Deep Learning in Data Science (DD2424) Report to Assignment 1

Cong Gao

March 24, 2020

1 Optional part

1.1 Optimize the performance of the network

I tried the following methods to improve the classification performance.
a) Use all the available training data for training (all five batches minus a small subset of the training images for a validation set) and decrease the size of validation set down to 1000, then I got the following result.
b) Play around with decaying the learning rate by a factor of 0.95 after each epoch.
c) Shuffle the order of the training examples at the beginning of each epoch.
d) Initialize wights with Xavier initialization.
e) Do a grid search to find good values for the amount of regularization, the learning rate and the batch size. $(\lambda = 0.04, epochs = 80, batchsize = 100, \eta = 0.001)$
f) Train for a longer time and use your validation set to make sure you don't overfit.
The graph of the total loss and the cost function on the training data and the validation data after each epoch of the mini-batch gradient descent algorithm is as follows.

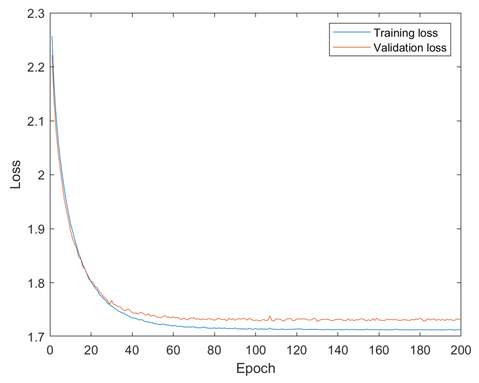


Figure 1: Graph of total loss after each epoch

Image representing the learnt weight matrix after the completion of training is as follows.

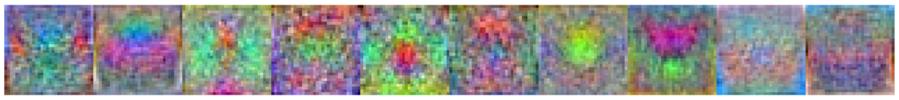


Figure 2: The learnt weight matrix

The classification accuracy on training set, validation set, and test set is as follows.

	Training data	Validation data	Test data
Accuracy	43.81%	41.09%	42.02%

Table 1: Classification Accuracy

The best test accuracy is 42.02%.
After applying the methods above, the classification accuracy on test data was enhanced by about 2.9% comparing to the result of setting 3 (which got the best performance among all the four settings). Among the six improving methods, method a) brought the largest gain.

1.2 Train network by minimizing the SVM multi-class loss

The SVM multi-class loss function is as follows:

$$l_{SVM} = \frac{1}{D} \sum_{(x,y)\in D} \sum_{j=1,j\neq y}^{10} \max(0, 1 - s_y + s_j) + \lambda \sum_{i,j} W_{i,j^2}$$

where $s_j = W_j^T x + b_j$. Then I solve this optimization problem via mini-batch gradient descent:

$$W^{t+1} = W^t - \eta \frac{\partial J}{\partial W^t}$$

$$b^{t+1} = b^t - \eta \frac{\partial J}{\partial b^t}$$

And for a single training example (x_i, y_i), the gradients of the SVM multi-class loss function w.r.t. weights and bias are shown as below.

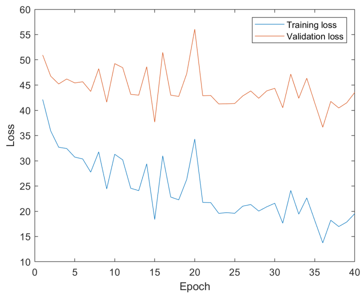
$$\frac{\partial J}{\partial W_j} = -(\sum_{j\neq y_i} ind(s_j - s_y + 1 > 0))x_i, \;\; if \;\; j = y_i$$

$$\frac{\partial J}{\partial W_j} = ind(s_j - s_y + 1 > 0)x_i, \;\; if \;\; j \neq y_i$$

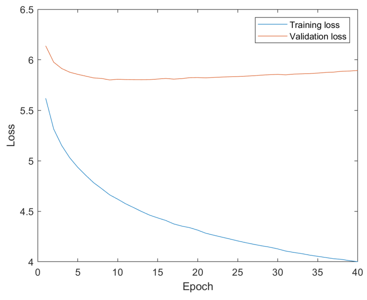
$$\frac{\partial J}{\partial b_j} = -(\sum_{j\neq y_i} ind(s_j - s_y + 1 > 0)), \;\; if \;\; j = y_i$$

$$\frac{\partial J}{\partial b_j} = ind(s_j - s_y + 1 > 0), \;\; if \;\; j \neq y_i$$

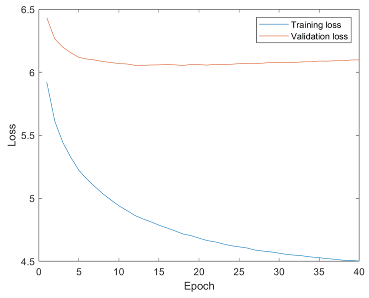
Where j is the row number of the weight matrix.
Then I added the gradients of all the training examples to the gradient of regularization term and got the final gradient.
I also tested the classification performance under the following settings which are the same as those in the basic part.
–Setting 1: $\lambda = 0, epochs = 40, batch = 100, \eta = 0.1$
–Setting 2: $\lambda = 0, epochs = 40, batch = 100, \eta = 0.001$
–Setting 3: $\lambda = 0.1, epochs = 40, batch = 100, \eta = 0.001$
–Setting 4: $\lambda = 1, epochs = 40, batch = 100, \eta = 0.001$
And I used the data in the file $data\_batch\_1.mat$ for teraining, the file $data\_batch\_2.mat$ for validation and the file $test\_batch.mat$ for test.
The graphs of the total loss and the cost function on the training data and the validation data after each epoch of the mini-batch gradient descent algorithm are as follow.



