# TP 3 – 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)
Out [1]:  Downloading LFW metadata: https://ndownloader.figshare.com/files/5976012
          Downloading LFW metadata: https://ndownloader.figshare.com/files/5976009
          Downloading LFW metadata: https://ndownloader.figshare.com/files/5976006
          Downloading LFW data (~200MB): https://ndownloader.figshare.com/files/5976015
In [2]:   print(faces.images.shape)
Out [2]:  (1348, 62, 47)
In [3]:   print(faces.target_names)
Out [3]:  ['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 [4]:   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 [5]:  from sklearn.decomposition import PCA
         n_components = 100
         pca = PCA(n_components=n_components, whiten=True)
         pca.fit(X_train)

Out [5]: PCA(copy=True, iterated_power='auto', n_components=100, random_state=None,
             svd_solver='auto', tol=0.0, whiten=True)

In [6]:  X_train_pca = pca.transform(X_train)
         X_test_pca = pca.transform(X_test)
         print(sum(pca.explained_variance_ratio_))

Out [6]: 0.9187259352183901
```

Now that the dataset is of acceptable dimension, learn a vanilla 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 [7]:  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 [7]: SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
             decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
             max_iter=-1, probability=False, random_state=None, shrinking=True,
             tol=0.001, verbose=False)

In [8]:  y_pred = svc.predict(X_test_pca)
         print(confusion_matrix(y_test, y_pred, labels=range(len(faces.target_names))))

Out [8]: [[ 10   0   0   4   1   0   0   0]
          [  0  47   1  12   0   0   0   1]
          [  0   2  18   5   0   0   0   0]
          [  0   2   0 142   0   0   0   1]
          [  0   0   1   5  21   0   0   0]
          [  0   3   0   5   0   8   0   0]
          [  0   0   1   5   0   0   7   1]
          [  0   1   0  11   0   0   0  22]]

In [9]:  print(classification_report(y_test, y_pred, target_names=faces.target_names))

Out [9]:                    precision    recall  f1-score   support

              Ariel Sharon       1.00      0.67      0.80        15
              Colin Powell       0.85      0.77      0.81        61
           Donald Rumsfeld       0.86      0.72      0.78        25
             George W Bush       0.75      0.98      0.85       145
         Gerhard Schroeder       0.95      0.78      0.86        27
               Hugo Chavez       1.00      0.50      0.67        16
         Junichiro Koizumi       1.00      0.50      0.67        14
                Tony Blair       0.88      0.65      0.75        34

                  accuracy                           0.82       337
                 macro avg       0.91      0.70      0.77       337
              weighted avg       0.84      0.82      0.81       337

In [10]: print(accuracy_score(y_test, y_pred))

Out [10]: 0.8160237388724035
```

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 $\gamma$ 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 [11]:  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, iid=False, cv=5)
          clf = clf.fit(X_train_pca, y_train)

Out [11]: /home/sylvain/.local/lib/python3.8/site-packages/sklearn/model_selection/_search.py:823:
          ↪ FutureWarning: The parameter 'iid' is deprecated in 0.22 and will be removed in 0.24.
            warnings.warn(

In [12]:  print(clf.best_estimator_)

Out [12]: SVC(C=5.994842503189409, break_ties=False, cache_size=200, class_weight=None,
              coef0=0.0, decision_function_shape='ovr', degree=3,
              gamma=0.004641588833612782, kernel='rbf', max_iter=-1, probability=False,
              random_state=None, shrinking=True, tol=0.001, verbose=False)

In [13]:  y_pred = clf.predict(X_test_pca)
          print(confusion_matrix(y_test, y_pred, labels=range(len(faces.target_names))))

Out [13]: [[ 12   0   0   2   1   0   0   0]
           [  2  50   2   6   0   0   0   1]
           [  1   1  21   1   1   0   0   0]
           [  2   4   1 137   0   0   0   1]
           [  0   0   3   2  22   0   0   0]
           [  0   3   0   1   1  10   0   1]
           [  0   1   1   1   0   0  10   1]
           [  0   0   0   6   1   0   0  27]]

In [14]:  print(classification_report(y_test, y_pred, target_names=faces.target_names))

Out [14]:                    precision    recall  f1-score   support

               Ariel Sharon       0.71      0.80      0.75        15
               Colin Powell       0.85      0.82      0.83        61
            Donald Rumsfeld       0.75      0.84      0.79        25
              George W Bush       0.88      0.94      0.91       145
          Gerhard Schroeder       0.85      0.81      0.83        27
                Hugo Chavez       1.00      0.62      0.77        16
          Junichiro Koizumi       1.00      0.71      0.83        14
                 Tony Blair       0.87      0.79      0.83        34

                   accuracy                           0.86       337
                  macro avg       0.86      0.79      0.82       337
               weighted avg       0.86      0.86      0.86       337

In [15]:  print(accuracy_score(y_test, y_pred))

Out [15]: 0.857566765578635
```

(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 [16]:  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,
              iid=False,
              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.