|  |
| --- |
| Rapport de projet - IMS |
| Détermination du taux de clairance de créatinine par la méthode des réseaux de neurones sur Matlab |
|  |
|  |

|  |
| --- |
|  |

DELEBARRE Fanny

DEVIENNE Emeline

GUEGAN Sarah

**13/05/2014**

**Sommaire**

Introduction 3

1. Contexte**4**
2. Les données**7**
3. Différentes méthodes possibles **8**
4. Méthodes de classification des classes8
5. Réseau de neurones8
6. Les réseaux de neurones**9**
7. Définition d’un neurone9
8. Réseaux de neurones9
9. Historique 9
10. Lois d’apprentissage 10
11. Modélisation d’un neurone 10

i. Fonction de transfert11

1. Architecture d’un réseau 13
2. Apprentissage d’un réseau de neurones 14
3. Mode d’apprentissage 14
4. Règle d’apprentissage15
5. Les premiers résultats et leurs limites**19**
6. Premiers résultats19
7. Les limites22
8. Evolution du projet**23**
9. Réseaux de neurones23
10. Premières fonctions23
11. Premiers résultats23
12. Modifications23
13. Programme Matlab**24**
14. Exploitation des résultats**28**
15. Groupe Femmes28

2) Groupe Hommes29

3) Comparaison avec la méthode MDRD et MDRD modifiée30

4) Pourcentages de réussite par classe31

1. Classes ANAES 31
2. Sous-divisions de classes 33

Conclusion **35**

Bibliographie **36**

**Introduction**

La maladie rénale chronique est très courante et est souvent détectée trop tard. C’est pour cela que le dépistage de cette maladie est fortement conseillé pour les populations à risque, afin d’éviter le passage de la maladie au stade terminal et les complications qui y sont liées. Ce dépistage peut se faire par plusieurs méthodes, comme la détection de marqueurs d’atteinte rénale (visibles à l’échographie ou par biopsie rénale..) ou par le calcul du taux de la clairance de créatinine.

Il existe aujourd’hui, une méthode de calcul qui permet de calculer le taux de clairance de créatinine, utilisée par les laboratoires et même disponible sur internet. Si le calcul prend en compte le sexe, l’âge et même la couleur de la peau, il présente néanmoins certaines limites. En effet, il est utilisé pour les personnes de corpulence dite « normale ». Il n’existe actuellement pas de modèle pour les personnes obèses.

Le sujet de notre projet va nous conduire alors à en mettre en place un, grâce à la méthode des réseaux de neurones sur Matlab.

En s’appuyant sur une thèse portant sur les maladies rénales chroniques, et en utilisant les caractéristiques physiologiques de centaines de personnes (obèses), nous allons proposer une méthode de classification de la clairance de créatinine.

1. **Contexte**

L’insuffisance rénale chronique se définit comme la diminution permanente du débit de filtration glomérulaire (volume de liquide filtré par le rein par unité de temps) inférieur à 60mL/min/1.73m² et elle se mesure avec la clairance de créatinine qui traduit la qualité de filtration du rein.

De nos jours, le nombre de personnes atteintes de cette pathologie est en augmentation, ce qui peut s’expliquer par le vieillissement de la population et par le nombre croissants de personnes diabétiques. De plus, la maladie n’est généralement pas prise à temps, car elle devient symptomatique uniquement au dernier stade. Ainsi, la prise en charge est plus difficile, et le traitement n’est souvent plus adéquat.

Depuis 2002, la maladie rénale chronique est soumise à une classification par stades de sévérité. Cela facilite la prise en charge précoce et de meilleures recommandations.

La classification proposée par l’Accréditation et d’évaluation en Santé (ANAES) se situe dans le tableau ci-dessous :

|  |  |
| --- | --- |
| **Stades** | **DFG (ml/min/1,73 m2)** |
| **1** | > 60 |
| **2** | 30-59 |
| **3** | 15-29 |
| **4** | < 15 |

Pour notre projet, afin d’obtenir une meilleure précision avec notre méthode (les réseaux de neurones), nous avons choisi de séparer encore les classes. Nous en obtenons 10 dont le tableau récapitulatif se trouve ci-dessous :

|  |  |  |
| --- | --- | --- |
| **Stades** | **DFG (ml/min/&,73 m2)** | **Matrice** |
| **1** | > 100 | 1 0 0 0 0 0 0 0 0 0 |
| **2** | 76-99 | 0 1 0 0 0 0 0 0 0 0 |
| **3** | 60-75 | 0 0 1 0 0 0 0 0 0 0 |
| **4** | 50-59 | 0 0 0 1 0 0 0 0 0 0 |
| **5** | 40-49 | 0 0 0 0 1 0 0 0 0 0 |
| **6** | 30-39 | 0 0 0 0 0 1 0 0 0 0 |
| **7** | 25-29 | 0 0 0 0 0 0 1 0 0 0 |
| **8** | 20-24 | 0 0 0 0 0 0 0 1 0 0 |
| **9** | 15-19 | 0 0 0 0 0 0 0 0 1 0 |
| **10** | < 15 | 0 0 0 0 0 0 0 0 0 1 |

L’intérêt de la classification est de diagnostiquer le plus rapidement possible et donc de donner le traitement précoce afin de ralentir la progression et les complications, et ainsi d’être capable d’identifier les personnes dites « à risque ».

Aujourd’hui, ce taux se mesure grâce à des algorithmes connus, comme la méthode MDRD (grâce à laquelle on peut mesurer sa valeur de CrCl depuis chez soi, sur internet), dont la formule est :

*(mL/min/1.73)*

avec :

* [Cr] en en µmol/l, la concentration en créatinine
* k = 1 pour les hommes et 0,742 pour les femmes (On ne distingue pas ici les sujets avec la peau noire)

Cette méthode, bien qu’utilisée très souvent, n’est précise et exacte que dans le cas des sujets considérés comme « normaux ». Or, nous traitons dans notre projet, le cas de sujets obèses, voire en cas d’obésité morbide.

Cette formule a fait l’objet d’une actualisation afin de pouvoir être utilisée avec des créatininémies obtenues par des méthodes raccordées à la spectrométrie de masse. Elle était jusqu’ici développée avec des créatininémies obtenues par méthode colorimétrique. La nouvelle formule est :

*(mL/min/1.73)*

avec les mêmes hypothèses que précédemment.

