

Ce TP est une introduction aux réseaux de neurones. A l'aide de la fonction `nnet` de la librairie `nnet`, nous simulerons un réseau de neurones pour étudier son comportement en faisant varier les paramètres propres à cette méthode.

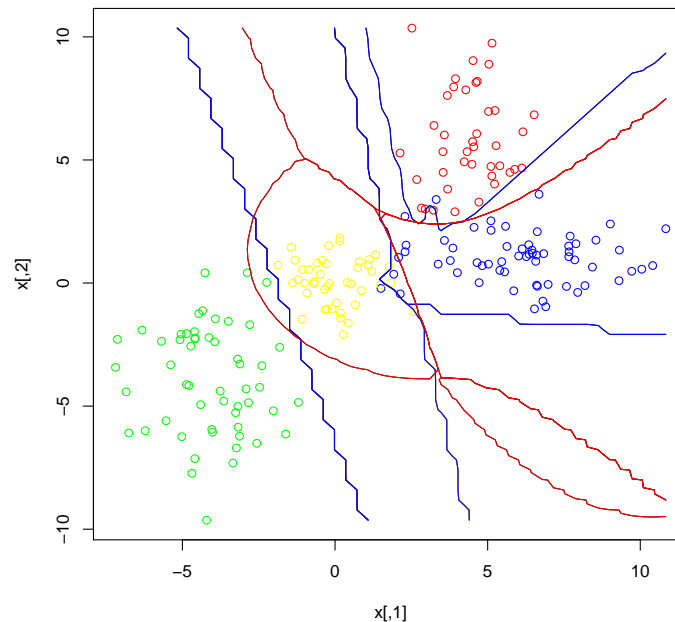
Dans la dernière partie, nous chercherons quelles valeurs de *size* et de *decay* donnent les meilleurs résultats pour des données crabes.

Enfin, nous conclurons ce TP en confrontant les résultats des réseaux de neurones avec l'analyse discriminante quadratique et l'analyse discriminante linéaire.

Question 1.

1) Frontières de décision

Ci-dessous, les frontières de décision obtenues avec un réseau de neurones (bleu) et les frontières de Bayes (rouge) pour le mélange de 4 vecteurs Gaussien.



2) Estimation des poids

Les poids de notre réseaux de neurones sont les suivants :

$$P = \begin{pmatrix} [1] & 28.878823 & 8.3440744 & 53.3825493 & -106.5337070 & 69.4008001 \\ [6] & -33.6124724 & 102.6807359 & -58.8415857 & 56.5035544 & -2.1808703 \\ [11] & 1.2888301 & 0.2520767 & -20.8067781 & -11.6369876 & -3.5229020 \\ [16] & -42.1579477 & 28.9905417 & 23.1741888 & 9.6386434 & 22.7751030 \\ [21] & -11.1007213 & 11.6648872 & -0.4817475 & -12.1847979 & -15.2710175 \\ [26] & -9.8486925 & 74.7604493 & -101.5021778 & -26.6435637 & 12.5338628 \\ [31] & 36.5421363 & 127.6647811 & -6.8851989 & 130.7274907 & -1.9219409 \\ [36] & -23.6756992 & -30.3509089 & -141.3414162 & -55.4818626 & \end{pmatrix}$$

On remarque qu'on a 39 poids. On retrouve ce nombre en faisant: (2 neurones d'entrée + 1 biais) * (5 neurones cachées) + (5 neurones cachées + 1 biais) * 4 neurones de sortie = $3 * 5 + 6 * 4 = 39$ poids.

3-4) Même procédure mais résultats différents

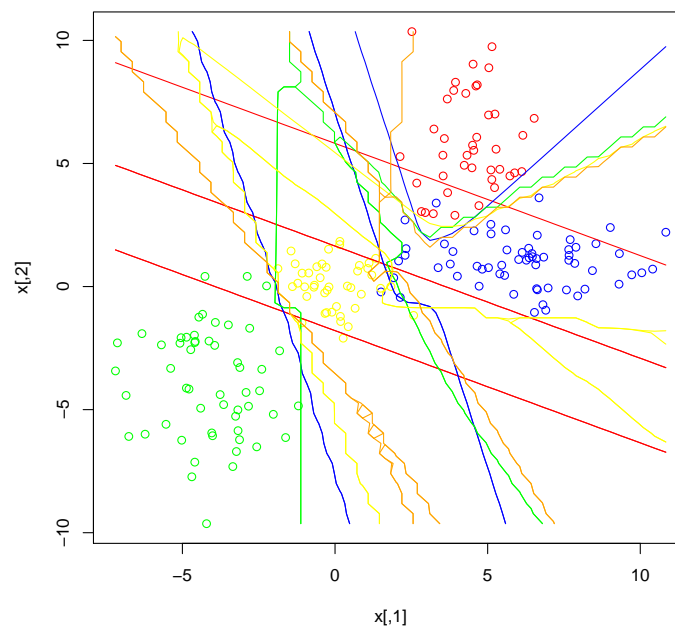
Après avoir lancé plusieurs fois la procédure nnet, on s'aperçoit que l'on n'obtient pas les mêmes résultats à chaque fois pour l'estimation des poids et des frontières de décision.

