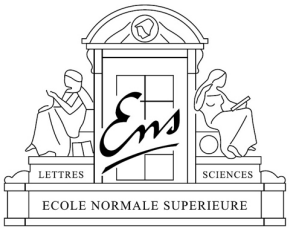
Rapport de stage de M2 MVA

Encadrants : Yannig Goude (EDF) et Gilles Stoltz (ENS de Paris)

Prévision de la consommation électrique (à court terme) par agrégation séquentielle de prédicteurs spécialisés

Pierre Gaillard



Pierre Gaillard
École Normale Supérieure
pierre.gaillard@ens.fr

Paris, le 9 septembre 2011

Table des matières

Je remercie vivement mes deux encadrants Yannig Goude et Gilles Stoltz pour le temps qu'ils m'ont consacré, pour leur aide, ainsi que pour les discussions que nous avons eues ensemble. Je souhaite également les remercier pour avoir partagé avec moi leur travail et leurs idées, pour m'avoir bien guidé tout au long du stage, tout en me laissant la liberté d'explorer ce qui me plaisait.

Plus largement, je tiens aussi à remercier à EDF toute l'équipe du groupe R39 d'EDF R&D, mes voisins du bureau B314 et sa bonne ambiance, ainsi que les autres stagiaires d'EDF qui m'ont permis de passer d'agréables pauses déjeuner. À l'ENS, je remercie tout le DMA et plus particulièrement le petit département de statistiques qui m'a si bien accueilli.

1. Introduction et motivations

L'énergie étant difficilement stockable, la prévision de la consommation électrique représente un problème majeur à EDF. Notre objectif est de prévoir chaque jour la consommation du lendemain de façon la plus précise possible. On ne part heureusement pas de rien, EDF a implémenté plusieurs méthodes de prévisions. Un certain nombre d'experts nous proposent ainsi chaque jour leurs prévisions pour la consommation du lendemain. Le problème consiste à savoir comment mélanger leurs conseils, en leur accordant différentes pondérations, afin de proposer notre propre prévision.

1.1. Cadre du stage

Mon stage s'est effectué dans deux lieux : à l'ENS de Paris, où j'étais encadré par Gilles Stoltz au département de mathématiques et applications ; et à EDF, dans le groupe R39 du département Osiris au centre R& D de Clamart, où Yannig Goude m'a supervisé. Le groupe R39 s'occupe de la prévision de consommation électrique et m'a fourni plusieurs méthodes de prévisions à court terme.

1.2. Travaux déjà effectués et contributions du stage

J'ai travaillé dans la continuation du stage de Marie Devaine en 2009. Celui-ci est résumé dans l'article [DGS11]. Durant mon stage, on s'est intéressé dans un premier temps à faire le point sur les algorithmes de mélange existants et à leur apporter quelques améliorations (généralisation à plusieurs fonctions de pertes,...). On a ensuite comparé ces méthodes de mélange déterministes avec des méthodes stochastiques crées à partir des forêts aléatoires. La généricité de celles-ci nous a amenés à produire quelques extensions pour en améliorer les performances dans le cadre de la consommation électrique.

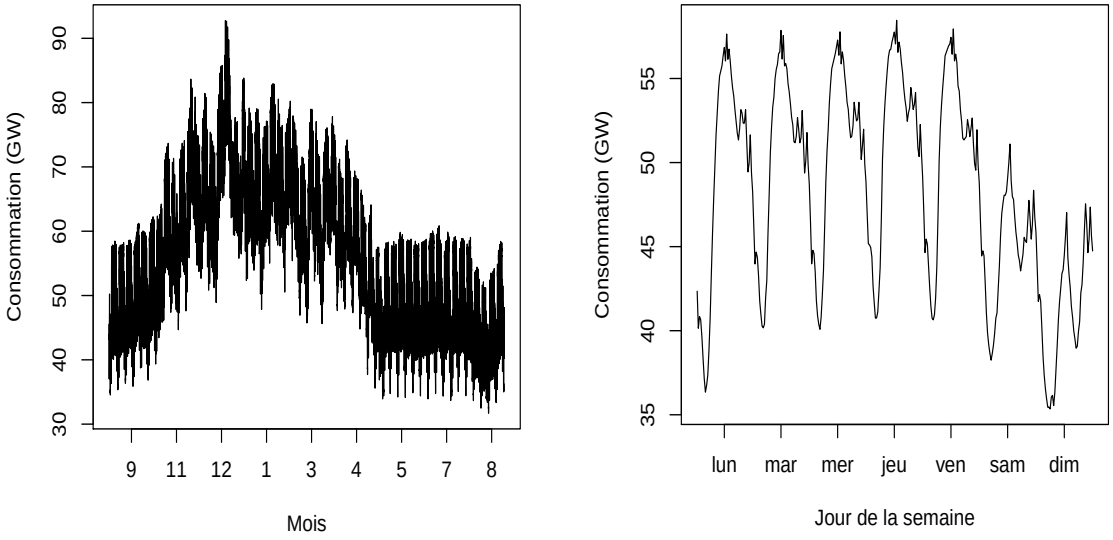


FIGURE 1 – Consommation électrique en France entre le 1er septembre 2007 et le 31 août 2008 [Figure gauche] et au cours d'une semaine typique [Figure droite]. La consommation d'énergie électrique est un processus temporel continu qui dépend de nombreuses variables contextuelles : des variables météo (température, nébulosité, vent, ...), des variables calendaires (type de jour, position dans l'année, ...) et des variables dues aux propriétés de temporelles (consommation de la veille, ...).

2. Théorie des suites individuelles dans le contexte de prédicteurs spécialisés

2.1. Introduction aux suites individuelles

On considère la prévision séquentielle basée sur des conseils d'experts d'une suite individuelle arbitraire. On dispose d'un ensemble $E = \{1, \dots, N\}$ d'experts. À chaque instant $t = 1, \dots, T$, certains experts sont actifs et donnent des prévisions dans un ensemble de sortie convexe $\mathcal{Y}$, typiquement $\mathbb{R}_+$. On note $E_t \subset E$, l'ensemble des experts actifs et f_{it} la prévision de l'expert i si $i \in E_t$. Un algorithme de mélange $\mathcal{A}$ forme alors un mélange $\boldsymbol{p}_t = (p_{1t}, \dots, p_{Nt}) \in \mathbb{R}^N$. Sa prévision est donnée par

$$\widehat{y}_t = \sum_{i \in E_t} p_{it} f_{it} \,.$$

La consommation réalisée y_t est alors révélée et l'instant $t+1$ commence.

On restreint souvent la prévision à un vecteur de poids convexe. C'est à dire, $\boldsymbol{p}_t \in \mathcal{X}_{E_t}$ où $\mathcal{X}_{E_t}$ est un sous-ensemble de $\mathbb{R}^N$ où pour tout $i \in E$, $p_{it} \geqslant 0$; $\sum_{j \in E_t} p_{jt} = 1$ et $\sum_{j \notin E_t} p_{jt} = 0$. Par simplification, $\mathcal{X}$ signifie $\mathcal{X}_E$.

2.2. Estimation de la qualité d'une séquence de prévisions

Pour mesurer la précision d'une prévision $\widehat{y}_t$ proposée à l'instant t pour l'observation y_t, on considère une fonction de perte $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. À chaque instant t, le mélange $\boldsymbol{p}_t$ proposé par l'algorithme est alors évalué par la fonction de perte $\ell_t : \mathcal{X} \to \mathbb{R}$ définie par

$$\ell_t(\boldsymbol{p}) = \ell\left(\sum_{j \in E_t} p_j f_{jt},\, y_t\right)$$

pour tout $\boldsymbol{p} \in \mathcal{X}$. Notre objectif est de trouver des méthodes de mélange $\mathcal{A}$ qui subissent une faible erreur moyenne,

$$\overline{\mathrm{ERR}}_T(\mathcal{A}) = \frac{1}{T}\sum_{t=1}^{T} \ell_t(\boldsymbol{p}_t)\,.$$

En pratique, on a considéré trois fonctions de pertes différentes : la perte carrée, la perte absolue, et le pourcentage d'erreur absolu. Elles nous ont semblé être les plus utilisées en statistiques et plus particulièrement pour la consommation d'énergie électriques. Elles sont toutes positives et convexes en leur premier argument, et permettent ainsi d'obtenir des bornes théoriques. À chacune, on fait correspondre une mesure de dispersion, qui permet de caractériser la variance de nos algorithmes de mélange. Des précisions supplémentaires sont disponibles en Annexe A.

2.3. Les oracles et les stratégies de référence

Intuitivement, si tous les experts sont mauvais, notre algorithme de mélange n'a pas de raison d'obtenir de bons résultats. Pour évaluer les performances de nos algorithmes, nous les comparerons donc à celles d'oracles et de méthodes de mélange de référence. Par oracles, on désigne les stratégies qui ne peuvent pas être définies en ligne mais qui requièrent la connaissance complète du jeu de données dès le départ. On présente ici ces oracles et ces méthodes de mélange de référence et comment on les définit dans le cadre d'experts spécialisés.

Les oracles :

- *Meilleur expert fixé.* On note δ_i l'algorithme qui suit toujours la prévision de l'expert i. Comme elle est mal définie sur tous les instants, on l'évalue uniquement en les instants où l'expert i est actif,

$$\overline{\mathrm{ERR}}_T(\delta_i) = \frac{1}{\sum_{t=1}^{T} \mathbb{1}_{\{i\in E_t\}}} \sum_{t=1}^{T} \ell(f_{it}, y_t)\, \mathbb{1}_{\{i\in E_t\}}\,.$$

  L'oracle du meilleur expert fixé $\mathcal{O}_\delta$ est alors défini par

$$\mathcal{O}_\delta \in \arg\min_{\delta_i} \overline{\mathrm{ERR}}_T(\delta_i)\,.$$

- *Meilleure combinaison linéaire fixée.* Elle correspond à l'utilisation d'un même vecteur de mélange fixé $\boldsymbol{u} \in \mathbb{R}^N$, renormalisé à chaque instant. Plus formellement, on généralise la définition de $\overline{\mathrm{ERR}}_T$ pour tout $\boldsymbol{u} \in \mathbb{R}^N$ par

$$\overline{\mathrm{ERR}}_T(\boldsymbol{u}) = \frac{1}{\sum_{t=1}^{T} \big|\tau_t(\boldsymbol{u})\big|} \sum_{t=1}^{T} \ell_t\!\left(\frac{\boldsymbol{u}}{\tau_t(\boldsymbol{u})}\right) \big|\tau_t(\boldsymbol{u})\big|\,,$$

  où $\tau_t(\boldsymbol{u}) = \dfrac{\sum_{j\in E_t} u_j}{\sum_{k=1}^{N} u_k}$.
  Notez qu'on autorise ici l'attribution de poids négatifs aux experts. L'intuition en pratique dans le cadre de la consommation d'énergie est difficile. Remarquez aussi que l'on retrouve pour $\boldsymbol{u} = \delta_i$ la définition précédente. La meilleure combinaison linéaire fixée est alors donnée par

$$\mathcal{O}^1_{\mathbb{R}^N} \in \arg\min_{\boldsymbol{u}\in\mathbb{R}^N} \overline{\mathrm{ERR}}_T(\boldsymbol{u})\,.$$

- *Meilleure combinaison convexe fixée.* Il s'agit d'un cas particulier de l'oracle précédent avec une contrainte convexe additionnelle sur les vecteurs de poids :

$$\overline{\mathrm{ERR}}_T(\boldsymbol{q}) = \frac{1}{\sum_{t=1}^{T} \tau_t(\boldsymbol{q})} \sum_{t=1}^{T} \ell_t\!\left(\frac{\boldsymbol{q}}{\tau_t(\boldsymbol{q})}\right) \tau_t(\boldsymbol{q})\,,$$

où $\tau_t(\boldsymbol{q}) = \sum_{j \in E_t} q_j$. La meilleure combinaison convexe fixée est alors donnée par

$$\mathcal{O}_{\mathcal{X}} \in \operatorname*{arg\,min}_{\boldsymbol{q} \in \mathcal{X}} \overline{\mathrm{ERR}}_T(\boldsymbol{q}) \,.$$

- *Meilleur expert composé.* L'activation et la désactivation d'experts empêche les algorithmes de mélange de choisir toujours le même expert au cours du temps. On autorise donc quelques changements d'experts et on considère les suites légales avec au plus m changements,

$$\mathcal{L}_\delta^m = \left\{ (\delta_{i_1}, \dots, \delta_{i_T}) \,\middle|\, \forall t \,,\ \ i_t \in E_t \ \text{ and } \ \#\{t,\ i_t \neq i_{t+1}\} \leqslant m \right\}.$$

  On définit ensuite la meilleure séquence ayant au plus m modifications par

$$\mathcal{O}_\delta^m \in \operatorname*{arg\,min}_{\mathcal{A} \in \mathcal{L}_\delta^m} \overline{\mathrm{ERR}}_T(\mathcal{A}) \,.$$

Les algorithmes de mélange de référence :

- *Mélange uniforme.* On note $\mathcal{U}_m$ cette stratégie. Elle forme une moyenne uniforme des prévisions des experts actifs :

$$\widehat{y}_t(\mathcal{U}_m) = \frac{1}{|E_t|} \sum_{i \in E_t} f_{it} \,.$$

- *Vecteur convexe uniforme.* On note $\mathcal{U}_c$ cette stratégie. Elle correspond à l'utilisation du vecteur de mélange uniforme $\mathbf{1}_N = \big(1/N, \dots, 1/N\big)$.

