Deep Learning

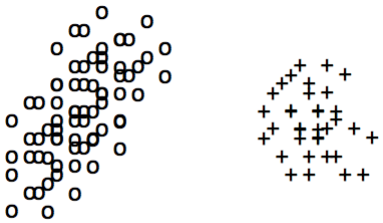
TP 1 : Introduction aux réseaux de neurones

Nistor Grozavu

nistor.grozavu@lipn.univ-paris13.fr

Note. Le TP doit être effectué en Python en utilisant numpy et scikit-learn. Préparez un rapport.

1. Générez un ensemble de données 2-D (voir figure ci-dessous) qui est composé de deux classes (notées -1 et 1) linéairement séparables d'objets répartis uniformément. Générez 50 objets pour chaque classe. Un moyen facile est de faire cela pour les coordonnées x et y séparément et les combiner ensuite. Étiquetez les variables par «area» et «perimeter».



2. Vérifiez le résultat obtenu en utilisant *scatter* et en récupérant les étiquettes des données et les étiquettes des variables.on va utiliser une matrice

3. On considère un réseau de type *Adaline*. On veut apprendre l'ensemble d'associations entre les données et les classes. L'algorithme d'apprentissage est le suivant (voir le cours) :

1- Tirer au hasard w_0

2- Présenter une forme $\left(x^k, d^k\right)$

3- Déterminer la valeur de l'écart

$$e^k(t) = \left(d^k - wx^k\right)$$

4- Calculer une approximation du gradient

$$\nabla_w \tilde{R}^k_{Adaline}(w) = -2e^k(t)x^k$$

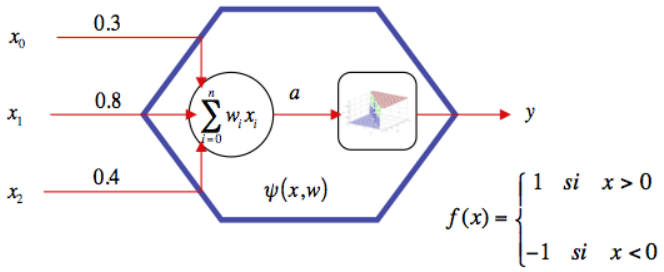
5- Adapter les poids $w(t)$

$$w(t+1) = w(t) - \varepsilon(t)\nabla_w \tilde{R}^k_{Adaline}(w)$$

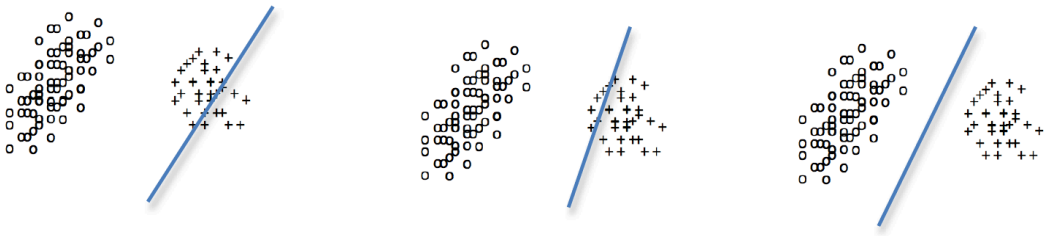
Où $\varepsilon(t)$ est le pas du gradient.

6- Répéter de 2 à 4 jusqu'à l'obtention d'une valeur acceptable de l'erreur

4. Ecrire une fonction **err_adal** permettant de calculer l'écart entre la sortie désirée (classe) et la sortie calculée par Adaline.
5. Ecrire une fonction **grad_adal** permettant de calculer une approximation du gradient.
6. Ecrire une fonction **adapt_adal** permettant de mettre à jour les poids des connexions dans *Adaline* en fonction du gradient.
7. Ecrire une fonction **Adaline** utilisant les fonctions précédentes permettant de faire un apprentissage du modèle *Adaline* sur la base de données générée précédemment. On utilisera la fonction de transition indiquée dans la figure ci-dessous, ainsi que les valeurs initiales des poids et un pas du gradient de **ε=0.1**.



8. A chaque itération de l'apprentissage représenter graphiquement dans un même repère orthogonal l'ensemble des échantillons et l'hyperplan séparant les deux classes.



à t=0 à t=10 à t=50

9. On propose de traiter les mêmes données avec cette fois le Perceptron dont l'algorithme est donné ci-dessous :

Algorithme d 'apprentissage : cas de 2 classes

1- A t=0, tirer au hasard $w(0) = \left(w_0(0), w_1(0), \ldots, w_n(0)\right)^T$

2- Présenter une forme $\left(x^k, d^k\right)$

3- Calculer la sortie du perceptron et la comparer à y^k

$$f\left(w_0 + \sum_{i=1}^{n} w_i \varphi_i(x)\right) = f\left(w^T \varphi^k\right)$$

4- Si x^k est bien classé : $f\left(w^T \varphi^k\right) = d^k$

$$w(t+1) = w(t)$$

Si x^k est mal classé : $f\left(w^T \varphi^k\right) \neq d^k$

$$w(t+1) = w(t) + \varepsilon(t)\, \varphi^k\, d^k$$

$$\begin{pmatrix} w_0(t+1) \\ w_1(t+1) \\ \vdots \\ w_n(t+1) \end{pmatrix} = \begin{pmatrix} w_0(t) \\ w_1(t) \\ \vdots \\ w_n(t) \end{pmatrix} + \varepsilon(t)\, d^k \begin{pmatrix} 1 \\ \varphi_1(x^k) \\ \vdots \\ \varphi_n(x^k) \end{pmatrix}$$

$$\varphi^k = \left(1, \varphi_1(x^k), \varphi_2(x^k), \ldots, \varphi_n(x^k)\right)$$

Où $\varepsilon(t)$ est le pas du gradient.

5- Répéter de 2 à 4 jusqu'à l'obtention d'une valeur acceptable de l'erreur

Ecrire une fonction **sortie_perc** permettant de calculer la sortie calculée par le Perceptron.

10. Ecrire une fonction **grad_perc** permettant de calculer une approximation du gradient.

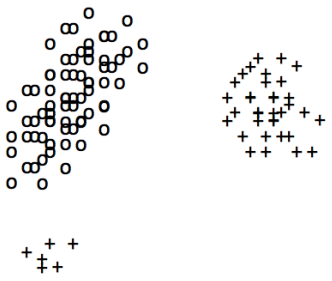
11. Ecrire une fonction **adapt_perc** permettant de mettre à jour les poids des connexions dans Adaline en fonction du gradient.

12. Ecrire une fonction **perceptron** utilisant les fonctions précédentes permettant de faire un apprentissage du modèle *Perceptron* sur la même base de données qu'*Adaline*. On utilisera les valeurs d'initialisation données dans la question 7.

13. A chaque itération de l'apprentissage représenter graphiquement dans un même repère orthogonal l'ensemble des échantillons et l'hyperplan séparant les deux classes.

Bonus :

14. Générez un ensemble de 10 données de la deuxième classe (voir figure ci-dessous) pour compléter la base de données initiale de telle façon que le problème reste linéairement séparable.



15. Exécutez les programmes Adaline et Perceptron sur la nouvelle base de données et représentez graphiquement dans un même repère orthogonal l'ensemble des échantillons et les deux hyperplans séparant les deux classes : celui d'Adaline et celui du Perceptron