
2. Induction: Mining Knowledge from Data
   - Decision tree construction (ID3 and C4.5)
   - Generating variable-valued logic rules (HCV)

3. Deduction of Induction Results
   - ‘No match’ and ‘multiple match’

4. KEshell2: An Intelligent Learning Database System
   - Practical Issues
1. Intelligent Learning Database Systems: A Definition

An ILDB system (Wu 1995, Wu 2000) provides mechanisms for

1. preparation and translation of standard (e.g. relational) database information into a form suitable for use by its induction engines,

2. using induction techniques to extract knowledge from databases, and

3. interpreting the extracted knowledge to solve users’ problems by deduction (or KBS technology).
2. Induction: Mining Knowledge from Data

- Induction paradigms
  - **Supervised classification** (*e.g.* C4.5-like algorithms) (with or without background knowledge)
  - Association analysis (*e.g.* Apriori and FT-Growth)
  - Unsupervised clustering (such as $k$-means, COBWEB and BIRCH)
  - Discovery of quantitative laws (*e.g.* BACON-like systems)

- Criteria for evaluating induction algorithms
  - Time complexity
  - Output compactness
  - Accuracy in different data environments
  * Database characteristics
2.1 Rule Induction: A Simplified Example

Rules produced by HCV:

\[
\begin{align*}
&[X2=[b]] \\
\lor & [X1=[0]] \\
& [X2=[a]] \\
\lor & [X1=[0]] \\
& [X4=[0]] \\
\to & \text{The } F \text{ class.}
\end{align*}
\]

The \( T \) class.

Advantages:

1. Implied but not explicit in the database
2. Smaller in volume
3. More general in description
2.2 Decision Tree Construction

<table>
<thead>
<tr>
<th>ORDER</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>a</td>
<td>#</td>
<td>1</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>a</td>
<td>a</td>
<td>0</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>b</td>
<td>c</td>
<td>1</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>b</td>
<td>b</td>
<td>0</td>
<td>T</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>a</td>
<td>c</td>
<td>1</td>
<td>T</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>b</td>
<td>a</td>
<td>#</td>
<td>T</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>c</td>
<td>c</td>
<td>0</td>
<td>F</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>c</td>
<td>b</td>
<td>1</td>
<td>F</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>c</td>
<td>b</td>
<td>0</td>
<td>T</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>a</td>
<td>a</td>
<td>0</td>
<td>T</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>c</td>
<td>c</td>
<td>1</td>
<td>F</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>c</td>
<td>a</td>
<td>0</td>
<td>T</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>a</td>
<td>b</td>
<td>0</td>
<td>F</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>a</td>
<td>a</td>
<td>1</td>
<td>T</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>b</td>
<td>a</td>
<td>1</td>
<td>T</td>
</tr>
</tbody>
</table>
Decision Tree Construction (2)

S1: \( T \leftarrow \) the whole training set. Create a \( T \) node.

S2: If all examples in \( T \) are positive examples of a specific class, create a ‘yes’ node with \( T \) as its parent and stop; If all examples in \( T \) are negative, create a ‘no’ node with \( T \) as its parent and stop.

S3: Select an attribute \( X \) with values \( V_1, ..., V_N \) and partition \( T \) into subsets \( T_1, ..., T_N \) according to their values on \( X \). Create \( N \) \( T_i \) nodes \((i = 1, ..., N)\) with \( T \) as their parent and \( X = V_i \) as the label of the branch from \( T \) to \( T_i \).

S4: For each \( T_i \) do: \( T \leftarrow T_i \) and goto S2.
ID3 and C4.5: Key ideas

Suppose $T = PE \cup NE$ where $PE$ is the set of positive examples and $NE$ is the set of negative examples, $p = |PE|$ and $n = |NE|$. 

$$I(p, n) = \begin{cases} \frac{-p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n} & \text{when } p \neq 0 \& n \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

(1)

If attribute $X$ with value domain $\{v_1, ..., v_N\}$ is used for the root of the decision tree, it will partition $T$ into $\{T_1, ..., T_N\}$ where $T_i$ contains those examples in $T$ that have value $v_i$ of $X$. Let $T_i$ contain $p_i$ examples of $PE$ and $n_i$ of $NE$. The expected information required for the subtree for $T_i$ is $I(p_i, n_i)$. 

