Decision Trees, Bagging, and Random Forests

CS145: Introduction to Data Mining

Spring 2024

Contents

1 Introduction

Decision trees are a popular non-linear, non-parametric supervised learning method used for both classification and regression tasks. They are simple to understand and interpret, can handle both categorical and numerical data, and require little data preparation. Bagging and random forests are ensemble learning methods that combine multiple decision trees to create more powerful models. In this lecture, we will cover the basics of information theory, decision tree construction, bagging, and random forests.

2 Information Theory Basics

Before diving into decision trees, let's review some fundamental concepts from information theory that are used in constructing decision trees.

2.1 Jensen's Inequality

Jensen's inequality states that for a convex function f and random variable X:

$$\mathbb{E}[f(X)] \geq f(\mathbb{E}[X]) \tag{1}$$

This inequality is used in deriving the information gain criterion for splitting nodes in decision trees.

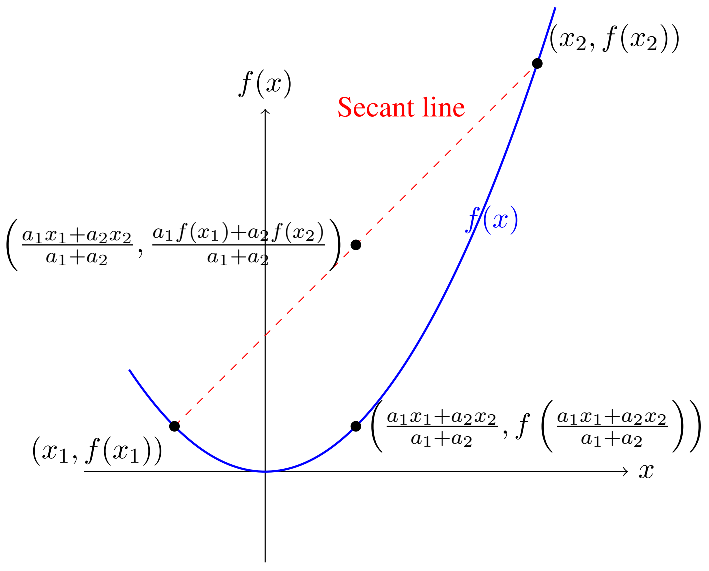
$(x_2, f(x_2))$
$f(x)$
Secant line
$\left(\frac{a_1x_1+a_2x_2}{a_1+a_2}, \frac{a_1f(x_1)+a_2f(x_2)}{a_1+a_2}\right)$
$f(x)$
$\left(\frac{a_1x_1+a_2x_2}{a_1+a_2}, f\left(\frac{a_1x_1+a_2x_2}{a_1+a_2}\right)\right)$
$(x_1, f(x_1))$
x

Figure 1: Illustration of the convex combination of two points on a convex function.

2.2 Entropy

Entropy is a measure of the uncertainty or randomness in a random variable. For a discrete random variable X with probability mass function $p(x)$, the entropy is defined as:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x) \tag{2}$$

where $\mathcal{X}$ is the set of possible values for X. Entropy is measured in bits and is always non-negative. It is maximized when all outcomes are equally likely and minimized when one outcome has a probability of 1.

Example 1: Consider a binary random variable X with $P(X = 0) = p$ and $P(X = 1) = 1 - p$. The entropy of X is:

$$H(X) = -p \log_2 p - (1 - p) \log_2 (1 - p) \tag{3}$$

Example 2: Consider a fair six-sided die. The entropy of the die roll is:

$$H(X) = -\sum_{i=1}^{6} \frac{1}{6} \log_2 \frac{1}{6} = \log_2 6 \approx 2.58 \text{ bits} \tag{4}$$

2.3 Conditional Entropy

Conditional entropy measures the uncertainty in a random variable Y given the value of another random variable X. It is defined as:

$$H(Y|X) = \sum_{x \in \mathcal{X}} p(x) H(Y|X = x) \tag{5}$$

where $H(Y|X = x)$ is the entropy of Y conditioned on $X = x$.

