14 Machine Learning

14.1 Introduction

14.2 Concept Learning

14.3 Learning Sets of Rules

14.1 Introduction

- Why machine learning?
- What is a well-defined learning problem?
- An example: learning to play checkers.
- What questions should we ask about Machine Learning?
Why Machine Learning?

- Recent progress in algorithms and theory.
- Growing flood of online data.
- Computational power is available.
- Budding industry.
- Niches for machine learning:
  - Data mining: using historical data to improve decisions.
  - Medical records \(\rightsquigarrow\) medical knowledge.
  - Software applications: we can’t program by hand.
  - Autonomous driving.
  - Speech recognition.
  - Self customizing programs.
  - Newsreader that learns user interests.
Typical Datamining Task

▶ **Data:**

- Patient103
  - time=1
  - Age: 23
  - FirstPregnancy: no
  - Anemia: no
  - Diabetes: no
  - PreviousPrematureBirth: no
  - Ultrasound: ?
  - Elective C–Section: ?
  - Emergency C–Section: ?
  - ...

- Patient103
  - time=2
  - Age: 23
  - FirstPregnancy: no
  - Anemia: no
  - Diabetes: YES
  - PreviousPrematureBirth: no
  - Ultrasound: abnormal
  - Elective C–Section: no
  - Emergency C–Section: ?
  - ...

- Patient103
  - time=n
  - Age: 23
  - FirstPregnancy: no
  - Anemia: no
  - Diabetes: no
  - PreviousPrematureBirth: no
  - Ultrasound: ?
  - Elective C–Section: no
  - Emergency C–Section: Yes
  - ...

▶ **Given:**

- 9714 patient records, each describing a pregnancy and birth
- Each patient record contains 215 features

▶ **Learn to predict:**

- Classes of future patients at high risk for Emergency Cesarean Section
Datamining Result

▶ Data:

Patient103 time=1 → Patient103 time=2 → ... → Patient103 time=n

Age: 23
FirstPregnancy: no
Anemia: no
Diabetes: no
PreviousPrematureBirth: no
Ultrasound: ?
Elective C−Section: ?
Emergency C−Section: ?
...

Age: 23
FirstPregnancy: no
Anemia: no
Diabetes: YES
PreviousPrematureBirth: no
Ultrasound: abnormal
Elective C−Section: no
Emergency C−Section: ?
...

Age: 23
FirstPregnancy: no
Anemia: no
Diabetes: no
PreviousPrematureBirth: no
Ultrasound: ?
Elective C−Section: no
Emergency C−Section: Yes
...

▶ One of 18 learned rules:

If No previous vaginal delivery, and
Abnormal 2nd Trimester Ultrasound, and
Malpresentation at admission
Then Probability of Emergency C−Section is 0.6

Over training data: 26/41 = .63,
Over test data: 12/20 = .60
Credit Risk Analysis

Data:

Customer103: (time=t0)
- Years of credit: 9
- Loan balance: $2,400
- Income: $52k
- Own House: Yes
- Other delinquent accts: 2
- Max billing cycles late: 3
- Profitable customer?: ?

... Customer103: (time=t1)
- Years of credit: 9
- Loan balance: $3,250
- Income: ?
- Own House: Yes
- Other delinquent accts: 2
- Max billing cycles late: 4
- Profitable customer?: ?

... Customer103: (time=tn)
- Years of credit: 9
- Loan balance: $4,500
- Income: ?
- Own House: Yes
- Other delinquent accts: 3
- Max billing cycles late: 6
- Profitable customer?: No

...
Credit Risk Analysis Continued

► Rules learned from synthesized data:

If Other-Delinquent-Accounts > 2, and Number-Delinquent-Billing-Cycles > 1
Then Profitable-Customer? = No
[Deny Credit Card application]

If Other-Delinquent-Accounts = 0, and (Income > $30k) OR (Years-of-Credit > 3)
Then Profitable-Customer? = Yes
[Accept Credit Card application]
Problems Too Difficult to Program by Hand

- ALVINN [Pomerleau] drives 70 mph on highways

![Diagram of neural network with inputs and outputs]
Where Is this Headed?

► Today: tip of the iceberg
  ▶ First-generation algorithms: neural nets, decision trees, regression ...
  ▶ Applied to well-formated database
  ▶ Budding industry

► Opportunity for tomorrow: enormous impact
  ▶ Learn across full mixed-media data
  ▶ Learn across multiple internal databases, plus the web and newsfeeds
  ▶ Learn by active experimentation
  ▶ Learn decisions rather than predictions
  ▶ Cumulative, lifelong learning
  ▶ Programming languages with learning embedded
Relevant Disciplines

- Artificial intelligence
- Bayesian methods
- Computational complexity theory
- Control theory
- Information theory
- Philosophy
- Psychology and neurobiology
- Statistics
- ...
What is the Learning Problem?

> Learning = Improving with experience at some task.

- Improve over task $T$,
- with respect to performance measure $P$,
- based on experience $E$.

**Example** Learn to play checkers:

- $T$: play checkers,
- $P$: percentage of games won in world tournament,
- $E$: opportunity to play against itself.

**Example** Learning to drive:

- $T$: driving on public four-lane highway using vision sensors,
- $P$: average distance travelled before an error,
- $E$: a sequence of images and steering commands recorded while observing a human driver.
Learning to Play Checkers

- $T$: play checkers
- $P$: percentage of games won in world tournament

- What experience?
- What exactly should be learned?
- How shall it be represented?
- What specific algorithm to learn it?
Type of Training Experience

- Direct or indirect?
- The problem of credit assignment.
- Teacher or not?
- **Problem** Is training experience representative of performance goal?
Choose the Target Function

- $\text{choosemove} : \text{Board} \rightarrow \text{Move}$
- $\nu : \text{Board} \rightarrow \mathbb{R}$
- ...

