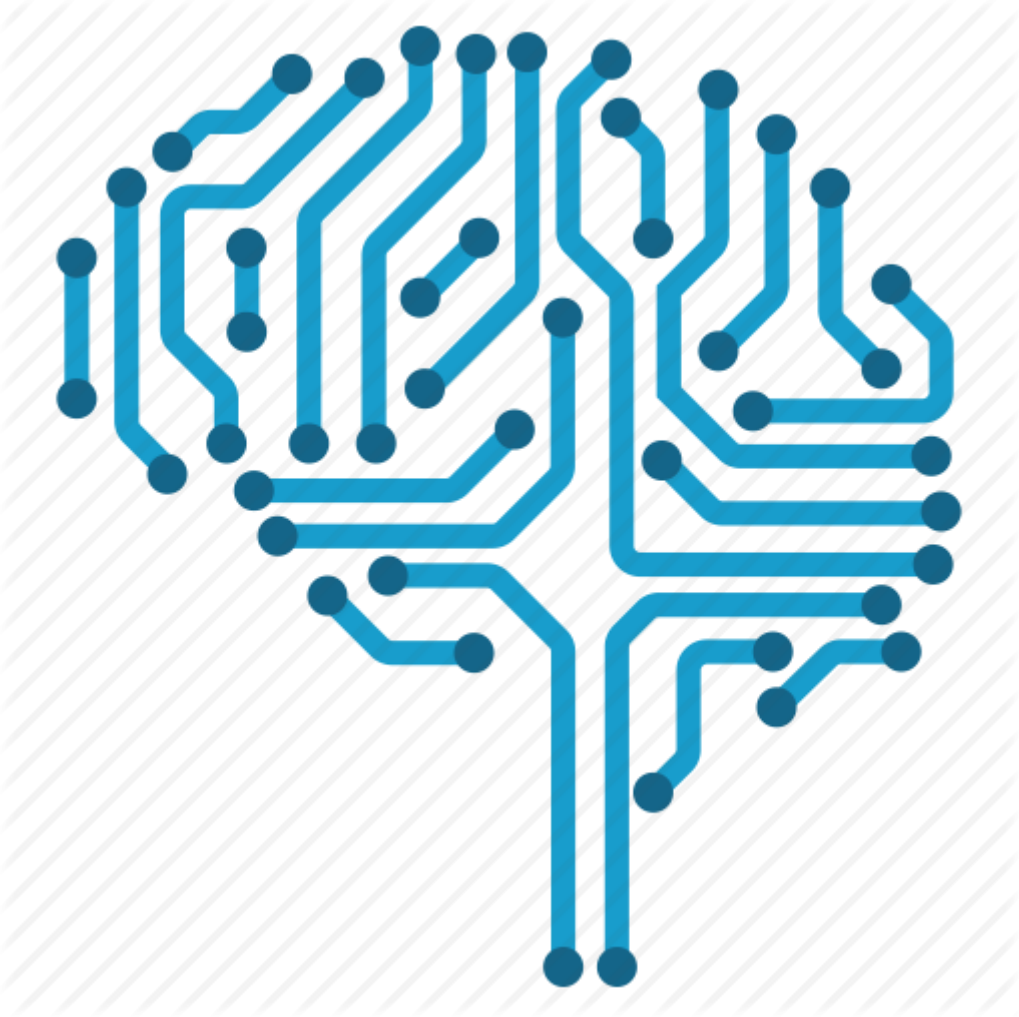
Projet Machine Learning

Reconnaissance d'images



Eurydice Ruggieri
Thomas von Ascheberg

POLYTECH PARIS-SUD

ET4 INFO - Groupe 2
25 mars 2019

Eurydice Ruggieri
Thomas von Ascheberg

Sommaire

N.B.

- Pour les méthodes utilisées ci-dessous, le taux d'erreur provient de la classification sur les données de développement, mais les temps donnés pour la classification ont été obtenus à partir des données de test.
- Toutes les valeurs obtenues l'ont été à partir d'une même machine, garantissant la cohérence et l'homogénéité des résultats.

