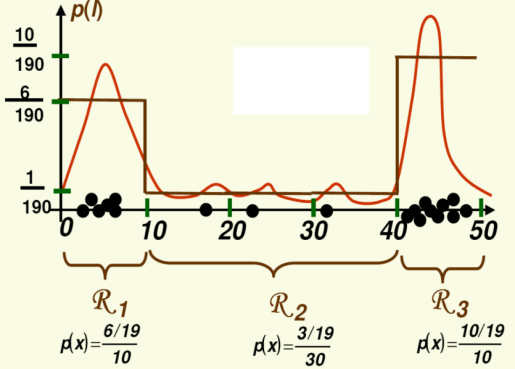
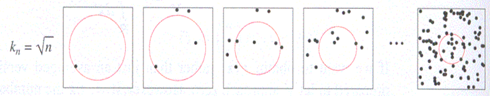
II – Classifieur K-PPV

Le classifieur est K voisins est une méthode non-paramétrique, c’est-à-dire que l’on ne suppose aucune loi de départ sur notre population. Il possède un paramètre, le nombre de voisins qui va influer sur le volume de la boule. La boule représente en fait la fenêtre sur laquelle on va travailler, et augmente jusqu’à contenir les voisins de .

1. b-

Le premier choix qui va influer sur les résultats de l’algorithme correspond à la sélection des bases de données. Pour que le fonctionne, il lui faut une base de validation qui va simuler l’arrivée des données. L’autre base correspond au training donc les données libellées qui va permettre de designer le classifieur.   
Seulement dans la base de training , il faut toujours simuler des données rentrantes dans le cas du pour voir l’impact du que l’on a pris.

C’est pourquoi on divise donc en 2 parties la base de training. Ainsi on se retrouve avec des bases découpées de la façon suivante:

Base de training (choix de ) Base de test (performances de notre classifieur)

|  |  |
| --- | --- |
| (Training)  Données labélisées qui correspond aux données que l’on a collectées | (Test)  Base de test finale qui simule l’arrivée de nouvelles données |
| (Validation)  Qui simule l’arrivée de nouvelles données |

Pour analyser le , nous utilisons 3 paramètres :

* Le taux d’erreur
* Le temps de calcul
* La fonction de densité

Bien que la fonction de densité n’ait pas de sens (densité infinie), elle peut être utile à la compréhension. Par exemple si on veut modéliser , on calcule pour chaque point de notre base de validation la probabilité

C’est le rapport du nombre de points de la classe dans la boule sur le nombre de points de la classe au total sur la boule.

est le nombre de point total de notre étiquette choisie.

Avec le nombre de points qui appartient à l’étiquette voulue, dans la zone définie par la boule.

le volume de l’hypersphère est différent pour chacun des points entrants, et correspond en fait à la largeur de la fenêtre (donc ) **pour une** **modélisation univariée**.   
La densité totale est donnée par la contribution de toutes les probabilités de chaque point.

Attention : Pour la modélisation des classes.   
Si on ne prend qu’une seule dimension il faut faire attention au choix de la distance. Si par la suite on calcule la boule que sur une seule composante, alors le ème voisin n’est pas forcément le plus éloigné :

Ex :

