Computing Eigenpairs

DSTA

1 From paper to Python

1.1 Background material

This unit is less about computing than about learning/revising how eigehpairs are computed in Mathematics.

Goal: give a feeling of how these almost-magical entities that are eigenpairs do come to help us understand *latent trends in data.*

Please cover this lab in class (if time allows) or at home with pen and paper, in the rest of the class e-pairs will always be given by the `numpy.linalg` submodule.

1.2 Exercise

From the MMDS, Ch. 11, pp. 385-387.

Let, M be a square matrix.

Let λ be a constant and $\mathbf{e}$ be a non-zero column vector with the same number of rows as M.

Then λ is an e-value of M and $\mathbf{e}$ is its corresponding e-vector if

$$M\mathbf{e} = \lambda\mathbf{e}.$$

This can be reformulated as $(M - \lambda I)e = 0$.

1.3 A worked-out ex.

Let

$M = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}$

. . .

Then

$M - \lambda I = \begin{bmatrix} 3-\lambda & 2 \\ 2 & 6-\lambda \end{bmatrix}$

1.4 The determinant

The determinant of $M - \lambda I$ is $(3-\lambda)(6-\lambda) - 4$.

What are the values of λ that make the determinant=0?

$|M - \lambda I| = 0$ has two roots: $\lambda_1 = 7$, and $\lambda_2 = 2$.

Now we can discover the associated eigenvetors.

Recall that $M\mathbf{e} = \lambda\mathbf{e}$

At the moment, $\mathbf{e}$ is unknown: $[x, y]^T$.

Let's substitute $\lambda = 7$ in the eq.

$\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = 7 \cdot \begin{bmatrix} x \\ y \end{bmatrix}$

---

by multiplying matrix and vector we get two equations:

$3x + 2y = 7x$

$2x + 6y = 7y$

Both of the equations really say that $y = 2x$.

Infinte solutions are possible, let's fix $\mathbf{e_1} =$

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

But $\mathbf{e_1}$ = vector, since its norm, i.e., the sum of the squares of its components, is 5, not 1.

Thus to get the unit vector in the same direction, we divide each components by $\sqrt{5}$.

So the principal (7 is the max λ value) e-vector is

$$\begin{bmatrix} 1/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix}$$

1.5 Can we solve it for the second eigenpair?

```
import numpy as np

from numpy import linalg as LA

M = np.array([[3,2],[2,6]])

print(M)

w, v = LA.eig(M)

print(w)
print(v)
```

---

2 Reference material: The PCA technique

2.1 Objective

What if we want to have less dimensions while still retaining the differences between data points?

In the Iris example:

-some 2D data plot collapsed datapoints very close to each other (hard to classify them)

-other data plots showed scattered ponts: easier to classify data

Spectral analysis leads to Principal Component Analysis

2.2 A new reference system

From MMDS, Ch. 11, pp. 391-396.

Let M represents the dataset.

If we find e-vectors of MM^T or M^TM, then the matrix made up of these e-vectors will represent a rigid rotation in the high dimensional space:

unlike the Mona lisa example, distances/shapes are unchanged.

