TP 6 – AOS1
Kernel methods
Corrigé

1 Faces classification

In this hands-on session we are going to classify faces with SVM. First download the dataset with the following intructions

```
from sklearn.datasets import fetch_lfw_people
faces = fetch_lfw_people(min_faces_per_person=60)
```

(1) By looking at the fetched object `faces`, tell how many samples there is, what are their dimensionality and what are the different classes.

```
In  [1]: from sklearn.datasets import fetch_lfw_people
         faces = fetch_lfw_people(min_faces_per_person=60)
         print(faces.images.shape)

Out [1]: (1348, 62, 47)

In  [2]: print(faces.target_names)

Out [2]: ['Ariel Sharon' 'Colin Powell' 'Donald Rumsfeld' 'George W Bush'
          'Gerhard Schroeder' 'Hugo Chavez' 'Junichiro Koizumi' 'Tony
          ↪  Blair']
```

Before learning, we split our dataset into a test set and a train set.

(2) Use the `train_test_split` function to split our dataset into `X_train`, `X_test`, `y_train` and `y_test`.

```
from sklearn.model_selection import train_test_split
```

```
In  [3]: from sklearn.model_selection import train_test_split
         X = faces.data
         y = faces.target
         X_train, X_test, y_train, y_test = train_test_split(X, y,
         ↪  test_size=0.25)
```

Next, to make the SVM learning more tractable we start by a reduction of dimension.

③ Use a PCA to reduce the dimension to 100. What is the percentage of explained variance?

```
In  [4]: from sklearn.decomposition import PCA
         n_components = 100
         pca = PCA(n_components=n_components, whiten=True)
         pca.fit(X_train)

Out [4]: PCA(n_components=100, whiten=True)

In  [5]: X_train_pca = pca.transform(X_train)
         X_test_pca = pca.transform(X_test)
         print(sum(pca.explained_variance_ratio_))

Out [5]: 0.9188534861896187
```

Now that the dataset is of acceptable dimension, learn a linear SVM on the train set and look at the training error and confusion matrix on the test set. You will need the following functions to do so:

```
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

```
In  [6]: from sklearn.svm import SVC
         from sklearn.metrics import confusion_matrix
         from sklearn.metrics import classification_report
         from sklearn.metrics import accuracy_score
         svc = SVC(kernel='rbf', gamma='auto')
         svc.fit(X_train_pca, y_train)

Out [6]: SVC(gamma='auto')

In  [7]: y_pred = svc.predict(X_test_pca)
         print(confusion_matrix(y_test, y_pred,
          ↪  labels=range(len(faces.target_names))))

Out [7]: [[ 10   1   0   4   0   0   0   0]
          [  0  48   0   3   0   0   0   0]
          [  0   1  17   5   1   0   0   0]
          [  0   1   0 147   0   0   0   0]
          [  0   0   1   8  15   0   0   2]
          [  0   2   0   7   1   8   0   0]
          [  0   0   0   2   0   0  12   0]
          [  0   0   0  11   1   0   0  29]]

In  [8]: print(classification_report(y_test, y_pred,
          ↪  target_names=faces.target_names))
```

Out [8]:

	precision	recall	f1-score	support
Ariel Sharon	1.00	0.67	0.80	15
Colin Powell	0.91	0.94	0.92	51
Donald Rumsfeld	0.94	0.71	0.81	24
George W Bush	0.79	0.99	0.88	148
Gerhard Schroeder	0.83	0.58	0.68	26
Hugo Chavez	1.00	0.44	0.62	18
Junichiro Koizumi	1.00	0.86	0.92	14
Tony Blair	0.94	0.71	0.81	41
accuracy			0.85	337
macro avg	0.93	0.74	0.80	337
weighted avg	0.87	0.85	0.84	337

In [9]:
```
print(accuracy_score(y_test, y_pred))
```

Out [9]:
```
0.8486646884272997
```

④ At this point, we have only used the default values for all hyperparameters to train our model. What are those hyperparameters?

There is the parameter γ in the Gaussian kernel and the parameter C that controls the regularization for the SVM model.
One could also have the kernel itself as a (qualitative) hyperparameter and the number of retained principal components in the PCA as another hyperparameter but this would make the grid search more difficult as the hyperparameters wouldn't be independent anymore.

⑤ Use the `GridSearchCV` object to perform a search on the 2 hyperparameters. What are the best hyperparameters?

```
from sklearn.model_selection import GridSearchCV
```

In [10]:
```
import numpy as np
from sklearn.model_selection import GridSearchCV

param_grid = {"C": np.logspace(-2, 3, 10), "gamma":
↪  np.logspace(-4, 1, 10)}
clf = GridSearchCV(SVC(kernel="rbf", gamma="auto"), param_grid,
↪  cv=5)
clf = clf.fit(X_train_pca, y_train)
print(clf.best_estimator_)
```

Out [10]:
```
SVC(C=5.994842503189409, gamma=0.004641588833612782)
```

```
In [11]: y_pred = clf.predict(X_test_pca)
         print(confusion_matrix(y_test, y_pred,
         ↪ labels=range(len(faces.target_names))))
```

```
Out [11]: [[ 13   0   1   1   0   0   0   0]
           [  0  47   0   1   1   0   0   2]
           [  0   1  22   1   0   0   0   0]
           [  1   5   4 137   0   0   0   1]
           [  0   1   1   2  20   0   0   2]
           [  1   1   0   2   0  10   1   3]
           [  0   0   0   1   0   0  13   0]
           [  0   0   1   6   1   0   0  33]]
```

```
In [12]: print(classification_report(y_test, y_pred,
         ↪ target_names=faces.target_names))
```

```
Out [12]:                    precision    recall  f1-score   support

             Ariel Sharon       0.87      0.87      0.87        15
             Colin Powell       0.85      0.92      0.89        51
          Donald Rumsfeld       0.76      0.92      0.83        24
            George W Bush       0.91      0.93      0.92       148
        Gerhard Schroeder       0.91      0.77      0.83        26
              Hugo Chavez       1.00      0.56      0.71        18
        Junichiro Koizumi       0.93      0.93      0.93        14
               Tony Blair       0.80      0.80      0.80        41

                 accuracy                           0.88       337
                macro avg       0.88      0.84      0.85       337
             weighted avg       0.88      0.88      0.87       337
```

```
In [13]: print(accuracy_score(y_test, y_pred))
```

```
Out [13]: 0.8753709198813057
```

(6) Suppose we want to include the number of principal components to the set of hyperparameters. Define a scikit-learn pipeline to achieve this.

In [14]:
```
from sklearn.pipeline import Pipeline

pca = PCA(whiten=True)
lin = SVC(kernel='rbf', gamma='auto')
pca_svc = Pipeline([("pca", pca), ("svc", svc)])
clf = GridSearchCV(
    estimator=pca_svc,
    cv=5,
    param_grid=dict(
        pca__n_components=[80, 90, 100, 110],
        svc__C=np.logspace(-2, 3, 2),
        svc__gamma=np.logspace(-4, 1, 2),
    ),
)
```

2 Problem

The paper by Burges and Schölkopf [1] is investigating a method the improve the accuracy and speed of SVM. First train a SVM with the same dataset (MNIST) with the kernel and the hyperparameter C they are suggesting.

Describe the technique they are using to improve the accuracy and implement it to see if it is working.

References

[1] Chris J.C. Burges and Bernhard Schölkopf. "Improving the Accuracy and Speed of Support Vector Machines". In: *Advances in Neural Information Processing Systems 9*. MIT Press, 1997, pp. 375–381.