3 plus proches voisins dist eucli entre 2 points de dimension

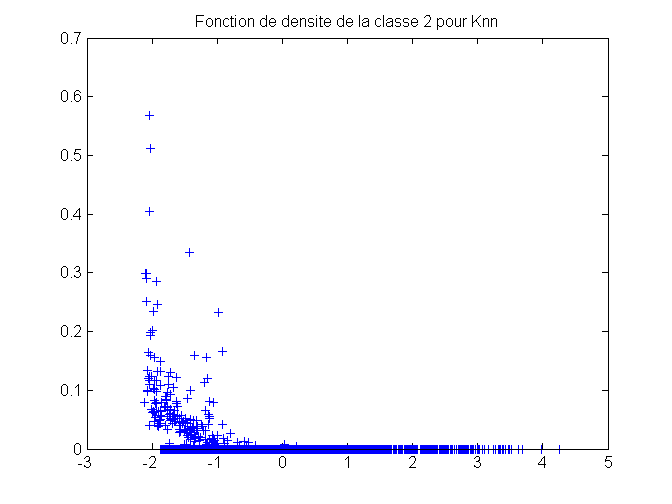
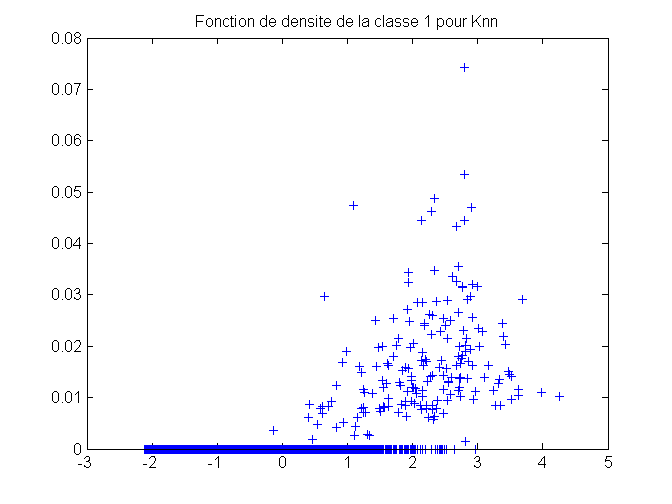
4.0562 4.1335 4.9184

Volume de la boule pour une seule dimension (norme euclidienne) :

0.2474 0.5114 3.1620e-04

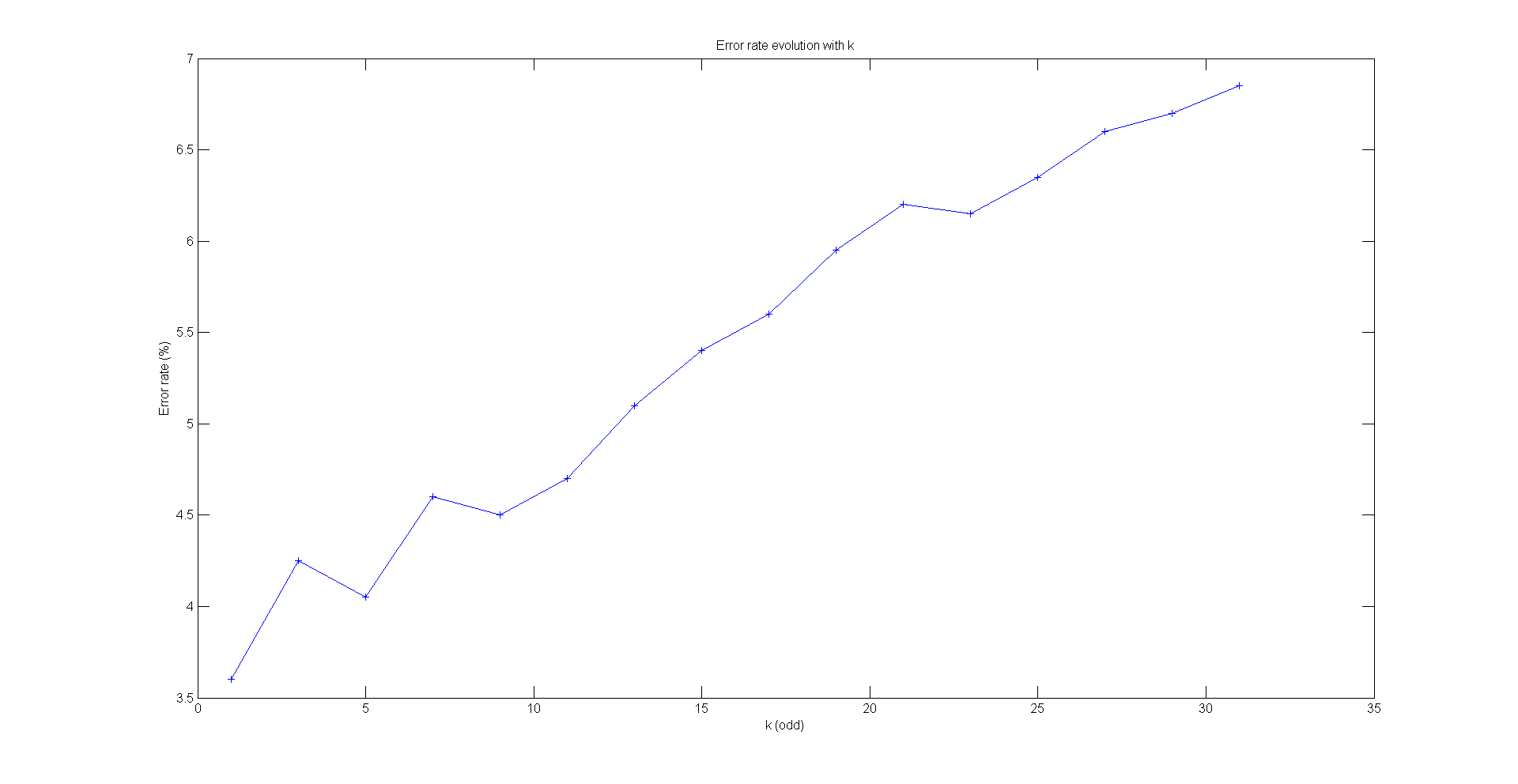
Donc pour la boule le 3ème voisin n’est parfois pas le plus loin (ici c’est même le plus proche) parce que on ne prend en compte qu’une seule dimension ! C’est pour ça qu’il faut prendre le max.

