Classification linéaire

Guillaume Wisniewski et Alexandre Allauzen
wisniews@limsi.fr

Janvier 2016

1 Culture générale : *catastrophic cancellation*

Ce cours exercice a pour objectif de mettre en évidence l'impact de l'utilisation d'une représentation approchées des réels lors de calculs numériques.

1. On considère un ensemble de 10^6 points distribués uniformément dans $[10^9, 10^9 + 1]$:

   ```
   import numpy as np

   n = 10 ** 6
   data = 10 ** 9 + np.random.uniform(0, 1, n)
   ```

   Quelle est la moyenne de cet ensemble de points ? sa variance ?

2. Écrivez un programme python permettant de calculer la moyenne et la variance. Qu'observez-vous ?

3. La méthode de Welford permet de calculer la moyenne et la variance en évitant les problèmes de stabilité numérique. Elle repose sur la définition récursive[1] de la moyenne et de la variance :

$$m_k = m_{k-1} + \frac{x_k - m_{k-1}}{k} \tag{1}$$

$$s_k = s_{k-1} + (x_k - m_{k-1}) \times (x_k - m_k) \tag{2}$$

   avec $m_1 = x_1$ et $s_1 = 0$. Implémentez cette méthode. Qu'en concluez-vous ?

Pour mémoire : la moyenne et la variance d'un ensemble de points $x = (x_i)_{i=1}^n$ sont définies (respectivement) comme :

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

$$s^2 = \frac{1}{n-1} \left(\sum_{i=1}^{n} x_i^2 - \bar{x}^2 \right)$$

2 Code fourni

Dans ce TP, toutes les données sont représentées par des couples (étiquette, observation) où l'observation est un vecteur de réels. Des méthodes pour lire les données des différents exercices sont disponibles sur le site du cours.

[1] Pour information : cette définition permet également de calculer la moyenne et la variance *on line*, au fur et à mesure que les points arrivent.

3 Classification binaire

3.1 Cas simple

L'objectif de cette partie est d'implémenter l'algorithme du perceptron pour résoudre un problème de classification binaire. Il s'agit de prédire, à partir de ses votes sur des sujets « sensibles » si un homme politique est démocrate ou républicain.

Le corpus est constitué de 435 exemples. Chaque exemple décrit un député américain et est décrit par l'appartenance politique de celui-ci (l'*étiquette*) et de 15 de ses votes (les *observations*). Une description détaillée des données est disponible à l'url `http://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records`. 100 exemples de la base seront utilisés comme ensemble de test, les autres comme ensemble d'apprentissage. Il est important que cette séparation soit fait de manière aléatoire (c.-à-d. sans prendre 100 exemples consécutifs), par exemple en mélangeant les données avant de les séparer :

```
import random

# corpus de données sous forme de liste
data = ...
random.shuffle(data)
```

La méthode fournie sur le site du cours permet de lire les données (téléchargeables à l'url précédente) et représente celles-ci sous la forme d'une paire (étiquette, observation). L'étiquette vaut soit 1 (démocrate), soit -1 (républicain). Les observations sont représentées par une instance de la classe `array` de la bibliothèque `numpy`. La méthode de lecture ajoute à l'observation un attribut qui vaut toujours 1.

4. Donnez deux exemples de tâches (autres que celles présentées dans le sujet) pouvant se formaliser comme des problèmes de classification binaires. Précisez à chaque fois la signification des étiquettes et donnez un exemple de caractéristiques pouvant être utilisées.

5. Pourquoi ajoute-t-on un attribut constant ?

Dans le cas binaire, un perceptron se représente par un unique vecteur de paramètres dont la taille est égale au nombre d'attributs des observations.

6. Écrivez une méthode `classify` qui prend en paramètre une observation et un vecteur de paramètres et qui renvoie l'étiquette associée à l'observation.

7. Écrivez une méthode `test` qui prend en paramètre un corpus et un vecteur de paramètres et qui renvoie le pourcentage d'erreurs sur le corpus.

8. Quel taux d'erreur obtenez-vous sur le corpus de test avec le vecteur de paramètres : `[25, -12, 67, -104, -43, 46, -18, -10, 45, -33, 54, -39, 43, -19, 5, -2, 55]`

On souhaite maintenant développer l'algorithme d'apprentissage. On suppose, pour le moment, que le *learning rate* η est constant et que le nombre de passes sur le corpus d'apprentissage est un paramètre choisi par l'utilisateur. En cas d'erreur, la règle de mise-à-jour du perceptron est alors :

$$\mathbf{w} \leftarrow \mathbf{w} + y \cdot \mathbf{x} \tag{3}$$

où y est l'étiquette réelle de l'exemple et $\mathbf{x}$ l'observation correspondante.

9. Implémentez la méthode `learn` qui permet d'apprendre le vecteur de paramètre d'un perceptron binaire.

10. Déterminez le taux d'erreur en test et en apprentissage pour différents nombres d'itérations. Que peut-on en conclure ?

3.2 Critère d'arrêt

Le choix d'un bon critère d'arrêt est capital pour obtenir de bonnes performances. Plusieurs critères sont envisageables :

- itération jusqu'à ce qu'il n'y ait plus d'erreurs sur le corpus d'apprentissage ;
- nombre fixe d'itérations ;
- itération jusqu'à ce que l'erreur sur un corpus de *développement* (distinct du corpus d'apprentissage et du corpus de test) cesse de descendre

11. Est-on sûr d'atteindre une erreur nulle sur le corpus d'apprentissage ?
12. Pourquoi considérer un corpus de développement distinct du corpus d'apprentissage et du corpus de test ?
13. Comparer ces trois méthodes sur les données présentées dans la partie précédente.

On considère maintenant un corpus de spam disponible à l'url `http://archive.ics.uci.edu/ml/datasets/Spambase`. Ce corpus comprend 4 601 mails décrits par 57 attributs et qui sont classés soit comme spam (étiquette -1) soit comment non-spam (étiquette 1). Les méthodes de lecture sont disponibles sur le site du cours ; comme pour le corpus précédent, ces méthodes ajoutent un attribut constant. Le corpus d'apprentissage est constitué des 3 500 premiers exemples, le corpus de test des exemples restant.

14. Reprenez la question précédente avec ce corpus.

3.3 Impact du *learning rate*

L'algorithme du perceptron est généralement présenté en incluant un paramètre supplémentaire qui permet de pondérer le poids des mises à jour. En cas d'erreur, la règle de mise-à-jour est alors :

$$\theta \leftarrow \theta + \alpha \cdot y \cdot \mathbf{x}$$

où α est une constante comprise entre 0 et 1.

15. Étudiez, sur les deux corpus du sujet, la vitesse de convergence (c.-à-d. le taux d'erreur en fonction du nombre d'itérations) pour différentes valeurs de α. On comparera en particulier les taux d'erreur « à convergence », après un grand nombre d'itérations.
16. Que peut-on en conclure ?