Université Paris 8

Implémentation SOM

Decembre, 2021

Contents

1 Récupération des données

- Récupérer les données sur : https://archive.ics.uci.edu/ml/datasets/iris.
- Parcourir le fichier et récupérer les valeurs de chaque colonne (sepal length, sepal width, petal length, petal width, class).
- Faire un tableau de structure (De la structure vec montrer ci-dessous) et y affecter chaque ligne présent dans le data set. Cela va donc faire un tableau de 150 sructure donc chaque éléments représente une ligne du dataset.
- Pour chaque vecteur on calcule sa norme et on l'ajoute à sa structure.
- On ajoute l'étiquette (la classe de la fleur) à sa structure.
- On mélange aléatoirement les données dans notre tableau de structure.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


typedef struct vec{
    double *v; //Tableau contenant les vecteur donnee
    double norme; //Norme du tableau de vecteur
    char *etiquette; //Tableau contenant les etiquettes (label) des donnees
}vec;
```

2 Initialisation du réseau

Pour initialiser le réseau SOM il faut initialiser sa structure qui est composée de:

- Le nombre de lignes de la grille du reseau.
- Le nombre de colonne de la grille du reseau.
- Le nombre de ligne de noeud du réseau. Nombre de noeud = $5 * \sqrt{150} * 1$ (on peut faire *2 *3 *4 plus on multiplie moins on zoom sur la map soi moins de détails). 150 = nombre de lignes dans le dataset.
- La BMU (best match unit) qui au début est vide vu qu'on a pas encore lancer l'apprentissage. La BMU est composée de :
  - Son numéro de colonne.
  - Son numéro de ligne.
  - Un pointeur vers la potentielle BMU suivante (Dans le cas où il y aurait plusieurs gagant)
- La map qui est une matrice de noeud (structure node). Un noeud est composé de:
  - Son poid (weight/mémmoire) qui est le vecteur avec lequel on l'initialise. Soit un vecteur générer aléatoirement avec comme borne inférieur la valeur la plus basse de la moyenne de tous les vecteur donnée et la bornée supérieur de la même façon.
  - Sa fonction d'activation. La distance euclidienne entre le vecteur weight/mémoire et le vecteur données qu'on va passer au réseau.

  - Son id

- Le nombre d'itération =
$$5 * tailleduvecteurdonne = 500 * 4 = 2000$$

- Le taux d'appprentissage (learning rate) = Entre 0.7 et 0.9 au début puis évolue avec la formule :
$$a^(t) = a_init * (1 - t/t_total)$$
avec t = le nombre d'itération.

- La taille du voisinnage = 50 % du nombre total de noeud. Si on a 100 noeuds (Noeud = neuronnes) on a un voisinnage de 50 noeuds.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


//Structure d'un neuronne
typedef struct node{
    double *weight; //Vecteur poid
    double act; //Fonction d'activation (distance euclidienne)
    char id; //Nom de la données (classe de la données)
};

//Structure d'une best matching unit
typedef struct bmu{
    int c;  //Colonne de la bmu
    int l;  //Ligne de la bmu
    struct bmu *next; //reference vers le potentiel "gagnant" suivant
};

//Structure de la "grille" du reseau du     SOM
typedef struct net{
    int nb_colonne; //Nombre de colonne
    int nb_ligne;   //Nombre de ligne
    int nb_node; //Nombre de neuronnes
    bmu *best_unit; //Coordonées de la BMU
    struct node** map;  //Reference vers la matrice de neuronne
    int nb_iteration;   //Nombre d'iteration
    double alpha;   //Taux d apprentissage (learning rate)
    int taille_voisinnage;  //Taille du voisionnage autorisee
};
```

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL

3 Apprentissage

Maintenant que les données et le SOM sont implémenté alors on peut l'entrainer. Pour ça on fait:

- On séléctionne une donnée dans notre dataset (le tableau de structure)
- On compare le vecteur donées avec tous les noeud et on sélectionne celui dont la distance euclienne entre le vecteur mémoire et le vecteur donnée est la plus petite. C'est notre BMU.
- On met à jour le vecteur mémoire avec la formule:

$$W_{i_j}^{(}t+1) = W_{i_j}^{(}t) + a^t * NHD * x_i - W_{i_j}^t$$

- $W_{i_j}^t$ = Le vecteur mémoire
- a = Taux d'apprentissage
- NHD = fonctione de voisinnage, mais on ne l'utilise pas dans ce projet donc en soit on n'aurait pu ne pas l'écrire.
- $x_i - W_{i_j}^{(}t)$ = La différence entre le vecteur donée et le vecteur mémoire.
- Puis on injecte le vecteur mémoire dans tous les voisinnage via la formule :

$$W_{i_j}^{(}t+1) = a^t * x_i - W_{i_j}^t$$

- On fait 1/4 des itération total comme ça, puis les 3/4 restant se font avec un voisinnage qui diminue de façon à ce que les voisins du noeud soit égale à 8 (Soit les noeuds adjacents)

L'apprentissage est finis ! :)

4 Visualisation

Pour la visualisation il existe des bibliothèque graphique en C ou python pour voir la map SOM en couleur en assotiant les valeurs numériques des vecteur à des couleurs etc :).