AML Report - HW1

Sapienza, Università di Roma, AY 20/21

October 27, 2021

Mattia Capparella
capparella.1746513@studenti.uniroma1.it

Flavia Penta de Peppo
pentadepeppo.1758942@studenti.uniroma1.it

Antonio Zappia
zappia.1747573@studenti.uniroma1.it

Q2 Backpropagation

let

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

a) Verify that the loss function defined in Eq. 1 has gradient w.r.t. z_i as below:

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

were Δ_i is a vector of K dimensions with:

$$(\Delta_i)_j = \begin{cases} 1, & \text{if } j = y_i \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$\begin{aligned}
\frac{\partial J}{\partial z_i} &= \frac{\partial \frac{1}{N}\sum_{i=1}^{N} -log\left[\frac{exp(z_i)_{y_i}}{\sum_{j=1}^{K} exp(z_i)_j}\right]}{\partial z_i} \\
&= \frac{\partial \frac{1}{N}\sum_{i=1}^{N} -log\left[exp(z_i)_{y_i}\right] + log\left[\sum_{j=1}^{K} exp(z_i)_j\right]}{\partial z_i} \\
&= \frac{1}{N}\frac{\partial \sum_{i=1}^{N} -log\left[e^{z_{iy_i}}\right] + log\left[\sum_{j=1}^{K} exp(z_i)_j\right]}{\partial z_i} \\
&= \frac{1}{N}\left(\frac{\partial log\left[\sum_{j=1}^{K} e^{z_{ij}}\right]}{\partial z_i} - \frac{\partial log\left[e^{z_{iy_i}}\right]}{\partial z_i}\right) \\
&= \frac{1}{N}\left(\frac{1}{\sum_{j=1}^{K} e^{z_{ij}}}\frac{\partial \sum_{j=1}^{K} e^{z_{ij}}}{\partial z_i} - \frac{\partial z_{iy_i} log(e)}{\partial z_i}\right) \\
&= \frac{1}{N}\left(\frac{1}{\sum_{j=1}^{K} e^{z_{ij}}}\frac{\partial \left[e^{z_{i1}} + \cdots + e^{z_{iK}}\right]}{\partial z_i} - \frac{z_{iy_i}}{\partial z_i}\right) \\
&= \frac{1}{N}\left(\frac{1}{\sum_{j=1}^{K} e^{z_{ij}}}\left[\frac{\partial e^{z_{i1}}}{\partial z_i} + \cdots + \frac{\partial e^{z_{iK}}}{\partial z_i}\right] - \Delta_i\right) \\
&= \frac{1}{N}\left(\frac{1}{\sum_{j=1}^{K} e^{z_{ij}}} e^{z_i} - \Delta_i\right) \\
&= \frac{1}{N}\left(\psi(z_i) - \Delta_i\right)
\end{aligned} \tag{4}$$

b) Compute the effect of the weight matrix W^2 on the loss in Eq. 1 incurred by the network. This is done applying the chain rule. Verify that the partial derivative of the loss w.r.t. $W^{(2)}$ is:

$$\begin{aligned}\frac{\partial J}{\partial W^{(2)}}(\theta, \{x_i, y_i\}_{i=1}^N) &= \sum_{i=1}^{N} \frac{\partial J}{\partial z_i^{(3)}} \frac{\partial z_i^3}{\partial W^{(2)}} \\ &= \sum_{i=1}^{N} \frac{1}{N} \left(\psi(z_i^3) - \Delta_i\right) a_i^{(2)^T}\end{aligned} \tag{5}$$

