Decision trees

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

---

Decision tree classification

- An alternative approach to classification:
  - Partition the input space to regions
  - Regress or classify independently in every region
Decision tree classification

- An alternative approach to classification:
  - **Partition the input space to regions**
  - **Regress or classify independently in every region**

**Decision tree model:**
- Split recursively the input space \( x \) using simple conditions on \( x_i \)
- Classify at the bottom of the tree

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

---

[Diagram of a decision tree with nodes and branches indicating the decision process for binary classification based on \( x_1, x_2, x_3 \).]
**Decision trees**

**Decision tree model:**
- Split recursively the input space $\mathbf{x}$ using simple conditions on $x_i$
- Classify at the bottom of the tree

**Example:**

*Binary classification \{0,1\}*

*Binary attributes* $x_1, x_2, x_3$

$\mathbf{x} = (x_1, x_2, x_3) = (1,0,0)$

- $x_3 = 0$
- $x_1 = 0$
- $x_2 = 0$

![Decision Tree Diagram](image)

classify $\rightarrow 1\ 0\ 0\ 1\ 1\ 0$
Decision tree model:
• Split recursively the input space \( x \) using simple conditions on \( x_i \)
• Classify at the bottom of the tree

Example:
Binary classification \{0,1\} \( \mathbf{x} = (x_1, x_2, x_3) = (1,0,0) \)
Binary attributes \( x_1, x_2, x_3 \)

Decision trees
**Learning decision trees**

**How to construct /learn the decision tree?**

- **Top-bottom algorithm:**
  - Find the best split condition (quantified based on the impurity measure)
  - Stops when no improvement possible
- **Impurity measure \(I(D)\):**
  - Measures the degree of mixing of the two classes in the subset of the training data \(D\)
  - Worst (maximum impurity) when # of 0s and 1s is the same
- **Splits:** *finite or continuous value attributes*

  **Continuous value attributes conditions:** \(x_3 \leq 0.5\)

---

**Impurity measure**

Let \(|D|\) - Total number of data instances in \(D\)

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

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

**Impurity measure \(I(D)\):**

- Measures the degree of mixing of the two classes in \(D\)
- 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)
    \[ I(D) = \text{Entropy}(D) = -\sum_{i=1}^{k} p_i \log p_i \]
  - **Gini measure** (Breiman, CART)
    \[ I(D) = \text{Gini}(D) = 1 - \sum_{i=1}^{k} p_i^2 \]

Example for k=2

**Gain due to split** – expected reduction in the impurity measure (entropy example)

\[
\text{Gain}(D, A) = \text{Entropy}(D) - \sum_{v \in \text{Values}(A)} \frac{|D^v|}{|D|} \text{Entropy}(D^v)
\]

\(|D^v|\) - a partition of D with the value of attribute A = v
Decision tree learning

• **Greedy learning algorithm:**
  – Builds the tree in the top-down fashion
  – Gradually expands the leaves of the partially built tree

**Algorithm sketch:**
Repeat until no or small improvement in the impurity
  – Find the attribute with the highest gain
  – Add the attribute to the tree and split the set accordingly

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)

---

Decision tree learning

• **Limitations of greedy methods**
  Cases in which only a combination of two or more attributes improves the impurity

```
          1 1 1
          0 0 0
          0 0
          1 1 1
          1
```
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. Build the tree then prune the branches
  – Build the tree, then eliminate leaves that overfit
  – Use validation set to test for the overfit

• Solution 2. Prune while building the tree
  – Test for the overfit in the tree building phase
  – Stop building the tree when performance on the validation set deteriorates

---

Decision tree learning

Backpruning: Prune branches of the tree built in the first phase in the bottom-up fashion by using the validation set to test for the overfit
**Decision tree learning**

**Backpruning:** Prune branches of the tree built in the first phase in the bottom-up fashion by using the validation set to test for the overfit.

\[
\text{Compare: } \#\text{Errors (V)} < \#\text{Error (V')} + \#\text{Errors(V'')}
\]
Decision tree learning

**Backpruning:** Prune branches of the tree built in the first phase in the bottom-up fashion by using the validation set to test for the overfit.

\[
\text{Compare: } \#\text{Errors (V)} < \#\text{Error (V')} + \#\text{Errors(V'')}
\]