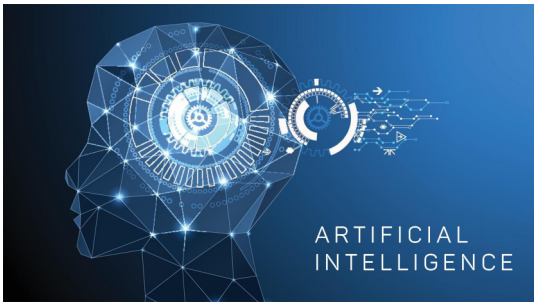
Chapitre 3 : Le perceptron



Pr. Mustapha BOURAHLA, Département d'Informatique, Université de M'Sila, Contact : mustapha.bourahla@univ-msila.dz

Mars 2021
v 1.0

Table des matières

Introduction

Présentation du perceptron et perceptron multicouches

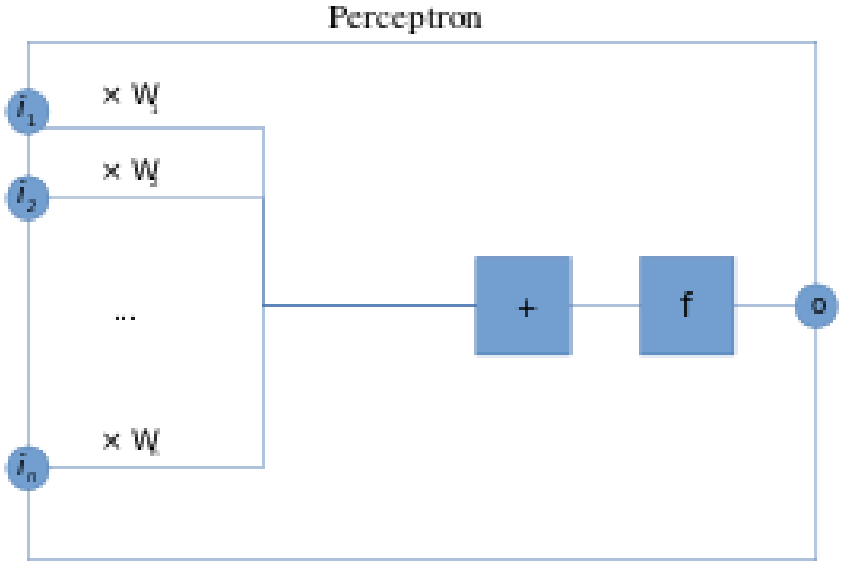
Perceptron

I

Définition

- Le perceptron est un algorithme d'apprentissage supervisé de classifieurs binaires (c'est-à-dire séparant deux classes).
- Il a été inventé en 1957 par Frank Rosenblatt au laboratoire d'aéronautique de l'université Cornell.
- Il s'agit d'un neurone formel muni d'une règle d'apprentissage qui permet de déterminer automatiquement les poids synaptiques de manière à séparer un problème d'apprentissage supervisé.
- Si le problème est linéairement séparable, un théorème assure que la règle du perceptron permet de trouver une séparatrice entre les deux classes.

- Le perceptron peut être vu comme le type de réseau de neurones le plus simple. C'est un classifieur linéaire.
- Ce type de réseau neuronal ne contient aucun cycle (il s'agit d'un réseau de neurones à propagation avant).
- Dans sa version simplifiée, le perceptron est mono-couche et n'a qu'une seule sortie (booléenne) à laquelle toutes les entrées (booléennes) sont connectées.
- Plus généralement, les entrées peuvent être des nombres réels.
- Un perceptron à n entrées $(x_1, \ldots, x_n)$ et à une seule sortie o est défini par la donnée de n poids (ou coefficients synaptiques) $(w_1, \ldots, w_n)$ et un biais (ou seuil) θ par:
  $o = f(z) = \begin{cases} 1 & \text{si} \quad \sum_{i=1}^{n} w_i x_i > \theta \\ 0 & \text{sinon} \end{cases}$
- La sortie o résulte alors de l'application de la fonction de Heaviside au potentiel post-synaptique $z = \sum_{i=1}^{n} w_i x_i - \theta$, avec:
  $\forall x \in \mathbb{R},\ H(x) = \begin{cases} 0 & \text{si} \quad x < 0 \\ 1 & \text{si} \quad x \geq 0. \end{cases}$
- Cette fonction non linéaire est appelée fonction d'activation.
- Une alternative couramment employée est $f = tanh()$, la tangente hyperbolique.



