Chapitre 3
Machine Learning

1. Petit bestiaire de la Data Science

Les descriptions suivantes ont pour seul objet de se familiariser avec le vocabulaire et les représentations souvent utilisées par les data scientists. Ne cherchez pas à construire un bel échafaudage logique à partir de ces informations ; contentez-vous d'enrichir le nombre d'associations d'idées entre vos découvertes et vos connaissances antérieures.

1.1 Les fondamentaux

1.1.1 Apprentissage et classification

Une machine "apprend" quand son paramétrage évolue en fonction des circonstances.

L'**apprentissage supervisé** est l'application la plus directe de ce constat. Après avoir collecté des informations, on scinde chaque observation en deux parties ; l'une dite explicative et l'autre expliquée. On choisit ensuite une mécanique calculatoire qui semble permettre de déduire les variables expliquées à partir des variables explicatives. La mise au point de ce modèle consiste alors à trouver le paramétrage de celui-ci qui semble être le plus efficace. Ce processus d'apprentissage est dit "supervisé" car il est sous le contrôle des variables expliquées de l'ensemble des données d'entraînement.

Quand on collecte de nouvelles observations ne possédant des valeurs connues que pour les variables explicatives, il suffit d'appliquer le modèle avec ses paramètres pour obtenir ce que l'on appelle élégamment une prédiction ou plus simplement une estimation.

Les autres formes d'apprentissages n'ont pas pour objet de prédire quelque chose, mais elles fournissent des estimations de *patterns* (c'est-à-dire des schémas identifiables et répétitifs) qui n'apparaissaient pas de prime abord.

En anglais on parle de ***Machine Learning*** (**ML**). Les variables explicatives sont nommées au choix : *features*, *attributes* ou *covariates*. Les variables expliquées sont, elles, nommées *response variables*.

Quand on intervient sur un processus d'apprentissage en introduisant une information qui n'était pas disponible au début de celui-ci, cela s'appelle du renforcement de l'apprentissage.

Quand la variable expliquée, et donc à prédire, est une classe (0/1, couleurs, genre...), on parle tout simplement de **classification**.

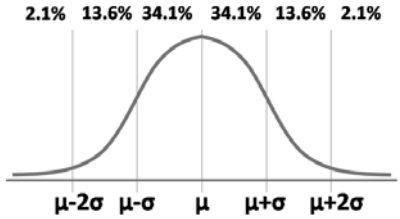
Il ne faut pas confondre la technique d'apprentissage nommée "classification", qui relève de l'apprentissage supervisé, avec la découverte de classes auparavant inconnues (*clusterisation* en anglais) qui, elle, n'est pas supervisée par une définition de classes a priori et qui constitue une des techniques de référence parmi les méthodes d'**apprentissage non supervisé**.

1.1.2 Petit vocabulaire graphique du Machine Learning

Les data scientists partagent certaines représentations mentales. Certains schémas visuels sont communs à une grande majorité de praticiens car ce sont les représentations qui les ont accompagnés pendant leur formation.

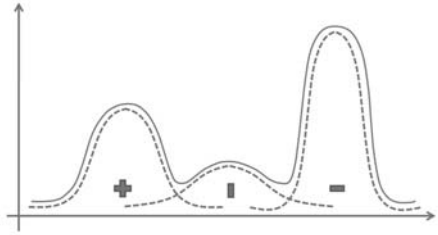
Ces représentations visuelles et le vocabulaire associé représentent parfois tout ce qui reste à une personne qui a étudié la Data Science sans la pratiquer, ou à un élève étourdi qui n'a pas peaufiné ses cours. Comme pour l'apprentissage des basiques d'une langue, par une méthode globale, avant même de comprendre, de questionner ou de chercher à retenir quoi que ce soit, il faut se familiariser graduellement avec les concepts à assimiler. Nous vous proposons l'expérience suivante : regardez attentivement et imprégnez-vous des 25 visuels du chapitre, quitte à commettre des interprétations erronées de leur signification. Si quelques formulations mathématiques vous échappent, peu importe, elles seront abordées plus loin dans l'ouvrage. Elles vous soutiendront pendant la lecture de cet ouvrage, mais aussi tout au long de vos autres lectures.

Johann Carl Friedrich Gauss (1777-1855) a laissé son nom à ce que nous appelons la loi normale ou courbe de Gauss. Quand un phénomène est réputé gaussien, que l'on en connaît sa moyenne et son écart-type, on peut facilement calculer la probabilité qu'un individu de la population ait une valeur donnée.



