Eighth essay Raul Adell

This essay may contain ideas given in class complemented by the book given on the references of the course as well as content from Stanford's online ITSL course given by R. Tibshirani and T. Hastie. Book section: 8. Tree-based methods. Along with reading the book chapter, and contrasting it with online videos and class notes, this is what I considered most important.

The Basics of Decision Trees:

Decision Trees are a popular and powerful tool in the field of machine learning. They are simple to understand, easy to interpret, and can be used for both classification and regression tasks. Decision Trees are a type of supervised learning algorithm that can be used for both classification and regression. They build models in the form of tree-like structures, where each 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 numerical value.

To build a decision tree, we start at the root node and recursively split the data into smaller subsets based on the values of the input features. The goal is to partition the data in a way that minimizes the impurity of the resulting subsets. There are different measures of impurity that can be used, such as entropy, Gini impurity, or classification error. The splitting process continues until we reach a stopping criterion, such as a maximum tree depth, a minimum number of samples per leaf, or a minimum improvement in impurity.

The mathematical formulation of Decision Trees involves minimizing the impurity of the resulting subsets. For a binary classification problem, let p be the proportion of positive samples in a subset S. The Gini impurity of S can be defined as:

$$Gini(S) = 2p(1-p)$$

The Gini impurity ranges from 0 (when all samples in S belong to the same class) to 0.5 (when the proportion of positive and negative samples is equal). The goal of the splitting process is to find the feature and the threshold that minimize the weighted average of the impurity of the resulting subsets. Let S1 and S2 be the subsets resulting from splitting S based on feature j and threshold t. The weighted Gini impurity of the split can be defined as:

$$Gini_{split}(S,j,t) = \frac{|S_1|}{|S|}Gini(S_1) + \frac{|S_2|}{|S|}Gini(S_2)$$

The splitting criterion is then defined as:

$$\min_{j,t} Gini_{split}(S,j,t)$$

Observe that the weights can be very helpful when dealing with unbalanced data, like in the practice session #6. This criterion can be optimized in several ways, but it is not the main focus of the essay.

One of the strengths of Decision Trees is their ability to handle both numerical and categorical data. This is because the splitting criterion can be defined based on the type of data. For example, for numerical data, we can use a threshold-based splitting criterion, while for categorical data, we can use a rule-based splitting criterion.

Another interesting property of Decision Trees is their ability to handle collinearity and normalization differently than other methods such as regression. Collinearity occurs when two or more input features are highly correlated with each other, which can lead to instability and high variability in the regression coefficients. In Decision Trees, collinearity is less of a concern because the splitting process considers only one feature at a time. This means that even if two features are highly correlated, the algorithm may still select one of them as the best feature to split on based on the impurity reduction.

Normalization, on the other hand, is not necessary for Decision Trees because the splitting process is based on the values of the features relative to each other, rather than their absolute values. This means that scaling or centering the data does not affect the performance of the algorithm.

In addition, Decision Trees have some other interesting properties that make them different from other methods. For example, they can handle non-linear relationships between the input features and the output variable, which is not always possible with linear models. They can also handle missing data by simply assigning the missing values to a separate branch of the tree.

Furthermore, Decision Trees have the ability to provide feature importance scores, which can be useful in feature selection or interpretation. These scores are based on the number of times a feature is used for splitting across all the trees in an ensemble, such as Random Forests.

Regression Trees:

Regression Trees are a type of Decision Tree that are used for regression problems, where the goal is to predict a continuous numerical output variable. In contrast to classification problems, where the output variable is discrete, regression problems require a different impurity measure and splitting criterion.

To build a Regression Tree, we start at the root node and recursively split the data into smaller subsets based on the values of the input features. The goal is to partition the data in a way that minimizes the variance of the resulting subsets. The variance of a subset S can be defined as:

$$Var(S) = \frac{1}{|S|} \sum_{i \in S} (y_i - \bar{y}_S)^2$$

where y_i is the output value of the i-th sample in S, and $\bar{y}_S$ is the mean output value of S. The splitting process continues until we reach a stopping criterion, such as a maximum tree depth, a minimum number of samples per leaf, or a minimum improvement in variance.

The splitting criterion for Regression Trees is similar to that of Decision Trees, but uses the variance reduction instead of the impurity reduction. Let S1 and S2 be the subsets resulting from splitting S based on feature j and threshold t. The variance reduction of the split can be defined as:

$$R_{split}(S, j, t) = Var(S) - \frac{|S_1|}{|S|} Var(S_1) - \frac{|S_2|}{|S|} Var(S_2)$$

One interesting property of Regression Trees is that they can handle non-linear relationships between the input features and the output variable. This is because the splitting process can create complex interactions between the features, which can capture non-linearities that may not be possible with linear models.

Another interesting property of Regression Trees is that they can provide a piecewise constant approximation of the input-output relationship. This means that the output variable is constant within each leaf node, and the output values of the samples within a leaf node are equal to the mean output value of that node. This can be useful for interpretability, as it allows us to understand the input-output relationship in terms of simple decision rules.

