LAB 6

Exercise : Try logistic regression on BuyComputer dataset and set Random state=Your_RollNumber

In [1]:

```
import numpy as np
import torch.nn as nn
import pandas as pd
import io
import matplotlib.pyplot as plt
import torch
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

In [2]:

```
# Read Data
data = pd.read_csv("/home/nihar/Desktop/SEM 7/ML/Lab/Lab6/BuyComputer.csv")

data.drop(columns=['User ID',],axis=1,inplace=True)
data.head()
```

Out[2]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

In [3]:

```
y = data.iloc[:,-1].values
X = data.iloc[:,:-1].values
```

In [4]:

```
n_samples, n_features = X.shape

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran
dom_state=129)
```

In [5]:

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [6]:

```
X_train = torch.from_numpy(X_train.astype(np.float32))
X_test = torch.from_numpy(X_test.astype(np.float32))
y_train = torch.from_numpy(y_train.astype(np.float32))
y_test = torch.from_numpy(y_test.astype(np.float32))

y_train = y_train.view(y_train.shape[0], 1)
y_test = y_test.view(y_test.shape[0], 1)
```

In [7]:

```
# Linear model f = wx + b , sigmoid at the end
class Model(nn.Module):
    def __init__(self, n_input_features):
        super(Model, self).__init__()
        self.linear = nn.Linear(n_input_features, 1)

    def forward(self, x):
        y_pred = torch.sigmoid(self.linear(x))
        return y_pred

model = Model(n_features)
```

In [8]:

```
num_epochs = 140
learning_rate = 0.01
criterion = nn.BCELoss()
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)
```

In [13]:

```
# loop
for epoch in range(num_epochs):
    # Forward pass and loss
    y_pred = model(X_train)
    loss = criterion(y_pred, y_train)

    # Backward pass and update
    loss.backward()
    optimizer.step()

    # zero grad before new step
    optimizer.zero_grad()

    if (epoch+1) % 10 == 0:
        print(f'epoch: {epoch+1}, loss = {loss.item():.4f}')


with torch.no_grad():
    y_predicted = model(X_test)
    y_predicted_cls = y_predicted.round()
    acc = y_predicted_cls.eq(y_test).sum() / float(y_test.shape[0])
    print(f'\n\naccuracy: {acc.item()*100:.2f}')
```

```
epoch: 10, loss = 0.4300
epoch: 20, loss = 0.4286
epoch: 30, loss = 0.4272
epoch: 40, loss = 0.4259
epoch: 50, loss = 0.4246
epoch: 60, loss = 0.4234
epoch: 70, loss = 0.4222
epoch: 80, loss = 0.4210
epoch: 90, loss = 0.4198
epoch: 100, loss = 0.4187
epoch: 110, loss = 0.4176
epoch: 120, loss = 0.4166
epoch: 130, loss = 0.4155
epoch: 140, loss = 0.4145


accuracy: 83.75
```