Non-Monotonic Reasoning (30th January 2008)
Possible Definition for Target Function $v$

- If $b$ is a final board state that is won, then $v(b) = 100$.
- If $b$ is a final board state that is lost, then $v(b) = -100$.
- If $b$ is a final board state that is drawn, then $v(b) = 0$.
- If $b$ is a not a final state in the game, then $v(b) = v(b')$, where $b'$ is the best final board state that can be achieved starting from $b$ and playing optimally until the end of the game.

⚠️ This gives correct values, but is not operational.

⚠️ Ultimate goal: Find an operational description of the ideal target function $v$.

⚠️ But we can often only acquire some approximation $\hat{v}$. 
Choose Representation for Target Function

- Collection of rules?
- Artificial neural network?
- Polynomial function of board features?
- ...
A Representation for Learned Function

\[ w_0 + w_1 \cdot bp(b) + w_2 \cdot rp(b) + w_3 \cdot bk(b) + w_4 \cdot rk(b) + w_5 \cdot bt(b) + w_6 \cdot rt(b) \]

- \( bp(b) \): number of black pieces on board \( b \).
- \( rp(b) \): number of red pieces on \( b \).
- \( bk(b) \): number of black kings on \( b \).
- \( rk(b) \): number of red kings on \( b \).
- \( bt(b) \): number of red pieces threatened by black (i.e., which can be taken on black’s next turn).
- \( rt(b) \): number of black pieces threatened by red.
Obtaining Training Examples

- \( v(b) \): the true target function
- \( \hat{v}(b) \): the learned function
- \( v_{\text{train}}(b) \): the training value
- One rule for estimating training values:
  \[ v_{\text{train}}(b) \leftarrow \hat{v}(\text{successor}(b)) \]
Choose Weight Tuning Rule

- Let \( D = \{ \langle b, v_{train}(b) \rangle \} \) be the set of training examples.
- Error on training examples:

\[
E = \sum_{\langle b, v_{train}(b) \rangle \in D} (v_{train}(b) - \hat{v}(b))^2.
\]

- Task: minimize \( E \).
- Do repeatedly:
  - Select a training example \( b \) at random.
  - Compute
    \[
    error(b) = v_{train}(b) - \hat{v}(b).
    \]
  - For each board feature \( f_i \), update weight \( w_i \):
    \[
    w_i := w_i + c \cdot f_i \cdot error(b)
    \]
    where \( c \) is some small constant, say 0.1, to moderate the rate of learning.
Design Choices

- Determine Type of Training Experience
  - Games against experts
  - Games against self
  - Table of correct moves

- Determine Target Function
  - Board
    - move
  - Board
    - value

- Determine Representation of Learned Function
  - Polynomial
  - Linear function of six features
  - Artificial neural network

- Determine Learning Algorithm
  - Gradient descent
  - Linear programming

Completed Design
Some Issues in Machine Learning

► What algorithms can approximate functions well (and when)?
► How does the number of training examples influence accuracy?
► How does noisy data influence accuracy?
► What are the theoretical limits of learnability?
► How can prior knowledge help?
► What clues can we get from biological learning systems?
► How can systems alter their own representations?
14.2 Concept Learning

- Learning from examples
- General-to-specific ordering over hypotheses
- Version spaces and candidate elimination algorithm
- Picking new examples
- The need for an inductive bias
- Note
  This is a simple approach assuming no noise and illustrating key concepts.
A Concept Learning Task

- **Concept learning** the process of inferring a boolean-valued function, i.e., a predicate, from training data of its input and output.

- **Example** Target concept: “days on which my friend enjoys sport”.

  ▶ Attributes, i.e., variables like *Sky, Temp, Humid, Wind, Water, Forecast*.

  ▶ Training examples for *enjoysport* / 6:

<table>
<thead>
<tr>
<th>Sky</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>Water</th>
<th>Forecast</th>
<th>enjoysport</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>warm</td>
<td>normal</td>
<td>strong</td>
<td>warm</td>
<td>same</td>
<td>⊤</td>
</tr>
<tr>
<td>sunny</td>
<td>warm</td>
<td>high</td>
<td>strong</td>
<td>warm</td>
<td>same</td>
<td>⊤</td>
</tr>
<tr>
<td>rainy</td>
<td>cold</td>
<td>high</td>
<td>strong</td>
<td>warm</td>
<td>change</td>
<td>⊥</td>
</tr>
<tr>
<td>sunny</td>
<td>warm</td>
<td>high</td>
<td>strong</td>
<td>cool</td>
<td>change</td>
<td>⊤</td>
</tr>
</tbody>
</table>

- **Attributes are sorted:**

<table>
<thead>
<tr>
<th>A</th>
<th>sort(A)</th>
<th>A</th>
<th>sort(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sky</td>
<td>sunny, cloudy, rainy</td>
<td>Wind</td>
<td>strong, weak</td>
</tr>
<tr>
<td>Temp</td>
<td>warm, cold</td>
<td>Water</td>
<td>warm, cool</td>
</tr>
<tr>
<td>Humid</td>
<td>normal, high</td>
<td>Forecast</td>
<td>same, change</td>
</tr>
</tbody>
</table>
Representing Hypotheses

- Many possible representations.
- Here, a **hypothesis** is a conjunction of constraints on a given set of attributes.
- Each **constraint** is of the from $A \approx s$, where $s$ is either
  - an element of $\text{sort}(A)$,
  - the symbol `?` denoting that any element of $\text{sort}(A)$ is possible,
  - or the symbol $\emptyset$ denoting that no element of $\text{sort}(A)$ is allowed.

