UNIVERSITÀ
DI SIENA
1240

Comprehensive Study on MNIST Character Recognition with 1 Hidden Layer

Machine Learning
Handwritten char recognition

A.A. 2022/2023

Duccio Meconcelli

Abstract

This report investigates the impact of various hyperparameters on the performance of a neural network model trained on the MNIST dataset for handwritten digit recognition. We conduct experiments comparing the performance of the model in terms of accuracy and training time.

Through a grid search, we identify the optimal hyperparameters for the model, achieving an accuracy of 98% on the test set with a single hidden layer network. This study demonstrates the importance of selecting appropriate hyperparameters, particularly batch size, the loss type and the weights initialization, in achieving high accuracy and computational efficiency in training neural network models for handwritten digit recognition.

Contents

1 Introduction

Handwritten digit recognition is a classic problem in machine learning that has been widely studied over the past few decades. The MNIST dataset, containing 60,000 training images and 10,000 test images of handwritten digits, has become a benchmark dataset for this problem.

The goal of our research was the following:
Based on the Python scripts and on the MNIST data made available write a program for handwritten char recognition whose objectives are the following:

1. *Test the performances by using the Quadratic and the Entropy Loss;*
2. *Test the performance and discuss the efficiency of the following learning mode protocols:*
   (a) *batch mode;*
   (b) *on-line mode;*
   (c) *mini-batch mode;*
3. *Discuss the role of the weight initialization for both the Quadratic and Entropy loss.*
4. *Finally, we added an additional experiment to investigate the difference between binary encoding and one-hot encoding of the input data.*

The objective of this report is to study the impact of hyperparameters on the performance of a neural network model trained on the MNIST dataset.

Hyperparameters are variables that determine the behavior and performance of the model during training. They include the learning rate, batch size, number of epochs, number of layers, and many others.

The choice of hyperparameters can have a significant impact on the accuracy and training time of the model. Therefore, it is crucial to understand how different hyperparameters affect the model's performance and choose the optimal hyperparameters for the specific problem at hand.

In addition we also focus on the time required to train the model with different parameters. Training a neural network model on large datasets like MNIST can be computationally expensive and time-consuming, particularly when using deep architectures or complex models.

By analyzing both the accuracy and time metrics, we aim to provide a comprehensive evaluation of the performance of the model.

2 Methods

In this study, we trained a neural network model with one hidden layer (Figure 1) on the MNIST dataset for handwritten digit recognition. The input images had a size of 28x28 pixels, and we used the backpropagation algorithm to update the weights during training. We implemented the backpropagation algorithm from scratch, and we wrote the code with the goal of minimizing the use of external libraries. The code is available on the GitHub Repo [1]. To find the best parameters we used the Grid Search.

