Deep Learning in Data Science, DD2424

Short report on assignment 1

Image classification with a one-layer network

Côme Lassarat

April 4, 2022

1 Introduction

This work aims to implement a one-layer network to classify images from the CIFAR-10 dataset. The network designed will be trained using mini-batch gradient descent and tested in different scenarios by varying the learning rate and integrating L2-regularization. First, the tests ran to verify if the different functions (such as the gradients computation) are correct will be presented. In a second place, the different scenarios with relevant plots will be described.

2 Implementation checking

One of the main difficulty of this assignment is to correctly implement the analytical gradients calculation. In order to do so, a separate function were used to compute the numerical gradients. Then, the relative error (element wise) between the numerical gradients and the analytical gradients (implemented in the network) was calculated on a batch from the dataset used. Finally, the maximum of these relative errors is analyzed, for different lambda settings (and $\eta = 0.01$) (Table 1). For every experience, the starting matrices W and b are always the same.

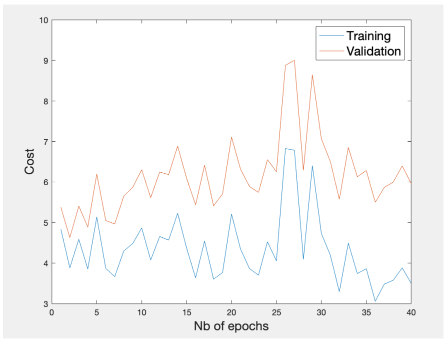
λ	Max relative error matrix W	Max relative error matrix b
0	$7.37e-9$	$4.10e-9$
0.001	$1.09e-8$	$4.10e-9$
0.01	$1.56e-8$	$4.10e-9$
0.1	$1.33e-8$	$4.10e-9$
1	$1.02e-7$	$4.22e-9$

Tabell 1: Gradient checking

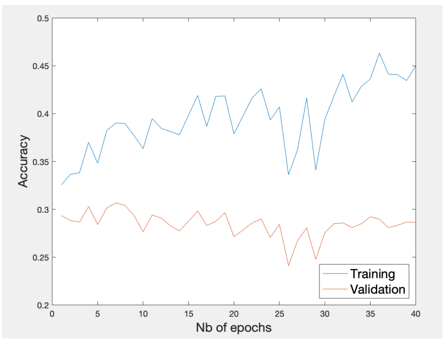
Since the gradient involved in this process are very small, the maximum relative error shouldn't be greater than $1e-7$ in order to consider our analytical expression is true, which is the case (we see an increase of the maximum relative error of matrix W to $1e-7$, but this won't be an issue since we are not likely to use bigger λ): the analytical expression of the gradient seems to be correctly implemented.

