Filière Informatique

Assignement 1

Groupe D :
Tanguy Pemeja

Encadrant :
Aurélie Bugeau

Version du 15 Mai 2020

1 Question 1 - Kmeans

Dans cette partie, nous allons utiliser Kmeans afin de déterminer des centroïdes afin d'identifier les différents chiffres. Dans un premier temps, nous allons nous intéresser à comment nous avons pu utiliser Kmeans afin de résoudre ce problème de classification des chiffres. Puis nous allons regarder les différentes caractéristiques de cette classification et voir si celle-ci est efficace.

1.1 Algorithmique

La classification grâce à Kmeans s'effectue en 2 étapes :

1. Définir les centroïdes
2. Associer à chaque centroïde un chiffre

La première étape consiste donc à créer `10` centroïdes, chacun correspondant à un chiffre, puis à entraîner sur un jeu de données X représentant les chiffres manuscrits. Cet entraînement se fait grâce à la commande `fit` :

```
kmeans = KMeans(n_clusters=10, random_state=0).fit(X)
```

Mais le problème de cette méthode est qu'elle n'associe pas les centroïdes dans l'ordre, c'est-à-dire que le premier centroïde trouvé n'est pas forcément associé au chiffre `"0"` et ça pour tous les centroïdes. C'est pourquoi il faut identifier le chiffre associé à chaque centroïde. Pour cela, on évalue à quelle centroïde est associé chaque chiffre manuscrit, et on définit le chiffre représenté par le centroïde comme étant le chiffre qui apparaît le plus de fois parmi ceux associé à ce centroïde.

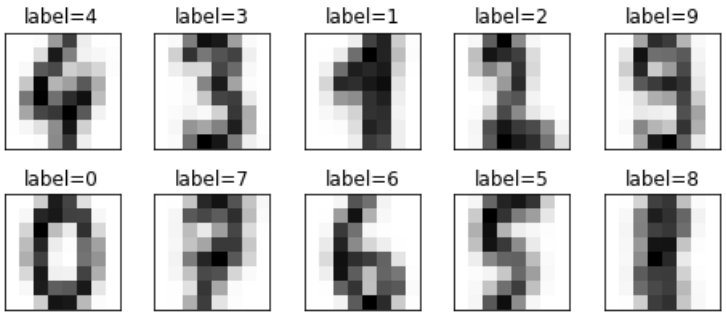

Figure 1 – Association chiffre-centroïde

1.2 Résultats

Suite à l'entraînement de cette classification, nous avons pu évaluer les caractéristiques de celle-ci. Tout d'abord, nous pouvons observer la précision ainsi que le rappel de notre classification pour chacun des chiffres.

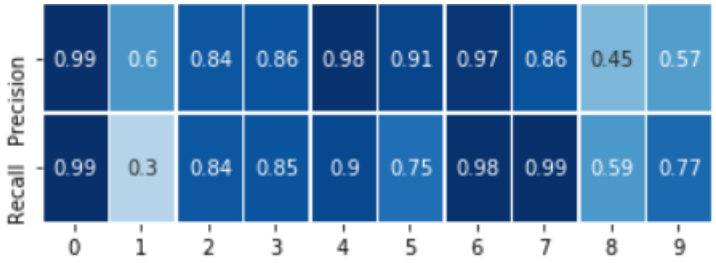

FIGURE 2 – Précision et rappel pour chaque chiffre

Nous pouvons observer que certains chiffres tels que `"0"` et `"6"` sont très bien reconnue par le classifieur. Tandis qu'un chiffre tel que `"1"` a une précision de , ce qui signifie que celui que certains nombres sont confondus avec `"1"`. Et a un rappel de , ce qui signifie que celui-ci est très souvent confondu avec d'autres chiffres.

Grâce à la matrice de confusion ci-dessous, nous pouvons observer que `"4"` et `"9"` sont parfois confondus avec `"1"` ce qui explique sa valeur de précision. Mais on peut surtout observer que `"1"` est souvent confondu avec `"2"` mais surtout avec `"8"`.

