Advanced Machine Learning
Exercise 2
Deep Neural Network and Backpropagation

Abbonato Diletta, Pezone Francesco, Testa Lucia

25 March 2020

Question 2

2.a

The input vector is a column vector $\mathbf{z_i}^{(3)} \in \mathbb{R}^{s\times 1}$.

$$\frac{\partial J}{\partial \mathbf{z_i}^{(3)}} = \begin{pmatrix} \frac{\partial}{\partial (z_i^{(3)})_1} \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial}{\partial (z_i^{(3)})_s} \end{pmatrix} \frac{1}{N}[-log(\psi(z_i^{(3)})_{y_i})] = \frac{1}{N} \begin{pmatrix} \psi(z_i^{(3)})_1 \\ \cdot \\ \cdot \\ \psi(z_i^{(3)})_{y_i} - 1 \\ \cdot \\ \cdot \\ \psi(z_i^{(3)})_s \end{pmatrix} = \frac{1}{N}[\boldsymbol{\psi}(\mathbf{z_i}^{(3)}) - \boldsymbol{\delta}_{i,y_i}] \tag{1}$$

From the formula (9) in the pdf we select only the i^{th} column vector, for this reason the sum disappears. Each element s^{th} of the vector is obtained as the product $\frac{\partial log(\psi(z_i^{(3)})_{y_i})}{\partial \psi(z_i^{(3)})_m} \frac{\partial \psi(z_i^{(3)})_m}{\partial (z_i^{(3)})_s}$ with (in the first formula we have implied the delta among m and y_i, we'll consider it in the second formula):

$$\frac{\partial}{\partial \psi(z_i^{(3)})_m} log(\psi(z_i^{(3)})_{y_i}) = \frac{1}{\psi(z_i^{(3)})_{y_i}}$$

$$\frac{\partial}{\partial (z_i^{(3)})_s} \psi(z_i^{(3)})_{y_i} = \begin{cases} \psi(z_i^{(3)})_{y_i}^2 - \psi(z_i^{(3)})_{y_i} & \text{if} \quad s = y_i \\ \psi(z_i^{(3)})_{y_i} \psi(z_i^{(3)})_s - \psi(z_i^{(3)})_{y_i} & \text{if} \quad s \neq y_i \end{cases}$$

2.b

We initially consider the single term of the summation, as in the previous point. The only thing we have to prove is $\frac{\partial \mathbf{z_i}^{(3)}}{\partial \mathbf{W}^{(2)}} = \frac{\partial}{\partial \mathbf{W}^{(2)}} \mathbf{W}^{(2)} \mathbf{a}_i^{(2)} = \mathbf{a}_i^{(2)^T}$.
To do this we generally write the formulas in the instructions file.

$$\mathbf{Y} = \boldsymbol{AB} \qquad \Longrightarrow \qquad \begin{cases} \frac{\partial L}{\partial \mathbf{A}} = \frac{\partial L}{\partial \mathbf{Y}} \mathbf{B}^T \\ \frac{\partial L}{\partial \mathbf{B}} = \mathbf{A}^T \frac{\partial L}{\partial \mathbf{Y}} \end{cases}$$

Observing the formulas, the demonstration is immediate.
The sum is due to the sum in the formula of J in the instructions file, since now we must consider all the column vectors.

The second thing to prove is $\frac{\partial}{\partial \mathbf{W}^{(2)}} \left\| \mathbf{W}^{(2)} \right\|_2^2 = 2\mathbf{W}^{(2)}$ since λ is a moltiplicative constant in $\mathbf{W}^{(2)}$. For a generic element of the matrix we have:

$$\frac{\partial}{\partial W_{m,n}^{(2)}} \left\| \mathbf{W}^{(2)} \right\|_2^2 = \frac{\partial}{\partial W_{m,n}^{(2)}} \sum_i \sum_j W_{i,j}^{(2)} = 2\delta_{m,i}\delta_{n,j} = 2W_{m,n}^{(2)}$$

So for each $W_{m,n}^{(2)}$ and multiplying by λ the proof is over.
At the end we have:

$$\frac{\partial J}{\partial \mathbf{W}^{(2)}} = \sum_{i=1}^{N} \frac{\partial J}{\partial \mathbf{z}_i^{(3)}} \mathbf{a}_i^{(2)^T} + 2\lambda \mathbf{W}^{(2)}$$

