Learning Agent

- An agent that improves its performance at some task through experience
Details, details

- How is knowledge represented?
- How is performance measured and critiqued?
- How is experience represented and obtained?
Automated knowledge acquisition
Discover new knowledge
Understand human learning
Agents need to adapt to unknown, dynamic environments

"If you invent a breakthrough in artificial intelligence, so machines can learn, that is worth ten Microsofts.”
Bill Gates, 2004
Applications

- Medical diagnosis
- Autonomous control (planes, trains, automobiles, robots, ...)
- Perception (speech, language, images, video)
- Recommendations (Amazon, Netflix)
- Prediction (business, financial, environment, health, energy, security, ...)
- Fraud/intrusion detection
- ...
Approaches

- Supervised Learning
  - Classification
  - Regression

- Unsupervised Learning
  - Clustering

- Reinforcement Learning
Supervised Learning

- Given training set of N examples of $y = f(x)$
  - $(x_1, y_1)$, $(x_2, y_2)$, ..., $(x_N, y_N)$
- Find hypothesis $h$ that approximates $f$
Example: “Family Car”

- Learn to classify cars into one of two classes: “family car” or “other”
- Each car is represented by two features: “engine power” and “price”
- Given several training examples of already-classified cars
- Output classifier that accurately classifies all cars
“Family Car” Target Concept
Note: \( h \) is consistent with the training set, but not the target concept \( C \).
If $A$ and $B$ are noise, then $h_2$ overfits.

If $A$ and $B$ are \textit{not} noise, then $h_1$ underfits.
Model (Hypothesis) Selection

- Model Complexity
- Model Error

- Underfitting
- Overfitting

- Training Data
- Testing (Validation) Data
Supervised Learning

Bayesian Learning
Neural Networks
Bayesian Learning

- Combines prior knowledge with evidence to make predictions
- Optimal classifier (not practical)
  - Need to know all true probabilities
- Naïve Bayes classifier (practical)
  - Assumes independence among features
**Bayes Rule**

\[
P(C_i \mid x) = \frac{P(x \mid C_i)P(C_i)}{P(x)}
\]

- \(C_i\) is the class, \(1 \leq i \leq K\)
- \(x\) is the feature values of an instance
- \(P(C_i \mid x)\) = probability that instance \(x\) belongs to class \(C_i\) \((\text{posterior})\)
- \(P(x \mid C_i)\) = probability that an instance drawn from class \(C_i\) would be \(x\) \((\text{likelihood})\)
- \(P(C_i)\) = probability of class \(C_i\) \((\text{prior})\)
- \(P(x)\) = probability of instance \(x\) \((\text{evidence})\)
Bayes Rule: Family Car

\[ P(\text{FamilyCar} \mid \text{EnginePower}, \text{Price}) = \frac{P(\text{EnginePower}, \text{Price} \mid \text{FamilyCar}) P(\text{FamilyCar})}{P(\text{EnginePower}, \text{Price})} \]
Bayes Classifier

- Classify instance $x$ as class $C_i$ such that
  $$i = \arg \max_{1 \leq k \leq K} P(C_k \mid x)$$

- Since only interested in maximum, can ignore denominator $p(x)$ (i.e., ignore normalization $\alpha$)
  $$i = \arg \max_{1 \leq k \leq K} P(x \mid C_k)P(C_k)$$

- If prior probability distribution of classes is uniform, then can ignore $P(C_i)$
  $$i = \arg \max_{1 \leq k \leq K} P(x \mid C_k)$$
Practical issue #1
- $P(x \mid C_i)$ is a joint probability distribution
- Need to know the probability of every possible instance given every possible class
- Even for $D$ boolean attributes and $K$ classes, that’s $K \times 2^D$ probabilities

Solution
- Assume attributes are independent of each other

$$p(x_1, x_2, \ldots, x_D \mid C_i) = \prod_{j=1}^{D} p(x_j \mid C_i)$$
Given training set $X$

Estimate probabilities from $X$

$$P(C_i) = \frac{\left| \{(x,y) \in X \mid y = C_i\} \right|}{|X|}$$

$$P\left(x_j = v \mid C_i\right) = \frac{\left| \{(x,y) \in X \mid x_j = v \land y = C_i\} \right|}{\left| \{(x,y) \in X \mid y = C_i\} \right|}$$

