Multi-class Classification on Image Data by using Multilayer Perceptron and Stochastic Gradient Descent

Zhengyuan Li
Li.zhengyuan@mail.mcgill.ca
Tianhao Shan
tianhao.shan@mail.mcgill.ca
Jiachen Huo
jiachen.huo@mail.mcgill.ca

Abstract

Similar to test classification, image classification is an important task in machine learning. In this project, we will use an image classifier by implementing a multi-layer perceptron (MLP) together with backpropagation and mini-batch gradient descent (SDG). To run the experiment on the MNIST[1] dataset which is a widely used image dataset that is pre-partitioned into testing and training set. We found that if using 10000 training data fitting the model, the performance of layers is 2 layers > 1 layers > 0 layer; the performance of activation function is ReLU>tanh>sigmoid. Under 2 layers with 128 hidden units each, we obtained the result of 93.4% accuracy when using ReLU, 93.04% when using tanh, and 85.53% when using sigmoid as active function.

1 Introduction

To make the model more diversified, we implemented three MLP models as no hidden layer, one hidden layer and two hidden layers respectively, together with different choices of activation functions (ReLU, tanh, sigmoid) to increase the non-linearity of the network. The dataset that we worked on is a large collection of hand-written digits that has been size-normalized and centered in 28x28 fixed-sized image pixels, together with its corresponding true labels in integer format[2]. As for the choice of selecting MLP models and activation functions, we implemented and ran a total of 7 cases, of which experiment 1 to 6 are trained with normalized images. Below is a breakdown of the 7 models:

1. No hidden layer.
2. One hidden layer with ReLU.
3. Two hidden layers each with ReLU.
4. Two hidden layers each with tanh.
5. Two hidden layers each with sigmoid.
6. Two hidden layers each with ReLU and add additional L2 regularization.
7. Two hidden layers each with ReLU, with unnormalized image.

Throughout the experiment, we found that ReLU activation function generally performs the best, it also takes fewer time compared to tanh and sigmoid. Also, accuracy is positively related to the number of hidden layers as we discovered in our project, however, since with no hidden layer we already obtained a high accuracy as 90%, adding more layers than two would not be that significant, compared to other improvements such as data augmentation and optimization approaches.

1.1 Related work

A research paper published in October 2020 (Sadra Shadkani, 2020)[4] conducted a similar comparative study of multiplayer perceptron gradient descent for predicting daily suspended sediment load in Mississippi River, U.S. In the study, three machine learning models—multi-layer perceptron (MLP), multi-layer perceptron-stochastic gradient descent (MLP-SGD), and gradient boosted tree (GBT)—were utilized to estimate the suspended sediment load at the St. Louis and Chester stations on the Mississippi River, U.S. Coincidentally, the study also used sigmoid function for function activation and back propagation as the training method of the multilayer perception network. The result of the study found that MLP models were improved by SGD optimization. Therefore, the MLP-SGD method is recommended as the most accurate model for SSL estimation.

2 Dataset

2.1 MNIST observation

he MNIST database (Modified National Institute of Standards and Technology database[1]) is a large database of handwritten digits that is commonly used for training various image processing systems. The pre-partitioned MNIST dataset includes 60000 training data and 10000 testing data, with each category (0~9) generally equally distributed with the frequency approximately ranges from 9% to 11% out of total. The following are two pie charts indicate the distribution of training and testing sets respectively.

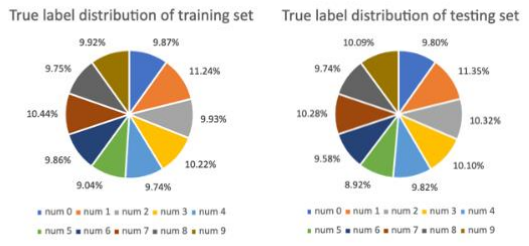


Fig 1. True label distribution of training and testing

2.2 Data Processing

After loading the data directly into training and testing set from TensorFlow[3]. package, we normalized the image to rescale the pixel range from [0, 255] to [0.0, 1.0] by dividing by 255. We also vectorized the original 28x28 matrix to a vector of length 748 for the convenience of further model fit and optimization.

