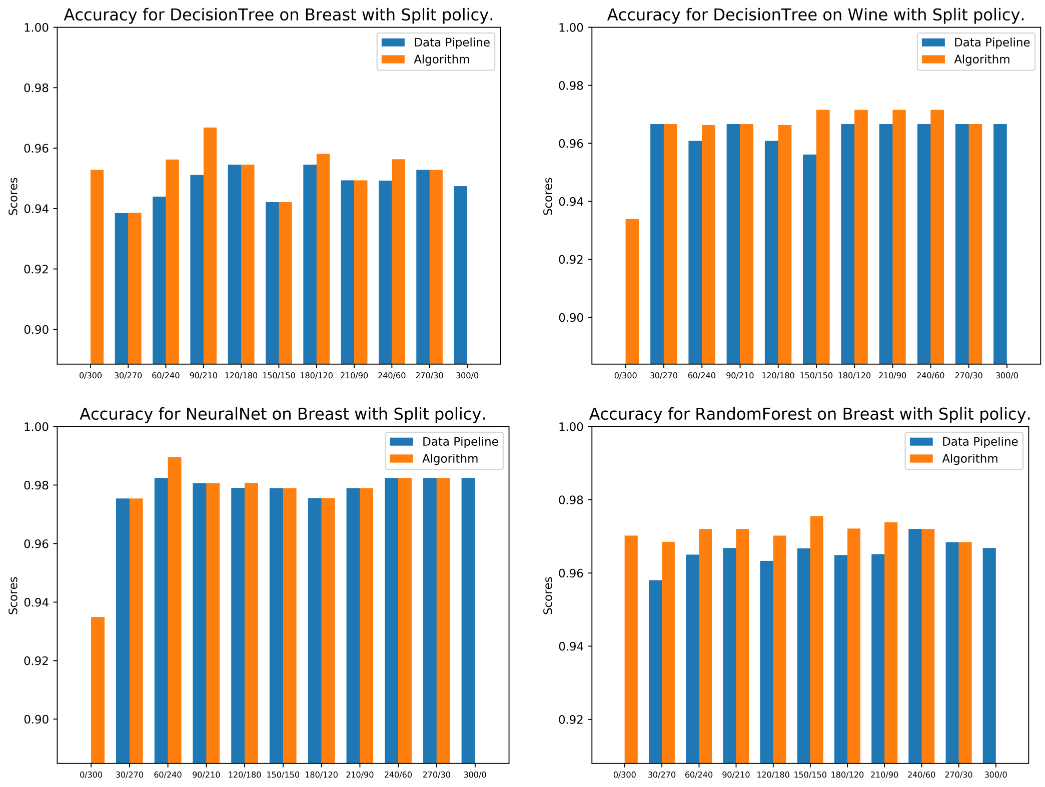


Figure 8.6: Accuracy depending on the time spent on each phase of the optimization process.

defined in Section 8.2.2. The Joint policy returns a lower or equal accuracy in all cases but Random Forest on `wine`. In general, it is far slower to reach a good score compared to other policies except for Random Forest on `iris` where it is the first policy to reach the plateau. It is particularly visible for Decision Tree on `iris` where all policies behave the same and reach the same score. In this case, the Joint policy took about 20 seconds to reach the best score against 5 seconds for the second worst and 2 seconds for the best.

The results of Split 0 policy with Neural Net are really poor for all datasets. The reason is the little number of configurations visited due to the time required to train the algorithm without feature selection. On the contrary, Split 0 performs relatively well for Random Forest on all datasets. This seems to indicate that the importance of both optimization stages are different depending on the algorithm.

The Adaptive policy reaches the best score in five cases, and Iterative policy in three cases. In general, at the exception of Decision Tree on `wine`, the Adaptive policy manages to reach similar or better scores with less configurations sampled. The best examples are Decision Tree on `breast` where the Adaptive policy reaches the best score but samples only 251 configurations against 320 and 427 for the second best policies, as well as Neural Network on `iris` with 49 configurations versus 22 and 124 for a lower score.

From those results, it appears that the Iterative and Adaptive policies are the most robust ones to quickly provide good results for any method and any dataset. However, depending on the specificities of the method and the dataset, spending 100% of the time on one phase

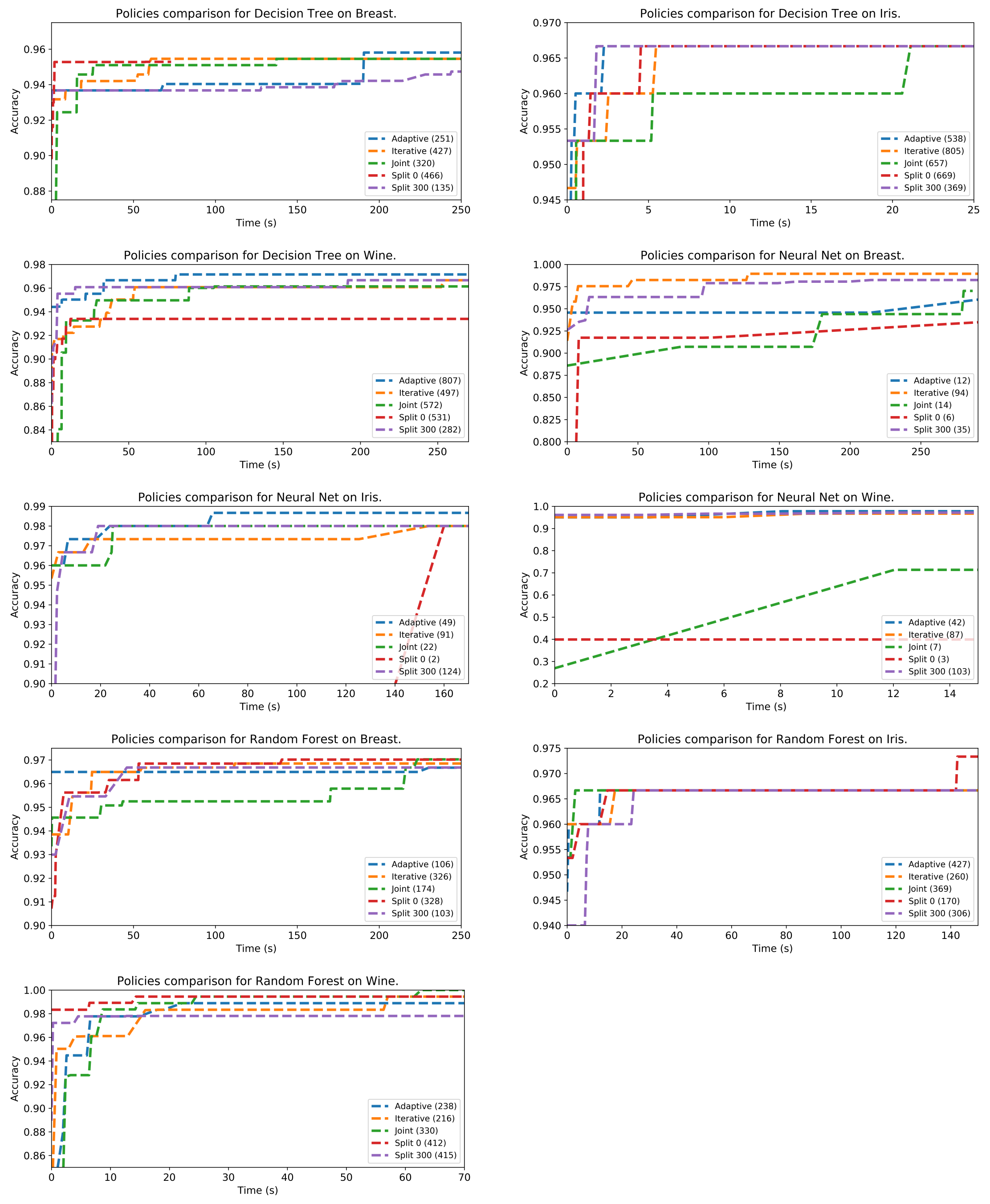


Figure 8.7: Evolution of the best score in time for different policies.

Split 300 and Split 0 are respectively fully dedicated to the data pipeline selection or the algorithm configuration. In brackets is indicated the number of configuration visited.

or the other might be better. In this case and in theory, the Adaptive policy is equivalent to one of those Split policies after some time, i.e. the optimization process spends most of its time on one phase.

8.5 Algorithm-specific configuration

A pipeline is obviously specific to the dataset or the data distribution it works on. However, our initial assumption was that the pipeline might be more or less independent from the algorithm. Therefore, we would like to quantify how much an optimal configuration is specific to an algorithm or is *universal*, i.e. works well regardless of the algorithm. To this end, the optimization process might be performed on a collection of methods $\mathcal{A} = \{A_i\}_{i=1}^{N}$. The result is a sample of optimal configurations $\mathbf{p}^* = \{p_i^*\}_{i=1}^{M}$ where $M \geq N$ since an algorithm might have several distinct optimal configurations. After normalizing the configuration space to bring each axis to $[0,1]$, the link between the processed data and the methods can be studied through a new indicator named Normalized Mean Absolute Deviation (NMAD) defined below. The idea behind this metric is to measure how much the optimal points are distant from a reference optimal point. If the optimal configuration does not depend on the algorithm, the expected distance between the optimal configurations is 0. Conversely, if a point is specific to an algorithm, the other points will be in average far from it.

Working in the normalized configuration space has two advantages. First, it forces all parameters to have the same impact. Secondly, it allows the comparison from one dataset to another since the NMAD belongs to $[0,1]$ for any number of algorithms or dimensions of the configuration space.

The Mean Absolute Deviation of a sample is a vector summarizing the average distance from a reference point r.

Definition 8.5.1 (Mean Absolute Deviation)**.**

$$MAD(\mathbf{p}^*, r) = \frac{1}{N}\sum_{i=1}^{N} |p_i^* - r|$$

The NMAD is the norm 1 of the Mean Absolute Deviation[6], divided by the number of dimensions K of the configuration space.

Definition 8.5.2 (Normalized Mean Absolute Deviation (NMAD))**.**

$$NMAD(\mathbf{p}^*, r) = \frac{1}{K}\frac{1}{N}||\big(\sum_{i=1}^{N} |p_i^* - r|\big)||_1$$

To measure how much each optimal point p_i^* is specific to an algorithm A_j, we use it as a reference point and calculate the NMAD using a sample composed of all the optimal points. However, an algorithm might have several optimal points and to be fair, we use as a representant of each algorithm, the closest point to the reference point.

[6]As we work on a discrete space, we used the norm 1, but the Euclidean norm is probably a better choice in continuous space.

8.5.1 Experimental Settings

We are interested in evaluating if the NMAD indicator is an effective metric to determine if a configuration is algorithm-specific or rather *universal*. To this end, we perform an exhaustive search on two different datasets and analyse the optimal configurations in the light of the computed NMAD.

As the configuration space described in Section 8.3.1 is not a metric space, we cannot directly use the NMAD. To avoid introducing bias with an *ad-hoc* distance, we perform another experiment with a configuration space that is embedded in $\mathbb{N}$.

As a dataset, we used 1000 judgements documents about the Article 6 from European Court of Humain Rights Database (ECHR-DB) described in Chapter 5. The ground truth corresponding to a violation or no violation. The cases have been selected such that the dataset is balanced. The conclusion part is removed. To confirm the results, we used a second dataset composed of 855 documents from the categories atheism and religion of `20newsgroups`.

Each document is preprocessed using a data pipeline consisting in tokenization, stop-words removal, followed by a n-gram generation. The processed documents are combined and the k most frequent tokens across the corpus are kept, forming the dictionary. Each case is turned into a Bag-of-Words using the dictionary.

There are two hyperparameters in the preprocessing phase: n the size of the n-grams, and k the number of tokens in the dictionary. We defined the parameter configuration domain as follows:

- $n \in \{1,2,3,4,5\}$,
- $k \in \{10,100,1000,5000,10000,50000,100000\}$.

We used the same four algorithms as in Section 8.3. As we are interested in the optimal configurations, we performed an exhaustive search.

8.5.2 Results

For both datasets, Figure 8.8 shows that the classifier returns poor results for a configuration with a dictionary of only 10 or 100 tokens. Both parameters influence the results, and too high values deteriorate the results.

Table 8.2 summarizes the best configurations per method. For the first dataset, there are 3 points that gives the optimal value for Random Forest and Linear SVM, however, in practice lowest parameters values are better because they imply a lower preprocessing and training time. It is interesting to notice that the configuration $(5,50000)$ returns the best accuracy for every model, as this point would be a sort of *universal* configuration for the dataset, taking the best out of the data source, rather than being well suited for a specific algorithm. On the contrary, on `20newsgroups`, all optimal points are different. Our hypothesis is that the more structured a corpus is, the less algorithm-specific are the optimal configurations, because the preprocessing steps become more important to extract markers used by the algorithms to reach good performances. As ECHR dataset describes standardized justice documents, it is far more structured than `20newsgroups`. This would also explain why generating n-grams for $n=5$ still improves the results on ECHR while degrading them on `20newsgroups`.

