Data Mining

Classification

- Part 3 -
Outline

1. What is Classification?
2. K-Nearest-Neighbors
3. Decision Trees
4. Rule Learning
5. Decision Boundaries
6. Model Evaluation
7. Naïve Bayes
8. Artificial Neural Networks
9. Support Vector Machines
10. Parameter Tuning
20.03.2019 – 16:00-18:00 Uhr
MINT-MARKTPLATZ
Fakultät für Wirtschaftsinformatik & Wirtschaftsmathematik
B6, 30-32, Bauteil E-F (Neubau) im 1.OG

Accenture
AppSphere
BASF
Brandt & Partner
BridgingIT
Camelot
Commerzebank
d-fine
EXA Deutschland
Inter-Versicherung
KPMG
Materna
Mayato
MSSV
MSW & Partner
Porsche
Procter & Gamble
Roche
SAP
Scheer
SAP
Stadt Mannheim
Stocard
Xenium
zeb.
Data Fest 2019 in Mannheim

- Visualize and mine from 3rd to 5th May 2019, Registration: 18th April
- https://hiwissml.github.io/datafest2019/
6. Naïve Bayes

- Probabilistic classification technique based on Bayes theorem.
  - Widely used and especially successful at classifying texts

- Goal: Estimate the most probable class label for a given record.

- Probabilistic formulation of the classification task:
  - consider each attribute and class label as random variables
  - Given a record with attributes \((A_1, A_2, \ldots, A_n)\), the goal is to find the class \(C\) that maximizes the conditional probability

\[
P(C | A_1, A_2, \ldots, A_n)
\]

- Example: Should we play golf?
  - \(P(\text{Play}=\text{yes} | \text{Outlook}=\text{rainy}, \text{Temperature}=\text{cool})\)
  - \(P(\text{Play}=\text{no} | \text{Outlook}=\text{rainy}, \text{Temperature}=\text{cool})\)

- Question: How to estimate these probabilities given training data?
Bayes Theorem

- Thomas Bayes (1701-1761)
  - British mathematician and priest
  - tried to formally prove the existence of God

- Bayes Theorem

\[ P(C|A) = \frac{P(A|C)P(C)}{P(A)} \]

- useful in situations where \( P(C|A) \) is unknown while \( P(A|C), P(A) \) and \( P(C) \) are known or easy to estimate
Bayes Theorem: Evidence Formulation

- **Prior probability** of event $H$:
  - Probability of event before evidence is seen.
  - We play golf in 70% of all cases $\Rightarrow P(H) = 0.7$

- **Posterior probability** of event $H$:
  - Probability of event after evidence is seen.
  - Evidence: It is windy and raining $\Rightarrow P(H \mid E) = 0.2$

- Probability of event $H$ given evidence $E$:

\[
P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}
\]
Applying Bayes Theorem to the Classification Task

1. Compute the probability $P(C \mid A)$ for all values of $C$ using Bayes theorem.
   - $P(A)$ is the same for all classes. Thus, we need to estimate $P(C)$ and $P(A \mid C)$

2. Choose value of $C$ that maximizes $P(C \mid A)$.

Example:

$$P(\text{Play} = \text{yes} \mid \text{Outlook} = \text{rainy}, \text{Temp} = \text{cool}) = \frac{P(\text{Outlook} = \text{rainy}, \text{Temp} = \text{cool} \mid \text{Play} = \text{yes})P(\text{Play} = \text{yes})}{P(\text{Outlook} = \text{rainy}, \text{Temp} = \text{cool})}$$

$$P(\text{Play} = \text{no} \mid \text{Outlook} = \text{rainy}, \text{Temp} = \text{cool}) = \frac{P(\text{Outlook} = \text{rainy}, \text{Temp} = \text{cool} \mid \text{Play} = \text{no})P(\text{Play} = \text{no})}{P(\text{Outlook} = \text{rainy}, \text{Temp} = \text{cool})}$$
The prior probability $P(C_j)$ for each class is estimated by:

1. counting the records in the training set that are labeled with class $C_j$
2. dividing the count by the overall number of records

Example:

- $P(\text{Play}=\text{no}) = 5/14$
- $P(\text{Play}=\text{yes}) = 9/14$
Naïve Bayes assumes that all attributes are statistically independent.

