Deep learning in public health, and contributions to statistical learning

Yiyang Yu


École doctorale de Sciences Mathématiques de Paris Centre (ED 386)

Laboratoire de Probabilités, Statistique et Modélisation (LPSM, UMR 8001)

Apprentissage profond en santé publique, et contributions en apprentissage statistique

Deep learning in public health, and contributions to statistical learning

Thèse pour obtenir le grade de docteur délivré par

Université Paris Cité

Mathématiques Appliquées

présentée et soutenue publiquement par

Yiyang Yu

le 6 avril 2022

Dirigée par : **Stéphane Gaïffas**
Et par : **Emmanuel Bacry**

Composition du jury :

Alexandre Gramfort,	DR, INRIA Saclay	Rapporteur
Tim Van Erven,	PR, University of Amsterdam	Rapporteur
Stéphane Gaïffas,	PR, Université de Paris	Directeur de thèse
Emmanuel Bacry,	DR, CNRS, Université Paris Dauphine PSL	Directeur de thèse
Chloé-Agathe Azencott,	MCF, Mines ParisTech PSL	Examinatrice
Gilles Stoltz,	DR, CNRS, Université Paris Saclay	Président du jury

Résumé

Le développement d'algorithmes efficaces pour apprendre des représentations appropriées de données structurées, telles que des séquences d'événements datés provenant de la vie réelle, est un défi majeur et central de l'apprentissage automatique. Dans cette optique, l'apprentissage profond est devenu populaire pour modéliser des données structurées, parfois combiné avec des techniques de pré-entraînement. En même temps, d'autres méthodes d'apprentissage statistique plus "classiques", comme les forêts aléatoires ou la régression, occupent toujours une place importante dans la pratique à cause de leur efficacité.

Dans cette thèse, nous apportons quelques contributions à l'étude théorique et numérique de certains problèmes de l'apprentissage statistique, ainsi que l'application de l'apprentissage profond aux données de la santé publique.

La première contribution consiste à introduire un nouveau modèle appelé ZiMM (Zero-inflated Mixture of Multinomial distributions), et une architecture Encodeur-Décodeur (ED) de réseaux de neurones profonds entraînés de-bout-en-bout, modélisant les parcours de soins pour la prédiction des complications post-chirurgicales. ZiMM-ED est appliqué aux données de santé de remboursement de soins provenant du Système National des Données de Santé (SNDS) en France, qui est une base de données non-clinique, contenant seulement les codes de remboursement datés d'achats de médicaments et des diagnostics hospitaliers. En particulier, nous considérons les complications jusqu'au 18e mois après la chirurgie, ce qui correspond à des observations "floues" car seulement observées à partir des achats de médicaments d'une famille spécifique. Nos expériences montrent les améliorations en termes de performance prédictive de ZiMM-ED par rapport à plusieurs modèles de référence. ZiMM-ED ouvre la voie de l'exploitation d'un tel jeu de données avec peu de prétraitement à grâce aux réseaux de neurones profonds. Cette base de données est jusque-là utilisée principalement pour des raisons administratives (remboursement des soins de santé), et nous montrons le pouvoir prédictif des réseaux de neurones profonds dessus sur une telle base de données avec un cas précis.

La deuxième contribution porte sur l'étude théorique de l'apprentissage contrastif de représentation, une technique récemment devenue populaire et expérimentalement efficace pour l'entraînement auto-supervisé. En se basant sur quelques résultats proposant des cadres d'étude théoriques, nous étendons la garantie pour la qualité des représentations apprises dans la phase pré-entrainement non-supervisé avec une perte contrastive et de multiples échantillons négatifs, la qualité étant mesurée en termes de performance prédictive pour les tâches supervisées en aval. En outre, nous fournissons une garantie de convergence quant à la minimisation de la perte contrastive avec la descente de gradient pour un encodeur de réseaux de neurones sur-paramétré. Ces résultats théoriques, combinant des expériences numériques, ouvrent des portes pour une meilleure compréhension des pratiques de

pré-entrainement - affinement très utilisées aujourd'hui en apprentissage profond.

La troisième contribution consiste à introduire un nouvel algorithme de type forêt aléatoire, que nous nommons WildWood. Alors que l'algorithme standard de forêt aléatoire utilise des échantillons bootstrap out-of-bag seulement pour calculer des scores, WildWood utilise ces échantillons pour améliorer les prédictions en calculant l'agrégation de tous les sous-arbres possibles de chaque arbre dans la forêt : ce calcul est exact et efficace grâce à l'algorithme de context tree weighting. Nous montrons que théoriquement, la perte induite par une telle agrégation est comparable à celle du meilleur sous-arbre possible. Nous proposons une implémentation Python open-source de WildWood avec une stratégie d'histogramme qui permet d'accélérer la recherche des coupures impliquées dans la construction des arbres. Notre implémentation est rapide et compétitive en comparaison avec d'autres algorithmes ensemblistes bien connus, par exemple la forêt aléatoire standard et les algorithmes d'extrême gradient boosting.

Enfin, le dernier chapitre de cette thèse est consacré à la régression logistique en ligne et considère le regret par rapport à la boule ℓ_2 de rayon B. Alors qu'il est connu que les algorithmes propres avec regret logarithmique en le nombre d'itérations n subissent nécessairement un facteur exponentiel en B dans leur borne de regret, quelques algorithmes impropres, bayésiens et non-bayésiens, ont été introduits récemment avec des meilleures garanties. Dans le but d'obtenir une garantie de regret optimale, nous proposons deux algorithmes impropres et non-bayésiens, OSMP et AOSMP, reposant sur une stratégie "minmax à une étape", avec la fonction de perte exacte pour OSMP, et une fonction de perte approchée pour AOSMP. Nos analyses de regret s'appuient entre autres sur la propriété de self-concordance généralisée de la fonction logistique. Pour OSMP, malgré une borne supérieure obtenue pour les regrets instantanés, nous expliquons en quoi l'amélioration des bornes de regret est une question difficile, à laquelle AOSMP apporte une réponse comparable à l'état de l'art de la garantie de regret.

Mots clefs : Apprentissage statistique, Apprentissage profond, Données de santé, Apprentissage contrastif, Forêts aléatoires, Régression logistique en ligne

Abstract

Developing efficient algorithms to learn appropriate representations of structured data, including sequences of timestamped events, is a major and central challenge in machine learning. To this end, deep learning has become popular in modeling structured data, sometimes combined with some pre-training techniques. At the same time, classical machine learning methods, such as random forest and regression, always have an important place because of their efficiency.

In this thesis, we make some contributions to the theoretical and numerical studies of some machine learning problems, as well as the application of deep learning on the public health data.

The first contribution introduces a new model ZiMM (Zero-inflated Mixture of Multinomial distributions), and an Encoder-Decoder (ED) end-to-end deep neural networks architecture, modelling the healthcare pathways for post-surgical complications prediction. ZiMM-ED is applied on the healthcare claims data coming from the French national system of health data (SNDS), which is a non-clinical database, only containing timestamped reimbursement codes for drugs purchases, medical procedures and hospital diagnoses. In particular, we consider the complication until the 18-th month after the surgery, which is also blurry since we only observe this from the purchase of drugs from a specific family. Our experiments show that ZiMM-ED improves several baselines in terms of prediction performance. At the same time, ZiMM-ED paves the way for leveraging such a dataset with little pre-processing, using deep neural networks. Hence, we take advantage of a database which is until now only used for administrative ends (healthcare expense reimbursement), and we show the predictive power of the deep neural networks on it with a precise application.

The second contribution focuses on the theoretical study of contrastive representation learning, a technique recently becoming popular and experimentally shown to be efficient for self-supervised learning. Relying on some previous works introducing the theoretical framework, we extend the guarantees for the quality of the representations learned during the pre-training phase, with the contrastive loss and multiple negative samples, the quality being measured by the predictive performance of some down-stream supervised tasks. Furthermore, we provide a convergence guarantee for the minimization of the contrastive training error with gradient descent of an over-parametrized neural network encoder. These theoretical results, combined with some illustrative experiments, open doors for a better understanding of the typical pre-training – fine-tuning practices.

Next, the third contribution consists of introducing a new algorithm of type random forest, that we name WildWood. While the standard random forest algorithm uses bootstrap out-of-bag samples to compute out-of-bag score, WildWood uses these samples to produce improved predictions given by an aggregation of all

the possible sub-trees for each fully-grown tree in the forest: this computation is precise and efficient thanks to the context tree weighting algorithm. We show that theoretically, the loss of such an aggregation is comparable to the loss of the best possible subtree. We propose an open-source Python implementation of WildWood with the histogram strategy which accelerates split finding. Our implementation is fast and competitive compared to other well-established ensemble methods, such as standard random forest and extreme gradient boosting algorithms.

Finally, the last chapter of this thesis is devoted to the problem of online logistic regression and considers the regret with respect to the ℓ_2-ball of radius B. While it is known that proper algorithms with a regret logarithmic in the number of rounds n necessarily suffer from an exponential factor in B, some improper algorithms, Bayesian and non-Bayesian, were recently introduced with better regret guarantees. For the purpose of obtaining an optimal regret guarantee, we introduce two improper non-Bayesian algorithms, OSMP and AOSMP, based on the one-step minmax strategy, with the exact loss function for OSMP, and an approximated loss function for AOSMP. Our regret analysis relies, among others, on the generalized self-concordance property of the logistic function. For OSMP, although we find an upper-bound for the instant regrets, we explain why obtaining improved regret upper-bounds is a difficult question, to which AOSMP brings an answer similar to the state-of-the-art one.

Keywords: Machine learning, Deep learning, Medical claims data, Contrastive learning, Random forests, Online logistic regression

Remerciements

Mes premiers remerciements vont à mes directeurs de thèse, sans qui cette thèse n'aurait pas été possible. Je remercie Emmanuel pour m'avoir introduite dans la recherche sur la représentation des parcours de soins, pour nos séances de brainstorming, et pour tes conseils. Je remercie Stéphane pour ta rigueur, pour ta patience, ainsi que pour le plaisir de discuter des maths, des modélisations et de la recherche, et bien évidemment pour tous ces astuces d'implémentation que tu as hâte de nous partager. Merci à tous les deux pour la confiance que vous avez eu en moi, pour votre bonne humeur et pour votre générosité. J'espère sincèrement que nous pourrons rester en contact dans l'avenir.

Thanks again to the reviewers of my thesis, Alexandre Gramfort and Tim Van Erven, I am sincerely honored, thank you for your reports with great attention. Merci à Chloé-Agathe Azencott et à Gilles Stoltz d'avoir accepté de faire partie du jury.

Un grand merci à mes co-auteurs. Merci à Anastasiia pour ta persévérance et ta bonne humeur, pour nos séances de débugging. Merci à Ibrahim pour ton enthousiasme, pour nos passionnantes discussions. Merci à Prof. Lukacs pour vos explications du contexte médical et pour Observapur. Merci à Jaouad pour m'avoir introduite dans le monde d'apprentissage en ligne, et pour ta rigueur.

Merci à Alain, Anastasiia, Ibrahim, Martin, Marcello, Maryan, Peng, Phong, Qing, Ruihua, et à Angel, Daniel, Dian, Firas, Philip, Youcef, avec qui j'ai eu le plaisir partager tant de discussions au sujet de nos projets de recherche pour mieux modéliser les parcours de soins ou pour rendre meilleure la santé publique, comme au sujet de la dégustation d'une tasse de café filtré, dans notre bureau 05-30-108, au Magnan, ou au Café Leffe.

Merci aux membres du LPSM avec qui j'ai partagé de grands moments. Je remercie les doctorants du labo, commençant avec ceux qui ont eu le privilège de partager le bureau Serpentard : Côme, Enzo, Guillaume S., Houzhi, Hoang, Junchao, Maximilian, Nissrine, William ; merci à Assaf, Cyril, Guillaume C.-K., Lucas, Hiroshi, Ibrahim, Sothea ; à Aaraona, Antoine, Bogdan, Clément, Fabio, Mohan, Nathan, Ottavio, Sylvain, Yann ; à Abdel, Azar, Benjamin, Barbara, Marc, Ziad. Je remercie Nathalie, Valérie, Florence et Amina qui ont fait un travail remarquable pour m'accompagner dans toutes mes démarches administratives.

Cette thèse n'aurait pas été possible sans le soutien de la Fondation des Sciences Mathématiques de Paris et de DIM Maths Innov, dont je suis reconnaissante. Merci à Dominique pour la gestion du financement, et qui je sais que je peux toujours compter sur.

Merci à mes amis, en Chine et en France, pour votre compagnie, et merci pour les bons moments passés ensemble.

Un immense merci à ma famille, qui me soutient toujours sans condition, qui

m'a offert le privilège de poursuivre mes rêves. Je pense en particulier à mes grand-parents et à mes parents, qui m'ont transmis le goût pour la science et pour les mathématiques dès mon plus jeune âge.

Merci à Heshu pour ton amour. Merci pour ton soutien indéfectible pendant ces années (mention spéciale aux heures passées avec moi pour chercher les citations les plus pertinentes à mettre au début de chaque chapitre). Je suis certaine que l'avenir nous réserve encore beaucoup d'aventures.

Contents

Résumé détaillé en français

Cette thèse traite du sujet général de l'apprentissage automatique *(Machine learning)*, un cadre général pour le problème de l'extraction d'informations et de la prise de décisions à partir de données. Alors que la dernière décennie a vu le renouveau des réseaux de neurones, rebaptisés apprentissage profond *(Deep learning)*, et ses nombreuses applications réussies dans le monde réel grâce à des jeux de données plus importants et à des capacités de calcul plus fortes, la recherche méthodologique en apprentissage automatique est plus que jamais cruciale pour le progrès d'un tel domaine, permettant la conception de nouveaux algorithmes avec des propriétés souhaitables ainsi que la compréhension plus profonde de certains succès pratiques.

Dans cette thèse, j'ai travaillé sur les applications de l'apprentissage profond sur des données de santé publique, ainsi que sur des études théoriques et numériques de certains problèmes d'apprentissage automatique. La première contribution consiste à exploiter les données de santé de remboursement de soins, disponibles dans le Système National des Données de Santé (SNDS), en utilisant l'apprentissage profond (Chapitre 2). Ensuite, les contributions se concentrent sur les thèmes de l'apprentissage contrastif (Chapitre 3), les méthodes basées sur les arbres d'ensemble (Chapitre 4), et la régression logistique en ligne (Chapitre 5).

Ce résumé en français presénte dans l'ordre chaque contribution de façon succincte.

Chapitre 2 : ZiMM, un modèle d'apprentissage profond pour les complications à long terme et floues avec des données de santé de remboursement de soins non-cliniques

Le travail présenté dans le Chapitre 2 est une tentative de relever certains défis dans l'application de l'apprentissage profond aux données de santé (EHRs ou celles de remboursement), sous forme d'une étude d'un cas médical concret.

Nous considérons les problèmes de modélisation et de prédiction d'une complication à long terme et "floue" qui se produit après une procédure médicale telle qu'une chirurgie. Il ne s'agit pas ici d'une complication à court terme liée à la procédure elle-même, mais d'une complication à long terme que les cliniciens ne peuvent pas expliquer facilement, car elle dépend de séquences inconnues d'événements passés qui se sont produits avant la procédure. La complication n'est observée qu'indirectement, de manière "floue", par le biais de prescriptions longitudinales de médicaments sur une longue période de temps après la procédure médicale. Notre motivation pour utiliser le SNDS, une base de données de réclamations médicales non cliniques, est son exhaustivité au niveau de la population nationale. En effet, nous considérons un jeu de données extraites contenant les remboursements de soins médicaux de presque tous les résidents français ayant subi une intervention chirurgicale pour des problèmes prostatiques, avec un historique entre 1,5 et 5 ans.

Pour mieux saisir les complications à long terme et floues, nous introduisons le modèle ZiMM *(Zero-inflated Mixture of Multinomial distributions model)*. Désignons par $n_i = \sum_{b=1}^{B} y_{i,b}$ le nombre total de complications floues du patient i, avec B le nombre de laps de temps post-chirurgical que nous considérons (18 mois en occurrence). Nous supposons que $n_i \in \{0, 1, \ldots, B\}$ est distribué (conditionnellement à x_i) comme $n_i \sim \text{Categorical}(\pi_0(x_i), \pi_1(x_i), \ldots, \pi_B(x_i))$, où $\pi_b(x_i)$ sont tels que $\sum_{b=0}^{B} \pi_b(x_i) = 1$ et $\pi_b(x_i) \geqslant 0$ (sortant d'une activation softmax par exemple). Ces paramètres correspondent à la distribution catégorielle propre au patient i. Ensuite, on suppose que la distribution de y_i conditionnelle à $n_i = b$ et x_i suit soit une distribution de Dirac sur le vecteur (de taille B) $[0, \ldots, 0]$ lorsque $b = 0$, soit une distribution multinomiale de paramètres b et $p_{b,1}(x_i), \ldots, p_{b,B}(x_i)$, à savoir

$$y_i | (x_i, n_i = b) \sim \begin{cases} \delta_{[0,\ldots,0]} & \text{si } b = 0, \\ \text{Multinomial}(b, p_{b,1}(x_i), \ldots, p_{b,B}(x_i)) & \text{sinon,} \end{cases}$$

où $p_{b,1}(x_i), \ldots, p_{b,B}(x_i)$ sont les paramètres d'une distribution multinomiale lorsque $n_i = b$, pour chaque $b = 1, \ldots, B$. Une fois encore, ces paramètres sont spécifiques au patient i.

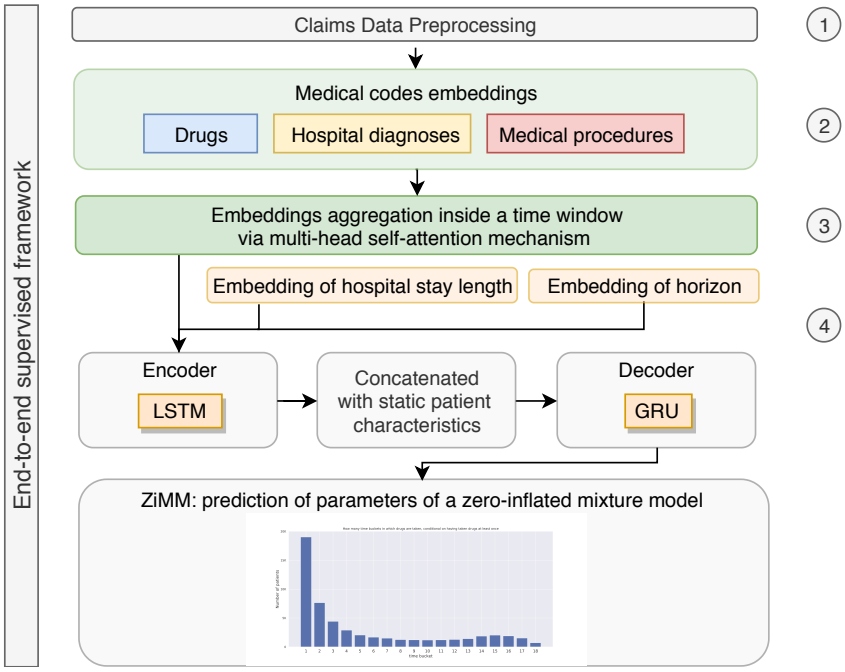

FIGURE 1 – Architecture ZiMM Encoder-Decoder de-bout-en-bout.

En plus de cette modélisation probabiliste des complications post-chirurgicales, nous construisons une architecture d'apprentissage profond de-bout-en-bout, appelée ZiMM Encodeur-Decodeur (ZiMM ED), capable d'apprendre à partir des modèles complexes, irréguliers, hétérogènes et épars d'événements de santé observés dans une base de données de remboursements de soins uniquement. Les composantes de ZiMM ED incluent embedding des codes médicaux, mécanisme d'attention pour agréger au sein d'une même fenêtre de temps, Long Short Term Memory, concaténation avec les embeddings issues des features statics, comme l'illustre la Figure 1.

Nos expériences montrent que ZiMM ED améliore plusieurs modèles prédictifs de comparaison, y compris les approches d'apprentissage classiques et d'apprentissage profond, et qu'il permet de travailler sur un tel ensemble de données avec un minimum de travail de prétraitement.

Chapitre 3 : Garanties théoriques sur l'apprentissage contrastif non supervisé de représentation pour la classification et sur sa convergence

La motivation du travail présenté dans le Chapitre 3 est de mieux comprendre les techniques d'entraînement non-supervisées et auto-supervisées récemment populaires, du point de vue de la modélisation.

En suivant le cadre proposé pour la première fois dans Saunshi et al. [2019], supposons que nous sommes capables d'échantillonner des paires *positives* (x, x^+) à partir de la distribution

$$\mathcal{D}_{\text{sim}}(x, x^+) = \sum_{c \in \mathcal{C}} \rho(c) \mathcal{D}_c(x) \mathcal{D}_c(x^+), \tag{1}$$

à savoir, (x, x^+) est échantillonné comme un mélange de paires indépendantes conditionnellement à une classe latente partagée, échantillonnée selon ρ. D'autre part, nous supposons que nous pouvons échantillonner *négatif* des échantillons x^- à partir de la distribution

$$\mathcal{D}_{\text{neg}}(x^-) = \sum_{c \in \mathcal{C}} \rho(c) \mathcal{D}_c(x^-). \tag{2}$$

Sous les hypothèses données dans les équations (1.1) et (1.2), nous considérons la perte contrastive non supervisée suivante avec N échantillons négatifs,

$$L_{\text{un}}^N(f) = \mathbb{E}_{\substack{(x,x^+)\sim\mathcal{D}_{\text{sim}} \\ X^-\sim\mathcal{D}_{\text{neg}}^{\otimes N}}} \left[-\log \left(\frac{\exp\left(f(x)^\top f(x^+)\right)}{\exp\left(f(x)^\top f(x^+)\right) + \sum_{x^- \in X^-} \exp\left(f(x)^\top f(x^-)\right)} \right) \right], \tag{3}$$

où $\mathcal{D}_{\text{neg}}^{\otimes N}$ représente le produit tensoriel N de la distribution $\mathcal{D}_{\text{neg}}$ donnée par l'équation (2).

Proposition (Proposition 3.3)**.** *Considérons la perte non supervisée $L_{\text{un}}^N(f)$ de l'équation (1.3) avec N échantillons négatifs. Supposons que ρ est uniforme sur $\mathcal{C}$ et que $2 \leqslant k+1 \leqslant N_{\mathcal{C}}$. Alors, pour toute fonction encodeur $f : \mathcal{X} \to \mathbb{R}^d$, nous avons*

$$L_{\text{sup},k}(f) \leqslant L_{\text{sup},k}^{\mu}(f) \leqslant \frac{k}{1 - \tau_N^+} \left(L_{\text{un}}^N(f) - \tau_N^+ \log(N+1) \right)$$

avec $\tau_N^+ = \mathbb{P}\left[c_i = c, \forall i \mid (c, c_1, \cdots, c_N) \sim \rho^{\otimes N+1} \right]$.

Ces résultats sont valables pour un grand nombre arbitraire de négatifs et sont découplés du nombre de tâches de classification. Ainsi, nous fournissons de nouvelles garanties théoriques pour la performance de classification des modèles entraînés de manière contrastive dans le cas de tâches de classification multivoie, en utilisant *multiple* échantillons négatifs. Nous étendons les résultats de Saunshi et al. [2019] pour montrer que la performance de l'entraînement non supervisé se reflète sur une tâche de classification ultérieure dans le cas de tâches multiples et lorsqu'un nombre élevé d'échantillons négatifs est utilisé.

Nous fournissons également un résultat de convergence (Théorème 3.5) pour un algorithme *explicite* (descente de gradient), lors de l'entraînement d'un réseau de neurones sur-paramétré pour l'apprentissage non supervisé de représentations contrastives.

Chapitre 4 : WildWood, un nouvel algorithme de forêt aléatoire

Dans le travail présenté dans le Chapitre 4, nous proposons WildWood, qui conserve tous les ingrédients de la forêt aléatoire (*Random Forest*, RF), y compris le bootstrap, le sous-échantillonnage des features, une procédure similaire de croissance des arbres, avec un prédicteur d'arbres amélioré et des calculs efficaces implémentés en `Python`.

Considérons un arbre $\mathcal{T}$ entièrement développé. Rappelons qu'en RF standard, le prédicteur d'arbre correspondant à $\mathcal{T}$ s'écrit $\widehat{f}(x) = \widehat{y}_{C_{\mathbf{v}}(x)}$ pour tout $x \in \mathcal{X}$, où $C(x)$ est la cellule contenant x associée à la feuille $\mathbf{v}$: il s'agit de trouver la plus petite cellule contenant x, et de produire la prédiction de cette cellule. Dans WildWood, nous améliorons ce prédicteur d'arbre par le mécanisme suivant : comme prédicteur pour $\mathcal{T}$, WildWood utilise

$$\widehat{f}(\cdot) = \frac{\sum_{T \subset \mathcal{T}} \pi(T) e^{-\eta L_T} \widehat{y}_T(\cdot)}{\sum_{T \subset \mathcal{T}} \pi(T) e^{-\eta L_T}} \quad \text{avec} \quad \pi(T) = 2^{-\|T\|}, \tag{4}$$

où la somme est calculée sur tous les sous-arbres T de $\mathcal{T}$ enracinés à `root`, $\eta > 0$ est le paramètre de température, $\|T\|$ désigne le nombre de noeuds dans T moins son nombre de feuilles qui sont également des feuilles de $\mathcal{T}$, L_T est la perte cumulative de la prédiction du sous-arbre T sur `oob` échantillons, *i.e.* $L_T \coloneqq \sum_{i \in I_{\texttt{oob}}} \ell(\widehat{y}_T(x_i), y_i)$, et $\widehat{y}_T$ est la prédiction du sous-arbre T de manière classique. La fonction de prédiction (4) peut être vue comme une agrégation des prédictions $\widehat{y}_T(\cdot)$ de tous les sous-arbres T, pondérées par leurs performances sur `oob` échantillons avec un prior $\pi(T) = 2^{-\|T\|}$. Il s'agit en effet d'une manière non-gloutonne d'élaguer les arbres : les poids ne dépendent pas seulement de la qualité d'une seule coupure, mais aussi de la performance de chacune des coupures successives.

En plus, le théorème suivant donne une garantie théorique sur les échantillons `oob`, pour l'agrégation de sous-arbres, sous une hypothèse sur l'exp-concavité de la fonction de perte. Ce théorème stipule que le prédicteur donné par (4) est capable de performer presque aussi bien que le meilleur sous-arbre oracle $T \subseteq \mathcal{T}$ sur les échantillons `oob`, avec un taux de $O(\|T\|/n_{\texttt{oob}})$ qui est optimal pour les inégalités oracle de sélection de modèle [Tsybakov, 2003] ; en comparaison, trouver un oracle $\operatorname{argmin}_{T \in \mathcal{T}} \sum_{i \in I_{\texttt{oob}}} \ell(\widehat{f}(x_i), y_i)$ est infaisable sur le plan computationnel, ce qui nécessite d'essayer tous les sous-arbres.

Theorem (Inégalité Oracle, Théorème 4.2 reformulé)**.** *Supposons que la fonction de perte ℓ soit η-exp-concave. Alors, la fonction de prédiction $\widehat{f}$ donnée par (4) satisfait l'inégalité oracle*

$$\frac{1}{n_{\texttt{oob}}} \sum_{i \in I_{\texttt{oob}}} \ell(\widehat{f}(x_i), y_i) \leqslant \inf_{T \subset \mathcal{T}} \left\{ \frac{1}{n_{\texttt{oob}}} \sum_{i \in I_{\texttt{oob}}} \ell(\widehat{y}_T(x_i), y_i) + \frac{C\|T\|}{\eta(n_{\texttt{oob}} + 1)} \right\},$$

où l'infimum est pris sur tous les sous-arbres $T \subset \mathcal{T}$ enracinés à `root`, et $C = \log 2$.

Une conséquence de ce théorème est un calcul efficace de $\widehat{f}(x)$, en exploitant le fait que les arbres dans WildWood sont développés en profondeur. Le calcul de $\widehat{f}(x)$ n'augmente en effet que d'un facteur 2 la complexité de calcul d'un RF standard.

En plus de ce calcul efficace de l'agrégation de sous-arbres, WildWood supporte nativement les caractéristiques catégorielles et implémente une stratégie d'histogramme, pour accélérer la recherche de fractionnement. Notre implémentation de WildWood est disponible sur le répertoire GitHub `https://github.com/pyensemble/wildwood`.

Nous avons fait des expériences extensives pour évaluer les performances prédictives et les temps d'exécution de notre implémentation de WildWood, voir les tableaux 4.1 et 4.2 du Chapitre 4 de ce manuscrit pour les paramètres et les résultats détaillés des expériences. Toutes les expériences peuvent être reproduites à l'aide de scripts `Python` sur le même répertoire. Dans l'ensemble, en tant qu'un algorithme Random Forest amélioré, WildWood a montré des performances compétitives en termes de pouvoir prédictif et de temps d'exécution, par rapport aux algorithmes Random Forest standard et Extreme Gradient Boosting.

Chapitre 5 : Régression logistique en ligne, vers un algorithme efficace avec une meilleure garantie de regret

Le point de départ du travail présenté dans le Chapitre 5 est la question suivante : avec éventuellement quelques adaptations, une version en ligne du SMP [Mourtada and Gaïffas, 2019] pourrait-elle atteindre une garantie de regret comparable à l'excès de risque du SMP ? Nous présentons deux algorithmes candidats, OSMP et AOSMP, ainsi que leur analyse du regret.

Commençons par considérer le regret λ-ridge pénalisé jusqu'au temps t pour la régression logistique en ligne, qui s'écrit

$$\sum_{s=1}^{t} \ell(\widehat{y}_s, y_s) - \inf_{\theta \in \Theta} \left(\sum_{s=1}^{t} \ell(\theta^\top x_s, y_s) + \lambda \|\theta\|^2 \right).$$

Nous introduisons le *One-Step Minmax Predictor* (OSMP), qui prend le "meilleur choix possible" de prédiction $\widehat{y}_t$ contre "le pire choix possible" de la vraie valeur $y_t \in \{-1, +1\}$ et du paramètre de l'adversaire θ, mesuré par le regret λ-ridge pénalisé ci-dessus, à savoir

$$\widehat{y}_t = \operatorname*{argmin}_{\hat{y} \in \mathbb{R}} \sup_{y_t \in \{-1,1\}} \sup_{\theta \in \mathbb{R}^d} \left\{ \ell(\hat{y}, y_t) + \widehat{L}_{t-1} - \left(\ell(\theta^\top x_t, y_t) + L_{\lambda, t-1}(\theta) \right) \right\},$$

avec $\widehat{L}_{t-1} \coloneqq \sum_{s=1}^{t-1} \ell(\widehat{y}_s, y_s)$, et $L_{\lambda,t-1}(\theta) \coloneqq \sum_{s=1}^{t-1} \ell(\theta^\top x_s, y_s) + \lambda \|\theta\|^2$. Nous montrons que de façon équivalente (Lemme 5.4), l'OSMP prédit

$$\widehat{y}_t = -L_{\lambda,t}^{+1\star} + L_{\lambda,t}^{-1\star}$$

avec $L_{\lambda,t}^{y\star} \coloneqq \inf_{\theta \in \mathbb{R}^d} \left\{ \ell(\theta^\top x_t, y) + L_{\lambda,t-1}(\theta) \right\}$ pour $y \in \{-1, +1\}$. Cela donne un algorithme explicite (l'Algorithme 1) pour le calcul de l'OSMP. Concernant l'analyse du regret de l'OSMP, nous écrivons le regret de la façon suivante

$$\mathrm{Regret}_n(\theta) = \sum_{t=1}^{n} \ell(\widehat{y}_t, y_t) - \sum_{t=1}^{n} \ell(\theta^\top x_t, y_t) \leqslant \lambda \|\theta\|^2 + \sum_{t=1}^{n} \widehat{r}_t,$$

où nous désignons $\widehat{r}_t \coloneqq \ell(\widehat{y}_t, y_t) - L_{\lambda,t}^{\star} + L_{\lambda,t-1}^{\star}$ le regret instantané du temps t. Nous déduisons la borne supérieure suivante

$$\widehat{r}_t \leqslant e \cdot \sigma'(\langle \theta_t, x_t \rangle) \cdot \|x_t\|^2_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}}$$

dont la preuve repose sur la self-concordance généralisée [Bach, 2010] de la fonction $L_{\lambda,t}$. Cependant, notre analyse de regret pour l'OSMP s'arrête là. En particulier, la valeur de la matrice hessienne $\nabla^2 L_{\lambda,t}(\theta_t)$ dépend du point θ_t où elle est évaluée, et nous ne connaissons pas de moyen de contrôler une telle matrice hessienne conduisant à une borne supérieure de regret qui est logarithmique en n et sans facteur exponentiel en BR, voir la Section 5.5.3 pour plus de détails.

Pour contourner cette difficulté, nous introduisons le *Approximated One-Step Minmax Predictor* (AOSMP) suivant, défini par

$$\widehat{y}_t = \operatorname*{argmin}_{\widehat{y}\in\mathbb{R}} \sup_{y_t\in\{-1,+1\}} \sup_{\theta\in\mathbb{R}^d} \left\{ \widehat{L}_{t-1} + \ell(\widehat{y}, y_t) - \left(\ell(\theta^\top x_t, y_t) + \tilde{L}_{t-1}(\theta) + \lambda \|\theta\|^2 \right) \right\}$$

avec $\tilde{L}_{\lambda,t-1}(\theta) \coloneqq \sum_{s=1}^{t-1} \tilde{\ell}(\theta^\top x_s, y_s) + \lambda\|\theta\|^2$. Cette formulation étant dans le même esprit minmax que l'OSMP, la différence est que dans l'AOSMP, nous remplaçons la fonction L_{t-1} par son approximateur quadratique $\tilde{L}_{t-1}$. Nous nous appuyons sur l'approximation quadratique de la fonction logistique introduite pour la première fois dans Jézéquel et al. [2020, Lemma 5], à savoir

$$\tilde{\ell}_t(\theta) \coloneqq \ell_t(\tilde{\theta}_t) + g_t^\top(\theta - \tilde{\theta}_t) + \frac{1}{2}\eta_t \left(x_t^\top(\theta - \tilde{\theta}_t) \right)^2,$$

avec $g_t \coloneqq \nabla \ell_t(\tilde{\theta}_t)$, $\eta_t \coloneqq \sigma'(\langle \tilde{\theta}_t, x_t \rangle)/(1 + BR)$ et nous choisissons

$$\tilde{\theta}_t = \operatorname*{argmin}_{\theta\in\mathbb{R}^d} \tilde{L}_{\lambda,t}^{y_t}(\theta) \quad \text{avec} \quad \tilde{L}_{\lambda,t}^{y}(\theta) \coloneqq \ell(\theta^\top x_t, y) + \tilde{L}_{\lambda,t-1}(\theta).$$

Pour les mêmes raisons que pour l'OSMP, l'AOSMP prédit

$$\widehat{y}_t = -\tilde{L}_{\lambda,t}^{+1\star} + \tilde{L}_{\lambda,t}^{-1\star}$$

avec $\tilde{L}_{\lambda,t}^{y\star} \coloneqq \inf_{\theta\in\mathbb{R}^d} \left\{ \ell(\theta^\top x_t, y) + \tilde{L}_{\lambda,t-1}(\theta) \right\}$ pour $y \in \{-1, +1\}$, et cela donne un algorithme explicite (Algorithme 2). Ensuite, nous sommes en mesure d'appliquer les techniques classiques avec les formes quadratiques pour borner la somme des pseudo regrets instantanés, pour obtenir la borne suivante (Theorème 5.19) sur le regret de l'AOSMP

$$\mathrm{Regret}_n(\theta) \leqslant e \cdot (1 + BR) d \log\left(1 + \frac{nR^2}{8d(1 + BR)\lambda} \right) + \lambda \|\theta\|^2.$$

En particulier, le choix $\lambda = R^2$ donne

$$\mathrm{Regret}_n \leqslant e \cdot (1 + BR) d \log\left(1 + \frac{n}{8d(1 + BR)} \right) + B^2 R^2.$$

Ainsi nous obtenons pour l'AOSMP une borne supérieure de regret similaire à celle d'AIOLI [Jézéquel et al., 2020], en partie parce que nous utilisons la même approximation quadratique.

Globalement, dans ce travail, nous proposons et analysons deux algorithmes inspirés du SMP [Mourtada and Gaïffas, 2019] pour la régression logistique en ligne : pour OSMP, nous avons seulement trouvé une limite supérieure pour le regret instantané $\widehat{r}_t \leqslant e \cdot \sigma'(\langle \theta_t, x_t \rangle) \cdot \|x_t\|^2_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}}$; pour AOSMP, nous avons prouvé une limite supérieure de regret dans un ordre similaire avec l'algorithme de pointe [Jézéquel et al., 2020] pour la régression logistique binaire en ligne au meilleur de notre connaissance. La question d'un algorithme efficace avec une meilleure limite supérieure de regret pour la régression logistique en ligne reste ouverte.

Chapter 1

Introduction

Contents

This thesis deals with the general topic of machine learning, a general framework for the problem of retrieving information and making decisions from data. While the recent decade has seen the revival of neural networks re-branded as deep learning [LeCun et al., 2015] and its many successful real-world applications with the help of larger datasets and stronger computational capacities, methodological research in machine learning is more than ever crucial to the progress of such a field, enabling the design of new algorithms with desirable properties as well as the deeper understanding of some practical success.

In this thesis, I worked on the applications of deep learning on public health data, and also on theoretical and numerical studies of some machine learning problems. The first contribution consists of leveraging the medical claims data available in French national system of health data using deep learning (Chapter 2). Then, the contributions are focused on the topics of contrastive learning (Chapter 3), ensemble tree-based methods (Chapter 4), and online logistic regression (Chapter 5).

This introduction chapter gives a quick overview for each of these works. In the rest of this opening chapter, we give a presentation of each of the problems that appears in this thesis, as well as their background and some necessary tools, followed by a brief summary of the contributions of the present thesis.

1.1 Deep Learning and applications in healthcare

Healthcare is an important application area of artificial intelligence and machine learning, with profound consequence and significant potential benefice for social good [Bommasani et al., 2021; Villani et al., 2018]. At the same time, developing efficient algorithms to learn appropriate representations of structured data, including sequences of timestamped events, is a major yet challenging problem in machine learning. As deep learning has shown impressive performance in many application domains (Computer Vision [Chen et al., 2020; He et al., 2020; Krizhevsky et al., 2012], Natural Languages Processing [Brown et al., 2020; Devlin et al., 2018]), the question of leveraging patient-centered healthcare records available in large observational databases using such techniques naturally become of great interest.

1.1.1 Large observational databases in healthcare: EHRs, medical claims data

In the domain of healthcare, with a fast adaptation towards information systems in the past decades, it became possible to collect and store large amounts of healthcare data, which are rich in information and sensitive by their nature. These large amounts of patient-centered data were made available through some different stakeholders in the domain of healthcare providers: clinics and hospitals, insurance companies, public institutions. This broad kind of large observational databases are the type of data that we are interested in this work.

We can roughly distinguish between two categories of healthcare data that are acquired by re-purposing administrative ones: *Electronic Health Records* (EHRs) and *medical claims data*. These two types of data share lots of similarities with some specificities [FDA, 2021], while sometime confusions are made in using the two terms.

Under the name of EHRs are generally designed data directly produced by healthcare providers, such as hospitals and clinics, for clinical care support and

billing purpose. EHRs are often fine-grained, including demographic information such as gender, age, location, sometimes also patients' living habits such as smoking status, alcohol consumption and medical history. Most importantly, EHRs often come with exact information on provided cares, such as drug prescriptions, medical procedures, diagnoses, bed-side vital signs in case of Intensive Care Units (ICU), sometimes including also free-text medical reports, laboratory analysis including for example imaging and imaging reports, and other clinical observations and reports. A freely accessible database of ICU data and hospital information MIMIC-III [Johnson et al., 2016] is often seen as a reference for EHRs, in particular for benchmarking works [Bellamy et al., 2020; Harutyunyan et al., 2019]. The recently released successor MIMIC-IV [Johnson et al., 2020] includes more complex and more complete data such as emergency department data, chest X-Ray images and free-text clinical notes.

Medical claims data are collected by insurance companies or agencies (thus indirectly from healthcare providers), with primarily administrative purpose of reimbursement. Medical claims data are often in form of timestamped events, *i.e.* drug prescriptions, medical procedures, hospital diagnoses. While medical claims data share many similar characteristics with EHRs, and often have the advantage of covering a larger number of population across numerous hospitals or healthcare providers, medical claims data often contain fewer details and may be less accurate than EHRs. For example, bed-side vital signs could rarely be included in medical claims data; the exam results of medical imaging is not contained, but the code for this imaging with indications on the body part is included in claims data (since this is required for reimbursement).

In France, the SNDS (National Health Data System – *Système National des Données de Santé*, formerly known as SNIIR-AM – National Health Insurance Information System – *Système National d'Information Inter-Régimes*), is an example of medical claims data [Tuppin et al., 2017]. It includes healthcare reimbursement records for nearly all French residents. Through the partnership between *Caisse Nationale de l'Assurance Maladie* (the French agency managing the national health insurance system – CNAM), and Data Science Initiative from École polytechnique, we had the opportunity to have access to an extract of the SNDS data with the help of SCALPEL3 [Bacry et al., 2020]. This database is at the core of our work presented in Chapter 2.

Challenges and opportunities with EHRs and medical claims data

Before going into details with SNDS, let us first describe in a non-exhaustive way some practical challenges relating to the healthcare data.

- Practitioners should pay particular attention on the **data quality**. As the primary goal for both EHRs and medical claims data is to assist the delivery of care and the billing, there could be mistakes in data entry, and biases from data collection. The data extraction step might also introduce some mistakes or biases.

- **Benchmarking** and **reproducibility** are important for the scientific community [Bellamy et al., 2020] to be able to compare across different datasets or different models, despite the sensitive nature of the healthcare data and possibly some compliance rules for in-house datasets.

- Although the predictive performance could be convincing for its own in some cases, models with **interpretability** are often preferred [Luo et al., 2019; Shickel et al., 2017; Xiao et al., 2018] in the domain of healthcare.
- **Data privacy** is particularly important when it comes to healthcare data. Effective de-identification of patient data is a crucial step before any research: this is indeed the case for the SNDS [Tuppin et al., 2017].

Furthermore, some specific attention should be paid in works involving EHRs and medical claims data, in particular in the modelling of the healthcare pathways from these data.

- As a particular form of **structured data** with sequences of timestamped events, such as drugs prescriptions, medical procedures, hospital diagnoses, EHRs and claims data can also arise patient demographic information, medication dosage, etc. It is vital that the designed model is adapted to these specificities. Some practical examples includes, how to take into account medication dosage information alongside the drug in form of its CIP-13 codes? How to take into account the patient age information, as we know the age itself can tell a lot on the health status?
- **Modelling the sequences of timestamped events** is a crucial problem. Designing models that are adapted to data structure is important, in particular when in comes to deep neural networks models. Some recent research [Li et al., 2020] draw the parallel to NLP, in which case the medical codes are viewed as "words" such that the encounters or patients are viewed as "sentences" or "documents". Besides, the irregularity of the timestamps for the medical events that occur on the patient timeline, and the aggregation at different levels of the patient timeline, are some additional difficulties to this modelling problem.
- Learning appropriate and useful **representations** for medical codes is an important step, and can be a goal itself. Since there is often no explicit label in EHRs and medical claims data, data preprocessor may need to look at long term drug prescription history to be able to deduce some long-term diseases such as diabetes [Morel et al., 2019]; another direction is to design surrogate tasks such as masked prediction [Li et al., 2020]. Next, efficiently building representations for healthcare status from learned medical codes representations is also challenging.
- Effective **evaluation** is critical when it comes to assess models and pipelines. In terms of prediction targets, while predicting the mortality and the length-of-stay are some classical targets in EHRs ICU data [Rajkomar et al., 2018], disease detection or classification [Li et al., 2020], sequential prediction of medical events are also popular tasks on both EHRs and medical claims data. Most of these tasks, if not all, require careful and proper data processing to avoid that the features indirectly contain the labels. The lower-dimensional projection and visualization of learned representations for the medical codes might be helpful for qualitative assessment.

However, the research interest on EHRs and medical claims data is growing, especially using deep learning models recently. The reason is first and foremost for

the applications opportunities in such a field, as well as enormous social impact it involves [Bommasani et al., 2021; Villani et al., 2018]. Also, efficiently and successfully modelling the sequences of timestamped events can be of interest for other situation other than the healthcare, for example user behavior modelling.

The French national system of health data (SNDS)

SNDS gathers reimbursement data from most state health insurance schemes, and aggregates data from multiple authorities such as hospitals and local agencies. At its creation in 2016, SNDS contained health reimbursements of 66 million French residents, representing 98.8% of the country's population [Tuppin et al., 2017]. Although SNIIR-AM was initially used to monitor health expenditures and to evaluate health care utilization across the country, from 2016, some drug safety studies [Bezin et al., 2017] and epidemiological studies [Aguade et al., 2020; Feldman et al., 2021] have been conducted on SNDS thanks to individual data availability.

The quality of SNDS data results from mandatory logging of reimbursed care in France, three data validation stages, and pseudonymization routines. Thanks to its history length, high population coverage, and quality, SNDS can be used to conduct epidemiological studies with a high statistical power and is almost exempt of representativity biases [Neumann et al., 2012; Tuppin et al., 2017].

There are two data sources for SNDS: DCIR (Inter-scheme consumption data — *Données de Consommation Inter-Régimes*) contains outpatients billing and reimbursement information, PMSI (medical information system program — *Programme de Médicalisation des Systèmes d'Information*) gathers hospital stay data.

DCIR contains demographic information of the beneficiaries (date of birth, gender, town of residence, and variables indicating whether patients are beneficiaries of specific social subsidies), and information on their possible disabilities or long-term diseases. More importantly, DCIR provides timestamped reimbursement information concerning drug purchases (drugs are coded with the ATC (Anatomical Therapeutic Chemical) classification system), medical procedures (coded with CCAM – the French medical procedures classification – *Classification Commune des Actes médicaux*), laboratory analyses (coded with NABM – the classification of clinical pathology procedures – *nomenclature des actes de biologie médicale*), and medical products (LPP – the list of product and services – *Liste des produits et prestations*).

PMSI is made up of four databases: MCO (acute care ward — *Médecine, Chirurgie, Obstétrique et Odontologie*), SSR (rehabilitation care – *Soins de Suite et Réadaptation*), HAD (home care — *Hospitalisation À Domicile*), and PSY (psychiatric care). These databases contain pseudonymized hospital stays summaries, *i.e.* starting and ending dates of hospital stays, diagnoses (coded with ICD-10 – the International Classification of Diseases, 10th revision), and medical procedures (coded with CCAM). In contrast to some well-know EHRs (for example MIMIC-III [Johnson et al., 2016]), a difficulty with PMSI is that it comes from hospital stay summaries hence there is no information regarding the order or the temporality of the events, but only start-end dates with all the events happening throughout one hospital stay. PMSI also contains similar timestamped information for outpatient consultations.

Accessing to SNDS data requires authorization from the CNIL (*Commission Nationale de l'Informatique et des Libertés*), the French data protection authority, according to legal compliance and public interest. Recently, the *Health Data*

Hub [Cuggia et al., 2018], a new governmental agency, was created. The aim is to make methodological research easier on SNDS and similar data, by centralizing health data, and by providing accesses with hardware and software infrastructure, at the same time under CNIL's compliance. Our extracted dataset is built upon the join and extraction work by the SCALPEL3 framework [Bacry et al., 2020].

1.1.2 Contribution: ZiMM, a deep learning model for long term and blurry relapses with non-clinical claims data

The work presented in Chapter 2 is an attempt to address some challenges described in the previous Section 1.1.1, in form of a study on a concrete medical example using recent tools from the deep learning.

In this work, we consider the problems of modeling and predicting a long-term and "blurry" relapse that occurs after a medical procedure, such as a surgery. We do not consider a short-term complication related to the procedure itself, but a long-term relapse that clinicians cannot explain easily, since it depends on unknown sets or sequences of past events that occurred before the procedure. The relapse is observed only indirectly in a "blurry" fashion, through longitudinal prescriptions of drugs over a long period of time after the medical procedure.

As stated previously, our motivation for using SNDS, a non-clinical medical claims database, is its exhaustivity population-wise, compared to EHRs coming from a single or a small set of hospitals. Indeed, we consider an extracted dataset containing the medical claims of almost all French residents who had surgery for prostatic problems, with a history between 1.5 and 5 years.

Patient timeline, study cohort, labels

A patient with several types of events is illustrated in Figure 1.1 using a timeline representation. This can be considered as a sequence of timestamped events $z_k^i = (d_k^i, t_k^i)$ with d_k^i medical code that happened as a medical event on the patient i at time t_k^i.

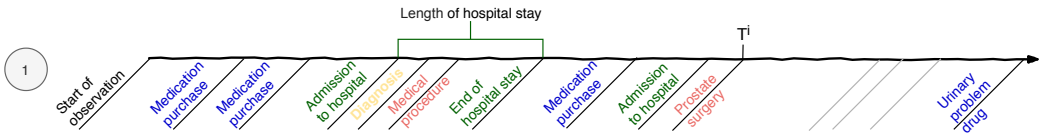

Figure 1.1 – Illustration of the sequence of claims observed for a patient. Through claims, we observe three types of events: drug purchases (blue), medical procedures (red) and diagnosis (yellow) before the medical act (prostate surgery in the example) that happens at time T^i for patient i. All events are timestamped, and the time delta is a day long. Several events can occur the same day and some days have no event: events are typically very irregularly sampled. All these events, observed before T^i, are used to learn the embedding vector $x_i \in \mathbb{R}^d$ of patient i. After T^i, we only keep the events corresponding to the blurry relapses considered (drug purchases among a set of drugs for urinary problems). These blurry relapses are used to build the label vector $y_i \in \mathbb{N}^B$ of patient i.

We extract patients with transurethral resection of the prostate (TURP) surgery in SNDS, through some specific CCAM codes provided by clinicians[1]. Some ad-

[1]We considered that a TURP surgery corresponds to the CCAM codes `JGFA005`, `JGFA009`, `JGFA015`, `JDPE002` or `JGNE003`.

ditional inclusion and exclusion criteria are applied to ensure data quality; in the main study, the final cohort includes 138 976 patients.

As far as the labels (post-surgical urinary problems) are concerned, a simple but efficient way to identify whether the urinary problems have not ceased or reappeared after a while is to see if the patient, at some point after the surgery, needs to take medications for these urination problems again. For that purpose, we use a list (provided by clinicians) of 136 CIP-13 drugs that are mainly related to urination problems. We choose to drive the prediction on a 18-months period after T^i. In other words, we chose the number of buckets $B = 18$ and bucket size of 30 days (540 days total).

Zero-inflated Multinomial Mixture model

Here we consider a long-term (18 months) relapse (urination problems still occur despite surgery), which is blurry since it is observed only through the reimbursement of a specific set of drugs for urination problems. To best capture long-term and blurry relapses, we introduce *Zero-inflated Mixture of Multinomial distributions model* (ZiMM).

We denote by $n_i = \sum_{b=1}^{B} y_{i,b}$ the overall number of blurry relapses of patient i. The following model is proposed to model the whole vector y_i, so that n_i is not fixed and includes zero-inflation, namely a parametrized likelihood for $n_i = 0$. We suppose that $n_i \in \{0, 1, \dots, B\}$ is distributed (conditionally to x_i) as

$$n_i \sim \text{Categorical}(\pi_0(x_i), \pi_1(x_i), \dots, \pi_B(x_i)),$$

which means that

$$\mathbb{P}(n_i = k | x_i) = \pi_k(x_i) \text{ for } k \in \{0, \dots, B\},$$

where $\pi_b(x_i)$ are such that $\sum_{b=0}^{B} \pi_b(x_i) = 1$ and $\pi_b(x_i) \geqslant 0$ (coming out of a softmax activation for instance). These parameters correspond to the categorical distribution specific to patient i. Then, we assume that the distribution of y_i conditional to $n_i = b$ and x_i follows either a Dirac distribution on vector (of size B) $[0, \dots, 0]$ whenever $b = 0$, or a multinomial distribution of parameters b and $p_{b,1}(x_i), \dots, p_{b,B}(x_i)$, namely

$$y_i | (x_i, n_i = b) \sim \begin{cases} \delta_{[0,\dots,0]} & \text{if } b = 0, \\ \text{Multinomial}(b, p_{b,1}(x_i), \dots, p_{b,B}(x_i)) & \text{otherwise,} \end{cases}$$

where $p_{b,1}(x_i), \dots, p_{b,B}(x_i)$ are the parameters of a multinomial distribution whenever $n_i = b$, for each $b = 1, \dots, B$. Once again, these parameters are specific to patient i.

On top of this probabilistic modelling of post-surgical complication, we build an end-to-end deep-learning architecture called ZiMM Encoder-Decoder (ZiMM ED) that can learn from the complex, irregular, highly heterogeneous and sparse patterns of health events that are observed through a claims-only database.

ZiMM Encoder

Codes are first tokenized, and each unique token is individually mapped to an embedding vector in $\mathbb{R}^{d_E}$ which is learned during training. We consider only tokens

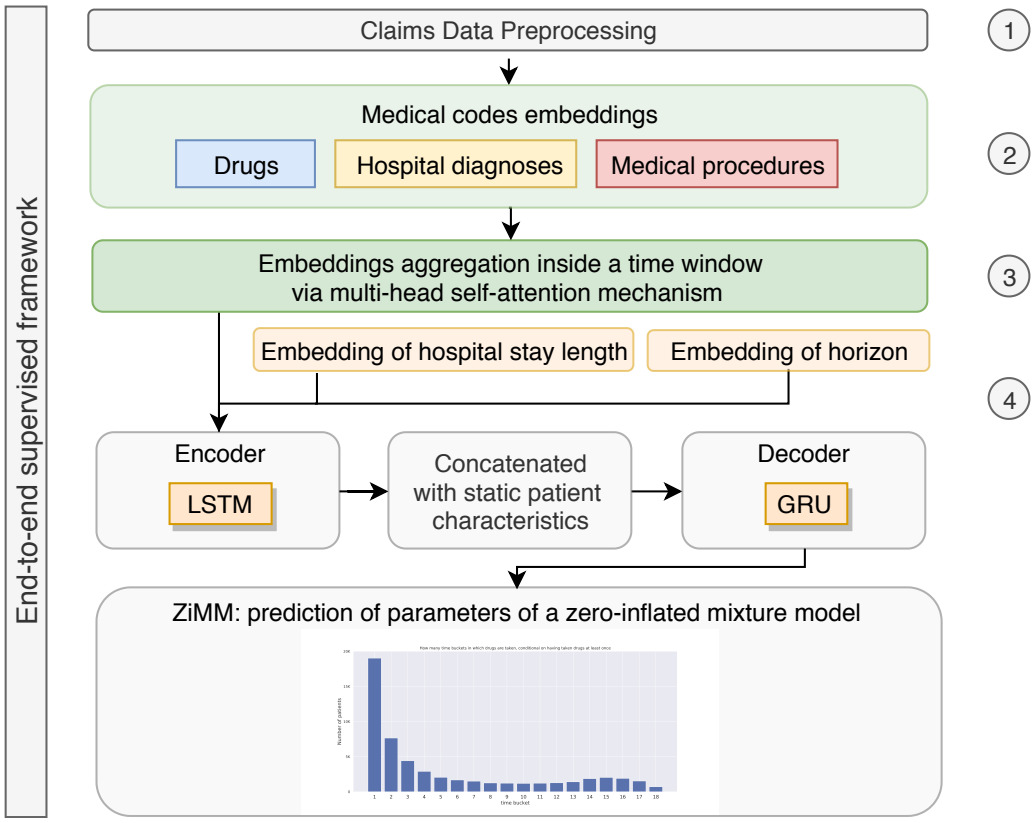

Figure 1.2 – ZiMM Encoder-Decoder end-to-end architecture.

that occur at least 50 times in the training dataset. For any event occurring at time $t \leqslant T^i$ in the observation period of patient i, we compute the "time horizon" as $T^i - t$, namely the distance (in days) between the event and medical act. Moreover, whenever it makes sense, we compute end $-$ start, the duration of the event, which corresponds to the duration of an hospital stay defined as the time between hospital admission and discharge. These two integers (in number of days) are also tokenized and replaced by a learned embedding vector. This allows the encoder to learn to put more or less emphasis on events that are close or far from T^i, and to exploit the duration of events as a proxy for severity of medical procedures and diagnoses. The patient age in years at T^i is also similarly bucketized and embedded.

Then, several events can occur at the same time (within the same day), so that the number of codes observed within a day is highly heterogeneous. Moreover, such codes are not likely to contribute equally to the vector representation of the day, and their order is not informative.

Hence, we use a self-attention mechanism [Lin et al., 2017; Vaswani et al., 2017] using a *bag-of-features* approach to learn how to combine embedding vectors within the same day, following previous successful applications of self-attention for fusing disease embeddings [Luo et al., 2019].

In the following, we obtain sequence of fixed-sized embedding vectors that encode both medical and time information at each (non-empty) timestamp $t \leqslant T^i$, as displayed in Step (4) of Figures 1.2. Now, this sequence of vectors is used as the input of a stack of layers, including recurrent layers (LSTM [Hochreiter and Schmidhuber, 1997], bi-directional LSTM [Schuster and Paliwal, 1997], GRU [Cho

et al., 2014]) or convolutional layers, in a *sequence-to-one* network architecture, since we want to output a single vector $x_i \in \mathbb{R}^d$ that encodes the full pathway of a patient before T^i.

Finally, the output vector of the encoder is concatenated with an embedding vector of the age of the patient, leading to the final vector $x_i \in \mathbb{R}^d$ that encodes the full pathway of patient i, representing her health status in a whole, serving as input to the following decoder.

ZiMM Decoder

The decoder uses vector $x_i \in \mathbb{R}^d$ representation of health status of patient i to construct the parameters of the ZiMM model, and the whole architecture is trained against the negative log-likelihood of the ZiMM model computed at the label vector $y_i = [y_{i,1}, \ldots, y_{i,B}]$ containing the blurry relapses. Since parameters $\{\pi_b(x_i)\}_{b=1\ldots,B}$ and $\{p_{b,b'}(x_i)\}_{b,b' \in \{1,\ldots,B\}^2}$ are highly dependent and are time-ordered, a specific architecture is used to model these dependencies: fully connected feed-forward layer (FFN) followed by a recurrent layer (RNN). Finally, a softmax activation is applied on h_t^b along $t = 1, \ldots, B$ to produce the parameters $p_{b,1}(x_i), \ldots, p_{b,B}(x_i)$.

Experiments

Our experiments show that ZiMM ED improves several baselines, including non-deep learning and deep-learning approaches, and that it allows working on such a dataset with minimal preprocessing work.

Table 1.1 – Predictive performances (on test data) of benchmark models and ZiMM ED architecture. ZiMM ED appears to perform the best among all models both for multi-output and binary prediction.

Model	mean-AP	AUC-ROC	AUC-PR
LRl2-SF	0.19	0.64	0.50
GBDT-SF	0.24	0.67	0.56
MLP-SF	0.18	0.64	0.49
LRl2-DF	0.21	0.65	0.53
GBDT-DF	0.25	0.68	0.57
MLP-DF	0.19	0.65	0.50
Word2vec-ISS	0.20	0.65	0.53
LSTM-ISS	0.21	0.67	0.54
Patient2Vec	-	0.68	0.55
ZiMM ED	**0.306**	**0.701**	**0.619**

We implemented ZiMM under `Tensorflow2`. All our experiments use the same random data splitting into 70% of patients for training, 15% of patients for validation and 15% for testing. As for evaluation metrics, we use *mean-AP*, defined as the average of the area under the precision-recall curve (AUC-PR), namely we compute the average over the buckets $b = 1, \ldots, B$ of the AUC-PR for each bucket b, i.e., the AUC-PR of the prediction of $y_{i,b}$. Moreover, we report also the AUC-PR and the AUC-ROC (area under the ROC curve) for the binary classification problem $n_i > 0$ against $n_i = 0$.

The predictive performance of all benchmarks and of ZiMM ED are presented in Table 1.1, where we report mean-AP scores on the test set, as well as AUC-PR and AUC-ROC scores for the binary classification problem. GBDT-based models on dynamic features (GBDT-DF) performs the best among all benchmark models, however the ZiMM ED architecture outperforms all the benchmark models both for the multi-output y_i prediction and for the binary prediction.

In the ablation study, we modify some hyperparameters or change some components of ZiMM ED default, and report the impacts on performances in details in Table 2.4, where the first line reports the performances of ZiMM ED default. And we made careful discussion around each component of the model.

To wrap up, the work presented in Chapter 2 addresses the problem of predicting the blurry relapses of the TURP surgery, which is the first step towards an evidence-based approach using machine-learning to help the clinical decision.