Classification Trees:

Classification Trees are a type of Decision Tree that are used for classification problems, where the goal is to predict a discrete output variable. The splitting criterion for Classification Trees is based on the impurity reduction, which can be measured using different impurity measures such as Gini impurity or entropy.

To build a Classification Tree, we start at the root node and recursively split the data into smaller subsets based on the values of the input features. The goal is to partition the data in a way that minimizes the impurity of the resulting subsets. The impurity of a subset S can be defined using one of the impurity measures mentioned above.

The splitting criterion for Classification Trees is similar to that of Decision Trees, but uses the impurity reduction instead of the variance reduction. Let S1 and S2 be the subsets resulting from splitting S based on feature j and threshold t. The impurity reduction of the split can be defined as:

$$R_{split}(S, j, t) = I(S) - \frac{|S_1|}{|S|} I(S_1) - \frac{|S_2|}{|S|} I(S_2)$$

where I(S) is the impurity of subset S, and I(S1) and I(S2) are the impurities of subsets S1 and S2, respectively.

The splitting criterion is then defined as:

$$\max_{j,t} R_{split}(S, j, t)$$

One interesting property of Classification Trees is that they can handle non-linear decision boundaries between the input features and the output variable. This is because the splitting process can create complex interactions between the features, which can capture non-linearities that may not be possible with linear models.

Another interesting property of Classification Trees is that they can provide interpretable models in the form of decision rules. Each leaf node corresponds to a decision rule that maps a set of input features to a class label. These decision rules can be easily understood and communicated to stakeholders, which can be useful for applications such as fraud detection or medical diagnosis.

Ensamble methods

Decision Trees are powerful and versatile models that can be used for both regression and classification tasks. However, they can also suffer from some limitations, such as high variance and instability to small changes in the data. Bagging, Random Forests, Boosting, and Bayesian Additive are ensemble methods that can address these issues and improve the performance of Decision Trees.

Bagging (Bootstrap Aggregating) is an ensemble method that combines multiple models to improve their performance. The basic idea of Bagging is to train multiple models on different subsets of the training data, and then aggregate their predictions to obtain a final output.

The subsets of the training data are created by bootstrapping, which involves randomly sampling the training data with replacement. This means that some samples may appear multiple times in a subset, while others may not appear at all. By doing so, Bagging can create diverse subsets of the data, which can lead to more robust and accurate models.

The predictions of the individual models are combined by taking a simple average (for regression problems) or a majority vote (for classification problems). This aggregation process can reduce the variance of the model and improve its generalization performance.

The number of models to be trained in Bagging can be chosen by cross-validation or other methods. The individual models can be trained using any algorithm, but Decision Trees are often used because of their flexibility and ease of interpretation.

The benefits of Bagging can be seen mathematically by considering the variance of the models. Let Y be the true output variable, and let Y^ be the predicted output variable from a model. The variance of Y can be decomposed into three terms:

$$Var(Y^{)}) = E[(Y^{-}E[Y^{]})^2] = \rho + \sigma^2$$

where rho is the squared bias, sigma^2 is the variance of the model, and E[.] denotes the expected value. The squared bias represents the difference between the average prediction of the model and the true output, while the variance represents the variability of the predictions around their average.

Bagging can reduce the variance of the models by decreasing the correlation between the individual models. Let Y^1, Y^2, ..., Y^B be the predictions of the individual models, and let Y^B be the average prediction. The variance of Y^B can be expressed as:

$$Var(Y^B) = \frac{1}{B^2}\sum_{b=1}^{B} Var(Y^b) + \frac{2}{B^2}\sum_{b=1}^{B}\sum_{b'=1}^{B} Cov(Y^b, Y^{b'})$$

where Cov(.,.) denotes the covariance. The first term represents the average variance of the individual models, and the second term represents the pairwise covariance between the models. Bagging can reduce the second term by creating diverse subsets of the data, which can lead to uncorrelated models.

Random Forests is a variant of Bagging that further reduces the correlation between the individual models by introducing randomization in the feature selection process. The basic idea of Random Forests is to train multiple Decision Trees on different subsets of the training data and a subset of the input features, and then aggregate their predictions to obtain a final output.

The subsets of the training data and input features are created by random sampling. In each split of the Decision Tree, only a subset of the input features is considered. This randomization can lead to different splits in different trees, which can decorrelate the models and reduce their variance.

The predictions of the individual trees are combined by taking a simple average (for regression problems) or a majority vote (for classification problems). This aggregation process can further reduce the variance of the model and improve its generalization performance.

The number of trees and the size of the subsets can be chosen by cross-validation or other methods. The randomization parameters can also be tuned to balance the bias-variance trade-off of the model.

The benefits of Random Forests can be seen mathematically by considering the variance of the models. Similar to Bagging, the variance of the predictions can be decomposed into the squared bias and the variance of the model. Random Forests can reduce the variance of the model by creating diverse subsets of the data and input features, and by decreasing the correlation between the individual trees.