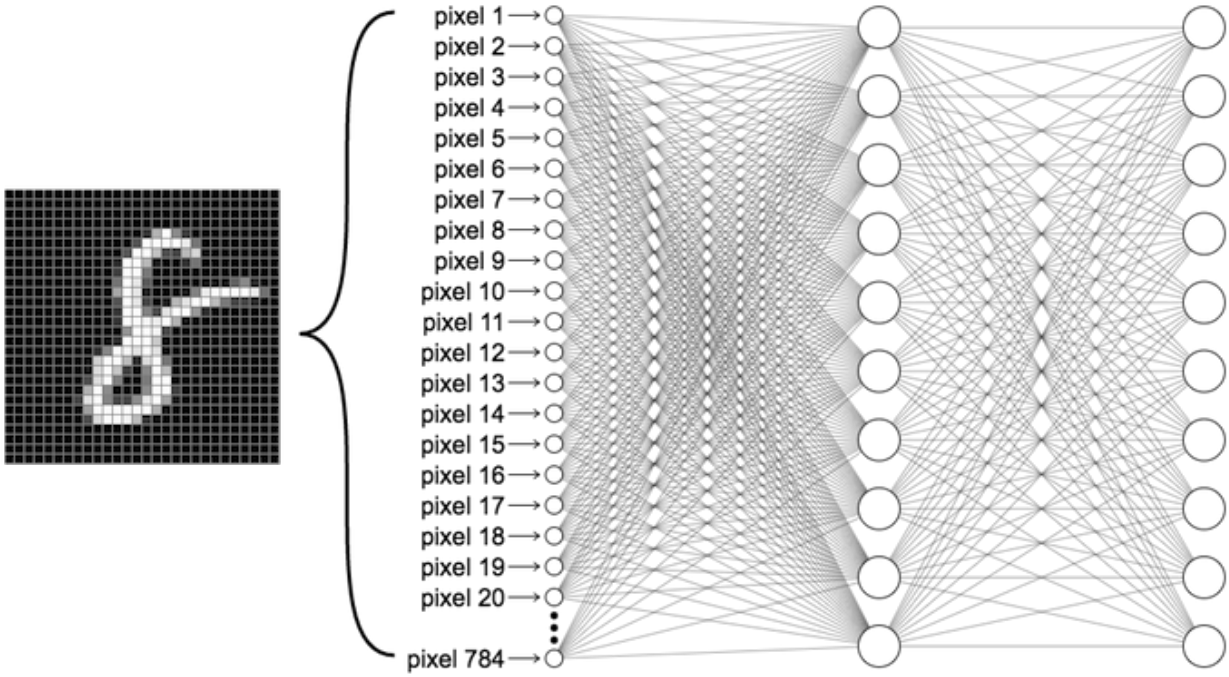


Figure 1: Model Architecture, image by [2]

2.1 Grid Search

The grid search allowed us to explore the impact of each hyperparameter on the performance of the model systematically. The grid search is a technique used in machine learning to determine the best combination of hyperparameters for a model. It involves defining a grid of all possible hyperparameter combinations and then training and evaluating a model for each combination. The final result is the combination of hyperparameters that provides the best performance on a given metric, such as accuracy.

The main benefit of using a grid search is that it helps to automate the process of finding the best hyperparameters for a model. Without a grid search, it can be time-consuming and challenging to manually test various combinations of hyperparameters to find the best one. Additionally, using a grid search helps to ensure that the model is not overfitting to the training data, as it evaluates the performance of the model on a separate validation dataset.

2.2 Hyperparameters

To explore the impact of different hyperparameters on the performance of the model, we conducted experiments varying the following hyperparameters:

- Batch size: mini-batch, online mode (batch size of 1), and full batch (batch size equal to the number of training samples)
- Number of epochs
- Learning rate
- Number of neurons in the hidden layer
- Type of data encoding
- Weight initialization: random standard initialization, Xavier initialization, or zero initialization
- Type of loss function: cross-entropy or quadratic loss
- Type of activation function for the hidden layer: Sigmoid or ReLU

2.3 Encoding Data

We tested two different encoding methods: binary encoding and one-hot encoding.

- One-hot encoding: This is a method of representing categorical data as a binary vector, where only one element of the vector is 1, and all others are 0. The index of the 1 corresponds to the category label. For example, the digit 3 can be represented as [0,0,0,1,0,0,0,0,0,0] using one-hot encoding.
- Binary encoding: This is a method of representing categorical data as a binary vector, where each category is represented by a unique binary pattern. For example, the digits 0-9 can be represented using 4-bit binary numbers, as follows: 0 = 0000, 1 = 0001, 2 = 0010, ..., 9 = 1001.

2.4 More on the Weights initialization

Weight initialization is a crucial step in training neural networks, as it can affect the model's performance and convergence rate. One common method of weight initialization is the Xavier initialization, named after its creator, Xavier Glorot.

Xavier initialization is a weight initialization method that sets the initial values of the weights using a

Gaussian distribution with mean 0 and variance 2/(number of input neurons + number of output neurons). The Xavier initialization can be written as:

$$\mathrm{W}_{i,j} \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_{in} + n_{out}}}\right) \tag{1}$$

The idea behind the Xavier initialization is to set the initial weights of the neural network in a way that balances the variances of the input and output of each neuron. This can help prevent vanishing or exploding gradients, which can hinder the learning process.

2.5 Losses

Quadratic loss: This is a loss function used in regression problems. It is defined as the sum of the squared differences between the predicted and actual values. The quadratic loss can be written as:

$$L = \frac{1}{2N} \sum_{i=1}^{N} \left(\hat{y}_i - y_i\right)^2 \tag{2}$$

where N is the number of examples in the dataset, y_i is the true label for example i, and $\hat{y_i}$ is the predicted label for example i.

The cross-entropy loss function is defined as:

$$L = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y_i}) + (1 - y_i) \log(1 - \hat{y_i})]$$

where N is the number of examples in the dataset, y_i is the true label for example i, and $\hat{y_i}$ is the predicted label for example i.

