Optimization and Data Analytics

Martin Tomko
Aarhus University
Computer Engineering
Aarhus, Denmark
martin@tomko.dk

***Abstract*—The project purpose is to investigate various types of image classifications that uses machine learning. Compare their accuracy and performance. Focus is on Nearest Class Centroid, Nearest Subclass Centroid, Nearest Neighbor, Backpropagation Perceptron and Mean Squared Error Perceptron. Python is used as implementation language for classifiers and visualization. Classifiers are tested on 2 types of datasets, ORL and MNIST. ORL has facial images, MNIST uses hand written digits. In experiment are tested and compared full dimensional data also as reduced 2-Dimensional data of datasets.**

Keywords—classification types, neural network, data visualization, machine learning, image classification

I. Introduction

In computer technologies, classification of images is based on searching for patterns in image feature set which are represented by pixels and classify them into pre-defined categories. There may be many various of categories. For example, we can prepare algorithm that will classify images according if they contain person or from images of people, we can classify them according of age or if its man/woman.

This task can look very trivial for human being. Human brain excels at searching for patterns and classification of data (vision for example). But everything happens subconscious and its hard to tell how exactly it works. In computer technology we can use machine learning for tasks of this type.

Machine learning which from the name we can say is algorithm in which computer (machine) learn something. And later, based on learned data, make some decision. If we want to train some model to do some decision first, we need to train it. For that we need higher number of samples with pre-defined categories. For example, set of human facial images with label that represent category (young, middle, old). We can weed those data to machine learning and train our model so based on image he can predict if person is young. It is also important to note that various images may have different features. For example, different number of pixels or different spectrum of colors. And if we train our model based on grey-scale pictures with 50x50 pixels, it is important to use same features to make correct prediction. So, there may be some pre-processing of images involved.

There is many types of learning algorithm and decision function which may have different performance for different type of data. This is also a primary topic of this project where we test 5 types of Classifier to perform on various types of data.

II. Image datasets

In this project are used 2 types of datasets. Facial images ORL and hand-written images of digits MNIST

A. MNIST Dataset

Modified National Institute of Standards and Technology database (MNIST database) is large database that contains 60 000 training images and 10 000 testing images. MNIST is combination of 2 NIST's databases which consist of digits written by high school students and employees of the United States Census Buerau.

All images in MNIST are greyscale with a resolution 28x28 pixels, which make every image to have 784 features. Some examples of MNIST data are shown in figure 1.

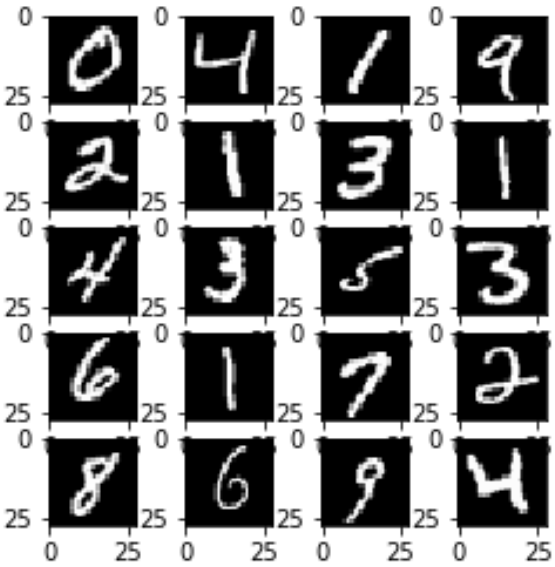

Figure 1 MNIST images examples

As we can see in Figure 1, there is various shape of number "4". The more samples we will feed model to train the better performance should be. There may be some confusion in classification for example between '4' and '9' or '1' and '7'. Because shapes are very similar.

B. ORL Dataset

ORL is facial image database which contains of images of 40 people. There are 10 images for every person which makes totally 400 samples in database. This dataset is not split to train data and test data like MNIST is. In the project we split this dataset into 70% as train data (280 samples) and

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

30% test data (120 samples). Pixels on ORL images are also transferred to greyscale with resolution of 30x40 pixels which makes totally 1200 features per image. Because of dataset is much smaller (400 samples) then MNIST dataset (70 000) samples. This may be seen on accuracy of our classifier.

