Machines à noyau : une très courte introduction intuitive

ou les SVM décryptées
ou les SVMs pour les nuls ...

Philippe Preux
INRIA & CRIStAL (UMR CNRS 9189)
Université de Lille
philippe.preux@inria.fr

Version du 18 février 2019 *

Résumé

Ce court texte a pour espoir d'expliquer ce que sont les machines à noyau aux gens qui ne connaissent à peu près rien à l'apprentissage automatique. Pour ceux qui ont une bonne connaissance de l'apprentissage automatique, ce texte est totalement inutile. Pour les autres, il ne se veut qu'un tout premier pas, pour aider, si nécessaire, à rentrer dans les mathématiques sous-jacentes.

Il est écrit par un informaticien dont la formation ne l'a absolument pas préparé aux mathématiques qui sont nécessaires ici. Il les présente comme il peut, comme il les a comprises et comme il croît qu'il peut les présenter à un public d'informaticien. Il y a très peu de formalisation, beaucoup d'intuition. Pour le lecteur intéressé, tout se formalise et est décrit dans d'autres textes.

Machines à noyau, machines à vecteurs supports, ou autres séparateurs à vaste marge (SVM) ont fait couler beaucoup d'encre et de toner depuis 10 ans. Ces machines sont entourées d'un certain mystère, de performances expérimentales incroyables et magiques, le tout étant censé être expliqué, et une conséquence qui doit sembler évidente, de quelques pages de mathématiques où l'on croise des notions d'analyse fonctionnelle, ces fameux « espaces de Hilbert à noyau reproduisant », de Lagrangiens et autres créatures fabuleuses.

D'un naturel curieux, et méfiant, j'ai voulu comprendre tout ça, vérifier les performances étonnantes de ces « machines » ; informaticien de formation, je me suis retrouvé au beau milieu d'un bain mathématique qui dépassait de beaucoup ma modeste formation acquise durant mes premières années d'université, où ces jolies notions et bien d'autres se mélangeaient allégrement

*La première version de ce texte date d'environ 2010. Depuis, je fais uniquement des corrections superficielles. Par rapport à la version précédente datant du 29 mars 2017, quelques typos corrigées, la note de bas de page 14 qui n'apparaissait pas apparaît maintenant, et j'explique la raison pour laquelle on résout le problème dual de la SVM.

en une espèce de bouillon indigeste, formant un véritable mur de connaissances à gravir. Cet effort m'a notamment permis de découvrir des territoires insoupçonnés des mathématiques que je trouve aujourd'hui passionnants, et indispensables à connaître, tellement ils sont magnifiques... et utiles.

Constatant toujours une assez grande curiosité autour de moi quant à ces machines à noyau, j'ai résolu de mettre sur le papier ce que j'avais compris, de la manière la plus intuitive qui soit. La lecture de la suite va ainsi montrer que l'intuition qui permet d'appréhender ces machines est d'une simplicité absolument redoutable. J'essaie ensuite de montrer comment les outils mathématiques plus sophistiqués, dont ceux mentionnés plus haut, s'organisent pour opérationnaliser cette intuition sous la forme d'algorithmes pratiques.

Pré-requis

Quelques notions d'analyse sont supposées connues : notions de dérivée, minimum/maximum d'une fonction, le fait qu'en un minimum/maximum, la dérivée d'une fonction s'annule... Soit, 1re année de licence de mathématiques.

1 Intuition initiale

- on considère les points du plan ; chaque point a une couleur, rouge ou bleu
- on dispose d'un ensemble d'exemples, c'est-à-dire des points dont on connaît la couleur
- on veut prédire la couleur (≪ la plus probable ≫ [1]) de n'importe quel point du plan
- il existe des tas de méthodes pour faire cela [2] ; l'une d'elles est l'utilisation de machine à noyaux : c'est quoi ?
- l'intuition est la suivante : je veux connaître la couleur d'un nouveau point : je regarde, parmi les exemples, lesquels en sont proches : s'ils sont tous bleus, je prédis la couleur bleu ; s'ils sont tous rouges, je prédis la couleur rouge ; s'il y a un mélange de couleur, c'est un peu plus compliqué mais voilà le principe
- on vient d'utiliser la notion de proximité, donc de distance ou dissimilarité entre des points. Cette notion entend formaliser la ressemblance entre des points. Pour quantifier cela, on utilise une fonction qui associe une similarité à toute paire de points.
- on appelle cette fonction un ≪ noyau ≫ [3]

1. expression à prendre dans un sens intuitif.

2. arbres de décision, classification bayesienne, réseau de neurones, plus proche voisin, séparateur à vaste marge, ... on pourra consulter mon poly de fouille de données [Preux, 2006] ou [Hastie et al., 2001].

3. Soyons clair : le mot ≪ noyau ≫ est utilisé, selon les auteurs, avec des sens très différents en mathématiques (noyau d'une application, noyau d'une intégrale, ...). Si l'on s'en tient aux sens du mot très proches de celui qui nous intéresse ici, la définition que j'utilise ici est très large : c'est simplement une fonction qui associe un réel à toute paire de données (je suis là notamment la définition donnée par Tom Mitchell dans son *Machine Learning*). Certains auteurs, notamment dans la littérature SVM, demandent que la fonction soit symétrique et définie positive pour l'appeler un noyau ; cela rejoint alors la notion de ≪ noyau de Hilbert ≫ en mathématiques. Dans la suite, nous gardons donc un sens très général au mot noyau, et nous préciserons les propriétés supplémentaires attendues lorsque cela sera nécessaire.

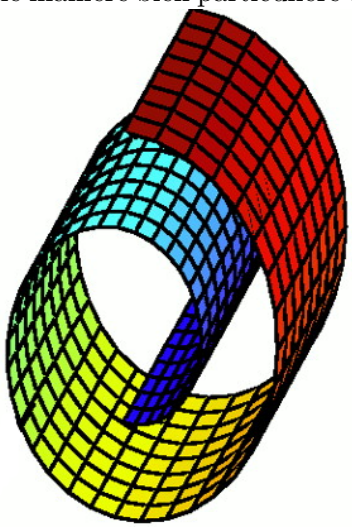
— formellement, ce noyau peut définir une distance [4] au sens mathématique (notamment respectant l'inégalité triangulaire) ou pas ; dans ce second cas, on parle de (dis)similarité : c'est moins fort qu'une distance, mais c'est mieux que rien !

— si ce noyau possède certaines propriétés mathématiques, des résultats magnifiques d'analyse fonctionnelle permettent, en faisant certaines hypothèses simples et intuitivement assez évidentes, d'obtenir une solution globalement optimale [5] au problème de la prédiction de la couleur des points du plan

— en l'absence de ces propriétés, le résultat est moins fort et on obtient une solution localement optimale au mieux, qui est déjà tout à fait convenable en général

— l'un des problèmes de cette approche est que définir un noyau pour des données qui ne sont pas représentées sous forme d'un vecteur de réels qui a un sens pour la donnée (images, musique, séquences d'images, séquence d'ADN, arbre, graphe, ...) est un problème ouvert en ce moment, même si de nombreuses propositions ont déjà été publiées ; une exception est constituée par les textes que l'on peut transformer facilement en vecteur de réels, avec des résultats expérimentaux tout à fait remarquables

— un autre problème est que, même dans le cas simple où les données sont des vecteurs réels, le noyau doit effectivement représenter la (dis)similarité entre les données : la simple distance euclidienne peut ne pas convenir du tout ; tout dépend de la topologie de l'espace [6] où sont situées les données. Pour illustrer ce point délicat, la figure suivante montre des points répartis d'une manière bien particulière dans l'espace tri-dimensionnel.



