

Mustererkennung - Übung 3

Marcel Schmidt, Samuel Gfrörer

2017-11-07

1 Gauß-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

```
.
|-- doc
|   |-- Uebung3.pdf
|   `-- ...
|-- scripts
|   `-- run_test.sh
|-- src
|   |-- PatternRecogn
|   |   |-- Gauss
|   |   |   |-- Classify.hs
|   |   |   |-- Types.hs
|   |   |   `-- Utils.hs
|   |   |-- Lina.hs
|   |   `-- ...
|   `-- PatternRecogn.hs
|-- test
|   |-- GaussTest.hs
|   `-- ...
```

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

- “./src/Gauss/Classify.hs”: hier findet sich die Implementierung des Algorithmus
- “./src/Gauss/Types.hs”: Die relevanten Typ-Definitionen
- “./src/Gauss/Utils.hs”: einige Hilfsfunktionen
- “./test/GaussTest.hs”: Hier ist der Test definiert

1.2.2 Quellcode-Ausschnitte:

- ./src/PatternRecogn/Gauss/Types.hs:

```
{-# LANGUAGE RecordWildCards #-}
module PatternRecogn.Gauss.Types where

import PatternRecogn.Types
import Data.List( intercalate )

type Classes = [Class]
data Class
  = Class {
      class_min :: Vector,
      class_cov :: Matrix
    }
  deriving( Show )

class_prettyShow :: Class -> String
class_prettyShow Class{..} =
  intercalate "\n" $
  [ concat $ ["average: ", show class_min]
  , concat $ ["cov: ", show $ class_cov]
  ]
```

- ./src/PatternRecogn/Gauss/Classify.hs:

```
{-# LANGUAGE FlexibleContexts #-}
module PatternRecogn.Gauss.Classify(
  ClassificationParam,
  calcClassificationParams,
  classify,
  infoStringForParam
) where

import PatternRecogn.Lina
import PatternRecogn.Gauss.Utils
import PatternRecogn.Gauss.Types
import PatternRecogn.Types
import PatternRecogn.Utils
```

```

import Data.List( intercalate, maximumBy )

type ClassificationParam = [(Class, Label)]

-----
-- general gauss classification:
-----

calcClassificationParams :: TrainingDataBundled -> ClassificationParam
calcClassificationParams trainingData =
  map `flip` trainingData $ mapToFst $
    \set ->
      let
        center = average $ toRows set
      in
        Class{
          class_min = center,
          class_cov = cov_SAFE center set
        }

classify :: ClassificationParam -> Matrix -> VectorOf Label
classify param =
  fromList
  .
  map classifySingleVec
  .
  toRows
  where
    classifySingleVec :: Vector -> Label
    classifySingleVec vec =
      snd $
        maximumBy (\x y -> fst x `compare` fst y) $
        map `flip` param $ mapToFst $
        \Class{ class_min = center, class_cov = cov } ->
          mahalanobis center cov vec

```

...

- ./src/PatternRecogn/Gauss/Utils.hs:

```

module PatternRecogn.Gauss.Utils where

import PatternRecogn.Types

```

```

average x =
  (/ fromIntegral (length x)) $
  sum $
  x

```

```

cov_SAFE center set =
  if det cov > 0.01
  then cov
  else cov + alpha * ident (rows cov)
  where
    cov = covariance center set
    alpha = 0.01

covariance :: Vector -> Matrix -> Matrix
covariance center set =
  let
    centered = set - repmat (asRow center) countSamples 1
    countSamples = rows set
  in
    (/ fromIntegral countSamples) $
    sum $
    map (\v -> v `outer` v) $
    toRows $
    centered

mahalanobis :: Vector -> Matrix -> Vector -> R
mahalanobis center cov x =
  let
    centeredX = x - center
    inputDist =
      centeredX `dot` (inv cov #> centeredX)
  in
    1/sqrt (det (2 * pi * cov))
    *
    exp (-1/2 * inputDist)

```

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

Der Test testet die Güte der Klassifizierung anhand des Gauss-Algorithmus. Zunächst werden für alle Paare aus der Liste [3,5,7,8] binäre Klassifikatoren gebaut und getestet. Danach wird ein Klassifikator für alle 4 Klassen gebaut und getestet.

1.5 Ausgabe des Programms

```
testing binary classification:
testing classification of "resource/train.3", "resource/train.5"
testing gauss classification:
training data quality: 0.999176276771005
confusion matrix:
2x2
0.541  0.000
0.001  0.458

test data quality: 0.9447852760736196
confusion matrix:
2x2
0.541  0.000
0.001  0.458

testing classification of "resource/train.3", "resource/train.7"
testing gauss classification:
training data quality: 1.0
confusion matrix:
2x2
0.505  0.000
0.000  0.495

test data quality: 0.987220447284345
confusion matrix:
2x2
0.505  0.000
0.000  0.495

testing classification of "resource/train.3", "resource/train.8"
testing gauss classification:
training data quality: 0.995
confusion matrix:
2x2
0.543  0.000
0.005  0.452

test data quality: 0.9367469879518072
confusion matrix:
2x2
0.543  0.000
0.005  0.452
```

testing classification of "resource/train.5", "resource/train.7"
testing gauss classification:
training data quality: 1.0
confusion matrix:
2x2
0.463 0.000
0.000 0.537

test data quality: 0.9869706840390879
confusion matrix:
2x2
0.463 0.000
0.000 0.537

testing classification of "resource/train.5", "resource/train.8"
testing gauss classification:
training data quality: 0.9972677595628415
confusion matrix:
2x2
0.504 0.000
0.003 0.494

test data quality: 0.9447852760736196
confusion matrix:
2x2
0.504 0.000
0.003 0.494

testing classification of "resource/train.7", "resource/train.8"
testing gauss classification:
training data quality: 0.9991575400168492
confusion matrix:
2x2
0.543 0.000
0.001 0.457

test data quality: 0.9712460063897763
confusion matrix:
2x2
0.543 0.000
0.001 0.457

testing classification:
testing classification of "resource/train.3", "resource/train.5", "resource/train.7", "res
testing gauss classification:
training data quality: 0.9954185755935027
confusion matrix:
4x4
0.271 0.000 0.000 0.000

```
0.000 0.230 0.000 0.000
0.000 0.000 0.268 0.000
0.002 0.001 0.000 0.226
```

test data quality: 0.9123630672926447

confusion matrix:

4x4

```
0.271 0.000 0.000 0.000
0.000 0.230 0.000 0.000
0.000 0.000 0.268 0.000
0.002 0.001 0.000 0.226
```

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 WS2017_exercise3_release
$ stack setup
$ stack build
```