

# Apprentissage statistique

## TP1 : descente de gradient et perceptron

Olivier Schwander <olivier.schwander@lip6.fr>

2020-2021

### Exercice 1 *Descente de gradient*

On cherche à minimiser une fonction  $F : \mathbb{R}^d \rightarrow \mathbb{R}$ , dérivable sur  $\mathbb{R}^d$ . On note  $\nabla F$  son gradient.

La descente de gradient correspond à l'algorithme itératif suivant :

- **Initialisation**  $x_0 \in \mathbb{R}^d$
- **Itération**  $x_{k+1} = x_k - \alpha \nabla F(x_k)$

L'itération est répétée jusqu'à ce qu'un critère d'arrêt soit atteint.

#### Question 1

Proposer des critères d'arrêt.

#### Question 2 *Python*

Implanter cet algorithme en Python. On définira une fonction qui prendra entre autre en argument la fonction  $F$  et son gradient  $\nabla F$ .

#### Question 3 *Convexité*

Que se passe-t-il si la fonction  $F$  n'est pas convexe ?

#### Question 4

Discuter (informellement) l'influence du point  $x_0$ , d'abord dans le cas non convexe puis dans le cas convexe.

### Exercice 2 *Perceptron*

Un perceptron à  $n$  entrées  $x_1 \dots x_n$  et une sortie est défini par  $n + 1$  constantes : le vecteur de poids synaptiques  $w_1 \dots w_n$  et le biais  $\theta$ . Le potentiel  $F(x)$  de la sortie est  $F(x) = \sum_i w_i x_i$ . L'activation  $O$  de la sortie vaut 1 si  $\sum_i w_i x_i > \theta$  et 0 sinon.

#### Question 1 *Séparation linéaire*

Montrer qu'un perceptron divise l'espace des entrées en deux sous-espaces séparés par un hyperplan. Que représente le vecteur  $w$  ?

#### Question 2

Dans la définition précédente, le biais  $\theta$  est présenté comme un hyper-paramètre. Montrer que l'on peut traiter  $\theta$  comme un poids synaptique particulier. Comment adapter le problème de l'estimation des paramètres du perceptron ?

### Question 3

Le perceptron ne peut classifier correctement que des classes séparées par un hyper-plan (dites *linéairement séparables*). Donner un exemple (simple) d'un jeu de points non-linéairement séparable.

### Question 4

Étant donné deux classes linéairement séparables, combien de points faut-il rajouter pour les rendre non linéairement séparables ?

### Question 5

Calculer la distance entre un point et l'hyperplan.

## Exercice 3 *Algorithmes*

### Un premier algorithme

- **Entrée** : un échantillon de  $N$  observations  $X = \{x^{(1)} \dots x^{(N)}\}$  et les étiquettes associées  $T_1, \dots, T_N$  ( $-1$  ou  $+1$ ).
  - $t = 0$
  - $w(t) = 0$
  - **Répéter**
    - Choisir un échantillon :  $x^{(k)} = (x_1, \dots, x_d)$  et son étiquette  $T_k$
    - Calculer la sortie :  $O(x^{(k)}) = h(\sum_i w_i x_i - \theta)$
    - **Si**  $O \neq T_k$  :
      - Mettre à jour les poids :  $w_i(t+1) = w_i(t) + (T_k - O)x_i$
- Jusqu'à la présentation de tous les échantillons et l'absence de changement dans les poids

La fonction  $h$  est la fonction de Heaviside :

$$h(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ -1 & \text{sinon} \end{cases}$$

Comme dans l'exercice précédent, on notera  $F(x) = \sum_i w_i x_i$ .

### Question 1

Que se passe-t-il si l'échantillon n'est pas linéairement séparable ?

### Méthode des moindres carrés ou de Widrow-Hoff

On redéfinit  $O(x)$  comme étant  $O(x) = F(x) - \theta$  (c'est donc un réel, et non plus une valeur binaire).

On définit l'erreur pour l'entrée  $x$  :

$$e(x) = \frac{1}{2}(T_x - O(x))^2$$

et l'erreur globale

$$E(X) = \frac{1}{|X|} \sum_x e(x)$$

Pour estimer les poids synaptiques, on cherche à minimiser l'erreur globale.

La règle de mise à jour devient :  $w_i(t+1) = w_i(t) + \epsilon (T_x - O(x)) x_i$  (où  $\epsilon$  est un paramètre).

**Question 2**

Montrer qu'il s'agit d'une descente de gradient (stochastique).

**Exercice 4 *Programmation***

On cherche à implanter en `Python` la méthode des moindres carrés.

Dans un premier temps, on pourra tester les fonctions sur le jeu de données suivant :

$x_1$	$x_2$	Étiquette : $x_1 \vee x_2$
0	0	-1
0	1	1
1	0	1
1	1	1

Dans un deuxième temps, on utilisera des données artificielles (voir Question ??).

Dans un troisième temps, on utilisera un jeu de données réel, la base de chiffres manuscrits USPS (voir partie ??).

**Question 1**

Écrivez une fonction prenant en argument le vecteur de poids, une observation et l'étiquette associée et réalisant la mise à jour du vecteur (on pourra modifier le vecteur des poids).

**Question 2**

Écrivez une fonction qui réalise la boucle et qui renvoie le vecteur des poids. Quelle condition d'arrêt choisir ?

