華南理工大學

South China University of Technology

The Experiment Report of *Machine Learning*

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Zhishang Zhou

Supervisor:
Mingkui Tan and Qingyao Wu

Student ID:
201720144931

Grade:
Graduate

December 15, 2017

Linear Regression, Linear Classification and Gradient Descent

***Abstract*—In this experiment report, I will show you how Linear Regression, Linear Classification and Gradient Descent works. Firstly, I will introduce how those things being proposed. Secondly, some details about those method and theory will being displayed. Thirdly, detailed experiments will be shown. Finally, I will give a brief conclusion.**

I. Introduction

IN statistics, linear regression is a linear approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression.
In the field of machine learning, the goal of statistical classification is to use an object's characteristics to identify which class (or group) it belongs to. A linear classifier achieves this by making a classification decision based on the value of a linear combination of the characteristics. An object's characteristics are also known as feature values and are typically presented to the machine in a vector called a feature vector. Such classifiers work well for practical problems such as document classification, and more generally for problems with many variables (features), reaching accuracy levels comparable to non-linear classifiers while taking less time to train and use.
Generally, we iteratively solve those two problem by using Gradient Descent method. Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point. If instead one takes steps proportional to the positive of the gradient, one approaches a local maximum of that function; the procedure is then known as gradient ascent.

II. Methods and Theory

Linear Regression: Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable. For example, a modeler might want to relate the weights of individuals to their heights using a linear regression model. Before attempting to fit a linear model to observed data, a modeler should first determine whether or not there is a relationship between the variables of interest. This does not necessarily imply that one variable causes the other (for example, higher SAT scores do not cause higher college grades), but that there is some significant association between the two variables. A scatterplot can be a helpful tool in determining the strength of the relationship between two variables. If there appears to be no association between the proposed explanatory and dependent variables (i.e., the scatterplot does not indicate any increasing or decreasing trends), then fitting a linear regression model to the data probably will not provide a useful model. A valuable numerical measure of association between two variables is the correlation coefficient, which is a value between -1 and 1 indicating the strength of the association of the observed data for the two variables.

A linear regression line has an equation of the form $Y = a + bX$, where X is the explanatory variable and Y is the dependent variable. The slope of the line is b, and a is the intercept (the value of y when $x = 0$).

Linear Classification: Given training data (x_i, y_i) for $i = 1...n$, with $x_i \in R_m$ and $y_i \in 1, 1$, learn a classfier $f(x)$ such that $y_i f(x_i) > 0$ for a correct classification. And $f(x)$ often takes the form just like Linear Regression's model.

Gradient Descent: To explain Gradient Descent I will use the classic mountaineering example. Suppose you are at the top of a mountain, and you have to reach a lake which is at the lowest point of the mountain (a.k.a valley). A twist is that you are blindfolded and you have zero visibility to see where you are headed. So, what approach will you take to reach the lake? The best way is to check the ground near you and observe where the land tends to descend. This will give an idea in what direction you should take your first step. If you follow the descending path, it is very likely you would reach the lake. To represent this graphically, notice the below graph.

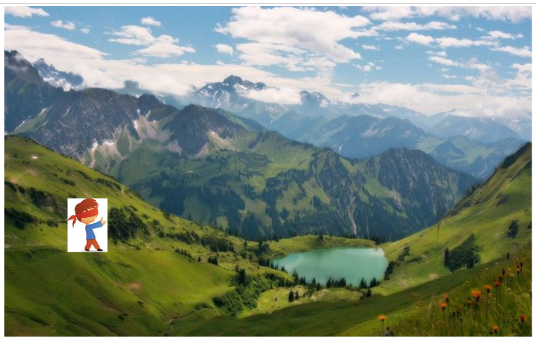
Fig. 1. Mountaineering example

Let us now map this scenario in mathematical terms. Suppose we want to find out the best parameters θ_1 and θ_2 for our learning algorithm. Similar to the analogy above, we see we find similar mountains and valleys when we plot our cost space. Cost space is nothing but how our algorithm would perform when we choose a particular value for a parameter.

