Quantitative and qualitative performance analysis of three different Machine Learning classification schemes

György Ihász

***Abstract*—Machine learning is an important field of modern information technology. However, many of the learning algorithms are only optimal for certain types of datasets and each of the algorithms can be customized with specific parameters. In this paper, I will compare the nearest class centroid classifier, the nearest sub-class centroid classifier (using number of subclasses in the set {2, 3, 5}) and the nearest neighbour classifier. All the three classification schemes will be measured on a precision scale, meaning, after successfully training the model, what is the chance of that it will successfully classify the unknown data. The analysis will be done using Python and Jupyter notebook tool.**

I. Introduction

THIS document is intended to provide an overview of the performance comparison between three classification schemes. Classification is the act or process of classifying or systematic arrangement in groups or categories according to established criteria. [1] In the terminology of machine learning, classification is considered an instance of supervised learning, i.e., learning where a training set of correctly identified observations is available. An algorithm that implements classification, especially in a concrete implementation, is known as a classifier. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category. [2]

This document will consider the following three classification schemes as a subject of performance comparison:

1. Nearest class centroid classifier
2. Nearest sub-class centroid classifier
3. Nearest neighbour classifier

II. Math

In this section, each of the classification schemes will be discussed in a mathematical perspective. This means that the algorithm will be described as mathematical expressions.

A. Nearest class centroid classifier

In machine learning, a nearest centroid classifier or nearest prototype classifier is a classification model that assigns to observations the label of the class of training samples whose mean (centroid) is closest to the observation. [3] „Given a set of N samples, each represented by a vector $x_i \in R^D$, and the cor- responding labels l_i we can define three classifiers that can be used in order to classify a new (unknown) sample $\mathbf{x}_* \in R^D$. The first one is called Nearest Class Centroid (NCC) classifier. NCC represents each class c_k with the corresponding mean class vector:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i,l_i=k} \mathbf{x}_i,\ k = 1, \ldots, K.$$

Given the class mean vectors, $\mathbf{x}_*$ is classified to the class ck corresponding to the smallest distance:

$$d(\mathbf{x}_*, \boldsymbol{\mu}_k) = \|\mathbf{x}_* - \boldsymbol{\mu}_k\|_2^2.$$

The NCC classifier corresponds to the case where we set the assumption that each class is unimodal and follows a Normal Distribution." [4]

B. Nearest sub-class centroid classifier

„For the cases where each class forms several subclasses, a variant of NCC called Nearest Subclass Classifiers (NSC) can be used. NSC assumes that each subclass m of class c_k follows a Normal Density and is represented by the mean subclass vector $\boldsymbol{\mu}_{km}$:

$$\boldsymbol{\mu}_{km} = \frac{1}{N_{km}} \sum_{i,l_i=k,q_i=m} \mathbf{x}_i.$$

Here we use the label q_i in order to denote the subclass label of vector x_i and N_{km} to denote the number of samples forming subclass m of class c_k. Given the subclass mean vectors, $\mathbf{x}_*$ is classified to the class c_k corresponding to the smallest distance:

$$d(\mathbf{x}_*, \boldsymbol{\mu}_{km}) = \|\mathbf{x}_* - \boldsymbol{\mu}_{km}\|_2^2.$$

In order to define the subclasses of each class, we usually apply a clustering algorithm (e.g. K-Means) using the samples of the corresponding class. The number of subclasses per class is a parameter of the NSC and needs to be decided beforehand." [4]

C. Nearest neighbour classifier

„In the extreme case where the number of subclasses per class is equal to the number of its samples, then the resulting classifier is called Nearest Neighbor (NN) classifier. That is, NN classifier classifies $\mathbf{x}*$ to the class of the sample closest to it." [4]

III. Methods

All the classifiers will be measured on accuracy with both the MNIST and ORL datasets. While the MNIST dataset is used with a 6:1 ratio, the ORL is used with a 70:30 train-test

ratio. This means that the models are trained i.e.: fitted with around 85% of the total data in case of the MNIST dataset and 70% in case of the ORL dataset. The test samples are picked randomly from the original dataset. For separating the random test samples from the originals, I used the train_test_split method from the sklearn.model_selection library. [5] Due to the lack of hardware resources, only 3000 records are used from the MNIST database.

