Decision Trees & Rule Induction

Michael van Lent
The Big Picture

- Problem
  - Classification
- Feedback
  - Supervised learning
  - Reinforcement learning
- Knowledge Representation
  - Decision tree
  - Rules
- Knowledge Source
  - Examples
Decision Trees

• Nodes represent attribute tests
  • One child for each possible value of the attribute

• Leaves represent classifications

• Classify by descending from root to a leaf
  • At root test attribute associated with root attribute test
  • Descend the branch corresponding to the instance’s value
  • Repeat for subtree rooted at the new node
  • When a leaf is reached return the classification of that leaf

• Decision tree is a disjunction of conjunctions of constraints on the attribute values of an instance
Example Problem

Classify how I should react to an object in the world

- **Facts about any given object include:**
  - Allegiance = <friendly, neutral, enemy>
  - Health = <low, medium, full>
  - Animate = <true, false>
  - RelativeHealth = <weaker, same, stronger>

- **Output categories include:**
  - Reaction = Attack
  - Reaction = Ignore
  - Reaction = Heal
  - Reaction = Eat
  - Reaction = Run

- <friendly, low, true, weaker> => Heal
- <neutral, low, true, same> => Heal
- <enemy, low, true, stronger> => Attack
- <enemy, medium, true, weaker> => Attack
Classifying with a Decision Tree

- Allegeance?
  - Friendly
  - Neutral
  - Enemy

- Health?
  - Low
    - Heal
  - Medium
  - Heal
  - Full
    - Heal
    - Ignore

- Attack?
  - Low
    - Heal
    - Ignore
  - Medium
    - Heal
    - Ignore
  - Full
    - Ignore
Classifying with a Decision Tree

![Decision Tree Diagram]

1. **Health?**
   - **Low**
   - **Medium**
   - **Full**

2. **Allegiance?**
   - **Friendly**
     - **Heal**
   - **Neutral**
     - **Heal**
     - **Ignore**
   - **Enemy**
     - **Heal**
     - **Ignore**

3. If **Full**, then **Ignore**.

**Attack**
Decision Trees are good when:

- Inputs are attribute-value pairs
  - With fairly small number of values
  - Numeric or continuous values cause problems
    - Can extend algorithms to learn thresholds
- Outputs are discrete output values
  - Again fairly small number of values
  - Difficult to represent numeric or continuous outputs
- Disjunction is required
  - Decision trees easily handle disjunction
- Training examples contain errors
  - Learning decision trees
  - More later
Learning Decision Trees

• Decision trees are usually learned by induction
  • Generalize from examples
  • Induction doesn’t guarantee correct decision trees

• Bias towards smaller decision trees
  • Occam’s Razor: Prefer simplest theory that fits the data
  • Too expensive to find the very smallest decision tree

• Learning is non-incremental
  • Need to store all the examples

• ID3 is the basic learning algorithm
  • C4.5 is an updated and extended version
Induction

• If X is true in every example X must always be true
  • More examples are better
  • Errors in examples cause difficulty
  • Note that induction can result in errors

• Inductive learning of Decision Trees
  • Create a decision tree that classifies the available examples
  • Use this decision tree to classify new instances
  • Avoid over fitting the available examples
    • One root to node path for each example
    • Perfect on the examples, not so good on new instances
Induction requires Examples

• Where do examples come from?
  • Programmer/designer provides examples
  • Observe a human’s decisions

• # of examples need depends on difficulty of concept
  • More is always better

• Training set vs. Testing set
  • Train on most (75%) of the examples
  • Use the rest to validate the learned decision trees
ID3 Learning Algorithm

- ID3 has two parameters
  - List of examples
  - List of attributes to be tested
- Generates tree recursively
  - Chooses attribute that best divides the examples at each step

ID3(examples, attributes)

if all examples in same category then
  return a leaf node with that category
if attributes is empty then
  return a leaf node with the most common category in examples
best = Choose-Attribute(examples, attributes)
tree = new tree with Best as root attribute test
foreach value $v_i$ of best
  examples$_i$ = subset of examples with best == $v_i$
  subtree = ID3(examples$_i$, attributes – best)
  add a branch to tree with best == $v_i$ and subtree beneath
return tree
Examples

