Stochastic Gradient Descent (SGD) & Neural Network Intro.

Authors: Joseph (Alex) McGraw, Sitong Li, & Christian Lee. (PDF)

The majority of this class was used for covering what SGD is, how it is generally a better alternative to gradient descent, and how the error calculations are utilized for updating the weights for each variable. We also covered a very quick introduction into Neural Networks, but the next class was more in depth compared to this one. 1. SGD is a much more computationally efficient method for finding the global minima of the error term when compared to gradient descent. 1. Using Gradient Descent, we would randomly assign weights for our model. Then, using these weights, we would calculate the cost over all the data points in our dataset, compute the gradient of the cost with respect to the weights and then perform our update of the weights. This process is repeated until we have reached the global minimum. With potentially millions of data points, we would be performing this cost calculation millions of times just to update the weights at each step. 2. With SGD, we instead compute the cost of a single data point, or the cost associated with mini batches of data points, and the corresponding gradient. The mini batch method would be used if we desired to keep some elements of Gradient Descent but wanted to speed up our optimization. It is in effect a balance between SGD and Gradient Descent. We update the weights after each of these calculations (as seen the figure below). This makes for more efficient optimization than gradient descent, while still allowing us the find the global minima.

$$\begin{aligned}\mathbf{w}^{(\tau+1)} &= \mathbf{w}^{(\tau)} - \eta \nabla E_n \\ &= \mathbf{w}^{(\tau)} + \eta(t_n - \mathbf{w}^{(\tau)\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n))\boldsymbol{\phi}(\mathbf{x}_n).\end{aligned}$$

Figure 1: image.png

3. SGD is also the preferred method when dealing with a large data set because it is less prone to local minima. Local minima are points where

the function value is smaller than at nearby points, but possibly greater than at a distant point. Global minima are the points where the function value is smaller than at all other feasible points. As we are optimizing our model to reduce cost, it is our goal to find the global minimum of the given function. SGD is less prone to local minima in large part because of the way it makes use of "noise", which allows the optimization to jump out of local minima and continue the search for the global minimum.

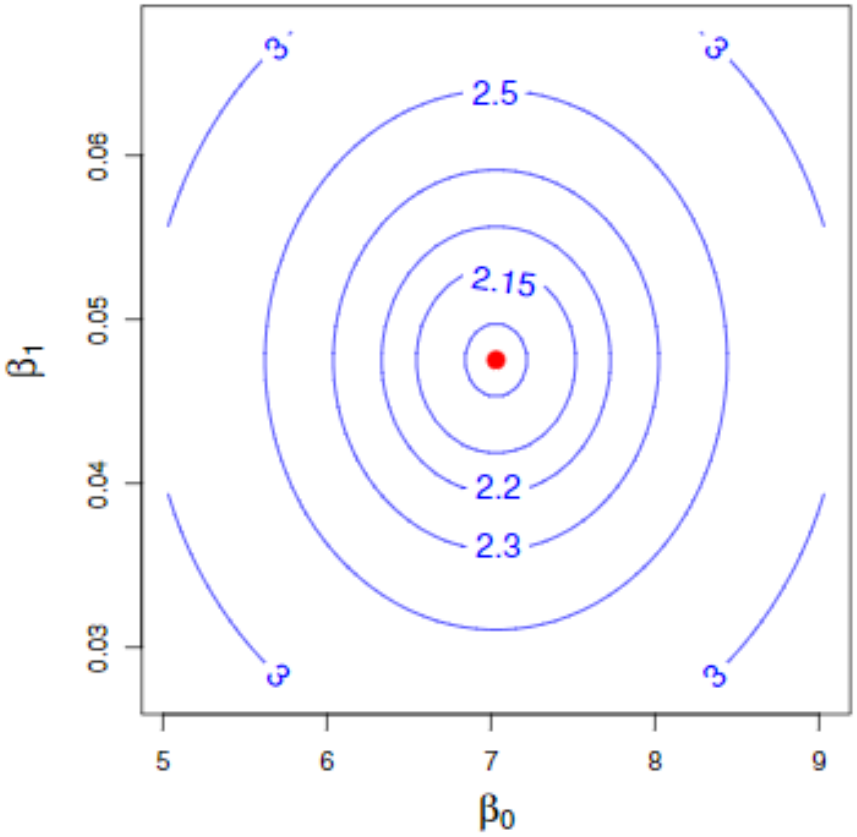

Figure 2: image.png

4. To reach the global minimum, SGD makes use of a learning rate, which is a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradient. Effective optimization with the goal of convergence requires that we reduce the learning rate with time, to prevent the optimization from jumping around the global minimum. However, it should be noted that the goal is not always to converge (ex. Netflix).