Si l'on suppose que la ressemblance entre deux points s'entend comme la proximité sur cette structure en spirale, il est clair que les points en bleu sombre sont éloignés des points jaunes, de même pour les points bleu clair et rouge sombre ; on constate que la distance euclidienne reflète très mal cette ressemblance et n'est donc pas la « bonne » mesure de

4. rappelons qu'une distance d se définit mathématiquement par : c'est une application qui a tout couple de points x et x' associe un réel positif $d(x, x')$, qui respecte les trois propriétés suivantes : $d(x, x') = 0 \iff x = x'$ (la distance entre un point et lui-même est nulle et si une distance entre deux points est nulle, c'est que ces deux points sont le même point), $d(x, x') = d(x', x)$ (symétrie) et $d(x, x') \leq d(x, x") + d(x", x')$ (inégalité triangulaire).

5. optimal par rapport à un critère fixé, que nous préciserons.

6. La « variété » ou vivent les données pour être plus précis.

dissimilarité, donc le bon noyau. Le ≪ bon ≫ noyau doit mesurer la proximité des points en suivant la spirale.

2 Allons un peu plus loin : un peu de vocabulaire

— le problème de prédiction de couleur ainsi qu'il a été présenté est un exemple du ≪ problème de classification supervisée ≫ : c'est un problème extrêmement classique

— ce problème peut se généraliser au cas où il y a plus de deux couleurs possibles : problème de classification supervisée alors dit ≪ multi-classes ≫ : c'est aussi un problème extrêmement classique

— il peut encore être généralisé au cas où on associe un rang à chaque point, problème alors dit de ≪ classification ordinale ≫ ; par rapport au cas multi-classes précédents, on a ici une relation entre les classes : c'est encore un problème extrêmement classique, bien que moins étudié que les deux précédents

— il peut encore être généralisé au cas où on associe un nombre réel à chaque point : problème alors dit de ≪ régression ≫ : c'est encore un problème extrêmement classique [7]

— on peut encore le généraliser au cas où on ne prédit pas une seule valeur pour chacun des points, mais un ensemble de valeurs (un vecteur, ...) : problème alors dit de ≪ sortie structurée ≫ : problème assez récemment posé et sujet très chaud depuis quelques années

— d'une manière générale, l'entité (couleur, nombre, ...) ainsi associée à une donnée est dénommée une ≪ étiquette ≫

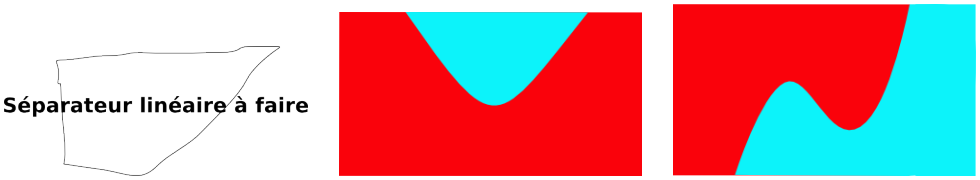
— naturellement, on peut considérer d'autres espaces de données que le plan, l'espace tridimensionnel ou tout espace $\subset \mathbb{R}^P$, borné ou non, compact (sans trou) ou non, convexe (en forme d'ellipse ou d'ellipsoïde plus généralement, pour faire simple [8]) ou non

3 Toujours plus loin

— reprenons l'exemple des points du plan et des deux couleurs. Si on représente graphiquement ce plan et la coloration des points, on peut avoir des tas de situations. On s'intéresse notamment à la nature de l'objet géométrique qui sépare les bleus des rouges, que l'on nomme un ≪ séparateur ≫. Par exemple :
  — les bleus et les rouges sont nettement séparés par une droite, une parabole, une cubique, ...

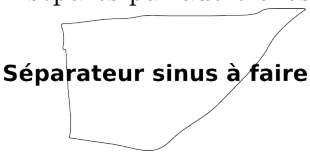
7. Il existe de nombreux synonymes tels qu'interpolation et approximation de fonction.

8. Plus précisèment, mais toujours intuitivement, un espace ext convexe si quelle que soit la paire de points considérée appartenant à cet espace, tous les points du segment de droite reliant ces deux points appartiennent à l'espace.



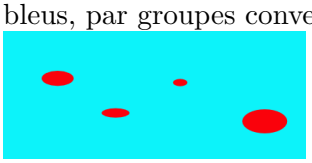
Dans le premier cas (bleus et rouges séparés par une droite), le séparateur est linéaire.
Dans les autres cas, il est non linéaire.
Quand les donnés sont situées dans le plan, un séparateur linéaire est une droite.
Quand les donnés sont situées dans le plan, un séparateur non linéaire est n'importe quoi, sauf une droite.

— les bleus et les rouges sont bien séparés par autre chose qu'une droite

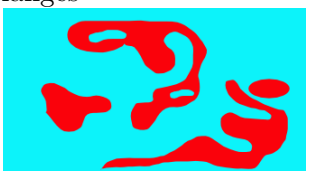


Dans ce cas, le séparateur est non linéaire.

— les rouges sont noyés dans les bleus, par groupes convexes



— les bleus et les rouges sont mélangés



Dans ces deux cas « bizarres », notamment celui où les rouges sont structurés en groupes convexes (c'est plus facile à comprendre dans ce cas, mais c'est pareil dans le suivant), on peut s'imaginer qu'en fait, ces ellipses rouges constituent la coupe selon un certain plan d'un objet géométrique de dimension P supérieure à 2 (3 ou plus). Cela peut s'interprêter comme le fait que les données sont en fait (on dit que les données « vivent ») dans un espace de cette dimension P et qu'on ne les représente que partiellement avec les deux dimensions du plan.
Pourquoi cela me direz-vous ?
Parce qu'il faut bien se rappeler d'où viennent les données que l'on manipule : ce sont des données que l'on a collectées : pour chaque donnée, on mesure un certain nombre de caractéristiques/attributs. Quand on fait cela, on espère bien sûr mesurer tous les attributs importants mais ce n'est qu'un espoir qui n'a aucune raison d'être accompli. Les attributs manquants, que l'on n'a pas mesurés, existent et sont importants ; comme on ne les a pas collectés, on ne les représente pas ce qui, si on a mesuré deux attributs pour chaque donnée, nous amène à une représentation des données dans le plan ; néanmoins, si un attribut a été omis, on comprend bien que l'on devrait représenter les données dans un espace à trois dimensions et que donc,

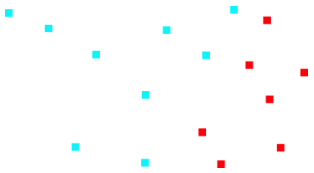
la représentation que l'on en fait avec les deux seuls attributs mesurés est une coupe dans cette représentation tridimensionnelle.

Là, on vient de toucher du doigt un point d'une importance capitale : les attributs mesurés et caractérisant les données que l'on traite peuvent être insuffisants pour décrire comme il le faut [9] les données, voire, ne pas être les bons attributs du tout et nous donner une image fausse des données que l'on tente d'analyser.

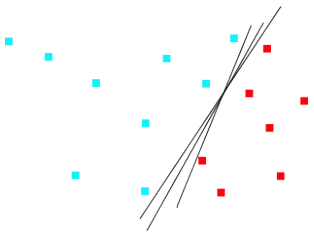
Ces remarques sont très profondes à mon avis : elles renvoient à notre perception des choses et à l'éternelle question de connaître et comprendre le lien entre ce que nous percevons et ce que les choses sont vraiment.