Remarquons que ces deux stratégies de référence donnent les mêmes prévisions mais que leurs performances sont évaluées différemment ; en effet,

$$\overline{\mathrm{ERR}}_T(\mathcal{U}_m) = \frac{1}{T} \sum_{t=1}^{T} \ell\left(\frac{1}{|E_t|} \sum_{i \in E_t} f_{it},\, y_t \right) ,$$

quand

$$\overline{\mathrm{ERR}}_T(\mathcal{U}_c) = \frac{1}{\sum_{t=1}^{T} \mathbf{1}_N(E_t)} \sum_{t=1}^{T} \ell\left(\frac{1}{|E_t|} \sum_{i \in E_t} f_{it},\, y_t \right) \mathbf{1}_N(E_T) \,.$$

3. Algorithmes de mélange et extensions considérés

On présente ici les algorithmes de mélange principaux, ainsi que deux extensions, considérés durant ce stage.

3.1. Les algorithmes de mélange

Trois première familles d'algorithmes de mélange

On considère pour commencer trois familles d'algorithmes de mélange : le mélange par poids exponentiels (EWA), le mélange specialist et le mélange fixed-share. Ces trois premier types d'algorithmes sont présentés en détail dans [DGS11] et rapidement décrit en Annexe D. Ils apparaissent chacun sous deux versions : une de base et une version qui se rapproche d'une descente gradient. On note respectivement $\mathcal{E}_\eta$, $\mathcal{S}_\eta$, et $\mathcal{F}_{\eta\alpha}$ leurs versions basiques et $\mathcal{E}_\eta^{\mathrm{grad}}$, $\mathcal{S}_\eta^{\mathrm{grad}}$, et $\mathcal{F}_{\eta\alpha}^{\mathrm{grad}}$ leurs versions gradient. Les performances théoriques des versions non gradient se rapprochent de l'oracle du meilleur expert. Les versions gradient ont des performances qui tendent vers celle de la meilleure combinaison convexe fixée. En pratique, les algorithmes battent même les oracles avec lesquels ils sont en compétition.

Suivre le leader pénalisé (Ridge)

La définition de la meilleure combinaison linéaire fixée $\mathcal{O}_{\mathbb{R}^N}$, permet de construire un algorithme qui aurait pour objectif de s'en rapprocher. Il s'agit d'une extension de l'algorithme de Ridge au cas des experts spécialisés, et à des fonctions de pertes plus générales que la perte carrée. Aucune borne théorique n'est encore associée à cet algorithme dans le cadre d'experts spécialisés. Le pseudo-code est donné en Algorithme 1.

Algorithme 1 Suivre le leader pénalisé (Ridge) $\mathcal{R}_\lambda$

`Entrée:` paramètre de régularisation λ

`Initialisation:` $u_1 = (1/N, \dots, 1/N)$

`pour` les instants t `de` 1 `à` T `faire`

$$\texttt{prévoir } \widehat{y}_t \leftarrow \frac{1}{\tau_t(\boldsymbol{u}_t)} \sum_{j \in E_t} u_{jt} f_{jt}$$

$$\boldsymbol{u}_{t+1} \leftarrow \operatorname*{arg\,min}_{\boldsymbol{u} \in \mathbb{R}^d} \sum_{s=2}^{t} \ell_s\left(\frac{\boldsymbol{u}}{\tau_s(\boldsymbol{u})}\right) \left|\tau_s(\boldsymbol{u})\right| + \lambda \left\|u\right\|_2^2$$

`fin pour`

Il peut être intéressant de trouver, en suivant la même idée que pour les algorithmes précédents, une version descente gradient de cet algorithme. Pour l'instant, les essais n'ont pas été fructueux.

Les forêts aléatoires comme méthode de mélange stochastique

Durant ce stage, j'ai considéré une méthode de mélange supplémentaire, stochastique. L'idée était non seulement de comparer nos algorithmes de mélange déterministes avec des méthodes stochastiques, mais aussi d'essayer de prendre en compte des variables explicatives, comme la température, que l'on pourrait observer avant de proposer notre vecteur de mélange. Notre méthode est basée sur les forêts aléatoires. On la présente en détails en partie 4.

3.2. Les extensions

Adaptation à une contrainte opérationnelle

Dans [DGS11], les algorithmes de mélange EWA et fixed-share sont implémentés avec une contrainte opérationnelle. Elle consiste à prévoir simultanément chaque jour à 12 :00 les consommations des 48 prochains instants (la prochaine journée). On a étendu ici les algorithmes specialist, EWA et Ridge à cette contrainte de façon générique (cf. Algorithme 2). Pour fixed-share, on a conservé celle proposée dans [DGS11] qui donne de meilleurs résultats. Dans la suite, on considérera toujours les extensions des algorithmes à cette contrainte opérationnelle et on conserve les notations précédentes.

Calibration automatique des paramètres

On présente dans cette partie (Algorithme 3) comment les algorithmes de mélange peuvent être adaptés en des algorithmes adaptatifs en les paramètres. L'idée consiste à considérer une grille de paramètres potentiels et à choisir à tout instant celui qui aurait été le meilleur jusqu'alors.

Prenons, par exemple, les algorithmes de type EWA et de type specialist qui ne dépendent que du paramètre d'apprentissage η. Dans les expériences, on a initialisé la grille autour de la valeur théorique optimale, $\Lambda \simeq 1/B$. Comme l'erreur moyenne n'est pas nécessairement une fonction convexe des paramètres (Figure 2), on doit traiter l'initialisation avec précaution. Une graine trop petite mène à un long temps de convergence et à une performance proche du mélange uniforme $\mathcal{U}_m$. Au contraire,

Algorithme 2 Extension d'un algorithme de mélange $\mathcal{A}$ à la contrainte opérationnelle.

`Entrée:` algorithme de mélange $\mathcal{A}$
`Initialisation:` vecteur de poids uniforme $\boldsymbol{w}_1 \in \mathcal{X}$

`pour` les instants t `de` 1 `à` T `faire`
  `prévoir` $\widehat{y}_t \leftarrow \dfrac{1}{\sum_{i\in E_t} w_{it}} \sum_{j\in E_t} w_{jt} f_{jt}$
  `si` $t = 48k$ `pour un certain` k
    $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{p}_{t+1}(\mathcal{A})$ // synchroniser, voir l'annotation [a]
  `sinon`
    $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t$ // ne pas mettre à jour
  `fin si`
`fin pour`

a. $\boldsymbol{p}_{t+1}(\mathcal{A})$ est le vecteur de poids convexe choisi par $\mathcal{A}$ après avoir observé $y_1, \dots, y_t$ et les prévisions des experts correspondantes

un paramètre trop grand mène à une instabilité et au risque de ne converger que vers un optimum local. Quand la valeur optimale $\eta^\star$ du paramètre atteint le bord supérieur (resp., inférieur) de la grille Λ, on agrandie celle-ci en ajoutant les valeurs $\eta^\star/2$, $\eta^\star/4$, et $\eta^\star/8$ (resp., $2\eta^\star$, $4\eta^\star$, et $8\eta^\star$). Ce n'est bien sûr pas la seule manière de procéder, mais elle semble proposer un bon compromis entre complexité et performance. On a en effet essayé d'agrandir la grille avec un nombre plus grand de paramètres, mais les bénéfices n'étaient pas considérables. On a de plus testé plusieurs valeurs pour le facteur géométrique et 2 donnait de bons résultats.

Algorithme 3 Extention d'une famille d'algorithme de mélange $(\mathcal{A}_\eta)$ vérifiant la contrainte opérationnelle à une famille d'algoritmes de mélange adaptative.

`Entrée:` grille initiale Λ de paramètre potentiels, famille d'algorithme $(\mathcal{A}_\eta)$ vérifiant la contrainte opérationnelle

`Initialisation:` $\eta^\star \leftarrow$ tiré aléatoirement dans Λ // voir annotation [a]
`pour` les instants t `de` 1 `à` T `faire`
  `prévoir` $\widehat{y}_t \leftarrow \widehat{y}_t\,(\mathcal{A}_{\eta^\star})$ // voir annotation [b]
  `si` $t = 48k$ `pour un certain` k `mettre à jour`
    $\eta^\star \in \arg\min_{\eta\in\Lambda} \overline{\mathrm{ERR}}_t\,(\mathcal{A}_\eta)$
    `si` $\eta^\star \in \partial\Lambda$ `agrandir` Λ de façon exponentielle // voir annotation [c]
  `fin si`
`fin pour`

a. Gardez en mémoire que la plupart des méthodes de mélange utilisent une moyenne uniforme des prévisions des experts jusqu'à ce qu'elles accèdent à un premier retour d'information (i.e., jusqu'à l'instant $t = 48$ quand la contrainte opérationnelle doit être satisfaite).

b. $\widehat{y}_t(\mathcal{A}_\eta)$ est la prévision de l'algorithme de mélange $\mathcal{A}_\eta$ (avec le paramètre constant η) après l'observation de la suite $y_1, \dots, y_t$ et des prévisions des experts correspondantes ; n'oubliez pas que $\mathcal{A}_\eta$ vérifie la contrainte opérationnelle et ne peut donc pas accéder à toute l'information passée.

c. $\partial\Lambda$ désigne la frontière de Λ.

Pour les algorithmes de type fixed-share, on peut considérer une grille fixe pour α qui est borné dans $[0, 1]$. La version adaptative de fixed-share s'en déduit ensuite en considérant une grille adaptative pour le paramètre η, comme on l'a fait plus tôt. On choisit alors à chaque instant le meilleur couple de paramètre (η, α) rétrospectivement dans la grille en cours.

4. Digression sur les forêts aléatoires et comment les utiliser pour le mélange

Dans cette partie, on adopte une approche stochastique. La consommation d'électricité $(Y_t) \in \mathbb{R}^T$ et les conseils d'experts $(F_{jt}) \in \mathbb{R}^{N\times T}$ sont à présent modélisés par un processus temporel. Ils peuvent dépendre de variables contextuelles $(X_t) \in \mathbb{R}^{d\times T}$ qui peuvent être observées à chaque instant avant que la prévision ne soit proposée. On note $(\mathcal{F}_t)$ une filtration du passé, définie pour tout $t \geqslant 1$ par

$$\mathcal{F}_t \quad = \quad \sigma\Big(\big\{(X_s, Y_s)\,,\ 1 \leqslant s \leqslant t-1\big\} \cup \big\{X_t\big\}\Big)\,.$$

On admet de plus que la consommation d'électricité Y_t peut s'écrire sous la forme

$$Y_t = \mathbb{E}_t[Y_t] + \varepsilon_t\,,$$

où $\mathbb{E}_t[\,\cdot\,] = \mathbb{E}\,[\,\cdot\,|\mathcal{F}_t]$ correspond à l'espérance conditionnelle sachant le passé $\mathcal{F}_t$ et chaque $\varepsilon_t \sim \mathcal{N}\big(0, \sigma^2\big)$ est un bruit aléatoire Gaussien indépendant de $\mathcal{F}_t$.

On se propose, dans le cadre de ce modèle, d'utiliser les forêts aléatoires pour créer une nouvelle méthode d'agrégation qui prend en compte l'information contextuelle (X_t) avant de proposer un mélange. Les performances des experts peuvent en effet dépendre des variables contextuelles (X_t).

Commençons par présenter rapidement les forêts aléatoires. Nous expliquerons ensuite comment les utiliser pour le mélange de prédicteurs.

4.1. Introduction aux forêts aléatoires

Les forêts aléatoires sont une modification du bagging [1] qui construit une grande collection d'arbres de prévision décorrélés, avant de les moyenner. Les arbres sont construits profondément, ils ont donc tendance à surapprendre les données. Ils ont un biais faible mais une forte variance. Les moyenner permet, s'ils sont non corrélés, de diminuer la variance, sans augmenter le biais.

