What is Learning?

- Examples
  - Riding a bike (motor skills)
  - Telephone number (memorizing)
  - Read textbook (memorizing and operationalizing rules)
  - Playing backgammon (strategy)
  - Develop scientific theory (abstraction)
  - Language
  - Recognize fraudulent credit card transactions
  - Etc.

(One) Definition of Learning

**Definition [Mitchell]:**
A computer program is said to learn from
- experience E with respect to some class of
- tasks T and
- performance measure P,
if its performance at tasks in T, as measured by P,
  improves with experience E.

Examples

- **Spam Filtering**
  - T: Classify emails HAM / SPAM
  - E: Examples (e₁,HAM),(e₂,SPAM),(e₃,HAM),(e₄,SPAM), ...
  - P: Prob. of error on new emails

- **Personalized Retrieval**
  - T: find documents the user wants for query
  - E: watch person use Google (queries / clicks)
  - P: # relevant docs in top 10

- **Play Checkers**
  - T: Play checkers
  - E: games against self
  - P: percentage wins

How can an Agent Learn?

**Learning strategies and settings**
- rote learning
- learning from instruction
- learning by analogy
- learning from observation and discovery
- learning from examples

- Carbonell, Michalski & Mitchell.

Inductive Learning / Concept Learning

- **Task:**
  - Learn (to imitate) a function f: X \rightarrow Y

- **Training Examples:**
  - Learning algorithm is given the correct value of the
    function for particular inputs \textit{training examples}
  - An example is a pair (x, f(x)), where x is the input
    and f(x) is the output of the function applied to x.

- **Goal:**
  - Learn a function h: X \rightarrow Y that approximates
    f: X \rightarrow Y as well as possible.
**Concept Learning Example**

<table>
<thead>
<tr>
<th>Food</th>
<th>Chat</th>
<th>Fast</th>
<th>Price</th>
<th>Bar</th>
<th>BigTip</th>
</tr>
</thead>
<tbody>
<tr>
<td>great</td>
<td>yes</td>
<td>yes</td>
<td>normal</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>great</td>
<td>no</td>
<td>yes</td>
<td>normal</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>mediocre</td>
<td>yes</td>
<td>no</td>
<td>high</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>great</td>
<td>yes</td>
<td>yes</td>
<td>normal</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Instance Space X:** Set of all possible objects described by attributes (often called features).

**Target Function f:** Mapping from Attributes to Target Feature (often called label) \( f \) is unknown.

**Hypothesis Space H:** Set of all classification rules \( h \) we allow.

**Training Data D:** Set of instances labeled with Target Feature

---

**Classification and Regression Tasks**

**Naming:**
- If \( Y \) is a the real numbers, then called “regression”.
- If \( Y \) is a discrete set, then called “classification”.

**Examples:**
- **Steering a vehicle:** image in windshield \( \rightarrow \) direction to turn the wheel (how far)
- **Medical diagnosis:** patient symptoms \( \rightarrow \) has disease / does not have disease
- **Forensic hair comparison:** image of two hairs \( \rightarrow \) match or not
- **Stock market prediction:** closing price of last few days \( \rightarrow \) market will go up or down tomorrow (how much)
- **Noun phrase coreference:** description of two noun phrases in a document \( \rightarrow \) do they refer to the same real world entity

---

**Inductive Learning Algorithm**

- **Task:**
  - Given: collection of examples
  - Return: a function \( h \) (hypothesis) that approximates \( f \)

- **Inductive Learning Hypothesis:**
  - Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over any other unobserved examples.

- **Assumptions of Inductive Learning:**
  - The training sample represents the population
  - The input features permit discrimination

---

**Inductive Learning Setting**

**Task:**
- Learner induces a general rule \( h \) from a set of observed examples that classifies new examples accurately.

---

**Instance-Based Learning**

- **Idea:**
  - Similar examples have similar label.
  - Classify new examples like similar training examples.

