Mustererkennung WiSe 12/13
Übung 2

Lutz Freitag, Sebastian Kürten

Aufgabe 1 - Visualisierung der Klassenmittelwerte

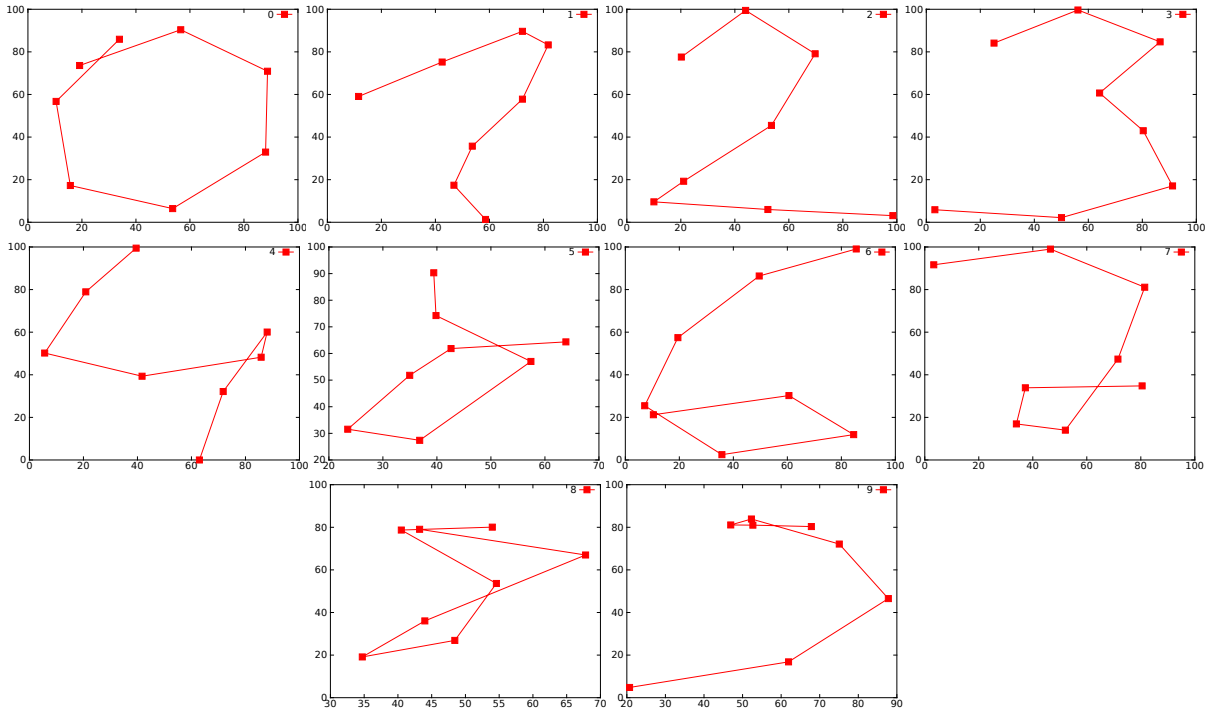

Figure 1: Darstellung der Klassenmittelwerte

Berechnung der Klassenzentren:

```
% lade Daten
trainingData = load("-ascii", "pendigits-training.txt");

mkdir pics

% loop over all digits
for digit = 0:9
        % select all samples labeled with 'digit'
        samples = trainingData(trainingData(:,17) == digit,:);
        % compute average
        digitmean = mean(samples);
        % plot digit
        [coordinates, label] = separateSample(digitmean);
        namePdf = sprintf ("pics/digit-%d.pdf", digit)
        namePng = sprintf ("pics/digit-%d.png", digit)
        plotDigit(coordinates, num2str(label), namePdf, namePng);
end
```

plotDigit.m:

```
function plotDigit(coordinates, label, namePdf, namePng)
        handle = figure('visible', 'off');
```

```
3           [x, y] = splitVector(coordinates);
4           plot(x,y, '--rs', 'markersize', 5, 'linewidth', 2);
5           legend(label);
6           print(namePdf);
7           print(namePng);
8           close(handle);
9   end;
```

Aufgabe 2 - Klassifikation

Die erste Spalte der Confusion-Matrix gibt hier die tatsächliche Klasse an, die erste Zeile benennt die vom Algorithmus ausgewählt Klasse.

NaN	0	1	2	3	4	5	6	7	8	9
0	341	0	0	0	0	0	0	0	22	0
1	0	350	12	0	1	0	0	0	1	0
2	0	8	355	0	0	0	0	1	0	0
3	0	9	0	320	0	1	0	1	0	5
4	0	0	0	0	362	0	0	0	0	2
5	0	0	0	1	0	323	0	0	2	9
6	0	0	0	0	0	0	325	0	11	0
7	0	28	0	0	0	0	0	314	5	17
8	0	0	0	0	0	0	0	0	336	0
9	0	5	0	0	0	0	0	1	1	329

Es wurden 3355 von 3498 Samples richtig klassifiziert, d.h. die Detection-Rate beträgt 95.9%.

```
correct =  3355
wrong =  143
detectionRate =  0.95912
failureRate =  0.040881
```

Folgendes Programm wurde verwendet:

```
1  % load data
2  trainingData = load("-ascii", "pendigits-training.txt");
3  testingData = load("-ascii", "pendigits-testing.txt");
4
5  % number of dimensions
6  n = 16;
7
8  % determine multivariate gaussian distribution for each class
9  covariances = {};
10
11 for digit = 0:9
12          % select all samples labeled with 'digit'
13          samples = trainingData(trainingData(:,17) == digit,:)(:,1:end - 1);
14          % compute average
15          mu = mean(samples);
16
17          % compute covariance matrix
18          cov = zeros(n, n);
19          for sample = samples'
20                  cov += (sample - mu') * (sample' - mu);
21          end
22
23          % normalize
24          cov = cov / size(samples)(1);
```

```
25
26          % add some noise if necessary
27          if det(cov) == 0
28                  cov += rand(size(cov)) / 1000;
29          end
30
31          covariances{digit + 1} = {cov, mu};
32  end
33
34  covariances;
35
36  % classify testing data
37
38  confusionMatrix = zeros(10,10);
39  correct = 0;
40  wrong = 0;
41
42  % iterate all samples in the testing dataset
43  for sample = testingData'
44          [vec, class] = separateSample(sample');
45          % calculate maximum likely class
46          bestProb = 0; % best likelihood seen so far
47          bestClass = 0; % best corresponding class
48          counter = 0; % label of the current class within the loop
49          for cluster = covariances
50                  % calculate likelihood
51                  cov = cluster{1}{1};
52                  mu = cluster{1}{2};
53                  likelihood = 1 / ((2 * pi)^(n/2) * sqrt(det(cov))) * \
54                          e^(-0.5 * (vec - mu) * cov^-1 * (vec - mu)');
55                  % check whether it is better than those before
56                  if (likelihood > bestProb)
57                          bestProb = likelihood;
58                          bestClass = counter;
59                  end
60                  counter += 1;
61          end
62          % add an entry to the confusion matrix
63          confusionMatrix(class + 1, bestClass + 1) += 1;
64          % update correct / wrong counter
65          if bestClass == class
66                  correct += 1;
67          else
68                  wrong += 1;
69          end
70  end
71
72  % print the confusion matrix
73  [NaN 0:9; [0:9]' confusionMatrix]
74
75  % print correctnessRate / failureRate
76  correct
77  wrong
78  detectionRate = correct / (correct + wrong)
79  failureRate = wrong / (correct + wrong)
```