Deep Learning: Homework1

1706 Wanru Zhao(17373240)

April 2020

1 Finding alternatives of softmax

$$\text{softmax}(x) = \frac{e^x}{\sum e^x}$$

Which of below work as alternative to Softmax?

$$\text{abs-max}(x) = \frac{|x|}{\sum |x|}$$

$$\text{square-max}(x) = \frac{x^2}{\sum x^2}$$

$$\text{plus-one-abs-max}(x) = \frac{1+|x|}{\sum 1+|x|}$$

$$\text{non-negative-max}(x) = \frac{\max(0,x)}{\sum \max(0,x)}$$

Figure 1: All Functions

1.1 Method

Since the softmax_cross_entropy function of tensorflow contains softmax inside, we cannot use the loss function of the baseline and need to modify the loss function according to

$$\mathrm{H}(p,q) = -\sum_x p(x) \log q(x) \tag{1}$$

The code is as follows

```
    loss = -tf.reduce_mean(
        tf.reduce_sum(tf.cast(label_onehot, dtype=tf.float32) *
        tf.log(tf.clip_by_value(preds, 1e-10, 1.0)), axis=-1))
        + loss_reg
```

Here, the tf.clip_by_value() method is used to smooth the predicted probability to ensure that there is no backward propagation of log operation with probability 0 in the result.

1.2 Results

Question	Function	Best Accuracy
baseline	soft-max	96.8%
q1.1	abs-max	28.7%
q1.2	square-max	43.8%
q1.3	plus-one-abs-max	95.1%
q1.4	non-negative-max	60.1%

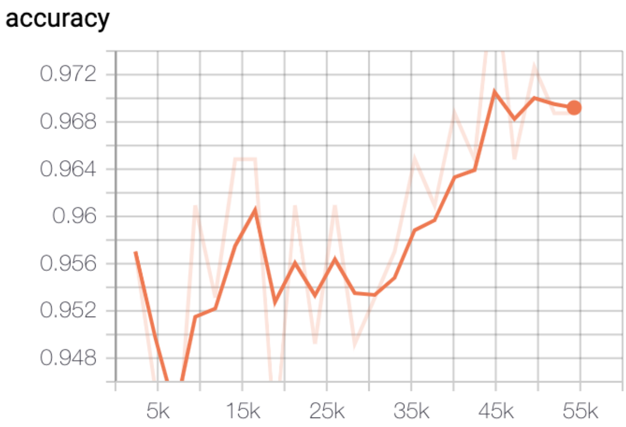
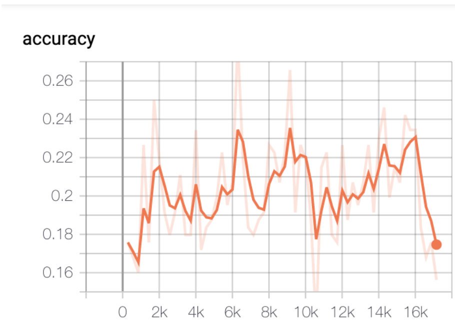
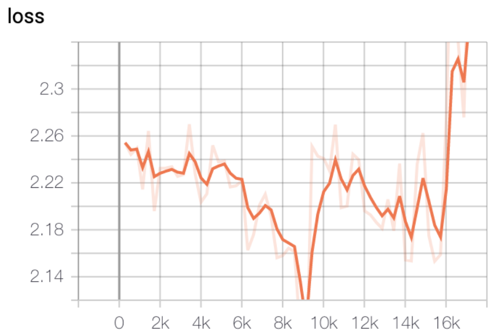


