15-381: AI

Decision Tree Learning – Part I

October 18, 2009

Manuela Veloso

Chapter 18, Russell and Norvig

Thanks to all past instructors

Carnegie Mellon
Outline

• Learning agents
• Inductive learning
• Decision tree learning
In general, it is hard to define learning, as we don't really know what learning is.

<table>
<thead>
<tr>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Definition?...</td>
</tr>
<tr>
<td>• Gathering more knowledge</td>
</tr>
<tr>
<td>– “Knowing more than was known before learning”</td>
</tr>
<tr>
<td>• Learning “substitutes” the need to model a priori</td>
</tr>
<tr>
<td>• Experience, feedback, refinement</td>
</tr>
<tr>
<td>• Learning modifies the agent’s decision mechanisms to improve performance</td>
</tr>
</tbody>
</table>
In this example, the learning agent is something that learns by observing and interacting with the environment.
Magic == lots of hacks

Need a hypothesis representation that is general enough to express what you want to learn, but specific enough that the search for the correct hypothesis isn't too large.

### Learning “Element”

- A bit of “magic”:
  - Which **components** of the performance element are to be learned
  - What **feedback** is available to learn these components
  - What **representation** is used for the components

- Type of feedback:
  - **Supervised learning**: correct labels/answers
  - **Unsupervised learning**: correct labels/answers missing
  - **Reinforcement learning**: occasional rewards
Inductive Learning

- Simplest form: learn a function from \textbf{examples}

  $f$ is the \textbf{target function}

  An \textbf{example} is a pair $(x, f(x))$ -- \textit{supervised learning}

  Problem: find a \textbf{hypothesis} $h$
  such that $h \approx f$
  given a \textbf{training set} of examples
Inductive Learning Method

- Construct/adjust $h$ to agree with $f$ on training set
- $h$ is consistent if it agrees with $f$ on all examples
- E.g., curve fitting:

![Diagram showing curve fitting with points and a curve]
Inductive Learning Method

- Construct/adjust $h$ to agree with $f$ on training set
- $h$ is consistent if it agrees with $f$ on all examples
- E.g., curve fitting

\[ f(x) \]

\[ x \]
Inductive Learning Method

- Construct/adjust $h$ to agree with $f$ on training set
- $h$ is consistent if it agrees with $f$ on all examples
- 
- E.g., curve fitting:
- 

\[ f(x) \]

\[ x \]
Inductive Learning Method

- Construct/adjust $h$ to agree with $f$ on training set
- $h$ is consistent if it agrees with $f$ on all examples
- E.g., curve fitting:

![Graph showing curve fitting](image)
Given any number of points, it is possible to find a hypothesis that is consistent for every training point. However we want the hypothesis to also predict points that were not training points. This slide is an example of overfitting. Yes, it is consistent with ever training example, but it is a poor predictor of points not in the training data set.
Inductive Learning Method - Bias

- Hypothesis “form” – bias on the learning outcome
- Ockham’s razor: prefer the simplest hypothesis consistent with data

There is an implicit bias in the hypothesis’s form. For example, if your hypothesis is a 4th degree polynomial, where you are learning the coefficient of each term, you can only represent curves of degree 4 and less.

Ockham's razor: when two hypothesis perform similarly, prefer the simpler one (ie if a line and a curve both fit points, then prefer a line).
Attribute-Based Data Sets

- Examples described by attribute values (Boolean, discrete, continuous)
- Function is class, wait/not wait for table at restaurant

<table>
<thead>
<tr>
<th>Example</th>
<th>Attr</th>
<th>Bar</th>
<th>Fri</th>
<th>Hun</th>
<th>Pat</th>
<th>Price</th>
<th>Rain</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
<th>Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_1</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>0-10</td>
<td>T</td>
</tr>
<tr>
<td>X_2</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>30-60</td>
<td>F</td>
</tr>
<tr>
<td>X_3</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>Some</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>0-10</td>
<td>T</td>
</tr>
<tr>
<td>X_4</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>10-30</td>
<td>T</td>
</tr>
<tr>
<td>X_5</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>X_6</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Italian</td>
<td>0-10</td>
<td>T</td>
</tr>
<tr>
<td>X_7</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>0-10</td>
<td>F</td>
</tr>
<tr>
<td>X_8</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Thai</td>
<td>0-10</td>
<td>T</td>
</tr>
<tr>
<td>X_9</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>X_{10}</td>
<td>T</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>F</td>
<td>Italian</td>
<td>10-30</td>
<td>F</td>
</tr>
<tr>
<td>X_{11}</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>0-10</td>
<td>F</td>
</tr>
<tr>
<td>X_{12}</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>30-60</td>
<td>T</td>
</tr>
</tbody>
</table>

