Decision Trees
Friday, November 7, 2014
2:35 PM

DECISION TREES

Decision Trees

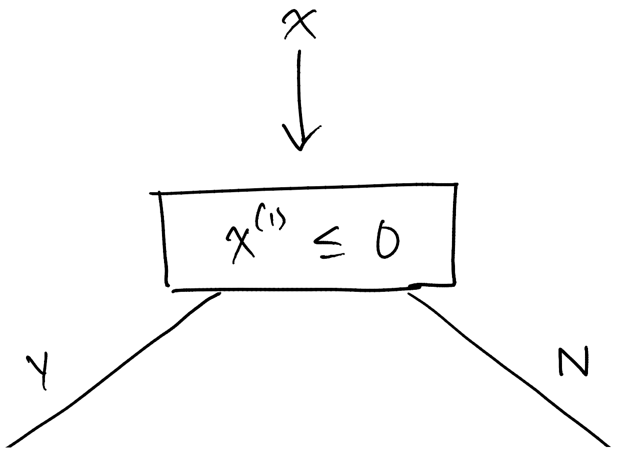
Consider the diabetes data we looked at on a previous homework. There is an 8-dimensional feature vector $x = [x^{(1)} \dots x^{(8)}]^T$ with features

1. Number of times pregnant
2. Plasma glucose concentration
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)

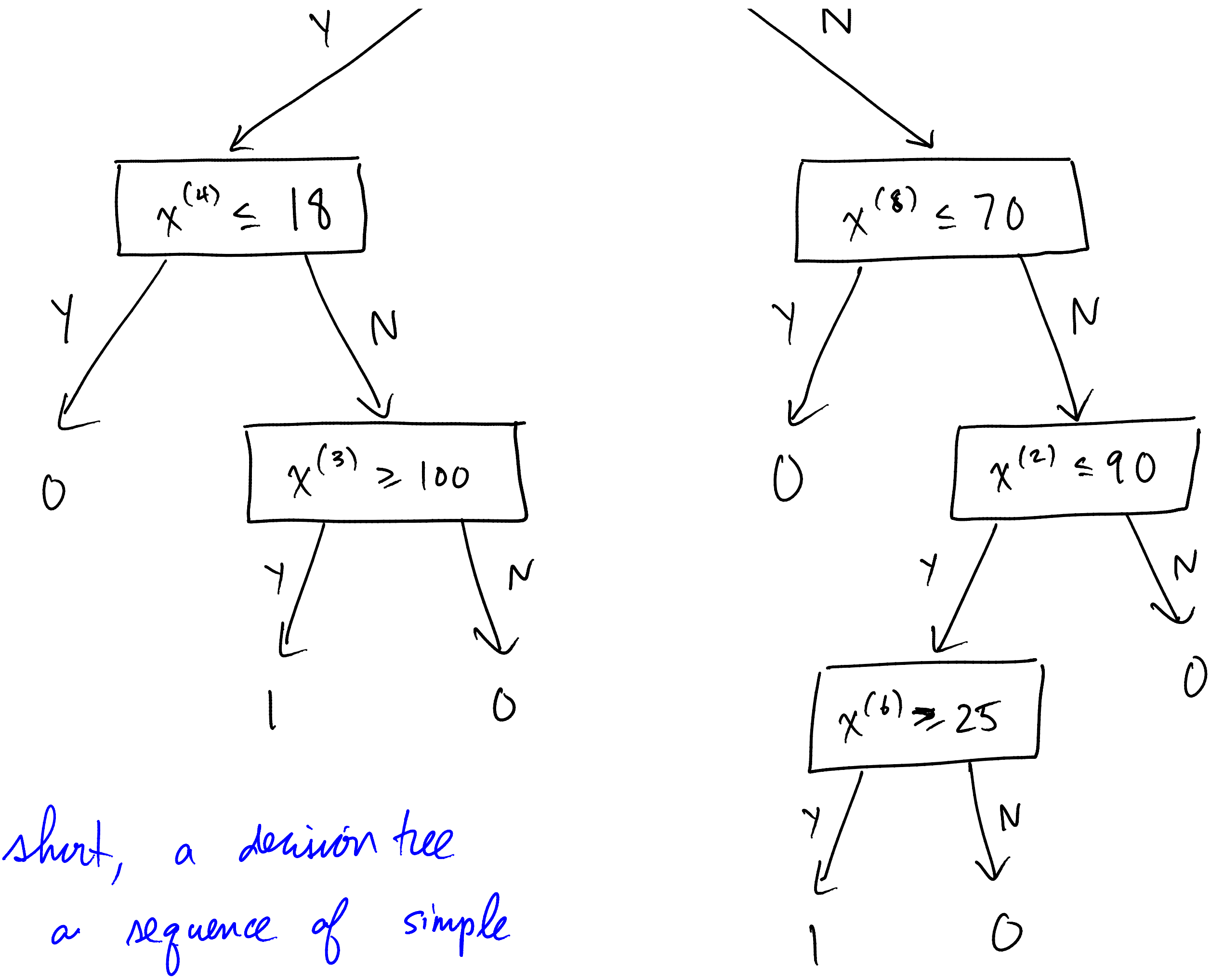
and labels

+1 = tested positive for diabetes
-1 = non-positive

An example of a decision tree is



The values in this chart are made up.



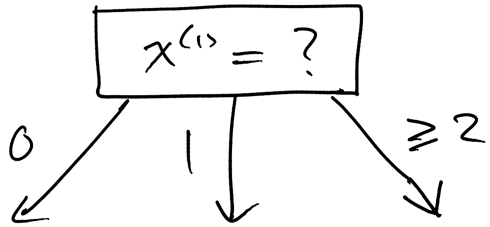
In short, a decision tree asks a sequence of simple questions about x, and predicts the label of x.

