Vietnam National University, Ho Chi Minh City

Ho Chi Minh City University of Technology

Project Assignment HAND-WRITTEN DIGIT RECOGNITION *(**Deadline:** December 3rd, 2022)*

Thinh Nguyen

October 2, 2022

Chapter 1

Motivation

Some classification problems are not binary (e.g. yes/no, dog/cat, 0/1, . . .). As an example, let's consider the hand-written digit classification problem. Given a picture of a hand-written digit ranging from 0 to 9, what is the digit appearing in the picture?

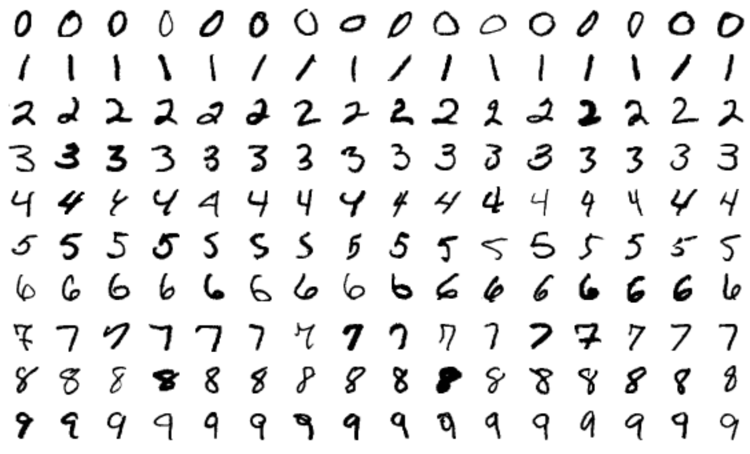

Figure 1.1: MNIST Dataset (en.wikipedia.org/wiki/MNIST_database)

Logistic Regression cannot be used for this problem due to the "sigmoid" "activation function" in the last stage of the algorithm. The idea is to replace the sigmoid function by the "softmax" function to enable the possibility of predicting more than two classes.

To understand those algorithms, we first consider the input images, each of them has the resolution 28×28 pixels, as vectors of 784 values ranging from 0 to 255 (see Fig. 1). Those "feature" vectors, which are vectors $\mathbf{x} \in \mathbb{R}^{784}$ (784-dimensional real vector space), are then mapped to the smaller vector space $\mathbb{R}^{10}$ (10-dimensional real vector space) representing the 10 classes of digits by means of the linear mapping function $f(\mathbf{x}) = \mathbf{W}^T\mathbf{x} + \mathbf{b} = \mathbf{y}$, where $\mathbf{W}$ is a 784×10 matrix, $\mathbf{b} \in \mathbb{R}^{10}$ is the intercept, $\mathbf{y} \in \mathbb{R}^{10}$ is the resulting vector, and T denotes the transpose operator. However, to determine the digit class of a picture, one needs to assign a probability to each class. It can be done with the softmax function g defined as follows.

$$[g(\mathbf{y})]_i = \frac{\exp(\mathbf{y}_i)}{\sum_{i=1}^{10} \exp(\mathbf{y}_i)}, \quad i = 1, \ldots, 10. \tag{1.1}$$