— donc, si on résume, apprendre à classer des données en rouge et bleu, cela revient à déterminer l'objet géométrique qui les sépare. L'approche la plus simple et la plus traditionnelle consiste à construire un objet géométrique dans l'espace dans lequel sont représentées les données (P dimensions donc) ; d'autres approches cherchent cet objet dans des espaces d'autres dimensions, cherchant donc en même temps la « bonne » dimension des objets. Restons-en pour l'instant au cas le plus simple pour nous forger et développer notre intuition et prenons $P = 2$ pour que l'on puisse faire des illustrations graphiques.

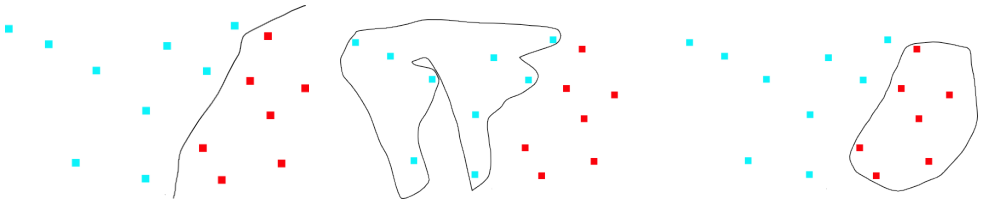
— reposons bien le problème. On a un ensemble de points colorés (des exemples) :



et on cherche un objet géométrique qui les séparent. Seule l'imagination nous limite ! une infinité de droites sont possibles :



sans parler du reste (séparateurs non linéaires) :



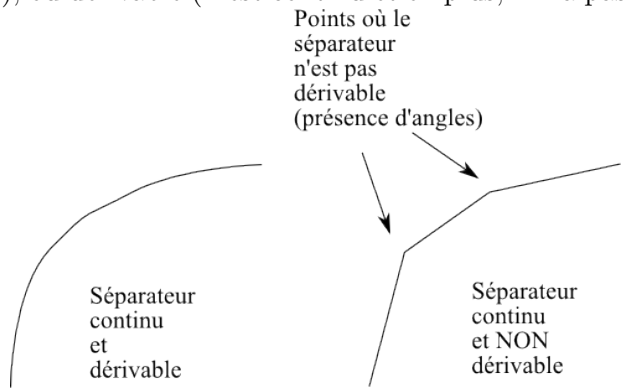
Que faire ? Quel type de séparateur choisir et comment le trouver ?

— un principe général en science et de toujours privilégier les hypothèses les plus simples ; parmi tous ces séparateurs, rien n'est plus simple qu'un droite ; donc, cherchons une

9. « comme il le faut » signifie ici, pour mener à bien la tâche que l'on tente de résoudre — ici, classer des données.

droite. Sur la figure ci-dessus, on voit déjà que ce n'est pas si simple : une infinité de droites séparent les exemples bleus des rouges : laquelle choisir et comment l'obtenir ?
Si on se dit que n'importe laquelle est bonne, alors l'algorithme du perceptron (règle Delta, connue aussi comme règle de Widrow-Hoff, règle Adaline, ...) nous la fournit. Ce n'est pas une machine à noyau, donc cela ne nous intéresse pas vraiment ici.
Si on veut que se soit celle qui passe juste au milieu, entre les rouges et les bleus, alors un SVM linéaire nous résout le problème. C'est une machine à noyau.
On peut aussi vouloir pratiquer comme on l'a expliqué au début de cette note : pour toute donnée, on regarde parmi les données les plus proches si elles sont plutôt bleu ou plutôt rouge. En général, cela ne va pas nous donner un séparateur linéaire et donc, on en parle juste en dessous.
Notons qu'un séparateur est un « modèle » : chercher le séparateur le plus simple, c'est donc la même chose que chercher le modèle le plus simple qui représente une situation, ici la tâche de classification que l'on essaie de résoudre [10].

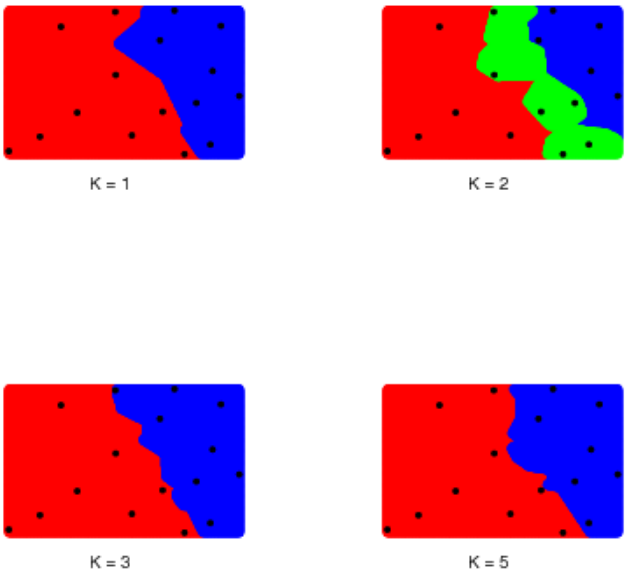
— si on cherche un séparateur non linéaire, il y en a tellement de différents qu'il faut se donner quelques principes. Un tel principe est par exemple de prendre un séparateur qui zigzague le moins possible, ou un séparateur continu (que l'on peut tracer sur le papier sans lever le crayon), ou dérivable (il est continu et en plus, il n'a pas de « pointes »), ...



Autrement dit, on se donne une famille de fonctions séparatrices parmi lesquelles on va essayer de trouver la plus adéquate.

— voyons ce que nous proposent les machines à noyau. La plus simple est le plus proche voisin : son principe est le suivant : pour déterminer la classe d'une donnée, on décide qu'elle est de même classe que l'exemple le plus proche. « Le plus proche » se définit par l'intermédiaire d'une fonction noyau qui mesure donc la proximité, ou la ressemblance, entre deux données.
Les plus proches voisins peuvent aussi prendre en considération les K exemples les plus proches du point à prédire et que l'on affecte la classe majoritaire, parmi ces K exemples. Cela est illustré sur la figure ci-dessous :

10. Ici, « simple » au sens mathématique, *i.e.*, l'équation qui représente le modèle : dans cette acception, rien n'est plus simple qu'une équation linéaire.



on a repris les exemples de la figure précédente et on a coloré chaque point du plan avec la classe qui lui est prédite par la méthode des K plus proches voisins pour $K = 1$, $K = 2$, $K = 3$ et $K = 5$. Pour $K = 1$, chaque point prend la couleur de l'exemple dont il est le plus proche.

Pour $K = 2$, on considère les deux exemples les plus proches ; trois cas peuvent se produire : les deux exemples les plus proches sont rouges : dans ce cas, le point est rouge ; les deux exemples les plus proches sont bleus : dans ce cas, le point est bleu : les deux exemples sont de couleur différente : dans ce cas, impossible d'affecter une couleur en suivant cette procédure : c'est une zone d'indécision, représentée ici en vert.

Une telle zone d'indécision est présente quand K est pair. Outre la parité de K, cette zone d'indécision est due au fait que les deux voisins les plus proches sont considérés avec une égale importante, même si l'un des voisins et plus proche que l'autre du point considéré. Cela peut paraître étonnant : si un point est tout près d'un exemple rouge et que son second plus proche voisin est bleu et éloigné, il peut paraître normal de lui affecter la couleur rouge.

