Decision Trees And Training Strategies

Arnav Gupta

October 21, 2024

Contents

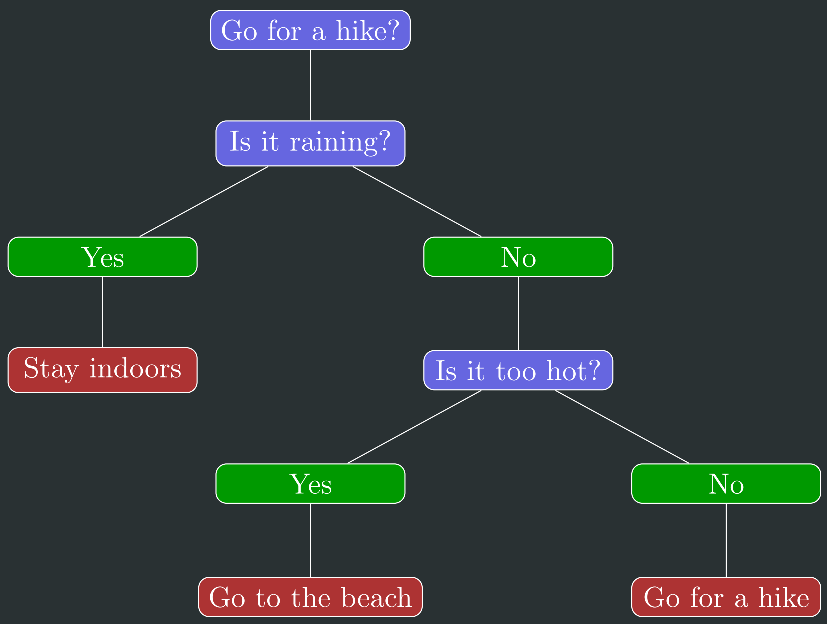
1 Decision Trees

Technique for supervised learning from discrete data:

- representation is a decision tree
- bias towards simple decision tree
- search through space of decision trees, from simple to complex

For each **decision tree**:

- **nodes**: input attributes/features
- **branches**: labeled with input feature values (can have multiple feature values on one branch)
- **leaves**: predictions for target features (point estimates)



To learn a decision tree:

1. split training data based some criteria (bias)
2. recursively solve sub-problems

Criteria for good decision trees: small, good classification (low error on training data), good generalization (low error on test data).

Algorithm 1 Decision Tree Learner

procedure DecisionTreeLearner(X, Y, E)
    if stopping criteria is met **then**
        return pointEstimate(Y, E)
    else
        select input feature $X_i \in X$
        for each value x_i of X_i **do**
            $E_i \leftarrow$ all examples in E where $X_i = x_i$
            $T_i \leftarrow \text{DecisionTreeLearner}(X \setminus \{X_i\}, Y, E_i)$
        end for
        return $\langle X_i, T_1, \ldots, T_N \rangle$
    end if
end procedure

Algorithm 2 Classify Example

procedure ClassifyExample(e, X, Y, DT)
    $S \leftarrow DT$
    while S is an internal node of the form $\langle X_i, T_1, \ldots, T_N \rangle$ **do**
        $j \leftarrow X_i(e)$
        $S \leftarrow T_j$
    end while
    return S
end procedure

1.1 Stopping Criteria

Related to the final return value, depends on what must be done.

Possible stopping criteria:

- no more features
- performance on training data good enough

1.2 Feature Selection

Choose sequence of features that result in smallest tree.

In practice, split based on what gives best performance.

Heuristics for best performing feature:

- most even split
- max info gain
- GINI index

2 Information Theory

n bits can distinguish 2^n items, but can do better with probabilities.

In general, need $-\log_2 P(x)$ bits to encode x. Each symbol requires on average

$$-P(x)\log_2 P(x) \text{ bits}$$

To transmit an entire sequence distributed according to $P(x)$,

$$\sum_x -P(x)\log_2 P(x) \text{ bits}$$

bits of info per symbol are needed on average, which is the **info content** or **entropy** of the sequence.

2.1 Information Gain

Given a set E of N training examples, if the number of examples with output feature $Y = y_i$ is N_i, then

$$P(Y = y_i) = P(y_i) = \frac{N_i}{N}$$

is the point estimate.

