Course: DD2424 - Assignment 1
Exercise 2: Optional for Bonus Points

Álvaro Orgaz Expósito

April 14, 2019

1 Optimize the performance of the network

I will compare the results of the following tricks or modifications with the parametrization case with the best loss in the validation set, parametrization case 2, which achieves 1.822 cross entropy loss and 36.47% accuracy. In the test data, it achieves a 36.75% accuracy.

- Case 2: lambda=0, n_epochs=40, n_batch=100, eta=.01

1.1 Use all the available training data for training

Firstly, I tried to use all the available data for training (all 5 batches minus a small subset of the training images for a validation set). Decreasing the size of the validation set down to 1000 I got the learning curve of figure 1 where we can see how the performance is a bit improved. As we will see this will be the trained network with the best test accuracy, although maybe it is not a fair comparison since much more training data is used.

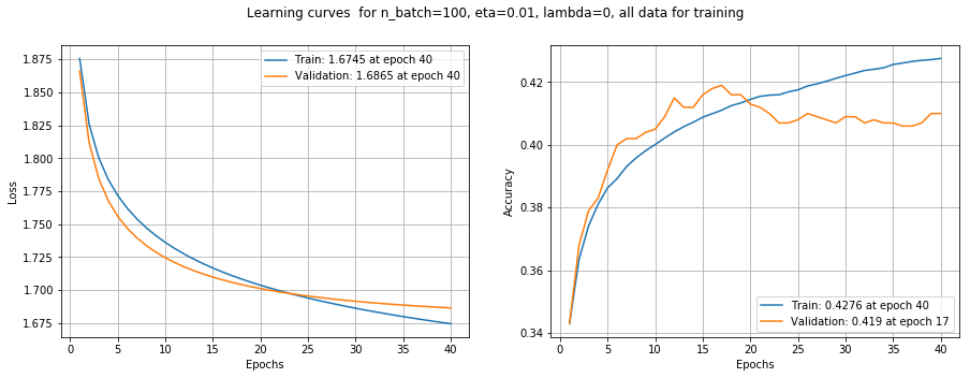

Figure 1: Learning curves for parametrization case 2 training with all 5 data batch except for 1000 images.

1.2 Train for a longer time

Secondly, I tried to train for longer (more epochs until 100) and I got the learning curve of figure 2 where we can see how the performance is almost the same (a bit of improvement).

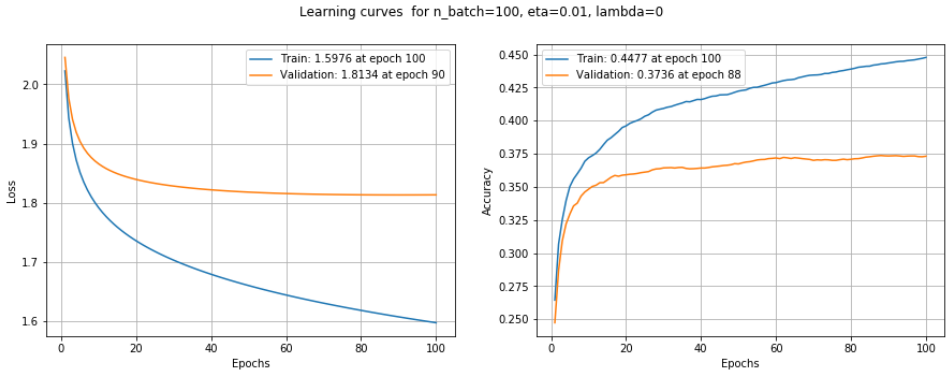

Figure 2: Learning curves for parametrization case 2 training for 100 epochs

1.3 Do a grid search to find good values

Thirdly, I tried to train many sensible parametrizations (grid search) and I got the results of training and validation loss of figure 3. I can conclude that the parametrization of case 2 is good since no significant improvement is achieved.