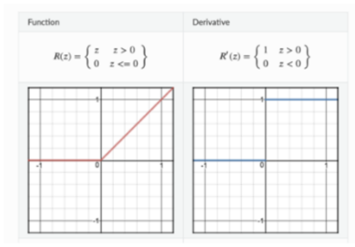
3 Models

A Multi-layer Perceptron (MLP) typically has three layers: the input layer, the optional hidden layers and the output layer. The structure of our model consists of 2 fully connected hidden layers, i.e. [128,128] units in each hidden layer. We use ReLU/tanh/sigmoid as the activation function followed by each hidden layer. The final layer is a 10-way SoftMax which produces a probability distribution over the 10 class labels.

We define the modeling process as how the input information from the input layer is processed by the hidden layer, yielding the output which is the obtained at the output layer.

As mentioned in the Introduction of this report, we selected 7 combinations of MLP models and activation functions, thus obtained 7 kinds of model evaluations. A more detailed explanation of each model is described below:

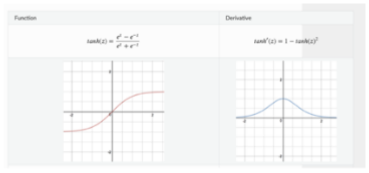
1. An MLP with no hidden layer. The inputs are directly mapped from input layer to output layer, not functioned by any hidden layer at the middle.
2. An MLP with a single hidden layer having 128 units and ReLU activations. Activation function is the type of function that transforms its inputs into outputs that have a certain range. These functions introduce nonlinear real-world properties to artificial neural networks. ReLU (Rectified Linear Unit) is one of the activation functions that has the following graph:



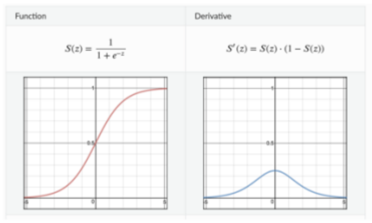
ReLU Activation Function and it's derivative

The function returns 0 if it receives any negative input. For any positive value x, it returns that value back. Thus, it gives an output that has a range from 0 to infinity.

3. An MLP with two hidden layers having 128 units and ReLU activations.
4. Two hidden layers each with tanh as activation function. The definition of tanh activation function is given below:



5. Two hidden layers each with sigmoid. Sigmoid function has the below graph:



Sigmoid function and it's derivative

6. Two hidden layers each with ReLU and add additional L2 regularization. Adding L2 regularization allow us to avoid overfitting to a specific training set by tuning the hyperparameter lambda, it follows the formula:

$$J(w)=L(y, yh) + \frac{\lambda}{2}||w||2$$

7. Two hidden layers each with ReLU, with unnormalized image.

4 Experiment

4.1 Initialization of weights

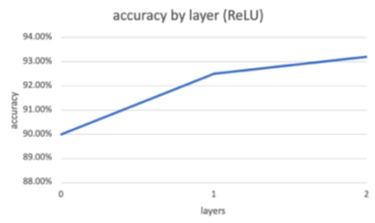
We initialize the weights of each layer by using *np. random. randn ()* times an initial factor 0.1. Throughout research and experiments we found that the combination of initial factor 0.1 and learning rate 0.09 will result in a high accuracy. Thus, in the following experiment we fix these two hyperparameters, unless tuning the learning rates.

4.2 Hyperparameters

We trained our model with different hyperparameters: the number of hidden layers, the number of units in the hidden layers, learning rate, the number of Gradient Descent iterations, activation function.