- Example,

  $$\text{Sky} \approx \text{sunny} \land \text{Airtemp} \approx ? \land \text{Humid} \approx ?$$
  $$\land \text{Wind} \approx \text{strong} \land \text{Water} \approx ? \land \text{Forecast} \approx \text{same}$$

  - Shorthand notation $(\text{sunny}, ?, ?, \text{strong}, ?, \text{same})$.
  - Most specific hypothesis $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$.
Prototypical Concept Learning Task

- Given
  - a set of closed n-tuples, e.g., ground instances of \((\text{Sky}, \text{Airtemp}, \text{Humidity}, \text{Wind}, \text{Water}, \text{Forecast})\),
  - a target concept, i.e., predicate name \(c/n\) like \emph{enjoysport}/6,
  - a set \(\mathcal{H}\) of hypotheses,
  - a set \(\mathcal{D}\) of (positive and negative) training data of the target concept 

\[
\{\langle t_1, c_{t_1}\rangle, \ldots, \langle t_m, c_{t_m}\rangle\}.
\]

- Determine hypothesis \(h \in \mathcal{H}\) such that \(h t = c t\) for all \(\langle t, c t\rangle \in \mathcal{D}\).

- The inductive learning hypothesis: Any hypothesis found to approximate the target function well over a sufficiently large set of training data will also approximate the target function well over other unobserved data.
Instances, Hypotheses, and the More-General-Than Ordering

Instances \( X \)

\[ x_1 = \langle \text{Sunny}, \text{Warm}, \text{High}, \text{Strong}, \text{Cool}, \text{Same} \rangle \]

\[ x_2 = \langle \text{Sunny}, \text{Warm}, \text{High}, \text{Light}, \text{Warm}, \text{Same} \rangle \]

Hypotheses \( H \)

\[ h_1 = \langle \text{Sunny}, ?, ?, \text{Strong}, ?, (?) \rangle \]

\[ h_2 = \langle \text{Sunny}, ?, ?, ?, ?, (?) \rangle \]

\[ h_3 = \langle \text{Sunny}, ?, ?, ?, \text{Cool}, (?) \rangle \]
More-General-Than Ordering – Formal Definition

- Let $h_j$ and $h_k$ be hypothesis and $\mathcal{D}$ a set of training data.

- $h_j$ is more-general-than-or-equal-to $h_k$ (written $h_j \geq h_k$) iff for all $\langle t, c_t \rangle \in \mathcal{D}$ we find that $h_k t$ implies $h_j t$.

- $h_j$ is (strictly) more-general-than $h_k$ (written $h_j > h_k$) iff $h_j \geq h_k$ and not $h_k \geq h_j$.

- $h_k$ is more-specific-than $h_j$ iff $h_j$ is more-general-than $h_k$.

- $\geq$ is a partial ordering over $\mathcal{H}$, i.e., it is reflexive, antisymmetric, transitive and usually not all pairs are ordered.
**FIND-S Algorithm**

- Initialize $h$ to be the most specific hypothesis in $\mathcal{H}$.
- For each training instance $\langle \bar{t}, \top \rangle \in \mathcal{D}$ do
  - For each constraint $A \approx s$ occurring in $\mathcal{H}$ do
    - If the constraint $A \approx s$ is satisfied by $\bar{t}$ then do nothing
    - else replace $A \approx s$ by the next more general constraint that is satisfied by $\bar{t}$.
- Output $h$. 
Hypothesis Space Search by FIND-S

Instances $X$

$\bullet x_3$

$\bullet x_4$

$\bullet x_0$

$\bullet x_2$

$\bullet x_1$

Hypotheses $H$

$h_0 = \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$

$h_1 = \langle \text{sunny warm normal strong warm same} \rangle$

$h_2 = \langle \text{sunny warm ? strong warm same} \rangle$

$h_3 = \langle \text{sunny warm ? strong warm same} \rangle$

$h_4 = \langle \text{sunny warm ? strong cool change} \rangle$

$x_1 = \langle \text{sunny warm normal strong warm same} \rangle$, +

$x_2 = \langle \text{sunny warm high strong warm same} \rangle$, +

$x_3 = \langle \text{rainy cold high strong warm change} \rangle$, -

$x_4 = \langle \text{sunny warm high strong cool change} \rangle$, +

Non-Monotonic Reasoning (30th January 2008)
Complaints about FIND-S

- Can’t tell whether it has learned a concept.
- Can’t tell when training data inconsistent.
- Picks a most specific \( h \).
- Depending on \( \mathcal{H} \), there might be several most specific hypothesis!

- Is it possible to describe all hypothesis consistent with the training data?
- Version spaces and the candidate-elimination algorithm.
Version Spaces

- **Idea:** Compute the set of all hypothesis consistent with the training examples.
- A hypothesis $h$ is **consistent** with a set of training data $D$ of target concept $c$ iff $h \bar{t} = \bar{c}$ for each training example $\langle \bar{t}, \bar{c} \rangle \in D$, i.e.,

$$consistent(h, D) \leftrightarrow (\forall \langle \bar{t}, \bar{c} \rangle \in D) \ h \bar{t} = \bar{c}.$$

- The **version space** $VS_{\mathcal{H},D}$ with respect to hypothesis space $\mathcal{H}$ and training data $D$ is the subset of hypotheses from $\mathcal{H}$ consistent with all training data in $D$, i.e.,

$$VS_{\mathcal{H},D} \leftrightarrow \{ h \in \mathcal{H} \mid consistent(h, D) \}$$

- How can we represent a version space?
The **List-Then-Eliminate** Algorithm:

- Let $\text{VersionSpace}$ be a list containing every hypothesis in $\mathcal{H}$.
- For each training example $\langle \bar{t}, c\bar{t} \rangle$ do
  - remove from $\text{VersionSpace}$ any hypothesis $h$ for which $h\bar{t} \neq c\bar{t}$.
- Output the list of hypotheses in $\text{VersionSpace}$.
- We need to find a more compact representation.
Example Version Space

- **Recall**

<table>
<thead>
<tr>
<th>Sky</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>Water</th>
<th>Forecast</th>
<th>enjoysport</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>warm</td>
<td>normal</td>
<td>strong</td>
<td>warm</td>
<td>same</td>
<td>yes</td>
</tr>
<tr>
<td>sunny</td>
<td>warm</td>
<td>high</td>
<td>strong</td>
<td>warm</td>
<td>same</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>cold</td>
<td>high</td>
<td>strong</td>
<td>warm</td>
<td>change</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>warm</td>
<td>high</td>
<td>strong</td>
<td>cool</td>
<td>change</td>
<td>yes</td>
</tr>
</tbody>
</table>