- <friendly, low, true, weaker> => Heal
- <neutral, full, false, same> => Eat
- <enemy, low, true, weaker> => Eat
- <enemy, low, true, same> => Attack
- <neutral, low, true, weaker> => Heal
- <enemy, medium, true, stronger> => Run
- <friendly, full, true, same> => Ignore
- <neutral, full, true, stronger> => Ignore
- <enemy, full, true, same> => Run
- <enemy, medium, true, weaker> => Attack
- <friendly, full, true, weaker> => Ignore
- <neutral, full, false, stronger> => Ignore
- <friendly, medium, true, stronger> => Heal

- 13 examples
  - 3 Heal
  - 2 Eat
  - 2 Attack
  - 4 Ignore
  - 2 Run
Entropy

- Entropy: how “mixed” is a set of examples
  - All one category: Entropy = 0
  - Evenly divided: Entropy = \( \log_2(\# \text{ of examples}) \)
- Given S examples Entropy(S) = \( S - \sum p_i \log_2 p_i \)
  where \( p_i \) is the proportion of S belonging to class i
  - 13 examples with 3 heal, 2 attack, 2 eat, 4 ignore, 2 run
    - Entropy([3,2,2,4,2]) = 2.258
  - 13 examples with all 13 heal
    - Entropy ([13,0,0,0,0]) = 0
  - Maximum entropy is \( \log_2 5 = 2.322 \)
    - 5 is the number of categories
Information Gain

• Information Gain measures the reduction in Entropy
  • \( \text{Gain}(S, A) = \text{Entropy}(S) - \frac{S \cdot \text{Entropy}(S_v)}{S} \)

• Example: 13 examples: \( \text{Entropy}([3,2,2,4,2]) = 2.258 \)
  • Information gain of Allegiance = \(<\text{friendly, neutral, enemy}>\)
    • Allegiance = friendly for 4 examples \([2,0,0,2,0]\)
    • Allegiance = neutral for 4 examples \([1,1,0,2,0]\)
    • Allegiance = enemy for 5 examples \([0,1,2,0,2]\)
    • \( \text{Gain}(S, \text{Allegiance}) = 0.903 \)

• Information gain of Animate = \(<\text{true, false}>\)
  • Animate = true for 11 examples \([3,1,2,3,2]\)
  • Animate = false for 2 examples \([0,1,0,1,0]\)
  • \( \text{Gain}(S, \text{Animate}) = 0.216 \)

• Allegiance has a higher information gain than Animate
  • So choose allegiance as the next attribute to be tested
Learning Example

- Information gain of Allegiance
  - 0.903

- Information gain of Health
  - 0.853

- Information gain of Animate
  - 0.216

- Information gain of RelativeHealth
  - 0.442

- So Allegiance should be the root test
Decision tree so far

Allegiance?

Friendly  Neutral  Enemy

?  ?  ?
Allegiance = friendly

- Four examples have allegiance = friendly
  - Two categorized as Heal
  - Two categorized as Ignore
  - We’ll denote this now as [# of Heal, # of Ignore]
  - Entropy = 1.0

- Which of the remaining features has the highest info gain?
  - Health: low [1,0], medium [1,0], full [0,2] => Gain is 1.0
  - Animate: true [2,2], false [0,0] => Gain is 0
  - RelativeHealth: weaker [1,1], same [0,1], stronger [1,0] => Gain is 0.5

- Health is the best (and final) choice
Decision tree so far

Allegiance?

Friendly
- Health
  - Low
    - Heal
  - Medium
  - Full
    - Heal

Neutral

Enemy
- ?
- ?

Friendly
- ?
- Heal
- Ignore
Allegiance = enemy

- Five examples have allegiance = enemy
  - One categorized as Eat
  - Two categorized as Attack
  - Two categorized as Run
  - We’ll denote this now as [# of Eat, # of Attack, # of Run]
  - Entropy = 1.5

- Which of the remaining features has the highest info gain?
  - Health: low [1,1,0], medium [0,1,1], full [0,0,1] => Gain is 0.7
  - Animate: true [1,2,2], false [0,0,0] => Gain is 0
  - RelHealth: weaker [1,1,0], same [0,1,1], stronger [0,0,1] => Gain is 0.7

- Health and RelativeHealth are equally good choices
Decision tree so far

Allegiance?

