Apprentissage et prévision séquentiels : bornes uniformes pour le regret linéaire et séries temporelles hiérarchiques

Malo Huard

▶ **To cite this version:**

Malo Huard. Apprentissage et prévision séquentiels : bornes uniformes pour le regret linéaire et séries temporelles hiérarchiques. Machine Learning [stat.ML]. Université Paris-Saclay, 2020. Français. NNT : . tel-02957602v1

HAL Id: tel-02957602

https://theses.hal.science/tel-02957602v1

Submitted on 5 Oct 2020 (v1), last revised 13 Nov 2020 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

université
PARIS-SACLAY

Apprentissage et prévision séquentiels : bornes uniformes pour le regret linéaire et séries temporelles hiérarchiques

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 574, École doctorale de mathématiques Hadamard (EDMH)
Spécialité de doctorat : Mathématiques appliquées
Unité de recherche : Université Paris-Saclay, CNRS, Laboratoire de mathématiques d'Orsay, 91405, Orsay, France.
Référent : Faculté des sciences d'Orsay

Thèse présentée et soutenue à Orsay, le 23 septembre 2020, par

Malo HUARD

Composition du jury :

Olivier Wintenberger Professeur, Sorbonne Université	Président
Anne Philippe Professeure, Université de Nantes	Rapportrice et examinatrice
Paul Doukhan Professeur, Université de Cergy-Pontoise	Rapporteur et examinateur
Christophe Giraud Professeur, Université Paris-Saclay	Examinateur
Yannig Goude P.A.S.T., Université Paris-Saclay / EDF Lab Paris-Saclay	Examinateur
Gilles Stoltz Directeur de Recherche, CNRS - Université Paris-Saclay	Directeur

Thèse de doctorat

NNT : 2020UPASM009

Remerciements

Mon doctorat n'aurait pas été le même (voire pas été du tout) sans de nombreuses personnes que je tiens à remercier ici.

Tout d'abord, je tiens à te remercier Gilles, pour tout ce que tu as fait pour moi. Ce doctorat aurait été impossible sans ton soutien indéfectible malgré quelques situations difficiles. Et, non content de m'avoir aiguillé dans la découverte d'un domaine dont tu as grandement participé à la construction, tu m'as aussi permis de progresser sur mes lacunes certaines tant en rédaction qu'en gestion administrative. J'espère continuer à faire vivre ton enseignement dans ma vie future et que nous continuerons à collaborer afin d'explorer de nouvelles applications de l'agrégation séquentielle. Mille merci donc.

Je tiens aussi à remercier mes jumeaux de thèse Margaux et Hédi pour tous ces bons moments passés ensemble. Merci Margaux d'avoir collaboré avec moi malgré mes capacités rédactionnelles limitées ; et merci Hédi, c'était un bonheur de retrouver un ami de longue date dans cette aventure. Je n'oublie pas mes grands frères Sébastien et Pierre, avec qui j'ai pu avoir des discussions très enrichissantes et qui ont pris du temps pour que notre collaboration puisse être menée à terme. Merci pour votre gentillesse et votre bienveillance.

Je remercie sincèrement les membres de mon jury. Je suis très honoré qu'Anne Philippe et Paul Doukhan aient accepté de rapporter ma thèse. Merci pour l'attention que vous avez portée à la lecture de ce manuscrit et d'en avoir rédigé les rapports. Un grand merci à toi Yannig pour ta gentillesse et ton intérêt pour mon travail, je suis très honoré de te retrouver dans mon jury de thèse qui n'aurait certainement pas été la même sans ton aide. Je remercie également le reste de mon jury, Christophe Giraud, dont les commentaires bienveillants lors de mon master m'ont beaucoup servi, et Olivier Wintenberger dont l'algorithme BOA n'a pas fini de me rendre service pour les applications pratiques. Merci à tous.

Ma thèse ayant commencé comme une CIFRE et puis s'étant transformée en thèse académique, il a fallu trouver des financements sans lesquels, je n'aurais pu mener ma thèse à bout. J'en profite pour te remercier encore Gilles, car sans ton énergie c'eût été impossible. Je tiens à remercier mes collaborateurs industriels pour les projets qui ont nourris ma thèse

et qui ont aussi permis de la financer. Un grand merci à EDF ainsi que Yannig, Margaux et Sandra pour ces différents projets au cours de la thèse qui ont constitué une source non négligeable de matière pour mes recherches. Je remercie aussi Cdiscount ainsi que Rémy Garnier pour m'avoir donné accès à des données de ventes précieuses ainsi qu'à Bruno Goutorbe pour le pilotage de la collaboration. Je remercie également le Labex mathématiques Hardamard (qui a financé un an de thèse), l'AMIES, l'équipe administrative du laboratoire et de l'école doctorale, en particulier Marie-Christine Myoupo, qui ont beaucoup œuvré pour le refinancement.

Mon doctorat n'aurait pas été le même sans les personnes avec qui j'ai pu passer des moments de détente ; merci Gohar et Ronan pour tous ces bobuns partagés, Vivien pour nos randonnées philosophiques, Sam pour les oasis ardéchois, Christian pour ton accueil salvateur dans la coloc, Mathieu, Jules et Brice pour nos créations autour du jeu de la vie, Gurvan pour nos soirées à faire la guerre, Marc pour ta détente inégalable, Charles pour tes concerts de guitare, Alice pour nos footings-discussions, Jean et Estelle pour votre accueil à Oxford, mes amis de prépa, Basile, Clément, Dominique, François, Klervie, Lisa, Louis, Romain et Thomas pour avoir continué de suivre mes aventures depuis longtemps, Nicolas pour nos partages socio-musicaux, Arthur, Jean, Josquin et Vincent pour nos parties de Mario Kart salées, Garance pour nos soirées films à la coloc, ceux avec qui j'ai partagé nos déjeuners sur la péniche du crous dont Antoine, Clément et Max, mes collègues Alexandre, Mansour et Thomas pour leur soutien dans les situations stressantes, Iris pour ton aide dans la phase finale de rédaction, et tous les autres qui ont suivi mon parcours avec attention.

Enfin, un grand merci à ma famille, à mon frère Athanaric qui a été à Paris avec moi depuis le début de mes études dans le supérieur, à ma sœur Zoïlé pour sa gentillesse malgré mes défauts, mes grands-parents, Mamie pour sa générosité hors-norme, Manou pour sa vivacité, et Popère, qui n'est pas là et à qui je dois tant, mes oncles et tantes, Maxence, Mélanie, Sylvie et tous les autres qui font vivre cet esprit de famille.

Je souhaiterais pour terminer remercier mes parents, Casarie et Patrick, qui m'ont encouragé et soutenu en toutes circonstances, allant même jusqu'à m'offrir un livre sur le nombre π pour Noël ! Merci pour tout, je vous dois la réussite de ma scolarité et de cette thèse sans aucun doute.

Table des matières

CHAPITRE 1

Introduction

Ce chapitre d'introduction a pour but de présenter les contributions principales de cette thèse tout en les situant dans leur environnement scientifique. On commencera par une présentation du cadre mathématique qui est commun à la majorité de la thèse : la prévision séquentielle déterministe par agrégation. On montrera ensuite comment s'articulent les travaux théoriques de cette thèse avec les résultats techniques antérieurs. Le but de cette théorie étant de prévoir des suites individuelles (non issues d'une modélisation stochastique précise), de nombreuses applications industrielles sont envisageables ; on en présentera quelques-unes dans cette introduction ainsi que de nouvelles explorées au cours cette thèse.

Sommaire

1.1 Cadre mathématique

Le processus d'apprentissage chez les êtres vivants est éminemment lié au temps car il fonctionne par étapes empiriques successives. Les premières théories de l'apprentissage de Vapnik and Chervonenkis [1974] distinguent néanmoins deux temps, l'apprentissage sur des données historiques et l'inférence sur de nouvelles données. Dans les années 90, des travaux théoriques de Vovk [1990], Cover [1991] et Littlestone and Warmuth [1994] ont permis de déterminer comment réaliser de façon conjointe l'apprentissage et la prévision afin de tirer pleinement partie de la dimension temporelle des données. Cela a aussi permis de se libérer du besoin de les modéliser statistiquement, permettant d'appliquer ces méthodes à un grand nombre de problèmes pratiques.

1.1.1 Régime séquentiel

La prévision des suites arbitraires est un cadre statistique particulier où l'on ne fait aucune hypothèse sur la stochasticité de la suite $y_1, \dots, y_T \in \mathcal{Y}$ à prévoir. Le statisticien produit une suite de prévisions $\widehat{y}_1, \dots, \widehat{y}_T \in \mathcal{Y}$ à l'aide de l'information accessible au temps t, c'est-à-dire $y_1, \dots, y_{t-1}$ ainsi qu'une suite d'observations $\mathbf{x}_1, \dots, \mathbf{x}_t \in \mathcal{X}$. Dans cette introduction, on choisira $\mathcal{Y} = \mathbb{R}$ et $\mathcal{X} = \mathbb{R}^d$; la notation en caractère gras mettant en évidence les vecteurs. L'écart entre les prévisions et les observations est mesuré à l'aide d'une fonction de perte $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. En général, et c'est le cas pour cette thèse, on supposera la fonction de perte convexe en son premier argument. L'objectif du statisticien est de minimiser la perte cumulée

$$\sum_{t=1}^{T} \ell(\widehat{y}_t, y_t) \, .$$

Exemples de fontions de perte. Les fonctions de perte convexes typiques pour la plupart des applications sont la perte quadratique $(x, y) \mapsto (x-y)^2$, la perte absolue $(x, y) \mapsto |x-y|$ ou le pourcentage d'erreur absolue $(x, y) \mapsto |x - y|/|y|$.

On peut résumer le processus d'apprentissage et de prévision de la façon suivante :

Pour $t = 1, 2, 3, \dots$:
- Le statisticien observe $\mathbf{x}_t$
- Le statisticien construit une prévision $\widehat{y}_t$
- Le statisticien observe y_t
- Le statisticien subit une perte $\ell(\widehat{y}_t, y_t)$

Le lecteur trouvera une très bonne introduction à ce domaine dans la monographie *Prediction, Learning, and Games* de Cesa-Bianchi and Lugosi [2006].

1.1.2 Prévision séquentielle par agrégation d'experts

Lorsque les $x_{1\,t}, \dots, x_{d\,t} \in \mathbb{R}$ sont des prévisions élémentaires (tels que pour tout i, $x_{i\,t}$ est proche de y_t), on parle d'agrégation d'experts. C'est pourquoi, dans ce cadre, la prévision des suites arbitraires devient la prévision séquentielle déterministe par agrégation. L'objectif le plus simple lorsque qu'on dispose d'un panel d'experts est d'en déterminer le meilleur.

On verra par la suite qu'on pourra faire presque aussi bien que la meilleure combinaison convexe, ou la meilleure combinaison linéaire.

Pourquoi a-t-on besoin de garanties ? On peut se demander pourquoi on a besoin d'une théorie et d'algorithmes alors que le simple choix du meilleur expert dans le passé peut sembler suffire. Pour voir cela, prenons un rapide exemple avec deux experts A et B. On note dans le tableau suivant les erreurs de chacun des experts, ainsi que celles de la méthode naïve consistant à choisir celui ayant été le meilleur à la dernière observation :

	1			$\rightarrow$			T	Perte cumulée
A	$1/4$	1	0	1	0			$\sim T/2$
B	$1/2$	0	1	0	1			$\sim T/2$
Meilleur expert	A	B	A	B	A	$\cdots$		
Choix		A	B	A	B			
Erreur		1	1	1	1			$\sim T$

On voit dans cet exemple que si l'on avait choisi l'un des experts pour tous les pas de temps, on aurait fait une erreur de l'ordre de $T/2$ alors qu'avec la méthode naïve on a une erreur cumulée de l'ordre de T.

En considérant une stratégie $\mathcal{S}$ définie à l'avance, on va évaluer sa capacité à être aussi performante que le meilleur expert. Pour ce faire, on divise l'erreur cumulée en 2 termes, un terme correspondant à la perte du meilleur expert (l'erreur d'approximation) et un autre que l'on appelle le regret d'une stratégie (l'erreur d'estimation) défini par $\mathcal{R}_T^{\text{sel}}(\mathcal{S}) \stackrel{\text{def}}{=} \sum_{t=1}^T \ell(\widehat{y}_t, y_t) - \min_{1\leqslant i\leqslant d} \sum_{t=1}^T \ell(x_{i\,t}, y_t)$. Le terme « sel » (pour « sélection ») dans la notation du regret est dû au fait que l'on se compare au meilleur expert sur la période considérée. On a, pour la perte cumulée, la décomposition suivante

$$\sum_{t=1}^T \ell(\widehat{y}_t, y_t) = \min_{1\leqslant i\leqslant d} \sum_{t=1}^T \ell(x_{i\,t}, y_t) + \mathcal{R}_T^{\text{sel}}(\mathcal{S})\,.$$

On recherche des stratégies d'estimation ayant un regret sous-linéaire $\mathcal{R}_T^{\text{sel}}(\mathcal{S}) = o(T)$, ou plus formellement :

$$\limsup_{T\to+\infty} \left(\sup_{y_{1:T}, \mathbf{x}_{1:T}} \left\{ \mathcal{R}_T^{\text{sel}}(\mathcal{S})/T \right\} \right) \leqslant 0\,, \tag{1.1}$$

où le second $\sup$ correspond à la borne supérieure sur toutes les suites $y_{1:T} = (y_1, \ldots, y_T)$ et $\mathbf{x}_{1:T} = (\mathbf{x}_1, \ldots, x_T)$. On impose donc sur les stratégies, la contrainte très forte, d'être uniformément performantes sur toutes ces suites. Cela permet de se passer d'une modélisation stochastique, même si en général, on les suppose tout de même bornées.

1.1.3 Mélanges par poids exponentiels

Une des premières stratégies introduites, provenant des travaux pionniers de Vovk [1990] et Littlestone and Warmuth [1994], consiste à utiliser des mélanges de poids exponentiels, d'où son nom EWA[1]. On calcule les prévisions par agrégation convexe. À l'étape t, on choisit des

[1] Exponential Weighted Average

poids $\widehat{u}_{1\,t}, \dots, \widehat{u}_{d\,t}$ et on calcule la prévision

$$\widehat{y}_t = \sum_{i=1}^{T} \widehat{u}_{i\,t} x_{i\,t} \,.$$

Définir une stratégie dans ce cadre c'est donner la règle de calcul des poids. Ceux de la stratégie EWA sont calculés de la façon suivante, pour chaque $i \in \{1, \dots, d\}$,

$$\widehat{u}_{i\,t} = \frac{\exp\Big(- \eta \sum_{s=1}^{t-1} \ell(x_{i\,s}, y_t)\Big)}{\sum_{j=1}^{d} \exp\Big(- \eta \sum_{s=1}^{t-1} \ell(x_{j\,s}, y_t)\Big)} \,. \tag{1.2}$$

où $\eta > 0$ est un paramètre de la stratégie relatif à la vitesse d'apprentissage. On voit que cette stratégie est une forme adoucie de la sélection du meilleur expert. En effet, si l'on prend $\eta = 0$, alors, on effectue une moyenne uniforme des experts et si l'on fait tendre η vers $+\infty$ on sélectionne le meilleur. Le regret de cette stratégie est donné par le théorème suivant, devenu un grand classique du domaine (voir Cesa-Bianchi and Lugosi, 2006, paragraphe 2.2 pour une preuve).

Théorème 1.1. *En supposant qu'il existe $B > 0$ tel que pour tout $y_t \in \mathcal{Y}$ et $x_{i\,t} \in \mathcal{X}$ les pertes sont bornées, i.e., $\big|\ell(x_{i\,t}, y_t)\big| < B$, on a*

$$\mathcal{R}_T^{\text{sel}}(\text{EWA}) = \sum_{t=1}^{T} \ell(\widehat{y}_t, y_t) - \min_{1 \leqslant i \leqslant d} \sum_{t=1}^{T} \ell(x_{i\,t}, y_t) \leqslant \frac{\ln d}{\eta} + \frac{\eta T B^2}{2} \,.$$

En choisissant $\eta = (1/B)\sqrt{(2 \ln d)/T}$ dans le terme de droite, on obtient comme borne $\mathcal{R}_T^{\text{sel}}(\text{EWA}) \leqslant B\sqrt{2T \ln d}$, qui est bien sous-linéaire.

1.1.4 Mélanges convexes

On a vu dans la partie précédente un algorithme de mélange qui contrôle le regret par rapport au meilleur expert. Dans cette partie, nous allons voir comment étendre ces stratégies (tout en restant des algorithmes de mélange) pour se comparer à la meilleure combinaison convexe des experts. On définit le regret par rapport à la meilleure combinaison convexe pour une stratégie $\mathcal{S}$ par $\mathcal{R}_T^{\text{cvx}}\big(\widetilde{\mathcal{S}}\big) \stackrel{\text{def}}{=} \sum_{t=1}^{T} \ell(\widehat{y}_t, y_t) - \min_{\mathbf{u} \in \Delta_d} \sum_{t=1}^{T} \ell(\mathbf{u} \cdot \mathbf{x}_t, y_t)$. On a noté Δ_d le d-simplexe et « cvx » pour « convexe ». On a alors la décomposition de la perte cumulée suivante

$$\sum_{t=1}^{T} \ell(\widehat{y}_t, y_t) = \min_{\mathbf{u} \in \Delta_d} \sum_{t=1}^{T} \ell(\mathbf{u} \cdot \mathbf{x}_t, y_t) + \mathcal{R}_T^{\text{cvx}}\big(\widetilde{\mathcal{S}}\big) \,. \tag{1.3}$$

La technique d'extension, appelée « gradient trick », est due à Kivinen and Warmuth [1997] et Cesa-Bianchi [1999]. Elle permet de transformer une stratégie $\mathcal{S}$ (de mélange) servant à contrôler le regret $\mathcal{R}_T^{\text{sel}}(\mathcal{S})$ en une stratégie $\widetilde{\mathcal{S}}$ (de mélange) qui contrôle le regret $\mathcal{R}_T^{\text{cvx}}\big(\widetilde{\mathcal{S}}\big)$. Pour cela, il suffit de remplacer les pertes $\ell(x_{i\,s}, y_t)$ dans l'équation (1.2) par les pseudo-pertes $\ell'(\widehat{y}_t, y_t) x_{i\,t}$, où $\ell'(\widehat{y}_t, y_t)$ est un sous gradient de la fonction convexe $x \mapsto \ell(x, y_t)$. On trouvera plus de détails sur cette technique en partie 3.6.3. En appliquant le

« gradient trick » à l'algorithme EWA, on obtient l'algorithme EG[2] dont les poids sont

$$\widehat{u}_{it} = \frac{\exp\Big(-\eta \sum_{s=1}^{t-1} \ell'(\widehat{y}_s, y_s) x_{i\,s}\Big)}{\sum_{j=1}^{d} \exp\Big(-\eta \sum_{s=1}^{t-1} \ell'(\widehat{y}_s, y_s) x_{j\,s}\Big)} \,.$$

Ce faisant, on obtient la même borne que le Théorème 1.1 mais sur le regret $\mathcal{R}_T^{\text{cvx}}$ où cette fois-ci, la constante B est une borne sur les pseudo-pertes. On pourra trouver une démonstration de ce résultat dans Cesa-Bianchi and Lugosi [2006, partie 2.5].

Théorème 1.2. *En supposant qu'il existe $B > 0$ tel que pour tout $y_t \in \mathcal{Y}$ et $x_{i\,t} \in \mathcal{X}$ les pertes sont bornées selon $\big|\ell'(\widehat{y}_t, y_t) x_{i\,t}\big| < B$, on a*

$$\mathcal{R}_T^{\text{cvx}}(\text{EG}) = \sum_{t=1}^{T} \ell(\widehat{y}_t, y_t) - \min_{\mathbf{u} \in \Delta_d} \sum_{t=1}^{T} \ell(\mathbf{u} \cdot \mathbf{x}_t, y_t) \leqslant \frac{\ln d}{\eta} + \frac{\eta T B^2}{2} \,.$$

De la même façon qu'à la partie précédente, en choisissant $\eta = (1/B)\sqrt{(2 \ln d)/T}$, on obtient comme borne $\mathcal{R}_T^{\text{cvx}}(\text{EG}) \leqslant B\sqrt{2T \ln d}$, ce qui montre que le regret est bien sous-linéaire.

1.1.5 Mélanges linéaires

Les deux algorithmes d'agrégation précédents conviennent très bien lorsque les experts ne sont pas biaisés. Cependant, comme ils utilisent des poids convexes, ils ne permettent pas de s'adapter au cas où un expert (ou plusieurs) sous ou sur-estime la variable cible. Pour résoudre ce problème, il faut relâcher la contrainte sur le vecteur de poids. Dans ce but, nous généralisons la notion de regret : pour une stratégie $\mathcal{S}$, on définit le regret par rapport à un vecteur de comparaison donné $\mathbf{u} \in \mathbb{R}^d$ par $\mathcal{R}_T^{\text{lin}}(\mathcal{S}, \mathbf{u}) \stackrel{\text{def}}{=} \sum_{t=1}^{T} \ell(\widehat{y}_t, y_t) - \sum_{t=1}^{T} \ell(\mathbf{u} \cdot \mathbf{x}_t, y_t)$, où l'on a noté « lin » pour « linéaire ». En notant, pour un sous-ensemble $\mathcal{D} \subset \mathbb{R}^d$, la borne supérieure sur son regret $\mathcal{R}_T^{\text{lin}}(\mathcal{S}, \mathcal{D}) \stackrel{\text{def}}{=} \sup_{\mathbf{u} \in \mathcal{D}} \mathcal{R}_T^{\text{lin}}(\mathcal{S}, \mathbf{u})$, on a la décomposition suivante

$$\sum_{t=1}^{T} \ell(\widehat{y}_t, y_t) = \inf_{\mathbf{u} \in \mathcal{D}} \sum_{t=1}^{T} \ell(\mathbf{u} \cdot \mathbf{x}_t, y_t) + \mathcal{R}_T^{\text{lin}}(\mathcal{S}, \mathcal{D}) \,.$$

On remarquera que si l'on prend $\mathcal{D} = \Delta_d$ on retrouve l'équation (1.3). Il est généralement nécessaire de prendre pour l'ensemble $\mathcal{D}$, un sous-ensemble compact de $\mathbb{R}^d$. Pour contrôler ce regret dans le cadre spécifique de la perte quadratique $\ell(\widehat{y}_t, y_t) = (y_t - \widehat{y}_t)^2$, une première proposition a été d'adapter l'algorithme ridge de Hoerl and Kennard [1970] au cadre séquentiel. Cette stratégie, *ridge regression*, consiste à faire le choix des poids de mélange suivant

$$\widehat{\mathbf{u}}_t \in \operatorname*{argmin}_{\mathbf{u} \in \mathbb{R}^d} \left\{ \sum_{s=1}^{t-1} (y_s - \mathbf{u} \cdot \mathbf{x}_s)^2 + \lambda \|\mathbf{u}\|^2 \right\} \,,$$

où $\lambda > 0$ est un paramètre de régularisation (qui peut être compris comme l'inverse de la vitesse d'apprentissage) et $\|\mathbf{u}\|$ est la norme euclidienne du vecteur $\mathbf{u}$. Il existe une borne sur le regret de cette stratégie mais celle-ci n'est pas satisfaisante car elle dépend de la quantité

[2] Exponentiated Gradient

$\max_{t \leqslant T}(y_t - \widehat{y}_t)^2$ et donc de la performance de l'algorithme. Une amélioration (en terme de garantie sur le regret), due à Vovk [2001] et Azoury and Warmuth [2001], propose d'ajouter une nouvelle régularisation sous la forme du terme $(\mathbf{u} \cdot \mathbf{x}_t)^2$. Cette stratégie, que l'on appelle *non-linear ridge regression* (noté « nl-ridge »), utilise les poids de mélange suivant :

$$\widehat{\mathbf{u}}_t \in \operatorname*{argmin}_{\mathbf{u} \in \mathbb{R}^d} \left\{ \sum_{s=1}^{t-1} (y_s - \mathbf{u} \cdot \mathbf{x}_s)^2 + (\mathbf{u} \cdot \mathbf{x}_t)^2 + \lambda \|\mathbf{u}\|^2 \right\}. \tag{1.4}$$

On dit cet algorithme non-linéaire car, les poids de mélange $\widehat{\mathbf{u}}_t$ dépendant du vecteur $\mathbf{x}_t$, la prévision $\widehat{y}_t = \widehat{\mathbf{u}}_t \cdot \mathbf{x}_t$ ne dépend plus linéairement de $\mathbf{x}_t$. Pour énoncer le résultat classique sur cet algorithme, on introduit quelques notations techniques.

Notation 1.3. *Pour $\mathbf{x}_1, \mathbf{x}_2, \ldots \in \mathbb{R}^d$, on appelle $\mathbf{G}_t = \sum_{s=1}^t \mathbf{x}_s \mathbf{x}_s^{\mathrm{T}}$ la matrice (carrée) de Gram de taille d à l'étape $t \geqslant 1$. Cette matrice est symétrique définie positive et admet d valeurs propres, que l'on trie par ordre croissant et que l'on note $\lambda_1(\mathbf{G}_t), \ldots, \lambda_d(\mathbf{G}_t)$. Pour tout $t \geqslant 1$, on notera $r_t = \operatorname{rank}(\mathbf{G}_t)$ le rang de $\mathbf{G}_t$.*

Le théorème suivant (énoncé au chapitre 2, avec précision des références originelles) donne une inégalité sur le regret de l'algorithme *non-linear ridge regression*.

Théorème 1.4 (voir Théorème 2.2)**.** *En prenant $\lambda > 0$ et $B > 0$, on a pour tout $T \geqslant 1$, toutes suites $\mathbf{x}_1, \ldots, \mathbf{x}_T \in \mathbb{R}^d$ et $y_1, \ldots, y_T \in [-B, B]$,*

$$\forall \mathbf{u} \in \mathbb{R}^d, \quad \mathcal{R}_T^{\text{lin}}(\text{nl-ridge}, \mathbf{u}) \leqslant \lambda \|\mathbf{u}\|^2 + B^2 \sum_{k=1}^{d} \ln\left(1 + \frac{\lambda_k(\mathbf{G}_T)}{\lambda}\right).$$

On peut simplifier ce résultat en éliminant les quantités liées à la matrice de Gram et en prenant l'ensemble $\mathcal{U} = \{\mathbf{u} \in \mathbb{R}^d \mid \|\mathbf{u}\| < U\}$, où $U > 0$. Cela donne le corollaire suivant.

Corollaire 1.5. *En supposant qu'il existe $B > 0$ et $X > 0$ tels que pour tout $y_t \in \mathcal{Y}$ et $x_{it} \in \mathcal{X}$, la cible et les observations sont bornées, c'est à dire $|y_t| < B$ et $\|\mathbf{x}_t\| < X$ pour tout $t \geqslant 1$, on a*

$$\mathcal{R}_T^{\text{lin}}(\text{ridge-nl}, \mathcal{U}) \leqslant \lambda U^2 + dB^2 \ln\left(1 + \frac{TX^2}{d\lambda}\right).$$

La démonstration du Théorème 1.4 reprenant les arguments de Cesa-Bianchi and Lugosi [2006] se trouve en partie 2.6. Les idées servant à démontrer le Corollaire 1.5 se trouvent à la fin de la démonstration du Corollaire 2.3 dans cette même partie.

On notera que l'algorithme *non-linear ridge regression* ne contrôle pas directement le regret $\mathcal{R}_T^{\text{lin}}\big(\text{ridge-nl}, \mathbb{R}^d\big)$, en effet le supremum sur $\mathbf{u} \in \mathbb{R}^d$ du terme de droite vaut $+\infty$. Pour obtenir une borne uniforme sur le regret, il faut, comme pour le Corollaire 1.5, restreindre l'ensemble des vecteurs $\mathbf{u}$ possibles à un sous-ensemble borné de $\mathbb{R}^d$. Le chapitre 2 s'attache à résoudre ce problème et à obtenir des bornes de regret uniformes sur $\mathbb{R}^d$.

1.1.6 Regret linéaire uniforme sur $\mathbb{R}^d$ (chapitre 2)

On se place dans le cadre des mélanges linéaires décrit à la partie précédente ; notre but est de construire une stratégie $\mathcal{S}$ maîtrisant le regret $\mathcal{R}_T^{\text{lin}}\big(\mathcal{S}, \mathbb{R}^d\big)$. Ici et dans la suite de la partie, B est une borne sur les observations telle que pour tout $t \leqslant T$ on ait $|y_t| \leqslant B$. Dans la suite on distinguera deux cadres, celui présenté précédemment où les prévisions d'experts $\mathbf{x}_t$ sont découvertes au fur et à mesure, et l'autre où elles sont toutes connues avant le début des prévisions. Ce second cadre permettra d'obtenir de meilleures bornes, mais il est évidemment moins naturel.

Quand les vecteurs $\mathbf{x}_t$ ne sont pas connus à l'avance

On obtient le corollaire suivant en combinant le Théorème 1.4 et le fait que la norme du vecteur atteignant la borne inférieure $\inf_{\mathbf{u}\in\mathbb{R}^d} \sum_{t=1}^T (y_t - \mathbf{u}\cdot\mathbf{x}_t)^2$ ne peut être trop grande. La démonstration complète se trouve en partie 2.6.

Corollaire 1.6 (voir Corollaire 2.3)**.** *Pour l'algorithme non-linear ridge regression* (1.4) *utilisant $\lambda > 0$, pour tout $T \geqslant 1$, pour tout $\mathbf{x}_1, \dots, \mathbf{x}_T \in \mathbb{R}^d$ avec $\|\mathbf{x}_t\| \leqslant X$ et tout $y_1, \dots, y_T \in [-B, B]$,*

$$\mathcal{R}_T^{\text{lin}}\big(\text{ridge-nl}, \mathbb{R}^d\big) \leqslant r_T B^2 \ln\left(1 + \frac{TX^2}{r_T\lambda}\right) + \frac{\lambda}{\lambda_{r_T}(\mathbf{G}_T)} TB^2 \,.$$

où $r_T = \operatorname{rank}(\mathbf{G}_T)$ et donc $\lambda_{r_T}(\mathbf{G}_T)$ est la plus petite valeur propre non nulle de $\mathbf{G}_T$.

En appliquant ce corollaire avec $\lambda = 1/T$, on trouve que l'algorithme *non-linear ridge regression* admet une borne de l'ordre de $2dB^2 \ln T + \mathcal{O}_T(1)$ où le terme $\mathcal{O}_T(1)$ cache une quantité variant en fonction de la suite $\mathbf{x}_1, \dots, \mathbf{x}_T$.

La borne ici n'est pas complètement satisfaisante car elle dépend fortement de la suite $\mathbf{x}_1, \dots, \mathbf{x}_T$ par le terme $\lambda_{r_T}(\mathbf{G}_T)$, qui est la plus petite valeur propre non nulle de la matrice de covariance et qui peut être arbitrairement petite. Ainsi, elle n'est pas sous-linéaire au sens de l'équation (1.1). Nous n'avons malheureusement pas réussi à remédier à cela mais nous avons obtenu une borne sensiblement meilleure par une analyse plus fine de la démonstration du Théorème 1.4. Cela nous a poussé à introduire l'algorithme suivant (qu'on note « ridge-nl-0 ») qui n'est autre que l'algorithme ridge avec $\lambda = 0$ dans l'équation (1.4). Cela donne : $\widehat{\mathbf{u}}_1 = (0, \dots, 0)^{\mathrm{T}}$ et pour $t \geqslant 2$,

$$\widehat{\mathbf{u}}_t \in \operatorname*{argmin}_{\mathbf{u}\in\mathbb{R}^d} \left\{ \sum_{s=1}^{t-1} (y_s - \mathbf{u}\cdot\mathbf{x}_s)^2 + (\mathbf{u}\cdot\mathbf{x}_t)^2 \right\} . \tag{1.5}$$

Cet algorithme a l'avantage d'être invariant par transformation d'échelle, c'est-à-dire que la multiplication de tous les vecteurs $\mathbf{x}_1, \dots, \mathbf{x}_T$ par une même matrice inversible ne change pas les prévisions. Le théorème suivant, qui est une adaptation du Théorème 1.4 donne des garanties sur le regret linéaire uniforme de cet algorithme.

Théorème 1.7 (voir le Théorème 2.6)**.** *L'algorithme non-linear ridge regression avec* $\lambda = 0$ *comme dans* (1.5) *admet pour tout* $T \geqslant 1$*, pour tout* $\mathbf{x}_1, \dots, \mathbf{x}_T \in \mathbb{R}^d$ *et tout* $y_1, \dots, y_T \in [-B, B]$*, la borne de regret uniforme*

$$\mathcal{R}_T^{\text{lin}}\Big(\text{ridge-nl-0}, \mathbb{R}^d\Big) \leqslant B^2 \sum_{k=1}^{r_T} \ln\Big(\lambda_k(\mathbf{G}_T)\Big) + B^2 \sum_{t \in [\![1,T]\!] \cap \mathcal{T}} \ln\left(\frac{1}{\lambda_{r_t}(\mathbf{G}_t)}\right) + r_T B^2$$

où r_t *et* λ_k *sont définis à la Notation 1.3, et où l'ensemble* $\mathcal{T}$ *contient les* r_T *étapes, données par le plus petit* $s \geqslant 1$ *pour lequel* $\mathbf{x}_s$ *n'est pas nul, et par tous les* $s \geqslant 2$ *tels que* $\operatorname{rank}(\mathbf{G}_{s-1}) \neq \operatorname{rank}(\mathbf{G}_s)$.

Par des arguments standards, on peut obtenir de ce théorème la borne suivante pour tout $X > 0$ et pour toute suite $\mathbf{x}_1, \mathbf{x}_2, \dots$ telle que $\|\mathbf{x}_t\| \leqslant X$,

$$\mathcal{R}_T^{\text{lin}}\Big(\text{ridge-nl-0}, \mathbb{R}^d\Big) \leqslant dB^2 \ln T + \underbrace{dB^2 + B^2 \sum_{t \in [\![1,T]\!] \cap \mathcal{T}} \ln\left(\frac{X^2}{\lambda_{r_t}(\mathbf{G}_t)}\right)}_{\mathcal{O}_T(1)}.$$

On notera que le terme $\mathcal{O}_T(1)$ cesse de croître lorsque la matrice $\mathbf{G}_T$ est de rang plein : ensuite, pour les étapes suivantes $T' \geqslant T$, seul le terme principal augmente en $dB^2 \ln T'$. Mais, comme pour celle du Corollaire 1.6, cette borne peut être arbitrairement grande si l'on prenait le supremum du regret aussi sur les suites $\mathbf{x}_1, \dots, \mathbf{x}_T$ possibles. Cependant, la dépendance en les valeurs propres des matrices de Gram est améliorée car logarithmique.

Quand les vecteurs $\mathbf{x}_t$ sont connus à l'avance

Dans la continuité des travaux de Bartlett et al. [2015], on s'est intéressé à un processus particulier où les vecteurs $\mathbf{x}_t$ sont connus à l'avance.

> Sachant $T > 0$ et $\mathbf{x}_1, \dots, \mathbf{x}_T \in \mathbb{R}^d$
> **Pour** $t = 1, 2, \dots, T$:
> - Le statisticien construit une prévision $\widehat{y}_t$
> - Le statisticien observe y_t
> - Le statisticien subit une perte $(\widehat{y}_t - y_t)^2$

Dans ce cadre, Bartlett et al. [2015] construisent un algorithme qui est minimax optimal en imposant de fortes conditions sur les suites $(\mathbf{x}_t)_{t \geqslant 1}$ et $(y_t)_{t \geqslant 1}$. Néanmoins, leur meilleure borne obtenue sans faire d'hypothèses particulières sur la nature de ces suites est de l'ordre de $2dB^2 \ln T$. Notre travail a consisté à adapter l'algorithme *non-linear ridge regression* pour obtenir de bonnes garanties dans ce cadre. Nous avons montré qu'en modifiant le terme de régularisation pour l'adapter aux données, on pouvait obtenir un algorithme ayant une borne sur le regret de l'ordre $dB^2 \ln T$; ce qui est optimal d'après une borne inférieure énoncée par Vovk [2001] et que nous rappellerons au Théorème 1.9. Ainsi, l'algorithme *non-linear ridge regression with adapted regularization* (que l'on note « ridge-adapt ») choisit des vecteurs de mélanges de la façon suivante : $\widehat{\mathbf{u}}_1 = (0, \dots, 0)^{\intercal}$ et pour $t \geqslant 2$,

$$\widehat{\mathbf{u}}_t \in \operatorname*{argmin}_{\mathbf{u} \in \mathbb{R}^d} \left\{ \sum_{s=1}^{t-1} (y_s - \mathbf{u} \cdot \mathbf{x}_s)^2 + (\mathbf{u} \cdot \mathbf{x}_t)^2 + \lambda \sum_{s=1}^{T} (\mathbf{u} \cdot \mathbf{x}_s)^2 \right\} \tag{1.6}$$

Cet algorithme diffère des algorithmes ridge standards (*ridge regression* et *non-linear ridge regression*) en adaptant la norme de la pénalisation aux données. Cela lui confère la propriété d'être invariant par transformation d'échelle, comme c'est le cas de l'algorithme défini par l'équation (1.5). C'est d'ailleurs cela qui nous permet de standardiser dans l'analyse ces vecteurs en les multipliant par la matrice $\mathbf{G}_T^{-1/2}$ de sorte à faire tourner l'algorithme sur les vecteurs $\widetilde{\mathbf{x}}_t = \mathbf{G}_T^{-1/2}\mathbf{x}_t$. Dans ce cas, on peut montrer que l'algorithme *non-linear ridge regression with adapted regularization* est en fait une instance de l'algorithme ridge original dont les vecteurs $\widetilde{\mathbf{x}}_t$ ont pour matrice de Gram l'identité. Cela permet d'appliquer le Théorème 1.4 et d'obtenir une meilleure borne (de l'ordre de $dB^2 \ln T$). Ce schéma de démonstration est détaillé de façon plus approfondie en partie 2.3.

Théorème 1.8 (voir Théorème 2.5)**.** *Pour l'algorithme non-linear ridge regression with adapted regularization* (1.6) *utilisant $\lambda > 0$, pour tout $T \geqslant 1$, pour tout $\mathbf{x}_1, \dots, \mathbf{x}_T \in \mathbb{R}^d$ et tout $y_1, \dots, y_T \in [-B, B]$,*

$$\mathcal{R}_T^{\text{lin}}\Big(\text{ridge-adapt}, \mathbb{R}^d\Big) \leqslant \lambda T B^2 + r_T B^2 \ln\left(1 + \frac{1}{\lambda}\right),$$

où $r_T = \operatorname{rank}(\mathbf{G}_T)$.

En prenant $\lambda = r_T/T$, on obtient la borne $r_T B^2\Big(1 + \ln(1 + T/r_T)\Big)$ qui par un rapide calcul se simplifie en

$$\mathcal{R}_T^{\text{lin}}\Big(\text{ridge-adapt}, \mathbb{R}^d\Big) \leqslant d\, B^2 \ln\left(1 + \frac{T}{d}\right) + d\, B^2\,.$$

On obtient donc bien une borne de l'ordre de $dB^2 \ln T$. On notera que la constante est 1, ce qui est optimal comme nous allons le discuter maintenant.

Borne inférieure

Afin de savoir si les algorithmes proposés sont optimaux, il est intéressant de déterminer quel est le regret minimal possible (pour toutes les stratégies $\mathcal{S}$) si l'on considère l'ensemble des suites $\mathbf{x}_1, \dots, \mathbf{x}_T$ et $y_1, \dots, y_T$. Nous avons légèrement amélioré un résultat dû à Vovk [2001] tout en modifiant une partie de la démonstration. Celle-ci s'appuie sur l'inégalité de van Trees (voir Gill and Levit, 1995), qui borne inférieurement l'erreur de n'importe quel estimateur. C'est une extension de l'inégalité de Cramér-Rao pour des estimateurs possiblement biaisés.

Théorème 1.9 (voir Théorème 2.4)**.** *Pour tout $T \geqslant 8$ et $B > 0$, on a*

$$\inf_{\mathcal{S}} \sup_{\mathbf{x}_t \in \mathbb{R}^d} \sup_{y_t \in [-B,B]} \mathcal{R}_T^{\text{lin}}\Big(\mathcal{S}, \mathbb{R}^d\Big) \geqslant dB^2\Big(\ln T - (3 + \ln d) - \ln\ln T\Big)\,.$$

La borne inférieure sur les stratégies $\mathcal{S}$ n'est pas limitée aux seules méthodes par agrégation, elle est générale. Ce théorème est valide si les $\mathbf{x}_1, \dots, \mathbf{x}_T$ sont connus à l'avance. Ainsi, il permet d'assurer que notre algorithme *non-linear ridge regression with adapted regularization* est optimal, i.e., sans davantage de contraintes sur la nature des suites, on ne peut pas faire mieux. Lorsque les $\mathbf{x}_1, \dots, \mathbf{x}_T$ ne sont pas connus à l'avance, il existe un écart

entre la borne de regret pour l'algorithme *non-linear ridge regression* avec $\lambda = 0$ et la borne inférieure : soit la stratégie *non-linear ridge regression* est sous-optimale (ou la borne sur son regret), soit il faut payer une prix pour ne pas connaître les $\mathbf{x}_1, \ldots, \mathbf{x}_T$ à l'avance.

1.2 Motivations pratiques et applications

Les méthodologies d'apprentissage traditionnelles fonctionnent en ajustant un modèle sur un ensemble d'entraînement constitué de données historiques. Il s'agit ensuite d'en faire l'inférence au fur et à mesure que les nouvelles données arrivent. C'est pourquoi encore aujourd'hui un grand nombre de statisticiens, lorsqu'ils construisent des modèles de séries temporelles, les traitent comme un ensemble d'observations non ordonnées. Ils construisent donc un modèle sur un ensemble d'entraînement (passé) et le test sur un autre ensemble (futur). Les méthodologies séquentielles décrites précédemment permettent d'aller plus loin en mettant à jour constamment le modèle, de façon contrôlée, ce qui donne lieu à des prévisions améliorées sans perdre en robustesse. C'est pourquoi, ces techniques sont très utiles dans la pratique industrielle où les séries temporelles sont extrêmement communes.

1.2.1 Succès de la méthodologie séquentielle

Les méthodes d'apprentissage séquentielle ne requièrent pas d'hypothèse sur la stochasticité des données, ce qui a permis à des applications dans des domaines variés de voir le jour. Sans être totalement exhaustif voici un certain nombre d'entre elles. Après des premières expérimentations sur la prévision de la qualité de l'air par Mallet et al. [2009], une application majeure a été la prévision de la consommation électrique (voir Goude, 2008). On pourra se référer aux articles méthodologiques Devaine et al. [2013] et Gaillard and Goude [2015] ainsi qu'au chapitre 3. Ces techniques ont démontré leur efficacité lors des compétitions internationales GEFCOM (voir Gaillard et al., 2016 à ce sujet), ou lors d'une compétition organisée par RTE[3] (voir la partie 1.2.4) et sont utilisées de façon industrielle par des entreprises comme EDF[4]. Depuis, ces méthodes ont été aussi appliquées pour la prévision de taux de change par Amat et al. [2018] ou la production de pétrole ou gaz par Raphaël et al. [2019]. Ces applications multiples sont facilitées par la librairie R dénommée Opera, développée par Gaillard and Goude [2016].

La méthodologie générale autour de l'agrégation séquentielle est commune dans toutes ces applications. Il faut néanmoins procéder à certaines adaptations pour répondre à des particularités spécifiques à chacune d'entre elles. Dans nos travaux d'application de cette méthodologie au e-commerce, commencés en début de thèse dans une start-up (voir partie 1.3), nous avons rapidement buté sur quelques difficultés spécifiques et décelé les questions théoriques et pratiques cruciales pour obtenir de bons résultats. L'idée qui a rapidement émergé fut d'utiliser la structure des données pour apprendre plus rapidement et augmenter les performances en prévisions. En particulier, nous nous sommes intéressés aux structures hiérarchiques, partant de l'observation que de nombreuses séries temporelles peuvent être organisées sous une forme d'arbre.

[3] Réseau de Transport d'Électricité

[4] Électricité de France

Exemple de données temporelles hiérarchiques. Les données démographiques sont organisées en régions hiérarchisées ; ainsi la population française se répartit sur 18 régions, on peut donc construire un arbre à 19 nœuds où la racine est la France et est reliée à 18 feuilles (les régions). Les données de ventes sont souvent structurées par famille (sous-famille, etc.) reflétant les comportements communs de leurs produits. Les outils de jardin se vendent par exemple mieux en été qu'en hiver, ce qui est l'inverse pour les jeux-vidéos. La consommation électrique peut, comme les données démographiques, être répartie par zones géographiques mais aussi par types de comportement, ces différents comportements pouvant être dus, par exemple, à une tarification différente.

Dans ces hiérarchies, la valeur observée d'un nœud (exemple : une famille de produits) est égale à la somme de ses nœuds enfants (exemple : une sous-famille de produits). Ainsi, on pourrait remplacer la prévision d'un nœud parent par la somme des prévisions de ses nœuds enfants. Notre intuition était qu'il est parfois plus simple d'apprendre les dynamiques à des niveaux plus fins car plus spécifiques, avec moins d'effets exogènes. Par exemple, le surcroît de ventes d'outils de jardinage en été est plus net si on s'intéresse uniquement à cette famille de produits. Pour la consommation électrique, on dispose d'une prévision météo plus adaptée si on se limite à une région donnée. Néanmoins, comme les données des nœuds enfants sont généralement plus bruitées car agrégeant un nombre plus faible d'observations, il nous fallait trouver une façon robuste de les utiliser.

1.2.2 Prévisions hiérarchiques (chapitre 3)

Les résultats connus précédemment le sont généralement pour des séries temporelles réelles $\mathbf{y}_1, \dots, \mathbf{y}_T \in \mathcal{Y} = \mathbb{R}$, et si l'on doit en prédire plusieurs, cela se fait indépendamment pour chaque série. Il n'y a pas d'opposition fondamentale dans le cadre et dans les principaux résultats théoriques (dont le théorème 1.1 et le théorème 1.2) à prendre $\mathcal{Y} = \mathbb{R}^d$. Néanmoins cela impose, pour toutes les séries temporelles prévues, de prendre les mêmes poids de mélange. On perd ici un avantage significatif de ces méthodes qui est de sélectionner automatiquement le meilleur modèle pour chacune des séries (dans le chapitre 3 ce nombre est d'environ 3000). On a donc cherché une façon de conserver cette capacité tout en utilisant la hiérarchie afin d'améliorer les prévisions. Un autre objectif recherché était de produire des prévisions cohérentes (c'est-à-dire respectant les relations de sommation), ce qui est plus rassurant pour les opérateurs devant prendre des décisions à partir de ces informations.

Bien que ces travaux aient été initiés lors d'une collaboration avec Cdiscount pour la prévision de séries ventes pour l'e-commerce (voir le chapitre 4), les principaux résultats théoriques ont d'abord été rédigés et testés sur des données de consommation électrique.

Cadre de la prévision hiérarchique séquentielle

On dispose d'un ensemble de séries temporelles $\left\{(y_t^\gamma)_{t\geqslant 0}, \gamma \in \Gamma\right\}$ reliées entres elle par des relations de sommation : certaines d'entre elles sont la somme d'autres. L'ensemble Γ représente les indices correspondants de ces suites. On va reprendre ici un exemple de hiérarchie simple, l'exemple 1 de la partie 3.2.1, mais on pourra en trouver d'autres dans celle-ci. Dans cet exemple, on a seulement une relation de sommation : les valeurs totales

y^{Tot} sont les sommes des valeurs à N autres nœuds notées $y^1, \dots, y^N$. On représente cette hiérarchie en Figure 1.1 par un arbre contenant une unique racine directement connectée à N feuilles. Cette hiérarchie représente bien, par exemple, les données de consommation électrique pour la France (Tot) et ses $N = 13$ régions métropolitaines. Pour chaque pas de temps t, les séries temporelles vérifient $y_t^{\text{Tot}} = y_t^1 + y_t^2 + \cdots + y_t^N$. On définit donc que le vecteur $\mathbf{y}_t = \left(y_t^{\text{Tot}}, y_t^1, \dots, y_t^N\right)^{\text{T}}$ est compatible avec la hiérarchie si et seulement si $\underline{\mathbf{K}}\mathbf{y}_t = 0$ où $\underline{\mathbf{K}} = \left(-1, 1, 1, \dots, 1\right)$. De façon plus générale, on définira une hiérarchie par l'ensemble Γ, la collection de séries temporelles $\left\{(y_t^\gamma)_{t>0}, \gamma \in \Gamma\right\}$ que l'on représente en une suite de vecteurs $\mathbf{y}_t \overset{\text{def}}{=} (y_t^\gamma)_{\gamma\in\Gamma}$ et la matrice de contrainte $\underline{\mathbf{K}}$ telle que $\underline{\mathbf{K}}\mathbf{y}_t = 0$ pour tout t.

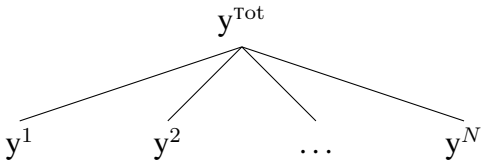


FIGURE 1.1 – Une hiérarchie à deux niveaux.

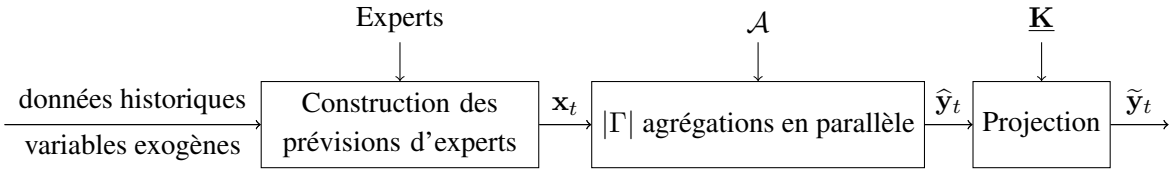
Ceci étant posé, on va maintenant détailler la démarche nous ayant conduit à notre solution (voir le paragraphe suivant) pour effectuer les prévisions de ces séries. Dans la lignée des travaux présentés en partie 1.2.1, nous nous sommes placés dans le cadre d'agrégation de prévisions d'experts. Il nous a fallu adapter la théorie de l'agrégation séquentielle pour l'étendre à des prévisions multivariées tout en conservant la possibilité d'avoir, pour chaque série à prévoir, des poids de mélange différenciés. En effet, comme cela a été dit en introduction de cette partie, les techniques d'agrégation traditionnelles permettent de faire des prévisions multivariées mais celles-ci résultent en des poids d'agrégation uniformes. On perd donc la capacité à sélectionner pour chaque nœud le meilleur expert. C'est pourquoi, on va utiliser non pas un vecteur de poids mais une matrice $\widetilde{\mathbf{U}}_t$ qui, par un produit matriciel avec le vecteur des prévisions d'experts $\mathbf{x}_t$, va former $\widetilde{\mathbf{y}}_t = \widetilde{\mathbf{U}}_t^{\text{T}}\mathbf{x}_t$, les prévisions de la stratégie $\mathcal{S}$. On s'assurera que celles-ci sont cohérentes, c'est-à-dire qu'elles respectent les contraintes encodées par la matrice $\underline{\mathbf{K}}$ (donc $\underline{\mathbf{K}}\widetilde{\mathbf{y}}_t = 0$). On définit le regret de la stratégie $\mathcal{S}$ par

$$\mathcal{R}_T(\mathcal{S}) \overset{\text{def}}{=} \sum_{t=1}^{T} \left\|\mathbf{y}_t - \widetilde{\mathbf{y}}_t\right\|^2 - \inf_{\underline{\mathbf{K}}\mathbf{U}^{\text{T}}=0} \sum_{t=1}^{T} \left\|\mathbf{y}_t - \mathbf{U}^{\text{T}}\mathbf{x}_t\right\|^2.$$

On peut noter plusieurs choses sur cette définition. Tout d'abord, nous avons choisi la perte quadratique pour évaluer les prévisions car cela s'est avéré nécessaire pour obtenir des garanties théoriques. Ensuite cette métrique est sommée pour tous les nœuds de la hiérarchie. On a essayé de varier les $\mathbf{x}_t$ par $\gamma \in \Gamma$, mais l'analyse théorique devenait difficile. Par ailleurs, si l'on dispose de plusieurs experts pour chaque nœud, il s'avère qu'on peut tous les rassembler en un unique vecteur $\mathbf{x}_t$ (par concaténation). Ainsi, on prend pour $\mathbf{x}_t$ les prévisions d'un ou plusieurs experts pour tous les nœuds de la hiérarchie. On trouvera une discussion sur les motivations théoriques de ce choix en partie 3.2.4. Finalement, on remarquera que l'on ne se compare pas à n'importe quelle matrice de mélange mais seulement à celles, dont on soit sûr, à peu de frais, qu'elles produisent toujours des prévisions cohérentes. En effet, $\underline{\mathbf{K}}\mathbf{U}^{\text{T}} = 0$ impose que, pour tout t et quelque soit $\mathbf{x}_t$, on ait $\underline{\mathbf{K}}\mathbf{U}^{\text{T}}\mathbf{x}_t = 0$. Nous allons maintenant détailler notre solution pour effectuer des prévisions admettant une borne sur ce regret.

Une prévision en 3 étapes

Notre solution procède en 3 étapes successives. On construit d'abord des prévisions d'experts que l'on agrège ensuite et qui sont finalement projetées. Les prévisions d'experts sont construites à partir de données historiques en utilisant des algorithmes d'apprentissage traditionnels (modèles auto-régressif, modèles additifs généralisés ou forêts aléatoires). Ces prévisions sont ensuite agrégées (linéairement) par un algorithme d'agrégation indépendamment pour chaque nœud γ. Finalement, on projette orthogonalement ces prévisions pour que celles-ci respectent les relations des sommations. Ces étapes sont représentées dans le diagramme suivant.



Cette méthode permet d'améliorer la prévision à un nœud en utilisant les prévisions d'experts des autres nœuds de façon robuste, ainsi que de produire des prévisions cohérentes hiérarchiquement. On démontre au Théorème 1.11 une borne sur le regret de cette procédure, en fonction de la borne de regret dont dispose l'algorithme d'agrégation $\mathcal{A}$ utilisé à chaque nœud. Il s'agit d'un résultat générique. Cet algorithme peut, par exemple, être EG avec pour borne celle du Théorème 1.2.

Notation 1.10 (voir la Notation 3.1)**.** *On suppose que, pour tout $\gamma \in \Gamma$, l'algorithme $\mathcal{A}^\gamma$, initialisé avec les paramètres $\mathbf{s}_0^\gamma$ assure que pour tout $T > 0$ et pour tout $\mathbf{u}^\gamma \in \mathbb{R}^{|\Gamma|}$, tout $\mathbf{x}_{1:T} = \mathbf{x}_1, \dots \mathbf{x}_T$ et tout $y_{1:T}^\gamma = y_1^\gamma, \dots, y_T^\gamma$,*

$$\mathcal{R}_T^\gamma(\mathcal{A}^\gamma, \mathbf{u}^\gamma) \stackrel{def}{=} \sum_{t=1}^T \left(y_t^\gamma - \widehat{y}_t^\gamma\right)^2 - \sum_{t=1}^T \left(y_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t\right)^2 \leqslant \mathrm{B}(\mathbf{x}_{1:T}, y_{1:T}^\gamma, \mathbf{s}_0^\gamma, \mathbf{u}^\gamma)\,. \tag{1.7}$$

En appliquant cette borne pour chacun des nœuds Γ, en les sommant, et en utilisant le théorème de Pythagore, on obtient le théorème suivant. On notera que la borne s'applique bien sur les prévisions finales après projection $\widetilde{\mathbf{y}}_t$ et non pas sur les prévisions en sortie des algorithmes d'agrégation $\widehat{\mathbf{y}}_t$.

Théorème 1.11 (voir le Théorème 3.2)**.** *Dans le cadre donné par la Notation 1.10, pour toute matrice $\mathbf{U}$ telle que $\underline{\mathbf{K}}\mathbf{U}^\intercal = 0$ et tout $T \geqslant 1$,*

$$\mathcal{R}_T(\mathbf{U}) \stackrel{def}{=} \sum_{t=1}^T \left\|\mathbf{y}_t - \widetilde{\mathbf{y}}_t\right\|^2 - \sum_{t=1}^T \left\|\mathbf{y}_t - \mathbf{U}^\intercal \mathbf{x}_t\right\|^2 \leqslant \sum_{\gamma \in \Gamma} \mathrm{B}\left(\mathbf{x}_{1:T}, y_{1:T}^\gamma, \mathbf{s}_0^\gamma, \mathbf{u}^\gamma\right).$$

Cette borne est très générale et peut en fait s'appliquer sans contrainte particulière sur la matrice $\underline{\mathbf{K}}$, tant que les relations entre les séries à prévoir sont linéaires.

Prévision de la consommation électrique de petits groupes de ménages

La prévision de la consommation électrique est intéressante pour les opérateurs nationaux car elle permet d'ajuster la production nécessaire et d'assurer la stabilité du réseau. De plus, la prévision à des niveaux moins agrégés est utile pour gérer la répartition de la charge sur le territoire. La prévision pour des petits groupes de consommateurs permet, quant à elle, de proposer une consommation plus efficace en incitant certains comportements induisant une consommation globale plus régulière. On s'est appuyé sur des données du « *Energy Demand Research Project*[5] », projet lancé en 2007 au Royaume-Uni, afin d'étudier la pertinence de notre méthode pour améliorer la qualité des prévisions. Après un prétraitement des données, il restait $1{,}545$ séries temporelles de consommation électrique d'une durée d'un an et demi. En plus d'informations présentes sur le profil des utilisateurs, nous avons ajouté les données météorologiques de la NOAA[6]. Nous avons testé la méthode présentée précédemment sur ces données (les trois derniers mois sont réservés pour l'évaluation, les précédents pour entraîner et calibrer les modèles), le but étant de voir si l'on pouvait diminuer l'erreur de prévision en utilisant la structure de celles-ci.

Dans nos expériences, on cherche à prévoir la consommation électrique de groupes de ménages 24 heures à l'avance. Pour cela, il faut réaliser 3 étapes : la construction de hiérarchies, suivit de l'entraînement et inférence de modèles experts (pour chacun des groupes de consommateurs) pour finalement en combiner les prévisions par agrégation puis projections conformément à la méthode précédemment décrite.

Étape 1. On construit une (ou plusieurs) hiérarchie(s) de consommateurs selon les critères suivants : les régions géographiques, les niveaux sociaux-économiques, les types de facturation, l'accès ou non au gaz de ville, etc. Une dernière hiérarchie s'appuie sur une segmentation automatique des ménages en fonction de leur profil de consommation. On trouvera le détail de ces méthodes de segmentation en partie 3.7.2. On a aussi considéré deux hiérarchies simultanément (de sorte que les consommateurs appartenaient à plusieurs groupes), cela constituant une hiérarchie croisée mais dont les relations de sommations étaient parfaitement représentables dans une matrice de contrainte $\underline{\mathbf{K}}$.

Étape 2. Ayant obtenu pour chaque regroupement une série temporelle de consommation électrique, on réalise l'apprentissage de modèles experts et ensuite la prévision sur de nouvelles données futures (les 3 derniers mois de validation). Nous avons testé trois familles de modèles (décrits en partie 3.5), les modèles autorégressifs, les modèles additifs généralisées ou les forêts aléatoires.

Étape 3. Finalement, nous les avons combinées, par la technique décrite à la partie précédente, c'est-à-dire une étage d'agrégation suivit d'une étape de projection. Nous avons comparé différents algorithmes d'agrégation afin de mesurer si les performances dépendaient fortement de la méthode choisie.

[5] www.ofgem.gov.uk/gas/retail-market/metering/transition-smart-meters/energy-demand-research-project
[6] National Oceanic and Atmospheric Administration, www.noaa.gov

Les résultats expérimentaux montrent l'apport en terme d'amélioration de la qualité des prévisions de notre méthode par rapport aux prévisions d'experts. On en présente un extrait dans cette introduction et on pourra trouver les autres en partie 3.7. On utilise ici une double hiérarchie croisée ; l'une basée sur les régions et l'autre basée sur la segmentation automatique en fonction du profil de consommation. Pour cette (double) hiérarchie, on construit des prévisions d'experts à l'aide de modèles additifs généralisés (GAM). Ensuite, on combine ces prévisions avec l'algorithme *non-linear ridge regression* (voir partie 1.1.5), noté « ridge-nl ». On mesure les performances à l'aide de la perte quadratique $\frac{1}{T}\sum_{t=1}^{T}\left\|\mathbf{y}_t - \widehat{\mathbf{y}}_t\right\|^2$ moyennée sur les 3 derniers mois, mis de côté pour la validation. On a reporté au Tableau 1.1 cette métrique pour différentes prévisions : celles de la procédure complète (« Agrégation + Projection »), celles qui suivent une agrégation seule (« Agrégation ») ou une étape de projection seule (« Projection »), et enfin, celles des experts, que l'on note « Benchmark ». On remarque que l'amélioration en terme d'erreur quadratique est principalement due à l'étape d'agrégation. La projection diminue seulement légèrement cette erreur mais elle a l'avantage de rendre les prévisions cohérentes.

	Erreur quadratique (kWh^2)
Benchmark (GAM)	455.5
Projection	450.7
Agrégation (ridge-nl)	407.6
Agrégation + Projection	405.9

TABLEAU 1.1 – L'erreur quadratique pour la hiérarchie $\frac{1}{T}\sum_{t=1}^{T}\left\|\mathbf{y}_t - \widehat{\mathbf{y}}_t\right\|^2$ suivant différentes méthodes de prévisions.

1.2.3 Prévision de demande pour l'e-commerce (chapitre 4)

La prévision de la demande est fondamentale pour tout commerce, qu'il soit traditionnel ou par internet. Elle permet de gérer les stocks et de trouver un équilibre entre coût de stockage et manque à gagner par déficit d'approvisionnement. Les séries de ventes dans l'e-commerce sont hautement dynamiques et irrégulières. Cela est dû à une plus forte activité commerciale (les ventes *flash*, par exemple) et la capacité à faire évoluer rapidement la présentation des produits (sur le site internet). Par ailleurs, comme les produits ont une durée de vie limitée, on ne peut facilement déceler les régularités sur le long terme. Pour lisser ces comportements, les logisticiens classent les produits dans des familles, sous-familles et sous-sous-familles. Les séries de vente à ces niveaux sont bien plus régulières et permettent aux statisticiens de déceler des tendances.

Prévision de demande : spécificités

On présente dans cette partie les principales spécificités du domaine de la prévision de demande, avec un accent sur les particularités du e-commerce. On étudiera d'abord la différence entre les ventes et la demande pour ensuite voir comment sont structurées ces données et finalement quelles modélisations sont traditionnellement utilisées dans ce secteur.

Nombre de ventes ou demande ? Le premier constat est que l'on ne peut directement observer la demande mais seulement son expression à travers les ventes. Lorsqu'il y a des stocks, les produits sont vendus en réponse à la demande et donc permettent de la mesurer. Au contraire, lorsqu'un produit vient à manquer, les ventes sont en deçà de la demande. Ceci peut avoir un effet pervers : les prévisions futures basées sur des données historiques sous-estimant la demande conduisent à une sous-estimation créant elle-même de nouvelles ruptures de stock. Il est donc important, pour les prévisions à l'échelle des produits, de traiter les observations en cas de rupture de stock comme des données manquantes. Par ailleurs, dans le secteur du e-commerce, les possibilités d'actions commerciales sont bien plus nombreuses que pour la vente au détail, et elles modifient significativement la demande. Il est donc également recommandé de traiter ces observations comme des données manquantes.

Regroupement hiérarchique. Pour gérer les problèmes liés à l'intermittence des données (nouveaux produits, abandon de commercialisation) ainsi que leur fort niveau de bruit (ruptures de stocks, actions commerciales), une solution est de les regrouper, en familles, sous-familles, sous-sous-familles, de façon à constituer une hiérarchie. Les données de ventes agrégées permettent de lisser les comportements imprévisibles liés aux ruptures de stock, aux promotions et autres. Les prévisions au niveau des produits sont alors réalisées se servant des prévisions de leurs regroupements (sous-sous-famille, sous-famille et famille d'appartenance), avec l'espoir que celles-ci ont pu capter une certaine régularité.

Modélisation traditionnelle. Les logisticiens, étant fortement orientés vers l'opérationnel, préfèrent des traitements simples plutôt que des modélisations statistiques complexes. C'est pourquoi ils utilisent généralement des modèles de lissage exponentiels (dont font partie les méthodes de Holt et de Holt-Winters) qui ont une interprétation simple, sont autorégressifs (n'utilisent pas de données auxiliaires) et ont un nombre très limité de paramètres. Théoriquement, dans l'e-commerce, au vu de la quantité de données auxiliaires accumulées, on devrait pouvoir améliorer les prévisions en les prenant en compte. Cela implique de disposer d'une infrastructure logiciel complexe et d'être capable d'extraire les bons signaux dans une masse de données bruitées. Le lecteur pourra trouver en fin de partie 4.1.5 l'indication de quelques développements récents allant dans ce sens.

Nous avons voulu étudier comment les méthodes d'agrégation séquentielle pouvaient être utiles pour les logisticiens. En particulier, compte tenu des données dont nous disposions (avec peu d'information sur les ruptures de stock et les opérations commerciales), nous avons décidé de nous focaliser sur les prévisions pour des données agrégées réparties dans une hiérarchie (et de ne pas considérer les prévisions au niveau des produits eux-mêmes). Ce problème est intéressant en soi mais ces prévisions peuvent aussi être utilisées comme information auxiliaire (*features*) pour prévoir les ventes au niveau des produits.

Choix de paramètres pour les méthodes de lissage par agrégation séquentielle

Un lissage exponentiel. Les méthodes de lissage exponentiel sont des méthodes autoregressives très simples. Par souci de concision, on n'en présentera qu'une mais on pourra se reporter à la partie 4.2.2 pour trouver davantage d'exemples. On note y_t la variable à prévoir

correspondant à la semaine $t \in \{1, 2, \dots\}$. Cela peut correspondre aux ventes de cette semaine t (ce que l'on va supposer ici pour simplifier), ou aux ventes de celle-ci ainsi que, par exemple, les 3 suivantes à venir (y_t serait alors les ventes pour les semaines $t, t+1, t+2$ et $t+3$). On se donne un horizon de prévision $h > 0$, cela veut dire qu'à la semaine t, ayant observé $y_1, \dots, y_t$, on cherche à prévoir la valeur de y_{t+h}. En réalité, on va, à la place, estimer la variation sur 52 semaines $d_{t+h} = y_{t+h} - y_{t+h-52}$, car cette quantité, éliminant la saisonnalité annuelle, est plus régulière. Il suffira d'ajouter y_{t+h-52}, à cette estimation pour obtenir la prévision. La quantité d_{t+h} est estimée à l'aide de $\widehat{d}_{t+h}$, lissage exponentiel simple de la série temporelle d_t :

$$\widehat{d}_{t_0+h} = d_{t_0} \qquad \text{et pour } t \geqslant t_0 + 1, \qquad \widehat{d}_{t+h} = \alpha\, d_t + (1-\alpha)\, \widehat{d}_{t-1+h}\,;$$

où $t_0 = 53$ est la première semaine pour laquelle on peut calculer d_{t_0} (et donc faire des prévisions) et $\alpha > 0$ est un paramètre permettant de contrôler l'intensité du lissage. La prévision finale est $\widehat{y}_{t+h} = y_{t+h-52} + \widehat{d}_{t+h}$. On voit que α est le seul paramètre du modèle. Traditionnellement, il est ajusté sur des données historiques en prenant celui qui minimise une métrique sur une période donnée.