- **Its version space:**

S: {<Sunny, Warm, ?, Strong, ?, ?>}
Representing Version Spaces

▶ The general boundary $\mathcal{H}_G$ of version space $\text{VS}_{\mathcal{H},\mathcal{D}}$ is the set of its most general members.
▶ The specific boundary $\mathcal{H}_S$ of version space $\text{VS}_{\mathcal{H},\mathcal{D}}$ is the set of its most specific members.
▶ Every member of the version space lies between these boundaries:

$$\text{VS}_{\mathcal{H},\mathcal{D}} = \{ h \in \mathcal{H} \mid (\exists h_S \in \mathcal{H}_S)(\exists h_G \in \mathcal{H}_G) (h_G \geq h \geq h_S) \}.$$
Candidate Elimination Algorithm

- Initialize $\mathcal{H}_G$ to be the set of most general hypotheses in $\mathcal{H}$.
- Initialize $\mathcal{H}_S$ to be the set of most specific hypotheses in $\mathcal{H}$.

- For each training example $\langle \bar{t}, c\bar{t} \rangle \in D$ do:
  - If $c\bar{t} = \top$ then do:
    - Remove from $\mathcal{H}_G$ any hypothesis not consistent with $\bar{t}$.
    - For each hypothesis $h_S \in \mathcal{H}_S$ that is not consistent with $\bar{t}$ do:
      - Remove $h_S$ from $\mathcal{H}_S$.
      - Add to $\mathcal{H}_S$ all minimal generalizations $h$ of $h_S$ such that
        - $h$ is consistent with $\bar{t}$, and
        - some member of $\mathcal{H}_G$ is more general than $h$.
    - Remove from $\mathcal{H}_S$ any hypothesis that is more general than another hypothesis in $\mathcal{H}_S$. 

Non-Monotonic Reasoning (30th January 2008)
Candidate Elimination Algorithm – Continued

▶ Remember: For each training example \( ⟨\bar{t}, c\bar{t}⟩ \in D \) do:
  
  ▶ If \( c\bar{t} = \bot \) then do:
    
    ▶ Remove from \( H_S \) any hypothesis not consistent with \( \bar{t} \).
    
    ▶ For each hypothesis \( h_G \in H_G \) that is not consistent with \( \bar{t} \) do:
      
      ▶▶ Remove \( h_G \) from \( H_G \)
      
      ▶▶ Add to \( H_G \) all minimal specializations \( h \) of \( h_G \) such that
        
        ▶▶▶ \( h \) is consistent with \( \bar{t} \), and
        
        ▶▶▶ some member of \( H_S \) is more specific than \( h \).
        
      ▶▶ Remove from \( H_G \) any hypothesis that is less general than another hypothesis in \( H_G \).
Example Trace

- **Final version space:**

  \[
  S: \{ \langle \text{Sunny}, \text{Warm}, ?, ? \rangle \} \\
  \langle \text{Sunny}, ?, ?, \text{Strong}, ?, ? \rangle \\
  \langle ?, \text{Warm}, ?, \text{Strong}, ?, ? \rangle \\
  \langle ?, \text{Warm}, ?, \text{Strong}, ?, ? \rangle \\
  \{ \} \\
  G: \{ \langle \text{Sunny}, ?, ?, ?, ?, \rangle, \langle ?, \text{Warm}, ?, ?, ?, \rangle \} \\
  \]

- **How should new instances be classified?**

  - (sunny, warm, normal, strong, cool, change) \( \leadsto \) ⊤
  - (rainy, cold, normal, light, warm, same) \( \leadsto \) ⊥
  - (sunny, warm, normal, light, warm, same) \( \leadsto \) don’t know
  - (sunny, cold, normal, strong, warm, same) \( \leadsto \) don’t know
An Un-Biased Learner

▶ Idea Choose $\mathcal{H}$ that expresses every teachable concept.

▶ Let $\mathcal{H}'$ be all hypothesis which can be formed by disjunctions, conjunctions and negations over $\mathcal{H}$. E.g.,

$$(Sky \approx \text{sunny} \land Airtemp \approx \text{warm} \land Humid \approx \text{normal}$$
$$\land Wind \approx ? \land Water \approx ? \land Forecast \approx ?)$$
$$\lor \neg(Sky \approx ? \land Airtemp \approx ? \land Humid \approx ?$$
$$\land Wind \approx ? \land Water \approx ? \land Forecast \approx \text{change})$$

▶ What are $\mathcal{H}_S$, $\mathcal{H}_G$ in this case?

▶ $\mathcal{H}_S$ will consist of all positive examples.

▶ $\mathcal{H}_G$ will consist of all negative examples.

▶ No generalization.
Inductive Bias

- Consider learning algorithm $L$ for a set of closed tuples $\mathcal{X}$.
- Let $\mathcal{D}_c$ be a set training data for concept $c/n$.
- Let $L(s, \mathcal{D}_c)$ denote the classification of $s \in \mathcal{X}$ by $L_c$ after training on $\mathcal{D}$.
- The inductive bias of $L$ is any minimal set of assertions $\mathcal{A}$ such that for any target concept $c/n$ and corresponding training data $\mathcal{D}_c$ we find

$$\mathcal{A} \cup \{ct \mid \langle t, \top \rangle \in \mathcal{D}_c\} \cup \{\neg ct \mid \langle t, \bot \rangle \in \mathcal{D}_c\} \models L(s, \mathcal{D}_c)$$

for all $s \in \mathcal{X}$. 
Inductive Systems and Equivalent Deductive Systems

Inductive system

Training examples

Candidate Elimination Algorithm

Using Hypothesis Space $H$

New instance

Classification of new instance, or "don’t know"

Equivalent deductive system

Training examples

Theorem Prover

New instance

Classification of new instance, or "don’t know"

Assertion "$H$ contains the target concept"

Inductive bias made explicit
Three Learners with Different Biases

► Rote learner:
  Store examples, classify \( \overline{t} \) iff it matches previously observed example.
  ▶ No inductive bias.

