Machine Learning

Subfield of AI concerned with learning from data.

Broadly, using:
- Experience
- To Improve Performance
- On Some Task

(Tom Mitchell, 1997)

Why?

Developing effective learning methods has proved difficult. Why bother?

Autonomous discovery
- We don’t know something, want to find out.

Hard to program
- Easier to specify task, collect data.

Adaptive behavior
- Our agents should adapt to new data, unforeseen circumstances.
Types

Depends on feedback available:

Labeled data:
  • Supervised learning

No feedback, just data:
  • Unsupervised learning

Sequential data, weak labels:
  • Reinforcement learning

Supervised Learning

Input:
\[ X = \{x_1, \ldots, x_n\} \quad \text{inputs} \]
\[ Y = \{y_1, \ldots, y_n\} \quad \text{labels} \]

Learn to predict new labels. Given \( x \): \( y? \)

Unsupervised Learning

Input:
\[ X = \{x_1, \ldots, x_n\} \quad \text{inputs} \]

Try to understand the structure of the data.

E.g., how many types of cars? How can they vary?

Reinforcement Learning

Learning counterpart of planning.

\[ \pi : S \rightarrow A \]

\[
\max_{\pi} R = \sum_{t=0}^{\infty} \gamma^t r_t
\]
Today: Supervised Learning

Formal definition:

Given training data:
\[ X = \{x_1, \ldots, x_n\} \] inputs
\[ Y = \{y_1, \ldots, y_n\} \] labels

Produce:
Decision function \( f : X \rightarrow Y \)

That minimizes error:
\[
\sum_i \text{err}(f(x_i), y_i)
\]

Classification vs. Regression

If the set of labels \( Y \) is discrete:
- Classification
- Minimize number of errors

If \( Y \) is real-valued:
- Regression
- Minimize sum squared error

Today we focus on classification.

Key Ideas

Class of functions \( F \), from which to find \( f \):
- \( F \) is known as the hypothesis space.

Learning:
- Search over \( F \) to find \( f \) that minimizes error.

Minimize error measured on what?
- Don’t get to see future data.
- Could use test data … but! may not generalize.

Test/Train Split

General principle:
Do not measure error on the data you train on!

Methodology:
- Split data into training set and test set.
- Fit \( f \) using training set.
- Measure error on test set.

Always do this.
Decision Trees

Let's assume:

- Discrete inputs.
- Two classes (true and false).
- Input $X$ is a vector of values.

Relatively simple classifier:

- Tree of tests.
- Evaluate test for each $x_i$, follow branch.
- Leaves are class labels.

How to make one?

Given

$X = \{x_1, \ldots, x_n\}$
$Y = \{y_1, \ldots, y_n\}$

repeat:

- if all the labels are the same, we have a leaf node.
- pick an attribute and split data on it.
- recurse on each half.

If we run out of splits, and data not perfectly in one class, then take a max.

\[
\begin{array}{c|c|c|c|c}
A & B & C & L \\
T & F & T & 1 \\
T & T & F & 1 \\
T & F & F & 1 \\
F & T & F & 2 \\
F & T & T & 2 \\
F & F & T & 1 \\
F & F & F & 1 \\
\end{array}
\]
Attribute Picking

Key question:
• Which attribute to split over?

Information contained in a data set:
\[ I(A) = -f_1 \log_2 f_1 - f_2 \log_2 f_2 \]

How many “bits” of information do we need to determine the label in a dataset?

Pick the attribute with the max information gain:

\[ Gain(B) = I(A) - \sum_i f_i I(B_i) \]

Example

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>1</td>
</tr>
</tbody>
</table>

Decision Trees

What if the inputs are real-valued?
• Have inequalities rather than equalities.

Hypothesis Class

What is the hypothesis class for a decision tree?
• Discrete inputs?
• Real-valued inputs?