Classifieur MNIST

Pierre Genty Olivier Léobal

1er décembre 2017

1 Utilisation

L'exécutable est `classifier.py`. Attention, le chemin des données à utiliser est fixe dans le programme : elles doivent donc avoir les noms par défaut, dans un répertoire `data/`.

On peut passer plusieurs options :

- `-h` (ou `help`) : aide
- `-p <n>` : activer la PCA, avec `<n>` composants
- `-a` : afficher les images moyennes de chaque catégorie à la fin de l'analyse
- `-o <path>` : écrire un fichier `<path>.npy` avec résultats de l'analyse

Le programme requiert `numpy` et `matplotlib` (pas de surprises). Si le PCA est activé, il importera aussi `sklearn` (scikit-learn), qui requiert SciPy pour son implémentation de la PCA.

2 Performance

Sur le jeu de test, le taux d'erreur global est de 38.7%. On remarque des performances particulièrement au dessus des autres sur les pantalons (classe 1) et baskets (classe 7), mais très inférieures avec les pull-overs (classe 2) et les chemises (6).

Catégorie	0	1	2	3	4	5	6	7	8	9	Total
Taux d'erreur (%)	39	7	73	24	34	46	83	10	51	18	38.7

La distribution des des choix du système est :

Catégorie	0	1	2	3	4	5	6	7	8	9
Proportion (%)	8	11	5	13	16	13	5	16	5	9

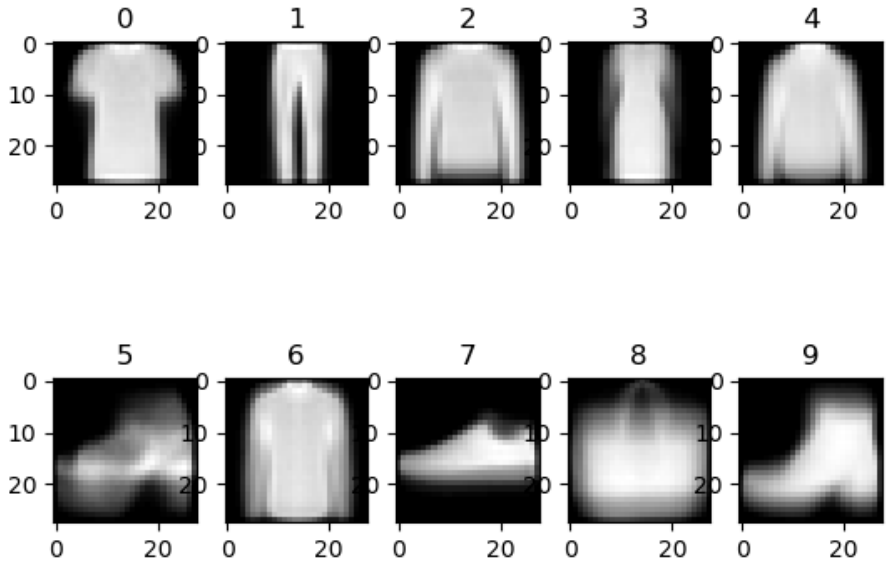
Totaux possiblement $\neq 100$ à cause des arrondis

Les pantalons (1) sont effectivement bien reconnus, mais le bon score des baskets (7) est en partie du à une sur-reconnaissance.

On peut en savoir plus grâce à la matrice de confusion :

		Label									
		0	1	2	3	4	5	6	7	8	9
Choix	0	312	6	5	7	1	0	84	0	1	0
	1	12	449	1	60	12	0	6	0	2	0
	2	3	5	139	0	48	0	29	0	2	0
	3	89	16	12	378	49	1	55	0	44	5
	4	18	2	202	24	342	0	144	0	62	1
	5	56	1	74	25	34	261	79	28	63	19
	6	13	2	88	5	33	0	81	0	8	0
	7	2	0	0	0	0	193	1	452	78	61
	8	2	0	0	1	2	0	3	0	260	0
	9	0	0	0	0	0	30	0	20	6	391

On voit que les pull-overs (2) sont très souvent pris pour des manteaux (4), qui ont une forme très similaire ; le même problème survient avec les chemises (6).



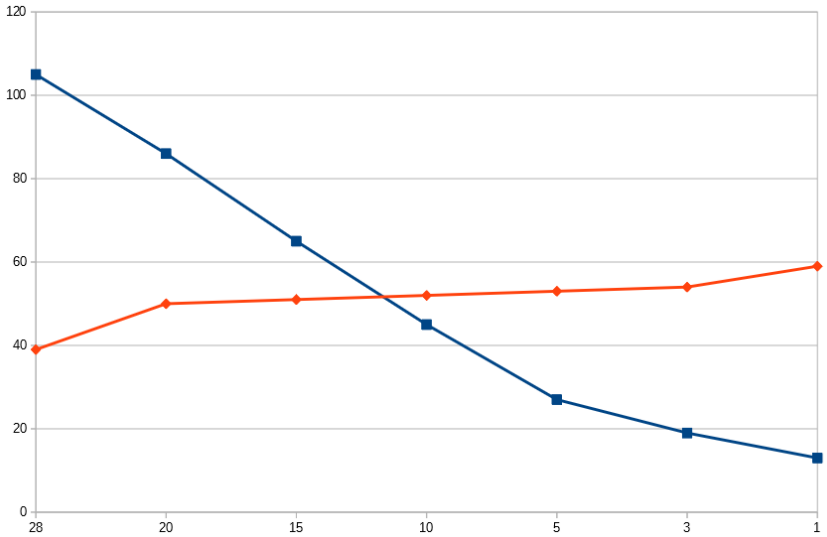
3 Analyse en composantes principales

On peut utiliser le PCA avec l'option `-p <nombre de composants>`.

Le PCA implique des opérations supplémentaires pour remettre en forme les données, mais accélère le traitement. En pratique, on observe :

# composantes	Temps d'exécution	Taux d'erreur
(sans PCA)	1m45	38.7%
20	1m26	50.7%
15	1m05	51.7%
10	0m45	52.2%
5	0m27	52.6%
3	0m19	53.7%
1	0m13	59.1%

Ou, sous forme de graphique :



Sur les PC de l'école : Linux Mint 18.1, i5-2400, 4Go RAM, interpréteur Anaconda. Moyenne de 5 exécutions à la suite, avec la commande `time`

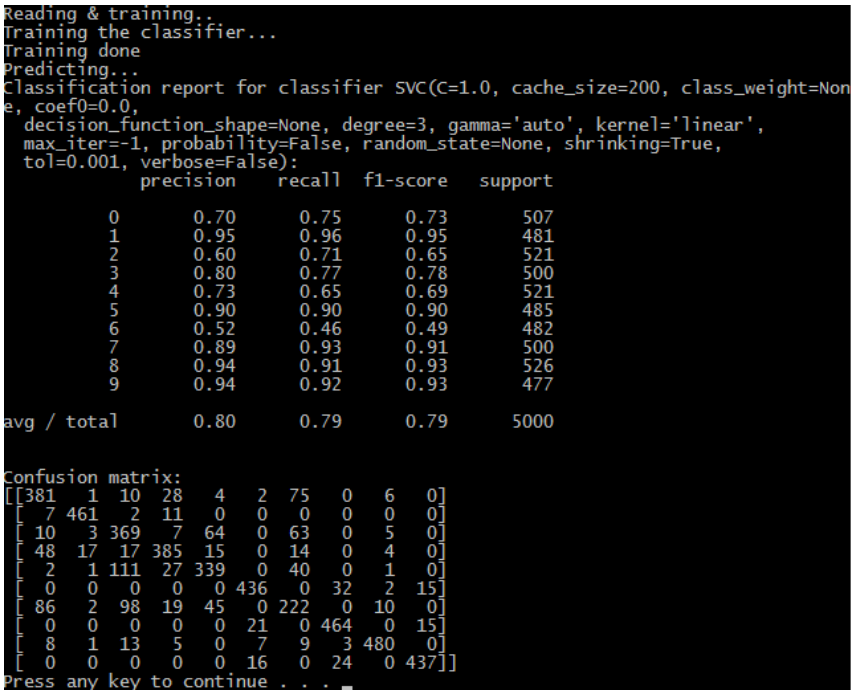
On augmente largement notre taux d'erreur, mais en proportion très moindre du temps d'exécution. Ici, on a divisé le temps d'exécution par 8 en perdant 20 points de précision. On réalise aussi que la PCA permet de réduire en fait le nombre de dimensions utiles à 84 voire 28 dimensions (3 ou 1 composantes) : davantage deviennent inutiles.