► Version space candidate elimination algorithm
  ▶ Inductive bias: Target concept can be represented in its hypothesis space.

► Find-S
  ▶ Inductive bias: Target concept can be represented in its hypothesis space and all instances are negative instances unless the opposite is entailed by its other knowledge.
Summary Points

- Concept learning as search through $\mathcal{H}$.
- General-to-specific ordering over $\mathcal{H}$.
- Version space candidate elimination algorithm.
- $\mathcal{H}_S$ and $\mathcal{H}_G$ boundaries characterize learner’s uncertainty.
- Learner can generate useful queries.
- Inductive leaps possible only if learner is biased.
- Inductive learners can be modelled by equivalent deductive systems.
14.3 Learning Sets of Rules – Overview

- Sequential covering algorithms
- FOIL
- Induction as inverse of deduction
- Inductive Logic Programming
Learning Disjunctive Sets of Rules

- **Idea** represent target concept by a set of if-then-rules.
- Various methods, e.g., learn decision tree and convert it into rules,
  - for details see literature.
- **Sequential covering algorithm**
  - learn one rule with high accuracy, any coverage,
  - remove positive training data covered by this rule,
  - repeat.
Sequential Covering Algorithm

- **SEQUENTIAL-COVERING**(*Target_concept*, *Attributes*, *Data*, *Threshold*)
  - $\text{Learned\_rules} \gets \emptyset$.
  - $\text{Rule} \gets \text{LEARN-ONE-RULE}(*Target\_concept*, *Attributes*, *Data*)$.
  - while $\text{PERFORMANCE}(\text{Rule}, \text{Data}) > \text{Threshold}$ do:
    - $\text{Learned\_rules} \gets \text{Learned\_rules} \cup \{\text{Rule}\}$.
    - $\text{Data} \gets \text{Data} \setminus \{\text{data correctly classified by Rule}\}$.
    - $\text{Rule} \gets \text{LEARN-ONE-RULE}(*Target\_concept*, *Attributes*, *Data*)$.
  - $\text{Learned\_rules} \gets \text{sort Learned\_rules accord to PERFORMANCE over Data}$
  - return $\text{Learned\_rules}$.
Example: Learning the Concept *playtennis*

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>playtennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>weak</td>
<td>⊥</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>strong</td>
<td>⊥</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>high</td>
<td>weak</td>
<td>⊤</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>weak</td>
<td>⊤</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>weak</td>
<td>⊤</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>strong</td>
<td>⊥</td>
</tr>
<tr>
<td>overcast</td>
<td>cool</td>
<td>normal</td>
<td>strong</td>
<td>⊤</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>high</td>
<td>weak</td>
<td>⊥</td>
</tr>
<tr>
<td>sunny</td>
<td>cool</td>
<td>normal</td>
<td>weak</td>
<td>⊤</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>normal</td>
<td>weak</td>
<td>⊤</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>normal</td>
<td>strong</td>
<td>⊤</td>
</tr>
<tr>
<td>overcast</td>
<td>mild</td>
<td>high</td>
<td>strong</td>
<td>⊤</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>normal</td>
<td>weak</td>
<td>⊤</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>strong</td>
<td>⊥</td>
</tr>
</tbody>
</table>
Learn-One-Rule

IF Wind=weak
THEN PlayTennis=yes

IF Wind=strong
THEN PlayTennis=no

IF Humidity=normal
THEN PlayTennis=no

IF Humidity=normal
Outlook=sunny
THEN PlayTennis=yes

IF Humidity=normal
Outlook=rain
THEN PlayTennis=yes

IF Humidity=high
THEN PlayTennis=no

IF Humidity=high
Wind=weak
THEN PlayTennis=yes

If Humidity=high
Wind=strong
THEN PlayTennis=yes

IF Humidity=high

...
Variants of Rule Learning Programs

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

- Can learn sets of rules such as:
  
  \[
  \text{ancestor}(X, Y) \leftarrow \text{parent}(X, Y) \\
  \text{ancestor}(X, Y) \leftarrow \text{parent}(X, Z) \land \text{ancestor}(Z, Y)
  \]

- Logic programs are sets of such rules.
- Here the only function symbols are constants.
  - datalogic,
  - still propositional in nature!
FOIL\((Target\_predicate, \ Predicates, \ Data)\)

\[\begin{align*}
\text{Pos} & := \text{positive Data}, \ \text{Neg} := \text{negative Data}, \ \text{Learned\_rules} := \emptyset. \\
\text{while Pos do:} \\
\text{Learn a NewRule :} \\
& \quad \text{NewRule} := \text{most general rule possible, NewRuleNeg} := \text{Neg} \\
& \quad \text{while NewRuleNeg do:} \\
& \quad \quad \text{Add a new literal to specialize NewRule :} \\
& \quad \quad \quad \text{Candidate\_literals} := \text{generate candidates} \\
& \quad \quad \quad \text{Best\_literal} := \arg\max_{L \in \text{Candidate\_literals}} \text{Foil\_Gain}(L, \text{NewRule}) \\
& \quad \quad \quad \text{add Best\_literal to the preconditions of NewRule} \\
& \quad \quad \text{NewRuleNeg} \\
& \quad \quad \quad := \text{subset of NewRuleNeg that satisfies NewRule preconditions} \\
& \quad \text{Learned\_rules} := \text{Learned\_rules} \cup \{\text{NewRule}\} \\
& \quad \text{Pos} := \text{Pos} \setminus \{\text{members of Pos covered by NewRule}\} \\
\text{Return Learned\_rules}
\end{align*}\]
Specializing Rules in FOIL

- Learning rule: \( p(X_1, \ldots, X_k) \leftarrow L_1 \land \ldots \land L_n \).
- Candidate specializations add new literal \( L_i \) of form:
  - \( p(V_1, \ldots, V_r) \), where at least one of the \( V_i \) must already exist as a variable in the rule,
  - \( X_j \approx X_k \), where \( X_j \) and \( X_k \) are variables already present in the rule,
  - and the negation of either of the above forms of atoms.
Learning of the target \textit{granddaughter}

\begin{itemize}
  \item \textit{Target\_predicate} := \textit{granddaughter}, \textit{Predicates} := \textit{female}, \textit{father}.

