menu

› Course Shortcuts

› Student Lounge

› Q&A

⌄ Create Decision Trees

Module Introduction: Create Decision Trees

Notations for Machine Learning

Build a Decision Tree

Explore Decision Trees

Measure a Partition's Purity

Formalize Impurity Functions

Determine the Minimum Entropy Split

CART Algorithm

CART Algorithm

CART Cheat Sheet

Minimize Impurity of Regression Trees

Squared-loss Impurity

Linear Time Regression Trees

Implement Regression Trees

Module Wrap-up: Create Decision Trees

› Choose the Right Model

› Data Science in the Wild

› Course Resources

Formalize Impurity Functions

As discussed in the video, you'll use the two impurity functions — the Gini impurity and the entropy — to quantify the heterogeneity of a subsection of data. Intuitively, the impurity of a set of points is higher when the points have many different labels and lower when most points have the same label. Below, let's define the two functions more formally:

☆ **Key Points**

Gini impurity and entropy are two formal ways of capturing the label "disagreement" of a set.

Impurity metrics are higher for groups with mixed labels.

Impurity metrics are lowest when every point in a set has the same label.

Impurity Functions

Data: $S=\{(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_n,y_n)\}, y_i\in\{1,\ldots,c\}$, where c is the number of classes.

Gini impurity

Let S_k be subsets of S such that points in S_k have label k. That is, $S_k\subseteq S$ where $S_k=\{(\mathbf{x},y)\in S: y=k\}$ (all inputs with label k). Clearly, we have the property that $S=S_1\cup\cdots\cup S_c$.

Now, the probability that a point picked uniformly at random from S has label k is equal to the fraction of points with label k relative to the size of S:

$$p_k=\frac{|S_k|}{|S|}\leftarrow \text{fraction of points in } S \text{ with label } k$$

Recall what the CART algorithm does to find splits in the data: it divides the points S into "left" and "right" branches S_L and S_R if possible; if not, it lets S be in a "leaf". For dividing, the algorithm finds the feature and the feature value that minimizes the impurity function, thus optimally partitioning S **at that step**. Below you will see how the impurity function is defined in the two cases: 1) when S ends up in a leaf, and 2) when S is divided into two branches.

Gini impurity of a leaf

Each split in the decision tree partitions the data into two distinct sets and each training point falls into one of the leaves. The Gini impurity for a leaf with corresponding set S is the probability of event E: that two points, both picked uniformly at random from set S, have *different* labels.

This event E is a union of all E_k: that two points, both picked uniformly at random from set S, have labels k and something else respectively. Therefore, $E=E_1\cup\cdots\cup E_c$. Note that any two different events E_k and $E_{k'}$ don't occur simultaneously as it is not possible for the first point to have both label k and k' simultaneously; consequently $P\left(E_k\cap E_{k'}\right)=0$. Therefore, the probability of event E is:

$$G(S)=P(E)=P\left(E_1\cup\cdots\cup E_c\right)=\textstyle\sum_{k=1}^{c}P\left(E_k\right)$$

Intuitively, we sum over all possible labels and compute the probability that the first point has a particular label (which has probability p_k) and the second point does not have that label $(1-p_k)$.

$$G(S)=\textstyle\sum_{k=1}^{c}P\left(E_k\right)=\sum_{k=1}^{c}p_k(1-p_k)$$

Gini impurity of a tree

The Gini impurity of a tree is defined recursively from the root of the tree. The Gini impurity at a leaf is computed just as stated above. The Gini impurity for all the intermediate nodes is the weighted sum of the Gini impurity of the children (weighted by the fraction of training points that fall into the left or right subtree).

$$G(S)=\tfrac{|S_L|}{|S|}G\left(S_L\right)+\tfrac{|S_R|}{|S|}G\left(S_R\right)$$

where:

- $S_L\leftarrow$ Data points falling into the left subtree
- $S_R\leftarrow$ Data points falling into the right subtree
- $S=S_L\cup S_R$
- $S_L\cap S_R=\emptyset\leftarrow S_L$ and S_R are disjoint sets of points (a data point cannot be in both simultaneously)
- $\frac{|S_L|}{|S|}\leftarrow$ Fraction of inputs in left subtree
- $\frac{|S_R|}{|S|}\leftarrow$ Fraction of inputs in the right subtree

Entropy

Entropy over a leaf

The entropy over a leaf captures the "amount of label information in the leaf". Intuitively, entropy of a set of points is the "surprise" expected of labels when points are drawn uniformly randomly. If all points in the set have the same label, then the surprise is minimum (minimum impurity) since only one label will turn up. If there are many labels in the set, then the surprise will be high (high impurity). If each point has a distinct label, then the surprise is maximum (maximum impurity) since your label prediction would not be better than predicting uniformly randomly.

Formally, the surprise is estimated by the number of bits required to transmit the information about which label turned up. We use $-\log(p_k)$ to estimate the number of bits. Here's why it works:

- If all points in the set have the same label, $p_k=1$ for one label and $p_{k'}=0$ for all other labels $k'\neq k$. Hence, you would not need to transmit anything, because the outcome is already known beforehand.
- If c points have distinct labels, thus $p_k=\frac{1}{c}$ for all k, Hence, you would need $\log(c)$ bits to transmit which label turned up, because the probability of any of the c labels turning up is equal to $\frac{1}{c}$.

Each label has a probability p_k of turning up, and so the entropy for a label k is $-p_k\log(p_k)$. Consequently, the entropy for all labels in the leaf is the total surprise:

$$H(S)=-\textstyle\sum_{k=1}^{c}p_k\log(p_k)$$

Entropy over a tree

The entropy over a tree is defined recursively (similar to the Gini impurity over a tree).

$$H(S)=\frac{|S_L|}{|S|}H\left(S_L\right)+\frac{|S_R|}{|S|}H\left(S_R\right)$$

‹ Previous

Next ›