Introduction au Machine Learning
Formalisation

Maxime Jumelle

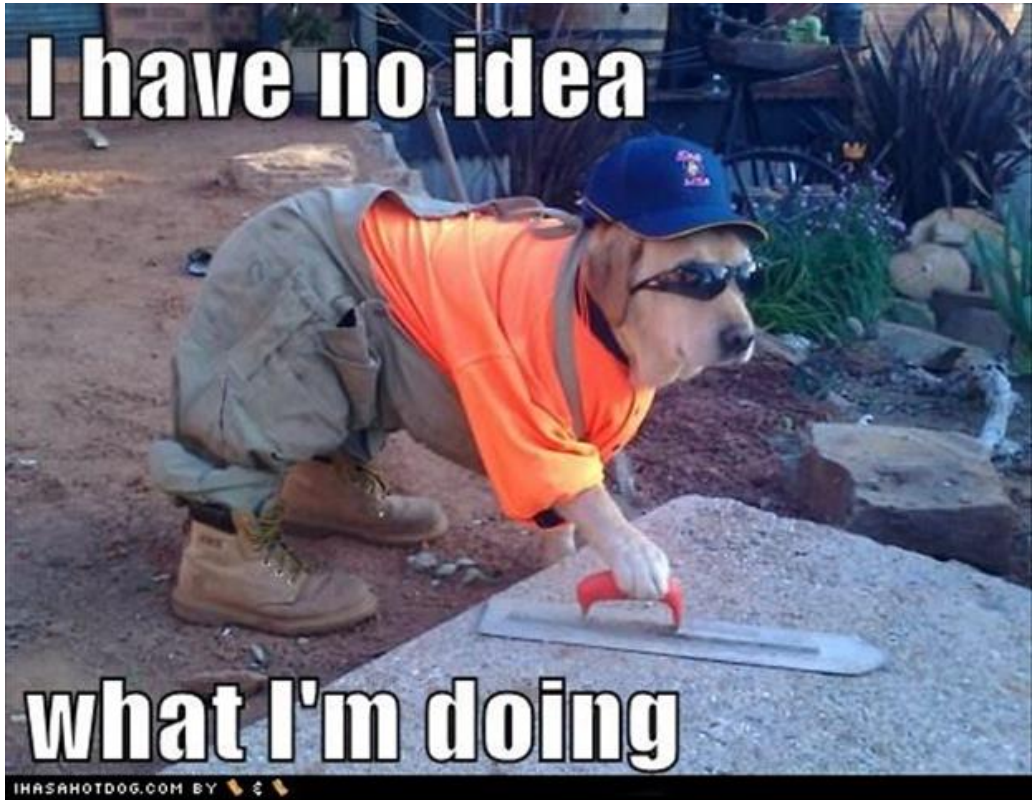
ESLSCA Big Data - MBA 2

2019 - 2020

Cette première introduction au Machine Learning présente plusieurs objectifs.

- Introduire les concepts élémentaires du Machine Learning et un formalisme rigoureux de l'apprentissage supervisé.
- Comprendre les algorithmes classiques d'apprentissage supervisé et les mettre en œuvre sous Python ou R.
- Prendre des décisions quant à l'utilisation de certains modèles et adopter un regard critique sur les résultats.
- Avoir connaissance des différentes problématiques (techniques et métiers) qui surviennent lors de la modélisation.

Ce module adopte une vision **pragmatique** du Machine Learning et se veut prudent quand à l'interprétation et l'utilisation des modèles. En clair, il ne faut pas tomber dans le piège qui consiste à faire une soupe d'algorithmes et choisir uniquement celui qui possède la meilleure performance.



Plan du cours

- Fondements de l'apprentissage supervisé
- Modèles linéaires
- Optimisation numérique : application au perceptron
- Arbres de décision
- *Ensemble Learning* à base d'arbres
- Transparence des algorithmes

Pré-requis

Afin de suivre le cours sans problème, il est fortement conseillé de connaître les outils suivants :

- **Analyse vectorielle** : fonction vectorielle, gradient et dérivées partielles.
- **Probabilités** : espérance, variance, loi des grands nombres et TCL.
- **Statistique** : distribution, marginales, techniques d'inférences (log-vraisemblance).
- **Programmation** : fonctions, traitement de données.

Génèse

Qu'est-ce que le **Machine Learning** (*apprentissage statistique* ou *automatique* en français) ?

