**Inductive Learning (1/2)**

**Decision Tree Method**

(If it’s not simple, it’s not worth learning it)

R&N: Chap. 18, Sect. 18.1-3

---

**Motivation**

- An AI agent operating in a complex world requires an awful lot of knowledge: state representations, state axioms, constraints, action descriptions, heuristics, probabilities, ...

- More and more, AI agents are designed to acquire knowledge through learning

---

**What is Learning?**

- Mostly generalization from experience:

  "Our experience of the world is specific, yet we are able to formulate general theories that account for the past and predict the future"


- Concepts, heuristics, policies
- Supervised vs. un-supervised learning

---

**Contents**

- Introduction to inductive learning
- Logic-based inductive learning:
  - Decision-tree induction
- Function-based inductive learning
  - Neural nets

---

**Logic-Based Inductive Learning**

- **Background** knowledge KB

- Training set D *(observed* knowledge) that is not logically implied by KB

- **Inductive inference:** Find $h$ such that KB and $h$ imply D

$h = D$ is a trivial, but un-interesting solution (data caching)

---

**Rewarded Card Example**

- Deck of cards, with each card designated by $[r,s]$, its rank and suit, and some cards “rewarded”

- **Background knowledge KB:**
  
  $((r=1) \lor \ldots \lor (r=10)) \Leftrightarrow \text{NUM}(r)$
  
  $((r=J) \lor (r=Q) \lor (r=K)) \Leftrightarrow \text{FACE}(r)$
  
  $((s=S) \lor (s=C)) \Leftrightarrow \text{BLACK}(s)$
  
  $((s=D) \lor (s=H)) \Leftrightarrow \text{RED}(s)$

- **Training set D:**

  \[
  \text{REWARD}([4,C]) \land \text{REWARD}([7,C]) \land \text{REWARD}([2,S]) \land
  \neg \text{REWARD}([5,H]) \land \neg \text{REWARD}([7,S])
  \]
Rewarded Card Example

- Deck of cards, with each card designated by \([r,s]\), its rank and suit, and some cards “rewarded”
- Background knowledge KB:
  \(((r=1) \lor \ldots \lor (r=10)) \iff \text{NUM}(r)\)
  \(((r=J) \lor (r=Q) \lor (r=K)) \iff \text{FACE}(r)\)
  \(((s=5) \lor (s=10)) \iff \text{BLACK}(s)\)
  \(((s=2) \lor (s=3)) \iff \text{RED}(s)\)
- Training set D:
  \text{REWARD}([4,C]) \land \text{REWARD}([7,C]) \land \text{REWARD}([2,S]) \land \lnot \text{REWARD}([5,H]) \land \lnot \text{REWARD}([J,S])\)
- Possible inductive hypothesis:
  \(h = (\text{NUM}(r) \land \text{BLACK}(s) \iff \text{REWARD}([r,s]))\)

Learning a Predicate (Concept Classifier)

- Set \(E\) of objects (e.g., cards)
- Goal predicate \(\text{CONCEPT}(x)\), where \(x\) is an object in \(E\), that takes the value True or False (e.g., \text{REWARD})
- Example:
  \(\text{CONCEPT}(x)\) describes the precondition of an action, e.g., \(\text{Unstack}(C,A)\)
  - \(E\) is the set of states
  - \(\text{CONCEPT}(x)\) \iff \(\text{HANDEMPKY} \in x, \text{BLOCK}(C) \in x, \text{BLOCK}(A) \in x, \text{CLEAR}(C) \in x, \text{ON}(C,A) \in x\)
- Learning \(\text{CONCEPT}\) is a step toward learning an action description

Example of Training Set

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play/Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D9</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>Strong</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

Example of Training Set

- Set \(E\) of objects (e.g., cards)
- Observable attributes \(A(x), B(x), \ldots\) (e.g., \text{NUM}, \text{RED})
- Training set: values of \(\text{CONCEPT}\) for some combinations of values of the observable predicates
- Find a representation of \(\text{CONCEPT}\) in the form:
  \(\text{CONCEPT}(x) \iff S(A,B,\ldots)\)
  where \(S(A,B,\ldots)\) is a sentence built with the observable predicates, e.g.:
  \(\text{CONCEPT}(x) \iff A(x) \land (\neg B(x) \lor C(x))\)

Learning a Predicate (Concept Classifier)

- Set \(E\) of objects (e.g., cards)
- Goal predicate \(\text{CONCEPT}(x)\), where \(x\) is an object in \(E\), that takes the value True or False (e.g., \text{REWARD})
- Observable predicates \(A(x), B(x), \ldots\) (e.g., \text{NUM}, \text{RED})
- Training set: values of \(\text{CONCEPT}\) for some combinations of values of the observable predicates
- Find a representation of \(\text{CONCEPT}\) in the form:
  \(\text{CONCEPT}(x) \iff S(A,B,\ldots)\)
  where \(S(A,B,\ldots)\) is a sentence built with the observable predicates, e.g.:
  \(\text{CONCEPT}(x) \iff A(x) \land (\neg B(x) \lor C(x))\)
Learning an Arch Classifier