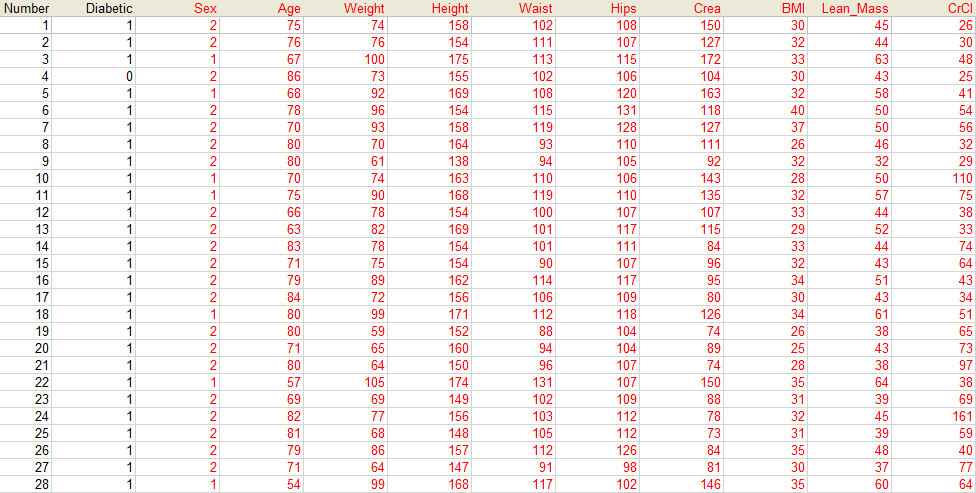
Nous cherchons donc à trouver une nouvelle classification de la CrCl, utilisable dans la situation considérée. Pour cela, nous disposons d’environs 300 sujets, et de leurs caractéristiques morphologiques et physiologiques (taille, poids, âge, sexe, tour de taille…)

Pour cela, nous allons développer un modèle de classification grâce au logiciel Matlab, et à la fonction des réseaux de neurones expliquée dans la partie suivante.

1. **Les données**

Pour notre étude, nous disposons de 277 sujets et de 11 paramètres caractéristiques :

* Présence ou non de diabète
* Sexe
* Age
* Poids
* Taille
* Tour de hanche
* Tour de taille
* Taux de créatinine
* Indice de Masse Corporelle (IMC)
* Masse maigre
* Taux de clairance de créatinine (CrCl)



*Tableau représentant les données brutes avec les différents paramètres*

Grâce à la classification faite plus haut, nous ajoutons pour chaque sujet la classe de la maladie dans laquelle il se situe (de 1 à 10).

Le taux de clairance de créatinine n’est pas utilisé comme entrée, il permet uniquement de déterminer le stade, qui lui sera la sortie.

Nous séparons ensuite aléatoirement nos données en deux, afin de procéder à une phase de test et une phase de validation. (Cette étape avait été faite pour le rapport intermédiaire)

Nous pouvons remarquer que, contrairement à la méthode MDRD, nous utilisons un grand nombre de paramètres (10 contre 3), cela s’explique par le fait que nous cherchons à entrainer notre fonction afin qu’elle réponde le plus précisément possible à nos attentes, ainsi, plus on lui donne d’entrées, plus elle sera précise.

1. **Les différentes méthodes possibles**

Avant toute chose, nous cherchons à étudier un potentiel lien entre les données (ici, la taille, le poids, le tour de hanche, de taille….) et la CrCl. Cela se fait grâce à l’étude de la corrélation, c’est à dire mesurer l’intensité de la liaison qui peut exister entre ces variables.

Pour cela, on effectue une régression (linéaire, de puissance, exponentielle ..) entre les données et la CrCl puis on observe le coefficient de corrélation (de Pearson) et la dispersion (carré du coefficient). Si cette dernière est supérieure à 0.75, on peut dire qu’il y a corrélation. Ce coefficient analyse les relations linéaires.

Il est possible d’utiliser aussi le coefficient de corrélation de Spearman qui lui analyse les relations non-linéaires monotones.

1. **Méthodes de classification des classes :**

Classification hiérarchique ascendante des classes. : Centrer réduire les variables et calculer les distances euclidiennes existantes entre elles. La méthode consiste à appliquer la règle des « plus proches voisins » : On constitue des classes de plus en plus grosses d’objets de moins en moins semblables et on calcule ensuite la distance entre deux classes par les objets les plus proches de ces deux classes.

Ceci a été fait dans le projet similaire au nôtre de l’année dernière. Nous ne nous attarderons pas dessus, car la méthode utilisée par nous sera les réseaux de neurones.

1. **Réseau de neurones :**

Cf partie suivante

1. **Les réseaux de neurones**
2. **Définition d’un neurone**

Un neurone est une cellule excitable qui constitue l’unité fonctionnelle de base du système nerveux. Il assure la transmission du signal biomécanique appelé influx nerveux. Un neurone est capable de répondre à des stimulations et de convertir celles-ci en impulsions nerveuses, qu’il peut alors transmettre (2 propriétés physiologiques : l’excitabilité et la conductivité)

Un neurone est composé d’un corps cellulaire (appelé soma) et de deux types de prolongements : l’axone (unique) et les dentrites (environ 7000 par neurone)

Dans notre cas, nous étudierons les neurones formels, qui sont la représentation mathématique et informatique du neurone biologique. On peut le comparer aux neurones biologiques sur certaines caractéristiques, comme les dentrites, les axones et les synapses. Il reproduit ces caractéristiques grâce à des fonctions et des valeurs numériques. Ils sont regroupés en réseaux de neurones.

1. **Réseaux de neurones**
2. Historique

* **1943 :** Définition du neurone formel (Mc Culloc & Pitts)
* **1948 :** Définition de l’automate cellulaire (Von Neumann)
* **1949 :** Définition de la « Synaptic plasticity » (Hebb)
* **1960 :** Premiers travaux sur le Perceptron (Rosenblatt)
* **1969 :** Limites du Perceptron (Minsky & Papert)
* **1970 :** Turning et jeu de la vie (Conway)
* **1982 :** Modélisation « Spin glasses » (Hopfield)
* **1985 :** Résolution du problème du voyageur de commerce avec un réseau de neurones (Hopfield)
* **1985 :** Définition et expérimentation de l’algorithme de rétro-propagation du gradient (Rumelhart)
* **1986 :** Définition et réalisation du réseau « NetTalk » d’apprentissage à la lecture (Sejnowski)

1. Lois d’apprentissage

Une des principales capacités d’un réseau de neurones est d’apprendre.

Loi de Hebb (1949)

Le coefficient synaptique augmente si les neurones pré et post synaptiques sont actifs simultanément.

où :

t le temps

Wj,i le coefficient synaptique entre le neurone pré-synaptique Nj et le neurone post-synaptique Ni

Vj et Vi les sorties des neurones post et pré-synaptiques

a la constante de renforcement de l’apprentissage

Loi de Hubel et Wiesel (1960)

* Activité conjointe pré et post -> Wj,i augmente
* Neurone pré actif, post repos -> Wj,i diminue
* Neurone post au repos -> Décroissance lente de Wj,i

où :

t le temps

Wj,i le coefficient synaptique entre le neurone pré-synaptique Nj et le neurone post-synaptique Ni

Vj et Vi les sorties des neurones post et pré-synaptiques

f une constante d’oubli

a une constante de renforcement de l’apprentissage

F et G des fonctions de filtrage des sorties

b un niveau de base

1. Modélisation d’un neurone

Un neurone est essentiellement consitué d’un intégrateur qui effectue la somme pondérée de ses entrées, n. Celle-ci est ensuite transformée par une fonction de transfert f. On obtient la sortie a du neurone.