Donnons l'intuition. Si les prévisions des arbres sont identiquement distribuées, de variance σ^2, avec un coefficient de corrélation deux à deux ρ, la variance de la moyenne arithmétique de K prévisions est alors,

$$\bar{\sigma}^2 = \frac{1}{K^2}\left(K\sigma^2 + K(K-1)\rho\sigma^2\right) = \rho\sigma^2 + \frac{1-\rho}{K}\sigma^2\,.$$

Quand le nombre K d'arbres dans la forêt augmente, le second terme disparaît mais le premier reste. La corrélation ρ entre les arbres, si elle est trop forte, limite l'intérêt de moyenner. L'idée des random forests est de diminuer la variance $\bar{\sigma}^2$, en décorrélant autant que possible les arbres les uns des autres sans trop augmenter leur variance σ^2. Cela se fait, à l'aide de sélections aléatoires des covariables sur lesquelles on coupe au niveau des nœuds des arbres.

Les arbres construits sont très similaires aux arbres CART. On remarque néanmoins deux légères différences. Dans CART, au niveau d'un nœud, on sélectionne la meilleure coupe parmi toutes les variables contextuelles, et non parmi un sous-ensemble aléatoire. Cet ajout n'a été motivé que par le fait que la forêt contient plusieurs arbres que l'on veut les plus indépendants possible les uns des autres. Dans les random forests, il n'y a de plus pas d'étape d'élagage (pruning) après la construction de l'arbre, comme dans CART. Cela avait en effet pour fonction de diminuer la variance, ce que l'on fait déjà ici par le bagging.

1. Abbréviation de "bootstrapping and averaging". Méthode qui consiste à construire une collection de prédicteurs faibles en ne sélectionnant qu'une partie de l'ensemble d'entraînement pour chacun d'eux (bootstrapping) avant de les moyenner (averaging).

Références bibliographiques

La méthode a été introduite par Leo Breiman [Bre01] bien que de nombreuses idées soient apparues plus tôt dans la littérature, comme le bagging [Bre96], ou les arbres CART. Elle est implémentée en `R` avec le paquet `randomForest`, maintenu par Andy Liaw, disponible sur le site du `CRAN`. Le site web `http://www.stat.berkeley.edu/~breiman/RandomForests` donne accès à de la documentation, du code et de nombreux rapport techniques. Une explication détaillée dans le cadre des algorithmes de classification ou de régression est disponible dans le livre [HTF09]. Les preuves de convergence ne sont pas évidentes et sont souvent obtenues pour des modèles simplifiés et éloignés de ce qui est considéré en pratique. Dans le cadre de la régression, [LJ06] a tenté d'expliquer les random forests par leur similitudes avec les K plus proches voisins. Plus récemment, [BDL08] ont prouvé des théorèmes de convergence universelle pour les algorithmes de moyennage, dont les random forests sont un cas particulier.

Le cadre théorique

On dispose d'un ensemble d'entraînement $(X_t, Y_t)_{t\in S_0}$, qui est modélisé par un processus supposé indépendant et identiquement distribué de loi $\mathcal{P}$. Pour $t \geqslant 0$, $X_t = (X_{t1}, \ldots, X_{tM})$ est la réalisation au temps t des M variables contextuelles, observables, pouvant expliquer la sortie $Y_t \in \mathbb{R}$. Chaque covariable X_{tm}, $1 \leqslant m \leqslant M$, est à valeurs dans un ensemble $\mathcal{X}_m$, qui est soit muni d'un ordre total, comme $\mathbb{R}$ par exemple, soit fini (ensemble de catégories).

L'objectif est, à partir de l'observation des variables contextuelles X, de prévoir la sortie Y d'un nouveau couple (X, Y) tiré selon $\mathcal{P}$, en faisant la plus petite erreur quadratique possible. Les random forests proposent une solution efficace à ce problème, présenté comme l'algorithme 4.

Algorithme 4 Régression par Random Forest

`Entrées :` $(X_t, Y_t)_{t\in S_0}$, ensemble d'entraînement ; $\boldsymbol{x} \in \mathcal{C}$, covariables pour la valeur à prévoir ; K, nombre d'arbres dans la forêt ; m, nombre de covariables sélectionnées à chaque coupe ; n_{feuille}, nombre de covariables X_t de l'ensemble d'entraînement maximal par feuille.

`Pour` k `de` 1 `à` K, construire l'arbre T_k ainsi :

1. Choisir un ensemble d'entraînement pour cet arbre (bootstrapping) :
   $S_k \leftarrow$ Tirer N fois avec remise et uniformément dans S_0
2. `Initialiser :` $T_k \leftarrow$ racine contenant tout l'ensemble bootstrap S_0
3. `Tant que` T_k contient une feuille ayant plus de n_{feuille} données `faire`
   a. $M' \leftarrow$ choisir uniformément m parmi les M covariables
   b. Choisir la meilleure variable et la meilleure coupe parmi les m covariables
   $$(j^\star, c^\star) = \operatorname*{arg\,min}_{j\in M',\, c\in C_j} \min_{a\in\mathbb{R}^2} \sum_{l=1}^{2} \sum_{X_i\in c_l} (Y_i - a_l)^2 ,$$
   où C_j est l'ensemble des façons de couper les données présentes en deux ensembles ordonnés si un ordre est disponible pour j.
   c. Transformer la feuille en un nœud avec deux feuilles. Associer à chaque feuille respectivement les ensembles de variables contextuelles $c_1^\star$ et $c_2^\star$ et les données d'entraînement correspondantes.
4. Faire descendre $\boldsymbol{x}$ dans l'arbre jusqu'à une feuille d'ensemble de covariable associé $\mathcal{F}_k(\boldsymbol{x})$ et prévoir
   $h_k(\boldsymbol{x}) \leftarrow \frac{1}{|\mathcal{F}_k(\boldsymbol{x})|} \sum_{t:X_t\in\mathcal{F}_k(\boldsymbol{x})} Y_t$

`Renvoyer :` $h(\boldsymbol{x}) = \frac{1}{K} \sum_{k=1}^{K} h_k(\boldsymbol{x})$

Trouver la bonne coupe

Chaque nœud interne d'un arbre des forêts aléatoires divise l'ensemble des données en deux. Un point essentiel lors de la construction des arbres est de déterminer la bonne partition qui regroupe au mieux les données selon la variable à expliquer Y_t. Pour diminuer la complexité du problème, on ne considère que les coupes se faisant le long d'une seule variable contextuelle. Notons C_j, l'ensemble des partitions possibles selon la covariable j. Si celle-ci est réelle, on ne considère dans C_j que les partitions en deux intervalles.

On peut définir l'erreur quadratique d'une couple covariable–partition $(j, c) \in \{1, \dots, M\} \times C_j$, où $c = (c_1, c_2)$ est une partition de l'ensemble $\mathcal{X}_m$ des covariables en deux, par

$$\mathcal{E}(j, c) = \min_{a \in \mathbb{R}^2} \sum_{l=1}^{2} \sum_{X_i \in c_l} (Y_i - a_l)^2 \ .$$

Le meilleur couple covariable–partition d'un sous ensemble de variables contextuelles M', est alors donné par le couple minimisant son erreur quadratique,

$$(j^\star, c^\star) = \underset{j \in M',\ c \in C_j}{\arg\min}\ \mathcal{E}(j, c)\,.$$

Pour chaque variable contextuelle, à valeurs dans un ensemble ordonné, la détermination d'une meilleure coupe se fait très rapidement (sa complexité est linéaire en le nombre de données présentes) grâce à l'égalité

$$\underset{a \in \mathbb{R}}{\arg\min} \quad \sum_{X_i \in I} (Y_i - a)^2 = \left\{ \frac{1}{\#\{i, X_i \in I\}} \sum_{X_i \in I} Y_i \right\} .$$

Cependant, si la covariable j prend ses valeurs dans un ensemble de q catégories non ordonnées, il y a $2^{q-1} - 1$ partitions possibles de l'espace en deux groupes, et le calcul devient vite impossible quand q est grand. La méthode consiste à ordonner les catégories par la moyenne des sorties Y_i appartenant à cette catégorie. On coupe alors comme si on avait une variable ordonnée. On est ramené à une complexité linéaire en q ! On peut montrer que cela donne bien la coupe optimale [Fis58]. Bien qu'intuitive la preuve est loin d'être triviale.

Proximités

Les random forests induisent une notion de proximité entre les entrées X_t. C'est l'un des outils les plus utiles de la méthode. Intuitivement, si deux ensembles de covariables X_{t_1} et X_{t_2} tombent souvent dans les mêmes feuilles des arbres, on peut supposer qu'elles expliquent la sortie Y de façon similaire. Plus formellement, après que les arbres de la forêt ont été construits, on fait descendre toutes les données au niveau des feuilles et on définit la proximité entre deux observations t_1 et t_1 par

$$\mathrm{prox}(X_{t_1}, X_{t_2}) = \frac{1}{K} \sum_{k=1}^{K} \mathbb{1}_{\{X_{t_1} \text{ et } X_{t_2} \text{ tombent dans la même feuille dans l'arbre } k\}} \cdot$$

On remarque que la prévision des random forests,

$$h_K(\boldsymbol{x}) = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{\#\mathcal{F}_k(\boldsymbol{x})} \sum_{t : X_t \in \mathcal{F}_k(\boldsymbol{x})} Y_t \,,$$

est proche de

$$\begin{aligned} \frac{1}{\sum_{k=1}^{K} \#\mathcal{F}_k(\boldsymbol{x})} \sum_{k=1}^{K} \sum_{t : X_t \in \mathcal{F}_k(\boldsymbol{x})} Y_t \quad &= \quad \frac{1}{\sum_{t \in S_0} \mathrm{prox}(\boldsymbol{x}, X_t)} \sum_{t \in S_0} \mathrm{prox}(\boldsymbol{x}, X_t)\, Y_t \\ &\in \quad \underset{a \in \mathbb{R}}{\arg\min} \sum_{s \in S_t} \mathrm{prox}(X_t, X_s)\, (Y_s - a)^2 \ , \end{aligned}$$

où l'égalité procède d'une simple récriture utilisant la définition de la proximité et où la réécriture comme un argmin est obtenue par une minimisation de la perte carrée pondérée par la proximité. C'est cette idée, que nous avons suivi, pour construire le prédicteur RF_1 et ses dérivés en Annexe G.

En pratique

Dans mes implémentations des random forests, j'ai pris :

$$\left\{\begin{array}{rcl} n_{\text{feuille}} & = & 3 \\ m & = & \lfloor M/3 \rfloor \simeq 19 \\ K & = & 400\,. \end{array}\right.$$

4.2. Les forêts aléatoires comme méthode de mélange stochastique

L'idée est de considérer les variables contextuelles (X_t) disponibles au début de chaque instant et dont les valeurs sont un bon indicateur de performance des différents experts.

À l'instant t, on note $\widehat{Y}_t$ la prévision de cette méthode de mélange et par $\widehat{L}_t = \ell(\widehat{Y}_t, Y_t)$ et $L_{it} = \ell(F_{it}, Y_t)$ les pertes respectivement subies par elle et par l'expert i. On suppose que toutes les pertes L_{it} peuvent être modélisées par

$$L_{it} = \mathbb{E}_t[L_{it}] + \varepsilon'_{it}\,,$$

où $\varepsilon'_{it} \sim \mathcal{N}(0, \sigma_i'^2)$ est un bruit Gaussien indépendant du passé (i.e., de $\mathcal{F}_t$).

À l'instant t, on obtient l'estimation $\widehat{\ell}_{it}$ de l'espérance conditionnelle de la perte subie par l'expert i. Cette estimation peut être obtenue à l'aide de n'importe quelle méthode de régression, comme les forêts aléatoires. On admet qu'elle peut se décomposer

$$\widehat{\ell}_{it} = \mathbb{E}_t[L_{it}] + \varepsilon_{it}\,,$$

où $\varepsilon_{it} \sim \mathcal{N}\big(0, \sigma_t^2\big)$ est un bruit Gaussien indépendant du passé $\mathcal{F}_t$ et des autres bruits $(\varepsilon_{jt})_{j\neq i}$ et (ε'_{jt}).