> *L'apprentissage automatique (en anglais machine learning, littéralement « l'apprentissage machine ») ou apprentissage statistique est un champ d'étude de l'intelligence artificielle qui se fonde sur des approches statistiques pour donner aux ordinateurs la capacité d' « apprendre » à partir de données, c'est-à-dire d'améliorer leurs performances à résoudre des tâches sans être explicitement programmés pour chacune. Plus largement, cela concerne la conception, l'analyse, le développement et l'implémentation de telles méthodes. (Page Wikipédia)*

Génèse

Lorsque l'on fait du Machine Learning, on parle très souvent de **modèle statistique/mathématique**. Un modèle statistique est une fonction mathématique f qui est construite **selon les règles qu'un algorithme détermine automatiquement**. Nous avons besoin

- de données (qui peuvent être structurées ou non-structurées).
- d'hypothèses pour choisir un algorithme adéquat.
- des connaissances métier pour orienter les décisions et interpréter les résultats.

Très souvent, on note $\mathbf{x}$ une observation (un individu, un objet, ...), et $\hat{f}$ le modèle **qui a été construit**. Il est important de bien différencier f et $\hat{f}$: $\hat{f}(\mathbf{x})$ est la **prédiction** de cette observation $\mathbf{x}$.

Génèse

Il existe plusieurs familles d'algorithmes selon le problème étudié pour fournir une solution adaptée.

- **Classification** : assigner une catégorie à chaque observation.
- **Labellisation** : assigner aucune, une ou plusieurs étiquettes à chaque observation.
- **Régresion** : prédire une quantité à chaque observation.
- **Clustering** : partitionner les observations selon des régions homogènes.
- **Réduction de dimension** : projeter les observations dans un espace de plus petite dimension.
- **Détection d'anomalies** : détecter et prévenir les anomalies dans les observations.
- **Génération de données** : échantillonner des observations depuis une distribution.

Génèse

Parmi les algorithmes de Machine Learning, il existe principalement deux classes d'algorithmes.

- Les **algorithmes supervisés** : nous disposons au préalable de données **labélisées** $(\mathbf{x}_i, y_i)_{1 \leq i \leq n}$, où les y_i sont les *labels* des individus (c'est-à-dire les réponses observées), et l'on cherche à construire $\hat{f}$ de sorte que, pour chaque observation $\mathbf{x_i}$, $\hat{f}(\mathbf{x_i})$ soit le plus proche possible de y_i.
- Les **algorithmes non supervisés** : cette fois-ci, nous n'avons pas connaissance des y_i, et l'on souhaite de modéliser des phénomènes présents dans les données pour aboutir à des prédictions $\hat{f}(\mathbf{x_i})$ cohérentes.

Cette année, nous nous concentrerons uniquement sur les algorithmes **supervisés**.

Vocabulaire

Le monde du Machine Learning a son vocabulaire bien à lui. En l'occurrence, vous risquerez de rencontrer les termes suivants au fur et à mesure des cours :

- **Entraîner** (ou *fit*) c'est construire la fonction $\hat{f}$ sur la base des observations $\mathbf{x}$ et des hypothèses choisies.
- **Prédire** (ou *predict*) c'est calculer $\hat{f}(\mathbf{x})$ pour un $\mathbf{x}$ donné.

Sommaire

Formalisation

Il est supposé exister un modèle f^* dit théorique tel que pour toute observation $\mathbf{x}$, $f^*(\mathbf{x}) = y$. En pratique, il n'est pas possible d'obtenir ce modèle théorique, et la principale tâche de l'apprentissage supervisé consiste alors à construire un modèle $\hat{f}$ qui approxime f^* de telle sorte que pour toute observation $\mathbf{x}$, $\hat{f}(\mathbf{x}) = \hat{y} \approx y$ (dont un formalisme plus rigoureux sera détaillé par la suite).

Exemple : diagnostic du cancer du sein

Le tableau ci-dessous est un exemple de données tabulaires où l'apprentissage supervisé est particulièrement efficace. Cet extrait du jeu de données *Breast Cancer Wisconsin* référence des informations quantitatives extraites depuis des images de biopsies du sein, en associant la variable `Diagnostic` indiquant si le tissu prélevé provient d'un cancer malin (`M`) ou bénin (`B`).

Perimeter	Area	Smoothness	Radius	Diagnostic
122.8	1001	0.1184	17.99	M
94.74	684.5	0.09867	14.68	M
85.63	520	0.1075	13.08	B

Le problème abordé ici est un problème de **classification binaire** : la variable réponse à prédire est discrète et ne peux prendre que deux valeurs (0 pour `B` et 1 pour `M`). Le modèle f à construire permettra alors, en fonction des caractéristiques du tissu prélevé (et donc de la connaissance des variables `Perimeter`, `Area`, etc), de prédire la variable `Diagnostic` en calculant $f(\mathbf{x})$ pour un individu $\mathbf{x}$.

