Machine Learning
Lecture 7

Lecture by Dr. Shai Fine
Typeset by Steven Karas

2017-12-03
Last edited 20:18:48 2017-12-03

HW2 Anyone who needs a two day extension can get one by emailing Shai at shai.fine@idc.ac.il

Agenda

- Decision Trees

1 Decision Trees

Decision Trees are a flow-chart like tree structure. Internal nodes represent a test to run on the given sample, where the branches represent the outcome of the test. Leaves represent decisions.

Note that in the case of a decision tree, the decision boundaries are aligned to the axes of the feature space.

By building a decision tree, we implicitly perform feature selection as features that don't partition the data well are simply not selected when constructing the tree.

Categorical (Nominal) Attributes Branch for each possible value of the attribute.

Numeric Attributes Numeric here means anything with a total ordering. Tests are typical ordering tests, such as less than, range, etc.

Practical use Works well to build models that are easily presentable to humans (and legal courts). Implictly performs feature selection, and is sometimes used for such.

However, learning the optimal decision tree is NP-Complete.

By their nature, they tend towards overfitting, especially for noisy training data.

1.1 Topdown Induction

This simple algorithm constructs a decision tree from the top down. It is guaranteed to generate a tree, and to stop. However, the depth of the tree is non-optimal (may produce a tree of depth n).

Top down Induction

```
create root node
add all samples to root node
add root node to the queue
until queue.empty?:
  current_node = queue.get
  if current_node.samples all have same class:
    continue

  select the best decision attribute A
  current_node.decision_attribute = A
  for each value of test on A:
    create new node as child of current_node
    assign samples that match to the new node
    queue.put(new node)
```

1.2 Selecting the best decision attribute

In top down induction, we need to select the best decision attribute. Typically, we use a greedy approach that chooses the one that improves classification the most.

We call such a measure impurity: it measures how far we are from dividing all samples into two groups of purely positive and purely negative samples.

Given a random variable x with k discrete values distributed according to $P = (p_1, \ldots, p_k)$. An impurity measure is a function $\phi : [0,1]^k \to \mathbb{R}$ that satisfies the following:

- $\phi(P) \geq 0$
- $\phi(P)$ is minimal if $\exists i, 1 \leq i \leq k$ such that $p_i = 1$
- $\phi(P)$ is maximal if $\forall i, 1 \leq i \leq k$, it holds that $p_i = 1/k$
- $\phi(P)$ is symmetric with respect to the components of P
- $\phi(P)$ is smooth (differentiable everywhere) in its range

Goodness of split Sometimes called the *gain* or *drop* of a measure. The reduction in impurity of the target function after partitioning S according to the values of A:

$$\Delta\phi(S,A) \equiv \underbrace{\phi(S)}_{\text{purity before split}} - \underbrace{\sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|}\phi(S_v)}_{\text{weighted avg of purity after split}}$$

Evaluating the gain Need to evaluate the gain for each potential attribute. Continuously valued attributes need to be tested at all possible partitions for that attribute.

Categories with many values will tend to be selected (index values are especially bad).

Gain Ratio

$$\text{GainRatio}(S,A) = \frac{\text{Gain}(S,A)}{\text{Split}(S,A)}$$

$$\text{Split}(S,A) = -\sum_{a \in A} \frac{|S_a|}{|S|} \log \frac{|S_a|}{|S|}$$

This protects us from splitting into too many groups with little mutual information[1].

1.2.1 Gini Index

Developed by Corrado Gini in 1912. Used in the Classification and Regression Tree (CART) algorithm presented by Breiman, et al in 1984. The Gini index measures dispersion. It is a classification interpretation of the expected error rate. It measures how often a randomly chosen element would be incorrectly classified.

$$\text{GiniIndex}(S) \equiv \sum_{i=1}^{k} p_i(1-p_i) = 1 - \sum_{i=1}^{k} p_i^2 = 1 - \sum_{i=1}^{k}\left(\frac{|S_i|}{|S|}\right)^2$$

The Gini gain is the gain acheived after splitting a node with respect to the values of attribute A:

$$\text{GiniGain}(S,A) = \text{GiniIndex}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|}\text{GiniIndex}(S_v)$$

[1]He only defined split for entropy

Example: Uniform distribution

$$\text{GiniIndex}\left(\frac{1}{k},\dots,\frac{1}{k}\right) = 1 - k\left(\frac{1}{k}\right)^2 = 1 - \frac{1}{c} \cong 1$$

$$\text{GiniIndex}\,(1,\dots,1) = 1 - 1 = 0$$

1.2.2 Entropy

Entropy is a measure of the information or uncertainty. The intuition is that the entropy is a measure of the average information missing about the value of a random variable. Given a system with unknown state, with k possible states, we need $\log k$ bits to represent that state. The entropy is the expected number of bits necessary to encode the values of a random variable.

$$H(P) \equiv \sum_i p_i \log \frac{1}{p_i} = -\sum_i p_i \log p_i$$

Note that the entropy function is convex.
Entropy for labelled classes:

$$\text{Entropy}(S) \equiv -\sum_{i=1}^{k} \frac{|S_i|}{|S|} \log \frac{|S_i|}{|S|}$$

$$\text{InformationGain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|}\text{Entropy}(S_v)$$

Implementations Used in the ID3 and C4.5 algorithms by Quinlan, 1986 and 1993 respectively. C4.5 is an improved version that is widely used. C5 is the commercial successor.

1.3 Preventing Overfitting

There are several options for this, either limiting the depth of the tree, or the size of a leaf node, etc.

A common approach is to prune nodes after growing the full tree that don't provide additional information.

2 Next week