Practical Machine Learning with Tensorflow

Assignment 1

1. Presence of large errors makes min-max scaling inefficient as the range increases and hence all the correct values are compressed together to almost equal values. Similarly log-transform also compresses the values to almost equal values as all the values lie close to each other (skewed) on the logarithmic scale.
2. Z-score = (x - mean(x)) / std(x)

   mean(x) = 210 std(x) = 128.06

   (x - mean(x)) = [-110., -160., 190., 90., -10.]

   Z-score = (x - mean(x))/std(x) = [-0.86, -1.25, 1.48, 0.7 , -0.08]

   Min-max = (x - min(x)) / (max(x) - min(x))

   min(x) = 50 max(x) = 400

   (x - min(x)) = [50, 0, 350, 250, 150]

   max(x) - min(x) = 350

   Min-max = (x - min(x)) / (max(x) - min(x)) = [0.14, 0. , 1. , 0.71, 0.43]

3. The data looks like coming from a quadratic equation.
4. Sigmoid σ (0.1*x) is a relatively smooth function. You can check it using calculator.
5. Each node from each layer starting from the first hidden layer is connected to all the nodes of the preceding layer - $h_i * h_{i+1}$ parameters. Each node also has a bias unit and hence we have h_{i+1} parameters. Since our counting started with the first hidden layer, the indices for the summation operation start with $i=0$ to $i=l-1$.
6. Here, df(w) = 2 + 2w, w=0

   w_best => df(w_best) = 0 => w_best = -1

   w_best = w - α(2+2w)

   -1 = 0 - 2α

   α = 0.5
7. The gradient descent algorithm does not perform same because squared-loss case in linear regression it is a quadratic function that has only one global minima. But for logistic regression, there are multiple local minima that exist. You can check it by finding double derivative of loss function. To minimize the loss function and find unique solution, its double derivative should be positive.
8. $\nabla \mathbf{f}(x_0, y_0) = (2x_0 , (y_0/50)) = (20, 0)$
9. $\nabla \mathbf{f}(x_0, y_0) = (2x_0 , (y_0/50)) = (4, 0.1)$.

   $\mathbf{f}(x_1, y_1) = \mathbf{f}(0, 4.95) = 0.245$
10. A model which blindly predicts the positive class for all the examples will achieve an accuracy of 90% on the dataset. We cannot say that the model has learnt anything from the data.
11. $J(1, 1) = \frac{1}{2*4} \sum_{i=0}^{4} [x_i + 1 - y_i]^2 = (1/8) (0 + 8 + 8 + 0) = 1$

12. $h(3) = 1.3*x + 3.5 = 8$
13. If the range of values of data varies widely, objective functions will not work properly without normalization. For example, many classifiers calculate the distance between two points by the Euclidean distance. If one of the features has a broad range of values, the distance will be governed by this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance.
14. As there is no interaction between any two features, the update equation for each variable indicates that it only depends on derivative of loss function wrt the variable itself and no other variable. Coefficient for that feature will be halved to get the same output.
15. As the complexity of the model increase, we have more parameters to represent observed data while training. This results in a decrease in the loss.
16. Updates from both stochastic gradient descent and mini batch gradient descent are approximations of the true gradient and are hence noisy. Batch gradient descent computes the true gradient over the entire dataset and hence, the cost is guaranteed to reduce or remain the same.
17. Mini gradient descent works by computing gradients on small batches and making updates iteratively over the entire dataset. Hence, statement (i) is false. As the number of computations over one epoch roughly remain the same, we cannot say that mini batch gradient descent is faster than batch gradient descent. Hence, statement (ii) is false.
18. Gradients calculated on larger mini batches can be looked at as an average of the gradient approximations of many smaller mini batches. Hence, we will have less variance in our updates.