DA_Homework_08_r09546042_TerryYang

May 16, 2021

1 Q1

1.0.1 import packages

```python
from PIL import Image
import numpy
from numpy import array
import numpy as np
from tkinter import _flatten
from sklearn.model_selection import train_test_split
from sklearn.preprocessing  import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model  import LogisticRegression
from sklearn import svm
```

1.0.2 reading ORL dataset

```python
image_matrix = np.zeros((400, 2576))
gender = np.array([])
for j in range(0, 40):
    for i in range(0, 10):
        image = Image.open(r"C:\Users\TerryYang\pythonwork\pythonwork\Data␣
 ↪Analytics Homework\ORL Faces\%s_%s.png" %(j+1, i+1))
        image_array = array(image)
        image_matrix[i+j*10] = image_array.flatten()
gender = [10*[0],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[0],10*[1],10*[0]
          ,10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1]
          ,10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1]
          ,10*[1],10*[0],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1]]
gender = list(_flatten(gender))
```

1.0.3 split and normalize the training, testing dataset

```python
X = image_matrix
y = gender

# Split data
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=1)

# Normalize
sc=StandardScaler()
sc.fit(X_train)
X_train_nor=sc.transform(X_train)
X_test_nor=sc.transform(X_test)
```

1.0.4 Logistic Regression

```
[4]: # Create LogisticRegression Model
     lr=LogisticRegression()
     lr.fit(X_train_nor,y_train)
     print("Accuracy of LR model: ",100*lr.score(X_test_nor, y_test),"%")
```

```
Accuracy of LR model:  96.25 %
```

```
C:\Users\TerryYang\anaconda3\envs\TENSORFLOW\lib\site-
packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

1.0.5 K-Nearest Neighbor

```
[5]: # Create KNN Model
     knn = KNeighborsClassifier(n_neighbors = 5)
     knn.fit(X_train_nor,y_train)
     print("Accuracy of KNN model: ",100*knn.score(X_test_nor, y_test),"%")
```

```
Accuracy of KNN model:  97.5 %
```

1.0.6 Support Vector Machine

```
[6]: # Create LogisticRegression Model
     clf=svm.SVC(kernel='linear',C=1)
     clf.fit(X_train,y_train)
     print("Accuracy of SVM model: ",100*clf.score(X_test_nor, y_test),"%")
```

```
Accuracy of SVM model:  88.75 %
```

1.0.7 Discussion:

1.0.8 KNN model (K=5) has better accuracy than LR, SVM model.

1.0.9 (The result might be different under different model settings.)

2 Q2

2.0.1 import packages

```
[7]: from sklearn.linear_model import Lasso
     import matplotlib.pyplot as plt
     import math
```

2.0.2 using LASSO reduce the size of data

```
[8]: lasso = Lasso(alpha = 0.001,normalize = True)
     lasso.fit(X, y)
```

```
[8]: Lasso(alpha=0.001, normalize=True)
```

2.0.3 finding & plotting down the important pixels

```
[9]: important_pixels = []
     for i in range(len(lasso.coef_)):
         if lasso.coef_[i] != 0:
             important_pixels.append(i)
     image = Image.open(r"C:\Users\TerryYang\pythonwork\pythonwork\Data Analytics␣
      ↪Homework\ORL Faces\1_1.png")
     img_array = np.array(image)
     #print(len(important_pixels),"important pixels at")
     for i in range(0, len(important_pixels)): #math.floor()
         col = math.floor(important_pixels[i]/46)
         row = important_pixels[i]-46*col
         #print("(",col,",", row,")")
         img_array[int(col)][int(row)]=255
     plt.imshow(img_array, interpolation='nearest')
     plt.show()
```
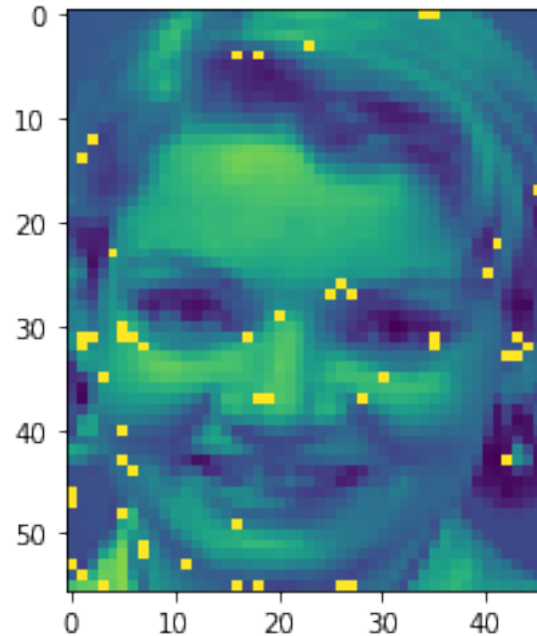
2.0.4 selecting important pixels

```
[10]: X_important = np.zeros((400, len(important_pixels)))
      X_important = X.T[important_pixels]
      X_important = X_important.T
```

2.0.5 split and normalize the training, testing dataset

```
[11]: # Split data
      X_important_train, X_important_test, y_train, y_test =␣
       ↪train_test_split(X_important, y, test_size=0.2, random_state=1)

      # Normalize
      sc=StandardScaler()
      sc.fit(X_important_train)
      X_important_train_nor=sc.transform(X_important_train)
      X_important_test_nor=sc.transform(X_important_test)
```

2.0.6 Logistic Regression

```
[16]: lr=LogisticRegression()
      lr.fit(X_important_train_nor,y_train)
      print("Accuracy of LR model after using LASSO reduction: ",100*lr.
       ↪score(X_important_test_nor, y_test),"%")
```

```
Accuracy of LR model after using LASSO reduction:  97.5 %
```

4

2.0.7 K-Nearest Neighbor

[17]:
```
# Create KNN Model
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_important_train_nor,y_train)
print("Accuracy of KNN model after using LASSO reduction: ",100*knn.
  ↪score(X_important_test_nor, y_test),"%")
```

Accuracy of KNN model after using LASSO reduction: 97.5 %

2.0.8 Support Vector Machine

[18]:
```
# Create LogisticRegression Model
clf=svm.SVC(kernel='linear',C=1)
clf.fit(X_important_train_nor,y_train)
print("Accuracy of SVM model after using LASSO reduction: ",100*clf.
  ↪score(X_important_test_nor, y_test),"%")
```

Accuracy of SVM model after using LASSO reduction: 96.25 %

2.0.9 Discussion:

2.0.10 Accuracy of LR, SVM model increased while using LASSO selecting the important pixels first.

2.0.11 (The result might be different under different model settings)