Data Mining und Maschinelles Lernen

Data Pre-Processing

- Data Mining
  - Motivation
  - Data Mining Process Models
- Pre-Processing
  - Supervised vs. Unsupervised

- Feature Subset Selection
  - Filter and Wrapper Approaches
- Discretization
  - Bottom-Up (Chi-Merge) and Top-Down (Entropy-Split)
- Sampling
  - Windowing
- Data Cleaning
  - Outlier Detection and Noise Filtering
Key steps in the **Knowledge Discovery cycle**:

1. **Data Cleaning**: remove noise and inconsistent data
2. **Data Integration**: combine multiple data sources
3. **Data Selection**: select the part of the data that are relevant for the problem
4. **Data Transformation**: transform the data into a suitable format (e.g., a single table, by summary or aggregation operations)
5. **Data Mining**: apply machine learning and machine discovery techniques
6. **Pattern Evaluation**: evaluate whether the found patterns meet the requirements (e.g., interestingness)
7. **Knowledge Presentation**: present the mined knowledge to the user (e.g., visualization)
Data Mining is a Process!

The steps are not followed linearly, but in an iterative process.

Another Process Model

Source: http://www.crisp-dm.org/
Pre-Processing

- Databases are typically not made to support analysis with a data mining algorithm
  - pre-processing of data is necessary

- Pre-processing techniques:
  - **Feature Engineering:**
    - find the right features/attribute set
    - *Feature Subset Selection:* select appropriate feature subsets
    - *Feature Transformation:* bring attributes into a suitable form (e.g., discretization)
    - *Feature Construction:* construct derived features
  - **Data Cleaning:**
    - remove inconsistencies from the data
  - **Sampling:**
    - select appropriate subsets of the data
Unsupervised vs. Supervised
Pre-processing

- Unsupervised
  - do not use information about the learning task
    - only prior information (from knowledge about the data)
    - and information about the distribution of the training data

- Supervised
  - use information about the learning task
    - e.g.: look at relation of an attribute to class attribute

- WARNING:
  - pre-processing may only use information from training data!
    - compute pre-processing model from training data
    - apply the model to training and test data
    - otherwise information from test data may be captured in the pre-processing step → biased evaluation
  - in particular: apply pre-processing to every fold in cross-validation
Feature Subset Selection

- Databases are typically not collected with data mining in mind
- Many features may be
  - irrelevant
  - uninteresting
  - redundant
- Removing them can
  - increase efficiency
  - improve accuracy
  - prevent overfitting
- Feature Subset Selection techniques try to determine appropriate features automatically
Unsupervised FSS

- Using domain knowledge
  - some features may be known to be irrelevant, uninteresting or redundant
- Random Sampling
  - select a random sample of the feature
  - may be appropriate in the case of many weakly relevant features and/or in connection with ensemble methods
Supervised FSS

- Filter approaches:
  - compute some measure for estimating the ability to discriminate between classes
  - typically measure feature weight and select the best n features
  - problems
    - redundant features (correlated features will all have similar weights)
    - dependent features (some features may only be important in combination (e.g., XOR/parity problems).
Supervised FSS: Filters

- **Feature Weighting**
  - a good attribute should discriminate between classes
  - use a measure of discrimination for determining the importance of attributes
    - decision tree splitting criteria (entropy/information gain, gini-index, …)
    - attribute weighting criteria (Relief, …), etc.

- **Advantage**
  - very fast

- **Disadvantage**
  - quality of each attribute is measured in isolation
  - some attributes may only be useful in combination with others

\[
\text{foreach attribute } A \\
\text{ } \quad W[A] = \text{feature weight according to some measure of discrimination} \\
\text{select the } n \text{ features with highest } W[A]
\]
Supervised FSS

- **Filter approaches:**
  - compute some measure for estimating the ability to discriminate between classes
  - typically measure feature weight and select the best n features
  - problems
    - redundant features (correlated features will all have similar weights)
    - dependent features (some features may only be important in combination (e.g., XOR/parity problems).

- **Wrapper approaches**
  - search through the space of all possible feature subsets
  - each search subset is tried with the learning algorithm
FSS: Wrapper Approach
(John, Kohavi, Pfleger, ICML-94)

- Wrapper Approach:
  - try a feature subset with the learner
  - improve it by modifying the feature sets based on the result
  - repeat

The induction algorithm itself is used as a “black box” by the subset selection algorithm.
FSS: Wrapper Approach

- **Forward selection:**
  1. start with empty feature set \( F \)
  2. for each attribute \( A \)
     - Estimate Accuracy of Learning algorithm on \( F \cup \{A\} \)
  3. \( F = F \cup \{\text{attribute with highest estimated accuracy}\} \)
  4. goto 2. until \( n \) features have been found

- **Backward elimination:**
  - start with full feature set \( F \)
  - try to remove attributes

