# Canonical Correlation Analysis

In this exercise, we consider canonical correlation analysis (CCA) on two simple problems, one in low dimensions and one in high dimensions. The goal is to implement the primal and dual versions of CCA to handle these two different cases. The first dataset consists of two trajectories in two dimensions. The dataset is extracted and plotted below. The first data points are shown in dark blue, and the last ones are shown in yellow.

In [1]:

```python
import numpy
import matplotlib
%matplotlib inline
from matplotlib import pyplot as plt
import utils

X,Y   = utils.getdata()
p1,p2 = utils.plotdata(X,Y)
```
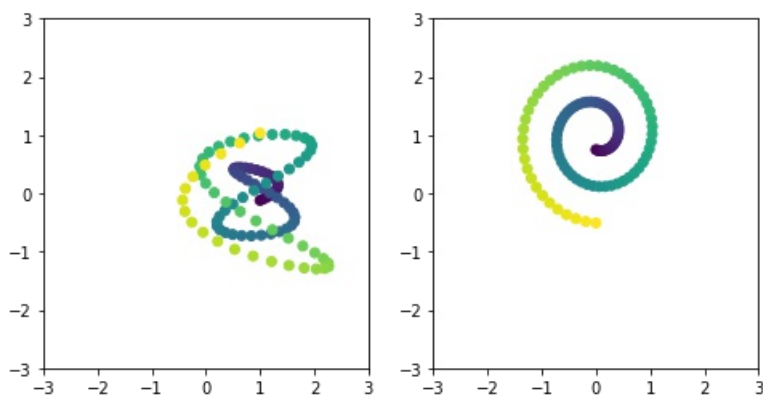

For these two trajetories, that can be understood as two different modalities of the same data, we would like determine under which projections they appear maximally correlated.

Implementing Primal CCA

As stated in the lecture, the CCA problem in its primal form consists of maximizing the cross-correlation objective:

$$J(w_x, w_y) = w_x^\top C_{xy} w_y$$

subject to autcorrelation constraints $w_x^\top C_{xx} w_x = 1$ and $w_y^\top C_{yy} w_y = 1$. Using the method of Lagrange multipliers, this optimization problem can be reduced to finding the first eigenvector of the generalized eigenvalue problem:

$$\begin{bmatrix} 0 & C_{xy} \\ C_{yx} & 0 \end{bmatrix} \begin{bmatrix} w_x \\ w_y \end{bmatrix} = \lambda \begin{bmatrix} C_{xx} & 0 \\ 0 & C_{yy} \end{bmatrix} \begin{bmatrix} w_x \\ w_y \end{bmatrix}$$

Your first task is to write a function that solves the CCA problem in the primal (i.e. that solves the generalized eigenvalue problem above). The function you need to implement receives two matrices `X` and `Y` of size `N` × `d1` and `N` × `d2` respectively. It returns two vectors of size `d1` and `d2` corresponding to the projections associated to the modalities `X` and `Y`. *(Hint: Note that the data matrices `X` and `Y` have not been centered yet.)*

In [2]:

```python
X.shape, Y.shape
```

Out[2]:

```
((100, 2), (100, 2))
```

In [3]:

```python
import numpy as np
```

```python
from scipy.linalg import eig

def CCAprimal(X,Y):
    N = X.shape[0]

    #center the data
    X = X - np.mean(X, axis=0)
    Y = Y - np.mean(Y, axis=0)

    #compute auto and cross covariances
    C_xx, C_yy = [(1/N)*data.T.dot(data) for data in [X,Y]]
    C_xy = (1/N)*X.T.dot(Y)
    C_yx = (1/N)*Y.T.dot(X)

    #size of wx and wy
    d1, d2 = C_xy.shape

    #assemble the formular
    cross_mat, auto_mat = [np.zeros((d1+d2,)*2) for _ in range(2)]
    cross_mat[:d1,d1:] = C_xy
    cross_mat[d1:,:d1] = C_yx
    auto_mat[:d1,:d1]  = C_xx
    auto_mat[d1:,d1:]  = C_yy

    #compute eigenvalues and eigenvectors
    eig_values, eig_vectors = eig(a=cross_mat,b=auto_mat)

    #eigenvectors are sorted in ascending order, therefore pick last one as solution
    wx_wy = eig_vectors[:,-1]
    wx,wy = wx_wy[:d1], wx_wy[d1:]

    return wx/np.linalg.norm(wx),wy/np.linalg.norm(wy)
```