	lambda	eta	n_batch	loss_train	loss_val
5	0.0000	0.0100	200.0	1.718204	1.821261
4	0.0000	0.0100	100.0	1.679381	1.821987
14	0.0001	0.0100	200.0	1.719216	1.822126
13	0.0001	0.0100	100.0	1.681038	1.823257
23	0.0100	0.0100	200.0	1.791565	1.881746
0	0.0000	0.0001	10.0	1.845626	1.893115
9	0.0001	0.0001	10.0	1.846098	1.893573
22	0.0100	0.0100	100.0	1.782893	1.894891
18	0.0100	0.0001	10.0	1.887810	1.933911
3	0.0000	0.0100	10.0	1.657690	2.068721
12	0.0001	0.0100	10.0	1.676956	2.077175
1	0.0000	0.0001	100.0	2.103332	2.115676
10	0.0001	0.0001	100.0	2.103647	2.115991
19	0.0100	0.0001	100.0	2.134378	2.146685
21	0.0100	0.0100	10.0	2.023454	2.155922
2	0.0000	0.0001	200.0	2.172661	2.180113
11	0.0001	0.0001	200.0	2.172969	2.180421
20	0.0100	0.0001	200.0	2.203169	2.210610
17	0.0001	0.1000	200.0	5.558980	6.054660
7	0.0000	0.1000	100.0	6.538373	7.253700
16	0.0001	0.1000	100.0	8.324900	9.028100
8	0.0000	0.1000	200.0	9.087156	9.492232
25	0.0100	0.1000	100.0	12.720298	13.060341
24	0.0100	0.1000	10.0	13.322783	13.719504
15	0.0001	0.1000	10.0	11.478234	13.822576
26	0.0100	0.1000	200.0	13.696139	13.971675
6	0.0000	0.1000	10.0	14.706487	17.508820

Figure 3: Grid search to find an optimal parametrization.

1.4 Play around with decaying the learning

Also, I tried to train to decay the learning rate by a factor 0.9 after each epoch. The corresponding learning curve is in figure 4 where we can see how the performance is not very improved respect to initial parametrization 2 results.

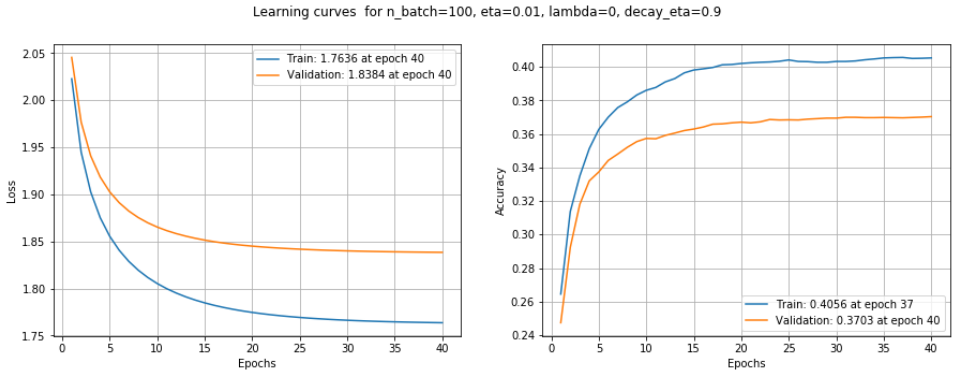

Figure 4: Learning curves for parametrization case 2 training with learning rate decay.

1.5 Implement Xavier initialization

Using Xavier initialization the resulting learning curve is in figure 5 where the performance levels are not enhanced.

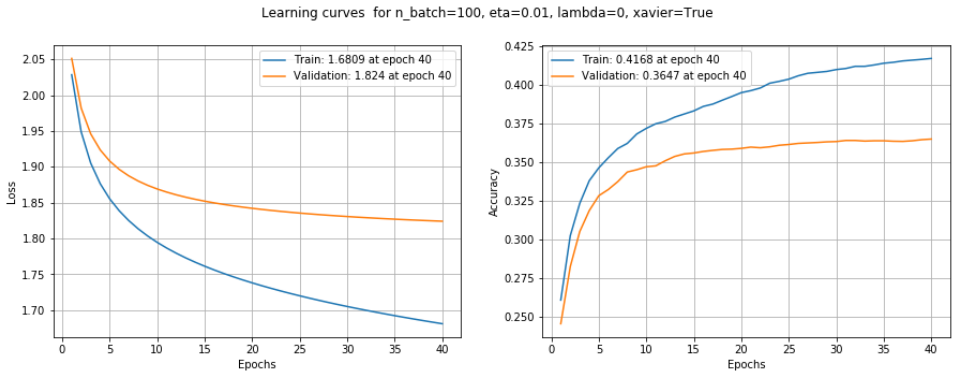

Figure 5: Learning curves for parametrization case 2 training with Xavier initialization.