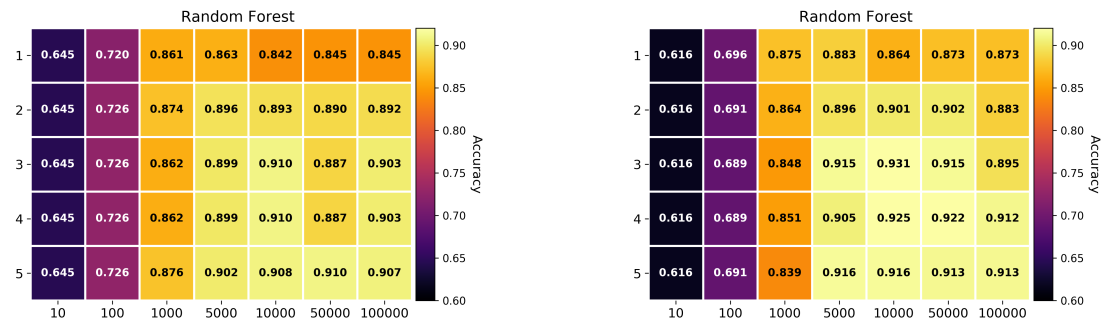


Figure 8.8: Heatmap depicting the accuracy depending on the pipeline parameter configuration on ECHR (left) and `20newsgroups` (right).

Table 8.2: Best configurations depending on the method.

Method	(n, k)	accuracy
	ECHR	
Decision Tree	(5, 50000)	0.900
Neural Network	(5, 50000)	0.960
Random Forest	(3, 10000), (4, 10000), (5, 50000)	0.910
Linear SVM	(3, 50000), (4, 50000), (5, 50000)	0.921
	Newsgroup	
Decision Tree	(4, 5000), (4, 100000)	0.889
Neural Network	(5, 50000)	0.953
Random Forest	(3, 10000)	0.931
Linear SVM	(2, 100000)	0.946

This hypothesis is partially confirmed by Table 8.3, where it is clear that the n-gram operator has a strong impact on the accuracy variation on ECHR dataset (up to 9.8% accuracy improvement) while almost none on the 20newsgroups dataset (at the exception of Random Forest).

Table 8.3: Impact of parameter n on the accuracy, measured as the relative difference between the best results obtained only using $(1,k)$ and the best results obtained for any configuration (n,k).

Method	$p=(1,k)$	$p=(n,k)$	Δ acc
ECHR			
Decision Tree	0.850	0.900	5.9%
Neural Network	0.874	0.960	9.8%
Random Forest	0.863	0.910	5.4%
Linear SVM	0.892	0.921	6.6%
Newsgroup			
Decision Tree	0.885	0.889	0.5%
Neural Network	0.949	0.953	0.4%
Random Forest	0.883	0.931	5.4%
Linear SVM	0.945	0.946	0.1%

Table 8.4 contains the NMAD value for each distinct optimal configuration reported in Table 8.2. As expected, the point $(5,50000)$ has a NMAD of 0 since the point is present for every algorithm. Thus, $(5,50000)$ represents a *universal* pipeline configuration for this data pipeline and dataset. The point $(4,50000)$ appears only once but it is really close to $(5,50000)$ (itself in the 3 other algorithms results) s.t. its NMAD is low. It can be interpreted as belonging to the same area of optimal values. On the opposite, $(3,10000)$ and $(4,10000)$ have high NMAD w.r.t. the other points, indicating they are isolated points and may be algorithm specific. Their NMAD values are rather low because despite the points are isolated, they differ significantly from the others points only on the second component. In comparison, if $(1,10)$ would be an optimal point for Random Forest, its NMAD would be 0.5. On the contrary, for 20newsgroup, the NMAD value is rather high and similar for all points, indicating that they are at a similar distance from each other and really algorithm specific.

To summarize, the NMAD metric is coherent with the conclusion drawn from the heatmaps and Table 8.2, and suggests that there exist two types of optimal configurations: *universal* pipeline configurations that work well on a large range of algorithms for a given dataset, and algorithm-specific configurations. Thus, we are confident the NMAD can be used in larger configuration spaces where heatmaps and exhaustive results are not available for graphical interpretation, and help to reuse configurations and initialize surrogate models.

Table 8.4: Normalized Mean Average Deviation for each optimal configuration found.

ECHR		Newsgroup	
Point	NMAD	Point	NMAD
(5, 50000)	0	(4, 5000)	0.306
(3, 10000)	0.275	(4, 100000)	0.300
(4, 10000)	0.213	(5, 50000)	0.356
(3, 50000)	0.175	(3, 10000)	0.294
(4, 50000)	0.094	(2, 100000)	0.362

8.6 Conclusion

The contributions from this chapters are as follows:

1. We proposed a reformulation of the main AutoML problem to fit better the way data scientists work in practice.
2. We studied the importance of the data pipeline and algorithm hyperparameter tuning phase on a reasonably realistic search space.
3. We found that spending more time on setting a proper data pipeline usually leads to better results.
4. We proposed a two-stage process to optimize a general machine learning workflow, articulated around the data pipeline construction and the hyperparameter tuning.
5. Under time constraint, we studied four different policies to allocate time between these two phases. We found that the best results are obtained with a tradeoff that seems to depend on both the algorithm and the dataset.
6. We found that the most robust policies, in terms of reaching a good score within little time, are the most advanced policies: iterative and adaptive.

In addition, we provided a metric to study if a data pipeline is more or less independent from the algorithm. This metric can be use to suggest good data pipelines depending on meta-features as described in [Bilalli 2017, Bilalli 2018]]. Future work should focus on an online version s.t. the pipeline is tuned in a streaming way. Also, the NMAD indicator works only in euclidian spaces which is not the case for the first experiment. Therefore, further work should focus on extending the NMAD to non-vector space.

More generally, we plan a larger experimental campaign using OpenML-CC18 benchmark suit [Bischl 2017] with different pipeline prototypes.

A generalization to multi-stage optimization is also considered. Each layer of the pipeline could benefit from a specific time allocation depending on its marginal contribution. Some preliminary work has been done toward this direction in [Chowdhury 2019].

Last but not least, we are currently working on a library based on the code of those experiments, dedicated to the creation of flexible pipelines and budget allocation strategies. By using the operator and pipeline prototype definitions provided in this thesis, we will provide an easy yet generic way to define and extend pipeline search spaces, which is currently not feasible by any AUTOML systems.

CHAPTER 9

Summary and Conclusions

In this doctoral thesis, we addressed two main research questions:

1. **can we learn a classification model in a space without any metric?**
2. **should we shift our focus from algorithm to data preparation? If yes, how to efficiently automate this preparation phase?**

To answer the first question, we proposed a new mathematical framework called HCBR for classification based on hypergraph representation and that can work in spaces without meaningful metric. The focus was put on the robustness and the easiness of use by non-experts. The training algorithm converges, and has few hyperparameters that does not even need to be tuned. These hyperparameters allow fine grain control over the model calibration. The model calibration usually depends on the application. For instance, one can decide not to have false positive, e.g. for the judicial decisions, or not to have false negatives, e.g., for medical diagnosis. We demonstrated the method robustness accross several datasets as it outperformed most algorithms for standard performance metrics such as accuracy, F1-score or MCC. It allows any non-expert users to obtain prediction by simply stacking any source of information e.g., textual, descriptive, etc.

Despite these good properties, there are many possible improvements left for future work. First, the model space complexity seems to be too limited and we offered two ways of extending it by taking more advantage of the hypergraph representation than what has been done here. Second, as most real-life problems happens in non-stationary environment, we would like to investigate an efficient online version of HCBR, that is to say, a version that can evolve with every new cases. Additional theoretical results are also needed to fully understand the link between the initial hypergraph representation and the final model performance. Last, we suggested that explanation about each decisions is easy to obtain but did not provide a clear way of presenting it to the user, which is still under investigation.

In the second part, we studied the problem of extending the linear discriminating rules to a generative continuous decision map. We modeled the problem as a heat propagation problem, which allowed us to find an exact analytical form thanks to Lax-Milgram theorem of decomposition of compact operators in Hilbert spaces. The proposed algorithm acts as a new classification technique with very specific time complexity: it is constant in the size of the training set but exponential in the dimension. Therefore, the technique is perfectly suited for low-dimensional problems, which is our case since HCBR decision space is two dimensional, regardless of the amount of available data. An obvious improvement would be to reduce the exponential complexity in dimension. This is left for future work.

In a third part, we showed that it is possible to use bayesian optimization to automatically construct complex data pipelines. We proposed a two stage optimization program to successfully build the data pipeline and then select and tune the algorithm. We showed that the data pipeline often provide more improvements to performance than spending time on selecting and tuning the algorithm, for reasonable search spaces, **thus answering our second research question**. Then, we proposed some time allocation policies between the two stages of the optimization program. We confirmed the preliminary result: the data pipeline construction often offers a more substantial improvement than selecting and tuning the algorithm. However, we showed that adaptive policies, that is too say policies such that the time spent in one or the other phase varies during the optimization process, lead to the best results. This is an interesting result that opens the door to end-to-end AutoML for other learning methods than Deep Learning. On top of larger scale experimental validation, this work will be extended in the future by a multi-stage optimization program to further optimize the time allocation between the different parts of the pipeline.

From the legal domain perspective, we improve the best accuracy from the literature on the ECHR by 15pp. We extended this work by showing that it is possible to tackle the more general problem of multilabel classification without degrading the performance. Moreover, we showed that in this configuration, with enough data, features available strictly before the judgment are sufficient to provide acceptable results, opening the road to real applications. These results have been obtained thanks to our contribution named ECHR-OD. Indeed, we showed that previous studies were actually using models that underfit. ECHR-OD, offering the largest and most complete database about the ECHR, allows to overcome this limitation. In addition, ECHR-OD provides data in several formats which allows purely descriptive analysis or the construction of complex and performant predictive models. ECHR-OD provides an API such that it can be integrated to any application. Last but not least, to ensure quality, trust and reproducibility in the data, the entire collection of scripts from data collection to curation is made available as open-source. Future work on this project include to regenerate the database every month to include the new cases, and creating more modern embedding using state-of-the-art techniques such as BERT-based methods.

To conclude, in this doctoral thesis we proposed a contribution to each stage of the standard machine learning workflow in order to create a fully end-to-end workflow for the classification problem. The modified workflow allows to work in spaces without meaningful metric, which translate for the user into an easy to use system without being a data scientist or machine learning expert, while still being able to perform well.

Part III

Appendix

APPENDIX A

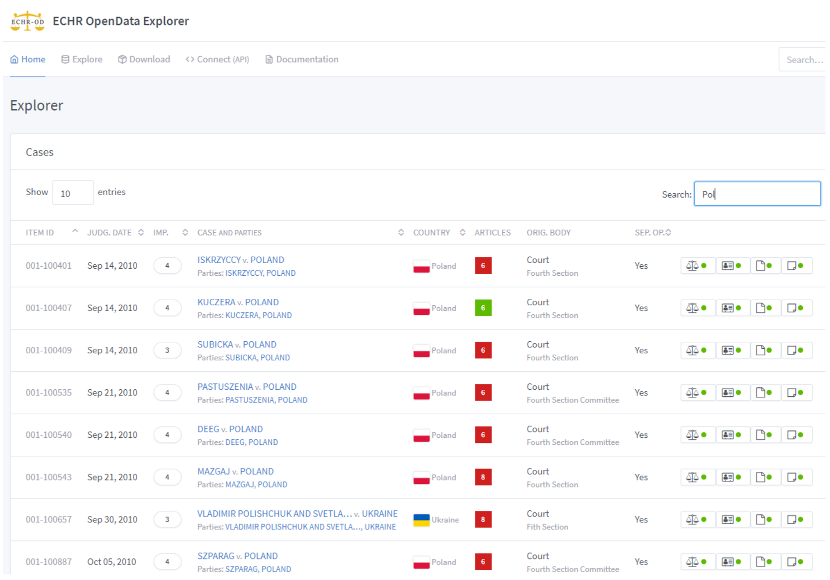
Additional features of *ECHR-DB*

In addition to the final database and files of *ECHR-DB*, a portal has been developped with two main features: an online explorer to browse cases and an API to interface the database with external applications.

A.1 The Explorer

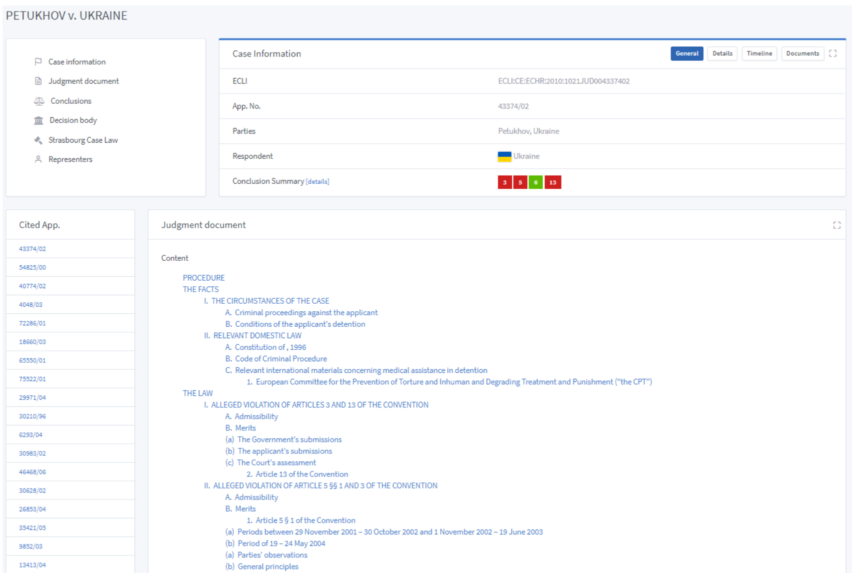
The explorer is a web application that allows to sort and search cases in real time and display every information gathered about a specific case, including the members of the decision body, the timeline, associated documents and detailled conclusion. Additionally, the judgment document is displayed as a tree and the cross-citations are extracted from each document. Figures A.1 and A.2 respectively shows the interface displaying the list of cases and a particular case.

Note that the explorer always uses the SQL database available to download, such that the explorer is always up to date with the latest version of the database.



