Learning Sets of Rules

• Sequential covering algorithms
• FOIL
• Induction as inverse of deduction
• Inductive Logic Programming

Web resources:

• http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/ilp.html
• http://www-ai.ijs.si/~ilpnet2
Learning Disjunctive Sets of Rules

Method 1: Learn decision tree, convert to rules

Method 2: Sequential covering algorithm:
1. Learn one rule with high accuracy, any coverage
2. Remove positive examples covered by this rule
3. Repeat
Sequential Covering Algorithm

**SEQUENTIAL-COVERING**(Target_attribute, Attributes, Examples, Threshold)

- Learned_rules ← {}
- Rule ← LEARN-ONE-RULE(Target_attribute, Attributes, Examples)
- while PERFORMANCE(Rule, Examples) > Threshold, do
  - Learned_rules ← Learned_rules + Rule
  - Examples ← Examples − {examples correctly classified by Rule}
  - Rule ← LEARN-ONE-RULE(Target_attribute, Attributes, Examples)
- Learned_rules ← sort Learned_rules according to PERFORMANCE over Examples
- return Learned_rules
Learn-One-Rule

- IF Wind=weak THEN PlayTennis=yes
- IF Wind=strong THEN PlayTennis=no
- IF Humidity=normal THEN PlayTennis=yes
- IF Humidity=high THEN PlayTennis=no
- IF Humidity=normal Wind=weak THEN PlayTennis=yes
- IF Humidity=normal Wind=strong THEN PlayTennis=yes
- IF Humidity=normal Outlook=sunny THEN PlayTennis=yes
- IF Humidity=normal Outlook=rain THEN PlayTennis=yes
- ...
Learn-One-Rule

• $Pos \leftarrow$ positive Examples
• $Neg \leftarrow$ negative Examples
• while $Pos$, do

  Learn a NewRule
  – $NewRule \leftarrow$ most general rule possible
  – $NewRuleNeg \leftarrow Neg$
  – while $NewRuleNeg$, do

    Add a new literal to specialize $NewRule$
    1. $Candidate\_literals \leftarrow$ generate candidates
    2. $Best\_literal \leftarrow \text{argmax}_L \in Candidate\_literals$ Performance($SpecializeRule($NewRule$, L)$)
    3. add $Best\_literal$ to $NewRule$ preconditions
    4. $NewRuleNeg \leftarrow$ subset of $NewRuleNeg$ that satisfies $NewRule$ preconditions
        – $Learned\_rules \leftarrow Learnt\_rules + NewRule$
        – $Pos \leftarrow Pos - \{\text{members of } Pos \text{ covered by } NewRule\}$

• Return $Learned\_rules$
Subtleties: Learn One Rule

1. May use beam search
2. Easily generalizes to multi-valued target functions
3. Choose evaluation function to guide search:
   - Entropy (i.e., information gain)
   - Sample accuracy: \( \frac{n_c}{n} \) where \( n_c = \) correct rule predictions, \( n = \) all predictions
   - \( m \) estimate: \( \frac{n_c + mp}{n + m} \)
Variants of Rule Learning Programs

• *Sequential* or *simultaneous* covering of data?
• General → specific, or specific → general?
• Generate-and-test, or example-driven?
• Whether and how to post-prune?
• What statistical evaluation function?
Learning First Order Rules

Why do that?

• Can learn sets of rules such as
  \[
  \text{Ancestor}(x, y) \leftarrow \text{Parent}(x, y)
  \]
  \[
  \text{Ancestor}(x, y) \leftarrow \text{Parent}(x, z) \land \text{Ancestor}(z, y)
  \]

• General purpose programming language PROLOG: programs are sets of such rules
First Order Rule for Classifying Web Pages

[Slattery, 1997]

course(A) ←
    has-word(A, instructor),
    Not has-word(A, good),
    link-from(A, B),
    has-word(B, assign),
    Not link-from(B, C)

Train: 31/31, Test: 31/34
FOIL

• First-Order Induction of Logic (FOIL)
• Learns Horn clauses without functions
• Allows negated literals in rule body
• Sequential covering algorithm
  – Greedy, hill-climbing approach
  – Seeks only rules for predicting True
• Each new rule generalizes overall concept \((S \rightarrow G)\)
• Each added conjunct specializes rule \((G \rightarrow S)\)
FOIL($Target\_predicate, Predicates, Examples$)