  \textbullet\ Most general rule:
  \[
  \text{granddaughter}(X, Y) \leftarrow
  \]

  \textbullet\ \textit{Candidate\_literals} := \[X \approx Y, \ female(X), \ female(Y), \]
  \[
  \text{father}(X, Y), \ text{father}(Y, X), \ text{father}(X, Z),
  \]
  \[
  \text{father}(Z, X), \ text{father}(Y, Z), \ text{father}(Z, Y),
  \]
  \[
  \text{and negations thereof.}
  \]

  \item More specific rule:
  \[
  \text{granddaughter}(X, Y) \leftarrow \text{father}(Y, Z)
  \]

  \textbullet\ \textit{Candidate\_literals} := \[Z \approx X, \ Z \approx Y, \ female(Z), \]
  \[
  \text{father}(Z, W), \ text{father}(W, Z), \ text{father}(X, W),
  \]
  \[
  \text{father}(W, X), \ text{father}(Y, W), \ text{father}(W, Y),
  \]
  \[
  \text{and negations thereof plus the abovementioned ones.}
  \]

  \item Eventually we reach:
  \[
  \text{granddaughter}(X, Y) \leftarrow \text{father}(Y, Z), \ text{father}(Z, X), \ text{female}(Y)
  \]
\end{itemize}
Positive and Negative Bindings

- **Data**:\[ \text{granddaughter}(victor, shannon), \text{female}(shannon), \text{father}(shannon, bob), \text{father}(tom, bob), \text{father}(bob, victor) \]

- We are making the closed world assumption!

- Reconsider
  \[ \text{granddaughter}(X, Y) \leftarrow \]

- There are \(2^4 = 16\) different bindings with domain \(\{X, Y\}\) and constants \(victor, shannon, bob, tom\).

- Only \(\{X \mapsto victor, Y \mapsto shannon\}\) yields a positive example, the others yield negative examples.

- Hence, \(\{X \mapsto victor, Y \mapsto shannon\}\) is called **positive binding**, all others are called **negative bindings**.

- FOIL prefers rules with more positive bindings.
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

- \(L\) := 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 \(R + L\)
- \(t\) := number of positive bindings of \(R\) also covered by \(R + L\)

\[\Box\] Note

- \(- \log_2 \frac{p_0}{p_0+n_0}\) is minimum number of bits needed to encode a positive binding covered by \(R\).
- \(- \log_2 \frac{p_1}{p_1+n_1}\) is minimum number of bits needed to encode a positive binding covered by \(R + L\).

\[\Box\] \text{Foil\_Gain}(L, R) is the reduction due to \(L\) in the total number of bits needed to encode the classification of all positive bindings for \(R\).
Another FOIL Example

Instances:

- pairs of nodes describing graph using predicate `linkedto`.

Target function:

- `canreach(X, Y)` is true iff there is a directed path from `X` to `Y`.

Hypothesis space:

- Each `H ∈ ℋ` is a set of Horn clauses built over `linkedto` and `canreach`.
Induction as Inverted Deduction

- Induction is finding $H \in \mathcal{H}$ such that for all $\langle G, c(G) \rangle \in \mathcal{D}$ we find

$$\mathcal{B} \cup \{H, G\} \models c(G)$$

where

- $G$ is a formula,
- $c(G)$ is the target formula for $G$,
- $\mathcal{B}$ is background knowledge.

- Idea design inductive algorithm by inverting operators for automated deduction!
Example

- Target predicate: ground instances of \( \text{child}(X, Y) \).

- Let

\[
G = \text{male}(\text{bob}) \land \text{female}(\text{shannon}) \land \text{father}(\text{shannon}, \text{bob}),
\]
\[
c(G) = \text{child}(\text{bob}, \text{shannon}),
\]
\[
\mathcal{B} = \{ \text{parent}(U, V) \leftarrow \text{father}(U, V) \}.
\]

- What satisfies for \( \langle G, c(G) \rangle \in \mathcal{D} \) the relation \( \mathcal{B} \cup \{ H, G \} \models c(G) \)?

- Among others:

\[
H_1 = \text{child}(U, V) \leftarrow \text{father}(V, U),
\]
\[
H_2 = \text{child}(U, V) \leftarrow \text{parent}(V, U).
\]
Induction

- [Jevons 1874] 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. . . .
The Search for Inductive Operators

- We have mechanical **deductive** operators $\vdash$ such that

$$\mathcal{F} \cup \mathcal{B} \vdash G$$

whenever $\mathcal{F} \cup \mathcal{B} \models G$.

- We need **inductive** operators $\models$ such that

$$\mathcal{B} \cup \mathcal{D} \models H$$

whenever for all $\langle G, c(G) \rangle \in \mathcal{D}$ we find $\mathcal{B} \cup \{H, G\} \models c(G)$. 
Features

▶ Positive

▷ Subsumes earlier idea of finding $H$ that “fits” training data.
▷ Background knowledge $B$ helps define meaning of “fit” the data.
▷ Suggests algorithms that search $H$ guided by $B$.

▶ Negative

▷ Doesn’t allow for noisy data.
▷ First order logic gives a huge hypothesis space $H$.