Additional modulation is involved in the measurements. The Principal Component Analysis (PCA). Principal Component Analysis seeks a space of a lower dimensionality, known as the principal subspace and denoted by the magenta line, such that the orthogonal projection of the data points (red dots) onto this subspace maximizes the variance of the projected points (green dots). An alternative definition of PCA is baed on minimizing the sum-of-squares of the projection errors, indicated by the blue lines. [6]

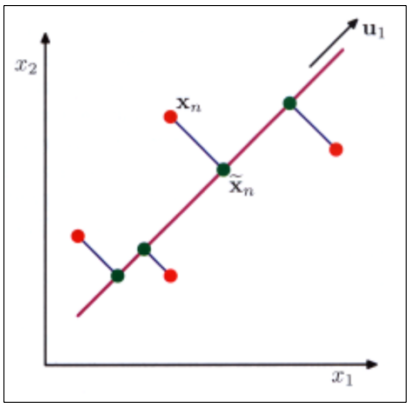

Figure III.1 - Principal Component Analysis (PCA) [6]

The code is written with the language Python. The integrated development environment used in this project is Visual Studio Code. The python code was operated through the default terminal on an instance of the operating system macOS. The graphs and figures shown in section IV are all exported with the default screen capture tool.

IV. Results

%	K1	K5	K19	C	S2	S3	S5
MNIST	**92**	**88**	**86**	**74**	**83**	**82**	**79**
ORL	**95**	**85**	**59**	**92**	**92**	**93**	**93**
M_PCA	**36**	**40**	**45**	**42**	**40**	**38**	**35**
O_PCA	**37**	**42**	**30**	**36**	**38**	**38**	**37**

Table IV.1 – Accuracy measurements of the different classifier models in percentage.

The columns K1, K5 and K19 are K-Nearest Neighbour classifiers with different neighbour-count values. The numbers on the right of each K are the number of nearest neighbours that are checked when the classification happens.

The column C represents the Nearest Centroid Classifier. The S2, S3 and S5 columns represent Nearest Sub-Class Centroid Classifier with different number of sub-classes. The number on the right of each S represent the number of sub-classes.

The values in the table are the accuracies. The accuracy is calculated by dividing the predicted true elements by the number of total test records. For the sake of example, lets consider 10 test records and each record has the true label. The classifier predicts the class of 8 records successfully, so this means 8/10 = 0.8. The model in the example has the accuracy of 80%.

The left column represents the datasets. M_PCA and O_PCA are the MNIST and ORL datasets respectively, transformed with PCA. This means, that their dimensionality is reduced to 2. Both datasets are image datasets, thus every record in these is a single image. A single record holds the features of the data. In case of images, every record has equal number of features as the number of pixels in the image. The MNIST dataset contains 70000 images of hand-written digits, and each image has a resolution of 28x28 pixels, thus 784 features per record. As opposed to the ORL dataset, where each image has a resolution of 30x40 pixels, so a single record holds 1200 features. It is definitely hard if not possible to visualize these datasets, however, with PCA, we can reduce the dimensionality to 2, and then it is possible to plot the data. This can be seen on Figure IV. 1

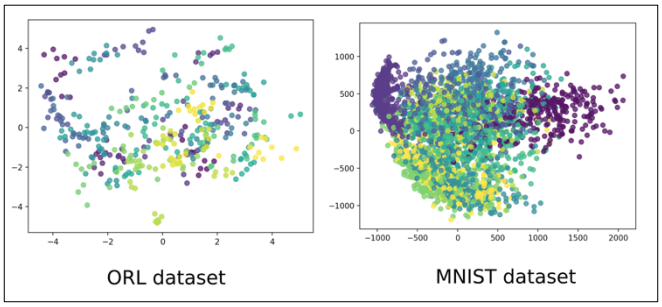

Figure IV.1 The ORL and MNIST datasets visualized after transforming both with PCA

The accuracy measurements are not different from the ones expected. Each classifier has a smaller or a more significant tendency in the direction of producing a better accuracy score. Consider the graphs shown in Figure IV. 2.

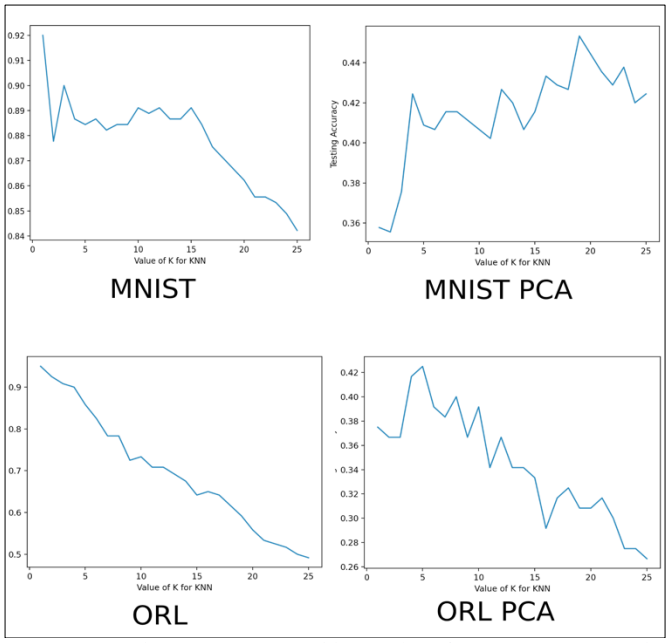

Figure IV.2 The K-Nearest Neighbour Classifier accuracies tested on different datasets with different number of neighbours considered.