- These objects are arches: (positive examples)

- These aren’t: (negative examples)

\[
\text{ARCH}(x) \iff \text{HAS-PART}(x, b_1) \land \text{HAS-PART}(x, b_2) \land \text{HAS-PART}(x, b_3) \land \text{IS-A}(b_1, \text{BRICK}) \land \text{IS-A}(b_2, \text{BRICK}) \land \neg \text{MEET}(b_1, b_2) \land (\text{IS-A}(b_3, \text{BRICK}) \lor \text{IS-A}(b_3, \text{WEDGE})) \land \text{SUPPORTED}(b_3, b_1) \land \text{SUPPORTED}(b_3, b_2)
\]

Example set

- An example consists of the values of CONCEPT and the observable predicates for some object \( x \)
- An example is positive if CONCEPT is True, else it is negative
- The set \( X \) of all examples is the example set
- The training set is a subset of \( X \), a small one!

Hypothesis Space

- An hypothesis is any sentence of the form:
  \[ \text{CONCEPT}(x) \iff S(A, B, \ldots) \]
  where \( S(A, B, \ldots) \) is a sentence built using the observable predicates
- The set of all hypotheses is called the hypothesis space \( H \)
- An hypothesis \( h \) agrees with an example if it gives the correct value of CONCEPT

Inductive Learning Scheme

- Deck of cards, with each card designated by \([r, s]\), its rank and suit, and some cards "rewarded"
- Background knowledge \( KB \):
  \[
  \begin{align*}
  ((r=1) \lor \ldots \lor (r=10)) & \iff \text{NUM}(r) \\
  ((r=J) \lor (r=Q) \lor (r=K)) & \iff \text{FACE}(r) \\
  ((s=S) \lor (s=C)) & \iff \text{BLACK}(s) \\
  ((s=D) \lor (s=H)) & \iff \text{RED}(s)
  \end{align*}
  \]
- Training set \( D \):
  \[
  \begin{align*}
  \text{REWARD}([4, C]) \land \text{REWARD}([7, C]) \land \text{REWARD}([2, S]) \land \neg \text{REWARD}([5, H]) \land \neg \text{REWARD}([J, S])
  \end{align*}
  \]

Size of Hypothesis Space

- \( n \) observable predicates
- \( 2^n \) entries in truth table defining \text{CONCEPT} and each entry can be filled with True or False
- In the absence of any restriction (bias), there are \( 2^n \) hypotheses to choose from
- \( n = 6 \rightarrow 2 \times 10^{19} \) hypotheses!

Multiple Inductive Hypotheses

- \( h_1 = \text{NUM}(r) \land \text{BLACK}(s) \iff \text{REWARD}([r, s]) \)
- \( h_2 = \neg (r=J) \iff \text{REWARD}([r, s]) \)
- \( h_3 = (r,s)=[4,C] \lor (r,s)=[7,C] \lor (r,s)=[2,S] \iff \text{REWARD}([r, s]) \)
- \( h_4 = (r,s)=[5,H] \lor (r,s)=[J,S] \iff \text{REWARD}([r, s]) \)

agree with all the examples in the training set
Multiple Inductive Hypotheses

* Deck of cards, with each card designated by [r,s], its rank and suit, and some cards "rewarded".

Need for a system of preferences - called a bias - to compare possible hypotheses

\[ h_1 = \text{NUM}(r) \land \text{BLACK}(s) \iff \text{REWARD}([r,s]) \]
\[ h_2 = \text{BLACK}(s) \land \neg (r=J) \iff \text{REWARD}([r,s]) \]
\[ h_3 = ([r,s]=[4,C]) \lor ([r,s]=[7,C]) \lor ([r,s]=[2,S]) \iff \text{REWARD}([r,s]) \]
\[ h_4 = \neg ([r,s]=[5,H]) \lor \neg ([r,s]=[J,S]) \iff \text{REWARD}([r,s]) \]

agree with all the examples in the training set.

Notion of Capacity

* It refers to the ability of a machine to learn any training set without error.
* A machine with too much capacity is like a botanist with photographic memory who, when presented with a new tree, concludes that it is not a tree because it has a different number of leaves from anything he has seen before.
* A machine with too little capacity is like the botanist's lazy brother, who declares that if it's green, it's a tree.
* Good generalization can only be achieved when the right balance is struck between the accuracy attained on the training set and the capacity of the machine.

Keep-It-Simple (KIS) Bias

Examples
- Use much fewer observable predicates than the training set.
- Constrain the learnt predicate, e.g., to use only "high-level" observable predicates such as NUM, FACE, BLACK, and RED and/or to have simple syntax.

Motivation
- If an hypothesis is too complex it is not worth learning it (data caching does the job as well).
- There are much fewer simple hypotheses than complex ones, hence the hypothesis space is smaller.