► Overfitting.
► Intractability of calculating all acceptable hypothesis.
Resolution and Inverted Resolution (Propositional)

▶ Here we consider clauses as sets of literals, where
\[ \neg L = \begin{cases} \neg A & \text{if } L = A \\ A & \text{if } L = \neg A \end{cases} \]

▶ Resolution

▷ Given clauses \( C_1 \) and \( C_2 \) such that there exists \( L \in C_1 \) with \( \neg L \in C_2 \).
▷ Resolvent \( C = (C_1 \setminus \{L\}) \cup (C_2 \setminus \{\neg L\}) \).

▶ Examples

▷ \( C_1 = \{\text{passexam}, \neg \text{knowmaterial}\} \), \( C_2 = \{\text{knowmaterial}, \neg \text{study}\} \), \( L = \neg \text{knowmaterial} \), \( C = \{\text{passexam}, \neg \text{study}\} \).

▶ Inverted Resolution

▷ Given \( C_1 \) and \( C \) such that \( L \in C_1 \) but \( L \not\in C \) and \( C_1 \setminus \{L\} \subseteq C \).
▷ Compute \( C_2 = (C \setminus (C_1 \setminus \{L\})) \cup \{\neg L\} \).
First Order Resolution

- Given clauses \( C_1 \) and \( C_2 \).
- Find \( L_1 \in C_1, L_2 \in C_2 \) and \( \theta \) such that \( \theta \) is an mgu of \( L_1 \) and \( \neg L_2 \).
- Resolvent \( C = (C_1 \setminus \{L_1\})\theta \cup (C_2 \setminus \{L_2\})\theta \).

- Clauses are standardized apart:
  - \( \text{Var}(C_1) \cap \text{Var}(C_2) = \emptyset \),
  - \( \text{Dom}(\theta) = \text{Var}(C_1) \cup \text{Var}(C_2) \),
  - \( \theta_1 = \theta|\text{Var}(C_1) \),
  - \( \theta_2 = \theta|\text{Var}(C_2) \),
  - \( \theta = \theta_1 \cup \theta_2 \),
  - \( C = (C_1 \setminus \{L_1\})\theta_1 \cup (C_2 \setminus \{L_2\})\theta_2 \).
Inverse Substitution

- Given atom $A$. An inverse substitution $\theta^{-1}$ of a substitution $\theta$ is a mapping from terms occurring in $A\theta$ to variables such that $A\theta\theta^{-1} = A$.

Examples

1. $A = \text{daughter}(X, Y), \theta = \{X \mapsto \text{mary}, Y \mapsto \text{ann}\}$

   $\theta^{-1} = \{\text{mary} \mapsto X, \text{ann} \mapsto Y\}$.

2. $A = \text{loves}(X, Y), \theta = \{X \mapsto \text{ann}, Y \mapsto \text{ann}\}$

   $\theta^{-1} = ?$

Solution  add positions to distinguish different occurrences of the same term:

2'. $A = \text{loves}(X, Y), \theta = \{X \mapsto \text{ann}, Y \mapsto \text{ann}\}$:

   $\theta^{-1} = \{(\text{ann}, 1) \mapsto X, (\text{ann}, 2) \mapsto Y\}$.

Such substitutions are called inverse substitutions with positions.
Inverse First-Order Resolution

- Given $C$ and $C_1$. Find $C_2$ assuming that $C_1$ and $C_2$ have no literals in common.

Recall

$$C = (C_1 \setminus \{L_1\})\theta_1 \cup (C_2 \setminus \{L_2\})\theta_2.$$  

- But then

$$C \setminus (C_1 \setminus \{L_1\})\theta_1 = (C_2 \setminus \{L_2\})\theta_2.$$  

- Hence,

$$C_2 = (C \setminus (C_1 \setminus \{L_1\})\theta_1)\theta_2^{-1} \cup \{\neg L_1\theta_1\theta_2^{-1}\}.$$  

is called inverse resolvent of $C$ and $C_1$.

- Non-determinism: Given $C$ as observed positive example.

  - Which clause from background knowledge $B$ shall we select as $C_1$?
  - Which literal from $C_1$ shall we select to resolve upon?
  - Which inverse substitution $\theta_2^{-1}$ shall we choose?
Example \( C_2 = (C \setminus (C_1 \setminus \{L_1\}))\theta_1)\theta_2^{-1} \cup \{\neg L_1 \theta_1 \theta_2^{-1}\} \)

- Let \( B = \{\{\text{female (mary)}\}, \{\text{parent (ann, mary)}\}\} \).

- Let \( C = \{\text{daughter (mary, ann)}\} \) and \( C_1 = \{\text{parent (ann, mary)}\} \).
  - \( L_1 = \text{parent (ann, mary)} \).
  - \( \theta_1 = \epsilon \).
  - Choose \( \theta_2^{-1} = \{\text{ann }\mapsto Y\} \).
  - We obtain \( C_2 = \{\text{daughter (mary, Y), } \neg \text{parent (Y, mary)}\} \).
  - Observe, \( B \cup \{C_2\} \models \{\text{daughter (mary, ann)}\} \).

- Let \( C = \{\text{daughter (mary, Y), } \neg \text{parent (Y, mary)}\} \) and \( C_1 = \{\text{female (mary)}\} \).
  - \( L_1 = \text{female (mary)} \).
  - \( \theta_1 = \epsilon \).
  - Choose \( \theta_2^{-1} = \{\text{mary }\mapsto X\} \).
  - We obtain \( C_2 = \{\text{daughter (X, Y), } \neg \text{parent (Y, X), } \neg \text{female (X)}\} \).
  - Observe, \( B \cup \{C_2\} \models \{\text{daughter (mary, ann)}\} \).
Most Specific Inverse Resolution

- **Idea** Use the most specific inverse substitution.

- Let \( B = \{\{\text{female (mary)}\}, \{\text{parent (ann, mary)}\}\} \).

  - Let \( C = \{\text{daughter (mary, ann)}\} \) and \( C_1 = \{\text{parent (ann, mary)}\} \).