$$o = f\left(\sum_{k=1}^{n} i_k \cdot W_k\right)$$

Règle de Hebb

- La règle de Hebb, établie par Donald Hebb, est une règle d'apprentissage des réseaux de neurones artificiels dans le contexte de l'étude d'assemblées de neurones.
- Cette règle suggère que lorsque deux neurones sont excités conjointement, ils créent ou renforcent un lier les unissant.
- Dans le cas d'un neurone artificiel seul utilisant la fonction signe comme fonction d'activation cela signifi que : $W'_i = W_i + \alpha(Y.X_i)$ où W'_i représente le poids i corrigé et α représente le pas d'apprentissage.
- Cette règle n'est malheureusement pas applicable dans certains cas bien que la solution existe.

Règle d'apprentissage du perceptron (loi de Widrow-Hoff)

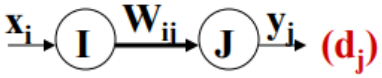
- Le perceptron de Frank Rosenblatt est très proche de la règle de Hebb, la grande différence étant qu'il tient compte de l'erreur observée en sortie.
- Cette fonction est recommandée lorsque la tangente hyperbolique (tanh) est utilisée comme fonction d'activation.
- $W'_i = W_i + \alpha(Y_t - Y)X_i$ W'_i = le poids i corrigé Y_t = sortie attendue Y = sortie observée α = le taux d'apprentissage X_i = l'entrée du poids i pour la sortie attendue Y_t W_i = le poids i actuel

Algorithme d'apprentissage du Perceptron

Apprentissage supervisé par correction d'erreur

Si la solution existe la règle converge

La règle de Widrow-Hoff:



$$W_{ij}(t+1)=W_{ij}(t)+k.(d_j-y_j).x_i$$

1- Initialisation des poids et du seuil Θ à des valeurs (petites) choisies au hasard.

2- Présentation d'une entrée $X = (x_1, \cdots, x_n)$ de la base d'apprentissage.

3- Calcul de la sortie obtenue $Y = (y_1, \cdots, y_m)$ pour cette entrée :

Pour chaque y_j Faire

$a_j = \Sigma_i(w_{ij}.x_{ij}) - \Theta$

$y_j = signe(a_j)$ (ici f est la fonction signe)

Fin Pour

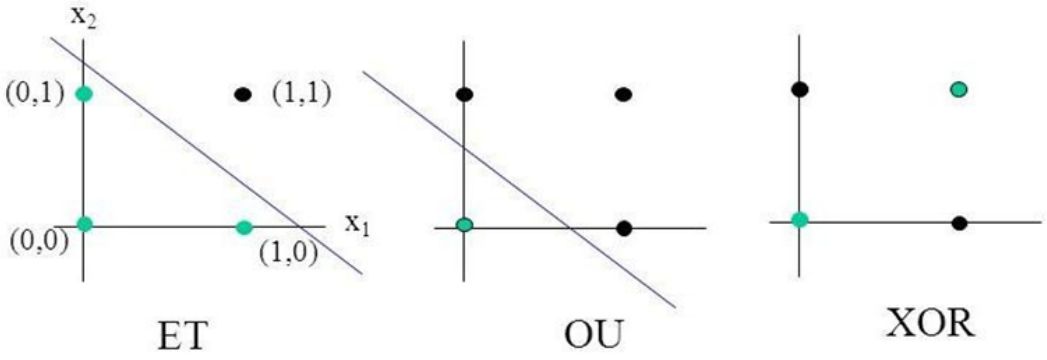
4- Si la sortie Y du Perceptron est différente de la sortie désirée D pour cet exemple d'entrée X alors modification des poids : $w_{ij} = w_{ij} + k.(d_j - y_j).x_i$

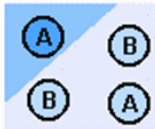
5- Tant que tous les exemples de la base d'apprentissage ne sont pas traités correctement (i.e. modification des poids), retour à l'étape 2.

