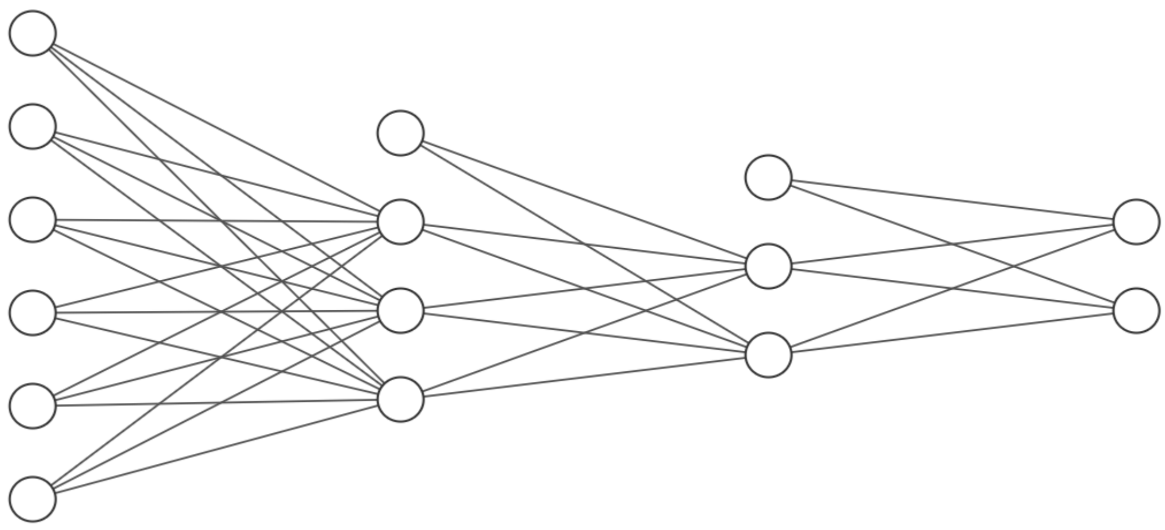
Aprendizagem 2021/22

Homework III – Group 6

I. Pen-and-paper

1) The Multi-layer Perceptron can be drawn as such:



The following are the original weight matrices ($W^{[i]}$) and bias vectors ($b^{[i]}$), the training example ($x = x^{[0]}$) and the target vector $\boldsymbol{t}$:

$$\boldsymbol{W}^{[1]} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix} \ \boldsymbol{b}^{[1]} = \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix} \ \boldsymbol{W}^{[2]} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix} \ \boldsymbol{b}^{[2]} = \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \end{bmatrix} \ \boldsymbol{W}^{[3]} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \ \boldsymbol{b}^{[3]} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \ \boldsymbol{x} = \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix} \ \boldsymbol{t} = \begin{bmatrix} \mathbf{1} \\ \mathbf{-1} \end{bmatrix}$$

a) To perform a stochastic gradient descent update, we must first do a forward propagation with the training example and then a backward propagation to update the weights and bias and improve the classifier.

Forward propagation:

$$\boldsymbol{z}^{[1]} = \boldsymbol{W}^{[1]}\boldsymbol{x}^{[0]} + \boldsymbol{b}^{[1]} = \begin{bmatrix} 6 \\ 1 \\ 6 \end{bmatrix} \quad \boldsymbol{x}^{[1]} = \begin{bmatrix} tanh\ 6 \\ tanh\ 1 \\ tanh\ 6 \end{bmatrix} \approx \begin{bmatrix} \mathbf{0.999988} \\ \mathbf{0.761594} \\ \mathbf{0.999988} \end{bmatrix}$$

$$\boldsymbol{z}^{[2]} = \boldsymbol{W}^{[2]}\boldsymbol{x}^{[1]} + \boldsymbol{b}^{[2]} = \begin{bmatrix} 2tanh(6) + tanh(1) + 1 \\ 2tanh(6) + tanh(1) + 1 \end{bmatrix} \approx \begin{bmatrix} \mathbf{3.761570} \\ \mathbf{3.761570} \end{bmatrix} \ \boldsymbol{x}^{[2]} = \begin{bmatrix} tanh(2tanh(6) + tanh(1) + 1) \\ tanh(2tanh(6) + tanh(1) + 1) \end{bmatrix} \approx \begin{bmatrix} \mathbf{0.998920} \\ \mathbf{0.998920} \end{bmatrix}$$

$$\boldsymbol{z}^{[3]} = \boldsymbol{W}^{[3]}\boldsymbol{x}^{[2]} + \boldsymbol{b}^{[3]} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \quad \boldsymbol{x}^{[3]} = \begin{bmatrix} tanh\ 0 \\ tanh\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