*Modèle d’un neurone*

*p : Matrice des entrées*

*w : Matrice des poids*

*b : Matrice des biais*

**n =**

n s’appelle le niveau d’activation du neurone. Lorsqu’il atteint ou dépasse le seuil b, l’argument de f devient positif ou nul, sinon il est négatif.

Un poids d’un neurone artificiel représente l’efficacité d’une connexion synaptique. Un poids négatif vient inhiber une entrée alors qu’un poids positif vient l’accentuer.

Pour simuler un réseau de neurones, on suppose que le temps est un facteur discret dans nos équations, autrement dit que les neurones sont synchrones, alors qu’ils sont en fait asynchrones dans les réseaux biologiques.

1. **Fonction de transfert**

On peut utiliser différentes fonction d’activation du neurone. Les trois plus utilisées sont les fonctions « seuil », « linéaire » et « sigmoïde ».

* **Fonction seuil**

Une entrée négative ne passe pas le seuil, la fonction retourne alors 0, tandis qu’une entrée positive ou nulle dépasse le seuil et la fonction retourne 1.

Dans notre cas, on a :

Le biais b détermine l’emplacement du seuil sur l’axe .



*Fonction seuil*

* **Fonction linéaire**

Elle affecte son entrée directement à sa sortie :

Dans notre cas, la sortie du neurone correspond à son niveau d’activation dont le passage à zéro se produit lorsque



*Fonction linéaire*

* **Fonction sigmoïde**

Son équation est donnée par :

Selon que l’on soit loin ou près de b, cette fonction ressemble soit à la fonction seuil, soit à la fonction linéaire. Elle est donc un bon compromis entre les deux.

Nous utiliserons cette fonction pour notre code Matlab.



*Fonction sigmoïde*

On notera qu’elle est une version symétrique de la fonction « tangente hyperbolique »

dont la formule est :

1. Architecture d’un réseau

Un réseau de neurones est un maillage de plusieurs neurones, qui sont généralement organisés en couches.



*Couche de N neurones*

Les S neurones sont branchés aux R entrées, on dit alors que la couche est totalement connectée. A chacune des connexions, on associe un poids , qui désigne le poids de la connexion du neurone i à son entrée j. L’ensemble des poids d’une couche forme une matrice W.

W =

*S (Les nombres de neurones et d’entrées sont indépendants.*

On considère que les S neurones forment un vecteur de neurones, on peut alors créer les vecteurs b, n et a (respectivement vecteur de biais, vecteur de niveaux d’activation des neurones et vecteurs de sorties)

Pour construire un réseau de neurones, on combine alors des couches.



*Réseau de neurones à 3 couches*

Chaque couche possède sa propre matrice de poids , où k désigne l’indice de couche.

Chaque couche a comme entrée la sortie de la couche précédente. On pourrait alors, en théorie, mettre à la suite un nombre de couche illimité avec un nombre quelconque de neurones sur chaque. En revanche, en pratique, il est conseillé de ne pas utiliser trop de neurones.

La dernière couche est appelée « couche de sortie » tandis que les précédentes sont les « couches cachées ».

Pour entrainer un réseau de neurones, on modifie la valeur de ses poids et de ses biais afin qu’il réalise la fonction que l’on souhaite.

Pour spécifier sa structure, on choisit le nombre de couches et le nombre de neurones sur chaque couche. Le nombre d’entrées du réseau ainsi que le nombre de neurones sur la couche de sortie sont fixés par le problème à résoudre. Si l’on souhaite x variables en entrées et y variables en sortie, alors R = x et = y (avec M indice de la couche de sortie et nombre de couches dans le réseau).

1. Apprentissage d’un réseau de neurones

L’apprentissage d’un réseau de neurones se définit comme la mise à jour des poids, afin qu’il nous donne les sorties attendues. C’est la caractéristique principale d’un réseau, et il peut se faire de différentes manières.

1. *Modes d’apprentissage*

* **Mode supervisé**

Le réseau s’adapte par comparaison entre le résultat calculé et la réponse attendue en sortie. Il se modifie jusqu’à trouver la bonne sortie correspondant à une entrée donnée. C’est la méthode que nous utiliserons.

* **Renforcement**

C’est une sorte d’apprentissage supervisé : Le réseau apprend la corrélation entrée/sortie via une estimation de son erreur. Il tend donc à maximiser son index de performance qui lui est fourni, qu’on appelle « signal de renforcement ». Ainsi, le système est capable de savoir si la réponse qu’il sort est correcte tout en ne sachant pas quelle est la bonne réponse.

* **Mode non-supervisé (auto-organisationnel)**

L’apprentissage est basé sur des probabilités. Le réseau se modifie en fonction des régularités statistiques de l’entrée et établit des catégories, en attribuant et en optimisant une valeur de qualité, aux catégories reconnues. Le réseau est libre de converger vers n’importe quel état final lorsqu’on lui présente un motif.

* **Mode hybride**

Il reprend les deux autres approches : Une partie des poids sera déterminée par apprentissage supervisé et l’autre par apprentissage non-supervisé.

1. *Règles d’apprentissage*

* **Non supervisé**
* Règle de Hebb

Soit un réseau totalement connecté avec :

: Valeur à l’instant du poids liant le neurone au neurone .

: Fonction d’activation du type tout ou rien

si

si

La règle de Hebb suit le comportement du neurone biologique :

***Si deux neurones interconnectés sont simultanément activés, alors le poids de la connexion qui les relie doit être renforcée.***

*avec  : Une constante positive définissant l’intensité de l’apprentissage*

On introduit dans l’algorithme un terme d’oubli dont le taux, noté , est une constante positive inférieure à 1, afin d’éviter des gains élevés.

La valeur maximale des poids est définie par et  : Si les neurones et sont activés en même temps, nous avons :

et à l’équilibre :

Soit

* Règle de Kohonen

Elle permet au neurone d’apprendre le vecteur présenté en entrée, et est souvent utilisée en reconnaissance de formes. Elle s’écrit sous la forme :

*avec :*

L’apprentissage se fait lorsque l’indice du neurone appartient à un ensemble , défini en fonction des objectifs.

* Règle Instar

Cette règle permet d’éviter une décroissance excessive des poids que la règle de Hebb peut provoquer.

Lorsque on a :

Si le neurone est inactif (), on a :

Sinon, si le neurone est actif ()

Dans ce cas, le vecteur poids est déplacé vers le vecteur d’entrée. Si les vecteurs d’entrée sont normalisés, les vecteurs poids le seront aussi après l’apprentissage.