Decession trees can also be applied to regression, where the labels at the "leaf nodes" are replaced by real numbers. For example, in the above example we could try to predict Y= systolic blood pressure.

Other generalizations include

- splits that involve more than one feature

- splits with more than two outcomes, e.g.,

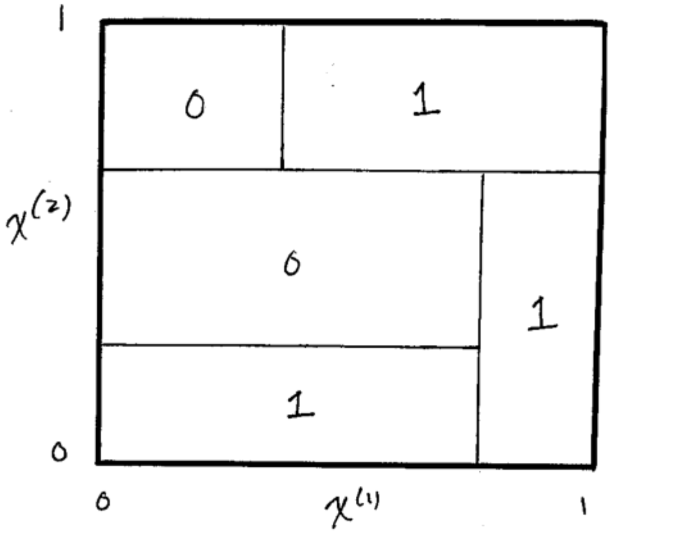


One should exercise care when using these generalizations because of the risk of overfitting.

For simplicity, assume that all splits are binary and are obtained from a single feature.

Terminology

Every decision tree is associated with a partition of the feature space. For example, in $\mathbb{R}^2$