Back propagation:

$$\boldsymbol{E}(\boldsymbol{x}^{[3]}, \boldsymbol{t}) = \frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{x}^{[3]} - \boldsymbol{t})^2 \Rightarrow \frac{\boldsymbol{dE}}{\boldsymbol{dx}^{[3]}} = \boldsymbol{x}^{[3]} - \boldsymbol{t}$$

$$\boldsymbol{\delta}^{[3]} = \frac{\boldsymbol{dE}}{\boldsymbol{dz}^{[3]}} = \frac{dE}{dx^{[3]}}\frac{dx^{[3]}}{dz^{[3]}} = (x^{[3]} - t) \odot (1 - \tanh^2(z^{[3]})) = \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) \odot \left(1 - \tanh^2\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)\right) = \begin{bmatrix} \mathbf{-1} \\ \mathbf{1} \end{bmatrix}$$

$$\boldsymbol{\delta}^{[2]} = \frac{\boldsymbol{dE}}{\boldsymbol{dz}^{[2]}} = \frac{dE}{dx^{[3]}}\frac{dx^{[3]}}{dz^{[3]}}\frac{dz^{[3]}}{dx^{[2]}}\frac{dx^{[2]}}{dz^{[2]}} = \left[\frac{dz^{[3]}}{dx^{[2]}}\right]^{\mathrm{T}} \delta^{[3]} \odot \frac{dx^{[2]}}{dz^{[2]}} = W^{[3]}\delta^{[3]} \odot (1 - \tanh^2(z^{[2]}))$$
$$= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} -1 \\ 1 \end{bmatrix} \odot \left(1 - \tanh^2\left(\begin{bmatrix} 3.761570 \\ 3.761570 \end{bmatrix}\right)\right) = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

$$\boldsymbol{\delta^{[1]}} = \boldsymbol{\frac{dE}{dz^{[1]}}} = \frac{dE}{dx^{[3]}}\frac{dx^{[3]}}{dz^{[3]}}\frac{dz^{[3]}}{dx^{[2]}}\frac{dx^{[2]}}{dz^{[2]}}\frac{dz^{[2]}}{dx^{[1]}}\frac{dx^{[1]}}{dz^{[1]}} = \left[\frac{dz^{[2]}}{dx^{[1]}}\right]^{\mathrm{T}} \delta^{[2]} \odot \frac{dx^{[1]}}{dz^{[1]}} = W^{[2]}\delta^{[2]} \odot \left(1 - \tanh^2\left(z^{[1]}\right)\right)$$
$$= \begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix} \odot \left(1 - \tanh^2\left(\begin{bmatrix}6\\1\\6\end{bmatrix}\right)\right) = \begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}$$

$$\boldsymbol{W^{[1]}_{new}} = \boldsymbol{W^{[1]}_{old}} - \boldsymbol{\eta}\boldsymbol{\frac{dE}{dW^{[1]}_{old}}} = W^{[1]}_{old} - \eta\left(\delta^{[1]}x^{[0]^T}\right) = \begin{bmatrix}1 & 1 & 1 & 1 & 1\\ 0 & 0 & 0 & 0 & 0\\ 1 & 1 & 1 & 1 & 1\end{bmatrix} - 0.1\begin{bmatrix}0\\0\\0\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix}^T = \begin{bmatrix}\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1}\end{bmatrix}$$

$$\boldsymbol{b^{[1]}_{new}} = \boldsymbol{b^{[1]}_{old}} - \boldsymbol{\eta}\boldsymbol{\frac{dE}{db^{[1]}_{old}}} = b^{[1]}_{old} - \eta\delta^{[1]} = \begin{bmatrix}1\\1\\1\end{bmatrix} - 0.1\begin{bmatrix}0\\0\\0\end{bmatrix} = \begin{bmatrix}\mathbf{1}\\ \mathbf{1}\\ \mathbf{1}\end{bmatrix}$$