ITEM ID	JUDG. DATE	IMP.	CASE AND PARTIES	COUNTRY	ARTICLES	ORIG. BODY	SEP. OP.	
001-100401	Sep 14, 2010	4	ISKRZYCCY v. POLAND Parties: ISKRZYCCY, POLAND	Poland	6	Court Fourth Section	Yes	● ● ● ●
001-100407	Sep 14, 2010	4	KUCZERA v. POLAND Parties: KUCZERA, POLAND	Poland	6	Court Fourth Section	Yes	● ● ● ●
001-100409	Sep 14, 2010	3	SUBICKA v. POLAND Parties: SUBICKA, POLAND	Poland	6	Court Fourth Section	Yes	● ● ● ●
001-100535	Sep 21, 2010	4	PASTUSZENIA v. POLAND Parties: PASTUSZENIA, POLAND	Poland	6	Court Fourth Section Committee	Yes	● ● ● ●
001-100540	Sep 21, 2010	4	DEEG v. POLAND Parties: DEEG, POLAND	Poland	6	Court Fourth Section Committee	Yes	● ● ● ●
001-100543	Sep 21, 2010	4	MAZGAJ v. POLAND Parties: MAZGAJ, POLAND	Poland	8	Court Fourth Section	Yes	● ● ● ●
001-100657	Sep 30, 2010	3	VLADIMIR POLISHCHUK AND SVETLA… v. UKRAINE Parties: VLADIMIR POLISHCHUK AND SVETLA…, UKRAINE	Ukraine	8	Court Fith Section	Yes	● ● ● ●
001-100887	Oct 05, 2010	4	SZPARAG v. POLAND Parties: SZPARAG, POLAND	Poland	6	Court Fourth Section Committee	Yes	● ● ● ●

Figure A.1: Listing of cases with real time sorting and search.
Each row provides the judgment date, the parties, the countrie(s), the main conclusion, and if the judgment file and processed documents are available.



PETUKHOV v. UKRAINE

Case Information

ECLI	ECLI:CE:ECHR:2010:1021JUD004337402
App. No.	43374/02
Parties	Petukhov, Ukraine
Respondent	Ukraine
Conclusion Summary [details]	3 5 6 13

Cited App.

43374/02
54825/00
40774/02
4048/03
72286/01
18660/03
65550/01
75522/01
29971/04
30210/96
6293/04
30983/02
46468/06
30628/02
26853/04
35421/05
9852/03
13413/04

Judgment document

Content

```
PROCEDURE
THE FACTS
    I. THE CIRCUMSTANCES OF THE CASE
        A. Criminal proceedings against the applicant
        B. Conditions of the applicant's detention
    II. RELEVANT DOMESTIC LAW
        A. Constitution of , 1996
        B. Code of Criminal Procedure
        C. Relevant international materials concerning medical assistance in detention
            1. European Committee for the Prevention of Torture and Inhuman and Degrading Treatment and Punishment ("the CPT")
THE LAW
    I. ALLEGED VIOLATION OF ARTICLES 3 AND 13 OF THE CONVENTION
        A. Admissibility
        B. Merits
        (a) The Government's submissions
        (b) The applicant's submissions
        (c) The Court's assessment
            2. Article 13 of the Convention
    II. ALLEGED VIOLATION OF ARTICLE 5 §§ 1 AND 3 OF THE CONVENTION
        A. Admissibility
        B. Merits
            1. Article 5 § 1 of the Convention
        (a) Periods between 29 November 2001 – 30 October 2002 and 1 November 2002 – 19 June 2003
        (b) Period of 19 – 24 May 2004
        (a) Parties' observations
        (b) General principles
```

Figure A.2: Partial display of a case.

In particular, the cited applications are extracted. The table of content shows the tree nature of the judgment document.

A.2 REST API

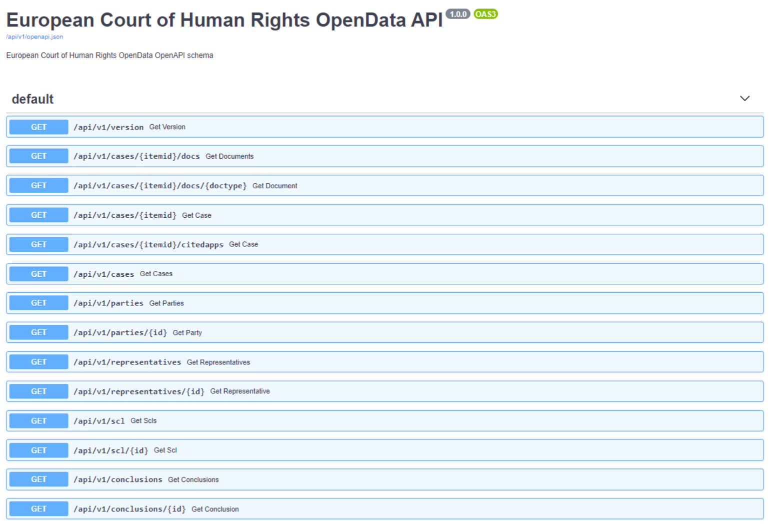
The standardized REST API provides a convenient programmatic way to retrieve data from *ECHR-DB*. The API allows to download specific documents, access cases, parties, representatives, citations and conclusions. The documentation is available at https://echr-opendata.eu/api/v1/docs and allows to try any request. Figure A.3 shows the documentation interface and most of the available endpoints.

Two examples of manual API calls are provided below. The first one returns the version of *ECHR-DB* used by the API, and the second returns the list of documents available for a specific case and how to download them.

```
curl -X GET "https://echr-opendata.eu/api/v1/version" -H "accept: application/json"

"2.0.0"

curl -X GET "https://echr-opendata.eu/api/v1/cases/001-100018/docs" -H "accept: application/json"

{
  "judgment": {
    "available": true,
    "uri": "/api/v1/cases/001-100018/docs/judgment"
  },
  "bow": {
    "available": true,
    "uri": "/api/v1/cases/001-100018/docs/bow"
  },
  "tfidf": {
    "available": true,
    "uri": "/api/v1/cases/001-100018/docs/tfidf"
  },
  "parsed_judgment": {
    "available": true,
    "uri": "/api/v1/cases/001-100018/docs/parsed_judgment"
  }
}
```

Therefore, to download the Bag-of-Words representation of case 001-100018, it is enough to call:

```
curl -X GET "https://echr-opendata.eu/api/v1/cases/001-100018/docs/bow"
```



European Court of Human Rights OpenData API 1.0.0 OAS3

/api/v1/openapi.json

European Court of Human Rights OpenData OpenAPI schema

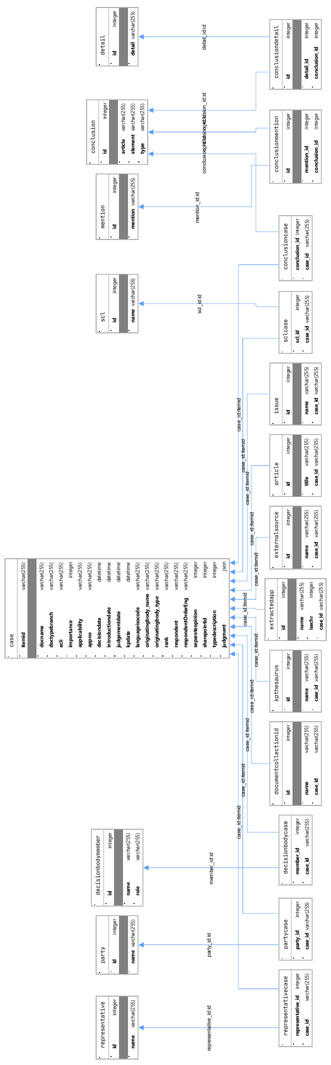
default

```
GET /api/v1/version Get Version
GET /api/v1/cases/{itemid}/docs Get Documents
GET /api/v1/cases/{itemid}/docs/{doctype} Get Document
GET /api/v1/cases/{itemid} Get Case
GET /api/v1/cases/{itemid}/citedapps Get Case
GET /api/v1/cases Get Cases
GET /api/v1/parties Get Parties
GET /api/v1/parties/{id} Get Party
GET /api/v1/representatives Get Representatives
GET /api/v1/representatives/{id} Get Representative
GET /api/v1/scl Get Scls
GET /api/v1/scl/{id} Get Scl
GET /api/v1/conclusions Get Conclusions
GET /api/v1/conclusions/{id} Get Conclusion
```

Figure A.3: Documentation of the REST API. Each endpoint can be tested online.

Relational Schema

This Appendix contains the relation schema of the SQL database described in Chapter 5.



APPENDIX C

HCBR: additional results

This appendix contains additional detailed results about the experiments performed in Section 6.3. Additionally, Appendix C.5 discuss the heuristic and learning curves obtained during these experiments. Finally, Appendix C.6 contains the hyperparameters for the genetic algorithm used in Section 6.5.

C.1 Average confusion matrix

Table C.1: Average confusion matrix obtained with a 10-fold cross-validation.

		TP	FN	FP	TN
`adult`	without tuning	2182.4	295.3	288.5	488.8
	with tuning	2226.6	245.3	311.4	472.7
`breast`	without tuning	23.0	1.4	0.7	43.9
	with tuning	23.5	1.1	0.4	44.0
`heart`	without tuning	12.4	1.8	1.9	9.9
	with tuning	13.5	1.0	1.4	10.1
`mushrooms`	without tuning	390.6	0.0	0.0	420.4
`phishing`	without tuning	595.4	23.8	19.8	465.0
	with tuning	599.1	19.2	15.9	468.9
`skin`	without tuning	4886.3	132.4	199.4	19286.9
	with tuning	4888.3	130.4	194.1	19292.2
`splice`	without tuning	155.7	9.1	8.5	142.7
	with tuning	156.2	8.6	6.9	144.3

C.2 Comparison of HCBR w.r.t. MCC

Table C.2: Comparison of HCBR with several methods (Scikit-Learn implementation) w.r.t. MCC.

Dataset	Method	MCC	#
adult	**HCBR**	0.5146	3
	AdaBoost	0.5455	1
	k-NN	0.4785	7
	Linear SVM	0.4918	5
	RBF SVM	0.5065	4
	Decision Tree	0.4821	6
	Rand. Forest	0.3776	8
	Neural Net	0.5349	2
	Naive Bayes	0.2493	9
	QDA	0.4785	7
breast	**HCBR**	0.9222	3
	AdaBoost	0.9023	6
	k-NN	0.9163	4
	Linear SVM	0.9126	5
	RBF SVM	0.8829	8
	Decision Tree	0.8760	9
	Rand. Forest	0.9296	1
	Neural Net	0.9280	2
	Naive Bayes	0.8991	7
	QDA	0.8616	10
heart	**HCBR**	0.7082	1
	AdaBoost	0.5972	6
	k-NN	0.5879	7
	Linear SVM	0.6849	4
	RBF SVM	0.6287	5
	Decision Tree	0.5763	8
	Rand. Forest	0.5703	9
	Neural Net	0.6995	2
	Naive Bayes	0.6932	3
	QDA	0.4500	10
mushrooms	**HCBR**	0.9995	2
	AdaBoost	1.0000	1
	k-NN	0.9993	3
	Linear SVM	1.0000	1
	RBF SVM	0.9990	5
	Decision Tree	0.9991	4
	Rand. Forest	0.8840	7
	Neural Net	1.0000	1
	Naive Bayes	0.9767	6
	QDA	1.0000	1
phishing	**HCBR**	0.9191	1
	AdaBoost	0.8637	6
	k-NN	0.9138	4
	Linear SVM	0.8740	5
	RBF SVM	0.9286	2
	Decision Tree	0.8585	7
	Rand. Forest	0.7582	8
	Neural Net	0.9448	1
	Naive Bayes	0.5292	10
	QDA	0.5872	9
skin	**HCBR**	0.9551	4
	AdaBoost	0.8552	8
	k-NN	0.9982	1
	Linear SVM	0.8090	9
	RBF SVM	0.9950	3
	Decision Tree	0.9544	5
	Rand. Forest	0.9539	6
	Neural Net	0.9967	2
	Naive Bayes	0.7600	10
	QDA	0.9483	7
splice	**HCBR**	0.8857	2
	AdaBoost	0.8801	3
	k-NN	0.6072	9
	Linear SVM	0.7282	8
	RBF SVM	0.8461	4
	Decision Tree	0.8998	1
	Rand. Forest	0.5925	10
	Neural Net	0.8390	5
	Naive Bayes	0.7595	7
	QDA	0.8251	6

C.3 Comparison of HCBR w.r.t. accuracy

Table C.3: Comparison of HCBR with several methods (Scikit-Learn implementation) with w.r.t. accuracy.