Eurydice Ruggieri
Thomas von Ascheberg

Classifieur à distance minimum

Choix d'implémentation

Pour cette méthode, on commence par l'apprentissage à partir des données. Pour cela, on calcule les centroïdes de chaque classe. Ensuite, pour chaque image des données de développement, on va tenter de prédire leur classe. Pour cela, on calcule les distances aux centroïdes des classes défini lors de la phase d'apprentissage. Comme son nom l'indique, le classifieur à distance minimum devine qu'une image appartient à la classe dont le centroïde est le plus proche.

Temps d'exécution

Le temps de l'apprentissage est de 0.0099 sec.
Le temps de la classification est de 0.308 sec.

On observe que le temps d'apprentissage est largement plus faible que le temps de classification. Cela s'explique par le fait que pour l'apprentissage, on calcule seulement la moyenne pour obtenir les centroïdes des différentes classes. Alors que lors de la classification, on calcule les distances aux centroïdes puis on compare deux à deux afin de trouver la distance minimale (ce qui est beaucoup plus long).

Taux de reconnaissance

On mesure la performance de l'implémentation de cette méthode en calculant le taux d'erreur (ou 1-taux de reconnaissance) qui correspond aux nombre d'images mal classées par rapport aux nombre total d'image dans l'ensemble de développement. Ici, le taux d'erreur est de 32,42%, ce qui est plutôt bien, puisque si on devinait au hasard, on serait plus proche de 90% d'erreur (une chance sur dix car on a 10 classes). Ainsi, en faisant un apprentissage basique avec le classifieur à distance minimum, on diminue par 3 le taux d'erreur par rapport à une classification aléatoire.

Eurydice Ruggieri
Thomas von Ascheberg

Analyse en composantes principales (PCA)

Choix d'implémentation

Ici on va diminuer le nombre de dimensions (diminuer la résolution) de nos images sur plusieurs classifieur pour mieux percevoir l'utilité de ce procédé. Dans un premier temps, on utilise à nouveau le classifieur à distance minimum.

Pour le classifieur à distance minimum, après plusieurs essais sur l'ensemble des dimensions, on a choisi de faire varier le nombre de dimensions gardées dans l'intervalle [0,30] (10 valeurs par pas de 3) car il s'agit de l'intervalle le plus intéressant à analyser : c'est dans celui-ci que les valeurs varient le plus.

Taux de reconnaissance et Temps d'exécution

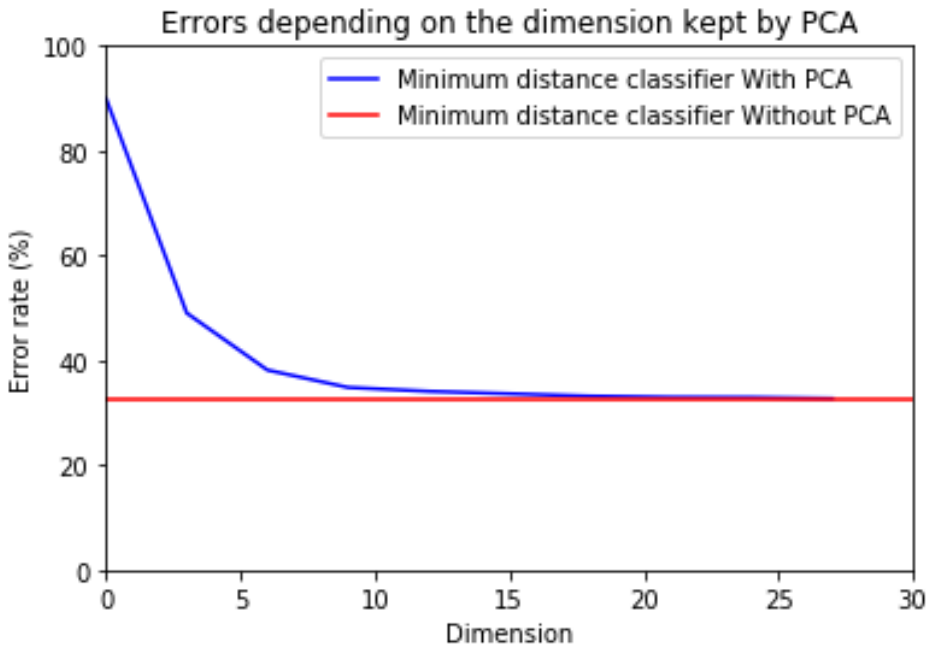
Taux d'erreur

Ici, on a donc choisi d'utiliser la PCA pour 10 valeurs qui sont selon nous les plus intéressantes. On a les résultats suivants :

