Advanced Machine Learning – Assignment #2

Alessio Sampieri 1765522
Michele Meo 1599032
Data Science, "Sapienza" University of Rome

Lorenzo Ceccomancini 1667798
Paolo Mandica 1898788
Due Date: November 19, 2020

Question 2

2.a

With the following demonstration we are going to verify that the loss function has gradient w.r.t. $z_i^{(3)}$ as:

$$\frac{\partial J}{\partial z_i^{(3)}} = \frac{1}{N}[-\frac{\partial log(\psi(z_i^{(3)}))}{\partial z_i^{(3)}} \frac{\partial \psi(z_i^{(3)})}{\partial z_i^{(3)}}] \tag{1}$$

it is important to notice that $z_i^{(3)}$ is a vector of K components, one component for each class of the problem, so Eq.(1) is a vector equation that contains K equations in one single expression. The sum over i in J doesn't appear because we are deriving w.r.t. the single element of the training set, i.
Now we have to evaluate, respectively, the first and the second term of Eq.(1):

$$\frac{\partial log(\psi(z_i^{(3)}))}{\partial z_i^{(3)}} = \frac{1}{\psi(z_i^{(3)})} \tag{2}$$

$$\frac{\partial \psi(z_i^{(3)})}{\partial z_i^{(3)}} = \psi(z_i^{(3)})(\Delta_{iy_i} - \psi(z_i^{(3)})) \tag{3}$$

where in Eq.(3) we have applied the softmax derivative and the Δ_{iy_i} will be a vector of K components associated to item i in training set, each component will be equal to 0, apart for the component corresponding to the true class for item i, pointed by y_i. With an abuse of notation, in the following we will write this vector as Δ_i.
Rewriting Eq.(1) using formulas obtained in Eq.(2), Eq.(3) and simplifying the expression, we verify the requested derivative:

$$\frac{\partial J}{\partial z_i^{(3)}} = \frac{1}{N}[(\psi(z_i^{(3)} - \Delta_i)] \tag{4}$$

2.b

Now we evaluate the partial derivative of the loss function w.r.t all the weights $W_{pq}^{(2)}$, expressing them and the equation related to each of them with a single expression of the matrix $W^{(2)}$

$$\frac{\partial J}{\partial W^{(2)}} = \sum_{i=1}^{N}[\frac{\partial J}{\partial z_i^{(3)}} \frac{\partial z_i^{(3)}}{\partial W^{(2)}}] \tag{5}$$

where the first term in the sum is given by Eq.(4) and the second term is $\frac{\partial z_i^{(3)}}{\partial W^{(2)}} = \frac{\partial [a_i^{(2)} W^{(2)} + b^{(2)}]}{\partial W^{(2)}} = {a_i^{(2)}}^T$.
Substituting Eq.(4) and using the last expression we wrote, we have:

$$\frac{\partial J}{\partial W^{(2)}} = \frac{1}{N}\sum_{i=1}^{N}[(\psi(z_i^{(3)}) - \Delta_i){a_i^{(2)}}^T] \tag{6}$$

Given that $\tilde{J} = J + \lambda(\|W^{(1)}\|_2^2 + \|W^{(2)}\|_2^2)$, we can easily get:

$$\frac{\partial \tilde{J}}{\partial W^{(2)}} = \frac{1}{N}\sum_{i=1}^{N}[(\psi(z_i^{(3)}) - \Delta_i){a_i^{(2)}}^T] + 2\lambda W^{(2)} \tag{7}$$

2.c

The partial derivative w.r.t the biases $b^{(2)}$ are:

$$\frac{\partial J}{\partial b^{(2)}} = \sum_{i=1}^{N} [\frac{\partial J}{\partial z_i^{(3)}} \frac{\partial z_i^{(3)}}{\partial b^{(2)}}] \tag{8}$$

that is similar to Eq.(5), a part for the evaluation of the term $\frac{\partial z_i^{(3)}}{\partial b^{(2)}} = 1$. So we have:

$$\frac{\partial \tilde{J}}{\partial b^{(2)}} = \frac{1}{N} \sum_{i=1}^{N} [(\psi(z_i^{(3)}) - \Delta_i)] \tag{9}$$

By repeatedly applying the chain rule, we can evaluate the derivative of the loss w.r.t. the first layer of weights and biases. We can write in a vector way the derivative w.r.t. the weights $W^{(1)}$ as follow:

$$\frac{\partial \tilde{J}}{\partial W^{(1)}} = \sum_{i=1}^{N} [\frac{\partial \tilde{J}}{\partial z_i^{(3)}} \frac{\partial z_i^{(3)}}{\partial a_i^{(2)}} \frac{\partial a_i^{(2)}}{\partial z_i^{(2)}} \frac{\partial z_i^{(2)}}{\partial W^{(1)}}] + 2\lambda W^{(1)} \tag{10}$$

where we do the following consideration for each term in the sum:

- we have the first term from Eq.(4) because $\frac{\partial \tilde{J}}{\partial z_i^{(3)}} = \frac{\partial J}{\partial z_i^{(3)}}$;
- for the second term we have that $\frac{\partial z_i^{(3)}}{\partial a_i^{(2)}} = {W^{(2)}}^T$;
- the third term is the derivative of the ReLu function w.r.t. its argument, so it is:

$$\frac{\partial a_i^{(2)}}{\partial z_i^{(2)}} = \frac{\partial \phi(z_i^{(2)})}{\partial z_i^{(2)}} = \begin{cases} 1 & z_i^{(2)} > 0 \\ 0 & z_i^{(2)} < 0 \end{cases} = \mathbb{1}_{(z_i^{(2)} > 0)}; \tag{11}$$

- the fourth term is $\frac{\partial z_i^{(2)}}{\partial W^{(1)}} = {a_i^{(1)}}^T$.

