ADPI, 2º MIT

LAB 4: MULTICLASS CLASSIFICATION OF HANDWRITTEN NUMBERS WITH SOFTMAX
COURSE 17/18

Note: this lab is quite based on the material in `http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/`

1. Introduction

The goal of this lab is to program a multiclass classifier. We propose the softmax as a widely extended solution for this type of problems. We apply it to the classification of handwritten numbers from 0 to 9. Therefore we have $Q = 10$ classes.
The MNIST data base is a well-known benchmark in machine learning. You may look in the web for some images of this database. It has a training and a test data set. The input to the classifier is a vector with the image of the scanned handwritten number in gray levels. Since the images in the MNIST are 28×28 we have vectors of length 748. This vector is augmented with a 1 to include an intercept term.
Since we are using the softmax we have to learn $Q - 1$ weights of $748 + 1$ entries.
We could perform any preprocessing to the images, including non-linear transformation, but in this exercise we just focus on the softmax applied to the raw inputs, with no preprocessing.

2. Exercise

2.1. Help code

A function is provided, ejercicioSoftmax.m, where you will see some "TO DO" lines.
The code loads the MNIST data base, adding an intercept term, then calls minFunc with the softmax_regression_vec.m file as the objective function.
In particular you should build this softmax_regression_vec.m function to provide the cost function and its gradient. Also, you should provide the output given some weights and inputs.
The code works with the weights arranged in a matrix with $Q - 1$ columns, the Q-th one is considered to be zero. However, the softmax_regression_vec.m has as input this matrix vectorized.
The code calls to minFunc to perform the gradient descent. Matlab has its own function, but it is quite slow. We recommend to use the functions in the folder "common", provided by the Standford University.

2.2. Goal

Therefore, your task is to implement the softmax_regression_vec.m file to compute the softmax objective function $J(\mathbf{w}; \mathbf{X}, \mathbf{y})$ and store it in the variable f. You must also compute the gradient $dJ/d\mathbf{w}$ and store it in the variable g. Don't forget that minFunc supplies the parameters $\mathbf{w}$ as a vector. You also need to remember to reshape the returned gradient g back into a vector using g=g(:).
You can start out with a for-loop version of the code if necessary to get the gradient right. However, you might find that this implementation is too slow to run the optimizer all the way through. After you get the gradient right with a slow version of the code, try to vectorize your code as well as possible before running the full experiment.

2.3. Some tips

Here are a few MATLAB tips that you might find useful for implementing or speeding up your code (though these may or may not be useful depending on your implementation strategy):
Suppose we have a matrix A and we want to extract a single element from each row, where the column of the element to be extracted from each row i is stored in y(i), where y is a row vector. We can use the sub2ind() function like this:

I=sub2ind(size(A), 1:size(A,1), y); values = A(I); This code will take each pair of indices (i,j) where i comes from the second argument and j comes from the corresponding element of the third argument, and compute the corresponding 1D index into A for the (i,j)'th element. So, I(1) will be the index for the element at location (1,y(1)), and I(2) will be the index for the element at (2,y(2)).
When you compute the predicted label probabilities try to use matrix multiplications and bsxfun to speed up the computation. For example, once **W** is in matrix form, you can compute the products for every example and the first 9 classes using a=X*W. (Recall that the 10th class is left out of **w**, so that a(10,:) is just assumed to be 0.)

3. Results

Provide the required function and how you compute the outputs once you have trained the softmax.
Explain if you improved the running time vectorizing the solution and how you did it.
Provide the accuracy for the training and the test data sets.