DD2424 Assignment1 Optional Part

Shuyuan Zhang shuyuanz@kth.se

March 2019

1 Optimize the performance of the network

In this part, I tried to implement 4 improvements to my model:

(a) Use all the available training data

(b) Train for a longer time

(c) Do a grid search to find good values for parameters

(d) Learning rate decay

1.1 Find good values for parameters

First, I want to do a grid search and find a set of good parameters. The parameter set which gives the best accuracy will be used as a baseline for further comparisons.

Test settings	Train accuracy	Test accuracy
lambda = 0, learning_rate = 0.01, batch_size = 25	0.4552	0.3683
lambda = 0, learning_rate = 0.01, batch_size = 50	0.4404	0.372
lambda = 0, learning_rate = 0.01, batch_size = 100	0.4158	0.3691
lambda = 0, learning_rate = 0.001, batch_size = 25	0.4129	0.3778
lambda = 0, learning_rate = 0.001, batch_size = 50	0.3979	0.3713
lambda = 0, learning_rate = 0.001, batch_size = 100	0.378	0.3589
lambda = 0.1, learning_rate = 0.01, batch_size = 25	0.3325	0.3136
lambda = 0.1, learning_rate = 0.01, batch_size = 50	0.3676	0.3455
lambda = 0.1, learning_rate = 0.01, batch_size = 100	0.3416	0.334
lambda = 0.1, learning_rate = 0.001, batch_size = 25	0.3839	0.3589
lambda = 0.1, learning_rate = 0.001, batch_size = 50	0.3848	0.3623
lambda = 0.1, learning_rate = 0.001, batch_size = 100	0.3721	0.3537

Table 1: Grid search

From all 12 settings above, $lambda = 0, learning\_rate = 0.001, batch\_size = 25$ shows the best test accuracy of 0.3778.

1.2 Use all the available training data

In this part, we use all the training data (50000 data, 49000 for training, 1000 for validation) and parameter setting:

$$lambda = 0, learning\_rate = 0.001, batch\_size = 25$$

The results are:

$$Train\_accuracy = 0.41576 \quad Test\_accuracy = 0.3987$$

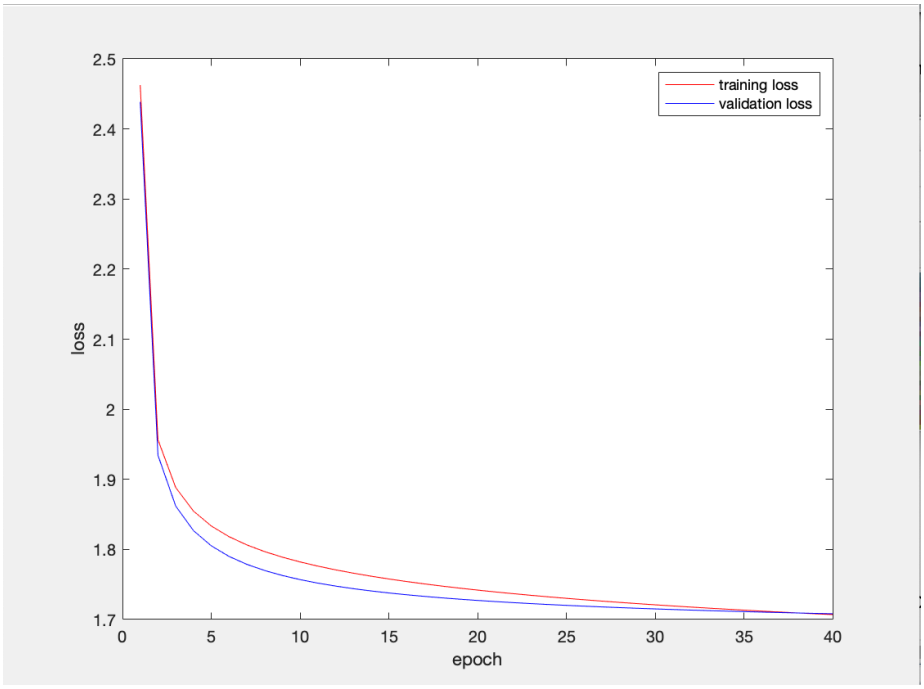

Figure 1: Train/validation loss of full data

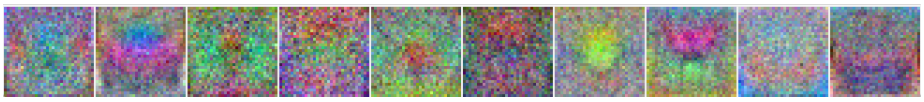

Figure 2: The learnt W matrix of full data

1.3 Train for a longer time

In this part, I increased n_epoch to 100 to train for a longer time.

parameter setting:

$$lambda = 0, learning\_rate = 0.001, batch\_size = 25$$

By checking the figure of loss functions, we can find that the loss function of validation set becomes smooth at the end and it seems that we shouldn't increase the number of epochs anymore because it may cause over-fitting.