La règle de Kohonen est équivalente à la règle Instar lorsque les fonctions d’activation sont du type tout ou rien et que est l’ensemble des indices des neurones activés à l’instant , tels que :

* Règle Outstar

On utilise cette règle lorsqu’on veut régénérer un patron donné à partir d’une entrée linéaire. Les composantes du vecteur qui définit le patron sont supposées de module inférieur ou égal à 1 : Nous avons alors un neurone avec une entrée scalaire et une sortie vectorielle.



Si l’entrée , alors le neurone reproduira le patron (caractérisé par le vecteur .

Les fonctions d’activation sont du type saturation

*telle que :*

*si*

*si*

*si*

Pour : , on a :

Le but de cet apprentissage est, lorsque , d’obtenir :

Cette règle impose un oubli qui est proportionnel à l’entrée

Si (taux de décroissance = taux d’apprentissage), alors on a :

On remarquera que :

Il n’y a pas d’apprentissage si

L’apprentissage s’effectue en une itération si

* **Supervisé**
* Règle du perceptron

Soit un réseau :



L’objectif sera d’apprendre les K patrons , et la fonction d’activation est un tout ou rien.

Pour la suite, nous noterons le patron présenté à l’instant , la sortie désirée et la sortie réelle .

A l’instant , lorsqu’on présente le patron , l’erreur sur le neurone est :

Cette méthode, comme dit précédemment, consiste à présenter successivement tous les patrons, et à ajuster à chaque étape le vecteur des poids des connexions des entrées du neurone. Si la sortie obtenue est celle désirée, alors les poids ne sont pas modifiés.

* Règle du pseudo-inverse

On considère un réseau monocouche



On cherche à déterminer les poids W du réseau, qui, à partir de vecteurs prototypes , minimisent la quantité :

*avec les couples fixés, étant une mesure de .*

On note :

On obtient :

inversible : On peut annuler l’erreur en posant

Sinon, on peut rendre l’erreur minimale en prenant avec la pseudo-inverse de telle que :

Pour une entrée quelconque , la sortie est un vecteur voisin du vecteur correspondant au vecteur le plus proche de au sens des moindres carrés.

1. **Premiers résultats et leurs limites**
2. **Premiers résultats**

Voici les premiers résultats les plus cohérents que nous obtenons ainsi que la méthode par laquelle nous sommes arrivés ces résultats.

* Nous chargeons d’abord nos données d’entrée et de sortie puis nous faisons la transposée

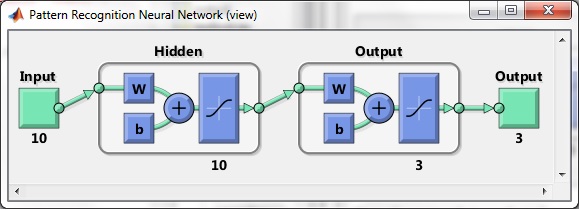
*Load input X=input’*

*Load output y=output'*

* Dans ce cas, nous chargeons 10 neurones cachés. Voici la représentation des réseaux de neurones tels que nous les utilisons dans cet exemple.

*net=patternnet(10);*

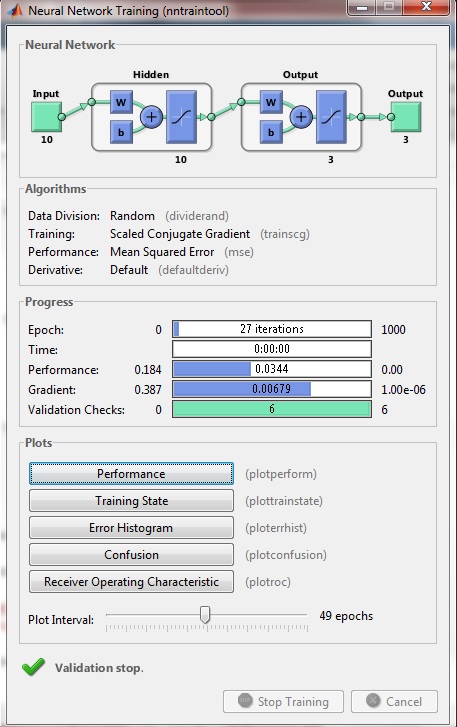
*view(net)*



* Nous lançons l’interface des réseaux de neurones grâce à la fonction « nntraintool »

*[net,tr]=train(net,x,y);*

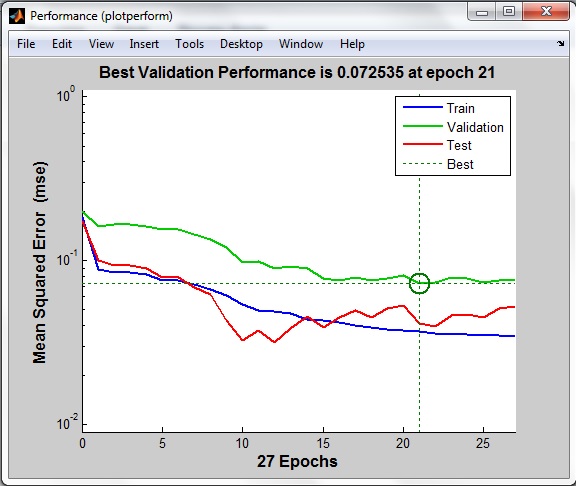
*nntraintool*

**

Les données sont automatiquement séparées en données de training, validation et tests.

* On observe ensuite la performance de l’essai durant la phase de test.

*plotperform(tr)*



Le travail qui a réalisé la meilleure phase de validation se situe après la phase d’essai.

* On teste ensuite les réseaux de neurones.

La moyenne de l’écart type de l’essai des reseaux de neurones peut maintenant être mesuré avec le respect des données du test. Les matrices de sorties de l’essai étant de valeur binaire (0 et 1), on utilise la fonction vec2ind afin de nous indiquer la position du haut élément de chaque matrice de sortie.

*testX=x(:,tr.testInd);*

*testY=y(:,tr.testInd);*

*testT=net(testX);*

*testIndices=vec2ind(testY)*

On obtient alors comme résultat :

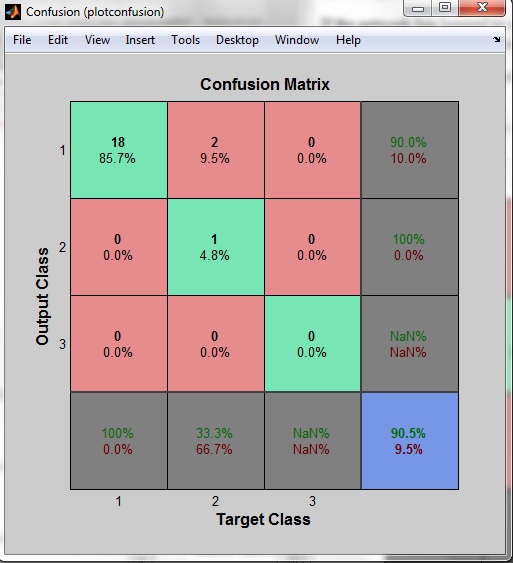
*testIndices = 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1*

Cela nous montre que sur le test effectué en validation, 3 personnes sont au stade 2 et 18 sont au stade 1.

