Neural Networks 2017: Assignment 1

Wojtek Kowalczyk
`wojtek@liacs.nl`

14 February 2017

Introduction

The objective of this assignment is to develop and evaluate several algorithms for classifying images of handwritten digits. You will work with a simplified version of the famous MNIST data set: a collection of 2707 digits represented by vectors of 256 numbers that represent 16x16 images. The data is split into a training set (1707 images) and a test set (1000 images). These data sets are stored in 4 files: `train_in.csv`, `train_out.csv`, `test_in.csv`, `test_out.csv`, where `_in` and `_out` refer to the input records (images) and the corresponding digits (class labels), respectively. These files are stored in `data.zip`.

You may find more information about the original problem of handwritten digit recognition, more data sets and an overview of accuracies of best classifiers (it is about 99.6%!) at `http://yann.lecun.com/exdb/mnist/`.

Task 1: Analyze distances between images

The purpose of this task is to develop some intuitions about clouds of points in highly dimensional spaces. In particular, you are supposed to develop a very simple algorithm for classifying hand-written digits.

Let us start with developing a simple distance-based classifier. For each digit d, $d = 0, 1, \ldots, 9$, let us consider a cloud of points in 256 dimensional space, C_d, which consists of all *training* images (vectors) that represent d. Then, for each cloud C_d we can calculate its center, c_d, which is just a 256-dimensional vector of means over all coordinates of vectors that belong to C_d. Once we have these centers, we can easily classify new images: to classify an image, calculate the distance from the vector that represents this image to each of the 10 centers; select as a label the closest one. But first let us take a closer look at out data.

For each cloud C_d, $d = 0, 1, \ldots, 9$, calculate its center, c_i, and the radius, r_i. The radius of C_d is defined as the biggest distance between the center of C_d and points from C_d. Additionally, find the number of points that belong to C_i, n_i. Clearly, at this stage you are supposed to work with the training set only.

Next, calculate the distances between the centers of the 10 clouds, $dist_{ij} = dist(c_i, c_j)$, for $i, j = 0, 1, \ldots 9$. Given all these distances, try to say something about the expected accuracy of your classifier. What pairs of digits seem to be most difficult to separate?

Task 2: Implement and evaluate the simplest classifier

Now, given the centers of 10 clouds, you should implement the simplest distance-based classifier that was described above. Apply your classifier to all points from the training set and calculate the percentage of correctly classified digits. Do the same with the test set, using the centers that were calculated from the training set. In both cases, generate a *confusion matrix* which should provide a deeper insight into classes that are difficult to separate. A confusion matrix is here a 10-by-10 matrix (c_{ij}), where c_{ij} contains the percentage (or count) of digits i that were classified as j. Which digits were most difficult to classify correctly? For calculating and visualising confusion matrices you may use the `sklearn` package. Describe your findings. Compare performance of your classifier on the train and test sets. How do the results compare to the observations you've made in Step 1? How would you explain it?

So far, we have implicitly assumed that you have measured the distance between two vectors with help of the Euclidean distance measure. However, this is not the only choice. Rerun your code using alternative distance measures that are implemented in `sklearn.metrics.pairwise.pairwise_distances`. Which distance measure provided best results (on the test set)?

Task 3: Implement a Bayes Rule classifier

Now approach the problem in the same way as in the `"a" or "b"?` example that was discussed during the course. To keep things simple, limit yourself to images of two digits only, for example, `5` and `7`. Which feature do you want to use to discriminate between the two classes? Create corresponding histograms and calculate the terms $P(X|C)$ and $P(C)$ so you could apply the Bayes rule to both the train and the test set. What accuracy have you achieved?

Task 4: Implement a multi-class perceptron algorithm

Implement (from scratch) a multi-class perceptron training algorithm (to be discussed on 15 Feb.) and use it for training a single layer perceptron with 10 nodes (one per digit), each node having 256+1 inputs and 1 output. Train your network on the train set and evaluate on both the train and the test set, in the same way as you did in the previous steps.

Task 5: Find the best possible model with existing software

This is probably the most rewarding, but also time consuming activity. Study the first two chapters of the book `http://neuralnetworksanddeeplearning.com/index.html` and play with the software that is discussed in these two chapters. When you mastered this software, modify and apply it to our original problem: find the best classifier, trained on our small train set (with 1707 records) and test it on the test set (with 1000 records).

Summarize your results in a table: the algorithms that you tried, key parameters you used, the obtained accuracies on the train and test sets.

Organization

Your solution should consist of a report in the pdf format (at most 10 pages) and neatly organized code that you've used in your experiments (but no data) so we could reproduce your results.

More details about the submission procedure and evaluation criteria will follow soon–we are working on a new method that will require your substantial involvement in the evaluation process!

The **hard deadline** for this assignment is **Monday, 13th March, 9:00AM**.