Notes de cours de réseaux de neurones
L1 Maths-Info
(Rédaction en cours.)

Philippe Preux
Uniersité de Lille

16 février 2024

Résumé

Ce cours est une introduction aux réseaux de neurones. On s'intéresse à leur fonctionnement et à la compréhension de ceux-ci. Il existe des tas de types de réseaux de neurones différents ; on ne traite que des plus populaires, les perceptrons multi-couches. Destiné à des étudiants de 1re année universitaire, les pré-requis en mathématiques et en informatique sont réduits : en mathématiques, il faut connaître la notion de dérivée d'une fonction à une variable réelle et celle de droite dans le plan ; en informatique, il faut connaître les rudiments de la programmation (variables, tests, boucles, fonctions).

1 Introduction

Cette première section donne quelques repères généraux sur les réseaux de neurones. Par la suite, on traite des réseaux de neurones de manière technique.

1.1 Repères historiques

- En 1791, Galvani reconnait la nature électrique des signaux nerveux.
- Fin XIXe siècle, Ramon y Cajal découvre les cellules nerveuses, les neurones.
- Dans les années 1930, des mathématiciens éclaircissent la notion de calcul et de fonction calculable. A. Turing propose sa machine (abstraite) capable de calculer mécaniquement toute fonction calculable. Ces fonctions calculables traitent des nombres entiers.
- En 1943, Mc Culloch et Pitts proposent une description mathématique du fonctionnement d'un neurone.
- En 1956, le mot « intelligence artificielle » est créé lors d'un congrès.
- En 1958, Rosenblatt propose le modèle du perceptron. Le perceptron est capable de réaliser des tâches très simples, autrement dit, de calculer des fonctions mathématiques simples. On mettra ensuite 30 ans à trouver comment calculer à peu près n'importe quelle fonction (1986).

1.2 Neurone naturel et neurone artificiel

Très schématiquement, le cerveau reçoit des signaux en entrées provenant de nos sens et émet des signaux en sortie qui contrôle nos muscles et nous fait donc agir et réagir dans notre environnement.

Le cerveau est constitué de cellules nerveuses, également nommées neurones.

Le neurone artificiel est une simplification extrême du neurone réel. Très grossièrement, un neurone réel est constitué d'un noyau, il reçoit des signaux électriques sur ses entrées (dendrites), et possède une sortie ramifiée (axone). La sortie d'un neurone est connectée à des dendrites via des synapses qui constituent l'interface entre un axone et une dendrite. Lorsqu'un neurone émet une décharge électrique sur son axone (cette décharge se nomme un potentiel d'action), elle se propage dans ses ramifications jusqu'à atteindre des synapses. Selon les conditions, la décharge est transmise dans la synapse puis parcourt une dendrite jusqu'à atteindre un neurone. Typiquement, un neurone reçoit de nombreuses décharges au fil du temps qui le chargent électriquement. Si sa charge électrique dépasse un certain seuil, il décharge en émettant un potentiel d'action sur son axone, *etc.* Les rayons lumineux qui atteignent nos yeux, les sons dans nos oreilles, les sensations de toucher, goût, etc entraînent des décharges électriques vers des neurones. Cela va entraîner une cascade d'activités neuronales pour, in fine, entraîner la décharge électrique d'un ou plusieurs neurones qui vont agir sur un ou des muscles. En conditions normales, dans un animal en bonne santé, ce processus met une fraction de seconde à se réaliser. La connexion entre deux neurones est plus ou moins forte, selon la valeur du poids synaptique. Dans notre cerveau, celui-change au fil du temps, en fonction de nos expériences. Le cerveau humain comprend environ 10^{11} neurones et il existe environ 10^{15} connexions entre neurones.