The function can now be called with our dataset. The learned projection vectors w_x and w_y are plotted as red arrows.
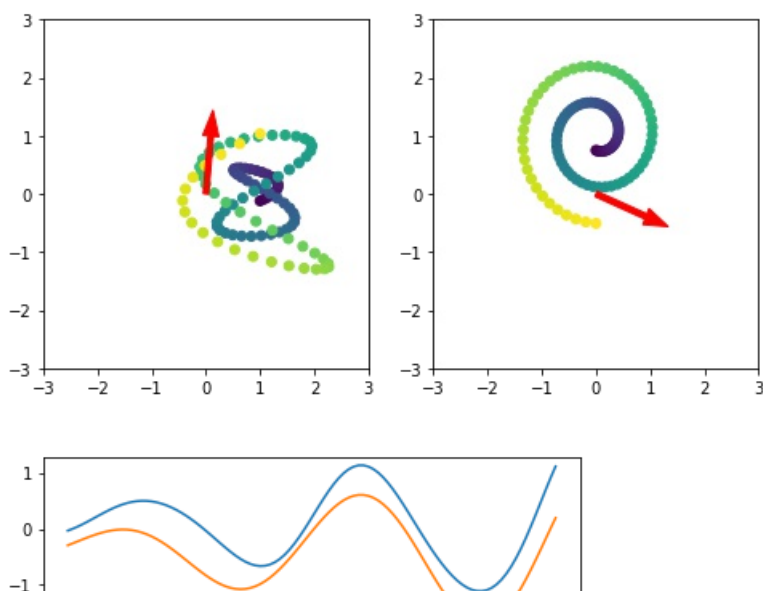
In [4]:
```python
wx,wy = CCAprimal(X,Y)
print(wx,wy)

p1,p2 = utils.plotdata(X,Y)
p1.arrow(0,0,1*wx[0],1*wx[1],color='red',width=0.1)
p2.arrow(0,0,1*wy[0],1*wy[1],color='red',width=0.1)
plt.show()

plt.figure(figsize=(6,2))
plt.plot(numpy.dot(X,wx))
plt.plot(numpy.dot(Y,wy))
plt.show()
```
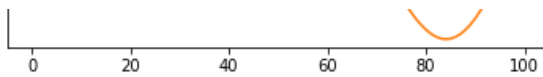
[0.09026166 0.99591809] [0.92124018 -0.38899427]

```
#checking my own solution above with scikit learn
from sklearn.cross_decomposition import CCA
cca = CCA(n_components=1).fit(X,Y)

wx,wy = np.squeeze(cca.x_weights_), np.squeeze(cca.y_weights_)
print(wx,wy)

p1,p2 = utils.plotdata(X,Y)
p1.arrow(0,0,1*wx[0],1*wx[1],color='red',width=0.1)
p2.arrow(0,0,1*wy[0],1*wy[1],color='red',width=0.1)
plt.show()

plt.figure(figsize=(6,2))
plt.plot(numpy.dot(X,wx))
plt.plot(numpy.dot(Y,wy))
plt.show()
```

[0.09404939 0.99556753] [0.91704916 -0.39877416]
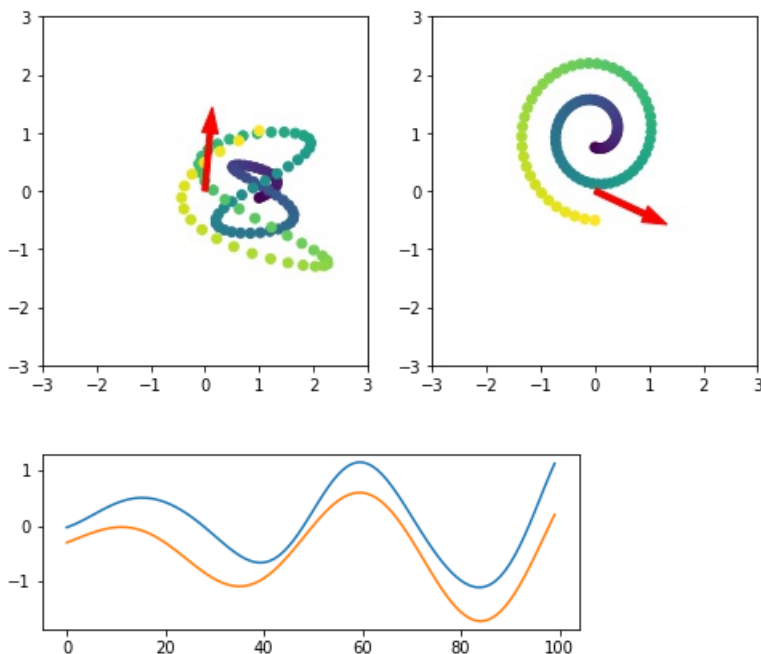


Clearly, the data is correlated in the projected space.

Implementing Dual CCA

In the second part of the exercise, we consider the case where the data is high dimensional (with `d` » `N`). Such high-dimensionality occurs for example, when input data are images. We consider the scenario where sources emit spatially, and two (noisy) receivers measure the spatial field at different locations. We would like to identify signal that is common to the two measured locations, e.g. a given source emitting at a given frequency. We first load the data and show one example.

```
X,Y = utils.getHDdata()

utils.plotHDdata(X[0],Y[0])
plt.show()
```
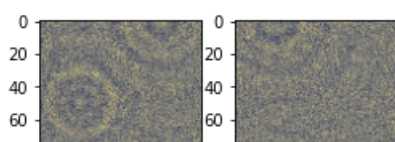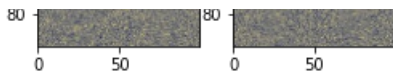
Several sources can be perceived, however, there is a significant level of noise. Here again, we will use CCA to find subspaces where the two modalities are maximally correlated. In this example, because there are many more dimensions than there are data points, it is more advantageous to solve CCA in the dual. Your task is to implement a CCA dual solver that receives two data matrices of size `N × d1` and `N × d2` respectively as input, and returns the associate CCA directions (two vectors of respective sizes `d1` and `d2`).

In [7]:

```python
def CCAdual(X,Y):
    N = X.shape[0]
    reg = 0.05

    X = X - np.mean(X, axis=0)
    Y = Y - np.mean(Y, axis=0)

    #compute kernels
    K_x = X.dot(X.T)
    K_y = Y.dot(Y.T)
    K_x_reg = K_x + reg*np.identity(N)
    K_y_reg = K_y + reg*np.identity(N)

    #compute block matrix components
    K_xx = K_x_reg.dot(K_x_reg)
    K_yy = K_y_reg.dot(K_y_reg)
    K_xy = K_x.dot(K_y)
    K_yx = K_y.dot(K_x)

    #get dims of X and Y
    d1, d2 = K_xy.shape

    #assemble block matrix to solve as eigenvalue problem
    cross_mat, auto_mat = [np.zeros((d1+d2,)*2) for _ in range(2)]
    cross_mat[:d1,d1:] = K_xy
    cross_mat[d1:,:d1] = K_yx
    auto_mat[:d1,:d1]  = K_xx
    auto_mat[d1:,d1:]  = K_yy

    #compute eigenvectors
    eigen_values, eig_vectors = eig(a=cross_mat,b=auto_mat)

    #pick eigenvector corresponding to largest eigenvalue
    idx = np.argmax(eigen_values)
    alpha = eig_vectors[:,idx]

    #get solutions in kernel space
    alpha_x,alpha_y = alpha[:d1], alpha[d1:]

    #compute solutions in input space
    wx, wy = X.T.dot(alpha_x), Y.T.dot(alpha_y)

    return wx, wy
```
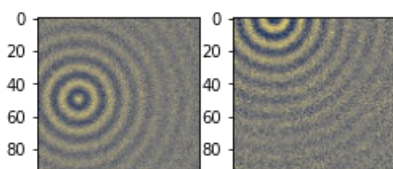
We now call the function we have implemented with a training sequence of 100 pairs of images. Because the returned solution is of same dimensions as the inputs, it can be rendered in a similar fashion.

In [8]:

```python
wx,wy = CCAdual(X[:100],Y[:100])

utils.plotHDdata(wx,wy)
plt.show()
```
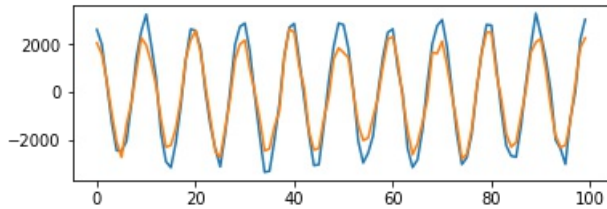
Here, we can clearly see a common factor that has been extracted between the two fields, specifically a point source emitting at a particular frequency. A test sequence of 100 pairs of images can now be projected on these two filters:

In [9]:

```
plt.figure(figsize=(6,2))
plt.plot(numpy.dot(X[100:],wx))
plt.plot(numpy.dot(Y[100:],wy))
plt.show()
```


Clearly the two projected signals are correlated and the input noise has been strongly reduced.

1a)
$$\begin{bmatrix} 0 & C_{xy} \\ C_{yx} & 0 \end{bmatrix}\begin{bmatrix} w_x \\ w_y \end{bmatrix} = \lambda \begin{bmatrix} C_{xx} & 0 \\ 0 & C_{yy} \end{bmatrix}\begin{bmatrix} w_x \\ w_y \end{bmatrix}$$

$\Leftrightarrow$ (1) $C_{xy}\, w_y = \lambda\, C_{xx}\, w_x$

(2) $C_{yx}\, w_x = \lambda\, C_{yy}\, w_y$

(1) $\Leftrightarrow$ $w_x^T C_{xy} w_y = \lambda \underbrace{w_x^T C_{xx} w_x}_{=1}$

Objective: $\max\left(w_x^T C_{xy} w_y\right)$ => pick solution (w_x, w_y) that corresponds to largest λ

analogously:

(2) $\Leftrightarrow$ $C_{yx}\, w_x = \lambda\, C_{yy}\, w_y$

$w_y^T C_{yx} w_x = \lambda \underbrace{w_y^T C_{yy} w_y}_{=1}$

$w_y^T C_{yx} w_x = w_x^T C_{xy} w_y = \lambda$

b) (1) $(-w_x)^T C_{xy}(-w_y) = (-1)\, w_x^T C_{xy}\, w_y \cdot (-1) = w_x^T C_{xy}\, w_y$

(2) $-w_x^T C_{xx}(-w_x) = w_x^T C_{xx}\, w_x$

analogously to (2) for second constraint $w_y^T C_{yy} w_y = 1$

As objective and restrictions have same results for (w_x, w_y) and $(-w_x, -w_y)$, $(-w_x, -w_y)$ is always as good a solution if w_x, w_y is a solution.

(2a)

$$\text{(1)} \quad C_{xy}\, w_y = \lambda\, C_{xx}\, w_x$$

$$\text{(2)} \quad C_{yx}\, w_x = \lambda\, C_{yy}\, w_y$$

with $C_{xx} = \frac{1}{N} XX^T \quad C_{yy} = \frac{1}{N} YY^T \quad C_{xy} = \frac{1}{N} XY^T \quad C_{yx} = \frac{1}{N} YX^T$

$$w_x = X\alpha_x \qquad w_y = Y\alpha_y$$

and $X \in \mathbb{R}^{d_1 \times N} \quad Y \in \mathbb{R}^{d_2 \times N} \quad \alpha_x, \alpha_y \in \mathbb{R}^N$

$$\text{(1)} \iff \frac{1}{N} XY^T Y \alpha_y = \lambda \frac{1}{N} XX^T X \alpha_x$$

$$\text{(3)} \quad K_y \alpha_y = \lambda K_x \alpha_x \qquad \text{with } K_x = X^T X \in \mathbb{R}^{N \times N}$$
$$\text{and } K_y = Y^T Y \in \mathbb{R}^{N \times N}$$

$$\text{(2)} \iff \frac{1}{N} YX^T X \alpha_x = \lambda \frac{1}{N} YY^T Y \alpha_y$$

$$\iff \quad K_x \alpha_x = \lambda K_y \alpha_y$$

$$\iff \text{(4)} \quad \alpha_y = \frac{K_y^{-1} K_x}{\lambda} \alpha_x$$

(4) in (3)

$$\frac{K_y K_y^{-1} K_x \alpha_x}{\lambda} = \lambda K_x \alpha_x$$

$$\iff \quad I \alpha_x = \lambda^2 \alpha_x \quad \Rightarrow \quad \forall \alpha_x \ \lambda = 1$$

You can pick any α_x and compute the corresponding α_y

from α_x : $K_y^{-1} K_x \alpha_x = \alpha_y$!

(2b) (1) on worksheet with $w_x = X\alpha_x$ $w_y Y\alpha_y$

Find $\alpha_x \in \mathbb{R}^M$ $\alpha_y \in \mathbb{R}^N$ maximizing $\alpha_x^T X^T X Y^T Y \alpha_y = \alpha_x^T A \cdot B \alpha_y$

subject to $\alpha_x^T X^T X X^T X \alpha_x = \alpha_x^T A^2 \alpha_x$

$\alpha_y^T Y^T Y Y^T Y \alpha_y = \alpha_y^T B^2 \alpha_y$

$$\mathcal{L} = \alpha_x^T A \cdot B \alpha_y - h_1\left(\alpha_x^T A^2 \alpha_x\right) - h_2\left(\alpha_y^T B^2 \alpha_y\right)$$

(2) $\dfrac{d\mathcal{L}}{d\alpha_x} = A B \alpha_y - h_1 A^2 \alpha_x = 0$

(3) $\dfrac{d\mathcal{L}}{d\alpha_y} = \alpha_x^T A B - h_2 B^2 \alpha_y = B A \alpha_x - h_2 B^2 \alpha_y = 0$ $\left|\begin{array}{l}\text{since}\\ A^T = A\\ B^T = B\end{array}\right.$

$\alpha_y^T(3) - \alpha_x^T(2)$: $\alpha_y^T B A \alpha_x - h_2 \underbrace{\alpha_y^T B^2 \alpha_y}_{=1} - \alpha_x^T A B \alpha_y - h_1 \underbrace{\alpha_x^T A^2 \alpha_x}_{=1} = 0$

(4) $-h_2 - h_1 = 0$ $\Rightarrow$ $h_2 = h_1 = h = \rho$

(4) in (2): $A B \alpha_y = \rho A^2 \alpha_x$

(4) in (3): $B A \alpha_x = \rho B^2 \alpha_y$ $= \begin{bmatrix} 0 & AB \\ BA & 0 \end{bmatrix}\begin{bmatrix} \alpha_x \\ \alpha_y \end{bmatrix} = \rho \begin{bmatrix} A^2 & 0 \\ 0 & B^2 \end{bmatrix}\begin{bmatrix} \alpha_x \\ \alpha_y \end{bmatrix}$

c) $A \cdot B \alpha_y = \rho A^2 \alpha_x$

$X^T X Y^T Y \alpha_y = \rho X^T X X^T X \alpha_x$

$\alpha_x^T X^T X Y^T Y \alpha_y = \rho \alpha_x^T X^T X X^T X \alpha_x$

$w_x^T C_{xy} w_y = \rho \underbrace{w_x^T C_{xx} w_x}_{=1}$

Since we want to find $\max(w_x^T C_{xy} w_y)$ we need to find the solution with largest ρ.

d) $w_x = X\alpha_x$ $w_y = Y\alpha_y$

③

For $X \in \mathbb{R}^{d_1 \times N}$ and $Y \in \mathbb{R}^{d_2 \times N}$ with $d_2 = 1$

from restraint $\boxed{\dfrac{w_y^T Y Y^T w_y}{N}} = 1$ it follows

$(\Rightarrow) \quad \dfrac{w_y^2 \|Y\|_2^2}{N} = 1 \quad (\Rightarrow) \quad w_y = \dfrac{\sqrt{N}}{\|y\|} > 0$

Simplyfy   (1) from worksheet

Find $w_x \in \mathbb{R}^{d_1}$ maximizing $w_x^T C_{xy} = w_x^T X Y^T$

$\qquad$ subject to $w_x^T C_{xx} w_x = 1 = w_x^T X X^T w_x$

$L = w_x^T X Y^T - \lambda w_x^T X X^T w_x$

$\dfrac{\partial L}{\partial w_x} = X Y^T - \lambda X X^T w_x = 0 \quad (\Rightarrow) \quad X Y^T = \lambda X X^T w_x$

$(\Rightarrow) \quad Y^T = \lambda X^T w_x \quad (\Rightarrow) \dfrac{1}{\lambda} Y^T = X^T w_x \quad$ (a)

The LSR Objective can be rewritten :

$\|X^T v - y\|^2 = v^T X X^T v - 2 v^T X y^T + \underbrace{\|y\|^2}_{\text{const}}$

$\min_v \overbrace{(v^T X X^T v - 2 v^T X y)}^{D}$

$\dfrac{\partial D}{\partial v} = X X^T v - 2 X y^T \overset{!}{=} 0 \quad (\Rightarrow) \quad X^T v = 2 y^T \quad$ (b)

so with $\lambda = \frac{1}{2}$ $w_x = v$