Dimension	Taux d'erreur (%)	Temps d'apprentissage (sec)	Temps de classification (sec)
0	89.94	0.0005	0.2295
3	49.02	0.0016	0.2562
6	38.14	0.0014	0.2576
9	34.82	0.0015	0.2573
12	34.08	0.0016	0.2577
15	33.62	0.0018	0.2606
18	33.16	0.0016	0.2580
21	32.94	0.0016	0.2601
24	32.9	0.0018	0.2606
27	32.72	0.0018	0.2620
784	32.42	0.0086	0.3307

On observe que pour le cas du classifieur à distance minimum, la PCA diminue très peu le taux d'erreur, et seulement quand on dépasse 20 en dimension. Quant aux temps d'apprentissage, on observe qu'ils sont très faibles plus les dimensions sont réduites. Dans notre cas présent, cela est presque négligeable comparé au temps de classification. En réduisant ici drastiquement le nombre de dimensions, on conserve un taux d'erreur très proche de celui obtenu sans PCA, ce qui est intéressant en termes de performances. Ainsi, la PCA permet d'accélérer l'apprentissage et un peu le temps de classification (en réduisant le nombre de calcul puisque l'on réduit le nombre de dimensions).

Courbe de comparaison



Support Vector Machines (SVM)

Choix d'implémentation

La méthode utilise un nombre de points de référence (vecteurs supports) pour la phase d'apprentissage.
A partir de ces vecteurs définis, on peut classifier les données de développement.
Ici, après avoir testé seulement l'option kernel = 'linear', nous avons fait le choix d'utiliser la SVM avec le paramètre gamma = 'scale', car ce dernier est plus efficace lorsque l'on fait une classification multiclasse avec des données denses (ce qui est notre cas). (On passe de 20% d'erreur de classification pour kernel = 'linear' à 13% pour gamma = 'scale')

Temps d'exécution

Le temps d'apprentissage est de 25.02 sec.
Le temps de classification est de 11.05 sec.

On observe que le temps d'apprentissage est beaucoup plus important que le temps de classification, car la méthode détermine les vecteurs supports et met en place la fonction qui permettra ensuite de déterminer la classe des futures données. La classification des données est plus rapide car il suffit d'utiliser la fonction pour les classer.

Taux de reconnaissance

Taux d'erreur

On a un taux d'erreur de 13.90%.
On voit donc que le taux d'erreur est bien plus bas que pour le classifieur à distance minimum. Cela montre que la méthode SVM est plus précise par rapport aux méthodes précédentes. Elle est cependant plus lente (car doit faire plus de calculs).

Matrice de confusion

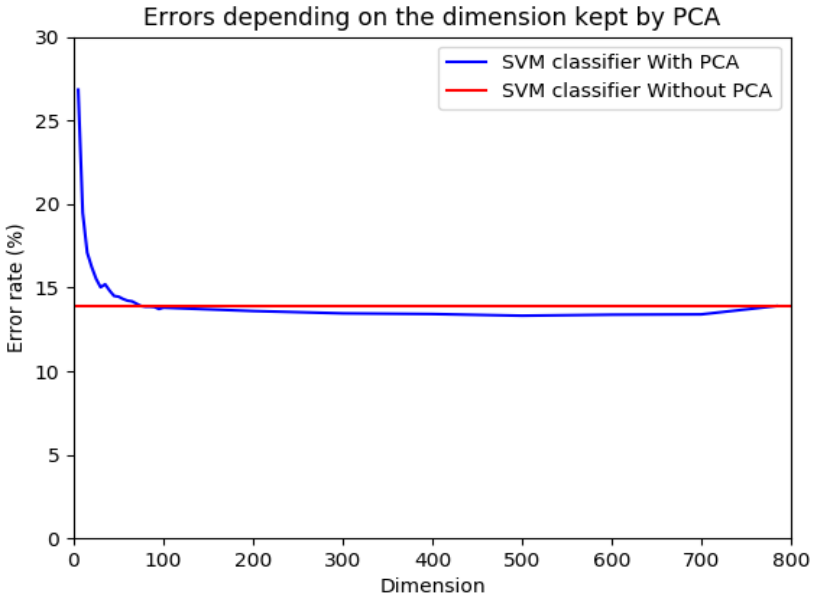
On calcule la matrice de confusion du système :

