Machine Learning

CSE 473
Machine Learning Outline

• Machine learning:
  What & why?
  Bias

• Supervised learning
  Classifiers
  A supervised learning technique in depth
  Induction of Decision Trees

• Ensembles of classifiers

• Overfitting
Why Machine Learning

• Flood of data
  WalMart - 25 Terabytes
  WWW - 1,000 Terabytes

• Speed of computer vs. %#@! of programming
  Highly complex systems (telephone switching systems)
  Productivity = 1 line code @ day @ programmer

• Desire for customization
  A browser that browses by itself?

• Hallmark of Intelligence
  How do children learn language?
Applications of ML

- Credit card fraud
- Product placement / consumer behavior
- Recommender systems
- Speech recognition

Most mature & successful area of AI
Examples of Learning

• Baby touches stove, gets burned, next time...

• Medical student is shown cases of people with disease X, learns which symptoms...

• How many groups of dots?
What *is* Machine Learning??
A program is said to **learn** from experience $E$ with respect to task $T$ and performance measure $P$, if it’s performance at tasks in $T$, as measured by $P$, improves with experience $E$.

- **Task T:**
  
  *Playing Othello*

- **Performance Measure P:**
  
  *Percent of games won against opponents*

- **Experience E:**
  
  *Playing practice games against itself*
Issues

• What feedback (experience) is available?
• How should these features be represented?
• What kind of knowledge is being increased?
• How is that knowledge represented?
• What prior information is available?
• What is the right learning algorithm?
• How avoid overfitting?
Choosing the Training Experience

- **Credit assignment problem:**
  - **Direct training examples:**
    - E.g. individual checker boards + correct move for each
    - Supervised learning
  - **Indirect training examples:**
    - E.g. complete sequence of moves and final result
    - Reinforcement learning
- **Which examples:**
  Random, teacher chooses, learner chooses
Choosing the Target Function

- What type of knowledge will be learned?
- How will the knowledge be used by the performance program?
- E.g. checkers program
  - Assume it knows legal moves
  - Needs to choose best move
  - So learn function: $F: \text{Boards} \rightarrow \text{Moves}$
    - hard to learn
  - Alternative: $F: \text{Boards} \rightarrow R$

Note similarity to choice of problem space
The Ideal Evaluation Function

- \( V(b) = 100 \) if \( b \) is a final, won board
- \( V(b) = -100 \) if \( b \) is a final, lost board
- \( V(b) = 0 \) if \( b \) is a final, drawn board
- Otherwise, if \( b \) is not final
  \[ V(b) = V(s) \text{ where } s \text{ is best, reachable final board} \]

Nonoperational…
Want operational approximation of \( V: \hat{V} \)
How Represent Target Function

- \( x_1 \) = number of black pieces on the board
- \( x_2 \) = number of red pieces on the board
- \( x_3 \) = number of black kings on the board
- \( x_4 \) = number of red kings on the board
- \( x_5 \) = num of black pieces threatened by red
- \( x_6 \) = num of red pieces threatened by black

\[ \hat{V}(b) = a + bx_1 + cx_2 + dx_3 + ex_4 + fx_5 + gx_6 \]

Now just need to learn 7 numbers!
Example: Othello

- **Task T:**
  *Playing othello*

- **Performance Measure P:**
  *Percent of games won against opponents*

- **Experience E:**
  *Playing practice games against itself*

- **Target Function**
  \[ V: \text{board} \rightarrow \mathbb{R} \]

- **Representation of approx. of target function**

\[ \hat{V}(b) = a + bx_1 + cx_2 + dx_3 + ex_4 + fx_5 + gx_6 \]
Target Function

• Profound Formulation:  
  *Can express any type of inductive learning as approximating a function*

• E.g., Checkers  
  V: boards -> evaluation

• E.g., Handwriting recognition  
  V: image -> word

• E.g., Mushrooms  
  V: mushroom-attributes -> \{E, P\}
More Examples

• **Given:** Training examples \( \langle x, f(x) \rangle \) for some unknown function \( f \).

• **Find:** A good approximation to \( f \).

**Example Applications**

• **Credit risk assessment**
  - \( x \): Properties of customer and proposed purchase.
  - \( f(x) \): Approve purchase or not.

