Exercise 9: Classification

Knowledge Discovery in Databases I
SS 2016
There already exists a very nice solution to exercise 9-1 from the previous year. You can find the slides under the following link (look for exercise 10-3):

http://www.dbs.ifi.lmu.de/Lehre/KDD/SS15/uebung/Tutorial08.pdf
Additional note to clarify some questions which came up in the exercise sessions:

- Bayes rule + Law of total probability:

\[ P(c_j|o) = \frac{P(o|c_j)P(c_j)}{P(o)} = \frac{P(o|c_j)P(c_j)}{\sum_{c_j \in C} P(o|c_j)P(c_j)} \]

- Thus: \( \sum_{c_j \in C} P(c_j|o) = 1 \)

- This also holds under the Naive Bayes assumption

- Note: The Naive Bayes assumption does not state that the attributes are independent, i.e. \( P(o) = \prod_{i=1}^{d} P(o_i) \), but that the attributes are conditionally independent given class \( c_j \), i.e. \( P(o|c_j) = \prod_{i=1}^{d} P(o_i|c_j) \)
The solution to Exercise 9-2 will be provided as a *jupyter* notebook.
Suppose, you have a 2-dimensional dataset consisting of 5 classes with 90 objects each, arranged as follows, and that the classes are linearly separable.
Suppose further, that someone has produced a poor implementation of the m-fold cross validation procedure and applied it in combination with a multiclass linear classifier to obtain the following results:

<table>
<thead>
<tr>
<th>$m$</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>20%</td>
</tr>
<tr>
<td>3</td>
<td>40%</td>
</tr>
<tr>
<td>5</td>
<td>0%</td>
</tr>
<tr>
<td>6</td>
<td>100%</td>
</tr>
<tr>
<td>10</td>
<td>100%</td>
</tr>
</tbody>
</table>
What is the problem with the implementation of the $m$-fold cross validation?

• **Observations:**
  - The classes are linearly separable.
  - If we have enough samples from every class in the training set, we can, in principle, train a multiclass linear classifier with no error. Thus, we could expect (almost) perfect accuracy.
  - On the other hand, if for one class no samples are in the training set, we cannot classify any object of that class correctly.

• **Problem with the implementation:**
  - The folds are constructed by simply cutting the data into consecutive blocks.
  - This is problematic, since the data is sorted, as we will see in the following.
Describe and explain the result for each value of $m$ in short and precise sentences.

- **$m = 2$**:  
  - Suppose, we use the first fold for training  
  - Then, the last two classes are not represented in the training data  
  - Thus, at least $\frac{4}{5}$ of the test samples are misclassified  
  - On the other hand, half of the samples of class $C_3$ are in the training set  
  - If we assume, that all test samples of class $C_3$ are classified correctly, we arrive at the observed accuracy of $\frac{1}{5} = 20\%$  
  - By symmetry: Same results, if we use the second fold for training
Exercise 9-3: m-fold Cross Validation

• $m = 3$:
  • Each fold consists of $\frac{5}{3}$ blocks
  • Suppose, we use the first two folds for training
  • By the same reasoning as for $m = 2$:
    • $\frac{3}{5}$ of the test sample are misclassified
    • $\frac{2}{5} = 40\%$ of the test samples can be classified correctly
  • Again by symmetry, we obtain the same results if we use any of the other folds for testing
Exercise 9-3: m-fold Cross Validation

• $m = 5$:
  • Now each fold corresponds to exactly one class
  • The class that is used for testing is not represented in the training data
  • Thus, all test samples are misclassified and we get an accuracy of 0%

• $m = 6$ and $m = 10$:
  • Now $m$ is large enough, such that a fold can never contain all samples from a certain class
  • Thus, all classes are represented in the training set and we can observe an accuracy of 100%
How could the implementation be improved?

- At least: All classes that appear in the dataset should always be represented in the training data.
- It is further reasonable, to construct training and test sets, such that the class distributions in both sets represent the class distribution in the whole dataset.
- This can be achieved by performing *stratified sampling*:
  - Divide each class ("stratum") separately into \( m \) chunks, either deterministically or by random sampling.
  - Construct a fold for the \( m \)-fold cross-validation by taking a chunk from each class and combining them.