Machines à vecteurs supports

- Machines à vecteurs supports -

Une machine à vecteurs supports (abrégée SVM) est une méthode d'apprentissage statistique introduite par Boser, Guyon et Vapnik au début des années 90 [1], puis largement développée par ce dernier [5] (voir [2, chapitre 12] et [4, chapitre 7] pour un descriptif détaillé). En tant qu'approche supervisée, une SVM s'appuie sur un ensemble d'apprentissage $\mathcal{D}_n = \{(\mathbf{x}_i, y_i),\ i \in [\![1, n]\!]\}$ contenant n couples d'observations explicatives $\mathbf{x}_i$ issues d'un espace vectoriel $\mathcal{X} \subset \mathbb{R}^p$ et d'observations expliquées y_i issues d'un espace discret (classification) ou continu (régression) $\mathcal{Y}$.

Il existe diverses façons de présenter les SVM, notamment à travers la théorie de la régularisation et l'analyse fonctionnelle. Nous nous restreindrons ici au cadre géométrique d'un classifieur binaire ($\mathcal{Y} = \{-1, +1\}$) mais ce discours s'étend facilement à la régression. On suppose donc que l'on dispose d'une fonction de redescription des données $\phi\colon \mathcal{X} \to \mathbb{R}^p$, associant à chaque observation $\mathbf{x}_i$ un point $\Phi(\mathbf{x}_i)$ dans un espace euclidien. Une SVM produit un estimateur affine f s'écrivant $f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x})\rangle + w_0$. On suppose donc que les données sont linéairement séparables dans l'espace de redescription. Géométriquement, le modèle f définit un hyperplan qui partitionne ainsi l'espace des données en deux parties (figure 1) : les points pour lesquels $f(\mathbf{x}) \geq 0$ et ceux pour lesquels $f(\mathbf{x}) < 0$.

Le modèle f est estimé en résolvant le problème d'optimisation (dit primal) :

$$\underset{\mathbf{w}\in\mathbb{R}^p, w_0\in\mathbb{R}}{\arg\min} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \ell(y_i, \langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + w_0).$$

Dans cette formulation, $\ell\colon \mathcal{Y} \times \mathbb{R} \to \mathbb{R}_+$ est une fonction de perte mesurant l'attache du modèle aux données, pondérée par le scalaire positif C (parfois appelé « paramètre de coût »). La perte la plus commune est la fonction charnière (*hinge*) : $\ell(y, \tilde{y}) = \max(0, 1 - y\tilde{y})$. La régularisation en norme 2 de $\mathbf{w}$, quant à elle, vise – entre autres – à prévenir le sur-apprentissage. Géométriquement, cette régularisation est liée à l'inverse de la marge $\frac{2}{\|\mathbf{w}\|}$ (figure 1) ; ainsi, une SVM détermine un hyperplan maximisant la marge entre les deux classes à séparer (puisqu'on minimise $\frac{1}{2}\|\mathbf{w}\|^2$).

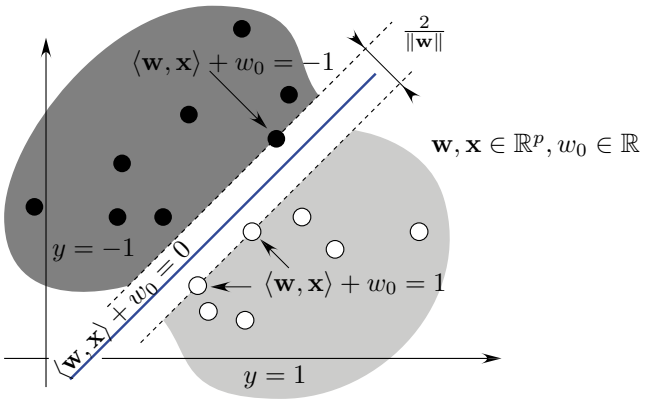

Figure 1 – Marge et hyperplan séparateur pour des classes séparables (ici, Φ est l'identité).

Il est possible de montrer qu'à l'optimalité du problème d'apprentissage, $\mathbf{w}$ s'écrit $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \Phi(\mathbf{x}_i)$, où $\boldsymbol{\alpha}$ est solution du problème d'optimisation (dit dual) :

$$\underset{\substack{\boldsymbol{\alpha}\in\mathbb{R}^n, 0\leq\alpha_i\leq C \\ \sum_{i=1}^n y_i\alpha_i = 0}}{\arg\min} \frac{1}{2}\sum_{1\leq i,j\leq n} \alpha_i\alpha_j y_i y_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)\rangle - \sum_{i=1}^{n} \alpha_i.$$

Les points d'apprentissage $\mathbf{x}_i$ pour lesquels α_i n'est pas nul sont appelés « vecteurs supports ». Eux seuls définissent en pratique la fonction f.

Pour un point inédit $\mathbf{x}$, la fonction de décision s'écrit : $\operatorname{sign}(f(\mathbf{x})) = \operatorname{sign}(\sum_{i=1}^n \alpha_i y_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle + w_0) \in \mathcal{Y}$. On remarque que l'évaluation tout comme l'apprentissage du modèle f ne requiert des données que leur produit scalaire dans l'espace de redescription et non la fonction de redescription Φ elle-même. On peut alors substituer à Φ un *noyau* $k \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (*i.e.* une fonction ayant les propriété d'un produit scalaire) : $\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle = k(\mathbf{x}, \mathbf{x}')$. Ce changement est couramment appelé l'« astuce du noyau » (*kernel trick*). Il permet de travailler dans des espaces de redescription complexes et de grande dimension (voire infinie) sans calculer les images des données par la fonction de redescription Φ, ni même connaître explicitement Φ. Quelques exemples de noyaux sont donnés ci-dessous :

- linéaire : $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$ (dans ce cas, Φ est l'identité) ;
- gaussien (Gaussian RBF) : $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2\right)$;
- polynomial : $k(\mathbf{x}, \mathbf{x}') = (\alpha + \beta \langle \mathbf{x},\ \mathbf{x}' \rangle)^\delta$ pour un $\delta > 0$;
- laplacien (Laplace RBF) : $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|\right)$ pour un $\gamma > 0$;
- tangente hyperbolique (sigmoïde) : $k(\mathbf{x}, \mathbf{x}') = \tanh\left(\alpha + \beta \langle \mathbf{x}, \mathbf{x}' \rangle\right)$ pour un couple $(\alpha, \beta) \in \mathbb{R}^2$.

Pour un noyau linéaire, on parle d'une SVM linéaire (f est linéaire par rapport à $\mathbf{x}$) ; sinon d'une SVM non-linéaire (f est non-linéaire par rapport à $\mathbf{x}$). Dans ce dernier cas, la frontière entre les classes dans l'espace des données $\mathcal{X}$ n'est plus un hyperplan mais une surface dont la forme dépend du noyau choisi.

Remarque

L'approche présentée jusqu'ici est appelée C-classification dans `scikit-learn` et est accessible grâce à l'objet `sklearn.svm.SVC`. Un point de vue différent, nommé ν-classification (`sklearn.svm.NuSVC`) considère le problème d'optimisation dual suivant :

$$\underset{\substack{\boldsymbol{\alpha} \in \mathbb{R}^n, 0 \leq \alpha_i \leq 1 \\ \sum_{i=1}^n y_i \alpha_i = 0}}{\arg\min} \frac{1}{2} \sum_{1 \leq i,j \leq n} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j), \quad \text{contraint par } \sum_{i=1}^n \alpha_i \geq \nu.$$

Ici, $\nu \in [0, 1]$ est un paramètre approchant le pourcentage de vecteurs supports parmi les données d'apprentissage. Ces deux approches sont équivalentes.