Dataset	Method	Acc.	#
adult	**HCBR**	0.8232	5
	AdaBoost	0.8444	1
	k-NN	0.8156	7
	Linear SVM	0.8274	4
	RBF SVM	0.8327	3
	Decision Tree	0.7918	8
	Rand. Forest	0.8223	6
	Neural Net	0.8378	2
	Naive Bayes	0.4675	10
	QDA	0.7528	9
breast	**HCBR**	0.9833	1
	AdaBoost	0.9563	4
	k-NN	0.9614	3
	Linear SVM	0.9614	3
	RBF SVM	0.9457	7
	Decision Tree	0.9429	9
	Rand. Forest	0.9543	5
	Neural Net	0.9671	2
	Naive Bayes	0.9533	6
	QDA	0.9430	8
heart	**HCBR**	0.8538	1
	AdaBoost	0.8037	6
	k-NN	0.7926	7
	Linear SVM	0.8444	3
	RBF SVM	0.8148	5
	Decision Tree	0.7556	9
	Rand. Forest	0.7741	8
	Neural Net	0.8519	2
	Naive Bayes	0.8444	3
	QDA	0.8185	4
mushrooms	**HCBR**	0.9998	2
	AdaBoost	1.0000	1
	k-NN	0.9996	3
	Linear SVM	1.0000	1
	RBF SVM	0.9995	4
	Decision Tree	0.9996	3
	Rand. Forest	0.9582	6
	Neural Net	1.0000	1
	Naive Bayes	0.9882	5
	QDA	1.0000	1

Dataset	Method	Acc.	#
phishing	**HCBR**	0.9645	3
	AdaBoost	0.9477	8
	k-NN	0.9505	7
	Linear SVM	0.9532	6
	RBF SVM	0.9550	5
	Decision Tree	0.9625	4
	Rand. Forest	0.9738	1
	Neural Net	0.9726	2
	Naive Bayes	0.7062	10
	QDA	0.7656	9
skin	**HCBR**	0.9847	4
	AdaBoost	0.9399	7
	k-NN	0.9994	1
	Linear SVM	0.9297	8
	RBF SVM	0.9984	3
	Decision Tree	0.9456	5
	Rand. Forest	0.9415	6
	Neural Net	0.9989	2
	Naive Bayes	0.8802	9
	QDA	0.8978	10
splice	**HCBR**	0.9430	3
	AdaBoost	0.9528	2
	k-NN	0.7843	10
	Linear SVM	0.8645	9
	RBF SVM	0.9230	7
	Decision Tree	0.9415	4
	Rand. Forest	0.9399	5
	Neural Net	0.9195	8
	Naive Bayes	0.9245	6
	QDA	0.9838	1

C.4 MCC depending on the training set size

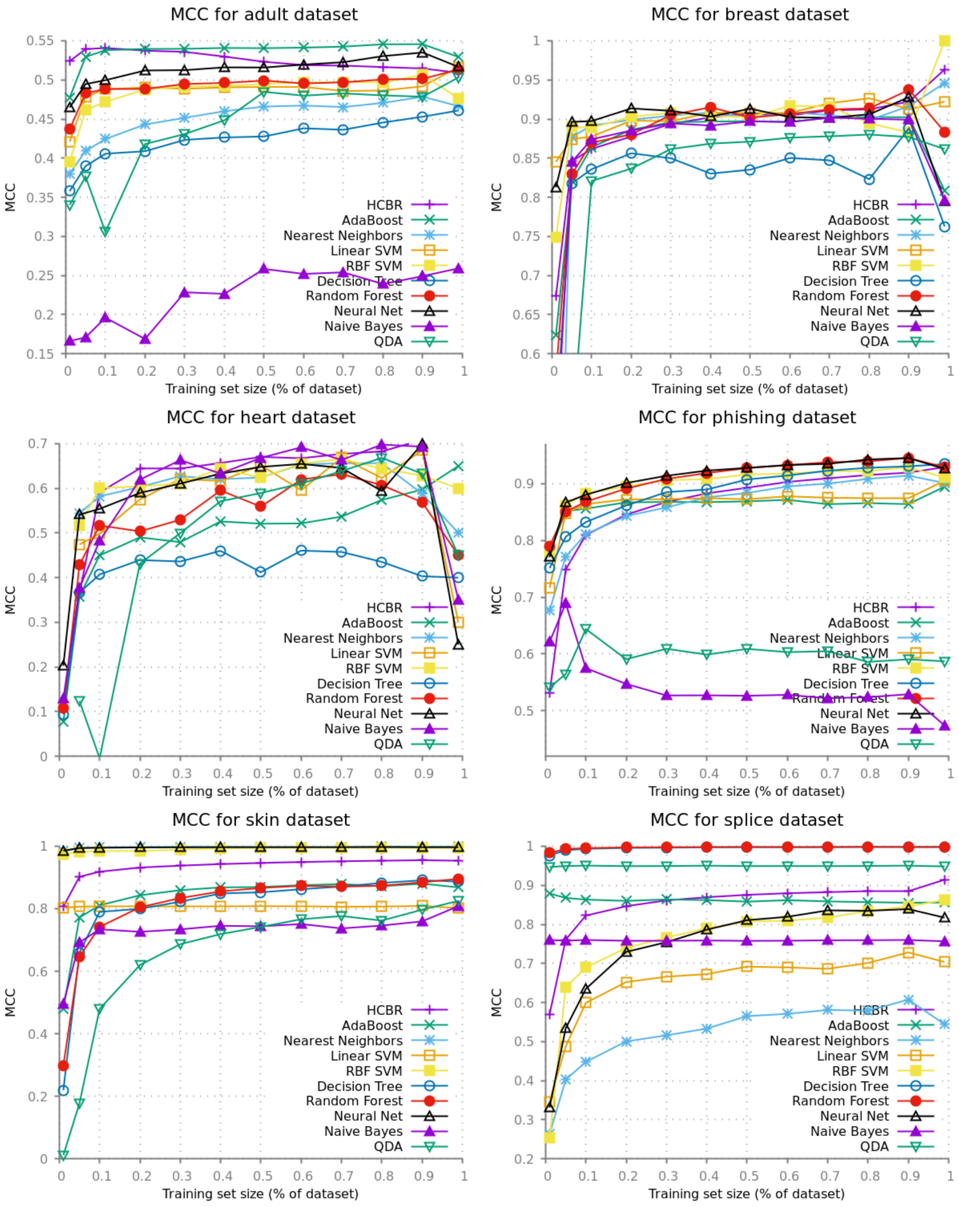


Figure C.1: Matthew Correlation Coefficient depending on the training set size

C.5 Discussion on the learning curves

C.5.1 Heuristic

In almost all datasets, in particular `phishing` and `skin`, many input vectors are the same, but with different output (e.g. for `skin` it's 583 cases i.e. 5.27% of the dataset). By definition, there is no way to distinguish between those vectors, so the best we can do is to assign the class with the highest prevalence.

In most cases the model choses itself the class with the highest prevalence among those redundant input vectors or if it does not, it has very little impact (because the prevalence is close to 0.5 and the size of the redundant vector set is small in comparison to the dataset size).

The heuristic consists in estimating the prevalence of the set of redundant vectors in the training set, and then in bypassing the model prediction with the class associated to this prevalence. For instance, the overall prevalence of phishing is 0.5562 but the prevalence of the redundant vectors is 0.9674. About 5.27% of the test set should consists of cases already in the casebase, among which 0.9674 are of class 1. It represents the insurance of a 5.27 x 0.9674 = 5.20% of the training set correctly labeled.

The reason why the heuristic works in this case is because the prevalence of the redundant vectors set is higher than the accuracy obtained by the model (about 92.5% versus 96%). The gain corresponds exactly to this 5.20% (because without the heuristic, with the grain used by the experiment, the model uses the wrong label).

This heuristic is independent of HCBR as it could be applied to any margin-based discrimative methods (e.g. SVM also cannot discriminate between redundant points with different labels).

C.5.2 Learning curve

The learning curves for `phishing` and `skin` are surprising for two reasons. First, the test accuracy is significantly higher than the training accuracy as depicted by Figure A6.1 and A6.2. The prediction phase on those two datasets uses the heuristic described above. For both datasets, the difference in accuracy between the two curves is much higher than the possible gain due to the heuristic. The learning curves for the same experiment without the heuristic are displayed on Figures A6.3 and A6.4 and they seem to explain perfectly the phenomena. Notice that this is specific to those two datasets that contains several redundant points with the same output. For instance, the heuristic is activated also on splice but as the number of redundant elements is very low w.r.t. the dataset size, the impact on the accuracy is not significant.

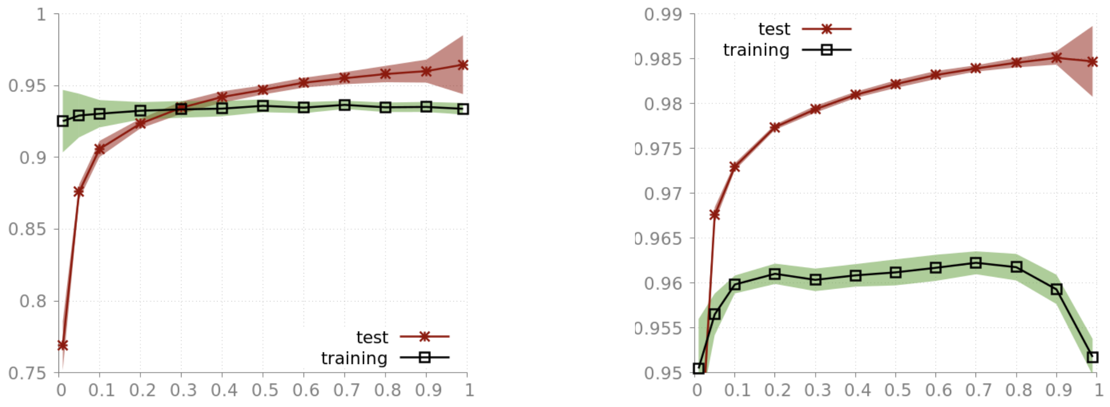
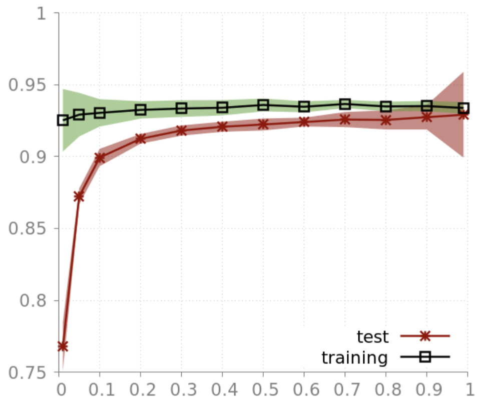


Figure C.2: Learning curves on `phishing` and `skin` datasets.
The accuracy on the test set is higher than on the training set. On the training set, the accuracy starts by increasing and remains globally stable (with a drop for `skin`).



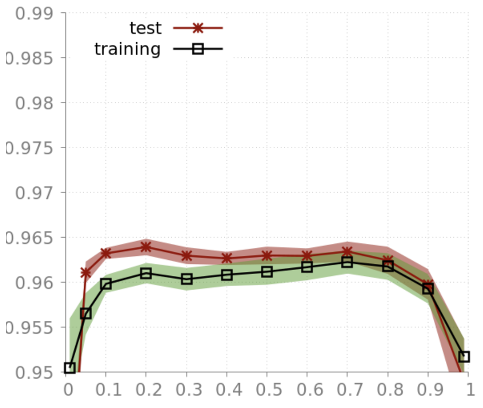


Figure C.3: Learning curves on phishing and skin datasets without the heuristic. Compared to Figure C.2, the learning curve behave as expected in theory.

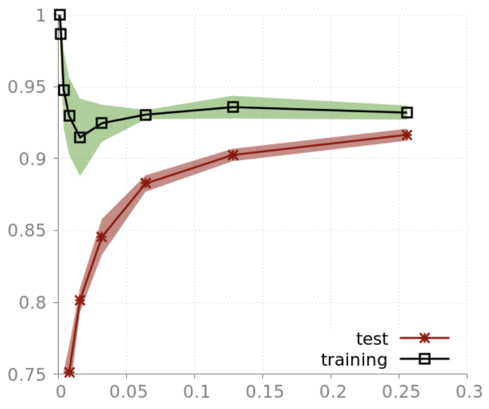


Figure C.4: Learning curves on phishing for very small training set sizes.

C.6 Hyperparameters for the genetic algorithm

Table C.4: Hyperparameters for the genetic algorithm.

Dataset	Generations	Mutation σ factor
adult	200	10000
breasts	200	10
heart	200	10
phishing	100	1000
skin	100	1000
splice	200	100

APPENDIX D

AutoML experiments

This Appendix contains the description of the search space used in the experiments of Chapter 8.

Table D.1: Pipeline search space.

	$\#\lambda$	$\|\Lambda\|$	impl.
Rebalance			
No operator	0	0	-
Near Miss	1	3	imblearn
Condensed Nearest Neighbour	1	3	imblearn
SMOTE	1	3	imblearn
Normalize			
No operator	0	0	-
Standard Scaler	2	4	sklearn
Power Transform	0	0	sklearn
MinMax Scaler	0	0	sklearn
Robust Scaler	3	12	sklearn
Features			
No operator	0	0	-
PCA	1	4	sklearn
SelectKBest (F-score)	1	4	sklearn
PCA $\cup$ SelectKBest	2	16	sklearn

The column $\#\lambda$ is the number of parameters while $|\Lambda|$ is the total number of values in the operator configuration space. The column *impl.* indicates the implementation of the operator (scikit-learn or imbalanced-learn).

List of Figures

List of Tables

Bibliography

[Aamodt 1994] A. Aamodt and E. Plaza. *Case-based reasoning: Foundational issues, methodological variations, and system approaches.* AI Communications, vol. 7, no. 1, pages 39–59, 1994. (Cited on pages 26 and 175.)

[Akay 2009] M. F. Akay. *Support vector machines combined with feature selection for breast cancer diagnosis.* Expert Systems with Applications, vol. 36, no. 2, pages 3240–3247, 2009. (Cited on pages 101 and 102.)

[Akrour 2014] R. Akrour, M. Schoenauer, M. Sebag and J.-C. Souplet. *Programming by Feedback.* In International Conference on Machine Learning (ICML), number 32 de JMLR Proceedings, pages 1503–1511, Pékin, China, 2014. (Cited on page 27.)