Figure 2: Baseline

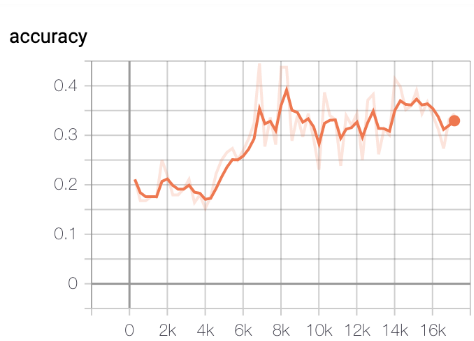


(a) accuracy curve

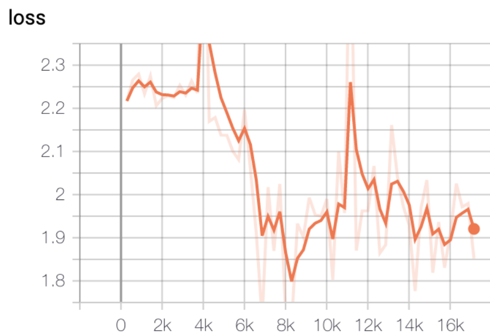


(b) loss curve

Figure 3: abs-max

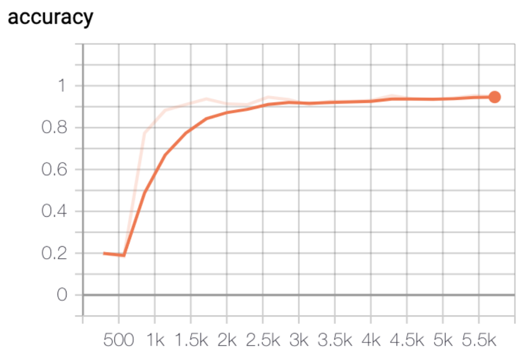


(a) accuracy curve

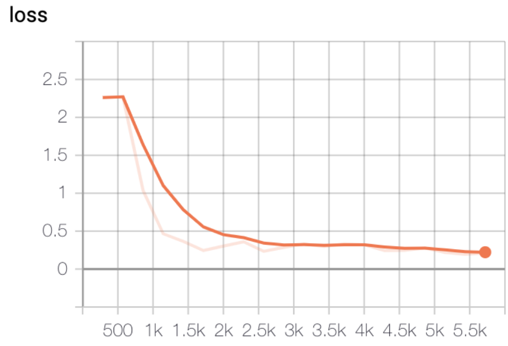


(b) loss curve

Figure 4: square-max

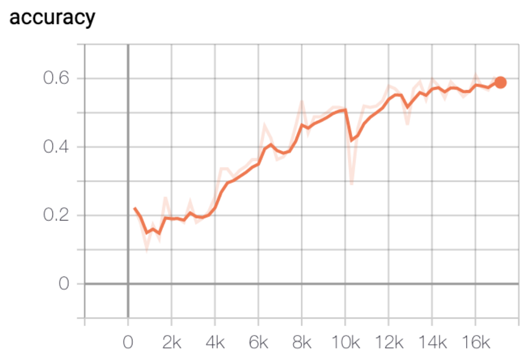


(a) accuracy curve

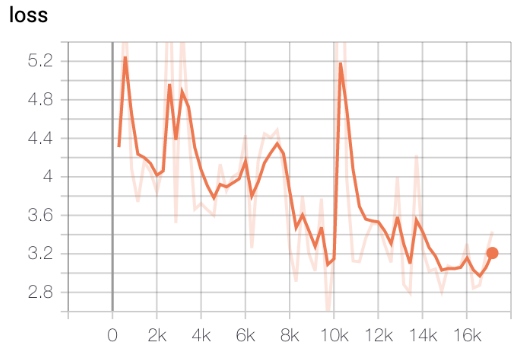


(b) loss curve

Figure 5: plus-one-abs-max



(a) accuracy curve



(b) loss curve

Figure 6: non-negative-max

1.3 Analysis

1 The softmax method of baseline enters into the cross entropy. The loss function is a convex function, and the greater the loss is, the greater the gradient is, which is convenient for back propagation and rapid convergence.

2 The abs - max method and square - max are even functions and not convex functions, so it's hard for them to converge.

3 The plus-one-abs-max method is the only one of the four methods that is close to the baseline. It can be an alternative of softmax.

4 The curve of the non-negative-max method is similar to that of the baseline, and it converges faster than the previous two methods, but less than the plus-one-abs-max method. It may also be an alternative of softmax.

2 Regression vs Classification

Change cross entropy loss to the square of euclidean distance between model predicted probability and one hot vector of the true label.

2.1 Method

Change the loss function to MSE according to

$$MSE\left(\mathrm{y}, \mathrm{y}'\right) = \frac{\sum_{i=1}^{n}\left(y_i - y_i'\right)^2}{n} \tag{2}$$

```
    if args.loss == 'regression':
        loss = tf.reduce_mean(tf.reduce_sum(tf.square(preds -
        tf.cast(label_onehot, dtype=tf.float32)), axis=1))
    else:
        loss = tf.losses.softmax_cross_entropy(label_onehot, logits) +
            loss_reg
```

2.2 Results

Question	Function	Accuracy
baseline	Cross Entropy Error	96.8%
q2	Mean Squared Error	95.1%

2.3 Analysis

The cross entropy loss is faster and converges better than the variance regression.

