Intelligence Artificielle
Réseaux de neurones

Mathieu Lefort

12 & 19 mars 2018

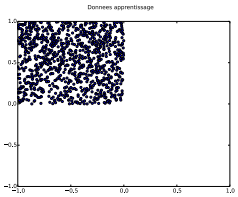
Ce TP peut être fait en binôme. Vous rendrez à la fin du TP et d'ici le 25/03 (respectivement le 01/04) pour le groupe 1 (respectivement 2) un compte rendu contenant le code (commenté) de votre programme et les réponses aux questions. Pensez à bien indiquer vos noms dans le fichier.

1 Objectif

L'objectif est d'implémenter et de comprendre l'algorithme de carte auto-organisatrice de Kohonen.

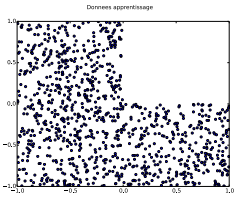
2 Données

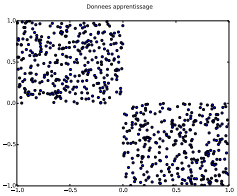
Vous avez à disposition plusieurs jeux de données :



— Des données distribuées uniformément dans le quadrant $[-1, 0] \times [0, 1]$:

— Des données distribuées uniformément dans les quadrants $[-1, 0] \times [0, 1]$, $[-1, 0] \times [-1, 0]$ et $[0, 1] \times [-1, 0]$:





— Des données distribuées uniformément dans les quadrants $[-1, 0] \times [0, 1]$ et $[0, 1] \times [-1, 0]$:

— La base de donnée MNIST qui contient des images de chiffres manuscrit (image de taille 28*28) : un exemple du chiffre 5 

3 Algorithme

Comme vu en cours, l'algorithme de Kohonen est le suivant :

— Initialisation

1. Initialisation des poids (=prototypes) à des valeurs aléatoires
2. Choix de la topologie de la carte [1]
3. Choix de la largeur du voisinage gaussien σ
4. Choix du taux d'apprentissage η

— Apprentissage
Pendant N pas de temps

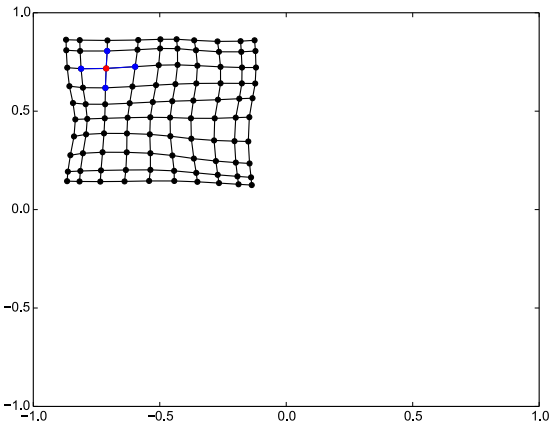
1. Choix d'une nouvelle entrée $\mathbf{X}$

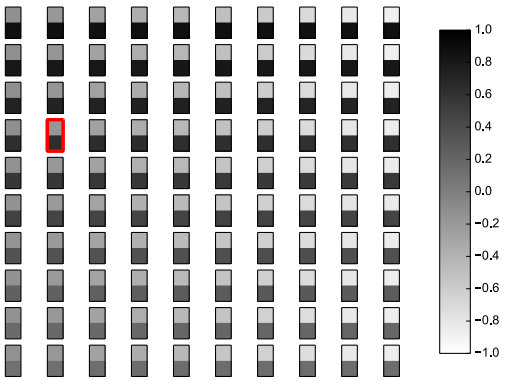
1. Dans un premier temps, vous prendrez une grille carrée comme vu en cours.

