DD 2424 - Assignment 1 Bonus

Jia Fu (jiafu@kth.se)

April 10, 2021

Optimize the performance of the network

From assignment 1, it can be seen that $\eta = 0.001$ and $\lambda = 0.1$ gives the best accuracy. Thus, the optimization is based on these two parameters' setting. `n_batch` and `n_epoch` keep 100 and 40 respectively. I explored three strategies to help bump up performance:

1. Shuffle the order of training examples at the beginning of every epoch.
2. Train the model using all five batches which minus a small subset of the training images for a validation set (size: 1000).
3. Play around with decaying the learning rate by a factor after each epoch.

I compared the accuracy of the network when using a single or a combination of above strategies versus using none. To make the results comparable, random seed was fixed as `rng(400)`.

Figure 1 shows that shuffling the order of training examples at the beginning of every epoch can slightly improve the accuracy. A more prominent improvement in accuracy (around 2%) can be seen in figure 2. Applying all available training data for training and reduce the validation set to a reasonable size is an effective strategy for us to consider.

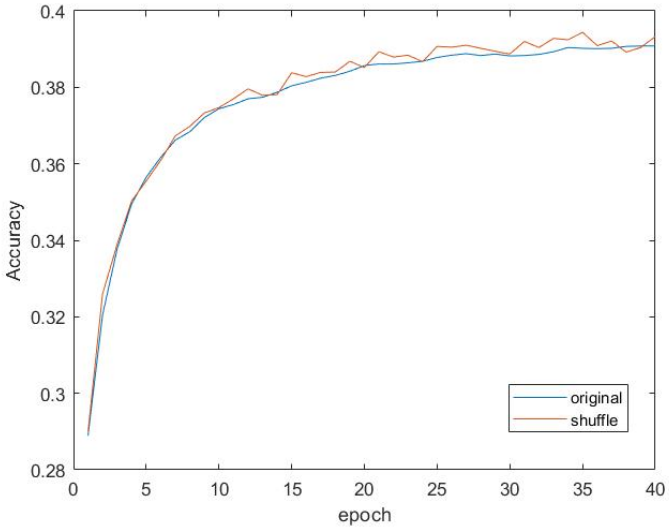


Figure 1: Accuracy comparison before and after shuffling the order of training examples

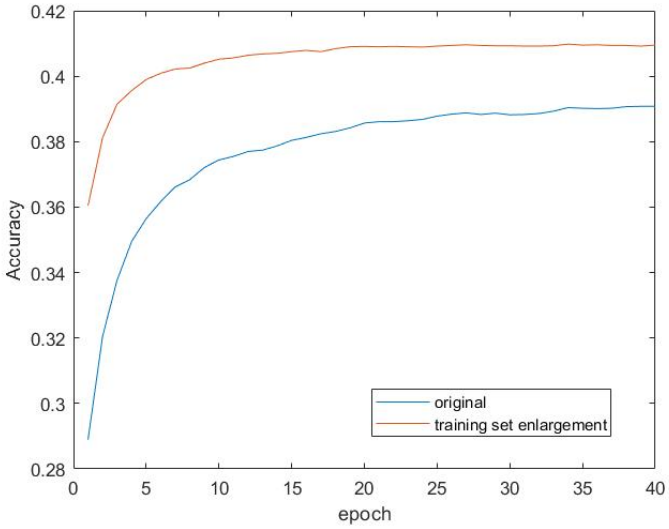
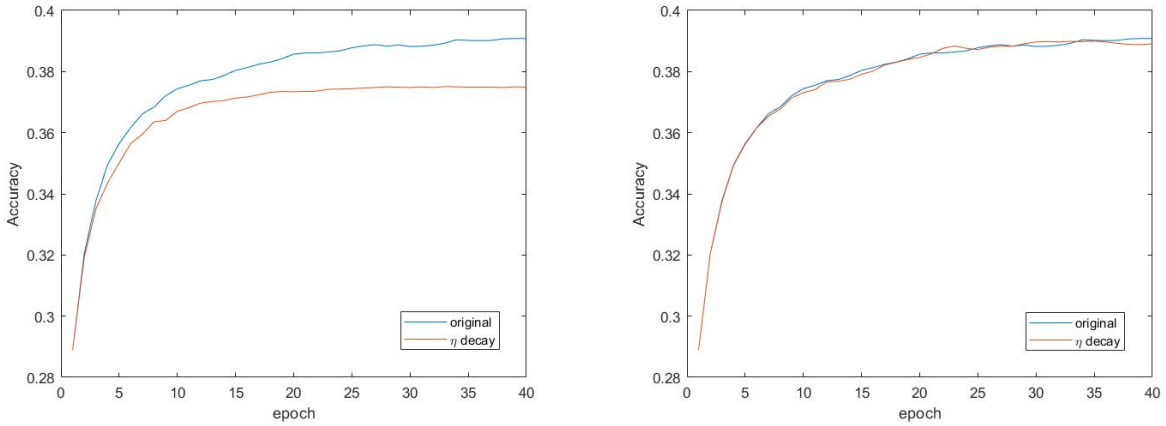


