Apprentissage Supervisé par un Perceptron

On considère un perceptron avec sa couche de N neurones d'entrée et son neurone de sortie (cf figure 1). Pour simplifier les notations, l'activité de chaque neurone pourra prendre deux valeurs, soit +1 (pour un neurone qui émet des potentiels d'action) soit -1 (pour un neurone qui n'émet pas de potentiels d'action). L'activité des neurones de la couche d'entrée (qu'on appellera un **patron d'activité**) est notée par une liste $\{\xi_j\}_{j=1,\ldots,N}$ (avec $\xi_j = \pm 1$). L'activité du neurone de sortie (qu'on appellera une **sortie**) est notée ν (avec $\nu = \pm 1$).

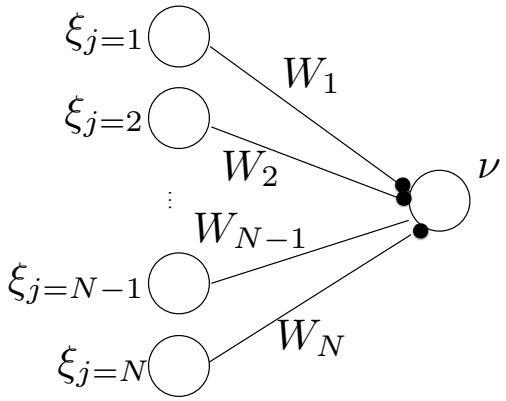

FIGURE 1 – Schéma d'un perceptron. L'activité d'un neurone i de la couche d'entrée est notée ξ_i, l'activité du neurone de sortie est notée ν. Le poids synaptique entre le neurone d'entrée i et le neurone de sortie est noté W_i

L'activité du neurone de sortie est déterminée par l'activité des neurones d'entrée et la valeurs des poids synaptiques, selon la règle

$$\begin{aligned} \nu &= 1 \quad \text{si :} \sum_{j=1}^{N} W_j \xi_j > 0 \\ &= -1 \quad \text{sinon} \end{aligned} \tag{1}$$

Apprentissage d'une association

On veut régler les poids synaptiques W_j pour apprendre une association entre un patron d'activité $\{\xi^1_j\}_{j=1,\ldots,N}$ et une sortie ν^1, c'est à dire, on veut avoir des W_j tels que :

$$\begin{aligned} \nu^1 &= 1 \quad \text{si :} \ \sum_{j=1}^{N} W_j \xi^1_j \ > \ 0 \\ &= -1 \quad \text{sinon} \end{aligned} \tag{2}$$

Pour ce faire, on propose de prendre les poids suivants :

$$W_j = \xi^1_j \nu^1 \tag{3}$$

Question 1 :
Expliquez pourquoi ce choix de poids synaptiques reflète une règle d'apprentissage de type Hebbien.

Question 2 :
Générez un code qui produit une patron d'activité $\{\xi^1_j\}_{j=1,\ldots,N}$ tel que :

$$\begin{aligned} \xi^1_j &= 1 \quad \text{avec probabilité} \ \frac{1}{2} \\ &= -1 \quad \text{avec probabilité} \ \frac{1}{2} \end{aligned} \tag{4}$$

et une sortie ν^1 telle que :

$$\begin{aligned} \nu^1 &= 1 \quad \text{avec probabilité} \ \frac{1}{2} \\ &= -1 \quad \text{avec probabilité} \ \frac{1}{2} \end{aligned} \tag{5}$$

On prendra $N = 10$ pour commencer.

Question 3 :
Générez un code qui, pour un patron d'activité $\{\xi^1_j\}_{j=1,\ldots,N}$ et une sortie ν^1, produit des poids synaptiques $W_{j=1,\ldots,N}$ selon la règle d'apprentissage (3).

Question 4 :
Générez une fonction/routine, qui prendra en entrée un patron d'activité $\{\xi_j^1\}_{j=1,...,N}$, une sortie ν^1 et des poids synaptiques $W_{j=1,...,N}$. Cette fonction/routine calculera l'activité du neurone de sortie selon la règle (1). La sortie de la fonction/routine sera un nombre, 1 si l'association est bien apprise, 0 si l'association est mal apprise. On vérifiera que pour la règle d'apprentissage (3), l'association entre $\{\xi_j^1\}_{j=1,...,N}$ et ν^1 est bien apprise.
Cette fonction/routine sera présentée sous la forme
classification = verifie_ association(patrons,sorties,W).

Apprentissage de plusieurs associations

On veut maintenant apprendre plusieurs associations entre P patrons d'activité $\{\xi_j^\mu\}_{j=1,...,N}$ et P sorties ν^μ avec $\mu = 1, ..., P$. Pour ce faire on propose l'algorithme suivant :

On démarre avec $\{W_j(t = 0) = 0, j = 1, ..., N\}$. On répétera les étapes suivantes autant de fois que nécessaire.

