Deep Learning Assignment1 Bonus

Mengdi Xue, mengdix@kth.se

April 2018

1 Exercise 2.1 Method

I used the following methods to improve the training accuracy:

1. Use all the available training data for training. All the five batches were loaded to train, except for the last 1000 pieces of data for validation.

2. Train for a longer time and keep a record of the best model to avoid overfitting. I increased n_epochs to 100 to train for a longer time.

3. Decay the learning rate by 0.9 after every 10 steps. This is the most helpful improvement which gains about 3% of accuracy.

2 Exercise 2.1 Result

The best test accuracy is 40.30%. The result figure and the loss is as in Figure 1 and Figure 2.

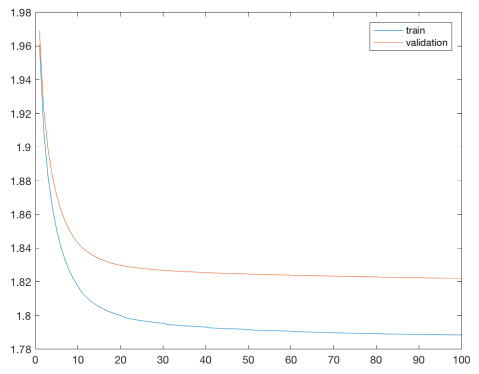

Figure 1: Cross-entropy loss

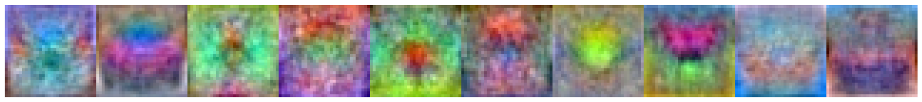

Figure 2: Result figure

3 Exercise 2.2 Method

In SVM multi-class loss, the cost function is:

$$J = \frac{1}{|D|} \sum_{(x,y)\in D} \sum_{j=1, j\neq y}^{C} max(0, s_j - s_y + 1) + \lambda R(W)$$

So in the code I rewrite the cost function according to the above equation.

Then, compared to cross-entropy loss, to apply SVM multi-class loss, we need to rewrite $g = l$. So I computed the gradient of l as following:

(1) When $k \neq y_i$:

$$\begin{aligned}\frac{\partial l_i}{\partial s_k} &= \frac{\partial(\sum_{j\neq y} max(0, s_j - s_y + 1))}{\partial s_k} \\ &= \frac{\partial(max(0, s_1 - s_{y_i} + 1) + max(0, s_2 - s_{y_i} + 1) + ... + max(0, s_C - s_{y_i} + 1))}{\partial s_k} \\ &= \frac{\partial(max(0, s_k - s_{y_i} + 1)}{\partial s_k} \\ &= \begin{cases} 0 & \text{if } s_k - s_{y_i} + 1 < 0 \\ 1 & \text{if } s_k - s_{y_i} + 1 \geq 0 \end{cases}\end{aligned}$$

(2) When $k = y_i$:

$$\begin{aligned}\frac{\partial l_i}{\partial s_k} &= \frac{\partial(\sum_{j\neq y} max(0, s_j - s_y + 1))}{\partial s_k} \\ &= -(M-1) \text{ (M is the number of data s.t. } s_k - s_{y_i} + 1 \geq 0)\end{aligned}$$

4 Exercise 2.2 Result

Then I applied two training parameter settings and compared them as following: The first setting is that:

$$lambda = 0.01; n\_epochs = 100; n\_batch = 100; eta = .001$$

I got accuracy of **36.22%** with SVM loss and **37.38%** with cross-entropy loss. The result loss is as in Figure 3 and Figure 4.