Agrégation de modèles élémentaires sur une hiérarchie. L'agrégation consiste à considérer plusieurs modèles (plusieurs paramètres α) en parallèle et à faire une combinaison convexe de leurs prévisions (voir partie 1.1.4), ce qui est plus stable que d'en sélectionner un seul. De plus, on ne veut pas prendre le même paramètre α pour tous les nœuds de la hiérarchie. En utilisant un modèle d'agrégation par nœud, on s'assure d'optimiser pour celui-ci la combinaison convexe des différemment modèles élémentaires. On n'est pas obligé de se limiter à une seule classe de modèles comme celle présentée ici. Ceci permet de laisser l'algorithme d'agrégation choisir s'il est plus intéressant, par exemple, de traiter la saisonnalité de façon additive ou multiplicative. Les données étant organisées dans une hiérarchie, conformément à ce que qui a été présenté en partie 1.2.2, les algorithmes d'agrégation sont exécutés en parallèle sur chaque nœud et ensuite projetées. Cette dernière étape de projection, de façon surprenante, ne permet pas d'améliorer significativement les performances. Elle a néanmoins l'avantage de produire des prévisions réconciliées (cohérentes vis-à-vis de la hiérarchie)

Expérimentation avec les données de ventes de Cdiscount

Nous avons travaillé sur des données de ventes mises à disposition par la société Cdiscount afin d'évaluer ces méthodes. Après le pré-traitement des données (voir en partie 4.3.1 pour une description complète), nous avons obtenu $3{,}628$ séries temporelles de ventes, celle du total, celles de 53 familles, de 570 sous-familles et $3{,}004$ sous-sous-familles. Ces séries débutent en juillet 2014 et terminent en décembre 2017, couvrant un total de 182 semaines. L'évaluation des prévisions se fait sur la dernière année (complète), que l'on appelle l'ensemble de test. Les données de 2014 à 2016 constituent l'ensemble d'entraînement. Il est nécessaire d'avoir au moins un an de données pour les modèles prenant en compte la saisonnalité (car on utilise l'observation de l'année précédente) avant de pouvoir faire la moindre prévision, et il faut en général un an supplémentaire pour que les quantités lissées (comme $\widehat{d}_t$)

soient stabilisées. On construit des experts élémentaires par des techniques de lissage exponentiel (voir partie 4.2.2 pour la liste complète). Nous comparons ensuite différentes méthodes (dont l'agrégation) pour obtenir une prévision pour chacune des séries de la hiérarchie. L'évaluation des prévisions se fait principalement à l'aide de l'erreur absolue (notée MAE) car l'erreur quadratique (RMSE) est très sensible aux valeurs aberrantes, nombreuses dans les données de ventes du e-commerce. Pour cette métrique d'erreur absolue, l'étape de projection n'assurant pas une décroissance de l'erreur de prévision, on ne l'applique pas systématiquement. Pour rester concis, on ne présente ici qu'une partie des résultats numériques mais ceux-ci se trouvent dans leur intégralité en partie 4.3. On cherche à prévoir les ventes $h = 7$ semaines à l'avance et on évalue ces prévisions pour toute la hiérarchie à l'aide des métriques MAE, RMSE et MAPE moyennées sur l'année 2017. On trouvera au Tableau 1.2 une comparaison entre les performances de la méthode traditionnelle notée « Loc-Train. » (pour *locally best models on the train set*), consistant à sélectionner les meilleurs modèles élémentaires (pour chaque nœud) sur l'ensemble d'entraînement, et celles de notre méthode s'appuyant l'agrégation séquentielle, notée « Agrég. ».

Métrique en k€ ou %	Loc-Train	Agrég.	Agrég. vs. Loc-Train
MAE	8.39	7.97	-5.1%
RMSE	125.68	120.39	-4.2%
MAPE	22.15%	21.08%	-4.9%

TABLEAU 1.2 – Erreurs moyennes en k€ (colonnes 3 et 4) et différences relatives des erreurs (colonne 5) pour la méthode de sélection traditionnelle et l'agrégation séquentielle.

1.2.4 Compétition RTE

On décrit dans cette partie un travail produit (l'obtention du deuxième prix d'une compétition de *data-science*) au début de la thèse mais qui n'a pas donné lieu à des développements ultérieurs et n'est donc pas détaillé dans un chapitre à part entière.

Les compétitions de *data-science* se sont beaucoup popularisées au cours des dix dernières années. Elles permettent, en définissant une métrique d'évaluation, de comparer des méthodes de prévisions dans un même contexte ; à la différence des études plus théoriques qui sont généralement testées sur des données différentes. La compétition RTE qui s'est déroulée de mai à juillet 2017 avait pour objectif de prévoir la consommation électrique de la France et de ses régions métropolitaines. Une particularité de cette compétition était sa composante « temps réel » : il fallait prévoir la consommation pour dix journées, la veille à 21h. Ce processus correspond exactement au cadre opérationnel, et ces prévisions sont critiques pour assurer la stabilité du réseau électrique.

L'organisation de la compétition. La compétition était libre d'accès et hébergée sur la plateforme datascience.net et elle offrait 3 prix ($10{,}000$€ pour le premier prix, puis $5{,}000$€ pour le deuxième et $3{,}000$€ pour le troisième). Les dix journées à prévoir s'étalaient sur

une période de deux mois, et sur chacune d'elles, il fallait prévoir la consommation quadri-horaire pour la France et ses 12 régions métropolitaines (donc $13 \times 96 = 1248$ valeurs). Ces journées étaient principalement des jours fériés et des ponts, dont la prévision est la plus difficile. Il y a eu au total 479 participants mais face aux contraintes opérationnelles seulement une cinquantaine ont effectué des prévisions pour la totalité des journées.

Notre méthode. Le fonctionnement en temps réel de la compétition était parfaitement adapté au test des méthodologies d'agrégation. Pour cela, dans la continuité des travaux de Gaillard et al. [2016] qui ont déjà démontré leur bonnes performances lors de la compétition GEFCOM, nous avons développé des experts individuels par des techniques d'apprentissage machine classiques pour ensuite les mélanger à l'aide d'algorithmes d'agrégation séquentielle. La raison pour laquelle on n'utilise pas ces modèles directement sur les données brutes est que ceux-ci sont au mieux linéaires, en général convexes.

Construction des experts individuels. Les experts individuels ont été construits pour extraire des données à disposition des prévisions de la consommation. On disposait de 5 ans d'historique de consommation bi-horaire ainsi que les prévisions météorologiques pour 35 stations réparties dans les 12 régions. On y a ajouté des données calendaires indiquant le jour de la semaine, si celui-ci était férié etc... Les modèles experts utilisés se répartissent en 3 grandes familles, les modèles additifs généralisés (voir Wood, 2006), les réseaux de neurones (voir Chollet et al., 2015) et les modèles de boosting d'arbres (voir Chen and Guestrin, 2016). Comme ceux-ci ne sont pas naturellement adaptés au cadre des séries temporelles, on les entraîne et valide suivant la procédure décrite à la figure 1.2. Ces modèles disposant d'un grand nombre d'hyper-paramètres, nous avons construit une procédure automatique destinée à tester en continu de nouvelles combinaisons permettant ainsi de proposer une variété de modèles à agréger.

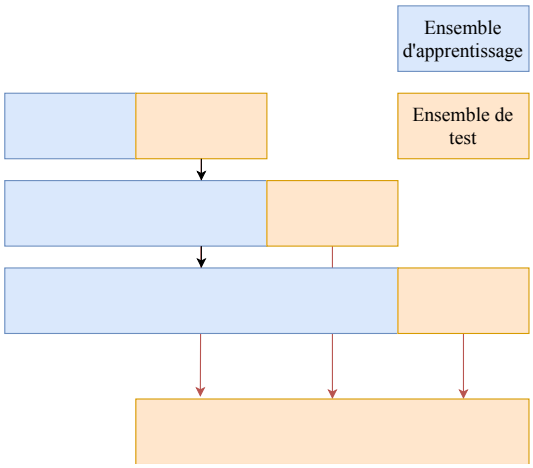


FIGURE 1.2 – Procédure pour entraîner un modèle classique sur des données séquentielles, la période d'apprentissage est le mois.

Agrégation séquentielle en temps réel. Le fait qu'il y ait 3 familles de modèles, les recherches d'hyper-paramètres, ainsi que la mise à jour des modèles chaque mois font que l'on a à disposition un grand nombre de modèles experts (240 dans la version finale) pour effectuer notre prévision. Les méthodes d'agrégation permettent de résoudre élégamment le problème de la sélection de modèle et ont, en pratique, un impact significatif sur les performances. En effet, on trouvera au Tableau 1.3 une comparaison entre l'agrégation, le meilleur modèle *a posteriori* et la moyenne uniforme des modèles. Cette démarche est aussi validée par l'obtention du deuxième prix de la compétition.

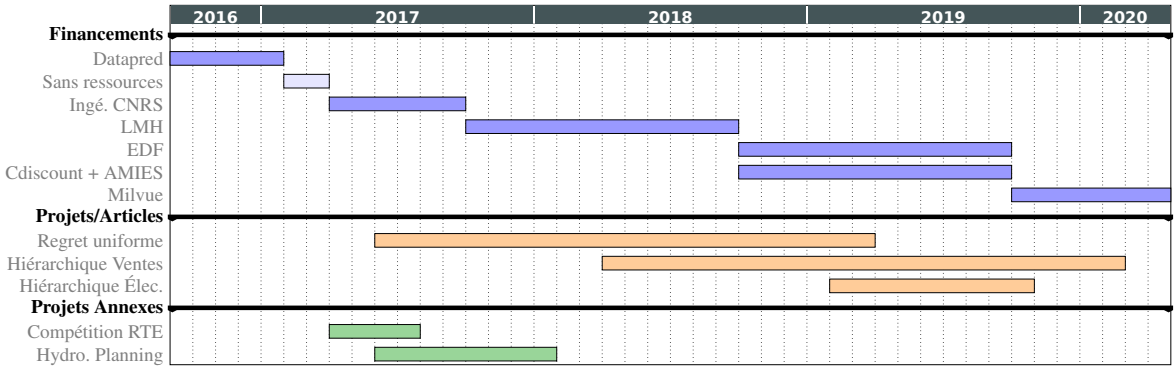
Méthode	MAPE
Meilleur expert	2.377
Ensemble uniforme	2.166
Agrégation	2.008

TABLEAU 1.3 – Pourcentage d'erreur absolue (MAPE)

Les apports des méthodes d'agrégation sont multiples. Elle permettent de résoudre le problème de la mise à jour des modèles non séquentiels. Il suffit de les ré-entraîner périodiquement et de les ajouter au modèle d'agrégation qui se charge d'effectuer de façon robuste la transition d'une version à l'autre. Comme pour les prévisions hiérarchiques, on fait beaucoup de prévisions simultanées (ici 1248), et il n'y a pas de raison qu'un même expert soit le plus performant pour tous les moments de la journée ou pour toutes les régions. Les méthodes d'agrégation permettent de sélectionner les meilleurs experts pour chacune de ces prévisions, et cela de façon robuste, ce qui est crucial car on ne disposait au début de la compétition que d'une soixantaine de jours comme données d'entraînement (du fait d'une différence entre les données historisées et les données de test). Finalement, comme les modèles ne sont entraînés qu'une fois par mois (pour des contraintes de temps de calcul), les méthodes d'agrégation permettent un ajustement de dernière minute sur les données en temps réel (la veille à 20h) pour, par exemple, corriger un biais dans les prévisions.

1.3 Déroulé et liste des livrables

On trouve ici un diagramme résumant le déroulé de la thèse ainsi que la liste des financements, articles, rapports techniques et communications orales en rapport avec celle-ci.



Refinancement

Cette thèse, qui a commencé comme une thèse CIFRE, a été refinancée à l'aide des fonds suivants : 6 mois en tant qu'ingénieur de recherche CNRS sur reliquats de contrat EDF-CNRS-HEC antérieur (contrat non lié à cette thèse), 12 mois sur une bourse LMH (LabEx Mathématiques Hadamard) et 12 mois via un contrat EDF-CNRS-PSud et un contrat Cdiscount-CNRS-PSud, plus abondement AMIES (agence mathématiques-entreprises) ; ces deux contrats étant, eux, liés aux travaux menés pendant cette thèse.

Articles issus de la thèse

Pierre Gaillard, Sébastien Gerchinovitz, Malo Huard et Gilles Stoltz. Uniform regret bounds over $\mathbb{R}^d$ for the sequential linear regression problem with the square loss. *Proceedings of the 30th Conference on Algorithmic Learning Theory (ALT 2019), in PMLR* 98:404–432, 2019.

Margaux Brégère et Malo Huard. Online hierarchical forecasting for power consumption data. arXiv preprint number 2003.00585, 2020.

Malo Huard, Gilles Stoltz et Rémy Garnier. Hierarchical robust aggregation of demand forecasts in e-commerce. arXiv preprint number 2006.03373, 2020

Rapports techniques

Sandra Claudel et Malo Huard. Hydropower scheduling: Learning handmade corrections. Rapport technique, 2017

Malo Huard. Prévision séquentielle déterministe par agrégation pour la compétition RTE. Rapport technique, 2017.

Communication orale dans un congrès international

Uniform regret bounds over $\mathbb{R}^d$ for the sequential linear regression problem with the square loss. 30th Conference on Algorithmic Learning Theory (ALT 2019), Chicago, 22 - 24 mars 2019.

Communications orales en France

Prévision séquentielle déterministe par agrégation pour la compétition RTE. Séminaire RTE, juillet 2017. Séminaire EDF, septembre 2017. Séminaire CEA, décembre 2017. Big Data Day @HEC, janvier 2018.

Hydropower production scheduling: learning handmade corrections. DataSciEn'2018 - Learning from Scientific Data in Energy, IFP Énergies nouvelles, Rueil-Malmaison, 17 janvier 2018

Prévision séquentielle déterministe par agrégation pour Cdiscount. 50èmes Journées de Statistique (JdS 2018), EDF Lab Paris-Saclay, Palaiseau, 28 mai - 1 juin 2018.

CHAPTER 2

Uniform regret bounds over $\mathbb{R}^d$ for sequential linear regression

In this chapter we consider the setting of online linear regression for arbitrary deterministic sequences, with the square loss. We are interested in the aim set by Bartlett et al. [2015]: obtain regret bounds that hold uniformly over all competitor vectors. When the feature sequence is known at the beginning of the game, they provided closed-form regret bounds of $2dB^2 \ln T + \mathcal{O}_T(1)$, where T is the number of rounds and B is a bound on the observations. Instead, we derive bounds with an optimal constant of 1 in front of the $dB^2 \ln T$ term. In the case of sequentially revealed features, we also derive an asymptotic regret bound of $dB^2 \ln T$ for any individual sequence of features and bounded observations. All our algorithms are variants of the online non-linear ridge regression forecaster, either with a data-dependent regularization or with almost no regularization.

Contents

This chapter is joint work with Pierre Gaillard, Sébastien Gerchinovitz and Gilles Stoltz, it has been published as a conference paper Gaillard et al. [2019] presented at ALT 2019.

2.1 Introduction and setting

We consider the setting of online linear regression for arbitrary deterministic sequences with the square loss, which unfolds as follows. First, the environment chooses a sequence of observations $(y_t)_{t\geqslant 1}$ in $\mathbb{R}$ and a sequence of feature vectors $(\mathbf{x}_t)_{t\geqslant 1}$ in $\mathbb{R}^d$. The observation sequence $(y_t)_{t\geqslant 1}$ is initially hidden to the learner, while the sequence of feature vectors (see Bartlett et al., 2015) may be given in advance or be initially hidden as well, depending on the setting considered: "beforehand-known features" (also called the fixed-design setting) or "sequentially revealed features". At each forecasting instance $t \geqslant 1$, Nature reveals $\mathbf{x}_t$ (if it was not initially given), then the learner forms a prediction $\widehat{y}_t \in \mathbb{R}$. The observation $y_t \in \mathbb{R}$ is then revealed and instance $t+1$ starts. In all results of this paper, the observations y_t will be assumed to be bounded in $[-B, B]$ (but the forecaster will have no knowledge of B), while we will avoid as much as possible boundedness assumptions of the features $\mathbf{x}_t$. See Figure 2.1.

Sequentially revealed features	Beforehand-known features
Given: [No input]	**Given:** $\mathbf{x}_1, \dots, \mathbf{x}_T \in \mathbb{R}^d$
For $t = 1, 2, \dots, T$, the learner:	**For** $t = 1, 2, \dots, T$, the learner:
• observes $\mathbf{x}_t \in \mathbb{R}^d$	
• predicts $\widehat{\mathbf{y}}_t \in \mathbb{R}$	• predicts $\widehat{\mathbf{y}}_t \in \mathbb{R}$
• observes $y_t \in [-B, B]$	• observes $y_t \in [-B, B]$
• incurs $(\widehat{\mathbf{y}}_t - y_t)^2 \in \mathbb{R}$	• incurs $(\widehat{\mathbf{y}}_t - y_t)^2 \in \mathbb{R}$

Figure 2.1 – The two online linear regression settings considered, introduced by Bartlett et al. [2015]; the learner has no knowledge neither of B nor (in the left case) of T.

The goal of the learner is to perform on the long run (when T is large enough) almost as well as the best fixed linear predictor in hindsight. To do so, the learner minimizes her cumulative regret,

$$\mathcal{R}_T(\mathbf{u}) = \sum_{t=1}^{T} (y_t - \widehat{\mathbf{y}}_t)^2 - \sum_{t=1}^{T} (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 \,,$$

either with respect to specific vectors $\mathbf{u} \in \mathbb{R}^d$ (e.g., in a compact subset) or uniformly over $\mathbb{R}^d$. In this chapter, and following Bartlett et al. [2015], we will be interested in

$$\sup_{\mathbf{u}\in\mathbb{R}^d} \mathcal{R}_T(\mathbf{u}) = \sum_{t=1}^{T} (y_t - \widehat{\mathbf{y}}_t)^2 - \inf_{\mathbf{u}\in\mathbb{R}^d} \sum_{t=1}^{T} (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 \,, \tag{2.1}$$

which we will refer to as the uniform regret over $\mathbb{R}^d$ (or simply, the uniform regret). The worst-case uniform regret corresponds to the largest uniform regret of a strategy, when considering all possible sequences of features $\mathbf{x}_t$ and (bounded) observations y_t; we will also refer to it as a twice uniform regret, see Section 2.4.1.

Notation. Bounded sequences of real numbers $(a_t)_{t\geqslant 1}$, possibly depending on external quantities like the feature vectors $\mathbf{x}_1, \mathbf{x}_2, \dots$, are denoted by $a_T = \mathcal{O}_T(1)$. For a given positive function f, the piece of notation $a_T = \mathcal{O}_T\Big(f(T)\Big)$ then indicates that we have $a_T/f(T) = \mathcal{O}_T(1)$. Also, the notation $a_T = \Theta_T(1)$ is a short-hand for the facts that

$a_T = \mathcal{O}_T(1)$ and $1/a_T = \mathcal{O}_T(1)$, i.e., for the fact that $(a_t)_{t\geqslant 1}$ is bounded from above and from below. We define a similar extension $a_T = \Theta_T\Big(f(T)\Big)$ meaning that $a_T/f(T) = \Theta_T(1)$.

Earlier works. Linear regression with batch stochastic data has been extensively studied by the statistics community. Our setting of online linear regression for arbitrary sequences is of more recent interest; it dates back to Foster [1991], who considered binary labels $y_t \in \{0,1\}$ and vectors $\mathbf{u}$ with bounded ℓ_1–norm. We refer the interested reader to the monograph by Cesa-Bianchi and Lugosi [2006, Chapter 11] for a thorough introduction to this literature and to Bartlett et al. [2015] for an overview of the state of the art. Here, we will mostly highlight some key contributions. One is by Vovk [2001] and Azoury and Warmuth [2001]: they designed the non-linear ridge regression recalled in Section 2.2.1, which achieves a regret of order $d\ln T$ only uniformly over vectors $\mathbf{u}$ with bounded ℓ_2–norm. Vovk [2001] also provided a matching minimax lower bound $dB^2\ln T - \mathcal{O}_T(1)$ on the worst-case uniform regret over $\mathbb{R}^d$ of any forecaster, where B is a bound on the observations $|y_t|$ (see also the lower bound provided by Takimoto and Warmuth, 2000). More recently, Bartlett et al. [2015] computed the minimax regret for the problem with beforehand-known features and provided an algorithm that is optimal under some (stringent) conditions on the sequences $(\mathbf{x}_t)_{t\geqslant 1}$ and $(y_t)_{t\geqslant 1}$ of features and observations. The best closed-form (but general: for all sequences) uniform regret they could obtain for this algorithm was of order $2dB^2\ln T$. This algorithm is scale invariant with respect to the sequence of features $(\mathbf{x}_t)_{t\geqslant 1}$. Their analysis emphasizes the importance of a data-dependent metric to regularize the algorithm, which is harder to construct when the features are only revealed sequentially. To that end, Malek and Bartlett [2018] show that, under quite intricate constraints on the features and observations, the backward algorithm of Bartlett et al. [2015] can also be computed in a forward (and thus legitimate) fashion in the case when the features are only revealed sequentially. It is thus also optimal; see, e.g., Lemma 39 and Theorem 46 therein.

Organization of the paper and contributions. We first recall and discuss the regret bound of the non-linear ridge regression algorithm (Section 2.2.1), whose proof will be a building block for our new analyses; we will show that perhaps surprisingly it enjoys a uniform regret bound $2dB^2\ln T + \mathcal{O}_T(1)$. For the sake of completeness, we also state and re-prove (Section 2.2.2) the regret lower bound by Vovk [2001] (and Takimoto and Warmuth, 2000), as most of the discussions in this paper will be about the optimal constant in front of the $dB^2\ln T$ regret bound; this optimal constant will be seen to equal 1. Our proof resorts to a general argument, namely, the van Trees inequality (see Gill and Levit, 1995), to lower bound the error made by any forecaster, while Vovk [2001] was heavily relying on the fact that in a Bayesian stochastic context, the optimal strategy can be determined. This new tool for the machine learning community could be of general interest to derive lower bounds in other settings. We also believe that our lower bound proof is enlightening for statisticians. It shows that the expectation of the regret is larger than a sum of quadratic estimation errors for a d–dimensional parameter. Each of these errors corresponds to an estimation based on a sample of respective length $t-1$, thus is larger than something of the order of d/t, which is the optimal parametric estimation rate. Hence the final $d(1+1/2+\ldots+1/T) \sim d\ln T$ regret lower bound.

We next show (Section 2.3) that in the case of beforehand-known features, the

non-linear ridge regression algorithm and its analysis may make good use of a proper metric $\|\cdot\|_{\mathbf{G}_T}$ described in (2.8) instead of the Euclidean norm. This leads to a worst-case bound of $dB^2\ln(1+T/d)+dB^2$ on the uniform regret over $\mathbb{R}^d$, which is optimal (with an optimal constant of 1) in view of the perfectly matching minimax lower bound of Section 2.2.2. To the best of our knowledge, earlier closed-form worst-case upper bounds were suboptimal by a factor of 2. See the corresponding discussions for the non-linear ridge regression algorithm, in Section 2.2.1, and for the minimax forecaster by Bartlett et al. [2015], in Remark 3 of Section 2.3.

The question then is (Section 2.4) whether a $dB^2\ln T+\mathcal{O}_T(1)$ regret bound can be achieved on the uniform regret in the most interesting setting of sequentially revealed features. Surprisingly enough, even if the traditional bound for the non-linear ridge regression forecaster blows up when the regularization parameter vanishes, $\lambda=0$ (see Section 2.2.1), an ad hoc analysis can be made in this case; it yields a uniform regret bound of $dB^2\ln T+\mathcal{O}_T(1)$. This bound holds for any fixed sequence of features and bounded observations; we do not impose stringent conditions as in Malek and Bartlett [2018]. Also, no parameter needs to be tuned, which is a true relief. The only drawback of this bound, compared to the bounds obtained in the case of beforehand-known features, is that the $\mathcal{O}_T(1)$ remainder term depends on the sequence of features. We thus could not derive a worst-case uniform regret bound.

Therefore, a final open question is stated in Section 2.4.1 and consists in determining if such a twice uniform regret bound of order $dB^2\ln T$ (over comparison vectors $\mathbf{u}\in\mathbb{R}^d$ and over feature vectors $\mathbf{x}_t\in\mathbb{R}^d$ and bounded observations y_t) may hold in the case of sequentially revealed features, or whether the lower bound should be improved. The proofs of the lower bound (the ones by Vovk, 2001, Takimoto and Warmuth, 2000, and our one) generate observations and feature vectors ex ante, independently of the strategy considered, and reveal them to the latter before the prediction game starts. However, it might be the case that truly sequential choices to annoy the strategy considered or generating feature sequences with difficult-to-predict sequences of Gram matrices lead to a larger regret being suffered.

2.2 Sequentially revealed features / Partially known results

In this section, we recall and reestablish some known results regarding the regret with the square loss function. We recall the definition and the regret bound (Section 2.2.1) of the non-linear ridge regression algorithm of Vovk [2001], Azoury and Warmuth [2001]. The (proof of this) regret bound is used later in this chapter to design and study our new strategies. We reestablish as well a

$$dB^2\Big(\ln T-(3+\ln d)-\ln\ln T\Big)$$

lower bound on the regret of any forecaster (Section 2.2.2), which implies that the worst-case uniform regret bound $dB^2\ln(1+T/d)+dB^2$ obtained in Section 2.3 is first-order optimal: it gets the optimal $dB^2\ln T$ main term.

2.2.1 Upper bound on the regret / Reminder of a known result + a new consequence of it

The *non-linear ridge regression algorithm* of Vovk, Azoury and Warmuth uses at each time-step t a vector $\widehat{\mathbf{u}}_t$ such that $\widehat{\mathbf{u}}_1 = (0, \dots, 0)^{\mathrm{T}}$ and for $t \geqslant 2$,

$$\widehat{\mathbf{u}}_t \in \operatorname*{argmin}_{\mathbf{u}\in\mathbb{R}^d} \left\{ \sum_{s=1}^{t-1} (y_s - \mathbf{u}\cdot\mathbf{x}_s)^2 + (\mathbf{u}\cdot\mathbf{x}_t)^2 + \lambda \|\mathbf{u}\|^2 \right\}, \tag{2.2}$$

where $\|\cdot\|$ denotes the Euclidean norm, and predicts $\widehat{\mathbf{y}}_t = \widehat{\mathbf{u}}_t \cdot \mathbf{x}_t$. No clipping can take place to form the prediction as the learner has no knowledge of the range $[-B, B]$ of the observations.

Note that the definition (2.2) is not scale invariant. By scale invariance, we mean that if the $\mathbf{x}_t$ are all multiplied by some $\gamma > 0$ (or even by an invertible matrix Γ), the vector $\widehat{\mathbf{u}}_t$ used should also be just divided by γ (or multiplied by Γ^{-1}). We may also define what a scale-invariant bound on the uniform regret is: a bound that is unaffected by a rescaling of the feature vectors $\mathbf{x}_t$ (as the vectors $\mathbf{u} \in \mathbb{R}^d$ compensate for the rescaling).

Notation 2.1. *Given features $\mathbf{x}_1, \mathbf{x}_2, \dots \in \mathbb{R}^d$, we denote by $\mathbf{G}_t = \sum_{s=1}^t \mathbf{x}_s \mathbf{x}_s^{\mathrm{T}}$ the associated $d \times d$ Gram matrix at step $t \geqslant 1$. This matrix is symmetric and positive semidefinite; it admits d eigenvalues, which we sort in non-increasing order and refer to as $\lambda_1(\mathbf{G}_t), \dots, \lambda_d(\mathbf{G}_t)$. Furthermore, we denote by $r_t = \operatorname{rank}(\mathbf{G}_t)$ the rank of $\mathbf{G}_t$. In particular, $\lambda_{r_t}(\mathbf{G}_t)$ is the smallest positive eigenvalue of G_t.*

For $\lambda > 0$, we have a unique, closed-form solution of (2.2): denoting $\mathbf{A}_t = \lambda \mathbf{I}_d + \mathbf{G}_t$, which is a symmetric definite positive thus invertible matrix, and $\mathbf{b}_{t-1} = \sum_{s=1}^{t-1} y_t \mathbf{x}_t$,

$$\widehat{\mathbf{u}}_t = \mathbf{A}_t^{-1} \mathbf{b}_{t-1} \,. \tag{2.3}$$

We recall the proof of the following theorem in Section 2.6, mostly for the sake of completeness and because we will use some standard inequalities extracted from it.

Theorem 2.2 (see Theorem **11.8** of Cesa-Bianchi and Lugosi, 2006)**.** *Let the non-linear ridge regression* (2.2) *algorithm be run with parameter $\lambda > 0$. For all $T \geqslant 1$, for all sequences $\mathbf{x}_1, \dots, \mathbf{x}_T \in \mathbb{R}^d$ and all $y_1, \dots, y_T \in [-B, B]$, for all $\mathbf{u} \in \mathbb{R}^d$,*

$$\mathcal{R}_T(\mathbf{u}) \leqslant \lambda \|\mathbf{u}\|^2 + B^2 \sum_{k=1}^{d} \ln\left(1 + \frac{\lambda_k(\mathbf{G}_T)}{\lambda}\right).$$

The regret bound above involves a $\lambda \|\mathbf{u}\|^2$ term, which blows up when the supremum over $\mathbf{u} \in \mathbb{R}^d$ is taken. However, under an additional boundedness assumption on the features $\mathbf{x}_t$, we could prove the following uniform regret bound. To the best of our knowledge, this is the first uniform regret bound proved for this well-known forecaster. Other uniform regret bounds (see Bartlett et al., 2015) were proved for ad-hoc and more involved forecasters, not for a standard, good old forecaster like the non-linear ridge regression (2.2).

However, despite our best efforts, the uniform regret bound we could prove is only of the form $2dB^2 \ln(T) + \mathcal{O}_T(1)$. It has two drawbacks: first, as we show in the next sections, the constant 2 in the leading term is suboptimal; second, the $\mathcal{O}_T(1)$ strongly depends on the

sequence of feature vectors. The proof is provided in Section 2.6 and essentially consists in noting that it is unnecessary to worry about vectors $\mathbf{u} \in \mathbb{R}^d$ with too large a norm, as they never achieve the infimum in (2.1).

Corollary 2.3. *Let the non-linear ridge regression* (2.2) *be run with parameter $\lambda > 0$. For all $T \geqslant 1$, for all sequences $\mathbf{x}_1, \dots, \mathbf{x}_T \in \mathbb{R}^d$ with $\|\mathbf{x}_t\| \leqslant X$ and all $y_1, \dots, y_T \in [-B, B]$,*

$$\sup_{\mathbf{u} \in \mathbb{R}^d} \mathcal{R}_T(\mathbf{u}) \leqslant r_T B^2 \ln\left(1 + \frac{TX^2}{r_T \lambda}\right) + \frac{\lambda}{\lambda_{r_T}(\mathbf{G}_T)} TB^2 \,.$$

Proper choices for λ to minimize the upper bound above are roughly of the order of $1/T$, to get rid of the linear part of the bound given by TB^2; because of the T/λ term in the logarithm, the resulting bound has unfortunately a main term of order $dB^2 \ln T^2 = 2dB^2 \ln T$. For instance, the choice $\lambda = 1/T$, that does not require any beforehand knowledge of the features $\mathbf{x}_t$, together with the bound $r_T \leqslant d$ and the fact that $u \mapsto (1/u)\ln(1+u)$ is decreasing over $(0, +\infty)$, leads to a regret bound less than

$$2dB^2 \ln T + \frac{B^2}{\lambda_{r_T}(\mathbf{G}_T)} + dB^2 \ln\Bigl(1 + X^2/d\Bigr) \,.$$

The $B^2/\lambda_{r_T}(\mathbf{G}_T)$ quantity in the regret bound is not uniformly bounded over sequences of features $\mathbf{x}_t$. In this respect, Corollary 2.3 only constitues a minor improvement on Theorem 2.2. We note a scaling issue: for a fixed sequence of observations $y_1, y_2, \dots$, while the uniform regret is not affected by a scaling of the feature vectors, the upper bound exhibited above is so. The deep reason for this issue is the lack of invariance of the non-linear ridge regression (2.2) itself.

2.2.2 Lower bound on the uniform regret / Improvement on known results

In this section, we study the uniform regret $\sup\Bigl\{\mathcal{R}_T(\mathbf{u}) : \mathbf{u} \in \mathbb{R}^d\Bigr\}$, in the minimax case (of beforehand-known features, which is the most difficult setting for a lower bound). That is, we are interested in

$$\mathcal{R}^\star_{T,\,[-B,B]} \stackrel{\text{def}}{=} \inf_{\text{forecasters}} \sup_{\mathbf{x}_1,\dots,\mathbf{x}_T \in \times [0,1]^d} \sup_{y_t \in [-B,B]} \left\{ \sum_{t=1}^T (y_t - \widehat{y}_t)^2 - \inf_{\mathbf{u} \in \mathbb{R}^d} \sum_{t=1}^T (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 \right\}, \tag{2.4}$$

where the first infimum is over all forecasters (all forecasting strategies) that can possibly access beforehand all the features $\mathbf{x}_1, \dots, \mathbf{x}_T$ that are considered next, and the second supremum is over all individual sequences $y_1, \dots, y_T \in [-B, B]$, that are sequentially revealed. Our result is the following; we carefully explain in Remark 2 why this result slightly improves on the existing literature.

Theorem 2.4. *For all $T \geqslant 8$ and $B > 0$, we have*

$$\mathcal{R}^\star_{T,\,[-B,B]} \geqslant dB^2 \Bigl(\ln T - (3 + \ln d) - \ln\ln T\Bigr) \,.$$

Remark 1. Note that the features $\mathbf{x}_t$ could be any element of $\mathbb{R}^d$ (by a scaling property on the $\mathbf{u}$), they do not necessarily need to be restricted to $[0,1]^d$; it is merely that our proof relies on such $[0,1]^d$–valued features. Compare to Theorem 2.5, where no boundedness assumption is required on the features.

Remark 2. Our proof reuses several ideas from the original proof of Vovk [2001, Theorem 2], namely, taking features $\mathbf{x}_t$ with only one non-zero input equal to 1 and Bernoulli observations y_t, resorting to a randomization with a Beta prior distribution, etc.; see also the proof of Takimoto and Warmuth [2000, Theorem 4]. However, we believe that we achieve a more satisfactory result than the $(d-\varepsilon)B^2 \ln T - C_\varepsilon$ lower bound of Vovk [2001, Theorem 2], where $\varepsilon > 0$ is a parameter and C_ε is a finite value; also, the proof technique somewhat relied on the boundedness of the features to derive the general case $d \geqslant 2$ from the special case $d = 1$. See also similar results for the case $d = 1$ in Takimoto and Warmuth [2000, Theorem 4], with the same issue for the generalization to $d \geqslant 2$. Our proof, on the contrary, directly tackles the d–dimensional case, which turns out to be more efficient (and more elegant). However, our alternative proof for the lower bound is admittedly a minor variation of existing results, it merely sheds a slightly different light on the bound, see the interpretation below in terms of parametric estimation rate.

The high-level idea of our proof of this known bound is to see the desired $d \ln T$ bound as a sum of parametric estimation errors in $\mathbb{R}^d$, each of order at least d/t. It is a classic result in parametric statistics that the estimation of a d–dimensional parameter based on a sample of size t can be performed at best at rate d/t in quadratic error, and this is exactly what is used in our proof. Vovk [2001, Theorem 2] was heavily relying on the fact that in a Bayesian stochastic context, the optimal strategy can be determined: his proof states that "since Nature's strategy is known, it is easy to find the best, on the average, strategy for Statistician (the Bayesian strategy)." In contrast, our argument does not require to explicitly compute the optimal strategy. It relies on the van Trees inequality (see Gill and Levit, 1995), that lower bounds the estimator error of any, possibly biased, forecaster—unlike the Cramér-Rao bound, which only holds for unbiased estimators. In this respect, the van Trees inequality could reveal itself a new tool of general interest for the machine learning community to derive lower bounds in other settings.

Proof.(sketch) The complete proof can be bound in Section 2.5 and we merely indicate here its most salient arguments. We start with a case where $y_t \in [0,1]$ and explain later how to draw the result for the desired case where $y_t \in [-B, B]$.

We fix any forecaster. A sequence $J_1, \ldots, J_T$ is drawn independently and uniformly at random over $\{1, \ldots, d\}$ and we associate with it the sequence of feature vectors $\mathbf{e}_{J_1}, \ldots, \mathbf{e}_{J_T}$, where $\mathbf{e}_j$ denotes the unit vector $(0, \ldots, 0, 1, 0, \ldots, 0)^{\mathrm{T}}$ along the j-th coordinate (the 1 is in position j). The forecaster is informed of this sequence of feature vectors and the sequential prediction problem starts. We actually consider several prediction problems, each indexed by $\theta^\star \in [0,1]^d$: conditionally on the feature vectors $\mathbf{e}_{J_1}, \ldots, \mathbf{e}_{J_T}$, at each round t the observation Y_t is drawn independently according to a Bernoulli distribution with parameter $\theta^\star \cdot \mathbf{e}_{J_t} = \theta^\star_{J_t}$. Expectations with respect to the randomization thus defined will be denoted by $\mathbb{E}_{\theta^\star}$.

Now, given the features considered above, that are unit vectors, each forecasting strategy can be termed as picking only linear combinations $\widehat{\mathbf{y}}_t = \widehat{\mathbf{u}}_t \cdot \mathbf{e}_{J_t}$ as predictions. Indeed, we

denote by $\widehat{\mathbf{y}}_t(j)$ the prediction output by the strategy when $J_t = j$ given the past observations $Y_1, \dots, Y_s$ and the features $J_1, \dots, J_T$. We then consider the vector $\widehat{\mathbf{u}}_t \in \mathbb{R}^d$ whose j–th component equals $\widehat{u}_{j,t} = \widehat{\mathbf{y}}_t(j)$. This way, in our specific stochastic setting, outputting direct predictions $\widehat{\mathbf{y}}_t$ of the observations or outputting vectors $\widehat{\mathbf{u}}_t \in \mathbb{R}^d$ to form linear combinations are the same thing.

The sketchy part of the proof starts here (again, details can be found in Section 2.5). By exchanging an expectation and an infimum and by repeated uses of the tower rule, we have, for each $\theta^\star \in [0,1]^d$:

$$\mathbb{E}_{\theta^\star}\left[\sup_{\mathbf{u}\in\mathbb{R}^d} \mathcal{R}_T(\mathbf{u})\right] \geqslant \sum_{t=1}^T \mathbb{E}_{\theta^\star}\left[(Y_t - \widehat{\mathbf{y}}_t)^2\right] - \inf_{\mathbf{u}\in\mathbb{R}^d} \sum_{t=1}^T \mathbb{E}_{\theta^\star}\left[(Y_t - \mathbf{u}\cdot\mathbf{e}_{J_t})^2\right] \geqslant \sum_{t=1}^T \frac{1}{d}\,\mathbb{E}_{\theta^\star}\left[\left\|\widehat{\mathbf{u}}_t - \theta^\star\right\|_2^2\right]. \tag{2.5}$$

The inequality above being valid for all forecasters accessing in advance to the entire sequence of feature vectors, we thus proved, for any prior π over $[0,1]^d$,

$$\mathcal{R}^\star_{T,\,[0,1]} \geqslant \inf_{\text{forecasters}} \sum_{t=1}^T \int_{[0,1]^d} \frac{1}{d}\,\mathbb{E}_{\theta^\star}\left[\left\|\widehat{\mathbf{u}}_t - \theta^\star\right\|_2^2\right] \mathrm{d}\pi(\theta^\star)\,.$$

An immediate application of the (multi-dimentional) van Trees inequality with a Beta(α,α) prior π shows that for all forecasters, all $t \geqslant 1$ and $\alpha \geqslant 3$,

$$\int_{[0,1]^d} \frac{1}{d}\,\mathbb{E}_{\theta^\star}\left[\left\|\widehat{\mathbf{u}}_t - \theta^\star\right\|_2^2\right] \mathrm{d}\pi(\theta^\star) \geqslant \frac{d}{4t + 2t/(\alpha-1) + 16d\alpha}\,,$$

which is roughly of order $d/(4t)$, as we will take large values of α (of order $\ln T$). Straightforward calculations conclude the proof and lead to a lower bound on $\mathcal{R}^\star_{T,\,[0,1]}$ of order $(d/4)\ln T$, which entails a lower bound of order $dB^2 \ln T$ on $\mathcal{R}^\star_{T,\,[-B,B]}$. □

2.3 Beforehand-known features / New result

In this section we assume that the features are known beforehand and exhibit a simple forecaster with a closed-form regret bound of $dB^2 \ln T + \mathcal{O}_T(1)$ uniformly over $\mathbb{R}^d$ and all sequences of features and of bounded observations. Combined with the minimax lower bound of Theorem 2.4, this upper bound implies that the minimax regret for beforehand-known features has a leading term exactly equal to $dB^2 \ln T$. It thus closes a gap between $dB^2 \ln T$ and $2dB^2 \ln T$ left open by earlier closed-form results such as those of Bartlett et al. [2015, Theorem 8]. See a more detailed discussion below, in Remark 3.

The *non-linear ridge regression algorithm with adapted regularization* will pick weight vectors as follows: $\widehat{\mathbf{u}}_1 = (0,\dots,0)^{\mathrm{T}}$ and for $t \geqslant 2$,

$$\widehat{\mathbf{u}}_t \in \operatorname*{argmin}_{\mathbf{u}\in\mathbb{R}^d} \left\{ \sum_{s=1}^{t-1} (y_s - \mathbf{u}\cdot\mathbf{x}_s)^2 + (\mathbf{u}\cdot\mathbf{x}_t)^2 + \lambda \sum_{s=1}^{T} (\mathbf{u}\cdot\mathbf{x}_s)^2 \right\} \tag{2.6}$$

with the constraint that $\widehat{\mathbf{u}}_t$ should be of minimal norm within all vectors of the stated argmin.

It then predicts $\widehat{\mathbf{y}}_t = \widehat{\mathbf{u}}_t \cdot \mathbf{x}_t$. As shown in Section 2.7.2, the closed-form expression for $\widehat{\mathbf{u}}_t$ reads

$$\widehat{\mathbf{u}}_t = \Big(\lambda\, \mathbf{G}_T + \mathbf{G}_t\Big)^{\dagger} \mathbf{b}_{t-1}\,, \tag{2.7}$$

where $\dagger$ denotes the Moore-Penrose inverse of a matrix (see Appendix 2.A.3) and where $\mathbf{b}_{t-1}$ was defined in (2.3).

The difference to (2.2) lies in the regularization term, which can be denoted by

$$\lambda \left\|\mathbf{u}\right\|_{\mathbf{G}_T}^2 \stackrel{\text{def}}{=} \lambda\, \mathbf{u}^{\mathrm{T}} \mathbf{G}_T \mathbf{u} = \lambda \sum_{s=1}^{T} (\mathbf{u} \cdot \mathbf{x}_s)^2\,; \tag{2.8}$$

that is, this regularization term can be seen as a metric adapted to the known-in-advance features $\mathbf{x}_1, \dots, \mathbf{x}_T$. This algorithm has the desirable property of being scale invariant. Actually, as will be clear from the equality (2.12) in the proof of Theorem 2.5,s the strategy (2.6) considered here consists of "whitening" the feature vectors $\mathbf{x}_t$ into $\widetilde{\mathbf{x}}_t = \mathbf{G}_T^{-1/2} \mathbf{x}_t$ and applying the "classic" non-linear ridge regression (2.2) to these whitened feature vectors $\widetilde{\mathbf{x}}_t$. Their associated Gram matrix is the identity, which helps obtaining a sharper bound from Theorem 2.2 than the suboptimal but general bound obtained in Corollary 2.3.

Theorem 2.5. *Let the non-linear ridge regression algorithm with adapted regularization* (2.6) *be run with parameter $\lambda > 0$. For all $T \geqslant 1$, for all feature sequences $\mathbf{x}_1, \dots, \mathbf{x}_T \in \mathbb{R}^d$ and all $y_1, \dots, y_T \in [-B, B]$,*

$$\sup_{\mathbf{u} \in \mathbb{R}^d} \mathcal{R}_T(\mathbf{u}) \leqslant \lambda T B^2 + r_T B^2 \ln\!\left(1 + \frac{1}{\lambda}\right),$$

where $r_T = \operatorname{rank}(\mathbf{G}_T)$.

By taking $\lambda = r_T/T$, we get the bound $r_T B^2 \Big(1 + \ln(1 + T/r_T)\Big)$. Of course, $r_T \leqslant d$ and since $u \mapsto (1/u)\ln(1+u)$ is decreasing over $(0, +\infty)$, the final optimized regret bound reads

$$\sup_{\mathbf{u} \in \mathbb{R}^d} \mathcal{R}_T(\mathbf{u}) \leqslant B^2 \left(r_T \ln\!\left(1 + \frac{T}{r_T}\right) + r_T \right) \leqslant d\, B^2 \ln\!\left(1 + \frac{T}{d}\right) + d\, B^2\,.$$

Note that the leading constant is 1, which is known to be optimal because of Theorem 2.4.

Remark 3. Bartlett et al. [2015] study some minimax uniform regret, namely

$$\mathcal{R}_T^{\star} = \sup_{\substack{\mathbf{x}_1, \dots, \mathbf{x}_T \in \mathbb{R}^d \\ \text{satisfying (2.9)}}} \inf_{\widehat{y}_1} \sup_{y_1 \in [-B,B]} \cdots \inf_{\widehat{y}_T} \sup_{y_T \in [-B,B]} \sup_{\mathbf{u} \in \mathbb{R}^d} \mathcal{R}_T(\mathbf{u})\,,$$

and design a forecaster called MM based on backward induction. It uses vectors $\widehat{\mathbf{u}}_t = \mathbf{P}_t \mathbf{b}_{t-1}$ for $t \geqslant 2$, where the sequence $\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_T$ is defined in a backward manner as

$$\mathbf{P}_T = \mathbf{G}_T^{\dagger} \qquad \text{and} \qquad \mathbf{P}_{t-1} = \mathbf{P}_t + \mathbf{P}_t \mathbf{x}_t \mathbf{x}_t^{\mathrm{T}} \mathbf{P}_t\,.$$

Because MM is minimax optimal if the (stringent) conditions

$$\forall t \in \{1, \dots, T\}, \qquad \sum_{s=1}^{t-1} \left| \mathbf{x}_s{}^{\mathrm{T}} \mathbf{P}_t \mathbf{x}_t \right| \leqslant 1 \,, \tag{2.9}$$

on the feature sequence $\mathbf{x}_1, \dots, \mathbf{x}_T$ are met, a consequence of Theorem 2.5 is that MM also satisfies the regret bound $dB^2\big(1 + \ln(1 + T/d)\big)$ for those feature sequences. Bartlett et al. [2015, Theorem 8] showed a regret bound with a leading term of $2dB^2 \ln T$. This closed-form bound actually also held for any sequence of features, not only the ones satisfying (2.9), but it is suboptimal by a multiplicative factor of 2. See Section 2.7.1 for further technical details. We do not know whether this suboptimal bound is unavoidable (i.e., is due to the algorithm itself) or whether a different analysis could lead to a better bound for the MM forecaster on sequences not satisfying (2.9).

Remark 4. It is worth to notice that our result holds in a less restrictive setting than beforehand-known features. Indeed, in the definition of the weight vector $\widehat{\mathbf{u}}_t$, see Equations (2.6) and (2.8), the only forward information used lies in the regularization term $\lambda\, \mathbf{u}^{\mathrm{T}} \mathbf{G}_T \mathbf{u}$. Therefore, our algorithm does not need to know the whole sequence of features $\mathbf{x}_1, \dots, \mathbf{x}_T$ in advance: it is enough to know the Gram matrix $\mathbf{G}_T$, in which case our results still hold true. A particular case is when the sequence of features is only known beforehand up to an unknown (and possibly random) permutation, as considered, e.g., by Kotłowski et al. [2017].

Proof. In order to keep things simple, we will assume here that $\mathbf{G}_T$ is full rank; the proof in the general case can be found in Section 2.7.2. Then, all matrices $\lambda\, \mathbf{G}_T + \mathbf{G}_t$ are full rank as well.

The proof of this theorem relies on the bound of the non-linear ridge regression algorithm of Section 2.2.1, applied on a modified sequence of features

$$\widetilde{\mathbf{x}}_t = \mathbf{G}_T^{-1/2} \mathbf{x}_t \,,$$

where $\mathbf{G}_T^{-1/2}$ is the inverse square root of the of the symmetric matrix $\mathbf{G}_T$. We successively prove the following two inequalities (where we replaced r_T by its value d, as $\mathbf{G}_T$ is full rank),

$$\sum_{t=1}^{T} (y_t - \widehat{\mathbf{u}}_t \cdot \mathbf{x}_t)^2 \leqslant \inf_{\mathbf{u} \in \mathbb{R}^d} \left\{ \sum_{t=1}^{T} (y_t - \mathbf{u} \cdot \widetilde{\mathbf{x}}_t)^2 + \lambda \left\| \mathbf{u} \right\|^2 \right\} + dB^2 \ln\left(1 + \frac{1}{\lambda}\right) \tag{2.10}$$

$$\leqslant \inf_{\mathbf{u} \in \mathbb{R}^d} \left\{ \sum_{t=1}^{T} (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 \right\} + \lambda T B^2 + dB^2 \ln\left(1 + \frac{1}{\lambda}\right). \tag{2.11}$$

Proof of (2.10). We first show that the strategy (2.2) on the $\widetilde{\mathbf{x}}_t$ leads to the same forecasts as the strategy (2.6) on the original $\mathbf{x}_t$; that is, we show that

$$\widetilde{\mathbf{u}}_t \cdot \widetilde{\mathbf{x}}_t = \widehat{\mathbf{u}}_t \cdot \mathbf{x}_t \,, \qquad \text{where} \qquad \widetilde{\mathbf{u}}_t \in \operatorname*{argmin}_{\mathbf{u} \in \mathbb{R}^d} \left\{ \sum_{s=1}^{t-1} (y_s - \mathbf{u} \cdot \widetilde{\mathbf{x}}_s)^2 + (\mathbf{u} \cdot \widetilde{\mathbf{x}}_t)^2 + \lambda \left\| \mathbf{u} \right\|^2 \right\}. \tag{2.12}$$

The equality above follows from the definition $\widetilde{\mathbf{x}}_t = \mathbf{G}_T^{-1/2} \mathbf{x}_t$ and the fact that $\widetilde{\mathbf{u}}_t = \mathbf{G}_T^{1/2} \widehat{\mathbf{u}}_t$.

Indeed, the closed-form expression (2.3) indicates that

$$\widetilde{\mathbf{u}}_t = \left(\lambda\,\mathbf{I}_d + \sum_{s=1}^{t} \widetilde{\mathbf{x}}_s \widetilde{\mathbf{x}}_s^{\mathrm{T}}\right)^{-1} \sum_{s=1}^{t-1} y_s \widetilde{\mathbf{x}}_s = \left(\lambda \mathbf{I}_d + \mathbf{G}_T^{-1/2} \mathbf{G}_t \mathbf{G}_T^{-1/2}\right)^{-1} \mathbf{G}_T^{-1/2} \mathbf{b}_{t-1}\,.$$

Now,

$$\left(\lambda \mathbf{I}_d + \mathbf{G}_T^{-1/2} \mathbf{G}_t \mathbf{G}_T^{-1/2}\right)^{-1} = \left(\mathbf{G}_T^{-1/2}\Big(\lambda \mathbf{G}_T + \mathbf{G}_t\Big)\mathbf{G}_T^{-1/2}\right)^{-1} = \mathbf{G}_T^{1/2}\Big(\lambda \mathbf{G}_T + \mathbf{G}_t\Big)^{-1}\mathbf{G}_T^{1/2}\,,$$

so that

$$\widetilde{\mathbf{u}}_t = \mathbf{G}_T^{1/2}\Big(\lambda \mathbf{G}_T + \mathbf{G}_t\Big)^{-1}\mathbf{G}_T^{1/2}\mathbf{G}_T^{-1/2}\mathbf{b}_{t-1} = \mathbf{G}_T^{1/2}\Big(\lambda \mathbf{G}_T + \mathbf{G}_t\Big)^{-1}\mathbf{b}_{t-1} = \mathbf{G}_T^{1/2}\widehat{\mathbf{u}}_t\,.$$

We apply Theorem 2.2 on sequences $\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_T \in \mathbb{R}^d$ and $y_1, \ldots, y_T \in [-B, B]$, to get, for all $\mathbf{u} \in \mathbb{R}^d$,

$$\sum_{t=1}^{T}(y_t - \widehat{\mathbf{y}}_t)^2 = \sum_{t=1}^{T}(y_t - \widetilde{\mathbf{u}}_t \cdot \widetilde{\mathbf{x}}_t)^2 \leqslant \sum_{t=1}^{T}(y_t - \mathbf{u}\cdot\widetilde{\mathbf{x}}_t)^2 + \lambda\,\|\mathbf{u}\|^2 + B^2 \sum_{k=1}^{d} \ln\left(1 + \frac{\lambda_k\Big(\sum_{t=1}^{T}\widetilde{\mathbf{x}}_t\widetilde{\mathbf{x}}_t^{\mathrm{T}}\Big)}{\lambda}\right)\,. \tag{2.13}$$

The Gram matrix of the $\widetilde{\mathbf{x}}_t$ equals

$$\sum_{t=1}^{T} \widetilde{\mathbf{x}}_t \widetilde{\mathbf{x}}_t^{\mathrm{T}} = \mathbf{G}_T^{-1/2}\left(\sum_{t=1}^{T} \mathbf{x}_t \mathbf{x}_t{}^{\mathrm{T}}\right)\mathbf{G}_T^{-1/2} = \mathbf{G}_T^{-1/2}\,\mathbf{G}_T\,\mathbf{G}_T^{-1/2} = \mathbf{I}_d\,, \tag{2.14}$$

so that

$$\sum_{k=1}^{d} \ln\left(1 + \frac{\lambda_k\Big(\sum_{t=1}^{T}\widetilde{\mathbf{x}}_t\widetilde{\mathbf{x}}_t^{\mathrm{T}}\Big)}{\lambda}\right) = d\ln\Big(1 + \frac{1}{\lambda}\Big)\,.$$

Taking the infinimum over $\mathbf{u}$ in $\mathbb{R}^d$ in (2.13) concludes the proof of (2.10).

Proof of (2.11). We bound

$$\inf_{\mathbf{u}\in\mathbb{R}^d}\left\{\sum_{t=1}^{T}(y_t - \mathbf{u}\cdot\widetilde{\mathbf{x}}_t)^2 + \lambda\,\|\mathbf{u}\|^2\right\}\,,$$

by evaluating it at $\mathbf{u}^\star \in \operatorname*{argmin}_{\mathbf{u}\in\mathbb{R}^d}\left\{\sum_{t=1}^{T}(y_t - \mathbf{u}\cdot\widetilde{\mathbf{x}}_t)^2\right\}$, which is a singleton with closed-form expression

$$\mathbf{u}^\star = \left(\sum_{t=1}^{T}\widetilde{\mathbf{x}}_t\widetilde{\mathbf{x}}_t^{\mathrm{T}}\right)^{-1}\left(\sum_{t=1}^{T} y_t\widetilde{\mathbf{x}}_t\right) = \mathbf{G}_T^{-1/2}\mathbf{b}_T\,,$$

where we used (2.14) and where $\mathbf{b}_T$ was defined in (2.3). We first bound $\|\mathbf{u}^\star\|^2$. By denoting

$$\mathbf{X}_T = \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_T \end{bmatrix} \qquad \text{and} \qquad y_T = \begin{bmatrix} y_1 \\ \vdots \\ y_T \end{bmatrix}\,,$$

which are respectively, a $d \times T$ and a $T \times 1$ matrix, we have

$$\mathbf{u}^\star = \mathbf{G}_T^{-1/2}\mathbf{X}_T y_T\,, \qquad \text{thus} \qquad \|\mathbf{u}^\star\|^2 = \underline{\mathbf{Y}}_T^{\mathrm{T}}\mathbf{X}_T^{\mathrm{T}}\mathbf{G}_T^{-1}\mathbf{X}_T y_T\,. \tag{2.15}$$

Noting that $\mathbf{X}_T^{\mathrm{T}}\mathbf{G}_T^{-1}\mathbf{X}_T$ is an orthogonal projection (on the image of $\mathbf{X}_T^{\mathrm{T}}$) entails the inequalities $\|\mathbf{u}^\star\|^2 \leqslant \|y_T\|^2 \leqslant TB^2$. Putting all elements together, we proved so far

$$\inf_{\mathbf{u}\in\mathbb{R}^d}\left\{\sum_{t=1}^T (y_t - \mathbf{u}\cdot\widetilde{\mathbf{x}}_t)^2 + \lambda\|\mathbf{u}\|^2\right\} \leqslant \inf_{\mathbf{u}\in\mathbb{R}^d}\left\{\sum_{t=1}^T (y_t - \mathbf{u}\cdot\widetilde{\mathbf{x}}_t)^2\right\} + \lambda T B^2\,.$$

We conclude the proof of (2.11) by a change of dummy variable $\mathbf{v} = \mathbf{G}_T^{1/2}\mathbf{u}$ and the fact that since $\mathbf{G}_T$ is full rank, its image is $\mathbb{R}^d$:

$$\inf_{\mathbf{u}\in\mathbb{R}^d}\left\{\sum_{t=1}^T (y_t - \mathbf{u}\cdot\widetilde{\mathbf{x}}_t)^2\right\} = \inf_{\mathbf{u}\in\mathbb{R}^d}\left\{\sum_{t=1}^T (y_t - \mathbf{G}_T^{1/2}\mathbf{u}\cdot\mathbf{x}_t)^2\right\} = \inf_{\mathbf{v}\in\mathbb{R}^d}\left\{\sum_{t=1}^T (y_t - \mathbf{v}\cdot\mathbf{x}_t)^2\right\}\,.$$

□

2.4 Sequentially revealed features / New result

In this section we do not assume that the features are known beforehand (i.e., unlike in the previous section) and yet exhibit a simple forecaster with a regret bound of $dB^2\ln T + \mathcal{O}_T(1)$ holding uniformly over $\mathbb{R}^d$. Perhaps unexpectedly, the solution that we propose is just to remove the regularization term $\lambda\|\mathbf{u}\|^2_{\mathbf{G}_T}$ in (2.6), which cannot be computed in advance. This amounts to considering the standard non-linear ridge regression algorithm (2.2) with a regularization factor $\lambda = 0$. The reason why this is a natural choice is explained in Remark 7 below.

Thus, weights vectors defined as in Equations (2.2) or (2.6) with regularization parameter $\lambda = 0$ are picked: $\widehat{\mathbf{u}}_1 = (0,\dots,0)^{\mathrm{T}}$ and for $t \geqslant 2$,

$$\widehat{\mathbf{u}}_t \in \operatorname*{argmin}_{\mathbf{u}\in\mathbb{R}^d}\left\{\sum_{s=1}^{t-1}(y_s - \mathbf{u}\cdot\mathbf{x}_s)^2 + (\mathbf{u}\cdot\mathbf{x}_t)^2\right\}, \qquad \text{hence} \qquad \widehat{\mathbf{u}}_t = \mathbf{G}_t^{\dagger}\mathbf{b}_{t-1}\,, \tag{2.16}$$

where the closed-form expression corresponds to (2.7). It then predicts $\widehat{\mathbf{y}}_t = \widehat{\mathbf{u}}_t\cdot\mathbf{x}_t$ (and as already indicated after (2.2), no clipping can take place as B is unknown to the learner).

Note that no parameter requires to be tuned in this case, which can be a true relief.

Remark 5. The traditional bound for the non-linear ridge regression forecaster blows up when the regularization parameter is set as $\lambda = 0$ (see Section 2.2.1) but, perhaps surprisingly, an ad hoc analysis could be performed here—see Theorem 2.6. It provides a new understanding of this well-known non-linear regression algorithm: the regularization term $\lambda\|\mathbf{u}\|^2$ in its defining equation (2.2) is not so useful, while the seemingly harmless regularization term $(\mathbf{u}\cdot\mathbf{x}_t)^2$ therein is crucial.

The theorem below follows from a combination of arguments all already present in the literature, namely, [Forster and Warmuth, 2003, Theorem 3.2], Cesa-Bianchi et al. [2005, Lemma D.1], Luo et al. [2016, Theorem 4 of Appendix D], with a slightly more careful analysis at only one small point in the proof of the latter; see details in Section 2.8. The proof is actually based on the proof of Theorem 2.2 but requires adaptations to account for the fact that $\widehat{\mathbf{u}}_t$ is defined in (2.16) in terms of a possibly non-invertible matrix $\mathbf{G}_t$. There are strong links between the results of Theorem 2.6 and Theorem 2.2; see Remark 6 below.

The result of Theorem 2.6 is not that straightforward, and in particular, some tricks that were suggested to us when presenting this work, e.g., neglecting finitely many rounds till the Gram matrix is full rank (if this ever happens), would probably work but would lead to an even larger constant term. Generally speaking, neglecting finitely many rounds may have important side-effects, see an illustration in Remark 6.

Theorem 2.6. *The non-linear ridge regression algorithm with $\lambda = 0$ as in* (2.16) *achieves for all $T \geqslant 1$, for all sequences $\mathbf{x}_1, \dots, \mathbf{x}_T \in \mathbb{R}^d$ and all $y_1, \dots, y_T \in [-B, B]$, the uniform regret bound*

$$\sup_{\mathbf{u}\in\mathbb{R}^d} \mathcal{R}_T(\mathbf{u}) \leqslant B^2 \sum_{t=1}^{T} \mathbf{x}_t^\intercal \mathbf{G}_t^\dagger \mathbf{x}_t \leqslant B^2 \sum_{k=1}^{r_T} \ln\Big(\lambda_k(\mathbf{G}_T)\Big) + B^2 \sum_{t\in[\![1,T]\!]\cap\mathcal{T}} \ln\left(\frac{1}{\lambda_{r_t}(\mathbf{G}_t)}\right) + r_T B^2$$

where r_t and λ_k are defined in Notation 2.1, and where the set $\mathcal{T}$ contains r_T rounds, given by the smallest $s \geqslant 1$ for whic $\mathbf{x}_s$ is not null, and all $s \geqslant 2$ such that $\operatorname{rank}(\mathbf{G}_{s-1}) \neq \operatorname{rank}(\mathbf{G}_s)$.

We recall in Appendix 2.A.1 that $\operatorname{rank}(\mathbf{G}_t)$ is a non-decreasing sequence, with increments of 1, hence the claimed cardinality r_T of $\mathcal{T}$, and the fact that $\lambda_{r_t}(\mathbf{G}_t) > 0$ for all $t \in \mathcal{T}$.

Note that the regret bound obtained is scale invariant, which is natural and was expected, as the forecaster also is; to see why this is the case, note that it only involves quantities $\lambda_k(\mathbf{G}_T)/\lambda_{r_t}(\mathbf{G}_t)$.

The same (standard) arguments as the ones at the end of the proof of Corollary 2.3 show the following consequence of this bound (which is scale invariant as far as multiplications of the features by scalar factors only are concerned): for all $X > 0$, for all sequences $\mathbf{x}_1, \mathbf{x}_2, \dots$ of features with $\|\mathbf{x}_t\| \leqslant X$,

$$\sup_{\mathbf{u}\in\mathbb{R}^d} \mathcal{R}_T(\mathbf{u}) \leqslant dB^2 \ln T + \underbrace{dB^2 + B^2 \sum_{t\in[\![1,T]\!]\cap\mathcal{T}} \ln\left(\frac{X^2}{\lambda_{r_t}(\mathbf{G}_t)}\right)}_{\text{this is our } \mathcal{O}_T(1) \text{ here}}.$$

Note that the $\mathcal{O}_T(1)$ term stops increasing once the matrix $\mathbf{G}_T$ is full rank: then, for rounds $T' \geqslant T$, only the leading term increases to $dB^2 \ln T'$. But this $\mathcal{O}_T(1)$ can admittedly be large and blows up as all sequences of feature vectors are considered, just as in the bound of Corollary 2.3. The dependency on the eigenvalues is however slightly improved to a logarithmic one, here.

The bound of Theorem 2.6 thus still remains somewhat weak, hence our main open question.

2.4.1 Open question—Double uniformity over $\mathbb{R}^d$: for the $\mathbf{u}$ and for the $\mathbf{x}_t$

For the time being, no $dB^2 \ln(T) + \mathcal{O}_T(1)$ regret bound simultaneously uniform over all comparison vectors $\mathbf{u} \in \mathbb{R}^d$ and over all features $\mathbf{x}_t$ with $\|\mathbf{x}_t\| \leqslant X$ and bounded observations $y_t \in [-B, B]$ is provided in the case of sequentially revealed features (what we called worst-case uniform regret bounds). Indeed, the bound of Theorem 2.2 is not uniform over the comparison vectors $\mathbf{u} \in \mathbb{R}^d$. The bound of Corollary 2.3 is of order $2dB^2 \ln(T)$ (and is not uniform over even bounded feature vectors). The bound of Theorem 2.5 enjoys the double uniformity and is of proper order, but only holds for beforehand-known features. The bound of Theorem 2.6 is uniform over the comparison vectors $\mathbf{u} \in \mathbb{R}^d$, is of proper order $dB^2 \ln(T)$ and holds in the sequential case, but is not uniform over bounded features $\mathbf{x}_t$ (its remainder term can be large).

The lower bound of Theorem 2.4 (and earlier lower bounds by Vovk, 2001 and Takimoto and Warmuth, 2000) are proved in the case of feature vectors that are initially revealed to the regression strategy. The open question is therefore whether we can improve the lower bound and make it larger for strategies that only discover the features on the fly, or if a doubly uniform regret upper bound of $dB^2 \ln(T) + \mathcal{O}_T(1)$ over the $\mathbf{u}$ and the $\mathbf{x}_t$, y_t is also possible in the case of sequentially revealed features. For the lower bound, it seems that choosing random feature vectors that are independent over time might not be a good idea, since the final normalized Gram matrix $\mathbf{G}_T/T$ may be concentrated around its expectation $\mathbf{G}$, and the regression strategy might use the possibly known $\mathbf{G}$ to transform the features $\mathbf{x}_t$ as in Theorem 2.5. Instead, choosing random features $\mathbf{x}_t$ that are dependent over time might make the task of predicting the final Gram matrix $\mathbf{G}_T$ virtually impossible, and might help to improve the lower bound. Alternatively, we could construct features $\mathbf{x}_t$ in a truly sequential manner, as functions of the strategy's past predictions, so as to annoy the regression strategy.

