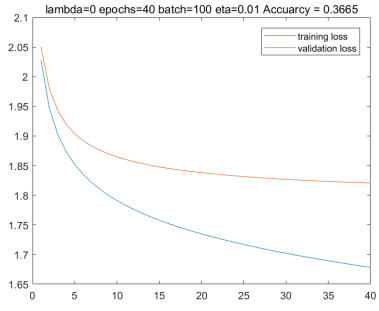
DD2424 Deep Learning in Data Science
Assignment 1 Bonus

Lingxi Xiong
lingxi@kth.se

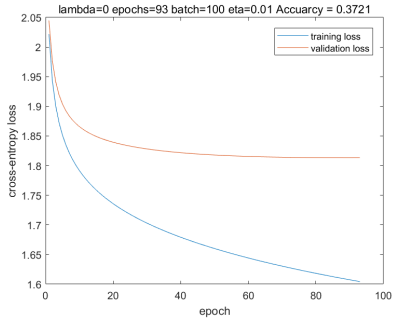
2nd April, 2019

Exercise 2.1

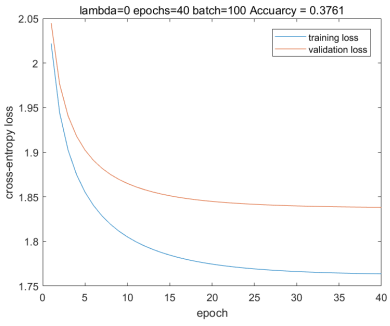
The three following way are tried to improve the performance of the model.



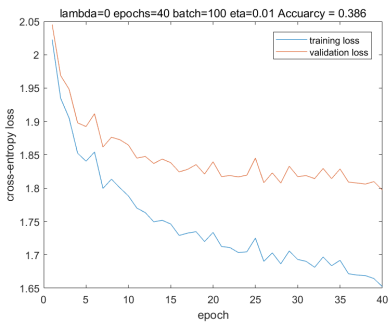
(a) Original



(b) Early stopping



(c) Learning rate decay



(d) Shuffle the training samples

Figure 1: Total loss on the training data and validation data

In Figure 1, (b) shows the change of loss during training by training for a longer time and use validation set to avoid overfitting. Here the criteria of early stopping is when the validation loss no longer decrease for 3 epochs. (c) is the performance with decaying the learning rate by 0.9 after each epoch. (d) shuffles the order of training samples at the beginning of every epoch. All of these test follow the same setting of the best result from Exercise 1, where the maximum number of epochs is 40(if not using early stopping), learning rate as 0.01(if not decaying the learning rate), $\lambda = 0$, batch size is 100 and the same initialization of W and b by using $rgn(400)$ seed. We can see that among these methods, shuffling the data achieve the largest gain, i.e. reach the better accuracy on the test data. And all method improve the model's performance.

Exercise 2.2

The SVM multi-class loss(hinge loss, without adding regularization term) for training example $\mathbf{x}$ with label y is defined in equation (1)

$$l = \sum_{j=1, j\neq y}^{C} max(0, s_j - s_y + 1) \tag{1}$$

where $s_j = f_j(\boldsymbol{x}; \boldsymbol{w_j}, b_j) = \boldsymbol{w_j^T x} + b_j$.

To rewrite it in matrix manner, the loss over all training data will be

$$\begin{aligned} L(\mathcal{D}, W, \mathbf{b}) &= \frac{1}{\mathcal{D}} \sum_{(x,y)\in\mathcal{D}} l(\mathbf{x}, y, W, \mathbf{b}) \\ &= \frac{1}{\mathcal{D}} \sum_{(x,y)\in\mathcal{D}} [max(\mathbf{0}, \mathbf{S} - \mathbf{1}(\mathbf{Y} \odot \mathbf{S}) + \mathbf{1}) - \mathbf{Y}] \\ &= \frac{1}{\mathcal{D}} \sum_{(x,y)\in\mathcal{D}} \mathbf{L} \end{aligned}$$

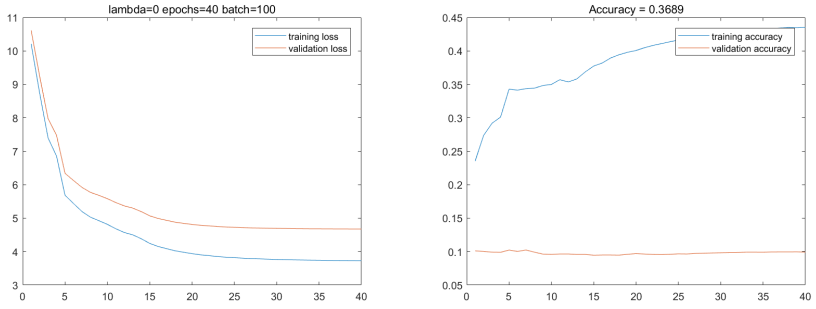
where $\mathbf{S} = W\mathbf{X} + \mathbf{b}$ and $\mathbf{Y}$ is the one-hot encoding matrix of y labels. K is the number of class and N is the number of data samples. Here $\odot$ symbol represent the Hadamard/element-wise product.

Summarizing from the definition of partial derivative of loss function with respect to $w_{i,j}$, the gradient matrix can be defined as:

$$\mathbf{G} = \mathbf{Ind}(\mathbf{L} > 0) - \mathbf{nzs} \odot \mathbf{Y} \tag{2}$$

Where $\mathbf{nzs}$ is the vector summing over $\mathbf{Ind}(\mathbf{L} > 0)$ with respect to the rows. The more detailed computation can be seen at the *ComputeSVMGradients* function in the code.

Figure 2 is an example with lambda=0, batch size=100 and learning rate decay from 0.01 after 40 epochs of training.



(a) Loss (b) Accuracy

Figure 2

Table 1 compares the test accuracy of the network trained with the SVM loss to the one trained with cross-entropy loss under several sensible training parameter settings. Here the learning rate decay method is applied to achieve a more sensible result. We can see that SVM loss approach performs better with regularization while training cross-entropy loss is the opposite. Also, the SVM loss approach is more sensitive to learning rate, i.e. need to be trained under a lower learning rate to avoid jumping/unstable loss.

Table 1: Test accuracy comparison

lambda	start_eta	epochs	batch	Accuracy(cross-entropy loss)	Accuracy(SVM loss)
0	0.1	50	100	0.3876	0.3451
0	0.01	50	100	0.3757	0.3705
0.01	0.01	50	100	0.3751	0.3659
0.1	0.01	50	100	0.3654	0.3718
1	0.01	50	100	0.3177	0.3594