Hand-writing characters recognition by convolution neuron network

Guanzhang Li
Boston University
gzli@bu.edu

Abstract

The goal of the Neural network is to recognize the hand-writting characters that in the Emnist Data-set. The code use PyTorch for convolution network and back-propagation. In addition, the reasons for the how the network is constructed were analysed and listed below.

1 Introduction

The EMnist data set is extend data set for Mnist. It include numbers of the images of hand-writing characters, and lable. The input size of the the image is 28*28. The neuron nwtwork was constructed by 1 convolutional neuron network, then a maxpooling layer, and the other convolutional neuron network, and a maxpooling layer. Next, it connects to two fully connected network and the result was output after the process of softmax. The structure of the network was borrowed from the famous LeNet-5 from LeCun[1].

2 Approach

For this part, I will introduce the framework of my code, and compare different constructions of the framework.

2.0.1 Convolutional Neuron Network

The size of the image is 28*28, and we need to padding the image. If we not padding the image, the pixels at the edge of the image would only be convoluted once. To prevent this situation, by considering the kernel size is 5*5, we need to pad the image by 2 pixels each. In the end, the input size is finally 32*32. For the first CNN, I choose the channel to be 20, so that I can extract 20 different feature from the original image. Because I have 20 channel input into second CNN. For the reason I choose to use 2 CNN is that more CNN can extract more different details than one single CNN.

2.0.2 Fully Connected Network

After maxpooling from the second CNN, there are 4*4*50, which is 800 pixels that pass to the fully connected neuron. First the image need to flatten into a 2d array, and the pixel information can be passed to the Fully connected neuron. Finally, after the softmax, the model will get the most likely prediction of the characters.

2.0.3 Back-Propagation

Back-propagation of the CNN is a little bit complex. From the pooling layer, only the biggest number in the pooling area affect the result. Therefore, the influence of the rest of the area's to the result is 0.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

The biggest part to the previous layer is δ.From the chain rule, if we know the lost from the last layer, for instance, Z^l. $\delta^l(x, y)$ be the

$$\delta^l(x,y) = \sum_{x'}\sum_{y'} \frac{dC}{dZ^{l+1}(x',y')} \frac{dZ^{l+1}(x',y')}{dZ^l(x,y)} = \sum_{x'}\sum_{y'} \delta^{l+1}(x',y') \frac{dZ^{l+1}}{dZ^l(x,y)}$$

After expanding and simplify the formula above, we get

$$\delta^l(x,y) = \sum_{x'}\sum_{y'} \delta^{l+1}(x',y') w(a,b) \sigma'(Z^l(x,y))$$

subject to $x' + a = x$ and $y' + b = y$. And then, we put $x' + a = x$ and $y' + b = y$ to the equation and then let $a' = -a$ and $n' = -b$ then we have the conclusion:

$$\delta^l(x,y) = \delta^{l+1}(x,y) * ROT180(w^{l+1}) \otimes \sigma'(Z^l)$$

Then for every layer, we need to get the gradient for the parameter. If it is fully connected network:

$$\frac{dC}{dW^l} = \delta^l (a^{l-1})^T, \frac{dC}{db^l} = \sigma^l$$

If it is CNN:

$$\frac{dC}{dW^l} = \delta^l * \sigma(Z^{l-1}), \frac{dC}{db^l} = \sum_x \sum_y \delta^l$$

Then for a batch of data we update the parameter:

$$W^l = W^l - \frac{lr}{batchsize} \sum \frac{dC}{dW^l}, b^l = b^l - \frac{lr}{batchsize} \sum \frac{dC}{db^l}$$

3 result

After first training. Test set: Average loss: 0.4612, Accuracy: 17707 / 20800 (0.85)
After second training. Test set: Average loss: 0.3128, Accuracy: 18768/20800 (0.90)
After third training. Test set: Average loss: 0.2765, Accuracy: 18980/20800 (0.91)

References

[1] Y. Lecun, L. & Bottou, Y. & Bengio and P. & Haffner, Gradient-based learning applied to document recognition,*in Proceedings of the IEEE* , vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.