1.3 Applications

Depuis les progrès réalisés en 1986, les neurones artificiels ont vu leur champ d'applications croître considérablement. On a commencé à réfléchir à la combinaison de plusieurs neurones, créant des réseaux de neurones. Vers 1990, ils sont utilisés pour reconnaître des chiffres écrits à la main et ainsi reconnaître les codes postaux sur les courriers et faire du tri automatique du courrier. Ils sont également utilisés pour réaliser un programme ayant appris à jouer au jeu de Backgammon à partir des règles du jeu et jouant au niveau des experts humains, et jouant des coups qui étonnent ceux-ci. Vers 2000, ils sont utilisés pour reconnaître les chiffres sur les chèques et ainsi lire automatiquement le montant des chèques au guichets automatiques des banques. Mis à part cela, le domaine stagne, même si les réseaux de neurones sont utilisés dans un très grand nombre d'applications. Par stagnation, j'entends que la complexité des tâches que l'on arrive à résoudre ne progresse pas. En effet, pour réaliser des tâches complexes, il faut combiner plusieurs neurones en un réseau et pendant 20 ans, on ne sait pas comment entraîner un tel réseau de neurones à calculer une certaine fonction dès que le nombre de neurones devient un peu grand. À cette époque, la puissance des ordinateurs est encore faible [1], ce qui limite la quantité de calculs que l'on peut réaliser. En étudiant avec précision les freins à l'utilisation de plus gros réseaux de neurones, la notion de réseau de neurones profonds apparaît à la fin de la décennie 2000. Avec ces plus gros réseaux, un premier résultat expérimental est obtenu en 2013 avec le réseau ImageNet qui est capable de détecter des objets dans des images avec une précision jamais vue précédemment, réalisant cette tâche en commettant moins d'erreurs que des êtres humaines confrontés à la même tâche. À la suite de ce premier résultat, les avancées vont être très rapides que ce soit dans le domaine de la détection d'objets dans des images (par exemple la génération automatique de la légende d'une image, 2015), dans le domaine des langues naturelles, ou dans le domaine des jeux (jeu de go en 2017), puis des modèles génératifs. Outre la disponibilité de capacités de calcul de plus en

1. Vers 2000, un bon ordinateur personnel de bureau est cadencé à 500 MHz et il est mono-cœur.

plus énormes, ces progrès s'appuient sur la disponibilité massive (voire le pillage massif) de données sur Internet. Devenue une technologie aussi puissante, et furtive, ces progrès ne vont pas sans leurs dangers, liés par exemple à son usage en vidéo-surveillance, dans les armes, la réalisation de faux et la manipulation de l'opinion.

1.4

Il existe de très nombreux types de neurones artificiels et de réseaux de neurones artificiels. Dans cette introduction, on ne traite que des plus connus : le neurone artificiel dénommé perceptron et le réseau de neurones le plus classique constitué d'un ensemble de perceptrons, le perceptron multi-couches.

2 Perceptron binaire à sortie binaire

Nous présentons plusieurs types de perceptrons :

— perceptron binaire à sortie binaire : entrées et sortie sont à valeur binaire.

— perceptron à sortie binaire : sortie à valeur binaire, entrées à valeur réelle (*cf.* sec. 3).

— perceptron réel : entrées et sortie à valeur réelles (*cf.* sec. 4).

Ces différents types de perceptron se ressemblent très fortement.

2.1 Définition

On définit un ensemble de termes et de notations :

— Un **perceptron binaire** est une unité de calcul constituée de P **entrées** à valeur binaire et d'une **sortie** à valeur binaire également.

— On note e_j l'entrée numéro j.

— Chaque entrée est pondérée par un **poids** à valeur réelle noté p_j.

— Il existe par ailleurs un poids particulier associée à une entrée qui vaut toujours 1. Ce poids est dénommé le **biais** et on le numérote 0, donc p_0 est le biais.

— Le **potentiel** du perceptron est noté v et il vaut : $v = p_0 + \sum_{j=1}^{j=P} p_j e_j$.

— La sortie s du perceptron est une certaine fonction notée φ de v, soit $s = \varphi(v)$. Cette fonction se nomme la **fonction d'activation** du perceptron. Dans le cas du perceptron à sortie binaire, on prendra :

$$\varphi(v) = \begin{cases} 1 \text{ si } v \geq 0, \\ 0 \text{ si } v < 0. \end{cases}$$

Cette fonction se nomme la **fonction de Heaviside**.

2.2 Exercice

On considère un perceptron à deux entrées. Calculer la sortie s pour des poids et des entrées ci-desous :

p_0	p_1	p_2	e_1	e_2	s
0.5	-1.5	2	1	0	
-1	0.25	-3	1	0	
-2	1	1	1	0	
-1	2	2	1	0	
-3	2	2	1	0	
-3	2	2	0	0	
-3	2	2	1	1	
-3	2	2	0	1	

2.3 Programmation

Programmer le calcul de la sortie d'un perceptron binaire (= TP 1 [2]).

