kernel_classifier

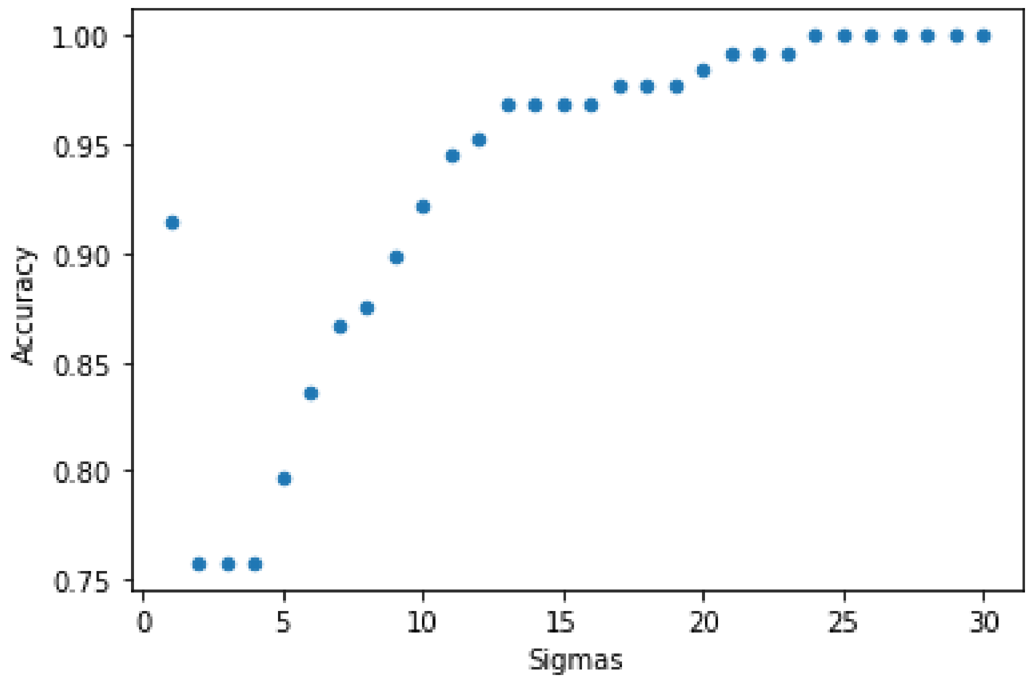
May 2, 2022

[16]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.io import loadmat
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
import random
from sklearn.model_selection import train_test_split

dataset = loadmat('face_emotion_data.mat')

X, y = dataset['X'], dataset['y']
n = len(X)

y[y==-1] = 0  # use 0/1 for labels instead of -1/+1
X = np.hstack((np.ones((n,1)), X))  # append a column of ones
n, p = np.shape(X)
```

[17]:
```python
# Train Classifier 1
sigmas = [i+1 for i in range(30)]
lam = 0.5

distsq=np.zeros((n,n),dtype=float)

for i in range(0,n):
    for j in range(0,n):
        d = np.linalg.norm(X[i,:]-X[j,:])
        distsq[i,j]=d**2

accuracy = []
Y_hat = np.zeros((n,n))
first = True
# Predict labels
for sigma in sigmas:
    y_hat = []
    K = np.exp(-distsq/(2*sigma**2))
    alpha = np.linalg.inv(K+lam*np.identity(n))@y
    for i in range(len(X)):
        for j in range(len(X)):
            Y_hat[i][j] = np.exp(-np.linalg.norm(X[i] - X[j])**2/
 ↪(2*sigma**2))*alpha[i][0]
        y_hat.append(Y_hat[i].sum())
    y_aux = [1 if i > 0.5 else 0 for i in y_hat ]
    accuracy.append(accuracy_score(y_aux,y))
    if first and accuracy_score(y_aux,y) == 1.0:
        no_error_sigma = sigma
        first = False
```

[18]:
```
df = pd.DataFrame({"Sigmas":sigmas,"Accuracy":accuracy})
df.plot.scatter("Sigmas","Accuracy")
```

[18]:
```
<AxesSubplot:xlabel='Sigmas', ylabel='Accuracy'>
```



[20]:
```
print("We achieve a 100% accuracy on the training set with a sigma␣
 ↪of",no_error_sigma)
```

```
We achieve a 100% accuracy on the training set with a sigma of 24
```

c)

[21]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state=42)
n = len(X_train)
```

```
distsq=np.zeros((n,n),dtype=float)

for i in range(0,n):
    for j in range(0,n):
        d = np.linalg.norm(X_train[i,:]-X_train[j,:])
        distsq[i,j]=d**2

accuracy = []
best_acc = None
best_sigma = None
Y_hat = np.zeros((n,n))
# Calculate best sigma
for sigma in sigmas:
    y_hat = []
    K = np.exp(-distsq/(2*sigma**2))
    alpha = np.linalg.inv(K+lam*np.identity(n))@Y_train
    for i in range(n):
        for j in range(n):
            Y_hat[i][j] = np.exp(-np.linalg.norm(X[i] - X[j])**2/
↪(2*sigma**2))*alpha[i][0]
        y_hat.append(Y_hat[i].sum())
    y_aux = [1 if i > 0.5 else 0 for i in y_hat ]
    current_acc = accuracy_score(y_aux,Y_train)
    accuracy.append(current_acc)
    if best_acc == None or best_acc < current_acc:
        best_acc = current_acc
        best_sigma = sigma
```

[23]:
```
n = len(X_test)
distsq=np.zeros((n,n),dtype=float)

for i in range(0,n):
    for j in range(0,n):
        d = np.linalg.norm(X_train[i,:]-X_train[j,:])
        distsq[i,j]=d**2

K = np.exp(-distsq/(2*best_sigma**2))
alpha = np.linalg.inv(K+lam*np.identity(n))@Y_test
Y_hat = np.zeros((n,n))
y_hat = []
for i in range(n):
    for j in range(n):
        Y_hat[i][j] = np.exp(-np.linalg.norm(X_test[i] - X[j])**2/
↪(2*sigma**2))*alpha[i][0]
    y_hat.append(Y_hat[i].sum())
y_aux = [1 if i > 0.5 else 0 for i in y_hat ]
accuracy_score(y_aux,Y_test)
```

```
[23]: 1.0
```

We can see from the accuracy_score that we got a perfect accuracy on the testing set