Exercice 1

Pour cet exercice, vous allez implémenter l'algorithm de K-NN.

Le classificateur des k plus proches voisins (kNN pour k Nearest Neighbor) est un algorithme qui classifie les objets selon leurs proximités aux données d'entraînement. C'est une approche de modélisation simple qui approxime la frontière de décision localement. Il n'y a pas d'entraînement proprement dit et les calculs sont seulement effectués lors de la classification. Comme la Figure 1 le démontre, la classification d'un exemple est effectuée en choisissant la classe la plus présente parmi les plus proches voisins. Bien que plusieurs mesures de distance puissent être utilisées pour définir le voisinage, la mesure la plus utilisée est la distance Euclidienne, définie par : \begin{equation*} distance(p,q) = \sqrt{\sum_{i=1}^{d}(p_i - q_i)^2} \end{equation*}

où p et q sont deux points dans un espace R^d

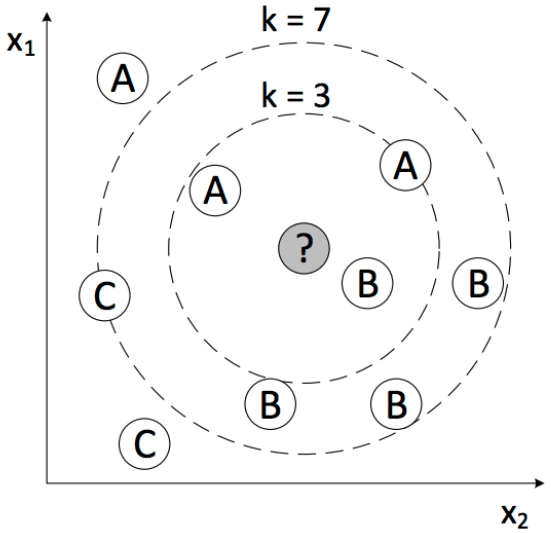

Figure 1 - Exemple de voisinage en utilisant 3 voisins et 7 voisins. Si on ne considère que 3 voisins on attribuera la classe A alors que si on considère 7 voisins, on attribuera plutôt la classe B.

Normalement, un KNN ne génere pas de probabilité, pour cette exercice on va modifier la fonction de classification pour introduire la notion de probabilité. \begin{equation*} proba = n/k \end{equation*} avec n = le nombre de voisins de la classe considérée et k le nombre total de voisins considérés

Le classificateur kNN est une méthode non-paramétrique qui ne nécessite pas d'établir une hypothèse au préalable sur la nature des distributions de données (contrairement à une régression linéaire, par exemple). Le seul paramètre à déterminer est la taille du voisinage (k). On choisit ce paramètre à partir des données. Tel qu'illustré à la Figure 2, une grande valeur de k réduit l'effet du bruit sur les données, mais définit des frontières de décisions sans tenir compte de particularités locales.

1. Charger les données dans le fichier 'ex1.data.mat' en utilisant scipy.io
2. Séparer les données en une partie d'entrainement, une partie test. Mettre ces données dans quatres variables: XTrain, XTest, YTrain, YTest en utilisant sklearn.model_selection.train_test_split (le pourcentage de séparation est 0.5)
3. Implémenter un K-nn permettant de utiliser n'importe quel k en paramètre, vous pouvez utiliser l'architecture fourni ou l'implémentez vous même
4. Calculer le taux de dététection pour 1-NN,2-NN jusqu'à 10-NN

5. Quelle est votre conclusion ?

In [29]:
```
class knn(object):
    def __init__(self, k):
        """
        k is the number of nearest neighbor that we are taking in acc
ount
        """

    def loadTrainData(self, XTrain, YTrain):
        """
        Load training data into the classifier
        """

    def distance(self, pointA, pointB):
        """
        Compute the distance between pointA and pointB
        """


    def classify(self, oneTestX):
        """
        Classify one data
        return its class
        """
```

Exercice 2

Pour cet exercice, vous allez classifier les iris.

