CS 440 - Homework 7

Matthew Liu - mliu56

1. Conceptual Questions

1. What is the example space, $\boldsymbol{X}$, for this problem?
   Since the input can be 0, 0.5, 1, $\boldsymbol{X}$ is 3-dimensional real space R^3

2. Which model do you expect to be more expressive: a binary classifier perceptron or a binary classifier multi-layer perceptron with one hidden layer? Why?
   A binary classifier multi-layer perceptron with one hidden layer will be more expressive, it has more VC dimensions then the former and thus will be the richer model and be able to represent more things.

3. Which one of the models described in question 2 do you think is more likely to overfit, and why?
   Even though the multi-layer perceptron with one hidden layer will be more expressive, it is also more likely to overfit when you do not have enough data.

4. Do you expect your perceptron to converge (assuming a constant learning rate)? Why or why not?
   Yes, because it seems there is plenty of data available to properly train the perceptron, and the data can be properly separated into positive and negative examples by a hyperplane

2. Implementing a Perceptron

The environment is Visual Studio 2017 with the latest .Net Framework.

1. Learning rate decays. Training example is randomized with Fisher-Yates shuffle. Weight randomized with uniform distribution. I tried multiple combinations of parameters, including constant/decaying rates, randomized/fixed order of training data, and zeros/uniform/gaussian distributions. All of them result in about the same training accuracy (give or take 2%), and about the same test accuracy (75%)

2. It was set at 200 epochs, it only takes about 150 epochs for the validation set to break early.

3. 84-86%

4. 77%

3. Using a Multilayer Perceptron

Because of my environment for Part 2, I could not find any Multilayer Perceptron library. The two that I tried are FANN (Fast Artificial Neural Network Library, open-source library) and ALGLIB. Both required a much older .Net Framework that I could not get to setup. I tried for 2 hours, but it always resulted in the same error (Needing a older version of .Net, corresponding to Visual Studio 2008). After asking a TA, I switched over to Python for this reason. The environment for Part 3 is Python3.6 with sklearn.

1. Most of my combinations are listed in hyperparameters.txt. I tried all three activation functions, and Rectified Linear Unit was the best by far, converging the fastest. I started with smaller numbers of hidden layers and neurons, and bumped those up. At 100 hidden layers and 100 neurons per layer, I got my best result. It is a little overkill, however as noted in hyperparameters.txt, I've also attempted 25, 250 (way more reasonable), but that only got to 89%. 100, 100 consistently hits 90% accuracy

2. I set the limit at 200 epochs so that it would have the same settings as Part 2. It converged at 56 epochs

3. 94.7%

4. 90.3%

5. The Multilayer Perceptron was significantly more accurate, on average 15% more accurate (only because I could not get the minimum 80% no matter what parameters i tried). The accuracy is due to the more complex nature of the Multilayer Perceptron, showing that linear separators are not enough to properly separate the data, but more complex hyperplanes are needed instead.