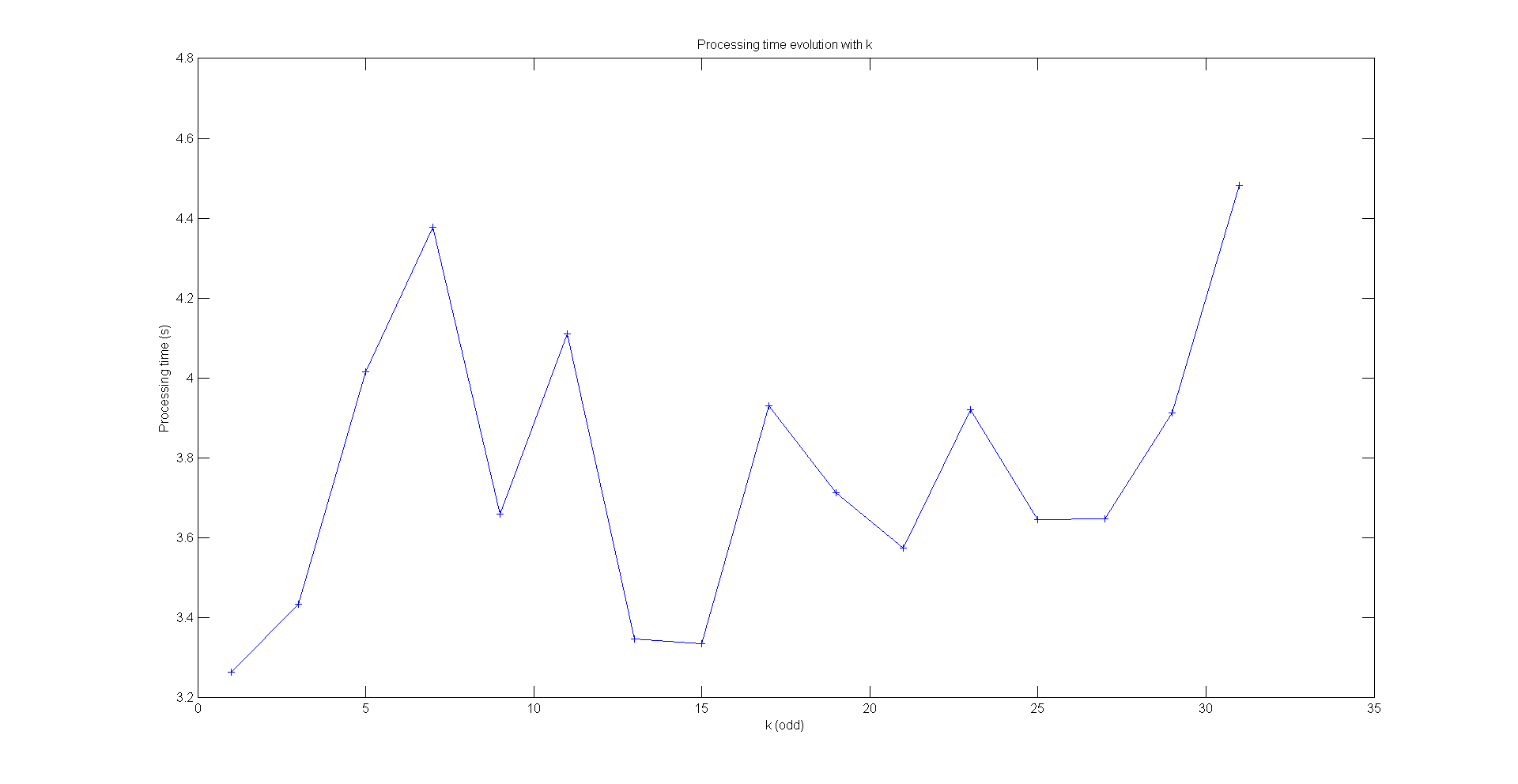
Voici deux exemples de modélisation pour la classe 1 (chiffre « 0 ») et la classe 2 (chiffre « 1 ») :



