Übung 2 - Lernmethoden für autonome Roboter

0.1 Feature Selection

Die Formel zur Berechnung des Korrelationskoeffizienten für zwei Klassen ist:

$$r_X(f_i, f_j) = \frac{\sum_{k=1}^{n} (X[k,i] - \overline{X[:,i]})(X[k,j] - \overline{X[:,j]})}{\sqrt{\sum_{k=1}^{n} (X[k,i] - \overline{X[:,i]})^2} \sqrt{\sum_{k=1}^{n} (X[k,j] - \overline{X[:,j]})^2}}$$

Aufbauend auf folgender Formel können wir nun die Korrelationskoeffizienten unter den Features und von den Features mit der Klasse berechnen:

Feature\Korrelation mit:	A	B	C	Klasse
A	1	1	-0.762	0.881
B	1	1	-0.762	0.881
C	-0.762	-0.762	1	-0.864

Subset	Average feature correlation	Average class correlation	Merrit
A	1	0.881	0.881
B	1	0.881	0.881
C	1	0.864	0.864
A,B	1	0.881	0.881
A,C	0.762	0.873	0.93
B,C	0.762	0.873	0.93
A,B,C	0.841	0.876	0.926

Laut den ausgerechneten Daten sind die Antworten wie folgt:

a) Der Naive Algorithmus würde zuerst A oder B, und dann das jeweilig andere Feature verwenden, da diese den höchsten Korrelationskoeffizienten mit der Klasse haben. Jedoch wäre dies eine schlechte Wahl, da A und B ziemlich stark miteinander korrelieren, also dem jeweils anderen Feature keine neuen Informationen hinzufügen.

b) Laut dem Merit sind am besten entweder A und C oder B und C. A und B sind vom Informationsgehalt sowieso austauschbar, und C fügt neue Informationen hinzu die den Merit verbessern.

c) Da das hinzufügen von dem Feature das wir bei b) weggelassen haben keine neuen Informationen hinzufügt würde sich der Merit dadurch verschlechtern. Die beste Lösung bei beliebiger Kardinalität wäre also die gleiche wie bei b).

0.1.1 Code und Kommandozeilenausgabe

../selection.py

```
1  import numpy as np
2  from itertools import combinations
3
4  def correlation(a,b):
5      a_mean = np.mean(a)
6      b_mean = np.mean(b)
7      return np.sum((a-a_mean) * (b-b_mean))/(np.sqrt(np.sum(np.power(a-a_mean, 2))) * np.sqrt(
           np.sum(np.power(b-b_mean, 2))))
8
9  def average_correlation(data, features, class_name):
10     if (len(features) > 1):
```

```
11          feature_correlation = 0
12          for a,b in combinations(features, 2):
13              feature_correlation += abs(correlation(data[a],data[b]))
14          feature_correlation /= len(list(combinations(features,2)))
15      else:
16          feature_correlation = 1
17
18      class_correlation = 0
19      for f in features:
20          class_correlation += abs(correlation(data[f], data[class_name]))
21      class_correlation /= len(features)
22      print("\nEvaluationg featureset {2}\nThe average feature feature correlation is {0}.\
            nThe average feature class correlation is: {1}.".format(feature_correlation,
            class_correlation, features))
23      return (feature_correlation, class_correlation)
24
25  def merit(data, features, class_name):
26      count = len(features)
27      res = average_correlation(data, features, class_name)
28      merrit = (count * res[1]) / (np.sqrt(count + count * (count - 1) * res[0]))
29      print("The merit is: {0}".format(merrit))
30      return merrit
31
32  if __name__ == "__main__":
33     data = np.zeros(4, dtype=[('a', 'f8'), ('b', 'f8'), ('c', 'f8'), ('class', 'f8')])
34     data[0] = (-2, -1, 0, -1)
35     data[1] = (-2, -1, 1.4, -1)
36     data[2] = (2, 1, -1, 1)
37     data[3] = (-.2, -.1, -1, 1)
38
39          merit(data, ['a'], 'class')
40          merit(data, ['b'], 'class')
41          merit(data, ['c'], 'class')
42          merit(data, ['a','b'], 'class')
43          merit(data, ['a','c'], 'class')
44          merit(data, ['b','c'], 'class')
45          merit(data, ['a','b','c'], 'class')
```

Kommandozeilenausgabe

```
1  Evaluationg featureset ['a']
2  The average feature feature correlation is 1.
3  The average feature class correlation is: 0.881218831921.
4  The merit is: 0.881218831921
5
6  Evaluationg featureset ['b']
7  The average feature feature correlation is 1.
8  The average feature class correlation is: 0.881218831921.
9  The merit is: 0.881218831921
10
11 Evaluationg featureset ['c']
12 The average feature feature correlation is 1.
13 The average feature class correlation is: 0.864158565218.
14 The merit is: 0.864158565218
15
16 Evaluationg featureset ['a', 'b']
17 The average feature feature correlation is 1.0.
18 The average feature class correlation is: 0.881218831921.
19 The merit is: 0.881218831921
20
21 Evaluationg featureset ['a', 'c']
22 The average feature feature correlation is 0.761512801436.
23 The average feature class correlation is: 0.87268869857.
24 The merit is: 0.929889722672
25
26 Evaluationg featureset ['b', 'c']
27 The average feature feature correlation is 0.761512801436.
28 The average feature class correlation is: 0.87268869857.
29 The merit is: 0.929889722672
30
31 Evaluationg featureset ['a', 'b', 'c']
```

```
32 The average feature feature correlation is 0.841008534291.
33 The average feature class correlation is: 0.875532076353.
34 The merit is: 0.925980669039
```