• **Disease diagnosis**
  - \( x \): Properties of patient (symptoms, lab tests)
  - \( f(x) \): Disease (or maybe, recommended therapy)

• **Face recognition**
  - \( x \): Bitmap picture of person’s face
  - \( f(x) \): Name of the person.
More Examples

- **Collaborative Filtering**
  Eg, when you look at book B in Amazon
  It says “Buy B and also book C together & save!”

- **Automatic Steering**
Supervised Learning

• **Inductive learning or “Prediction”:**
  *Given* examples of a function \((X, F(X))\)
  *Predict* function \(F(X)\) for new examples \(X\)

• **Classification**
  \(F(X) = \text{Discrete}\)

• **Regression**
  \(F(X) = \text{Continuous}\)

• **Probability estimation**
  \(F(X) = \text{Probability}(X)\):
Why is Learning Possible?

Experience alone never justifies any conclusion about any unseen instance.

Learning occurs when PREJUDICE meets DATA!
Bias

• The nice word for prejudice is “bias”.

• What kind of hypotheses will you consider?
  What is allowable range of functions you use when approximating?

• What kind of hypotheses do you prefer?
Some Typical Bias
The world is simple

Occam’s razor
“It is needless to do more when less will suffice”
- William of Occam,
died 1349 of the Black plague

MDL - Minimum description length
Concepts can be approximated by
... conjunctions of predicates
... by linear functions
... by short decision trees
A Learning Problem

\[ y = f(x_1, x_2, x_3, x_4) \]

<table>
<thead>
<tr>
<th>Example</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Hypothesis Spaces

- **Complete Ignorance.** There are $2^{16} = 65536$ 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 7 examples, we still have $2^9$ possibilities.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
</tbody>
</table>
Hypothesis Spaces (2)

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

<table>
<thead>
<tr>
<th>Rule</th>
<th>Counterexample</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Rightarrow y$</td>
<td>1</td>
</tr>
<tr>
<td>$x_1 \Rightarrow y$</td>
<td>3</td>
</tr>
<tr>
<td>$x_2 \Rightarrow y$</td>
<td>2</td>
</tr>
<tr>
<td>$x_3 \Rightarrow y$</td>
<td>1</td>
</tr>
<tr>
<td>$x_4 \Rightarrow y$</td>
<td>7</td>
</tr>
<tr>
<td>$x_1 \land x_2 \Rightarrow y$</td>
<td>3</td>
</tr>
<tr>
<td>$x_1 \land x_3 \Rightarrow y$</td>
<td>3</td>
</tr>
<tr>
<td>$x_1 \land x_4 \Rightarrow y$</td>
<td>3</td>
</tr>
<tr>
<td>$x_2 \land x_3 \Rightarrow y$</td>
<td>3</td>
</tr>
<tr>
<td>$x_2 \land x_4 \Rightarrow y$</td>
<td>3</td>
</tr>
<tr>
<td>$x_3 \land x_4 \Rightarrow y$</td>
<td>4</td>
</tr>
<tr>
<td>$x_1 \land x_2 \land x_3 \Rightarrow y$</td>
<td>3</td>
</tr>
<tr>
<td>$x_1 \land x_2 \land x_4 \Rightarrow y$</td>
<td>3</td>
</tr>
<tr>
<td>$x_1 \land x_3 \land x_4 \Rightarrow y$</td>
<td>3</td>
</tr>
<tr>
<td>$x_2 \land x_3 \land x_4 \Rightarrow y$</td>
<td>3</td>
</tr>
<tr>
<td>$x_1 \land x_2 \land x_3 \land x_4 \Rightarrow y$</td>
<td>3</td>
</tr>
</tbody>
</table>

No simple rule explains the data. The same is true for simple clauses.
**Terminology**

- **Training example.** An example of the form \( \langle x, f(x) \rangle \).

- **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 not yet been ruled out by a training example.
Two Strategies for ML

• Restriction bias: use prior knowledge to specify a restricted hypothesis space.
  Version space algorithm over conjunctions.
• Preference bias: use a broad hypothesis space, but impose an ordering on the hypotheses.
  Decision trees.
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 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.