$$\boldsymbol{W^{[2]}_{new}} = \boldsymbol{W^{[2]}_{old}} - \boldsymbol{\eta}\frac{dE}{d\boldsymbol{W^{[2]}_{old}}} = W^{[2]}_{old} - \eta\left(\delta^{[2]}x^{[1]^T}\right) = \begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} - 0.1\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0.999988\\0.761594\\0.999988\end{bmatrix}^T = \begin{bmatrix}\mathbf{1} & \mathbf{1} & \mathbf{1}\\ \mathbf{1} & \mathbf{1} & \mathbf{1}\end{bmatrix}$$

$$\boldsymbol{b^{[2]}_{new}} = \boldsymbol{b^{[2]}_{old}} - \boldsymbol{\eta}\boldsymbol{\frac{dE}{db^{[2]}_{old}}} = b^{[2]}_{old} - \eta\delta^{[2]} = \begin{bmatrix}1\\1\end{bmatrix} - 0.1\begin{bmatrix}0\\0\end{bmatrix} = \begin{bmatrix}\mathbf{1}\\ \mathbf{1}\end{bmatrix}$$

$$\boldsymbol{W^{[3]}_{new}} = \boldsymbol{W^{[3]}_{old}} - \boldsymbol{\eta}\boldsymbol{\frac{dE}{dW^{[3]}_{old}}} = W^{[3]}_{old} - \eta\left(\delta^{[3]}x^{[2]^T}\right) = \begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix} - 0.1\begin{bmatrix}-1\\1\end{bmatrix}\begin{bmatrix}0.998920\\0.998920\end{bmatrix}^T = \begin{bmatrix}\mathbf{0.099892} & \mathbf{0.099892}\\ \mathbf{-0.099892} & \mathbf{-0.099892}\end{bmatrix}$$

$$\boldsymbol{b^{[3]}_{new}} = \boldsymbol{b^{[3]}_{old}} - \boldsymbol{\eta}\boldsymbol{\frac{dE}{db^{[3]}_{old}}} = b^{[3]}_{old} - \eta\delta^{[3]} = \begin{bmatrix}0\\0\end{bmatrix} - 0.1\begin{bmatrix}-1\\1\end{bmatrix} = \begin{bmatrix}\mathbf{0.1}\\ \mathbf{-0.1}\end{bmatrix}$$

b) The forward propagation stage is very similar to that of **a)**, with the exception that, in the final layer, the activation is $\boldsymbol{softmax(x)}$, which is defined such that, for a given vector $\boldsymbol{x = (x_1, \dots, x_n)}$:

$$\boldsymbol{softmax(x)_i} = \frac{\boldsymbol{e^{x_i}}}{\boldsymbol{\sum_{j=1}^{n} e^{x_j}}}, i = 1, \dots, n$$

As such, only $x^{[3]}$ differs between both **forward propagations**:

$$x^{[3]} = softmax\begin{bmatrix}0\\0\end{bmatrix} = \begin{bmatrix}\dfrac{e^0}{e^0 + e^0}\\[2ex] \dfrac{e^0}{e^0 + e^0}\end{bmatrix} = \begin{bmatrix}0.5\\0.5\end{bmatrix}$$

Back propagation:

$$\boldsymbol{E}\left(x^{[3]}, t\right) = -\sum_{\boldsymbol{k}} \boldsymbol{t_k\, ln\left(x^{[3]}_k\right)}$$

$$\boldsymbol{\frac{dx^{[3]}_k}{dz^{[3]}_i}} = \begin{cases}\boldsymbol{x^{[3]}_k\left(1 - x^{[3]}_k\right)} & \boldsymbol{if\ k = i}\\ \boldsymbol{-x^{[3]}_k x^{[3]}_i} & \boldsymbol{if\ k \neq i}\end{cases}$$