3 The different scenarios

3.1 Scenario 1: $\lambda = 0$, $n_{epochs} = 40$, $n_{batch} = 100$, $\eta = 0.1$

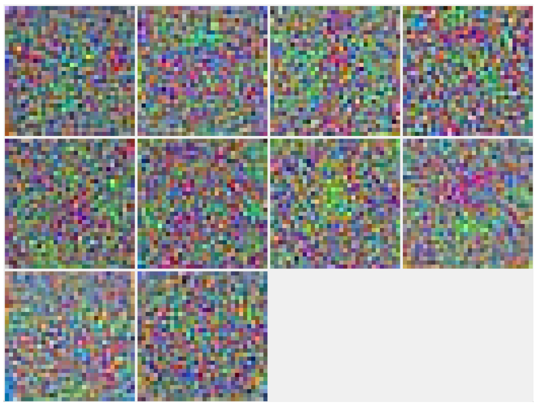


(a) Plot of the cost function (also loss here) according to the number of epochs



(b) Plot of the accuracy according to the number of epochs

Figur 1

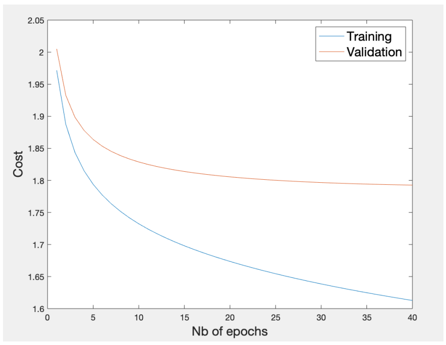


Figur 2: Representation of the matrix W

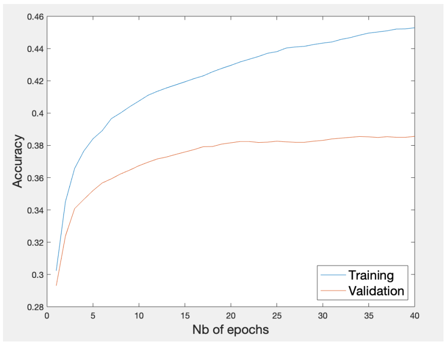
Training accuracy	Validation accuracy	Test accuracy
45.00%	28.64%	29.49%

Tabell 2: Final accuracies

3.2 Scenario 2: $\lambda = 0$, $n_{epochs} = 40$, $n_{batch} = 100$, $\eta = 0.001$

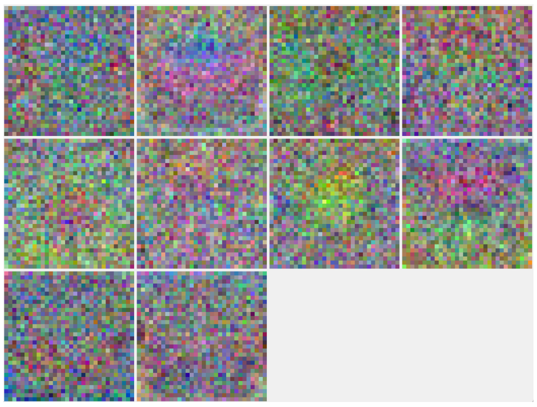


(a) Plot of the cost function (also loss here) according to the number of epochs



(b) Plot of the accuracy according to the number of epochs

Figur 3

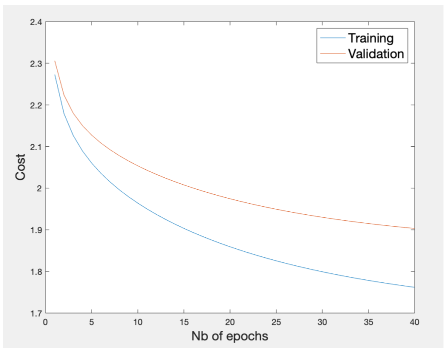


Figur 4: Representation of the matrix W

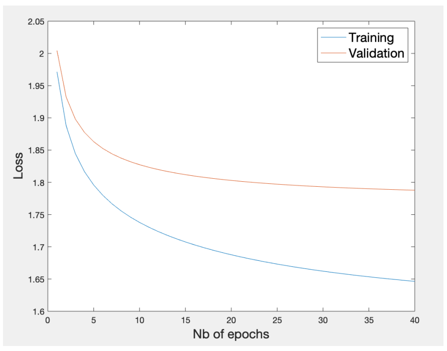
Training accuracy	Validation accuracy	Test accuracy
45.29%	38.56%	38.61%

