Assignment 1 in DD2424
Deep Learning in Data Science

Marie-Ange Stefanos

mars 2022

Exercice 1

First of all, let us take a look at the training set (fig. 1) used in this section. The validation (fig. 18) and the test sets (fig. 19) can be found in appendix A.

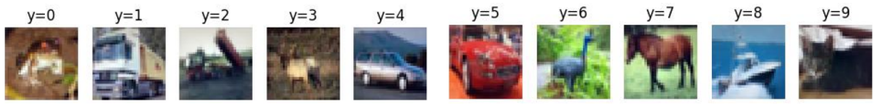

Figure 1: Training set

To begin with, one can check that the accuracy is correctly computed by taking a look at the accuracy obtained on the training set with a random weight matrix W. Whatever the initialization of W, you obtain an accuracy near 10% (often around 8 or 9%, which is coherent since there are 10 possible labels.

Before moving on to the core algorithm, you should also check that the analytical gradients of W and b that you compute (*ComputeGradients()* function) are close enough to the numerical gradients computed thanks to the given functions *ComputeGradsNum()* and *ComputeGradsNumSlow()*.

```
Checking error on gradients:

max(grad_W_err_fast) = 0.000175976
max(grad_b_err_fast) = 0.000000247
max(grad_W_err_slow) = 0.000000549
max(grad_b_err_slow) = 0.000000001
max(grad_W_err_given) = 0.000176500
max(grad_b_err_given) = 0.000000247
```

Figure 2: Relative error obtained between the analytical and the numerical (fast and slow) gradients, and between both numerical gradients

As can be seen on figure 2, three comparisons are made here (for the weight matrix W and the bias b each) using the relative error formula:

1. between the analytical gradient and the fast but less accurate numerical (finite difference formula) gradient
2. between the analytical gradient and the more accurate but slow numerical (centered difference formula) gradient
3. between both numerical given gradients (in order to have a basis of the required precision)

Figure 2 shows that the analytical gradient is as close to the fast gradient method than the two numerical method are to each other, that is less than a relative error of 1e-3% for the weight matrix and 1e-6% for the bias. Besides, the comparison of the analytical gradient with the more accurate numerical gradient shows that a relative error lower than 1e-6% for W and 1e-8% for the bias. We can conclude from those results that the analytical gradient is correctly computed and that one can use it to go further in the assignment.

Checking that the mini-batch gradient descent algorithm is correctly implemented as well, I have used it on the same training and validation sets as in the instruction file, with the same parameters. I obtained similar curves for the cost function (that equals the loss function here since lambda = 0: fig. 5), accuracy (I got 38.47%, compared to 38.83% for the validation set, see fig. 3) and final learnt weight matrix *Wstar* (fig. 4).

```
Accuracy of the training set:    45.62%
Accuracy of the validation set:  38.47%
Accuracy of the test set:        39.04%
```

Figure 3: Accuracy obtained for the training, the validation and the test sets. The network was trained with the following parameter settings: n_batch=100, eta=.001, n_epochs=40 and lambda=0.

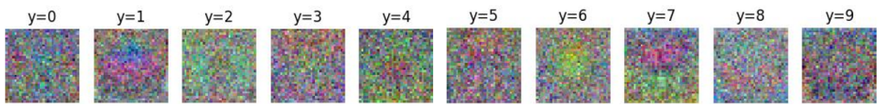

Figure 4: The learnt Wstar matrix visualized as class template images. The network was trained with the following parameter settings: n_batch=100, eta=.001, n_epochs=40 and lambda=0.

In this case, lambda=0 so the cost and the loss are the same. That is why I only plotted the cost for this example (figure 5), which will be the same for scenarii 1 and 2. We can observe that the cost of the validation set (orange curve) is always higher than the cost of the training set (blue curve), which is coherent. The plot show that both quantities decrease quite quickly through the first 5 epochs. Then, the training cost keeps decreases slower than before but still. However, the validation cost decreases as well but ever slower and almost remains almost constant, meaning that we are close to convergence.