Classify new instance $x$ as class $C_i$ such that

$$i = \arg \max_{1 \leq k \leq K} P(C_k) \cdot \prod_{j=1}^{D} P(x_j \mid C_k)$$
Example

- \( P(\text{FamilyCar}) \)
  - \( P(\text{FamilyCar}=\text{yes}) = \frac{3}{8} \)
  - \( P(\text{FamilyCar}=\text{no}) = \)

- \( P(\text{Price} \mid \text{FamilyCar}) \)
  - \( P(\text{Price}=10000 \mid \text{FamilyCar}=\text{yes}) = \frac{2}{3} \)
  - \( P(\text{Price}=20000 \mid \text{FamilyCar}=\text{yes}) = \)
  - \( P(\text{Price}=30000 \mid \text{FamilyCar}=\text{yes}) = \)
  - \( P(\text{Price}=10000 \mid \text{FamilyCar}=\text{no}) = \)
  - \( P(\text{Price}=20000 \mid \text{FamilyCar}=\text{no}) = \)
  - \( P(\text{Price}=30000 \mid \text{FamilyCar}=\text{no}) = \)

- \( P(\text{EnginePower} \mid \text{FamilyCar}) \)
  - \( P(\text{EnginePower}=100 \mid \text{FamilyCar}=\text{yes}) = \frac{2}{3} \)
  - \( P(\text{EnginePower}=200 \mid \text{FamilyCar}=\text{yes}) = \)
  - \( P(\text{EnginePower}=300 \mid \text{FamilyCar}=\text{yes}) = \)
  - \( P(\text{EnginePower}=100 \mid \text{FamilyCar}=\text{no}) = \)
  - \( P(\text{EnginePower}=200 \mid \text{FamilyCar}=\text{no}) = \)
  - \( P(\text{EnginePower}=300 \mid \text{FamilyCar}=\text{no}) = \)

- \( P(\text{FamilyCar}=\text{yes} \mid \text{Price}=20000 \land \text{EnginePower}=200) = ? \)

<table>
<thead>
<tr>
<th>Price</th>
<th>EnginePower</th>
<th>FamilyCar</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>100</td>
<td>yes</td>
</tr>
<tr>
<td>10000</td>
<td>200</td>
<td>yes</td>
</tr>
<tr>
<td>10000</td>
<td>300</td>
<td>no</td>
</tr>
<tr>
<td>20000</td>
<td>100</td>
<td>yes</td>
</tr>
<tr>
<td>20000</td>
<td>300</td>
<td>no</td>
</tr>
<tr>
<td>30000</td>
<td>100</td>
<td>no</td>
</tr>
<tr>
<td>30000</td>
<td>200</td>
<td>no</td>
</tr>
<tr>
<td>30000</td>
<td>300</td>
<td>no</td>
</tr>
</tbody>
</table>
Example (cont.)

- \( P(\text{FamilyCar}=\text{yes} \mid \text{Price}=20000 \land \text{EnginePower}=200) = ? \)
Naïve Bayes Classifier

- **Practical issue #2**
  - What if no examples in class $C_i$ have $x_j = v$?

$$P(x_j = v \mid C_i) = 0$$

$$P(C_i) \cdot \prod_{j=1}^{D} P(x_j \mid C_i) = 0$$

- **Solution**

$$p(x_j = v \mid C_i) = \frac{|\{(x, y) \in X \mid x_j = v \text{ and } y = C_i\}| + 1}{|\{(x, y) \in X \mid y = C_i\}| + |\text{values}(x_j)|}$$
Another Example

- $P(\text{FamilyCar}=\text{yes} \mid \text{Price}=30000 \land \text{EnginePower}=200) = ?$
  
  $= \alpha \times P(\text{Price}=30000 \mid \text{FamilyCar}=\text{yes}) \times$
  
  $P(\text{EnginePower}=200 \mid \text{FamilyCar}=\text{yes}) \times$
  
  $P(\text{FamilyCar}=\text{yes})$
  
  $= \alpha \times (0/3)(1/3)(3/8) = ?$

- values(Price) = \{10000, 20000, 30000\}
  
  $\mid$ values(Price) $\mid = 3$

- $P(\text{Price}=30000 \mid \text{FamilyCar}=\text{yes}) = ?$
Practical issue #3
  ◦ What if $x_j$ is a continuous feature?

