Comparison of Gradient Descent and Least Squares Algorithms in Deep Model

Lianchao Jin and Shengming Jiang
Marine Internet Technology Laboratory, Shanghai Maritime University, Shanghai, China
Email: 18830433390 @163.com; smjjiang@shmtu.edu.cn

Abstract. Deep neural networks have a wide range of applications in stock forecasting, big data forecasting of influenza outbreaks, prediction of game outcomes and so on. They all involve a common problem—regression prediction problems. Deep model can be applied to solve regression prediction problems by optimizing model structure through optimization algorithm. The optimization algorithm generally uses the gradient descent method to optimize the model structure by inputting much real data. When the previous mathematical model solves the problem of regression analysis, one of the most basic optimization methods is the least squares method. This paper studies the performance of the vgg16 convolutional neural network model combined with transfer learning reconstruction when the amount of data is large. By analyzing their algorithm structure to more clearly understand how they are optimized for deep model, better algorithms can be used for future model optimization.

Keywords. Deep model; least squares method; gradient descent; transfer learning.

1. Introduction
Deep neural network is a kind of structure, not an algorithm, and machine learning algorithm is the driving force of the development of artificial intelligence, helping people to deal with things and to find some unknown things. In many machine learning optimization algorithms, such as Gradient Descent, AdaGrad and Adam. Gradient descent is the most commonly used optimization algorithm. Solving model parameters is also called the unconstrained optimization problems. At present, gradient descent algorithm is mainly divided into three kinds: standard gradient descent (GD), stochastic gradient descent (SGD) and batch gradient descent (BGD), used to recursively approximate the minimum deviation model, such as Artificial neural networks and Logistic regression, is widely used in data mining, and another classical mathematical methods is least squares algorithm.

In supervised learning, in order to solve the regression problem fitting input samples (Xi, Yi), we need to define the fitting function. For samples with only a single feature, we can define the fitting function as:

\[ h\theta (x) = \theta_0 + \theta_1 x \]  

(1)

Next, we need to assume a loss function to evaluate the quality of our fitting function, the direct application here is the sum of the squares of the error between the real value and the calculated value. Specifically, the error has a special name of “Residual” (deviation between true value and output value) [1]. We define our loss function using the L2-norm, the sum of the squares of the residuals, defined as:
\[ f(\theta, \theta) = \sum_{i=1}^{n} (h_{\theta}(x) - y) \]  

(2)

2. Algorithm Structure

2.1. Gradient Descent Algorithm Structure

The more intuitive explanation of the gradient descent method is like a person going down from the top of the mountain, but he doesn't know how to go down the mountain. He just walks down the steep place step by step. Every time he goes to a position, the current gradient is calculated [2], such as the function \( f(x, y) \), Find partial derivatives for \( x, y \), the obtained gradient vector is \( (\partial f/\partial x, \partial f/\partial y)^T \), he will step along the reverse direction of the gradient, Go down one step until he reach the place where the gradient is 0. The person who goes down the mountain may feel that he has reached the bottom of the mountain, but it is possible that he only walks to a place halfway up the mountain. This is where the gradient descent algorithm is limited. Due to the uncertainty of the initial parameters, with the process of optimization, the model may not converge. For complex model optimization, in the case of saddle point or valley, the optimization is very difficult. The least square algorithm is relatively direct to the optimization problem, the matrix method is generally used to describe the gradient descent algorithm.

Suppose the following linear regression equation:

\[ h_{\theta}(x_1, x_2, \ldots x_n) = \theta_0 + \theta_1 x_1 + \ldots \theta_n + x_n \]  

(3)

The matrix expression is:

\[ h(X) = X_\theta \]  

(4)

Among them, the shape of \( h_{\theta}(x) \) is \( m \times 1 \) and \( \theta \) is \( (n + 1) \times 1 \), there are \( n+1 \) algebraic methods for the model. The shape of \( X \) is \( m \times (n + 1) \), \( m \) and \( n \) represent the sample size and number of features per sample, respectively. The loss function is as follows:

\[ J(\theta) = \frac{1}{2} (X_\theta - Y)^T(X_\theta - Y) \]  

(5)

The shape of \( Y \) is \( m \times 1 \) from the output of the model, \( \theta \) is default parameters and \( \alpha \) represents the learning rate of the model.

