Université de Montpellier **HAX907X**
2023/2024 **B. Charlier, J. Salmon, N. Verzelen (et A. Gramfort)**

TP n° 3 : Support Vector Machine (SVM)

Liens utiles pour ce TP :

⋆⋆⋆ 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

- Introduction et fondements mathématiques -

Les SVM ont été introduites par Vapnik [3], et sont abordées au chapitre 12 du livre [1]. Des détails plus mathématiques peuvent être trouvés dans le chapitre 7 du livre [2]. La popularité des méthodes SVM, pour la classification binaire en particulier, provient du fait qu'elles reposent sur l'application d'algorithmes de recherche de règles de décision linéaires : on parle d'hyperplans (affine) séparateurs. Toutefois, cette recherche s'effectue dans un espace de caractères (*feature space*, en anglais) de très grande dimension qui est l'image de l'espace d'entrée original par une transformation Φ non linéaire.

Le but de ce TP est de mettre en pratique ce type de techniques de classification sur données réelles et simulées au moyen du package `scikit-learn` (lequel met en œuvre la librairie en `C` libsvm) et d'apprendre à contrôler les paramètres garantissant leur flexibilité (hyper-paramètres, noyau).

Définitions et notations

On rappelle que dans la cadre de la classification binaire supervisée on utilise les notations :

- $\mathcal{Y}$ l'ensemble des étiquettes (ou *labels* en anglais), communément $\mathcal{Y} = \{-1, 1\}$ dans le cas de la classification binaire,
- $\mathbf{x} = (x_1, \ldots, x_p) \in \mathcal{X} \subset \mathbb{R}^p$ est une observation (ou un exemple),
- $\mathcal{D}_n = \{(\mathbf{x}_i, y_i), i = 1, \ldots n\}$ un ensemble d'apprentissage contenant n exemples et leurs étiquettes,
- Il existe un modèle probabiliste qui gouverne la génération de nos observations selon des variables aléatoires X et Y : $\forall i \in \{1, \ldots, n\}, (\mathbf{x}_i, y_i) \overset{i.i.d}{\sim} (X, Y)$.
- On cherche à construire à partir de l'ensemble d'apprentissage $\mathcal{D}_n$ une fonction $\hat{f} : \mathcal{X} \mapsto \{-1, 1\}$ qui pour un point inconnu $\mathbf{x}$ (*i.e.,* qui n'est pas présent dans l'ensemble d'apprentissage) prédit son étiquette : $\hat{f}(\mathbf{x})$. La règle que l'on considère ici est dite linéaire, dans le sens où l'on sépare l'espace par un hyperplan (affine) : selon la position par rapport à celui-ci, on prédit alors $+1$ ou -1.

SVM et noyaux pour la classification binaire

Les techniques SVM (non linéaires) font appel à une fonction implicite Φ transformant l'espace d'entrée $\mathcal{X} \subset \mathbb{R}^p$ en un espace hilbertien $(\mathcal{H}, \langle \cdot, \cdot \rangle)$ de plus grande dimension. L'apprentissage s'effectue alors à partir du modèle $(\Phi(X), Y)$ dans l'espace $\mathcal{H}$, de dimension plus grande certes, mais dans lequel on espère que les données soient "davantage linéairement séparables". Du point de vue pratique, il convient de noter que le calcul des projections $\Phi(X)$ n'est pas utilisé dans la méthode, seuls les produits scalaires $\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$, $(\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2$, sont requis. Or, ceux-ci sont donnés par un noyau K, via la relation *kernel trick* (en anglais : astuce du noyau) :

$$K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle.$$

La méthode requiert donc de sélectionner un noyau (ainsi que d'autre paramètres). Parmi les choix possibles, on compte en particulier les noyaux suivants

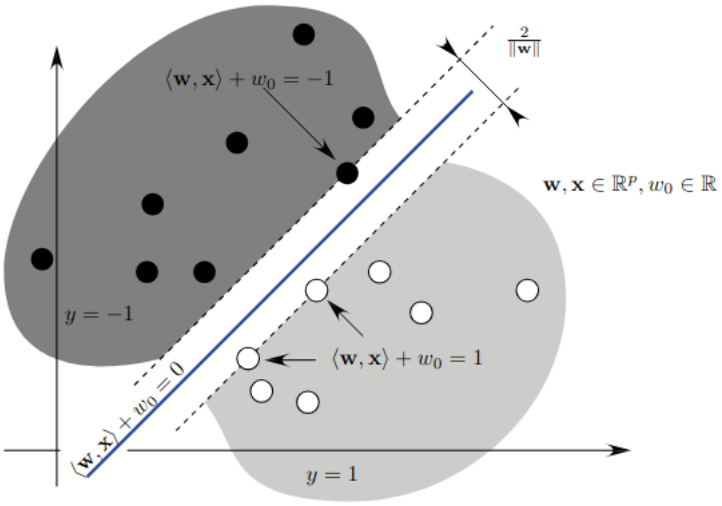

FIGURE 1 – Marge et hyperplan séparateur dans le cadre de classes séparables (cas d'un noyau linéaire)

