

TDK-dolgozat

Szerző:

Bán Péter

University of Miskolc



FACULTY OF MECHANICAL ENGINEERING AND INFORMATICS
INSTITUTE OF INFORMATICS

SCIENTIFIC STUDENT CONFERENCE THESIS
Analysis of multiway split regression trees

WRITTEN BY:

Bán Péter

student of the Computer Science Engineering study program at BSc/MSc level

SUPERVISOR:

Prof. Dr. Kovács László

associate professor

Miskolc, 2024.

Introduction

One of the most well-known topics in information technology today is artificial intelligence. However, accurately defining this field is not a simple task. It's not even clear what we consider it intelligence, as it is debatable. Alan Turing developed a test[35] to determine whether artificial intelligence is truly intelligent. In this test, if a person cannot distinguish between a "machine" and a human after a conversation, the machine can be considered intelligent. The problem with this test is that it only covers a small aspect of true intelligence. Therefore, it is more useful to think of artificial intelligence as a combination of algorithms, methods, and tools that are capable of learning from a dataset and solving problems that require human-like thinking.[1]

The application of artificial intelligences aids and accelerates the solution of certain problems. The areas of focus include regression and classification, which encompass various algorithms. In classification, elements are grouped into classes based on existing attributes, while in regression, the goal is to determine a real value based on the attributes. Several tools have been developed for these tasks, including decision trees. [5]

Decision trees have proven their usefulness in solving real-world problems in multiple analyses. For example, the decision tree method has been successfully applied to assess the likelihood of flooding in certain areas throughout a year and to identify the factors that can predict the occurrence of floods [2]. The CART algorithm was used in this context and achieved 83% accuracy, even allowing for the creation of a map showing which areas are more prone to flooding.[2]

In the energy sector, particularly in the context of renewable energy, decision trees have also been highly beneficial. To reduce the time required to determine the ideal location for a wind farm (which would otherwise involve lengthy measurements), decision trees were used to generate the necessary information. The results closely matched the real data, achieving an accuracy of 95%.[3]

Decision trees were primarily designed for classification, and the majority of examples in the literature also focus on this use. However, with proper adaptation, they can also perform regression tasks. Two widely known and used decision tree algorithms are ID3 and CART. The former is used for classification, while the latter can handle both classification and regression. The ID3 algorithm was published by J. Ross Quinlan in 1986 [4]. He emphasized that there are multiple areas within artificial intelligence. Some systems try to improve their performance by adjusting their internal parameters (fine-tuning) based on their own results. Another approach is when a system acquires structured knowledge during learning. However, the methods available at the time had limitations, as it would have required teaching the systems hundreds or even thousands of

rules.[4] ID3 is a simple knowledge-based system that is easy to train. It was designed for classification, and there are now several improved versions, one of the most well-known being C4.5. This improvement was necessary because ID3 was sensitive to attributes with relatively many target values. [4] In real life, many data—such as temperature or weight—consists of continuous values, making classification an unsuitable method. In such cases, a regression tree can be used.

Related results

Supervised Learning

In supervised learning, the algorithm is trained using a training set T , in which objects can be divided into two main parts: dependent and independent variables. Independent variables consist of attributes, while dependent variables are the values we want to determine based on the independent variables. In classification, these values are discrete, while in regression, they are continuous. Based on the independent variables, the algorithm tries to predict the corresponding dependent variables. Adjustments are made to the model based on its accuracy, so that it can provide more accurate results in the future. Neural networks, for example, learn in this way. This method can further be divided into two parts: classification and regression. Classification assigns objects to predefined classes, while regression aims to determine continuous values from the attributes with the highest accuracy possible.[5]

Decision Trees

The first decision tree in the literature appeared in 1963 [6], known as the Automatic Interaction Detection (AID). This decision tree was designed to solve regression tasks. Starting from the root node, the algorithm splits the data set into two parts at each node. “impurity” is the metric that determines how optimal a split is, and for this, the squared deviation from the mean was used. The algorithm executed the split at the point where this deviation was the smallest. Once a certain predefined heterogeneity was reached, the splitting stopped. The predicted value was the average of the target values in the final node.[7]

