ps1_problem4

April 20, 2020

0.1 IDS/ACM/CS 158: Fundamentals of Statistical Learning

0.1.1 PS1, Problem 4: K-NN and Linear Regression for Classification

Name: Li, Michael

Email address: mlli@caltech.edu

Notes: Please use python 3.6

You are required to properly comment and organize your code.

- Helper functions (add/remove part label according to the specific question requirements)

[1]:
```
import numpy as np
import numpy.matlib

def load_data(filename):
    """
    filename - filename to open and load

    Returns file as matrix where last column is
    y_i and columns up to last one is x_i
    """
    res = np.loadtxt(open(filename, "rb"), delimiter=",", skiprows=1)
    return res

def linreg_regression(D, X):
    """
    D - training data consisting of pairs of p-dimensional vectors and output
    X - a column p-vector that represents a new input

    Returns the linear regression of X using D
    """

    x = D[:,:-1]
    y = D[:,-1]

    # add bias term to training data
    bias = np.matlib.repmat(1, len(x), 1)
```

```
    x = np.concatenate((bias, x), axis=1)

    # calculate beta
    intermediate = np.matmul(x.transpose(), x)
    inverse_intermediate = np.linalg.inv(np.array(intermediate))
    pseudo_x = np.matmul(inverse_intermediate, x.transpose())

    beta = np.matmul(pseudo_x, y)

    # apply beta weight to X
    return np.matmul(np.insert(X, 0, 1), beta)
```

- Part A

[2]:
```
def knn_classification(K, D, X):
    """
    K - number of neighbors
    D - training data consisting of pairs of p-dimensional vectors and outputs
    X - a column p-vector that represents a new input

    Returns the K-NN classification of X using D
    """

    train_x = D[:,:-1]
    train_y = D[:,-1]

    # find distances to X and sort points in D by that
    dists = np.sqrt(np.sum((train_x - np.matlib.repmat(X, len(train_x), 1))**2,␣
↪axis=1))
    inds = dists.argsort()

    # count the occurences of a class within first K and choose the most common␣
↪one
    nearest_neighbors = train_y[inds][:K].tolist()
    return max(nearest_neighbors, key=nearest_neighbors.count)
```

- Part B

[3]:
```
def linreg_classification(D, X):
    """
    D - training data consisting of pairs of p-dimensional vectors and output
    X - a column p-vector that represents a new input

    Returns the linear regression classification of X using D
    """

    if linreg_regression(D, X) < .5:
        return 0
```

```
    else:
        return 1
```

- Part C

I would expect linear regression classification to work better on dataset 3 since it looks mostly linearly separable with some noise. I would expect k-NN with K=1 to work better on dataset 4 since it does not appear to be linearly separable. It would be better in this case to just take the closest point in the training set since it doesn't seem like any line could split the data well.

[4]:
```
def knn_vs_linear_reg(train_filename, test_filename, dataset):
    """
    train_filename - filename of training data to load
    test_filename - filename of test data to load
    dataset - number of dataset

    Prints Results for KNN vs LinReg
    """
    K = 1
    training_data = load_data(train_filename)
    test_data = load_data(test_filename)

    test_x = test_data[:,:-1]
    test_y = test_data[:,-1]

    # run KNN and Linear Regression on all points in test dataset
    knn = [knn_classification(K, training_data, test_x[i]) for i in␣
↪range(len(test_x))]
    lr = [linreg_classification(training_data, test_x[i]) for i in␣
↪range(len(test_x))]

    # compute the zero-one loss of both models
    Err_knn = np.mean(np.not_equal(test_y, knn))
    Err_lr = np.mean(np.not_equal(test_y, lr))
    R = Err_knn / Err_lr

    print('For dataset {}\n Err_knn is {:1.4f} \n Err_lr is {:1.4f} \n R = {:1.
↪4f}'.format(dataset, Err_knn, Err_lr, R))
    if R > 1:
        print(' Linear regression is better.')
    else:
        print(' k-NN is better.')
```

[5]:
```
knn_vs_linear_reg('dataset3_train.csv', 'dataset3_test.csv', 3)
print()
knn_vs_linear_reg('dataset4_train.csv', 'dataset4_test.csv', 4)
```

```
For dataset 3
```

```
 Err_knn is 0.3410
 Err_lr is 0.2470
 R = 1.3806
 Linear regression is better.

For dataset 4
 Err_knn is 0.2040
 Err_lr is 0.2960
 R = 0.6892
 k-NN is better.
```