p5

August 7, 2018

1 DMG2 Assignment : Problem 5

k-Nearest Neighbours Classifier, Parzen Window Classifier

```
In [1]: import pandas as pd
        import numpy as np
        import os
        import matplotlib.pyplot as plt
        import seaborn as sns

        from sklearn.preprocessing import StandardScaler
        from sklearn.decomposition import PCA
        from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn import preprocessing
        from sklearn.neighbors import KernelDensity

        sns.set_style('whitegrid')

In [2]: DATA_DIR = '/home/jishnu/Documents/ISB/Term3/dmg2/assignments/hw_assignment1/dmg2/datas

In [3]: train = pd.DataFrame(columns=['V{}'.format(i) for i in range(1,785)] + ['label'])
        test = pd.DataFrame(columns=['V{}'.format(i) for i in range(1,785)] + ['label'])
        for num in range(10):
            # Consolidating training data
            temp_train = pd.read_csv(os.path.join(DATA_DIR,'train{0}.csv'.format(num)),usecols
            temp_train['label'] = num
            train = train.append(temp_train,ignore_index=True)
            # Consolidating test data
            temp_test = pd.read_csv(os.path.join(DATA_DIR,'test{0}.csv'.format(num)),usecols=[
            temp_test['label'] = num
            test = test.append(temp_test,ignore_index=True)

In [4]: train.shape

Out[4]: (36470, 785)

In [5]: test.shape
```

```
Out[5]: (24190, 785)
```

```
In [6]: train[train.isnull().any(axis=1)].groupby(by='label')['label'].value_counts()
```

```
Out[6]: label  label
        4      4         46
        5      5        299
        6      6          1
        8      8         42
        Name: label, dtype: int64
```

```
In [7]: test[test.isnull().any(axis=1)].groupby(by='label')['label'].value_counts()
```

```
Out[7]: label  label
        4      4         35
        5      5        203
        6      6          4
        8      8         30
        Name: label, dtype: int64
```

There are missing values in both the training and test data. Shown above is the count of rows with missing values, and the associated labels.

```
In [8]: train.groupby(by='label')['label'].value_counts()
```

```
Out[8]: label  label
        0      0        3567
        1      1        4034
        2      2        3582
        3      3        3677
        4      4        3567
        5      5        3567
        6      6        3567
        7      7        3763
        8      8        3567
        9      9        3579
        Name: label, dtype: int64
```

```
In [9]: test.groupby(by='label')['label'].value_counts()
```

```
Out[9]: label  label
        0      0        2356
        1      1        2708
        2      2        2376
        3      3        2454
        4      4        2356
        5      5        2356
        6      6        2356
        7      7        2502
        8      8        2356
        9      9        2370
        Name: label, dtype: int64
```

Considering the number of complete data for each label, we can safely remove the rows with missing values for our analysis.

```
In [10]: train = train.dropna()
         test = test.dropna()
```

```
In [11]: train.isnull().values.any()
```

```
Out[11]: False
```

```
In [12]: test.isnull().values.any()
```

```
Out[12]: False
```

There are no missing values in the training and test data now

```
In [13]: X_train = train.iloc[:,:784]
         Y_train = train.iloc[:,784]

         X_test = test.iloc[:,:784]
         Y_test = test.iloc[:,784]
```

```
In [14]: # Standardizing feature values
         X_train = StandardScaler().fit_transform(X_train)
         X_test = StandardScaler().fit_transform(X_test)
```

1.1 Applying PCA

```
In [15]: # Applying PCA
         pc = PCA(n_components=9).fit_transform(X_train)
         d1_train = pd.DataFrame(data=pc,columns=['pc{0}'.format(i) for i in range(1,10)])
         d1_train['label'] = Y_train.values
         d1_train.head(5)

         pc = PCA(n_components=9).fit_transform(X_test)
         d1_test = pd.DataFrame(data=pc,columns=['pc{0}'.format(i) for i in range(1,10)])
         d1_test['label'] = Y_test.values
         d1_test.head(5)
```

```
Out[15]:          pc1       pc2       pc3       pc4        pc5       pc6       pc7  \
         0   1.751833 -6.389755 -2.021087 -2.694718  -6.427682  1.025097 -0.554033
         1   5.884343 -7.690717 -2.390982  0.260438  -4.924087 -0.382575  0.298304
         2  16.381093  5.663541 -1.865832 -3.368391   4.353517 -3.098186 -5.629483
         3  12.814322 -7.638621 -4.434501 -7.660043  -0.550385 -1.785665 -0.076132
         4  11.123304  7.252218  5.007200 -0.333093  15.020012 -0.071782 -0.972011

                 pc8       pc9 label
         0  5.296349  3.807623     0
         1  6.510170  3.098220     0
         2 -3.894353  2.471996     0
         3  1.813952  0.309337     0
         4 -2.298600  0.127185     0
```