2.4.2 Some further technical remarks

We provide details on two claims issued above.

Remark 6 (Links between Theorem 2.6 and Theorem 2.2)**.** Assume that we use the nonlinear ridge regression algorithm with $\lambda = 0$ as in (2.16) but feed it first with d warm-up feature vectors $\mathbf{x}_{-t} = (0, \ldots, 0, \sqrt{\lambda}, 0, \ldots, 0)$, where the $\sqrt{\lambda}$ is in position $t \in \{1, \ldots, d\}$, and that the observations are $y_{-t} = 0$. Then for each $\mathbf{u} \in \mathbb{R}^d$, a cumulative loss of $\lambda \|\mathbf{u}\|^2$ is suffered, and to neglect these d additional rounds in the regret bound obtained by Theorem 2.6, we need to add a $\lambda \|\mathbf{u}\|^2$ term to it. As all terms corresponding to the new eigenvalues introduced $\lambda_{r_t}(\mathbf{G}_t)$ are equal to λ, given the choice of these warm-up features, we are thus essentially back to the bound of Theorem 2.2.

Remark 7 (How we came up with the forecaster (2.16))**.** A natural attempt to transform the forecaster (2.6) designed for the case of beforehand-known features into a fully sequential algorithm is to replace the matrix $\mathbf{G}_T$ that is unknown at the beginning of round t by its sequential estimate $\mathbf{G}_t$ and to regularize at time t with $(\mathbf{u} \cdot \mathbf{x}_t)^2 + \lambda \|\mathbf{u}\|_{\mathbf{G}_t}^2$ instead of $(\mathbf{u} \cdot \mathbf{x}_t)^2 + \lambda \|\mathbf{u}\|_{\mathbf{G}_T}^2$ as in (2.6). However, in this case, the closed-form expression for the vector $\widehat{\mathbf{u}}_t$ is $\widehat{\mathbf{u}}_t = \mathbf{G}_t^\dagger \mathbf{b}_{t-1}/(1+\lambda)$, that is, the λ only acts as a multiplicative bias to the vector otherwise considered in (2.16). The analysis we followed led to a regret bound increasing in λ, so that we finally picked $\lambda = 0$ and ended up with our non-linear ridge regression algorithm with $\lambda = 0$ as in (2.16).

2.5 Details on the proof of Theorem 2.4

Details on getting (2.5)

By exchanging an expectation and an infimum, the expectation of the uniform regret of any fixed forecaster considered can be bounded as

$$\mathbb{E}_{\theta^\star}\left[\sup_{\mathbf{u}\in\mathbb{R}^d} \mathcal{R}_T(\mathbf{u})\right] \geqslant \sum_{t=1}^T \mathbb{E}_{\theta^\star}\left[(Y_t - \widehat{\mathbf{y}}_t)^2\right] - \inf_{\mathbf{u}\in\mathbb{R}^d} \sum_{t=1}^T \mathbb{E}_{\theta^\star}\left[(Y_t - \mathbf{u}\cdot\mathbf{e}_{J_t})^2\right]. \tag{2.17}$$

Since $\widehat{\mathbf{y}}_t$ is measurable w.r.t. $\mathcal{F}_{t-1}$, the σ–algebra generated by the information available at the beginning of round t, namely, $J_1, \dots, J_T$ and $Y_1, \dots, Y_{t-1}$, and since Y_t is distributed, conditionally on $\mathcal{F}_{t-1}$ according to a Bernoulli distribution with parameter $\theta^\star_{J_t}$, a conditional bias–variance decomposition yields

$$\begin{aligned}\mathbb{E}_{\theta^\star}\left[(\widehat{\mathbf{y}}_t - Y_t)^2 \,\middle|\, \mathcal{F}_{t-1}\right] &= (\widehat{\mathbf{y}}_t - \theta^\star_{J_t})^2 + \mathbb{E}_{\theta^\star}\left[(Y_t - \theta^\star_{J_t})^2 \,\middle|\, \mathcal{F}_{t-1}\right] \\ &= \left(\widehat{u}_{J_t,t} - \theta^\star_{J_t}\right)^2 + \theta^\star_{J_t}\left(1 - \theta^\star_{J_t}\right),\end{aligned}$$

where we also used that by construction, $\widehat{\mathbf{y}}_t = \widehat{\mathbf{u}}_t \cdot \mathbf{e}_{J_t} = \widehat{u}_{J_t,t}$. Similarly, for all $\mathbf{u} \in \mathbb{R}^d$,

$$\mathbb{E}_{\theta^\star}\left[(Y_t - \mathbf{u}\cdot\mathbf{e}_{J_t})^2 \,\middle|\, \mathcal{F}_{t-1}\right] = \left(u_{J_t} - \theta^\star_{J_t}\right)^2 + \theta^\star_{J_t}\left(1 - \theta^\star_{J_t}\right).$$

By the tower rule and since the variance terms $\theta^\star_{J_t}\left(1 - \theta^\star_{J_t}\right)$ cancel out, we thus proved that

$$\begin{aligned}\mathbb{E}_{\theta^\star}\left[\sup_{\mathbf{u}\in\mathbb{R}^d} \mathcal{R}_T(\mathbf{u})\right] &\geqslant \sum_{t=1}^T \mathbb{E}_{\theta^\star}\left[(\widehat{\mathbf{y}}_t - Y_t)^2\right] - \inf_{\mathbf{u}\in\mathbb{R}^d} \sum_{t=1}^T \mathbb{E}_{\theta^\star}\left[(Y_t - \mathbf{u}\cdot\mathbf{e}_{J_t})^2\right] \\ &= \sum_{t=1}^T \mathbb{E}_{\theta^\star}\left[\left(\widehat{u}_{J_t,t} - \theta^\star_{J_t}\right)^2\right] - \inf_{\mathbf{u}\in\mathbb{R}^d} \sum_{t=1}^T \mathbb{E}_{\theta^\star}\left[\left(u_{J_t} - \theta^\star_{J_t}\right)^2\right] \\ &= \sum_{t=1}^T \mathbb{E}_{\theta^\star}\left[(\widehat{u}_{J_t,t} - \theta^\star_{J_t})^2\right].\end{aligned}$$

Now, by resorting to the tower rule again, integrating over J_t conditionally on $Y_1, \dots, Y_{t-1}$ and $J_1, \dots, J_{t-1}, J_{t+1}, \dots, J_T$, we get

$$\mathbb{E}_{\theta^\star}\left[\sup_{\mathbf{u}\in\mathbb{R}^d} \mathcal{R}_T(\mathbf{u})\right] \geqslant \sum_{t=1}^{T} \mathbb{E}_{\theta^\star}\left[(\widehat{u}_{J_t,t} - \theta^\star_{J_t})^2\right] = \sum_{t=1}^{T} \frac{1}{d}\,\mathbb{E}_{\theta^\star}\left[\left\|\widehat{\mathbf{u}}_t - \theta^\star\right\|_2^2\right]. \tag{2.18}$$

We now show that each term in the sum is larger than something of the order of d/t. This order of magnitude d/t is the parametric rate of optimal estimation; indeed, due to the randomness of the J_s, over t periods, each component is used about t/d times, while the rate of convergence in quadratic error of any d–dimensional estimator based on $\tau = t/d$ unbiased i.i.d. observations is at best $d/\tau = d^2/t$. Taking into account the $1/d$ factor gets us the claimed d/t rate. The next steps (based on the van Trees inequality) transform this intuition into formal statements.

Conclusion of the proof, given the application of the van Trees inequality

We resume at (2.18) and consider a prior π over the $\theta^\star \in [0,1]^d$. Since an expectation is always smaller than a supremum, we have first, given the defining equation (2.4) of $\mathcal{R}^\star_{T,\,[0,1]}$,

$$\begin{aligned}\mathcal{R}^\star_{T,\,[0,1]} &\geqslant \inf_{\text{forecasters}} \int_{[0,1]^d} \mathbb{E}_{\theta^\star}\left[\sup_{\mathbf{u}\in\mathbb{R}^d} \mathcal{R}_T(\mathbf{u})\right] \mathrm{d}\pi(\theta^\star) \\ &\geqslant \inf_{\text{forecasters}} \sum_{t=1}^{T} \int_{[0,1]^d} \frac{1}{d}\,\mathbb{E}_{\theta^\star}\left[\left\|\widehat{\mathbf{u}}_t - \theta^\star\right\|_2^2\right] \mathrm{d}\pi(\theta^\star)\,,\end{aligned}$$

where the second inequality follows by mixing both sides of (2.18) according to π. Now, an immediate application of the (multi-dimentional) van Trees inequality with a Beta(α,α) prior π shows that for all forecasters, all $t \geqslant 1$ and $\alpha \geqslant 3$,

$$\int_{[0,1]^d} \frac{1}{d}\,\mathbb{E}_{\theta^\star}\left[\left\|\widehat{\mathbf{u}}_t - \theta^\star\right\|_2^2\right] \mathrm{d}\pi(\theta^\star) \geqslant \frac{d}{4(t-1) + 2(t-1)/(\alpha-1) + 16d\alpha}\,,$$

see Lemma 2.7 below. We thus proved

$$\begin{aligned}\mathcal{R}^\star_{T,\,[0,1]} &\geqslant \sum_{t=0}^{T-1} \frac{d}{\big(4 + 2/(\alpha-1)\big)t + 16d\alpha} \geqslant d\int_0^T \frac{1}{\big(4 + 2/(\alpha-1)\big)t + 16d\alpha}\,\mathrm{d}t \\ &= \frac{d}{4 + 2/(\alpha-1)} \ln \frac{\big(4 + 2/(\alpha-1)\big)T + 16d\alpha}{16d\alpha} \\ &\geqslant \frac{d}{4 + 2/(\alpha-1)} \ln \frac{4T}{16d\alpha} \\ &= \frac{d}{4 + 2/(\alpha-1)} \big(\ln T - \ln(4d\alpha)\big)\,,\end{aligned}$$

which we lower bound in a crude way by resorting to $1/(1+u) \geqslant 1-u$ and by taking α such that $\alpha - 1 = \ln T$; this is where our condition $T \geqslant 8 > \mathbf{e}^2$ is used, to ensure that $\alpha \geqslant 3$. We also use that since $T \geqslant \mathbf{e}^2$, we have $1 \leqslant (\ln T)/2$ thus $4d\alpha \leqslant 4d(1 + \ln T) \leqslant 6d \ln T$.

We get

$$\mathcal{R}^\star_{T,\,[0,1]} \geqslant \frac{d}{4}\underbrace{\left(1-\frac{1}{2(\alpha-1)}\right)}_{\geqslant 0}\Big(\ln T - \ln(6d\ln T)\Big)$$
$$\geqslant \frac{d}{4}\left(1-\frac{1}{2\ln T}\right)(\ln T - \ln(6d) - \ln\ln T) \geqslant \frac{d}{4}\Big(\ln T - (3+\ln d) - \ln\ln T\Big)\,. \tag{2.19}$$

The factor 3 above corresponds to $1/2 + \ln 6 \leqslant 3$. So, we covered the case of $\mathcal{R}^\star_{T,\,[0,1]}$ and now turn to $\mathcal{R}^\star_{T,\,[-B,B]}$ for a general $B > 0$.

Going from $\mathcal{R}^\star_{T,\,[0,1]}$ to $\mathcal{R}^\star_{T,\,[-B,B]}$

To get a lower bound of exact order $d\ln T$, that is, to get rid of the annoying multiplicative factor of $1/4$, we proceed as follows. With the notation above, $Z_t = 2B(Y_t - 1/2)$ lies in $[-B, B]$. Denoting by $\widehat{z}_t$ the forecasts output by a given forecaster sequentially fed with the $(Z_s, \mathbf{e}_{J_s})$, we have

$$(\widehat{z}_t - Z_t)^2 = 4B^2(\widehat{\mathbf{y}}_t - Y_t)^2 \qquad \text{where the} \qquad \widehat{\mathbf{y}}_t = \frac{\widehat{z}_t + 1/2}{2B}$$

also correspond to predictions output by a legitimate forecaster, and

$$\inf_{\mathbf{v}\in\mathbb{R}^d}\sum_{t=1}^T \mathbb{E}\Big[(Z_t - \mathbf{v}\cdot\mathbf{e}_{J_t})^2\Big] = 4B^2\inf_{\mathbf{v}\in\mathbb{R}^d}\sum_{t=1}^T \mathbb{E}\left[\left(Y_t - \frac{1}{2} - \frac{\mathbf{v}\cdot\mathbf{e}_{J_t}}{2B}\right)^2\right]$$
$$= 4B^2\inf_{\mathbf{u}\in\mathbb{R}^d}\sum_{t=1}^T \mathbb{E}\Big[(Y_t - \mathbf{u}\cdot\mathbf{e}_{J_t})^2\Big]$$

by considering the transformation $\mathbf{v} \leftrightarrow \mathbf{u}$ given by $u_j = v_j/(2B) - 1/2$. (We use here that the sum of the components of the $\mathbf{e}_{J_t}$ equal 1.) We thus showed that $\mathcal{R}^\star_{T,\,[-B,B]}$ is larger than $4B^2$ times the lower bound (2.19) exhibited on (2.17), which concludes the proof.

Details on the application of the van Trees inequality

The van Trees inequality is a Bayesian version of the Cramér-Rao bound, but holding for any estimator (not only the unbiased ones); see Gill and Levit [1995, Section 4] for a multivariate statement (and refer to Van Trees, 1968 for its first statement).

Recall that we denoted above by $\mathcal{P}_{\theta^\star}$ the distribution of the sequence of pairs (J_t, Y_t), with $1 \leqslant t \leqslant T$, considered in Section 2.2.2 for a given $\theta^\star \in [0,1]^d$. We also considered the family $\mathcal{P}$ of these distributions and thus, for clarity, indexed all expectations $\mathbb{E}$ by the underlying parameter $\theta^\star$ at hand. We introduce a product of independent Beta(α,α) distributions as a prior π on the $\theta^\star \in [0,1]^d$; its density with respect to the Lebesgue measure equals

$$\beta^{(d)}_{\alpha,\alpha}(t_1,\dots,t_d) \longmapsto \beta_{\alpha,\alpha}(t_1)\cdots\beta_{\alpha,\alpha}(t_d)\,, \qquad \text{where} \qquad \beta_{\alpha,\alpha} : t \mapsto \frac{\Gamma(2\alpha)}{\big(\Gamma(\alpha)\big)^2}\,t^{\alpha-1}(1-t)^{\alpha-1}\,.$$

The reason why Beta distributions are considered is because of the form of the Fisher information of the $\mathcal{P}$ family, see calculations (2.22) below.

The multivariate van Trees inequality ensures that for all estimators $\widehat{\mathbf{u}}_t$, that is, for all random variables which are measurable functions of $J_1, \dots, J_T$ and $Y_1, \dots, Y_{t-1}$, we have

$$\int_{[0,1]^d} \mathbb{E}_{\theta^\star}\left[\left\|\widehat{\mathbf{u}}_t - \theta^\star\right\|_2^2\right] \beta_{\alpha,\alpha}^{(d)}(\theta^\star)\, \mathrm{d}\theta^\star \geqslant \frac{(\operatorname{Tr} \mathbf{I}_d)^2}{\operatorname{Tr}\mathcal{I}\Big(\beta_{\alpha,\alpha}^{(d)}\Big) + \displaystyle\int_{[0,1]^d}\Big(\operatorname{Tr}\mathcal{I}(\theta^\star)\Big)\, \beta_{\alpha,\alpha}^{(d)}(\theta^\star)\, \mathrm{d}\theta^\star}\,, \tag{2.20}$$

where $\mathrm{d}\theta^\star$ denotes the integration w.r.t. Lebesgue measure, Tr is the trace operator, $\mathcal{I}(\theta^\star)$ stands for the Fisher information of the family $\mathcal{P}$ at $\theta^\star$, see (2.21), while each component (i,i) of the other matrix in the denominator is given by

$$\mathcal{I}\Big(\beta_{\alpha,\alpha}^{(d)}\Big)_{i,i} \stackrel{\text{def}}{=} \int_{[0,1]^d} \left(\frac{\partial \beta_{\alpha,\alpha}^{(d)}}{\partial \theta_i^\star}(\theta^\star)\right)^2 \frac{1}{\beta_{\alpha,\alpha}^{(d)}(\theta^\star)}\, \mathrm{d}\theta^\star\,,$$

which may equal $+\infty$ (in which case the lower bound is void). There are conditions for the inequality to be satisfied, we detail them in the proof of the lemma below.

Lemma 2.7. *When the family $\mathcal{P}$ is equipped with a prior given by a product of independent Beta(α,α) distributions, where $\alpha \geqslant 3$, it follows from the van Trees inequality and from simple calculations that*

$$\int_{[0,1]^d} \mathbb{E}_{\theta^\star}\left[\left\|\widehat{\mathbf{u}}_t - \theta^\star\right\|_2^2\right] \beta_{\alpha,\alpha}^{(d)}(\theta^\star)\, \mathrm{d}\theta^\star \geqslant \frac{d^2}{16d\alpha + 4(t-1) + 2(t-1)/(\alpha-1)}\,.$$

Proof. We denote by

$$f_{\theta^\star} : (j_1, \dots, j_T,\, y_1, \dots, y_{t-1}) \in \{1,\dots,d\}^T \times \{0,1\}^{t-1} \longmapsto \frac{1}{d^T} \prod_{s=1}^{t-1} {\theta_{j_s}^\star}^{y_s} (1-\theta_{j_s}^\star)^{1-y_s}$$

the density of $\mathcal{P}_{\theta^\star}$ w.r.t. to the counting measure μ on $\{1,\dots,d\}^T \times \{0,1\}^{t-1}$.

The sufficient conditions of Gill and Levit [1995, Section 4] for (2.20) are met, since on the one hand $\beta_{\alpha,\alpha}^{(d)}$ is C^1–smooth, vanishes on the border of $[0,1]^d$, and is positive on its interior, while on the other hand, $\theta^\star \mapsto f_{\theta^\star}(j_1,\dots,j_T,\, y_1,\dots,y_{t-1})$ is C^1–smooth for all $(j_1,\dots,j_T,\, y_1,\dots,y_{t-1})$, with, for all $i \in \{1,\dots,d\}$,

$$L_i(\theta^\star) = \frac{\partial}{\partial \theta_i^\star} \ln f_{\theta^\star}(J_1,\dots,J_T,\, Y_1,\dots,Y_{t-1}) = \sum_{s=1}^{t-1} \left(\frac{Y_s}{\theta_{J_s}^\star} - \frac{1-Y_s}{1-\theta_{J_s}^\star}\right) \mathbb{1}_{\{J_s = i\}}$$

being square integrable, so that the Fisher information matrix $\mathcal{I}(\theta^\star)$ of the $\mathcal{P}$ model at $\theta^\star$ exists and has a component (i,i) given by

$$\begin{aligned}\mathcal{I}(\theta^\star)_{i,i} \stackrel{\text{def}}{=} \mathbb{E}_{\theta^\star}\Big[L_i(\theta^\star)^2\Big] &= (t-1)\, \mathbb{E}_{\theta^\star}\left[\left(\frac{Y_1}{\theta_{J_1}^\star} - \frac{1-Y_1}{1-\theta_{J_1}^\star}\right)^2 \mathbb{1}_{\{J_1=i\}}\right] \\ &= \frac{t-1}{d}\left(\frac{1}{\theta_i^\star} + \frac{1}{1-\theta_i^\star}\right) = \frac{t-1}{d\,\theta_i^\star(1-\theta_i^\star)}\,,\end{aligned} \tag{2.21}$$

and therefore, is such that $\theta^\star \mapsto \sqrt{\mathcal{I}(\theta^\star)}$ is locally integrable w.r.t. the Lebesgue measure.

The second inequality in (2.21) is because $L_i(\theta^\star)$ is a sum of $t-1$ centered, independent and identically distributed variables, while the third inequality is obtained by the tower rule, by first taking the conditional expectation with respect to J_1.

We now compute all elements of the denominator of (2.20). First, by symmetry and then by substituting (2.21),

$$\begin{aligned}
&\int_{[0,1]^d} \Big(\operatorname{Tr} \mathcal{I}(\theta^\star)\Big)\, \beta_{\alpha,\alpha}^{(d)}(\theta^\star)\, \mathrm{d}\theta^\star \\
&= d \int_{[0,1]^d} \mathcal{I}(\theta^\star)_{1,1}\, \beta_{\alpha,\alpha}^{(d)}(\theta^\star)\, \mathrm{d}\theta^\star \\
&= d \int_{[0,1]^d} \frac{t-1}{d\,\theta_1^\star(1-\theta_1^\star)}\, \frac{\Gamma(2\alpha)}{\Big(\Gamma(\alpha)\Big)^2}\, (\theta_1^\star)^{\alpha-1}(1-\theta_1^\star)^{\alpha-1}\, \beta_{\alpha,\alpha}(\theta_2^\star)\cdots \beta_{\alpha,\alpha}(\theta_d^\star)\, \mathrm{d}\theta^\star \qquad (2.22) \\
&= (t-1)\, \frac{\Gamma(2\alpha)}{\Big(\Gamma(\alpha)\Big)^2} \int_{[0,1]} z^{\alpha-2}(1-z)^{\alpha-2}\, \mathrm{d}z = (t-1)\, \frac{\Gamma(2\alpha)}{\Big(\Gamma(\alpha)\Big)^2} \frac{\Big(\Gamma(\alpha-1)\Big)^2}{\Gamma\Big(2(\alpha-1)\Big)}\,,
\end{aligned}$$

where we used the expression of the density of the Beta$(\alpha-1,\alpha-1)$ distribution for the last equality. Using that $x\,\Gamma(x) = \Gamma(x+1)$ for all real numbers $x>0$, we finally get

$$\int_{[0,1]^d} \Big(\operatorname{Tr} \mathcal{I}(\theta^\star)\Big)\, \beta_{\alpha,\alpha}^{(d)}(\theta^\star)\, \mathrm{d}\theta^\star = \frac{(2\alpha-1)(2\alpha-2)}{(\alpha-1)^2}\,(t-1) = 4(t-1) + \frac{2(t-1)}{\alpha-1}\,.$$

Second, as far as the $\operatorname{Tr}\mathcal{I}\Big(\beta_{\alpha,\alpha}^{(d)}\Big)$ in (2.20) is concerned, because $\beta_{\alpha,\alpha}^{(d)}$ is a product of univariate distributions,

$$\mathcal{I}\Big(\beta_{\alpha,\alpha}^{(d)}\Big)_{i,i} = \int_{[0,1]^d} \left(\frac{\partial \beta_{\alpha,\alpha}^{(d)}}{\partial \theta_i^\star}(\theta^\star)\right)^2 \frac{1}{\beta_{\alpha,\alpha}^{(d)}(\theta^\star)}\, \mathrm{d}\theta^\star = \int_{[0,1]} \left(\frac{\partial \beta_{\alpha,\alpha}}{\partial z}(z)\right)^2 \frac{1}{\beta_{\alpha,\alpha}}(z)\, \mathrm{d}z\,,$$

so that $\operatorname{Tr}\mathcal{I}\Big(\beta_{\alpha,\alpha}^{(d)}\Big)$ equals d times this value, that is, d times

$$\begin{aligned}
&\int_{[0,1]} \frac{\Gamma(2\alpha)}{\Big(\Gamma(\alpha)\Big)^2}\, \frac{\Big((\alpha-1)\,z^{\alpha-2}(1-z)^{\alpha-1} - (\alpha-1)\,z^{\alpha-1}(1-z)^{\alpha-2}\Big)^2}{z^{\alpha-1}(1-z)^{\alpha-1}}\, \mathrm{d}z \\
&= \frac{(\alpha-1)^2\,\Gamma(2\alpha)}{\Big(\Gamma(\alpha)\Big)^2} \int_{[0,1]} (1-2z)^2\; z^{\alpha-3}(1-z)^{\alpha-3}\, \mathrm{d}z \\
&= \frac{(\alpha-1)^2\,\Gamma(2\alpha)}{\Big(\Gamma(\alpha)\Big)^2}\, \frac{\Big(\Gamma(\alpha-2)\Big)^2}{\Gamma\Big(2(\alpha-2)\Big)}\, \mathbb{E}\Big[(1-2Z_{\alpha-2})^2\Big] \\
&= \frac{(\alpha-1)^2\,\Gamma(2\alpha)}{\Big(\Gamma(\alpha)\Big)^2}\, \frac{\Big(\Gamma(\alpha-2)\Big)^2}{\Gamma\Big(2(\alpha-2)\Big)}\, 4\operatorname{Var}(Z_{\alpha-2})
\end{aligned}$$

where $Z_{\alpha-2}$ is a random variable following the Beta$(\alpha-2,\alpha-2)$ distribution; its expectation equals indeed $\mathbb{E}[Z_{\alpha-2}] = 1/2$ by symmetry of the distribution w.r.t. $1/2$, so that

$$\mathbb{E}\Big[(1-2Z_{\alpha-2})^2\Big] = 4\;\mathbb{E}\Big[(1/2-Z_{\alpha-2})^2\Big] = 4\operatorname{Var}(Z_{\alpha-2}) \quad \text{where} \quad \operatorname{Var}(Z_{\alpha-2}) = \frac{1}{4(2\alpha-3)}$$

by a classical formula. Collecting all elements together and using again that for all real numbers $x > 0$, $x\,\Gamma(x) = \Gamma(x+1)$, we get

$$\operatorname{Tr}\mathcal{I}\Big(\beta^{(d)}_{\alpha,\alpha}\Big) = d\,\underbrace{\frac{(\alpha-1)^2\,\Big(\Gamma(\alpha-2)\Big)^2}{\Big(\Gamma(\alpha)\Big)^2}}_{1/(\alpha-2)^2}\,\underbrace{\frac{\Gamma(2\alpha)}{(2\alpha-3)\,\Gamma\Big(2(\alpha-2)\Big)}}_{=(2\alpha-1)(2\alpha-2)(2\alpha-4)} = d\,\frac{4(2\alpha-1)(\alpha-1)}{\alpha-2}$$

hence the upper bound $\operatorname{Tr}\mathcal{I}\Big(\beta^{(d)}_{\alpha,\alpha}\Big) \leqslant 16d\alpha$ for $\alpha \geqslant 3$, which concludes the proof. □

2.6 Proof of Theorem 2.2 and of Corollary 2.3

We start with the proof of Corollary 2.3.

Proof. We assume that the Gram matrix G_T is full rank; otherwise, we may adapt the proof below by resorting to Moore-Penrose pseudoinverses, just as we do in Section 2.7.2 for the proof of Theorem 2.5.

Theorem 2.2 indicates that

$$\sum_{t=1}^{T}(y_t - \widehat{\mathbf{u}}_t\cdot\mathbf{x}_t)^2 \leqslant \inf_{\mathbf{u}\in\mathbb{R}^d}\left\{\sum_{t=1}^{T}(y_t-\mathbf{u}\cdot\mathbf{x}_t)^2 + \lambda\,\|\mathbf{u}\|^2\right\} + B^2\sum_{k=1}^{d}\ln\left(1+\frac{\lambda_k(\mathbf{G}_T)}{\lambda}\right).$$

Now, as in (2.3), we have a closed-form expression of the unique vector achieving the following, infimum:

$$\inf_{\mathbf{u}\in\mathbb{R}^d}\left\{\sum_{t=1}^{T}(y_t-\mathbf{u}\cdot\mathbf{x}_t)^2\right\} = \sum_{t=1}^{T}(y_t-\mathbf{u}^\star\cdot\mathbf{x}_t)^2$$

Namely, $\mathbf{u}^\star = \mathbf{G}_T^{-1}\mathbf{b}_T$, so that

$$\begin{aligned}\|\mathbf{u}^\star\| = \Big\|\mathbf{G}_T^{-1/2}\,\mathbf{G}_T^{-1/2}\mathbf{b}_T\Big\| \leqslant \lambda_1\Big(\mathbf{G}_T^{-1/2}\Big)\,\Big\|\mathbf{G}_T^{-1/2}\mathbf{b}_T\Big\| &= \frac{1}{\sqrt{\lambda_d(\mathbf{G}_T)}}\,\Big\|\mathbf{G}_T^{-1/2}\mathbf{b}_T\Big\| \\ &\leqslant \frac{1}{\sqrt{\lambda_d(\mathbf{G}_T)}}\,B\sqrt{T}\,, \qquad (2.23)\end{aligned}$$

where we used, for the final inequality, an elementary argument of orthogonal projection that is at the heart of the proof of Theorem 2.5: see (2.15) and the sentence after it. In addition, Jensen's inequality (or the alternative treatment of Cesa-Bianchi and Lugosi, 2006, page 320) indicates that

$$\sum_{k=1}^{d}\ln\left(1+\frac{\lambda_k(\mathbf{G}_T)}{\lambda}\right) \leqslant d\ln\left(1+\frac{\sum_{k=1}^{d}\lambda_k(\mathbf{G}_T)}{d\lambda}\right) = d\ln\left(1+\frac{\operatorname{Tr}(\mathbf{G}_T)}{d\lambda}\right) \leqslant d\ln\left(1+\frac{TX^2}{d\lambda}\right)$$

where Tr is the trace operator. All in all, we get

$$\sum_{t=1}^{T}(y_t-\widehat{\mathbf{u}}_t\cdot\mathbf{x}_t)^2 \leqslant \inf_{\mathbf{u}\in\mathbb{R}^d}\left\{\sum_{t=1}^{T}(y_t-\mathbf{u}\cdot\mathbf{x}_t)^2+\right\} + \lambda\,\|\mathbf{u}^\star\|^2 + dB^2\ln\left(1+\frac{TX^2}{d\lambda}\right)$$

and the claimed bound follows by substituting the bound (2.23). □

Now, we move to the proof of Theorem 2.2, which we essentially extract from Cesa-Bianchi and Lugosi [2006, Chapter 11]. We merely provide it because we will later need the first inequality of (2.24) in the proof of Theorem 2.6 and we wanted this chapter to be self-complete. But of course, this is extremely standard content and it should be skipped by any reader familiar with the basic results of sequential linear regression.

Proof. We successively prove the following two inequalities,

$$\mathcal{R}_T(\mathbf{u}) \leqslant \lambda \|\mathbf{u}\|^2 + \sum_{t=1}^{T} y_t^2 \, \mathbf{x}_t^{\mathrm{T}} \mathbf{A}_t^{-1} \mathbf{x}_t \leqslant \lambda \|\mathbf{u}\|^2 + B^2 \sum_{k=1}^{d} \ln\left(1 + \frac{\lambda_k(\mathbf{G}_T)}{\lambda}\right) \tag{2.24}$$

Proof of the first inequality in (2.24). We denote by L_{t-1}^{reg} the cumulative loss up to round $t-1$ included, to which we add the regularization term:

$$L_{t-1}^{\text{reg}}(\mathbf{u}) = \sum_{s=1}^{t-1} (y_s - \mathbf{u} \cdot \mathbf{x}_s)^2 + \lambda \|\mathbf{u}\|^2$$

For all $t \geqslant 1$, we denote by

$$\breve{\mathbf{u}}_t \in \operatorname*{argmin}_{\mathbf{u} \in \mathbb{R}^d} \left\{ \sum_{s=1}^{t-1} (y_s - \mathbf{u} \cdot \mathbf{x}_s)^2 + \lambda \|\mathbf{u}\|^2 \right\} = \operatorname*{argmin}_{\mathbf{u} \in \mathbb{R}^d} L_{t-1}^{\text{reg}}(\mathbf{u}) \,,$$

the vector output by the (ordinary) ridge regression; that is, when no $(\mathbf{u} \cdot \mathbf{x}_t)^2$ term is added to the regularization. In particular, $\breve{\mathbf{u}}_1 = (0, \ldots, 0)^{\mathrm{T}}$. By the very definition of $\breve{\mathbf{u}}_{T+1}$, for all $\mathbf{u} \in \mathbb{R}^d$,

$$L_T^{\text{reg}}(\breve{\mathbf{u}}_{T+1}) \leqslant \sum_{t=1}^{T} (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 + \lambda \|\mathbf{u}\|^2 \,,$$

so that, for all $\mathbf{u} \in \mathbb{R}^d$,

$$\begin{aligned} \mathcal{R}_T(\mathbf{u}) &\leqslant \sum_{t=1}^{T} (y_t - \widehat{\mathbf{y}}_t)^2 + \lambda \|\mathbf{u}\|^2 - L_T^{\text{reg}}(\breve{\mathbf{u}}_{T+1}) \\ &= \lambda \|\mathbf{u}\|^2 + \sum_{t=1}^{T} \Big((y_t - \widehat{\mathbf{y}}_t)^2 + L_{t-1}^{\text{reg}}(\breve{\mathbf{u}}_t) - L_t^{\text{reg}}(\breve{\mathbf{u}}_{t+1}) \Big) \,, \end{aligned}$$

where the equality comes from a telescoping argument together with $L_0^{\text{reg}}(\breve{\mathbf{u}}_0) = 0$. We will prove by means of direct calculations that

$$(y_t - \widehat{\mathbf{y}}_t)^2 + L_{t-1}^{\text{reg}}(\breve{\mathbf{u}}_t) - L_t^{\text{reg}}(\breve{\mathbf{u}}_{t+1}) = (\breve{\mathbf{u}}_{t+1} - \widehat{\mathbf{u}}_t)^{\mathrm{T}} \mathbf{A}_t (\breve{\mathbf{u}}_{t+1} - \widehat{\mathbf{u}}_t) - (\widehat{\mathbf{u}}_t - \breve{\mathbf{u}}_t)^{\mathrm{T}} \mathbf{A}_{t-1} (\widehat{\mathbf{u}}_t - \breve{\mathbf{u}}_t) \,; \tag{2.25}$$

the first inequality in (2.24) will then be obtained, as the second term in (2.25) is negative and as the first term in (2.25) can be rewritten as $y_t^2 \, \mathbf{x}_t^{\mathrm{T}} \mathbf{A}_t^{-1} \mathbf{x}_t$ thanks to the equality (2.27) below, which states $\mathbf{A}_t(\breve{\mathbf{u}}_{t+1} - \widehat{\mathbf{u}}_t) = y_t \mathbf{x}_t$.

To prove (2.25), we recall the closed-form expression (2.3), that is, $\widehat{\mathbf{u}}_t = \mathbf{A}_t^{-1} \mathbf{b}_{t-1}$, and note that we similarly have $\breve{\mathbf{u}}_{t+1} = \mathbf{A}_t^{-1} \mathbf{b}_t$. Now, L_t^{reg} rewrites, for all $\mathbf{u} \in \mathbb{R}^d$,

$$L_t^{\text{reg}}(\mathbf{u}) = \left(\sum_{s=1}^{t} y_s^2 \right) - 2 \mathbf{b}_t^{\mathrm{T}} \mathbf{u} + \mathbf{u}^{\mathrm{T}} \mathbf{A}_t \mathbf{u} \,,$$

so that the minimum of this quadratic form, achieved at $\mathbf{u} = \breve{\mathbf{u}}_{t+1} = \mathbf{A}_t^{-1}\mathbf{b}_t$, equals

$$L_t^{\text{reg}}(\breve{\mathbf{u}}_{t+1}) = \left(\sum_{s=1}^{t} y_s^2\right) - 2\underbrace{\mathbf{b}_t^{\mathsf{T}}\mathbf{A}_t^{-1}}_{=\breve{\mathbf{u}}_{t+1}^{\mathsf{T}}}\mathbf{A}_t\breve{\mathbf{u}}_{t+1} + \breve{\mathbf{u}}_{t+1}^{\mathsf{T}}\mathbf{A}_t\breve{\mathbf{u}}_{t+1} = \left(\sum_{s=1}^{t} y_s^2\right) - \breve{\mathbf{u}}_{t+1}^{\mathsf{T}}\mathbf{A}_t\breve{\mathbf{u}}_{t+1}\,.$$

In particular,

$$L_{t-1}^{\text{reg}}(\breve{\mathbf{u}}_t) - L_t^{\text{reg}}(\breve{\mathbf{u}}_{t+1}) = -y_t^2 + \breve{\mathbf{u}}_{t+1}^{\mathsf{T}}\mathbf{A}_t\breve{\mathbf{u}}_{t+1} - \breve{\mathbf{u}}_t^{\mathsf{T}}\mathbf{A}_{t-1}\breve{\mathbf{u}}_t\,. \tag{2.26}$$

We now expand the first term in (2.25). To that end, we use that from the closed-form expressions of $\widehat{\mathbf{u}}_t$ and $\breve{\mathbf{u}}_{t+1}$,

$$\mathbf{A}_t(\breve{\mathbf{u}}_{t+1} - \widehat{\mathbf{u}}_t) = \mathbf{A}_t\Big(\mathbf{A}_t^{-1}\mathbf{b}_t - \mathbf{A}_t^{-1}\mathbf{b}_{t-1}\Big) = \mathbf{b}_t - \mathbf{b}_{t-1} = y_t\mathbf{x}_t\,. \tag{2.27}$$

Therefore, $y_t\widehat{\mathbf{y}}_t = y_t\mathbf{x}_t^{\mathsf{T}}\widehat{\mathbf{u}}_t = (\breve{\mathbf{u}}_{t+1} - \widehat{\mathbf{u}}_t)^{\mathsf{T}}\mathbf{A}_t\widehat{\mathbf{u}}_t$ and

$$\begin{aligned}(y_t - \widehat{\mathbf{y}}_t)^2 = \mathbf{y}_t^2 - 2y_t\widehat{\mathbf{y}}_t + \widehat{\mathbf{y}}_t^2 &= \mathbf{y}_t^2 - 2(\breve{\mathbf{u}}_{t+1} - \widehat{\mathbf{u}}_t)^{\mathsf{T}}\mathbf{A}_t\widehat{\mathbf{u}}_t + \widehat{\mathbf{u}}_t^{\mathsf{T}}\mathbf{x}_t\mathbf{x}_t^{\mathsf{T}}\widehat{\mathbf{u}}_t \\ &= \mathbf{y}_t^2 - 2(\breve{\mathbf{u}}_{t+1} - \widehat{\mathbf{u}}_t)^{\mathsf{T}}\mathbf{A}_t\widehat{\mathbf{u}}_t + \widehat{\mathbf{u}}_t^{\mathsf{T}}(\mathbf{A}_t - \mathbf{A}_{t-1})\widehat{\mathbf{u}}_t\,,\end{aligned} \tag{2.28}$$

where in the last equality we used that by definition $\mathbf{A}_t - \mathbf{A}_{t-1} = \mathbf{x}_t\mathbf{x}_t^{\mathsf{T}}$.

Putting (2.26) and (2.28) together, we proved

$$\begin{aligned}&(y_t - \widehat{\mathbf{y}}_t)^2 + L_{t-1}^{\text{reg}}(\breve{\mathbf{u}}_t) - L_t^{\text{reg}}(\breve{\mathbf{u}}_{t+1}) \\ &= -2(\breve{\mathbf{u}}_{t+1} - \widehat{\mathbf{u}}_t)^{\mathsf{T}}\mathbf{A}_t\widehat{\mathbf{u}}_t + \widehat{\mathbf{u}}_t^{\mathsf{T}}(\mathbf{A}_t - \mathbf{A}_{t-1})\widehat{\mathbf{u}}_t + \breve{\mathbf{u}}_{t+1}^{\mathsf{T}}\mathbf{A}_t\breve{\mathbf{u}}_{t+1} - \breve{\mathbf{u}}_t^{\mathsf{T}}\mathbf{A}_{t-1}\breve{\mathbf{u}}_t \\ &= \breve{\mathbf{u}}_{t+1}^{\mathsf{T}}\mathbf{A}_t\breve{\mathbf{u}}_{t+1} - 2\breve{\mathbf{u}}_{t+1}^{\mathsf{T}}\mathbf{A}_t\widehat{\mathbf{u}}_t + \widehat{\mathbf{u}}_t^{\mathsf{T}}\mathbf{A}_t\widehat{\mathbf{u}}_t - \Big(\widehat{\mathbf{u}}_t^{\mathsf{T}}\mathbf{A}_{t-1}\widehat{\mathbf{u}}_t - 2\widehat{\mathbf{u}}_t^{\mathsf{T}}\underbrace{\mathbf{A}_t\widehat{\mathbf{u}}_t}_{=\mathbf{A}_{t-1}\breve{\mathbf{u}}_t} + \breve{\mathbf{u}}_t^{\mathsf{T}}\mathbf{A}_{t-1}\breve{\mathbf{u}}_t\Big)\,.\end{aligned}$$

In the last equation, we are about to use the equality $\mathbf{A}_t\widehat{\mathbf{u}}_t = \mathbf{A}_{t-1}\breve{\mathbf{u}}_t = \mathbf{b}_{t-1}$, which we get from the closed-form expressions of $\widehat{\mathbf{u}}_t$ and $\breve{\mathbf{u}}_t$. We then recognize the desired difference between two quadratic forms:

$$\begin{aligned}&(y_t - \widehat{\mathbf{y}}_t)^2 + L_{t-1}^{\text{reg}}(\breve{\mathbf{u}}_t) - L_t^{\text{reg}}(\breve{\mathbf{u}}_{t+1}) \\ &= \Big(\breve{\mathbf{u}}_{t+1}^{\mathsf{T}}\mathbf{A}_t\breve{\mathbf{u}}_{t+1} - 2\breve{\mathbf{u}}_{t+1}^{\mathsf{T}}\mathbf{A}_t\widehat{\mathbf{u}}_t + \widehat{\mathbf{u}}_t^{\mathsf{T}}\mathbf{A}_t\widehat{\mathbf{u}}_t\Big) - \Big(\widehat{\mathbf{u}}_t^{\mathsf{T}}\mathbf{A}_{t-1}\widehat{\mathbf{u}}_t - 2\widehat{\mathbf{u}}_t^{\mathsf{T}}\mathbf{A}_{t-1}\breve{\mathbf{u}}_t + \breve{\mathbf{u}}_t^{\mathsf{T}}\mathbf{A}_{t-1}\breve{\mathbf{u}}_t\Big) \\ &= (\breve{\mathbf{u}}_{t+1} - \widehat{\mathbf{u}}_t)^{\mathsf{T}}\mathbf{A}_t(\breve{\mathbf{u}}_{t+1} - \widehat{\mathbf{u}}_t) - (\widehat{\mathbf{u}}_t - \breve{\mathbf{u}}_t)^{\mathsf{T}}\mathbf{A}_{t-1}(\widehat{\mathbf{u}}_t - \breve{\mathbf{u}}_t)\,.\end{aligned}$$

Proof of the second inequality in (2.24). Because $y_t^2 \leqslant B^2$, we only need to prove

$$\sum_{t=1}^{T}\mathbf{x}_t^{\mathsf{T}}\mathbf{A}_t^{-1}\mathbf{x}_t \leqslant \sum_{k=1}^{d}\ln\left(1 + \frac{\lambda_k(\mathbf{G}_T)}{\lambda}\right)\,.$$

Now, Lemma 2.8 below shows that

$$\sum_{t=1}^{T}\mathbf{x}_t^{\mathsf{T}}\mathbf{A}_t^{-1}\mathbf{x}_t = \sum_{t=1}^{T}\left(1 - \frac{\det(\mathbf{A}_{t-1})}{\det(\mathbf{A}_t)}\right)\,.$$

We then use $1 - u \leqslant -\ln u$ for $u > 0$ and identify a telescoping sum,

$$\sum_{t=1}^{T} \left(1 - \frac{\det(\mathbf{A}_{t-1})}{\det(\mathbf{A}_t)}\right) \leqslant \sum_{t=1}^{T} \ln \frac{\det(\mathbf{A}_t)}{\det(\mathbf{A}_{t-1})} = \ln \frac{\det(\mathbf{A}_T)}{\det(\mathbf{A}_0)}\,.$$

All in all, we proved so far

$$\sum_{t=1}^{T} \mathbf{x}_t^{\mathrm{T}} \mathbf{A}_t^{-1} \mathbf{x}_t \leqslant \ln \frac{\det(\mathbf{A}_T)}{\det(\mathbf{A}_0)}\,,$$

and may conclude by noting that

$$\det(\mathbf{A}_T) = \det(\lambda\,\mathbf{I}_d + \mathbf{G}_T) = \prod_{k=1}^{d} (\lambda + \lambda_k(\mathbf{G}_T)) \qquad \text{and} \qquad \det(\mathbf{A}_0) = \det(\lambda\,\mathbf{I}_d) = \lambda^d\,.$$

□

Lemma 2.8. *Let V an arbitrary $d \times d$ full-rank matrix, let $\mathbf{u}$ and $\mathbf{v}$ two arbitrary vectors of $\mathbb{R}^d$, and let $\mathbf{U} = \mathbf{V} - \mathbf{u}\mathbf{v}^{\mathrm{T}}$. Then*

$$\mathbf{v}^{\mathrm{T}} \mathbf{V}^{-1} \mathbf{u} = 1 - \frac{\det(\mathbf{U})}{\det(\mathbf{V})}\,.$$

Proof. If $V = \mathbf{I}_d$, we are left to show that $\det(\mathbf{I}_d - \mathbf{u}\mathbf{v}^{\mathrm{T}}) = 1 - \mathbf{v}^{\mathrm{T}}\mathbf{u}$. The result follows from taking the determinant of every term of the equality

$$\begin{bmatrix} \mathbf{I}_d & 0 \\ \mathbf{v}^{\mathrm{T}} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_d - \mathbf{u}\mathbf{v}^{\mathrm{T}} & -\mathbf{u} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_d & 0 \\ -\mathbf{v}^{\mathrm{T}} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{I}_d & -\mathbf{u} \\ 0 & 1 - \mathbf{v}^{\mathrm{T}}\mathbf{u} \end{bmatrix}.$$

Now, we can reduce the case of a general $\mathbf{V}$ to this simpler case by noting that

$$\det(\mathbf{U}) = \det\Big(\mathbf{V} - \mathbf{u}\mathbf{v}^{\mathrm{T}}\Big) = \det(\mathbf{V}) \det\Big(\mathbf{I}_d - \Big(\mathbf{V}^{-1}\mathbf{u}\Big)\mathbf{v}^{\mathrm{T}}\Big) = \det(\mathbf{V})\Big(1 - \mathbf{v}^{\mathrm{T}}\mathbf{V}^{-1}\mathbf{u}\Big)\,.$$

□

2.7 Technical complements to Section 2.3

In this section we provide some additional discussions to those of Remark 3 (Section 2.7.1) and also extend the proof of Theorem 2.5 to work in the general case (Section 2.7.2).

2.7.1 Complements to Remark 3

We detail here why the derivation of a closed-form bound as led by Bartlett et al. [2015] only entails a bound of the order of $2dB^2 \ln T$ and why it cannot easily be improved.

Indeed, Theorem 5 by Bartlett et al. [2015] indicates, in the case where $d = 1$ and $B = 1$, that

$$\forall\, T \geqslant 1, \qquad \mathcal{R}_T^\star \leqslant f(T) \tag{2.29}$$

for any function $f : \{1, 2, \ldots\} \to \mathbb{R}_+$ satisfying $e^{-f(T)/2} \leqslant f(T+1) - f(T)$ for all $T \geqslant 1$. As they showed, the function $f(T) = 2\ln(1 + T/2) + 1$ is a suitable choice, but it leads to the extra multiplicative factor of 2 that we pointed out.

However, this choice for f does not seem to be easily improvable; for instance, functions f of the form $T \mapsto a \ln T + b$ for some $a < 2$ and $b \in \mathbb{R}$ are such that

$$e^{-f(T)/2} = \Theta_T\Big(T^{-a/2}\Big) \qquad \text{and} \qquad f(T+1) - f(T) = a \ln\bigg(1 + \frac{1}{T}\bigg) = \mathcal{O}_T\Big(T^{-1}\Big)\,,$$

hence, are not suitable choices for the bound (2.29).

2.7.2 Proof of Theorem 2.5 in the general case

In this section we extend the proof of Theorem 2.5, provided only in the case of a full-rank Gram matrix G_T in Section 2.3, to the general case of a possibly non-invertible Gram matrix G_T.

To that end, we first explain how the closed-form expression (2.7) is derived. We rewrite the definition equation (2.6) of $\widehat{\mathbf{u}}_t$ as

$$\widehat{\mathbf{u}}_t \in \operatorname*{argmin}_{\mathbf{u}\in\mathbb{R}^d}\Big\{\mathbf{u}^{\mathrm{T}}(\lambda\,\mathbf{G}_T + \mathbf{G}_t)\mathbf{u} - 2\mathbf{b}_{t-1}^{\mathrm{T}}\mathbf{u}\Big\}\,.$$

Because the matrix $\lambda\,\mathbf{G}_T + \mathbf{G}_t$ is positive semidefinite, the considered argmin is also the set of values $\mathbf{u}'$ where the gradient vanishes: $(\lambda\,\mathbf{G}_T + \mathbf{G}_t)\mathbf{u}' = \mathbf{b}_{t-1}$. This system is possibly under-defined because $\mathbf{u}' \in \mathbb{R}^d$ and $\lambda\,\mathbf{G}_T + \mathbf{G}_t$ is a matrix of size $\mathrm{d} \times \mathrm{d}$, possibly not full rank. The system has at least one solution but the one with minimal Euclidean norm is given by the Moore-Penrose inverse, see Corollary 2.12 (e):

$$\widehat{\mathbf{u}}_t = \Big(\lambda\,\mathbf{G}_T + \mathbf{G}_t\Big)^{\dagger}\mathbf{b}_{t-1}\,.$$

We may now turn to the general proof of Theorem 2.5. For an integer $k \geqslant 1$, we denote therein by $\mathbf{I}_k$ the $k \times k$ identity matrix.

Proof. As a consequence of the spectral theorem applied to the symmetric matrix $\mathbf{G}_T$, there exists a matrix $\mathbf{U}$ of size $d \times r_T$ and a full rank square matrix $\mathbf{\Sigma}$ of size $r_T \times r_T$ such that $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}_{r_T}$ and $\mathbf{G}_T = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{\mathrm{T}}$. We could even impose that the matrix $\mathbf{\Sigma}$ be diagonal but this property will not be used in this proof.

We will apply the (already proven) bound of Theorem 2.5 in the full rank case. To that end, we consider the modified sequence of features

$$\widetilde{\mathbf{x}}_t = \mathbf{U}^{\mathrm{T}}\mathbf{x}_t$$

and first prove that the strategy (2.6) on the $\widetilde{\mathbf{x}}_t$ leads to the same forecasts as the same strategy on the original features $\mathbf{x}_t$; that is,

$$\widetilde{\mathbf{u}}_t \cdot \widetilde{\mathbf{x}}_t = \widehat{\mathbf{u}}_t \cdot \mathbf{x}_t\,, \quad \text{where} \quad \widetilde{\mathbf{u}}_t \in \operatorname*{argmin}_{\mathbf{v}\in\mathbb{R}^{r_T}}\bigg\{\sum_{s=1}^{t-1}(y_s - \mathbf{v}\cdot\widetilde{\mathbf{x}}_s)^2 + (\mathbf{v}\cdot\widetilde{\mathbf{x}}_t)^2 + \lambda\sum_{s=1}^{T}(\mathbf{v}\cdot\widetilde{\mathbf{x}}_s)^2\bigg\}\,.$$

It suffices to prove $\mathbf{U}\widetilde{\mathbf{u}}_t = \widehat{\mathbf{u}}_t$, which we do below. Then, from this equality and the definition $\widetilde{\mathbf{x}}_t = \mathbf{U}^{\mathrm{T}}\mathbf{x}_t$, we have, as desired,

$$\widetilde{\mathbf{u}}_t \cdot \widetilde{\mathbf{x}}_t = \widetilde{\mathbf{u}}_t \cdot \Big(\mathbf{U}^{\mathrm{T}}\mathbf{x}_t\Big) = \Big(\mathbf{U}\widetilde{\mathbf{u}}_t\Big) \cdot \mathbf{x}_t = \widehat{\mathbf{u}}_t \cdot \mathbf{x}_t \,.$$

Now, to prove $\mathbf{U}\widetilde{\mathbf{u}}_t = \widehat{\mathbf{u}}_t$, we resort to the closed-form expression (2.7), which gives that

$$\mathbf{U}\widetilde{\mathbf{u}}_t = \mathbf{U}\left(\lambda \sum_{s=1}^{T} \widetilde{\mathbf{x}}_s\widetilde{\mathbf{x}}_s^{\mathrm{T}} + \sum_{s=1}^{t} \widetilde{\mathbf{x}}_s\widetilde{\mathbf{x}}_s^{\mathrm{T}}\right)^{\dagger} \sum_{s=1}^{t-1} y_s\widetilde{\mathbf{x}}_s = \mathbf{U}\Big(\mathbf{U}^{\mathrm{T}}(\lambda\mathbf{G}_T + \mathbf{G}_t)\mathbf{U}\Big)^{\dagger}\mathbf{U}^{\mathrm{T}}\mathbf{b}_{t-1} \,.$$

To simplify this expression, we use twice the property of Moore-Penrose pseudoinverses stated in Corollary 2.12 (b), once with $\mathbf{M} = \mathbf{U}$ and the second time with $\mathbf{N} = \mathbf{U}^{\mathrm{T}}$, which both satisfy the required condition for Corollary 2.12 (b), as well as the matrix equalities in Corollary 2.12 (c), and we get

$$\mathbf{U}\Big(\mathbf{U}^{\mathrm{T}}(\lambda\mathbf{G}_T + \mathbf{G}_t)\mathbf{U}\Big)^{\dagger}\mathbf{U}^{\mathrm{T}} = \Big(\mathbf{U}\mathbf{U}^{\mathrm{T}}(\lambda\mathbf{G}_T + \mathbf{G}_t)\mathbf{U}\mathbf{U}^{\mathrm{T}}\Big)^{\dagger} = (\lambda\mathbf{G}_T + \mathbf{G}_t)^{\dagger} \,,$$

where the last equality comes from

$$\mathbf{U}\mathbf{U}^{\mathrm{T}}(\lambda\mathbf{G}_T + \mathbf{G}_t)\mathbf{U}\mathbf{U}^{\mathrm{T}} = \lambda\mathbf{G}_T + \mathbf{G}_t \,. \tag{2.30}$$

Indeed, from $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}_{r_T}$ we get $\mathbf{U}\mathbf{U}^{\mathrm{T}} = P_{\mathrm{Im}(\mathbf{G}_T)}$, the orthogonal projector on the image of $\mathbf{G}_T$; we recall in (2.39) why $\mathrm{Im}(\mathbf{G}_t) \subseteq \mathrm{Im}(\mathbf{G}_T)$, which implies $\mathbf{U}\mathbf{U}^{\mathrm{T}}(\lambda\mathbf{G}_T + \mathbf{G}_t) = \lambda\mathbf{G}_T + \mathbf{G}_t$. Transposing this leads to $(\lambda\mathbf{G}_T + \mathbf{G}_t)\mathbf{U}\mathbf{U}^{\mathrm{T}} = \lambda\mathbf{G}_T + \mathbf{G}_t$, from which the desired equality (2.30) follows by a left multiplication again by $\mathbf{U}\mathbf{U}^{\mathrm{T}} = P_{\mathrm{Im}(\mathbf{G}_T)}$.

We may now apply the bound of the Theorem 2.5 in the full rank case on feature sequences $\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_T \in \mathbb{R}^{r_T}$ and observations $y_1, \ldots, y_T \in [-B, B]$; this is because the associated Gram matrix $\mathbf{U}^{\mathrm{T}}\mathbf{G}_T\mathbf{U} = \mathbf{\Sigma}$ is now full rank. We get, for all $\mathbf{v} \in \mathbb{R}^{r_T}$,

$$\sum_{t=1}^{T}(y_t - \widehat{\mathbf{u}}_t\cdot\mathbf{x}_t)^2 = \sum_{t=1}^{T}(y_t - \widetilde{\mathbf{u}}_t\cdot\widetilde{\mathbf{x}}_t)^2 \leqslant \sum_{t=1}^{T}(y_t - \mathbf{v}\cdot\widetilde{\mathbf{x}}_t)^2 + \lambda T B^2 + r_T B^2 \ln\Big(1 + \frac{1}{\lambda}\Big) \,. \tag{2.31}$$

To conclude the proof, its only remains to show that

$$\inf_{\mathbf{v}\in\mathbb{R}^{r_T}} \sum_{t=1}^{T}(y_t - \mathbf{v}\cdot\widetilde{\mathbf{x}}_t)^2 = \inf_{\mathbf{u}\in\mathbb{R}^d} \sum_{t=1}^{T}(y_t - \mathbf{u}\cdot\mathbf{x}_t)^2 \,. \tag{2.32}$$

Now, a basic argument of linear algebra, recalled in (2.38) of Appendix 2.A, indicates $\mathrm{Im}(\mathbf{G}_t) = \mathrm{Im}(\mathbf{X}_t)$. Together with the inclusion $\mathrm{Im}(\mathbf{G}_t) \subseteq \mathrm{Im}(\mathbf{G}_T)$ and the fact that $\mathbf{U}\mathbf{U}^{\mathrm{T}} = P_{\mathrm{Im}(\mathbf{G}_T)}$, both already used above, we get $\mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{x}_t = \mathbf{x}_t$. A direct consequence is that for any $\mathbf{u}$ in $\mathbb{R}^d$,

$$\mathbf{u}\cdot\mathbf{x}_t = \mathbf{u}\cdot\Big(\mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{x}_t\Big) = \Big(\mathbf{U}^{\mathrm{T}}\mathbf{u}\Big)\cdot\Big(\mathbf{U}^{\mathrm{T}}\mathbf{x}_t\Big) = \Big(\mathbf{U}^{\mathrm{T}}\mathbf{u}\Big)\cdot\widetilde{\mathbf{x}}_t \,,$$

from which (2.32) follows, by considering $\mathbf{v} = \mathbf{U}^{\mathrm{T}}\mathbf{u}$ and by the surjectivity of $\mathbf{U}^{\mathrm{T}}$ onto $\mathbb{R}^{r_T}$ (recall that $\mathbf{U}$ and $\mathbf{U}^{\mathrm{T}}$ are of rank r_T). □

2.8 Proof of Theorem 2.6

We recall in Appendix 2.A many basic properties of Gram matrices and Moore-Penrose pseudoinverses to be used in the proof below.

Proof. We successively prove the following two inequalities,

$$\mathcal{R}_T(\mathbf{u}) \leqslant \sum_{t=1}^{T} y_t^2 \mathbf{x}_t^{\mathrm{T}} \mathbf{G}_t^{\dagger} \mathbf{x}_t \leqslant B^2 \sum_{k=1}^{r_T} \ln\Bigl(\lambda_k(\mathbf{G}_T)\Bigr) + B^2 \sum_{t \in [\![1,T]\!] \cap \mathcal{T}} \ln\left(\frac{1}{\lambda_{r_t}(\mathbf{G}_t)}\right) + r_T B^2 \,, \tag{2.33}$$

where actually, the first inequality is a classic inequality already proved by Forster and Warmuth [2003, Theorem 3.2]. We provide its derivation for the sake of completeness only.

Proof of the first inequality in (2.33). We obtain it as a limit case. To do so, we start by exactly rewriting the first inequality of (2.24), where a $\lambda > 0$ regularization factor was considered:

$$\sum_{t=1}^{T} \Bigl(y_t - \mathbf{x}_t^{\mathrm{T}} (\lambda \mathbf{I}_d + \mathbf{G}_t)^{-1} \mathbf{b}_{t-1}\Bigr)^2 - \sum_{t=1}^{T} (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 \leqslant \sum_{t=1}^{T} y_t^2 \, \mathbf{x}_t^{\mathrm{T}} (\lambda \mathbf{I}_d + \mathbf{G}_t)^{-1} \mathbf{x}_t + \lambda \left\|\mathbf{u}\right\|^2 . \tag{2.34}$$

Since

$$\mathbf{G}_t = \mathbf{X}_t \mathbf{X}_t^{\mathrm{T}} \qquad \text{where} \qquad \mathbf{X}_t = \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_t \end{bmatrix}$$

we note that $\mathbf{x}_t^{\mathrm{T}} (\lambda \mathbf{I}_d + \mathbf{G}_t)^{-1}$ is the last line of the matrix $\mathbf{X}_t^{\mathrm{T}} \Bigl(\lambda \mathbf{I}_d + \mathbf{X}_t \mathbf{X}_t^{\mathrm{T}}\Bigr)^{-1}$, which tends to $\mathbf{X}^{\dagger}$ when $\lambda \to 0$ as indicated by Corollary 2.12 (d). Now, $\mathbf{X}^{\dagger} = \mathbf{X}_t^{\mathrm{T}} \Bigl(\mathbf{X}_t \mathbf{X}_t^{\mathrm{T}}\Bigr)^{\dagger} = \mathbf{X}_t^{\mathrm{T}} \mathbf{G}_t^{\dagger}$ by Corollary 2.12 (a), thus

$$\lim_{\lambda \to 0} \mathbf{x}_t^{\mathrm{T}} (\lambda \mathbf{I}_d + \mathbf{G}_t)^{-1} = \mathbf{x}_t^{\mathrm{T}} \mathbf{G}_t^{\dagger} \,.$$

Therefore, the desired inequality for the considered forecaster,

$$\mathcal{R}_T(\mathbf{u}) = \sum_{t=1}^{T} \Bigl(y_t - \mathbf{x}_t^{\mathrm{T}} \mathbf{G}_t^{\dagger} \mathbf{b}_{t-1}\Bigr)^2 - \sum_{t=1}^{T} (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 \leqslant \sum_{t=1}^{T} y_t^2 \mathbf{x}_t^{\mathrm{T}} \mathbf{G}_t^{\dagger} \mathbf{x}_t \,,$$

is obtained by taking the limit $\lambda \to 0$ in (2.34).

Proof of the second inequality in (2.33). The first part of our derivation is similar to what is performed in Luo et al. [2016, Theorem 4 of Appendix D], while the second part slightly improves on their result thanks to a more careful analysis using however the same ingredients.

Because $y_t^2 \leqslant B^2$, we only need to prove

$$\sum_{t=1}^{T} \mathbf{x}_t^{\mathrm{T}} \mathbf{G}_t^{\dagger} \mathbf{x}_t \leqslant \sum_{k=1}^{r_T} \ln\Bigl(\lambda_k(\mathbf{G}_T)\Bigr) + \sum_{t \in \mathcal{T} \cap [\![1,T]\!]} \ln\left(\frac{1}{\lambda_{r_t}(\mathbf{G}_t)}\right) + r_T \,.$$

Now, Lemma 2.9 below shows that

$$\sum_{t=1}^{T} \mathbf{x}_t^{\mathrm{T}} \mathbf{G}_t^{\dagger} \mathbf{x}_t = \sum_{t=1}^{T} \left(1 - \prod_{k=1}^{r_t} \frac{\lambda_k(\mathbf{G}_{t-1})}{\lambda_k(\mathbf{G}_t)}\right) ;$$

we assumed with no loss of generality that $\mathbf{x}_1$ is not the null vector, hence all $\mathbf{G}_t$ are at least of rank 1. Indeed, when $\mathbf{x}_t$ is the null vector, all linear combinations result in the same prediction equal to 0 and incur the same instantaneous quadratic loss.

Now, given the definition of the set $\mathcal{T}$, whose cardinality is r_T, we have $\lambda_{r_t}(\mathbf{G}_{t-1}) = 0$ when $t \in \mathcal{T}$ (and this includes $t = 1$, with the convention that $\mathbf{G}_0$ is the null matrix), while $r_{t-1} = r_t$ if $t \notin \mathcal{T}$. Therefore,

$$\begin{aligned}\sum_{t=1}^{T} \mathbf{x}_t^{\mathrm{T}} \mathbf{G}_t^{\dagger} \mathbf{x}_t &\leqslant \sum_{t \in \mathcal{T} \cap [\![1,T]\!]} \left(1 - \prod_{k=1}^{r_t} \frac{\lambda_k(\mathbf{G}_{t-1})}{\lambda_k(\mathbf{G}_t)}\right) + \sum_{t \in [\![1,T]\!] \setminus \mathcal{T}} \left(1 - \prod_{k=1}^{r_t} \frac{\lambda_k(\mathbf{G}_{t-1})}{\lambda_k(\mathbf{G}_t)}\right) \\ &= r_T + \sum_{t \in [\![1,T]\!] \setminus \mathcal{T}} \left(1 - \frac{D_{t-1}}{D_t}\right),\end{aligned}$$

where $D_t = \prod_{k=1}^{r_t} \lambda_k(\mathbf{G}_t)$ is the product of the positive eigenvalues of $\mathbf{G}_t$.

Now (this is where our analysis is more careful), using $1 - u \leqslant -\ln u$ for $u > 0$, we get an almost telescoping sum,

$$\sum_{t \in [\![1,T]\!] \setminus \mathcal{T}} \left(1 - \frac{D_{t-1}}{D_t}\right) \leqslant \sum_{t \in [\![1,T]\!] \setminus \mathcal{T}} \ln \frac{D_t}{D_{t-1}} = \ln \frac{D_T}{D_1} + \sum_{t \in \mathcal{T} \cap [\![2,T]\!]} \ln \frac{D_{t-1}}{D_t}$$

(note that we dealt separately with $t = 1$, which belongs to $\mathcal{T}$). Because eigenvalues cannot decrease with t, see (2.40), we have in particular $\lambda_k(\mathbf{G}_{t-1}) \leqslant \lambda_k(\mathbf{G}_t)$ for all $1 \leqslant k \leqslant r_t - 1$. Thus, for $t \in \mathcal{T}$ with $t \neq 1$, we have

$$\ln \frac{D_{t-1}}{D_t} \leqslant \ln\left(\frac{1}{\lambda_{r_t}(\mathbf{G}_t)}\right),$$

Substituting the definition of D_T and the equality $D_1 = \lambda_{r_1}(\mathbf{G}_1)$, and collecting all bounds together leads to the second inequality in (2.33). □

The lemma below was essentially stated and proved by Cesa-Bianchi et al. [2005, Lemma D.1].

Lemma 2.9 (Rewriting of $\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\dagger}\mathbf{x}$)**.** *Let $\mathbf{B}$ be a $d \times d$ symmetric positive semidefinite matrix (possibly the null matrix), let $\mathbf{x} \in \mathbb{R}^d$, and and let $\mathbf{A} = \mathbf{B} + \mathbf{x}\mathbf{x}^{\mathrm{T}}$. Denote by r the rank of $\mathbf{A}$ and assume that $r \geqslant 1$. Then*

$$\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\dagger}\mathbf{x} = 1 - \prod_{k=1}^{r} \frac{\lambda_k(\mathbf{B})}{\lambda_k(\mathbf{A})}. \tag{2.35}$$

Proof. This lemma is a consequence of the less general Lemma 2.8. As a consequence of the spectral theorem applied to the symmetric matrix $\mathbf{A}$, there exists a matrix $\mathbf{U}$ of size $d \times r$ and a full rank square matrix $\mathbf{\Sigma}$ of size $r \times r$ such that $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}_r$ and $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{\mathrm{T}}$. We can and will even impose that the matrix $\mathbf{\Sigma}$ is diagonal, with diagonal values equal to $\lambda_1(\mathbf{A}), \ldots, \lambda_r(\mathbf{A})$, the positive eigenvalues of $\mathbf{A}$. Let $\mathbf{\Gamma} = \mathbf{\Sigma} - \mathbf{U}^{\mathrm{T}}\mathbf{x}(\mathbf{U}^{\mathrm{T}}\mathbf{x})^{\mathrm{T}}$.