The algorithm process is as follows:

1. Find the current gradient, take a derivative of \( \theta \), the following formula can be obtained

\[ \frac{\partial}{\partial \theta} J(\theta) = X^T(X_\theta - Y) \]  

(6)

2. The current gradient is multiplied by the learning rate \( \alpha \) to get the step size. The update expression of \( \theta \) is as follows:

\[ \theta = \theta - \alpha \frac{\partial}{\partial \theta} J(\theta) \]  

(7)

3. After performing multiple iterations, when the loss function reaches a certain minimum, we can assume that the model is convergent.

For the gradient descent method to solve the optimization problem, the step size is too short for the step size selection. The algorithm efficiency is lower. If the step size is set too long, the optimal solution may be missed, so the step size should be selected according to the actual situation [3]. For
initializing the parameters of the fitting function, the local optimal solution may appear after tuning with the gradient descent algorithm. The optimal solution can be easily obtained by many iterations, but in order to make the optimization effect better and to avoid the risk of local optimal solutions, different parameters can be initialized. For the gradient descent method, different gradient descent algorithms have been evolved by this algorithm. For example, given a sample set M, a random copy N is taken instead of the original sample M as complete set to train the model. Since this training is to extract part of the data, there is a greater chance that a local optimal solution is obtained. But one obvious benefit is that if the sample is extracted within the appropriate range, the result is obtained and the speed is fast.

2.2. Least Squares Algorithm Structure

The least squares method is a commonly used method in machine learning [4, 5]. The least squares method is often used in solving the function fitting or the method of finding the extremum of the function. Solving regression problems in many machine learning libraries for least squares math operations, the matrix method is often used to solve the analytical solution for function fitting. And the matrix method is simpler than the algebra method, and the matrix operation can replace the loop. Therefore, we use the matrix method to describe the operation method of the least squares method [6].

When solving multiple linear regression problems, such as functions:

\[
h_\theta(x_1, x_2, \ldots, x_n) = \theta_1 + \theta_2 x_1 + \ldots + \theta_{n+1} x_n
\]

The matrix expression is:

\[
h_\theta(X) = X_\theta
\]

Among them, the shape of \(h_\theta(x)\) is \(m \times 1\) and \(\theta\) is \(n \times 1\), there are \(n\) algebraic methods for the model. The shape of \(X\) is \(m \times n\), \(m\) and \(n\) represent the sample size and number of features per sample, respectively. The loss function is as follows:

\[
J(\theta) = \frac{1}{2} (X_\theta - Y)^T (X_\theta - Y)
\]

For the mathematical model expressed by the loss function, we need to derive the \(\theta\) vector and take 0. The result is as follows:

\[
\frac{\partial}{\partial \theta} J(\theta) = X^T (X_\theta - Y) = 0
\]

Here we apply two matrix operation chain derivation rules:

\[
\frac{\partial}{\partial \theta} (XX^T) = 2X
\]

\[
\frac{\partial}{\partial \theta} (X_\theta) = X^T
\]

The analytic formula for \(\theta\) can be obtained by arranging the formula after the \(\theta\) derivative of the loss function is zeroed:

\[
\theta = (XX^T)^{-1} X^T Y
\]

In this way, the solution of \(\theta\) can be calculated from this analytical formula, and the function can be obtained:

\[
h_\theta(x_1, x_2, \ldots, x_n) = \theta_1 + \theta_2 x_1 + \ldots + \theta_{n+1} x_n
\]

When solving the nonlinear regression problem, we can see that we set the multivariate linear function as the formula (1.8) puts the formula:
Bringing in the above function equation, we can get an expression about the unary nonlinear function:

\[ h_p(x, x_2, \ldots x^n) = \theta_1 + \theta_2 x + \ldots + \theta_n x^n \]  

(17)

For this function, is nonlinear for x, but for the parameter \( \theta \), the function is linear, then we can use the matrix method to solve the unary nonlinear function. The solution steps are as follows:

(1) Let the fitting function be:

\[ H_a(x) = a_1 + a_2 x + \ldots + a_n x^n \]  

(18)

(2) The sum of squared deviations, that is, the sum of the distances from each point to the curve:

\[ R^2 = \sum_{i=0}^{n} (y - H(x))^2 \]  

(19)