1.2 Applying Fisher LDA

```
In [16]: fisher = LinearDiscriminantAnalysis(n_components=9).fit_transform(X_train,Y_train.asty
         d2_train = pd.DataFrame(data=fisher,columns=['f{0}'.format(i) for i in range(1,10)])
         d2_train['label'] = Y_train.values
         d2_train.head(5)

         fisher = LinearDiscriminantAnalysis(n_components=9).fit_transform(X_test,Y_test.astype
         d2_test = pd.DataFrame(data=fisher,columns=['f{0}'.format(i) for i in range(1,10)])
         d2_test['label'] = Y_test.values
         d2_test.head(5)
```

```
/home/jishnu/anaconda3/lib/python3.6/site-packages/sklearn/discriminant_analysis.py:388: UserWa
  warnings.warn("Variables are collinear.")
/home/jishnu/anaconda3/lib/python3.6/site-packages/sklearn/discriminant_analysis.py:442: UserWa
  UserWarning)
/home/jishnu/anaconda3/lib/python3.6/site-packages/sklearn/discriminant_analysis.py:388: UserWa
  warnings.warn("Variables are collinear.")
```

```
Out[16]:          f1        f2        f3        f4        f5        f6        f7  \
         0 -2.833687 -1.079133 -0.845838 -0.764987 -0.484096  0.313129 -0.955681
         1 -3.849851 -3.435085 -1.935596  0.478846 -2.420500 -0.475621  0.572627
         2 -3.350865 -4.500847 -3.900589 -1.086583 -2.654538 -1.739049 -1.565408
         3 -3.283216 -2.503597 -3.138834 -0.103059 -1.582149 -0.295842  1.330582
         4 -2.231976 -1.552025 -3.889057 -0.017441 -0.577388 -2.299944 -0.684894

                  f8        f9 label
         0 -0.997196  1.076134     0
         1 -0.989022  0.779083     0
         2 -0.213139 -0.024020     0
         3 -3.239352  1.473339     0
         4  0.945457  0.404140     0
```

1.3 k-Nearest Neighbors Classification

```
In [17]: d1_train_X = d1_train.iloc[:,:9]
         d1_train_Y = d1_train.iloc[:,9].astype('int')

         d1_test_X = d1_test.iloc[:,:9]
         d1_test_Y = d1_test.iloc[:,9].astype('int')

         d2_train_X = d2_train.iloc[:,:9]
         d2_train_Y = d2_train.iloc[:,9].astype('int')

         d2_test_X = d2_test.iloc[:,:9]
         d2_test_Y = d2_test.iloc[:,9].astype('int')
```

```
In [18]: d1_knn = pd.DataFrame(columns=['k','acc_type','acc'])
         d2_knn = pd.DataFrame(columns=['k','acc_type','acc'])
```

```
In [19]: for k in range(1,18,2):
             knn1 = KNeighborsClassifier(n_neighbors=k).fit(d1_train_X,d1_train_Y)
             knn2 = KNeighborsClassifier(n_neighbors=k).fit(d2_train_X,d2_train_Y)
             d1_knn = d1_knn.append({'k' : k, 'acc_type' : 'training', 'acc' : np.round(knn1.sc
             d1_knn = d1_knn.append({'k' : k, 'acc_type' : 'test', 'acc' : np.round(knn1.score
             d2_knn = d2_knn.append({'k' : k, 'acc_type' : 'training', 'acc' : np.round(knn2.sc
             d2_knn = d2_knn.append({'k' : k, 'acc_type' : 'test', 'acc' : np.round(knn2.score
```

