Assignment 3. Neural Network Classifier with Two Hidden Layers

Due Date: December 7th, 11:30 pm

Description:

In this assignment you are required to build three 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). You are required to build three classifiers: one making the prediction based on the first two features, another one predicting based on the first three features and, finally, a classifier based on all four features.

Before starting building your classifiers you have to split the data into the test set, the validation set and the training set. 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 - see slide 22 in Topic 2) 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. For each of the three classifiers (i.e., with $D = 2$, 3 and 4 input units, respectively) you have to choose the number of units in the hidden layer (denoted n_1 for hidden layer 1 and n_2 for hidden layer 2) using the validation set. Consider all 9 combinations formed with 2, 3 and 4 units on each hidden layer. 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, as well as the training misclassification error and the validation misclassification error. At the end choose the weights that achieve the **smallest validation misclassification error**. At the end plot the learning curve, i.e. the curve of the cost function (the training cross-entropy loss) versus the number of epochs. Also include in the figure the plots of the validation cross-entropy loss, the training misclassification error and the validation misclassification error.

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 three times starting with different weight assignments and choose the weights that achieve the smallest validation misclassification error.

You have to write a report to present your results and their discussion. The report must include three tables, one for each $D \in \{2, 3, 4\}$, containing three validation misclassification errors for each pair (n_1, n_2) (27 values in all). The table must also include the test misclassification error for the best classifier for the corresponding D (note that if two classifiers have the same performance on the validation set, the one with a smaller size - i.e., with smaller number of weights - should be chosen). The report should also specify the configuration and the weights of the best classifier for each D. Include other observations that you find valuable (for instance, the comparison of the three classifiers).

You are also required to include in the report nine figures containing the plots of the training and validation cross-entropy losses and misclassification errors versus the epoch number, obtained while running the SGD for $D = 2$. Include one figure for each pair (n_1, n_2). Discuss the results. What do you notice about the learning curves? Are they nonincreasing? Are the other curves nonincreasing? Why? How are the training and validation errors compared to each other. How are the cross-entropy errors and misclassification errors compared to each other? Are these results expected? Why?

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!**

You may use the following code for feature normalization:

```
import numpy as np
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train[:, :]  = sc.fit_transform(X_train[:, :])
X_valid[:, :]  = sc.transform(X_valid[:, :])
X_test[:, :]  = sc.transform(X_test[:, :])
```

Note that the normalization is performed based on the training data. Then the same transformations are applied to the validation and test data in order to be able to use the trained predictor on the validation and test sets.

Submission Instructions:

- Submit the files in the Assignments Box on Avenue. Specific instructions for the format of the submitted files will follow.