Figure 2: Accuracy comparison before and after adjusting the training and validation set

However, decaying learning rate by a factor after each epoch is not always successful. When initial value of η is appropriate enough for training, an unreasonable attenuation factor will decrease the accuracy rate. Here applying decay factor 0.9 on $\eta = 0.001$ cause the reduction in accuracy shown by figure 3. Although adjust decay factor to 0.99 will eliminate the gap between the results of applying the attenuation or not, it won't bring the increase in accuracy.

When initial η setting is not suitable for the network, this method is certainly worth considering. Shown by figure 4, decaying $\eta = 0.01$ by a factor of 0.9 after each epoch will bring a nearly four percent improvement in accuracy.

At last, all three above approaches are employed to give the best optimization shown in figure 5 where `n_batch` $= 100$, `n_epoch` $= 40$, `eta` $= .001$, `lambda` $= .1$ and `decay_rate` $= .99$. Finally, the accuracy reaches around 41%.



(a) decay by 0.9

(b) decay by 0.99

Figure 3: Accuracy comparison before and after decaying the learning rate after each epoch

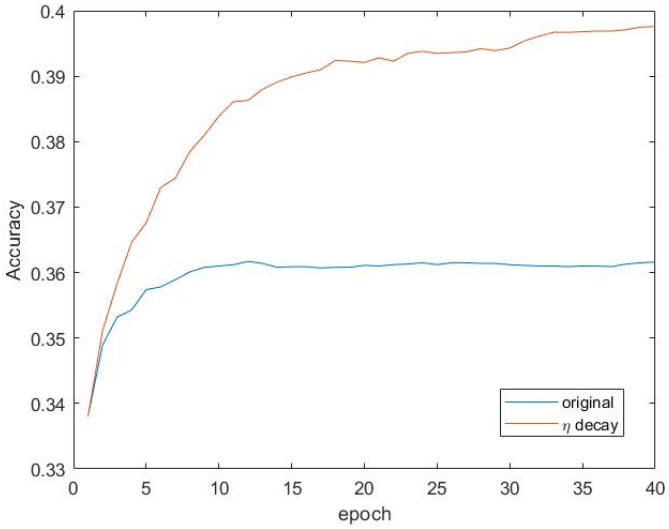


Figure 4: Accuracy comparison before and after decaying the initial learning rate 0.01 by factor 0.9

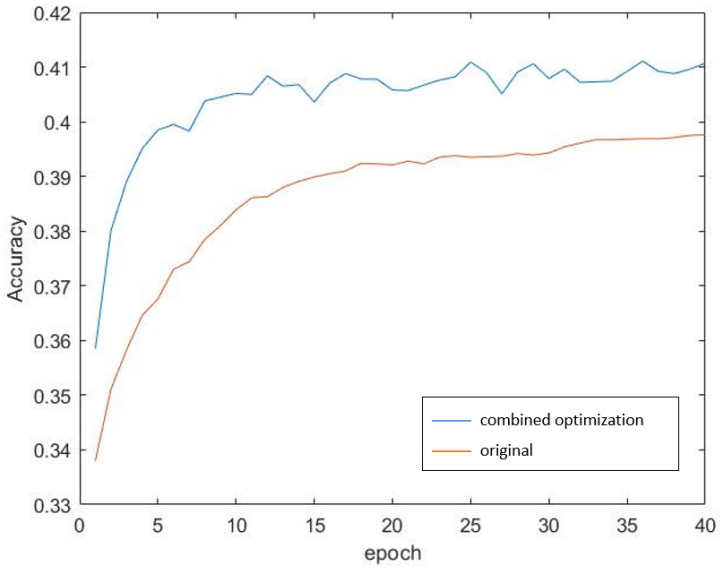


Figure 5: Accuracy comparison before and after using combined optimization methods

Train network by minimizing the SVM multi-class loss

For a single datapoint, the SVM multi-class loss fuction is:

$$L_i = \sum_{j \neq y_i} \left[\max\left(0, w_j^T x_i + b_j - w_{y_i}^T x_i - b_{y_i} + 1\right) \right]$$

Taking the gradient with respect to w_{y_i} and b_{y_i} we obtain:

$$\nabla_{w_{y_i}} L_i = -\left(\sum_{j \neq y_i} \mathbf{1}\left(w_j^T x_i + b_j - w_{y_i}^T x_i - b_{y_i} + 1 > 0\right) \right) x_i$$

$$\nabla_{b_{y_i}} L_i = -\sum_{j \neq y_i} \mathbf{1}\left(w_j^T x_i + b_j - w_{y_i}^T x_i - b_{y_i} + 1 > 0\right)$$