Lemma 2.8 with $\mathbf{\Gamma}$, $\mathbf{\Sigma}$ and $\mathbf{U}^\mathsf{T}\mathbf{x}$ indicates that

$$\mathbf{x}^\mathsf{T}\Big(\mathbf{U}\mathbf{\Sigma}^{-1}\mathbf{U}^\mathsf{T}\Big)\mathbf{x} = \Big(\mathbf{U}^\mathsf{T}\mathbf{x}\Big)^\mathsf{T}\mathbf{\Sigma}^{-1}\Big(\mathbf{U}^\mathsf{T}\mathbf{x}\Big) = 1 - \frac{\det(\mathbf{\Gamma})}{\det(\mathbf{\Sigma})} \qquad \text{where} \qquad \det(\mathbf{\Sigma}) = \prod_{k=1}^{r} \lambda_k(\mathbf{A})\,.$$

Now, it can be easily checked (by noting that all four properties in Proposition 2.11 are satisfied) that $\mathbf{A}^\dagger = \mathbf{U}\mathbf{\Sigma}^{-1}\mathbf{U}^\mathsf{T}$, so that from the above equality, it suffices to show that

$$\det(\mathbf{\Gamma}) = \prod_{k=1}^{r} \lambda_k(\mathbf{B})$$

to conclude the proof. To do so, we first remark that $\mathbf{B} = \mathbf{A} - \mathbf{x}\mathbf{x}^\mathsf{T} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\mathsf{T} - \mathbf{x}\mathbf{x}^\mathsf{T}$, which yields

$$\mathbf{U}^\mathsf{T}\mathbf{B}\mathbf{U} = \mathbf{U}^\mathsf{T}\mathbf{U}\mathbf{\Sigma}\mathbf{U}^\mathsf{T}\mathbf{U} - \mathbf{U}^\mathsf{T}\mathbf{x}\mathbf{x}^\mathsf{T}\mathbf{U} = \mathbf{\Sigma} - \mathbf{U}^\mathsf{T}\mathbf{x}\mathbf{x}^\mathsf{T}\mathbf{U} = \mathbf{\Gamma}\,.$$

Using again that $\mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{I}_r$, we note that $\mathbf{u}^\mathsf{T}\mathbf{u} = (\mathbf{U}\mathbf{u})^\mathsf{T}\mathbf{U}\mathbf{u}$ for all $\mathbf{u} \in \mathbb{R}^r$. From this and $\mathbf{U}^\mathsf{T}\mathbf{B}\mathbf{U} = \Gamma$, we get in particular

$$\sup_{0\neq\mathbf{u}\in\mathbb{R}^r} \frac{\mathbf{u}^\mathsf{T}\mathbf{\Gamma}\mathbf{u}}{\mathbf{u}^\mathsf{T}\mathbf{u}} = \sup_{0\neq\mathbf{u}\in\mathbb{R}^r} \frac{(\mathbf{U}\mathbf{u})^\mathsf{T}\mathbf{B}(\mathbf{U}\mathbf{u})}{(\mathbf{U}\mathbf{u})^\mathsf{T}\mathbf{U}\mathbf{u}}\,. \tag{2.36}$$

Next we show that

$$\sup_{0\neq\mathbf{u}\in\mathbb{R}^r} \frac{(\mathbf{U}\mathbf{u})^\mathsf{T}\mathbf{B}(\mathbf{U}\mathbf{u})}{(\mathbf{U}\mathbf{u})^\mathsf{T}\mathbf{U}\mathbf{u}} = \sup_{0\neq\mathbf{v}\in\mathbb{R}^d} \frac{\mathbf{v}^\mathsf{T}\mathbf{B}(\mathbf{v})}{\mathbf{v}^\mathsf{T}\mathbf{v}}\,, \tag{2.37}$$

which indicates, together with (2.36) and the characterization (2.41) of the eigenvalues of symmetric positive semidefinite matrices, that $\mathbf{B}$ and $\mathbf{\Gamma}$ have the same top r eigenvalues, as claimed. Now, to show (2.37), we recall that $\mathbb{R}^d = \ker(\mathbf{B}) \oplus \mathrm{Im}(\mathbf{B})$ for any symmetric matrix $\mathbf{B}$, so that,

$$\sup_{0\neq\mathbf{v}\in\mathbb{R}^d} \frac{\mathbf{v}^\mathsf{T}\mathbf{B}(\mathbf{v})}{\mathbf{v}^\mathsf{T}\mathbf{v}} = \sup_{0\neq\mathbf{v}\in\mathrm{Im}(\mathbf{B})} \frac{\mathbf{v}^\mathsf{T}\mathbf{B}(\mathbf{v})}{\mathbf{v}^\mathsf{T}\mathbf{v}}\,.$$

This leads to (2.37) via the inclusions

$$\mathrm{Im}(\mathbf{B}) \subseteq \mathrm{Im}(\mathbf{U}) \subseteq \mathbb{R}^d$$

which themselves follow from the inclusions

$$\mathrm{Im}(\mathbf{B}) \subseteq \mathrm{Im}(\mathbf{A}) \subseteq \mathrm{Im}(\mathbf{U})\,.$$

Indeed, $\mathrm{Im}(\mathbf{A}) \subseteq \mathrm{Im}(\mathbf{U})$ because $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\mathsf{T}$ and $\mathrm{Im}(\mathbf{B}) \subseteq \mathrm{Im}(\mathbf{A})$, or equivalently, given that we are considering symmetric matrices, $\ker\mathbf{A} \subseteq \ker\mathbf{B}$, as for all $\mathbf{y} \in \mathbb{R}^d$,

$$\mathbf{A}\mathbf{y} = 0 \implies \mathbf{y}^\mathsf{T}\mathbf{A}\mathbf{y} = 0 \implies \Big[\mathbf{y}^\mathsf{T}\mathbf{B}\mathbf{y} = 0 \text{ and } \mathbf{y}^\mathsf{T}\mathbf{x}\mathbf{x}^\mathsf{T}\mathbf{y} = 0\Big] \implies \sqrt{\mathbf{B}}\mathbf{y} = 0 \implies \mathbf{B}\mathbf{y} = 0\,,$$

where we used $\mathbf{A} = \mathbf{B} + \mathbf{x}\mathbf{x}^\mathsf{T}$ to get the second implication, and where we multiplied $\sqrt{\mathbf{B}}\mathbf{y}$ by $\sqrt{\mathbf{B}}$ to get the final implication. □

Appendices

2.A Some basic facts of linear algebra

We gather in this appendix some useful results of linear algebra, that are either reminder of well-known facts or are easy to prove (yet, we prefer prove them here rather for the proofs above to be more focused).

2.A.1 Gram matrices versus matrices of features

Recall that we denoted by

$$\mathbf{X}_t = \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_t \end{bmatrix}$$

the $d \times t$ matrix consisting the first t features. By definition,

$$\mathrm{Im}(\mathbf{X}_t) = \mathrm{span}\{\mathbf{x}_1, \ldots, \mathbf{x}_t\} \qquad \text{and} \qquad \mathbf{G}_t = \mathbf{X}_t \mathbf{X}_t^{\mathrm{T}} .$$

The aim of this section is to show that, for all $t \geqslant 1$,

$$\mathrm{Im}(\mathbf{G}_t) = \mathrm{Im}(\mathbf{X}_t) . \tag{2.38}$$

which in turn implies that for all $t \geqslant 2$,

$$\mathrm{Im}(\mathbf{G}_{t-1}) \subseteq \mathrm{Im}(\mathbf{G}_t) , \tag{2.39}$$

and that $\mathrm{rank}(\mathbf{G}_{t-1})$ and $\mathrm{rank}(\mathbf{G}_t)$ differ from at most 1.

First, as for any (not necessarily square) matrix $\mathbf{M}$ we have $\mathrm{Im}(\mathbf{M}) = \ker\Big(\mathbf{M}^{\mathrm{T}}\Big)^{\perp}$, we note that (2.38) is equivalent to $\ker\Big(\mathbf{G}_t\Big)^{\perp} = \ker\Big(\mathbf{X}_t^{\mathrm{T}}\Big)^{\perp}$, thus to $\ker\Big(\mathbf{G}_t\Big) = \ker\Big(\mathbf{X}_t^{\mathrm{T}}\Big)$. It is clear by definition of $\mathbf{G}_t$ that $\ker\Big(\mathbf{X}_t^{\mathrm{T}}\Big) \subseteq \ker\Big(\mathbf{G}_t\Big)$; furthermore, for any vector $\mathbf{u} \in \mathbb{R}^d$, we have the equality $\mathbf{u}^{\mathrm{T}}\mathbf{G}_t\mathbf{u} = \|\mathbf{X}_t^{\mathrm{T}}\mathbf{u}\|^2$, which yields the opposite inclusion $\ker\Big(\mathbf{G}_t\Big) \subseteq \ker\Big(\mathbf{X}_t^{\mathrm{T}}\Big)$.

The inclusion (2.39) follows from (2.38) as by definition, the image of $\mathbf{X}_t$ is generated by the image of $\mathbf{X}_{t-1}$ and $\mathbf{x}_t$.

2.A.2 Dynamic of the eigenvalues of Gram matrices

The above result gives us an idea of how eigenspaces and eigenvalues of the covariance matrix evolve. Another relationship is the following one: for $t \geqslant 1$,

$$\lambda_k\Big(\mathbf{G}_{t-1}\Big) \leqslant \lambda_k\Big(\mathbf{G}_t\Big) , \tag{2.40}$$

where we recall that $\lambda_k\Big(\mathbf{G}_t\Big)$ denotes the k$^{\text{th}}$ eigenvalue of $\mathbf{G}_t$ in decreasing order. To prove this we remark that for all $\mathbf{u} \in \mathbb{R}^d$, we have

$$\mathbf{u}^{\mathrm{T}}\mathbf{G}_{t-1}\mathbf{u} \leqslant \mathbf{u}^{\mathrm{T}}\mathbf{x}_t\mathbf{u} + \mathbf{u}^{\mathrm{T}}\mathbf{G}_{t-1}\mathbf{u} = \mathbf{u}^{\mathrm{T}}\mathbf{G}_t\mathbf{u}$$

and use the fact that for all symmetric positive semidefinite matrices $\mathbf{M}$,

$$\lambda_k(\mathbf{M}) = \max\left\{ \min_{\mathbf{u}} \left\{ \frac{\mathbf{u}^{\mathrm{T}}\mathbf{M}\mathbf{u}}{\mathbf{u}^{\mathrm{T}}\mathbf{u}} \mid \mathbf{u} \in U \text{ and } \mathbf{u} \neq 0 \right\} \,\middle|\, U \text{ vector space with } \dim(U) = k \right\} \tag{2.41}$$

2.A.3 Moore-Penrose pseudoinverses: definition and basic properties

In this section, we recall the definition and some basic properties of the Moore-Penrose pseudoinverse. It was introduced by E.H. Moore in 1920 and is a generalization of the inverse operator for non-invertible (and non-square) matrices.

Definition 2.10 (Moore-Penrose pseudoinverse)**.** *The Moore-Penrose pseudoinverse of an $m \times n$ matrix $\mathbf{M}$ is a $n \times m$ matrix denoted by $\mathbf{M}^\dagger$ and defined as*

$$\mathbf{M}^\dagger \stackrel{\text{def}}{=} \lim_{\alpha \to 0} \left(\mathbf{M}^{\mathrm{T}}\mathbf{M} + \alpha I_n\right)^{-1}\mathbf{M}^{\mathrm{T}},$$

where $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ is the identity matrix and $\alpha \to 0$ while $\alpha > 0$.

We have the following characterization of $\mathbf{M}^\dagger$.

Proposition 2.11. *Let $\mathbf{M}$ be a $m \times n$ matrix. Its Moore-Penrose pseudoinverse $\mathbf{M}^\dagger$ is unique and is characterized as the only $n \times m$ matrix simultaneously satisfying the following four properties:*

(P1) $\mathbf{M}\mathbf{M}^\dagger\mathbf{M} = \mathbf{M}$
(P2) $\mathbf{M}^\dagger\mathbf{M}\mathbf{M}^\dagger = \mathbf{M}^\dagger$
(P3) $\left(\mathbf{M}\mathbf{M}^\dagger\right)^{\mathrm{T}} = \mathbf{M}\mathbf{M}^\dagger$
(P4) $\left(\mathbf{M}^\dagger\mathbf{M}\right)^{\mathrm{T}} = \mathbf{M}^\dagger\mathbf{M}$

The proof can be found in Penrose [1955]. In particular, in our analysis we use the following consequences of Proposition 2.11. (We leave the standard proofs to the reader.)

Corollary 2.12. *Let $\mathbf{M}$ be a $m \times n$ matrix and $\mathbf{N}$ a $n \times p$ matrix. Then,*

(a) $\mathbf{M}^\dagger = \mathbf{M}^{\mathrm{T}}\left(\mathbf{M}\mathbf{M}^{\mathrm{T}}\right)^\dagger$;
(b) if $\mathbf{M}^{\mathrm{T}}\mathbf{M} = \mathbf{I}_n$ *or* $\mathbf{N}\mathbf{N}^{\mathrm{T}} = \mathbf{I}_n$ *then* $\left(\mathbf{M}\mathbf{N}\right)^\dagger = \mathbf{N}^\dagger\mathbf{M}^\dagger$;
(c) if $\mathbf{M}^{\mathrm{T}}\mathbf{M} = \mathbf{I}_n$*, then* $\mathbf{M}^{\mathrm{T}} = \mathbf{M}^\dagger$ *and* $\mathbf{M} = \left(\mathbf{M}^{\mathrm{T}}\right)^\dagger$;
(d) $\mathbf{M}^\dagger = \lim_{\alpha \to 0} \mathbf{M}^{\mathrm{T}}\left(\lambda \mathbf{I}_m + \mathbf{M}\mathbf{M}^{\mathrm{T}}\right)^{-1}$;
(e) if the equation $\mathbf{M}\mathbf{x} = \mathbf{z}$ *with unknown* $\mathbf{z} \in \mathbb{R}^m$ *admits a solution* $\mathbf{x} \in \mathbb{R}^n$*, then* $\mathbf{M}^\dagger\mathbf{z}$ *is the solution in* $\mathbb{R}^n$ *with minimal Euclidean norm.*

CHAPTER 3

Online hierarchical forecasting for power consumption data

In this chapter we study the forecasting of the power consumptions of a population of households and of subpopulations thereof. These subpopulations are built according to location, to exogenous information and/or to profiles we determined from historical households consumption time series. Thus, we aim to forecast the electricity consumption time series at several levels of households aggregation. These time series are linked through some summation constraints which induce a hierarchy. Our approach consists in three steps: feature generation, aggregation and projection. Firstly (feature generation step), we build, for each considering group for households, a benchmark forecast (called features), using random forests or generalized additive models. Secondly (aggregation step), aggregation algorithms, run in parallel, aggregate these forecasts and provide new predictions. Finally (projection step), we use the summation constraints induced by the time series underlying hierarchy to re-conciliate the forecasts by projecting them in a well-chosen linear subspace. We provide some theoretical guaranties on the average prediction error of this methodology, through the minimization of a quantity called regret. We also test our approach on households power consumption data collected in Great Britain by multiple energy providers in the '*Energy Demand Research Project*' context. We build and compare various population segmentations for the evaluation of our approach performance.

Contents

This chapter is joint work with Margaux Brégère. It is currently under review for publication (see Brégère and Huard, 2020).

3.1 Introduction

Motivation: Electricity Forecasting. New opportunities come with the recent deployment of smart grids and the installation of meters: they record consumption quasi instantaneously in households. From these records, time series of demand are obtained at various levels of aggregation, such as consumption profiles and regions. For privacy reasons, household records may not be used directly. Moreover, consumption at individual level is erratic and unpredictable. This is why we focus on household aggregations. For demand management, it is useful to predict the global consumption. Furthermore, to dispatch correctly the electricity into the grid, forecasting demand at a regional level is also an important goal. Finally, a good estimation of the consumption of some groups of consumers (with the same profile) may be helpful for the electricity provider which may adapt its offer to perform effective demand side management. Thus, forecasts at various aggregated levels (entire population, geographical areas, groups of same consumption profiles) are useful for an efficient management of consumption. In this work, we first build at each aggregation level, and independently, benchmark forecasts (called features) using random forests or generalized additive models. Noticing that these time series may be correlated (the consumption of a given region may be close to the one of a neighboring region) and connected to each other through summation constraints (e.g., the global consumption is the sum of the region consumptions), the problem considered falls under the umbrella of hierarchical time series forecasting. Using these hierarchical relationships may improve the benchmark forecasts that were generated. Our approach consists in combining two methods: feature aggregation and and projection in a constrained space. Our aim is to improve forecasts both at the global and at the local levels.

Literature Discussion for Hierarchical Forecasting. Traditionally two types of methods have been used for hierarchical forecasting: bottom-up and top-down approaches. In the bottom-up approaches (see Dunn et al., 1976) forecasts are constructed for lower-level quantities and are then summed up to obtain forecasts at the upper levels. In contrast, top-down approaches (see Gross and Sohl, 1990) work by forecasting aggregated quantities and then by determining dis-aggregate proportions to compute lower level predictions. Shlifer and Wolff [1979] compare these two families of methods and conclude that bottom-up approaches work better. Recently, it has indeed proven successful for load forecasting to improve the global consumption prediction error (see among others Auder et al., 2018). Other approaches (neither bottom-up nor top-down) were recently introduced, for example Hyndman et al. [2011] forecast all nodes in the hierarchy and reconcile them by orthogonal projection. Moreover, Van Erven and Cugliari [2015] introduce a game-theoretically optimal reconciliation method to improve a given set of forecasts. Firstly, one comes up with some forecasts for the time series without worrying about hierarchical constraints and then a reconciliation procedure is used to make the forecasts aggregate consistent. This generalizes the previous orthogonal projection to other possible projections in the constrained space (which ensures that the forecasts satisfy the hierarchy). Finally, if we restrict here to mean forecasting, some follow-up works from Taieb et al. [2017] allow to make probabilistic forecasting in this context of hierarchical prediction.

Literature Discussion for Aggregation Methods. Aggregation methods (also called ensemble methods) for individual sequences forecasting originate from theoretical works by Vovk [1990], Cover [1991] and Littlestone and Warmuth [1994]; their distinguishing feature with respect to classical ensemble methods is that they do not rely on any stochastic modeling of the observations and thus, are able to combine forecasts independently of their generating process. They have been proved to be very effective to predict time series (see for instance Mallet et al., 2009 and Devaine et al., 2013) and those methods were used to win forecasting competitions (see Gaillard et al., 2016). This aggregation approach has recently been extended to the hierarchical setting by Goehry et al. [2019]; they used a bottom-up forecasting approach which consists in aggregating the consumption forecasts of small customers clusters.

In this chapter we combine the reconciliation approach based on orthogonal projection with various aggregation algorithms to provide new methods to which we were able to prove strong theoretical guaranties. We then illustrate the proposed methods using smart meter data collected in Great Britain by multiple energy providers (see Schellong, 2011 and AECOM, 2018). '*Energy Demand Research Project*' data gathers multiple households power consumption data. We compare various population segmentations and evaluate the performance of four strategies for the forecasting of the electricity consumption time series at the several aggregation levels: features, aggregated features, projected features and finally aggregated and projected features.

Notation. Without further indications, $\|\mathbf{x}\|$ denotes the Euclidean norm of a vector $\mathbf{x}$. For the other norms, there will be a subscript: e.g., the Frobenius norm of $\mathbf{x}$ is denoted by $\|\mathbf{x}\|_F$. Moreover, vectors will be in bold type and unless stated otherwise, they are column vectors, while matrices will be in bold underlined. We denote the inner product of two vectors $\mathbf{x}$ and $\mathbf{y}$ of the same size by $\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^{\mathrm{T}}\mathbf{y}$. Finally, the cardinal of a finite set $\mathcal{D}$ is denoted by $|\mathcal{D}|$.

3.2 Methodology

We consider a set of time series $\left\{(y_t^\gamma)_{t>0}, \gamma \in \Gamma\right\}$ connected to each other by some summation constraints: a few of them are equal to the sum of several others – see further for a definition of Γ. To forecast these time series, a set of features is generated. At any time step t, we want to forecast the vector of the values of the $|\Gamma|$ times series at t, denoted by $\mathbf{y}_t \overset{\text{def}}{=} (y_t^\gamma)_{\gamma\in\Gamma}$. We propose a three-step method to obtain relevant forecasts from these features.

3.2.1 Modeling of the Hierarchical Relationships

The relationships between the time series induce a hierarchy which should be exploited to improve forecasts. These summation constraints may be represented by one or more trees, the value at each node being equal to the sum of the ones at its leaves. Let us denote by Γ the set of the tree's nodes and $|\Gamma|$ its cardinal. There are as many summation constraints as there are nodes with leaves. Subsequently, we will introduce a matrix

$\underline{\mathbf{K}}$ to encode these relationships. Each line of $\underline{\mathbf{K}}$ is related to one of the summation constraints with -1 at the associated node and 1 at its leaves. Thus, for any instance t, the vector of the values of the $|\Gamma|$ times series at t, denoted by $\mathbf{y}_t$, is in the kernel of $\underline{\mathbf{K}}$. Details on and examples of $\underline{\mathbf{K}}$ are provided below. Example 1 treats a single summation constraint. Examples 2 and 3 present more complex relationships between the time series, considering a hierarchy with two levels and two different partitions of the same time series, respectively. Finally, Example 4 combines the two previous cases. In our experiments of Section 3.7, the underlying hierarchies will be of the form of the ones of Examples 1 and 4.

Example 1: Two-level Hierarchy. The simplest approach consists in considering a single equation connecting the time series. Here, y^{Tot} stands for the one which is the sum of the N others which are denoted by $y^1, \dots, y^N$. The underlying hierarchy is represented in Figure 1 by a tree with a single root directly connected to N leaves. For any instance t, the time series satisfy $y_t^{\text{Tot}} = y_t^1 + y_t^2 + \cdots + y_t^N$ and the vector $\mathbf{y}_t = \left(y_t^{\text{Tot}}, y_t^1, \dots, y_t^N\right)^{\text{T}}$ respects the hierarchy if and only if $\underline{\mathbf{K}}\mathbf{y}_t = 0$ with $\underline{\mathbf{K}} = \Big(-1, 1, 1, \dots, 1\Big)$. In Section 3.7, we consider the power consumption of a population of households which are distributed in N regions. This setting will correspond to the present example.

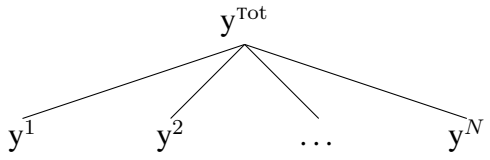


Figure 1 – Representation of a two-level hierarchy.

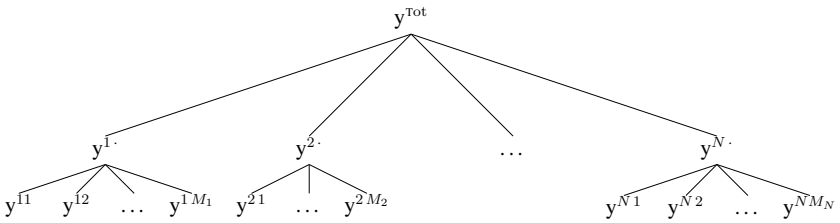


Figure 2 – Representation of a three-level hierarchy.

Example 2: Three-level Hierarchy. A few leaves of the tree of Example 1 may be broken down into new time series and so on. Figure 2 represents a complete three-level hierarchy (although we could consider any multilevel hierarchy) leading to the following summation equations, for each instance t,

$$y_t^{\text{Tot}} = y_t^{1\cdot} + y_t^{2\cdot} + \cdots + y_t^{N\cdot} \tag{1}$$

$$y_t^{i\cdot} = y_t^{i1} + y_t^{i2} + \cdots + y_t^{iM_i}, \quad \forall\, i = 1, \dots, N. \tag{2i}$$

We order the time series in lexicographical order:

$$\mathbf{y}_t = (y_t^{\text{Tot}}, y_t^{1\cdot}, y_t^{2\cdot}, \dots, y_t^{N\cdot}, y_t^{11}, y_t^{12}, \dots, y_t^{1M_1}, y_t^{21}, \dots, y_t^{NM_N}),$$

and define the constraint matrix $\underline{\mathbf{K}}$ below; each line of $\underline{\mathbf{K}}$ corresponds to one of the constraints mentioned above, either (1) or one of the N constraints $(2i)$ in a way that $\underline{\mathbf{K}}\mathbf{y}_t = \mathbf{0}$ if and only if $\mathbf{y}_t$ respects the hierarchy.

$$\underline{\mathbf{K}} = \begin{pmatrix} -1 & \overbrace{1 \quad \cdots \quad 1}^{N} & & & \leftarrow \quad (1) \\ & -1 & \overbrace{1 \quad \cdots \quad 1}^{M_1} & & \leftarrow \quad (2\text{i}) \\ & & \ddots & \ddots & \\ & & -1 & \overbrace{1 \quad \cdots \quad 1}^{M_N} & \end{pmatrix}$$

This hierarchy corresponds to the concrete example above where, for $1 \leqslant n \leqslant N$, the n region would be further divided into $M_1, \ldots, M_n$ municipalities.

Example 3: Two Hierarchies of the Same Time Series. It is also possible to consider two partitions of the same time series y_t^{Tot}. For example, in our experiments of Section 3.7, in addition to the geographical clustering, we introduce a segmentation of the households based on their profiles. Indeed, they are distributed in N_1 regions but also in N_2 groups depending on their consumption habits. These two different partitions induce the following two equations

$$\begin{aligned} y_t^{\text{Tot}} &= y_t^{1\cdot} + y_t^{2\cdot} + \cdots + y_t^{N_1\cdot} \\ y_t^{\text{Tot}} &= y_t^{\cdot 1} + y_t^{\cdot 2} + \cdots + y_t^{\cdot N_2}, \end{aligned}$$

and the two trees associated with these constraints which share the same root and are represented on Figure 3.

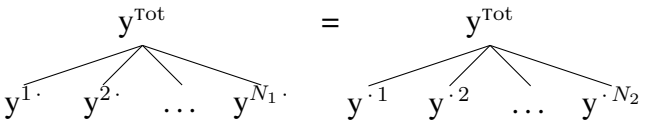


Figure 3 – Representation of two two-level hierarchies.

For any instance t, the times series vector $\mathbf{y}_t = (y_t^{\text{Tot}}, y_t^{1\cdot}, y_t^{2\cdot}, \ldots, y_t^{N_1\cdot}, y_t^{\cdot 1}, y_t^{\cdot 2}, \ldots, y_t^{\cdot N_2})$ satisfies the above equations if and only if $\underline{\mathbf{K}}\mathbf{y}_t = 0$ with

$$\underline{\mathbf{K}} = \begin{pmatrix} -1 & \overbrace{1 \quad \cdots \quad 1}^{N_1} & \\ -1 & & \overbrace{1 \quad \cdots \quad 1}^{N_2} \end{pmatrix}.$$

The equality of the roots of the two trees is always satisfied in this model. Indeed there is a single time series y_t^{Tot} to forecast and there are therefore only two summation constraints to take into account.

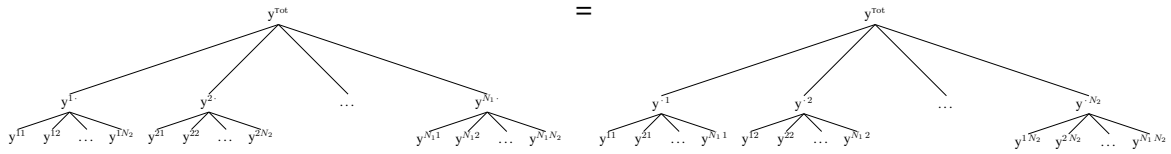


Figure 4 – Representation of two crossed hierarchies.

Example 4: Two Crossed Hierarchies. Considering two partitions, the time series can be represented with two three-level trees sharing the same root and leaves. Only the intermediate levels differs according to which partition is firstly taking into account. The leaves of the trees form a $N_1 \times N_2$-matrix $\left(y^{i\,j}\right)_{1\leqslant i\leqslant N_1,\,1\leqslant j\leqslant N_2}$. An intermediate node of the first tree $y^{i\,\cdot}$ is the sum of the line i while a node $y^{\cdot\,j}$ of the second tree is the sum of the column j. Whether we sum rows or columns first, the sum of all coefficients is y_t^{Tot}. In the experiments of Section 3.7, one partition refers to a geographic distribution of the households while the other classifies them according to their consumption behaviours. The first tree considers breaks down consumption firstly by the N_1 regions and then by the N_2 household profiles. The second one divides the households according to their habits before splitting them geographically. Both trees are represented in Figure 4. For any instance t, the time series satisfy the $2 + N_1 + N_2$ equations

$$
\begin{aligned}
y_t^{\mathrm{Tot}} &= y_t^{1\,\cdot} + y_t^{2\,\cdot} + \cdots + y_t^{N_1\,\cdot} && (1)\\
y_t^{i\,\cdot} &= y_t^{i\,1} + y_t^{i\,2} + \cdots + y_t^{i\,N_2}, \quad \forall\, i = 1, \ldots, N_1 && (2\mathrm{i})\\
y_t^{\mathrm{Tot}} &= y_t^{\cdot\,1} + y_t^{\cdot\,2} + \cdots + y_t^{\cdot\,N_2} && (3)\\
y_t^{\cdot\,j} &= y_t^{1\,j} + y_t^{2\,j} + \cdots + y_t^{N_1\,j}, \quad \forall\, j = 1, \ldots, N_2. && (4\mathrm{j})
\end{aligned}
$$

Equations (1) and (3) refer to the first level of the trees while the $N_1 + N_2$ Equations (2i) and (4j) refer to second levels. At an instance t, by ordering the time series in the vector as

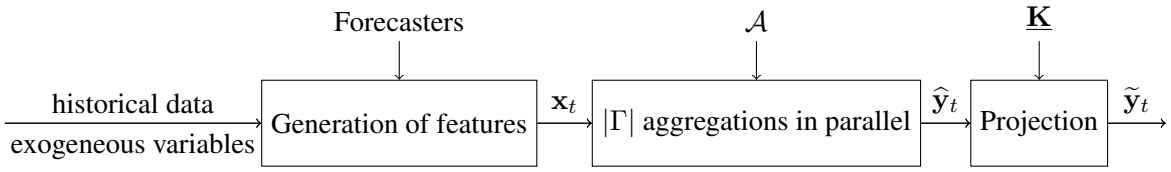
$$
\mathbf{y}_t = (y_t^{\mathrm{Tot}}, y_t^{1\,\cdot}, \ldots, y_t^{N_1\,\cdot}, y_t^{\cdot\,1}, \ldots, y_t^{\cdot\,N_1}, y_t^{1\,1}, y_t^{1\,2}, \ldots, y_t^{N_1 N_2}),
$$

it may be seen that they respect the hierarchy if and only if $\underline{\mathbf{K}}\mathbf{y}_t = 0$ with

$$
\underline{\mathbf{K}} = \begin{pmatrix}
-1 & \overbrace{1 \;\cdots\; 1}^{N_1} & & & \overbrace{1 \;\cdots\; 1}^{N_2} & & & & \leftarrow\ (1)\\
 & -1 & & & & & & & \\
 & & \ddots & & & & \ddots & & \leftarrow\ (2\mathrm{i})\\
 & & & -1 & & & & \overbrace{1 \;\cdots\; 1}^{N_2} & \\
-1 & & & & \overbrace{1 \;\cdots\; 1}^{N_2} & & & & \leftarrow\ (3)\\
 & & & & -1 & & 1 & & \\
 & & & & & \ddots & & \ddots & \leftarrow\ (4\mathrm{j})\\
 & & & & & -1 & & 1 &
\end{pmatrix}.
$$

3.2.2 A Three-step Forecast

Step 1: For each node $\gamma \in \Gamma$, at each instance t, thanks to an historical data set of the γ time series and to some exogenous variables proper to the node γ, a forecaster makes the prediction x_t^γ. These $|\Gamma|$ benchmark forecasts are then collected into the feature vector $\mathbf{x}_t \stackrel{\text{def}}{=} \left(x_t^\gamma\right)_{\gamma\in\Gamma}$. We propose to use the knowledge of all of the features, namely the $|\Gamma|$ benchmark forecasts, and of the summation constraints to improve these $|\Gamma|$ predictions. Step 2: For each node γ and each instance t, we form our prediction $\widehat{y}_t^\gamma$ by linearly combining the components of the feature vector $\mathbf{x}_t$ thanks to a so-called aggregation algorithm (a copy $\mathcal{A}^\gamma$ of an aggregation algorithm $\mathcal{A}$ is run separately for each node γ). That is, we use all $|\Gamma|$ benchmark forecasts to predict y_t^γ, not only x_t^γ. We explain below why this is a good idea – the main reason is given by correlations between time series. The forecasts thus obtained are then gathered into a vector $\widehat{\mathbf{y}}_t \stackrel{\text{def}}{=} \left(\widehat{y}_t^\gamma\right)_{\gamma\in\Gamma}$. Step 3: Finally, a re-conciliation step will update the forecast vector so that it is in the kernel $\underline{\mathbf{K}}$. Let us denote by $\widetilde{\mathbf{y}}_t$ the final vector of forecasts. We detail below each step of our procedure.



First Step: Generation of features. At a fixed node $\gamma \in \Gamma$, for any instance t, a forecasting method, which may depend on γ, predicts x_t^γ with the historical data and the exogenous variables of the node γ. The forecasting methods we use in the experiments of Section 3.7 are described in Section 3.5 and include non linear sequential ridge regression, fully adaptive Bernstein online aggregation and polynomially weighted average forecaster with multiple learning rates. These benchmark forecasts are henceforth called features and are gathered in $\mathbf{x}_t = \left(x_t^\gamma\right)_{\gamma\in\Gamma}$. This feature vector is used in the aggregation step that comes next to predict again each time series; we discuss below and in Section 3.2.4 why we do so (the main reasons being that it is a good idea because of the correlations between the times series and also because it eases the description of our method). We focus here on $|\Gamma|$ benchmark forecasts – one for each of the nodes; however, we could also have considered several predictions per nodes.

Second Step: Aggregation. The above features are generated independently with different exogenous variables and possibly different methods. Yet, the observations $\left(y_t^\gamma\right)_{\gamma\in\Gamma}$ may be correlated. For example, considering load forecasting, the consumptions associated with two nearby regions can be strongly similar. Furthermore, the observations are related though the summations constraints (although we disregard these equations here). This is why linearly combining the features may refine some forecasts – this is exactly what this step does. Formally, an aggregation algorithm outputs at each round a vector of weights $\widehat{\mathbf{u}}_t^\gamma$ and returns the forecast $\widehat{y}_t^\gamma \stackrel{\text{def}}{=} \widehat{\mathbf{u}}_t^\gamma \cdot \mathbf{x}_t$. It does so based on the information available, that is, the feature vector $\mathbf{x}_t$ and past data. We consider an aggregation rule $\mathcal{A}$ (see Section 3.6) and form a copy $\mathcal{A}^\gamma$ for each node γ, which we feed with an input parameter vector $\mathbf{s}_0^\gamma$. These

predictions are then gathered into the vector $\widehat{\mathbf{y}}_t = (y_t^\gamma)_{\gamma\in\Gamma}$. This algorithm aims for the best linear combination of features and there are theoretical performance guaranties associated with these aggregation algorithms, see Section 3.6 for details.

Instead of this approach based on benchmark forecasting and aggregation node by node, we could have considered a meta-model to directly predict the time series vector $\left(y_t^\gamma\right)_{\gamma\in\Gamma}$ at each instant t (with a common forecaster and therefore without any aggregation step). Once this global forecast would have been obtained, we would have gone straight to the projection stage. In such a model, the number of variables to be taken into account (the historical data of the time series but also the exogenous variables specific to each node) would have been considerable and getting relevant forecasts would have not been an easy task. But actually, a practical choice motivated our method for the most. Indeed, the forecasters may be black boxes proper to each node and the exogenous variables of a node γ may be unknown at a node γ'. In our experiments, we followed this three-step approach. However, our method totally operates if, for each node γ and at each instance t, an external expert provides the forecast x_t^γ. How these features have been obtained is no longer an issue and the aim is to improve these benchmark forecasts with aggregation and reconciliation steps. Thus, at each instance t, only the features are reveal at time t and by skipping the generation of features step, we go straight to the aggregation step.

Meta-algorithm 1 Aggregation and projection of features with summation constraints

Input
Set of nodes Γ and constraint matrix $\underline{\mathbf{K}}$
Feature generation technique, see Section 3.5
Aggregation algorithm $\mathcal{A}$ taking parameter vector $\mathbf{s}_0$, see Section 3.6
Compute the orthogonal projection matrix $\Pi_{\underline{\mathbf{K}}} = \left(I_{|\Gamma|} - \underline{\mathbf{K}}^{\mathrm{T}}(\underline{\mathbf{K}}\underline{\mathbf{K}}^{\mathrm{T}})^{-1}\underline{\mathbf{K}}\right)$
for $\gamma \in \Gamma$ **do**
Create a copy of $\mathcal{A}$ denoted by $\mathcal{A}^\gamma$ and run with $\mathbf{s}_0^\gamma$
for $t = 1, \ldots$ **do**
Generate features $\mathbf{x}_t$
for $\gamma \in \Gamma$ **do**
$\mathcal{A}^\gamma$ outputs $\widehat{y}_t^\gamma = \mathbf{u}_t^\gamma \cdot \mathbf{x}_t$
Collect forecasts: $\widehat{\mathbf{y}}_t = (\widehat{y}_t^\gamma)_{\gamma\in\Gamma}^{\mathrm{T}}$
Project forecasts: $\widetilde{\mathbf{y}}_t = \Pi_{\underline{\mathbf{K}}}(\widehat{\mathbf{y}}_t)$
for $\gamma \in \Gamma$ **do**
$\mathcal{A}^\gamma$ observes y_t^γ
Suffer a prediction error $\frac{1}{|\Gamma|}\sum_{\gamma\in\Gamma}\left(y_t^\gamma - \widetilde{y}_t^\gamma\right)^2$
aim
Minimize the average prediction error $\dfrac{1}{T}\displaystyle\sum_{t=1}^{T}\frac{1}{|\Gamma|}\left\|\mathbf{y}_t - \widetilde{\mathbf{y}}_t\right\|^2 = \frac{1}{T\,|\Gamma|}\sum_{t=1}^{T}\sum_{\gamma\in\Gamma}\left(y_t^\gamma - \widetilde{y}_t^\gamma\right)^2$.

Third Step: Projection. As the $|\Gamma|$ executions of Algorithm $\mathcal{A}$ are run in parallel and independently, the obtained forecast vector $\widehat{\mathbf{y}}_t$ does not necessary respect hierarchical constraints. To correct that, we consider the orthogonal projection of $\widehat{\mathbf{y}}_t$ onto the kernel of $\underline{\mathbf{K}}$, which we denote by $\Pi_{\underline{\mathbf{K}}}(\widehat{\mathbf{y}}_t)$. This updated forecast $\widetilde{\mathbf{y}}_t \overset{\text{def}}{=} \Pi_{\underline{\mathbf{K}}}(\widehat{\mathbf{y}}_t)$ fulfills the hierarchical constraints.

To sum up, at each instant t, we first generate benchmark forecasts – also called features – $\mathbf{x}_t$. These predictions are then aggregated to form a new vector of forecast $\widehat{\mathbf{y}}_t$, which is itself updated in the projection step in $\widetilde{\mathbf{y}}_t$. This procedure is stated in Meta-algorithm 1. Moreover, we can also directly project the features, skipping the aggregation step; this leads to the forecasts $\Pi_{\underline{\mathbf{K}}}(\mathbf{x}_t)$. Thus, we get four forecasts ($\mathbf{x}_t$, $\Pi_{\underline{\mathbf{K}}}(\mathbf{x}_t)$, $\widehat{\mathbf{y}}_t$ and $\widetilde{\mathbf{y}}_t$) for each node and each instant. The performance of our strategies is measured in mean squared error. In Section 3.7, we compare these four methods in the scope of power consumption forecasting.

3.2.3 Assessment of the Forecasts – Form of the Theoretical Guaranties Achieved

Our forecast are linear combinations of the features and are evaluated by the average prediction error

$$\widetilde{L}_T \overset{\text{def}}{=} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\Gamma|}\sum_{\gamma\in\Gamma}\left(y_t^\gamma - \widetilde{y}_t^\gamma\right)^2. \tag{3.1}$$

We want to compare our method to constant linear combinations of features. For example, recalling that, for $\gamma \in \Gamma$, x_t^γ is the benchmark prediction of y_t^γ, using $\boldsymbol{\delta}^\gamma \overset{\text{def}}{=} \mathbf{1}_{\{i=\gamma\}}$ (the standard basis vector that points in the γ direction) as weights should be a good first choice to define a constant linear combination (for any $\gamma \in \Gamma$, this strategy provides $\boldsymbol{\delta}^\gamma \cdot \mathbf{x}_t = x_t^\gamma$ as forecast for y_t^γ). Thus, the matrix $\left(\boldsymbol{\delta}^\gamma\right)_{\gamma\in\Gamma}$ defines a constant benchmark strategy and its cumulative prediction error is

$$L_T\Big(\left(\boldsymbol{\delta}^\gamma\right)_{\gamma\in\Gamma}\Big) \overset{\text{def}}{=} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\Gamma|}\sum_{\gamma\in\Gamma}\left(y_t^\gamma - x_t^\gamma\right)^2.$$

As soon as the features $(x_t^\gamma)_{\gamma\in\Gamma}$ are well-chosen, this quantity is small. But, these benchmark predictions do not satisfy the summation constraints *a priori* and it won't be fair to compare our forecasts (which do respect to hierarchy – projection step ensures it) to these benchmark forecasts – or any other constant linear combinations of features. Thus, we introduce, in Section 3.2.3.1, the set $\mathcal{C}$ which contains all the constant strategies which satisfy the hierarchical constraints and we also detail how a such strategy can be represented by a $|\Gamma| \times |\Gamma|$-matrix $\mathbf{U} \in \mathcal{C}$. In Section 3.2.3.2, we decompose, for any $\mathbf{U} \in \mathcal{C}$, the average prediction error into an approximation error $L_T(\mathbf{U})$ – the average prediction error of $\mathbf{U}$ – and a sequential estimation error $\mathcal{E}_T(\mathbf{U})$. To achieve almost as well as the best constant combination of features, we want to obtain some guarantee of the form:

$$\widetilde{L}_T \leqslant \inf_{\mathbf{U}\in\mathcal{C}}\Big\{L_T(\mathbf{U}) + \mathcal{E}_T(\mathbf{U})\Big\}, \quad \text{where} \quad \mathcal{E}_T(\mathbf{U}) = \mathcal{O}\Big(\frac{1}{\sqrt{T}}\Big). \tag{3.2}$$

Indeed, if $\mathcal{E}_T(\mathbf{U}) \underset{T\to+\infty}{\longrightarrow} 0$, the average prediction error of our strategy tends to $L_T(\mathbf{U})$ – and classical convergence rate are in $\frac{1}{\sqrt{T}}$ (see Section 3.2.3.2 and Section 3.6). We will explain how this aim is equivalent to minimizing the quantity called regret that we define below.

3.2.3.1 Class of Comparison

We consider here a constant strategy, namely $|\Gamma|$ linear combinations of the features. More formally, let us denote by $\mathbf{u}^\gamma$ a constant weight vector which provides, for any instance t, the forecast $\mathbf{u}^\gamma \cdot \mathbf{x}_t$ for the time series y_t^γ. By batching these $|\Gamma|$ vectors into a matrix $\mathbf{U} \overset{\text{def}}{=} (\mathbf{u}^\gamma)_{\gamma\in\Gamma} \in \mathcal{M}_{|\Gamma|}$, predictions satisfy the constraints for an instance t if $\mathbf{U}^\intercal \mathbf{x}_t \in \mathrm{Ker}(\underline{\mathbf{K}})$. For it to be true for any t (except for a few particular case – for instance if all features vector are null), this requires that the image of $\mathbf{U}^\intercal$ is in the kernel of $\underline{\mathbf{K}}$. We introduce the following set of matrices, for which associated forecasts necessarily satisfy the hierarchical constraints

$$\mathcal{C} \overset{\text{def}}{=} \left\{ \mathbf{U} = \left(\mathbf{u}^1 \,\middle|\, \dots \,\middle|\, \mathbf{u}^{|\Gamma|}\right) \,\middle|\, \mathrm{Im}\left(\mathbf{U}^\intercal\right) \subset \mathrm{Ker}\left(\underline{\mathbf{K}}\right) \right\}.$$

Note that, for any matrix $\mathbf{U} \in \mathcal{M}_{|\Gamma|}$, by definition of the orthogonal projection $\Pi_{\underline{\mathbf{K}}}$, the forecast vector $\Pi_{\underline{\mathbf{K}}}\mathbf{U}^\intercal\mathbf{x}_t$ satisfies the hierarchical relationships so the set $\mathcal{C}$ contains the matrix $\mathbf{U}\Pi_{\underline{\mathbf{K}}}^\intercal$. This implies that the set $\mathcal{C}$ is not empty. To compare our methods to any constant strategy $\mathbf{U} \in \mathcal{C}$, we now introduce the common notion of regret.

3.2.3.2 Aim: Regret Minimization

We want to compare the average prediction error $\widetilde{L}_T$ to $L_T(\mathbf{U})$, where $\mathbf{U} \in \mathcal{C}$ so the forecasts associated with $\mathbf{U}$ satisfy the hierarchical constraints – otherwise, the two strategies would not be comparable because our predictions do respect the hierarchy. Good algorithms should ensure that $\widetilde{L}_T$ is not too far from the best $L_T(\mathbf{U})$. We thus define, for any $\mathbf{U} = (\mathbf{u}^\gamma)_{\gamma\in\Gamma} \in \mathcal{C}$, the cumulative prediction error of the associated constant linear combinations of features by

$$L_T(\mathbf{U}) \overset{\text{def}}{=} \frac{1}{T}\sum_{t=1}^{T} \frac{1}{|\Gamma|} \sum_{\gamma\in\Gamma} \left(y_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t\right)^2 = \frac{1}{T|\Gamma|}\sum_{t=1}^{T} \left\|\mathbf{y}_t - \mathbf{U}^\intercal\mathbf{x}_t\right\|^2.$$

In order to obtain a theoretical guarantee of the form of Equation (3.2), we decompose the average prediction error as

$$\widetilde{L}_T = L_T(\mathbf{U}) + \frac{\mathcal{R}_T(\mathbf{U})}{T|\Gamma|}, \tag{3.3}$$

where, the quantity $\mathcal{R}_T(\mathbf{U})$, commonly called regret is defined as the difference between the cumulative prediction error of our method and the one for weights $\mathbf{U}$:

$$\mathcal{R}_T(\mathbf{U}) \overset{\text{def}}{=} T|\Gamma| \times \left(\widetilde{L}_T - L_T(\mathbf{U})\right) = \sum_{t=1}^{T} \left\|\mathbf{y}_t - \widetilde{\mathbf{y}}_t\right\|^2 - \sum_{t=1}^{T} \left\|\mathbf{y}_t - \mathbf{U}^\intercal\mathbf{x}_t\right\|^2.$$

In the light of Equation (3.3), the average prediction error $\widetilde{L}_T$ we attempt to minimize breaks down into an approximation error $L_T(\mathbf{U})$ (the best prediction error we can hope for) and a sequential estimation error (dependent of how quickly the model estimate $\mathbf{U}$), proportional to the regret $\mathcal{R}_T(\mathbf{U})$. As stated before, the aim for algorithms is that $\widetilde{L}_T$ is as close as pos-

sible to $\min_{\mathbf{U}\in\mathcal{C}} L_T(\mathbf{U})$ (with $\mathcal{C}$ the class of comparison defined above), which is equivalent to $\max_{\mathbf{U}\in\mathcal{C}} \mathcal{R}_T(\mathbf{U})$ being small. This point of view is very common for online forecasting methods (see, among others, Devaine et al., 2013 and Mallet et al., 2009), and for an algorithm to be useful, $\max_{\mathbf{U}\in\mathcal{C}} \mathcal{R}_T(\mathbf{U})$ need to be sub-linear in T (otherwise the error remains constant – or even worst: it increases with time). Typical theoretical guaranties provide bounds of order $\sqrt{T}$ (see for example, Raphaël et al., 2019 and Amat et al., 2018).

3.2.4 Technical Discussion: why we require the same features at each node.

In this section, we explain why we consider the same features vector for each nodes. *A priori*, we could have a different set of features at each node $\mathbf{x}_t^\gamma$, created with methods specific to this node. Also the size of feature vector d^γ associated with the node γ could vary. Prediction of a time series y_t^γ associated to a d^γ-vector $\mathbf{u}^\gamma$ is $\widehat{y}_t^\gamma = \mathbf{u}^\gamma \cdot \mathbf{x}_t^\gamma$. Therefore, a global constant strategy is a set $\left\{\mathbf{u}^1, \dots, \mathbf{u}^{|\Gamma|}\right\} \subset \mathbb{R}^{d^1\times\cdots\times d^{|\Gamma|}}$. First is is a little less practical because unlike the previous setting, the vectors $\mathbf{u}^1, \dots, \mathbf{u}^{|\Gamma|}$ and $\mathbf{x}^1, \dots, \mathbf{x}^{|\Gamma|}$ are of different sizes, so it is less easy to use matrix notations. Moreover, it becomes tricky to specify the class of constant strategies to compare to. As said before, the forecast vector $\widehat{\mathbf{y}}_t = (\widehat{y}_t^\gamma)_{\gamma\in\Gamma}$ satisfies the summation constraints if and only if it is in the kernel of $\underline{\mathbf{K}}$. Thus, the following set, which contains the constant strategies fulfilling the hierarchical constraints for all $t > 0$,

$$\left\{ \left(\mathbf{u}^1, \dots, \mathbf{u}^{|\Gamma|}\right) \in \mathbb{R}^{d^1\times\cdots\times d^{|\Gamma|}} \,\middle|\, \forall t > 0,\ \left(\mathbf{u}^1 \cdot \mathbf{x}_t^1, \dots, \mathbf{u}^{|\Gamma|} \cdot \mathbf{x}_t^{|\Gamma|}\right)^{\mathrm{T}} \in \mathrm{Ker}\left(\underline{\mathbf{K}}\right) \right\},$$

is not explicitly defined and may be empty because of the number of constraints on $(\mathbf{u}^1, \dots, \mathbf{u}^{|\Gamma|}) \in \mathbb{R}^{d^1\times\cdots\times d^{|\Gamma|}}$ which increases at each time step. If there is no restrictions on the feature vectors, these constraints could be linearly independent, leading to an empty set. Indeed, if we consider that the times series are connected by K summations relationships, at each instance t, the $d^1 + \cdots + d^{|\Gamma|}$ coefficients of vectors $\mathbf{u}^1, \dots, \mathbf{u}^{|\Gamma|}$ are linked by K equations. As features are proper to each node, theses constraints have no reason to be dependent, so as soon as $T \times K > d^1 + \cdots + d^{|\Gamma|}$, the above set may likely be empty. Because of that, it is not clear how to define the regret in this setting. For this reason, we decided to use the same features vectors $\mathbf{x}_t$ for all nodes of Γ; which has also the benefit of allowing a simpler presentation.

3.3 Main Theoretical Result

From now on, let us introduce the following notation concerning the regret bound of Algorithm $\mathcal{A}$.

Notation 3.1. *We assume that, for any $\gamma \in \Gamma$ with the initialization parameter vector $\mathbf{s}_0^\gamma$, Algorithm $\mathcal{A}^\gamma$ ensures, for $T > 0$ and for any $\mathbf{u}^\gamma \in \mathbb{R}^{|\Gamma|}$, any $\mathbf{x}_{1:T} = \mathbf{x}_1, \dots \mathbf{x}_T$ and any $y_{1:T}^\gamma = y_1^\gamma, \dots, y_T^\gamma$,*

$$\mathcal{R}_T^\gamma(\mathbf{u}^\gamma) \overset{def}{=} \sum_{t=1}^{T} \left(y_t^\gamma - \widehat{y}_t^\gamma\right)^2 - \sum_{t=1}^{T} \left(y_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t\right)^2 \leqslant \mathrm{B}(\mathbf{x}_{1:T}, y_{1:T}^\gamma, \mathbf{s}_0^\gamma, \mathbf{u}^\gamma). \tag{3.4}$$

Details and examples of these regret bounds are provided in Section 3.6 that describes the aggregation algorithms considered in the experiments of Section 3.7. As getting a linear bound is trivial (by using the common assumption that prediction errors are bounded), these bounds have to be sub-linear to be of interest. Referring to the average prediction error decomposition of Equation (3.3), the sub-linearity ensures that the sequential estimation error $\mathcal{R}_T^\gamma(\mathbf{u}^\gamma)/T$ tends to 0. This notation makes it possible to establish a bound of the cumulative regret.

Theorem 3.2. *Under Notation 3.1, for any matrix* $\mathbf{U} \in \mathcal{C}$ *and any* $T \geqslant 1$,

$$\mathcal{R}_T(\mathbf{U}) \overset{def}{=} \sum_{t=1}^{T} \left\|\mathbf{y}_t - \widetilde{\mathbf{y}}_t\right\|^2 - \sum_{t=1}^{T} \left\|\mathbf{y}_t - \mathbf{U}^{\mathrm{T}}\mathbf{x}_t\right\|^2 \leqslant \sum_{\gamma\in\Gamma} \mathrm{B}\Big(\mathbf{x}_{1:T}, y_{1:T}^\gamma, \mathbf{s}_0^\gamma, \mathbf{u}^\gamma\Big).$$

The regret $\mathcal{R}_T(\mathbf{U})$ is not just the sum over all the nodes of the regrets $\mathcal{R}_T^\gamma(\mathbf{u}^\gamma)$ of Equation 3.4. Indeed, we do not evaluate here the forecasts $\widehat{\mathbf{y}}_t$ but those obtained after the projection step: $\widetilde{\mathbf{y}}_t$. The projection step provides a diminishing of the square prediction error and we just have to sum Equation 3.4 on all nodes to get the bound.

Proof. This regret bound results from two main arguments: Pythagorean theorem, on the one hand, and Notation 3.1, on the other hand. For any $t \geqslant 1$, as $\mathbf{y}_t \in \mathrm{Ker}(\underline{\mathbf{K}})$, the Pythagorean theorem ensures

$$\left\|\mathbf{y}_t - \widetilde{\mathbf{y}}_t\right\|^2 = \left\|\mathbf{y}_t - \Pi_{\underline{\mathbf{K}}}(\widehat{\mathbf{y}}_t)\right\|^2 \leqslant \left\|\mathbf{y}_t - \widehat{\mathbf{y}}_t\right\|^2. \tag{3.5}$$

Let us fix a matrix $\mathbf{U} = \Big(\mathbf{u}^1 | \ \ldots \ | \, \mathbf{u}^{|\Gamma|}\Big) \in \mathcal{C}$. Firstly, the application of Pythagorean theorem ensures that the projection step reduces regret. Rewriting the regret as a sum over the nodes, we then use Notation 3.1 independently for each node of Γ to conclude the proof.

$$\begin{aligned}
\mathcal{R}_T(\mathbf{U}) &= \sum_{t=1}^{T} \left\|\mathbf{y}_t - \widetilde{\mathbf{y}}_t\right\|^2 - \sum_{t=1}^{T} \left\|\mathbf{y}_t - \mathbf{U}^{\mathrm{T}}\mathbf{x}_t\right\|^2 \\
&\overset{(3.5)}{\leqslant} \sum_{t=1}^{T} \left\|\mathbf{y}_t - \widehat{\mathbf{y}}_t\right\|^2 - \sum_{t=1}^{T} \left\|\mathbf{y}_t - \mathbf{U}^{\mathrm{T}}\mathbf{x}_t\right\|^2 \\
&= \sum_{\gamma\in\Gamma}\sum_{t=1}^{T} \Big(y_t^\gamma - \widehat{y}_t^\gamma\Big)^2 - \sum_{\gamma\in\Gamma}\sum_{t=1}^{T} \Big(y_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t\Big)^2 = \sum_{\gamma\in\Gamma} \mathcal{R}_T^\gamma\Big(\mathbf{u}^\gamma\Big) \\
&\overset{(3.4)}{\leqslant} \sum_{\gamma\in\Gamma} \mathrm{B}\Big(\mathbf{x}_{1:T}, y_{1:T}^\gamma, \mathbf{s}_0^\gamma, \mathbf{u}^\gamma\Big).
\end{aligned}$$

□

Remark 8. For an initialization parameter vector $\mathbf{s}_0^\gamma$, and a subset $\mathcal{D} \subset \mathbb{R}^{|\Gamma|}$, some aggregation algorithms provide a uniform regret bound of the following form:

$$\mathcal{R}_T^\gamma(\mathcal{D}) \overset{\text{def}}{=} \sum_{t=1}^{T} \Big(y_t^\gamma - \widehat{y}_t^\gamma\Big)^2 - \min_{\mathbf{u}^\gamma\in\mathcal{D}} \sum_{t=1}^{T} \Big(y_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t\Big)^2 \leqslant \mathrm{B}\Big(\mathbf{x}_{1:T}^\gamma, y_{1:T}^\gamma, \mathbf{s}_0^\gamma\Big).$$

In this case, let us introduce, for any subset $\mathcal{B} \subset \mathcal{M}_{|\Gamma|}$, the subset of comparaison matrices such we have a regret bound for every line vector $\mathcal{B}_{|\mathcal{D}} \overset{\text{def}}{=} \Big\{\mathbf{U} \in \mathcal{B} \,|\, \forall\gamma \in \Gamma,\ \mathbf{u}^\gamma \in \mathcal{D}\Big\}$. Then,

we bound the cumulative regret $\mathcal{R}_T(\mathcal{D})$ defined just below with

$$\mathcal{R}_T(\mathcal{D}) \stackrel{\text{def}}{=} \max_{\mathbf{U}\in\mathcal{C}_{|\mathcal{D}}} \mathcal{R}_T(\mathbf{U}).$$

With the same previous arguments we get the uniform regret bound

$$\mathcal{R}_T(\mathcal{D}) = \sum_{t=1}^{T} \left\|\mathbf{y}_t - \widetilde{\mathbf{y}}_t\right\|^2 - \min_{\mathbf{U}\in\mathcal{C}_{|\mathcal{D}}} \sum_{t=1}^{T} \left\|\mathbf{y}_t - \mathbf{U}^{\mathrm{T}}\mathbf{x}\right\|^2 \leqslant \sum_{\gamma\in\Gamma} \mathrm{B}\left(\mathbf{x}_{1:T}^{\gamma}, y_{1:T}^{\gamma}, \mathbf{s}_0^{\gamma}\right).$$

3.4 On one Operational Constraint: Half-Hourly Predictions with One-Day-Delayed Observations

In this section, we highlight the differences between the previous theoretical setting and the practical setting of our experiments and how these changes affect the regret bound. In Section 3.7, we aim to forecast power consumptions at half-hourly intervals. Meta-algorithm 1 makes the implicit assumption that historical time series values are available and to forecast at an instance t, we can use $\mathbf{y}_{1:t-1}$. We thus assume that very recent past observations, up to half an hour ago, would be available – and it is not realistic at all. Indeed, there is some operational constraints on the power network and on meters that make it difficult to instantly access the data: it is common to obtain load records with a delay of a few hours or even a few days. Although this delay is becoming shorter with the deployment of smart meters and the evolution of grids, we cannot consider we have access to the consumption of the previous half-hour. To take into account these operational constraints and to carry out experiments under practical conditions, we make the classic assumption that we have access to consumptions with a delay of 24 hours (see among others Fan and Hyndman, 2011 and Gaillard et al., 2016). As now, only past observations $\mathbf{y}_{1:t-48}$ are available at an instance t, we adapt the previous method a bit.

As we will see in Section 3.5, the half-hour of the day is a crucial variable for power consumption forecasting and to obtain relevant forecasts, we will consider the consumption of the previous day at the same half-hour (but never the one of the previous half-hour). Thus the delay in the access to consumption observation is not an issue for feature generation. But it becomes especially problematic for online learning (in our experiments, features are generated offline with models trained on historical data). Indeed, in the aggregation step of our method, we assume to observe, for each node γ and at each instance t, the consumption y_{t-1}^{γ} – that is not possible anymore. To deal with this issue we initially considered two solutions. In our first approach, for any $\gamma \in \Gamma$, the time series (y_t^{γ}) is divided into 48 time series with daily time steps. Then, 48 aggregations are done in parallel and, as $t-1$ now refers to the previous day, there is no more delay issue. The 48 series are then collected to reconstruct a time series at half-hour time step. For a constant strategy $\mathbf{u}^{\gamma}$, the regret of the global aggregation $\mathcal{R}_T^{\gamma}(\mathbf{u}^{\gamma})$ is simply the sum of the 48 regrets – that refer to the 48 aggregation run in parallel on the 48 daily time series – denoted by $\left(R_{T/48}^{\gamma\,h}(\mathbf{u}^{\gamma})\right)_{1\leqslant h\leqslant 48}$, so we have

$$\mathcal{R}_T^{\gamma}(\mathbf{u}^{\gamma}) = \sum_{h=1}^{48} R_{T/48}^{\gamma\,h}(\mathbf{u}^{\gamma}).$$

If we consider an aggregation algorithm that ensures a bound of the form of Notation 3.1 where the bound B depends only on the horizon time – namely, $\mathcal{R} \leqslant B$ for all h – the regret associated with the half-hourly time series (y_t^γ) satisfies:

$$\mathcal{R}_T^\gamma(\mathbf{u}^\gamma) \leqslant 48 \times B(T/48).$$

Joulani et al. [2013] provide an overview of work on online learning under delayed feedback and for our framework, which refers to full information setting with general feedback. The bound above matches their results. In a second approach, we "ignore" the delay in a sense that we apply the aggregation algorithms as if the delayed observations $\mathbf{y}_{t-48}$ were $\mathbf{y}_t$. Thus, in Meta-algorithm 1, at each node γ and any instance t, instead of outputting the forecast $y_t^\gamma = \mathbf{u}_t^\gamma \cdot \mathbf{x}_t$, algorithm $\mathcal{A}^\gamma$ outputs $y_t^\gamma = \mathbf{u}_{t-48}^\gamma \cdot \mathbf{x}_t$. For simplicity of notation, the aggregation algorithms of Section 3.6 are presented in their original version, namely assuming that observations at $t-1$ are available at an instance t. Such adaptations have already been tested: Algorithm 15 of Gaillard [2015] gives a delayed version of Algorithm 4 that we also use in Section 3.7. After testing both approaches, we kept the second one, which achieves a much better performance. Our choice was also supported by Chapter 9 of Gaillard [2015] experiments, which drew similar conclusions.

3.5 Generation of the Features

Here we describe the forecasting methods we use in the experiments of Section 3.7 to generate the benchmark predictions that will be used as features in the sequel. We recall (see Section 3.2) that throughout this work, we consider that, at each node $\gamma \in \Gamma$ and for any instance t, a forecaster provides a benchmark prediction x_t^γ based on historical data of the time series $(y_t^\gamma)_{\gamma\in\Gamma}$ and on exogenous variables relative to the node γ. These $|\Gamma|$ forecasters independently generate the $|\Gamma|$ forecasts (x_t^γ) in parallel and the set of features $\mathbf{x}_t$ is made up of the above $|\Gamma|$ benchmark predictions. Forecasts can be the output of any predictive model. In the experiments of Section 3.7, we consider three forecasting methods, that are described in the following Sections 3.5.1, 3.5.2 and 3.5.3.

Notation. Sections 3.5.1 and 3.5.2 present parametric methods. For any parameter a of the model, we will denote by $\widehat{a}$ its estimation (no matter the method we use).