On a mesuré les sépales et les pétales en centimètre de plusieurs fleurs qui ont été identifiées par un expert. Les données sont arrangées en matrice où une ligne correspond à une fleur (instance) et les colonnes sont les mesures suivantes :

1. longueur du sépale en cm
2. largeur du sépale en cm
3. longueur du pétale en cm
4. largeur du pétale en cm

Les espèces (classes) sont représentées par des chiffres de 1 à 3 correspondant à :

1. Setosa
2. Versicolor
3. Virginica

Ces données se trouvent dans 'ex1.data.mat'.

1. Chargez les données
2. Séparez les données en train et test en utlisant train_test_split
3. Entrainez un SVM (vous pouvez utiliser le svm de sklearns) linéaire sur les données d'entrainement
4. Calculez le taux de classification sur les données de test
5. Utilisez la validation croisée (10-folds) sur les données d'entrainement, vous pouvez utiliser cross_val_score de sklearn.mode_selection, afficez le score moyen et la déviation standard.
6. Testez les valeurs de C = [0.01 0.1 1 10 100 1000 10000 100000] pour la validation croisée et affichez le score moyen et la déviation standard
7. Tracez les résultats moyens obtenus pour chacune des configurations
8. Choisissez la meilleure configuration et utilisez-la pour classifier les exemples de la base de test.
9. Rapportez l’erreur de classification obtenue sur la base de test et comparez-la avec celle obtenue en validation.
10. Commentez ces résultats

Exercice 3

Les données de cet exercice ont été générées de manière artificielle.

1. Charger les données de 'ex2.data.mat'
2. Affichez ces données à l'aide de la fonction scatter. Que remarquez-vous ?
3. Utilisez le SVM linéaire de l'expérience précédente pour classifier les données. Il vous faut bien sûr optimiser le paramètre C pour ces nouvelles données.
4. Utilisez la validation croisée et rapportez dans un graphique les résultats moyens obtenus pour chaque configuration.
5. Obtenez l'erreur de classification sur la base de test. Combien y-a-t-il de vecteurs de support?
6. Calculez la borne supérieure de l'erreur espérée en test (voir vos notes).
7. Affichez les vecteurs de support par-dessus les données à l'aide de la fonction scatter.
8. Utilisez maintenant un noyau gaussien avec votre SVM afin d'obtenir de meilleurs résultats de classification. Il vous faut optimiser conjointement le paramètre C et le paramètre γ qui contrôle la dispersion du noyau gaussien.
9. Utilisez la validation croisée et rapportez dans un graphique les résultats moyens obtenus pour chaque configuration.

Comme on veut rapporter les configurations conjointes de C et γ, ce graphique est maintenant une surface obtenue à l'aide de la fonction surf.

1. Obtenez l'erreur de classification sur la base de test. Combien y-a-t-il de vecteurs de support?
2. Calculez la borne supérieure de l'erreur espérée en test (voir vos notes). Affichez les vecteurs de support par-dessus les données à l'aide de la fonction scatter.
3. Que constatez-vous en comparant les deux résultats finaux? Commentez.
4. Que constatez-vous en comparant le nombre de vecteurs de support? Commentez.

Exercice 4

Vous voulez concevoir un système qui trie automatiquement les types de verre afin d'en effectuer le recyclage. Il s'agit d'un exemple fictif avec des données réelles. Vous pouvez prendre les mesures suivantes sur le verre :

1. L'indice de réfraction Pourcentage du poids en oxyde de :
2. Na: Sodium
3. Mg: Magnesium
4. Al: Aluminum
5. Si: Silicon
6. K: Potassium
7. Ca: Calcium
8. Ba: Barium
9. Fe: Iron

Les types de verre reconnus sont :

