COGS 188 - Assignment 1: Perceptron & KNN

You must submit this file in pdf or html version in the end on Canvas.

The goal of this assignment is to be able to use Python to run Percepetron and KNN on a breast cancer dataset.

First part. Build a Perceptron to do breast cancer prediction

In [1]:
```
from sklearn.datasets import load_breast_cancer
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import sklearn.datasets
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

%matplotlib inline
```

Load the breast cancer data.

In [2]:
```
breast_cancer = sklearn.datasets.load_breast_cancer()
```

In [3]:
```
data = pd.DataFrame(breast_cancer.data, columns = breast_cancer.feature_names)
data['class'] = breast_cancer.target
```

In [4]:
```
X = data.drop('class', axis = 1)
Y = data['class']
```

Separate training and testing data randomly.

In [5]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.1, stratify= Y, random_state = 1)
```

In [6]:
```
X_train = X_train.values
X_test = X_test.values
```

In [7]:
```
print(X_train.shape)
print(Y_train.shape)
print(X_test.shape)
print(Y_test.shape)
```

```
(512, 30)
(512,)
(57, 30)
(57,)
```

In [8]:
```
class Perceptron:
    def __init__(self):
        self.w = None
        self.b = None

    def model(self, x):
        # returns the prediction for a single example x
        #to be completed
        #print(self.w, x)
        return 1 if (np.dot(self.w, x) + self.b >= 0) else 0


    def predict(self, X):
        # returns the predictions for multiple examples X
        Y = []
        for x in X:
            res = self.model(x)
            Y.append(res)
        return np.array(Y)

    def fit(self, X, Y, epochs = 500, learning_rate = 0.01):
        accuracy = {}
        wt_matrix = []
        max_accuracy = 0
        self.w = np.ones(X.shape[1])
        self.b = 0

        for i in range(epochs):
            for x, y in zip(X, Y):
                y_pred = self.model(x)
                if y_pred == 1 and y == 0:
                    #Hint: use variable learning_rate
                    #Want to return 0 instead of 1
                    self.w = self.w - learning_rate*x #to be completed
                    self.b = self.b - learning_rate*1 #to be completed

                elif y_pred == 0 and y == 1:
                    #Want to return 1 istead of 0
                    self.w = self.w + learning_rate*x #to be completed
                    self.b = self.b + learning_rate*1 #to be completed

            wt_matrix.append(self.w)

            accuracy[i] = accuracy_score(self.predict(X), Y)
            if(accuracy[i] > max_accuracy):
                max_accuracy = accuracy[i]
```

```
                chkptw = self.w #to be completed
                chkptb = self.b #to be completed


        self.w = chkptw
        self.b = chkptb
        print(max_accuracy)
```

Initialize and train the Perceptron

In [9]:
```
perceptron = Perceptron()
```

In [10]:
```
perceptron.fit(X_train, Y_train, 1000, 0.1)
```

```
0.9375
```

Test the perceptron

In [11]:
```
Y_pred_test = perceptron.predict(X_test)
print(accuracy_score(Y_pred_test, Y_test))
```

```
0.9122807017543859
```

Second part. Use KNN to do malignant prediction

In [12]:
```
import pandas as pd
import numpy as np
from sklearn import neighbors, preprocessing
from sklearn import model_selection as cross_validate

data = pd.read_csv('breast-cancer-wisconsin.data')

# delete the unwanted id column
data.drop(['id'], 1, inplace=True)

# make up for missing entries
data.replace('?', -9999, inplace=True)

# get our attributes and classes in place
X = np.array(data.drop(['class'], 1))
y = np.array(data['class'])

# split data into training and testing sections
X_train, X_test, y_train, y_test = cross_validate.train_test_split(X, y, test_size=0.2)

# initialize our classifier
knn = neighbors.KNeighborsClassifier()

# fit the classifier with the training data
knn.fit(X_train, y_train) #to be completed


# calculating accuracy with test data
accuracy = knn.score(X_test, y_test)

# let's make a prediction
new_tests = np.array([[10, 10, 2, 3, 10, 2, 1, 8, 44], [10, 1, 12, 3, 1, 12, 1, 8, 12], [3, 1, 1, 3, 1, 12, 1,
2, 1]])
new_tests = new_tests.reshape(len(new_tests), -1)
prediction = knn.predict(new_tests)

# print out details
print ("Accuracy: ", accuracy)

print ("Predictions:")
for pred in prediction:
        if pred == 2:
                print(pred, "Benign")
        else: print(pred, "Malignant")
```

```
Accuracy:  0.9714285714285714
Predictions:
4 Malignant
4 Malignant
2 Benign
```