Économétrie et *Machine Learning*

Econometrics and Machine Learning

Arthur Charpentier*, Emmanuel Flachaire et Antoine Ly*****

Résumé – L'économétrie et l'apprentissage automatique semblent avoir une finalité en commun : construire un modèle prédictif, pour une variable d'intérêt, à l'aide de variables explicatives (ou *features*). Pourtant, ces deux approches se sont développées en parallèle, créant ainsi deux cultures différentes. La première visait à construire des modèles probabilistes permettant de décrire des phénomènes économiques. La seconde utilise des algorithmes qui vont apprendre de leurs erreurs, dans le but, le plus souvent, de classer (des sons, des images, etc.). Or récemment, les modèles d'apprentissage se sont montrés plus efficaces que les techniques économétriques traditionnelles (bien qu'au prix d'un moindre pouvoir explicatif) et ils arrivent à gérer des données beaucoup plus volumineuses. Dans ce contexte, il devient nécessaire que les économètres comprennent ce que sont ces deux cultures, ce qui les oppose et surtout ce qui les rapproche, afin de s'approprier des outils développés par la communauté de l'apprentissage statistique pour les intégrer dans des modèles économétriques.

Abstract *– On the face of it, econometrics and machine learning share a common goal: to build a predictive model, for a variable of interest, using explanatory variables (or features). However, the two fields have developed in parallel, thus creating two different cultures. Econometrics set out to build probabilistic models designed to describe economic phenomena, while machine learning uses algorithms capable of learning from their mistakes, generally for classification purposes (sounds, images, etc.). Yet in recent years, learning models have been found to be more effective than traditional econometric methods (the price to pay being lower explanatory power) and are, above all, capable of handling much larger datasets. Given this, econometricians need to understand what the two cultures are, what differentiates them and, above all, what they have in common in order to draw on tools developed by the statistical learning community with a view to incorporating them into econometric models.*

Codes JEL / JEL Classification : C18, C52, C55
Mots-clés : apprentissage, données massives, économétrie, modélisation, moindres carrés
Keywords: learning, Big Data, econometrics, modelling, least squares

** Université de Rennes 1 & CREM (arthur.charpentier@univ-rennes1.fr)*
*** Aix-Marseille Université, AMSE, CNRS & EHESS (emmanuel.flachaire@univ-amu.fr)*
**** Université Paris-Est (antoine.ly.pro@gmail.com)*

Reçu le 2 septembre 2017, accepté après révisions le 29 mai 2018

Pour citer cet article : Charpentier, A., Flachaire, E. & Ly, A. (2018). Econometrics and Machine Learning. *Economie et Statistique / Economics and Statistics,* 505-506, 147–169. https://doi.org/10.24187/ecostat.2018.505d.1970

L'utilisation de techniques quantitatives en économie remonte probablement au 16e siècle (Morgan, 1990). Mais il faudra attendre le début du 20e siècle pour que le terme « économétrie » soit utilisé pour la première fois, donnant naissance à l'*Econometric Society* en 1933. Les techniques d'apprentissage automatique sont plus récentes. C'est à Arthur Samuel, considéré comme le père du premier programme d'auto-apprentissage, que l'on doit le terme *machine learning* qu'il définit comme « *a field of study that gives computer the ability without being explicitly programmed* » (Samuel, 1959). Parmi les premières techniques, on peut penser à la théorie des assemblées de neurones proposée dans Hebb (1949) (qui donnera naissance au « perceptron » dans les années 1950, puis aux réseaux de neurones) dont Widrow et Hoff (1960) montreront quinze ans plus tard les liens avec les méthodes des moindres carrés, aux SVM (*support vector machine*) et plus récemment aux méthodes de *boosting*. Si les deux communautés ont grandi en parallèle, les données massives imposent de créer des passerelles entre les deux approches, en rapprochant les « deux cultures » évoquées par Breiman (2001a), opposant la statistique mathématique, que l'on peut rapprocher de l'économétrie traditionnelle (Aldrich, 2010), à la statistique computationnelle, et à l'apprentissage machine de manière générale.

L'économétrie et les techniques d'apprentissage statistique supervisé sont proches, tout en étant très différentes. Proches au départ, car toutes les deux utilisent une base (ou un tableau) de données, c'est-à-dire des observations $\{(y_i, x_i)\}$, avec $i = 1, \cdots, n$, $x_i \in \mathcal{X} \subset \mathbb{R}^p$ et $y_i \in \mathcal{Y}$. Si y_i est qualitative, on parlera d'un problème de classification[1], sinon d'un problème de régression. Proches à l'arrivée, car dans les deux cas, on cherche à construire un « modèle », c'est à dire une fonction $m : \mathcal{X} \mapsto \mathcal{Y}$ qui sera interprétée comme une prévision.

Mais entre le départ et l'arrivée, il existe de réelles différences. Historiquement, les modèles économétriques s'appuient sur une théorie économique, avec le plus souvent des modèles paramétriques. On a alors recours aux outils classiques de l'inférence statistique (comme le maximum de vraisemblance ou la méthode des moments) pour estimer les valeurs d'un vecteur de paramètres θ, dans un modèle paramétrique $m_\theta(\cdot)$, par une valeur $\hat{\theta}$. Comme en statistique, avoir des estimateurs sans biais est important car on peut quantifier une borne inférieure pour la variance (borne de Cramér-Rao). La théorie asymptotique joue alors un rôle important (développements de Taylor, loi des grands nombres, et théorème central limite). En apprentissage statistique, en revanche, on construit souvent des modèles non-paramétriques, reposant presque exclusivement sur les données (i.e. sans hypothèse de distribution), et les méta-paramètres utilisés (profondeur de l'arbre, paramètre de pénalisation, etc.) sont optimisés par validation croisée.

Au-delà des fondements, si l'économétrie étudie abondamment les propriétés (souvent asymptotiques) de $\hat{\theta}$ (vu comme une variable aléatoire, grâce à la représentation stochastique sous-jacente), l'apprentissage statistique s'intéresse davantage aux propriétés du modèle optimal $m^*(\cdot)$ suivant un critère qui reste à définir, voire simplement $m^*(x_i)$ pour quelques observations i jugées d'intérêt par exemple dans une population de test. Le problème de choix de modèle est aussi vu sous un angle assez différent. Suivant la loi de Goodhart (« *si une mesure devient un objectif, elle cesse d'être une mesure* »), les économètres pénalisent *ex post* la qualité d'ajustement d'un modèle par sa complexité lors de la phase de validation ou de choix, alors qu'en apprentissage statistique, c'est la fonction objectif qui tiendra compte d'une pénalisation.

De la grande dimension aux données massives

Dans cet article, une variable sera un vecteur de $\mathbb{R}^n$, de telle sorte qu'en concaténant les variables on puisse les stocker dans une matrice X, de taille $n \times p$, avec n et p potentiellement grands[2]. Le fait que n soit grand n'est pas un problème en soi. De nombreux théorèmes en économétrie et en statistique sont obtenus lorsque $n \to \infty$. En revanche, le fait que p soit grand est problématique, en particulier si $p > n$.

Portnoy (1988) a montré que l'estimateur du maximum de vraisemblance conserve la propriété de normalité asymptotique si p

1. Nous utiliserons le terme « classification » lorsque y *est un ensemble de classes, typiquement une classification binaire,* $y = \{0,1\}$*, ce cas correspondant à la réalisation d'une variable indicatrice. Ce terme est moins daté que « discrimination » et plus général que la constitution d'un « score » (souvent une étape intermédiaire). Il ne doit pas être confondu avec la classification non-supervisée (comme la classification ascendante hiérarchique) qui est la constitution de classes homogènes à partir d'une mesure de similarité (on utilisera parfois, dans ce cas, le terme de « constitution de classes » ou de « clusters »).*

2. Des extensions sont possibles avec des images de type IRM comme variables prédictives, ou des données climatiques avec des cartes en variables prédictives. Il est possible de se ramener dans le cas usuel de données sous formes de vecteurs en utilisant la décomposition de Tucker (Kolda & Bader, 2009).

reste petit devant n ($p^2/n \to 0$ lorsque $n, p \to \infty$). Aussi, il n'est pas rare de parler de grande dimension dès lors que $p > \sqrt{n}$. Un autre concept important est celui de « sparsité », qui repose non pas sur la dimension p mais sur la dimension effective, autrement dit le nombre de variables effectivement significatives. Il est alors possible d'avoir $p > n$ tout en ayant des estimateurs convergents.

La grande dimension peut faire peur à cause de la « malédiction de la dimension » (Bellman, 1957). Le volume de la sphère unité, en dimension p, tend vers 0 lorsque $p \to \infty$. On dit alors que l'espace est « parcimonieux » – c'est-à-dire que la probabilité de trouver un point proche d'un autre devient de plus en plus faible (on pourrait parler d'espace « clairsemé »). L'idée de réduire la dimension en considérant une analyse en composante principale peut paraître séduisante, mais l'analyse souffre d'un certain nombre de défauts en grande dimension. La solution est alors souvent la sélection de variables (qui pose le problème des tests multiples ou du temps de calcul).

Pour reprendre la terminologie de Bühlmann & van de Geer (2011), les problèmes que nous évoquons ici correspondent à ceux observés en grande dimension, problème essentiellement statistique. D'un point de vue informatique, on peut aller un peu plus loin, avec des données réellement massives. Dans ce qui précède, les données étaient stockées dans une matrice X, de taille $n \times p$. Il peut y avoir des soucis à stocker une telle matrice, voire manipuler une matrice abondamment utilisée en économétrie, $X^T X$ ($n \times n$). La condition du premier ordre du modèle linéaire est associée à la résolution de $X^T(X\beta - y) = 0$. En dimension raisonnable, on utilise la décomposition de Gram-Schmidt. En grande dimension, on peut utiliser des méthodes numériques de descente de gradient, où le gradient est approché sur un sous-échantillon de données (Zinkevich *et al.,* 2010). Cet aspect informatique est souvent oublié alors qu'il a été à la base de bon nombre d'avancées méthodologiques en économétrie.

Statistique computationnelle et non-paramétrique

L'objet de cet article est d'expliquer les différences majeures entre l'économétrie et l'apprentissage statistique, correspondant aux deux cultures mentionnées par Breiman (2001a), lorsqu'il évoque en modélisation statistique la *data modeling culture* (reposant sur un modèle stochastique, comme la régression logistique ou le modèle de Cox) et la *algorithmic modeling culture* (reposant sur la mise en œuvre d'un algorithme, comme dans les forêts aléatoires ou les supports vecteurs machines ; une liste exhaustive est présentée dans Shalev-Shwartz & Ben-David, 2014). Mais la frontière entre les deux est très poreuse. À l'intersection se retrouve, par exemple, l'économétrie non-paramétrique. Cette dernière repose sur un modèle probabiliste (comme l'économétrie), tout en insistant davantage sur les algorithmes, et leurs performances, plutôt que sur des théorèmes asymptotiques.

Quelques outils d'apprentissage automatique

Réseaux de neurones

Les réseaux de neurones sont des modèles semi-paramétriques. Néanmoins, cette famille de modèles peut être appréhendée de la même manière que les modèles non-paramétriques : la structure des réseaux de neurones (présentée par la suite) peut être modifiée afin d'étendre la classe des fonctions utilisées pour approcher une variable d'intérêt. Plus précisément, Cybenko (1989) a démontré que l'ensemble des fonctions neuronales est dense dans l'espace des fonctions continues sur un compact. En d'autres termes, on a un cadre théorique permettant de garantir une forme d'approximation universelle. Il impose en outre une définition d'un neurone et met en avant l'existence d'un nombre de neurones suffisant pour approcher toute fonction continue sur un compact. Ainsi, un phénomène continu peut être approché par une suite de neurones : on appellera cette suite « réseau de neurones à une couche ». Si ce théorème d'approximation universelle est démontré en 1989, le premier neurone artificiel fonctionnel fut introduit par Franck Rosenblatt au milieu du 20e siècle, dans Rosenblatt (1958). Ce neurone, qualifié de nos jours de « neurone élémentaire », porte le nom de « perceptron ». Il a permis dans ses premières utilisations de déterminer le sexe d'un individu sur la base d'une photo. Il introduit le premier formalisme mathématique d'un neurone biologique :

- les synapses apportant l'information à la cellule sont formalisées par un vecteur réel. La dimension du vecteur d'entrée du neurone (qui n'est autre qu'une fonction) correspond biologiquement au nombre de connections synaptiques ;

- chaque signal apporté par une synapse est ensuite analysé par la cellule. Mathématiquement, ce schéma est transcrit par la pondération des différents éléments constitutifs du vecteur d'entrée ;

- en fonction de l'information acquise, le neurone décide de retransmettre ou non un signal. Ce phénomène est répliqué par l'introduction d'une fonction d'activation. Le signal de sortie est modélisé par un nombre réel calculé comme image par la fonction d'activation du vecteur d'entrée pondéré.

Ainsi, un neurone artificiel est un modèle semi-paramétrique. Le choix de la fonction d'activation est en effet laissé à l'utilisateur. On peut alors définir un neurone élémentaire formellement par :

1. un espace d'entrée $\mathcal{X}$, généralement $\mathbb{R}^k$ avec $k \in \mathbb{N}^*$;

2. un espace de sortie $\mathcal{Y}$, généralement $\mathbb{R}$ ou un ensemble fini (classiquement $\{0,1\}$, mais on préférera ici $\{-1,+1\}$) ;

3. un vecteur de paramètres $w \in \mathbb{R}^p$;

4. une fonction d'activation $\phi : \mathbb{R} \to \mathbb{R}$. Cette fonction doit être dans l'idéal monotone, dérivable et bornée (on dira ici « saturante ») afin de s'assurer de certaines propriétés de convergence.

