Learning
Schedule

• Search
• Machine learning
• Knowledge based systems
• Discovery
Function Approximation

• Problem:
  – Storing Q or U,T,R for each state in a table is too expensive, if number of states is large
  – Does not exploit “similarity” of states (i.e. agent has to learn separate behavior for each state, even if states are similar)

• Solution:
  – Approximate function using parametric representation
    \[ U(s) = \bar{w} \cdot \Phi(s) \]
  – For example:
    • \( \Phi(s) \) is feature vector describing the state
      – “Material values” of board
      – Is the queen threatened?
      – ...

Bonus: Can predict utilities in areas it has never been to...
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 \((e_1, \text{HAM}), (e_2, \text{SPAM}), (e_3, \text{HAM}), (e_4, \text{SPAM}), \ldots\)
  – 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 (memorization, like RL)
• learning from instruction (being told)
• learning by analogy (from known to new, adaptation)
• learning from examples (inductive)
• learning from observation and discovery (unsupervised)

--- 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 $\rightarrow$ 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.
<table>
<thead>
<tr>
<th>Food (3)</th>
<th>Chat (2)</th>
<th>Fast (2)</th>
<th>Price (3)</th>
<th>Bar (2)</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>

Will the following situation yield a big tip?
Food=great  chat=no  fast=yes  price=high  bar=no
A=Yes      B=No
## 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 $h_i$ 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
Challenge

• Design a Fashion advisor: Looks at your online store options and helps you decide
Challenge 2

• Backseat driver: Watches your driving and “helps”
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
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 $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$
  - Attribute vectors: $\vec{x}_i \in X$
  - Target attribute: $y_i \in \{-1, +1\}$

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

- **Prediction rule**
  - New example $x'$
  - K-nearest neighbors: $k$ training examples with largest $K(\vec{x}_i, \vec{x}')$

$$h(\vec{x}') = \arg \max_{y \in Y} \left\{ \sum_{i \in \text{knn}(\vec{x}')} 1[y_i = y] \right\}$$
## KNN Example

<table>
<thead>
<tr>
<th>Food (3)</th>
<th>Chat (2)</th>
<th>Fast (2)</th>
<th>Price (3)</th>
<th>Bar (2)</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)  
    - 21
  - (mediocre, yes, no, normal, no)  
    - 31

\[
\text{A = YES \hspace{1cm} B = NO}
\]
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
    - Edit distance
KNN for Real-Valued Attributes

• Similarity Functions:
  – Gaussian: \( K(\vec{x}_i, \vec{x}') \sim e^{-(\vec{x}_i - \vec{x}')^2} \)
  – Cosine: \( K(\vec{x}_i, \vec{x}') = \cos(\vec{x}_i, \vec{x}') \)
Other distance metrics

Scaled Euclidean (diagonal covariance)

Mahalanobis (full covariance)

PCA

$L_1$ norm

$L_\infty$ (max) norm

Copied from Andrew Moore
Computing answers from samples

• Classification
  – Majority vote

• Regression
  – Weighted sum
  – Local hyper-plane fit
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

Hastie, Tibshirani, Friedman 2001
128 samples with 10-neighborhoods on two query points, and the Gaussian distribution that generated the samples.
Estimation of density using $k = 3, 10, 40$
Cross Validation

• Train with 80% of data, test on remaining 20%
  – Repeat 5 times with other subsets
  – Can be different ratios

• Try for different k’s, choose best
Curse-of-Dimensionality

- Dataset size $N$ in $d$-dimensional space ($d$ attributes) in unit cube
- Looking for hypercubic neighborhood of size $b^d$ to contain $k$ neighbors
  - $B^d = k/N \Rightarrow b = (k/N)^{1/d}$
  - $d=100, k=10, N=1,000,000 \Rightarrow b=0.89$
  - $d=2, k=10, N=1,000,000 \Rightarrow b=0.003$
Curse-of-Dimensionality

- Prediction accuracy can quickly degrade when number of attributes grows.
  - Irrelevant attributes easily "swamp" information from relevant attributes

\[ K(\bar{x}_i, \bar{x}'_i) \sim e^{-\left(\sum_{j \in A_{rel}} (\bar{x}_i[j] - \bar{x}'_i[j])^2 + \sum_{j \in A_{irrel}} (\bar{x}_i[j] - \bar{x}'_i[j])^2\right)} \]

⇒ When many irrelevant attributes, similarity measure becomes less reliable
Curse-of-Dimensionality

• Remedy
  – Try to remove irrelevant attributes in pre-processing step
  – Weight attributes differently

• How can we use Cross-validation to do this?
Performance

• Need to search through all points
  – $O(n)$ per query

• How can we improve?
  – Hierarchical access
  – Random subsampling
  – Thin data: Remove duplicate points
What about Uncertainty?

• Confidence is an important factor in learning
  – How can we estimate uncertainty in our answer?
• Classification
  – Voting balance levels
• Regression
  – Linear fit error
• General
  – Compare results from subsets of data
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.
Remarks on KNN

• Memorizes all observed instances and their class
• Is this rote learning?
• Is this really learning?
• When does the induction take place?
“The End of Science”

Chris Anderson

Wired July 16, 2008

“Correlation is enough. Faced with massive data, [the Scientific Method] is becoming obsolete. We can stop looking for models.”