4A : RESEAUX DE NEURONES
Nom et prénom : SHE Houyu, GUIZANI Ahmed, Moudoub Lila
classe : 4A41

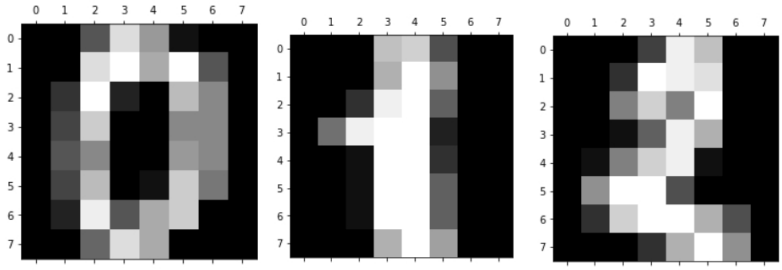
TP3 (durée : 3h)
Reconnaissance d'écriture par réseaux de neurones

1. Analyse des données

On dispose d'une base de données contenant 1797 images des 10 chiffres manuscrits.

a) Charger la base de données *digits disponibles* sous sklearn.

Déterminer la dimension *D* des données et le nombre d'exemple par classe. Observer quelques images :
- la dimension *D* des données : 8x8 = 64
- le nombre d'exemple par classe : 10 classes



b) Séparer une fois pour toutes la base initiale en deux : apprentissage (70%) et test (30%) (*model_selection.train_test_split*).

```
from sklearn import model_selection
X_train,X_test,y_train,y_test = model_selection.train_test_split(X,y,test_size=0.3, random_state=42)
```

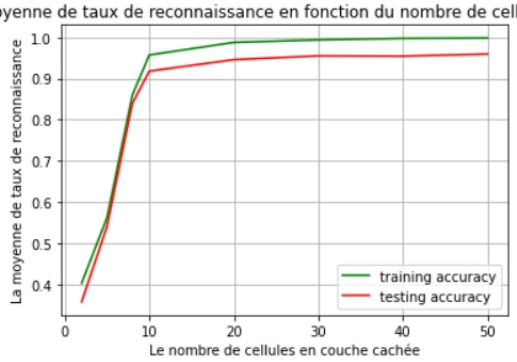
2. Apprentissage

Définir le réseau :

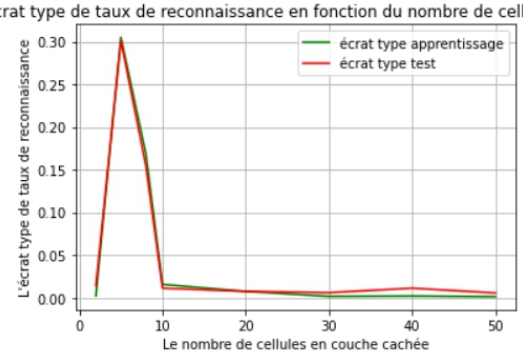
*clf1 = MLPClassifier(hidden_layer_sizes=**C**, activation='tanh', solver='sgd', batch_size=1, alpha=0, learning_rate='adaptive', verbose=1)*

Entraîner le réseau (fonction *fit*). Optimiser la structure du réseau de neurones (nombre de cellules en couche cachée). Étudier l'influence du nombre de neurones cachés sur les taux de reconnaissance en apprentissage et en généralisation (fonction *score*). Conclure sur l'architecture optimale. Vous pouvez modifier les paramètres en fonction des conclusions tirées au TP2.

Le graphe de moyenne de taux de reconnaissance en fonction du nombre de cellules en couche cachée



Le graphe d'écrat type de taux de reconnaissance en fonction du nombre de cellules en couche cachée



On remarque que plus qu'on augmente le nombre de neurones cachés le réseau devient bien meilleur

On conclure que notre nombre de cellules optimal sur la couche cachée est de 20. En effet, on obtient un bon taux de reconnaissance sur la base d'apprentissage mais aussi un écart-type minimum. De ce fait, on peut dire que notre réseau à une faible variance et arrive relativement bien à généraliser sur la base de test.

```
la moyenne de taux en apprentissage :  0.9875364624767967
l'écrat type de taux en apprentissage 0.007391864155272513


la moyenne de taux en test :  0.945679012345679
l'écrat type de taux en test 0.00748417015616181
```

3. Cross-validation

Afin d'améliorer les performances en généralisation du réseau de neurones, on se propose de mettre en œuvre un apprentissage avec arrêt précoce (*early_stopping*) par cross-validation.

Changer les paramètres du réseau pour séparer la base d'apprentissage précédente en deux sets : apprentissage (80%) et validation croisée (20% : *validation _fraction=0.2*).

Entraîner un réseau de neurones avec arrêt par cross-validation (fonction *fit*). Optimiser le nombre de neurones cachés C : répéter 10 fois l'apprentissage et calculer la moyenne et l'écart-type des taux en apprentissage et en validation afin de minimiser le biais et la variance (fonction *score*). Comparer avec les résultats obtenus précédemment. Conclure.

