Assignment 5. Neural Network Classifier with Two Hidden Layers

Due Date: November 17, 11:30 pm

Late submission

If you submit the assignment after the deadline, the following penalty is applied:

◇ 10% penalty if the submission is before December 1, 11:30 pm (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);

◇ 50% penalty if the submission is after December 1, 11:30 pm and before Dec. 6, 11:30 pm.

Description:

In this assignment you are required to build a neural network classifier 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.

It is your decision what activation function to use at the hidden units. At the output unit, use logistic sigmoid as the activation function. Train your network with gradient descent (GD) and early stopping using the average cross-entropy loss on the training set as the cost function. It is your decision what variant of GD to use (batch GD, mini-batch GD or stochastic GD). To choose the learning rate, you can start with 0.005 and then adjust it as appropriate.

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) with $n_1 + n_2 \leq 8$, and for each choice of (n_1, n_2), train the network and use early stopping to determine the weights. You may use a simpler form of early stopping as follows: fix a number of epochs/iterations to run the GD algorithm; after each epochs/iteration, 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. Note that the number of epochs/iterations has to be sufficiently large in order to obtain a U-shape validation learning curve.

Alternatively, you may implement the early stopping method described in class.

For each network configuration that you have to assess (i.e., for each choice (n_1, n_2)), run the GD algorithm at least three times starting with different initial weights. Then choose the best network.

You have to write a report to present your results and their discussion. In your report you have to describe all the choices that you made (type of GD, activation functions used, how you initialized the weights, what early stopping stategy you used, number of epochs or iterations - if applicable, etc.). Specify the training and the validation cross-entropy loss for each model that you trained (i.e., for each pair (n_1, n_2) and each initialization). Organize this information nicely in one table (or more if necessary, but not too many). Specify the pair (n_1, n_2) that you finally chose for your model and the weights of the trained model.

For the final model, also specify the misclassification error on the training set, validation set and test set and plot the learning curves (showing the cross-entropy loss for the training set and the validation set as the number of epochs/iterations increases). Note that it is expected to obtain a U-shape for the validation learning curve, otherwise, it means that you stopped too early.

Include other observations that you find valuable. Is your classifier good enough? If not, specify how you would try to improve the performance if you had more time to work on the assignment.

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. The main Python file in the project should be clearly distinguishable. Some feedback might be written on your report, so, please DO NOT ZIP YOUR FILES. Submit the files in the Assignments Box on Avenue.

  Naming convention:
  "studentMacId_studentNumber_A4_report", "studentMacId_studentNumber_A4_code".