Later a significant development was the CART algorithm, published in 1984 [8], which could perform both regression and classification tasks. The name reflects its purpose: "Classification and Regression Tree." One of the key innovations of this algorithm is that instead of stopping the tree's growth based on a rule, it allows the tree to grow large and then prunes it afterward. This addresses

the issues of underfitting and overfitting, which were problems with AID. However, this approach requires more computational capacity. Additionally, it is capable of handling missing data.[7]

The ID3 (developed by J. Ross Quinlan in 1986) [7] is a classification tree that cannot handle regression tasks. It is a well-functioning algorithm, but it has its flaws. If an attribute has many possible values, it becomes biased toward that attribute, and it cannot handle noisy datasets well. C4.5, an improvement of ID3 developed by Quinlan in 1993 [7], corrects these flaws and also enables regression tasks. To avoid bias, C4.5 uses a new method for measuring node impurity: instead of Information Gain, it uses Gain Ratio, which can filter out biases in the dataset.[7]

These methods form the foundation of many other studies that continue to have an impact today, affecting not only experts but also the general public. In the field of research and development, decision trees have been used to assist engineers in their work. Systems have been designed to implement this through various methods, including KNN, SVM, decision trees, and regression trees. In one study [10], various methods and algorithms were tested on two datasets: one involving engines and the other involving compressors. The regression tree achieved around 90% accuracy in most cases. For determining stator height, Random Forest achieved 91% accuracy, LS BOOST 84%, and SVM Linear 86%. For determining rotor height, the regression tree achieved 89% accuracy, while KNN only reached 84%.[10]

Self-driving technology is rapidly advancing, and more cars are equipped with some form of it, though limitations still exist. Since weather conditions can greatly affect camera "visibility," a study used radar sensor data to determine road conditions, building a CART decision tree that yielded accurate results in approximately 95% of cases.[11]

The regression tree has also proven useful in the food industry. Researchers wanted to determine how to maintain the quality of blueberries until they reached the consumer. Factors such as temperature and humidity played a crucial role. A BRT model was used, which could predict blueberry weight loss with high accuracy, allowing for preventive measures to be taken.[12]

ID3 Algorithm

Let T be or training data consisting of pairs of predictor vector and response variables $(\vec{x}, y) \in T$, where y is qualitative.

The decision tree is constructed from top to bottom. Starting from the root node, a condition is defined, which determines how the dataset is split at each node. This process continues recursively, moving down the tree until the termination criteria is reached. The method of determining the

condition, performing the split, and when to stop varies depending on the implementation. The nodes, where the termination criteria is met are the terminal nodes.[14]

For the ID3 algorithm, the splitting continues until every attribute has been used at most once (to avoid infinite loops) or until the target variable set becomes homogeneous. The attribute that provides the highest "information gain" is used to split the dataset. Information gain helps determine the optimal attribute to use for splitting.[15]

To calculate information gain, we first need entropy, which measures the heterogeneity of a node. The formula for entropy is as follows:

$$E_n = - \sum_{i=1}^c p_i \times \log_2 p_i$$

Where E_n is entropy, c is the number of target categories, and p_i is the probability of the i th category. If entropy is 0, it means the set of target variables is entirely homogeneous. As entropy approaches 1, homogeneity decreases (entropy cannot exceed 1).[15]

For example, if we have two categories a and b and our set looks like $\{a, a, b, b\}$, the entropy would be calculated as:

$$E_n = - \left(\frac{1}{2} \times \log_2 \frac{1}{2} + \frac{1}{2} \times \log_2 \frac{1}{2} \right) = 1$$

This indicates a lack of homogeneity. However, if our set looks like $\{a, a\}$, the entropy would be:

$$E_n = -(1 \times \log_2 1) = 0$$

This means the dataset is perfectly homogeneous.

