CS6491 Project
Convex Optimization of Logistic Regression

Zongyuan Lu, 56859500

April 2022

1 Introduction

Logistic Regression algorithm is not a regression model, but one used to deal with classification problems labeled as dichotomous. Logistic Regression algorithm fits the data into a logistic function, which enables the prediction of the probability of the occurrence of an event. The essence of Logistic Regression algorithm is that the data is assumed to obey this distribution, and then the parameters are estimated using the maximum likelihood estimation. Specifically, Logistic function is

$$L(z)=\frac{1}{1+e^{-z}} \tag{1}$$

Regression function is

$$\mathrm{y}=a+bx \tag{2}$$

The value of the logistic function represents the probability, with eigenvalue equals 1 when y is greater than 0.5 (usually), or eigenvalue equals 0 when y is smaller than 0.5. Additionally, Logistic Regression algorithm generally takes cross-entropy as the loss function, which is denoted as

$$J(\theta)=-\frac{1}{m}\sum_{i=1}^{m}\left[\left(y^{(i)}\log h_\theta\left(x^{(i)}\right)+\left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right]\tag{3}$$

where m is number of samples, i is the ith sample, $h_\theta(x)$ is the prediction of the sample, y is the sample label.

In general, Logistic Regression algorithm is fast and easy to implement and interpret. However, it also requires complex feature processing and engineering, is easy to be underfitting, and is even unsuitable for large eigenvalue space.

2 Problem Statement

Logistic regression is widely used for classification in machine learning process, how to optimize logistic regression algorithm to enhance its efficiency is a valuable problem. Before solving it, we should firstly know whether logistic regression function is convex or not.

3 Solution Designed

3.1 Concavity of Logistic Regression

Logistic function is

$$L(z)=\frac{1}{1+e^{-z}} \tag{4}$$

Specifically,

$$z=w_0+w_1x1+w_2x_2+...+w_nx_n=\sum_{i=0}^{n}\theta_ix_i=\theta^Tx \tag{5}$$

Derive equation (4) and equation (5), the optimization problem of logistic regression can be shown as:

$$\arg\min_{\theta} L(\theta) = \sum_{i=1}^{n} \log(1 + \exp(-y_i \theta^T x_i)) \tag{6}$$

We can calculate the second derivative of equation(6) to prove concavity of this problem. Let

$$l_i(\theta) = \log(1 + \exp(-y_i \theta^T x_i)) \tag{7}$$

Calculate the first derivative of equation(7):

$$\bigtriangledown L_i(\theta) = \frac{-y_i x_i}{1 + \exp(y_i \theta^T x_i)} \tag{8}$$

Calculate the second derivative:

$$\bigtriangledown^2 L_i(\theta) = \frac{\exp(y_i \theta^T x_i)}{1 + \exp(y_i \theta^T x_i)^2} x_i x_i^T \tag{9}$$

Certainly, $\bigtriangledown^2 L_i(\theta) \geq 0$, which means that we can use convex optimization approach to solve the optimization problem of logistic regression.

3.2 Gradient Descent Method

A classic approach of solving logistic regression convex optimization problem is gradient descent method. It finds appropriate value by adjusting parameters continuously and using gradient information.

We already know the loss function of logistic regression $J(\theta)$ in equation (3), and the procedure of gradient descent method can be denoted as:

$$\theta_j := \theta j - \alpha \frac{\delta}{\delta_{\theta_j}} J(\theta) \tag{10}$$

Assuming that we use random gradient descent method, the descent is denoted as:

$$\begin{aligned}
\frac{\delta}{\delta_{\theta_j}} J(\theta) &= -\frac{1}{m} \sum_{i=1}^{m} \left[\left(y^{(i)} \log h_\theta \left(x^{(i)} \right) + \left(1 - y^{(i)} \right) \log \left(1 - h_\theta \left(x^{(i)} \right) \right) \right) \right] \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \frac{1}{g(\theta^T x^{(i)})} - (1 - y^{(i)}) \frac{1}{1 - g(\theta^T x^{(i)})} \right] \frac{\delta}{\delta_{\theta j}} g(\theta^T x^{(i)}) \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \frac{1}{g(\theta^T x^{(i)})} - (1 - y^{(i)}) \frac{1}{1 - g(\theta^T x^{(i)})} \right] g(\theta^T x^{(i)})(1 - g(\theta^T x^{(i)})) \frac{\delta}{\delta_{\theta_j}} \theta^T x^{(i)} \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} (1 - g(\theta^T x^{(i)})) - (1 - y^{(i)}) g(\theta^T x^{(i)}) \right] x_j^{(i)} \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} - g(\theta^T x^{(i)}) \right] x_j^{(i)} \\
&= \frac{1}{m} \sum_{i=1}^{m} \left[h_\theta(x^{(i)}) - y^{(i)} \right] x_j^{(i)}
\end{aligned} \tag{11}$$

Logistic function shown in equation (1) has an important property:

$$L'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = L(x)(1 - L(x)), \tag{12}$$

which is used in the third row of equation (11).

Hence, the formula of gradient descend can be represented as:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \tag{13}$$

After knowing the principle of gradient descend method, we can use it to solve the real problem with logistic regression algorithm.

4 Illustrative Examples

There are two many problems that need classification algorithms to solve in machine learning. I choose famous MNIST dataset to study the positive influence of gradient descend method on the logistic regression classification algorithm.