- (*) Pour $\mu = 1, ..., P$:
- Calculez la quantité $\Delta^\mu = \left(\sum_{j=1}^N W_j(t)\xi_j^\mu\right)\nu^\mu$.
- Si $\Delta^\mu > 0$, ne rien faire.
- Si $\Delta^\mu \leq 0$; effectuez une étape d'apprentissage :
  Pour chaque $j = 1, ..., N$,

$$W_j(t+1) = W_j(t) + \xi_j^\mu \, \nu^\mu \tag{6}$$

- Une fois passé sur $\mu = 1...P$, vérifiez si les poids permettent de classifier correctement les P patrons d'activité. Si oui, arrêtez l'algorithme, sinon retournez à (*)

Les questions qui suivent visent à vous permettre d'implémenter cet algorithme d'apprentissage.

Question 5 :
Expliquez pourquoi la troisième étape de l'algorithme revient à constater que l'association μ est bien apprise. Expliquez pourquoi la quatrième étape de l'algorithme revient à constater que l'association μ est mal apprise.

Question 6 :
Générez une fonction/routine qui, pour des valeurs de N et P données, renvoie P listes de N éléments tirés selon la règle (4) (on pourra renvoyer ceci sous la forme d'un tableau/matrice de taille $N \times P$), ainsi que P valeurs de sortie selon la règle (5) (on pourra renvoyer ceci sous la forme d'un tableau/matrice de taille $1 \times P$)
Cette fonction/routine sera présentée sous la forme
[patrons,sorties] = generation_patrons_sorties(N,P).

Question 7 :
Ici on veut écrire une fonction/routine qui permet de vérifier si des poids synaptiques permettent bien d'apprendre les P associations. Généralisez la fonction/routine écrite à la question 4 pour le cas de P associations. La fonction/routine aura donc pour sortie une liste de P éléments qui prennent des valeurs 1 ou 0 selon qu'une association est bien ou mal apprise.
Cette fonction/routine sera présentée sous la forme
classification = verifie_associations(patrons,sorties,W).

Question 8 :
Générez une fonction/routine qui, pour P patrons d'activité $\{\xi_j^\mu\}_{j=1,...,N}$ et P sorties ν^μ avec $\mu = 1, ..., P$, renvoie les poids synaptiques produits par l'algorithme d'apprentissage décrit ci-dessus. On procédera par étape en codant chacun des points de l'algorithme d'apprentissage. On pourra réutiliser la fonction/routine de la question précédente.
Cette fonction/routine sera présentée sous la forme
W = apprentissage_perceptron(patrons,sorties).

Question 9 :
Pour $N = 10$ et $P = 2$, générez des associations avec la fonction/routine de la question 6, et faites tourner la fonction/routine de la question 8 pour apprendre ces associations.

Capacité du perceptron

Dans cette partie on s'intéresse à combien d'associations il est possible d'apprendre avec un perceptron de N neurones. C'est à dire, quel est le P maximal pour lequel l'algorithme d'apprentissage converge.

Question 10 :
Si on demande au perceptron d'apprendre un trop grand nombre d'associations, l'algorithme d'apprentissage risque de ne pas converger. Pour éviter de rentrer dans une boucle infinie, modifiez la fonction/routine de la question 8 en ajoutant un nombre limite de fois où l'on passe par l'étape 1 de l'algorithme d'apprentissage. On prendra $P \times 1000$ pour cette limite. En plus de retourner un tableau de poids synaptiques, la fonction/routine retournera une variable **convergence** qui vaudra 1 si l'algorithme a convergé, 0 sinon. Cette fonction/routine sera une modification de la fonction/routine de la question 8, présentée sous la forme
[convergence,W] = apprentissage_perceptron(patrons,sorties).

Question 11 :
On définit $\alpha = \frac{P}{N}$. Pour $P = 10$ et $N = 10$ (c'est à dire $\alpha = 1$), on générera des associations et fera tourner l'algorithme d'apprentissage $n = 50$ fois. On gardera trace des 50 valeurs de la variable **convergence** qui sera retournée, puis on calculera la probabilité que l'algorithme converge, c'est à dire la probabilité qu'on ait réussi à apprendre les P associations.

Question 12 :
On répétera la question 11 avec toujours $N = 10$ et $P = 2; 5; 10; 15; 20; 25; 30$ (c'est à dire $\alpha = 0.2; 0.5; 1; 1.5; 2; 2.5; 3$). Tracez la probabilité de convergence en fonction de α.

Question 13 :
Répétez la question 12 pour $N = 20$ et $N = 30$ et les valeurs $\alpha = 0.2; 0.5; 1; 1.5; 2; 2.5; 3$.

Question 14 :
Discutez le nombre maximal d'associations qu'un perceptron peut apprendre.