La raison: A chaque création d'un réseau de neurones, les poids sont initialisés aléatoirement. Par la suite, lors de l'apprentissage, les poids ne convergent pas forcément vers le même minima local. On a donc des estimations des poids et des frontières de décision différentes en fonction de l'initialisation.

Question 2.

1-2) Variation du paramètre *size*

Ci-dessous, l'aperçu graphique pour *size* variant entre 1 et 5. rouge: *size* = 1, bleu: *size* = 2, vert: *size* = 3, jaune: *size* = 4, orange: *size* = 5. Bien sur, il y a de plus en plus de poids, mais surtout la forme des frontières est de plus en plus complexe et le nombre de points mal classés diminue lorsque *size* augmente. Néanmoins, le nombre de points mal classés ne bouge plus trop à partir de *size* = 3.



En dessus de *size* = 5, la complexité ne semble plus trop bouger et le nombre de points mal classés semble même augmenter.

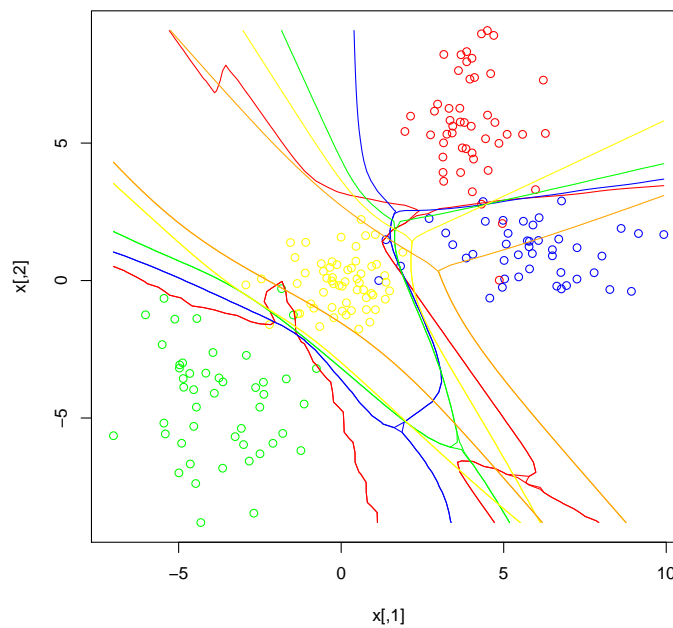
3-4) Variabilité du modèle

En relançant la procédure sur un nouveau jeu de données en faisant varier *size* de 1 à 10, on s'aperçoit que pour *size* = 1, le modèle est le même et plus *size* augmente, plus le modèle est propre aux données et donc sujet à des variations.

Question 3.

1) Variation du paramètre *decay*

Ci-dessous, l'aperçu graphique pour *decay* variant entre 0.001 et 10. rouge: *decay* = 0.001, bleu: *decay* = 0.01, vert: *decay* = 0.1, jaune: *decay* = 1, orange: *decay* = 10.



2-3) Poids et frontières

En regardant les poids, on s'aperçoit qu'en augmentant la valeur de *decay*, ces derniers ont une bien moindre amplitude. Pour *decay* = 0.001, les poids varient entre -16 et 17. Pour *decay* = 10, les poids varient entre -0.5 et 0.5.

Ce phénomène s'explique comme suit: *decay* est une sorte de coefficient multiplicateur de la fonction d'erreur. Plus *decay* est proche de 0, moins les poids varient et leur valeurs absolues sont plus faibles.

Ce changement se traduit par des frontières plus droites, moins "biscornues" ou accidentées car il y a moins d'écart entre les poids.

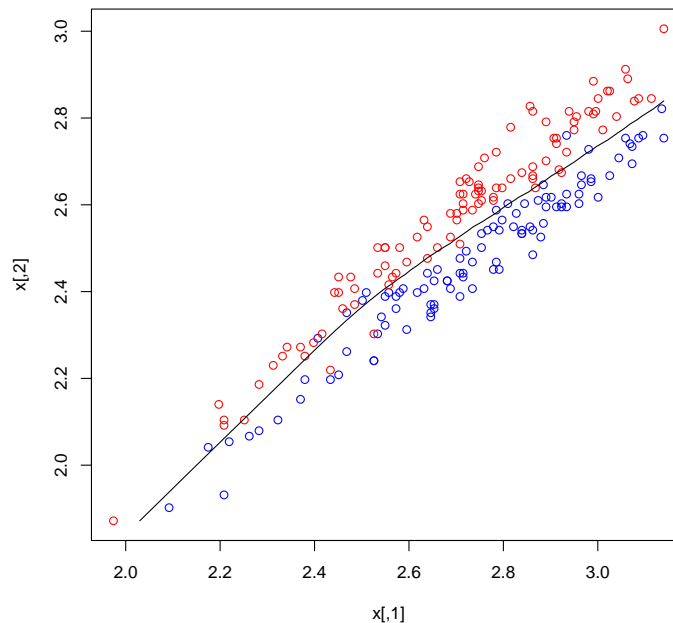
Question 4.

