MASTER 2 INFORMATIQUE 2016-2017 Université des antilles

FICHIER D'EXPÉRIENCE

BARNACIN Endrick

Table des matières

Partie 1 : PARZEN CLASSIFIER

Le but des expériences de cette partie est d'évaluer les performances du classifieur à fenêtres de parzen sur notre jeu de données.

Présentation du window parzen classifier :

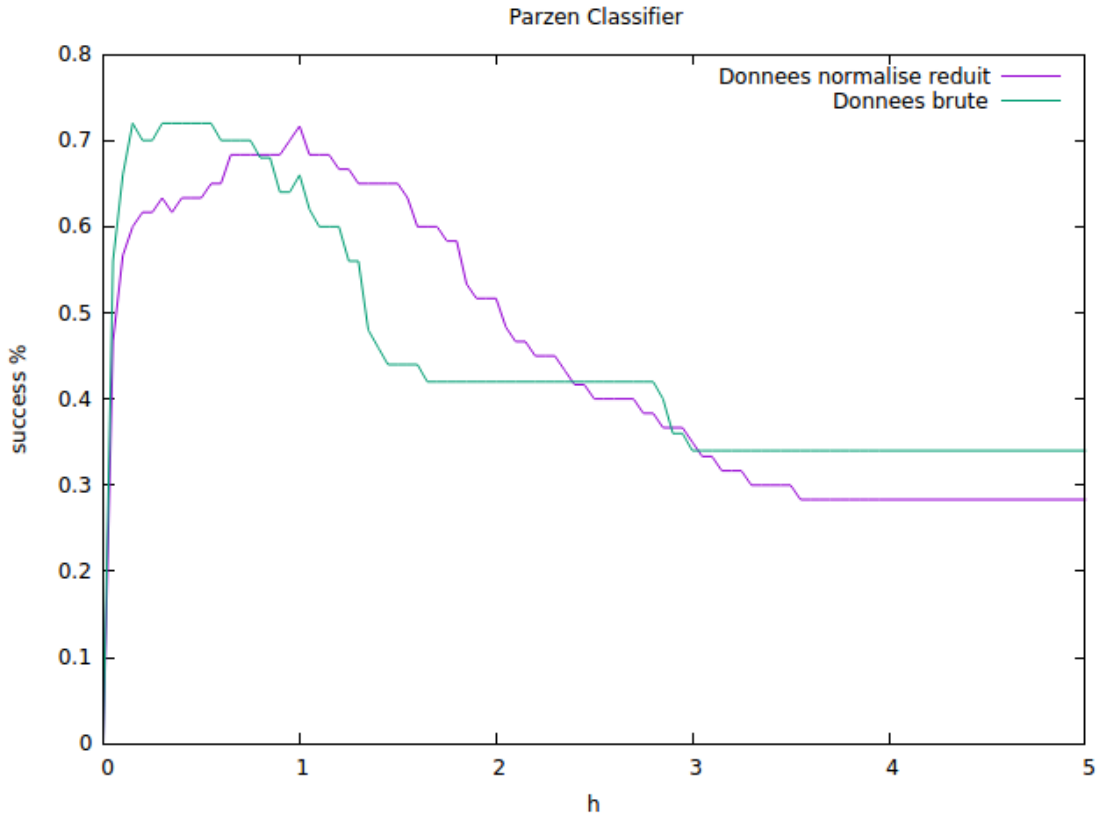
En statistique, l'estimation par noyau (ou encore méthode de Parzen-Rosenblatt) est une méthode non-paramétrique d'estimation de la densité de probabilité d'une variable aléatoire. Elle se base sur un échantillon d'une population statistique et permet d'estimer la densité en tout point du support. En ce sens, cette méthode généralise astucieusement la méthode d'estimation par un histogramme.

Dans le classifieur à fenêtre de parzen, nous estimons la densité de probabilité de chaque classe et classifions les exemples test par la classe correspondant le plus.

Expérience 1 :

Dans cette expérience nous classifierons nos données brute et données centrées réduite a l'aide d'un windows parzen classifieur. Le noyau choisi sera le noyau gaussien dont la formule est la suivante :
.

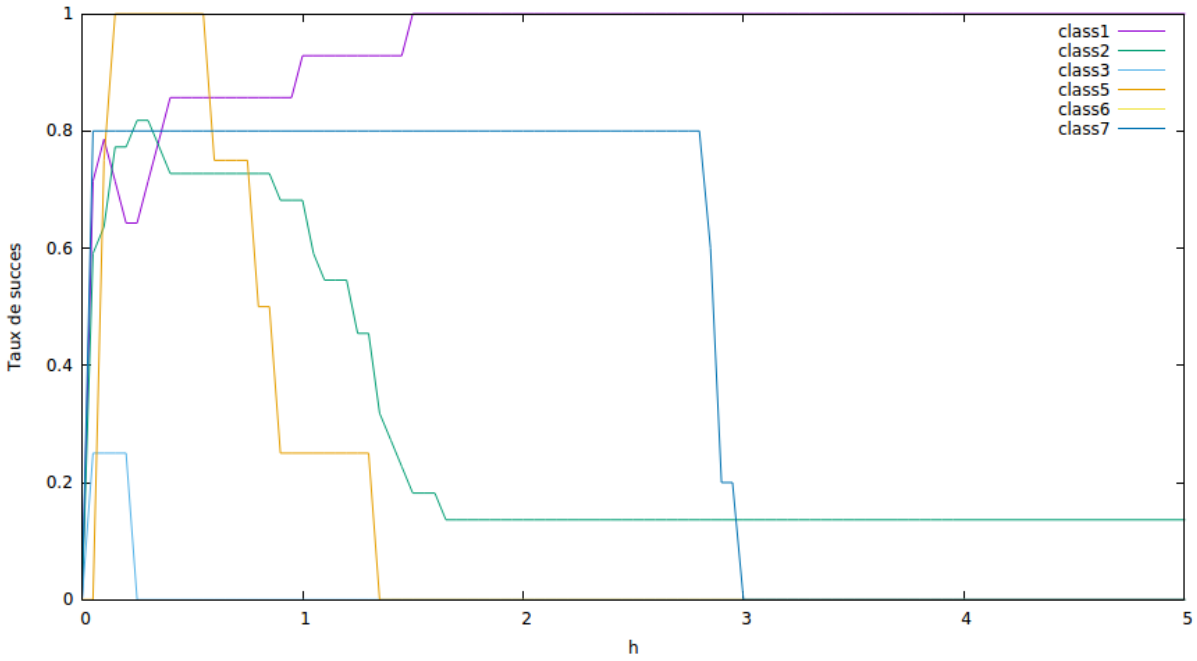
Enfin nous ferons varier le h pour visualiser l'évolution du taux de succès en fonction de celui-ci



