Sapienza Università di Roma
Data Science

Assignment 2

Students name: Giuliano Martinelli 1915652, Gabriele Giannotta 1909375, Mario Dhimitri 1910181

Course: *Advanced Machine Learning* – Professor: *Fabio Galasso*
Due date: *November 19th, 2020*

Question 2 - Backpropagation

2. a

Verify that the loss function defined in Eq. (1) has gradient w.r.t. $z^{(3)}$ as Eq. (2):

$$J\left(\theta, \{x_i, y_i\}_{i=1}^N\right) = \frac{1}{N}\sum_{i=1}^N -\log\left[\frac{\exp\left(z_i^{(3)}\right)_{y_i}}{\sum_{j=1}^K \exp\left(z_i^{(3)}\right)_j}\right] \tag{1}$$

$$\frac{\partial J}{\partial z_i^{(3)}}\left(\theta, \{x_i, y_i\}_{i=1}^N\right) = \frac{1}{N}\left(\psi\left(z_i^{(3)}\right) - \delta_{iy_i}\right) \tag{2}$$

Where δ is the Kronecker delta:

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

It is possible to verify the initial assumption by calculating the gradient:

1. $f(x) = \frac{1}{N} log(x) \rightarrow \frac{\partial f(x)}{\partial x} = -\frac{1}{Nx}$

   $J\left(\theta, \{x_i, y_i\}_{i=1}^N\right) = f(x) = -\frac{1}{N} log(\psi(z_i^{(3)})_{y_i}), \qquad x = \psi(z_i^{(3)})_{y_i}$

   $\frac{\partial J}{\partial \psi(z_i^{(3)})_{y_i}} = -\frac{1}{N\psi(z_i^{(3)})_{y_i}} = \frac{\partial J}{\partial a_i^{(3)}}$

2. $f(x) = \psi(x) \rightarrow \frac{\partial f(x)}{\partial x} = \psi(x)(1 - \psi(x))$

   $a_i^{(3)} = f(x) = \psi(z_i^{(3)}), \qquad x = z_{y_i}^{(3)}$

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

3. $\frac{\partial J}{\partial z_i^{(3)}} = \frac{\partial J}{\partial a_i^{(3)}} \frac{\partial a_i^{(3)}}{\partial z_i^{(3)}} = -\frac{1}{N\psi(z_i^{(3)})_{y_i}}\ \psi(z_i^{(3)})_{y_i}(1 - \psi(z_i^{(3)})_{y_i}) = \frac{1}{N}(\psi(z_i^{(3)})_{y_i} - 1)$

   This because $\frac{\partial J}{\partial a_i^{(3)}}$ is the upstream gradient and $\frac{\partial a_i^{(3)}}{\partial z_i^{(3)}}$ is the local gradient.

2. b

To verify that the partial derivative of the loss w.r.t. $W^{(2)}$ is:

$$\begin{aligned}\frac{\partial J}{\partial W^{(2)}}\left(\theta, \{x_i, y_i\}_{i=1}^N\right) &= \sum_{i=1}^N \frac{\partial J}{\partial z_i^{(3)}} \cdot \frac{\partial z_i^{(3)}}{\partial W^{(2)}} \\ &= \sum_{i=1}^N \frac{1}{N}\left(\psi\left(z_i^{(3)}\right) - \delta_{iy_i}\right) a_i^{(2)^T}\end{aligned}$$

We can use the property as follows:

$f(x) = aW, \qquad \frac{\partial f(x)}{\partial a} = W, \qquad \frac{\partial f(x)}{\partial W} = a$

Using upstream and local gradient, we can apply the chain rule:

$\frac{\partial J}{\partial W_2} = \frac{\partial J}{\partial z_i^{(3)}} \frac{\partial z_i^{(3)}}{\partial W_2} \qquad \frac{\partial z_i^{(3)}}{\partial W_2} = a_i^{(2)} \qquad since \;\; z_i^{(3)} = W_2 a_i^{(2)} + b$