3 Lp pooling

Change all pooling layers to Lp pooling. [1]

3.1 Method

$$O = \left(\sum \sum I(i,j)^P * G(x,y) \right)^{\frac{1}{P}} \tag{3}$$

Lp pooling is a biologically inspired pooling layer modelled on complex cells who's operation can be summarized in equation (1), where G is a Gaussian kernel, I is the input feature map and O is the output feature map. It can be imagined as giving an increased weight to stronger features and suppressing weaker features. Two special cases of Lp pooling are notable. P = 1 corresponds to a simple Gaussian averaging, whereas P = corresponds to max-pooling (i.e only the strongest signal is activated).

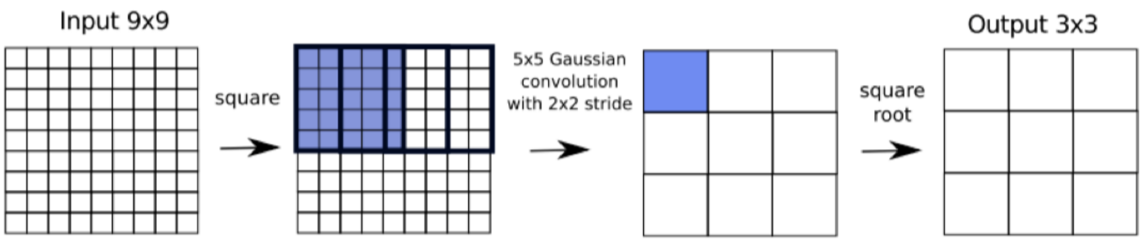


Figure 7: a simple example of L2-pooling

Compute the gaussian kernel in advance.

```
def gauss(ksize):
    sigma = ((ksize - 1) * 0.5 - 1) * 0.3 + 0.8
    sum_val = 0
    kernel = np.zeros((ksize, ksize))
    center = ksize // 2
    for i in range(ksize):
        for j in range(ksize):
            kernel[i, j] = np.exp(-((i - center) ** 2 + (j - center) **
                 2) / (2 * sigma ** 2))
            sum_val += kernel[i, j]
    kernel = kernel / sum_val
    return kernel
```

```

G(x, y):
[[0.05711826 0.12475775 0.05711826]
 [0.12475775 0.27249597 0.12475775]
 [0.05711826 0.12475775 0.05711826]]
```

The input feature dimension [WHC] is disassembled into c [wh1] features, and the corresponding [KKC] convolution is disassembled into c [kk1] gaussian kernel, each of which is put into the convolution function to generate c [wh1] output features, and spliced into [WHC] size. So we're convolving the characteristics of each channel. Finally, the stride length in convolution is set to 2, and finally the pooling effect is achieved by convolution.

3.2 Results

Question	Function	Accuracy
baseline	max-pooling	96.8%
q3	lp-pooling p=-1	92.1%
q3	lp-pooling p=1	93.4%
q3	lp-pooling p=2	96.2%
q3	lp-pooling p=4	95.1%

It can be seen from the experimental results that the lp-pooling test accuracy is significantly higher than that of the max-pooling at the baseline, indicating that the gaussian kernel selects better features than the maximum value after the convolution smoothing operation of the input features.

3.3 Analysis

4 Regularization

- Try Lp regularization with different p. Pick one number p with best accuracy.
- Set Lp regularization to a minus number. (L_model + L_reg to L_model - L_reg)

4.1 Method

When weight_decay was equal to 1e-10, changing the P value made little difference to the result. So we set weight_decay1e-3.

```
self.reg = lambda x: tf.reduce_sum(tf.pow(tf.abs(x), config.lp_reg))
            * config.weight_decay
```

Set p=2 in q.2.

```
self.reg = lambda x: tf.reduce_sum(tf.pow(tf.abs(x), config.lp_reg))
            * config.weight_decay * (-1)
```

4.2 Results

Question	Method	Accuracy
q4.1	p=-1	82.8%
q4.1	p=1	90.9%
q4.1	p=2	93.6%
q4.1	p=4	95.4%

Method	Accuracy
baseline(no extra$_3$232)	94.3%
q4.2(no extra$_3$232)	94.3%

4.3 Analysis

- When p is negative, the results are worse. When p=4, the test accuracy is the highest. So the Lp4 regularization constraint on the result is the most effective in this experiment.
- Whether the regularization is positive or negative has little effect on the accuracy of the whole neural network.

References

[1] Pierre Sermanet, Soumith Chintala, and Yann LeCun. Convolutional neural networks applied to house numbers digit classification. *CoRR*, abs/1204.3968, 2012.