Deep Learning Homework 1

Gezhi Xiu
1801110566

May 23, 2019

baseline model

We used softmax as a baseline normalizing function here.

The results is shown in 1

average loss	average accuracy
0.395	0.943

Table 1: Results for baseline model after 30 epoches.

1 alternatives of softmax

Softmax is a function that put all the features together with the exponential functions which makes the most important feature dominant. Softmax function has the form like

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

In this section, we are going to alternate it with some other forms of normalizers.

1.1 abs-max

The form of abs-max is

$$\text{abs-max}(x_i) = \frac{|x_i|}{\sum_j |x_j|}.$$

The results is shown in 2

average loss	average accuracy
0.394	0.659

Table 2: Results for abs-max model after 30 epoches.

It is totally not working as well as Softmax.

1.2 square-max

The form of square-max is

$$\text{square-max}(x_i) = \frac{x_i^2}{\sum_j x_j^2}.$$

The results is shown in 3

average loss	average accuracy
0.394	0.573

Table 3: Results for square-max model after 30 epoches.

It is even worse than abs-max. So it is not an appropriate way to substitute softmax.

1.3 plus-one-abs-max

The form of plus-one-abs-max is

$$\text{plus-one-abs-max}(x_i) = \frac{1 + |x_i|}{\sum_j |1 + x_j|}.$$

The results is shown in 4
The results is still rather poor.

average loss	average accuracy
0.399	0.558

Table 4: Results for plus-one-abs-max model after 30 epoches.

1.4 non-negative-max

The form of non-negative-max is

$$\text{non-negative-max}(x_i) = \frac{\max(x_i, 0)}{\sum_j \max(x_i, 0)}.$$

The results is shown in 5

average loss	average accuracy
0.498	0.943

Table 5: Results for non-negative-max model after 60 epoches.

I accidentally trained the model with 60 epoches. But the result was also better at the 30th epoch. than the 3 previous ones. So I found out that the best substitute for softmax is this non-negative-max function!

2 Regression vs Classification

In this part, I changed cross entropy loss to the square of Euclidean distance between model predicted probability and one hot vector of the true label. The model converges faster than before. And we got a result like

average loss	average accuracy
0.008	0.949

Table 6: Results for square of Euclidean distance loss function.

3 $\mathcal{L}_p$ pooling

Here, we changed the pooling function to $\mathcal{L}_p$ pooling. And assume that $p = 2$ by default. I found that the model couldn't catch the features properly at the beginning, but developing fast later on, and finally get a result are shown in the following table:

average loss	average accuracy
0.037	0.766

Table 7: Results for $\mathcal{L}_2$ pooling.

4 Regularization

4.1 $\mathcal{L}_p$ regularization with different p

Here, I experimented $p = 3$ or $p = 4$ to compare with $p = 2$ in the baseline model.

p	average loss	average accuracy
2	0.395	0.943
3	0.009	0.946
4	0.008	0.947

Table 8: Results for $\mathcal{L}_p$ regularization.

and we saved the best result in q4.1.

4.2 Setting Lp regularization to a minus number.

Surprisingly, the result for this change sometimes is fascinating. It exceeded 95% at some epoches.

average loss	average accuracy
0.008	0.949

Table 9: Results for setting $\mathcal{L}_p$ regularization to a minus number.