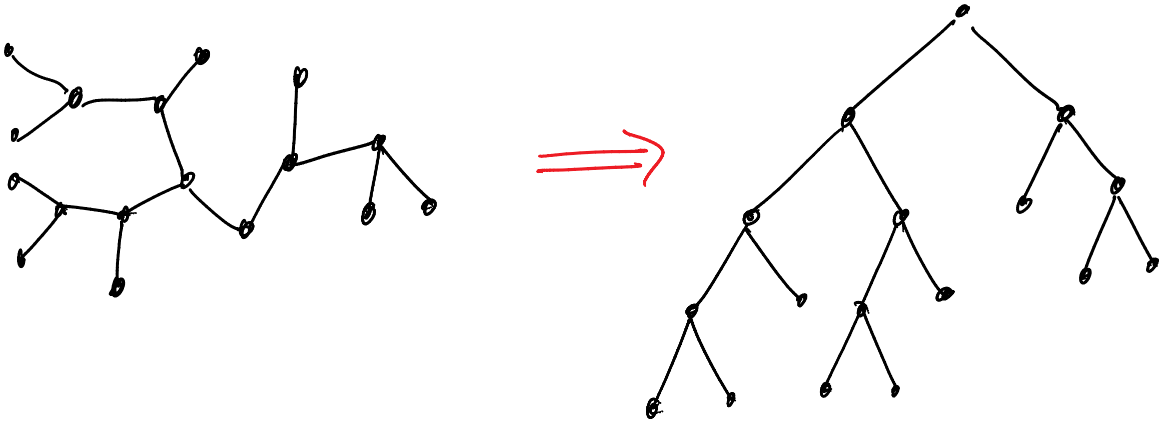


The elements of the partition are referred to as cells.

Recall that a graph is a collection of nodes, some of which are joined by edges. The degree of a node is the number of edges incident on that node.

A tree is a connected graph with no cycles.

A rooted binary tree is a tree where one node, called the root, has degree 2, and all other nodes have degree 1 or 3.



Nodes with degree one are called leaf or terminal nodes.

Nodes with degree ≥ 2 are called internal nodes.

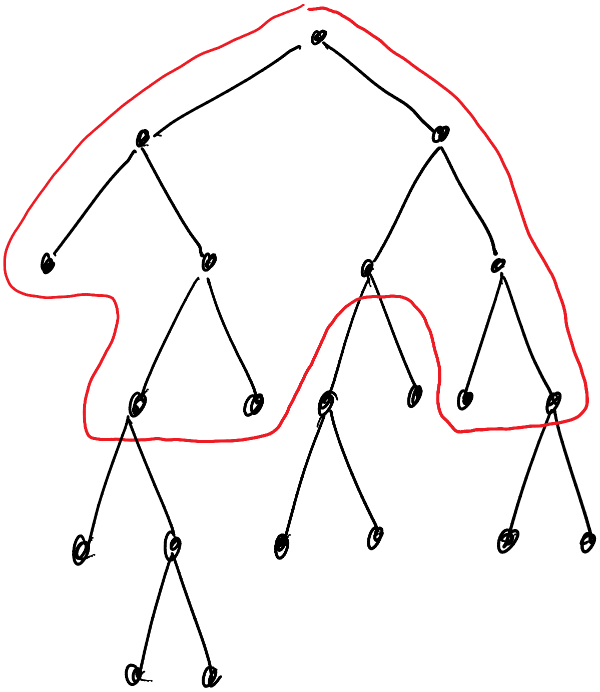
The depth of a node is the length (assuming each edge has unit length) from that node to the root.

The parent of a node is the neighbor of the node whose depth is one less.

Two nodes are siblings if they have the same parent.

A subtree is a subgraph that is also a tree.

A rooted binary subtree is a subtree that contains the root and is such that if any non-root node is in the subtree, so is its sibling.



Finally, a binary decision tree is a rooted binary tree where each internal node is associated with a binary classifier, and each leaf node with a label.

Learning Decision Trees

Let $\mathcal{T}$ denote the set of all binary decision trees. The basic strategy for inferring a decision tree from labelled training data $(x_1,y_1),\dots,(x_n,y_n)$ is penalized empirical risk minimization

$$\min_{T\in\mathcal{T}} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, T(x_i)) + \lambda|T|$$

where ℓ is an appropriate loss, e.g.

- 0/1 loss for classification
- squared error loss for regression

and $|T|$ denotes the number of leaf nodes. Unfortunately this optimization problem is intractable, and therefore a two stage learning procedure is commonly employed:

1. Grow a very large tree T_0 in a greedy fashion.
2. Prune T_0 by solving

$$\min_{T\in\mathcal{T}_0} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, T(x_i)) + \lambda|T|$$

where $\mathcal{T}_0$ is the set of all decision trees based on rooted binary subtrees of T_0.

Growing a Decision Tree

Construction of T_0 follows the following greedy (looking just one step ahead) strategy.

1. Start at root node (= entire feature space)
2. Decide whether to stop growing tree. If yes, assign a label. Stop.
3. If no, consider all possible ways to split the data reaching the current node, and select the best one.
4. For each branch of the split, create a new node and go to 2.