Exemple : catégorisation d'images

La base de données d'images CIFAR-10, très connue et régulièrement utilisée comme *benchmarking* par les algorithmes d'apprentissage supervisé pour la vision par ordinateur, représente des objets du quotidien. Chaque image est représentée par une matrice de 32×32 pixels, chaque pixel étant encodé sur un octet (c'est-à-dire une liste à 3 valeurs allant de 0 à 255 pour les composantes rouge, verte et bleue). Chaque image est associée à une classe parmi les 10 existantes dans le jeu de données. Certaines de ces images sont représentées sur la figure 4.18. Le problème considéré ici est un problème de classification multiple, puisque la variable à prédire est discrète et peut prendre plusieurs valeurs (de 0 à 9 pour chaque catégorie). Un modèle d'apprentissage supervisé f a donc pour objectif d'associer, pour une image $\mathbf{x}$ donnée, un catégorie $f(\mathbf{x})$. Dans cette configuration, le modèle prédit une distribution de probabilités pour chaque classe.

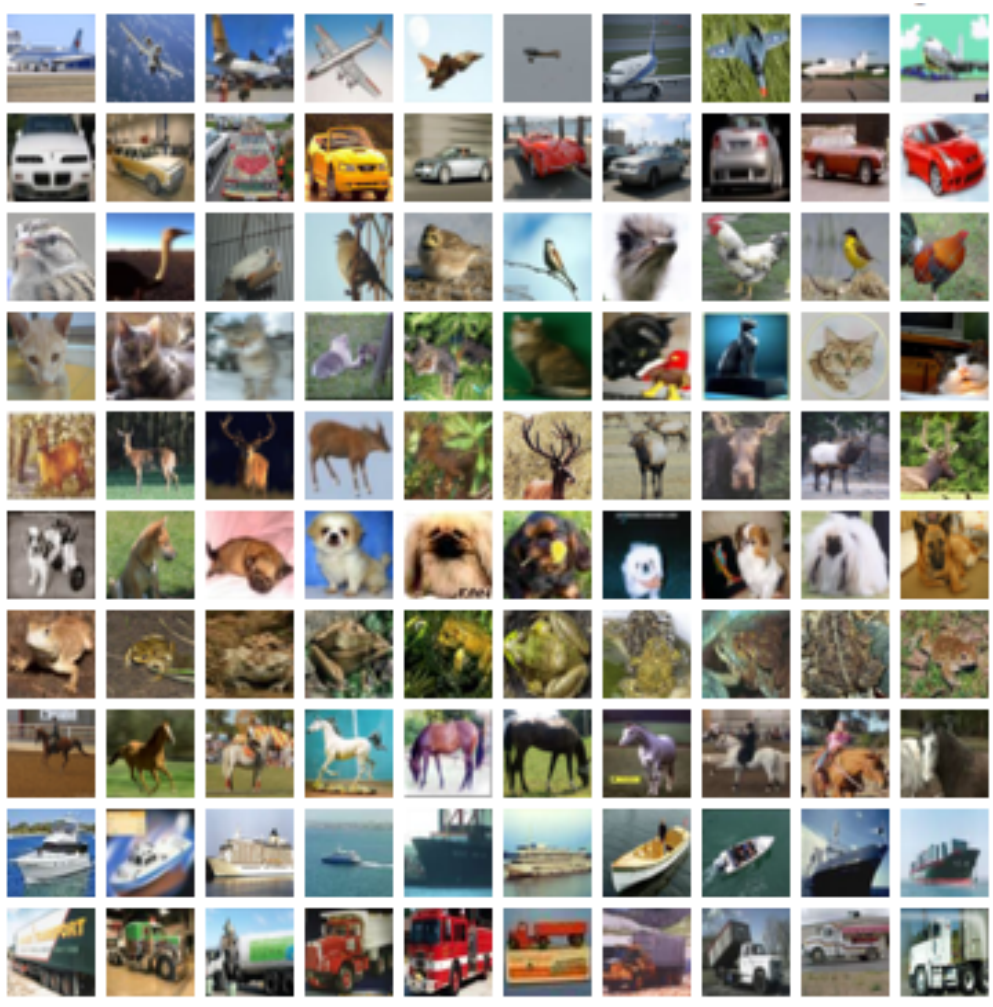


FIGURE – Exemples d'images tirées depuis le jeu de données CIFAR-10. Chaque ligne représente des images appartenant à une même catégorie.