2.4 Trouver les poids qui permettent de calculer une certaine fonction. Mise en bouche

La question essentielle que l'on se pose est : quelle valeur donner aux poids d'un perceptron pour que la sortie de celui-ci corresponde à ce que l'on veut.

2.4.1 Exercice

On considère les fonctions logiques à 2 entrées. Trouver, si c'est possible, des poids pour qu'un perceptron calcule : le ET logique, le OU logique, le NON-ET logique, le NON-OU logique, le OU-EXCLUSIF logique.

Par exemple pour le ET, on trouve en tatonnant : $p_0 = -5, p_1 = 2, p_2 = 4$, ou $p_0 = -1, p_1 = 0{,}5, p_2 = 0{,}5$.

On peut aussi réfléchir sur les contraintes à satisfaire :

— 0 et 0 $\rightarrow$ 0 : donc, $p_0 < 0$

— 0 et 1 $\rightarrow$ 0 : donc, $p_0 + p_1 < 0$

— 1 et 0 $\rightarrow$ 0 : donc, $p_0 + p_2 < 0$

— 1 et 1 $\rightarrow$ 1 : donc, $p_0 + p_1 + p_2 \geq 0$

et en déduire des poids qui vérifient ces 4 contraintes.

2.4.2 Interprétation géométrique

Le potentiel d'un perceptron à deux entrées est $v = p_0 + p_1 e_1 + p_2 e_2$. La sortie est le signe du potentiel, 1 si le potentiel est positif, 0 s'il est négatif.

Intéressons-nous à l'équation : $0 = p_0 + p_1 e_1 + p_2 e_2$.

Que l'on peut écrire : $e_2 = -\frac{p_0}{p_2} - \frac{p_1}{p_2} e_1$ qui a la forme $y = ax + b$. C'est donc l'équation d'une droite dans le plan e_1 en absicsses, e_2 en ordonnées.

Une droite $y = ax + b$ sépare le plan en deux sous-espaces : l'un correspond à $y \geq ax + b$ et l'autre à $y < ax + b$.

On peut ré-écrire : $y \geq ax + b \rightarrow 0 \geq ax + b - y$ et l'autre à $y < ax + b \rightarrow 0 < ax + b - y$.

2. Les sujets des TP sont dispnibles sur la page du cours à l'url `https://philippe-preux.github.io/ensg/l1-mi/`

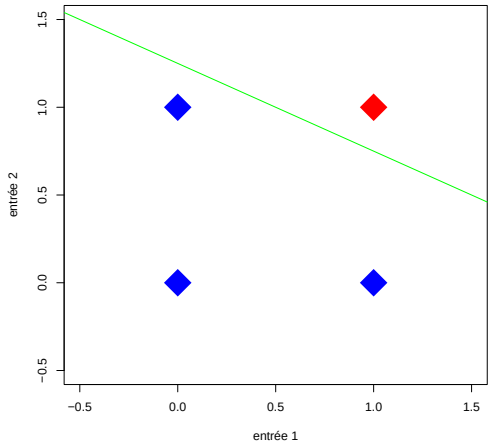


FIGURE 1 – Comme indiqué dans le texte, on a représenté les 4 exemples de la fonction ET à deux entrées, la couleur indiquant la valeur associée (0 si bleu, 1 si rouge), c'est-à-dire la sortie attendue du perceptron pour cette donnée. Et on a indiqué la droite $e_2 = \frac{5}{4} - \frac{2}{4}e_1$ qui sépare le rouge des bleus.

Reprenons l'exemple du ET logique et le jeu de poids $p_0 = -5, p_1 = 2, p_2 = 4$.
Dessinons le plan (e_1, e_2).
Plaçons-y les 4 points et colorons la valeur en ce point en bleu si c'est 0, rouge si c'est 1.
Traçons la droite correspondant aux 3 poids, soit : $e_2 = \frac{5}{4} - \frac{2}{4}e_1$.
Les points bleus sont d'un côté de la droite, le(s) rouge(s) de l'autre. Voir la figure 1.

2.4.3 Exercice

Appliquer la même démarche aux fonctions OU, NON-ET, NON-OU. Que se passe-t-il pour le OU-EXCLUSIF ?

OU-EXCLUSIF : géométriquement (*cf.* fig. 2), on voit que quelle que soit la droite tracée, on n'arrivera jamais à séparer les 0 des 1.