The KNN with the dataset MNIST transformed by PCA has a tendency to increase performance in parallel with the increasing number of neighbours taken into account. What is interesting is that though both datasets are based on images, the ORL dataset with reduced dimensionality behaves in a completely different way.

By comparing the following figures, we can see the slight tendencies in each classifier with different parameters.

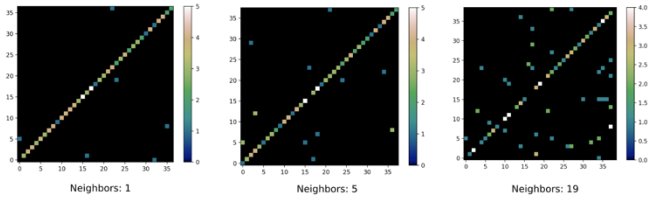

Figure IV.3 – Confusion matrices of KNN Classifiers with 1, 5 and 19 neighbour parameters with the default ORL dataset

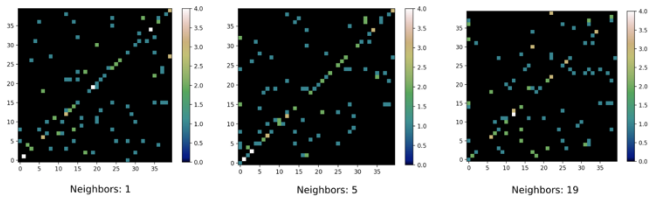

Figure IV.4 - Confusion matrices of KNN Classifiers with 1, 5 and 19 neighbour parameters with PCA-transformed ORL dataset

The confusion matrix represents the wrong predictions where both the actual true value and the wrong prediction can be seen simultaneously. Also, we can see that after a PCA transformation, the accuracy drops. It is almost independent of the number of neighbours.

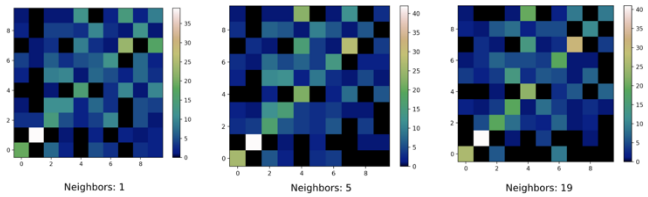

Figure IV.5 - Confusion matrices of KNN Classifiers with 1, 5 and 19 neighbour parameters with the PCA-transformed MNIST dataset

Here it can be seen that there is a really small tendency to improve accuracy. The confusion matrix and the graph in Figure IV.2 proves the same.

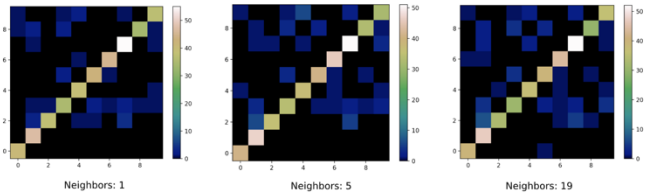

Figure IV.6 - Confusion matrices of KNN Classifiers with 1, 5 and 19 neighbour parameters with the default MNIST dataset

In figure IV.6, the number of wrongly predicted test records are increasing along with the number of neighbours involved.

The next 4 Figures are containing information related to the Nearest Sub-Class Classifier.

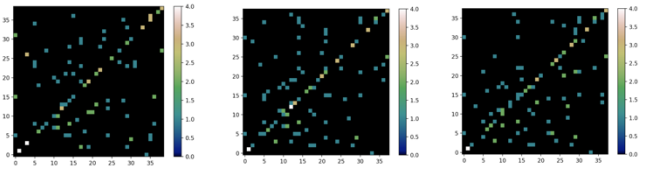

Figure IV.7 - Confusion matrices of Nearest Sub-Class Classifiers with 2, 3 and 5 subclasses. Classification is performed on the PCA-transformed ORL dataset.

