Scientific Computing - Practice sessions : Audio command recognition by DTW and classification

Group name:

Names :

Surnames :

The Practice sessions will permit testing the dynamic programming algorithm (DTW) seen in Exercise session (TD) and then implement an audio recognition system for isolated words (constituting orders for drones).

These sessions are divided into 3 parts:

- Part I: DTW and application of the TD
- Part II: Audio control word recognition system
- Part III: Comparison of dynamic programming with a classification method after data pre-processing by PCA

For **parts II and III**, you will test the audio recognition system on two sets of voices that will serve as a learning base (references) and a test base (sounds to be recognized) respectively. The list of the 13 drone commands are: *Landing, Takeoff, Takeoff, Advance, Right turn, Backward, Left turn, Right, Flip, Left, Stop, Higher, Lower and State of Emergency.*

To do this, you must per group of 2 students (number of students **MANDATORY**):

1. **Propose a study** that you will detail on a report. For example, *influence male voices VS female voices, compare your own voices to the database, test the impact of different background noises on recognition...*];
2. Create, according to the objective of your study, your own learning base and test base from the proposed corpus and the voices and sounds you have recorded [*audio parameters: 16 KHz, mono, 16 bits, .wav format*];
3. Test the DTW and a classification method with pre-processing by PCA;
4. Evaluate the results;
5. Write a pdf report presenting the study, the results by the 2 methods and your comments and conclusions on your study (Max. length: 8 pages).

Entrée [ ]:

```
import matplotlib.pyplot as plt
from numpy import array, zeros, full, argmin, inf, ndim
import scipy
import sklearn
import math
```

Part I: Implementation of the dynamic programming algorithm

1. Write a function in DTW python that implements the calculation and display of the cost matrix defined in TD.
2. In order to easily adapt the cost calculation according to the nature of the data (and therefore the distances used), write a function for each distance (Euclidean, letters, sounds) that will appear as a parameter of the DTW function.

Entrée [ ]:

```
```

Application to exercises

1. Test your programs on the exercises seen in TD.
2. Modify the local constraints, i.e. the weights according to the directions.
3. Add the consideration of global constraints, i. e. non-calculation when the boxes are too far from the diagonal (see exercise TD DNA sequence). From which position do global constraints not change the results?

Entrée [ ]:

```
```

Part II: Audio control word recognition system

On the shared space, you will find audio recordings of command words for a quadricopter drone composed of several male french speakers (noted M01...M13) and female french speakers (F01...F05).

You can thus divide all the data into learning bases that will serve as references and test bases to evaluate recognition by dynamic programming.

Entrée [ ]:

```
import librosa
```

The following lines of code allow you to transform the audio file into a matrix of parameters called MFCC (Mel Frequency Cepstral Coefficient) using the *librosa* python library. These settings are used to extract the best possible frequency voice content from the audio signal.

The output matrix is composed of as many column vectors as analysis frames. The number of lines corresponds to the size of the representative vector: here 12.

Audio file upload:

Entrée [ ]:

```
y, sr = librosa.load(audio_file, offset=30, duration=5)
```

MFCC extraction

Entrée [ ]:

```
mfcc = librosa.feature.mfcc(y=y, sr=sr, hop_length=1024, htk=True, n_mfcc=12)
```

Application of DTW

1. Carry out a study that you will detail on a report (for example, *influence male voices VS female voices, compare your own voice with the database, test the impact of different background noises on recognition...*) and create your own learning database and test database from the corpus and the voices and noises you have recorded.
2. Apply DTW to your corpora.

Settings for audio recordings of your personal voices:

16 KHz, mono, 16 bits, *.wav* format

Entrée [ ]:

```

```

Assessment of recognition

1. Calculate the system confusion matrix (in line with the references and in column the system outputs). You can use the *confusion_matrix* function of the *sklearn* library.

2. Calculate the recognition score: number of well recognized files on number of tested files.

Verification:

- if you use the M01 reference and test file, you must get no errors.
- if you use as M01 reference file and M02 test file, you must get two errors.

Entrée [ ]:

```

```

Part III: Comparison of dynamic programming with a classification method after data pre-processing

In this section, we will compare the results of DTW with those of a data classification method: the k-nearest neighbors (k-nn).