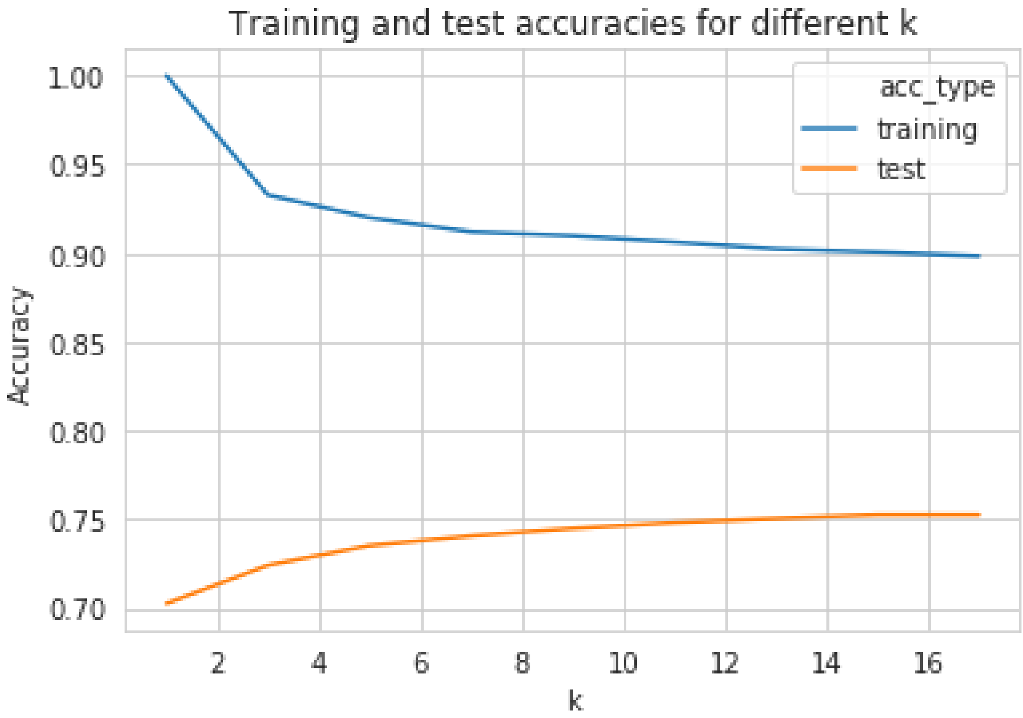
```
In [20]: d1_knn.head()
```

```
Out[20]:    k  acc_type     acc
         0  1  training  1.0000
         1  1      test  0.7025
         2  3  training  0.9328
         3  3      test  0.7241
         4  5  training  0.9200
```

1.3.1 Plotting training and test accuracy for kNN Classification

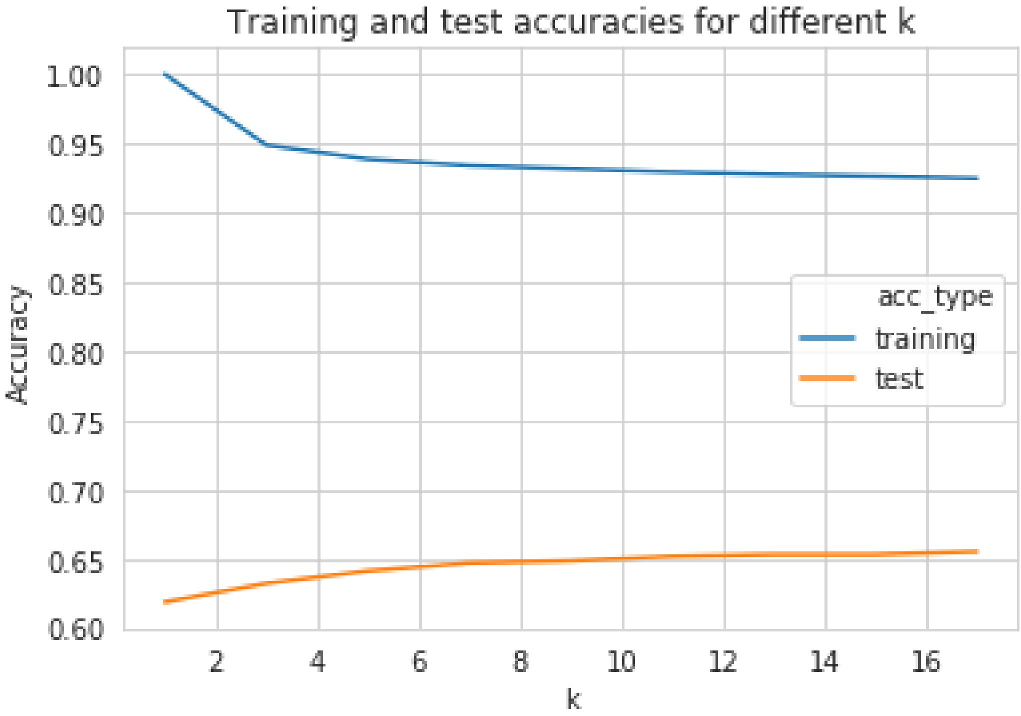
D1 Dataset

```
In [21]: sns.lineplot(x='k',y='acc',hue='acc_type',data=d1_knn,ci=0,)
         plt.title('Training and test accuracies for different k')
         plt.xlabel('k')
         plt.ylabel('Accuracy')
         plt.show();
```