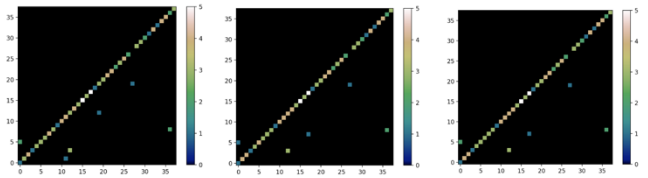

Figure IV.8 - Confusion matrices of Nearest Sub-Class Classifiers with 2, 3 and 5 subclasses. Classification is performed on the default ORL dataset.

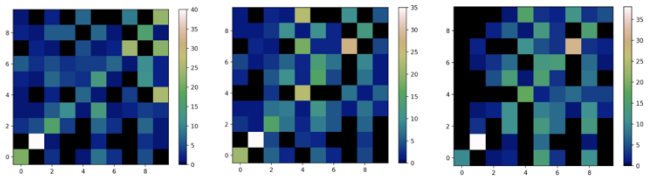

Figure IV.9 - Confusion matrices of Nearest Sub-Class Classifiers with 2, 3 and 5 subclasses. Classification is performed on the PCA-transformed MNIST dataset.

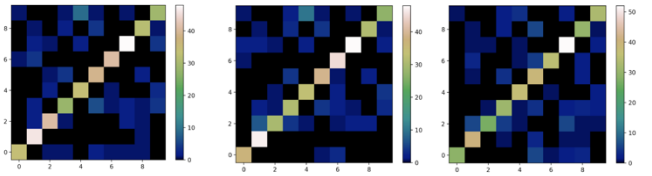

Figure IV.10 - Confusion matrices of Nearest Sub-Class Classifiers with 2, 3 and 5 subclasses. Classification is performed on the default MNIST dataset.

In Figure IV.10, there is an observable decrease in accuracy while we increase the ammount of sub-classes. Although the periferical guesses are getting better in Figure IV.9 with the increasing number sub-classes used, the overall accuracy is getting worse as stated in Table IV.1.

The following figure is representing the performance of the Centroid Classifier.

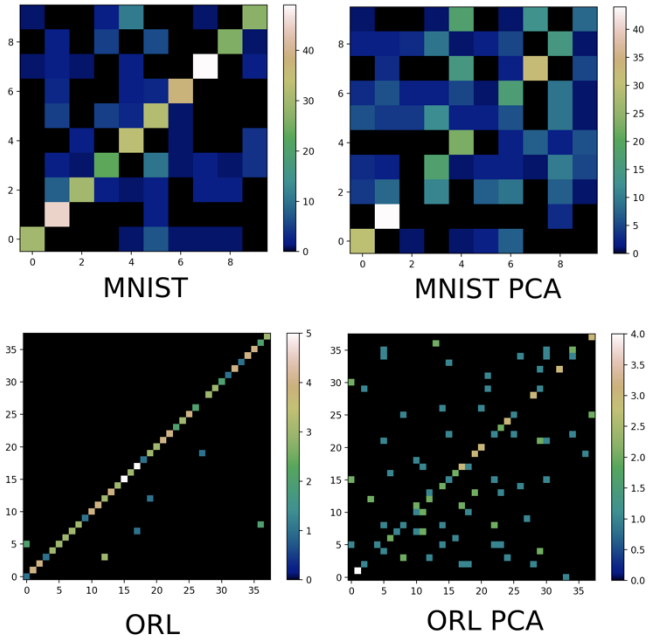

Figure IV.11 - The confusion matrices of the Centroid Classifier on both MNIST and ORL default and PCA-transformed datasets.

Figure IV.11 shows, that independently from the dataset, the PCA-transformation has made the performance worse. The Centroid Classifier provided a 92% accuracy on the default ORL dataset.

V. Conclusion

The accuracy measurements are perfectly representing how the different type of classification schemes can be applied to different datasets. There is a huge potential in machine learning, and this project is intended to be improved in the future. More computational resources will be involved, and a variety of datasets will be tested.

VI. References

[1] "Merriam Webster," [Online]. Available: https://www.merriam-webster.com/dictionary/classification. [Accessed 03 11 2020].

[2] "Wikipedia - Statistical classification," [Online]. Available: https://en.wikipedia.org/wiki/Statistical_classification. [Accessed 03 11 2020].

[3] "Wikipedia - Nearest centroid classifier," [Online]. Available: https://en.wikipedia.org/wiki/Nearest_centroid_classifier. [Accessed 03 11 2020].

[4] A. Iosifidis, "Introduction to Machine Learning," *Course notes,* 2018.

[5] s. learn, "sklearn.model_selection.train_test_split," scikit-learn developers, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html. [Accessed 23 11 2020].

[6] C. M. Bishop, "Pattern Recognition and Machine Learning," 2006, p. 226.