We will use the functions to calculate the PCA and k-nn via the python library *scikit-learn*.

Entrée [ ]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from mpl_toolkits.mplot3d import Axes3D
```

PCA preprocessing

To test a classification method, the size of the MFCCs must first be reduced:

1. From all the records in the learning database, perform a Principal Component Analysis (PCA) using the *PCA* function of the *scikit-learn* library and then project the test data into this new database.

Note: You can also implement the PCA by extracting the 3 eigenvectors, noted $X_1 X_1$, $X_2 X_2$, $X_3 X_3$, associated with the 3 largest eigenvalues of the variance-covariance $\Sigma_{App} \Sigma_{App}$ (by the functions *np.cov* and *np.linalg.eig*). These eigenvectors will constitute the new benchmark P. Then project the data from the learning and test database into this new database by multiplying each vector by the database $P = [X_1 X_2 X_3] P = [X_1 X_2 X_3]$.

Entrée [ ]:

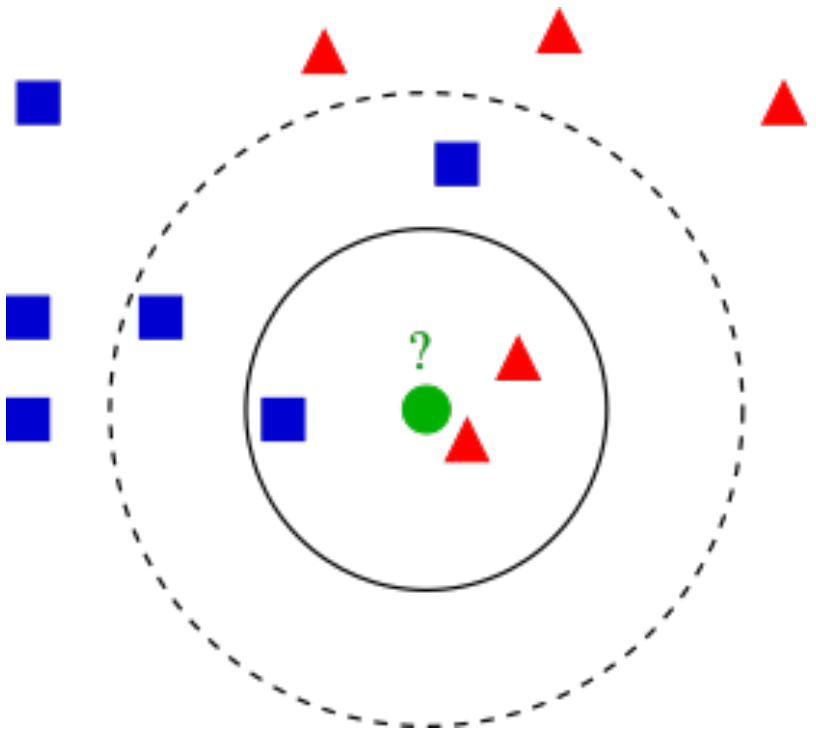
```

```

Classification by k nearest neighbors

In artificial intelligence, the k nearest neighbor method (*k-nn*) is a supervised method. In this context, there is a learning database of "label-data" pairs. To estimate the output associated with a new input xx, the kk nearest neighbor method consists of taking into account (in the same way) the kk learning samples whose input is closest to the new input xx, according to a distance to be defined. The associated algorithm and an example are given below.

k-Nearest Neighbor
Classify (**X**, **Y**, *x*) // **X**: training data, **Y**: class labels of **X**, *x*: unknown sample
for $i = 1$ **to** m **do**
  Compute distance $d(\mathbf{X}_i, x)$
end for
Compute set I containing indices for the k smallest distances $d(\mathbf{X}_i, x)$.
return majority label for $\{\mathbf{Y}_i$ where $i \in I\}$



Example of classification by k-nn. The test sample (green circle) must be classified either in the first class of blue squares, or in the second class of red triangles. If k = 3 (full circle), it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dotted circle), it is assigned to the first class (3 squares against 2 triangles inside the outer circle)

1. Using the *KNeighborsClassifier* function of the *sklearn.neighbors library*, perform a classification by k-nn on the learning and test basis you have predefined (take $k = 1 k = 1$).
2. Evaluate the k-nn method by calculating the confusion matrix and the recognition rate.
3. Change the value of $k k$ for k-nn. Do you improve recognition scores?
4. Compare your results with those of DTW.

Entrée [ ]:

Scientific Computing - Practice sessions : Audio command recognition by DTW and classification

Group name:

Names :

Surnames :

The Practice sessions will permit testing the dynamic programming algorithm (DTW) seen in Exercise session (TD) and then implement an audio recognition system for isolated words (constituting orders for drones).

These sessions are divided into 3 parts:

- Part I: DTW and application of the TD
- Part II: Audio control word recognition system
- Part III: Comparison of dynamic programming with a classification method after data pre-processing by PCA

