Eric Fortney  
Backpropagation Report  
2/19/19

1) Implement the backpropagation algorithm. This is probably the most intense lab, so start early! This and the remaining models are implementations you should be able to use on real world problems in your careers, group projects, etc. Your implementation should include:

◦ ability to create an network structure with at least one hidden layer and an arbitrary number of nodes

◦ random weight initialization (small random weights with 0 mean)

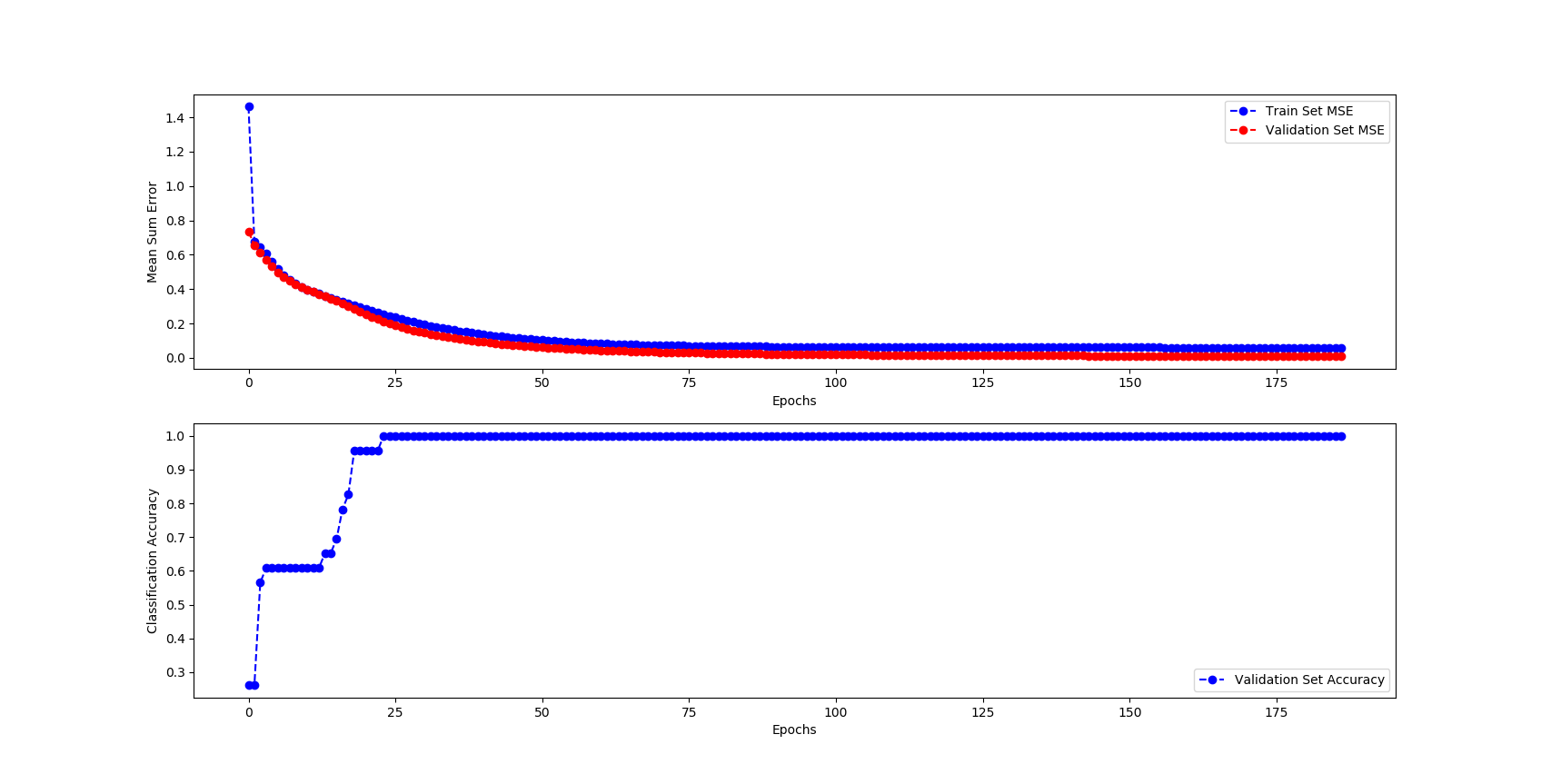
◦ on-line/stochastic weight update

◦ a reasonable stopping criterion

◦ training set randomization at each epoch

◦ an option to include a momentum term

See attached code

**2)  *Use your backpropagation learner, with stochastic weight updates, for the*[*iris*](https://learningsuite.byu.edu/plugins/Upload/fileDownload.php?fileId=ff209867-StCI-l8Xd-2JHS-af83968a9e62)*classification problem. Use one layer of hidden nodes with the number of hidden nodes being twice the number of inputs. Always use bias weights to each hidden and output node.  Use a random 75/25 split of the data for the training/test set and a learning rate of .1. Use a validation set (VS) for your stopping criteria for this and the remaining experiments. Note that with a VS you do not stop the first epoch that the VS does not get an improved accuracy.  Rather, you keep track of the best solution so far (bssf) on the VS and consider a window of epochs (e.g. 5) and when there has been no improvement over bssf in terms of VS MSE for the length of the window, then you stop. Create one graph with the MSE (mean squared error) on the training set, the MSE on the VS, and the classification accuracy (% classified correctly) of the VS on the y-axis, and number of epochs on the x-axis. (Note two scales on the y-axis). The results for the different measurables should be shown with a different color, line type, etc. Typical backpropagation accuracies for the Iris data set are 85-95%.  (Showing this all in one graph is best, but if you need to use two graphs, that is OK).*

3)  For 3-5 you will use the [vowel](https://learningsuite.byu.edu/plugins/Upload/fileDownload.php?fileId=9b75ad3b-sVWS-GNi3-lBdw-Yw566fa316e2) dataset, which is a more difficult task (what would the baseline accuracy be?).

The baseline accuracy will be the number of vowels sounds producible. The provided dataset divides them into 10 categories (hid, hId, hEd, hAd, hYd, had, hOD, hod, hUd, hud) therefore the baseline accuracy will be 1/10, 10%.

Typical backpropagation accuracies for the Vowel data set are ~60%. Consider carefully which of the given input features you should actually use (Train/test, speaker, and gender?) and discuss why you chose the ones you did.