Plus précisément, au cours de nos expériences, on a construit une forêt aléatoire de base utilisant l'information disponible jusqu'à l'instant en cours comme ensemble d'entrainement $(X_s, Y_s)_{1\leqslant s\leqslant t-1}$. L'estimation de la perte de l'expert i est alors donnée par

$$\widehat{\ell}_{it} = \frac{1}{\sum_{s=1}^{t-1} \operatorname{prox}(s,t)} \sum_{s=1}^{t-1} \operatorname{prox}(X_s, X_t)\, L_{is}\,. \tag{1}$$

On en déduit l'algorithme de mélange stochastique présenté en Algorithme 5.

Théorème 1. *Sous les hypothèses précédentes, le regret de l'algorithme de mélange par les forêts aléatoires décrit ci-dessus, calculé avec la suite de paramètres d'apprentissage (η_t), peut être borné par*

$$\sum_{t=1}^{T} \mathbb{E}_t\Big[\widehat{L}_t\Big] - \min_{1\leqslant j\leqslant N} \mathbb{E}_t[L_{jt}] \leqslant N^2 \left(\sum_{t=1}^{T} \sigma_t + \frac{1}{N\eta_t}\right).$$

La preuve de la borne se trouve en Annexe F. La meilleure valeur théorique est à la vue de cette borne $\eta_t = +\infty$. Cela revient à donné un poids de 1 à l'expert ayant la plus faible estimation de perte $\widehat{\ell}_{jt}$ et 0 à tous les autres experts. La borne devient alors

$$\sum_{t=1}^{T} \mathbb{E}_t\Big[\widehat{L}_t\Big] - \min_{1\leqslant j\leqslant N} \mathbb{E}_t[L_{jt}] \leqslant N^2 \sum_{t=1}^{T} \sigma_t$$

Algorithme 5 Mélange stochastique par forêts aléatoires $\mathcal{T}_\eta$

`Entrée:` suite de paramètre d'apprentissage (η_t), possiblement constante

`pour` les instants t `de` 1 `à` T `faire`
  `Construire` la forêt aléatoire avec les données $(X_s, Y_s)_{s \leqslant t-1}$ observées jusqu'à présent
  `pour` l'expert j `de` 1 `à` N `faire`
    `observer` F_{jt}
    `obtenir` $\widehat{\ell}_{jt}$ // estimée par la forêt construite ci-dessus en utilisant (1)
    $w_{jt} \leftarrow e^{-\eta_t \widehat{\ell}_{jt}}$ // mettre à jour
  `fin pour`
  `prévoir` $\widehat{y}_t \leftarrow \dfrac{1}{\sum_{i \in E_t} w_{it}} \sum_{j \in E_t} w_{jt} F_{jt}$
  `observer` y_t
`fin pour`

(et peut être facilement améliorée en une borne linéaire en N). Cependant, pour $\eta_t \simeq (1/N) \sum_{t=1}^{T} \sigma_t$, on ne perd qu'un facteur constant en T par rapport à cette borne théorique optimale alors que la performance en pratique est améliorée d'environ 10 pour cent.

La borne est écrite avec un paramètre η_t qui peut évoluer avec le temps t. Cependant, en pratique, nous n'avons utilisé que des paramètres η_t fixes. Il peut être intéressant dans un deuxième temps d'étudier la calibration en ligne du paramètre η_t, comme on l'a fait pour les autres méthodes d'agrégation.

5. Expériences dans le cadre de la prévision de consommation électrique

5.1. Description du jeu de données

Le jeu de données, dorénavant appelé ensemble de prévisions, consiste en les observations au pas demi-horaires de la consommation d'électricité en France du 1er septembre 2007 au 31 août 2008 et des prévisions sur cette période de 24 experts. Les unités sont des Gigawatts (GW). Comme la perte carrée et la perte absolue ne sont pas invariantes au changement d'échelle, les différents paramètres en jeu peuvent dépendre de l'unité choisie pour ces deux fonctions de perte (au contraire de ce qui arrive avec le pourcentage d'erreur absolue).

Nombre de jours D	320
Intervalles de temps	30 minutes
Nombre d'instants T	15 360 $(= 320 \times 48)$
Nombre d'experts N	24 $(= 15 + 8 + 1)$
Unité	GW
Médiane des y_t	56.33
Borne B sur les y_t	92.76

TABLE 1 – Quelques caractéristiques des observations y_t (consommations demi-horaire) du jeu de données considéré.

Les experts ont été construits grâce à un ensemble d'entrainement formé par la consommation d'électricité réalisée ainsi que quelques variables contextuelles observées pendant une période de temps précédent l'ensemble de prévisions. La taille et la nature de l'ensemble d'entrainement dépendent de l'expert. Les trois familles d'experts sont : les modèles paramétrique (15 experts), les modèles semi-paramétriques (8 experts), et un expert fonctionnel. Les experts sont spécialisés, c'est à dire qu'ils ne proposent pas des prévisions à tous les instants. Plus de précisions sur les experts considérés, leurs caractéristiques sont disponibles en Annexe B.

Le Tableau 2 apporte les performances de référence que nos algorithmes ont pour objectif de concurrencer ou de battre. On remarque que le mélange uniforme apporte déjà une nette amélioration de performance par rapport aux experts pris séparément. Ce qui souligne l'intérêt de les agréger.

Stratégies et oracles	RMSE (MW)
Meilleur expert	782 ± 10
Meilleure combinaison convexe	658 ± 9
Meilleure combinaison linéaire	625 ± 7
Mélange uniforme	724 ± 11
Meilleur expert composé $m = 13$	629
$m = 50$	534
$m = T - 1 = 15\,359$	223

TABLE 2 – Performances des oracles et de la stratégie de mélange de référence

5.2. Performances des algorithmes de mélange considérés

Les algorithmes de mélange considérés, introduits en Partie 3, peuvent sembler automatiques mais ne le sont en fait qu'en partie. Ils dépendent sensiblement de paramètres d'apprentissages qui doivent être calibrés ou connus à l'avance. Dans nos expériences, nous commençons par comparer leurs performances pour les meilleurs paramètres fixés choisis à postériori sur des grilles. Dans une deuxième étape, nous calibrons de manière automatique ces paramètres de façon à obtenir des méthodes de mélange en ligne totalement automatiques.

Les algorithmes dépendent de plus de la fonction de perte considérée. Durant le stage, on a étudié trois fonctions de pertes présentées en Annexe A. Les algorithmes sont apparus assez robuste à la fonction de perte considérée et leur performances n'en étaient que très peu altérées. Aussi, par soucis de concision, nous ne considérons dans ce rapport que la fonction de perte carrée, définie pour tout $x, y \in \mathbb{R}$ par $\ell(x, y) = (x - y)^2$.

Performances pour des valeurs fixées des paramètres

Pour commencer, on rapporte les performances des algorithmes de mélange considérés pour des valeurs constantes de paramètres. Le paramètre η des algorithme $\mathcal{E}_\eta$, $\mathcal{E}_\eta^{\text{grad}}$, $\mathcal{S}_\eta$, $\mathcal{S}_\eta^{\text{grad}}$ et $\mathcal{T}_\eta$ a été optimisé sur la grille

$$\Lambda = \left\{ m \cdot 10^k, \quad m \in \{1, \dots, 9\} \quad \text{and} \quad k \in \{-7, \dots, 1\} \right\}.$$

Algorithme de mélange	$\mathcal{E}_\eta$	$\mathcal{E}_\eta^{\text{grad}}$	$\mathcal{S}_\eta$	$\mathcal{S}_\eta^{\text{grad}}$	$\mathcal{F}_{\eta\alpha}$	$\mathcal{F}_{\eta\alpha}^{\text{grad}}$	$\mathcal{R}_\lambda$	$\mathcal{T}_\eta$
Paramètres optimaux	$1 \cdot 10^{-4}$	2	$6 \cdot 10^{1}$	$3 \cdot 10^{-2}$	$(1400, 0.05)$	$(1, 0.01)$	$1 \cdot 10^{3}$	2
RMSE (MW)	718 ± 12	629 ± 8	691 ± 10	631 ± 9	632 ± 11	599 ± 9	650 ± 9	676 ± 13

TABLE 3 – Performances des algorithmes de mélange pour les meilleurs choix de paramètres constants sur leurs grilles respectives.

Pour les algorithmes de type fixed-share $\mathcal{F}_{\eta\alpha}$ et $\mathcal{F}_{\eta\alpha}^{\text{grad}}$, le couple (α, η) a été optimisé sur la grille

$$\Lambda_{\mathcal{F}} = \left\{ \big(m{\cdot}10^k, \alpha\big),\ m \in \big\{1, \dots, 9\big\},\ k \in \big\{-5, \dots, 3\big\},\ \text{and}\ \alpha \in \big\{0, 0.005, 0.01, 0.05, 0.1, 0.2, 0.5, 1\big\} \right\}.$$

Le paramètre λ de l'algorithme de type Ridge $\mathcal{R}_\lambda$ a quand à lui été optimisé sur la grille

$$\Lambda_{\mathcal{R}} = \left\{ 10^i,\ i \in \{-6, \dots, 7\} \right\}.$$

Ces grilles n'ont pas été choisies au hasard, mais sont de l'ordre de grandeur des valeurs théoriques optimales des paramètres. Elles ont été ensuite ajustées en pratique pour avoir des extrémités suffisamment éloignées. Les performances sont rapportées Tableau 4.

On remarque que les versions de base des algorithmes $\mathcal{E}_\eta$, $\mathcal{S}_\eta$, $\mathcal{F}_{\eta\alpha}$ et $\mathcal{T}_\eta$ battent toute largement la performance du meilleur expert fixé, qui a un RMSE de 782 ± 10 MW. Ce n'est pas surprenant sachant que même le mélange uniforme atteignait de meilleurs résultats.

Les versions gradient $\mathcal{E}_\eta^{\text{grad}}$, $\mathcal{S}_\eta^{\text{grad}}$ et $\mathcal{F}_{\eta\alpha}^{\text{grad}}$ battent aussi facilement leur oracle concurrent : la meilleur combinaison convexe fixée, qui atteint un RMSE de 658 ± 9 MW. Leurs résultats sont même comparable au meilleur expert composé avec au plus 13 changements (RMSE = 629 MW). Nos algorithmes battent donc quelqu'un qui toutes les trois semaines connaitrait à l'avance le meilleur expert à venir et le choisirait.

On est cependant déçu par la performance de l'algorithme de type Ridge $\mathcal{R}_\lambda$ qui cherche à atteindre la performance de la meilleure combinaison linéaire (RMSE = 625 ± 7 MW), même si l'on ne dispose pour l'instant d'aucune preuve théorique. Sa performance reste cependant honorable comparée aux versions basiques des algorithmes précédents. L'utilité des combinaisons linéaires dans notre cas est cependant limitée. Les experts étant pour la plupart très bon, leur attribuer des poids négatifs n'a que peu de sens.

L'algorithme de type forêts aléatoire, qui prend en compte l'information contextuelle disponible avant d'effectuer le mélange atteint de meilleure performances qu' EWA basique. Les résultats restent toutefois décevants. Durant ce stage, j'ai essayé de faire des versions descente gradient de $\mathcal{R}_\lambda$ et $\mathcal{T}_\eta$, cela n'a cependant pour l'instant pas été fructueux. À la vue des excellents résultats des versions gradient des algorithmes précédent, pousser un peu cette voie peut-être une piste intéressante.

Performances pour des paramètres calibrés en ligne

On présente maintenant les résultats pratiques des trois premières familles d'algorithmes de mélange dans leur versions totalement séquentielles présentées en partie 3.2. Nous avons pas eu le temps d'adapter les algorithmes de type Ridge $\mathcal{R}_\lambda$ et de type forêts aléatoires $\mathcal{T}_\eta$.

Pour les algorithmes de type fixed-share, on a utilisé la grille fixée

$$\Lambda_\alpha = \left\{ 0\,,\ 0.005\,,\ 0.01\,,\ 0.05\,,\ 0.1\,,\ 0.2\,,\ 0.5\,,\ 1 \right\}$$

pour calibrer le paramètre α.