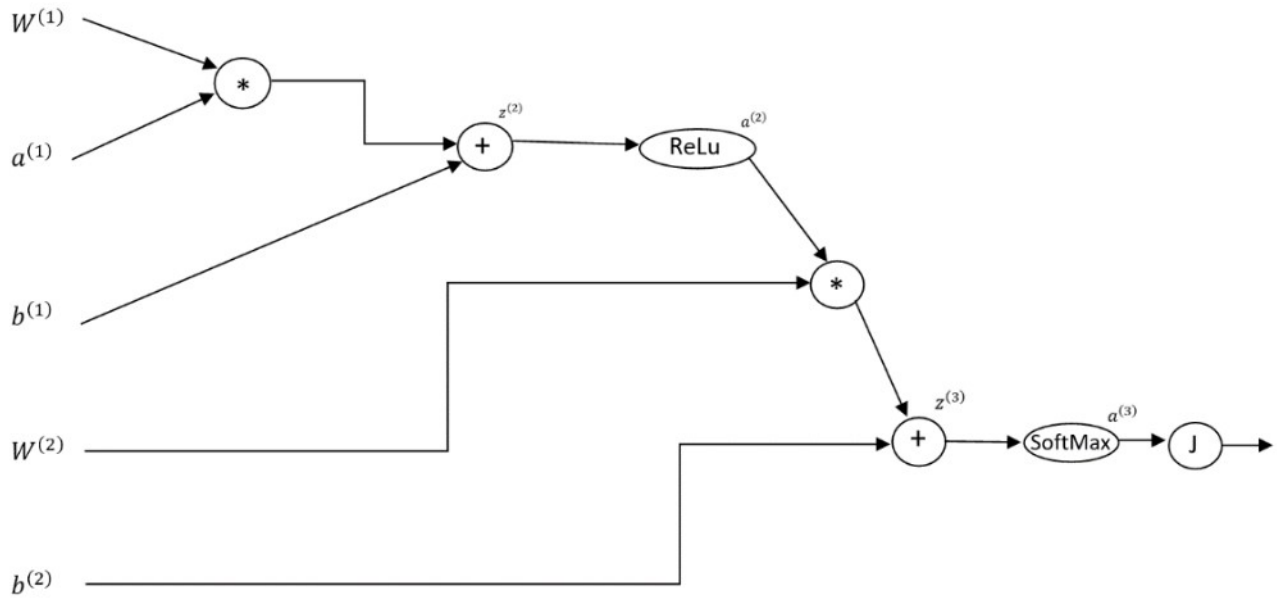

Figure 1: Computational Graph of J

The first term in the summation in 5 has just been computed, and from the 1 we can see that the second term can be computed by applying the rule relative to the "MulGate" seen in class. A final step must be done in order to have the terms (and their dimensions) in the right order: as seen in the "Backprop with Matrices" section, since $\frac{\partial L}{\partial w} = x^T \left(\frac{\partial L}{\partial y}\right)$, we must reformulate 5 as:

$$\sum_{i=1}^{N} \frac{1}{N} a_i^{(2)^T} \left(\psi(z_i^3) - \Delta_i\right)$$

The left term of 5 has dimensions $[H, C]$, the right term has dimensions: $[H, 1]\,[1, C] \rightarrow [H, C]$: the dimensions are consistent.

Let the regularized loss be defined by:

$$\tilde{J}(\theta, \{x_i, y_i\}_{i=1}^N) = \frac{1}{N}\sum_{i=1}^{N} -log\left[\frac{exp(z_i)_{y_i}}{\sum_{j=1}^{K} exp(z_i)_j}\right] + \lambda\left(\|W^{(1)}\|_2^2 + \|W^{(2)}\|_2^2\right) \tag{6}$$

where $\|.\|_2^2$ is the squared L_2 norm. For example:

$$\|W\|_2^2 = \sum_{p=1}^{P}\sum_{q=1}^{Q}(W_{pq})^2$$

Similarly, verify that the regularized loss has the derivative w.r.t. $W^{(2)}$ as:

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

The first part of the right hand term of is given for free by the computation we already did, while the second part is given by:

$$\begin{aligned}
\frac{\partial\lambda\left(\|W^{(1)}\|_2^2 + \|W^{(2)}\|_2^2\right)}{\partial W^{(2)}} &= \frac{\partial\lambda\left(\|W^{(1)}\|_2^2\right)}{\partial W^{(2)}} + \frac{\partial\lambda\left(\|W^{(2)}\|_2^2\right)}{\partial W^{(2)}} \\
&= 0 + \frac{\partial\lambda\left(\|W^{(2)}\|_2^2\right)}{\partial W^{(2)}} \\
&= \lambda\frac{\partial\sum_{p=1}^{P}\sum_{q=1}^{Q}(W_{pq})^2}{\partial\left(W^{(2)}\right)} \\
&= \lambda\frac{\sum_{p=1}^{P}\sum_{q=1}^{Q}\partial(W_{pq})^2}{\partial\left(W^{(2)}\right)} \\
&= 2\lambda W^{(2)}
\end{aligned}$$