- Bi-directional search is also possible
Example: Forward Search for Best 3 Features

- **Attrs**: current set of attributes
- **Est**: accuracy estimated by wrapper
- **Real**: "real" accuracy
Stopping Criteria for Wrapper algorithms

- Select the best \( n \) attributes
  - Like pseudo-code on the previous slide
- Add an attribute if it increases accuracy
  - Might be too greedy
  - e.g., in the previous example, the search would have stopped after adding the first attribute
- Add an attribute until the last \( k \) added attributes did not increase attribute
  - e.g., for \( k = 2 \), the last example would have found the final 3-value set
- Add an attribute if it does not significantly decrease accuracy
  - Significance test can be performed with → sign test or → t-test
Wrapper Approaches - Discussion

- Advantage:
  - find feature set that is tailored to learning algorithm
  - considers combinations of features, not only individual feature weights
  - can eliminate redundant features
    (picks only as many as the algorithm needs)

- Disadvantage:
  - very inefficient: many learning cycles necessary
Comparison Wrapper / Filter (Relief)

**Note:** RelieveD is a version of Relief that uses all examples instead of a random sample.

- on these datasets:
  - forward selection reduces attributes w/o error increase
  - in general, it may also reduce error
Feature Transformation

- **numerization**
  - some algorithms can **only use numeric data**
  - nominal → binary
    - a nominal attribute with n values is converted into n binary attributes
  - binary → numeric
    - binary features may be viewed as special cases of numeric attributes with two values

- **standardization**
  - **normalize** numerical attributes to useful ranges
  - sometimes logarithmic transformations are necessary

- **discretization**
  - some algorithms can **only use categorical data**
    - transform numeric attributes into (ordered) categorical values
Discretization

- **Supervised vs. Unsupervised:**
  - **Unsupervised:**
    - only look at the distribution of values of the attribute
  - **Supervised:**
    - also consider the relation of attribute values to class values

- **Merging vs. Splitting:**
  - **Merging** (bottom-up discretization):
    - Start with a set of intervals (e.g., each point is an interval) and successively combine neighboring intervals
  - **Splitting** (top-down discretization):
    - Start with a single interval and successively split the interval into sub-intervals
Unsupervised Discretization

- domain-dependent:
  - suitable discretizations are often known
  - age (0-18) →
    baby (0-3), child (3-6), school child (6-10), teenager (11-18)

- equal-width:
  - divide value range into a number of intervals with equal width
  - age (0-18) → (0-3, 4-7, 8-11, 12-15, 16-18)

- equal-frequency:
  - divide value range into a number of intervals so that (approximately) the same number of datapoints are in each interval
  - e.g., N = 5: each interval will contain 20% of the training data
  - good for non-uniform distributions (e.g., salary)
Supervised Discretization: Chi-Merge (Kerber, AAAI-92)

**Basic Idea:** merge neighboring intervals if the class information is independent of the interval an example belongs to

- **Initialization:**
  - sort examples according to feature value
  - construct one interval for each value

- **Interval merging:**
  - compute $\chi^2$ value for each pair of adjacent intervals
    
    $\chi^2 = \sum_{i=1}^{2} \sum_{j=1}^{c} \frac{(A_{ij} - E_{ij})^2}{E_{ij}}$

    where
    
    $E_{ij} = N_i \frac{C_j}{N_1 + N_2}$

    $C_j = A_{1j} + A_{2j}$

    $A_{ij} =$ number of examples in $i$-th interval that are of class $j$

    $E_{ij} =$ expected number of examples in $i$-th interval that are of class $j$

    $= \text{examples in } i\text{-th interval } N_i \times \text{fraction of examples of class } j \text{ in both intervals}$

  - merge those with lowest $\chi^2$ value

- **Stop**
  - when the $\chi^2$ values of all pairs exceed a significance threshold
Supervised Discretization: Entropy-Split (Fayyad & Irani, IJCAI-93)

**Basic Idea:** grow a decision tree using a single numeric attribute and use the value ranges in the leaves as ordinal values

- **Initialization:**
  - initialize intervals with a single interval covering all examples $S$
  - sort all examples according to the attribute value
  - initialize the set of possible split points
    - simple: all values

- **Interval splitting:**
  - select split point with the minimum weighted entropy
    \[ T_{\text{max}} = \arg \min_T \left( \frac{|S_{A<T}|}{|S|} \text{Entropy}(S_{A<T}) + \frac{|S_{A\geq T}|}{|S|} \text{Entropy}(S_{A\geq T}) \right) \]
  - recursively apply Entropy-Split to $S_{A<T_{\text{max}}}$ and $S_{A \geq T_{\text{max}}}$

- **Stop**
  - when a given number of splits is achieved
  - or when splitting would yield too small intervals
  - or MDL-based stopping criterion (Fayyad & Irani, 1993)
Example