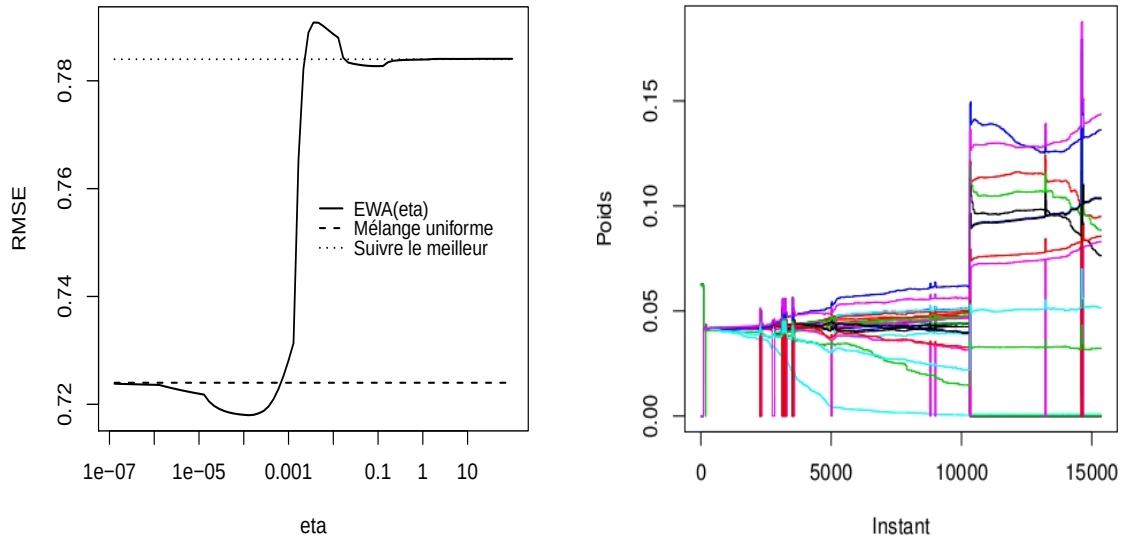


FIGURE 2 – [Figure gauche] Performance de l'algorithme $\mathcal{E}_\eta$ en fonction de la valeur de son paramètre d'apprentissage η. [Figure droite] Évolution des poids accordés à chaque expert par l'algorithme $\mathcal{E}_\eta$.

On remarque avec satisfaction Tableau 4 que les performances des algorithmes sont peu détériorées par la calibration automatique. Les RMSE ne valant que 1 à 2 pour cent de plus. La hausse de RMSE de $\mathcal{F}_{\eta\alpha}^{\text{grad}}$ s'explique par le fait qu'il n'atteignait en fait essentiellement que par hasard une si bonne performance (RMSE = 599 MW) pour le couple de paramètre $(\eta, \alpha) = (1, 0.01)$, et que le moindre changement de paramètres le ramène à un RMSE de l'ordre de 620 MW.

Algorithme de mélange	$\mathcal{E}_\eta$	$\mathcal{E}_\eta^{\text{grad}}$	$\mathcal{S}_\eta$	$\mathcal{S}_\eta^{\text{grad}}$	$\mathcal{F}_{\eta\alpha}$	$\mathcal{F}_{\eta\alpha}^{\text{grad}}$
RMSE (MW)	724 ± 11	637 ± 9	715 ± 12	635 ± 9	639 ± 11	623 ± 11

TABLE 4 – Performances des versions adaptatives des algorithmes de mélange.

6. Conclusion

Durant ce stage, j'ai donc étudié des nouvelles versions d'algorithmes de mélange : Ridge et les forêts aléatoires et j'ai comparé leurs résultats avec ceux des algorithmes déjà existants. Leurs performances, bien que n'égalant pas les anciennes méthodes, sont prometteuses.

J'ai profité des forêts aléatoires pour créer de nouveaux experts (Annexe G). Cela m'a permis d'explorer quelques améliorations possibles des forêts aléatoires, en considérant par exemple la notion fondamentale de proximité.

Enfin, n'oublions pas tous les petits problèmes théoriques qui m'ont suivi tout au long du stage : recherche de borne théorique pour les forêts aléatoires, y en a t-il une pour ridge ? De nombreuses pistes restent à explorer : peut-on expliquer la rupture de la figure 2 ? Peut-on créer des versions gradient de Ridge ou des forêts aléatoires ? À la vue des comportements différentes méthodes de mélange, que donnerait en pratique un méta-algorithme qui les mélangerait à nouveau entre elles ?

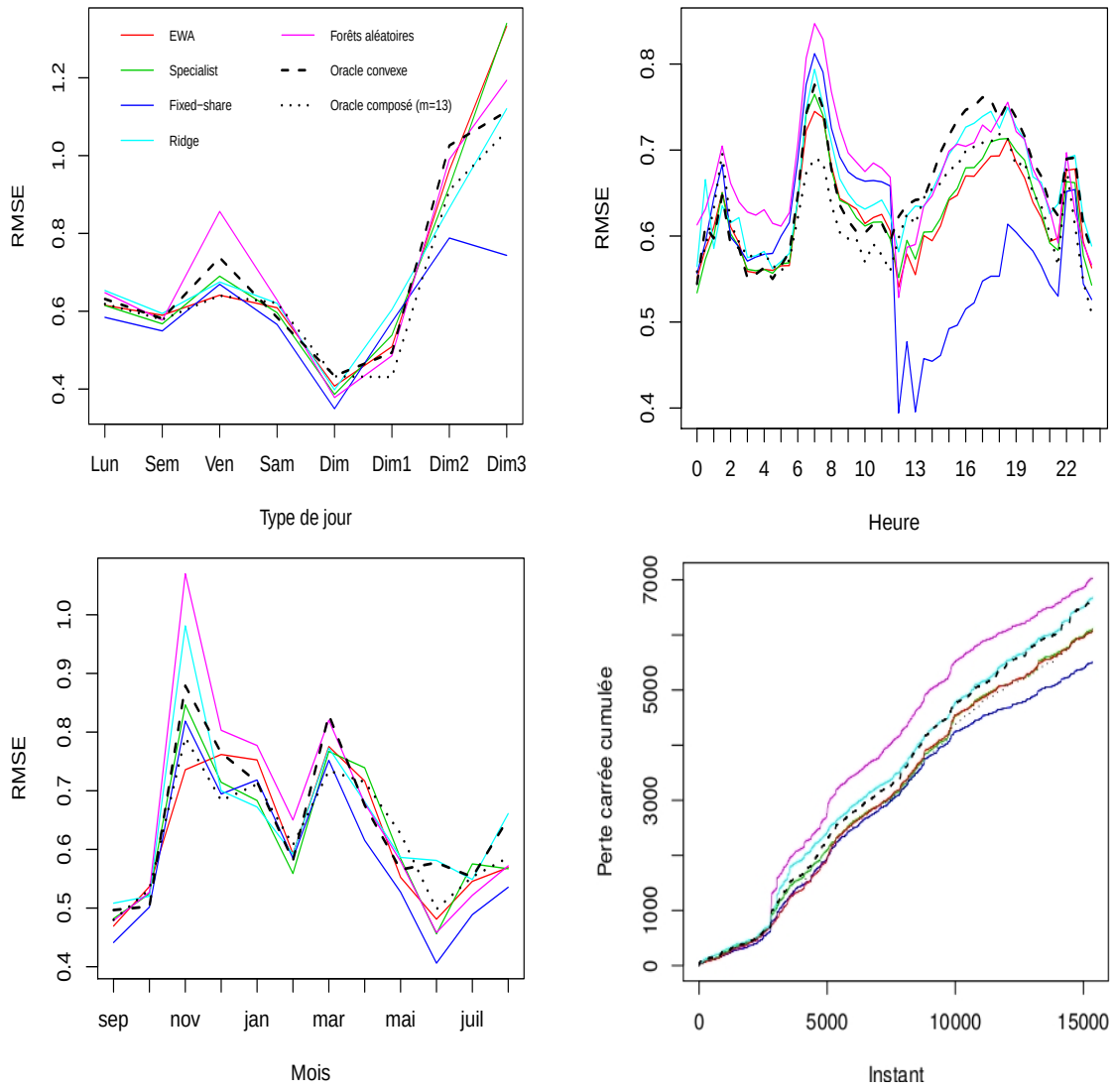


FIGURE 3 – Performance moyenne des différents algorithmes et de deux oracles selon le type de jour [haut gauche], selon l'heure de la journée [haut droite] et selon le mois de l'année [bas gauche]. Évolution de la perte carrée cumulée de chaque algorithme en fonction de l'instant t [bas droite]. Les trois premières familles d'algorithmes EWA, Specialist et Fixed-share ont été expérimentées dans leurs versions gradient.

Références

[BDL08] G. Biau, L. Devroye, and G. Lugosi. Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 9 :2015–2033, 2008.

[BLNZ95] R.H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16 :1190–1208, 1995.

[BM07] A. Blum and Y. Mansour. From external to internal regret. *Journal of Machine Learning Research*, 8 :1307–1324, 2007.

[Bre96] L. Breiman. Bagging predictors. *Machine Learning*, 24(2) :123–140, 1996.

[Bre01] L. Breiman. Random forests. *Machine Learning*, 45(1) :5–32, 2001.

[DGS11] M. Devaine, Y. Goude, and G. Stoltz. Forecasting the electricity consumption by aggregating specialized experts. 2011.

[Fis58] W.D. Fisher. On grouping for maximum homogeneity. *Journal of the American Statistical Association*, 53(284) :789–798, 1958.

[GGS11] P. Gaillard, Y. Goude, and G. Stoltz. A further look at the forecasting of the electricity consumption by aggregation of specialized experts. *Technical Report*, 2011.

[HK10] E. Hazan and S. Kale. Extracting certainty from uncertainty : Regret bounded by variation in costs. *Machine Learning*, 80(2) :165, 2010.

[HTF09] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning : Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009.

[LJ06] Y. Lin and Y. Jeon. Random forests and adaptive nearest neighbors. *Journal of American Statistical Association*, 101 :578–590, 2006.

[LW02] A. Liaw and M. Wiener. Classification and regression by `randomForest`. *R News*, 2/3 :18, 2002.

L'écriture de ce rapport a été grandement inspirée par le rapport technique [GGS11].

Annexes

Table des annexes

A. Fonctions de perte considérées

Dans nos expériences, on a utilisé trois fonctions de pertes différentes. Les méthodes de mélange conventionnelles placent des poids plus important sur les experts les plus précis, leurs définitions dépendent donc des pertes subies par les experts dans le passé. On a donc trois versions de chaque algorithme, qui dépend de la fonction de perte utilisée pour déterminer la qualité des experts. La performance de chaque version est alors caractérisée par son erreur moyenne ainsi que par une mesure de dispersion, dépendant de la fonction de perte sous la main.

- *La perte carrée* est définie pour tout $x, y \in \mathbb{R}_+$ par

$$\ell(x, y) = (x - y)^2 .$$

Dans ce cas, plutôt que l'erreur moyenne $\overline{\mathrm{ERR}}_T(\mathcal{A})$ on utilisera la racine de l'erreur moyenne de la perte carrée

$$\mathrm{RMSE}_T(\mathcal{A}) = \sqrt{\overline{\mathrm{ERR}}_T(\mathcal{A})} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(\widehat{y}_t - y\right)^2}$$

pour reporter la qualité de l'algorithme de mélange $\mathcal{A}$ selon la perte carrée. On peut obtenir la mesure de dispersion correspondante à l'aide de la méthode delta et du lemme de Slutsky (cf. Appendix A),

$$\widehat{s}_T = \sqrt{\frac{\frac{1}{T}\sum_{t=1}^{T}\left(\left(\widehat{y}_t - y_t\right)^2 - \frac{1}{T}\sum_{t'=1}^{T}\left(\widehat{y}_{t'} - y_{t'}\right)^2\right)^2}{4\,\frac{1}{T}\sum_{t=1}^{T}\left(\widehat{y}_t - y_t\right)^2}} .$$

On rapporte l'erreur type à 95%, $1.96\ \widehat{s}_T/\sqrt{T}$ dans les tableaux.

- *La perte absolue* est définie pour tout $x, y \in \mathbb{R}_+$ par

$$\ell(x, y) = |x - y| .$$

Pour cette perte, comme on utilise l'erreur moyenne

$$\mathrm{MAE}_T(\mathcal{A}) = \frac{1}{T}\sum_{t=1}^{T}\left|\widehat{y}_t - y_t\right| ,$$

pour quantifier la qualité d'un algorithme de mélange $\mathcal{A}$, la mesure de dispersion correspondante est définie comme l'erreur type de l'échantillon,

$$\widehat{\sigma}_T = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(\left|\widehat{y}_t - y_t\right| - \frac{1}{T}\sum_{t'=1}^{T}\left|\widehat{y}_{t'} - y_{t'}\right|\right)^2}\,.$$

On rapporte l'erreur type à 95%, $1.96\ \widehat{\sigma}_T/\sqrt{T}$ dans les tableaux.