Keep-It-Simple (KIS) Bias

Examples
- Use much fewer observable predicates than the training set.
- Constrain the learnt predicate, e.g., to use only "high-level" observable predicates such as NUM, FACE, BLACK, and RED and/or to have simple syntax.

Motivation
- If an hypothesis is too complex it is not worth learning it (data caching does the job as well).
- There are much fewer simple hypotheses than complex ones, hence the hypothesis space is smaller.

Keep-It-Simple (KIS) Bias

Examples
- Use much fewer observable predicates than the training set.
- Constrain the learnt predicate, e.g., to use only "high-level" observable predicates such as NUM, FACE, BLACK, and RED and/or to have simple syntax.

Motivation
- If an hypothesis is too complex it is not worth learning it (data caching does the job as well).
- There are much fewer simple hypotheses than complex ones, hence the hypothesis space is smaller.

Putting Things Together

Einstein: "A theory must be as simple as possible, but not simpler than this."
Predicate as a Decision Tree

The predicate \( \text{CONCEPT}(x) \leftrightarrow A(x) \land (\neg B(x) \lor C(x)) \) can be represented by the following decision tree:

Example:
A mushroom is poisonous iff
it is yellow and small, or yellow,
big and spotted
- \( x \) is a mushroom
- \( \text{CONCEPT} = \text{POISONOUS} \)
- \( A = \text{YELLOW} \)
- \( B = \text{BIG} \)
- \( C = \text{SPOTTED} \)

<table>
<thead>
<tr>
<th>Ex. #</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>CONCEPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>6</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>7</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>8</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>9</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>10</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>11</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>12</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>13</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>14</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>15</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>16</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Possible Decision Tree
Getting Started: Top-Down Induction of Decision Tree

The distribution of training set is:

<table>
<thead>
<tr>
<th>True</th>
<th>6, 7, 8, 9, 10, 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>1, 2, 3, 4, 5, 11, 12</td>
</tr>
</tbody>
</table>

Without testing any observable predicate, we could report that CONCEPT is False (majority rule) with an estimated probability of error \( P(E) = \frac{6}{13} \).

Assuming that we will only include one observable predicate in the decision tree, which predicate should we test to minimize the probability of error (i.e., the # of misclassified examples in the training set)? → Greedy algorithm

Assume It’s A

If we test only A, we will report that CONCEPT is True if A is True (majority rule) and False otherwise.

→ The number of misclassified examples from the training set is 2

Assume It’s B

If we test only B, we will report that CONCEPT is False if B is True and True otherwise.

→ The number of misclassified examples from the training set is 5

Assume It’s C

If we test only C, we will report that CONCEPT is True if C is True and False otherwise.

→ The number of misclassified examples from the training set is 4
Assume It’s D

If we test only D, we will report that CONCEPT is True if D is True and False otherwise.

→ The number of misclassified examples from the training set is 5

Assume It’s E

If we test only E we will report that CONCEPT is False, independent of the outcome.

→ The number of misclassified examples from the training set is 6

Assume It’s E

So, the best predicate to test is A, independent of the outcome.

→ The number of misclassified examples from the training set is 6

Choice of Second Predicate

→ The number of misclassified examples from the training set is 1

Choice of Third Predicate

→ The number of misclassified examples from the training set is 1

Final Tree

CONCEPT ⇔ A ∧ (C v ¬B)

CONCEPT ⇔ A ∧ (¬B v C)
**Top-Down Induction of a DT**

\[ DTL(\Delta, \text{Predicates}) \]

1. If all examples in \( \Delta \) are positive then return True
2. If all examples in \( \Delta \) are negative then return False
3. If \( \text{Predicates} \) is empty then return failure
4. A \( \leftarrow \) error-minimizing predicate in \( \text{Predicates} \)
5. Return the tree whose:
   - root is A,
   - left branch is \( DTL(\Delta^A, \text{Predicates} - A) \),
   - right branch is \( DTL(\Delta^{-A}, \text{Predicates} - A) \)

**Comments**

- Widely used algorithm
- Greedy
- Robust to noise (incorrect examples)
- Not incremental

**Using Information Theory**

- Rather than minimizing the probability of error, many existing learning procedures minimize the expected number of questions needed to decide if an object \( x \) satisfies CONCEPT
- This minimization is based on a measure of the "quantity of information" contained in the truth value of an observable predicate
- See R&N p. 659-660

**Miscellaneous Issues**

- Assessing performance:
  - Training set and test set
  - Learning curve

Risk of using irrelevant observable predicates to generate an hypothesis that agrees with all examples in the training set
### Miscellaneous Issues

- **Assessing performance:**
  - Training set and test set
  - Learning curve
- **Overfitting**
  - Tree pruning

  **Terminate recursion when # errors / information gain is small**

- **Risk of using irrelevant observable predicates to generate an hypothesis that agrees with all examples in the training set**

### Applications of Decision Tree

- **Medical diagnostic / Drug design**
- **Evaluation of geological systems for assessing gas and oil basins**
- **Early detection of problems (e.g., jamming) during oil drilling operations**
- **Automatic generation of rules in expert systems**