Optimization and data analytics assignment

Vinh Trung Thai
Aarhus university
School of engineering
Aarhus, Denmark
201610455 / Au554243

***Abstract*—This paper will introduce 3 different machine learning algorithm used for classification of images.These algorithms are implemented with Python using the IDE Anaconda. Furthermore some algorithms will be implemented using the frameworks provided for Scikit-learn libraries and some will be implemented manually. The three different classifyers can be described as Nearest Class Centroid(NCC), Nearest Subclass Centroid(NSC) and Nearest Neighbour (NN). The algorithm will be performed on two well known dataset, namely MNIST - handwritten digits and ORL - olivetti face images. The performance of each algorithm will be analysed and discussed in this paper.**

***Index Terms*—Machine learning, classification, visual results of high dimension data, principal component analysis, MNIST ORL data classified.**

I. Introduction

One could say Machine Learning is the biggest knew thing within technology. Everyone has had some experience machine learning whether it be spam mail sorting, fingerprint recognition or even stock prediction. Machine learning is basically telling a computer to recognize a pattern and similarities in data in order to solve tasks or make decision. There exist different types of machine learning which can be described as Supervised learning, unsupervised learning and reinforcement learning, which each have their own subcategory, which some will be discussed. Classification is ”teaching” the algorithm to recognize patterns in a set of data. This will allow the application to ”guess” the certain class a specific sample belongs to. Classification can be done either supervised or unsupervised. Supervised classification is giving the algorithm a data set which contains data already labelled so the application can ”learn” on samples which already has been classified.
Unsupervised classification is to teach the application on unlabelled data. By giving the algorithm unlabelled data, the algorithm bases its decision by purely detect similarities in the samples and thereby base its decision on this knowledge.
To sum it up, the concept of teaching an algorithm to classify is by giving it a data set, separate the data set into sections which can be labelled a training set and test set and then see how accurate it is to classify or predict.

This paper the algorithm will be based on the Scikit-learn frameworks [1] and some will be implemented manually. The source code can be found in the source code folder. The classification algorithms will be performed on the datasets MNIST [2] and ORL [3].

II. Data Analysis

In order to the classification algorithm to perform, a set of training data is needed. The earlier mentionend data sets MNIST and ORL will be split into workable sections. The MNIST data set consists of 70k 28x 28 pixel images of hand drawn numbers ranging from 0 to 9 and the ORL consists of 400 40x30 facial images depiction 40 persons. As this is a lot of data, it would be hard for the common person to visualize. In order to visualize this, the Principal Component Analysis(PCA) is utilised.

A. PCA

PCA is a method used to reduce the dimensions of data while retaining the highest possible variance of the data. To give an example, imagine a data set of 1000 dogs, containing the height, weight and length of each dog. In most scenarios these variables are related, as one would imagine, if the weights goes up so tends the height and length as well. PCA transforms these related variables into unrelated variables(x,y,z) which a mathmetical abstractions. This allows the algorithm to ignore one(or more) of the three variables and only use one variable if the data was very relate-able to each other. This transformation can be visualized into a coordinate system which shows the greatest variances.

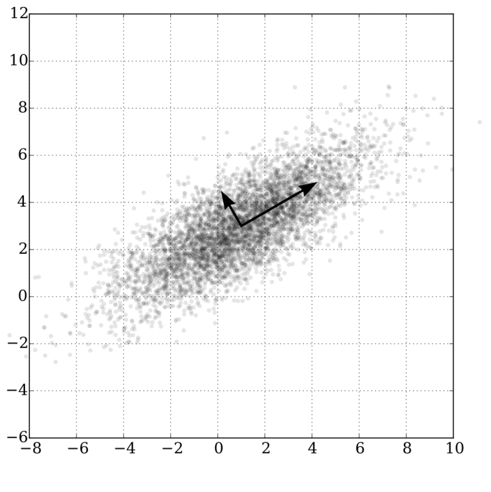


Fig. 1. PCA on multivariate gaussian distribution, where the greatest variances are depicted by the arrows [4]

On figure 1 an image of coordinate system can be seen. The image shows the PCA on a multivariate Gaussian distribution as well as the arrows depicting the greatest variances.

B. MNIST

As mentionen the MNIST data set consists of 70k 28x28 images of handwritten numbers. The algorithm for classification will learn on this dataset by splitting the images into 60k samples for training and 10k for testing and verification.

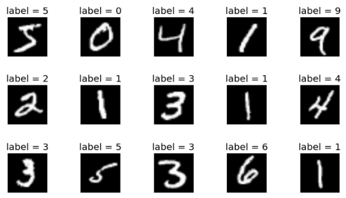


Fig. 2. Labelled images of the MNIST data set [5]

On figure 2 images of labelled handwritten numbers can be seen. As it can be seen, there ore some variances in handwriting, so the images are distinguishable and not exact copies of each other.

C. ORL

As mentioned the ORL face dataset or also known as the Olivetti Research Laboratory face dataset, consists of 400 40x30 images of faces depicting 40 persons. The data will be split into a set of 280 images for training and 120 for testing.

Fig. 3. ORL face dataset images [6]

Both dataset will be used on the algorithm with and without PCA for comparison. The following images shows both datasets after applying PCA.

Figure 4 shows a scatter plot of MNIST and ORL transformed with PCA. Here are the classes seperated into colours and placed as dots in the coordinate system. It can be observed that the MNIST dataset contains more dots compared to ORL, due to it consisting of way more samples. However it only holds 10 classes as unique numbers only goes from 0 to 0. On the other hand ORL quite as many dots as the dataset only consists 400 images, but it does contain images depicting 40 people, so 40 classes can be expected.

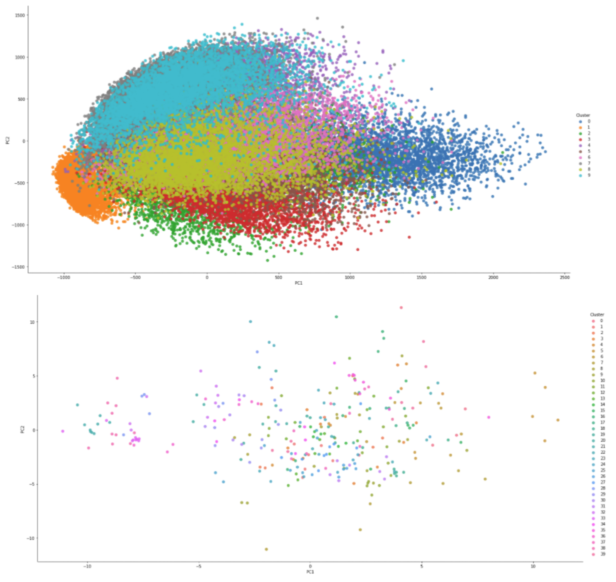


Fig. 4. A scatter plot of the PCA performed on both MNIST(upper) and ORL(Lower)

III. Methods

A. Nearest Centroid Classifier(NCC)

The Nearest Class Centroid Classifier algorithm is a supervised learning algorithm. This algorithm computes the mean vector(centroid) for all samples for each class in the dataset. Then, when a new sample applied to this algorithm, the algorithm will predict the class with the nearest centroid based on the distance metric, Euclidean distance. So in short, an algorithm will be given a labelled dataset, in which it will calculate the centroid while training. After training, when given a sample or input, the algorithm will calculated the distances between the sample and each class centroid. The shortest distance between the sample and centroid will then be picked. [**?**] The NCC in this paper will be performed with different shrink threshold to see how they compare. The benefits of shrinkage can be that it may make the classifier more accuate by reducing the effect of noisy genes.

B. Nearest Subclass Centroid(NSC)

The Nearest Subclass Centroid classifier uses a combination of the NCC and clustering algorithms. NSC is a combination of NCC and a clustering algorithm. The algorithm works by finding a number of prototypes(clusters) for each class, such that the variance is "covered" by each prototype. This also means the algorithm is supervised, as it needs the training data to be labelled. Then it can calculate the clusters with a mean vector, for each cluster to represent the subclasses. The classification can then be calculated based on the distance from the test sample and to the nearest centroid. To cluster the data from the datasets, the KMeans clustering [7] function has been used. The Kmeans algorithm initiates by "randomly" selecting centroids based on the specific number of centroids, decided by the developer. For this paper, the centroids has been selected to be 2,3 and 5.

