|  |
| --- |
|  |
| GPA749 |
| Laboratoire 2 |
|  |
| **Flavien Deschaux – Adrien Vassal** |
|  |

|  |
| --- |
|  |

Table des matières

Introduction

L’objectif de ce laboratoire est d’entrainer un réseau de neurones à reconnaitre les contours d’une image.  
Dans notre cas, il s’agira d’entrainer un réseau MLP en testant différente valeur pour l’ensemble de hyper-paramètres ce celui-ci.  
Ainsi le but sera de définir l’impact de ces hyper-paramètres sur la solution obtenue et de trouver les meilleurs paramètres possibles afin que notre réseau soit le plus performant.  
Les trois paramètres sur lesquels nous allons nous concentrer sont : la constante d’apprentissage, l’apprentissage par batch, et le nombre d’époque optimal.

# Le réseau MLP

Dans le précédent laboratoire nous avons étudiés le réseau perceptron et mis en lumière son plus gros défaut : il ne peut pas classifier que des données linéairement séparables.  
Or en pratique il se trouve que les données testées ne sont (pratiquement) jamais linéairement séparable.  
C’est suite à ce besoin de trouver une méthode de classification pour des données non linéairement séparable qu’est nait le MLP ou Multi Layer Perceptron (perceptron multicouche).

## Architecture

Un MLP comme son nom l’indique est constitué de plusieurs couche. L’entrée du MLP peut être de taille variable en fonction des données du problème.  
La couche de sortie du MLP est elle aussi de taille variable, mais on impose que sa fonction d’activation soit dérivable, on utilisera la fonction sigmoïde.

Les couches situées entre l’entrée et la sortie sont appelées couches cachées. Elles sont composées d’un nombre variable de neurones de McCulloch&Pitts.

L’architecture peut être décrite par le graphique suivant :

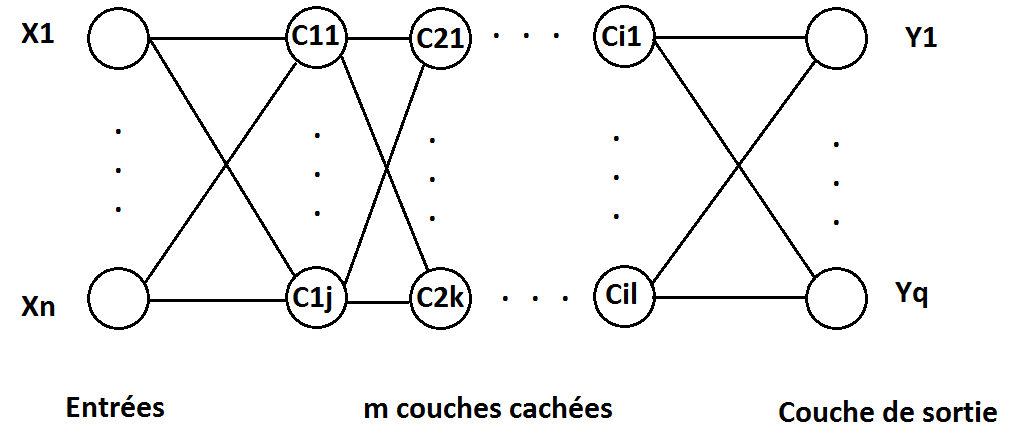


Figure : Représentation graphique d'un MLP

Sur le graphique ci-dessus :

* le nombre d’entrée est variable
* le nombre de couches cachées est variable
* le nombre de neurones par couche cachée est variable
* le nombre de sortie est variable

Le réseau est à peine crée qu’il y a déjà plusieurs paramètres à définir. Généralement ces paramètres sont à définir en fonction du problème et des données d’entrées.

## Entrainement du réseau

Pour entrainer un tel réseau il faut procéder en deux phases :

* La propagation directe

On présente une entrée au réseau, on calcul sa sortie et on en déduit l’erreur obtenue.

* La rétropropagation

C’est cette étape qui va nous permettre de définir une façon de modifier les poids synaptiques de notre réseau.

Basée sur l’algorithme delta généralisé, l’idée est de minimiser l’erreur quadratique totale (aussi appelé Mean Square Error) faite par notre réseau.

Pour ce faire nous allons réaliser la rétropropagation du gradient de l’erreur.   
Le gradient étant défini (en un point) comme la pente la plus forte qui point vers le maximum local.   
Ainsi pour minimiser l’erreur quadratique, il faut se déplacer dans le sens inverse du gradient de cette erreur.  
 Fort de ces observations, il ne reste plus qu’à exprimer l’impact de chaque neurone sur la sortie, autrement dit exprimer le gradient de l’erreur en fonction des poids synaptiques associés à chaque neurone.

(Qui dit gradient dit dérivée, ce qui explique pourquoi la couche de sortie doit avoir une fonction dérivable)

Ainsi après calcul nous avons les équations suivantes :

Où :

* k : le nombre de neurone de la couche de sortie
* j : le nombre de neurone de la couche cachée considérée
* wkj : le poids de la connexion entre le la sortie k et le neurone j
* : la variation à apporter au poids wkj
* : la variation à apporter au poids wij
* : le paramètre d’apprentissage

A noter que nous introduisons un nouveau paramètre à régler pour notre réseau : , ce paramètre sert à exprimer le fait que les poids synaptiques du réseau ne doivent pas changer du tout au tout à chaque rétropropagation .

## Notre cas

Dans notre cas nous allons utiliser un réseau possédant une unique sortie. En effet nous cherchons à savoir si un pixel est un contour ou non. Il s’agit donc d’une information binaire, ce qui explique pourquoi nous n’avons qu’une seule sortie.

Pour des questions de facilité d’implémentation, nous allons restreindre le nombre de couche cachée à 1.  
Cependant le nombre d’entrée et le nombre de neurone dans la couche cachée sont des paramètres que nous allons pouvoir modifier.

# Implémentation du réseau MLP

Deux approches sont possibles pour implémenter ce réseau :

Soit nous appliquons les formules avec des sommes et nous travaillons élément par élément, soit nous considérons des calculs matriciels.  
La deuxième solution étant bien plus optimal en termes de temps de calcul, c’est celle-là qui a été choisie.

# Expérience 1 : impacte du paramètre d’apprentissage

# Expérience 2 : impacte de l’apprentissage par batch

# Expérience 3 : visualisation du surentrainement

# Expérience 4 : impacte du surentrainement

# Expérience 5 : impacte des hyper-paramètres sur les résultats

### La construction des batch

### Nombre de neurones de la couche cachée

### Nombre de neurones de la couche d’entrée

# Bonus

George POWER !