Apprentissage Statistique et Optimisation – TD No 1

Marc Schoenauer [*] et Michèle Sebag [†]

3 octobre 2010

Exercice 1. Ecrire une fonction sauvegardée dans `Hypotenuse.m` qui prend en argument les longueurs des cotés droits d'un triangle rectangle et renvoie la longueur de l'hypoténuse.

1 Exercices sur les vecteurs et matrices

Exercice 2 (A la découverte de `find`)**.** Découvrir par vous même l'instruction `find` en testant les lignes suivantes :

```
>> A=rand(1,5)
>> find(A>0.5)
>> find(A)
>> find(A==0.2)
```

puis tester et comprendre les deux lignes suivantes

```
A=[1 2 -1 -1; -1 1 0 3]; find(A>0)
A=[1 2 -1 -1; -1 1 0 3]; [r,c,v]=find(A>0)
```

Exercice 3. On note A, B et C les matrices suivantes

$$A = \begin{pmatrix} 1 & 3 & 2 \\ -5 & 3 & 1 \\ -10 & 0 & 3 \\ 1 & 0 & -2 \end{pmatrix}, B = \begin{pmatrix} 1 & -2 & 5 \\ 6 & 1 & -1 \end{pmatrix}, C = \begin{pmatrix} 10 & -5 \\ 3 & 1 \end{pmatrix}$$

1. Calculer les matrices AB, BA et AB^T
2. Calculer $D = I_2 - BB^T$
3. Calculer les déterminants des matrices A, B, C, D et $E = AA^T$
4. Calculer l'inverse des matrices A, B, C, D, E
5. Calculer les valeurs propres de la matrice E ainsi que celle de A

Exercice 4. Considérons les matrices

$$A = \begin{pmatrix} 1 & -1 & 7 \\ -4 & 2 & 11 \\ 8 & 0 & 3 \end{pmatrix}, B = \begin{pmatrix} 3 & -2 & -1 \\ 7 & 8 & 6 \\ 5 & 1 & 3 \end{pmatrix}$$

Que font les instructions suivantes

```
3*A; A.*B; A./B; cos(A); exp(B);
```

[*] INRIA Saclay-Ile-de-France, marc.schoenauer@inria.fr
[†] LRI - UMR CNRS 8623, michele.sebag@lri.fr

Attention : à partir d'ici, travaillez à partir de scripts, que vous pourrez conserver pour un usage ultérieur - et paramétrez vos scripts autant que faire se peut.

Exercice 5.
1. Résoudre dans Octave par deux méthodes différentes le système

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & -2 & 4 \\ 0 & -2 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 2 \\ 7 \\ 3 \end{pmatrix} \tag{1}$$

2. Créer artificiellement (e.g. de manière aléatoire) des matrices et des seconds membres de plus en plus grands, et appliquer les 2 méthodes en les chronométrant. Que constate-t-on ?
3. Tracer les courbes des temps de calcul de chacune des méthodes en fonction de la taille des matrices.
4. (optionnel) Dans quelle mesure le temps de calcul dépend-il des contenus des matrices ? Essayer de quantifier votre réponse si vous avez déjà une idée de ce que sont des moyennes et des écarts-types.
5. (optionnel) Refaire l'étude dans le cas où l'on ne considère que des matrices ayant une structure particulière (par exemple, des matrices triangulaires, tridiagonales, ...) ?

2 Approximation de $\frac{\pi}{4}$ par Monte-Carlo

On désire approximer π par méthode de Monte-Carlo. Un point tiré uniformément dans $[0,1] \times [0,1]$ est dans le quart de disque vérifiant $x^2 + y^2 \leq 1$ avec probabilité $\frac{\pi}{4}$.

Exercice 6.
1. Ecrire une fonction qui :
   - prend en argument un entier N,
   - tire N points indépendemment et uniformément sur $[0,1] \times [0,1]$,
   - affiche en rouge les n_1 points dans le quart de disque de rayon 1,
   - affiche en bleu les autres points,
   - retourne le rapport $\frac{n_1}{N}$.
2. Afficher l'évolution de $\frac{n_1}{N}$ en fonction de N.

3 Générateur suivant la loi multinomiale

La loi multinomiale est une généralisation de la loi de Bernouilli [1]. Elle prend pour paramètre $(p_i)_{1 \leq k}$ avec $\forall i \quad p_i \leq 0$ et $\sum_i p_i = 1$.

Une variable aléatoire suivant la loi multinomiale de paramètre (p_i) prend la valeur $j \in \{1, \ldots, k\}$ avec probabilité p_j.

Exercice 7.
1. Ecrire une fonction prenant en paramètre un entier N et un vecteur p, renvoyant un vecteur dont les N coordonnées sont indépendantes et suivent la loi multinomiale de paramètre (p_i).
2. Vérifier le comportement de votre générateur grace à un histogramme et faire varier N.

1. Attention, il s'agit d'un abus de langage utilisé en Machine Learning. Pour les statisticiens, la loi multinomiale généralise la loi binomiale, et a donc deux paramètres, n et (p_i).