4.2.1 Compare the number of hidden layers

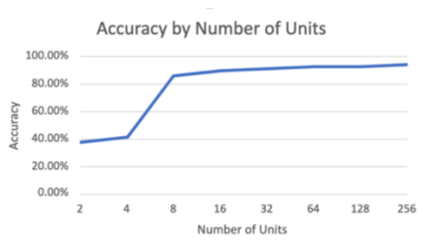
Here we fix 128 units, ReLU function, 1000 iterations, with 10000 train data, and we found 2 hidden layers resulted in the best accuracy 93.21%



From the experiment results above, we can conclude that more hidden layers will improve the accuracy. Since no hidden layer can only represent a linearly separable function. And non-linearity increases as the number of hidden layers increases.

4.2.2 Compare the number of units per hidden layer

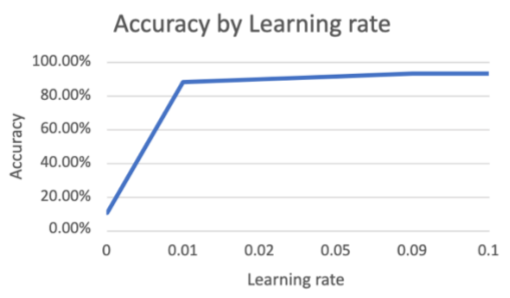
Here we fix 2 layers, ReLU function, 1000 iterations, with 10000 train data, and we found 256 hidden units resulted in highest accuracy



From the experiment results above, we can conclude that more units in hidden layers will improve the accuracy. However, too few hidden units will get high training error and high generalization error due to underfitting and high statistical bias; too many hidden units would cause low training error but still have high generalization error due to overfitting and high variance.

4.2.3 compare learning rates

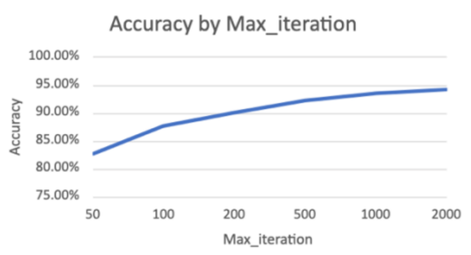
Here we fix 2 layers, ReLU function, 1000 iterations, with 10000 train data, tuning a list of learning rates [0.0, 0.01, 0.02, 0.05, 0.09, 0.1], we found with learning rate 0.1 would result in the highest accuracy of 93.82%



From the experiment results above, we can conclude that the model with higher learning rate Gradient Descent has higher accuracy. The learning rate affects how quickly our model can converge to a local minimum (i.e. arrive at the best accuracy). But the learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck.

4.2.4 compare the max number of iterations

Here we fix 128 units, 2 layers, ReLU function



From the experiment results above, we can conclude that the model with Gradient Descent of larger number of maximum iterations has higher accuracy. Since the optimization technique Gradient Descent is used in backpropagation while fitting the model, which will reduce the training error.

4.2.5 Compare activation function

128 units, 2 hidden layers, full train data

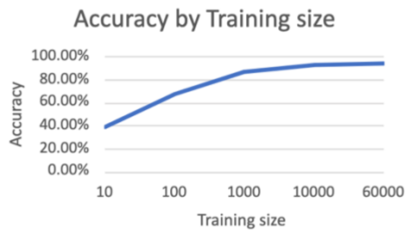
	Accuracy
ReLU	94.2%
tanh	94.04%
sigmoid	87.53%

Sigmoid squish the input into a [0,1] range thus output would be all positive; tanh has output in the range [-1,0] and is generally better than sigmoid since tanh is centered at the origin. Whereas ReLU only activate a specific unit if its linear transformation is positive (0 if negative). Thus, this selection method on activating the unit would perform better than tanh and sigmoid.

The biggest advantage of ReLU is that it does not restrict its gradient in certain directions, which greatly accelerates the convergence of Gradient Descent compared to the sigmoid / tanh functions, therefore the MLP model using ReLU will achieve higher accuracy.

4.3 Comparison of training size

To test the effect of training size on model accuracy, here we fix our model to 128 units, 2 layers, ReLU function and 1000 iterations, we found that full training data resulted in the highest accuracy of 94.2% accuracy.



