1 Consider a three-layer neural network (Figure 3 in the ANN note) with the dimension $n = 2$ and $m = 1$, i.e., $\mathbf{f} : R^2 \to R$. Also use $\ell = 2$. Implement the back-propagation algorithm for the network in python. Test the algorithm for the training pair $\mathbf{x}_d = [0,1]^T$ and $y_d = 1$. Play with a step size η. The initial weights are $\alpha_1 = [\alpha_{11},\alpha_{12}]^T = [0.1,0.3]^T$, $\alpha_2 = [\alpha_{21},\alpha_{22}]^T = [0.3,0.4]^T$, and $\beta_1 = [\beta_{11},\beta_{12}]^T = [0.4,0.6]^T$. For activation, use the sigmoid function.

2 In this problem, you will use a neural network to predict if a product will be returned or not (backorder) for an online retail company using the 'backorders.csv' dataset. Carry out the following tasks:

Data Processing

(a) Drop the SKU column as it contains unique values, not adding any information to our model.

(b) Check for null values. Impute the null values with the mean value of the respective columnsif you find null values.

(c) Check for the correct data types of columns based on column descriptions (numerical orcategorical variables).

(d) Convert categorial columns to numerical values using One-Hot Encoding.

(e) Standardize (i.e., normalize, or scale to 0 to 1) numerical data. Note that it is always recommended to standardize the data for neural networks.

(f) Spilt the data in to train and test sets (90:10).

Model Building

You will use the Keras framework to build all models using the Sequential API. Please note that you have to specify the input dimension for the first layer in Keras. Feel free to experiment with number of Epochs (iterations).

- Plot train/development accuracy and loss against epochs for each model to track performance (you can use the inbuilt history dictionary from the Keras model).
- Use the validation split option while fitting the model to create a development set internally.
- Print the summary of all models to convince yourself of the architecture of the model and the number of weights the network has to learn.
- Report performance on test data.

Build the following models:

Model 1

- Number of layers (dense) = 1, neurons = 1, activation : Sigmoid
- Compile the model with: loss = binary cross-entropy, optimizer = 'sgd', metrics = accuracy
- Fit the model for 100 epochs (iterations) with default batch size.

- Can you think of another model you have built in previous homework that would give the same result as this network?

Model 2

- Number of layers (dense) = 2, neurons layer1 = 15, activation = tanh
- What should be the number of nodes and activation for the second layer given we are performing binary classification?
- Compile the model with: loss = binary cross-entropy, optimizer = 'sgd', metrics = acurracy
- Report results

Model 3

- 1st layer: 25 nodes, activation = tanh
- 2nd layer: 15 nodes, activation: tanh
- 3rd layer: 1 node, activation : sigmoid
- Report results

Model 4

- Do you notice overfitting of the model?
- Modify first two layers of the model to add a Regularizer using the L2 Regularizer option of the dense layer
- Feel free to experiment with weight decay (λ) values to try and reduce overfitting.
- Report results

3 Consider the Hepatitis dataset given (hepatitis.csv). The goal is to predict if the person will live or not given medical parameters of the person. You will build a SVM model for the binary classification. Carry out the following tasks:

Data Processing

Process data using the same steps as in Problem 2. Drop 'ID' column in this case.

Model Building

- Build a SVM model using Scikit-Learn (SVC - Support Vector Classifier) with the default parameters. Report performance on test data.
- Experiment with hyper-parameters (Slack and Kernels) for the SVM model using GridSearchCV function in Scikit-Learn. Report performance on test data using best model from tuning hyperparameters.