- knowing the value of one attribute says nothing about the value of another
- this independence assumption is almost never correct!
- but … this scheme works well in practice

The independence assumption allows the joint probability $P(A \mid C)$ to be reformulated as the product of the individual probabilities $P(A_i \mid C_j)$:

$$P(A_1, A_2, \ldots, A_n \mid C_j) = P(A_1 \mid C_j) \times P(A_2 \mid C_j) \times \ldots \times P(A_n \mid C_j)$$

$$P(\text{Outlook}=\text{rainy}, \text{Temperature}=\text{cool} \mid \text{Play}=\text{yes}) = P(\text{Outlook}=\text{rainy} \mid \text{Play}=\text{yes}) \times P(\text{Temperature}=\text{cool} \mid \text{Play}=\text{yes})$$

Result: The probabilities $P(A_i \mid C_j)$ for all $A_i$ and $C_j$ can be estimated directly from the training data.
Estimating the Probabilities $P(A_i \mid C_j)$

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Yes 2 No 3</td>
<td>High 3</td>
<td>False 6</td>
<td>No 9</td>
</tr>
<tr>
<td>Overcast</td>
<td>Yes 4 No 0</td>
<td>Normal 6</td>
<td>True 3</td>
<td>No 3</td>
</tr>
<tr>
<td>Rainy</td>
<td>Yes 3 No 2</td>
<td>Cool 3</td>
<td>False 6</td>
<td>No 9</td>
</tr>
<tr>
<td>Sunny</td>
<td>Yes 2/9 No 3/5</td>
<td>High 3/9</td>
<td>True 3/9</td>
<td>Yes 9/14 5/14</td>
</tr>
<tr>
<td>Overcast</td>
<td>Yes 4/9 No 0/5</td>
<td>Normal 6/9</td>
<td>False 6/9 2/5</td>
<td>No 9/14 5/14</td>
</tr>
<tr>
<td>Rainy</td>
<td>Yes 3/9 No 2/5</td>
<td>Cool 3/9</td>
<td>True 3/9</td>
<td>No 3/5</td>
</tr>
</tbody>
</table>

The probabilities $P(A_i \mid C_j)$ are estimated by:
1. counting how often an attribute value appears together with class $C_j$
2. dividing the count by the overall number of records belonging to class $C_j$

Example:
2 times “Yes” together with “Outlook=sunny” out of altogether 9 “Yes” examples
$\Rightarrow p(\text{Outlook}=\text{sunny}|\text{Yes}) = \frac{2}{9}$
Classifying a New Day

Unseen Record

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp.</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Cool</td>
<td>High</td>
<td>True</td>
<td>?</td>
</tr>
</tbody>
</table>

\[
\Pr[yes \mid E] = \Pr[Outlook = Sunny \mid yes] \\
\times \Pr[\text{Temperature} = Cool \mid yes] \\
\times \Pr[\text{Humidity} = High \mid yes] \\
\times \Pr[\text{Windy} = True \mid yes] \\
\times \frac{\Pr[yes]}{\Pr[E]} \\
= \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} \\
= \frac{\Pr[E]}{\Pr[E]}
\]
Classifying a New Day: Weigh the Evidence!

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>2/9</td>
<td>3/9</td>
<td>3/9</td>
<td>9/14</td>
</tr>
<tr>
<td>Overcast</td>
<td>4/9</td>
<td>0/5</td>
<td>4/9</td>
<td>5/14</td>
</tr>
<tr>
<td>Rainy</td>
<td>3/9</td>
<td>2/5</td>
<td>3/9</td>
<td>5/14</td>
</tr>
</tbody>
</table>

- A new day:

Prior probability
Evidence

Choose Maximum

Likelihood of the two classes

For “yes” = $\frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.0053$

For “no” = $\frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{1}{5} \times \frac{5}{14} = 0.0206$

Conversion into a probability by normalization:

$P(\text{“yes”}) = \frac{0.0053}{0.0053 + 0.0206} = 0.205$

$P(\text{“no”}) = \frac{0.0206}{0.0053 + 0.0206} = 0.795$
Handling Numerical Attributes

– Option 1:
  **Discretize** numerical attributes before learning classifier.
  - Temp = 37°C → “Hot”
  - Temp = 21°C → “Mild”

