Machine Learning: Project 1

Cody Worsnop

December 8, 2018

Overview

As a documentation overview, I'll provide key functions and their general implementation. I'll do my best to discuss design and implementation strategies for the overall problem as well.

Note: My network is a rather slow learner. Be patient, it'll come around, it's just a bit shy.

Calculate Loss

The loss for this project is given by the formula

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{n \in N} \sum_{i \in C} y_{n,c} \, Log \, \hat{y}_{n,i} \tag{1}$$

So for every prediction, we are summing each class and each value in each class. For example, if we have a softmax prediction of ¡0.4, 0.6¿ then the loss function would be (0 * 0.4) + (1 * 0.6). As a side affect, we are only concerned with the correct prediction. This is a property of cross-entropy loss.

My implementation runs the prediction function and then runs the above formula on all the predictions. Once done, the final error is returned.

Predict

For this project, I have numerous prediction functions. This is due to the orignal write up for the project only specifying that one feature be passed in at a time in the function. Thus, I have the following functions:

predict(model, x)

This is the normal prediction function as designed in the project report. It runs foward propagation on a single feature and returns the prediction of the network.

predict_many(model, X, outputSize=2)

This is the predict many function that takes the entire feature vector and returns the prediction for all the features in one large vector.

Build Model

Build model is where all the business happens. For every epoch (num_passes) we predict all of the features, output the loss if necessary, and run gradient descent on the network. All the weights and biases, including the different activations (a, h, and z) are saved in a dictionary that get's populated at the start of build model.

Build Model 691

This function extends the functionality to three outputs. This was a rather straightforward process once build_model was done. Mostly, changes were making the matrix dimensions for the weights and biases to 3 instead of 2. Also, I reused my predict_many function from above by using the outputSize parameter.

plots