Lecture 13: Face Recognition Lab

CS 167: Machine Learning

Face Recognition

Face Recognition: classification task - identify a person from a photograph of their face

Our Data Set: Labeled Faces in the Wild from University of Massachusetts

http://vis-www.cs.umass.edu/lfw/

Labeled Faces in the Wild

All photos in this set are 250 $\times$ 250 pixels

Each pixel is an attribute: lots of attributes 250*250 pixels * 3 colors = 1875000

Convert to grayscale: 62500 total attributes

scikit-learn will load this dataset for you: by default converts to grayscale and crops/resizes to 62 $\times$ 47

Let's do it

```
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.cross_validation import train_test_split
from sklearn.datasets import fetch_lfw_people
from sklearn.metrics import accuracy_score
from sklearn.neural_network import MLPClassifier
import numpy

lfw_people = fetch_lfw_people(min_faces_per_person=70)
print(len(lfw_people.images))
print(lfw_people.target_names)
```

How many people do they have 40 or more photos of?

Looking at some of the data

```
image_num = 0 #try plugging in some other numbers too

print("image: ",lfw_people.images[image_num])

plt.imshow(lfw_people.images[image_num],cmap='gray')
plt.show()

img_target = lfw_people.target[image_num]
print(img_target)
print(lfw_people.target_names[img_target])

print("data: ", lfw_people.data[image_num])
print("data length: ",len(lfw_people.data[image_num]))
```

What's the difference between
`lfw_people.images` **and** `lfw_people.data`**?**
`lfw_people.target` **and** `lfw_people.target_names`**?**

Training, Testing

Note: the `train_test_split` is a little different because the predictive columns and target column are not in the same dataframe - actually, it's not even a dataframe but a `numpy` array

```
(train_data, test_data, train_target, test_target)\
    = train_test_split(lfw_people.data, lfw_people.target,\
                       test_size = 0.2)
mlp_classifier = MLPClassifier()
mlp_classifier.fit(train_data,train_target)
predictions = mlp_classifier.predict(test_data)
print(accuracy_score(predictions,test_target))
```

How did we do? Is that good? Is it overfitting?

See next slide for code to create a gallery of your predictions

Prediction Gallery

```
for i in range(len(predictions)):
    plt.imshow(test_data[i].reshape(62,47),cmap='gray')
    plt.show()
    print("predicted: ",\
       lfw_people.target_names[predictions[i]],\
       " actual: ", lfw_people.target_names[test_target[i]])
```

since we didn't split the `lfw_people.images` version, we have to reshape the `data` array to be like the `image` 2D array

Exercise: *Tweak the Neural Network parameters until you start getting good predictions. Make sure to try different network structures (number of layers and nodes within each layer)*

Let's figure out some other data

I've heard that these data sets don't necessarily work for linear models:

https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice

https://archive.ics.uci.edu/ml/datasets/Forest+Fires

Try the following

- Get a baseline accuracy/MSE with a linear model
- Try to find a good neural net.