- *Le pourcentage d'erreur absolue* est défini pour tout $x, y \in \mathbb{R}_+$ par

$$\ell(x, y) = \frac{|x - y|}{y}\,.$$

On utilise

$$\textsc{mape}_T(\mathcal{A}) = \frac{1}{T}\sum_{t=1}^{T}\frac{\left|\widehat{y}_t - y_t\right|}{y_t}$$

pour mesurer la qualité de l'algorithme $\mathcal{A}$ et l'erreur type de l'échantillon,

$$\widehat{\sigma}_T = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(\frac{\left|\widehat{y}_t - y_t\right|}{y_t} - \frac{1}{T}\sum_{t'=1}^{T}\frac{\left|\widehat{y}_{t'} - y_{t'}\right|}{y_{t'}}\right)^2}\,,$$

pour quantifier la dispersion. On rapporte alors l'erreur type à 95%, $1.96\ \widehat{\sigma}_T/\sqrt{T}$ dans les tableaux.

Au cours de ce stage, pour quantifier la qualité d'un algorithme de mélange $\mathcal{A}$, nous avons également considéré la corrélation entre les prévisions et les observations. Elle est définie pour un algorithme de mélange $\mathcal{A}$ proposant les prévisions $\widehat{y}_1, \ldots, \widehat{y}_T$ comme

$$\textsc{corr}_T(\mathcal{A}) = \frac{\displaystyle\sum_{t=1}^{T}\Big(\widehat{y}_t - \overline{\widehat{y}}\Big)\big(y_t - \overline{y}\big)}{\sqrt{\displaystyle\sum_{t=1}^{T}\Big(\widehat{y}_t - \overline{\widehat{y}}\Big)^2\sum_{t=1}^{T}\big(y_t - \overline{y}\big)^2}}\,,$$

où $\overline{y} = \dfrac{1}{T}\displaystyle\sum_{t=1}^{T} y_t$ and $\overline{\widehat{y}} = \dfrac{1}{T}\displaystyle\sum_{t=1}^{T}\widehat{y}_t$. Cependant, la quasi totalité des algorithmes de mélange, obtient une corrélation de 99.8%, je ne la rapporte donc pas dans ce rapport.

Mesure de dispersion de la perte carrée

On justifie maintenant la mesure de dispersion considérée pour la perte carrée. Soit X une variable aléatoire de moyenne $\mu = \mathbb{E}\big[X\big]$ et d'écart type $\sigma = \sqrt{\mathrm{Var}\big(X\big)}$. Soit $(X_1, \ldots, X_T)$ une suite de variables aléatoires indépendantes et identiquement distribuées selon la loi de X. On cherche à donner une mesure de dispersion pour $\sqrt{\mathbb{E}\big[X\big]}$. D'après le théorème central limite, on a

$$\sqrt{T}\left(\frac{\overline{X}_T - \mu}{\sigma}\right) \rightsquigarrow \mathcal{N}\big(0, 1\big)\,,$$

quand $T \to \infty$, où l'on note $\overline{X}_T$ la moyenne empirique, $\overline{X}_T = \dfrac{1}{T}\displaystyle\sum_{t=1}^{T} X_t$.

La méthode delta donne alors,

$$\sqrt{T}\left(\sqrt{\overline{X}_T} - \sqrt{\mu}\right) \rightsquigarrow \mathcal{N}\left(0, \left(1/2\sqrt{\mu}\right)^2 \sigma^2\right),$$

ce qui entraine,

$$\sqrt{T}\frac{\left(\sqrt{\overline{X}_T} - \sqrt{\mu}\right)}{\sigma/2\sqrt{\mu}} \rightsquigarrow \mathcal{N}\big(0,1\big)\,.$$

On obtient, de plus par, la méthode des moments,

$$\widehat{s}_T \cdot \frac{2\sqrt{\mu}}{\sigma} \xrightarrow{\mathbb{P}} 1\,,$$

où $\widehat{s}_T$ est définit par

$$\widehat{s}_T = \sqrt{\frac{\frac{1}{T}\sum_{t=1}^{T}\Big(X_t - \overline{X}_T\Big)^2}{4\,\overline{X}_T}}\,.$$

Par conséquent, le lemme de Slutsky entraine

$$\sqrt{T}\left(\frac{\sqrt{\overline{X}_T} - \sqrt{\mu}}{\widehat{s}_T}\right) \rightsquigarrow \mathcal{N}(0,1)\ .$$

On obtient ainsi un intervalle de confiance asymptotique au niveau de confiance $(1-\alpha)$ pour $\sqrt{\mathbb{E}\big[X\big]}$,

$$I = \left[\sqrt{\overline{X}_T} \pm \frac{z_\alpha}{\sqrt{T}}\widehat{s}_T\right]\,,$$

où z_α correspond au α-quantile de la loi normale.

B. Précisions sur les experts considérés

Les experts considérés ont été formé à partir de trois modèles statistiques principaux : paramétrique, semi-paramétrique et non-paramétrique. Nous représentons en figure 4 leurs performances ainsi que leurs pourcentages d'activations.

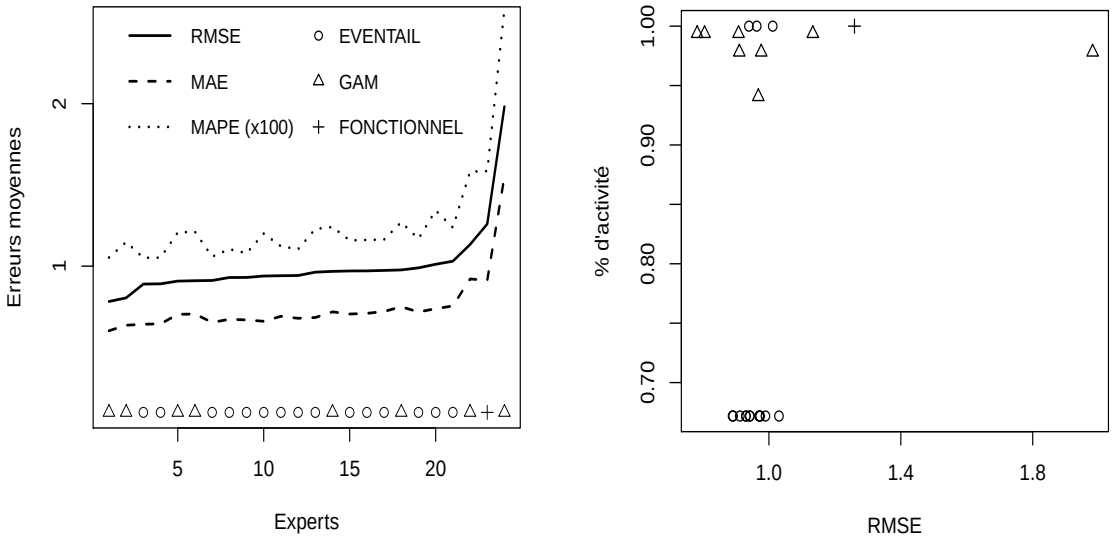


FIGURE 4 – [Figure gauche] Erreurs moyennes et RMSEs des experts (axe-y) triés selon leur RMSEs (axe-x). [Figure droite] Frequences d'activité des experts (axe-y) selon leur RMSEs (axe-x). Les experts Eventail sont indexés par ◯, les experts GAM par △, et l'expert fonctionnel par +.

C. Calcul de la performance du meilleur expert composé

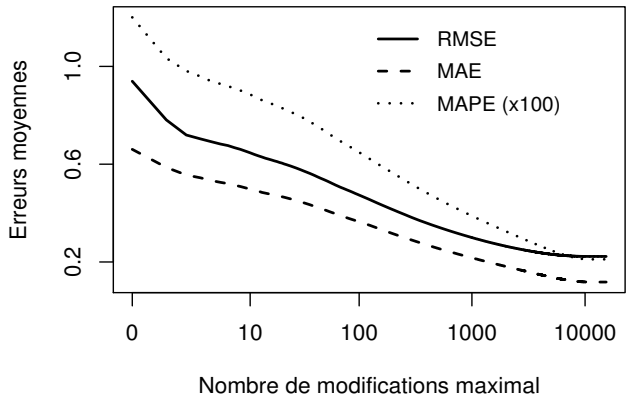


FIGURE 5 – Evolution de l'erreur moyenne du meilleur expert composé $\mathcal{O}_\delta^m$ (y-axis) en fonction du nombre de modifications m (x-axis).

On détaille ici comment la performance du meilleur expert composé avec au plus m modifications, $\mathcal{O}_\delta^m$, peut être obtenue par programmation dynamique. L'idée est de mettre à jour à chaque instant t, pour chaque expert actif i et chaque nombre de modifications m, la perte cumulée $L(m,i,t)$ soufferte jusqu'à l'instant t (inclus) par le meilleur expert composé avec au plus m modifications terminant avec l'expert i. La perte finale encourue par $\mathcal{O}_\delta^m$ est alors donnée par

$$\overline{\mathrm{ERR}}_T(\mathcal{O}_\delta^m) = \frac{1}{T} \min_{m' \leqslant m} \min_{j \in E_T} L(m', j, T)\,.$$

Algorithme 6 Meilleur expert composé avec au plus m modifications, $\mathcal{O}_\delta^m$.

`Entrée:` observations (y_t), conseils d'experts (f_{it}), ensembles d'experts actifs (E_t), fonction de perte ℓ
`Initialisation:` $L(0,i,0) \leftarrow 0$ `pour tout` $i \in \{1,\dots,N\}$

`pour` les instants t `de` 1 `à` T `faire`
  `pour tout` nombre de modifications $m \in \{0,\dots,t-1\}$ `et tout` expert $i \in E_t$ `mettre à jour`

$$L(m,i,t) \leftarrow \min\left\{ L(m,i,t-1),\ \min_{j \in E_{t-1} \setminus \{i\}} L(m-1,j,t-1) \right\} + \ell\big(f_{it}, y_t\big)$$

  `fin pour`
`fin pour`
`retourner` $\dfrac{1}{T} \min_{m' \leqslant m} \min_{j \in E_T} L(m', j, T)$

D. Algorithmes de mélange EWA, Specialist et Fixed-share.

Dans cette annexe, nous présentons trois algorithmes de mélange détaillés dans [DGS11]. Leurs regrets (qui comparent leurs performances avec celle d'un oracle de référence comme celle du meilleur expert fixe) possèdent des bornes déterministes et valables à chaque pas de temps.

En remarquant que les fonctions de pertes considérées sont convexes en leur premier argument, on peut dériver une version gradient de chacun de ces algorithmes, qui peuvent s'interpréter comme des descentes gradient. Ces versions gradient sont en général plus performantes que les version de base et atteigne des performances de l'ordre de la meilleure combinaison convexe d'expert quand les version de bases ne se comparent qu'au meilleur expert ou à la meilleur séquence d'expert avec peut de modification.

Les algorithmes dépendent tous d'un ou deux paramètres d'apprentissages qui s'assimilent au fameux compromis biais-variance en apprentissage. Leur calibration joue un rôle essentiel en pratique.

EWA

L'algorithme exponentied weighed average (EWA, Algorithme 7) consiste à donner à chaque expert un poids exponentiel en la perte cumulée qu'il a subit jusqu'alors.

Algorithme 7 Algorithme de mélange EWA $\mathcal{E}_\eta$.

`Entrée:` paramètre d'apprentissage $\eta > 0$
`Initialisation:` w_1 est le vecteur convexe uniforme, $w_{i1} = 1/N$ pour $i = 1, \dots, N$

`pour` les instants t `de` 1 `à` T `faire`
  `prévoir` $\widehat{y}_t = \frac{1}{\sum_{i \in E_t} w_{it}} \sum_{j \in E_t} w_{jt} f_{jt}$
  `observer` y_t
  `pour` les experts i `de` 1 `à` T `mettre à jour`

$$w_{it+1} = \begin{cases} w_{it}\, e^{\eta\left(\ell_t\left(\frac{w_t}{\sum_{j \in E_t} w_{jt}}\right) - \ell_t(\delta_i)\right)} & \text{si } i \in E_t \quad \text{// voir annotation } a \\ w_{it} & \text{si } i \notin E_t \end{cases}$$

  `fin pour`
`fin pour`

a. $\mathcal{E}_\eta^{\text{grad}}$ correspond au remplacement de ℓ_s par la pseudo-perte $\widehat{\ell}_s$ definie pour tout $\boldsymbol{u} \in \mathbb{R}^{\mathbb{N}}$ par $\widehat{\ell}_s(\boldsymbol{u}) = \nabla(\boldsymbol{u}_s) \cdot \boldsymbol{u}$ où $\nabla(\boldsymbol{u}_s)$ est un sous-gradient de la fonction de perte ℓ_s.

Specialist

L'algorithme de mélange specialist (Algorithme 8) est très semblable au précédent et présente des bornes théoriques et des performances similaires. Seule la différence de pondération entre les experts actifs et les experts inactifs à l'instant précédent est modifiée.