Example: Consider two random variables X and Y, where X represents the outcome of a fair coin toss (0 for tails, 1 for heads) and Y represents the outcome of a die roll (1-6). The conditional entropy of Y given X is:

$$\begin{aligned} H(Y|X) &= P(X = 0)H(Y|X = 0) + P(X = 1)H(Y|X = 1) \\ &= 0.5 \cdot \log_2 6 + 0.5 \cdot \log_2 6 \\ &= \log_2 6 \approx 2.58 \text{ bits} \end{aligned}$$

2.4 Cross Entropy

Cross entropy measures the difference between two probability distributions p and q over the same set of events. It is defined as:

$$H(p, q) = -\sum_{x \in \mathcal{X}} p(x) \log_2 q(x) \tag{6}$$

Cross entropy is minimized when $p = q$.

Example: Consider two probability distributions p and q over a binary random variable:

$$\begin{aligned} p &= (0.7, 0.3) \\ q &= (0.6, 0.4) \end{aligned}$$

The cross entropy between p and q is:

$$H(p, q) = -0.7 \log_2 0.6 - 0.3 \log_2 0.4 \approx 0.88 \text{ bits}$$

2.5 KL-Divergence

The Kullback-Leibler (KL) divergence measures the difference between two probability distributions p and q. It is defined as:

$$D_{\text{KL}}(p \parallel q) = \sum_{x \in \mathcal{X}} p(x) \log_2 \frac{p(x)}{q(x)} \tag{7}$$

KL-divergence is always non-negative and is zero if and only if $p = q$.

Example: Using the same probability distributions p and q from the cross entropy example, the KL-divergence between p and q is:

$$D_{\text{KL}}(p \parallel q) = 0.7 \log_2 \frac{0.7}{0.6} + 0.3 \log_2 \frac{0.3}{0.4} \approx 0.08 \text{ bits}$$

2.6 Mutual Information

Mutual information measures the amount of information obtained about one random variable by observing another random variable. It is defined as:

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{8}$$

Mutual information is symmetric, non-negative, and zero if and only if X and Y are independent.

Example: Consider two binary random variables X and Y with the following joint probability distribution:

$P(X,Y)$	$Y=0$	$Y=1$
$X=0$	0.4	0.1
$X=1$	0.1	0.4

The mutual information between X and Y is:

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= -0.5 \log_2 0.5 - 0.5 \log_2 0.5 - (0.5 \cdot 0.8 \log_2 0.8 + 0.5 \cdot 0.8 \log_2 0.8) \\ &\approx 0.19 \text{ bits} \end{aligned}$$

3 Decision Trees

3.1 Non-Linear Functions

Decision trees are non-linear models that can learn complex decision boundaries by splitting the feature space into rectangles (for 2D features) or hyperrectangles (for higher-dimensional features). Unlike linear models, decision trees can capture interactions between features and handle non-linear relationships between the features and the target variable.

3.2 Basic Construction of Decision Trees

A decision tree is a hierarchical structure consisting of internal nodes, branches, and leaf nodes. Each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label or a regression value. Decision trees are constructed using a top-down, greedy search approach called recursive binary splitting.

3.3 Decision Functions

At each internal node, a decision function is used to determine the splitting criterion. For classification trees, the decision function aims to maximize the purity of the resulting subsets, while for regression trees, the decision function aims to minimize the mean squared error (MSE) or mean absolute error (MAE) of the resulting subsets.

3.4 Impurity Measures

Impurity measures are used to evaluate the quality of a split in classification trees. The goal is to minimize the impurity of the resulting subsets. Common impurity measures include:

3.4.1 0/1 Loss

The 0/1 loss measures the misclassification rate of a subset. It is defined as:

$$L_{0/1}(S) = \frac{1}{|S|} \sum_{i \in S} \mathbb{I}(y_i \neq \hat{y}) \tag{9}$$

where S is the subset, y_i is the true class label, $\hat{y}$ is the predicted class label, and $\mathbb{I}$ is the indicator function.

3.4.2 Entropy

Entropy, as defined in the previous section, can be used as an impurity measure. The entropy of a subset S with C classes is:

$$H(S) = -\sum_{c=1}^{C} p_c \log_2 p_c \tag{10}$$

where p_c is the proportion of examples in S that belong to class c.

3.4.3 Gini Index

The Gini index measures the probability of misclassification if a random example from the subset is labeled according to the class distribution in the subset. It is defined as:

$$G(S) = 1 - \sum_{c=1}^{C} p_c^2 \tag{11}$$

The Gini index is maximized when the classes are equally distributed in the subset and minimized when one class dominates the subset.

