Instituto Superior Técnico

**Aprendizagem 2021/22**

Second Exam, 25 February 2022

## 1. [5 pts] **Neural Networks**

Consider a MLP classifier characterized by the following weights

$$\mathrm{W}^{[1]} = \begin{pmatrix} 2 & 0 & -3 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & 2 & -2 \end{pmatrix}, \mathrm{b}^{[1]} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \mathrm{W}^{[2]} = \begin{pmatrix} -0.1 & 1 & 0 \\ -2 & 0 & -1 \end{pmatrix}, \mathrm{b}^{[2]} = \begin{pmatrix} 0.65 \\ 0.40 \end{pmatrix}$$

and activation function $\mathrm{f(x)} = \frac{1}{1+\mathrm{e}^{-2\mathrm{x}}} = \mathrm{sigmoid}(2\mathrm{x}) = \sigma(2\mathrm{x})$ for every unit of the hidden layer.

Consider the example $\mathrm{x} = (1,1,1,1)^{\mathrm{T}}$, $\mathrm{z} = (1,0)^{\mathrm{T}}$ and the cross-entropy error.

To make the math easier, consider $\sigma(2.2) = 0.9$ and $\sigma(-\mathrm{x}) = 1 - \sigma(\mathrm{x})$.

**a)** [1v] Do forward propagation

$$NET^{[1]} = \begin{pmatrix} 2 & 0 & -3 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & 2 & -2 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, a^{[1]} = \begin{pmatrix} \sigma(2*0) \\ \sigma(2*0) \\ \sigma(2*0) \end{pmatrix} = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$$

$$NET^{[2]} = \begin{pmatrix} -0.1 & 1 & 0 \\ -2 & 0 & -1 \end{pmatrix}\begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix} + \begin{pmatrix} 0.65 \\ 0.40 \end{pmatrix} = \begin{pmatrix} 0.45 + 0.65 \\ -1.5 + 0.40 \end{pmatrix} = \begin{pmatrix} 1.1 \\ -1.1 \end{pmatrix}, a^{[2]} = \begin{pmatrix} \sigma(2.2) \\ \sigma(-2.2) \end{pmatrix} = \begin{pmatrix} \sigma(2.2) \\ 1 - \sigma(2.2) \end{pmatrix} = \begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix}$$

**b)** [1v] Compute $\delta^{[2]} = \frac{\partial \mathrm{Error}}{\partial \mathrm{NET}^{[2]}}$

$$\delta^{[2]} = \begin{pmatrix} 2a_1^{[2]}\left(1 - a_1^{[2]}\right) & 0 \\ 0 & 2a_2^{[2]}\left(1 - a_2^{[2]}\right) \end{pmatrix}\begin{pmatrix} -\frac{z_1}{a_1^{[2]}} \\ -\frac{z_2}{a_2^{[2]}} \end{pmatrix} = \begin{pmatrix} -2a_1^{[2]}\left(1 - a_1^{[2]}\right)\frac{z_1}{a_1^{[2]}} \\ -2a_2^{[2]}\left(1 - a_2^{[2]}\right)\frac{z_2}{a_2^{[2]}} \end{pmatrix}$$

$$= \begin{pmatrix} -2\left(1 - a_1^{[2]}\right)z_1 \\ -2\left(1 - a_2^{[2]}\right)z_2 \end{pmatrix} = \begin{pmatrix} -2(1 - 0.9)1 \\ -2(1 - 0.1)0 \end{pmatrix} = \begin{pmatrix} -0.2 \\ 0 \end{pmatrix}$$

**c)** [0.5v] Update $\mathrm{W}^{[2]}$ using a learning rate of 1

$$W^{[2]} = W^{[2]} - 0.1\delta^{[2]}a^{[1]^T} = \begin{pmatrix} -0.1 & 1 & 0 \\ -2 & 0 & -1 \end{pmatrix} - 0.1\begin{pmatrix} -0.2 \\ 0 \end{pmatrix}(0.5 \quad 0.5 \quad 0.5)$$

$$= \begin{pmatrix} -0.1 & 1 & 0 \\ -2 & 0 & -1 \end{pmatrix} - 0.1\begin{pmatrix} -0.1 & -0.1 & -0.1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} -0.09 & 1.01 & 0.01 \\ -2 & 0 & -1 \end{pmatrix}$$