Cette dernière fonction ϕ fait penser aux transformations logistique ou probit, populaires en économétrie (qui sont des fonctions de répartition, à valeur dans $[0,1]$, idéal quand $\mathcal{Y}$ est l'ensemble $\{0,1\}$). Pour les réseaux de neurones, on utilisera plutôt la tangente hyperbolique, la fonction arc-tangente ou les fonctions sigmoïdes pour des problèmes de classification sur $\mathcal{Y} = \{-1,+1\}$ (ces dernières évoqueront la transformation logistique des économètres). On appellera neurone toute application f_w de $\mathcal{X}$ dans $\mathcal{Y}$ définie par :

$$y = f_w(x) = \phi\left(w^T x\right), \quad \forall x \in \mathcal{X}$$

Pour le perceptron introduit par Rosenblatt (1958), on assimile un neurone élémentaire à la fonction :

$$y = f_w(x) = signe\left(w^T x\right), \quad \forall x \in \mathcal{X}$$

Selon cette formalisation, beaucoup de modèles statistiques, comme par exemple les régressions logistiques, pourraient être vus comme des neurones. Tout modèle GLM (*Generalized Linear Model*) pourrait s'interpréter comme un neurone formel où la fonction d'activation ϕ n'est d'autre que l'inverse de la fonction de lien canonique. Si g désigne la fonction de lien du GLM, w le vecteur de paramètres, y la variable à expliquer et x le vecteur des variables explicatives de même dimension que w :

$$g(\mathbb{E}[Y \mid X = x]) = w^T x$$

On retrouve la modélisation neuronale en prenant $\phi = g^{-1}$. Cependant, la différence majeure entre les GLM et le modèle neuronal est que ce dernier n'introduit aucune hypothèse de distribution sur $Y \mid X$ (on n'a d'ailleurs pas besoin d'introduire ici de modèle probabiliste). D'autre part, lorsque le nombre de neurones par couche augmente, la convergence n'est pas nécessairement garantie si la fonction d'activation ne vérifie pas certaines propriétés (qu'on ne retrouve pas dans la majorité des fonctions de liens canoniques des GLM). Cependant, la théorie des réseaux de neurones introduit des contraintes mathématiques supplémentaires sur la fonction g (détaillé dans Cybenko, 1989). Ainsi par exemple, une régression logistique peut être perçue comme un neurone alors que les régressions linéraires généralisées ne vérifient pas toutes les hypothèses nécessaires.

Toujours par analogie avec le fonctionnement du système nerveux, il est alors possible de connecter différents neurones entre eux. On parlera de structure de réseaux de neurones par couche. Chaque couche de neurones recevant à chaque fois le même vecteur d'observation. Pour revenir à une analogie plus économétrique, on peut imaginer passer par une étape intermédiaire, par exemple en ne faisant pas une régression sur les variables brutes x mais un ensemble plus faible de variables orthogonales obtenues suite à une analyse en composantes principales. Soit A la matrice associée à cette transformation linéaire, avec A de taille $k \times p$ si on souhaite utiliser les p premières composantes. Notons z la transformation de x, au sens où $z = A^T x$ ($z_j = A_j^T x$). Une généralisation du modèle précédant peut-être de poser :

$$y = f(x) = \phi\left(w^T z\right) = \phi\left(w^T A^T x\right), \quad \forall x \in \mathcal{X}$$

où $w \in \mathbb{R}^p$. On a ici une transformation linéaire (en considérant une analyse en composantes principales) mais on peut imaginer une généralisation avec des transformées non-linéaires :

$$y = f(x) = \phi\left(w^T F_A(x)\right), \quad \forall x \in \mathcal{X}$$

où F est une fonction $\mathbb{R}^k \to \mathbb{R}^p$. C'est le réseau de neurone à deux couches. Plus généralement,

pour formaliser la construction, on introduit les notations suivantes :

• $K \in \mathbb{N}^*$: nombre de couches ;

• $\forall k \in \{1,\cdots K\}$, p_k représente le nombre de neurones dans la couche k ;

• $\forall k \in \{1,\cdots K\}$, W_k désigne la matrice des paramètres associés à la couche k. Plus précisément, W_k est une matrice $p_k \times p_{k-1}$ et pour tout $\in \{1,\cdots p_k\}$, $w_{k,l} \in \mathbb{R}^{p_{k-1}}$ désigne le vecteur de poids associé au neurone élémentaire l de la couche k ;

• on appellera $W = \{W_1,..,W_K\}$, l'ensemble des paramètres associés au réseau de neurones ;

• $F^k_{W_k} : \mathbb{R}^{p_{k-1}} \to \mathbb{R}^{p_k}$ désigne la fonction de transfert associé à la couche k. Pour des raisons de simplification, on pourra également écrire F^k ;

• $\widehat{y}_k \in \mathbb{R}^{p_k}$ représentera le vecteur image de la couche $k \in \{1,\cdots,K\}$;

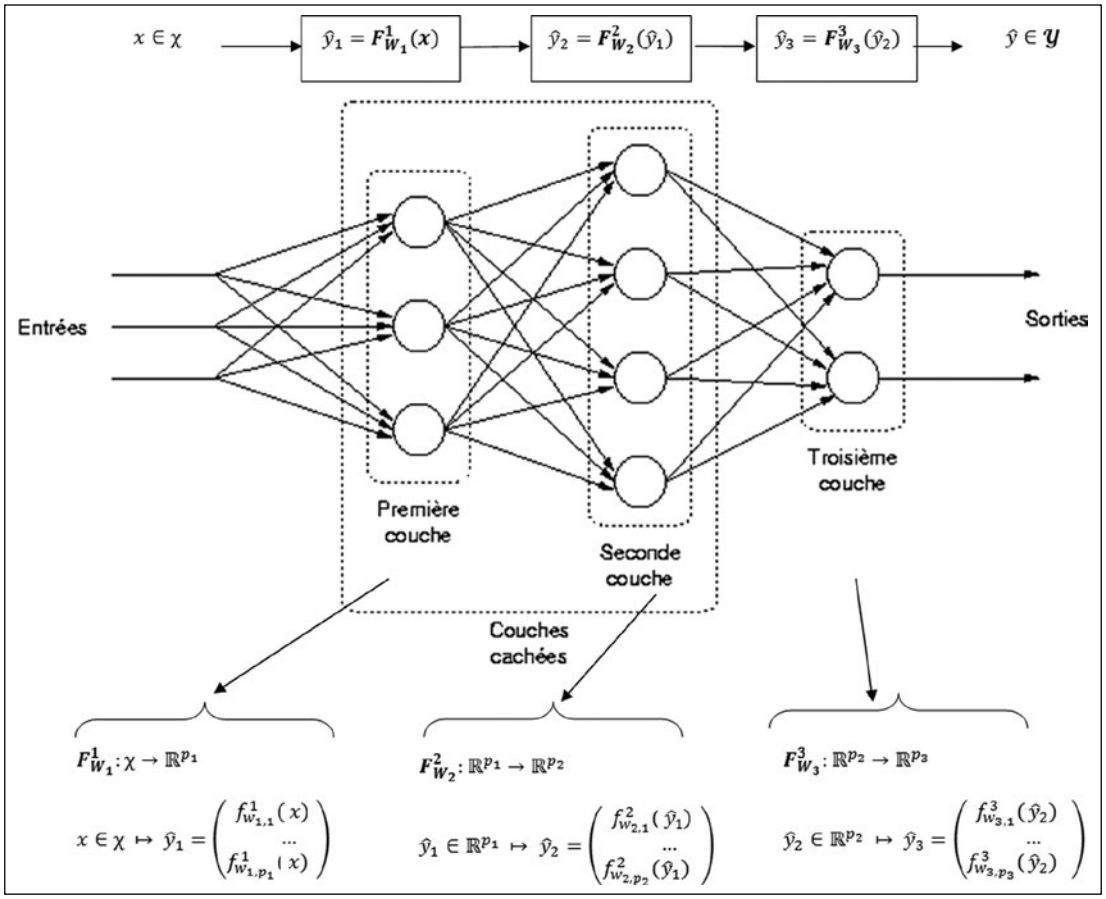
• on appellera $F = F_W = F^K \circ \cdots \circ F^1$ la fonction de transfert associée au réseau global. À ce titre, si $x \in \mathcal{X}$, on pourra noter $\widehat{y} = F_W(x)$.

Le schéma 1 permet d'illustrer les notations présentées ici[3]. Chaque cercle représente un neurone élémentaire. Chaque rectangle englobant plusieurs cercles représente une couche. On parle de couche d'entrée pour la première couche prenant en « input » les observations $x \in \mathcal{X}$, de couche de sortie pour la couche fournissant en « output » la prédiction $\widehat{y} \in \mathcal{Y}$. Les autres couches sont couramment appelées couches cachées. Un réseau de neurones multicouches est donc également un modèle semi-paramétrique dont les paramètres sont l'ensemble des composantes des matrices W_k pour tout entier k de $\{1,\cdots,K\}$. Chaque fonction d'activation associée à chaque neurone (chaque cercle de la figure I) est à déterminer par l'utilisateur.

Une fois que les paramètres à calibrer du modèle sont identifiés (ici, les réels constituant les matrices W_k pour chaque couche $k \in \{1,\cdots,K\}$), il est nécessaire de fixer une fonction de perte ℓ. En effet, on rappelle que

3. *Voir : http://intelligenceartificielle.org.*

Schéma 1
Exemple de notations associées aux réseaux de neurones multicouche



l'objectif de l'apprentissage supervisé sur une base d'apprentissage de $n \in \mathbb{N}^*$ couples $(y_i, x_i) \in \mathcal{Y} \times \mathcal{X}$ est de minimiser le risque empirique (voir complément en ligne[4]) :

$$\widehat{\mathcal{R}}_n(F_W) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, F_W(x_i)\right)$$

Afin d'illustrer ces propos, intéressons-nous à l'exemple suivant qui illustrera également la démarche opérée. Supposons que nous observons un phénomène y au travers de n observations $y_i \in [-1,1]$. On souhaiterait expliquer ce phénomène à partir des variables indépendantes x que l'on suppose à valeurs réelles. La « théorie de l'approximation universelle » nous indique qu'un réseau à une couche de neurones devrait permettre de modéliser le phénomène (sous hypothèse qu'il soit continu). Mais ce théorème ne donne pas de vitesse de convergence. L'utilisateur garde le choix de la structure, celle-ci pourrait être un simple neurone dont la fonction d'activation serait la fonction tangente hyperbolique :

$$y_1 = \tanh(w_0 + w_1 x)$$

où les paramètres w_0, w_1 sont à optimiser pour minimiser le risque empirique sur les données d'apprentissage.

Si l'on suit la philosophie du théorème d'approximation universelle, en ajoutant plusieurs neurones, l'erreur est censée diminuer. Cependant, ne connaissant pas la fonction à estimer, on ne peut l'observer qu'aux travers de l'échantillon. Mécaniquement, l'erreur sur la base d'apprentissage diminue lorsqu'on ajoute des paramètres. L'analyse de l'erreur sur la base de test permet alors d'évaluer notre capacité à généraliser (encadré 1).

On peut ainsi s'intéresser à un second modèle qui cette fois utilise plusieurs neurones. Par exemple :

$$y_2 = w_a \tanh(w_0 + w_1 x) + w_b \tanh(w_2 + w_3 x) + w_c \tanh(w_4 + w_5 x)$$

où les paramètres $w_0, .., w_5$ ainsi que w_a, w_b, w_c sont les paramètres à optimiser. Calibrer un réseau de neurones revient alors à réitérer ces étapes de modification de la structure jusqu'à minimisation du risque sur la base de test.

Pour une structure de réseau de neurones fixée (c'est-à-dire nombre de couches, nombre de neurones par couches et fonctions d'activation fixés), le programme revient donc à déterminer l'ensemble de paramètres $W^* = (W_1, ..., W_K)$ de sorte que :

$$W^* \in \underset{W=(W_1,...,W_K)}{\operatorname{argmin}} \left\{ \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, F_W(x_i)\right) \right\}.$$

De cette formule apparaît l'importance du choix de la fonction ℓ. Cette fonction de perte quantifie l'erreur moyenne commise par notre modèle F_W sur la base d'apprentissage. *A priori*, ℓ peut être choisie arbitrairement. Cependant, dans l'optique de résoudre un programme d'optimisation, on préféra des fonctions de coût sous-différentiables et convexes afin de garantir la convergence des algorithmes d'optimisation. Parmi les fonctions de perte classiques, en plus de la fonction de perte quadratique $\ell_2(y, \hat{y}) = (y - \hat{y})^2$ on retiendra la fonction dite $Hinge - \ell(y, \hat{y}) = max(0, 1 - y\hat{y})$ – ou la fonction dite logistique $-\ell(y, \hat{y}) = \log(1 - e^{-y\hat{y}})$.

