A Shallow Neural Network for Simple Image Recognition

Qianwen Yang, Wenxia Xu and Gaoquan Hu

ABSTRACT

Taking the fashion mnist data set proposed by zalando research team in 2017 as the object, this paper constructed a "shallow" neural network, whose test result accuracy reached 0.98, and then took the mnist data set as the test object to display the test results. when constructing the model, the loss function, activation function, gradient optimizer, overfitting and learning rate were explained, among which the gradient optimizer and overfitting had the greatest influence on the overall model accuracy.

1. INTRODUCTION

Nowadays, the application of neural network is more and more extensive, and with the deepening of learning, the improvement of algorithm in it leads to the constant change of the structure of artificial neural network (ANN), from the first two layers to three layers, four layers, to the present hundreds of layers, thousands of layers. Since a single neural network structure cannot handle different classes of objects, the types of neural networks began to change, and many neural network types evolved, such as later CNN, RNN, LRNN, RBF, HPFNN and so on. In the field of machine vision, CNN is widely used. When it is mentioned, face recognition and obstacle recognition of driverless cars come to mind. In addition to image identification, CNN is also used in fault prediction. With the emergence of Deep Learning and Reinforcement Learning, people have forgotten the simple neural network. Therefore, this paper constructs a "shallow" neural network model (SNN) for simple image recognition, and the data sets used for testing are Fashion MNIST[5] and MNIST.
For scholars who are new to the field of machine vision, these two data sets are the basic data sets in this field. If the model constructed has a low recognition rate for simple images, it means that the model built is very poor. The structure of SNN model is firstly tested by activation function, gradient optimizer, overfitting and learning rate, and then the parameters that make the best calculation results are selected to construct the SNN model. The framework used in model construction is Tensorflow, which is described in the second part, described in the third part and summarized in the fourth part.

2. MODEL ESTABLISHMENT

2.1 Active Function

Since the neural network used in the model construction in this paper is feedforward neural network, which is very familiar to all of us, this network will not be described here. The structure flow chart of the model constructed in this paper is shown in figure 1 below. The SNN structure proposed in this paper is a three layers neural network. In the figure, X represents the input, Y represents the output, W represents the weight, b represents the bias, Z represents the input of the hidden layer in the middle, and a is the activation function.

Including \( Z^i_j = W^i_j X^i_j + b^i_j \), \( a^1_j = g(Z^1_j) \), \( Y^n_j = W^n_j a^n_j + b^n_j \), i and n all represent the number of layers, j represents the number of elements. The SNN model structure is shown in Figure 1. The framework flow chart of the model constructed in this paper is shown in Figure 2. Sigmoid is the most widely used activation function, and the pattern has the characteristic of exponential function shape. Sigmoid neurons are nearly saturated when the value is 0 or 1, and the gradient in these regions is almost 0. Therefore, in the back propagation, the local gradient will be multiplied by the gradient of the entire cost function with respect to the output of the unit, and the result will be close to 0. In this way, almost no signal passes through the neuron to the weights and data, so the gradient makes no contribution to the updating of the model.

![Figure 1. The structure of SNN.](image-url)
The specific calculation formula is shown below.

\[ f(x) = \frac{1}{1+e^{-x}} \]  
(1)

Tanh function evolves from Sigmoid function. Compared with Sigmoid function, Tanh function is better than Sigmoid function in the hidden layer. Its advantage lies in that its value range is between -1 and 1, and the average value of data is 0, unlike Sigmoid which is 0.5. The formula is shown below.

\[ f(x) = \frac{1-e^{-2x}}{1+e^{-2x}} \]  
(2)

Although the Tanh function is symmetric about the origin center, it solves two problems of Sigmoid, but it also has the problems of gradient disappearance and power operation. So in order to solve the gradient disappearance problem, the Relu function appears. The derivative of the Relu function is much simpler than the first two activation functions. SGD with Relu converges much faster than Sigmoid and Tanh. Compared to Sigmoid and Tanh, Relu only needs a threshold to get the activation value, without having to do a lot of complicated operations. The formula is shown below.

\[ y = \begin{cases} 
  x & \text{if } x \geq 0 \\
  0 & \text{if } x < 0 
\end{cases} \]  
(3)

Leaky Relu function is developed on the basis of Relu function, because when the input value of Relu function is negative, the output value is always 0, and the first derivative is always 0, which causes the Neuron cannot update its parameters, that is, the Neuron does not learn, this phenomenon is called "Dead Neuron". To solve the shortcoming of Relu function, a Leaky value is introduced into the negative half interval of Relu function, so it is called Leaky Relu function.