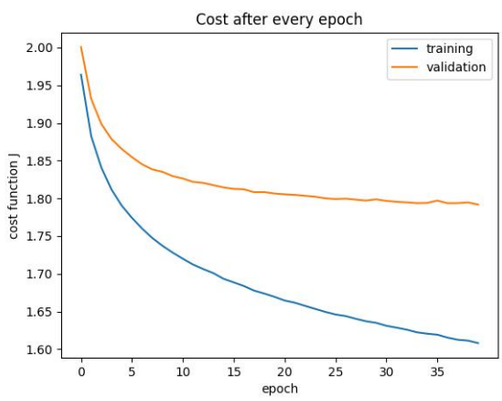

Figure 5: The graph of the training and validation cost function computed after every epoch. The network was trained with the following parameter settings: n_batch=100, eta=.001, n_epochs=40 and lambda=0.

Now that you know your network works as it should, you can begin to change the parameter settings. Here I mainly worked on the 4 following scenarii:

1. lambda=0, n_epochs=40, n_batch=100, eta=.1
2. lambda=0, n_epochs=40, n_batch=100, eta=.001
3. lambda=.1, n_epochs=40, n_batch=100, eta=.001
4. lambda=1, n_epochs=40, n_batch=100, eta=.001

For each of them, the results will be illustrated using several criteria (cost and loss functions through the epochs, accuracy of each set and the final learnt matrix Wstar) and then interpreted, in order to understand the impact of tuning each parameter independently from the others.

Scenario 1

As a reminder, the parameters used in this subsection are the following: lambda=0, n_epochs=40, n_batch=100, eta=.1, meaning that there is no regularization taken into account. Figure 6 shows that the cost oscillates and thus diverges through the epochs, whether it be for the training set (blue curve) or the validation set (orange curve), whereas one would rather to get decreasing curves, meaning increasing accuracies. However, the training set cost seems to globally decrease, which is consistent with the good accuracy on the training set of 46% as shows figure 7. Contrary to this good result, the accuracy for the validation set and for the test set are smaller (less than 30%).

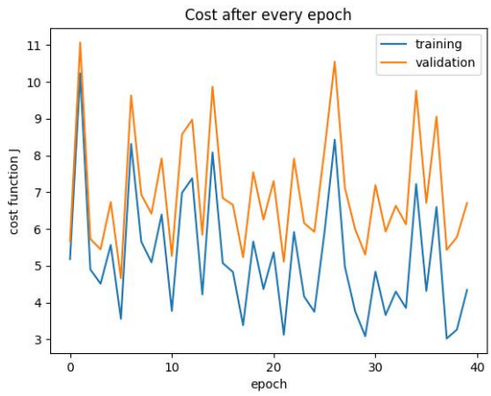

Figure 6: The graph of the training and validation cost function computed after every epoch. The network was trained with the following parameter settings: lbda=0, n_epochs=40, n_batch=100, eta=0.1 (scenario 1).

```
SCENARIO 1:

Accuracy of the training set:    43.78%
Accuracy of the validation set:  29.64%
Accuracy of the test set:        29.47%
```

Figure 7: Accuracy obtained for the training, the validation and the test sets. The network was trained with the following parameter settings: n_batch=100, eta=.001, n_epochs=40 and lambda=0 (scenario 1).

Another way to analyse the results of this scenario is to take a look at the learnt Wstar matrix of the weights that can be visualized as class template images (figure 8). Each image is quite of mix of colours, that does not seem completely random but still one cannot distinguish clearly different zones of colors.

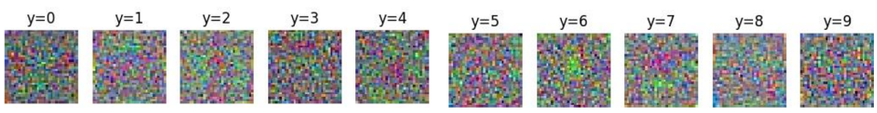

Figure 8: The learnt Wstar matrix visualized as class template images. The network was trained with the following parameter settings: lbda=0, n_epochs=40, n_batch=100, eta=0.1 (scenario 1).

Thus, these parameter do not make the algorithm converge satisfactorily. This can be due to several factors. The first one is the absence of regularization, and the second one is a learning rate that is too high, making the convergence impossible. That is the second option that will be investigated in scenario 2, by decreasing it.

Scenario 2

As a reminder, the parameters used in this subsection are the following: lambda=0, n_epochs=40, n_batch=100, eta=.001, meaning that the learning rate η has been changed from the previous scenario (from 0.1 to 0.001), meaning that there is no regularization either taken into account and that the learning rate has been reduced. Figure 9 shows that the cost does not oscillate anymore and seem to decrease through the epochs and thus converge, whether it be for the training set (blue curve) or the validation set (orange curve). This should go with a higher accuracy than previously, which can be verified on figure 10: 45% of accuracy for the training set, and more than 38% for both validation and test sets, which is almost a 10% higher accuracy than before decreasing the learning rate.