**Question 3**

Écrivez une fonction qui prédit l'étiquette d'une entrée.

**Question 4**

Modifier la fonction précédente pour afficher l'erreur globale au cours des itérations.

**Question 5**

Tracer la courbe de l'erreur globale (éventuellement en modifiant le code pour stocker les valeurs de l'erreur).

**Question 6**

Afficher sous forme d'image le vecteur des poids. Générer la séquence d'images correspondants aux itérations.

**Question 7**

Écrire une fonction pour calculer la précision de la classification.

**Question 8**

Évaluer le perceptron sur des données artificielles.

On pourra générer des données aléatoirement de la façon suivante :

```

In [1]: import numpy as np

In [2]: data0 = np.random.randn(10, 2) + 3
In [3]: data1 = np.random.randn(10, 2) - 3

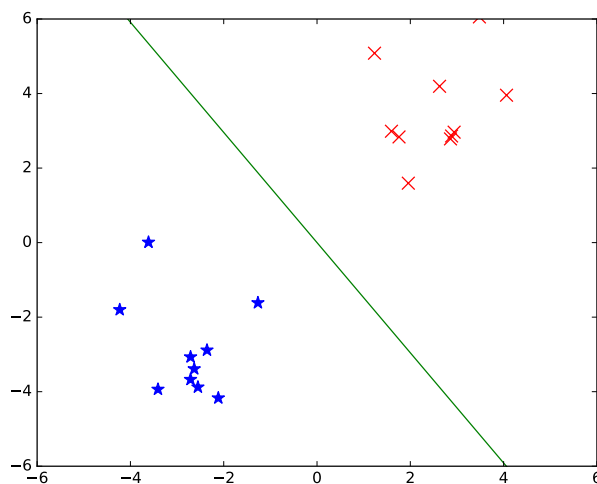
In [4]: labels0 = np.ones(10)
In [5]: labels1 = -np.ones(10)

In [6]: data = np.concatenate([data0, data1])
In [7]: data.shape
Out[7]: (20, 2)

In [8]: labels = np.concatenate([labels0, labels1])
In [9]: labels.shape
Out[9]: (20,)

```

On pourra générer plus de points, augmenter la dimension, tracer les points générés, etc. Par exemple :



## Exercice 5 *Jeu de données USPS*

Les chiffres sont représentées par des images en niveau de gris de taille  $16 \times 16$ .

Les fichiers sont disponibles sur la page <https://web.stanford.edu/~hastie/ElemStatLearn/data.html> (rubrique *ZIP code*). On trouve une description des données ainsi qu'une base d'apprentissage (*Training*) et une base de test (*Test*). On effectuera l'estimation des paramètres sur la base d'apprentissage et l'évaluation de performances sur la base de test.

Les données sont formatées comme suit :

CHIFFRE PIXEL1 PIXEL2 PIXEL3 ... PIXEL256

Par exemple :

```

6.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -0.6310 0.8620 -0.1670 -
1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -
1.0000 -1.0000 -0.9920 0.2970 1.0000 0.3070 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -
1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -0.4100 1.0000 0.9860 -0.5650 -
1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -
1.0000 -0.6830 0.8250 1.0000 0.5620 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -
1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -0.9380 0.5400 1.0000 0.7780 -0.7150 -1.0000 -

```

```

1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 0.1000 1.0000 0.9220
0.4390 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -
1.0000 -1.0000 -0.2570 0.9500 1.0000 -0.1620 -1.0000 -1.0000 -1.0000 -0.9870 -0.7140 -
0.8320 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -0.7970 0.9090 1.0000 0.3000 -0.9610 -
1.0000 -1.0000 -0.5500 0.4850 0.9960 0.8670 0.0920 -1.0000 -1.0000 -1.0000 -1.0000 0.2780 1.0000 0.8
0.8240 -1.0000 -0.9050 0.1450 0.9770 1.0000 1.0000 1.0000 0.9900 -0.7450 -1.0000 -
1.0000 -0.9500 0.8470 1.0000 0.3270 -1.0000 -1.0000 0.3550 1.0000 0.6550 -0.1090 -
0.1850 1.0000 0.9880 -0.7230 -1.0000 -1.0000 -0.6300 1.0000 1.0000 0.0680 -0.9250 0.1130 0.9600 0.30
0.8840 -1.0000 -0.0750 1.0000 0.6410 -0.9950 -1.0000 -1.0000 -0.6770 1.0000 1.0000 0.7530 0.3410 1.0
0.9420 -1.0000 -1.0000 0.5450 1.0000 0.0270 -1.0000 -1.0000 -1.0000 -0.9030 0.7920 1.0000 1.0000 1.0
0.6300 -1.0000 -1.0000 -1.0000 -1.0000 -0.4520 0.8280 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.00
1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -0.4830 0.8130 1.0000 1.0000 1.0000 1.0000 1.0000 1.00
0.9430 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -0.9740 -0.4290 0.3040 0.8230 1.0000
0.4740 -0.9910 -1.0000 -1.0000 -1.0000 -1.0000

```

### Question 1

Écrire une fonction qui charge les données.

On utilisera la fonction Python `loadtxt` disponible dans le module `numpy` :

```
In [1]: import numpy as np
```

```
In [2]: train = np.loadtxt("zip.train.gz")
```

```
In [3]: train.shape
```

```
Out[3]: (7291, 257)
```

### Question 2

Évaluer le perceptron sur ces données.