Linear Regression Model Learning using Gradient Descent

Athlete's Weight-Age-Height Linear Model

Fairuz Astra Pratama
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
13514104@std.stei.itb.ac.id

***Abstract*—Linear regression is a statistical method of making a linear equation that can predict and/or model a dataset with the least error. On the other hand, gradient descent is an optimization algorithm that can find the best parameter to minimize a function with. Combining these two method, we can design a machine learning algorithm that can accept a dataset and return a linear relationship between its attribute that can be used to predict data value in the same domain.**

Keywords—machine learning, linear regression, gradient descent

I. Introduction

In both statistic and machine learning, linear regression is a well-known method for its simplicity; making it one of the most common introductory topic in undergraduate computer science program. Nevertheless, this method is quite effective to be implemented at several domains such as determining the relationship between the weight and height of a person. Unfortunately, as the name suggest, this method is ineffective where the relationship between data's attributes is not linear, such as the data of population amount and growth rate.

Gradient descent, on the other hand, is one of the more common optimization algorithm, that can be used to find the values of coefficient in a function that minimizes its cost. Seeing that the model of linear regression is a form of function, we can use gradient descent to design an algorithm that perform linear regression to data to produces a function that map the relationship between its attributes. There are also many more uses for gradient descent in machine learning, such as using logarithmic regression rather than linear.

This paper will be focused on cover both the theoretical basis and implementation details of this algorithm; and will be using basketball athlete medical record as case study, where the algorithm will output the relationship model between the weight, age and height of athlete.

II. Literature Study

A. Linear Regression

Although primarily and originally used in statistic, linear regression has been used by machine learning to model relationship between two or more attribute. The basic concept is very simple; a linear model represents a relationship between data in form of a linear equation. This equation will accept several inputs (x's) and output its prediction for that input value (y); just like a standard equation. The goal of linear regression is to make such model where it can accurately predict an attribute's value of a data instance given the value of the other attribute with minimal error.

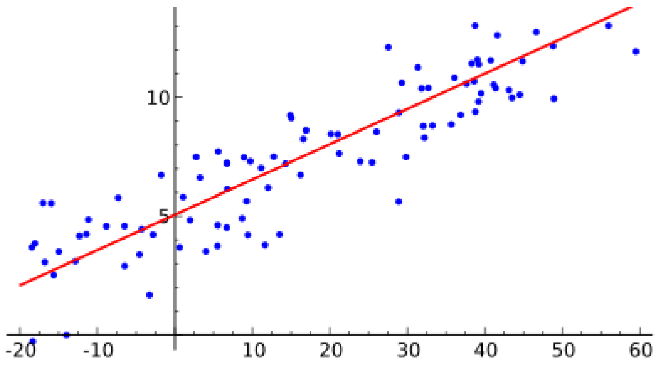

Figure 1 Example of Linear regression in 2D Cartesian Diagram source: https://en.wikipedia.org/wiki/Linear_regression

For example, the red line above is a linear model that can be used to predict the value of 'y' given the 'x' of the blue dots. For this case study, imagine we have a data containing health information with "weight" and "height" as the attributes, and we want to know how much does a person's weight given the height. The linear model needed for this data will be:

$$weight = c_1 * height + c_2$$

Which is very similar to a linear equation of a line in 2 dimensional plane ($y = mx + c$). Now, imagine our data had another field in it named "age." We will need to add another variables and constant to the equation, which resulted in:

$$weight = c_1 * height + c_2 * age + \ c_3$$

Which is a linear equation of a plane in 3 dimensional plane. If our data had yet another attribute, we just need to continue adding constant and variables pair to our equation.

In the example above, we see several constants in our equation. These constants will determine the output of the function, and as such, how accurate the prediction of the model. As stated before, linear regression goal is to assign a value to each constant as such the formula can imitate the relationship between attributes as closely as possible.

There are several methods to find the best constant value, from using statistical properties of the data, to using the "Ordinary Least Squares" method. In this paper, we will focus on one of the possible method to programmatically make a linear model from a set of data, which is gradient descent.

B. *Gradient Descent*

Gradient descent is a kind of optimization algorithm; it is generally used for finding the values of parameters of a function that will results in the lowest or highest return value. It does this by repeatedly evaluating the return value of a parameter set in that function, and generating a better parameter set (resulting in an even lower/higher result) from the analyzation result.

The way gradient descent generate a better parameter set from the previous one is by using a concept called gradient, more generally referred to as derivative. Derivative is a concept in calculus that can be easily described as the slope at a certain point of a plane (in a three dimensional space). Imagine a three dimensional function ($z = ax + by + c$) forming some sort of hill. The derivative of that function will take a coordinate, and return how much change in z will happen if you moved in the positive x and y direction from that point.

This in itself is not very useful, but there is another concept called partial derivative. Compared to full derivative, partial derivative of x in a three dimensional function used above will return how much change in z will happen if you moved in the positive x direction from the given point. By using this information, we can identify which direction each parameter should be moved by (smaller or bigger number) in order to reach the desired output.