- Classification of examples is positive (T) or negative (F)

How should data be represented? Here, each row is one example, each column is an attribute.

Each attribute takes a value. Boolean attributes take true/false, etc. Some attributes are discrete, some are continuous. As you'll see later on, continuous valued attributes can complicate things.
A learner should learn that the type, director and your mood are important to whether or not you liked the movie, while the company is not so important.
Input Data \[ \rightarrow \text{Attributes} \]

\[ X_1 = x_1 \]

\[ \vdots \]

\[ X_M = x_M \]

\[ \rightarrow \text{Classifier} \]

\[ \rightarrow \text{Class prediction} \]

\[ Y = y \]

\[ \text{Training data} \]
This is a decision tree to decide whether or not you are going to wait for a table at a restaurant.

Here, hypotheses are in the form of decision trees. There are many possible decision trees (hypotheses). You are searching for the simplest decision tree (hypothesis) that is most consistent with the training data

Is this example the simplest?
Expressiveness

• Decision trees can express any function of the input attributes
  – E.g., for Boolean functions, truth table row → path to leaf

• Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless $f$ nondeterministic in $x$)
  – But... it probably won't generalize to new examples – goal of learning...

• Goal: Find more “compact” decision trees
Trying to classify examples as P or G based on their attributes.

If tree height is what determines which hypothesis is simpler, there is no simpler, consistent hypothesis.

---

**Building a Decision Tree**

- Three variables:
  - Hair = {blond, dark}
  - Height = {tall, short}
  - Country = {Gromland, Polvia}

Training data:
- (B, T, P)
- (B, T, P)
- (B, S, G)
- (D, S, G)
- (D, T, G)
- (B, S, G)

---

P:2 G:4
Hair = B?  
Hair = D?

P:2 G:2
Height = T?  
Height = S?

P:0 G:2
P:2 G:0

P:0 G:2
At each level of the tree, we split the data according to the value of one of the attributes.

After enough splits, only one output class is represented in the node—a pure node → a terminal leaf of the tree.

‘G’ is the output for this node.
The class of a new input can be classified by following the tree all the way down to a leaf and by reporting the output of the leaf. For example:

(B,T) is classified as P; (D,S) is classified as G
General Decision Tree

\[ X_1 = \text{first possible value for } X_1? \]

\[ X_1 = \text{nth possible value for } X_1? \]

\[ \cdots \]

Output class \( Y = y_1 \)

\[ X_j = \text{ith possible value for } X_j? \]

\[ \cdots \]

Output class \( Y = y_c \)
Basic Questions

- How **to choose the attribute/value to split** on at each level of the tree?
- When **to stop splitting**? When should a node be declared a leaf?
- If a leaf node is impure, how should the **class label be assigned**?
- If the tree is too large, how can it be **pruned**?
### Set of Training Examples

- **Type:** drama, comedy, thriller
- **Company:** MGM, Columbia
- **Director:** Bergman, Spielberg, Hitchcock
- **Mood:** stressed, relaxed, normal

<table>
<thead>
<tr>
<th>Movie</th>
<th>Type</th>
<th>Company</th>
<th>Director</th>
<th>Mood</th>
<th>Likes-movie?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_1$</td>
<td>thriller</td>
<td>MGM</td>
<td>Bergman</td>
<td>normal</td>
<td>No</td>
</tr>
<tr>
<td>$m_2$</td>
<td>comedy</td>
<td>Columbia</td>
<td>Spielberg</td>
<td>stressed</td>
<td>Yes</td>
</tr>
<tr>
<td>$m_3$</td>
<td>comedy</td>
<td>MGM</td>
<td>Spielberg</td>
<td>relaxed</td>
<td>No</td>
</tr>
<tr>
<td>$m_4$</td>
<td>thriller</td>
<td>MGM</td>
<td>Bergman</td>
<td>relaxed</td>
<td>No</td>
</tr>
<tr>
<td>$m_5$</td>
<td>comedy</td>
<td>MGM</td>
<td>Hitchcock</td>
<td>normal</td>
<td>Yes</td>
</tr>
<tr>
<td>$m_6$</td>
<td>drama</td>
<td>Columbia</td>
<td>Borgman</td>
<td>relaxed</td>
<td>Yes</td>
</tr>
<tr>
<td>$m_7$</td>
<td>drama</td>
<td>Columbia</td>
<td>Bergman</td>
<td>normal</td>
<td>No</td>
</tr>
<tr>
<td>$m_8$</td>
<td>drama</td>
<td>MGM</td>
<td>Spielberg</td>
<td>stressed</td>
<td>No</td>
</tr>
<tr>
<td>$m_9$</td>
<td>drama</td>
<td>MGM</td>
<td>Hitchcock</td>
<td>normal</td>
<td>Yes</td>
</tr>
<tr>
<td>$m_{10}$</td>
<td>comedy</td>
<td>Columbia</td>
<td>Spielberg</td>
<td>relaxed</td>
<td>No</td>
</tr>
<tr>
<td>$m_{11}$</td>
<td>thriller</td>
<td>MGM</td>
<td>Spielberg</td>
<td>normal</td>
<td>No</td>
</tr>
<tr>
<td>$m_{12}$</td>
<td>thriller</td>
<td>Columbia</td>
<td>Hitchcock</td>
<td>relaxed</td>
<td>No</td>
</tr>
</tbody>
</table>
Choosing the “Best” Attribute

- Ideal attribute – partitions examples into all positive or all negative (or from the same class in each partition).
- Attribute that results in higher “discrimination.”

How good is the attribute “Type” of movie?

Any attribute splits the data (though some groups might be empty). Here, the type attribute splits the data into three groups.

The optimal attribute would split the data into pure nodes. Here, it would be great if just knowing the type of movie would tell you whether or not the movie was liked.
High entropy because all the information is mixed up, it is chaotic.
Low entropy because the information is organized and orderly.
Entropy (Shannon and Weaver 1949)

• In general, entropy is the average number of bits necessary to encode \( n \) values:

\[
H = -\sum_{i=1}^{n} P_i \log_2 P_i
\]

• \( P_i \) = probability of occurrence of value \( i \)
  - High entropy \( \rightarrow \) All the classes are (nearly) equally likely
  - Low entropy \( \rightarrow \) A few classes are likely; most of the classes are rarely observed

Know the formula for entropy, it is very, very useful!

Log base conversion from base \( a \) to base \( b \): \( \log_a(x) = \log_b(x)/\log_b(a) \)
Entropy for Attribute Choice

- Measure of information provided by the attribute
- Entropy of a set of examples $S$ as the information content of $S$.

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

where $p_i = \frac{|S_i|}{|S|}$

- $c$ classes, $S_i$ size of the data set for class $i$
Entropy

- Unit – 1 bit of information =
  - the information content of the actual answer when there are two possible answers equally probable.

\[ E(S) = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1 \]
Imagine having a basket full of red and black balls. If \( p \) is probability of being black, then when everything is red, \( p = 0 \) and entropy is 0. When everything is black, and \( p = 1 \), then entropy is 0. If half are red and half are black, \( p = 0.5 \) and entropy is 1.
4 positive examples and 8 negative examples. Entropy is still very high.