Remarque

On peut construire des Perceptrons capables de réaliser les fonctions logiques : ET / OU

1. Mais le Perceptron est incapable de distinguer les patrons non séparables linéairement.
2. OU Exclusif (XOR)
3. Données non linéairement séparables !

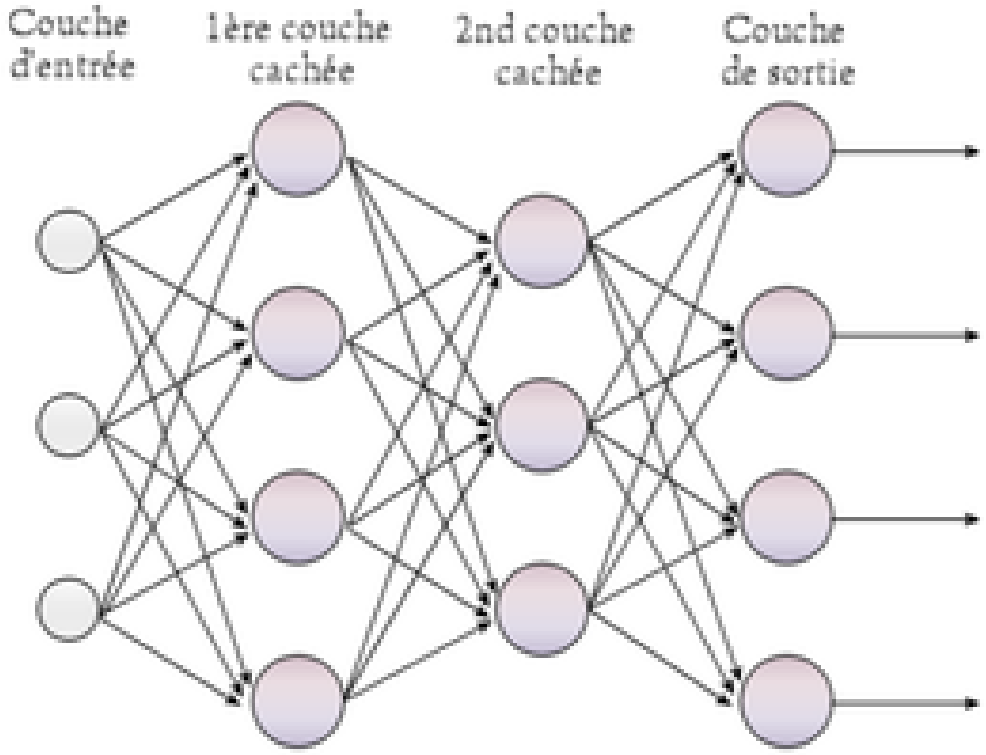


Structure	Type of Decision Regions	Exclusive-Or Problem	Classes with Meshed Regions	Most General Region Shapes
Single-layer 	Half plane bounded by hyperplane			
Two-layer 	Convex open or closed regions			
Three-layer 	Arbitary (Complexity limited by number of nodes)			

Perceptron multicouche

II

- Le perceptron multicouche (multilayer perceptron MLP) est un type de réseau neuronal artificiel organisé en plusieurs couches au sein desquelles une information circule de la couche d'entrée vers la couche de sortie uniquement ; il s'agit donc d'un réseau à propagation directe (feedforward).
- Chaque couche est constituée d'un nombre variable de neurones, les neurones de la dernière couche (dite « de sortie ») étant les sorties du système global.
- Le perceptron a été inventé en 1957 par Frank Rosenblatt au Cornell Aeronautical Laboratory. Dans cette première version le perceptron était alors mono-couche et n'avait qu'une seule sortie à laquelle toutes les entrées étaient connectées.