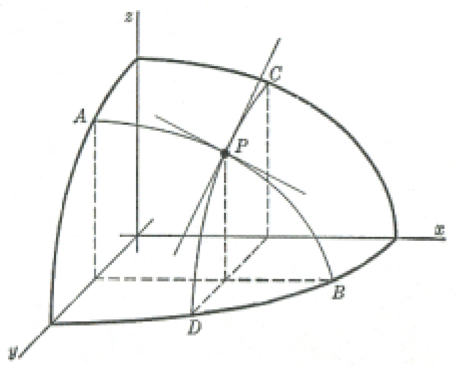

Figure 2 Example of Partial Derivative in 3D Plane source: http://www.solitaryroad.com/c353.html

At point P in the image above, the two straight line forming a cross are the partial derivative of x and y (where z is the function output.) In the image the partial derivative of x at point P is negative (pointing down as the value of x increase.) This means that if we want to maximize the height of z at point P, the x value should be decreased, and as we can see in the graph, the z value is indeed increase to the left of point P.

Now, imagine that we want to maximize the output of a function, and the partial derivative of x at the current parameter is positive. We should then increase the value of x for the next iteration, since it will result in a higher output.

III. Method

By viewing linear regression as an optimization method; where the equation's constant value is we constantly changed in order to minimize error; we can use gradient descent to make our linear model from any data. For this case study we will try and make the model that represent the relationship between height, weight and age of human. The linear model for this data is:

$$height = c_1 * weight + c_2 * age + c_3$$

To determine the best value for the three constant in the formula above, we must first declare the error function we want to minimize. In this case, we will use the average of squared difference between actual output and the one predicted by the model for all data instance. We can write this formula into:

$$Error = \frac{1}{N}\sum_{i=1}^{N}(desired_i - actual_i)^2$$

Where N is the amount of data in the training set. Substituting desired and actual height using the previous linear model for this data, will result in:

$$Error = \frac{1}{N}\sum_{i=1}^{N}(height_i - (c_1 * weight_i + c_2 * age_i + c_3))^2$$

Lastly, we need to calculate the partial derivative formula of every constant in the above formula in order to make the gradient descent algorithm. After some calculation the partial derivative for all three constant can be defined as such:

$$\frac{\partial}{\partial c_1}(Error) = -\frac{2}{N}\sum_{i=1}^{N} weight * (c_1 * weight_i + c_2 * age_i + c_3)$$

$$\frac{\partial}{\partial c_2}(Error) = -\frac{2}{N}\sum_{i=1}^{N} age * (c_1 * weight_i + c_2 * age_i + c_3)$$

$$\frac{\partial}{\partial c_3}(Error) = -\frac{2}{N}\sum_{i=1}^{N}(c_1 * weight_i + c_2 * age_i + c_3)$$

After we know the error formula and the partial derivative for each constant, the gradient descent algorithm itself is quite simple. The pseudocode for the implementation used in this case study is as such:

```
1. Initialize the constants with a semi-random value
2. Calculate the error of the current model
3. Check if the error change between iteration is too little, or if too much iteration has been executed
    a. Stop if one of the condition is true
    b. Output the final model and error value
4. Calculate the partial derivative for all constant
5. Subtract each of the model's constant with its partial derivative (to try reaching for the minimum error) times the learning rate (to prevent model from jumping over the minimum point)
```

IV. Results

After implementing the gradient descent algorithm described above (using python for this experiment), we can supply the program with training data to generate its model. For this case study we will use the statistic online computational resource provided by the University of California which can be accessed at http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_MLB_HeightsWeights[n]. The data contains 1035 records of heights (inch), weight (pound) and age of Major League Baseball player.

After several experimentations, we reach the minimum error for the training data supplied by using constant instantiation of 0, learning rate of 0.0001 and 1000 maximum iteration. The resulting model from this training is as such:

$$height = 2.53 * weight + 0.55 * age - 0.67$$

The model above yields an average error of 331.73. Since the error formula is the square of difference, using square root, we can determine that on average the model's height prediction is off by about 18.2 inch. We can also visualize this model and the data distribution by using a three dimensional graph below:

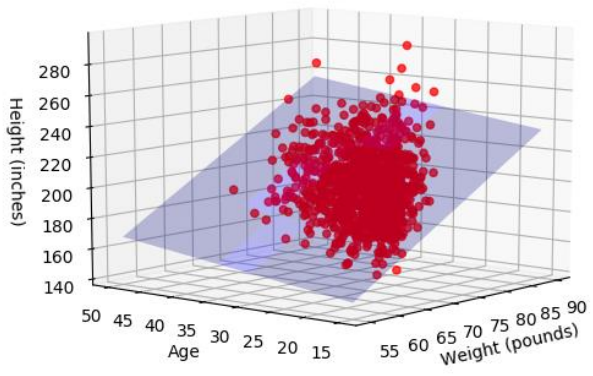

Figure 3 Scatter Plot and Model Plane Results of Experiment

Where the blue plane is the linear model of the data, and the red circles are the instances of the training data.