(3) In order to find the value that accords with \( a_n \), the following formula can be obtained by calculating the partial derivative of \( a_n \):

\[ -2 \sum_{i=0}^{n} (y - H(x))x^k = 0(k = 0, 1, \ldots n) \]  

(20)

(4) Simplify the left side of the equation and write it in matrix form:

\[
\begin{pmatrix}
\sum_{i=1}^{n} Y_i \\
\sum_{i=1}^{n} Y_i \\
\vdots \\
\sum_{i=1}^{n} Y_i 
\end{pmatrix}
= 
\begin{pmatrix}
\sum_{i=1}^{n} x_i \\
\sum_{i=1}^{n} x_i^2 \\
\vdots \\
\sum_{i=1}^{n} x_i^n 
\end{pmatrix}
\begin{pmatrix}
a_0 \\
a_1 \\
\vdots \\
a_n 
\end{pmatrix}
\]

(21)

(5) After simplifying Van dermond matrix, we can get:

\[
\begin{pmatrix}
1 & x_1 & \ldots & x_1^k \\
1 & x_2 & \ldots & x_2^k \\
\vdots & \vdots & \ddots & \vdots \\
1 & x_n & \ldots & x_n^k 
\end{pmatrix}
\begin{pmatrix}
a_0 \\
a_1 \\
\vdots \\
a_n 
\end{pmatrix}
= 
\begin{pmatrix}
y_0 \\
y_1 \\
\vdots \\
y_n 
\end{pmatrix}
\]

(22)

(6) That is \( XA=Y \). Solving this matrix equation yields the coefficient matrix A. At the same time, we obtain the analytical equation of the fitted curve.

3. Comparison of Two Algorithms under Deep Convolutional Neural Networks

We compare the different effects of the two optimization algorithms on the regression problem on the vgg16 convolutional neural network model. Vgg16 has been trained in 1000 different types of animal pictures. Therefore, the feature extraction layer of the network can extract and classify and judge the characteristics of different animals in 1000, but the VGG network is a convolutional neural network for classification tasks. We need to make VGG recognize the animal type and to predict animal length,
So we changed the fully connected layer of vgg16 for the classification task (such as the fully connected+ReLU and softmax layers on figure 1), and switched to the fully connected layer for animal length identification (the structure of the first layer of fully connected layer is: Input parameters extracted from the feature layer, the output is 256 parameters. The structure of the second layer fully connected layer is: the input is 256 parameters of the first layer full connection layer output, and the output is 1 parameters in order to predict animal length), so that it can be constructed a neural network for solving regression problems and identifying the length of an animal [7]. For this we get 2 thousand pictures of tigers and cats from ImageNet, but since the pictures on the data set do not contain the length of the corresponding animals, we give them lengths for cats and tigers, as shown in figure 2. The pink dot is the length given to the tiger, and the length ranges from 20 to 150 cm. Cats range in length from 20 to 80 cm. Reconstruct the vgg16 network parameters by using 2,000 pictures of cats and tigers. Since vgg16 has been trained in different categories of animals in 1000, we do not need to train his feature extraction layer and only to train our new fully connected layers to achieve the purpose of predicting cat and tiger length.

First, we use the least squares method to set the fitting function using the matrix method to fit the predictive function, where x represents the label of each image. When performing function optimization using the least squares method, we only performed the prediction of the length of the tiger, because predicting the length of the tiger can well represent the optimization ability of two different algorithms in the prediction problem. We can see from figure 2 that the length of the tiger is randomly assigned in the range of 20 to 150 cm. We use a 9th-order function for the least squares matrix operation to fit the length of the tiger. The more complex the curve is, the greater the order of the curve is, the more difficult the model calculation is, in order to compare the two optimization algorithms, we select the fitting curve of the ninth order. The fitting curve is shown in the green curve in figure 2. It can be seen that the optimization algorithm basically achieves the purpose of prediction. After solving, we can get the deviation of the mode, and the loss value obtained by the average.
squared error is 685.2, the results basically reach the predicted purpose.