The total info content for E is

$$I(E) = -\sum_{y_i \in Y} P(y_i)\log(P(y_i))$$

After splitting E into E_1 and E_2 based on input feature X_i, the information content is

$$I(E_{split}) = \frac{N_1}{N} I(E_1) + \frac{N_2}{N} I(E_2)$$

so the desirable X_i is the one that maximizes **info gain**:

$$I(E) - I(E_{split}) = -\sum_{y \in Y} P(y)\log P(y) + \sum_{x \in X, y \in Y} P(x,y)\log\frac{P(x,y)}{P(x)}$$

which gives the inequality

$$IG = -\sum_{x \in X, y \in Y} P(x,y)\log\frac{P(x)P(y)}{P(x,y)} \geq -\log\left(\sum_{x \in X, y \in Y} P(x,y)\frac{P(x)P(y)}{P(x,y)}\right)$$

Info gain is the reduction in uncertainty about the output feature T given the value of a certain input feature X.

Jensen's inequality: for a convex function $f(x)$, $E[f(x)] \geq f(E[x])$.

3 Training Strategies

3.1 Final Return Value

Point estimate (prediction of target features) of Y over all examples.

A point estimate could be mean, median, mode, or

$$P(Y = y_i) = \frac{N_i}{N}$$

3.2 Priority Queue

Sort leaves using a priority queue ranked by how much info can be gained with the best feature at that leaf. Always expand the leaf at the top of the queue.

Algorithm 3 Decision Tree Learner

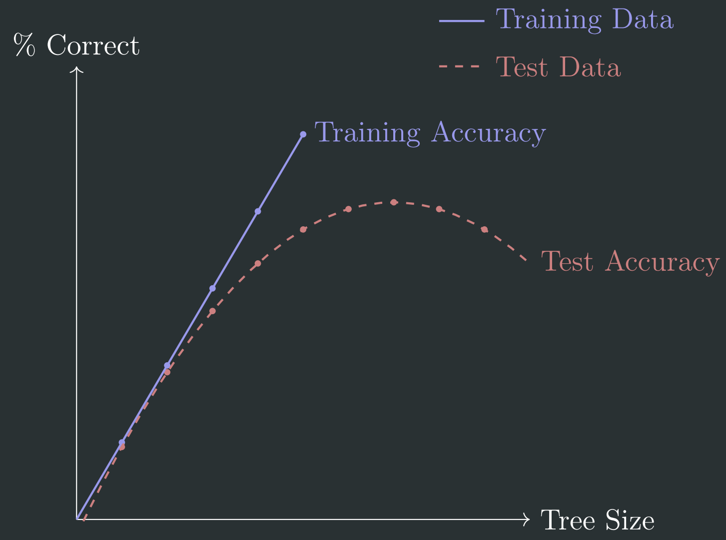
procedure DecisionTreeLearner(X, Y, E)
    $DT \leftarrow$ pointEstimate(Y, E) ▷ initial decision tree
    $\{X', \Delta I\} \leftarrow$ best feature and Information Gain value for E
    $PQ \leftarrow \{(DT, E, X', \Delta I)\}$ ▷ priority queue of leaves ranked by ΔI
    while stopping criteria is not met **do**
        $\{S_\ell, E_\ell, X_\ell, \Delta I_\ell\} \leftarrow$ leaf at the head of PQ
        for each value x_i of X_ℓ **do**
            $E_i \leftarrow$ all examples in E_ℓ where $X_\ell = x_i$
            $\{X_j, \Delta I_j\} \leftarrow$ best feature and value for E_i
            $T_i \leftarrow$ pointEstimate(Y, E_i)
            insert $\{T_i, E_i, X_j, \Delta I_j\}$ into PQ according to ΔI_j
        end for
        $S_\ell \leftarrow \langle X_\ell, T_1, \ldots, T_N \rangle$
    end while
    return DT
end procedure

3.3 Overfitting

When there is not enough data, the decision tree does not generalize to test data.

Methods to avoid overfitting:

- **regularization**: prefer small decision trees, so add a complexity penalty to stopping criteria
- **pseudocounts**: add data based on prior knowledge
- **cross validation**



Test set errors are caused by:

- **bias**: error due to the algorithm finding an imperfect model
  - **representation bias**: model too simple
  - **search bias**: not enough search
- **variance**: error due to lack of data
- **noise**: error due to data depending on features not modeled or because process generating data is stochastic
- **bias-variance trade-off**
  - complicated model, not enough data
  - simple model, lots of data

Capacity: ability of a model to fit a wide variety of functions (inverse of bias)

Cross Validation:

- split training data into training and validation set
- use validation set as fake test set
- optimize decision maker to perform well on validation set
- repeat with different validation sets