3.5.1 Auto-Regressive Model

A simple approach consists in considering an auto-regressive model. Let us fix $\gamma \in \Gamma$ and assume that, to predict the time series $(y_t^\gamma)_{t>0}$, we have access to historical observations. For an instance t, the model specifies that the output variable y_t^γ depends linearly on its own previous values. In Section 3.7, we consider the power consumption at half-hourly intervals. For an instance t, to forecast the time series y_t^γ we assume to have access to the power consumption at D-1 and D-7, which correspond to y_{t-48}^γ and $y_{t-7\times48}^\gamma$, respectively. We predict the consumption half-hour by half-hour thanks to linear models taking as explanatory variables its values at D-1 and D-7. We assume that these 48 auto-regressive models have

the same coefficients. Thus, for this modeling, the power consumption associated with the node γ equals

$$y_t^\gamma = a_1^\gamma y_{t-48}^\gamma + a_7^\gamma y_{t-7\times 48}^\gamma + \text{noise}\,.$$

For each $\gamma \in \Gamma$, we estimate the coefficients a_1^γ and a_7^γ using ordinary least squares regression on a training data set. Therefore, at an new instant t, we predict

$$x_t^\gamma = \widehat{a}_1^\gamma y_{t-48}^\gamma + \widehat{a}_7^\gamma y_{t-7\times 48}^\gamma\,.$$

3.5.2 General Additive Model

Generalized additive models (see the monograph of Wood, 2006 an in-depth presentation) are effective semi-parametric approaches to forecast electricity consumption (see, among others, Goude et al., 2014 and Gaillard et al., 2016). They model the power demand as a sum of independent exogenous (possibly non-linear) variable effects. We describe this model using the specification we chose in our experiments. In Section 3.7, for a node $\gamma \in \Gamma$, we take into account some local meteorological variables at the half-hour time step: the temperature τ^γ and the smoothed temperature $\bar{\tau}^\gamma$, the visibility ν^γ, and the humidity κ^γ. For an instant t, we also introduce calendar variables: the day of the week d_t (equal to 1 for Monday, 2 for Tuesday, etc.), the half-hour of the day $h_t \in \{1, ..., 48\}$ and the position in the year $\rho_t \in [0, 1]$, which takes linear values between $\rho_t = 0$ on January 1st at 00:00 and $\rho_t = 1$ on December the 31st at 23:59. As the effect of the half-hour h_t is crucial to forecast load, it is often more efficient to consider a model per half-hour (see Fan and Hyndman, 2011 and Goude et al., 2014). The global model is then the sum of 48 daily models, one for each half-hour of the day. More precisely, we consider the following additive model for the load, which breaks down time by half hours:

$$y_t^\gamma = \sum_{h=1}^{48} \mathbf{1}_{h_t=h}\Big[a_h^\gamma y_{t-7\times 48}^\gamma + s_{1,h}^\gamma\Big(y_{t-48}^\gamma\Big) + s_{\tau,h}^\gamma(\tau_t^\gamma) + s_{\bar{\tau},h}^\gamma(\bar{\tau}_t^\gamma) + s_{\nu,h}^\gamma(\nu_t^\gamma) + s_{\kappa,h}^\gamma(\kappa_t^\gamma) + \sum_{d=1}^{7} w_{d,h}^\gamma \mathbf{1}_{d_t=d} + s_{\rho,h}^\gamma(\rho_t)\Big] + \text{noise}.$$

The $s_{1,h}^\gamma$, $s_{\tau,h}^\gamma$, $s_{\bar{\tau},h}^\gamma$, $s_{\nu,h}^\gamma$, $s_{\kappa,h}^\gamma$ and $s_{\rho,h}^\gamma$ functions catch the effect of the consumption lag, the meteorological variables and of the yearly seasonality. They are cubic splines: $\mathcal{C}^2$-smooth functions made up of sections of cubic polynomials joined together at points of a grid. The coefficients a_h^γ and $w_{d,h}^\gamma$ model the influence of the consumption at D-7 and of the day of the week. Indeed, we consider a linear effect for the consumption at D-7 (it achieved a better performance than a spline effect in our experiments) and as the day of the week takes only 7 values, we write its effect as a sum of indicator functions, and thus 7 coefficients $w_{d,h}^\gamma$ are considered. As we consider a model per half-hour, all the coefficients and splines are indexed by h. To estimate each model, we use the Penalized Iterative Re-Weighted Least Square (P-IRLS) method Wood, 2006, implemented in the `mgcv` R-package, on a training

data set. At any node $\gamma \in \Gamma$, for a new round t, we then output the forecast

$$x_t^\gamma = \sum_{h=1}^{48} \mathbf{1}_{h_t=h} \Big[\widehat{a}_h^\gamma y_{t-7\times 48}^\gamma + \widehat{s}_{1,h}^\gamma\Big(y_{t-48}^\gamma\Big) + \widehat{s}_{\tau,h}^\gamma(\tau_t^\gamma) + \widehat{s}_{\bar{\tau},h}^\gamma(\bar{\tau}_t^\gamma) + \widehat{s}_{\nu,h}^\gamma(\nu_t^\gamma) + \widehat{s}_{\kappa,h}^\gamma(\kappa_t^\gamma) + \sum_{d=1}^{7} \widehat{w}_{d,h} \mathbf{1}_{d_t=d} + \widehat{s}_{\rho,h}^\gamma(\rho_t) \Big].$$

3.5.3 Random Forests

Random forests form a powerful learning method for classification and regression that constructs a collection of decision trees from training data and output, for each new data point, the mean prediction of the individual trees. Introduced by Breiman [2001], theses approaches operate well on many applications. Recent work demonstrates their efficiency in forecasting power consumption (see, among others Goehry et al., 2019 and Fan and Hyndman, 2011). A random forest is made up of a set $\big(T_k^\gamma\big)_{1\leqslant k\leqslant K}$ of decision trees grown in the following way (see Breiman et al., 1984 for further details). For each $k = 1, \ldots, K$, we first randomly draw, with replacement, n points from the training data set and start at the root, that contains all the points of the sub-sample. At each node $\mathcal{N}$ with more than m data points, V variables are randomly selected among the exogenous variables. Given a variable $v \in V$ and a threshold s, each point of the node $\mathcal{N}$ is assigned to the left daughter node $\mathcal{N}_L$ if its value in v is lower than s or to the right daughter node $\mathcal{N}_R$ otherwise. Considering only these V variables, the best split – given by a pair (v, s) of variable and an associated threshold – to separate the points into two set $\mathcal{N}_L$ and $\mathcal{N}_R$ is determined by minimizing the variance criterion indicated below. For any node $\mathcal{N}$ let us define the variance $\mathrm{Var}(\mathcal{N})$ by

$$\mathrm{Var}(\mathcal{N}) \stackrel{\text{def}}{=} \frac{1}{|\mathcal{N}|} \sum_{i\in\mathcal{N}} \big(y_i^\gamma - \bar{y}_\mathcal{N}^\gamma\big)^2, \quad \text{with} \quad \bar{y}_\mathcal{N}^\gamma \stackrel{\text{def}}{=} \frac{1}{|\mathcal{N}|} \sum_{i\in\mathcal{N}} y_i^\gamma.$$

Each node $\mathcal{N}$ is split in the two daughter nodes $\mathcal{N}_R^\star$ and $\mathcal{N}_L^\star$ (determined by the choice of v and s) minimizing the following criterion

$$\big(\mathcal{N}_R^\star, \mathcal{N}_L^\star\big) \in \operatorname*{argmin}_{\mathcal{N}_R, \mathcal{N}_L} \frac{|\mathcal{N}_R|}{n} \mathrm{Var}\big(\mathcal{N}_R\big) + \frac{|\mathcal{N}_L|}{n} \mathrm{Var}\big(\mathcal{N}_L\big). \tag{3.6}$$

Thus, we create a binary test to split the points of the node. When all the leaves contain fewer than m points, we associate with each leaf the mean of its data points. For a new point, we look at the values of its variables. For each $k = 1, \ldots, K$, we browse the tree T_k^γ and predict the value of the corresponding leaf. The K resulting forecasts are then averaged out. Algorithm 1 describes the above procedure and is implemented in the `ranger` R-package. In the experiments of Section 3.7, we take n equal to the number of data points in the training set, $m = 5$ and $K = 500$ (default parameters of `ranger`). The number V has been optimized by grid search; what we obtained is that, for each node, we keep two-thirds of the variables to split it (these variables are the same as the ones described in the previous section).

Algorithm 1 Random Forest for Regression

Parameters
Number of trees K
Sample size n
Minimal node size m
Number of variables to possibly split at in each node V
for $k = 1, \dots, K$ **do**
Draw a sample (with replacement) of size n from training data
Construct the tree T_k starting at the root with all the n data points
while a leaf contains more than m data points **do**
for each leaf of more than m data points **do**
Select V variables
Split the node into two nodes using the variance criterion (3.6) among the chosen variables
Output $\left(T_k^\gamma\right)_{1\leqslant k\leqslant K}$
Prediction at a new data point
Mean of the K forecasts output by the trees $\left(T_k^\gamma\right)_{1\leqslant k\leqslant K}$

With $\left(T_k\right)_{1\leqslant k\leqslant K}$, the trees constructed by Algorithm 1 run on a training data set, the forecast of any node $\gamma \in \Gamma$, at a new round t, is then

$$x_t^\gamma = \frac{1}{K}\sum_{k=1}^{K} T_k^\gamma\Big(y_{t-7\times 48}^\gamma, y_{t-48}^\gamma, \tau_t^\gamma, \bar{\tau}_t^\gamma, \nu_t^\gamma, \kappa_t^\gamma, \rho_t, d_t, h_t\Big).$$

3.6 Aggregation Algorithms

This section describes the three aggregation algorithms we use in the experiments of Section 3.7. At an instance t, for a node $\gamma \in \Gamma$, a copy $\mathcal{A}^\gamma$ of an aggregation algorithm $\mathcal{A}$ takes the feature vector $\mathbf{x}_t$ (generated with one of methods of the previous section) as an input and outputs the forecast $\widehat{y}_t^\gamma = \mathbf{u}_t^\gamma \cdot \mathbf{x}_t$. Therefore, to forecast the node γ, we use $\mathbf{x}_t$, which contains the predictions of all the nodes (including that of the considering node). We remind that the features $\left(x_t^\gamma\right)_{\gamma\in\Gamma}$ are generated independently with possibly different exogenous variables but that the observations $\left(y_t^\gamma\right)_{\gamma\in\Gamma}$ may be strongly correlated. This is why we consider aggregation to refine some forecasts by combining the features. Our experiments demonstrate that this aggregation step improves the forecasts. Section 3.6.1 presents a trick to empirically standardize the features and the observations first. On the one hand, this preprocessing justifies boundedness assumptions (3.7) on observations and features, that ensure some theoretical guaranties of the form requested by Notation 3.1. On the other hand, this preprocessing simplifies hyper-parameters search (for the aggregation step) as we can choose the same for every series since they have similar statistics (scale and variance). Following Sections 3.6.2 and 3.6.3 introduce the aggregation algorithms and some technical tricks implemented in the experiments of Section 3.7.

3.6.1 Standardization

In empirical machine learning, it is known that standardizing observations and features may significantly improve results, and sequential learning is no exception (see Chapter 2). In addition, standardization makes the calibration of the parameters of the algorithm common to all the nodes, namely for each algorithm $\mathcal{A}^\gamma$, we choose the hyper-parameters $\mathbf{s}_0^\gamma = \breve{\mathbf{s}}_0$. We can do so, because thanks to the preprocessing below, features and observations will be of the same order. Let us fix $\gamma \in \Gamma$ and $t > 0$. We consider the following transformations, relying on statistics S^γ and $\underline{\breve{\mathbf{E}}}$ computed on T_0 historical time steps:

$$
\begin{aligned}
y_t^\gamma \to \breve{y}_t^\gamma &\overset{\text{def}}{=} \frac{y_t^\gamma - x_t^\gamma}{S^\gamma} && \text{Observations tranform} \\
\mathbf{x}_t \to \breve{\mathbf{x}}_t &\overset{\text{def}}{=} \underline{\breve{\mathbf{E}}}\mathbf{x}_t && \text{Features transform}
\end{aligned}
$$

$$
\text{with} \quad S^\gamma = \max_{1-T_0 \leqslant t \leqslant 0} |y_t^\gamma - x_t^\gamma| \qquad \text{and} \qquad \underline{\breve{\mathbf{E}}} \overset{\text{def}}{=} \left(\frac{1}{T_0} \sum_{t=1-T_0}^{0} \mathbf{x}_t \, \mathbf{x}_t^{\mathrm{T}} \right)^{-1/2} .
$$

We thus assume that the Gram matrix $\frac{1}{T_0} \sum_{t=1-T_0}^{0} \mathbf{x}_t \mathbf{x}_t^{\mathrm{T}}$ is invertible, which is a reasonable assumption as soon as T_0 is large enough. Our standardization process differs from the usual methods (see details below) but it provides the theoretical guaranties set out below. Furthermore, it makes sense for the following reasons. Fixing $\gamma \in \Gamma$, when features and observations are bounded, S^γ is an estimation of a bound on $y_t^\gamma - x_t^\gamma$. The re-scaling of $(y_t^\gamma - x_t^\gamma)$ by S^γ should provide transformed observations lying in $[-1, 1]$ or a some neighboring range. It also reduces and homogenizes the variances for all the nodes. A simple example may illustrate this variance reduction. For deterministic features, the variance of non-transformed observations satisfy $\mathrm{Var}\Big(y_t^\gamma\Big) = \mathrm{Var}\Big(y_t^\gamma - x_t^\gamma\Big)$. The variance of standardized observations is then divided by $\Big(S^\gamma\Big)^2$ and we have $\mathrm{Var}\Big(\breve{y}_t^\gamma\Big) = \mathrm{Var}\Big(y_t^\gamma\Big) / \Big(S^\gamma\Big)^2$. For T_0 large enough, the variance of transformed observations should be less than 1. Indeed, with high probability, the maximum of the absolute values of the random variable $(y_t^\gamma - x_t^\gamma)$ on $t = 1 - T_0, \ldots, 0$ (which is S^γ), is higher than its standard deviation $\sqrt{\mathrm{Var}\Big(y_t^\gamma\Big)}$ and thus $\Big(S^\gamma\Big)^2 > \mathrm{Var}\Big(y_t^\gamma\Big)$. Moreover, the expectation of $(y_t^\gamma - x_t^\gamma)$ should be close to 0 as soon as the features are correctly generated. Indeed, the more the benchmark forecast are relevant, the more the observations are re-centered. Concerning the features, our standardization is classic in the case of centered features. The matrix $\underline{\breve{\mathbf{E}}}^2$ would then be an estimation of the inverse of the co-variance matrix of vectors $\mathbf{x}_t$, and the multiplication of the features by $\underline{\breve{\mathbf{E}}}$ would provide transformed features whose co-variance matrix is close to the identity matrix. Here, we do not recenter observations and features with some empirical mean as it is classically done (this would be unconvenient for our regret analysis). Anyway, Section 3.7.3 provides some experimental results which confirm that our preprocessing standardizes reasonably well observations and features. Moreover, we tested classical standardization (with re-centering) on features and obtained results similar to those presented in Section 3.7 (but, as hinted at above, no theoretical guaranties would be associated with this classical standardization).

We run Algorithm $\mathcal{A}^\gamma$ on transformed features and observations with the initialization parameter vector $\breve{\mathbf{s}}_0$ (which does not depend on γ) and obtain a standardized prediction at node γ, denoted by $\bar{y}_t^\gamma$. Then, we transform this output to get the (non-standardized) forecast

$$\widehat{y}_t^\gamma \overset{\text{def}}{=} S^\gamma \bar{y}_t^\gamma + x_t^\gamma.$$

For any vector $\breve{\mathbf{u}}^\gamma \in \mathbb{R}^{|\Gamma|}$, we introduce the standardized regret associated with transformed observations and features, denoted by $\breve{\mathcal{R}}_T^\gamma(\breve{\mathbf{u}}^\gamma)$ as:

$$\begin{aligned}
\breve{\mathcal{R}}_T^\gamma(\breve{\mathbf{u}}^\gamma) &\overset{\text{def}}{=} \sum_{t=1}^T \left(\breve{y}_t^\gamma - \bar{y}_t^\gamma\right)^2 - \sum_{t=1}^T \left(\breve{y}_t^\gamma - \breve{\mathbf{u}}^\gamma \cdot \breve{\mathbf{x}}_t\right)^2 \\
&= \sum_{t=1}^T \left(\frac{y_t^\gamma - x_t^\gamma}{S^\gamma} - \frac{\widehat{y}_t^\gamma - x_t^\gamma}{S^\gamma}\right)^2 - \sum_{t=1}^T \left(\frac{y_t^\gamma - x_t^\gamma}{S^\gamma} - \breve{\mathbf{u}}^\gamma \cdot \left(\underline{\breve{\mathbf{E}}}\mathbf{x}_t\right)\right)^2 \\
&= \frac{1}{\left(S^\gamma\right)^2} \sum_{t=1}^T \left(y_t^\gamma - \widehat{y}_t^\gamma\right)^2 - \frac{1}{\left(S^\gamma\right)^2} \sum_{t=1}^T \left(y_t^\gamma - \underbrace{\left(x_t^\gamma + S^\gamma\left(\underline{\breve{\mathbf{E}}}\breve{\mathbf{u}}^\gamma\right) \cdot \mathbf{x}_t\right)}_{\mathbf{u}^\gamma \cdot \mathbf{x}_t}\right)^2.
\end{aligned}$$

In the equations above, we define $\mathbf{u}^\gamma \overset{\text{def}}{=} \boldsymbol{\delta}^\gamma + S^\gamma\left(\underline{\breve{\mathbf{E}}}\breve{\mathbf{u}}^\gamma\right)$ where $\boldsymbol{\delta}^\gamma \overset{\text{def}}{=} \left(\mathbf{1}_{\{i=\gamma\}}\right)_{i\in\Gamma}$ denotes the standard basis vector that points in the γ direction. Equivalently, $\breve{\mathbf{u}}^\gamma = \underline{\breve{\mathbf{E}}}^{-1}(\mathbf{u}^\gamma - \boldsymbol{\delta}^\gamma)/S^\gamma$, so there is a bijective correspondence between the vectors $\mathbf{u}^\gamma$ and $\breve{\mathbf{u}}^\gamma$. Therefore, by noticing that $x_t^\gamma = \boldsymbol{\delta}^\gamma \cdot \mathbf{x}_t$, the regret associated with original features and observations is related to the regret of transformed data by the following equation:

$$\breve{\mathcal{R}}_T^\gamma(\breve{\mathbf{u}}^\gamma) = \frac{1}{\left(S^\gamma\right)^2} \sum_{t=1}^T \left(y_t^\gamma - \widehat{y}_t^\gamma\right)^2 - \frac{1}{\left(S^\gamma\right)^2} \sum_{t=1}^T \left(y_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t\right)^2 = \frac{\mathcal{R}_T^\gamma(\mathbf{u}^\gamma)}{\left(S^\gamma\right)^2}.$$

Furthermore, as for any $\breve{\mathbf{u}} \in \mathbb{R}^{|\Gamma|}$, Notation 3.1 ensures

$$\breve{\mathcal{R}}_T^\gamma(\breve{\mathbf{u}}^\gamma) = \sum_{t=1}^T \left(\breve{y}_t^\gamma - \bar{y}_t^\gamma\right)^2 - \sum_{t=1}^T \left(\breve{y}_t^\gamma - \breve{\mathbf{u}}^{\gamma\intercal}\breve{\mathbf{x}}_t\right)^2 \leqslant \mathrm{B}\left(\breve{\mathbf{x}}_{1:T}, \breve{y}_{1:T}^\gamma, \breve{\mathbf{s}}_0, \breve{\mathbf{u}}^\gamma\right),$$

Combining the two previous equations yields the following proposition.

Proposition 3.3. *For any $\gamma \in \Gamma$ and any $\mathbf{u}^\gamma \in \mathbb{R}^{|\Gamma|}$, if Notation 3.1 holds for Algorithm $\mathcal{A}^\gamma$ run on transformed observations and features $\breve{y}_{1:T}^\gamma$ and $\breve{\mathbf{x}}_{1:T}$, with the initialization parameter vector $\breve{\mathbf{s}}_0$, we have, for $T > 0$,*

$$\mathcal{R}_T^\gamma\left(\mathbf{u}^\gamma\right) \leqslant \left(S^\gamma\right)^2 \mathrm{B}\left(\breve{\mathbf{x}}_{1:T}, \breve{y}_{1:T}^\gamma, \breve{\mathbf{s}}_0, \breve{\mathbf{u}}^\gamma\right) \quad \text{where} \quad \breve{\mathbf{u}}^\gamma = \underline{\breve{\mathbf{E}}}^{-1}(\mathbf{u}^\gamma - \boldsymbol{\delta}^\gamma)/S^\gamma.$$

Throughout the section, without loss of generality and to simplify the notation, we now replace the features and observations with the standardized ones. Thus, we will write y_t^γ for $\breve{y}_t^\gamma$, $\mathbf{x}_t$ for $\breve{\mathbf{x}}_t$ and so on. Moreover, we make the following assumption on the boundedness of features and observations. - Boundedness assumptions. For any $t > 0$ and any $\gamma \in \Gamma$ we assume that there is a constant $C > 0$ such that

$$|y_t^\gamma| \leqslant C \quad \text{and} \quad |x_t^\gamma| \leqslant C. \tag{3.7}$$

Some boudedness assumptions on features and observations are frequently required to establish theoretical guaranties. Here, the constant is common to all the nodes. Practically, this assumption makes sens because of the previous transformations. As explained above, it centers and normalizes observations and features. Section 3.7.3 presents statistics on features and observation before and after standardization and indicates possible values of the constant C.

In the two next subsections, we introduce the aggregation algorithms we implemented in Section 3.7. We recall that, for any $\gamma \in \Gamma$, at a round t, the algorithm $\mathcal{A}^\gamma$ provides a weight vector $\mathbf{u}_t^\gamma$ and thus forecasts y_t^γ with $\mathbf{u}_t^\gamma \cdot \mathbf{x}_t$. In Section 3.6.2, we consider a linear aggregation algorithm: there is no restriction on the computed weight vectors. In Section 3.6.3, the two algorithms output convex combinations of features: the weight vectors are in the $|\Gamma|$-simplex denoted by $\Delta_{|\Gamma|}$. However, there is no reason to consider such a restriction and this is why the last paragraph of the subsection presents a trick to extend the previous algorithms to output linear combinations of features for which the weight vectors are in a L_1-ball. Thus, there are no longer restrictions on the sum or the sign of the weights.

3.6.2 Linear Aggregation: Sequential Non-Linear Ridge Regression

The first aggregation algorithm that we consider is the sequential non-linear ridge regression of Vovk [2001] and Azoury and Warmuth [2001]. So, for any $\gamma \in \Gamma$, Algorithm $\mathcal{A}^\gamma$ refers here to Algorithm 2 run with regularization parameter $\mathbf{s}_0 = \lambda$. For any instance $t \geqslant 2$, this algorithm, chooses vectors $\mathbf{u}_t^\gamma$ as follow:

$$\mathbf{u}_t^\gamma \in \operatorname*{argmin}_{\mathbf{u}^\gamma \in \mathbb{R}^d} \sum_{s=1}^{t-1} \Big(y_s^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_s\Big)^2 + \Big(\mathbf{u}^\gamma \cdot \mathbf{x}_t\Big)^2 + \lambda \|\mathbf{u}^\gamma\|^2. \tag{3.8}$$

The solution of this minimization problem is given by:

$$\mathbf{u}_t^\gamma = \Big(\lambda \Big(\mathbf{1}_{\{i=j\}}\Big)_{(i,j)\in\Gamma^2} + \sum_{s=1}^{t} \mathbf{x}_s \mathbf{x}_s^{\mathrm{T}}\Big)^\dagger \sum_{s=1}^{t-1} y_s^\gamma \mathbf{x}_s,$$

where $A^\dagger$ denotes the pseudo-inverse of the matrix A. Algorithm 2 provides a sequential implementation of the solution of this convex minimization problem.

Algorithm 2 Non-Linear Sequential Ridge Regression

aim
Predict the time series $\Big(y_t^\gamma\Big)_{1\leqslant t\leqslant T}$
parameter Regularization parameter $\lambda > 0$
initialization $\mathbf{A}_0 = \lambda\Big(\mathbf{1}_{\{i=j\}}\Big)_{(i,j)\in\Gamma^2}$ and $\mathbf{b}_0 = (0,\dots,0)^{\mathrm{T}}$
for $t = 1,\dots,T$ **do**
Update matrix $\mathbf{A}_t = \mathbf{A}_{t-1} + \mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}$
Compute the vector $\mathbf{u}_t^\gamma = \mathbf{A}_t^{-1}\mathbf{b}_{t-1}$
Output prediction $\widehat{y}_t^\gamma = \mathbf{u}_t^\gamma \cdot \mathbf{x}_t$
Update vector $\mathbf{b}_t = \mathbf{b}_{t-1} + y_t^\gamma \mathbf{x}_t$

The above non-linear ridge regression is a penalized ordinary least-squares regression. Since the features may be strongly correlated, the least squares estimator, $\mathbf{u}_t^\gamma \in \operatorname{argmin}_{\mathbf{u}^\gamma \in \mathbb{R}^d} \sum_{s=1}^{t-1} \left(y_s^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_s\right)^2$, could lead to very large prediction if a new features vector belongs to an eigenspace of the empirical gram matrix associated to a small value. The regularization term $\lambda\|\mathbf{u}^\gamma\|^2$ ensures that eigenvalues of the empirical gram matrix are not too small. We then add the regularization term $\left(\mathbf{u}^\gamma \cdot \mathbf{x}_t\right)^2$ which is the last term of the cumulative prediction error $\left(y_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t\right)^2$ where we have replaced unknown y_t^γ by our best guess 0. It is known to improve the regret bound (see Vovk, 2001 and Chapter 2). In our case (standardized targets), it particularly makes sense because it biases predictions towards 0; which, because of the standardization, biases aggregated predictions towards benchmark predictions.

Under the boundedness assumptions (3.7), for any vector $\mathbf{u}^\gamma \in \mathbb{R}^{|\Gamma|}$, with the algorithm $\mathcal{A}^\gamma$ set to the non-linear ridge regression (3.8) run with regularization parameter λ, Theorem 11.8 of the monograph *Prediction, Learning, and Games* by Cesa-Bianchi and Lugosi [2006] or Theorem 2.2 of Chapter 2 provide the following theoretical guaranties:

$$\mathcal{R}_T^\gamma(\mathbf{u}^\gamma) \leqslant \lambda\|\mathbf{u}^\gamma\|^2 + |\Gamma|C^2 \ln\left(1 + \frac{C^2 T}{\lambda}\right).$$

So, for any $\mathbf{U} = \left(\mathbf{u}^1 | \ldots | \mathbf{u}^{|\Gamma|}\right) \in \mathcal{C}$, as $\left\|\mathbf{U}\right\|_F^2 = \sum_{\gamma\in\Gamma} \|\mathbf{u}^\gamma\|^2$, Theorem 3.2 ensures

$$\mathcal{R}_T(\mathbf{U}) = \sum_{\gamma\in\Gamma} \mathcal{R}_T^\gamma(\mathbf{u}^\gamma) \leqslant \lambda\left\|\mathbf{U}\right\|_F^2 + |\Gamma|^2 C^2 \ln\left(1 + \frac{C^2 T}{\lambda}\right) = \mathcal{O}\Big(|\Gamma|^2 \ln T\Big).$$

That is, since the sequential non-linear ridge regression provides a logarithmic regret bound, Meta-algorithm 1 achieves a bound of the same order.

3.6.3 Convex Aggregation

We focus here on uniform bounds and use notation introduced in Remark 8. The following two algorithms were initially designed to compete against the best feature. Namely, for a node $\gamma \in \Gamma$, the Bernstein online aggregation (BOA, see Wintenberger, 2017) and polynomially weighted average forecaster with multiple learning rates (ML-Pol, see Gaillard, 2015) provide some bound on the difference between the cumulative prediction error $L_T^\gamma \stackrel{\text{def}}{=} \sum_{t=1}^T \left(y_t^\gamma - \widehat{y}_t^\gamma\right)^2$ of the strategy and $\min_{i\in\Gamma} \sum_{t=1}^T \left(y_t^\gamma - x_t^i\right)^2$. At each instance t, both strategies compute weight vector $\mathbf{u}_t^\gamma = \left(u_t^{\gamma\, i}\right)_{i\in\Gamma}$ based on historical data. These vectors are in the $|\Gamma|$-simplex, which we denote by $\Delta_{|\Gamma|}$. For each feature $i \in \Gamma$, the weight $u_t^{\gamma\, i}$ is, for BOA, an exponential function of a regularized cumulative prediction error of the feature x_t^i and, for ML-Pol, a polynomial function of the cumulative prediction error of x_t^i. However, by using gradients of prediction errors instead of the original prediction errors the average error of these algorithms may come close to $\min_{\mathbf{u}^\gamma\in\Delta_{|\Gamma|}} \frac{1}{T}\sum_{t=1}^T \left(y_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t\right)^2$. This "gradient trick" (see Cesa-Bianchi and Lugosi, 2006, Section 2.5) is presented in the next paragraph and is already integrated in the statements of the algorithms below. Moreover, for both algo-

rithms, the computed weight vectors are in $\Delta_{|\Gamma|}$. As we do not necessarily want to impose such a restriction, we use another trick, introduced by Kivinen and Warmuth [1997] and presented in the last paragraph. It extends the class of comparison from the $|\Gamma|$-simplex to an L_1-ball of radius α denoted by $\mathcal{B}_\alpha \stackrel{\text{def}}{=} \Big\{\mathbf{u}^\gamma \in \mathbb{R}^{|\Gamma|} \,\Big|\, \|\mathbf{u}\|_1 = \sum_{i\in\Gamma} |u^{\gamma\, i}| \leqslant \alpha\Big\}$. The aim is then to come close to the cumulative error $\min_{\mathbf{u}^\gamma\in\mathcal{B}_\alpha} \sum_{t=1}^T \Big(y_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t\Big)^2$.

Gradient Trick: from the best feature to the best convex combination of features. We consider an aggregation algorithm that takes as input, at any time step $t+1$, the previous prediction errors of each feature $(y_t^\gamma - x_t^i)^2$, for any $i \in \Gamma$, and that of the forecast outputted at t: $(y_t^\gamma - \widehat{y}_t^\gamma)^2$. Although this trick generalizes to various prediction errors, we focus here to its application in our case, namely the quadratic prediction error. We assume that the algorithm provides a bound on the quantity (see notation of Remark 8)

$$\mathcal{R}_T^\gamma\Big(\delta_{|\Gamma|}\Big) \stackrel{\text{def}}{=} \sum_{t=1}^T (y_t^\gamma - \widehat{y}_t^\gamma)^2 - \min_{i\in\Gamma} \sum_{t=1}^T (y_t^\gamma - x_t^i)^2\,.$$

where $\delta_{|\Gamma|} \stackrel{\text{def}}{=} \Big\{(\boldsymbol{\delta}^i)_{i\in\Gamma}\Big\}$ is the set of canonical basis vectors (so we have $\boldsymbol{\delta}^i \cdot \mathbf{x}_t = x_t^i$). The gradient trick consists in giving, instead of the prediction errors $(y_t^\gamma - \widehat{y}_t^\gamma)^2$ and $(y_t^\gamma - x_t^i)^2$, for any $i \in \Gamma$, the pseudo prediction errors functions defined below as input to algorithm $\mathcal{A}^\gamma$. This will provide a bound on the pseudo regret denoted by $\widetilde{\mathcal{R}}_T^\gamma\Big(\delta_{|\Gamma|}\Big)$. We will prove that the same bound is achieved for the minimum of $\mathcal{R}_T^\gamma(\mathbf{u}^\gamma)$ taken over $\mathbf{u}^\gamma \in \Delta_{|\Gamma|}$ (and not only $\delta_{|\Gamma|}$), namely $\mathcal{R}_T^\gamma\Big(\Delta_{|\Gamma|}\Big)$. We detail here how the trick works and gives:

$$\mathcal{R}_T^\gamma\Big(\Delta_{|\Gamma|}\Big) \leqslant \widetilde{\mathcal{R}}_T^\gamma\Big(\delta_{|\Gamma|}\Big).$$

Let us fix a vector $\mathbf{u}^\gamma = \Big(u^{\gamma\, i}\Big)_{i\in\Gamma} \in \Delta_{|\Gamma|}$, we have for each $t = 1, \ldots, T$

$$\begin{aligned}
\Big(y_t^\gamma - \widehat{y}_t^\gamma\Big)^2 - \Big(y_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t\Big)^2 &= \Big(2y_t^\gamma - \widehat{y}_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t\Big)\Big(\mathbf{u}^\gamma \cdot \mathbf{x}_t - \widehat{y}_t^\gamma\Big) \\
&= 2\Big(\widehat{y}_t^\gamma - y_t^\gamma\Big)\Big(\widehat{y}_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t\Big) - \Big(\widehat{y}_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t\Big)^2 \\
&\leqslant 2\Big(\widehat{y}_t^\gamma - y_t^\gamma\Big)\Big(\widehat{y}_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t\Big)\,. \qquad (3.9)
\end{aligned}$$

By plugging this equation into the definition of the regret, we obtain

$$\begin{aligned}
\mathcal{R}_T^\gamma\Big(\mathbf{u}^\gamma\Big) &\stackrel{\text{def}}{=} \sum_{t=1}^T (y_t^\gamma - \widehat{y}_t^\gamma)^2 - \sum_{t=1}^T (y_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t)^2 \\
&\stackrel{(3.9)}{\leqslant} \sum_{t=1}^T 2\Big(\widehat{y}_t^\gamma - y_t^\gamma\Big)\Big(\widehat{y}_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t\Big) = \sum_{t=1}^T 2\Big(\widehat{y}_t^\gamma - y_t^\gamma\Big)\widehat{y}_t^\gamma - \sum_{t=1}^T \sum_{i\in\Gamma} u^{\gamma\, i} 2\Big(\widehat{y}_t^\gamma - y_t^\gamma\Big)x_t^i.
\end{aligned}$$

As $\mathbf{u}^\gamma$ belongs to the $|\Gamma|$-simplex (so $\forall i \in \Gamma$, $u^{\gamma\, i} \geqslant 0$ and $\sum_{i\in\Gamma} u^{\gamma\, i} = 1$), we get:

$$\sum_{i\in\Gamma} u^{\gamma\, i} x_t^i \;\geqslant\; \min_{j\in\Gamma} x_t^j \sum_{i\in\Gamma} u^{\gamma\, i} = \min_{j\in\Gamma} x_t^j$$

Therefore, for any vector $\mathbf{u}^\gamma \in \Delta_{|\Gamma|}$, the regret $\mathcal{R}_T^\gamma\big(\mathbf{u}^\gamma\big)$ is bounded by

$$\mathcal{R}_T^\gamma\big(\mathbf{u}^\gamma\big) \leqslant \sum_{t=1}^{T} 2\big(\widehat{y}_t^\gamma - y_t^\gamma\big)\widehat{y}_t^\gamma - \min_{j\in\Gamma} \sum_{t=1}^{T} 2\big(\widehat{y}_t^\gamma - y_t^\gamma\big)x_t^j \stackrel{\text{def}}{=} \widetilde{\mathcal{R}}_T^\gamma\big(\delta_{|\Gamma|}\big).$$

Thus, we now give the pseudo prediction errors associated with each feature $2\big(\widehat{y}_t^\gamma - y_t^\gamma\big)x_t^i$, with $i \in \Gamma$, and with the outputted forecast $2\big(\widehat{y}_t^\gamma - y_t^\gamma\big)\widehat{y}_t^\gamma$ as input to algorithm $\mathcal{A}^\gamma$. It provides a bound on the pseudo regret defined above $\widetilde{\mathcal{R}}_T^\gamma\big(\delta_{|\Gamma|}\big)$; and we get the same bound on $\mathcal{R}_T^\gamma\big(\Delta_{|\Gamma|}\big)$. As a final note, we emphasize that the boundedness assumptions (3.7) allow to establish that pseudo prediction errors $2(\widehat{y}_t^\gamma - y_t^\gamma)x_t^i$ are bounded by $4C^2$. Indeed, for any $(\gamma, i) \in \Gamma^2$, they ensure $|y_t^\gamma| \leqslant C$ and $|x_t^i| \leqslant C$. In addition, as $\mathbf{u}_t^\gamma \in \Delta_\Gamma$, the output forecasts $\widehat{y}_t^\gamma = \mathbf{u}_t^\gamma \cdot \mathbf{x}_t$ are also bounded by:

$$|\widehat{y}_t^\gamma| = \left|\sum_{j\in\Gamma} u_t^{\gamma\,j} x_t^j\right| \leqslant \sum_{j\in\Gamma} u_t^{\gamma\,j}\big|x_t^j\big| \leqslant \sum_{j\in\Gamma} u_t^{\gamma\,j} C = C.$$

Hence, for any $i \in \Gamma$, the pseudo prediction error associated with feature i satisfies

$$\big|2\big(\widehat{y}_t^\gamma - y_t^\gamma\big)x_t^i\big| \leqslant 4C^2. \tag{3.10}$$

Algorithm 3 Fully adaptive Bernstein Online Aggregation (BOA) with gradient trick

aim
Predict the time series $\big(y_t^\gamma\big)_{1\leqslant t\leqslant T}$
parameter Bound on pseudo prediction errors E:
for any $t = 1, \ldots, T$ and any $i \in \Gamma$, $|2(\widehat{y}_t^\gamma - y_t^\gamma)x_t^i| \leqslant E$
initialization
$\mathbf{u}_1^\gamma = (1/|\Gamma|, \ldots, 1/|\Gamma|)$
$\widehat{y}_1^\gamma = \mathbf{u}_1^\gamma \cdot \mathbf{x}_1$
For $i \in \Gamma$, $\widetilde{\mathcal{R}}_0^{\gamma\,i} = 0$
For $i \in \Gamma$, $\eta_0^{\gamma\,i} = 0$
for $t = 1, \ldots, T-1$ **do**
For each $i \in \Gamma$, update the cumulative quantity $\widetilde{Q}^{\gamma\,i}$ for feature i

$$\widetilde{Q}_t^{\gamma\,i} = \widetilde{Q}_{t-1}^{\gamma\,i} + \widetilde{r}_t^{\gamma\,i}\big(1 + \eta_{t-1}^{\gamma\,i}\widetilde{r}_t^{\gamma\,i}\big) \quad \text{where} \quad \widetilde{r}_t^{\gamma\,i} \stackrel{\text{def}}{=} 2(\widehat{y}_t^\gamma - y_t^\gamma)(\widehat{y}_t^\gamma - x_t^i)$$

For each $i \in \Gamma$, compute the learning rate

$$\eta_t^{\gamma\,i} = \min\left\{\frac{1}{2E}, \sqrt{\frac{\ln|\Gamma|}{\sum_{s=1}^t \big(\widetilde{r}_t^{\gamma\,i}\big)^2}}\right\}$$

Compute the weight vector $\mathbf{u}_{t+1}^\gamma = (u_{t+1}^{\gamma\,i})_{i\in\Gamma}$ defined as

$$u_{t+1}^{\gamma\,i} = \frac{\exp\big(\eta_t^{\gamma\,i}\widetilde{Q}_t^{\gamma\,i}\big)}{\sum_{j\in\Gamma}\exp\big(\eta_t^{\gamma\,j}\widetilde{Q}_t^{\gamma\,j}\big)}$$

Output prediction $\widehat{y}_{t+1}^\gamma = \mathbf{u}_{t+1}^\gamma \cdot \mathbf{x}_{t+1} = \sum_{i\in\Gamma} u_{t+1}^{\gamma\,i} x_{t+1}^i$

Bernstein Online Aggregation. Wintenberger [2017] introduces an aggregation procedure called Bernstein Online Aggregation for which weights are exponential function of the cumulative prediction errors. Algorithm 3 describes this strategy combined with this gradient trick. Let us fix a node $\gamma \in \Gamma$ and set $\mathcal{A}^\gamma$ to Algorithm 3 which takes as input the bound E on pseudo prediction errors ($E = 4C^2$ is a suitable choice):

$$\forall t = 1, \dots, T, \forall i \in \Gamma, \quad 2(\widehat{y}_t^\gamma - y_t^\gamma)x_t^i \leqslant E.$$

Theorem 3.4 of Wintenberger, 2017 ensures that

$$\begin{aligned}\mathcal{R}_T^\gamma(\Delta_{|\Gamma|}) &\leqslant \sqrt{T+1}E\left(\frac{\sqrt{2\ln|\Gamma|}}{\sqrt{2}-1} + \frac{\ln(1+\ln T/2)}{\sqrt{\ln|\Gamma|}}\right) + E\Big(2\ln|\Gamma| + 2\ln(1+\ln T/2) + 1\Big) \\ &\lesssim \mathcal{O}\Big(\sqrt{T}\ln\ln T\Big). \end{aligned} \tag{3.11}$$

Thanks to Equation (3.10), we replace E by $4C^2$ in Equation (3.11) and we get, for each node $\gamma \in \Gamma$, an upper bound on $\mathcal{R}_T^\gamma(\Delta_{|\Gamma|})$. By applying Theorem 3.2, we obtain the following uniform regret bound:

$$\mathcal{R}_T(\Delta_{|\Gamma|}) \lesssim \mathcal{O}\Big(|\Gamma|\sqrt{T}\ln\ln T\Big),$$

which is of order $\sqrt{T}$ (up to poly-logarithmic terms).

Algorithm 4 Polynomially weighted average forecaster with Multiple Learning rates (ML-Pol) and gradient trick

aim
Predict the time series $\left(y_t^\gamma\right)_{1\leqslant t\leqslant T}$
parameter Bound on pseudo prediction errors E:
for any $t = 1, \dots, T$ and any $i \in \Gamma$, $|2(\widehat{y}_t^\gamma - y_t^\gamma)x_t^i| \leqslant E$
initialization
$\mathbf{u}_1^\gamma = (1/|\Gamma|, \dots, 1/|\Gamma|)$
$\widehat{y}_1^\gamma = \mathbf{u}_1^\gamma \cdot \mathbf{x}_1$
For $i \in \Gamma$, $\widetilde{\mathcal{R}}_0^{\gamma\, i} = 0$
For $i \in \Gamma$, $\eta_0^{\gamma\, i} = 0$
for $t = 1, \dots, T-1$ **do**
For each $i \in \Gamma$, update the cumulative pseudo-regret of feature i

$$\widetilde{\mathcal{R}}_t^{\gamma\, i} = \widetilde{\mathcal{R}}_{t-1}^{\gamma\, i} + \widetilde{r}_t^{\gamma\, i} \quad \text{where} \quad \widetilde{r}_t^{\gamma\, i} \stackrel{\text{def}}{=} 2(\widehat{y}_t^\gamma - y_t^\gamma)(\widehat{y}_t^\gamma - x_t^i)$$

For each $i \in \Gamma$, compute the learning rate

$$\eta_t^{\gamma\, i} = \left(E + \sum_{s=1}^{t}\left(\widetilde{r}_t^{\gamma\, i}\right)^2\right)^{-1}$$

Compute the weight vector $\mathbf{u}_{t+1}^\gamma = (u_{t+1}^{\gamma\, i})_{i\in\Gamma}$ defined as

$$u_{t+1}^{\gamma\, i} = \frac{\eta_t^{\gamma\, i}\left(\widetilde{\mathcal{R}}_t^{\gamma\, i}\right)_+}{\sum_{j\in\Gamma}\eta_t^{\gamma\, j}\left(\widetilde{\mathcal{R}}_t^{\gamma\, j}\right)_+}$$

Output prediction $\widehat{y}_{t+1}^\gamma = \mathbf{u}_{t+1}^\gamma \cdot \mathbf{x}_{t+1} = \sum_{i\in\Gamma} u_{t+1}^{\gamma\, i} x_{t+1}^i$

Polynomially Weighted Average Forecaster. Gaillard et al. [2014] consider an aggregation method based on weights that are polynomial functions of the cumulative prediction errors. We use this procedure combined with the gradient trick and present it in Algorithm 4. In this description, $(\mathbf{x})_+$ denotes the vector of non-negative parts of the components of $\mathbf{x}$. With the same notation as in the previous paragraph, for any node $\gamma \in \Gamma$, Theorem 5 of Gaillard et al. [2014] provides the following regret bound:

$$\mathcal{R}_T^\gamma(\Delta_{|\Gamma|}) \leqslant E\sqrt{|\Gamma|(T+1)\Big(1+\ln(1+T)\Big)}. \tag{3.12}$$

With $E \leqslant 4C^2$ and by applying Theorem 3.2, we obtain an upper bound on the uniform regret $\mathcal{R}_T(\Delta_{|\Gamma|})$, which is also of order $\sqrt{T}$ (up to poly-logarithmic terms):

$$\begin{aligned}\mathcal{R}_T(\Delta_{|\Gamma|}) &\leqslant 4C^2|\Gamma|\sqrt{|\Gamma|(T+1)\Big(1+\ln(1+T)\Big)} \\ &\lesssim \mathcal{O}\Big(|\Gamma|^{3/2}\sqrt{T\ln T}\Big).\end{aligned}$$

3.6.4 A scheme to extend the class of comparison from the simplex to an L_1-ball

For the previous two algorithms, we obtained an upper bound on $\mathcal{R}_T^\gamma(\Delta_{|\Gamma|})$. However, there is no reason for the best linear combination of features to be convex. Algorithm 5 presents a trick introduced by Kivinen and Warmuth [1997] which extends the class of comparison from the $|\Gamma|$-simplex to an L_1-ball of radius $\alpha > 0$ denoted by $\mathcal{B}_\alpha$ and provides a bound on $\mathcal{R}_T^\gamma(\mathcal{B}_\alpha)$. Let us fix a node $\gamma \in \Gamma$. The trick consists in transforming, at each round t, the feature vector $\mathbf{x}_t$ into the $2|\Gamma|$-vector $\bar{\mathbf{x}}_t = (\alpha\mathbf{x}_t| - \alpha\mathbf{x}_t)$. The algorithm $\mathcal{A}^\gamma$ is then run with these new features and it outputs the weight vector $\bar{\mathbf{u}}_t^\gamma \in \Delta_{2|\Gamma|}$. Finally, a $|\Gamma|$-vector $\mathbf{u}_t^\gamma \in \mathcal{B}_\alpha$ is computed from $\bar{\mathbf{u}}_t^\gamma$ to provide the forecast $\mathbf{u}_t^\gamma \cdot \mathbf{x}_t = \bar{\mathbf{u}}_t^\gamma \cdot \bar{\mathbf{x}}_t$. We will actually see that we may associate any $|\Gamma|$-vector $\mathbf{u} \in \mathcal{B}_\alpha$ with a vector $\bar{\mathbf{u}} \in \Delta_{2|\Gamma|}$ such as $\bar{\mathbf{u}} \cdot \bar{\mathbf{x}}_t = \mathbf{u} \cdot \mathbf{x}_t$; the trick actually defines a surjection from $\Delta_{2|\Gamma|}$ to $\mathcal{B}_\alpha$. Thus, to compete against the best linear combination of features in $\mathcal{B}_\alpha$, it is enough to compete against the best convex combination of features $\bar{\mathbf{x}}_t$ in a lifted space (which we may achieve, thanks to algorithm $\mathcal{A}^\gamma$). We now give all the details on how this trick works and indicate its impact on the stated regret bounds. The following lemma introduces the surjection from $\Delta_{2|\Gamma|}$ to $\mathcal{B}_\alpha$, which is used in Algorithm 5.

Lemma 3.4. *For any real $\alpha > 0$, the following function ψ is a surjection from $\Delta_{2|\Gamma|}$ to $\mathcal{B}_\alpha$:*

$$\psi : \begin{array}{lcl} \Delta_{2|\Gamma|} & \longrightarrow & \mathcal{B}_\alpha \\ \bar{\mathbf{u}} = \Big(\bar{\mathbf{u}}^+ \,|\, \bar{\mathbf{u}}^-\Big) & \longmapsto & \alpha(\bar{\mathbf{u}}^+ - \bar{\mathbf{u}}^-), \end{array}$$

where the vector $\bar{\mathbf{u}} \in \Delta_{2|\Gamma|}$ is decomposed in the two $|\Gamma|$-vectors $\bar{\mathbf{u}}^+$ and $\bar{\mathbf{u}}^-$, which correspond to the $|\Gamma|$ first and the $|\Gamma|$ last coefficients of $\bar{\mathbf{u}}$, respectively.

By running Algorithm $\mathcal{A}^\gamma$ with transformed features $\bar{\mathbf{x}}_t \stackrel{\text{def}}{=} \Big(\alpha\mathbf{x}_t \,|\, -\alpha\mathbf{x}_t\Big)$ and parameter s_0^γ (which provides weight vectors $\bar{\mathbf{u}}_t^\gamma$), we get the bound

$$\mathcal{R}_T^\gamma(\Delta_{2|\Gamma|}) \stackrel{\text{def}}{=} \sum_{t=1}^T \Big(y_t^\gamma - \bar{\mathbf{u}}_t^\gamma \cdot \bar{\mathbf{x}}_t\Big)^2 - \min_{\bar{\mathbf{u}}^\gamma \in \Delta_{2|\Gamma|}} \sum_{t=1}^T \Big(y_t^\gamma - \bar{\mathbf{u}}^\gamma \cdot \bar{\mathbf{x}}_t\Big)^2 \leqslant \mathrm{B}(\bar{\mathbf{x}}_{1:T}, y_{1:T}^\gamma, \mathbf{s}_0^\gamma, \bar{\mathbf{u}}^\gamma).$$

For any instance $t = 1, \dots, T$, and for any $\bar{\mathbf{u}}^\gamma \in \Delta_{2|\Gamma|}$, we obtain the equality of the two scalar products $\bar{\mathbf{u}}^\gamma \cdot \bar{\mathbf{x}}_t$ and $\psi(\mathbf{u}^\gamma) \cdot \mathbf{x}_t$:

$$\bar{\mathbf{u}}^\gamma \cdot \bar{\mathbf{x}}_t = \Big(\bar{\mathbf{u}}^{\gamma+} \,|\, \bar{\mathbf{u}}^{\gamma-}\Big) \cdot \Big(\alpha \mathbf{x}_t \,|\, -\alpha \mathbf{x}_t\Big) = \alpha \Big(\bar{\mathbf{u}}^{\gamma+} - \bar{\mathbf{u}}^{\gamma-}\Big) \cdot \mathbf{x}_t = \psi(\bar{\mathbf{u}}^\gamma) \cdot \mathbf{x}_t \,.$$

Lemma 3.4 implies that for any $\mathbf{u}^\gamma \in \mathcal{B}_\alpha$, there is at least one vector $\bar{\mathbf{u}}^\gamma \in \Delta_{2|\Gamma|}$ such that $\psi(\bar{\mathbf{u}}^\gamma) = \mathbf{u}^\gamma$ and we get the equality:

$$\min_{\mathbf{u}^\gamma \in \mathcal{B}_\alpha} \sum_{t=1}^T \Big(y_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t\Big)^2 = \min_{\bar{\mathbf{u}}^\gamma \in \Delta_{2|\Gamma|}} \sum_{t=1}^T \Big(y_t^\gamma - \psi(\bar{\mathbf{u}}^\gamma) \cdot \mathbf{x}_t\Big)^2 = \min_{\bar{\mathbf{u}}^\gamma \in \Delta_{2|\Gamma|}} \sum_{t=1}^T \Big(y_t^\gamma - \bar{\mathbf{u}}^\gamma \cdot \bar{\mathbf{x}}_t\Big)^2 .$$

So with, for any instance $t = 1, \dots, T$, $\mathbf{u}_t^\gamma \stackrel{\text{def}}{=} \psi(\bar{\mathbf{u}}_t^\gamma)$, we obtain

$$\mathcal{R}_T^\gamma(\mathcal{B}_\alpha) \stackrel{\text{def}}{=} \sum_{t=1}^T \Big(y_t^\gamma - \mathbf{u}_t^\gamma \cdot \mathbf{x}_t\Big)^2 - \min_{\mathbf{u}^\gamma \in \mathcal{B}_\alpha} \sum_{t=1}^T \Big(y_t^\gamma - \mathbf{u}^\gamma \cdot \mathbf{x}_t\Big)^2 = \mathcal{R}_T^\gamma(\Delta_{2|\Gamma|}) \,.$$

This equality provides a bound on $\mathcal{R}_T^\gamma(\mathcal{B}_\alpha)$ when predictions are $\widehat{y}_t^\gamma = \psi^{-1}(\bar{\mathbf{u}}_t^\gamma) \cdot \mathbf{x}_t = \mathbf{u}_t^\gamma \cdot \mathbf{x}_t$. With this trick, the previous bounds (3.11) and (3.12) are still true by replacing $|\Gamma|$ (the dimension of the features $\mathbf{x}_t$) by $2|\Gamma|$ (the dimension of the new features $\bar{\mathbf{x}}_t$) and the bound E (previously equals to $4C^2$) by $2\alpha(\alpha+1)C^2$ (the bound on the new pseudo prediction errors are calculated below):

$$\begin{aligned}
\mathcal{R}_T(\mathcal{B}_\alpha) \leqslant\ & |\Gamma| \Bigg(2\,\alpha\,(\alpha+1)\,C^2\,\sqrt{T+1} \Bigg(\frac{\sqrt{2\ln|\Gamma|}}{\sqrt{2}-1} + \frac{\ln(1+2^{-1}\ln T)}{\sqrt{\ln|\Gamma|}} \Bigg) \\
& \qquad + 2\,\alpha\,(\alpha+1)\,C^2 \Big(2\ln|\Gamma| + 2\ln(1+2^{-1}\ln T) + 1\Big) \Bigg) \qquad \text{for BOA} \\
\leqslant\ & 2\,\alpha\,(\alpha+1)\,C^2\,|\Gamma| \sqrt{|\Gamma|(T+1)\Big(1+\ln(1+T)\Big)} \qquad \text{for ML-Poly.}
\end{aligned}$$

The complete online algorithm leading to these bounds is summarized in Algorithm 5.

Algorithm 5 Scheme for on-line linear regression.

input Algorithm $\mathcal{A}^\gamma$ and bound on the weight vectors $\alpha > 0$

for $t = 1, \dots, T$ **do**

Get the feature vector $\mathbf{x}_t$ and denote (where $|$ is the concatenation operator between vectors)

$$\bar{\mathbf{x}}_t \stackrel{\text{def}}{=} \Big(\alpha \mathbf{x}_t \,|\, -\alpha \mathbf{x}_t\Big) \in \mathbb{R}^{2|\Gamma|}$$

Run algorithm $\mathcal{A}^\gamma$ on node γ with $\bar{\mathbf{x}}_t$ and get the weight vector $\bar{\mathbf{u}}_t^\gamma = \Big(\bar{\mathbf{u}}_t^{\gamma+} \,|\, \bar{\mathbf{u}}_t^{\gamma-}\Big)$

Output the weight vector $\mathbf{u}_t^\gamma = \alpha\Big(\bar{\mathbf{u}}_t^{\gamma+} - \bar{\mathbf{u}}_t^{\gamma-}\Big)$ and predicts $\widehat{y}_t^\gamma = \mathbf{u}_t^\gamma \cdot \mathbf{x}_t$

Bound on new pseudo prediction errors. Since boundedness assumptions (3.7) hold, the transformed features $\bar{\mathbf{x}}_t^\gamma$ are bounded by αC. Moreover, $\bar{\mathbf{u}}_t^\gamma \in \Delta_{2|\Gamma|}$ implies $\Big\|\bar{\mathbf{u}}_t^\gamma\Big\|_1 = 1$, so we get

$$\Big|\widehat{y}_t^\gamma\Big| = \Big|\bar{\mathbf{u}}_t^\gamma \cdot \bar{\mathbf{x}}_t\Big| \leqslant \Big\|\bar{\mathbf{u}}_t^\gamma\Big\|_1 \Big\|\bar{\mathbf{x}}_t\Big\|_\infty = \alpha C \,.$$

Moreover we have $|y_t^\gamma - \widehat{y}_t^\gamma| \leqslant |y_t^\gamma| + |\widehat{y}_t^\gamma| \leqslant \big(\alpha+1\big)C$ as the observations are still bounded by C, and we obtain a bound on the pseudo prediction errors:

$$\left|\widetilde{\ell}_t^\gamma(\bar{\mathbf{x}}_t)\right| = \left\|2\big(\widehat{y}_t^\gamma - y_t^\gamma\big)\bar{\mathbf{x}}_t\right\|_\infty \leqslant 2\alpha(1+\alpha)C^2\,.$$

Proof of Lemma 3.4. Denoting respectively by $(\mathbf{u})_+$ and $(\mathbf{u})_-$ the non-negative and non-positive parts of any vector $\mathbf{u}$ and by $\mathbf{1}_{|\Gamma|}$ the vector of size $|\Gamma|$ of which all coordinates are 1, we introduce the inverse function ψ^{-1}:

$$\psi^{-1}: \begin{array}{ccc} \mathcal{B}_\alpha & \longrightarrow & \Delta_{2|\Gamma|} \\ \mathbf{u} & \longmapsto & \frac{1}{\alpha}\left(\frac{\alpha-\|\mathbf{u}\|_1}{2|\Gamma|}\mathbf{1}_{|\Gamma|} + (\mathbf{u})_+ \,\middle|\, \frac{\alpha-\|\mathbf{u}\|_1}{2|\Gamma|}\mathbf{1}_{|\Gamma|} + (\mathbf{u})_-\right). \end{array}$$

First we will show that function images are in the right sets, meaning that for any $\mathbf{u} \in \mathcal{B}_\alpha$, $\psi^{-1}(\mathbf{u}) \in \Delta_{2|\Gamma|}$ and for any $\bar{\mathbf{u}} \in \Delta_{2|\Gamma|}$, $\psi(\bar{\mathbf{u}}) \in \mathcal{B}_\alpha$. Secondly, we obtain the surjectivity of ψ by proving that for any $\mathbf{u} \in \mathcal{B}_\alpha$, $\psi(\psi^{-1}(\mathbf{u})) = \mathbf{u}$.

Proof that for any $\mathbf{u} \in \mathcal{B}_\alpha$, $\psi^{-1}(\mathbf{u}) \in \Delta_{2|\Gamma|}$. We set $\mathbf{u} \in \mathcal{B}_\alpha$. By definition for any $i \in \Gamma$, $(u^i)_\pm \geqslant 0$ and as $\mathbf{u} \in \mathcal{B}_\alpha$, $(\alpha - \|\mathbf{u}\|_1)/(2|\Gamma|) \geqslant 0$. So, all the coefficients of $\psi^{-1}(\mathbf{u})$ are non-negative. Since $\sum_{i\in\Gamma}(u^i)_+ + (u^i)_- = \sum_{i\in\Gamma}|u^i| = \|\mathbf{u}\|_1$, the sum of the coefficients of the vector $\psi^{-1}(\mathbf{u})$ equals 1:

$$\sum_{i\in\Gamma}\Big(\psi^{-1}(\mathbf{u})\Big)^{i+} + \Big(\psi^{-1}(\mathbf{u})\Big)^{i-} = \frac{1}{\alpha}\sum_{i\in\Gamma}\left(\big(u^i\big)_+ + \big(u^i\big)_- + \frac{\alpha-\|\mathbf{u}\|_1}{|\Gamma|}\right) = \frac{1}{\alpha}\big(\|\mathbf{u}\|_1 + \alpha - \|\mathbf{u}\|_1\big)$$

and thus $\bar{\mathbf{u}} = \psi(\mathbf{u}) \in \Delta_{2|\Gamma|}$.

Proof that for any $\bar{\mathbf{u}} \in \Delta_{2|\Gamma|}$, $\psi(\bar{\mathbf{u}}) \in \mathcal{B}_\alpha$. With $\bar{\mathbf{u}} = \big(\bar{\mathbf{u}}^+ \,|\, \bar{\mathbf{u}}^-\big) \in \Delta_{2|\Gamma|}$, using that all the coefficients of $\bar{\mathbf{u}}$ are non-negative and that their sum equals 1 that is $\left\|\bar{\mathbf{u}}\right\|_1 = 1$, we get

$$\|\psi(\bar{\mathbf{u}})\|_1 \stackrel{\text{def}}{=} \left\|\alpha\bar{\mathbf{u}}^+ - \alpha\bar{\mathbf{u}}^-\right\|_1 \leqslant \alpha\left\|\bar{\mathbf{u}}^+\right\|_1 + \alpha\left\|\bar{\mathbf{u}}^-\right\|_1 = \alpha\left\|\bar{\mathbf{u}}\right\|_1 = \alpha.$$

Proof that for any $\mathbf{u} \in \mathcal{B}_\alpha$, $\psi\Big(\psi^{-1}(\mathbf{u})\Big) = \mathbf{u}$.

$$\psi\Big(\psi^{-1}(\mathbf{u})\Big) = \frac{\alpha-\|\mathbf{u}\|_1}{2|\Gamma|}\mathbf{1}_{|\Gamma|} + (\mathbf{u})_+ - \frac{\alpha-\|\mathbf{u}\|_1}{2|\Gamma|}\mathbf{1}_{|\Gamma|} - (\mathbf{u})_- = \mathbf{u}\,. \qquad \square$$

3.7 Experiments

Our application relies on electricity consumption data of a large number of households to which we have added meteorological data (see Section 3.7.1). Non-temporal information (sociological type, region, type of heating fuel and type of electricity contract) on the households is also provided. From these temporal and non-temporal data, we dispatch the households into clusters thanks to the methods presented in Section 3.7.2. We describe the experiments and analyze the results in Sections 3.7.3 and 3.7.4.

Variable	Description	Range / Value
Acorn	Acorn category value	From 1 to 6
Region	UK NUTS of level 3	UK- H23, -J33, -L15, -L16, -L21, -M21, or -M27
Fuel	Type of heating fuel	Electricity (E) or Electricity and Gas (EG)
Tariff	Contract type	Standard (Std) or Time of Use tariff (ToU)
Temperature	Air temperature	From $-20°$ to $30°$
Visibility	Air visibility	From 0 to 10 (integer)
Humidity	Air humidity percentage	From 0% to 100%
Date	Current time	From April 20, 2009 to July 31, 2010 (half-hourly)
Consumption	Power consumption	From 0.001 to 900 kWh
Fuel + Tariff	Cross of Fuel and Tariff variables	"E - Std", "EG - Std", "E - ToU" or "EG - ToU"
Half-hour	Half-hour of the day	From 1 to 48 (integer)
Day	Day off the week	From 1 (Monday) to 7 (Sunday) (integer)
Position in the year	Linear values	From 0 (Jan 1, 00:00) to 1 (Dec 31, 23:59)
Smoothed temperature	Smoothed air temperature	From $-20°$ to $30°$

Table 1 – Summary of the variables provided and created for each household of the data set.

3.7.1 The Underlying Real Data Set

The project "*Energy Demand Research Project*[1]", managed by Ofgem on behalf of the UK Government, was launched in late 2007 across Great Britain (see AECOM, 2018 and Schellong, 2011). Power consumptions of approximately 18,000 households with smart-type meters were collected at half-hourly intervals for about two years. We detail below how we select only the consumption of 1,545 households over the period from April 20, 2009 to July 31, 2010 – Taieb et al. [2017], who used the same data, performed similiar pre-processing in their experiments. Four non-temporal variables are associated with each household: the Region (the initial data set provides the level-4 NUTS[2] codes but we consider larger subdivisions – from 150,000 to 800,000 inhabitants – and associate each household with its level-3 code), the Acorn category value (an integer between 1 and 6 associated with an United Kingdom's population demographic type – this segmentation was developed by the company CACI Limited), the type of heating fuel ("electricity" or "electricity and gas") and the contract type ("Standard" or "Time of Use tariff" for households containing an electricity meter with a dynamic time of use tariff) for each household. In a first data cleaning step, we removed households with more than 5 missing consumption records over the period April 20, 2009 to July 31, 2010 (around $1,600$ households are thus kept) – the remaining missing consumption data points are imputed by a linear interpolation. Among the various clusterings of the households we consider in our experiments, three of them rely on three qualitative variables: "Region", "Tariff" and "Fuel + Tariff" (which is based on both the heating fuel type and the contract type). If one of the values of these qualitative variables had fewer than 20 occurrences, we have removed from the data set the households associated with that value. The final data set then contains the electrical consumption records of the 1,545 remaining households. From now on, we will denote by $\mathcal{I}$ the set of households and by $(y_{i\,t})_{1-T_0 \leqslant t \leqslant T}$ the time series of the half-hourly power consumption of the $i \in \mathcal{I}$ household. Finally, we added the temperature, visibility and humidity for each region from the NOAA[3] data: we

[1] www.ofgem.gov.uk/gas/retail-market/metering/transition-smart-meters/energy-demand-research-project

[2] *Nomenclature des Unités Territoriales Statistiques* (nomenclature of territorial units for statistics)

[3] National Oceanic and Atmospheric Administration, www.noaa.gov

selected a weather station (with records available over the considered period) in each region and linearly interpolated the meteorological data to get 48 measurements per day (compared to 8 initially). Table 1 sums up the available variables of our data set and gives their range.

3.7.2 Clustering of the Households

We present, in Paragraphs 3.7.2.1 to 3.7.2.3, three methods to cluster the households and we compare them in the last paragraph of this subsection. After choosing a segmentation (or two crossed segmentations), we only consider, for each cluster, the aggregated consumption of its households. Thus, for any subset $\gamma \subset \mathcal{I}$, we compute the time series $y_t^\gamma \stackrel{\text{def}}{=} \sum_{i\in\gamma} y_{i\,t}$ that we want to forecast and once clusterings are chosen, we never consider individual power consumption.

3.7.2.1 Random Clustering

We first consider the simplest way to cluster households: the segmentation is built randomly. In the experiments of Section 3.7.4, the number of clusters varies from 4 to 64. As an example here, we consider 4 clusters and we randomly assign a number between 1 and 4 to each household and obtain the weekly profiles plotted in Figure 5. In the following, we will call "Random (k)", a segmentation of k clusters built randomly. Naturally, the curves are similar and the clusters are therefore rather homogeneous.

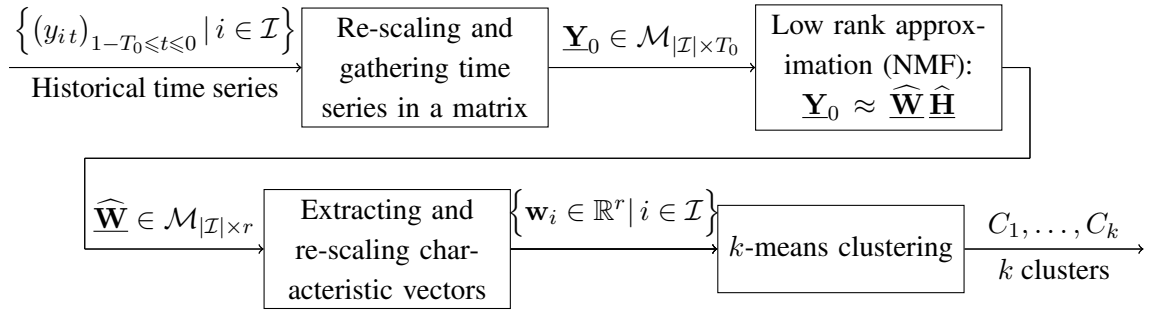
3.7.2.2 Segmentation Based on Qualitative Household Variables

The second approach consists in grouping households according to the provided non-temporal information. We consider the natural segmentations "Region", "Acorn" and "Fuel + Tariff" based on the corresponding qualitative variables and we plot the weekly profile of each cluster on Figures 6, 7 and 8. Regions have an impact on the consumption profile: the evening consumption peak time varies by location. Moreover, consumption of the Wales regions (UKL15, UKL16 and UKL21) is lower than that of the other regions (see Figure 6). In the Acorn classification, the lower the value, the richer the household, thus Figure 7 shows that wealthiest households consume the most (as expected). Finally, the type of heating fuel does not seem to have a significant impact on the weekly consumption profile (although we have observed that when the heating is partly gas, the consumption is slightly lower and in winter, it is less sensitive to the temperature drops). Similarly, it seems that the type of contract does not influence the consumption profiles. Peak consumption in the evening is however less important for a dynamic time of fuse tariff than for the standard tariff. It should be noted that since time slots of prices may change from day to day, it is difficult to quantify here the impact of the tariff, as we are only showing average consumption profiles.

3.7.2.3 Clustering Based on Non-Negative Matrix Factorization and k-Means Method

The last method relies on an historical individual time series of household power consumption (April 20, 2009 to April 20, 2010). We propose a method to extract from these time series a low number – denoted by r – of combined household characteristics and to use them to build relevant clusterings. The diagram below sums up the steps of the procedure described here quickly. We then further detail them one by one. The $|\mathcal{I}|$ historical times series

$\big(y_{i\,t}\big)_{1-T_0\leqslant t\leqslant 0}$ are firstly re-scaled and gathered into a matrix $\underline{\mathbf{Y}}_0 \in \mathcal{M}_{|\mathcal{I}|\times T_0}$. We then reduce the dimension of data with a non-negative matrix factorization (NMF): we approximate $\underline{\mathbf{Y}}_0$ by $\widehat{\underline{\mathbf{W}}}\,\widehat{\underline{\mathbf{H}}}$, where $\widehat{\underline{\mathbf{W}}}$ and $\widehat{\underline{\mathbf{H}}}$ are $|\mathcal{I}| \times r$ and $r \times T_0$-non-negative matrices, respectively. As soon as this approximation is good enough, line i of the matrix $\widehat{\underline{\mathbf{W}}}$ is sufficient to reconstruct the historical time series of household i (with the knowledge of matrix $\widehat{\underline{\mathbf{H}}}$ - which is not used for the clustering). Thus, we assign, to each household, r characteristics: the lines of $\widehat{\underline{\mathbf{W}}}$. After a re-scaling step – to give the same importance to each of those characteristics – we get the r-vectors $(\mathbf{w}_i)_{i\in\mathcal{I}}$. With this low-dimension representation of households in $\mathbb{R}^r$, we use k-means clustering algorithm in $\mathbb{R}^r$ to provide the k clusters $C_1, \ldots, C_k$ and we write "NMF (k)" for such a clustering.