    - \( L_1 = \text{parent (ann, mary)} \).
    - \( \theta_1 = \varepsilon \).
    - Choose \( \theta_2^{-1} = \varepsilon \).
    - We obtain \( C_2 = \{\text{daughter (mary, ann)}, \neg\text{parent (ann, mary)}\} \).
    - Observe, \( B \cup \{C_2\} \models \{\text{daughter (mary, ann)}\} \).
Most Specific Inverse Resolution – Continued

Let $\mathcal{B} = \{ \{ \text{female}(\text{mary}) \}, \{ \text{parent}(\text{ann}, \text{mary}) \} \}$.

Let $C = \{ \text{daughter}(\text{mary}, \text{ann}), \neg \text{parent}(\text{ann}, \text{mary}) \}$
and $C_1 = \{ \text{female}(\text{mary}) \}$.

$L_1 = \text{female}(\text{mary})$.

$\theta_1 = \varepsilon$.

Choose $\theta_2^{-1} = \varepsilon$.

We obtain $C_2 = \{ \text{daughter}(\text{mary}, \text{ann}), \neg \text{parent}(\text{ann}, \text{mary}), \neg \text{female}(\text{mary}) \}$.

Observe, $\mathcal{B} \cup \{ C_2 \} \models \{ \text{daughter}(\text{mary}, \text{ann}) \}$.
A Unifying Framework for Generalization

- Let $\mathcal{B} = \{ \{ \text{female} (\text{mary}) \}, \{ \text{parent} (\text{ann}, \text{mary}) \}, \{ \text{female} (\text{eve}) \}, \{ \text{parent} (\text{tom}, \text{eve}) \} \}$. 

- Training Data: $\{ \text{daughter} (\text{mary}, \text{ann}) \}, \{ \text{daughter} (\text{eve}, \text{tom}) \}$. 

- Using most specific inverse resolution we obtain:

$$\{ \text{daughter} (\text{mary}, \text{ann}), \neg \text{parent} (\text{ann}, \text{mary}), \neg \text{female} (\text{mary}) \},$$
$$\{ \text{daughter} (\text{eve}, \text{tom}), \neg \text{parent} (\text{tom}, \text{eve}), \neg \text{female} (\text{eve}) \}.$$ 

- Compute the so-called least general generalization:

$$\{ \text{daughter} (X, Y), \neg \text{parent} (Y, X), \neg \text{female} (X) \}.$$ 

- What is the least general generalization?
\(\theta\)-Subsumption

- Let \(C_1\) and \(C_2\) be two clauses.
  - \(C_1\) \(\theta\)-subsumes \(C_2\) if there exists a substitution \(\theta\) such that \(C_1 \theta \subseteq C_2\).
  - A clause is reduced if it is not \(\theta\)-subsumption equivalent to any proper subset of itself.

**Example:** Consider

\[
C_1 = \{ \text{daughter}(X, Y), \neg \text{parent}(Y, X) \} \\
C_2 = \{ \text{daughter}(\text{mary}, Y), \neg \text{female}(\text{mary}), \neg \text{parent}(Y, \text{mary}) \}
\]

\(C_1\) \(\theta\)-subsumes \(C_2\) under \(\theta = \{ X \mapsto \text{mary} \} \).

- \(C_1\) is at least as general as \(C_2\), in symbols \(C_1 \leq C_2\), if \(C_1\) \(\theta\)-subsumes \(C_2\).
- \(C_1\) is more general than \(C_2\), in symbols \(C_1 < C_2\), if \(C_1 \leq C_2\) and \(C_2 \not< C_1\).
  - \(C_1\) is a generalization of \(C_2\).
  - \(C_2\) is a specialization of \(C_1\).
\(\theta\)-Subsumption: Properties

- If \(C_1 \leq C_2\) then \(\{C_1\} \models C_2\).
- \(\leq\) induces a lattice on the set of reduced clauses.
  - \(\text{lub}\) and \(\text{glb}\) always exist
  - and are unique modulo variable renaming.
- The least general generalization of two reduced clauses \(C_1\) and \(C_2\), in symbols \(\text{lgg}(C_1, C_2)\), is the \(\text{lub}\) of \(C_1\) and \(C_2\) in the \(\theta\)-subsumption lattice.
- \(\theta\)-subsumption provides a generality ordering for hypothesis.
- It can be used to prune the search space.
Computing Least General Generalizations

▶ Terms

- \( lgg(t, t) = t \).
- \( lgg(f(s_1, \ldots, s_n), f(t_1, \ldots, t_n)) = f(lgg(s_1, t_1), \ldots, lgg(s_n, t_n)) \).
- \( lgg(f(s_1, \ldots, s_n), g(t_1, \ldots, t_m)) = V \), where \( f \neq g \).
- \( lgg(s, t) = V \), where \( s \neq t \), at least one of \( s \) and \( t \) is a variable.

▶ Atoms

- \( lgg(p(s_1, \ldots, s_n), p(t_1, \ldots, t_n)) = p(lgg(s_1, t_1), \ldots, lgg(s_n, t_n)) \).
- \( lgg(p(s_1, \ldots, s_n), q(t_1, \ldots, t_m)) \) is undefined if \( p \neq q \).

▶ Negated atoms

- \( lgg(\neg p(s_1, \ldots, s_n), \neg p(t_1, \ldots, t_n)) = \neg p(lgg(s_1, t_1), \ldots, lgg(s_n, t_n)) \).
- \( lgg(\neg p(s_1, \ldots, s_n), q(t_1, \ldots, t_m)) \) is undefined.

▶ Clauses

- Let \( C_1 = \{ L_1, \ldots, L_n \} \) and \( C_2 = \{ K_1, \ldots, K_m \} \) be two clauses.
- \( lgg(C_1, C_2) = \{ lgg(L_i, K_j) \mid L_i \in C_1, K_j \in C_2, lgg(L_i, K_j) \) defined\}. 

Further Reading