ID3 and C4.5: Key ideas (2)

The expected information required for the tree with $X$ as root, $EI(X)$, is then obtained as weighted average.

$$EI(X) = \frac{1}{N} \sum_{i=1}^{N} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$  \hspace{1cm} (2)

where the weight for the $i$-th branch is the proportion of the examples in $T$ that belong to $T_i$. The information gained by branching on $X$, $G(X)$, is therefore

$$G(X) = I(p, n) - EI(X).$$  \hspace{1cm} (3)
ID3 and C4.5: Key ideas (3)

ID3 examines all candidate attributes, chooses $X$ to maximize $G(X)$, and then uses the same process recursively to construct decision trees for residual subsets $T_1, \ldots, T_N$.

C4.5 adopts a new heuristic, the gain ratio criterion, instead of $G(X)$, for selecting tests in decision tree generation. In the gain ratio criterion, $G(X)/IV(X)$ is used to replace $G(X)$ where

$$IV(X) = \frac{N}{\sum_{i=1}^{N} \frac{p_i + n_i}{p + n} \log_2 \left( \frac{p_i + n_i}{p + n} \right)}$$  \hspace{1cm} (4)
Other topics in C4.5-like algorithms

- Incremental induction: The *windowing* technique.
- Noise handling: Halt tree growth when no more significant information gain can be found.
- Sensible mechanisms for handling missing information.
- Post-pruning of decision trees.
- Decompiling decision trees into production rules.
- Binarization of decision trees.
- Structured induction.
- Real-valued attributes: Discretization.
- Classification of more than 2 classes with a decision tree.
2.3 Generating variable-valued logic rules

The variable-valued logic is a calculus for representing decision problems where decision variables can take on some range of values.

Its principal syntactic entity is a selector with the general form

\[ [X \# R] \] (5)

where \( X \) is a variable or attribute, \( \# \) is a relational operator (such as \( =, \neq, <, >, \leq, \) and \( \geq \)), and \( R \), called a reference, is a list of one or more values that \( X \) could take on.

A well-formed rule in the logic is similar to a production rule but with selectors as the basic components of both its left-hand and right-hand sides.
The HCV algorithm

Let attributes: \[ \{X_1, \ldots, X_a\} = a; \]
positives of a specific concept \( C \): \[ \{e_1^+, \ldots, e_p^+\} = p; \]
negatives: \[ \{e_1^-, \ldots, e_n^-\} = n. \]

1. Partition \( PE \) into \( p' \) \((p' \leq p)\) intersecting groups,
2. Apply a set of heuristics (in HFL) to find a conjunctive \( Hf\ell \) for each intersecting group,
3. Give the final disjunctive rule for the concept \( C \) by logically ORing all the \( Hf\ell \)'s.
The HCV algorithm (2)

**Definition 1.** The extension matrix of $e_k^+$ against $NE$

$$EM_k = (r_{ijk})_{n*a}$$

where

$$r_{ijk} = \begin{cases} * & \text{when } v_{jk}^+ = NEM_{ij} \\ NEM_{ij} & \text{when } v_{jk}^+ \neq NEM_{ij} \end{cases}$$

and ‘*’ denotes a dead element.

**Definition 2.** In an $EM_k$, a set of $n$ nondead elements $r_{iji}$ ($i = 1, ..., n$, $j_i \in \{1, ..., a\}$) that come from the $n$ different $i$ rows is called a **path** in the extension matrix.