1.2 Contrastive learning, unsupervised learning and supervised learning

The motivation of the work presented in Chapter 3 is to better understand the recently popular unsupervised and self-supervised training techniques, from a modelling point of view.

The problem of learning appropriate and useful representations for words and sentences naturally arises in languages modelling, in the unsupervised setting. Let us trace the line of research of the recent successes of contrastive learning.

The skip-gram [Mikolov et al., 2013a] model considers the conditional probabilities $p(c \mid w)$ where w stands for a word and c stands for a context. Denote θ the parameter for $p(c \mid w; \theta)$, then the goal is to find θ that maximizes the corpus probability

$$\prod_{w} \prod_{c \in C(w)} p(c \mid w; \theta) = \prod_{(w,c) \in D} p(c \mid w; \theta)$$

where $C(w)$ denotes the set of all contexts related to word w, D denotes the set of all context and word pairs. Extracting such pairs (w, c) from a text is natural and easy. Denote v_c and v_w the vectorized representations (or *embeddings*) for a context c and a word w respectively. The neural networks language modelling literature usually models the conditional probability using the softmax function, namely

$$p(c \mid w; \theta) = \frac{\exp(v_c^\top v_w)}{\sum_{c' \in C} \exp(v_c'^\top v_w)}$$

where C denotes the set of all available context. Taking the log and combining both expressions leads to

$$\sum_{(w,c) \in D} \log p(c \mid w; \theta) = \sum_{(w,c) \in D} \left(v_c^\top v_w - \log \sum_{c' \in C} \exp(v_c'^\top v_w) \right).$$

Practitioners believe [Goldberg and Levy, 2014; Mikolov et al., 2013b] that training neural networks with the objective above will result in good words embeddings, in the sense that *similar words that occur in similar contexts, will have similar embeddings*. A downside of skip-gram is that it is expensive to compute the denominator term in the softmax, as well as the term $\log \sum_{c' \in C} \exp(v_c'^\top v_w)$, as it involves to sum over all possible contexts $c' \in C$.

Based on this observation, the *negative sampling* [Mikolov et al., 2013b] aims to derive meaningful embeddings while being computationally efficient. Consider $p(P = 1 \mid w, c; \theta)$ the probability that a pair (w, c) actually comes from the corpus text, for any word w and any context c; and $p(P = 0 \mid w, c; \theta)$ the probability that a pair (w, c) actually does not come from the corpus text. These probabilities are modeled with the sigmoid function $\sigma(u) = 1/(1 + \exp(-u))$, with

$$p(P = 1 \mid w, c; \theta) = \sigma(v_c^\top v_w) = 1 - p(P = 0 \mid w, c; \theta).$$

Then, the approach aims to maximize the following quantity, for any pair (w, c) in corpus,

$$p(P = 1 \mid w, c; \theta) \prod_{(w,c_i) \notin D} p(P = 0 \mid w, c_i; \theta).$$

Taking the log, it yields

$$\log \sigma(v_c^\top v_w) + \sum_{(w,c_i) \notin D} \log \sigma(-v_{c'}^\top v_w).$$

With some practical tricks such as dynamic window size, down-sampling of frequent words, rare words pruning, this approach is experimentally shown [Mikolov et al., 2013b] to be efficient to learn words embedding from text.

Several recent works further extend the idea of using naturally pair occurrence to unsupervisedly learn representations. Contrastive Predictive Coding (CPC) [Oord et al., 2018] learns representations of objects that natural lie on a sequence, by predicting future observations using a contrastive loss. The idea is to maximize the mutual information

$$I(x, c) = \sum_{x,c} p(x, c) \log \frac{p(x \mid c)}{p(x)}$$

between the encoded representations of (future) target x and (present) context c. The objective of CPC is the following contrastive loss, also called InfoNCE,

$$\mathcal{L} = \mathbb{E}_X \left[-\log \frac{f_k(x_{i,t+k}, c_{i,t})}{\sum_{x_j \in X} f_k(x_j, c_{i,t})} \right]$$

where $X = \{x_1, \ldots, x_N\}$ is a set of N random samples containing one positive sample from $p(x_{t+k} \mid c_t)$ and $N - 1$ negative samples from the distribution $p(x_{t+1})$; and $f_k(x_{t+k}, c_t)$ stands for the density ratio between x_{t+k} and c_k, $f_k(x_{t+k}, c_t) \propto p(x_{t+k} \mid c_k)/p(x_{t+k})$, modeled by

$$f_k(x_{t+k}, c_t) = \exp(g_{\text{encoder}}(x_{t+k})^\top W_k c_k)$$

where $W_k c_k$ denote a linear transformation of c_k. CPC has practical success in audio, vision and natural languages [Oord et al., 2018].

In particular, combined with a pre-training strategy [Hénaff et al., 2020], representations learned with the contrastive loss in an unsupervised fashion are shown to be useful for some down-stream classification tasks. Hénaff et al. [2020] uses a pretrained image encoder followed by simple classification layers, that are trained on a fraction of the labels available, allowing to achieve an accuracy comparable to that of a fully supervised end-to-end training. In the same pre-training – fine-tuning fashion with contrastive loss during the pre-training phase, MoCo [He et al., 2020],

SimCLR [Chen et al., 2020], SwAV [Caron et al., 2020], BYOL [Grill et al., 2020] are some improvements to successfully learn visual representations.

Let us summarize the common pre-training approach: provided a dataset, an encoder is trained using a contrastive loss whose minimization allows to learn encoders giving embeddings that are *similar* for pairs of samples (called the *positives*) that are close to each other (such as pairs of random data augmentations of the same image, see Chen et al. [2020]; He et al. [2020]), while such embeddings are *contrasted* for dissimilar pairs (called the *negatives*).

Although Saunshi et al. [2019] proposed a formalism together with results on classification performance based on unsupervisedly learned representation, these results do not explain the performance gain that is observed empirically [Chen et al., 2020; He et al., 2020] when a high number of negative samples are used. On the other hand, Allen-Zhu et al. [2019] proved that, over sufficient over-parametrization of deep neural networks, the optimization with GD and SDG of the ℓ^2-loss converges, as well as for RNN.

However, despite growing efforts [Saunshi et al., 2019; Wang and Isola, 2020], few theoretical results have been obtained as of the time of our work. For instance, there is no clear theoretical explanation of how a supervised task could benefit from an upstream unsupervised pre-training phase, or what could be the theoretical guarantees for the convergence of the minimization procedure of the contrastive loss during this pretraining phase.

1.2.1 Contribution: theoretical guarantees on the contrastive unsupervised representation learning for classification and on its convergence

Following the framework first proposed in Saunshi et al. [2019], assume that we are able to sample *positive* pairs (x, x^+) from the distribution

$$\mathcal{D}_{\text{sim}}(x, x^+) = \sum_{c \in \mathcal{C}} \rho(c) \mathcal{D}_c(x) \mathcal{D}_c(x^+), \tag{1.1}$$

namely, (x, x^+) is sampled as a mixture of independent pairs conditionally to a shared latent class, sampled according to ρ. On the other hand, we assume that we can sample *negative* samples x^- from the distribution

$$\mathcal{D}_{\text{neg}}(x^-) = \sum_{c \in \mathcal{C}} \rho(c) \mathcal{D}_c(x^-). \tag{1.2}$$

Under the assumptions given in Equations (1.1) and (1.2), we consider the following unsupervised contrastive loss with N negative samples,

$$L_{\text{un}}^N(f) = \mathbb{E}_{\substack{(x,x^+)\sim\mathcal{D}_{\text{sim}} \\ X^-\sim\mathcal{D}_{\text{neg}}^{\otimes N}}} \left[-\log \left(\frac{\exp\left(f(x)^\top f(x^+)\right)}{\exp\left(f(x)^\top f(x^+)\right) + \sum_{x^- \in X^-} \exp\left(f(x)^\top f(x^-)\right)} \right) \right], \tag{1.3}$$

where $\mathcal{D}_{\text{neg}}^{\otimes N}$ stands for the N tensor product of the $\mathcal{D}_{\text{neg}}$ distribution given by Equation (1.2).

Proposition (Proposition 3.3)**.** *Consider the unsupervised loss $L_{\text{un}}^N(f)$ from Equation (1.3) with N negative samples. Assume that ρ is uniform over $\mathcal{C}$ and that*

$2 \leqslant k+1 \leqslant N_{\mathcal{C}}$. *Then, any encoder function* $f : \mathcal{X} \to \mathbb{R}^d$ *satisfies*

$$L_{\text{sup},k}(f) \leqslant L^{\mu}_{\text{sup},k}(f) \leqslant \frac{k}{1-\tau^+_N}\left(L^N_{\text{un}}(f) - \tau^+_N \log(N+1)\right)$$

with $\tau^+_N = \mathbb{P}\left[c_i = c, \forall i \mid (c, c_1, \cdots, c_N) \sim \rho^{\otimes N+1}\right]$.

These results hold for an arbitrary large number of negatives and decoupled from the number of classification tasks. We provide new theoretical guarantees for the classification performance of contrastively trained models in the case of multiway classification tasks, using *multiple* negative samples. We extend results from Saunshi et al. [2019] to show that unsupervised training performance reflects on a subsequent classification task in the case of multiple tasks and when a high number of negative samples is used.

Also, we provide a convergence result (Theorem 3.5) for an *explicit* algorithm (gradient descent), when training overparametrized deep neural network for unsupervised contrastive representation learning. We explain how results from Allen-Zhu et al. [2019] about training convergence of overparametrized deep neural networks can be applied to a contrastive learning objective.

Let us remark that since the time of this work, theoretical foundations for contrastive learning and self-supervised learning is an extremely active and recently competitive research sub-area [Chuang et al., 2020; Tian et al., 2021; Tosh et al., 2021; Wei et al., 2020].

1.3 From Random Forest to WildWood

In the work presented in Chapter 4, we focus on ensemble tree-based algorithms. Under the batch supervised learning setting, data comes as a set of i.i.d. samples (x_i, y_i) for $i = 1, \ldots, n$, with features $x_i \in \mathcal{X}$ and labels $y_i \in \mathcal{Y}$. We consider quantitative and categorical features.

1.3.1 Random Forest: general principles

Let us first provide some background on Random Forest [Breiman, 2001], an important building block of WildWood.

Ensemble methods

In machine learning, under the name of *ensemble methods*, we refer to the general procedures that generate multiple predictors then combine them into one of better quality. The two principal families in the ensemble methods are boosting and bagging.

- *Boosting* [Freund and Schapire, 1997; Friedman, 2001] consists in successively combining base classifiers returned by a weak learner, to create a more accurate learner. In other words, boosting algorithms combine "simple" predictors (*i.e.* predictors with big approximation error), for example decision trees with a small depth.

- *Bagging* (or bootstrap aggregation) [Breiman, 1996] consists in combining multiple realizations over bootstrap samples of a "complex" predictor (*i.e.*

noisy but approximately unbiased predictor), for example fully grown decision trees.

Random Forests (RF), proposed in Breiman [2001], and inspired by previous works of Amit and Geman [1997]; Ho [1998], was in the spirit of bagging, based on tree models [Breiman et al., 1984]. Tree models are able to capture complex patterns, but fully grown trees overfit very easily on the training data: such property makes such tree-based models an ideal candidate for bagging. Denote M the number of individual trees. RF builds each tree on *bootstrap* samples: each time, n elements of index of training samples $I = \{1, \dots, n\}$, corresponding to in-the-bag (`itb`) samples, are uniformly selected at random with replacement. In classification tasks, RF takes the majority vote over predictions of all decision trees, or returns the average of probabilities of each class predicted by each tree; in regression tasks, RF takes the average predictions of all decision trees, *i.e.*

$$\widehat{g}(\cdot;\Pi) = \begin{cases} \text{majority vote}\{\widehat{f}(\cdot;\Pi_m)\}_{m=1,\dots,M}, & \text{in classification tasks;} \\ \frac{1}{M}\sum_{m=1}^{M}\widehat{f}(\cdot;\Pi_m), & \text{in classification or regression tasks,} \end{cases}$$

with Π_m denoting the randomness (including that from samples bootstrap and from features sub-sampling) corresponding to the m-th tree, $\widehat{f}(\cdot;\Pi_m)$ the predictor corresponding to the m-th tree, $\widehat{g}(\cdot;\Pi)$ the RF predictor, and $\Pi = (\Pi_1, \dots, \Pi_m)$ with $\Pi_1, \dots, \Pi_m$ i.i.d. realizations.

Overall, in Breiman's words [Breiman, 2001], *"each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest"*. We note that each tree is built independently of each others, so that tree training can be done in parallel.

Tree and tree predictor

In the following, we describe, in RF, the construction of each tree [Breiman et al., 1984], and the predictor that corresponds to the tree, *i.e.* how $\widehat{f}(\cdot;\Pi_m)$ is constructed. For simplicity, we omit the index $m = 1, \dots, M$, and use $I_{\texttt{itb}}$ to denote indexes of in-the-bag samples for the tree we consider. We focus on binary trees here.

A tree-based predictor partitions the features space $\mathcal{X}$ into rectangles (for continuous features) or discrete subsets (for categorical features) called *cells*, and provides simple (often constant) predictions on each cell. Since the partition can be complex, even with simple cell predictions, tree-based methods are powerful and able to capture complex and non-linear patterns. For tree growing, we recursively partition the features space, by successively applying *splits* into two cells. A decision tree predictor $\widehat{f} : \mathcal{X} \to \widehat{\mathcal{Y}}$ has the following properties.

- A finite partition $\mathscr{C}$ of $\mathcal{X}$ is said to be a *tree partition* if it is associated to
  - a finite ordered binary tree $\mathcal{T}$: each node $\mathbf{v}$ that is different to `root` has a unique parent, and either has two child nodes – with left child node denoted $\mathbf{v}0$ and right one denoted $\mathbf{v}1$, in which case $\mathbf{v}$ is an *interior node*, or no child node – in which case $\mathbf{v}$ is a *leaf*;
  - a family of splits $\Sigma = (\sigma_{\mathbf{v}})_{\mathbf{v}}$ indexed by the interior nodes of $\mathcal{T}$: a split is denoted by $\sigma_{\mathbf{v}} = (j_{\mathbf{v}}, t_{\mathbf{v}})$, where $j_{\mathbf{v}} \in \{1, \dots, d\}$ indicates the index of the feature, and $t_{\mathbf{v}}$ indicates a range of the feature's values less than

a threshold (split on a numerical feature) or a subset of the feature's modalities (split on a categorical feature).

We associate to each node $\mathbf{v}$ of $\mathcal{T}$ a cell $C_{\mathbf{v}} \subseteq \mathcal{X}$, recursively in the following way

- the cell associated to `root` is $C_{\texttt{root}} = \mathcal{X}$;
- for each interior node $\mathbf{v}$, given its cell $C_{\mathbf{v}}$ and split $\sigma_{\mathbf{v}} = (j_{\mathbf{v}}, t_{\mathbf{v}})$, we define
$$C_{\mathbf{v}0} = \{x \in C_{\mathbf{v}} : x_{j_{\mathbf{v}}} \in t_{\mathbf{v}}\} \quad \text{and} \quad C_{\mathbf{v}1} = C_{\mathbf{v}} \setminus C_{\mathbf{v}0}.$$

From this, the cells $C_{\mathbf{v}}$ of all leaves $\mathbf{v}$ of $\mathcal{T}$ form a partition $\mathscr{C}$ of $\mathcal{X}$, associated to $(\mathcal{T}, \Sigma)$.

- Then, we define the predictor of the tree $(\mathcal{T}, \Sigma)$ by $\widehat{f}_{(\mathcal{T},\Sigma)} : \mathcal{X} \to \widehat{\mathcal{Y}}, x \mapsto \widehat{y}_{C(x)}$, where $C(x)$ is the unique leaf containing x. A prediction $\widehat{y}_{C_{\mathbf{v}}}$ is associated to each leaf $\mathbf{v}$ of $\mathcal{T}$, *i.e.* to each cell $C_{\mathbf{v}}$ of $\mathscr{C}$. In regression tasks, the predictor of a leaf cell $C_{\mathbf{v}}$ is usually the empirical mean of all `itb` training samples in $C_{\mathbf{v}}$
$$\widehat{y}_{C_{\mathbf{v}}} = \frac{1}{\text{card}\{i \in I_{\texttt{itb}}, x_i \in C_{\mathbf{v}}\}} \sum_{i \in I_{\texttt{itb}}, x_i \in C_{\mathbf{v}}} y_i$$
In classification tasks, the predictor of $C_{\mathbf{v}}$ can be the majority vote of all `itb` training samples in $C_{\mathbf{v}}$
$$\widehat{y}_{C_{\mathbf{v}}} = \text{majority vote}\{y_i\}_{i \in I_{\texttt{itb}}, x_i \in C_{\mathbf{v}}}$$
or the density estimation based on all training samples $\{(x_i, y_i)\}_{i \in I_{\texttt{itb}}, x_i \in C_{\mathbf{v}}}$ in the cell, e.g. empirical frequencies for each class, or a Bayes predictive posterior.

A tree is grown successively by finding a split at each node $\mathbf{v}$, until meeting the stopping criterion. To find such a split $\sigma_{\mathbf{v}}$, we proceed greedily: for each feature indexed by j, we determine the optimal split $\sigma_{\mathbf{v},j}$ along the feature j, in the sense of the criterion; we do so and loop over all the currently available features $j \in \text{sub-sampling}\{1, \ldots, d\}$ (as for features sub-sampling); then we take the best of such splits $\sigma_{\mathbf{v},j}$ in the sense of the criterion, as the actual split $\sigma_{\mathbf{v}}$. To be able to do so, we need a *criterion* $\mathscr{P}(\mathcal{X}) \to \mathbb{R}$ to measure the qualities of possible splits: the smaller the criterion function value is, the more suitable the split is on the data contained in the current cell $C_{\mathbf{v}}$, the better the split is. In classification, the Gini index and the cross-entropy are popular choices for the criterion. In regression, the sum of squares is usually used as the criterion.

In RF [Breiman, 2001], an additional source of randomness is the feature *sub-sampling*: for each split, $m \leqslant d$ features are randomly and uniformly selected (without replacement); only these m features can be candidates for the current split. Typical values are $m = \sqrt{d}$ or $m = \log d$ or $m = d/3$ (for regression). Feature sub-sampling allows to introducing more variability and to further reducing correlation between the trees – a particularly nice property with bagging. In RF, trees often stop growing only very lately, a typical stopping criterion checks that only a few number of training samples remains on a leaf or on a split.

Random Forest. We recap the main ingredients in Random Forest as presented in Breiman [2001] in the following.

- A bootstrapped sample of the training dataset is generated for each tree: each time, n samples are uniformly chosen with replacement from the training dataset $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.
- Trees are constructed in parallel, and are fully grown until they meet a stopping criterion. With no tree pruning, each cell usually contains few data points.
- The randomness of the partition choice comes from the bootstrap and also from the features sub-sampling.

Since its introduction, Random Forest is one of the most widely used supervised learning algorithms, in both classification and regression tasks. It combines good predictive performances, weak training and inference cost while being parallel computation friendly, very few number of hyperparameters to tune, and at the same time enjoys some interpretability. Some development about Random Forest methodology include extremely randomized trees [Geurts et al., 2006], variable importance [Louppe et al., 2013], pruning method for ensemble [Ren et al., 2015], regularization mechanism with depth limitation in Random Forest [Zhou and Mentch, 2021], to cite few of them.

1.3.2 Aggregation algorithms

The exponential weight aggregation algorithm and the histogram strategy are the two principal novelties that WildWood introduce comparing to Random Forest.

Exponential weights algorithms

In the classical online learning literature such as Cesa-Bianchi and Lugosi [2006], the exponential weights aggregation is introduced as a strategy for prediction of deterministic individual sequences with expert advices. Formally, let Θ be the measurable space of the parameters of estimators, $\widehat{y}_{\theta,t}$ the predictor related to the parameter θ at time t, and π a probability measure on Θ named prior. In the online learning setting, the exponential weight aggregation algorithm proposes to use the aggregator, at time t,

$$\widehat{y}_t \coloneqq \int_\Theta \widehat{y}_{\theta,t} v_t(\mathrm{d}\theta) \quad \text{where} \quad \frac{\mathrm{d}v_t}{\mathrm{d}\pi}(\theta) \coloneqq \frac{e^{-\eta L_{\theta,t-1}}}{\int_\Theta e^{-\eta L_{\theta',t-1}} \pi(\mathrm{d}\theta')}$$

with $L_{\theta,t}$ denoting the loss of the estimator $\widehat{y}_{\theta,t}$ related to θ at time t, evaluated on all data until time t in hindsight. This predictor is largely studied in Cesa-Bianchi and Lugosi [2006]. For example, with exp-concave loss function, uniform prior π, and discrete and finite-numbered parameters set Θ, running exponential weights algorithm guarantees a regret that is logarithmic in the size of Θ.

The same spirit can be translated into the general supervised learning setting. Denote $\widehat{f}_\theta$ the predictor related to the parameter θ, the exponential weight aggregation algorithm uses the aggregator

$$\widehat{f}(\cdot) \coloneqq \int_\Theta \widehat{f}_\theta(\cdot) v(\mathrm{d}\theta) \quad \text{where} \quad \frac{\mathrm{d}v}{\mathrm{d}\pi}(\theta) \coloneqq \frac{e^{-\eta L_\theta}}{\int_\Theta e^{-\eta L_{\theta'}} \pi(\mathrm{d}\theta')}$$

with L_θ the loss of the estimator $\widehat{f}_\theta$, evaluated on all the training data. In regression and squared loss setting, a study [Dalalyan and Tsybakov, 2007] of exponential weights aggregation obtained sharp PAC-Bayesian risk bounds.

When it comes to the aggregation of subtrees of a tree, we need to consider particular forms of prior π over the tree structure. Context tree weighting [Catoni, 2004] is indeed useful in designing such aggregations efficiently.

Popular gradient boosting algorithms

In the family of boosting algorithms, extreme gradient boosting algorithms are powerful and often popular in data science challenges. Some successful examples are XGBoost [Chen and Guestrin, 2016] which provides a popular scalable tree boosting system widely adopted in industry, LightGBM [Ke et al., 2017] which introduces the "histogram strategy" for faster split finding, together with clever down-sampling and features grouping algorithms in order to achieve high performance in reduced computation times, and CatBoost [Prokhorenkova et al., 2017] which pays particular attention to categorical features using target encoding, while addressing the potential bias issues associated to such an encoding.

1.3.3 Contribution: WildWood, a new random forest algorithm

On the one hand, as a bagging algorithm, RF builds each tree based on bootstrapped training samples, using out-of-bag (`oob`) samples only for computing scores. On the other hand, as far as we know, there is no efficient implementation of Random Forest algorithm that implements histogram strategy and has native support for categorical features. In the work presented in Chapter 4, we propose WildWood (WW), which keeps all ingredients of RF including bootstrap, fully grown trees, features sub-sampling, similar tree growing procedure, with an improved tree predictor and efficient computations implemented in `Python`.

Improving the tree predictor by the subtrees aggregation with exponential weights

Consider a fully grown tree $\mathcal{T}$. Recall that in standard RF, the tree predictor corresponding to $\mathcal{T}$ writes $\widehat{f}(x) = \widehat{y}_{C_{\mathbf{v}}(x)}$ for any $x \in \mathcal{X}$, where $C(x)$ is the cell containing x associated to leaf $\mathbf{v}$: this is to find the smallest cell containing x, and outputs the prediction of this cell. In WW, we improve that tree predictor by the following mechanism: as the predictor for $\mathcal{T}$, WW uses

$$\widehat{f}(\cdot) = \frac{\sum_{T \subset \mathcal{T}} \pi(T) e^{-\eta L_T} \widehat{y}_T(\cdot)}{\sum_{T \subset \mathcal{T}} \pi(T) e^{-\eta L_T}} \quad \text{with} \quad \pi(T) = 2^{-\|T\|}, \tag{1.4}$$

where the sum is over all subtrees T of $\mathcal{T}$ rooted at `root`, $\eta > 0$ is temperature parameter, $\|T\|$ denotes the number of nodes in T minus its number of leaves that are also leaves of $\mathcal{T}$, L_T is the cumulative loss of the prediction of subtree T over `oob` samples, *i.e.* $L_T := \sum_{i \in I_{\texttt{oob}}} \ell(\widehat{y}_T(x_i), y_i)$, and $\widehat{y}_T$ is prediction of subtree T in its classical way. The prediction function (1.4) can be understood as an aggregation of the predictions $\widehat{y}_T(\cdot)$ of all subtrees T, weighted by their performances on `oob` samples together with prior $\pi(T) = 2^{-\|T\|}$; this is indeed a *non-greedy way to prune trees*: the weights depend not only on the quality of one single split but also on the performance of each subsequent splits.

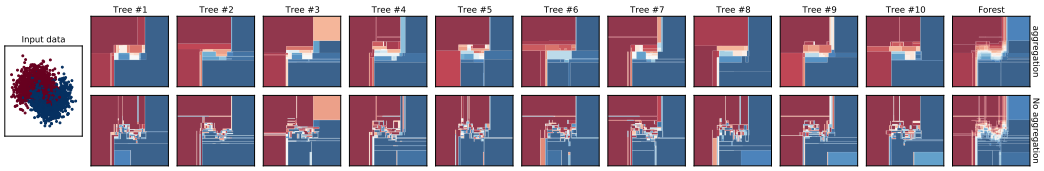

Figure 1.3 – WW decision functions illustrated on a toy dataset (left) with subtrees aggregation (top) and without it (bottom). Subtrees aggregation improves trees predictions, as illustrated by smoother decision functions in the top compared with the bottom, improving overall predictions of the forest (last column).

Furthermore, the following Theorem gives a theoretical guarantee over `oob` samples, for subtree aggregation, under an assumption on exp-concavity of the loss function. This Theorem states that the predictor given by (1.4) is able to perform almost as well as the best oracle subtree $T \subseteq \mathcal{T}$ on `oob` samples, with an $O(\|T\|/n_{\texttt{oob}})$ rate which is optimal for model-selection oracle inequalities [Tsybakov, 2003]; in contrast, finding an oracle $\operatorname{argmin}_{T \in \mathcal{T}} \sum_{i \in I_{\texttt{oob}}} \ell(\widehat{f}(x_i), y_i)$ is computationally infeasible, which requires trying out all subtrees.

Theorem (Oracle inequality, Theorem 4.2 restated)**.** *Assume that the loss function ℓ is η-exp-concave. Then, the prediction function $\widehat{f}$ given by (1.4) satisfies the oracle inequality*

$$\frac{1}{n_{\texttt{oob}}} \sum_{i \in I_{\texttt{oob}}} \ell(\widehat{f}(x_i), y_i) \leqslant \inf_{T \subset \mathcal{T}} \left\{ \frac{1}{n_{\texttt{oob}}} \sum_{i \in I_{\texttt{oob}}} \ell(\widehat{y}_T(x_i), y_i) + \frac{C\|T\|}{\eta(n_{\texttt{oob}} + 1)} \right\},$$

where the infimum is taken over all subtrees $T \subset \mathcal{T}$ rooted at `root`*, and $C = \log 2$.*

As a consequence, we proved that the predictor (1.4) given by subtree aggregation, is almost as good as the best of any pruning of the same tree, *i.e.* any predictor that is constant on its leaves, see Corollary 4.3 for classification tasks with the log-loss, and Corollary 4.4 for regression tasks with the least-square loss.

Efficient implementation of WildWood

A naive computation of prediction function (1.4) would involve summing over all subtrees, the computation cost would be exponential in the number of nodes, which is infeasible, WW computes (1.4) exactly and efficiently thanks to context tree weighting, as stated in the following Theorem.

Theorem (Theorem 4.1 simplified)**.** *The prediction function (1.4) can be computed recursively along the path of x in $\mathcal{T}$, with a computational cost in the same order with that of a standard RF.*

A consequence of this Theorem is an efficient computation of $\widehat{f}(x)$, exploiting the fact that trees in WW are grown in a depth-first fashion. Computing $\widehat{f}(x)$ indeed only increases by a factor 2 the computational complexity of a standard RF.

Besides this efficient computation of subtree aggregation, WW natively supports categorical features and implements a histogram strategy, to accelerate split finding. Our implementation of WildWood is available at the GitHub repository `https://github.com/pyensemble/wildwood`.

We conducted extensive experiments assessing the predictive performances and running times of our implementation of WildWood, see Tables 4.1 and 4.2 in Chapter 4 of this manuscript for detailed experiments settings and results. All experiments can be reproduced using `Python` scripts on the same repository. Overall, as an improved Random Forest algorithm, WildWood showed competitive performance in terms of predictive power and running time, compared to standard Random Forest and extreme gradient boosting algorithms.

1.4 Online logistic regression: towards an efficient algorithm with better regret guarantee

In the previous section, we consider the batch supervised learning, where data are a set of observations $\{(x_1, y_1), \dots, (x_n, y_n)\}$ obtained at once. Now in the work presented in Chapter 5, we move to the online learning setting, where data are revealed sequentially. In such a setting, in general, we do not make any assumption on the data distribution, and we look for some algorithms with a performance guarantee that is valid for any *individual sequence* $(x_t, y_t)_t$. Some classical references include Cesa-Bianchi and Lugosi [2006] as well as Hazan [2019]; Orabona [2019].

1.4.1 Online learning: general principles

Let us start by providing some general background on online learning. Denote $\widehat{\mathcal{Y}}$ the space of predictions and $\mathcal{Y}$ the space of labels. Given a loss function $\ell : \widehat{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}$, the objective of online learning is to produce some step-by-step predictions for observations $y_1, y_2, \dots \in \mathcal{Y}$ that are accurate in the sense of the loss function.

More precisely, we formulate the setting of online learning as a game between an agent (who produces predictions) and an environment (which generates ground-truths). At each time step $t = 1, 2, \dots$,

- the agent chooses a prediction $\widehat{y}_t$, based on all previous observations $S_{t-1} = (y_1, \dots, y_{t-1})$;
- the environment reveals the actual label y_t;
- the agent suffers the loss $\ell(\widehat{y}_t, y_t)$ at time t.

A *learning algorithm* $\mathcal{A}$, or a *strategy*, is a function of all previous observations that returns the prediction at time t, *i.e.* following $\mathcal{A}$, the agent predicts $\widehat{y}_t = \mathcal{A}(S_{t-1})$ at each time t.

The objective of the agent is to produce predictions $(\widehat{y}_t)_t$ that minimize the cumulative loss $\sum_{t=1}^{n} \ell(\widehat{y}_t, y_t)$, where the number of total rounds n might or might not be known in advance. However, since we do not make any assumption on the distribution of $(y_t)_t$, the cumulative loss can be big or small according to the "difficulty" of the observations (e.g. the environment may choose adversarially y_t with the knowledge of the agent's prediction $\widehat{y}_t$). To take this fact into account, we introduce the notion of *regret* to effectively evaluate the quality of the successive predictions $(\widehat{y}_t)_t$, or the quality of an algorithm $\mathcal{A}$. Let $f \in \widehat{\mathcal{Y}}$ be a predictor, the regret of successively playing $(\widehat{y}_t)_t$ compared to f, is the excess cumulative loss versus using f as the single fixed predictor in hindsight,

$$\mathrm{Regret}_n(f) := \sum_{t=1}^{n} \ell(\widehat{y}_t, y_t) - \sum_{t=1}^{n} \ell(f, y_t)$$

where n stands for the number of total rounds, or the time-horizon. Consider a subclass $\mathcal{F} \subseteq \widehat{\mathcal{Y}}$ of predictors, the regret of predictions $(\widehat{y}_t)_t$ relative to $\mathcal{F}$ is defined by the excess cumulative loss compared to the best predictor $f \in \mathcal{F}$ as the single fixed predictor in hindsight, namely

$$\text{Regret}_n \coloneqq \sum_{t=1}^{n} \ell(\widehat{y}_t, y_t) - \inf_{f \in \mathcal{F}} \sum_{t=1}^{n} \ell(f, y_t). \tag{1.5}$$

In this case $\mathcal{F}$ is called *comparison class*. The objective of most of the studies in online learning [Cesa-Bianchi and Lugosi, 2006; Hazan, 2019; Orabona, 2019] is to obtain guarantees on regret that is valid for any *individual sequence* $(y_1, \dots, y_n)$. Therefore, we are usually interested in upper-bounds on the worst case regret of an algorithm $\mathcal{A}$, *i.e.* upper-bounds on

$$\text{Regret}_n = \sum_{t=1}^{n} \ell(\mathcal{A}(S_{t-1}), y_t) - \inf_{f \in \mathcal{F}} \sum_{t=1}^{n} \ell(f, y_t).$$

The exponential weight aggregation, as mentioned in Section 1.3.2, is a fundamental strategy of online learning.

Supervised online learning

The setting of *supervised online learning* is slightly different to the general setting of online learning: at time t, features $x_t \in \mathcal{X}$ are revealed, then the agent predicts $\widehat{y}_t$ with the knowledge of x_t. Formally, at each time step $t = 1, 2, \dots,$

- the environment reveals the features $x_t \in \mathcal{X}$;
- the agent chooses a prediction $\widehat{y}_t$, based on all previous observations $S_{t-1} = ((x_1, y_1), \dots, (x_{t-1}, y_{t-1}))$ and x_t;
- the environment reveals the actual label y_t;
- the agent suffers the loss $\ell(\widehat{y}_t, y_t)$.

Under the setting of supervised online learning, predictors are functions of the features $f : \mathcal{X} \to \widehat{\mathcal{Y}}, x_t \mapsto f(x_t) = \widehat{y}_t$. The inputs to a learning algorithm $\mathcal{A}$ include all the previous observations as well as the features of time t, *i.e.* the agent predicts $\widehat{y}_t = \mathcal{A}(S_{t-1}, x_t)$. If the predictor returned by the learning algorithm $x_t \mapsto \mathcal{A}(S_t, x_t)$ belongs to the comparison class $\mathcal{F}$, we say that the algorithm is *proper*; otherwise, the algorithm is *improper*. Classical online learning methods often focus on predictors of the form $x_t \mapsto \mathcal{A}(S_{t-1})(x_t)$, e.g. in the OCO setting described in the next paragraph: they are usually proper algorithms. There are similar notions in statistical learning theory [Bousquet et al., 2003; Shalev-Shwartz and Ben-David, 2014] for proper (or *plug-in* estimator, when the estimator takes value inside the comparison class) and improper (when the estimator is not restricted to the comparison class) algorithm. Let us conclude this remark on the proper and improper algorithms by pointing out that, in the case where an algorithm $\mathcal{A}$ effectively uses the features x_t to build the predictor f at time t, $\mathcal{A}$ is necessarily improper.

Consider the parametrized comparison class $\mathcal{F} = \{f_\theta : \mathcal{X} \to \widehat{\mathcal{Y}}, \theta \in \Theta\}$ where Θ is the space of the comparison parameter θ. Then, the regret of an algorithm $\mathcal{A}$ relative to $\mathcal{F}$ writes

$$\text{Regret}_n \coloneqq \sum_{t=1}^{n} \ell(\mathcal{A}(S_{t-1}, x_t), y_t) - \inf_{\theta \in \Theta} \sum_{t=1}^{n} \ell(f_\theta(x_t), y_t).$$

Online convex optimization (OCO)

Assume that loss function $\ell : \widehat{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}$ is convex and differentiable, *i.e.* for any $y \in \mathcal{Y}$, the function $\widehat{y} \mapsto \ell(\widehat{y}, y)$ is convex and differentiable. We follow the general setting of online learning. It is practical to introduce the notation ℓ_t to denote the loss function induced by the observation y_t at time t, *i.e.* $\ell_t : \widehat{y} \mapsto \ell(\widehat{y}, y_t)$. Then, for any $\widehat{y} \in \widehat{\mathcal{Y}}$,

$$\ell_t(\widehat{y}_t) - \ell_t(\widehat{y}) \leqslant \langle \nabla \ell_t(\widehat{y}_t), \widehat{y}_t - \widehat{y} \rangle$$

because of its convexity. Hence, denoting $h_t \coloneqq \nabla \ell_t(\widehat{y}_t)$, we can write an upper-bound of the regret of the predictions $(\widehat{y}_t)$ as the regret produced by the linear functions $\langle h_t, \cdot \rangle$. Namely,

$$\text{Regret}_n = \sum_{t=1}^{n} \ell(\widehat{y}_t, y_t) - \inf_{\widehat{y} \in \widehat{\mathcal{Y}}} \sum_{t=1}^{n} \ell(\widehat{y}, y_t) \leqslant \sum_{t=1}^{n} \langle h_t, \widehat{y}_t \rangle - \inf_{\widehat{y} \in \widehat{\mathcal{Y}}} \sum_{t=1}^{n} \langle h_t, \widehat{y} \rangle \,.$$

Therefore, we use the convexity of the loss function to reduce the problem into that of minimizing the linear regret. Although this reduction might not always be optimal, the use of convexity offers a practical starting point to design OCO [Hazan, 2019; Orabona, 2019] algorithms, usually efficient and proper.

Online-to-batch conversions

We present some notions in the batch learning, in particular their similarities and relationship with the online setting. Let π be a probability measure on $\mathcal{Y}$, the quality of a predictor $\widehat{y} \in \widehat{\mathcal{Y}}$ is assessed by the expected loss, or its *risk* $R(\widehat{y}) \coloneqq \mathbb{E}_{Y \sim \pi}[\ell(\widehat{y}, Y)]$. The *excess risk* of a predictor $\widehat{y} \in \widehat{\mathcal{Y}}$ related to a comparison class $\mathcal{F} \subseteq \widehat{\mathcal{Y}}$ is the difference between the risk induced by $\widehat{y}$ and that of the best predictor $f \in \mathcal{F}$, namely

$$\mathcal{E}(\widehat{y}) \coloneqq R(\widehat{y}) - \inf_{f \in \mathcal{F}} R(f).$$

We can see that the notion of excess risk in batch learning is analogous to the notion of regret (Equation (1.5)) in the online setting. Under i.i.d. assumption for data and with a convex loss function, an upper-bound on the regret of an online algorithm entails an upper-bound on the expected excess risk, using the average predictor, as explained by the following Proposition.

Proposition 1.1 (Online-to-batch conversion, Cesa-Bianchi et al. [2004])**.** *Assume that $\widehat{\mathcal{Y}}$ is convex and that $\widehat{y} \mapsto \ell(\widehat{y}, y)$ is convex for any $y \in \mathcal{Y}$. Let $\widehat{Y}_1, \ldots, \widehat{Y}_n$ be predictors returned by an online algorithm, such that $\widehat{Y}_t$ depends on the sequence $S_{t-1} = (Y_1, \ldots, Y_{t-1})$. Consider the average predictor $\bar{Y}_n$ defined by*

$$\bar{Y}_n = \frac{1}{n} \sum_{t=1}^{n} \widehat{Y}_t.$$

Hence, $\bar{Y}_n$ only depends on $(Y_1, \ldots, Y_{n-1})$. Let π be a probability measure on $\mathcal{Y}$, and assume that $Y_1, \ldots, Y_{n+1}$ are i.i.d. random variables following π. Denote Regret_n *the regret (Equation 1.5) of the predictions $(\widehat{Y}_1, \ldots, \widehat{Y}_n)$ against the comparison class $\mathcal{F}$, on the sequence $(Y_1, \ldots, Y_n)$. Then*

$$\mathbb{E}[R(\bar{Y}_n)] - \inf_{f \in \mathcal{F}} R(f) \leqslant \frac{1}{n} \mathbb{E}\left[\text{Regret}_n\right], \tag{1.6}$$

where the expectations are taken with respect to the joint distribution of $(Y_1, \cdots, Y_n)$.

Proof. For any time $t = 1, \dots, n$, since $\widehat{Y}_t$ only depends on $(Y_1, \dots, Y_{t-1})$ and is independent of Y_t,

$$\mathbb{E}[R(\widehat{Y}_t)] = \mathbb{E}\left[\mathbb{E}_{Y_t \sim \pi}[\ell(\widehat{Y}_t, Y_t)] \,|\, \widehat{Y}_t\right] = \mathbb{E}[\ell(\widehat{Y}_t, Y_t)]$$

where the expectations are taken with respect to the joint distribution of $(Y_1, \cdots, Y_t)$. Since the loss function ℓ is convex, the risk R is convex. Then, for any $f \in \mathcal{F}$,

$$\begin{aligned}\mathbb{E}[R(\bar{Y}_n)] - R(f) &\leqslant \frac{1}{n}\sum_{t=1}^{n}\Big(\mathbb{E}[R(\widehat{Y}_t)] - R(f)\Big) \\ &= \frac{1}{n}\mathbb{E}\left[\sum_{t=1}^{n}\ell(\widehat{Y}_t, Y_t) - \ell(f, Y_t)\right] \leqslant \frac{\mathbb{E}\left[\mathrm{Regret}_n\right]}{n}\end{aligned}$$

where the expectations are also taken with respect to the joint distribution of $(Y_1, \cdots, Y_n)$; the first inequality uses the convexity of R, and the last inequality uses the definition of the regret (Equation (1.5)) on the sequence $(Y_1, \dots, Y_n)$. Then we obtain the Equation (1.6) by taking the supremum over $f \in \mathcal{F}$. □

Proposition 1.1 gives a general procedure using the average predictor to convert an online algorithm into a batch algorithm, with guaranteed excess risk in expectation; the same average predictor is also showed with generalization ability with big probability, see for example Cesa-Bianchi et al. [2004]. Other forms of online-to-batch conversion are also possible, for example a non-uniform average predictor was reported in Zhang [2004].

Besides, we remark that in Proposition 1.1, we can get rid of the convexity assumption. In that case, instead of taking the average predictor, we choose $\bar{Y}_n = \widehat{Y}_\tau$ with τ uniform drawn from $\{1, \dots, n\}$, and it yields the same excess risk guarantee in expectation.

1.4.2 Online logistic regression, related works

Online binary logistic regression falls into the setting of supervised online learning, with $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{-1, +1\}$, and $\mathcal{F}$ corresponding to the logistic models $\{f_\theta : \theta \in \mathbb{R}^d\}$, such that

$$p(1 \mid x) = 1 - p(-1 \mid x) = f_\theta(x) = \sigma(\langle \theta, x \rangle)$$

for any $x, \theta \in \mathbb{R}^d$, where σ denotes the sigmoid function $\sigma(u) \coloneqq e^u/(1+e^u)$. The loss function for this problem is given by

$$\ell(\langle \theta, x \rangle, y) = -\log \sigma(y \langle \theta, x \rangle) = -\mathbf{1}_{y=1} \log \sigma(\langle \theta, x \rangle) - \mathbf{1}_{y=-1} \log \sigma(-\langle \theta, x \rangle).$$

Usually, we study online binary logistic regression under an assumption on the bounded features $\|x_t\| \leqslant R$ and bounded comparison class $\mathcal{F} = \{x \mapsto \langle \theta, x \rangle, \|\theta\| \leqslant B\}$, with some $R, B > 0$.

As an instance of online convex optimization, gradient methods such as Online Gradient Descent (OGD) [Zinkevich, 2003] and Online Newton Steps (ONS) [Hazan et al., 2007] apply for online logistic regression. However, under the proper setting and by constructing adversary examples, Hazan et al. [2014] showed that in particular, any $O(B \log n)$ regret is impossible for proper algorithms.

However, better regret guarantees are possible with improper algorithms. The following studies indeed moved on to some improper Bayesian and non-Bayesian

algorithms. Bayesian improper method Foster et al. [2018] enjoys an optimal regret guarantee but suffers from its computational complexity. Non-Bayesian improper method AIOLI [Jézéquel et al., 2020] is built upon a new quadratic approximation of the logistic regression and a regularization by samples, coming with an extra factor BR in its regret guarantee but with efficient computations. Recent works GAF [Jézéquel et al., 2021] and FOLKLORE [Agarwal et al., 2021] both extend the quadratic approximation first introduced in AIOLI [Jézéquel et al., 2020] into the multi-class case, respectively propose a Bayesian improper algorithm and a non-Bayesian improper algorithm for multi-class logistic regression, with regret guarantee similar to the AIOLI one. On the other hand, in the batch setting (in particular batch logistic regression), Mourtada and Gaïffas [2019] introduces a min-max estimator. This improper algorithm achieves an excess risk guaranteed in the optimal order.

A detailed literature review is proposed in Chapter 5 of the manuscript. Let us summarize here in Table 1.2 the regret upper-bounds and the computational costs of these algorithms for online binary logistic regression.

Table 1.2 – Regret upper-bounds and computational costs in $O(\cdot)$ of several algorithms for online binary logistic regression: OGD [Zinkevich, 2003], ONS [Hazan et al., 2007], Foster [Foster et al., 2018], GAF [Jézéquel et al., 2021], AIOLI [Jézéquel et al., 2020], FOLKLORE [Agarwal et al., 2021]. AOSMP stands for Approximated One-Step Minmax Predictor (Algorithm 2), C_{GD} stands for the computational cost of the gradient descent for finding $L_t^{y\star}$. Algorithms with * also handle the multi-class case.

Algorithm	Type	Regret upper-bound	Computational cost
OGD	Proper	$BR\sqrt{n}$	nd
ONS	Proper	$de^{BR}\log(n)$	nd^2
Foster	Improper, Bayesian	$d\log(BRn)$	$B^6n^{12}(Bn+d)^{12}$
GAF*	Improper, Bayesian	$dBR\log(n)+dB^2$	nd^2+n^4
AIOLI	Improper, non-Bayesian	$dBR\log(BRn)$	$nd^2+nBR\log(n)$
FOLKLORE*	Improper, non-Bayesian	$dBR\log(n)$	$nd^2+nBR\log(n)$
AOSMP	Improper, non-Bayesian	$dBR\log(n)+B^2R^2$	nd^2+nC_{GD}

1.4.3 Contribution: study of some non-Bayesian improper algorithms

The starting point of the work presented in Chapter 5 is the following question: with possibly some adaptation, could an online version of SMP [Mourtada and Gaïffas, 2019] achieve a regret guarantee comparable to SMP's excess risk? We present two candidate algorithms, OSMP and AOSMP, and their regret analysis.

Let us start by considering the λ-ridge-penalized regret until time t for online logistic regression, which writes,

$$\sum_{s=1}^{t}\ell(\widehat{y}_s,y_s)-\inf_{\theta\in\Theta}\left(\sum_{s=1}^{t}\ell(\theta^\top x_s,y_s)+\lambda\left\|\theta\right\|^2\right).$$

We introduce the *One-Step Minmax Predictor* (OSMP), taking "best possible" choice of prediction $\widehat{y}_t$ against "the worst possible" ground-truth value of $y_t\in\{-1,+1\}$ and the adversary parameter θ, measure by the λ-ridge-penalized regret

above. Namely,

$$\widehat{y}_t = \operatorname*{argmin}_{\widehat{y}\in\mathbb{R}} \sup_{y_t\in\{-1,1\}} \sup_{\theta\in\mathbb{R}^d} \left\{ \ell(\widehat{y}, y_t) + \widehat{L}_{t-1} - \Big(\ell(\theta^\top x_t, y_t) + L_{\lambda,t-1}(\theta) \Big) \right\},$$

with $\widehat{L}_{t-1} \coloneqq \sum_{s=1}^{t-1} \ell(\widehat{y}_s, y_s)$, and $L_{\lambda,t-1}(\theta) \coloneqq \sum_{s=1}^{t-1} \ell(\theta^\top x_s, y_s) + \lambda\|\theta\|^2$. We show that equivalently (Lemma 5.4), OSMP predicts

$$\widehat{y}_t = -L_{\lambda,t}^{+1\star} + L_{\lambda,t}^{-1\star}$$

with $L_{\lambda,t}^{y\star} \coloneqq \inf_{\theta\in\mathbb{R}^d} \left\{ \ell(\theta^\top x_t, y) + L_{\lambda,t-1}(\theta) \right\}$ for $y \in \{-1, +1\}$. This gives the explicit Algorithm 1 for the computation of OSMP. On the regret analysis of OSMP, we first remark that

$$\mathrm{Regret}_n(\theta) = \sum_{t=1}^{n} \ell(\widehat{y}_t, y_t) - \sum_{t=1}^{n} \ell(\theta^\top x_t, y_t) \leqslant \lambda \|\theta\|^2 + \sum_{t=1}^{n} \widehat{r}_t,$$

where we denote $\widehat{r}_t \coloneqq \ell(\widehat{y}_t, y_t) - L_{\lambda,t}^\star + L_{\lambda,t-1}^\star$ the instant regret of time t. We deduce the following upper-bound for the instant regret, whose proof relies on the generalized R-self-concordance [Bach, 2010] of the function $L_{\lambda,t}$.

Algorithm 1 One-Step Minmax Predictor (OSMP) description.

Inputs: Parameters $\lambda > 0$, $n, d \geqslant 1$ constants $B, R > 0$
Initialize: $L_{\lambda,0}(\theta) = \lambda \|\theta\|^2$
for $t = 1, \ldots, n$ **do**
  Receive x_t
  Compute $L_{\lambda,t}^{y\star} \leftarrow \inf_{\theta\in\mathbb{R}^d} \left\{ \ell(\theta^\top x_t, y) + L_{\lambda,t-1}(\theta) \right\}$ for $y = -1$ and $y = +1$
  Predict $\widehat{y}_t \leftarrow -L_{\lambda,t}^{+1\star} + L_{\lambda,t}^{-1\star}$
  Receive y_t
  Update function $L_{\lambda,t}(\theta) = L_{\lambda,t-1}(\theta) + \ell(\theta^\top x, y_t)$
end for

Proposition (Proposition 5.9)**.** *Denote $\theta_t \coloneqq \operatorname{argmin}_{\theta\in\mathbb{R}^d} L_{\lambda,t}(\theta)$, assume $\|x_t\| \leqslant R$ for some $R > 0$. We run OSMP (Algorithm 1) with $\lambda \geqslant R^2$. Then the instant regret as defined above is upper-bounded,*

$$\widehat{r}_t \leqslant e \cdot \sigma'(\langle \theta_t, x_t \rangle) \cdot \|x_t\|^2_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}}.$$

However, this Proposition is as far as goes our analysis of OSMP. In particular, the value of Hessian $\nabla^2 L_{\lambda,t}(\theta_t)$ depends on the point θ_t where it is evaluated, and we are not aware of a way to control such Hessian matrix leading to a regret upper-bound which is logarithmic in n and without an exponential factor in BR, see Section 5.5.3 for more details.

To get around this difficulty, we introduce the following *Approximated One-Step Minmax Predictor* (AOSMP), defined by

$$\widehat{y}_t = \operatorname*{argmin}_{\widehat{y}\in\mathbb{R}} \sup_{y_t\in\{-1,+1\}} \sup_{\theta\in\mathbb{R}^d} \left\{ \widehat{L}_{t-1} + \ell(\widehat{y}, y_t) - \Big(\ell(\theta^\top x_t, y_t) + \tilde{L}_{\lambda,t-1}(\theta) \Big) \right\}$$

with $\tilde{L}_{\lambda,t-1}(\theta) \coloneqq \sum_{s=1}^{t-1} \tilde{\ell}(\theta^\top x_s, y_s) + \lambda\|\theta\|^2$. This formulation is in the same minmax spirit with OSMP. The difference is that in AOSMP, we replace the evaluation function on the adversary parameter $L_{\lambda,t-1}$ by its quadratic surrogate $\tilde{L}_{\lambda,t-1}$. We rely

on the quadratic approximation of the logistic function first introduced in Jézéquel et al. [2020, Lemma 5], namely

$$\tilde{\ell}_t(\theta) \coloneqq \ell_t(\tilde{\theta}_t) + g_t^\top(\theta - \tilde{\theta}_t) + \frac{1}{2}\eta_t \left(x_t^\top(\theta - \tilde{\theta}_t)\right)^2, \tag{1.7}$$

with $g_t \coloneqq \nabla \ell_t(\tilde{\theta}_t)$, $\eta_t \coloneqq \sigma'(\langle \tilde{\theta}_t, x_t \rangle)/(1 + BR)$ and we choose

$$\tilde{\theta}_t = \operatorname*{argmin}_{\theta \in \mathbb{R}^d} \tilde{L}_{\lambda,t}^{y_t}(\theta) \quad \text{with} \quad \tilde{L}_{\lambda,t}^{y}(\theta) \coloneqq \ell(\theta^\top x_t, y) + \tilde{L}_{\lambda,t-1}(\theta).$$

For the same reasons for OSMP, AOSMP predicts

$$\widehat{y}_t = -\tilde{L}_{\lambda,t}^{+1\star} + \tilde{L}_{\lambda,t}^{-1\star}$$

with $\tilde{L}_{\lambda,t}^{y\star} \coloneqq \inf_{\theta \in \mathbb{R}^d} \left\{ \ell(\theta^\top x_t, y) + \tilde{L}_{\lambda,t-1}(\theta) \right\}$ for $y \in \{-1, +1\}$, and that gives Algorithm 2 for the computation of AOSMP. Then, we are able to apply the classical techniques with the quadratic forms to upper-bound the sum of the pseudo instant regrets, to obtain the following Theorem on the regret of AOSMP.

Algorithm 2 Approximated One-Step Minmax Predictor (AOSMP) description.

Inputs: Parameters $\lambda > 0$, $n, d \geqslant 1$, constants $B, R > 0$
Initialize: Function $\tilde{L}_{\lambda,0}(\theta) = \lambda \|\theta\|^2$
for $t = 1, \ldots, n$ **do**
  Receive x_t
  Compute $\tilde{L}_{\lambda,t}^{y\star} \leftarrow \inf_{\theta \in \mathbb{R}^d} \left\{ \ell(\theta^\top x_t, y) + \tilde{L}_{\lambda,t-1}(\theta) \right\}$ for $y = -1$ and $y = +1$
  Predict $\widehat{y}_t \leftarrow -\tilde{L}_{\lambda,t}^{+1\star} + \tilde{L}_{\lambda,t}^{-1\star}$
  Receive y_t
  Compute $\tilde{\theta}_t \leftarrow \operatorname{argmin}_{\theta \in \mathbb{R}^d} \left\{ \ell(\theta^\top x_t, y_t) + \tilde{L}_{\lambda,t-1}(\theta) \right\}$, and function $\tilde{\ell}_t(\theta)$ as specified in Equation (1.7)
  Update function $\tilde{L}_{\lambda,t}(\theta) = \tilde{L}_{\lambda,t-1}(\theta) + \tilde{\ell}_t(\theta)$
end for

Theorem (Theorem 5.19)**.** *Let $R, B > 0$ and $d, n \geqslant 1$. Let $(x_1, y_1), \ldots, (x_n, y_n) \in [-R, R]^d \times \{-1, +1\}$ be an arbitrary sequence of observations. We run AOSMP (Algorithm 2) with $\lambda \geqslant R^2$. Then its regret against any $\theta \in \mathscr{B}(\mathbb{R}^d, B)$ satisfies the following upper-bound,*

$$\mathrm{Regret}_n(\theta) \leqslant e \cdot (1 + BR) d \log\left(1 + \frac{nR^2}{8d(1 + BR)\lambda}\right) + \lambda \|\theta\|^2.$$

In particular, choosing $\lambda = R^2$ yields

$$\mathrm{Regret}_n \leqslant e \cdot (1 + BR) d \log\left(1 + \frac{n}{8d(1 + BR)}\right) + B^2 R^2.$$

We obtain for AOSMP a regret upper-bound similar to the one of AIOLI [Jézéquel et al., 2020], as we use the same quadratic approximation (1.7).

Overall, in this work, we propose and analyze two SMP-inspired [Mourtada and Gaïffas, 2019] algorithms for online logistic regression: for OSMP, we only found an upper-bound for instant regret $\widehat{r}_t \leqslant e \cdot \sigma'(\langle \theta_t, x_t \rangle) \cdot \|x_t\|^2_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}}$; for AOSMP, we proved a regret upper-bound in a similar order with the state-of-the-art algorithm [Jézéquel et al., 2020] for binary online logistic regression to the best of our knowledge. The question of an efficient algorithm with better regret upper-bound for online logistic regression remains open.

Bibliography

N. Agarwal, S. Kale, and J. Zimmert. Efficient methods for online multiclass logistic regression. *arXiv preprint arXiv:2110.03020*, 2021. 29

A. S. Aguade, C. Gastaldi-Ménager, P. Fontaine, D. Karsenty, A. Fagot-Campagna, and C. Amadou. Time trends in diabetes medication prescription and factors associated with metformin discontinuation in people with newly diagnosed type 2 diabetes: A national population-based study. *Diabetic Medicine*, 38, 2020. 11

Z. Allen-Zhu, Y. Li, and Z. Song. A Convergence Theory for Deep Learning via Over-Parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019. 18, 19

Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural computation*, 9(7):1545–1588, 1997. 20

F. Bach. Self-concordant analysis for logistic regression. *Electronic Journal of Statistics*, 4(none):384 – 414, 2010. doi: 10.1214/09-EJS521. URL https://doi.org/10.1214/09-EJS521. 6, 30

E. Bacry, S. Gaiffas, F. Leroy, M. Morel, D.-P. Nguyen, Y. Sebiat, and D. Sun. Scalpel3: a scalable open-source library for healthcare claims databases. *International Journal of Medical Informatics*, page 104203, 2020. 9, 12

D. Bellamy, L. Celi, and A. L. Beam. Evaluating progress on machine learning for longitudinal electronic healthcare data. *Technical Report*, 2020. 9

J. Bezin, M. Duong, R. Lassalle, C. Droz, A. Pariente, P. Blin, and N. Moore. The national healthcare system claims databases in france, sniiram and egb: powerful tools for pharmacoepidemiology. *Pharmacoepidemiology and drug safety*, 26(8): 954–962, 2017. 11

R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. 8, 11

O. Bousquet, S. Boucheron, and G. Lugosi. Introduction to statistical learning theory. In *Summer school on machine learning*, pages 169–207. Springer, 2003. 26

L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 19

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 19, 20, 21, 22

L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and regression trees. wadsworth int. *Group*, 37(15):237–251, 1984. 20

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot

learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf. 8

M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 18

O. Catoni. *Statistical Learning Theory and Stochastic Optimization: Ecole d'Eté de Probabilités de Saint-Flour XXXI - 2001*, volume 1851 of *Lecture Notes in Mathematics*. Springer-Verlag Berlin Heidelberg, 2004. ISBN 9783540445074. URL https://books.google.fr/books?id=3Ih8CwAAQBAJ. 23

N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006. 22, 25, 26

N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9): 2050–2057, 2004. 27, 28

T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016. 23

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pages 1597–1607. PMLR, 2020. 8, 18

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 14

C.-Y. Chuang, J. Robinson, Y.-C. Lin, A. Torralba, and S. Jegelka. Debiased contrastive learning. In *NeurIPS*, 2020. 19

M. Cuggia, D. Polton, G. Wainrib, and S. Combes. Health data hub - mission de préfiguration. *Technical Report*, Oct. 2018. URL https://www.vie-publique.fr/rapport/37719-health-data-hub-mission-de-prefiguration. 12

A. S. Dalalyan and A. B. Tsybakov. Aggregation by exponential weighting and sharp oracle inequalities. In *International Conference on Computational Learning Theory*, pages 97–111. Springer, 2007. 23

J. Devlin, M.-W. Chang, K. Lee, and K. N. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *North American Association for Computational Linguistics (NAACL)*, 2018. URL https://arxiv.org/abs/1810.04805. 8

FDA. Real-world data: Assessing electronic health records and medical claims data to support regulatory decision-making for drug and biological products – guidance for industry. *Technical Report*, Sept. 2021. URL https://www.fda.gov/regulatory-information/search-fda-guidance-documents/real-world-data-assessing-electronic-health-records-and-medical-claims-data-support-r
8

S. F. Feldman, T. Lesuffleur, V. Olié, C. Gastaldi-Ménager, Y. Juillière, and P. Tuppin. French annual national observational study of 2015 outpatient and inpatient healthcare utilization by approximately half a million patients with previous heart failure diagnosis. *Archives of Cardiovascular Diseases*, 114(1):17–32, 2021. ISSN 1875-2136. doi: https://doi.org/10.1016/j.acvd.2020.05.009. URL https://www.sciencedirect.com/science/article/pii/S1875213620301613. 11

D. J. Foster, S. Kale, H. Luo, M. Mohri, and K. Sridharan. Logistic regression: The importance of being improper. In S. Bubeck, V. Perchet, and P. Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 167–208. PMLR, 06–09 Jul 2018. URL http://proceedings.mlr.press/v75/foster18a.html. 29

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1): 119–139, 1997. 19

J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001. 19

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006. 22

Y. Goldberg and O. Levy. word2vec explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014. 16

J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent - a new approach to self-supervised learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/f3ada80d5c4ee70142b17b8192b2958e-Paper.pdf. 18

H. Harutyunyan, H. Khachatrian, D. C. Kale, G. Ver Steeg, and A. Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 6(1):96, 2019. ISSN 2052-4463. doi: 10.1038/s41597-019-0103-9. URL https://doi.org/10.1038/s41597-019-0103-9. 9

E. Hazan. Introduction to online convex optimization. *arXiv preprint arXiv:1909.05207*, 2019. 25, 26, 27

E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007. 28, 29

E. Hazan, T. Koren, and K. Y. Levy. Logistic regression: Tight bounds for stochastic and online optimization. In *Conference on Learning Theory*, pages 197–209. PMLR, 2014. 28

K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020. 8, 17, 18

O. J. Hénaff, A. Srinivas, J. De Fauw, A. Razavi, C. Doersch, S. Eslami, and A. v. d. Oord. Data-efficient image recognition with contrastive predictive coding. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4182–4192. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/henaff20a.html. 17

T. K. Ho. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844, 1998. 20

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 14

R. Jézéquel, P. Gaillard, and A. Rudi. Efficient improper learning for online logistic regression. In *Conference on Learning Theory*, pages 2085–2108. PMLR, 2020. 6, 29, 31

R. Jézéquel, P. Gaillard, and A. Rudi. Mixability made efficient: Fast online multiclass logistic regression. working paper or preprint, Oct. 2021. URL https://hal.archives-ouvertes.fr/hal-03370530. 29

A. Johnson, L. Bulgarelli, T. Pollard, S. Horng, L. A. Celi, and R. Mark. "MIMIC-IV" (version 0.4). *PhysioNet*, 2020. URL https://doi.org/10.13026/a3wn-hq05. 9

A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-Wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark. MIMIC-III, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016. 9, 11

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. LightGBM: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017. 23

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 8

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. 8

Y. Li, S. Rao, J. R. A. Solares, A. Hassaine, R. Ramakrishnan, D. Canoy, Y. Zhu, K. Rahimi, and G. Salimi-Khorshidi. BEHRT: transformer for electronic health records. *Scientific reports*, 10(1):1–12, 2020. 10

Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=BJC_jUqxe. 14

G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts. Understanding variable importances in forests of randomized trees. *Advances in neural information processing systems 26*, 2013. 22

D. Luo, H. Xu, and L. Carin. Interpretable ICD code embeddings with self-and mutual-attention mechanisms. *arXiv preprint arXiv:1906.05492*, 2019. 10, 14

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a. 16

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b. 16, 17

M. Morel, E. Bacry, S. Gaïffas, A. Guilloux, and F. Leroy. ConvSCCS: convolutional self-controlled case-seris model for lagged adverser event detection. *Biostatistics*, 2019. doi: 10.1093/biostatistics/kxz003. URL https://hal.archives-ouvertes.fr/hal-02413920. 10

J. Mourtada and S. Gaïffas. An improper estimator with optimal excess risk in misspecified density estimation and logistic regression. *arXiv preprint arXiv:1912.10784*, 2019. 5, 6, 29, 31

A. Neumann, A.Weill, P. Ricordeau, J. Fagot, F. Alla, and H. Allemand. Pioglitazone and risk of bladder cancer among diabetic patients in france: a population-based cohort study. *Diabetologia*, 55:1953–1962, 2012. doi: https://doi.org/10.1007/s00125-012-2538-9. 11

A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 17

F. Orabona. A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*, 2019. 25, 26, 27

L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. Catboost: unbiased boosting with categorical features. *arXiv preprint arXiv:1706.09516*, 2017. 23

A. Rajkomar, E. Oren, K. Chen, A. M. Dai, N. Hajaj, M. Hardt, P. J. Liu, X. Liu, J. Marcus, M. Sun, et al. Scalable and accurate deep learning with electronic health records. *NPJ Digital Medicine*, 1(1):1–10, 2018. 10

S. Ren, X. Cao, Y. Wei, and J. Sun. Global refinement of random forest. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 723–730, 2015. 22

N. Saunshi, O. Plevrakis, S. Arora, M. Khodak, and H. Khandeparkar. A theoretical analysis of contrastive unsupervised representation learning. In *International Conference on Machine Learning*, pages 5628–5637, 2019. 3, 18, 19

M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997. 14

S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014. 26

B. Shickel, P. J. Tighe, A. Bihorac, and P. Rashidi. Deep EHR: a survey of recent advances in deep learning techniques for electronic health record (EHR) analysis. *IEEE journal of biomedical and health informatics*, 22(5):1589–1604, 2017. 10

Y. Tian, O. J. Henaff, and A. v. d. Oord. Divide and contrast: Self-supervised learning from uncurated data. *arXiv preprint arXiv:2105.08054*, 2021. 19

C. Tosh, A. Krishnamurthy, and D. Hsu. Contrastive estimation reveals topic posterior information to linear models. *Journal of Machine Learning Research*, 22(281):1–31, 2021. 19

A. B. Tsybakov. Optimal rates of aggregation. In *Learning theory and kernel machines*, pages 303–313. Springer, 2003. 4, 24

P. Tuppin, J. Rudant, P. Constantinou, C. Gastaldi-Ménager, A. Rachas, L. de Roquefeuil, G. Maura, H. Caillol, A. Tajahmady, J. Coste, C. Gissot, A. Weill, and A. Fagot-Campagna. Value of a national administrative database to guide public decisions: From the système national d'information interrégimes de l'assurance maladie (SNIIRAM) to the système national des données de santé (SNDS) in france. *Revue d'Épidémiologie et de Santé Publique*, 65: S149–S167, 2017. ISSN 0398-7620. doi: https://doi.org/10.1016/j.respe.2017.05.004. URL https://www.sciencedirect.com/science/article/pii/S0398762017304315. Réseau REDSIAM. 9, 10, 11

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 14

C. Villani, M. Schoenauer, Y. Bonnet, C. Berthet, A.-C. Cornut, F. Levin, and B. Rondepierre. Donner un sens à l'intelligence artificielle : pour une stratégie nationale et européenne. *Technical Report*, Mar. 2018. URL https://www.vie-publique.fr/rapport/37225-donner-un-sens-lintelligence-artificielle-pour-une-strategie-nation. 8, 11

T. Wang and P. Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere, 2020. 18

C. Wei, K. Shen, Y. Chen, and T. Ma. Theoretical analysis of self-training with deep networks on unlabeled data. In *International Conference on Learning Representations*, 2020. 19

C. Xiao, E. Choi, and J. Sun. Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. *Journal of the American Medical Informatics Association*, 25(10):1419–1428, 2018. 10

T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116, 2004. 28

S. Zhou and L. Mentch. Trees, forests, chickens, and eggs: When and why to prune trees in a random forest. *arXiv preprint arXiv:2103.16700*, 2021. 22

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pages 928–936, 2003. 28, 29

Chapter 2

ZiMM: a deep learning model for long term and blurry relapses with non-clinical claims data [1]

" 吾生也有涯，而知也无涯。
There is a limit to our life, but to knowledge there is no limit. "

Zhuang Zhou (translated by James Legge)

Contents

[1] This chapter is based on the joint work with A.Kabeshova, S. Gaiffas, E. Bacry, B. Lukacs.

Abstract

This work considers the problems of modeling and predicting a long-term and "blurry" relapse that occurs after a medical procedure, such as a surgery. We do not consider a short-term complication related to the act itself, but a long-term relapse that clinicians cannot explain easily, since it depends on unknown sets or sequences of past events that occurred before the act. The relapse is observed only indirectly, in a "blurry" fashion, through longitudinal prescriptions of drugs over a long period of time after the medical act. We introduce a new model, called ZiMM (Zero-inflated Mixture of Multinomial distributions) in order to capture long-term and blurry relapses. On top of it, we build an end-to-end deep-learning architecture called ZiMM Encoder-Decoder (ZiMM ED) that can learn from the complex, irregular, highly heterogeneous and sparse patterns of health events that are observed through a claims-only database. ZiMM ED is applied on a "non-clinical" claims database, that contains only timestamped reimbursement codes for drug purchases, medical procedures and hospital diagnoses, the only available clinical feature being the age of the patient. This setting is more challenging than a setting where bedside clinical signals are available. Our motivation for using such a non-clinical claims database is its exhaustivity population-wise, compared to clinical electronic health records coming from a single or a small set of hospitals. Indeed, we consider a dataset containing the claims of almost *all French citizens* who had surgery for prostatic problems, with a history between 1.5 and 5 years. We consider a long-term (18 months) relapse (urination problems still occur despite surgery), which is blurry since it is observed only through the reimbursement of a specific set of drugs for urination problems. Our experiments show that ZiMM ED improves several baselines, including non-deep learning and deep-learning approaches, and that it allows working on such a dataset with minimal preprocessing work.

2.1 Introduction

The increasing volume of Electronic Health Records (EHR) systems of national medical organizations in recent years allows to capture data from millions of individuals over many years. Each individual's EHR can link data from many sources and hence contain "concepts" such as diagnoses, interventions, lab tests, clinical narratives, and more. This provides great opportunities for data scientists to collaborate on different aspects of healthcare research by applying advanced analytics to these EHR clinical data [Rajkomar et al., 2018].. There are several challenges in processing EHR data [Cheng et al., 2016]: data quality, high-dimensionality, temporality which refers to the sequential nature of clinical events, sparsity in both medical codes representation and in timestamp representation, irregularly timed observations, biases such as systematic errors in data collection, and mixed data types with missing data. Representation learning can overcome these challenges [Rajkomar et al., 2018] and the choice of data representation or feature representation plays a significant role in the success of this approach.

2.1.1 Related works

Recently, much work has been done on developing compact and functional representations of medical records, including the use of deep learning over EHR data [Bajor et al., 2018; Shickel et al., 2018]. A notable attempt is stacked denoising autoencoders [Miotto et al., 2016]. Denoising autoencoders are also used in [Beaulieu-Jones and Greene, 2016] to develop patient representation from various binary clinical descriptors on synthetic data. Sums of word-level skip-gram embedded vectors of clinical codes are used in Choi et al. [2016d] to create full-record representations. Word-level semantic embeddings for diagnosis and intervention codes are constructed in Pham et al. [2017], using pooling and concatenation to aggregate embeddings into a vector representing a single admission, while Nguyen et al. [2017] uses word-level embedding as preprocessing for a CNN architecture.

Only a few studies were made based on claims data due to its complexity and rare availability. Choi et al. learned distributed representations of medical codes (e.g. diagnoses, medications, procedures) from electronic health records (EHRs) and claims data using Skip-gram and applied them to predict future clinical codes and risk groups Choi et al. [2016b]. Cui2vec is a recent study in learning clinical concept embeddings [Beam et al., 2018], which applied word2vec [Mikolov et al., 2013] and Glove [Pennington et al., 2014] on multiple medical resources such as structured claims data, biomedical journal articles and unstructured clinical notes. Xiao et al. [2018] proposed learning a distributional representation of clinical concepts considering temporal dependencies along the longitudinal sequence of a patient's records based on claims data.

Other efforts have aimed at encoding temporal aspects of EHR data for predictive tasks. In Choi et al. [2016a], time-stamped events are used as inputs to a particular type of RNNs to predict future disease diagnosis, while a graph-based attention model is used in Choi et al. [2019] to learn concept representations. Another interesting recent effort is Rajkomar et al. [2018], which maps raw EHR data to the FHIR format (Fast Healthcare Interoperability Resources [Bender and Sartipi, 2013]) to encode EHR information for several different sequence-oriented models.

RETAIN [Choi et al., 2016c] and GRAM [Choi et al., 2017] are two state-of-the-art models using RNNs for predicting future diagnoses. However, they cannot handle long sequences effectively. LSTM or bidirectional RNNs [Ma et al., 2017] can be trained using all available input information in the past and future, and have been used to alleviate the effect of the long sequence problem and improve the predictive performance. The work of [Lipton et al., 2016] has been the first to successfully use LSTM to predict patients diagnosis, for multi-label classification. BEHRT [Li et al., 2019] is one of the most recent researches that provides the field with an accurate predictive model for the prediction of next diseases based on the transformer-based architecture in NLP [Devlin et al., 2018].

Although fixed timesteps are perfectly suitable for many RNN applications, EHRs often contain event-driven and asynchronously sampled samples, in particular for the application considered in the present paper. Recent works try to capture time irregularity using recurrent neural networks. Phased LSTM [Neil et al., 2016] tries to model the time information by adding a time gate to LSTM, which controls the update of the cell state, the hidden state and thus the final output. Another attempt is Zhu et al. [2017] where time gates are added to the LSTM in order to model time intervals and specifically better capture both of short-term and long-term sequential events. Time-aware LSTM [Baytas et al., 2017] addresses

time irregularity between two events in an LSTM architecture by decomposing the memory of the previous timesteps into short-term memory and long-term memory. Let us point out that, however, most of the works on EHR data cited above either ignores subsequence-level irregularity by partitioning the data into regular time windows and by aggregating the data within each window or handles this irregularity by simply adding the time-span as one coordinate of the features vector, see Choi et al. [2016a] for instance.

2.1.2 Aim of this chapter and contributions

This chapter aims at the construction of a predictive model for long-term and "blurry" relapses that occur after a medical act. We do not consider a short-term complication related to the act itself, but a long-term relapse which is observed only indirectly. In the example considered here, the medical act is a precise surgery, and the relapse is observed through longitudinal prescriptions of a specific set of drugs for urinary problems, over a period of 18 months after surgery. We use all the data available in SNIIRAM (French national health insurance information system, a huge database containing health reimbursements claims of almost all French citizens since 2015, see Section 2.3.1 for details) for patients on which such a surgery has been performed (TURP surgery, see Section 2.3.2).

The ZiMM model. The first natural idea is to cast this problem as a classification problem for "relapse" versus "no relapse" after the medical act. However, such a naive approach cannot work here. Indeed, the definition of a binary label would require threshold choices, both about time and dosage: after how many time and amount of drugs prescribed do we consider that there is a relapse? Such thresholds might depend on various clinical practices, habits of the patient, and many other exogenous factors. In this work, we propose to work directly on the data observed, by using all the "blurry" observations at once to train an end-to-end architecture. For this purpose, we introduce a new methodological contribution, namely the ZiMM model (Zero-inflated Multinomial Mixture) which is described in Section 2.2.1 below.

The ZiMM Encoder-Decoder (ZiMM ED). We introduce an end-to-end Encoder-Decoder architecture trained with an objective, based on the negative log-likelihood of the ZiMM model. All available patients' claims before the medical act are first encoded into a single embedding vector by the Encoder. This Encoder combines embedding layers, self-attention layers and recurrent layers to output an embedding vector of the full patient pathway prior to the medical act. This embedding is used as input to a Decoder which is dedicated to the learning of the parameters of the ZiMM model. This architecture is described in Sections 2.2.3 to 2.2.5 below.

Specifics of this work. The specificity of our approach lies in several points. First of all, it is performed on non-clinical EHR data, and we consider long-term predictions (18 months ahead). Our end-to-end architecture addresses several challenges, such as variable-size inputs, confounding interactions between medical codes, long-term predictions. Finally, we do not inject any prior knowledge: the embeddings of patient pathways are fully trained in an end-to-end fashion, we do not use

any strong preprocessing or simplifying aggregations, the data is used almost in its raw form. We keep the smallest granularity possible both on the codes (we use the actual medical codes instead of encompassing categories) and on time, namely we work on the original 1-day time scale.

Organization of the chapter. The ZiMM model is described in Section 2.2.1 while the ZiMM ED architecture is described in Sections 2.2.3 to 2.2.5. Some details concerning data preprocessing are provided in Section 2.2.2. Section 2.3 provides results and a comparison with strong baselines, together with some variations of the architecture to fine-tune the final performance and we conclude in Section 2.4.

2.2 Proposed architecture

The ZiMM ED architecture has three main building blocks: the ZiMM (Zero-Inflated Multinomial Mixture) model (see Section 2.2.1), an Encoder (see Section 2.2.3) and a Decoder (see Section 2.2.4), and is described in Figure 2.1 below. Given a patient $i \in \{1, \dots, n\}$ (among n patients), with medical act of interest occurring at time T^i, the steps through this architecture are roughly as follows:

- All the medical codes (drugs, diagnosis, medical acts) observed longitudinally in the life of patient i before T^i are embedded, then aggregated in a time window using a self-attention mechanism. The time distance (in days) to T^i of each day with a non-empty set of codes is embedded as well, and we embed as well hospital stay durations. This sequence of vectors is then fed to a (or several) recurrent layers. This corresponds to Steps 1,2 and 3 in Figure 2.1. This leads to an embedding vector $x_i \in \mathbb{R}^d$ of the full pathway of patient i prior to T^i.
- The vector $x_i \in \mathbb{R}^d$ is concatenated with other static features of the patient, and used as the input of the Decoder, which is based on several recurrent layers, in order to produce the parameters of the ZiMM model. This corresponds to Step 4 in Figure 2.1.

Precise descriptions of all steps are provided below, starting with the ZiMM model, followed by some details concerning data preprocessing, and the Encoder-Decoder architecture.

2.2.1 ZiMM: Zero-inflated Multinomial Mixture model

Assume for now that we have an embedding vector $x_i \in \mathbb{R}^d$ that encodes the full health pathway of patient $i \in \{1, \dots, n\}$ prior to T^i. This vector is the output of the ZiMM-Encoder described in Section 2.2.3 below. Moreover, we observe a vector of labels $y_i = [y_{i,1}, \dots, y_{i,B}] \in \mathbb{N}^B$, which corresponds to the blurry observation of the relapse after T^i. Here, B corresponds to the number of time intervals considered after T^i and $y_{i,b} \geqslant 1$ is the number of blurry relapses observed in time bucket b, so that $y_{i,b} = 0$ means that no blurry relapse is observed in time bucket b. In the example considered Section 2.3 below, $B = 18$ corresponds to a 18-months period and $y_{i,b}$ is the number of drugs (among a set of drugs for urinary problems, see Section 2.3.3), purchased by patient i in time bucket b.

We introduce $n_i = \sum_{b=1}^{B} y_{i,b} \in \mathbb{N}$, the overall number of blurry relapses of patient i. Let us point out that a binary classification problem ($n_i > 0$ versus

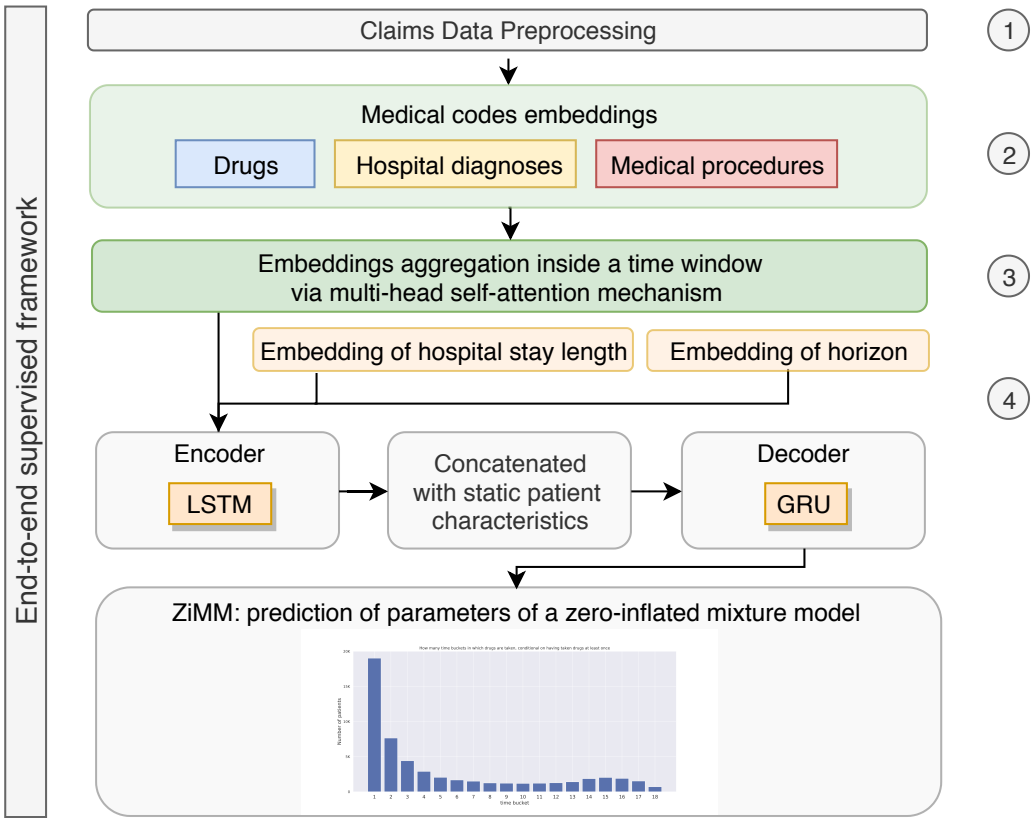

Figure 2.1 – Architecture of the ZiMM Encoder-Decoder end-to-end architecture. A detailed graphical representation of Steps 2-4 is shown in Figure 2.3 below.

$n_i = 0$) may wrongly classify very different situations corresponding to the same n_i. Consider for instance two extreme situations where first, we have $n_i = 1$ with $y_{i,1} = 1$ (and consequently $y_{i,b} = 0$ for $b = 2, \dots, B$), and second, we have $n_i = 1$ with $y_{i,B} = 1$ (and consequently $n_{i,b} = 0$ for $b = 1, \dots, B-1$). The first case can be due to exogenous factors, such as the patient simply kept buying the drug after the surgery just out of a pure habit (prescriptions can run for a long period), while in the second example, there is no doubt that a relapse is occurring.

Therefore, we need to find a way to model the whole vector y_i, so that n_i is not fixed and includes zero-inflation, namely a parametrized likelihood for $n_i = 0$. For that purpose, we introduce the following ZiMM model, where we suppose that $n_i \in \{0, 1, \dots, B\}$ is distributed (conditionally to x_i) as

$$n_i \sim \text{Categorical}(\pi_0(x_i), \pi_1(x_i), \dots, \pi_B(x_i)),$$

which means that

$$\mathbb{P}(n_i = k | x_i) = \pi_k(x_i) \text{ for } k \in \{0, \dots, B\},$$

where $\pi_b(x_i)$ are such that $\sum_{b=0}^{B} \pi_b(x_i) = 1$ and $\pi_b(x_i) \geqslant 0$ (coming out of a softmax activation for instance). These parameters correspond to the categorical distribution specific to patient i, and are constructed by the ZiMM-Decoder (see Section 2.2.4 below) from x_i. Then, we assume that the distribution of y_i conditional to $n_i = b$ and x_i follows either a Dirac distribution on vector (of size B) $[0, \dots, 0]$ whenever $b = 0$, or a multinomial distribution of parameters b and

$p_{b,1}(x_i), \ldots, p_{b,B}(x_i)$, namely

$$y_i|(x_i, n_i = b) \sim \begin{cases} \delta_{[0,\ldots,0]} & \text{if } b = 0, \\ \text{Multinomial}(b, p_{b,1}(x_i), \ldots, p_{b,B}(x_i)) & \text{otherwise,} \end{cases}$$

where $p_{b,1}(x_i), \ldots, p_{b,B}(x_i)$ are the parameters of a multinomial distribution whenever $n_i = b$, for each $b = 1, \ldots, B$. Once again, these parameters are specific to patient i thanks to the embedding vector $x_i \in \mathbb{R}^d$, and are constructed by the ZiMM-Decoder. These parameters must satisfy $\sum_{b'=1}^{B} p_{b,b'}(x_i) = 1$ for $b = 1, \ldots, B$, and $p_{b,b'}(x_i) \geqslant 0$ for any $b, b' = 1, \ldots, B$, which is easily achieved with a softmax activation applied row-wise. The distribution of y_i conditionally on x_i is therefore given by the following mixture model:

$$y_i|x_i \sim \pi_0(x_i)\delta_{[0,\ldots,0]} + \sum_{b=1}^{B} \pi_b(x_i)\text{Multinomial}(b, p_{b,1}(x_i), \ldots, p_{b,B}(x_i)).$$

Zero-inflation corresponds to the case where $n_i = 0$ and has likelihood measured by $\pi_0(x_i)$. The ZiMM model has two sets of parameters that are functions of the feature vector x, namely $\{\pi_b(x)\}_{b\in\{1,\ldots,B\}}$ that parametrizes the distribution of the total number of blurry relapses and $\{p_{b,b'}(x)\}_{b,b'\in\{1,\ldots,B\}^2}$ that parametrizes the dynamics of these relapses. As explained in Section 2.2.4 below, the dynamics of $\{p_{b,b'}(x)\}_{b'\in\{1,\ldots,B\}}$ for each b are learned by dedicated recurrent layers in ZiMM-Decoder.

2.2.2 Preprocessing of the data

After a preliminary preprocessing based on our SCALPEL3 library described in Bacry et al. [2019], the data considered in this paper comes in the form of a table, where each row represents an event with the patient ID, the type of event (drug, diagnosis or medical procedure), its corresponding code, the start date and the end date of the event. Figure 2.2 provides an illustration of the sequence of claims observed for a single patient.

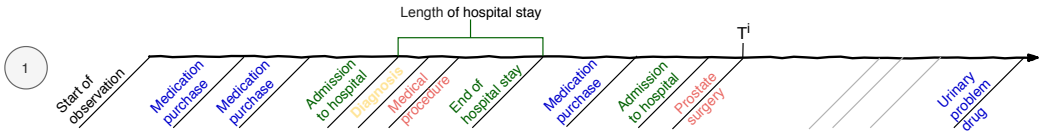

Figure 2.2 – Illustration of the sequence of claims observed for a patient. This corresponds to the output of Step 1 from Figure 2.1. Through claims, we observe three types of events: drug purchases (blue), medical procedures (red) and diagnosis (yellow) before the medical act (prostate surgery in the example) that happens at time T^i for patient i. All events are timestamped, and the time delta is a day long. Several events can occur the same day and some days have no event: events are typically very irregularly sampled. All these events, observed before T^i, are used to learn the embedding vector $x_i \in \mathbb{R}^d$ of patient i. After T^i, we only keep the events corresponding to the blurry relapses considered (drug purchases among a set of drugs for urinary problems). These blurry relapses are used to build the label vector $y_i \in \mathbb{N}^B$ of patient i.

If an event is related to a hospital stay, the start date is the first day of hospitalization while the end is the exit date from the hospital. Otherwise, the start

and end dates are the same. The time delta is a day-long, and no aggregation is performed to keep the data as raw as possible.

Multiple events can occur within the same day, and the precise ordering of such events does not carry any information. Thus, we shuffle at random within-day events, to avoid any bias that could occur from the data collection mechanism (a similar strategy can be found in Choi et al. [2016d]).

All the events observed through claims before T^i are used as inputs to the Encoder described in Section 2.2.3 below. The output of the Encoder is the embedding vector $x_i \in \mathbb{R}^d$ of patient i. The blurry relapses observed after T^i are used to build the label vector $y_i \in \mathbb{N}^B$ of patient i.

2.2.3 The ZiMM Encoder

The ZiMM Encoder corresponds to Steps (2) to (4) in Figure 2.1, a more detailed illustration is shown in Figure 2.3. Let us provide now details about each step, following the flow of the data.

Medical codes and timestamps embeddings. The following is applied separately for each type of event code (drugs, medical procedures and diagnoses). Codes are first tokenized, and each unique token is individually mapped to an embedding vector in $\mathbb{R}^{d_E}$ which is learned during training, where d_E is a hyper-parameter to be tuned (see Section 2.3.7). We consider only tokens that occur at least 50 times in the training dataset. For any event occurring at time $t \leqslant T^i$ in the observation period of patient i, we compute the "time horizon" as $T^i - t$, namely the distance (in days) between the event and medical act. Moreover, whenever it makes sense, we compute end $-$ start, the duration of the event, which corresponds to the duration of an hospital stay defined as the time between hospital admission and discharge. These two integers (in number of days) are also tokenized and replaced by a learned embedding vector. This allows the encoder to learn to put more or less emphasis on events that are close or far from T^i, and to exploit the duration of events as a proxy for severity of medical procedures and diagnoses. The patient age in years at T^i is also similarly bucketized and embedded. The embedding of medical codes corresponds to Step (2) of both Figures 2.1 and 2.3. In our experiments, the default model uses $d_E = 64$ for the three types of codes and several regularization strategies are applied to avoid overfitting: weights decay (ℓ_2 regularization with strength 10^{-2}), multiplicative Gaussian noise, and layer normalization [Ba et al., 2016]. Since we observed empirically that these embeddings are prone to overfitting for the application considered in the paper, a comparison of the different regularization strategies is provided in Section 2.3.7.

Embeddings aggregation through self-attention. Several events can occur at the same time (within the same day), so that the number of codes observed within a day is highly heterogeneous. Moreover, such codes are not likely to contribute equally to the vector representation of the day, and their order is not informative. Therefore, we need to perform a trainable aggregation of these codes in order to produce a representation of the patient history at each timestamp. An approach can consist of using a hierarchy relationship between different diagnosis and treatment inside each patient visit [Choi et al., 2019]. However, in the data considered here, diagnosis codes are not explicit each time there is treatment; in particular, drugs

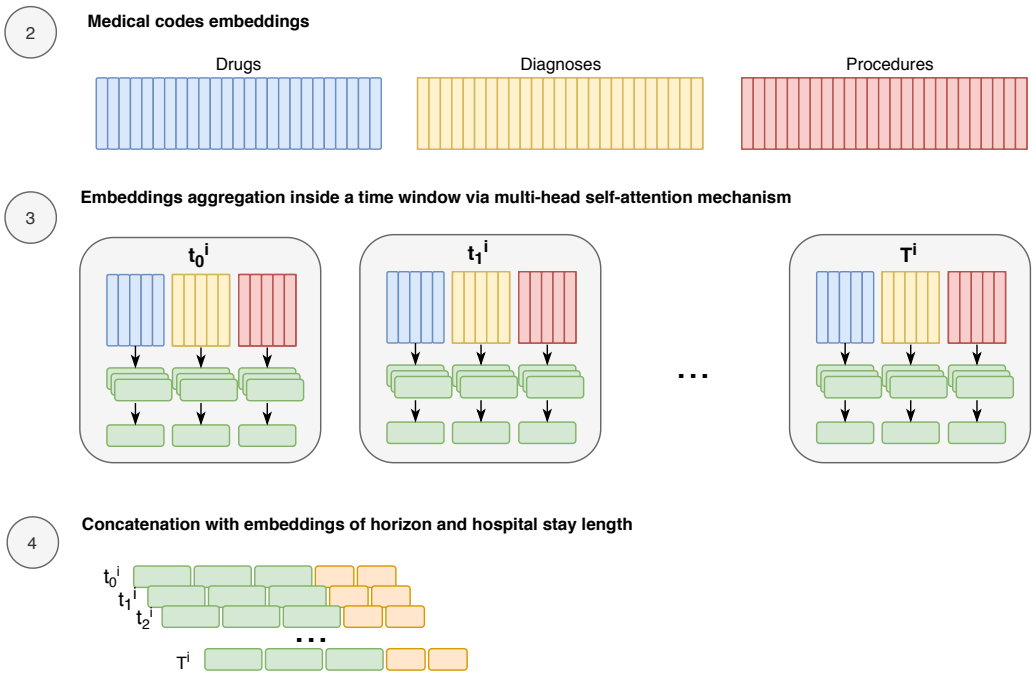

Figure 2.3 – A graphical illustration of representation learning of patient pathway in the ZiMM Encoder: (2) embedding tensors for each type of medical code; (3) aggregation of embedding vectors through multi-head self-attention inside a time window, where each gray block corresponds to one day $(t_0^i, t_1^i, \ldots)$, and T^i is the day of the medical act of interest; (4) concatenation of aggregated embedding vectors with embeddings for time horizon and hospital length stay.

purchases are almost never related to a diagnosis code (unless they are related to a hospitalization). Hence, we use a self-attention mechanism [Lin et al., 2019; Vaswani et al., 2017] using a *bag-of-features* approach to learn how to combine embedding vectors within the same day, following previous successful applications of self-attention for fusing disease embeddings [Luo et al., 2019].

Let C be a set of codes with cardinality $|E|$ (for drugs, medical procedures or diagnoses) and let $E_C \in \mathbb{R}^{d_E \times |C|}$ be the corresponding matrix of concatenated embedding vectors. The multi-headed self-attention aggregation mechanism contains two layers, the first with K heads, each of which is a self-attention function that generates a probability vector of size $|E|$ from E_C:

$$w_k = \text{softmax}(\alpha_k^\top \tanh(A_k E_C)), \tag{2.1}$$

for $k = 1, \ldots, K$, where $\alpha_k \in \mathbb{R}^{1 \times d_E}$ and $A_k \in \mathbb{R}^{d_E \times d_E}$ are to be trained and the second layer outputs a d_E-dimensional vector given by

$$E_C \mu_C \quad \text{where} \quad \mu_C = \text{softmax}(b^\top \tanh(BW)), \tag{2.2}$$

with $W = \text{concatenate}(w_1, \ldots, w_k) \in \mathbb{R}^{K \times |C|}$ and b and B are trainable parameters. This self-attention mechanism is trained and performed separately for each type of code, as displayed in Step (3) of Figure 2.3. Note that this aggregation approach is permutation-invariant for codes that occur within the same day. We use dropout, drop-connect [Wan et al., 2013], Frobenius-norm weight penalization from [Lin et al., 2019] and weight-decay to regularize this self-attention mechanism.

Health pathway encoder. At each timestamp, the following embedding vectors are concatenated:

- Medical code embedding computed by Equations (2.1) and (2.2);
- Time-horizon embedding (see above);
- Duration of the event embedding (see above).

For each patient, this leads to a sequence of fixed-sized embedding vectors that encode both medical and time information at each (non-empty) timestamp $t \leqslant T^i$, as displayed in Step (4) of Figures 2.1 and 2.3. Now, this sequence of vectors is used as the input of a stack of layers, including recurrent layers (LSTM [Hochreiter and Schmidhuber, 1997], bi-directional LSTM [Schuster and Paliwal, 1997], GRU [Cho et al., 2014]) or convolutional layers, in a *sequence-to-one* network architecture, since we want to output a single vector $x_i \in \mathbb{R}^d$ that encodes the full pathway of a patient before T^i. The results obtained with different types of layers are shown in Section 2.3.7 below. We use again dropout on the inputs and on the recurrent units to prevent overfitting. Finally, the output vector of the encoder is concatenated with an embedding vector of the age of the patient, leading to the final vector $x_i \in \mathbb{R}^d$ that encodes the full pathway of patient i before T^i. This vector is the input of the ZiMM Decoder described in the next Section.

2.2.4 The ZiMM Decoder

The decoder uses the input vector $x_i \in \mathbb{R}^d$ of patient i to construct the parameters of the ZiMM model, and the whole architecture is trained against the negative log-likelihood of the ZiMM model from Section 2.2.1 computed at the label vector $y_i = [y_{i,1}, \ldots, y_{i,B}]$ containing the blurry relapses. The parameters $\{\pi_b(x_i)\}_{b=1\ldots,B}$ and $\{p_{b,b'}(x_i)\}_{b,b' \in \{1,\ldots,B\}^2}$ are highly dependent and are time-ordered, so that a specific architecture is used to model these dependencies. The mixture probabilities are learned through a fully connected feed-forward layer (FFN) that use as input x_i, namely

$$\pi_0(x_i), \ldots, \pi_B(x_i) = \text{softmax}(\text{FFN}(x_i)),$$

and another recurrent layer (RNN) uses as well x_i in order to produce hidden states given by

$$h_t = \text{RNN}(h_{t-1}, x_i) \tag{2.3}$$

for $t = 1, \ldots, B$. Then, we use different recurrent layers (RNN_b) in parallel for each value $n_i = b \in \{1, \ldots, B\}$ using

$$h_t^b = \text{RNN}_b(h_{t-1}^b, h_t)$$

for $t = 1, \ldots, B$, since the multinomial distributions vary strongly conditionally on $n_i = b$. Finally, a softmax activation is applied on h_t^b along $t = 1, \ldots, B$ to produce the parameters $p_{b,1}(x_i), \ldots, p_{b,B}(x_i)$. Details on the type of layers used and the tuning of hyperparameters is provided in Section 2.3 below.

2.2.5 Training

The full ZiMM Encoder-Decoder architecture is trained in an end-to-end fashion by minimizing the average negative-log likelihood over all patients $i = 1, \ldots, n$:

$$\ell_i(\Theta) = \log(\pi_0(x_i))\mathbf{1}_{n_i=0} + \sum_{b=1}^{B}\Big[\log(\pi_b(x_i)) + \log\Big(\frac{b!}{\Pi_{b'=1}^{B} y_{i,b'}!}\Big) + \sum_{b'=1}^{B} y_{i,b'}\log(p_{b,b'}(x_i))\Big]\mathbf{1}_{n_i=b},$$

where Θ stands for the concatenation of all the trainable parameters involved in the layers described in Sections 2.2.3 and 2.2.4. The choices of optimizer, learning rate schedule and other hyperparameters are described in Section 2.3.

2.3 Application: prediction of post-surgical relapse of urinary problems

In this section, we apply the ZIMM ED architecture to predict the blurry relapse of urinary problems after a TURP surgery for patients with benign prostatic hyperplasia (see Section 2.3.2 below) using a cohort constructed from the French SNIIRAM database (see Section 2.3.1 below). Then, we explain in details the way the labels are built (Section 2.3.3), the different steps involved in the cohort construction (Section 2.3.4), the evaluation metric and implementation details (Section 2.3.5). Then, the remaining of this section describes the baselines and a comparison with ZiMM ED, followed by an ablation study.

2.3.1 SNIIRAM: A non-clinical claims dataset

SNIIRAM (French national health insurance information system) contains health reimbursements claims of almost all French citizens since 2015 (more than 65 million) and has been used for health research on many topics, to cite but a few Atramont et al. [2018]; Fonteneau et al. [2017]; Scailteux et al. [2019]. Since, in France, most health-cares are at least partially reimbursed by the administration, this database contains records corresponding to very various information going from hospital stays to drug purchases in city pharmacies, all coded with different systems [Tuppin et al., 2017]. Absence of clinical information and "forced" normalization makes this data highly complex and heterogeneous. A consequence is that it requires a lot of domain expertise about the way healthcare is reimbursed in France in order to prepare it as training data for machine learning algorithms. In this paper, we exploit medication, procedure and diagnosis codes only. Diagnoses are primarily coded with ICD-10, while procedures are coded with CCAM, a French medical classification of clinical procedures. Medications codes use the French pharmaceutical categories CIP13 [Bezin et al., 2017] that we map to the international ATC (Anatomical Therapeutic Chemical Classification) classification system.

Let us stress that while this data is extremely rich and almost population-wide over France, it is not clinical data, but only claims data, clinical information being only latent in the codes appearing in reimbursement information. No vital bedside information, lab tests results or natural language notes from clinicians is available. An important amount of data preprocessing and domain expertise is required for data extraction, in particular in order to clean existing inaccuracies in codes and

timestamps. This is achieved through our SCALPEL3 library [Bacry et al., 2019], which is a separate topic of research that this paper builds upon. Furthermore, the identification of a disease state (indication, concomitant disease or outcome) does not rely only on a single source of information but on the convergence of information from different sources. This might include, for instance, the presence of a chronic disease registration, hospital diagnoses, tracer drugs, and procedures or lab tests [Bezin et al., 2017], that are all observed in the data through claims.

We believe that these downsides are counter-balanced by the exhaustive nature of this data. It is exhaustive both in terms of population (it includes nearly all the population living in France) and in terms of healthcare events. Indeed, in France, almost all healthcare spendings are at least partially reimbursed by the government, so almost each healthcare of each individual is associated to at least one event in the database. This makes it a very rich and powerful database, with a relatively small bias, and many works have already proved successful in improving population-level health [Atramont et al., 2018; Fonteneau et al., 2017; Morel et al., 2019; Scailteux et al., 2019]. Though most EHR studies are dealing with clinical EHR's Coorevits et al. [2013], it is clearly crucial to exploit such claims-only EHR, and we hope that our work is a first step towards broader use, in healthcare and machine learning research community, of non-clinical claims datasets.

Let us note that, since 2016, the access to this database, for public interest research, is possible through the SNDS access pipeline [SNDS, 2019]. It mainly offers access through the SAS software. An access using classical open source big data and AI frameworks (e.g. R, Python, Spark) including the SCALPEL3 library will be possible, by the end of 2020, through the Health Data Hub Hub [2019], a 80M$-funded national project which aims to be the national unique gateway to most French health data for operating public interest research (operated by both public or private entities) on modern infrastructures.

2.3.2 Benign prostatic hyperplasia (BPH)

BPH is a common urological disease that affects aging men all over the world [Dahm et al., 2017; Silva et al., 2014]. It causes urinary tract obstruction due to the unregulated growth of the prostate gland, causing lower urinary tract symptoms (LUTS) [Kim et al., 2016]. The options for management of BPH include watchful waiting, pharmacotherapy, transurethral resection of the prostate (TURP), and other minimally invasive surgical treatments (MISTs) and open prostatectomy. Most studies consider TURP as the gold standard for surgical management of BPH [Hashim and Abrams, 2015]. Patients suffering from prostatic hyperplasia regularly take drugs for urination problems. Successful surgery should cure such problems so that patients should not need to continue their drugs for urination problems. However, it is often observed that patients retake such drugs after surgery. In some cases, it merely comes from a habit of taking routine drugs, but in many cases, it is related to a persisting urination problem [Chung and Woo, 2018; Macey and Raynor, 2016].

An important problem, from a clinical point of view, is, therefore, to predict the outcome of such a surgery, on a long-time period following it (18 months is considered here), to improve the decision making of the clinicians, in particular, to help decide when surgery should be performed. Use of long-term data after TURP surgery is very scarce in the literature, with only a few studies available [Cornu

et al., 2015]. A recent analysis of 20 contemporary randomized clinical trials with a maximum follow-up of five years reports that TURP resulted in a significant improvement of maximum flow-rate and quality of life [Cornu et al., 2018; Lourenco et al., 2010]. A second prostatic operation (re-TURP) has been reported at a constant annual rate of approximately 1-2%. A review analyzing 29 RCTs found a re-treatment rate of 2.6% after a mean follow-up of sixteen months [Cornu et al., 2018]. In a large-scale study of 20,671 men, the overall re-treatment rates (i.e., either re-TURP, urethrotomy or bladder neck incision) were 5.8%, 12.3%, and 14.7%, respectively at one, five, and eight years follow-up. More specifically, the respective incidence of re-TURP was 2.9%, 5.8% and 7.4% [Madersbacher et al., 2005]. However, urology guidelines highlight the lack of extended follow-up after the surgery and no clear evidence explaining re-treatment.

Building a model with the ability to predict the outcome of this surgery is of primary importance for improving population-level health, especially since prostatic problems are a common condition for aging men. In this section, we use ZiMM ED to train such a model using a cohort based on SNIIRAM. The outcome of this surgery is evaluated by the distribution of the blurry relapses, observed through the use of specific drugs related to urination problems.

2.3.3 Surgery identification and labels construction

The identification of a TURP surgery in SNIIRAM is made through some specific CCAM codes provided by clinicians[2]. However, it happens that two TURP surgeries can occur in a pretty small amount of time. This is likely to correspond to a case where a surgery has clearly failed and that a second surgery is required quickly, which is not the type of complication we are interested in. If the amount of time between the two surgeries is small, it would be improper to say that it corresponds to a relapse. We thus define, following clinicians recommendations, a six week period after the first surgery, as a *single surgery block*. If another TURP surgery occurs inside the same block, we consider that it is part of the same medical act. The timestamp of the event corresponds to the last surgery within the block, and provides the value of T^i for any patient i.

As far as the labels are concerned, a simple but efficient way to identify whether the urinary problems have not ceased or reappeared after a while is to see if the patient, at some point after the surgery, needs to take medications for these urination problems again[3]. For that purpose, we use a list (provided by clinicians) of 136 CIP-13 drugs that are mainly related to urination problems. We choose to drive the prediction on a 18-months period after T^i. So, using the notation of Section 2.2.1, we chose the number of buckets $B = 18$ and bucket size of 30 days (540 days total).

2.3.4 Cohort construction and exploration

The construction of the cohort follows the flowchart illustrated in Figure 2.4. We start by selecting all SNIIRAM patients alive, with at least one medical event re-

[2]We considered that a TURP surgery corresponds to the CCAM codes JGFA005, JGFA009, JGFA015, JDPE002 or JGNE003.

[3]Following clinicians recommendations, we do not consider TURP surgeries that occur after the first surgery block, and consider only drugs prescriptions as blurry relapses. Consequently, we remove patients with repeated surgeries that are not part of the same block from the cohort construction, as explained in Section 2.3.4.

lated to urination problem (observed through medication or surgery codes) between 2010/01/01 and 2015/12/31. This selection gathers a little bit more than 5 million patients. Then, we keep only male patients over 18 years old on 2014/12/31. This reduces the size of the cohort by roughly 1 million patients. Keeping only patients with a TURP surgery between 2010/01/01 and 2013/06/30 and with at least 2 events (whatever the types) occurring in the follow-up 18-months period leads to a cohort containing 231 747 patients.

As explained in Section 2.3.3, we compute surgery blocks (any surgery that occurs within a six week period after the first surgery is in the same block), that are considered as a single event of surgery. We remove all the patients that have another surgery out of this first block. We choose a follow-up period of 540 days, corresponding to 18 buckets of 30 days, so that the *follow-up* period is of the order of 18 months. Indeed, clinicians expect that such a relapse should occur not long after the first year following surgery. As shown in Figure 2.4, the final cohort has 138 976 patients.

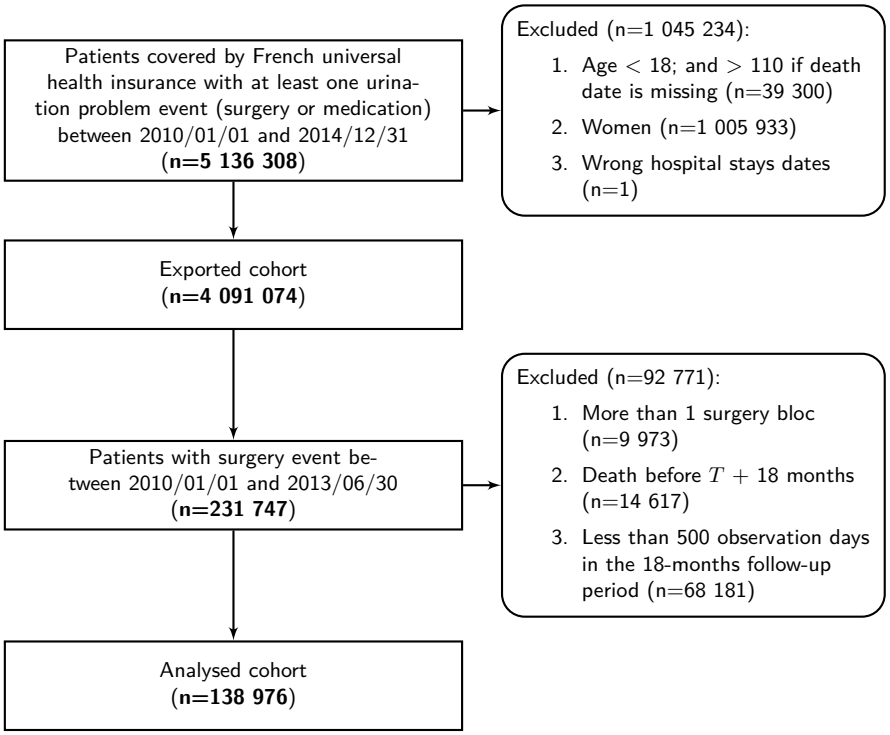

Figure 2.4 – Flowchart leading to the final cohort considered in the experiments.

The numbers of medical codes, namely medications, procedures and diagnoses observed in the final cohort, are displayed in Table 2.1. We count the number of unique codes, the number of codes observed at least 50 times and the number of codes remaining when keeping only the ones prior to T^i for each patient i. The statistics reported in Table 2.1 below and in all figures from this section use the raw coding scheme (that use CIP-13 for drugs), in order to better describe the statistics of the data, while for training ZiMM ED we replace CIP-13 by ATC (for drugs) since, as illustrated in Section 2.3.7 below, it allows to reduce the vocabulary size of drugs while keeping the same predictive performance.

On the left-hand side of Figure 2.5, we show, for this final cohort, the distri-

Table 2.1 – Number of medical codes observed and remaining when applying the filters used in the cohort construction. Bold values correspond to the vocabulary sizes of the medical codes used when training the ZiMM ED architecture.

	Medications	Medical procedures	Diagnoses	Total
#Unique	12 785	9 578	5 885	28 248
#Unique before T^i	10 664	7 460	4 563	22 687
#Unique seen $\geqslant 50$ times before T^i	**6 713**	**1 155**	**1 403**	**9 271**

bution of n_i, namely the number of patients with a given total number of drugs prescriptions (related to urinary problems) during the follow-up 18 months period. Namely, the y-axis of Figure 2.5 (left-hand side) displays $\#\{i : n_i = b\}$ where $b = 1, \ldots, B$ is given by the x-axis. We observe a sharp decrease for $b = 1, \ldots, 5$ from the mode $b = 1$, which corresponds to patients who continue to buy their drugs and just keep doing it (for several months) after their surgery. However, we observe a second mode around $b = 15$ which certainly corresponds to the relapse we are interested in. This plot shows that the considered problem is in a "weak signal" setting, since the second mode is very small compared to the first (at $b = 1$), and that the prediction of this relapse requires a dedicated methodology indeed. This observation is corroborated by the right-hand side of Figure 2.5, which displays the number of patients having at least one drug prescription (related to urinary problems) on a specified bucket, namely the y-axis displays $\#\{i,\ y_{i,b} \geq 1\}$ as a function of b given by the x-axis. Once again, the function is decreasing from the mode $b = 1$ and plateaus between $b = 7$ and $b = 18$, because of patients who continue to purchase the drug after the surgery. We do not observe a second mode as we did in the left-hand side, since the relapse we are interested in is blurry: the drug purchases that define this blurry relapse are heterogeneously distributed among patients and end up flattened out when aggregated in the display of the right-hand side. This motivates, once again, the dedicated methodology proposed by the ZiMM model.

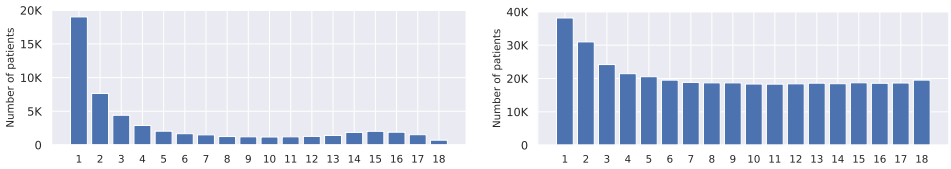

Figure 2.5 – *Left-hand side.* Number of patients as a function of the overall number of drugs prescriptions related to urinary problems during the follow-up 18 months period. The y-axis displays $\#\{i : n_i = b\}$ where $b = 1, \ldots, B$ is given by the x-axis. We observe a sharp decrease from the mode $b = 1$ followed by a second weak mode around $b = 15$. The number of patients with $n_i = 0$ is equal to 84 328 and is not displayed for readability of the plot. *Right-hand side.* Number of patients having at least one drug prescription (related to urinary problems) in each time bucket: the y-axis displays $\#\{i,\ y_{i,b} \geq 1\}$ as a function of b given by the x-axis. We observe a sharp decrease from the mode $b = 1$ and a plateau between $b = 7$ and $b = 18$, since patients often continue to purchase the drug after the surgery.

In the left-hand side of Figure 2.6, we show the distribution of the observation period of each patient before T^i. We observe that it is highly heterogeneous: the maximum is reached at 1 274 days while the average is 486 days, and some patients have a very short observation period. This variability is related to the variability of

the medical practice itself: some patients are treated for urological problems with drugs for a long time before the surgery, while other patients have surgery sooner. On the right-hand side of Figure 2.6, we show the distribution of the durations of all hospitalizations before T^i. We observe that most hospitalizations are one-day short, and that the distribution is heavy-tailed, since these hospitalizations can be related to a large set of possible health problems, leading to very heterogeneous durations. Let us stress that we are considering *all* hospitalizations that occur before T^i, and not only the ones related to a surgery or a urological problem, since all the health events of a patient i are kept before T^i.

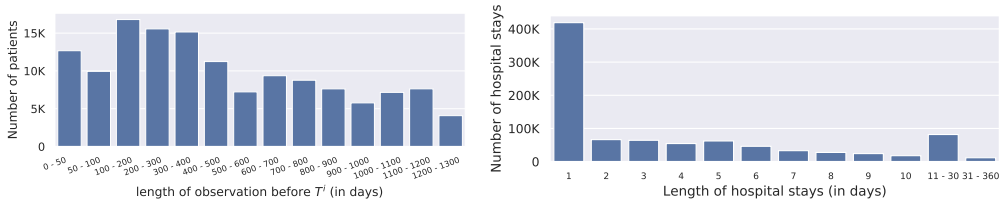

Figure 2.6 – *Left-hand side.* Distribution of the observation period of each patient before T^i. The strong variability of the observation period displayed here is related to the variability of the medical practice itself: some patients are treated for urological problems with drugs for a long time before the surgery, while other patients have surgery sooner. *Right-hand side.* Distribution of the durations of all hospitalizations before T^i. Most hospitalizations are one-day short, and the distribution is heavy-tailed, since these hospitalizations can be related to a large set of possible health problems.

In the left-hand side of Figure 2.7, we display the distribution of the number of unique events per patient. We observe that the bulk of the distribution is between 5 and 200 unique events, with an average of 10 unique events. The right-hand side of Figure 2.7 shows the number of days with at least one medical code in the history of patients before T^i. We observe that most patients have more than 10 days in their history with medical codes. Both displays from Figure 2.7 show that the health pathways of most patients before T^i contain enough variability when assessed by the number of distinct codes, and the number of distinct days with at least one medical code, corresponding to a sufficient amount of variability to carry information for the prediction of the blurry relapse after T^i.

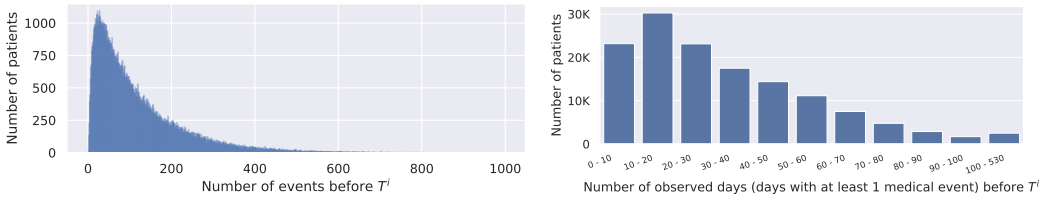

Figure 2.7 – *Left-hand side.* Distribution of the number of events per patient before T^i. The bulk of the distribution is between 5 and 200 unique events, with an average of 117 events per patient. *Right-hand side.* Distribution of the number of days with at least one medical code in the history of patients before T^i. We observe that most patients have more than 10 days in their history with medical codes. Both figures show that most patients have a significant amount of medical code variability and time variability before T^i.

2.3.5 Implementation and evaluation metrics

Our models are implemented with TensorFlow 2 [Abadi et al., 2015]. The ZiMM ED architecture is open-source in GitHub[4]. All the models are trained on a machine equipped with 4 GeForce GTX 1080Ti GPUs and another machine with 3 Tesla V100 GPUs. All the models are trained with the Nadam optimizer [Dozat, 2016; Kingma and Ba, 2015] with learning rate 0.001. The hyper-parameters of all the models are selected through an extensive random grid search, whereas in order to reduce computation time, some hyper-parameters are fixed "by hand". An ablation study showing the sensitivity to hyper-parameters is provided in Section 2.3.7 below.

Metrics: mean-AP, AUC-PR and AUC-ROC. In order to evaluate the quality of the prediction of $y_i = [y_{i,1}, \ldots, y_{i,B}]$ (when using ZIMM ED architecture or any other benchmark models) we use *mean-AP*, defined as the average of the area under the precision-recall curve (AUC-PR), namely we compute the average over the buckets $b = 1, \ldots, B$ of the AUC-PR for each bucket b, i.e., the AUC-PR of the prediction of $y_{i,b}$. Moreover, we report also the AUC-PR and the AUC-ROC (area under the ROC curve) for the binary classification problem $n_i > 0$ against $n_i = 0$.

Train, validation and test sets. All our experiments use the same random data splitting into 70% of patients for training, 15% of patients for validation and 15% for testing. We checked the stationarity across the three splits of labels distribution and the main drugs, diagnoses and medical procedures. We report performances on both the validation and test sets.

2.3.6 Baselines

The prediction performances of ZIMM-ED is compared with several baselines, involving several featuring strategies and different predictive models, both for binary prediction ($n_i > 0$ versus $n_i = 0$) using the AUC-PR and AUC-ROC metrics and for the prediction of $y_i = [y_{i,1}, \ldots, y_{i,B}]$ (the mean-AP score defined in Section 2.3.5). Results are reported in Table 2.2 below, where we observe that ZIMM ED improves all the considered baselines, in particular for the prediction of y_i, since the ZiMM model is dedicated to this task. We consider the following baseline models, more details are provided below:

- LRl2: Logistic regression with ℓ_2 penalization using the scikit-learn library [Pedregosa et al., 2011];
- GBDT: Gradient boosting using XGBoost [Chen et al., 2015];
- MLP: Multilayer Perceptron model with 128 hidden units;
- Word2vec: we first train embeddings of all medical codes following [Mikolov et al., 2013], and use these pre-trained embeddings in a MLP with 128 units;
- LSTM: a single embedding layer and one forward LSTM layer with 128 hidden units;
- Patient2Vec [Zhang et al., 2018].

We evaluated them using 3 types of input features described below.

[4] https://github.com/stephanegaiffas/zimm.git

Static features (SF). This featuring is used for LRl2, GBDT and MLP models. Inputs correspond to aggregated counts of grouped medical codes over the entire observation period of a patient. Number of occurrences of each code is then multiplied by the corresponding one-hot encoding. Hence, the input is a N-dimensional vector representing a patient's medical history. One logistic regression (LRl2) and one gradient boosting decision tree (GBDT) is trained for each bucket b. This input is also used for the multi-layer perceptron (MLP).

Dynamic features (DF). This featuring is used for LRl2, GBDT and MLP models. We split the sequence into subsequences of 60 days, on which we compute the same features as with SF, but within each interval, in order to incorporate longitudinal information into LRl2, GBDT and MLP.

Irregularly-spaced sequence (ISS). This featuring is used for Word2vec, LSTM and Patient2Vec models. In this case, we consider the original patient sequence. For both the Word2vec model and the LSTM one, the events are just gathered in a sequence in which the time-stamps or duration of the events are not used. In the Word2vec case, the codes embeddings are trained using Word2vec [Mikolov et al., 2013] and prediction is performed with a MLP with 128 hidden nodes. In the LSTM case, an embedding layer and one forward LSTM layer with 128 hidden units are trained in an end-to-end fashion and is applied to the sequence. The Patient2vec uses the same input features as the ZIMM ED architecture (i.e., sequence, including time-stamps and duration of events). It has been only used for the binary classification problem (i.e., $n_i > 0$ versus $n_i = 0$).

The predictive performance of all benchmarks and of ZiMM ED are presented in Table 2.2, where we report mean-AP scores on the test set, as well as AUC-PR and AUC-ROC scores for the binary classification problem (see Section 2.3.5 for definition of these scores). According to this table, GBDT-based models on dynamic features (GBDT-DF) performs the best among all benchmark models, however the ZIMM ED architecture outperforms all the benchmark models both for the multi-output y_i prediction and for the binary prediction.

Table 2.2 – Predictive performances (on test data) of benchmark models and ZiMM ED architecture. ZIMM ED appears to perform the best among all models both for multi-output and binary prediction.

Model	mean-AP	AUC-ROC	AUC-PR
LRl2-SF	0.19	0.64	0.50
GBDT-SF	0.24	0.67	0.56
MLP-SF	0.18	0.64	0.49
LRl2-DF	0.21	0.65	0.53
GBDT-DF	0.25	0.68	0.57
MLP-DF	0.19	0.65	0.50
Word2vec-ISS	0.20	0.65	0.53
LSTM-ISS	0.21	0.67	0.54
Patient2Vec	-	0.68	0.55
ZiMM ED (best model)	**0.306**	**0.701**	**0.619**

2.3.7 Ablation study: model performances and variations

The performance reported in Table 2.2 for the ZiMM ED architecture relies on a careful tuning of several hyper-parameters. This corresponds to the so-called *ZiMM ED default* architecture, where the hyperparameters used are described in Table 2.3 below.

