Rule Induction

- General

- The general framework of Inductive Inferences & Assertions
  - Inductive Inference: The general Scheme
    - (A theory & Methodology of Inductive learning ... Michalski 1983)
  - Inductive Inference: Selection Criteria – the LEF (Lexicographic Evaluation Function)
  - Generalization Rules

- The AQ algorithm
  - The star concept
  - The Star generation/ formation algorithm
  - An example
  - Annotated Predicate Calculus: An example
Rule Induction: General

- A symbolic ML method
- Representation Formalism: Predicate Calculus (in most cases)

General Idea:
- Direct rule induction vs. decision-tree’s branches → rules
- Each rule covers a single-set subset of examples
  (most from a single class)
- One-class-at-a-time induction process: POSitive vs. NEGative class vs all-other-classes

Rule Representation:
- Disjunctive Normal Form (DNF)

\[
\begin{array}{ll}
\text{# Conditions} & \text{# Rules} \\
R_i: \text{ AND } A_{ij} & \text{ OR } R_i \Rightarrow C \\
i=1 & i=1
\end{array}
\]
Rule Induction: General

**Advantages**

- Comprehensibility \(\Leftarrow\) tries to explore and fit to a specific class
- Low computational-space cost
- Partitioning of examples/cases into sub-spaces \(\Rightarrow\) conceptual clustering

**Restrictions/ Disadvantages**

- Slow training time \(\Leftarrow\) explores all(?) the alternatives
  - Tuning of many variables \(\Rightarrow\) [AQ]
  - Add ID3 space-search component \(\Rightarrow\) CN2
Inductive Inferences: General Scheme

A Theory and Methodology of Inductive Learning [Michalski 1983]

The RULES induction approach offers a GENERAL INDUCTIVE METHODOLOGY/
FORMALISM \( \rightarrow \) Models GENERALIZATION

Given:

I. Observations (facts) \( F \)
   - Each fact represents specific knowledge about some objects, situations, processes, ...
   - The examples: \( F: e_{ik} \Rightarrow C \)

II. Background Knowledge \( BK \)
   - Assumptions and constraints imposed on observations and generated candidate
     inductive inferences, and any relevant problem domain-knowledge including preference
     criteria characterizing the desirable properties of potential inductive assertions

Find:

An inductive assertion (inference, Hypothesis) \( H \) which tautologically implies the
observations and satisfy the background knowledge

\[ H \Rightarrow_{ti} F \]

\( H > F \) \( \ldots \) \( H \) specializes to \( F \)
\( F | < H \) \( \ldots \) \( G \) generalizes to \( H \)

\( H \) can be characterized as a set of concept recognition rules
Inductive Inferences: Syntactic & Semantic validity

- Inductive assertions should meet the syntactic constraints imposed by the specified representation language and other user-defined syntactic criteria (domain dependent)

- Inductive assertions should meet the following semantic conditions:

  **Completeness:** All examples of one class \( c \in C \), imply all the concept descriptors \( D_c \) (combination/construct of attribute-values), used to describe that class

  \[
  (\forall c \in C) \ (E_c \Rightarrow D_c)
  \]

  **Consistency:** Every concept descriptor \( D_c \) implies that the examples covered by this does not belong to a set of examples of different class

  \[
  (\forall c_i, c_j \in C) \ (D_{c_i} \Rightarrow \neg E_{c_j})
  \]
Inductive Inferences: Selection Criteria

The **PREFERENCE** criterion

- The number of inductive assertions consistent with the observations may be *unlimited* (even when imposing special constraints introduced by BK)
- Need for a criterion that selects *desirable* inductive assertions

**Quality**

- Overall *simplicity* for human comprehension
  - Number of single-descriptors in rules, number of operators used (AND, OR, ...)
- The *degree of fit* between inductive assertions and observations
  - Uncertainty $\rightarrow$ probabilistic assertions, ...
- The *cost* of measuring the values of descriptors
  - Computational-time cost
  - Computational-space (memory) cost
- The amount of information needed for *encoding* the assertion using pre-defined operators and general expressions
  - **Annotated Predicate Calculus - APC**
**Inductive Inferences: Selection Criteria**

**Lexicographic Evaluation Function - LEF**

LEF: \((c_1, t_1), (c_2, t_2), (c_3, t_3), \ldots\)

- **\(c_i\):** elementary criterion
- **\(t_i\):** tolerance threshold  
  e.g., \(t_i \in [80, 100\%]\)

**LEF-based evaluation of inductive assertions**

For each assertion:

1. Apply criteria in LEF in **order**
2. Only inductive assertions succeeded on each criterion are past to the next test
3. Stop when only one (the **BEST**) inductive assertion is left or, when all criteria are applied; In the later case all the remaining assertions are equivalent from the viewpoint of LEF (LEF-equivalent)
Generalization: General concepts

Construction of an Inductive Inference Mechanism

Heuristic State-Space Search

✓ States: Symbolic descriptions
  - Initial State = Set of given observations (facts)

✓ Operators: Inference Rules
  - Generalization, Specialization, Reformulation, ...???