A path $\{r_{1j_1}, ..., r_{nj_n}\}$ in an $EM_k$ corresponds to a **conjunctive formula**

$$L = \bigwedge_{i=1}^{n} [X_{ji} \neq r_{iji}]$$

which covers $e_k^+$ against $NE$ and vice versa.
The HCV algorithm (3)

**Definition 3.** Matrix $EMD = (r_{ij})_{n \times a}$ with

$$r_{ij} = \begin{cases} * & \text{when } \exists k_1 \in \{i_1, ..., i_k\} : EM_{k_1}(i, j) = * \\ \bigvee_{k_2=1}^{k} EM_{i_{k_2}}(i, j) & \text{otherwise} \end{cases}$$

is called the disjunction matrix of the positive example set $\{e_{i_1}^+, ..., e_{i_k}^+\}$ against $NE$ or the disjunction matrix of $EM_{i_1}, ..., EM_{i_k}$.

A path $\{r_{1j_1}, ..., r_{nj_n}\}$ in the $EMD$ of $\{e_{i_1}^+, ..., e_{i_k}^+\}$ against $NE$ corresponds to a conjunctive formula or cover

$$L = \bigwedge_{i=1}^{n} [X_{ji} \neq r_{iji}]$$

which covers all of $\{e_{i_1}^+, ..., e_{i_k}^+\}$ against $NE$ and vice versa.

**Definition 4.** If there exists at least one path in the $EMD$ of $\{e_{i_1}^+, ..., e_{i_k}^+\}$ against $NE$, all the positive examples intersect and the positive example set is called an intersecting group.
The HCV algorithm (4)

Heuristics used in HCV:

- **Partitioning**: Divide all the positive examples into intersection groups. Start with the lowest ordered, uncovered positive example each time and try to form an intersecting group as large as possible. Use the following heuristics to generate a conjunctive rule for each intersecting group.

- The *fast* strategy. In an extension matrix $EM_k = (r_{ij})_{n \times a}$, if there is no dead element in a (say $j$) column, then $[X_j \neq r_j]$ where $r_j = \bigvee_{i=1}^{a} r_{ij}$ is chosen as the one selector cover for $EM_k$.

- The *precedence* strategy. When a $r_{ij}$ in column $j$ is the only nondead element of a row $i$, the selector $[X_j \neq r_j]$ is called an inevitable selector and thus is chosen with top precedence.

- The *elimination* strategy. When each appearance of some nondead element in the $j_1$-th column of some row is always coupled with another nondead element in the $j_2$-th column of the same row, $X_{j_1}$ is called an eliminable attribute and thus eliminated by $X_{j_2}$.

- The *least-frequency* strategy. Exclude a least-frequency selector which has least nondead elements in its corresponding column in the extension matrix.
2.4 Processing real-valued attributes

- The simplest class-separating method: Place interval borders between each adjacent pair of examples that belong to different classes.

- Bayesian Classifiers:
  1. For each class $C_i$, draw a distribution probability curve $P(C_i|x)P(C_i)$ where $P(C_i|x)$ indicates the probability of an example belonging to class $C_i$ when the example takes the value of $x$ on the continuous attribute in question.
  2. Cut points are placed so that in each two adjacent intervals the class of the predominant curve changes.
Processing real-valued attributes (2)

- Information gain methods:
  - Binarization of continuous attributes (in C4.5, for example)
  - Find a cut point (e.g. the average) between each adjacent pair of example values of two different classes; and split the current interval where the cut point is located in if (1) the splitting on the value produces information gain and (2) the number of examples in the current interval is greater than a specific threshold.
2.5 Noise handling

Sources of noise:

- Erroneous attribute values
- Missing attribute values (*Don’t Know* values, denoted by `?`)
- Misclassifications (including contradictions)
  - Uneven distribution of training examples in the example space
  - Redundant data: multiple copies of the same examples

*Don’t Care* (`#`) values are not noise!
Stages of noise handling (1)

- Preprocessing of training examples
  - Remove redundant and contradictory data with attention to ? and # values
  - Replace ? values with the most frequent attribute values, (for examples of the same class only or the whole training set)
  - Sensible mechanisms for handling ? values
  - Generate negative/positive examples under the closed world assumption
- Induction time (pre-pruning): Avoid overfitting
Stages of noise handling (2)