Figure 2. The Framework flow chart of the model.

\[ \text{Active function} \]
\[ \text{GradientDescent} \]
\[ \text{Optimizer} \]
\[ \text{Overfitting} \]
\[ \text{Learning rate} \]
\[ \text{Iterative way} \]
\[ \text{Input} \]
\[ \text{Output} \]
\[ y_i = \begin{cases} x_i & \text{if } x_i \geq 0 \\ \frac{x_i}{a_i} & \text{if } x_i < 0 \end{cases} \] (4)

2.2 Gradient Descent Optimizer

When we talk about gradient optimizers, we need to understand what an optimizer is. The goal of deep learning is to continuously change the network parameters so that the parameters can fit the input output with various nonlinear transformations. In essence, it is to find the global optimal solution according to a function, so how to update the parameters is the focus of deep learning research. Generally, the algorithm of updating parameters is called optimizer, and the commonly used optimizer is gradient descent. Gradient descent is the most popular algorithm in ANN optimization at present. The understanding of gradient descent algorithm is that the direction of function gradient represents the direction with the fastest growth rate of function value, and the opposite direction can be regarded as the direction with the fastest decrease rate of function value.

For the optimization of machine learning model, when the goal is set to solve the minimum value of the objective function, the optimal value can be approached continuously as long as the direction of gradient descent is advanced. Different gradient optimizers may be selected to obtain different optimal values, some are local minimum, some are global minimum. According to the updated parameters of sample quantity, gradient descent is divided into three types of gradient descent algorithms: BGD (batch gradient descent), SGD[6] (stochastic gradient descent), and MBGD (minimum batch gradient update). Adaptive vector optimizer also appeared, including Adagrad[3], Adadelta[1], RMSProp, Adam optimizer[4,7], such as behind the experiment with GD, Adagrad, Adadelta, Adam gradient optimizer, and the training time, stability and test precision accuracy comparison, finally come to the conclusion.
2.3 Loss Function

And when we talk about gradient descent, have to talk about the loss function. In order to evaluate the model fitting, loss function is usually used to measure the fitting degree. Loss function minimization means that the fitting degree is the best, and the corresponding model parameters are the optimal parameters. In linear regression, the loss function is usually the squared difference between the sample output and the hypothesis function. For example, for the sample \((x_i, y_i)\) (\(i=1,2,...N\)), linear regression is adopted, and the loss function is shown as follows.

\[
J(\theta_0, \theta_1) = \sum_{i=1}^{m}(h_{\theta}(x_i) - y_i)^2 \tag{5}
\]

where, \(x_i\) represents the ith element of sample characteristic \(X\), \(y_i\) represents the ith element of sample output \(y\), and \(h_{\theta}(x_i)\) is the hypothesis function.

2.4 Over Fitting

In the process of training model, if we have many model parameters and too many training samples, the result may appear "perfect prediction", that is, the hypothesis result may match the training result very well, but the prediction result will be poor when new features appear. To solve this problem, there are methods to solve overfitting problems, such as regularization (\(L_1\) regularization and \(L_2\) regularization), Dropout, Early stopping, weight sharing, adding noise, and so on. Dropout [2] is mainly used in the full connection layer. During the forward propagation, the activation value of a neuron stops working at a certain probability \(P\) (in fact, deleting a neuron), which can make the model more generalized, because it will not rely too much on some local features.