2.c

Let's start with $\frac{\partial J}{\partial \mathbf{b}^{(2)}}$, and as before we first consider the case with a single input vector; with the chain rule we have:

$$\frac{\partial J}{\partial \mathbf{z}_i^{(3)}}\frac{\partial \mathbf{z}_i^{(3)}}{\partial \mathbf{b}^{(2)}} = \begin{pmatrix} \frac{\partial}{\partial b_1^{(2)}} \\ . \\ . \\ . \\ \frac{\partial}{\partial b_s^{(2)}} \end{pmatrix} \begin{pmatrix} (z_i^{(3)})_1 & . & . & . & (z_i^{(3)})_s \end{pmatrix} \frac{\partial J}{\partial \mathbf{z}_i^{(3)}} = \mathbb{I}_{s\times s}\frac{\partial J}{\partial \mathbf{z}_i^{(3)}} = \frac{\partial J}{\partial \mathbf{z}_i^{(3)}}$$

As in point b, when we consider all the N vectors we have to add up all the columns, therefore:

$$\frac{\partial J}{\partial \mathbf{b}^{(2)}} = \sum_{i=1}^{N}\frac{\partial J}{\partial \mathbf{z}_i^{(3)}}\frac{\partial \mathbf{z}_i^{(3)}}{\partial \mathbf{b}^{(2)}} = \sum_{i=1}^{N}\frac{\partial J}{\partial \mathbf{z}_i^{(3)}}$$

In order to find the other two it is necessary to continue the backpropagation, for this we calculate the new term, we use the formula at the beginning of point b, note than we are considering the whole $\mathbf{z}^{(3)} \in \mathbb{R}^{s\times N}$ made of vector columns $\mathbf{z}_i^{(3)}$:

$$\frac{\partial J}{\partial \mathbf{a}^{(2)}} = \frac{\partial J}{\partial \mathbf{z}^{(3)}}\frac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{a}^{(2)}} = \mathbf{W}^{(2)^T}\frac{\partial J}{\partial \mathbf{z}^{(3)}}$$

Now we have to go back to the relu; since it is $max\{0, \mathbf{z}^{(2)}\}$ its derivative is the known Heaviside step function. The result should be a matrix, for each column of $\mathbf{z}^{(3)}$ but if we introduce the Hadamard product elementwise we can save a lot of memory space.

$$\frac{\partial J}{\partial \mathbf{z}^{(2)}} = \frac{\partial J}{\partial \mathbf{a}^{(2)}}\frac{\partial \mathbf{a}^{(2)}}{\partial \mathbf{z}^{(2)}} = \frac{\partial J}{\partial \mathbf{a}^{(2)}} \odot \mathbf{H}(\mathbf{z}^{(2)})$$

And now we can finally start with the last two elements (notice that we are going to use the same steps of the upper layers):

$$\frac{\partial J}{\partial \mathbf{b}^{(1)}} = \sum_{i=1}^{N}\frac{\partial J}{\partial \mathbf{z}_i^{(2)}}\frac{\partial \mathbf{z}_i^{(2)}}{\partial \mathbf{b}^{(1)}} = \sum_{i=1}^{N}\frac{\partial J}{\partial \mathbf{z}_i^{(2)}}$$

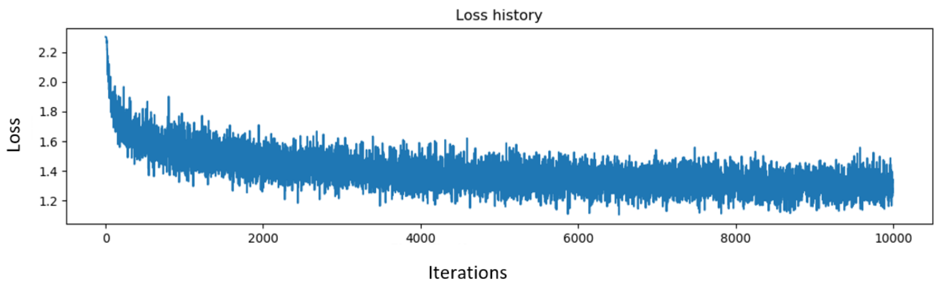
$$\frac{\partial J}{\partial \mathbf{W}^{(1)}} = \sum_{i=1}^{N}\frac{\partial J}{\partial \mathbf{z}_i^{(2)}}\frac{\partial \mathbf{z}_i^{(2)}}{\partial \mathbf{W}^{(1)}} + 2\lambda\mathbf{W}^{(1)} = \sum_{i=1}^{N}\frac{\partial J}{\partial \mathbf{z}_i^{(2)}}\mathbf{a}_i^{(1)^T} + 2\lambda\mathbf{W}^{(1)}$$