C. Nearest Neighbour (NN)

The nearest neighbour classifier is an algorithm which is trained on samples with labels and will then predict the class of a test sample based on the class most common amongst the specified neighbours. But compared to the other two methods, NN classifiers is instance-based learning. So instead of creating a general model, this algorithm stores instances of training data. [?] The neighbours can be found with the use of a distance metric, such as Manhattan, Minkowski, Hamming, etc. For this paper, the euclidian distance has been used. The classification is computed based on a majority vote of the nearest neighbors of each point.

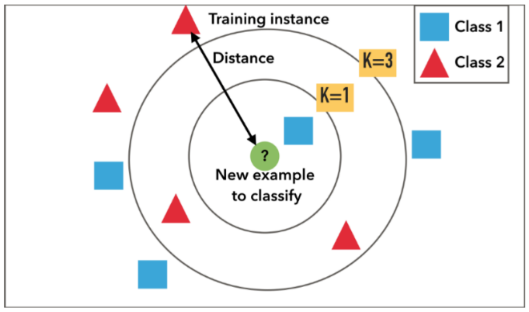


Fig. 5. An illustration of KNN functionality [10]

Figure 5 represents the (K)NN algorithm. The algorithm decides which class the new sample belongs to based on the majority of its neighbours. Increasing K increases the radius so more neighbours will be taking into account. This can both increase and decrease accuracy of the classifier. Finding the optimal K, is called hyperparameter tuning and is used to find optimal setting of the algorithm.

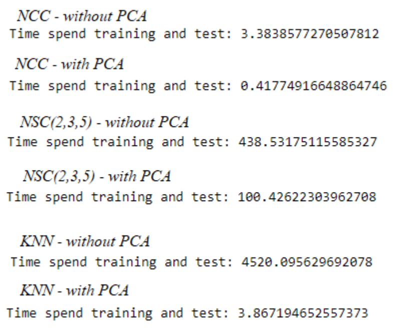
IV. Results

The results of the mentioned classifiers performance will be shown in percentage. Here the percentage will depict the correctness of the classifier and how well it was to classify the test data set.

Algorithms	Classification correctness in %			
	MNIST	MNISTPCA	ORL	ORLPCA
NCC	80.30	43.06	92.5	10
NSC(2)	85.01	12.39	95	5
NSC(3)	87.21	11.16	95	7.5
NSC(5)	89.72	11.08	97.5	5
NN	97.04	10.99	97.5	5

TABLE I
TABULAR OF THE RESULTS FROM EACH CLASSIFIERS

On table I the results for each classifier performance can be seen. It can bee seen that most classifiers performed well, ranging from 80-95% correctness. The ORL data shows very accurate classification but might be due to the low amounts of test samples. As mentioned the ORL data set only consisted of 400 samples and might contribute to less diversity. It can be seen that NN did perform the best compared to the other classifiers. But while NN did perform the best, it was also the most time comsuming classifier.



```
NCC - without PCA
Time spend training and test: 3.3838577270507812
NCC - with PCA
Time spend training and test: 0.41774916648864746
NSC(2,3,5) - without PCA
Time spend training and test: 438.53175115585327
NSC(2,3,5) - with PCA
Time spend training and test: 100.42622303962708
KNN - without PCA
Time spend training and test: 4520.095629692078
KNN - with PCA
Time spend training and test: 3.867194652557373
```

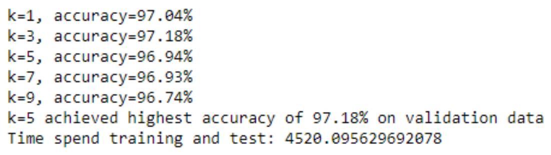
Fig. 6. Time performance of each classifier using the regular data set and the PCA transformed data set

Figure 6 shows the time performance of each classifier with the given data set. It can be seen that NN is much slower compared to the other classifiers. It is worth mentioning, the NN classification was done with KNN with K going from 1 to 30 and might therefor contribute to the time consumption. But it can also been that the PCA transformation did reduce the time consumption significantly.