V. Discussion

An error margin of 18.2 inch is actually not that bad considering how diverse human weight-height ratio are. Even so, there is some factor in gradient descent algorithm that may result in suboptimum model. In this section, we will cover one of the most probable cause, which is local minimum.

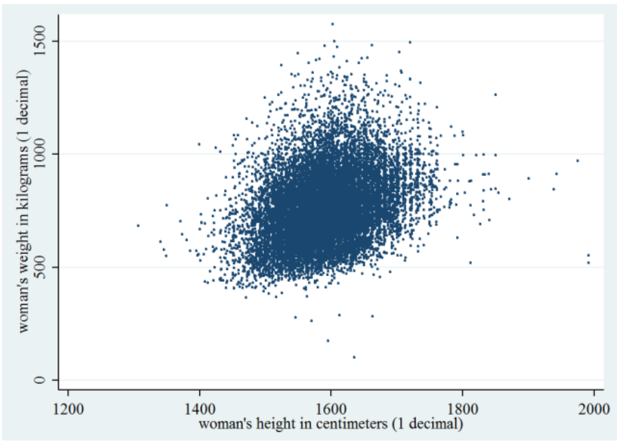

Figure 4 Scatter Plot of 2014 Egyptian Female Weight-Height Data According to the Demographic and Health Survey (DHS) source: https://opendata.stackexchange.com/questions/7793/

The error function that our algorithm tried to minimize is quite complex, forming a four dimensional (there are three input since we tried to get the values for the constant, not the actual variables) quadratic equation. Compared to simple equation, such as an equation that formed a mountain or bowl; no matter where the initial point is, we will always reach the maximum / minimum since the partial derivative at all position will point at the maximum / minimum point.

This, however, is most likely not the case with more complex function. There is a high chance of several "valley" called local minimum to exists in our function, but only one is the optimum solution (lowest value.) Since at every local minimum, the partial derivative for all variables is positive (pointing upwards), our algorithm will be stuck inside one of these points according to the initiation value, not knowing whether or not this point is the absolute minimum error for the model.

After further experiment with the initiation value, it is confirmed that different initial value will yield different model, even if the maximum iteration hadn't been crossed yet. For example, an initial constant value of $c_1 = 50$, $c_2 = 0$, $c_3 = -100$ will yield a model with a better average error of 306.07 (off by about 17.5 inch) even though both had reached had stopped before the max iteration (reached a minimum point.) The resulting model is as such:

$$height = 3.81 * weight + 0.74 * age - 100.62$$

There are several methods that exist to mitigate this effect this, one of such is the stochastic gradient descent; that use stochastic approximation to try and avoid local minimum. This method uses only a random part of the training set during calculating the partial derivative at each step; rather than using the entire dataset. This way, the variance of parameter update can be reduced, and may "leap over" local minimum, converging at the global minimum.

This method also make the algorithm run much faster, making it more feasible to run a bigger number of iteration process. However, it is best to lower the learning rate in this method, since the variance of partial derivative is much higher than the usual gradient descent; and it may be hard to determine a good learning rate value without using trial and error method.

VI. Conclusions

Using gradient descent to implement linear regression in machine learning is very efficient for the right domain. This method is also very easy to implement, and can be used as baseline to the other machine learning method to compare performance. However, this method is less effective against data with non-linear relationship and complex data; since the algorithm may only resulted in a model with error that points towards a local minimum (rather than the global minimum).

To remedy that, stochastic gradient descent may be used with lower learning rate and more iteration to try and "skip" the local minimum in the error function.

References

[1] Jason Brownlee (2016) *Linear Regression for Machine Learning* [Online] Available: http://machinelearningmastery.com/linear-regression-for-machine-learning/ [Accessed 2 May 2017]

[2] Jason Brownlee (2016) *Gradient Descent for Machine Learning* [Online] Available: http://machinelearningmastery.com/gradient-descent-for-machine-learning// [Accessed 2 May 2017]

[3] Matt Nedrich (2014) *An Introduction to Gradient Descent and Linear Regression* [Online] Available: https://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/ [Accessed 3 May 2017]

[4] Statistics Online Computational Resource (No date) *SOCR Data MLB HeightsWeights* [Online] Available: http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_MLB_HeightsWeights [Accessed 1 May 2017]

[5] Unsupervised Feature Learning and Deep Learning (No date) *Optimization: Stochastic Gradient Descent* [Online] Available: http://ufldl.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/ [Accessed 3 May 2017]

[6] https://en.wikipedia.org/wiki/Linear_regression [Accessed 3 May 2017] as image sources

[7] http://www.solitaryroad.com/c353.html [Accessed 3 May 2017] as image sources

[8] https://opendata.stackexchange.com/questions/7793/ [Accessed 3 May 2017] as image sourc

Declaration

With this, the writer hereby declare that this paper is written on his own, not an adaptation, or translation of someone else's paper, and not the result of plagiarism.

Bandung, 5 May 2017

[illegible]

Fairuz Astra Pratama - 13514104