For **parts II and III**, you will test the audio recognition system on two sets of voices that will serve as a learning base (references) and a test base (sounds to be recognized) respectively. The list of the 13 drone commands are: *Landing, Takeoff, Takeoff, Advance, Right turn, Backward, Left turn, Right, Flip, Left, Stop, Higher, Lower and State of Emergency.*

To do this, you must per group of 2 students (number of students **MANDATORY**):

1. **Propose a study** that you will detail on a report. For example, *influence male voices VS female voices, compare your own voices to the database, test the impact of different background noises on recognition...*];
2. Create, according to the objective of your study, your own learning base and test base from the proposed corpus and the voices and sounds you have recorded [*audio parameters: 16 KHz, mono, 16 bits, .wav format*];
3. Test the DTW and a classification method with pre-processing by PCA;
4. Evaluate the results;
5. Write a pdf report presenting the study, the results by the 2 methods and your comments and conclusions on your study (Max. length: 8 pages).

Entrée [ ]:

Part I: Implementation of the dynamic programming algorithm

1. Write a function in DTW python that implements the calculation and display of the cost matrix defined in TD.
2. In order to easily adapt the cost calculation according to the nature of the data (and therefore the distances used), write a function for each distance (Euclidean, letters, sounds) that will appear as a parameter of the DTW function.

Entrée [ ]:

Application to exercises

1. Test your programs on the exercises seen in TD.
2. Modify the local constraints, i.e. the weights according to the directions.
3. Add the consideration of global constraints, i. e. non-calculation when the boxes are too far from the diagonal (see exercise TD DNA sequence). From which position do global constraints not change the results?

Entrée [ ]:

Part II: Audio control word recognition system

On the shared space, you will find audio recordings of command words for a quadricopter drone composed of several male french speakers (noted M01...M13) and female french speakers (F01...F05).

You can thus divide all the data into learning bases that will serve as references and test bases to evaluate recognition by dynamic programming.

Entrée [ ]:

The following lines of code allow you to transform the audio file into a matrix of parameters called MFCC (Mel Frequency Cepstral Coefficient) using the *librosa* python library. These settings are used to extract the best possible frequency voice content from the audio signal.

The output matrix is composed of as many column vectors as analysis frames. The number of lines corresponds to the size of the representative vector: here 12.

Audio file upload:

Entrée [ ]:

MFCC extraction

Entrée [ ]:

Application of DTW

1. Carry out a study that you will detail on a report (for example, *influence male voices VS female voices, compare your own voice with the database, test the impact of different background noises on recognition...*) and create your own learning database and test database from the corpus and the voices and noises you have recorded.
2. Apply DTW to your corpora.

Settings for audio recordings of your personal voices:

16 KHz, mono, 16 bits, *.wav* format

Entrée [ ]:

Assessment of recognition

1. Calculate the system confusion matrix (in line with the references and in column the system outputs). You can use the *confusion_matrix* function of the *sklearn* library.

2. Calculate the recognition score: number of well recognized files on number of tested files.

Verification:

- if you use the M01 reference and test file, you must get no errors.
- if you use as M01 reference file and M02 test file, you must get two errors.

Entrée [ ]:

Part III: Comparison of dynamic programming with a classification method after data pre-processing

In this section, we will compare the results of DTW with those of a data classification method: the k-nearest neighbors (k-nn).

We will use the functions to calculate the PCA and k-nn via the python library *scikit-learn*.

Entrée [ ]:

PCA preprocessing

To test a classification method, the size of the MFCCs must first be reduced:

1. From all the records in the learning database, perform a Principal Component Analysis (PCA) using the *PCA* function of the *scikit-learn* library and then project the test data into this new database.

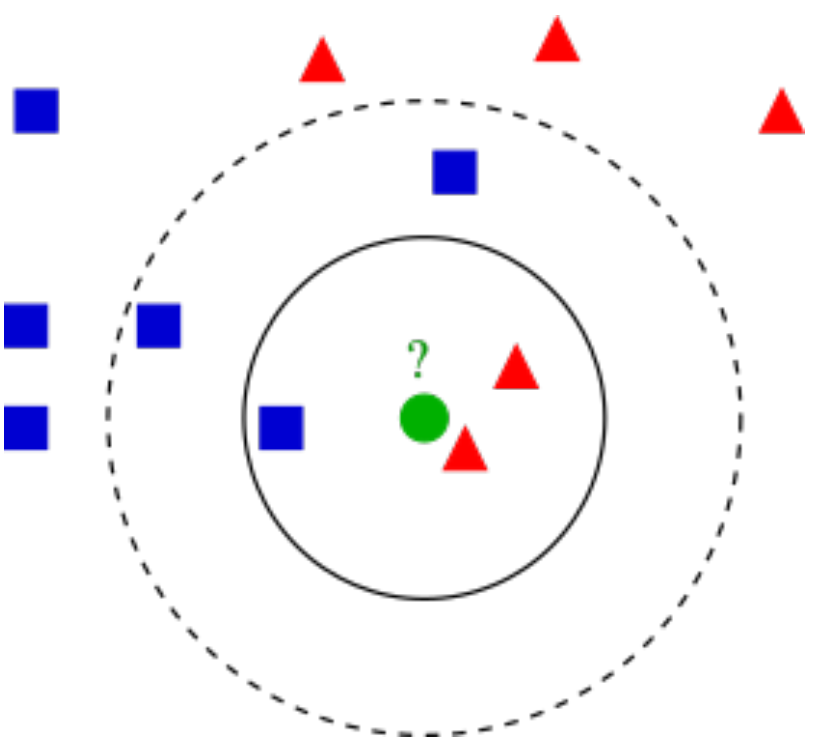
Note: You can also implement the PCA by extracting the 3 eigenvectors, noted X_1, X_2, X_3, associated with the 3 largest eigenvalues of the variance-covariance Σ_{App} (by the functions *np.cov* and *np.linalg.eig*). These eigenvectors will constitute the new benchmark P. Then project the data from the learning and test database into this new database by multiplying each vector by the database $P = [X_1 X_2 X_3]$.

Entrée [ ]:

Classification by k nearest neighbors

In artificial intelligence, the k nearest neighbor method (*k-nn*) is a supervised method. In this context, there is a learning database of "label-data" pairs. To estimate the output associated with a new input x, the k nearest neighbor method consists of taking into account (in the same way) the k learning samples whose input is closest to the new input x, according to a distance to be defined. The associated algorithm and an example are given below.

k-Nearest Neighbor
Classify $(\mathbf{X}, \mathbf{Y}, x)$ // $\mathbf{X}$: training data, $\mathbf{Y}$: class labels of $\mathbf{X}$, x: unknown sample
for $i = 1$ **to** m **do**
  Compute distance $d(\mathbf{X}_i, x)$
end for
Compute set I containing indices for the k smallest distances $d(\mathbf{X}_i, x)$.
return majority label for $\{\mathbf{Y}_i$ where $i \in I\}$



Example of classification by k-nn. The test sample (green circle) must be classified either in the first class of blue squares, or in the second class of red triangles. If k = 3 (full circle), it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dotted circle), it is assigned to the first class (3 squares against 2 triangles inside the outer circle)

1. Using the *KNeighborsClassifier* function of the *sklearn.neighbors library*, perform a classification by k-nn on the learning and test basis you have predefined (take $k = 1$).
2. Evaluate the k-nn method by calculating the confusion matrix and the recognition rate.
3. Change the value of k for k-nn. Do you improve recognition scores?
4. Compare your results with those of DTW.

Entrée [ ]: