Data Mining
and
Concept Learning

Fall 2012

Stan Matwin
stan@site.uottawa.ca
Distinguished University Professor
Electrical Engineering and Computer Science
University of Ottawa
Canada
Goals of this course

• to convey and teach basic concepts of Machine Learning (ML) and Data Mining (DM)
• to outline interesting areas of current and future research
• to provide hands-on DM experience to students
• to present successful applications
• to suggest method selection principles for a given application
• to enable advanced self-study on ML/DM
Course outline


2. Symbolic learning: Decision Trees; learning as search; basic Evaluation of learning/mining results;


5. Bayesian methods for learning from text data

7. Ensembles of learners: boosting, bagging, random forest

8. Advance evaluation measures – ROC, AUC

9. Pre- and post-processing: Feature selection; combining classifiers; discretization

10. Distance-based classification. Clustering

11. Data mining concepts and techniques

12. Semi-supervised learning: co-training

13. Data mining and privacy

NOTE: if time allows, we will cover advanced methods for sequential data (HMM, Conditional Random Fields, Gibbs sampling)
1. Machine Learning / Data Mining: basic terminology

• Machine Learning:
  – given a certain task, and a data set that constitutes the task,
  – ML provides algorithms that resolve the task based on the data, and the solution improves with time

• Examples:
  – predicting lottery numbers next Saturday
  – Detecting oil spills on sea surface
  – Assisting Systematic Reviews
  – Profiling
    » Dreams
    » Digital game players
    » Fraudulent CC use
• Data Mining: extracting regularities from a VERY LARGE dataset/database as part of a business/application cycle

• examples:
  – customer churn profiling
  – direct mail targeting/ cross sell
  – security applications: monitoring of
    » Social networks
    » Computer networks
  – prediction of aircraft component failures
  – clustering of genes wrt their behavior
Basic ML tasks

• Supervised learning
  – classification/concept learning
  – estimation: essentially, extrapolation

• Unsupervised learning:
  – clustering: finding groups of “similar” objects
  – associations: in a database, finding that some values of attributes go with some other
Concept learning (also known as classification): a definition

• Data are given as vectors of attribute values, where the domain of possible values for attribute $j$ is denoted as $A_j$, for $1 \leq j \leq N$. Moreover, a set $C = \{c_1, \ldots, c_k\}$ of $k$ classes is given; this can be seen as a special attribute or label for each record. Often $k = 2$, in which case we are learning a binary classifier.

• Inducing, or learning a classifier, means finding a mapping

$$F: A_1 \times A_2 \times A_N \to C,$$

given a finite training set $X_1 = \{<x_{ij}, c_i>, 1 \leq j \leq N, c_i \in C, 1 \leq i \leq M\}$ of $M$ labeled examples [comment on noise]
• We assume that data is represented as fixed size vectors of attributes (AVL representation): eg all patients are represented by the same 38 attributes, perhaps in conceptual groupings into personal, social, medical

• F belongs to a fixed language, e.g. F can be
  – a set of \( n-1 \) dimensional hyperplanes partitioning an \( n \)-dimensional space into \( k \) subspaces, or
  – a decision tree with leaves belonging to \( C \), or
  – a set of rules with consequents in \( C \).

• We also want \( F \) to perform well, in terms of its predictive power on (future) data not belonging to \( X_1 \)
– In data base terminology, we “model” one relation
– There are methods that deal with multi-relational representations (multiple tables),
  - multi-relational learning AKA Inductive Logic Programming
Data Mining Process

• An iterative process which includes the following steps
  – Formulate the problem e.g. Classification/Numeric Prediction
  – Collect the relevant data (No data, no model)
  – Represent the Data in the form of labeled examples (a.k.a instances) to be learned from
  – Learn a model/predictor
  – Evaluate the model
  – Fine tune the model as needed
Tests: T1, …, T5
Test Outcomes: O_{11}, …, O_{52}
Predictions: C1, …, C3
Example 1: Are We Going to Play Outdoors

- Predict whether there will be an outdoor game depending on the existing weather conditions
- Classification: yes (play), no (don’t play)
- Data: A collection of past observation about the days that there was or was not an outdoor game. The data is already collected and labeled.
Representation for Outdoor Game Prediction

- Each example records the following information (a.k.a features or attributes)
  - outlook \{sunny, overcast, rainy\}
  - temperature real
  - humidity real
  - windy \{TRUE, FALSE\}
  - play \{yes, no\} -> the label
Outdoor Game Play Prediction
Data for WEKA

% ARFF file for the weather data with some numeric features
%
@relation weather

@attribute outlook { sunny, overcast, rainy }
@attribute temperature numeric
@attribute humidity numeric
@attribute windy { true, false }
@attribute play? { yes, no }

@data
%
% 14 instances
%
sunny, 8, 85, false, no
sunny, 8, 90, true, no
overcast, 83, 86, false, yes
rainy, 80, 96, false, yes
rainy, 6, 80, false, yes
rainy, 6, 70, true, no
overcast, 64, 65, true, yes
sunny, 95, false, no
sunny, 70, false, yes
rainy, 5, 80, false, yes
sunny, 5, 70, true, yes
overcast, 72, 90, true, yes
overcast, 81, 75, false, yes
rainy, 7, 91, true, no
Outdoor Game Play Decision Tree

- **outlook**
  - **sunny**
    - **humidity**
      - ≤ 75: yes
      - > 75: no
  - **overcast**: yes
- **rainy**
  - **windy**
    - True: no
    - False: yes
Outdoor Game Play Model (As output by WEKA)

J48 pruned tree

outlook = sunny
|   humidity <= 75: yes (2.0)
|   humidity > 75: no (3.0)
outlook = overcast: yes (4.0)
outlook = rainy
|   windy = TRUE: no (2.0)
|   windy = FALSE: yes (3.0)

Number of Leaves :  5
Size of the tree :  8
### Confusion Matrix

Correctly Classified Instances  9  64.2857 %
Incorrectly Classified Instances  5  35.7143 %

```markdown
<table>
<thead>
<tr>
<th></th>
<th>a = yes</th>
<th>b = no</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
```

Correctly classified entries (row&column): aa, bb
Incorrectly Classified entries (row&column): ab, ba
False Positive/Days incorrectly classified as play: ba
False Negative/Days incorrectly classified as no play: ab
Example 2: Who would survive Titanic’s sinking

• Predict whether a person on board would have survived the tragic sinking
• Classification: yes (survives), no (does not survive)
• Data: The data is already collected and labeled for all 2201 people on board the Titanic.
Example 2: Representation for the Titanic Survivor Prediction

- Each example records the following attributes
  - social class \{first class, second class, third class, crew member\}
  - age \{adult, child\}
  - sex \{male, female\}
  - survived \{yes, no\}
Titanic Survivor model

J48 pruned tree

sex = male
|   social_class = first
|   |   age = adult: no (175.0/57.0)
|   |   age = child: yes (5.0)
|   social_class = second
|   |   age = adult: no (168.0/14.0)
|   |   age = child: yes (11.0)
|   social_class = third: no (510.0/88.0)
|   social_class = crew: no (862.0/192.0)

sex = female
|   social_class = first: yes (145.0/4.0)
|   social_class = second: yes (106.0/13.0)
|   social_class = third: no (196.0/90.0)
|   social_class = crew: yes (23.0/3.0)

Number of Leaves : 10
Size of the tree : 15

Correctly Classified Instances
1737  78.9187 %

Incorrectly Classified Instances
464   21.0813 %

=== Confusion Matrix ===

a    b   <-- classified as
267  444 |    a = yes
20 1470 |    b = no
Induction of decision trees: an algorithm building a DT from data…

building a *univariate (single attribute is tested)* decision tree from a set $T$ of training cases for a concept $C$ with classes $C_1, \ldots, C_k$

Consider three possibilities:

• $T$ contains 1 or more cases all belonging to the same class $C_j$. The decision tree for $T$ is a leaf identifying class $C_j$