1) Ajustement *size/decay*

Dans ces données, une droite offrirait une bonne discrimination, donc $size = 1$ serait acceptable. Et finalement, en augmentant la valeur de $size$, on n'y gagne pas tant que ça. Avec $size = 2$ et $size = 3$, les résultats sont bons (légèrement meilleurs qu'avec $size = 1$) et le réseau neuronal ne semble pas trop s'être spécialisé pour cet échantillon. Avec $size = 4$, les résultats sont souvent légèrement meilleurs, mais on commence à avoir un réseau de neurones qui s'est spécialisé pour l'échantillon.

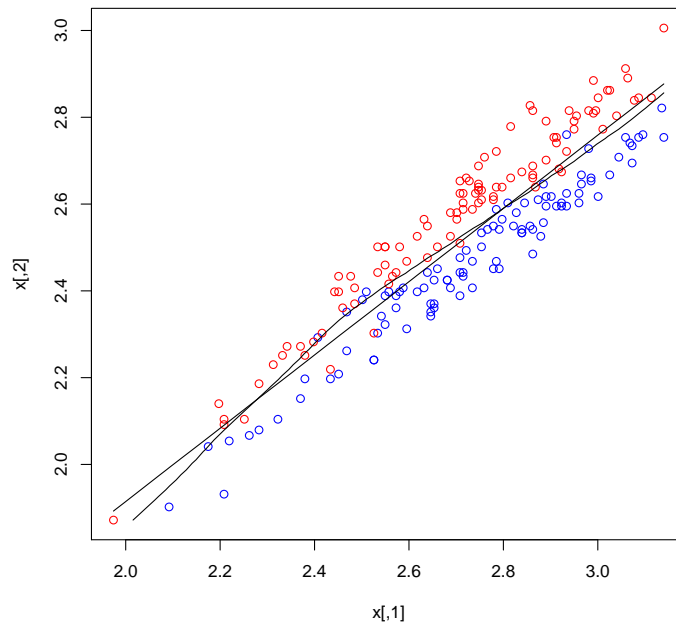
Le paramètre *decay* semble quant à lui ne pas être d'une grande utilité pour ces données. $decay = 0.1$ ou $decay = 1$ ne donne pas de résultats pertinents quelque soit la valeur du paramètre *size*.

Ci-dessous, une classification avec $size = 3$ et $decay = 0$.



2-3) Comparaisons et constatations

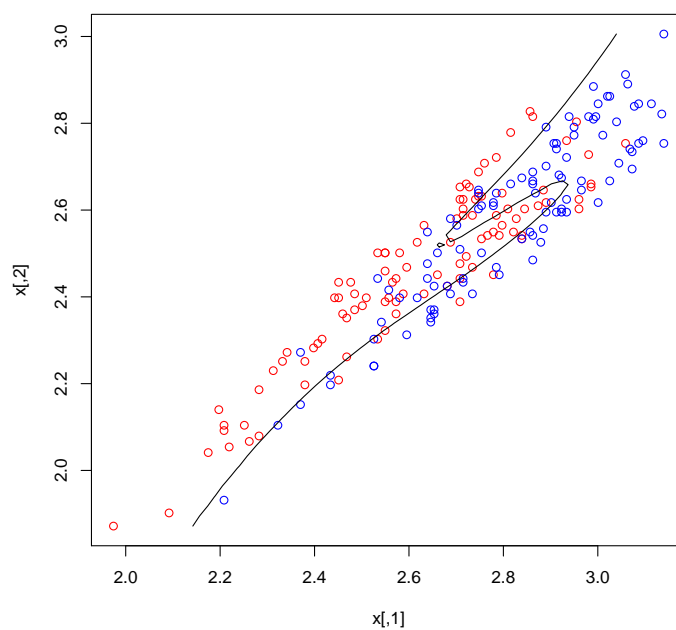
Ci-dessous, on a sur le même graphique une analyse discriminante quadratique et une analyse discriminante linéaire. On s'aperçoit que les résultats se tiennent, même si l'analyse discriminante quadratique donne un résultat légèrement meilleur.



Question 5.

1) Ajustement size / decay

Ici, la frontière entre les 2 classes est très confuse. Néanmoins, il semble qu'avec des valeurs $size = 3$ ou $size = 4$, on obtient de bons résultats. Peut être un peu trop spécialisés, mais ne connaissant que cette échantillon, les résultats paraissent ici intéressants. En essayant des valeurs de $decay$ différentes de 0, on n'obtient aucun résultat intéressant. Ci-dessous, une classification avec $size = 3$ et $decay = 0$.



2-3) Comparaisons et constatations

Ci-dessous, une analyse discriminante quadratique et une analyse discriminante linéaire sur le même graphique. les résultats se tiennent encore une fois, même si l'analyse discriminante quadratique donne une nouvelle fois un résultat légèrement meilleur.