Example: Consider a subset S with 10 examples, where 7 belong to class A and 3 belong to class B. The entropy and Gini index of S are:

$$H(S) = -\frac{7}{10}\log_2\frac{7}{10} - \frac{3}{10}\log_2\frac{3}{10} \approx 0.88 \text{ bits}$$

$$G(S) = 1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2 = 0.42$$

Algorithm 1 ID3 Algorithm

```
1: function ID3($S, A, y$)
2:     if all examples in $S$ have the same class then
3:         return a leaf node with the class label
4:     end if
5:     $a^* \leftarrow \text{argmax}_{a \in A} \text{IG}(S, a)$
6:     Create a root node with feature $a^*$
7:     for each value $v$ of $a^*$ do
8:         Create a branch for condition $a^* = v$
9:         $S_v \leftarrow \{(x, y) \in S | x_{a^*} = v\}$
10:        if $S_v$ is empty then
11:            Create a leaf node with the most common class in $S$
12:        else
13:            Recursively call ID3($S_v, A \setminus \{a^*\}, y$)
14:        end if
15:    end for
16: end function
```

3.5 ID3 Algorithm

The ID3 (Iterative Dichotomiser 3) algorithm is a simple decision tree learning algorithm that uses entropy as the impurity measure. The algorithm works as follows:

The information gain (IG) is used to select the best attribute for splitting. It is defined as the reduction in entropy after splitting on an attribute a:

$$\text{IG}(S, a) = H(S) - \sum_{v \in \text{Values}(a)} \frac{|S_v|}{|S|} H(S_v) \tag{12}$$

where S_v is the subset of examples in S with attribute value $a = v$.

3.6 Top-Down Training

Decision trees are trained using a top-down, greedy search approach called recursive binary splitting. The algorithm starts at the root node and recursively splits the data into subsets based on the selected attributes. The process continues until a stopping criterion is met, such as a maximum depth or a minimum number of examples in a leaf node.

3.7 Stopping Conditions

To prevent overfitting, several stopping conditions can be used to control the growth of the decision tree:

- Maximum depth: Stop splitting when the maximum depth is reached.
- Minimum number of examples in a leaf: Stop splitting when the number of examples in a leaf node is below a threshold.
- Minimum impurity reduction: Stop splitting when the impurity reduction is below a threshold.
- Statistical significance: Stop splitting when the split is not statistically significant based on a chi-square test or other statistical tests.

3.8 Example Decision Tree

Figure 2 shows an example decision tree for a binary classification problem with two features: x_1 and x_2. The tree has a depth of 2 and splits the feature space into four regions, each corresponding to a leaf node with a class label (0 or 1).

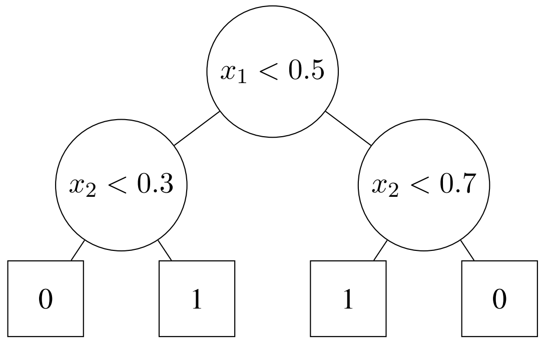

Figure 2: Example decision tree for a binary classification problem.

3.9 Limitations of Decision Trees

While decision trees are intuitive and interpretable, they have some limitations:

- Decision trees can easily overfit the training data, especially when grown deep or with small minimum leaf sizes.
- Decision trees are sensitive to small changes in the training data, which can lead to instability and high variance.
- Decision trees are biased towards features with many levels or continuous features, as they can be split into more subsets.
- Decision trees cannot learn certain simple concepts, such as XOR, without creating complex trees.

4 Bagging

Bagging (Bootstrap Aggregating) is an ensemble learning method that combines multiple base learners to improve the stability and accuracy of the model. The basic idea is to train multiple base learners on bootstrap samples of the training data and then aggregate their predictions by voting (for classification) or averaging (for regression).