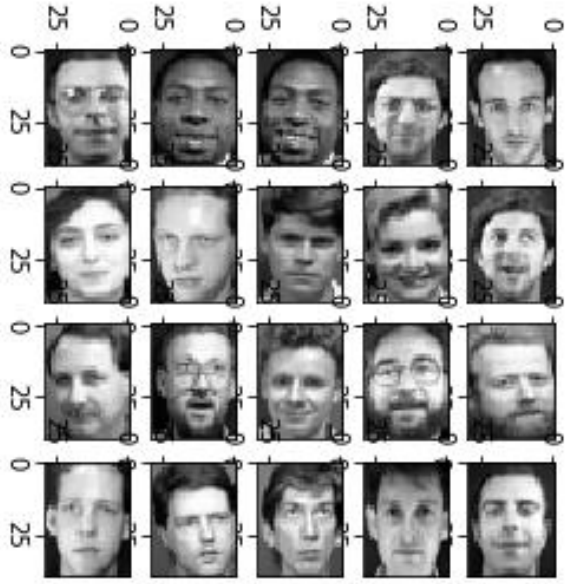

Figure 2. Examples of ORL images

In Figure 2. are shown examples of ORL images. Even for human being are those data much harder to classify. They contain more features and characteristic pattern for subject is not so clear as it was with MNIST data.

III. Principal Component Analysis

Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of correlated data to linearly uncorrelated data called principal components.

PCA is mostly used as a tool in data analysis and predictive models that uses machine learning. It is very useful step for visualizing and processing high-dimensional data and still retain as much of the variance as possible. It is important to note that by reducing dimensions of data samples we are losing information which may cause patterns harder to distinguish

A. Applying PCA on MNIST dataset

Because of MNIST data has many samples and only 10 categories, we can see clusters of categories very clear on Figure 3. Also, we can see that clusters for similar shapes like ‘1’ and ‘7’ or ‘4’ and ‘9’ are close to each other.

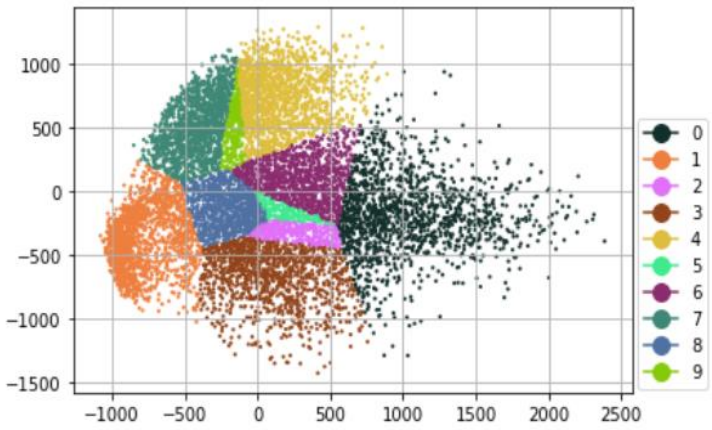

Figure 3. PCA of MNIST transformation

B. Applying PCA on ORL dataset

ORL dataset doesn’t have so many samples and has more categories. This can make some data entries put much further from each other as is shown on Figure 4. Some clusters have higher spread, and some are clustered and distanced from other clusters. Those clusters should have much higher rate of classifying data correctly because of much more characteristic pattern.

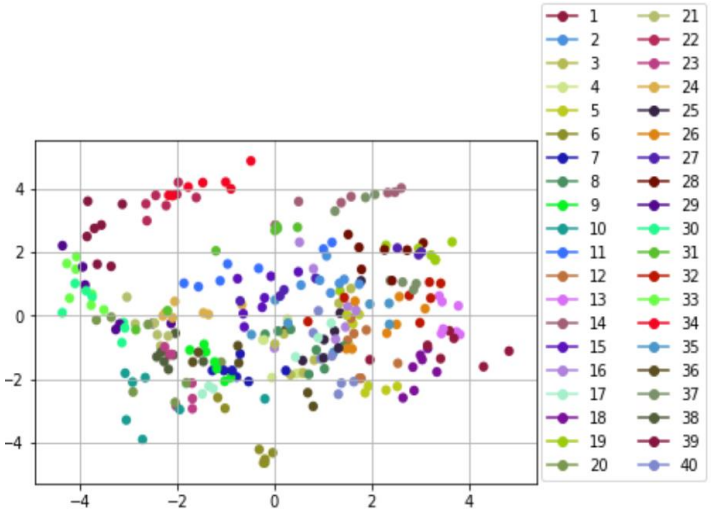

Figure 4. PCA of ORL data transformation

IV. Classifications

A. Nearest Class Centroid

Nearest Class Centroid (NCC) classifier is model that assigns the label of the class of training samples whose centroid (mean) is closest. So, for test samples, distance to each centroid is calculated and entry is assigned to one which lowest distance.

First, we need to calculate centroids for all possible classes as shown in [1] where X is our data set, Y class set, and C_l is set of samples belonging to class l.

$$\mu_l = \frac{1}{|C_l|}\sum_{i \epsilon C_l} X_i$$

Equation 1 NCC Centroids

For prediction function we need to find closes centroid as shown in [2].

$$y = \arg min_{l \epsilon Y} \left\| \vec{\mu_l} - \vec{X} \right\|$$

Equation 2 NCC Predict function

Performance of this classifier is good, results from test data are shown in table. Classifier excels on ORL 1200D data with classification accuracy of 95%.

NCC Classifier results			
MNIST 784D	**MNSIT PCA**	**ORL 1200D**	**ORL PCA**
82.03%	43.65%	95%	39.16%

Table 1 NCC Results

B. Nearest Subclass Centroid Classifier

The nearest subclass centroid classifier (NSC) is very similar to NCC, but in this case classes can be formed into multiple subclasses. So, model takes every class C_k and separate it into two or more subclasses m where every subclass has its own centroid (mean) μ_{km} [3], where N_{km} is number of samples in subclass, and q_i is label. Then, distance to each of centroid is calculated and data are classified by the closest centroid same as in NCC

$$\mu_{km} = \frac{1}{N_{km}} \sum_{i, l_i = k, q_i = m} x_i$$

Equation 3

Number of subclasses are passed to model as parameter. In this project we used values 2, 3, 5. Subclasses are created using K-means algorithm.

	NSC Classifier results			
sub	**MNIST 784D**	**MNSIT PCA**	**ORL 1200D**	**ORL PCA**
2	86.07%	43.01%	0%	3.33%
3	88.17%	42.49%	0%	4.16%
5	90.42%	41.82%	0%	1.66%

Table 2 NSC Results

In this experiment MNIST data performed well while using ORL data I could not get any accuracy. On table we can see that accuracy on MNSIT 784D data increased by increasing number of subclasses used by NSC on other side, reduced PCA data, there accuracy of classification started decreasing by adding more subclasses.

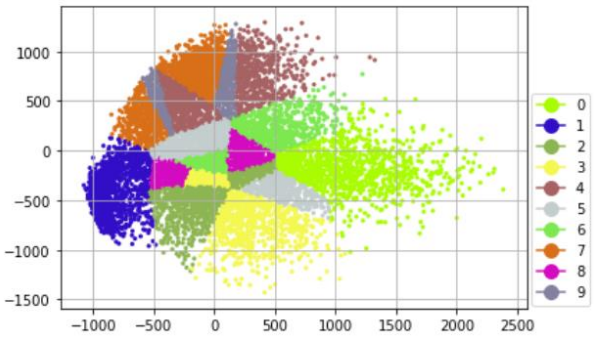

Figure 5 MNIST PCA of NSC with 2 subclasses

C. Nearest Neighbour

Nearest Neighbor (NN) classifier is advanced type of nearest subclass classifier where every class have same number of subclasses as there is samples in specific class. NN algorithm uses same method for calculating distance to nearest subclasses as previously mentioned NSC and NCC but because of possibility of enormous number of subclasses, this algorithm is very heavy for computational power.

In addition to this algorithm we can specify parameter *k* which means how many nearest neighbors should be considered. Parameter *k* should ne multiple number of number classes in our dataset. Let's say we have 3 classes in our dataset, and we set *k=3* in this situation tie can occur where all 3 nearest neighbor which we are considering are from different class.

	NN Classifier results			
k	**MNIST 784D**	**MNSIT PCA**	**ORL 1200D**	**ORL PCA**
1	96.91%	38.99%	97.5%	42.5%
2	96.91%	38.96%	95.83%	35.83%
3	97.17%	40.17%	95%	47.5%

Table 3 NN Results

In *Table 3* we can see results of using NN algorithm used on bot ORL and MNIST dataset. Computation power is heavy but results are very good. We can see that on non-reduced data results reached over 95% of accuracy classifying our test datasets. Tests were performed using *k = {1,2,3}*. We can see that mnist performed best considering 3 nerest neighbors wit haccuracy of 97.17% while for ORL data it was test where we considered only one nearest neighbor with accuracy of 97.5%. Results for OLR data are very impressive considering that we have only 400 samples of 40 subject as train data.

D. Backpropagation perceptron

To understand backpropagation perceptron (BPE), we first need to look at perceptron itself. In machine learning, perceptron is a classifier algorithm for supervised learning. Structure of perceptron is based on "neurons" and synapses (weight vectors) between them. Every perceptron has at least 2 layers of neurons. Input layer and output layer. Input layer has size of features our sample and output layer also depends on technique that we use. In this case of perceptron, we have input layer

directly connected by weights to his output layer. So, we have one weight vector which we will be during training adjusting to match our output.

In more sophisticate perceptron we can add more hidden layers which are positioned between our input layer and output layer. In this case we will have set of weight vectors which connects layer by layer from input through hidden layers into output layer. If 1 or more hidden layers are present, we need to use more advanced algorithm for training and one of them is backpropagation. Backpropagation efficiently calculates gradient of loss function with respect to weights vectors for single input-output sample.