Algébriquement, on peut écrire le système d'inégalités :

1. 0 et 0 $\rightarrow$ 0 : donc, $p_0 < 0$
2. 0 et 1 $\rightarrow$ 0 : donc, $p_0 + p_1 \geq 0$
3. 1 et 0 $\rightarrow$ 0 : donc, $p_0 + p_2 \geq 0$
4. 1 et 1 $\rightarrow$ 1 : donc, $p_0 + p_1 + p_2 < 0$

En ajoutant 1. et 4., on obtient $2p_0 + p_1 + p_2 < 0$.
En ajoutant 2. et 3., on obtient $2p_0 + p_1 + p_2 \geq 0$.
Ces deux inégalités ne peuvent pas être satisfaites en même temps, il n'y a donc pas de solution.

2.5 Exercice

Afficheur 7 LEDs : celui-ci est utilisé pour afficher un chiffre :

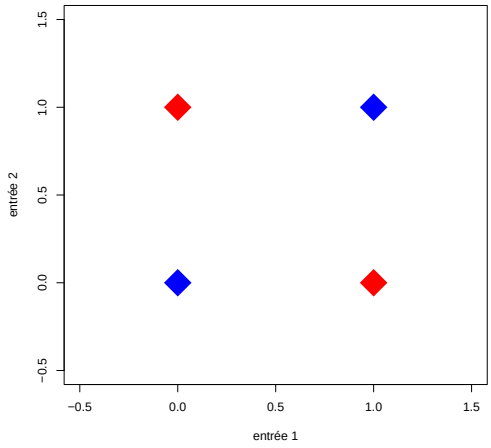
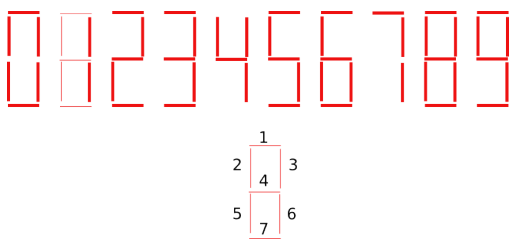


FIGURE 2 – On a représenté les 4 exemples de la fonction OU-EXCLUSIF à deux entrées, la couleur indiquant la sortie attendue du perceptron (0 si bleu, 1 si rouge) pour cette donnée. Impossible de tracer une droite qui sépare les points bleus des points rouges.



On suppose que chaque entrée d'un perceptron est associée à un segment de l'afficheur et qu'elle reçoit 0 si le segment est éteint, 1 s'il est allumé.

Peut-on trouver un perceptron à 7 entrées qui :

— détermine que le chiffre affiché est 6 ?

— détermine la parité du chiffre affiché ?

— détermine que le chiffre affiché est < 5 ?

— détermine que le chiffre affiché est $\in \{3, ..., 7\}$?

2.6 Méthode pour calculer les poids d'un perceptron binaire

On peut écrire un système d'inégalités : dans le cas des fonctions logiques plus haut, tous les cas sont énumérables, il y en a 2^n si on considère les fonctions logiques ayant n entrées. Cependant, résoudre ce genre de système d'inégalités n'est pas si simple (pas au programme de la L1 !). Et 2^n augmente très vite et cette méthode n'est pas applicable pour n pas trop petit.

Dans le cas de l'afficheur, on ne s'intéresse pas à toutes les combinaisons mais à quelques-unes (10 par rapport aux $2^7 = 128$ combinaisons possibles d'entrées). Cela entraîne le fait que le nombre d'inégalités est très petit par rapport au nombre de combinaisons possibles, ce qui complique les choses [3].

3. En L1, on résout des systèmes de n équations linéaires à n inconnues, le nombre d'équations est égal au nombre d'inconnues. Quand ce n'est pas le cas, la résolution du système est moins simple.

3 Perceptron à entrées réelles

On considère maintenant un perceptron dont les valeurs en entrées peuvent être n'importe quel réel. La sortie est binaire.

Il y a plusieurs manières de définir la fonction d'activation. Pour l'instant, la sortie est calculée comme pour le perceptron binaire plus haut.

Un tel perceptron permet de résoudre des problèmes de « classification supervisée ».