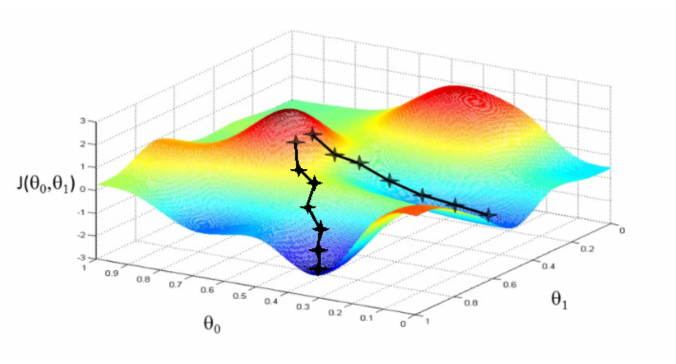

Fig. 2. Gradient in mathematics view

So on the y-axis, we have the cost $J(\theta)$ against our parameters θ_1 and θ_2 on x-axis and z-axis respectively. Here, hills are represented by red region, which have high cost, and valleys are represented by blue region, which have low cost.

III. Experiments

A. Dataset

Linear Regression uses Housing in LIBSVM Data, including 506 samples and each sample has 13 features. We divide it into training set, validation set as 80:20. Linear classification uses australian in LIBSVM Data, including 690 samples and each sample has 14 features.We divide it into training set, validation set as 80:20.

B. Implementation

Linear Regression and Gradient Descent:

1. Load the experiment data. I use load_svmlight_file function in sklearn library.
2. Devide dataset. I divide dataset into training set and validation set using train_test_split function with the split ratio 0.2. Test set is not required in this experiment.
3. Initialize linear model parameters.I use normal distribution to initialize my model parameters with $\mu = 0$ and $\sigma = 0.1$.
4. Choose loss function and derivation: I choose mean square loss function to demonstrate the model's behavior.
5. Calculate full batch gradient G toward loss function from all samples.
6. Denote the opposite direction of gradient G as D.
7. Update model: $W_t = W_{t-1} + \eta D$. η is learning rate, a hyper-parameter that we can adjust.
8. Get the loss L_{train} under the training set and $L_{validation}$ by validating under validation set.
9. Repeate step 5 to 8 for several times, and drawing graph of L_{train} as well as $L_{validation}$ with the number of iterations.

Linear Classification and Gradient Descent

1. Load the experiment data. I use load_svmlight_file function in sklearn library.
2. Divide dataset into training set and validation set. I divide dataset into training set and validation set using train_test_split function with the split ratio 0.2. Test set is not required in this experiment.
3. Initialize SVM model parameters. I use normal distribution to initialize my model parameters with $\mu = 0$ and $\sigma = 0.1$.
4. I use hinge loss function to demonstrate my model's behavior. Specificly, I want to minimize $\frac{||w||^2}{2} + \frac{C}{n}\sum_{i=1}^{n} max(0, 1 - y_i(\mathbf{w}^T\mathbf{x} + b))$, and C is choosen to be 10.
5. Calculate gradient G toward loss function from all samples.
6. Denote the opposite direction of gradient G as D.
7. Update model: $W_t = W_{t-1} + \eta D$. η is learning rate, a hyper-parameter that we can adjust.
8. Select the appropriate threshold, mark the sample whose predict scores greater than the threshold as positive, on the contrary as negative. Here the threshold is simply set as 0.5. Get the loss L_{train} under the trainin set and $L_{validation}$ by validating under validation set.
9. Repeate step 5 to 8 for several times, and drawing graph of L_{train} as well as $L_{validation}$ with the number of iterations.