- $Pos \leftarrow$ positive $Examples$
- $Neg \leftarrow$ negative $Examples$
- while $Pos$, do

  Learn a NewRule
  - $NewRule \leftarrow$ most general rule possible
  - $NewRuleNeg \leftarrow Neg$
  - while $NewRuleNeg$, do

    Add a new literal to specialize $NewRule$
    1. $Candidate\_literals \leftarrow$ generate candidates
    2. $Best\_literal \leftarrow$
        $\arg \max_{L \in Candidate\_literals} Foil\_Gain(L, NewRule)$
    3. add $Best\_literal$ to $NewRule$ preconditions
    4. $NewRuleNeg \leftarrow$ subset of $NewRuleNeg$ that satisfies $NewRule$ preconditions

- $Learned\_rules \leftarrow Learned\_rules + NewRule$
- $Pos \leftarrow Pos - \{\text{members of } Pos \text{ covered by } NewRule\}$

- Return $Learned\_rules$
Specializing Rules in FOIL

Learning rule: $P(x_1, x_2, \ldots, x_k) \leftarrow L_1 \ldots L_n$
Candidate specializations add new literal of form:

- $Q(v_1, \ldots, v_r)$, where at least one of the $v_i$ in the created literal must already exist as a variable in the rule.
- $Equal(x_j, x_k)$, where $x_j$ and $x_k$ are variables already present in the rule.
- The negation of either of the above forms of literals
Information Gain in FOIL

\[ \text{Foil\_Gain}(L, R) \equiv t \left( \log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right) \]

Where

- \( L \) is the candidate literal to add to rule \( R \)
- \( p_0 \) = number of positive bindings of \( R \)
- \( n_0 \) = number of negative bindings of \( R \)
- \( p_1 \) = number of positive bindings of \( R + L \)
- \( n_1 \) = number of negative bindings of \( R + L \)
- \( t \) is the number of positive bindings of \( R \) also covered by \( R + L \)

Note

- \(- \log_2 \frac{p_0}{p_0+n_0}\) is optimal number of bits to indicate the class of a positive binding covered by \( R \)
FOIL Example

Instances:

- pairs of nodes, e.g. \( \langle 1, 5 \rangle \), with graph described by literals \( \text{LinkedTo}(0,1), \neg \text{LinkedTo}(0,8) \) etc.

Target function:

- \( \text{CanReach}(x,y) \) true iff directed path from \( x \) to \( y \)

Hypothesis space:

- Each \( h \in H \) is a set of horn clauses using predicates \( \text{LinkedTo} \) (and \( \text{CanReach} \))
Induction as Inverted Deduction

Induction is finding $h$ such that

$$(\forall \langle x_i, f(x_i) \rangle \in D) \ B \land h \land x_i \vdash f(x_i)$$

where

- $x_i$ is $i$th training instance
- $f(x_i)$ is the target function value for $x_i$
- $B$ is other background knowledge

So let’s design inductive algorithm by inverting operators for automated deduction!
Induction as Inverted Deduction

“pairs of people, \( \langle u, v \rangle \) such that child of \( u \) is \( v \),”

\[ f(x_i) : \quad \text{Child}(\text{Bob}, \text{Sharon}) \]

\[ x_i : \quad \text{Male}(\text{Bob}), \text{Female}(\text{Sharon}), \text{Father}(\text{Sharon}, \text{Bob}) \]

\[ B : \quad \text{Parent}(u, v) \leftarrow \text{Father}(u, v) \]

What satisfies \( (\forall \langle x_i, f(x_i) \rangle \in D) \ B \land h \land x_i \vdash f(x_i) \)?

\[ h_1 : \quad \text{Child}(u, v) \leftarrow \text{Father}(v, u) \]

\[ h_2 : \quad \text{Child}(u, v) \leftarrow \text{Parent}(v, u) \]
Induction is, in fact, the inverse operation of deduction, and cannot be conceived to exist without the corresponding operation, so that the question of relative importance cannot arise. Who thinks of asking whether addition or subtraction is the more important process in arithmetic? But at the same time much difference in difficulty may exist between a direct and inverse operation; . . . it must be allowed that inductive investigations are of a far higher degree of difficulty and complexity than any questions of deduction. . . .