Résultat :

FIG 1- Evolution du taux de succès de la méthode de parzen selon le h

Quand nous faisons varier le h nous remarquons qu'au-delà d'une certaine valeur plus nous augmentons celui-ci plus le taux de succès a tendance à décroître et à trouver un plateau en deçà duquel il plongera difficilement. Les meilleures valeurs sont obtenues pour un h faible (<1,5) .



Sur la courbe ci-dessus nous pouvons constater que la classification des exemples de chaque classe diffère selon le h. Notons que la taux de succès obtenus pour la classe 1, est le seul qui progresse quand h croît jusqu'à atteindre 100% de succès.

Partie 2 : BIBLIOTHÈQUE FAST ARTIFICIAL NEURAL NETWORKS

Dans cette partie nous utiliserons la librairie FANN retrouvable à l'adresse suivante leenissen.dk/fann/wp/. Cette librairie laisse à disposition de son utilisateur principalement 4 paramètres de MLP, il s'agit du nombre de couche, du nombre de neurones par couche cachée, du nombre d'époque et de l'erreur désirée.

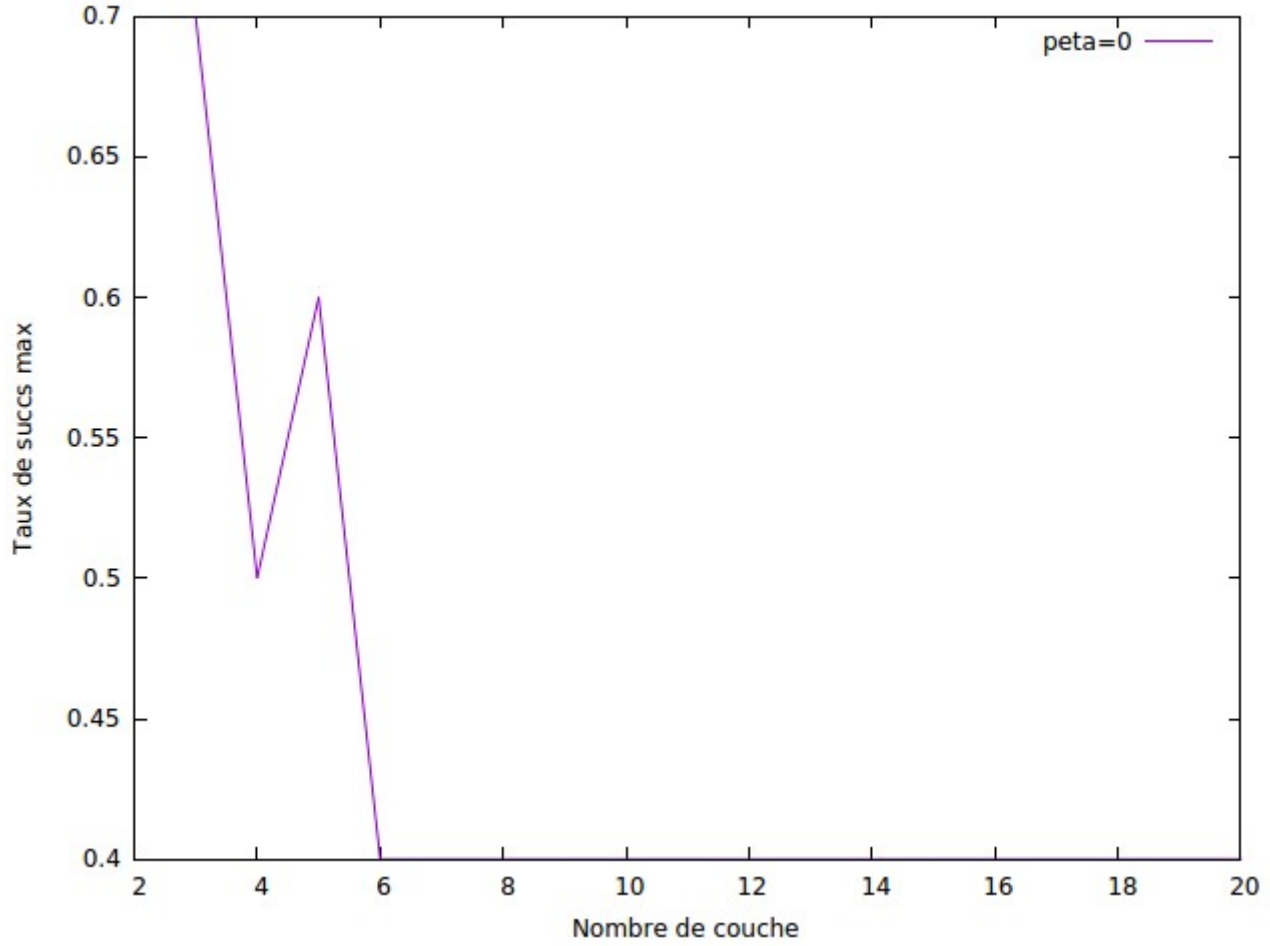
-l'erreur désirée définit l'erreur cible en apprentissage (int). Plus elle est basse, plus le réseau sera entraîné. Mais si elle est trop basse, on risque le sur-apprentissage !