$$\boldsymbol{\delta_i^{[3]}} = \frac{\boldsymbol{dE}}{\boldsymbol{dz_i^{[3]}}} = \frac{d}{dz_i^{[3]}}\left(-\sum_k t_k\, ln\left(x_k^{[3]}\right)\right) = -\sum_k t_k \frac{d}{dz_i^{[3]}}\left(ln\left(x_k^{[3]}\right)\right) = -\sum_k \frac{t_k}{x_k^{[3]}}\frac{dx_k^{[3]}}{dz_i^{[3]}}$$

$$= -\sum_{k=i} \frac{t_k x_k^{[3]}\left(1 - x_k^{[3]}\right)}{x_k^{[3]}} + \sum_{k\neq i} \frac{t_k x_k^{[3]} x_i^{[3]}}{x_k^{[3]}} = t_k\left(x_i^{[3]} - 1\right) + x_i^{[3]}\sum_{k\neq i} t_k = x_i^{[3]}\sum_k t_k - t_i$$

$$= \boldsymbol{x_i^{[3]}} - \boldsymbol{t_i}\ \left(since\ \sum_{\boldsymbol{k}} \boldsymbol{t_k} = \boldsymbol{1}\right) \Rightarrow \boldsymbol{\delta^{[3]}} = \boldsymbol{x^{[3]}} - \boldsymbol{t} = \begin{bmatrix}0.5\\0.5\end{bmatrix} - \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}\mathbf{-0.5}\\\mathbf{0.5}\end{bmatrix}$$

$$\boldsymbol{\delta^{[2]}} = \left[\frac{\boldsymbol{dz^{[3]}}}{\boldsymbol{dx^{[2]}}}\right]^{\mathbf{T}} \boldsymbol{\delta^{[3]}} \odot \frac{\boldsymbol{dx^{[2]}}}{\boldsymbol{dz^{[2]}}} = W^{[3]}\delta^{[3]} \odot \left(1 - \tanh^2\left(z^{[2]}\right)\right) = \begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}\begin{bmatrix}-0.5\\0.5\end{bmatrix} \odot \left(1 - \tanh^2\left(\begin{bmatrix}3.761570\\3.761570\end{bmatrix}\right)\right) = \begin{bmatrix}\mathbf{0}\\\mathbf{0}\end{bmatrix}$$

$$\boldsymbol{\delta^{[1]}} = \left[\frac{\boldsymbol{dz^{[2]}}}{\boldsymbol{dx^{[1]}}}\right]^{\mathbf{T}} \boldsymbol{\delta^{[2]}} \odot \frac{\boldsymbol{dx^{[1]}}}{\boldsymbol{dz^{[1]}}} = W^{[1]}\delta^{[2]} \odot \left(1 - \tanh^2\left(z^{[1]}\right)\right) = \begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix} \odot \left(1 - \tanh^2\left(\begin{bmatrix}6\\1\\6\end{bmatrix}\right)\right) = \begin{bmatrix}\mathbf{0}\\\mathbf{0}\\\mathbf{0}\end{bmatrix}$$

$$\boldsymbol{W_{new}^{[1]}} = \boldsymbol{W_{old}^{[1]}} - \boldsymbol{\eta}\frac{\boldsymbol{dE}}{d\boldsymbol{W_{old}^{[1]}}} = W_{old}^{[1]} - \eta\left(\delta^{[1]}x^{[0]^T}\right) = \begin{bmatrix}1 & 1 & 1 & 1 & 1\\0 & 0 & 0 & 0 & 0\\1 & 1 & 1 & 1 & 1\end{bmatrix} - 0.1\begin{bmatrix}0\\0\\0\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix}^T = \begin{bmatrix}\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1}\\\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1}\end{bmatrix}$$

$$\boldsymbol{b_{new}^{[1]}} = \boldsymbol{b_{old}^{[1]}} - \boldsymbol{\eta}\frac{\boldsymbol{dE}}{\boldsymbol{db_{old}^{[1]}}} = b_{old}^{[1]} - \eta\delta^{[1]} = \begin{bmatrix}1\\1\\1\end{bmatrix} - 0.1\begin{bmatrix}0\\0\\0\end{bmatrix} = \begin{bmatrix}\mathbf{1}\\\mathbf{1}\\\mathbf{1}\end{bmatrix}$$