✓ Goal State: Inductive assertion implying observations, satisfying BK, and maximizes the given preference criterions (= meets the maximum of elementary LEF criteria).

Generalization: From concept-descriptor(s) ➔ More general concept-descriptors

\[( E \rightarrow K ) < ( D \rightarrow K ) \iff ( E \Rightarrow D )\]

Thus, to obtain a generalization rule one may use tautological implication of formal logic.

Example

Implication Law: \( P \land Q \Rightarrow P \)

Generalization Rule: \( ( P \land Q \ii K ) < ( P \ii K ) \)
Selective Generalization Rules

**Dropping condition:** simply remove a conjunct

\[
(\text{CTX} \land S \rightarrow K) \mid< (\text{CTX} \rightarrow K)
\]

S is a predicate or, logical expression

**Adding alternative:** add disjunctively a ConteXT

\[
(\text{CTX}_i \rightarrow K) \mid< (\text{CTX}_i \lor \text{CTX}_j \rightarrow K)
\]

**Reformulation**

**Extending reference:** a special case of adding alternative rules

\[
(\text{CTX} \land [D=R_i] \rightarrow K) \mid< (\text{CTX} \land [D=R_j] \rightarrow K)
\]

\[R_i \subseteq R_j \subseteq \text{DOMAIN}(D) = \text{domain (reference) of description } D\]

**Example**

\[\text{DOMAIN(color)} = \{\text{red, green, blue}\}\]

\[
(\text{CTX} \land [\text{color} = \text{red}] \rightarrow K) \mid< (\text{CTX} \land [\text{color} = \{\text{green, blue}\}] \rightarrow K)
\]

If \(R_j = \text{DOMAIN}(D)\), extents to the whole domain, then \(D\) is redundant and can be removed
Selective Generalization Rules

-2

- Closing interval: a special case of extending reference rule applied to linear descriptors ... from numbers to interval(s) of numbers

(CTX ∧ [D=v_i] ::> K) and (CTX ∧ [D=v_j] ::> K) and v_i ≤ v_j

|< (CTX ∧ [D ∈ [v_i ..v_j]] ::> K)

- Climbing generalization tree: a special case of extending reference rule applied to structured descriptors

(CTX ∧ [D=a] ::> K) and (CTX ∧ [D=b] ::> K) and ...(CTX ∧ [D=i] ::> K)

|< (CTX ∧ [D=s] ::> K)

S is the lower-parent node whose descendants include nodes a, b, ..., i

Example

∃ P

(CTX ∧ [shape(P)=triangle] ::> K)
(CTX ∧ [shape(P)=rectangular] ::> K)

polygon ← triangle ∨ rectangular

|< (CTX ∧ [shape(P)=polygon] ::> K)
Selective Generalization Rules

- Turning constraints into variables: from constants to their general type
  \[ F[a], F[b], \ldots, F[x] \text{ and } a,b,\ldots,x \text{ constants of type } X \]
  \[ \langle (\forall X) (F[X]) \rangle \]

Example:
{john, peter} of type <man>
man(john) \Rightarrow mortal(john), and man(peter) \Rightarrow mortal(peter)
\[ \langle \text{man}(X) \Rightarrow \text{mortal}(X) \rangle \]

- Turning conjunction into disjunction:
  powerful but dangerous !!!
  \[ (CTX_i \land CTX_j \Rightarrow K) \quad \langle (CTX_i \lor CTX_j \Rightarrow K) \rangle \]
Selective Generalization Rules

- Inductive Resolution: an inductive elaboration of the resolution principle

Resolution principle:

\[( P \Rightarrow F_i) \land (\neg P \Rightarrow F_j) \vdash (F_i \lor F_j)\]

Inductive Resolution:

\[ ( P \land F_i ::> K ) \text{ and } (\neg P \land F_j ::> K ) \mid< ( F_i \lor F_j ::> K ) \]

Example:
P: “John goes to a movie when he has company”
F₁: “John goes to a movie that has high rating”
F₂: “John goes to a movie when he has plenty of time”

Extension against: identify descriptor’s values that discriminate completely between classes and forget the Context!!

\[ ( CTX \land [ D=r_i ] ::> K_i ) \text{ and } ( CTX \land [ D=r_j ] ::> K_j ) \mid< ( [ D \neq Sr_j ] ::> K_i ) \]

Srₖ is a superset of rₖ, Srₖ ∩ rₖ = ∅

The basic Rule for the AQ family of inductive algorithms
Applying carefully Generalization Rules: The AQ algorithm

Given:

**POS**: number of examples assigned to a pre-specified class

**NEG**: number of examples assigned to classes other than the pre-specified class

Find:

Generalized descriptions of POS examples

Outline of the AQ algorithm

**Step-1**: Randomly select an example \(e\) from \(POS\) \(e_{POS}\)

**Step-2**: Generate a bounded star \(G(e|NEG,m)\). The star of \(e\) against the set of negative examples NEG with no more than \(m\) elements. In the process of star generation apply generalization rules and other domain (in)dependent heuristics.