Solution 1
  ◦ Assume some parameterized distribution for $x_j$
    • E.g., normal
  ◦ Learn parameters of distribution from data
    • E.g., mean and variance of $x_j$ values

Solution 2
  ◦ Discretize feature
  ◦ E.g., price $\in \mathbb{R}$ to price $\in \{\text{low, medium, high}\}$
Naïve Bayes for Regression

- Bayes learning for classification

\[ i = \arg \max_{1 \leq k \leq K} P(C_k | x) \]

- Bayes learning for regression

\[ y^* = \max_y P(y|x) = \max_y P(x|y)P(y) \]

- Naïve Bayes learning for regression

\[ y^* = \max_y P(x_1|y) \times \cdots \times P(x_D|y) \times P(y) \]

- Each probability is a multivariate normal distribution
  - Learn these from the data
Bayesian Learning: Summary

- Optimal learner, in theory
- Naïve Bayes learner, practical
  - Independence assumption rarely true
    - E.g., Is “price” independent of “engine power”?  
  - Naïve Bayes learner still does surprisingly well
  - Simple, effective baseline for other learners
Neural Networks
Inspired from brain
  ◦ Brain consists of interconnected neurons
  ◦ Humans still outperform machines in most areas of intelligence
Human Brain vs. Computer

- Processors
  - Computer
    - CPUs: 20 cores (10^9 Hz)
    - GPUs: 10^4 cores (10^9 Hz)
  - Human brain
    - 10^{11} neurons (10^3 Hz)

- Parallelism
  - Computer
    - CPUs some, GPUs lots
    - Few interconnections
  - Human brain
    - 10^{15} synapses
    - Each neuron has \sim 10^4 connections to other neurons
The Singularity

1. The accelerating pace of change...
   - Agricultural Revolution: 8,000 years
   - Industrial Revolution: 120 years
   - Light-bulb: 90 years
   - Moon landing: 22 years
   - World Wide Web: 9 years
   - Human genome sequenced

2. ...and exponential growth in computing power...
   - Computer technology, shown here climbing dramatically by powers of 10, is now progressing more each hour than it did in its entire first 90 years.

3. ...will lead to the Singularity
   - Apple II
     - At a price of $1,298, the compact machine was one of the first massively popular personal computers.
   - UNIVAC I
     - The first commercially marketed computer, used to tabulate the U.S. Census, occupied 943 cu. ft.
   - Colossus
     - The electronic computer, with 1,500 vacuum tubes, helped the British crack German codes during WW II.

Time Magazine (Feb 2011)
What Happens Then…?

**Perceptron (Rosenblatt, 1962)**

**Learn function f**
Modify weights so $y = f(x)$

$$y = \sum_{j=1}^{d} w_j x_j + w_0 = \mathbf{w} \cdot \mathbf{x}$$

$$\mathbf{w} = [w_0, w_1, \ldots, w_d]$$

$$\mathbf{x} = [1, x_1, \ldots, x_d]$$
Perceptron Regression

- \( y = wx + w_0 \)

\( x_0 = +1 \)
If \((wx + w_0 > 0)\) Then \(y = 1\) Else \(y = 0\)
Perceptron Training

- Change weights to reduce error
- Gradient descent

\[ \Delta w_i = \eta (y_j - o_j) x_{ji} \]

- \( o_j \) = output of perceptron for example \( j \)
- Learning rate \( \eta \) controls rate of descent
Example

<table>
<thead>
<tr>
<th>Price</th>
<th>EnginePower</th>
<th>FamilyCar</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ \Delta w_i = \eta (y_j - o_j) x_{ji} \]

Let \( \eta = 0.5 \)

\[ x_0 = 1 \]
\[ x_1 = \text{Price} \]
\[ x_2 = \text{EnginePower} \]

\[ w_0 = 1 \]
\[ w_1 = 1 \]
\[ w_2 = 1 \]

If \( \Sigma w_i x_i \geq 0 \) Then 1 (yes) Else 0 (no)
Linearly Separable
Logistic Function

