Assignment 5. Neural Network Classifier with Two Hidden Layers

Due Date: December 9, 11:30 pm

If you submit the assignment after the deadline, but before December 16, 11:30 pm, a 10% late penalty will be applied (if the mark before applying the penalty is 78 out of 100, after applying the penalty it is 78 - 7.8 = 70.2 out of 100).
Note that **you cannot MSAF** this assignment since the system does not allow MSAFs to be filed after the end of classes, to the best of my knowledge.

Description:

In this assignment you are required to build a neural network classifiers with two hidden layers for binary classification. You will use the banknote authentication data set (provided in the text file 'data_banknote_authentication.txt'). The data set consists of 1372 examples, each example representing a banknote. There are four predictor variables (i.e., features). The goal is to predict if a banknote is authentic (class 0) or a forgery (class 1).

Before starting building your classifiers you have to split the data into the test set, the validation set and the training set. You decide what splitting ratio to choose and include it in the report. Use as the random state when splitting any four-digit number that contains the last four digits of your student ID, in any order. After you separate the test, validation and training data, you have to standardize the features, (i.e. center and scale them) in the training set and then apply the transformations to the features in the validation and test sets.

Use ReLU as the activation function at the hidden units. Train your networks using the average cross-entropy loss on the training set. Stochastic gradient descent (SGD) can be used in the optimization process. You can choose the learning rate 0.005 or another value.

You have to choose the number of units in the hidden layers (denoted by n_1 for hidden layer 1 and n_2 for hidden layer 2) using the validation set. Consider different pairs (n_1, n_2) and for each choice of (n_1, n_2), train the network and use early stopping to determine the weights. In other words, fix a number of epochs to run the SGD algorithm (for instance, 100 epochs - one epoch is one pass through all the training examples). After each epoch, compute the training cross-entropy loss and the validation cross-entropy loss. At the end, choose the weights that achieve the lowest validation cross-entropy loss.

Your goal is to find the best pair (n_1, n_2). You decide what pairs to try. The idea is that you have to gradually increase the capacity of the model until you either observe overfitting or some other behaviour that indicates that you may or have to stop.

Recall that SGD starts with a random assignment of values to the weights of the network. It then proceeds in iterations, which are organized in epochs. Each epoch represents one pass through all the training data. At the beginning of each epoch, shuffle the training examples, then iterate through all of them. At each iteration, you have to compute the current gradient using the forward pass followed by the backward pass. Next, use the gradient to update the weights. Keep in mind that the result of the SGD when training a neural network depends on the initial values of the weights. Therefore, for each network configuration that you have to assess (i.e., for each choice (n_1, n_2)), run the algorithm at least five times starting with different weight assignments and choose the weights that achieve the smallest validation cross-entropy loss.

You have to write a report to present your results and their discussion. In your report you have to describe your strategy for selecting the pairs (n_1, n_2) that you used in the validation process and to explain why you think that they were sufficient. Specify the number of hidden units that you finally chose for your model and the weights of the model. Specify the cross-entropy loss obtained with each of the model used in the validation process and the missclassification error. For the final model specify the misclassification error on the test set and plot the learning curve. Include other observations that you find valuable. Specify how you would try to improve the performance of the neural network classifier if you had more time to work on the assignment.

You may earn bonus points if you make further improvements and you document them in the report. In particular, you may earn up to 3% of the course grade if you implement training with weight decay and a procedure to choose the related hyperparameter.

Besides the report, you have to submit your numpy code. The code has to be modular. Write a function for each of the main tasks (e.g., to train the network for a given configuration). Also, write a function for each task that is executed multiple times (e.g, to compute the average error, to compute the gradient of the cost function, etc). The code should include instructive comments. **Use vectorization when computing the average error!**

Submission Instructions:

- Submit the report in pdf format, the python file (with extension ".py") containing your code, and a short demo video. The video should be 1 min or less. In the video, you should scroll down your code, show that it runs and that it outputs the results for each part of the assignment. Submit the files in the Assignments Box on Avenue.