[Aletras 2016] N. Aletras, D. Tsarapatsanis, D. Preoţiuc-Pietro and V. Lampos. *Predicting judicial decisions of the European Court of Human Rights: a Natural Language Processing perspective.* PeerJ Computer Science, vol. 2, page e93, 2016. (Cited on pages vi, x, 18, 19, 31, 55, 66, 107, 108 and 109.)

[Aleven 1997] V. Aleven and K. D. Ashley. *Evaluating a Learning Environment for Case-based Argumentation Skills.* In International Conference on Artificial Intelligence and Law (ICAIL), pages 170–179, New York, NY, USA, 1997. ACM. (Cited on pages vi, 29 and 31.)

[Ali 2017] S. M. F. Ali and Wrembel R. *From conceptual design to performance optimization of ETL workflows: current state of research and open problems.* Very Large Data Base Endowment (VLDB), vol. 26, no. 6, pages 777–801, 2017. (Cited on page 53.)

[Allaire 2007] G. Allaire. Numerical analysis and optimization : an introduction to mathematical modelling and numerical simulation. Oxford University Press, Oxford, 2007. (Cited on page 124.)

[Amgoud 2005] L. Amgoud. *A Unified Setting for Inference and Decision: An Argumentation-based Approach.* In Uncertainty in Artificial Intelligence (UAI), pages 26–33, Arlington, Virginia, United States, 2005. AUAI Press. (Cited on page 30.)

[Andrew 2004] M. D. Andrew, K. M. Quinn and L. Epstein. *Median Justice on the United States Supreme Court.* North Carolina Law Review, vol. 83, page 1275, 2004. (Cited on pages 23 and 31.)

[Antoniou 2018] G. Antoniou, G. Baryannis, S. Batsakis, G. Governatori, L. Robaldo, G. Siragusa and I. Tachmazidis. *Legal reasoning and big data: opportunities and challenges.* In Workshop on MIning and REasoning with Legal texts (MIREL), 2018. (Cited on page 50.)

[Ashley 1988] K. D. Ashley. Arguing by analogy in law: A case-based model, pages 205–224. Springer Netherlands, Dordrecht, 1988. (Cited on page 29.)

[Ashley 2002] K. D. Ashley. *An AI model of case-based legal argument from a jurisprudential viewpoint.* Artificial Intelligence and Law, vol. 10, no. 1, pages 163–218, 2002. (Cited on page 29.)

[Athakravi 2015] D. Athakravi, K. Satoh, K. Law, K. Broda and A. Russo. *Automated Inference of Rules with Exception from Past Legal Cases Using ASP.* In Logic Programming and Nonmonotonic Reasoning, pages 83–96. Springer Nature, 2015. (Cited on pages 30 and 31.)

[Baroni 2015] P. Baroni, M. Romano, F. Toni, M. Aurisicchio and G. Bertanza. *Automatic evaluation of design alternatives with quantitative argumentation.* Argument & Computation, vol. 6, no. 1, pages 24–49, 2015. (Cited on pages vi, 30, 31 and 32.)

[Basterrech 2015] S. Basterrech, A. Mesa and N.-T. Dinh. *Generalized Linear Models Applied for Skin Identification in Image Processing.* In Intelligent Data Analysis and Applications, pages 97–107. Springer, 2015. (Cited on page 102.)

[Batista 2002] G. Batista, M. C. Monard*et al. A Study of K-Nearest Neighbour as an Imputation Method.* HIS, vol. 87, no. 251-260, page 48, 2002. (Cited on page 40.)

[Becker 1973] L. C. Becker. *Analogy in Legal Reasoning.* Ethics, vol. 83, no. 3, pages 248–255, 1973. (Cited on page 27.)

[Bellet 2012] A. Bellet, A. Habrard and M. Sebban. *Similarity Learning for Provably Accurate Sparse Linear Classification.* In International Conference on Machine Learning (ICML), United Kingdom, 2012. (Cited on page 39.)

[Bellet 2013] A. Bellet, A. Habrard and M. Sebban. *A Survey on Metric Learning for Feature Vectors and Structured Data.* CoRR, vol. abs/1306.6709, 2013. (Cited on pages v, 38 and 119.)

[Bench-Capon] Trevor Bench-Capon. *The need for good old fashioned ai and law.* International Trends in Legal Informatics: A Festschrift for Erich Schweighofer. (Cited on page 33.)

[Bench-Capon 2002] T. J. M. Bench-Capon. *Representation of Case Law as an Argumentation Framework.* In International Conference on Legal Knowledge and Information Systems (JURIX), pages 103–112, 2002. (Cited on pages 30 and 31.)

[Bench-Capon 2003] T. J. M. Bench-Capon. *Try to See It My Way: Modelling Persuasion in Legal Discourse.* Artificial Intelligence and Law, vol. 11, no. 4, pages 271–287, 2003. (Cited on pages 30 and 31.)

[Berge 1984] C. Berge. Hypergraphs: combinatorics of finite sets, volume 45. Elsevier, 1984. (Cited on page 41.)

[Bergstra 2011] J. Bergstra, R. Bardenet, Y. Bengio and B. Kégl. *Algorithms for Hyper-parameter Optimization.* In Neural Information Processing Systems (NeurIPS), pages 2546–2554, 2011. (Cited on page 44.)

[Bergstra 2012] J. Bergstra and Y. Bengio. *Random search for hyper-parameter optimization.* Journal of Machine Learning Research (JMLR), vol. 13, no. Feb, pages 281–305, 2012. (Cited on page 44.)

[Bergstra 2015] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins and D. Cox. *Hyperopt: a python library for model selection and hyperparameter optimization.* Computer Science & Discovery, vol. 8, no. 1, page 014008, 2015. (Cited on pages v, 44 and 145.)

[Berline 2003] N. Berline, E. Getzler and M. Vergne. Heat kernels and dirac operators. Springer Science & Business Media, 2003. (Cited on page 128.)

[Bernard 2006] M. Bernard, A. Habrard and M. Sebban. *Learning stochastic tree edit distance.* In European Conference on Machine Learning (ECML), pages 42–53. Springer, 2006. (Cited on page 119.)

[Bhatt 2009] R. B. Bhatt, G. Sharma, A. Dhall and S. Chaudhury. *Efficient Skin Region Segmentation Using Low Complexity Fuzzy Decision Tree Model.* In IEEE India Conference, pages 1–4, 2009. (Cited on pages 101 and 102.)

[Bilalli 2017] B. Bilalli, A. Abelló and T. Aluja-Banet. *On the Predictive Power of Meta-features in OpenML.* International Journal of Applied Mathematics and Computer Science, vol. 27, no. 4, pages 697–712, 2017. (Cited on pages v, 40, 45 and 155.)

[Bilalli 2018] B. Bilalli, A. Abelló, T. Aluja-Banet and R. Wrembel. *Intelligent assistance for data pre-processing.* Computer Standards & Interfaces, pages 101–109, 2018. (Cited on page 155.)

[Bischl 2017] B. Bischl, G. Casalicchio, M. Feurer, F. Hutter, M. Lang, R. G Mantovani, J. N. van Rijn and J. Vanschoren. *OpenML benchmarking suites and the OpenML100.* arXiv preprint arXiv:1708.03731, 2017. (Cited on page 155.)

[Blazy 2011] R. Blazy, B. Chopard, E. Langlais and Y. Ziane. *Personal Bankruptcy Law, Fresh Starts, and Judicial Practice.* Technical report 2011-15, University of Paris West - Nanterre la Défense, EconomiX, 2011. (Cited on page 14.)

[Blazy 2012] R. Blazy, Chopard B., E. Langlais and Y. Ziane. *L'effacement des dettes des particuliers surendettés : Une étude empirique des décisions judiciaires.* Technical report 2012-10, University of Paris West - Nanterre la Défense, EconomiX, 2012. (Cited on page 14.)

[Blazy 2013a] R. Blazy, B. Chopard and N Nigam. *Building legal indexes to explain recovery rates: An analysis of the French and English bankruptcy codes.* Journal of Banking & Finance, vol. 37, no. 6, pages 1936–1959, 2013. (Cited on page 14.)

[Blazy 2013b] R. Blazy, B. Deffains, G. Umbhauer and L. Weill. *Severe or gentle bankruptcy law: Which impact on investing and financing decisions?* Economic Modelling, vol. 34, pages 129–144, 2013. (Cited on page 14.)

[Borwein 2013] Jonathan M Borwein, Richard E Crandall*et al.* *Closed forms: what they are and why we care.* Notices of the AMS, vol. 60, no. 1, pages 50–65, 2013. (Cited on page 123.)

[Boucheron 2005] S. Boucheron, O. Bousquet and G. Lugosi. *Theory of classification: A survey of some recent advances.* ESAIM: Probability and Statistics, vol. 9, pages 323–375, 2005. (Cited on page 36.)

[Branting 1991] L. K. Branting. *Building Explanations from Rules and Structured Cases.* International Journal of Man-Machine Studies, vol. 34, no. 6, pages 797–837, 1991. (Cited on page 28.)

[Breiman 2001] L. Breiman. *Random forests.* Machine Learning, vol. 45, no. 1, pages 5–32, 2001. (Cited on page 38.)

[Breiman 2017] L. Breiman. Classification and regression trees. Routledge, 2017. (Cited on page 38.)

[Brezis 1999] H. Brezis, P. Ciarlet and J. L. Lions. Analyse fonctionnelle : théorie et applications, volume 91. Dunod Paris, 1999. (Cited on page 125.)

[Brezis 2010] H. Brezis. Functional analysis, sobolev spaces and partial differential equations. Springer Science & Business Media, 2010. (Cited on page 124.)

[Brodersen 2010] K. H. Brodersen, C. S. Ong, K. E. Stephan and J. M. Buhmann. *The balanced accuracy and its posterior distribution.* In International Conference Pattern Recognition, pages 3121–3124. IEEE, 2010. (Cited on page 70.)

[Brüninghaus 2005] S. Brüninghaus and K. D. Ashley. *Generating Legal Arguments and Predictions from Case Texts.* In International Conference on Artificial Intelligence and Law (ICAIL), page 65–74, New York, NY, USA, 2005. Association for Computing Machinery. (Cited on page 28.)

[Calabresi 1970] G. Calabresi. The cost of accidents: A legal and economic analysis. Yale University Press, 1970. (Cited on page 13.)

[Çatak 2017] F. Ö. Çatak. *Classification with boosting of extreme learning machine over arbitrarily partitioned data.* Soft Computing, vol. 21, no. 9, pages 2269–2281, 2017. (Cited on pages 101 and 102.)

[Cayrol 2005] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. *On the acceptability of arguments in bipolar argumentation frameworks.* In European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU), pages 378–389. Springer Berlin Heidelberg, 2005. (Cited on page 30.)

[Cazzolato 2013] M. T. Cazzolato and M. X. Ribeiro. *A statistical decision tree algorithm for medical data stream mining.* In IEEE Symposium on Computer-Based Medical Systems, pages 389–392, 2013. (Cited on pages 101 and 102.)

[Chalkidis 2019] I. Chalkidis I. Androutsopoulos and N. Aletras. *Neural Legal Judgment Prediction in English.* pages 4317–4323, 2019. (Cited on pages vi, 19, 50, 57 and 58.)

[Chang 2010] Y.-W. Chang, C.-J. Hsieh, K.-W. K. Chang, M. Ringgaard and C.-J. Lin. *Training and testing low-degree polynomial data mappings via linear SVM.* Journal of Machine Learning Research (JMLR), vol. 11, no. Apr, pages 1471–1490, 2010. (Cited on page 123.)

[Chechik 2010] G. Chechik, V. Sharma, U. Shalit and S. Bengio. *Large Scale Online Learning of Image Similarity Through Ranking.* Journal of Machine Learning Research (JMLR), vol. 11, pages 1109–1135, 2010. (Cited on page 39.)

[Chen 2011] H.-L. Chen, B. Yang, J. Liu and D.-Y. Liu. *A support vector machine classifier with rough set-based feature selection for breast cancer diagnosis.* Expert Systems with Applications, vol. 38, no. 7, pages 9014–9022, 2011. (Cited on pages 101 and 102.)

[Chen 2018] B. Chen, H. Wu, W. Mo, I. Chattopadhyay and H. Lipson. *Autostacker: A compositional evolutionary learning system.* In Genetic and Evolutionary Computation Conference (GECCO), pages 402–409, 2018. (Cited on page 45.)

[Chessell 2014] M. Chessell, F. Scheepers, N. Nguyen, R. van Kessel and R. van der Starre. *Governing and managing big data for analytics and decision makers.* IBM Redguides for Business Leaders, 2014. (Cited on pages iii, 3, 40 and 138.)

[Chicco 2017] D. Chicco. *Ten quick tips for machine learning in computational biology.* BioData Mining, vol. 10, no. 1, 2017. (Cited on pages 61 and 100.)

[Chopra 2005] S. Chopra, R. Hadsell and Y. LeCun. *Learning a similarity metric discriminatively, with application to face verification.* In Conference on Computer Vision and Pattern Recognition, volume 1, pages 539–546. IEEE, 2005. (Cited on page 39.)

[Chowdhury 2019] A. Chowdhury, M. Magdon-Ismail and B. Yener. *Quantifying error contributions of computational steps, algorithms and hyperparameter choices in image classification pipelines.* CoRR, vol. abs/1903.02521, 2019. (Cited on page 155.)

[Cichowski 2017] R. Cichowski and Chrun. *European Court of Human Rights Database, Version 1.0 Release 2017.* 2017. `http://depts.washington.edu/echrdb/`. (Cited on page 50.)

[Clinton 2004] J. Clinton, S. Jackman and D. Rivers. *The statistical analysis of roll call data.* American Political Science Review, pages 355–370, 2004. (Cited on page 22.)