1.6 Shuffle the order of your training examples by epochs

Moreover, I implemented the permutation option to train each epoch using a shuffled version of the data (batch using the mini batch version of gradient descent). This is easy to implement and common practice because random shuffling should make it harder for the network to memorize the training set, then I would expect this to improve validation error. In this case, the results are not as good and it deletes the smoothness of the curves. You can see the results in figure 6.

1.7 Apply ensemble to several networks initializations

Here I train several networks using different initializations (weights randomizations) of the network's parameters. Then apply the ensemble to a test image and use the majority vote as

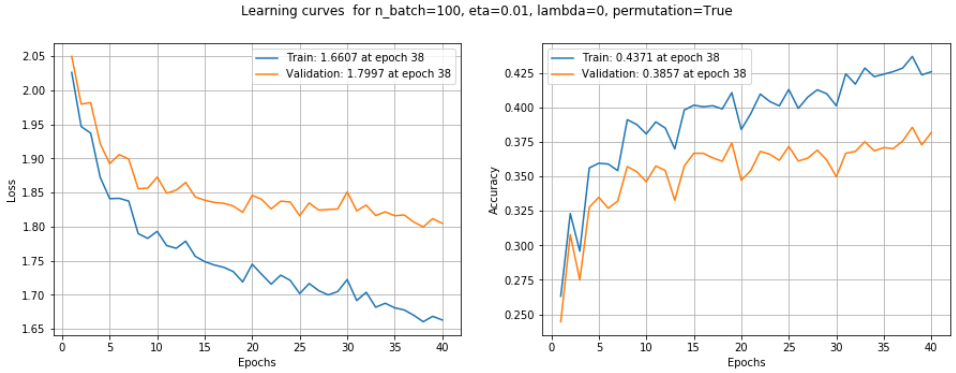

Figure 6: Learning curves for parametrization case 2 training with data permutation in each epoch.

the prediction. Finally, the validation accuracy achieved by an ensemble of size 10 networks is not an improvement of the network achieving a value of 36.36% compared to the initial case 2 validation accuracy 36.47%.

1.8 Trained network with the best test accuracy

After applying all these modifications to the best parametrization case 2, the trained network with best test accuracy is:

- With modification *training with all data* the test accuracy is: 40.16%
- With modification *training for long* the test accuracy is: 37.05%
- With modification *training with learning rate decay* the test accuracy is: 37.51%
- With modification *training with Xavier initialization* the test accuracy is: 36.62%
- With modification *training with data permutation* the test accuracy is: 38.14%
- With modification *ensembling* the test accuracy is: 36.9%

2 Train network by minimizing the SVM multi-class loss

Now, the loss function to minimise respect to weights has changed to:

$$J = L + \lambda||W||^2 = \frac{1}{N}\sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq y^i}}^{C} max(0, s_{ji} - s_{y^i i} + \delta) + \lambda \sum_{c,d}^{C,D} W_{cd}^2 \tag{1}$$

where y^i is the true class of the observation i and delta is a defined margin (such as 1).

Firstly, I have developed a grid search to compare the test accuracy of SVM and cross entropy storing the test accuracy for both loss function and parametrization. We can find the table of results (ordered by the test accuracy of SVM) in the following figure 7. We can see how some good parametrizations for cross entropy does not work well for SVM and vice versa, but it seems that the optimal parametrization for SVM (highest test accuracy) gets very similar performance applied to cross entropy loss function.