2.6 Evaluate the model

To evaluate the performance of the model, we used accuracy as the evaluation metric. Accuracy is defined as the ratio of the number of correct predictions to the total number of predictions made by the model. It is calculated as:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \tag{3}$$

In addition to accuracy, another important metric for comparing the different parameter settings was the training time. The training time refers to the amount of time required to train the neural network on

the training data. This is an important consideration since faster training times can allow for more efficient experimentation with different hyperparameters and can also be critical in real-world applications where speed is a primary concern. Additionally, longer training times can lead to overfitting, where the model performs well on the training data but poorly on the test data. Therefore, finding a balance between accuracy and training time is crucial in selecting the optimal hyperparameters for the neural network.

3 Results

The results obtained from the grid search highlight the impact of hyperparameters on the performance of the neural network model for the MNIST dataset. The complete list of all grid search results is available in the GitHub Repo.

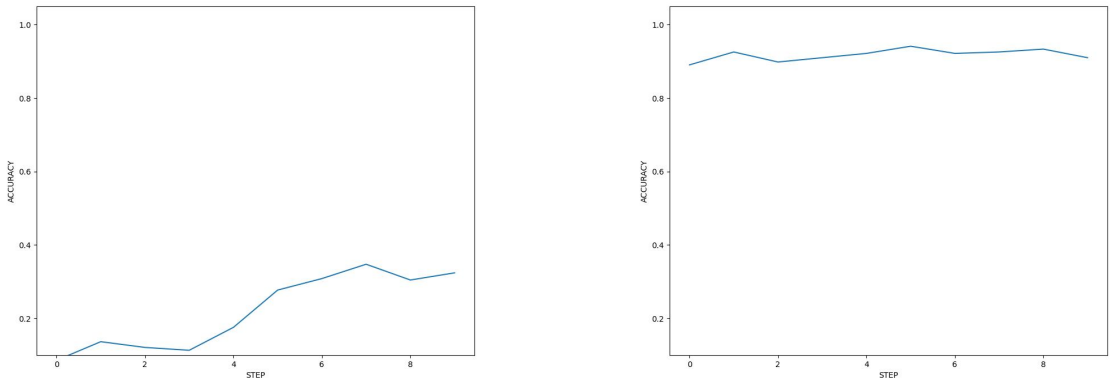
The comparison of quadratic and cross entropy loss functions showed that cross entropy performs better in recognizing the MNIST characters (Table 1). This is because cross entropy is more sensitive to misclassifications, making it a better fit for classification tasks. In fact, the cross-entropy loss considers the predicted probabilities for each class and their distance from the actual label, whereas the quadratic loss only looks at the difference between the predicted and actual values. In the case of the MNIST dataset, where the classes are well separated and the goal is to assign a probability to each digit, the cross-entropy loss is better suited to this task. It also tends to converge faster than the quadratic loss, as it provides more informative feedback to the model during training.

Epoch	Neurons	L Rate	Loss	Encoding	Batch	W Init	Acc	Time(s)
10	50	0.2	cross-entropy	one_hot	256	Xavier	0.93	7.93
10	50	0.2	quadratic	one_hot	256	Xavier	0.89	14.12
10	50	0.2	cross-entropy	one_hot	256	random	0.88	7.29
10	50	0.2	quadratic	one_hot	256	random	0.49	7.73
10	50	0.2	cross-entropy	one_hot	256	0	0.33	7.07
10	50	0.2	quadratic	one_hot	256	0	0.11	7.24

Table 1: Differences between Weight initialization with different Loss. With the other parameters being equal, greater accuracy is obtained with Xavier. Likewise greater accuracy is obtained with cross-entropy loss

Regarding the batch size, the results showed that online mode provides higher accuracy but at the cost of longer training times (Table 3). Mini batch provides a good balance between accuracy and efficiency, making it the preferred choice for training the MNIST dataset. For the size of the mini batch, 32 and 256 elements were used and in a similar way to how it happens in the online mode, the smaller the batch, the more we obtain greater accuracy by extending the training time.

The initialization of the weights was found to have a significant impact on the model's performance. The results showed that the Xavier initialization method leads to better accuracy (Table 1) and faster convergence (Figure 2) compared to random standard initialization or null values initialization. This is due to the fact that the Xavier initialization balances the variances of the input and output of each neuron, providing a better starting point for optimization. Compared to standard random weight initialization, the Xavier initialization can lead to faster convergence and better overall performance of the model. In summary, the Xavier initialization is a widely used technique for weight initialization in neural networks that can improve the performance and convergence rate of the model compared to random standard initialization.