4 Analyse des performances des classifieurs SciKit-learn

Les classifieurs sont dans le module scikit-compare.py, ce module ne dispose malheureusement pas d'opttion variées et se lance en exécution seule.

4.1 Un classifieur SVM: le SVC

Le SVC est lancé avec un kernel linéaire apporte les performances détaillées sur le rapport suivant (généré par les metrics de scikit):



```
Reading & training..
Training the classifier...
Training done
Predicting...
Classification report for classifier SVC(C=1.0, cache_size=200, class_weight=Non
e, coef0=0.0,
  decision_function_shape=None, degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False):
             precision    recall  f1-score   support

          0       0.70      0.75      0.73       507
          1       0.95      0.96      0.95       481
          2       0.60      0.71      0.65       521
          3       0.80      0.77      0.78       500
          4       0.73      0.65      0.69       521
          5       0.90      0.90      0.90       485
          6       0.52      0.46      0.49       482
          7       0.89      0.93      0.91       500
          8       0.94      0.91      0.93       526
          9       0.94      0.92      0.93       477

avg / total       0.80      0.79      0.79      5000


Confusion matrix:
[[381   1  10  28   4   2  75   0   6   0]
 [  7 461   2  11   0   0   0   0   0   0]
 [ 10   3 369   7  64   0  63   0   5   0]
 [ 48  17  17 385  15   0  14   0   4   0]
 [  2   1 111  27 339   0  40   0   1   0]
 [  0   0   0   0   0 436   0  32   2  15]
 [ 86   2  98  19  45   0 222   0  10   0]
 [  0   0   0   0   0  21   0 464   0  15]
 [  8   1  13   5   0   7   9   3 480   0]
 [  0   0   0   0   0  16   0  24   0 437]]
Press any key to continue . . .
```

On remarque que là aussi, les pull-overs (2), les manteaux (4) et les chemises (6) sont les catégories aux plus hauts taux d'erreur, étant très similaires en termes de formes.

4.2 Un classifieur par voisins proches: le KNeighborsClassifier

Le KNeighborsClassifier est lancé avec le paramètre de base 5 voisins les plus proches et donne le rapport de performances suivant:

```
Reading & training..
Training the classifiers...
Training done
Predicting...
Classification report for classifier KNeighborsClassifier(algorithm='auto', leaf
_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform'):
             precision    recall  f1-score   support

          0       0.73      0.82      0.77       507
          1       0.98      0.95      0.97       481
          2       0.67      0.74      0.71       521
          3       0.86      0.82      0.84       500
          4       0.73      0.68      0.71       521
          5       0.99      0.79      0.88       485
          6       0.58      0.55      0.56       482
          7       0.85      0.93      0.89       500
          8       0.97      0.91      0.94       526
          9       0.86      0.97      0.91       477

avg / total       0.82      0.82      0.82      5000


Confusion matrix:
[[416   1  15  18   3   0  49   1   4   0]
 [  6 459   3   9   1   0   3   0   0   0]
 [ 15   0 388   3  54   0  61   0   0   0]
 [ 33   6  10 412  27   0  10   0   2   0]
 [  3   0  81  26 355   0  55   0   1   0]
 [  3   0   0   1   0 384   0  56   2  39]
 [ 90   1  72   9  41   0 263   0   6   0]
 [  0   0   0   0   0   3   0 463   1  33]
 [  2   1  10   0   5   1  15   9 481   2]
 [  0   0   0   0   0   1   1  14   0 461]]
Press any key to continue . . .
```

On remarque une performance très constante avec une moyenne à 82%, et tout comme auparavant, les mêmes catégories de vêtements accusent un taux de précision plus bas que la moyenne: les pull-overs (2), les manteaux (4) et les chemises (6).