4. *Lien vers les compléments en ligne en fin d'article.*

ENCADRÉ 1 – **Échantillons d'apprentissage et de test**

Dans la littérature en apprentissage, juger de la qualité d'un modèle sur les données qui ont servi à le construire ne permet en rien de savoir comment le modèle se comportera sur des nouvelles données. Il s'agit du problème dit de « généralisation ». L'approche classique consiste alors à séparer l'échantillon (de taille n) en deux : une partie pour entraîner le modèle (la base d'apprentissage, *in-sample*, de taille m) et l'autre pour le tester (la base de test, *out-of-sample*, de taille $n - m$). Cette dernière permet de mesurer un vrai risque prédictif. Supposons que les données soient générées par un modèle linéaire $y_i = x_i^T \beta_0 + \varepsilon_i$ où les ε_i sont des réalisations de lois indépendantes et centrées. Le risque quadratique empirique *in-sample* est :

$$\frac{1}{m}\sum_{i=1}^{m} \mathbb{E}\left(\left[x_i^T \hat{\beta} - x_i^T \beta_0\right]^2\right) = \mathbb{E}\left(\left[x_i^T \hat{\beta} - x_i^T \beta_0\right]^2\right)$$

pour n'importe quelle observation i. Si les résidus ε sont Gaussiens, ce risque vaut $\sigma^2 p / m$, où p est la taille des vecteurs x_i. En revanche le risque quadratique empirique *out-of-sample* est :

$$\mathbb{E}\left(\left[x^T \hat{\beta} - x^T \beta_0\right]^2\right)$$

où x est une nouvelle observation, indépendante des autres. On peut noter que :

$$\mathbb{E}\left(\left[x^T \hat{\beta} - x^T \beta_0\right]^2 \mid x\right) = \sigma^2 x^T (X^T X)^{-1} x$$

et en intégrant par rapport à x :

$$\mathbb{E}\left(\left[x^T \hat{\beta} - x^T \beta_0\right]^2\right) = \mathbb{E}\left(\mathbb{E}\left(\left[x^T \hat{\beta} - x^T \beta_0\right]^2 \mid x\right)\right) = \sigma^2 \operatorname{trace}\left(\mathbb{E}[xx^T]\mathbb{E}[X^T X]^{-1}\right)$$

➔

ENCADRÉ 1 (suite)

L'expression est alors différente de celle obtenue *in-sample*, et en utilisation la majoration de Groves et Rothenberg (1969), on peut montrer que :

$$\mathbb{E}\left(\left[x^T\widehat{\beta}-x^T\beta_0\right]^2\right)\geq\sigma^2\frac{p}{m}$$

Hormis certains cas simples, il n'y a pas de formule simple. Notons toutefois que si $x\sim\mathcal{N}\left(0,\sigma^2\mathbb{I}\right)$, alors x^Tx suit une loi de Wishart, et :

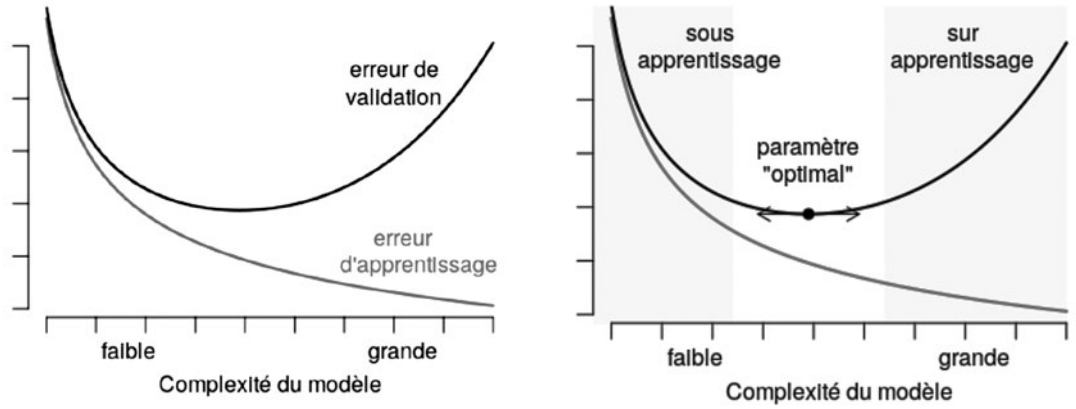
$$\mathbb{E}\left(\left[x^T\widehat{\beta}-x^T\beta_0\right]^2\right)=\sigma^2\frac{p}{m-p-1}$$

Si on regarde maintenant la version empirique : si $\widehat{\beta}$ est estimé sur les m premières observations,

$$\widehat{\mathcal{R}}^{\text{IS}}=\sum_{i=m+1}^{m}[y_i-x_i^T\widehat{\beta}]^2 \text{ et } \widehat{\mathcal{R}}^{\text{OS}}=\sum_{i=m+1}^{n}[y_i-x_i^T\widehat{\beta}]^2$$

comme l'a noté Leeb (2008), $\widehat{\mathcal{R}}^{\text{IS}}-\widehat{\mathcal{R}}^{\text{OS}}\approx 2\cdot\nu$ où ν représente le nombre de degrés de libertés. La figure A montre l'évolution respective de $\widehat{\mathcal{R}}^{\text{IS}}$ et $\widehat{\mathcal{R}}^{\text{OS}}$ en fonction de la complexité du modèle (nombre de degrés dans une régression polynomiale, nombre de noeuds dans des splines, etc). $\widehat{\mathcal{R}}^{\text{IS}}$ diminue toujours avec la complexité (courbe claire). Mais $\widehat{\mathcal{R}}^{\text{IS}}$ est non monotone (courbe foncée). Si le modèle est trop simple, il prédit mal, mais s'il est trop complexe, on est dans une situation de « sur-apprentissage » : il commence à modéliser le bruit.

Figure A
Généralisation et sur-apprentissage



Lecture : la courbe claire représente le risque empirique *in-sample* sur l'échantillon d'apprentissage, la courbe foncée le risque *out-of-sample* sur l'échantillon de test.

Les réseaux de neurones ont été utilisés très tôt en économie et en finance, en particulier sur les défauts d'entreprises (Tam & Kiang, 1992 ; Altman *et al.,* 1994) ou plus récemment la notation de crédit (Blanco *et al.,* 2013 ; Khashman, 2011). Cependant les structures telles que présentées précédemment sont généralement limitées. L'apprentissage profond (ou *deep learning*) caractérise plus particulièrement des réseaux de neurones plus complexes (parfois plus d'une dizaine de couches avec des centaines de neurones par couche). Aujourd'hui, ces structures sont très populaires en analyse du signal (image, texte, son) car elles sont capables à partir d'une quantité d'observations très importante d'extraire des informations que l'humain ne peut percevoir et de faire face à des problèmes non linéaires (LeCun *et al.,* 2015).

L'extraction d'informations peut, par exemple, se faire grâce à la convolution. Procédé non supervisé, il a permis notamment d'obtenir d'excellentes performances dans l'analyse d'image. Techniquement, cela peut s'apparenter à une transformation à noyaux (comme utilisé dans les techniques SVM, voir section suivante). Si une image peut être perçue comme une matrice dont chaque coordonnée représente un pixel, une convolution reviendrait à appliquer une transformation sur un point (ou une zone) de cette matrice générant ainsi une nouvelle donnée. Le procédé peut ainsi être répété en appliquant des transformations différentes (d'où la notion de couches convolutives). Le vecteur final obtenu peut alors alimenter un modèle neuronal. Plus généralement, une couche de convolution peut être perçue comme un filtre qui permet de transformer la donnée initiale.

Une explication intuitive pour laquelle l'apprentissage approfondi, en particulier les réseaux profonds, est si puissant pour décrire des relations complexes dans les données, c'est leur construction autour de l'approximation fonctionnelle simple et l'exploitation d'une forme de hiérarchie (Lin *et al.,* 2016). Néanmoins les modèles de type *deep learning* sont plus difficiles à appréhender car ils nécessitent beaucoup de jugement empirique. Si aujourd'hui les bibliothèques *open sources* (keras, torch, etc.) permettent de paralléliser plus facilement les calculs en utilisant par exemple les GPU (*Graphical Processor Units*), il reste néanmoins à l'utilisateur de déterminer la structure du réseau de neurones le plus approprié.

Support vecteurs machine

Comme nous l'avons noté auparavant, dans les problèmes de classification en apprentissage machine (comme en traitement du signal) on préférera avoir des observations dans l'ensemble $\{-1,+1\}$ (plutôt que $\{0,1\}$ en économétrie). Avec cette notation, Cortes & Vapnik (1995) ont posé les bases théoriques des modèles dits *Support Vector Machine* (SVM), alternative aux réseaux de neurones alors très populaires. L'idée initiale des méthodes SVM consiste à trouver un hyperplan séparateur divisant l'espace en deux ensembles de points le plus homogène possible (i.e. contenant des labels identiques). En dimension deux, l'algorithme consiste à déterminer une droite séparant l'espace en deux zones les plus homogènes possibles. La résolution de ce problème possédant parfois une infinité de solutions (il peut en effet exister une infinité de droites qui séparent l'espace en deux zones distinctes et homogènes), on rajoute généralement une contrainte supplémentaire : l'hyperplan séparateur doit se trouver le plus éloigné possible des deux sous-ensembles homogènes qu'il engendre (schéma 2). On parlera ainsi de marge. L'algorithme ainsi décrit est alors un SVM linéaire à marge.

Si un plan peut être caractérisé entièrement par un vecteur directeur *w* orthogonal à ce dernier et une constante b, appliquer un algorithme SVM à un ensemble de $n \in \mathbb{N}^*$ points x_i de $\mathbb{R}^p$ labellisés par $y_i \in \{-1,1\}$ revient alors à résoudre un programme d'optimisation sous contrainte similaire à celui d'un lasso (distance quadratique sous contrainte linéaire, voir compléments en ligne). Particulièrement, on sera amené à résoudre :

$$\left(w^\star, b^\star\right) = \underset{w,b}{\operatorname{argmin}}\left\{\| w \|^2\right\} = \underset{w,b}{\operatorname{argmin}}\left\{w^T w\right\}$$

sous contraintes

$$\forall i \in \{1,\cdots,n\}, \begin{cases} \omega^T x_i + b \geq +1 \text{ lorsque } y_i = +1 \\ \omega^T x_i + b \leq -1 \text{ lorsque } y_i = -1 \end{cases}.$$

La contrainte peut être relâchée en autorisant que, dans un sous-ensemble, un point puisse ne pas être du même label que la majeure partie des points de ce sous-ensemble à condition de ne pas être trop loin de la frontière. C'est ce qu'on appelle les SVM linéaire à marge légère (*soft margin*). De manière heuristique, comme en pratique, bien souvent, on ne peut pas avoir $y_i\left(w^T x_i + b\right) - 1 \geq 0$ pour tout $i \in \{1,\cdots,n\}$, on relâche en introduisant des variables positives ξ telle que :

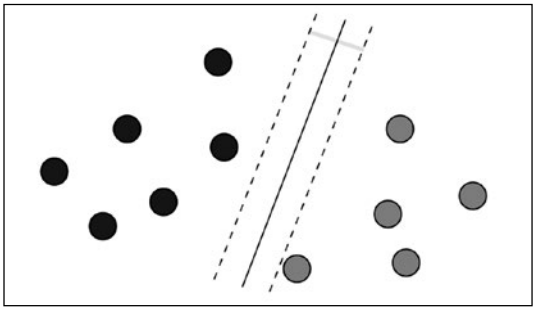
$$\begin{cases} \omega^T x_i + b \geq +1 - \xi_i \text{ lorsque } y_i = +1 \\ \text{É}^{\text{T}}\text{x}_{\text{i}} + b \leq -1 + \xi_i \text{ lorsque } y_i = -1 \end{cases} \quad (1)$$

avec $\xi_i \geq 0$. On a une erreur de classification si $\xi_i > 1$, et on va alors introduire une pénalité, un coût à payer pour chaque erreur commise. On cherche alors à résoudre un problème quadratique :

$$\min\left\{\frac{1}{2}\omega^T \omega + C1^T 1_{\xi>1}\right\}$$

sous la contrainte (1), qui pourra être résolu de manière numérique très efficacement par descente par coordonnées.

Schéma 2
Illustration d'un SVM à marge



Source : Vert (2017).

S'il n'est pas possible de séparer les points, une autre possibilité consiste à les transformer dans une dimension supérieure, de sorte que les données deviennent alors linéairement séparables. Trouver la bonne transformation qui sépare les données est toutefois très difficile. Une astuce mathématique pour résoudre ce problème avec élégance consiste à définir les transformations $T(\cdot)$ et les produits scalaires *via* un

noyau $K(x_1, x_2) = \langle T(x_1), T(x_2) \rangle$. L'un des choix les plus courants pour une fonction de noyau est la fonction de base radiale (noyau gaussien) $K(x_1, x_2) = \exp\left(-\|x_1 - x_2\|^2\right)$. Il n'existe néanmoins pas de règles à ce jour permettant de choisir le « meilleur » noyau. Cette technique est basée sur de la minimisation de distance, et il n'a aucune prévision de la probabilité d'être positif ou négatif, mais une interprétation probabiliste est néanmoins possible (Grandvalet *et al.*, 2005).

Arbres, *bagging* et forêts aléatoires

Les arbres de classification ont été introduits dans Breiman *et al.* (1984) puis Quinlan (1986). On parle de modèle CART pour *Classification And Regression Tree*. L'idée est de diviser consécutivement (par une notion de branchement) les données d'entrée jusqu'à ce qu'un critère d'affectation (par rapport à la variable cible) soit atteint, selon une règle prédéfinie.