Information gain is the expected reduction in entropy when splitting on a particular attribute [14]. We compute the entropy of the parent node and subtract the weighted entropies of the child nodes.[14][15]

The formula for information gain is:

$$I_g = E_p - \frac{\sum_{i=1}^k (n_i \times E_i)}{N}$$

Where E_p is the entropy of the parent node, n_i is the size of the i th child node, E_i is the entropy of the i th child node, and N is the total number of objects.[15]

When constructing the tree, we recursively split the dataset using each attribute, calculating the entropy of the resulting subsets. We then choose the split with the highest I_g . Once all attributes are used or the target set is homogeneous, the tree stops building, and the categories at the leaf nodes represent the final decisions.[14][15]

C4.5 Algorithm

C4.5 is an extension of ID3, designed to address some of its limitations. A major issue in ID3 is that if an attribute has many possible target values, the algorithm tends to overemphasize that attribute, leading to sub optimal splits [7]. This issue has been observed in various real-world scenarios [7].

For instance, in one healthcare application, ID3 chose the "patient's age" as the splitting attribute, but experts determined that this attribute was less significant than others. ID3 made this choice because "patient's age" had relatively many values. [7]

To better understand the problem, imagine an attribute where all values are different and random. In such a case, the information gain I_g would be maximized, and ID3 would select this attribute for splitting, even though it's not useful for classification. [7]

To overcome this, C4.5 introduces a new criterion called "Gain Ratio," instead of using pure information gain. The Gain Ratio accounts for the bias present in ID3.

To compute the Gain Ratio, we first need another function called "Split Info,"[7] calculated as:

$$S_i = - \sum_i^n \frac{i}{p} \times \log_2 \frac{i}{p}$$

Where p is the proportion of target categories in the i -th test. The Split Info helps eliminate ID3's bias[7][16]. The Gain Ratio is then computed as:

$$G_r = \frac{I_g}{S_i}$$

Gain Ratio normalizes the information gain, providing a more realistic evaluation and preventing bias [7][16]. C4.5 also handles continuous-valued attributes [17]. These values are grouped into intervals by sorting the attribute values $A_1, A_2, \dots, A_u$ in increasing order and then assigning them into groups $C_1, C_2, \dots, C_v$. For example, if $A_i > A_m$ and $A_i \leq A_n$, then we assign A_i to group C_k , where A_m, A_n are the lower and upper bounds of C_k [17]

Regression

In a regression task, the goal is to determine continuous or real values in a mathematical sense. During regression analysis, we are given dependent and independent variables. Using the independent variables, the aim is to predict the target values as accurately as possible. The task is to use the best-fitting function from a family of functions within the model. Various methods exist for this, such as linear regression and polynomial regression. Polynomial regression can be reduced to linear regression, making it a special case of linear regression.[5][18][19][24][25]

Regression Tree

As the name suggests, a regression tree is capable of solving regression tasks. It does this by splitting the training set into parts at each node. To determine the value at which the split should occur, we sort our dataset based on a chosen attribute denoted by h . To find the best split point denoted by s , iterate over the dataset at the given node, then take the midpoint between two numbers and split the data at this value [15].

The formula for the candidate s' is:

$$s' = \frac{x_{h,i} + x_{h,i+1}}{2}$$

Where s' represents the candidate split point, $x_{h,i}$ is the i -th value for attribute h .

We continue this process until we have iterated through all the values in the given node. To choose the final s value, sum the variances of the two resulting sets, and the one with the smallest value is used for splitting the dataset.[15][21]

This method recursively builds the tree until we reach a predefined depth or have used every attribute a predetermined number of times [15]. Since an infinite loop cannot occur in this process, attributes can be used multiple times. If splits continue long enough, every element ends up in a separate node, ending the tree-building process. However, this is not optimal as it leads to excessive splits and overfitting, meaning the model won't generalize well.