From the experiment results above, we can conclude that the model with larger training dataset performs better. Since lower train data size might result in model not giving enough accuracy on newer data.

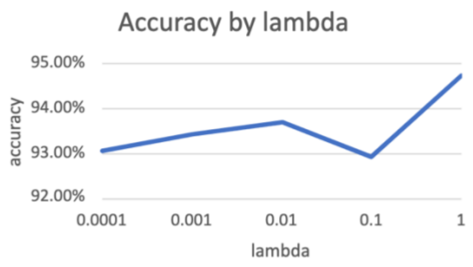
5 Overfitting

During the experiments, one of the most common problems while training MLP is overfitting. Overfitting occurs when a model tries to predict a trend in data that is too noisy. This is due to an overly complex model with too many parameters. A model that is overfitted is inaccurate because the trend does not reflect the reality present in real data. And we try to use regularization to prevent overfitting while training the model, which is a technique to reduce the complexity of the model. It does so by adding a penalty term to the loss function

1) L2 regularization: It reduces overfitting by modifying the cost function. The L2 penalty aims to minimize the squared magnitude of the weights. This is mathematically shown in the below formula.

$$\frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L}\|w^{[L]}\|_F^2$$

   And L2 regularization can learn complex data patterns and not robust to outliers. Here we fix 128 units, ReLU function,2 layers, and 1000 iterations, tuning a list of λ values [0.0001, 0.001, 0.01, 0.1, 1], we found that λ=1 has highest accuracy of 94.76%



2) Dropouts: It reduces overfitting by randomly dropping nodes from the multilayer perceptron during training in each iteration. And it is equivalent to training different networks when we drop different sets of nodes. And these different networks would have distinct overfit, so the net effect of dropout will be reduced.

6 Conclusion

We trained Multilayer Perception on the MNIST dataset for the handwritten digits' classification task in this project. After tuning the hyperparameters and optimizer, our model achieves highest the test accuracy of 94.2% by using ReLU with 2 layers. However, the unnormalized data showed a low accuracy of 9.8%, the performance could be improved better in several ways.

7 Further Improvements

Generally, improvements to neural networks include altering the ratios of training and testing datasets, the number of hidden nodes, and the training iterations. In our project, one possible improvement can be to modify the tanh function. According to a research paper by Krzysztof Halawa. To reduce the number of numerical calculations, the hyperbolic tangent can be replaced by the bipolar function $f(x) = \frac{2}{1+\exp(-2x)} - 1$ or by the function $f(x) = \frac{1}{1+\exp(-2x)}$.

Many microcontrollers have a low processing capacity. Look-up tables with values necessary for interpolation of the activation function or piecewise polynomial models may be used in devices under the control of such microcontrollers.

And the model could also be improved by choosing different optimizer (Gradient descent optimizers and Adaptive optimizers). Optimizers in machine learning are used to tune the parameters of a neural network to minimize the cost function. And Adam will perform better than the original Gradient Descent since it is more likely to return good results without an advanced fine tuning.

8 Contribution of Works

All authors together build the report.
Zhengyuan li: Research on the topic and analyzed the result; Tianhao shan: Data processing and run the experiment; Jiachen Huo: Implemented the model and run the experiment.

Reference

1. MNIST dataset from TensorFlow. (n.d.). Image Classification Dataset from TensorFlow. https://www.tensorflow.org/datasets/catalog/mnist

2. Dataset description. (n.d.). The MNIST Database of handwritten digits. http://yann.lecun.com/exdb/mnist/

3. Loading dataset. (n.d.). MNIST Classification Tutorial (Including Tutorial on Loading Data). https://www.tensorflow.org/quantum/tutorials/mnist

4. Comparative study of multilayer perceptron-stochastic gradient descent and gradient boosted trees for predicting daily suspended sediment load: The case study of the Mississippi River, U.S. (2020, October 14). *MLP and Gradient Descent*, 1. https://www.sciencedirect.com/science/article/abs/pii/S1001627920300986