L'intuition : l'entropie $H(x)$ est associée à la quantité de désordre dans les données x par rapport aux modalités prises par la variable de classification y, et chaque partition vise à réduire ce désordre. L'interprétation probabiliste est de créer les groupes les plus homogènes possible, en réduisant la variance par groupe (variance intra), ou de manière équivalente en créant deux groupes aussi différents que possible, en augmentant la variance entre les groupes (variance inter). À chaque étape, nous choisissons la partition qui donne la plus forte réduction de désordre (ou de variance). L'arbre de décision complet se développe en répétant cette procédure sur tous les sous-groupes, où chaque étape k aboutit à une nouvelle partition en 2 branches, qui subdivise notre ensemble de données en 2. Enfin, on décide quand mettre fin à cette constitution de nouvelles branches, en procédant à des affectations finales (nœuds dits foliaires). Il existe plusieurs options. L'une est de construire un arbre jusqu'à ce que toutes les feuilles soient pures, c'est-à-dire composées d'une seule observation. Une autre option est de définir une règle d'arrêt liée à la taille, ou à la décomposition, des feuilles. Les exemples de règles d'arrêt peuvent être d'une taille minimale (au moins 5 éléments par feuille), ou une entropie minimale. On parlera alors d'élagage de l'arbre : on laisse l'arbre grossir, puis on coupe certaines branches *a posteriori* (ce qui est différent de l'introduction d'un critère d'arrêt *a priori* au processus de croissance de l'arbre – par exemple en imposant une taille minimale aux feuilles, ou d'autres critères discutés dans Breiman *et al.*, 1984).

À un nœud donné, constitué de n_0 observations (x_i, y_i) avec $i \in \mathcal{I}_0$, on va couper en deux branches (une à gauche et une à droite), partitionnant ainsi $\mathcal{I}_0$ en $\mathcal{I}_g$ et $\mathcal{I}_d$. Soit I le critère d'intérêt, comme l'entropie du nœud :

$$I(y_0) = -n_0 p_0 \log p_0 \text{ où } p_0 = \frac{1}{n_0} \sum_{i \in \mathcal{I}_0} y_i$$

ou la variance du nœud :

$$I(y_0) = n_0 p_0 (1 - p_0) \text{ où } p_0 = \frac{1}{n_0} \sum_{i \in \mathcal{I}_0} y_i$$

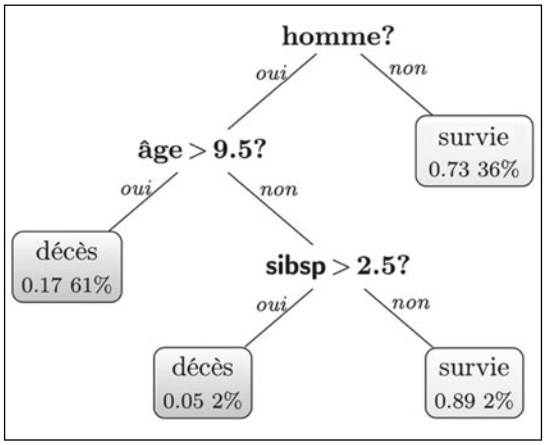
ce dernier étant également l'indice d'impureté de Gini.

On partitionnera entre la branche gauche et la branche droite si le gain $I(y_0) - \left[I(y_g) + I(y_d) \right]$ est suffisamment important. Lors de la construction des arbres, on va chercher la partition qui donne le gain le plus important possible. Ce problème combinatoire étant complexe, Breiman *et al.* (1984) suggère un découpage suivant une des variables, avec $\mathcal{I}_g = \{i \in \mathcal{I}_0 : x_{k,i} < s\}$ et $\mathcal{I}_d = \{i \in \mathcal{I}_0 : x_{k,i} > s\}$, pour une variable k et un seuil s (si la variable est continue, sinon on considère des regroupements de modalités pour des variables qualitatives).

Les arbres de décision ainsi décrits sont simples à obtenir et faciles à interpréter (comme le montre le schéma 3 sur les données du Titanic[5]),

5. *Ce jeu de données, contenant des informations sur tous les passagers et membres d'équipage du Titanic, dont la variable* y *indiquant si la personne a survécu a été abondamment utilisée pour illustrer les techniques de classification, voir https://www.kaggle.com/c/titanic/data.*

Schéma 3
Illustration d'un arbre de décision permettant de prédire le taux de survie d'un passager du Titanic



Lecture : une femme (homme : non) avait 73 % chances de survie et elles représentaient 36 % de la population.

mais ils sont peu robustes, et leur pouvoir prédictif est souvent très faible, en particulier si l'arbre est très profond. Une idée naturelle est de développer un ensemble de modèles d'arbres à peu près indépendants, qui prédisent conjointement mieux qu'un modèle d'arbre unique. On va utiliser le *bootstrap*, en tirant (avec remise) n observations parmi $\{(x_i, y_i)\}$. Chaque échantillon ainsi généré permet d'estimer un nouvel arbre de classification, formant ainsi une forêt d'arbres. C'est l'agrégation de tous ces arbres qui conduit à la prévision. Le résultat global est moins sensible à l'échantillon initial et donne souvent de meilleurs résultats de prévision. Ces techniques, appelées *bagging* pour *boostrap aggregating*, ressemblent aux techniques de *bootstrap* en régression (par exemple pour construire des tubes de confiance dans une régression fonctionnelle).

Le principe du *bagging* consiste à générer des échantillons aléatoires, en tirant avec remise dans l'échantillon d'origine, comme pour le *bootstrap*. Les forêts aléatoires, ou *random forests*, reposent sur le même principe que le *bagging*, mais lors de la construction d'un arbre de classification, à chaque branche, un sous-ensemble de m covariables est tiré aléatoirement. Autrement dit, chaque branche d'un arbre ne s'appuie pas sur le même ensemble de covariables. Cela permet d'amplifier la variabilité entre les différents arbres et d'obtenir, au final, une forêt composée d'arbres moins corrélés les uns aux autres.

Sélection de modèle de classification

Étant donné un modèle $m(\cdot)$ approchant $\mathbb{E}[Y \mid X = x]$, et un seuil $s \in [0,1]$, posons :

$$\widehat{y}^{(s)} = 1[m(x) > s] = \begin{cases} 1 \text{ si } m(x) > s \\ 0 \text{ si } m(x) \leq s \end{cases}$$

La matrice de confusion est alors le tableau de contingence associé aux comptages $N = \left[N_{u,v}\right]$ avec :

$$N_{u,v}^{(s)} = \sum_{i=1}^{n} 1\left(\widehat{y}^{(s)} = u, y_j = v\right)$$

pour $(u,v) \in \{0,1\}$. Le tableau 1 présente une telle matrice, avec le nom de chacun des éléments : VP pour vrais positifs, correspondant aux 1 prédits en 1, VN pour vrais négatifs, correspondant aux 0 prédits en 0, FP pour faux positifs, correspondant aux 0 prédits en 1, et FN pour faux négatifs, correspondant aux 1 prédits en 0.

Tableau 1
Matrice de confusion, ou tableau de contingence pour un seuil *s* donné

	$y = 0$	$y = 1$	
$\hat{y}_s = 0$	VN_s	FN_s	$VN_s + FN_s$
$\hat{y}_s = 1$	FP_s	VP_s	$FP_s + VP_s$
	$VN_s + FP_s$	$FN_s + VP_s$	n

Plusieurs quantités sont dérivées de ce tableau. La sensibilité correspond à la probabilité de prédire 1 dans la population des 1, ou taux de vrais positifs. La spécificité est la probabilité de prédire 0 dans la population des 0 ou taux de vrais négatifs. On s'intéressera toutefois davantage au taux de faux négatifs, c'est-à-dire la probabilité de prédire 1 dans la population des 0. La représentation de ces deux valeurs lorsque s varie donne la courbe ROC (*receiver operating characteristic*) :

$$ROC_s = \left(\frac{FP_s}{FP_s + VN_s}, \frac{VP_s}{VP_s + FN_s}\right)$$
$$= \left(sensibilité_s, 1 - spécificité_s\right) \quad \text{pour } s \in [0,1]$$

Une telle courbe est présentée dans la partie suivante, sur des données réelles. Les deux grandeurs intensivement utilisées en apprentissage automatique sont l'indice κ, qui compare la précision observée avec celle espérée avec un modèle aléatoire (Landis & Koch, 1977) et l'AUC correspondant à l'aire sous la courbe ROC. Pour le premier indice, une fois choisi s, notons $N^{\perp}$ le tableau de contingence correspond aux cas indépendants (défini à partir de N dans le test d'indépendance du chi²). On pose alors :

$$précision\ totale = \frac{VP + VN}{n}$$

alors que :

$$précision\ aléatoire = \frac{[VN + FP] \cdot [VP + FN] + [VP + FP] \cdot [VN + FN]}{n^2}$$

On peut alors définir :

$$\kappa = \frac{précision\ totale - précision\ aléatoire}{1 - précision\ aléatoire}$$

Classiquement s sera fixé égal à 0.5, comme dans une classification Bayésienne naïve, mais d'autres valeurs peuvent être retenues, en particulier si les deux erreurs ne sont pas symétriques. Il existe des compromis entre des modèles simples et complexes mesurés par leur nombre de paramètres (ou plus généralement les degrés de liberté) en matière de performance et de coût. Les modèles simples

sont généralement plus faciles à calculer, mais peuvent conduire à des ajustements plus mauvais (avec un biais élevé par exemple). Au contraire, les modèles complexes peuvent fournir des ajustements plus précis, mais risquent d'être coûteux en termes de calcul. En outre, ils peuvent surpasser les données ou avoir une grande variance et, tout autant que des modèles trop simples, ont de grandes erreurs de test. Comme nous l'avons rappelé auparavant, dans l'apprentissage machine, la complexité optimale du modèle est déterminée en utilisant le compromis de biais-variance.

De la classification à la régression

Historiquement, les méthodes d'apprentissage automatique se sont orientées autour des problèmes de classification (avec éventuellement plus de 2 modalités[6]), et assez peu dans le cas où la variable d'intérêt y est continue. Néanmoins, il est possible d'adapter quelques techniques, comme les arbres et les forêts aléatoires, le *boosting*, ou les réseaux de neurones.

Pour les arbres de régression, Morgan et Sonquist (1963) ont proposé la méthode AID, basée sur la formule de décomposition de la variance avec un algorithme proche de celui de la méthode CART décrite plus haut. Dans le contexte de la classification, on calculait, à chaque nœud (dans le cas de l'indice d'impureté de Gini) en sommant sur la feuille de gauche $\{x_{k,i} < s\}$ et celle de droite $\{x_{k,i} > s\}$:

$$I = \sum_{i:x_{k,i}<s} \bar{y}_g\left(1-\bar{y}_g\right) + \sum_{i:x_{k,i}>s} \bar{y}_d\left(1-\bar{y}_d\right)$$

où $\bar{y}_g$ et $\bar{y}_d$ désignent les fréquences de 1 dans la feuille de gauche et de droite, respectivement. Dans le cas d'un arbre de régression, on utilisera :

$$I = \sum_{i:x_{k,i}<s} (y_i-\bar{y}_g)^2 + \sum_{i:x_{k,i}>s} (y_i-\bar{y}_d)^2$$

qui va correspondre à la somme (pondérée) des variances intra. Le partage optimal sera celui qui aura le plus de variance intra (on veut les feuilles les plus homogènes possibles).

Dans le contexte des forêts aléatoires, on utilise souvent un critère majoritaire en classification (la classe prédite sera la classe majoritaire dans une feuille), alors que pour la régression, on utilise la moyenne des prédictions, sur tous les arbres. Dans un contexte de régression (variable y continue), l'idée est de créer une succession de modèles selon la méthode de *boosting* (encadré 2), qui prend ici la forme :

$$m^{(k)}(x) = m^{(k-1)}(x) + \alpha_k \underset{h\in\mathcal{H}}{argmin}\left\{\sum_{i=1}^{n}(y_i - m^{(k-1)}(x) + h(x))^2\right\}$$

où α_k est un paramètre de *shrinkage* et où le second terme correspond à un arbre de régression, sur les résidus, $y_i - m^{(k-1)}(x_i)$.

Mais il existe d'autres techniques permettant d'apprendre de manière séquentielle. Dans un modèle additif (GAM) on va chercher une écriture de la forme :

$$m(x) = \sum_{j=1}^{p} m_j\left(x_j\right) = m_1\left(x_1\right) + \cdots + m_p\left(x_p\right)$$

L'idée de la poursuite de projection repose sur une décomposition non pas sur les variables explicatives, mais sur des combinaisons linéaires. On va ainsi considérer un modèle :

$$m(x) = \sum_{j=1}^{k} g_j\left(\omega_j{}^T x\right) = g_1\left(\omega_1{}^T x\right) + \cdots + g_k\left(\omega_k{}^T x\right)$$

Tout comme les modèles additifs, les fonctions $g_1, \cdots, g_k$ sont à estimer, tout comme les directions $\omega_1, \cdots, \omega_k$. Cette écriture est relativement générale, et permet de tenir compte d'interactions et d'effets croisés (ce que nous ne pouvions pas faire avec les modèles additifs qui ne tiennent compte que de non-linéarités). Par exemple en dimension 2, un effet multiplicatif $m(x_1, x_2) = x_1 \cdot x_2$ s'écrit :

$$m(x_1, x_2) = x_1 \cdot x_2 = \frac{(x_1+x_2)^2}{4} - \frac{(x_1-x_2)^2}{4}$$

autrement dit $g_1(x) = x^2/4$, $g_2(x) = -x^2/4$, $\omega_1 = (1,1)^T$ et $\omega_2 = (1,-1)^T$. Dans la version simple, avec $k = 1$, avec une fonction de perte quadratique, on peut utiliser un développement de Taylor pour approcher $[y_i - g\left(\omega^T x_i\right)]^2$, et construire classiquement un algorithme itératif. Si on dispose d'une valeur initiale ω_0, notons que :

$$\sum_{i=1}^{n}[y_i - g\left(\omega^T x_i\right)]^2 \approx \sum_{i=1}^{n} g'(\omega_0{}^T x_i)^2 \left[\omega^T x_i + \frac{y_i - g\left(\omega_0{}^T x_i\right)}{g'\left(\omega_0{}^T x_i\right)} - \omega^T x_i\right]^2$$

qui correspondrait à l'approximation dans les modèles linéaires généralisés sur la fonction $g(\cdot)$ qui était la fonction de lien (supposée connue). On reconnaît un problème de moindres carrés pondérés. La difficulté ici est que les fonctions $g_j(\cdot)$ sont inconnues.