On remarque que pour le chiffre « 0 », les valeurs projetées par ACP sont plutôt comprise entre et . Alors que pour le chiffre « 1 » on a plutôt des valeurs concentrées entre et .

Voici les tracés du taux d’erreur en fonction de et du temps de process :



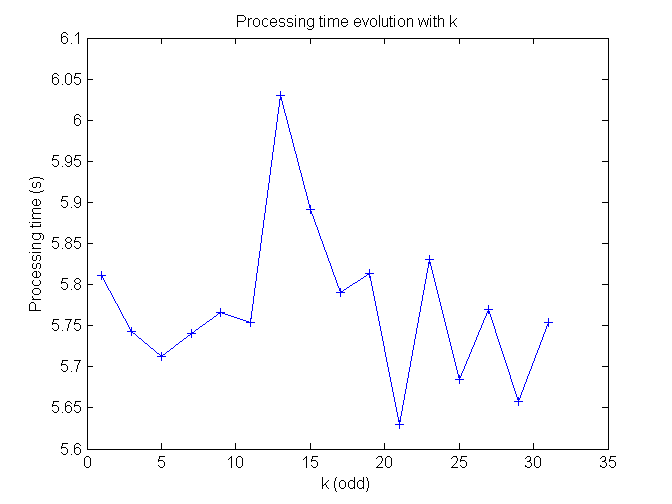
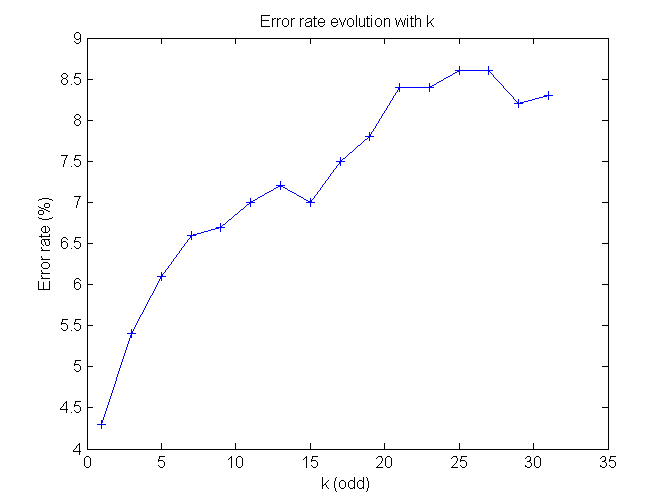


On choisira un , en effet d’après les graphiques semblent plus convenable (taux d’erreur convenable et moins de temps de calcul) seulement cela nous semble trop « risqué ». Il est plus logique de procéder par vote c’est pourquoi on choisira un qui propose un temps de calcul assez faible pour un des plus petits taux d’erreur.

En pratique on sélectionne proportionnel à . Pour choisir on peut soit y allez « au hasard » et comparer les résultats, soit y aller avec une approche de grille ou alors on utilise une sélection de modèles. Un bon choix de est :

