Tuan Tran – HW2 – CS595

1. File description:

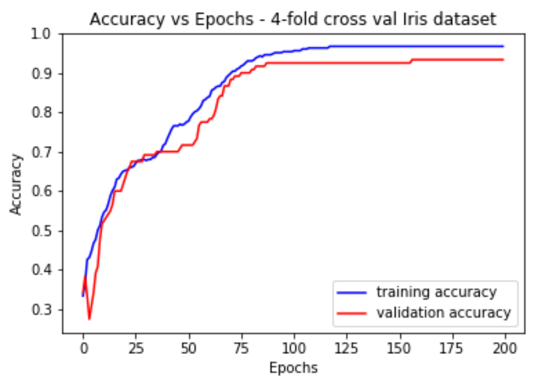
- impl/implementation.py includes the code for helper functions, as well as classes for different node types, computation graph and the dense neural network model
- The notebook training_notebook contains code to retrieve training, validating and testing data for 2 different 3-class classification datasets and code to train and evaluate model.

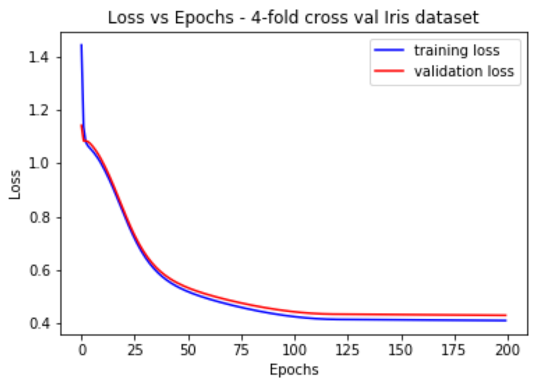
2. Training and evaluating:

- The procedure for training and evaluating the model is similar to the procedure for last homework, that is, training, then tune hyper-parameters and finally train the best model and evaluate on test set. **Overall**, the network written behaves similarly to the network from keras. Some examples of this would be:
  + Sum of 3 softmax outputs = 1
  + The loss value matched what was calculated by hand
  + If the learning rate is too high, or unsuitable, then similar to the network from keras, the performance of this network will be very unstable and it tends to overshoot, not able to decrease the loss
  + If the learning rate is too small, then the network will need more epochs to get to a reasonably low loss value
  + The plots of accuracy vs epochs and loss vs epochs is also reasonable
- For SGD, I made sure to shuffle the data at the beginning of each epochs

a. Iris dataset:

- This is a very small dataset (around 150 instances), thus I used K-fold cross validation to train and evaluate the model. Also, the feature vector is only a 4-dimensional vector, thus I chose a small number of units per hidden layer (around 10).
- Training the best model (after tuning), with 10 units per hidden layer, for 200 epochs, batch size = 16, and learning_rate=0.01 using k-fold (k=4) cross validation will yield the following result:



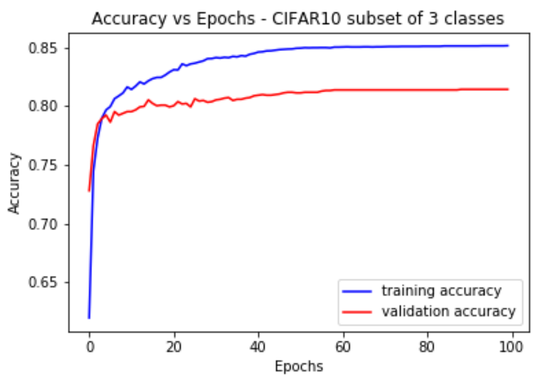


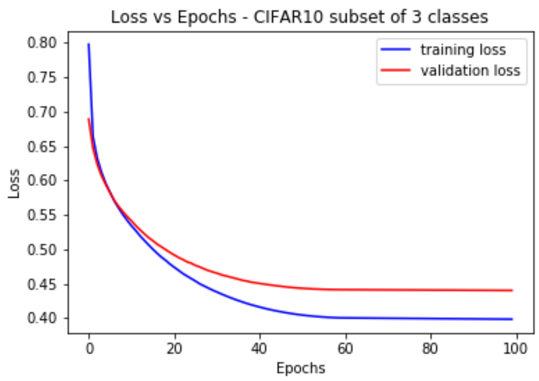
+ I was able to get these graphs because I average the training loss/accuracy and validating loss/accuracy across the 4 folds, and got 4 lists: averaged (across folds) training loss, accuracy, and averaged validating loss/accuracy. Overall, the graph looks very reasonable as the loss is decreasing and converging to some small values, and the accuracy is increasing and also converging at some value close to 1.0 as epochs increases. This means that our model is "learning"

- Finally, evaluating on the final test set yields an accuracy of 0.9666

b. CIFAR10 with a subset of 3 classes:

- I use the same dataset and the same training/validating/testing set retrieval procedure as last homework. After tuning the hyperparameters, the best model is: units per hidden layer = 512, epochs = 100, learning rate = 0.00001, and batch_size=512
- Training this model on training set yields the following result:





+ Again, the graphs look reasonable and the model is "learning". One problem would be overfitting. Since this homework does not require implementation of regularization method, we will ignore this problem

- Evaluating on the final test set yields an accuracy of 0.8355