6. *Par exemple dans le cas de reconnaissance de lettres ou de chiffres.*

ENCADRÉ 2 – **Apprentissage lent par *boosting***

L'idée du *boosting*, tel qu'introduit par Shapire & Freund (2012), est d'apprendre, lentement, à partir des erreurs du modèle, de manière itérative. À la première étape, on estime un modèle m_1 pour y, à partir de X, qui donnera une erreur ε_1. À la seconde étape, on estime un modèle m_2 pour ε_1, à partir de X, qui donnera une erreur ε_2, etc. On va alors retenir comme modèle, au bout de k itérations :

$$\begin{aligned} m^{(k)}(\cdot) &= \underset{\sim y}{m_1(\cdot)} + \underset{\sim \varepsilon_1}{m_2(\cdot)} + \underset{\sim \varepsilon_2}{m_3(\cdot)} + \cdots + \underset{\sim \varepsilon_{k-1}}{m_k(\cdot)} \\ &= m^{(k-1)}(\cdot) + m_k(\cdot) \end{aligned} \quad (2)$$

Ici, l'erreur ε est vue comme la différence entre y et le modèle $m(x)$, mais elle peut aussi être vue comme le gradient associé à la fonction de perte quadratique.

L'équation (2) peut se voir comme une descente du gradient, mais écrit de manière duale. Le problème va alors se réécrire comme un problème d'optimisation :

$$m^{(k)} = m^{(k-1)} + \underset{h \in \mathcal{H}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \ell\left(y_i - m^{(k-1)}(x_i), h(x_i)\right) \right\} \quad (3)$$

où l'espace $\mathcal{H}$ est relativement simple (on parlera de *weak learner*). Classiquement, les fonctions $\mathcal{H}$ sont des fonctions en escalier (que l'on retrouvera dans les arbres de classification et de régression) appelées *stumps*. Afin de s'assurer que l'apprentissage est effectivement lent, il n'est pas rare d'utiliser un paramètre de *shrinkage*, et au lieu de poser, par exemple, $\varepsilon_1 = y - m_1(x)$, on posera $\varepsilon_1 = y - \alpha \cdot m_1(x)$ avec $\alpha \in [0,1]$.

Applications

Les données massives ont rendu nécessaire le développement de techniques d'estimation permettant de pallier les limites des modèles paramétriques, jugés trop restrictifs, et des modèles non-paramétriques classiques, dont l'estimation peut être difficile en présence d'un nombre élevé de variables. L'apprentissage statistique, ou apprentissage machine, propose de nouvelles méthodes d'estimation non-paramétriques, performantes dans un cadre général et en présence d'un grand nombre de variables[7]. Toutefois, l'obtention d'une plus grande flexibilité s'obtient au prix d'un manque d'interprétation qui peut être important.

En pratique, une question importante est de savoir quel est le meilleur modèle. La réponse à cette question dépend du problème sous-jacent. Si la relation entre les variables est correctement approximée par un modèle linéaire, un modèle paramétrique correctement spécifié devrait être performant. Par contre, si le modèle paramétrique n'est pas correctement spécifié, car la relation est fortement non-linéaire et/ou fait intervenir des effets croisés non-négligeables, alors les méthodes statistiques issues de l'apprentissage automatique devraient être plus performantes.

La bonne spécification d'un modèle de régression est une hypothèse souvent posée, elle est rarement vérifiée et justifiée. Dans les applications qui suivent, nous montrons comment les méthodes statistiques issues de l'apprentissage automatique peuvent être utilisées pour justifier la bonne spécification d'un modèle de régression paramétrique, ou pour détecter une mauvaise spécification.

Les ventes de sièges auto pour enfants (classification)

Nous reprenons ici un exemple utilisé dans James *et al.* (2013). Le jeu de données contient les ventes de sièges auto pour enfants dans 400 magasins (*sales*), ainsi que plusieurs variables, dont la qualité de présentation en rayonnage (*shelveloc*, égal à « mauvais », « moyen », « bon ») et le prix (*price*)[8]. Une variable dépendante binaire est artificiellement créée, pour qualifier une forte vente ou non (*high* = « oui » si *sales* > 8 et à « non » sinon). Dans cette application, on cherche à évaluer les déterminants d'un bon niveau de vente. Dans un premier temps, on considère un modèle de régression linéaire latent :

$$y^\star = \gamma + x^T\beta + \varepsilon, \quad \varepsilon \sim G(0,1), \quad (4)$$

où x est composé de k variables explicatives, β est un vecteur de k paramètres inconnus et ε est un terme d'erreur *i.i.d.* avec une fonction de répartition G d'espérance nulle et de variance unitaire. La variable dépendante y^* n'est pas observé, mais seulement y, avec :

$$y = \begin{cases} 1 & \text{si } y^\star > \xi \\ 0 & \text{si } y^\star \leq \xi \end{cases} \quad (5)$$

On peut alors exprimer la probabilité d'avoir y égal à 1, comme suit :

$$\mathbb{P}(Y=1) = G\left(\beta_0 + x^T\beta\right) \quad (6)$$

7. *Entre autres, voir Hastie* et al. *(2009) et James* et al. *(2013).*
8. *C'est le jeu de données Carseats de la bibliothèque ISLR.*

où $\beta_0 = \gamma - \xi$[9]. L'estimation de ce modèle se fait par maximum de vraisemblance en sélectionnant *a priori* une loi paramétrique G. Si on suppose que G est la loi Normale, c'est un modèle probit, si on suppose que G est la loi logistique, c'est un modèle logit. Dans un modèle logit/probit, il y a deux sources potentielles de mauvaise spécification :

- la relation linéaire $\beta_0 + x^T\beta$ est mal spécifiée ;
- la loi paramétrique utilisée G n'est pas la bonne.

En cas de mauvaise spécification, de l'une ou l'autre sorte, l'estimation n'est plus valide. Le modèle le plus flexible est le suivant :

$$\mathbb{P}\left[Y = 1 | X = x\right] = G\left(h\left(x\right)\right) \tag{7}$$

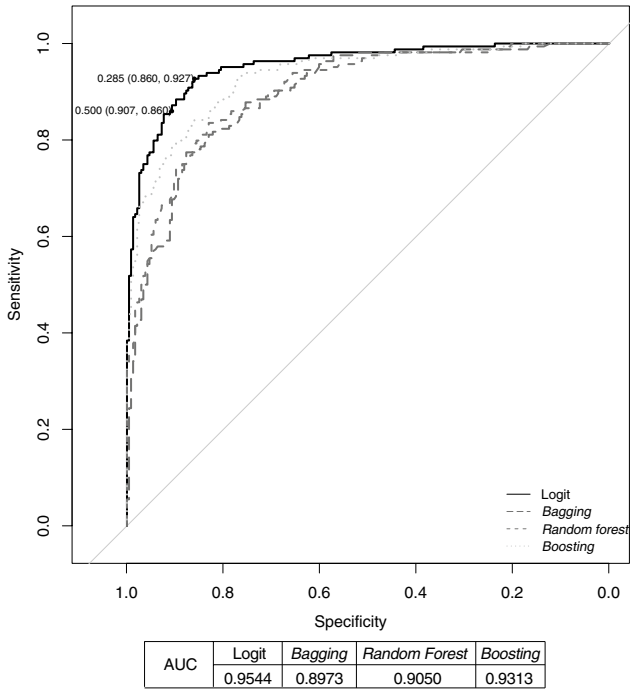
où h est une fonction inconnue et G une fonction de répartition inconnue. Les modèles de *bagging*, de forêt aléatoire et de *boosting* permettent d'estimer ce modèle général sans faire de choix *à priori* sur la fonction h et sur la distribution G. L'estimation du modèle logit/probit est néanmoins plus performante si h et G sont correctement spécifiés.

Nous estimons le modèle (6) avec la loi logistique pour G, et le modèle (7) avec les méthodes de *bagging*, de forêt aléatoire et de *boosting*. Nous faisons une analyse de validation croisée par 10 blocs (encadré 3). Les probabilités individuelles des données *out-of-sample*, c'est-à-dire de chacun des blocs, non-utilisées pour l'estimation, sont utilisées pour évaluer la qualité de la classification.

La figure I présente la courbe ROC, ainsi que l'aire sous la courbe (AUC), pour les estimations logit, *bagging*, forêt aléatoire et *boosting*. La courbe ROC est un graphique qui représente simultanément la qualité de la prévision dans les deux classes, pour des valeurs différentes du seuil utilisé pour classer les individus (on parle de *cutoff*). Une manière naturelle de classer les individus consiste à les attribuer dans la classe pour laquelle ils ont la plus grande probabilité estimée. Dans le cas d'une variable binaire, cela revient à prédire la classe d'appartenance pour laquelle la probabilité estimée est supérieure à 0.5. Mais un autre seuil pourrait être utilisé. Par exemple, dans la figure I, un point de la courbe ROC du modèle logit indique qu'en prenant un seuil égal à 0.5, la réponse

9. $\mathbb{P}[Y=1] = \mathbb{P}[Y^* > \xi] = \mathbb{P}[\gamma + x^T\beta + \varepsilon > \xi] = \mathbb{P}[\varepsilon > \xi - \gamma - x^T\beta]$ *qui peut finallement s'écrire* $\mathbb{P}[\varepsilon < \gamma - \xi + x^T\beta]$. *En posant* $\gamma - \xi = \beta_0$, *on obtient* $\mathbb{P}[Y=1] = G(\beta_0 + x^T\beta)$. *En général, on suppose que le terme d'erreur est de variance* σ^2, *auquel cas les paramètres du modèle (6) deviennent* β_0/σ *et* β/σ, *ce qui veut dire que les paramètres du modèle latent (4) ne sont pas identifiables, ils sont estimés à un paramètre d'échelle près.*

Figure I
Ventes de sièges auto : courbes ROC et aires sous la courbe (AUC)



AUC	Logit	*Bagging*	*Random Forest*	*Boosting*
	0.9544	0.8973	0.9050	0.9313

Source : données simulées de 400 points de vente de siège auto pour bébé avec le jeu de données *carseats* de James *et al.* (2013), https://CRAN.R-project.org/package=ISLR.

ENCADRÉ 3 – **Validation croisée par *k*-blocs**

La validation croisée repose sur l'idée de construire un estimateur en enlevant une observation. Comme on souhaite construire un modèle prédictif, on va comparer la prévision obtenue avec le modèle estimé, et l'observation manquante :

$$\widehat{\mathcal{R}}^{\,\mathrm{CV}} = \frac{1}{n}\sum_{i=1}^{n}\ell\left(y_i, \widehat{m}_{(i)}\left(x_i\right)\right)$$

Le principal problème de cette méthode (dite *leave-one-out*) est qu'elle nécessite de calibrer n modèles, ce qui peut être problématique en grande dimension. Une méthode alternative est la validation croisée par k-blocs (dite *k-fold cross validation*) qui consiste à utiliser une partition de $\{1,\cdots,n\}$ en k groupes (ou blocs) de même taille, $\mathcal{I}_1,\cdots,\mathcal{I}_k$ (notons $\mathcal{I}_{\bar{j}} = \{1,\cdots,n\} \setminus \mathcal{I}_j$). En notant $\widehat{m}_{(j)}$ construit sur l'échantillon $\mathcal{I}_{\bar{j}}$, on pose alors :

$$\widehat{\mathcal{R}}^{\,k-\mathrm{CV}} = \frac{1}{k}\sum_{j=1}^{k}\mathcal{R}_j \text{ où } \mathcal{R}_j = \frac{k}{n}\sum_{i\in\mathcal{I}_j}\ell\left(y_i, \widehat{m}_{(j)}\left(x_i\right)\right)$$

Utiliser $k = 5,10$ a un double avantage par rapport à $k = n$ (correspondant à la méthode *leave-one-out*) : le nombre d'estimations à effectuer est beaucoup plus faible, 5 ou 10 plutôt que n ; les échantillons utilisés pour l'estimation sont moins similaires et donc, moins corrélés les uns aux autres, ce qui tend à éviter les excès de variance (James *et al.,* 2013).