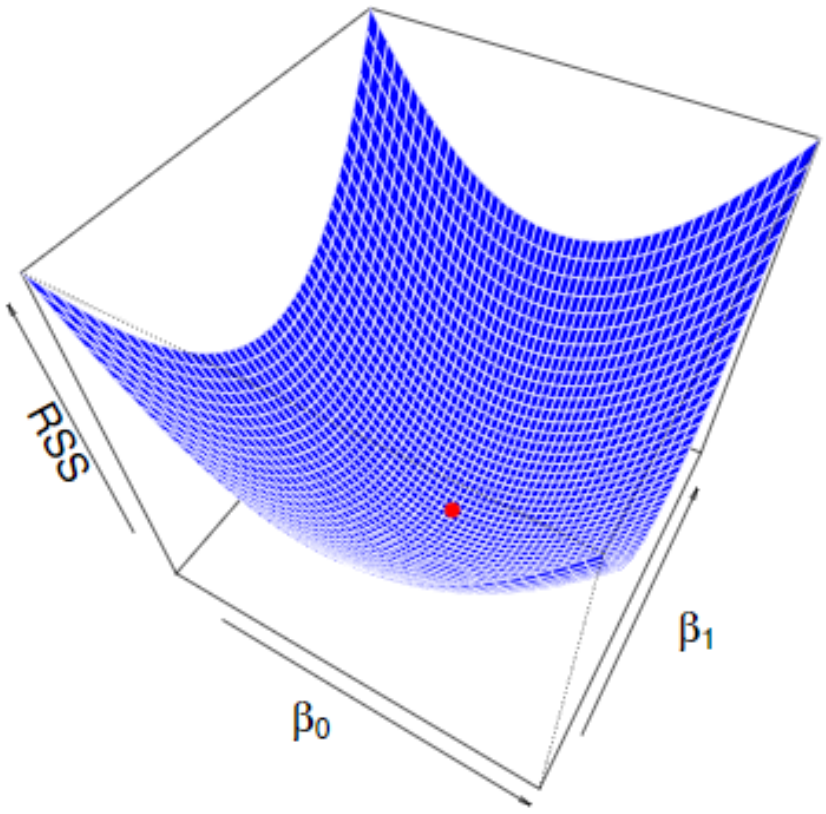

Figure 3: image.png

5. Another of SGD's benefits is its adaptiveness to new information. Once SGD reaches a global minimum, it does not stay put with each new observation; it will move around this minimum but stay in the general vicinity. This model is very efficient with dynamic environments where there is always new data to be interpreted.