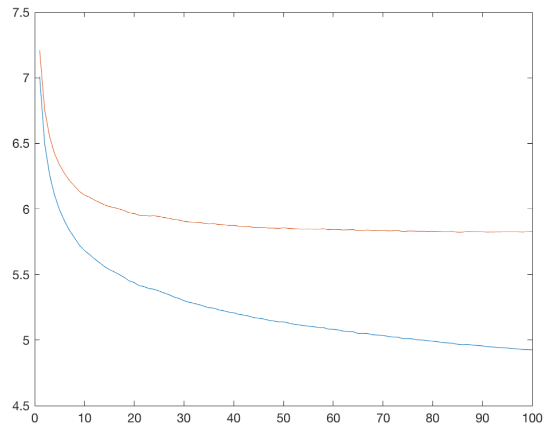

Figure 3: SVM loss: $lambda = 0.01; n\_epochs = 100; n\_batch = 100; eta = .001$

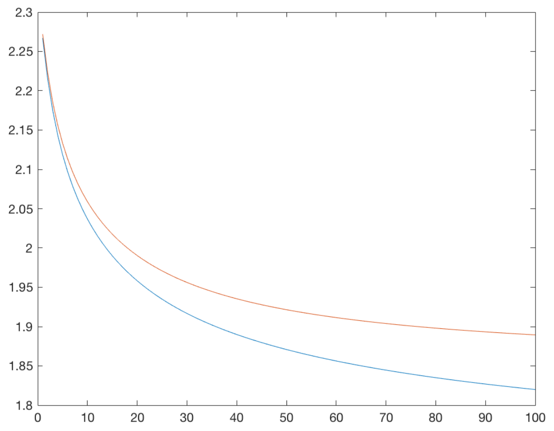

Figure 4: Cross-entropy loss: $lambda = 0.01; n\_epochs = 100; n\_batch = 100; eta = .001$

The second setting is:

$$lambda = 0; n\_epochs = 40; n\_batch = 100; eta = .01$$

I got accuracy of **30.97%** with SVM loss and **36.87%** with cross-entropy loss. The result loss is as in Figure 5 and Figure 6.

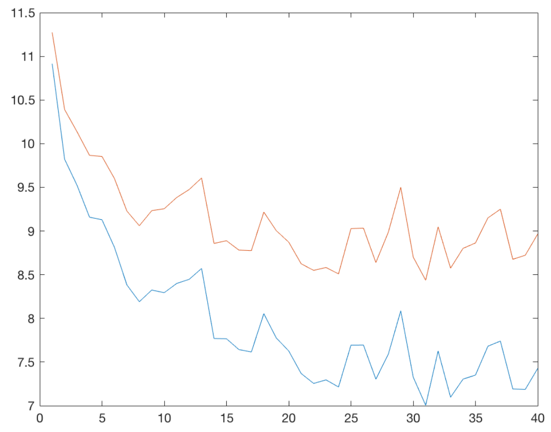

Figure 5: SVM loss: $lambda = 0; n\_epochs = 40; n\_batch = 100; eta = .01$

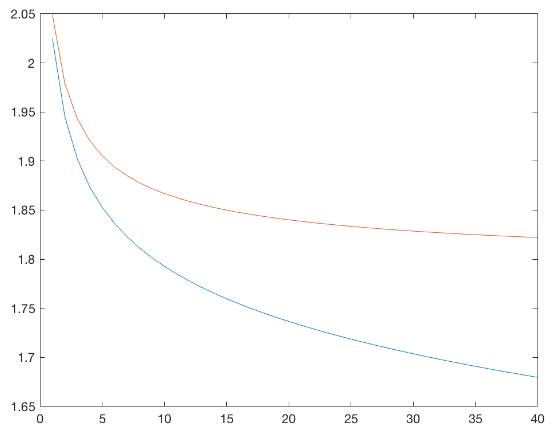

Figure 6: Cross-entropy loss: $lambda = 0; n\_epochs = 40; n\_batch = 100; eta = .01$

5 Exercise 2.2 Conclusion

1. In SVM loss, the learning rate should be much less than in cross-entropy loss in order to get a smoother decreasing of the cost. Because SVM Loss is a non-smooth and non-differentiable loss, while cross-entropy loss is smooth and differentiable.
2. SVM loss lacks of scalability, while cross-entropy loss can solve large scale problems better.