	lambda	eta	n_batch	Test Accuracy CE	Test Accuracy SVM
0	0.0000	0.0001	10.0	0.3576	0.3622
9	0.0001	0.0001	10.0	0.3576	0.3620
18	0.0100	0.0001	10.0	0.3566	0.3619
19	0.0100	0.0001	100.0	0.2782	0.3486
1	0.0000	0.0001	100.0	0.2776	0.3485
10	0.0001	0.0001	100.0	0.2777	0.3485
2	0.0000	0.0001	200.0	0.2318	0.3390
20	0.0100	0.0001	200.0	0.2317	0.3390
11	0.0001	0.0001	200.0	0.2318	0.3389
13	0.0001	0.0100	100.0	0.3674	0.3174
4	0.0000	0.0100	100.0	0.3675	0.3075
3	0.0000	0.0100	10.0	0.3287	0.2884
12	0.0001	0.0100	10.0	0.3292	0.2851
21	0.0100	0.0100	10.0	0.3121	0.2813
22	0.0100	0.0100	100.0	0.3637	0.2811
7	0.0000	0.1000	100.0	0.2682	0.2775
14	0.0001	0.0100	200.0	0.3733	0.2744
16	0.0001	0.1000	100.0	0.2087	0.2744
15	0.0001	0.1000	10.0	0.2916	0.2704
8	0.0000	0.1000	200.0	0.2644	0.2691
6	0.0000	0.1000	10.0	0.2609	0.2675
5	0.0000	0.0100	200.0	0.3734	0.2647
17	0.0001	0.1000	200.0	0.2811	0.2579
23	0.0100	0.0100	200.0	0.3723	0.2500
26	0.0100	0.1000	200.0	0.1875	0.2317
25	0.0100	0.1000	100.0	0.1246	0.1975
24	0.0100	0.1000	10.0	0.1872	0.1831

Figure 7: Grid search to comparing test accuracy of SVM and cross entropy.

Now choosing the optimal parametrization of SVM case in terms of test accuracy, I will plot the learning curves as well as the confusion matrix and accuracy in the test set. In this case, I can conclude that using SVM function it is possible to achieve a very similar performance in the *batch test* with respect to the initial parametrization case 2 with cross entropy.

- Optimal SVM: lambda=0, n_epochs=40, n_batch=10, eta=.0001

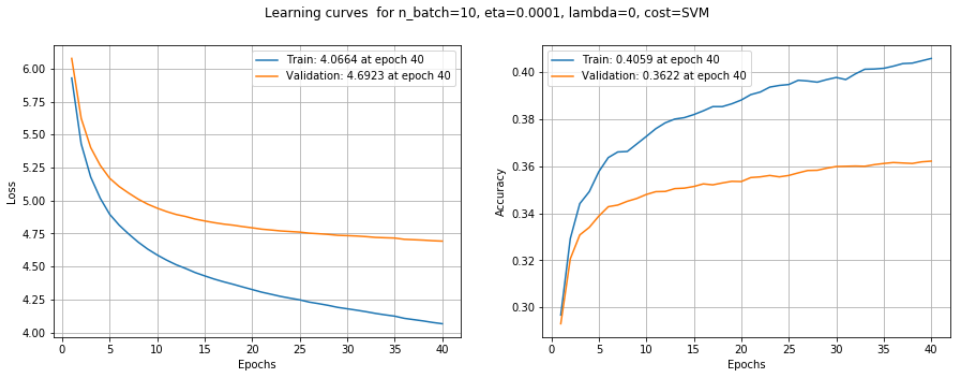

Figure 8: Learning curves for SVM optimal parametrization.

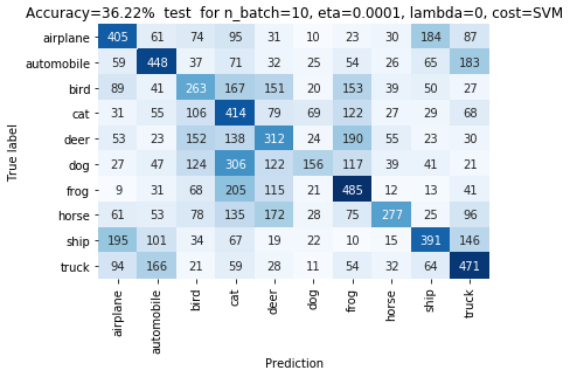

Figure 9: Test accuracy and confusion matrix for SVM optimal parametrization.