When looking at the time analysis of each classifier, the classifications using the PCA transformed dataset did the training and testing in a significantly lower time for the MNIST dataset. The PCA did not improve the time consumption for the ORL that significantly due to small data set. However, when looking at the MNIST dataset time consumption can be reduced significantly but this comes at the cost of precision. While saving time is training these algorithms, the developer must still ensure functionality, so sacrificing 40-90% accuracy is not feasible. So tweaking the PCA transformation to contain more variables so a more desirable accuracy might be suitable solution.

V. Discussion

While it was not shown the results of the nearest neighbour can vary in performance. If KNN was performed instead of NN for the MNIST dataset an even greater performance can be achieved. As NN can be described as KNN with $k = 1$, an attempt with different K's was performed to see the influence in precision.



```
k=1, accuracy=97.04%
k=3, accuracy=97.18%
k=5, accuracy=96.94%
k=7, accuracy=96.93%
k=9, accuracy=96.74%
k=5 achieved highest accuracy of 97.18% on validation data
Time spend training and test: 4520.095629692078
```

Fig. 7. Performance of different K-values for KNN

On figure 7 the results of different K's for the MNIST dataset can be seen. The figure shows that $K = 1$ did not achieve the highest accuracy. While the increase in accuracy is very miniscule it is still a slight increase. However, to compare different value of K, NN has to be run for the data set for each, which may take a substantial amount of time. The figure above shows it took 4520 seconds which roughly translates to 1 hour and 15 minutes. This could be even worse for bigger data set.

The classification of the PCA transformed data did not perform well, a significant reduction is time was seen. The PCA performed in this paper, reduced the variables down to 2, which might have resulted in a great loss of variance in the data. A slightly less reduction of variables may have resulted in better performance for the classifiers, but due to shortage of time, a new set of training and test was not done.

VI. Conclusion

The classification was implemented successfully in Python using Jupyter notebook. While it wasn't the easiest tool to debug, a successful implementation was still achieved. The overall performance of each classifier used in this paper performed really well with a correctness between 80 to 95%, when handling the non-changed data set. The classifiers were able to recognize patterns from the images and thereby decide which class an image may belong to. What is worth noting, is that the NN performed slightly better compared to the other classifiers. Another benefit of using NN or KNN is that it performs well with multi-modal classes since its bases its decision on it's neighbours of similar objects. However, while the KNN or NN is very accurate it is very slow in performance as the process takes some time depending on the data size. The algorithm did however perform poorly when handling the PCA analysed data, in terms of correctness, but was due to loss of variance. However the PCA did reduce the overall time spent training and test, so with slightly less reduction in dimensions, a compromise could be made, where the implementation might not be as accurate but takes significantly less time to implement.

References

[1] https://scikit-learn.org/stable/ The different Frameworks can be found in https://scikit-learn.org/stable/supervised_learning.html#supervised-learning - Accessed 20/11/2020

[2] The description of the MNIST data set can be found in http://yann.lecun.com/exdb/mnist/ - Accessed 20/11/2020

[3] While the official ORL data set homepage can't be found a description can be found in https://cam-orl.co.uk/facedatabase.html - Accessed 20/11/2020

[4] The image and description of PCA can be found on: https://en.wikipedia.org/wiki/Principal_component_analysis - Accessed 21/11/2020

[5] The image of the labelled MNIST images can be found on: https://towardsdatascience.com/improving-accuracy-on-mnist-using-data-augmentation-b5c38eb5a903 - Accessed 21/11/2020

[6] The image can be found on: https://www.researchgate.net/figure/Samples-of-ORL-face-database_fig3_264167711 from the article: https://www.researchgate.net/publication/264167711_Face_Recognition _Based_on_Improved_Fuzzy_RBF_Neural_Network_for_Smar_t_Device - Accessed 21/11/2020

[7] Documentation for the KMeans cluster method can be found on https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html - Accessed 21/11/2020

[8] A description of NCC can be found here https://machinelearningmastery.com/nearest-shrunken-centroids-with-python/ - Accessed 21/11/2020

[9] a description of KNN can be found on https://scikit-learn.org/stable/modules/neighbors.html - Accessed 23/11/2020

[10] Image found on https://medium.com/@sonish.sivarajkumar/k-nearest-neighbours-knn-algorithm-9900c1427726 - Accessed 23/11/2020