Luke Beebe
Assignment 8, Neural Nets

The purpose of this assignment was to understand the basics of how neural nets work by commenting the code that Professor LuValle provided us and running his greedy random neural net optimizer to fit a regression model on it. Because my R script comments what each line of code is doing, I will use this word document to provide a more general and intuitive explanation of his neural net.

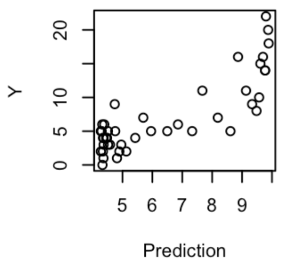
my.neuralnet takes a few parameters, most importantly X1, our inputs, and Y, what we're training the model to predict. It then takes the length of the X1 and creates random weights for each input times the hidden parameter, which represents the number of neurons in our neural net. It then passes this to two other functions which do a lot of the background math to ready it for gradient descent. The background math makes the neurons behave like neurons so that they fire at a certain point and multiplies the inputs by the weights. After this it returns to the main function where it uses gradient descent to optimize the weights and passes the best set of parameters to the variable wfinal, wich represents our final weights. It uses these parameters to predict Y values and plots the predictions vs the actual values so we can see how accurate the function is.

my.nuralnet.multilayer takes two more parameters, num.layers, which creates the number of hidden layers for the neural net, and lambda, which is the regularization coefficient that shrinks the betas for a (hopefully) more accurate model. If num.layers is greater than 1, there is a loop within the function that fills a matrix of dimensions hidden by hidden with weights and continues to do this for as many layers as specified. If lambda is specified as a number other than 0, then it adds lambda*sum(w0^2) to llik, which is extracted and passes to myoptfunc, which then uses these higher values to perform gradient descent with to find our final weights for our final plot.

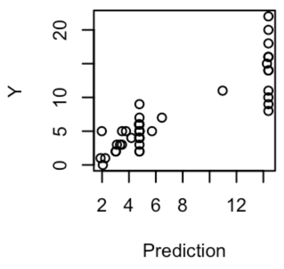
I then used **my.greedy.neuralnet** which sets all of the parameters and runs my.neuralnet.multilayer the specified number of times. Because the weights are random before gradient descent, it naturally creates different outputs each trial. It saves the output of the most accurate model based on ss (sum of squares), and after running each trial, plots the most accurate model. I ran it a few times until I understood it, and got the output posted on the second page. The most accurate model I got had a SS of approximately 173.8. You can see each model it spit out varied in accuracy.

See attached R file for commented code.

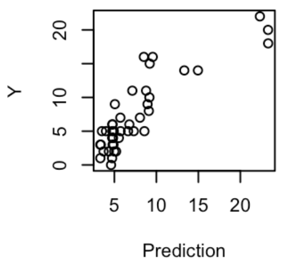
llik= 627.076973514028
SS= 603.602359656231



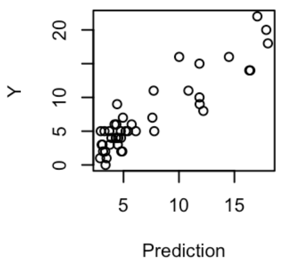
llik= 284.057276924313
SS= 283.930615790818



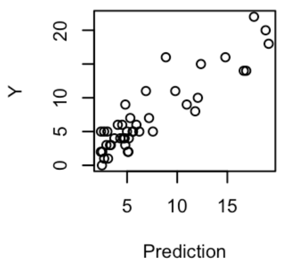
llik= 321.965035125933
SS= 321.433070996907



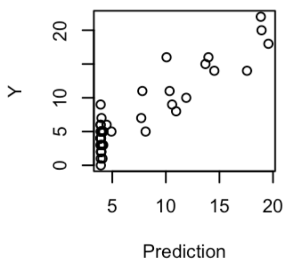
llik= 222.972041082879
SS= 222.884215024473



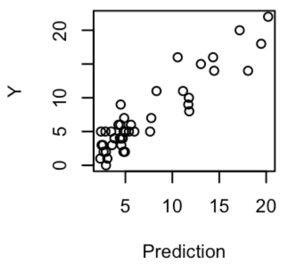
llik= 228.643851744381
SS= 228.641592548249



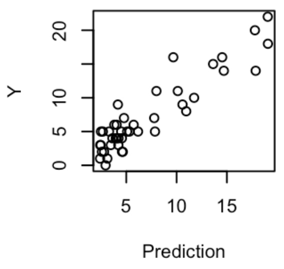
llik= 209.158266497907
SS= 209.118921243572



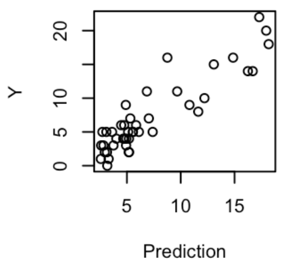
llik= 188.857213719664
SS= 188.854393403964



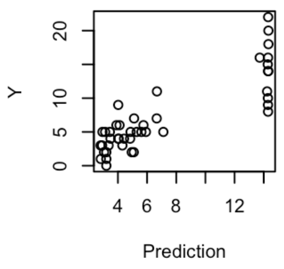
llik= 193.795057659178
SS= 193.779225658004



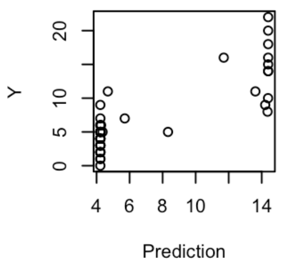
llik= 227.258763566376
SS= 226.870237493232



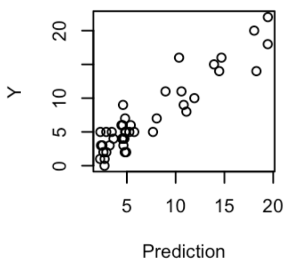
llik= 328.703390344471
SS= 328.673321617922



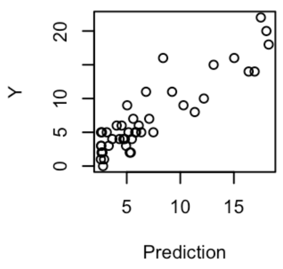
llik= 588.971532865627
SS= 588.823992930169



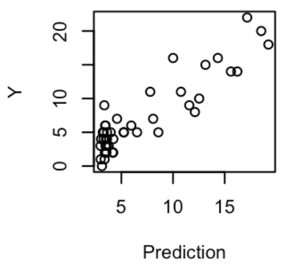
llik= 174.172361884135
SS= 173.783983986848



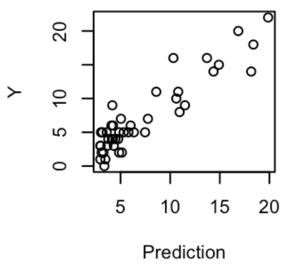
llik= 232.547096387363
SS= 232.497764622716



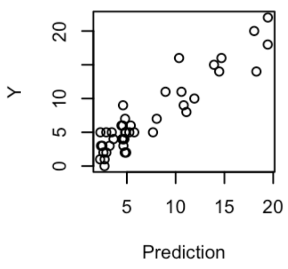
llik= 235.962702365967
SS= 235.834040375917



llik= 189.02352253539
SS= 188.90727942171



llik= 174.172361884135
SS= 173.783983986848


Y
Prediction