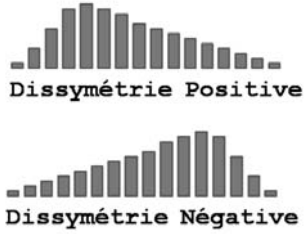
Loi normale, moyenne et écart-type

Les populations comprennent parfois des mélanges issus de diverses classes qui suivent individuellement des lois normales. Il existe des algorithmes, comme l'algorithme EM (*Expectation-Maximization*), qui arrivent à extraire les classes d'origine. Cet algorithme est exprimé dans sa forme actuelle depuis 1977 par messieurs Arthur Dempster, Nan Laird et Donald Rubin.



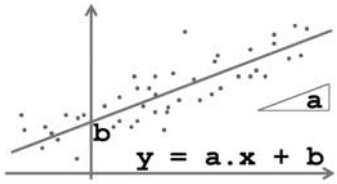
Mixture gaussiène séparable via l'algorithme EM

Toutes les lois ne sont pas symétriques, y compris quand ce ne sont pas des mixtures gaussiennes. Il existe une façon de caractériser la dissymétrie au travers d'un indicateur nommé *skewness* par les Anglais.



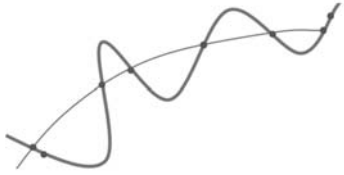
Skewness

La détermination de l'appartenance à une classe se concrétise par le fait de labelliser son échantillon avec le nom de la classe : classe_1, classe_2... Ce qui donne une vue pointilliste du monde (dite discrète). Pour prédire non plus une classe, mais une valeur continue, il faut utiliser une mécanique dite de régression. L'ancêtre de celle-ci se nomme la régression linéaire (de Ruder Josip Boškovic, milieu du XVIII^{e} siècle), à laquelle on associe souvent une notion de coefficient de corrélation, ou son carré, le coefficient de détermination, afin de concrétiser le niveau de qualité de l'approximation effectuée.



Régression linéaire

Un nuage de points qui n'est pas aligné sur une droite se laisse mal approximer par une régression linéaire ! La panoplie du data scientist contient de nombreux outils pour résoudre ce problème. La régression polynomiale est de ceux-ci, car elle permet d'approximer des nuages de point en courbes de natures variées. Malheureusement, c'est aussi la technique qui illustre le mieux la grande peur de la profession, le "sur-apprentissage", ou *overfitting* en anglais. Son effet est désastreux car, alors, au lieu de trouver une bonne approximation, le fait de vouloir approcher au plus près tous les points d'un nuage rend le modèle impropre et non généralisable à la prédiction d'un nouveau point. Le danger d'*overfitting* n'est pas propre à la régression polynomiale, mais saute aux yeux quand on utilise mal cet outil.

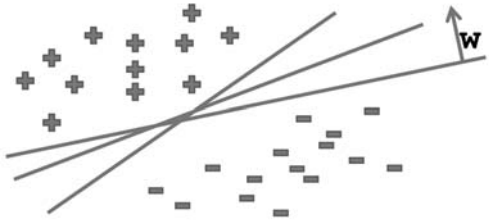


Régression polynomiale et danger d'overfitting

Remarque

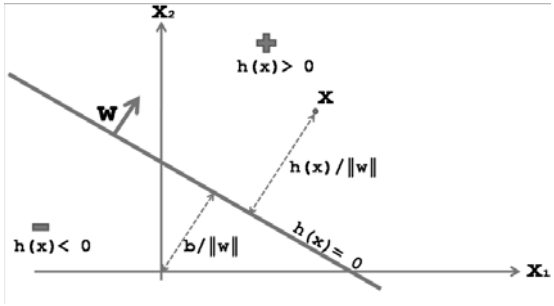
Ici, le fait d'avoir voulu faire passer la courbe par les 2 x 2 points extrêmes du nuage de points a complètement déformé l'approximation qui est devenue accidentellement oscillante.

Le **problème de régression** est moins facile à appréhender que le **problème de classification**. L'un et l'autre sont originellement liés par la notion de droite, ou sa généralisation dans des espaces plus grands en ce que l'on nomme des hyperplans. En effet, le problème de classification le plus simple à représenter prend la forme de deux classes séparées par une droite ou un hyperplan. L'objet de nos algorithmes est alors de trouver l'hyperplan qui sépare le plus intelligemment les deux classes (plus bas représentées par les symboles **+** et **-**). Il est commode de caractériser un hyperplan par un vecteur **w** perpendiculaire à celui-ci, tel que représenté dans le schéma suivant.



