TP - Machines à vecteurs supports.

L'objet de cette séance de TP est de vous familiariser avec les machines à vecteurs supports. Il s'agit d'une famille d'algorithmes pouvant servir à diverses tâches d'apprentissage (classification, régression...). Nous nous intéresserons ici uniquement aux problème de classification à deux classes.

libSVM

Nous allons nous servir de la librairie libSVM disponible à l'adresse `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`. Le site contient une archive contenant le code nécessaire à l'entraînement des SVMs, la documentation sur le fonctionnement de la librairie et un guide pratique pour l'utilisation des machines à vecteurs supports.

Avant de commencer le TP, téléchargez et compilez libSVM en utilisant la commande `make` dans le répertoire MATLAB depuis Octave.

1 SVM linéaires

1.1 Cas séparable

Dans cette première partie, nous nous intéressons à une SVM linéaire à « marges rigides ». Cette première version de SVM fournit un classifieur (une fonction linéaire h) pour deux classes linéairement séparables, i.e. telles qu'il existe un hyperplan séparateur (d'équation $h(x) = 0$).

Formellement, on pose $(\mathbf{x_i}, y_i)_{i=1}^n$ un ensemble de n points d'entraînement, avec $\mathbf{x_i} \in \mathbb{R}^d$ et $y_i \in \{-1, 1\}$. On résoud ensuite :

$$\begin{cases} \text{minimiser} & \frac{1}{2}\ ||\mathbf{w}||^2 \\ \text{sous contraintes} & \forall i, y_i(<\mathbf{w}, \mathbf{x_i}> + b) \geq 1 \end{cases}$$

On obtient alors une suite de coefficients réels $(\alpha_i)_{i=1}^n$ et

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i} \tag{1}$$

$$h(\mathbf{x}) = <\mathbf{w}, \mathbf{x}> + b. \tag{2}$$

Les vecteurs $\mathbf{x_i}$ tels que $\alpha_i \neq 0$ sont appelés **vecteurs supports**.

Exercice 1

On se place dans le cas où $d = 2$.

1. Un ensemble de points d'entraînement se présente pour libSVM comme la données de deux matrices X et *label*. X est une matrice dont chaque ligne représente un point d'entraînement (n lignes $\times$ d colonnes). *label* est une vecteur colonne contenant la classe de chaque exemple.

On propose :

$$X = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix} \qquad label = \begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \end{pmatrix}$$

2. Tracez les éléments de l'ensemble d'entraînement, en distinguant les classes.
3. Entraînez une SVM linéaire sur cet ensemble.
   Aide : l'instruction `model = svmtrain(label, X, 'options')` retourne le résultat de l'entraînement d'une SVM suivant les options spécifiées en argument. (Tapez `svmtrain` dans Octave ou consultez la documentation pour plus de détails).
   Par exemple, ici : `model = svmtrain(label, X, '-s 0 -t 0 -c 1e6')`.
4. Que constatez-vous ? Quelle explication simple pouvez-vous proposer ?
5. Écrivez un script permettant de générer un ensemble de n points d'entraînement linéairement séparables. On commence en définissant la classe 1 comme les $(a, b) \in \mathbb{R}^2$ tels que $a \geq 0$.
6. Tracez les éléments de ce nouvel ensemble d'entraînement, en distinguant les classes.
7. Entraînez une SVM avec les mêmes options.
8. Tracez sur un même graphe l'ensemble d'entraînement, les vecteurs supports et la droite de séparation.
   Indications :
   - Après appel de `svmtrain`, la structure `model` contient l'ensemble des paramètres de la SVM : vecteurs supports (`model.SVs`), coefficients α_i (`model.sv_coef`) et b (égal à `- model.rho`).
   - Les matrices retournées sont sous forme `sparse`, utilisez `full` pour obtenir la matrice dense correspondante.
9. Proposez un nouvel ensemble d'entraînement linéairement séparable et testez de la même manière une SVM à marges rigides.

1.2 Cas non séparable

Comme vous avez pu le constater, un inconvénient majeur des machines linéaires à marge rigides est leur incapacité à traiter le cas où l'ensemble de points d'entraînement n'est pas linéairement séparable.

Pour pallier à ce défaut, les machines à marges « souples » ont étés introduites. L'idée employée est de d'autoriser la mauvaise classification de quelques exemples d'entraînement.

Nous introduisons pour cela les « variables ressorts » $\zeta_i > 0$ et le problème d'optimisation devient :

$$\begin{cases} \text{minimiser} & \frac{1}{2} \, ||\mathbf{w}||^2 + C \sum_{i=1}^{n} \zeta_i \\ \text{sous contraintes} & \forall i, y_i(< \mathbf{w}, \mathbf{x_i} > + b) \geq 1 - \zeta_i \\ & \forall i, \zeta_i \geq 0 \end{cases}$$

La constante $C > 0$ (coût d'erreur) est un paramètre contrôlant la pénalisation des erreurs de classification. La résolution aboutit à une fonction de décision h similaire au cas non séparable (éq. **??**)

Exercice 2

La valeur de C est spécifiée par l'option '-c val' dans libSVM. [1]

1. Essayer l'exemple du XOR avec une SVM à marges souples ($C = 1$ par ex.). Que constatez-vous ?
2. Modifiez le script de générations de points de façon à ajouter du bruit à l'exemple précédent : avec probabilité p, le point $x_i = (x_{i,1}, x_{i,2})$ a le label -1 alors que $x_{i,1} > 0$.
3. Écrivez un script qui, pour un couple (n, p, C) donné :

1. En toute rigueur, libSVM n'implémente pas les SVM à marges rigides. Nous les avons simulé dans l'exercice 1 en prenant une valeur de C extrêmement élevée.

