Note du cours 4 de ML : Noyaux, SVM

Nicolas Baskiotis
MLIA, LIP6, Sorbonne Université

27 mars 2020

Résumé

Ces notes ne sont malheureusement pas un cours et ne peuvent remplacer complétement le cours magistral. Elles ont été écrites très rapidement, elles sont destinées à vous guider autant que possible dans la lecture des slides en soulignant le message important de chacun. J'ai essayé de donner des références pour tous les points techniques, n'hesitez pas à vous y reporter pour affiner votre compréhension.

1 Retour sur le perceptron

Slide 3 Rappel : le perceptron permet de trouver une séparatrice linéaire $f_{\mathbf{w}}(\mathbf{x}) =< \mathbf{x}, \mathbf{w} > +b$ par une descente de gradient stochastique (un seul exemple est examiné à chaque itération, tiré au hasard, pour la mise à jour des poids $\mathbf{w}$). La fonction de coût considérée est $L(\hat{y}, y) = max(0, -y\hat{y})$ (avec $\hat{y} = f_{\mathbf{w}}(\mathbf{x})$) : si pas d'erreur - $-y\hat{y} < 0$, le coût est nul et le gradient également, sinon le gradient est $-y\mathbf{x}$.

Le vecteur $\mathbf{w}$ de poids correspond à la normale de l'hyperplan séparateur : $< \mathbf{w}.\mathbf{x} >= 0$ seulement si $\mathbf{x}$ appartient à l'hyperplan séparateur. Le produit scalaire $< \mathbf{x}, \mathbf{w} >= cos(\mathbf{x}, \mathbf{w})\|x\|\|w\|$ permet de situer le point $\mathbf{x}$ par rapport à l'hyperplan. La règle de mise à jour des poids peut être interprétée géométriquement comme une correction de l'angle de l'hyperplan afin de rapprocher l'hyperplan du point mal classifié et ainsi de réduire itérativement l'erreur.

Slide 4-5 Trois problèmes principaux dans le cadre de cet algorithme : 1) dans le cas d'un jeu de données séparable, la séparatrice trouvée est généralement collée aux données - l'algorithme s'arrête dès qu'il n'y a plus d'erreurs ; Or, intuitivement une séparatrice plus centrée par rapport aux deux nuages de données devrait être plus robuste ; 2) il n'y a pas de tolérance aux erreurs, des points bruités peuvent participer au coût et gêner l'apprentissage (en pratique, un pas d'apprentissage ϵ décroissant est utilisé pour aboutir à une convergence) ; 3) seul des données linéairement séparables peuvent être traitées (sauf en augmentant manuellement la dimension des exemples par projection polynomiale ou autre).

L'approche SVM permet de répondre à ces 3 problèmes et c'est le sujet de ce cours.

2 Support Vector Machine (SVM) : principe

Slide 7 Le produit scalaire $< \mathbf{w}.\mathbf{x} >$ en plus de donner l'angle entre les 2 vecteurs indique également la distance par rapport à la séparatrice. Soit d la distance entre $\mathbf{x}$ et la séparatrice (rappel : la distance entre un point et un hyperplan est le minimum de la distance entre $\mathbf{x}$ et tous les points de l'hyperplan, atteint au point projeté orthogonale de $\mathbf{x}$ sur l'hyperplan). Soit A la projection orthogonale de $\mathbf{x}$ sur l'hyperplan, petit calcul simple de lycée : $\mathbf{x} = A + d\frac{\mathbf{w}}{\|\mathbf{w}\|}$, $\mathbf{x}$ est le translaté de A selon le vecteur directeur unité de l'hyperplan d'une distance d. On sait de plus que $< \mathbf{w}, A > +b = 0$ comme A se trouve sur l'hyperplan, donc $< \mathbf{w}, \mathbf{x} - d\frac{\mathbf{w}}{\|\mathbf{w}\|} > +b = 0$, donc $d = \frac{<\mathbf{w},\mathbf{x}>+b}{\|\mathbf{w}\|} = \frac{f(\mathbf{x})}{\|\mathbf{w}\|}$.

Ainsi, $f(\mathbf{x})$ est d'autant plus grand que $\mathbf{x}$ est éloigné de l'hyperplan.

Slide 8 Première idée importante des SVMs : la notion de marge. On aimerait avoir autour de l'hyperplan une zone de sécurité qui ne comporte aucun point de l'ensemble d'apprentissage. Plus cette zone - la marge - est grande, plus on aura confiance en la séparatrice apprise. Donc plutôt que de chercher une séparatrice d'orientation quelconque, on va chercher deux hyperplans parallèles (les droites rouges sur le dessin) collés l'un au données positives, l'autre aux données négatives, de façon à ce que entre les deux hyperplans il n'y ait aucun point de l'ensemble d'apprentissage. Notre séparatrice est alors l'hyperplan du milieu (droite noire), équidistant des deux hyperplans précédents à une distance γ. On définit une sorte de "tube" de rayon γ le plus grand possible.

Supposons le problème résolu, avec la donnée d'une séparatrice $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}.\mathbf{x} + b$ et d'une distance γ définissant les deux marges. On note qu'il existe alors une infinité de vecteurs $\mathbf{w}'$ qui définissent de manière identique la séparatrice et les deux marges - il suffit de prendre $\mathbf{w}' = \alpha\mathbf{w}$. La distance géométrique γ n'est pas changée par cette mise à l'échelle, par contre la distance *fonctionnelle* $f_{\mathbf{w}}(\mathbf{x})$ elle change. Cette distance fonctionnelle à peu d'intérêt, on préfère garantir l'unicité de la solution. On fixe donc arbitrairement que la distance fonctionnelle à la marge doit être de 1 : $f_{\mathbf{w}}(\mathbf{x}_1) = 1$ pour $\mathbf{x}_1$ sur la marge positive et -1 pour $\mathbf{x}_1$ sur la marge négative. Notez bien que cette contrainte n'est mise que pour obtenir une solution unique $\mathbf{w}$, elle ne change en rien la "vraie" distance géométrique, toujours de γ.

Examinons maintenant la signification de la norme de $\mathbf{w}$ par rapport à γ. Pour un problème parfait, γ est ainsi la distance entre d'une part l'hyperplan séparateur et le point positif le plus proche, et d'autre part le point négatif le plus proche. Pour ces deux points, $f_{\mathbf{w}}(\mathbf{x}) = +/-1$. Tous les points positifs du jeu de données sont tels que $f_{\mathbf{w}}(\mathbf{x}) \geq 1$ et les points négatifs tels que $f_{\mathbf{w}}(\mathbf{x}_2) \leq -1$, soit tels que $y f_{\mathbf{w}}(\mathbf{x}) \geq 1$ (en considérant des labels $y \in \{-1, 1\}$). Par ailleurs, si on prend $\mathbf{x}_1$ sur la marge positive et $\mathbf{x}_2$ sa projection sur l'hyperplan séparateur, $\mathbf{x}_1 - \mathbf{x}_2 = \gamma \frac{\mathbf{w}}{\|\mathbf{w}\|}$. De plus, $f_{\mathbf{w}}(x_1) - f_{\mathbf{w}}(x_2) = 1 = \mathbf{w}(\mathbf{x}_1 - \mathbf{x}_2) = \gamma \frac{\mathbf{w}.\mathbf{w}}{\|\mathbf{w}\|} = \gamma \|\mathbf{w}\|.$, soit $\|\mathbf{w}\| = \frac{1}{\gamma}$. Ainsi maximiser la marge équivaut à minimiser $\|\mathbf{w}\|$.

Ceci nous donne une nouvelle formulation du problème de classification posé : minimiser $\|\mathbf{w}\|$ tel que pour tous les points $\mathbf{x}^i$ du jeu de données $(\mathbf{w}\mathbf{x}^i + b)y^i \geq 1$. Dans la pratique, on minimise $\|\mathbf{w}\|^2$. Cette optimisation sous contraintes quadratique est la formulation de base des SVMs : trouver un hyperplan séparateur $\mathbf{w}$ avec une marge maximale $\frac{1}{\mathbf{w}}$ tel que tous les points soient bien classés.

Slide 9 Bien sûr les problèmes réels ne sont pas parfait et en raison du bruit on aimerait tolérer des points mal classés. On aimerait s'autoriser à faire quelques erreurs. On pourrait rajouter une pénalisation sur le nombre d'erreurs dans la minimisation de $\mathbf{w}$: $\|\mathbf{w}\|^2 + K\#erreurs$, mais une telle pénalisation transforme le problème en un problème d'optimisation combinatoire - quels points ignorés ? - qui est très difficile à résoudre (gradient nul comme pour le coût 0-1).

Slide 10 L'idée est de faire de l'optimisation continue en introduisant des variables appelées *slack - ressorts*. On introduit une telle variable ξ_i pour chaque exemple $\mathbf{x}^i$ de l'ensemble d'apprentissage et on modifie la contrainte associée : $(\mathbf{w}\mathbf{x}^i + b)y^i \geq 1 - \xi_i$ avec $\xi_i \geq 0$:

— lorsque $\xi_i = 0$, la variable n'a pas d'effet, la contrainte originale est vérifiée et le point est bien classé ;

— lorsque $\xi_i > 0$, alors le point $\mathbf{x}^i$ est du mauvais côté de la marge - il peut toujours être bien classé si $\xi_i \leq 1$, ou mal classé dans le cas contraire - la contrainte originale est violée.

On souhaite bien entendu qu'il y ait le plus possible de $\xi_i = 0$ et pour les quelques uns non nuls, qu'ils soient le plus petit possible. Du coup le problème de minimisation devient : minimiser $\|\mathbf{w}\|^2 + K \sum_{j=1}^{N} \xi_j$ tel que $(\mathbf{w}\mathbf{x}^i + b)y^i \geq 1 - \xi_i$, $\xi_i \geq 0$ pour tout $i \in \{1, \ldots, N\}$, N le nombre d'exemples d'apprentissage. La constante K permet de balancer entre une pénalisation importante des erreurs (K grand) et au contraire obtenir une marge plus grande mais avec beaucoup plus d'erreurs (K petit) : c'est un hyper-paramètre qui permet de régler l'expressivité du modèle (un $\mathbf{w}$ avec une norme grande à plus de dimensions exprimées, voir l'analogie en TME 3 avec le perceptron et la base de projection gaussienne), que l'on choisit par cross validation.

On peut reformuler le problème d'optimisation en introduisant le hinge loss.

3 Optimisation sous contrainte

Slide 12 La surface du dessin décrit la fonction à optimiser $f(x,y)$. Les seules solutions admissibles sont celles qui respectent la contrainte $g(x,y) = 0$ - sur le dessin une ellipse. Un tel problème s'appelle optimisation sous contraintes. On peut avoir des contraintes d'égalité comme ici, ou des contraintes d'inégalité du type $g(x,y) \leq 0$. On peut avoir plusieurs contraintes également. Le point optimal ne correspond bien sûr pas forcément au minimum de la fonction f.

Slide 13 Un exemple en 2D, avec la courbe rouge qui représente la contrainte d'égalité et les courbes en dégradé les lignes de niveau de la fonction à optimiser. Une condition très importante et qui n'est pas forcément très intuitive : lorsqu'on atteint le point optimal $\mathbf{x}_0$, les gradients $\nabla f(\mathbf{x}_0)$ et $\nabla g(\mathbf{x}_0)$ sont alignés. Si on est au minimum global de f, alors le gradient est nul et donc la condition est vérifiée. Sinon, une manière de le voir est de considérer que l'on se promène sur la courbe rouge de la contrainte, en diminuant petit à petit la valeur de f ; au moment où l'on ne décroît plus et qu'au prochain pas f va augmenter, la courbe g va être tangente à l'isocourbe de f. Si ce n'était pas le cas, en progressant un tout petit peu, on traverserait une autre isocourbe de f avec une valeur plus petite. On a donc une condition d'optimalité : $\nabla f(\mathbf{x}_0) = \lambda \nabla g(\mathbf{x}_0)$.

Slide 14-15 Pour résoudre ce type de problème, on utilise un outil classique : le lagrangien. Il s'agit d'introduire une fonction auxiliaire à optimiser sans contraintes, en introduisant de nouvelles variables. Pour chaque contrainte d'inégalité $c_i(\mathbf{x}) \leq 0$, on introduit une variable $\lambda_i \geq 0$, et pour chaque contrainte d'égalité $g_i(\mathbf{x}) = 0$ une variable $\mu_i \in \mathbb{R}$. Le lagrangien (ou la forme duale) associée est la fonction : $f(\mathbf{x}) + \sum_i \lambda_i c_i(\mathbf{x}) + \sum_j \lambda_j \mu_j g_j(\mathbf{x})$. On cherche à optimiser cette fonction selon les variables $\mathbf{x}$, μ_i, λ_j. Les conditions KKT sont les conditions d'optimalité.

Un très bon cours sur l'optimisation sous contraintes de Stéphane Canus à l'INSA Rouen, et la page de son cours sur l'optimisation en général avec les ressources associées.

4 SVM : optimisation

Les détails des calculs ne sont pas expliqués dans ces notes. Vous aurez à les refaire en TD. Seules les interprétations les plus importantes sont données.

Slide 17-20 Formulation duale du problème du SVM dans le cas sans et avec variable slack.

Point très important : la solution $\mathbf{w}$ est une combinaison linéaire des exemples d'apprentissage : il appartient à l'espace vectoriel engendré par les exemples. Les $\alpha_i = 0$ pour les points qui ne sont pas sur la marge, et $\alpha_i \geq 0$ pour les points sur la marge. Donc très peu de α_i non nuls. Donc $\mathbf{w}$ s'exprime avec très peu d'exemples ! Les points dont les α_i sont non nuls sont appelés vecteurs supports. C'est les seuls points qui sont réellement utilisés, les autres points de l'apprentissage pourraient ne pas être présent la solution serait la même. Intuitivement, peu importe les points qui sont dans les zones très positives/négatives, pour trouver la frontière seul les points proches d'elle sont utiles ...

Par ailleurs, le problème d'optimisation s'exprime uniquement à l'aide de produit scalaire entre exemples ! On y reviendra dans la suite.

Dans le cas de variables ressorts : α_i nul le point est bien classé au delà de la marge ; $\alpha_i = K$ le point est dans la marge ou mal classé. α_i entre 0 et K le point est sur la marge.

5 Kernel Trick

Slide 22-26 Tout au long de la résolution, de l'optimisation à la solution, les formules ne font intervenir que le produit scalaire entre exemples. L'hyperplan solution s'écrit lui aussi que en fonction du produit scalaire : $f(\mathbf{x}) = \sum_i \alpha_i y_i < \mathbf{x}^i, \mathbf{x} > + b$. Si on projette les exemples à l'aide d'une fonction ϕ afin d'accroître l'expressivité (projection polynomiale par exemple, cf TME 3 et cours 3), ce produit scalaire est remplacé par un nouveau produit scalaire : $K(\mathbf{x}^i, \mathbf{x}) = < \phi(\mathbf{x}^i), \phi(\mathbf{x}) >$. Le problème est que ce type de projection peut être très couteux à calculer. Le kernel trick consiste à se passer de la projection ϕ et de considérer directement le nouveau produit scalaire $K(x,x')$: si on peut donner une forme explicite à cette fonction - appelée noyau - ce n'est plus la peine de passer

par l'étape intermédiaire du calcul de $\phi(x)$ puis du produit scalaire entre les projettés par ϕ. Effectivement, l'algorithme du SVM n'utilise que le produit scalaire final et jamais $\phi(x)$ en tant que tel. Beaucoup de fonctions peuvent être utiliser comme noyau, en fait toute fonction $K(x,x')$ qui peut s'exprimer comme produit scalaire d'une projection ϕ. Le slide 24 donne un exemple de l'intérêt de passer par la fonction noyau : le noyau polynomiale de degré 2 à une complexité linéaire par rapport au nombre de dimensions alors que la projection elle est quadratique. L'autre avantage est de pouvoir utiliser des projections dans des bases infinis (par exemple noyau gaussien, slide 26). Intuitivement, la valeur du noyau entre deux exemples est proche de 0 lorsque les deux exemples n'ont pas de relations, et est d'autant plus grande que les deux exempels sont similaires. On peut définir des noyaux sur des objets autres que des vecteurs réels, par exemple des phrases (le noyau compte le nombre de sous-mots d'une longueur donnée communs entre les 2 phrases, voir TME), des graphes, ... C'est tout l'intérêt du kernel trick : non seulement la non-linéarité est induite par le choix d'un noyau, mais il suffit de définir une fonction de similarité entre objet qui soit semi-définie positive (voir slide 25) pour pouvoir utiser un SVM.

Le cours de Gilles Gasso et Stéphane Canus sur les noyaux

6 Take home message

Par rapport aux modèles linéaires, le SVM introduit :

— la notion de marge (et donc d'unicité de la solution) afin de construire un hyperplan robuste
— des variables ressorts - une relaxation des contraintes dures de bonnes classifications des exemples - afin de traiter le bruit des données
— la notion de noyau qui permet de traiter des problèmes non linéairement séparables et des données quelconques (en utilisant le même algorithme d'optimisation !)

L'optimisation se fait en passant généralement à l'aide du Lagrangien en posant un problème dual sans contraintes mais avec plus de variables. L'hyperplan séparateur est exprimé par une combinaison linéaire de quelques exemples - les vecteurs supports et uniquement à l'aide du produit scalaire entre une entrée et les vecteurs supports. C'est un algorithme très puissant, toujours très utilisé, surtout lorsque les données sont en nombre limitées. Le noyau et les hyperparamètres sont choisis par cross validation.

Ressources :

— La très bonne page de Stéphane Canus, avec un résumé du cours et beaucoup de ressources.
— Le cours de Alex Smola, un des papes des SVMs
— Le tutoriel de Smola et Scholkopf
— Sur l'optimisation en général, la bible de Boyd et Vandenberghe ainsi que le site associé avec slides et MOOC.
— Un cours en français de L1 éco, Dumont et al
— Un autre de C. Rau plus matheux
— Pour les curieux, le cours de S. Mallat au collège de France sur les SVMs.