[Coase 1960] R. Coase. *The Problem of Social Cost.* The Journal of Law and Economics, vol. 3, 1960. (Cited on page 13.)

[Cog 2019] *Data Engineering, Preparation, and Labeling for AI 2019.* Technical report, Cognilytica Research, 2019. (Cited on pages iii, 3 and 40.)

[Conrad 2018] L. Karl Conrad Jack G.and Branting. *Introduction to the special issue on legal text analytics.* Artificial Intelligence and Law, vol. 26, no. 2, pages 99–102, 2018. (Cited on page 50.)

[Couronné 2018] R. Couronné, P. Probst and A.-L. Boulesteix. *Random forest versus logistic regression: a large-scale benchmark experiment.* BMC Bioinformatics, vol. 19, no. 1, page 270, 2018. (Cited on pages iii, 3, 38, 101 and 138.)

[Cox 1958] D. R. Cox. *The regression analysis of binary sequences.* Journal of the Royal Statistical Society, pages 215–242, 1958. (Cited on page 36.)

[Crammer 2006] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz and Y. Singer. *Online passive-aggressive algorithms.* Journal of Machine Learning Research (JMLR), vol. 7, no. Mar, pages 551–585, 2006. (Cited on page 39.)

[Crone 2006] S. F. Crone, S. Lessmann and R. Stahlbock. *The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing.* The European Journal of Operational Research (EJOR), vol. 173, no. 3, pages 781–800, 2006. (Cited on pages v, 40, 137 and 138.)

[Csáji 2001] B. C. Csáji. *Approximation with artificial neural networks.* Faculty of Sciences, Etvs Lornd University, Hungary, vol. 24, no. 48, page 7, 2001. (Cited on pages iv and 2.)

[Cucker 2002] F. Cucker and S. Smale. *On the mathematical foundations of learning.* Bulletin of the American mathematical society, vol. 39, no. 1, pages 1–49, 2002. (Cited on page 37.)

[Dalvi 2009] N. Dalvi, P. Bohannon and F. Sha. *Robust Web Extraction: An Approach Based on a Probabilistic Tree-edit Model.* In International Conference on Management of Data (SIGMOD), SIGMOD, pages 335–348, New York, NY, USA, 2009. ACM. (Cited on page 119.)

[Das 2001] S. Das. *Filters, Wrappers and a Boosting-Based Hybrid for Feature Selection.* In International Conference on Machine Learning (ICML), pages 74–81, 2001. (Cited on pages 101 and 102.)

[Dasu 2003] T. Dasu and T. Johnson. Exploratory data mining and data cleaning, volume 479. John Wiley & Sons, 2003. (Cited on pages 40 and 138.)

[Datta 2016] R. P. Datta and S. Saha. *Applying rule-based classification techniques to medical databases: an empirical study.* International Journal of Business Intelligence and Systems Engineering, vol. 1, no. 1, pages 32–48, 2016. (Cited on pages 101 and 102.)

[Davis 2007] J. V. Davis, B. Kulis, P. Jain, S. Sra and I. S. Dhillon. *Information-theoretic metric learning.* In International Conference on Machine Learning (ICML), pages 209–216. ACM, 2007. (Cited on pages v and 38.)

[De Mantaras 2005] L.p R. De Mantaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M. L. Maher, M. T. Cox, K. Forbus, M. Keane, A. Aamodt and I. Watson.

Retrieval, Reuse, Revision and Retention in Case-based Reasoning. The Knowledge Engineering Review, vol. 20, no. 3, pages 215–240, 2005. (Cited on page 27.)

[Deffains 2009] B. Deffains and E. Langlais. Analyse économique du droit: Principes, méthodes, résultats. Ouvertures économiques. De Boeck Supérieur, 2009. (Cited on page 13.)

[Dehez 2012] P. Dehez and S. Ferey. *How to share joint liability: a cooperative game approach.* Technical report, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2012. (Cited on page 14.)

[Delgado 2007] P. Delgado. *Survey of CaseBased Reasoning as Applied to the Legal Domain.* 2007. Unpublished. (Cited on page 28.)

[Domhan 2015] T. Domhan, J. T. Springenberg and F. Hutter. *Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves.* In International Conference on Machine Learning (ICML), 2015. (Cited on page 45.)

[Dung 1995] P. M. Dung. *On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games.* Artificial Intelligence, vol. 77, pages 321–357, 1995. (Cited on page 29.)

[Dung 2006] P. M. Dung, R. A. Kowalski and F. Toni. *Dialectic Proof Procedures for Assumption-based, Admissible Argumentation.* Artificial Intelligence, vol. 170, no. 2, pages 114–159, 2006. (Cited on pages vi, 30, 31 and 32.)

[Dung 2008] P. M. Dung and P. M. Thang. *Towards an argument-based model of legal doctrines in common law of contracts.* vol. 7, pages 111–126, 2008. (Cited on page 30.)

[Dung 2010] P. M. Dung and P. M. Thang. *Towards (Probabilistic) Argumentation for Jury-based Dispute Resolution.* In Conference on Computational Models of Argument (COMMA), pages 171–182, 2010. (Cited on page 30.)

[Eggensperger 2013] K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos and K. Leyton-Brown. *Towards an Empirical Foundation for Assessing Bayesian Optimization of Hyperparameters.* In Workshop on Bayesian Optimization in Theory and Practice (BayesOpt) @ Neural Information Processing Systems (NeurIPS), 2013. (Cited on page 145.)

[Elshawi 2019] R. Elshawi, M. Maher and S. Sakr. *Automated Machine Learning: State-of-The-Art and Open Challenges,* 2019. (Cited on pages 43, 45 and 137.)

[Fallahi 2011] A. Fallahi and S. Jafari. *An expert system for detection of breast cancer using data preprocessing and bayesian network.* International Journal of Advanced Science and Technology (IJAST), vol. 34, pages 65–70, 2011. (Cited on pages 101 and 102.)

[Fan 2014] X. Fan and F. Toni. *On Computing Explanations in Abstract Argumentation.* Frontiers in Artificial Intelligence and Applications, vol. 263, no. ECAI 2014, page 1005–1006, 2014. (Cited on page 30.)

[Ferreira 2009] J. C. Ferreira and V. A. Menegatto. *Eigenvalues of integral operators defined by smooth positive definite kernels.* Integral Equations and Operator Theory, vol. 64, no. 1, pages 61–81, 2009. (Cited on page 128.)

[Feurer 2015] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum and F. Hutter. *Efficient and Robust Automated Machine Learning.* In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett, editors, Advances in Neural Information Processing Systems (NeurIPS), pages 2962–2970. Curran Associates, Inc., 2015. (Cited on pages v, 44, 45 and 101.)

[Fortin 2012] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau and C. Gagné. *DEAP: Evolutionary Algorithms Made Easy.* Journal of Machine Learning Research (JMLR), vol. 13, pages 2171–2175, 2012. (Cited on page 112.)

[Frazier 2018] P. I. Frazier. *A tutorial on Bayesian optimization.* arXiv preprint arXiv:1807.02811, 2018. (Cited on page 44.)

[Friedman 1997] N. Friedman, D. Geiger and M. Goldszmidt. *Bayesian network classifiers.* Machine Learning, vol. 29, no. 2-3, pages 131–163, 1997. (Cited on page 38.)

[Friedman 2001] J. H. Friedman. *Greedy function approximation: a gradient boosting machine.* Annals of Statistics, pages 1189–1232, 2001. (Cited on page 38.)

[Frydman 2005] B. Frydman. Le sens des lois: histoire de l'interprétation et de la raison juridique. Collection Penser le droit. Bruylant, 2005. (Cited on page 14.)

[Fuchs 2014] B. Fuchs, J. Lieber, A. Mille and A. Napoli. *Differential adaptation: An operational approach to adaptation for solving numerical problems with CBR.* Knowledge-Based Systems, vol. 68, pages 103–114, 2014. (Cited on page 27.)

[Furche 2016] T. Furche, G. Gottlob, L. Libkin, G. Orsi and N. Paton. *Data Wrangling for Big Data: Challenges and Opportunities*, 2016. (Cited on page 40.)

[Gabel 2004] Thomas Gabel and Armin Stahl. *Exploiting Background Knowledge when Learning Similarity Measures.* volume 3155, pages 169–183, 2004. (Cited on page 27.)

[GarcíA 2013] A. J. GarcíA, C. I. ChesñEvar, N. D. Rotstein and G. R. Simari. *Formalizing Dialectical Explanation Support for Argument-based Reasoning in Knowledge-based Systems.* Expert Systems with Applications, vol. 40, no. 8, pages 3233–3247, 2013. (Cited on page 30.)

[Geurts 2006] P. Geurts, D. Ernst and L. Wehenkel. *Extremely Randomized Trees.* Machine Learning, vol. 63, no. 1, pages 3–42, 2006. (Cited on pages 18 and 38.)

[Grattan-Guinness 2005] I. Grattan-Guinness. *Joseph Fourier, Théorie analytique de la chaleur (1822).* In Landmark Writings in Western Mathematics 1640-1940, pages 354–365. Elsevier, 2005. (Cited on page 124.)

[Guimerà 2011] R. Guimerà and M. Sales-Pardo. *Justice Blocks and Predictability of U.S. Supreme Court Votes.* PLoS ONE, vol. 6, no. 11, page e27188, 2011. (Cited on page vi.)

[Guimerà 2011] Roger Guimerà and Marta Sales-Pardo. *Justice Blocks and Predictability of U.S. Supreme Court Votes.* PLOS ONE, vol. 6, no. 11, pages 1–8, 2011. (Cited on pages vi, 16, 18, 19, 21 and 31.)

[Haas 2005] L. M. Haas, M. A. Hernández, H. Ho, L. Popa and M. Roth. *Clio grows up: from research prototype to industrial tool.* In International Conference on Management of Data (SIGMOD), pages 805–810. ACM, 2005. (Cited on page 40.)

[Habermas 1988] J. Habermas. *Law and morality.* The Tanner Lectures on human values, vol. 8, pages 217–279, 1988. (Cited on page 32.)

[Hadi 2017] W. Hadi, G. Issa and A. Ishtaiwi. *ACPRISM: Associative classification based on PRISM algorithm.* Information Sciences, vol. 417, no. Supplement C, pages 287–300, 2017. (Cited on pages 101 and 102.)

[Hart 2012] H. L. A. Hart, H. L. A. Hart and L. Green. The concept of law. Oxford University Press, 2012. (Cited on page 14.)

[He 2009] H. He and E. A. Garcia. *Learning from Imbalanced Data.* IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9, pages 1263–1284, 2009. (Cited on page 103.)

[Hinton 2002] G. Hinton. *Training products of experts by minimizing contrastive divergence.* Neural Computation, vol. 14, no. 8, pages 1771–1800, 2002. (Cited on page 24.)

[Ho 1995] T. K. Ho. *Random decision forests.* In International Conference on Document Analysis and Recognition (ICDAR), volume 1, pages 278–282. IEEE, 1995. (Cited on page 38.)

[Hofmann 2008] T. Hofmann, B. Schölkopf and A. J. Smola. *Kernel methods in machine learning.* Annals of Statistics, pages 1171–1220, 2008. (Cited on page 37.)

[Huang 2010] Y. Huang, Q. Liu, S. Zhang and D. N. Metaxas. *Image retrieval via probabilistic hypergraph ranking.* In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 3376–3383, 2010. (Cited on page 41.)

[Hüllermeier 2007] E. Hüllermeier. Case-based approximate reasoning. Theory and Decision Library B. Springer Netherlands, 2007. (Cited on page 27.)

[Hume 1739] D. Hume. A treatise of human nature. Oxford University Press, 1739. (Cited on page 13.)

[Hutter 2011] F. Hutter, H. H. Hoos and K. Leyton-Brown. *Sequential Model-based Optimization for General Algorithm Configuration.* In International Conference on Learning and Intelligent Optimization (LION), pages 507–523, Berlin, Heidelberg, 2011. Springer-Verlag. (Cited on pages v and 44.)

[Hutter 2019] F. Hutter, L. Kotthoff and J. Vanschoren. *Automatic machine learning: methods, systems, challenges.* Challenges in Machine Learning, 2019. (Cited on pages 43 and 137.)

[Islam 2016] M. R. Islam, K. S. M. T. Hossain, S. Krishnan and N. Ramakrishnan. *Inferring Multi-dimensional Ideal Points for US Supreme Court Justices.* In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16, pages 4–12. AAAI Press, 2016. (Cited on pages 19, 25 and 31.)

[Jamieson 2016] K. Jamieson and A. Talwalkar. *Non-stochastic best arm identification and hyperparameter optimization.* In Artificial Intelligence and Statistics, pages 240–248, 2016. (Cited on page 45.)

[Jian 2015] C. Jian, T. Zhe and L. Zhenxing. *A Review and Analysis of Case-Based Reasoning Research.* In International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS), pages 51–55, 2015. (Cited on page 27.)

[Jiang 2012] L. Jiang. *Learning Instance Weighted Naive Bayes from Labeled and Unlabeled Data.* Journal of Intelligent Information Systems, vol. 38, no. 1, pages 257–268, 2012. (Cited on pages 101, 102 and 105.)

[Jøsang 2016] A. Jøsang. Subjective logic. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer, 2016. (Cited on page 30.)

[Kandel 2011] S. Kandel, J. Heer, C. Plaisant, J. Kennedy, F. van Ham, N. H. Riche, C. Weaver, B. Lee, D. Brodbeck and P. Buono. *Research Directions in Data Wrangling: Visuatizations and Transformations for Usable and Credible Data.* Information visualization, vol. 10, no. 4, pages 271–288, 2011. (Cited on page 40.)