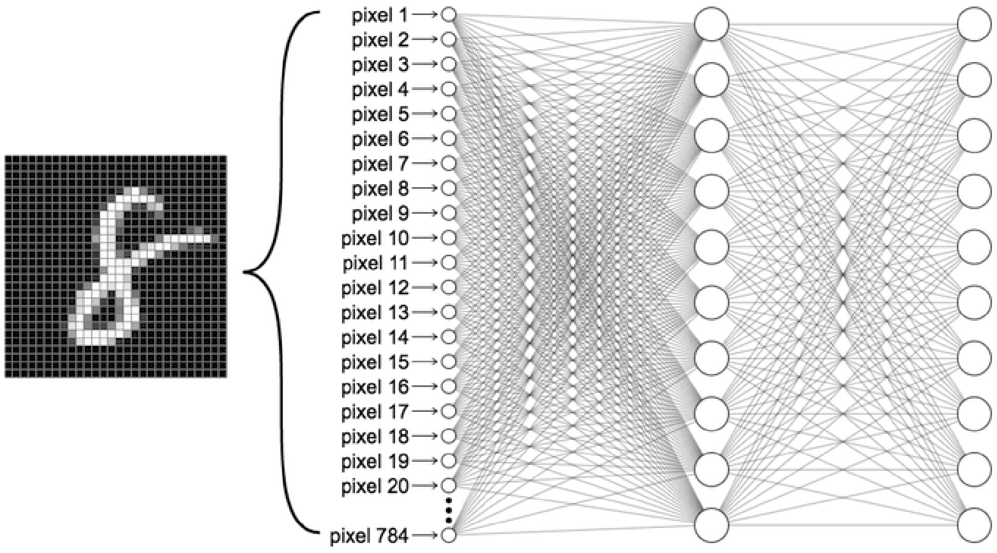

Figure 1.2: Fully connected layers (andreaprovino.it/tensorflow-mnist-tutorial-semplice-tensorflowlayers

The softmax function indeed shows the probability of each class. The predicted class is the index i if $[g(\mathbf{y})]_i$ is the maximum component of $g(\mathbf{y})$. The predicted class is not necessarily correct. This fact is dependent on the "weight" matrix $\mathbf{W}$ that we use to compute the probability of each class. Thus we need to "teach" the model learning the "correct" weight that should be used. To this end, we define the "loss function"

$$J(\mathbf{W}, \mathbf{b}) = \sum_{j=1}^{m} \sum_{i=1}^{10} |\{g[f(\mathbf{x}^j)] - \mathbf{y}^j\}_i|^2, \tag{1.2}$$

where $\mathbf{y}^j = (0, \ldots, 1, \ldots, 0) \in \mathbb{R}^{10}$ represents the real digit appearing in the jth picture $\mathbf{x}^j$ among the m pictures of the dataset; the only non-zero component of $\mathbf{y}^j$ is located at the ith position. This kind of labeling is called "one-hot encoding". In fact, if the digit i appears in the jth picture then the probability vector $g[f(\mathbf{x}^j)]$ must have the ith component very close to 1 (the highest probability). Therefore, the loss function (1.2) is exactly the total error of the prediction and our aim is to solve the optimization problem

$$\arg\min J(\mathbf{W}, \mathbf{b}). \tag{1.3}$$

If one can find the weight $\mathbf{W}_*$ and the intercept $\mathbf{b}_*$ so that they minimize the total error, we should use them to compute the probability of each class and make prediction. Fortunately, the problem can be solved approximately by means of the "Gradient Descent" method.

Gradient Descent Algorithm:

```
1. Initialize $\boldsymbol{W}_0$ and $\boldsymbol{b}_0$ randomly.
2. Repeat
       $(\boldsymbol{W}_{i+1}, \boldsymbol{b}_{i+1}) \leftarrow (\boldsymbol{W}_i, \boldsymbol{b}_i) - \eta \nabla J(\boldsymbol{W}_i, \boldsymbol{b}_i)$
   until $|J(\boldsymbol{W}_{i+1}, \boldsymbol{b}_{i+1}) - J(\boldsymbol{W}_i, \boldsymbol{b}_i)| \leq \delta$ for some $\delta$ small enough.
```

As a consequence, $J(\mathbf{W}_i, \mathbf{b}_i) \approx J(\mathbf{W}_*, \mathbf{b}_*)$ for large i. Here the constant η is called "learning rate". It is the coefficient for adjusting the distance from the current point $(\mathbf{W}_i, \mathbf{b}_i)$ to the next one. Because we can adjust the jump distance, the approximate total error sequence may converge quickly to the expected minimum value. The notation ∇J stands for the derivative with respect to $\mathbf{W}$ and $\mathbf{b}$.

After we found the optimal weight and the intercept, we can evaluate the model efficiency by counting the number of correct predictions. For example, I tested 10 random images as follows.