Taux de succès par rapport au nombre de couche

Expérience :

Nous observerons dans cette expérience le comportement du réseau de neurone vis-à-vis du nombre de couche qui le constitue. Pour cela nous incrémenterons progressivement son nombre de couche.

Résultat :



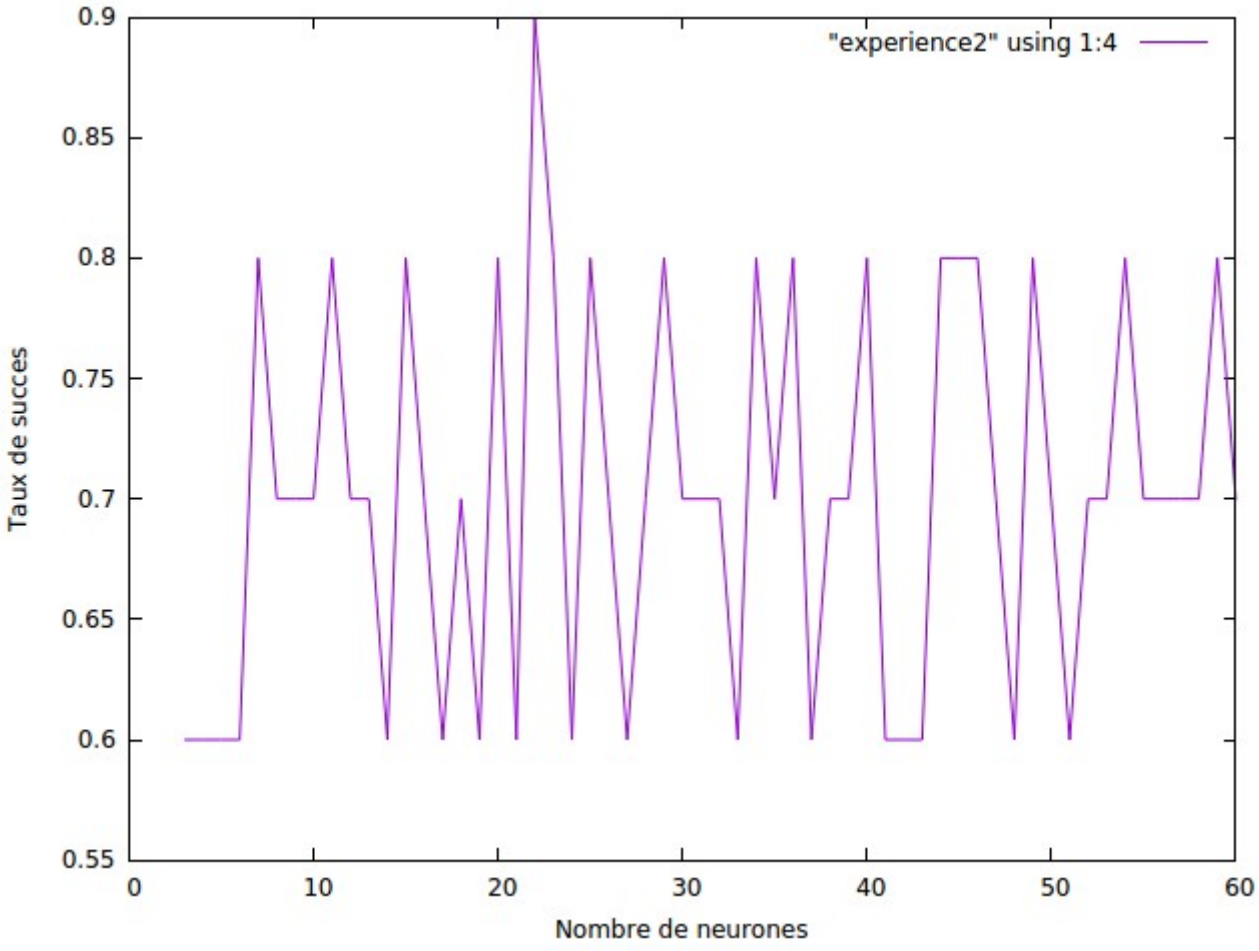
La figure ci-dessus nous montre sans surprise que le taux de succès diminue drastiquement avec le nombre de couche, nous atteignons 0,7 % avec un couche caché de 5 neurones. Remarquons qu'au-delà de 6 couches cachées le taux de succès semble se stabiliser à 0,4 et ne plus changer avec les couches supplémentaires.

Taux de succès par rapport au nombre de neurones cachés sur un réseau avec une seule couche cachée.