On suppose que l'on dispose d'un ensemble d'*exemples*, c'est-à-dire de couples (entrées du perceptron, sortie du perceptron). Par exemple pour le ET-logique rencontré plus haut, il y a 4 exemples :

— exemple 1 : (entrée $= (0, 0)$, sortie $= 0$)
— exemple 2 : (entrée $= (0, 1)$, sortie $= 0$)
— exemple 3 : (entrée $= (1, 0)$, sortie $= 0$)
— exemple 4 : (entrée $= (1, 1)$, sortie $= 1$)

3.1 Recherche des poids d'un perceptron

3.1.1 Intuition

Comment trouver les poids ?

On cherche des poids tels que la sortie du perceptron soit celle attendue en fonction des entrées.

On procède itérativement :

— **initialisation** : on commence avec des poids pris aléatoirement (en 2D si le perceptron a deux entrées, cela revient à prendre une droite au hasard dans le plan)
— **tant qu**'il y a des différences entre la sortie attendue et la sortie calculée par le perceptron :

1. **pour chaque exemple** :
   (a) on le place en entrée et on calcule la sortie du perceptron ;
   (b) **si** la sortie du perceptron n'est pas celle attendue, **alors** on modifie légérement les poids (cela revient à modifier légérement la position de la droite).

On peut dire qu'il existe une fonction des poids f ($f(p_0, p_1, p_2)$ en 2D) et que l'image de f est le nombre d'erreurs du perceptron, une erreur étant le fait que la sortie du perceptron pour une entrée donnée n'est pas la sortie attendue.

L'objectif est alors de minimiser (si possible annuler) le nombre d'erreurs, c'est-à-dire trouver un jeu de poids (p_0^*, p_1^*, p_2^*) (en 2D) pour lequel f est minimale. C'est ce que l'on appelle un « problème de minimisation de fonction ». Il existe plusieurs méthodes pour résoudre un tel problème. On va voir la méthode qui est utilisée dans le cadre des perceptrons, méthode qui est très générale et peut s'appliquer à la minimisation de tas de fonctions, pas seulement pour trouver les poids d'un perceptron.

3.1.2 Optimisation de fonction par descente de gradient

Intuition : on veut trouver le point où une fonction est minimale. On prend un x_0 initial quelconque (*cf.* Fig. 3.(b)). On calcule $f(x_0)$ et selon la pente de f en x_0, c'est-à-dire, selon le signe de $f'(x_0)$, on modifie légérément la valeur de $x_0 \to x_1$ (*cf.* Fig. 3.(b)). On itère jusqu'à ce que la pente soit nulle (*cf.* Fig. 3.(c)).

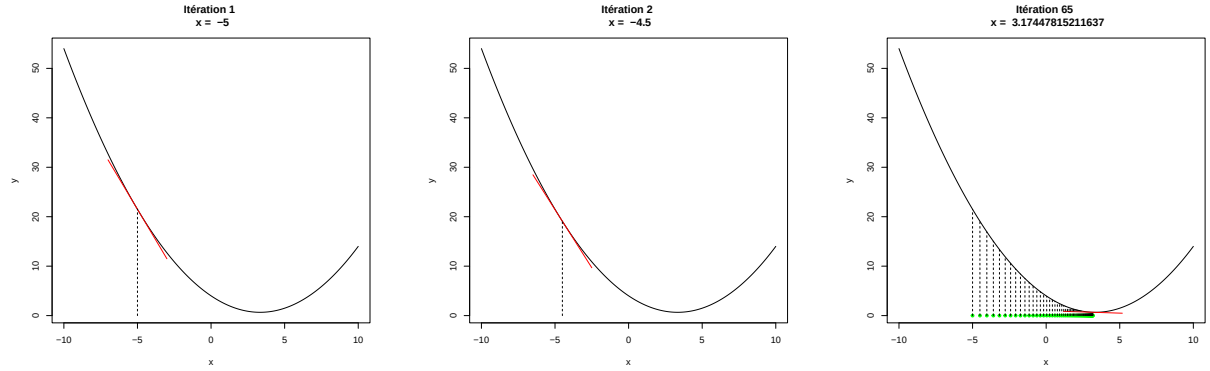


FIGURE 3 – Quelques étapes d'une descente de gradient stochastique pour trouver le minimum de cette fonction. (a) : x_0 choisi au hasard à partir duquel on commence la descente. (b) : x_1. (c) : la succession des x_k au fil des itérations de la descente de gradient stochastique. Sur chaque figure, le petit trait rouge indique la tangente à la courbe au point courant.