Linear Regression

In linear regression, we create a function or linear model that, based on an attribute, can predict target values. The purpose of using a linear model is to exploit the linear relationships present in the dataset.[19][22]

The univariate linear model can be written as:

$$\hat{y} = ax + b$$

Where $\hat{y}$ is the function value, a is the slope of the function, x is the input value, and b is the intercept.

For multivariate linear regression:

$$\hat{y} = \theta_0 x_0 + \theta_1 x_1 + \dots + b$$

Where $\hat{y}$ is the function value, θ_i is the coefficient for input x_i and b is the intercept. [23]

Polynomial Regression

In polynomial regression, we create a polynomial model that, based on the attributes, predicts the target value. Polynomial regression is a special case of linear regression. For instance, if we interpret x^2 not as the square of x but as an entirely independent variable, we've reduced the problem to multivariate linear regression.[24][25]

The polynomial model can be rewritten as:

$$\hat{y} = \theta_0 x_0^{k_0} + \theta_1 x_1^{k_1} + \dots + b$$

Where $\hat{y}$ is the function value, θ_i is the coefficient for input x_i , and b is the intercept.

We aim to make the linear model fit the dataset as well as possible. To achieve this, an error function is used. One commonly used error function is the Mean Squared Error (MSE)[25], calculated as:

$$MSE = \frac{1}{N} \times \sum_{i=1}^N (\hat{y}_i - y_i)^2$$

Where N is the size of the dataset, $\hat{y}_i$ is the predicted value for the i -th case, and y_i is the actual result.

The goal is to minimize MSE, which occurs when the function's derivative equals zero, indicating an extremum. When this minimum is reached, we retain the function parameters for future use.

To obtain these parameters, the Gradient Descent method can be applied. This iterative method helps find the minimum of the MSE function. Initially, random parameters are used, as the optimal ones are unknown. We then compute the partial derivatives over the entire dataset, multiply them by the learning rate L , and update the parameters iteratively until a maximum number of iterations is reached or the step size becomes too small.[25][26]

In a univariate linear model, the parameter updates are:

$$a = a - \sum_{i=1}^n \frac{\partial MSE}{\partial a} \times L$$

$$b = b - \sum_{i=1}^n \frac{\partial MSE}{\partial b} \times L$$

Where L is the learning rate, which determines the size of the steps along with the slope at a given point on the Mean Squared Error (MSE) function, helping to approximate the function's minimum. During the process, we calculate the partial derivatives for the entire dataset per iteration, then multiply them by L, which gives us the step size. Finally, we subtract this from our previous parameter, and we repeat this until we reach a maximum number of iterations or the step size becomes too small.[25][26]

One major issue in regression is underfitting and overfitting. This means that the model was not able to generalize properly based on the dataset. This can result from deficiencies in the model, inappropriate parameters, or an inadequate dataset.[5]

Overfitting and Underfitting

A common issue in regression is overfitting and underfitting, meaning the model either fails to generalize based on the dataset or overfits to it. This can result from inadequate model structure, improper parameters, or an unsuitable dataset.[27]

M5 Model Regression Tree

A function-class-based regression tree differs from other regression trees in that the nodes and leaves contain multivariate linear models rather than just values. As shown in J.R. Quinlan's research on the M5 algorithm [27], this approach builds more accurate decision trees and handles previously unseen cases better because the linear model can extend to unseen values.[27]

Hierarchical Agglomerative Clustering (HAC)