For my backpropagation classifier I choose to use all explanatory variables except for speaker and ‘Train or Test’. Factoring in who is speaking is irrelevant to broadly classifying vowel usage for novel people. (This is similar to the discussed social security number problem as we talked about in class wherein the unique identifiers can serve as a direct mapping across the training set that does not generalize broadly) How the previous dataset owners defined their test and training sets is of little interest to us.

Use one layer of hidden nodes with the number of hidden nodes being twice the number of inputs. Use random 75/25 splits of the data for the training/test set. Try some different learning rates (LR).  For each LR find the best VS solution (in terms of VS MSE). Note that each LR will probably require a different number of epochs to learn.  Also note that the proper approach in this case would be to average the results of multiple random initial conditions (splits and initial weight settings) for each learning rate. To minimize work you may just do each learning rate once with the same initial conditions. If you would like you may average the results of multiple initial conditions (e.g. 3) per LR, and that obviously would give more accurate results. The same applies for parts 4 and 5.  Create one graph with MSE for the training set, VS, and test set, at your chosen VS stopping spot for each tested learning rate on the x-axis.  Create another graph showing the number of epochs needed to get to the best VS solution on the y-axis for each tested learning rate on the x-axis.  In general, whenever you are testing a parameter such as LR, # of hidden nodes, etc., test values until no more improvement is found. For example, if 20 hidden nodes did better than 10, you would not stop at 20, but would try 40, etc., until you saw that you no longer got improvement.

4) Using the best LR you discovered, experiment with different numbers of hidden nodes. Start with 1 hidden nodes, then 2, and then double them for each test until you get no more improvement in accuracy.  For each number of hidden nodes find the best VS solution (in terms of VS MSE).  Create one graph with MSE for the training set, VS, and test set, on the y-axis and # of hidden nodes on the x-axis.

|  |  |
| --- | --- |
| Number of Hidden Nodes | Best VS Solution (in terms of VS MSE) |
| 1 |  |
| 2 |  |
| 4 |  |
| 8 |  |
| 16 |  |
| 32 |  |
| 64 |  |

5) Try some different momentum terms in the learning equation using the best number of hidden nodes and LR from your earlier experiments. Graph as in step 4 but with momentum on the x-axis and number of epochs until VS convergence on the y-axis. You are trying to see how much momentum speeds up learning.

Hey

6) Do an experiment of your own regarding backpropagation learning.  Do something more interesting than just trying BP on a different task, or just a variation of the other requirements above. Include a description of your experiment, results (often in the form of a chart or table, as appropriate), and discussion.

Hey