| | |
|---|---|
| EISTI Ecole Internationale des Sciences du Traitement de l'Information | **ING2-MI : GENIE MATHEMATIQUE**<br>**DATA MINING : RESEAUX DE NEURONES** |
| Rédigé par : Astrid Jourdan | A l'intention de : Elèves ING2-MI |
| Durée : 5h | |

# Credit Scoring

Dans une banque, un service s'occupe d'accorder (ou pas) des prêts à des particuliers. Pour tous les prêts déjà accordés, elle dispose des informations suivantes : salaire, métier (fonction et secteur d'activité), nombre d'enfants, situation matrimoniale, …. Elle désire mettre en place un outil afin de prévoir à l'avance si un futur emprunteur posera ou pas des problèmes si on lui accorde un prêt.

- Identifier la tâche de Data Mining permettant de résoudre le problème.
- Quelles sont les informations nécessaires pour alimenter l'outil ?

# Exercice 1 : OR *vs* XOR

1) Considérons le problème du « OR » : les points (1,0), (1,1), (0,1) sont positifs et le point (0,0) est négatif. Soit le réseau de neurones,

$$\hat{y} = f(x_1 + x_2 - 0.5)$$

avec comme activation la fonction de Heaviside, $f(x) = 1$ si $x \geq 0$ et 0 sinon.

(a) Dessiner le réseau de neurones.

(b) Vérifier qu'il correspond bien au OR.

(c) Faire une représentation graphique des points de la base d'apprentissage et de la frontière obtenue.

(d) La solution est-elle unique ? Peut-on utiliser une autre fonction d'activation ?

2) Considérons le problème du « XOR » (ou exclusif) : les points (1,0) et (0,1) sont positifs et les points (0,0) et (1,1) sont négatifs.

(a) Faire une représentation graphique des points de la base d'apprentissage. Pensez-vous qu'un réseau sans couche cachée soit suffisant ici ?

Considérons maintenant un réseau avec 2 neurones cachés. Notons ($x_1$,$x_2$) la couche d'entrée ($x_3$,$x_4$) la couche cachée et $\hat{y}$ =$x_5$ la couche de sortie. On choisit les poids suivants : $w_{13}$=1, $w_{14}$=-1, $w_{23}$=-1, $w_{24}$=1, $w'_{35}$=-1, $w'_{45}$=-1, un biais de poids 1 entre la couche cachée et la couche de sortie, et toujours la fonction de Heaviside.

(b) Dessiner le réseau de neurones.

(c) Vérifier qu'il correspond bien au XOR.

# Exercice 2

L'objectif est de comprendre l'algorithme de rétro-propagation du gradient sur un exemple simple en dimension 2. Il s'agit d'ajuster un réseau à un seul neurone,

$$\hat{y} = f(w_1 x_1 + w_2 x_2 + w_0)$$

avec comme activation la fonction de Heaviside, $f(x) = 1$ si $x \geq 0$ et 0 sinon.

La base d'apprentissage est réduite aux deux points,

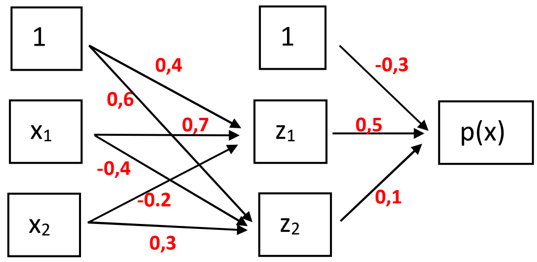
| | $x_1$ | $x_2$ | y |
|---|---|---|---|
| *a* | 1 | 2 | 1 |
| *b* | 2 | 1 | 0 |

1) Expliquez que le séparateur est une droite. Supposons que le biais est nul. Qu'est-ce que cela signifie ? Que va optimiser l'algorithme ?

2) Ecrire la fonction de mise à jour des poids pour un taux d'apprentissage $\alpha$.
3) On initialise les poids $w_1=w_2=1$. Quelle est l'initialisation de la pente de la droite ?
4) Exprimez la pente de la droite en fonction de $\alpha$ à l'issue de la 1ère itération. Quelle valeur de $\alpha$ faudrait-il poser pour que l'algorithme s'arrête à la première itération ?
5) On pose $\alpha$
6) =0,2. Déroulez l'algorithme jusqu'à la quatrième itération. Tracez les droites séparatrices au fur et à mesure de l'algorithme. Que se passe-t-il à la quatrième itération ?

## Exercice 3

Soit le réseau de neurones suivant pour modéliser une sortie binaire, $p(\boldsymbol{x})=P(Y=1|\boldsymbol{x})$, où $\boldsymbol{x}=(x_1,x_2)$




On suppose une fonction d'activation logistique inverse entre les couches.

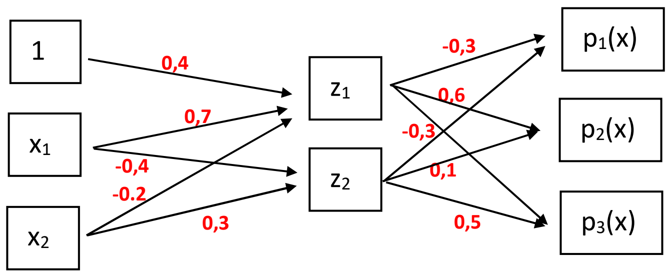
1) Explicitez les deux neurones de la couche cachée en fonction de $x_1$, $x_2$ et le biais.
2) Explicitez $p(\boldsymbol{x})$ en fonction des deux neurones de la couche cachée et du biais.
3) On considère l'exemple $\boldsymbol{x}=(0,1)$. Calculez la sortie du réseau.

<u>Questions supplémentaires facultatives</u>

4) On suppose que la variable cible de cet exemple est y=1. Calculez le delta de la couche de sortie
5) Calculez les delta de la couche cachée
6) Ajustez les poids en prenant un taux d'apprentissage $\alpha$=0,25

## Exercice 4

Soit le réseau de neurones suivant pour modéliser une sortie ayant trois classes, $Y\in\{1,2,3\}$. On note, $p_k(\boldsymbol{x})=P(Y=k|\boldsymbol{x})$, où $\boldsymbol{x}=(x_1,x_2)$ et k=1,2,3.




On suppose une fonction d'activation tangente hyperbolique entre les couches d'entrée et cachée.

1) Explicitez les deux neurones de la couche cachée en fonction de $x_1$, $x_2$ et le biais.
2) Quelle est la fonction d'activation entre la couche cachée et la couche de sortie ? Explicitez $p_k(\boldsymbol{x})$ en fonction des deux neurones de la couche cachée.

3) On considère l'exemple $\boldsymbol{x}$=(0,1). Calculez la sortie du réseau.

## Exercice 5

Dans cet exercice il s'agit de mettre en œuvre un réseau de neurones avec la fonction

`NN=nnet(Y~X1+X2+…,data=datatrain,size=2,decay=0.1,…)`

du package `nnet` de R. L'affichage du réseau de neurones se fait à l'aide de la fonction

`plotnet(NN)`

du package `NeuralNetTools`.

L'optimisation des hyperparamètres du réseau se fait avec la fonction

`tune.nnet(Y~X1+X2+…,data=datatrain,size=2:10,decay=c(0,0.1,1,2,3),maxit=100…)`

du package `e1071`

1) A partir de l'aide sur la fonction `nnet`, répondez aux questions suivantes.
   a) Comment sont initialisés les poids par défaut. Qu'est-ce que cela implique pour les variables d'entrée ? Préparez le jeu de données « iris » en conséquence.
   b) Construisez un réseau avec 5 neurones pour les données « iris » (on utilisera toute la base pour apprendre le réseau). Combien y-a-t-il de poids à ajuster dans ce réseau (combien sur la première couche et combien sur la deuxième couche)? A quoi correspondent les informations affichées ? Est-ce que l'algorithme de rétro-propagation a convergé ? Quel autre indicateur peut-on récupérer pour savoir si l'algorithme a convergé ?
   c) Quels sont les critères d'arrêt possibles pour stopper l'algorithme de rétro-propagation ? Faites varier ces critères de façon à ce que l'algorithme converge.
   d) Quelle est la fonction d'activation de la couche de sortie ? de la couche cachée ? Quelle est la fonction de coût ?
2) Construisez un réseau avec deux neurones et utilisez la fonction `plotnet` pour afficher le réseau de neurones. Déterminez le poids de chaque branche.
3) Etudiez les valeurs de la fonction coût en fonction des valeurs du paramètre de régularisation decay$\in$ [0,0.1,0.2,0.5].
4) Utiliser la fonction `tune.nnet` pour trouver les hyperparametres optimaux.
5) Utilisez la fonction `predict` pour prédire la probabilité des chacune des classes des exemples de la base d'apprentissage. Calculez la matrice de confusion.

## Exercice 6

L'objectif de cet exercice est d'illustrer les frontières de séparation créées par le réseau de neurones.

1) Simulez un jeu de données de 2000 exemples tel que

   $X_1$~U[-0.5,0.5]

   $X_2$~U[0,1]

   Y=1 si $0.1X_2>X_1^2$ et Y=0 sinon

   Faites une représentation graphique du nuage de points avec les classes. Est-ce qu'il est linéairement séparable ?
2) Ajustez un réseau avec deux neurones et avec 1000 itérations maxi.
3) La frontière séparatrice pour un neurone $\Sigma$ est la droite d'équation : $w_0+w_1x_1+w_2x_2=0$. Sur le nuage de points, ajoutez les droites engendrées par les deux neurones.
4) On considère le nouveau repère défini par les deux neurones

$$Z_1 = \frac{1}{1+e^{-\Sigma_1}} \quad \text{et} \quad Z_2 = \frac{1}{1+e^{-\Sigma_2}}$$

Tracez le nuage de points dans ce nouveau repère. Est-il linéairement séparable ?

5) Dans ce nouveau repère, la frontière est définie par la droite d'équation : $w'_0+w'_1z_1+w'_2z_2=0$. Ajoutez cette droite sur le nouveau nuage de points.

## Exercice 7

Nous allons maintenant utiliser la fonction `NeuralNet` du package du même nom. Avec cette fonction, on peut ajuster un réseau avec plus d'une couche cachée, faire plusieurs initialisations des poids et la fonction retourne le meilleur résultat, et choisir son algorithme d'optimisation des poids.

1) A partir de la fonction d'aide de la fonction, déterminer les éléments qui permettent de
   - gérer la convergence de l'algorithme.
   - gérer l'initialisation des poids
   - choisir la fonction de coût
2) Ajuster un réseau de neurones avec 2 couches cachées, 3 neurones sur la $1^{ère}$ couche et 2 neurones sur la $2^{ème}$, avec 5 initialisations aléatoires des poids.
3) A-t-on un meilleur résultat si on change la fonction de coût ? les fonctions d'activation ? le learning rate ? l'algorithme d'optimisation ?....

## Question facultative

Dans l'algorithme de rétro-propagation, déterminez les $\delta_k$ et $\delta_j$ lorsque la fonction coût est l'entropie croisée.