4.1 Bias-Variance Decomposition

The bias-variance decomposition is a way to analyze the prediction error of a learning algorithm. The expected prediction error can be decomposed into three terms:

$$\mathbb{E}[(y - \hat{f}(x))^2] = \text{Bias}(\hat{f}(x))^2 + \text{Var}(\hat{f}(x)) + \sigma^2 \tag{13}$$

where $\text{Bias}(\hat{f}(x))$ is the bias of the estimator, $\text{Var}(\hat{f}(x))$ is the variance of the estimator, and σ^2 is the irreducible error.

Bagging can reduce the variance of the base learners by averaging their predictions, which helps to improve the overall accuracy of the ensemble.

4.2 Bagging Algorithm

The bagging algorithm works as follows:

Algorithm 2 Bagging Algorithm

```
Require: Training set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, base learner $L$, number of bootstrap samples $B$
1: for $b = 1, \ldots, B$ do
2:     $S_b \leftarrow$ bootstrap sample from $S$
3:     $f_b \leftarrow L(S_b)$
4: end for
5: return $\hat{f}(x) = \frac{1}{B}\sum_{b=1}^{B} f_b(x)$ (for regression) or $\hat{f}(x) = \text{argmax}_c \sum_{b=1}^{B} \mathbb{I}(f_b(x) = c)$ (for classification)
```

4.3 Advantages of Bagging

Bagging has several advantages over using a single base learner:

- Bagging reduces the variance of the base learners, which can lead to improved accuracy and stability.
- Bagging is simple to implement and can be parallelized easily, as each base learner is trained independently.
- Bagging can be used with any base learner, making it a versatile ensemble method.

4.4 Limitations of Bagging

Despite its advantages, bagging has some limitations:

- Bagging can only reduce the variance of the base learners, not the bias. If the base learners are biased, bagging will not improve their bias.
- Bagging can be computationally expensive, as it requires training multiple base learners on different bootstrap samples.
- Bagging may not work well with stable base learners, such as linear models, as they will produce similar predictions on different bootstrap samples.

5 Random Forests

Random forests are an extension of bagging that use decision trees as the base learners. In addition to bootstrapping the training data, random forests also randomly select a subset of features at each split in the decision trees. This helps to decorrelate the trees and further reduce the variance of the ensemble.

5.1 Random Forest Algorithm

The random forest algorithm works as follows:

The main difference between random forests and bagging is the use of a random subset of features at each split in the decision trees. This helps to reduce the correlation between the trees and further improve the variance reduction of the ensemble.

Algorithm 3 Random Forest Algorithm

```
Require: Training set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, number of trees $B$, number of features to consider at each split $m$
 1: for $b = 1, \ldots, B$ do
 2:     $S_b \leftarrow$ bootstrap sample from $S$
 3:     $f_b \leftarrow$ RandomizedTreeLearn($S_b, m$)
 4: end for
 5: return $\hat{f}(x) = \frac{1}{B}\sum_{b=1}^{B} f_b(x)$ (for regression) or $\hat{f}(x) = \text{argmax}_c \sum_{b=1}^{B} \mathbb{I}(f_b(x) = c)$ (for classification)
 6: function RandomizedTreeLearn($S, m$)
 7:     if all examples in $S$ have the same class or other stopping criteria met then
 8:         return a leaf node with the class label or regression value
 9:     end if
10:     $F \leftarrow$ random subset of $m$ features
11:     $a^* \leftarrow \text{argmax}_{a \in F}\, \text{IG}(S, a)$
12:     Create a root node with feature $a^*$
13:     for each value $v$ of $a^*$ do
14:         Create a branch for condition $a^* = v$
15:         $S_v \leftarrow \{(x, y) \in S | x_{a^*} = v\}$
16:         if $S_v$ is empty then
17:             Create a leaf node with the most common class or average regression value in $S$
18:         else
19:             Recursively call RandomizedTreeLearn($S_v, m$)
20:         end if
21:     end for
22: end function
```

5.2 Advantages of Random Forests

Random forests have several advantages over decision trees and bagging:

- Random forests can achieve higher accuracy than decision trees and bagging by reducing both the bias and variance of the ensemble.
- Random forests are less prone to overfitting than decision trees, as the randomization of features and bootstrap sampling helps to prevent the trees from learning noise in the data.
- Random forests can handle high-dimensional data and feature interactions well, as the random feature subsets allow different trees to focus on different subsets of features.
- Random forests provide a built-in feature importance measure, which can be used for feature selection and interpretation.

