# 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?

| Individual | Property | Value |
|---|---|---|
| *joe* | *likes* | *resort_14* |
| *joe* | *dislikes* | *resort_35* |
| ... | ... | ... |
| *resort_14* | *type* | *resort* |
| *resort_14* | *near* | *beach_18* |
| *beach_18* | *type* | *beach* |
| *beach_18* | *covered_in* | *ws* |
| *ws* | *type* | *sand* |
| *ws* | *color* | *white* |
| ... | ... | ... |

Values of properties may be meaningless names.

# Example: trading agent

Possible theory that could be learned:

$$
\begin{aligned}
& prop(joe, likes, R) \leftarrow \\
& \qquad prop(R, type, resort) \wedge \\
& \qquad prop(R, near, B) \wedge \\
& \qquad prop(B, type, beach) \wedge \\
& \qquad prop(B, covered\_in, S) \wedge \\
& \qquad prop(S, type, sand).
\end{aligned}
$$

Joe likes resorts that are near sandy beaches.

# Inductive Logic Programming: Inputs and Output

- $A$ 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

# Inductive Logic Programming: Inputs and Output

- $A$ 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

# Inductive Logic Programming: Inputs and Output

- $A$ 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 $A$, or more restricted set.

# Inductive Logic Programming: Inputs and Output

- $A$ 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 $A$, or more restricted set.

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

$B \wedge h \models E^+$ and

$B \wedge h \not\models E^-$

# 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 $H_1$.

# 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:

- $a \leftarrow b.$
- $a \leftarrow b \wedge c.$
- $a \leftarrow b. \quad a \leftarrow c.$
- $a.$

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:

- $a \leftarrow b.$
- $a \leftarrow b \wedge c.$
- $a \leftarrow b. \quad a \leftarrow c.$
- $a.$

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:

- $a \leftarrow b.$
- $a \leftarrow b \wedge c.$
- $a \leftarrow b. \quad a \leftarrow c.$
- $a.$

Which is the most general? Least general?

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

# 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.

# 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.

# 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.
- Start with a hypothesis that fits the data and keep making it simpler while still fitting the data.

# 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.
- 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.

# Inductive Logic Programming: General to Specific Search

Maintain a logic program $G$ that entails the positive examples.
Initially:

# Inductive Logic Programming: General to Specific Search

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 $G \models S$.

# Inductive Logic Programming: General to Specific Search

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: $a \leftarrow b \wedge c$ and $a \leftarrow b \wedge \neg c$.

# Inductive Logic Programming: General to Specific Search

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: $a \leftarrow b \wedge c$ and $a \leftarrow b \wedge \neg c$.
- Split clause $a \leftarrow b$ on variable $X$, producing:

$$a \leftarrow b \wedge X = t_1.$$
$$\ldots$$
$$a \leftarrow b \wedge X = t_k.$$

  where the $t_i$ are terms.

# Inductive Logic Programming: General to Specific Search

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: $a \leftarrow b \wedge c$ and $a \leftarrow b \wedge \neg c$.
- Split clause $a \leftarrow b$ on variable $X$, producing:

  $$a \leftarrow b \wedge X = t_1.$$

  $$\ldots$$

  $$a \leftarrow b \wedge X = t_k.$$

  where the $t_i$ are terms.
- Remove any clause not necessary to prove the positive examples.

# Top-down Inductive Logic Program

**procedure** $TDInductiveLogicProgram(t, B, E^+, E^-, R)$
  $t$: an atom whose definition is to be learned
  $B$: background knowledge is a logic program
  $E^+$: positive examples
  $E^-$: negative examples
  $R$: set of specialization operators
  **Output**: logic program that classifies $E^+$ positively and $E^-$ negatively or $\perp$ if no program can be found

# Top-down Inductive Logic Program

**procedure** $TDInductiveLogicProgram(t, B, E^+, E^-, R)$

    $t$: an atom whose definition is to be learned

    $B$: background knowledge is a logic program

    $E^+$: positive examples

    $E^-$: negative examples

    $R$: set of specialization operators

    **Output**: logic program that classifies $E^+$ positively and $E^-$ negatively or $\perp$ if no program can be found

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

    **while** there is $e \in E^-$ such that $B \cup H \models e$ **do**

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

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

            $H \leftarrow r(H)$

        **else**

            **return** $\perp$

    **return** $H$