Une nouvelle méthode d'apprentissage : Les SVM. Séparateurs à vaste marge.

Antoine Cornuéjols

Maître de Conférence à L'I.I.E.
Et chercheur au LRI, Université de Paris-Sud, Orsay

Ces dernières années, il a soufflé comme un vent de révolution en apprentissage artificiel. Du côté des applications, on s'est tourné vers la fouille de très grandes bases de données réelles, avec les défis attenants en termes de complexité en calcul et en espace, de prétraitement nécessaire des données, et de présentation des résultats aux utilisateurs. Avec aussi de nouveaux besoins d'interactivité entre utilisateurs et outils d'apprentissage. Mais le bouleversement n'a pas semblé moindre du côté conceptuel. Depuis 1995, nul n'est sensé ignorer les enseignements de Vapnik, et il est devenu quasiment impossible de ne pas essayer les SVMs ou le boosting sur un nouveau problème d'apprentissage. De quoi s'agit-il ?

Nous nous intéressons ici aux SVM. Étant donnée leur extraordinaire fortune, qui dépasse les cercles de la communauté de l'apprentissage artificiel, cet article a pour objectif de les présenter de manière simple, mais assez complète, pour un public plus large. Une remarque préalable : SVM est l'acronyme de Support Vector Machines en anglais, méthode et terme inventés par Vapnik principalement. Nous traduisons ici ce terme par Séparateurs à Vaste Marge. Nous verrons pourquoi.

1. Un retour aux origines : le perceptron

Le perceptron, inventé par Rosenblatt, date du tout début des années soixante. Le problème que traite cet algorithme est le suivant : supposons que nous ayons une séquence d'observations $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, …, $\boldsymbol{x}_m$, décrites par des mesures sur un ensemble prédéfini d'attributs (par exemple : le poids, l'âge, la tension, …), chacune de ces observations étant affectée à une classe C prise dans $\{C_1, C_2\}$ (par exemple : « risque cardio-vasculaire élevé », « risque cardio-vasculaire faible »). À partir de cet échantillon d'apprentissage, nous cherchons à trouver les paramètres d'un automate tel que celui de la figure 1 afin de permettre de prédire la classe de nouvelles observations à l'avenir. Il s'agit d'une tâche d'apprentissage supervisé de concept ; supervisé car on fournit à l'algorithme d'apprentissage l'étiquette associée à chaque observation ; de concept car il s'agit d'apprendre à distinguer deux classes d'observations, ou encore une classe contre le reste.

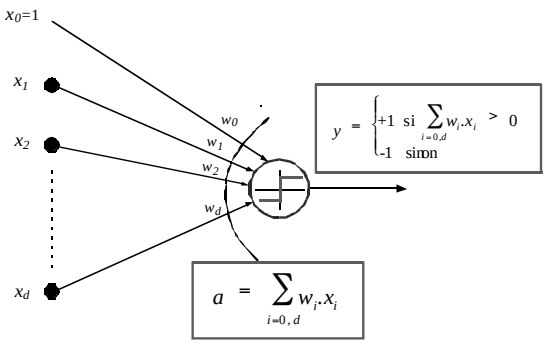

Figure 1 : Schéma d'un perceptron à un « neurone ». Le neurone reçoit en entrée la description d'une observation $\boldsymbol{x}$, décrite par d attributs $x_1,\ldots x_d$. Chacune de ces composantes x_i est pondérée par un poids synaptique w_i, (on ajoute aussi une entrée fictive $x_0 = 1$ pour permettre l'obtention de séparatrice ne passant pas par l'origine dans l'espace des observations) et la sortie y du neurone, –1 ou +1 est décidée par le signe de cette somme pondérée.

Cet automate est en fait l'ancêtre des réseaux connexionnistes. Il ne comporte qu'un " neurone " recevant en entrée les valeurs des d attributs de description, opérant une combinaison linéaire de ces entrées grâce à une pondération par le vecteur des poids des « synapses » et produisant en sortie un signal de valeur –1 si la somme pondérée calculée est inférieure à un certain seuil q, et la valeur 1 sinon. On peut traduire cela par l'équation (1) suivante :

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_O \begin{cases} \geq 0 & \Rightarrow \quad h(x) = +1 \\ < 0 & \Rightarrow \quad h(x) = -1 \end{cases} \quad (1)$$

Cette équation montre immédiatement, que le perceptron est en fait un système de recherche d'une séparatrice linéaire dans l'espace des attributs. Idéalement, cette séparatrice doit séparer parfaitement les observations affectées à une classe de celles affectées à l'autre classe (on parlera fréquemment de la classe + et de la classe – ainsi que d'exemples positifs et d'exemples négatifs). L'apprentissage revient ici à chercher un vecteur de poids w permettant la séparation des exemples positifs et des exemples négatifs dans l'échantillon d'apprentissage.

Pour ce faire, on définit d'abord une fonction de coût traduisant la plus ou moins bonne séparation des exemples des deux classes, puis on utilise une méthode de descente de gradient dans l'espace des poids w pour minimiser cette fonction de coût. Si la fonction de coût prend une forme quadratique, la descente de gradient est garantie de converger vers l'optimum global. En gros, cela signifie qu'il existe une « meilleure » séparatrice des deux classes d'exemples d'apprentissage, celle qui « passe par le milieu ». Nous aurons l'occasion de revenir sur cette notion.