**Step-3**: In the obtained star find descriptor \(D\) with the highest preference according to LEF.

**Step-4**: If descriptor \(D\) covers set \(POS\) then goto step-6.

**Step-5**: Reduce set \(POS\) to contain only examples not covered by \(D\), and repeat the whole process from step-1.

**Step-6**: The disjunction of all generated descriptions \(D\) is a complete and consistent set of inductive assertions for POS. Apply reformulation rules (e.g., introduced by background knowledge) to obtain simpler rules.
The AQ algorithm: The concept of a **Star**

**POS:** number of examples assigned to a pre-specified class  
**NEG:** number of examples assigned to classes other than the pre-specified class

---

**Star of an event e under constraints E**  
All possible alternative non-redundant descriptions of event e that do not violate constraints E

\[ G(e|E) \]  
\[ E = \text{constraints} \]

Redundancy: \((A \text{ and } B)\) is redundant of \((A \text{ or } B)\)

---

**Star of example e against set NEG**  
The set of all maximally generated c-expressions that cover event e and do not cover any (or, cover a pre-specified proportion) of the negative examples in NEG

- **c-expression:** \(<\text{Quantifier}> <\text{Conjunction of relational statements}>\)

  **APC:** Annotated Predicate Calculus

  **Example:** \((\exists P0,P1) ([\text{contains}(P0,P1)] ([\text{ontop}(P1,P0)]) ([\text{length}(P1) = 3 .. 5]) \ldots.)\)
The AQ algorithm: Forming a Star

1. **Elementary star:** \( G(e/ e_i), \quad e_i \in E \)
   
   \( e = (x_1, x_2, \ldots, x_g) \)
   
   \( e_i = (r_1, r_2, \ldots, r_g) \)
   
   \[ G(e/ e_i) = \{x_j \mid [x_j \neq r_j]\}, \quad j = 1, 2, \ldots, g, \quad 1 \leq |G(e/e_i)| \leq g \]

2. **Complete (bounded) star:**
   
   \[ G(e/ NEG,m) = \bigwedge_{i=1}^{\text{NEG}} G(e/e_i), \quad e_i \in NEG \]
   
   The complete star is formed by the LOGICAL MULTIPLICATION of the elementary stars \( G(e/e_i) \).

3. **Reduction and simplification of complete stars:**
   
   - **Sparseness:** \( r'(\text{complex}) = 1 - \frac{\text{coveredPOSexamples}}{\text{uncoveredPOSexamples}} \)
   
   - Generalization of complexes using Generalization rules
Forming a Star: Example

**Examples**: \( E \)

- \( e_1: a_1, b_1 \rightarrow c_1 \) (POS)
- \( e_2: a_1, b_2 \rightarrow c_2 \) (NEG)
- \( e_3: a_2, b_1 \rightarrow c_2 \) (NEG)
- \( e_4: a_2, b_2 \rightarrow c_2 \) (NEG)

**Step-1**: \( e_1 \) from POS is selected (… the SEED)

1. **Elementary stars**
   - \( G(e_1/e_2) = \{b_1\} \) .... complex-1
   - \( G(e_1/e_3) = \{a_1\} \) .... complex-2
   - \( G(e_1/e_4) = \{a_1, b_1\} \) .... complex-3

2. **Complete star**
   2.1. \( G(e_1|NEG) = \{b_1\} \) AND \( \{a_1\} \) AND \( \{a_1, b_1\} \) .... LOGICAL MULTIPLICATIONS
   2.2. Absorb redundant complexes = \( \{a_1, b_1\} \)

3. **LEF** on \( \{a_1, b_1\} \)
   3.1. **Sparseness**
   
   \[ r(\{a_1, b_1\}) = 1 - \frac{\text{coveredPOS}}{\text{uncoveredPOS}} = 1 - \frac{1}{0} = 0 \]

   ✓ ‘0’ is the lowest sparseness

**Step-2**: Set \( E = E - \{e_1\} \) ... goto step-1

**Step-3**: Repeat for NEG examples
The complete AQ algorithm

Let POS be a set of positive examples of class C
Let NEG be a set of negative examples of class C

Procedure AQ(POS, NEG)

Let COVER be the empty cover.
While COVER does not cover all examples in POS,
    Select a SEED (a positive example not covered by COVER).
    Let STAR be STAR(SEED|NEG) (a set of complexes that cover SEED but
        that cover no examples in NEG)
    Let BEST be the complex in STAR (according to user-specified
        criteria ... LEF)
    Add BEST as an extra disjunct to COVER
Return COVER.