Format des données

Les individus que nous étudions sont représentés par une matrice X de taille $n \times p$, où n est le nombre d'observations et p le nombre de variables. Chaque individu $\mathbf{x}$ vit dans un espace $\mathcal{X}$ (attention à la calligraphie) de dimension p. En pratique, on considérera que $\mathcal{X} = \mathbb{R}^d$ (où d n'est pas nécessaire égal à p en fonction des hypothèses d'encodage). Dans la littérature scientifique, cet espace est très souvent appelé **espace des caractéristiques** (ou *feature space*). Les réponses y vivent dans un espace $\mathcal{Y}$, généralement $\mathbb{R}$ pour un problème de régression ou $\{0, 1\}^K$ pour un problème de classification à K classes.

$$X = \begin{pmatrix} \mathbf{x}_{11} & \dots & \mathbf{x}_{1p} \\ \vdots & \ddots & \vdots \\ \mathbf{x}_{n1} & \dots & \mathbf{x}_{np} \end{pmatrix} \qquad Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Par simplicité, on notera $\mathbf{x}_{\bullet,j}$ ou, plus simplement, x_j la j-ème variable de n'importe quel individu (attention aussi à la police de caractères).

Résumons toutes ces notations introduites :

- X c'est la base de variables explicatives dont nous disposons.
- X_j est la j-ème variable de la distribution jointe $(X_1, \ldots, X_p)$.
- $\mathbf{x}_i$ est le i-ème individu de la base.
- $\mathbf{x}_{ij}$ est la j-ème variable du i-ème individu de la base.
- x_j est la j-ème variable de n'importe quel individu.

Dans tous le cours, nous utiliserons souvent l'indice i pour les individus et l'indice j pour les variables.

Sommaire

Fonction de perte

Plus tôt, nous avions souhaité vouloir construire $\hat{f}$ qui approxime les réponses y. En particulier, nous devons définir rigoureusement $\hat{f}(\mathbf{x}) \approx y$.

Fonction de perte

Une fonction de perte $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ est une métrique vérifiant les propriétés suivantes

- Pour tout $y, y' \in \mathcal{Y}$ tel que $y = y'$ alors $l(y, y') = 0$.
- Pour tout $y \neq y'$, $l(y, y') > 0$.

Par exemple, pour un problème de régression, la métrique couramment utilisée est la **perte quadratique** $l(y, \hat{y}) = \|y - \hat{y}\|_2^2$. Un « bon » modèle est un modèle qui minimise la perte entre les prédictions $\hat{y}$ et les réponses y.

Fonction objectif

Un modèle statistique **doit** avoir une **fonction objectif** à minimiser. C'est elle qui permet de « guider » le modèle vers une bonne solution.

En reprenant la perte quadratique $l(y, \hat{y}) = \|y - \hat{y}\|_2^2$, le modèle cherche à approximer toutes les observations de sorte que $\hat{f}(\mathbf{x}) = \hat{y} \approx y$ soit vérifiée partout. Une façon de lui indiquer un objectif **clair** et de lui dire de minimiser la somme de toutes les erreurs commises :

$$\mathcal{L}(y, \hat{y}) = \sum_{i=1}^{n} \|y - \hat{y}\|_2^2$$

Ainsi, plus les approximations sont correctes, plus $\mathcal{L}(y, \hat{y})$ tend vers 0.

Différentiabilité de la fonction de perte

En théorie, on pourrait se dire que pour contraindre fortement le modèle, nous pourrions employer une fonction objectif plus restrictive, notamment en pénalisant le modèle sur l'observation où ce dernier est le moins efficace :

$$\mathcal{L}(y, \hat{y}) = \max_{1 \leq i \leq n} \|y_i - \hat{y}_i\|_2^2$$

Malheureusement, cette fonction est **non différentiable**, et beaucoup d'algorithmes ne peuvent fournir une solution optimale puisque le problème devient NP-complexe ou alors non utilisable par les algorithmes types descente de gradient.

Notons tout de même que certains algorithmes (dont ceux utilisant la descente de gradient) peuvent tout de même fonctionner sur des fonctions de pertes **sous différentiables** avec des opérateurs proximal.

Exemples de fonctions objectifs classiques

Mean Squared Error (MSE) :

$$\mathcal{L}(y, \hat{y}) = \sum_{i=1}^{n} \|y_i - \hat{y}_i\|_2^2$$

Entropie croisée binaire :

$$\mathcal{L}(y, \hat{y}) = -\sum_{i=1}^{n} \{y_i \log p + (1 - y_i) \log(1 - p)\}$$

Le problème du supervisé

En apprentissage supervisé, nous cherchons à minimiser **l'espérance de la fonction de perte**.

$$\min_f \mathbb{E}[l(y, f(x))]$$

Ce problème de minimisation est difficile, puisqu'il nécessite de déterminer, parmi toutes les fonctions, celle qui minimise la quantité étudiée. Afin d'obtenir un problème dont une approximation de la solution peut être obtenue, il convient de restreindre le problème à une classe de fonctions « candidates » $\mathcal{F}$ avec les hyper-paramètres que l'on choisi en fonction du problème étudié.

$$\min_{f \in \mathcal{F}} \mathbb{E}[l(y, f(x))]$$

Problème : en fonction du choix de $\mathcal{F}$, il est possible de ne plus pouvoir déterminer f^*, car potentiellement $f^* \notin \mathcal{F}$.

Le problème du supervisé

En pratique, c'est la fonction objectif que l'on a choisi qui va nous permettre de construire un modèle qui se rapproche le plus du modèle théorique.

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \mathcal{L}(y, f(\mathbf{x}))$$

Trois facteurs rentrent en jeu pour déterminer une solution optimale au problème précédent :

- La fonction objectif $\mathcal{L}$ à utiliser.
- La classe de fonction $\mathcal{F}$ candidates pour le problème étudié.
- Un algorithme d'optimisation numérique.

Sommaire

Métriques

Comme nous l'avons vu, la fonction objectif est un moyen de fournir au modèle un mode d'emploi pour trouver une solution satisfaisante. Néanmoins, ces fonctions objectifs peuvent être plus difficiles à interpréter en terme de **mesure de qualité** des modèles.

Au cours des recherches, **plusieurs métriques** ont été développés afin de fournir une indication sur la qualité de prédiction du modèle. La plupart de ces métriques fournissent un score facilement interprétable (entre 0 et 100%) permettant ainsi de juger de la performance d'un ou plusieurs modèles.

Métriques en régression

Score R^2

Dans un problème de régression, il est très courant de choisir une fonction de perte quadratique. La métrique adaptée est le **coefficient de détermination** R^2.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} w_i \|y_i - \hat{y}_i\|_2^2}{\sum_{i=1}^{n} w_i \|y_i - \bar{y}\|_2^2}$$

avec w_i un poids associé à la i-ème observation. Si l'on ne privilégie pas d'observations, on utilise $w_i = 1, 1 \leq i \leq n$.

Métriques en classification binaire

Mesurer la performance d'un modèle de classification binaire est plus complexe : le choix de la métrique va également dépendre des données.

La métrique la plus simple utilisée en classification binaire est l'**accuracy** (ou métrique 0/1).

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}_{\{\hat{y}_i=y_i\}}$$

Pour chaque observation $\mathbf{x}_i$, on compare la prédiction $\hat{y}_i$ avec la réponse y_i. S'il y a égalité, on rajoute $\frac{1}{n}$ au score, de sorte à ce que plus il y a d'observations correctement prédites, plus le score se rapproche de 1.

Métriques en classification binaire

Un des principaux défauts de la métrique $0/1$ est qu'elle ne prends pas en compte les **faux positifs** et les **faux négatifs**. En effet, dans certaines situations, prédire la classe 1 alors que la vraie classe est 0 peut ne pas avoir le même impact que dans le sens inverse.

Exercice

Imaginons que l'on ait 1000 individus dont 900 appartiennent à la classe 0 et les 100 autres à la classe 1. Considérons l'estimateur naïf suivant :

$$f(\mathbf{x}) = 0$$

Autrement dit, ce modèle prédit tout le temps la classe 0, quelles que soient les valeurs des variables. Quelle est la valeur de la métrique $0/1$ pour ce modèle ? Ce résultat vous paraît-il réaliste ?

