Assignment 1

Lucas Rodés Guirao
lucasrg@kth.se

I. Introduction

In this paper I show my results for the first assingment and briefly discuss the approaches that I have taken.

In this assignment we work with a single layer neural network for image classification. In particular we work with the CIFAR-10 dataset, which contains 10 different classes. To accomplish this, we train a neural network using the forward-backward pass approach, trying to minimize the cost function defined in the lectures.

Once the network has been trained, we evaluate its performance obtaining the accuracy on a test set.

A. Organization

Section II presents the results from the first exercise. Section III explains which optimization techniques have been implemented to improve the performance of the network obtained in exercise 1.

B. Other files

Please take a look at the README file to get an overview of all the files provided in this directory.

II. Exercise 1

In this exercise, first several MATLAB functions needed to be implemented. These are the building blocks of the Mini-Batch Gradient Descent algorithm implementation.

Key aspect here was to correctly implement the Gradient analytical expression. For this, I used the *Gradient check* approach, comparing the results from the implemented analytical expression and the results obtained from a numerical implementation based on the *cen-tered difference* formula. Once this test was succesfully passed, I implemented the Mini-Batch Gradient descent with batches of 100 samples.

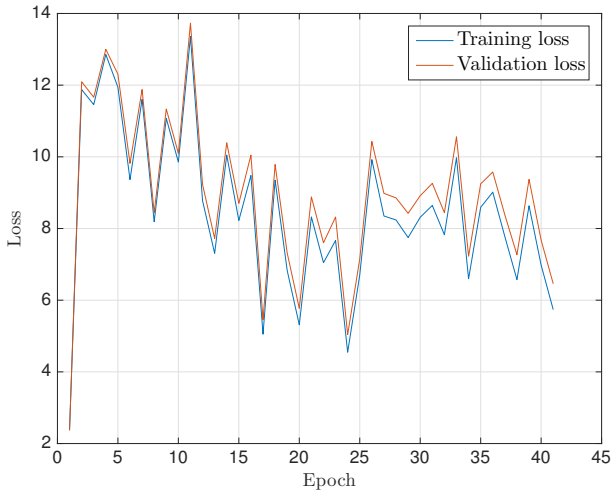
Table I shows the accuracies obtained for different parameter settings of the network. Namely we tried the settings proposed in the lab tutorial (learning rate `lambda`, number of epochs `n_epochs`, batch size `n_batch` and learning rate `eta`).

Examiner: Prof. Josephine Sullivan, DD2424 Deep Learning in Data Science, KTH Royal Institute, HT 2017.

TABLE I: Accuracy obtained for different parameter settings.

lambda	n_epochs	n_batch	eta	Accuracy
0	40	100	0.1	22.55 %
0	40	100	0.01	36.65 %
0.1	40	100	0.01	33.37 %
1	40	100	0.01	21.92 %

Furthermore, Figs 1-4 illustrate the corresponding loss curve for the different parameter settings both on the training and validation sets.



(a) Loss curve (Valdiation and Training sets)

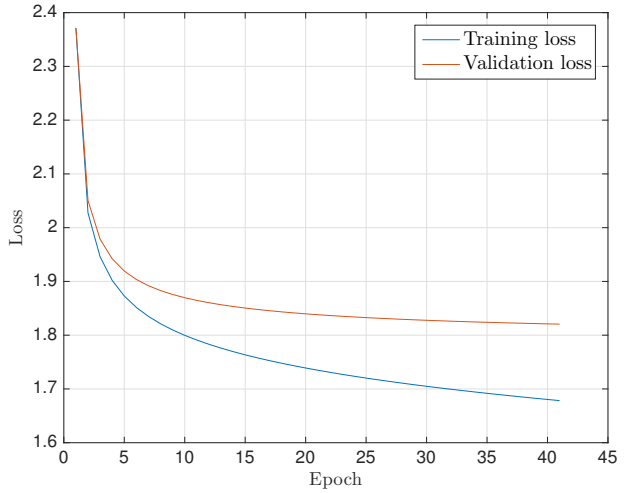
(b) Learnt weight vectors for each class

Fig. 1: `lambda` = 0, `n_epochs` = 40, `n_batch` = 100, `eta` = 0.1

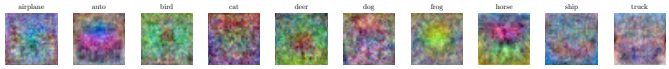
A. Discussion

In Fig. 1 we observe that the cost curve is very noisy. This is mainly due to the too high learning rate value, which prevents the gradient descent algorithm to converge to a local minima. Thus, we should use a lower learning rate. Note, however, that too low learning rate will lead to slow convergence (time consuming). The accuracy obtained for this setting is of 22.55%.

Fig. 2 illustrates the best result obtained from this four settings with an accuracy of 36.65%. This has obtained repeating exactly the same parameter setting as before but decreasing the learning rate by a factor of 10.