* On cherche alors à obtenir la matrice de confusion.

Cette matrice montre le pourcentage de classification correcte (vert) et incorrecte (rouge).

*plotconfusion(testY,testT)*



*[c,cm]=confusion(testY,testT)*

On obtient :

*c = 0.0952*

*cm = 18 0 0*

*2 1 0*

*0 0 0*

C représente la valeur de classification incorrecte et cm représente la matrice de confusion (comme représenté sur la figure ci-dessus.

* On cherche maintenant le pourcentage exact de classification correcte et incorrecte.

*fprintf('Percentage Correct Classification : %f%%\n',100\*(1-c));*

*fprintf('Percentage Incorrect Classification : %f%%\n',100\*c);*

On obtient :

*Percentage Correct Classification : 90.476190%*

*Percentage Incorrect Classification : 9.523810%*

1. **Les limites**

Avec un pourcentage de corrélation élevé, on peut dire que les résultats de nos premiers essais sont cohérents.

Malgré tout, nous constatons que nos résultats ne sont pas optimaux.

De plus, ces résultats ont été effectués avec des réseaux de neurones avec apprentissage. Nos résultats ne nous permettent donc pas de dire qu’ils seront toujours cohérents avec des résultats provenant de l’utilisation des réseaux de neurones sans apprentissage.

1. **Evolution du projet**
2. **Réseaux de neurones**

Le concept de « réseaux de neurones » et en particulier sous le logiciel Matlab nous était jusqu’alors inconnu. Le choix de notre sujet s’est plus porté sur l’alliance de l’informatique et du médical qui se rapprochait de notre projet professionnel.

Nous avons du alors faire beaucoup de recherches, parfois soldées par des incompréhensions sur les réseaux. Nous avons eu la chance d’être aiguillées par un ancien élève HEI (aujourd’hui ingénieur dans un hôpital), qui nous a alors expliqué les notions de base ainsi que certaines fonctions sur le logiciel.

Ceci, ajouté à nos recherches intensives et l’aide de nos référents, nous a permis d’appréhender de manière plus sereine la suite du projet.

1. **Premières fonctions**

Nous avons commencé à créer nos réseaux de neurones sans faire de distinction entre les hommes et les femmes de notre base de données de sujets.

En suivant les conseils de nos professeurs ainsi qu’en découvrant que la méthode MDRD distinguait le sexe, nous avons à notre tour séparer nos sujets, créant ainsi un groupe Femme et un groupe Homme.

Nos premières fonctions sous Matlab n’ont hélas sorti que des résultats que nous ne savions pas exploiter, ne sachant pas encore vraiment comment utiliser cette boîte à outils.

1. **Premiers résultats**

A une période, nous pensions avoir des résultats qui à l’époque, nous semblaient cohérents et répondaient tout à fait au sujet demandé.

Malheureusement, nous avons détecté une erreur dans notre script, qui rendait alors nos matrices de sorties inexploitables et erronées.

Malgré tout, ceci nous a permis d’envisager nos résultats sous un nouveau angle et en détectant les erreurs commises, de pouvoir aller plus loin.

Nous savions alors exactement comment obtenir les matrices finales recherchées et sur quels paramètres jouer pour obtenir des meilleurs résultats.

1. **Modifications**

Malgré tout ça, les résultats que nous obtenions étaient soit peu concluants (taux de réussite trop faible) soit nous n’obtenions pas la convergence souhaitée malgré un nombre d’itérations important.

Nous avons alors fait le choix de revenir au nombre de classes initial (4 classes) et nous avons réussi à sortir des résultats plus proches de ceux que nous souhaitions.

En effet, un nombre de 10 classes entrainait un nombre d’erreur plus important, étant donné que les intervalles étaient plus réduits.

Cependant, nous remarquons encore qu’une incompréhension persiste, en effet, nous arrivons à faire converger le groupe Femme plus rapidement que le groupe Homme (6000 contre plus de 25000 itérations.) Ceci s’explique peut-être par la base de donnée utilisée.

1. **Programme Matlab**

Que ce soit pour les hommes ou pour les femmes, nous utilisons 50 neurones cachés.

On utilise un réseau de neurones par rétro-propagation.

**Script de la fonction**

%Valeurs spécifiques utilisées

hidden\_neurons = 50;

epochs = 30000;

% ------- Téléchargement des données -------

train\_inp = entrees;

train\_out = sorties;

% Vérifier qu'il y ait le même nombre d'entrées que de sorties

if size(train\_inp,1) ~= size(train\_out,1)

disp('ERROR: data mismatch')

return

end

%standardiser les données

%entrées

mu\_inp = mean(train\_inp);

sigma\_inp = std(train\_inp);

train\_inp = (train\_inp(:,:) - mu\_inp(:,1)) / sigma\_inp(:,1);

%sorties

train\_out = train\_out';

mu\_out = mean(train\_out);

sigma\_out = std(train\_out);

train\_out = (train\_out(:,:) - mu\_out(:,1)) / sigma\_out(:,1);

train\_out = train\_out';

%lire la nombre d'entrées

patterns = size(train\_inp,1);

%ajouter un biai à une entrée

bias = ones(patterns,1);

train\_inp = [train\_inp bias];

%lire le nombre de paramètres des entrées + le biai

inputs = size(train\_inp,2);

%---------- Données téléchargées ------------

%--------- Ajouter des boutons de contrôle ---------

%ajouter un bouton pour arrêter plus tôt

hstop = uicontrol('Style','PushButton','String','Stop', 'Position', [5 5 70 20],'callback','earlystop = 1;');

earlystop = 0;

%ajouter un bouton pour réinitialiser les poids

hreset = uicontrol('Style','PushButton','String','Reset Wts', 'Position', get(hstop,'position')+[75 0 0 0],'callback','reset = 1;');

reset = 0;

%ajouter un curseur pour ajuster le taux d'apprentissage

hlr = uicontrol('Style','slider','value',.1,'Min',.01,'Max',1,'SliderStep',[0.01 0.1],'Position', get(hreset,'position')+[75 0 100 0]);

% ---------- Ensemble des poids -----------------

%Ensemble des poids aléatoires initiaux

weight\_input\_hidden = (randn(inputs,hidden\_neurons) - 0.5)/10;

weight\_hidden\_output = (randn(1,hidden\_neurons) - 0.5)/10;

%-----------------------------------

%--- l'apprentissage commence! -----

%-----------------------------------

%faire un nombre d'itérations

for iter = 1:epochs

%Obtenir le taux d'apprentissage par le curseur

alr = get(hlr,'value');

blr = alr / 10;