2. Qu'est-ce que l'apprentissage ?

Une première question concerne l'espace des concepts que l'on peut ainsi apprendre. Il est évidemment très restreint dans le cas du perceptron puisque seuls les concepts correspondant à des séparatrices linéaires dans l'espace des attributs sont apprenables. Plus généralement, il faudra toujours s'interroger sur l'adéquation de l'espace des hypothèses accessibles à l'algorithme d'apprentissage à l'espace des concepts cibles potentiels pour l'application considérée. Mais l'espace des hypothèses intervient aussi à un autre titre.

En effet, le but poursuivi en apprentissage inductif est de parvenir à classer correctement les futures observations grâce à la connaissance apprise (ici sous la forme d'un automate) à propos d'un échantillon limité de données. Une question centrale de l'apprentissage est donc de savoir comment utiliser les données pour avoir une bonne performance en généralisation, c'est-à-dire sur les observations inconnues à venir.

Remarquons que cette question, c'est à peine si elle nous a effleuré lorsque nous avons décrit l'algorithme du perceptron. Il a semblé absolument naturel que la séparatrice qui se comporte le mieux vis-à-vis des données d'apprentissage, c'est-à-dire par exemple qui minimise le nombre d'exemples positifs ou négatifs mal classés, soit aussi celle qui permettra au mieux de classer les observations à venir, mais encore inconnues. D'où l'algorithme d'apprentissage visant à minimiser la mesure de coût sur les exemples d'apprentissage. On appelle cette mesure un risque empirique car elle est mesurée empiriquement sur les données de l'échantillon d'apprentissage. Ce risque est la somme des coûts mesurés pour chaque exemple d'apprentissage et prend donc la forme :

$$R_{emp}(h) = \frac{1}{m} \sum_{i=1}^{m} l[h(\boldsymbol{x}_i), u_i]$$

où $\boldsymbol{x}_i$ est un des m exemples d'apprentissage, u_i l'étiquette associée et l la fonction de perte mesurant la distance entre la réponse produite $h(\boldsymbol{x}_i)$ et la réponse désirée u_i. (On notera ici une hypothèse fondamentale sous-tendant toute la théorie statistique de l'apprentissage, à savoir que les exemples sont tirés indépendamment les uns des autres, selon une même distribution).

Mais ce qui nous intéresse *in fine*, c'est de minimiser l'espérance des coûts sur les observations à venir, ce que l'on appelle le *risque réel*, et qui prend la forme :

$$R_{réel} = \int_{X \times U} l[h(\boldsymbol{x}_i), u_i] \; dF(\boldsymbol{x}, u)$$

où cette fois l'intégrale prend en compte la distribution inconnue F des exemples sur X x U , le produit cartésien de l'espace des observations X et de l'espace des étiquettes U.

La question qui doit alors nous préoccuper est de savoir si lorsque nous choisissons une hypothèse minimisant le risque empirique, nous minimisons aussi le risque réel, ce qui est notre vrai objectif. On appelle ce critère de sélection d'hypothèse : *principe de minimisation du risque empirique.* Il est à la base de la majorité des techniques d'apprentissage inductif (à côté de principes inductifs liés à la décision bayésienne). Est-il justifié ? C'est la question centrale des investigations de Vapnik durant une vingtaine d'années.

La réponse est que le lien entre le risque empirique mesuré et le risque réel espéré est fonction de la « richesse » de l'espace des hypothèses accessibles à

l'apprenant. Si l'espace est très contraint, c'est-à-dire que le choix d'une hypothèse est très limité, le risque empirique mesuré sur la meilleure hypothèse est probablement proche du risque réel. En revanche, si l'espace d'hypothèses est riche et donc permet de trouver facilement une hypothèse de risque empirique faible, alors cela ne donne pas de garantie sur la performance en généralisation, c'est-à-dire le risque réel, de cette hypothèse. Pour des raisons théoriques qui sortent du cadre de cet article, on mesure la richesse d'un espace d'hypothèses, ou encore sa capacité, par un nombre : *la dimension* de Vapnik-Chervonenkis. En notant cette dimension dVC, on peut exprimer le lien entre le risque empirique d'une hypothèse et son risque réel par une équation du type :

$$P\left(\max_{h\in H} \left|R_{réel}(h) - R_{emp}(h) \;\geq\; \varepsilon\right|\right) \;<\; G(\mathrm{d}_{VC}, m, \varepsilon)$$

Cette équation signifie que la probabilité que l'écart entre le risque empirique mesuré et le risque réel visé dépasse une certaine valeur e est bornée par *G*, une fonction de la richesse de l'espace des hypothèses mesurée par $\mathrm{d_{VC}}$, de la taille m de l'échantillon d'apprentissage et de l'écart admis ε.

On remarquera que cette majoration du risque réel n'est obtenue qu'en probabilité, car tout dépend de la représentativité de l'échantillon d'apprentissage, et c'est seulement en probabilité (avec une probabilité croissante avec la taille de l'échantillon) que cet échantillon est représentatif de la distribution des exemples dans la tâche d'apprentissage.