C. Results

We run the training loops for 1000 times and plot the loss value shown as below:

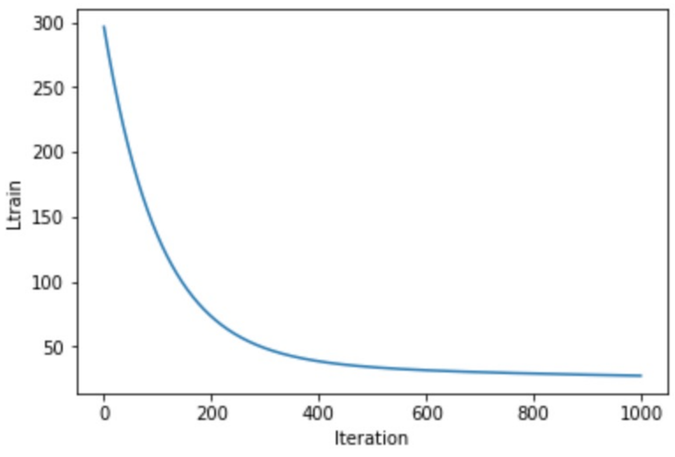

Fig. 3. Linear Regression's L_{train}

IV. Conclusion

In this experiments, I have implemented a linear regression model which behaves well using gradient descent optimization method. I also made a linear SVM classification method with penalty which classifies the data very well. As I implemented those models and methods from head to tail, I faced a lot of bugs which really confused me and depressed me. As soon as I made my first linear regression model, the works afterwords become steady. I really learned a lot in it.

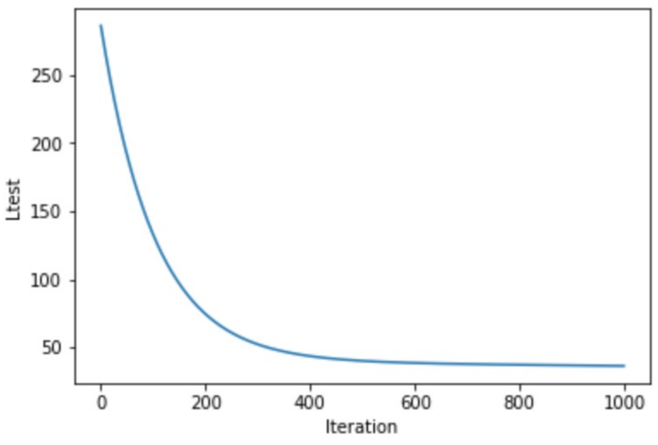

Fig. 4. Linear Regression's $L_{validation}$

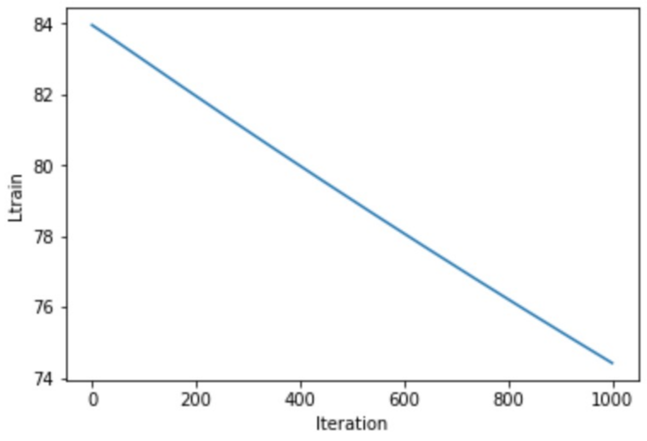

Fig. 5. Linear Classification's L_{train}

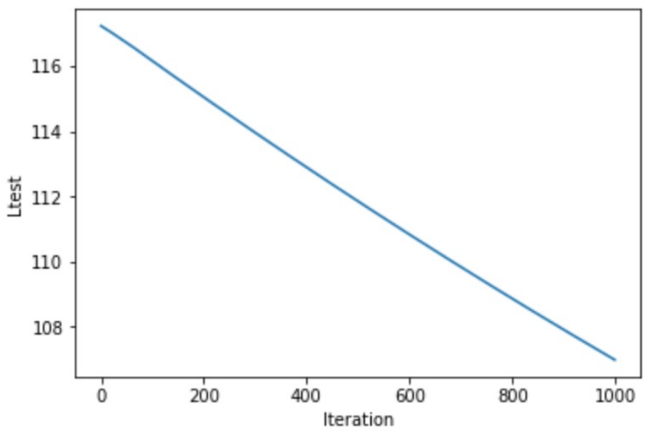

Fig. 6. Linear Classification's $L_{validation}$