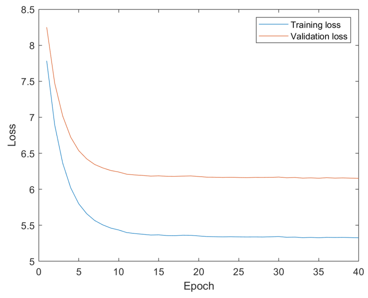
(a) Graph of total loss for setting 1



(b) Graph of total loss for setting 2



(c) Graph of total loss for setting 3



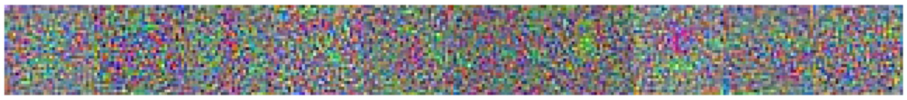
(d) Graph of total loss for setting 4

The classification accuracy on training set, validation set, and test set is as follows.

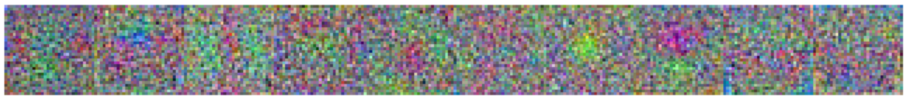
	Training data	Validation data	Test data
Setting 1	37.01%	26.18%	25.58%
Setting 2	47.88%	34.73%	34.75%
Setting 3	47.29%	36.23%	36.33%
Setting 4	41.65%	35.94%	36.57%

Table 2: Classification Accuracy

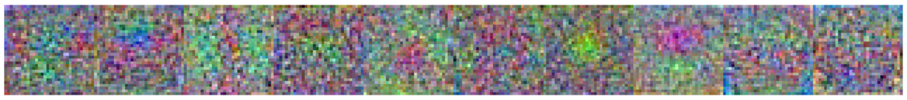
Images representing the learnt weight matrix after the completion of training are as follow.



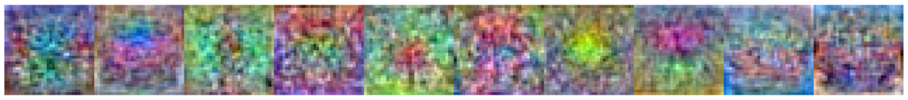
(a) The learnt weight matrix for setting 1



(b) The learnt weight matrix for setting 2



(c) The learnt weight matrix for setting 3



(d) The learnt weight matrix for setting 4

The conclusions about the effect of the regularization and the effect of learning rate are the same as before. The classification accuracy when using SVM multi-class loss function is worse than that when using cross-entropy loss function.

Then I also applied six improving methods which are the same as before.

a) Use all the available training data for training (all five batches minus a small subset of the training images for a validation set) and decrease the size of validation set down to 1000, then I got the following result.

b) Play around with decaying the learning rate by a factor of 0.95 after each epoch.

c) Shuffle the order of the training examples at the beginning of each epoch.

d) Initialize wights with Xavier initialization.

e) Do a grid search to find good values for the amount of regularization, the learning rate and the batch size. ($\lambda = 1.1, epochs = 30, batchsize = 400, \eta = 0.001$)

f) Train for a longer time and use your validation set to make sure you don't

overfit.
The graph of the total loss and the cost function on the training data and the validation data after each epoch of the mini-batch gradient descent algorithm is as follow.

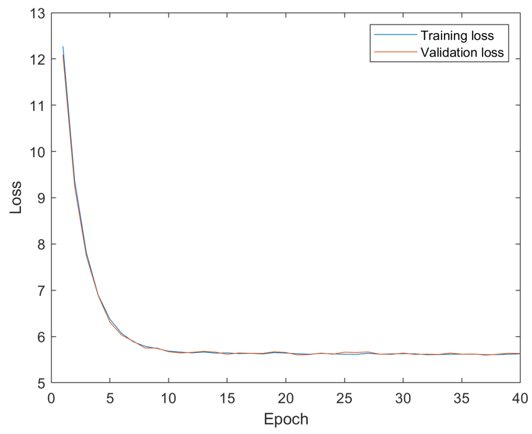


Figure 5: Graph of total loss after each epoch

Images representing the learnt weight matrix after the completion of training are as follow.

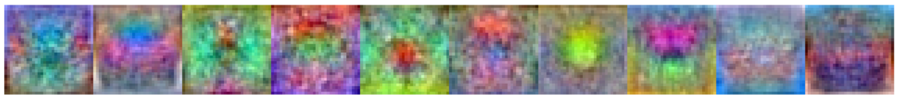


Figure 6: The learnt weight matrix

The classification accuracy on training set, validation set, and test set is as follows.

	Training data	Validation data	Test data
Accuracy	40.45%	39.90%	39.65%

Table 3: Classification Accuracy

The best test accuracy is 39.65%.