Fixed-share

L'algorithme de mélange fixed-share (Algorithme 9) est motivé par l'oracle du meilleur expert composé. Il s'adapte plus facilement aux changements de régime et atteint ainsi les meilleures performances. On remarque que son vecteur de poids est mis à jour à chaque instant en deux étape. La première suit l'idée d'EWA, où les experts sont repondérés selon leur performance passée de manière exponentielle. La seconde redistribue les poids sur les experts actifs en s'assurant que chacun s'attribue un poids minimal ; c'est la clef qui permet d'être compétitif face au meilleur expert composé ou à la meilleure combinaison convexe composée.

Algorithme 8 Algorithme de mélange Specialist $\mathcal{S}_\eta$.

`Entrée:` paramètre d'apprentissage $\eta > 0$
`Initialisation:` w_1 est le vecteur convexe uniforme, $w_{i1} = 1/N$ pour $i = 1, \ldots, N$

`pour` les instants t `de` 1 `à` T `faire`
  `prévoir` $\widehat{y}_t = \frac{1}{\sum_{i \in E_t} w_{it}} \sum_{j \in E_t} w_{jt} f_{jt}$
  `observer` y_t
  `pour` les experts i `de` 1 `à` T `mettre à jour`

$$w_{it+1} = \begin{cases} w_{it} e^{-\eta \ell_t(\delta_i)} \frac{\sum_{j \in E_t} w_{jt}}{\sum_{k \in E_t} w_{kt} e^{-\eta \ell_t(\delta_k)}} & \text{si } i \in E_t \quad \text{// voir annotation } a \\ w_{it} & \text{si } i \notin E_t \end{cases}$$

  `fin pour`
`fin pour`

a. $\mathcal{S}_\eta^{\text{grad}}$ correspond au remplacement de ℓ_s par la pseudo-perte $\widehat{\ell}_s$ definie pour tout $\boldsymbol{u} \in \mathbb{R}^{\mathbb{N}}$ par $\widehat{\ell}_s(\boldsymbol{u}) = \nabla(\boldsymbol{u}_s) \cdot \boldsymbol{u}$ où $\nabla(\boldsymbol{u}_s)$ est un sous-gradient de la fonction de perte ℓ_s.

Algorithme 9 Algorithme de mélange fixed-share $\mathcal{F}_{\eta\alpha}$.

`Entrée:` paramètre d'apprentissage $\eta > 0$ et de mélange $0 \leqslant \alpha \leqslant 1$
`Initialisation:` w_1 est la vecteur convexe uniforme, $w_{i1} = 1/N$ pour $i = 1, \ldots, N$

`pour` les instants t `de` 1 `à` T `faire`
  `prévoir` $\widehat{y}_t = \frac{1}{\sum_{i \in E_t} w_{it}} \sum_{j \in E_t} w_{jt} f_{jt}$
  `observer` y_t
  `pour` les experts i `de` 1 `à` N `mettre à jour`

$$v_{it} = \begin{cases} w_{it}\, e^{-\eta \ell_t(\delta_i)} & \text{si } i \in E_t \\ \text{non défini} & \text{si } i \notin E_t \end{cases} \qquad \text{// voir annotation } a$$

$$w_{it+1} = \begin{cases} \frac{1}{|E_{t+1}|} \sum_{i \in E_t \setminus E_{t+1}} v_{i,t} + \frac{\alpha}{|E_{t+1}|} \sum_{i \in E_t \cap E_{t+1}} v_{i,t} + (1-\alpha)\, \mathbb{1}_{\{j \in E_t \cap E_{t+1}\}}\, v_{j,t} & \text{si } i \in E_{t+1} \\ 0 & \text{si } i \notin E_{t+1} \end{cases}$$

  `fin pour`
`fin pour`

a. $\mathcal{F}_{\eta\alpha}^{\text{grad}}$ correspond au remplacement de ℓ_s par la pseudo-perte $\widehat{\ell}_s$ definie pour tout $\boldsymbol{u} \in \mathbb{R}^{\mathbb{N}}$ par $\widehat{\ell}_s(\boldsymbol{u}) = \nabla(\boldsymbol{u}_s) \cdot \boldsymbol{u}$ où $\nabla(\boldsymbol{u}_s)$ est un sous-gradient de la fonction de perte ℓ_s.

E. Une borne améliorée dans le cadre d'experts spécialisés

Dans cette annexe, on propose une borne améliorée du regret dans le cas d'experts spécialisés. La preuve originale a été proposée par Blum et Mansour [BM07, Partie 6]. Elle est basée sur un algorithme EWA calculé avec des pertes pénalisées (voir Algorithme **??**). On utilise la notation $\ell_{jt} = \ell(f_{jt}, y_t)$ pour dénoter la perte subie par l'expert j à l'instant $t \geqslant 1$. On assume de plus que la fonction de perte ℓ est convexe en son premier argument et a valeurs dans l'intervalle $[0,1]$. À cause des inégalités (5) and (6), il est essentiel que ℓ prenne ses valeurs dans $[0,1]$. Si l'image de ℓ est un intervalle différent $[m,M]$, les valeurs de m et M doivent être connues à l'avance afin de pouvoir considérer la fonction de perte normalisée $(\ell - m)/(M-m)$ au lieu de ℓ.

Algorithme 10 L'algorithme de mélange EWA amélioré de Blum-Mansour pour les experts spécialisés.

`Initialisation:` vecteur de poids uniforme $\boldsymbol{p}_1 \in \mathcal{X}$

`pour` les instants t `de` 1 `à` T `faire`
  `prévoir` $\widehat{y}_t \leftarrow \sum_{j \in E_t} p_{jt} f_{jt}$
  `subir la perte` $\widehat{\ell}_t \leftarrow \ell(\widehat{y}_t, y_t)$
  `Pour` l'expert i `de` 1 `à` N `mettre à jour`
    $w_{it} \leftarrow e^{-\eta_i \sum_{s=1}^{t} (\ell_{is} - e^{-\eta_i \widehat{\ell}_s}) \mathbb{1}_{\{i \in E_s\}}}$
    $p_{it+1} \leftarrow \dfrac{w_{it}\,(1 - e^{-\eta_i})\,\mathbb{1}_{\{i \in E_{t+1}\}}}{\sum_{j \in E_{t+1}} w_{jt}\,(1 - e^{-\eta_j})\,\mathbb{1}_{\{j \in E_{t+1}\}}}$
  `fin pour`
`fin pour`

Théorème 2. *Si la fonction de perte ℓ est convexe en son premier argument et a valeurs dans $[0,1]$, le regret de l'algorithme de mélange de Blum et Mansour agrégation décrit ci-dessus peut être borné par*

$$\sum_{t=1}^{T} \left(\widehat{\ell}_t - \ell_{it}\right) \mathbb{1}_{\{i \in E_t\}} \leqslant \frac{\ln N}{\eta_i} + \eta_i \sum_{t=1}^{T} \mathbb{1}_{\{i \in E_t\}}\,.$$

Corollaire 3. *Le choix de $\eta_i = \sqrt{\dfrac{\ln N}{\sum_{t=1}^{T} \mathbb{1}_{\{i \in E_t\}}}}$ dan le Théorème 2 mène à la borne*

$$\forall i \in \{1, \ldots, N\}\,, \qquad \sum_{t=1}^{T} \left(\widehat{\ell}_t - \ell_{it}\right) \mathbb{1}_{\{i \in E_t\}} \leqslant 2\sqrt{\ln N \sum_{t=1}^{T} \mathbb{1}_{\{i \in E_t\}}}\,.$$

Démonstration. L'idée principale de la preuve est de contrôler $w_{1t} + \cdots + w_{Nt}$; en particulier, on va montrer par récurrence que pour tout $t \geqslant 0$,

$$\sum_{i=1}^{N} w_{it} \leqslant N\,. \tag{2}$$

Ce sera suffisant pour conclure. En effet, on aura en particulier que $w_{iT} \leqslant N$ pour tout expert i ; ou, écrit différemment,

$$-\eta_i \sum_{t=1}^{T} \left(\ell_{it} - e^{-\eta_i \widehat{\ell}_t}\right) \mathbb{1}_{\{i \in E_t\}} \leqslant \ln N\,.$$

Comme $e^{-x} \geqslant 1-x$, pour tout $x \in \mathbb{R}$, on a

$$\sum_{t=1}^{T} \Big((1-\eta_i)\,\widehat{\ell}_t - \ell_{it} \Big)\, \mathbb{1}_{\{i \in E_t\}} \leqslant \frac{\ln N}{\eta_i}\,,$$

$$\sum_{t=1}^{T} \Big(\widehat{\ell}_t - \ell_{it} \Big)\, \mathbb{1}_{\{i \in E_t\}} \leqslant \frac{\ln N}{\eta_i} + \eta_i \sum_{t=1}^{T} \widehat{\ell}_t\, \mathbb{1}_{\{i \in E_t\}}\,,$$

qui mène à la borne proposée en bornant $\widehat{\ell}_t$ par 1.

Il suffit donc de prouver (2), ce que l'on fait par récurrence sur t. Par définition des poids, on a

$$\begin{aligned} \sum_{i=1}^{N} w_{it} &= \sum_{i=1}^{N} w_{it-1}\, \exp\Big(-\eta_i \Big(\ell_{it} - e^{-\eta_i} \widehat{\ell}_t \Big) \mathbb{1}_{\{i \in E_t\}} \Big) && (3) \\ &= \sum_{i=1}^{N} w_{it-1}\, \exp\Big(-\eta_i \ell_{it} \mathbb{1}_{\{i \in E_t\}} \Big) \exp\Big(\eta_i e^{-\eta_i} \widehat{\ell}_t \mathbb{1}_{\{i \in E_t\}} \Big)\,. && (4) \end{aligned}$$

Par convexité de $x \mapsto e^{\eta x}$ dans $[0,1]$, pour tout $\eta \in \mathbb{R}$, on a pour tout $x \in [0,1]$,

$$e^{\eta x} \leqslant (1-x)\, e^0 + x e^{\eta} = 1 - x\,\big(1 - e^{\eta}\big)\,.$$

On en déduit les deux inégalités suivantes, valides pour tout $x \in [0,1]$ et $\eta > 0$,

$$\begin{aligned} e^{-\eta x} &\leqslant 1 - \big(1 - e^{-\eta}\big)\, x\,, && (5) \\ e^{\eta x} &\leqslant 1 - \big(1 - e^{-\eta}\big)\, e^{\eta} x\,. && (6) \end{aligned}$$

Par conséquent, (4) conduit à

$$\begin{aligned} \sum_{i=1}^{N} w_{it} &\leqslant \sum_{i=1}^{N} w_{it-1} \Big(1 - \big(1 - e^{-\eta_i}\big) \ell_{it} \mathbb{1}_{\{i \in E_t\}} \Big) \Big(1 + \big(1 - e^{-\eta_i}\big) e^{\eta_i}\, e^{-\eta_i} \widehat{\ell}_t \mathbb{1}_{\{i \in E_t\}} \Big) \\ &\leqslant \sum_{i=1}^{N} w_{it-1} \Big(1 + \big(1 - e^{-\eta_i}\big) \big(\widehat{\ell}_t - \ell_{it} \big) \mathbb{1}_{\{i \in E_t\}} \Big)\,. \end{aligned}$$

Par convexité de la fonction de perte, on obtient $\widehat{\ell}_t \leqslant \sum_{j=1}^{N} p_{jt} \ell_{jt}$, dont on déduit

$$\begin{aligned} \sum_{i=1}^{N} w_{it} - \sum_{i=1}^{N} w_{it-1} &\leqslant \sum_{i=1}^{N} \underbrace{w_{it-1} \big(1 - e^{-\eta_i}\big) \mathbb{1}_{\{i \in E_t\}}}_{p_{it} Z_t} \Big(\widehat{\ell}_t - \ell_{it} \Big) \\ &\leqslant \sum_{i=1}^{N} p_{it} Z_t \left(\sum_{j=1}^{N} p_{jt} \ell_{jt} - \ell_{it} \right) \\ &= Z_t \sum_{i=1}^{N} p_{it} \sum_{j=1}^{N} p_{jt} \ell_{jt} - Z_t \sum_{i=1}^{N} p_{it} \ell_{it} = 0\,, \end{aligned}$$

où $Z_t = \sum_{j \in E_t} w_{jt-1} \left(1 - e^{-\eta_j}\right) \mathbb{1}_{\{j \in E_t\}}$ est le facteur de normalisation dans la définition de $\boldsymbol{p}_t$.