Question 3

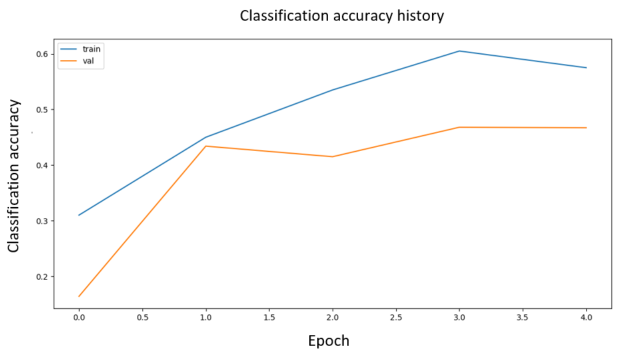
3.b

SGD works one example at a time as it updates the parameters using only a single training instance in each iteration that is randomly chosen and converges quickly with noisy estimates of the error gradient. The best way to set the parameters was to monitor the validation/test of the loss and tuning the hyperparameters for a few epochs. The first step in finding the best configuration was to cyclically increase the learning rate, leaving the other hyperparameters unchanged. Once we found an optimal value that would lead the analysis to exceed 0.48, we continued in changing of the other hyperparameters. Increasing and decreasing the batch size and the number of iterations. Initially increasing the number of iterations a noticeable increase was noticed, from 0.48 to 0.50, remaining with a batch size of 200. After that, a better performance was noticed by increasing the batch size from 200 to 300, also bringing the number of iterations to 5000.

Plots when accuracy = 0.48

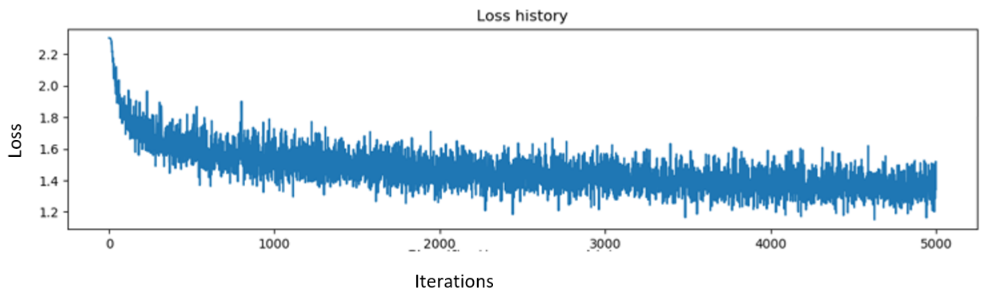


(a)

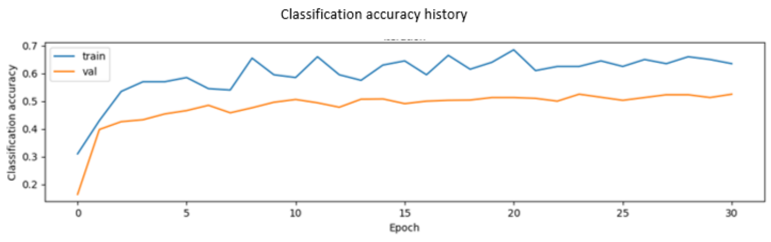


(b)

Plots with intermediate values, with accuracy= 0.522

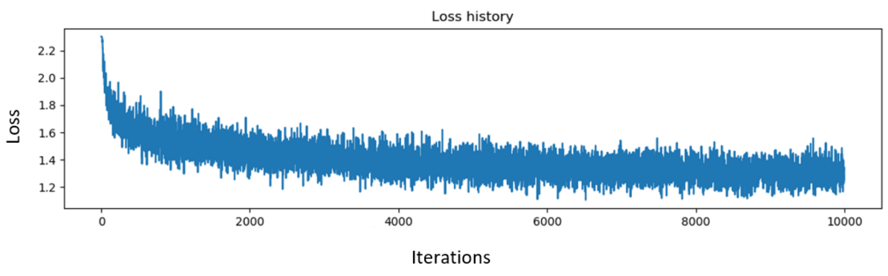


(c)

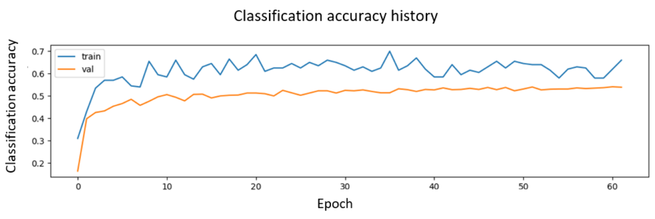


(d)

Plots with best values, with accuracy= 0.534



(e)



(f)

Question 4

4.c

Shown below are training and validation results with 1,3,4 and 5 layers. From the results obtained we can see that as the hidden size increases the accuracy improves.

Hidden layers	**Accuracy Train**	**Accuracy Test**
default	50.8	49.5
3	52.1	51.7
4	52.0	51.8
5	52.7	53.3

It must be said, however, that the default results were obtained with the initial weights while for the other values we used the default weights given by Pytorch.