A derivative of a scalar w.r.t. a matrix has indeed the dimension of the matrix, and here the components of the matrix are $\frac{\partial\left[w_{11}^2+\cdots+w_{PQ}^2\right]}{\partial W_{ij}^{(2)}} = 2w_{ij}\,\forall i,j$

c) Derive the expressions for the derivatives of Eq. 6 w.r.t. $W^{(1)}, b^{(1)}, b^{(2)}$**.**

For the derivative w.r.t. $W^{(1)}$ we have:

$$\begin{aligned}\frac{\partial \tilde{J}}{\partial W^{(1)}} &= \frac{\partial z^{(2)}}{\partial W^{(1)}}\frac{\partial a^{(2)}}{\partial z^{(2)}}\frac{\partial z^{(3)}}{\partial a^{(2)}}\frac{\partial J}{\partial z^{(3)}} \\ &= X^T\left(\partial Relu \times \left(W^{(2)^T}\frac{1}{N}\left(\psi(z)-\Delta\right)\right)\right) \\ &= X^T\left(\partial Relu \times \left(\frac{1}{N}\left(\psi(z)-\Delta\right)W^{(2)^T}\right)\right)\end{aligned} \tag{7}$$

where in the code, $\partial Relu \times ...$ is transformed into an implicit multiplication, as seen in slide 107 ("Backprop with Vectors").

For the derivative w.r.t. $b^{(2)}$ we have:

$$\begin{aligned}\frac{\partial \tilde{J}}{\partial b^{(2)}} &= \frac{\partial z^{(3)}}{\partial b^{(2)}}\frac{\partial \tilde{J}}{\partial z^{(3)}} \\ &= \frac{\partial W^{(2)}a^{(2)}+b^{(2)}}{\partial b^{(2)}}\frac{\partial \tilde{J}}{\partial z^{(3)}} \\ &= (\mathbf{0}+\mathbf{1})\frac{\partial \tilde{J}}{\partial z^{(3)}} \\ &= (\mathbf{0}+\mathbf{1})\frac{1}{N}\left(\psi(z)-\Delta\right)\end{aligned} \tag{8}$$

where $\mathbf{0}$ is a tensor with dimensions $[(1\times C)(N\times C)]$,
and $\mathbf{1}$ is a tensor with dimensions $[(1\times C)(1\times C)]$, hence the dimension of $\frac{\partial \tilde{J}}{\partial b^{(2)}}$ is: $[1\times C]$.

For the derivative w.r.t. $b^{(1)}$ we have:

$$\begin{aligned}\frac{\partial \tilde{J}}{\partial b^{(1)}} &= \frac{\partial z^{(2)}}{\partial b^{(1)}}\frac{\partial a^{(2)}}{\partial z^{(2)}}\frac{\partial z^{(3)}}{\partial a^{(2)}}\frac{\partial \tilde{J}}{\partial z^{(3)}} \\ &= \frac{\partial W^{(1)}a^{(1)}+b^{(1)}}{\partial b^{(1)}}\,\partial ReLU\,\frac{\partial W^{(2)}a^{(2)}+b^{(2)}}{\partial a^{(2)}}\frac{1}{N}\left(\psi(z)-\Delta\right) \\ &= (\mathbf{0}+\mathbf{1})\;\partial ReLU \times \left(W^{2^T}+\mathbf{0}\right)\frac{1}{N}\left(\psi(z)-\Delta\right)\end{aligned} \tag{9}$$

Q3 SGD - Stochastic Gradient Descent

b) Once you have tuned your hyper-parameters, and get validation accuracy greater than 48% run your best model on the test set once and report the performance.

In order to achieve higher Validation Accuracy we implemented a Grid of parameters with a discrete number of values for each (in particular: *learning rates* and *regularization strength*) to be explored iteratively. At the end of each training phase we plot both the Loss and Accuracies Histories to get better insights on how the current configuration performed during the training. It can be seen from the Accuracy plot that altough the goal is reached, the model is overfitting: the next step would be hence tweaking the *hidden size* of the model or introducing some strategies as *Dropout*.