• $T$ contains no cases. The tree is a leaf, but the label is assigned heuristically, e.g. the majority class in the parent of this node
• T contains cases from different classes. T is divided into subsets that seem to lead towards collections of cases. A test t based on a single attribute is chosen, and it partitions T into subsets \(\{T_1,\ldots,T_n\}\). The decision tree consists of a decision node identifying the tested attribute, and one branch for each outcome of the test. Then, the same process is applied recursively to each sub-set. \(T_i\)
Choosing the test

• why not explore all possible trees and choose the simplest (Occam’s razor)? But this is an NP complete problem. E.g. in the ‘Titanic’ example there are millions of trees consistent with the data
• idea: to choose an attribute that best separates the examples according to their class label
• This means to maximize the difference between the info needed to identify a class of an example in T, and the same info after T has been partitioned in accordance with a test X
• Entropy is a measure from information theory [Shannon] that measures the quantity of information
• information measure (in bits) of a message is $- \log_2$ of the probability of that message

• notation: $S$: set of the training examples; $\text{freq}(C_i, S) = \text{number of examples in } S \text{ that belong to } C_i$;
selecting 1 case and announcing its class has info measure - \( \log_2(f_{\text{freq}(C_i, S)/|S|}) \) bits

to find information pertaining to class membership in all classes: \( \text{info}(S) = -\sum_{i} (f_{\text{freq}(C_i, S)/|S|}) \times \log_2(f_{\text{freq}(C_i, S)/|S|}) \)

after partitioning according to outcome of test \( X \):

\( \text{info}_X(T) = \sum_{i} |T_i|/|T| \times \text{info}(T_i) \)

\( \text{gain}(X) = \text{info}(T) - \text{info}_X(T) \) measures the gain from partitioning \( T \) according to \( X \)

We select \( X \) to maximize this gain
Other splitting criteria

Note that the information gain formula is a special case of a more general criterion:

\[ I(s) = P(L)f(P(\text{+}|L_s), P(-|L_s)) + P(R)f(P(\text{+}|R_s), P(-|R_s)) \]

where \( P(L), P(R) \) are probabilities of an example going left or right, and assuming two classes \(-\) and \(+\)

And \( f(a, b) \) is the impurity function

For \( f(a, b) = a\log_2(a) + b\log_2(b) \) we have info gain, for \( f(a, b) = 2ab \) we have a so called Gini criterion. For \( f(a, b) = \sqrt{ab} \) we have a criterion known as DKM. It is known as the most stable of the three.
Data for learning the weather (play/don’t play) concept (Witten p. 10)

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temp</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Ovcest</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Rain</td>
<td>Col</td>
<td>Normal</td>
<td>Weak</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Ovcest</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Ovcest</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>Ovcest</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

\[ \text{Info}(S) = -(9/14)\log_2(9/14)-(5/14)\log_2 (5/14) = 0.940 \]
Selecting the attribute

- \( \text{Gain}(S, \text{Outlook}) = 0.246 \)
- \( \text{Gain}(S, \text{Humidity}) = 0.151 \)
- \( \text{Gain}(S, \text{Wind}) = 0.048 \)
- \( \text{Gain}(S, \text{Temp}) = 0.029 \)

- Choose Outlook as the top test
How does info gain work?

Which attribute should be tested here?

$S_{sunny} = \{D_1, D_2, D_8, D_9, D_{11}\}$

$Gain(S_{sunny}, \text{Humidity}) = .970 - (3/5) 0.0 - (2/5) 0.0 = .970$

$Gain(S_{sunny}, \text{Temperature}) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570$

$Gain(S_{sunny}, \text{Wind}) = .970 - (2/5) 1.0 - (3/5) .918 = .019$

See on p. 27
Gain ratio

- info gain favours tests with many outcomes (patient id example)

- consider split info(X) = \[ \sum \frac{|T_i|}{|T|} \log\left(\frac{|T_i|}{|T|}\right) \]

measures potential info. generated by dividing T into n classes (without considering the class info)

gain ratio(X) = gain(X)/split info(X)

shows the proportion of info generated by the split that is useful for classification: in the example (Witten p. 96), \( \log(k)/\log(n) \)

maximize gain ratio
In fact, learning DTs with the gain ratio heuristic is a search:

- Hill-climbing search
- Info gain is the search heuristic
- Covering the examples is the search criterion
- Inductive bias: sorter trees are preferred