Si l'on formalise un peu le cas où il y a deux classes. Plutôt que rouge et bleu, on peut associer les valeurs numériques -1 à la classe bleu et $+1$ à la classe rouge. Notons $y(x)$ la (vraie) classe de la donnée x et $\hat{y}(x)$ la classe prédite pour la donnée x ; $y(x)$ est inconnue, mis à part pour les exemples si on suppose que l'on connaît parfaitement la classe des exemples, *i.e.*, il n'y a pas d'erreur, ou de « bruit ».

Si l'on considère les K plus proches voisins : notons $\mathcal{V}_K(x)$ l'ensemble des K exemples les plus proches de x. La classe prédite est alors la somme des classes des K plus proches voisins :

$$\hat{y}(x) = \text{sgn}(\sum_{x \in \mathcal{V}_K(x)} y(x)) \ .$$

où

$$\text{sgn}(x) = \begin{cases} -1 \text{ si } x < 0 \\ 1 \text{ si } x \geq 0 \end{cases}$$

et où les x_i sont les exemples et x la donnée dont on veut prédire la classe.

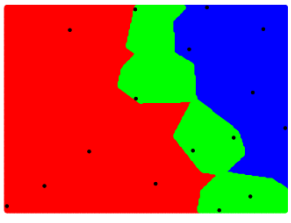
k mesure la distance, ou la dissimilarité, entre deux données. Le mot « distance » est utilisé dans son sens mathématique, *i.e.* une distance vérifie : la distance entre deux points est nulle si ces deux points sont les mêmes ; la distance du point 1 vers le point 2 est la même du point 2 vers le point 1 (symétrique) ; la distance du point 1 au point 3 est inférieure ou égale à la distance du point 1 vers le point 2, puis du point 2 vers le point 3 (inégalité triangulaire). Le mot « dissimilarité » relève de la notion de pseudo-distance : l'inégalité triangulaire n'est plus exigée : c'est la contrainte la plus forte, les deux autres étant assez évidentes à respecter dans la pratique.

Cette fonction k associe un réel à un couple de donnée : c'est là une définition de la notion de fonction noyau, qui n'impose pas d'autres conditions particulières sur k, comme c'est souvent le cas en apprentissage statistiques où le mot « noyau » implique certaines propriétés bien précises.

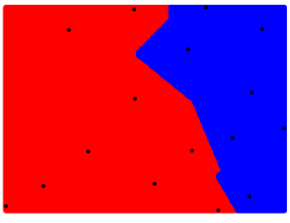
— Une autre approche consiste donc à considérer les voisins en fonction de leur proximité/ressemblance : comme on vient de l'exprimer, si parmi les K plus proches voisins, il y en a plus de bleus que de rouges, mais que les bleus sont beaucoup plus éloignés, il est assez raisonnable de considérer que la classe est rouge.

Ces approches sont dites « pondérées ». On peut considérer K voisins, voire tous les points, ce qui, d'un point de vue de l'implantation, est plus simple et, éventuellement, plus rapide : trouver les K points les plus proches d'un point donné est coûteux [11]. L'influence de chacun des exemples est pondérée par sa proximité avec le point que l'on veut prédire.

Sur l'exemple précédent, pour $K = 2$, on obtient :



K = 2



K = 2, noyau gaussien

11. une implantation efficace utilise des structures de données adaptées, tels les *kd-trees*. Néanmoins, ces structures de données ne fonctionnent que si la dimension P de l'espace des données est petite ($P \leq 5$).

La zone d'indécision a disparu et les couleurs affectées aux points sont cohérentes.
Cette méthode est dite fainéante (*lazy-learning* ou *memory-based-learning* ou *case based learning*) : le qualificatif de « fainéant » vient du fait qu'aucun calcul n'est réalisé avant de devoir prédire un point. *Stricto senso*, il n'y a pas d'apprentissage ici dans la mesure où « apprentissage » signifie que l'on construit une représentation, un modèle, du problème à partir des exemples, que l'on utilise ensuite à chaque fois que l'on veut effectuer une prédiction.
Formellement, on obtient cette fois-ci :

$$\hat{y}(x) = \sum_{x \in \mathcal{V}_K(x)} f(k(x, x_i))y(x) \quad .$$

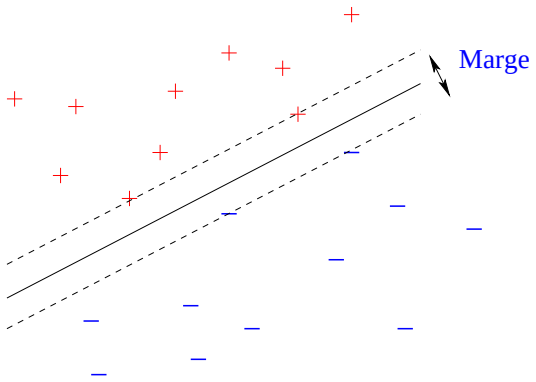
où f est une certaine fonction de la fonction noyau ; ce peut être la fonction identité, ou quelque chose de plus compliquée. Il est très courant de rencontrer ici une fonction gaussienne qui donne un poids élevé aux voisins les plus proches, à peu près négligeable aux voisins plus éloignés ; cet éloignement au-delà duquel un voisin est négligé est réglé par l'écart-type (ou largeur de bande σ) :

$$f(k(x, x')) \equiv e^{-\frac{k(x,x')^2}{2\sigma^2}}$$

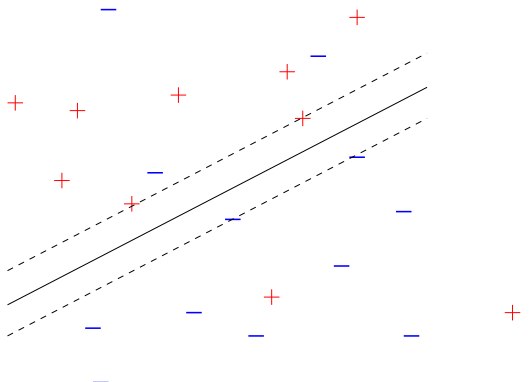
Ainsi, un voisin x_i de x pour lequel $k(x, x_i)$ est supérieur à 3σ est négligé. Cette définition est très naturelle si k représente la distance euclidienne ; l'utilisation d'une gaussienne en place de f a pour effet de fixer un horizon au-delà duquel l'influence d'un point sur un autre n'est plus sensible.
L'idée étant simple, elle a été inventée et déclinée à de nombreuses reprises : approche localement pondérée, locfit, loess, ...

— quand on augmente le nombre de voisins pris en compte, on observe que le séparateur tend à devenir linéaire.

— Maintenant que l'on a présenté tout ce contexte, on peut aborder les célèbres SVM. Si l'on reprend la figure vue plus haut où les deux classes sont séparables linéairement, on avait posé la question : quelle séparateur linéaire choisir parmi toutes ces droites ? La réponse SVM est : la droite qui passe le plus loin à la fois des bleus et des rouges, autrement dit, celle qui passe bien au milieu entre les deux classes. C'est alors un exercice de géométrie que de calculer la distance entre une droite donnée et l'exemple le plus proche de chacune des deux classes.
Sur la figure suivante, la séparatrice est indiquée, ainsi que les deux droites qui lui sont parallèles de chaque côté, telles qu'aucune autre droite parallèle plus proche de la séparatrice ne contienne un exemple de chacune des classes. L'écart entre ces deux droites se nomme la « marge » ; l'objectif d'un SVM est de déterminer la séparatrice telle que la marge soit la plus grande.