Output of algorithm depends on weighted sum of all its input as shown in *[4]* where *w1* and *w2* are weights of synapses. The loss function maps values of one or more variables into real number representing some "cost" associated with values. In propagation algorithm it calculates a difference between network output and expected output *[5]* where *y* is network output and *y'* is expected output. In addition to this algorithm we can add *Learning rate* parameter which will adjust size of changes in our weight vector per iteration, same as iterations itself and, we can specify size of our hidden layer.

$$y = x_1 w_1 + x_2 w_2,$$

Equation 4 Output formula

$$E(y, y') = \frac{1}{2}\|y - y'\|^2$$

Equation 5 Loss function formula

BPE Classifier results			
MNIST 784D	**MNSIT PCA**	**ORL 1200D**	**ORL PCA**
92.21%	44.26%	0%	1.66%

Table 4 BPE Classification results

From experiments using backpropagation algorithm we can see on *Table 4* that classifier also performed very well on MNIST data with accuracy of 92.21% on original data and 44.26% reduced PCA data. ORL data again performed very poorly with almost 0 percentage accuracy. For this test we used learning rate 0.1, 2500 iterations and hidden layer of 50 neurons.

E. MSE perceptron

Mean-Squared-Error perceptron tries to minimize classification error by using MSE instead of the samples with wrong classification. In this case we optimize *w* by mapping training vectors *X* $[x_1...x_n]$ to arbitrary target values *b* $[b_1..b_n]$ *[6]*

$$\mathbf{X}^T\mathbf{w} = \mathbf{b}$$

Equation 6

We optimize *w* by minimizing MSE and that corresponds to [7].

$$\mathcal{J}_s = \sum_{i=1}^{N}(\mathbf{w}^T x_i - b_i)^2 = \|\mathbf{X}^T\mathbf{w} - \mathbf{b}\|_2^2$$

Equation 7 Optimizing MSE

MSE Classifier results			
MNIST 784D	**MNSIT PCA**	**ORL 1200D**	**ORL PCA**
86.02%	32.48%	0%	1.66%

Table 5 MSE results

In *Table 5* we can see results from experiment using implemented MSE classifier. Again, classifier performed quite well on MNIST data but poorly again on ORL data. MNIST data reached accuracy of classifying 86.02% for original data and 32.48% for reduced PCA. For ORL it was almost 0%.

V. Implementation information

Classifiers are implemented in python using "Jupyter lab" which is open source web-based development environment for jupyter notebooks. Jupyter notebooks is interface that support various workflows and visualization that are used in data optimization and visualization. Some classifications are used from library "scikit-learn" while BPE and MSE is written in python from scratch. All implementation can be found in ZIP file as appendix to this project.

VI. Conclusion

For MNIST data all classifications performed very well and we reached more than 80% in each classifier tested. Reduced PCA version of data received in classification and that is around ~40%. From the results we can see that using PCA will make much easier to visualize data because we are working with 2D data instead of 1200D for example. But we will loose many features there and classification accuracy is also very reduced. ORL data on other side didn't got any results some of the classifications. They performed very well on Nearest Neighbor and Nearest Class Centroid but on the others we got ~0% accuracy. It would be expected on backpropagation algorithm because this one is mostly usable on linear data, while in ORL dataset of facial images we have small amount of samples with many subject. And patterns there are not linear. But for the MSE algorithm some higher accuracy was expected.

Best results were provided by Nearest neighbor classifier which is also computing power heaviest. NN classified MNSIT data with accuracy of 97.17% considering 3 nearest neighbors and ORL data with accuracy of 97.5% considering single nearest neighbor. Those results are very impressive.

References

[1] Wikipedia.org, “MNIST Database”, 20.11.2019. [Online]. Available: https://en.wikipedia.org/wiki/MNIST_database

[2] Wikipedia.org, “Principal component analysis”, 20.11.2019. [Online]. Available: https://en.wikipedia.org/wiki/Principal_component_analysis

[3] Wikipedia.org, “Nearest centroid classifier”, 20.11.2019. [Online]. Available: https://en.wikipedia.org/wiki/Nearest_centroid_classifier

[4] Wikipedia.org, “K-nearest neighbors algorithm”, 21.11.2019. [Online]. Available: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[5] Wikipedia.org, “Perceptron”, 22.11.2019. [Online]. Available: https://en.wikipedia.org/wiki/Perceptron

[6] Wikipedia.org, “Backpropagation”, 22.11.2019. [Online]. Available: https://en.wikipedia.org/wiki/Backpropagation

[7] Jupyter.org, “Jupyter lab”, 23.11.2019. [Online]. Available: https://jupyter.org/

[8] Wikipedia.org, “Mean squared error”, 20.11.2019. [Online]. Available: https://en.wikipedia.org/wiki/Mean_squared_error