To implement this strategy, we need

(a) A list of possible splits

When considering splits based on a single real-valued feature, splits have the form

$$x^{(j)} \leq t \ ?$$

$j = 1, \ldots, d$. Since there are n data points, only $n-1$ values of t need to be considered for each j. If $x^{(j)}$ is discrete or categorical, other simple splits can be used, like

$$x^{(j)} = \text{"blue"} \ ?$$

(b) Labelling rule

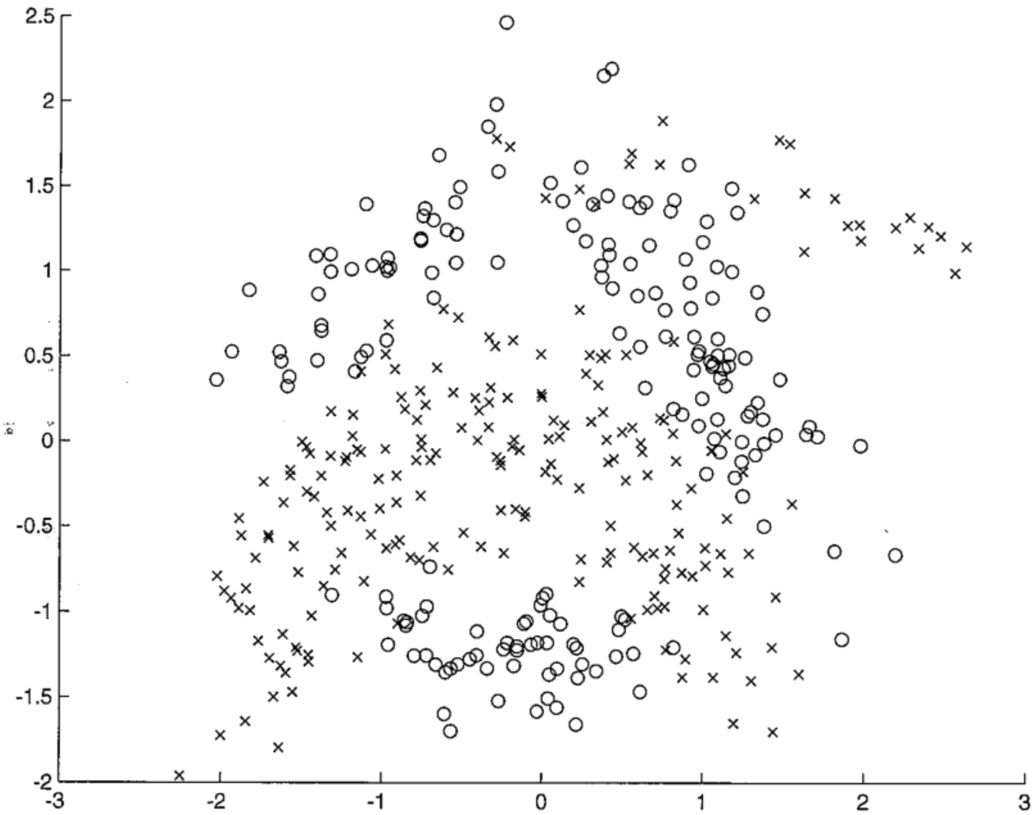
For classification just assign labels by majority vote over data reaching the given node. For regression, take the average y_i over the node.

(c) Rule for stopping splitting

The most common strategy is to just split until each leaf node contains a single data point.

(d) Rule for selecting the best split.

We'll talk about this in detail below.



Split Selection with Impurity Measures

Let's focus on binary classification. Suppose N is a leaf node at some stage in the growing process. We think of N as a cell in a partition of the feature space, and also as

$\{x_i : x_i \in N\}$.

Intuitively, a good split leads to children that are more homogeneous or pure than their parent. To quantify this, we'll define a notion of impurity. Assume the class labels are $\{0,1\}$, denote

$$q := \frac{|\{i : x_i \in N, y_i = 0\}|}{|\{i : x_i \in N\}|}.$$

This ratio define a probality distribution of the labels.

An impurity measure is a function $i(N)$ such that

- $i(N) \geq 0$, with $i(N) = 0$ iff N consists of a single class
- a larger value of $i(N)$ indicates that the distribution defined by q is closer to the uniform distribution.

