Summary and Problem Statement

In this exercise, we are conducting an experiment to evaluate neural network performance on a binary classification task against the performance of logistic regression, SMV and random forest classification models. Given a set of movie reviews, the goal is to use the text comments to build a model to predict if a movie will get a thumbs-up or down.

Methodology

The basic TensorFlow (TF) model built in assignment 7was appended to the instructor's sample code, and the TF code updated to work with new text input. I ran several iterations of the code using different numbers of layers and nodes; the other hyperparameters used the default API settings. The best results were used in the creation of a second TF model which allowed for changing a number of the hyperparameters. I tried several things, including: different optimizers, activations, learning rates, and steps. The full results are shown in the spreadsheet[1]. For each run, a single parameter was changed, the code was run, and results recorded.

Code Overview

The bulk of the code came from the professor's sample, and from the course text[2]. A lot happens in the instructor's code, but a key data preparation/manipulation piece must be noted: a sparse matrix is created which has the positive and negative words[3] as columns, with 1's and 0's indicating if each review contains that word. This matrix is used as the data for training and testing the models. Thumbs up and down categorical data are also converted to binary values by the provided code.

The first TensorFlow model is from the textbook[2] via assignment 7, then modified to work with the new data. The second model was created by looking at TF API documentation and Google's samples, the expanding the classifier definition to allow for tweaking the different hyperparameters.

Results and Observations

I tried a number of different hyperparameters; different activations, different learning rates, different optimizers, and different tuning rates within the optimizers. A subset of the full results[1], highlighting the key findings are shown below:

Model	**Hyperparameters**	**Training Accuracy**	**Test Accuracy**
Logistic Regression		76.8	73.6
SVM	defaults	76.5	74.5
Random Forest	Number of trees = 10	95.6	70.7
Basic DNN	2 layers - layer1 - 200 nodes, layer2 - 100 nodes	96.7	**72.4**
Tuned DNN	2 layers with 200 and 100 nodes. Adagrad optimizer, Relu activation, learning rate =.001, initial_accumulator = .001, 10K steps	91.4	**75.5**

My basic neural network with 2 layers did beat the one with 5. Various numbers of nodes were tried, l1=200 l2=100 was best. My overall best model achieved 75.5% accuracy on the test set, which beat the best non-neural network model by 1 point. I did find the discrepancies between the training accuracy and test accuracy interesting; there is a big drop in the results on the test set for the random forest and most but not all of the neural net models; larger drops than I would have expected. Clearly the parameters make a difference, since not all parameter sets showed the discrepancy.

[1] Model_Results.pdf
[2] *Hands-on Machine Learning with Scikit-Learn and TensorFlow* by Aurélien Geron, ch. 10
[3] Hu, M. and Liu, B. (2004). *Mining and summarizing customer reviews.* In SIGKDD-04