**d)** [1v] Compute $\delta^{[1]} = \frac{\partial \mathrm{Error}}{\partial \mathrm{NET}^{[1]}}$

$$\delta^{[1]} = \begin{pmatrix} 2a_1^{[1]}(1 - a_1^{[1]}) & 0 & 0 \\ 0 & 2a_2^{[1]}(1 - a_2^{[1]}) & 0 \\ 0 & 0 & 2a_3^{[1]}(1 - a_3^{[1]}) \end{pmatrix} W^{[2]^T}\delta^{[2]}$$

$$= \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{pmatrix}\begin{pmatrix} -0.1 & -2 \\ 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} -0.2 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.01 \\ -0.1 \\ 0 \end{pmatrix}$$

**e)** [0.5v] Update $b^{[1]}$ using a learning rate of 0.1

$$b^{[1]} = b^{[1]} - 0.1\delta^{[1]} = \begin{pmatrix}0\\0\\0\end{pmatrix} - 0.1\begin{pmatrix}0.01\\-0.1\\0\end{pmatrix} = \begin{pmatrix}0.001\\-0.01\\0\end{pmatrix}$$

**f)** [1v] Under certain conditions, the phenomenon of "*vanishing gradients*" appears when learning starts progressing very slowly. Looking at $\delta^{[2]}$, what are these conditions? What can we change in the architecture to avoid this? Answer each question with math and a short sentence.
*Hint*: look into the limits of $\delta^{[2]}$ for an increasing $NET^{[2]}$

From b) we know that $\delta^{[2]} = \begin{pmatrix}-2(1-a_1^{[2]})z_1\\-2(1-a_2^{[2]})z_2\end{pmatrix}$

Following the hint, we can write it in terms of $\mathrm{NET}^{[2]}$ as $\delta^{[2]} = \begin{pmatrix}-2(1-\sigma(\mathrm{NET}^{[2]}{}_1))z_1\\-2(1-\sigma(\mathrm{NET}^{[2]}{}_2))z_2\end{pmatrix}$,

or index-wise $\delta^{[2]}{}_i = -2(1-\sigma(\mathrm{NET}^{[2]}{}_i))z_i$

If we analyze what happens when NET values get very large:

$$\lim_{\mathrm{NET}^{[2]}{}_i\to\inf}\delta^{[2]}{}_i = \lim_{\mathrm{NET}^{[2]}{}_i\to\inf} -2(1-\sigma(\mathrm{NET}^{[2]}{}_i))z_i = -2\left(1-\lim_{\mathrm{NET}^{[2]}{}_i\to\inf}\sigma\left(\mathrm{NET}^{[2]}{}_i\right)\right)z_i$$
$$= -2(1-1)z_i = -2(0)z_i = 0$$

So, whenever NET values get large, the delta of the last layer will go to zero, making the gradient very small and learning comes to a stop.
A solution would be to use as an activation function, some function that does not asymptote for large values like ReLU for instance where the gradient is always one for NET values larger than 0.

## 2. [6 pts] **Clustering, regression and PCA**

Consider the following observations in a Euclidean space:

| | $y_1$ | $y_2$ | $z$ |
|---|---|---|---|
| $\mathbf{x}_1$ | 0 | 0 | 0.1 |
| $\mathbf{x}_2$ | 1 | 2 | 0.4 |
| $\mathbf{x}_3$ | 2 | 2 | 0.4 |
| $\mathbf{x}_4$ | 4 | 4 | 0.2 |

**a)** [1.5v] Using input variables, identify the $k$-means clustering solution with $k = 3$, and $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ observations as initial centers (seeds). How many iterations are necessary for convergence?
*Hint*: you can compute k-means visually, indicating the results per iteration

Centroids after first iteration: $c_1 = \mathbf{x}_1,\ c_2 = \mathbf{x}_2,\ c_3 = \begin{pmatrix}3\\3\end{pmatrix}$

Centroids after second iteration: $c_1 = \mathbf{x}_1,\ c_2 = \begin{pmatrix}1.5\\2\end{pmatrix},\ c_3 = \mathbf{x}_4$

Centroids after third iteration: unchanged.