- Also called “sigmoid function”
Logistic Function
Perceptron Classification

\[ y = \text{logistic} \left( w \cdot x \right) = \frac{1}{1 + e^{-w \cdot x}} \]
Learning Nonlinear Functions

- Perceptrons can only approximate linear functions
- But multiple layers of logistic perceptrons can approximate any nonlinear function
Multilayer Perceptrons

\[ y_i = v_i \cdot z = \sum_{h=1}^{H} v_{ih}z_h + v_{i0} \]

\[ z_h = \text{logistic} \left( w_h \cdot x \right) \]

\[ = \frac{1}{1 + \exp \left( - \left( \sum_{j=1}^{d} w_{hj}x_j + w_{h0} \right) \right)} \]

Learn weights by error backpropagation. (Rumelhart et al., 1986)
Overfitting in MLPs

\[ f(x) = \sin(6x) \]
Overfitting in MLPs

- Similar overfitting behavior if training continued too long
- More and more weights move from zero
- Overtraining
Deep Learning

- Using deep (many-layered) neural nets to learn complex abstractions (features)
Deep Convolutional Neural Net (CNN) for Image Classification

Conv_1
Convolution (5 x 5) kernel valid padding

Max-Pooling (2 x 2)

Conv_2
Convolution (5 x 5) kernel valid padding

Max-Pooling (2 x 2)

ReLU

fc_3
Fully-Connected Neural Network
ReLU activation

fc_4
Fully-Connected Neural Network

(with dropout)

INPUT
(28 x 28 x 1)

n1 channels
(24 x 24 x n1)

n1 channels
(12 x 12 x n1)

n2 channels
(8 x 8 x n2)

n2 channels
(4 x 4 x n2)

Flattened

n3 units

OUTPUT

0
1
2
...
9
Unsupervised Learning

k-Means Clustering
**$k$-means Clustering**

- Unsupervised learning
  - Just feature values, no classes
- Partition instances into $k$ disjoint sets
- Each set has a representative instance $m_i$
- Place instance $x$ into set $i$ such that $\text{distance}(x, m_i)$ is minimal
- Choose new central $m_i$ for each set
- Repeat until $m_i$ converge
Artificial Intelligence
Choosing $k$?
- Try several ($2 \leq k \leq N$)
- Choose $k$ minimizing intra-cluster distance and maximizing inter-cluster distance
ML Software Tools

- Waikato Environment for Knowledge Analysis (WEKA)
  - Java based
  - http://www.cs.waikato.ac.nz/ml/weka

- SciKit Learn
  - Python based
  - http://scikit-learn.org

- Deep Learning (mostly Python)
  - TensorFlow (https://www.tensorflow.org)
  - Keras (https://keras.io)
Reinforcement Learning
Reinforcement Learning

- Agent in some state in the environment, takes an action and sometimes receives reward, and the state changes
- Delayed reward
- Credit-assignment
- Learn a policy
  - $\pi$: State $\to$ Action
- Applications
  - Game-playing
  - Robot control
Reinforcement Learning Agents

- **Utility-based agent**
  - Learn utility function on states
  - Choose actions to maximize expected utility
  - Requires action model
    - Action: state $\rightarrow$ state

- **Q-learning agent**
  - Learns Q function: value of taking action A in state S
  - No action model, but learning slower

- **Passive learning of utility or Q functions**
  - Policy given

- **Active learning: Also learn policy**
  - Exploration
Simple Environment

- Actions: Up, Down, Left, Right
- 80% chance intended move executed
- 20% chance agent moves 90 degrees from intended direction
- Agent bumps into walls and obstacle
- Reward
  - +1 for reaching (4,3)
  - −1 for reaching (4,2)
  - −0.04 for other moves
- Fully observable
Passive Reinforcement Learning

- Given policy $\pi(s)$
- Learn utility function $U^\pi(s)$
Utility Estimation

- Utility of state $s$ is the reward in state $s$ plus the expected utility of its successor states

$$U^\pi(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) U^\pi(s')$$

- $\gamma$ is the discount factor ($0 < \gamma \leq 1$)
Utility Estimation: Example