Hyperplans séparateurs, permettant une classification binaire (deux classes)

Pour les matheux, le schéma qui suit ne posera pas trop de problème. Pour les autres, retenez qu'il existe une fonction, ici nommée h(x), telle que les points x où h(x) est nul sont situés sur l'hyperplan séparateur.

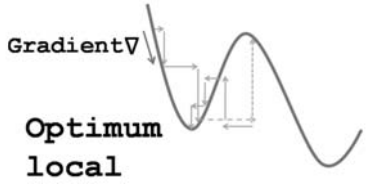


Formulation mathématique d'un hyperplan caractérisé par un vecteur w

Globalement, ce schéma représente un hyperplan (en dimension 2, un tel hyperplan est une droite). L'équation générale des hyperplans est donnée (**$h(x) = w^T .x + b = 0$**), ainsi que la façon d'utiliser celle-ci pour séparer la classe positive de la classe négative. Cette formulation est très importante, on la retrouve d'ailleurs peu ou prou dans la formulation des modèles modernes utilisant les réseaux neuronaux.

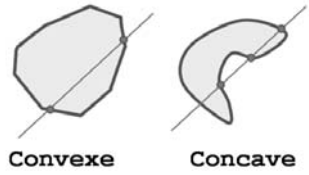
Pour des raisons de commodité, et en cas de doute, la distance de l'hyperplan à l'origine est mise en valeur, ainsi que la distance d'un point quelconque avec l'hyperplan. Ces deux distances étant fonction du module (la norme) du vecteur perpendiculaire à l'hyperplan qui contient les paramètres de celui-ci.

Pour déterminer ces hyperplans, il faut utiliser divers algorithmes. Parmi eux, de nombreux ont un point commun : à un moment ou un autre il faut trouver le minimum (ou le maximum) d'une fonction qui sert à contrôler l'algorithme, par exemple une fonction qui mesure le taux d'erreur (*loss*). Une des méthodes les plus communément employées se nomme la méthode de la "**descente du gradient**". Elle possède de nombreuses versions et de nombreuses adaptations. Parfois l'algorithme utilisé pour trouver l'optimum est mis en défaut, par exemple celui-ci reste bloqué sur un optimum local alors qu'il existe un meilleur optimum, ou alors l'algorithme ne converge pas assez vite (ou pas du tout !). La recherche d'optimum est en fait une discipline en soi qui a fait l'objet de nombreuses recherches.



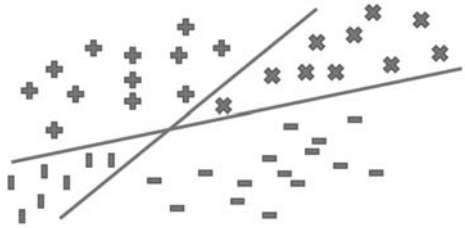
Recherche infructueuse d'un optimum par une descente de gradient

Ces méthodes sont sensibles aux conditions initiales, c'est-à-dire à la façon dont on les initialise. Elles sont également sensibles à la topologie du problème posé, c'est-à-dire aux caractéristiques des fonctions que l'on veut optimiser. La notion la plus couramment utilisée pour déterminer si une fonction va être simple ou pas à optimiser sur son domaine de définition, est la notion de convexité. Pour déterminer si une forme fermée est convexe, il faut imaginer des hyperplans tangeants en chaque point de sa frontière où il est possible de définir un hyperplan tangeant. Si aucun de ces différents hyperplans n'intersecte l'intérieur de la forme, alors elle est convexe. En deux dimensions, l' identification de la convexité est très intuitive.



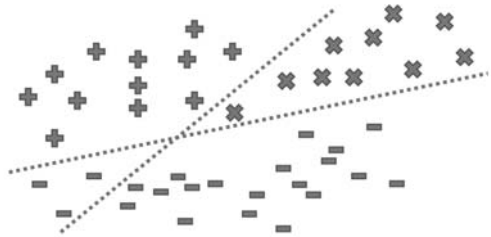
Identification de la convexité d'une forme en deux dimensions

Les modèles à partir desquels on veut séparer plusieurs classes peuvent être abordés en déterminant plusieurs hyperplans. Suivant les topologies, il sera alors possible ou pas de séparer les classes de façon linéaire. Ici, deux hyperplans séparent linéairement quatre classes :



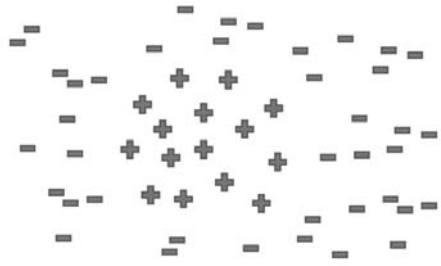
Linéairement séparable, quatre classes

Dans certains cas, pourtant apparemment simples, il n'existe pas de manière triviale de séparer l'espace au travers d'hyperplans complets.