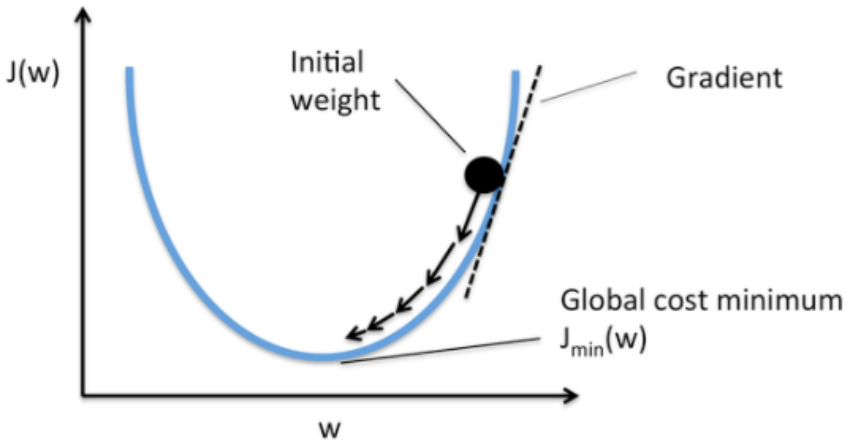

Figure 4: image.png

Second part of the class introduced Neural Network 1. 1-layer (Artificial) Neural Network
1. In a way, neurons can be understood as the subunits of a neural network in a biological brain. 2. Activation function(stimulus) is a function that relats the neural body to the output of the neuron 3. The more the input activity, the higher the output (monotonic) 2. Adaline (Adaptive Linear Element) 1. Developed by Professor Bernard Widrow and his graduate student Ted Hoff 2. An early single-layer artificial neural network with multiple nodes where each node accepts multiple inputs and generates one output 3. In contrast to the perceptron rule, the delta rule updates the weights based on a linear activation function rather than a unit step function 4. For example, if we have activation function: $a_j = \sum W_{j,i} X_i$
5. Then the output: $Y_i = \sigma \left(a_j \right) = \sigma \left(\sum W_{j,i} X_i \right)$

Difference between the perceptron rule and Adaline:

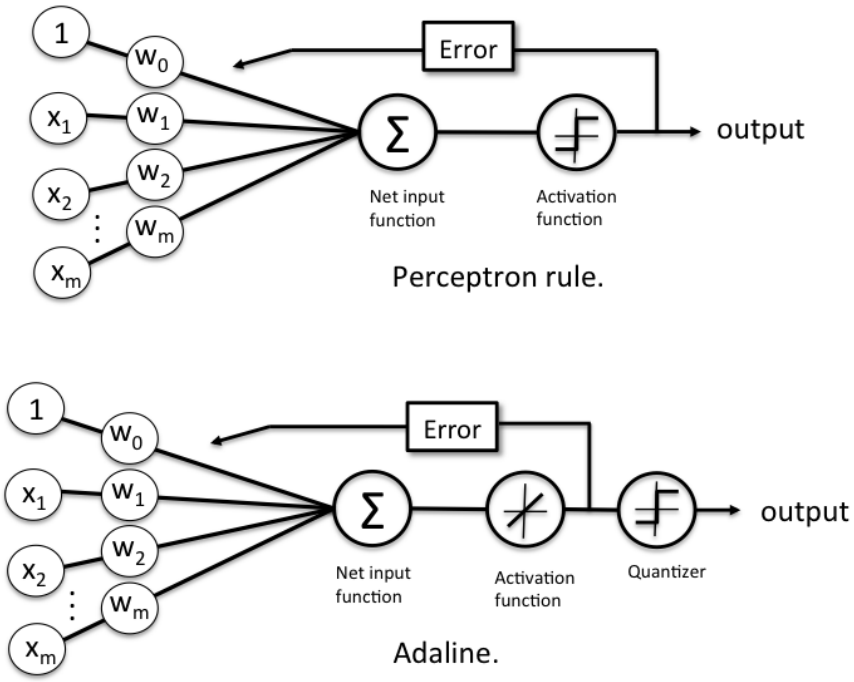

Figure 5: perceptron_vs_adaline.png

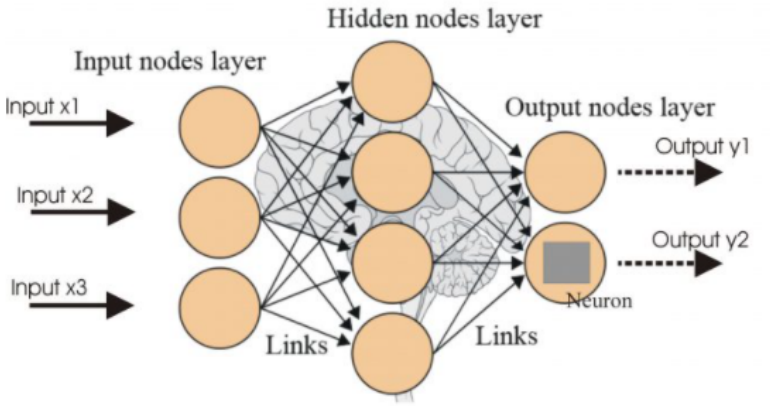

Figure 6: image.png