# Mustererkennung - Übung 1

Samuel Gfrörer

2017-10-25

## 1 Übung 1 - K-NN Klassifikator

### 1.1 Ausführen des Programms

Das Programm kann folgendermaßen ausgeführt werden:

```
$ ./scripts/run_test.sh
```

### 1.2 Implementierung

Der Code wird als Github-Repository verwaltet (siehe https://github.com/EsGeh/pattern-recognition). Um das Programm selbst zu installieren und zu kompilieren, siehe unten.

#### 1.2.1 Ordnerstruktur

```
.
|-- test
|   `-- NearestNeighboursTest.hs
|-- resource
|   |-- ...
|-- src
|   |-- PatternRecogn
|   |   `-- NearestNeighbours.hs
|   `-- ...
|-- app
|-- plots
|   |-- ...
```

Das Programm teilt sich auf in eine eine Bibliothek (siehe Verzeichnis "./src") und Tests (siehe Verzeichnis "./test"). Für diese Übung sind folgende Dateien im Quellcode relevant:

- ./src/NearestNeighbours.hs: die Implementierung des K-NN Algorithmus
- ./test/NearestNeighboursTest.hs: testen des Algorithmus

### 1.2.2 Qellcode-Ausschnitte:

./src/NearestNeighbours.hs

```
{-# LANGUAGE ScopedTypeVariables #-}
module PatternRecogn.NearestNeighbours where

import PatternRecogn.Lina as Lina
import PatternRecogn.Types

import Data.List( sortBy )
import Data.Tuple( swap )
import qualified Data.Map as M


-- this method will use the training set for classification
type ClassificationParam =
    TrainingData

-----------------------------------------------
-- calculate classification parameter:
-----------------------------------------------

-- nothing to do here: just use training data as parameter for classification
calcClassificationParams :: TrainingData -> ClassificationParam
calcClassificationParams = id


-----------------------------------------------
-- classify new feature vectors based on the classification parameter:
-----------------------------------------------

-- classify a row of unknown feature vectors based on the training set
classify ::
    Int -- ^ how many neighbours to consider
    -> ClassificationParam ->
    Matrix -> VectorOf Label
classify k trainingData =
    Lina.fromList
    .
    map (classifySingleVector k trainingData)
    .
```

```
    Lina.toRows

classifySingleVector ::
    Int -- ^ how many neighbours to consider
    -> ClassificationParam ->
    Vector -> Label
classifySingleVector k trainingData input =
    snd . M.findMax . swapMap $
    count $
    map snd $ -- extract the label
    take k $
    sortBy (\x y -> closestInput (fst x) (fst y)) $
    trainingData
    where
        closestInput x y =
            (norm_2 $ x - input)
            `compare`
            (norm_2 $ y - input)

swapMap :: Ord a => M.Map k a -> M.Map a k
swapMap = M.fromList . map swap . M.toList

count :: forall a . Ord a => [a] -> M.Map a Int
count =
    foldr conc M.empty
    where
        conc :: a -> M.Map a Int -> M.Map a Int
        conc x acc =
            case M.lookup x acc of
                Nothing -> M.insert x 1 acc
                Just _ ->
                    M.adjust (+1) x acc
```

## 1.3 Eingabe- und Ausgabedateien

Der Test kann mit folgendem Befehl ausgeführt werden:

```
$ ./scripts/run_test.sh
```

Im Ordner "./resource" befinden sich die Trainingsdatensätze und der Testdatensatz. Die Ausgabe erfolgt über die Standardausgabe.

## 1.4 Die Funktionalität des Tests

Der Trainingsdatensatz für die Tests befindet sich im Verzeichnis "./resource/train.*". Es handelt sich um einen Datensatz zur Erkennung der Ziffern 0,...,9 aus Handschrift. Der Datensatz stammt von der Webseite zum Buch:

- Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani: An Introduction to Statistical Learning with Applications in R

Das Programm testet die Güte der Klassifizierung anhand des K-NN Algorithmus mit $k = 1, ..., 5$

## 1.5 Ausgabe des Programms

```
$ ./scripts/test_neuralNetworks.sh
loading all data from file...
testing nearest neighbour classification...
k = 1
training data quality: 1.0
confusion matrix:
10x10
0.16  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.14  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.10  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.08  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.07  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09

test data quality: 0.9436970602889886
confusion matrix:
10x10
0.16  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.14  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.10  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.08  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.07  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09
```

```
k = 2
training data quality: 0.9883417912494856
confusion matrix:
10x10
0.16  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.14  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.10  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.07  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.07  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09

test data quality: 0.935226706527155
confusion matrix:
10x10
0.16  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.14  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.10  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.07  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.07  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09

k = 3
training data quality: 0.9872445480729667
confusion matrix:
10x10
0.16  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.14  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.10  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.07  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.07  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09

test data quality: 0.9456900847035377
confusion matrix:
```

```
10x10
0.16  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.14  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.10  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.07  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.07  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09

k = 4
training data quality: 0.9823069537786312
confusion matrix:
10x10
0.16  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.14  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.10  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.07  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.07  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09

test data quality: 0.9436970602889886
confusion matrix:
10x10
0.16  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.14  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.10  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.07  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.07  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09

k = 5
training data quality: 0.9798381566314635
confusion matrix:
10x10
0.16  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
```

```
0.00  0.14  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.10  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.07  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.07  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09

test data quality: 0.9431988041853513
confusion matrix:
10x10
0.16  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.14  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.10  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.07  0.00  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09  0.00  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.07  0.00
0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.09
```

## 1.6 Kompilieren des Programms

### 1.6.1 Abhängigkeiten

- git (siehe https://git-scm.com/)
- stack (siehe https://docs.haskellstack.org/)

### 1.6.2 Kompilieren

```
$ git clone https://github.com/EsGeh/pattern-recognition
$ git checkout exercise8-release
$ stack setup
$ stack build
```