Course: Numerical Analysis for Machine Learning

Prof. E. Miglio - February 7th 2023

Duration of the exam: 2.5 hours.

✓**Exercise 1**

Load the picture of Politecnico's facade `polimi_compressed.jpg`. Please notice that the RGB object is list of three matrices, each one corresponding to a different "channel" (i.e. red, greed, or blue).

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.image import imread

image_path = 'polimi_compress.jpg'
img = imread(image_path)

img_RGB = [np.array(img[:,:,i], dtype = np.float64)/255 for i in range(3)]

def plot_image(RGB, ax = None):
  if ax is None:
    _, ax = plt.subplots(1,1, figsize = (12,5))
  ax.imshow(np.clip(np.stack(RGB, axis = 2),0,1))
  plt.axis('off')

plot_image(img_RGB)
```

Now, randomly replace 70% of the pixels with random colors.

```
damage_fraction = 0.7

np.random.seed(0)
mask_remove = np.random.choice(a=[True, False], size=(img.shape[0],img.shape[1]), \
p=[damage_fraction, 1-damage_fraction])
mask_keep = np.logical_not(mask_remove)

img_damaged_RGB = [img_RGB[i].copy() for i in range(3)]
for i in range(3):
    img_damaged_RGB[i][mask_remove] = np.random.rand(np.sum(mask_remove))
```

✓1. Display the damaged image.

✓2. Implement the singular value truncation (SVT) algorithm to reconstruct the picture from `img_damaged_RGB`. (*Hint:* apply the algorithm independently to each channel.)

✓3. Try to optimize by trial and error the threshold on the singular values. Run the algorithm for 20 iterations and plot the resulting image against the original and the damaged one.

✓4. Comment on the impact of the threshold on the results.

??? **Exercise 2**

Explain how a perceptron works and how to apply the gradient descent method to optimize the parameters of the perceptron.

Consider the following set of data

$$
\begin{aligned}
\mathbf{x}_1 &= (2,-1), y_1 = 1,\\
\mathbf{x}_2 &= (-1,1), y_2 = 0,\\
\mathbf{x}_3 &= (2,0.5), y_3 = 1,\\
\mathbf{x}_4 &= (0.2,-0.2), y_4 = 0,\\
\mathbf{x}_5 &= (0.5,-1), y_5 = 1,\\
\mathbf{x}_6 &= (2,1), y_6 = 0.
\end{aligned} \tag{1}
$$

- Construct a possible separating hyperplane by hand. Express the parameters of the hyperplane in terms of the three parameters of the perceptron.
- Implement and apply the perceptron algorithm using $\mathbf{w}_0 = (1,0)$ and $b_0 = 0$. Consider a learning rate $\eta = 2$. Draw on a figure the obtained hyperplane along with the data.
- Apply again the algorithm with $\eta = 0.5$. Draw on a figure the obtained hyperplane along with the data.

✓**Exercise 3**

Suppose that the output $\hat{y}_k$ of a given unit in a neural network is given by the softmax function *i.e.*:

$$\hat{y}_k = \frac{\exp(a_k)}{\sum_j \exp(a_j)}. \tag{2}$$

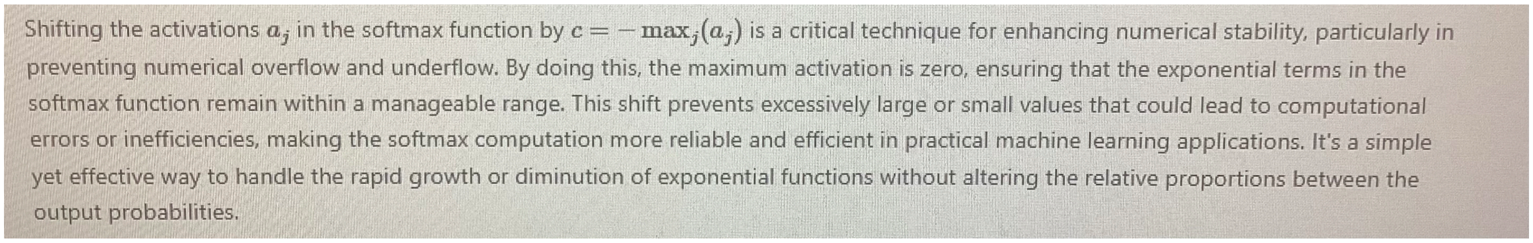
- ✓ Show that the output of the softmax function does not change if you shift, in all components, the activations a_j by some constant c.
- ✓ Explain why the shift $c = -\max_j(a_j)$ can be useful.

- evaluate the softmax in $\bar{a}_j = a_j + c$

$$\hat{\bar{y}}_k = \frac{e^{\bar{a}_k}}{\sum e^{\bar{a}_j}} = \frac{e^{a_k + c}}{\sum e^{a_j + c}} = \frac{e^c e^{a_k}}{e^c \sum e^{a_j}} = \frac{e^{a_k}}{\sum e^{a_j}} = \hat{y}_k$$

They are the same!

-



Shifting the activations a_j in the softmax function by $c = -\max_j(a_j)$ is a critical technique for enhancing numerical stability, particularly in preventing numerical overflow and underflow. By doing this, the maximum activation is zero, ensuring that the exponential terms in the softmax function remain within a manageable range. This shift prevents excessively large or small values that could lead to computational errors or inefficiencies, making the softmax computation more reliable and efficient in practical machine learning applications. It's a simple yet effective way to handle the rapid growth or diminution of exponential functions without altering the relative proportions between the output probabilities.