Tout ceci mène finalement à (2) et conclut la preuve. □

F. Preuve de la borne sur le mélange par forêts aléatoires

Théorème. *Sous les hypothèses de la partie 4.2, le regret de l'Algorithme 5, calculé avec la suite de paramètres d'apprentissage (η_t), peut être borné par*

$$\sum_{t=1}^{T} \mathbb{E}_t\Big[\widehat{L}_t\Big] - \min_{1\leqslant j\leqslant N} \mathbb{E}_t[L_{jt}] \leqslant N^2 \left(\sum_{t=1}^{T} \sigma_t + \frac{1}{N\eta_t}\right).$$

Démonstration. Soit $t \geqslant 1$. On définit

$$r_t = \mathbb{E}_t\Big[\widehat{L}_t\Big] - \mathbb{E}_t[L_{j^*t}]\,,$$

où $j^* \in \underset{1\leqslant j\leqslant N}{\arg\min}\, \mathbb{E}_t[L_{jt}]$. On cherche à majorer $r_t \leqslant 3N^2\sigma_t/\sqrt{2\pi}$.

Par convexité de la fonction de perte ℓ en sont premier argument et par indépendance conditionnelle de (L_{jt}) et $(\widehat{\ell}_{it})$ sachant $\mathcal{F}_t^{\text{context}}$, on a

$$\begin{aligned}
r_t &\leqslant \mathbb{E}_t\left[\sum_{j=1}^{N} \frac{e^{-\eta_t\widehat{\ell}_{jt}}}{\sum_{i=1}^{N} e^{-\eta_t\widehat{\ell}_{it}}} L_{jt}\right] - \mathbb{E}_t[L_{j^*t}] && \text{// par convexité de } \ell \\
&= \mathbb{E}_t\left[\sum_{j=1}^{N} \frac{e^{-\eta_t\widehat{\ell}_{jt}}}{\sum_{i=1}^{N} e^{-\eta_t\widehat{\ell}_{it}}} \mathbb{E}_t[L_{jt}] - \mathbb{E}_t[L_{j^*t}]\right] && \text{// par indépendance de } (L_{jt}) \text{ et } \left(\widehat{\ell}_{it}\right) \text{ sachant } \mathcal{F}_t^{\text{context}} \\
&= \mathbb{E}_t\left[\sum_{j=1}^{N} \frac{\mathbb{E}_t[L_{jt}] - \mathbb{E}_t[L_{j^*t}]}{1 + \sum_{i\neq j} e^{-\eta_t\left(\widehat{\ell}_{it} - \widehat{\ell}_{jt}\right)}}\right] \\
&\leqslant \mathbb{E}_t\left[\sum_{j=1}^{N} \frac{\mathbb{E}_t[L_{jt}] - \mathbb{E}_t[L_{j^*t}]}{1 + e^{\eta_t\left(\widehat{\ell}_{jt} - \widehat{\ell}_{j^*t}\right)}}\right] && \text{// on minore le dénominateur}
\end{aligned}$$

On utilise maintenant la décomposition de $\widehat{\ell}_{jt}$ et le fait que, par définition, $\mathbb{E}_t[L_{jt}] - \mathbb{E}_t[L_{j^*t}] \geqslant 0$, pour obtenir la borne

$$r_t = \mathbb{E}_t\left[\sum_{j=1}^{N} \frac{\mathbb{E}_t[L_{jt}] - \mathbb{E}_t[L_{j^*t}]}{1 + e^{\eta_t\left(\mathbb{E}_t[L_{jt}] - \mathbb{E}_t\left[L_{j^*t}\right] + \varepsilon_{jt} - \varepsilon_{j*t}\right)}}\right] \leqslant N\ \mathbb{E}_t\left[\max_{x\in\mathbb{R}_+} \frac{x}{1 + e^{\eta_t(x - \varepsilon_t'')}}\right],$$

où $\varepsilon_t'' = \sum_{j=1}^{N} |\varepsilon_{jt}|$. Comme $\eta_t \geqslant 0$, la fonction

$$x \longmapsto \frac{x}{1 + e^{\eta_t(x-\varepsilon_t'')}}$$

est bornée sur $\mathbb{R}_+$. Son maximum est donné par la fonction W de Lambert (ou fonction Omega), qui est l'application inverse de $x \in \mathbb{R}_+ \mapsto xe^x$. Ainsi,

$$r_t \leqslant N\ \mathbb{E}_t\left[\frac{W\Big(e^{-1+\eta_t\varepsilon_t''}\Big)}{\eta_t}\right].$$

Maintenant, W est croissante et satisfait les deux inégalités suivantes,

$$\begin{aligned}
\forall x \geqslant 1, \qquad & W(e^x) \leqslant x\,, \\
\forall x \leqslant 1, \qquad & W(e^x) \leqslant W(e) = 1\,,
\end{aligned}$$

ce qui conduit à

$$\begin{aligned}
\mathbb{E}_t\Big[W\Big(e^{-1+\eta_t\varepsilon_t''}\Big)\Big] &\leqslant \mathbb{E}_t\Big[W\Big(e^{-1+\eta_t\varepsilon_t''}\Big)\,\mathbb{1}_{\{\varepsilon_t''\geqslant 2/\eta_t\}} + W\Big(e^{-1+\eta_t\varepsilon_t''}\Big)\,\mathbb{1}_{\{\varepsilon_t''<2/\eta_t\}}\Big] \\
&\leqslant \mathbb{E}_t\Big[\big(-1+\eta_t\varepsilon_t''\big)\,\mathbb{1}_{\{\varepsilon_t''\geqslant 2/\eta_t\}} + \mathbb{1}_{\{\varepsilon_t''<2/\eta_t\}}\Big] \\
&\leqslant \mathbb{E}_t\Big[\eta_t\,\varepsilon_t''\,\mathbb{1}_{\{\varepsilon_t''\geqslant 2/\eta_t\}} + 1\Big] \\
&\leqslant \eta_t\,\mathbb{E}_t\big[\varepsilon_t''\big] + 1\,. \qquad \text{// comme } \varepsilon_t'' \geqslant 0
\end{aligned}$$

Ce qui mène à la borne

$$r_t \leqslant N\,\left(\mathbb{E}_t\big[\varepsilon_t''\big] + \frac{1}{\eta_t}\right) = N\,\left(N\,\mathbb{E}_t\Big[|\varepsilon|\Big] + \frac{1}{\eta_t}\right)\,,$$

où $\varepsilon \sim \mathcal{N}\big(0,\sigma_t^2\big)$ et où l'égalité est vérifiée par la définition de ε_t''. On utilise aussi le fait que par l'inégalité de Cauchy-Schwarz, $\mathbb{E}_t[|\varepsilon|] \leqslant \sqrt{\mathbb{E}_t[\varepsilon^2]} = \sigma_t$. Finalement, $r_t \leqslant N\,(N\sigma_t + 1/\eta_t)$; on conclut enfin la preuve en sommant cette inégalité selon t. □

G. Construction de nouveaux experts par les forêts aléatoires

On conserve les notations de la partie 4 et on considère ici les forêts aléatoires comme un moyen d'estimer $\mathbb{E}_t[Y_t]$ pour construire de nouveaux experts (F_{jt}), avec $j > N$ et $1 \leqslant t \leqslant T$ que l'on pourrait plus tard ajouter à nos mélanges.

L'ensemble d'entrainement consiste en les paires entrée–sortie (X_t, Y_t) du 1er septembre 2002 au 31 Août 2007. Les variables contextuelles considérées sont les variables calendaires (type de jour, saisonnalité, etc.), des variables météo (température, nébulosité, vent, etc.), et des variables de séries temporelles (consommation de la journée précédente).

On traite indépendamment chaque consommation demi-horaire. On considère donc 48 séries temporelles séparées.

Le prédicteur par forêts aléatoires basique est noté RF_0. Ses performances sur l'ensemble de prévision, c'est à dire du 1er september 2007 au 31 août 2008, sont décrites en Tableau 6. Ce prédicteur reste cependant un peu trop générique et requiert quelque améliorations conduite par nos données pour atteindre de meilleur performances. On construit ainsi trois nouveaux prédicteurs RF_1–RF_3, présentés ci-après dans un ordre croissant de sophistication. Leurs caractéristiques sont résumées dans le Tableau 5.

Predicteur	Propriétés de bases	Ensemble d'entrainement en ligne	Modèle linéaire	Correction du biais
RF_0	×			
RF_1	×	×		
RF_2	×	×	×	
RF_3	×	×	×	×

Table 5 – Propriétés des prédicteurs de type forêts aléatoires.

- *Ensemble d'apprentissage en ligne :* Cette amélioration, que possèdent les prédicteurs RF_1–RF_3, consiste à aussi prendre en compte les données de l'ensemble de prévision disponibles jusqu'à l'instant t. Elle est motivée par les mauvais résultats atteint par RF_0 dans les zones inexplorées, comme la fin du mois de décembre 2007, quand la consommation d'énergie a été plus forte que jamais. Plus formellement, RF_1 prévoit la consommation d'énergie à l'instant t par

$$\widehat{Y}_t = \frac{1}{\sum_{s \in A_t} \mathrm{prox}(t,s)} \sum_{s \in A_t} \mathrm{prox}(t,s)\, Y_s \,,$$

  où A_t est l'ensemble des instants correspondant à la même heure que t dans l'ensemble d'entrainement et dans l'ensemble de prévision disponible jusqu'à présent. Les règles RF_2–RF_3 sont décrite ci-après.

- *Modèle linéaire :* Il consiste à prévoir un modèle linéaire de $m \leqslant M$ variables contextuelles plutôt que simplement calculer une moyenne pondérée des observations passées Y_s. De cette manière, le prédicteur devrait être capable de mieux extrapoler l'information assimilée jusqu'à présent vers de nouvelles zones inexplorées. La méthode nous a semblé être nouvelle. On note $\phi : \{1, \dots, m\} \to \{1, \dots, M\}$ l'application injective déterminant les m attributs considérés dans le modèle linéaire. Le prédicteur prévoit alors à l'instant t,

$$\widehat{Y}_t = \sum_{n=1}^{m} X_{\phi(n)t} \widehat{a}_{nt} + \widehat{a}_{m+1t} \,, \qquad \text{(Modèle linéaire)}$$

Version	366 jours	320 jours
RF_0	1353	1297
RF_1	1204	1133
RF_2	1161	1074
RF_3	1130	1035

TABLE 6 – Performances des prédicteurs de type forêts aléatoires en RMSE (MW) sur l'ensemble de l'année 2007, ainsi que sur le jeu de donnés utilisé dans nos expériences sur les algorithmes de mélange.

où $(\widehat{a}_t) \in \underset{a\in\mathbb{R}^{m+1}}{\arg\min} \sum_{s\in S_t} \mathrm{prox}(t,s) \left(Y_s - \sum_{n=1}^{m} X_{\phi(n)s} a_n - a_{m+1} \right)^2 .$

RF_2 correspond à $m = 1$ et ne considère que la consommation d'énergie avant la dernière mise à jour (i.e., dernier midi). On a considéré de nombreux autres prédicteurs de ce type, en faisant varier m, le nombre de variable contextuelles pris en compte dans le modèle linéaire, et leur nature (température,...). Par soucis de consision, on n'a conservé dans ce rapport que le prédicteur donnant les meilleurs résultats.

- *Correction du biais :* la consommation d'énergie n'est pas vraiment un processus stationnaire, mais a tendance à dériver et s'accroitre d'année en année. Les prévisions basées sur un ensemble d'entrainement plus ancien se retrouvent ainsi biaisées. Peut-on les rajuster ? Dans cette optique, on tente d'estimer le biais courant avant de former la prévision et la corriger. Plus formellement, on considère une correction de type Ridge :

$$\widehat{Y}'_t = \widehat{\theta}_t \widehat{Y}_t \,, \qquad \text{(Correction du biais)}$$

où

$$\widehat{\theta}_t = \underset{\theta\in\mathbb{R}}{\arg\min} \sum_{s\in S_t} \left(Y_s - \theta \widehat{Y}_s \right)^2 = \frac{\sum_{s\in S_t} X_s Y_s}{\sum_{s\in S_t} X_s^2}$$

est l'estimation du biais (multiplicatif) courant et $S_t = [\![\sigma(t) - h, \sigma(t)]\!]$ est une fenêtre glissante de longueur h et terminant à la dernière mise à jour $\sigma(t)$ (en raison de la contrainte opérationnelle). D'autres estimations du biais, comme $\widehat{\theta}_t = 1/|S_t| \sum_{s\in S_t} Y_s/X_s$, mènent à des résultats similaires.