(a) zero initialization **(b)** Xavier initialization

Figure 2: Differences between Weight initialization convergence speed

The comparison between one-hot encoding and binary encoding showed that there is no significant difference in performance, although theoretically binary encoding should add an extra level of complexity. However, this difference was not reflected in the results obtained from the experiments (Table 2).

The variation in the number of hidden neurons between 100 and 50 did not make a significant difference in terms of accuracy, and therefore for efficiency reasons, it is better to use 50 neurons. It is worth noting that the MNIST problem is relatively easy compared to other real-world problems, and in more complex tasks, the number of hidden neurons could play a more significant role.

The difference between 10 and 20 epochs had a slight impact on the overall accuracy of the results, but it was relatively negligible for small batch sizes. However, as the batch size increased, it became essential to train for more epochs to achieve optimal results.

Regarding the activation function, the difference between using Relu and Sigmoid was negligible; the small difference in performance between the two activation functions can be attributed to the simplicity of the problem and the nature of the input data where the input values are normalized between 0 and 1.

Finally, the difference between 0.1 and 0.2 as the learning rate was negligible and did not lead to significant changes in the results.

In conclusion, the grid search results demonstrate the importance of hyperparameter tuning in achieving optimal performance for neural network models. The cross entropy loss function, mini batch size (32), and Xavier weight initialization were found to be the most effective choices for the MNIST dataset.

Epoch	Neurons	L Rate	Loss	Encoding	Batch	W Init	Activation	Acc	Time(s)
20	100	0.2	quadratic	one_hot	32	Xavier	ReLu	0.98	45.13
20	100	0.2	quadratic	binary	32	Xavier	ReLu	0.98	77.79
20	100	0.1	quadratic	binary	32	Xavier	ReLu	0.97	39.61
20	100	0.1	quadratic	one_hot	32	Xavier	ReLu	0.97	61.58
20	100	0.2	quadratic	one_hot	32	random	ReLu	0.54	44.93
20	100	0.2	quadratic	binary	32	random	ReLu	0.48	74.54
20	100	0.1	quadratic	binary	32	random	ReLu	0.33	49.67
20	100	0.1	quadratic	one_hot	32	random	ReLu	0.2	46.87

Table 2: Differences between encoding

N	Loss	Encoding	Batch Size	Weight Init	Activation	Accuracy	Time (s)
601	cross-entropy	binary	1	Xavier	Sigmoid	0.98	663.18
389	cross-entropy	one_hot	32	Xavier	ReLu	0.98	44.23
413	cross-entropy	binary	32	Xavier	ReLu	0.98	23.64
577	cross-entropy	one_hot	1	Xavier	Sigmoid	0.97	673.19
629	quadratic	one_hot	1	Xavier	ReLu	0.97	572.46
625	quadratic	one_hot	1	Xavier	Sigmoid	0.97	585.1
585	cross-entropy	one_hot	1	random	Sigmoid	0.97	663.1
609	cross-entropy	binary	1	random	Sigmoid	0.97	653.74
437	quadratic	one_hot	32	Xavier	ReLu	0.96	30.79
409	cross-entropy	binary	32	Xavier	Sigmoid	0.96	28.34
385	cross-entropy	one_hot	32	Xavier	Sigmoid	0.96	32.92
461	quadratic	binary	32	Xavier	ReLu	0.96	23.72
5	cross-entropy	one_hot	256	Xavier	ReLu	0.96	11.24
29	cross-entropy	binary	256	Xavier	ReLu	0.96	8.58
633	quadratic	one_hot	1	random	Sigmoid	0.95	589.69
397	cross-entropy	one_hot	32	random	ReLu	0.94	32.12
421	cross-entropy	binary	32	random	ReLu	0.94	30.88
393	cross-entropy	one_hot	32	random	Sigmoid	0.93	27.98
417	cross-entropy	binary	32	random	Sigmoid	0.93	31.57
433	quadratic	one_hot	32	Xavier	Sigmoid	0.92	31.34

Table 3: Top 20 Accuracy parameter with 100 hidden neuron and 10 epochs

References

[1] Duccioo/handwritten-char-recognition: Assignment for fundamentals of machine learning exam 2023. https://github.com/Duccioo/Handwritten-Char-Recognition.

[2] How to build your own neural net from the scratch | zitao's web. https://zitaoshen.rbind.io/project/machine_learning/how-to-build-your-own-neural-net-from-the-scrach/.