$\frac{\partial J}{\partial W_2} = \frac{1}{N}(\psi z_i^{(3)} - 1) a_i^{(2)}$

To verify that the regularized loss in Eq. (3) has the derivative as Eq. (4):

$$\tilde{J}\left(\theta, \{x_i, y_i\}_{i=1}^N\right) = \frac{1}{N}\sum_{i=1}^N -\log\left[\frac{\exp\left(z_i^{(3)}\right)_{y_i}}{\sum_{j=1}^K \exp\left(z_i^{(3)}\right)_j}\right] + \lambda\left(\left\|W^{(1)}\right\|_2^2 + \left\|W^{(2)}\right\|_2^2\right) \tag{3}$$

$$\frac{\partial \tilde{J}}{\partial W^{(2)}} = \sum_{i=1}^N \frac{1}{N}\left(\psi\left(z_i^{(3)}\right) - \delta_{iy_i}\right) a_i^{(2)^T} + 2\lambda W^{(2)} \tag{4}$$

We can do the following:

$f(x) = \lambda\left(\left\|W^{(1)}\right\|_2^2 + \left\|W^{(2)}\right\|_2^2\right) = \lambda\left(\left\|W^{(1)}\right\|_2^2\right) + \lambda\left(\left\|W^{(2)}\right\|_2^2\right) =$

$\frac{f}{W_2} = 0 + \lambda \frac{\partial(\sum\sum(W_2)^2)}{\partial W_2} = 2\lambda W_2$

$\frac{\partial J}{\partial W_2} = \frac{1}{N}(\psi(z_i^{(3)}) - 1) a_2^T + 2\lambda W_2$

2. c

We now derive the expressions for the derivatives of the regularized loss in Eq. (3) w.r.t. W(1), b(1), b(2), recalling that we used $\triangle_i$ to refer to the Kronecker delta.

- $\frac{\partial J}{\partial z_i^{(3)}} = -\frac{1}{N}(\psi(z_i^{(3)}) - \triangle_i$

- $z_i^{(3)} = a_i^{(3)} W^{(2)} + b^{(2)}$, so as we have seen in 2.b:

  $\frac{\partial J}{\partial W_2} = \frac{1}{N}(\psi(z_i^{(3)} - \triangle_i) a_{2i}$

  For the same reason:

- $\frac{\partial J}{\partial a_i^{(2)}} = \frac{1}{N}(\psi(z_i^{(3)} - \triangle_i) W_2$

- $\frac{\partial J}{\partial b_2} = \frac{\partial J}{\partial z_i^{(3)}} = \frac{1}{N}(\psi(z_i^{(3)}) - \triangle_i)$

  Using the chain rule:

- $\frac{\partial J}{\partial z_i^{(2)}} = \frac{\partial J}{\partial a_i^{(2)}} \frac{\partial a_i^{(2)}}{\partial z_i^{(2)}} \rightarrow a_i^{(2)} \begin{cases} 0, & \text{if } z_i^{(2)} < 0 \\ z_i^{(2)}, & \text{if } z_i^{(2)} \geq 0 \end{cases} \rightarrow \frac{\partial a_i^{(2)}}{\partial z_i^{(2)}} = \delta_i = \begin{cases} 0, & \text{if } z_i^{(2)} < 0 \\ 1, & \text{if } z_i^{(2)} \geq 0 \end{cases}$

  $\frac{\partial J}{\partial z_i^{(2)}} = \frac{1}{N}(\psi(z_i^{(2)}) - \triangle_i)\delta_i$

- $\frac{\partial J}{\partial b_1} = \frac{\partial J}{\partial z_i^{(2)}} = \frac{1}{N}(\psi(z_i^{(2)}) - \triangle_i)\delta_i$

- $\frac{\partial J}{\partial W_1} = \frac{\partial J}{\partial z_i^{(2)}} a_1^{(1)} = \frac{1}{N}(\psi(z_i^{(2)}) - \triangle_i)\delta_i \; a_1^{(1)}$

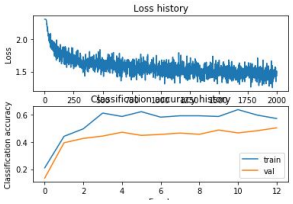
- $\frac{\partial J}{\partial a_i} = \frac{\partial J}{\partial z_i^{(2)}} W_1 = \frac{1}{N}(\psi(z_i^{(2)}) - \triangle_i)\delta_i \; W_1$

Question 3 - Stochastic gradient descent

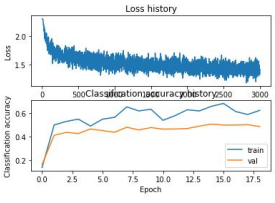
Stochastic Gradient Descent is a type of the gradient descent algorithm that calculates the error and updates the model for each record in the training dataset. Memory consumption will be low for this type of gradient descent. The way we approached this task, was through testing different batch sizes and number of iterations, in order to spot the best results for the validation accuracy, loss, and test accuracy. The statistics we got from this process were the following:

Iteration	Batch Size	Learning Rate	Validation Accuracy	Test Accuracy
2000	300	2.00E-03	0.502	0.483
3000	300	2.00E-03	0.486	0.496
3000	200	2.00E-03	0.502	0.489
4000	200	2.00E-03	0.518	0.505
4000	300	2.00E-03	0.5	0.506
5000	300	2.00E-03	0.516	0.529
5000	200	2.00E-03	0.503	0.519
5000	100	2.00E-03	0.511	0.506
6000	300	2.00E-03	0.52	0.524
6000	200	2.00E-03	0.527	0.511
6000	500	2.00E-03	0.525	0.512
6000	100	2.00E-03	0.501	0.503

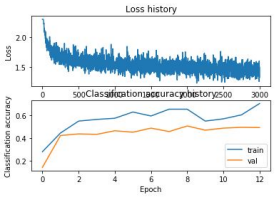
Iteration size we chose was in the range [2000-6000], batch size [100-300]. We got the following plots for the testing. We can see that we tend to have less significant changes after 5000 iterations in our accuracies.



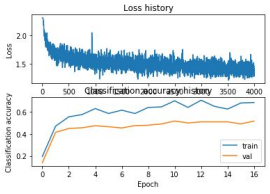
(a) It=2000, Batch Size=300



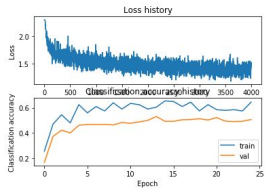
(b) It=3000, Batch Size=300



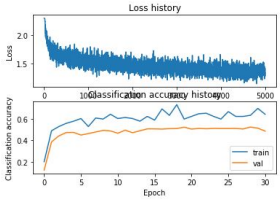
(c) It=3000, Batch Size=200



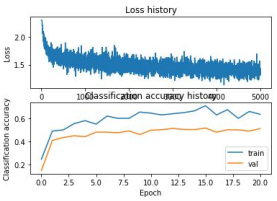
(d) It=4000, Batch Size=200



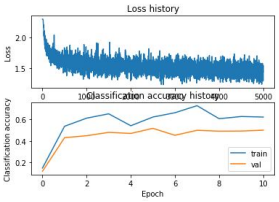
(e) It=4000, Batch Size=300



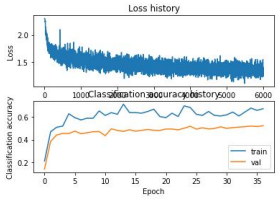
(f) It=5000, Batch Size=300



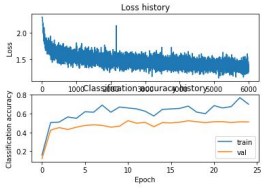
(g) It=5000, Batch Size=200



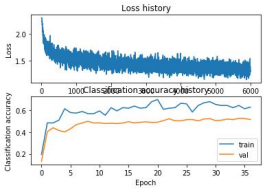
(h) It=5000, Batch Size=100



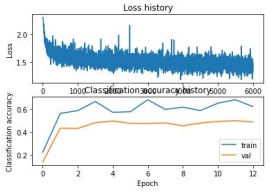
(i) It=6000, Batch Size=300



(j) It=6000, Batch Size=200



(k) It=6000, Batch Size=500



(l) It=6000, Batch Size=100

Figure 1: Loss and Train/ Validation accuracies

Question 4 - Multi-layer perceptron using PyTorch

4. c

Table 1: Train and Test Accuracy for multiple layers.

Total Layers	Epochs	Hidden Layer Size	Train Accuracy	Test Accuracy
3	20	250, 50	55.3 %	56.8 %
2	20	100	55.8 %	54.0 %
3	10	250, 50	53.4 %	53.1 %
2	10	100	52.2 %	51.5 %
5	20	200, 200, 250, 150	9.8 %	10.0 %
4	10	150, 100, 200	9.8 %	9.0 %
4	20	150, 100, 200	9.8 %	9.0 %
5	10	200, 200, 250,150	9.8 %	9.0 %

In table 1 we can see that for all tests with a number of layers greater than 3, the accuracy obtained is less than 10%. The best accuracies were obtained in networks with 2 and 3 layers where the number of epochs was equal to 20. In particular, the best result among those tested was obtained by setting 2 hidden layers with number of neurons of 250 and 50 respectively, and a number of epochs equal to 20.