- Let $\gamma=1$
- Using previous example and policy
  - Initially all $U(s)=0$
  - Except $U([4,3])=1$ and $U([4,2])=-1$
- $U([3,3]) =$ ?, policy says go Right

Eventually, $U([3,3])$
- $= (-0.04) + (0.8)*(1) + (0.1)*(0.918) + (0.1)*(0.660)$
- $= 0.918$
Temporal Difference Learning

- Observe transition from $s$ to $s'$
- Adjust utility of state $s$ to be closer to utility of state $s'$

$$U^\pi(s) = U^\pi(s) + \alpha(R(s) + \gamma U^\pi(s') - U^\pi(s))$$

- Use learning rate $\alpha$ to control rate of adjustment
- If $\alpha$ decreases with the number of times state $s$ is visited, then TD will converge to correct $U^\pi(s)$
Active Reinforcement Learning

- Learn policy $\pi(s)$
- Agent in state $s$ takes action maximizing expected utility

$$\pi(s) = \arg \max_a \sum_{s'} P(s' | s, a) * U(s')$$

- Initial actions random
- Estimate $P(s' | s, a)$ and $U(s)$ over time
- Can result in suboptimal policy
Exploration

- Force agent to explore unvisited states to avoid suboptimal policies
- Exploration function

\[ f(u,n) = \begin{cases} 
R^+ & \text{if } n < N_e \\
u & \text{otherwise} 
\end{cases} \]

- \( u \) is the utility estimate so far
- \( n \) is the number of times state \( s \) has been visited
- \( R^+ \) is estimate of best possible reward
- \( N_e \) is the number of times an action-state must be tried before relying on utility estimate
Q-Learning

- Active temporal-difference learning agent
- Define $Q(s, a)$ as the value of taking action $a$ in state $s$

$$U(s) = \max_a Q(s, a)$$

$$Q(s, a) = R(s) + \gamma \sum_{s'} P(s' | s, a) \max_{a'} Q(s', a')$$

- $Q$ function combines $\pi(s)$, $P(s' | s, a)$ and $U(s)$
Q-Learning

- Use TD approach to eliminate $P(s'|s,a)$

$$Q(s, a) \leftarrow Q(s, a) + \alpha(R(s) + \gamma \max_{a'} Q(s', a') - Q(s, a))$$

- Calculated when observe transition from $s$ to $s'$ using action $a$
Q-Learning Agent

function Q-LEARNING-AGENT (percept) returns an action
inputs: percept, a percept indicating current state $s'$ and reward signal $r'$
persistent: $Q$, table of action values indexed by state and action, initially zero
$N_{sa}$, table of frequencies for state-action pairs, initially zero
$s, a, r$, previous state, action and reward, initially null

if TERMINAL?(s) then
   foreach $a' \in$ ACTIONS(s)
      $Q[s,a'] \leftarrow r'$
if $s$ is not null then
   increment $N_{sa}[s,a]$
   $Q[s,a] \leftarrow Q[s,a] + \alpha(N_{sa}[s,a])(r + \gamma \max_{a'} Q[s',a'] - Q[s,a])$
$s, a, r \leftarrow s', \arg\max_{a'} f(Q[s',a'], N_{sa}[s',a']), r'$
return $a$

Exploration function $f(u,n)$
Learning rate (decreasing with $n$): $\alpha(n) = \frac{C}{C + n - 1}$
Wumpus World

- Q-learning agent
- Limited to 4x4 worlds
- 512 states
  - 16 locations, 4 orientations, hasGold, hasArrow, inCave
- 6 actions
- Parameters
  - Best possible reward $R^+ = 1000$
  - Minimum state–action occurrences $N_e = 5$
  - Discount factor $\gamma = 0.9$
  - Learning rate $\alpha(n) = 100 / (99 + n)$
RL Software Tools

- OpenAI Gym
  - RL environments
  - gym.openai.com

Cart Pole
Deep Reinforcement Learning

- Deep Q-Learning Network (DQN)
  - Use Deep Learning Network to learn Q function
  - Agent “replays” past games
Summary: Learning

- Improving performance at some task through experience
- Supervised learning methods
  - Naïve Bayes
  - Neural network
- How to choose the right model?
  - Overfitting
- Unsupervised learning
  - Clustering
- Reinforcement learning