Decision trees

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Announcement

• Term project:
  – Reports due on Wednesday, April 23 at 2pm
  – Project presentations:
    • When: Friday, April 25, 2003 at 1pm
    • Where: 5313 Sennott Square
    • 10 minutes ppt presentations
  – Example project reports are on the course web site.
Decision trees

- Back to the supervised learning
- An alternative approach to what we have seen so far:
  - Partition the input space to regions
  - Regress or classify independently in every region

Example:
Binary classification \{0,1\}
Binary attributes \(x_1, x_2, x_3\)
Decision trees

How to construct the decision tree?

- **Top-bottom algorithm:**
  - Find the best split condition (quantified based on the impurity measure)
  - Stops when no improvement possible

- **Impurity measure:**
  - Measures how well are the two classes separated
  - Ideally we would like to separate all 0s and 1

- **Splits of finite vs. continuous value attributes**
  Continuous value attributes conditions: \( x_3 \leq 0.5 \)

**Impurity measure**

Let \( |D| \) - Total number of data entries

\[ |D_i| \] - Number of data entries classified as \( i \)

\[ p_i = \frac{|D_i|}{|D|} \] - ratio of instances classified as \( i \)

- **Impurity measure** defines how well the classes are separated
- In general the impurity measure should satisfy:
  - Largest when data are split evenly for attribute values
    \[ p_i = \frac{1}{\text{number of classes}} \]
  - Should be 0 when all data belong to the same class
Impurity measures

• There are various impurity measures used in the literature
  – Entropy based measure (Quinlan, C4.5)
    \[ I(D) = Entropy(D) = -\sum_{i=1}^{k} p_i \log p_i \]
  
  Example for \( k=2 \)

  \[ \sum \]

  \[ = \]

  \[ \sum \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]

  \[ = \]
Decision tree learning

- **Greedy learning algorithm:**
  
  Repeat until no or small improvement in the purity
  
  - Find the attribute with the highest gain
  
  - Add the attribute to the tree and split the set accordingly

- Builds the tree in the top-down fashion
  
  - Gradually expands the leaves of the partially built tree
- The method is greedy
  
  - It looks at a single attribute and gain in each step
  
  - May fail when the combination of attributes is needed to improve the purity (parity functions)

**Limitations of greedy methods**

Cases in which a combination of two or more attributes improves the impurity
Decision tree learning

By reducing the impurity measure we can grow very large trees

**Problem: Overfitting**

- We may split and classify very well the training set, but we may do worse in terms of the generalization error

**Solutions to the overfitting problem:**

- **Solution 1.**
  - Prune branches of the tree built in the first phase
  - Use validation set to test for the overfit
- **Solution 2.**
  - Test for the overfit in the tree building phase
  - Stop building the tree when performance on the validation set deteriorates

---

Mixture of experts

- **Clustering before classification/regression:**
  - The reduction is not tuned towards the prediction task
  - Two or more clusters may be covered by a simple predictor
- **Solution:**
  - Cover different input regions with many (simple) networks
  - A kind of predictive clustering with regard to the prediction accuracy
- **Mixture of experts**

  Expert
  - network learner

---
Mixture of experts

- **Gating network**: decides what expert to use

\[ g_1, g_2, \ldots, g_k \] - gating functions