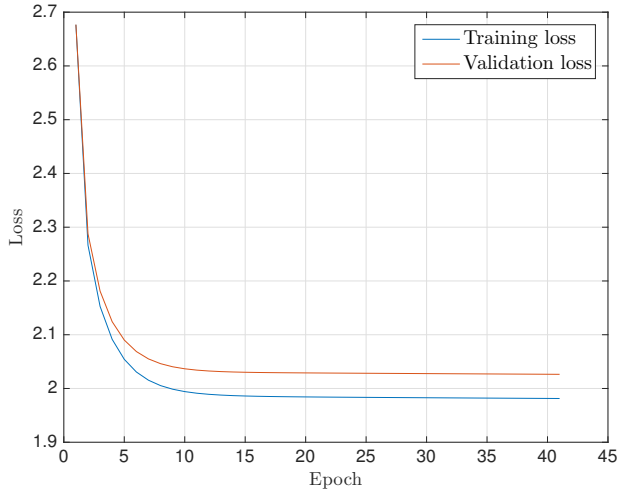


(a) Loss curve (Valdiation and Training sets)



(b) Learnt weight vectors for each class

Fig. 2: `lambda` = 0, `n_epochs` = 40, `n_batch` = 100, `eta` = 0.01

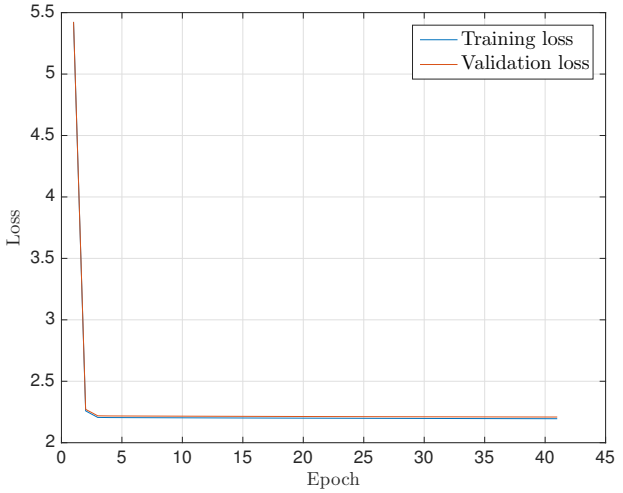


(a) Loss curve (Valdiation and Training sets)

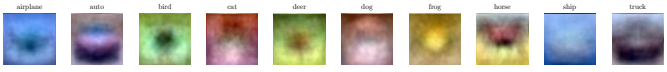
(b) Learnt weight vectors for each class

Fig. 3: `lambda` = 0.1, `n_epochs` = 40, `n_batch` = 100, `eta` = 0.01

Finally, Fig 3 and Fig 4 illustrate the effect of increasing the strength of the regularization term. In Fig 3 we note that the cost curves on both the validation and training sets have come close to each other. However the accuracy has decreased to 33.37%. A reason for this might be that we are underfitting, that is we are trying to fit a too simplistic model. In the last figure we observe that the cost evaluation in the validation and



(a) Loss curve (Valdiation and Training sets)



(b) Learnt weight vectors for each class

Fig. 4: `lambda` = 1, `n_epochs` = 40, `n_batch` = 100, `eta` = 0.01

training set is almost identical, but accuracy is now 21.92%. From this we learn that while having similar loss in the training and validation set might be an indicator of not-overfitting, it does not necessarily mean that we are performing better. In particular, we can underfit!

III. Exercise 2

In this exercise, we will first try to optimize the results obtained in the first exercise. Next, I will implement the same network but with the SVM multi-class loss.

A. Optimization of Exercise 1

I have tried the following optimization methods:

1. Permute order of samples
2. Center the data
3. Increase training set (decrease validation set)
4. Decrease factor for the learning rate
5. Add noise to training samples
6. Change batch size

Some of the listed methods were proven to increase the performance. However, specific combination of all of those had to be obtained experimentally.

In this regard, I took the parameter setting that performed the best in exercise 1, that is `lambda` = 0, `n_epochs` = 40, `n_batch` = 100, `eta` = 0.01, and tried to boost its performance.

A very simple yet powerfull improvement is to use Method 1, i.e. randomize the order of the data samples for each epoch. Doing so, we avoid using the exact same batches in

the same exact order at each epoch and thus improve our chances of better generalization. With this, we achieve an accuracy of 38.08 %.

Next, we center all the sets using the mean of the training set (Method 2). This simple change leads to an increase of the accuracy up to 39.48 %.

To improve the generalization a very intuitive approach is to increase the size of the training set (Method 3). In this regard, we use 19000 samples for the training set, leaving just 1000 for the validation set. This change leads to an accuracy of 40.23 %.

After some trial and error, I decided to keep the learning rate constant.

Next, I tried Method 5, i.e. adding noise to the training samples, since it has been reported in the literature [1] to improve the generalization. Adding zero-mean noise with standard deviation of 0.001 showed as slight increase up to 40.32 % accuracy.

Next, I tried modifying the batch size to 50, but did not improve the accuracy.

Ensemble method

B. SVM multi-class loss

REFERENCES

[1] Guozhong An. The effects of adding noise during backpropagation training on a generalization performance. *Neural computation*, 8(3):643–674, 1996.