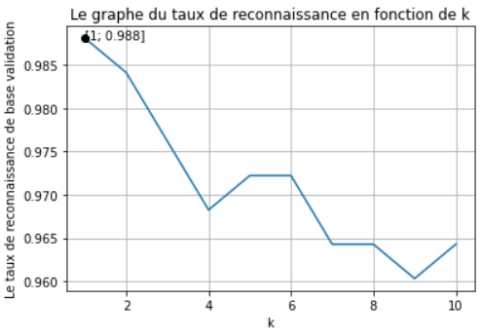
Nous avons choisi d'utiliser 20 neurones sur la couche cachée. En effet, pour C=20, nous obtenons un très bon taux de reconnaissance sur la base de validation et sur la base d'entraînement. De plus, nous disposons d'une variance relativement faible en observant nos écarts-types.

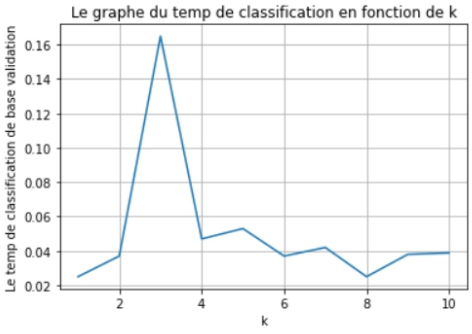
```
la moyenne de taux en apprentissage :  0.9287562189054727
l'écrat type de taux en apprentissage 0.04182314700576484
la moyenne de taux en validation :  0.8896825396825395
l'écrat type de taux en validation 0.04006866813789455
```

Conserver les poids du réseau optimal. Donner la matrice de confusion sur la base de test.

```
La matrice de confusion :
 [[51  0  1  0  0  0  1  0  0  0]
 [ 0 43  1  0  0  1  0  1  4  0]
 [ 0  2 43  1  0  0  1  0  0  0]
 [ 0  0 14 33  0  1  0  0  1  5]
 [ 0  4  0  0 50  1  2  3  0  0]
 [ 0  2  1  0  0 58  0  1  1  3]
 [ 2  0  0  0  0  3 48  0  0  0]
 [ 0  0  0  0  2  4  0 48  1  0]
 [ 0  1  5  2  0  4  1  0 28  2]
 [ 0  0  0  8  0 12  0  1  1 37]]
```

Comparer les résultats obtenus avec ceux de l'algorithme des *k*-plus-proches-voisins en termes de taux de reconnaissance et de temps de classification. Régler *k* sur une base de validation.





En effet, pour l'algorithme des k-plus-proches-voisins (K=1), nous obtenons un très bon taux de reconnaissance sur la base de validation (0.988) par rapport au réseau de neurones (0.928). De plus, nous disposons d'un temps de classification relativement faible (0.03 second). On conclut que l'algorithme des k-plus-proches-voisins est mieux que le réseau de neurones ainsi que les résultats obtenus précédemment sont relativement mieux que les résultats obtenus avec le cross validation.

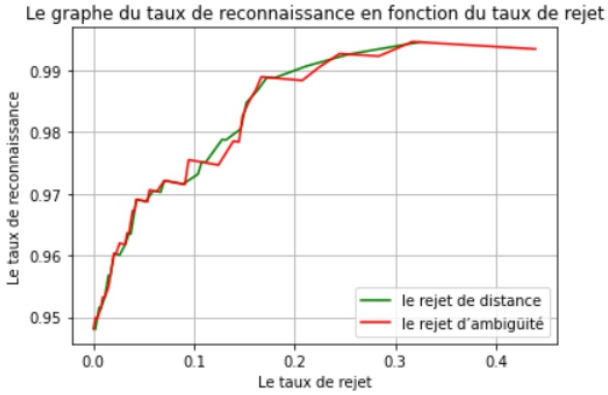
4. Rejet

On se propose d'améliorer les performances précédentes en autorisant le rejet dans l'étape de décision. On étudiera successivement :

- Le rejet de distance : *argmax_reject_threshold*
- Le rejet d'ambigüité : *argmax_top2_reject_threshold*

On utilisera la fonction *predict_proba* pour obtenir les probabilités *a posteriori* des classes.
Faire varier le seuil (*threshold*) de 0 à 1 par pas de 10^{-2}. Pour chaque valeur, calculer :
- Le taux de rejet (#exemple rejetés/#exemples total)
- Le taux de reconnaissance (#exemple bien classés / #exemples classés)

Tracer dans les deux cas la courbe (taux de reconnaissance en fonction du taux de rejet). Choisir la méthode la plus efficace et le seuil associé (meilleur rapport #exemples bien classés / #exemples rejetés).




```
Le meilleur rapport se trouve en rejet  d'ambigüité :  512.0
threshold =  0.06
```

On se basent sur les résultats obtenus, on conclut que la méthode la plus efficace est **le rejet d'ambiguïté,** ainsi que le seuil associé est 512.