SVM et validation croisée

Guillaume Wisniewski
guillaume.wisniewski@limsi.fr

mars 2016

L'objectif de ce TP est de mettre en œuvre les SVM sur une tâche de classification multiclasse en optimisant les différents hyper-paramètres mis en jeux dans cette méthode d'apprentissage.

1 Approche directe

Nous allons considérer, tout au long de ce TP, une tâche de reconnaissance de chiffres manuscrits et utiliserons la base de données MNIST [1]. Les données sont téléchargeables sur le site du cours et peuvent être « chargées » à l'aide du code suivant :

```
import pickle
import gzip
import numpy

# Load the dataset
with gzip.open("mnist.pkl.gz", "rb") as ifile:
    train, valid, test = pickle.load(ifile, encoding="latin-1")
```

Le corpus MNIST définit un ensemble de test, d'apprentissage et de validation « officiels ». Nous utiliserons, dans un premier temps, cette partition des données.

Nous souhaitons apprendre un SVM utilisant un noyau gaussien ou polynomial et considérant comme uniques caractéristiques la valeur des pixels. On pourra utiliser la classe `sklearn.svm.SVC` de la bibliothèque `scikit-learn`.

1. `http://yann.lecun.com/exdb/mnist/`

1. Quels sont les paramètres et les hyper-paramètres de la méthode proposée ? Pourquoi distingue-t-on ces deux types de paramètres.

Pour choisir la valeur des hyper-paramètres on utilise, en général, une méthode « force brute » qui consiste simplement à tester toutes les combinaisons de paramètres possibles et à garder celle obtenant le plus grand score sur un ensemble de validation.

2. Pourquoi distingue-t-on ensemble de validation et ensemble de test ?
3. En considérant un SVM linéaire, tracer la performance obtenue en test et en validation en fonction de C [2]. Commentez.
4. Relancer en considérant, cette fois, tous les hyper-paramètres. Commentez.

2 Validation croisée

L'utilisation d'un ensemble de validation et de test distincts n'est possible que lorsque l'on dispose de beaucoup de données.

5. Pourquoi ?

Lorsque seul un petit nombre de données est disponible, on utilise une méthode dite de *validation croisée* pour construire un ensemble de validation (on supposera, par soucis de simplification que l'on dispose quand même d'un ensemble de test).

Dans la validation croisée à k *fold*, on divise l'échantillon original en k échantillons, puis on sélectionne un des k échantillons comme ensemble de validation et les $(k-1)$ autres échantillons constitueront l'ensemble d'apprentissage. On calcule alors l'erreur obtenue par une combinaison d'hyper-paramètres sur l'ensemble de validation. Puis on répète l'opération en sélectionnant un autre échantillon de validation parmi les $(k-1)$ échantillons qui n'ont pas encore été utilisés pour la validation du modèle. L'opération se répète ainsi k fois pour qu'en fin de compte chaque sous-échantillon ait été utilisé exactement une fois comme ensemble de validation. La moyenne des k erreurs est enfin calculée pour estimer l'erreur de prédiction et choisir la valeur optimale des hyper-paramètres.

6. Implémentez cette méthode en utilisant la classe `sklearn.grid_search.GridSearchCV`.
7. Comparer les performances à celles obtenues dans la section précédente. Commentez

2. On considère généralement $C \in \{10^i, i \in [\![-4, 4]\!]\}$.