Non linéairement séparable, trois classes

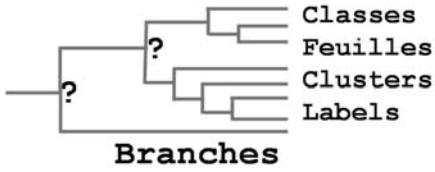
Les classes, même quand elles sont bien constituées, ne sont pas toujours séparables par des structures linéaires ; pour s'en convaincre, il suffit de visualiser ce schéma :



Non linéairement séparable, deux classes

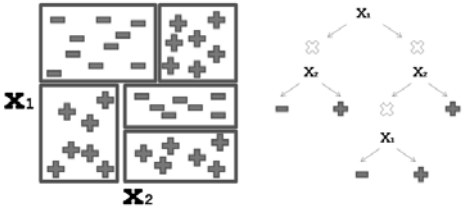
Dans ces cas, mais aussi dans les cas linéairement séparables difficiles, la classification reste possible en introduisant d'autres algorithmes. Une première façon de procéder est de s'appuyer sur des algorithmes générant des arbres de décision utilisant des séparations partielles de l'espace.

La représentation via des dendrogrammes, comme ci-après, est souvent utilisée pour décrire les classes "devinées par la machine et inconnues auparavant" que l'on nomme alors *clusters*. On peut aussi l'utiliser pour décrire l'arborescence qui mène à la séparation des observations sur des classes (labels) dont l'existence était connue a priori (ce qui est le cas pour ce que nous venons d'aborder dans les schémas précédents).



Vocabulaire des arbres de décision et des dendogrammes

Le schéma ci-dessous a pour objet de mettre en évidence comment un ensemble non séparable de deux classes peut être facilement représenté par un arbre de décision.

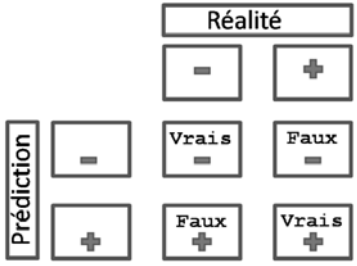


Modélisation d'un ensemble de classes au travers d'un arbre de décision

Remarque

Ici, on a tout d'abord une règle de décision sur la séparation de deux blocs via une valeur de x1, puis, dans chacun de ces deux blocs, on sépare en fonction d'une valeur de x2. Enfin, dans le bloc de droite, on sépare à nouveau en fonction d'une valeur de x1.

Un algorithme de classification peut être confronté à des cas où certains points sont noyés dans d'autres classes, ou à des cas où les frontières de classification sont un peu floues. De plus, les algorithmes et leurs paramétrages ne sont pas toujours optimisés pour les données que l'on veut traiter. En fait, une grande part du travail d'un data scientist consiste à améliorer ses choix d'algorithmes et de paramètres pour s'adapter à ces cas de figure. La notion d'*overfitting* a été introduite plus haut : en voulant trop bien faire, il est possible de s'égarer et de surdéterminer le modèle. Le concept de matrice de confusion est un des plus simples à mettre en œuvre pour percevoir le taux d'erreur de classification. Il suffit de **compter comment se répartissent les classes trouvées versus les classes réelles**.



Matrice de confusion, tri des types d'erreur de classification

L'observation de cette matrice fait apparaître des notions simples et intéressantes, comme le nombre de "vrais négatifs", de "faux négatifs", de "faux positifs" et de "vrais positifs". Le ratio de tous les "vrais" divisé par le nombre total d'observations reflète le taux d'erreur total, il se dénomme *accuracy* en anglais.

À partir de ces informations, il existe une technique de calcul nommée ROC (*Receiver Operating Characteristic*), qui a été développée dans le cadre de l'interprétation des signaux radars pendant la Seconde Guerre mondiale, pour éviter un deuxième "Pearl Harbor". Schématiquement, la courbe ROC est construite en faisant varier une quantité et en notant sur un graphique les couples de valeurs correspondants. Les deux membres du couple sont le taux de faux positifs (qui correspond à la soustraction à 1 de ce que l'on nomme spécificité, reporté sur l'axe des x), et le taux de vrais positifs (nommé sensibilité et reporté sur l'axe des y). Plus la valeur située sous cette courbe est importante, meilleure est la qualité de la prédiction. C'est très pratique pour comparer les performances de deux classificateurs binaires (c'est-à-dire deux algorithmes associés avec leurs paramétrages respectifs). La surface sous la courbe se nomme AUC (*Area Under the Curve*).