Then describe the dropout neural network model, assuming that a neural network has \(N\) hidden layers. Set \(n \in \{1,...,N\}\) represents the number of hidden layers in the network, \(a^n\) represents the input vector in the hidden layer, and \(y^n\) represents the output vector \((y^{(0)} = x)\) in the hidden layer. \(W^{(n)}\) and \(b^{(n)}\) are weights and biases at the number of \(n\) layer.

\[
a_j^{(n+1)} = W_j^{(n+1)} y^n + b_j^{(n+1)} \tag{6}
\]

\[
y_j^{(n+1)} = g(a_j^{(n+1)}) \tag{7}
\]

where, \(f\) is the activation function, which can be Sigmoid, Tanh, Relu, etc. After adding Dropout, the formula changes to the following.

\[
a_j^{(n)} \sim Bernoulli(p), \tilde{y}^{(n)} = a^{(n)} \ast y^{(n)} \tag{8}
\]

\[
a_j^{(n+1)} = W_j^{(n+1)} \tilde{y}^{(n)} + b_j^{(n+1)}, y_j^{(n+1)} = g(a_j^{(n+1)}) \tag{9}
\]

3. EXPERIMENT

In this paper, MNIST and Fashion-MNIST [5] data sets are adopted. Here, a brief introduction is made to the Fashion-MNIST data set. The Fashion-MNIST
dataset replaces the MNIST dataset and is directly used for the benchmark machine learning algorithm, which is also a good method to learn CNN. In 2017, it was developed by Zalando Research, consisting of images of 70,000 fashion products from 10 categories, with the size of 28*28. Also divided into training set and test set, the training set has 60,000 images and the test set has 10,000 images. Among them, the Fashion-MNIST data set includes four files, including training sample labels (train-labels-idx1-ubyte), training sample images (train-images-idx1-ubyte), test sample labels (t10k-labels-idx1-ubyte) and test sample images (t10k-images-idx3-ubyte). These data files can be downloaded directly online. Fashion-MNIST dataset is divided into 10 categories, including T-shirts/top, trouser, pullover, dress, coat, sandals, T-shirt, sneaker, bag and ankle boots.

![Graph](image1.png)

**Figure 3.** The test accuracy of Sigmoid function.

![Graph](image2.png)

**Figure 4.** The test accuracy of Tanh function.

In the model construction of the experimental part, when selecting the optimal parameters, only one parameter is changed and the others remain the same. The optimal parameters are obtained through comparative analysis of the experimental results. Firstly, compare the accuracy of the gradient optimizer in the same
activation function, and then analyse the gradient optimizer that can maintain the highest accuracy in different activation functions, and finally select the optimal gradient optimizer.

As can be seen from Figure 3, in the activation function Sigmoid and the data set is Fashion-MNIST, the test accuracy obtained by the Adadelta optimizer is the lowest, the highest precision is only 0.52, and the initial accuracy is only 0.10, while the other optimizer accuracy exceeds 0.85 at the beginning. The above results are all obtained by taking epoch as the cycle unit and iterating 101 cycles. It can be seen from Figure 4 that the test accuracy graph obtained in Tanh of activation function and that obtained in Sigmoid basically have the same changes, and the optimizer corresponding to the highest accuracy is Adam, with the highest accuracy reaching 0.98. The precision curve of Adadelta optimizer is slightly different from that of Sigmoid at the beginning. In the Tanh function, the curve of Adadelta increases gradually and then gradually stabilizes.

Figure 5. The test accuracy of Relu function.

Figure 6. The test accuracy of Leaky Relu function.
As shown in Figure 5, among the test accuracy obtained by Relu function as activation function, the test accuracy obtained by Adadelta optimizer is the lowest, with the highest accuracy approaching 0.6, while that obtained by Adam optimizer is the highest, reaching 0.98. As can be seen in Figure 6, when the activation function is Leaky Relu, as the iteration period lengthens, the testing accuracy of GD gradually approaches that of Adam, and Adam's test accuracy is still the highest, at 0.98. The experiment only in Fashion-MNIST data concentration gradient of the optimizer compared to choose, can be found in the process in which activation function, Adam optimizer gains the highest precision, precision curve is stable, the contrast analysis was carried out on the activation function, at the same time in the MNIST test, and then select the highest accuracy activation function.