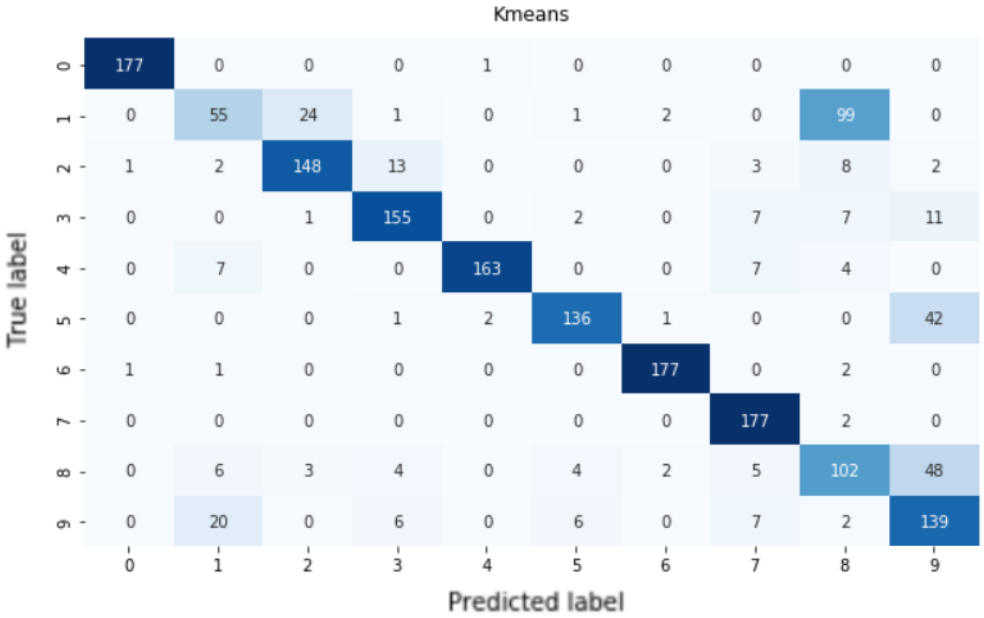

FIGURE 3 – Matrice de confusion

Au final, cette classification obtient précision de `80%`. De plus, sa courbe de précision-rappel montre que la précision est faible lorsque le rappel l'est aussi.

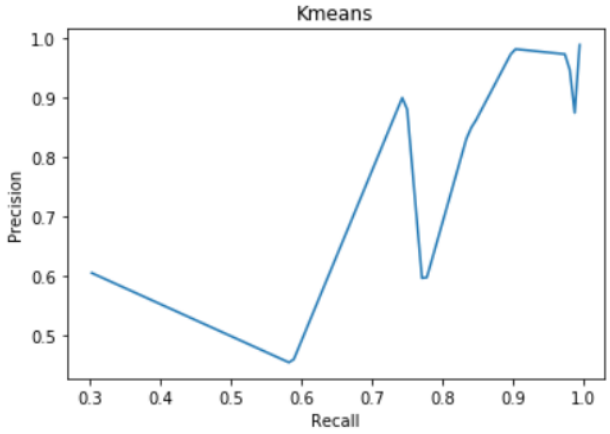

FIGURE 4 – Courbe de précision-rappel

2 Question 2 - Naive Bayes

Dans cette partie, nous allons utiliser une classification probabiliste nommée `"Naive Bayes classification"`. Dans un premier temps, nous allons voir comment utiliser cette classification pour la reconnaissance de chiffres manuscrits. Puis nous allons évaluer les caractéristiques de la classification suivant les différents paramètres choisis.

2.1 Algorithmique

La classification s'effectue en 3 étapes :

1. Définir un set d'entraînement et un set de test
2. Entraîner le classifieur sur le set d'entraînement
3. Déterminer si la classification est correcte sur le set de test

La première étape est de séparer notre jeu de données en 2 sets, un set d'entraînement afin de faire apprendre à notre classifieur à reconnaître les chiffres puis un deuxième set afin d'analyser la capacité de reconnaissance de notre classifieur.
Ensuite, il faut définir le type de distribution pour le classifieur et l'entraîner sur le set d'entraînement .Cet entraînement se fait aussi avec la commande `fit` mais une fois définie la distribution :

```
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB
#Create a Gaussian Classifier
clf = GaussianNB()
# Train the model using the training sets
clf.fit(training_set_X,training_set_Y)
```

Maintenant que le classifieur a été entrainé, il faut pouvoir définir si celui-ci est convenable. Pour cela, on détermine les labels des chiffres manuscrits du set de test grâce à la commande `predict` :

```
y_pred = clf.predict(testing_set_X)
```

Et dès lors, nous pouvons effectuer différentes évaluations de notre classifieur en utilisant la prédiction des labels sur le set de test et les vrais labels du set de test pour calculer des éléments tel que la matrice de confusion, la précision de notre classification, etc.

2.2 Résultats

L'objectif de cette partie est de déterminer la meilleure distribution possible afin de réaliser notre classification de chiffre manuscrit. Nous allons comparer les résultats obtenus entre 2 distributions : la `distribution multinomiale` et la `distribution gaussienne`. Dans un premier temps, nous allons comparer la matrice de confusion de ces 2 distributions :

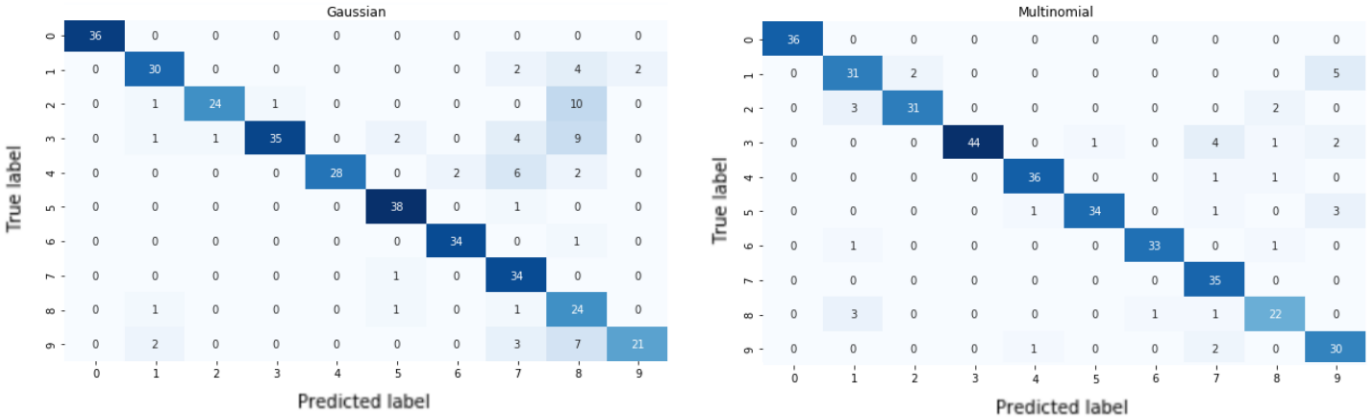

FIGURE 5 – Matrice de confusion pour les distributions gaussienne et multinomiale

En comparant ces deux matrices de confusion, nous pouvons observer que les erreurs de labélisations ne produisent pas avec les mêmes chiffres. Néanmoins, la matrice de confusion de la distribution gaussienne a des erreurs avec des valeurs plus importantes notamment avec la labélisation de `"8"` qui se retrouve prédite pour d'autres nombres comme `"2"`, `"3"` et `"9"`. Ce nombre plus grand d'erreur se retrouve lorsqu'on calcule la précision de nos 2 classificateurs.

Distribution	Gaussienne	Multinominale
Précision	82%	90%

Pour une classification avec une distribution multinominale, la précision est meilleure. De plus en étudiant la courbe précision-rappel de ces 2 distributions, nous pouvons dire qu'une distribution multinominale est préférable dans le cas de la classification des chiffres manuscrits. En effet, la courbe de cette dernière est au-dessus quelle que soit la valeur de rappel. Ainsi pour une même valeur de rappel, la précision est meilleure pour une classification basée sur une distribution multinominale.

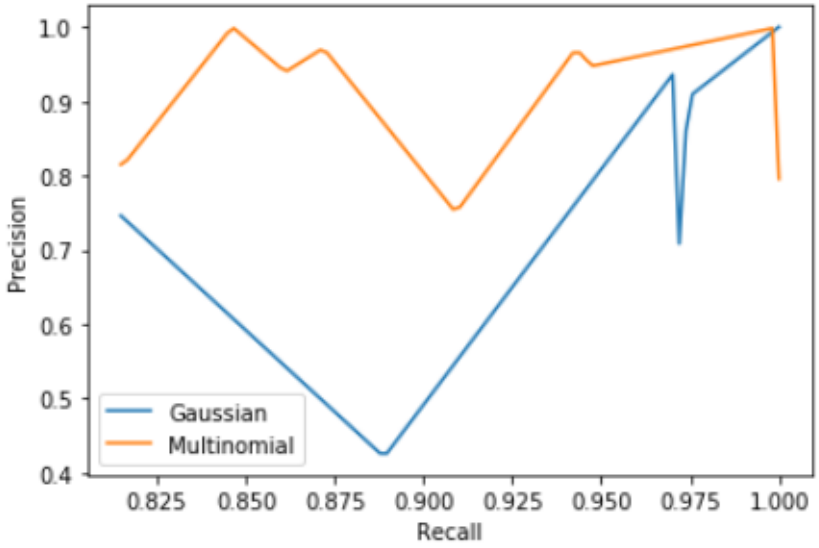

FIGURE 6 – Courbe de précision-rappel pour des distributions gaussienne et multinomiale

Néanmoins il faut garder à l'esprit que les sets de d'entraînement et de test sont définis aléatoirement. Ainsi suivant le jeu de set, ces valeurs sont différentes. Toutefois, la distribution multinominale s'est avérée toujours plus efficace avec un écart plus ou moins grand.

3 Question 3 - SVM

Cette fois-ci, nous allons nous concentrer sur la classification SVM. Et regarder de la même manière l'implémentation d'un classifieur basé sur la classification SVM, puis analyser les statistiques de notre classification suivant différents paramètres.

3.1 Algorithmique

La classification s'effectue en 3 étapes :

1. Définir un set d'entraînement et un set de test
2. Entraîner le classifieur sur le set d'entraînement
3. Déterminer si la classification est correcte sur le set de test

De la même manière que la classification probabiliste vu précédemment, il faut définir 2 sets afin d'entraîner le système sur le premier jeu de données. Puis analyser, les prédictions de notre classifieur grâce au second jeu de données.

3.2 Résultats

Dans le cas d'une classification SVM, il est possible de choisir différents paramètres tels que `C`, `gamma` et `kernel`. Le premier paramètre déterminant est le paramètre de régulation `C` qui permet de modifier l'influence des contraintes lors de l'apprentissage.

1. Pour `C` :

C	Précision
0.01	98.82%
0.1	98.82%
0.5	99.12%
1	99.12%
2	99.12%
4	99.41%
10	99.41%
100	99.41%

Table 1 – Précision en fonction de C

Nous pouvons observer grâce à ce tableau que la précision augmente en fonction de la valeur de C. Ainsi la classification a besoin de bien prendre en compte ce qu'elle a appris durant l'entraînement. De plus, nous pouvons remarquer que celle-ci stagne à part de 4. Pour obtenir une meilleure classification, il faut donc que $C \geq 4$.

2. Pour `gamma` :

gamma	Précision
scale	99.41%
auto	98.82%

Table 2 – Précision en fonction de gamma

Pour le choix de gamma, les valeurs définies par `scale` et `auto` ne semblent pas entraîner de sur-entraînement du système. Toutefois le choix de `scale` semble plus adapter à notre problème et permet de meilleur résultat.

3. Pour `kernel` :

kernel	Précision
linear	97.94%
poly	99.12%
rbf	99.41%
sigmoid	92.04%

TABLE 3 – Précision en fonction du kernel

En ce qui concerne le choix du kernel afin de déterminer l'hyperplan utilisé, les valeurs obtenues avec le jeu de donnés test nous indiquent que `poly` et `rbf` sont plus adaptés à notre classification. Et `rbf` est celui qui permet la meilleure précision.

Au final, il est possible d'obtenir une classification SVC ayant une précision de 99.41% grâce aux paramètres, $C \geq 4$, gamma valant "scale" et un kernel valant"rbf".