%boucle des entrées choisies aléatoirement

for j = 1:patterns

%sélectionner une entrée aléatoire

patnum = round((rand \* patterns) + 0.5);

if patnum > patterns

patnum = patterns;

elseif patnum < 1

patnum = 1;

end

%Obtenir l'entrée actuelle

this\_pat = train\_inp(patnum,:);

act = train\_out(patnum,1);

%Calculer l'erreur grâce à cette entrée

hval = (logsig(this\_pat\*weight\_input\_hidden)');

pred = hval'\*weight\_hidden\_output';

error = pred - act;

% Ajuster les poids des neurones de sortie

delta\_HO = error.\*blr .\*hval;

weight\_hidden\_output = weight\_hidden\_output - delta\_HO';

% Ajuster les poids des neurones d'entrée

delta\_IH= alr.\*error.\*weight\_hidden\_output'.\*(1-(hval.^2))\*this\_pat;

weight\_input\_hidden = weight\_input\_hidden - delta\_IH';

end

% -- une autre itération est finie

%tracer l'erreur à la fin de chaque itération

pred = weight\_hidden\_output\*logsig((train\_inp\*weight\_input\_hidden)');

error = pred' - train\_out;

err(iter) = (sum(error.^2))^0.5;

figure(1);

plot(err)

%réinitialiser le poids si c'est nécessaire

if reset

weight\_input\_hidden = (randn(inputs,hidden\_neurons) - 0.5)/10;

weight\_hidden\_output = (randn(1,hidden\_neurons) - 0.5)/10;

fprintf('weights reaset after %d epochs\n',iter);

reset = 0;

end

%arrêter si nécessaire

if earlystop

fprintf('stopped at epoch: %d\n',iter);

break

end

%arrêter si l'erreur est faible

if err(iter) < 0.01

fprintf('converged at epoch: %d\n',iter);

break

end

end

%-----FIN---------

%affichage réel, de prédiction et d'erreur

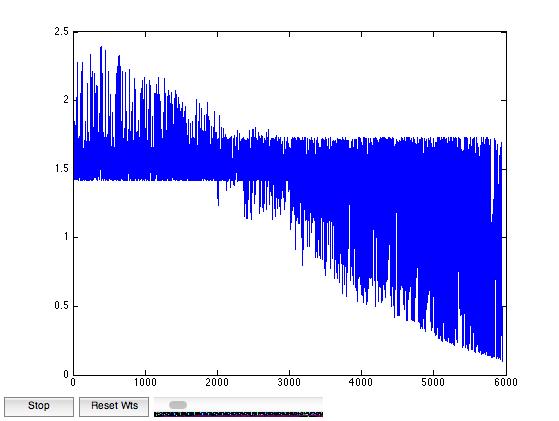
fprintf('state after %d epochs\n',iter);

a = (train\_out\* sigma\_out(:,1)) + mu\_out(:,1);

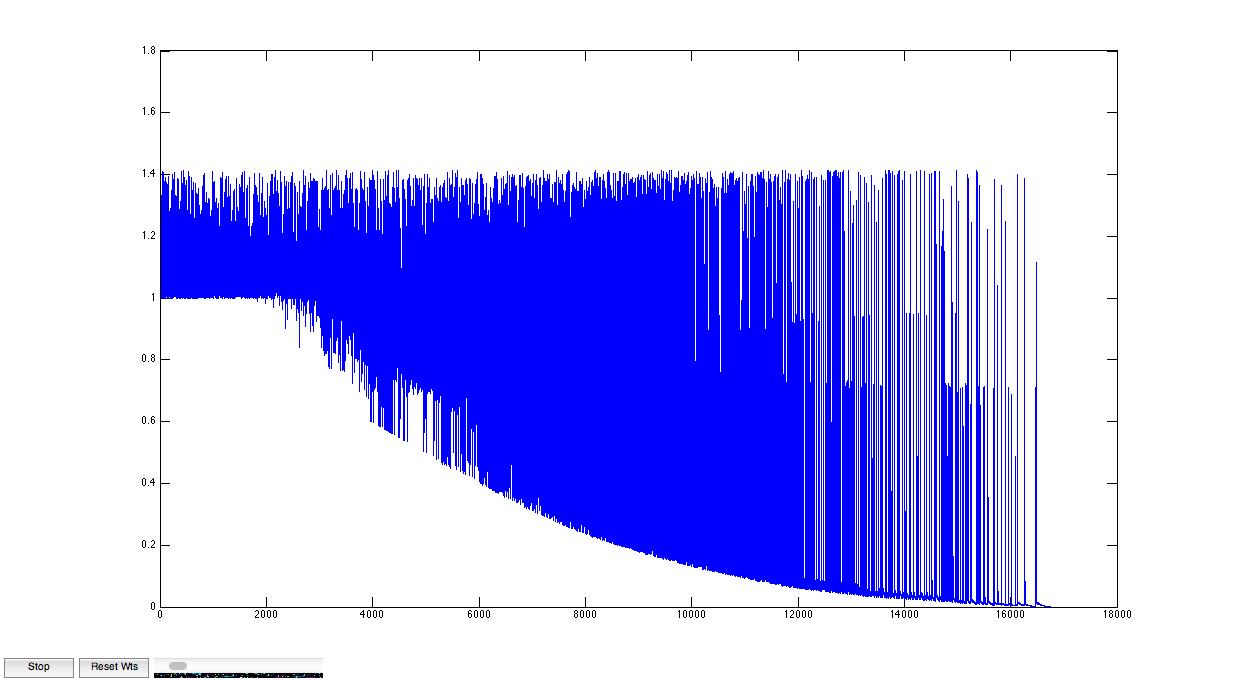
b = (pred'.\* sigma\_out(:,1)) + mu\_out(:,1);

act\_pred\_err = [a b b-a];

Avec nos entrées hommes et femmes, nous obtenons les courbes de convergence suivantes :



*Courbe de convergence Femmes*



*Courbe de convergence Hommes*

1. **Exploitation des résultats**
2. **Groupe Femme**



*Tableau Excel fonction Matlab (Femmes)*

Grâce à ce tableau Excel, nous avons pu calculer les classes de nos sujets par la méthode des réseaux de neurones, et les regrouper dans le tableau suivant.



*Tableau Excel comparatif des Femmes*

Pour les méthodes MDRD et MDRD modifiée, nous avons gardé les sous-divisions de classes, en revanche, comme expliqué précédemment, pour la méthode des réseaux de neurones, nous avons uniquement travaillé avec les classes de 1 à 4.

Dans le tableau, nous avons comparé les résultats obtenus avec les réseaux de neurones aux résultats donnés. En rouge, nous trouvons les résultats qui ne correspondent pas.

Nous obtenons finalement un pourcentage de réussite de près de 89%.

1. **Groupe Homme**



*Tableau Excel fonction Matlab (Hommes)*

Même principe que pour les femmes.