- Noyau linéaire : $K(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$ (correspondant aux SVM linéaires)
- Noyau Gaussien radial (Gaussian RBF) $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2\right)$
- Noyaux polynomiaux $K(\mathbf{x}, \mathbf{x}') = (\alpha + \beta \langle \mathbf{x},\ \mathbf{x}' \rangle)^{\delta}$ pour un $\delta > 0$
- Noyau radial de Laplace (Laplace RBF) $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|\right)$ pour un $\gamma > 0$
- Noyau tangente hyperbolique (sigmoïde) $K(\mathbf{x}, \mathbf{x}') = \tanh\left(\alpha + \beta \langle \mathbf{x},\ \mathbf{x}' \rangle\right)$ pour un couple $(\alpha, \beta) \in \mathbb{R}^2$

Un classifieur SVM est de la forme

$$\hat{f}_{\mathbf{w}, w_0}(\mathbf{x}) = \text{sign}\left(\langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + w_0\right), \tag{1}$$

où $\mathbf{w} \in \mathcal{H}$ et $w_0 \in \mathbb{R}$ sont des paramètres ajustés lors de la phase d'apprentissage à partir d'un échantillon d'exemples i.i.d., $\mathcal{D}_n = \{(\mathbf{x}_i, y_i), i = 1, \dots n\}$. À noter : dans le cas où la fonction Φ est l'identité sur $\mathbb{R}^p$, on trouve simplement que la règle de décision est

$$\hat{f}_{\mathbf{w}, w_0}(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{p} w_i x_i + w_0\right).$$

La frontière associée à la règle de décision (1) est l'ensemble $\{\mathbf{x} : \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + w_0 = 0\}$. Elle correspond à un hyperplan dans l'espace $\mathcal{H}$, mais est beaucoup plus complexe dans $\mathcal{X}$ (selon la forme du noyau choisi).

Dans $\mathcal{H}$, l'hyperplan est obtenu en maximisant la marge séparant les deux classes, ce qui revient à résoudre un problème d'optimisation sous contraintes linéaires :

$$\begin{cases} (\mathbf{w}^*, w_0^*, \xi^* \in \mathbb{R}^n) \in \underset{\mathbf{w} \in \mathcal{H}, w_0 \in \mathbb{R}, \xi \in \mathbb{R}^n}{\arg\min} \left(\dfrac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i \right) & \\ \text{s.c} \quad \xi_i \geq 0, & \forall i \in \{1, \cdots, n\}, \\ \quad\ \ y_i \left(\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + w_0 \right) \geq 1 - \xi_i, & \forall i \in \{1, \cdots, n\}. \end{cases} \tag{2}$$

On peut montrer que la solution $\mathbf{w}$ peut s'exprimer de la façon suivante :

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \Phi(\mathbf{x}_i).$$

Les indices i pour lesquels $\alpha_i^* \neq 0$ sont ceux pour lesquels l'égalité est réalisée dans la contrainte, les points $\mathbf{x}_i$ correspondants sont appelés *support vectors* (vecteurs supports en français) de la décision. Les coefficients α_i^* désignent les solutions du problème dual qui est un programme quadratique (QP) sous contraintes affines :

$$\begin{cases} \alpha^* \in \underset{\alpha \in \mathbb{R}^n}{\arg\max} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{1 \leq i,\, j \leq n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right) \\ \text{s.c} \quad 0 \leq \alpha_i \leq C, \quad \forall i \in \{1, \cdots, n\}, \\ \quad \sum_{i=1}^{n} \alpha_i y_i = 0. \end{cases} \tag{3}$$

Le paramètre C contrôle la complexité du classifieur dans la mesure où il détermine le coût d'une mauvaise classification : plus C est grand, plus la règle obtenue est complexe (le nombre de points pour lesquels on veut minimiser l'erreur de classification croît). Cette approche est appelée C-classification et s'utilise avec l'objet `sklearn.svm.SVC` dans le module `scikit-learn`.