class	1	2	3	4	5	6	7	8	9	10
1	382	2	15	22	2	0	74	1	5	0
2	9	491	1	8	1	0	2	0	0	0
3	12	1	323	7	63	0	58	0	4	0
4	25	15	12	399	10	0	22	0	1	0
5	4	1	114	32	312	0	45	0	1	0
6	0	0	0	0	0	457	1	33	1	11
7	90	1	86	14	62	0	263	0	11	0
8	0	0	0	0	0	25	0	417	1	22
9	6	0	14	0	5	5	7	3	456	0
10	1	0	0	0	0	14	0	29	1	488

On observe que notre classification est plutôt efficace, car les valeurs les plus importantes sont sur la diagonales qui sont les vrais “positifs”, c’est à dire les images qui ont été effectivement classées dans leur vraie classe.

PCA et SVM

On va ensuite appliquer la PCA sur la SVM pour essayer d'augmenter les performances de la SVM. (Le tableau complet est disponible en annexe)



En plus d’accélérer (légèrement) l’apprentissage et la classification, ici la PCA appliquée à la SVM rend ce classifieur plus précis car elle réduit le bruit (information inutile de l’image).

Plus Proches Voisins (K Neighbors)

Choix d'implémentation

La méthode utilise le principe de localité pour déterminer à quelle classe appartient un échantillon à partir de la classe de ses n plus proches voisins. Pour tester ce classifieur, on a choisi de la paramétrer le nombre de voisin et de voir quel semble être le nombre optimal de voisin. Dans un premier temps nous allons regarder les résultats si le nombre de voisins consultés n = 2. On fera par la suite varier cet hyperparamètre pour essayer d'obtenir le n optimal.

Temps d'exécution

Pour n = 2 :
Le temps de l'apprentissage est de 0.5351 sec.
Le temps de la classification est de 47.5968 sec.

On peut expliquer ces valeurs par la technique utilisée. En effet, à chaque fois qu'une image est classifiée, la distance aux voisins doit être calculée pour obtenir les n distances des voisins les plus proches. Puis l'image est classifiée selon le nombre maximal de voisin la même classe parmi ces n proches voisins, ce qui prend du temps, d'autant plus quand on augmente le nombre n de ceux-ci.

Taux de reconnaissance

Taux d'erreur

On a un taux d'erreur de 18.8%.
La méthode est donc très efficace, notamment car elle utilise la proximité avec des éléments déjà classifiés pour déterminer la classe à laquelle appartient une image.

Matrice de confusion

class	1	2	3	4	5	6	7	8	9	10
1	451	1	12	10	3	0	23	1	2	0
2	5	502	0	5	0	0	0	0	0	0
3	13	2	383	3	47	0	20	0	0	0
4	38	14	9	399	20	0	4	0	0	0
5	8	0	130	48	298	0	25	0	0	0
6	0	0	1	2	0	433	2	40	0	25
7	130	1	107	15	59	0	214	0	1	0
8	0	0	0	0	0	9	0	442	0	14
9	8	1	7	1	5	1	18	4	451	0
10	1	0	0	0	1	7	0	37	0	487

Cette matrice nous permet de visualiser plus précisément l'efficacité de la méthode. On voit ici que sur la diagonale des éléments correctement classifiés, on a les nombres les plus importants, ce qui montre bien que la méthode fonctionne bien. On peut cependant voir le détail des classes les plus difficile à classifier, ici particulièrement la classe 7.