	Hyper-parameter	Value
Data preprocessing	Maximum #days observed in patients' sequence before T^i	50
	Maximum #medical events within the same day	24
	Vocabulary size for medications coded with ATC	1036
	Vocabulary size for diagnoses coded with ICD-10	1391
	Vocabulary size for medical procedures coded with CCAM	1146
Embedding of medical codes and time	Embedding dimension of medical codes	64
	L2 penalization rate for medical codes embedding	0.005
	Gaussian dropout rate	0.3
	Embedding dimension for time horizon and hospitalization duration	4
	L2 penalization rate for time embedding	0.01
	Batch normalization epsilon	1e-06
Embeddings aggregation	Aggregation mode	self-attention
	Number of heads	3
	Weight drop-connect rate	0.3
	Dropout rate	0.2
	L2 penalization rate	0.01
ZiMM Encoder	Recurrent layer type	LSTM
	#hidden units per layer	64
	#layers	1
	Dropout rate	0.3
	Recurrent dropout rate	0.2
ZiMM Decoder	Recurrent layer type	GRU
	#hidden units per layer	32
	#RNN layers used in parallel for each value $n_i = b$	1
	#common RNN layers	1
	Gaussian dropout rate	0.3
	Recurrent dropout rate	0.2
Training	Optimizer type	Nadam
	Learning rate	0.001
	Batch size	256

Table 2.3 – ZiMM-ED default parameters.

The hyper-parameters described in Table 2.3 have been selected using an extensive random search for the best mean-AP metric on the validation set. In Figure 2.8 below, we illustrate the value of this metric (y-axis) for some models that were evaluated during the random search. Each point corresponds to a single model with fixed hyper-parameters, and the set of models is the same on all four displays of Figure 2.8. In each of these displays we "align" all the models that share a specific hyper-parameter, namely, respectively from left to right: the number of heads used in the self-attention layer, the number of hidden units used in the recurrent layer of the ZiMM Encoder, the number of stacked recurrent layers in ZiMM Encoder and finally the number of hidden units used in the ZiMM Decoder. The red point on all four Figures corresponds to the best model overall, that leads to the ZiMM ED default architecture described in Table 2.3. Finally, note that the random search

included many other hyper-parameters, such as several learning-rate scheduling strategies, dropout rates, types of dropout regularization including input, output, embedding, cell-state and multiplicative Gaussian, that we do not report for the sake of conciseness.

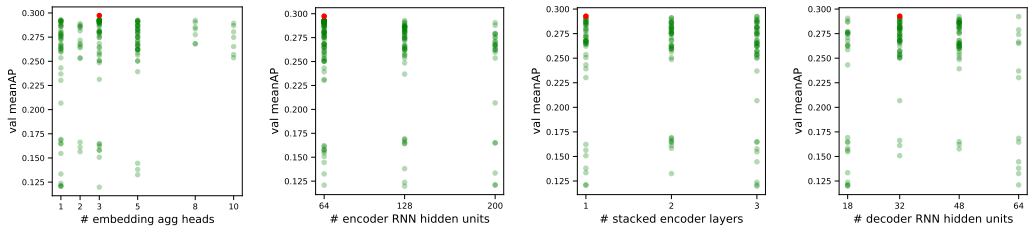

Figure 2.8 – Validation mean-AP (y-axis) of some models evaluated during random search, where each point corresponds to a single model. The set of models is the same on all four displays and the best one (ZiMM ED default) is indicated by a red point. Each display aligns all the models that share a specific hyper-parameter, being, respectively from left to right, the number of heads used in the self-attention layer, the number of hidden units used in the recurrent layer of the ZiMM Encoder, the number of stacked recurrent layers in ZiMM Encoder and finally the number of hidden units used in the ZiMM Decoder.

The remaining of this section proposes an ablation study: we modify some hyperparameters or change some components of ZiMM ED default, and report the impacts on performances in Table 2.4 below, where the first line reports the performances of ZiMM ED default. A discussion around these ablations is given below.

Data preprocessing. The first part (PP) of Table 2.4 shows results obtained with variations of the maximum sequence lengths and the classification system used for drugs encoding. The ZiMM ED default uses ATC encoding instead of the raw CIP-13 encoding, which allows to reduce the vocabulary size from 10 664 for CIP-13 to 1 105 for ATC. As observed in Table 2.4, using CIP-13 (which is a much larger vocabulary for drugs) instead of ATC actually hurts strongly the performances. We observe also that considering longer sequences (100 days) instead of what we do in ZiMM ED default (50 days) deteriorates the performances as well.

Embedding of medical codes and time. Variations around the way embeddings are produced for medical codes and time are reported in the second part (Embedding) of Table 2.4. We observe that increasing the embedding dimensions does not offer performance increase (using $E_{\text{dim}} = 128$ instead of the default $E_{\text{dim}} = 64$), as well as using the same embedding space for all medical codes. In the ZiMM ED default model, tokenized durations and distance to T^i (see Section 2.2.3 above) are embedded with trainable embedding vectors. We observe in Table 2.4 that without these time embeddings, the model performance deteriorates. Also, we find that using trigonometric functions for "positional encoding" as proposed in Vaswani et al. [2017] instead of learned embeddings does not help in our setting.

Embeddings aggregation. Results obtained through different embeddings aggregation techniques are presented in the third part (Aggregation) of Table 2.4. We applied penalization and dropout on attention weights as proposed in [Lin et al., 2019], as well as DropConnect [Wan et al., 2013] for regularizing weights in large

fully-connected layers. As explained in Vaswani et al. [2017], multi-head attention allows the model to jointly attend information from different representation subspaces at different positions, however in our setting, increasing N_{heads} (number of heads) in the self-attention layer does not clearly improve the performance. We tested $N_{\text{heads}} = 8$, as reported in Luo et al. [2019], but the best performance in our case was achieved with $N_{\text{heads}} = 3$.

ZiMM Encoder. We report in the fourth part (Encoder) of Table 2.4 the results of extensive experiments performed in order to identify the best combination of hyperparameters for both recurrent and convolutional layers (CNN): number of hidden units, types of recurrent layers and different dropout rates used in the recurrent layers. Only the best results for each type of layer are reported in Table 2.4. We observe that GRU layers perform generally as well as LSTM layers. Increasing the number of hidden units in recurrent layers to $N_{\text{units}} = 128$ instead of the default choice $N_{\text{units}} = 64$ does not significantly improve performance, as well as using two stacked LSTM layers $N_{\text{layers}} = 2$ instead of the default $N_{\text{layers}} = 1$. We observed also that in our setting, CNN layers badly under-perform compared to the other types of layers. Finally, we tested "Multi-head transformer" which corresponds to the transformer encoder [Vaswani et al., 2017] which takes as input all patient sequence of events. Clinical codes are embedded in the same way as for ZiMM model, which is then passed to the encoder block with a 4-head self-attention mechanism through the entire patient sequence.

Table 2.4 – Performances of some variations around the ZiMM ED default architecture, for which all hyperparameters are given in Table 2.3 above. PP* stands for "preprocessing".

	Architecture modification	# params $\times 10^5$	val set mean-AP	test set mean-AP	test set AUC-ROC	test set AUC-PR
	ZiMM ED default	3.5	0.292	0.304	0.704	0.619
PP*	CIP-13 drugs encoding	7.1	0.279	0.286	0.690	0.604
	100-days sequence	3.5	0.291	0.300	0.701	0.616
Embedding	Common embedding space	13.9	0.292	0.298	0.698	0.613
	Without Δt embedding	3.5	0.293	0.300	0.702	0.617
	Positional encoding	3.5	0.292	0.302	0.702	0.620
	$E_{\text{dim}} = 128$	7.4	0.291	0.300	0.701	0.615
Aggregation	SA $N_{\text{heads}} = 1$	3.2	0.293	0.303	0.701	0.619
	SA $N_{\text{heads}} = 5$	3.7	0.290	0.300	0.704	0.620
	SA $N_{\text{heads}} = 8$	4.1	0.292	0.300	0.701	0.616
	mean	3.1	0.287	0.297	0.696	0.612
Encoder	$N_{\text{units}} = 128$	4.6	0.293	0.301	0.701	0.618
	$N_{\text{layers}} = 2$	3.8	0.290	0.306	0.701	0.619
	bi-LSTM	4.2	0.290	0.300	0.700	0.617
	GRU	3.3	0.294	0.300	0.702	0.614
	Conv1D	3.2	0.231	0.234	0.649	0.536
	Multi-head transformer	6.2	0.279	0.287	0.694	0.607
Decoder	FC layer	3.3	0.290	0.300	0.691	0.617
	LSTM	3.5	0.291	0.301	0.703	0.619
	basic RNN	3.4	0.292	0.301	0.701	0.619
	$N_{\text{units}} = 18$	3.4	0.292	0.298	0.702	0.619
	$N_{\text{layers}} = 2$	3.5	0.288	0.297	0.699	0.615

ZiMM Decoder. Finally, we consider some variations around the ZiMM Decoder, and report the results in the fifth part (Decoder) of Table 2.4. While the ZiMM Decoder described in Section 2.2.4 uses a single feed-forward layer to predict the mixture probabilities, a GRU layer to learn hidden states and GRU layers to predict the parameters of each multinomial distributions, one could use instead fully-connected layers to predict all the parameters of the ZiMM distribution, or alternatively simple RNN layers or LSTM layers. The fully-connected layers deteriorates the most the performance, while replacing the GRU layers by RNN or LSTM layers only deteriorates it mildly. We also report the performance obtained with two stacked GRU $N_{\text{layers}} = 2$ instead of the default one $N_{\text{layers}} = 1$ to produce the hidden states (see Equation (2.3)) and with a smaller hidden size $N_{\text{units}} = 18$ instead of $N_{\text{units}} = 32$.

2.3.8 Visualization of the embeddings produced for diagnoses and drugs codes

We explore the embeddings produced for diagnoses and drugs as a by-product of the ZiMM ED architecture (see Section 2.2.3 about the embeddings of medical codes). We use UMAP [McInnes et al., 2018] in order to reduce the dimension from 64 to 2. The resulted projections are shown in Figure 2.9. The UMAP algorithm requires four hyper-parameters: the number of neighbors to consider when approximating the local metric (n-neighbors), the desired separation between close points in the embedding space (min-dist), the number of training epochs (n-epochs) and finally the projection dimension (d). We fixed $d = 2$ and n-epochs $= 500$. ICD10 disease codes are mapped in PheWAS phenotypes [Diogo et al., 2018] and drugs codes are mapped to the first level of the ATC classification (main anatomical group consisting of a single letter), leading to the color scheme used in Figure 2.9.

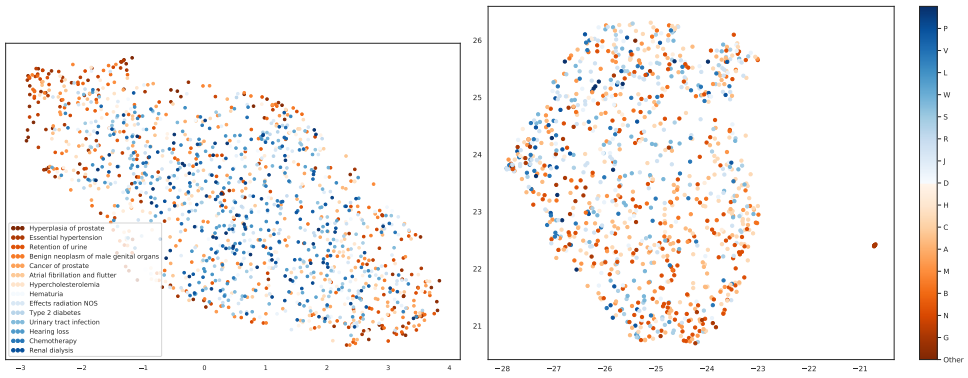

Figure 2.9 – UMAP projections of the embeddings of medical codes. *Left-hand side.* Each point is the projection of the embedding vector of an ICD code, colored with its corresponding PheWAS phenotypes. Note that many points share the same color since many ICD10 codes have the same PheWAS. The legend provides only the most common diagnoses. *Right-hand side.* UMAP projections of the drugs embeddings, colored by the first level of the ATC classification (one letter).

On the left-hand side of Figure 2.9, we can observe the UMAP projections of the embedding vectors of each ICD10 code, colored by the corresponding PheWAS phenotype. The hyper-parameters used here for UMAP are n-neighbors=20 and min-dist=0.1. Many points share the same color since many ICD10 codes have the same PheWAS. We observe here that diagnoses that are known to co-occur

and belong to the same clinical groups are projected close to each other. The red-orange-highlighted points are the phenotypes the most related to the urogenital system such as *Hyperplasia of prostate*, *Retention of urine*, *Benign neoplasm of male genital organs*, and *Cancer of prostate*. We can also observe a visually strong association of these phenotypes with *Essential hypertension*. This is confirmed by Michel et al. [2004], where the association of hypertension and symptoms of benign prostatic hyperplasia is well-studied and understood. Other clinical concepts that are projected close to each other are *Hearing loss* and *Chemotherapy*, *Urinary tract infection* and *Hematuria* (blue-highlighted points), which is also confirmed by Lin et al. [2015]; Taborelli et al. [2019], where it is shown that renal dialysis is associated with a higher risk of cancer.

On the right-hand side of Figure 2.9, we observe the UMAP projections of drugs embeddings, colored by the first level of the ATC classification (first letter). The UMAP hyper-parameters used here are n-neighbors=50 and min-dist=0.2. The dark blue points correspond to *Various* ATC class (V), that mainly correspond to contrast agents for MRI. We observe that these points do not form any cluster but are dispersed over the whole space. This can be explained by the fact that the considered cohort contain only men with BPH (see Section 2.3.4), that often have an MRI of the prostate. Furthermore, we observe that the drugs from the class *Genito-urinary system and sex hormones* (G), that include drugs for urination problems, form one cluster with other age-related medications from the class *Nervous system* (N), namely analgesics, anti-thrombotics, psycholeptics, psychotropic, and anti-parkinsonian drugs.

2.4 Conclusion and future works

In this work, we propose ZiMM Encoder Decoder (ZiMM ED), an end-to-end deep learning model trained against the negative log-likelihood of the new ZiMM (Zero-Inflated Mixture of Multinomial) model, for the modeling of long-term and blurry relapses. This deep learning model is trained on a cohort based on a large electronic health record database, that contains only claims and no clinical data, from the whole French population. We show that it improves the performances of a large number of baselines, including the state-of-the-art, for the considered predictive task. ZiMM ED allows to represent the full health pathways of patients, using all available information, with minimal preprocessing. Therefore, this end-to-end architecture can be used for other tasks as well, through transfer learning or fine tuning of the model. Future works will consider a multi-task version of ZiMM ED (several types of relapses, or other types of events), and other improvements using alternative architectures for attention modeling [Dai et al., 2019; Kitaev et al., 2020; Lan et al., 2020; Wu et al., 2019].

This work addresses the problem of predicting the blurry relapses of the TURP surgery, which is the first step towards an evidence-based approach using machine-learning to help the clinical decision. The next step is to exploit these predictions to help to decide the timing of the surgery: given the current health pathway of the patient, what is his probability of a relapse, so that it can help the clinician to decide when to perform the surgery, or to consider alternative treatments.

Acknowledgments

This research benefited from the National Health Insurance Fund (CNAM) partnership with Data Science initiative at École polytechnique. We thank Maryan Morel, Dinh Phong Nguyen, Youcef Sebiat, Dian Sun, researchers and engineers working on this project. Yiyang Yu was supported by grants from Région Ile-de-France. Anastasiia Kabeshova was supported by CNAM-Polytechnique research partnership. This research is also supported by the Agence Nationale de la Recherche as part of the "Investissements d'avenir" program (reference ANR-19-P3IA-0001; PRAIRIE 3IA Institute). Finally, we wish to thank Pr. Eric Vicaut, for his help in the understanding and the usage of the database.

Bibliography

M. A. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org. 56

A. Atramont, D. Bonnet-Zamponi, I. Bourdel-Marchasson, I. Tangre, A. Fagot-Campagna, and P. Tuppin. Health status and drug use 1 year before and 1 year after skilled nursing home admission during the first quarter of 2013 in France: a study based on the French National Health Insurance Information System. *European journal of clinical pharmacology*, 74(1):109–118, jan 2018. ISSN 1432-1041. doi: 10.1007/s00228-017-2343-y. URL http://www.ncbi.nlm.nih.gov/pubmed/28975381. 50, 51

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer Normalization. *NIPS 2016 Deep Learning Symposium*, jul 2016. URL http://arxiv.org/abs/1607.06450. 47

E. Bacry, S. Gaïffas, F. Leroy, M. Morel, D. P. Nguyen, Y. Sebiat, and D. Sun. SCALPEL3: a scalable open-source library for healthcare claims databases. pages 1–14, 2019. URL http://arxiv.org/abs/1910.07045. 46, 51

J. M. Bajor, D. A. Mesa, T. J. Osterman, and T. A. Lasko. Embedding Complexity In the Data Representation Instead of In the Model: A Case Study Using Heterogeneous Medical Data. feb 2018. URL http://arxiv.org/abs/1802.04233. 42

I. M. Baytas, C. Xiao, X. Zhang, F. Wang, A. K. Jain, and J. Zhou. Patient subtyping via time-aware lstm networks. In *KDD*, 2017. 42

A. L. Beam, B. Kompa, A. Schmaltz, I. Fried, G. Weber, N. P. Palmer, X. Shi, T. Cai, and I. S. Kohane. Clinical Concept Embeddings Learned from Massive Sources of Multimodal Medical Data. apr 2018. URL http://arxiv.org/abs/1804.01486. 42

B. K. Beaulieu-Jones and C. S. Greene. Semi-supervised learning of the electronic health record for phenotype stratification. *Journal of Biomedical Informatics*, 64: 168–178, may 2016. ISSN 15320464. doi: 10.1016/j.jbi.2016.10.007. 42

D. Bender and K. Sartipi. HL7 FHIR: An agile and RESTful approach to healthcare information exchange. In *Proceedings of CBMS 2013 - 26th IEEE International Symposium on Computer-Based Medical Systems*, pages 326–331. IEEE, jun 2013. 42

J. Bezin, M. Duong, R. Lassalle, C. Droz, A. Pariente, P. Blin, and N. Moore. The national healthcare system claims databases in France, SNIIRAM and EGB: Powerful tools for pharmacoepidemiology. *Pharmacoepidemiology and drug safety*, 26(8):954–962, aug 2017. ISSN 1099-1557. doi: 10.1002/pds.4233. URL http://www.ncbi.nlm.nih.gov/pubmed/28544284. 50, 51

T. Chen, T. He, M. Benesty, V. Khotilovich, and Y. Tang. Xgboost: extreme gradient boosting. *R package version 0.4-2*, pages 1–4, 2015. 56

Y. Cheng, F. Wang, P. Zhang, and J. Hu. Risk Prediction with Electronic Health Records: A Deep Learning Approach. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 432–440, Philadelphia, PA, jun 2016. Society for Industrial and Applied Mathematics. ISBN 978-1-61197-434-8. doi: 10.1137/1.9781611974348.49. URL https://epubs.siam.org/doi/10.1137/1.9781611974348.49. 41

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 49

E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun. Doctor AI: Predicting Clinical Events via Recurrent Neural Networks. *JMLR workshop and conference proceedings*, 56:301–318, nov 2016a. ISSN 1938-7288. URL http://arxiv.org/abs/1511.05942http://www.ncbi.nlm.nih.gov/pubmed/28286600{%}0Ahttp://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5341604. 42, 43

E. Choi, M. T. Bahadori, E. Searles, C. Coffey, M. Thompson, J. Bost, J. Tejedor-Sojo, and J. Sun. Multi-layer representation learning for medical concepts. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 13-17-August-2016:1495–1504, feb 2016b. doi: 10.1145/2939672.2939823. URL http://arxiv.org/abs/1602.05568. 42

E. Choi, M. T. Bahadori, J. Sun, J. Kulas, A. Schuetz, W. F. Stewart, J. Sun, J. Kulas, A. Schuetz, W. F. Stewart, and J. Sun. RETAIN: Interpretable Predictive Model in Healthcare using Reverse Time Attention Mechanism. In *Advances in Neural Information Processing Systems*, number Nips, pages 3504–3512, 2016c. 42

E. Choi, M. T. Bahadori, L. Song, W. F. Stewart, and J. Sun. GRAM: Graph-based attention model for healthcare representation learning. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Part F129685:787–795, nov 2017. doi: 10.1145/3097983.3098126. URL http://arxiv.org/abs/1611.07012. 42

E. Choi, Z. Xu, Y. Li, M. W. Dusenberry, G. Flores, Y. Xue, and A. M. Dai. Graph Convolutional Transformer: Learning the Graphical Structure of Electronic Health Records. jun 2019. URL http://arxiv.org/abs/1906.04716. 42, 47

Y. Choi, C. Y.-I. Chiu, and D. Sontag. Learning Low-Dimensional Representations of Medical Concepts. *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science*, 2016:41–50, 2016d. ISSN 2153-4063. URL http://www.ncbi.nlm.nih.gov/pubmed/27570647http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5001761. 42, 47

A. S. Chung and H. H. Woo. Update on minimally invasive surgery and benign prostatic hyperplasia. *Asian Journal of Urology*, 5(1):22–27, jan 2018. ISSN 22143890. doi: 10.1016/j.ajur.2017.06.001. 51

P. Coorevits, M. Sundgren, G. Klein, A. Bahr, B. Claerhout, C. Daniel, M. Dugas, D. Dupont, A. Schmidt, P. Singleton, G. De Moor, and D. Kalra. Electronic health records: New opportunities for clinical research. *Journal of internal medicine*, 274, 08 2013. doi: 10.1111/joim.12119. 51

J. N. Cornu, S. Ahyai, A. Bachmann, J. De La Rosette, P. Gilling, C. Gratzke, K. McVary, G. Novara, H. Woo, and S. Madersbacher. A systematic review and meta-analysis of functional outcomes and complications following transurethral procedures for lower urinary tract symptoms resulting from benign prostatic obstruction: An update. *European Urology*, 67(6):1066–1096, jun 2015. ISSN 18737560. doi: 10.1016/j.eururo.2014.06.017. 51

N. Cornu, M. Drake, M. Gacci, C. Gratzke, T. Herrmann, S. Madersbacher, C. Mamoulakis, and K. Tikkinen. EAU Guidelines: Management of Non-neurogenic Male LUTS | Uroweb. Technical report, 2018. URL https://uroweb.org/guideline/treatment-of-non-neurogenic-male-luts/. 52

P. Dahm, M. Brasure, R. MacDonald, C. M. Olson, V. A. Nelson, H. A. Fink, B. Rwabasonga, M. C. Risk, and T. J. Wilt. Comparative Effectiveness of Newer Medications for Lower Urinary Tract Symptoms Attributed to Benign Prostatic Hyperplasia: A Systematic Review and Meta-analysis, apr 2017. ISSN 18737560. 51

Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. *arXiv:1901.02860 [cs, stat]*, June 2019. URL http://arxiv.org/abs/1901.02860. arXiv: 1901.02860. 62

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *North American Association for Computational Linguistics (NAACL)*, oct 2018. URL http://arxiv.org/abs/1810.04805. 42

D. Diogo, C. Tian, C. S. Franklin, M. Alanne-Kinnunen, M. March, C. C. Spencer, C. Vangjeli, M. E. Weale, H. Mattsson, E. Kilpeläinen, P. M. Sleiman, D. F. Reilly, J. McElwee, J. C. Maranville, A. K. Chatterjee, A. Bhandari, K. D. H.

Nguyen, K. Estrada, M. P. Reeve, J. Hutz, N. Bing, S. John, D. G. MacArthur, V. Salomaa, S. Ripatti, H. Hakonarson, M. J. Daly, A. Palotie, D. A. Hinds, P. Donnelly, C. S. Fox, A. G. Day-Williams, R. M. Plenge, and H. Runz. Phenome-wide association studies across large population cohorts support drug target validation. *Nature Communications*, 9(1), dec 2018. ISSN 20411723. doi: 10.1038/s41467-018-06540-3. 61

T. Dozat. Incorporating Nesterov Momentum into Adam. *ICLR Workshop*, (1): 2013–2016, 2016. 56

L. Fonteneau, N. Le Meur, A. Cohen-Akenine, C. Pessel, C. Brouard, F. Delon, G. Desjeux, J. Durand, J. Kirchgesner, N. Lapidus, M. Lemaitre, S. Tala, A. Thiébaut, L. Watier, J. Rudant, and L. Guillon-Grammatico. The use of administrative health databases in infectious disease epidemiology and public health. *Revue d'epidemiologie et de sante publique*, 65 Suppl 4:S174–S182, oct 2017. ISSN 0398-7620. doi: 10.1016/j.respe.2017.03.131. URL http://www.ncbi.nlm.nih.gov/pubmed/28624133. 50, 51

H. Hashim and P. Abrams. Transurethral resection of the prostate for benign prostatic obstruction: Will it remain the gold standard? *European Urology*, 67 (6):1097–1098, jun 2015. ISSN 18737560. doi: 10.1016/j.eururo.2014.12.022. 51

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 49

H. D. Hub. https://www.health-data-hub.fr, 2019. 51

E. H. Kim, J. A. Larson, and G. L. Andriole. Management of Benign Prostatic Hyperplasia. *Annual review of medicine*, 67:137–51, 2016. ISSN 1545-326X. doi: 10.1146/annurev-med-063014-123902. URL http://www.ncbi.nlm.nih.gov/pubmed/26331999. 51

D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, dec 2015. URL http://arxiv.org/abs/1412.6980. 56

N. Kitaev, L. Kaiser, and A. Levskaya. Reformer: The Efficient Transformer. *arXiv:2001.04451 [cs, stat]*, Jan. 2020. URL http://arxiv.org/abs/2001.04451. arXiv: 2001.04451. 62

Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv:1909.11942 [cs]*, Jan. 2020. URL http://arxiv.org/abs/1909.11942. arXiv: 1909.11942. 62

Y. Li, S. Rao, J. R. A. Solares, A. Hassaine, D. Canoy, Y. Zhu, K. Rahimi, and G. Salimi-Khorshidi. BEHRT: Transformer for Electronic Health Records. jul 2019. URL http://arxiv.org/abs/1907.09538. 42

M. Y. Lin, M. C. Kuo, C. C. Hung, W. J. Wu, L. T. Chen, M. L. Yu, C.-C. Hsu, C.-H. Lee, H.-C. Chen, and S.-J. Hwang. Association of Dialysis with the Risks of Cancers. *PLoS ONE*, 10(4), Apr. 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0122856. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4395337/. 62

Z. Lin, M. Feng, C. N. Dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2019. URL http://arxiv.org/abs/1703.03130. 48, 59

Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzel. Learning to diagnose with LSTM recurrent neural networks. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, nov 2016. URL http://arxiv.org/abs/1511.03677. 42

T. Lourenco, M. Shaw, C. Fraser, G. MacLennan, J. N'Dow, and R. Pickard. The clinical effectiveness of transurethral incision of the prostate: a systematic review of randomised controlled trials. *World journal of urology*, 28(1):23–32, feb 2010. ISSN 1433-8726. doi: 10.1007/s00345-009-0496-8. URL http://www.ncbi.nlm.nih.gov/pubmed/20033744. 52

D. Luo, H. Xu, and L. Carin. Interpretable ICD Code Embeddings with Self- and Mutual-Attention Mechanisms. jun 2019. URL http://arxiv.org/abs/1906.05492. 48, 60

F. Ma, R. Chitta, J. Zhou, Q. You, T. Sun, and J. Gao. Dipole: Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume Part F129685, pages 1903–1911. Association for Computing Machinery, aug 2017. ISBN 9781450348874. doi: 10.1145/3097983.3098088. 42

M. R. Macey and M. C. Raynor. Medical and Surgical Treatment Modalities for Lower Urinary Tract Symptoms in the Male Patient Secondary to Benign Prostatic Hyperplasia: A Review. *Seminars in Interventional Radiology*, 33(3):217–223, sep 2016. ISSN 10988963. doi: 10.1055/s-0036-1586142. 51

S. Madersbacher, J. Lackner, C. Brössner, M. Röhlich, I. Stancik, M. Willinger, G. Schatzl, and Prostate Study Group of the Austrian Society of Urology. Reoperation, myocardial infarction and mortality after transurethral and open prostatectomy: a nation-wide, long-term analysis of 23,123 cases. *European urology*, 47 (4):499–504, apr 2005. ISSN 0302-2838. doi: 10.1016/j.eururo.2004.12.010. URL http://www.ncbi.nlm.nih.gov/pubmed/15774249. 52

L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv:1802.03426 [cs, stat]*, Dec. 2018. URL http://arxiv.org/abs/1802.03426. arXiv: 1802.03426. 61

M. C. Michel, U. Heemann, H. Schumacher, L. Mehlburger, and M. Goepel. Association of hypertension with symptoms of benign prostatic hyperplasia. *The Journal of Urology*, 172(4 Pt 1):1390–1393, Oct. 2004. ISSN 0022-5347. doi: 10.1097/01.ju.0000139995.85780.d8. 62

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. 42, 56, 57

R. Miotto, L. Li, B. A. Kidd, and J. T. Dudley. Deep Patient: An Unsupervised Representation to Predict the Future of Patients from the Electronic Health Records. *Scientific Reports*, 6, may 2016. ISSN 20452322. doi: 10.1038/srep26094. 42

M. Morel, E. Bacry, S. Gaïffas, A. Guilloux, and F. Leroy. ConvSCCS: convolutional self-controlled case series model for lagged adverse event detection. *Biostatistics (Oxford, England)*, mar 2019. ISSN 1468-4357. doi: 10.1093/biostatistics/kxz003. URL http://www.ncbi.nlm.nih.gov/pubmed/30851046. 51

D. Neil, M. Pfeiffer, and S. C. Liu. Phased LSTM: Accelerating recurrent network training for long or event-based sequences. *Advances in Neural Information Processing Systems*, pages 3889–3897, oct 2016. ISSN 10495258. URL http://arxiv.org/abs/1610.09513. 42

P. Nguyen, T. Tran, N. Wickramasinghe, and S. Venkatesh. Deepr: A Convolutional Net for Medical Records. *IEEE Journal of Biomedical and Health Informatics*, 21(1):22–30, jul 2017. ISSN 21682194. doi: 10.1109/JBHI.2016.2633963. URL http://arxiv.org/abs/1607.07519. 42

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011. 56

J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162. 42

T. Pham, T. Tran, D. Phung, and S. Venkatesh. Predicting healthcare trajectories from medical records: A deep learning approach. *Journal of biomedical informatics*, 69:218–229, 2017. ISSN 1532-0480. doi: 10.1016/j.jbi.2017.04.001. URL http://www.ncbi.nlm.nih.gov/pubmed/28410981. 42

A. Rajkomar, E. Oren, K. Chen, A. M. Dai, N. Hajaj, M. Hardt, P. J. Liu, X. Liu, J. Marcus, M. Sun, P. Sundberg, H. Yee, K. Zhang, Y. Zhang, G. Flores, G. E. Duggan, J. Irvine, Q. Le, K. Litsch, A. Mossin, J. Tansuwan, D. Wang, J. Wexler, J. Wilson, D. Ludwig, S. L. Volchenboum, K. Chou, M. Pearson, S. Madabushi, N. H. Shah, A. J. Butte, M. D. Howell, C. Cui, G. S. Corrado, and J. Dean. Scalable and accurate deep learning with electronic health records. *npj Digital Medicine*, 1(1), jan 2018. ISSN 2398-6352. doi: 10.1038/s41746-018-0029-1. URL https://arxiv.org/abs/1801.07860. 41, 42

L.-M. Scailteux, C. Droitcourt, F. Balusson, E. Nowak, S. Kerbrat, A. Dupuy, E. Drezen, A. Happe, and E. Oger. French administrative health care database (SNDS): The value of its enrichment. *Therapie*, 74(2):215–223, apr 2019. ISSN 1958-5578. doi: 10.1016/j.therap.2018.09.072. URL http://www.ncbi.nlm.nih.gov/pubmed/30392702. 50, 51

M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997. 49

B. Shickel, P. J. Tighe, A. Bihorac, and P. Rashidi. Deep EHR: A Survey of Recent Advances in Deep Learning Techniques for Electronic Health Record (EHR) Analysis. *IEEE Journal of Biomedical and Health Informatics*, 22(5):1589–1604, sep 2018. ISSN 21682194. doi: 10.1109/JBHI.2017.2767063. 42

J. Silva, C. M. Silva, and F. Cruz. Current medical treatment of lower urinary tract symptoms/BPH: do we have a standard? *Current opinion in urology*, 24 (1):21–8, jan 2014. ISSN 1473-6586. doi: 10.1097/MOU.0000000000000007. URL http://www.ncbi.nlm.nih.gov/pubmed/24231531. 51

SNDS. Processus d'accès aux données | SNDS, 2019. URL https://www.snds.gouv.fr/SNDS/Processus-d-acces-aux-donnees. 51

M. Taborelli, F. Toffolutti, S. Del Zotto, E. Clagnan, L. Furian, P. Piselli, F. Citterio, L. Zanier, G. Boscutti, D. Serraino, S. Shalaby, R. Petrara, P. Burra, G. Zanus, S. Zanini, P. Rigotti, M. Rendina, A. Di Leo, F. P. Schena, G. Grandaliano, M. Fiorentino, A. Lauro, A. D. Pinna, P. Di Gioia, S. Pellegrini, C. Zanfi, M. P. Scolari, S. Stefoni, P. Todeschini, L. Panicali, C. Valentini, U. Baccarani, A. Risaliti, G. L. Adani, D. Lorenzin, G. M. Ettorre, G. Vennarecci, M. Colasanti, M. Coco, F. Ettorre, R. Santoro, L. Miglioresi, F. Nudo, M. Rossi, G. Mennini, L. Toti, G. Tisone, A. Casella, L. Fazzolari, D. Sforza, G. Iaria, C. Gazia, C. Belardi, C. Cimaglia, A. Agresta, G. D'Offizi, U. V. Comandini, R. Lionetti, M. Montalbano, C. Taibi, G. Fantola, F. Zamboni, G. B. Piredda, M. B. Michittu, M. G. Murgia, B. Onano, L. Fratino, L. D. Maso, P. De Paoli, D. Verdirosi, E. Vaccher, F. Pisani, A. Famulari, F. Delreno, S. Iesari, L. De Luca, M. Iaria, E. Capocasale, E. Cremaschi, S. Sandrini, F. Valerio, V. Mazzucotelli, N. Bossini, G. Setti, M. Veroux, P. Veroux, A. Giaquinta, D. Zerbo, G. Busnach, L. Di Leo, M. L. Perrino, M. Querques, V. Colombo, M. C. Sghirlanzoni, P. Messa, A. Leoni, L. Galatioto, S. Gruttadauria, V. Sparacino, F. Caputo, B. Buscemi, F. Citterio, G. Spagnoletti, M. P. Salerno, E. Favi, G. P. Segoloni, L. Biancone, A. Lavacca, M. C. Maresca, C. Cascone, B. Virgilio, D. Donati, F. Dossi, A. Fontanella, A. Ambrosini, M. Di Cicco, and for the Italian Transplant & Cancer Cohort Study. Increased cancer risk in patients undergoing dialysis: a population-based cohort study in North-Eastern Italy. *BMC Nephrology*, 20(1):107, Mar. 2019. ISSN 1471-2369. doi: 10.1186/s12882-019-1283-4. URL https://doi.org/10.1186/s12882-019-1283-4. 62

P. Tuppin, J. Rudant, P. Constantinou, C. Gastaldi-Ménager, A. Rachas, L. de Roquefeuil, G. Maura, H. Caillol, A. Tajahmady, J. Coste, C. Gissot, A. Weill, and A. Fagot-Campagna. Value of a national administrative database to guide public decisions: From the système national d'information interrégimes de l'Assurance Maladie (SNIIRAM) to the système national des données de santé (SNDS) in France. *Revue d'epidemiologie et de sante publique*, 65 Suppl 4: S149–S167, oct 2017. ISSN 0398-7620. doi: 10.1016/j.respe.2017.05.004. URL http://www.ncbi.nlm.nih.gov/pubmed/28756037. 50

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017-December:5999–6009, 2017. ISSN 10495258. 48, 59, 60

L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In S. Dasgupta and D. McAllester, editors, *Proceedings*

of the 30th International Conference on Machine Learning, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA, Jun 2013. PMLR. URL http://proceedings.mlr.press/v28/wan13.html. 48, 59

J. Wu, C. Xiong, T. Schnabel, Y. Zhang, W. Y. Wang, and P. Bennett. Combiner: Inductively Learning Tree Structured Attention in Transformers. Sept. 2019. URL https://openreview.net/forum?id=B1eySTVtvB. 62

C. Xiao, E. Choi, and J. Sun. Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. *Journal of the American Medical Informatics Association : JAMIA*, 25(10):1419–1428, oct 2018. ISSN 1527-974X. doi: 10.1093/jamia/ocy068. URL http://www.ncbi.nlm.nih.gov/pubmed/29893864http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC6188527. 42

J. Zhang, K. Kowsari, J. H. Harrison, J. M. Lobo, and L. E. Barnes. Patient2Vec: A Personalized Interpretable Deep Representation of the Longitudinal Electronic Health Record. *IEEE Access*, 6:65333–65346, 2018. ISSN 21693536. doi: 10.1109/ACCESS.2018.2875677. 56

Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai. What to do next: Modeling user behaviors by time-lstm. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3602–3608, 2017. doi: 10.24963/ijcai.2017/504. URL https://doi.org/10.24963/ijcai.2017/504. 42

Chapter 3

About contrastive unsupervised representation learning for classification and its convergence [1]

“ 性相近也，习相远也。
By nature, men are nearly alike; by practice, they get to be wide apart. ”

Confucius (translated by James Legge)

Contents

[1] This chapter is based on the joint work with I.Merad, S. Gaiffas, E. Bacry.

Abstract

Contrastive representation learning has been recently proved to be very efficient for self-supervised training. These methods have been successfully used to train encoders which perform comparably to supervised training on downstream classification tasks. A few works have started to build a theoretical framework around contrastive learning in which guarantees for its performance can be proven. We provide extensions of these results to training with multiple negative samples and for multiway classification. Furthermore, we provide convergence guarantees for the minimization of the contrastive training error with gradient descent of an overparametrized deep neural encoder, and provide some numerical experiments that complement our theoretical findings.

3.1 Introduction

The aim of this work is to provide additional theoretical guarantees for *contrastive learning* [van den Oord et al., 2018], which corresponds to methods allowing to learn useful data representations in an *unsupervised* setting. Unsupervised representation learning was initially approached with a fair amount of success by training through the minimization of losses coming from "pretext" tasks, a technique known as *self-supervision* [Doersch and Zisserman, 2017], where labels can be automatically constructed. Notable examples of pretext tasks in computer vision include colorization [Zhang et al., 2016], transformation prediction [Dosovitskiy et al., 2014; Gidaris et al., 2018] or predicting patch relative positions [Doersch et al., 2015]. Some theoretical guarantees [Lee et al., 2020] were recently proposed to support training on pretext tasks.

Contrastive learning is also known to be very effective for pretraining supervised methods [Caron et al., 2020; Chen et al., 2020a,b; Grill et al., 2020], where we can observe that, quite surprisingly, the gap between unsupervised and supervised performance has been closed for tasks such as image classification: the use of a pretrained image encoder on top of simple classification layers, that are trained on a fraction of the labels available, allows to achieve an accuracy comparable to that of a fully supervised end-to-end training [Grill et al., 2020; Hénaff et al., 2019]. Contrastive methods show also strong success in natural language processing [Devlin et al., 2018; Logeswaran and Lee, 2018; Mikolov et al., 2013; van den Oord et al., 2018], video classification [Sun et al., 2019], reinforcement learning [Srinivas et al., 2020] and time-series [Franceschi et al., 2019].

Although the papers cited above introduce methods with considerable variations, they mostly agree on the following basic pretraining approach: provided a dataset, an encoder is trained using a contrastive loss whose minimization allows to learn embeddings that are *similar* for pairs of samples (called the *positives*) that are close to each other (such as pairs of random data augmentations of the same image, see Chen et al. [2020a]; He et al. [2020]), while such embeddings are *contrasted* for dissimilar pairs (called the *negatives*).

However, despite growing efforts [Saunshi et al., 2019; Wang and Isola, 2020], as of today, few theoretical results have been obtained. For instance, there is still no clear theoretical explanation of how a supervised task could benefit from

an upstream unsupervised pretraining phase, or of what could be the theoretical guarantees for the convergence of the minimization procedure of the contrastive loss during this pretraining phase. Getting some answers to these questions would undoubtedly be a step towards a better theoretical understanding of contrastive representation learning.

Our contributions in this paper are twofold. In Section 3.3, we provide new theoretical guarantees for the classification performance of contrastively trained models in the case of multiway classification tasks, using *multiple* negative samples. We extend results from Saunshi et al. [2019] to show that unsupervised training performance reflects on a subsequent classification task in the case of multiple tasks and when a high number of negative samples is used. In Section 3.4, we prove a convergence result for an *explicit* algorithm (gradient descent), when training overparametrized deep neural network for unsupervised contrastive representation learning. We explain how results from Allen-Zhu et al. [2019] about training convergence of overparametrized deep neural networks can be applied to a contrastive learning objective. The results and major assumptions of both Sections 3.3 and 3.4 are illustrated in Section 2.3 through experiments on a few simple datasets.

3.2 Related work

A growing literature attempts to build a theoretical framework around contrastive learning and to provide justifications for its success beyond intuitive ideas. In Saunshi et al. [2019] a formalism is proposed together with results on classification performance based on unsupervisedly learned representation. However, these results do not explain the performance gain that is observed empirically [Chen et al., 2020a; He et al., 2020] when a high number of negative samples are used, while the results proposed in Section 3.3 below hold for an arbitrary large number of negatives (and decoupled from the number of classification tasks). A more recent work [Wang and Isola, 2020] emphasizes the two tendencies encouraged by the contrastive loss: the encoder's outputs are incentivized to spread evenly on the unit hypersphere, and encodings of same-class samples are driven close to each other while those of different classes are driven apart. Interestingly, this work also shows how the trade-off between these two aspects can be controlled, by introducing weight factors in the loss leading to improved performance. Chuang et al. [2020] considers the same setting as Saunshi et al. [2019] and addresses the bias problem that comes from collisions between positive and negative sampling in the unsupervised constrastive loss. They propose to simulate unbiased negative sampling by assuming, among other things, extra access to positive sampling. However, one has to keep in mind that excessive access to positive sampling gets the setting closer to that of supervised learning.

In a direction that is closer to the result proposed in Section 3.4 below, Wen [2020] provides a theoretical guarantee on the training convergence of gradient descent for an overparametrized model that is trained with an unsupervised contrastive loss, using earlier works by Allen-Zhu et al. [2019]. However, two separate encoders are considered instead of a single one: one for the query, which corresponds to a sample from the dataset, and one for the (positive and negative) samples to compare the query to. In this setting, it is rather unclear how the two resulting encoders are to be used for downstream classification. In Section 3.4 below, we explain how the results from Allen-Zhu et al. [2019] can be used for the more realistic setting of a single encoder, by introducing a reasonable assumption on the encoder

outputs.

3.3 Unsupervised training improves supervised performance

In this section, we provide new results in the setting previously considered in Saunshi et al. [2019]. We assume that data are distributed according to a finite set $\mathcal{C}$ of latent classes, and denote $N_{\mathcal{C}} = \mathrm{card}(\mathcal{C})$ its cardinality. Let ρ be a discrete distribution over $\mathcal{C}$ that is such that

$$\sum_{c\in\mathcal{C}} \rho(c) = 1 \quad \text{and} \quad \rho(c) > 0$$

for all $c \in \mathcal{C}$. We denote $\mathcal{D}_c$ a distribution over the feature space $\mathcal{X}$ from a class $c \in \mathcal{C}$. In order to perform unsupervised contrastive training, on the one hand we assume that we can sample *positive* pairs (x, x^+) from the distribution

$$\mathcal{D}_{\mathrm{sim}}(x, x^+) = \sum_{c\in\mathcal{C}} \rho(c)\mathcal{D}_c(x)\mathcal{D}_c(x^+), \tag{3.1}$$

namely, (x, x^+) is sampled as a mixture of independent pairs conditionally to a shared latent class, sampled according to ρ. On the other hand, we assume that we can sample *negative* samples x^- from the distribution

$$\mathcal{D}_{\mathrm{neg}}(x^-) = \sum_{c\in\mathcal{C}} \rho(c)\mathcal{D}_c(x^-). \tag{3.2}$$

Given $k \leqslant N_{\mathcal{C}} - 1$, a $(k+1)$-way classification task is a subset $\mathcal{T} \subseteq \mathcal{C}$ of cardinality $|\mathcal{T}| = k + 1$, which induces the conditional distribution

$$\mathcal{D}_{\mathcal{T}}(c) = \rho(c \mid c \in \mathcal{T})$$

for $c \in \mathcal{C}$ and we define

$$\mathcal{D}_{\mathcal{T}}(x, c) = \mathcal{D}_{\mathcal{T}}(c)\mathcal{D}_c(x).$$

In particular, we denote as $\mathcal{C}$, whenever there is no ambiguity, the $N_{\mathcal{C}}$-way classification task where the labels are sampled from ρ, namely $\mathcal{D}_{\mathcal{C}}(x, c) = \rho(c)\mathcal{D}_c(x)$.

Supervised loss, mean classifier

For an encoder function $f : \mathcal{X} \to \mathbb{R}^d$, we define a supervised loss (cross-entropy with the best possible linear classifier on top of the representation) over task $\mathcal{T}$ as

$$L_{\mathrm{sup}}(f, \mathcal{T}) = \inf_{W\in\mathbb{R}^{|\mathcal{T}|\times d}} \mathbb{E}_{(x,c)\sim\mathcal{D}_{\mathcal{T}}} \left[-\log \left(\frac{\exp\left(Wf(x)\right)_c}{\sum_{c'\in\mathcal{T}} \exp\left(Wf(x)\right)_{c'}} \right) \right]. \tag{3.3}$$

Then, it is natural to consider the *mean* or *discriminant* classifier with weights W^μ which stacks, for $c \in \mathcal{T}$, the vectors

$$W^\mu_{c,:} = \mathbb{E}_{x\sim\mathcal{D}_c} \left[f(x) \right] \tag{3.4}$$

and whose corresponding (supervised) loss is given by

$$L^\mu_{\mathrm{sup}}(f, \mathcal{T}) = \mathbb{E}_{(x,c)\sim\mathcal{D}_{\mathcal{T}}} \left[-\log \left(\frac{\exp\left(W^\mu f(x)\right)_c}{\sum_{c'\in\mathcal{T}} \exp\left(W^\mu f(x)\right)_{c'}} \right) \right]. \tag{3.5}$$

Note that, obviously, one has $L_{\mathrm{sup}}(f, \mathcal{T}) \leqslant L^\mu_{\mathrm{sup}}(f, \mathcal{T})$.

Unsupervised contrastive loss

We consider the unsupervised contrastive loss *with N negative samples* given by

$$L_{\text{un}}^{N}(f) = \mathbb{E}_{\substack{(x,x^+)\sim\mathcal{D}_{\text{sim}} \\ X^-\sim\mathcal{D}_{\text{neg}}^{\otimes N}}}\left[-\log\left(\frac{\exp\left(f(x)^\top f(x^+)\right)}{\exp\left(f(x)^\top f(x^+)\right)+\sum_{x^-\in X^-}\exp\left(f(x)^\top f(x^-)\right)}\right)\right], \tag{3.6}$$

where $\mathcal{D}_{\text{sim}}$ is given by Equation (3.1) and where $\mathcal{D}_{\text{neg}}^{\otimes N}$ stands for the N tensor product of the $\mathcal{D}_{\text{neg}}$ distribution given by Equation (3.2). When a single negative sample is used ($N = 1$), we will use the notation $L_{\text{un}}(f) = L_{\text{un}}^{1}(f)$. In the rest of the chapter, N will stand for the number of negatives used in the unsupervised loss (3.6).

3.3.1 Inequalities for unsupervised training with multiple classes

The following Lemma states that the unsupervised objective with a single negative sample can be related to the supervised loss for which the target task is classification over the whole set of latent classes $\mathcal{C}$.

Lemma 3.1. *For any encoder $f : \mathcal{X} \to \mathbb{R}^d$, one has*

$$L_{\text{sup}}(f,\mathcal{C}) \leqslant L_{\text{sup}}^{\mu}(f,\mathcal{C}) \leqslant \frac{1}{p_{\text{min}}^{\rho}} L_{\text{un}}(f) + \log N_{\mathcal{C}}, \tag{3.7}$$

where $p_{\text{min}}^{\rho} = \min_c \rho(c)$.

The proof of Lemma 3.1 is given in the appendix, and uses a trick from Lemma 4.3 in Saunshi et al. [2019] relying on Jensen's inequality. This Lemma relates the unsupervised and the supervised losses, a shortcoming being the introduction of p_{min}^{ρ}, which is small for a large $N_{\mathcal{C}}$ since obviously $p_{\text{min}}^{\rho} \leqslant 1/N_{\mathcal{C}}$.

The analysis becomes more difficult with a larger number of negative samples. Indeed, in this case, one needs to carefully keep track of how many distinct classes will be represented by each draw. This is handled by Theorem B.1 of Saunshi et al. [2019], but the bound given therein only estimates an expectation of the supervised loss w.r.t. the random subset of classes considered (so called tasks). For multiple negative samples, the approach adopted in the proof of Lemma 3.1 above further degrades, since p_{min}^{ρ} would be replaced by the minimum probability among tuple draws, an even much smaller quantity.

We propose the following Lemma, which assumes that the number of negative samples is large enough compared to the number of latent classes.

Lemma 3.2. *Consider the unsupervised objective with N negative samples as defined in Equation (3.6) and assume that N satisfies $N = \Omega(N_{\mathcal{C}} \log N_{\mathcal{C}})$. Then, we have*

$$L_{\text{sup}}(f,\mathcal{C}) \leqslant L_{\text{sup}}^{\mu}(f,\mathcal{C}) \leqslant \frac{1}{p_{cc}^{\rho}(N)} L_{\text{un}}^{N}(f), \tag{3.8}$$

where $p_{cc}^{\rho}(N)$ is the probability to have all coupons after N draws in an $N_{\mathcal{C}}$-coupon collector problem with draws from ρ.

The proof of Lemma 3.2 is given in the appendix. In this result, $p_{cc}^{\rho}(N)$ is related to the following coupon collector problem. Assume that ρ is the uniform distribution over $\mathcal{C}$ and let T be the random number of necessary draws until each $c \in \mathcal{C}$ is drawn at least once. It is known (see for instance Motwani and Raghavan

[1995]) that the expectation and variance of T are respectively given by $N_{\mathcal{C}} H_{N_{\mathcal{C}}}$ and $(N_{\mathcal{C}}\pi)^2/6$, where H_n is the n-th harmonic number $H_n = \sum_{i=1}^{n} 1/i$. This entails using Chebyshev's inequality that

$$\mathbb{P}\left(|T - N_{\mathcal{C}} H_{N_{\mathcal{C}}}| \geqslant \beta N_{\mathcal{C}}\right) \leqslant \frac{\pi^2}{6\beta^2}$$

for any $\beta > 0$, so that whenever ρ is sufficiently close to a uniform distribution and $N = \Omega(N_{\mathcal{C}} \log N_{\mathcal{C}})$, the probability p^{ρ}_{cc} is reasonably high. Due to the randomness of the classes sampled during training, it is difficult to obtain a better inequality than Lemma 3.2 if we want to upper bound $L^N_{\text{un}}(f)$ by the supervised $L_{\text{sup}}(f, \mathcal{C})$ on all classes. However, the result can be improved by considering the average loss over tasks $L_{\text{sup},k}(f)$, as explained in the next Section.

3.3.2 Guarantees on the average supervised loss

In this Section, we bound the average of the supervised classification loss on tasks that are subsets of $\mathcal{C}$. Towards this end, we need to assume (only in this Section) that ρ is uniform. We consider supervised tasks consisting in distinguishing one latent class from k other classes, given that they are distinct and uniformly sampled from $\mathcal{C}$. We define the average supervised loss of f for $(k+1)$-way classification as

$$L_{\text{sup},k}(f) = \mathbb{E}_{\mathcal{T}\sim\mathcal{D}^{k+1}}\left[L_{\text{sup}}\left(f, \mathcal{T}\right)\right], \tag{3.9}$$

where $\mathcal{D}^{k+1}$ is the uniform distribution over $(k+1)$-way tasks, which means uniform sampling of $\{c_1, \cdots, c_{k+1}\}$ *distinct* classes in $\mathcal{C}$. We define also the average supervised loss of the mean classifier

$$L^{\mu}_{\text{sup},k}(f) = \mathbb{E}_{\mathcal{T}\sim\mathcal{D}^{k+1}}\left[L^{\mu}_{\text{sup}}\left(f, \mathcal{T}\right)\right], \tag{3.10}$$

where we recall that $L^{\mu}_{\text{sup}}\left(f, \mathcal{T}\right)$ is given by (3.5). The next Proposition is a generalization to arbitrary values of k and N of Lemma 4.3 from Saunshi et al. [2019], where it is assumed $k = 1$ and $N = 1$.

Proposition 3.3. *Consider the unsupervised loss $L^N_{\text{un}}(f)$ from Equation (3.6) with N negative samples. Assume that ρ is uniform over $\mathcal{C}$ and that $2 \leqslant k+1 \leqslant N_{\mathcal{C}}$. Then, any encoder function $f : \mathcal{X} \to \mathbb{R}^d$ satisfies*

$$L_{\text{sup},k}(f) \leqslant L^{\mu}_{\text{sup},k}(f) \leqslant \frac{k}{1-\tau^+_N}\left(L^N_{\text{un}}(f) - \tau^+_N \log(N+1)\right)$$

with $\tau^+_N = \mathbb{P}\left[c_i = c, \forall i \mid (c, c_1, \cdots, c_N) \sim \rho^{\otimes N+1}\right]$.

The proof of Proposition 3.3 is given in the appendix. This Proposition states that, in a setting similar to that of Saunshi et al. [2019], on average, the $(k+1)$-way supervised classification loss is upper-bounded by the unsupervised loss (both with $N = 1$ negative or $N > 1$ negatives), that contrastive learning algorithms actually minimize. Therefore, these results give hints for the performances of the learned representation for downstream tasks.

Also, while Saunshi et al. [2019] only considers an unsupervised loss with $N = k$ negatives along with $(k+1)$-way tasks for evaluation, the quantities N and k are decoupled in Proposition 3.3. Furthermore, whenever ρ is uniform, one has $\tau^+_N = \sum_{c\in\mathcal{C}} \rho(c)^{N+1} = N_{\mathcal{C}}^{-N}$, which decreases to 0 as $N \to +\infty$, so that a larger

number of negatives N makes $k/(1-\tau_N^+)$ smaller and closer to k. This provides a step towards a better understanding of what is actually done in practice with unsupervised contrastive learning. For instance, $N = 65536$ negatives are used in He et al. [2020].

While we considered a generic encoder f and a generic setting in this Section, the next Section 3.4 considers a more realistic setting of an unsupervised objective with a fixed available dataset, and the study of an *explicit* algorithm for the training of f.

3.4 Convergence of gradient descent for contrastive unsupervised learning

This section leverages results from Allen-Zhu et al. [2019] to provide convergence guarantees for gradient-descent based minimization of the contrastive training error, where the unsupervisedly trained encoder is an overparametrized deep neural network.

Deep neural network encoder

We consider a family of encoders f defined as a deep feed-forward neural network following Allen-Zhu et al. [2019]. We quickly restate its structure here for the sake of completeness. A deep neural encoder f is parametrized by matrices $A \in \mathbb{R}^{m \times d_x}$, $B \in \mathbb{R}^{d \times m}$ and $W_1, \dots, W_L \in \mathbb{R}^{m \times m}$ for some depth L. For an input $x \in \mathbb{R}^{d_x}$, the feed-forward output $y \in \mathbb{R}^d$ is given by

$$\begin{aligned} &g_0 = Ax, \quad h_0 = \phi(g_0), \quad g_l = W_l h_{l-1}, \quad h_l = \phi(g_l) \quad \text{for} \quad l = 1, \dots, L, \\ &y = Bh_L, \end{aligned}$$

where ϕ is the ReLU activation function. Note that the architecture can also include residual connections and convolutions, as explained in Allen-Zhu et al. [2019].

We know from Allen-Zhu et al. [2019] that, provided a δ-separation condition on the dataset (x_i, y_i) for $i = 1, \dots, n$ with $\delta > 0$ and sufficient overparametrization of the model ($m = \Omega\left(\text{poly}(n, L, \delta^{-1}) \cdot d\right)$), the optimisation of the least-squares error $\frac{1}{2}\sum_{i=1}^n \|\widehat{y}_i - y_i\|_2^2$ using gradient descent provably converges to an arbitrarily low value $\epsilon > 0$, where $\widehat{y}_i = f(x_i)$ are the network outputs. Moreover, the convergence is linear i.e. the number of required epochs is $T = O(\log(1/\varepsilon))$, although involving a constant of order $\text{poly}(n, L, \delta^{-1})$. Although this result does not directly apply to contrastive unsupervised learning, we explain below how it can be adapted provided a few additional assumptions.

Ideally, we would like to prove a convergence result on the unsupervised objective defined in Equation (3.6). However, we need to define an objective through an explicitly given dataset so that it falls within the scope of Allen-Zhu et al. [2019]. Regarding this issue, we assume in what follows that we dispose of a set of fixed triplets $(x, x^+, x^-) \in (\mathbb{R}^{d_x})^3$ we train on.

Objective function

Let us denote this fixed training set $\{(x_i, x_i^+, x_i^-)\}_{i=1}^n$. Each element leads to an output $z_i = (f(x_i), f(x_i^+), f(x_i^-))$ by the encoder and we optimize the empirical

objective

$$\widehat{L}_{\text{un}}(f) = \sum_{i=1}^{n} \zeta(f(x_i)^T (f(x_i^-) - f(x_i^+))) = \sum_{i=1}^{n} \ell(z_i), \tag{3.11}$$

where we introduced the loss function $\ell(z_i) = \ell(z_{i,1}, z_{i,2}, z_{i,3}) = \zeta(z_{i,1}^T(z_{i,3} - z_{i,2}))$ with $\zeta(x) = \log(1 + e^x)$. Note that $\widehat{L}_{\text{un}}(f)/n$ is the empirical counterpart of the unsupervised loss (3.6). Our management of the set of training triplets can be compared to that of Wen [2020] who similarly fixes them in advance but uses multiple negatives and the same x_i as a positive. However, two distinct encoders are trained therein, one for the reference sample x_i and another for the rest. We consider here the more realistic case where a single encoder is trained. Our approach also applies to multiple negatives, but we only use a single one here for simplicity. We need the following data separation assumption from Allen-Zhu et al. [2019].

Assumption 3.1. We assume that all the samples $x \in \mathcal{X}_{\text{data}} = \bigcup_{i=1}^{n} \{x_i, x_i^+, x_i^-\}$ are normalized $\|x\| = 1$ and that there exists $\delta > 0$ such that $\|x - x'\|_2 \geqslant \delta$ for any $x, x' \in \mathcal{X}_{\text{data}}$.

Note that sampling the positives and negatives x_i^+, x_i^- need not to be made through simple draws from the dataset. A common practice in contrastive learning [Chen et al., 2020a] is to use data augmentations, where we replace $x_i^\pm$ by $\psi(x_i^\pm)$ for an augmentation function ψ also drawn at random. Such an augmentation can include, whenever inputs are color images, Gaussian noise, cropping, resizing, color distortion, rotation or a combination thereof, with parameters sampled at random in prescribed intervals. The setting considered here allows the case where $x_i^\pm$ are actually augmentations (we won't write $\psi(x_i^\pm)$ but simply $x_i^\pm$ to simplify notations), provided that Assumption 3.1 is satisfied and that such augmentations are performed and fixed before training. Note that, in practice, the augmentations are themselves randomly sampled at each training iteration [Chen et al., 2020a]. Unfortunately, this would make the objective intractable and the convergence result we are about to derive does not apply in that case.

In order to apply the convergence result from Allen-Zhu et al. [2019], we need to prove that the following gradient-Lipschitz condition

$$\ell(z + z') \leqslant \ell(z) + \langle \nabla \ell(z), z' \rangle + \frac{L_{\text{smooth}}}{2} \|z'\|^2 \tag{3.12}$$

holds for any $z, z' \in \mathbb{R}^{3d}$, for some constant $L_{\text{smooth}} > 0$, where ℓ is the loss given by (3.11). However, as defined previously, ℓ does not satisfy (3.12) without extra assumptions. We propose to bypass this problem by making the following additional assumption on the norms of the outputs of the encoder.

Assumption 3.2. For each element $x \in \mathcal{X}_{\text{data}}$, the output $z = f(x) \in \mathbb{R}^d$ satisfies

$$\eta < \|z\| < C$$

during and at the end of the training of the encoder f, for some constants $0 < \eta < C < +\infty$.

In Section 2.3, we check experimentally on three datasets (see Figure 3.3 herein) that this assumption is rather realistic. The lower bound $\eta > 0$ is necessary and used in Lemma 3.6 below, while the upper bound C is used in the next Lemma 3.4, which establishes the gradient-Lipschitz smoothness of the unsupervised loss ℓ and provides an estimation of L_{smooth}.

Lemma 3.4. *Consider the unsupervised loss ℓ given by (3.11), grant Assumption 3.2 and define the set*

$$B^3 = \Big\{ z = (z_1, z_2, z_3) \in (\mathbb{R}^d)^3 \ : \ \max_{j=1,2,3} \|z_j\|_2^2 \leqslant C^2 \Big\}$$

where $C > 0$ is defined in Assumption 3.2. Then, the restriction of ℓ to B^3 satisfies (3.12) with a constant $L_{\text{smooth}} \leqslant 2 + 8C^2$.

The proof of Lemma 3.4 is given in the appendix. Now, we can state the main result of this Section.

Theorem 3.5. *Grant both Assumptions 3.1 and 3.2, let $\epsilon > 0$ and let $\widehat{L}_{\text{un}}(f)$ be the loss given by (3.11). Then, assuming that*

$$m \geqslant \Omega\Big(\frac{\text{poly}(n, L, \delta^{-1}) \cdot d}{\epsilon}\Big),$$

the gradient descent algorithm with a learning rate ν and a number of steps T such that

$$\nu = \Theta\Big(\frac{d\delta}{\text{poly}(n, L) \cdot m}\Big) \quad \text{and} \quad T = O\Big(\frac{\text{poly}(n, L)}{\delta^2 \epsilon^2}\Big),$$

finds a parametrization of the encoder f satisfying

$$\widehat{L}_{\text{un}}(f) \leqslant \epsilon.$$

The proof of Theorem 3.5 is given in the appendix. Although it uses Theorem 6 from Allen-Zhu et al. [2019], it is actually *not* an immediate consequence of it. Indeed, in our case, the Theorem 6 therein only allows us to conclude that $\|\nabla \widehat{L}_{\text{un}}(f)\| \leqslant \epsilon$, where the gradient is taken w.r.t. the outputs of f. The convergence of the objective itself is obtained thanks to the following Lemma whose proof is given in the appendix.

Lemma 3.6. *Grant Assumption 3.2 and assume that the parameters of the encoder f are optimized so that $\|\nabla \widehat{L}_{\text{un}}(f)\| \leqslant \epsilon$ with $\epsilon < \eta/2$, where η is defined in Assumption 3.2. Then, for any $i = 1, \dots, n$, we have $\ell(z_i) \leqslant 2\epsilon/\eta$ where $z_i = (f(x_i), f(x_i^+), f(x_i^-))$.*

This Lemma is crucial for proving Theorem 3.5 as it allows to show, in this setting, that the reached critical point is in fact a global minimum.

A natural idea would be then to combine Theorem 3.5 with Proposition 3.3 in order to prove that gradient descent training of the encoder using the unsupervised contrastive loss helps to minimize the supervised loss. This paper makes a step towards such a result, but let us stress that it requires much more work, to be considered in future papers, the technical problems to be addressed being as follows. Firstly, the result of Theorem 3.5 applies to $\widehat{L}_{\text{un}}(f)$ and cannot be directly extrapolated on $L_{\text{un}}(f)$. Doing so would require a sharp control of the generalization error, while Theorem 3.5 is about the training error only. Secondly, Assumption 3.1 requires that all samples are separated and, in particular, distinct. This cannot hold when the objective is defined through an expectation as we did in Section 3.3. Indeed, it would be invalidated simply by reusing a sample in two different triples.

3.5 Experiments

In this section, we report experiments that illustrate our theoretical findings.

Datasets and Experiments

We use a small convolutional network as encoder on MNIST [LeCun and Cortes, 2010] and FashionMNIST [Xiao et al., 2017], and VGG-16 [Simonyan and Zisserman, 2015] on CIFAR-10 [Krizhevsky et al., 2009]. Experiments are performed with PyTorch [Paszke et al., 2019].

Results

Figure 3.1 provides an illustration of Lemma 3.1, where we display the values of L_{un} (i.e., L_{un}^N with $N = 1$) and $L_{\text{sup}}^{\mu}(f, \mathcal{C})$ along training iterations over 5 separate runs (and their average). We observe that Inequality (3.7) is satisfied on these experiments, even when the $\log N_{\mathcal{C}}$ term is discarded. Moreover, both losses follow a similar trend. Figure 3.2 illustrates Lemma 3.2 for several values of N. Once again, we observe that both losses behave similarly, and that Inequality (3.8) seems to hold even without the $1/p_{cc}^{\rho}$ term (removed for these displays).

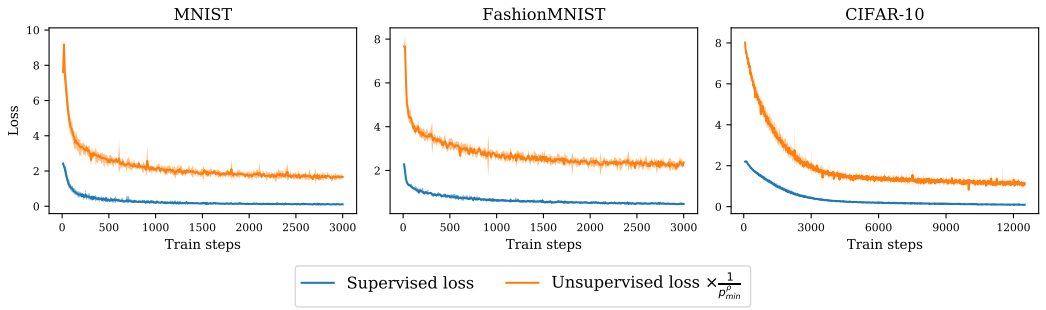

Figure 3.1 – Illustration of Lemma 3.1: we observe that Inequality (3.7) is satisfied on these examples, even without the $\log N_{\mathcal{C}}$ term, and that both losses behave similarly (5 runs are displayed together with their average).

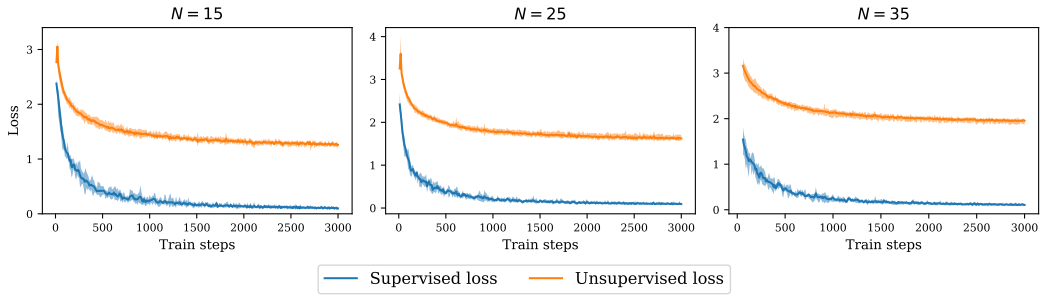

Figure 3.2 – Illustration of Lemma 3.2 with $N = 15, 25, 35$ on MNIST. We observe again that both the unsupervised and supervised losses behave similarly and that Inequality (3.8) is satisfied in these experiments, even without the $1/p_{cc}^{\rho}$ factor (5 runs are displayed together with their average).

Finally, Figure 3.3 displays the minimum and maximum Euclidean norms of the outputs of the encoder along training. On these examples, we observe that one can indeed assume these norms to be lower and upper bounded by constants, as stated in Assumption 3.2.

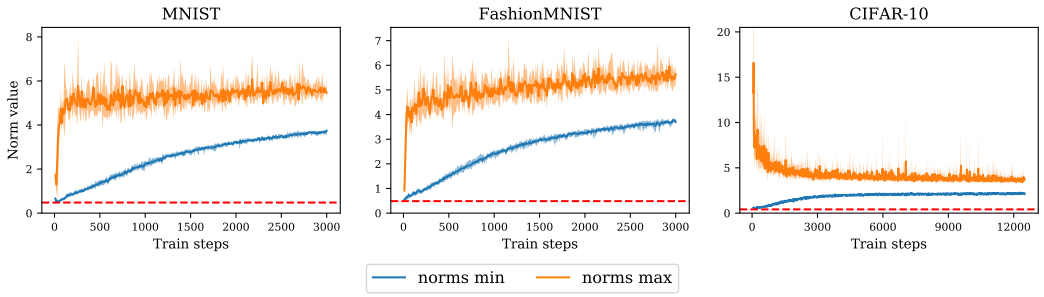

Figure 3.3 – Minimum and maximum Euclidean norms of the outputs of the encoder along contrastive unsupervised training. We observe that Assumption 3.2 is satisfied on these examples (5 runs are displayed together with their average), the dashed line shows that the minimum norms are away from 0 even in the early iterations.

3.6 Conclusion

This work provides extensions to previous results on contrastive unsupervised learning, in order to somewhat improve the theoretical understanding of the performance that is empirically observed with pre-trained encoders used for subsequent supervised task. The main hindrance to tighter bounds in Section 3.3 is the blind randomness of negative sampling, which is unavoidable in the unsupervised setting. Section 3.4 explains how recent theoretical results about gradient descent training of overparametrized deep neural networks can be used for unsupervised contrastive learning, and concludes with an explanation of why combining the results from Sections 3.3 and 3.4 requires many extra technicalities to be considered in future works. Let us conclude by stressing, once again, our motivations for doing this: unsupervised learning theory is much less developed than supervised learning theory, and recent empirical results (see Section 2.1) indicate that some forms of contrastive learning enable the learning of powerful representations without supervision. In many fields of application, labels are too difficult, too expensive or too invasive to obtain (in medical applications, see for instance Ching et al. [2018]). We believe that a better understanding of unsupervised learning is therefore of utmost importance.

3.7 Appendix: technical proofs

3.7.1 Proofs for Section 3.3

Apart from the similarity between the unsupervised and supervised loss, the proof of Lemma 3.1 uses properties of log-sum-exp.

Proof of Lemma 3.1. We first rewrite the unsupervised loss as:

$$L_{\mathrm{un}}(f) = \mathbb{E}_{(x,x^+)\sim\mathcal{D}_{\mathrm{sim}},x^-\sim\mathcal{D}_{\mathrm{neg}}} \log\left(1+\exp\left(f(x)^T(f(x^-)-f(x^+))\right)\right)$$

where we recognize the ζ function $\zeta(x) = \log(1+e^x)$. We start by using Jensen's

inequality

$$\begin{aligned}
L_{\text{un}}(f) &= \mathbb{E}_{\substack{(x,x^+)\sim\mathcal{D}_{\text{sim}} \\ x^-\sim\mathcal{D}_{\text{neg}}}} \left[\zeta\left(f(x)^T(f(x^-) - f(x^+))\right)\right] \\
&\geqslant \mathbb{E}_{c,c^-\sim\rho,x\sim\mathcal{D}_c} \left[\zeta\left(f(x)^T(\mu_{c^-} - \mu_c)\right)\right] \\
&\geqslant p_{\text{min}}^{\rho} \mathbb{E}_{c\sim\rho,x\sim\mathcal{D}_c} \left[\max_{c^-} \zeta\left(f(x)^T(\mu_{c^-} - \mu_c)\right)\right] \\
&= p_{\text{min}}^{\rho} \mathbb{E}_{c\sim\rho,x\sim\mathcal{D}_c} \left[\max_{c^-} \text{LSE}\left(0, f(x)^T(\mu_{c^-} - \mu_c)\right)\right] \\
&\geqslant p_{\text{min}}^{\rho} \left(\mathbb{E}_{c\sim\rho,x\sim\mathcal{D}_c} \left[\text{LSE}\left(f(x)^T(\mu_{c_1} - \mu_c), \ldots, f(x)^T(\mu_{c_{N_\mathcal{C}}} - \mu_c)\right)\right] - \log N_\mathcal{C}\right) \\
&= p_{\text{min}}^{\rho} \left(L_{\text{sup}}^{\mu}(f, \mathcal{C}) - \log N_\mathcal{C}\right)
\end{aligned}$$

where we have used properties of the log-sum-exp function

$$\max(x_1, \ldots, x_n) \leqslant \text{LSE}(x_1, \ldots, x_n) \leqslant \max(x_1, \ldots, x_n) + \log n,$$

the fact that LSE is non-negative whenever one of its arguments is, and for $x \in \mathbb{R}^{2n}$ we have

$$\text{LSE}(x) = \text{LSE}(\text{LSE}(x_1, x_2), \ldots, \text{LSE}(x_{2n-1}, x_{2n})) \leqslant \max_{j=1,\ldots,n} \text{LSE}(x_{2j-1}, x_{2j}) + \log n.$$

□

The proof of Lemma 3.2 considers the sample draws where all classes are represented.

Proof of Lemma 3.2. Let $I \in [N_\mathcal{C}]^N$ the random vector of classes for each negative sample ($I \sim \rho^{\otimes N}$) and let J be the set of represented classes i.e. $J = \{I_j \mid j \in [N]\}$. We have, again with Jensen's inequality

$$\begin{aligned}
L_{\text{un}}^N(f) &= \mathbb{E}_{x,x^+,x_1^-,\ldots,x_N^-} \left[\text{LSE}\left(0, f(x)^T(f(x_1^-) - f(x^+)), \ldots, f(x)^T(f(x_N^-) - f(x^+))\right)\right] \\
&\geqslant \mathbb{E}_{c\sim\rho,I\sim\rho^{\otimes N},x\sim\mathcal{D}_c} \left[\text{LSE}\left(0, f(x)^T(\mu_{I_1} - \mu_c), \ldots, f(x)^T(\mu_{I_N} - \mu_c)\right)\right] \\
&\geqslant \mathbb{P}\left(|J| = N_\mathcal{C}\right) \mathbb{E}_{\substack{c\sim\rho \\ I\sim\rho^{\otimes N} \\ x\sim\mathcal{D}_c}} \left[\text{LSE}\left(0, f(x)^T(\mu_{I_1} - \mu_c), \ldots, f(x)^T(\mu_{I_N} - \mu_c)\right) \mid |J| = N_\mathcal{C}\right] \\
&\geqslant \mathbb{P}\left(|J| = N_\mathcal{C}\right) L_{\text{sup}}^{\mu}(f, \mathcal{C}),
\end{aligned}$$

where we used that for $\mathcal{S} \subset [n]$ and $x \in \mathbb{R}^n$ we have $\text{LSE}(x_\mathcal{S}) \leqslant \text{LSE}(x)$ with $x_\mathcal{S}$ the restriction of x to the indices in $\mathcal{S}$. Finally, we have $\mathbb{P}\left(|J| = N_\mathcal{C}\right) = p_{cc}^{\rho}(N)$. □

We restate Proposition 3.3 for cases $N = 1$ and $N > 1$. The proof uses Jensen's inequality and the uniformity of ρ.

Proposition 3.3 (restated). *Consider the unsupervised loss $L_{\text{un}}^N(f)$ from Equation (3.6) with N negative samples. Assume that ρ is uniform over $\mathcal{C}$ and that $2 \leqslant k + 1 \leqslant N_\mathcal{C}$. Then,*
(1) any encoder function $f : \mathcal{X} \to \mathbb{R}^d$ satisfies

$$L_{\text{sup},k}(f) \leqslant L_{\text{sup},k}^{\mu}(f) \leqslant \frac{k}{1 - \tau^+}\left(L_{\text{un}}(f) - \tau^+\right)$$

with $\tau^+ = \mathbb{P}_{c,c'\sim\rho^2}(c = c')$, where $L_{\text{un}}(f)$ is the unsupervised loss from Equation (3.6) with $N = 1$ negative sample;

(2) more generally,

$$L_{\text{sup},k}(f) \leqslant L^{\mu}_{\text{sup},k}(f) \leqslant \frac{k}{1-\tau^+_N}\left(L^N_{\text{un}}(f) - \tau^+_N \log(N+1)\right)$$

with $\tau^+_N = \mathbb{P}(c_i = c, \forall i \mid c \sim \rho, (c_1, \cdots, c_N) \sim \rho^N)$*, and where* $L^N_{\text{un}}(f)$ *is the unsupervised loss from Equation* (3.6)*.*

Proof of Proposition 3.3. Let's start with (1). By Jensen's inequality, then use $\log = \log_2$ without loss of generality, and split the expectation into cases $c^- \neq c$ and $c^- = c$,

$$\begin{aligned}
L_{\text{un}}(f) &= \mathbb{E}_{(c,c^-)\sim\rho^2}\mathbb{E}_{x,x^+\sim\mathcal{D}_c,x^-\sim\mathcal{D}_{c^-}}\left[\log\left(1+\exp\left(f(x)^T\left(f(x^-)-f(x^+)\right)\right)\right)\right] \\
&\geqslant \mathbb{E}_{(c,c^-)\sim\rho^2,x\sim\mathcal{D}_c}\left[\log\left(1+\exp\left(f(x)^T\left(\mu_{c^-}-\mu_c\right)\right)\right)\right] \\
&= (1-\tau^+)\mathbb{E}_{c\sim\rho,x\sim\mathcal{D}_c}\mathbb{E}_{c^-\sim\rho}\left[\log\left(1+\exp\left(f(x)^T\left(\mu_{c^-}-\mu_c\right)\right)\right)\middle| c^- \neq c\right] + \tau^+.
\end{aligned}$$

Let us write explicitly the uniform distribution ρ on $\mathcal{C}$. On the one hand,

$$\begin{aligned}
&\mathbb{E}_{c^-\sim\rho}\left[\log\left(1+\exp\left(f(x)^T\left(\mu_{c^-}-\mu_c\right)\right)\right)\middle| c^- \neq c\right] \\
=&\sum_{c^-\in\mathcal{C}\backslash\{c\}}\frac{1}{N_{\mathcal{C}}-1}\log\left(1+\exp\left(f(x)^T\left(\mu_{c^-}-\mu_c\right)\right)\right),
\end{aligned}$$

on the other hand, And this is for every $c^- \in \mathcal{C}\backslash\{c\}$. We rearrange the double sum according to c^-

Hence, using the uniformity of ρ,

$$\begin{aligned}
&\mathbb{E}_{c^-\sim\rho}\left[\log\left(1+\exp\left(f(x)^T\left(\mu_{c^-}-\mu_c\right)\right)\right)\middle| c^- \neq c\right] \\
=&\frac{1}{k}\mathbb{E}_{c_1,\ldots,c_k\sim\rho^{\otimes k}}\left[\sum_{i=1}^k\log\left(1+\exp\left(f(x)^T\left(\mu_{c_i}-\mu_c\right)\right)\right)\middle|\{c,c_1,\ldots,c_k\}\text{ distinct}\right] \\
\geqslant&\frac{1}{k}\mathbb{E}_{c_1,\ldots,c_k\sim\rho^{\otimes k}}\left[\log\left(1+\sum_{i=1}^k\exp\left(f(x)^T\left(\mu_{c_i}-\mu_c\right)\right)\right)\middle|\{c,c_1,\ldots,c_k\}\text{ distinct}\right].
\end{aligned}$$

That means we have

$$\begin{aligned}
L_{\text{un}}(f) &\geqslant \frac{1-\tau^+}{k}\mathbb{E}_{\substack{c\sim\rho,x\sim\mathcal{D}_c\\ c_1,\ldots,c_k\sim\rho^{\otimes k}}}\left[\log\left(1+\sum_{i=1}^k\exp\left(f(x)^T\left(\mu_{c_i}-\mu_c\right)\right)\right)\middle|\{c,c_1,\ldots,c_k\}\text{ distinct}\right]+\tau^+ \\
&= \frac{1-\tau^+}{k}\mathbb{E}_{\mathcal{T}\sim\mathcal{D}^{k+1}}\mathbb{E}_{(x,c)\sim\mathcal{D}_{\mathcal{T}}}\left[-\log\left(\frac{\exp(f(x)^T\mu_c)}{\exp(f(x)^T\mu_c)+\sum_{\substack{c^-\in\mathcal{T}\\ c^-\neq c}}\exp\left(f(x)^T\mu_{c^-}\right)}\right)\right]+\tau^+ \\
&= \frac{1-\tau^+}{k}L^{\mu}_{\text{sup},k}(f)+\tau^+.
\end{aligned}$$

As for (2), again by Jensen's inequality, and split the expectation into cases

$c_i^- = c, \forall i$ and $\exists c_i^- \neq c$,

$$L_{\text{un}}^N(f) = \mathbb{E}_{(c,c_i^-)\sim\rho^{N+1}} \mathbb{E}_{x,x^+\sim\mathcal{D}_c, x_i^-\sim\mathcal{D}_{c_i^-}} \left[\log\left(1 + \sum_{i=1}^N \exp\left(f(x)^T\left(f(x_i^-) - f(x^+)\right)\right)\right)\right]$$
$$\geqslant \mathbb{E}_{(c,c_i^-)\sim\rho^{N+1}, x\sim\mathcal{D}_c} \left[\log\left(1 + \sum_{i=1}^N \exp\left(f(x)^T\left(\mu_{c_i^-} - \mu_c\right)\right)\right)\right]$$
$$= (1-\tau_N^+)\mathbb{E}_{\substack{c\sim\rho \\ x\sim\mathcal{D}_c}} \mathbb{E}_{c_i^-\sim\rho^N} \left[\log\left(1 + \sum_{i=1}^N \exp\left(f(x)^T\left(\mu_{c^-} - \mu_c\right)\right)\right) \middle| \exists c_i^- \neq c\right] + \tau_N^+ \log(N+1)$$

with

$$\tau_N^+ = \mathbb{P}(c_i = c, \forall i \mid c\sim\rho, c_i\sim\rho^N) = \sum_{c\in\mathcal{C}} \rho(c)^{N+1} = N_{\mathcal{C}}^{-N}.$$

Considering the fact that

$$\mathbb{E}_{c_i^-\sim\rho^N} \left[\log\left(1 + \sum_{i=1}^N \exp\left(f(x)^T\left(\mu_{c^-} - \mu_c\right)\right)\right) \middle| \exists c_i^- \neq c\right] \geqslant$$
$$\mathbb{E}_{c^-\sim\rho}\left[\log\left(1 + \exp\left(f(x)^T\left(\mu_{c^-} - \mu_c\right)\right)\right) \middle| c^- \neq c\right],$$

then by similar computations as in (1), we have

$$L_{\text{un}}^N(f) \geqslant \frac{1-\tau_N^+}{k} L_{\text{sup},k}^{\mu}(f) + \tau_N^+ \log(N+1).$$

□

3.7.2 Proofs for Section 3.4

Let us first prove that under Assumption 3.2, the objective is gradient-Lipschitz w.r.t. the network outputs.

Lemma 4.1. *Consider the unsupervised loss ℓ given by (3.11), grant Assumption 3.2 and define the set*

$$B^3 = \left\{z = (z_1, z_2, z_3) \in (\mathbb{R}^d)^3 \ : \ \max_{j=1,2,3} \|z_j\|_2^2 \leqslant C^2\right\}$$

where $C > 0$ is defined in Assumption 3.2. Then, the restriction of ℓ to B^3 satisfies (3.12) with a constant $L_{\text{smooth}} \leqslant 2 + 8C^2$.

Proof. We will prove this result by bounding the norm of the Hessian matrix.

Let us write the gradient of $\ell(z)$ with respect to z first. We have $z \in \mathbb{R}^{3d}$. For ease of writing, we define the matrices $A_1, A_2, A_3 \in \mathbb{R}^{3d\times d}$ as

$$A_1 = \begin{pmatrix} I_d \\ 0_d \\ 0_d \end{pmatrix} \quad A_2 = \begin{pmatrix} 0_d \\ I_d \\ 0_d \end{pmatrix} \quad A_3 = \begin{pmatrix} 0_d \\ 0_d \\ I_d \end{pmatrix}$$

where $I_d, 0_d \in \mathbb{R}^{d\times d}$ are the identity and zero matrix respectively. With this notation, we have $z_i = A_i^T z$ for $i = 1, 2, 3$ the three contiguous thirds of z's coordinates.

Our purpose is to compute

$$\frac{\partial}{\partial z}\ell(z) = \frac{\partial}{\partial z}\left[-\log\left(\frac{\exp\left(z_1^T z_2\right)}{\exp\left(z_1^T z_2\right)+\exp\left(z_1^T z_3\right)}\right)\right].$$

Denote $\cos_{i,j} = z_i^T z_j$, we can now compute for $i,j \in \{1,2,3\}$ $(i \neq j)$

$$\frac{\partial}{\partial z}\cos_{i,j} = \left(A_i A_j^T + A_j A_i^T\right) z =: \partial \cos_{i,j} \in \mathbb{R}^{3d}.$$

Now, denote $v = \operatorname{softmax}\left(\cos_{1,2}, \cos_{1,3}\right) \in \mathbb{R}^2$, we can write

$$\frac{\partial}{\partial z}\ell(z) = (v_1 - 1)\partial\cos_{1,2} + v_2\partial\cos_{1,3}\,.$$

We proceed with the following computation

$$\frac{\partial^2}{\partial z^2}\cos_{i,j} = A_i A_j^T + A_j A_i^T,$$

which we will denote simply as $\partial^2 \cos_{i,j}$. Before we get the Hessian of loss, we still need to compute

$$\partial v := \frac{\partial v}{\partial z} = \left(\operatorname{diag}(v) - vv^T\right)\begin{pmatrix}\partial\cos_{1,2}^T \\ \partial\cos_{1,3}^T\end{pmatrix} \in \mathbb{R}^{2\times 3d}.$$

Now we can write

$$\frac{\partial^2}{\partial z^2}\ell(z) = (v_1-1)\partial^2\cos_{1,2} + v_2\partial^2\cos_{1,3} + \begin{pmatrix}\partial\cos_{1,2} & \partial\cos_{1,3}\end{pmatrix}\partial v.$$

We can now estimate the norm of this matrix which will provide an estimation for the Lipschitz constant.

We find that

$$\left\|\partial\cos_{i,j}\right\| \leqslant 2\max\left(\left\|z_i\right\|, \left\|z_j\right\|\right),$$

keeping in mind that the matrix $\operatorname{diag}(v) - vv^T$ has norm at most $1/2$, this leads to

$$\left\|\begin{pmatrix}\partial\cos_{1,2} & \partial\cos_{1,3}\end{pmatrix}\partial v\right\| = 8\max_{i,j}\left(\left\|z_i\right\|\left\|z_j\right\|\right).$$

We have also that $\left\|\partial^2\cos_{i,j}\right\| = 1$.

All in all, we have $\left\|\frac{\partial^2}{\partial z^2}\ell(z)\right\| = 2 + 8\max_{i,j}\left(\left\|z_i\right\|\left\|z_j\right\|\right)$. Recalling that we restricted $\mathbb{R}^{3d}$ so that we have $\max_i \left\|z_i\right\| \leqslant C$ the result follows. □

Theorem 3.5 is actually obtained in two steps. First, Theorem 6 from Allen-Zhu et al. [2019] allows us to obtain that the gradient of the objective $\nabla\widehat{L}_{\text{un}}(f)$ with respect to the network outputs reaches arbitrarily low values. Then, combining this with Assumption 3.2, this result can be extended into the objective itself.

Following appendix A of Allen-Zhu et al. [2019], we need to define the ℓ vectors for our model. These are originally defined as $\ell_i = y_i - y_i^*$ (y_i and y_i^* are respectively the output and label corresponding to an input x_i from the dataset) for the ℓ^2 loss. More generally, for a network output z_i, they are defined as

$$\ell_i = \nabla_z \ell(z_i).$$

Following the unsupervised training protocol, samples are fed into the network three at a time x, x^+ and x^-. Let us denote θ the parameters of the network f, for a triplet (x_i, x_i^+, x_i^-), the trick is to write:

$$\frac{\partial}{\partial \theta}\ell(z_i) = \frac{\partial z}{\partial \theta}\underbrace{\frac{\partial}{\partial z}\ell(z_i)}_{\ell}$$

with z_i the concatenation of $f(x_i), f(x_i^+), f(x_i^-)$.

By denoting $(x_1, x_2, x_3) = (x_i, x_i^+, x_i^-)$, the previous writing is equivalent to

$$\sum_{j=1}^{3} \frac{\partial f(x_j)}{\partial \theta} A_j^T \frac{\partial}{\partial z}\ell(z_i)$$

and by letting $\ell_{i,j} = A_j^T \frac{\partial}{\partial z}\ell(z_i)$, we obtain a triplet of loss vectors for each data triple (matrices A_j defined in the previous proof).

Lemma 3.7. *Grant Assumption 3.1 and let $\widehat{L}_{\text{un}}(f)$ be the loss incurred by f:*

$$\widehat{L}_{\text{un}}(f) = \sum_{i=1}^{n} \ell(f(x_i), f(x_i^+), f(x_i^-))$$

and let $\epsilon > 0$ be the desired precision. Then, assuming $m \geqslant \Omega\left(\text{poly}(n, L, \delta^{-1}) \cdot d\epsilon^{-1}\right)$, the gradient descent with learning rate $\nu = \Theta\left(\frac{d\delta}{\text{poly}(n,L)\cdot m}\right)$ finds parameters such that

$$\|\nabla \widehat{L}_{\text{un}}(f)\| \leqslant \epsilon$$

after a number of steps $T = O\left(\frac{\text{poly}(n,L)}{\delta^2\epsilon^2}\right)$.

Proof. This result follows from Allen-Zhu et al. [2019] (see Theorem 6 and appendix A). It corresponds to the case of a non-convex bounded loss function. We only need to check the used loss function ℓ is bounded and gradient-Lipschitz smooth. The latter condition is verified due to Lemma 4.1 and Assumption 3.2.

As for the boundedness, it is also a consequence of Assumption 3.2 and the fact that the softplus function satisfies

$$\zeta(x) \sim^{x\to+\infty} x \quad \text{and} \lim_{x\to-\infty} \zeta(x) = 0.$$

□

From here, we can derive a result for the objective itself (Theorem 3.5) thanks to the following Lemma.

Lemma 4.2. *Grant Assumption 3.2 and assume that the parameters of the encoder f are optimized so that $\|\nabla \widehat{L}_{\text{un}}(f)\| \leqslant \epsilon$ with $\epsilon < \eta/2$, where η is defined in Assumption 3.2. Then, for any $i = 1, \ldots, n$, we have $\ell(z_i) \leqslant 2\epsilon/\eta$ where $z_i = (f(x_i), f(x_i^+), f(x_i^-))$.*

Proof. Since we assume $\|\nabla \widehat{L}_{\text{un}}(f)\| \leqslant \epsilon$, this also implies that $\max_{i,j} \|\ell_{i,j}\| \leqslant \epsilon$ (see Theorem 3 of Allen-Zhu et al. [2019] and its variant in appendix A).

We can write the norms $\|\ell_{i,j}\|$ as:

$$\begin{aligned}
\|\ell_{i,1}\| &= \|(v_1 - 1)z_{i,2} + v_2 z_{i,3}\| \\
\|\ell_{i,2}\| &= |v_1 - 1| \, \|z_{i,1}\| \\
\|\ell_{i,3}\| &= v_2 \, \|z_{i,1}\|
\end{aligned}$$

where we defined $v = \text{softmax}(z_1^T z_2, z_1^T z_3)$.

Thanks to Assumption 3.2, we can argue that $\|z_{i,j}\| \geqslant \eta$. These quantities can be small for $v_1 \to 1$ and $v_2 \to 0$. Since we have $\max_{i,j} \|\ell_{i,j}\| \leqslant \epsilon$, this implies in particular that for all i we get $\|\ell_{i,3}\| \leqslant \epsilon$ which means $v_2 \leqslant \epsilon/\eta$, and we have $v_2 = \sigma(z_1^T(z_3 - z_2))$. So for an instance $i \in [n]$ the loss term in the objective is:

$$\begin{aligned}
\zeta(z_{i,1}^T(z_{i,3} - z_{i,2})) &= \log(1 + \exp(z_{i,1}^T(z_{i,3} - z_{i,2}))) \\
&= -\log(\sigma(-z_{i,1}^T(z_{i,3} - z_{i,2}))) \\
&= -\log(1 - \sigma(z_{i,1}^T(z_{i,3} - z_{i,2}))) \\
&= -\log(1 - v_2) \leqslant \frac{v_2}{1 - v_2} \leqslant 2v_2 \leqslant 2\epsilon/\eta,
\end{aligned}$$

where we used the inequality $-\log(1-x) \leqslant \frac{x}{1-x}$ for $0 \leqslant x < 1$, and the assumption that $\epsilon < \eta/2$. □

Lemma 4.2 allows us to deduce that the objective is well optimized (we treated the loss term for a single triplet here but the same methods can be applied to the whole objective with a number of gradient steps which is still polynomial).

Proof of Theorem 3.5. Theorem 3.5 is the consequence of combining Lemma 3.7 applied using $\frac{\epsilon\eta}{2n}$ instead of ϵ and Lemma 4.2 (the $1/n$ factor can be absorbed by the $\text{poly}(n, L)$ factors in the bounds of Lemma 3.7). □

Bibliography

Z. Allen-Zhu, Y. Li, and Z. Song. A Convergence Theory for Deep Learning via Over-Parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019. 73, 77, 78, 79, 85, 86

M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments, 2020. 72

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations, 2020a. 72, 73, 78

T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton. Big self-supervised models are strong semi-supervised learners, 2020b. 72

T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P.-M. Agapow, M. Zietz, M. M. Hoffman, et al. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387, 2018. 81

C.-Y. Chuang, J. Robinson, L. Yen-Chen, A. Torralba, and S. Jegelka. Debiased contrastive learning, 2020. 73

J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805. 72

C. Doersch and A. Zisserman. Multi-task self-supervised visual learning. *CoRR*, abs/1708.07860, 2017. URL http://arxiv.org/abs/1708.07860. 72

C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. *CoRR*, abs/1505.05192, 2015. URL http://arxiv.org/abs/1505.05192. 72

A. Dosovitskiy, J. T. Springenberg, M. A. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. *CoRR*, abs/1406.6909, 2014. URL http://arxiv.org/abs/1406.6909. 72

J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi. Unsupervised scalable representation learning for multivariate time series. In *Advances in Neural Information Processing Systems*, pages 4650–4661, 2019. 72

S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. *CoRR*, abs/1803.07728, 2018. URL http://arxiv.org/abs/1803.07728. 72

J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020. 72

K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020. 72, 73, 77

O. J. Hénaff, A. Srinivas, J. D. Fauw, A. Razavi, C. Doersch, S. M. A. Eslami, and A. van den Oord. Data-efficient image recognition with contrastive predictive coding. *CoRR*, abs/1905.09272, 2019. URL http://arxiv.org/abs/1905.09272. 72

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009. 80

Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/. 80

J. D. Lee, Q. Lei, N. Saunshi, and J. Zhuo. Predicting what you already know helps: Provable self-supervised learning, 2020. 72

L. Logeswaran and H. Lee. An efficient framework for learning sentence representations. In *International Conference on Learning Representations*, 2018. 72

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. 72

R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. doi: 10.1017/CBO9780511814075. 75

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett,

editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. 80

N. Saunshi, O. Plevrakis, S. Arora, M. Khodak, and H. Khandeparkar. A theoretical analysis of contrastive unsupervised representation learning. In *International Conference on Machine Learning*, pages 5628–5637, 2019. 72, 73, 74, 75, 76

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 80

A. Srinivas, M. Laskin, and P. Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning, 2020. 72

C. Sun, F. Baradel, K. Murphy, and C. Schmid. Learning video representations using contrastive bidirectional transformer. *arXiv preprint arXiv:1906.05743*, 2019. 72

A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748. 72

T. Wang and P. Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere, 2020. 72, 73

Z. Wen. Convergence of end-to-end training in deep unsupervised contrasitive learning, 2020. 73, 78

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, 2017. 80

R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. *CoRR*, abs/1603.08511, 2016. URL http://arxiv.org/abs/1603.08511. 72

Chapter 4

WildWood: a new Random Forest Algorithm [1]

“ 莫听穿林打叶声，何妨吟啸且徐行。竹杖芒鞋轻胜马，谁怕？一蓑烟雨任平生。
Listen not to the rain beating against the trees, Why don't you slowly walk and chant at ease? Better than saddled horse I like sandals and cane, O I would fain? Spend a straw-cloaked life in mist and rain. ”

Su Shi (translated by Xu Yuanchong)

Contents

[1] This chapter is based on the joint work with S. Gaiffas, I. Merad.

Abstract

We introduce WildWood (WW), a new ensemble algorithm for supervised learning of Random Forest (RF) type. While standard RF algorithms use bootstrap out-of-bag samples to compute out-of-bag scores, WW uses these samples to produce improved predictions given by an aggregation of all possible subtrees of each fully grown tree in the forest. This is achieved by aggregation with exponential weights computed over out-of-bag samples, that are computed exactly and very efficiently thanks to an algorithm called context tree weighting. This improvement, combined with a histogram strategy to accelerate split finding, makes WW fast and competitive compared with other well-established ensemble methods, such as standard RF and extreme gradient boosting algorithms.

4.1 Introduction

We introduce WildWood (WW), a new ensemble method of Random Forest (RF) type [Breiman, 2001]. The main contributions of this work and the main advantages of WW are as follows.

Firstly, we use out-of-bag samples (trees in a RF use different bootstrapped samples) very differently than what is done in standard RF [Biau and Scornet, 2016; Louppe, 2014]. Indeed, WW uses these samples to compute an aggregation of the predictions of all possible subtrees of each tree in the forest, using aggregation with exponential weights [Catoni, 2004]. This leads to much improved predictions: while only leaves contribute to the predictions of a tree in standard RF, the full tree structure contributes to predictions in WW. An illustration of this effect is given in Figure 4.1 on a toy binary classification example, where we can observe that subtrees aggregation leads to improved and regularized decision functions for each individual tree and for the forest. We further illustrate in Figure 4.2 that each tree

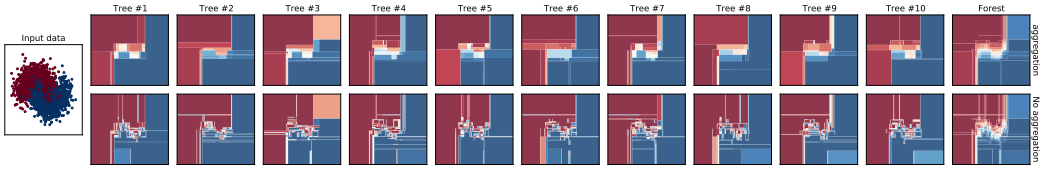

Figure 4.1 – WW decision functions illustrated on a toy dataset (left) with subtrees aggregation (top) and without it (bottom). Subtrees aggregation improves trees predictions, as illustrated by smoother decision functions in the top compared with the bottom, improving overall predictions of the forest (last column).

becomes a stronger learner, and that excellent performance can be achieved even when WW uses few trees. A remarkable aspect of WW is that this improvement comes only at a small computational cost, thanks to a technique called "context tree weighting", used in lossless compression or online learning to aggregate all subtrees of a given tree [Catoni, 2004; Helmbold and Schapire, 1997; Mourtada et al., 2021; Willems, 1998; Willems et al., 1995]. Also, the predictions of WW do not rely on MCMC approximations required with Bayesian variants of RF [Chipman et al., 1998, 2010; Denison et al., 1998; Taddy et al., 2011], which makes our approach drastically different from such approaches.

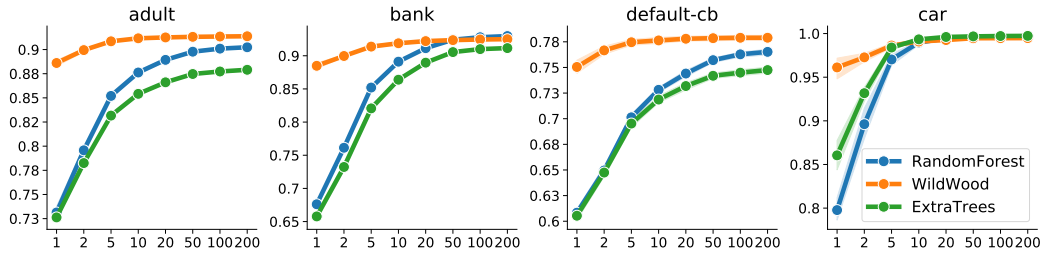

Figure 4.2 – Mean test AUC and standard-deviations (y-axis) using 10 train/test splits for WW and `scikit-learn`'s implementations of RF [Louppe, 2014] and Extra Trees [Geurts et al., 2006], using default hyperparameters, on several datasets. Thanks to subtrees aggregation, WW improves these baselines, even with few trees (x-axis is the number of trees).

Secondly, WW uses feature binning ("histogram" strategy), similarly to what is done in extreme gradient boosting (EGB) libraries such as XGBoost [Chen and Guestrin, 2016], LightGBM [Ke et al., 2017] and CatBoost [Dorogush et al., 2018; Prokhorenkova et al., 2017]. This strategy helps to accelerate computations in WW compared with standard RF algorithms, that typically require to sort features locally in nodes and try a larger number of splits [Louppe, 2014]. This combination of subtrees aggregation and of the histogram strategy makes WW comparable with state-of-the-art implementations of EGB libraries, as illustrated in Figure 4.3. Moreover, WW supports optimal split finding for categorical features and missing

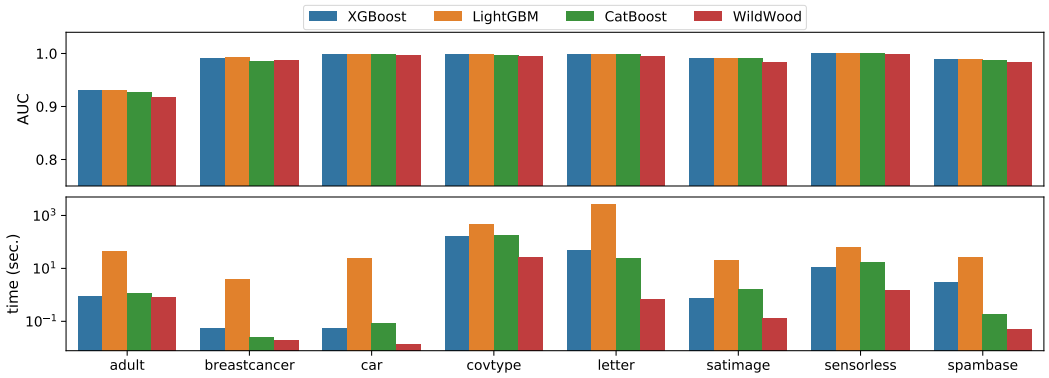

Figure 4.3 – Test AUC (top) and training time (bottom) of WW compared with very popular EGB libraries (after hyperoptimization of all algorithms, see Section 2.3 for details). WW's performance, which uses only 10 trees in this display, is only slightly below such strong baselines, but is faster (training times are on a logarithmic scale) on the considered datasets.

values, with no need for particular pre-processing (such as one-hot encoding [Chen and Guestrin, 2016] or target encoding [Dorogush et al., 2018; Prokhorenkova et al., 2017]). Finally, WW is supported by some theoretical evidence, since we prove that for a general loss function, the subtrees aggregation considered in WW leads indeed to a performance close to that of the best subtree.

Related works

Since their introduction [Breiman, 2001], RF algorithms have become one of the most popular supervised learning algorithm thanks to their ease of use, robustness to hyperparameters [Biau and Scornet, 2016; Probst et al., 2019] and applicability to

a wide range of domains, recent examples include bioinformatics [Qi, 2012], genomic data [Chen and Ishwaran, 2012], predictive medicine [Alghatani et al., 2021; Subasi et al., 2017], intrusion detection [Chen et al., 2021], astronomy [Hoffman et al., 2021], car safety [Tavakoli et al., 2021], differential privacy [Patil and Singh, 2014], COVID-19 [She et al., 2020] among many others. A non-exhaustive list of developments about RF methodology include extremely randomized forests [Geurts et al., 2006], decision forests [Criminisi et al., 2012], prediction intervals [Calviño, 2020; Roy and Larocque, 2020; Zhang et al., 2020], ranking [Zhou and Qiu, 2018], nonparametric smoothing [Verdinelli and Wasserman, 2021], variable importance Kazemitabar et al. [2017]; Louppe et al. [2013]; Loyal et al. [2021], combination with boosting [Ghosal and Hooker, 2021], generalized RF [Athey et al., 2018], robust forest Li and Martin [2017], online learning [Lakshminarayanan et al., 2014; Mourtada et al., 2021] and results aiming at a better theoretical understanding of RF [Arlot and Genuer, 2014; Biau, 2012; Biau et al., 2008; Genuer, 2012; Mentch and Zhou, 2020; Mourtada et al., 2020; Scornet, 2016a,b; Scornet et al., 2015; Zhou and Mentch, 2021].

A recent empirical study Zhou and Mentch [2021] suggests that tree depth limitation in RF is an effective regularization mechanism which improves performance on low signal-to-noise ratio datasets. In Mourtada et al. [2021], an improvement of Mondrian Forests [Lakshminarayanan et al., 2014] is introduced for online learning, using subtrees aggregation with exponential weights, which is particularly convenient in the online learning setting. However, this paper considers only the online setting, with purely random trees (splits are not optimized using training data), leading to poor performances compared with realistic decision trees. In WW, we use a similar subtrees aggregation mechanism for batch learning differently: we make the most of the bootstrap, one of the key ingredients of RF, which provides in-the-bag and out-of-bag samples, to perform aggregation with exponential weights, together with efficient decision trees grown using the histogram strategy.

Extreme boosting algorithms are another type of ensemble methods. XGBoost [Chen and Guestrin, 2016] provides an extremely popular scalable tree boosting system which has been widely adopted in industry. LightGBM [Ke et al., 2017] introduced the "histogram strategy" for faster split finding, together with clever downsampling and features grouping algorithms in order to achieve high performance in reduced computation times. CatBoost [Probst et al., 2019] is another boosting library which pays particular attention to categorical features using target encoding, while addressing the potential bias issues associated to such an encoding.

Limitations. Our implementation of WW is still evolving and is not yet at the level of maturity of state-of-the-art EGB libraries such as [Chen and Guestrin, 2016; Ke et al., 2017; Probst et al., 2019]. It does not outperform such strong baselines, but proposes an improvement of RF algorithms, and gives an interesting balance between performance and computational efficiency.

4.2 WildWood: a new Random Forest algorithm

We consider batch supervised learning, where data comes as a set of i.i.d training samples (x_i, y_i) for $i = 1, \dots, n$ with vectors of numerical or categorical features $x_i \in \mathcal{X} \subset \mathbb{R}^d$ and $y_i \in \mathcal{Y}$. Our aim is to design a RF predictor $\widehat{g}(\,\cdot\,; \mathbf{\Pi}) =$

$\frac{1}{M}\sum_{m=1}^{M}\widehat{f}(\,\cdot\,;\Pi_m) : \mathcal{X} \to \widehat{\mathcal{Y}}$ computed from training samples, where $\widehat{\mathcal{Y}}$ is the prediction space. Such a RF computes the average of M randomized trees predictions $\widehat{f}(\,\cdot\,;\Pi_m)$ following the principle of bagging [Breiman, 1996; Oza and Russell, 2001], with $\mathbf{\Pi} = (\Pi_1, \ldots, \Pi_M)$ where $\Pi_1, \ldots, \Pi_M$ are i.i.d realizations of a random variable corresponding to bootstrap and feature subsampling (see Section 4.2.1 below). Each tree is trained independently of each other, in parallel. In what follows we describe only the construction of a single tree and omit from now on the dependence on $m = 1, \ldots, M$.

Feature binning

The split finding strategy described in Section 4.2.1 below works on binned features. While this technique is of common practice in EGB libraries [Chen and Guestrin, 2016; Ke et al., 2017; Prokhorenkova et al., 2017], we are not aware of an implementation of it for RF. The input $n \times d$ matrix $\boldsymbol{X}$ of features is transformed into another same-size matrix of "binned" features denoted $\boldsymbol{X}^{\text{bin}}$. To each input feature $j = 1, \ldots, d$ is associated a set $B_j = \{1, \ldots, b_j\}$ of bins, where $b_j \leqslant b_{\max}$ with $b_{\max}$ a hyperparameter corresponding to the maximum number of bins a feature can use (default is $b_{\max} = 256$ similarly to Ke et al. [2017], so that a single byte can be used for entries of $\boldsymbol{X}^{\text{bin}}$). When a feature is continuous, it is binned into $b_{\max}$ bins using inter-quantile intervals. If it is categorical, each modality is mapped to a bin whenever $b_{\max}$ is larger than its number of modalities, otherwise sparsest modalities end up binned together. If a feature j contains missing values, its rightmost bin in B_j is used to encode them. After binning, each column satisfies $\boldsymbol{X}^{\text{bin}}_{\bullet,j} \in B_j^n$.

4.2.1 Random trees

Let $C = \prod_{j=1}^{d} B_j$ be the binned feature space. A random tree is a pair $(\mathcal{T}, \Sigma)$, where $\mathcal{T}$ is a finite ordered binary tree and Σ contains information about each node in $\mathcal{T}$, such as split information. The tree is random and its source of randomness Π comes from the bootstrap and feature subsampling as explained below.

Finite ordered binary trees

A finite ordered binary tree $\mathcal{T}$ is represented as a finite subset of the set $\{0,1\}^* = \bigcup_{n\geqslant 0}\{0,1\}^n$ of all finite words on $\{0,1\}$. The set $\{0,1\}^*$ is endowed with a tree structure (and called the complete binary tree): the empty word `root` is the root, and for any $\mathbf{v} \in \{0,1\}^*$, the left (resp. right) child of $\mathbf{v}$ is $\mathbf{v}0$ (resp. $\mathbf{v}1$). We denote by $\text{intnodes}(\mathcal{T}) = \{\mathbf{v} \in \mathcal{T} : \mathbf{v}0, \mathbf{v}1 \in \mathcal{T}\}$ the set of its interior nodes and by $\text{leaves}(\mathcal{T}) = \{\mathbf{v} \in \mathcal{T} : \mathbf{v}0, \mathbf{v}1 \notin \mathcal{T}\}$ the set of its leaves, both sets are disjoint and the set of all nodes is $\text{nodes}(\mathcal{T}) = \text{intnodes}(\mathcal{T}) \cup \text{leaves}(\mathcal{T})$.

Splits and cells

The split $\sigma_{\mathbf{v}} = (j_{\mathbf{v}}, t_{\mathbf{v}}) \in \Sigma$ of each $\mathbf{v} \in \text{intnodes}(\mathcal{T})$ is characterized by its dimension $j_{\mathbf{v}} \in \{1, \ldots, d\}$ and a subset of bins $t_{\mathbf{v}} \subsetneq \{1, \ldots, b_{j_{\mathbf{v}}}\}$. We associate to each $\mathbf{v} \in \mathcal{T}$ a cell $C_{\mathbf{v}} \subseteq C$ which is defined recursively: $C_{\texttt{root}} = C$ and for each $\mathbf{v} \in \text{intnodes}(\mathcal{T})$ we define

$$C_{\mathbf{v}0} := \{x \in C_{\mathbf{v}} : x_{j_{\mathbf{v}}} \in t_{\mathbf{v}}\} \quad \text{and} \quad C_{\mathbf{v}1} := C_{\mathbf{v}} \setminus C_{\mathbf{v}0}.$$

When $j_{\mathbf{v}}$ corresponds to a continuous feature, bins have a natural order and $t_{\mathbf{v}} = \{1, 2, \ldots, s_{\mathbf{v}}\}$ for some bin threshold $s_{\mathbf{v}} \in B_{j_{\mathbf{v}}}$; while for a categorical split, the whole set $t_{\mathbf{v}}$ is required. By construction, $(C_{\mathbf{v}})_{\mathbf{v} \in \text{leaves}(\mathcal{T})}$ is a partition of C.

Bootstrap and feature subsampling

Let $I = \{1, \ldots, n\}$ be the training samples indices. The randomization Π of the tree uses bootstrap: it samples uniformly at random, with replacement, elements of I corresponding to in-the-bag (`itb`) samples. If we denote as $I_{\texttt{itb}}$ the indices of unique `itb` samples, we can define the indices of out-of-bag (`oob`) samples as $I_{\texttt{oob}} = I \setminus I_{\texttt{itb}}$. A standard argument shows that $\mathbb{P}[i \in I_{\texttt{itb}}] = 1 - (1 - 1/n)^n \to 1 - e^{-1} \approx 0.632$ as $n \to +\infty$, known as the 0.632 rule [Efron and Tibshirani, 1997]. The randomization Π uses also feature subsampling: each time we need to find a split, we do not try all the features $\{1, \ldots, d\}$ but only a subset of them of size $d_{\max}$, chosen uniformly at random. This follows what standard RF algorithms do [Biau and Scornet, 2016; Breiman, 2001; Louppe, 2014], with the default $d_{\max} = \sqrt{d}$.

Split finding on histograms

For K-class classification, when looking for a split for some node $\mathbf{v}$, we compute the node's "histogram" $\text{hist}_{\mathbf{v}}[j, b, k] = \sum_{i \in I_{\texttt{itb}} : x_i \in C_{\mathbf{v}}} \mathbf{1} x_{i,j} = b, y_i = k$ for each sampled feature j, each bin b and label class k seen in the node's samples (actually weighted counts to handle bootstrapping and sample weights). Of course, one has $\text{hist}_{\mathbf{v}} = \text{hist}_{\mathbf{v}0} + \text{hist}_{\mathbf{v}1}$, so that we don't need to compute two histograms for siblings $\mathbf{v}0$ and $\mathbf{v}1$, but only a single one. Then, we loop over the set of non-constant (in the node) sampled features $\{j : \#\{b : \sum_k \text{hist}_{\mathbf{v}}[j, b, k] \geqslant 1\} \geqslant 2\}$ and over the set of non-empty bins $\{b : \sum_k \text{hist}_{\mathbf{v}}[j, b, k] \geqslant 1\}$ to find a split, by comparing standard impurity criteria computed on the histogram's statistics, such as gini or entropy for classification and variance for regression.

Bin order and categorical features

The order of the bins used in the loop depends on the type of the feature. If it is continuous, we use the natural order of bins. If it is categorical and the task is binary classification (labels in $\{0, 1\}$), we use the bin order that sorts $\text{hist}_{\mathbf{v}}[j, b, 1] / \sum_{k=0,1} \text{hist}_{\mathbf{v}}[j, b, k]$ with respect to b, namely the proportion of labels 1 in each bin. This allows to find the optimal split with complexity $O(b_j \log b_j)$, see Theorem 9.6 in Breiman et al. [1984], the logarithm coming from the sorting operation, while there are $2^{b_j - 1} - 1$ possible splits. This trick is used by EGB libraries as well, using an order of gradient/hessian statistics of the loss considered [Chen and Guestrin, 2016; Ke et al., 2017; Prokhorenkova et al., 2017].

For K-class classification with $K > 2$, we consider two strategies: (1) one-versus-rest, where we train MK trees instead of M, each tree trained with a binary one-versus-rest label, so that trees can find optimal categorical splits and (2) heuristic, where we train M trees and where split finding uses K loops over bin orders that sort $\text{hist}_{\mathbf{v}}[j, b, k] / \sum_{k'} \text{hist}_{\mathbf{v}}[j, b, k']$ (w.r.t b) for $k = 0, \ldots, K-1$. If a feature contains missing values, we do not loop only left to right (along bin order), but right to left as well, in order to compare splits that put missing values on the left or on the right.

Split requirements

Nodes must hold at least one `itb` and one `oob` sample to apply aggregation with exponential weights, see Section 4.2.2 below. A split is discarded if it leads to children with less than $n_{\text{min-leaf}}$ `itb` or `oob` samples and we do not split a node with less than $n_{\text{min-split}}$ `itb` or `oob` samples. These hyperparameters only weakly impact WW's performances and sticking to default values ($n_{\text{min-leaf}} = 1$ and $n_{\text{min-split}} = 2$, following `scikit-learn` [Louppe, 2014; Pedregosa et al., 2011]) is usually enough (see Appendix 4.7.4 from the supplementary material).

Related works on categorical splits

In Coppersmith et al. [1999], an interesting characterization of an optimal categorical split for multiclass classification is introduced, but no efficient algorithm is, to the best of our understanding, available for it. A heuristic algorithm is proposed therein, but it requires to computing, for each split, the top principal component of the covariance matrix of the conditional distribution of labels given bins, which is computationally too demanding for an RF algorithm intended for large datasets. Regularized target encoding is shown in Pargent et al. [2021] to perform best when compared with many alternative categorical encoding methods. Catboost [Prokhorenkova et al., 2017] uses target encoding, which replaces feature modalities by label statistics, so that a natural bin order can be used for split finding. To avoid overfitting on uninformative categorical features, a debiasing technique uses random permutations of samples and computes the target statistic of each element based only on its predecessors in the permutation. However, for multiclass classification, target encoding is influenced by the arbitrarily chosen ordinal encoding of the labels. LightGBM [Ke et al., 2017] uses a one-versus-rest strategy, which is also one of the approaches used in WW for categorical splits on multiclass tasks. For categorical splits, where bin order depends on labels statistics, WW does not use debiasing as in Prokhorenkova et al. [2017], since aggregation with exponential weights computed on `oob` samples allows to deal with overfitting.

Tree growth stopping. We do not split a node and make it a leaf if it contains less than $n_{\text{min-split}}$ `itb` or `oob` samples. The same applies when a node's impurity is not larger than a threshold ε ($\varepsilon = 0$ by default). When only leaves or non-splittable nodes remain, the growth of the tree is stopped. Trees grow in a depth-first fashion so that childs $\mathbf{v}0$ and $\mathbf{v}1$ have memory indexes larger than their parent $\mathbf{v}$ (as required by Algorithm 3 below).

4.2.2 Prediction function: aggregation with exponential weights

Given a tree $\mathcal{T}$ grown as described in Sections 4.2.1 and 4.2.1, its prediction function is an aggregation of the predictions given by all possible subtrees rooted at `root`, denoted $\{T : T \subset \mathcal{T}\}$. While $\mathcal{T}$ is grown using `itb` samples, we use `oob` samples to perform aggregation with exponential weights, with a branching process prior over subtrees, that gives more importance to subtrees with a good predictive `oob` performance.

Node and subtree prediction

We define $\mathbf{v}_T(x) \in \text{leaves}(T)$ as the leaf of T containing $x \in C$. The prediction of a node $\mathbf{v} \in \text{nodes}(\mathcal{T})$ and of a subtree $T \subset \mathcal{T}$ is given by

$$\widehat{y}_{\mathbf{v}} = h((y_i)_{i \in I_{\texttt{itb}} \,:\, x_i \in C_{\mathbf{v}}}) \quad \text{and} \quad \widehat{y}_T(x) = \widehat{y}_{\mathbf{v}_T(x)}, \tag{4.1}$$

where $h : \cup_{n \geqslant 0} \mathcal{Y}^n \to \widehat{\mathcal{Y}}$ is a generic "forecaster" used in each cell and where a subtree prediction is the one of its leaf containing x.

A standard choice for regression ($\mathcal{Y} = \widehat{\mathcal{Y}} = \mathbb{R}$) is the empirical mean forecaster

$$\widehat{y}_{\mathbf{v}} = \frac{1}{n_{\mathbf{v}}} \sum_{i \in I_{\texttt{itb}} \,:\, x_i \in C_{\mathbf{v}}} y_i, \tag{4.2}$$

where $n_{\mathbf{v}} = |\{i \in I_{\texttt{itb}} \,:\, x_i \in C_{\mathbf{v}}\}|$.

For K-class classification with $\mathcal{Y} = \{1, \ldots, K\}$ and $\widehat{\mathcal{Y}} = \mathcal{P}(\mathcal{Y})$, the set of probability distributions over $\mathcal{Y}$, a standard choice is a Bayes predictive posterior with a prior on $\mathcal{P}(\mathcal{Y})$ equal to the Dirichlet distribution $\mathsf{Dir}(\alpha, \ldots, \alpha)$, namely the *Jeffreys prior* on the multinomial model $\mathcal{P}(\mathcal{Y})$, which leads to

$$\widehat{y}_{\mathbf{v}}(k) = \frac{n_{\mathbf{v}}(k) + \alpha}{n_{\mathbf{v}} + \alpha K}, \tag{4.3}$$

for any $k \in \mathcal{Y}$, where $n_{\mathbf{v}}(k) = |\{i \in I_{\texttt{itb}} \,:\, x_i \in C_{\mathbf{v}}, y_i = k\}|$. By default, WW uses $\alpha = 1/2$ (the *Krichevsky-Trofimov* forecaster [Tjalkens et al., 1993]), but one can perfectly use any $\alpha > 0$, so that all the coordinates of $\widehat{y}_{\mathbf{v}}$ are positive. This is motivated by the fact that WW uses as default the log loss to assess `oob` performance for classification, which requires an arbitrarily chosen clipping value for zero probabilities. Different choices of α only weakly impact WW's performance, as illustrated in Appendix 4.7.4.

We use `oob` samples to define the cumulative losses of the predictions of all $T \subset \mathcal{T}$

$$L_T = \sum_{i \in I_{\texttt{oob}}} \ell(\widehat{y}_T(x_i), y_i), \tag{4.4}$$

where $\ell : \widehat{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}^+$ is a loss function. For regression problems, a default choice is the quadratic loss $\ell(\widehat{y}, y) = (\widehat{y} - y)^2$ while for multiclass classification, a default is the log-loss $\ell(\widehat{y}, y) = -\log \widehat{y}(y)$, where $\widehat{y}(y) \in (0, 1]$ when using (4.3), but other loss choices are of course possible.

Prediction function

Let $x \in C$. The prediction function $\widehat{f}$ of a tree $\mathcal{T}$ in WW is given by

$$\widehat{f}(x) = \frac{\sum_{T \subset \mathcal{T}} \pi(T) e^{-\eta L_T} \widehat{y}_T(x)}{\sum_{T \subset \mathcal{T}} \pi(T) e^{-\eta L_T}} \quad \text{with} \quad \pi(T) = 2^{-\|T\|}, \tag{4.5}$$

where the sum is over all subtrees T of $\mathcal{T}$ rooted at `root`, where $\eta > 0$ is temperature parameter and $\|T\|$ is the number of nodes in T minus its number of leaves that are also leaves of $\mathcal{T}$. Note that π is the distribution of the branching process with branching probability $1/2$ at each node of $\mathcal{T}$, with exactly two children when it branches. A default choice is $\eta = 1$ for the log-loss (see in particular Corollary 4.3 in Section 4.3 below), but it can also be tuned through hyperoptimization, although

we do not observe strong performance gains, see Appendix 4.7.4 from the supplementary material. The prediction function (4.5) is an aggregation of the predictions $\widehat{y}_T(x)$ of all subtrees T, weighted by their performance on `oob` samples and with prior $\pi(T) = 2^{-\|T\|}$. This aggregation procedure can be understood as a *non-greedy way to prune trees*: the weights depend not only on the quality of one single split but also on the performance of each subsequent split.

Computing $\widehat{f}$ from Equation (4.5) is computationally and memory-wise infeasible for a large $\mathcal{T}$, since it involves a sum over all $T \subset \mathcal{T}$ and requires one weight for each T. Indeed, the number of subtrees of a minimal tree that separates n points is exponential in the number of nodes, and hence *exponential in* n. However, it turns out that one can compute exactly and very efficiently $\widehat{f}$ thanks to the prior choice π together with an adaptation of *context tree weighting* [Catoni, 2004; Helmbold and Schapire, 1997; Willems, 1998; Willems et al., 1995].

Theorem 4.1. *The prediction function* (4.5) *can be written as* $\widehat{f}(x) = \widehat{f}_{\texttt{root}}(x)$, *where* $\widehat{f}_{\texttt{root}}(x)$ *satisfies the recursion*

$$\widehat{f}_{\mathbf{v}}(x) = \frac{1}{2}\frac{w_{\mathbf{v}}}{w^{\text{den}}_{\mathbf{v}}}\widehat{y}_{\mathbf{v}} + \Big(1 - \frac{1}{2}\frac{w_{\mathbf{v}}}{w^{\text{den}}_{\mathbf{v}}}\Big)\widehat{f}_{\mathbf{v}a}(x) \tag{4.6}$$

for $\mathbf{v}, \mathbf{v}a \in \texttt{path}(x)$ $(a \in \{0,1\})$ *the path in* $\mathcal{T}$ *going from* `root` *to* $\mathbf{v}_{\mathcal{T}}(x)$, *where* $w_{\mathbf{v}} := \exp(-\eta L_{\mathbf{v}})$ *with* $L_{\mathbf{v}} := \sum_{i \in I_{\texttt{oob}} : x_i \in C_{\mathbf{v}}} \ell(\widehat{y}_{\mathbf{v}}, y_i)$ *and where* $w^{\text{den}}_{\mathbf{v}}$ *are weights satisfying the recursion*

$$w^{\text{den}}_{\mathbf{v}} = \begin{cases} w_{\mathbf{v}} & \text{if } \mathbf{v} \in \text{leaves}(\mathcal{T}), \\ \frac{1}{2}w_{\mathbf{v}} + \frac{1}{2}w^{\text{den}}_{\mathbf{v}0}w^{\text{den}}_{\mathbf{v}1} & \text{otherwise.} \end{cases} \tag{4.7}$$

The proof of Theorem 4.1 is given in Appendix 4.6.1 of the supplementary material, a consequence of this Theorem being a very efficient computation of $\widehat{f}(x)$ is described in Algorithms 3 and 4 below. Algorithm 3 computes the weights $w^{\text{den}}_{\mathbf{v}}$ using the fact that trees in WW are grown in a depth-first fashion, so that we can loop *once*, leading to a $O(|\text{nodes}(\mathcal{T})|)$ complexity in time and in memory usage, over nodes from a data structure that respects the parenthood order. Direct computations can lead to numerical over- or under-flows (many products of exponentially small or large numbers are involved), so Algorithm 3 works recursively over the logarithms of the weights (line 6 uses a log-sum-exp function that can be made overflow-proof). Algorithm 3 is applied once $\mathcal{T}$ is fully grown, so that WW is ready

Algorithm 3 Computation of $\log(w^{\text{den}}_{\mathbf{v}})$ for all $\mathbf{v} \in \text{nodes}(\mathcal{T})$.

1: **Inputs:** $\mathcal{T}$, $\eta > 0$ and losses $L_{\mathbf{v}}$ for all $\mathbf{v} \in \text{nodes}(\mathcal{T})$. Nodes from $\text{nodes}(\mathcal{T})$ are stored in a data structure `nodes` that respects parenthood order: for any $\mathbf{v} = \texttt{nodes}[i_{\mathbf{v}}] \in \text{intnodes}(\mathcal{T})$ and children $\mathbf{v}a = \texttt{nodes}[i_{\mathbf{v}a}]$ for $a \in \{0,1\}$, we have $i_{\mathbf{v}a} > i_{\mathbf{v}}$.
2: **for** $\mathbf{v} \in \text{reversed}(\texttt{nodes})$ **do**
3: **if** $\mathbf{v}$ is a leaf **then**
4: Put $\log(w^{\text{den}}_{\mathbf{v}}) \leftarrow -\eta L_{\mathbf{v}}$
5: **else**
6: Put $\log(w^{\text{den}}_{\mathbf{v}}) \leftarrow \log(\frac{1}{2}e^{-\eta L_{\mathbf{v}}} + \frac{1}{2}e^{\log(w^{\text{den}}_{\mathbf{v}0}) + \log(w^{\text{den}}_{\mathbf{v}1})})$
7: **end if**
8: **end for**
9: **return** The set of log-weights $\{\log(w^{\text{den}}_{\mathbf{v}}) : \mathbf{v} \in \text{nodes}(\mathcal{T})\}$

to produce predictions using Algorithm 4 below. Note that hyperoptimization of η

or α, if required, does not need to grow $\mathcal{T}$ again, but only to update $w_{\mathbf{v}}^{\text{den}}$ for all $\mathbf{v} \in \text{nodes}(\mathcal{T})$ with Algorithm 3, making hyperoptimization of these parameters particularly efficient. The recursion used in Algorithm 4 has a complexity $O(|\texttt{path}(x)|)$

Algorithm 4 Computation of $\widehat{f}(x)$ for any $x \in C$.

Inputs: Tree $\mathcal{T}$, losses $L_{\mathbf{v}}$ and log-weights $\log(w_{\mathbf{v}}^{\text{den}})$ computed by Algorithm 3
Find $\mathbf{v}_{\mathcal{T}}(x) \in \text{leaves}(\mathcal{T})$ (the leaf containing x) and put $\mathbf{v} \leftarrow \mathbf{v}_{\mathcal{T}}(x)$
Put $\widehat{f}(x) \leftarrow \widehat{y}_{\mathbf{v}}$ (the node $\mathbf{v}$ forecaster, such as (4.2) for regression or (4.3) for classification)
while $\mathbf{v} \neq \texttt{root}$ **do**
    Put $\mathbf{v} \leftarrow \text{parent}(\mathbf{v})$
    Put $\alpha \leftarrow \frac{1}{2}\exp(-\eta L_{\mathbf{v}} - \log(w_{\mathbf{v}}^{\text{den}}))$
    Put $\widehat{f}(x) \leftarrow \alpha \widehat{y}_{\mathbf{v}} + (1-\alpha)\widehat{f}(x)$
end while
return The prediction $\widehat{f}(x)$

which is the complexity required to find the leaf $\mathbf{v}_{\mathcal{T}}(x)$ containing $x \in C$: Algorithm 4 *only increases by a factor* 2 the prediction complexity of a standard RF (in order to go down to $\mathbf{v}_{\mathcal{T}}(x)$ and up again to $\texttt{root}$ along $\texttt{path}(x)$). More details about the construction of Algorithms 3 and 4 can be found in Appendix 4.6.1 of the supplementary material.

4.3 Theoretical guarantees

This section proposes some theoretical guarantees on the subtrees aggregation used in WW, see (4.5). We say that a loss function ℓ is η-exp-concave for some $\eta > 0$ whenever $z \mapsto \exp(-\eta \ell(z, y))$ is concave for any $y \in \mathcal{Y}$. We consider a fully-grown tree $\mathcal{T}$ computed using $\texttt{itb}$ samples and the set of $\texttt{oob}$ samples $(x_i, y_i)_{i \in I_{\texttt{oob}}}$ on which L_T is computed using (4.4), and we denote $n_{\texttt{oob}} := |I_{\texttt{oob}}|$.

Theorem 4.2 (Oracle inequality)**.** *Assume that the loss function ℓ is η-exp-concave. Then, the prediction function $\widehat{f}$ given by (4.5) satisfies the oracle inequality*

$$\frac{1}{n_{\texttt{oob}}} \sum_{i \in I_{\texttt{oob}}} \ell(\widehat{f}(x_i), y_i) \leqslant \inf_{T \subset \mathcal{T}} \left\{ \frac{1}{n_{\texttt{oob}}} \sum_{i \in I_{\texttt{oob}}} \ell(\widehat{y}_T(x_i), y_i) + \frac{\log 2}{\eta} \frac{\|T\|}{n_{\texttt{oob}} + 1} \right\},$$

where the infimum is over any subtree $T \subset \mathcal{T}$ and where we recall that $\|T\|$ is the number of nodes in T minus its number of leaves that are also leaves of $\mathcal{T}$.

Theorem 4.2 proves that for a general loss function, the prediction function of WW is able to perform nearly as well as the best *oracle* subtree $T \subset \mathcal{T}$ on $\texttt{oob}$ samples, with an $O(\|T\|/n_{\texttt{oob}})$ rate which is optimal for model-selection oracle inequalities [Tsybakov, 2003] ($\|T\| = O(\log N_{\mathcal{T}})$ with a number of "experts" $N_{\mathcal{T}} = |\{T : T \subset \mathcal{T}\}|$ for a well-balanced $\mathcal{T}$). Let us stress again that, while finding an oracle $\text{argmin}_{T \subset \mathcal{T}} \sum_{i \in I_{\texttt{oob}}} \ell(\widehat{y}_T(x_i), y_i)$ is computationally infeasible, since it requires trying out all possible subtrees, WW's prediction function (4.5) comes at a cost comparable to that of a standard Random Forest, as explained in Section 4.2.2 above.

The proof of Theorem 4.2 is given in Section 4.6.2 of the supplementary material, and relies on techniques from PAC-Baysesian theory [Catoni, 2007; McAllester, 1998, 1999]. Compared with Mourtada et al. [2021] about online learning, our

proof differs significantly: we do not use results specialized to online learning such as Vovk [1998] nor online-to-batch conversion Cesa-Bianchi et al. [2004]. Note that Theorem 4.2 does not address the generalization error, since it would require to study the generalization error of the random forest itself (and of the fully grown tree $\mathcal{T}$), which is a topic way beyond the scope of this paper, and still a very difficult open problem: recent results [Arlot and Genuer, 2014; Genuer, 2012; Mourtada et al., 2020; Scornet, 2016a,b; Scornet et al., 2015] only study stylized versions of RF (called purely random forests).

Consequences of Theorem 4.2 are Corollary 4.3 for the log-loss (classification) and Corollary 4.4 for the least-squares loss (regression).

Corollary 4.3 (Classification)**.** *Consider K-class classification ($\mathcal{Y} = \{1, \dots, K\}$) and consider the prediction function $\widehat{f}$ given by (4.5), where node predictions are given by (4.3) with $\alpha = 1/2$ (WW's default), where ℓ is the log-loss and where $\eta = 1$. Then, we have*

$$\frac{1}{n_{\texttt{oob}}} \sum_{i \in I_{\texttt{oob}}} \ell(\widehat{f}(x_i), y_i) \leqslant \inf_{T \subset \mathcal{T}} \left\{ \frac{1}{n_{\texttt{oob}}} \sum_{i \in I_{\texttt{oob}}} \ell(g_T(x_i), y_i) + \frac{K + 4\log 2 - 1}{4} \frac{\|T\| + 1}{n_{\texttt{oob}}} \right\},$$

where g_T is any constant function on the leaves of T.

Corollary 4.4 (Regression)**.** *Consider regression with $\mathcal{Y} = [-B, B]$ for some $B > 0$ and the prediction function $\widehat{f}$ given by (4.5), where node predictions are given by (4.2), where ℓ is the least-squares loss and where $\eta = 1/(8B^2)$. Then, we have*

$$\frac{1}{n_{\texttt{oob}}} \sum_{i \in I_{\texttt{oob}}} \ell(\widehat{f}(x_i), y_i) \leqslant \inf_{T \subset \mathcal{T}} \left\{ \frac{1}{n_{\texttt{oob}}} \sum_{i \in I_{\texttt{oob}}} \ell(g_T(x_i), y_i) + 8(\log 2) B^2 \frac{\|T\|}{n_{\texttt{oob}}} \right\},$$

where g_T is any function constant on the leaves of T.

The proofs of Corollaries 4.3 and 4.4 are given in Section 4.6.2 of the supplementary material. These corollaries motivate the default hyperparameter values of η, in particular $\eta = 1$ for classification.

4.4 Experiments

Our implementation of WildWood is available at the `GitHub` repository `https://github.com/pyensemble/wildwood`. It is open-sourced under the `BSD 3-Clause` license on `GitHub`. It is a `Python` package that follows `scikit-learn`'s API conventions, that is JIT-compiled to machine code using `numba` [Lam et al., 2015]. Trees in the forest are grown in parallel using `joblib` [Joblib Development Team, 2020] and CPU threads, GPU training will be supported in future updates.

We compare WildWood (denoted `WW`n for n trees) with several strong baselines including `RF`n: `scikit-learn`'s implementation of Random Forest [Louppe, 2014; Pedregosa et al., 2011] using n trees; `HGB`: an histogram-based implementation of extreme gradient boosting (inspired by LightGBM) from `scikit-learn`; and several state-of-the-art and widely adopted extreme gradient boosting libraries including `XGB`: XGBoost [Chen and Guestrin, 2016]; `LGBM`: LightGBM [Ke et al., 2017] and `CB`: CatBoost [Dorogush et al., 2018; Prokhorenkova et al., 2017]. We used a 32-cores server with two Intel Xeon Gold CPUs, two Tesla V100 GPUs and 384GB RAM for the experiments involving hyperoptimization (Table 4.1) and used a 12-cores Intel i7 MacBook Pro with 32GB RAM and no GPU to obtain training times achievable by a "standard user" (Table 4.2). All experiments can be reproduced using `Python` scripts on the repository.

4.4.1 Description of the experiments

We use publicly available and open-source datasets from the UCI repository [Dua and Graff, 2017], including small datasets (hundreds of rows) and large datasets (millions of rows), their main characteristics are given in Table 4.5 together with URLs in Table 4.6, see the supplementary material. Each dataset is randomly split into a training set (70%) and a test set (30%). We specify which features are categorical to algorithms that natively support it (`HGB`, `LGBM`, `CB` and `WW`n) and simply integer-encode them, while we use one-hot encoding for other algorithms (`RF`n, `XGB`).

For each algorithm and dataset, hyperoptimization is performed as follows: from the training set, we use 4/5 for training and 1/5 for validation and do 50 steps of sequential optimization using the Tree Parzen Estimator implemented in the `hyperopt` library [Bergstra et al., 2015]. More details about hyperoptimization are provided in Appendix 4.7.2 of the supplementary material. Then, we refit on the whole training set with the best hyperparameters and report scores on the test set. This is performed 5 times in order to report standard deviations.

We use the area under the ROC curve (AUC), for K-class datasets with $K > 2$ we average the AUC of each class versus the rest. This leads to the test AUC scores displayed in Table 4.1 (the same scores with standard deviations are available in Table 4.3 of the supplementary material). We report also in Table 4.2 (see Table 4.4 from supplementary material for standard deviations) the test AUC scores obtained with default hyperparameters of all algorithms on the 5 largest considered datasets together with their training times (timings can vary by several orders of magnitude with varying hyperparameters for EGB libraries, as observed by the timing differences between Figure 4.3 and Table 4.2).

Table 4.1 – Test AUC of all algorithms after hyperoptimization on the considered datasets. Standard-deviations are reported in Table 4.3 from the supplementary material. We observe that WW has better (or identical in some cases) performances than RF on all datasets and that it is close to that of EGB libraries (bold is for best EGB performance, underline for best `RF`n or `WW`n performance).

	XGB	LGBM	CB	HGB	RF10	RF100	WW10	WW100
adult	0.930	**0.931**	0.927	0.930	0.916	0.919	0.918	0.919
bank	0.933	**0.935**	0.925	0.930	0.917	0.929	0.924	0.931
breastcancer	0.991	0.993	0.987	**0.994**	0.974	0.978	0.992	0.992
car	0.999	**1.000**	**1.000**	**1.000**	0.996	0.996	0.997	0.998
covtype	**0.999**	**0.999**	0.998	**0.999**	0.996	0.998	0.996	0.998
default-cb	0.780	**0.783**	0.780	0.779	0.748	0.774	0.773	0.778
higgs	0.853	**0.857**	0.847	0.853	0.812	0.834	0.818	0.835
internet	0.934	0.910	**0.938**	0.911	0.841	0.911	0.923	0.928
kddcup	**1.000**	**1.000**	**1.000**	**1.000**	0.997	0.998	1.000	1.000
kick	**0.777**	0.770	**0.777**	0.771	0.736	0.752	0.756	0.763
letter	**1.000**	**1.000**	**1.000**	**1.000**	0.997	0.999	0.996	0.999
satimage	**0.991**	**0.991**	**0.991**	0.987	0.980	0.989	0.983	0.991
sensorless	**1.000**	**1.000**	**1.000**	**1.000**	1.000	1.000	1.000	1.000
spambase	**0.990**	**0.990**	0.987	0.986	0.980	0.986	0.983	0.987

4.4.2 Discussion of experiments results

We observe in Table 4.1 that EGB algorithms, when hyperoptimized, lead to the best performances over the considered datasets compared with RF algorithms, and we observe that WW always improves the performance of RF, at the exception of few datasets for which the performance is identical. When using default hyperparameters for all algorithms, we observe in Table 4.2 that the test AUC scores can decrease significantly for EGB libraries while RF algorithms seem more stable, and that there is no clear best performing algorithm in this case.

The results on both tables show that WW is competitive with respect to all baselines both in terms of performance and computational times: it manages to always reach at least comparable performance with the best algorithms despite only using 10 trees as a default. In this respect, WW maintains high scores at a lower computational cost.

Table 4.2 – Training times (seconds) of all algorithms with their default hyperparameters (no hyperoptimization) on the 5 largest considered datasets and test AUC corresponding to these training times. Test AUC scores are worse than that of Table 4.1, since no hyperoptimization is used. WW, which uses only 10 trees here (default number of trees), is almost always the fastest algorithm, for performances comparable to that of all baselines (bold is for best EGB training time or performance, underline for best `RF` or `WW` training time or performance). Standard deviations are reported in Table 4.4 of the supplementary material.

	Training time (seconds)						Test AUC					
	`XGB`	`LGBM`	`CB`	`HGB`	`RF`	`WW`	`XGB`	`LGBM`	`CB`	`HGB`	`RF`	`WW`
covtype	10	**3**	120	14	21	3	0.986	0.978	**0.989**	0.960	0.998	0.979
higgs	36	**30**	653	85	1389	179	0.823	0.812	**0.840**	0.812	0.838	0.813
internet	9	**4**	188	8	0.4	0.3	**0.918**	0.828	0.910	0.500	0.862	0.889
kddcup	175	41	2193	**31**	208	12	**1.000**	0.638	0.988	0.740	0.998	1.000
kick	7	**0.4**	50	0.7	31	5	0.768	0.757	**0.781**	0.773	0.747	0.751

4.5 Conclusion

We introduced WildWood, a new Random Forest algorithm for batch supervised learning. Tree predictions in WildWood are aggregation with exponential weights of the predictions of all subtrees, with weights computed on bootstrap out-of-the-bag samples. This leads to improved predictions in each individual tree, at a small computational cost, since WildWood's prediction complexity is similar to that of a standard Random Forest. Moreover, thanks to the histogram strategy, WildWood's implementation is competitive with strong baselines including popular extreme boosting libraries, both in terms of performance and training times. Note also that WildWood has few hyperparameters to tune and that the performances obtained with default hyperparameters are usually good enough in our experiments.

WildWood's implementation is still evolving and many improvements coming with future updates are planned, including the computation of feature importance, GPU training, distributed training (we only support single-machine training for now), among other enhancements that will further improve performances and accelerate computations. Room for improvement in WildWood comes from the fact that the overall forest prediction is a simple arithmetic mean of each tree prediction,

while we could perform also exponentially weighted aggregation between trees. Future works include a WildWood-based implementation of isolation-forest [Liu et al., 2008], using the same subtrees aggregation mechanism with the log loss for density estimation, to propose a new algorithm for outliers detection.

4.6 Appendix: technical proofs

4.6.1 Proof of Theorem 4.1 and construction of Algorithms 3 and 4

The expression in Equation (4.5) involves sums over all subtrees T of the fully grown tree $\mathcal{T}$ (involving an exponential in the number of leaves of $\mathcal{T}$). However, it can be computed efficiently because of the specific choice of the prior π. More precisely, we will use the following lemma [Helmbold and Schapire, 1997, Lemma 1] several times to efficiently compute sums of products. Let us recall that $\text{nodes}(\mathcal{T})$ stands for the set of nodes of $\mathcal{T}$.

Lemma 4.5. *Let $g : \text{nodes}(\mathcal{T}) \to \mathbb{R}$ be an arbitrary function and define $G : \text{nodes}(\mathcal{T}) \to \mathbb{R}$ as*

$$G(\mathbf{v}) = \sum_{T \subset \mathcal{T}_\mathbf{v}} 2^{-\|T\|} \prod_{\mathbf{v}' \in \text{leaves}(T)} g(\mathbf{v}'), \tag{4.8}$$

where the sum over $T \subset \mathcal{T}_\mathbf{v}$ means the sum over all subtrees T of $\mathcal{T}$ rooted at $\mathbf{v}$. Then, $G(\mathbf{v})$ can be computed recursively as follows:

$$G(\mathbf{v}) = \begin{cases} g(\mathbf{v}) & \text{if } \mathbf{v} \in \text{leaves}(\mathcal{T}) \\ \frac{1}{2} g(\mathbf{v}) + \frac{1}{2} G(\mathbf{v}0) G(\mathbf{v}1) & \text{otherwise,} \end{cases}$$

for each node $\mathbf{v} \in \text{nodes}(\mathcal{T})$.

For the sake of completeness, we include a proof of this statement.

Proof. First, let us notice that the case $\mathbf{v} \in \text{leaves}(\mathcal{T})$ is straightforward since there is only one pruning T of $\mathcal{T}_\mathbf{v}$ which satisfies $\|T\| = 0$ (recall that $\|T\|$ is the number of internal nodes and leaves in T *minus* the number of leaves in T that are also leaves of $\mathcal{T}_\mathbf{v}$). For the second case, we can expand $G(\mathbf{v})$ by taking into account the pruning which only leaves $\mathbf{v}$ as a leaf, the rest of the prunings can be expressed through pairs of prunings T_0 and T_1 of $\mathcal{T}_{\mathbf{v}0}$ and $\mathcal{T}_{\mathbf{v}1}$ respectively. Moreover, it can be shown that such a pruning T satisfies $\|T\| = 1 + \|T_0\| + \|T_1\|$, thus we get the following expansion

$$\begin{aligned} G(\mathbf{v}) &= \frac{1}{2} g(\mathbf{v}) + \sum_{T_0 \subset \mathcal{T}_{\mathbf{v}0}} \sum_{T_1 \subset \mathcal{T}_{\mathbf{v}1}} 2^{-(1+\|T_0\|+\|T_1\|)} \prod_{\mathbf{v}' \in T_0} g(\mathbf{v}0\mathbf{v}') \prod_{\mathbf{v}'' \in T_1} g(\mathbf{v}1\mathbf{v}'') \\ &= \frac{1}{2} g(\mathbf{v}) + \frac{1}{2} \Big(\sum_{T_0 \subset \mathcal{T}_{\mathbf{v}0}} 2^{-\|T_0\|} \prod_{\mathbf{v}' \in T_0} g(\mathbf{v}0\mathbf{v}') \Big) \cdot \Big(\sum_{T_1 \subset \mathcal{T}_{\mathbf{v}1}} 2^{-\|T_1\|} \prod_{\mathbf{v}'' \in T_1} g(\mathbf{v}1\mathbf{v}'') \Big) \\ &= \frac{1}{2} g(\mathbf{v}) + \frac{1}{2} G(\mathbf{v}0) G(\mathbf{v}1). \end{aligned}$$

This concludes the proof of Lemma 4.5. □

Let us introduce $w_T = \pi(T) \exp(-\eta L_T)$ for any $T \subset \mathcal{T}$, so that Equation (4.5) writes

$$\widehat{f}(x) = \frac{\sum_{T \subset \mathcal{T}} w_T \widehat{y}_T(x)}{\sum_{T \subset \mathcal{T}} w_T}, \tag{4.9}$$

where the sums hold over all the subtrees T of $\mathcal{T}$ rooted at `root` (the root of the full tree $\mathcal{T}$). We will show how to efficiently compute and update the numerator and denominator in Equation (4.9). Note that w_T may be written as

$$\begin{aligned}
w_T &= \pi(T)\exp(-\eta L_T) \\
&= 2^{-\|T\|}\exp\Big(-\eta \sum_{i\in I_{\texttt{oob}}} \ell(\widehat{y}_{\mathbf{v}_T(x_i)}, y_i)\Big) \\
&= 2^{-\|T\|}\exp\Big(-\eta \sum_{\mathbf{v}\in\text{leaves}(T)} \sum_{i\in I_{\texttt{oob}}\,:\,x_i\in C_{\mathbf{v}}} \ell(\widehat{y}_{\mathbf{v}_T(x_i)}, y_i)\Big) \qquad (4.10)\\
&= 2^{-\|T\|}\exp\Big(-\eta \sum_{\mathbf{v}\in\text{leaves}(T)} \sum_{i\in I_{\texttt{oob}}\,:\,x_i\in C_{\mathbf{v}}} \ell(\widehat{y}_{\mathbf{v}}, y_i)\Big) \qquad (4.11)\\
&= 2^{-\|T\|}\exp\Big(-\eta \sum_{\mathbf{v}\in\text{leaves}(T)} L_{\mathbf{v}}\Big) \\
&= 2^{-\|T\|} \prod_{\mathbf{v}\in\text{leaves}(T)} w_{\mathbf{v}}, \qquad (4.12)
\end{aligned}$$

where we recall that

$$L_{\mathbf{v}} = \sum_{i\in I_{\texttt{oob}}\,:\,x_i\in C_{\mathbf{v}}} \ell(\widehat{y}_{\mathbf{v}}, y_i) \quad \text{and} \quad w_{\mathbf{v}} = \exp(-\eta L_{\mathbf{v}}).$$

Equality (4.10) comes from the fact that the set of cells $\{C_{\mathbf{v}} : \mathbf{v} \in \text{leaves}(T)\}$ is a partition of C by construction, and that the stopping criterion used to build $\mathcal{T}$ ensures that each leaf node in leaves(T) contains at least one sample from $I_{\texttt{oob}}$ (see Section 4.2.1). Equality (4.11) comes from the fact that the prediction of a node is constant and equal to $\widehat{y}_{\mathbf{v}}$ for any $x \in C_{\mathbf{v}}$.

Denominator of Equation (4.9). For each node $\mathbf{v} \in \text{nodes}(\mathcal{T})$, denote

$$w_{\mathbf{v}}^{\text{den}} = \sum_{T\subset\mathcal{T}_{\mathbf{v}}} 2^{-\|T\|} \prod_{\mathbf{v}'\in\text{leaves}(T)} w_{\mathbf{v}'}, \qquad (4.13)$$

where once again the sum over $T \subset \mathcal{T}_{\mathbf{v}}$ means the sum over all subtrees T of $\mathcal{T}$ rooted at $\mathbf{v}$. We have that (4.12) entails

$$w_{\texttt{root}}^{\text{den}} = \sum_{T\subset\mathcal{T}_{\texttt{root}}} 2^{-\|T\|} \prod_{\mathbf{v}\in\text{leaves}(T)} w_{\mathbf{v}} = \sum_{T\subset\mathcal{T}_{\texttt{root}}} w_T = \sum_{T\subset\mathcal{T}} w_T\,. \qquad (4.14)$$

So, we can compute recursively $w_{\texttt{root}}^{\text{den}}$ very efficiently, using a recursion on the weights $w_{\mathbf{v}}^{\text{den}}$ using Lemma 4.5 with $g(\mathbf{v}) = w_{\mathbf{v}}$. This leads to the recursion stated in Theorem 4.1, see Equation (4.7).

Now, we can exploit the fact that decision trees are built in a depth-first fashion in WildWood: all the nodes $\mathbf{v} \in \mathcal{T}$ are stored in a "flat" array, and by construction both the child nodes $\mathbf{v}0$ and $\mathbf{v}1$ have indexes that are larger than the one of $\mathbf{v}$. So, we can simply loop over the array of nodes in reverse order, and compute $w_{\mathbf{v}}^{\text{den}} = w_{\mathbf{v}}$ if $\mathbf{v} \in \text{leaves}(\mathcal{T})$ and $w_{\mathbf{v}}^{\text{den}} = \frac{1}{2}w_{\mathbf{v}} + \frac{1}{2}w_{\mathbf{v}0}^{\text{den}}w_{\mathbf{v}1}^{\text{den}}$ otherwise: we are guaranteed to have computed $w_{\mathbf{v}0}^{\text{den}}$ and $w_{\mathbf{v}1}^{\text{den}}$ before computing $w_{\mathbf{v}}^{\text{den}}$. This algorithm is described in Algorithm 3. Since these computations involve a large number of products with exponentiated numbers, it typically leads to strong over- and

under-flows: we describe in Algorithm 3 a version of this algorithm which works recursively over the logarithms of the weights. At the end of this loop, we end up at $\mathbf{v} = \texttt{root}$ and have computed $w^{\text{den}}_{\texttt{root}} = \sum_{T \subset \mathcal{T}} w_T$ with a very efficient $O(|\text{nodes}(\mathcal{T})|)$ complexity. Note also that it is sufficient to store both $w_{\mathbf{v}}$ and $w^{\text{den}}_{\mathbf{v}}$ for all $\mathbf{v} \in \mathcal{T}$, which makes for a $O(|\text{nodes}(\mathcal{T})|)$ memory consumption.

Numerator of Equation (4.9). The numerator of Equation (4.9) almost follows the exact same argument as the denominator, but since it depends on the input vector $x \in C$ of features for which we want to produce a prediction, it is performed at inference time. Recall that $\texttt{path}(x)$ is the sequence of nodes that leads to the leaf $\mathbf{v}_{\mathcal{T}}(x)$ containing $x \in C$ and define, for any $\mathbf{v} \in \text{nodes}(\mathcal{T})$, $\widehat{w}_{\mathbf{v}}(x) = w_{\mathbf{v}}\widehat{y}_{\mathbf{v}}(x)$ if $\mathbf{v} \in \texttt{path}(x)$, and $\widehat{w}_{\mathbf{v}}(x) = w_{\mathbf{v}}$ otherwise. We have

$$\begin{aligned}
\sum_{T \subset \mathcal{T}} w_T \widehat{y}_T(x) &= \sum_{T \subset \mathcal{T}_{\texttt{root}}} w_T \widehat{y}_{\mathbf{v}_T(x)} \\
&= \sum_{T \subset \mathcal{T}_{\texttt{root}}} 2^{-\|T\|} \prod_{\mathbf{v} \in \text{leaves}(T)} w_{\mathbf{v}} \widehat{y}_{\mathbf{v}_T(x)} \qquad (4.15) \\
&= \sum_{T \subset \mathcal{T}_{\texttt{root}}} 2^{-\|T\|} \prod_{\mathbf{v} \in \text{leaves}(T)} \widehat{w}_{\mathbf{v}}(x). \qquad (4.16)
\end{aligned}$$

Note that (4.15) comes from (4.12) while (4.16) comes from the definition of $\widehat{w}_{\mathbf{v}}(x)$ (note that a single term from the product over $\mathbf{v} \in \text{leaves}(T)$ corresponds to $\mathbf{v} = \mathbf{v}_T(x)$ since $\{C_{\mathbf{v}} : \mathbf{v} \in \text{leaves}(T)\}$ is a partition of C). We are now in position to use again Lemma 4.5 with $g(\mathbf{v}) = \widehat{w}_{\mathbf{v}}(x)$. Defining

$$w^{\text{num}}_{\mathbf{v}}(x) = \sum_{T \subset \mathcal{T}_{\mathbf{v}}} 2^{-\|T\|} \prod_{\mathbf{v}' \in \text{leaves}(T)} \widehat{w}_{\mathbf{v}'}(x)\,,$$

we can conclude that

$$w^{\text{num}}_{\texttt{root}}(x) = \sum_{T \subset \mathcal{T}} w_T \widehat{y}_T(x) \qquad (4.17)$$

and that the following recurrence holds:

$$w^{\text{num}}_{\mathbf{v}}(x) = \begin{cases} \widehat{w}_{\mathbf{v}}(x) & \text{if } \mathbf{v} \in \text{leaves}(\mathcal{T}) \\ \frac{1}{2}\widehat{w}_{\mathbf{v}}(x) + \frac{1}{2} w^{\text{num}}_{\mathbf{v}0}(x) w^{\text{num}}_{\mathbf{v}1}(x) & \text{otherwise.} \end{cases} \qquad (4.18)$$

This recurrence allows to compute $w^{\text{num}}_{\mathbf{v}}(x)$ from $\widehat{w}_{\mathbf{v}}(x)$, but note that a direct use of this formula would lead to a complexity $O(|\text{nodes}(\mathcal{T})|)$ to produce a prediction for a single input $x \in C$. It turns out can we can do much better than that.

Indeed, whenever $\mathbf{v} \notin \texttt{path}(x)$, we have by definition that $\widehat{w}_{\mathbf{v}}(x) = w_{\mathbf{v}}$ and that $\widehat{w}_{\mathbf{v}'}(x) = w_{\mathbf{v}'}$ for any descendant $\mathbf{v}'$ of $\mathbf{v}$, which entails by induction that $w^{\text{num}}_{\mathbf{v}}(x) = w^{\text{den}}_{\mathbf{v}}$ for any $\mathbf{v} \notin \texttt{path}(x)$. Therefore, we only need to explain how to compute $w^{\text{num}}_{\mathbf{v}}(x)$ for $\mathbf{v} \in \texttt{path}(x)$. This is achieved recursively, thanks to (4.18), starting at the leaf $\mathbf{v}_{\mathcal{T}}(x)$ and going up in the tree to $\texttt{root}$:

$$w^{\text{num}}_{\mathbf{v}}(x) = \begin{cases} w_{\mathbf{v}}\widehat{y}_{\mathbf{v}} & \text{if } \mathbf{v} = \mathbf{v}_{\mathcal{T}}(x) \\ \frac{1}{2} w_{\mathbf{v}}\widehat{y}_{\mathbf{v}} + \frac{1}{2} w^{\text{den}}_{\mathbf{v}(1-a)} w^{\text{num}}_{\mathbf{v}a}(x) & \text{otherwise, where } a \in \{0,1\} \text{ is s.t. } \mathbf{v}a \in \texttt{path}(x). \end{cases} \qquad (4.19)$$

Let us explain where this comes from: firstly, one has obviously that $\text{leaves}(\mathcal{T}) \cap \texttt{path}(x) = \{\mathbf{v}_{\mathcal{T}}(x)\}$, so that $w^{\text{num}}_{\mathbf{v}}(x) = g(\mathbf{v}) = \widehat{w}_{\mathbf{v}}(x) = w_{\mathbf{v}}\widehat{y}_{\mathbf{v}}(x)$ for $\mathbf{v} = \mathbf{v}_{\mathcal{T}}(x)$.

Secondly, we go up in the tree along $\texttt{path}(x)$ and use again (4.18): whenever $\mathbf{v} \in \text{intnodes}(\mathcal{T})$ and $\mathbf{v}a \in \texttt{path}(x)$ for $a \in \{0,1\}$, we have $w^{\text{num}}_{\mathbf{v}(1-a)}(x) = w^{\text{den}}_{\mathbf{v}(1-a)}$ since $\mathbf{v}(1-a) \notin \texttt{path}(x)$. This recursion has a complexity $O(|\texttt{path}(x)|)$ where $|\texttt{path}(x)|$ is the number of nodes in $\texttt{path}(x)$, and is typically orders of magnitude smaller than $|\text{nodes}(\mathcal{T})|$ (in a well-balanced binary tree, one has the relation $|\texttt{path}(x)| = O(\log_2(|\text{nodes}(\mathcal{T})|))$). Moreover, we observe that the recursions used in (4.7) and (4.19) only need to save both $w_{\mathbf{v}}$ and $w^{\text{den}}_{\mathbf{v}}$ for any $\mathbf{v} \in \text{nodes}(\mathcal{T})$.

Finally, we have using (4.14) and (4.17) that

$$\widehat{f}(x) = \frac{\sum_{T \subset \mathcal{T}} w_T \widehat{y}_T(x)}{\sum_{T \subset \mathcal{T}} w_T} = \frac{w^{\text{num}}_{\texttt{root}}(x)}{w^{\text{den}}_{\texttt{root}}} =: \widehat{f}_{\texttt{root}}(x),$$

and we want to compute $\widehat{f}_{\texttt{root}}(x)$ recursively from $\widehat{f}_{\mathbf{v}}(x)$ where $\mathbf{v} \in \texttt{path}(x)$. First, whenever $\mathbf{v} = \mathbf{v}_{\mathcal{T}}(x)$ we have

$$\widehat{f}_{\mathbf{v}}(x) = \frac{w^{\text{num}}_{\mathbf{v}}(x)}{w^{\text{den}}_{\mathbf{v}}} = \frac{w_{\mathbf{v}} \widehat{y}_{\mathbf{v}}}{w_{\mathbf{v}}} = \widehat{y}_{\mathbf{v}},$$

while for $\mathbf{v} \neq \mathbf{v}_{\mathcal{T}}(x)$ and $\mathbf{v} \in \texttt{path}(x)$, we write

$$\widehat{f}_{\mathbf{v}}(x) = \frac{w^{\text{num}}_{\mathbf{v}}(x)}{w^{\text{den}}_{\mathbf{v}}} = \frac{\frac{1}{2} w_{\mathbf{v}} \widehat{y}_{\mathbf{v}} + \frac{1}{2} w^{\text{den}}_{\mathbf{v}(1-a)} w^{\text{num}}_{\mathbf{v}a}(x)}{w^{\text{den}}_{\mathbf{v}}} \tag{4.20}$$

$$= \frac{1}{2} \frac{w_{\mathbf{v}}}{w^{\text{den}}_{\mathbf{v}}} \widehat{y}_{\mathbf{v}} + \frac{1}{2} \frac{w^{\text{den}}_{\mathbf{v}(1-a)} w^{\text{den}}_{\mathbf{v}a}}{w^{\text{den}}_{\mathbf{v}}} \frac{w^{\text{num}}_{\mathbf{v}a}(x)}{w^{\text{den}}_{\mathbf{v}a}} \tag{4.21}$$

$$= \frac{1}{2} \frac{w_{\mathbf{v}}}{w^{\text{den}}_{\mathbf{v}}} \widehat{y}_{\mathbf{v}} + \Big(1 - \frac{1}{2} \frac{w_{\mathbf{v}}}{w^{\text{den}}_{\mathbf{v}}}\Big) \widehat{f}_{\mathbf{v}a}(x), \tag{4.22}$$

where (4.20) comes from (4.19) while (4.22) comes from (4.7). This proves the recursion stated in Equation (4.6) from Theorem 4.1, and to Algorithm 4. This concludes the proof of Theorem 4.1. □

4.6.2 Proofs of the results from Section 4.3

The proof of Theorem 4.2 is partly inspired from the proof of Theorem 2 in Dalalyan and Tsybakov [2008], that we generalize to exp-concave losses, while only least-squares regression is considered therein. Let E be a measurable space and P, Q be probability measures on it. The Kullback-Leibler divergence between P and Q is defined by

$$\text{KL}(P,Q) = \int_E \log\Big(\frac{\text{d}P}{\text{d}Q}\Big) \text{d}P$$

whenever P is absolutely continuous with respect to Q and equal to $+\infty$ otherwise. Also, if $h : E \to \mathbb{R}$ is a measurable function such that $\int_E h \text{d}P$ is well-defined on $\mathbb{R} \cup \{-\infty, +\infty\}$, we introduce

$$P_h := \frac{h}{\int_E h \text{d}P} \cdot P,$$

the probability measure on E with density $h / \int h \text{d}P$ with respect to P. A classical result is the Donsker-Varadhan variational formula [Donsker and Varadhan, 1976], which is at the core of the proofs of many PAC-Bayesian theorems [Catoni, 2007;

McAllester, 1999] and that we use here as well in the proof of Theorem 4.2. Its states that

$$\log\Big(\int_E \exp(h)\mathrm{d}Q\Big) + \mathrm{KL}(P,Q) - \int h\mathrm{d}P = \mathrm{KL}(P, Q_{\exp(h)}) \tag{4.23}$$

holds for any probability measures P and Q on E and any measurable function $h : E \to \mathbb{R}$. This entails in particular that

$$\log\Big(\int_E \exp(h)\mathrm{d}Q\Big) = \sup_P \Big\{\int h\mathrm{d}P - \mathrm{KL}(P,Q)\Big\},$$

where the supremum is over all probability measures on E, and where the supremum is achieved for $P = Q_{\exp(h)}$ whenever the term on the left-hand side is finite.

Proof of Theorem 4.2. Recall that the `oob` loss of a subtree $T \subset \mathcal{T}$ is given by

$$L_T = \sum_{i \in I_{\mathtt{oob}}} \ell(\widehat{y}_T(x_i), y_i)$$

and let us introduce

$$p_T = \frac{\pi(T)\exp(-\eta L_T)}{\sum_{T'}\pi(T')\exp(-\eta L_{T'})} \tag{4.24}$$

for any subtree $T \subset \mathcal{T}$. First, we use the fact that ℓ is a η-exp-concave loss function, hence η-mixable, see Section 3.3 from Cesa-Bianchi and Lugosi [2006], which entails, since p_T is a probability measure over the set of all subtrees $T \subset \mathcal{T}$, that

$$\ell\Big(\sum_T p_T \widehat{y}_T(x_i), y_i\Big) \leqslant -\frac{1}{\eta}\log\Big(\sum_T p_T \exp(-\eta\ell(\widehat{y}_T(x_i), y_i))\Big),$$

where the sums over T are over all subtrees $T \subset \mathcal{T}$. Now, summing this inequality over $i \in I_{\mathtt{oob}}$ and using the convexity of the log-sum-exp function leads to

$$\begin{aligned}
\sum_{i\in I_{\mathtt{oob}}} \ell\Big(\sum_T p_T \widehat{y}_T(x_i), y_i\Big) &\leqslant -\frac{1}{\eta}\sum_{i\in I_{\mathtt{oob}}} \log\Big(\sum_T p_T \exp(-\eta\ell(\widehat{y}_T(x_i), y_i))\Big) \\
&\leqslant -\frac{n_{\mathtt{oob}}}{\eta}\log\Big(\sum_T p_T \exp\Big(-\frac{\eta}{n_{\mathtt{oob}}}\sum_{i\in I_{\mathtt{oob}}}\ell(\widehat{y}_T(x_i), y_i)\Big)\Big) \\
&= -\frac{n_{\mathtt{oob}}}{\eta}\log\Big(\sum_T p_T \exp\Big(-\frac{\eta}{n_{\mathtt{oob}}}L_T\Big)\Big).
\end{aligned}$$

By plugging the definition of p_T into the previous expression, and by introducing $\rho(T) := \eta L_T / n_{\mathtt{oob}}$, we obtain

$$\begin{aligned}
&\frac{S}{n_{\mathtt{oob}}} := \frac{1}{n_{\mathtt{oob}}}\sum_{i\in I_{\mathtt{oob}}} \ell(\widehat{f}(x_i), y_i) \\
&\leqslant -\frac{1}{\eta}\log\Big(\sum_T \pi(T)\exp\big(-(n_{\mathtt{oob}}+1)\rho(T)\big)\Big) + \frac{1}{\eta}\log\Big(\sum_T \pi(T)\exp\big(-n_{\mathtt{oob}}\rho(T)\big)\Big).
\end{aligned}$$

The Hölder inequality implies that

$$\sum_T \pi(T)\exp\big(-n_{\mathtt{oob}}\rho(T)\big) \leqslant \Big(\sum_T \pi(T)\exp\big(-(n_{\mathtt{oob}}+1)\rho(T)\big)\Big)^{n_{\mathtt{oob}}/(n_{\mathtt{oob}}+1)},$$

thus

$$\frac{S}{n_{\mathtt{oob}}} \leqslant -\frac{1}{\eta(n_{\mathtt{oob}}+1)}\log\Big(\sum_T \pi(T)\exp\big(-(n_{\mathtt{oob}}+1)\rho(T)\big)\Big).$$

Using (4.23) with $h(T) = -(n_{\texttt{oob}} + 1)\rho(T)$ and $Q = \pi$, we have

$$\log\Big(\sum_T \pi(T)\exp\big(-(n_{\texttt{oob}}+1)\rho(T)\big)\Big) = -\sum_T P(T)(n_{\texttt{oob}}+1)\rho(T) - \mathrm{KL}(P,\pi) + \mathrm{KL}(P,\pi_{\exp(h)})$$

for any probability measure P over the set of subtrees of $\mathcal{T}$. This leads to

$$\frac{1}{n_{\texttt{oob}}}\sum_{i\in I_{\texttt{oob}}} \ell(\widehat{f}(x_i), y_i) \leqslant \frac{1}{n_{\texttt{oob}}}\sum_T P(T)L_T + \frac{1}{\eta(n_{\texttt{oob}}+1)}\mathrm{KL}(P,\pi)$$

for any P, since $\mathrm{KL}(P,\pi_{\exp(h)}) \geqslant 0$. So for the particular choice $P = \delta_T$ (the Dirac mass at T) for any subtree $T \subset \mathcal{T}$, we have

$$\begin{aligned}\frac{1}{n_{\texttt{oob}}}\sum_{i\in I_{\texttt{oob}}} \ell(\widehat{f}(x_i), y_i) &\leqslant \frac{1}{n_{\texttt{oob}}}L_T + \frac{1}{\eta(n_{\texttt{oob}}+1)}\log\big(\pi(T)^{-1}\big)\\ &\leqslant \frac{1}{n_{\texttt{oob}}}L_T + \frac{\log 2}{\eta}\frac{\|T\|}{n_{\texttt{oob}}+1},\end{aligned}$$

which concludes the proof of Theorem 4.2. □

The proof of Corollary 4.3 requires the next Lemma.

Lemma 4.6. *Consider classification with $\mathcal{Y} = \{1,\dots,K\}$ and a node $\mathbf{v} \in \mathrm{nodes}(\mathcal{T})$. Denote $n_{\mathbf{v}}(k)$ the number of samples of class k in node $\mathbf{v}$. We consider the Krichevsky-Trofimov estimator*

$$\widehat{y}(k) = \frac{n_{\mathbf{v}}(k) + 1/2}{n_{\mathbf{v}} + K/2}$$

where $n_{\mathbf{v}} = \sum_{k=1}^K n_{\mathbf{v}}(k)$ and the log loss $\ell(y', y) = -\log y'(y)$. Then, the following inequality holds

$$\sum_i \ell(\widehat{y}, y_i) - \inf_p \sum_i \ell(p, y_i) \leqslant \frac{K-1}{2}.$$

Proof. We know that the optimal p is given by $p_k = n_{\mathbf{v}}(k)/n_{\mathbf{v}}$. Indeed, it is the solution to the following constrained convex optimization problem

$$\min_p \; -\sum_{k=1}^K n_{\mathbf{v}}(k)\log p_k \quad \text{subject to} \quad \sum_{k=1}^K p_k = 1,$$

where we consider non-negativity to be already enforced by the objective and imposing $p_k \leqslant 1$ is redundant with the constraint $\sum_{k=1}^K p_k = 1$. We can write the Lagrangian function as

$$L(p,\lambda) = -\sum_{k=1}^K n_{\mathbf{v}}(k)\log p_k + \lambda\Big(\sum_{k=1}^K p_k - 1\Big)$$

and one can check the KKT conditions when taking $p_k = n_{\mathbf{v}}(k)/n_{\mathbf{v}}$ and $\lambda = n_{\mathbf{v}}$. Since we are dealing with a convex problem with linear constraints, this is a sufficient

optimality condition. Straightforward computations give

$$\begin{aligned}
\sum_i \ell(\widehat{y}, y_i) - \inf_p \sum_i \ell(p, y_i) &= \sum_{k=1}^{K} -n_{\mathbf{v}}(k) \log(\widehat{y}(k)) - \sum_{k=1}^{K} -n_{\mathbf{v}}(k) \log p_k \\
&= \sum_{k=1}^{K} -n_{\mathbf{v}}(k) \log \frac{n_{\mathbf{v}}(k) + 1/2}{n_{\mathbf{v}} + K/2} - \sum_{k=1}^{K} -n_{\mathbf{v}}(k) \log \frac{n_{\mathbf{v}}(k)}{n_{\mathbf{v}}} \\
&= \sum_{k=1}^{K} -n_{\mathbf{v}}(k) \Big(\log \frac{n_{\mathbf{v}}}{n_{\mathbf{v}} + K/2} + \log \frac{n_{\mathbf{v}}(k) + 1/2}{n_{\mathbf{v}}} - \log \frac{n_{\mathbf{v}}(k)}{n_{\mathbf{v}}} \Big) \\
&= -n_{\mathbf{v}} \log \frac{n_{\mathbf{v}}}{n_{\mathbf{v}} + K/2} + \sum_{k=1}^{K} -n_{\mathbf{v}}(k) \Big(\log \frac{n_{\mathbf{v}}(k) + 1/2}{n_{\mathbf{v}}} - \log \frac{n_{\mathbf{v}}(k)}{n_{\mathbf{v}}} \Big) \\
&= n_{\mathbf{v}} \log \frac{n_{\mathbf{v}} + K/2}{n_{\mathbf{v}}} + \sum_{k=1}^{K} n_{\mathbf{v}}(k) \log \frac{n_{\mathbf{v}}(k)}{n_{\mathbf{v}}(k) + 1/2}.
\end{aligned}$$

Now, using the concavity of the logarithm gives

$$\sum_i \ell(\widehat{y}, y_i) - \inf_p \sum_i \ell(p, y_i) \leqslant n_{\mathbf{v}} \log \frac{n_{\mathbf{v}} + K/2}{n_{\mathbf{v}}} + n_{\mathbf{v}} \log \Big(\sum_{k=1}^{K} \frac{n_{\mathbf{v}}(k)}{n_{\mathbf{v}}} \frac{n_{\mathbf{v}}(k)}{n_{\mathbf{v}}(k) + 1/2} \Big),$$

and the fact that $x \mapsto x/(x + 1/2)$ is non-decreasing and $n_{\mathbf{v}}(k) \leqslant n_{\mathbf{v}}$ leads to

$$\begin{aligned}
\sum_i \ell(\widehat{y}, y_i) - \inf_p \sum_i \ell(p, y_i) &\leqslant n_{\mathbf{v}} \log \frac{n_{\mathbf{v}} + K/2}{n_{\mathbf{v}}} + n_{\mathbf{v}} \log \frac{n_{\mathbf{v}}}{n_{\mathbf{v}} + 1/2} \\
&= n_{\mathbf{v}} \log \frac{n_{\mathbf{v}} + 1/2 + (K-1)/2}{n_{\mathbf{v}} + 1/2} \\
&= n_{\mathbf{v}} \log \Big(1 + \frac{K-1}{2n_{\mathbf{v}} + 1} \Big) \leqslant \frac{K-1}{2}.
\end{aligned}$$

This concludes the proof of Lemma 4.6. □

Proof of Corollary 4.3. The log-loss is trivially 1-exp-concave, so that we can choose $\eta = 1$. Following Theorem 4.2, it remains to bound the regret of the tree forecaster T with respect to the optimal labeling of its leaves. For classification and the log loss, we use Lemma 4.6 to obtain

$$\sum_{i \in I_{\text{oob}} : x_i \in C_{\mathbf{v}}} \ell(\widehat{y}_T(x_i), y_i) - \inf_p \sum_{i \in I_{\text{oob}} : x_i \in C_{\mathbf{v}}} \ell(p, y_i) \leqslant \frac{K-1}{2}$$

for any subtree T and any $\mathbf{v} \in \text{leaves}(T)$. Now, summing over $\mathbf{v} \in \text{leaves}(T)$, of cardinality $(\|T\| + 1)/2$, leads to

$$\sum_{i \in I_{\text{oob}}} \ell(\widehat{f}(x_i), y_i) - \sum_{i \in I_{\text{oob}}} \ell(g_T(x_i), y_i) \leqslant \|T\| \log 2 + \frac{(K-1)(\|T\| + 1)}{4},$$

which concludes the proof of Corollary 4.3. □

Proof of Corollary 4.4. The square loss is $1/(8B^2)$-exp-concave on $[-B, B]$, see Cesa-Bianchi and Lugosi [2006], so we can choose $\eta = 1/(8B^2)$. Following Theorem 4.2, it remains to bound the regret of the tree forecaster T with respect to the optimal

labeling of its leaves. For regression with the least-squares loss, since we use the empirical mean forecaster (4.2), we have

$$\sum_{i \in I_{\texttt{oob}}: x_i \in C_{\mathbf{v}}} \ell(\widehat{y}_T(x_i), y_i) - \inf_b \sum_{i \in I_{\texttt{oob}}: x_i \in C_{\mathbf{v}}} \ell(b, y_i) = 0$$

for any subtree T and any leaf $\mathbf{v} \in \text{leaves}(T)$. The rest of the proof follows that of Corollary 4.3. □

4.7 Appendix: experiments

4.7.1 Supplementary details on experiments

We report in Table 4.3 the same test AUC scores as in Table 4.1 of all algorithms after hyperoptimization on the considered datasets. Standard-deviations displayed between parentheses are computed from 5 trainings with different random seeds. We observe that WW has better (or identical in some cases) performances than RF on all datasets and that it is close to that of EGB libraries (bold is for best EGB performance, underline for best `RF`n or `WW`n performance).

We report also in Table 4.4 the same training time and test AUC as in Table 4.2, with standard-deviations displayed between parentheses computed from 5 trainings with different random seeds, all with default hyperparameters of each algorithm. We observe that WW is almost always the fastest algorithm, for performances comparable to ones of all baselines (bold is for best EGB training time or performance, underline for best `RF` or `WW` training time or performance).

4.7.2 Supplementary details about hyperparameters tuning

In this Section, we provide extra information about hyperparameters optimization. For XGBoost, LightGBM and CatBoost, with all other hyperparameters fixed, we use early stopping by monitoring the log loss on the validation set, the maximum number of boosting iterations being set at $5,000$. The best number of iterations is used together with other best hyperparameters to refit over the whole training set before evaluation on the test set. For `scikit-learn`'s Random Forest and WildWood, we report results both for 10 and 100 trees, note that the default choice is 10 for WildWood (since subtrees aggregation allows to use fewer trees than RF) while default is 100 in `scikit-learn`. We list the hyperparameters search space of each algorithm below.

XGBoost

- `eta`: log-uniform distribution $[e^{-7}, 1]$;
- `max_depth`: discrete uniform distribution $[2, 10]$;
- `subsample`: uniform $[0.5, 1]$;
- `colsample_bytree`: uniform $[0.5, 1]$;
- `colsample_bylevel`: uniform $[0.5, 1]$;
- `min_child_weight`: log-uniform distribution $[e^{-16}, e^{5}]$;

Table 4.3 – The same test AUC scores as in Table 4.1 of all algorithms after hyperoptimization on the considered datasets. Standard-deviations displayed between parentheses are computed from 5 trainings with different random seeds. We observe that WW has better (or identical in some cases) performances than RF on all datasets and that it is close to that of EGB libraries (bold is for best EGB performance, underline for best RFn or WWn performance).

	XGB	LGBM	CB	HGB	RF10	RF100	WW10	WW100
adult	0.930 (2.7e-04)	**0.931** (1.2e-04)	0.927 (2.9e-04)	0.930 (2.9e-04)	0.916 (3.1e-04)	0.919 (1.8e-04)	0.918 (2.8e-04)	0.919 (1.6e-04)
bank	0.933 (1.5e-04)	**0.935** (4.1e-05)	0.925 (6.5e-04)	0.930 (7.4e-04)	0.917 (6.4e-04)	0.929 (2.3e-04)	0.924 (9.1e-04)	0.931 (3.5e-04)
breastcancer	0.991 (4.4e-04)	0.993 (1.1e-04)	0.987 (6.7e-03)	**0.994** (0.0e+00)	0.974 (3.1e-03)	0.978 (1.1e-03)	0.992 (1.5e-03)	0.992 (5.9e-04)
car	0.999 (2.3e-04)	**1.000** (3.8e-05)	**1.000** (6.0e-05)	**1.000** (0.0e+00)	0.996 (6.0e-04)	0.996 (2.2e-04)	0.997 (9.5e-04)	0.998 (1.2e-04)
covtype	**0.999** (7.5e-06)	**0.999** (3.6e-06)	0.998 (2.5e-05)	**0.999** (5.2e-06)	0.996 (2.2e-04)	0.998 (7.6e-05)	0.996 (1.2e-04)	0.998 (6.1e-05)
default-cb	0.780 (3.5e-04)	**0.783** (1.2e-04)	0.780 (3.9e-04)	0.779 (6.7e-04)	0.748 (4.2e-03)	0.774 (9.5e-04)	0.773 (6.1e-04)	0.778 (4.8e-04)
higgs	0.853 (6.7e-05)	**0.857** (1.8e-05)	0.847 (2.2e-05)	0.853 (7.0e-05)	0.812 (1.4e-04)	0.834 (3.1e-05)	0.818 (7.8e-05)	0.835 (2.3e-05)
internet	0.934 (2.9e-04)	0.910 (1.8e-04)	**0.938** (1.3e-03)	0.911 (1.1e-16)	0.841 (1.2e-02)	0.911 (4.4e-03)	0.923 (1.5e-03)	0.928(6.3e-04)
kddcup	**1.000** (6.1e-08)	**1.000** (4.1e-07)	**1.000** (5.8e-07)	**1.000** (5.7e-07)	0.997 (9.6e-04)	0.998 (2.1e-03)	1.000 (1.4e-06)	1.000 (6.0e-6)
kick	**0.777** (7.7e-04)	0.770 (2.8e-04)	**0.777** (6.8e-04)	0.771 (1.6e-03)	0.736 (1.9e-03)	0.752 (8.1e-04)	0.756 (1.5e-03)	0.763 (1.2e-03)
letter	**1.000** (1.3e-05)	**1.000** (2.7e-06)	**1.000** (4.1e-06)	**1.000** (2.1e-05)	0.997 (2.0e-04)	0.999 (1.8e-04)	0.996 (3.3e-04)	0.999 (3.5e-05)
satimage	**0.991** (2.1e-04)	**0.991** (3.6e-05)	**0.991** (2.3e-04)	0.987 (0.0e+00)	0.980 (1.3e-03)	0.989 (1.7e-04)	0.983 (7.1e-04)	0.991 (3.2e-04)
sensorless	**1.000** (4.9e-07)	**1.000** (1.4e-07)	**1.000** (4.4e-06)	**1.000** (2.9e-06)	1.000 (4.4e-05)	1.000 (1.2e-05)	1.000 (1.5e-05)	1.000 (5.8e-06)
spambase	**0.990** (1.5e-04)	**0.990** (5.2e-05)	0.987 (1.2e-03)	0.986 (0.0e+00)	0.980 (4.7e-04)	0.986 (2.6e-04)	0.983 (1.1e-03)	0.987 (2.7e-04)

Table 4.4 – The same training time table as in Table 4.2, as average over 5 runs, with standard deviation computed from 5 runs reported between parenthesis, for default parameters for each model. Top: training time in seconds; bottom: test AUC. We observe that WW is almost always the fastest algorithm, for performances comparable to ones of all baselines (bold is for best EGB training time or performance, underline for best `RF` or `WW` training time or performance).

	Training time (seconds)					
	XGB	LGBM	CB	HGB	RF	WW
covtype	10 (0.6)	**3** (0.1)	120 (9.3)	14 (7.7)	21 (0.9)	3 (0.1)
higgs	36 (0.6)	**30** (1.4)	653 (8.7)	85 (0.2)	1389 (11.1)	179 (4.5)
internet	9 (0.7)	**4** (0.1)	188 (2.4)	8 (0.3)	0.4 (0.0)	0.3 (0.0)
kddcup	175 (5.1)	41 (2.6)	2193 (13.2)	**31** (0.2)	208 (3.8)	12 (0.8)
kick	7 (0.2)	**0.4** (0.0)	50 (0.7)	0.7 (0.1)	31 (0.1)	5 (0.0)

	Test AUC					
	XGB	LGBM	CB	HGB	RF	WW
covtype	0.986 (2e-04)	0.978 (2e-03)	**0.989** (9e-05)	0.960 (1e-02)	0.998 (6e-05)	0.979 (5e-04)
higgs	0.823 (3e-04)	0.812 (2e-04)	**0.840** (8e-05)	0.812 (2e-04)	0.838 (9e-05)	0.813 (1e-04)
internet	**0.918** (2e-05)	0.828 (0e+00)	0.910 (8e-03)	0.500 (0e+00)	0.862 (3e-03)	0.889 (7e-03)
kddcup	**1.000** (3e-07)	0.638 (3e-02)	0.988 (7e-03)	0.740 (6e-02)	0.998 (2e-03)	1.000 (3e-06)
kick	0.768 (4e-04)	0.757 (0e+00)	**0.781** (3e-04)	0.773 (2e-03)	0.747 (2e-03)	0.751 (2e-03)

- `alpha`: 0 with probability 0.5, and log-uniform distribution $[e^{-16}, e^2]$ with probability 0.5;
- `lambda`: 0 with probability 0.5, and log-uniform distribution $[e^{-16}, e^2]$ with probability 0.5;
- `gamma`: 0 with probability 0.5, and log-uniform distribution $[e^{-16}, e^2]$ with probability 0.5;

LightGBM

- `learning_rate`: log-uniform distribution $[e^{-7}, 1]$;
- `num_leaves`: discrete log-uniform distribution $[1, e^7]$;
- `feature_fraction`: uniform $[0.5, 1]$;
- `bagging_fraction`: uniform $[0.5, 1]$;
- `min_data_in_leaf`: discrete log-uniform distribution $[1, e^6]$;
- `min_sum_hessian_in_leaf`: log-uniform distribution $[e^{-16}, e^5]$;
- `lambda_l1`: 0 with probability 0.5, and log-uniform distribution $[e^{-16}, e^2]$ with probability 0.5;
- `lambda_l2`: 0 with probability 0.5, and log-uniform distribution $[e^{-16}, e^2]$ with probability 0.5;

CatBoost

- `learning_rate`: log-uniform distribution $[e^{-7}, 1]$;
- `random_strength`: discrete uniform distribution over $\{1, 20\}$;

- `one_hot_max_size`: discrete uniform distribution over $\{0, 25\}$;
- `l2_leaf_reg`: log-uniform distribution $[1, 10]$;
- `bagging_temperature`: uniform $[0, 1]$.

HistGradientBoosting

- `learning_rate`: log-uniform distribution $[e^{-4}, 1]$;
- `max_leaf_nodes`: discrete log-uniform distribution $[1, e^{7}]$;
- `min_samples_leaf`: discrete log-uniform distribution $[1, e^{6}]$;
- `l2_regularization`: 0 with probability 0.5, and log-uniform distribution $[e^{-16}, e^{2}]$ with probability 0.5;

RandomForest

- `max_features`: `None` with probability 0.5, and `sqrt` with probability 0.5;
- `min_samples_leaf`: uniform over $\{1, 5, 10\}$ and we set `min_samples_split` $=$ $2 \times$ `min_samples_leaf`;

WildWood

- `multiclass`: `multinomial` with probability 0.5, and `ovr` with probability 0.5;
- `min_samples_leaf`: uniform over $\{1, 5, 10\}$ and we set `min_samples_split` $=$ $2 \times$ `min_samples_leaf`;
- `step`: log-uniform distribution $[e^{-3}, e^{6}]$;
- `dirichlet`: log-uniform distribution $[e^{-7}, e^{2}]$;
- `cat_split_strategy`: `binary` with probability 0.5, and `all` with probability 0.5;
- `max_features`: `None` with probability 0.5, and `sqrt` with probability 0.5.

4.7.3 Datasets

The main characteristics of the datasets used in the paper are summarized in Table 4.5. We provide in Table 4.6 the URL of all the datasets used, most of them are from the UCI machine learning repository Dua and Graff [2017].

4.7.4 Sensitivity of hyperparameters of Wildwood

In Table 4.7 we illustrate the effects of hyperparameters on WW's performance on a few datasets, measured by the test AUC. We can observe in this table that it is only weakly affected by different combinations of hyperparameters.

Dataset	# Samples	# Features	# Categorical features	# Classes	Gini
adult	48,842	14	8	2	0.36
bank	45,211	16	10	2	0.21
breastcancer	569	30	0	2	0.47
car	1,728	6	6	4	0.46
covtype	581,012	54	0	7	0.62
default_cb	30,000	23	3	2	0.34
higgs	11,000,000	28	0	2	0.50
internet	10,108	70	70	46	0.88
kddcup99	4,898,431	41	7	23	0.58
kick	72,983	32	18	2	0.22
letter	20,000	16	0	26	0.96
satimage	5,104	36	0	6	0.81
sensorless	58,509	48	0	11	0.91
spambase	4,601	57	0	2	0.48

Table 4.5 – Main characteristics of the datasets used in experiments, including number of samples, number of features, number of categorical features, number of classes and the Gini index of the class distribution on the whole datasets (rescaled between 0 and 1), in order to quantify class unbalancing.

Dataset	URL
adult	https://archive.ics.uci.edu/ml/datasets/Adult
bank	https://archive.ics.uci.edu/ml/datasets/bank+marketing
breastcancer	https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic)
car	https://archive.ics.uci.edu/ml/datasets/car+evaluation
covtype	https://archive.ics.uci.edu/ml/datasets/covertype
default_cb	https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients
higgs	https://archive.ics.uci.edu/ml/datasets/HIGGS
internet	https://kdd.ics.uci.edu/databases/internet_usage/internet_usage.html
kddcup99	https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
kick	https://www.openml.org/d/41162
letter	https://archive.ics.uci.edu/ml/datasets/letter+recognition
satimage	https://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite)
sensorless	https://archive.ics.uci.edu/ml/datasets/dataset+for+sensorless+drive+diagnosis
spambase	https://archive.ics.uci.edu/ml/datasets/spambase

Table 4.6 – The URLs of all the datasets used in the paper, giving direct download links and supplementary details.

Table 4.7 – Areas under the ROC curves (AUC) obtained on test samples with WildWood (using 100 trees in the forest) on the adult, bank and car datasets with combinations of several hyper parameters. We observe that WildWood's performance does not vary significantly with respect to these hyperparameters.

adult			
$n_{\text{min-leaf}}$	α	η	AUC
1	0.1	0.1	0.913
1	0.1	1.0	0.916
1	0.1	10.0	0.918
1	0.5	0.1	0.913
1	0.5	1.0	0.917
1	0.5	10.0	0.919
1	2.5	0.1	0.913
1	2.5	1.0	0.917
1	2.5	10.0	0.919
5	0.1	0.1	0.913
5	0.1	1.0	0.916
5	0.1	10.0	0.918
5	0.5	0.1	0.913
5	0.5	1.0	0.916
5	0.5	10.0	0.918
5	2.5	0.1	0.913
5	2.5	1.0	0.917
5	2.5	10.0	0.918
10	0.1	0.1	0.913
10	0.1	1.0	0.916
10	0.1	10.0	0.917
10	0.5	0.1	0.913
10	0.5	1.0	0.916
10	0.5	10.0	0.917
10	2.5	0.1	0.913
10	2.5	1.0	0.916
10	2.5	10.0	0.918

bank			
$n_{\text{min-leaf}}$	α	η	AUC
1	0.1	0.1	0.919
1	0.1	1.0	0.926
1	0.1	10.0	0.929
1	0.5	0.1	0.920
1	0.5	1.0	0.927
1	0.5	10.0	0.929
1	2.5	0.1	0.921
1	2.5	1.0	0.927
1	2.5	10.0	0.929
5	0.1	0.1	0.919
5	0.1	1.0	0.926
5	0.1	10.0	0.928
5	0.5	0.1	0.920
5	0.5	1.0	0.926
5	0.5	10.0	0.928
5	2.5	0.1	0.921
5	2.5	1.0	0.927
5	2.5	10.0	0.928
10	0.1	0.1	0.920
10	0.1	1.0	0.926
10	0.1	10.0	0.927
10	0.5	0.1	0.920
10	0.5	1.0	0.926
10	0.5	10.0	0.927
10	2.5	0.1	0.920
10	2.5	1.0	0.926
10	2.5	10.0	0.928

car			
$n_{\text{min-leaf}}$	α	η	AUC
1	0.1	0.1	0.992
1	0.1	1.0	0.995
1	0.1	10.0	0.995
1	0.5	0.1	0.992
1	0.5	1.0	0.994
1	0.5	10.0	0.995
1	2.5	0.1	0.990
1	2.5	1.0	0.992
1	2.5	10.0	0.993
5	0.1	0.1	0.990
5	0.1	1.0	0.992
5	0.1	10.0	0.992
5	0.5	0.1	0.990
5	0.5	1.0	0.992
5	0.5	10.0	0.992
5	2.5	0.1	0.987
5	2.5	1.0	0.991
5	2.5	10.0	0.991
10	0.1	0.1	0.983
10	0.1	1.0	0.987
10	0.1	10.0	0.987
10	0.5	0.1	0.983
10	0.5	1.0	0.987
10	0.5	10.0	0.987
10	2.5	0.1	0.981
10	2.5	1.0	0.985
10	2.5	10.0	0.986

4.7.5 Supplementary details about assets used (versions and licenses)

The versions and licenses of the libraries used in our experiments are:

- `catboost` (0.25.1), Apache License 2.0
- `hyperopt` (0.2.5), license: https://github.com/hyperopt/hyperopt/blob/master/LICENSE.txt
- `joblib` (0.17), BSD-3-Clause License
- `lightgbm` (3.2.1), MIT License
- `matplotlib` (3.3.1), license: https://github.com/matplotlib/matplotlib/blob/master/LICENSE/LICENSE
- `numba` (0.52), BSD 2-Clause "Simplified" License
- `numpy` (1.19.2), BSD-3-Clause License
- `pandas` (1.2.4), BSD-3-Clause License
- `python` (3.7.9), Python Software Fundation Licence version 2
- `scikit-learn` (0.24.2), BSD 3-Clause License
- `scipy` (1.5.4), BSD 3-Clause License
- `seaborn` (0.11), BSD-3-Clause License
- `xgboost` (1.4.1), Apache License 2.0

All the datasets used are publicly accessible and have no copyright restrictions.

Bibliography

K. Alghatani, N. Ammar, A. Rezgui, A. Shaban-Nejad, et al. Predicting intensive care unit length of stay and mortality using patient vital signs: Machine learning model development and validation. *JMIR Medical Informatics*, 9(5):e21347, 2021. 94

S. Arlot and R. Genuer. Analysis of purely random forests bias. *arXiv preprint arXiv:1407.3939*, 2014. 94, 101

S. Athey, J. Tibshirani, and S. Wager. Generalized random forests, 2018. 94

J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, 2015. 102

G. Biau. Analysis of a random forests model. *The Journal of Machine Learning Research*, 13(1):1063–1095, 2012. 94

G. Biau and E. Scornet. A random forest guided tour. *TEST*, 25(2):197–227, 2016. ISSN 1863-8260. doi: 10.1007/s11749-016-0481-7. URL http://dx.doi.org/10.1007/s11749-016-0481-7. 92, 93, 96

G. Biau, L. Devroye, and G. Lugosi. Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 9(9), 2008. 94

L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 95

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL http://dx.doi.org/10.1023/A:1010933404324. 92, 93, 96

L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*. The Wadsworth statistics/probability series. CRC, Monterey, CA, 1984. 96

A. Calviño. On random-forest-based prediction intervals. In C. Analide, P. Novais, D. Camacho, and H. Yin, editors, *Intelligent Data Engineering and Automated Learning – IDEAL 2020*, pages 172–184, Cham, 2020. Springer International Publishing. ISBN 978-3-030-62362-3. 94

O. Catoni. *Statistical Learning Theory and Stochastic Optimization: Ecole d'Eté de Probabilités de Saint-Flour XXXI - 2001*, volume 1851 of *Lecture Notes in Mathematics*. Springer-Verlag Berlin Heidelberg, 2004. ISBN 9783540445074. URL https://books.google.fr/books?id=3Ih8CwAAQBAJ. 92, 99

O. Catoni. *PAC-Bayesian Supervised Classification: The Thermodynamics of Statistical Learning*, volume 56 of *IMS Lecture Notes Monograph Series*. Institute of Mathematical Statistics, 2007. ISBN 9780940600720. URL https://books.google.fr/books?id=acnaAAAAMAAJ. 100, 107

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, Cambridge, New York, USA, 2006. 108, 110

N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9): 2050–2057, Sept 2004. ISSN 0018-9448. doi: 10.1109/TIT.2004.833339. 101

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016. URL http://arxiv.org/abs/1603.02754. 93, 94, 95, 96, 101

X. Chen and H. Ishwaran. Random forests for genomic data analysis. *Genomics*, 99(6):323–329, 2012. 94

Z. Chen, L. Zhou, and W. Yu. Adasyn-random forest based intrusion detection model, 2021. 94

H. A. Chipman, E. I. George, and R. E. McCulloch. Bayesian CART model search. *Journal of the American Statistical Association*, 93(443):935–948, 1998. doi: 10.1080/01621459.1998.10473750. URL http://dx.doi.org/10.1080/01621459.1998.10473750. 92

H. A. Chipman, E. I. George, and R. E. McCulloch. BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010. 92

D. Coppersmith, S. Hong, and J. Hosking. Partitioning nominal attributes in decision trees. *Data Mining and Knowledge Discovery*, 3:197–217, 01 1999. doi: 10.1023/A:1009869804967. 97

A. Criminisi, J. Shotton, E. Konukoglu, et al. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. 2012. 94

A. Dalalyan and A. B. Tsybakov. Aggregation by exponential weighting, sharp pac-bayesian bounds and sparsity. *Machine Learning*, 72(1):39–61, Aug 2008. ISSN 1573-0565. doi: 10.1007/s10994-008-5051-0. URL http://dx.doi.org/10.1007/s10994-008-5051-0. 107

D. G. T. Denison, B. K. Mallick, and A. F. M. Smith. A bayesian CART algorithm. *Biometrika*, 85(2):363–377, 1998. doi: 10.1093/biomet/85.2.363. URL http://dx.doi.org/10.1093/biomet/85.2.363. 92

M. D. Donsker and S. S. Varadhan. Asymptotic evaluation of certain markov process expectations for large time—iii. *Communications on pure and applied Mathematics*, 29(4):389–461, 1976. 107

A. V. Dorogush, V. Ershov, and A. Gulin. Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018. 93, 101

D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml. 102, 114

B. Efron and R. Tibshirani. Improvements on cross-validation: the 632+ bootstrap method. *Journal of the American Statistical Association*, 92(438):548–560, 1997. 96

R. Genuer. Variance reduction in purely random forests. *Journal of Nonparametric Statistics*, 24(3):543–562, 2012. 94, 101

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006. 93, 94

I. Ghosal and G. Hooker. Generalised boosted forests, 2021. 94

D. P. Helmbold and R. E. Schapire. Predicting nearly as well as the best pruning of a decision tree. *Machine Learning*, 27(1):51–68, 1997. ISSN 1573-0565. doi: 10.1023/A:1007396710653. URL http://dx.doi.org/10.1023/A:1007396710653. 92, 99, 104

K. Hoffman, J. Y. Sung, and A. Zazzera. Multi-output random forest regression to emulate the earliest stages of planet formation, 2021. 94

Joblib Development Team. Joblib: running python functions as pipeline jobs, 2020. URL https://joblib.readthedocs.io/. 101

S. J. Kazemitabar, A. A. Amini, A. Bloniarz, and A. Talwalkar. Variable importance using decision trees. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 425–434, 2017. 94

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf. 93, 94, 95, 96, 97, 101

B. Lakshminarayanan, D. M. Roy, and Y. W. Teh. Mondrian forests: Efficient online random forests. In *Advances in Neural Information Processing Systems 27*, pages 3140–3148. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5234-mondrian-forests-efficient-online-random-forests.pdf. 94

S. K. Lam, A. Pitrou, and S. Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450340052. doi: 10.1145/2833157.2833162. URL https://doi.org/10.1145/2833157.2833162. 101

A. H. Li and A. Martin. Forest-type regression with general losses and robust forest. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2091–2100. PMLR, 06–11 Aug 2017. URL http://proceedings.mlr.press/v70/li17e.html. 94

F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008. doi: 10.1109/ICDM.2008.17. 104

G. Louppe. *Understanding random forests: From theory to practice*. PhD thesis, University of Liege, 2014. 92, 93, 96, 97, 101

G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts. Understanding variable importances in forests of randomized trees. *Advances in neural information processing systems 26*, 2013. 94

J. D. Loyal, R. Zhu, Y. Cui, and X. Zhang. Dimension reduction forests: Local variable importance using structured random forests, 2021. 94

D. A. McAllester. Some pac-bayesian theorems. In *Proceedings of the 11th Annual conference on Computational Learning Theory (COLT)*, pages 230–234. ACM, 1998. 100

D. A. McAllester. Pac-bayesian model averaging. In *Proceedings of the 20th Annual Conference on Computational Learning Theory (COLT)*, pages 164–170. ACM, 1999. 100, 108

L. Mentch and S. Zhou. Randomization as regularization: A degrees of freedom explanation for random forest success, 2020. 94

J. Mourtada, S. Gaïffas, E. Scornet, et al. Minimax optimal rates for mondrian trees and forests. *Annals of Statistics*, 48(4):2253–2276, 2020. 94, 101

J. Mourtada, S. Gaïffas, and E. Scornet. AMF: Aggregated Mondrian forests for online learning. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2021. doi: https://doi.org/10.1111/rssb.12425. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/rssb.12425. 92, 94, 100

N. C. Oza and S. Russell. Online bagging and boosting. In *Proceedings of the 8th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2001. 95

F. Pargent, F. Pfisterer, J. Thomas, and B. Bischl. Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features, 2021. 97

A. Patil and S. Singh. Differential private random forest. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2623–2630, 2014. doi: 10.1109/ICACCI.2014.6968348. 94

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 É. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 97, 101

P. Probst, M. N. Wright, and A.-L. Boulesteix. Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3):e1301, 2019. 93, 94

L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. Catboost: unbiased boosting with categorical features. *arXiv preprint arXiv:1706.09516*, 2017. 93, 95, 96, 97, 101

Y. Qi. Random forest for bioinformatics. In *Ensemble machine learning*, pages 307–323. Springer, 2012. 94

M.-H. Roy and D. Larocque. Prediction intervals with random forests. *Statistical Methods in Medical Research*, 29(1):205–229, 2020. doi: 10.1177/0962280219829885. URL https://doi.org/10.1177/0962280219829885. PMID: 30786820. 94

E. Scornet. On the asymptotics of random forests. *Journal of Multivariate Analysis*, 146:72–83, 2016a. 94, 101

E. Scornet. Random forests and kernel methods. *IEEE Transactions on Information Theory*, 62:1485–1500, 2016b. 94, 101

E. Scornet, G. Biau, and J.-P. Vert. Consistency of random forests. *The Annals of Statistics*, 43(4):1716–1741, 08 2015. doi: 10.1214/15-AOS1321. URL http://dx.doi.org/10.1214/15-AOS1321. 94, 101

Z. She, Z. Wang, T. Ayer, A. Toumi, and J. Chhatwal. Estimating County-Level COVID-19 Exponential Growth Rates Using Generalized Random Forests. *arXiv e-prints*, art. arXiv:2011.01219, Oct. 2020. 94

A. Subasi, E. Alickovic, and J. Kevric. Diagnosis of chronic kidney disease by using random forest. In *CMBEBIH 2017*, pages 589–594. Springer, 2017. 94

M. A. Taddy, R. B. Gramacy, and N. G. Polson. Dynamic trees for learning and design. *Journal of the American Statistical Association*, 106(493):109–123, 2011. doi: 10.1198/jasa.2011.ap09769. URL http://dx.doi.org/10.1198/jasa.2011.ap09769. 92

A. Tavakoli, S. Kumar, M. Boukhechba, and A. Heydarian. Driver state and behavior detection through smart wearables, 2021. 94

T. J. Tjalkens, Y. M. Shtarkov, and F. M. J. Willems. Sequential weighting algorithms for multi-alphabet sources. In *6th Joint Swedish-Russian International Workshop on Information Theory*, pages 230–234, 1993. 98

A. B. Tsybakov. Optimal rates of aggregation. In *Learning theory and kernel machines*, pages 303–313. Springer, 2003. 100

I. Verdinelli and L. Wasserman. Forest guided smoothing, 2021. 94

V. Vovk. A game of prediction with expert advice. *Journal of Computer and System Sciences*, 56(2):153–173, 1998. 101

F. M. J. Willems. The context-tree weighting method: Extensions. *IEEE Transactions on Information Theory*, 44(2):792–798, Mar 1998. 92, 99

F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens. The context-tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41(3): 653–664, May 1995. 92, 99

H. Zhang, J. Zimmerman, D. Nettleton, and D. J. Nordman. Random forest prediction intervals. *The American Statistician*, 74(4):392–406, 2020. doi: 10.1080/00031305.2019.1585288. URL https://doi.org/10.1080/00031305.2019.1585288. 94

S. Zhou and L. Mentch. Trees, Forests, Chickens, and Eggs: When and Why to Prune Trees in a Random Forest. *arXiv e-prints*, art. arXiv:2103.16700, Mar. 2021. 94

Y. Zhou and G. Qiu. Random forest for label ranking. *Expert Systems with Applications*, 112:99–109, 2018. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2018.06.036. URL https://www.sciencedirect.com/science/article/pii/S095741741830383X. 94

Chapter 5

Online logistic regression: towards an efficient algorithm with better regret? [1]

> “ 路漫漫其修远兮，吾将上下而求索。
> *On and on stretched my road, long it was and far, I would go high and go low in this search that I made.* ”
>
> Qu Yuan (translated by Stephen Owen)

Contents

[1] This chapter is based on the joint work with I. Merad, J. Mourtada, S. Gaiffas.

5.1 Introduction

We consider online logistic regression: an online learning problem with binary labels and logistic loss. Under this setting, at each time step $t = 1, 2, \ldots$, an agent successively makes predictions $\widehat{y}_t$, based on all the previous observations $((x_1, y_1), \ldots, (x_{t-1}, y_{t-1}))$ and the features x_t of time t. Once the true binary label $y_t \in \{-1, +1\}$ is unveiled, the agent suffers a logistic loss

$$\ell(\widehat{y}_t, y_t) \coloneqq \log\left(1 + \exp\left(-\widehat{y}_t y_t\right)\right).$$

The quality of the successive predictions $(\widehat{y}_t)_t$ is assessed by the *regret*, defined as the excess loss induced by $(\widehat{y}_t)_t$ versus an algorithm that uses a logistic model with the best possible fixed parameter $\theta \in \Theta$ in hindsight,

$$\mathrm{Regret}_n \coloneqq \sum_{t=1}^{n} \ell(\widehat{y}_t, y_t) - \inf_{\theta \in \Theta} \sum_{t=1}^{n} \ell(\theta^\top x_t, y_t)$$

where the space of parameters $\Theta \subset \mathbb{R}^d$ is called *comparison class*, and $n \in \mathbb{N}$ denotes the number of total rounds.

Online logistic regression falls into the supervised online learning setting, where a learning algorithm $\mathcal{A}$ takes as input at time t all the previous observations $S_{t-1} = ((x_1, y_1), \ldots, (x_{t-1}, y_{t-1}))$ as well as the features x_t of time t, and predicts $\widehat{y}_t = \mathcal{A}(S_{t-1}, x_t)$. An algorithm $\mathcal{A}$ is *proper* if the predictor returned by the learning algorithm $x \mapsto \mathcal{A}(S_t, x)$ belongs to the space of functions $\{x \mapsto \theta^\top x, \theta \in \Theta\}$ for all time steps $t \geqslant 1$; otherwise, the algorithm is *improper*.

- In the proper setting, the predictor $\mathcal{A}(S_{t-1})(\cdot)$ cannot use x_t. Follow-The-Leader-type methods apply, and the problem can also be seen as instance of Online Convex Optimization (OCO).
- In the improper setting, the predictor is allowed to use the knowledge of x_t: indeed, the predictor writes $x_t \mapsto \mathcal{A}(S_{t-1}, x_t)(x_t)$.

We can further separate algorithms into two families: Bayesian and non-Bayesian methods. In this chapter, we introduce a new improper non-Bayesian algorithm, to investigate if it is possible to achieve an optimal regret while remaining computationally efficient. We also propose a survey of recent advances on the topic of online logistic regression.

Excellent introductory manuals on the general subject of online learning include Hazan [2019]; Orabona [2019] and Cesa-Bianchi and Lugosi [2006]. We formally formulate the online logistic regression problem and introduce some notations useful in the following. We consider the online problem over a series of rounds. At each round t, an example feature $x_t \in \mathcal{X}$ is showed; a label $y_t \in \mathcal{Y}$ will be later observed. We denote the features space $\mathcal{X} \subset \mathbb{R}^d$, $\mathcal{Y} = \{-1, +1\}$ for binary labels and the prediction space $\widehat{\mathcal{Y}} = \mathbb{R}$. In the point of view of online density estimation, an algorithm $\mathcal{A}$ plays $\theta \in \mathbb{R}^d$, such that the probability of a label for an example x is

$$p(y \mid x, \theta) \coloneqq \frac{1}{1 + \exp(-y \cdot \theta^\top x)} = \sigma(y \cdot \theta^\top x)$$

with σ the sigmoid function $\sigma(u) \coloneqq e^u/(1 + e^u)$ for $u \in \mathbb{R}$. The loss at time t for the logistic model θ is the log-loss

$$\ell(\theta^\top x_t, y_t) = -\log p(y_t \mid x_t, \theta) = -\log \sigma(y_t \cdot \theta^\top x_t).$$

Equivalently, the algorithm $\mathcal{A}$ produces, for an example x, the prediction

$$\widehat{y} = \log \frac{p(+1 \mid x, \theta)}{p(-1 \mid x, \theta)} = \theta^\top x,$$

and suffers the logistic loss $\ell : \widehat{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}$ defined as

$$\ell(\widehat{y}, y) = \log\left(1 + \exp\left(-\widehat{y}y\right)\right) = -\log \sigma(\widehat{y}y).$$

Consider an individual sequence $(x_t, y_t)_t$ and the successive predictions $(\widehat{y}_t)_t$ produced by an algorithm $\mathcal{A}$, the regret against a comparator $\theta \in \mathbb{R}^d$ for n total rounds writes

$$\mathrm{Regret}_n(\theta) \coloneqq \sum_{t=1}^n \ell(\widehat{y}_t, y_t) - \sum_{t=1}^n \ell(\theta^\top x_t, y_t),$$

and the regret against the comparison class $\Theta \subseteq \mathbb{R}^d$ for total rounds n writes

$$\mathrm{Regret}_n \coloneqq \sum_{t=1}^n \ell(\widehat{y}_t, y_t) - \inf_{\theta \in \Theta} \sum_{t=1}^n \ell(\theta^\top x_t, y_t).$$

The goal is to obtain guarantees on the regret for any individual sequence, including in the worst-case scenario.

Assumption 5.1 (Bounded features)**.** The features are bounded *i.e.* $\|x_t\|_2 \leqslant R$ with some constant $R > 0$, for all t.

Assumption 5.2 (Comparison class)**.** We use the ℓ_2 ball of radius B centered at the origin as the comparison class, *i.e.* $\Theta = \mathscr{B}(\mathbb{R}^d, B) \coloneqq \{\theta \in \mathbb{R}^d : \|\theta\|_2 \leqslant B\}$ with some constant $B > 0$. Such Θ is compact and convex, it is sometimes called the feasible set.

About the order of BR. The results that we present in this section are valid for arbitrary parameters $B, R > 0$ and $d, n \geqslant 1$. In order to make these bounds more concrete, we now discuss some natural scaling for the norm BR, following Remark 2 of Mourtada and Gaïffas [2019].

Remark 5.1 (Parameter scaling)*.* In the case $n \gg d$, consider a random variable $X \in \mathbb{R}^d$ and denote $\Sigma = \mathbb{E}[XX^\top]$. Assume that Σ is well-conditioned, $c \coloneqq \|\Sigma\|_{\mathrm{op}} \|\Sigma^{-1}\|_{\mathrm{op}} = O(1)$; this means that X is approximately isotropic, and can be ensured in practice by rescaling covariates. Assume also the bounded leverage condition $\|\Sigma^{-1/2} X\| \leqslant \rho\sqrt{d}$ for some $\rho \geqslant 1$, and denote $\psi \coloneqq \|\theta\|_\Sigma = \mathbb{E}[\langle \theta, X\rangle^2]^{1/2}$ as the signal strength. Then,

$$\|\theta\| \, \|X\| \leqslant \|\Sigma^{-1/2}\|_{\mathrm{op}} \, \|\theta\|_\Sigma \, \|\Sigma^{1/2}\|_{\mathrm{op}} \|\Sigma^{-1/2} X\| \leqslant c^{1/2} \psi \rho \sqrt{d},$$

namely we have $BR \leqslant c^{1/2}\psi\rho\sqrt{d} = O(\sqrt{d})$.

5.2 Properties of the logistic function

The algorithms with better regret guarantees for online logistic regression often come from some more advanced studies on the logistic function. Denote $f_t(\theta) \coloneqq \ell(\theta^\top x_t, y_t) = -\log \sigma(y_t \cdot \theta^\top x_t)$ for $\theta \in \mathbb{R}^d$, the logistic loss function taking the

parameter θ, induced by (x_t, y_t). The function f_t is infinitely differentiable, its first and second derivatives write

$$\nabla f_t(\theta) = -y_t \sigma(-y_t x_t^\top \theta) x_t, \quad \nabla^2 f_t(\theta) = \sigma'(-y_t x_t^\top \theta) x_t x_t^\top.$$

In the following, we make some observations on the logistic function and point out the relationships between these properties and some known algorithms for online logistic regression. Denote θ_t the point around which we make the approximation.

- Under Assumption 5.1, f_t is R-Lipschitz, since $\|\nabla f_t(\theta)\| \leqslant R$ for any (x_t, y_t) and any θ. Under the same condition, ∇f_t is R^2-Lipschitz, *i.e.* f_t is R^2-smooth.

- The logistic loss function is convex, $\nabla^2 f_t(\theta) \succcurlyeq 0$ for any θ and any (x_t, y_t). Hence, for any fixed θ_t, its first order approximation around θ_t defined as

$$\tilde{f}_t : \theta \mapsto f_t(\theta_t) + \nabla f_t(\theta_t)^\top (\theta - \theta_t) \tag{5.1}$$

  is a lower-bound of f_t for any θ. This is the key property in the theoretical guarantees for Online Gradient Descent [Zinkevich, 2003].

- The logistic loss is $\exp(-BR)$-exp-concave on the ball $\mathscr{B}(\mathbb{R}^d, B)$ under Assumptions 5.1 and 5.2. The exp-concavity can be understood as a property stating that the Hessian is large in the direction of the gradient. Indeed, α-exp-concavity of a function f_t is equivalent to $\nabla^2 f_t(\theta) \succcurlyeq \alpha \nabla f_t(\theta) \nabla f_t(\theta)^\top$ for any θ; for the logistic function, this comes from $\sigma'(u) = \sigma(u)(1 - \sigma(u)) = \exp(-u)\sigma^2(u)$, hence $\sigma'(u) \geqslant \exp(-BR)\sigma^2(u)$ whenever $u \leqslant BR$. Online Newton Step [Hazan et al., 2007] relies on the following quadratic approximation of the logistic loss

$$\tilde{f}_t : \theta \mapsto f_t(\theta_t) + \nabla f_t(\theta_t)^\top (\theta - \theta_t) + \frac{\beta}{2} (\theta - \theta_t)^\top \nabla f_t(\theta_t) \nabla f_t(\theta_t)^\top (\theta - \theta_t) \tag{5.2}$$

  with $\beta \leqslant \exp(-BR)$ the exp-concavity parameter. Such $\tilde{f}_t$ is showed [Hazan et al., 2007] to be a lower-bound of f_t for any θ and any $\theta_t \in \mathscr{B}(\mathbb{R}^d, B)$. Also relying on the exp-concavity are the Bayesian exponential weighted algorithms as exposed in Hazan et al. [2007]; Kakade and Ng [2005].

- While the self-concordance [Nesterov, 2003] is a property on the control of the third derivative by the second derivative, namely $|f'''| \leqslant 2(f'')^{3/2}$, the logistic function is *not* self-concordant. Bach [2010] introduced "generalized R-self-concordance" defined by $|f'''| \leqslant R f''$, with some $R > 0$. For the logistic function, its generalized self-concordance comes from a more precise study on the derivatives of the sigmoid function, $\sigma''(u) = (1 - 2\sigma(u))\sigma'(u)$, hence $|\sigma''(u)| \leqslant \sigma'(u)$ for any $u \in \mathbb{R}$. A consequence of the generalized R-self-concordance (under Assumption 5.1) of the logistic function states that

$$\exp(-R\|\theta - \theta_t\|) \nabla^2 f_t(\theta_t) \preccurlyeq \nabla^2 f_t(\theta) \preccurlyeq \exp(R\|\theta - \theta_t\|) \nabla^2 f_t(\theta_t) \tag{5.3}$$

  for any θ and any θ_t. The property (5.3) controls the variation of the Hessian around θ_t, hence it can deduce the local strong convexity around θ_t. Moreover, such bounds (5.3) on second derivatives can be applied on the function f_t to

deduce bounds of $f_t(\theta)$ in expression of $f_t(\theta_t)$. Especially, the approximation function $\tilde{f}_t$ defined by

$$\tilde{f}_t : \theta \mapsto f_t(\theta_t) + \nabla f_t(\theta_t)^\top (\theta - \theta_t) + \frac{\exp(-R\|\theta - \theta_t\|) + R\|\theta - \theta_t\| - 1}{R^2\|\theta - \theta_t\|^2} (\theta - \theta_t)^\top \nabla^2 f_t(\theta_t)(\theta - \theta_t) \quad (5.4)$$

is showed [Bach, 2010, Proposition 1] to be a lower-bound of $f_t(\theta)$ for any θ and any θ_t, again under Assumption 5.1. In the batch learning setting, Sample Minmax Predictor [Mourtada and Gaïffas, 2019] in the case of ridge-penalized logistic regression also relies on the property (5.3) in their analysis for the excess risk.

- Recent work of Jézéquel et al. [2020] introduced a new quadratic approximation of the logistic function

$$\tilde{f}_t : \theta \mapsto f_t(\theta_t) + \nabla f_t(\theta_t)^\top (\theta - \theta_t) + \frac{\exp(y_t \theta_t^\top x_t)}{2(1 + BR)} (\theta - \theta_t)^\top \nabla f_t(\theta_t) \nabla f_t(\theta_t)^\top (\theta - \theta_t), \quad (5.5)$$

and showed that under Assumption 5.1, $\tilde{f}_t$ is a lower-bound of f_t on the ball $\mathscr{B}(\mathbb{R}^d, B)$, for any θ_t. The proof of this lower-bound relies on a study of the logistic function which is more sophisticated than the exp-concavity: instead of a fixed coefficient β being uniform and exponential in BR in (5.2), the second-order coefficient in (5.5) adaptively depends on $y_t \theta_t^\top x_t$. Recent works [Agarwal et al., 2021; Jézéquel et al., 2021] also rely on the lower-bound (5.5) while generalizing it to the multi-class case.

- Also, the logistic function $\ell(\widehat{y}, y) = -\log \sigma(\widehat{y}y)$ is 1-mixable, in the sense that, for any probability distribution π over $\widehat{\mathcal{Y}} = \mathbb{R}$, there exists a prediction $\widehat{y}_\pi$ such that, for any $y \in \{-1, +1\}$,

$$\mathbb{E}_{\widehat{y} \sim \pi} \left[\exp(-\eta \ell(\widehat{y}, y)\right] \leqslant \exp(-\eta \ell(\widehat{y}_\pi, y)) \quad (5.6)$$

with $\eta = 1$. To see this for the logistic function, we construct

$$\widehat{y}_\pi = \sigma^+ \left(\mathbb{E}_{\widehat{y} \sim \pi}[\sigma(\widehat{y})]\right)$$

where $\sigma^+ : (0, 1) \to \mathbb{R}$ is the inverse of the sigmoid function, $\sigma^+(p) = \log \frac{p}{1-p}$, hence $\sigma^+(\sigma(u)) = u$ for any $u \in \mathbb{R}$. Such $\widehat{y}_\pi$ satisfies (5.6) with equality. Bayesian methods Foster et al. [2018] and Jézéquel et al. [2021] rely on the 1-mixability of the logistic function.

5.3 Proper algorithms

We propose a literature review of online logistic regression methods. While some of presented methods also apply to other problems, here we only present them as solutions to the online logistic regression, and we highlight the necessary conditions in each case. Within proper algorithms, there are Follow-The-Leader methods, gradient-based methods and Bayesian methods.

Follow-The-Leader methods

Let us start with probably the most intuitive idea: gradient-free methods. *Follow-The-Leader* (FTL) consists in choosing, at time t, the parameter θ that would have been the best to use on the previous $t-1$ rounds in hindsight, *i.e.* to choose

$$\theta_t = \operatorname*{argmin}_{\theta \in \mathbb{R}^d} L_{t-1}(\theta) \quad \text{with} \quad L_{t-1}(\theta) \coloneqq \sum_{s=1}^{t-1} \ell_s(\theta).$$

However, the regret of this strategy can be linear in the number of rounds, considering for example the case where $d = 1$, $x_t = 1$ and y_t alternate between -1 and $+1$.

Follow-The-Regularized-Leader (FTRL) is introduced in order to stabilize the FTL solution, where the function to minimize is given with an additional regularization term on θ, *i.e.* the algorithm shall choose at time t,

$$\theta_t = \operatorname*{argmin}_{\theta \in \mathbb{R}^d} \left\{ L_{t-1}(\theta) + \operatorname{pen}(\theta) \right\},$$

where the penalization term $\operatorname{pen}(\theta)$ could be in ℓ_1, ℓ_2 or other forms. For example, in a COLT Open Problem 2012, in $d = 1$ dimension and $x_t \in \{-R, 0, +R\}$ setting, McMahan and Streeter [2012] used FTRL with a beta regularizer

$$\operatorname{pen}(\theta) = c(\theta, \lambda, \lambda) \quad \text{with} \quad c(\theta, N, P) \coloneqq P \log(1 + \exp(-\theta)) + N \log(1 + \exp(\theta))$$

with a constant $\lambda > 0$. They showed that $O(\sqrt{BR} + \log(n))$ regret is granted with $\lambda = \sqrt{2/BR}$. However, it was not clear whether this result is generalizable to larger dimensions and more general features.

Another variant of FTL, *Follow-The-Approximate-Leader* (FTAL), consists in choosing

$$\theta_t = \operatorname*{argmin}_{\theta \in \mathbb{R}^d} \tilde{L}_{t-1}(\theta) \quad \text{with} \quad \tilde{L}_{t-1}(\theta) \coloneqq \sum_{s=1}^{t-1} \tilde{\ell}_s(\theta),$$

where each $\tilde{\ell}_s$ is an approximation function of the true loss function ℓ_s. Indeed, several gradient methods (OGD, ONS) could be seen as FTAL methods [Hazan et al., 2007; Zinkevich, 2003], see details in the following paragraph. Interestingly, sub-linear or logarithmic regret upper-bounds are granted with these methods.

Gradient-based methods

Online gradient descent (OGD) for online logistic regression was first reported in Zinkevich [2003]. It starts with an arbitrary $\theta_1 \in \Theta$, then successively updates the predictions by

$$\theta_t = \operatorname{Proj}_\Theta(\theta_{t-1} - \eta_t \nabla \ell_{t-1}(\theta_{t-1}))$$

with $\eta_t > 0$ the step-size parameter at time t, and $\operatorname{Proj}_\Theta$ the Euclidean projector operator onto the compact convex feasible set Θ. The regret analysis of OGD is based on the convexity of the logistic function, and on the fact that a convex function is lower-bounded by its tangent hyperplans, see Equation (5.1). This results in a regret upper-bound $O(BR\sqrt{n})$ with the choice of step-sizes $\eta_t = B/(R\sqrt{t})$, as well

as a regret lower-bound $\Omega(BR\sqrt{n})$ in the worst case [Hazan, 2019] which is sub-linear in the number of rounds n. OGD can also be seen[2] as choosing at time t,

$$\theta_t = \operatorname*{argmin}_{\theta\in\Theta} \left\{ \nabla\ell_{t-1}(\theta_{t-1})^\top(\theta - \theta_{t-1}) + \frac{1}{2\eta_t}\|\theta - \theta_{t-1}\|^2 \right\}.$$

OGD results in $O(nd)$ computational complexity in time for n total rounds, plus the complexity costs in the projection steps.

Then, *Online Newton Step* (ONS) was proposed and analyzed in Hazan et al. [2007]. It starts with an arbitrary $\theta_1 \in \Theta$, then uses the predictions

$$\theta_t = \mathrm{Proj}_{\Theta,A_{t-1}}(A_{t-1}^{-1}b_{t-1})$$

with

$$A_t \coloneqq \sum_{s=1}^{t} \nabla\ell_s\nabla\ell_s^\top, \quad b_t \coloneqq \sum_{s=1}^{t} \nabla\ell_s\nabla\ell_s^\top\theta_s - \frac{1}{\beta}\nabla\ell_s, \quad \text{where} \quad \nabla\ell_t \coloneqq \nabla\ell_t(\theta_t)$$

with $\beta = \frac{1}{2}\min\{\frac{1}{4BR}, \exp(-BR)\}$ as in (5.2), and $\mathrm{Proj}_{\Theta,A_{t-1}}$ the projector operator onto the compact convex feasible set Θ under the norm induced by A_{t-1}. The regret analysis of ONS is based on the fact that the logistic loss function is R-Lipschitz and $\exp(-BR)$-exp-concave on the bounded feasible set, implying the quadratic lower-bound (5.2) of the logistic function. The regret of ONS admits an upper-bound [Hazan et al., 2007] in $O(\exp(BR)d\log(n))$, which is logarithmic in the number of rounds n, but with an exponential factor in BR due to the exp-concavity coefficient. ONS can be seen as running FTAL [Hazan et al., 2007], the approximation function being the quadratic lower-bound function (5.2) *i.e.* choosing at time t

$$\theta_t = \operatorname*{argmin}_{\theta\in\Theta} \tilde{L}_{t-1}(\theta) \quad \text{with} \quad \tilde{L}_{t-1}(\theta) \coloneqq \sum_{s=1}^{t-1} \tilde{\ell}_s(\theta),$$

where

$$\tilde{\ell}_t(\theta) \coloneqq \ell_t(\theta_t) + \nabla\ell_t^\top(\theta - \theta_t) + \frac{\beta}{2}(\theta - \theta_t)^\top\nabla\ell_t\nabla\ell_t^\top(\theta - \theta_t).$$

At each step the computational cost is $O(d^2)$ as we need to iteratively update A_t^{-1}, thanks to the matrix inversion lemma; the total computational complexity is $O(nd^2)$ in time for n total rounds, plus the complexity costs in the projection steps; and $O(d^2)$ space in memory is needed.

However, under the proper setting and by constructing adversary examples, Hazan et al. [2014] showed that in the regime $n = O(\exp(B))$, for one-dimensional inputs the regret is at least $\Omega(B^{2/3}n^{1/3})$ in the worst case [Hazan et al., 2014, Corollary 7]; and for any larger dimension $d \geqslant 2$, $\Omega(\sqrt{Bn})$ regret in the worst case [Hazan et al., 2014, Corollary 8]. Therefore, any regret bound of form $O(B\log n)$ is impossible for proper algorithms.

Bayesian proper method

Kakade and Ng [2005] was a first result suggesting that a regret of $O(d\log(n/d))$ is achievable for online logistic regression, by analyzing the following Bayesian Model

[2] An excellent post on FTRL-version and OMD-version of OGD https://www.timvanerven.nl/blog/ftrl-vs-omd/.

Averaging with Gaussian prior $p_0 = \mathcal{N}(\theta; 0, \nu^2 I_d)$. It predicts at time t,

$$\widehat{y}_t = \left(\int_\Theta \theta \cdot p_{t-1}(\mathrm{d}\theta) \right)^\top x_t$$

and updates the distribution p_t by

$$p_t(\theta) \propto \exp\left(- \sum_{s=1}^{t} \ell(\theta^\top x_s, y_s) \right) p_0(\theta).$$

For this procedure, with variational techniques, Kakade and Ng [2005] showed a regret upper-bound in $O(d \log(Bn/d))$, plus an additional penalty term that depends on the prior, as well as on the squared ℓ_2-norm of the best comparator $\theta^\star$ [Kakade and Ng, 2005, Theorem 2.2]. It is not clear how to efficiently compute such a predictor.

5.4 Improper algorithms

Within improper algorithms, there are Bayesian and non-Bayesian methods.

Bayesian improper methods

Foster et al. [2018] used a uniform prior over Θ in Bayesian mixture with exponential weights Vovk's Aggregation Algorithm [Vovk, 1990]. In the case of online binary logistic regression and without smoothing, the algorithm of Foster et al. [2018] initializes p_0 as the uniform distribution over Θ, successively predicts

$$\widehat{y}_t = \sigma^+(\mathbb{E}_{\theta \sim p_{t-1}}[\sigma(\theta^\top x_t)])$$

where $\sigma^+ : (0,1) \to \mathbb{R}$ is the inverse of the sigmoid function, $\sigma^+(p) = \log \frac{p}{1-p}$, and updates the distribution p_t over Θ by

$$p_t(\theta) \propto \exp\left(- \sum_{s=1}^{t} \ell(\theta^\top x_s, y_s) \right).$$

With this procedure, the regret has an upper-bound in $O(d \log(BRn/d))$. This work can be generalized into m-class softmax regression, with a regret in $O(md \log(n/md))$. However, this algorithm is computationally expensive, as it relies on Monte Carlo methods and the log-concavity of the probability distributions p_t: its computational complexity is in $O(B^6 n^{12}(Bn + d)^{12})$.

Recent work of Shamir [2020] investigated the achievable minimax regret rate of online logistic regression from the scope of information theory. Using the redundancy capacity Theorem, Shamir [2020] proposed regret lower- and upper-bounds under Bayesian setting. Among other bounds, in particular, for "smaller d", with $\|\theta\|_2 \leqslant B$,

$$\frac{d}{2} \log \frac{n}{d^2} \leqslant \mathrm{Regret}_n \leqslant \frac{d}{2} \log \frac{B^2 n}{d}.$$

In a similar direction, another recent work Jacquet et al. [2021] computed the precise minimax regret in $O(\frac{d}{2} \log(2n/\pi))$ in the case of categorical feature values. Note that these works are results on the characterization of the minimax regret, and do not provide an explicit algorithm.

Finally, a recent work Jézéquel et al. [2021] focus on the multi-class logistic regression, proposing Gaussian Aggregating Forecaster (GAF) which relies on the quadratic approximation (5.5) and on the Bayesian mixture with Gaussian distributions.

Non-Bayesian improper algorithms

As a non-Bayesian improper algorithm, AIOLI [Jézéquel et al., 2020] is based on a new quadratic lower bound (5.5) of the logistic function, with adaptive curvature which is better than exp-concavity. It also relies on a Vovk-Azoury-Warmuth-inspired [Azoury and Warmuth, 2001; Vovk, 2001] and SMP-inspired [Mourtada and Gaïffas, 2019] regularization, which is a regularization by samples. AIOLI computes the following estimator at time t,

$$\theta_t = \operatorname*{argmin}_{\theta\in\mathbb{R}^d} \left\{ \tilde{L}_{t-1}(\theta) + \ell(\theta^\top x_t, +1) + \ell(\theta^\top x_t, -1) + \lambda \left\|\theta\right\|^2 \right\}$$

with $\tilde{L}_{t-1}(\theta) \coloneqq \sum_{s=1}^{t-1} \tilde{\ell}_s(\theta)$, where the surrogate loss function is defined by

$$\tilde{\ell}_t(\theta) \coloneqq \ell_t(\theta_t) + \nabla\ell_t(\theta_t)^\top(\theta-\theta_t) + \frac{\exp(y_t\theta_t^\top x_t)}{2(1+BR)}(\theta-\theta_t)^\top \nabla\ell_t(\theta_t)\nabla\ell_t(\widehat{\theta}_t)^\top(\theta-\theta_t)$$

as in (5.5). The regret of AIOLI is upper-bounded by $O(dBR\log(BRn))$ with a computational cost in $O(nd^2 + nBR\log(n))$. Recent work FOLKLORE [Agarwal et al., 2021] generalizes the quadratic lower bound (5.5) to the multi-class case and proposes an improper algorithm with similar regret and computational cost compared to AIOLI.

However, the regret upper-bound of AIOLI might be seen as suboptimal for the following reasons: 1) from Remark 5.1, BR scales as $O(\sqrt{d})$, then regret in $O(d\sqrt{d})$ of AIOLI is not optimal; 2) consider the online-to-batch conversion on AIOLI, the algorithm's expected bound of excess risk is in $O(dBR\log(BRn))$; apart from the factor $\log n$, unavoidable in the online setting, this bound is of a factor BR worse than the expected excess risk of SMP [Mourtada and Gaïffas, 2019].

Sample Minmax Predictor (SMP) [Mourtada and Gaïffas, 2019] is an algorithm for the batch setting. In particular, for binary logistic regression with $(X_i, Y_i)_{i=1,\dots,n}$ i.i.d. data and $\lambda > 0$ the penalization parameter, the ridge-penalized SMP computes the estimator

$$\widehat{\theta}^z_{\lambda,n} = \operatorname*{argmin}_{\theta\in\mathbb{R}^d} \left\{ \frac{1}{n+1}\left(\sum_{i=1}^n \ell(\langle\theta, Z_i\rangle) + \ell(\langle\theta, z\rangle) \right) + \lambda\left\|\theta\right\|^2 \right\}$$

where we denote $Z_i = -Y_iX_i$, and $z = (x, y)$ a virtual sample. Then SMP predicts the conditional probability, for any $(x, y) \in \mathbb{R}^d \times \{-1, +1\}$, by

$$p(y \mid x) = \frac{\sigma\big(y\langle\widehat{\theta}^{(x,y)}_{\lambda,n}, x\rangle\big) e^{-\lambda\|\widehat{\theta}^{(x,y)}_{\lambda,n}\|^2}}{\sigma\big(\langle\widehat{\theta}^{(x,+1)}_{\lambda,n}, x\rangle\big) e^{-\lambda\|\widehat{\theta}^{(x,+1)}_{\lambda,n}\|^2} + \sigma\big(-\langle\widehat{\theta}^{(x,-1)}_{\lambda,n}, x\rangle\big) e^{-\lambda\|\widehat{\theta}^{(x,-1)}_{\lambda,n}\|^2}}.$$

The expected excess risk of ridge-penalized SMP for logistic regression is in $O((d + B^2R^2)/n)$. But it was not clear whether this procedure could be applied in the online setting and how.

An open problem. As far as we know, there is no algorithm with an optimal regret guarantee for the problem of online logistic regression. Bayesian algorithms [Foster et al., 2018; Jézéquel et al., 2021] enjoy better regret upper-bounds but suffer from computational complexity; non-Bayesian algorithms [Agarwal et al., 2021; Hazan et al., 2007; Jézéquel et al., 2020], less computationally demanding, fail to provide similar regret upper-bounds. We believe that there is a room between them, and that an online version of SMP [Mourtada and Gaïffas, 2019] or SMP-inspired methods, which would be some non-Bayesian improper algorithms, could achieve this goal. In this direction, we investigate some non-Bayesian improper algorithms with realistic computational complexity, and their regret upper-bounds, even though we do not improve the state-of-the-art of this problem.

We summarize the regret upper-bounds and the computational costs of these algorithms in Table 5.1 below.

Table 5.1 – Regret upper-bounds and computational costs in $O(\cdot)$ of several algorithms for online binary logistic regression: OGD [Zinkevich, 2003], ONS [Hazan et al., 2007], Foster [Foster et al., 2018], GAF [Jézéquel et al., 2021], AIOLI [Jézéquel et al., 2020], FOLKLORE [Agarwal et al., 2021]. AOSMP stands for Approximated One-Step Minmax Predictor (Algorithm 6 in this work), C_{GD} stands for the computational cost of the gradient descent for finding $L_t^{y\star}$. Algorithms with * also handle the multi-class case.

Algorithm	Type	Regret upper-bound	Computational cost
OGD	Proper	$BR\sqrt{n}$	nd
ONS	Proper	$de^{BR}\log(n)$	nd^2
Foster	Improper, Bayesian	$d\log(BRn)$	$B^6n^{12}(Bn+d)^{12}$
GAF*	Improper, Bayesian	$dBR\log(n)+dB^2$	nd^2+n^4
AIOLI	Improper, non-Bayesian	$dBR\log(BRn)$	$nd^2+nBR\log(n)$
FOLKLORE*	Improper, non-Bayesian	$dBR\log(n)$	$nd^2+nBR\log(n)$
AOSMP	Improper, non-Bayesian	$dBR\log(n)+B^2R^2$	nd^2+nC_{GD}

5.5 OSMP: One-Step Minmax Predictor

In this section, we explore a SMP-inspired [Mourtada and Gaïffas, 2019] predictor for online logistic regression, and propose a study on the corresponding regret.

5.5.1 Algorithm

Under the setting of online binary logistic regression, at time step t, we have seen $(x_1, y_1), ..., (x_{t-1}, y_{t-1})$, predicted $\widehat{y}_1, ..., \widehat{y}_{t-1}$; now we are provided with x_t, and we need to propose our prediction $\widehat{y}_t$. Consider the "λ-ridge penalized regret" at time t with some $\lambda > 0$,

$$\sum_{s=1}^{t} \ell(\widehat{y}_s, y_s) - \left(\sum_{s=1}^{t} \ell(\theta^\top x_s, y_s) + \lambda \|\theta\|^2 \right). \tag{5.7}$$

As for the prediction at time t, the *One-Step Minmax Predictor* (OSMP) aims to minimize this quantity with respect to the worst case over all possible values of label $y_t \in \{-1, 1\}$ and over all possible "adversary parameter" θ. Let us study this predictor under Assumptions 5.1 and 5.2.

Definition 5.2 (OSMP)**.** Given $\lambda > 0$ the ridge penalization parameter. Denote $\Delta \subseteq \mathbb{R}^d$ the search space for the parameter θ, such that $\Theta \subseteq \Delta$ and such that Δ is convex. OSMP computes the estimator

$$\widehat{y}_t = \operatorname*{argmin}_{\widehat{y} \in \mathbb{R}} \sup_{y_t \in \{-1,1\}} \sup_{\theta \in \Delta} \left\{ \widehat{L}_{t-1} + \ell(\widehat{y}, y_t) - \left(L_{t-1}(\theta) + \ell(\theta^\top x_t, y_t) + \lambda \|\theta\|^2 \right) \right\} \tag{5.8}$$

with notations

$$\widehat{L}_{t-1} \coloneqq \sum_{s=1}^{t-1} \ell(\widehat{y}_s, y_s), \quad L_{t-1}(\theta) \coloneqq \sum_{s=1}^{t-1} \ell(\theta^\top x_s, y_s),$$

where $\widehat{L}_{t-1}$ stands for actual sum of losses suffered by successively proposed predictions $(\widehat{y}_s)$ until time $t-1$; L_{t-1} is the function taking parameter θ, corresponding to the sum of losses until time $t-1$ suffered by a fixed parameter θ.

OSMP can be seen as a minimax strategy and a very pessimistic one: it always considers the worst possible scenario, in terms of the label y_t and the adversary parameter $\theta \in \Delta$, measured by the λ-ridge penalized regret at time t; from this, OSMP aims to find the best possible (one-step) predictor of time t against the worst scenario.

In Equation (5.8), we may search θ exactly in the same comparison class $\Delta = \Theta$; using a larger search space $\Delta \supseteq \Theta$ is also possible, meaning that our minimax strategy is even more pessimistic.

Remark 5.3. OSMP defined in Equation (5.8) is equivalent to predict probabilities $p(+1 \mid x_t) = \widehat{p}_t$ and $p(-1 \mid x_t) = 1 - \widehat{p}_t$ with

$$\widehat{p}_t = \operatorname*{argmin}_{p \in (0,1)} \sup_{y_t \in \{-1,1\}} \sup_{\theta \in \Delta} \left\{ \widehat{L}_{t-1} - \log p - \left(L_{t-1}(\theta) + \ell(\theta^\top x_t, y_t) + \lambda \|\theta\|^2 \right) \right\} \tag{5.9}$$

using $\widehat{p}_t = \sigma(\widehat{y}_t)$.

Notations

Let us introduce some useful notations. Denote

$$L_{\lambda,t}(\theta) \coloneqq L_t(\theta) + \lambda \|\theta\|^2, \quad L^\star_{\lambda,t} \coloneqq \inf_{\theta \in \Delta} L_{\lambda,t}(\theta), \tag{5.10}$$

and

$$L^y_{\lambda,t}(\theta) \coloneqq \ell(\theta^\top x_t, y) + L_{\lambda,t-1}(\theta) \quad \text{for} \quad y \in \{-1, +1\},$$

and their infimum

$$L^{y\star}_{\lambda,t} \coloneqq \inf_{\theta \in \Delta} \left\{ \ell(\theta^\top x_t, y) + L_{\lambda,t-1}(\theta) \right\} \quad \text{for} \quad y \in \{-1, +1\}.$$

We also introduce the notations

$$\theta_t \coloneqq \operatorname*{argmin}_{\theta \in \Delta} L_{\lambda,t}(\theta), \quad \theta_t^{(x,y)} \coloneqq \operatorname*{argmin}_{\theta \in \Delta} \left\{ \ell(\theta^\top x, y) + L_{\lambda,t-1}(\theta) \right\} \text{ for } y \in \{-1, +1\}.$$

We remark that θ_t as well as $\theta_t^{(x,y)}$ exist and are unique, as the minimizers of strongly convex functions over the convex set Δ.

Alternative expression and explicit algorithm

Relying on the specificities of the log-loss, we deduce the following alternative expression of OSMP, which allows the explicit algorithm computing OSMP.

Lemma 5.4. *OSMP as defined in Equation (5.8) is equivalent to*

$$\widehat{y}_t = -L_{\lambda,t}^{+1\star} + L_{\lambda,t}^{-1\star}. \tag{5.11}$$

Proof. Since $\widehat{L}_{t-1}$ does not depend on y_t nor on θ, we can drop it from argmin; also, we can swap $\sup_{y_t}$ and $\sup_\theta$, and rewrite the Equation (5.8) as

$$\widehat{y}_t = \operatorname*{argmin}_{\widehat{y}\in\mathbb{R}} \sup_{\theta\in\Delta} \left\{ \sup_{y_t\in\{-1,1\}} \left\{ \ell(\widehat{y}, y_t) - \ell(\theta^\top x_t, y_t) \right\} - L_{\lambda,t-1}(\theta) \right\}$$

where we notice that

$$\begin{aligned} \sup_{y_t\in\{-1,1\}} \left\{ \ell(\widehat{y}, y_t) - \ell(\theta^\top x_t, y_t) \right\} &= \sup_{y_t\in\{-1,1\}} \log \frac{\sigma(y_t \theta^\top x_t)}{\sigma(y_t \widehat{y})} \\ &= \max \left\{ \log \frac{\sigma(\theta^\top x_t)}{\sigma(\widehat{y})}, \quad \log \frac{\sigma(-\theta^\top x_t)}{\sigma(-\widehat{y})} \right\}. \end{aligned}$$

Recall that $\sigma(-\widehat{y}) = 1 - \sigma(\widehat{y})$ for the sigmoid function, hence

$$\begin{aligned} \widehat{y}_t = \operatorname*{argmin}_{\widehat{y}\in\mathbb{R}} \sup_{\theta\in\Delta} \Bigg\{ \max \Bigg\{ & \log \frac{\sigma(\theta^\top x_t) \exp(-L_{\lambda,t-1}(\theta))}{\sigma(\widehat{y})}, \\ & \log \frac{\sigma(-\theta^\top x_t) \exp(-L_{\lambda,t-1}(\theta))}{1 - \sigma(\widehat{y})} \Bigg\} \Bigg\} \end{aligned}$$

then

$$\begin{aligned} \widehat{y}_t = \operatorname*{argmin}_{\widehat{y}\in\mathbb{R}} \max \Bigg\{ & \frac{\sup_{\theta\in\Delta} \left\{ \sigma(\theta^\top x_t) \exp\left(-L_{\lambda,t-1}(\theta)\right) \right\}}{\sigma(\widehat{y})}, \\ & \frac{\sup_{\theta\in\Delta} \left\{ \sigma(-\theta^\top x_t) \exp\left(-L_{\lambda,t-1}(\theta)\right) \right\}}{1 - \sigma(\widehat{y})} \Bigg\}. \end{aligned}$$

Consider the fact that

$$\widehat{p} \coloneqq \operatorname*{argmin}_{p\in[0,1]} \max \left\{ \frac{a}{p}, \frac{b}{1-p} \right\} = \frac{a}{a+b}, \quad \text{for} \quad a, b > 0$$

as to solve the minimax problem of two parametrized linear functions, we have

$$\sigma(\widehat{y}_t) = \frac{\sup_{\theta\in\Delta} \left\{ \sigma(\theta^\top x_t) \exp\left(-L_{\lambda,t-1}(\theta)\right) \right\}}{\sup_{\theta\in\Delta} \left\{ \sigma(\theta^\top x_t) \exp\left(-L_{\lambda,t-1}(\theta)\right) \right\} + \sup_{\theta\in\Delta} \left\{ \sigma(-\theta^\top x_t) \exp\left(-L_{\lambda,t-1}(\theta)\right) \right\}}$$

and hence the expression of $\widehat{y}_t$ as in the statement of the Lemma. □

Now we are ready to summarize the computation of OSMP in Algorithm 5.

