Lab 2. Linear Regression with TensorFlow

In this lab, we will use TensorFlow to create a simple linear regression model and use it to make some predictions from the training data. This data correlates the GDP per capita of a country vs the life satisfaction that the citizens report.

# Part 1. Plot the input data

1. Plot the data as a scatter plot using matplotlib. Label the X-axis ‘GDP per capita ($). Label the Y-axis ‘Life Satisfaction’

# Part 2. Create the Linear Regression TensorFlow model.

Recall that in the linear regression model, the predictions are a linear combination of the input features. In other words, the predicted v alue can be computed using the following equation:

+ + … + where:

* , , … , are the input features
* , , … , are the corresponding weights
* *b* is the bias term (also called the intercept term)

We can express this equation compactly as:

Where

* is the input feature vector.
* is the weight vector.
* is the inner product of vectors **v** and **w**.

It is often more convenient to handle vectors as matrices with a single column (i.e. a column vector). The inner product (**x**,**w**) is replaced with the matrix dot product where is the transpose of the column vector **x**.

Lastly, we can compute predictions for many instances at once by putting all the input features into a matrix **X**. The vector containing the predictions for every instance can be computed using the following equation:

where

is the prediction vector, containing the prediction for all *m* instances

= Each row represents an instance. Each column represents a feature. Note that the matrix dot product X dot w returns a column vector, so when adding the bias term *b* we mean adding that value to each and every element in the column vector (this is called *broadcasting*).

Create a graph containing:

* A constant X initialized with X\_train, which contains 70 percent of the input features of the training instances. In this particular lab, there is only a single feature per instance (the GDP per capita)
* A constant y, initialized with y\_train, which contains 70 percent labels of each instance (the life satisfaction).
* A variable b, representing the bias (initialized to 0.0)
* A variable w, representing the weight vector (initialized to a column vector full of zeros, using tf.zeros()). Since there is only one input feature per instance, this column vector contains a single row (it is a matrix with a single item)
* An operation y\_pred that computes the equation presented above .You will need to use tf.matmul()*.*
* Don’t forget to initialize your variables when you run your session.

# Part 3. Run the graph.

1. Start the graph
2. Run the init() operation and evaluate the predictions *y\_pred.*  Since both variables w and b are initialized with zeros, you should get a vector full of zeros.
3. Add the mean square error cost function given by Use the tf.reduce\_mean() and tf.square()TensorFlow methods to calculate the cost function.
4. Again, initialize the session and evaluate the cost function. Note that the cost function is quite high as is to be expected since we have only run this for a single instance of w and b.
5. Now we will find the optimal values for the model parameters **w** and b. In order to do this, we will use the standard Gradient Descent optimization algorithm. Use the standard tensorflow method tf. gradientDescentOptimizer(learning\_rate).minimize(cost\_function) to automatically optimize our weights and bias to achieve the optimal line fit.
6. In the evaluation of the TensorFlow graph, start a session, initialize the variables, then write a loop that will repeatedly evaluate the assignment operations 1000 times. Every 100 times, evaluate the MSE and print it out. Hopefully, the MSE should drop below 1.0 and we’d like a target of about 0.18.
7. Add summaries for the operations and variables and write a log to the disk. Open the Tensorboard web page and view the value of the cost function as well as the weights and biases to view the results of the optimizer.
8. Plot the data with matplotlib and use the final values of **w** and b to visualize the line through the points that best fits the data.
9. Now test the model by inputting evaluating it and feeding it the test data and test labels to see how well it performs.