• Post-pruning of induction results
  – The TRUNC method in AQ15
  – Reduced error pruning with a separate pruning set of examples
  – Pessimistic pruning in C4.5 with the user-specified confidence level
  – Transformation of the representation language of induction results and pruning the new representation

• Deduction time: Process ‘no match’ and ‘multiple match’ examples.
3. Deduction of Induction Results

• Single match: a test example matches with the description of a specific class

• No match

• Multiple match
3.1 No match

- Largest class
- Measure of fit
- Fuzzy edges
Measure of fit

1. $MF$ for a selector ($sel$) against a no match example, $e$:

$$MF(sel, e) = \begin{cases} 1 & \text{if selector } sel \text{ is satisfied by } e \\ \frac{v}{V} & \text{otherwise} \end{cases}$$ (6)

where $v$ is the number of disjunctively linked attribute values in $sel$ and $V$ is the size of the attribute’s domain

2. $MF$ for a conjunctive rule, $conj$:

$$MF(conj, e) = \frac{n(conj)}{N} \prod_k MF(sel_k, e)$$ (7)

where $n(conj)$ is the number of examples in the training set that $conj$ covers and $N$ is the total number of examples.

3. $MF$ for a class $c_i$ is the probabilistic sum of its conjunctive rules:

$$MF(c_i, e) = MF(conj_1, e) + MF(conj_2) - MF(conj_1, e)MF(conj_2, e)$$ (8)

4. Assign $e$ to the closest class with the highest $MF$ value
3.2 Multiple match

• First hit
• Largest class
• Largest conjunctive rule
• Estimate of Probability
• Fuzzy Interpretation of Discretized Intervals
Estimate of Probability

\[
EP(con_j, e) = \begin{cases} 
\frac{n(con_j)}{N} & \text{if } con_j \text{ is satisfied by } e \\
0 & \text{otherwise}
\end{cases}
\]  
(9)

\[
EP(c_i, e) = EP(con_j_1, e) + EP(con_j_2) - EP(con_j_1, e)EP(con_j_2, e)
\]  
(10)
3.2 Fuzzy Interpretation of Discretized Intervals (Wu 1999)

- Fuzzy borders: each discretized interval is associated with a specific membership function.

- Fuzzy interpretation: a value can be classified into a few different intervals at the same time, with varying degrees, and the interval with the greatest degree is chosen as the value’s discrete value.

- Results: Fuzzy matching works well with no matches, but less encouraging with multiple matches, hence integration of probability estimation and fuzzy matching is recommended.
4. KEshell2: An intelligent learning database system

Monitor: a man-machine interface
KB: knowledge bases
DB: data bases
OS: operating system facilities
Access Storage Interface: KB/DB operators
Utility: a set of common procedures
DBMS: a database management subsystem
KBMS: a KB maintenance subsystem
K.A. Engine: a knowledge acquisition engine
I/D Engine: an inference/deduction engine
Practical Issues in Building ILDB Systems

- Dealing with noise and different types of data (such as symbolic and real-valued): wide attention has been received.
- Instance selection: select or search for a portion of a large database that can be used in data mining instead of using the whole database.
- Structured induction: decompose of a complex problem into a number of subproblems by using domain knowledge and apply an induction algorithm to each of the subproblems.
- Incremental induction in the case of large, dynamic real-world databases.
5. Conclusions

• All the functions and capacities shown in KEshell2 have demonstrated that the target of building intelligent learning database systems to implement automatic KA from DBs is no longer difficult.

• Knowledge discovery from databases is still an important research frontier for both machine learning and database technology.

• ILDB systems have wide potential application to various classification problems, such as diagnosis, pattern recognition, prediction, and decision making, where there are large amounts of data sets to be sorted.

• Data mining has its distinctive goal from related fields (such as machine learning and databases), and require distinctive tools. An ILDB system is one of such tools.