Procedure STAR(SEED, NEG)

Let STAR be the set containing the empty complex.
While any complex in STAR covers some negative examples in NEG,
    Select a negative example E_{neg} covered by a complex in STAR.
    Specialize complexes in STAR to exclude E_{neg} by:
        Let EXTENSION be all selectors that cover SEED but not E_{neg}
        Let STAR be the set \{x \land y \mid x \in STAR, y \in EXTENSION\}.
        Remove all complexes in STAR subsumed by other complexes.
    Repeat until size of STAR \leq maxstar (a user-specified maximum).
    Remove the worst complex from STAR.
Return STAR.
Annotated Predicate Calculus: An example

∃CELL, B1, B2, B3, ...
[contains(CELL1, B1, B2, B3, ...)]
[circle(CELL1)=8]
[pplasm(CELL1)=A]
[shape(B1)=ellipse] [texture(B1)=strps] [weight(B1)=4] [orientation(b1)=NW]
[shape(B2)=circle] [contains(B2,B3)] [texture(B2)=blank] [weight(B2)=3] ...

::> [class = cancerous_cell]

... from segmentation

e: a cancerous cell
Step-1: Descriptions and their annotations

Global descriptors
- \textit{circ}: number of segments in the circumference
  Type: linear
  Domain: \{1, 2, ..., N\}
- \textit{pplasm}: the type of protoplasm
  Type: nominal
  Domain: \{A, B, C, ...\}

Local descriptors
- \textit{shape}(Bi): the shape of bodies in the cell
  Type: structured
  Domain: \{triangle, circle, ellipse, heptagon, square, boat, ...\}
  Domain Background Knowledge:
  \[ \text{shape=triangle OR square OR heptagon} \ \Rightarrow \ \text{shape=polygon} \]
  \[ \text{shape=oval OR polygon} \ \Rightarrow \ \text{shape=regular} \]
  \[ \text{shape=spring OR boat} \ \Rightarrow \ \text{shape=irregular} \]
- \textit{texture}(Bi): the texture of bodies
  Type: nominal
  Domain: \{blank, shaded, solid-black, solid-grey, stripes, crossed, wavy, ...\}
weight(Bi): weight of bodies
Type: linear
Domain: \{1, 2, ..., N\}

orientation(Bi): orientation of bodies
Type: linear
Domain: \{N, NE, E, SE, S, SW, W, NW, ...\}

contains(C,B1,B2,......): bodies contained in cell
Type: nominal
Domain: \{T(\text{rue}), F(\text{alse})\}

hastails(B,L1,L2,...): if a body has tails L1, L2, ...
Type: nominal
Domain: \{T(\text{rue}), F(\text{alse})\}

Step-2: State observations (... examples)
\exists \text{CELL, B1, B2, B3, ...}

.............................

::> class
Step-3: Constructive induction rules (Background Knowledge)

\[ \text{[black\_boat\_body}(B)]] \leftarrow \text{[texture}(B) = \text{solid\_black}] \text{ and } \text{[shape}(B) = \text{boat}] \\
\text{[total\_bodies\_in\_cell]} \leftarrow <\text{number}> \\
\text{[sub\_bodies\_in\_b}(B,n)] \leftarrow <\text{number}> \\

Step-4: LEFs

- **Constraints** on the output

- **Characteristic** descriptors: Maximize the length of the complete inductive assertions

- **Discriminant** descriptions: Minimize the length of consistent and complete inductive assertions
**Annotated Predicate Calculus: An example**

---

**weight**(\(B_i\)): weight of bodies  
Type: linear  
Domain: \{1, 2, ..., N\}

**orientation**(\(B_i\)): orientation of bodies  
Type: linear  
Domain: \{N, NE, E, SE, S, SW, W, NW, ...\}

**contains**(\(C, B_1, B_2, ...\)): bodies contained in cell  
Type: nominal  
Domain: \{\text{T}(\text{true}), \text{F}(\text{false})\}

**hastails**(\(B, L_1, L_2, ...\)): if a body has tails \(L_1, L_2, ...\)  
Type: nominal  
Domain: \{\text{T}(\text{true}), \text{F}(\text{false})\}

---

**Step-2: State observations (... examples)**  
\(\exists\text{CELL, } B_1, B_2, B_3, ...\)

..............................................

::> class