Tabell 3: Final accuracies

3.3 Scenario 3: $\lambda = 0.1$, $n_{epochs} = 40$, $n_{batch} = 100$, $\eta = 0.001$

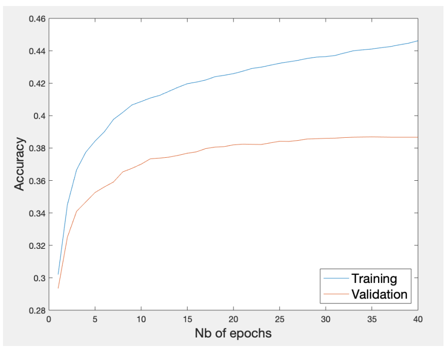


(a) Plot of the cost function according to the number of epochs

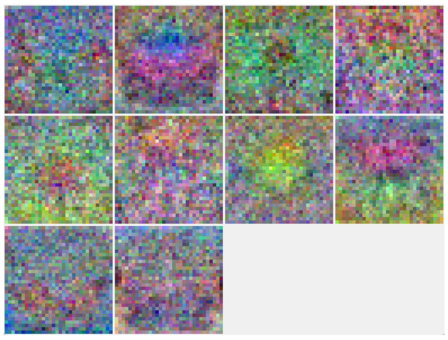


(b) Plot of the loss function according to the number of epochs

Figur 5



(a) Plot of the accuracy according to the number of epochs



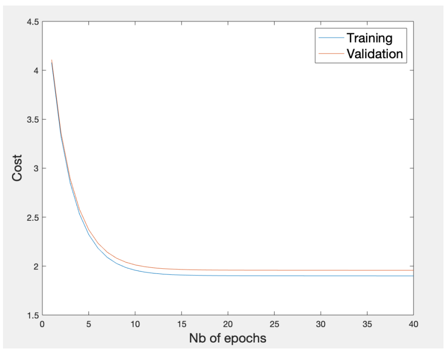
(b) Representation of the matrix W

Figur 6

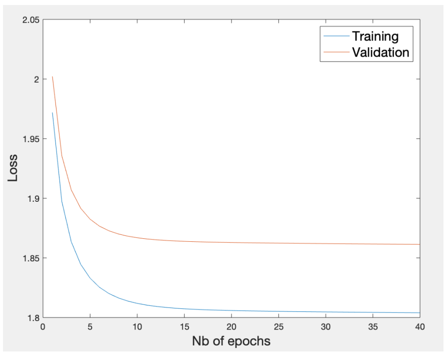
Training accuracy	Validation accuracy	Test accuracy
44.62%	38.67%	39.15%

Tabell 4: Final accuracies

3.4 Scenario 4: $\lambda = 1$, $n_{epochs} = 40$, $n_{batch} = 100$, $\eta = 0.001$

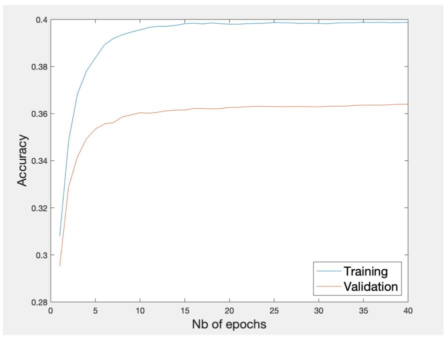


(a) Plot of the cost function according to the number of epochs

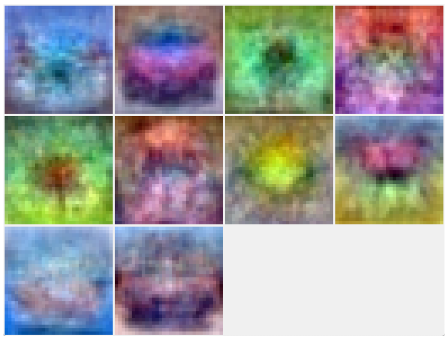


(b) Plot of the loss function according to the number of epochs

Figur 7



(a) Plot of the accuracy according to the number of epochs



(b) Representation of the matrix W

Figur 8

Training accuracy	Validation accuracy	Test accuracy
39.87%	36.40%	37.49%

Tabell 5: Final accuracies

4 Conclusion: observations and comments

4.1 General comments

From a global perspective and independently of the parameters η and λ chosen, the network implemented seems to be functional: over the epochs, the cost and loss functions decrease whereas the accuracy increases (at least on average for Figure 1): this illustrates the gradient descent process

4.2 Learning rate η

When looking at Figure 1, it appears that a high value of $\eta (= 0.1)$ (for a given value of λ) leads to an unstable network: the cost function doesn't decrease on average and the accuracy doesn't increase much. Thus, a lower value of $\eta = 0.001$ appears to give smoother convergence (Figure 3) and better accuracies (Table 3): 29.49% vs 38.61% test accuracy.

4.3 Regularization term λ

When increasing the regularization term from 0 to 0.1 (for $\eta = 0.001$ fixed), the training and validation accuracy curves get closer to each others, and the validation and test accuracy increase respectively from 38.56% and 38.61% to 38.67% and 39.15% whereas the training accuracy decreases (from 45.29% to 44.62%) : there is a better generalization. When trying higher values of λ like $\lambda = 1$, the test accuracy decreases to 37.49% and the training accuracy deceases to 39.87%: a high value of lambda leads to too much generalization and so bad learning abilities. These differences of values might not be significant (because the network only has one layer) but it illustrates the trade-off over λ: the higher λ is, the more the network generalizes (which is good until a certain point where the network doesn't learn) and the lower λ is, the less the network generalizes (and the more likely the network is to overfitt the training data).