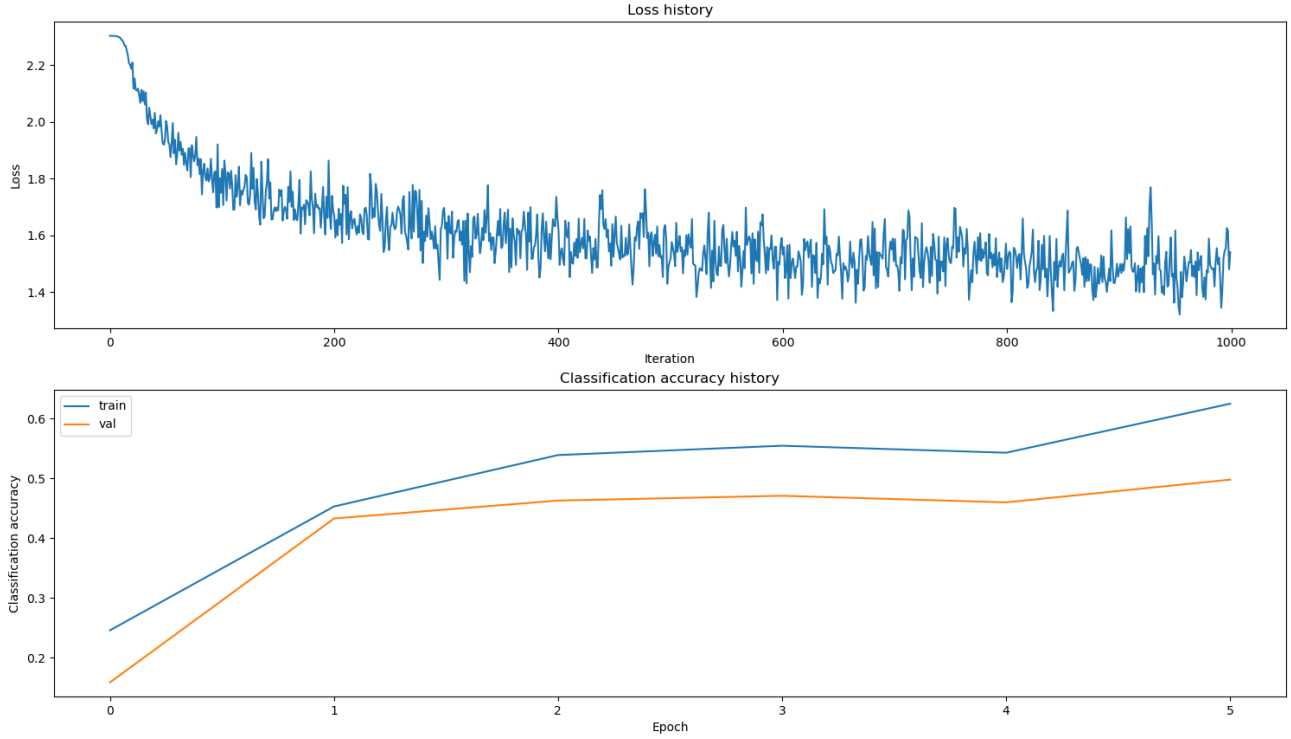

Figure 2: Loss and Accuracy Histories

Once we trained a model that reached the threshold of $ValidationAccuracy > 48\%$ we tested it on the Test Set and got the result:

```
iteration 800 / 1000: loss 1.437744
iteration 900 / 1000: loss 1.550037
Validation accuracy:  0.507
Test accuracy:  0.494
```

Figure 3: Test Accuracy with best network

In the Fig. 4 we have a visualization of the final parameters learned by the best network.

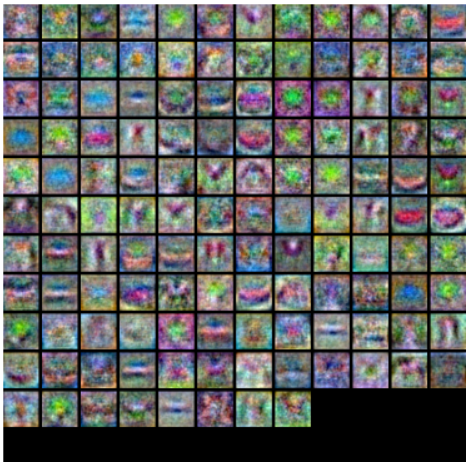

Figure 4: Weights of the best network

As expected, using the *dropout* strategy we have been able to reduce the overfitting, while keeping the accuracies reasonably high:

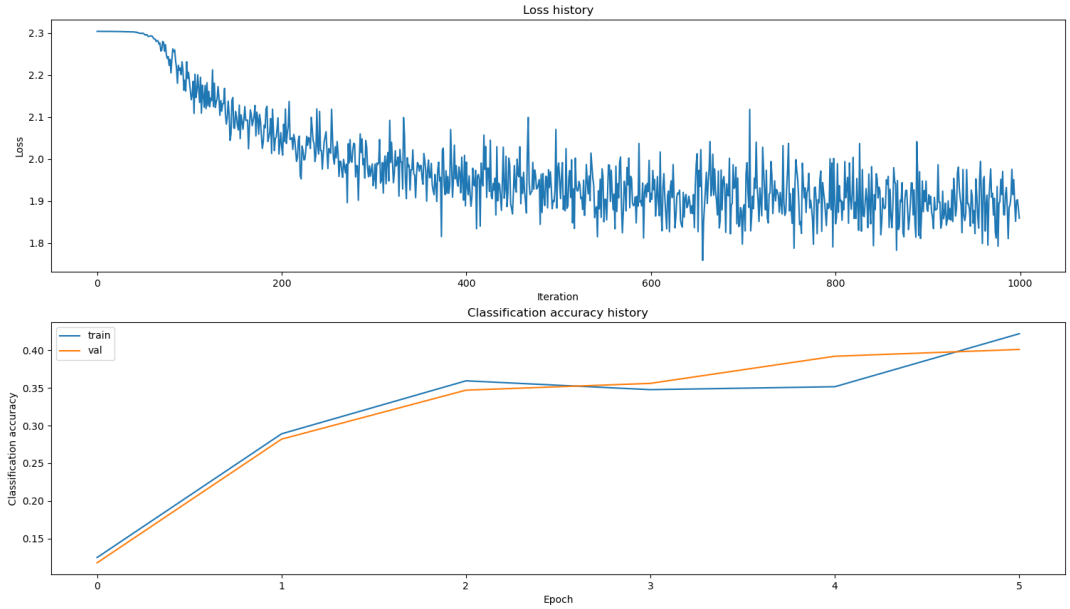

Figure 5: Loss and Accuracy Histories with Dropout strategy

We tested also this model on the Test Set achieving:

```
Test accuracy:  0.357
```

Figure 6: Test Accuracy with Dropout strategy

Q4 MLP using PyTorch

b) Report the final validation accuracy you achieve with a two-layer network.

The validation accuracy we achieve is: 50.4%

c) Try increasing the network depth to see if you can improve the performance. Experiment with networks of at least 2, 3, 4, and 5 layers, of your chosen configuration. Report the training (TA) and validation (VA) accuracies and discuss your observations. Run the evaluation on the test set with the best model and report the test accuracy (TEA)

We tried out 7 different configurations, as reported here, achieving the following results:

1. Linear $\rightarrow$ ReLU $\rightarrow$ Linear $\rightarrow$ ReLU $\rightarrow$ Linear $\rightarrow$ ReLU $\rightarrow$ Linear
   - TA: 9.94%
   - VA: 7.9%
2. Linear $\rightarrow$ ReLU $\rightarrow$ Linear $\rightarrow$ Linear $\rightarrow$ Linear
   - TA: 38.08%
   - VA: 39.2%
3. Linear $\rightarrow$ Linear $\rightarrow$ Linear $\rightarrow$ Linear $\rightarrow$ Linear $\rightarrow$ Linear
   - TA: 9.69%
   - VA: 7.80%
4. Linear $\rightarrow$ Sigmoid $\rightarrow$ Linear $\rightarrow$ ReLU $\rightarrow$ Linear $\rightarrow$ ReLU $\rightarrow$ Linear
   - TA: 36.80%
   - VA: 36.80%
5. Linear $\rightarrow$ Sigmoid $\rightarrow$ Linear $\rightarrow$ ReLU $\rightarrow$ Linear
   - TA: 43.88%
   - VA: 42.50%
6. Linear $\rightarrow$ ReLU $\rightarrow$ Linear $\rightarrow$ Dropout $\rightarrow$ Linear
   - TA: 51.90%
   - VA: 50.90%
7. Linear $\rightarrow$ ReLU $\rightarrow$ Linear $\rightarrow$ ReLU $\rightarrow$ Linear
   - TA: 53.02%
   - VA: 51.50%
   - **TEA**: 42.40%

While testing different architectures, we noted that adding too many FC layers in the network, either sequentially or sparse but without Dropout or MaxPooling, the VA decreases w.r.t. the TA: this is a clear index of Overfitting. The reason behind this behaviour is that with so many parameters the model is "memorizing" rather than "learning", and while it can achieve good results on the Train Set, it performs badly on new data.