Expérience :

Dans cette expérience nous incrémenterons progressivement le nombre de neurone présent sur un réseau à une trois couche afin d'évaluer l'influence du nombre de neurone sur le taux de succès

Résultat :



La figure ci-dessus montre l'évolution du taux de succès en fonction du nombre de neurones sur la couche cachée d'un réseau a 3 couche (1 cachée) . Remarquons que le taux de succès reste compris dans un intervalle compris entre 0,6 et 0,8 sauf une seule et unique fois ou il a atteint 0,9.

Partie 3 : PERCEPTRON MULTI-COUCHE LANGAGE C

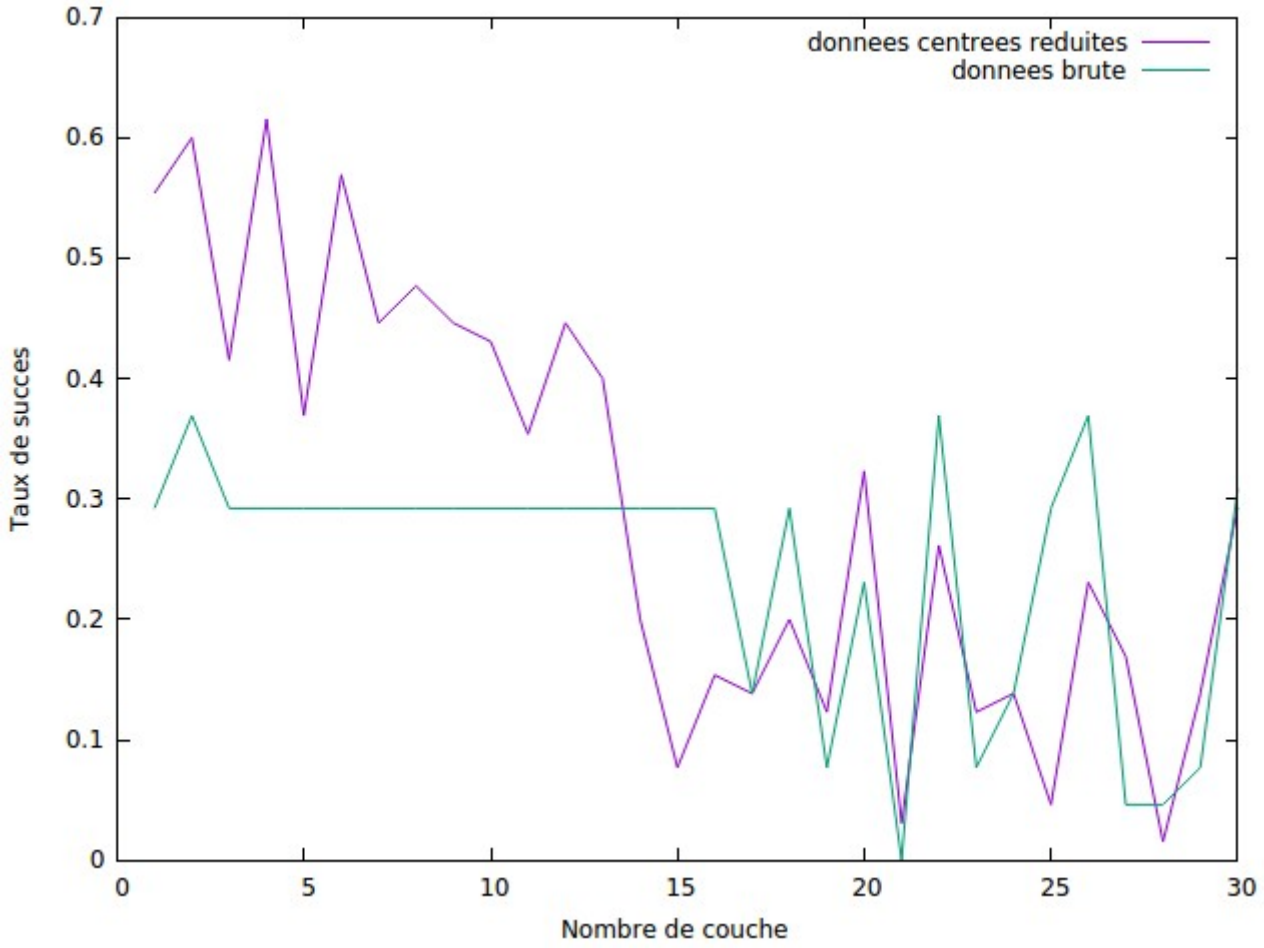
(Mme Paugam-Moisy Code)

