华南理工大学

South China University of Technology

The Experiment Report of Deep Learning

SCHOOL: SCHOOL OF SOFTWARE

SUBJECT: SOFTWARE ENGINEERING

Author:
Yingting Huang, Yingying Chen
and Sentao Chen

Supervisor:
Mingkui Tan

Student ID：201720145136,
201710106604 and 201610106216

Grade:
Graduate

December 19, 2017

Handwritten Digit Recognition Based on Shallow Neural Network

Abstract—This experiment implements a convolutional neural network for the task of handwritten digital recognition via the deep learning framework pytorch. We utilize the famous dataset MNIST for training and testing a shallow neural network.

I. INTRODUCTION

In machine learning, neural network is a very powerful tool for expressing complex data patterns. Learning a neural network from training data is actually doing function approximation based on some predefined criterions. Unlike kernel method, which directly constructs complex function through feature mapping and performs linear combination on the new feature, it achieves this via function composition. And it can be mathematically proved that a neural network with only a hidden layer can approximate any smooth function as long as there are enough hidden units. As an outstanding representative of the neural network family, convolutional neural network (CNN, or ConvNet) is wildly applied to analyzing visual imagery. Basically, CNN is a class of deep, feed-forward artificial neural networks. It uses a variation of multilayer perceptrons designed to require minimal preprocessing. Another renowned network is the generative adversary nets(GAN). The basic idea behind GAN is to train a generative model(usually a network) through playing a game with the discriminator(can be another network). But we will be focusing on the CNN in this experiment.

In the deep learning community, gradient methods are usually used to trained a neural network. Specifically, stochastic gradient descent and mini-batch stochastic gradient are two popular methods since they are feasible and efficient when applied to very large datasets. They only take a small proportion of the training samples and compute the relative gradient at every iteration. There are also variants based on these methods such as Nesterov accelerated gradient (NAG), RMSprop, Adadelta and Adaptive Moment Estimation (Adam). However, computing the gradient of a highly composite function is not easy. It requires constantly applying the chain rule from the back end to the very beginning of the network. We call this procedure as back propagation(BP).

Armed with BP, we train a network on the MNIST handwritten digit dataset and get a test error of 95%.

II. METHODS AND THEORY

We use cross entropy loss in our experiment. The benefit in using this loss function is that it is differentiate and that the output of the network makes probabilistic sense. We minimize the empirical loss through running the stochastic gradient descent algorithm. Note that the empirical loss function is not guaranteed to be a convex function, which means that it might have many local optimal solutions.

III. Experiment

A. Dataset

We use the handwritten digital dataset - MNIST, which contains 60,000 hand-written digital images for training and 10,000 hand-written digital images for validation. Each image is 28×28 pixels in size.

B. Implementation

We perform the experiment by using the deep learning framework—pytorch. We take the following steps:

1, Build shallow neural network LeNet5.

2, Complete the network training and validation process.

2.1,Usetorch.utils.data.DataLoader,torchvision.datasets.MNIST and torchvision.transforms to load dataset.

2.2,Instantiate torch.optim.SGD to get the optimizer.

2.3,Instantiate LeNet5 to get net.

2.4 ,Input data, which size is batch size, into the network for forward propagation to get predict target.

2.5, Use torch.nn.CrossEntropyLoss to calculate the loss between predict target and ground truth target.

2.6, Calculate gradient using loss.backward and optimize the net using optimizer.step.

2.7, Calculate accuracy of recognition.

2.8 ,Repeat step 2.4 to 2.7 until all data are inputed to the net.

2.9 ,Define the number of selected training epoch, repeat the training process 2.5--2.9

3, Perform forward propagation of the learned network on the test data and compute the test error.

We run the epoch for 5 times and record the training accuracy in the Table 1:

Table 1

Train Epoch 1	94.8950%
Train Epoch 2	95.9383%
Train Epoch 3	97.7583%
Train Epoch 4	97.8450%
Train Epoch 5	98.2583%

We also get a test accuracy of 95.2900% on the left-out test data.

IV. CONCLUSION

Through this experiment, we learn how to construct a deep neural network using pytorch. We have also deepened our understanding of the concepts of convolution, pooling, and ReLu.