« non » est correctement prédite à 90.7 % (*specificity*), et la réponse « oui » à 86 % (*sensitivity*). Un autre point indique qu'en prenant un seuil égal à 0.285, la réponse « non » est correctement prédite à 86 % (*specificity*), et la réponse « oui » à 92.7 % (*sensitivity*). Comme décrit auparavant, un modèle de classification idéal aurait une courbe ROC de la forme Γ. Le meilleur modèle est celui dont la courbe est au-dessus des autres. Un critère souvent utilisé pour sélectionner le meilleur modèle est celui dont l'aire sous la courbe ROC est la plus grande (AUC). L'avantage d'un tel critère est qu'il est simple à comparer et qu'il ne dépend pas du choix du seuil de classification.

Dans notre exemple, la courbe ROC du modèle logit domine les autres courbes, et son aire sous la courbe est la plus grande (AUC=0.9544). Ces résultats indiquent que ce modèle fournit les meilleures prévisions de classification. N'étant dominé par aucun autre modèle, ce constat suggère que le modèle linéaire logit est correctement spécifié et qu'il n'est pas utile d'utiliser un modèle plus général et plus complexe.

L'achat d'une assurance caravane (classification)

Nous reprenons à nouveau un exemple utilisé dans James *et al.* (2013). Le jeu de données contient 85 variables sur les caractéristiques démographiques de 5 822 individus[10]. La variable dépendante (*purchase*) indique si l'individu a acheté une assurance caravane, c'est une variable binaire, égale à « oui » ou « non ». Dans le jeu de données, seulement 6 % des individus ont pris une telle assurance. Les classes sont donc fortement déséquilibrées.

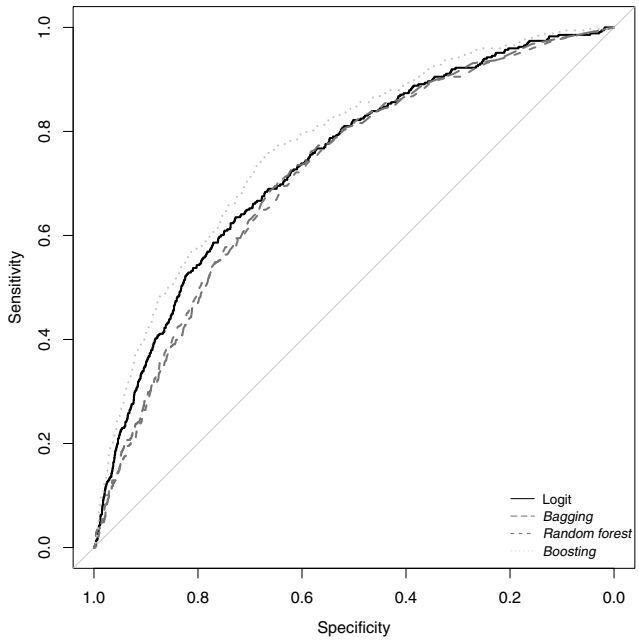
Nous estimons le modèle (6) avec la loi logistique et le modèle (7) avec les méthodes *bagging*, forêt aléatoire et *boosting* (les paramètres de *tuning* sont ceux de James *et al.* (2013), *n.trees* = 1 000 et *shrinkage* = 0.01). Nous faisons une analyse de validation croisée par 10 blocs. Les probabilités individuelles des données *out-of-sample*, c'est-à-dire de chacun des blocs non-utilisée pour l'estimation, sont utilisées pour évaluer la qualité de la classification.

La figure II présente la courbe ROC, ainsi que l'aire sous la courbe (AUC), pour les estimations logit, *bagging*, forêt aléatoire et *boosting*. La courbe du modèle *boosting* domine les autres courbes, son aire sous la courbe est la plus grande (AUC = 0.7691). Ces résultats indiquent que le *boosting* fournit les meilleures prévisions de classification. Comparées à l'exemple précédent, les courbes sont assez éloignées de la forme en coude, ce qui suggère que la classification ne sera pas aussi bonne.

Il faut faire attention aux résultats d'une classification standard, c'est-à-dire avec un seuil de classification égal à 0.5, qui est souvent pris par défaut dans les logiciels (la prédiction de la réponse de l'individu i est « non » si la probabilité estimée qu'il réponde « non » est supérieure à 0.5, sinon c'est « oui »). La partie gauche du tableau 2 présente les taux de classifications correctes avec ce seuil (seuil à 0.5), pour les différentes méthodes. Avec le meilleur modèle et le seuil standard (*boosting* et seuil

10. C'est le jeu de données Caravan de la bibliothèque ISLR sous R.

Figure II
Achat d'assurance: courbes ROC et aires sous la courbe (AUC)



AUC	Logit	*Bagging*	*Random forest*	*Boosting*
	0.7372	0.7198	0.7154	0.7691

Source : jeu de données expérimental *caravan* sur la consommation d'assurance caravane, James *et al.* (2013). https://CRAN.R-project.org/package=ISLR

à 0.5), les réponses « non » sont correctes à 99.87 % et les réponses « oui » sont toutes fausses. Cela équivaut à utiliser un modèle qui prédit que personne ne prend d'assurance caravane. Sélectionner un tel modèle est absurde pour l'analyste, qui est surtout intéressé par les 6 % des individus qui en ont pris une. Ce résultat est dû à la présence de classes fortement déséquilibrées. En effet, en prévoyant que personne n'achète d'assurance, on fait « seulement » 6 % d'erreur. Mais ce sont des erreurs qui conduisent à ne rien expliquer.

Plusieurs méthodes peuvent être utiles pour pallier à ce problème, lié aux classes fortement déséquilibrées (Kuhn & Johnson, 2013, chapitre 16). Une solution simple consiste à utiliser un seuil de classification différent. La courbe ROC présente les résultats en fonction de plusieurs seuils de classification, où la classification parfaite est illustrée par le couple (*specificity*, *sensitivity*) = (1,1), c'est-à-dire par le coin supérieur gauche dans le graphique. On choisit comme seuil de classification optimal celui qui correspond au point de la courbe ROC le plus proche de ce coin. La partie droite du tableau 2 présente les taux de classifications correctes avec les seuils optimaux, pour les différentes méthodes (les seuils optimaux des méthodes logit, *bagging*, forêt aléatoire et

Tableau 2
Achat d'assurance : sensibilité au choix du seuil de classification

	Seuil à 0.5		Seuils optimaux	
	Specificity	Sensitivity	Specificity	Sensitivity
Logit	0.9967	0.0057	0.7278	0.6351
Bagging	0.9779	0.0661	0.6443	0.7069
Random forest	0.9892	0.0316	0.6345	0.6954
Boosting	0.9987	0.0000	0.6860	0.7385

Source : jeu de données expérimental *caravan* sur la consommation d'assurance caravane, James *et al.* (2013). https://CRAN.R-project.org/package=ISLR

boosting sont, respectivement, égaux à 0.0655, 0.0365, 0.0395 et 0.0596). Avec le *boosting* et un seuil optimal, les réponses « non » sont correctes à 68.6 % et les réponses « oui » à 73.85 %. L'objet de l'analyse étant de prévoir correctement les individus susceptibles d'acheter une assurance caravane (classe « oui »), et les distinguer suffisamment des autres (classe « non »), le choix du seuil optimal est beaucoup plus performant que le seuil standard 0.5. Avec un modèle logit et un seuil optimal, le taux de classifications correctes de la classe « non » est de 72.78 %, celui de la classe « oui » de 63.51 %. Par rapport au *boosting*, le logit prédit un peu mieux la classe « non », mais nettement moins bien la classe « oui ».

Les défauts de remboursement de crédits particuliers (classification)

Considérons la base allemande de crédits aux ménages, utilisée dans Nisbet *et al*. (2001) et Tufféry (2001), avec 1 000 observations et 19 variables explicatives, dont 12 qualitatives : en les disjonctant (en créant une variable indicatrice pour chaque modalité), on obtient 48 variables explicatives potentielles. Une question récurrente en modélisation est de savoir quelles sont les variables qui mériteraient d'être utilisées. La réponse la plus naturelle pour un économètre pourrait être une méthode de type *stepwise* (parcourir toutes les combinaisons possibles de variables étant *a priori* un problème trop complexe en grande dimension, *forward* ou *backward*). La suite des variables dans une approche *backward* est présentée dans la première colonne du tableau 3 (voir l'encadré 4 pour les principes de la pénalisation et des choix de variables explicatives). Ce tableau compare avec deux autres approches. Tout d'abord le Lasso, en pénalisant convenablement la norme ℓ_1 du vecteur de paramètres β (dernière colonne). On note que les deux premières variables considérées comme non nulles (pour un λ assez grand) sont les deux premières à ressortir lors d'une procédure *backward*. Enfin, une dernière méthode a été proposée par Breiman (2001b), en utilisant tous les arbres créés lors de la construction d'une forêt aléatoire : l'importance de la

Tableau 3
Crédit : choix de variables, tri séquentiel, par approche *stepwise*, par fonction d'importance dans une forêt aléatoire et par lasso

Stepwise	AIC	Random Forest	Gini	Lasso
checking_statusA14	1112.1730	checking_statusA14	30.818	checking_statusA14
credit_amount(4e+03,Inf]	1090.3467	installment_rate	20.786	credit_amount(4e+03,Inf]
credit_historyA34	1071.8062	residence_since	19.853	credit_historyA34
installment_rate	1056.3428	duration(15,36]	11.377	duration(36,Inf]
purposeA41	1044.1580	credit_historyA34	10.966	credit_historyA31
savingsA65	1033.7521	credit_amount	10.964	savingsA65
purposeA43	1023.4673	existing_credits	10.483	housingA152
housingA152	1015.3619	other_payment_plansA143	10.470	duration(15,36]
other_payment_plansA143	1008.8532	telephoneA192	10.218	purposeA41
personal_statusA93	1001.6574	Age	10.072	installment_rate
savingsA64	996.0108	savingsA65	9.547	property_magnitudeA124
other_partiesA103	991.0377	checking_statusA12	9.502	age(25,Inf]
checking_statusA13	985.9720	housingA152	8.757	checking_statusA13
checking_statusA12	982.9530	jobA173	8.734	purposeA43
employmentA74	980.2228	personal_statusA93	8.716	other_partiesA103
age(25,Inf]	977.9145	property_magnitudeA123	8.634	employmentA72
purposeA42	975.2365	personal_statusA92	8.438	savingsA64
duration(15,36]	972.5094	purposeA43	8.362	employmentA74
duration(36,Inf]	966.7004	employmentA73	8.225	purposeA46
purposeA49	965.1470	employmentA75	8.090	personal_statusA93
purposeA410	963.2713	duration(36,Inf]	8.030	personal_statusA92
credit_historyA31	962.1370	purposeA42	8.026	savingsA63
purposeA48	961.1567	property_magnitudeA122	7.909	telephoneA192

Source: jeu de données crédit de la bibliothèque casdataset de R, crédits aux ménages en Allemagne (Nisbet *et al.*, 2001 ; Tufféry, 2001). http://cas.uqam.ca/

ENCADRÉ 4 – **Pénalisation et méthodes de choix de variables explicatives**

Pour sélectionner les variables explicatives pertinentes en économétrie, on peut utiliser *ex post* des critères de qualité du modèle pénalisant la complexité, en pratique le nombre de variables explicatives (comme le R^2 ajusté ou le critère d'Akaike – AIC – voir le complément en ligne). Dans la méthode dite *forward*, on commence par régresser sur la constante, puis on ajoute une variable à la fois, en retenant celle qui améliore le plus le modèle selon le critère choisi, jusqu'à ce que rajouter une variable diminue la qualité du modèle. Dans la méthode dite *backward*, on commence par régresser sur toutes les variables, puis on enlève une variable à la fois, en retirant celle qui améliore le plus la qualité du modèle, jusqu'à ce que retirer une variable détériore le modèle. Les méthodes dites *stepwise* introduisent les méthodes ensemblistes pour limiter le nombre de tests.

La stratégie en apprentissage machine consiste à pénaliser *ex-ante* dans la fonction objectif, quitte à construire un estimateur biaisé. Typiquement, on va construire :

$$\left(\hat{\beta}_{0,\lambda},\hat{\beta}_{\lambda}\right)=\operatorname{argmin}\left\{\sum_{i=1}^{n}\ell\left(y_i,\beta_0+x^T\beta\right)+\lambda\,\textit{pénalisation}\left(\beta\right)\right\} \quad (9)$$

où la fonction de pénalisation sera souvent une norme $\|\cdot\|$ choisie *a priori*, et un paramètre de pénalisation λ.

variable x_k dans une forêt de T arbres est donnée par :

$$Importance\left(x_k\right)=\frac{1}{T}\sum_{t=1}^{n}\sum_{j\in N_{t,k}}p_t\left(j\right)\Delta\mathcal{I}\left(j\right)$$

où $N_{t,k}$ désigne l'ensemble des nœuds de l'arbre t utilisant la variable x_k comme variable de séparation, $p_t\left(j\right)$ désigne la proportion des observations au nœud j, et $\Delta\left(j\right)$ est la variation d'indice au nœud j (entre le nœud précédant, la feuille de gauche et celle de droite). Dans la colonne centrale du tableau 3 sont présentées les variables par ordre d'importance décroissante, lorsque l'indice utilisé est l'indice d'impureté de Gini.

Avec l'approche *stepwise* et l'approche lasso, on reste sur des modèles logistiques linéaires. Dans le cas des forêts aléatoires (et des arbres), des interactions entre variables peuvent être prises en compte, lorsque 2 variables sont présentes. Par exemple la variable *residence_since* est présente très haut parmi les variables prédictives (troisième variable la plus importante).