[Kannai 2014] R. Kannai, U. J. Schild and J. Zeleznikow. There is more to legal reasoning with analogies than case based reasoning, but what? Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. (Cited on page 28.)

[Katz 2017a] D. M. Katz, M. J. Bommarit and J. Blackman. *A general approach for predicting the behavior of the Supreme Court of the United States.* PLOS ONE, vol. 12, no. 4, page e0174698, 2017. (Cited on page vi.)

[Katz 2017b] D. M. Katz, M. J. Bommarit and J. Blackman. *A general approach for predicting the behavior of the Supreme Court of the United States.* PLOS ONE, vol. 12, no. 4, page e0174698, 2017. (Cited on pages vi, 16, 17, 18, 19, 26, 27 and 31.)

[Kaye 2005] T. S. Kaye. *Legal Reason: The Use of Analogy in Legal Argument.* 2005. (Cited on page 27.)

[Kelsen 1967] H. Kelsen. Pure theory of law. California library reprint series. University of California Press, 1967. (Cited on page 14.)

[Kim 2017] J. Kim, S. Kim and S. Choi. *Learning to warm-start Bayesian hyperparameter optimization.* arXiv preprint arXiv:1710.06219, 2017. (Cited on page 45.)

[Kimball 2004] B. A. Kimball. *Christopher Langdell: the Case of an 'abomination' in Teaching Practice.* 2004. (Cited on page 28.)

[Kingma 2014] D. P. Kingma and J. Ba. *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014. (Cited on page 46.)

[Klerman 2012] D. Klerman. *The Selection of 13th-Century Disputes for Litigation.* Journal of Empirical Legal Studies, vol. 9, no. 2, pages 320–346, 2012. (Cited on page 19.)

[Kocsor 2004] A. Kocsor, K. Kovács and C. Szepesvári. *Margin maximizing discriminant analysis.* In European Conference on Machine Learning (ECML), pages 227–238. Springer, 2004. (Cited on page 39.)

[Kolodner 1993] J. Kolodner. Case-based reasoning. Artificial intelligence. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. (Cited on page 27.)

[Kotthoff 2017] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter and K. Leyton-Brown. *Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA.* Journal of Machine Learning Research (JMLR), vol. 18, no. 1, pages 826–830, 2017. (Cited on pages v and 44.)

[Kou 2012] G. Kou, Y. Lu, Y. Peng and Y. Shi. *Evaluation of classification algorithms using MCDM and rank correlation.* International Journal of Information Technology & Decision Making, vol. 11, no. 01, pages 197–225, 2012. (Cited on pages 101 and 102.)

[Kowalski 1991] R. A. Kowalski. *Case-based Reasoning and the Deep Structure Approach to Knowledge Representation.* In International Conference on Artificial Intelligence and Law (ICAIL), pages 21–30, New York, NY, USA, 1991. ACM. (Cited on page 16.)

[Kowalski 1996] R. A. Kowalski and F. Toni. *Abstract argumentation.* Artificial Intelligence and Law, vol. 4, no. 3, pages 275–296, 1996. (Cited on page 30.)

[Kuhn 2013] M. Kuhn and K. Johnson. Applied predictive modeling, volume 26. Springer, 2013. (Cited on page 40.)

[Lauderdale 2014] B. E. Lauderdale and T. S. Clark. *Scaling Politically Meaningful Dimensions Using Texts and Votes.* American Journal of Political Science, vol. 58, no. 3, pages 754–771, 2014. (Cited on pages 19, 20, 25, 26 and 31.)

[Lax 1955] P. D. Lax and A. N. Milgram. *IX. Parabolic Equations.* In Contributions to the Theory of Partial Differential Equations. (AM-33), pages 167–190. Princeton University Press, 1955. (Cited on page 125.)

[LeCun 2015] Y. LeCun, Y. Bengio and G. Hinton. *Deep learning.* Nature, vol. 521, no. 7553, page 436, 2015. (Cited on pages 1 and 38.)

[Lee 2001] Y.-J. Lee and O.L. Mangasarian. *SSVM: A Smooth Support Vector Machine for Classification.* Computational Optimization and Applications, vol. 20, no. 1, pages 5–22, 2001. (Cited on pages 101 and 102.)

[Leite 2011] J. Leite and J. Martins. *Social Abstract Argumentation.* In International Joint Conference on Artificial Intelligence (IJCAI), pages 2287–2292. AAAI Press, 2011. (Cited on pages 30, 31 and 32.)

[Leith 2010] P. Leith. *The Rise and Fall of the Legal Expert System.* European Journal of Law and Technology, vol. 1, pages 1–16, 2010. (Cited on page 32.)

[Lemberger 2020] Pirmin Lemberger and Ivan Panico. *A Primer on Domain Adaptation.* arXiv preprint arXiv:2001.09994, 2020. (Cited on page 66.)

[Li 2018] L. Li and K. Jamieson. *Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization.* Journal of Machine Learning Research (JMLR), vol. 18, pages 1–52, 2018. (Cited on page 45.)

[Lifan 2017] S. Lifan, G. Yue, Z. Xibin, W. Hai, G. Ming and S. Jiaguang. *Vertex-Weighted Hypergraph Learning for Multi-View Object Classification.* In International Joint Conferences on Artificial Intelligence (IJCAI), pages 2779–2785, 2017. (Cited on page 41.)

[Lin 2004] Y. Lin. *A note on margin-based loss functions in classification.* Statistics & Probability Letters, vol. 68, no. 1, pages 73–82, 2004. (Cited on page 36.)

[Lundberg 2017] S. M. Lundberg and S.-I. Lee. *A unified approach to interpreting model predictions.* In Advances in Neural Information Processing Systems (NeurIPS), pages 4765–4774, 2017. (Cited on page 117.)

[Mahalanobis 1936] P. C. Mahalanobis. *On the generalised distance in statistics.* In Proceedings of the Indian National Science Academy, volume 2, pages 49–55, 1936. (Cited on page 38.)

[Marcano-Cedeño 2011] A. Marcano-Cedeño, J. Quintanilla-Domínguez and D. Andina. *WBCD breast cancer database classification applying artificial metaplasticity neural network.* Expert Systems with Applications, vol. 38, no. 8, pages 9573–9579, 2011. (Cited on pages 101 and 102.)

[Martin 2004a] A. D. Martin, K. M. Quinn, P. T. Kim and T. W. Ruger. *Competing approaches to predicting supreme court decision making.* Perspectives on Politics, pages 761–767, 2004. (Cited on pages vi, 17 and 31.)

[Martin 2004b] A. D. Martin, K. M. Quinn, T. W. Ruger and P. T. Kim. *Competing Approaches to Predicting Supreme Court Decision Making.* Perspectives on Politics, vol. 2, no. 4, pages 761—-767, 2004. (Cited on page vi.)

[Martinez-Cantin 2014] R. Martinez-Cantin. *Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits.* Journal of Machine Learning Research, vol. 15, no. 1, pages 3735–3739, 2014. (Cited on page v.)

[McFadden 1973] D. McFadden*et al. Conditional logit analysis of qualitative choice behavior.* 1973. (Cited on page 23.)

[Medvedeva 2020] M. Medvedeva, M. Vols and M. Wieling. *Using machine learning to predict decisions of the European Court of Human Rights.* Artifficial Intelligence and Law, 2020. (Cited on pages vi, 19, 55 and 66.)

[Mercer 1909] J. Mercer. *Functions of positive and negative type, and their connection the theory of integral equations.* Philosophical transactions of the royal society of London. Series A., vol. 209, no. 441-458, pages 415–446, 1909. (Cited on page 128.)

[Minton 1990] S. Minton. *Quantitative results concerning the utility of explanation-based learning.* Artificial Intelligence, vol. 42, no. 2, pages 363–391, 1990. (Cited on page 27.)

[Močkus 1975] J. Močkus. *On Bayesian methods for seeking the extremum.* In Optimization Techniques IFIP Technical Conference, pages 400–404. Springer, 1975. (Cited on page 44.)

[Modgil 2009] S. Modgil, N. Faci, F. Meneguzzi, N. Oren, S. Miles and M. Luck. *A Framework for Monitoring Agent-based Normative Systems.* In International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pages 153–160, Richland, SC, 2009. (Cited on pages 30, 31 and 32.)

[Mohri 2018] M. Mohri, A. Rostamizadeh and A. Talwalkar. Foundations of machine learning. 2018. (Cited on pages 37 and 129.)

[Montgomery 2017] D. C. Montgomery. Design and analysis of experiments. John wiley & sons, 2017. (Cited on page 44.)

[Mróz 2020] P. Mróz, A. Quemy, M. Ślażyński, K Kluza and P. Jemioło. *GBEx, towards Graph-Based Explainations.* International Conference Tools with Artificial Intelligence (ICTAI), 2020. (Cited on pages 7 and 105.)

[Nalepa 2018] J. Nalepa, M. Myller, S. Piechaczek, K. Hrynczenko and M. Kawulok. *Genetic Selection of Training Sets for (Not Only) Artificial Neural Networks.* In Beyond Databases, Architectures and Structures (BDAS), pages 194–206, 2018. (Cited on page 45.)

[Nawi 2013] N. M. Nawi, W. H. Atomi and M. Z. Rehman. *The Effect of Data Pre-processing on Optimized Training of Artificial Neural Networks.* Procedia Technology, vol. 11, pages 32–39, 2013. International Conference on Electrical Engineering and Computer Sciences (EECS). (Cited on page 138.)

[Olson 2016] R. S. Olson, N. Bartley, R. J. Urbanowicz and J. H. Moore. *Evaluation of a tree-based pipeline optimization tool for automating data science.* In Genetic and Evolutionary Computation Conference (GECCO), pages 485–492. ACM, 2016. (Cited on pages 45 and 46.)

[Oncina 2006] J. Oncina and M. Sebban. *Learning Stochastic Edit Distance: application in handwritten character recognition.* Pattern Recognition, vol. 39, pages 1575–1587, 2006. (Cited on page 119.)

[Ontañón 2008] S. Ontañón and E. Plaza. *An Argumentation-based Framework for Deliberation in Multi-agent Systems.* In International Conference on Argumentation in Multi-agent Systems (ArgMAS), pages 178–196, Berlin, Heidelberg, 2008. Springer-Verlag. (Cited on pages 30 and 31.)

[Oren 2007] Nir Oren. *An Argumentation Framework Supporting Evidential Reasoning with Applications to Contract Monitoring*. PhD thesis, University of Aberdeen, 2007. (Cited on pages 30, 31 and 32.)

[Pa1 2018] *Data Warehouse Trends Report*. Technical report, Panoply, 2018. (Cited on pages iii, 3 and 40.)

[Paige 1987] R. Paige and R. E. Tarjan. *Three Partition Refinement Algorithms*. SIAM Journal on Computing, vol. 16, no. 6, pages 973–989, 1987. (Cited on page 98.)

[Pedregosa 2011] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay. *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research (JMLR), vol. 12, pages 2825–2830, 2011. (Cited on pages 57, 101, 133 and 144.)

[Polat 2007] K. Polat and S. Güneş. *Breast cancer diagnosis using least square support vector machine*. Digital Signal Processing, vol. 17, no. 4, pages 694–701, 2007. (Cited on pages 101 and 102.)

[Polyzotis 2017] N. Polyzotis, S. Roy, S. E. Whang and M. Zinkevich. *Data Management Challenges in Production Machine Learning*. In International Conference on Management of Data (SIGMOD), pages 1723–1726. ACM, 2017. (Cited on page v.)

[Posner 2005] R. A. Posner. *Reasoning by analogy*, 2005. (Cited on page 27.)

[Posner 2011] R. A. Posner. Economic analysis of law. Aspen Publishers, 2011. (Cited on page 13.)

[Prakken 1998] H. Prakken and G. Sartor. Modelling reasoning with precedents in a formal dialogue game, pages 127–183. Springer Netherlands, Dordrecht, 1998. (Cited on page 28.)

[Preda 2010] C. Preda, G. Saporta and M. H. B. H. Mbarek. *The NIPALS algorithm for missing functional data*. Revue Roumaine de Mathématiques Pures et Appliquées, vol. 55, no. 4, pages 315–326, 2010. (Cited on page 40.)

[Pu 2012] L. Pu and B. Faltings. *Hypergraph learning with hyperedge expansion*. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD), pages 410–425. Springer, 2012. (Cited on page 117.)

[Quemy] A. Quemy. *Predictions of the European Court of Human Rights*. `https://github.com/aquemy/ECHR-OD_predictions`. (Cited on pages 52 and 57.)

[Quemy 2017] A. Quemy. *Data Science Techniques for Law and Justice: Current State of Research and Open Problems*. In Advances in Databases and Information Systems (ADBIS) Workshops and Short papers, pages 302–312. Springer, 2017. (Cited on pages 8 and 15.)

[Quemy 2018a] A. Quemy. *AI for the legal domain: an explainability challenge.* In PhD Student Research Competition, IFIP World Computer Congress, 2018. (Cited on pages 7 and 15.)

[Quemy 2018b] A. Quemy. *Binary Classification With Hypergraph Case-Based Reasoning.* In International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP) @ International Conference on Extending Database Technology/International Conference on Database Theory (EDBT/ICDT), 2018. (Cited on pages x, xi, 7 and 81.)

[Quemy 2019a] A. Quemy. *Binary classification in unstructured space with hypergraph case-based reasoning.* Information Systems, vol. 85, pages 92–113, 2019. (Cited on pages x, xi, 7 and 81.)