**b)** [1.2v] Compute the silhouette of the larger cluster.

$$silhouette(c2) = \frac{silhouette(x2) + silhouette(x3)}{2} = \frac{\left(1-\frac{1}{\sqrt{5}}\right)+\left(1-\frac{1}{\sqrt{8}}\right)}{2}$$

**c)** [1.2v] Estimate the quantities of training observations, $\hat{z} = g(\mathbf{x})$ where $g$ is given by a linear regression model with $\phi(\mathbf{x}) = \|\mathbf{x}\|_1$ and $\mathbf{w} = (-0.2, 0.2)^T$

$$\Phi = \begin{pmatrix} 0 \\ 3 \\ 4 \\ 8 \end{pmatrix} \text{ and } \hat{\mathbf{z}} = \begin{pmatrix} -0.2 + 0.2 \times 0 \\ -0.2 + 0.2 \times 3 \\ -0.2 + 0.2 \times 4 \\ -0.2 + 0.2 \times 8 \end{pmatrix} = \begin{pmatrix} -0.2 \\ 0.4 \\ 0.6 \\ 1.4 \end{pmatrix}$$

**d)** [0.6v] Using the estimates, identify the training root mean squared error (RMSE) of $g$.

$$RMSE = \sqrt{\frac{1}{4}((0.1 + 0.2)^2 + (0.4 - 0.4)^2 + (0.4 - 0.6)^2 + (0.2 - 1.4)^2)}$$

**e)** [1.5v] The following covariance matrix and eigenvectors were produced for the given dataset:

$$C = \begin{pmatrix} 2.917 & 2.667 \\ 2.667 & 2.667 \end{pmatrix}, \quad \mathbf{u}_A = \begin{pmatrix} -0.690 \\ 0.723 \end{pmatrix}, \quad \mathbf{u}_B = \begin{pmatrix} 0.723 \\ 0.690 \end{pmatrix}$$

Which eigenvector, $\mathbf{u}_A$ or $\mathbf{u}_B$, explains more data variability?

$$C\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

Solving these simple equations yield $\lambda_A = 0.122 \; and \; \lambda_B = 5.461$.
Eigenvector $\mathbf{u}_B$ explains more variation.

## 3. [5 pts] **Bayes and tree learning**

Consider a decision tree learned from a dataset with two binary input variables and 10 observations. The following association rules define this decision tree:

- If $y_1 = 0$ then $class = Positive$ (3 observations correctly predicted at this leaf)
- If $y_1 = 1 \wedge y_2 = 0$ then $class = Positive$ (2 observations correctly predicted and 2 observations incorrectly predicted at this leaf)
- If $y_1 = 1 \wedge y_2 = 1$, then $class = Negative$ (3 observations correctly predicted at this leaf)

**a)** [1.5v] Compute the confusion matrix and precision of the provided decision tree.

$$TP = 5, TN = 3, FP = 2, FN = 0; \; precision = TP/(TP + FP) = 7/5$$

**b)** [2v] Given $\mathbf{x} = [1,0]^T$, estimate $P(\mathbf{x} \mid Positive)$. *Hint*: recover the original data.

$$p(\mathbf{x}) = \frac{4}{10}, \quad p(Positive) = \frac{1}{2}, \quad p(\mathbf{x}|Positive) = \frac{2}{5}, \quad P(Positive \mid \mathbf{x}) = \frac{1/2 \times 2/5}{4/10} = 0.5$$

**c)** [1v] Classify $\mathbf{x} = [0,0]^T$ using $k$NN with Hamming distance, $k = 7$ and uniform weights.

Given Hamming distance, the 7-NN has 5 positive and 2 negative observations. Hence, positive.

**d)** [1v] Is the given data separable by a SVM with a RBF kernel? Justify

No. Presence of collocated observations with different classes. No (kernel-based) hyperplane is able to separate.

## 4. [2 pts] **Model complexity**

Consider the regression problem of estimating the spam degree of a document described by $m$ features.

Estimate the complexity of a cluster-based RBF network, and a MLP with two hidden layers of 10 nodes each. Both networks have RELU activations on hidden and output nodes.

Identify the number of clusters, $k$, as a function of the number of features, that guarantees that the RBF network has lower (parameter) complexity than the MLP.