c -

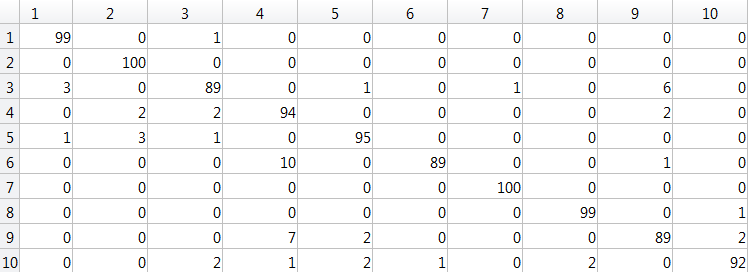
Finalement en fixant on trouve un taux d’erreur de sur la base de test ce qui est convenable, de nos labels sont cohérents. Nous avons décidé de comparer avec d’autres pour voir si l’on a bien choisi celui-ci. Et il apparaît avec le graphique suivant qu’un correspond bien à un taux d’erreur des plus faibles (le graphique supporte donc l’hypothèse que le taux d’erreur augmente avec ).



Le temps de calcul a été estimé à (CPU Intel i5-2410M cadencé à 2 cœurs) qui est un des temps le plus faibles, lorsque l’on compare avec d’autres.

d -

Il est aussi intéressant de déterminer la matrice de confusion, qui nous donne plus d’informations sur les erreurs de classification commises. Sur les colonnes on retrouve la classe que l’on a labellé avec le et en ligne on retrouve les vrais labels de classes.

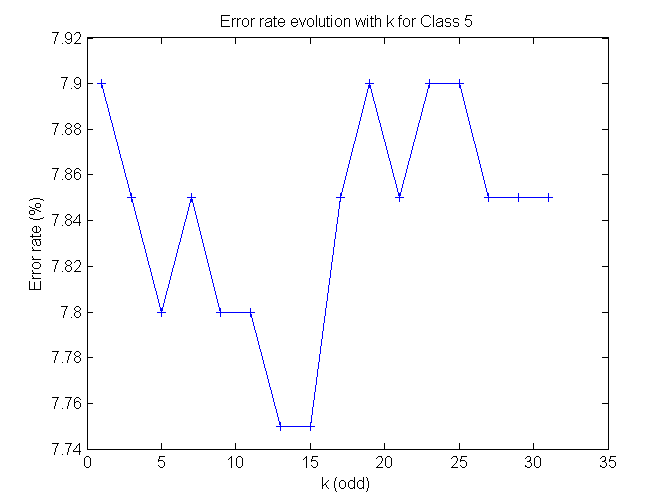
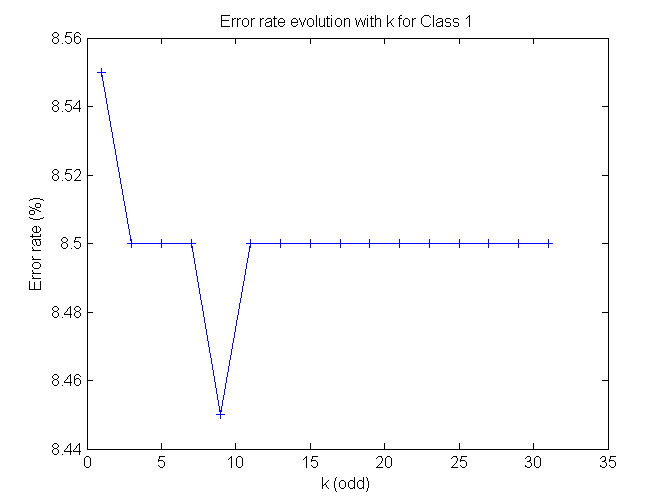


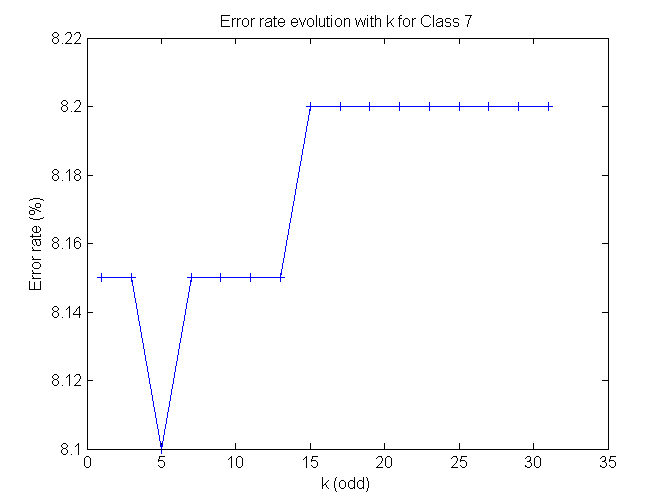
Pour la matrice de confusion, en faisant la somme sur les colonnes, on retrouve ce que l’on classe le plus souvent. On trouve ainsi que l’on a tendance à classifier plus de « 3 » (classe 4) en le confondant avec un « 5 » ou un « 8 ». En effet, ces classes possèdent un recouvrement assez élevé (Lab 1) ce qui expliquer que le les confonde plus facilement. On a tendance à confondre le « 5 » et « 8 » avec un « 3 », ainsi que le « 2 » avec un « 8 ». Ceci s’explique par la ressemblance de ces chiffres.