Random Forests also have some interesting properties with respect to collinearity and normalization. Collinearity among the input features can be handled naturally by the randomization process, which can select different subsets of features in different trees. Normalization of the input features is not necessary in Random Forests, because the trees are not sensitive to monotonic transformations of the features.

Boosting can be applied to Decision Trees in a similar way as to other weak learning models. The main difference is in the choice of the loss function and the update rule.

For regression problems, the loss function used in Boosting for trees is the mean squared error (MSE), which measures the average squared difference between the predicted and actual values of the response variable. The update rule is to fit a tree to the residuals of the previous model, i.e., the difference between the actual and predicted values.

For classification problems, the loss function used in Boosting for trees is the exponential loss, which measures the negative log-likelihood of the predicted class probabilities. The update rule is to fit a tree to the negative gradient of the loss function, which is equivalent to the residual of the log-odds ratio.

The trees in Boosting are typically shallow, with a small number of nodes and splits, to avoid overfitting and increase the diversity of the models. The number of iterations (i.e., the number of trees) can be chosen by cross-validation or other methods, and the learning rate (i.e., the shrinkage parameter) can be tuned to control the contribution of each model.

The main advantage of Boosting for trees is that it can improve the accuracy of the models by focusing on the samples that are difficult to classify. The models are trained iteratively, and each subsequent model tries to correct the mistakes of the previous models. This can lead to a significant reduction in the bias and variance of the models, and a better fit to the data.

The main drawback of Boosting for trees is that it can be sensitive to noisy or irrelevant features, which can lead to overfitting or poor performance. Feature selection and regularization techniques can be used to address this issue, but they can also reduce the flexibility and expressive power of the models.

Bayesian Additive Regression Trees (BART) is a Bayesian model that combines the flexibility of Decision Trees with the regularization of Bayesian methods. The idea behind BART is to model the response variable as a sum of tree-based predictors, where each tree is a member of a large ensemble and contributes a small amount to the final prediction.

The BART model can be written as:

$$y = f(x) + \varepsilon$$

where y is the response variable, f(x) is the sum of the tree-based predictors, x is the vector of predictors, and ε is the error term. The tree-based predictors are modeled as:

$$f(x) = \sum t=1\text{^}T\ g(x, \theta\_t)$$

where T is the number of trees, g(x, θ_t) is the prediction of the t-th tree for the input vector x, and θ_t is the vector of parameters that define the structure and splits of the tree.

The Bayesian nature of BART comes from the prior distributions placed on the parameters θ_t and the error term ε. The prior distributions provide a way to incorporate prior knowledge or assumptions about the parameters and to control the complexity and smoothness of the models.

The posterior distribution of the parameters and the tree-based predictors can be estimated using Markov Chain Monte Carlo (MCMC) methods or other Bayesian inference techniques. The posterior distribution provides a way to compute posterior means, variances, credible intervals, and other probabilistic measures of uncertainty and prediction.

The main advantages of BART are its flexibility, robustness, and ability to handle high-dimensional data and nonlinear relationships between the predictors and the response. BART can capture complex interactions and nonlinearities that are difficult to model with other methods, and can also handle missing data and outliers.

The main drawbacks of BART are its computational complexity and the difficulty of interpreting the tree-based predictors. BART requires the estimation of a large number of parameters and the sampling of a large number of trees, which can be computationally intensive and time-consuming. The tree-based predictors are also difficult to interpret and visualize, especially for large ensembles and high-dimensional data.

We try to make now a final recap of the ensemble methods. Bagging and Random Forests are closely related methods that reduce the variance and overfitting of individual trees by using bootstrapping and aggregation. Bagging creates an ensemble of trees by training each tree on a random subset of the training data with replacement, and averaging the predictions of the trees. Random Forests improve Bagging by adding an additional level of randomness that randomly selects a subset of the predictors for each tree, which reduces the correlation between the trees and further reduces the variance and overfitting. Both methods are computationally efficient and suitable for high-dimensional data and noisy or correlated predictors.
Boosting is a method that sequentially trains a sequence of weak trees that are designed to improve the predictions of the previous trees. Boosting works by iteratively adjusting the weights of the training data to focus on the observations that are most difficult to predict, and by combining the predictions of the trees with weighted averages. Boosting methods such as AdaBoost and Gradient Boosting are more computationally intensive than Bagging and Random Forests, but can achieve higher accuracy and robustness, especially in low-dimensional settings.

Bayesian Additive Regression Trees (BART) is a Bayesian model that combines the flexibility of Decision Trees with the regularization of Bayesian methods. BART models the response variable as a sum of tree-based predictors, where each tree is a member of a large ensemble and contributes a small amount to the final prediction. BART is a powerful and flexible method that can handle high-dimensional data and nonlinear relationships, but requires substantial computational resources and expertise in Bayesian inference.