Le $F1$ score

Pour résoudre ce défaut, une des possibilités est de prendre justement en compte ces erreurs de premier et de second ordre. Il existe deux quantités qui permettent d'interpréter ces différentes erreurs :

- La **précision**, qui est la proportion d'observations correctement prédites positivement parmi toutes celles qui sont prédites positivement.

$$\text{Precision} = \frac{TP}{TP + FP}$$

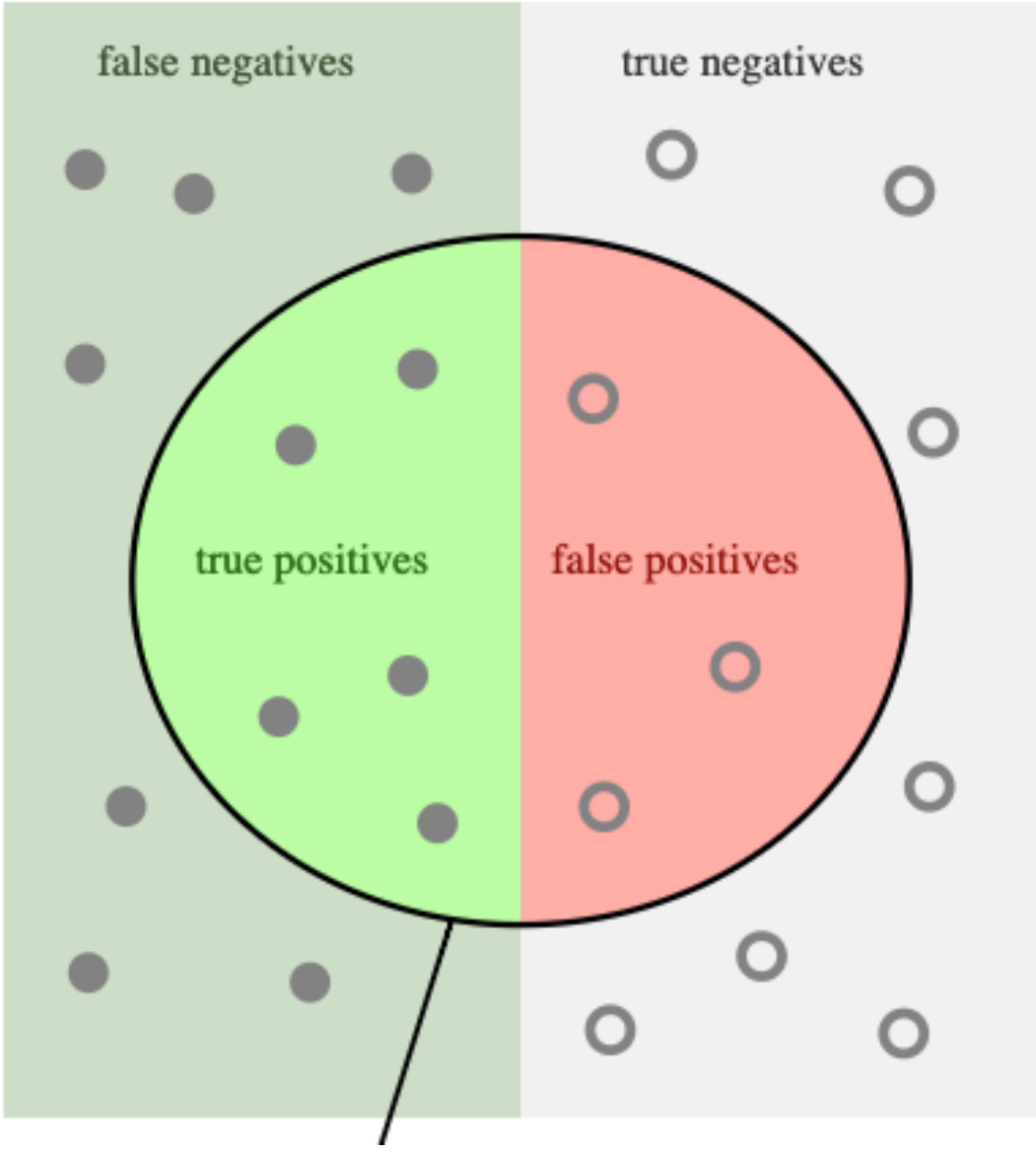
- Le **rappel**, qui est la proportion d'observations correctement prédites positivement parmi toutes celles qui devraient être prédites positivement.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Cela aboutit donc à la création du $F1$ score :

$$\text{Score}_{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Précision et rappel



Précision et rappel

Exercice

Montrer que Score_{F1} est toujours compris entre 0 et 1. Dans quels cas peut-on avoir un score nul ? Et un score à 100% ?

Métrique en classification multiple

En classification multiple, il est toujours possible d'utiliser l'**accuracy**, mais cela transpose les mêmes faiblesses au cas multiple.

Une bonne pratique consiste à calculer un $F1$ score sur chacune des classes, et ensuite de les agréger (soit en moyennant les scores, soit en prenant le score le plus faible, etc). L'avantage de cette méthode est qu'elle permet toujours de montrer si certaines classes manquent de précision ou de rappel.