– Option 2:
  Make assumption that numerical attributes have a **normal distribution** given the class.
  - Use training data to estimate parameters of the distribution (e.g., mean and standard deviation)
  - Once the probability distribution is known, it can be used to estimate the conditional probability $P(A_i|C_j)$
Handling Numerical Attributes

- The probability density function for the normal distribution is

\[ f(x) = \frac{1}{\sqrt{2\pi \sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \]

- It is defined by two parameters:
  - **Sample mean** \( \mu \)
    \[ \mu = \frac{1}{n} \sum_{i=1}^{n} x_i \]
  - **Standard deviation** \( \sigma \)
    \[ \sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)^2} \]

- Both parameters can be estimated from the training data.
Statistics for the Weather Data

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>2</td>
<td>3</td>
<td>64, 68,</td>
<td>65, 71,</td>
</tr>
<tr>
<td>Overcast</td>
<td>4</td>
<td>0</td>
<td>69, 70,</td>
<td>72, 80,</td>
</tr>
<tr>
<td>Rainy</td>
<td>3</td>
<td>2</td>
<td>72, ...</td>
<td>85, ...</td>
</tr>
</tbody>
</table>

Sunny: \( \mu = 73 \), \( \sigma = 6.2 \)
Overcast: \( \mu = 75 \), \( \sigma = 7.9 \)
Rainy: \( \mu = 79 \), \( \sigma = 10.2 \)

Example calculation:

\[
f(t_{\text{temp}} = 66 \mid \text{yes}) = \frac{1}{\sqrt{2\pi \sigma^2}} e^{-\frac{(t_{\text{temp}} - \mu)^2}{2\sigma^2}} = 0.0340
\]
Classifying a New Day

Unseen Record

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp.</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>66</td>
<td>90</td>
<td>true</td>
<td>?</td>
</tr>
</tbody>
</table>

Likelihood of “yes” = \( \frac{2}{9} \times 0.0340 \times 0.0221 \times \frac{3}{9} \times \frac{9}{14} = 0.000036 \)
Likelihood of “no” = \( \frac{3}{5} \times 0.0291 \times 0.0380 \times \frac{3}{5} \times \frac{5}{14} = 0.000136 \)
P(“yes”) = \( \frac{0.000036}{0.000036 + 0.000136} = 20.9\% \)
P(“no”) = \( \frac{0.000136}{0.000036 + 0.000136} = 79.1\% \)

But note: Some numeric attributes are not normally distributed and you may thus need to choose a different probability density function or use discretization.
Handling Missing Values

- Missing values may occur in training and classification examples.
- **Training**: Instance is not included in frequency count for attribute value-class combination.
- **Classification**: Attribute will be omitted from calculation.
- **Example**:

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp.</th>
<th>Humidity</th>
<th>windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>Cool</td>
<td>High</td>
<td>True</td>
<td>?</td>
</tr>
</tbody>
</table>

  Likelihood of “yes” = \( \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.0238 \)

  Likelihood of “no” = \( \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.0343 \)

  \[ P(“yes”) = \frac{0.0238}{0.0238 + 0.0343} = 41\% \]

  \[ P(“no”) = \frac{0.0343}{0.0238 + 0.0343} = 59\% \]
The Zero-Frequency Problem

- What if an attribute value doesn’t occur with every class value? (e.g. no “Outlook = overcast” for class “no”)
  - Class-conditional probability will be zero! \( P[\text{Outlook} = \text{overcast} \mid \text{no}] = 0 \)
  - Problem: Posterior probability will also be zero! (No matter how likely the other values are!)
    \( P[\text{no} \mid E] = 0 \)

- Remedy: Add 1 to the count for every attribute value-class combination (Laplace Estimator)

- Result: Probabilities will never be zero! (also: stabilizes probability estimates)

Original: \( P(A_i \mid C) = \frac{N_{ic}}{N_c} \)

Laplace: \( P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + c} \quad c: \text{number of classes} \)
Naïve Bayes in RapidMiner
Naïve Bayes in RapidMiner: The Distribution Table