4.1 Real Problem Introduction

MNIST is a set of preprocessed handwritten digital image data sets. It provides an opportunity for beginners of machine learning to practice and solve problems with learned algorithms on real data. Because many machine learning tutorials take MNIST as an introductory project, it is also known as "Hello world" in the field of machine learning.

Each sample in MNIST is a grayscale image with a length of 28 and a width of 28, which contains a number of 0-9. What we need to do is to build a model based on the training data to identify the numbers in the input picture. This is a typical classification problem. The input of each sample is a 784 dimensional vector: a picture has $28 * 28 = 784$ pixels, and each point uses a floating-point number to represent its brightness; The output is a 10 dimensional vector, and the ten components respectively represent the possibility that the number in the input graph is 0-9, of which the most likely is the result predicted by the algorithm.

A number of algorithm can be chosen to solve MNIST classification problem, such as logistic regression, naive Bayes' algorithm, support vector machine(SVM), decision tree and K-nearest neighbor, etc. In this project, we mainly focus on the use of logistic regression in MNIST classification.

There are lots of tools to solve classification problems like Python and Matlab. I will use Matlab here to show the core code and the test result of logictic regression classification.

4.2 Simple Logistic Regression

4.2.1 Introduction

According to the properties of MNIST, I randomly select 50% of the dataset as training set, another 50% of the dataset as test set. After pre-processing, each image in MNIST can be regraded as a 784-dimensional vector.

In this section, I choose the linear model without gradient descent to estimate the efficiency of algorithm. Fortunately, there is no need that we implement this algorithm from zero, libraries such as $glmfit$ in Matlab can help us implement logistic regression easily.

4.2.2 Core Code

In this section, I will show the core code of logistic regression algorithm in *logisticRegression.m*. The pre-processing of original dataset is ignored, which is irrelevant to the final result.

logisticRegression.m

```
function [] = logisticRegression()
    % Training the linear logistic regression model
    regression_model = glmfit(trainX, trainY);
    result = glmval(regression_model, testX, 'logit');
    % Calculating the test accuracy
    acc = 0.;
    for i = 1 : test_num
        % Test matches if the corresponding result is greater than 0.5
        if result(i) > 0.5
            acc = acc + 1;
        end
    end
    acc = acc / test_num;
```

4.2.3 Result Display

In this experiment, I randomly select training set and test set twice and divide the experiment into two trials. Core code shown before will output the test accuracy of each trial. Additionally, I also calculate the mean and the deviation of two trials, in which I can observe the stability of logistic regression classification algorithm.

The above data is shown in Table 1.

Table 1: Result of the linear logistic regression without gradient descent method.

Trial	**1**	**2**	**mean(std)**
Logistic Regression	0.9455	0.9545	0.9500(0.0064)

4.3 Logistic Regression with Gradient Descent Method

As mentioned above, gradient descent method can be used in classification process to increase the accuracy of logistic regression algorithm. In this section, I will show the core code of gradient descent method, list the result and compare the result to above.

4.3.1 Core Code

There are some important parameters in gradient descent method, such as study rate α, number of iterations *num_iters* and *theta*, etc. The core code of function *gradientDescent* is shown in *gradientDescents.m*.

gradientDescents.m

```matlab
function [theta] = gradientDescents(X, y, theta, alpha, num_iters)

% The number of training data.
m = length(y)
for iter = 1 : num_iters
    theta_temp = theta;
    for j = 1 : length(theta)
        % Variety 'temp' means sum of subtrahend of equation (13)
        temp = 0;
        for i = 1 : length(X)
            % 'L_function' represents the logistic function.
            % Implementation of equation (13)
            L_function = 1 / (1 + exp(-dot(theta', X(i, :))));
            temp = temp + alpha * dot((L_function - y(i)), X(i, j));
        end
        theta_temp(j) = theta - temp;
    end
    theta = theta_temp;
end
```

Then we can call the function *gradientDescents*() in the logistic regression algorithm and calculate the accuracy.

4.3.2 Result Display

Certainly, accuracy of the gradient descent method depends on the parameter learning rate α, I keep the number of iterations as 10, adjust α to 0.05, 0.1 and 0.5 and calculate test accuracy of each trial. The result data is shown in Table 2.

Compared to result data in Table 1, we can find that the accuracy gets better after using gradient descent method in logistic regression classification algorithm. More important, the efficiency of gradient descent depends on the number of iterations and parameters selected.

Table 2: Result of the linear logistic regression without gradient descent method.

Trial	**1**	**2**	**mean(std)**
$\alpha = \mathbf{0.05}$	0.9360	0.9920	0.9780(0.0191)
$\alpha = \mathbf{0.1}$	0.9360	0.9920	0.9640(0.0396)
$\alpha = \mathbf{0.5}$	0.8980	0.9925	0.9453(0.0668)

5 Conclusion

In this project, I study the convex optimization of logistic regression algorithm. First, I prove that the concavity of logistic regression optimization problem. Second, the gradient descent method is found that can be used to solve the optimization problem. Third, I write Matlab program to implement logistic regression without gradient descent method and calculate accuracy. Fourth, after derivation of gradient descent formula, I write program of regression with optimization and compare the accuracy with the simple logistic regression. Finally, we can draw a conclusion that the gradient descent method does improve the accuracy of logistic regression algorithm, as long as the parameters are appropriate.

The source code of this project will be submitted with this report.

6 Future Works

Because the device and time limit, I think the parameters in this project are not the most efficient, the number of iterations are also insufficient, but it does cost me too much time to run the whole problem. This optimization problem can be solved better if the devices are powerful enough, which I should do in the future.