The results are:

$$Train\_accuracy = 0.4394 \quad Test\_accuracy = 0.386$$

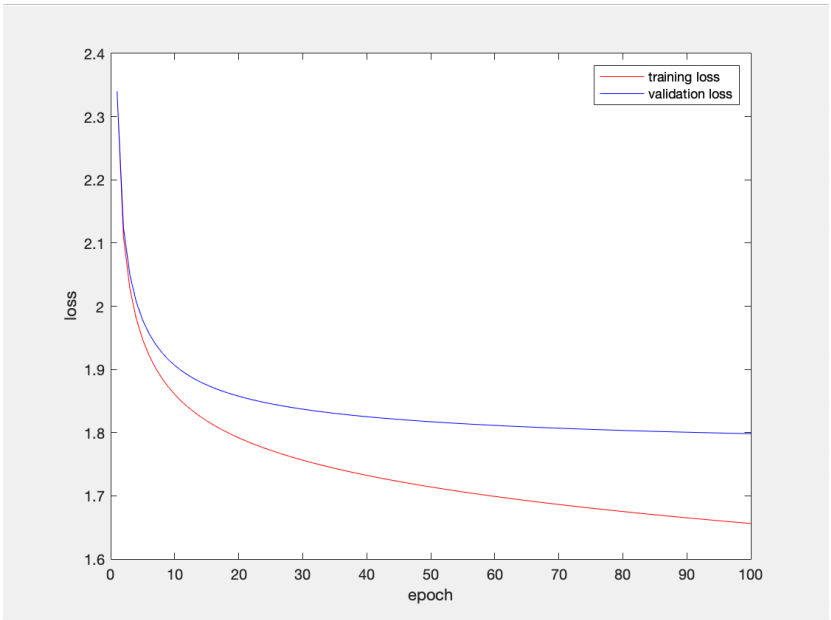

Figure 3: Train/validation loss of longer training time

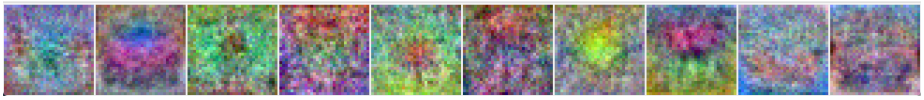

Figure 4: The learnt W matrix of longer training time

1.4 Learning rate decay

By decaying the learning rate by a factor 0.99 after every 10 epoch, I achieved a better performance on the test set.

parameter setting:

$$lambda = 0, learning\_rate = 0.001, batch\_size = 25, decay\_rate = 0.99$$

The results are:

$$Train\_accuracy = 0.4087 \quad Test\_accuracy = 0.3791$$

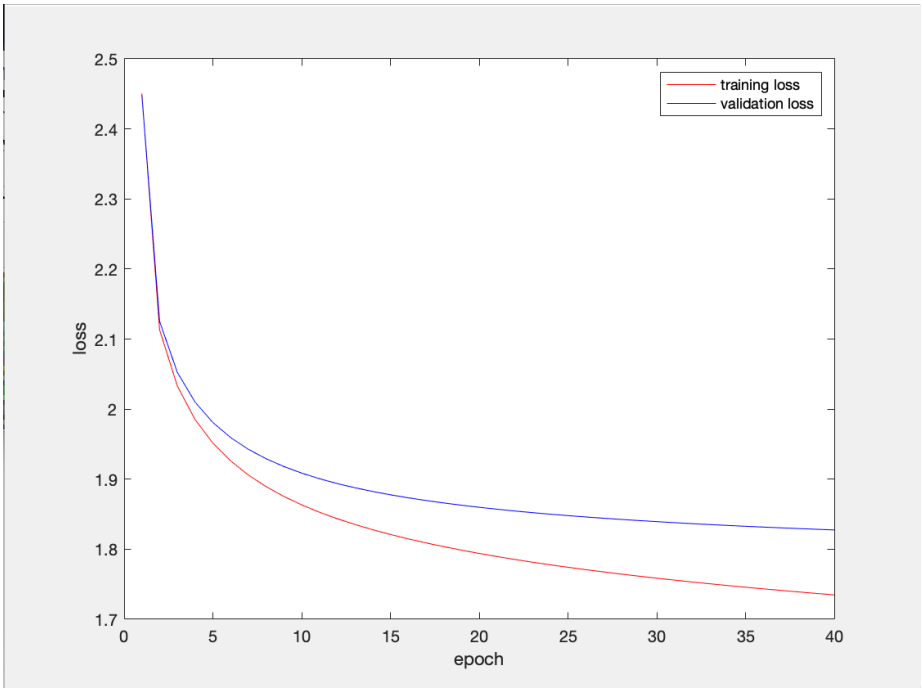

Figure 5: Train/validation loss of learning rate decay

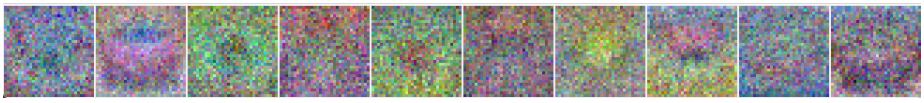

Figure 6: The learnt W matrix of learning rate decay

1.5 Combining all methods

By combining all the methods mentioned above, I achieved the best test accuracy of 0.41

parameter setting:

$lambda = 0, learning\_rate = 0.001, batch\_size = 25, decay\_rate = 0.99/10\ epochs$

$$n\_epoch = 100$$

The results are:

$$Train\_accuracy = 0.43294 \quad Test\_accuracy = 0.409$$

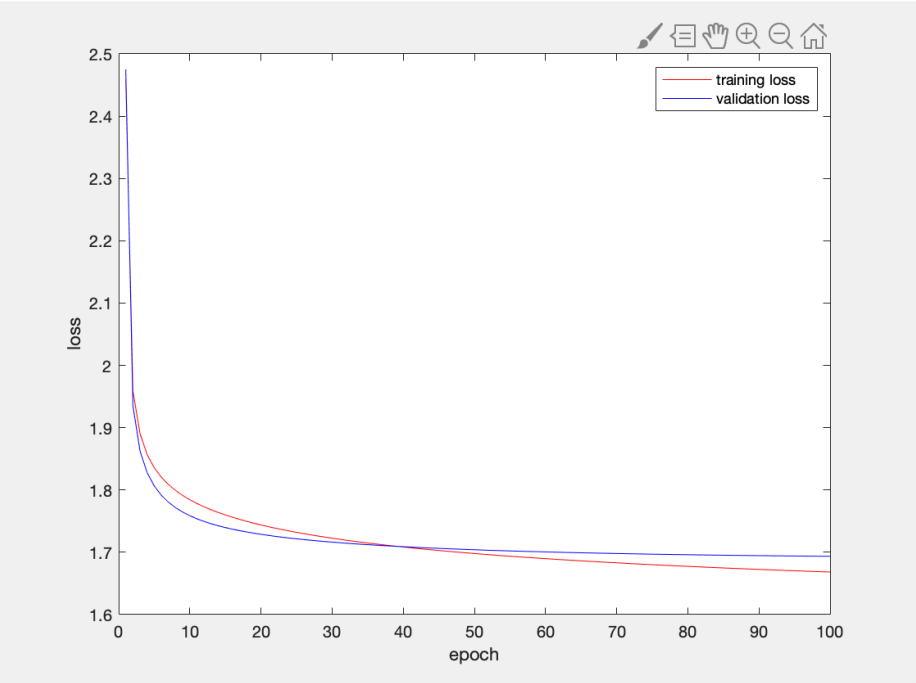

Figure 7: Train/validation loss of all methods combined

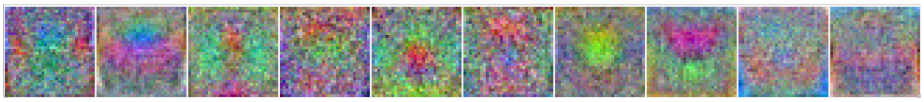

Figure 8: The learnt W matrix of all methods combined

2 SVM-loss

2.1 Theory

To use SVM-loss as the cost function, we need to figure out the gradient of it to perform GD numerically. The gradient of SVM-loss can be expressed by:

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

Figure 9: SVM-loss gradient

So I calculated gradients as follows:

```
function [grad_W, grad_b] = ComputeGradientsSVM(X, y, S, W, lambda)
    n = size(X, 2);
    d = size(X, 1);
    K = size(S, 1);
    grad_W = zeros(K, d);
    grad_b = zeros(K, 1);
    for i = 1 : n
        %s_j - s_y + 1
        S(:, i) = S(:, i) - S(y(i) + 1, i) + 1;
        % indicator function
        S(:, i) = S(:, i) > 0;
        % -sum of indicator for j=y
        S(y(i) + 1, i) = S(y(i) + 1, i) - sum(S(:, i));
        grad_W = grad_W + S(:, i) * X(:, i)';
        % equivalent to X(:, i) = 1 for grad_b
        grad_b = grad_b + S(:, i);
    end
    grad_W = grad_W / n + 2 * lambda * W;
    grad_b = grad_b / n;
end
```

2.2 Results

parameter setting:

$$lambda = 0, learning\_rate = 0.01, batch\_size = 100, n\_epoch = 40$$

The results are:

$$Train\_accuracy = 0.3913 \quad Test\_accuracy = 0.335$$

While the classifier trained with cross-entropy loss and the same parameters has:

$$Train\_accuracy = 0.4177 \quad Test\_accuracy = 0.3685$$

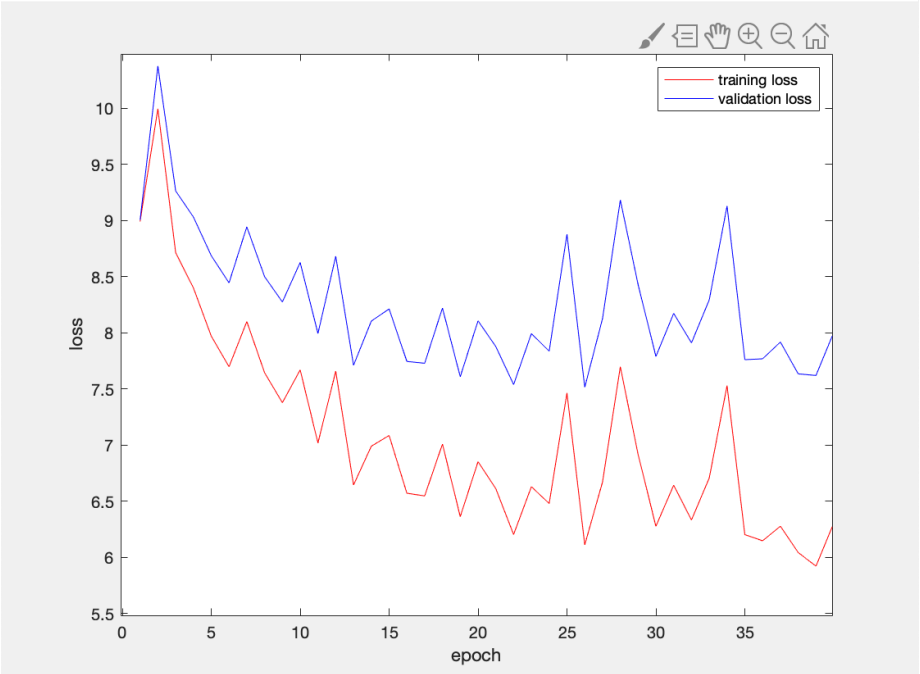

Figure 10: Train/validation loss of SVM 1

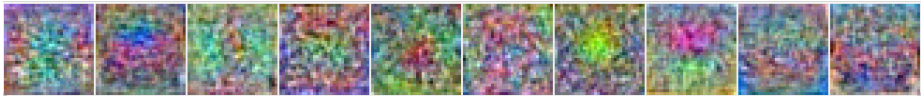

Figure 11: The learnt W matrix of SVM 1

parameter setting:

$$lambda = 0.1, learning\_rate = 0.01, batch\_size = 100, n\_epoch = 40$$

The results are:

$$Train\_accuracy = 0.3294 \quad Test\_accuracy = 0.305$$

While the classifier trained with cross-entropy loss and the same parameters has:

$$Train\_accuracy = 0.3419 \quad Test\_accuracy = 0.3338$$

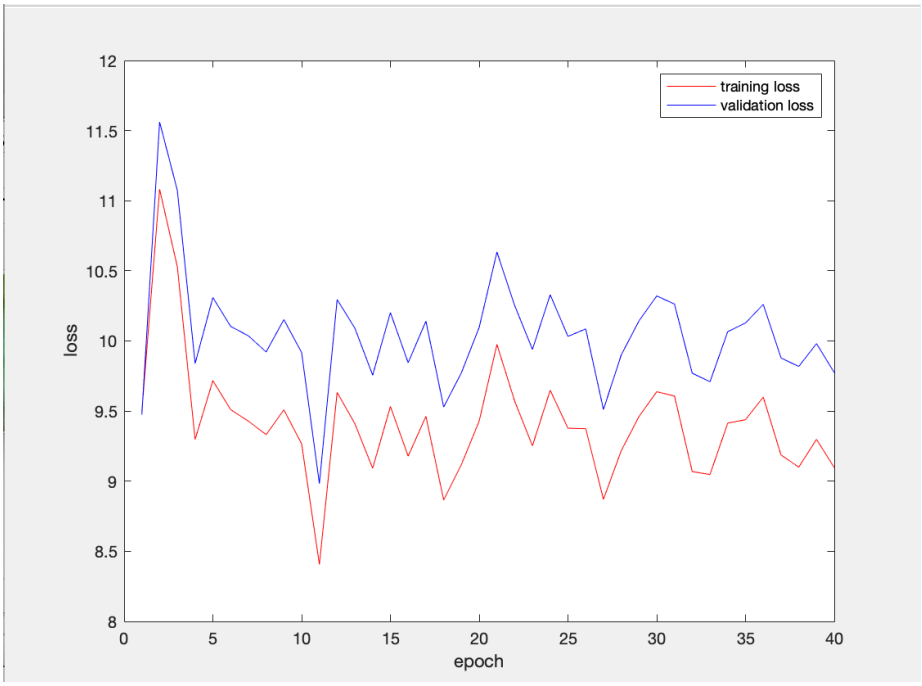

Figure 12: Train/validation loss of SVM 2

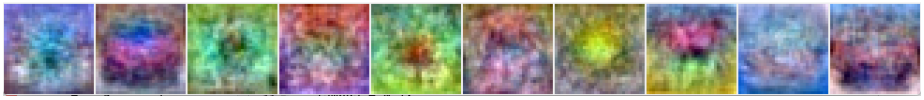

Figure 13: The learnt W matrix of SVM 2

parameter setting:

$$lambda = 0, learning\_rate = 0.001, batch\_size = 100, n\_epoch = 40$$

The results are:

$$Train\_accuracy = 0.3982 \quad Test\_accuracy = 0.3563$$

While the classifier trained with cross-entropy loss and the same parameters has:

$$Train\_accuracy = 0.378 \quad Test\_accuracy = 0.3589$$

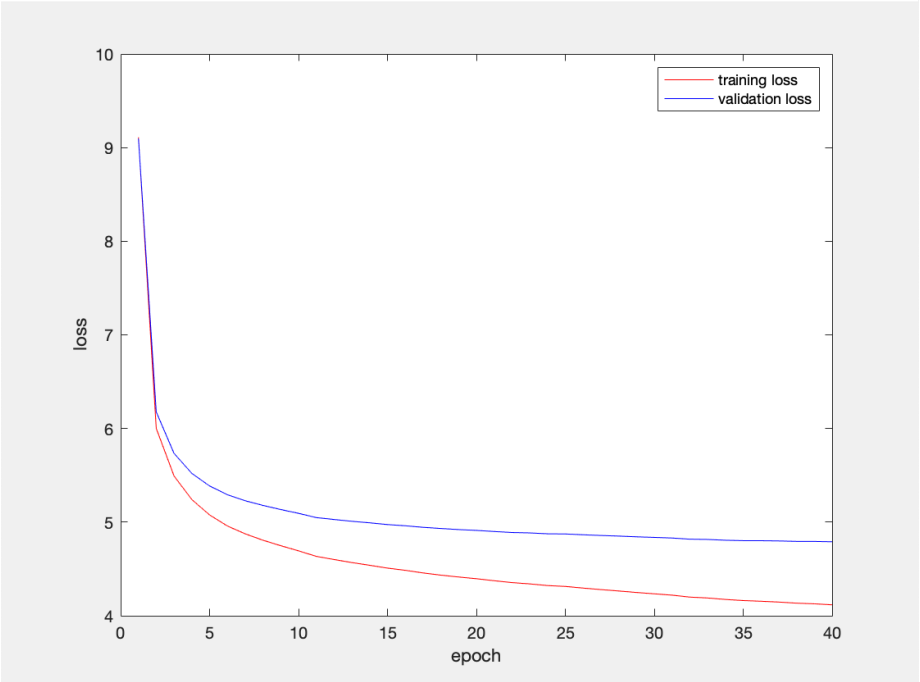

Figure 14: Train/validation loss of SVM 3

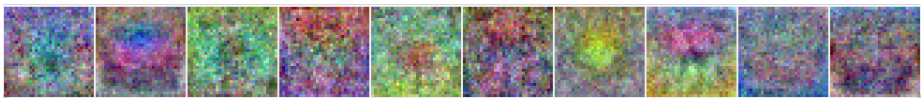

Figure 15: The learnt W matrix of SVM 3

parameter setting:

$$lambda = 0.1, learning\_rate = 0.001, batch\_size = 100, n\_epoch = 40$$

The results are:

$$Train\_accuracy = 0.3808 \quad Test\_accuracy = 0.3464$$

While the classifier trained with cross-entropy loss and the same parameters has:

$$Train\_accuracy = 0.3721 \quad Test\_accuracy = 0.3537$$

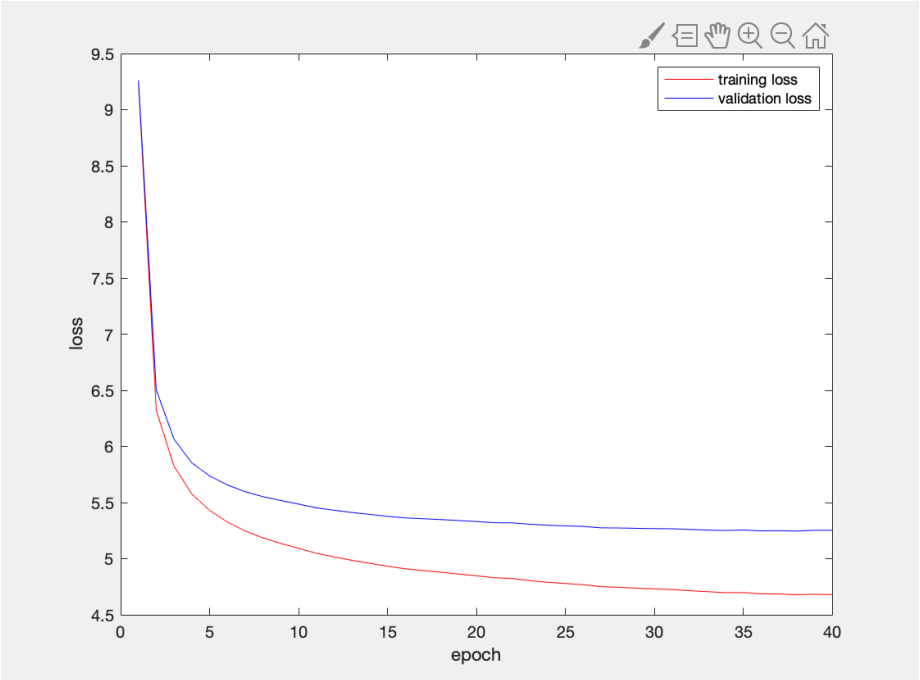

Figure 16: Train/validation loss of SVM 4

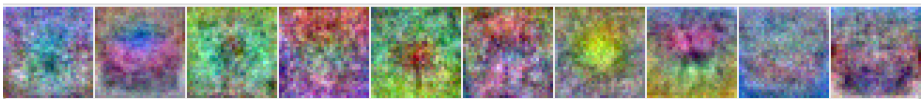

Figure 17: The learnt W matrix of SVM 4

2.3 Conclusions

When the learning rate is relatively large (about 0.01), classifier with SVM-loss is unstable when training, thus its results on the test set is worse than classifier with cross entropy loss. This may due to SVM-loss is highly sensitive to large changes in W and b. As I decreased the learning rate, the performance improved, but accuracy on test set is still lower than the accuracy of cross entropy loss. So at least in my 4 trials, the classifier with SVM-loss is worse.
Also, the training process of classifier with SVM-loss is slower.