Re-scaling and Gathering Time Series in a Matrix. For $T_0 > 0$, we consider the $|\mathcal{I}|\times T_0$-matrix $\underline{\mathbf{Y}}_0$ which contains the re-scaled historical power consumption time series: for any $i \in \mathcal{I}$ and any $1 - T_0 \leqslant t \leqslant 0$,

$$(\underline{\mathbf{Y}}_0)_{i\,t} \stackrel{\text{def}}{=} \frac{y_{i\,t}}{\bar{y}_i}, \quad \text{with} \quad \bar{y}_i \stackrel{\text{def}}{=} \frac{1}{T_0}\sum_{t=1-T_0}^{0} y_{i\,t}.$$

Low Rank Approximation. Since we are interested in power consumption, all the coefficients of $\underline{\mathbf{Y}}_0$ are non-negative - we will write $\underline{\mathbf{Y}}_0 \geqslant 0$ and say that this matrix is non-negative. To reduce dimension of non-negative matrices, Paatero and Tapper [1994] and Lee and Seung [1999] propose a factorization method whose distinguishing feature is the use of non-negativity constraints. Let us fix some integer $r \ll \min(|\mathcal{I}|, T_0)$, which will ensure a reduction of the dimension (we chose $r = 10$ in the experiments of the next subsection). The non-negative matrix factorization (NMF) approximates matrix $\underline{\mathbf{Y}}_0$ by $\underline{\mathbf{Y}}_0 \approx \underline{\mathbf{W}}^\star\underline{\mathbf{H}}^\star$, where $\underline{\mathbf{W}}^\star$ and $\underline{\mathbf{H}}^\star$ are $|\mathcal{I}| \times r$ and $r \times T_0$ non-negative matrices. They are computed by solving:

$$\big(\underline{\mathbf{W}}^\star, \underline{\mathbf{H}}^\star\big) \in \underset{\underline{\mathbf{W}},\,\underline{\mathbf{H}}\geqslant 0}{\operatorname{argmin}} \big\|\underline{\mathbf{Y}}_0 - \underline{\mathbf{W}}\underline{\mathbf{H}}\big\|_F^2 = \underset{\underline{\mathbf{W}},\,\underline{\mathbf{H}}\geqslant 0}{\operatorname{argmin}} \sum_{i,t}\Big(y_{i\,t} - \big(\underline{\mathbf{W}}\underline{\mathbf{H}}\big)_{i\,t}\Big)^2.$$

We use the function `NMF` of the `Python`-library `sklearn.decomposition` to approach a local minimum with a coordinate descent solver and denote by $\widehat{\underline{\mathbf{W}}}$ the approximation of $\underline{\mathbf{W}}^\star$. Thanks to the NMF, for any $i \in \mathcal{I}$, r characteristics (the i^{th} line of matrix $\widehat{\underline{\mathbf{W}}}$) are thus computed.

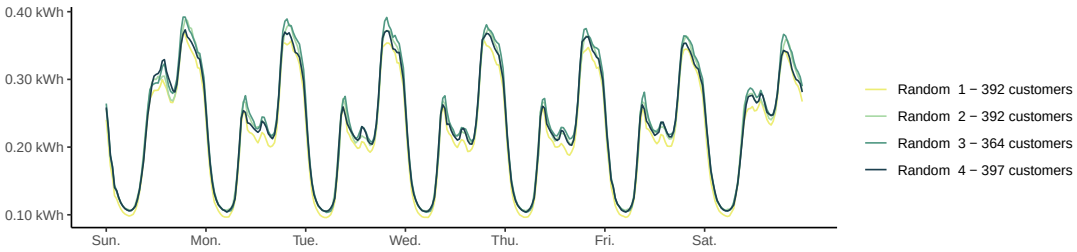


Figure 5 – Mean consumption per week and per cluster, with households randomly assigned to an integer from 1 to 4.

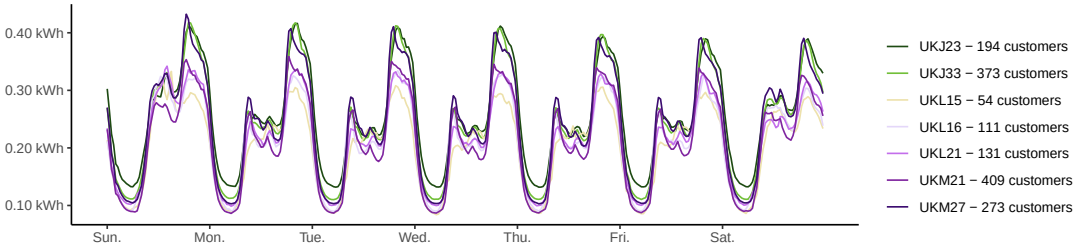


Figure 6 – Mean consumption per week and per region (UK NUTS of level 3).

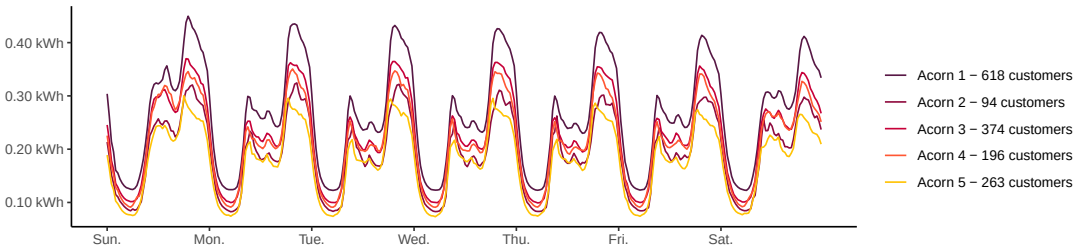


Figure 7 – Mean consumption per week and per Acorn category value (from 1 to 5).

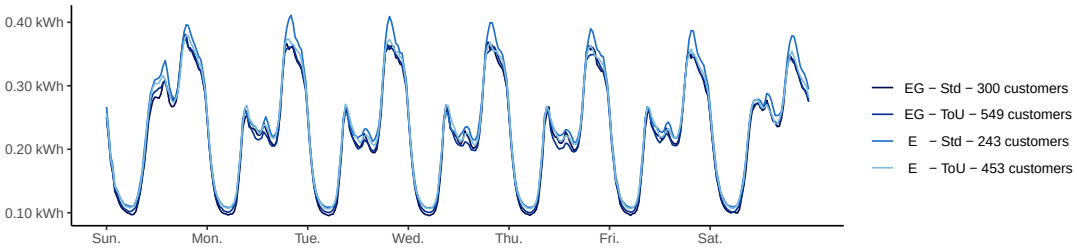


Figure 8 – Mean consumption per week for the households clustered according “Fuel + Tariff".

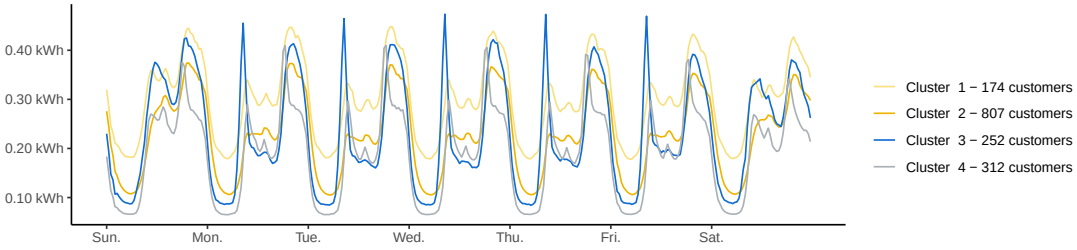


Figure 9 – Mean consumption per week and per cluster, with each household assigned to one of the four groups according to the NMF and k-means procedure (“NMF (4)" clustering).

Extracting and Re-scaling Characteristic Vectors. To give the same impact to each of these characteristics, we re-scale the columns of $\underline{\widehat{\mathbf{W}}}$ and define, for each household i, the vector

$$\mathbf{w}_i = \left(\frac{\underline{\widehat{\mathbf{W}}}_{i\,1}}{\sum_{j\in\mathcal{I}} \underline{\widehat{\mathbf{W}}}_{j\,1}}, \cdots, \frac{\underline{\widehat{\mathbf{W}}}_{i\,r}}{\sum_{j\in\mathcal{I}} \underline{\widehat{\mathbf{W}}}_{j\,r}} \right).$$

k-Means Clustering. The k-means algorithm (introduced by MacQueen [1967]) is then used on these r- vectors to cluster the households into a fixed number k of groups (which varies from 4 to 64 in our experiments). We recall below how this algorithm works. With $\{C_1, \ldots, C_k\}$ a k-clustering of set $\mathcal{I}$, for any $1 \leqslant \ell \leqslant k$, we define the center $\bar{\mathbf{w}}_\ell$ and the variance $\mathrm{Var}(C_\ell)$ of cluster C_ℓ by

$$\bar{\mathbf{w}}_\ell \overset{\text{def}}{=} \frac{1}{|C_\ell|} \sum_{i\in C_\ell} \mathbf{w}_i \quad \text{and} \quad \mathrm{Var}(C_\ell) \overset{\text{def}}{=} \frac{1}{C_\ell} \sum_{i\in C_\ell} \|\mathbf{w}_i - \bar{\mathbf{w}}_\ell\|^2.$$

In k-means clustering, each household belongs to the cluster with the nearest center. The best set of clusters, denoted by $\left\{C_1^\star, \ldots, C_k^\star\right\}$ – namely the best set of centers – is obtained by minimizing the following criterion:

$$\left\{C_1^\star, \ldots, C_k^\star\right\} \in \underset{\{C_1,\ldots,C_k\}}{\operatorname{argmin}} \sum_{\ell=1}^{k} \sum_{\mathbf{w}\in C_\ell} \left\|\mathbf{w} - \bar{\mathbf{w}}_\ell\right\|^2 = \underset{\{C_1,\ldots,C_k\}}{\operatorname{argmin}} \sum_{\ell=1}^{k} |C_\ell| \mathrm{Var}(C_\ell).$$

In practice, we use the use `KMeans` function of the `Python`-library `sklearn.cluster` to compute clusters.

Description and Analysis of "NMF (4)". For $k = 4$, weekly profiles are plotted in Figure 9. This clustering seems to detect consumption behaviors much more specific than any of the previous ones. Indeed, Clusters 3 and 4 present a peak of consumption early in the morning on working days, while the consumption of Cluster 2– which includes the largest number of households – remains almost flat throughout the morning. Moreover, the evening peak for Cluster 4 arrives earlier than for the other clusters. Finally, the consumption of Cluster 1 is generally the highest, while that of Cluster 2 is the lowest.

3.7.2.4 Comparison of Clusterings

To measure similarity between the clusterings above, we calculate the adjusted rand index (ARI) – see Rand [1971] – for each segmentation pair and report the values thus obtained in Table 2. Given a set elements $\mathcal{I}$ and two partitions to compare, for example the segmentation "Region" $\{R_1, \ldots R_N\}$ and another clustering $\{C_1, \ldots C_k\}$, the ARI is defined by

$$ARI \overset{\text{def}}{=} \frac{\displaystyle\sum_{\ell=1}^{k}\sum_{n=1}^{N} \binom{|C_\ell \cap R_n|}{2} - \left[\sum_{\ell=1}^{k} \binom{|C_\ell|}{2} \sum_{n=1}^{N} \binom{|R_n|}{2}\right] \Big/ \binom{|\mathcal{I}|}{2}}{\frac{1}{2}\left[\displaystyle\sum_{\ell=1}^{k} \binom{|C_\ell|}{2} + \sum_{n=1}^{N} \binom{|R_n|}{2} - \sum_{\ell=1}^{k} \binom{|C_\ell|}{2} \sum_{n=1}^{N} \binom{|R_n|}{2}\right] \Big/ \binom{|\mathcal{I}|}{2}}.$$

ARI lies in $[-1, 1]$ by construction, it is equal to 0 for a random matching between clusters of the two considered segmentations and to 1 for a perfect alignment. Similarity between our different household partitions is very low, only “Region" is slightly correlated with all other clusterings, and “NMF (4)" with “ACORN". But these correlations remain low and the clustering “NMF (4)" therefore seems to extract, from historical time series, some households information that are not contained in other clusterings. Its use should improve forecasts – this will be confirmed by the experiments below.

	Region	**NMF (4)**	**Acorn**	**Fuel + Tariff**
Random (4)	-0.000	0.000	0.003	-0.000
Fuel + Tariff	0.016	-0.001	0.004	
Acorn	0.043	0.018		
NMF (4)	0.011			

Table 2 – ARI (Adjusted Rand index) for each segmentation pair.

3.7.3 Experiment Design

Thanks to the above methods, we established several partitions of the household set $\mathcal{I}$. As explained below, choosing one or two of them amounts to considering a two-level hierarchy (Example 1) or two crossed hierarchies (Example 4). We also detail the corresponding set of node Γ. We then describe how we build meteorological data for each node $\gamma \in \Gamma$ and generate corresponding features. Finally, we focus on standardization and online calibration of aggregation hyper-parameters. We have divided the data set into training data: one-year of historical data (from April 20, 2009 to April 19, 2010) – used for NMF clusterings, feature generation method training, and standardization – and testing data. As aggregation algorithms start from scratch, they work poorly during the first rounds. We therefore withdraw the first 10 days of testing data from the performance evaluation period. So, April 20, 2010 to April 30, 2010 is left for initializing aggregation algorithms and the hyper-parameters calibration and our methods are then tested during the last three months (from May 1, 2010 to July 31, 2010). We summarize in Table 3 the range of dates for each step of the procedure.

	Start date	End date
NMF Clusterings		
Feature Generation Model Training	April 20, 2009	April 19, 2010
Features and Observations Standardization		
Initialization of the Aggregation	April 20, 2010	April 30, 2010
Model Evaluation	May 1, 2010	July 31, 2010

Table 3 – Date range for the steps of the proposed method

Underlying Hierarchy. As detailed in Section 3.2, we aim to forecast a set of power consumption time series $\left\{(y_t^\gamma)_{t>0}, \gamma \in \Gamma\right\}$ connected to each other by some summation constraints. These constraints are represented by one (or more) tree(s) and Γ denotes the set of

its (or their) nodes. We refer to Example 1 if we consider a single segmentation and to Example 4 for two crossed clusterings. We detail below the set Γ, which will contain some subsets of households set $\mathcal{I}$, for these two configurations. We recall that we denote the average power consumption of a group of households $\gamma \subset \mathcal{I}$ by $y_t^\gamma \stackrel{\text{def}}{=} \sum_{i\in\gamma} y_{i\,t}$. Considering a single clustering $(C_1, \dots C_N)$ of $\mathcal{I}$, we want to forecast the consumption of each cluster C_ℓ, and also the global consumption (namely, the one for $\gamma = \mathcal{I}$). Thus, we set $\Gamma = \{C_\ell\}_{1\leqslant\ell\leqslant N} \cup \{\mathcal{I}\}$ and the associated time series respect the hierarchy of Figure 1 – where y^{Tot} refers to the time series associated with $\mathcal{I}$ and $y^1, y^2, \dots, y^N$ with the ones of clusters $C_1, C_2, \dots C_N$. We now consider two partitions. The first one $R_1, \dots R_N$ refers to segmentation "Region" and the second one, $C_1, \dots C_k$ to any other clustering. We would like to forecast the global consumption ($\gamma = \mathcal{I}$), the consumption associated with each region ($\gamma = R_n$, for $n = 1, \dots, N$) and with each cluster ($\gamma = C_\ell$, for $\ell = 1, \dots, k$) but also the power consumption of cluster C_1 in region R_1 ($\gamma = C_1 \cap R_1$), of cluster C_1 in region R_2 ($\gamma = C_1 \cap R_2$), and so on. Thus,we consider the set of nodes

$$\Gamma = \{C_\ell \cap R_n\}_{1\leqslant\ell\leqslant k,\, 1\leqslant n\leqslant N} \cup \{C_\ell\}_{1\leqslant\ell\leqslant k} \cup \{R_n\}_{1\leqslant n\leqslant N} \cup \{\mathcal{I}\}.$$

The hierarchy associated with such crossed segmentations is represented in Figure 4 (with $N_1 = k$ and $N_2 = N$) – where the global consumption, associated with $\mathcal{I}$, is denoted by y^{Tot}, the one of cluster C_ℓ by $y^{\ell\,\cdot}$, the one of region R_n, by $y^{\cdot\,n}$ and where $y^{\ell\,n}$ refers to the local consumption of $C_\ell \cap R_n$.

Meteorological Data of any Set of Households. Methods presented in Section 3.5 for feature creation implicitly assume that meteorological data are available. We recall that we collected meteorological data for each of the N regions. Thus when $\gamma \in \Gamma$ refers to one of these regions, we can directly apply the feature generation methods. However, if node γ groups households from different regions, these data are not directly available and one may even wonder what they should correspond to. We take convex combinations of regional meteorological data, in proportions corresponding to the locations of the households. More precisely, for each meteorological variable (temperature, visibility or humidity), we built the meteorological variable of γ as a convex combination of the N meteorological variables of the N regions. The weight associated with region n corresponds to the proportion of this region in γ, in terms of contribution to the consumption – this contribution is determined from historical data.

Feature Creation. For each node γ, we now have access to calendar and meteorological data. Considering an exponential smoothed temperature – that models the thermal inertia of buildings – is likely to improve forecasts (see among others, Taylor, 2003 and Goude et al., 2014), so we create the a-exponential smoothing of the temperature $\bar{\tau}_t^\gamma \stackrel{\text{def}}{=} a\bar{\tau}_{t-1}^\gamma + (1-a)\tau_t^\gamma$, where $a \in [0, 1]$. After testing several values and evaluating their performance on the training set, we set $a = 0.999$.

We then apply methods of Section 3.5 using available explanatory variables to generate features $\mathbf{x}_t$. Each model (auto-regressive model, generalized additive model or random forest) is trained on a year of historical data (from April 20, 2009 to April 20, 2010). Then, forecasts are computed on the period April 20, 2010 to July 31, 2010. On the left of Figure 10,

we represent these benchmark predictions and the observations for the global consumption (namely $\gamma = \mathcal{I}$) over the last three days of the test period. On the right, we plot daily signed errors, $\frac{1}{48}\sum_{s=t}^{t+48}\left(y_s^\gamma - x_s^\gamma\right)$, for $\gamma = \mathcal{I}$ over the last week of the test period. Finally, daily mean squared errors, $\frac{1}{48}\sum_{s=t}^{t+48}\left(y_s^\gamma - x_s^\gamma\right)^2$, are computed for each test period day and represented by box-plots on Figure 11. The generalized additive model seems to perform the best (and the auto-regressive model the worst), this will be confirmed by the numerical results of the next subsection.

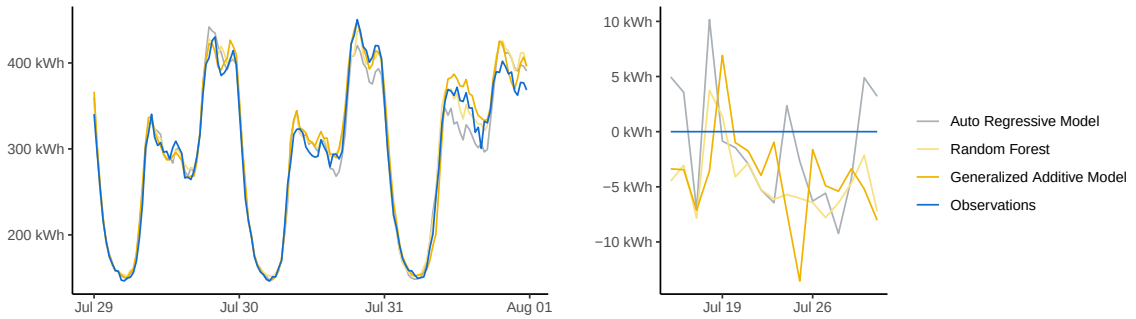


Figure 10 – Left picture: benchmark forecasts (auto-regressive model, generalized additive model, random forest) and observations of global consumption ($\gamma = \mathcal{I}$) at half-hour intervals on the last three days of the test period. Right picture: corresponding daily average signed errors on the last week of the test period.

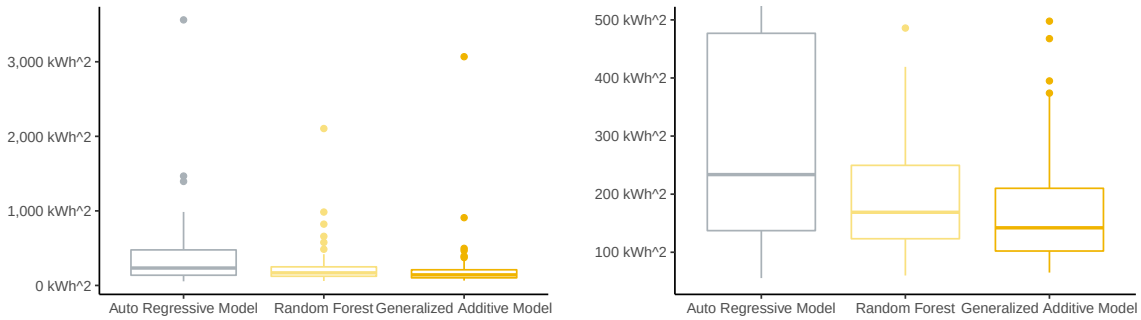


Figure 11 – Distribution over the test period of daily mean squared error of global consumption benchmark forecasts (auto-regressive model, generalized additive model, random forest). Left picture: original boxplots; Right picture: boxplots trimmed at 500 kWh^2.

Observations and Features Standardization. Once above features computed, they are standardized using the protocol presented in Section 3.6.1. We assess the quality of the standardization for one given configuration, namely "Region + NMF (16)", with features generated by the general additive model (this configuration, which refers to the two crossed clusterings "Region" and "NMF (16)", reaches the lower predictions errors – see Table 8). As there are 7 regions, the set Γ consists of $16\times7+16+7+1 = 136$ nodes, but only 129 are non-empty. For both standardized and non-standardized observations and features, we compute, for each node $\gamma \in \Gamma$, the empirical mean and empirical standard deviation over the test period. The distributions are plotted in Figures 12 and 13, respectively. Since the abscissa for non-standardized data is in logarithmic scale, the mean and standard deviation of data differ

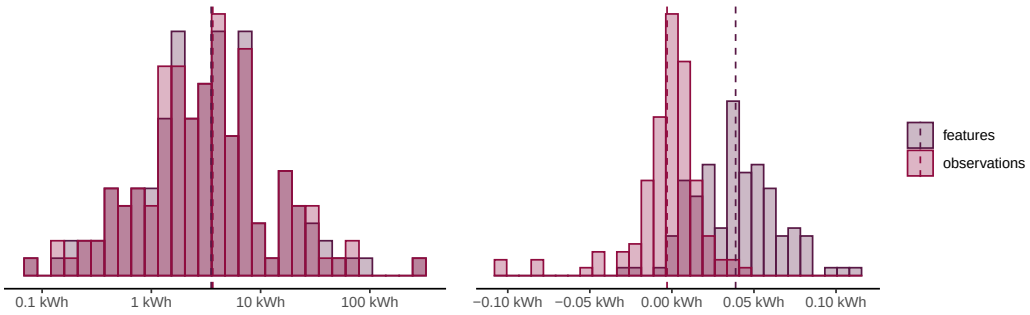


Figure 12 – Distribution of empirical means per cluster, for non-standardized and standardized observations and features.

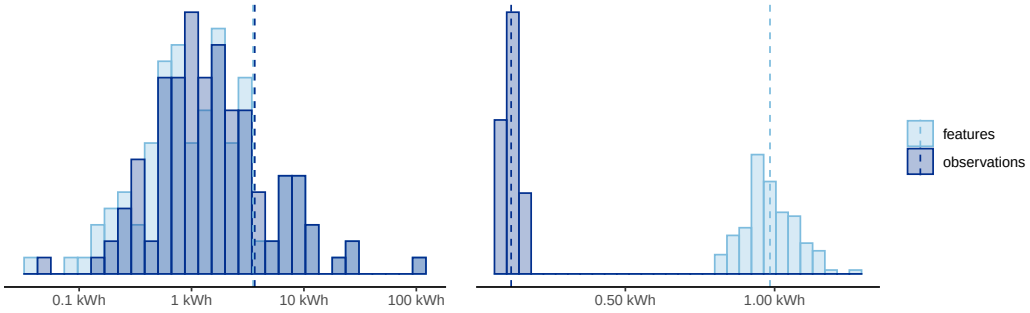


Figure 13 – Distribution of empirical standard deviations per cluster, for non-standardized (left) and standardized (right) observations and features.

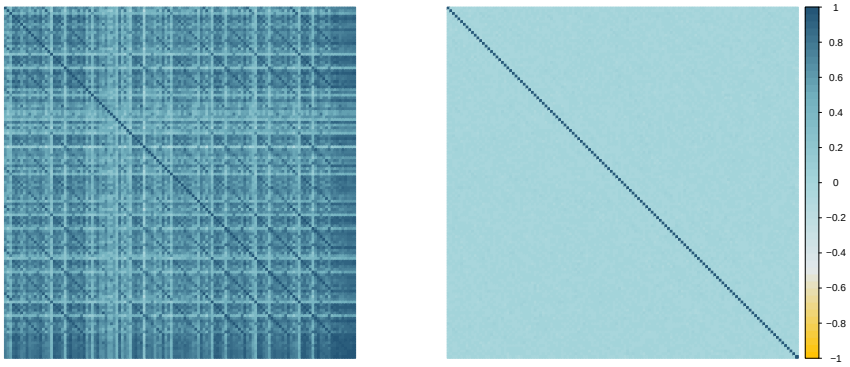


Figure 14 – Correlation matrix of non-standardized (left) and standardized (right) feature vectors.

	Mean	Bound	Standard deviation
Observations	9.53	570.02	3.65
Features	9.54	570.87	3.53
Standardized observations	-0.003	1.27	0.12
Standardized features	0.04	18.9	0.98

Table 4 – Mean, and maximum of absolute value and standard deviation of observations and features before and after standardization

a lot from a node to another. For example, the right-hand point is the global consumption ($\gamma = \mathcal{I}$), while points on the left correspond to the consumptions of small clusters. Thus, standardization centers data and decreases standard deviations of observations, as desired. In addition, standard deviations of features are close to 1. Figure 14 represents correlation matrices of the $|\Gamma|$-vectors $(\mathbf{x}_t)_{1\leqslant t\leqslant T}$ and $(\breve{\mathbf{x}}_t)_{1\leqslant t\leqslant T}$, that contain the non-standardized and standardized features over the test period. This shows that our standardization process is centering, re-scaling and de-correlating features.

Finally, Table 4 gathers numerical values of the average, over $\gamma \in \Gamma$, of empirical means and standard deviations (these values are indicated by dashed vertical lines on Figures 12 and 13). We also compute the maximum of the absolute value of features and observations – "Bound" column of the table. This gives an empirical approximation of the boundedness constant C – see boundedness assumptions (3.7).

Calibration of Hyper-Parameters. Once features and observations are standardized, we choose one of the algorithms presented in Section 3.6 and run it, on the $|\Gamma|$ nodes, in parallel with the same hyper-parameter. For the sequential non-linear ridge regression (NL-Ridge), we have to choose the regularization parameter λ (see Equation 3.8) and for BOA and ML-Pol algorithms, we need to set α, the radius of the $L1$-ball (see Algorithm 5). Henceforth,, we denote by β this hyper-parameter (which is equal to λ for NL-Ridge and to α for BOA and ML-Pol). We optimize the choice of β by grid search, which is simply an exhaustive search in a specified finite subset G of the hyper-parameter space. This optimization is performed sequentially. Indeed, for any node γ and any instance $t > 48$, we run $|G|$ algorithms in parallel and we chose the one – denoted by β_t – which minimizes the average prediction error on past available data. Thus, with $\widehat{y}_s^{\gamma}(\beta)$ the output, at an instance s, of algorithm $\mathcal{A}^{\gamma}$ run with β, we choose the parameter β_t as follows:

$$\beta_t \in \operatorname*{argmin}_{\beta\in G} \frac{1}{t-48} \sum_{s=1}^{t-48} \frac{1}{|\Gamma|} \sum_{\gamma\in\Gamma} \left(y_s^{\gamma} - \widehat{y}_s^{\gamma}(\beta)\right)^2.$$

In our experiments, to reduce the computational burden, we set $G = \left\{4^i \,|\, i = -5, -4, \ldots, 5\right\}$, so (only) 11 aggregations are run in parallel. At each new day, we check that we never reach the bounds 4^{-5} and 4^5. This kind of online calibration has shown good performance in load forecasting (see, for example, Devaine et al., 2013).

3.7.4 Results

In this subsection, we compare the four forecasting strategies detailed below by evaluating them on the testing period (May 1, 2010 to July 31, 2010), for each forecasting method of Section 3.5, for each aggregation algorithm of Section 3.6 and for various households clusterings. To do so, we introduce some prediction error defined below as well as a confidence bound on this error. We recall that we aim to forecast, at each instance t, a vector of time series $\mathbf{y}_t = (y_t^{\gamma})_{\gamma\in\Gamma}$. The first strategy, that we call "Benchmark", consists simply in providing the features $\mathbf{x}_t$ as forecasts. The second one considers only the projection step and thus skips the aggregation step (we will refer to it as the "Projection" strategy), the associated forecasts are thus the projected features $\Pi_{\underline{\mathbf{K}}}(\mathbf{x}_t)$. To measure the impact of the aggregation

step, without projection, we also evaluate the forecasts $\widehat{\mathbf{y}}_t$ (which do not necessary satisfy the hierarchical constraints) – this strategy is called "Aggregation". Finally, the strategy "Aggregation + Projection" provides the predictions $\widetilde{\mathbf{y}}_t = \Pi_{\underline{\mathbf{K}}}(\widehat{\mathbf{y}}_t)$. To allow for an evaluation of the accuracy of the prediction of some time series only, we define the prediction error $E_T(\Lambda)$, for some subset of nodes $\Lambda \subset \Gamma$. In the results below, this subset can be equal to Γ (to evaluate the strategies on all the nodes), to the singleton $\{\mathcal{I}\}$ (to focus on the global consumption – namely the consumption of all the households), or to the set of leaves of the tree associated with the considered segmentation(s), denoted by Γ_0 (to evaluate the performance of local forecasts only). Note that $E_T(\Gamma)$ will correspond to $\widetilde{L}_T \times |\Gamma|$ for the "Aggregation + Projection" strategy (see Equation 3.1). We now define, for any subset $\Lambda \subset \Gamma$, the prediction error $E_T(\Lambda)$. First of all, for a node $\gamma \in \Lambda$ and an instance t, let us denote by ε_t^γ the instantaneous squared error. It corresponds to $\left(y_t^\gamma - x_t^\gamma\right)^2$ for the "Benchmark" strategy, to $\left(y_t^\gamma - \left(\Pi_{\underline{\mathbf{K}}}(\mathbf{x}_t)\right)^\gamma\right)^2$ for "Projection", to $\left(y_t^\gamma - \widehat{y}_t^\gamma\right)^2$ for "Aggregation", and to $\left(y_t^\gamma - \widetilde{y}_t^\gamma\right)^2$ for the "Aggregation + Projection" strategy. We then consider the average (over time) squared error (which is cumulated over Λ):

$$E_T(\Lambda) \stackrel{\text{def}}{=} \sum_{\gamma \in \Lambda} \frac{1}{T} \sum_{t=1}^{T} \varepsilon_t^\gamma.$$

We associate with this error a confidence bound and present our results (see Tables 5– 8) in the form:

$$E_T(\Lambda) \pm \frac{\sigma_T(\Lambda)}{\sqrt{T}}, \quad \text{where} \quad \sigma_T(\Lambda)^2 = \frac{1}{T} \sum_{t=1}^{T} \sum_{\gamma \in \Lambda} \left(\varepsilon_t^\gamma - E_T(\Lambda)\right)^2. \tag{3.13}$$

We choose the quantity $\sigma_T(\Lambda)/\sqrt{T}$ as it is reminiscent of the error margin provided by asymptotic confidence intervals on the mean of independent and identically distributed random variables.

In the next paragraph, we consider the "Region + NMF(16)" configuration and, for each of the three benchmark forecasting methods of Section 3.5 and for each of the three aggregation algorithms presented in Section 3.6, we compute these errors and confidence bounds for the four above foresting strategies. Finally, in the last paragraph, we set the benchmark forecasting method (generalized additive model) and the aggregation algorithm (ML-Pol) to test various households clusterings.

3.7.4.1 Impact of the Benchmark Forecasting Methods and of the Aggregation Algorithms

We consider here the two crossed hierarchies "Region + NMF (16)" and we vary the benchmark forecasting approaches and the aggregation algorithms. Indeed we compute forecasts for the three methods of Section 3.5 – auto-regressive model, generalized additive model and random forest – and for the three algorithms of Section 3.6 – NL-Ridge and BOA and ML-Pol. Table 5 sums up $E_T(\Gamma) \pm \sigma_T(\Gamma)/\sqrt{T}$, where Γ refers to the set of nodes associated with "Region + NMF (16)". Regarding forecasting methods, the general additive model provides the best benchmark predictions and the auto-regressive model, which is the most

naive method, does not perform well. This was actually already illustrated in Figures 10 and 11. Moreover, as the theory guarantees, projection (with or without an aggregation step) always improves the forecasts. The projection step without aggregation leads to a decrease of prediction error of around 1% for the general additive and auto-regressive models and of 5% for random forest. Note that for parametric (or semi-parametric) methods, the model is assumed to be the same at all nodes. Forecasts are thus closely linked and seem to almost already satisfy the hierarchical constraints. On the contrary, for random forest methods, the forecasts seem less correlated and thus projection improves significantly the predictions. The impact of aggregation step is notable: the prediction error decreases by about 10% for NL-Ridge and BOA and by about 15% for ML-Pol. Finally, our global strategy always gives the best forecasts, which, in addition, satisfy the hierarchical constraints.

	NL-Ridge	**ML-Pol**	**BOA**
General Additive Model			
Benchmark		455.5 ± 1.1	
Projection		450.7 ± 1.1	
Aggregation	407.6 ± 1.1	397.9 ± 1.0	406.0 ± 1.0
Aggregation + Projection	405.9 ± 1.1	396.0 ± 1.0	403.5 ± 1.0
Random Forest			
Benchmark		528.1 ± 1.0	
Projection		500.8 ± 1.0	
Aggregation	459.3 ± 1.0	467.3 ± 1.0	470.9 ± 1.0
Aggregation + Projection	451.1 ± 1.0	464.0 ± 1.0	468.1 ± 1.0
Auto-Regressive Model			
Benchmark		736.4 ± 1.6	
Projection		734.3 ± 1.6	
Aggregation	690.7 ± 1.6	690.1 ± 1.6	698.2 ± 1.6
Aggregation + Projection	689.8 ± 1.6	687.3 ± 1.6	693.1 ± 1.6

Table 5 – $E_T(\Gamma) \pm \sigma_T(\Gamma)/\sqrt{T}$ (see Equation 3.13) where Γ refers to the set of nodes associated with "Region + NMF (16)" clustering, for the three benchmark forecasting methods of Section 3.5 (General Additive Model, Random Forest and Auto-Regressive Model), for the three aggregation algorithms of Section 3.6 (NL-Ridge, ML-Pol and BOA) and for the four strategies defined in Subsection 3.7.4 ("Benchmark", "Projection", "Agregation" and "Aggregation + Projection"). $E_T(\Gamma)$ corresponds to $\widetilde{L}_T \times |\Gamma|$ for the "Aggregation + Projection" strategy. For strategies "Benchmark" and "Projection", the forecasts do not depend on the chosen aggregation algorithm, so the errors and the confidence bounds are the same for the three algorithms. The dark gray area corresponds to the best prediction error of the table and the light gray area to the best one, for a given benchmark forecasting method.

Even though theoretical guarantees (see Theorem 3.2) are only ensured for errors summed over all nodes, we investigate the impact of our methods on global consumption predictions and on most local predictions (i.e., predictions at leaves). Thus, Tables 6 and 7 contain $E_T\Big(\{\mathcal{I}\}\Big) \pm \sigma_T(\{\mathcal{I}\})/\sqrt{T}$ and $E_T(\Gamma_0) \pm \sigma_T(\Gamma_0)/\sqrt{T}$ (where Γ_0 is the set of leaves), respectively. By denoting by $R_1, \ldots, R_N$, the N regions and by $C_1, \ldots C_{16}$, the 16 clusters provided by "NMF (16)", we have, in this "Region + NMF (16)" configuration, $\Gamma_0 \stackrel{\text{def}}{=} \Big\{C_\ell \cap R_n\Big\}_{1 \leqslant \ell \leqslant 16,\, 1 \leqslant n \leqslant N}$. Concerning global consumption, a mere projection improves the forecasts, except in the case of auto-regressive model and, in all cases, our strategy "Ag-

gregation + Projection" outperforms the three strategies “Benchmark", “Aggregation" and “Projection". The prediction error associated with Γ_0 also decreases thanks to our procedure. Therefore, our method improves the forecasting of both global and local power consumptions. Finally, Figure 15 represents the global power consumption on the three last day of the testing period and the daily average signed error on the last week for the four forecasts obtained with features generated with general additive model and aggregated with ML-Pol algorithm. The distributions of the daily mean squared errors for these strategies are represented in Figure 16. We draw the same conclusions for the daily prediction errors as for the average error on the entire test period (three months): aggregation greatly improves the forecasts, projection does too, but to a lesser extent. The box plots show that the variance of the error also decreases after the aggregation step.

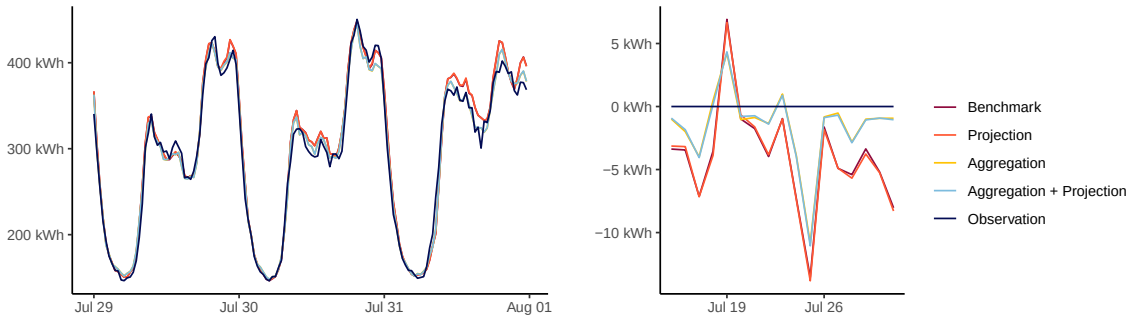


Figure 15 – Left picture: forecasts associated with the four strategies defined in Section 3.7.4 (“Benchmark", “Projection", “Agregation" and “Aggregation + Projection"), with benchmark forecasts generated with the generalized additive model and aggregated with ML-Pol algorithm in the “Region + NMF(16)" configuration, and observations of global consumption ($\gamma = \mathcal{I}$) at half-hour intervals on the last three days of the test period. Right picture: corresponding daily average signed errors on the last week of the test period.

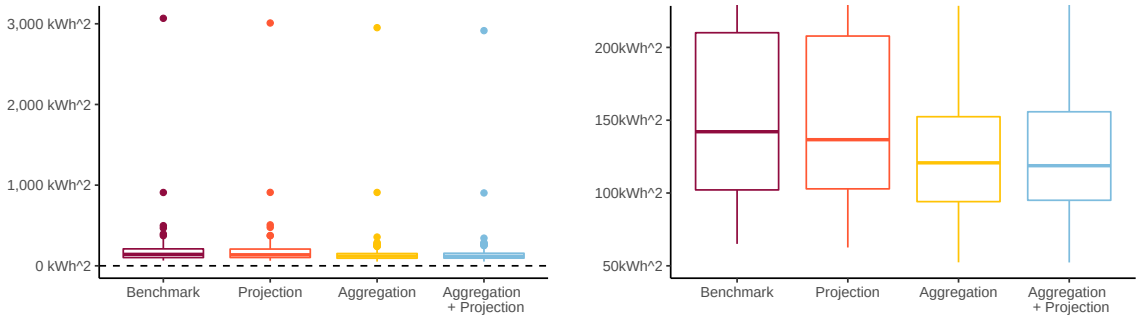


Figure 16 – Distribution over the test period of daily mean squared error of global consumption for the four strategies defined in Section 3.7.4 (“Benchmark", “Projection", “Agregation" and “Aggregation + Projection"), with benchmark forecasts generated with the generalized additive model and aggregated with ML-Pol algorithm in the “Region + NMF(16)" configuration. Left picture: original boxplots. Right picture: boxplots trimmed at 220 kWh^2.

3.7.4.2 Impact of the Clustering

We now assess the impact of household segmentation on the quality of our predictions. In view of the foregoing, we set the aggregation algorithm to ML-Pol and the benchmark fore-

	NL-Ridge	ML-Pol	BOA
General Additive Model			
Benchmark		205.8 ± 9.3	
Projection		200.8 ± 9.2	
Aggregation	179.2 ± 8.9	172.0 ± 8.6	178.8 ± 8.8
Aggregation + Projection	177.6 ± 8.8	170.3 ± 8.5	176.3 ± 8.7
Random Forest			
Benchmark		231.4 ± 8.6	
Projection		228.8 ± 8.2	
Aggregation	207.1 ± 8.4	214.8 ± 8.4	218.7 ± 8.3
Aggregation + Projection	206.4 ± 8.2	212.4 ± 8.1	216.8 ± 8.2
Auto-Regressive Model			
Benchmark		380.3 ± 13.4	
Projection		380.4 ± 13.4	
Aggregation	368.6 ± 13.5	370.8 ± 13.6	376.1 ± 13.6
Aggregation + Projection	368.2 ± 13.4	369.4 ± 13.5	373.6 ± 13.5

Table 6 – $E_T(\{\mathcal{I}\}) \pm \sigma_T(\{\mathcal{I}\})/\sqrt{T}$ (see Equation 3.13) for "Region + NMF (16)" clustering, for the three benchmark forecasting methods of Section 3.5 (General Additive Model, Random Forest and Auto-Regressive Model), for the three aggregation algorithms of Section 3.6 (NL-Ridge, ML-Pol and BOA) and for the four strategies defined in Subsection 3.7.4 ("Benchmark", "Projection", "Agregation" and "Aggregation + Projection"). The prediction error $E_T(\{\mathcal{I}\})$ corresponds to the mean squared error (over the testing period) of the global consumption. For strategies "Benchmark" and "Projection", the forecasts do not depend on the chosen aggregation algorithm, so the errors and the confidence bounds are the same for the three algorithms. The dark gray area corresponds to the best prediction error of the table and the light gray area to the best one, for a given benchmark forecasting method.

	NL-Ridge	ML-Pol	BOA
General Additive Model			
Benchmark		66.3 ± 0.1	
Projection		66.3 ± 0.1	
Aggregation	61.6 ± 0.1	61.2 ± 0.1	61.0 ± 0.1
Aggregation + Projection	61.5 ± 0.1	61.1 ± 0.1	61.0 ± 0.1
Random Forest			
Benchmark		78.7 ± 0.1	
Projection		68.9 ± 0.1	
Aggregation	66.8 ± 0.1	65.7 ± 0.1	64.8 ± 0.1
Aggregation + Projection	63.9 ± 0.1	65.7 ± 0.1	64.8 ± 0.1
Auto-Regressive Model			
Benchmark		84.4 ± 0.1	
Projection		84.3 ± 0.1	
Aggregation	73.8 ± 0.1	72.8 ± 0.1	73.2 ± 0.1
Aggregation + Projection	73.8 ± 0.1	72.0 ± 0.1	72.0 ± 0.1

Table 7 – $E_T(\Gamma_0) \pm \sigma_T(\Gamma_0)/\sqrt{T}$ (see Equation 3.13) where Γ_0 refers to the set of leaves associated with "Region + NMF (16)" clustering, for the three benchmark forecasting methods of Section 3.5 (General Additive Model, Random Forest and Auto-Regressive Model), for the three aggregation algorithms of Section 3.6 (NL-Ridge, ML-Pol and BOA) and for the four strategies defined in Subsection 3.7.4 ("Benchmark", "Projection", "Agregation" and "Aggregation + Projection"). $E_T(\Gamma_0)$ corresponds to a prediction errors associated with local consumptions forecasts. For strategies "Benchmark" and "Projection", the forecasts do not depend on the chosen aggregation algorithm, so the errors and the confidence bounds are the same for the three algorithms. The dark gray area corresponds to the best prediction error of the table and the light gray area to the best one, for a given benchmark forecasting method.

casting method to the general additive model. As clusters change from a segmentation to another, the associated sets of nodes Γ also change. Errors related to Γ or Γ_0 can therefore not be compared from a segmentation to another. We thus focus here on the global consumption (namely, we compute errors related to $\{\mathcal{I}\}$). We compare our methods to a naive bottom-up strategy: at each instance t, we forecast the global consumption $y_t^{\{\mathcal{I}\}}$ with the sum of local consumptions $\sum_{\gamma\in\Gamma_0} x_t^\gamma$ – in lieu of the benchmark predictions $x_t^{\{\mathcal{I}\}}$. Table 8 contains the prediction errors and the confidence bounds for the five strategies and for several household segmentations. For the “Bottom-up" strategy, the geographical clustering “Region" provides the lowest prediction error, that are much better than the one of benchmark forecasts. While when a single clustering based on household profiles or generated randomly is considered, the benchmark forecasts $x_t^{\{\mathcal{I}\}}$ are more relevant – in terms of mean squared error. Thus, taking into account regional consumptions, which depend on local meteorological variables, improves prediction. In the same way, projection significantly improves the forecasts when the regions are taken into account. Moreover, for a fixed number of clusters – for example, we compare “Fuel+Tariff", “Random (4) and“NMF (4)" – the aggregation step seems more efficient when clusters present different consumption profiles (see Figures 5 - 9). Indeed, aggregation provides much better performance for “NMF (4)" than for “Random (4)". As we had anticipated, contrary to “NMF" and “Region", clusterings “Acorn" and “Fuel + Tariff", that do not seem to detect consumption profiles, perform as well as “Random". When the number of clusters becomes too large, the performance of the strategy stagnates or even decreases. Typically for “Random" or “NMF", a number of clusters equals to 32 or 64 does not seem to improve the results compared to smaller numbers 4, 8 or 16. Another result is that aggregation and projection are robust to large number of clusters. Indeed, the performance are good for a sufficiently large number of clusters but does not decrease too much with the number of clusters – either for “ Random" or “NMF" clusterings. Finally, our strategy “Aggregation + Projection" always outperforms the other four (“Bottom-up", “Benchmark", “Projection" and “Aggregation") and the “Region + NMF (16)" clustering reaches the lowest prediction error.

Clustering	Benchmark	Bottom-up	Projection	Aggregation	Aggregation + Projection
Region	205.8 ± 9.3	189.9 ± 8.3	201.3 ± 9.1	187.8 ± 8.4	186.7 ± 8.4
Region + Acorn	**—**	$\mathbf{194.2 \pm 8.4}$	$\mathbf{200.8 \pm 9.2}$	$\mathbf{182.5 \pm 8.3}$	$\mathbf{181.2 \pm 8.3}$
Acorn	—	205.7 ± 9.5	205.0 ± 9.3	203.3 ± 9.3	202.9 ± 9.3
Region + Fuel + Tariff	**—**	$\mathbf{199.1 \pm 8.7}$	$\mathbf{201.2 \pm 9.2}$	$\mathbf{185.4 \pm 8.6}$	$\mathbf{184.1 \pm 8.6}$
Fuel + Tariff	—	207.1 ± 9.7	205.5 ± 9.4	201.5 ± 9.4	201.4 ± 9.5
Region + Random (4)	**—**	$\mathbf{198.4 \pm 8.7}$	$\mathbf{201.3 \pm 9.2}$	$\mathbf{186.1 \pm 8.6}$	$\mathbf{184.6 \pm 8.6}$
Random (4)	—	208.0 ± 9.7	205.7 ± 9.4	199.5 ± 9.4	199.7 ± 9.4
Region + Random (8)	**—**	$\mathbf{202.3 \pm 8.7}$	$\mathbf{201.3 \pm 9.2}$	$\mathbf{182.4 \pm 8.7}$	$\mathbf{181.0 \pm 8.7}$
Random (8)	—	212.9 ± 9.8	205.7 ± 9.3	194.4 ± 9.1	194.4 ± 9.1
Region + Random (16)	**—**	$\mathbf{205.1 \pm 8.7}$	$\mathbf{201.3 \pm 9.2}$	$\mathbf{180.5 \pm 8.7}$	$\mathbf{178.8 \pm 8.7}$
Random (16)	—	218.4 ± 10.0	205.7 ± 9.3	188.6 ± 8.7	188.5 ± 8.7
Region + Random (32)	**—**	$\mathbf{205.3 \pm 8.5}$	$\mathbf{201.2 \pm 9.2}$	$\mathbf{180.4 \pm 8.8}$	$\mathbf{178.9 \pm 8.7}$
Random (32)	—	222.9 ± 10.1	205.6 ± 9.3	189.6 ± 8.7	189.5 ± 8.7
Random (64)	—	222.9 ± 9.8	205.6 ± 9.3	185.7 ± 8.8	185.5 ± 8.8
Region + NMF (4)	**—**	$\mathbf{196.0 \pm 8.6}$	$\mathbf{200.8 \pm 9.2}$	$\mathbf{187.4 \pm 9.1}$	$\mathbf{185.5 \pm 8.9}$
NMF (4)	—	205.7 ± 9.5	205.0 ± 9.3	197.0 ± 8.8	196.8 ± 8.9
Region + NMF (8)	**—**	$\mathbf{197.2 \pm 8.5}$	$\mathbf{200.7 \pm 9.2}$	$\mathbf{176.4 \pm 8.9}$	$\mathbf{174.1 \pm 8.8}$
NMF (8)	—	206.7 ± 9.6	205.0 ± 9.3	186.1 ± 8.9	185.7 ± 8.9
Region + NMF (16)	**—**	$\mathbf{201.0 \pm 8.5}$	$\mathbf{200.8 \pm 9.2}$	$\mathbf{172.0 \pm 8.6}$	$\mathbf{170.3 \pm 8.5}$
NMF (16)	—	208.4 ± 9.6	205.2 ± 9.3	179.3 ± 8.4	179.3 ± 8.4
Region + NMF (32)	**—**	$\mathbf{204.1 \pm 8.5}$	$\mathbf{201.0 \pm 9.1}$	$\mathbf{173.2 \pm 8.7}$	$\mathbf{171.5 \pm 8.6}$
NMF (32)	—	211.1 ± 9.6	205.4 ± 9.3	179.7 ± 8.8	179.5 ± 8.8
NMF (64)	—	214.9 ± 9.4	205.6 ± 9.3	181.9 ± 8.6	181.7 ± 8.6

Table 8 – $E_T(\{\mathcal{I}\}) \pm \sigma_T(\{\mathcal{I}\})/\sqrt{T}$ (see Equation 3.13) for the fives strategies defined in Subsection 3.7.4 (“Benchmark", “Bottom-up, “Projection", “Agregation" and “Aggregation + Projection"), with benchmark predictions ($x_t^{\{\mathcal{I}\}}$ that are the same for all clusterings) made with General Additive Models and aggregated with ML-Pol algorithm, for many segmentations (defined in Subsection 3.7.2). The prediction error $E_T(\{\mathcal{I}\})$ corresponds to the mean squared error (over the testing period) of the global consumption. The dark gray area corresponds to the best prediction error of the table and the light gray area to the best one, for a given strategy.

CHAPTER 4

Hierarchical robust aggregation of demand forecasts in e-commerce

In this chapter revisit the interest of using classical statistical techniques for sales forecasting like exponential smoothing and extensions thereof (as Holt's linear trend method). We do so by considering ensemble forecasts, given by several instances of these classical techniques tuned with different (sets of) parameters, and by forming convex combinations of the elements of ensemble forecasts over time, in a robust and sequential manner. The machine-learning theory behind this is called "robust online aggregation", or "prediction with expert advice", or "prediction of individual sequences" (see Cesa-Bianchi and Lugosi, 2006). We apply this methodology to a hierarchical data set of sales provided by the e-commerce company Cdiscount and output forecasts at the levels of subsubfamilies, subfamilies and families of items sold, for various forecasting horizons (up to 6–week-ahead). The performance achieved is better than what would be obtained by optimally tuning the classical techniques on a train set and using their forecasts on the test set. The performance is also good from an intrinsic point of view (in terms of mean absolute percentage of error). While getting these better forecasts of sales at the levels of subsubfamilies, subfamilies and families is interesting per se, we also suggest to use them as additional features when forecasting demand at the item level.

Contents

This chapter is joint work with Rémy Garnier and Gilles Stoltz, it is currently under review for publication (see Huard et al., 2020). This work was financially supported by AMIES ["Agence pour les mathématiques en interaction avec l'entreprise et la société", a French agency dedicated to interactions of mathematics with business and society] and the company Cdiscount.

4.1 Introduction and literature review

Sales data in e-commerce are highly dynamic and volatile: reactive methods are required (and these methods are often sophisticated). We provide a detailed discussion on these newer methods in Section 4.1.5; they stem from the machine learning toolbox. On the other hand, in retail merchandising, classical statistical techniques for sales forecasting like exponential smoothing and extensions thereof (as Holt's linear trend method) are effective and have been widely used since the 1950s (see Gardner, 1985, 2006 and Hyndman et al., 2008). Other such classical techniques include autoregressive models like ARIMA and its variants (Box et al., 1994). A review of the use of these classical techniques may be found in the monograph by Chatfield [2000].

The aim of this chapter is to forecast sales in e-commerce based on exponential smoothing and extensions thereof. By "based on", we mean that two layers will be considered in our methodology: the first layer is to build several instances of exponential smoothing and Holt's linear trend method (tuned with different parameters). They will be called elementary predictors. The forecasts of these elementary predictors are then combined, prediction step after prediction step, via a so-called aggregation algorithm (see Cesa-Bianchi and Lugosi, 2006 for an introduction to the field of robust online aggregation). The aggregation algorithms considered output convex weights, that evolve over time in a reactive way depending on performance, and the aggregated forecasts are simply given by convex combinations of the forecasts issued by the elementary predictors.

Since we are dealing with e-commerce data, the items considered are grouped into a hierarchy (of subsubfamilies, subfamilies, and families of products). We only forecast sales at these aggregated levels (not for individual items), which, admittedly, is an easier forecasting task (see Mentzer and Cox, 1984). We do so by aggregating the forecasts of elementary predictors separately at each node of the hierarchy and by reconciling the thus obtained aggregated forecasts through a projection. Cross-series information is thus shared through the hierarchical constraints. Our methodology is fully automated, scalable, and robust—three key requirements stated by Seeger et al. [2016].

Sales forecasting at these aggregated levels may be considered interesting per se, but we also see it as a way to obtain extra features for sales forecasting at the item level; these extra features (demand forecasts for all items of the same subsubfamily) can then be provided as an extra input to the sophisticated and reactive machine-learning methods currently constructed (see Section 4.1.5 for a more detailed literature review).

4.1.1 Presentation of the problem of hierarchical forecasting and of the data set

What follows is detailed in Sections 4.2.1 and 4.3.1. Our data was provided by the e-commerce company Cdiscount and spans from July 2014 to December 2017—a period of 182 weeks. We use July 2014 to December 2016 as a training period (containing 130 weeks), and January 2017 – December 2017 (containing 52 weeks) as a test period; the test period thus features all major commercial events (sales, Black Friday and Christmas shopping, etc.). The data set features the daily sales of 620,749 items hierarchically ordered in 3,004 subsubfamilies, 570 subfamilies and 53 families. We add up daily sales to get weekly sales. Many

time series of weekly sales thus created are intermittent (but as will get clearer in the sequel, we do not apply any specific trick or tool to deal with intermittent demand).

Our notion of a hierarchy means that we organize the subsubfamilies, subfamilies and families into a tree Γ, whose root node consists of total sales. The sales (numbers of units sold, or money value) achieved at a node γ (i.e., for a given subsubfamily, subfamily or family) during week t are denoted by $s_{t,\gamma}$. Summation constraints are considered: e.g., if $\gamma \in \Gamma$ is some (sub)family and $\mathcal{C}(\gamma)$ denotes the (sub)subfamilies that belong to it, we have

$$s_{t,\gamma} = \sum_{c \in \mathcal{C}(\gamma)} s_{t,c} \, .$$

An arbitrary collection of forecasts $\widehat{s}_{t+h,\gamma}$ of the sales at an horizon of h weeks, where γ spans the tree Γ, may be transformed into a collection $\widetilde{s}_{t+h,\gamma}$ of such forecasts abiding by the summation constraints indicated by Γ by a projection onto a suitable vector space. We further detail this in Section 4.2.4. Such a projection actually shares information between related subsubfamilies, subfamilies and families.

Related literature on hierarchical forecasting. We provide hierarchical predictions but in a simple manner, actually in the simplest possible manner: by independently computing forecasts at each node of the hierarchy and by reconciling them by a projection step. For a description of fancier approaches to hierarchical forecasting, we refer to the specific literature review provided in the Section 3.1 of Chapter 3.

4.1.2 Robust aggregation (a.k.a. prediction with expert advice, prediction of individual sequences)

The methodology discussed in this section is described in detail in Sections 4.2.2 and 4.2.3. It aims at providing node-by-node forecasts (series of forecasts for each given node $\gamma \in \Gamma$ of the hierarchy).

Our methodology relies on ensemble forecasts (Section 4.2.2): several elementary predictors are considered, all of them but a few given by instances of exponential smoothing or Holt's linear trend method, with different sets of parameters. As the series of sales all exhibit some seasonality, but with different cycles depending on the considered node γ, as some have a linear trend and some others do not, as some are highly regular while some others exhibit a more erratic behavior, it is clear that no single instance of exponential smoothing or Holt's linear trend method can be simultaneously suited for all series. A typical way to deal with this issue is to tune the parameters on a train set and use the thus-tuned method on the test set; i.e., to select a given elementary predictor. We compute the results thus obtained on our data set and show that they are consistently inferior to the methodology followed, which relies on aggregating (combining) the forecasts of all the elementary predictors.

There are various techniques to aggregate forecasts via machine-learning or statistical methods. Some of these aggregation techniques deal with stochastic data: the observations to be forecast are modeled by some stochastic process. On the contrary, other techniques work on deterministic data and come with theoretical guarantees of performance even when the observations cannot be modeled by a stochastic process. Examples of popular aggrega-

tion methods include Bayesian model averaging (see Hoeting et al., 1999 for a tutorial and Raftery et al., 2005 for an application to ensemble forecasts) and random forests (introduced by Breiman, 2001), both of them being stochastic approaches, as well as robust online aggregation, which is a deterministic approach. We are interested in the latter approach, given the erratic nature of the series of sales in e-commerce (they are notoriously difficult to model).

Robust online aggregation is also known as prediction of individual sequences, or prediction with expert advice (see the monograph by Cesa-Bianchi and Lugosi, 2006 and references therein, see also the numerous references provided in Section 4.2.3). This sequential aggregation technique, developed in the 1990s, provides a robust framework to make forecasts on a regular (e.g., weekly) basis. It does not rely on any specific assumption or need for stochastic modeling; it may handle any (bounded) time series, possibly extremely erratic. At each time step, a weighted average of the forecasts of the elementary predictors is issued, where the (convex) weights used are picked based on the past performance of the elementary predictors. These weights thus change over time, which guarantees that the aggregation algorithm may quickly adapt to changes in the environment, a key feature for e-commerce that batch forecasting methods (the methods that use a train set) do not possess. In a nutshell, the robust online aggregation algorithms considered are online and adaptive by nature, which is an advantage over batch methods that are less often updated.

These robust online aggregation algorithms also come with strong theoretical guarantees of performance: they almost achieve or outperform the performance of the best elementary predictor (and sometimes, the best constant convex combination of elementary predictors).

Previous successful applications of robust aggregation. As the methodology described above does not rely on any specific assumption or need for stochastic modeling, and is therefore extremely general, it was already successfully applied on different applications. The R package Opera written by Gaillard and Goude [2016] is now a popular tool to use this methodology and it is difficult to cite all applications already performed. However, among them, we may cite the forecasting of air quality (Mallet et al., 2009), of electricity load (Devaine et al., 2013; Gaillard and Goude, 2015 and Chapter 3), of exchange rates (Amat et al., 2018), of oil and gas production (Raphaël et al., 2019).

However, while the methodology is general, the application to each specific domain is still challenging: some theoretical adaptations might be needed (in the present case, dealing with a hierarchy), and more importantly, proper elementary predictors need to be designed. The ones used for the forecasting of electricity load are actually quite fancy (see a specific discussion below, in Section 4.1.5), and the same can be said for air quality (complex PDE models with different data inputs, see Mallet et al., 2009) and oil and gas production (complex numerical solvers were used to model the production fields, see Raphaël et al., 2019). In the present chapter, we want to show that classical and simple time series methods like exponential smoothing and Holt's linear trend method can be useful elementary predictors. Of course, more complex forecasting models (using more side information) could be used as elementary predictors.

Also, in most references of this paragraph, aggregation algorithms outputting linear weights were considered (e.g., ridge regression), while we restrict our attention to convex weights (to get safer predictions: within the range of forecasts issued by elementary predictors).

4.1.3 Brief summary of the numerical results obtained

The numerical results obtained are discussed in detail in Section 4.3. We illustrate the good performance of our forecasting methodology in two manners.

First, we provide a study of relative performance and show that the aggregation algorithms considered consistently outperform the natural benchmark given by the best locally predictors on the train set (i.e., what is achieved by selecting, for each node, the best elementary predictor on the train set, and by using it on the test set), by about 5%. This observation holds in mean absolute error [MAE] and in root mean square error [RMSE], for various forecasting horizons (from 1–week-ahead to 6–week-ahead). We note that the performance of the aggregation algorithms does not vary much by the algorithm.

Second, we study the absolute (intrinsic) performance achieved, by reporting mean absolute percentages of errors. Aggregation algorithms obtain a global MAPE of about 20% (again, this is valid for different forecasting horizons). This MAPE can be broken down by the level: it equals about 30% for subsubfamilies. These low values correspond to the consideration of aggregated levels; we do not work at the item level.

Finally, we provide some graphical evolutions of convex weights picked over time, for different families and for total sales. In general, these weights change much over time, which illustrates the flexibility and reactivity of the aggregation algorithms over time.

4.1.4 Outline of the chapter

The chapter is organized as follows. Section 4.1.5 reviews the literature on demand forecasting, including the approaches specific to e-commerce. Section 4.2 presents the methodology followed while Section 4.3 discusses the results obtained on our data set.

More precisely, Section 4.2 starts with a statement of our setting of hierarchical prediction of sales (Section 4.2.1). It then describes the elementary predictors considered, based on exponential smoothing or on Holt's linear trend method (Section 4.2.2). The aggregation methodology briefly hinted at above is described in details in Section 4.2.3.1 and three specific aggregation algorithms are stated, and adapted where needed, in Section 4.2.3.2 (and a general trick to boost their performance is provided in Section 4.2.3.3). However, Section 4.2.3 is only concerned with node-by-node aggregation and this is why Section 4.2.4 explains how the node-by-node aggregation results may be extended for the entire hierarchy of nodes.

Then, Section 4.3 first provides a detailed description of the real data set considered and of its division into a train set and a test set (Section 4.3.1), and discusses the performance of the elementary predictors at various forecasting horizons (Section 4.3.2). The main results consist of a tabulation of the performance achieved by the three aggregation algorithms studied, in MAE and RMSE (Section 4.3.3) and in MAPE (Section 4.3.4). Two complementary studies are finally provided: on the distributions of errors (Section 4.3.5) and on the evolution of the weights put on each elementary predictor by the aggregation algorithms (Section 4.3.6).

4.1.5 Specific literature review on demand forecasting

So far, we only discussed general references on time-series predictions (for exponential smoothing, Holt's linear trend method, ARIMA models) and on ensemble methods, and in particular, on robust aggregation (also known as prediction with expert advice or prediction of individual sequences, see Section 4.1.2). This is because our approach is designed to be general and independent of the specific context of application. However, we now provide a literature review focused on the goal of the present contribution, namely, demand forecasting, and even more precisely, demand forecasting for e-commerce.

Demand forecasting tackles the prediction of the level of demand for a product or a service in the future. This demand may not match the exact number of sales for a product for different reasons (stock shortage, change of prices). Demand forecasts has various applications, among others: electric load forecasting (see Alfares and Nazeeruddin, 2002 for a survey, see also the aforementioned contributions by Devaine et al., 2013; Gaillard and Goude, 2015 and Chapter 3); urban water demand forecasts (see Emmanuel A. Donko et al., 2014 for a survey); and sales forecasting. General surveys on sales forecasting (not centered on e-commerce) were written by Beheshti-Kashi et al. [2015] and Carbonneau et al. [2008]. These applications differ on a number of criteria. First, the demand variable may be continuous (case of electric load) or discrete (case of sales in retail business), with different aggregated times step (daily, weekly, monthly). In some cases, the demand variable may also be intermittent (see Xu et al., 2012 and Seeger et al., 2016 for examples and details). Second, the forecasting horizons differ between the considered applications, from short-term prediction to longer-term horizons. The exact definition of short- and long-term may differ with applications, but a prediction horizon of more than two week is a long-term horizon for most applications. The issue is that the generally best method for a given problem may differ for long-term and short-term predictions (see discussions by Emmanuel A. Donko et al., 2014). Third, the dimensions of the demand variables may differ. In the simplest case, a unique value for a given time step is to be predicted; however, in more complex cases, several values are to be predicted, for exemple, levels of sales of multiple items at a given or at various time steps. These multiple items may be organized in a hierarchy of products (as we do) or in related groups (see Chapados, 2014). This is why each of these applications presents some specific challenges to tackle. We now detail two popular applications: electricity load forecasting and sales forecasting for e-commerce.

Electricity load forecasting. Traditional time series methods (based on exponential smoothing or autoregressive models, and their extensions like Holt's linear trend method or ARIMA models) have of course been extensively used and tailored to the needs of this application. For instance, a lot of attention was put to add seasonality to this kind of models (see Taylor, 2003, 2010 for recent examples).

