Principal Component Analysis and Support Vector Machines

1 Introduction

The objective of this first practical is to get a grasp of Principal Component Analysis (PCA) and Support Vector Machines (SVM) with Python.

Your goal will be to classify botanical data associated to flower species, using supervised machine learning.

The method is supervised, in the sense that you have access to `flowerTrain.data`, a dataset quantifying the morphologic variation of flowers along three species: *interior*, *versicolor* and *convoluta*. The dataset (which you can read with any text editor) consists of 48 samples for each of the three species. Four features have been measured from each sample: length and width of the sepals, and length and width of the petals (all in cm, see Fig. 1).

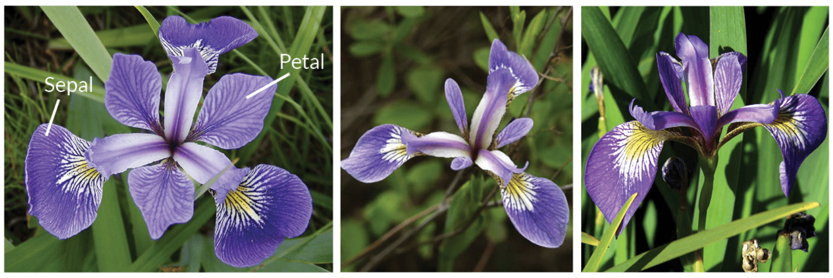

Figure 1: Flower samples of versicolor, interior and convoluta (Left to right).

Based on the combination of these four features, you must develop a linear discriminant model for determining whether a new sample is an *interior* flower or not (i.e., one of the two other species).

To this end, you will proceed as follows.

1. Practice Principal Component Analysis, by implementing it on 2-D datasets of four points, to reduce them to 1-D (Sect. 2.1).
2. Extend PCA to the whole 4-D flower dataset, to reduce it to two dimensions (Sect. 2.2).
3. Design a Support Vector Machine on the dataset of dimension reduced, to classify samples as either *interior* or *not interior* (Sect. 3).
4. Test the classifier on new samples (Sect. 4).

For this practical, you will need some python packages.

Add the following lines to your python script to ensure that all packages are imported correctly.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.svm import SVC
```

You are already familiar with `numpy` and with `matplotlib`.

`Pandas` is an open source data analysis and manipulation tool.

`sklearn` (Scikit-learn) is a library for machine learning, which you will specifically use here for support vector classification (`SVC`).

Let us get a first feeling of the data, by plotting it. You can do this with the following lines of code.