Remark 5.5. OSMP predicts probabilities $p(+1 \mid x_t) = \widehat{p}_t$ and $p(-1 \mid x_t) = 1 - \widehat{p}_t$ with

$$\widehat{p}_t = \frac{\exp(-L_{\lambda,t}^{+1\star})}{\exp(-L_{\lambda,t}^{+1\star}) + \exp(-L_{\lambda,t}^{-1\star})} \tag{5.12}$$

for $y \in \{-1, +1\}$, by remarking that $\widehat{p}_t = \sigma(\widehat{y}_t)$.

Algorithm 5 OSMP description.

Inputs: Parameters $\lambda > 0$, $n, d \geqslant 1$, constants $B, R > 0$, search space Δ
Initialize: Function $L_{\lambda,0}(\theta) = \lambda \|\theta\|^2$
for $t = 1, \ldots, n$ **do**
  Receive x_t
  Compute $L_{\lambda,t}^{y\star} \leftarrow \inf_{\theta \in \Delta} \left\{ \ell(\theta^\top x_t, y) + L_{\lambda,t-1}(\theta) \right\}$ for $y = -1$ and $y = +1$
  Predict $\widehat{y}_t \leftarrow -L_{\lambda,t}^{+1\star} + L_{\lambda,t}^{-1\star}$
  Receive y_t
  Update function $L_{\lambda,t}(\theta) = L_{\lambda,t-1}(\theta) + \ell(\theta^\top x, y_t)$
end for

Equation (5.12) is analogous to Equation (47) (which comes from its Theorem 2) of Mourtada and Gaïffas [2019] for the Sample Minmax Predictor under the batch setting. In place of the penalization $\exp\big(-\lambda \|\theta_t^{(x_t,y)}\|^2\big)$ of their Equation (47), in our case we have $\exp\big(-L_{\lambda,t-1}(\theta_t^{(x_t,y)})\big)$ in each term of Equation (5.12), as the penalization for the loss until time $t-1$, precisely due to the online setting.

Remark 5.6. Recall that the FTRL consists in using $\langle \theta_{t-1}, x_t \rangle$ as the prediction at time t, and that the oracle would use $\langle \theta_t, x_t \rangle$ as the solution for the penalized regret minimization problem until time t. Denote $\theta_t^{-y_t} = \theta_t^{(x_t,-y_t)}$. We remark for OSMP

$$y_t \langle \theta_t^{-y_t}, x_t \rangle \leqslant y_t \widehat{y}_t = L_{\lambda,t}^{-y_t\star} - L_{\lambda,t}^{\star} \leqslant y_t \langle \theta_t, x_t \rangle ,$$

and we can understand $\widehat{y}_t$ from Equation (5.11) as a linear combination of $\langle \theta_t^{-y_t}, x_t \rangle$ and $\langle \theta_t, x_t \rangle$. Namely, there is $0 \leqslant \alpha \leqslant 1$ such that

$$\widehat{y}_t y_t = \alpha \cdot \langle \theta_t, x_t \rangle + (1-\alpha) \cdot \langle \theta_t^{-y_t}, x_t \rangle$$

with

$$\alpha = \frac{L_{\lambda,t}^{-y_t\star} - L_{\lambda,t}^{\star} - y_t \langle \theta_t^{-y_t}, x_t \rangle}{y_t \langle \theta_t, x_t \rangle - y_t \langle \theta_t^{-y_t}, x_t \rangle}.$$

Depending on the possible values for $y_t \in \{-1, +1\}$, the oracle prediction for $\widehat{y}_t$, as the solution of the penalized regret minimization problem until time t, is either $\langle \theta_t, x_t \rangle$ or $\langle \theta_t^{-y_t}, x_t \rangle$. Interestingly, $\widehat{y}_t$ is an average of these two potential oracle solutions, weighted by their "performance" measured by α. Note that this formulation cannot be used to compute $\widehat{y}_t$ for prediction, as it requires the knowledge of y_t (the environment only reveals true y_t after the prediction).

5.5.2 Analysis of regret and instant regret

We move to the analysis of the regret of the successive predictions $(\widehat{y}_t)_t$ produced by OSMP (Algorithm 5). Recall that for a sequence $(x_t, y_t)_t$ and some predictions $(\widehat{y}_t)_t$, the regret until round n against θ writes

$$\mathrm{Regret}_n(\theta) = \sum_{t=1}^{n} \ell(\widehat{y}_t, y_t) - \sum_{t=1}^{n} \ell(\theta^\top x_t, y_t).$$

Note that $\sum_{t=1}^{n} \ell(\theta^\top x_t, y_t) = L_{\lambda,n}(\theta) - \lambda \|\theta\|^2$. Since we are under Assumption 5.2, and we assume $\Theta \subseteq \Delta$, for any comparison parameter $\theta \in \Theta$,

$$\mathrm{Regret}_n(\theta) \leqslant \sum_{t=1}^{n} \left(\ell(\widehat{y}_t, y_t) - \inf_{\theta' \in \Delta} L_{\lambda,t}(\theta') + \inf_{\theta' \in \Delta} L_{\lambda,t-1}(\theta') \right) + \lambda \|\theta\|^2 ,$$

where we can simply upper-bound the last term $\lambda \|\theta\|^2$ by λB^2. Then it remains to upper-bound the sum. We introduce *instant regret* as the component term in the sum of this expression.

Definition 5.7 (Instant regret)**.** With some abuse of language, we call "instant regret" at time t the quantity defined by

$$\widehat{r}_t \coloneqq \ell(\widehat{y}_t, y_t) - L^\star_{\lambda,t} + L^\star_{\lambda,t-1}, \tag{5.13}$$

where $L^\star_{\lambda,t}$ is defined in Equation (5.10).

The following Lemma is analogous to Theorem 2 of Mourtada and Gaïffas [2019], in particular its Equation (15) on the upper-bound of the excess risk.

Lemma 5.8. *The instant regret defined in (5.13) also writes*

$$\widehat{r}_t = \log\left(\exp(-L^{+1\star}_{\lambda,t} + L^\star_{\lambda,t-1}) + \exp(-L^{-1\star}_{\lambda,t} + L^\star_{\lambda,t-1})\right). \tag{5.14}$$

Proof. We plug the expression (5.11) of $\widehat{y}_t$ into the definition of $\widehat{r}_t$ to obtain

$$\begin{aligned}\widehat{r}_t &= \log\left(1 + \exp\left(-y_t(-L^{+1\star}_{\lambda,t} + L^{-1\star}_{\lambda,t})\right)\right) - L^\star_{\lambda,t} + L^\star_{\lambda,t-1} \\ &= \log\left(1 + \exp\left(L^{y_t\star}_{\lambda,t} - L^{-y_t\star}_{\lambda,t}\right)\right) - L^\star_{\lambda,t} + L^\star_{\lambda,t-1},\end{aligned}$$

then, we notice that $L^{y_t}_{\lambda,t} = L_{\lambda,t}$ hence $L^{y_t\star}_{\lambda,t} = L^\star_{\lambda,t}$, and we obtain the statement in Lemma. □

Next we need to upper-bound the sum of the instant regrets to obtain an upper-bound of the regret on the OSMP.

5.5.3 An instant regret upper-bound via generalized self-concordance

From now on, we use the search space $\Delta = \mathbb{R}^d$ as this is technically practical. Denote $\|x\|_H \coloneqq \langle Hx, x\rangle^{1/2}$ for any $x \in \mathbb{R}^d$ and $H \in \mathbb{R}^{d\times d}$ symmetric positive matrix. The following Proposition provides an upper-bound of the instant regrets of OSMP.

Proposition 5.9. *Under Assumption 5.1, we run OSMP (Algorithm 5) with $\lambda \geqslant R^2$ and $\Delta = \mathbb{R}^d$. Consider its instant regret $\widehat{r}_t$ defined in Equation (5.13). The instant regret is upper-bounded, namely*

$$\widehat{r}_t \leqslant e \cdot \sigma'(\langle \theta_t, x_t\rangle) \cdot \|x_t\|^2_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}}, \tag{5.15}$$

where $\nabla^2 L_{\lambda,t}(\theta_t)$ stands for the Hessian of $L_{\lambda,t}$ evaluated on θ_t,

$$\nabla^2 L_{\lambda,t}(\theta_t) = \sum_{s=1}^{t} \sigma'(\langle \theta_t, x_s\rangle) x_s x_s^\top + 2\lambda I_d. \tag{5.16}$$

Proof. Recall that

$$\theta_t = \operatorname*{argmin}_{\theta\in\mathbb{R}^d} L_{\lambda,t}(\theta) = \operatorname*{argmin}_{\theta\in\mathbb{R}^d} \left\{\ell(\theta^\top x_t, y_t) + L_{t-1}(\theta) + \lambda\|\theta\|^2\right\},$$

denote $z_t = -y_t x_t$, and denote

$$\begin{aligned}\theta_t^{-y_t} &\coloneqq \operatorname*{argmin}_{\theta\in\mathbb{R}^d} \left\{\ell(\theta^\top x_t, -y_t) + L_{t-1}(\theta) + \lambda\|\theta\|^2\right\} \\ &= \operatorname*{argmin}_{\theta\in\mathbb{R}^d} \left\{L_t(\theta) - \langle\theta, z_t\rangle + \lambda\|\theta\|^2\right\},\end{aligned}$$

as $\ell(\theta^\top x_t, -y_t) = -\log\sigma(\theta^\top z_t) = -\log\sigma(-\theta^\top z_t) - \langle\theta, z_t\rangle = \ell(\theta^\top x_t, y_t) - \langle\theta, z_t\rangle$. Then from Equation (5.14),

$$\begin{aligned}
\widehat{r}_t &= \log\Big(\exp(-(L_{\lambda,t}^{+1\star} - L_{\lambda,t-1}^\star)) + \exp(-(L_{\lambda,t}^{-1\star} - L_{\lambda,t-1}^\star))\Big) \\
&= \log\Big(\exp(-\ell(\langle\theta_t, x_t\rangle, y_t) - (L_{\lambda,t-1}(\theta_t) - L_{\lambda,t-1}^\star)) \\
&\qquad + \exp(-\ell(\langle\theta_t^{-y_t}, x_t\rangle, -y_t) - (L_{\lambda,t-1}(\theta_t^{-y_t}) - L_{\lambda,t-1}^\star))\Big) \\
&= \log\Big(\sigma(-\langle\theta_t, z_t\rangle)\exp(-(L_{\lambda,t-1}(\theta_t) - L_{\lambda,t-1}^\star)) \\
&\qquad + \sigma(\langle\theta_t^{-y_t}, z_t\rangle)\exp(-(L_{\lambda,t-1}(\theta_t^{-y_t}) - L_{\lambda,t-1}^\star))\Big).
\end{aligned}$$

We use the fact that $L_{\lambda,t-1}(\theta_t) \geqslant L_{\lambda,t-1}^\star$ and $L_{\lambda,t-1}^\star(\theta_t^{-y_t}) \geqslant L_{\lambda,t-1}^\star$, and $\sigma(-u) = 1 - \sigma(u)$, then $\log(1+u) \leqslant u$ for any $u > -1$, to write

$$\begin{aligned}
\widehat{r}_t &\leqslant \log\Big(1 - \sigma(\langle\theta_t, z_t\rangle) + \sigma(\langle\theta_t^{-y_t}, z_t\rangle)\Big) \\
&\leqslant \sigma(\langle\theta_t^{-y_t}, z_t\rangle) - \sigma(\langle\theta_t, z_t\rangle).
\end{aligned} \tag{5.17}$$

Next, to upper-bound the RHS, we roughly follow similar techniques as in the proof of Theorem 5 of Mourtada and Gaïffas [2019]. For the sake of completeness, we present the main steps here.

Since $L_{\lambda,t}(\theta) - \langle\theta, z_t\rangle = L_{\lambda,t}^{-y_t}(\theta)$, and the function $L_{\lambda,t}$ is 2λ-strongly convex (and $\|x_t\| \leqslant R$ from Assumption 5.1), it follows from Lemma 5.10 below that

$$\|\theta_t^{-y_t} - \theta_t\| \leqslant \frac{R}{2\lambda}, \quad 0 \leqslant \langle\theta_t^{-y_t} - \theta_t, z_t\rangle \leqslant \frac{R^2}{2\lambda} \leqslant \frac{1}{2}$$

where we also use the assumption $\lambda \geqslant R^2$ for the last inequality.

Next, since $\log(\sigma')' = \sigma''/\sigma' = 1 - 2\sigma < 1$, we have for every $u \in \mathbb{R}, v \in [0, 1/2]$, $\log\sigma'(u+v) - \log\sigma'(u) \leqslant v$ hence $\sigma'(u+v) \leqslant e^v\sigma'(u) \leqslant e^{1/2}\sigma'(u)$. Hence, $\sigma(u+v) - \sigma(u) \leqslant e^{1/2}\cdot\sigma'(u)\cdot v$ for every $u \in \mathbb{R}, v \in [0, 1/2]$. With $u = \langle\theta_t, z_t\rangle$ and $v = \langle\theta_t^{-y_t} - \theta_t, z_t\rangle$, we have

$$\sigma(\langle\theta_t^{-y_t}, z_t\rangle) - \sigma(\langle\theta_t, z_t\rangle) \leqslant e^{1/2}\cdot\sigma'(\langle\theta_t, z_t\rangle)\cdot\langle\theta_t^{-y_t} - \theta_t, z_t\rangle. \tag{5.18}$$

Next, since the function $L_{\lambda,t}$ is generalized R-self-concordant under Assumption 5.1, from Equation (5.3), we have

$$\nabla^2 L_{\lambda,t}(\theta + \beta) \succcurlyeq e^{-R\|\beta\|}\nabla^2 L_{\lambda,t}(\theta)$$

for any β and θ. By letting $\theta = \theta_t$ and $\beta = \theta' - \theta_t$, we obtain that the function $L_{\lambda,t}$ is $e^{-(1/2+\varepsilon)}\nabla^2 L_{\lambda,t}(\theta_t)$-strongly convex on the open ball $\Omega_\varepsilon = \{\theta' \in \mathbb{R}^d : R\|\theta' - \theta_t\| < 1/2 + \varepsilon\}$, for any $\varepsilon > 0$. In addition, $L_{\lambda,t}^{-y_t}(\theta) = L_{\lambda,t}(\theta) - \langle\theta, z_t\rangle$ reaches its minimum at $\theta_t^{-y_t}$ and $\theta_t^{-y_t} \in \Omega_\varepsilon$. Using Lemma 5.10 again, we have

$$\langle\theta_t^{-y_t} - \theta_t, z_t\rangle \leqslant e^{1/2+\varepsilon}\,\|x_t\|^2_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}}$$

and

$$\|\theta_t^{-y_t} - \theta_t\|_{\nabla^2 L_{\lambda,t}(\theta_t)} \leqslant e^{1/2+\varepsilon}\,\|x_t\|_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}}.$$

Taking $\varepsilon \to 0$, we have

$$\langle\theta_t^{-y_t} - \theta_t, z_t\rangle \leqslant e^{1/2}\,\|x_t\|^2_{\nabla^2 L_{\lambda,t}(\theta_t)^{-1}} \tag{5.19}$$

and

$$\|\theta_t^{-y_t} - \theta_t\|_{\nabla^2 L_{\lambda,t}(\theta_t)} \leqslant e^{1/2} \|x_t\|_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}} .$$

Hence, putting together Equations (5.18) and (5.19), we obtain

$$\begin{aligned}\sigma(\langle \theta_t^{-y_t}, z_t\rangle) - \sigma(\langle \theta_t, z_t\rangle) &\leqslant e \cdot \sigma'(\langle \theta_t, z_t\rangle) \cdot \|x_t\|^2_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}} \\ &= e \cdot \sigma'(\langle \theta_t, z_t\rangle) \cdot \left\langle (\nabla^2 L_{\lambda,t}(\theta_t))^{-1} x_t, x_t \right\rangle \\ &= e \cdot \mathrm{Tr}\left\{ (\nabla^2 L_{\lambda,t}(\theta_t))^{-1} \sigma'(\langle \theta_t, z_t\rangle) x_t x_t^\top \right\}\end{aligned}$$

that we plug back into Equation (5.17), and we remark that σ' is an even function, to conclude on the statement of Proposition. □

The following Lemma is a generalization of Lemma 4 of Mourtada and Gaïffas [2019] by allowing f to be a differentiable function (instead of a linear one); its proof is based the similar arguments.

Lemma 5.10. *(Stability) Let Ω be a nonempty open convex subset of $\mathbb{R}^d$, and $F : \Omega \to \mathbb{R}$ a differentiable function. Assume that F is Σ-strongly convex on Ω, where Σ is a $d \times d$ symmetric positive matrix, in the sense that, for every $x, x' \in \Omega$,*

$$F(x') \geqslant F(x) + \langle \nabla F(x), x' - x\rangle + \frac{1}{2} \|x' - x\|^2_\Sigma . \tag{5.20}$$

Assume $G : x \mapsto F(x) + f(x)$ with f a differentiable function, such that G is strongly convex and G reaches its minimum at some $\tilde{x} \in \Omega$. Denote $x^* \in \Omega$ the minimum of F. Then

$$\|\tilde{x} - x^*\|_\Sigma \leqslant \|\nabla f(\tilde{x})\|_{\Sigma^{-1}}$$

and

$$\langle -\nabla f(\tilde{x}), \tilde{x} - x^*\rangle \leqslant \|\nabla f(\tilde{x})\|^2_{\Sigma^{-1}} .$$

Proof. Since x^* and $\tilde{x}$ are respectively minimizers of F and G, we have $0 = \nabla F(x^*)$ and $0 = \nabla F(\tilde{x}) + \nabla f(\tilde{x})$, the latter implies

$$\langle \nabla F(\tilde{x}), \tilde{x} - x^*\rangle = \langle -\nabla f(\tilde{x}), \tilde{x} - x^*\rangle .$$

Since F is Σ-strongly convex, by substituting x and x', for every x and x',

$$\langle \nabla F(x') - \nabla F(x), x' - x\rangle \geqslant \|x' - x\|^2_\Sigma .$$

Letting $x' = \tilde{x}$ and $x = x^*$, we have

$$\langle \nabla F(\tilde{x}), \tilde{x} - x^*\rangle \geqslant \|\tilde{x} - x^*\|^2_\Sigma .$$

On the other hand, the Cauchy-Schwarz inequality implies that

$$\langle -\nabla f(\tilde{x}), \tilde{x} - x^*\rangle \leqslant \|\nabla f(\tilde{x})\|_{\Sigma^{-1}} \|\tilde{x} - x^*\|_\Sigma ,$$

plugging in, we have

$$\|\tilde{x} - x^*\|^2_\Sigma \leqslant \|\nabla f(\tilde{x})\|_{\Sigma^{-1}} \|\tilde{x} - x^*\|_\Sigma$$

hence

$$\|\tilde{x} - x^*\|_\Sigma \leqslant \|\nabla f(\tilde{x})\|_{\Sigma^{-1}} \quad \text{and} \quad \langle -\nabla f(\tilde{x}), \tilde{x} - x^*\rangle \leqslant \|\nabla f(\tilde{x})\|^2_{\Sigma^{-1}} . \qquad \square$$

Proposition 5.9 is as far as goes our analysis of OSMP, for the reasons described below. The classical approach [Cesa-Bianchi and Lugosi, 2006; Hazan et al., 2007] of online learning literature to upper-bound the instant regret is to use the particular structure of the quadratic form associated to the Hessian matrix of the loss. However, with the upper-bound (5.15), the Hessian $\nabla^2 L_{\lambda,t}(\theta_t)$ depends on θ_t. Let us explain how it works when there is not such a dependence. In that case, we typically bring the upper-bound on the instant regret into an expression of the form

$$\|x_t\|^2_{A_t^{-1}} = \langle x_t, A_t^{-1} x_t \rangle, \quad \text{where} \quad A_t \coloneqq \sum_{s=1}^{t} x_s x_s^\top + 2\lambda I_d. \tag{5.21}$$

Then, we can use Cesa-Bianchi and Lugosi [2006, Lemma 11.11] which leads to

$$\langle x_t, A_t^{-1} x_t \rangle = 1 - \frac{\det A_{t-1}}{\det A_t} \leqslant \log \frac{\det A_t}{\det A_{t-1}} \tag{5.22}$$

and ends up with a telescopic sum on RHS. We recall Lemma 11.11 of Cesa-Bianchi and Lugosi [2006] below.

Lemma 5.11 (Lemma 11.11 of Cesa-Bianchi and Lugosi [2006])**.** *Let B be an arbitrary $d \times d$ full-rank matrix, and $x \in \mathbb{R}^d$ an arbitrary vector. Let $A = B + xx^\top$. Then*

$$x^\top A^{-1} x = 1 - \frac{\det(B)}{\det(A)}.$$

In our case, from Proposition 5.9, we have an upper-bound of the instant regret that writes (up to the constant factor e)

$$\sigma'\left(\langle \theta_t, x_t \rangle\right) \|x_t\|^2_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}}, \quad \text{where} \quad \nabla^2 L_{\lambda,t}(\theta_t) = \sum_{s=1}^{t} \sigma'(\langle \theta_t, x_s \rangle) x_s x_s^\top + 2\lambda I_d,$$

which is indeed very similar to the expression in Equation (5.21). Unfortunately, directly applying Equation (5.22) would not bring us to a telescopic sum, as the Hessian $\nabla^2 L_{\lambda,t}(\theta_t)$ *does* depend on θ_t in each term of the sum *i.e.* each term $\sigma'(\langle \theta_t, x_s \rangle) x_s x_s^\top$ depends on θ_t. We can only write

$$\begin{aligned} \sigma'\left(\langle \theta_t, x_t \rangle\right) \|x_t\|^2_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}} &= 1 - \frac{\det \nabla^2 L_{\lambda,t-1}(\theta_t)}{\det \nabla^2 L_{\lambda,t}(\theta_t)} \\ &\leqslant \log \frac{\det \nabla^2 L_{\lambda,t}(\theta_t)}{\det \nabla^2 L_{\lambda,t-1}(\theta_t)}, \end{aligned}$$

but the sum of RHS terms is not telescopable.

In the following, we might want to control each term $\sigma'(\langle \theta_t, x_s \rangle)$ for indices $s < t$, and in particular we might want to find a lower-bound of $\sigma'(\langle \theta_t, x_s \rangle)$ in function of $\sigma'(\langle \theta_t, x_t \rangle)$ that eventually leads a term in a telescopic sum. We give below a list of directions that we tried together with the specific difficulties in each.

- A first idea is to use the uniform lower-bound, *i.e.* for $\|\theta_t\| \leqslant B$ and $\|x_s\| \leqslant R$,

$$\sigma'(\langle \theta_t, x_s \rangle) \geqslant \exp(-BR), \tag{5.23}$$

  then

$$\nabla^2 L_{\lambda,t}(\theta_t) \succcurlyeq \exp(-BR) \sum_{s=1}^{t} x_s x_s^\top + 2\lambda I_d.$$

From this, we simply use $\sigma'(\langle\theta_t, x_t\rangle) \leqslant 1/4$ and apply Lemma 11.11 of Cesa-Bianchi and Lugosi [2006], with $A_t \coloneqq \exp(-BR)\sum_{s=1}^{t} x_s x_s^\top + 2\lambda I_d$, we obtain

$$\begin{aligned}\sigma'(\langle\theta_t, x_t\rangle)\,\|x_t\|^2_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}} &\leqslant \frac{1}{4}\|x_t\|^2_{A_t^{-1}} \\ &= \frac{\exp(BR)}{4}\left(1 - \frac{\det A_{t-1}}{\det A_t}\right) \leqslant \frac{\exp(BR)}{4}\log\frac{A_t}{A_{t-1}}.\end{aligned}$$

Putting this into Equation (5.15), we can upper-bound the sum of instant regrets by a telescopic sum which results

$$\sum_{t=1}^{n} \widehat{r}_t \leqslant \frac{\exp(BR+1)}{4}\log\frac{\det A_n}{\det A_0}$$

where we can use

$$\log\det(A_n/2\lambda) \leqslant d\log\left(1 + \frac{e^{-BR}nR^2}{2d\lambda}\right).$$

Finally, with the choice $\lambda = R^2$, we have

$$\mathrm{Regret}_n \leqslant \frac{e^{BR+1}}{4} d\log\left(1 + \frac{e^{-BR}}{2d}n\right) + B^2R^2.$$

In this way, we obtain for OSMP a regret upper-bound that is logarithmic in the number of total rounds n; but exactly because of the use of the uniform lower-bound (5.23), the factor $\exp(BR)$ in the regret is unavoidable. That is the reason why we might need to use some bounds that are more precise than the uniform one (5.23).

- From the stability lemma (Lemma 5.10), we know that θ_t is "not very far" from $\theta_t^{-y_t}$, and that θ_t is "not very far" from θ_{t-1}. For example, using the fact that the function $L_{\lambda,t}$ is 2λ-strongly convex together with the stability lemma, we know that the distance between θ_t and θ_{t-1} is upper-bounded by a constant, *i.e.* $\|\theta_t - \theta_{t-1}\| \leqslant \frac{R}{2\lambda}$. On the other hand, since $\log(\sigma')' = \sigma''/\sigma' = 1 - 2\sigma < 1$, we have for every $u \in \mathbb{R}, v \in [0, 1/2]$, $\log\sigma'(u+v) - \log\sigma'(u) \leqslant v$ hence $\sigma'(u+v) \leqslant e^v\sigma'(u) \leqslant e^{1/2}\sigma'(u)$. Applying this with $u = \langle\theta_t, x_s\rangle$ and $v = \langle\theta_{t-1} - \theta_t, x_s\rangle$ yields (up to change (u, v) to $(-u, -v)$ to guarantee $v \geqslant 0$, as σ' is an even function)

$$\sigma'(\langle\theta_t, x_s\rangle) \geqslant e^{-1/2}\sigma'(\langle\theta_{t-1}, x_s\rangle) \tag{5.24}$$

for any indices $t > 1$ and s. We plug this into the Hessian $\nabla^2 L_{\lambda,t}(\theta_t)$ (Equation (5.16)), then

$$\begin{aligned}\nabla^2 L_{\lambda,t}(\theta_t) &\succcurlyeq \sigma'(\langle\theta_t, x_t\rangle)x_t x_t^\top + e^{-1/2}\sum_{s=1}^{t-1}\sigma'(\langle\theta_{t-1}, x_s\rangle)x_s x_s^\top + 2\lambda I_d \\ &\succcurlyeq \sigma'(\langle\theta_t, x_t\rangle)x_t x_t^\top + e^{-1/2}\nabla^2 L_{\lambda,t-1}(\theta_{t-1}).\end{aligned}$$

Hence, applying Lemma 11.11 of Cesa-Bianchi and Lugosi [2006], we obtain

$$\begin{aligned}\sigma'(\langle\theta_t, x_t\rangle)\,\|x_t\|^2_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}} &\leqslant 1 - \frac{e^{-d/2}\det\nabla^2 L_{\lambda,t-1}(\theta_{t-1})}{\det\nabla^2 L_{\lambda,t}(\theta_t)} \\ &\leqslant \log\frac{\det\nabla^2 L_{\lambda,t}(\theta_t)}{\det\nabla^2 L_{\lambda,t-1}(\theta_{t-1})} + \frac{d}{2}.\end{aligned}$$

Summing over the first term is telescopic, but summing over the second term would result in a linear term. Indeed, with

$$\log\det\left(\nabla^2 L_{\lambda,t}(\theta_t)/2\lambda\right) \leqslant d\log\left(1+\frac{nR^2}{8d\lambda}\right),$$

and the choice $\lambda = R^2$, we would finally obtain for OSMP

$$\mathrm{Regret}_n \leqslant ed\log\left(1+\frac{n}{8d}\right)+\frac{dn}{2}+B^2R^2.$$

In this way, we get rid of the exponential factor in the regret upper-bound, but we have a term that is linear in the number of total rounds n: the approach with (5.24) fails to show a logarithmic regret. We might notice that the $d/2$ additional term comes exactly from the factor $e^{-1/2}$ in Equation (5.24); to avoid the linear term, a lower-bound with a factor more precise than $e^{-1/2}$ might be useful instead of Equation (5.24).

- Indeed, we may more heavily exploit the generalized R-self-concordance of the function $L_{\lambda,t}$, hence more precisely characterize the distance between θ_t and θ_{t-1}. As consequence, $L_{\lambda,t}$ is not only 2λ-strongly convex but also "locally" $e^{-1/2}\nabla^2 L_{\lambda,t}(\theta_t)$-strongly convex. Using this together with Lemma 5.10, we have

$$\|\theta_t-\theta_{t-1}\|_{\nabla^2 L_{\lambda,t}(\theta_t)} \leqslant e^{1/2}\cdot\|\sigma(\langle\theta_t,z_t\rangle)x_t\|_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}}.$$

On the other hand, with similar arguments as before, namely from $\log(\sigma')' = \sigma''/\sigma' = 1-2\sigma < 1$, we have for every $u\in\mathbb{R}, v>0$, $\log\sigma'(u+v)-\log\sigma'(u)\leqslant v$ hence $\sigma'(u+v)\leqslant e^v\sigma'(u)$; applying this for $u=\langle\theta_t,x_s\rangle$ and $v=\langle\theta_{t-1}-\theta_t,x_s\rangle$ yields (up to change (u,v) to $(-u,-v)$ to guarantee $v\geqslant 0$, as σ' is an even function)

$$\sigma'(\langle\theta_t,x_s\rangle)\geqslant\exp(-|\langle\theta_t-\theta_{t-1},x_s\rangle|)\sigma'(\langle\theta_{\lambda,t-1},z_s\rangle),$$

where we can use the Cauchy-Schwarz inequality

$$|\langle\theta_t-\theta_{t-1},x_s\rangle| \leqslant \|\theta_t-\theta_{t-1}\|_{\nabla^2 L_{\lambda,t}(\theta_t)}\|x_s\|_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}}.$$

We plug this into the Hessian $\nabla^2 L_{\lambda,t}(\theta_t)$ (Equation (5.16)), then

$$\begin{aligned}
&\nabla^2 L_{\lambda,t}(\theta_t)\\
&\succcurlyeq\sigma'(\langle\theta_t,x_t\rangle)x_tx_t^\top+\sum_{s=1}^{t-1}\exp(-|\langle\theta_t-\theta_{t-1},x_s\rangle|)\sigma'(\langle\theta_{t-1},x_s\rangle)x_sx_s^\top+2\lambda I_d\\
&\succcurlyeq\sigma'(\langle\theta_t,x_t\rangle)x_tx_t^\top+2\lambda I_d\\
&\quad+\sum_{s=1}^{t-1}\exp(-e^{1/2}\|x_t\|_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}}\|x_s\|_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}})\sigma'(\langle\theta_{t-1},x_s\rangle)x_sx_s^\top.
\end{aligned}$$

But it is not clear what can be done with this.

Finally, we also remark that there is a similar situation in the proof of Theorem 6 of Mourtada and Gaïffas [2019], Equations (92)-(94). Under the setting of i.i.d. data and while looking for upper-bounds in expectation, they used an argument of exchangeability ($Z_1,\dots,Z_{t-1},Z_t$ follow the same distribution). Unfortunately, this argument is not possible in the online setting, as we do not make any assumption on the distribution of data $(x_t,y_t)_t$, and we look for a regret upper-bound that is valid for any individual sequence.

5.5.4 Some special cases in dimension 1

In dimension $d = 1$, we rewrite Equation (5.15) of Proposition 5.9 as

$$\widehat{r}_t \leqslant e \cdot \frac{x_t^2 \sigma'(\theta_t x_t)}{\sum_{s=1}^t x_s^2 \sigma'(\theta_t x_s) + 2\lambda}. \tag{5.25}$$

For x_t taking values in $\{-R, 0, R\}$. Let us start with a simplified situation where $x_t \in \{-1, 1\}$. Denote $z_t = -y_t x_t$, then $z_t \in \{-1, +1\}$. Denote $P_t \coloneqq \mathrm{card}\{s \leqslant t : z_s = 1\}$ the number of $z_s = 1$, and $N_t \coloneqq \mathrm{card}\{s \leqslant t : z_s = -1\}$ the number of $z_s = -1$. Then,

$$L_{\lambda,t}(\theta) = -P_t \log \sigma(-\theta) - N_t \log \sigma(\theta) + \lambda\theta^2. \tag{5.26}$$

- For $\lambda > 0$ (and we are in particular interested in the case $\lambda \geqslant 1$), although we do not have a closed form for θ_t or $L_t^\star$, we can make the following observation: since σ' is an even function, $z_s \in \{-1, +1\}$, $x_s^2 \sigma'(\theta_t z_s)$ takes the same value for all $s \leqslant t$. Therefore, it can be deduced from Equation (5.25) that

$$\widehat{r}_t \leqslant e \cdot \frac{1}{t}, \tag{5.27}$$

  hence $\mathrm{Regret}_n \leqslant e \cdot \log n + B^2 + O(1)$ with $\lambda = 1$.

- This upper-bound in form of the one of (5.27) remains true if we allow x_t to take value 0: in Equation (5.25), we can simply ignore all terms in $s < t$ with $x_s = 0$ as they don't affect the value of the RHS of Equation (5.25).

Now consider the case where x_t only takes values in $\{-R, 0, R\}$ for some $R > 0$. We remark that for all $s \leqslant t$,

$$x_s^2 \sigma'(\theta_t z_s) = \begin{cases} 0, & x_s = 0, \\ \xi_t, & \text{otherwise}, \end{cases}$$

with $\xi_t = R^2 \sigma'(R\theta_t)$. We can still deduce an upper-bound in form of (5.27) by ignoring terms with $x_s = 0$, hence $\mathrm{Regret}_n \leqslant e \cdot \log n + B^2 R^2 + O(1)$ with $\lambda = R^2$.

5.6 AOSMP: Approximated One-Step Minmax Predictor

In this section, we study a variant of the minmax approach presented in the previous Section: instead of doing minmax on the exact cumulative loss function, we propose to do minmax on an approximation of the cumulative loss function. We rely on the following quadratic approximation of the logistic function as in Jézéquel et al. [2020].

Lemma 5.12 (Lemma 5 of Jézéquel et al. [2020] restated)**.** *Given $\|x_t\| \leqslant R$ for all time t. For the logistic loss function induced by (x_t, y_t) the features and the label of time t, $\ell_t(\theta) = -\log \sigma(y_t \theta^\top x_t)$, consider the quadratic surrogate of $\ell_t(\theta)$, developed around any $\tilde{\theta}_t \in \mathbb{R}^d$, defined as*

$$\tilde{\ell}_t(\theta) \coloneqq \ell_t(\tilde{\theta}_t) + g_t^\top (\theta - \tilde{\theta}_t) + \frac{1}{2}\eta_t \left(x_t^\top (\theta - \tilde{\theta}_t)\right)^2 \tag{5.28}$$

with

$$g_t \coloneqq \nabla \ell_t(\tilde{\theta}_t) = -y_t \sigma(\langle \tilde{\theta}_t, x_t \rangle) x_t, \quad \eta_t \coloneqq \frac{\sigma'_t}{1 + BR} \coloneqq \frac{\sigma'(\langle \tilde{\theta}_t, x_t \rangle)}{1 + BR}$$

Then, for any $\theta \in \mathscr{B}(\mathbb{R}^d, B)$,

$$\tilde{\ell}_t(\theta) \leqslant \ell_t(\theta).$$

Interested readers can refer to [Jézéquel et al., 2020] for the proof of this Lemma, which relies on an exact study of the logistic function. The quadratic lower-bound (5.28) is better than the one from exp-concavity, as the curvature parameter η_t depends adaptively on the scalar product between the features x_t and the point $\tilde{\theta}_t$ from which we do the development.

5.6.1 Algorithm

Now we present the approximated version of OSMP. We directly write the *Approximated One-Step Minmax Predictor* (AOSMP) with the search space $\Delta = \mathbb{R}^d$.

Definition 5.13 (AOSMP)**.** Given $\lambda > 0$ the ridge penalization parameter, the AOSMP computes the following estimator

$$\widehat{y}_t = \operatorname*{argmin}_{\hat{y} \in \mathbb{R}} \sup_{y_t \in \{-1,+1\}} \sup_{\theta \in \mathbb{R}^d} \left\{ \widehat{L}_{t-1} + \ell(\hat{y}, y_t) - \left(\tilde{L}_{t-1}(\theta) + \ell(\theta^\top x_t, y_t) + \lambda \left\| \theta \right\|^2 \right) \right\} \tag{5.29}$$

with $\widehat{L}_{t-1} \coloneqq \sum_{s=1}^{t-1} \ell(\widehat{y}_s, y_s)$ the actual sum of losses suffered by successively proposed predictions $(\widehat{y}_s)$ until time $t-1$, and $\tilde{L}_{t-1}$ the function taking parameter θ, corresponding to the sum of the approximated losses until time $t-1$ suffered by a fixed parameter θ, *i.e.* $\tilde{L}_{t-1}(\theta) \coloneqq \sum_{s=1}^{t-1} \tilde{\ell}_s(\theta)$ where $\tilde{\ell}_s$ are successive quadratic approximation functions defined in Equation (5.28) with the choice of $\tilde{\theta}_s$ specified below (Equation (5.30)).

AOSMP (5.29) is very much similar to the "exact" one defined in (5.8): both are minmax, taking the best against the worst possible scenario in terms of real label y_t and the adversary parameter θ. The difference is that in (5.29), in the evaluation of the adversary parameter θ, we replace the actual cumulative loss function until the previous time L_{t-1} by the cumulative surrogate quadratic form $\tilde{L}_{t-1}$. Replacing the true loss function by its quadratic surrogate is a common practice in the literature of online logistic regression [Hazan et al., 2007; Jézéquel et al., 2020], as the quadratic functions are easier to compute, and at the same time the quadratic forms usually allow telescopic sum in the computations in the regret analysis, as we will see later in this section.

Notations and the choice of $\tilde{\theta}_t$

We introduce the notations

$$\tilde{L}_{\lambda,t}(\theta) \coloneqq \tilde{L}_t(\theta) + \lambda \left\| \theta \right\|^2, \quad \tilde{L}^\star_{\lambda,t} \coloneqq \inf_{\theta \in \mathbb{R}^d} \left\{ \tilde{L}_t(\theta) + \lambda \left\| \theta \right\|^2 \right\}$$

and for $y \in \{-1, +1\}$, the functions

$$\tilde{L}^y_{\lambda,t}(\theta) \coloneqq \ell(\theta^\top x_t, y) + \tilde{L}_{\lambda,t-1}(\theta)$$

and their infimum

$$\tilde{L}^{y\star}_{\lambda,t} \coloneqq \inf_{\theta \in \mathbb{R}^d} \left\{ \ell(\theta^\top x_t, y) + \tilde{L}_{\lambda,t-1}(\theta) \right\} \quad \text{for} \quad y \in \{-1, +1\}.$$

Denote for $y \in \{-1, +1\}$,

$$\tilde{\theta}_t^{(x_t,y)} \coloneqq \operatorname*{argmin}_{\theta \in \mathbb{R}^d} \tilde{L}_{\lambda,t}^{y}(\theta) \quad \text{with} \quad \tilde{L}_{\lambda,t}^{y}(\theta) \coloneqq \ell(\theta^\top x_t, y) + \tilde{L}_{\lambda,t-1}(\theta).$$

We remark that $\tilde{\theta}_t^{(x_t,y)}$ exists and is unique as the minimizer of a strongly convex function $\tilde{L}_t^y$. In particular, we choose

$$\tilde{\theta}_t \coloneqq \tilde{\theta}_t^{(x_t,y_t)} \tag{5.30}$$

such that $\tilde{\theta}_t$ is the point around which we do the quadratic development for $\tilde{\ell}_t$ in Equation (5.28). Making such choice is possible after the true label y_t is revealed.

Alternative expression and explicit algorithm

Similarly to OSMP, we have an explicit expression for AOSMP. We skip the proof of this Lemma as it is very much similar to the one of Lemma 5.4, relying on the properties of the log-loss.

Lemma 5.14. *AOSMP defined in Equation (5.29) is equivalent to*

$$\widehat{y}_t = -\tilde{L}_{\lambda,t}^{+1\star} + \tilde{L}_{\lambda,t}^{-1\star} \tag{5.31}$$

Remark 5.15. AOSMP predicts the probabilities $p(+1 \mid x_t) = \widehat{p}_t$ and $p(-1 \mid x_t) = 1 - \widehat{p}_t$ with

$$\widehat{p}_t = \frac{\exp(-\tilde{L}_{\lambda,t}^{+1\star})}{\exp(-\tilde{L}_{\lambda,t}^{+1\star}) + \exp(-\tilde{L}_{\lambda,t}^{-1\star})} \tag{5.32}$$

Let us summarize AOSMP in Algorithm 6.

Algorithm 6 AOSMP description: computation of $\widehat{y}_t$.

Inputs: Parameters $\lambda > 0$, $n, d \geqslant 1$, constants $B, R > 0$
Initialize: Function $\tilde{L}_{\lambda,0}(\theta) = \lambda \|\theta\|^2$
for $t = 1, \ldots, n$ **do**
  Receive x_t
  Compute $\tilde{L}_{\lambda,t}^{y\star} \leftarrow \inf_{\theta \in \mathbb{R}^d} \left\{ \ell(\theta^\top x_t, y) + \tilde{L}_{\lambda,t-1}(\theta) \right\}$ for $y = -1$ and $y = +1$
  Predict $\widehat{y}_t \leftarrow -\tilde{L}_{\lambda,t}^{+1\star} + \tilde{L}_{\lambda,t}^{-1\star}$
  Receive y_t
  Compute $\tilde{\theta}_t \leftarrow \operatorname{argmin}_{\theta \in \mathbb{R}^d} \left\{ \ell(\theta^\top x_t, y_t) + \tilde{L}_{\lambda,t-1}(\theta) \right\}$, and function $\tilde{\ell}_t(\theta)$ as specified in Equation (5.28)
  Update function $\tilde{L}_\lambda(\theta) = \tilde{L}_{\lambda,t-1}(\theta) + \tilde{\ell}_t(\theta)$
end for

5.6.2 Pseudo instant regret and regret upper-bound

Now we move to the regret analysis of the AOSMP. As before, we introduce *pseudo instant regret* as the component term in the sum over time steps.

Definition 5.16 (Pseudo instant regret)**.** We call "pseudo instant regret" at time t the quantity defined by

$$\widehat{r}_t \coloneqq \ell(\widehat{y}_t, y_t) - \tilde{L}_{\lambda,t}^{\star} + \tilde{L}_{\lambda,t-1}^{\star}. \tag{5.33}$$

Lemma 5.17. *An alternative expression for the pseudo instant regret writes*

$$\widehat{r}_t = \log\left(\exp(-\tilde{L}_{\lambda,t}^{+1\star} + \tilde{L}_{\lambda,t-1}^{\star}) + \exp(-\tilde{L}_{\lambda,t}^{-1\star} + \tilde{L}_{\lambda,t-1}^{\star})\right). \tag{5.34}$$

Proof. We plug the expression of $\widehat{y}_t$ (Equation (5.31)) into the definition of $\widehat{r}_t$ (Equation (5.33)) to obtain

$$\begin{aligned}\widehat{r}_t &= \log\left(1 + \exp\left(-y_t(-\tilde{L}_{\lambda,t}^{+1\star} + \tilde{L}_{\lambda,t}^{-1})\right)\right) - \tilde{L}_{\lambda,t} + \tilde{L}_{\lambda,t-1} \\ &= \log\left(1 + \exp\left(\tilde{L}_{\lambda,t}^{y_t\star} - \tilde{L}_{\lambda,t}^{-y_t\star}\right)\right) - \tilde{L}_{\lambda,t}^{\star} + \tilde{L}_{\lambda,t-1}^{\star}.\end{aligned}$$

Furthermore, we notice that $\ell_t(\tilde{\theta}_t) = \tilde{\ell}_t(\tilde{\theta}_t)$ by the definition of $\tilde{\ell}_t$, then

$$\tilde{L}_{\lambda,t}^{y_t\star} = \ell(\tilde{\theta}_t^\top x_t, y_t) + \tilde{L}_{\lambda,t-1}(\tilde{\theta}_t) = \tilde{L}_{\lambda,t}(\tilde{\theta}_t),$$

and since $0 = \nabla\ell_t(\tilde{\theta}_t) + \nabla\tilde{L}_{\lambda,t-1}(\tilde{\theta}_t) = \nabla\tilde{L}_{\lambda,t}(\tilde{\theta}_t)$, we therefore have

$$\tilde{\theta}_t = \operatorname*{argmin}_{\theta\in\mathbb{R}^d} \tilde{L}_{\lambda,t}(\theta) \quad \text{and} \quad \tilde{L}_{\lambda,t}^{y_t\star} = \ell_t(\tilde{\theta}_t) + \tilde{L}_{\lambda,t-1}(\tilde{\theta}_t) = \tilde{L}_{\lambda,t}^{\star}.$$

Thus, we obtain the statement in Lemma. □

The following Lemma provides an upper-bound for the sum of the pseudo instant regrets $\widehat{r}_t$.

Lemma 5.18. *Assume $\lambda \geqslant R^2$, then the sum of pseudo instant regrets is upper-bounded, namely*

$$\sum_{t=1}^{n} \widehat{r}_t \leqslant e\cdot(1+BR)\log\det(A_n/2\lambda)$$

with $A_t \in \mathbb{R}^{d\times d}$ defined by

$$A_t := \nabla^2\tilde{L}_{\lambda,t}(\tilde{\theta}_t) = \frac{1}{1+BR}\sum_{s=1}^{t}\sigma_s' x_s x_s^\top + 2\lambda I_d.$$

Proof. We start from rewriting Equation (5.34), while denoting $z_t = -y_t x_t$ and $\tilde{\theta}_t^{-y_t} = \tilde{\theta}_t^{(x_t,-y_t)}$,

$$\begin{aligned}\widehat{r}_t =& \log\left(\sigma(y_t\langle\tilde{\theta}_t^{y_t}, x_t\rangle)\exp(-\tilde{L}_{\lambda,t-1}(\tilde{\theta}_t^{y_t}) + \tilde{L}_{\lambda,t-1}^{\star})\right. \\ &\left. + \sigma(-y_t\langle\tilde{\theta}_t^{-y_t}, x_t\rangle)\exp(-\tilde{L}_{\lambda,t-1}(\tilde{\theta}_t^{-y_t}) + \tilde{L}_{\lambda,t-1}^{\star})\right) \\ \leqslant& \log\left(\sigma(-\langle\tilde{\theta}_t, z_t\rangle) + \sigma(\langle\tilde{\theta}_t^{-y_t}, z_t\rangle)\right) \\ \leqslant& \sigma(\langle\tilde{\theta}_t^{-y_t}, z_t\rangle) - \sigma(\langle\tilde{\theta}_t, z_t\rangle)\end{aligned} \tag{5.35}$$

where we use $\tilde{L}_{\lambda,t-1}(\tilde{\theta}_t^{y_t}) \geqslant \tilde{L}_{t-1}^{\star}$ and $\tilde{L}_{\lambda,t-1}(\tilde{\theta}_t^{-y_t}) \geqslant \tilde{L}_{t-1}^{\star}$ on the second line, $\sigma(-u) = 1-\sigma(u)$ and $\log(1+u) \leqslant u$ on the third line. Then, using similar arguments (namely the generalized R-self-concordance of the function $\tilde{L}_{\lambda,t}^{y_t}$ under Assumption 5.1, then $\lambda \geqslant R^2$, followed by a study of the function σ) as in the proof of Proposition 5.9, we obtain

$$\sigma(\langle\tilde{\theta}_t^{-y_t}, z_t\rangle) - \sigma(\langle\tilde{\theta}_t, z_t\rangle) \leqslant e\cdot\sigma'(\langle\tilde{\theta}_t, x_t\rangle)\cdot\left\langle(\nabla^2\tilde{L}_{\lambda,t}^{y_t}(\tilde{\theta}_t))^{-1}x_t, x_t\right\rangle. \tag{5.36}$$

The Hessian $\nabla^2\tilde{L}_{\lambda,t}^{y_t}$ evaluated at $\tilde{\theta}_t$ writes

$$\nabla^2\tilde{L}_{\lambda,t}^{y_t}(\tilde{\theta}_t) = \sigma'(\langle\tilde{\theta}_t, x_t\rangle)x_t x_t^\top + \sum_{s=1}^{t-1}\frac{1}{1+BR}\sigma'(\langle\tilde{\theta}_s, x_s\rangle)x_s x_s^\top + 2\lambda I_d,$$

hence, with the definition of A_t in the statement of the Lemma,

$$\nabla^2 \tilde{L}_{\lambda,t}^{y_t}(\tilde{\theta}_t) = (1+BR)(A_t - A_{t-1}) + A_{t-1} \succcurlyeq A_t. \tag{5.37}$$

Recall that $A_{t-1} = A_t - \frac{\sigma_t'}{1+BR} x_t x_t^\top$, then from Cesa-Bianchi and Lugosi [2006, Lemma 11.11],

$$\sigma_t' \left\langle A_t^{-1} x_t, x_t \right\rangle = (1+BR)\left(1 - \frac{\det A_{t-1}}{\det A_t}\right). \tag{5.38}$$

Putting together Equations (5.35), (5.36), (5.37) and (5.38) implies

$$\widehat{r}_t \leqslant e \cdot (1+BR)\left(1 - \frac{\det A_{t-1}}{\det A_t}\right) \leqslant e \cdot (1+BR) \log \frac{\det A_t}{\det A_{t-1}}.$$

By summing for t from 1 to n, using $\tilde{L}_0(\theta) = 0$, $A_0 = 2\lambda I_d$, and the telescopic sum, we have

$$\sum_{t=1}^{n} \widehat{r}_t \leqslant e \cdot (1+BR) \log\det(A_n/2\lambda). \qquad \square$$

Now we are ready to state the Theorem on the regret upper-bound of AOSMP.

Theorem 5.19. *Let $\lambda, R, B > 0$ and $d, n \geqslant 1$. Let $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$ be an arbitrary sequence of observations. Under Assumptions 5.1 and 5.2, we run AOSMP (Algorithm 6) with $\lambda \geqslant R^2$. Then, its regret satisfies the following upper-bound, against any $\theta \in \mathscr{B}(\mathbb{R}^d, B)$,*

$$\mathrm{Regret}_n(\theta) \leqslant e \cdot (1+BR) d \log\left(1 + \frac{nR^2}{8d(1+BR)\lambda}\right) + \lambda \|\theta\|^2.$$

In particular, by choosing $\lambda = R^2$, it yields

$$\mathrm{Regret}_n \leqslant e \cdot (1+BR) d \log\left(1 + \frac{n}{8d(1+BR)}\right) + B^2 R^2.$$

Proof. Recall that for the sequence $(x_t, y_t)_t$ and the predictions $(\widehat{y}_t)_t$, the regret against $\theta \in \mathscr{B}(\mathbb{R}^d, B)$ until the round n writes

$$\mathrm{Regret}_n(\theta) = \sum_{t=1}^{n} \ell(\widehat{y}_t, y_t) - \sum_{t=1}^{n} \ell(\theta^\top x_t, y_t).$$

We upper-bound each $\ell_t(\theta)$ by $\tilde{\ell}(\theta)$, using Lemma 5.12, to obtain

$$\mathrm{Regret}_n(\theta) \leqslant \sum_{t=1}^{n} \ell(\widehat{y}_t, y_t) - \sum_{t=1}^{n} \tilde{\ell}_t(\theta) = \sum_{t=1}^{n} \ell(\widehat{y}_t, y_t) - \tilde{L}_n(\theta),$$

that we bring into a sum of the pseudo instant regrets

$$\begin{aligned}\mathrm{Regret}_n(\theta) &\leqslant \sum_{t=1}^{n} \ell(\widehat{y}_t, y_t) - \inf_{\theta' \in \mathbb{R}^d} \left\{ \tilde{L}_n(\theta') + \lambda \|\theta'\|^2 \right\} + \lambda \|\theta\|^2 \\ &= \sum_{t=1}^{n} \left(\ell(\widehat{y}_t, y_t) - \tilde{L}_{\lambda,t}^\star + \tilde{L}_{\lambda,t-1}^\star \right) + \lambda \|\theta\|^2.\end{aligned}$$

Then, applying Lemma 5.18 as we assume $\lambda \geqslant R^2$ to upper-bound the sum of pseudo instant regrets term, and using the notations of the same Lemma, we obtain

$$\mathrm{Regret}_n(\theta) \leqslant e \cdot (1+BR) \log\det(A_n/2\lambda) + \lambda \|\theta\|^2.$$

To upper-bound the RHS, we recall that $A_n = \frac{1}{1+BR}\sum_{t=1}^n \sigma'_t x_t x_t^\top + 2\lambda I_d$ with $0 < \sigma'_t \leqslant 1/4$. Denote $C_n = \sum_{t=1}^n \sigma'_t x_t x_t^\top$, then

$$\log\det(A_n/2\lambda) = \sum_{k=1}^d \log\left(1 + \frac{\lambda_k(C_n)}{2(1+BR)\lambda}\right)$$

where $\lambda_k(C_n)$ is the k-th largest eigenvalue of C_n. We remark that $\lambda_k(C_n) \geqslant 0$ for all k, and $\sum_{k=1}^d \lambda_k(C_n) = \mathrm{Tr}(C_n) \leqslant nR^2/4$. Then the RHS is maximized under the constraint $\sum \lambda_k(C_n) \leqslant nR^2/4$ when all the eigenvalues are equal, *i.e.* $\lambda_k(C_n) = nR^2/(4d)$ for all k. That leads to

$$\log\det\left(A_n/2\lambda\right) \leqslant d\log\left(1 + \frac{nR^2}{8d(1+BR)\lambda}\right).$$

To wrap up, for any $\theta \in \mathscr{B}(\mathbb{R}^d, B)$, under the condition $\lambda \geqslant R^2$, we have

$$\mathrm{Regret}_n(\theta) \leqslant e\cdot(1+BR)d\log\left(1 + \frac{nR^2}{8d(1+BR)\lambda}\right) + \lambda\left\|\theta\right\|^2,$$

which means that with the choice $\lambda = R^2$, we have

$$\mathrm{Regret}_n \leqslant e\cdot(1+BR)d\log\left(1 + \frac{n}{8d(1+BR)}\right) + B^2R^2. \qquad \square$$

We obtain for AOSMP, a regret upper-bound in $O(dBR\log(n) + B^2R^2)$, which is logarithmic in the number of total rounds n, with a multiplicative constant of order dBR, and an additional term B^2R^2. Considering Remark 5.1, $B^2R^2 \asymp d$, the regret upper-bound is in $O(d\sqrt{d}\log(n))$. Finally, we remark that the upper-bound of AOSMP is of the same order as the one of AIOLI [Jézéquel et al., 2020], since we use the same quadratic approximation (Lemma 5.12).

5.7 Discussion and perspective

In this work, for online logistic regression, we investigated two new candidate algorithms inspired by SMP [Mourtada and Gaïffas, 2019].

- For OSMP, we found an upper-bound for instant regret $\widehat{r}_t \leqslant e\cdot\sigma'(\langle\theta_t, x_t\rangle)\cdot\left\|x_t\right\|^2_{(\nabla^2 L_{\lambda,t}(\theta_t))^{-1}}$. Despite our initial thoughts, we were facing some technical difficulties and were not able to obtain a regret upper-bound at the same level with best known algorithm.

- For AOSMP which uses a quadratic surrogate of the logistic function, we proved a regret upper-bound which is similar to the one of Jézéquel et al. [2020], state-of-the-art for binary online logistic regression to the best of our knowledge.

The question for better regret with an efficient algorithm for online logistic regression remains open. Also, we believe that the generalization of OSMP and AOSMP in the case of multi-class logistic regression is certainly possible.

Bibliography

N. Agarwal, S. Kale, and J. Zimmert. Efficient methods for online multiclass logistic regression. *arXiv preprint arXiv:2110.03020*, 2021. 127, 131, 132

K. S. Azoury and M. K. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43(3): 211–246, 2001. 131

F. Bach. Self-concordant analysis for logistic regression. *Electronic Journal of Statistics*, 4(none):384 – 414, 2010. doi: 10.1214/09-EJS521. URL https://doi.org/10.1214/09-EJS521. 126, 127

N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006. 124, 139, 140, 146

D. J. Foster, S. Kale, H. Luo, M. Mohri, and K. Sridharan. Logistic regression: The importance of being improper. In S. Bubeck, V. Perchet, and P. Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 167–208. PMLR, 06–09 Jul 2018. URL http://proceedings.mlr.press/v75/foster18a.html. 127, 130, 132

E. Hazan. Introduction to online convex optimization. *arXiv preprint arXiv:1909.05207*, 2019. 124, 129

E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007. 126, 128, 129, 132, 139, 143

E. Hazan, T. Koren, and K. Y. Levy. Logistic regression: Tight bounds for stochastic and online optimization. In *Conference on Learning Theory*, pages 197–209. PMLR, 2014. 129

P. Jacquet, G. Shamir, and W. Szpankowski. Precise minimax regret for logistic regression with categorical feature values. In V. Feldman, K. Ligett, and S. Sabato, editors, *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*, volume 132 of *Proceedings of Machine Learning Research*, pages 755–771. PMLR, 16–19 Mar 2021. URL http://proceedings.mlr.press/v132/jacquet21a.html. 130

R. Jézéquel, P. Gaillard, and A. Rudi. Efficient improper learning for online logistic regression. In *Conference on Learning Theory*, pages 2085–2108. PMLR, 2020. 127, 131, 132, 142, 143, 147

R. Jézéquel, P. Gaillard, and A. Rudi. Mixability made efficient: Fast online multiclass logistic regression. working paper or preprint, Oct. 2021. URL https://hal.archives-ouvertes.fr/hal-03370530. 127, 131, 132

S. M. Kakade and A. Ng. Online bounds for bayesian algorithms. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005. URL https://proceedings.neurips.cc/paper/2004/file/c60d870eaad6a3946ab3e8734466e532-Paper.pdf. 126, 129, 130

H. B. McMahan and M. Streeter. Open problem: Better bounds for online logistic regression. In *COLT/ICML Joint Open Problem Session, JMLR: Workshop and Conference Proceedings*, 2012. 128

J. Mourtada and S. Gaïffas. An improper estimator with optimal excess risk in misspecified density estimation and logistic regression. *arXiv preprint arXiv:1912.10784*, 2019. 125, 127, 131, 132, 135, 136, 137, 138, 141, 147

Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003. 126

F. Orabona. A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*, 2019. 124

G. I. Shamir. Logistic regression regret: What's the catch? In J. D. Abernethy and S. Agarwal, editors, *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pages 3296–3319. PMLR, 2020. URL http://proceedings.mlr.press/v125/shamir20a.html. 130

V. Vovk. Competitive on-line statistics. *International Statistical Review / Revue Internationale de Statistique*, 69(2):213–248, 2001. ISSN 03067734, 17515823. URL http://www.jstor.org/stable/1403814. 131

V. G. Vovk. Aggregating strategies. In *Proceedings of the third annual workshop on Computational learning theory*, pages 371–386, 1990. 130

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pages 928–936, 2003. 126, 128, 132