Mustererkennung - Übung5

Semjon Kerner, Philip Schmidt, Samuel Gfrörer

2016-11-25

1 Fischerdiskrimination

1.1 Ausführen des Programms

Das Programm kann folgendermaßen ausgeführt werden:

```
$ ./scripts/run.sh
```

1.2 Implementierung

Der gesamte Code sowie die ausführbare Datei befindet sich im beigefügten Archiv. 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

```
.
|-- app
|   `-- Main.hs
|-- plots
|   |-- ...
|-- resource
|   |-- ...
|-- src
|   |-- PatternRecogn
|   |   `-- Gauss.hs
|   `-- ...
```

Der Code ist folgendermaßen aufgeteilt:

1. Ausführbare Datei (siehe "./app")

   In der "./app/Main.hs" befindet sich der Code zum Einlesen der Test-Daten.

2. Bibliothek (siehe "./src")

   Hier befindet sich die eigentliche Funktionalität des Klassifizierungsalgorithmus

3. Eingabe- und Ausgabedateien

Im Ordner "./resource" befinden sich die Trainingsdatensätze und der Testdatensatz. Im Ordner "./plots" befindet sich nach der Ausführung des Programms die Diagramme der Verteilungskurven entlang dem mittels der Fischerdiskrimination errechneten Vektor.

1.2.2 Die Funktionalität des Programms

Das Programm wählt nacheinander alle 2-Tupel der Trainingsdatensätze in "./resource/train.*" und klassifiziert die Testdaten in "./resource/zip.test":

```
41 main :: IO ()
42 main =
43     handleErrors $
44     do
45         mapM_
46             (uncurry4 testWithData . uncurry testParamsFromLabels) $
47             allPairs [3,5,7,8]
48     where
49         handleErrors x =
50             ...
```

Das heißt die Funktion "testWithData" wird z.B. mit den Parametern zweier Dateinamen und den entsprechenden Labels aufgerufen.

```
65 testWithData :: FilePath -> FilePath -> Label -> Label -> ErrT IO ()
66 testWithData trainingFile1 trainingFile2 label1 label2 =
67     do
68         ...
69         testInput <-
70             readTestInput
71                 trainingFile1 trainingFile2
72                 label1 label2
73             :: ErrT IO (AlgorithmInput, Vector)
74         testGauss label1 label2 testInput >>= \quality ->
75             liftIO $ putStrLn $ concat $ ["gauss quality:", show $ quality]
76         testProjectedGauss label1 label2 testInput >>= \quality ->
77             liftIO $ putStrLn $ concat $ ["projected gauss quality:", show $ quality]
78         testLinearRegression label1 label2 testInput >>= \quality ->
79             liftIO $ putStrLn $ concat $ ["linear regression quality:", show $ quality]
```

Hier werden die geladenen Daten mittels 3-er Algorithmen klassifiziert und deren

Trefferquote gemessen. Außerdem werden im Fall der Fischerdiskrimination als Seiteneffekt errechneten Verteilungen beider Klassen als Diagramm in das Verzeichnis “plots” gespeichert.

1.2.3 Ergebnis

Bei der Ausführung führt das Programm 3 verschiedene Klassifizierungsalgorithmen auf den Daten aus, und berechnet deren Trefferquote:

```
-------------------------------------------
classifying to labels [3,5] in files ["resource/train.3","resource/train.5"]
gauss quality:0.9447852760736196
projected gauss quality:0.9294478527607362
linear regression quality:0.9294478527607362
-------------------------------------------
classifying to labels [3,7] in files ["resource/train.3","resource/train.7"]
gauss quality:0.987220447284345
projected gauss quality:0.9840255591054313
linear regression quality:0.9744408945686901
-------------------------------------------
classifying to labels [3,8] in files ["resource/train.3","resource/train.8"]
gauss quality:0.9367469879518072
projected gauss quality:0.9457831325301205
linear regression quality:0.9518072289156626
-------------------------------------------
classifying to labels [5,7] in files ["resource/train.5","resource/train.7"]
gauss quality:0.9869706840390879
projected gauss quality:0.9837133550488599
linear regression quality:0.9837133550488599
-------------------------------------------
classifying to labels [5,8] in files ["resource/train.5","resource/train.8"]
gauss quality:0.9447852760736196
projected gauss quality:0.9631901840490797
linear regression quality:0.9631901840490797
-------------------------------------------
classifying to labels [7,8] in files ["resource/train.7","resource/train.8"]
gauss quality:0.9712460063897763
projected gauss quality:0.9776357827476039
linear regression quality:0.9776357827476039
```

Offensichtlich sind die Unterschiede von Fall zu Fall unterschiedlich. Keiner der Algorithmen ist in jedem Falle optimal. Die Diagramme mit den Verteilungen nach der Projektion auf den Vektor u der mittels Fischerdiskrimination gefunden wurde sind im Verzeichnis “.plots/” zu finden.

1.3 Kompilieren des Programms

1.3.1 Abhängigkeiten

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

1.3.2 Kompilieren

```
$ git clone https://github.com/EsGeh/pattern-recognition
$ git checkout exercise5-release
$ stack setup
$ stack build
```

1.3.3 Ausführen mittels Stack

```
$ stack exec patternRecogn-exe
```