4 Régressions linéaires

Etant donné un ensemble de $N \in \mathbb{N}$ exemples $(x_i, y_i) \in (\mathbb{R}^d \times \mathbb{R})^N$ ($d \in \mathbb{N}^*$ est la dimension de l'espace des attributs), le problème général de la régression est de trouver une fonction f de $\mathbb{R}^d$ dans $\mathbb{R}$ qui minimise dans une certaine classe de fonctions l'erreur au sens de moindres carrés

$$Err(f) = \sum_{i=1}^{i=N} (f(x_i) - y_i)^2$$

La régression linéaire s'intéresse au cas où la classe considérée est celle des fonctions linéaires, c'est-à-dire qui s'écrivent

$$f(x) = \langle w, x \rangle + b$$

pour un vecteur $w \in \mathbb{R}^d$ et un scalaire $b \in \mathbb{R}$

Exercice 8. On s'intéressera dans un premier temps au cas où $d = 1$. Dans ce cas, on recherche deux paramètres réels a et b qui minimisent la quantité

$$Err(w, b) = \sum_{i=1}^{i=N} (w * x_i + b - y_i)^2$$

Il s'agit d'une fonction quadratique en fonction de w et b, elle est minimale pour les valeurs de w et b qui annulent le gradient ($\frac{\partial Err}{\partial w}$, $\frac{\partial Err}{\partial b}$).

1. Calculer le gradient explicitement, et résoudre le système de 2 équations à 2 inconnues pour trouver les valeurs optimales de w^{opt} et b^{opt}.
2. Générer des données artificielles en utilisant une fonction simple connue, par exemple $y = x^2$ sur $[0, 10]$ ou sur $[-10, 10]$, ou $y = sin(x)$ sur $[-\frac{\pi}{2}, \frac{\pi}{2}]$ puis sur $[-\pi, \pi]$
3. Ecrire le programme Octave qui calcule les coefficients w^{opt} et b^{opt} de l'approximation linéaire des données, ainsi que l'erreur d'approximation Err
4. Visualiser sur un même graphique les données (des points) et l'approximation (une droite). N'hésitez pas à essayer d'autres valeurs de w et b pour vous convaincre que ce sont bien les bonnes, à la fois via le calcul de l'erreur et visuellement.
5. Rajouter du bruit à vos données, et refaire les mêmes expériences. Plusieurs modèles de bruit sont possibles, additif ($y_i = f(x_i) + \mathbb{U}([0, \varepsilon])$) ou multiplicatif ($y_i = f(x_i)(1 + \mathbb{U}([0, \varepsilon]))$, pour un réel $\varepsilon > 0$, la notation $\mathbb{U}(I)$ représentant une réalisation d'une variable aléatoire uniforme sur l'intervalle I.
6. Faire varier le nombre d'exemples N, l'intensité et le type de bruit ε, et observer la variation de w^{opt} et b^{opt} ainsi que celle de l'erreur associée.

5 Régressions polynomiales

En fait, toutes les régressions linéaires précédentes peuvent se faire dans MATLAB par ... un simple appel à la fonction prédéfinie `polyfit` !
De plus, la fonction `polyfit` permet également d'effectuer des approximations des données par des polynomes, en spécifiant le degré du polynome.

Exercice 9.

1. Trouver et afficher les approximations polynomiales des données artificielles utilisées en dimension 1.

2. Augmenter à nouveau le niveau de bruit, pour un petit nombre de données, et observer à nouveau les comportements de l'erreur et de la fonction approximante lorsque vous augmentez le degré de l'approximation. Que constate-t-on ?

Exercice 10. Effectuer des régressions polynomiales à degrés divers des données que vous avez récupérées dans l'exercice 5 et visualisez les résultats sur un même graphique. Y a-t-il un meilleur degré d'approximation ?

6 Ensemble d'apprentissage et ensemble de test

On va s'intéresser maintenant à la **généralité** de ce qui est appris. Pour cela, nous allons utiliser deux ensembles d'exemples créés par le même générateur, et allons estimer la qualité de généralisation de la méthode d'apprentissage en apprenant une fonction d'approximation sur l'un des ensembles d'exemples (dit ensemble d'apprentissage), et en mesurant son erreur sur l'autre (dit ensemble de test), qui n'a pas été "vu" durant l'apprentissage.

Exercice 11.
1. Générer 2 ensembles d'exemples.
2. Faire une approximation polynomiale en n'utilisant que l'un des 2 ensembles, et mesurer l'erreur d'approximation sur l'autre.
3. Ajouter du bruit, et varier le degré de l'approximation. Tracer, pour un niveau de bruit donné, la variation de l'erreur sur l'ensemble d'apprentissage et de l'erreur sur l'ensemble de test sur le même dessin. Que constate-t-on ?
4. (Optionnel) Refaire la même expérience en partant de plusieurs ensembles d'apprentissage et de test, et observer la variation de l'erreur de test.