Perceptron multicouche à rétropropagation

- Les premiers réseaux de neurones n'étaient pas capables de résoudre des problèmes non linéaires ;
- cette limitation fut supprimée au travers de la rétropropagation du gradient de l'erreur dans les systèmes multicouches, proposé par Paul Werbos en 1974 et mis au point douze années plus tard, en 1986 par David Rumelhart.
- Dans le perceptron multicouche à rétropropagation, les neurones d'une couche sont reliés à la totalité des neurones des couches adjacentes.
- Ces liaisons sont soumises à un coefficient altérant l'effet de l'information sur le neurone de destination.

- Ainsi, le poids de chacune de ces liaisons est l'élément clef du fonctionnement du réseau : la mise en plac
  d'un Perceptron multicouche pour résoudre un problème passe donc par la détermination des meilleurs
  poids applicables à chacune des connexions inter-neuronales.
- Ici, cette détermination s'effectue au travers d'un algorithme de rétropropagation.

Apprentissage : Algorithme de rétropropagation

```
    Présentation d'un motif d'entraînement au réseau.
    Comparaison de la sortie du réseau avec la sortie ciblée.
    Calcul de l'erreur en sortie de chacun des neurones du réseau.
    Calcul, pour chacun des neurones, de la valeur de sortie qui aurait été correcte.
    Définition de l'augmentation ou de la diminution nécessaire pour obtenir cette valeur (erreur locale).
    Ajustement du poids de chaque connexion vers l'erreur locale la plus faible.
    Attribution d'un blâme à tous les neurones précédents.
    Recommencer à partir de l'étape 4, sur les neurones précédents en utilisant le blâme comme erreur.
```

Calcul de l'erreur :

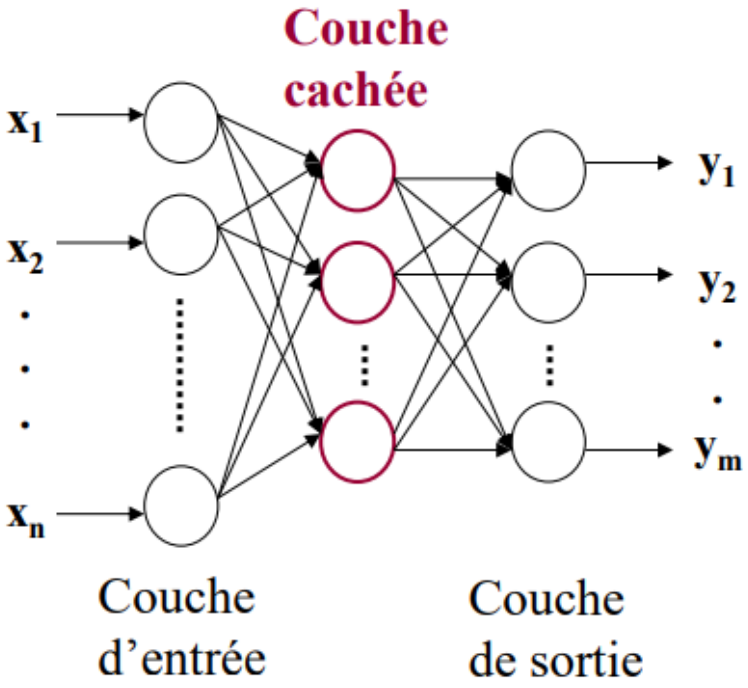
- En connaissant la valeur attendue e_i à la sortie d'un perceptron pour des entrées données, on peut calculer l'écart avec la prédiction grâce à une fonction objectif, le plus souvent l'erreur quadratique moyenne (abrégée MSE), telle que :
- $$\mathrm{MSE}(e_i, y_i) = \frac{1}{n}\sum_{i=0}^{n}(y_i - e_i)^2;$$
- Cette fonction n'est pas linéaire, et sa dérivée est plus grande si la prédiction est éloignée de la valeur attendue, permettant ainsi un apprentissage plus rapide.
- Au contraire, l'erreur moyenne absolue (MAE) a une dérivée constante, et donc un taux d'apprentissage qui ne varie pas :
- $$\mathrm{MAE}(e_i, y_i) = \frac{1}{n}\sum_{i=0}^{n}|y_i - e_i|;$$
- En minimisant ces fonctions objectif, les prédictions gagnent en précision.