*Tableau Excel comparatif des Hommes*

On procède de la même façon pour les hommes. Nous obtenons finalement un taux de réussite d’environ 73%.

1. **Comparaison avec la méthode MDRD et MDRD modifiée**

Comme expliqué précédemment, la méthode aujourd’hui utilisée pour le calcul de la clairance de la créatinine est celle de la MDRD et MDRD modifiée. Cependant, elles conviennent pour des personnes de corpulence dite « normale » mais n’est pas adaptée pour les personnes obèses, comme celles étudiées pour notre projet.

Ainsi, après calcul de leur taux de clairance de créatinine, nous avons classé les sujets avec la méthode MDRD, MDRD modifiée et celle des réseaux de neurones mise en place, d’abord par sous-divisions de classes (de 1 à 10) puis par classes actuellement utilisées et proposées dans la thèse sur laquelle nous nous sommes appuyé (de 1 à 4).

Nous avons ensuite procédé à la comparaison des résultats afin de déterminer si notre méthode était mieux adaptée pour ce type de sujet.

Dans notre tableau, les classes de 1 à 4 sont appelées « Classe ANAES », les autres sont les sous-divisions.

On remarque que les résultats sont peu précis avec les méthodes MDRD et MDRD modifiée en sous-divisant les classes (Autour de 30% de résultats justes), alors qu’avec les classes « ANAES », on obtient des résultats de 85% de cohérence avec les sujets de la thèse.

En effet, en divisant les classes, l’intervalle se réduit, ce qui entraine forcément plus d’erreur. On peut voir que l’écart type est très proche de 1 pour les hommes et les femmes, ainsi, les résultats calculés se situent le plus généralement soit dans la classe en dessous, soit dans celle au-dessus que celle attendue.

1. **Pourcentages de réussite par classe**
   1. Classes ANAES

Nous avons tout d’abord calculé, pour chaque méthode, le nombre de sujets se trouvant dans chaque classe. Les résultats sont présentés dans le tableau ci-dessous :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Classes** | **Femmes** | | | | **Hommes** | | | |
|  | *CrCl donnée* | *MDRD* | *MDRD modifiée* | *Réseau de neurones* | *CrCl donnée* | *MDRD* | *MDRD modifiée* | *Réseau de neurones* |
| **1** | 132 | 141 | 137 | 140 | 101 | 97 | 93 | 80 |
| **2** | 27 | 23 | 26 | 21 | 12 | 16 | 20 | 32 |
| **3** | 5 | 0 | 1 | 3 | 0 | 0 | 0 | 1 |
| **4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Nombre de sujets par classe*

Ensuite, pour chaque méthode et pour chaque classe, nous avons compté le nombre de sujets ne devant pas se trouver dans la classe étudiée, en comparaison avec les résultats de la thèse. Grâce à cela, nous avons pu calculer l’erreur moyenne (la classe qui présente le plus de résultats erronés) ainsi que l’écart type.

Erreur moyenne :

Avec :

 : Nombre de sujets mal classé (par classe)

: Nombre total de sujets mal classés

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Classes** | **Femmes** | | | **Hommes** | | |
|  | *MDRD* | *MDRD modifiée* | *Réseau de neurones* | *MDRD* | *MDRD modifiée* | *Réseau de neurones* |
| **1** | **15** | 14 | 14 | 7 | 6 | 5 |
| **2** | 9 | 11 | 5 | 11 | 14 | 25 |
| **3** | 0 | 0 | 0 | 0 | 0 | 1 |
| **4** | 0 | 0 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |
| **Erreur Moyenne** | 1,38  (1) | 1,44  (1) | 1,26  (1) | 1,61  (2) | 1,7  (2) | 1,87  (2) |
| **Ecart Type** | 1,12 | 1,11 | 1,15 | 1 | 1 | 1,05 |

*Nombre de sujets mal classés*

Par exemple, nous trouvons ici **15 sujets** de classe 1 parmi les 141 comptés en MDRD qui devraient être en classe 2, 3 ou 4.

Pour finir, nous avons regroupé dans le tableau ci-dessous les pourcentages de réussite pour chaque classe.

Le calcul se fait de la manière suivante :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Classes** | **Femmes** | | | | **Hommes** | | | |
|  | *CrCl donnée* | *MDRD* | *MDRD modifiée* | *Réseau de neurones* | *CrCl donnée* | *MDRD* | *MDRD modifiée* | *Réseau de neurones* |
| **1** | 100% | 95,45% | 93,18% | 95,45% | 100% | 89,11% | 76,24% | 74,25% |
| **2** | 100% | 51,85% | 55,26% | 59,26% | 100% | 41,66% | 50% | 58,33% |
| **3** | 100% | 0% | 20% | 60% | 100% | 100% | 100% | 0% |
| **4** | X | X | X | X | X | X | X | X |

*Pourcentage de réussite par classe*

On peut ainsi dire que pour le groupe Femmes, notre réseau de neurones donne des résultats satisfaisants, en comparaison avec les autres méthodes.

N’ayant aucun sujet se trouvant en classe 4, il nous est impossible de calculer un pourcentage d’erreur pour cette classe. Mais nous pouvons voir que, Hommes et Femmes confondus, toutes les méthodes admettent un pourcentage de réussite plus élevé pour la classe 1.

Les résultats mitigés (pour les réseaux de neurones) pour la classe 3 s’expliquent par le faible nombre de sujets présents dans cette classe, dans le panel proposés.

Nos résultats dans le groupe Hommes pour les réseaux de neurones, même s’ils restent cohérents, ne nous semblent peu satisfaisants.

Nous avions pour but de créer une méthode adaptée aux personnes dites « obèses », cependant, les résultats trouvés restent à peu près similaires à ceux donnés par les autres méthodes.

Même si nous n’avons pas réussi à obtenir des résultats avec nos réseaux de neurones pour les classes de 1 à 10, nous avons fait les mêmes calculs que précédemment avec ceux obtenus grâce aux méthodes MDRD.