On peut appliquer l'idée à n'importe quelle fonction, quelle que soit la dimension de son domaine de définition. Il faut pouvoir calculer sa dérivée f' en tout point du domaine donc que f soit continue et dérivable sur son domaine de définition. Cela donne l'algorithme 1.

Algorithme 1 Minimisation d'une fonction réelle à une variable par descente de gradient stochastique.

Nécessite: un seuil d'arrêt ϵ
Nécessite: une valeur pour η
Nécessite: fonction à minimiser : f
Nécessite: dérivée de la fonction à mnimiser : f'
1: $k \leftarrow 0$
2: initialiser x_k (aléatoirement ou autrement)
3: **tant-que** $f'(x_k) > \epsilon$ **faire**
4: $\quad x_{k+1} \leftarrow x_k - \eta f'(x_k)$
5: $\quad k \leftarrow k + 1$
6: **fin tant-que**

Quelques remarques concernant la mise en pratique de cet algorithme :

— on arrête les itérations quand la pente est petite (voir ligne 3 de l'algorithme 2), car en pratique, elle ne sera jamais exactement nulle.

— La valeur de η doit diminuer au fil des itérations. D'une manière générale, savoir comment faire décroître η est loin d'être un sujet simple et pour notre brève présentation, nous nous contenterons de garder fixe la valeur de η.

— En gardant fixe la valeur de η, on ne peut pas atteindre le minimum, on ne fait que s'en approcher.

— Cette méthode de minimisation de fonctions n'est pas à utiliser pour des fonctions « habi-

tuelles » en mathématiques, en particulier pour des polynômes. On dispose de méthodes bien plus performantes pour celles-ci. Par contre, dans le cas du perceptron, c'est ce type de méthodes qui est utilisé, certes avec quelques raffinements dont la description dépasseraient l'esprit de ce cours d'introduction.

— η porte différents noms. Dans ce cours, on l'appelle le **taux d'apprentissage**.

3.1.3 Programmation

Implanter la DGS pour une fonction quelconque (= TP 3).

3.1.4 Optimisation de fonction générale par descente de gradient pour le perceptron

On définit quelques notations : on suppose disposer de N exemples. Un exemple i est un couple (x_i, y_i) où x_i est une donnée et y_i son étiquette. Une donnée est un vecteur de P nombres que l'on appelle les « attributs » de la donnée. On note $x_{i,j}$ la valeur de l'attribut j de la donnée i. C'est un nombre réel. Chacun des attributs d'une donnée est placé sur l'une des entrées du perceptron qui compte donc $P+1$ entrées : l'entrée e_j reçoit l'attribut numéro j. Quand une donnée x_i est placée en entrée du perceptron, on veut alors que la sortie du perceptron soit y_i. Dans cette partie du cours, on considère que la sortie du perceptron, donc les étiquettes y_i, peuvent prendre deux valeurs distinctes (perceptron à sortie binaire). Pour l'instant, on a considéré que ces deux valeurs sont 0 et 1, donc $y_i \in \{0, 1\}$.

La question est : comment trouver les $P+1$ poids tels que si l'on place la donnée x_i sur les entrées du perceptron, sa sortie soit y_i ?

Quand on place x_i en entrée du perceptron et que sa sortie n'est pas y_i, on dit que le perceptron commet une **erreur de prédiction**.

Pour trouver les poids, on procède par descente de gradient stochastique.

Comme on l'a dit plus haut, on cherche à minimiser le nombre d'erreurs commises par le perceptron. L'ensemble des exemples étant fixé, ce nombre d'erreurs est une fonction des poids. C'est cette fonction que l'on veut minimiser. À chaque ensemble de N exemples correspond un nombre d'erreurs de prédiction. L'idée est donc de modifier légérement les poids pour faire diminuer ce nombre d'erreurs.

Pour cela, la règle est la suivante : on suppose que l'on a placé x_i en entrée du perceptron. La sortie du perceptron est s. Si s est différent de y_i, alors :

1. on calcule $d = s - y_i$,
2. on modifie le biais $p_0 \leftarrow p_0 - \eta d$,
3. on modifie les P poids numérotés 1 à P comme suit : $p_j \leftarrow p_j - \eta d x_{i,j}, \forall j \in \{1, ..., P\}$.

Cette modification des poids doit être réalisée pour tous les exemples dont l'étiquette ne correspond pas à la sortie du perceptron.

