PMDS603P Deep Learning Lab Experiment 2

July 2025

1 Work to do today

Note: Make a single PDF file of the work you are doing in a Jupyter notebook. Upload with the proper format. Please mention your name and roll no properly with the Experiment number on the first page of your submission.

Question1. Today, we will try to recall the logistic regression model that you have seen in your Machine learning Course. With the given dataset 'liver_patient.csv' we will first fit a logistic regression model. Then we can see how a deep learning model framework can be used to create a model that does the same job.

• First we can try to import some necessary libraries and load the data to a dataframe. Here i have dropped the column Gender as well. Then i'm applying MinMaxScaler() to scale the data to the range 0 to 1.

```
import pandas as pd
import numpy as np
```

```
data = pd.read_csv('liver_patient.csv')
data.drop('Gender', axis=1, inplace = True)
from sklearn import preprocessing
MM = preprocessing.MinMaxScaler()
x = MM.fit_transform(data)
print(x)
```

Next we will split the data into training and testing set after deciding our input and output.

```
X = x[:,0:9]
Y = x[:,9]
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size=0.20,random_state=1)
```

Now we can use the logistic regression class from sklearn.linear_model and fit the model and find the accuracy associated with the model.

```
from sklearn.linear_model import LogisticRegression
logisticR = LogisticRegression()
logisticR.fit(X_train,y_train)
print(logisticR.predict(X_test))
from sklearn.metrics import accuracy_score
print(accuracy_score(logisticR.predict(X_test),y_test))
```

Now lets see how we can use the inbuild tools like keras and tensorflow to fit a simple deep learning model that does the same work. we will define a small network which has 9 neurons in the input layer, and one neuron in the output layer with the activation function as sigmoid.

```
import keras
from keras.models import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(1, activation='sigmoid', input_shape=(9,)))
model.summary()
```

Now you can see from the model summary how many parameters are learned in this model. Next is to compile the model by giving appropriate error function. Here we have a binary classification problem so we choose the error BinaryCrossEntropy. You also see optimizer which is given as SGD that is the nothing but the back propogation part where we choose the Stochastic gradient descent for updation or optimization of the parameters in the model.

```
from keras.optimizers import SGD
model.summary()
model.compile(loss='BinaryCrossentropy', optimizer='SGD',metrics=['accuracy'])
model.fit(X_train,y_train,batch_size =50, epochs=500,verbose=1,validation_data=(X_test, y_test))
```

Now you can see the validation set here is set as test set only. So the validation accuracy is nothing but the testing accuracy for your model that you see after training the model for 500 epochs or iterations. During the training process the weight updation is done batch after batch not by handling each training example in the dataset in the forward and backward pass technique. The batch size here we use is also a hyperparameter that can be tuned. you see that you get almost same accuracy from this deep learning model as well.

Now lets try the same problem by using two neurons in the output layer by using the concept of one hot encoding and train a model with 9 neurons in the input layer and 2 neurons in the output layer. So in this case we use the error function CategoricalCrossEntropy. But here we need to do the one hot encoding of the **y** labels for using this model.(And this same technique is used for multiclass classification problems).

```
y_train = keras.utils.to_categorical(y_train, 2)
y_test = keras.utils.to_categorical(y_test, 2)
```

The 2 here above represent the number of classes in your classification problem. The above function does the one-hot encoding for you.
Now lets define the model.

```
model = Sequential()
model.add(Dense(2, activation='softmax', input_shape=(9,)))
model.summary()
```

Now compile and fit the model to see the accuracy of the model.

```
from keras.optimizers import SGD
model.summary()
model.compile(loss='CategoricalCrossentropy', optimizer='SGD',metrics=['accuracy'])
model.fit(X_train,y_train,batch_size =50, epochs=500,verbose=1,validation_data=(X_test, y_test))
```

Now you can see the accuracy of this model as well. For having the prediction of your test set you can use this code. Here we use argmax function that just gives you the class with higher probability 0 or 1 class.

```
y_pred_probs = model.predict(X_test)
y_pred_classes = np.argmax(y_pred_probs, axis=1)
print(y_pred_classes)
```

If you look at this model

```
model = Sequential()
model.add(Dense(10, activation='sigmoid', input_shape=(9,)))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

Here we are planning to have a different architecture for the model. The model has 9 neurons in the input layer, 10 neurons in the hidden layer and one neuron in the output layer with all activation functions as sigmoid. Train this model and find the accuracy of the model for the same dataset.

Question:2 There are many activation function we can use in out process of creating models. Appart from the sigmoid activation function one can also use activation function like relu (which is defined as relu(z) = max(0,z)) in the intermediate layer neurons. Try to create a model which gives a better accuracy than the above ones you had by altering the architecture and activation functions. You can also try to change the batch_size in your training process.
Question:3 Next let us try a multi class classification problem where we will work with the mnist dataset. It constains 70000 handwritten images of digits from 0 to 9. So its a 10 class classification problem. Lets try to create a model that can do the classification task.

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import SGD
import matplotlib.pyplot as plt
batch_size = 128
num_classes = 10
epochs = 10
(x_train, y_train), (x_test, y_test) = mnist.load_data()
plt.imshow(x_train[3])
print(x_train[3])
```

Next let us reshape the whole images in to a single vector and normalize them.

```
x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
```

Now, when we feed the images in a particular fashion to our model. All the images have resolution of 28*28. So in our neural network we are going to create we will choose the input layer with 28*28 = 784 neurons and then two or three hidden layers and a final layer with 10 neurons. So the input shape in the model will be (784,).

Task: Now with all these informations and the details you have seen from the previous question come up with a model with better accuracy.

We dont have full knowledge with us to write this but i want you to attempt it to write this code. Where you need to define a class neural network which has methods like forwardpass, backwardpass and train. Figure out how we can do this.

Question:4 Next our task is to write a scratch code for a simple feed forward network that does a task. This model has inputs as [0, 0, 1], [0, 1, 1], [1, 0, 1], [1, 1, 1] and the expected output as [0], [1], [1], [0] in each case. So there are three features in our dataset as you see above. The activation function is to be taken as sigmoid. The architecture is like we have only one hidden layer and an output layer with one neuron. Take the error function as $(1/2)(y - \hat{y})^2$.