Supervised (Inductive) Learning

(Slides by Pedro Domingos)

Supervised Learning

- Given: Training examples \((x_i, y_i)\) for some unknown function \(f\).
- Find: A good approximation to \(f\).

Example Applications

- Credit risk assessment
  - Propensity of customers and proposed purchase.
  \(f(x):\) Approve purchase or not.
- Disease diagnosis
  - Propensity of patient (symptoms, lab tests).
  \(f(x):\) Disease (or maybe, recommended therapy).
- Face recognition
  - Bitmap picture of person’s face
  \(f(x):\) Name of the person.
- Automatic Steering
  - Bitmap picture of road surface in front of car.
  \(f(x):\) Degrees to turn the steering wheel.

Appropriate Applications for Supervised Learning

- Situations where there is no human expert
  - Graph for a new molecule.
  \(f(x):\) Predicted binding strength to HIV protease molecule.
- Situations where humans can perform the task but can’t describe how they do it.
  - Bitmap picture of hand-written character
  \(f(x):\) Anti-code of the character.
- Situations where the desired function is changing frequently
  - Description of stock prices and trades for last 30 days.
  \(f(x):\) Recommended stock transactions.
- Situations where each user needs a customized function \(f\)
  - Becoming email message.
  \(f(x):\) Importance score for putting in box (or deeming without putting).

A Learning Problem

\[
\begin{array}{cccc|c}
\text{Example: } x_1, x_2, x_3, x_4, y \\
1 & 0 & 0 & 1 & 0 \\
2 & 0 & 1 & 0 & 0 \\
3 & 0 & 0 & 1 & 1 \\
4 & 1 & 1 & 0 & 1 \\
5 & 0 & 1 & 1 & 0 \\
6 & 1 & 1 & 0 & 0 \\
7 & 0 & 1 & 1 & 0 \\
\end{array}
\]

Hypothesis Spaces

- Complete Ignorance: There are \(2^k\) possible boolean functions over four input features. We can’t figure out which one is correct until we’ve seen every possible input-output pair. After \(2^k\) examples, we still have \(2^k\) possibilities.

- Simple Rules: There are only 16 simple conjunctive rules.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Counterexample</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_1 \rightarrow y)</td>
<td>0</td>
</tr>
<tr>
<td>(x_2 \rightarrow y)</td>
<td>1</td>
</tr>
<tr>
<td>(x_1 \land x_2 \rightarrow y)</td>
<td>0</td>
</tr>
<tr>
<td>(x_1 \land x_2 \land x_3 \rightarrow y)</td>
<td>0</td>
</tr>
<tr>
<td>(x_1 \land x_2 \land x_3 \land x_4 \rightarrow y)</td>
<td>1</td>
</tr>
</tbody>
</table>

No single rule explains the data. The name is true for simple clauses.
Hypothesis Space (3)

- no-of-rules: There are 32 possible rules (include simple conjunctions and clauses).

<table>
<thead>
<tr>
<th>variables</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_1)</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x_2)</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x_3)</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x_4, x_5)</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x_6, x_7)</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x_8)</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x_9)</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x_{10})</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x_{11}, x_{12})</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>(x_{13}, x_{14})</td>
<td>2</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>(x_{15})</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>(x_{16}, x_{17})</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>(x_{18}, x_{19})</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Two Views of Learning

- Learning is the removal of our remaining uncertainty. Suppose we know that the unknown function was an odd function, then we could use the training examples to infer which function it is.
- Learning requires guessing a good, small hypothesis class. We can start with a very small class and replace it until it contains an hypothesis that fits the data.

We could be wrong!

- Our prior knowledge might be wrong.
- Our guess of the hypothesis class could be wrong.

Example: $x_1 \land \text{even}(x_2) \Rightarrow y$ is also consistent with the training data.

Example: $x_1 \land \neg x_2 \Rightarrow y$ is also consistent with the training data.

If either of these is the unknown function, then we will make errors when we are given new $x$ values.

Two Strategies for Machine Learning

- Develop Languages for Expressing Prior Knowledge: Rule grammars and stochastic models.
- Develop Flexible Hypothesis Spaces: Nested collections of hypotheses.
- Decision trees, rules, neural networks, cases.

In either case:

- Develop Algorithms for Finding an Hypothesis that Fits the Data.

Terminology

- Training example. As example of the form $(x_i, f(x_i))$.
- Target function (target concept). The true function $f$.
- Hypothesis. A proposed function $h$ believed to be similar to $f$.
- Concept. A boolean function. Examples for which $f(x) = 1$ are called positive examples or positive instances of the concept. Examples for which $f(x) = 0$ are called negative examples or negative instances.
- Classifier. A discrete-valued function. The possible values $f(x) \in \{1, \ldots, K\}$ are called the classes or class labels.
- Hypothesis Space. The space of all hypotheses that can, in principle, be output by a learning algorithm.
- Version Space. The space of all hypotheses in the hypothesis space that have yet to be ruled out by a training example.

Key Issues in Machine Learning

- What are good hypothesis spaces?
  - Which spaces have been useful in practical applications and why?
- What algorithms can work with these spaces?
  - Are there general design principles for machine learning algorithms?
- How can we optimize accuracy on future data points?
  - This is sometimes called the "problem of overfitting."
- How can we have confidence in the results?
  - How much training data is required to find accurate hypotheses? (the statistical question)
- Are some learning problems computationally intractable?
  - (the computational question)
- How can we formulate application problems as machine learning problems? (the engineering question)

A Framework for Hypothesis Spaces

- Size. Does the hypothesis space have a fixed size or variable size?
  - Fixed-size spaces are easier to understand, but variable-size spaces can generally more useful. Variable-size spaces introduce the problem of overfitting.
- Randomness. Is such hypothesis deterministic or stochastic?
  - This affects how we evaluate hypotheses. With a deterministic hypothesis, a training example is either contained (correctly predicted) or not contained (incorrectly predicted). With a stochastic hypothesis, a training example is more likely or less likely.
- Parameterization. Is such hypothesis described by a set of symbolic (discrete) changes or is it described by a set of continuous parameters? If both are required, we say this hypothesis space has a mixed parameterization.
  - Discrete parameters must be found by combinational search methods, continuous parameters can be found by numerical search methods.
A Framework for Learning Algorithms

- **Search Procedure**
  - **Direction Computation**: solve for the hypothesis directly.
  - **Local Search**: start with an initial hypothesis, make small improvements until a local optimum.
  - **Constructive Search**: start with an empty hypothesis, gradually add structure to it until local optimum.

- **Timing**
  - **Eager**: Analyze the training data and construct an explicit hypothesis.
  - **Lazy**: Store the training data and wait until a test data point is presented, then construct an ad hoc hypothesis to classify that one data point.

- **Online vs. Batch** (for eager algorithms)
  - **Online**: Analyze each training example as it is presented.
  - **Batch**: Collect training examples, analyze them, output an hypothesis.