Et on doit continuer d'effectuer ces corrections tant que la somme des (valeurs absolues des) corrections apportées à l'ensemble des N exemples demeure supérieure à un certain seuil. Ces corrections correspondent à la dérivée que nous avons rencontrée dans la descente de gradient stochastique (Sec. 3.1.2).

Rédigé sous la forme d'un algorithme, cela nous donne l'algorithme de descente de gradient pour calculer les poids d'un perceptron binaire. Cet algorithme a été découvert plusieurs fois et a été nommé

avec des noms différents par ses différents découvreurs. Ainsi, il se nomme également la règle Delta, ou Adaline, ou rège d'apprentissage du perceptron. Il est décrit précisément par l'algorithme 2.

Algorithme 2 La règle d'apprentissage du perceptron.

Nécessite: N exemples (x_i, y_i)
Nécessite: une valeur pour η
Nécessite: une valeur pour ϵ
initialiser les poids $p_j, j \in \{0, ...P\}$ (aléatoirement ou autrement)
corrections $\leftarrow 1$
tant-que corrections $> \epsilon$ **faire**
  corrections $\leftarrow 0$
  pour chaque exemple i **faire**
    calculer la sortie s du perceptron quand x_i est placé sur ses entrées.
    $d = s - y_i$
    si $d \neq 0$ **alors**
      $p_0 \leftarrow p_0 - \eta d$
      corrections $\leftarrow$ corrections $+ \eta|d|$
      pour chaque poids $p_j, j \in \{1, ..., P\}$ **faire**
        corrections $\leftarrow$ corrections $+\eta|dx_{i,j}|$
        $p_j \leftarrow p_j - \eta dx_{i,j}$
      fin pour
    fin si
  fin pour
fin tant-que

Comme on l'a vu plus haut, les poids d'un perceptron ont une interprétation géométrique. Dans le cas d'un perceptron à deux entrées, on peut visualiser un jeu de poids sous la forme d'une droite ; modifier légérement la valeur des poids entraîne une modification légère de la droite correspondante. Ainsi, géométriquement, le jeu de poids initialement aléatoire correspond à une droite prise au hasard dans le plan. Au fil de l'algorithme de descente de gradient stochastique, la droite se positionne peu à peu dans une configuration dans laquelle les données étiquetées 0 sont séparées des données étiquetées 1. Cela est illustré par la figure 4.

3.1.5 Programmation

Programmer la descente de gradient pour le calcul des poids d'un perceptron à partir d'un ensemble d'exemples (= TP 4).

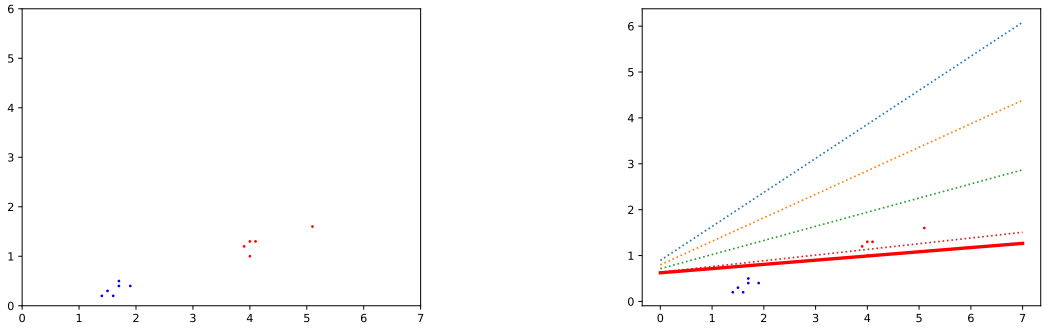


FIGURE 4 – Représentation graphique du déroulement du calcul des poids d'un perceptron à deux entrées durant une descente de gradient stochastique. La figure de gauche représente 11 données, la couleur indiquant la sortie attendue pour chacune d'entre-elles. À droite, l'initialisation aléatoire des poids du perceptron correspond à la droite la plus haute. On voit ensuite la succession de droites obtenues à la sortie de la boucle **pour**, ligne 16 dans l'algorithme 2. À la dernière itération, on obtient les poids correspondant à la droite rouge en trait gras : cette droite sépare les points bleus des points rouges et en utilisant les poids correspondant, la sortie du perceptron est celle attendue pour les 11 données.