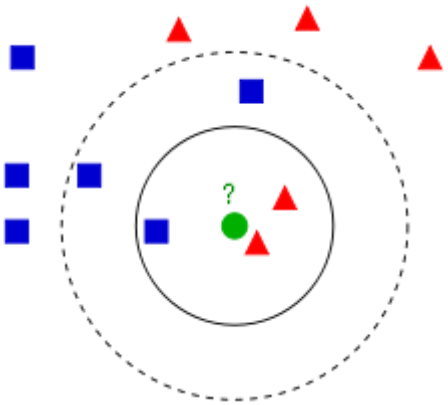
Optimisation des hyperparamètres

On a ensuite voulu faire varier le nombre de voisins n, pour voir les variations de performance :

Nb voisins	Taux d'erreur	Tps apprentissage	Tps de classification
4	17.52	0.5352	48.2900
5	17.02	0.5465	49.3084
6	17.26	0.5343	48.9825
8	17.28	0.5411	49.2662
10	17.98	0.5390	50.1634
15	18.2	0.5395	50.2781
20	18.48	0.5398	51.0291
30	19.54	0.5476	50.1701
40	19.86	0.5443	50.8613
50	20.38	0.5446	50.7566

60	20.7	0.5475	50.8705
70	21.2	0.5470	51.0323

On observe que lorsque l'on augmente le nombre de voisin, on augmente inévitablement le temps de classification. Quant au taux d'erreur, celui-ci diminue d'abord vers un nombre optimum, pour n = 5, puis augmente à nouveau. Cela s'explique par le fait qu'en premier, en augmentant le nombre de voisins, on augmente les points de comparaison proches, donc la chance de tomber dans la classe exacte. Mais lorsque l'on prend trop de voisins, on compare l'élément à classifier avec des éléments lointains, qui ne sont donc plus cohérents pour la classification (On perd le principe de localité).