2. Pour chaque neurone j, calcul de la distance entre son prototype $\mathbf{W_j}$ et $\mathbf{X}$
3. Détermination du neurone gagnant j^*, i.e. le plus proche de l'entrée courante
4. Pour chaque neurone j, modification des poids de toutes ses connexions : $\Delta w_{ij} = \eta e^{-\frac{||j-j^*||^2}{2\sigma^2}}(x_i - w_{ij})$

Un fichier *kohonen.py* vous est fourni et contient une partie affichage et le squelette de l'algorithme. Vous pouvez lancer le fichier avec la commande suivante : *python kohonen.py*. Vous privilégierez l'utilisation de la librairie de calcul matriciel numpy qui fournit des performances d'exécutions bien supérieures à l'utilisation des boucles *for* en python [2].

Les affichages fournis sont les suivants :





Sur la figure de gauche (dans l'espace d'entrée), le poids de chaque neurone est représenté par un point et ses voisins dans la carte sont représentés par un lien entre leurs poids respectifs. Par exemple, le neurone à la position (3,3) dans la carte a les poids représentés par le cercle rouge (ici [-0.6,0.7]) et les neurones aux positions (4,3), (2,3), (3,4) et (3,2) ont les poids indiqués par les cercles bleus.

Sur la figure de droite (dans l'espace de la carte) sont affichés les poids de chaque neurone. Par exemple, le rectangle rouge indique les poids du neurone à la position (4,2) dans la carte, la valeur des poids étant indiquée en niveau de gris (ici [-0.2,0.8]))

Le programme calcule également l'erreur de quantification vectorielle faite par le réseau sur le jeu d'apprentissage $\{\mathbf{X}\} : \frac{1}{|\{\mathbf{X}\}|}\sum_{\mathbf{X}\in\{\mathbf{X}\}} \min_j ||\mathbf{W_j} - \mathbf{X}||^2$.

4 Travail à fournir

Question 1 : Étude théorique du comportement de l'algorithme :

— Quelle sera la prochaine valeur du poids du neurone gagnant dans le cas où $\eta = 0$? $\eta = 1$? En déduire géométriquement la prochaine valeur du poids dans le cas normal où $\eta \in]0, 1[$.

— Si σ augmente, est-ce que les neurones vont plus ou moins apprendre l'entrée courante ? En déduire l'influence que doit avoir σ sur la "grille" de neurone, sera-t-elle plus "lâche" ou plus "serrée" si σ augmente ?

— Prenons le cas d'une carte avec un seul neurone qui reçoit 2 entrées $\mathbf{x_1}$ et $\mathbf{x_2}$. Durant l'apprentissage $\mathbf{x_1}$ (respectivement $\mathbf{x_2}$) est présenté n_1 (respectivement n_2) fois. Après l'apprentissage où se situera géométriquement le poids du neurone ?

Question 2 : Implémentez l'algorithme dans le fichier *kohonen.py* (les sections TODO vous indiquent les sections de code à modifier/écrire). Vous pouvez lancer le fichier tuto_numpy.py pour avoir une liste de commande numpy utiles ou la documentation en ligne `http://docs.scipy.org/doc/numpy-1.10.1/reference/index.html`. Le paramétrage qui vous est fourni dans le code devrait fonctionner si vous respectez les équations demandées.

Question 3 : Étude pratique du comportement de l'algorithme (à mettre en parallèle avec votre étude de la question 1). Étudiez l'influence des éléments suivants sur le fonctionnement de l'algorithme de Kohonen (qualitativement et quantitativement avec la mesure d'erreur de quantification vectorielle) :

— taux d'apprentissage η
— largeur du voisinage σ
— nombre de pas de temps d'apprentissage N
— taille de la carte
— jeu de données. En particulier créez vos propres jeux de données avec des données non uniformément distribuées pour étudier la répartition des poids des neurones. Étudiez séparemment le jeu de données MNIST.
— la topologie de la carte (Bonus). En particulier au lieu d'utiliser une grille carrée, utilisez une grille hexagonale.

Question 4 (Bonus) : Comment ce type de réseau pourrait-il être utilisé pour faire de la classification/reconnaissance des chiffres ?

2. En fait numpy est une interface python vers des algorithmes compilés en C.