RAPPORT DE CONCEPTION D'UN ALGORITHME

K-NEAREST Neighbors (KNN)

Réalisé par Pierre Pétillion

ESILV
ENGINEERING SCHOOL
DE VINCI PARIS

ESILV
LÉONARD
DE VINCI

ÉCOLE
D'INGÉNIEURS
PARIS-LA DÉFENSE

Table des matières

Introduction

Dans le cadre d'un projet du second semestre au sein de l'ESILV, nous avons dû développer un algorithme représentant la méthode des k plus proches voisins. L'objectif est de prédire les classes d'un ensemble de données grâce à un autre dataset.

Méthode de résolution initiale

Principe de mon algorithme initial :

- Les fichiers 'data', 'preTest' et 'finalTest' sont récupérés dans des DataFrame.
- La meilleure valeur « k » est déterminée par la prédiction du jeu de données preTest grâce à l'analyse du contenu du DataFrame data.
- Une fois le meilleur « k » trouvé, il est utilisé pour la prédiction du jeu de données finalTest que l'on enverra par la suite dans un autre fichier nommé 'result.txt'.

Choix effectués pour cet exercice :

- Evaluation du meilleur k possible compris entre 1 et 15.
- Utilisation des DataFrame.
- Conservation des prédictions du fichier preTest pour le calcul du fichier finalTest

Problème rencontré :

```
Debut Calcul ratioPrediction k =  2  : OK
nombrelementIncorrect :  238 compteurVarConnu 803
Predicted  classA  classB  classC  classD  classE
Actual
classA        262       4       6       0       0
classB        127     198       1       3       1
classC         36       0     475       4       0
classD         36       3      17     332       0
classE          0       0       0       0     101
Ratio : 70.36114570361146
Fin Analyse du k =  2  : OK
```

Le ratio du meilleur « k » restait très faible (entre 68 et 75%). Or nous avons vu en cours que nous devions obtenir un ratio d'environ 90%. J'ai donc réalisé une multitude de tests et voici la méthode de résolution que j'ai fini par adopter.

Méthode de résolution Retenu

Solution retenue :

- Récupération des fichiers data, preTest et finalTest dans des DataFrame.
- Détermination de la meilleure valeur « k » possible en fusionnant les jeux de données de data et preTest afin d'améliorer l'apprentissage mais en masquant certains résultats définis grâce à un pourcentage de chance. On essaye ensuite de les prédire puis on calcul le ratio de chacun des « k ».
- Une fois le meilleur « k » trouvé, il est utilisé pour la prédiction du jeu de données finalTest que l'on enverra par la suite dans un fichier nommé 'result.txt'.

Choix effectués pour cet exercice :

- Choix du meilleur « k » possible compris entre 1 et 15.
- Utilisation des Dataframe.
- Choix d'un taux de 75% pour l'apprentissage des données de data et preTest.

```
wdir='C:/Users/petil/OneDrive/Bureau/Etude/ESILV 2020/S
Lancement de l'IA : OK ...
Debut Analyse du k =  1  : OK
Debut Calcul ratioPrediction k =  1  : OK
nombrelementIncorrect :  40 compteurVarConnu 1221
Predicted  classA  classB  classC  classD  classE
Actual
classA        256      12       4       0       0
classB          6     321       2       1       0
classC          3       0     509       3       0
classD          1       5       3     379       0
classE          0       0       0       0     101
Ratio : 89.6103896103896
Fin Analyse du k =  1  : OK
Debut Analyse du k =  2  : OK
```

On peut remarquer un ratio nettement supérieur à celui obtenu avec la méthode initiale.

Processus d'importation de données

```python
#Return le dataframe du csv fournit dans le path
def importDataWithDataFrame(path):
    df=pd.read_csv(path)
    if(df.axes[1][0] != '0'):
        df = pd.read_csv(path,header=None)
        df.columns = df.columns.map(str)

    df["prediction"] = "None";
    return df
```

La première étape du projet était d'appeler la fonction 'importDataWithDataFrame' prenant comme argument un chemin d'accès et renvoie un DataFrame.

Chaque DataFrame contient un jeu de données mais également une colonne supplémentaire nommée «prédiction» qui permettra de stocker la prédiction de l'algorithme.