Extensions au cas multi-classe

Il existe plusieurs manières d'étendre un classifieur binaire à plusieurs classes, parmi lesquelles :

un contre un : construire des classifieurs binaires sur toutes les pairs de classes possibles. La décision finale est donnée par la classe prédite le plus de fois.

un contre tous : construire des classifieurs binaires pour chaque classe (toutes les autres étant regroupées sous une même classe étiquetée -1). La décision finale est donnée par la classe prédite avec la plus grande décision $f(\mathbf{x})$.

Crammer et Singer : modifier la fonction de perte de sorte à considérer plusieurs classes [3].

Questions

1. À partir de la documentation : http://scikit-learn.org/modules/svm.html, écrire un script estimant une SVM sur les classes 1 et 2 du jeu de donnée iris. Vous n'utiliserez que les deux premières variables et un noyau linéaire. Afficher le score moyen et la frontière de décision.
2. En laissant la moitié des données de côté, évaluer la performance en généralisation du modèle. Pour ce faire, vous déterminerez C par validation croisée. Comparer le résultat avec une SVM polynomiale.
3. En vous inspirant de l'exemple http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html, afficher une carte de performance pour le noyau gaussien (on n'utilisera que 10 valeurs pour C et γ : `C_range, gamma_range = np.logspace(-2, 10, 10), np.logspace(-9, 3, 10)`).

4. Exécuter le script `svm_gui.py`. Il permet dévaluer en temps réel l'impact du choix du noyau et du paramètre de régularisation C. Effectuer quelques tests et commenter les observations.
5. Générer un jeu de données très déséquilibré (beaucoup plus de points dans une classe que dans l'autre). Avec un noyau linéaire, diminuer progressivement la valeur de C. Commenter. Ce phénomène peut être corrigé en pondérant d'avantage l'attache aux données sur la classe la moins présente (paramètre `class_weight` de `SVC`).
6. Dans la suite, on s'intéresse à un problème de classification de visages. La base de données est disponible à l'adresse suivante : http://perso.telecom-paristech.fr/~gramfort/lfw.zip. En modifiant le script `svm_lfw.py`, montrer l'influence du paramètre de régularisation. On pourra par exemple afficher l'erreur de prédiction en fonction de C sur une échelle logarithmique entre `1e-6` et `1e3`.
7. Le script que vous utilisez centre et normalise les données. Décrire comment et expliquer l'intérêt de cette opération.
8. En ajoutant des variables de nuisances (augmentant ainsi le nombre de variables à nombre de points d'apprentissage fixé), montrer que la performance chute.
9. Quel est l'effet du choix d'un noyau non-linéaire gaussien sur la prédiction ?
10. L'exemple http://scikit-learn.org/stable/auto_examples/svm/plot_separating_hyperplane.html#example-svm-plot-separating-hyperplane-py explique comment accéder au paramètres estimés lors de l'apprentissage : vecteur de coefficients $\mathbf{w}$ dans l'attribut `coef_`, w_0 attribut `intercept`, liste des vecteurs de supports, coefficients du problème dual). À partir de cet exemple, écrire un script qui calcule la valeur des fonctionnelles primale et duale. Vérifier que les valeurs sont proches (attention, les étiquettes doivent être -1 ou 1). Comment varie la différence entre les deux valeurs quand on fait varier la tolérance sur l'optimisation (paramètre `tol` de `SVC`) ?

- Liens utiles -

⋆⋆⋆ http://scikit-learn.org/stable/modules/svm.html

⋆⋆ http://en.wikipedia.org/wiki/Support_vector_machine

⋆⋆ http://fr.wikipedia.org/wiki/Machine_%C3%A0_vecteurs_de_support

Références

[1] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. In *Conference on Learning Theory*, 1992. 1

[2] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer Series in Statistics. Springer, New York, second edition, 2009. http://www-stat.stanford.edu/~tibs/ElemStatLearn/. 1

[3] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. 13 :415–425, 2002. 2

[4] B. Schölkopf and A. J. Smola. *Learning with kernels : Support vector machines, regularization, optimization, and beyond*. MIT press, 2002. 1

[5] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer New York, 1995. 1