La règle d'apprentissage d'un Perceptron $W_{ij} = W_{ij} + k.(d_j - y_j).x_i$ est non applicable sur un PMC

Problème : Quelles sont les sorties désirées de la couche cachée ?

Solution : rétro-propagation de l'erreur

- D'une façon distribuée
- Pondérée par les poids initiaux
- E est l'erreur commise par le réseau elle représente la différence$(D - Y)$
  Minimisation de l'erreur quadratique donnée par : $E = 1/2 \times \sum_i (d_i - y_i)^2$
  Descente du gradient : $\Delta w_{ij} = w_{ij}(t+1) - w_{ij}(t) = -\eta\frac{\partial E}{\partial w_{ij}}$



Algorithme d'apprentissage

- Propagation en avant (Forward) : $y_i = F(a_i) = F(\sum_j w_{ij}.x_j)$ avec $F(x) = \frac{1}{1 - e^{-x}}$
- Rétro-propagation (Backword) : $w_{ij}(t+1) = w_{ij}(t) - \eta.\delta_i.\frac{\partial a_i}{\partial w_{ij}}$ avec $\delta_i = \frac{\partial E}{\partial a_i}$
- Pour les poids entre la couche de sortie et celle qui la précède : $\delta_i = F'(a_i).(y_i - d_i)$
- Pour les connexions entre les autres couches : $\delta_j = F'(a_j).\sum_i \delta_i.w_{ij}$

Mise en œuvre

- Nombre de neurones dans la couche d'entrée correspond à la dimension de l'espace d'entrée
- Nombre de neurones dans la couche de sortie correspond aux nombre de classes des données à classer
- Nombre de neurones dans la couche cachée est déterminé d'une manière empirique : on choisit le nombre qui donne les meilleurs résultats
- Nombre de couches cachées est déterminé empiriquement

Exercices sur le troisième chapitre

III

Exercice : Perceptron

[solution n°1 p.15*]*

Donner la catégorie de

perceptron multi-couche

perceptron mono-couche

séparation linéaire	séparation non-linéaire

Exercice :

IV

Question

[solution n°2 p.15*]*

Dérouler l'algorithme d'apprentissage pour deux itérations sur un perceptron mono-couche avec les données suivantes :

- trois entrées $x_1 = 1, x_2 = 0, x_3 = 2 \implies X = \begin{bmatrix} 1 & 0 & 2 \end{bmatrix}$
- deux sorties $y_1 = 2, y_2 = 0 \implies Y = \begin{bmatrix} 2 & 0 \end{bmatrix}$
- les poids : $w_{11} = 0.2, w_{12} = 0.4, w_{21} = 0.3, w_{22} = 0.6, w_{31} = 0.5, w_{32} = 0.4 \implies W = \begin{bmatrix} 0.2 & 0.4 \\ 0.3 & 0.6 \\ 0.5 & 0.4 \end{bmatrix}$
- le seuil $\Theta_1 = 0.5, \Theta_2 = 0.5 \implies B = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}$
- la fonction d'activation $signe$
- le taux d'apprentissage $k = 0.7$

Exercice :

V

Question

[solution n°3 p.16*]*

Dérouler l'algorithme d'apprentissage pour une seule itération sur un perceptron multi-couche avec les données suivantes :