[Quemy 2019b] A. Quemy. *Data Pipeline Selection and Optimization.* In International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP) @ International Conference on Extending Database Technology/International Conference on Database Theory (EDBT/ICDT) Joint Conference, 2019. (Cited on pages viii, xi, 7 and 138.)

[Quemy 2020a] A. Quemy. *Two-stage optimization for machine learning workflow.* Information Systems, vol. 92, page 101483, 2020. (Cited on pages viii, xi, 7, 66, 69 and 138.)

[Quemy 2020b] A. Quemy and R. Wrembel. *ECHR-DB: On Building an Integrated Open Repository of Legal Documents for Machine Learning Applications.* Information Systems (submitted), 2020. (Cited on pages xii, 7 and 50.)

[Quemy 2020c] A. Quemy and R. Wrembel. *On Integrating and Classifying Legal Text Documents.* International Conference on Database and Expert Systems Applications (DEXA), vol. 12391, 2020. (Cited on pages xii, 7 and 50.)

[Quemy 2021] A. Quemy. *A Physical Approach to Classification.* In To be submitted to International Conference on Machine Learning (ICML), 2021. (Cited on pages 7 and 124.)

[Quinlan 1996] J. R. Quinlan. *Improved use of continuous attributes in C4. 5.* Journal Artificial Intelligence Research, vol. 4, pages 77–90, 1996. (Cited on pages 101 and 102.)

[Quinn 2002] K. M. Quinn and A. D. Martin. *Dynamic Ideal Point Estimation via Markov Chain Monte Carlo for the U.S. Supreme Court, 1953-1999.* Political Analysis, vol. 10, no. 2, pages 134–153, 2002. (Cited on pages 19, 21 and 31.)

[Quinn 2006] K. M. Quinn, J. H. Park and A. D. Martin. *Improving judicial ideal point estimates with a more realistic model of opinion content*, 2006. Unpublished. (Cited on pages 19, 21 and 31.)

[Rago 2016] A. Rago, F. Toni, M. Aurisicchio and P. Baroni. *Discontinuity-Free Decision Support with Quantitative Argumentation Debates.* In Principles of Knowledge Representation and Reasoning (KR), pages 63–73, 2016. (Cited on pages 30, 31 and 32.)

[Rakotoarison 2018] H. Rakotoarison and M. Sebag. *AutoML with Monte Carlo Tree Search*. In Workshop AutoML @ International Conference on Machine Learning (ICML), Stockholm, Sweden, 2018. Brazdil, P. and Giraud-Carrier, C. and Guyon, I. (Cited on pages 44 and 46.)

[Řehůřek 2010] R. Řehůřek and P. Sojka. *Software Framework for Topic Modelling with Large Corpora*. In Worshop on New Challenges for NLP Frameworks, pages 45–50. ELRA, 2010. (Cited on page 56.)

[Ribeiro 2016] M. T. Ribeiro, S. Singh and C. Guestrin. *Why should i trust you?: Explaining the predictions of any classifier*. In International Conference on Knowledge Discovery and Data Mining (SIGKDD), pages 1135–1144. ACM, 2016. (Cited on page 117.)

[Rissland 1991] E. L. Rissland and D. B. Skalak. *CABARET: Rule Interpretation in a Hybrid Architecture*. International Journal of Man-Machine Studies, vol. 34, no. 6, pages 839–887, 1991. (Cited on page 28.)

[Rissland 2006] E. L. Rissland. *AI and Similarity*. IEEE Intelligent Systems, vol. 21, no. 3, pages 39–49, 2006. (Cited on pages iii and 2.)

[Robertson 2005] G. G. Robertson, M. P. Czerwinski and J. E. Churchill. *Visualization of mappings between schemas*. In Conference on Human Factors in Computing Systems, pages 431–439. ACM, 2005. (Cited on page 40.)

[Rosenblatt 1958] F. Rosenblatt. *The perceptron: a probabilistic model for information storage and organization in the brain*. Psychological Review, vol. 65, no. 6, page 386, 1958. (Cited on page 36.)

[Rosenfeld 1998] M. Rosenfeld. Just interpretations: Law between ethics and politics. Just Interpretations: Law Between Ethics and Politics. University of California Press, 1998. (Cited on page 14.)

[Ross 2004] A. Ross. On law and justice. Lawbook Exchange, 2004. (Cited on page 14.)

[Royston 2004] P. Roystonet *al*. *Multiple imputation of missing values*. Stata Journal, vol. 4, no. 3, pages 227–41, 2004. (Cited on page 40.)

[Ruger 2004] T. W. Ruger, P. T. Kim, A. D. Martin and K. M. Quinn. *The Supreme Court Forecasting Project: Legal and Political Science Approaches to Predicting Supreme Court Decisionmaking*. Columbia Law Review, vol. 104, no. 4, page 1150–1210, 2004. (Cited on pages vi, 4, 16, 17, 28 and 31.)

[Sagir 2017] A. M. Sagir and S. Sathasivam. *A Hybridised Intelligent Technique for the Diagnosis of Medical Diseases*. Pertanika Journal of Science & Technology (JST), vol. 25, no. 2, 2017. (Cited on pages 101 and 102.)

[Schmidhuber 2015] J. Schmidhuber. *Deep learning in neural networks: An overview*. Neural Networks, vol. 61, pages 85–117, 2015. (Cited on pages 1 and 38.)

[Segal 1989] J. A. Segal and A. D. Cover. *"Ideological Values and the Votes of U.S. Supreme Court Justices"*. American Political Science Review, vol. 83, no. 2, pages 557–565, 1989. (Cited on pages 19, 20, 23 and 31.)

[Segal 1995] J. A. Segal, L. Epstein, C. M. Cameron and H. J. Spaeth. *Ideological Values and the Votes of U.S. Supreme Court Justices Revisited*. The Journal of Politics, vol. 57, no. 3, pages 812–823, 1995. (Cited on pages 19, 20 and 31.)

[Sim 2014] Y. Sim, B. R. Routledge and N. A. Smith. *The Utility of Text: The Case of Amicus Briefs and the Supreme Court*. CoRR, vol. abs/1409.7985, 2014. (Cited on pages 19, 23, 25 and 31.)

[Smith 1759] Adam Smith. The theory of moral sentiments. McMaster University Archive for the History of Economic Thought, 1759. (Cited on page 13.)

[Smyth 1998] B. Smyth and M. T. Keane. *Adaptation-guided retrieval: questioning the similarity assumption in reasoning*. Artificial intelligence, vol. 102, no. 2, pages 249–293, 1998. (Cited on page 27.)

[Snoek 2012] J. Snoek, H. Larochelle and R. P. Adams. *Practical bayesian optimization of machine learning algorithms*. In Neural Information Processing Systems (NeurIPS), pages 2951–2959, 2012. (Cited on page 44.)

[Spitzer 1994] M. L. Spitzer and L. Cohen. *Solving the Chevron Puzzle*. Journal of Law & Contemporary Problems, vol. 57, page 65, 1994. (Cited on page 19.)

[Stahl 2003] A. Stahl and T. Gabel. *Using evolution programs to learn local similarity measures*. In International Conference on Case-Based Reasoning, pages 537–551. Springer, 2003. (Cited on page 27.)

[Stekhoven 2011] D. J. Stekhoven and P. Bühlmann. *MissForest—non-parametric missing value imputation for mixed-type data*. Bioinformatics, vol. 28, no. 1, pages 112–118, 2011. (Cited on page 40.)

[Stigler 1971] G. Stigler. *The Theory of Economic Regulation*. Bell Journal of Economics, vol. 2, no. 1, pages 3–21, 1971. (Cited on page 13.)

[Stigler 1974] G. Stigler. *Good Economics, Bad Law*. Virginia Law Review, vol. 60, no. 3, pages 483–492, 1974. (Cited on page 13.)

[Stranieri 1999] A. Stranieri, J. Zeleznikow, M. Gawler and B. Lewis. *A hybrid rule – neural approach for the automation of legal reasoning in the discretionary domain of family law in Australia*. Artificial Intelligence and Law, vol. 7, no. 2, pages 153–183, 1999. (Cited on page 28.)

[Süli 2003] E. Süli and D. F. Mayers. An introduction to numerical analysis. Cambridge university press, 2003. (Cited on page 126.)

[Sun 2019] X. Sun, J. Lin and B. Bischl. *ReinBo: Machine Learning pipeline search and configuration with Bayesian Optimization embedded Reinforcement Learning.* CoRR, vol. abs/1904.05381, 2019. (Cited on page 45.)

[Sunstein 1998] C. R. Sunstein. *How Law Constructs Preferences.* Georgetown Law Journal, vol. 86, pages 2637–2652, 1998. (Cited on page 14.)

[Swersky 2014] K. Swersky, J. Snoek and A. R. Prescott. *Freeze-thaw Bayesian optimization.* arXiv preprint arXiv:1406.3896, 2014. (Cited on page 45.)

[Tahir 2016] M. A. U. H. Tahir, S. Asghar, A. Zafar and S. Gillani. *A Hybrid Model to Detect Phishing-Sites Using Supervised Learning Algorithms.* In International Conference on Computational Collective Intelligence (ICCCI), pages 1126–1133, 2016. (Cited on pages 101 and 102.)

[Tanenhaus 1967] J. Tanenhaus. *The Judicial Mind: Attitudes and Ideologies of Supreme Court Justices 1946-1963. Glendon Schubert.* The Journal of Politics, vol. 29, no. 3, pages 677–679, 1967. (Cited on page 19.)

[Thabtah 2016] F. Thabtah, R. M. Mohammad and L. McCluskey. *A dynamic self-structuring neural network model to combat phishing.* In International Joint Conference on Neural Network, pages 4221–4226, 2016. (Cited on pages 101 and 102.)

[Thornton 2013] C. Thornton, F. Hutter, H. H. Hoos and K. Leyton-Brown. *Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms.* In International Conference on Knowledge Discovery and Data Mining (SIGKDD), pages 847–855. ACM, 2013. (Cited on page 44.)

[Tolosi 2011] L. Tolosi and T. Lengauer. *Classification with correlated features: unreliability of feature ranking and solutions.* Bioinformatics, vol. 27, no. 14, pages 1986–1994, 2011. (Cited on page 18.)

[Troper 2001] M. Troper. La théorie du droit, le droit, l'état. Léviathan (Paris). Presses universitaires de France, 2001. (Cited on page 14.)

[Übeyli 2007] E. D. Übeyli. *Implementing automated diagnostic systems for breast cancer detection.* Expert Systems with Applications, vol. 33, no. 4, pages 1054–1062, 2007. (Cited on pages 101, 102 and 105.)

[Van Buuren 2018] S. Van Buuren. Flexible imputation of missing data. Chapman and Hall/CRC, 2018. (Cited on page 40.)

[Vanschoren 2018] J. Vanschoren. *Meta-learning: A survey.* arXiv preprint arXiv:1810.03548, 2018. (Cited on page 45.)

[Vapnik 2013] V. Vapnik. The nature of statistical learning theory. Springer science & business media, 2013. (Cited on page 36.)

[Čyras 2016a] K. Čyras, K. Satoh and F. Toni. *Abstract Argumentation for Case-based Reasoning.* In International Conference on Principles of Knowledge Representation and Reasoning (KR), pages 549–552. AAAI Press, 2016. (Cited on pages 30 and 31.)

[Čyras 2016b] K. Čyras, K. Satoh and F. Toni. *Explanation for Case-Based Reasoning via Abstract Argumentation.* Frontiers in Artificial Intelligence and Applications, vol. 287, no. Computational Models of Argument, page 243–254, 2016. (Cited on page 30.)

[Venables 2002] W. N. Venables and B. D. Ripley. *Tree-based methods.* In Modern Applied Statistics with S, pages 251–269. Springer, 2002. (Cited on page 38.)

[Wang 2015] F. Wang and J. Sun. *Survey on Distance Metric Learning and Dimensionality Reduction in Data Mining.* Data Mining and Knowledge Discovery, vol. 29, no. 2, pages 534–564, 2015. (Cited on pages v, 38 and 119.)

[Weinberger 2006] K. Q. Weinberger, J. Blitzer and L. K. Saul. *Distance metric learning for large margin nearest neighbor classification.* In Advances in Neural Information Processing Systems (NeurIPS), volume 10, pages 1473–1480, 2006. (Cited on page 38.)

[Weinberger 2009] K. Q. Weinberger and L. K. Saul. *Distance metric learning for large margin nearest neighbor classification.* Journal of Machine Learning Research (JMLR), vol. 10, no. 2, 2009. (Cited on page 38.)

[Weinreb 2005] Lloyd L. Weinreb. Legal reason: The use of analogy in legal argument. Cambridge University Press, 2005. (Cited on page 27.)

[Williams 2006] C. KI Williams and C. E. Rasmussen. Gaussian processes for machine learning, volume 2. MIT Press Cambridge, 2006. (Cited on page 37.)

[Wilson 2018] J. Wilson, F. Hutter and M. Deisenroth. *Maximizing acquisition functions for Bayesian optimization.* In Neural Information Processing Systems (NeurIPS), pages 9884–9895, 2018. (Cited on page 44.)

[Wolpert 1996] D. H. Wolpert. *The lack of a priori distinctions between learning algorithms.* Neural Computation, vol. 8, no. 7, pages 1341–1390, 1996. (Cited on page 138.)

[Zhou 2007] D. Zhou, Jiayuan H. and B. Schölkopf. *Learning with Hypergraphs: Clustering, Classification, and Embedding.* In B. Schölkopf, J. C. Platt and T. Hoffman, editors, Advances in Neural Information Processing Systems (NeurIPS), pages 1601–1608. MIT Press, 2007. (Cited on page 41.)