$$\boldsymbol{W_{new}^{[2]}} = \boldsymbol{W_{old}^{[2]}} - \boldsymbol{\eta}\frac{\boldsymbol{dE}}{\boldsymbol{dW_{old}^{[2]}}} = W_{old}^{[2]} - \eta\left(\delta^{[2]}x^{[1]^T}\right) = \begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\end{bmatrix} - 0.1\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0.999988\\0.761594\\0.999988\end{bmatrix}^T = \begin{bmatrix}\mathbf{1} & \mathbf{1} & \mathbf{1}\\\mathbf{1} & \mathbf{1} & \mathbf{1}\end{bmatrix}$$

$$\boldsymbol{b_{new}^{[2]}} = \boldsymbol{b_{old}^{[2]}} - \boldsymbol{\eta}\frac{\boldsymbol{dE}}{\boldsymbol{db_{old}^{[2]}}} = b_{old}^{[2]} - \eta\delta^{[2]} = \begin{bmatrix}1\\1\end{bmatrix} - 0.1\begin{bmatrix}0\\0\end{bmatrix} = \begin{bmatrix}\mathbf{1}\\\mathbf{1}\end{bmatrix}$$

$$\boldsymbol{W_{new}^{[3]}} = \boldsymbol{W_{old}^{[3]}} - \boldsymbol{\eta}\frac{\boldsymbol{dE}}{\boldsymbol{dW_{old}^{[3]}}} = W_{old}^{[3]} - \eta\left(\delta^{[3]}x^{[2]^T}\right) = \begin{bmatrix}0 & 0\\0 & 0\end{bmatrix} - 0.1\begin{bmatrix}-0.5\\0.5\end{bmatrix}\begin{bmatrix}0.998920\\0.998920\end{bmatrix}^T = \begin{bmatrix}\mathbf{0.049946} & \mathbf{0.049946}\\\mathbf{-0.049946} & \mathbf{-0.049946}\end{bmatrix}$$

$$\boldsymbol{b_{new}^{[3]}} = \boldsymbol{b_{old}^{[3]}} - \boldsymbol{\eta}\frac{\boldsymbol{dE}}{\boldsymbol{db_{old}^{[3]}}} = b_{old}^{[3]} - \eta\delta^{[3]} = \begin{bmatrix}0\\0\end{bmatrix} - 0.1\begin{bmatrix}-0.5\\0.5\end{bmatrix} = \begin{bmatrix}\mathbf{0.05}\\\mathbf{-0.05}\end{bmatrix}$$

II. Programming and critical analysis

2) Using the code in the Appendix (random_state = 0, alpha = 3), we obtained the following confusion matrices:

Without early stopping:

		Predicted class	
		Positive	Negative
Real class	Positive	#TP = 226	#FN = 13
	Negative	#FP = 26	#TN = 418

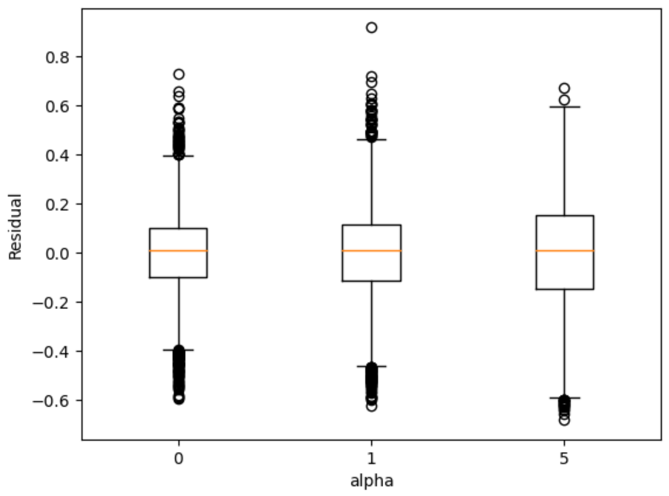
With early stopping:

		Predicted class	
		Positive	Negative
Real class	Positive	#TP = 234	#FN = 5
	Negative	#FP = 54	#TN =390

Although we expected a higher accuracy when using early stopping (since its utility is based on reducing generalisation errors), that wasn't the case. This can be explained by two factors:

- The reduced size of the dataset implies that the validation set, needed by the early stopping method for each fold to evaluate the number of epochs that return the minimum error, i.e., maximum accuracy, is not big enough to reflect the patterns of the whole set. As such, the early stopping point might easily be miscalculated, resulting in a lower accuracy when using the testing set. Additionally, the training set itself becomes even smaller, since the validation set is a subset of it, further increasing the chances of generalisation errors;
- While evaluating the minimum error, it is possible that the model found a local minimum, instead of a global one, which results in a premature stop of the training and leads to worse accuracies.