```
data = pd.read_csv("flowerTrain.data", names=['sepal
    length','sepal width','petal length','petal width','
    species'])
x = data.iloc[:,0:4]
plt.scatter(data.iloc[:,0], data.iloc[:,1], alpha=0.2,
    s=100*data.iloc[:,2], c=100*data.iloc[:,3])
```

The first line reads the data and converts it to Pandas' DataFrame type.

The second line stores the features in a numpy array.

The third line plots the data, as in Fig. 2. To represent the four dimensional data by points in a plane, we use the following convention. For each sample: the point abscissa and ordinate correspond to the sepal length and width, the point size is proportional to the petal length, and the color is proportional to the petal width. It is clear from the figure that classifying samples from this 4-D data is not trivial.

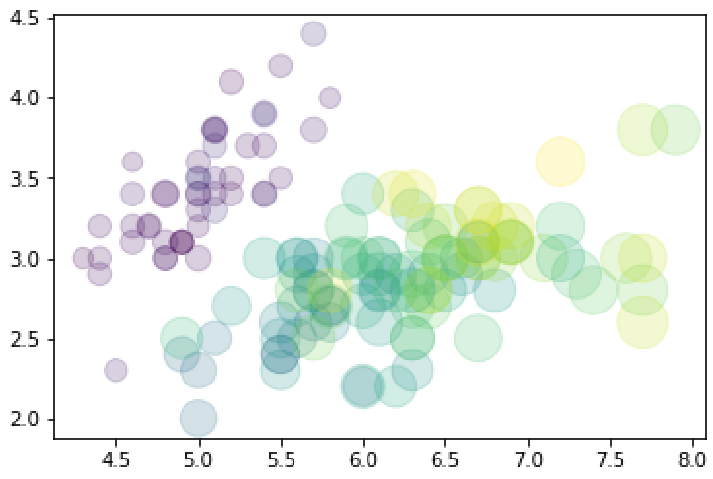

Figure 2: Scattered plot of the flower training dataset.

In the next Section, you will project the 4-D data to 2-D by selecting the Principal Components, to simplify classification.

2 Principal Component Analysis

Consider m samples in a n dimensional space:

$$\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_m] \in \mathbb{R}^{n \times m}, \tag{1}$$

which we want to reduce to the p principal dimensions. Principal Component Analysis consists in the following six steps:

1. Compute the mean of all measures:

$$\bar{\mathbf{x}} = \frac{1}{m} \sum_{1}^{m} \mathbf{x}_i \in \mathbb{R}^n. \tag{2}$$

2. Shift matrix $\mathbf{X}$ by this mean (this is equivalent to translating the coordinate system to the location of the mean):

$$\bar{\mathbf{X}} = [\mathbf{x}_1 - \bar{\mathbf{x}} \dots \mathbf{x}_m - \bar{\mathbf{x}}] \,. \tag{3}$$

3. Build the dataset covariance matrix

$$\mathbf{C} = \frac{1}{m-1} \bar{\mathbf{X}} \bar{\mathbf{X}}^\top. \tag{4}$$

4. Using Singular Value Decomposition, extract eigenvalues and normalized eigenvectors of $\mathbf{C}$:
$$\mathbf{C} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}^\top. \tag{5}$$

5. Choose the p largest eigenvalues and reduce $\mathbf{U}$ to the corresponding p eigenvectors (columns): $\mathbf{U}_p$

6. Project each sample to reduce its dimensions:
$$\mathbf{s} = \mathbf{U}_p^\top \left(\mathbf{x} - \bar{\mathbf{x}}\right). \tag{6}$$

In the rest of this section, you will apply PCA starting with two simple examples, and then with the flower dataset.

2.1 Practicing PCA on two examples

Consider the following data:

$$\mathbf{X} = \begin{bmatrix} 1 & 5 & 3 & 3 \\ 4 & 4 & 3 & 5 \end{bmatrix} \in \mathbb{R}^{2\times 4}. \tag{7}$$

The goal it to reduce the data from 2 dimensions to 1 dimension. Design a function `def PCA(X , reducedDim)` to implement the six steps of PCA. The function should plot the four samples (points) along with the eigenvectors, with the length of each eigenvector equal to the corresponding eigenvalue.

Useful functions: np.mean, np.linalg.eigh, np.argsort, plt.arrow...

Apply the same function to the following data:

$$\mathbf{X} = \begin{bmatrix} 1.268 & 4.732 & 3.5 & 2.5 \\ 3 & 5 & 3.134 & 4.866 \end{bmatrix} \in \mathbb{R}^{2\times 4}, \tag{8}$$

and comment the result.

2.2 Applying PCA to the flower dataset

You may now apply `PCA(X , reducedDim)` to the flower dataset. Since the dataset has dimension four, plotting the eigenvectors is not relevant here. The two-dimensional plot of the samples should look like Fig. 3. To draw this plot, you may need the `scatterplot` function present within the seaborn package:

```
import seaborn as sb
```

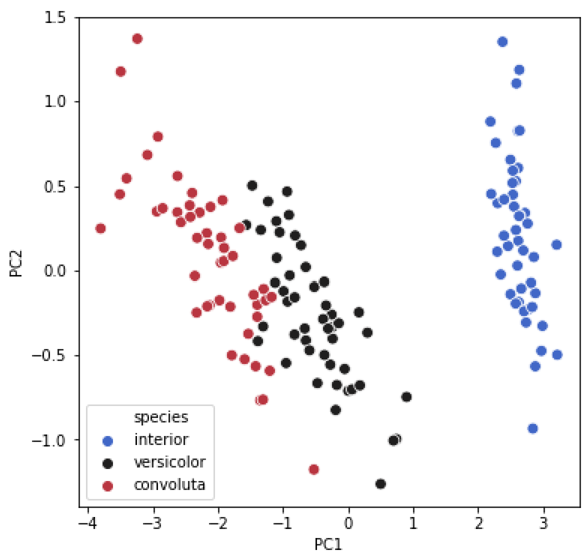

Figure 3: Principal components of the flower training dataset.

In this plot of the two principal components, clusters of samples of the three species are much clearer than in the original 4-D space. In particular, there is a clear separation between *interior* and the two other species. In the next section, you will design a linear classifier (support vector machine) for determining whether a sample corresponds to an *interior* flower or not.

3 Support Vector Machine

Let us recall the basic of classic SVM for binary linear classification. Consider a training dataset of m points in the form

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m),$$

with $y_i = 1$ or $y_i = -1$, depending to which class $\mathbf{x}_i$ belongs. Then the SVM classifier is:

$$\mathbf{x} \mapsto \operatorname{sgn}\left(\mathbf{w}^\top \mathbf{x} - b\right) \tag{9}$$

where $\mathbf{w}$ and b are the hyperplane parameters, determined by solving an optimization problem. In Scikit-learn, you can design the support vector machine by invoking

```
svm = SVC(kernel='linear', C=1E10)
```

and fit your data using `svm.fit`.

Using the new (PCA) data representation, design such `svm`. Then, draw the hyperplane $\mathbf{w}^\top \mathbf{x} - b = 0$ and the two decision boundaries $\mathbf{w}^\top \mathbf{x} - b = \pm 1$. The plot should look like Fig. 4.

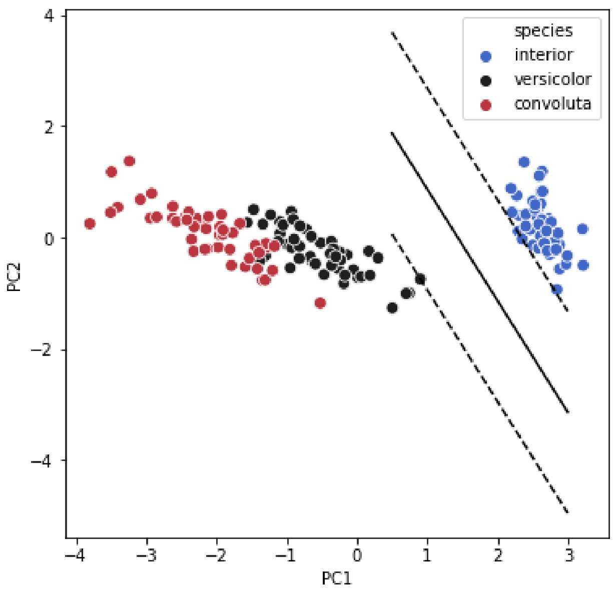

Figure 4: Hyperplane and decision boundaries in the PCA representation.

4 Classifying new samples

Now that you have designed a flower classifier using the trained data, you can apply to new samples, to determine whether they correspond to an *interior* flower or not. Try with the following samples:

Table 1: New samples to be classified.

Sepal Length	**Sepal Width**	**Petal Length**	**Petal Width**
5.1	3.5	1.4	0.2
7.0	3.2	4.7	1.4
6.4	3.2	4.5	1.5
6.3	3.3	6.0	2.5
5.8	2.7	5.1	1.9
4.9	3.0	1.4	0.2