Cette équation exprime d'une manière mathématique un dilemme bien connu : pour avoir une chance de trouver une bonne hypothèse, il faut un espace d'hypothèses riche (le risque empirique dans la somme ci-dessus sera alors faible ou nul), malheureusement, plus l'espace d'hypothèses est riche, et moins il y a de garantie que le risque réel soit proche du risque empirique mesuré. L'une des difficultés de l'apprentissage est donc de savoir régler ce compromis. Naturellement, l'idéal est de disposer d'informations *a priori* sur l'espace des concepts cible. Dans ce cas, on peut choisir un espace d'hypothèses adapté, c'est-à-dire dans lequel on sait *a priori* qu'existe une hypothèse de risque empirique faible ou nul, et suffisamment limité pour qu'il y ait de bonnes garanties que le risque réel sera proche du risque empirique.

De fait la théorie nous apporte même une information supplémentaire. La taille de l'échantillon d'apprentissage suffisante pour avoir, avec une confiance donnée, un risque réel proche de moins de ε du risque empirique, croît en $1/\varepsilon^2$ pour un espace d'hypothèses quelconque. Donc, si on veut diviser par deux l'écart maximal probable ε, il faut multiplier par quatre le nombre d'exemples d'apprentissage. Cela caractérise la vitesse de convergence de l'apprentissage. Mais si l'on est certain que l'espace d'hypothèses considéré contient le concept cible, alors la vitesse de convergence est plus forte, en $1/\varepsilon$. On a donc doublement intérêt à savoir cerner à l'avance le bon espace d'hypothèses.

En résumant, nous devons donc résoudre le dilemme suivant : d'une part, nous avons intérêt à considérer un espace d'hypothèses le plus riche possible puisque cela augmente nos chances d'y trouver le concept cible, ce qui au passage améliore la convergence en apprentissage ; mais, d'autre part, nous avons intérêt aussi à limiter autant que possible la richesse de l'espace d'hypothèses sous peine de distendre inconsidérément le lien entre le risque empirique mesuré et le risque réel et donc sous peine d'apprendre une hypothèse n'ayant rien à voir avec le vrai concept cible.

En dehors de bénéficier d'un délit d'initié nous donnant des informations *a priori* sur le bon espace d'hypothèses, pouvons-nous avoir une méthode générale permettant de résoudre le dilemme posé ?

3. Les SVM (séparateurs à vastes marges)

Lorsqu'une observation est fournie en entrée à un perceptron, celui-ci produit une réponse qui indique seulement si cette observation est du côté de la classe « + » (sortie 1) ou du côté de la classe « – » (sortie –1). Si maintenant nous considérons une variante du perceptron dans laquelle la sortie non seulement fournit le signe (le côté dans lequel tombe l'observation par rapport à l'hyperplan séparateur) mais fournit aussi une distance à l'hyperplan, pouvons-nous tirer parti de ce supplément d'information ?

Cette fois-ci : $h(\boldsymbol{x}) = \boldsymbol{w}\boldsymbol{x} + w_o$, et non plus : $h(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{w}\boldsymbol{x} + w_o)$

En supposant qu'il existe un hyperplan permettant de séparer les exemples positifs des exemples négatifs, nous n'allons plus nous contenter d'en trouver un, mais

nous allons en plus chercher parmi ceux-ci celui qui passe « au milieu » des points des deux classes d'exemples. Pourquoi ? Intuitivement, cela revient à chercher l'hyperplan le « plus sûr ». En effet, supposons qu'un exemple n'ait pas été décrit parfaitement, une petite variation ne modifiera pas sa classification si sa distance à l'hyperplan est grande.

Formellement, cela revient à chercher un hyperplan dont la distance minimale aux exemples d'apprentissage est maximale (voir la figure 2). On appelle cette distance « marge » entre l'hyperplan et les exemples. Comme on cherche à maximiser cette marge, on parlera de *méthode des séparateurs à vaste marge.*

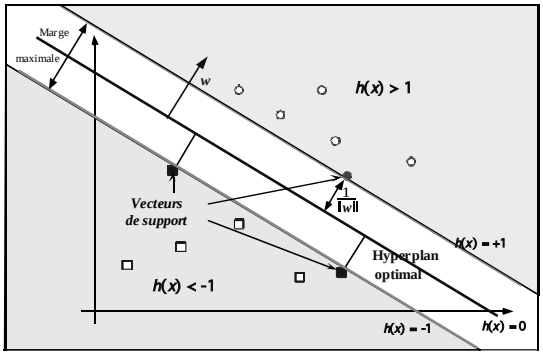

Figure 2 : L'hyperplan optimal séparant les points de deux classes est celui qui passe « au milieu » de ces classes, c'est-à-dire dont la distance aux points les plus proches est maximale. Ces exemples les plus proches qui suffisent à déterminer cet hyperplan sont appelés vecteurs de support, ou encore exemples critiques. La distance séparant l'hyperplan de ces points est appelée « marge ».

