Prof. Dr. Hauke Schramm
Department of Computer Science
Kiel University

Pattern Recognition

Exercises

Practice Sheet 3

Exercise L-3.1 (Data organisation)

In this exercise we will organize the data for classification of an iris plant into one of the following three categories: (1) Iris Setosa, (2) Iris Versicolour and (3) Iris Virginica. Four plant attributes have been collected by R.A. Fisher in his classical paper

Fisher,R.A. "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to Mathematical Statistics" (John Wiley, NY, 1950).

These are:

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm

The sepal and petal of an iris plant are shown in Figure 1.

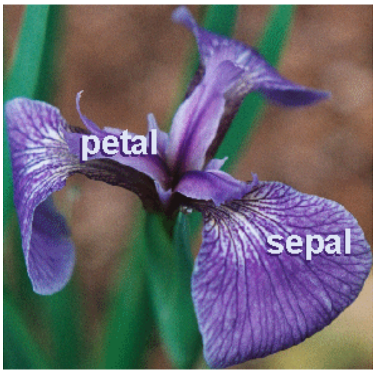

Figure 1. Iris plant with illustration of petal and sepal.

The data is stored in the OLAT laboratory folder (sheet3).

iris_data.mat

Gives the measurements described above with the following implicit column headers:

“Sepal Length” “Sepal Width” “Petal Length” “Petal Width”

50 measurements are provided for each class.

iris_species.mat

Gives the class of the respective plant for which measurements are given in the same row in iris_data.mat.

Task 1: Load the data and organize it into a 50-by-4-by-3 multidimensional array named `iris`. The first dimension (size: 50) labels the observations for each plant class, the second (size: 4) labels the 4 measurements given for each plant and the third (size: 3) labels the class.

Here is an example for the usage of the iris-array: A listing of the first five observations of sepal length in the Iris Virginica data is obtained by:

```
>> iris(1:5, 1, 3)

ans =

     6.3000
     5.8000
     7.1000
     6.3000
     6.5000
```

Note, that this array is a pure numerical array.

Hints: You may use the Octave functions `strcmp` and `cat` to solve this task.

Task 2: In the next step we want to overcome the implicit nature of the array (i.e. no labeling of the array-dimensions) by combining it with additional metadata in a **cell array** `iris1`. An Octave cell array allows you to combine variables of different types and sizes into a single data structure. The following hints may help you with this task:

- Initialize the cell array as follows:

```
>> iris1 = cell(51, 5, 3); % we need a larger array for
                           % saving the additional labels
```

- num2cell converts a numeric array to a cell array

Task 3: Write a function `printcell` for printing an arbitrary 3D cell array. Demonstrate its functionality with the given cell array `iris1`.

Example:

```
>> printcell(iris1)

Setosa      SepalLength   SepalWidth   PetalLength   PetalWidth
Obs1           5.10          3.50          1.40          0.20
Obs2           4.90          3.00          1.40          0.20
…              …             …             …             …
Obs50          5.00          3.30          1.40          0.20

Versicolor  SepalLength   SepalWidth   PetalLength   PetalWidth
Obs1           7.00          3.20          4.70          1.40
Obs2           6.40          3.20          4.50          1.50
…              …             …             …             …
Obs50          5.70          2.80          4.10          1.30

Virginica   SepalLength   SepalWidth   PetalLength   PetalWidth
Obs1           6.30          3.30          6.00          2.50
Obs2           5.80          2.70          5.10          1.90
…              …             …             …             …
Obs50          5.90          3.00          5.10          1.80
```

Task 4: For the organization of heterogeneous data, cell arrays are very useful. However, they are a little cumbersome when it comes to manipulating and analyzing the data. Since statistical functions in Octave do not accept cell array data, it must be extracted to a numerical container variable. This can be achieved using row, column curly brace indexing, as shown in the following example:

```
>> subset = reshape([iris1{2:6, 2:3, 1}], 5, 2)
```

In this example, curly brace indexing is used in order to access the data elements (i.e. extract the numbers instead of the cells).

Define the container variables `setosa`, `versicolor` and `virginica` and fill them with the corresponding measurements for Sepal Length, Sepal Width, Petal Length and Petal Width.

Example:

```
>>  setosa(1:3,:)

setosa =

   5.10000   3.50000   1.40000   0.20000
   4.90000   3.00000   1.40000   0.20000
   4.70000   3.20000   1.30000   0.20000
```

Task 5: Compute mean and variance of Iris Versicolour's sepal and petal length and width and write them into a cell array `mv_array.` Print the array using your function `printcell`.

```
>> printcell(mv_array)

                      SepalLength      SepalWidth      PetalLength      PetalWidth
mean                    xxxxxx           xxxxxx          xxxxxx           xxxxxx
variance                xxxxxx           xxxxxx          xxxxxx           xxxxxx
```

xxx: actual result for mean or variance

<u>Visualizations:</u> Next, we want to visualize the data in a **scatter plot** using the function `plot`. This function visualizes data points in a 2D array `x` as blue circles by

```
>> plot(x(:,1),x(:,2),"bo");
```

Alternatively, you may use "r*" for red stars.

Task 6: Generate scatter plots for the different subsets of data, e.g. visualize the distribution of data points for iris setosa and versicolor sepal length and width. Use different markers to illustrate the individual classes.

Which data sets can / can not be separated using a linear decision boundary?

Label the axes appropriately, e.g. with "Sepal Length" and "Sepal Width" using the functions `xlabel` and `ylabel`. Add an adequate heading using the function `title`. Use the function `legend` to add a labeling of the different data points.

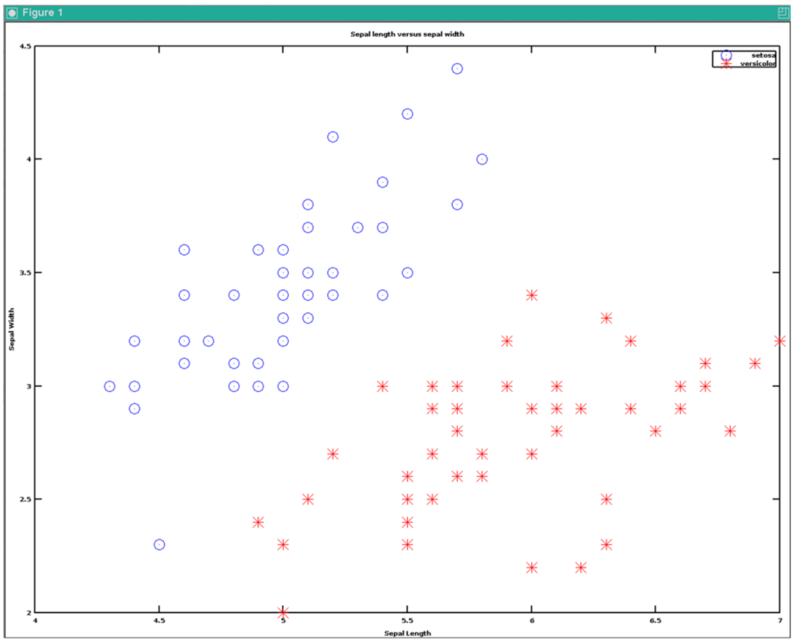

Figure 2. Scatter plot for sepal length and width for iris setosa and versicolor.