For every hidden neuron: m (centroid) + 1 (optional variance parameter)
Complexity of RBF network: $(m+1)k+1$
Complexity of MLP network: $(10+1)m+(10+1)\times 10+11$
Condition $k+1<\frac{11m+121}{m+1} \Leftrightarrow k<\frac{11m+121}{m+1}-1$

## 5. [1.5 pts] **Deep learning**

Consider a multilayer perceptron for classification with $H$ hidden layers. Assume that all weights and biases are initialized to zero. Now consider the following scenarios:

- **i.** all activation functions are equal to $f(net)=net^2$ and the error function is squared error
- **ii.** all activation functions are equal to $f(net)=e^{net}$ and the error function is squared error

For each scenario, what do you predict will happen? Is learning possible? Why?
Justify by computing the *deltas* and one short sentence.

For **scenario 1** a delta of the final layer will be given by:

$$\delta_i^{[H+1]}=\frac{\partial Err}{\partial a_i^{[H+1]}}\frac{\partial a_i^{[H+1]}}{\partial NET_i^{[H+1]}}=\frac{\partial Err}{\partial a_i^{[H+1]}}\frac{\partial f\left(NET_i^{[H+1]}\right)}{\partial NET_i^{[H+1]}}=\frac{1}{2}2\left(a_i^{[H+1]}-z_i\right)2NET_i^{[H+1]}=2\left(a_i^{[H+1]}-z_i\right)NET_i^{[H+1]}$$

Since all the weights and biases are zero, then $NET_i^{[H+1]}=0$.
Thus $\delta_i^{[H+1]}=0$. So, the delta of the last layer will always be zero.
The other deltas will follow the recursion $\delta_j^{[l]}=2NET_j^{[l]}W_{ji}^{[l+1]}\delta_i^{[l+1]}$.
So, they will also all be zero: $\delta_j^{[l]}=2NET_j^{[l]}W_{ji}^{[l+1]}0=0$
Substituting that on any of the gradient descent update rules, we will get:

$$W^{[l]}=W^{[l]}-\eta\frac{\partial Err}{\partial W^{[l]}}=W^{[l]}-\eta\delta^{[l]}a^{[l-1]^T}=W^{[l]}-\eta 0a^{[l-1]^T}=W^{[l]}$$

$$b^{[l]}=b^{[l]}-\eta\frac{\partial Err}{\partial b^{[l]}}=b^{[l]}-\eta\delta^{[l]}=b^{[l]}-\eta 0=b^{[l]}$$

Which implies that neither weights nor biases will ever be updated. Thus making learning impossible. □

For **scenario 2** a delta of the final layer will be given by:

$$\delta_i^{[H+1]}=\frac{\partial Err}{\partial a_i^{[H+1]}}\frac{\partial a_i^{[H+1]}}{\partial NET_i^{[H+1]}}=\frac{\partial Err}{\partial a_i^{[H+1]}}\frac{\partial f\left(NET_i^{[H+1]}\right)}{\partial NET_i^{[H+1]}}=\frac{1}{2}2\left(a_i^{[H+1]}-z_i\right)e^{NET_i^{[H+1]}}=\left(a_i^{[H+1]}-z_i\right)e^{NET_i^{[H+1]}}$$

Now, since all the weights and biases are zero, then $NET_i^{[H+1]}=0$
Thus $\delta_i^{[H+1]}=\left(a_i^{[H+1]}-z_i\right)e^{NET_i^{[H+1]}}=\left(a_i^{[H+1]}-z_i\right)e^0=\left(a_i^{[H+1]}-z_i\right)$.
So, the delta of the last layer will not be zero.
The other deltas will follow the recursion $\delta_j^{[l]}=e^{NET_j^{[l]}}W_{ji}^{[l+1]}\delta_i^{[l+1]}$.
Since the weights start at zero, in the first iteration, the other deltas will be zero: $\delta_j^{[l]}=e^{NET_j^{[l]}}0\delta_i^{[l+1]}=0$.
So, in the first iteration, only the last layer H+1 is updated. In the second iteration the weights of the final layer are no longer zero, so layer H will be updated. The third iteration layer H-1 will get updated and so on… Thus, learning is possible. □