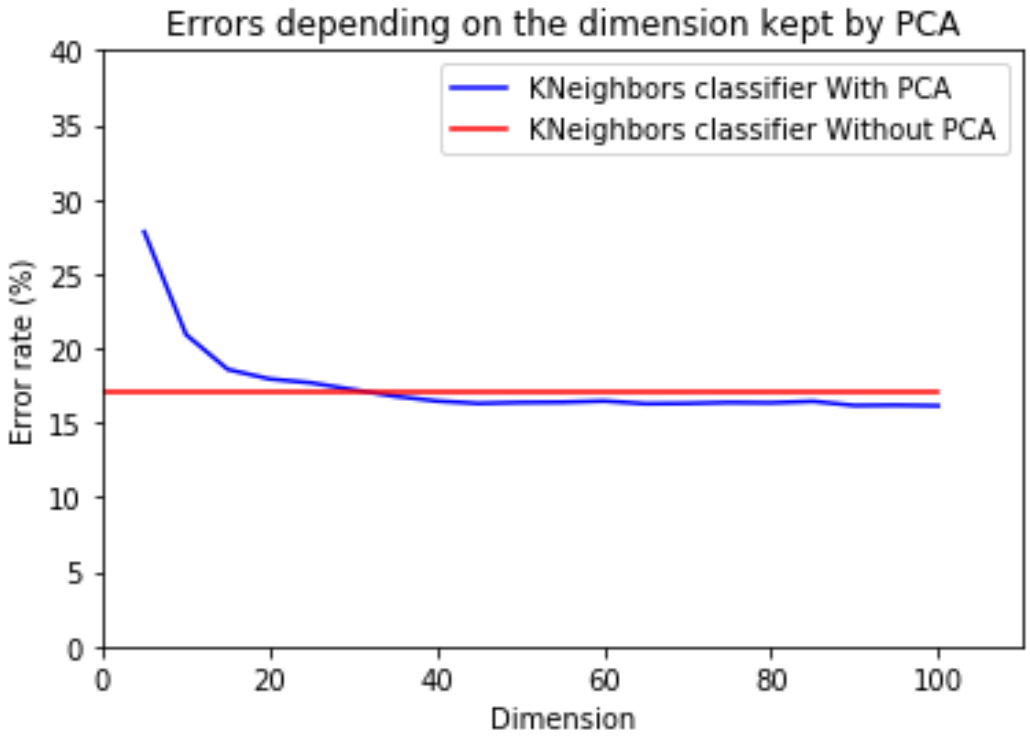


Source image : Wikipedia

On a ensuite appliqué la PCA au système à 5 voisins, qui est le plus performant, pour voir si on pouvait encore améliorer les performances :

Dimension	Taux d'erreur (%)	Temps d'apprentissage (sec)	Temps de classification (sec)
5	27.82	0.0049	0.1207
10	20.92	0.0064	0.1755
15	18.56	0.0080	0.2347
20	17.96	0.096	0.3146
25	17.76	0.0113	0.3680
30	17.24	0.0128	0.4367
35	16.78	0.0145	0.5502
40	16.42	0.0158	0.5854
45	16.22	0.0178	0.6605
50	16.52	0.0194	0.7899
55	16.32	0.0218	0.9269

60	16.5	0.0238	1.0421
65	16.34	0.0259	1.0803
70	16.34	0.0288	1.2133
75	16.43	0.0313	1.2133
80	16.36	0.0376	1.4648
85	16.24	0.0355	1.5439
90	16.2	0.0376	1.6683
95	16.32	0.0433	1.8570
100	16.28	0.0506	2.1134
784	18.8	0.5351	47.5968