– Génère un ensemble de n points d'entraînement bruité comme précédemment.
– Entraîne une SVM sur cet ensemble de points avec la valeur spécifiée de C.
– Affiche sur un même graphe, les points d'entraînement, les vecteurs supports, et la droite de séparation.
Testez différentes valeurs de C et p.

4. Fixez C et tracez la précision du classifieur en fonction de p.
   Indication : `[predicted_label, accuracy]=svmpredict(label, X, model)` calcule à partir de `model` les labels prédits des points de X. Le passage en paramètre des classes exactes des points de X dans `label` permet de calculer le taux de précision de la SVM.
5. Écrivez une fonction `accuracy = testC(n, m, p, valC)` :
   – Génère, toujours de la même façon, un ensemble de $n + m$ points.
   – Découpe cet ensemble en un ensemble de n points d'apprentissage et un ensemble de m points de test.
   – Entraîne une SVM linéaire pour chaque $c \in valC$ sur l'ensemble d'apprentissage.
   – Calcule et retourne la précision de chacune de ces machines sur l'ensemble de test.
   Indications :
   – `num2str` convertit un nombre en chaîne de caractères.
   – `cstrcat` permet de concaténer deux chaînes de caractères.
6. Tracez la précision en fonction de C en fixant les autres paramètres.
7. Tracez la précision en fonction de C dans le cas où les classes sont définies selon une fonction ϕ non linéaire. Par exemple, $x_i = (x_{i,1}, x_{i,2})$ est dans la classe 1 ssi $\phi(x_{i,1}, x_{i,2}) > 0$ avec $\phi(a, b) = a + b^3$.

2 Noyaux

L'≪ astuce du noyau ≫ est une méthode permettant d'étendre l'espace $\mathcal{H}$ d'hypothèses. L'idée est de plonger les données d'entraînements via une fonction ϕ dans un espace de grande dimension (*feature space*).
Formellement, il s'agit de résoudre le problème suivant :

$$\begin{cases} \text{minimiser} & \frac{1}{2}\ ||\mathbf{w}||^2 + C\sum_{i=1}^{n} \zeta_i \\ \text{sous contraintes} & \forall i, y_i(< \mathbf{w}, \phi(\mathbf{x_i}) > +b) \geq 1 - \zeta_i \\ & \forall i, \zeta_i \geq 0 \end{cases}$$

La fonction de décision h s'écrit maintenant

$$h(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x_i}, \mathbf{x}) + b. \tag{3}$$

Exercice 3

Dans cette exercice, nous travaillerons avec des noyaux gaussiens : $K(\mathbf{x}, \mathbf{y}) = exp(-\gamma\|\mathbf{x} - \mathbf{y}\|^2)$, avec $\gamma > 0$ paramètre.

1. Reprenez encore une fois l'exemple du XOR en entraînant une SVM avec noyau gaussien ($C = 1, \gamma = 1$). Que constate-t-on ?
   On reprend la fonction ϕ précédente.
2. Écrivez une fonction `plotzone(model)` traçant sur en couleurs distinctes, l'ensemble des points du plan tels que $h(x_1, x_2) > 0$ et $h(x_1, x_2) < 0$.
3. En fixant C à 1, prenez des valeurs de plus en plus grandes pour γ et tracez l'ensemble d'apprentissage sur un graphe, et les frontières de décisions sur un autre graphe.
   Que remarquez-vous ?

Exercice 4

Une SVM à noyau gaussien fait intervenir deux paramètres, C et γ, influençant les performances de classification. Nous ne pouvons pas déterminer *a priori* pour chaque tâche de classification, les paramètres les plus efficaces.

Pour déterminer les paramètres, il est souvent fait appel à la méthode de validation croisée à n plis :

1. l'ensemble d'apprentissage est divisé en n sous-ensembles de même taille.
2. pour un jeu de paramètres donné, chacun de ces n sous-ensembles est testé avec une machine entraînée sur les $n-1$ autres sous-ensembles.
3. On obtient ainsi, pour chaque couple (C, γ) une valeur de précision correspondant au pourcentage d'exemples bien classés. On conserve le couple de précision maximale.

L'option `'-v n'` de la fonction `svmtrain` réalise la validation croisée à n plis et retourne l'indice de précision correspondant au couple (C, γ) courant.

1. Réalisez une fonction `[vc, c, gamma]=gridsearch(label, X, valC, valGamma, n)` qui calcule, pour chaque couple (C, γ) passé en paramètres, la précision associée et qui retourne le score maximal et le couple correspondant.
2. Effectuez la recherche du meilleur couple pour ϕ.

   Conseil : Prenez des puissances de 2 pour C et γ. Par exemple, $C = 2^{-5}, \ldots, 2^{10}$ et $\gamma = 2^{-}15, \ldots, 2^{3}$, quitte à affiner votre recherche dans un second temps.
3. Tracez la précision en fonction de C et γ (carte de chaleur).
4. Tracez la solution optimale que vous avez obtenue.