AI 201
Programming Assignment 3
Implementing the Backpropagation Algorithm
Instructor: Pros Naval
Due: April 19, 2024 (12:00 noon)

In this Programming Assignment, you will implement a Multilayer Perceptron Neural Network, train it using the Backpropagation Algorithm on a challenging dataset and characterize its performance on that challenging dataset.

Instructions

1. Using Jupyter notebook write Python code to implement a 2-layer Multilayer Preceptron Neural Network with Backpropagation as its training algorithm. **Use only the numpy library for your forward and backpropagation computations.** Your code should have the following features:
   a) use vectorized code that allows processing of training data in mini-batches with user-specified value of mini-batch size (default value: 8);
   b) allow user to specify the type of activation function (Logistic, Tanh, and Leaky ReLU) for each layer
   c) use the Generalized Delta Rule which includes the momentum term to speed up computation (default: alpha = 0.9);
   d) save to a file the training sum of square errors per epoch. You will use this plot the training error;
   e) on the validation set, measure every 5 epochs the number of misclassifications and the sum of square errors and save these values to a file. You will use the sum of square errors to plot the validation error. The training and validation error plots are called the learning curves;
   f) at the end of training:
      - display the learning curves (training and validation errors vs epoch)
      - save the confusion matrix to a file
      - calculate the Accuracy, Precision, Recall, F1 scores, Matthews Correlation Coefficient and save them to a file
      - save the weights to a file which can be loaded for production mode

   g) upload previously-trained weights and use these to evaluate the performance of your network on a validation set
2. Unzip the file AI201_PA3_Data.zip provided.
3. Use a spreadsheet to view the contents of the following dataset files:

   data.csv : 3486 instances each having 354 attributes or features
   data_labels.csv : class labels (1,2,…8) for each of the 3486 instances
   test_set.csv : test instances without labels

   For this dataset, the MLP should have 354 inputs and 8 output nodes (one for each class). Use the tanh activation function (a = 1.716, b = 2/3) for the hidden neurons in both hidden layers. Use the logistic activation function

(a = 2.0) for the output layer. We call this Network A henceforth. The data_labels values should be one-hot encoded.

4. Note that we have here a highly imbalanced dataset (i.e., there are huge differences in the number of instances for each class). Use the Synthetic Minority Oversampling Technique on *data.csv* to create a balanced dataset. See for reference this article: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/. **You may import the libraries used in this article only for the purpose of generating a balanced dataset but not for the rest of this PA.**
5. **Randomly partition** the balanced dataset you made into a training set named *training_set.csv* and *training_labels.csv* and a validation set named *validation_set.csv* and *validation_labels.csv*. Make sure that each entry in *training_set.csv*/*validation_labels.csv* corresponds to the right set of input features in *training_labels.csv*/*validation_set.csv*. The validation set should contain exactly 800 samples. Since the validation set has been created from random partitioning, the number of samples for each class will not be exactly the same but close to one another in value (i.e., close to 100).
6. Train your ANN on the training set and use the validation set to monitor your errors over training epochs. Monitor also the training time and include this in you report.
7. Change the number of nodes in each hidden layer and the learning rate parameter so that the error over the validation set is minimized. You may also change the momentum constant alpha to speed up computations. Once the network has learned, save the weights to a file named *trained_weights.csv*
8. Repeat 7 for the same number of nodes in each hidden found in 7 but this time using the Leaky ReLU (slope parameter = 1 or 0.01 for input > 0 or input < 0 respectively) as the activation function of all neurons in both hidden layers. We call this Network B henceforth. Network A is a tanh-tanh-logistic network while Network B is a LeakyReLU-LeakyReLU-logistic network.
9. Load *trained_weights.csv* use it to predict the labels for each of the test set instances in *data.csv*. **Do not shuffle the order of presentation of the test instances.** Convert the one-hot encoded network outputs into class labels (1,2...,8) and save these labels in a csv file named *predictions_for_test_tanh.csv* and *predictions_for_test_leakyrelu.csv* with the same format as *data_labels.csv*.
10. Document the entire process. Analyze the learning curves, and performance metrics, and describe how the hyperparameters were tuned. Compare the performance of the network when Tanh and Leaky ReLU activations are used for the hidden neurons for Networks A and B respectively. Compare the training times for the networks with different hidden layer activations.

Deliverables:

a) Jupyter notebook of ANN code in Python for Networks A and B with self-documenting comments. Include another one for generating the balanced data using SMOTE.
b) The following csv files:

i) training_set.csv and *training_labels.csv (balanced using SMOTE)*
ii) *validation_set.csv* and *validation_labels.csv (balanced using SMOTE)*
iii) *predictions_for_test_tanh.csv and predictions_for_test_leakyrelu.csv*

c) Documentation (in the same style as the one for PA1) describing the following:
   i) how the dataset was partitioned and details on how you dealt with the highly imbalanced dataset
   ii) how the number of nodes in the hidden layers was selected, how you tuned the hyperparameters (learning rate parameter, momentum constant alpha) for Network A
   iii) Separate plots of training and validation errors vs epoch number for Networks A and B.
   iv) Timing Comparisons for Networks A and B
   v) Confusion matrix, Accuracy, Precision, Recall, F1 scores, Matthews Correlation Coefficient values and their interpretation (one each for Networks A and B). You may include other metrics from our Classifier Evaluation Lecture if you wish.
   vi) Conclusion

Note: **Make sure that your MLP code uses only the numpy library for the forward and backpropagation computations, otherwise the rest of your submission will not be checked and you will get zero credit.** You may import the matplotlib and csv libraries.

Submit the deliverables to submit2pcnaval@gmail.com with "[CS280 PA2 (Neural Network) Submission] <Your_Name>" on the subject line by 12:00 noon of April 19, 2024.