D2 Dataset

```
In [22]: sns.lineplot(x='k',y='acc',hue='acc_type',data=d2_knn,ci=0,)
         plt.title('Training and test accuracies for different k')
         plt.xlabel('k')
         plt.ylabel('Accuracy')
         plt.show();
```



The optimal k for both datasets is 8 when considering the test accuracies.

1.4 Parzen-Window Classification

For each data point in test set, find the kernel function of the form $exp(-(X_i - X_t)^2/2\sigma^2)/\sigma$

The distance function used in euclidean, and the sum of kernel function value is found for all training data for each class to come up with the score of the class.

This score is converted to a probaibility to find the predicted class with maximum probability

Sampling values from each class for better performance

```
In [23]: d1_train.groupby('label', group_keys=False).count()
```

```
Out[23]:         pc1   pc2   pc3   pc4   pc5   pc6   pc7   pc8   pc9
         label
         0      3567  3567  3567  3567  3567  3567  3567  3567  3567
```

```
1      4034  4034  4034  4034  4034  4034  4034  4034  4034
2      3582  3582  3582  3582  3582  3582  3582  3582  3582
3      3677  3677  3677  3677  3677  3677  3677  3677  3677
4      3521  3521  3521  3521  3521  3521  3521  3521  3521
5      3268  3268  3268  3268  3268  3268  3268  3268  3268
6      3566  3566  3566  3566  3566  3566  3566  3566  3566
7      3763  3763  3763  3763  3763  3763  3763  3763  3763
8      3525  3525  3525  3525  3525  3525  3525  3525  3525
9      3579  3579  3579  3579  3579  3579  3579  3579  3579
```

```
In [24]: d1_test.groupby('label', group_keys=False).count()
```

```
Out[24]:        pc1   pc2   pc3   pc4   pc5   pc6   pc7   pc8   pc9
         label
         0      2356  2356  2356  2356  2356  2356  2356  2356  2356
         1      2708  2708  2708  2708  2708  2708  2708  2708  2708
         2      2376  2376  2376  2376  2376  2376  2376  2376  2376
         3      2454  2454  2454  2454  2454  2454  2454  2454  2454
         4      2321  2321  2321  2321  2321  2321  2321  2321  2321
         5      2153  2153  2153  2153  2153  2153  2153  2153  2153
         6      2352  2352  2352  2352  2352  2352  2352  2352  2352
         7      2502  2502  2502  2502  2502  2502  2502  2502  2502
         8      2326  2326  2326  2326  2326  2326  2326  2326  2326
         9      2370  2370  2370  2370  2370  2370  2370  2370  2370
```

There are around 3000 - 4000 data points for each class in train and 2000 - 3000 data points for each class in test. Let's sample 10 data points from each class

```
In [25]: d1_train = d1_train.groupby('label', group_keys=False).apply(lambda x: x.sample(min(le
         d1_test = d1_test.groupby('label', group_keys=False).apply(lambda x: x.sample(min(len

         d2_train = d2_train.groupby('label', group_keys=False).apply(lambda x: x.sample(min(le
         d2_test = d2_test.groupby('label', group_keys=False).apply(lambda x: x.sample(min(len

         d1_train_X = d1_train.iloc[:,:9]
         d1_train_Y = d1_train.iloc[:,9].astype('int')

         d1_test_X = d1_test.iloc[:,:9]
         d1_test_Y = d1_test.iloc[:,9].astype('int')

         d2_train_X = d2_train.iloc[:,:9]
         d2_train_Y = d2_train.iloc[:,9].astype('int')

         d2_test_X = d2_test.iloc[:,:9]
         d2_test_Y = d2_test.iloc[:,9].astype('int')
```

```
In [26]: def d1_train_ker(row_x,row_y,sigma):
             dist = np.linalg.norm(row_x - row_y)
             kernel_fn = np.exp(-dist**2/(2*sigma**2)) / sigma
```