Taux de succès par rapport au nombre de couche

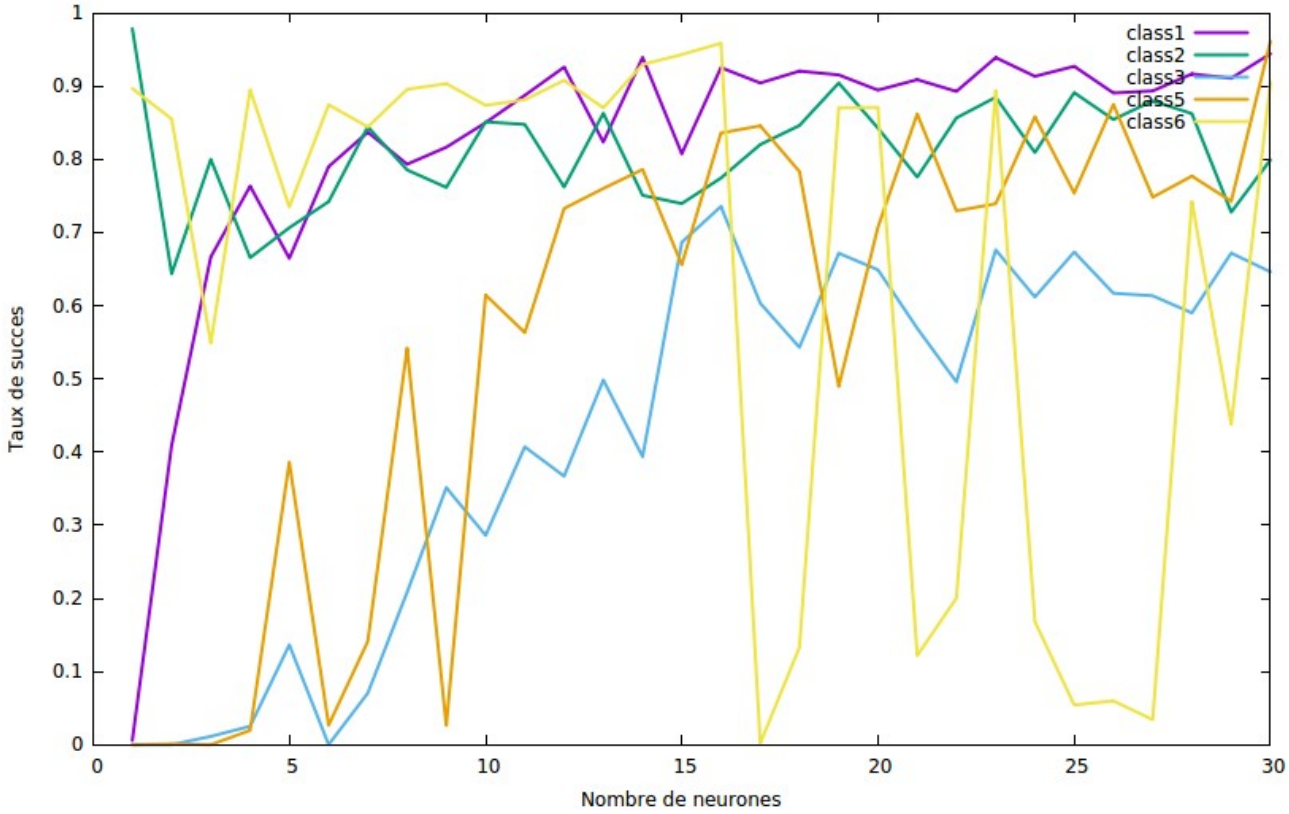
Expérience :

Dans cette expérience nous évaluerons le taux de succès en fonction du nombre de couche que possède le réseau de neurones.

Résultat :

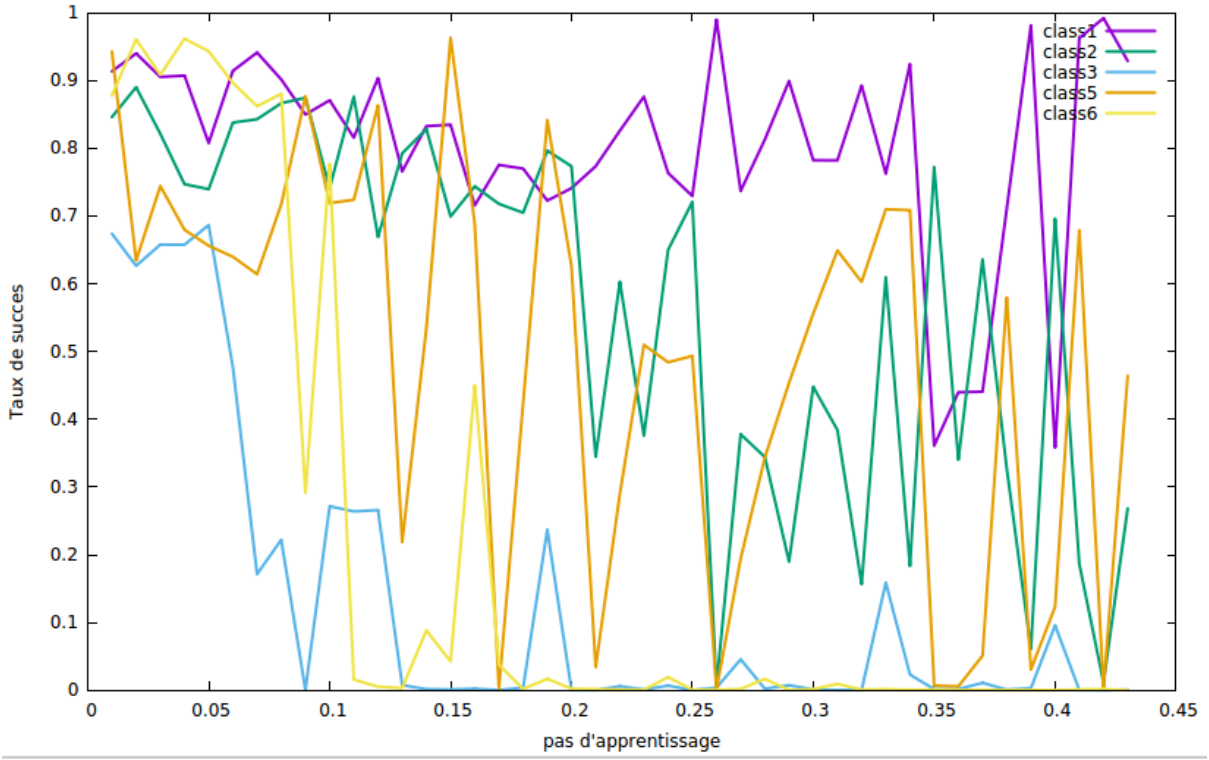


Suite à cette expérience nous remarquons que nous obtenons de bien meilleures performances avec les données centrées réduites qu'avec données brutes. Notons également que le taux de succès diminue fortement quand le nombre de couche croit au point même d'atteindre une performance nulle.