Friendly
- Health
  - Low
    - Heal
  - Medium
  - Heal
  - High
    - Ignore

Neutral

Enemy
- Health
  - Low
    - ?
  - Medium
  - ?
  - Full
    - Run
Final Decision Tree

- Allegiance?
  - Friendly
  - Neutral
  - Enemy

- RelHealth
  - Health
    - Heal
      - Low
        - Heal
      - Medium
        - Heal
      - Full
        - Ignore
          - Low
            - Heal
          - Medium
            - Heal
          - Full
            - Ignore
            - Eat
              - Low
                - Eat
              - Medium
                - Eat
              - Full
                - Ignore
                - Run
                  - Attack
                    - Attack
                      - Attack
                      - Attack
                      - Attack
                      - Attack
                      - Run
                      - Run
Generalization

• Previously unseen examples can be classified
  • Each path through the decision tree doesn’t test every feature
  • <neutral, low, false, stronger> => Eat

• Some leaves don’t have corresponding examples
  • (Allegiance=enemy) & (Health=low) & (RelHealth=stronger)
  • Don’t have any examples of this case
  • Generalize from the closest example
  • <enemy, low, false, same> => Attack
  • Guess that: <enemy, low, false, stronger> => Attack
Decision trees in Black & White

• Creature learns to predict the player’s reactions
  • Instead of categories, range [-1 to 1] of predicted feedback
  • Extending decision trees for continuous values
    • Divide into discrete categories
    • ...

• Creature generates examples by experimenting
  • Try something and record the feedback (tummy rub, slap…)
  • Starts to look like reinforcement learning

• Challenges encountered
  • Ensuring everything that can be learned is reasonable
  • Matching actions with player feedback
Decision Trees and Rules

- Decision trees can easily be translated into rules
  - and vice versa

If (Allegiance=friendly) & ((Health=low) | (Health=medium)) then Heal
If (Allegiance=friendly) & (Health=high) then Ignore
If (Allegiance=neutral) & (Health=low) then Heal

... If (Allegiance=neutral) & (Health=low) then Heal

If (Allegiance=enemy) then Attack
Rule Induction

- **Specific to General Induction**
  - First example creates a very specific rule
  - Additional examples are used to generalize the rule
  - If rule becomes too general create a new, disjunctive rule

- **Version Spaces**
  - Start with a very specific rule and a very general rule
  - Each new example either
    - Makes the specific rule more general
    - Makes the general rule more specific
  - The specific and general rules meet at the solution
Learning Example

• First example: <friendly, low, true, weaker> => Heal
  • If (Allegiance=friendly) & (Health=low) & (Animate=true) & (RelHealth=weaker) then Heal

• Second example: <neutral, low, true, weaker> => Heal
  • If (Health=low) & (Animate=true) & (RelHealth=weaker) then Heal
    • Overgeneralization?
  • If ((Allegiance=friendly) | (Allegiance=neutral)) & (Health=low) & (Animate=true) & (RelHealth=weaker) then Heal

• Third example: <friendly, medium, true, stronger> => Heal
  • If ((Allegiance=friendly) | (Allegiance=neutral)) & ((Health=low) | (Health=medium)) & (Animate=true) & ((RelHealth=weaker) | (RelHealth=stronger)) then Heal
Advanced Topics

• **Boosting**
  • Manipulate the set of training examples
  • Increase the representation of incorrectly classified examples

• **Ensembles of classifiers**
  • Learn multiple classifiers (i.e. multiple decision trees)
    • All the classifiers vote on the correct answer (only one approach)
  • “Bagging”: break the training set into overlapping subsets
    • Learn a classifier for each subset
  • Learn classifiers using different subsets of features
    • Or different subsets of categories
  • Ensembles can be more accurate than a single classifier
Games that use inductive learning

• Decision Trees
  • Black & White

• Rules
Inductive Learning Evaluation

• **Pros**
  • Decision trees and rules are human understandable
  • Handle noisy data fairly well
  • Incremental learning
  • Online learning is feasible

• **Cons**
  • Need many, good examples
  • Overfitting can be an issue
  • Learned decision trees may contain errors

• **Challenges**
  • Picking the right features
  • Getting good examples
References

• Quinlan: Combining instance-based and model-based learning, 10th International Conference on Machine Learning, 1993.
• AI Game Programming Wisdom.
• AI Game Programming Wisdom 2.