Other modern machine statistical methods have been introduced to overcome the limitation of traditional times series methods. We cite two of them. First, generalized additive models [GAMs], used in a autoregressive way (with past load values as features) and with additional covariates (e.g., meteorological variables), are now a standard and efficient method to forecast the electricity load, at least for short-term horizons; see Pierrot and Goude [2011] and Wijaya et al. [2015], as well as their use by Devaine et al. [2013] combined with robust

aggregation. Such GAM models rely on a discretization of the load into a sequence of values within a day by considering aggregated time steps, typically, half hours. On the contrary, a second family of statistical models directly predicts load curves; see Antoniadis et al. [2006] and the discussions therein. Of course, other methods, from the machine learning community, that may suffer from a lack of interpretability, were also considered, like random forests: see Dudek [2015].

We may now provide a detailed comparison to the study of Chapter 3, as promised in Section 4.1.1. The focus therein is the short-term forecasting (one day ahead) of electricity load. Customers are grouped into a hierarchy (created by clustering) so that the elementary predictors considered for each cluster (based on sophisticated GAM models or on random forests) can be better adjusted. Predictions are then formed cluster by cluster through robust aggregation algorithms and reconciled through a projection step, exactly as in the present chapter. Actually, the present chapter was initiated before and inspired the study of Chapter 3. More importantly, our focus here is to consider elementary predictors that are truly elementary and general—and so are predictors based on exponential smoothing and Holt's linear trend method, while GAM models or random forest are not. The latter are powerful methods that are already efficient per se.

Sales forecasting for e-commerce. This special case of demand forecasting comes with the following specific difficulties. First, the number of items in e-commerce is generally large (much larger than in traditional retail); these items are organized in a hierarchy of (subsub)families, as described in Section 4.1.1. Second, modern considerations in logistics and supply chain tend to limit supplies and emphasizes *just-on-time* resupply. This implies that e-commerce companies generally need medium-term prediction for their sales, typically around 1-month (or 1-month-and-a-half) ahead. However, most of the existing predictive models for supply chain were linear and were not able to deal with the more erratic behaviour of real-world sales data in e-commerce. Moreover, they were not able to exploit cross-product information. This is why virtually all of the forecasting methods for sales in e-commerce rely on sophisticated techniques stemming from the machine-learning community. To name just a few, let us recall that a Bayesian modeling relying on a hierarchical state-space model was proposed for sales data by Chapados [2014] (and it allows to share information between products). Neural network models have also been widely used, e.g., Bandara et al. [2019] used a recurrent neural network for e-commerce sales data. Finally, Amazon developed a probabilistic neural network for demand forecast called DeepAR, described by Salinas et al., 2019. All these sophisticated methods are difficult to tune and maintain because they rely on a large number of parameters; in contrast, our methodology is simple, computationally efficient, and fully automated (once the elementary predictors are chosen).

4.2 Setting

In this section we first describe the aim of the forecasting task (Section 4.2.1), the elementary predictors considered, including Holt's linear trend predictors (Section 4.2.2), and the aggregation methodology followed. The latter first takes place node by node (Section 4.2.3) and then is extended to hold for the entire hierarchy of nodes (Section 4.2.4). The description of the node-by-node aggregation will be broken down into a general presentation of the concept of aggregation (Section 4.2.3.1), the statement of three specific aggregation algorithms considered in the sequel (Section 4.2.3.2), and the description of the "gradient trick" (Section 4.2.3.3), which is a general trick to boost the performance of aggregation algorithms.

4.2.1 Aim: hierarchical prediction of sales

The products sold are grouped in a hierarchical way, given by a tree Γ; nodes of the tree will be indexed by γ. The root of Γ gathers all products. The children of the root are called families, and are further broken down into subfamilies, and then subsubfamilies. The leaves of the tree correspond to the products. A product corresponds to a unique subsubfamily, which itself corresponds to a unique subfamily, which itself corresponds to a unique family.

We consider weekly sales, where weeks are indexed by $t \in \{1, 2, \ldots\}$. We denote by $s_{t,\gamma}$ the sales achieved for family γ during week t. They can be measured in units or in total value. The aim is to predict sales at all nodes of the hierarchy Γ, at a given horizon $h \geqslant 1$; that is, to issue forecasts of the future quantities

$$s_{t+h,\gamma}, \quad \gamma \in \Gamma\,.$$

This aim was expressed to predict the sales during $n = 1$ week, but we may possibly group $n \geqslant 2$ weeks, with $n \leqslant h$, and forecast the quantities

$$y_{t+h,\gamma} = \frac{1}{n} \sum_{\tau=t+h-n+1}^{t+h} s_{\tau,\gamma}, \quad \gamma \in \Gamma\,,$$

which correspond to average sales over a period of n weeks ending at the horizon of h weeks. Put differently, the goal is to forecast $h - n$–week-ahead a group of n weeks (the group of n weeks starts at week $t + h - n + 1$ after $h - n$ complete weeks have passed after the current week t).

The y and the s are equal in case $n = 1$, and this is why, with no loss of generality, we only discuss below the forecast of the y. We defined the y as averages for them to all share the same order of magnitude, independently of the value of $n \geqslant 1$. Typical values for n are in $\{1, 2, 3, 4\}$.

Summation constraints. The sales achieved at a given node are the sum of the sales achieved at its children nodes. More formally, denoting by $\mathcal{C}(\gamma)$ the children of a given node $\gamma \in \Gamma$, we have, whenever $\mathcal{C}(\gamma)$ is not the empty set:

$$y_{t+h,\gamma} = \sum_{c \in \mathcal{C}(\gamma)} y_{t+h,c}\,.$$

It is thus natural to expect that the forecasts $\widehat{y}$ of the y satisfy the same summation constraints: for all $\gamma \in \Gamma$ with non-empty set $\mathcal{C}(\gamma)$ of children nodes,

$$\widehat{y}_{t+h,\gamma} = \sum_{c \in \mathcal{C}(\gamma)} \widehat{y}_{t+h,c}\,.$$

4.2.2 Elementary predictors / node by node

In this section, we fix a given node $\gamma \in \Gamma$ and describe the elementary forecasting methods considered.

We introduce three sets or families of elementary forecasting methods (or elementary predictors): simple exponential smoothing, with an additive or a multiplicative treatment of seasonality, relying on a parameter $\alpha \in [0,1]$; Holt's linear trend method, with an additive or a multiplicative treatment of seasonality, relying on parameters $\alpha \in [0,1]$ and $\beta \in [0,1]$; other elementary forecasts, provided by benchmarks. Simple exponential smoothing and Holt's linear trend methods are popular methods for demand forecasting in e–commerce (see Bandara et al., 2019).

Note that valid forecasts for the $y_{t+h,\gamma}$ quantities must rely only on present and past sales, i.e., on sales $s_{\tau,\gamma}$ with $\tau \leqslant t$. In particular, present and past average sales $y_{\tau,\gamma}$, with $\tau \leqslant t$, may be used.

Other elementary forecasts. They consist of

- the sales achieved one year (52 weeks[1]) ago, $\widehat{y}_{t+h,\gamma} = y_{t+h-52,\gamma}$;
- the sales currently achieved, $\widehat{y}_{t+h,\gamma} = y_{t,\gamma}$;
- the null sales, $\widehat{y}_{t+h,\gamma} = 0$, given that a significant number of pairs of subsubfamilies and weeks have no sales (most time series of sales are sparse, i.e., the demand of the corresponding is intermittent; see the sparsity statistics provided in Section 4.3.1).

Simple exponential smoothing with an additive treatment of seasonality. We use simple exponential smoothing to forecast the difference $d_{t+h,\gamma}$ between the quantity of interest, $y_{t+h,\gamma}$, and its value one year ago, $y_{t+h-52,\gamma}$. This is a first (additive) way for taking seasonality into account. Each instance of simple exponential smoothing is parameterized by a number $\alpha \in [0,1]$.

More precisely, given the needed history, forecasts can only be issued after week t_0 (whose value is indicated below) and are provided by

$$\widehat{d}_{t_0+h,\gamma} = d_{t_0,\gamma} \qquad \text{and for } t \geqslant t_0+1, \qquad \widehat{d}_{t+h,\gamma} = \alpha\, d_{t,\gamma} + (1-\alpha)\, \widehat{d}_{t-1+h,\gamma}\,;$$

that is, $\widehat{y}_{t_0+h,\gamma} = y_{t_0+h-52,\gamma} + (y_{t_0,\gamma} - y_{t_0-52,\gamma})$ and more generally, for $t \geqslant t_0$,

$$\widehat{y}_{t+h,\gamma} = y_{t+h-52,\gamma} + \sum_{j=0}^{t-t_0-1} \alpha(1-\alpha)^j \Big(y_{t-j,\gamma} - y_{t-j-52,\gamma}\Big) + (1-\alpha)^{t-t_0}\Big(y_{t_0,\gamma} - y_{t_0-52,\gamma}\Big)\,.$$

[1]Here, and at all subsequent places, the value 52 weeks for a year could be replaced by 53 weeks, which works equally well. To alleviate notation, we did not set a parameter T_{year} for this value but could have done so, of course.

The threshold t_0 is such that the y with the smallest time index above, that is, y_{t_0-52}, is well defined; it is defined as an average of n weekly sales starting at time $t_0 - 52 - n + 1$, which must be at least 1. Thus, $t_0 = 52 + n$.

Simple exponential smoothing with a multiplicative treatment of seasonality. A second (multiplicative) way for handling seasonality is to replace the difference $d_{t+h,\gamma} = y_{t+h,\gamma} - y_{t+h-52,\gamma}$ by the ratio $y_{t+h,\gamma}/y_{t+h-52,\gamma}$. We actually consider a variant of this ratio, given by

$$z_{t+h,\gamma} = y_{t+h,\gamma}/r_{t+h-52,\gamma}\,, \qquad \text{where} \qquad r_{\tau,\gamma} = \frac{y_{\tau,\gamma}}{\sum_{j=-26}^{25} y_{\tau+j,\gamma}}$$

denotes, for τ large enough, the ratio between the sales $y_{\tau,\gamma}$ for a given week τ and yearly sales centered at this week. Simple exponential smoothing is then used to forecast the z quantities.

More precisely, given the needed history, forecasts can only be issued after a given week t'_0 (whose value is indicated below) and are given by

$$\widehat{z}_{t'_0+h,\gamma} = z_{t'_0,\gamma} \qquad \text{and, for } t \geqslant t'_0 + 1, \qquad \widehat{z}_{t+h,\gamma} = \alpha\, z_{t,\gamma} + (1-\alpha)\widehat{z}_{t-1+h,\gamma}\,.$$

(We skip the closed-form expressions that could be derived for the $\widehat{z}_{t+h,\gamma}$.) The forecasts of the quantities of interest are then provided, for $t \geqslant t'_0$, by

$$\widehat{y}_{t+h,\gamma} = r_{t+h-52,\gamma}\,\widehat{z}_{t+h,\gamma}\,.$$

The threshold t'_0 after which forecasts $\widehat{y}_{t'_0+h}$ can be issued is such that $\widehat{z}_{t+h,\gamma} = z_{t'_0,\gamma}$ and $r_{t'_0+h-52,\gamma}$ are well defined. It is necessary and sufficient to that end that $r_{t'_0-52,\gamma}$ be well defined. The latter is an average of values y_τ starting at the index $t'_0 - 52 - 26$; the starting value $y_{t'_0-52-26}$ is itself an average of n weekly sales starting at time $t'_0 - 52 - 26 - n + 1$, which must be at least 1. Thus, $t'_0 = 52 + 26 + n$.

Holt's linear trend method with a multiplicative treatment of seasonality. We extend and generalize the approach followed in the previous paragraph by allowing for a trend. Two parameters $\alpha \in [0,1]$ and $\beta \in [0,1]$ are set. The forecasting equations are, for $t \geqslant t'_0 + 2$ (where t'_0 was defined in the previous paragraph):

$$\begin{aligned} &\text{[level]} && \ell_{t+h,\gamma} = \alpha\, z_{t,\gamma} + (1-\alpha)(\ell_{t-1+h,\gamma} + b_{t-1+h,\gamma}) \\ &\text{[trend]} && b_{t+h,\gamma} = \beta(\ell_{t+h,\gamma} - \ell_{t-1+h,\gamma}) + (1-\beta) b_{t-1+h,\gamma} \end{aligned}$$

with an initialization consisting of

$$\ell_{t'_0+1+h,\gamma} = z_{t'_0+1,\gamma} \qquad \text{and} \qquad b_{t'_0+1+h,\gamma} = z_{t'_0+1,\gamma} - z_{t'_0,\gamma}\,.$$

The forecasts of the quantities of interest are then provided, for $t \geqslant t'_0 + 1$, by

$$\widehat{y}_{t+h,\gamma} = r_{t+h-52,\gamma}\left(\ell_{t+h,\gamma} + h\, b_{t+h,\gamma}\right).$$

Remark. The choice $\beta = 0$ with the initialization $b_{t'_0+1+h,\gamma} = 0$ (so that all b values are null) corresponds to simple exponential smoothing.

Holt's linear trend method with an additive treatment of seasonality. We finally extend simple exponential smoothing with an additive treatment of seasonality by also allowing for a trend; we use again the time $t_0 = 52 + n$ defined therein. The forecasting equations are, for $t \geqslant t_0 + 2$,

$$\begin{aligned} &\text{[level]} && \ell_{t+h,\gamma} = \alpha(y_{t,\gamma} - y_{t-52,\gamma}) + (1-\alpha)(\ell_{t-1+h,\gamma} + b_{t-1+h,\gamma}) \\ &\text{[trend]} && b_{t+h,\gamma} = \beta(\ell_{t+h,\gamma} - \ell_{t-1+h,\gamma}) + (1-\beta) b_{t-1+h,\gamma} \end{aligned}$$

with an initialization consisting of

$$\ell_{t_0+1+h,\gamma} = y_{t_0+1,\gamma} - y_{t_0+1-52,\gamma} \quad \text{and} \quad b_{t_0+1+h,\gamma} = (y_{t_0+1,\gamma} - y_{t_0+1-52,\gamma}) - (y_{t_0,\gamma} - y_{t_0-52,\gamma})\,.$$

The forecasts of the quantities of interest are then provided, for $t \geqslant t_0 + 1$, by

$$\widehat{y}_{t+h,\gamma} = y_{t+h-52,\gamma} + \left(\ell_{t+h,\gamma} + h\, b_{t+h,\gamma}\right).$$

Remark. The choice $\beta = 0$ with the initialization $b_{t_0+1+h,\gamma} = 0$ (so that all b values are null) corresponds to simple exponential smoothing.

4.2.3 Tuning issue: aggregating rather than selecting forecasts / node by node

In this section, we describe the concept of robust aggregation of predictors at a given node γ. The next section (Section 4.2.4) will explain how to extend this concept to predictions at all nodes of the hierarchy considered.

Tuning issue. When only one set of forecasts (e.g., Holt's linear trend method with a multiplicative treatment of seasonality) is considered, it suffices to tune the two parameters α and β. This may typically be performed via cross-validation, on a training set. This may be performed locally (the parameters α_γ and β_γ picked depend on the node γ) or globally (the same parameters α and β are used at all nodes). However, in our case, several (sets of) elementary forecasts are available, which is more realistic. It may indeed be difficult to determine beforehand whether seasonality should be addressed in an additive or a multiplicative way. Also, the simple forecasts like the null sales may be particularly efficient for some subsubfamilies with rare sales. This is why we rather resort to aggregation of elementary forecasts coming from various models instead of selecting one particular forecasting method. This methodology was developed in the machine learning community in the 1990s and in the 2000s, see the monograph by Cesa-Bianchi and Lugosi [2006]. Its first application was to construct portfolios to invest in the stock market (Cover, 1991) and it has since then been successfully applied to a number of fields (see the end of Section 4.1.2 for a detailed list).

To further describe the concept of aggregation of forecasts we discuss first the evaluation of the forecasts issued.

Evaluating the quality of forecasts. We recall that sales y may be evaluated in units or in total value (we will pick the latter measure in our experiments). Two metrics are classically considered in logistics: the mean absolute error [MAE] and the root mean square error [RMSE].

Consider a sequence $y_{1+h,\gamma}, \dots, y_{T+h,\gamma}$ of sales that were to be predicted for a node γ, and assume that forecasts $\widehat{y}_{1+h,\gamma}, \dots, \widehat{y}_{T+h,\gamma}$ were issued. The MAE and the RMSE of these forecasts are respectively defined by

$$\text{MAE} = \frac{1}{T}\sum_{t=1}^{T}\Big|y_{t+h,\gamma} - \widehat{y}_{t+h,\gamma}\Big| \qquad \text{and} \qquad \text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\Big(y_{t+h,\gamma} - \widehat{y}_{t+h,\gamma}\Big)^2}\,.$$

4.2.3.1 Aggregation methods: principle and guarantees

Since several elementary forecasting methods (possibly tuned with different sets of parameters), say J methods, we index their forecasts by a superscript $j \in \{1, \dots, J\}$: they provide the forecasts $\widehat{y}^{(j)}_{t+h,\gamma}$. At each prediction step, these elementary forecasts are combined in a convex way: convex weights $w^{(1)}_{t+h,\gamma}, \dots, w^{(J)}_{t+h,\gamma}$ are picked, i.e., non-negative numbers summing up to 1, and the aggregated forecast

$$\widehat{f}_{t+h,\gamma} = \sum_{j=1}^{J} w^{(j)}_{t+h,\gamma}\, \widehat{y}^{(j)}_{t+h,\gamma}\,.$$

Specific algorithms for picking these convex weights are described in Section 4.2.3.2. Weights will be picked node by node.

The associated guarantees are typically of the following form: at each node, the aggregated forecasts are at least almost as good as the best individual elementary forecasting method, in MAE or in RMSE, while the aggregation algorithms do not know in advance which elementary forecasting method is the most efficient. In addition, no stochastic assumptions on the generating processes of the sales or of the elementary forecasts are required.

More precisely, we denote by $[0, Y_\gamma]$ the range for the sales and forecasts of sales for node γ. The MAE guarantees read: for all sequences of sales $y_{t+h,\gamma} \in [0, Y_\gamma]$ and all sequences of elementary forecasts $\widehat{y}^{(j)}_{t+h,\gamma} \in [0, Y_\gamma]$,

$$\frac{1}{T}\sum_{t=1}^{T}\Big|y_{t+h,\gamma} - \widehat{f}_{t+h,\gamma}\Big| \leqslant \varepsilon_{T,\gamma} + \min_{j=1,\dots,J}\frac{1}{T}\sum_{t=1}^{T}\Big|y_{t+h,\gamma} - \widehat{y}^{(j)}_{t+h,\gamma}\Big|\,, \qquad \text{where} \qquad \varepsilon_{T,\gamma} \to 0\,. \tag{4.1}$$

The bounds $\varepsilon_{T,\gamma}$ only depend on Y_γ and on T, they are uniform over the sequences considered. Similary, the RMSE guarantees read

$$\sqrt{\frac{1}{T}\sum_{t=1}^{T}\Big(y_{t+h,\gamma} - \widehat{f}_{t+h,\gamma}\Big)^2} \leqslant \varepsilon'_{T,\gamma} + \min_{j=1,\dots,J}\sqrt{\frac{1}{T}\sum_{t=1}^{T}\Big(y_{t+h,\gamma} - \widehat{y}^{(j)}_{t+h,\gamma}\Big)^2} \tag{4.2}$$

where the $\varepsilon'_{T,\gamma}$ only depend on Y_γ and on T and satisfy $\varepsilon'_{T,\gamma} \to 0$.

We now state the aggregation algorithms considered and hint at their associated guarantees, i.e., their associated values for the bounds $\varepsilon_{T,\gamma}$ or $\varepsilon'_{T,\gamma}$.

4.2.3.2 Aggregation methods: three examples

Three specific and popular aggregation algorithms are considered: first, the polynomially weighted average forecaster with multiple learning rates [ML-Poly] and the Prod forecaster with multiple learning rates [ML-Prod], both introduced by Gaillard et al. [2014]; second, the Bernstein Online Aggregation [BOA] of Wintenberger [2017]. Their statements in our context can be found in Algorithms 6, 8, and 10. The implementation of these algorithms depends on the guarantees (4.1) or (4.2) to be achieved. Indeed, as can be seen from their statements, they require a loss function: this should be the absolute loss $\ell(y,f)=|y-f|$ in case the MAE guarantee (4.1) is targeted, and the quadratic loss $\ell(y,f)=(y-f)^2$ for the RMSE guarantee (4.2). Given our specific context, several adaptations with respect to the original statements of these algorithms had to be performed, which are detailed below. We first provide some intuition on what the various quantities maintained in the statements of the algorithms stand for, and explain why we picked these algorithms.

Why these three algorithms? / What the various quantities maintained stand for. Both ML-Prod and BOA are variants of an alma matter aggregation algorithm called Hedge or the exponentially weighted average [EWA] predictor, and introduced by Vovk [1990] and Littlestone and Warmuth [1994]. It relies on a learning rate $\eta>0$ and picks weights (when adapted to our setting)

$$w_{t+h,\gamma}^{(j)}=\frac{\exp\left(-\eta\sum_{\tau=1}^{t}\ell\left(y_{\tau,\gamma},\widehat{y}_{\tau,\gamma}^{(j)}\right)\right)}{\sum_{k=1}^{J}\exp\left(-\eta\sum_{\tau=1}^{t}\ell\left(y_{\tau,\gamma},\widehat{y}_{\tau,\gamma}^{(k)}\right)\right)}=\frac{\exp\left(\eta\sum_{\tau=1}^{t}\left(\sum_{i=1}^{J}w_{t,\gamma}^{(i)}\,\ell\left(y_{t,\gamma},\widehat{y}_{t,\gamma}^{(i)}\right)-\ell\left(y_{\tau,\gamma},\widehat{y}_{\tau,\gamma}^{(j)}\right)\right)\right)}{\sum_{k=1}^{J}\exp\left(\eta\sum_{\tau=1}^{t}\left(\sum_{i=1}^{J}w_{t,\gamma}^{(i)}\,\ell\left(y_{t,\gamma},\widehat{y}_{t,\gamma}^{(i)}\right)-\ell\left(y_{\tau,\gamma},\widehat{y}_{\tau,\gamma}^{(k)}\right)\right)\right)}$$

ML-Prod is an adaptation of the second formulation of EWA on two main elements. First, the learning rate η depends on each elementary predictor k and is tuned over time: its value is given by $f(S_{t,\gamma}^{(k)},S_{t,\gamma}^{(k)})$. Second, the exponential reweighting through the $\exp(-\eta x)$ function is replaced by a multiplicative update by $1-\eta x$, which is a first-order approximation of the exponent. Similarly, BOA is an adaptation of the first formulation of EWA, where, in particular, prediction errors

$$\ell\left(y_{\tau,\gamma},\widehat{y}_{\tau,\gamma}^{(j)}\right)\qquad\text{are replaced by}\qquad\ell\left(y_{\tau,\gamma},\widehat{y}_{\tau,\gamma}^{(j)}\right)\left(1+\eta\ell\left(y_{\tau,\gamma},\widehat{y}_{\tau,\gamma}^{(j)}\right)\right),$$

which are only slightly larger quantities (as the learning rates are expected not to be too large). ML-Prod and BOA were both designed based on EWA and carefully adapted to get better theoretical guarantees and to not depend on any learning parameter (they are tuned automatically). They are also known for exhibiting better performance in general than EWA (see, e.g., discussions in the PhD thesis of Gaillard, 2015 and private feedback collected from the users of the Opera package by Gaillard and Goude, 2016).

As for ML-Poly, it is an adaptation of the polynomially weighted average [PWA] pre-

dictor (see Cesa-Bianchi and Lugosi, 2003), which uses weights based on a polynomial reweighting scheme of the form

$$w_{t+h,\gamma}^{(j)} = \frac{\max\left\{0,\ \sum_{\tau=1}^{t}\left(\sum_{i=1}^{J} w_{t,\gamma}^{(i)}\,\ell\Big(y_{t,\gamma},\widehat{y}_{t,\gamma}^{(i)}\Big) - \ell\Big(y_{\tau,\gamma},\widehat{y}_{\tau,\gamma}^{(j)}\Big)\right)\right\}^{p-1}}{\sum_{k=1}^{J}\max\left\{0,\ \sum_{\tau=1}^{t}\left(\sum_{i=1}^{J} w_{t,\gamma}^{(i)}\,\ell\Big(y_{t,\gamma},\widehat{y}_{t,\gamma}^{(i)}\Big) - \ell\Big(y_{\tau,\gamma},\widehat{y}_{\tau,\gamma}^{(k)}\Big)\right)\right\}^{p-1}},$$

for some $p \geqslant 2$. ML-Poly corresponds to $p = 2$ and will further reweight the nonnegative sums above (known as the cumulative regret of each elementary predictor) by quantities denoted by $B_{t,\gamma}^{(j)} + S_{t,\gamma}^{(j)}$ in Algorithm 6.

The three algorithms discussed above are implemented "from the book" except for the needed adaptations described below.

Adaptations needed. First, the range of the prediction errors (i.e., of the loss functions) was assumed to be known in the original references, while in our case, this range strongly depends on the numerous (subsub)families considered; there is no reason for knowing the orders of magnitude of the sales, thus of the prediction errors, for each (subsub)family. To cope for that, we maintain estimations $B_{t,\gamma}^{(j)}$ of the prediction errors (for BOA) or squared excess prediction errors (for ML-Poly and ML-Prod) and use these estimates in lieu of the known bounds of the original formulations of the algorithms.

Second, these algorithms were initially designed to forecast the next value of a time series, i.e., at time instance t, they issue forecasts of y_{t+1}. This corresponds, with our notation, to the case $h = n = 1$. For other cases, we performed the adaptations relative to (i) the information available at round t when forecasting sales (ii) at an horizon h. For (i), we note that the grouped sales $y_{\tau,\gamma}$ involve averages over n weeks, so that they are only defined for $\tau \geqslant n$; for rounds $\tau \leqslant n$, the algorithms get no input and pick uniform aggregations of the elementary forecasts. This is why time steps $t \in \{1, \dots, n-1\}$ are handled separately. For (ii), we use the value of the weights at round t to aggregate the elementary forecasts for the sales $y_{t+h,\gamma}$; this is in contrast with the original versions of the algorithms where such a combination is performed to forecast the next element, not the next h–th element of the time series.

Third, in the case of ML-Prod, the weight update

$$W_{t,\gamma}^{(j)} = \Big(W_{t-1,\gamma}^{(j)}\Big)^{f(B_{t,\gamma}^{(j)},S_{t,\gamma}^{(j)})/f(B_{t-1,\gamma}^{(j)},S_{t-1,\gamma}^{(j)})}\Big(1 + f\Big(B_{t,\gamma}^{(j)},S_{t,\gamma}^{(j)}\Big)\,e_{t,\gamma}^{(j)}\Big)$$

that may be read in Algorithm 8 slightly differs from the one that would have been obtained "from the book", namely,

$$W_{t,\gamma}^{(j)} = \Big(W_{t-1,\gamma}^{(j)}\Big(1 + f\Big(B_{t-1,\gamma}^{(j)},S_{t-1,\gamma}^{(j)}\Big)\,e_{t,\gamma}^{(j)}\Big)\Big)^{f(B_{t,\gamma}^{(j)},S_{t,\gamma}^{(j)})/f(B_{t-1,\gamma}^{(j)},S_{t-1,\gamma}^{(j)})};$$

the former is a first-order approximation of the latter, and ensures that weights are well-

defined: by definition of all quantities maintained in the algorithm,

$$\left| f\left(B_{t,\gamma}^{(j)}, S_{t,\gamma}^{(j)}\right) e_{t,\gamma}^{(j)} \right| \leqslant \frac{1}{2B_{t,\gamma}^{(j)}} e_{t,\gamma}^{(j)} \leqslant \frac{1}{2e_{t,\gamma}^{(j)}} e_{t,\gamma}^{(j)} \leqslant \frac{1}{2}\,,$$

while no specific guarantee holds on $f\left(B_{t-1,\gamma}^{(j)}, S_{t-1,\gamma}^{(j)}\right) e_{t,\gamma}^{(j)}$, which could be smaller than -1 if $e_{t,\gamma}^{(j)}$ is a large negative number.

Without these adaptations, the three algorithms ensure theoretical guarantees (4.1) and (4.2) of respective[2] orders $1/\sqrt{T}$ for $\varepsilon_{T,\gamma}$ and $T^{-1/4}$ for $\varepsilon'_{T,\gamma}$. Such guarantees should still hold under the two adaptations performed (estimated range and larger horizons $h \geqslant 2$).

4.2.3.3 Comparison to the best convex combination of elementary predictors (= the gradient trick)

The guarantees (4.1) and (4.2) can be strengthened, so that the performance of the aggregation algorithm is almost as good as that of the best constant convex combination of the elementary forecasts, i.e., the target

$$\min_{(q_1,\ldots,q_J)\in\mathcal{X}} \frac{1}{T}\sum_{t=1}^{T}\left| y_{t+h,\gamma} - \sum_{j=1}^{J} q_j\, \widehat{y}_{t+h,\gamma}^{(j)} \right| \leqslant \min_{j=1,\ldots,J} \frac{1}{T}\sum_{t=1}^{T}\left| y_{t+h,\gamma} - \widehat{y}_{t+h,\gamma}^{(j)} \right|$$

is considered for MAE (and a similar target for RMSE), where $\mathcal{X}$ denotes the set of all convex combinations, i.e., of all vectors $(q_1, \ldots, q_J)$ such that $q_j \geqslant 0$ for all j and $q_1 + \ldots + q_J = 1$. Put differently, uniform bounds of the form

$$\frac{1}{T}\sum_{t=1}^{T}\left| y_{t+h,\gamma} - \widehat{f}_{t+h,\gamma} \right| \leqslant \varepsilon_{T,\gamma} + \min_{(q_1,\ldots,q_J)\in\mathcal{X}} \frac{1}{T}\sum_{t=1}^{T}\left| y_{t+h,\gamma} - \sum_{j=1}^{J} q_j\, \widehat{y}_{t+h,\gamma}^{(j)} \right|, \quad \text{where} \quad \varepsilon_{T,\gamma} \to 0\,,$$

and

$$\sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(y_{t+h,\gamma} - \widehat{f}_{t+h,\gamma} \right)^2} \leqslant \varepsilon'_{T,\gamma} + \min_{(q_1,\ldots,q_J)\in\mathcal{X}} \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(y_{t+h,\gamma} - \sum_{j=1}^{J} q_j\, \widehat{y}_{t+h,\gamma}^{(j)} \right)^2},$$
$$\text{where} \qquad \varepsilon'_{T,\gamma} \to 0\,,$$

may be achieved, where the orders of magnitude of the $\varepsilon_{T,\gamma}$ and $\varepsilon'_{T,\gamma}$ are still $1/\sqrt{T}$ and $T^{-1/4}$.

To do so, the so-called "gradient trick" is applied (see, e.g., Cesa-Bianchi and Lugosi, 2006, Section 2.5 and references therein, in particular, Kivinen and Warmuth, 1997 and Cesa-Bianchi, 1999). It basically consists in replacing prediction errors by their gradients.

[2] The $T^{-1/4}$ rate for the RMSE is obtained through an initial bound on the mean square errors of the form

$$\frac{1}{T}\sum_{t=1}^{T}\left(y_{t+h,\gamma} - \widehat{f}_{t+h,\gamma} \right)^2 \leqslant \left(\varepsilon'_{T,\gamma}\right)^2 + \min_{j=1,\ldots,J} \frac{1}{T}\sum_{t=1}^{T}\left(y_{t+h,\gamma} - \widehat{y}_{t+h,\gamma}^{(j)} \right)^2$$

with $\left(\varepsilon'_{T,\gamma}\right)^2$ of the order of $1/\sqrt{T}$, combined with the inequality $\sqrt{a+b} \leqslant \sqrt{a} + \sqrt{b}$ for all non-negative numbers a, b.

Algorithm 6 Polynomially weighted average forecaster with multiple learning rates [ML-Poly], plain version

Parameters

Node γ, prediction horizon h, and week span n with $1 \leqslant n \leqslant h-1$

Loss function ℓ (absolute loss or quadratic loss)

Initialization

Set $R_{n-1,\gamma}^{(j)} = 0$, and $B_{n-1,\gamma}^{(j)} = 0$, and $S_{n-1,\gamma}^{(j)} = 0$ for all $j \in \{1, \dots, J\}$

for $t = 1, \dots, n-1$ **do**

Observe the elementary forecasts $\widehat{y}_{t+h,\gamma}^{(j)}$, where $j \in \{1, \dots, J\}$

Combine them uniformly, i.e., pick $w_{t+h,\gamma}^{(j)} = 1/J$ and form $\widehat{f}_{t+h,\gamma} = \dfrac{1}{J} \displaystyle\sum_{j=1}^{J} \widehat{y}_{t+h,\gamma}^{(j)}$

for $t = n,\ n+1,\ \dots$ **do**

Observe the sales $y_{t,\gamma}$

for $j \in \{1, \dots, J\}$ **do** // For each elementary predictor j

Set $e_{t,\gamma}^{(j)} = \left(\displaystyle\sum_{k=1}^{J} w_{t,\gamma}^{(k)}\, \ell\Big(y_{t,\gamma}, \widehat{y}_{t,\gamma}^{(k)}\Big) \right) - \ell\Big(y_{t,\gamma}, \widehat{y}_{t,\gamma}^{(j)}\Big)$ // Excess prediction error at t

Set $R_{t,\gamma}^{(j)} = R_{t-1,\gamma}^{(j)} + e_{t,\gamma}^{(j)}$ // Cumulated excess prediction error

Set $B_{t,\gamma}^{(j)} = \max\Big\{ B_{t-1,\gamma}^{(j)},\ \Big(e_{t,\gamma}^{(j)}\Big)^2 \Big\}$ // Bound on squared excess errors

Set $S_{t,\gamma}^{(j)} = S_{t-1,\gamma}^{(j)} + \Big(e_{t,\gamma}^{(j)}\Big)^2$ // Sum of squared excess errors

Observe the elementary forecast $\widehat{y}_{t+h,\gamma}^{(j)}$

Choose weights

$$w_{t+h,\gamma}^{(j)} = \frac{\max\Big\{0,\ R_{t,\gamma}^{(j)} / \Big(B_{t,\gamma}^{(j)} + S_{t,\gamma}^{(j)}\Big)\Big\}}{\displaystyle\sum_{k=1}^{J} \max\Big\{0,\ R_{t,\gamma}^{(k)} / \Big(B_{t,\gamma}^{(k)} + S_{t,\gamma}^{(k)}\Big)\Big\}}$$

Form the aggregated forecast $\widehat{f}_{t+h,\gamma} = \displaystyle\sum_{j=1}^{J} w_{t+h,\gamma}^{(j)}\, \widehat{y}_{t+h,\gamma}^{(j)}$

Algorithm 7 ML-Poly, version with the gradient trick

Same as above, except for the line defining $e_{t,\gamma}^{(j)}$, which should be replaced by

Set $e_{t,\gamma}^{(j)} = \psi\Big(\widehat{f}_{t,\gamma} - y_{t,\gamma}\Big)\ \Big(\widehat{f}_{t,\gamma} - \widehat{y}_{t,\gamma}^{(j)}\Big)$

where $\psi(x) = 2x$ for the quadratic loss ℓ and $\psi(x) = \operatorname{sign}(x)$ for the absolute loss ℓ

Algorithm 8 Prod forecaster with multiple learning rates [ML-Prod], plain version

Parameters

Node γ, prediction horizon h, and week span n with $1 \leqslant n \leqslant h-1$

Loss function ℓ (absolute loss or quadratic loss)

Notation

For $x, y > 0$, we define $f(x,y) = \min\left\{ \frac{1}{2x},\ \sqrt{\frac{\ln J}{x^2+y}} \right\}$

Initialization

Set $W_{n-1,\gamma}^{(j)} = 0$, and $B_{n-1,\gamma}^{(j)} = 0$, and $S_{n-1,\gamma}^{(j)} = 0$ for all $j \in \{1,\dots,J\}$

for $t = 1, \dots, n-1$ **do**

Observe the elementary forecasts $\widehat{y}_{t+h,\gamma}^{(j)}$, where $j \in \{1,\dots,J\}$

Combine them uniformly, i.e., pick $w_{t+h,\gamma}^{(j)} = 1/J$ and form $\widehat{f}_{t+h,\gamma} = \frac{1}{J}\sum_{j=1}^{J} \widehat{y}_{t+h,\gamma}^{(j)}$

for $t = n,\ n+1,\ \dots$ **do**

Observe the sales $y_{t,\gamma}$

for $j \in \{1,\dots,J\}$ **do** // For each elementary predictor j

Set $e_{t,\gamma}^{(j)} = \left(\sum_{k=1}^{J} w_{t,\gamma}^{(k)}\, \ell\Big(y_{t,\gamma}, \widehat{y}_{t,\gamma}^{(k)}\Big) \right) - \ell\Big(y_{t,\gamma}, \widehat{y}_{t,\gamma}^{(j)}\Big)$ // Excess prediction error at t

Set $B_{t,\gamma}^{(j)} = \max\left\{ B_{t-1,\gamma}^{(j)},\ \left| e_{t,\gamma}^{(j)} \right| \right\}$ // Bound on excess errors

Set $S_{t,\gamma}^{(j)} = S_{t-1,\gamma}^{(j)} + \left(e_{t,\gamma}^{(j)} \right)^2$ // Cumulated excess prediction error

Set $W_{t,\gamma}^{(j)} = \left(W_{t-1,\gamma}^{(j)} \right)^{f(B_{t,\gamma}^{(j)}, S_{t,\gamma}^{(j)}) / f(B_{t-1,\gamma}^{(j)}, S_{t-1,\gamma}^{(j)})} \left(1 + f\Big(B_{t,\gamma}^{(j)}, S_{t,\gamma}^{(j)}\Big)\, e_{t,\gamma}^{(j)} \right)$

// Multiplicative update of the weight maintained for predictor j

Observe the elementary forecast $\widehat{y}_{t+h,\gamma}^{(j)}$

Choose weights

$$w_{t+h,\gamma}^{(j)} = \frac{f(B_{t,\gamma}^{(j)}, S_{t,\gamma}^{(j)})\, W_{t,\gamma}^{(j)}}{\sum_{k=1}^{J} f(S_{t,\gamma}^{(k)}, S_{t,\gamma}^{(k)})\, W_{t,\gamma}^{(k)}}$$

Form the aggregated forecast $\widehat{f}_{t+h,\gamma} = \sum_{j=1}^{J} w_{t+h,\gamma}^{(j)}\, \widehat{y}_{t+h,\gamma}^{(j)}$

Algorithm 9 ML-Prod, version with the gradient trick

Same as above, except for the line defining $e_{t,\gamma}^{(j)}$, which should be replaced by

Set $e_{t,\gamma}^{(j)} = \psi\Big(\widehat{f}_{t,\gamma} - y_{t,\gamma}\Big)\Big(\widehat{f}_{t,\gamma} - \widehat{y}_{t,\gamma}^{(j)}\Big)$

where $\psi(x) = 2x$ for the quadratic loss ℓ and $\psi(x) = \mathrm{sign}(x)$ for the absolute loss ℓ

Algorithm 10 Bernstein Online Aggregation [BOA], plain version

Parameters

Node γ, prediction horizon h, and week span n with $1 \leqslant n \leqslant h-1$

Loss function ℓ (absolute loss or quadratic loss)

Notation

For $x, y > 0$, we define $f(x,y) = \min\left\{\dfrac{1}{2x},\ \sqrt{\dfrac{\ln J}{y}}\right\}$

Initialization

Set $L^{(j)}_{n-1,\gamma} = 0$, and $B^{(j)}_{n-1,\gamma} = 0$, and $S^{(j)}_{n-1,\gamma} = 0$, and $\eta^{(j)}_{n-1,\gamma} = 0$ for all $j \in \{1,\dots,J\}$

for $t = 1, \dots, n-1$ **do**

Observe the elementary forecasts $\widehat{y}^{(j)}_{t+h,\gamma}$, where $j \in \{1,\dots,J\}$

Combine them uniformly, i.e., pick $w^{(j)}_{t+h,\gamma} = 1/J$ and form $\widehat{f}_{t+h,\gamma} = \dfrac{1}{J}\displaystyle\sum_{j=1}^{J} \widehat{y}^{(j)}_{t+h,\gamma}$

for $t = n,\ n+1,\ \dots$ **do**

Observe the sales $y_{t,\gamma}$

for $j \in \{1,\dots,J\}$ **do** // For each elementary predictor j

Set $e^{(j)}_{t,\gamma} = \ell\Big(y_{t,\gamma}, \widehat{y}^{(j)}_{t,\gamma}\Big)$ // Prediction error at t

Set $L^{(j)}_{t,\gamma} = L^{(j)}_{t-1,\gamma} + e^{(j)}_{t,\gamma}\left(1 + \eta^{(j)}_{t-1,\gamma} e^{(j)}_{t,\gamma}\right)$ // Cumulated (slightly enlarged) prediction errors

Set $B^{(j)}_{t,\gamma} = \max\Big\{B^{(j)}_{t-1,\gamma},\ e^{(j)}_{t,\gamma}\Big\}$ // Bound on prediction errors

Set $S^{(j)}_{t,\gamma} = S^{(j)}_{t-1,\gamma} + \Big(e^{(j)}_{t,\gamma}\Big)^2$ // Cumulative squared prediction errors

Set $\eta^{(j)}_{t,\gamma} = f\Big(B^{(j)}_{t,\gamma},\ S^{(j)}_{t,\gamma}\Big)$ // Weighting factor

Observe the elementary forecast $\widehat{y}^{(j)}_{t+h,\gamma}$

Choose weights

$$w^{(j)}_{t+h,\gamma} = \frac{\eta^{(j)}_{t,\gamma}\ \exp\Big(-\eta^{(j)}_{t,\gamma} L^{(j)}_{t,\gamma}\Big)}{\displaystyle\sum_{k=1}^{J} \eta^{(k)}_{t,\gamma}\ \exp\Big(-\eta^{(k)}_{t,\gamma} L^{(k)}_{t,\gamma}\Big)}$$

Form the aggregated forecast $\widehat{f}_{t+h,\gamma} = \displaystyle\sum_{j=1}^{J} w^{(j)}_{t+h,\gamma}\, \widehat{y}^{(j)}_{t+h,\gamma}$

Algorithm 11 BOA, version with the gradient trick

Same as above, except for the line defining $e^{(j)}_{t,\gamma}$, which should be replaced by

Set $e^{(j)}_{t,\gamma} = \psi\Big(\widehat{f}_{t,\gamma} - y_{t,\gamma}\Big)\, \widehat{y}^{(j)}_{t,\gamma}$

where $\psi(x) = 2x$ for the quadratic loss ℓ and $\psi(x) = \operatorname{sign}(x)$ for the absolute loss ℓ

More precisely, the three algorithms stated above are modified as follows. In each statement, only the line defining $e_{t,\gamma}^{(j)}$ based on the losses $\ell\Big(y_{t,\gamma}, \widehat{y}_{t,\gamma}^{(k)}\Big)$ needs to be changed. These losses are replaced by $\psi\Big(\widehat{f}_{t,\gamma} - y_{t,\gamma}\Big)\, \widehat{y}_{t,\gamma}^{(j)}$, where $\psi : \mathbb{R} \to \mathbb{R}$ is defined as follows. For the quadratic loss $\ell(y, f) = (y - f)^2$, we define $\psi(x) = 2x$. For the absolute loss $\ell(y, f) = |y - f|$, we define $\psi(x) = \mathrm{sign}(x)$, the sign of x, that is,

$$\mathrm{sign}(x) = \begin{cases} +1 \text{ if } x > 0 \\ \ \ 0 \text{ if } x = 0 \\ -1 \text{ if } x < 0 \end{cases}$$

For the sake of clarity, the modified algorithms are stated below the original algorithms; see Algorithms 7, 9 and 11.

4.2.4 Providing aggregated forecasts for the entire hierarchy

So far, we discussed the node-by-node prediction of sales, independently for each (sub-sub)family, thus discarding for the time being the summation constraints indicated in Section 4.2.1. We now focus our attention on reconciling these independent predictions.

Overall performance discarding the summation constraints. To that end, we first define the MAE and the RMSE of a family of sequences of forecasts over time (similarly to what we did in Section 4.2.3 for a single sequence of forecasts over time). Consider a family of sequences $y_{1+h,\gamma}, \ldots, y_{T+h,\gamma}$ of sales that were to be predicted for a hierarchy of nodes $\gamma \in \Gamma$, and assume that families of sequences of forecasts $\widehat{y}_{1+h,\gamma}, \ldots, \widehat{y}_{T+h,\gamma}$, $\gamma \in \Gamma$, were issued. The MAE and the RMSE of these families of sequences of forecasts are respectively defined by

$$\text{MAE} = \frac{1}{T|\Gamma|} \sum_{t=1}^{T} \sum_{\gamma \in \Gamma} \Big| y_{t+h,\gamma} - \widehat{y}_{t+h,\gamma} \Big| \qquad \text{and} \qquad \text{RMSE} = \sqrt{\frac{1}{T|\Gamma|} \sum_{t=1}^{T} \sum_{\gamma \in \Gamma} \Big(y_{t+h,\gamma} - \widehat{y}_{t+h,\gamma} \Big)^2},$$

where $|\Gamma|$ denotes the cardinality of Γ.

When the guarantees (4.1) and (4.2) hold for all $\gamma \in \Gamma$, the overall performance achieved is almost as good as that of the best local elementary forecasting methods; that is, by summing prediction errors along the hierarchy Γ, the following is guaranteed: uniformly over sequences of sales and of elementary forecasts,

$$\frac{1}{T|\Gamma|} \sum_{t=1}^{T} \sum_{\gamma \in \Gamma} \Big| y_{t+h,\gamma} - \widehat{f}_{t+h,\gamma} \Big| \leqslant \varepsilon_T + \frac{1}{|\Gamma|} \sum_{\gamma \in \Gamma} \min_{j=1,\ldots,J} \frac{1}{T} \sum_{t=1}^{T} \Big| y_{t+h,\gamma} - \widehat{y}_{t+h,\gamma}^{(j)} \Big|, \quad \text{where} \quad \varepsilon_T \to 0\,, \tag{4.3}$$

and[3]

$$\sqrt{\frac{1}{T|\Gamma|}\sum_{t=1}^{T}\sum_{\gamma\in\Gamma}\Big(y_{t+h,\gamma}-\widehat{f}_{t+h,\gamma}\Big)^2}\leqslant\varepsilon'_T+\sqrt{\frac{1}{|\Gamma|}\sum_{\gamma\in\Gamma}\min_{j=1,\dots,J}\frac{1}{T}\sum_{t=1}^{T}\Big(y_{t+h,\gamma}-\widehat{y}^{(j)}_{t+h,\gamma}\Big)^2},$$

$$\text{where}\qquad \varepsilon'_T\to 0\,. \quad (4.4)$$

The performance achieved by the best local elementary forecasting methods (the performance reported in the right-hand sides above) will be called the oracle performance in the sequel.

Projections to abide by the summation constraints. Now, there is no reason for the aggregated forecasts $\widehat{f}_{t+h,\gamma}$ picked node by node as discussed in Section 4.2.3 to abide by the summation constraints indicated in Section 4.2.1. This situation is similar to the one where a given elementary forecasting method (e.g., Holt's linear trend method with a multiplicative treatment of seasonality) is tuned node by node (e.g., by independent cross-validations), for the sake of efficiency: possibly different parameters $\widehat{\alpha}_\gamma$, $\widehat{\beta}_\gamma$ are picked for each node γ and the elementary forecasts issued do not abide by the summation constraints, in general.

A simple patch is however to project a vector of forecasts not abiding by the summation constraints onto the vector space $\mathcal{H}$ of those abiding by them; formally, we define $\mathcal{H}$ as the vector space of vectors $(f_\gamma)_{\gamma\in\Gamma}$ such that for all nodes $\gamma\in\Gamma$ with non-empty set $\mathcal{C}(\gamma)$ of children nodes,

$$f_\gamma=\sum_{c\in\mathcal{C}(\gamma)}f_c\,.$$

The projection may take place in Euclidean norm or in absolute norm. Let us denote by $\big(\widetilde{f}_{t+h,\gamma}\big)_{\gamma\in\Gamma}$ the projection of $\big(\widehat{f}_{t+h,\gamma}\big)_{\gamma\in\Gamma}$ onto $\mathcal{H}$ in some norm and let us review the theoretical guarantees, or lack thereof, associated with each norm.

Euclidean norm: theoretical guarantees. The theoretical guarantee that follows is already mentioned in Chapter 3. When the projection is in Euclidean norm, the Pythagorean theorem ensures that

$$\sqrt{\frac{1}{T|\Gamma|}\sum_{t=1}^{T}\sum_{\gamma\in\Gamma}\Big(y_{t+h,\gamma}-\widetilde{f}_{t+h,\gamma}\Big)^2}\leqslant\sqrt{\frac{1}{T|\Gamma|}\sum_{t=1}^{T}\sum_{\gamma\in\Gamma}\Big(y_{t+h,\gamma}-\widehat{f}_{t+h,\gamma}\Big)^2}\,.$$

Thus, whenever the guarantees (4.3) and (4.4) are satisfied for aggregated forecasts, they are also satisfied for their Euclidean projections. The latter may only improve performance and ensure that the summation constraints are satisfied, i.e., the forecasts issued are consistent with the hierarchy considered.

We implement the Euclidean projection $\Pi_{\mathcal{H}}$ as follows. We introduce the set $\mathcal{L}(\Gamma)$ of

[3]The bound in RMSE is obtained by first summing the initial bounds described in Footnote 2 and then taking square roots.

leaves of Γ and a matrix S indexed by $\Gamma \times \mathcal{L}(\Gamma)$, where for all $\gamma \in \Gamma$ and $\gamma' \in \mathcal{L}(\Gamma)$,

$$S_{\gamma,\gamma'} = \begin{cases} 1 \text{ if } \gamma = \gamma', \\ 1 \text{ if } \gamma \text{ is the parent node of } \gamma', \\ 0 \text{ otherwise.} \end{cases}$$

The image of S is exactly $\mathcal{H}$. Since S is injective and its image is $\mathcal{H}$, it may be shown that the Euclidean projection onto $\mathcal{H}$ is given by the matrix

$$\Pi_{\mathcal{H}} = S(S^{\mathrm{T}}S)^{-1}S^{\mathrm{T}}\,.$$

Absolute norm: no theoretical guarantee. The projection $\left(\widetilde{f}_{t+h,\gamma}\right)_{\gamma\in\Gamma}$ of $\left(\widehat{f}_{t+h,\gamma}\right)_{\gamma\in\Gamma}$ onto $\mathcal{H}$ in absolute norm is defined as:

$$\left(\widetilde{f}_{t+h,\gamma}\right)_{\gamma\in\Gamma} \in \underset{(z_\gamma)_{\gamma\in\Gamma}\in\mathcal{H}}{\arg\min} \frac{1}{T|\Gamma|}\sum_{t=1}^{T}\sum_{\gamma\in\Gamma}\left|z_\gamma - \widehat{f}_{t+h,\gamma}\right|$$

There are no theoretical guarantees on the performance of the projected forecasts, as no Pythagorean-type theorem is able to relate

$$\frac{1}{T|\Gamma|}\sum_{t=1}^{T}\sum_{\gamma\in\Gamma}\left|y_{t+h,\gamma} - \widetilde{f}_{t+h,\gamma}\right| \qquad \text{to} \qquad \frac{1}{T|\Gamma|}\sum_{t=1}^{T}\sum_{\gamma\in\Gamma}\left|y_{t+h,\gamma} - \widehat{f}_{t+h,\gamma}\right|.$$

Even worse, numerical results discussed in Section 4.3.3 show that the projection in absolute norm may even increase the prediction error.

4.3 Numerical results

We now apply the forecasting methodology described in the previous section to our data set and more particularly, we consider the three algorithms described therein (ML-Poly, ML-Prod and BOA) under the various implementations possible: with a loss function given by the absolute loss or the quadratic loss, with or without the gradient trick, with or without a Euclidean or absolute-norm projection step after all local forecasts were issued (to meet the hierarchical constraints).

We compare the various implementations of these algorithms at different time horizons (h, n). We recall that n denotes the number of weeks of sales considered in the forecasts and h the forecasting horizon, i.e., after the week considered, there are $h-1$ weeks, and then starts the group of n weeks to forecast; the first week of this group is in h weeks. Put differently, the group of weeks to forecast is $h-1$–week-ahead.

Outline of the empirical study. We first provide a description of the real data set provided by the company Cdiscount and how we divided it into a train set and a test set (Section 4.3.1). We then tabulate and graphically illustrate the performance of the elementary predictors considered (Section 4.3.2) depending on the cases (h, n) considered; the case $(h, n) = (7, 1)$ is a challenging case, which is also representative of a typical case from a business viewpoint.

We may then compare the three algorithms and their various implementations on the case where $(h, n) = (7, 1)$. We illustrate that the performance varies only slightly with the algorithm picked and its specific implementation (loss function, gradient trick, projection) and improves the locally best elementary predictors picked on the train set, the natural benchmark, by a about 5%. This observation generalizes to all pairs (h, n) considered (Section 4.3.3). So far, performance is studied only in terms of MAE or RMSE. We then move (Section 4.3.4) to an evaluation in terms of mean absolute percentage of error [MAPE], to get a better grasp of the forecasting performance (Section 4.3.4). Again, aggregation methods improve by about 5% the performance of natural benchmarks like the locally best elementary predictors picked on the train set, achieving a MAPE of about 20%. This global MAPE is then broken down by the levels of the hierarchy, and as expected, is larger for subsubfamilies (about 32%) than for subfamilies and families (about 22% and 18%) or for the total node (only about 12%).

Two complementary studies are finally provided. As all results previously discussed were on average only, we check that the better average performance obtained was so through a shift of the distributions of errors towards zero (Section 4.3.5). We also give an idea of how the weights put on each elementary predictor evolve, on families: they are far from converging to anything and they show that the aggregation methods are reactive to changes (Section 4.3.6).

4.3.1 Description of the data set

Our data set is a real data set provided by the e-commerce company Cdiscount. Our data spans from July 2014 to December 2017—a period of 182 weeks. It features the daily sales of 620,749 products gathered in 3,004 subsubfamilies, 570 subfamilies and 53 families; that is, the cardinality of the hierarchy Γ is $3{,}004 + 570 + 53 + 1 = 3{,}628$ nodes, including the leaves (subsubfamilies) and the root node (the total sales). We added up daily sales to get weekly sales.

Figure 1 depicts some series of weekly sales: the total sales (top left picture) and series associated with two families, two subfamilies and one subsubfamily. These series all exhibit some seasonality, but with different cycles. Some have a linear trend. Some are highly regular, some others exhibit a more erratic behavior. It is clear that no single elementary predictor of Section 4.2.2 can be simultaneously suited for all series.

Table 1 provides some descriptive statistics (minimum and maximum, median and means) on the weekly sales, by levels of the hierarchy of products. This table also shows that many weekly-sales data points are null: 45.3% of the $3{,}004 \times 182$ weekly sales for subsubfamilies, 48.3% of the 570×182 weekly sales for subfamilies, and even 38.1% of the 53×182 weekly sales for families. Part of these null values corresponds to intermittent demand, but it turns out that some nodes of the hierarchy encounter null sales during the entire period considered. More precisely, for 133 (out of $3{,}004$) subsubfamilies, 37 (out of 570) subfamilies, and 6 (out of 53) families, there are absolutely no sales during the 182 weeks considered. These high sparsity rates observed (on this data set and on other similar data sets of e-commerce data) explain why the null elementary predictor defined in Section 4.2.2 was considered.

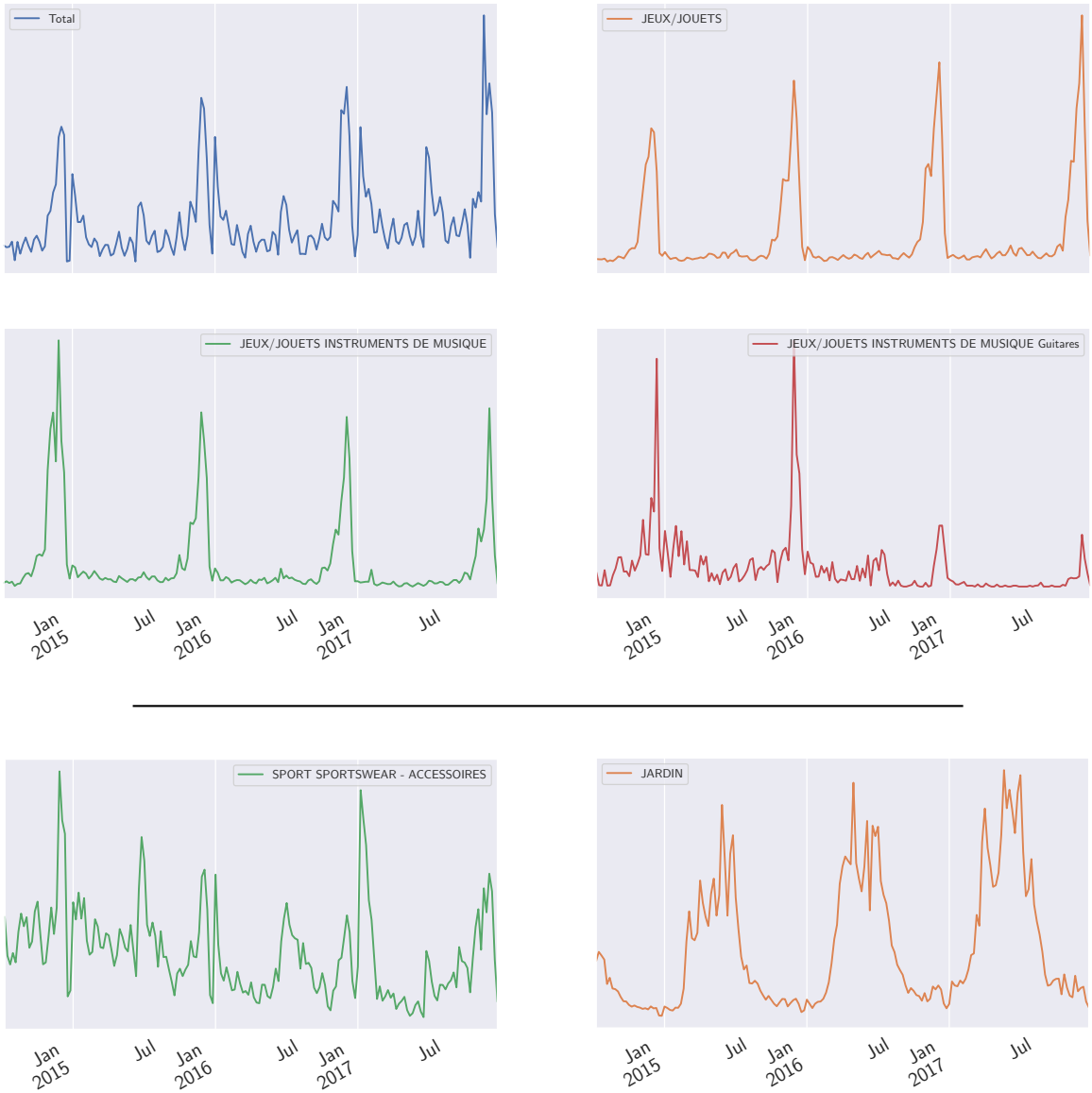


Figure 1 – Weekly sales (*y–values*) over time (*x–axes*) at some nodes of the hierarchy. The scales of the y–axes vary by graphs and are hidden for confidentiality reasons.

The *top four graphs* correspond to a given path in the hierarchy, corresponding to the subsubfamily of kids' guitars (red plot), which is a part of the subfamily of kids' music instruments (green plot), which itself belong to the family of toys (orange plot). The evolution of the total sales at the root node is also provided (blue plot). This path reads in French (see the legends on each graph): Total > Jeux/Jouets > Instruments de musique > Guitares.

The *bottom two graphs* feature the sales for the subsubfamily (green plot, legend "Sport - Sportswear Accessoires") of sportswear accessories and the family of garden products (orange plot, legend "Jardin"), respectively.

Train set, test set

We recall that our data spans from July 2014 to December 2017 (and features 182 weeks in total). We use July 2014 to December 2016 as a training period (containing 130 weeks), and January 2017 – December 2017 (containing 52 weeks) as a test period. The test period thus features all major commercial events (sales, Black Friday and Christmas shopping, etc.). More precisely, after week $52 + 26 + n \leqslant 80$ (given the values $n \leqslant 4$ considered below), all elementary forecasting methods of Section 4.2.2 provide predictions and are aggregated via the algorithms described in Section 4.2.3, for the remaining part of the train period and also during the test period. The performance obtained is however computed only on the test period, in MAE or RMSE, as explained at the beginning of Section 4.2.3.

Table 1 – Some descriptive statistics on the weekly sales (units: thousands of euros [k€]) for the 182 weeks considered, by hierarchy level: subsubfamilies, subfamilies, families, and the root node ("Total"). The numbers of nodes ("Count") available for each level are recalled in the top part of the table. Classical descriptive statistics (minimum and maximum, mean and median) are provided in the middle part of the table. Specific descriptive statistics pertaining to the sparsity of the sequences of weekly sales are given in the bottom part of the table: the fraction of data points that are null ("Global sparsity rate") among all data points of this level, and counts of entire sequences of weekly sales that are null ("Null series: count") among the sequences of this level (there are "Count" of them).

	Subsubfamilies	Subfamilies	Families	Total
Count	3,004	570	53	1
Maximum	5.6×10^6	9.4×10^6	17.5×10^6	88.6×10^6
Mean	10.1×10^3	53.4×10^3	574.6×10^3	30.5×10^6
Median	18.4	7.1	246.8	26.0×10^6
Minimum	0	0	0	18.6×10^3
Global sparsity rate	45.3%	48.3%	38.1%	0%
Null series: count	133	37	6	0

4.3.2 Performance of the elementary forecasting methods

We introduced three groups of elementary predictors in Section 4.2.2. The first group features the null predictor, the predictor picking the sales achieved exactly one year ago, and the predictor picking the current value of sales. The second group features simple exponential smoothing (which relies on a tuning parameter $\alpha \in [0, 1]$), with an additive or a multiplicative treatment of seasonality, while the third group is made of Holt's linear trend method (which relies on two tuning parameters $\alpha, \beta \in [0, 1]$), again with an additive or a multiplicative treatment of seasonality. We pick a finite number of possible values for α and β for our numerical experiments, namely:

$$\alpha \in \left\{2^{-6}, 2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 1/2, 1\right\} \qquad \text{and} \beta \in \left\{2^{-4}, 2^{-3}, 2^{-2}, 1/2\right\}$$

(as the case $\beta = 1$ essentially corresponds to simple exponential smoothing). As illustrated by Figure 2, this leads to 73 elementary predictors: 3 predictors in the first group, 2×7

predictors based on simple exponential smoothing, and $2 \times (7 \times 4)$ predictors based on Holt's linear trend method.

Definition of three meta-predictors. Based on these elementary predictors, we define three meta-predictors: one legal meta-predictor and two forward-looking ones (they "cheat" and use future data to pick among the elementary predictors).

The legal meta-predictor is to use at each node of the hierarchy on the test set the elementary predictor that obtained the best performance on the train set. We call this meta-predictor the locally best elementary predictors on the train set; this is maybe the most natural meta-predictor in the eyes of practitioners.

A first forward-looking meta-predictor called the oracle prediction was already defined in Section 4.2.4: it picks the locally best elementary predictors on the test set, that is, with the notation of Section 4.2.4, it achieves a performance in terms of

$$\frac{1}{|\Gamma|}\sum_{\gamma\in\Gamma}\min_{j=1,\dots,J}\frac{1}{T}\sum_{t=1}^{T}\ell\left(y_{t+h,\gamma},\widehat{y}^{(j)}_{t+h,\gamma}\right), \tag{4.5}$$

where ℓ is the loss function (absolute loss or squared loss) at hand.

Finally, we define a second forward-looking meta-predictor given by the globally best elementary predictor on the test set, that is, the elementary predictor that obtains the best performance on the test set when used on all nodes of the hierarchy; it achieves a performance in terms of

$$\min_{j=1,\dots,J}\frac{1}{|\Gamma|}\sum_{\gamma\in\Gamma}\frac{1}{T}\sum_{t=1}^{T}\ell\left(y_{t+h,\gamma},\widehat{y}^{(j)}_{t+h,\gamma}\right), \tag{4.6}$$

The notion of "best" depends on the underlying metric: MAE or RMSE. The globally best elementary predictor on the test set may differ for each metric; the same can be said for locally best elementary predictors on the train or test set.

Figure 2: Graphical comparison of these elementary predictors and meta-predictors. Figure 2 reports the performance of the elementary predictors and meta-predictors recalled or defined above, in MAE and RMSE, for the case $(h, n) = (7, 1)$, that is, for 6–week-ahead forecasts relative to 1 week of sales. The four most interesting performance to read therein are, in order: the predictor picking the sales achieved exactly one year ago (worst performance), the locally best elementary predictors on the train set, the globally best elementary predictor on the test set, and the oracle (i.e., the locally best elementary predictors on the test set; best performance). The best two such [meta-]predictors are forward-looking ones. The gap between the locally best elementary predictors on the train set (the best legal meta-predictor) and the globally best elementary predictor on the test set (a forward-looking meta-predictor) is much larger in the case of RMSE than for MAE; it is almost null in the case of MAE.

As we show in the next sections, the performance of the aggregation algorithms considered will be close to (but usually slightly larger than) the one of the globally best elementary predictor on the test set, and in any case, significantly better than the one of the locally best elementary predictors on the train set.

