Relational Learning

- flat or modular or hierarchical
- explicit states or features or **individuals and relations**
- static or finite stage or indefinite stage or infinite stage
- fully observable or partially observable
- deterministic or stochastic dynamics
- goals or complex preferences
- single agent or multiple agents
- knowledge is given or **knowledge is learned**
- perfect rationality or bounded rationality
Relational Learning

- Often the values of properties are not meaningful values but names of individuals.
- It is the properties of these individuals and their relationship to other individuals that needs to be learned.
- Relational learning has been studied under the umbrella of “Inductive Logic Programming” as the representations are often logic programs.
Example: trading agent

What does Joe like?

<table>
<thead>
<tr>
<th>Individual</th>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>joe</td>
<td>likes</td>
<td>resort_14</td>
</tr>
<tr>
<td>joe</td>
<td>dislikes</td>
<td>resort_35</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>resort_14</td>
<td>type</td>
<td>resort</td>
</tr>
<tr>
<td>resort_14</td>
<td>near</td>
<td>beach_18</td>
</tr>
<tr>
<td>beach_18</td>
<td>type</td>
<td>beach</td>
</tr>
<tr>
<td>beach_18</td>
<td>covered_in</td>
<td>ws</td>
</tr>
<tr>
<td>ws</td>
<td>type</td>
<td>sand</td>
</tr>
<tr>
<td>ws</td>
<td>color</td>
<td>white</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Values of properties may be meaningless names.
Example: trading agent

Possible theory that could be learned:

\[
\begin{align*}
&\text{prop}(joe, \text{likes}, R) \leftarrow \\
&\quad \text{prop}(R, \text{type}, \text{resort}) \land \\
&\quad \text{prop}(R, \text{near}, B) \land \\
&\quad \text{prop}(B, \text{type}, \text{beach}) \land \\
&\quad \text{prop}(B, \text{covered in}, S) \land \\
&\quad \text{prop}(S, \text{type}, \text{sand}).
\end{align*}
\]

Joe likes resorts that are near sandy beaches.
A is a set of atoms whose definitions the agent is learning.

$E^+$ is a set of ground atoms observed true: **positive examples**

$E^-$ is the set of ground atoms observed to be false: **negative examples**

$B$ is a set of clauses: **background knowledge**

$H$ is a space of possible hypotheses. $H$ can be the set of all logic programs defining $A$.

The aim is to find a simplest hypothesis $h \in H$ such that

\[
B \land h \models E^+ \quad \text{and} \quad B \land h \not\models E^-
\]
Generality of Hypotheses

Hypothesis $H_1$ is more general than $H_2$ if $H_1$ logically implies $H_2$. $H_2$ is then more specific than $H_1$. 

Consider the logic programs:

\begin{align*}
  & a \leftarrow b. \\
  & a \leftarrow b \land c. \\
  & a \leftarrow \neg c. \\
  & a.
\end{align*}

Which is the most general? Least general?

For target relation $A = \{ t(X_1, \ldots, X_n) \}$ what is the most general logic program?

What is the least general logic program that is consistent with $E^+$ and $E^-$?
Hypothesis $H_1$ is more general than $H_2$ if $H_1$ logically implies $H_2$. $H_2$ is then more specific than $H_1$.

Consider the logic programs:

- $a \leftarrow b$
- $a \leftarrow b \land c$
- $a \leftarrow b$. $a \leftarrow c$
- $a$

Which is the most general? Least general?
Generality of Hypotheses

Hypothesis $H_1$ is **more general** than $H_2$ if $H_1$ logically implies $H_2$. $H_2$ is then **more specific** than $H_1$.

Consider the logic programs:

- $a \leftarrow b$.
- $a \leftarrow b \land c$.
- $a \leftarrow b$.  $a \leftarrow c$.
- $a$.

Which is the most general? Least general?

- For target relation $A = \{t(X_1, \ldots, X_n)\}$ what is the most general logic program?
Generality of Hypotheses

Hypothesis $H_1$ is more general than $H_2$ if $H_1$ logically implies $H_2$. $H_2$ is then more specific than $H_1$. Consider the logic programs:

- $a ← b$.
- $a ← b ∧ c$.
- $a ← b$. $a ← c$.
- $a$.

Which is the most general? Least general?

- For target relation $A = \{t(X_1, \ldots, X_n)\}$ what is the most general logic program?
- What is the least general logic program that is consistent with $E^+$ and $E^-$?
Single target relation: \( A = \{ t(X_1, \ldots, X_n) \} \).

Two main approaches:

- Start with the most general hypothesis and make it more complicated to fit the data.

  Initially the logic program can be \( E^+ \). Operators simplify the program, ensuring it fits the training examples.
Inductive Logic Programming: Main Approaches

Single target relation: $A = \{t(X_1, \ldots, X_n)\}$.

Two main approaches:

- Start with the most general hypothesis and make it more complicated to fit the data. Most general hypothesis is $t(X_1, \ldots, X_n)$.

Keep adding conditions, ensuring it always implies the positive examples. At each step, exclude some negative examples.
Single target relation: $A = \{ t(X_1, \ldots, X_n) \}$.
Two main approaches:

- Start with the most general hypothesis and make it more complicated to fit the data. Most general hypothesis is $t(X_1, \ldots, X_n)$.

  Keep adding conditions, ensuring it always implies the positive examples. At each step, exclude some negative examples.

- Start with a hypothesis that fits the data and keep making it simpler while still fitting the data.
Single target relation: $A = \{t(X_1, \ldots, X_n)\}$.

Two main approaches:

- Start with the most general hypothesis and make it more complicated to fit the data. Most general hypothesis is $t(X_1, \ldots, X_n)$.

  Keep adding conditions, ensuring it always implies the positive examples. At each step, exclude some negative examples.

- Start with a hypothesis that fits the data and keep making it simpler while still fitting the data. Initially the logic program can be $E^+$. Operators simplify the program, ensuring it fits the training examples.
Maintain a logic program $G$ that entails the positive examples. Initially:

$$G = \{ t(X_1, \ldots, X_n) \leftarrow \}$$

A specialization operator takes $G$ and returns set $S$ of clauses that specializes $G$. Thus $G \models S$.

Three primitive specialization operators:

- Split a clause in $G$ on condition $c$. Clause $a \leftarrow b$ in $G$ is replaced by two clauses: $a \leftarrow b \land c$ and $a \leftarrow b \land \neg c$.
- Split clause $a \leftarrow b$ on variable $X$, producing:
  $$a \leftarrow b \land X = t_1.$$  
  $$\ldots$$  
  $$a \leftarrow b \land X = t_k.$$  
  where the $t_i$ are terms.
- Remove any clause not necessary to prove the positive examples.
Top-down Inductive Logic Program

1: procedure TDInductiveLogicProgram\( (t, B, E^+, E^-, R) \)

2: \( t \): an atom whose definition is to be learned

3: \( B \): background knowledge is a logic program

4: \( E^+ \): positive examples

5: \( E^- \): negative examples

6: \( R \): set of specialization operators

7: Output: logic program that classifies \( E^+ \) positively and \( E^- \) negatively or \( \bot \) if no program can be found

8: \( H \leftarrow \{ t(X_1, \ldots, X_n) \leftarrow \} \)

9: while there is \( e \in E^- \) such that \( B \cup H \models e \) do

10: if there is \( r \in R \) such that \( B \cup r(H) \models E^+ \) then

11: Choose \( r \in R \) such that \( B \cup r(H) \models E^+ \)

12: \( H \leftarrow r(H) \)

13: else

14: return \( \bot \)

15: return \( H \)