- Deux entrées $x_1 = 1, x_2 = 0 \implies X = \begin{bmatrix} 1 & 0 \end{bmatrix}$
- Une sortie $y = 1 \implies Y = \begin{bmatrix} 1 \end{bmatrix}$
- les poids de la première couche FC(2,2): $w_{11} = 0.2, w_{12} = 0.4, w_{21} = 0.3, w_{22} = 0.6 \implies W = \begin{bmatrix} 0.2 & 0.4 \\ 0.3 & 0.6 \end{bmatrix}$
- les poids de la deuxième couche FC(2,1): $w_{11} = 0.3, w_{21} = 0.5 \implies W = \begin{bmatrix} 0.3 \\ 0.5 \end{bmatrix}$
- les seuils de la première couche : $\Theta_{11} = 0.5, \Theta_{12} = 0.3 \implies B = \begin{bmatrix} 0.5 & 0.3 \end{bmatrix}$
- les seuils de la deuxième couche : $\Theta_{21} = 0.6 \implies B = \begin{bmatrix} 0.6 \end{bmatrix}$
- la fonction d'activation est sigmoïde
- La fonction de perte est MSE
- le taux d'apprentissage $k = 0.7$

Conclusion

Ce troisième chapitre a présenté un aperçu sur le perceptron

Solutions des exercices

> **Solution** n°1

Exercice p. 11

Donner la catégorie de

séparation linéaire	**séparation non-linéaire**
perceptron mono-couche	perceptron multi-couche

> **Solution** n°2

Exercice p. 12

1- Initialisation des poids et du seuil Θ à des valeurs (petites) choisies au hasard.

2- Présentation d'une entrée $X = (x_1, \cdots, x_n)$ de la base d'apprentissage.

3- Calcul de la sortie obtenue $Y = (y_1, \cdots, y_m)$ pour cette entrée :

Pour chaque y_j Faire

$a_j = \Sigma_i(w_{ij}.x_i) - \Theta_j$

$y_j = signe(a_j)$ (ici f est la fonction signe)

Fin Pour

4- Si la sortie Y du Perceptron est différente de la sortie désirée D pour cet exemple d'entrée X alors modification des poids : $w_{ij} = w_{ij} + k.(d_j - y_j).x_i$

5- Tant que tous les exemples de la base d'apprentissage ne sont pas traités correctement (i.e. modification des poids), retour à l'étape 2.

$a_j = \Sigma_i(w_{ij}.x_i) - \Theta_j$

$y_j = signe(a_j)$ (ici f est la fonction signe)

Pour la sortie 1 :

$a_1 = w_{11}.x_1 + w_{21}.x_2 + w_{31}.x_3) - 0.5 = 0.2 \times 1 + 0.3 \times 0 + 0.5 \times 2 - 0.5 = 0.2 + 0 + 1 - 0.5 = 0.7$

$y_1 = signe(a_1) = +1$

Pour la sortie 2 :

$a_2 = w_{12}.x_1 + w_{22}.x_2 + w_{32}.x_3) - 0.5 = 0.4 \times 1 + 0.6 \times 0 + 0.4 \times 2 - 0.5 = 0.4 + 0 + 0.8 - 0.5 = 0.7$

$y_2 = signe(a_2) = +1$

Mise à jour des poids :

$$w_{ij} = w_{ij} + k.(d_j - y_j).x_i$$

$$w_{11} = w_{11} + k.(d_1 - y_1).x_1 = 0.2 + 0.7 \times (2 - 1) \times 1 = 0.2 + 0.7 = 0.9$$

$$w_{12} = w_{12} + k.(d_2 - y_2).x_1 = 0.4 + 0.7 \times (0 - 1) \times 1 = 0.4 - 0.7 = -0.3$$

$$w_{21} = w_{21} + k.(d_1 - y_1).x_2 = 0.3 + 0.7 \times (2 - 1) \times 0 = 0.3 + 0 = 0.3$$

$$w_{22} = w_{22} + k.(d_2 - y_2).x_2 = 0.6 + 0.7 \times (0 - 1) \times 0 = 0.6 + 0 = 0.6$$

$$w_{31} = w_{31} + k.(d_1 - y_1).x_3 = 0.5 + 0.7 \times (2 - 1) \times 2 = 0.5 + 1.4 = 1.9$$

$$w_{32} = w_{32} + k.(d_2 - y_2).x_3 = 0.4 + 0.7 \times (0 - 1) \times 2 = 0.4 - 1.4 = -1$$

En utilisant les matrices :

$$Y = X \times W - B$$

$$Y = signe(\begin{bmatrix}1 & 0 & 2\end{bmatrix} \times \begin{bmatrix}0.2 & 0.4\\0.3 & 0.6\\0.5 & 0.4\end{bmatrix} - \begin{bmatrix}0.5 & 0.5\end{bmatrix}) = signe(\begin{bmatrix}0.7 & 0.7\end{bmatrix}) = \begin{bmatrix}1 & 1\end{bmatrix}$$

$$W = W + k \times X^T \times (D - Y) = \begin{bmatrix}0.2 & 0.4\\0.3 & 0.6\\0.5 & 0.4\end{bmatrix} + 0.7 \times \begin{bmatrix}1\\0\\2\end{bmatrix} \times (\begin{bmatrix}2 & 0\end{bmatrix} - \begin{bmatrix}1 & 1\end{bmatrix}) = \begin{bmatrix}0.2 & 0.4\\0.3 & 0.6\\0.5 & 0.4\end{bmatrix} +$$

$$0.7 \times \begin{bmatrix}1\\0\\2\end{bmatrix} \times \begin{bmatrix}1 & -1\end{bmatrix} = \begin{bmatrix}0.2 & 0.4\\0.3 & 0.6\\0.5 & 0.4\end{bmatrix} + 0.7 \times \begin{bmatrix}1 & -1\\0 & 0\\2 & -2\end{bmatrix} = \begin{bmatrix}0.9 & -0.3\\0.3 & 0.6\\1.9 & -1\end{bmatrix}$$

> **Solution** n°3

Exercice p. 13

La propagation en avant :

$$Y = X \times W + B$$

$$E = \frac{1}{n} \sum_{i=1}^{n} (y_i^* - y_i)^2$$

La couche numéro 1 :

$$\begin{bmatrix}y_1 & y_2\end{bmatrix} = \begin{bmatrix}1 & 0\end{bmatrix} \times \begin{bmatrix}0.2 & 0.4\\0.3 & 0.6\end{bmatrix} + \begin{bmatrix}0.5 & 0.3\end{bmatrix} = \begin{bmatrix}0.7 & 0.7\end{bmatrix}$$

$$\begin{bmatrix}y_1 & y_2\end{bmatrix} = tanh(\begin{bmatrix}y_1 & y_2\end{bmatrix}) = tanh(\begin{bmatrix}0.7 & 0.7\end{bmatrix}) = \begin{bmatrix}0.6 & 0.6\end{bmatrix}$$

La couche numéro 2 :

$$\begin{bmatrix}y\end{bmatrix} = \begin{bmatrix}0.6 & 0.6\end{bmatrix} \times \begin{bmatrix}0.3\\0.5\end{bmatrix} + \begin{bmatrix}0.6\end{bmatrix} = \begin{bmatrix}1.08\end{bmatrix}$$

$$\begin{bmatrix}y\end{bmatrix} = tanh(\begin{bmatrix}y\end{bmatrix}) = tanh(\begin{bmatrix}1.08\end{bmatrix}) = \begin{bmatrix}0.8\end{bmatrix}$$

Calcul de l'erreur E du réseau (MSE) :

$$E = \frac{1}{n} \sum_{i=1}^{n} (y_i^* - y_i)^2 = (1 - 0.8)^2 = 0.04$$

La propagation en arrière (la rétro-propagation) :

Pour chaque couche de la dernière vers la première faire :

calcul des erreurs de sortie (dérivée de MSE) : $\frac{\partial E}{\partial Y} = \frac{2}{n}(Y - Y^*)$

calcul des erreurs des biais : $\frac{\partial E}{\partial B} = \frac{\partial E}{\partial Y}$ **et mise à jour des biais :** $B = B - k \times \frac{\partial E}{\partial B}$

calcul des erreurs des poids : $\frac{\partial E}{\partial W} = X^t \times \frac{\partial E}{\partial Y}$ **et mise à jour des poids :** $W = W - k \times \frac{\partial E}{\partial W}$

calcul des erreurs des entrées : $\frac{\partial E}{\partial X} = \frac{\partial E}{\partial Y} \times W^t$ **qui deviennent les erreurs de sorties de la couche précédente**

La couche numéro 2 :

$\frac{\partial E}{\partial B} = \frac{\partial E}{\partial Y} = \frac{2}{n}(Y - Y^*) = \begin{bmatrix} -0.4 \end{bmatrix} \Longrightarrow B = B - k \times \frac{\partial E}{\partial B} = \begin{bmatrix} 0.6 \end{bmatrix} - 0.7 \times \begin{bmatrix} -0.4 \end{bmatrix} = \begin{bmatrix} 0.88 \end{bmatrix}$

$\frac{\partial E}{\partial W} = X^t \times \frac{\partial E}{\partial Y} = \begin{bmatrix} 0.6 \\ 0.6 \end{bmatrix} \times \begin{bmatrix} -0.4 \end{bmatrix} = \begin{bmatrix} -0.24 \\ -0.24 \end{bmatrix} \Longrightarrow W = W - k \times \frac{\partial E}{\partial W} = \begin{bmatrix} 0.3 \\ 0.5 \end{bmatrix} - 0.7 \times \begin{bmatrix} -0.24 \\ -0.24 \end{bmatrix} = \begin{bmatrix} 0.47 \\ 0.67 \end{bmatrix}$

$\frac{\partial E}{\partial X} = \frac{\partial E}{\partial Y} \times W^t = \begin{bmatrix} -0.4 \end{bmatrix} \times \begin{bmatrix} 0.3 & 0.5 \end{bmatrix} = \begin{bmatrix} -0.6 & -1 \end{bmatrix} \Longrightarrow \frac{\partial E}{\partial Y} = \begin{bmatrix} -0.12 & -0.2 \end{bmatrix}$ **pour la couche précédente**

La couche numéro 1 :

$\frac{\partial E}{\partial B} = \frac{\partial E}{\partial Y} = \begin{bmatrix} -0.12 & -0.2 \end{bmatrix} \Longrightarrow B = B - k \times \frac{\partial E}{\partial Y} = \begin{bmatrix} 0.5 & 0.3 \end{bmatrix} - 0.7 \times \begin{bmatrix} -0.12 & -0.2 \end{bmatrix} = \begin{bmatrix} 0.58 & 0.44 \end{bmatrix}$

$\frac{\partial E}{\partial W} = X^t \times \frac{\partial E}{\partial Y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \times \begin{bmatrix} -0.12 & -0.2 \end{bmatrix} = \begin{bmatrix} -0.12 & -0.2 \\ 0 & 0 \end{bmatrix} \Longrightarrow W = W - k \times \frac{\partial E}{\partial W} = \begin{bmatrix} 0.2 & 0.4 \\ 0.3 & 0.6 \end{bmatrix} - 0.7 \times \begin{bmatrix} -0.12 & -0.2 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.28 & 0.54 \\ 0.3 & 0.6 \end{bmatrix}$

$\frac{\partial E}{\partial X} = \frac{\partial E}{\partial Y} \times W^t = \begin{bmatrix} -0.12 & -0.2 \end{bmatrix} \times \begin{bmatrix} 0.2 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} = \begin{bmatrix} -0.104 & -0.15 \end{bmatrix}$