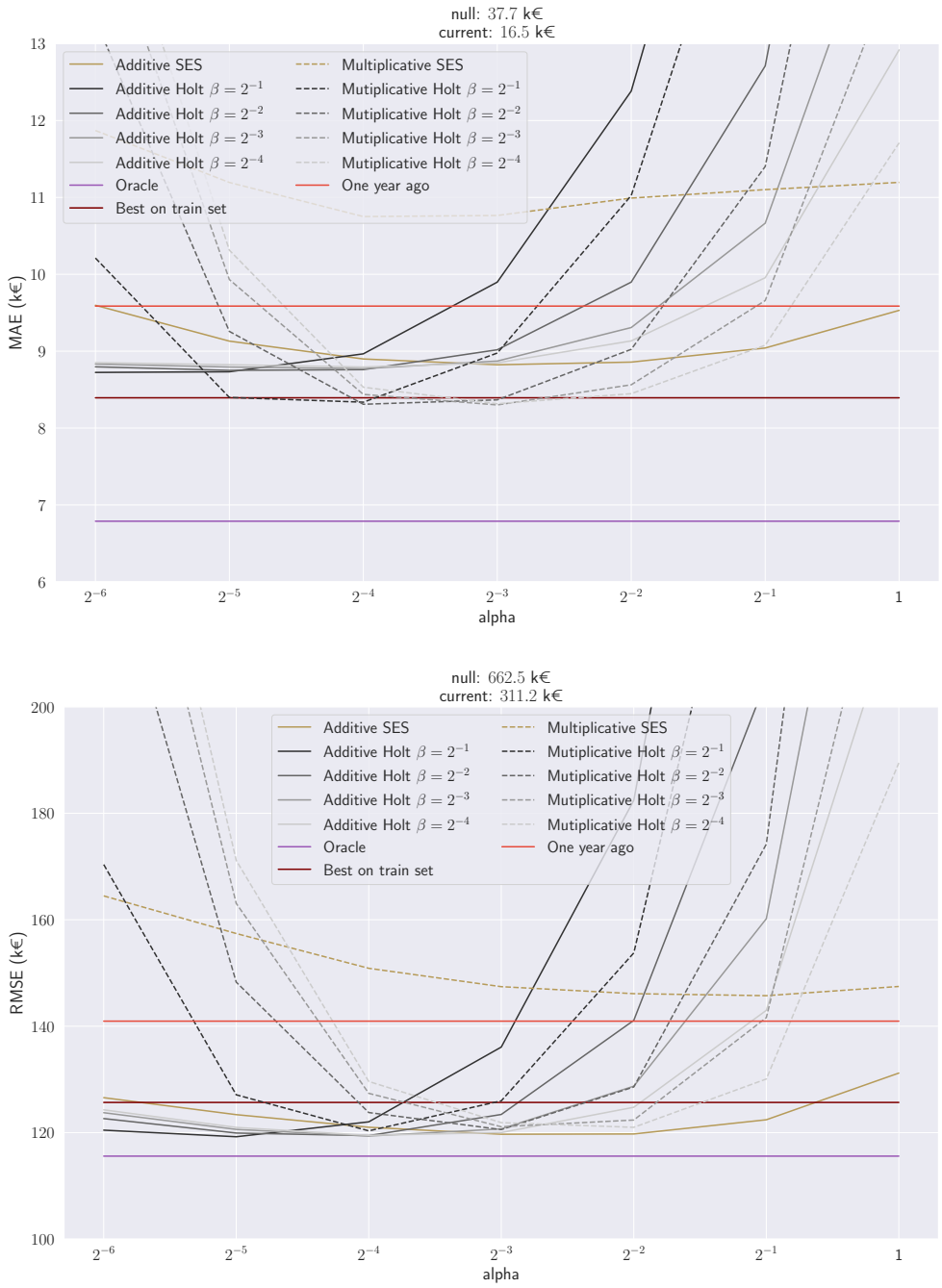


Figure 2 – Performance [*y–axis*, nominal scale] of the elementary forecasting methods to forecast sales 6-week-ahead for 1 week (i.e., for $h = 7$ and $n = 1$), in MAE [*top figure*] and RMSE [*bottom figure*], depending on a tuning parameter α [*x–axis*, logarithmic scale]. The acronym `SES` stands for simple exponential smoothing, which can be implemented in a `Multiplicative` or an `Additive` treatment of seasonality. `Holt`'s linear trend method can also be implemented with a `Multiplicative` or an `Additive` treatment of seasonality and depends on a second parameter β: this is why `Holt` elementary forecasting methods are instantiated for several values of β.

The `Null`, `Current` and `One year ago` elementary forecasting methods are the first three ones described in Section 4.2.2 and do not depend on α. The same can be said for the `Oracle` performance described in Equation (4.4) as well as for the locally `Best on train set` predictor introduced in the beginning of Section 4.3.2. The performance of `One year ago`, `Best on train set`, and `Oracle` are therefore depicted by horizontal lines, while the one of `Null` and `Current` can be found above the legend.

Table 2: Numerical comparison of these elementary predictors and meta-predictors. The considerations above and the graphical comparison offered by Figure 2 for the case $(h, n) = (7, 1)$ show that our main indicators are given by the performance of three meta-predictors: the locally best elementary predictors on the train set (the legal meta-predictor), the globally best elementary predictor on the test set (the first forward-looking meta-predictor), and the oracle (i.e., the locally best elementary predictors on the test set; the second forward-looking meta-predictor). Table 2 reports these indicators in MAE and RMSE, for various pairs (h, n) of forecasting horizon $h - n$ and number n of weeks to be forecast.

The main lessons are first that as expected, the farther away the horizon $h - n$, the more important the average errors (in MAE or RMSE), and the larger the number n of weeks to be forecast, the smaller the average errors (a law-of-large-number probably smoothes out sales when they are averaged over $n \geqslant 2$ weeks). The number n of weeks to forecast seems to have a greater impact on the average errors than the horizon $h - n$, both for MAEs and RMSEs (actually, the RMSEs seem to be almost independent of the horizon $h - n$).

Second, with one exception out of 18 cases of metric and (h, n) pair considered, the meta-predictors are consistently ranked, in terms of average errors, as discussed above: the worst performance is achieved by the legal meta-predictor (the locally best elementary predictors on the train set) and the best performance is obtained by the oracle (the locally best elementary predictors on the test set), with the other forward-looking meta-predictor (the globally best elementary predictor on the test set) lying between them. The exception corresponds to the case of MAE and $(h, n) = (5, 1)$. This ranking may look surprising: the globally best elementary predictor on the test set picks the same predictor at each node of the hierarchy and is less flexible than the legal meta-predictor, that pick independently the elementary predictors at each node (based on their performance on the train set). This probably means that the train set is much different from the test set, which is probably due to highly non-stationary nature of e-commerce data. This is why more flexible methods are welcome, like the online aggregation methods used in this chapter.

Discussion on the two metrics considered: MAE and RMSE. We add a final note on the orders of magnitude between MAEs and RMSEs. They differ by a factor of 10 to 15, with the RMSEs being roughly 10 to 15 times larger than the MAEs. This is because RMSEs are extremely sensitive to extreme values. These extreme values may correspond, in e-commerce, to external interferences (sales periods, disruptions in supply of some products, launches of new products, crises: financial, sanitary, social crises). The impact of such external interferences needs to be forecast separately, with ad hoc models and methods. The scope of the present chapter is therefore rather on forecasting sales in stationary regimes, that is, for "ordinary" or routine circumstances. And in such regimes and circumstances, extreme values are rare and less important in the eyes of the decision-makers than typical values. This is why, in the sequel, while reporting both MAEs and RMSEs, we will be more interested in the performance in MAE.

Table 2 – Average errors (MAEs or RMSEs) in k€ for three meta-predictors (columns 5, 6, 7) depending on the metric considered (column 1) as well as the horizon and number of weeks to be forecast (columns 2, 3, 4, where we recall that the horizon is given by $h-n$) The three meta-predictors considered were introduced in the beginning of Section 4.3.2: the locally best elementary predictors on the train set ("Locally best on train set", column 5), the globally best elementary predictor on the test set ("Globally best on test set", column 6), and the oracle (which corresponds to the locally best elementary predictors on the test set, abbreviated as "Locally best on test set", column 7). .

Metric in k€	Horizon	Group	Pair (h, n)	Locally best on train set	Globally best on test set	Locally best on test set (= Oracle)
MAE	6-week-ahead	for 1 week	$(7, 1)$	8.39	8.30	6.79
MAE	6-week-ahead	for 2 weeks	$(8, 2)$	7.39	7.34	5.70
MAE	6-week-ahead	for 4 weeks	$(10, 4)$	6.80	6.59	4.85
RMSE	6-week-ahead	for 1 week	$(7, 1)$	125.68	119.24	115.59
RMSE	6-week-ahead	for 2 weeks	$(8, 2)$	97.26	90.94	85.78
RMSE	6-week-ahead	for 4 weeks	$(10, 4)$	82.36	73.92	68.23
MAE	4-week-ahead	for 1 week	$(5, 1)$	8.18	8.20	6.78
MAE	4-week-ahead	for 2 weeks	$(6, 2)$	7.27	7.13	5.66
MAE	4-week-ahead	for 4 weeks	$(8, 4)$	6.58	6.34	4.77
RMSE	4-week-ahead	for 1 week	$(5, 1)$	126.04	119.94	117.12
RMSE	4-week-ahead	for 2 weeks	$(6, 2)$	98.31	90.79	85.84
RMSE	4-week-ahead	for 4 weeks	$(8, 4)$	79.69	72.82	67.67
MAE	1-week-ahead	for 1 week	$(2, 1)$	7.63	7.42	6.42
MAE	1-week-ahead	for 2 weeks	$(3, 2)$	6.69	6.48	5.42
MAE	1-week-ahead	for 4 weeks	$(5, 4)$	6.18	5.90	4.60
RMSE	1-week-ahead	for 1 week	$(2, 1)$	124.36	118.17	115.23
RMSE	1-week-ahead	for 2 weeks	$(3, 2)$	95.48	89.37	85.11
RMSE	1-week-ahead	for 4 weeks	$(5, 4)$	78.86	71.35	66.90

4.3.3 Average performance of the aggregation algorithms: MAE, RMSE

Now that we identified some benchmark performance, we may compare the performance of the aggregation algorithms considered to this performance. We proceed in two steps: first, we tabulate the performance of these algorithms under their various specifications on a given case, namely, $(h, n) = (7, 1)$ corresponding to 6-week-ahead forecasting of 1 week of sales. We show that they all achieve a rather similar performance. For the second part of the study, we thus set (somewhat arbitrarily) a given algorithm with given specifications, namely, ML-Poly with absolute loss, without the gradient trick and with projection, and tabulate its performance depending on (h, n), that is, depending on the forecasting horizon $h - n$ and the number n of weeks to be forecast.

First part: Little impact of the algorithm picked and of its specifications. As explained above and as is summarized in Table 3, we consider three algorithms under $2 \times 2 \times 3 = 12$ possible specifications (given by choices made for the loss function, gradient trick, and projection step). We report the performance of each specification in MAE and RMSE for the case $(h, n) = (7, 1)$.

In terms of RMSEs, the various algorithms and specifications thereof (with one exception) are virtually undistinguishable, with RMSEs all around 120 k€ when the gradient trick is not applied (and slightly larger, up to 125.4 k€ when it is applied). The Euclidean projection barely improves the RMSE (Section 4.2.4 recalls why this projection must improve the RMSE). The exception to the virtually undistinguishable performance is ML-Prod without the gradient trick, which fares much worse than ML-Prod with the gradient trick or the various specifications of ML-Poly and BOA.

A summary of the same kind may be written for MAEs: many of the algorithms and specifications thereof have MAEs around 8 k€ (slightly larger values are suffered when the Euclidean projection is applied). The projection in absolute norm slightly worsens the results (Section 4.2.4 recalls why this projection came with no positive guarantee on its impact on the MAE). The loss function ℓ and the gradient trick have little impact, though the absolute loss seems a slightly better choice than the square loss, and though it seems better not to resort to the gradient trick.

The conclusion from this study is that the choice of the specific aggregation algorithm and of its specification is not of utmost importance. For the rest of the study, we will fix an algorithm (namely, ML-Poly) with the simplest specification: no gradient trick, no projection, and absolute loss (which is in line with our focus on MAE). The BOA algorithm under this simplest specification gets a better performance on the case $(h, n) = (7, 1)$ but we have a personal preference for ML-Poly, which was designed by one of the collaborators of this work.

Second part: Relative performance compared to the meta-predictors. Table 4 studies the performance of a given algorithm under a given specification, namely, ML-Poly with absolute loss, no gradient trick, no projection, as concluded from the paragraphs above. It compares its performance to the one of the three meta-predictors discussed in Section 4.3.2. Two main benchmarks were outlined in the latter section: the locally best predictors on the train set (which is a legal meta-predictor) and the globally best predictor on the test set (which is a forward-looking meta-predictor).

The aggregation algorithm consistently outperforms the locally best predictors on the train set, for all cases (h, n) of horizon $h - n$ and number n of weeks to be forecast, both in MAE and RMSE. The improvement is typically around 5% (it ranges from a minimal 3.6% to a maximal 7.1% relative improvement). We recall that the the locally best predictors on the train set actually depend on the underlying metric: MAE or RMSE.

The situation is two-fold for the comparison of the aggregation algorithm to the globally best predictor on the test set: the former consistently outperforms the latter in our favorite metric, namely, MAE, with relative improvements in the range 2.0%–4.5%. On the opposite, the aggregation algorithm is consistently outperformed by the globally best predictor on the test set in RMSE, within a 0.5%–5.3% range.

Table 5 studies the performance of the same algorithm, ML-Poly, under a slightly different specification: still without a gradient trick and without projection, but with square loss instead of absolute loss. This should favor RMSE performance. The picture is about the same: consistent improvement in performance over the locally best predictors on the train set (with range 2.3%–8.3%); mixed pictures for the comparison to the globally best predictor on the test set, and indeed, the RMSE performance is globally improved.

However, given that our aim is to predict "ordinary" (and not extreme) values, we are more interested in the MAE performance. For MAE performance, the aggregation algorithm considered in Table 4 is consistently better than the forward-looking meta-predictor picking the globally best predictor on the test set (on all 9 cases). For the one of Table 5 (for which we changed the loss function into square loss), the improvement holds for 7 out of 9 cases (and for the 2 other ones, the difference in performance is negligible, smaller than 0.4%).

Table 3 – Average errors in k€ (in MAE, columns 4–6, and in RMSE, columns 7–9) for the case $(h, n) = (7, 1)$, that is, for 6-week-ahead forecasts of 1 week of sales, for the three algorithms considered (ML-Poly, rows 1–4; BOA, rows 5–8; ML-Prod, rows 9–12), under various specifications thereof: loss function used (see Section 4.2.3.2), either the absolute loss $|\cdot|$ or the square loss $(\cdot)^2$, as indicated in column 3; whether the gradient trick (see Section 4.2.3.3) is applied or not (column 2, "yes" or "no"); whether a projection step (see Section 4.2.4) is added or not ("no proj.", columns 4 and 7), and if so, whether a projection in Euclidean norm ("L2–proj.", columns 5 and 8) or in absolute norm ("L1–proj.", columns 6 and 9) is used.

Case $(h, n) = (7, 1)$, i.e., 6-week-ahead forecasts for 1 week								
Algor.	Gradient	Loss	MAE in k€			RMSE in k€		
	trick	ℓ	no proj.	L2–proj.	L1–proj.	no proj.	L2–proj.	L1–proj.
ML-Poly	no	$\lvert\cdot\rvert$	7.97	8.80	8.07	120.39	120.31	120.03
ML-Poly	no	$(\cdot)^2$	8.04	8.85	8.10	119.79	119.71	119.39
ML-Poly	yes	$\lvert\cdot\rvert$	8.05	8.79	8.11	121.90	121.79	121.25
ML-Poly	yes	$(\cdot)^2$	8.26	8.91	8.35	125.40	125.31	124.90
ML-Prod	no	$\lvert\cdot\rvert$	13.28	16.26	13.06	278.45	278.04	226.17
ML-Prod	no	$(\cdot)^2$	12.99	16.39	12.62	277.51	276.98	215.62
ML-Prod	yes	$\lvert\cdot\rvert$	7.94	8.49	8.01	120.54	120.48	120.32
ML-Prod	yes	$(\cdot)^2$	8.10	8.64	8.14	120.17	120.12	119.98
BOA	no	$\lvert\cdot\rvert$	7.93	8.72	8.07	120.39	120.30	120.94
BOA	no	$(\cdot)^2$	8.06	8.68	8.17	120.44	120.37	120.89
BOA	yes	$\lvert\cdot\rvert$	7.97	8.79	8.04	121.73	121.61	121.45
BOA	yes	$(\cdot)^2$	8.12	8.70	8.21	122.60	122.53	122.64

Table 4 – Average errors in k€ (columns 3–6) and relative differences in errors (columns 7–8) for a given aggregation algorithm under a given specification (namely, ML-Poly with the absolute loss ℓ, no gradient trick, no projection), depending on the pairs (h, n) and on the metrics (MAE or RMSE) considered (see columns 1–2). The same cases and meta-predictors as in Table 2 are tabulated: columns 1–4 and 6 are exactly equal to the corresponding columns in Table 2. Column 5 reports the absolute performance of the aggregation algorithm considered ("Aggregation", with short-hand "Aggreg"). Columns 7 and 8 report the relative performance of the aggregation algorithm considered, compared either to the locally best predictors on the train set (short-hand "Loc-Train", column 7) or to the globally best predictor on the test set (short-hand "Glob-Test", column 8). Negative (respectively, positive) numbers in columns 7 and 8 indicate that the performance of the aggregation algorithm is better (respectively, worse) than to the meta-predictor it is compared with. The line titled "Legal meta-predictor" recalls which meta-predictors are legal (i.e., only rely on information available at the time they issues their forecasts, "Yes") and which of them are using future data ("No").

Algorithm ML-Poly, with specifications: ℓ is the absolute loss, no gradient trick, no projection							
Metric in k€	Pair (h, n)	Locally best on train set (= Loc-Train)	Globally best on test set (= Glob-Test)	Aggregation (= Aggreg)	Locally best on test set (= Oracle)	Aggreg. vs. Loc-Train	Aggreg. vs. Glob-Test
Legal meta-predictor		Yes	No	Yes	No		
MAE	$(7, 1)$	8.39	8.30	7.97	6.79	-5.1%	-4.0%
MAE	$(8, 2)$	7.39	7.34	7.12	5.70	-3.6%	-3.1%
MAE	$(10, 4)$	6.80	6.59	6.42	4.85	-5.5%	-2.5%
RMSE	$(7, 1)$	125.68	119.24	120.39	115.59	-4.2%	$+1.0\%$
RMSE	$(8, 2)$	97.26	90.94	93.62	85.78	-3.7%	$+3.0\%$
RMSE	$(10, 4)$	82.36	73.92	78.02	68.23	-5.3%	$+5.3\%$
MAE	$(5, 1)$	8.18	8.20	7.83	6.78	-4.4%	-4.5%
MAE	$(6, 2)$	7.27	7.13	6.83	5.66	-6.0%	-4.2%
MAE	$(8, 4)$	6.58	6.34	6.21	4.77	-5.6%	-2.0%
RMSE	$(5, 1)$	126.04	119.94	120.84	117.12	-4.1%	$+0.8\%$
RMSE	$(6, 2)$	98.31	90.79	92.46	85.84	-6.0%	$+1.8\%$
RMSE	$(8, 4)$	79.69	72.82	75.60	67.67	-5.1%	$+3.8\%$
MAE	$(2, 1)$	7.63	7.42	7.11	6.42	-6.8%	-4.2%
MAE	$(3, 2)$	6.69	6.48	6.30	5.42	-5.8%	-2.8%
MAE	$(5, 4)$	6.18	5.90	5.74	4.60	-7.1%	-2.7%
RMSE	$(2, 1)$	124.36	118.17	118.71	115.23	-4.6%	$+0.5\%$
RMSE	$(3, 2)$	95.48	89.37	90.17	85.11	-5.6%	$+0.9\%$
RMSE	$(5, 4)$	78.86	71.35	73.82	66.90	-6.4%	$+3.5\%$

Table 5 – Same content as in Table 4, still with ML-Poly as an aggregation algorithm, but under a slightly different specification: no gradient trick, no projection (as in Table 4), but with the square loss ℓ (instead of the absolute loss as in Table 4). Only the values of columns 5, 7, 8 differ from the ones of Table 4.

Algorithm ML-Poly, with specifications: ℓ is the square loss, no gradient trick, no projection							
Metric in k€	Pair (h, n)	Locally best on train set (= Loc-Train)	Globally best on test set (= Glob-Test)	Aggregation (= Aggreg)	Locally best on test set (= Oracle)	Aggreg. vs. Loc-Train	Aggreg. vs. Glob-Test
Legal meta-predictor		Yes	No	Yes	No		
MAE	$(7, 1)$	8.39	8.30	8.04	6.79	-4.2%	-3.2%
MAE	$(8, 2)$	7.39	7.34	7.22	5.70	-2.3%	-1.7%
MAE	$(10, 4)$	6.80	6.59	6.61	4.85	-2.8%	$+0.4\%$
RMSE	$(7, 1)$	125.68	119.24	119.79	115.59	-4.7%	$+0.5\%$
RMSE	$(8, 2)$	97.26	90.94	93.23	85.78	-4.1%	$+2.5\%$
RMSE	$(10, 4)$	82.36	73.92	77.44	68.23	-6.0%	$+4.8\%$
MAE	$(5, 1)$	8.18	8.20	7.90	6.78	-3.5%	-3.7%
MAE	$(6, 2)$	7.27	7.13	6.86	5.66	-5.7%	-3.9%
MAE	$(8, 4)$	6.58	6.34	6.36	4.77	-3.3%	$+0.3\%$
RMSE	$(5, 1)$	126.04	119.94	121.55	117.12	-3.6%	$+1.3\%$
RMSE	$(6, 2)$	98.31	90.79	91.53	85.84	-6.9%	$+0.8\%$
RMSE	$(8, 4)$	79.69	72.82	75.62	67.67	-5.1%	$+3.8\%$
MAE	$(2, 1)$	7.63	7.42	7.18	6.42	-5.9%	-3.2%
MAE	$(3, 2)$	6.69	6.48	6.28	5.42	-6.2%	-3.2%
MAE	$(5, 4)$	6.18	5.90	5.79	4.60	-6.2%	-1.8%
RMSE	$(2, 1)$	124.36	118.17	118.93	115.23	-4.4%	$+0.6\%$
RMSE	$(3, 2)$	95.48	89.37	89.17	85.11	-6.6%	-0.2%
RMSE	$(5, 4)$	78.86	71.35	72.33	66.90	-8.3%	$+1.4\%$

4.3.4 An intrinsic evaluation of performance: mean percentages of error

So far, we have been discussing performance in MAE or RMSE and needed benchmarks to assess the quality of the forecasts issued by the aggregation algorithms (and the latter outperformed these benchmarks: the locally best predictors on the train set and the globally best predictor on the test set). Put differently, we were only discussing relative performance. We now want to move to a more intrinsic evaluation of the performance of the aggregation algorithms (and of the meta-predictors). To that end, we use a mean absolute percentage of error [MAPE] as our criterion. The latter is not so easy to define, as many sales $y_{t,\gamma}$ are null (see Section 4.3.1), and therefore, the classical definition

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{\gamma\in\Gamma}\frac{\left|y_{t+h,\gamma}-\widehat{y}_{t+h,\gamma}\right|}{y_{t+h,\gamma}}$$

fails. This is why we adapt this classical definition of MAPE to our needs, as follows. We provide this adaptation for a given a subset $\Gamma_{\mathrm{sub}} \subseteq \Gamma$ of nodes (sometimes Γ_{sub} will be the set Γ of all nodes, and sometimes a strict subset, e.g., given by all subsubfamilies):

$$\mathrm{MAPE} = \frac{1}{T}\sum_{t=1}^{T}\frac{\sum_{\gamma\in\Gamma_{\mathrm{sub}}}\left|y_{t+h,\gamma}-\widehat{y}_{t+h,\gamma}\right|}{\sum_{\gamma\in\Gamma_{\mathrm{sub}}} y_{t+h,\gamma}}$$

When the subset Γ_{sub} is large enough (whenever it contains a significant number of nodes), the denominator is positive and the MAPE is well defined in this way.

4.3.4.1 On the entire hierarchy Γ

We first discuss global performance, on the entire hierarchy of nodes Γ. Figure 3 and Table 6 are counterparts of similar figures and a similar table in the case of MAE and RMSE. They display graphically (Figure 3) the performance in MAPE of the elementary predictors and meta-predictors introduced in Section 4.3.2, as well as the one of a given aggregation algorithm, namely, ML-Poly with the absolute loss, no gradient trick, no projection (just as in Section 4.3.3 above). Of course, all meta-predictors defined in terms of a "best predictor" or "best predictors" as in (4.5) or (4.6)are defined with respect to the loss function

$$\ell\Big(y_{t+h,\gamma},\widehat{y}^{(j)}_{t+h,\gamma}\Big)=\frac{\left|y_{t+h,\gamma}-\widehat{y}^{(j)}_{t+h,\gamma}\right|}{\sum_{g\in\Gamma} y_{t+h,g}}\,. \tag{4.7}$$

(We should actually add arguments to ℓ, as the loss computed depends on all observations $y_{t+h,g}$, not just the one at the node γ.)

In terms of relative performance, Figure 3 and Table 6 for MAPE show a similar ranking as Figure 2 and Table 4 for MAE: the aggregation algorithm consistently outperforms the locally best predictors on the train set and the globally best predictor on the test set.

We are more interested in an intrinsic evaluation of the performance, which is why we

considered MAPE in the first place. The MAPEs of the aggregation algorithm lie between 15.24% and 21.08% (these MAPEs are larger when the horizon is farther away and/or the number of weeks to be forecast is smaller). This is a nice performance, but we break it down by levels of the hierarchy before issuing any deeper comments.

4.3.4.2 Level by level

We now explore MAPE performance by levels of the hierarchy: by taking subsets $\Gamma_{\rm sub}$ given by all subsubfamilies, or by all subfamilies, or by all families. We also report the MAPE for predicting the total sales, i.e., $\Gamma_{\rm sub}$ is the root-node singleton: $\Gamma_{\rm sub} = \{\text{root}\}$. When considering a "best predictor" or "best predictors" for our meta-predictors, similarly to the definition given by (4.7), by updating the summation in the denominator of the latter, we resort to the loss function

$$\ell\Big(y_{t+h,\gamma}, \widehat{y}^{(j)}_{t+h,\gamma}\Big) = \frac{\Big|y_{t+h,\gamma} - \widehat{y}^{(j)}_{t+h,\gamma}\Big|}{\sum\limits_{g\in\Gamma_{\rm sub}} y_{t+h,g}}\,. \tag{4.8}$$

Put differently, "best" is now in terms of MAPE and of the considered level of the hierarchy.

Results are reported in Table 7. We first discuss the intrinsic performance of the aggregation algorithm: it obtains an MAPE of about 32% on the case of all subsubfamilies, which is the most important case to consider. Indeed, the volumes of sales at this level are then broken down into specific products, either existing ones or new products to be launched. The forecasts at this level support and drive the decision-making. This 32% MAPE is comparable to MAPEs observed for the forecasting of sales in retail distribution.

The MAPE performance of course improves as we go up in the hierarchy: it equals about 22% for all subfamilies, 18% for all families, and 12% for the root note. We recall that the MAPE performance for the entire hierarchy (i.e., putting together all levels) equals about 21%.

Now, in terms of relative performance (i.e., when the aggregation algorithm is compared to meta-predictors), we observe that the aggregation algorithm consistently outperforms the legal meta-predictor given by the locally best predictors on the train set, while it outperforms the forward-looking meta-predictor given the globally best predictor on the test set on the two cases that are of most interest for us: all subsubfamilies, and the entire hierarchy; it is outperformed by that forward-looking meta-predictor on the three other cases: root node (also known as total node), all families, all subfamilies (very slightly).

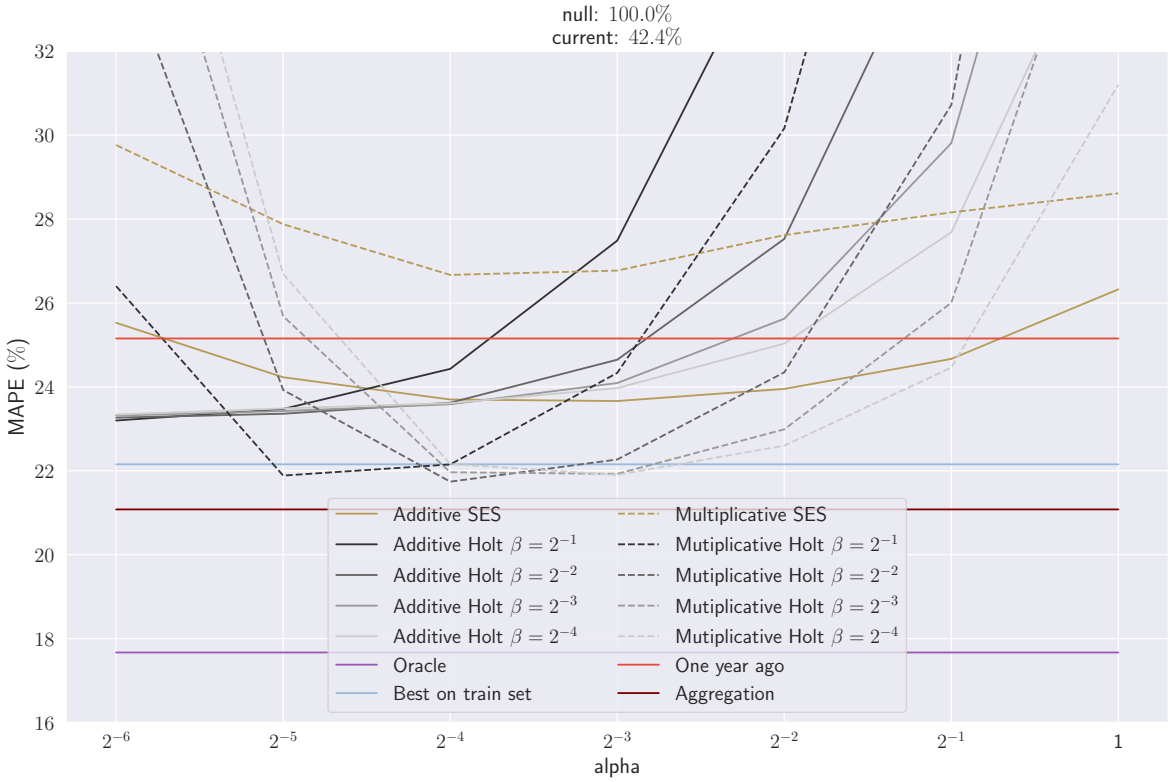


Figure 3 – Performance in MAPE [*y–axis*, in %] of the elementary forecasting methods, of some meta-predictors, and of a given aggregation algorithm to forecast sales 6-week-ahead for 1 week (i.e., for $h = 7$ and $n = 1$), depending on a tuning parameter α [*x–axis*, logarithmic scale]. The same acronyms are used as in Figure 2, with the addition of an `Aggregation` algorithm, namely, ML-Poly with the absolute value, no gradient trick and no projection; its performance is independent of α and is therefore depicted by an horizontal line.

Table 6 – Performance in MAPE (columns 3–6) and relative differences in MAPE (columns 7–8) for a given aggregation algorithm under a given specification (namely, ML-Poly with the absolute loss ℓ, no gradient trick, no projection), depending on the pairs (h, n). The same structure and conventions are used as for Table 4.

Algorithm ML-Poly, with specifications: ℓ is the absolute loss, no gradient trick, no projection							
Metric	Pair (h, n)	Locally best on train set (= Loc-Train)	Globally best on test set (= Glob-Test)	Aggregation (= Aggreg)	Locally best on test set (= Oracle)	Aggreg. vs. Loc-Train	Aggreg. vs. Glob-Test
Legal meta-predictor		Yes	No	Yes	No		
MAPE	$(7, 1)$	22.15	21.74	21.08	17.67	-4.9%	-3.0%
MAPE	$(8, 2)$	19.46	19.04	18.74	14.69	-3.7%	-1.6%
MAPE	$(10, 4)$	18.21	17.46	17.19	12.86	-5.6%	-1.6%
MAPE	$(5, 1)$	21.43	21.34	20.50	17.56	-4.3%	-3.9%
MAPE	$(6, 2)$	18.94	18.47	17.89	14.53	-5.6%	-3.1%
MAPE	$(8, 4)$	17.57	16.75	16.66	12.59	-5.2%	-0.5%
MAPE	$(2, 1)$	19.76	19.33	18.45	16.63	-6.7%	-4.6%
MAPE	$(3, 2)$	17.53	16.76	16.32	13.90	-6.9%	-2.7%
MAPE	$(5, 4)$	16.53	15.57	15.24	12.09	-7.8%	-2.1%

Table 7 – Performance in MAPE (columns 2–5) and relative differences in MAPE (columns 6–7) for a given aggregation algorithm under a given specification (namely, ML-Poly with the absolute loss ℓ, no gradient trick, no projection), depending on the hierarchy level(s) considered. The line "Entire hierarchy" corresponds to taking $\Gamma_{\text{sub}} = \Gamma$, while the four other lines correspond each to an element of a partition of Γ by levels: $\Gamma_{\text{sub}} = \{\text{root}\}$ for the line "Total node", Γ_{sub} the subsets of all families, subfamilies, subsubfamilies, respectively. A similar structure of the results as for Table 4 is used.

MAPE; case $(h, n) = (7, 1)$; algorithm ML-Poly, with specifications: ℓ is the absolute loss, no gradient trick, no projection						
Level	Locally best on train set (= Loc-Train)	Globally best on test set (= Glob-Test)	Aggregation (= Aggreg)	Locally best on test set (= Oracle)	Aggreg. vs. Loc-Train	Aggreg. vs. Glob-Test
Legal	Yes	No	Yes	No		
Entire hierarchy	22.15%	21.74%	21.08%	17.67%	−4.9%	−3.0%
Total node	12.46%	11.34%	11.71%	11.34%	−6.0%	+3.3%
Families	18.70%	16.96%	18.32%	15.56%	−2.0%	+8.0%
Subfamilies	23.46%	22.11%	22.20%	18.49%	−5.4%	+0.4%
Subsubfamilies	33.99%	36.00%	32.09%	25.29%	−5.6%	−10.9%

4.3.5 Beyond average performance

We go beyond average performance measures in this section and illustrate that the performance of the aggregation algorithms is not only better on average but everywhere, compared to, e.g., the natural benchmark given by the locally best predictors on the train set. To do so, we consider the absolute errors suffered for predicting the sales of each of the 3,004 subsubfamilies on each of the 52 weeks of the test set, which leads to $52 \times 3{,}004 = 156{,}208$ absolute errors. We do so for the case $(h, n) = (7, 1)$, i.e., for 6-week-ahead-forecasting of 1 week of sales.

Figure 4 explains where differences in performance between the locally best predictors on the train set, the globally best predictor on the test set, and the aggregation algorithm lie: not on small absolute errors (less than 90 k€, say), but half on medium-sized errors (between 90 and 700 k€, say) and half on large errors (more than 700 k€, say).

Figure 5 shows that there are not many errors that are larger then 700 k€ out of the 156,208 errors considered: fewer than 40 or so. Yet, they account for a significant part of the difference in performance. The aggregation algorithm considered (still ML-Poly with the absolute loss, no gradient trick, no projection) gets fewer of these large errors, and the maximal error it suffers equals about 2 M€, while the maximal error for the locally best predictors on the train set and for the globally best predictor on the test set equal about 4 M€ and 5 M€, respectively.

Figure 6 depicts the histograms of the small absolute errors (smaller than 75 k€). These histograms are, first, virtually indistinguishable, and second, account for most of the errors: they contain almost all of the 156,208 absolute errors considered. Yet, this is not where differences in performance mostly take place.

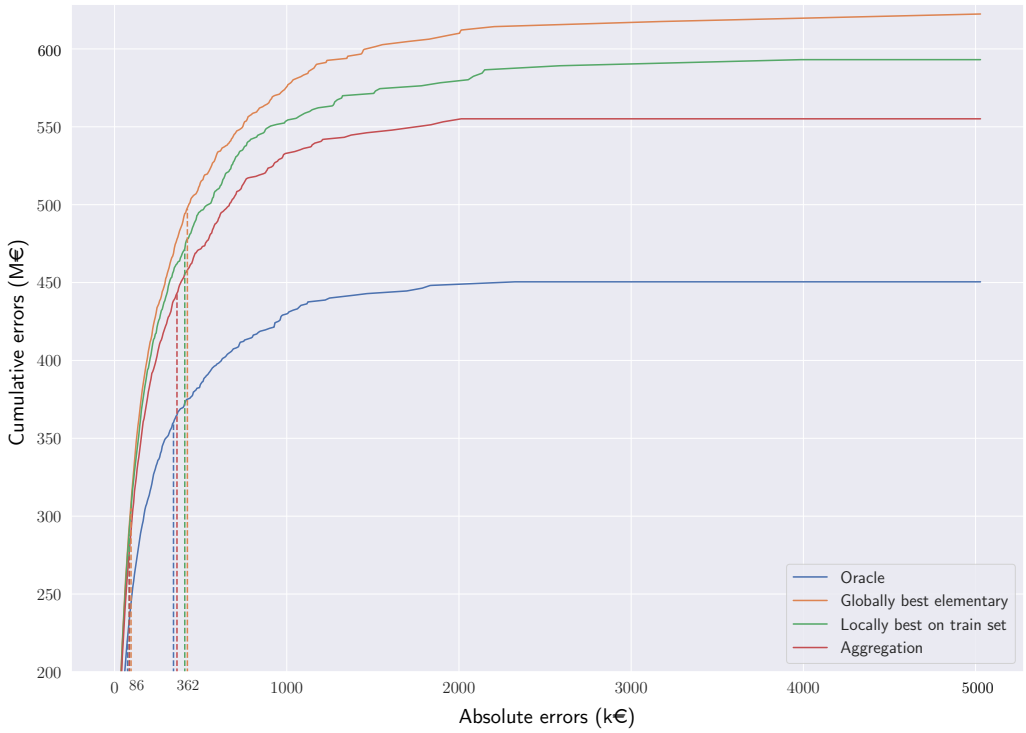


Figure 4 – Cumulative absolute errors (y–axis, units: M€) according to absolute errors (x–axis, units: k€), for three meta-predictors (the locally best predictors on the train set, the globally best predictor on the test set, and the oracle, which corresponds to the locally best predictors on the train set) and an aggregation algorithm (ML-Poly with the absolute loss, no gradient trick, no projection), for the case $(h, n) = (7, 1)$, i.e., for 6-week-ahead-forecasting of 1 week of sales. The dotted vertical lines indicate for each curve the preimages of 50% and 80% of the total cumulative errors; e.g., for the oracle, 50% (respectively, 80%) of the total error is suffered with individual errors smaller than 86 k€ (respectively, 362 k€).

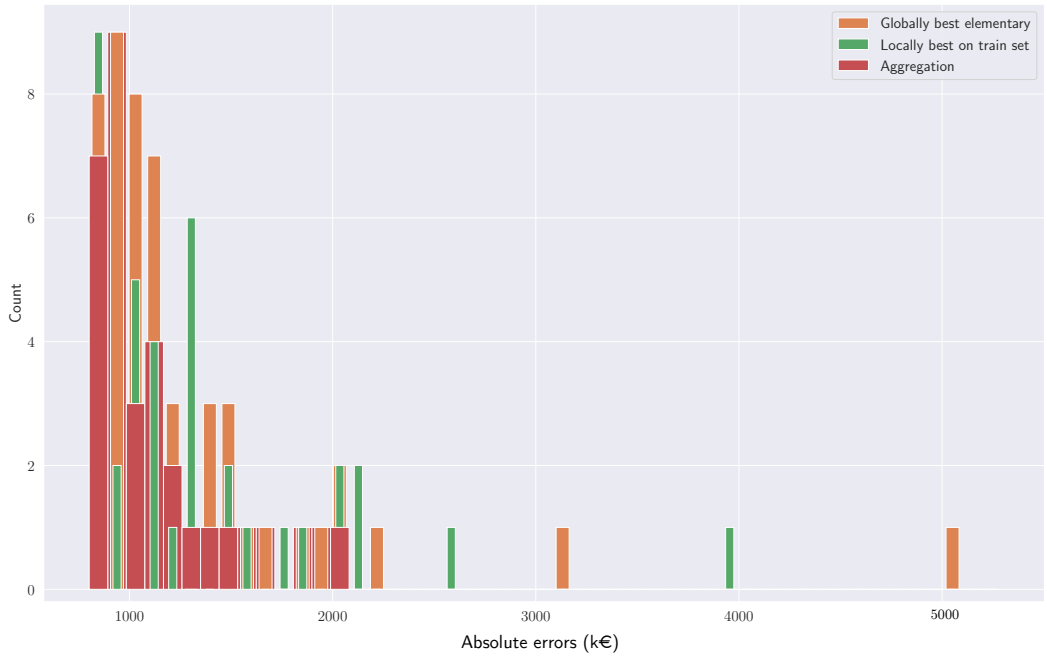


Figure 5 – Histogram count of large absolute errors (larger than 600 k€, see x–axis) for two meta-predictors (the locally best predictors on the train set and the globally best predictor on the test set) and an aggregation algorithm (ML-Poly with the absolute loss, no gradient trick, no projection), for the case $(h, n) = (7, 1)$, i.e., for 6-week-ahead-forecasting of 1 week of sales.

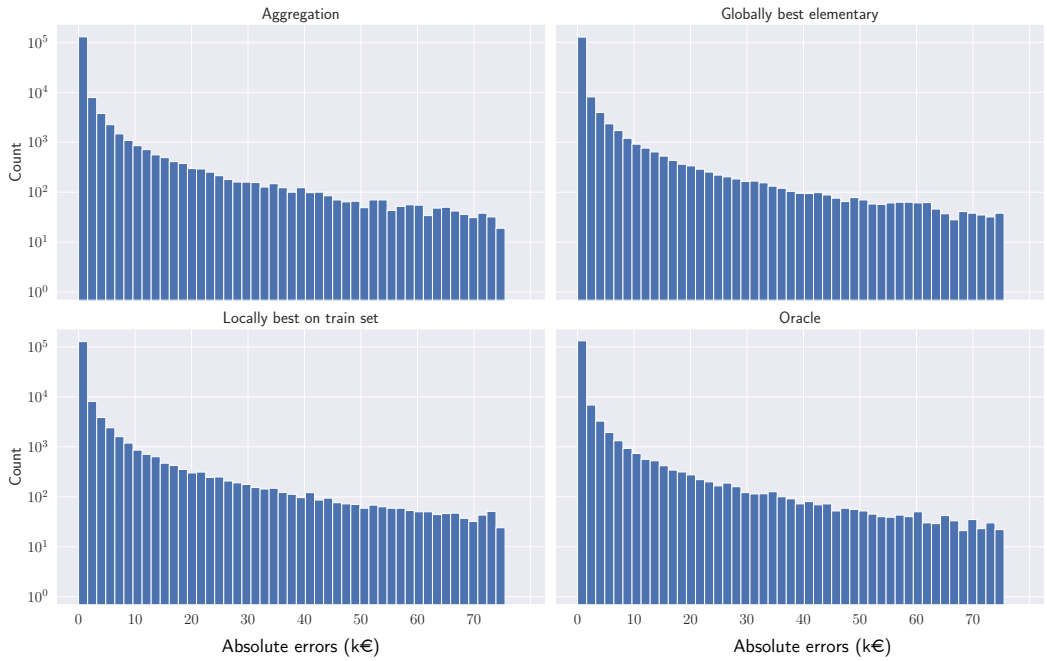


Figure 6 – Histogram counts of small absolute errors (smaller than 75 k€, see x–axis) for three meta-predictors (the locally best predictors on the train set, the globally best predictor on the test set, and the oracle, which corresponds to the locally best predictors on the train set) and an aggregation algorithm (ML-Poly with the absolute loss, no gradient trick, no projection), for the case $(h, n) = (7, 1)$, i.e., for 6-week-ahead-forecasting of 1 week of sales.

4.3.6 Evolution of the weights issued by the aggregation algorithms

The aggregation algorithms considered in Section 4.2.3 issue convex weights: at each prediction step, the forecast $\widehat{y}^{(j)}_{t+h,\gamma}$ of the j–th elementary predictor is assigned a weight $w^{(j)}_{t+h,\gamma}$ and an aggregated forecast is formed according to

$$\sum_{j=1}^{J} w^{(j)}_{t+h,\gamma}\, \widehat{y}^{(j)}_{t+h,\gamma}\,.$$

The vectors $\underline{w}_{t+h,\gamma} = \left(w^{(j)}_{t+h,\gamma}\right)_{1\leqslant j\leqslant J}$ are convex weight vectors: their elements are nonnegative and sum up to 1. A natural question is: do they have any particular structure? Do they converge, e.g., to a Dirac mass on a given elementary predictor?

Section 4.2.2 defined $J = 73$ elementary predictors. Figure 7 depicts the evolutions of the weight vectors picked over time ML-Poly (with the absolute loss, no gradient trick, and no projection step) for the root note (the total sales) and 6 families, which form a representative subset of the 53 families. The main observation is that weights never converge to a Dirac mass on a given elementary predictor. For all cases depicted, at least 5 or 6 elementary predictors, and typically rather 10–15 of them, are used. We see that weights evolve significantly over time, sometimes in a smooth way, sometimes in a more radical way. Only one picture depicts no evolution at all (weights remain uniform): it corresponds to the family "deals", which is one of the 6 families for which the entire series of sales are null (see Table 1).

The evolutions depicted on Figure 7 illustrate that aggregation algorithms are reactive to changes and may reallocate the weights put on elementary predictors when needed. This is in contrast with a meta-predictor like the locally best predictors on the train set, which would need to be recomputed periodically from scratch to accommodate changes.

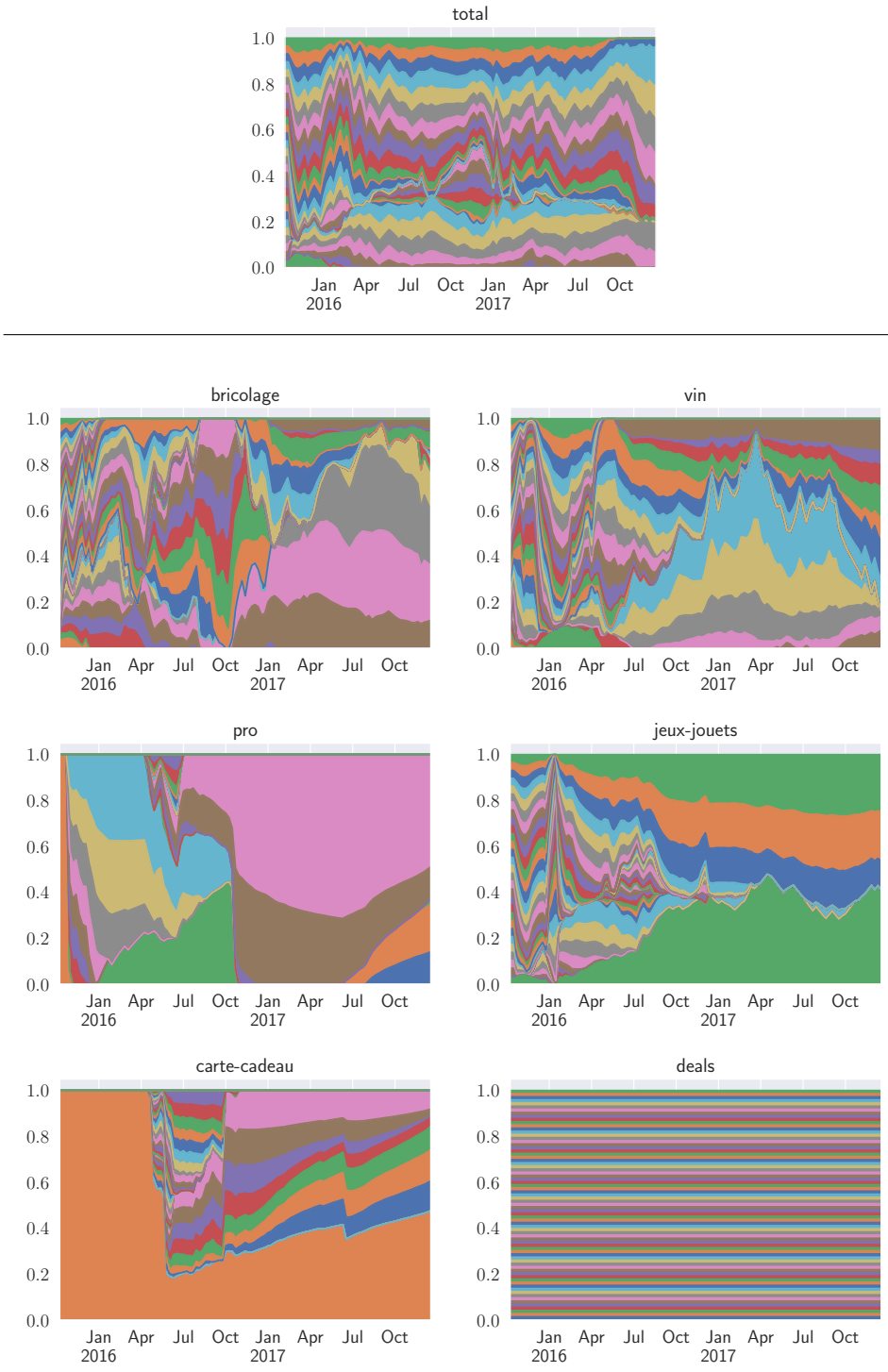


Figure 7 – Evolution of the convex weights put on each elementary predictor over time (from the point in time when all elementary predictors are defined: end of year 2015 to end of year 2017) by ML-Poly (with the absolute loss, no gradient trick, and no projection step), for the prediction of total sales (first line) and the sales of a representative subset of families (lines 2, 3 and 4); namely, from left to right and from top to bottom: DIY–supplies (“bricolage”), wine (“vin”), equipment for professionals (“pro”), toys (“jeux–jouets”), gift cards (‘carte-cadeau”), special offers (“deals”, for which the entire series of sales is null). The weights for each of the 73 elementary predictors are associated with a given color on a given graph and sum up to 1.

Bibliographie

AECOM. Energy Demand Research Project: Early Smart Meter Trials, 2007-2010. Technical report, UK Data Service, 2018. (Cité en pages 58 et 83)

Hesham K. Alfares and Mohammad Nazeeruddin. Electric load forecasting: Literature survey and classification of methods. *International Journal of Systems Science*, 33(1):23–34, 2002. (Cité en page 107)

Christophe Amat, Tomasz Michalski, and Gilles Stoltz. Fundamentals and Exchange Rate Forecastability with Simple Machine Learning Methods. *Journal of International Money and Finance*, 88:1–24, 2018. (Cité en pages 10, 66 et 105)

Anestis Antoniadis, Efstathios Paparoditis, and Theofanis Sapatinas. A functional wavelet kernel approach for time series prediction. *Journal of the Royal Statistical Society: Series B*, 68:837–857, 2006. (Cité en page 108)

Benjamin Auder, Jairo Cugliari, Yannig Goude, and Jean-Michel Poggi. Scalable clustering of individual electrical curves for profiling and bottom-up forecasting. *Energies*, 11(7): 1893, 2018. (Cité en page 57)

Katy S. Azoury and Manfred K. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43(3):211–246, 2001. (Cité en pages 6, 26, 27 et 75)

Kasun Bandara, Peibei Shi, Christoph Bergmeir, Hansika Hewamalage, Quoc Tran, and Brian Seaman. Sales Demand Forecast in E-commerce Using a Long Short-Term Memory Neural Network Methodology. In *Neural Information Processing*, pages 462–474, 2019. (Cité en pages 108 et 110)

Peter L. Bartlett, Wouter M. Koolen, Alan Malek, Eijy Takimoto, and Manfred K. Warmuth. Minimax Fixed-Design Linear Regression. In *Proceedings of The 28th Conference on Learning Theory*, volume 40, pages 226–239, 2015. Proceedings of COLT'2015. (Cité en pages 8, 23, 25, 26, 27, 28, 31, 32, 33 et 46)

Samaneh Beheshti-Kashi, Hamid Reza Karimi, Klaus-Dieter Thoben, Michael Lütjen, and Michael Teucke. A survey on retail sales forecasting and prediction in fashion markets. *Systems Science & Control Engineering*, 3(1):154–161, 2015. (Cité en page 107)

George E. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice-Hall, Englewood Cliffs, 1994. (Cité en page 103)

Margaux Brégère and Malo Huard. Online Hierarchical Forecasting for Power Consumption Data. arXiv preprint number 2003.00585, 2020. (Cité en page 56)

Leo Breiman. Random Forests. *Machine learning*, 45(1):5–32, 2001. (Cité en pages 71 et 105)

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, 1984. (Cité en page 71)

Réal Carbonneau, Kevin Laframboise, and Rustam Vahidov. Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, 184(3):1140–1154, 2008. (Cité en page 107)

Nicolò Cesa-Bianchi. Analysis of two gradient-based algorithms for on-line regression. *Journal of Computer and System Sciences*, 59(3):392–411, 1999. (Cité en pages 4 et 116)

Nicolò Cesa-Bianchi and Gábor Lugosi. Potential-Based Algorithms in On-Line Prediction and Game Theory. *Machine Learning*, 51(3):239–261, 2003. (Cité en page 115)

Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006. (Cité en pages 2, 4, 5, 6, 26, 28, 43, 44, 76, 101, 103, 105, 112 et 116)

Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. A second-order Perceptron algorithm. *SIAM Journal on Computing*, 34(3):640–668, 2005. (Cité en pages 36 et 50)

Nicolas Chapados. Effective Bayesian Modeling of Groups of Related Count Time Series. In *Proceedings of the 31st International Conference on International Conference on Machine Learning*, volume 32, pages 1395–1403, 2014. (Cité en pages 107 et 108)

Chris Chatfield. *Time-Series Forecasting*. Chapman & Hall/CRC, 2000. (Cité en page 103)

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016. (Cité en page 19)

François Chollet et al. Keras, 2015. (Cité en page 19)

Sandra Claudel and Malo Huard. Hydropower scheduling: Learning handmade corrections. Technical report, 2017.

Thomas M. Cover. Universal Portfolios. *Mathematical Finance*, 1(1):1–29, 1991. (Cité en pages 2, 58 et 112)

Marie Devaine, Pierre Gaillard, Yannig Goude, and Gilles Stoltz. Forecasting Electricity Consumption by Aggregating Specialized Experts. *Machine Learning*, 90(2):231–260, 2013. (Cité en pages 10, 58, 66, 92, 105 et 107)

Grzegorz Dudek. Short-Term Load Forecasting Using Random Forests. In *Proceedings of Intelligent Systems'2014*, volume 323, 2015. (Cité en page 108)

Douglas M. Dunn, William H. Williams, and T. L. DeChaine. Aggregate versus Subaggregate Models in Local Area Forecasting. *Journal of the American Statistical Association*, 71(353):68–71, 1976. (Cité en page 57)

Emmanuel A. Donko, Thomas A. Mazzuchi, Soyer, Refik, and Alan Roberson. Urban water demand forecasting: Review of methods and models. *Journal of Water Resources Planning and Management*, 140(2):146–159, 2014. (Cité en page 107)

Shu Fan and Rob J. Hyndman. Short-Term Load Forecasting Based on a Semi-Parametric Additive Model. *IEEE Transactions on Power Systems*, 27(1):134–141, 2011. (Cité en pages 68, 70 et 71)

Jürgen Forster and Manfred K. Warmuth. Relative Loss Bounds for Temporal-Difference Learning. *Machine Learning*, 51:23–50, 2003. (Cité en pages 36 et 49)

Dean P. Foster. Prediction in the Worst Case. *The Annals of Statistics*, 19(2):1084–1090, 1991. (Cité en page 26)

Pierre Gaillard. *Contributions à l'agrégation Séquentielle Robuste d'experts : Travaux Sur l'erreur d'approximation et La Prévision En Loi. Applications à La Prévision Pour Les Marchés de l'énergie.* PhD thesis, Université Paris-Sud, 2015. (Cité en pages 69, 76 et 114)

Pierre Gaillard and Yannig Goude. Forecasting Electricity Consumption by Aggregating Experts; How to Design a Good Set of Experts. In *Modeling and Stochastic Learning for Forecasting in High Dimensions*, Lecture Notes in Statistics, pages 95–115, 2015. (Cité en pages 10, 105 et 107)

Pierre Gaillard and Yannig Goude. Opera package for R: Online Prediction by ExpeRt Aggregation, 2016. (Cité en pages 10, 105 et 114)

Pierre Gaillard, Gilles Stoltz, and Tim van Erven. A Second-order Bound with Excess Losses. In *Proceedings of the 27th Conference on Learning Theory*, volume 35, pages 176–196, 2014. (Cité en pages 80 et 114)

Pierre Gaillard, Yannig Goude, and Raphaël Nedellec. Additive Models and Robust Aggregation for GEFCom2014 Probabilistic Electric Load and Electricity Price Forecasting. *International Journal of Forecasting*, 32(3):1038–1050, 2016. (Cité en pages 10, 19, 58, 68 et 70)

Pierre Gaillard, Sébastien Gerchinovitz, Malo Huard, and Gilles Stoltz. Uniform regret bounds over $\mathbb{R}^d$ for the sequential linear regression problem with the square loss. In

Proceedings of the 30th Conference on Algorithmic Learning Theory, volume 98, pages 404–432, 2019. (Cité en page 24)

Everette S. Gardner. Exponential smoothing: The state of the art. *Journal of Forecasting*, 4 (1):1–28, 1985. (Cité en page 103)

Everette S. Gardner. Exponential smoothing: The state of the art, Part II. *International Journal of Forecasting*, 22:637–666, 2006. (Cité en page 103)

Richard D. Gill and Boris Y. Levit. Applications of the van Trees inequality: A Bayesian Cramér-Rao bound. *Bernoulli*, 1(1–2):59–79, 1995. (Cité en pages 9, 26, 30, 40 et 41)

Benjamin Goehry, Yannig Goude, Pascal Massart, and Jean-Michel Poggi. Aggregation of Multi-Scale Experts for Bottom-up Load Forecasting. *Preprint*, 2019. (Cité en pages 58 et 71)

Yannig Goude. *Mélange de Prédicteurs : Application à La Prévision de Consommation d'électricité*. PhD thesis, Université Paris-Sud, 2008. (Cité en page 10)

Yannig Goude, Raphael Nedellec, and Nicolas Kong. Local Short and Middle Term Electricity Load Forecasting with Semi-Parametric Additive Models. *IEEE Transactions on Smart Grid*, 5(1):440–446, 2014. (Cité en pages 70 et 89)

Charles W. Gross and Jeffrey E. Sohl. Disaggregation methods to expedite product line forecasting. *Journal of Forecasting*, 9(3):233–254, 1990. (Cité en page 57)

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. (Cité en page 5)

Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–417, 1999. (Cité en page 105)

Malo Huard. Prévision séquentielle déterministe par agrégation pour la compétition RTE. Technical report, 2017.

Malo Huard, Gilles Stoltz, and Rémy Garnier. Hierarchical robust aggregation of demand forecasts in e-commerce. arXiv preprint number 2006.03373, 2020. (Cité en page 102)

Rob. J. Hyndman, B. Koehler Koehler, Keith Ord, and Ralph D. Snyder. *Forecasting with Exponential Smoothing: The State Space Approach*. Springer, 2008. (Cité en page 103)

Rob J. Hyndman, Roman A. Ahmed, George Athanasopoulos, and Han Lin Shang. Optimal Combination Forecasts for Hierarchical Time Series. *Computational Statistics and Data Analysis*, 55(9):2579–2589, 2011. (Cité en page 57)

Pooria Joulani, Andras Gyorgy, and Csaba Szepesvári. Online Learning under Delayed Feedback. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 1453–1461, 2013. (Cité en page 69)

Jyrki Kivinen and Manfred K. Warmuth. Exponentiated Gradient versus Gradient Descent for Linear Predictors. *Information and Computation*, 132(1):1–63, 1997. (Cité en pages 4, 77, 80 et 116)

Wojciech Kotłowski, Wouter M. Koolen, and Alan Malek. Random Permutation Online Isotonic Regression. In *Advances in Neural Information Processing Systems 30*, pages 4183–4192, 2017. (Cité en page 33)

Daniel D. Lee and H. Sebastian Seung. Learning the Parts of Objects by Non-Negative Matrix Factorization. *Nature*, 401(6755):788, 1999. (Cité en page 85)

Nick Littlestone and Manfred K. Warmuth. The Weighted Majority Algorithm. *Information and Computation*, 108(2):212–261, 1994. (Cité en pages 2, 3, 58 et 114)

Haipeng Luo, Alekh Agarwal, Nicolò Cesa-Bianchi, and John Langford. Efficient second order online learning by sketching. In *Advances in Neural Information Processing Systems 29*, pages 902–910, 2016. (Cité en pages 36 et 49)

James MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967. (Cité en page 87)

Alan Malek and Peter L. Bartlett. Horizon-Independent Minimax Linear Regression. In *Advances in Neural Information Processing Systems 31*, pages 5264–5273, 2018. (Cité en pages 26 et 27)

Vivien Mallet, Gilles Stoltz, and Boris Mauricette. Ozone Ensemble Forecast with Machine Learning Algorithms. *Journal of Geophysical Research: Atmospheres*, 114(D5), 2009. (Cité en pages 10, 58, 66 et 105)

John T. Mentzer and James E. Cox. Familiarity, application, and performance of sales forecasting techniques. *Journal of Forecasting*, 3(1):27–36, 1984. (Cité en page 103)

Pentti Paatero and Unto Tapper. Positive Matrix Factorization: A Non-Negative Factor Model with Optimal Utilization of Error Estimates of Data Values. *Environmetrics*, 5 (2):111–126, 1994. (Cité en page 85)

Roger Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413, 1955. (Cité en page 53)

Amandine Pierrot and Yannig Goude. Short-Term Electricity Load Forecasting With Generalized Additive Models. In *Proceedings of the 16th Intelligent System Applications to Power Systems Conference*, pages 410–415, 2011. (Cité en page 107)

Adrian E. Raftery, Tilmann Gneiting, Fadoua Balabdaoui, and Michael Polakowski. Using Bayesian model averaging to calibrate forecast ensembles. *Monthly Weather Review*, 133 (5):1155–1174, 2005. (Cité en page 105)

William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971. (Cité en page 87)

Deswarte Raphaël, Gervais Véronique, Stoltz Gilles, and Da Veiga Sébastien. Sequential model aggregation for production forecasting. *Computational Geosciences*, 23(5):1107–1124, 2019. (Cité en pages 10, 66 et 105)

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 2019. (Cité en page 108)

Wolfgang Schellong. Energy Demand Analysis and Forecast. *Energy Management Systems*, pages 101–120, 2011. (Cité en pages 58 et 83)

Matthias W. Seeger, David Salinas, and Valentin Flunkert. Bayesian intermittent demand forecasting for large inventories. In *Advances in Neural Information Processing Systems 29*, pages 4646–4654. Curran Associates, Inc., 2016. (Cité en pages 103 et 107)

Eli Shlifer and Ronald W. Wolff. Aggregation and Proration in Forecasting. *Management Science*, 25(6):594–603, 1979. (Cité en page 57)

Souhaib Ben Taieb, James W. Taylor, and Rob J. Hyndman. Coherent Probabilistic Forecasts for Hierarchical Time Series. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 3348–3357, 2017. (Cité en pages 57 et 83)

Eiji Takimoto and Manfred Warmuth. The Minimax Strategy for Gaussian Density Estimation. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 100–106, 2000. (Cité en pages 26, 27, 30 et 37)

James W. Taylor. Short-term electricity demand forecasting using double seasonal exponential smoothing. *Journal of the Operational Research Society*, 54(8):799–805, 2003. (Cité en pages 89 et 107)

James W. Taylor. Triple seasonal methods for short-term electricity demand forecasting. *European Journal of Operational Research*, 204(1):139–152, 2010. (Cité en page 107)

Tim Van Erven and Jairo Cugliari. Game-Theoretically Optimal Reconciliation of Contemporaneous Hierarchical Time Series Forecasts. In *Modeling and Stochastic Learning for Forecasting in High Dimensions*, Lecture Notes in Statistics, pages 297–317. Springer, 2015. (Cité en page 57)

Harry L. Van Trees. *Detection, Estimation and Modulation Theory*. Wiley & Sons, 1968. (Cité en page 40)

Vladimir Vapnik and Alexey Chervonenkis. *Theory of Pattern Recognition (in Russian)*. Nauka, Moscow, 1974. (Cité en page 2)

Vladimir Vovk. Aggregating strategies. In *Proceedings of the 3rd Workshop on Computational Learning Theory*, pages 372–383, 1990. (Cité en pages 2, 3, 58 et 114)

Vladimir Vovk. Competitive On-Line Statistics. *International Statistical Review*, 69(2): 213–248, 2001. (Cité en pages 6, 8, 9, 26, 27, 30, 37, 75 et 76)

Tri Kurniawan Wijaya, Mathieu Sinn, and Bei Chen. Forecasting uncertainty in electricity demand. In *AAAI-15 Workshop on Computational Sustainability*, volume 60, 2015. (Cité en page 107)

Olivier Wintenberger. Optimal Learning with Bernstein Online Aggregation. *Machine Learning*, 106(1):119–141, 2017. (Cité en pages 76, 79 et 114)

Simon Wood. *Generalized Additive Models: An Introduction with R*. CRC Press, 2006. (Cité en pages 19 et 70)

Qingzheng Xu, Na Wang, and Heping Shi. Review of Croston's method for intermittent demand forecasting. In *Proceedings of the 9th International Conference on Fuzzy Systems and Knowledge Discovery*, pages 1456–1460, 2012. (Cité en page 107)

université PARIS-SACLAY | ÉCOLE DOCTORALE de mathématiques Hadamard (EDMH)

Titre: Apprentissage et prévision séquentiels : bornes uniformes pour le regret linéaire et séries temporelles hiérarchiques

Mots clés: Apprentissage séquentiel, Séries temporelles, Suites individuelles, Bornes de regret, Hiérarchie, Prévisions de ventes

Résumé: Ce travail présente quelques contributions théoriques et pratiques à la prévision des suites arbitraires. Dans ce domaine, la prévision se déroule séquentiellement en même temps que l'apprentissage. À chaque étape, on ajuste le modèle sur les données passées afin de prévoir la prochaine observation. Le but de ce modèle est de faire les meilleures prévisions possibles, c'est-à-dire celles qui minimisent leurs écarts avec les observations. Les méthodes d'apprentissage séquentielles sont évaluées par leur regret, qui mesure à quelle point une stratégie est proche de la meilleure possible, qui est seulement connue une fois l'ensemble des données disponible. Un des résultats de cette thèse est d'étendre l'ensemble des stratégies auxquels on se compare lorsque l'on fait de la régression linéaire séquentielle. Nous avons adapté un algorithme existant en améliorant ses garanties théoriques pour lui permettre de se comparer à n'importe quelle combinaison linéaire constante sans restriction sur la norme de ses poids de mélange. Un deuxième travail a consisté à étendre les méthodes de prévisions séquentielles lorsque les données à prévoir sont hiérarchiquement organisées. Nous avons testé ces méthodes hiérarchiques sur deux applications pratiques, la prévision de consommation électrique des ménages et la prévision de ventes pour le e-commerce.

Title: Sequential learning and prediction: uniform regret bounds and hierarchical time series

Keywords: Sequential Learning, Time series, Individual sequences, Regrets bounds, Hierarchy, Sales predictions

Abstract: This work presents some theoretical and practical contributions to the prediction of arbitrary sequences. In this domain, forecasting takes place sequentially at the same time as learning. At each step, the model is fitted on the past data in order to predict the next observation. The goal of this model is to make the best possible predictions, i.e. those that minimize their deviations from the observations, which are made *a posteriori*. Sequential learning methods are evaluated by their regret, which measures how close strategies are to the best possible, known only after all the data is available. In this thesis, we extend the set of weights vectors a method is compared to when doing sequential linear regression. We have adapted an existing algorithm by improving its theoretical guarantees allowing it to be compared to any constant linear combination without restriction on the norm of its mixing weights. A second work consisted in extending sequential forecasting methods when forcasted data is organized in a hierarchy. We tested these hierarchical methods on two practical applications, household power consumption prediction and demand forecasts in e-commerce.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France