Les déterminants des salaires (régression)

Afin d'expliquer les salaires (individuels) en fonction du niveau d'étude, de l'expérience de la personne, et de son sexe, il est classique d'utiliser l'équation de salaire de Mincer (Mincer, 1974 ; Lemieux, 2006) :

$$\log\left(wage\right)=\beta_0+\beta_1\,ed+\beta_2\,exp+\beta_3\,exp^2+\beta_4\,fe+\varepsilon \quad (9)$$

où *ed* est le niveau d'études, *exp* l'expérience professionnelle et *fe* une variable indicatrice égale à 1 si l'individu est une femme. D'après la théorie du capital humain, le salaire espéré augmente avec l'expérience, de moins en moins vite, pour atteindre un maximum avant de diminuer. L'introduction du carré de *exp* permet de prendre en compte une telle relation. La présence de la variable *fe* permet quant à elle de mesurer une éventuelle discrimination salariale entre les hommes et les femmes.

Le modèle (9) impose une relation linéaire entre le salaire et le niveau d'étude, et une relation quadratique entre le salaire et l'expérience professionnelle. Ces relations peuvent paraître trop restrictives. Plusieurs études montrent notamment que le salaire ne diminue pas après un certain âge, et qu'une relation quadratique ou un polynôme de degré plus élevé est plus adapté (Murphy & Welch, 1990 ; Bazen & Charni, 2015).

Le modèle (9) impose également que la différence salariale entre les hommes et les femmes est indépendante du niveau d'étude et de l'expérience. Il est trop restrictif si, par exemple, on suspecte que l'écart de salaire moyen entre les hommes et les femmes est faible pour les postes non-qualifiés et fort pour les postes qualifiés, ou faible en début de carrière et fort en fin de carrière (effets d'interactions). Le modèle le plus flexible est le modèle entièrement non-paramétrique :

$$\log\left(wage\right)=m\left(ed,exp,fe\right)+\varepsilon \quad (10)$$

où $m\left(\cdot\right)$ est une fonction quelconque. Il a l'avantage de pouvoir tenir compte de relations non-linéaires quelconques et d'interactions complexes entre les variables. Mais sa grande flexibilité se fait au détriment d'une interprétation plus difficile du modèle. En effet, il faudrait un graphique en 4 dimensions pour représenter la fonction m. Une solution consiste à représenter la fonction m en 3 dimensions, en fixant la valeur de l'une des variables, mais la fonction

représentée peut être très différente avec une valeur fixée différente.

Nous utilisons les données d'une enquête de l'US Census Bureau de mai 1985, issues de l'ouvrage de Berndt (1990) et disponibles sous R[11]. Nous estimons les deux modèles et utilisons une analyse de validation croisées par 10 blocs pour sélectionner la meilleure approche. Le modèle paramétrique (9) est estimé par moindres carrés ordinaires (MCO). Le modèle entièrement non-paramétrique (10) est estimé par la méthode des *splines*, car il en comprend peu de variables, ainsi que par les méthodes *bagging*, *random forest* et *boosting*.

Le tableau 4 présente les résultats de la validation croisée en 10 blocs (*10-fold cross-validation*). Le meilleur modèle est celui qui minimise le critère $\widehat{\mathcal{R}}^{10-CV}$. Les résultats montrent que le modèle (9) est au moins aussi performant que le modèle (10), ce qui suggère que le modèle paramétrique (9) est correctement spécifié.

Les déterminants des prix des logements à Boston (régression)

Nous reprenons ici l'un des exemples utilisé dans James *et al.* (2013), dont les données sont disponibles sous R. Le jeu de données[12] contient les valeurs médianes des prix des maisons (*medv*) dans $n = 506$ quartiers autour de Boston, ainsi que 13 autres variables, dont le nombre moyen de pièces par maison (*rm*), l'âge moyen des maisons (*age*) et le pourcentage de ménages dont la catégorie socio-professionnelle est peu élevée (*lstat*).

Considérons le modèle de régression linéaire suivant :

$$medv = \alpha + x^T \beta + \varepsilon \tag{11}$$

où x = [chas,nox,age,tax,indus,rad,dis,lstat,crim, black,rm,zn,ptratio] est un vecteur en dimension 13 et β est un vecteur de 13 paramètres. Ce modèle spécifie une relation linéaire entre la valeur des maisons et chacune des variables explicatives. Le modèle le plus flexible est le modèle entièrement non-paramétrique :

$$medv = m(x) + \varepsilon \tag{12}$$

L'estimation de ce modèle avec les méthodes du noyau ou les splines peut être problématique, car le nombre de variables est relativement élevé (il y a ici 13 variables), ou au moins trop élevé pour envisager estimer une surface en dimension 13. Nous estimons les deux modèles et utilisons une analyse de validation croisée par 10 blocs pour sélectionner la meilleure approche. Le modèle paramétrique (11) est estimé par moindres carrés ordinaires (MCO) et le modèle entièrement non-paramétrique (12) est estimé par trois méthodes différentes : *bagging*, forêt aléatoire et *boosting* (nous utilisons ici les valeurs par défaut utilisées dans James *et al.*, 2013, pp. 328–331).

Le tableau 5 présente les résultats de la validation croisée en 10 blocs. À partir des résultats *in-sample* (sur les données d'apprentissage), les méthodes de *bagging* et de forêt aléatoire paraissent incroyablement plus performantes que l'estimation MCO du modèle linéaire (11), le critère $\widehat{\mathcal{R}}^{10-CV}$ passant de 21.782 à 1.867 et 1.849. Les résultats *out-of-sample* (sur d'autres données que celles servant à estimer le modèle) vont dans le même sens, mais la différence est moins importante, le critère $\widehat{\mathcal{R}}^{10-CV}$ passant de 24.082 à 9.59 et 9.407. Ces résultats illustrent un phénomène classique des méthodes non-linéaires, comme le *bagging* et la forêt aléatoire, qui peuvent être très performantes pour prédire les données utilisées pour l'estimation, mais moins performantes pour prédire des données hors-échantillon. C'est pourquoi la sélection de la meilleure

11. Jeu de données CPS1985 de la bibliothèque AER.

12. Jeu de données Boston de la librairie MASS. Pour une description complète des données, voir : https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/Boston.html.

Tableau 4
Salaire : analyse de validation croisée par blocs (*K* = 10) : performances de l'estimation des modèles linéaire (9) et entièrement non-paramétrique (10)

$\widehat{\mathcal{R}}^{10-\mathrm{CV}}$	Modèle (9)	Modèle (10)			
	MCO	*Splines*	*Bagging*	Forêts aléatoires	*Boosting*
Out-of-sample	0.2006	0.2004	0.2762	0.2160	0.2173

Source: recensement de la population, États-Unis, 1985 ; Berndt (1990), jeu de données CPS1985 de la bibliothèque AER. https://rdrr.io/cran/AER/man/CPS1985.html

Tableau 5
Prix des logements à Boston – Analyse de validation croisée par blocs (*K* = 10) : performances de l'estimation des modèles linéaire (11) et entièrement non-paramétrique (12)

$\widehat{\mathcal{R}}^{10-\mathrm{CV}}$	Modèle (11)	Modèle (12)		
	MCO	*Splines*	Forêts aléatoires	*Boosting*
In-sample	21.782	1.867	1.849	7.012
Out-of-sample	24.082	9.590	9.407	11.789

Champ : quartiers de l'agglomération de Boston.
Source: James *et al.* (2013), jeu de données Boston de la librairie MASS. https://stat.ethz.ch/R-manual/Rdevel/library/MASS/html/Boston.html

estimation est habituellement basée sur une analyse *out-of-sample*.

La différence entre l'estimation des modèles (11) et (12) est importante. Un tel écart suggère que le modèle linéaire est mal spécifié, et que des relations non-linéaire et/ou des effets d'interactions sont présentes dans la relation entre le prix des logements et les variables explicatives x. Ceci conduit à chercher une meilleure spécification paramétrique. À partir du modèle paramétrique (11), et afin de prendre en compte d'éventuelles non-linéarités, le modèle additif généralisé (GAM) suivant peut être considéré :

$$medv = m_1(x_1) + m_2(x_2) + \ldots + m_{13}(x_{13}) + \varepsilon, \quad (13)$$

où $m_1, m_2, \ldots m_{13}$ sont des fonctions inconnues. L'avantage de ce modèle est qu'il permet de considérer n'importe quelle relation non-linéaire entre la variable dépendante et chacune des variables explicatives. De plus, il ne souffre pas de « la malédiction » de la dimension, car chacune des fonctions est de dimension 1, et il est facilement interprétable. Toutefois, il ne prend pas en compte d'éventuels effets d'interactions. L'estimation du modèle additif généralisé (13) par la méthode des *splines*, dans le cadre d'une analyse de validation croisée par 10 blocs, donne une valeur $\widehat{\mathcal{R}}^{10-CV} = 13.643$. Par rapport au modèle paramétrique (11), il y a un gain important (13.643 vs. 24.082). Mais la différence avec le modèle entièrement non-paramétrique (12) reste conséquente (13.643 vs 9.590, 9.407, 11.789). Une telle différence suggère que la prise en compte de relations individuelles pouvant être fortement non-linéaires n'est pas suffisante, et que des effets d'interactions entre les variables sont présents. Nous pourrions inclure dans le modèle les variables d'interactions les plus simples entre toutes les paires de variables ($x_i \times x_j$), mais cela impliquerait de rajouter un très grand nombre de variables au modèle initial (78 dans notre cas), qui ne serait pas sans conséquence sur la qualité de l'estimation du modèle. Quoi qu'il en soit, nous pouvons dire pour le moment que le modèle linéaire est mal spécifié et qu'il existe des effets d'interactions pouvant être forts dans la relation entre *medv* et X, l'identification de tels effets restant délicat.

Afin d'aller plus loin, les outils développés en apprentissage statistique peuvent être à nouveau d'un grand recours. Par exemple, l'estimation de forêt aléatoire s'accompagne de mesures de l'importance de chacune des variables dans l'estimation du modèle. Le tableau 6 présente ces mesures dans le cadre du modèle (12), estimé sur l'échantillon complet. Les résultats suggèrent que les variables *rm* et *lstat* sont les variables les plus importantes

Tableau 6
Prix des logements : mesures de l'importance de chacune des variables dans l'estimation de forêt aléatoire du modèle (12), en considérant tout l'échantillon

	% IncMSE	IncNodePurity
rm	61.35	18 345.41
lstat	36.20	15 618.22
dis	29.37	2601.72
nox	24.91	1034.71
age	17.86	554.50
ptratio	17.43	626.58
tax	16.60	611.37
crim	16.26	1701.73
indus	9.45	237.35
black	8.72	457.58
rad	4.53	166.72
zn	3.10	35.73
chas	0.87	39.05

Champ : quartiers de l'agglomération de Boston.
Source : James *et al.* (2013), jeu de données Boston de la librairie MASS. https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/Boston.html

pour expliquer les variations des prix des logements *medv*. Ce constat nous conduit à enrichir la relation initiale, en rajoutant les d'interactions liées à ces deux variables seulement, qui sont les plus importantes.

Nous estimons le modèle additif généralisé incluant les variables d'intéractions, sur l'échantillon complet :

$$medv = m_1\left(x_1\right) + \cdots + m_{13}\left(x_{13}\right) + \left(rm : x\right)\gamma + \left(lstat : x\right)\delta + \varepsilon, \quad (14)$$

où $\left(rm : x\right)$ représente les variables d'interactions de *rm* avec toutes les autres variables de x et $\left(lstat : x\right)$ représente les variables d'interactions de lstat avec toutes les autres variables de x[13]. L'analyse des résultats de cette estimation suggère que les fonctions $\widehat{m}_i$ sont linéaires pour toutes les variables, sauf pour la variable dis, dont la relation estimée est présentée dans la figure III. Cette variable mesure la distance moyenne à cinq centres d'emplois de la région. L'effet semble diminuer plus rapidement avec la distance, lorsque celle-ci n'est pas très élevée. Au-delà d'une certaine distance (au-delà de 2, en log), l'effet est réduit, il continue à diminuer mais plus doucement. Cette relation non-linéaire peut être approchée par une régression linéaire par morceaux en considérant un nœud.

Finalement, l'analyse précédente nous conduit à considérer le modèle linéaire suivant :

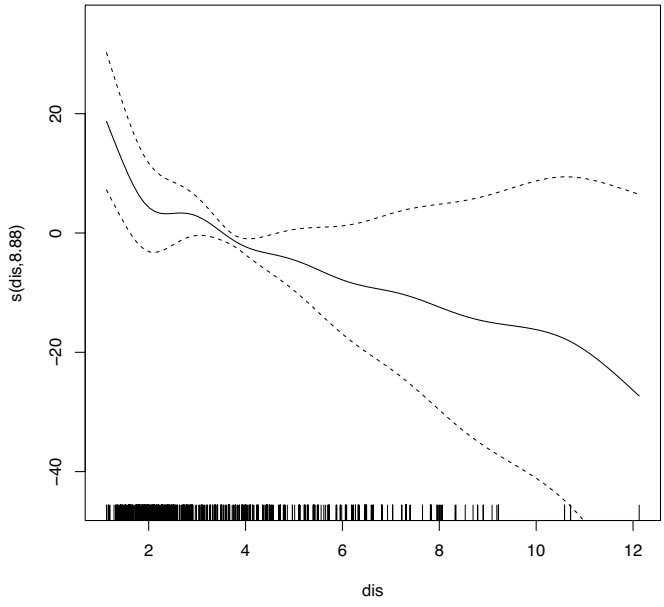
$$medv = \alpha + x^T\beta + (dis - 2) + \theta + \left(rm : x\right)\gamma + \left(lstat : x\right)\delta + \varepsilon \quad (15)$$

où $(dis - 2)$ est égal à la valeur de son argument si ce dernier est positif, et à 0 sinon. Par rapport au modèle linéaire initial, ce modèle inclut une relation linéaire par morceaux avec la variable dis, ainsi que des effets d'interactions entre *rm*, *lstat* et chacune des autres variables de x.