7 Classification linéaire

On va maintenant s'intéresser à un problème de classification – qui peut dans un premier temps être vu comme un autre problème de régression dans le cas particulier où les y_i ne prennent que les valeurs -1 ou 1. A noter cependant que, bien que l'on continue à utiliser la minimisation de l'erreur au sens des moindres carrés pour trouver la fonction d'approximation, l'objectif ici est de minimiser le nombre d'erreurs de classification : si f est la fonction d'approximation, on classera chaque exemple dans la classe du signe de $f(x)$, et on comptera le nombre de mal classés (ce qui n'empêchera pas de continuer à observer l'erreur au sens des moindres carrés).

Par ailleurs, pour générer les données, nous allons utiliser un procédé un peu différent, en définissant a priori des régions de l'espace $\mathbb{R}^d$ qui contiendront respectivement les x_i de valeur -1 et 1.

Exercice 12. Dans le cas de la dimension d quelconque, on peut refaire une analyse semblable à celle qui a été faite en dimension 1 pour déterminer les valeurs des coordonnées du vecteur $w^{opt} \in \mathbb{R}^d$ et le scalaire b qui minimise

$$Err(w, b) = \sum_{i=1}^{i=N} (\langle w, x_i \rangle + b - y_i)^2 \tag{2}$$

1. Dans le cas de la dimension 2, écrire le système d'équations 3×3 que satisfont w_1^{opt}, w_2^{opt} et b^{opt} qui minimisent (2).

2. Ecrire le programme qui génère des données des 2 classes à partir d'une fonction φ des coordonnées des points du plan, commençant par des polynomes de petit degré. Les exemples positifs sont donc les points (x, y) pour lesquels $\varphi(x, y) > 0$ (et réciproquement). On pourra compliquer la tâche du classifieur linéaire en augmentant graduellement le degré, voire en choisissant d'autres familles de fonctions. En l'absence d'imagination, on pourra prendre par exemple $\varphi(x, y) = 10x + y^3$ sur $[-10, 1-]^2$.
3. Visualisez les isovaleurs des fonctions, et la courbe séparatrice des deux classes (correspondant à $\varphi(x, y) = 0$) pour vérifier visuellement la difficulté de la tâche pour un classifieur linéaire (utilisez la fonction `contour`). Ajouter les exemples avec des symboles différents (ordre `plot`).
4. Programmer la résolution du système 3×3 de la première question, et en déduire w_1^{opt}, w_2^{opt} et b^{opt}.
5. Calculer l'erreur de classification (nombres de mal classés), et tracer sur un même graphique la droite séparatrice ($w_1x_1 + w_2x_2 + b = 0$), les exemples, et la courbe exacte.
6. A nouveau, on va introduire du bruit dans les données, mais ici le bruit sera lui aussi booléen : un certain pourcentage d'exemples d'apprentissage sera dans la mauvaise classe. Ecrire la procédure qui bruite les données avec une probabilité $p < 1$ et étudiez l'évolution de la classification en fonction du bruit (la probabilité p).

Apprentissage Statistique et Optimisation

Complément TD No 2 et préparation TD No 3

8 La base MNIST

Vous allez enfin pouvoir toucher du doigts des vraies données ! La base MNIST contient 60000 chiffres manuscrits discrétisés en matrices de 28×28 pixels en niveaux de gris (0-255). Elle est disponible à l'URL `http://yann.lecun.com/exdb/mnist/` et a été utilisée dans d'innombrables travaux d'apprentissage (voir les références à l'URL ci-dessus). Il s'agit de reconnaître le chiffre à partir de la matrice des pixels – et même à l'oeil, ce n'est pas si facile. Mais cette base a surtout, par rapport à des bases de données "réelles", l'énorme avantage d'être déjà pré-traitée et nettoyée (attendez de voir des vraies données pour comprendre combien cela fait gagner de temps).

Exercice 13. Les données sont stockées dans 4 fichiers, disponibles dans le répertoire `/home/lri/marc/Matlab/MNIST`, et dont la structure est décrite en détail dans la page sus-mentionnée. Il y a un ensemble d'apprentissage (les matrices de pixel dans un fichier, les labels dans un autre), et un ensemble de test (même chose).

1. Ecrire un programme qui lit les fichiers, et les stocke en mémoire. On utilisera la fonction `fread` (après `fopen`), sachant que la taille des entiers codés en début de fichiers est `int32` et les taille des pixels est `uchar`. On pourra utilement faire appel à des tableaux de cellules, permettant de stocker facilement des tableaux de matrices en mémoire. Attention, vu la taille des données, prévoir de ne lire qu'une petite partie de celles-ci via un paramètre passé à la routine de lecture.
2. Ecrire un programme qui affiche quelques chiffres (par exemple en tableau avec `n` lignes et `c` colonnes).
3. Par rapport à l'exercice 12, nous nous trouvons ici devant un problème à 10 classes. Pour ne pas compliquer les choses, nous n'allons utiliser que 2 de ces classes, que nous essaierons de séparer. Ecrire le programme qui extrait de la base les classes correspondant aux chiffres 4 et 9. Les labelliser -1 et 1 pour se ramener au problème précédent.
4. Appliquer la technique de régression linéaire développée dans l'exercice 12 et mesurer les erreurs en apprentissage et en test.

Il existe une manière plus efficace de calculer l'hyperplan séparateur optimal qui sera vue en cours lors du cours sur les méthodes à base de noyaux.