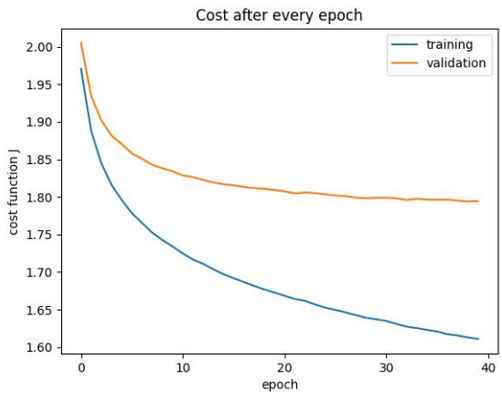

Figure 9: The graph of the training and validation cost function computed after every epoch. The network was trained with the following parameter settings: lbda=0, n_epochs=40, n_batch=100, eta=0.001 (scenario 2).

```
SCENARIO 2:

Accuracy of the training set:    45.67%
Accuracy of the validation set:  38.52%
Accuracy of the test set:        38.62%
```

Figure 10: Accuracy obtained for the training, the validation and the test sets. The network was trained with the following parameter settings: lbda=0, n_epochs=40, n_batch=100, eta=0.001 (scenario 2).

Taking a look at the learnt Wstar matrix (figure 11) enable to see that some images have some zones with colors with seem less randomly distributed than for scenario 1. For instance, label $y = 6$ shows a green zone in the center of its image. Another example is label $y = 1$, that has 2 notable zones, a blue one and a magenta one. This means that the label are better recognized now that the learning rate has been reduced, making the learning converge and thus obtain a better accuracy.

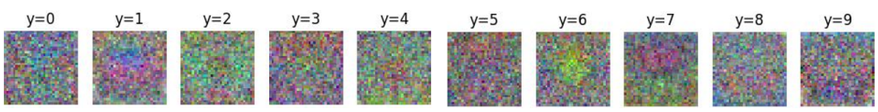

Figure 11: The learnt Wstar matrix visualized as class template images. The network was trained with the following parameter settings: lbda=0, n_epochs=40, n_batch=100, eta=0.001 (scenario 2).

Scenario 3

As a reminder, the parameters used in this subsection are the following: lambda=0.1, n_epochs=40, n_batch=100, eta=.001, meaning that the regularization coefficient λ has been changed from the previous scenario (from 0 to 0.01). This is the first scenario where regularization is taken into account. The cost function (left-hand panel of the figure 12) has a similar aspect as in previous scenario but it still decreases after 40 epochs, meaning that it eventually converges. However, the loss function (right-hand panel of the figure 12), which is different from the cost this time since there is a non zero regularization coefficient, shows lower value than the cost but still have the same evolution. The accuracy on the validation and the test sets is a little bit higher than in scenario 2 (around 39%) but it is the same order of magnitude.

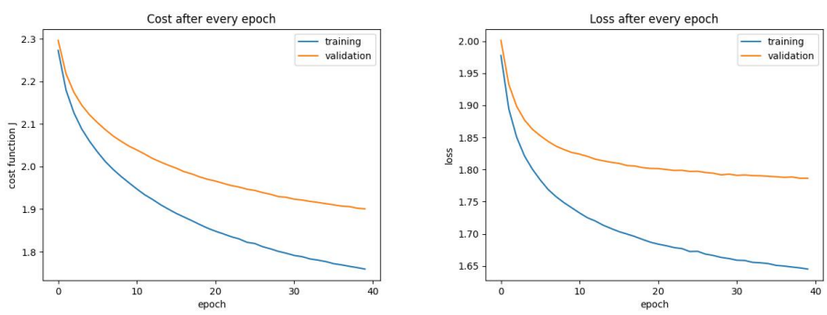

Figure 12: The graph of the training and validation cost and loss functions computed after every epoch. The network was trained with the following parameter settings: lbda=0.1, n_epochs=40, n_batch=100, eta=0.001 (scenario 3).

```
SCENARIO 3:

Accuracy of the training set:    44.76%
Accuracy of the validation set:  38.8%
Accuracy of the test set:        39.33%
```

Figure 13: Accuracy obtained for the training, the validation and the test sets. The network was trained with the following parameter settings: lbda=0.1, n_epochs=40, n_batch=100, eta=0.001 (scenario 3).

These better results can be also seen through the learnt Wstar matrix that shows every clearer colored zones on the images from the different labels.

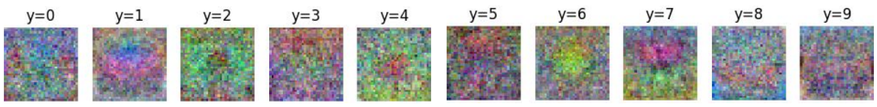

Figure 14: The learnt Wstar matrix visualized as class template images. The network was trained with the following parameter settings: lbda=0.1, n_epochs=40, n_batch=100, eta=0.001 (scenario 3).

Scenario 4

As a reminder, the parameters used in this subsection are the following: lambda=0.1, n_epochs=40, n_batch=100, eta=.001, meaning that the regularization coefficient λ has been changed from the previous scenario (from 0.1 to 1). Thus, the regularization has a higher effect, as show the curves of the cost for instance (left-hand panel of the figure 15), which are way closer than for the previous scenarii. One can also note that this is for sure the effect of regularization since loss curves (right-hand panel of the figure 15) are not that close. The accuracies (figure 16) are quite smaller than for previous scenarii though.

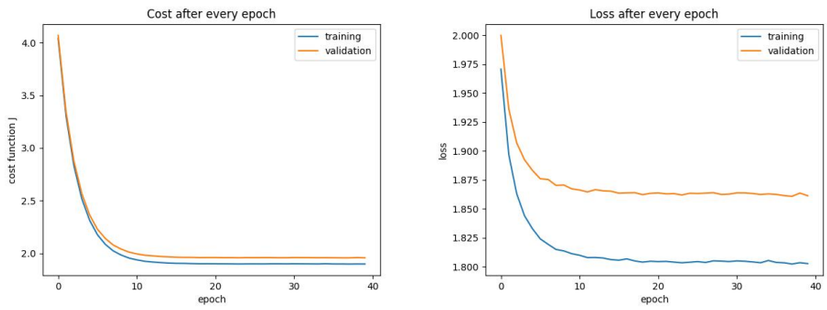

Figure 15: The graph of the training and validation cost and loss functions computed after every epoch. The network was trained with the following parameter settings: lbda=1, n_epochs=40, n_batch=100, eta=0.001 (scenario 4).

```
SCENARIO 4:

Accuracy of the training set:    40.07%
Accuracy of the validation set:  36.57%
Accuracy of the test set:        37.27%
```

Figure 16: Accuracy obtained for the training, the validation and the test sets. The network was trained with the following parameter settings: lbda=1, n_epochs=40, n_batch=100, eta=0.001 (scenario 4).

The effect of a high regularization coefficient can also be seen on the learnt Wstar matrix of weights (figure 17) where some colored shapes appear on the class template images, showing how much the network has been able to learn this time.

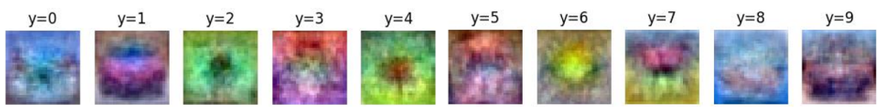

Figure 17: The learnt Wstar matrix visualized as class template images. The network was trained with the following parameter settings: lbda=1, n_epochs=40, n_batch=100, eta=0.001 (scenario 4).

Conclusion

In this lab, the effect of the different parameters (learning rate η and coefficient of regularization λ) has been studied. A learning rate that is too high will prevent the network from converging, whereas a too low learning rate will always converge but too slowly. Without regularization, the difference between the cost function for the training set and the validation set is higher than with regularization. The results have been analyzed thanks to different tools, such as the cost function. The loss function is the cost function we would obtain if we do not take into account regularization. However, since it is the cost that one want to minimize here, the lost would have evolved differently if there was no regularization. The accuracy shows the capacity of generalization of the network and the learnt Wstar matrix visualized as class template images enables to show the weights learnt by the network corresponding to each label.

Appendix A : Validation and test sets

Figure 18: Validation set

Figure 19: Test set