3) Using the code in the Appendix we obtained the following boxplots (using alpha ∈ {0, 1, 5}):



After analysing the boxplots, we concluded that, similarly to **2)**, the outputs didn't match our expectations. In this case, increasing the value of alpha didn't result in smaller residuals (increasing them, in fact). This problem can be mitigated with solutions such as:

- Increasing the MLP's number of iterations (the default is 200), in order to converge properly;
- Increasing the number of hidden layers, as using only 2 hidden layers might not be enough to efficiently train the MLP (e.g. the default is 100);
- Increasing the dataset size, so as to avoid generalisation errors;
- Apply perturbations to the weight matrices to allow the model to converge quicker, and reduce generalisation errors such as overfitting.

III. APPENDIX

```
#Imports

def loadDataFrame(file_name, flag):
    data = arff.loadarff(file_name)
    df = pd.DataFrame(data[0])
    if flag:
        df['Class'] = df['Class'].str.decode('utf-8')
    return df.values.tolist()

def splitFeatureLabel(df):
    df_features = [x[:-1] for x in df]
    df_labels = [x[-1] for x in df]
    return df_features, df_labels

def predict(alpha, early_stopping, df_features, df_labels, cv, MLP):
    clf = MLP(
        hidden_layer_sizes = (3,2),
        activation = 'relu',
        alpha = alpha,
        early_stopping = early_stopping,
        random_state = 0)
    return cross_val_predict(clf, df_features, df_labels, cv = cv)

def computeConfusionMatrix():
    def print_confusion_matrix(true,pred):
        tn, fp, fn, tp = confusion_matrix(true,pred).ravel()
        print("TN: " + str(tn))
        print("FN: " + str(fn))
        print("TP: " + str(tp))
        print("FP: " + str(fp))
        print("------------------------")
    cv1 = StratifiedKFold(n_splits = 5, shuffle = True, random_state = 0)
    df1 = loadDataFrame("breast.w.arff", True)
    df1_features, df1_labels = splitFeatureLabel(df1)
    pred_no_early_stopping = predict(3, False, df1_features, df1_labels, cv1, MLPClassifier)
    print_confusion_matrix(df1_labels, pred_no_early_stopping)
    pred_early_stopping = predict(3, True, df1_features, df1_labels, cv1, MLPClassifier)
    print_confusion_matrix(df1_labels, pred_early_stopping)

def computeBoxPlot():
    cv2 = KFold(n_splits = 5, shuffle = True, random_state = 0)
    df2 = loadDataFrame("kin8nm.arff", False)
    df2_features, df2_labels = splitFeatureLabel(df2)
    pred_no_reg = predict(0,False, df2_features, df2_labels, cv2, MLPRegressor)
    residuals_no_reg = [df2_labels[i] - pred_no_reg[i] for i in range(len(pred_no_reg))]
    pred_reg1 = predict(1, False, df2_features, df2_labels, cv2, MLPRegressor)
    residuals_reg1 = [df2_labels[i] - pred_reg1[i] for i in range(len(pred_reg1))]
    pred_reg5 = predict(5, False, df2_features, df2_labels, cv2, MLPRegressor)
    residuals_reg5 = [df2_labels[i] - pred_reg5[i] for i in range(len(pred_reg5))]
    plt.xlabel("alpha")
    plt.ylabel("Residual")
    plt.boxplot([residuals_no_reg,residuals_reg1,residuals_reg5], positions=[1, 2, 3], labels =
["0","1","5"])
    plt.savefig("boxplot.png")

if __name__ == "__main__":
    computeConfusionMatrix()
    computeBoxPlot()
```

END