<table>
<thead>
<tr>
<th>Temperature</th>
<th>64</th>
<th>65</th>
<th>68</th>
<th>69</th>
<th>70</th>
<th>71</th>
<th>72</th>
<th>72</th>
<th>75</th>
<th>75</th>
<th>80</th>
<th>81</th>
<th>83</th>
<th>85</th>
</tr>
</thead>
<tbody>
<tr>
<td>Play</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Slide taken from Witten & Frank
Example

- Possible Split points:
  64.5, 66.5, 68.5, 69.5, 70.5, 71.5, 73.5, 77.5, 80.5, 82.0, 84.0

- Compute Information gain for every split point
  ▪ As in decision tree induction for numeric attributes

- Select the point with the highest information gain
  ▪ In this case 84.0 (→ point A in graph in previous slide)

- Repeat in both successor nodes until a full decision tree is grown
  ▪ In the example only the left branch contains examples

Note:

- One can proof that a split point can only lie on a change between classes, i.e., we would only have to consider split points
  64.5, 66.5, 70.5, 71.5, 73.5, 77.5, 80.5, 84.0
  (we cannot split the yes/no examples at 72.0, so we have to split left and right of it)
Resulting Tree

- Leaf nodes of the resulting tree correspond to intervals.
- Generate one discrete value for each interval.
  - In this example, we get a nominal attribute with 7 values.

Note:
- The tree structure does not always degenerate to a list.
- But there is a selection bias towards split points near the end of the value ranges.
Unsupervised Feature Construction

- based on domain knowledge
  - Example: Body Mass Index

- automatic
  - Examples:
    - kernel functions
      - may be viewed as feature construction modules
      - \( \rightarrow \) support vector machines
    - principal components analysis
      - transforms an n-dimensional space into a lower-dimensional subspace w/o losing much information
    - GLEM:
      - uses an Apriori-like algorithms to compute all conjunctive combinations of basic features that occur at least n times
      - application to constructing evaluation functions for game Othello

\[
BMU = \frac{\text{weight (kg)}}{\text{height (m)}^2}
\]
Supervised Feature Construction

- use the class information to construct features that help to solve the classification problem

Examples:
- Wrapper approach
  - use rule or decision tree learning algorithm
  - observe frequently co-occurring features or feature values
  - encode them as separate features
- Neural Network
  - nodes in hidden layers may be interpreted as constructed features
Scalability

- databases are often too big for machine learning algorithms
  - ML algorithms require frequent counting operations and multi-dimensional access to data
  - only feasible for data that can be held in main memory

- two strategies to make DM algorithms scalable
  - design algorithms that are explicitly targetted towards minimizing the number of database operations (e.g., Apriori)
  - use sampling to work on subsets of the data
Windowing

- **Idea:**
  - focus the learner on the parts of the search space that are not yet correctly covered

- **Algorithm:**
  1. Initialize the window with a random subsample of the available data
  2. Learn a theory from the current window
  3. If the learned theory correctly classifies all examples (including those outside of the window), return the theory
  4. Add some mis-classified examples to the window and goto 2.

- **Properties:**
  - may learn a good theory from a subset of the data
  - problems with noisy data
Outlier Detection

unsupervised Data Cleaning method

- **Goal:**
  - detect examples which deviate a lot from other examples
  - they are probably due to measurement errors

- **2-Sigma Rule:**
  - common statistical Method for outlier detection
  - An example is classified as an outlier if
    - there exists one (numerical) attribute A
    - whose value deviates from the mean by more than two standard deviations

\[|x_A - \bar{x}_A| > 2 \cdot \sigma_A\]
Identifying Mislabeled Examples
(Friedl & Brodley, 1999)

- Identify noisy examples
  - correct them or remove them from the database
  - train the classifier on a corrected database

\[\text{Training Instances} \xrightarrow{\text{Filter}} \text{“Correctly Labeled” Training Instances} \xrightarrow{\text{Learning Algorithm}} \text{Classifier}\]
Robust Decision Trees
(John, KDD-95)

- supervised data cleaning method

1. train a decision tree T
2. remove all training examples that are misclassified by T
3. learn a new tree from the remaining examples
4. repeat until convergence

- thus the final tree is trained on a subset of original data
  - but may not only be simpler but also more accurate
  - may be viewed as an inverse windowing
Ensemble Filters

- Generalization of the previous approach to ensembles
  - filter an example if $\geq c\%$ of the base classifiers misclassify it

- Majority Filter
  - filter if more than half of the classifiers mislabel the example

- Consensus Filter
  - special case where only unanimous misclassifications count
Experimental Comparison  
(Friedl & Brodley, 1999)

Typical results:

- majority performs best
- consensus is too conversative
  - not enough examples removed
- single algorithm filter (≈ robust decision trees) is too loose
  - too many examples removed