Une autre façon de contrôler la complexité (*i.e.,* le nombre de vecteurs supports), appelée ν-classification, revient à considérer, à la place du problème dual décrit ci-dessus, le problème suivant :

$$\begin{cases} \alpha^* \in \underset{\alpha \in \mathbb{R}^n}{\arg\min} \left(\frac{1}{2} \sum_{1 \leq i,\, j \leq n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right) \\ \text{s.c} \quad 0 \leq \alpha_i \leq 1, \quad \forall i \in \{1, \cdots, n\}, \\ \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \\ \quad \sum_{i=1}^{n} \alpha_i \geq \nu. \end{cases} \tag{4}$$

où $\nu \in [0, 1]$ est un paramètre approchant le pourcentage de vecteurs supports parmi les données d'apprentissage. Cette approche s'utilise avec l'objet `sklearn.svm.NuSVC` dans le module `scikit-learn`.

Extensions au cas multi-classe

Dans le cas où la variable de sortie Y compte plus de deux modalités, il existe plusieurs façon d'étendre directement les méthodes du cas binaire.

"Un contre un". Dans le cas où l'on cherche à prédire un label pouvant prendre $K \geq 3$ modalités, on peut considérer toutes les paires de labels (k, l) possibles, $1 \leq k < l \leq K$ (il y en a $K(K-1)/2$) et ajuster un classifieur $C_{k,l}(X)$ pour chacune d'entre elles. La prédiction correspond alors au label qui a gagné le plus de "duels".

"Un contre tous". Pour chaque modalité k, on apprend un classifieur permettant de discriminer entre les populations $Y = k$ et $Y \neq k$. A partir des estimations des probabilités a posteriori, on affecte le label estimé le plus probable.

- Mise en oeuvre -

On utilisera l'objet `sklearn.svm.SVC` :

```
from sklearn.svm import SVC
```

Le fichier `svm_script.py` vous fournit un exemple complet de classification utilisant la fonction `SVC`. Utilisez cet exemple pour vous familiariser avec la syntaxe puis reproduisez une expérience similaire avec les données `iris` comme suggéré ci-dessous.

1) En vous basant sur la documentation à l'adresse suivante :

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

écrivez un code qui va classifier la classe 1 contre la classe 2 du dataset `iris` en utilisant les deux premières variables et un noyau linéaire. En laissant la moitié des données de côté, évaluez la performance en généralisation du modèle. Le dataset `iris` s'obtient avec les lignes suivantes :

```
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data
y = iris.target
X = X[y != 0, :2]
y = y[y != 0]
```

2) Comparez le résultat avec un SVM basé sur noyau polynomial.

SVM GUI (optionnel)

— Lancez le script `svm_gui.py` disponible à l'adresse : http://scikit-learn.org/stable/auto_examples/applications/svm_gui.html Cette application permet en temps réel d'évaluer l'impact du choix du noyau et du paramètre de régularisation C.

— Générez un jeu de données très déséquilibré avec beaucoup plus de points dans une classe que dans l'autre (au moins 90% vs 10%).

3) **Question bonus :** À l'aide d'un noyau linéaire et en diminuant le paramètre C qu'observez vous ?

Ce phénomène peut être corrigé en pratique en pondérant d'avantage les erreurs sur la classe la moins présente (paramètre `class_weight` de SVC) ou par une technique de re-calibration (utilisée avec `SVC(..., probability=True)`). On pourra voir aussi https://scikit-learn.org/stable/modules/svm.html#unbalanced-problems

Classification de visages

L'exemple suivant est un problème de classification de visages. La base de données à utiliser est disponible à l'adresse suivante : http://vis-www.cs.umass.edu/lfw/lfw-funneled.tgz.

En modifiant l'exemple de code donné dans la dernière partie du fichier `svm_script.py` :

4) Montrez l'influence du paramètre de régularisation. On pourra par exemple afficher l'erreur de prédiction en fonction de C sur une échelle logarithmique entre `1e5` et `1e-5`.

5) En ajoutant des variables de nuisances, augmentant ainsi le nombre de variables à nombre de points d'apprentissage fixé, montrez que la performance chute.

6) Améliorez la prédiction à l'aide d'une réduction de dimension basée sur l'objet `sklearn.decomposition.PCA(svd_solver='randomized')`.

Références

[1] T. J. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, New York, second edition, 2009. 1

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

[3] V. N. Vapnik. *Statistical learning theory*. Wiley, 1998. 1