- **Algorithm:**
  - Given some new example \( x \) for which we need to predict its class \( y \)
  - Find most similar training examples
  - Classify \( x \) “like” these most similar examples

- **Questions:**
  - How to determine similarity?
  - How many similar training examples to consider?
  - How to resolve inconsistencies among the training examples?

---

**K-Nearest Neighbor (KNN)**

- **Given:** Training data \( \{(x_1, y_1), \ldots, (x_m, y_m)\} \)
  - Attribute vectors: \( x_i \in \mathbb{X} \)
  - Target attribute: \( y \in \{-1, +1\} \)

- **Parameter:**
  - Similarity function: \( R : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R} \)
  - Number of nearest neighbors to consider: \( k \)

- **Prediction rule**
  - New example \( x' \) : 
    - K-nearest neighbors: \( k \) training examples with largest \( R(x', x_i) \)
    
    \[
    h(x') = \arg \max_{y \in \{-1, +1\}} \left\{ \sum_{i=1}^{k} 1_{x_i \in \text{neighbors}(x')} \right\}
    \]
KNN Example

<table>
<thead>
<tr>
<th>Food</th>
<th>Chat</th>
<th>Fast</th>
<th>Price</th>
<th>Bar</th>
<th>BigTip</th>
</tr>
</thead>
<tbody>
<tr>
<td>great</td>
<td>yes</td>
<td>yes</td>
<td>normal</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>great</td>
<td>no</td>
<td>yes</td>
<td>normal</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>mediocre</td>
<td>yes</td>
<td>no</td>
<td>high</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>great</td>
<td>yes</td>
<td>yes</td>
<td>normal</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

• New examples:
  – (great, no, no, normal, no)
  – (mediocre, yes, no, normal, no)

Types of Attributes

• Symbolic (nominal)
  – EyeColor {brown, blue, green}
• Boolean
  – anemic {TRUE,FALSE}
• Numeric
  – Integer: age [0, 105]
  – Real: length
• Structural
  – Natural language sentence: parse tree
  – Protein: sequence of amino acids

KNN for Real-Valued Attributes

• Similarity Functions:
  – Gaussian: $d(x, x') = e^{-d(x, x')^2}$
  – Cosine: $d(x, x') = \arccos(x \cdot x')$

Selecting the Number of Neighbors

• Increase k:
  – Makes KNN less sensitive to noise
• Decrease k:
  – Allows capturing finer structure of space
  ➤ Pick k not too large, but not too small (depends on data)

Example: Effect of k

Advantages and Disadvantages of KNN

• Simple algorithm
• Need similarity measure and attributes that “match” target function.
• For large training sets, requires large memory is slow when making a prediction.
• Prediction accuracy can quickly degrade when number of attributes grows.
<table>
<thead>
<tr>
<th>Curse-of-Dimensionality</th>
<th>Remarks on KNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Prediction accuracy can quickly degrade when number of attributes grows.</td>
<td>• Memorizes all observed instances and their class</td>
</tr>
<tr>
<td>– Irrelevant attributes easily “swamp” information from relevant attributes</td>
<td>• Is this rote learning?</td>
</tr>
<tr>
<td>[ K(x_i, y) \sim \frac{1}{(1 + d(x, y))^p} ]</td>
<td>• Is this really learning?</td>
</tr>
<tr>
<td>[ \text{When many irrelevant attributes, similarity measure} ]</td>
<td>• When does the induction take place?</td>
</tr>
<tr>
<td>becomes less reliable</td>
<td></td>
</tr>
<tr>
<td>• Remedy</td>
<td></td>
</tr>
<tr>
<td>– Try to remove irrelevant attributes in pre-processing step</td>
<td></td>
</tr>
<tr>
<td>– Weight attributes differently</td>
<td></td>
</tr>
<tr>
<td>– Increase k (but not too much)</td>
<td></td>
</tr>
</tbody>
</table>