### Attribute Distribution Table

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Parameter</th>
<th>no</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outlook</td>
<td>value=rain</td>
<td>0.392</td>
<td>0.331</td>
</tr>
<tr>
<td>Outlook</td>
<td>value=overcast</td>
<td>0.014</td>
<td>0.438</td>
</tr>
<tr>
<td>Outlook</td>
<td>value=sunny</td>
<td>0.581</td>
<td>0.223</td>
</tr>
<tr>
<td>Outlook</td>
<td>value=unknown</td>
<td>0.014</td>
<td>0.008</td>
</tr>
<tr>
<td>Temperature</td>
<td>mean</td>
<td>74.600</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>standard deviation</td>
<td>7.893</td>
<td></td>
</tr>
<tr>
<td>Humidity</td>
<td>mean</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td>Humidity</td>
<td>standard deviation</td>
<td>9.618</td>
<td></td>
</tr>
<tr>
<td>Wind</td>
<td>value=true</td>
<td>0.589</td>
<td></td>
</tr>
<tr>
<td>Wind</td>
<td>value=false</td>
<td>0.397</td>
<td></td>
</tr>
<tr>
<td>Wind</td>
<td>value=unknown</td>
<td>0.014</td>
<td></td>
</tr>
</tbody>
</table>

### Graphical Representation

- **Attribute:** Humidity
- **Plot:** Density vs. Humidity
- **Graph:** Two curves, one for `no` and one for `yes`, showing the distribution of humidity values.
Naïve Bayes in RapidMiner: Confidence Scores

<table>
<thead>
<tr>
<th>Row No.</th>
<th>Play</th>
<th>confidence(no)</th>
<th>confidence(yes)</th>
<th>prediction(Play)</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>yes</td>
<td>0.711</td>
<td>0.289</td>
<td>no</td>
<td>sunny</td>
<td>85</td>
<td>85</td>
<td>false</td>
</tr>
<tr>
<td>2</td>
<td>no</td>
<td>0.058</td>
<td>0.942</td>
<td>yes</td>
<td>overcast</td>
<td>80</td>
<td>90</td>
<td>true</td>
</tr>
<tr>
<td>3</td>
<td>yes</td>
<td>0.014</td>
<td>0.986</td>
<td>yes</td>
<td>overcast</td>
<td>83</td>
<td>78</td>
<td>false</td>
</tr>
<tr>
<td>4</td>
<td>yes</td>
<td>0.412</td>
<td>0.588</td>
<td>yes</td>
<td>rain</td>
<td>70</td>
<td>96</td>
<td>false</td>
</tr>
<tr>
<td>5</td>
<td>yes</td>
<td>0.460</td>
<td>0.540</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td>true</td>
</tr>
<tr>
<td>6</td>
<td>no</td>
<td>0.336</td>
<td>0.664</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>true</td>
</tr>
<tr>
<td>7</td>
<td>yes</td>
<td>0.010</td>
<td>0.990</td>
<td>yes</td>
<td>overcast</td>
<td>64</td>
<td>65</td>
<td>true</td>
</tr>
<tr>
<td>8</td>
<td>no</td>
<td>0.596</td>
<td>0.404</td>
<td></td>
<td>sunny</td>
<td>72</td>
<td>95</td>
<td>false</td>
</tr>
<tr>
<td>9</td>
<td>yes</td>
<td>0.248</td>
<td>0.752</td>
<td></td>
<td>sunny</td>
<td>69</td>
<td>70</td>
<td>false</td>
</tr>
<tr>
<td>10</td>
<td>no</td>
<td>0.407</td>
<td>0.593</td>
<td></td>
<td>sunny</td>
<td>75</td>
<td>80</td>
<td>false</td>
</tr>
<tr>
<td>11</td>
<td>yes</td>
<td>0.496</td>
<td>0.504</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>true</td>
</tr>
<tr>
<td>12</td>
<td>yes</td>
<td>0.038</td>
<td>0.962</td>
<td>yes</td>
<td>overcast</td>
<td>81</td>
<td>75</td>
<td>true</td>
</tr>
<tr>
<td>13</td>
<td>no</td>
<td>0.027</td>
<td>0.973</td>
<td>yes</td>
<td>overcast</td>
<td>81</td>
<td>75</td>
<td>true</td>
</tr>
<tr>
<td>14</td>
<td>yes</td>
<td>0.453</td>
<td>0.547</td>
<td></td>
<td>rain</td>
<td>71</td>
<td>80</td>
<td>true</td>
</tr>
</tbody>
</table>

Classifier is quite sure

Classifier is not sure
Characteristics of Naïve Bayes

- Naïve Bayes works surprisingly well for many classification tasks.
  - even if independence assumption is clearly violated
  - Why? Because classification doesn’t require accurate probability estimates as long as maximum probability is assigned to correct class

- Robust to isolated noise points as they will be averaged out
- Robust to irrelevant attributes as \( P(A_i | C) \) distributed uniformly for \( A_i \)
- Adding too many redundant attributes can cause problems.
  - Solution: Select attribute subset as Naïve Bayes often works better with just a fraction of all attributes.

