**Course: Numerical Analysis for Machine Learning**

Prof. E. Miglio - July 8th 2022

Duration of the exam: 2.5 hours.

**Exercise 1**

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars, belonging to three different customer segments. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data',
       header = None)
A = np.array(data)[:,1:].astype(np.float64).T # matrix containing the data (num features x num wines)
labels = np.array(data)[:,0].astype(np.int32) # the label of each wine (i.e. customer segment)
groups = (1,2,3) # customer segments: 1 = low quality; 2 = medium quality; 3 = high quality
features = ['Alcohol', 'Malic_Acid', 'Ash', 'Ash_Alcanity', 'Magnesium', 'Total_Phenols',
            'Flavanoids', 'Nonflavanoid_Phenols', 'Proanthocyanins', 'Color_Intensity',
            'Hue', 'OD280', 'Proline'] # features descriptions
```

**1**. How many features? How many samples? How many samples belong to each customer segment?

**2**. Normalize the data so that each feature has zero mean and unitary standard deviation.

**3**. Perform PCA on the dataset by means of the SVD decomposition. Then, plot the trend of:

- the singular value $\sigma_k$;
- the cumulate fraction of singular values: $\frac{\sum_{i=1}^{k} \sigma_i}{\sum_{i=1}^{q} \sigma_i}$;
- the fraction of the "explained variance": $\frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{q} \sigma_i^2}$.

Comment on the results.

**4**. Compute a matrix containing the principal components associated with the dataset.

**5**. Generate a scatterplot of the first two principal components of the dataset, grouped by label. Draw a line that separates, as well as you manage, the category of low customer segment wines from the rest of the wines (chose intercept and slope of the line by hand, trying to achieve a good result).

**6**. Based on the line of point 4, define a predictor to detect wines of low customer segment. Then, compute the number of true positives (TP), false positives (FP), true negatives (TN), false negatives (FN). Finally, compute:

- sensitivity: (TP) / (TP+FN);
- specificity: (TN) / (TN+FP);
- accuracy: (TP+TN) / (TP+TN+FP+FN).

**Exercise 2**

Consider the minimization of the 2D quadratic function:

$$f(\mathbf{x}) = \frac{1}{2}(x_1^2 + \eta x_2^2), \tag{1}$$

where $\mathbf{x} = (x_1, x_2)$ and $\eta > 0$.

1. Use the `contourf` command to plot the contourlines of the function for different values of $\eta$; use the following syntax:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(xmin,xmax,Nx)
y = np.linspace(ymin,ymax,Ny)
[XX,YY] = np.meshgrid(x,y)
F = ( XX ** 2 + eta * YY ** 2 ) / 2
plt.contourf(XX,YY,F,Ncontour)
```

2. How does $\eta$ impact on the numerical solution of the problem ?

3. Let us consider the gradient descent (GD) algorithm; the iteration relation is given by

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \tau_k \nabla f(\mathbf{x}^{(k)}), \tag{2}$$

with $\tau_k > 0$. In the convex case, if $f$ is of class $C^2$, in order to have convergence, we must have

$$0 < \tau_k < \frac{2}{\sup_x \|Hf(\mathbf{x})\|}. \tag{3}$$

where $Hf(\mathbf{x})$ is the Hessian of $f$. Compute the maximum value of $\tau_k$ to have convergence.

4. Implement the GD method with constant step size. Assume $\eta = 4$ and $\mathbf{x}_0 = (0.9, 0.3)$. Display the iterations, on the contour figure, for different values of the step size. Comment on the results.

5. Consider the *exact line search method* for the dynamic choice of the step size. In this method at each step we choose

$$\tau_k = \operatorname*{argmin}_s f(\mathbf{x}^{(k)} - s\nabla f(\mathbf{x}^{(k)})) \tag{4}$$

Compute the explict expression of $\tau_k$ given by the *exact line serach method* for the function (1).

6. Implement the GD method with variable step size using the expression derived at the previous point and apply the method to the minimization of function (1). Display the iterations, on the contour figure. Comment on the results.

**Exercise 3**

Consider a neural network in which a vectored node $v$ feeds into two distinct vectored nodes $h_1$ and $h_2$ computing different functions. The functions computed at the nodes are $h_1 = ReLU(W_1 v)$ and $h_2 = sigmoid(W_2 v)$. We do not know anything about the values of the variables in other parts of the network, but we know that $h_1 = [2,\,-1,\,3]^T$ and $h_2 = [0.2,\,0.5,\,0.3]^T$, that are connected to the node $v = [2,\,3,\,5,\,1]^T$. Furthermore the loss gradients are $\dfrac{\partial L}{\partial h_1} = [-2,\,1,\,4]^T$ and $\dfrac{\partial L}{\partial h_2} = [1,\,3,\,2]^T$, respectively. Show that the backpropagated loss gradient $\dfrac{\partial L}{\partial v}$ can be computed in terms of $W_1$ and $W_2$ as follows:

$$\frac{\partial L}{\partial v} = W_1^T \begin{bmatrix} -2 \\ 0 \\ 4 \end{bmatrix} + W_2^T \begin{bmatrix} 0.16 \\ 0.75 \\ -0.42 \end{bmatrix} \tag{5}$$

What are the sizes of $W_1, W_2$ and $\dfrac{\partial L}{\partial v}$ ?

Remember that $ReLU(x) = \max(0, x)$ and $sigmoid(x) = \dfrac{\exp(x)}{(\exp(x) + 1)}$.