Cet hyperplan optimal est défini par le vecteur de poids w vérifiant l'équation :

$$\underset{\boldsymbol{w},w_0}{\text{Arg max}}\ \min\left\{\|\boldsymbol{x}-\boldsymbol{x}_i\| \ :\ x\in\Re^d,\ (\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}\ +\ w_O)\ =\ 0,\ i=1,...,m\right\}$$

Pour cet hyperplan, la marge vaut $1/\|\boldsymbol{w}\|$, et donc la recherche de l'hyperplan optimal revient à minimiser $\|\boldsymbol{w}\|$, soit à résoudre le problème suivant qui porte sur les paramètres $\boldsymbol{w}$ et w_0 :

$$\begin{cases} \text{minimiser} & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{sous les contraintes} & \mathrm{u}_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i\ +\ w_O)\ \geq\ 1,\ \ i=1,...,m \end{cases} \qquad (2)$$

Cette écriture du problème, appelée *formulation primale,* implique le réglage de *d*+1 paramètres, *d* étant la dimension de l'espace des entrées *X*. Cela est possible avec des méthodes de programmation quadratique pour des valeurs de *d* assez petites, mais devient inenvisageable pour des valeurs de *d* dépassant quelques centaines. Heureusement, il existe une transformation de ce problème dans une formulation duale que l'on peut résoudre en pratique.

D'après la théorie de l'optimisation, un problème d'optimisation possède une forme duale dans le cas où la fonction objectif et les contraintes sont strictement convexes. Dans ce cas, la résolution de l'expression duale du problème est équivalente à la solution du problème original. Ces critères de convexité sont réalisés dans le problème défini ci-dessus.

Après transformation, le problème devient celui de la recherche de paramètres a vérifiant le système d'équations :

$$\begin{cases} \underset{\alpha}{\text{Max}}\ \left\{\sum_{i=1}^{m}\alpha_i\ -\ \frac{1}{2}\ \sum_{i,j=1}^{m}\alpha_i\alpha_j u_i u_j(\boldsymbol{x}_i.\boldsymbol{x}_j)\right\} \\ \alpha_i\geq 0,\quad i=1,...,m \\ \sum_{i,j=1}^{m}\alpha_i u_i\ =\ 0 \end{cases} \qquad (3)$$

L'hyperplan solution correspondant peut alors s'écrire :

$$h(\boldsymbol{x})\ =\ (\boldsymbol{w}^*.\boldsymbol{x})\ +\ w_0^*\ =\ \sum_{i=1}^{m}\alpha_i^* u_i.(\boldsymbol{x}_i.\boldsymbol{x}_j)\ +\ w_0^* \qquad (4)$$

où les α_i^* sont solution de l'équation (3) et w_0^* est obtenue en utilisant n'importe quel exemple critique $(\boldsymbol{x}_c,u_c)$ dans l'équation :

$$\alpha_i\ \left[u_i.((\boldsymbol{x}_i.\boldsymbol{w}^*)+w_0)\ -\ 1\right]\ =\ 0$$

À ce point, remarquons deux choses. D'abord, que l'hyperplan solution ne requiert que le calcul des produits scalaires $\langle\boldsymbol{x}_i.\boldsymbol{x}\rangle$ entre des vecteurs de l'espace d'entrée *X*. Cette observation aura de profondes répercussions. Ensuite, la solution ne dépend plus de la dimension *d* de l'espace d'entrée, mais de la taille *m* de l'échantillon de données et même du nombre m_c d'exemples critiques qui est généralement bien inférieur à *m*. Les méthodes d'optimisation quadratique standard suffisent donc pour de nombreux problèmes pratiques.

Tout cela est très bien, et l'on se croit autorisé à

être fort satisfait, sauf que si l'esprit critique ne s'est pas complètement endormi, on ne voit guère en quoi cet enrichissement de l'algorithme du perceptron a résolu le problème posé par le dilemme entre l'utilité d'avoir un espace d'hypothèses riche et la nécessité de limiter cette richesse.

De fait les perceptrons ont un pouvoir expressif très limité qui semble devoir les confiner à des problèmes très particuliers, comme l'avaient fort bien montré en leur temps Minsky et Papert dans leur célébrissime ouvrage *Perceptrons*, publié la première fois en 1969.

Cependant, on n'a retenu depuis que le côté pessimiste du livre de Minsky et Papert. Ils avaient en effet souligné que si les entrées du perceptron résultaient d'un prétraitement non linéaire sur l'espace des attributs définissant les observations, la séparatrice trouvée par le perceptron dans cet espace de redescription pouvait correspondre à une séparatrice non linéaire dans l'espace des attributs initiaux. Le seul problème, mais il était de taille, était de trouver les bonnes fonctions de redescription. C'est de ce constat que se sont nourris les efforts ultérieurs pour trouver une règle d'apprentissage pour des perceptrons multicouches non linéaires. Or l'étude des SVM allait être l'occasion de reprendre les travaux sur la découverte de fonctions de redescription permettant de passer d'un espace de description initial des données à un espace de redescription dans lequel il est possible de trouver une séparatrice linéaire des données.

Une approche combinatoire permet en effet d'imaginer une telle transformation sans connaissance *a priori* sur le problème.

En effet, plus la dimension de l'espace de description est grande, plus la probabilité de pouvoir trouver un hyperplan séparateur entre les exemples et les contre-exemples est élevée. En transformant l'espace d'entrée en un espace de redescription de très grande dimension, éventuellement infinie, il devient donc possible d'envisager d'utiliser la méthode des SVM.