Examples

- Entropy

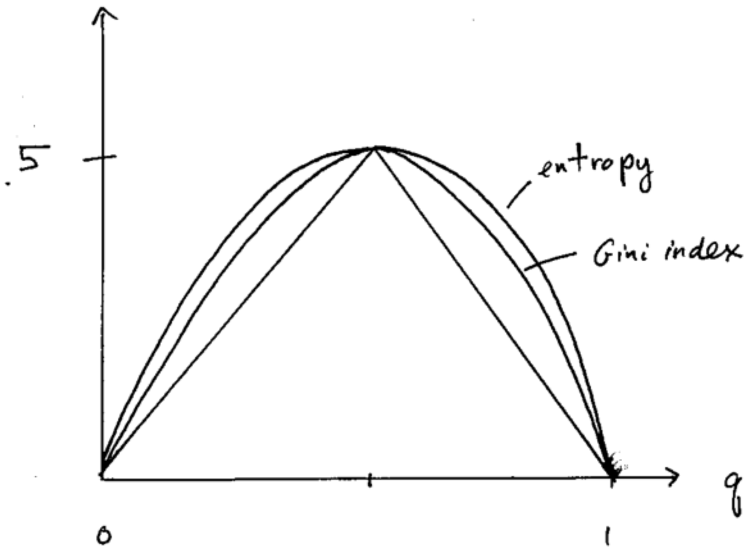
$$i(N) = -[q \log q + (1-q) \log (1-q)]$$

- Gini index

$$i(N) = 2q(1-q)$$

- misclassification rate (of majority vote classifier)

$$i(N) = \min(q, 1-q)$$



To select the best split, we maximize the decrease in impurity.

Thus, let N_1 and N_2 be two children of N. Define

$$p(N_1) = \frac{|N_1|}{|N|}, \quad p(N_2) = \frac{|N_2|}{|N|}$$

The decrease in (expected) impurity is

$$i(N) - \left[p(N_1)\, i(N_1) + p(N_2)\, i(N_2) \right].$$

When i is entropy, this is called the information gain.

By Jensen's inequality, we have the following result.

Proposition If i is concave as a function of q, then

$$i(N) - [p(N_1)\, i(N_1) + p(N_2)\, i(N_2)] \geq 0$$

$\forall N_1, N_2$. If i is strictly concave and $N_1 \neq \phi \neq N_2$, then equality holds iff $q = q_1 = q_2$, where q_i is the proportion of class 0 in N_i.

Proof Let us write $i(N) = \phi(q)$ where ϕ is concave. Then

$$i(N) = \phi(q) = \phi(p(N_1) q_1 + p(N_2) q_2)$$

$$\geq p(N_1)\, \phi(q_1) + p(N_2)\, \phi(q_2) \qquad \text{(Jensen's inequality)}$$

$$= p(N_1)\, i(N_1) + p(N_2)\, i(N_2).$$

Since N_1 and N_2 are assumed nonempty, $p(N_1) > 0$ and $p(N_2) > 0$, and therefore equality holds iff $q_1 = q_2$, in which case $q = q_1 = q_2$. □

This result explains why strictly concave impurity measures are preferred.

This result generalizes easily to multiclass classification.

Pruning

To solve

$$\min_{T \in \mathcal{T}_0} \quad J(T) := \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, T(x_i)) + \lambda |T|$$

we rely on the fact that the objective function is additive in the following sense. For any tree T, let $\Pi(T) = \{A_1, A_2\}$ be the partition of the feature space corresponding to the children of the root node. Then

$$J(T) = \sum_{A \in \Pi(T)} J(T_A)$$

where T_A is the subtree rooted at A. This immediately suggests a recursive algorithm. Alternatively, there is an efficient "bottom-up" dynamic programming algorithm to solve the pruning problem.

Question: Why should we grow and then prune as opposed to growing and just stopping when the decrease in impurity is negligible?

Answer: Because of ancillary splits: These are splits that have no value by themselves, but that enable useful splits later on.

Final Thoughts

Advantages of decision trees:

- interpretable
- rapid evaluation
- easily handles categorical data, multiple classes

Disadvantages of decision trees

- Greedy growing is suboptimal
- Unstable: a slight perterbation of training data could drastically alter learned tree
- Jagged decision boundaries

The latter two issues can be addressed by incorporating decision trees into an ensemble method.

A good reference for decision trees is

Ripley, Pattern Recognition and Neural Networks,
Cambridge University Press, 1996.