When the optimizer is gradient descent, due to the irregularity of the data and the problems caused by the change of the network structure, the gradient descent optimization algorithm is used to generate the gradient explosion problem. When the training times are deepened, the gradient descent algorithm unable to make the correct prediction. So, we selected the RMSPropOptimizer optimizer to optimize the loss function. After several iterations, the tiger prediction curve is shown in figure 3. After solving, we can get the deviation of the model. The loss value obtained by the average squared error is 1223.6. In order to test the fitting effect of the RMSPropOptimizer optimization algorithm, we randomly select a tiger picture and a cat picture from 2,000 pictures for prediction. The prediction result is shown in figure 4. It can be seen that the method basically predicts the length of the animal within the acceptable error. Although the optimization using the RMSPropOptimizer optimizer does not produce a gradient explosion problem, the fitting curve shows a slight over-fitting problem and the loss value is larger than the loss value fitted by the least square method.

By on a small data set and a convolutional neural network model, we can initially get the difference between the two optimization algorithms. That is, if the sample size is not large and there is an analytical solution, the least squares method has an advantage over the gradient descent method, and the calculation speed is fast. If the amount is large, the least squares method requires a super large inverse matrix, which is difficult or slow to solve the analytical solution, the iterative gradient descent method has advantages. The least squares method can only solve multivariate linear problems and unary nonlinear problems. The gradient descent method can apply in many optimization scenarios. However, gradient descent algorithm is more prone to gradient explosion than other optimization algorithms. The gradient descent method is an iterative solution, and the least squares method is to calculate an analytical solution. If there is no analytical solution, the gradient solution can only be used to gradually approximate the optimal solution. When loss function is a convex function and the least squares method has an analytical solution, the gradient descent method after multiple iterations can obtain a global optimal solution similar to the least square method. If the loss function is not a convex function, there may be a local optimal solution obtained by the gradient descent method, and the analytical solution of the least square method must be a global optimal solution.

![Figure 3. RMSPropOptimizer algorithm results in predicting tiger length.](image-url)

**Figure 3. RMSPropOptimizer algorithm results in predicting tiger length.**

![Figure 4. Prediction results under the RMSPropOptimizer optimization algorithm (the left side of the graph is the cat’s prediction result, and the right side of the graph is the tiger’s prediction result).](image-url)

**Figure 4. Prediction results under the RMSPropOptimizer optimization algorithm (the left side of the graph is the cat’s prediction result, and the right side of the graph is the tiger’s prediction result).**
4. Conclusion
In this paper, different performances of gradient descent method and least squares algorithm in solving the function optimization problem of the regression problem are designed, and the fitting effects of the two different functions are compared in nonlinear cases on a significant number of data sets respectively. The effects of two different optimization functions are compared on the convolutional neural network, and the advantages and disadvantages of two algorithms in different situations are obtained. The basic optimization algorithm on the tensorflow platform is based on the gradient descent. It is obvious that the use of the gradient descent method is more common than the least squares algorithm. The least squares algorithm only appears in some machine learning libraries, and the model is applied in the form of a matrix to optimize, through this paper, we can understand that there are many similarities between the two different algorithms in solving the optimal problem, and they can also exert their respective advantages in solving different problems. Through this paper, we can better understand the advantages and disadvantages of various optimization algorithms, and select appropriate optimization methods for learning and training in future machine learning.

Acknowledgment
This work is supported by the National Natural Science Foundation of China (Grant No. U1701265).

References
[1] Hong X, Mitchell R and Di Fatta G 2019 Simplex basis function based sparse least squares support vector regression Neurocomputing 330 394-402.
[2] Cai J, Tang Y and Wang J 2016 Kernel canonical correlation analysis via gradient descent Neurocomputing 182 322-331.
[3] Dong X, and Zhou D X 2008 Learning gradients by a gradient descent algorithm Journal of Mathematical Analysis & Applications 341 (2) 1018-1027.
[4] Haykin S S and Gwynn R 2009 Neural Networks and Learning Machines (China Machine Press).
[5] Hu T, Fan J, Wu Q and Zhou D X 2015 Regularization schemes for minimum error entropy principle Analysis and Applications 13 (04) 437-455.
[6] Escandell-Montero P, Martinez-Martinez J M, Martin-Guerrero J D, Soria-Olivas E and Gomez-Sanchis J 2014 Least-squares temporal difference learning based on an extreme learning machine Neurocomputing 141 37-45.
[7] Huang Z, Siniscalchi S M and Lee C H 2016 A unified approach to transfer learning of deep neural networks with applications to speaker adaptation in automatic speech recognition Neurocomputing 218 448-459.