e -

Pour trouver le optimal pour chaque classe on utilisera la base de training que l’on découpera de la même manière que précédemment. Mais cette fois-ci, la base de test inclus dans la validation n’aura que des labels de la classe que l’on souhaite étudier.

Pour trouver le optimal, nous avons pris en compte le taux d’erreur et aussi quelques fois le temps de process. Voici quelques exemples de graphe que l’on peut retrouver :





Finalement :

Classe 1 :

Classe 2 :

Classe 3 :

Classe 4 :

Classe 5 : (même taux d’erreur que k=15 mais moins de temps de process)

Classe 6 :

Classe 7 :

Classe 8 :

Classe 9 :

Classe 10 :

On remarque que un est assez récurent ce qui réconforte sur le choix de notre pour toute cette partie.

III – SVM

IV - Comparaison des trois classifieurs

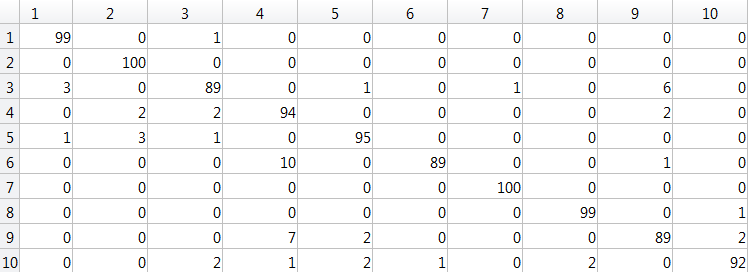
Maintenant que nous avons nos 3 classifieurs, il est nécessaire de les comparer pour savoir lequel utiliser. Pour comparer les classifieurs nous utiliserons 5 critères de choix :

* La capacité de généralisation qui est en fait le résultat de la matrice de confusion.
* Le temps d’apprentissage pour estimer nos paramètres (hyperplan, loi etc…)
* Le temps de test.
* La complexité de l’algorithme correspond à la difficulté de compréhension de la méthode ainsi que le nombre de cycles nécessaire au CPU.
* Enfin, la consommation en mémoire du programme en Octets.

Par la suite, nous avons utilisé un ordinateur comprenant un CPU Intel i5-2410M cadencé à 2 cœurs.

**KPP**

* Capacité de généralisation : 4.9% d’erreur



* Temps d’apprentissage : Ce qui est assez délicat avec l’algorithme des Kpp, c’est qu’il n’y a pas vraiment de temps de process pour la phase d’apprentissage, mis à part le chargement des points. Là où il est nécessaire de calculer les moyennes et covariance pour Bayes, l’hyperplan de décision pour le SVM, le Knn n’a besoin « que » des points directement.
* Temps de la phase de test :
* Complexité de l’algorithme : Moyen, il est nécessaire de calculer les distances euclidiennes entre chacun des points rentrants et les voisins. Ce n’est pas complexe à implémenter mais un peu complexe en termes d’opérations pour le processeur.
* Consommation de la mémoire : Pour ce critère, on ne prendra en compte que les variables les plus gourmandes du programme. On retrouve dans l’ordre

|  |  |  |
| --- | --- | --- |
| **Objet** | **Type** | **Taille** |
| Matrice des distances euclidiennes | Double [] | MB |
| Matrice des distances | Double [] | KB |
| Matrice de la base de test | Double [] | KB |
| Matrice de la base de training | Double [] | KB |
| Vecteur de label de test | Double [] | KB |
| Vecteur de label de training | Double [] | KB |
| Paramètre du classifieur | Double | B |

Au total, le programme nécessite une allocation en mémoire de MB pour fonctionner.