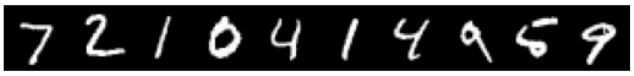

Figure 1.3: Ten random images taken from the MNIST dataset

The real labels are respectively 7, 2, 1, 0, 4, 1, 4, 9, 5, and 9. However, the digit 5 is predicted as the digit 6 by my trained model, which means that the number of correct predictions is 9/10 (i.e. 90% of accuracy). Of course, I used a part of the dataset that has never been seen by the model yet. The minimum value of the loss function that my model reached is approximately 1.672. This result is somehow good enough but one can still improve it and I let it as an exercise for all of you.

Chapter 2

Exercises

Students do the following exercises, write a report, and submit together with coding files no later than the given deadline.

Question 1. [Compulsory] In the previous chapter, we inserted the images directly to the model. It seems that the model has missed some important features of the digits so that it couldn't distinguish 5 and 6 well for example. Adding some "convolutional layers" to before the "fully connected layers" as in Fig. 1 may help the model capture more significant features of the digits. The first exercise is to implement the idea in the previous chapter and to study Convolutional Neural Networks to make the predictions more precise. (*4 points*)

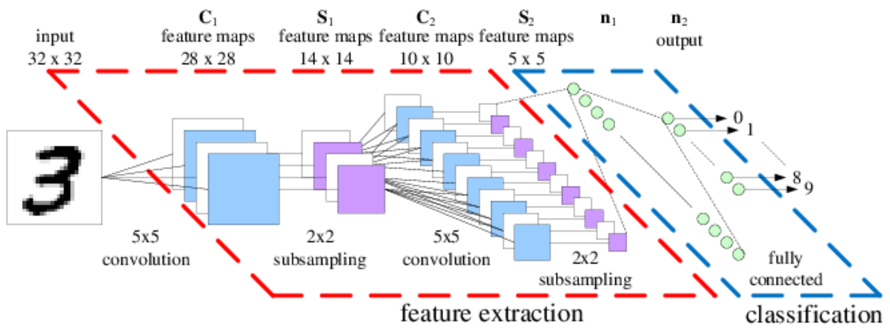

Figure 2.1: A Convolutional Neural Network architecture to predict hand-written digits (researchgate.net/publication/220785200_Efficiency_Optimization_of_Trainable_Feature_Extractors_for_a_Consumer_Platform)

Question 2. [Compulsory] It seems that the loss function (1.2) is not an effective one when we deal with probability. Indeed, we can use the Cross-Entropy loss function as an alternative. Study the Cross-Entropy loss function and implement it to see if it improve the precision of the predictions. (*3 points*)

Question 3. [Compulsory] The Gradient Descent algorithm is well-known because of its simplicity and effectiveness. However, to train the model with a "big enough" dataset is a problem. It can make the running time longer than usual. This fact is due to the computation of the gradient ∇J at every data point of the dataset. Therefore, we can train the model with a smaller subset of data, which is called a "batch of data", for each step of approximation instead of using the whole dataset. Sometime, people can train the model by chosing one random data point in each step. This technique is known as "Stochastic Gradient Descent". With a suitable learning rate η, e.g., $\eta = 0.001, 0.05, 0.1$, train the model with either Batch Gradient Descent or Stochastic Gradient Descent to see their effects on the results. (*3 points*)

Question 4. [Optional] For further interesting problems such as how to adjust the learning rate η for a better result or the recognition of more than one digits appearing in a picture. Those generalizations require rather complicated techniques. If you are interested in those problems, you can choose one and study it carefully. (*3 points*)

Bibliography

[1] https://machinelearningcoban.com/2017/02/17/softmax/.

[2] https://www.kaggle.com/c/digit-recognizer.

[3] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[4] Samay Pashine, Ritik Dixit, and Rishika Kushwah. Handwritten digit recognition using machine and deep learning algorithms. *arXiv preprint arXiv:2106.12614*, 2021.

[5] Fathma Siddique, Shadman Sakib, and Md Abu Bakr Siddique. Recognition of handwritten digit using convolutional neural network in python with tensorflow and comparison of performance for various hidden layers. In *2019 5th International Conference on Advances in Electrical Engineering (ICAEE)*, pages 541–546. IEEE, 2019.