Substituting all the obtained expressions, we get the gradients for all the weights $W^{(1)}$:

$$\frac{\partial \tilde{J}}{\partial W^{(1)}} = \frac{1}{N} \sum_{i=1}^{N} [(\psi(z_i^{(3)}) - \Delta_i)] {a_i^{(1)}}^T W^{(2)} \mathbb{1}_{(z_i^{(2)} > 0)} + 2\lambda W^{(1)} \tag{12}$$

The last gradient we have to evaluate is w.r.t. the first set of biases $b^{(1)}$, that is similar to the gradient in Eq.(12) a part for the absence of the term $a^{(1)}$ and the term due to the regularization:

$$\frac{\partial \tilde{J}}{\partial b^{(1)}} = \frac{1}{N} \sum_{i=1}^{N} [(\psi(z_i^{(3)}) - \Delta_i)] W^{(2)} \mathbb{1}_{(z_i^{(2)} > 0)} \tag{13}$$

Question 3

3.b

In this exercise we used the newly constructed model which uses SGD to converge faster (because it uses small portions of the training set chosen randomly), to the detriment of noise. Therefore it is necessary to look at the predictions on the test/validation set in order to find the right parameters for the model.
Therefore, we have implemented a grid search with different hyper-parameters around the default values. The parameters that were most incisive in the increase in performance were $learning\_rate = 0.001$, $reg = 0.01$ and $hidden\_size = 200$.
Finally, to get our best performance it was necessary to adjust the number of iterations from 3000 to 5000.
After all these changes we were able to improve the model from 0.48 to 0.533 accuracy.

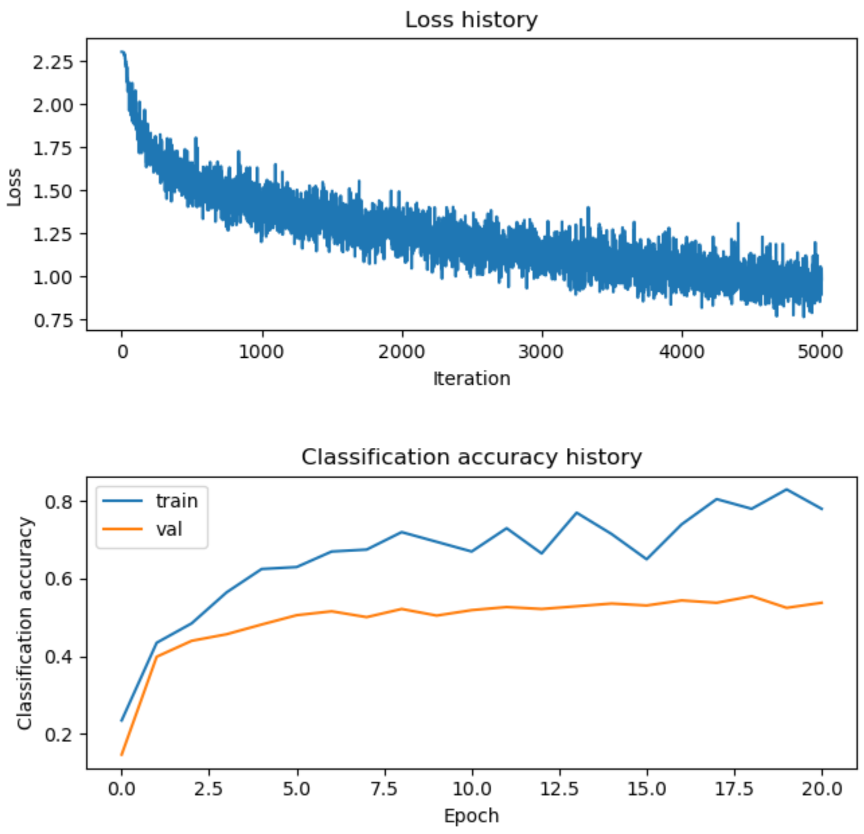

Figure 1: Best Model

Question 4

4.c

The model implemented on PyTorch gives us similar results to the model implemented from scratch with regard to 2-layers. By increasing the number of hidden layers we noticed that the accuracy did not improve so much. Our best performance is achieved with 3-layers, each with $hidden\_size = 50$.
The results obtained are expressed in the following table.

Layers	*Validation Accuracy*	*Test Accuracy*
2	*50.4%*	*51.1%*
3	*52.5%*	*52.8%*
4	*52.1%*	*51.2%*
5	*51.6%*	51.9%

Table 1: Accuracy of the model on validation and test set using configurations with different number of layers.

Once we achieved these results, we started trying to improve the 3-layers model by changing activation function, adding batch normalization, dropout and other methods.
In the end, we found our best model using a training set of 45,000 (90%) images and the test set of 5,000 (10%). The activation function taken into consideration is the LeakyRelu(0.1) and we have also added the Batch Normalization. The hidden_sizes are [400, 400, 400]. The Dropout didn't give us any improvements.
In conclusion, we have been able to obtain a validation accuracy equal to 56.1% and a test accuracy equal to 56.4%.

Note: With more than 2-layers, we noticed that the model did not update the weights, thus, we decided to use the default weights of PyTorch by removing the line of code n.167 (in our script) "model.apply(weights_init)".