Brandon TerLouw & Warren Cho

Milestone B Progress Report

1. A revolutionary classifier

2. Brandon TerLouw – Development of neural network

Warren Cho – Development of decision tree

3. The classifier is supposed to use data to predict the class of a specific instance. Our classifiers were used on the Iris data set and a food data set that we made. The food data set consisted of 3 different classes of food (bread, hotdog and peanutbutter). The characteristics of the food was it calorie count, fat content, protein content and carbs. Our programs will use most of the data to build a model, which is then tested on data that was not seen during training.

4. Two layer feed forward neural network – A neural network is one way to implement a classifier. Ours is made of two layers of perceptrons. One layer is the hidden layer while the other is the output layer. A perceptron is a device that takes an array of weights, and an array of input values, then matrix multiplies them. It then uses the product of matrix multiplication in a sigmoid function and outputs this. The sigmoid function we use outputs a value between 0 and 1. A neural network can be trained by changing the weights until the error is close to a minimum. Weights are changed using back propagation, backpropagation is basically coordinate decent. Once a neural network has been trained, ideally it will be able to take data and compute it’s to a class.

Random Forest – A random forest of decision trees is another way to implement a classifier. Using trees that handle all of the various features of objects, a random sample subset of features of an object can be taken and compared against these trees. This results in a prediction of an outcome and the class with the most number of votes/decisions is the final prediction of the random forest. Random forests are an ensemble learning method and is fairly accurate via bagging (bootstrap aggregating) by using randomness (entropy) to its advantage.

5. Neural Network:

We trained a neural network using the Iris data set. The parameters we used were 10 test instances, 8 hidden nodes, stop training once reaching 10 error per epoch and a .005 training rate. It takes about 30 seconds to train and gets around 93% accuracy on training data and 90% on never seen data.

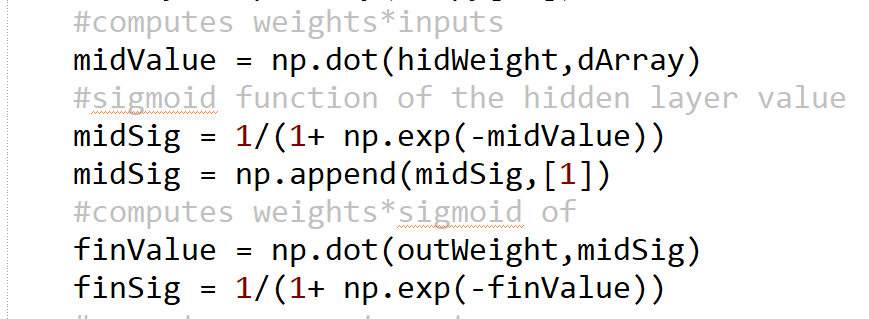
\*insert screenshot

Random Forest: No test results yet available.

6. Neural Network – To run demo you must run ‘presentation.py’ with all files from zip in folder. Follow the prompts from number 5.

Random Forest – No functional demo yet available.

7. Neural Network – Below is a screen shot of the code used for a perceptron. It takes a numpy matrix of weights, multiplies by input, computes sigmoid of hidden layer. Then does the same for the output layer.



Random Forest – below is a snippit of code used for the class prediction using decision trees within the random forest. This returns the prediction with the highest occurrence (mode).

8. Brandon TerLouw - I have learned how a basic 2-layer neural network works. I learned what it takes to program and train a simple perceptron in order for it to classify linear separable data. I have also learned to implement backpropagation and did so for a 2-layer neural network. To use large amounts of data I taught myself to import data from a csv file and use to train a neural network model.

Warren Cho – I have learned that a random forest learning method is that high variance is not always a bad thing – in this case it helps immensely. The randomness of the random forest allows for various evaluations which can be controlled by calibrating the split point to our liking. I also learned that it’s fairly easy to adjust the model for regression vs classification – although we are just using it for classification.

9. Neural Network – With more time I would like to add functionality for picture and other data that has many dimensions. I tried my neural network with the CIFAR-10 data set but training was too slow. This is due to the large matrices involved with many dimensions. To fix this I must preprocess the data in a way to reduce the dimensionality of the pictures.

Random Forest – Ideally we would add better support/implementation of bagging for increased accuracy.

10. Neural Network – For the most part I used information on slides to program and train my neural network. I did have to consult sources to better understand how to implement back propagation.

This is the article I read to better understand math behind back propagation:

https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Random Forest – This website provided very helpful high-level explanations and pseudocode for the random forest algorithm. *How the Random Forest Algorithm Works in Machine Learning* by Saimadhu Polamuri (http://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/).

This website was also very helpful for the same reasons as the previous link but provided a more detailed case example as well as some code examples. *Random Forest Python* by Mahsa Hassakashi (https://www.codeproject.com/Articles/1197167/Random-Forest-Python).