Donc, ayant formulé la distance entre une droite donnée et l'exemple le plus proche de chacune des deux classes, on peut chercher la droite qui maximise cette marge. C'est un problème de maximisation qui, si on a les connaissances nécessaires en mathématiques, est classique. Il n'y a donc plus qu'à le résoudre. Comme je suppose ici que mon lecteur, tout comme moi il y a quelques années, ne possède pas ces connaissances indispensables pour que la résolution du problème de maximisation soit évidente, on va faire un détour par quelques compléments. Avant cela, ajoutons qu'outre le problème simple où les deux classes sont linéairement séparables, la méthode des SVM doit, pour être vraiment utile, pouvoir s'appliquer quand les deux classes ne sont pas linéairement séparables, notamment le cas où elles le sont presque, à quelques exemples près, comme sur la figure suivante :



et les cas où les deux classes ne sont pas du tout linéairement séparables.

4 Les SVM déchiffrées

Informaticien, j'ai eu du mal à comprendre la littérature SVM car ne connaissant pas plusieurs notions importantes qui y étaient utilisées, mélangées les unes aux autres, considérées comme connues. Toutes ces notions sont supposées connues des lecteurs qui, s'il ne les connaissent pas, se retrouvent totalement perdus ; finalement, on est incapable de voir ce qui est nouveau de ce qui est connu depuis longtemps.

On peut résumer, et regrouper, les pré-requis en deux parties :

1. l'optimisation non linéaire :
   — la méthode de Lagrange, le lagrangien et les multiplicateurs de Lagrange ;

— les notions de problèmes primal et dual ;
— la notion de problème convexe et les méthodes de résolution des problèmes convexes. Les problèmes convexes sont considérés comme faciles à résoudre ; en outre, ces problème ont l'immense avantage de n'avoir qu'un seul optimum qui est donc global.

2. L'analyse fonctionnelle, notamment les notions d'espaces de Hilbert et les espaces de Hilbert à noyau reproduisant.

Quand on connaît tout cela, on comprend que la réelle nouveauté dans les SVM est [12] :
— la formulation du problème de classification supervisée comme un problème convexe (c'est-à-dire, la maximisation de la marge est formulée comme un problème convexe) ;
— l'utilisation de noyau pour obtenir des séparateurs non linéaires ;
— la constatation que les données ne sont manipulées qu'au travers d'un noyau, jamais individuellement (≪ astuce du noyau ≫)
— la construction d'arguments expérimentaux : ça marche mieux que les meilleurs algorithmes connus à l'époque, et c'est facile à utiliser.

On va maintenant aborder ces connaissances qui doivent être connues si on veut pouvoir vraiment comprendre ce que sont les SVM, et espérer comprendre une grande partie des travaux en apprentissage automatique de nos jours (depuis la seconde moitié des années 1990 en fait).

4.1 L'optimisation non linéaire

Loin de moi l'idée de faire ici une introduction exhaustive à l'optimisation non linéaire : c'est impossible en moins de plusieurs centaines de pages ! Je veux néanmoins lister les points importants, utiles à la compréhension des machines à noyau. Cela fait, j'indique comment cela s'utilise dans le cadre des SVM.

La première notion à connaître, la plus importante ici, est la méthode de Lagrange. Cette méthode transforme un problème d'optimisation sous contraintes en un problème d'optimisation sans contraintes : la fonction à optimiser (fonction objectif) et combinée avec les contraintes, cette combinaison se nommant le lagrangien.

On peut poser le problème d'optimisation comme suit :
— on se donne une fonction f définie sur $\mathbb{R}^P$, dite ≪ fonction objectif ≫ ;
— on cherche le point de $\mathbb{R}^P$ où f est optimale (minimale ou maximale) ;

12. J'ajouterai que le succès des SVM est du à plusieurs choses :
— une approche mathématique formelle, très élégante quand on la comprend, ce qui est nouveau dans le domaine de l'apprentissage automatique à l'époque, et a ouvert une nouvelle ère de l'apprentissage automatique ;
— des preuves expérimentales extrêmement convaincantes : l'application ≪ brutale ≫ des SVM a mieux résolu le problème de classification de caractères manuscrits, étudié notamment depuis plus de 10 ans par la communauté réseau de neurones ;
— la mise à disposition d'implantation libre, sources ouverts, de la méthode qui permettait à tout un chacun d'utiliser les SVM sans vraiment comprendre de quoi il s'agissait, comme une boîte noire ;
— une campagne de ≪ publicité ≫ scientifique : en effet, en recherche scientifique aussi, la communication est très importante.

— on pose des contraintes sur l'ensemble des points admissibles : x est dans le demi-plan positif ; x est situé sur telle courbe ou dans telle zone de $\mathbb{R}^P$ plus généralement.

Pour donner un exemple très simple dans $\mathbb{R}^2$: soit le problème dénoté $\mathcal{P}$ consistant à trouver le point $(x, y)^*$ auquel la fonction $f(x, y) = x^2 + y^2 + y - 1$ est minimale avec la contrainte $x^2 + y^2 \leq 1$, c'est-à-dire que le minimum doit appartenir au disque de rayon unité, centré à l'origine. Les couples (x, y) qui satisfont la condition et qui sont donc susceptibles d'être une solution au problème d'optimisation, sont dits « faisables » ou « acceptables »

On écrit la contrainte sous forme canonique : $x^2 + y^2 - 1 \leq 0$.

On écrit alors le lagrangien $\mathcal{L}$:

$$\mathcal{L} = \underbrace{x^2 + y^2 + y - 1}_{\text{Fonction objectif}} + \lambda(\underbrace{1 - x^2 - y^2}_{\text{Contrainte}})$$

Le minimum $(x, y)^*$ du problème initial et le minimum de $\mathcal{L}$ en $(x, y, \lambda)^*$ coïncident.

Pour ce qui est du vocabulaire, λ est dénommé un « multiplicateur de Lagrange ». Il y en a autant qu'il y a de contraintes. Ce sont des réels positifs.

On peut sauter ce paragraphe en première lecture.

On donne ici une intuition sur l'écriture du lagrangien. On veut minimiser une fonction f avec une certaine contrainte $x^2 + y^2 - 1 \leq 0$.

Pour toute solution acceptable (qui respecte cette contrainte), $1 - x^2 - y^2 \geq 0$. λ étant positif, le terme $\lambda(\underbrace{1 - x^2 - y^2}) \geq 0$. Comme on veut minimiser f, le minimum de $\mathcal{L}$ sera obtenu pour $\lambda = 0$ et alors $\mathcal{L}(x, y) = f(x, y)$.

On peut donc voir ce terme qui est ajouté à la fonction à minimiser dans le lagrangien comme une pénalité : si elle est nulle, le lagrangien est égal à la fonction à minimiser ; sinon, le lagrangien est strictement supérieur à la fonction à minimiser.

Pour ceux qui connaissent, le lien avec un terme de régularisation est immédiat.

Comme on le sait, comme pour toute fonction, $\mathcal{L}$ est optimal (minimal ou maximal) quand sa dérivée par rapport à chacune de ses variables est nulle. Sur l'exemple précédent, on obtient :

$$\frac{\partial \mathcal{L}}{\partial x} = 2\ x - 2\lambda\ x = 0 \tag{1}$$

$$\frac{\partial \mathcal{L}}{\partial y} = 2\ y + 1 - 2\lambda\ y = 0 \tag{2}$$

(1)$= 0 \iff x = 0$ ou $\lambda = 1$.

Si $\lambda = 1$, (2) entraîne $1 = 0$. Donc, la seule possibilité est que $x = 0$.

