Optimization and Data Analytics
A research project on image classification

Theo Morales [201703024]

***Abstract*—This research paper treats different optimization algorithms applied on the classical image classification problem. The algorithms are implemented in C++, using the Eigen library for linear algebra, and applied on two different datasets: the MNIST dataset of handwritten digits (grayscale images), and the ORL dataset of faces (grayscale images). The results of the different algorithms are visualized by applying the Principal Component Analysis and plotting the 2D data. Throughout this paper, the algorithms in question will be described, then compared based on their execution times and their success rates.**

***Keywords*—Optimization, machine learning, image classification.**

I. Introduction

The image classification problem is a very common case of study when it comes to machine learning and data analytics. There is a substential amount of different algorithms that can be used to treat this problem, however, the lack of computational power has slowed down the study of those. Nowadays, this is no longer a problem, as computers have become significantly faster than when these algorithms were thought and designed, and it is now a lot easier to compare them in depth. Machine learning is becoming a trend, and as it can be utilized on affordable hardware, it can be applied to a large variety of problems.

This study focuses on classification of two sets of images: the MNIST dataset, which is a collection of 70000 images representing handwritten digits from 0 to 9, in grayscale, and the ORL dataset, containing 400 grayscale facial images, representing a total of 40 different individuals.

The following optimization algorithms will be applied on the said data samples:

- Nearest Centroid Classifier
- Nearest Sub-class Centroid Classifier
- Nearest Neighbour Classifier
- Perceptron trained using Backpropagation
- Perceptron trained using Mean Square Error

In order to establish a benchmark for these classifiers, their computation time and their accuracy will be compared, and they will be applied on the two datasets reduced to two dimensions after applying the Principal Component Analysis (PCA), which will also allow for the plotting of the results.

II. Methods

The methods, or classification algorithms in this case, will be described and explained in this section.

A. Nearest Centroid Classifier

This first algorithm is rudimentary and very simple to understand. Firstly, during the training phase, a mean vector is calculated for each class, by averaging all the training samples of the given class. Then, in the classification phase, each test sample is classified by calculating its euclidean distance to each mean vector: the lowest distance indicates the nearest class for the test element. The euclidean distance formula is:

$$||x_n - m_c||_2^2 \tag{1}$$

with x_n being the tested sample of index n, and m_c the mean vector of class c. The distance is calculated via the l_2-norm of the substraction of the two vectors, to the power of two.

B. Nearest Sub-class Centroid Classifier

This classifier is basically an enhanced version of the Nearest Centroid Classifier, where the K-means algorithm has been applied in the training phase. This results in a clustered class, which holds N mean vectors corresponding to N sub-classes (clusters) instead of just one mean class vector. The number of sub-classes N is, theoretically, giving better and more accurate results for a higher value, as it yields more choices for the testing phase.

The classification of elements is done the same way as for the Nearest Centroid, however each test sample is compared to each mean sub-class vector of each class. The number of sub-classes is to be fine-tuned empirically, depending on the desired precision/speed ratio.

C. Nearest Neighbour Classifier

The last euclidean distance-based classifier, Nearest Neighbour, is the most time-consuming and resource-demanding algorithm, but in theory one of the most accurates. This classification method demands no training phase, as the classification process is very straight-forward. To classify a test sample, its distance to each individual training sample is computed using the euclidean distance, and the lowest is then used to give it the class of the training element, accordingly. This algorithm doesn't introduce any interesting aspects in terms of classification, but is still interesting to use for comparison ends.

D. Perceptron trained using Backpropagation

A perceptron is a linear discriminant method of classification, that uses a discriminant function to determine wether a sample is belonging to a given class or not: it is a binary

classifier. As one perceptron can only be used for one given class, several of them will be used to form a neural network, that will them be fed input data such as the aforementioned ORL image vectors, in order to train each perceptron to accept its given class and to reject every other. The discriminant function can be represented as:

$$g(\mathbf{w}) = \mathbf{w}^T\mathbf{x} + \omega_0 \tag{2}$$

with $\mathbf{w}$ being the weights vector, and ω_0 the bias, together constituting the discriminant. When the result of this function is above 0, the given $\mathbf{w}$ vector can be classified as belonging to the perceptron's class, otherwise it doesn't.

During the training phase, the weight vectors have to be computed in the optimal way, so that they will be able to classify an input with the best success rate. In order to do so, a perceptron is fed with input vectors belonging to all the classes of the training set, and the following criterion function is used to update the weights vector of one perceptron:

$$\mathcal{J}_p(\mathbf{w}) = \sum_{x_i \in \mathcal{X}} -l_i \tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}_i \tag{3}$$

with χ being the set of misclassified samples for the perceptron, l_i being the binary label of the misclassified sample, $\tilde{\mathbf{w}}$ corresponding to the augmented weights vector, and $\tilde{\mathbf{x}}_i$ the misclassidied training sample. This optimization problem can be solved using the gradient descent method as follow:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \sum_{x_i \in \chi} -l_i \mathbf{x}_i \tag{4}$$

with η being the learning rate, which defines the step size of the gradient descent, also referred to as hyperparameter.

Finally, during the classification phase, the aforementioned function $g(X)$ is applied to each input vector, and the binary result is used to determine which class the input sample belongs to.

E. Perceptron trained using Mean Square Error

This last classifier is essentialy similar to the previous perceptron, except that it treats the criterion function as a quadratic problem and directly tries to solve it. This training method for the perceptron yields better results for cases where the data classes are not linearly separable. The criterion function can be written as:

$$\mathcal{J}_p(\mathbf{w}) = ||\mathbf{X}^T\mathbf{w} - \mathbf{b}||_2^2 \tag{5}$$

with $\mathbf{X}$ the matrix of training sample vectors, and $\mathbf{b}$ containing the output labels for the training samples.

III. Results

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

IV. Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Appendix A
Proof of the First Zonklar Equation

Some text for the appendix.

Acknowledgment

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

John Doe Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.