5.3 Limitations of Random Forests

Despite their advantages, random forests have some limitations:

- Random forests can be computationally expensive, especially for large datasets or a large number of trees.
- Random forests are less interpretable than decision trees, as the ensemble combines the predictions of many trees.
- Random forests may not perform well on datasets with a large number of noisy or irrelevant features, as the random feature subsets may still include these features.

6 Comparison of Decision Trees, Bagging, and Random Forests

Table 1 summarizes the main differences between decision trees, bagging, and random forests.

	Decision Trees	Bagging	Random Forests
Base Learner	-	Any	Decision Trees
Bootstrap Sampling	-	Yes	Yes
Random Feature Subsets	-	No	Yes
Bias Reduction	-	No	Yes
Variance Reduction	-	Yes	Yes
Interpretability	High	Low	Low
Computational Cost	Low	High	High

Table 1: Comparison of decision trees, bagging, and random forests.

In general, random forests are the most powerful among the three methods, as they can reduce both bias and variance and handle high-dimensional data well. However, they are also the most computationally expensive and least interpretable. Decision trees are the simplest and most interpretable, but they are prone to overfitting and instability. Bagging is a good compromise between decision trees and random forests, as it can reduce variance and improve accuracy while being more interpretable than random forests.

7 Conclusion

Decision trees are a powerful and interpretable non-linear learning method that can be used for both classification and regression tasks. Bagging and random forests are ensemble learning methods that combine multiple decision trees to improve the stability and accuracy of the model. By understanding the basics of information theory, decision tree construction, bagging, and random forests, you can effectively apply these methods to a wide range of real-world problems.

A Derivations

A.1 Entropy

The entropy of a discrete random variable X with probability mass function $p(x)$ is derived as follows:

$$\begin{aligned} H(X) &= -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x) \\ &= -\mathbb{E}[\log_2 p(X)] \\ &= -\int p(x) \log_2 p(x) dx \end{aligned}$$

The last step follows from the definition of expected value for a continuous random variable.

A.2 Conditional Entropy

The conditional entropy of a random variable Y given another random variable X is derived as follows:

$$\begin{aligned} H(Y|X) &= \sum_{x \in \mathcal{X}} p(x) H(Y|X = x) \\ &= \sum_{x \in \mathcal{X}} p(x) \left(-\sum_{y \in \mathcal{Y}} p(y|x) \log_2 p(y|x) \right) \\ &= -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log_2 p(y|x) \\ &= -\mathbb{E}[\log_2 p(Y|X)] \end{aligned}$$

The third step follows from the definition of joint probability $p(x, y) = p(x)p(y|x)$.

A.3 Mutual Information

The mutual information between two random variables X and Y is derived as follows:

$$\begin{aligned} I(X; Y) &= H(X) - H(X|Y) \\ &= -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x) + \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log_2 p(x|y) \\ &= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log_2 \frac{p(x, y)}{p(x)p(y)} \\ &= D_{\text{KL}}(p(x, y) \parallel p(x)p(y)) \end{aligned}$$

The last step follows from the definition of KL-divergence, showing that mutual information is equivalent to the KL-divergence between the joint distribution $p(x, y)$ and the product of marginal distributions $p(x)p(y)$.

A.4 Information Gain

The information gain for splitting a dataset S on an attribute a is derived as follows:

$$\begin{aligned}
\text{IG}(S, a) &= H(S) - \sum_{v \in \text{Values}(a)} \frac{|S_v|}{|S|} H(S_v) \\
&= -\sum_{c=1}^{C} p_c \log_2 p_c + \sum_{v \in \text{Values}(a)} \frac{|S_v|}{|S|} \sum_{c=1}^{C} p_{c,v} \log_2 p_{c,v} \\
&= -\sum_{c=1}^{C} p_c \log_2 p_c + \sum_{v \in \text{Values}(a)} \sum_{c=1}^{C} p(c, v) \log_2 p(c|v) \\
&= I(C; A)
\end{aligned}$$

where p_c is the proportion of examples in S that belong to class c, $p_{c,v}$ is the proportion of examples in S_v that belong to class c, and $p(c, v) = \frac{|S_{c,v}|}{|S|}$ is the joint probability of class c and attribute value v. The last step follows from the definition of mutual information, showing that information gain is equivalent to the mutual information between the class variable C and the attribute A.