- Technical advantages
  - Learning Naïve Bayes classifiers is computationally cheap as probabilities can be estimated doing one pass over the training data.
  - Storing the probabilities does not require a lot on memory.
Further Classification Methods

- There are various methods of classification
  - e.g., RapidMiner implements 53 different methods

- So far, we have seen
  1. k-NN
  2. Decision Trees
  3. C4.5 and Ripper
  4. Naive Bayes

- Now: Brief introduction to
  1. Artificial Neural Networks
  2. Support Vector Machines
7. Artificial Neural Networks (ANN)

- Inspiration
  - one of the most powerful super computers in the world
Example:
Output Y is 1 if at least two of the three inputs are equal to 1.
Artificial Neural Networks (ANN)

Training Data

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$Y = I (0.3 X_1 + 0.3 X_2 + 0.3 X_3 - 0.4 > 0)$

where $I(z) = \begin{cases} 
1 & \text{if } z \text{ is true} \\
0 & \text{otherwise} 
\end{cases}$
Artificial Neural Networks (ANN)

- Model is an assembly of inter-connected nodes (called neurons) and weighted links.

- Output node sums up each of its input values according to the weights of its links.

- Classification decision: Compare output node against some threshold $t$.

Perceptron Model

$$Y = I\left(\sum_i w_i X_i - t > 0\right) \quad \text{or} \quad Y = \text{sign}\left(\sum_i w_i X_i - t\right)$$
Multi-Layer Artificial Neural Networks

Training ANN means learning the weights of the neurons.
Algorithm for Training ANNs

1. Initialize the weights \((w_0, w_1, \ldots, w_k)\), e.g., all with 1 or random

2. Adjust the weights in such a way that the output of ANN is as consistent as possible with class labels of the training examples

   • Objective function:
     
     \[
     E = \sum_{i} [Y_i - f(w_i, X_i)]^2
     \]

   • Find the weights \(w_i\)'s that minimize the error \(E\)

   • using for example the \textbf{back propagation algorithm}
     (see Tan Steinbach, Chapter 5.4)
Deep Neural Networks (DNN)

- Hype topic as Google successfully uses DNN for
  - computer vision
  - speech recognition
  - NLP

- Require
  - lots of training data
  - GPUs to calculate weights

- Details
  - Data Mining II
  - Lecture on 18th of March

Source: NVIDIA
Artificial Neural Networks in RapidMiner

RapidMiner offers three alternatives:

1. RapidMiner‘s own **Neural Net** operator
2. wrapper for **H2O** Deep Learning
3. wrapper for **Keras** (Python) as separate extension
Characteristics of Artificial Neural Networks

- ANNs can be used for classification as well as numerical regression tasks (more on this next week)
- Multi-layer neural networks are universal approximators
  - meaning that they can approximate any target function
- Very important but difficult to choose the right network topology
  - Expressive hypothesis space often leads to overfitting
  - Possible approaches to deal with overfitting:
    - Use more training data (a lot more might be necessary)
    - Step-by-step simplify the topology (also called regularization)
      1. Start with several hidden layers and larger number of nodes
      2. Estimate generalization error using validation dataset
      3. Step by step remove nodes as long as generalization error improves
- Model building can be time consuming, model application is fast
8. Support Vector Machines

- Support vector machines (SVMs) are algorithms for learning linear classifiers for
  • two class problems (a positive and a negative class)
  • from examples described by continuous attributes.

- SVMs achieve very good results especially for high dimensional data.

- SVMs were invented by V. Vapnik and his co-workers in 1970s in Russia and became known to the West in 1992.
Support Vector Machines

- SVMs find a linear hyperplane (decision boundary) that will separate the data.
Which one is better? B1 or B2?

How do you define “better”?
Which Hyperplane is better?

- In order to avoid overfitting and to generalize for unseen data, SVMs find the hyperplane that maximizes the margin to the closest points (support vectors).