Évolution du taux de succès par rapport au nombre de neurones pour les données centrées réduites

La répartition du taux de succès selon la classe est très chaotique pour des nombres de neurones différents. Ce schéma ne nous permet pas de tirer grande conclusion si ce n'est que pour les classes 1 et 2 la précision du réseau de neurone semble être meilleure. Ces classes seraient donc plus faciles à classifier que les autres. Enfin nous conclurons en disant tout simplement que le nombre de neurone affecte énormément les frontières de séparation des classes que le MLP construit au cours de la rétropropagation du gradient.



Évolution du taux de succès par rapport au pas d'apprentissage pour les données centrées réduites (15 neurones, 1 couche)

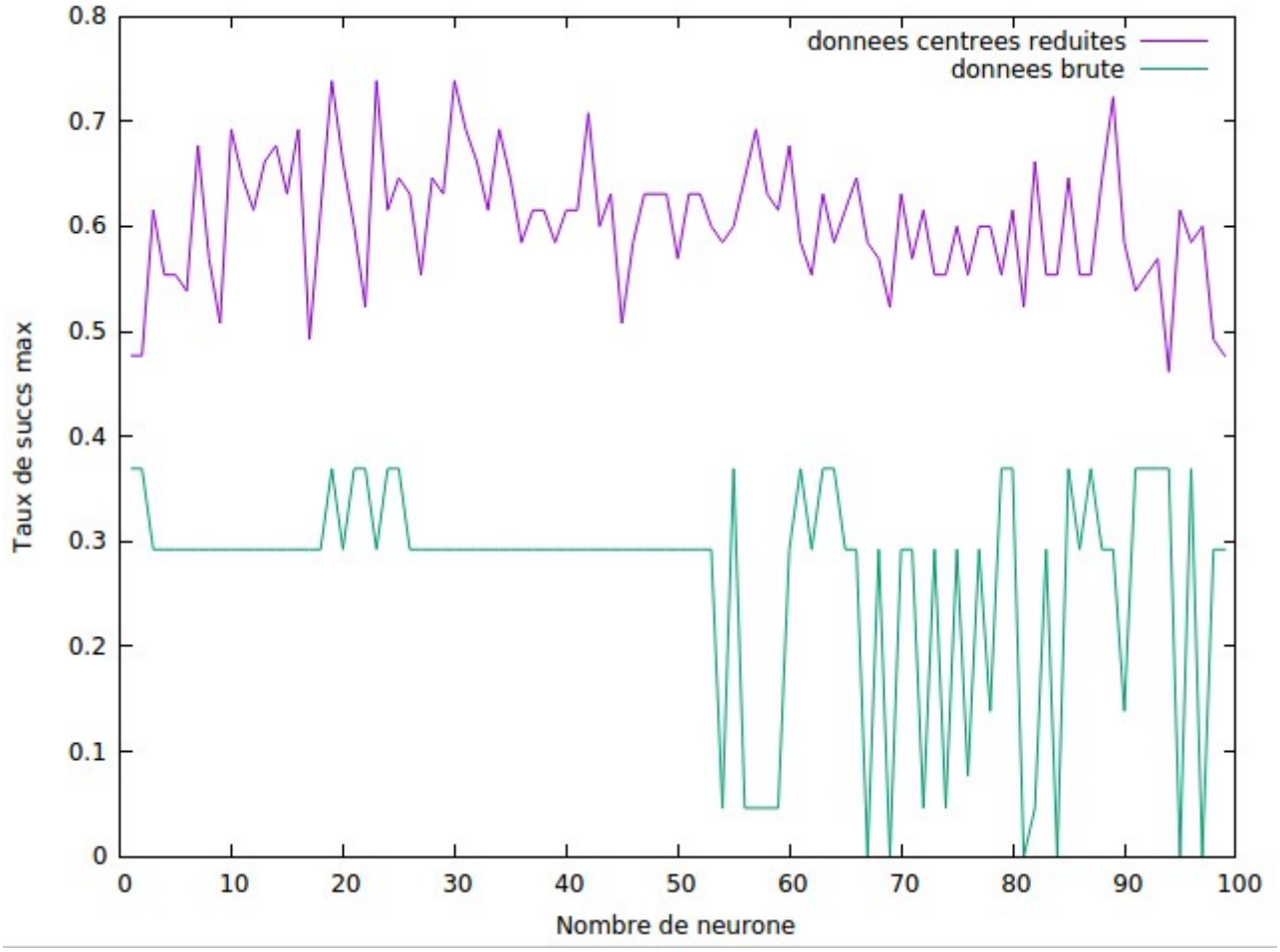
Nous remarquons ci-dessus que l'augmentation du pas d'apprentissage affecte négativement la capacité du réseau à classifier correctement les classes. Les classes 3 et 6 sont celles qui sont les moins bien classées par le réseau lorsque le pas augmente tandis que la classe 1 est celle qui produit le meilleure score.

Taux de succès par rapport au nombre de neurone caché sur un réseau avec une seule couche caché.

Expérience :

Dans cette expérience nous observerons l'évolution du taux de succès vis-à-vis du nombre de neurones présent sur la couche cachée du réseau de neurone.

Résultat :



À la suite de cette expérience nous constatons une fois encore que les données centrées réduites superforment les données brutes. Le taux de succès quant à lui oscille entre 0,5 et 0,7 et nous obtenons un taux de succès maximal de 0.738462 pour 19 neurones.

Taux de réussite par rapport au pas d'apprentissage

Expérience :

Dans cette expérience nous évaluerons l'impact du pas d'apprentissage sur le taux de succès en incrément le pas d'apprentissage.

Résultat :

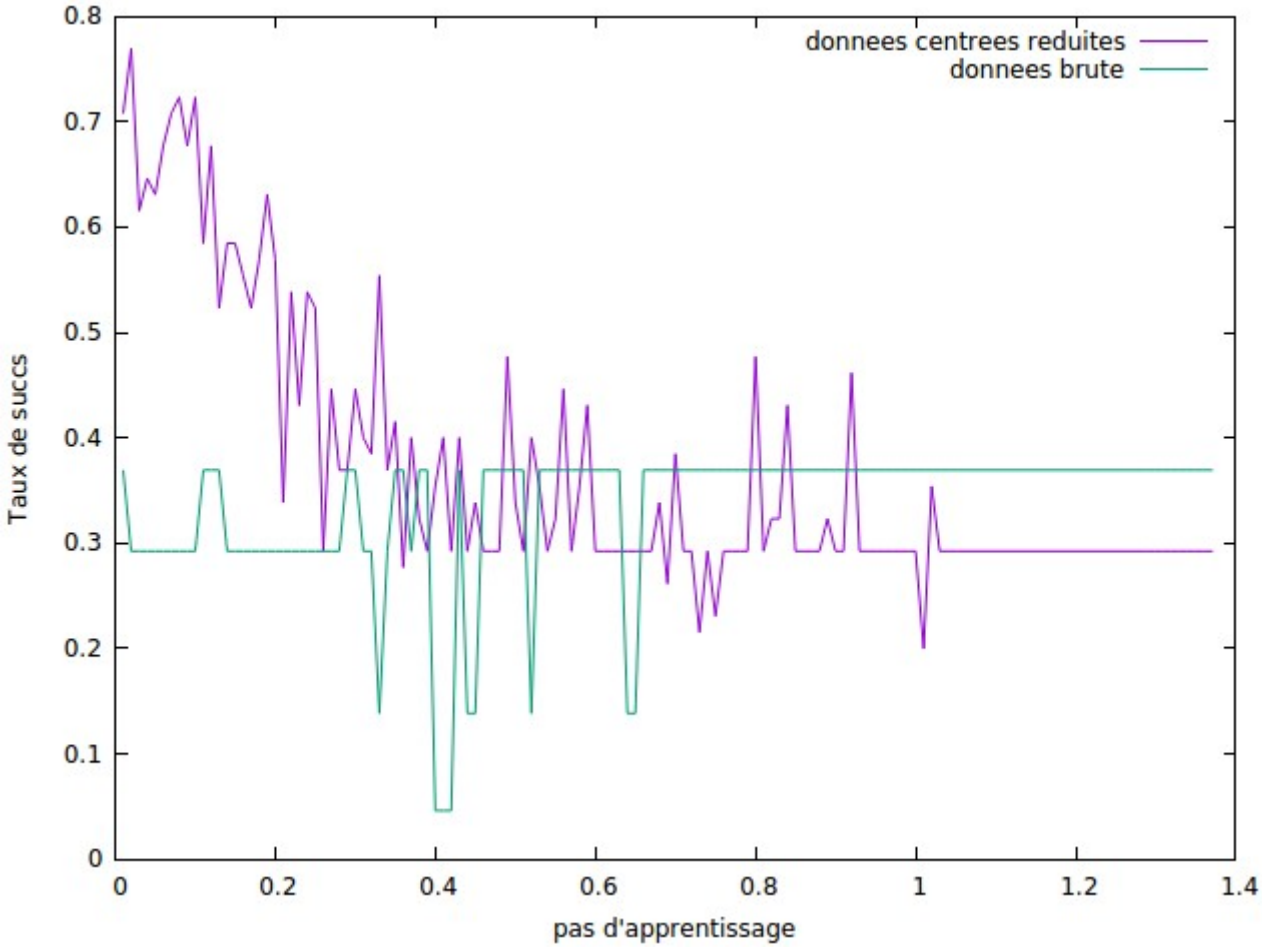


FIG 1- Evolution du taux de succès de la méthode de parzen selon le h

Sur la courbe ci-dessus nous voyons que les meilleurs résultats sont obtenus pour les données centrées réduites et un pas d'apprentissage inférieurs à 0,1 au-delà les résultats faiblissent jusqu'à trouver un point plateau de 0,3 pour les données centrées réduites et 0,4 pour les données brutes.

Partie 4: Support Vector Machine (Lib SVM)

Lib SVM est une librairie opensource qui implémente des supports vector machine pour la classification.

Dans les expériences qui suivront, nous utiliserons pour valider nos résultats la k-fold cross validation déjà implémenté dans la libSVM. De ce fait nous utiliserons la totalité des expériences pour ce qui suivra.

1-Influence du noyau

Expérience :

Dans cette expérience nous essayerons tous les noyaux proposées par la lib svm afin de connaître le noyau qui s'adapte le mieux à notre jeu de données.

Résultat :

0 -- linear: u'*v 64.5%
1 -- polynomial: (gamma*u'*v + coef0)^degree 53%
2 -- radial basis function: exp(-gamma*|u-v|^2) 69%
3 -- sigmoid: tanh(gamma*u'*v + coef0) 43.5%

3-Influence de l'hyper-paramètre gamma

Expérience :
Dans cette expérience nous augmenterons gamma afin de voir s'il existe une valeur tel que le profit soit maximal.

Résultat :

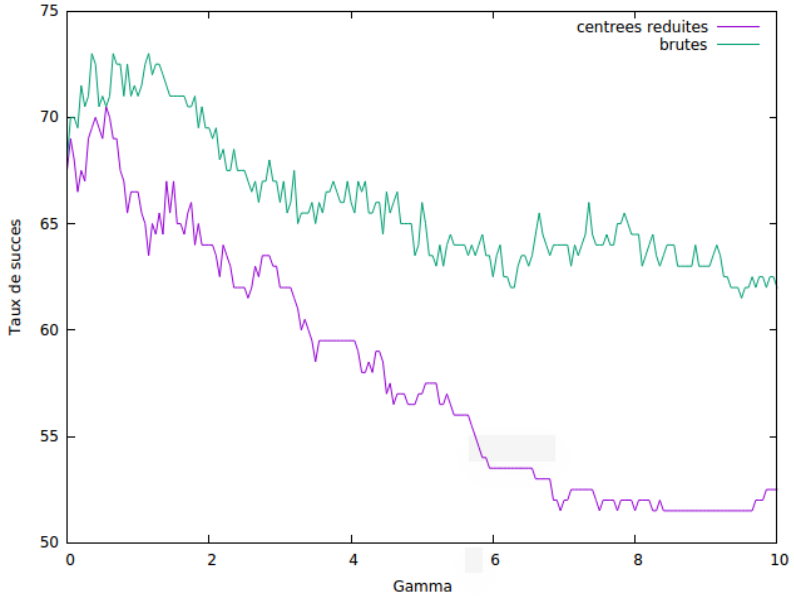


FIG 1- Evolution du taux de succès de la méthode de parzen selon le h

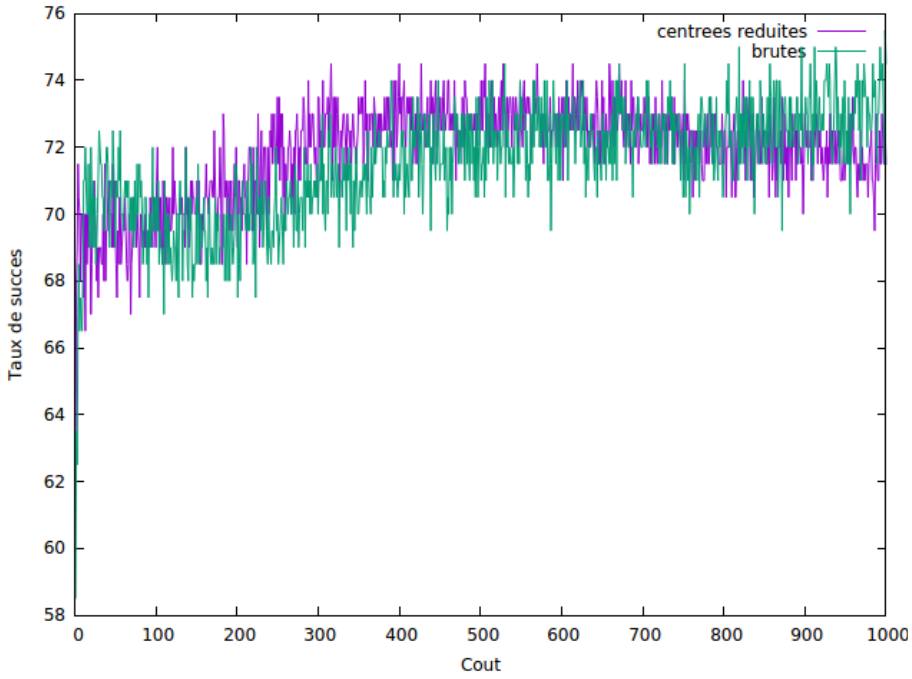
La figure ci-dessus nous montre que l'hyper-paramètre gamma doit être petit (<2) afin d'obtenir de meilleures performances. Notons que tout au long des itérations de l'expérience nous obtenons toujours de meilleurs résultats avec les données centrées réduites qu'avec les données brutes.

5-Influence de l’hyper-paramètre cost

Expérience :

Dans cette expérience nous augmenterons a valeur de l’hyper-paramètre cost et verrons comment celui-ci influe sur notre taux de réussite.

Résultat :



Sur le graphique de la figure ci-dessus nous voyons que bien que pendant un temps nous ayons obtenus de meilleurs résultats avec les données centrées réduites, c’est avec les données brutes que nous obtenons les meilleurs résultats. Notons que le taux de succès maximal obtenu est 75,5 % pour un paramètre cost de 999.

Test avec le script d’optimisation grid.py.

Grid.py est un script d’optimisation fournie par les créateurs de la libsvm afin de trouver les meilleurs paramètres cost et gamma pour la SVM.

Lorsque que nous testons celui-ci sur nos données brute et centrés réduite, nous obtenons un résultat de 73 % avec gamma = 0.03125 et cost = 128.0 pour les données brutes et gamma= 0.0078125 cost =8192.0 pour les données centrées réduites.

SVM matlab :

Enfin pour finir un test des support vecteur machine a également été effectué avec matlab. Les performances obtenues avec celui-ci était de l'ordre de 66 % avec nos bases d'apprentissage et de test. Notons que matlab nous donne un contrôle bien moindre des paramètres de la svm car celui-ci ne nous permet pas de contrôler les différents paramètres tel que gamma.