**Entropy – Example**

- \(| S | = 12, c = 2(+,-), | S_+ | = 4, | S_- | = 8\)

\[
E(S) = -\frac{4}{12} \log_2 \frac{4}{12} - \frac{8}{12} \log_2 \frac{8}{12} = 0.918
\]
Choosing the “Best” Attribute

- Attribute with **highest information gain** – expected reduction in entropy of the set $S$ if partitioned according to attribute $A$.

$$
\text{Gain}(S,A) = \text{Entropy}(S) - \sum_{j=1}^{\text{values}(A)} \frac{|S_{v_j}|}{|S|} \text{Entropy}(S_{v_j})
$$

$$
\text{Gain}(S, \text{Type}) = E(4^+, 8^-) - \frac{4}{12} E(0^+, 4^-) - \frac{4}{12} E(2^+, 2^-) - \frac{4}{12} E(2^+, 2^-)
$$

$$
= 0.252
$$

Split on the attribute that gives the most information gain (reduces the entropy the most). This is not necessarily optimal, but it is a good heuristic.
**Other Attributes**

\[ E(4^+, 8^-) - \frac{7}{12} E(2^+, 5^-) - \frac{5}{12} E(2^+, 3^-) \]
\[ = 0.0102 \]

\[ E(4^+, 8^-) - \frac{2}{12} E(1^+, 1^-) - \frac{5}{12} E(1^+, 4^-) - \frac{5}{12} E(2^+, 3^-) \]
\[ = 0.046 \]

\[ E(4^+, 8^-) - \frac{4}{12} E(1^+, 3^-) - \frac{5}{12} E(1^+, 4^-) - \frac{3}{12} E(2^+, 1^-) \]
\[ = 0.118 \]
Commitment to One Hypothesis

What is the next best attribute? Recursive.
Split Data and Continue

Gain($S_d$, Mood) = $E(2^+, 2^-) - \frac{1}{4} E(0^+, 1^-) - \frac{1}{4} E(1^+, 0^-) - \frac{3}{4} E(1^+, 1^-)$

= 0.5

Gain($S_d$, Company) = $E(2^+, 2^-) - \frac{2}{4} E(1^+, 1^-) - \frac{2}{4} E(1^+, 1^-)$

= 0

Gain($S_d$, Director) = $E(2^+, 2^-) - \frac{2}{4} E(1^+, 1^-) - \frac{1}{4} E(0^+, 1^-) - \frac{3}{4} E(1^+, 0^-)$

= 0.5
Learned Tree

You need some default answer for missing data.
Learn which values of each attribute is the best to split on

**Continuous Case - How to Split?**

- Two classes (red circles/green crosses)
- Two attributes: $X_1$ and $X_2$
- 11 points in training data
- Idea → Construct a decision tree such that the leaf nodes predict correctly the class for all the training examples
Good and Bad Splits

Good

Bad
This node is "pure" because there is only one class left → No ambiguity in the class label

This node is almost "pure" → Little ambiguity in the class label

These nodes contain a mixture of classes → Do not disambiguate between the classes
We want to find the most compact, smallest size tree (Occam’s razor), that classifies the training data correctly → We want to find the split choices that will get us the fastest to pure nodes.

This node is "pure" because there is only one class left → No ambiguity in the class label.

This node is almost "pure" → Little ambiguity in the class label.

These nodes contain a mixture of classes → Do not disambiguate between the classes.
\[
\text{IG} = H - \left( H_L \times \frac{4}{11} + H_R \times \frac{7}{11} \right)
\]

\[
\text{IG} = H - \left( H_L \times \frac{5}{11} + H_R \times \frac{6}{11} \right)
\]

\[
H_L = 0 \quad H_R = 0.58
\]

\[
H_L = 0.97 \quad H_R = 0.92
\]
Choose this split because the information gain is greater than with the other split.

- \(H = 0.99\)
  - \(IG = 0.62\)
    - \(H_L = 0\)
    - \(H_R = 0.58\)
  - \(IG = 0.052\)
    - \(H_L = 0.97\)
    - \(H_R = 0.92\)
More Complete Example

- Red circles = 20 training examples from class A
- Green crosses = 20 training examples from class B
- Attributes = $X_1$ and $X_2$ coordinates
Best split value (max Information Gain) for $X_1$ attribute: 0.24 with IG = 0.138
Best split value (max Information Gain) for $X_2$ attribute: 0.234 with IG = 0.202
Best $X_1$ split: 0.24, IG = 0.138
Best $X_2$ split: 0.234, IG = 0.202

Split on $X_2$ with 0.234

Total data points = 7
7 A
0 B

Total data points = 33
13 A
20 B
There is no point in splitting this node further since it contains only data from a single class → return it as a leaf node with output ‘A’

This node is not pure so we need to split further

Total data points = 7
7 A
0 B

Total data points = 33
13 A
20 B
Best split value (max Information Gain) for $X_1$ attribute: 0.22 with IG $\sim$ 0.182
Best split value (max Information Gain) for $X_2$ attribute: 0.75 with IG ~ 0.353
Best $X_1$ split: 0.22, IG = 0.182
Best $X_2$ split: 0.75, IG = 0.353

Split on $X_2$ with 0.75

Total data points = 26
6 A
20 B

Total data points = 7
7 A
0 B
There is no point in splitting this node further since it contains only data from a single class → return it as a leaf node with output ‘A’
Each of the leaf nodes is pure → contains data from only one class
Final decision tree
Given an input $(X,Y) \rightarrow$
Follow the tree down to a leaf.
Return corresponding output class for this leaf

Example $(X,Y) = (0.5,0.5)$
Induction of Decision Trees

\textbf{ID3} (examples, attributes)
if there are no examples
then return default
else
  if all examples are members of the same class
  then return the class
  else
    if there are no attributes
      then return \text{Most\_Common\_Class (examples)}
    else
      \text{best\_attribute} \leftarrow \text{Choose\_Best (examples, attributes)}
      \text{root} \leftarrow \text{Create\_Node\_Test (best\_attribute)}
      for each value \( v_i \) of \text{best\_attribute}
        \text{examples}_{v_i} \leftarrow \text{subset of examples with best\_attribute} = v_i
        \text{subtree} \leftarrow \text{ID3(examples}_{v_i}, \text{attributes - best\_attribute})
      set subtree as a child of root with label \( v_i \)
return root
Basic Questions

• How to choose the attribute/value to split on at each level of the tree?
• When to stop splitting? When should a node be declared a leaf?
• If a leaf node is impure, how should the class label be assigned?
• If the tree is too large, how can it be pruned?
Comments on Tree Termination

- Zero entropy in data set, perfect classification
  - Type = thriller, 0+ 4
  - Type = drama, Mood = stressed, 0+ 1-
  - Type = drama, Mood = relaxed, 1+ 0-
  - Type = drama, Mood = normal, Director = Bergman, 0+ 1-
  - Type = drama, Mood = normal, Director = Hitchcock, 1+ 0-

- No examples available
  - Type = drama, Mood = normal, Director = Spielberg

- Indistinguishable data
  - Type = comedy, attribute Company:
    - m2, comedy, Columbia, Yes, and m10, comedy, Columbia, No
    - m3, comedy, MGM, No, and m5, comedy, MGM, Yes
  - Type = comedy, attribute Director:
    - m2, Spielberg, Yes, and m3, Spielberg, No
    - m5, Hitchcock, Yes, and m10, Spielberg, No
  - (Type = comedy, attribute Mood, could have split?).
Errors

Split labeled data $D$ into training and validation sets

- Training set error - fraction of training examples for which the decision tree disagrees with the true classification

- Validation set error - fraction of testing examples - from given labeled examples - for which the decision tree disagrees with the true classification
**Overfitting**

Tree too specialized – “perfectly” fit the training data.

A tree $T$ overfits the training data, if there is an alternate $T'$ such that

- for training set, error with $T <$ error with $T'$
- for complete $D$, error with $T >$ error with $T'$

- Node Pruning
- Rule Post-pruning
- Cross validation
Reduced-Error Pruning

- Remove subtree, make node a leaf node, assign the most common classification of the examples of that node.
- Check validation set error
- Continue pruning until error does not increase with respect to the error of the unpruned tree
- Rule post-pruning: represent tree as set of rules; remove rules independently; check validation set error; stop with same criteria
Other Issues

- Attributes with many values
  - Information gain may select it.
  - Consider splitting value and use the ratio between Gain and the splitting value

- Attributes with cost
  - Use other metrics

- Attributes with missing values
  - Infer most common value from examples at node
Summary

- Inductive learning – supervised learning – classification
- Decision trees represent hypotheses
- DT learning driven by information gain heuristic
- Recursive algorithm to build decision tree
- Next class:
  - More on continuous values
  - Missing, noisy attributes
  - Overfitting, pruning
  - Different attribute selection criteria