Machine Learning: Decision Trees & Statistical Learning
Discussion: Feedforward ANNs & Backprop

William H. Hsu
Department of Computing and Information Sciences, KSU

KSOL course page: http://snipurl.com/v9v3
Course web site: http://www.kddresearch.org/Courses/CIS730
Instructor home page: http://www.cis.ksu.edu/~bhsu

Reading for Next Class:
Chapter 20, Russell and Norvig

Example Trace

\[G_0 = G_1 = G_2\]
\[G_3 \rightarrow \langle ?, ?, ?, ?, ? \rangle \]
\[G_4 \rightarrow \langle ?, ? \rangle \rightarrow \langle ?, ?, ?, ?, ? \rangle \]
\[S_4 \rightarrow \langle ?, ? \rangle \rightarrow \langle ?, ?, ?, ?, ? \rangle \]
\[S_2 = S_3 \rightarrow \langle ?, ? \rangle \rightarrow \langle ?, ?, ?, ?, ? \rangle \]
\[S_1 \rightarrow \langle ?, ?, ?, ?, ?, ? \rangle \]
\[S_0 \rightarrow \langle ?, ?, ?, ?, ?, ? \rangle \]
**What Next Training Example?**

- **Active Learning:** What query should the learner make next?
- **How should these be classified?**
  - <Sunny, Warm, Normal, Strong, Cool, Change>
  - <Rainy, Cold, Normal, Light, Warm, Same>
  - <Sunny, Warm, Normal, Light, Warm, Same>

**What Justifies This Inductive Leap?**

- **Example:** Inductive Generalization
  - Positive example: <Sunny, Warm, Normal, Strong, Cool, Change, Yes>
  - Positive example: <Sunny, Warm, Normal, Light, Warm, Same, Yes>
  - Induced S: <Sunny, Warm, Normal, ?, ?, ?>
- **Why believe we can classify the unseen?**
  - e.g., <Sunny, Warm, Normal, Strong, Warm, Same>
  - When is there enough information (in a new case) to make a prediction?
• Inductive Bias
  – Any preference for one hypothesis over another, besides consistency
  – Example: $H$ = conjunctive concepts with don’t cares
  – What concepts can $H$ not express? (Hint: what are its syntactic limitations?)

• Idea
  – Choose unbiased $H'$: expresses every teachable concept (i.e., power set of $X$)
  – Recall: $|A \rightarrow B| = |B| \cdot |A| (A = X; B = \text{labels}); H = A \rightarrow B$
  – $\{(\text{Rainy}, \text{Sunny}, \text{Cloudy}) \times \{\text{Warm}, \text{Cold}\} \times \{\text{Normal}, \text{High}\} \times \{\text{None-Mild}, \text{Strong}\} \times \{\text{Cool}, \text{Warm}\} \times \{\text{Same}, \text{Change}\}\} \rightarrow \{0, 1\}$

• An Exhaustive Hypothesis Language
  – Consider: $H' = \text{disjunctions (v), conjunctions (\&), negations (~) over } H$
  – $|H'| = 2^{3 \cdot 2 \cdot 3 \cdot 2 \cdot 2} = 2^{96} ; \ |H| = 1 + (3 \cdot 3 \cdot 3 \cdot 4 \cdot 3 \cdot 3) = 973$

• What Are $S, G$ For The Hypothesis Language $H'$?
  – $S \leftarrow \text{disjunction of all positive examples}$
  – $G \leftarrow \text{conjunction of all negated negative examples}$

• Components of An Inductive Bias Definition
  – Concept learning algorithm $L$
  – Instances $X$, target concept $c$
  – Training examples $D_c = \{<x_i, c(x_i)>\}$
  – $L(x_i, D_c) = \text{classification assigned to instance } x_i \text{ by } L \text{ after training on } D_c$

• Definition
  – The inductive bias of $L$ is any minimal set of assertions $B$ such that, for any target concept $c$ and corresponding training examples $D_c$
  – $\forall x_i \in X\. \{[B \land D_c \land x_i] \rightarrow L(x_i, D_c)\}$
  – Informal idea: preference for (i.e., restriction to) certain hypotheses by structural (syntactic) means

• Rationale
  – Prior assumptions regarding target concept
  – Basis for inductive generalization
### Inductive Systems & Equivalent Deductive Systems

**Inductive System**

- Training Examples → Candidate Elimination Algorithm → Using Hypothesis Space $H$ → Classification of New Instance (or "Don’t Know")
- New Instance

**Equivalent Deductive System**

- Training Examples → Theorem Prover → Classification of New Instance (or "Don’t Know")
- New Instance

- Assertion \{ $c \in H$ \}
- Inductive bias made explicit

---

### Three Learners With Different Biases

- **Rote Learner**
  - Weakest bias: anything seen before, i.e., no bias
  - Store examples
  - Classify $x$ *if and only if* it matches previously observed example

- **Version Space Candidate Elimination Algorithm**
  - Stronger bias: concepts belonging to conjunctive $H$
  - Store extremal generalizations and specializations
  - Classify $x$ *if and only if* it “falls within” $S$ and $G$ boundaries (all members agree)

- **Find-S**
  - Even stronger bias: most specific hypothesis
  - Prior assumption: any instance *not observed to be positive* is negative
  - Classify $x$ based on $S$ set
Removal of (Remaining) Uncertainty

- Suppose unknown function was known to be m-of-n Boolean function
- Could use training data to infer the function

Learning and Hypothesis Languages

- Possible approach to guess a good, small hypothesis language:
  - Start with a very small language
  - Enlarge until it contains a hypothesis that fits the data
- Inductive bias
  - Preference for certain languages
  - Analogous to data compression (removal of redundancy)
  - Later: coding the “model” versus coding the “uncertainty” (error)

We Could Be Wrong!

- Prior knowledge could be wrong (e.g., \( y = x_4 \wedge \text{one-of (} x_1, x_3 \text{)} \) consistent)
- If guessed language was wrong, errors will occur on new cases

Approaches to Learning

- Develop Ways to Express Prior Knowledge
  - Role of prior knowledge: guides search for hypotheses / hypothesis languages
  - Expression languages for prior knowledge
    - Rule grammars; stochastic models; etc.
    - Restrictions on computational models; other (formal) specification methods

- Develop Flexible Hypothesis Spaces
  - Structured collections of hypotheses
    - Agglomeration: nested collections (hierarchies)
    - Partitioning: decision trees, lists, rules
    - Neural networks; cases, etc.
  - Hypothesis spaces of adaptive size

Either Case: Develop Algorithms for Finding A Hypothesis That Fits Well

- Ideally, will generalize well
- Later: Bias Optimization (Meta-Learning, Wrappers)
When to Consider Using Decision Trees

- Instances Describable by Attribute-Value Pairs
- Target Function Is Discrete Valued
- Disjunctive Hypothesis May Be Required
- Possibly Noisy Training Data
- Examples
  - Equipment or medical diagnosis
  - Risk analysis
    - Credit, loans
    - Insurance
    - Consumer fraud
    - Employee fraud
  - Modeling calendar scheduling preferences (predicting quality of candidate time)

Decision Trees & Decision Boundaries

- Instances Usually Represented Using Discrete Valued Attributes
  - Typical types
    - Nominal (red, yellow, green)
    - Quantized (low, medium, high)
  - Handling numerical values
    - Discretization, a form of vector quantization (e.g., histogramming)
    - Using thresholds for splitting nodes
- Example: Dividing Instance Space into Axis-Parallel Rectangles
**Decision Tree Learning: Top-Down Induction**

- **Algorithm Build-DT (Examples, Attributes)**
  
  IF all examples have the same label THEN RETURN (leaf node with label)
  
  ELSE
  
  IF set of attributes is empty THEN RETURN (leaf with majority label)
  
  ELSE
  
  Choose best attribute \(A\) as root
  
  FOR each value \(v\) of \(A\)
  
  Create a branch out of the root for the condition \(A = v\)
  
  IF \(\{x \in \text{Examples}: x.A = v\} = \emptyset\) THEN RETURN (leaf with majority label)
  
  ELSE Build-DT \(\{x \in \text{Examples}: x.A = v\}, \text{Attributes} \setminus \{A\}\)
  
- **But Which Attribute Is Best?**

  - \(A_1^{[29+, 35-]}, [21+, 5-]\)
  - \(A_2^{[29+, 35-]}, [8+, 30-]\)
  - \(A_3^{[18+, 33-]}, [11+, 2-]\)

**Choosing “Best” Root Attribute**

- **Objective**
  
  – Construct a decision tree that is as small as possible (Occam’s Razor)
  
  – Subject to: consistency with labels on training data

- **Obstacles**
  
  – Finding minimal consistent hypothesis (i.e., decision tree) is \(\mathcal{NP}\)-hard
  
  – Recursive algorithm (Build-DT)
  
  - A greedy heuristic search for a simple tree
  
  - Cannot guarantee optimality

- **Main Decision: Next Attribute to Condition On**
  
  – Want: attributes that split examples into sets, each relatively pure in one label
  
  – Result: closer to a leaf node
  
  – Most popular heuristic
  
  - Developed by J. R. Quinlan
  
  - Based on information gain
  
  - Used in ID3 algorithm
**Entropy:**

**INTUITIVE NOTION**

- **A Measure of Uncertainty**
  - **The Quantity**
    - Purity: how close a set of instances is to having just one label
    - Impurity (disorder): how close it is to total uncertainty over labels
  - **The Measure: Entropy**
    - Directly proportional to impurity, uncertainty, irregularity, surprise
    - Inversely proportional to purity, certainty, regularity, redundancy

- **Example**
  - For simplicity, assume $H = \{0, 1\}$, distributed according to $Pr(y)$
    - Can have (more than 2) discrete class labels
    - Continuous random variables: differential entropy
  - Optimal purity for $y$: either
    - $Pr(y = 0) = 1$, $Pr(y = 1) = 0$
    - $Pr(y = 1) = 1$, $Pr(y = 0) = 0$
  - What is the least pure probability distribution?
    - $Pr(y = 0) = 0.5$, $Pr(y = 1) = 0.5$
    - Corresponds to maximum impurity/uncertainty/irregularity/surprise
    - Property of entropy: concave function ("concave downward")

**Entropy:**

**INFORMATION THEORETIC DEFINITION [1]**

- **Components**
  - $D$: set of examples $\{<x_1, c(x_1)>, <x_2, c(x_2)>, \ldots, <x_m, c(x_m)>\}$
  - $p_+ = Pr(c(x) = +)$, $p_- = Pr(c(x) = -)$

- **Definition**
  - $H$ is defined over a probability density function $p$
  - $D$: examples whose frequency of $+$ and $-$ indicates $p_+, p_-$ for observed data
  - The entropy of $D$ relative to $c$ is:
    \[
    H(D) = -p_+ \log_b(p_+) - p_- \log_b(p_-)
    \]

- **What Units is $H$ Measured In?**
  - Depends on base $b$ of log (bits for $b = 2$, nats for $b = e$, etc.)
  - Single bit required to encode each example in worst case ($p_+ = 0.5$)
  - If there is less uncertainty (e.g., $p_+ = 0.8$), we can use less than 1 bit each
**Entropy:** Information Theoretic Definition [2]

- **Partitioning on Attribute Values**
  - Recall: a partition of $D$ is a collection of disjoint subsets whose union is $D$
  - Goal: measure the uncertainty removed by splitting on the value of attribute $A$

- **Definition**
  - The information gain of $D$ relative to attribute $A$ is the expected reduction in entropy due to splitting (“sorting”) on $A$:
    $$
    Gain(D, A) = H(D) - \sum_{v \in \text{values}(A)} \frac{|D_v|}{|D|} H(D_v)
    $$
  - Idea: partition on $A$; scale entropy to the size of each subset $D_v$

- **Which Attribute Is Best?**
  - Table:
    | Day  | Outlook | Temperature | Humidity | Wind  | PlayTennis? |
    |------|---------|-------------|----------|-------|-------------|
    | 1    | Sunny   | Hot         | High     | Light | No          |
    | 2    | Sunny   | Hot         | High     | Strong| No          |
    | 3    | Overcast| Hot         | High     | Light | Yes         |
    | 4    | Rain    | Mild        | High     | Light | Yes         |
    | 5    | Rain    | Cool        | Normal   | Light | Yes         |
    | 6    | Rain    | Cool        | Normal   | Strong| No          |
    | 7    | Overcast| Cool        | Normal   | Strong| Yes         |
    | 8    | Sunny   | Mild        | High     | Light | No          |
    | 9    | Sunny   | Cool        | Normal   | Light | Yes         |
    | 10   | Rain    | Mild        | Normal   | Light | Yes         |
    | 11   | Sunny   | Mild        | Normal   | Strong| Yes         |
    | 12   | Overcast| Mild        | High     | Strong| Yes         |
    | 13   | Overcast| Hot         | Normal   | Light | Yes         |
    | 14   | Rain    | Mild        | High     | Strong| No          |

- **ID3 Build-DT using Gain(·)**
- **How Will ID3 Construct A Decision Tree?**
• Selecting The Root Attribute

- Prior (unconditioned) distribution: 9+, 5-
  - $H(D) = -(9/14) \log_2 (9/14) - (5/14) \log_2 (5/14) \text{ bits} = 0.94 \text{ bits}$
  - $H(D, \text{Humidity} = \text{High}) = -(3/7) \log_2 (3/7) - (4/7) \log_2 (4/7) = 0.985 \text{ bits}$
  - $H(D, \text{Humidity} = \text{Normal}) = -(6/7) \log_2 (6/7) - (1/7) \log_2 (1/7) = 0.592 \text{ bits}$
  - $\text{Gain}(D, \text{Humidity}) = 0.94 - (7/14) \times 0.985 + (7/14) \times 0.592 = 0.151 \text{ bits}$
- $\text{Gain}(D, \text{Wind}) = 0.94 - (8/14) \times 0.811 + (6/14) \times 1.0 = 0.048 \text{ bits}$

• Selecting The Root Attribute

- Prior (unconditioned) distribution: 9+, 5-
  - $H(D) = -(9/14) \log_2 (9/14) - (5/14) \log_2 (5/14) \text{ bits} = 0.94 \text{ bits}$
  - $H(D, \text{Humidity} = \text{High}) = -(3/7) \log_2 (3/7) - (4/7) \log_2 (4/7) = 0.985 \text{ bits}$
  - $H(D, \text{Humidity} = \text{Normal}) = -(6/7) \log_2 (6/7) - (1/7) \log_2 (1/7) = 0.592 \text{ bits}$
  - $\text{Gain}(D, \text{Humidity}) = 0.94 - (7/14) \times 0.985 + (7/14) \times 0.592 = 0.151 \text{ bits}$
- $\text{Gain}(D, \text{Wind}) = 0.94 - (8/14) \times 0.811 + (6/14) \times 1.0 = 0.048 \text{ bits}$

• Selecting The Root Attribute

- Prior (unconditioned) distribution: 9+, 5-
  - $H(D) = -(9/14) \log_2 (9/14) - (5/14) \log_2 (5/14) \text{ bits} = 0.94 \text{ bits}$
  - $H(D, \text{Humidity} = \text{High}) = -(3/7) \log_2 (3/7) - (4/7) \log_2 (4/7) = 0.985 \text{ bits}$
  - $H(D, \text{Humidity} = \text{Normal}) = -(6/7) \log_2 (6/7) - (1/7) \log_2 (1/7) = 0.592 \text{ bits}$
  - $\text{Gain}(D, \text{Humidity}) = 0.94 - (7/14) \times 0.985 + (7/14) \times 0.592 = 0.151 \text{ bits}$
- $\text{Gain}(D, \text{Wind}) = 0.94 - (8/14) \times 0.811 + (6/14) \times 1.0 = 0.048 \text{ bits}$

• Selecting The Root Attribute

- Prior (unconditioned) distribution: 9+, 5-
  - $H(D) = -(9/14) \log_2 (9/14) - (5/14) \log_2 (5/14) \text{ bits} = 0.94 \text{ bits}$
  - $H(D, \text{Humidity} = \text{High}) = -(3/7) \log_2 (3/7) - (4/7) \log_2 (4/7) = 0.985 \text{ bits}$
  - $H(D, \text{Humidity} = \text{Normal}) = -(6/7) \log_2 (6/7) - (1/7) \log_2 (1/7) = 0.592 \text{ bits}$
  - $\text{Gain}(D, \text{Humidity}) = 0.94 - (7/14) \times 0.985 + (7/14) \times 0.592 = 0.151 \text{ bits}$
- $\text{Gain}(D, \text{Wind}) = 0.94 - (8/14) \times 0.811 + (6/14) \times 1.0 = 0.048 \text{ bits}$

• Selecting The Next Attribute (Root of Subtree)

- Continue until every example is included in path or purity = 100%
- What does purity = 100% mean?
- Can $\text{Gain}(D, A) < 0$?
• **Selecting The Next Attribute (Root of Subtree)**

- Convention: $\log (0/a) = 0$
- $\text{Gain}(\text{Sunny, Humidity}) = 0.97 - (3/5) \times 0 - (2/5) \times 0 = 0.97 \text{ bits}$
- $\text{Gain}(\text{Sunny, Wind}) = 0.97 - (2/5) \times 1 - (3/5) \times 0.92 = 0.02 \text{ bits}$
- $\text{Gain}(\text{Sunny, Temperature}) = 0.57 \text{ bits}$

• **Top-Down Induction**

- For discrete-valued attributes, terminates in $O(n)$ splits
- Makes at most one pass through data set at each level (why?)

---

### Constructing Decision Tree

#### For **PlayTennis using ID3** [3]

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Overcast</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
</tbody>
</table>

#### For **PlayTennis using ID3** [4]

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Overcast</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
</tbody>
</table>

---

### Construction of Decision Trees

#### For **PlayTennis using ID3** [3]

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14

#### For **PlayTennis using ID3** [4]

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14

---

### Decision Tree Structure

- **Outlook?**
  - Sunny
  - Overcast
- **Humidity?**
  - Yes
  - No
- **Wind?**
  - Yes
  - No
- **Rain?**
  - Yes
  - No

---

### Decision Tree Evaluation

- **Day:** Sunny
- **Temperature:** Hot
- **Humidity:** High
- **Wind:** Light
  - **PlayTennis:** No

---

### Decision Tree Example

- **Day:** Sunny
- **Temperature:** Hot
- **Humidity:** High
- **Wind:** Light
  - **PlayTennis:** No

---

### Decision Tree Summary

- **Top-Down Induction**
  - For discrete-valued attributes, terminates in $O(n)$ splits
  - Makes at most one pass through data set at each level (why?)
Search Problem
- Conduct a search of the space of decision trees, which can represent all possible discrete functions
  - Pros: expressiveness; flexibility
  - Cons: computational complexity; large, incomprehensible trees (next time)
- Objective: to find the best decision tree (minimal consistent tree)
- Obstacle: finding this tree is NP-hard
- Tradeoff
  - Use heuristic (figure of merit that guides search)
  - Use greedy algorithm
  - Aka hill-climbing (gradient "descent") without backtracking

Statistical Learning
- Decisions based on statistical descriptors \( p_+, p_- \) for subsamples \( D_x \)
- In ID3, all data used
- Robust to noisy data

Hypothesis Space Search

Inductive Bias in ID3
- Heuristic: Search :: Inductive Bias: Inductive Generalization
  - \( H \) is the power set of instances in \( X \)
  - \( \Rightarrow \) Unbiased? Not really...
    - Preference for short trees (termination condition)
    - Preference for trees with high information gain attributes near the root
    - \( Gain() \): a heuristic function that captures the inductive bias of ID3
  - Bias in ID3
    - Preference for some hypotheses is encoded in heuristic function
    - Compare: a restriction of hypothesis space \( H \) (previous discussion of propositional normal forms: \( k\)-CNF, etc.)
- Preference for Shortest Tree
  - Prefer shortest tree that fits the data
  - An Occam’s Razor bias: shortest hypothesis that explains the observations
### Terminology

- **Decision Trees (DTs)**
  - Boolean DTs: target concept is binary-valued (i.e., Boolean-valued)
  - Building DTs
    - Histogramming: method of vector quantization (encoding input using bins)
    - Discretization: continuous input into discrete (e.g., histogramming)

- **Entropy and Information Gain**
  - Entropy $H(D)$ for data set $D$ relative to implicit concept $c$
  - Information gain $Gain(D, A)$ for data set partitioned by attribute $A$
  - Impurity, uncertainty, irregularity, surprise vs. purity, certainty, regularity, redundancy

- **Heuristic Search**
  - Algorithm $Build-DT$: greedy search (hill-climbing without backtracking)
  - $ID3$ as $Build-DT$ using the heuristic $Gain(·)$
  - Heuristic: Search :: Inductive Bias : Inductive Generalization

- **MLC++ (Machine Learning Library in C++)**
  - Data mining libraries (e.g., MLC++) and packages (e.g., MineSet)
  - Irvine Database: the Machine Learning Database Repository at UCI

### Summary Points

- **Decision Trees (DTs)**
  - Can be boolean ($c(x) \in \{+, -\}$) or range over multiple classes
  - When to use DT-based models

- **Generic Algorithm** $Build-DT$: Top Down Induction
  - Calculating best attribute upon which to split
  - Recursive partitioning

- **Entropy and Information Gain**
  - Goal: to measure uncertainty removed by splitting on a candidate attribute $A$
    - Calculating information gain (change in entropy)
    - Using information gain in construction of tree
  - $ID3$ $Build-DT$ using $Gain(·)$

- **ID3 as Hypothesis Space Search (in State Space of Decision Trees)**

- **Heuristic Search and Inductive Bias**

- **Data Mining using MLC++ (Machine Learning Library in C++)**

- **Next: More Biases (Occam’s Razor); Managing DT Induction**
Human Brains

- Neuron switching time: ~ 0.001 (10^{-3}) second
- Number of neurons: ~10-100 billion (10^{10} – 10^{11})
- Connections per neuron: ~10-100 thousand (10^{4} – 10^{5})
- Scene recognition time: ~0.1 second
- 100 inference steps doesn’t seem sufficient! → highly parallel computation

Definitions of Artificial Neural Networks (ANNs)

- “… a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes.” - DARPA (1988)
- NN FAQ List: [http://www.ci.tuwien.ac.at/docs/services/nnfaq/FAQ.html](http://www.ci.tuwien.ac.at/docs/services/nnfaq/FAQ.html)

Properties of ANNs

- Many neuron-like threshold switching units
- Many weighted interconnections among units
- Highly parallel, distributed process
- Emphasis on tuning weights automatically

When to Consider Neural Networks

- Input: High-Dimensional and Discrete or Real-Valued
  - e.g., raw sensor input
  - Conversion of symbolic data to quantitative (numerical) representations possible
- Output: Discrete or Real Vector-Valued
  - e.g., low-level control policy for a robot actuator
  - Similar qualitative/quantitative (symbolic/numerical) conversions may apply
- Data: Possibly Noisy
- Target Function: Unknown Form
- Result: Human Readability Less Important Than Performance
  - Performance measured purely in terms of accuracy and efficiency
  - Readability: ability to explain inferences made using model; similar criteria
- Examples
  - Speech phoneme recognition [Waibel, Lee]
  - Image classification [Kanade, Baluja, Rowley, Frey]
  - Financial prediction
Autonomous Learning Vehicle in a Neural Net (ALVINN)

Pomerleau et al
- Drives 70mph on highways

Perceptron: Single Neuron Model
- aka Linear Threshold Unit (LTU) or Linear Threshold Gate (LTG)
- Net input to unit: defined as linear combination
  $$\text{net} = \sum_{i=0}^{n} w_i x_i$$
- Output of unit: threshold (activation) function on net input (threshold $\theta = w_0$)
  $$\sigma(x) = \text{sgn}(\hat{\mathbf{w}} \cdot \mathbf{x}) = \begin{cases} 1 & \text{if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 & \text{otherwise} \end{cases}$$
  Vector notation: $$\sigma(\hat{x}) = \text{sgn}(\hat{\mathbf{w}} \cdot \mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} > 0 \\ -1 & \text{otherwise} \end{cases}$$

Perceptron Networks
- Neuron is modeled using a unit connected by weighted links $w_i$ to other units
- Multi-Layer Perceptron (MLP): next lecture
Perceptron: Can Represent Some Useful Functions

- LTU emulation of logic gates (McCulloch and Pitts, 1943)
- e.g., What weights represent \( g(x_1, x_2) = \text{AND}(x_1, x_2) \)? \( \text{OR}(x_1, x_2) \)? \( \text{NOT}(x) \)?

Some Functions Not Representable

- e.g., not linearly separable
- Solution: use networks of perceptrons (LTUs)

Learning Rules for Perceptrons

- Learning Rule = Training Rule
  - Not specific to supervised learning
  - Context: updating a model
- Hebbian Learning Rule (Hebb, 1949)
  - Idea: if two units are both active (“firing”), weights between them should increase
  - \( w_{ij} = w_{ij} + r o_i o_j \) where \( r \) is a learning rate constant
  - Supported by neuropsychological evidence
- Perceptron Learning Rule (Rosenblatt, 1959)
  - Idea: when a target output value is provided for a single neuron with fixed input, it can incrementally update weights to learn to produce the output
  - Assume binary (boolean-valued) input/output units; single LTU
  - \( w_t = w_i + \Delta w_i \)
  - \( \Delta w_i = r(t - o)x_i \)
  - where \( t = c(x) \) is target output value, \( o \) is perceptron output, \( r \) is small learning rate constant (e.g., 0.1)
  - Can prove convergence if \( D \) linearly separable and \( r \) small enough
Simple Gradient Descent Algorithm

- Applicable to concept learning, symbolic learning (with proper representation)

**Algorithm Train-Perceptron** ($D = \{<x, t(x) = c(x)>\}$)

- Initialize all weights $w_i$ to random values
- **WHILE** not all examples correctly predicted **DO**
  - **FOR** each training example $x \in D$
    - Compute current output $o(x)$
    - **FOR** $i = 1$ to $n$
      - $w_i \leftarrow w_i + r(t - o)x_i$ // perceptron learning rule

Perceptron Learnability

- Recall: can only learn $h \in \mathcal{H}$ - i.e., linearly separable (LS) functions
- Minsky and Papert, 1969: demonstrated representational limitations
  - e.g., parity ($n$-attribute XOR: $x_1 \oplus x_2 \oplus \ldots \oplus x_n$)
  - e.g., symmetry, connectedness in visual pattern recognition
  - Influential book *Perceptrons* discouraged ANN research for ~10 years
- NB: $64K$ question - “Can we transform learning problems into LS ones?”

**Linear Separators**

- **Functional Definition**
  - $f(x) = 1$ if $w_1 x_1 + w_2 x_2 + \ldots + w_n x_n \geq \theta$, 0 otherwise
  - $\theta$: threshold value

- **Linearly Separable Functions**
  - NB: $D$ is LS does not necessarily imply $c(x) = f(x)$ is LS!
  - Disjunctions: $c(x) = x_1 \lor x_2 \lor \ldots \lor x_n$
  - $m$ of $n$: $c(x) = \text{at least } 3 \text{ of } (x_1, x_2, \ldots, x_n)$
  - Exclusive OR (XOR): $c(x) = x_1 \oplus x_2$
  - General DNF: $c(x) = T_1 \lor T_2 \lor \ldots \lor T_m = l_1 \land l_2 \land \ldots \land l_k$

- **Change of Representation Problem**
  - Can we transform non-LS problems into LS ones?
  - Is this meaningful? Practical?
  - Does it represent a significant fraction of real-world problems?
Perceptron Convergence

- **Perceptron Convergence Theorem**
  - Claim: If there exist a set of weights that are consistent with the data (i.e., the data is linearly separable), the perceptron learning algorithm will converge.
  - Proof: well-founded ordering on search region (“wedge width” is strictly decreasing) - see Minsky and Papert, 11.2-11.3
  - **Caveat 1:** How long will this take?
  - **Caveat 2:** What happens if the data is not LS?

- **Perceptron Cycling Theorem**
  - Claim: If the training data is not LS the perceptron learning algorithm will eventually repeat the same set of weights and thereby enter an infinite loop.
  - Proof: bound on number of weight changes until repetition; induction on $a$, the dimension of the training example vector - MP, 11.10

How to Provide More Robustness, Expressivity?

- Objective 1: develop algorithm that will find closest approximation (today)
- Objective 2: develop architecture to overcome representational limitation (next lecture)

Gradient Descent: Principle

- **Understanding Gradient Descent for Linear Units**
  - Consider simpler, unthresholded linear unit:
    $$ o(x) = \text{net}(x) = \sum_{i} w_i x_i $$
  - Objective: find “best fit” to $D$

- **Approximation Algorithm**
  - Quantitative objective: minimize error over training data set $D$
  - Error function: sum squared error (SSE)
    $$ E[w] = \text{error}_{\Omega}[w] = \frac{1}{2} \sum_{x \in D} (t(x) - o(x))^2 $$

- **How to Minimize?**
  - Simple optimization
  - Move in direction of steepest gradient in weight-error space
    - Computed by finding tangent
    - i.e. partial derivatives (of $E$) with respect to weights ($w$)
Gradient Descent: Derivation of Delta/LMS (Widrow-Hoff) Rule

**Definition: Gradient**

\[
\nabla E(\mathbf{w}) = \left[ \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n} \right]
\]

**Modified Gradient Descent Training Rule**

\[
\Delta \mathbf{w} = -r \nabla E(\mathbf{w})
\]

\[
\Delta w_i = -r \frac{\partial E}{\partial w_i}
\]

\[
\frac{\partial E}{\partial w_i} = \frac{1}{2} \sum_{n \in D} (t(x) - o(x))^2 = \frac{1}{2} \sum_{n \in D} \left[ \frac{\partial}{\partial w_i} (t(x) - o(x))^2 \right]
\]

\[
= \frac{1}{2} \sum_{n \in D} \left[ 2(t(x) - o(x)) \frac{\partial}{\partial w_i} (t(x) - o(x)) \right] = \sum_{n \in D} \left[ (t(x) - o(x)) \frac{\partial}{\partial w_i} (t(x) - o(x)) \right]
\]

**Algorithm Gradient-Descent \((D, r)\)**

- Each training example is a pair of the form \((x, t(x))\), where \(x\) is the vector of input values and \(t(x)\) is the output value, \(r\) is the learning rate (e.g., 0.05)
- Initialize all weights \(w_i\) to (small) random values
- Until the termination condition is met, do
  - Initialize each \(\Delta w_i\) to zero
  - For each \((x, t(x))\) in \(D\), do
    - Input the instance \(x\) to the unit and compute the output \(o\)
    - For each linear unit weight \(w_i\), do
      - \(\Delta w_i \leftarrow \Delta w_i + r (t(x) - o(x)) x_i\)
      - \(w_i \leftarrow w_i + \Delta w_i\)
  - Return final \(w\)

**Mechanics of Delta Rule**

- Gradient is based on a derivative
- Significance: later, will use nonlinear activation functions (aka transfer functions, squashing functions)
• **LS Concepts: Can Achieve Perfect Classification**
  - Example A: perceptron training rule converges
• **Non-LS Concepts: Can Only Approximate**
  - Example B: not LS; delta rule converges, but can’t do better than 3 correct
  - Example C: not LS; better results from delta rule
• **Weight Vector** $w = \sum_{x \in D}$
  - Perceptron: minimize $w$
  - Delta Rule: minimize $\frac{\partial E}{\partial w}$

**Gradient Descent:**
Perceptron Rule versus Delta/LMS Rule

**Example A**

**Example B**

**Example C**

**Review: Backprop, Feedforward**

• **Intuitive Idea:** Distribute *Blame* for Error to Previous Layers

**Algorithm Train-by-Backprop (D, r)**
  - Each training example is a pair of the form $(x, t(x))$, where $x$ is the vector of input values and $t(x)$ is the output value, $r$ is the learning rate (e.g., 0.05)
  - Initialize all weights $w$ to (small) random values
  - UNTIL the termination condition is met, DO
    - FOR each $(x, t(x))$ in $D$, DO
      - Input the instance $x$ to the unit and compute the output $o(x) = \sigma(\text{net}(x))$
      - FOR each output unit $k$, DO
        - $\delta_k = o_k(x)(1 - o_k(x))(t_k(x) - o_k(x))$
        - FOR each hidden unit $j$, DO
          - $\delta_j = h_j(x)(1 - h_j(x)) \sum_{\text{output}} v_{k,j} \delta_k$
          - Update each $w = u_{i,j}$ ($a = h_j$) or $w = v_{j,k}$ ($a = o_k$)
      - RETURN final $u, v$
**Review: Derivation of Backprop**

- **Recall: Gradient of Error Function**
  $\nabla E[\theta] = \left[ \frac{\partial E}{\partial \theta_1}, \frac{\partial E}{\partial \theta_2}, ..., \frac{\partial E}{\partial \theta_n} \right]$

- **Gradient of Sigmoid Activation Function**
  $\frac{\partial E}{\partial w_i} \bigg|_{\theta} = \frac{1}{2} \sum_{(x,y) \in D} (t(x) - o(x))^2$

  $= \frac{1}{2} \sum_{(x,y) \in D} \left[ 2(t(x) - o(x)) \frac{\partial (t(x) - o(x))}{\partial w_i} \right]$

  $= \sum_{(x,y) \in D} \left[ (t(x) - o(x)) \frac{\partial o(x)}{\partial w_i} \right]$

- **But We Know:**
  $\frac{\partial o(x)}{\partial \text{net}(x)} = o(x)(1-o(x))$

  $\frac{\partial \text{net}(x)}{\partial x_j} = x_j$

- **So:**
  $\frac{\partial E}{\partial w_i} \bigg|_{\theta} = \sum_{(x,y) \in D} \left[ (t(x) - o(x)) \frac{\partial o(x)}{\partial \text{net}(x)} \right] \frac{\partial \text{net}(x)}{\partial x_j} x_j$