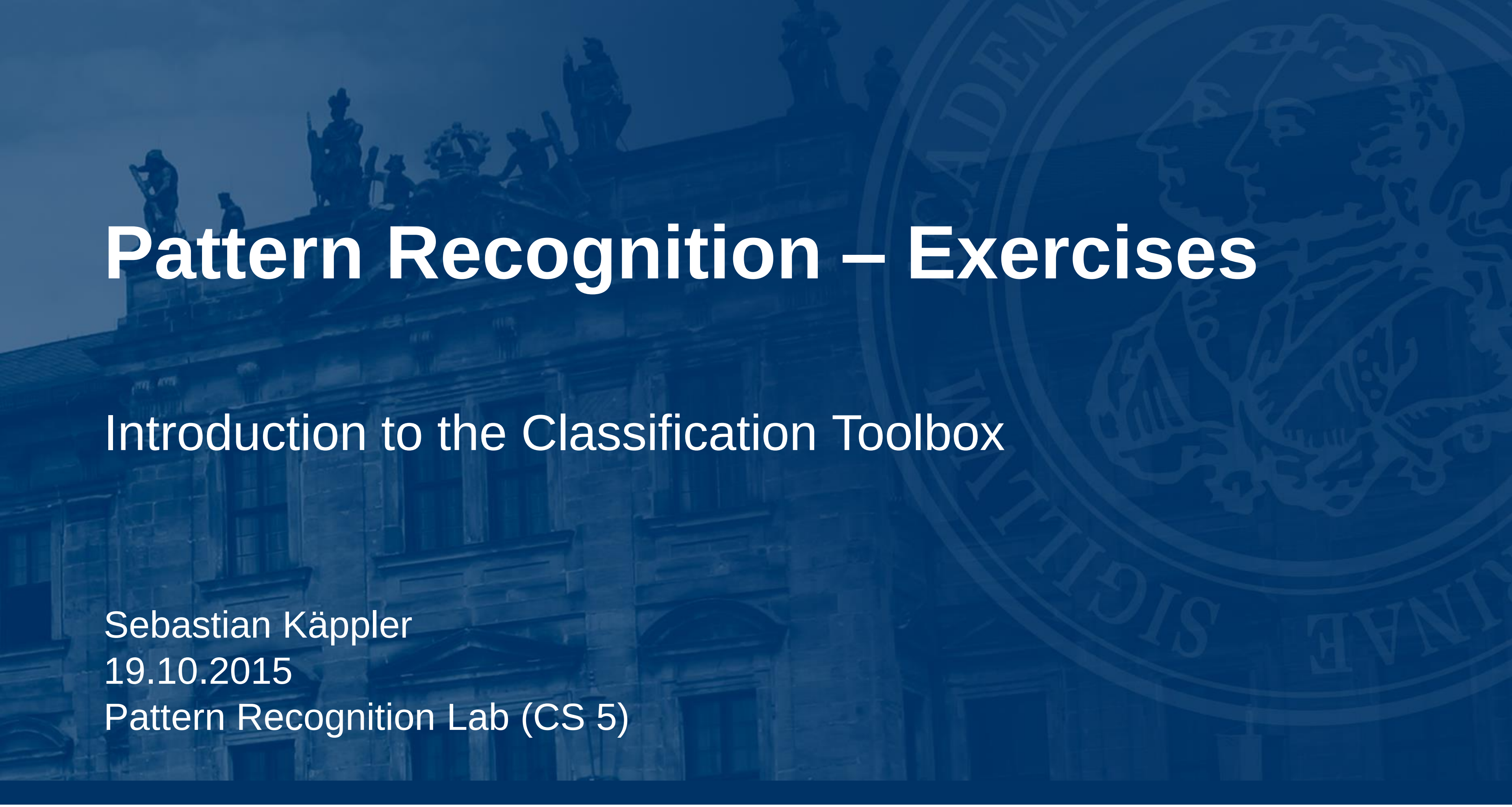


Pattern Recognition – Exercises

Introduction to the Classification Toolbox

Sebastian Käppler
19.10.2015
Pattern Recognition Lab (CS 5)

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Exercises

- Theoretical and practical assignments
- No need to hand-in your results

- Requirements
  - Mathematical background: statistics, calculus, linear algebra
  - Useful reference for linear algebra: The Matrix Cookbook
  - MATLAB programming

- Programming tasks: Extending the functionality of the **classification toolbox**
  - Preprocessing algorithms
  - Data sampling
  - Classification algorithms

Classification Toolbox

- Set of algorithms for pattern classification implemented in MATLAB
- Based on the Computer Manual in MATLAB to accompany Pattern Classification (Richard Duda)
- Types of files
  - Control routines for the GUI (e.g. classifier)
  - Preprocessing and feature selection algorithms
  - Error estimation methods
  - Clustering algorithms (e.g. k-means)
  - Classification algorithms

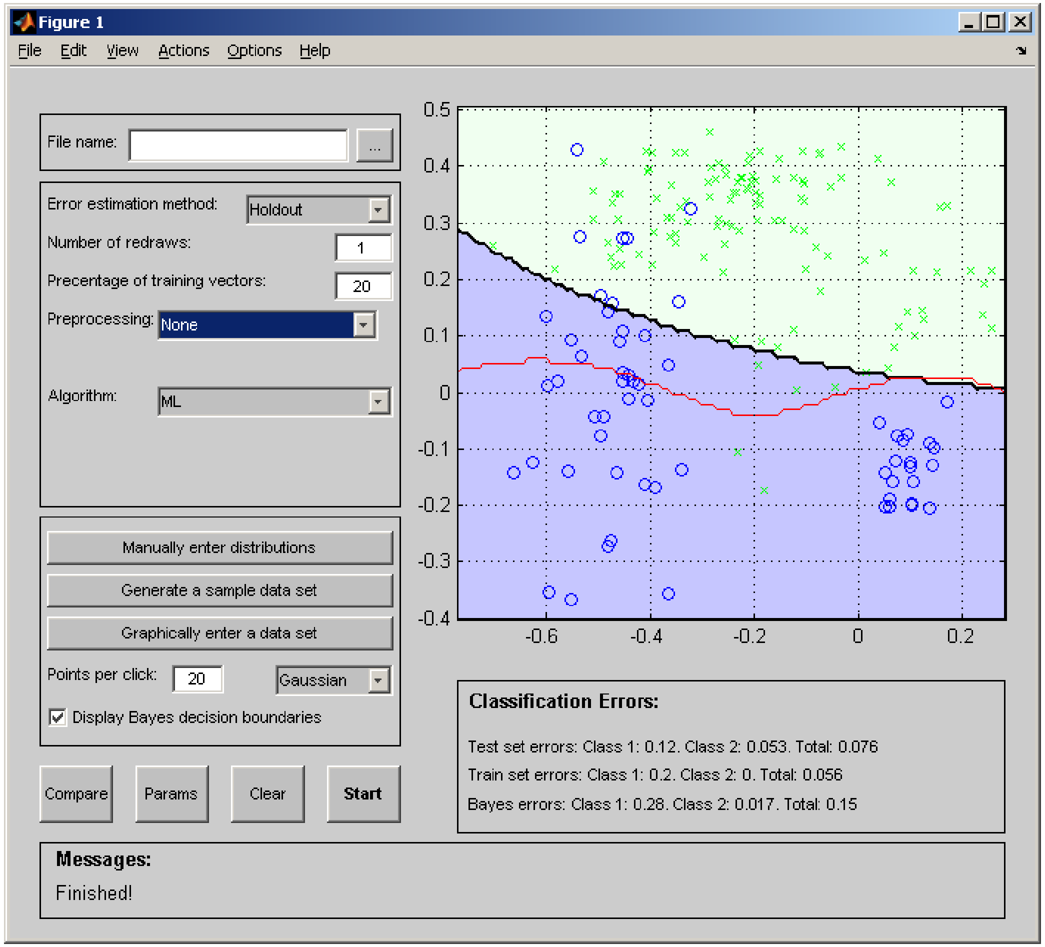
Starting Point

- Download base package of the classification toolbox from the exercise homepage http://www5.cs.fau.de/lectures/ws-1415/pattern-recognition-pr/exercises/ and expand it to a local directory **(Login required!)**

- The base package does not contain classification algorithms
- You will implement the algorithms of the lecture during the semester

Usage of the Classification Toolbox

- Set MATLAB working directory to the toolbox
- Call toolbox starting routine:
  `>> classifier`
- GUI can be used to
  - Create samples
  - Preprocess
  - Classify
  - Evaluate results



File name:

Error estimation method: Holdout

Number of redraws: 1

Precentage of training vectors: 20

Preprocessing: None

Algorithm: ML

Points per click: 20 Gaussian

☑ Display Bayes decision boundaries

Classification Errors:

Test set errors: Class 1: 0.12. Class 2: 0.053. Total: 0.076

Train set errors: Class 1: 0.2. Class 2: 0. Total: 0.056

Bayes errors: Class 1: 0.28. Class 2: 0.017. Total: 0.15

Messages:

Finished!

Toolbox Architecture

- Copy M-file of algorithm to toolbox directory
- Add a reference line in `Classification.txt` with the format:

  `<Algorithm name>@<Caption>@<Default Parameters>@<Display field>`

  - `<Algorithm name>:` name of algorithm AND M-file
  - `<Caption>:` caption to be displayed near the parameter entry box
  - `<Default parameters>:` set of parameters given as initial set
  - `<Display field>:` indicates whether parameters are needed or not
    - Type `N` in this field if no parameters are needed
    - Type `S` to open a short parameter window in the GUI
    - Type `L` to open a long parameter window whenever algorithm is invoked

- Describe the algorithm in `contents.m`

Toolbox Architecture

- Examples: `Classification.txt`
  - `NearestNeighbor@ @ @N`
  - `KNearestNeighbor@Num of nearest neighbors:@3@S`
  - `SVM@Kernel, Ker param, Solver, Slack:@['RBF', 0.05, 'Perceptron', inf]@L`

- Examples: `Contents.m`

```
% Parametric classification algorithms
%     ML - Maximum likelihood algorithm
%
% Non-parametric classification algorithms
%     NearestNeighbor - Nearest neighbor algorithm
```

Toolbox Architecture

- Inputs:
  - Patterns used for training: `train_patterns`
  - Labels for training samples: `train_targets`
  - Patterns used for test: `test_patterns`
  - Optional parameters: `parameters`

- Output: `test_targets`

- Example:

```
% Classify using the nearest neighbor algorithm
function test_targets =
NearestNeighbor(train_patterns, train_targets,
test_patterns, parameters)
```

Nearest neighbor algorithm

- Nearest neighbor classifier:
  assign a test pattern $(\boldsymbol{x}^*, y^*)$ to the class of the closest training pattern $(\boldsymbol{x}_i, y_i)$

$$y^* = y_{i^*}$$

$$i^* = \underset{i}{\operatorname{argmin}} \|\boldsymbol{x}^* - \boldsymbol{x}_i\|$$

- Implement as MATLAB function
- Integrate M-file to classification toolbox
- Generate training/test patterns in GUI and test the algorithm