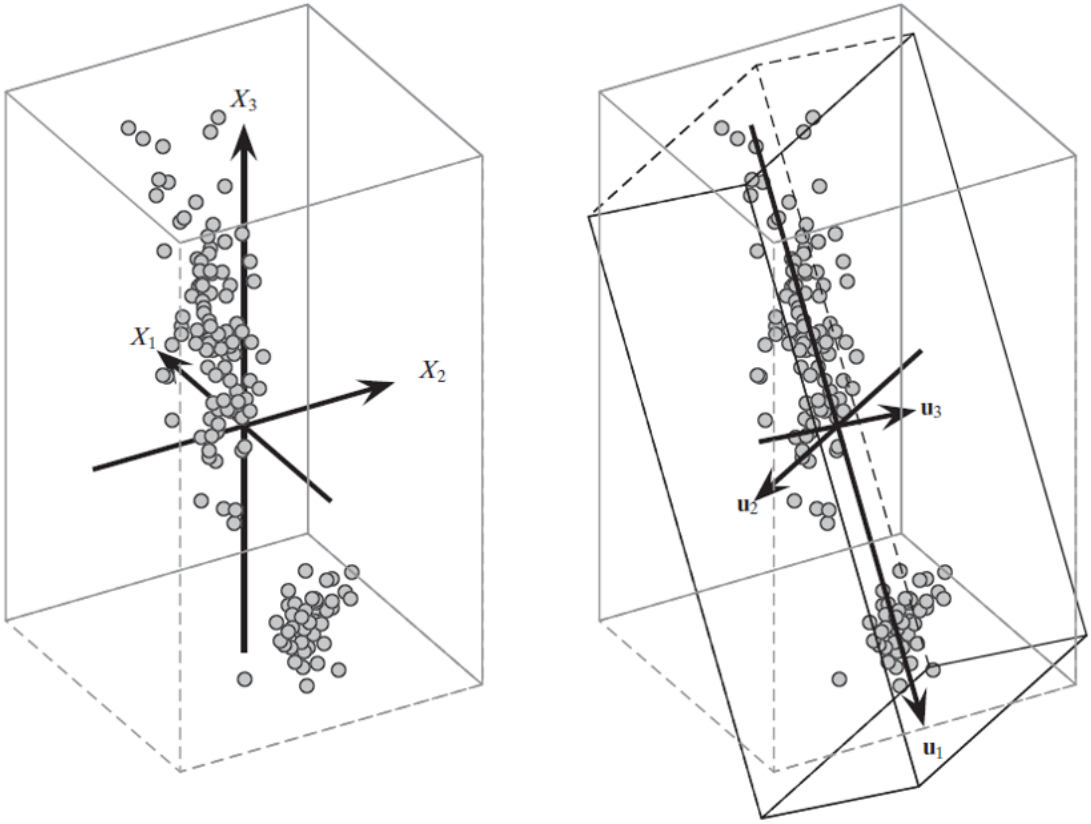
Result: the axis corresponding to the principal e-vector is the one along which the points *are most spread out.*

Try the following program and observe the phenomenon.

```
M=np.array([[1,2], [2,1], [3,4], [4,3]])

MtM = np.matmul(M.T, M)

w,v = LA.eig(MtM)

print(w)
print(v)

ME = np.matmul(M, v)

print(ME)
```

2.3 Iris: rotation

The principal e-pair (see above) gives the direction of most spread



(a) Original Basis (b) Optimal Basis

Figure 7.1. Iris data: optimal basis in three dimensions.

2.4 Iris: compression

The principal e-pair (see above) gives the best 3D-to-1D approximation

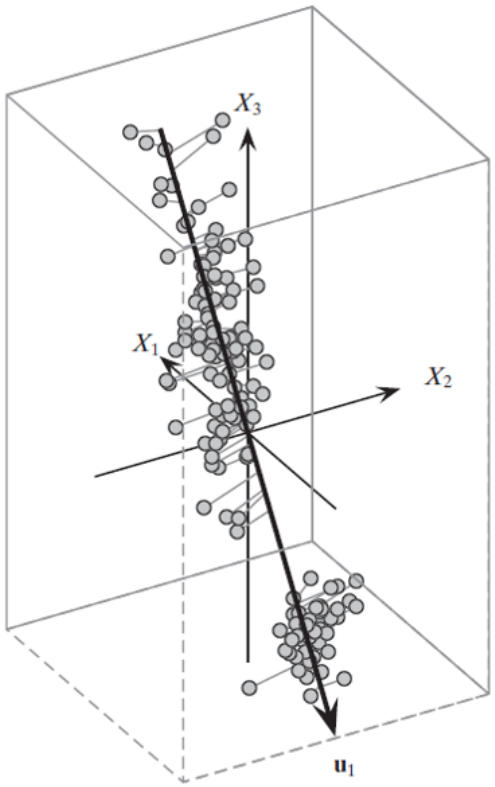


Figure 7.2. Best one-dimensional or line approximation.