Notons Φ une transformation non linéaire de l'espace d'entrée X en un espace de *redescription* $\Phi(X)$:

$$\boldsymbol{x} = (x_1,...,x_d)^{\mathrm{T}} \quad \text{a} \quad \Phi(\boldsymbol{x}) = (\Phi_1(\boldsymbol{x}),...,\Phi_d(\boldsymbol{x}),...)^{\mathrm{T}}$$

où x_k est la k-ième composante du vecteur $\boldsymbol{x}$, et où nous avons voulu indiquer que généralement, le vecteur image $\Phi(\boldsymbol{x})$ est de dimension supérieure à d, la dimension de l'espace d'origine.

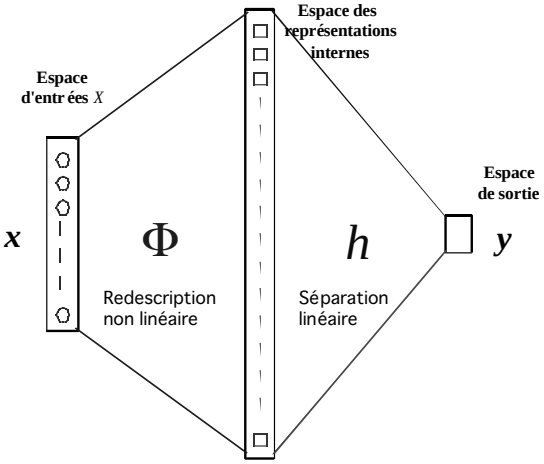

Figure 3 : Le passage par une redescription des données peut permettre une séparation linéaire des exemples.

Le problème d'optimisation se transcrit dans ce cas par :

$$\begin{cases} \underset{\alpha}{\mathrm{Max}} \left\{ \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j u_i u_j \left\langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \right\rangle \right\} \\ \alpha_i \geq 0, \quad i = 1,...,m \\ \sum_{i,j=1}^{m} \alpha_i u_i = 0 \end{cases} \tag{5}$$

où $\langle .,. \rangle$ dénote le produit scalaire.

L'équation de l'hyperplan séparateur dans le nouvel espace devient :

$$h(\boldsymbol{x}) = (\boldsymbol{w}^*.\boldsymbol{x}) + w_0^* = \sum_{i=1}^{m} \alpha_i^* u_i . \left\langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \right\rangle + w_0^*$$

où les coefficients α_i^* et w_0^* sont obtenus comme précédemment par résolution de l'équation (5).

Tout cela est très bien, sauf que l'objection immédiate du praticien concerne le produit scalaire

$$\left\langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \right\rangle$$

qui devient rapidement impossible à calculer quand la dimension de $\Phi(X)$ augmente (sans parler du cas de la

dimension infinie), ceci d'autant plus que l'on utilisera des transformations non linéaires des descripteurs d'entrée. Pour donner une idée, supposons que l'on cherche à classer des images de 16 x 16 pixels (par exemple pour faire de la reconnaissance d'écriture manuscrite), et que l'on suppose nécessaire pour cela de tenir compte des corrélations entre 5 pixels quelconques au plus dans les images. L'espace de redescription, qui contient toutes les combinaisons de 5 pixels quelconques parmi 256, est alors de dimension de l'ordre de 10^{10}. Calculer des produits scalaires dans un tel espace est impraticable.

Il se trouve heureusement que l'on peut dans certains cas s'arranger pour littéralement court-circuiter le passage par les calculs dans l'espace de redescription. En effet, il existe des fonctions bilinéaires symétriques positives $K(\boldsymbol{x},\boldsymbol{y})$, appelées fonctions noyau, faciles à calculer et dont on peut montrer qu'elles correspondent à un produit scalaire dans un espace de grande dimension. Lorsqu'une telle correspondance est exploitable, le problème d'optimisation (5) est équivalent au problème suivant :

$$\begin{cases} \underset{\alpha}{\text{Max}} \left\{ \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j u_i u_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \right\} \\ \alpha_i \geq 0, \quad i = 1,...,m \\ \sum_{i,j=1}^{m} \alpha_i u_i = 0 \end{cases} \tag{6}$$

dont la solution est l'hyperplan séparateur d'équation :

$$h(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i^* u_i . K(\boldsymbol{x}_i, \boldsymbol{x}_j) + w_0^*$$

où les coefficients α_i^* et w_0^* sont obtenus comme précédemment par résolution de l'équation (6).

Par exemple, il peut être montré que la *fonction noyau polynomiale* : $K(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}.\boldsymbol{y})^n$ réalise implicitement un produit scalaire dans l'espace des descripteurs correspondant à tous les produits d'exactement n dimensions. Ainsi pour n=2 et $\boldsymbol{x}, \boldsymbol{y} \in \Re^2$, nous avons :

$$(\boldsymbol{x}.\boldsymbol{y})^2 = (x_1^1, x_2^2, \sqrt{2}x_1x_2)(y_1^1, y_2^2, \sqrt{2}y_1y_2)^{\mathrm{T}} = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{y}) \rangle$$

qui correspond au changement de description par la fonction : $\Phi(\boldsymbol{x}) = (x_1^1, x_2^2, \sqrt{2}x_1x_2)$, et qui nous fait donc passer par une transformation non linéaire dans un espace de dimension trois, au lieu de deux initialement.

Autre exemple : la fonction noyau $K(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x},\boldsymbol{y} + c)^n$ avec $c \geq 0$ correspond à un produit scalaire dans l'espace des descripteurs correspondant à tous les produits d'ordre $\leq n$.

Les fonctions noyau décrites impliquent des calculs dans l'espace X, donc dans un espace de dimension d. Ils sont par conséquent faciles à calculer. Mais comment déterminer quelle fonction noyau aisément calculable est associée à un espace de redescription $\Phi(X)$ dont on pense qu'il peut être intéressant pour trouver un séparateur linéaire des données ?

En fait, la démarche est inverse : on cherche des fonctions noyau dont on peut avoir la garantie *a priori* qu'elles correspondent à un produit scalaire dans un certain espace qui agira comme un espace de redescription, mais qui restera virtuel, jamais explicité. Cela signifie que l'utilisateur devra opérer par essais et erreurs : essayer des fonctions noyau associées à des produits scalaires dans des espaces de redescription et voir si elles permettent l'obtention de bonnes séparatrices. Si cet aspect tâtonnant de la méthode peut sembler peu satisfaisant, en revanche, le choix de la fonction noyau devient le seul paramètre à régler, contrairement à d'autres techniques d'apprentissage.

Toute fonction bilinéaire symétrique n'est pas nécessairement une fonction noyau dans la mesure où elle peut ne pas correspondre à un produit scalaire dans un espace. Il faut pour cela qu'une certaine condition mathématique appelée condition de Mercer soit vérifiée. En pratique, quelques familles de fonctions paramétrables sont connues pour satisfaire à ces conditions, et l'utilisateur de SVM effectue des tests sur ces familles de fonctions pour déterminer celle qui convient le mieux pour son application. Le tableau 1 donne les fonctions noyau les plus courantes.

La solution s'exprime sous la forme :

$$h(\boldsymbol{x}) = \sum_{i=1}^{m_c} \alpha_i^* u_i . K(\boldsymbol{x}_i, \boldsymbol{x}_j) + w_0^*$$

Fonction noyau	Forme fonctionnelle	Commentaire
- polynomiale	$K(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}.\boldsymbol{y} + c)^n$	La puissance *n* est déterminée *a priori* par l'utilisateur.
- fonctions à base radiale	$K(\boldsymbol{x},\boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}\right)$	L'écart type σ^2, commun à tous les noyaux, est spécifié a priori par l'utilisateur.
- fonctions sigmoïdes	$K(\boldsymbol{x},\boldsymbol{y}) = \tanh((a\ (\boldsymbol{x}.\boldsymbol{y}) - b))$	Le théorème de Mercer n'est vérifié que pour certaines valeurs de *a* et *b*.

Tableau 1 : Les fonctions noyau les plus courantes avec leurs paramètres.

Il est clair que les fonctions noyau encodent des connaissances essentielles sur le problème d'induction étudié, en particulier :

- Une mesure de similarité sur les données.
- La forme fonctionnelle des fonctions de décisions possibles.
- Le type de contrôle réalisé sur les hypothèses : par exemple les fonctions noyau Gaussiennes pénalisent les dérivées de tous les ordres et favorisent donc les solutions « régulières ».
- Le type de covariance dans l'espace des entrées stipulant comment des points en divers endroits de l'espace sont liés les uns aux autres. On peut ainsi chercher des fonctions noyau invariantes pour certaines transformations de l'espace d'entrée, comme par exemple la rotation.

Il faut donc les choisir avec soin en essayant de traduire grâce à elles le maximum de connaissances préalables dont on dispose sur le problème et sur les données.

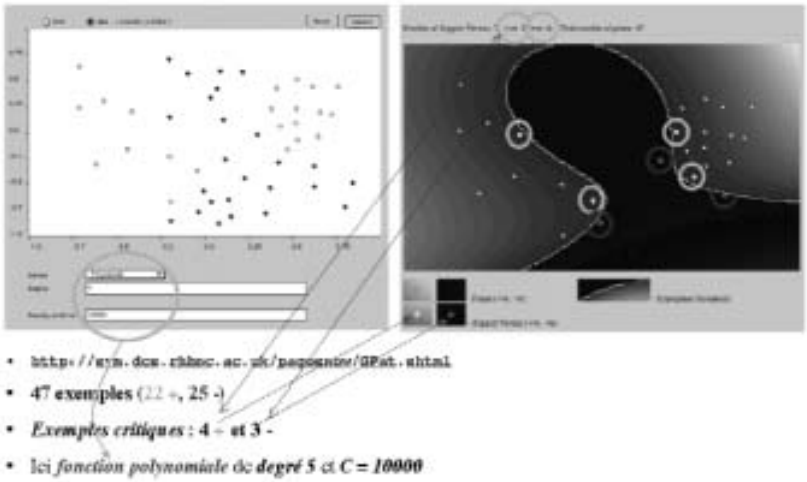

Figure 4 : Dans la fenêtre de gauche, quarante-sept exemples d'apprentissage (vingt-deux de la classe « + » et 25 de la classe « – ») ont été disposés par l'utilisateur. La fenêtre de droite montre la frontière de décision obtenue avec un SVM de noyau polynomial de degré 5 avec une constante $C = 10000$ (pénalisant beaucoup les exemples mal classés). La frontière obtenue s'appuie sur sept exemples critiques, dont quatre dans la classe « + » et trois dans la classe « – ».

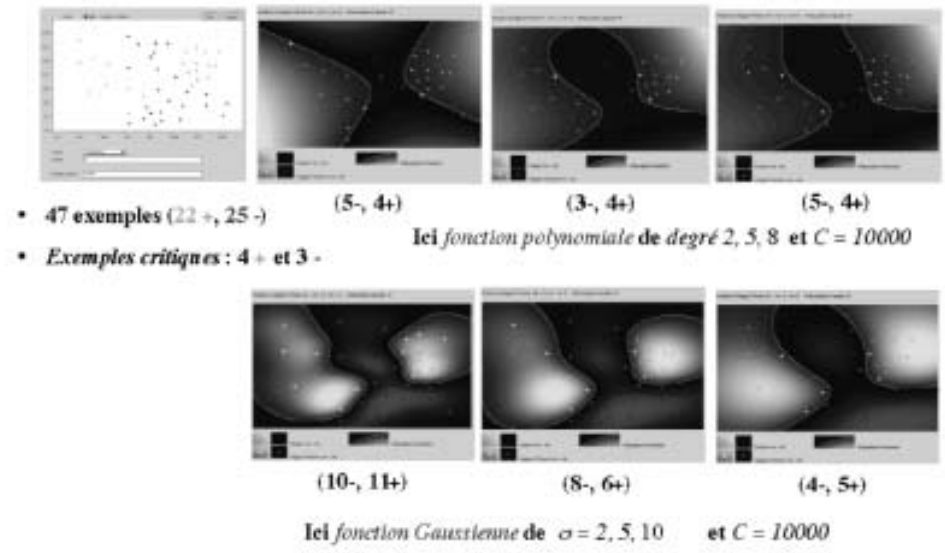

Figure 5 : Pour les mêmes exemples d'apprentissage que ceux de la figure 4, on voit l'effet de différents choix de la fonction noyau (polynomiale de degré 2, 5 et 8 en haut, et Gaussienne d'écart-type 2, 5, et 10 en bas). Le nombre d'exemples critiques de chaque classe est indiqué entre parenthèses en-dessous de chaque imagette.

4. La mise en œuvre des SVM

La réalisation d'un programme d'apprentissage par SVM se ramène essentiellement à résoudre un problème d'optimisation impliquant un système de résolution de programmation quadratique dans un espace de dimension conséquente. C'est pourquoi ces programmes utilisent des méthodes spéciales pour y parvenir de manière efficace et il n'est pas recommandé de chercher à réaliser soi-même un tel programme. Le lecteur pourra trouver un ensemble d'outils disponibles gratuitement sur le site : http://www.kernel-machines.org/ .

L'utilisation de ces programmes revient surtout à sélectionner une bonne famille de fonctions noyau et à régler les paramètres de ces fonctions (par exemple l'exposant pour les fonctions noyau polynomiale, ou bien l'écart type pour les fonctions à base radiale). Ces choix sont le plus souvent faits par une technique de validation croisée, dans laquelle on estime la performance du système en la mesurant empiriquement sur des exemples n'ayant pas été utilisés en cours d'apprentissage. L'idée est de chercher les paramètres permettant d'obtenir la performance maximale. Cependant, en opérant de cette manière, c'est-à-dire en utilisant l'échantillon de données pour régler les paramètres du système, on risque d'identifier des paramètres suradaptés à cet échantillon de données. C'est pourquoi on utilise un troisième jeu de données, n'ayant pas servi au processus d'optimisation du système, pour évaluer la performance vraie du système.

Si la mise en œuvre d'un algorithme de SVM est en général peu coûteuse en temps, il faut cependant compter que la recherche des meilleurs paramètres peut requérir des phases de test assez longues.

Sur ce thème, nous finirons en notant que, pour des raisons de simplification de cette présentation, nous avons laissé de côté un détail d'importance. En effet, il arrive que, même en utilisant un espace de redescription, l'on ne puisse trouver une séparatrice linéaire séparant parfaitement les exemples d'une classe de ceux de l'autre. Cela peut être dû par exemple à des descriptions des données imparfaites ou bruitées. Dans ce cas, on se contentera de trouver une séparatrice optimale, ne laissant que peu d'exemples mal classés de chaque côté de la séparatrice choisie. Cette notion introduit à son tour un nouveau paramètre d'optimisation, sous la forme d'une constante C, qu'il faut donc aussi optimiser.

5. Mais finalement, pourquoi ça marche ?

Nous avons démarré cet article en discutant la légitimité du principe inductif reposant sur la minimisation du risque empirique, mesuré sur les données d'apprentissage, pour sélectionner la meilleure hypothèse, celle qui se comportera bien sur les observations à venir. L'analyse théorique a en effet montré que le lien entre le risque empirique et le risque réel était fonction de la richesse de l'espace d'hypothèse. Si celui-ci est trop riche, la corrélation entre le risque mesuré et le risque réel est hasardeuse, nullement garantie. Il faut donc régler au mieux un compromis entre avoir un espace trop pauvre pour contenir des hypothèses intéressantes et avoir un espace trop riche, pour lequel plus aucune garantie n'existe sur la performance à venir de l'hypothèse choisie sur la base d'un échantillon limité de données. Or qu'avons-nous fait avec les SVM ?

Il semble qu'en voulant échapper aux limitations des perceptrons classiques, et en passant par l'utilisation d'un espace de redescription, nous soyons passés d'un extrême à l'autre. Certes le stratagème du passage par un espace de redescription des données nous aurait permis d'identifier aisément une fonction séparatrice des données, quelles qu'elles soient, mais nous aurions en même temps perdu toute assurance qu'elle se révèle performante sur les futures observations. C'est ici que nous allons remettre en lumière la notion de marge évoquée plus haut, et qui se trouve au centre des réflexions en apprentissage artificiel en ce moment.

Il faut commencer par remarquer que la marge autour d'une séparatrice linéaire est associée à la richesse de l'espace des hypothèses. En effet, plus la marge est grande, et plus il est difficile de faire passer une séparatrice linéaire avec cette marge entre les données des deux classes (revoir la figure 2). Intuitivement, en cherchant une séparatrice de marge maximale entre les données, nous nous obligeons de fait à considérer un espace d'hypothèses plus limité et cela nous permet de maintenir malgré tout un lien entre le risque empirique et le risque réel. Cela correspond à la justification initialement avancée par Vapnik pour les SVM. Selon lui, cette méthode permettait de régler directement le compromis entre adéquation aux données et limitation de la richesse de l'espace des hypothèses.

Cependant, il a été souligné depuis que, dans ce schéma, la richesse de l'espace des hypothèses, au lieu d'être déterminée *a priori* avant de commencer l'apprentissage, comme c'est le cas lorsque l'on choisit par exemple *a priori* une architecture dans un réseau connexionniste, est ici fixée *a posteriori*, à partir des données d'apprentissage. La philosophie, et plus que ça, est alors différente. L'analyse théorique initiale de Vapnik et des autres théoriciens était destinée à borner la différence entre risque empirique et risque réel quelle que soit la distribution des données. C'était donc une analyse dans le pire cas. L'analyse des SVM semble requérir une nouvelle approche dans laquelle on s'intéresse aux données observées et où l'on cherche à exploiter au mieux la chance que l'on a eue de tomber sur une distribution particulière des données telle que le révèle l'échantillon d'apprentissage. Si cette distribution est favorable, alors la corrélation entre risque empirique et risque réel peut être beaucoup plus étroite que ne le prédit l'analyse dans le pire cas. Le manque de place nous interdit de rentrer davantage dans les détails ici.

Si des cadres théoriques se mettent en place pour rendre compte du succès des méthodes à vaste marge, notons que le débat est encore largement ouvert et offre des opportunités de recherche pour ceux que l'analyse théorique de nature statistique attire.

6. Conclusion

La méthode des SVM est aisée d'emploi. Les logiciels de SVM sont disponibles sur Internet et les expériences sont faciles à réaliser. On obtient souvent en les utilisant des résultats comparables à ceux obtenus avec d'autres techniques et les paramètres à régler sont moins nombreux. Cette méthode est applicable pour des tâches de classification à deux classes, mais il existe des extensions pour la classification multiclasse. Plus encore, bien que nous n'en ayons pas parlé ici pour des raisons d'espace limité, les SVM peuvent également s'utiliser pour des tâches de régression, c'est-à-dire de prédiction d'une variable continue en fonction d'autres variables, comme c'est le cas par exemple dans de la prédiction de consommation électrique en fonction de la période de l'année, de la température, etc. Le champ d'application des SVM est donc large. Si ils ne sont pas la panacée peut être un peu vite annoncée, ils représentent une classe de méthodes très séduisantes. L'un des axes de recherche actuel est de parvenir à coder des

connaissances *a priori* dans ces systèmes, c'est-à-dire à mieux comprendre le rôle des fonctions noyau.

Par ailleurs, du point de vue conceptuel, la notion de marge a renouvelé la vision des méthodes inductives en général, et a stimulé tout un courant de recherche dont on peut espérer qu'il débouchera sur de nouvelles classes de méthodes, encore mieux comprises et encore plus adaptables à nos problèmes.

Bibliographie (très partielle):

- Anthony, M. and P. Bartlett (1999). Neural network learning : theoretical foundations, Cambridge University Press. (Pour ceux qui veulent aller voir de plus près les fondements théoriques des SVM.)
- Burges, C. (1998). « A tutorial on support vector machines for pattern recognition. » Data Mining and Knowledge Discovery 2(2): 121-167. (Un article d'introduction excellent malgré sa date de parution déjà ancienne).
- Cornuéjols, A. and L. Miclet (2002). L'apprentissage artificiel. Méthodes et concepts, Eyrolles. (Ouvrage général sur l'apprentissage artificiel. L'article présent est une version simplifiée du chapitre 9 de cet ouvrage.)
- Cristianini, N. and J. Shawe-Taylor (2000). Support Vector Machines and other kernel-based methods, Cambridge University Press. (Un bon ouvrage d'introduction aux SVM.)
- Schölkopf, B. and A. Smola (2002). Learning with kernels. Support vector machines, regularization, optimization, and beyond, MIT Press. (L'ouvrage le plus complet et le plus récent sur la question des SVM, et plus largement des méthodes à vaste marge).