Le tableau 7 présente les résultats de la validation croisée en 10 blocs (*10-fold cross-validation*) de l'estimation des modèles paramétriques (11) et (15), estimés par moindres carrés ordinaires (MCO), et du modèle additif généralisé (14) estimé par les *splines*. Il montre que l'ajout des variables d'interactions et de la relation linéaire par morceaux dans le modèle (15) donne des résultats beaucoup plus performants que le modèle initial (11) : le critère $\widehat{\mathcal{R}}^{10-CV}$ est divisé par plus de deux, il passe de 24.082 à 11.759. En comparant ces résultats

13. On a $(rm:x)=[rm\times chas, rm\times nox, rm\times age, rm\times tax, rm\times indus, rm\times rad, rm\times dis, rm\times lstat, rm\times crim, rm\times black, rm\times zn, rm\times ptratio]$ et $(lstat:x)=[lstat\times chas, lstat\times nox, lstat\times age, lstat\times tax, lstat\times indus, lstat\times rad, lstat\times dis, lstat\times crim, lstat\times black, lstat\times zn, lstat\times ptratio]$.

Figure III
Estimation de la relation $m_7(x_7)$ dans le modèle additif généralisé (14), où x_7 = dis



Note : estimation de la relation m7 (x7) pour la variable dis dans le modèle additif généralisé; les lignes en pointillé correspondent aux intervalles de confiance à 95 %.
Champ : quartiers de l'agglomération de Boston.
Source : James *et al.* (2013), jeu de données Boston de la librairie MASS. https://stat.ethz.ch/R-manual/Rdevel/library/MASS/html/Boston.html

Tableau 7
Prix des logements à Boston - Analyse de validation croisée par blocs (*K* = 10) : performances de l'estimation des modèles linéaires (11) et (15) et du modèle (14) incluant les effets d'interactions et une non-linéarité par morceaux

$\widehat{\mathcal{R}}^{10-\mathrm{CV}}$	Modèle (11)	Modèle (14)	Modèle (15)
	MCO	*Splines*	MCO
Out-of-sample	24.082	13.643	11.759

Champ : quartiers de l'agglomération de Boston.
Source : James *et al.* (2013), jeu de données Boston de la librairie MASS. https://stat.ethz.ch/R-manual/Rdevel/library/MASS/html/Boston.html

avec ceux du tableau 5, on constate également que le modèle paramétrique (15), estimé par MCO, est aussi performant que le modèle général (12) estimé par *boosting* ($\widehat{\mathcal{R}}^{10-CV} = 11.789$). La différence avec les méthodes *bagging* et forêt aléatoire n'est quant à elle pas très importante ($\widehat{\mathcal{R}}^{10-CV} = 9.59, 9.407$). Finalement, les méthodes *bagging*, forêt aléatoire et *boosting* ont permis de mettre en évidence une mauvaise spécification du modèle paramétrique initial, puis de trouver un modèle paramétrique beaucoup plus performant, en prenant compte des effets de non-linéarités et d'interactions appropriées.

* *
*

Si les deux cultures (ou les deux communautés) de l'économétrie et de l'apprentissage automatique se sont développées en parallèle, le nombre de passerelles entre les deux ne cesse d'augmenter. Alors que Varian (2014) présentait les apports importants de l'économétrie à la communauté de l'apprentissage automatique, nous avons tenté ici de présenter des concepts et des outils développés au fil du temps par ces derniers, qui pourraient être utiles aux économètres, dans un contexte d'explosion du volume de données. Les fondements probabilistes de l'économétrie sont incontestablement sa force, avec non seulement une interprétabilité des modèles, mais aussi une quantification de l'incertitude. Néanmoins, les performances prédictives des modèles d'apprentissage automatique sont intéressantes, car elles permettent de repérer une mauvaise spécification d'un modèle économétrique. De la même manière que les techniques non-paramétriques permettent d'avoir un point de référence pour juger de la pertinence d'un modèle paramétrique, les outils d'apprentissage automatique permettent d'améliorer un modèle économétrique, en détectant un effet non-linéaire ou un effet croisé oublié.

Une illustration des interactions possibles entre les deux communautés se trouve par exemple dans Belloni *et al.* (2010 ; 2012), dans un contexte de choix d'instrument dans une régression. Reprenant les données de Angrist & Krueger (1991) sur un problème de réussite scolaire, ils montrent comment mettre en œuvre efficacement les techniques d'économétrie instrumentale quand on peut choisir parmi 1 530 instruments disponibles (problème qui deviendra récurrent avec l'augmentation du volume de données). Comme nous l'avons vu tout au long de cet article, même si les approches peuvent être fondamentalement différentes dans les deux communautés, bon nombre d'outils développés par la communauté de l'apprentissage automatique méritent d'être utilisés par les économètres. □

Lien vers les compléments en ligne : https://www.insee.fr/fr/statistiques/fichier/3706230?sommaire=3706255/505-506_Charpentier-Flachaire-Ly_complement.pdf

BIBLIOGRAPHIE

Aldrich, J. (2010). The Econometricians' Statisticians, 1895-1945. *History of Political Economy*, 42(1), 111–154.
https://doi.org/10.1215/00182702-2009-064

Altman, E., Marco, G. & Varetto, F. (1994). Corporate distress diagnosis: Comparisons using linear discriminant analysis and neural networks (the Italian experience). *Journal of Banking & Finance*, 18(3), 505–529.
https://doi.org/10.1016/0378-4266(94)90007-8

Angrist, J. D. & Krueger, A. B. (1991). Does Compulsory School Attendance Affect Schooling and Earnings? *The Quarterly Journal of Economics*, 106(4**)**, 979–1014.
https://doi.org/10.2307/2937954

Bazen, S. & Charni, K. (2017). Do earnings really decline for older workers? I*nternational Journal of Manpower*, 38(1), 4 -24.
https://doi.org/10.1108/IJM-02-2016-0043

Bellman, R. E. (1957). *Dynamic Programming*. Princeton, NJ: Princeton University Press.

Belloni, A., Chernozhukov, V. & Hansen, C. (2010). Inference for High-Dimensional Sparse Econometric Models. *Advances in Economics and Econometrics*, *10th World Congress of Econometric Society*, 245–295
https://doi.org/10.1017/CBO9781139060035.008

Belloni, A., Chen, D., Chernozhukov, V. & Hansen, C. (2012). Sparse Models and Methods for Optimal Instruments With an Application to Eminent Domain. *Econometrica*, 80(6), 2369–2429.
https://doi.org/10.3982/ECTA9626

Blanco, A. Pino-Mejias, M., Lara, J. & Rayo, S. (2013). Credit scoring models for the microfinance industry using neural networks: Evidence from Peru. *Expert Systems with Applications*, 40(1), 356–364.
https://doi.org/10.1016/j.eswa.2012.07.051

Breiman, L. Fiedman, J., Olshen, R. A. & Stone, C. J. (1984). *Classification And Regression Trees*. Chapman & Hall/CRC Press Online.
https://www.stat.wisc.edu/~loh/treeprogs/guide/wires11.pdf

Breiman, L. (2001a). Statistical Modeling: The Two Cultures. *Statistical Science*, 16(3), 199–231.
https://doi.org/10.1214/ss/1009213726

Breiman, L. (2001b). Random forests. *Machine learning*, 45(1), 5–32.
https://doi.org/10.1023/A:1010933404324

Bühlmann, P. & van de Geer, S. (2011). *Statistics for High Dimensional Data: Methods, Theory and Applications*. Berlin: Springer Verlag.
https://doi.org/10.1007/978-3-642-20192-9

Cortes, C. & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, 20(3), 273–297.
https://doi.org/10.1023/A:1022627411411

Grandvalet, Y., Mariéthoz, J., & Bengio, S. (2005). Interpretation of SVMs with an Application to Unbalanced Classification. *Advances in Neural Information Processing Systems* N° 18.
https://papers.nips.cc/paper/2763-a-probabilistic-interpretation-of-svms-with-an-application-to-unbalanced-classification.pdf

Groves, T. & Rothenberg, T. (1969). A note on the expected value of an inverse matrix. *Biometrika*, 56(3), 690–691.
https://doi.org/10.1093/biomet/56.3.690

Hastie, T., Tibshirani, R. & Friedman, J. (2009). *The Elements of Statistical Learning*. New York: Springer Verlag.
https://doi.org/10.1007/978-0-387-84858-7

Hebb, D. O. (1949). *The organization of behavior*. New York: Wiley.
https://doi.org/10.1002/1097-4679(195007)6:3<307::AID-JCLP2270060338>3.0.CO;2-K

James, G., D. Witten, T. Hastie, & Tibshirani, R. (2013). An Introduction to Statistical Learning. *Springer Texts in Statistics* 103.
https://doi.org/10.1007/978-1-4614-7138-7

Khashman, A. (2011). Credit risk evaluation using neural networks: Emotional versus conventional models. *Applied Soft Computing*, 11(8), 5477–5484.
https://doi.org/10.1016/j.asoc.2011.05.011

Kolda, T. G. & Bader, B. W. (2009). Tensor Decompositions and Applications. *SIAM Review*, 51(3), 455–500.
https://doi.org/10.1137/07070111X

Kuhn, M. & Johnson, K. (2013). *Applied Predictive Modeling*. New York: Springer Verlag.
https://doi.org/10.1007/978-1-4614-6849-3

Landis, J. R. & Koch, G.G. (1977). The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1), 159–174.
https://doi.org/10.2307/2529310

LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep learning. *Nature*, **521,** 436–444.
https://doi.org/10.1038/nature14539

Leeb, H. (2008). Evaluation and selection of models for out-of-sample prediction when the sample size is small relative to the complexity of the data-generating process. *Bernoulli*, 14(3), 661–690.
https://doi.org/10.3150/08-BEJ127

Lemieux, T. (2006). The "Mincer Equation" Thirty Years After Schooling, Experience, and Earnings. In: Grossbard, S. (Ed.), *Jacob Mincer: A Pioneer of Modern Labor Economics*, pp. 127–145. Boston, MA: Springer Verlag.
https://doi.org/10.1007/0-387-29175-X_11

Lin, H. W., Tegmark, M. & Rolnick, D. (2016). Why does deep and cheap learning work so well?
https://arxiv.org/abs/1608.08225

Mincer, J. (1974). Schooling, Experience and Earnings. New York: NBER.
https://www.nber.org/books/minc74-1

Morgan, J. N. & Sonquist, J. A. (1963). Problems in the Analysis of Survey Data, and a Proposal. *Journal of the American Statistical Association*, 58(302), 415–434.
https://doi.org/10.1080/01621459.1963.10500855

Morgan, M. S. (1990). *The history of econometric ideas*. Cambridge, UK: Cambridge University Press.

Murphy, K. M. & Welch, F. (1990). Empirical Age-Earnings Profiles. *Journal of Labor Economics*, 8(2), 202–229.
https://doi.org/10.1086/298220

Nisbet, R., Elder, J. & Miner, G. (2011). *Handbook of Statistical Analysis and Data Mining Applications*. New York: Academic Press.

Portnoy, S. (1988). Asymptotic Behavior of Likelihood Methods for Exponential Families when the Number of Parameters Tends to Infinity. *Annals of Statistics*, 16(1), 356–366.
https://doi.org/10.1214/aos/1176350710

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1), 81–106.
https://doi.org/10.1007/BF00116251

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 386–408.
https://doi.org/10.1037/h0042519

Samuel, A. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3(3), 210–229.
https://doi.org/10.1147/rd.33.0210

Shalev-Shwartz, S. & Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge, UK: Cambridge University Press.

Shapire, R. E. & Freund, Y. (2012). *Boosting*. *Fondations and Algorithms*. Cambridge, A MIT Press.

Tam, K. Y. & Kiang, M. Y. (1992). Managerial applications of neural networks: The case of bank failure predictions. *Management Science*, 38, 926–947.
https://doi.org/10.1287/mnsc.38.7.926

Tufféry, S. (2001). *Data Mining and Statistics for Decision Making*. Hoboken, NJ: Wiley.

Varian, H. R. (2014). Big Data: New Tricks for Econometrics. *Journal of Economic Perspectives*, 28(2), 3–28.
https://doi.org/10.1257/jep.28.2.3

Vert, J. P. (2017). Machine learning in computational biology. Cours à l'Ensae ParisTech.
http://members.cbio.mines-paristech.fr/~jvert/teaching/

Widrow, B. & Hoff, M. E. Jr. (1960). Adaptive Switching Circuits. *IRE WESCON Convention Record*, 4, 96–104.
https://apps.dtic.mil/dtic/tr/fulltext/u2/241531.pdf

Zinkevich M. A., Weimer, M., Smola, A. & Li, L. (2010). Parallelized Stochastic Gradient Descent. *Advances in neural information processing systems* 23, 2595–2603.
https://papers.nips.cc/paper/4006-parallelized-stochastic-gradient-descent.pdf