Divergence de Kullback-Leibler

Tout de même, nous pouvons utiliser des mesures de dissimilarité pour évaluer les performances.

Divergence de Kullback-Leibler

Soient $\mathbb{P}$ et $\mathbb{Q}$ deux mesures avec $\mathbb{P}$ absolument continue par rapport à $\mathbb{Q}$ sur un même espace mesurable (E, μ). La divergence de Kullback-Leibler de $\mathbb{P}$ par rapport à $\mathbb{Q}$ s'exprime à l'aide de la dérivée de Radon-Nikodym entre les deux mesures :

$$D_{KL}(\mathbb{P}\|\mathbb{Q}) = \int_E p \log \frac{p}{q} d\mu$$

où $\mathbb{P} = p d\mu$ et $\mathbb{Q} = q d\mu$.

Divergence de Kullback-Leibler

Considérons maintenant $E = \{e_1, \ldots, e_K\}$ un ensemble discret muni de la mesure de Dirac δ, et soient les mesures

$$\mathbb{P} = \sum_{k=1}^{K} p_k \delta_{e_i} \qquad \mathbb{Q} = \sum_{k=1}^{K} q_k \delta_{e_i}$$

Dans un telle situation, la divergence de Kullback-Leibler s'exprime par

$$D_{KL}(\mathbb{P} \| \mathbb{Q}) = \sum_{k=1}^{K} p_k \log \frac{p_k}{q_k}$$

Cette mesure permet alors de comparer la distribution prédite avec la distribution théorique : plus les deux distributions sont proches, plus la divergence KL se rapproche de 0. À l'inverse, plus la divergence KL est forte, moins la capacité du modèle à modéliser la distribution théorique est élevée.

Métriques

Notons que la divergence KL est également utilisée dans certaines fonctions objectifs.

Attention

Bien qu'utiles, les métriques ne permettent pas d'expliquer le modèle, elle fournissent uniquement une évaluation quantitative de la performance d'un modèle. De plus, cela n'a pas de sens de comparer les métriques entre-elles, car elles ne cherchent pas à évaluer les mêmes choses.

Sommaire

Sur-apprentissage

La **sur-apprentissage** (ou *over-fitting*) est un phénomène qui peut apparaître lors de la phase d'entraînement des modèles. Lorsqu'un modèle est en sur-apprentissage, il a "trop" appris sur les données dans le sens où, en essayant de minimiser le plus possible l'erreur sur les prédictions, il s'est également calibré sur le bruit qui est naturellement présent dans le jeu de données.

Ce bruit est une **variation aléatoire** que l'on ne peut corriger : il n'est pas possible d'expliciter analytiquement, pour une observation, le bruit associé par rapport à une valeur moyenne (au sens de la distribution).

$\Rightarrow$ Problème pour généraliser la prédiction à de nouvelles observations.