(Jevons 1874)
Induction as Inverted Deduction

We have mechanical *deductive* operators $F(A, B) = C$, where $A \land B \vdash C$

need *inductive* operators

$O(B, D) = h$ where $(\forall \langle x_i, f(x_i) \rangle \in D) (B \land h \land x_i) \vdash f(x_i)$
Induction as Inverted Deduction

Positives:

• Subsumes earlier idea of finding \( h \) that “fits” training data

• Domain theory \( B \) helps define meaning of “fit” the data

\[
B \land h \land x_i \vdash f(x_i)
\]

• Suggests algorithms that search \( H \) guided by \( B \)
Induction as Inverted Deduction

Negatives:

- Doesn’t allow for noisy data. Consider

\[(\forall \langle x_i, f(x_i) \rangle \in D) (B \land h \land x_i) \vdash f(x_i)\]

- First order logic gives a huge hypothesis space \(H\)
  \(\rightarrow\) overfitting...
  \(\rightarrow\) intractability of calculating all acceptable \(h\)’s
Deduction: Resolution Rule

\[ P \lor L \]
\[ \neg L \lor R \]
\[ \frac{P \lor R}{P \lor R} \]

1. Given initial clauses \( C_1 \) and \( C_2 \), find a literal \( L \) from clause \( C_1 \) such that \( \neg L \) occurs in clause \( C_2 \).

2. Form the resolvent \( C \) by including all literals from \( C_1 \) and \( C_2 \), except for \( L \) and \( \neg L \). More precisely, the set of literals occurring in the conclusion \( C \) is

\[ C = (C_1 - \{L\}) \cup (C_2 - \{\neg L\}) \]

where \( \cup \) denotes set union, and “−” denotes set difference.
Inverting Resolution
Inverted Resolution (Propositional)

1. Given initial clauses $C_1$ and $C$, find a literal $L$ that occurs in clause $C_1$, but not in clause $C$.

2. Form the second clause $C_2$ by including the following literals

$$C_2 = (C - (C_1 - \{L\})) \cup \{\neg L\}$$
First order resolution

First order resolution:

1. Find a literal $L_1$ from clause $C_1$, literal $L_2$ from clause $C_2$, and substitution $\theta$ such that $L_1\theta = \neg L_2\theta$

2. Form the resolvent $C$ by including all literals from $C_1\theta$ and $C_2\theta$, except for $L_1\theta$ and $\neg L_2\theta$. More precisely, the set of literals occurring in the conclusion $C$ is

\[ C = (C_1 - \{L_1\})\theta \cup (C_2 - \{L_2\})\theta \]
Inverting First order resolution

\[ C_2 = (C - (C_1 - \{L_1\})\theta_1)\theta_2^{-1} \cup \{-L_1\theta_1\theta_2^{-1}\} \]
Cigol

```
Father (Tom, Bob)  \quad GrandChild(y,x) \lor \neg Father(x,z) \lor \neg Father(z,y)

{Bob/y, Tom/z}

Father (Shannon, Tom)  \quad GrandChild(Bob,x) \lor \neg Father(x, Tom)

{Shannon/x}

GrandChild(Bob, Shannon)
```
PROGOL: Reduce comb. explosion by generating the most specific acceptable $h$

1. User specifies $H$ by stating predicates, functions, and forms of arguments allowed for each

2. **PROGOL** uses sequential covering algorithm. For each $\langle x_i, f(x_i) \rangle$
   - Find most specific hypothesis $h_i$ s.t.
     $$B \land h_i \land x_i \vdash f(x_i)$$
     – actually, considers only $k$-step entailment

3. Conduct general-to-specific search bounded by specific hypothesis $h_i$, choosing hypothesis with minimum description length
Summary: Learning Rule Sets

- Sequential (set) covering
- Inductive Logic Programming (ILP)
  - FOIL
  - Inverse resolution