Multi-class classification

Alexandre Allauzen
Nicolas Pécheux

2015

This assignment will be graded and is due Tuesday March 10 at 23h59. You should write a short report (up to 4 pages, additionnal pages are allowed for figures) explaining what you have done, showing your results, providing answers to the questions and insightful comments for any difficulty you encountered. You may work by pairs and submit joint work. Send your report only as a *Portable Document Format* `lastname1_lastname2.pdf`. Your code will not be assessed, but should be provided packaged as a bitstream archive `lastname1_lastname2.tar.gz`. Send your submission to `elena.knyazeva@limsi.fr`.

1 Overview

The MNIST dataset is rather designed for multiclass classification. The goal of this assignment is to set up a classifier that takes as input an image and that outputs the label (which digit is associated to this image).

2 One against all classification with the perceptron algorithm

The perceptron, on the other hand, is a binary classifier: it can only assign a binary label (a yes or no answer). Last week, you saw that we can train a perceptron to recognize one digit, for example, you can train a perceptron classifier that can assign to an image the label yes or no (+1 or -1), whether the image corresponds to the digit 7 or not. Since there are ten classes (the possible label for an image is an integer between 0 and 9), a solution is to train ten classifiers, one per possible class. Each classifier is therefore associated to one particular digit and answer yes or no for its digit. For instance, there will be one classifier that can recognize the digit 0, one for the digit 1, ... , and one for the digit 9.

For inference, we can ask to each classifier what is its decision for the input image. The final answer corresponds then to the classifier that is the more confident. The confidence score of the perceptron can be derived from the value

of the dot product between the input and the parameter vector. The higher this score is, the more the classifier recognizes the digit.

3 Training phase

The first step aims at training 10 perceptron classifiers, one per digit. **You can adapt or wrap the code wrote for the previous assignment (binary classification on MNIST data).** Of course, to train the classifiers you will use the training part of the data.

Question 1. To monitor the learning process, plot the learning curve for each classifier. The learning curve can be defined as follows: during each epoch, count the number of misclassified training examples and plot this number of errors on the y-axis *vs* the number of epochs on the x-axis.

Try with different amount of training data (for example 100 and 1000). Make sure that you the learning process is finished (the convergence is reached). Don't forget to store the parameters of each classifier for both training configurations, we will need them for the next part.

Question 2. If your implementation is correct, when training on 1000 images, lots more epochs are necessary to reach the convergence than with 100 training images. Why?

Question 3. Try now with 10000 training images. How long does it takes to reach convergence? You may try different values for the learning rate η (for example, 0.1, 0.01 or $\eta = \frac{1}{i}$ where i is the epoch).

4 One against all: implementation

Since ten binary classifiers are now trained, the next step consists in the implementation of the *one against all strategy*.

Question 4. Compute the overall error rate of this strategy on the training corpus (i.e. the one you just used), as well as the the error rate for each digit. How does the error rate change when moving from 100 to 10000 examples? Do you think this is positive? Which one would you use as the "best" classifier?

5 Testing phase

Evaluate now the *one against all* classifier on the *test data*.

Question 5. Plot the evolution of the error rate (on the test set!) with respect to the number of training epochs, for 100, 1000 and 10000 training examples. How does the error rate on the *test set* change when moving from 100 to 10000 examples? Does this change your answers to the question 4?

Question 6. Compute the confusion matrix for the tree training configurations. In this matrix the cell $c(i, j)$ represents the number of images that are classified as digit i, whereas the good answer was j. Analyze and propose an interpretation of these matrices.

Question 7. To improve your result, you can also try averaging the weight vector (save the weight parameters after each training epoch and use their average at the test phase). Why? How can you interpret the result?