2. Assignment

kNN and Bayes Theorem

Deadline: 18.11.2016

Description of assignment

This assignment covers the following topics:

- kNN classifier (task 1)
- Bayes theorem (task 2)

The aim is to get familiar with these topics, solve a simple classification problem and compare the results. This assignment is based on the lectures 01 to 03.

Guidelines

(1) NEW: Edit the provided file `readme.txt` to inform us about your environment (i.e.: Octave or Matlab version, operating system).
(2) It is of utmost importance that your **code is executable**. Make sure that all results are printed and plotted by executing `main.m` and no errors occur.
(3) In the **second line of every file** fill in the following information as a comment: first name, last name and student id.
(4) All provided script and function files have to be used and completed if necessary. If you create additional files make sure that the code is still executable by `main.m`.
(5) All files have to be collected in a **zip-archive** and be uploaded in TUWEL (until the deadline).

Fisher's Iris data set

This data set contains 50 samples of three lilies. Four features have been collected for each sample:

(1) length of Sepalum
(2) width of Sepalum
(3) length of Petalum
(4) width of Petalum

For the toy example in task 1, this data set is reduced to a *two-class-problem*. In Figure 1 you can see the two lilies used for this assignment.

FIGURE 1. Left: Iris virginica (class 1); Right: Iris versicolor (class 2).

The lily data set has already been imported for you in `main.m`. Look at the data file `datasetLily.m` to get an idea of the data.

Leafs data set

This data set is much more complex than the Fisher's Iris data set. It consists of 30 classes, where for each sample the following 14 features have been extracted:

(1) Eccentricity
(2) Aspect Ratio
(3) Elongation
(4) Solidity
(5) Stochastic Convexity
(6) Isoperimetric Factor
(7) Maximal Indentation Depth
(8) Lobedness
(9) Average Intensity
(10) Average Contrast
(11) Smoothness
(12) Third moment
(13) Uniformity
(14) Entropy

For details about the extracted features look at `Leafs.pdf`. The Leafs data set has also been imported for you in `main.m`. Look at the data files `trainingsetLeafs.m` and `testsetLeafs.m` to get an idea of the data.

Task 1: k-Nearest-Neighbors

Total points: 15
Files: `kNNTasks.m`, `normalizeData.m`, `kNN.m`, `selectFeatures.m`, `evaluate.m`

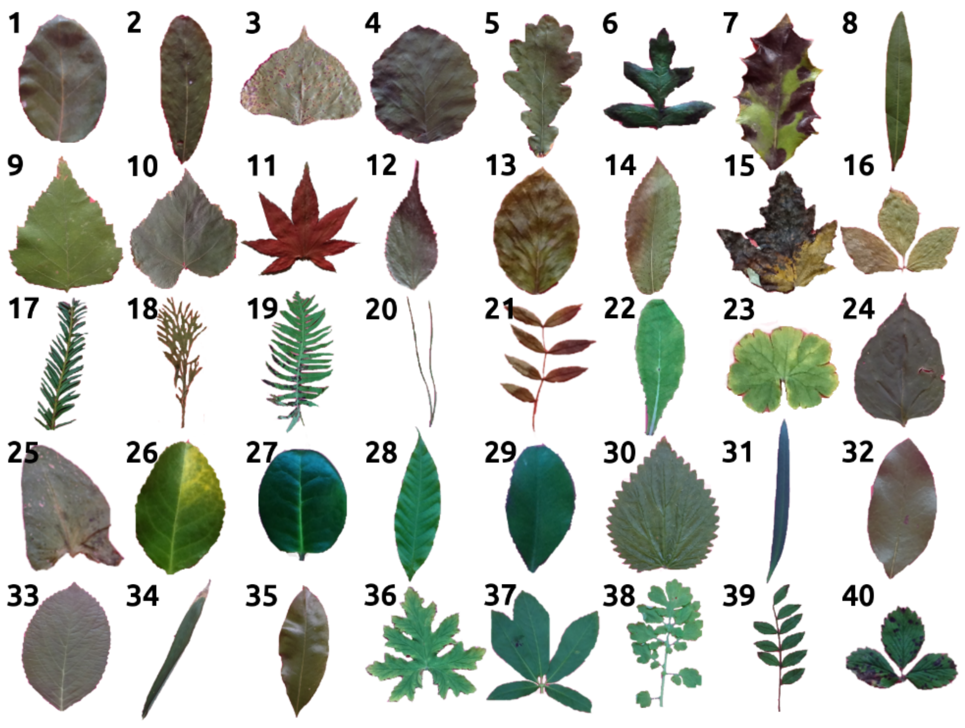

FIGURE 2. Leafs 1 – 15 and 22 - 36 are included in the data set.

Task 1.1: Normalization [2 Points]. It is necessary to prepare the data of the two data sets, before you can solve the classification problem. Normalize the imported data of the two data sets as in the following equation:

$$\overline{x_i} = \frac{x_i - x_{min}}{x_{max} - x_{min}}, \tag{1}$$

where x is a certain type of feature, x_i is a measurement of the feature for sample i, $\overline{x_i}$ is its normalized value, and x_{min} and x_{max} are the smallest and the biggest measurement of feature x.

Implement the provided function `normalizeData(...)` (see `TASK 1.1`).

Important remark: Do not normalize the class labels (first row in the data set matrices).

Task 1.2: Your own kNN classifier [8 Points]. This task is about implementing your own kNN classifier (see `TASK 1.2` in `kNN.m`). Hence, it is not allowed to use the Octave/Matlab function `knnsearch`.

Implement the functionality of the existing function `kNN(...)`. Your kNN classifier should be able to:

- classify all samples of the test set based on the samples in the training set

- using the neighborhood size k
- handle sets with arbitrary number of features (1 or more)
- classify samples into any number of classes (2 or more)

You can assume that the user feeds the function with correct input parameters. E.g. number of features in training and test set match, etc.

The output parameter `classification` of the function is a vector containing the class labels of the samples in the test set.

Remark: This output parameter is used in the `selectFeatures(...)` function to find the smallest error and the best k (already implemented). Line 8 and 14 in `kNNTasks.m` print out the smallest error and best k if all features are selected for the Lily and the Leafs data set.

Task 1.3: Feature selection Lily data set [5 Points]. In this task you should use the provided function `selectFeatures(...)` to experiment with different feature combinations (call `selectFeatures(...)` in `kNNTasks.m`). First, make yourself familiar with the parameters and the functionality of this function.

Then, use the function to answer the following questions in the report (`report.odt`) for the **Lily data set**:

(1) What is the smallest error you can achieve on the test set?
(2) What is the best feature combination based on the error?
(3) How did you find the best combination?
(4) What is the smallest k, which led to the smallest error?
(5) Why is it a good idea to normalize the data before applying the kNN classifier?

Remark: It does not matter with which k you achieve the smallest error. As long as k lies in the predefined interval $[1, 23]$.

Bonus Task 1.4: Feature selection Leafs data set [5 Points]. This task is optional and can be used to collect bonus points.

Answer the following questions (in the report) regarding the **Leafs data set**:

(1) Which feature combination outperforms the use of all features? (the best you can find based on the error)
(2) For which k can you achieve this result? Why do you think this k leads to the best result?
(3) What could be done to improve the performance (reduce the error)? Some ideas ...

Task 2: Bayes theorem

Total points: 15
Files: `bayesTasks.m`, `determineWeightedLikelihood.m`, `determineEvidence.m`, `determinePosterior.m`, `bayesDecision.m`, `determineError.m`

This task is about the Bayes theorem for *two-class-problems*. Its aim is to study and understand the different elements of the Bayes theorem:

$$posterior = \frac{likelihood \times prior}{evidence} \tag{2}$$

Task 2.1: Weighted Likelihoods [2 points]. In this task, the aim is to calculate the numerator of Equation 2, the weighted likelihoods.

We assume that the density functions of the classes (class conditional pdfs) are normally distributed $N = (\mu, \sigma^2)$.

In `bayesTasks.m`, the parameters of the distributions of the two classes (w1 and w2) are set as follows:

```
% means
meanW1 = 2;
meanW2 = 4;

% variances
varW1 = 2;
varW2 = 2;
```

Implement the provided function `determineWeightedLikelihood(...)`, which determines the weighted likelihood of a class in the provided interval `X = minX:0.001:maxX;`. Hence, the output parameter is a vector containing the weighted likelihoods (see `TASK 2.1`).

After implementing this function, the script in `bayesTasks.m` should run error-free until line 42 and plot a figure of the calculated weighted likelihoods (see Figure 3).

Task 2.2: Evidence [1 point]. From the weighted likelihoods it is now possible to determine the evidence (normalization factor).

Complete the existing function `determineEvidence(...)` (see `TASK 2.2`).

If you are successful you should be able to plot the evidence with the provided code (see Figure 4).

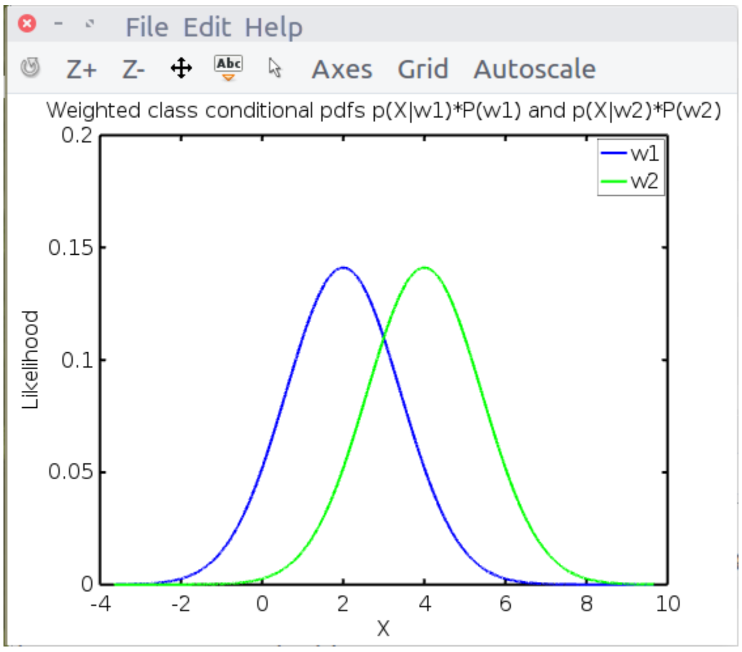

FIGURE 3. Likelihoods of classes w1 and w2.

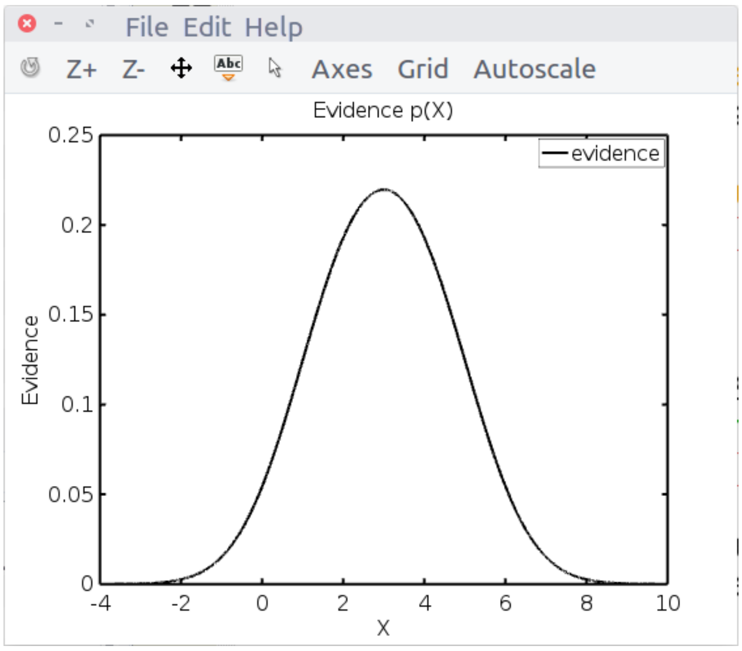

FIGURE 4. Evidence.

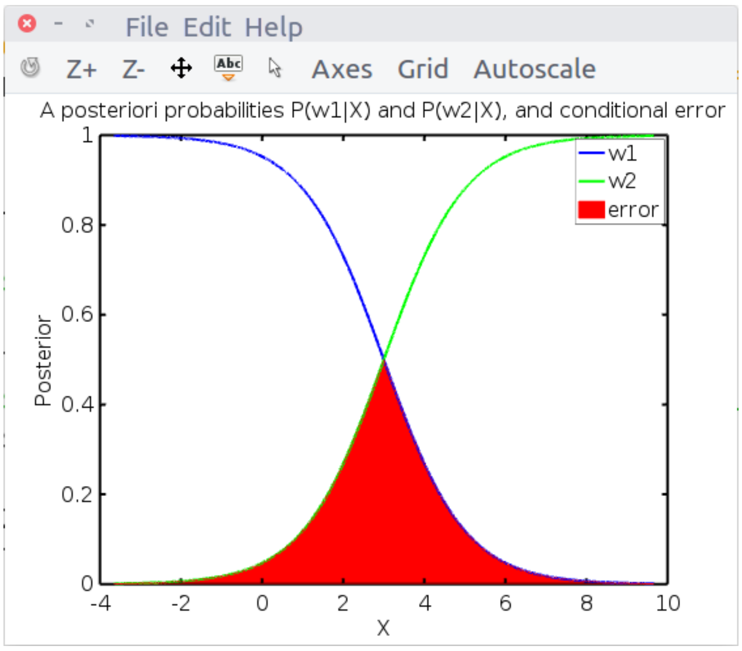

FIGURE 5. Posteriors of classes w1 and w2, and conditional error.

Task 2.3: Posteriors and classification [4 points]. The aim of this task is to calculate the posteriors from the likelihoods, priors and evidence. Afterwards, you are able to apply the Bayes decision rule.

Complete the existing function `determinePosterior(...)` (see `TASK 2.3.1`) to determine the posterior for each class in the interval X.

Then, apply the Bayes decision rule based on the previously determined posteriors. For this, implement the Bayes decision rule in the prepared function `bayesDecision(...)` (see `TASK 2.3.2`). The result of the classification with `bayesDecision(...)` for all elements in X should be stored in the output parameter (vector) `classification`.

Task 2.4: Conditional error probability [2 points]. Based on the Bayes decision rule you can determine the conditional error probability $P(error|X)$.

Calculate the error probability for all X by the function `determineError(...)` (see `TASK 2.4`).

Afterwards, it is possible to visualize the posteriors and the conditional error probability by the provided plotting code (see Figure 5).

Task 2.5: Experiments and discussion [6 points]. All questions in this task should be answered/discussed in the report (`report.odt`).

Currently, the Priors of both classes (w1 and w2) are set to $0,5$. Modify the Priors so that $P(w_1) = 0,3$ and $P(w_2) = 0,7$, and $P(w_1) = 0,7$ and $P(w_2) = 0,3$. Now you have three plots for likelihoods, evidence and posteriors. Put the plots into your report. Answer the following questions (about 1/2 a page) based on what you can see in the plots:

(1) How are the weighted likelihoods influenced?
(2) How are the posteriors influenced?
(3) What happens to the decision boundary (based on Bayes decision rule)?