- Visual solution:
  - B1 is better than B2

- Mathematical solution:
  - Constrained optimization that can be solved using quadratic programming.
  - See Tan/Steinbach/Kumar, Chapter 5.5
Dealing with Not Linearly Separable Data

- What if the problem is not linearly separable due to noise points?

- Solution: Introduce slack variables in margin computation which result in a penalty for each data point that violates decision boundary.
Dealing with Non-Linear Decision Boundaries

- Problem: What if decision boundary is not linear?
- Solution: Transform data into higher dimensional space where there is a linear separation
  - Details: Higher mathematics (see Tan/Steinbach, Chapter 5.5)
  - Different types of kernel functions are used for this transformation
Characteristics of Support Vector Machines

- SVMs were often the most successful classification technique for high dimensional data before DNNs appeared.
- Application areas of SVMs include
  - Text classification
  - Machine vision, e.g., face identification
  - Handwritten digit recognition
  - SPAM detection
  - Bioinformatics
- Parameter tuning often has a high impact on the performance of SVN.
  - see next slide
SVMs in RapidMiner

Tuning a SVM

1. Transform all attributes to numeric scale (Operator: Nominal to Numeric)
2. Normalize all value ranges to [0,1] (Operator: Normalize)
3. Use the RBF kernel function
4. Use nested cross-validation to find the best values for the parameters
   1. C = weight of slack variables (Range: 0.03 to 30000)
   2. gamma = kernel parameter (Range: 0.00003 to 8)

9. Parameter Tuning

- Many learning methods require parameters
  - k for k-nearest-neighbors
  - pruning thresholds for trees and rules
  - hidden layers configuration for ANN
  - gamma and C for SVM

- Some methods often work rather poorly with default parameters

- How to determine the optimal parameters?
  - Play around with different parameters yourself
  - Alternative: Let your data mining tool test different parameter settings for you
Parameter Optimization

The **Optimize Parameters** operator allows you to automatically test various parameter combinations.
Parameter Optimization

Alternative optimization approaches that are more efficient than brute force grid search are beam search and evolutionary algorithms.
Parameter Optimization and Generalization Error

- keeping training and test set **strictly separate** is crucial in order to not estimate the generalization error too optimistic
- **Wrong**: Parameter optimization using inner cross-validation
  - parameter value “leaks” information about test set into the model
  - model overfits training and test data $\Rightarrow$ higher error on unseen data
- **Right**: **Nested Cross-Validation**
  - Outer Cross-Validation estimates generalization error using test set
  - Inner Cross-Validation
    - estimates optimal parameter setting using cross-validation to split training set from outer cross-validation into training and validation set.
    - Once the optimal parameters are found, a model is learned with these parameters using the complete outer training data.
    - Result: Good estimate of generalization error as no information from test set is leaked into parameter setting.

https://rapidminer.com/resource/correct-model-validation/
Nested Cross-Validation for Parameter Optimization

- **Outer Cross Validation**
- **Optimize Parameters**
- **Inner Cross Validation**
What If Nested Cross-Validation Gets Too Slow?

- **Nested Cross-Validation**
  - Outer X-Val: 3 times
  - X-Val parameters: $3 \times 30$ times
  - Learn using best parameters: 1
  - $3 \times (3 \times 30 + 1) = \textbf{273 models learned}$

- **X-Val with Inner Hold-Out Validation**
  - Outer X-Val: 3 times
  - Parameter optimization: 30 times
  - Learn using best parameters: 1
  - $3 \times (30 + 1) = \textbf{93 models learned}$

- **Parameter Optimization using Hold-Out Validation**
  - Parameter Search: 30
  - Learning using parameters: 1
  - $30 + 1 = \textbf{31 models learned}$
Attribute Subset Selection

The **Optimize Selection** operator allows you to find the optimal subset of the available attributes.

**Forward selection:** Find best single attribute, add further attributes, test again

**Backward selection:** Start with all attribute, remove attributes, test again

Chapter 5.3: Naïve Bayes  
Chapter 5.4: Artificial Neural Networks  
Chapter 5.5: Support Vector Machines

Videos

- Parameter Optimization  
  http://www.youtube.com/watch?v=R5vPrTLMzng
- Attribute Subset Selection  
  - Part 1: http://www.youtube.com/watch?v=7IC3lQEduxA  
  - Part 2: http://www.youtube.com/watch?v=j5vhwbLIZWg