1. Fenêtres de bâtiment (float processed)
2. Fenêtres de bâtiment (non float processed)
3. Fenêtres de véhicule
4. Lampes
5. Contenant
6. Vaisselle

Vous voulez déterminer quel serait le meilleur modèle entre un SVM avec un noyau Gaussien et un kNN pour cette tâche. Afin d'avoir des résultats plus fiables, vous voulez utiliser un protocole expérimental différent du protocole hold out tel que fait jusqu'à maintenant. On dit hold out lorsqu'on réserve une partie des données pour le test. Quand il y a peu de données dans la base, il peut arriver que les données contenues dans la base de test représentent mal le problème ou que les données utilisées pour s'entrainer soient insuffisantes. Incidemment, les résultats obtenus peuvent être trompeurs et mener à faire une mauvaise sélection de modèle. Afin d'éviter ce problème, vous voulez tester avec le plus de base de test différentes possible et utiliser un maximum de données d'entrainement. Pour ce faire, vous utiliserez une fois de plus la validation croisée. Par contre, cette fois, la base de test fait partie de la validation croisée. Vous commencerez par diviser aléatoirement vos données en 10 parties. Chacune des parties doit contenir les mêmes proportions pour chacune des classes. Commencez par réserver une partie comme base de test. Les 9 autres parties constituent votre base d'entrainement. À partir de cette base d'entrainement vous devez optimiser les paramètres de vos classificateurs, tel que fait à l'expérience 2. Vous vous trouvez donc à faire une validation croisée à l'intérieur de votre validation croisée. À la fin, vous obtiendrez 10 mesures de performances, une pour chaque base de test. Faites la moyenne afin d'obtenir votre performance finale. Dans votre rapport, illustrez le protocole expérimental décrit ci-haut. Vous pouvez utiliser une image, un organigramme, un diagramme bloc ou du pseudocode. Tracez les résultats obtenus en validation pour le SVM et le kNN. N'illustrez pas les 10 courbes obtenues, faites plutôt une moyenne. Quel classificateur choisiriez-vous pour cette tâche? Quelles sont les résultats que vous avez utilisés pour faire votre choix. Outre le taux d'erreur, qu'est-ce qui pourrait motiver votre choix?

Dans cette expérience, on veut détecter les bactéries qui possèdent une protéine particulière (science…). Nous appellerons cette bactérie la « vlimeuse ». Ces bactéries ont été prélevées dans de la nourriture pour nourrissons. Si ingérée, la bactérie vlimeuse rend le nourrisson très malade. Pour chacune des bactéries, on a appliqué plusieurs méthodes de mesure propres au domaine de la biologie qu'il est inutile de décrire ici.

Optimisez un classificateur SVM avec un noyau linéaire, avec un noyau Gaussien et avec un noyau polynomial. Optimisez aussi un classificateur kNN et un réseau de neurones simple couche avec la fonction d'activation logistique : \begin{equation} *f(x) = \frac{1}{1 + e^{-x}}* \end{equation}

Vous devez utiliser la validation croisée pour optimiser le paramètre T. Vous devriez donc avoir 5 classificateurs différents. Expliquez le protocole que vous avez utilisé comme pour les expériences précédentes. Rapportez les paramètres que vous avez utilisés pour chacun des classificateurs. Comparez l'erreur de classification obtenue sur la base de test pour chacun des classificateurs. Quel est le meilleur classificateur selon cette métrique? À votre avis est-ce que dans ce type de problème, un faux positif a le même impact qu'un faux négatif? Devraient-ils être comptés de la même manière? Expliquez. Tracez les courbes ROC pour chacun des classificateurs sur un même graphique. Choisissez le meilleur classificateur. Comment avez-vous choisi ce classificateur. Quelle proportion de bactéries vlimeuses détecterait chacun des classificateurs si on acceptait 10% fausses détections. Choisiriezvous le même classificateur? Choisiriez-vous le même seuil de décision? Commentez.

In [ ]: