Students:
Cruoglio Antonella, Iovino Giuliana,
Mascolo Davide, Napoli Mario

Advanced Machine Learning
Winter Semester 2022

Exercise 1: Deep Neural Network and Backpropagation

---

1. **Question 1: Implementing the feedforward model**
   See code for solution. (two_layernet.py, ex2_FCnet.py)

2. **Question 2: Backpropagation**

   Point a

   So, in conclusion:

   Verify that $\tilde{J}\left(\theta, \{x_i, y_i\}_{i=1}^N\right) = \frac{1}{N}\sum_{i=1}^N - \log\left[\frac{\exp\left(z_i^{(3)}\right)_{y_i}}{\sum_{j=1}^K \exp\left(z_i^{(3)}\right)_j}\right]$ has gradient w.r.t. z_i^3 as below:

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

   We can demonstrate that:

   - if $i = y_i$, $\quad \frac{\partial \psi(z_i^3)}{\partial z_i 3} = p_{y_i} \times (1 - p_i) = \frac{\exp\left(z_i^{(3)}\right)_{y_i}}{\sum_{j=1}^K \exp\left(z_i^{(3)}\right)_j} \times \frac{\left(\sum_{j=1}^K \exp\left(z_i^{(3)}\right)_i - \exp(z_i^{(3)})_i\right)}{\sum_{j=1}^K \exp\left(z_i^{(3)}\right)_j}$

   - if $i \neq y_i$, $\quad \frac{\partial \psi(z_i^3)}{\partial z_i 3} = -p_i \times p_{y_i} = \frac{\exp\left(z_i^{(3)}\right)_i}{\sum_{j=1}^K \exp\left(z_i^{(3)}\right)_j} \times \frac{\exp\left(z_i^{(3)}\right)_{y_i}}{\sum_{j=1}^K \exp\left(z_i^{(3)}\right)_j}$

   So, in conclusion:

$$\frac{\partial p_i}{\partial z_i 3} = \begin{cases} p_{y_i} \times (1 - p_i), & \text{if } i = y_i \\ -p_{y_i} \times p_i, & \text{if } i \neq y_i \end{cases}$$

   So, the derivative of the loss is:

$$- \frac{\partial J}{\partial z_i^{(3)}} = \frac{1}{N} \text{-} \left(\frac{\sum_{j=1}^K \exp\left(z_i^{(3)}\right)_j}{\exp\left(z_i^{(3)}\right)_{y_i}}\right) \times \left(\frac{\exp\left(z_i^{(3)}\right)_{y_i}}{\sum_{j=1}^K \exp\left(z_i^{(3)}\right)_j} \times \frac{\left(\sum_{j=1}^K \exp\left(z_i^{(3)}\right)_j - \exp\left(z_i^{(3)}\right)_i\right)}{\sum_{j=1}^K \exp\left(z_i^{(3)}\right)_j}\right) =$$

$$= \frac{1}{N} \times \left(-\frac{\left(\sum_{j=1}^K \exp\left(z_i^{(3)}\right)_j - \exp\left(z_i^{(3)}\right)_i\right)}{\sum_{j=1}^K \exp\left(z_i^{(3)}\right)_j}\right) = \frac{1}{N} \times (p_{y_i} - 1), \quad \text{if } y_i \neq i$$

- $\dfrac{\partial J}{\partial z_i^{(3)}} = \frac{1}{N}$- $\left(\dfrac{\sum_{j=1}^{K} \exp\left(z_i^{(3)}\right)_j}{\exp\left(z_i^{(3)}\right)_{y_i}}\right) \times \left(\dfrac{-\exp\left(z_i^{(3)}\right)_i}{\sum_{j=1}^{K} \exp\left(z_i^{(3)}\right)_j} \times \dfrac{\exp\left(z_i^{(3)}\right)_{y_i}}{\sum_{j=1}^{K} \exp\left(z_i^{(3)}\right)_j}\right) =$

$$= \frac{1}{N} \times \left(\frac{\exp\left(z_i^{(3)}\right)_i}{\sum_{j=1}^{K} \exp\left(z_i^{(3)}\right)_j}\right) = \frac{1}{N} \times p_i, \quad \text{if } y_i = i$$

So, in conclusion:

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

where:
- $\psi\left(z_i^{(3)}\right) = p_i$
- $\Delta_i(j) = \begin{cases} 1, & \text{if } i = y_i \\ 0, & \text{otherwise} \end{cases}$

Point c

Now, we derive the expression of the derivatives of the regularized loss w.r.t. $\Theta = \left(W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}\right)$.

- for $W^{(2)}$:

$$\frac{\partial \tilde{J}}{\partial W^{(2)}} = \frac{\partial \tilde{J}}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial W^{(2)}} = \sum_{j=1}^{K} \frac{1}{N} \left(\psi\left(z_i^{(3)}\right) - \Delta_i\right) \cdot a_i^{(2)^T} + 2\lambda W^{(2)} \tag{1}$$

- for $b^{(2)}$:

$$\frac{\partial \tilde{J}}{\partial b^{(2)}} = \frac{\partial \tilde{J}}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial b^{(2)}} = \sum_{j=1}^{K} \frac{1}{N} \left(\psi\left(z^{(3)}\right) - \Delta_i\right) \tag{2}$$

- for $b^{(1)}$:

$$\frac{\partial \tilde{J}}{\partial b^{(1)}} = \frac{\partial \tilde{J}}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial b^{(1)}} = \sum_{j=1}^{K} \frac{1}{N} \left(\psi\left(z_i^{(3)}\right) - \Delta_i\right) \cdot W^{(2)} \cdot 1[z^{(2)} > 0] \tag{3}$$

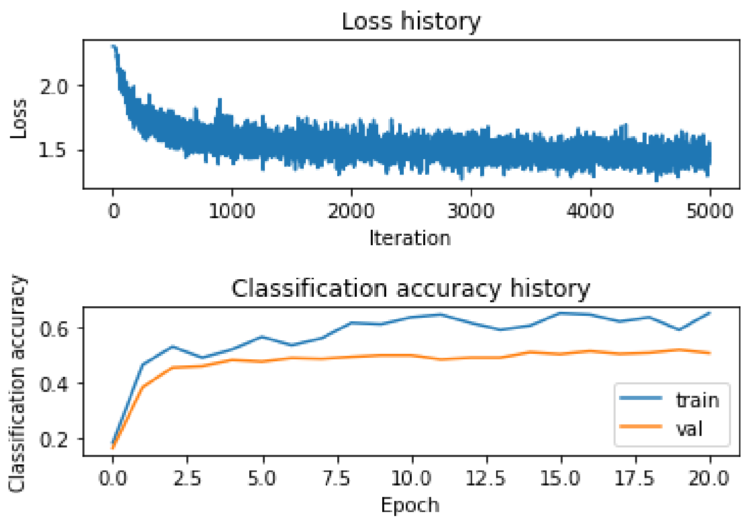
- for $W^{(1)}$:

$$\frac{\partial \tilde{J}}{\partial W^{(1)}} = \frac{\partial \tilde{J}}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)} \cdot \frac{\partial z^{(2)}}{\partial W^{(1)}}} = \sum_{j=1}^{K} \frac{1}{N} \left(\psi\left(z_i^{(3)}\right) - \Delta_i\right) \cdot W^{(2)} \cdot 1[z^{(2)} > 0] + 2\lambda W^{(2)} \tag{4}$$

3. **Question 3: Stochastic gradient descent training**

   See code for solution. (two_layernet.py, ex2_FCnet.py)

   We chose different hyper-parameters to improve the performances of the model. We set the hidden size equal to 80, the number of iterations equal to 50, the learning rate to 1e-3 and we increased the regularization term to 0.5. This set up led to a validation accuracy of **52.4%** and a test accuracy of **53.6%**.



4. **Question 4: Implement multi-layer perceptron using PyTorch library**

   Using a two-layer network with a batch size equal to 100, we achieved an accuracy of **53.8%** on the validation set and **53.0%** on the test set.

   Then we tried increasing the network depth to see if you can improve the performance. We noticed that increasing the depth up to 3 layers we got also an slight improvement of the accuracy on the tezt set.

   So at the end we set 3 layers and got an accuracy of **53.4%** on the validation set and **53.8%** on the test set.