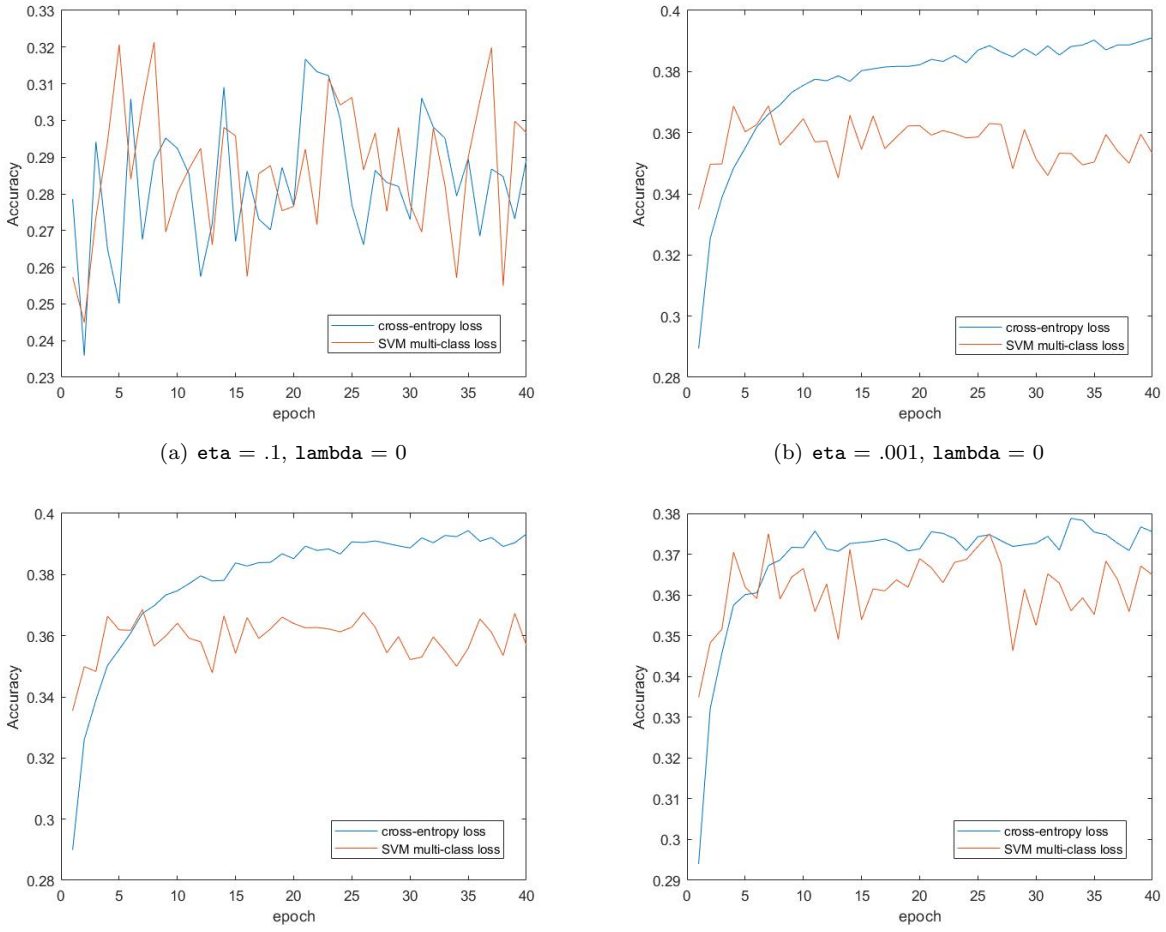
where $\mathbf{1}$ is the indicator function that is one if the condition inside is true or zero otherwise. For $j \neq y_i$, we have:

$$\nabla_{w_j} L_i = \mathbf{1}\left(w_j^T x_i + b_j - w_{y_i}^T x_i - b_{y_i} + 1 > 0\right) x_i$$

$$\nabla_{b_j} L_i = \mathbf{1}\left(w_j^T x_i + b_j - w_{y_i}^T x_i - b_{y_i} + 1 > 0\right)$$

Once we gain the expressions for the gradients it is straight-forward to implement the expressions and use them to perform the gradient update.

Figure 6 shows the results of accuracy comparisons between the models trained with SVM multi-class loss and the cross-entropy loss when applying the same parameter settings. When the η and λ are set proper, the model trained by cross-entropy approach has the better performance. Its accuracy curve fluctuates less and the gap between two accuracy curves can reach around 3% after training.



(a) `eta` = .1, `lambda` = 0

(b) `eta` = .001, `lambda` = 0

(c) `eta` = .001, `lambda` = .1

(d) `eta` = .001, `lambda` = 1

Figure 6: Accuracy of the network trained with the SVM multi-class loss compared to the cross-entropy loss