Processus d'apprentissage des datas

On commence par choisir aléatoirement les données de data et preTest que nous devrons prédire.

```python
def choixValApprentissage(data):
    global compteurVarConnu
    newdata = copy.deepcopy(data);
    newdata["prediction"] = "None";
    for i in newdata.index:
        if(rd.random() < ratioValApprentissage):
            newdata.loc[i,"prediction"] = newdata.loc[i,str(len(newdata.columns) - 2
            compteurVarConnu = compteurVarConnu + 1
    return newdata;
```

```python
def calculEuclidien(i,j,data):
    res = 0;
    for k in range(0,(len(data.columns) - 2)):
        res = res + (data[str(k)][i] - data[str(k)][j])**2

    return [math.sqrt(res),data['prediction'][j]];

def knn(data,i):
    distancePointTest = []
    for j in data.loc[data['prediction'] != "None"].index:
            distancePointTest.append(calculEuclidien(i,j,data));

    #on trie
    distancePointTest = sorted(distancePointTest);

    return distancePointTest;


def remplissagePoint(data,k):
    newdata = copy.deepcopy(data);
    #pour tout les points qui n'ont pas de prédiction
    for i in newdata.loc[newdata['prediction'] == "None"].index:
            #On calcul distance euclidienne
            knnTab = knn(data,i);
            #On short au k choisi
            FinalChoixTab = [];
            for j in range(0,k+1):
                FinalChoixTab.append(knnTab[j][1]);
            #On determine la classe
            FinalChoixTab = Counter(FinalChoixTab);
            #on l'ajoute a la tab data
            newdata.loc[i, 'prediction'] = list(FinalChoixTab.most_common(1)[0])[0];
    return newdata;
```

Pour chaque point à prédire, on calcule sa distance euclidienne avec tous les points connus puis on définit sa classe grâce à ses « k » voisins.

On vérifie ensuite si la valeur prédite correspond bien à la valeur attendue et on pourra en déduire le ratio de k.

Une fois tous les ratios de k trouvé compris entre 1 et 15, il ne reste plus qu'à sélectionner le k ayant le meilleur ratio de réussite.

Processus de remplissage de finalTest

```
148
149 def predictionFinal(bestApprentissage):
150     newdata = importDataWithDataFrame(pathExecutable);
151     newdata2 = bestApprentissage[2];
152     newdata2 = newdata2.append(newdata, ignore_index=True);
153
154     fichier = open(pathWrite, "w")
155     for i in newdata.index:
156         print(i)
157         #On calcul distance euclidienne
158         knnTab = knn(newdata2,i);
159         #On short au k choisi
160         FinalChoixTab = [];
161         for j in range(0,bestApprentissage[0] + 1):
162                 FinalChoixTab.append(knnTab[j][1]);
163         #On determine la classe
164         FinalChoixTab = Counter(FinalChoixTab);
165         #on l'ajoute a la tab data
166         newdata.loc[i, 'prediction'] = list(FinalChoixTab.most_common(1)[0])[0];
167         fichier.write(list(FinalChoixTab.most_common(1)[0])[0] + "\n")
168     fichier.close()
169     checklabel()
170
```

Une fois la meilleure valeur de k trouvé, on répète le processus précédent sur le jeu de données final et l'on renvoie les valeurs dans un fichier text.

Temps d'exécution du programme

Les tests de rapidité ont été effectués sur un ordinateur portable équipé d'un processeur Intel i7 de 9ème génération. Voici un exemple d'affichage :

```
2999
Labels Check : Successfull!
temps d'execution du programme :  988.65625 s
Fin du programme

In [246]:
```

Le temps émis par mon programme est relativement conséquent. Cela est dû à la façon dont nous parcourons le DataFrame.

Problème rencontré

Ce projet fût de mon point de vue relativement compliqué dû à l'objectif que je mettais fixé d'utiliser que des DataFrame afin de bien comprendre leur fonctionnement.

La première difficulté fut lors des copies de DataFrame. En effet la copie était réalisée comme un pointeur et non comme une variable classique.

Une autre difficulté fut l'ajout d'un DataFrame à un autre. En effet les colonnes devenaient des objets et cela fut assez compliqué à comprendre ainsi qu'à résoudre.

Enfin, le temps de compilation est devenu extrêmement important lors du passage des 'iris.data' à 'data.csv'. Malgré des tentatives d'optimisation, cela n'a malheureusement eu qu'un impact mineur sur le temps de compilation.

Conclusion

Ce projet s'est avéré passionnant et très enrichissant. Il m'a permis de renforcer mes connaissances en Python et d'améliorer ma compréhension du fonctionnement de l'Intelligence Artificielle.