À l'optimum, on a (*complementary slackness conditions*) $\lambda(1 - x^2 - y^2) = 0$ qui se ré-écrit :

$$\lambda > 0 \Rightarrow x^2 + y^2 = 1 \tag{3}$$

$$\lambda = 0 \Rightarrow x^2 + y^2 < 1 \tag{4}$$

$$\tag{5}$$

L'ensemble de ces conditions (1)-(4) constituent les conditions de Kuhn-Tucker.

— Si $x^2 + y^2 = 1$, $y = \pm 1$:

— si $y = 1$, alors $\lambda = \frac{3}{2}$.
— Si $y = -1$, alors $\lambda = \frac{1}{2}$.

On a deux solutions possibles : $(0, 1)$ et $(0, -1)$.

— Si $x^2 + y^2 < 1$, alors $\lambda = 0$. Donc, (2) entraîne $y = -\frac{1}{2}$. Donc, une troisième solution est $(x, y) = (0, -\frac{1}{2})$.

Ces trois solutions donnent les optimaux ; parmi ceux-ci, il faut trouver le/les minima. On calcule donc l'image par f de ces trois points :

— $f(0, 1) = 1$
— $f(0, -1) = -1$
— $f(0, -\frac{1}{2}) = -\frac{5}{4}$

Le minimum est donc $(0, -\frac{1}{2})$.

Il y a deux familles de problèmes d'optimisation que l'on sait résoudre exactement : les problèmes linéaires et les problèmes convexes. Sorti de là, on n'a pas de méthode générale pour trouver la solution exacte [13] au problème d'optimisation.

Un problème linéaire est un problème dans lequel la fonction objectif et les contraintes mettent en jeu des équations dans lesquelles il n'y a pas de produit de variables, ni même d'exponentiation de variables : ce sont juste des polynômes de degré 1.

Un problème convexe est un problème dans lequel la fonction objectif et les contraintes mettent en jeu des équations dans lesquelles il n'y a rien de plus compliqué que des produits de deux variables (une variable au carré, ou un produit de deux variables différentes). (Un problème linéaire est un cas particulier de problème convexe.)

Le problème d'optimisation utilisé comme exemple ci-dessus est convexe.

Concernant sa résolution, un problème linéaire peut être résolu par différentes méthodes, notamment par programmation linéaire, soit le simplexe et les méthodes de point intérieur pour les approches exactes.

Pour un problème convexe, on dispose de différentes méthodes : méthode de Lagrange, méthode de point intérieur, méthode de gradient, ... En effet, un problème convexe a un et un seul optimum et la fonction est dérivable ; donc on peut suivre le gradient puisqu'il existe et est continu, et celui-ci nous mène forcèment à l'optimum. Le point délicat est que dans certaines zones, le gradient peut être très faible : on peut alors perdre la direction de la plus forte pente, mais surtout perdre beaucoup de temps à effectuer de très petits pas (même si c'est dans la bonne direction).

4.2 SVM linéaire et optimisation non linéaire

Revenons aux SVM, dans le cas linéaire pour l'instant. Dans le cadre de la classification supervisée, appliquer une SVM consiste à trouver le séparateur linéaire qui produit la plus grande marge, tout en minimisant le nombre d'erreur de classifications sur les exemples.

Pour cela, on formule le problème d'optimisation suivant :

— maximiser la marge et minimiser le nombre d'exemples mal classés,

13. par exact, on veut dire l'optimum global, le trouver de manière certaine et savoir quand on l'a trouvé.

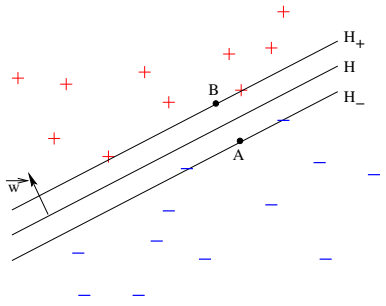
— en respectant au mieux la contrainte que la classe de chaque exemple est bien prédite.

4.2.1 Cas linéairement séparable

Pour $P = 1$, le séparateur linéaire H est une droite dans le plan, d'équation $y = ax + b$; on cherche alors a et b qui optimisent le mieux les critères SVM. Pour un exemple (x_i, y_i), le SVM prédit la classe $\hat{y}(x_i) = \text{sgn}(ax_i + b)$; on constate que $(ax_i + b)y_i \geq 0$ si la classe de x_i est correctement prédite, < 0 si la classe de l'exemple est mal prédite.

Supposons que les exemples soient linéairement séparables par une droite H. Considérons H_- la droite parallèle la plus proche de H qui contient un exemple de classe -1, H_+ la droite parallèle la plus proche de H qui contient un exemple de classe $+1$. H_- et H_+ ont pour équation $y = ax + b_-$ et $y = ax + b_+$ respectivement. On cherche la distance entre H_- et H_+, qui est la marge, que l'on veut maximiser. On sait aussi que H est au milieu entre H_- et H_+, donc $|b_+ - b| = |b - b_-|$ et la marge est $|b_+ - b_-|$.

Prenons $B \in H_+$ quelconque, et $A \in H_-$ le point de B appartenant à H_- :



— $B \in H_+$ est donc de classe $+1$, $A \in H_-$ de classe -1 ;
— ...
— on a $\overrightarrow{OB} = \overrightarrow{OA} + \overrightarrow{AB}$ où $||\overrightarrow{AB}||$ est la marge et $\overrightarrow{AB}$ est perpendiculaire à H ;
— la droite orthogonale à $y = ax + b$ est $y = -\frac{1}{a}x + b$;
— donc la droite portée par $\overrightarrow{AB}$ est d'équation $y = -\frac{1}{a}x + b$;
— et $\overrightarrow{AB}$ s'écrit $\lambda \begin{pmatrix} a \\ -1 \end{pmatrix} = a\lambda \begin{pmatrix} 1 \\ \frac{-1}{a} \end{pmatrix}$
....

Maximiser la marge revient à minimiser la norme du vecteur $\overrightarrow{w}$, tout en classant bien les exemples (on suppose ici qu'ils sont tous classables sans erreur : cas séparable).

Minimiser cette norme, son carré, ou la moitié de son carré fournit la même solution. La dernière (moitié du carré) permet de simplifier pas mal de choses.

On doit donc résoudre :

$$\min \frac{1}{2}||w||^2 \tag{6}$$

$$\text{tel que } y_i(< w, x_i > +b) \geq 1 \text{ [14]} \tag{7}$$

14. Si vous vous demandez pourquoi 1 plutôt que 0, c'est une excellente question. Voir [Pontil and Verri, 1998].

Le lagrangien de ce problème est tout simplement :

$$\mathcal{L} = \frac{1}{2}||w||^2 - \sum_i \lambda_i y_i (< w, x_i > + b) + \sum_i \lambda_i \tag{8}$$

que l'on veut minimiser. C'est un problème convexe, on peut le faire... mais on pousse le raisonnement plus loin pour pouvoir traiter le cas non linéairement séparable. Pour cela, on exprime le problème dual comme suit.

Les λ_i sont les multiplicateurs de Lagrange. Il y en a autant que d'exemples, N.

En écrivant :

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_i \lambda_i y_i = 0 \tag{9}$$

$$\frac{\partial \mathcal{L}}{\partial w} = w - \sum_i \lambda_i y_i x_i = 0 \tag{10}$$

En substituant (9) et (10) dans (8), on obtient le lagrangien dual :

$$\mathcal{L}_D = \sum_i \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j < x_i, x_j > \tag{11}$$

Dans le cas d'un problème convexe, le maximum du lagrangien dual est atteint pour le minimum du lagrangien primal (8). Donc, maximiser ce dual fournit le minimum du primal.

Ce problème peut se ré-écrire comme :

$$\max -\frac{1}{2} \Lambda \mathbf{D} \Lambda + \sum_i \lambda_i \tag{12}$$

$$\text{tel que } \sum_i y_i \lambda_i = 0 \tag{13}$$

$$\lambda_i \geq 0 \tag{14}$$

$\mathbf{D}$ est une matrice dont les termes sont $D_{i,j} = y_i y_j < x_i, x_j >$. Λ est le vecteur dont les éléments sont les λ_i.

Le dual est un problème convexe, ayant donc un seul optimum qui est donc global.

(10) entraîne : $\mathbf{w} = \sum_i \lambda_i y_i \mathbf{x_i}$.

Les vecteurs supports sont les exemples dont le multiplicateur de Lagrange est non nul. (ces exemples soutiennent [15] la marge). Ils sont situés au bord de la marge.

Pour prédire la classe d'une donnée $\mathbf{x}$, on calcule $\hat{y}(x) = \text{sgn}\ (\sum_i \lambda_i y_i (< x_i, x > + b))$.

15. "support" en anglais.

4.2.2 Cas non séparable linéairement

Dans le cas où des exemples d'une classe sont mélangés dans l'autre classe, on peut chercher le séparateur linéaire avec la contrainte de minimiser le nombre d'exemples mal classés. On obtient alors un autre problème de minimisation avec contraintes, pour lequel on peut écrire le lagrangien.

Les contraintes s'expriment comme suit :

$$\begin{cases} < w, x_i > +b \geq 1 - \xi_i \text{ si } y_i = +1 \\ < w, x_i > +b \leq -1 + \xi_i \text{ si } y_i = -1 \\ \xi_i \geq 0 \end{cases}$$

et la fonction objectif à minimiser est maintenant $\frac{1}{2}||w||^2 + C \sum_i \xi_i$, où C est une constante à fixer judicieusement [16].

Les ξ_i sont des réels positifs. $\xi_i = 0$ si l'exemple x_i est bien prédit ; $\xi_i > 0$ s'il est mal prédit. ξ_i quantifie de combien un exemple est mal classé : plus ξ_i est petit, plus x_i est mal classé et éloigné de la marge.

À nouveau, on construit le lagrangien de ce problème, d'où on déduit le problème dual :

$$\max -\frac{1}{2}\lambda \mathbf{D} \lambda + \sum_i \lambda_i \text{tel que } \sum_i y_i \lambda_i = 0 \tag{15}$$

$$0 \leq \lambda_i \leq C \tag{16}$$

Sa résolution fournit des multiplicateurs de Lagrange $\lambda_i \in [0, C]$. Si $\lambda_i = C$, alors $\xi_i \neq 0$ ce qui signifie donc que l'exemple x_i est mal classé ; si $\lambda_i \in [0, C[$, alors la classe de x_i est bien prédite.

On a maintenant résolu le problème du SVM linéaire ; on va passer à la recherche d'un séparateur non linéaire. Pour cela, quelques compléments de mathématiques sont utiles.

4.3 Les espaces de Hilbert à noyau reproduisant

Dans l'espace euclidien $\mathbb{R}^P$, le produit scalaire de deux vecteurs v et w est $< v, w >= \sum_{i=1}^{i=P} v_i w_i$.

Rappelons qu'un produit scalaire se définit comme suit : on considère d'abord un espace vectoriel [17] $\mathcal{D}$. Un produit scalaire k sur $\mathcal{D}$ vérifie :

- k est une application qui associe un réel positif à un couple de vecteurs : $k : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}^+$
- $k(v, w) = 0 \iff v = 0$ ou $w = 0$
- k est symétrique : $k(v, w) = k(w, v)$
- $k(\alpha v, w) = \alpha k(v, w), \forall \alpha \in \mathbb{R}$

16. arbre qui cache la montagne !

17. un terme qui provoque parfois des frayeurs tellement cette notion est mal enseignée : un espace vectoriel est simplement un ensemble de vecteurs (des flêches) que l'on est autorisé à additionner deux à deux, ou à multiplier un vecteur par un nombre (pour l'allonger ou le rétrécir).

— $k(v+w,z) = k(v,z) + k(w,z)$

Dans le cas euclidien, cette fonction k se définit par : $k(v,w) = \sum_i v_i w_i$. On peut vérifier facilement que cette fonction respecte bien les propriétés ci-dessus.

Maintenant, on peut se poser la question : peut-on caractériser de la manière la plus générale qui soit les propriétés que doit respecter une fonction k pour qu'elle corresponde à un produit scalaire ?

La réponse est : k doit être un noyau, *i.e.* une fonction qui associe un réel positif à un couple d'éléments d'un ensemble et qui est symétrique et définie positive. « Définie positive » est une propriété assez complexe à comprendre et surtout difficile à vérifier : face à un noyau, la symétrie est facile à vérifier, mais qu'il soit défini positif est quelque chose de bien plus compliqué à démontrer [18].

Concrétement [19], cela signifie que si on a un noyau k symétrique et défini positif, alors il existe un espace vectoriel $\mathcal{E}$ et une fonction $\Phi : \mathcal{D} \to \mathcal{E}$ tel que $v, w \in \mathcal{D}, k(v,w) =< \Phi(v), \Phi(w) >_{\mathcal{E}}$. Reprenons calmement. Typiquement, la fonction Φ transforme une donnée d'un espace de représentation (ici, $v \in \mathcal{D}$, c'est-à-dire que v est une donnée représentée par ses attributs et que donc $\mathcal{D}$ est l'espace dont les axes de coordonnées sont les attributs caractérisant v) dans un autre espace de représentation. On a l'habitude de rencontrer des projections qui envoient une donnée dans un espace de plus petite dimension ; on peut aussi l'envoyer vers un espace de plus grande dimension : on parle alors de « pulvérisation » et, pour une raison qui sera explicitée bientôt, c'est ce que l'on fait ici : Φ pulvérise les données dans un espace de très grande dimension, voire de dimension infinie. Cet espace où l'on pulvérise est l'espace vectoriel noté $\mathcal{E}$ ci-dessus. Dans cet espace $\mathcal{E}$, il existe un produit scalaire : on le note $< ., . >_{\mathcal{E}}$ pour indiquer que ce n'est pas le produit scalaire de l'espace euclidien. Comme $\Phi(v) \in \mathcal{E}$ et $\Phi(w) \in \mathcal{E}$, $< \Phi(v), \Phi(w) >_{\mathcal{E}}$ a un sens très clair. Alors, toute la magie/beauté (ou tout ce qu'il faut savoir) réside exactement ici : on n'a pas besoin de connaître Φ ou de savoir calculer $\Phi(v)$ ou le représenter pour pouvoir calculer le produit scalaire de la pulvérisation dans $\mathcal{E}$ de deux vecteurs de $\mathcal{D}$. En effet, on a dit ci-dessus que connaître et savoir calculer k suffit : k simule ce produit scalaire ; on dit que k « reproduit » le produit scalaire et que c'est un « noyau reproduisant ». Ainsi, comme on a dit que l'on pouvait pulvériser dans un espace de dimension infinie, c'est une excellente nouvelle car si on peut utiliser un noyau k pour simuler le produit scalaire dans un espace de dimension infinie, et si le calcul de k ne pose pas de problème pratique, on peut calculer le produit scalaire de données pulvérisées dans un espace de dimension infinie.

k étant symétrique et définie positif, l'espace $\mathcal{E}$ dont il reproduit le produit scalaire possède des propriétés intéressantes (que nous ne détaillerons pas) : c'est un espace de Hilbert [20].

Quelques exemples de noyau :

— le produit scalaire euclidien

— le noyau polynomial : $k_{poly}(x_1, x_2) = (1 + < x_1, x_2 >)^P$

— le noyau RBF : $k_{rbf}(x_1, x_2) = e^{-\frac{||x_1 - x_2||^2}{2\ \sigma}}$

18. il faut qu'il vérifie la propriété de Mercer.
19. c'est normal que ce qui suit ne soit pas immédiatement évident.
20. d'où cette dénomination d'espace de Hilbert à noyau reproduisant.

Reste à comprendre à quoi ça peut servir.

4.4 SVM non linéaire

À quoi ça peut servir la pulvérisation dans des espaces de dimension infinie et ces noyaux reproduisants ? On comprend tout de suite si on connaît le théorème de Cover qui dit : dans une tâche de classification supervisée, plus la dimension des données est grande (plus ils ont d'attributs linéairement indépendants), plus la probabilité que les classes soient linéairement séparables est grande [Cover, 1965].

Dit autrement, prenons des données non linéairement séparables ; pulvérisons-les dans un espace de très grande dimension (une infinité de dimension même !) et la probabilité de pouvoir les séparer linéairement augmente.

Maintenant que l'on a cette motivation, si l'on regarde de près les équations du problème dual de la SVM linéaire, on constate que les données n'y apparaissent que dans des produits scalaires, jamais individuellement (voir la matrice notée $\mathbf{D}$ plus haut). Aussi, l'idée est d'une simplicité extrême (cette idée se nomme l'≪ astuce du noyau ≫) : on se donne un noyau symétrique défini positif et au lieu de calculer des produits scalaires euclidiens, on utilise ce noyau qui calcule des produits scalaires sur les données pulvérisées dans un espace de (très) grande dimension. Si on n'est pas en dimension infinie, rien ne garantit que les données seront linéairement séparables, mais la probabilité qu'elles le soient a augmenté, et le nombre d'exemples mal classés dans cet espace est réduit. On peut donc, dans cet espace, appliquer les techniques des SVM linéaires sur des exemples bruités.

C'est tout en ce qui concerne la théorie. Pour la pratique, la difficulté est de trouver un noyau qui marche pour les données que l'on veut traiter.

Dans la légende des SVM, il y a l'idée que l'algorithme des SVM ne peut être surpassé. C'est bien évidemment complètement faux. L'algorithme des SVM est optimal d'un certain point de vue, mais on peut avoir un autre point de vue, pas très différent, et obtenir des algorithmes plus performants.

5 Pour aller plus loin

Dans cette note, je n'ai touché que la surface de la problématique des machines à noyau et en particulier, les machines à vecteurs supports. Si mon lecteur a atteint cette section après avoir lu le reste, il est peut-être désireux de poursuivre sur le sujet. Je propose ici diverses pistes.

Concernant les lectures, je ne saurais trop recommendé des articles tels que celui de Pontil et Verri [Pontil and Verri, 1998], le Bennett et Campbell [Bennett and Campbell, 2000], puis le livre de Shawe-Taylor et Cristianini [Shawe-Taylor and Cristianini, 2004].

Pour la mise en œuvre, je l'ai dit plus haut, à mon avis, le succès des SVM est largement du à la mise à disposition de logiciel libre implémentant la méthode dès 1998. Aujourd'hui, le logiciel `libSVM` représente l'état de l'art ; il est disponible sur la page web de ses auteurs, mais aussi comme un paquet Ubuntu ; il est facilement utilisable en ligne de commande ou via un

logiciel comme R. Actuellement, les algorithmes les plus performants pour résoudre les SVMs sont des descentes de gradient stochastiques [Bottou and Lin, 2007].

De multiples formulations du problème ont été proposées durant les années 2000, basées sur d'autres mesures d'erreur ($\sum |y(x) - \hat{y}(x)|$ au lieu de l'erreur quadratique habituelle $\sum (y(x) - \hat{y}(x))^2$, ou la perte-charnière [21] pour ne citer que les deux plus connues) et sur d'autres termes de régularisation (régularisation ℓ_1 : $\sum |w_i|$ qui produit des solutions parcimonieuses, de la régularisation par groupes, ...). La recherche a été foisonnante et le demeure sur ces questions, du fait des applications qui poussent à l'utilisation de certaines mesure d'erreur et certains termes de régularisation.

Un problème pratique comme la découverte automatique de la valeur du coefficient de régularisation λ la plus adéquate a été très étudié. Initialement obtenue par résolution du problème pour plusieurs valeurs de λ puis sélection de la meilleure, il a été découvert que l'on pouvait calculer la solution correspondant à toutes les valeurs de λ à la fois pour certaines mesures d'erreur et régularisations. Imaginons l'espace des paramètres (les composantes du vecteur $\mathbf{w}$) soit $\mathbb{R}^N$; à chaque valeur de λ correspond une solution, donc un point dans cet espace ; le fait est que la fonction qui associe ce point à λ est continue : donc, l'ensemble des points associé à toutes les valeurs de $\lambda > 0$ forment une courbe continue que l'on nomme le « chemin de régularisation » du problème. Dans certains cas, cette courbe est constituée de segments de droite (voir [Rosset and Zhu, 2007] pour les conditions). Connaissant les points (valeurs de λ) correspondant aux extrêmités des segments, on peut facilement trouver les paramètres pour toutes les valeurs de λ comprise dans l'intervalle par une simple interpolation linéaire. Le chemin de régularisation du problème de SVM décrit dans cette note est ainsi composé d'un ensemble de segments de droite, de même que le problème identique avec une régularisation ℓ_1 auquel est associé l'algorithme connu sous l'acronyme LARS [Efron et al., 2004].

Crédits images : la figure page 3 (spirale colorée) provient de `http://www.iro.umontreal.ca/~lisa/twiki/pub/Sandbox/RevueApprentissageAutomatique/LLE-example.jpeg`. J'ai réalisé toutes les autres figures.

21. *hinge-loss.*

Références

[Bennett and Campbell, 2000] Bennett, K. and Campbell, C. (2000). Support vector machines : Hype or hallelujah ? *ACM SIGKDD Explorations*, 2(2).

[Bottou and Lin, 2007] Bottou, L. and Lin, C.-J. (2007). Support vector machine solvers. In Bottou, L., Chapelle, O., DeCoste, D., and Weston, J., editors, *Large Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA.

[Cover, 1965] Cover, T. (1965). Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, EC-14(3) :326–334. `http://www.stanford.edu/~cover/papers/paper2.pdf`.

[Efron et al., 2004] Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, 32(2) :407–499.

[Hastie et al., 2001] Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The elements of statistical learning*. Springer.

[Pontil and Verri, 1998] Pontil, M. and Verri, A. (1998). Properties of support vector machines. 10(4) :955–974. `http://www.mitpressjournals.org/doi/abs/10.1162/089976698300017575`.

[Preux, 2006] Preux, P. (2006). Notes de cours de fouille de données. `http://www.grappa.univ-lille3.fr/~ppreux/fouille`.

[Rosset and Zhu, 2007] Rosset, S. and Zhu, J. (2007). Piecewise linear regularized solution paths. *Annals of Statistics*, 35(3) :1012.

[Shawe-Taylor and Cristianini, 2004] Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge University Press, 3 edition.