**TABLE I. COMPARE THE ACCURACY RESULTS OF ACTIVATION FUNCTIONS TESTED IN ADAMOPTIMIZER.**

| Active function | Parameter | Test accuracy |
|-----------------|-----------|---------------|
|                 |           | Fashion | MNIST  |
| Sigmoid         | 3layer,epoch=101,AdamOptimizer,batch_size=50 | 98.18% | 97.99% |
| Tanh            | 3layer,epoch=101,AdamOptimizer,batch_size=50 | 98.13% | 98.20% |
| Relu            | 3layer,epoch=101,AdamOptimizer,batch_size=50 | 98.56% | 98.46% |
| Leaky Relu      | 3layer,epoch=101,AdamOptimizer,batch_size=50 | 98.05% | 98.14% |

Among the test accuracy obtained by Relu function as activation function, the test accuracy obtained by Adadelta optimizer is the lowest, with the highest accuracy approaching 0.6, while that obtained by Adam optimizer is the highest, reaching 0.98.

The experiment only in Fashion MNIST data set to choose gradient of the optimizer, it can be found in the process in which activation function, Adam in the process in which activation function, Adam optimizer gains the highest precision, precision curve is stable, the contrast analysis was carried out on the activation function, at the same time in the MNIST test, and then select the highest accuracy activation function. As can be seen from the table, in the case that other parameters remain unchanged, the learning rate is used to update the iteration period, and the iteration period is 100 cycles. Among them, the activation function Relu has the highest accuracy, and the Relu activation function still has the highest accuracy when tested in the MNIST data set. Therefore, Relu is selected as the activation function in the model.

After adding the gradient optimizer, it involves the selection of learning rate. Generally, the optimal learning rate corresponding to different gradient optimizers is different. For example, Adam optimizer generally selects 0.01 for 1e-4 and Gradient Descent optimizer, but the so-called most learning rate may not be the best in fact, that is, the test accuracy obtained is not necessarily the highest, so automatic adjustment of the learning rate is added, 51 cycles are iterated, and the test accuracy
in the data set of Fashion-MNIST increases to 0.97. After completing the selection of activation function and optimizer, Dropout was selected to solve the over-fitting problem. After joining Dropout, the keep_prob value was selected between 0.5 and 0.9 in most cases. In this paper, keep_prob was set to 0.5 in training and 1.0 in testing.

4. CONCLUSIONS

The shallow neural network model (SNN) proposed in this paper is firstly constructed by taking the Fashion MNIST data set as the object, firstly the overall framework is built, and then the parameters are adjusted, which is the so-called parameter adjustment. At the same time, it is found that the overall accuracy does not change much after adding a hidden layer, so SNN still adopts a 3-layer structure. Then, the accuracy reached 0.98 in the MNIST data set. Meanwhile, compared with CNN with two convolutional layers, it is found that the test accuracy of SNN is higher than that of CNN.

Meanwhile, compared with MNIST, the image identification in Fashion MNIST is still difficult. For the purpose of this paper, the SNN model for image recognition cannot reach 100%, and then it will be adjusted to strive for the highest accuracy.

REFERENCES

1. Zeiler M D. 2012. Adadelta: An Adaptive Learning Rate Method. Computer Science, 2012.
2. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 2014, 15(1):1929-1958.
3. Ward R, Wu X, Bottou L. 2018. AdaGrad stepsizes: Sharp convergence over nonconvex landscapes, from any initialization. 2018.
4. Kingma D P, Ba J. 2017. Adam: A Method for Stochastic Optimization. Computer Science, 2014.
5. Xiao H, Rasul K, Vollgraf R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.
6. Ioffe S, Szegedy C. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. International Conference on Machine Learning. JMLR.org, 2015.
7. Guangxin Cui, Diankui Li. 2018. Research on Handwritten Digit Recognition Based on Adam Optimizer Self-encoding. Journal of Jiamusi University (Natural Science Edition), 2018, 154(03):35-38