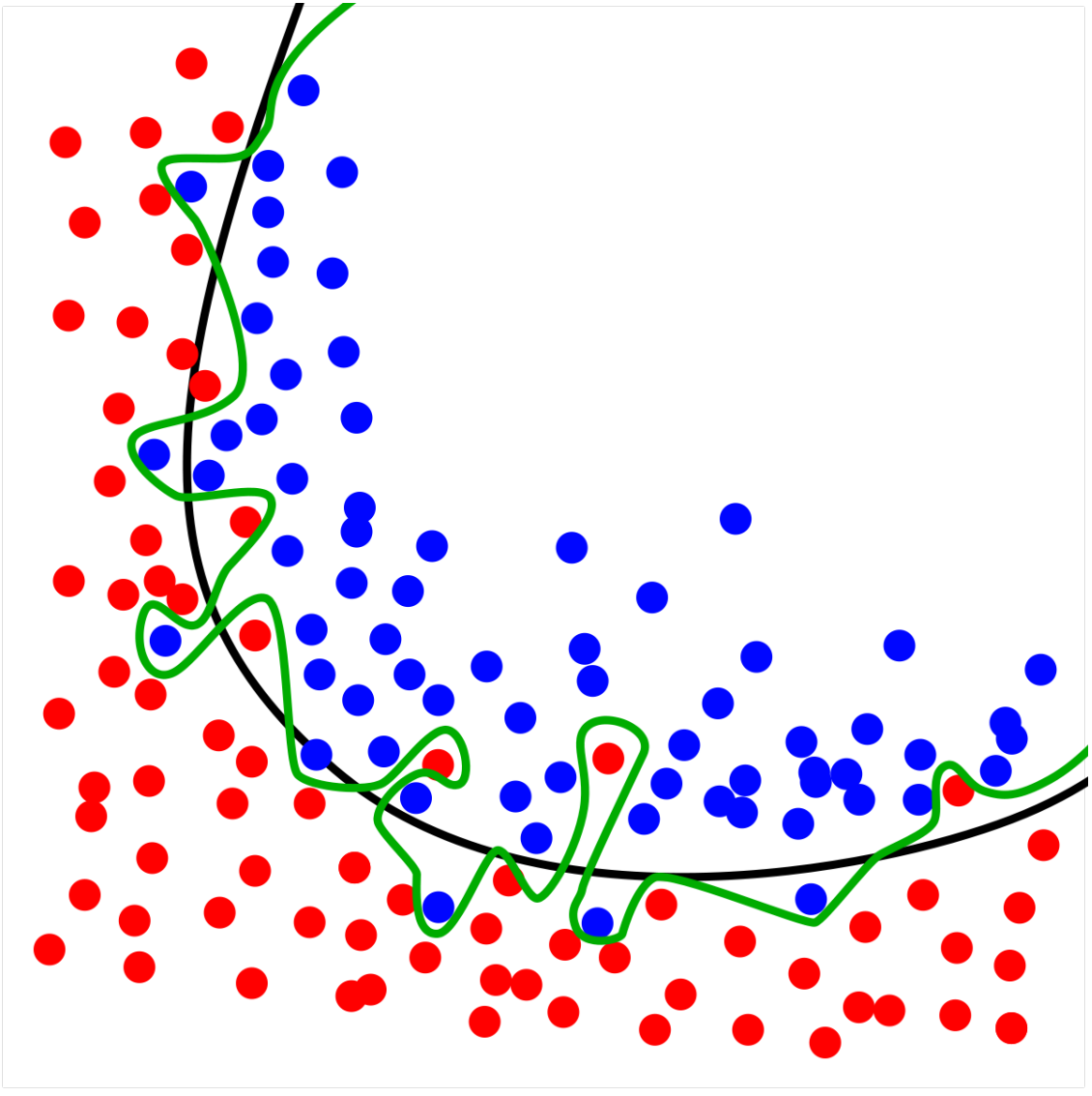
Sur-apprentissage

Si l'on suppose que toutes les observations $\mathbf{x}_i$ sont issues d'un même phénomène modélisé par une variable aléatoire X, alors chaque observation est une réalisation de la variable aléatoire

$$X + \varepsilon$$

où $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ est une loi normale centrée de variance σ^2. Un modèle est sensé converger vers la distribution de X : un sur-apprentissage mènera donc à converger vers la distribution de $X + \varepsilon$, qui n'est pas ce que l'on cherche à faire.

Sur-apprentissage



Sur-apprentissage

Plusieurs facteurs peuvent expliquer un sur-apprentissage :

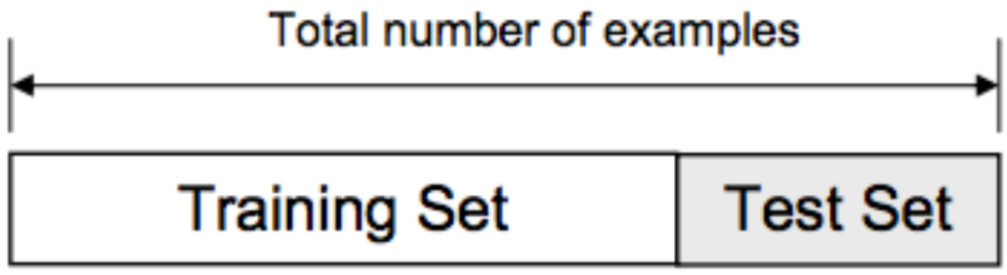
- Un modèle non-paramétrique ou sur-paramétrée.
- Une faible hétérogénéité dans les variables (peu de modalités différentes, effets de lignes dupliquées).
- Petite taille du jeu de données.

Sur-apprentissage

Comme évoqué auparavant, le problème du sur-apprentissage est que le modèle est très performant **pour les observations sur lequel il s'est entraîné**, mais ne l'est pas lorsqu'il s'agit de prédire de nouvelles observations.

En Machine Learning, une pratique très courante consiste à **séparer le jeu de données** en deux échantillons :

- Une base d'entraînement (**train set**) qui permettra d'entraîner le modèle.
- Une base de test (**test set**) qui permettra d'évaluer le modèle sur sa capacité à généraliser pour de nouvelles observations qu'il n'a jamais vu.



Sur-apprentissage