Randomized Decision Trees

compiled by Alvin Wan from Professor Jitendra Malik's lecture

1 Discrete Variables

First, let us consider some terminology. We have primarily been dealing with **real-valued** data, where our features are continuous variables. For example, we would consider $X \in \mathbb{R}^n$. However, some variables are naturally discrete, such as zip codes, political parties, or letter grades. We can sub-divide discrete variables into two categories:

- **Nominal variables** have at least two categories but have no intrinsic order e.g., hair color, zip code, gender, kind of housing
- **Ordinal variables** have a natural order e.g., education, letter grades. These can often be converted into real numbers. e.g., years of education, percentages

2 Value of Information

2.1 Surprise

In essence, knowing a certain event has occurred gives us no new information. Knowing that a rare event has occurred is more informative. In other words, the more likely it is, the less the surprise.

$$-\log_b(p)$$

Note that when $p = 1$, surprise is 0, and as the probability of our event decreases or as $p \to 0$, we see greater and greater surprise, $-\log_b(p) \to \infty$.

Example Consider a random variable X, an indicator for a coin with bias p. Let us assume that we know p is fairly large, so that $p \approx 1$. Then, we intuitively receive more information when we see the coin results in tails.

2.2 Entropy

Entropy is the expected value of the surprise. The **expected value of information** is maximized when all the n values are equally likely.

$$\textsc{Entropy} = -\sum_{i=1}^{n} p_i \log_b(p_i)$$

Consider a case where $p_1 = 1, p_2 = 0$. Then, we have that

$$\begin{aligned}\textsc{Entropy} &= -(p_1 \log_2 p_1 + p_2 \log_2 p_2) \\ &= -(1 \log_2 1 + 0 \log_2 0) \\ &= 0\end{aligned}$$

Now, consider $p_1 = p_2 = \frac{1}{2}$. Then, we have that

$$\begin{aligned}\textsc{Entropy} &= -(p_1 \log_2 p_1 + p_2 \log_2 p_2) \\ &= -(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}) \\ &= -(-\frac{1}{2} - \frac{1}{2}) \\ &= 1 \textbf{ bit}\end{aligned}$$

For base 2, entropy is in units of *bits*. For base e, entropy is in units of *nats*.

Example Consider an n-sided coin. We receive most information when $P(X_i) = \frac{1}{n}$.

3 Classification

Example Consider the game, "20 questions", where one player thinks of a country and the other has the opportunity to ask 20 yes-no questions. The tree of possibilities is a binary tree but more generally, the decision tree associated with this example.

Definition Formally, a **decision tree** models a series of *decisions* and the *consequences* of those decisions.

3.1 Classification

At **training time**, we construct the tree by picking the "questions" at each node of the tree. This is done to decrease entropy with each level of the tree. A single leaf node is then associated with a set of training examples.

At **test time**, we evaluate the "questions" from the root node. Once a leaf node is reached, we predict the class to be the one with the most examples - from the training set - at this node.

Note that training time is slow but that testing is fast. Additionally, this model can accommodate real-valued features. We can achieve this by using, for example, inequalities to binarize decisions. In this sense, we see that a question could be similar to a perceptron, where our decision is $w_i x_i > \textsc{Threshold}$.

3.2 Example Training

Take some data $x \in \mathbb{R}^d, y \in \mathbb{R}$. We will take the following decision tree:

- At the root, test if $f_1(x) > \textsc{Threshold}$.
- If no, test if $f_{11}(x) > \textsc{Threshold}$.
- If yes, test if $f_{12}(x) > \textsc{Threshold}$.

  $\vdots$

Eventually, x will land at a leaf. Ideally, this leaves should be **pure**, meaning that all outcomes at this leaf node belong to a single class. In a sense, the decision tree divides our sample space into boxes, using axis-parallel splits.

At test time, we simply classify a new example x' and for whatever leaf node it lands at, we will take a "vote" across all of the training samples that ended at the same leaf node. If this is a classification problem, we take a majority vote across all k classes. If this is a regression problem, simply predict the average of all training samples at that leaf node.

3.3 Random Forests

From the empirical distribution at a leaf, we can infer the posterior probability. A **Random Forest** is a family of decision trees, across which we average posterior probabilities. Simple-minded averaging works quite well. With *boosting*, we re-weight particular decision trees. This comes at a risk of increasing weights for mis-labeled data.

Random forests benefit from the "wisdom of the crowds", which is the idea that the average guess across many participants is more accurate than a single expert's guess.

4 Design

At the leaves, we want low entropy. Assume that we there exist 256 countries. At the root of our associated decision tree, our entropy is 8.

$$\sum_{i=1}^{256} -\frac{1}{256}\log_2\frac{1}{256} = \sum_{i=1}^{256}\frac{1}{256}\log_2 256 = \sum_{i=1}^{256}\frac{8}{256} = 8$$

Note that entropies are computed from empirical frequencies and not probabilities. It is not the true underlying probability but the frequency that we observe.

4.1 A/B Testing

We have a total of $n_1 + n_2$ samples, and at the root node with entropy H, we ask question A. We see n_2 positive classifications that proceed to a node with entropy H_+ and n_1 negative classifications that proceed to a node with entropy H_-. Our entropy at the second level is thus

$$\frac{n_1 H_- + n_2 H_+}{n_1 + n_2}$$

To measure the cost or benefit, we evaluate our **information gain**. This is the decrease in entropy, or

$$H - \frac{n_1 H_- + n_2 H_+}{n_1 + n_2}$$

To conduct A/B testing, we simply use the question that results in the highest information gain. This is not guaranteed to find the optimal decision tree but it gives a reasonable approximation.

4.2 Examples

The simplest version of a question tests a single feature against a threshold.