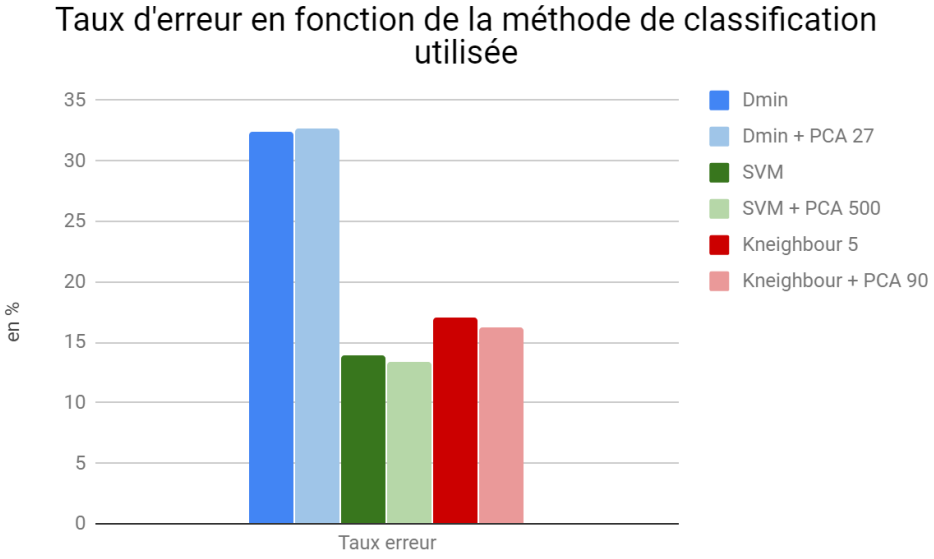


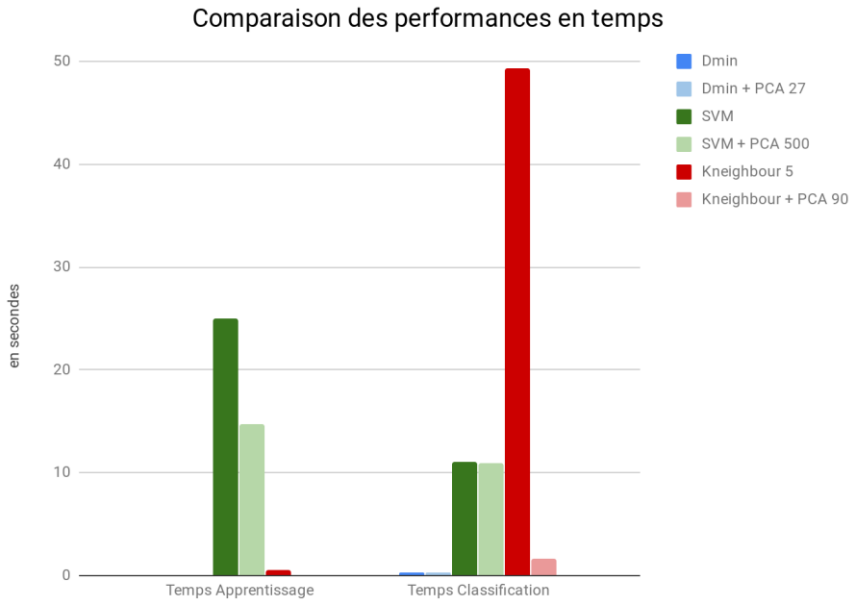
On se rend compte ici qu'en plus d'être beaucoup moins lourde en calcul, la PCA réduit le taux d'erreur. En effet la PCA en réduisant les détails de l'image, réduit aussi le "bruit" de l'image, ce qui explique ce meilleur résultat.

Conclusion

Comparaison des performances

Ainsi, comme nous l'avons vu, le classifieur à distance minimal est le moins précis, suivi du classifieur K Neighbor, puis de la SVM. On observe aussi, que généralement la PCA appliquée à ces classifieurs permet d'améliorer le taux de reconnaissance (légèrement). Quant aux temps d'exécution, on remarque que c'est la SVM qui est la plus lente en apprentissage alors que le classifieur K Neighbor est de loin le plus lent pour classifier. De même, la PCA permet d'améliorer ces temps. Voici quelques graphiques récapitulant nos résultats :





Choix du meilleur système

Au vu des résultats obtenus, notre choix du système le plus précis se porte donc sur la méthode SVM combiné avec la PCA dimension 500. On voit en effet que c'est le système ayant le taux d'erreur le plus faible, et dont le temps d'exécution n'est pas excessif.
Selon nos conclusions, il nous apparaît donc comme le classifieur à choisir pour obtenir un taux d'erreur minimal pour une classification optimale sur les données de test.

Conclusion du Projet

Ce projet nous as permis de découvrir et mettre en œuvre une méthodologie de travail axée sur la performance. Ainsi, au travers l'étude des différentes méthodes de classification, nous avons pu comprendre leurs différences mais surtout quantifier le coût en temps d'un gain en performance (précision). Ainsi grâce à ce projet nous avons désormais une connaissance plus pratique du Machine Learning. De même, ce projet nous as permis de développer nos connaissance en Python. Ces capacités nous permettront à l'avenir de pouvoir analyser des situations similaires et mettre en place des réponses adaptées.

Annexe 1 : Tableau complet de performance SVM + PCA

Dimension	Taux d'erreur	Tps d'apprentissage	Tps de classification
5	26.82	1.0293	0.8201
10	19.5	1.0418	0.8430
15	17.08	1.1008	0.8493
20	16.28	1.2021	0.9251
25	15.54	1.3178	0.9992
30	15.6	1.4273	1.0843
35	15.14	1.5537	1.1966
40	14.8	1.6337	1.3141
45	14.46	1.8426	1.3551
50	14.42	1.8504	1.4870
55	14.26	1.9950	1.5449
60	14.3	2.1442	1.6326
65	14.12	2.1749	1.7438
70	14.06	2.3558	1.8425
75	13.82	2.4310	1.9480
80	13.84	2.5514	1.9761
85	13.78	2.5993	2.0426
90	13.8	2.7029	1.5691
95	13.72	2.8254	2.2336
100	13.7	2.9506	2.3172
200	13.62	5.5379	4.5209
300	13.4	8.4836	6.8629
400	13.44	11.3581	8.8607
500	13.34	14.7215	10.8866
600	13.36	16.6992	12.8168
700	13.4	19.6343	14.8269
784	13.9	19.5973	16.2187