* 1. Sous-divisions de classes

Pour les classes de 1 à 10, les calculs restent identiques. Les tableaux de résultats sont proposés ci-dessous.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Classes** | **Femmes** | | | **Hommes** | | |
|  | *CrCl donnée* | *MDRD* | *MDRD modifiée* | *CrCl donnée* | *MDRD* | *MDRD modifiée* |
| **1** | 63 | 43 | 30 | 59 | 31 | 21 |
| **2** | 46 | 64 | 66 | 26 | 45 | 48 |
| **3** | 23 | 34 | 41 | 16 | 21 | 24 |
| **4** | 9 | 12 | 14 | 6 | 8 | 12 |
| **5** | 9 | 8 | 9 | 3 | 6 | 6 |
| **6** | 9 | 3 | 3 | 3 | 2 | 2 |
| **7** | 5 | 0 | 1 | 0 | 0 | 0 |
| **8** | 0 | 0 | 0 | 0 | 0 | 0 |
| **9** | 0 | 0 | 0 | 0 | 0 | 0 |
| **10** | 0 | 0 | 0 | 0 | 0 | 0 |

*Nombre de sujets par classe*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classes** | **Femmes** | | **Hommes** | |
|  | *MDRD* | *MDRD modifiée* | *MDRD* | *MDRD modifiée* |
| **1** | 18 | 11 | 9 | 7 |
| **2** | 45 | 44 | 36 | 39 |
| **3** | 25 | 32 | 15 | **19** |
| **4** | 11 | 13 | 7 | 11 |
| **5** | 7 | 7 | 6 | 6 |
| **6** | 2 | 2 | 2 | 2 |
| **7** | 0 | 0 | 0 | 0 |
| **8** | 0 | 0 | 0 | 0 |
| **9** | 0 | 0 | 0 | 0 |
| **10** | 0 | 0 | 0 | 0 |
|  |  |  |  |  |
| **Erreur Moyenne** | 2,5 (2) | 2,7 (3) | 2,61 (3) | 2,71 (3) |
| **Ecart Type** | 1,63 | 1,70 | 1,69 | 1,55 |

*Nombre de sujets mal classés (par classe)*

Par exemple, nous trouvons **19 sujets** en classe 3 parmi les 24 comptés en MDRD modifiée qui devraient être dans une autre classe.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Classes** | **Femmes** | | | **Hommes** | | |
|  | *Crcl donnée* | *MDRD* | *MDRD modifiée* | *CrCl donnée* | *MDRD* | *MDRD modifiée* |
| **1** | 100% | 39,68% | 30,16% | 100% | 37,28% | 23,70% |
| **2** | 100% | 41,30% | 47,83% | 100% | 34,60% | 34,60% |
| **3** | 100% | 43,48% | 39,13% | 100% | 37,50% | 31,25% |
| **4** | 100% | 11,11% | 11,11% | 100% | 16% | 16% |
| **5** | 100% | 11,11% | 22,22% | 100% | 0% | 0% |
| **6** | 100% | 11,11% | 11,11% | 100% | 0% | 0% |
| **7** | 100% | 0% | 20% | X | X | X |
| **8** | X | X | X | X | X | X |
| **9** | X | X | X | X | X | X |
| **10** | X | X | X | X | X | X |

*Pourcentage de réussite par classe*

Comme prévu, les résultats avec les méthodes MDRD et MDRD modifiée sont encore moins satisfaisants que ceux obtenus avec les classes ANAES. De plus, les méthodes n’étant pas adaptées pour le type de notre panel, les pourcentages de réussite sont faibles.

**Conclusion**

Ce projet nous a mené à créer un programme sous Matlab afin de faire une classification de la clairance de créatinine.

Nous avons dû pour cela comprendre le fonctionnement d’un réseau de neurones et son application dans le logiciel.

Après quelques hésitations et des essais peu concluants, nous avons finalement réussi à obtenir des résultats cohérents, en revenant à un système de classes allant de 1 à 4, au lieu des sous-divisions (de 1 à 10).

Nous restons cependant insatisfaites quant aux résultats finaux, plus spécialement sur la base de données des hommes. En effet, si notre programme est satisfaisant dans la classification pour les femmes, il reste moins efficace pour le sexe masculin. Nous obtenons des pourcentages de réussite inférieurs à ceux donnés par les méthodes déjà existantes (MDRD et MDRD modifiée), pourtant non conçues pour les sujets étudiés, soit des personnes dites « obèses ». Cela peut s’expliquer par notre base de données des hommes qui pourrait alors être peu adaptée dans nos calculs.

Malgré quelques interrogations pendant la durée du projet, et des débuts avec la méthode un peu difficiles, ce projet nous a semblé très intéressant dans le sens où il nous a permis, pour la première fois, de lier le côté informatique au médical et aussi de nous perfectionner dans la prise en main du logiciel Matlab. De plus il s’est avéré être en adéquation avec notre projet professionnel.

**Bibliographie**

**Contexte**

* Thèse de Sandrine LE TURNIER épouse CHAN

*« Evaluation calculée de la clairance de la créatinine : comparaison des modèles proposés et évaluation de nouvelles formulations »*

* [www.soc-nephrologie.org/PDF/epart/rein/ABM-MRC.pdf](http://www.soc-nephrologie.org/PDF/epart/rein/ABM-MRC.pdf)

**Réseaux de neurones**

* <http://www-igm.univ-mlv.fr/~dr/XPOSE2002/Neurones/index.php?rubrique=Apprentissage>
* [http://books.google.fr/books?id=P1GgimurnGoC&pg=PA21&lpg=PA21&dq=régle+d'apprentissage+par+compétitions&source=bl&ots=oJJS4qOWDx&sig=r0U4PKDEtuVDoilMisRyBz4njho&hl=fr&sa=X&ei=dkBrU-2DDKbL0AXy84CQAw&ved=0CEEQ6AEwAg#v=onepage&q=régle%20d'apprentissage%20par%20compétitions&f=false](http://books.google.fr/books?id=P1GgimurnGoC&pg=PA21&lpg=PA21&dq=régle+d'apprentissage+par+compétitions&source=bl&ots=oJJS4qOWDx&sig=r0U4PKDEtuVDoilMisRyBz4njho&hl=fr&sa=X&ei=dkBrU-2DDKbL0AXy84CQAw&ved=0CEEQ6AEwAg#v=onepage&q=régle%20d'apprentissage%20par%)
* <http://olivier.teytaud.pagesperso-orange.fr/publis/serpilliere.pdf>
* <http://www.episerv.org/reading/RNF%20-%20Reseaux%20de%20neurones.pdf>
* <http://www.futura-sciences.com/magazines/high-tech/infos/dico/d/informatique-reseau-neuronal-601/>
* <http://82.245.102.21/html/reseau_de_neurones.pdf>
* <http://www.neurones.espci.fr/Theses_PS/Stricker_M/CHAP6.pdf>

**Script**

* [www.philbrierley.com](http://www.philbrierley.com)
* <http://deptinfo.cnam.fr/new/spip.php?rubrique247>
* [http://www2.ift.ulaval.ca/lamontagne/ift17587/modules/module3/r%C3%A9seauxNeurones.pdf](http://www2.ift.ulaval.ca/~lamontagne/ift17587/modules/module3/r%C3%A9seauxNeurones.pdf)
* [http://public.enst-bretagne.fr/dgueriot/Sujet%20TP%20-%20RN.pdf](http://public.enst-bretagne.fr/~dgueriot/Sujet%20TP%20-%20RN.pdf)
* « Apprendre et Maitriser MATLAB » versions 4&5 SIMULINK

M.Mokhtari/A.Mesbah (Edition Springer)

**Méthode MDRD**

* <http://www.leblogmedical.fr/wp-content/Calcul_FR.htm>