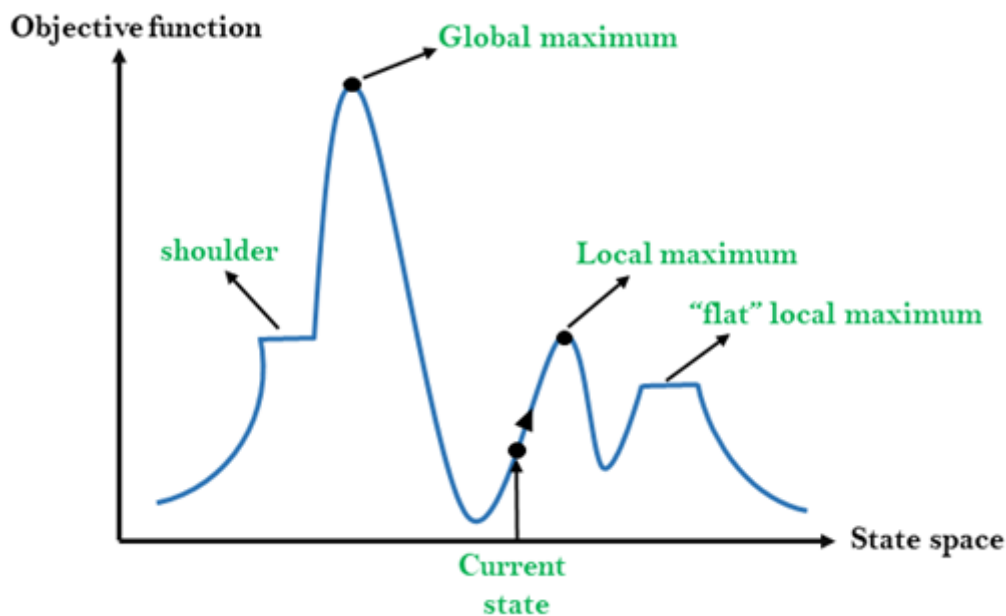
HAC is an unsupervised algorithm using a bottom-up approach. Initially, each data point forms its own cluster. Then, the distances between clusters are calculated, and the two closest ones are merged. This process continues until a certain criterion, such as a predefined number of clusters, is met. The algorithm allows the use of different methods to calculate cluster distances, such as Ward, complete, single, and average methods. By visualizing the process in a dendrogram, one can track the formation of clusters, providing valuable insights.[28]

Hill Climbing

Hill climbing is a widely used algorithm that can be used for both search and optimization problems. Regarding the algorithm that will be introduced later, here I will focus on how it can be used for optimization. Hill climbing uses a heuristic approach to find an optimal solution. Hill climbing algorithm uses an initial position chosen arbitrary or randomly and generate solutions. After evaluating these solutions, it does a small adjustment into the direction which provides a solution closer to the minimum. It continues until it reaches a local minimum. The problem of this algorithm is that if it has found a minimum it stops although it might not be the global.[36][37]

There are different types of hill climbing algorithms. Simple Hill climbing algorithm evaluates each neighbouring solutions and selects the first one which is an improvement to the current state. In the other hand Steepest-Ascent hill climbing evaluates all neighbouring solutions and chooses the one with largest improvement. Stochastic Hill Climbing introduces randomness by choosing a random neighbouring solution, which broadens the search for the global optimum. The main point of this variation is that it might not get stuck at a local optimum.[36][37]

The hill climbing algorithm has some difficulties when the neighbouring solutions are similar value which makes it hard to decide in which direction to move and it is called flat local maximum.[36][37]



1.figure[38] showing how hill climbing works

Existing Implementations

Nowadays, there are numerous implementations available on the internet. The implementations I will introduce are found in the Scikit-learn Python library.

Decision Tree

If you want to use the decision tree for classification, you can use the `DecisionTreeClassifier` class. It takes two arrays as input. One is `X`, which is a two-dimensional array with the dimensions `n_samples` (number of samples) and `n_features` (number of features), containing the attributes, and the `y` array, which contains the classes corresponding to the attributes. [29][30] After creating an instance of the `DecisionTreeClassifier` class, you can build a tree using the `fit()` method. Finally, the `predict()` method can be used to return the most likely class for the given attributes based on the tree structure. Alternatively, you can use the `predict_proba()` method, which returns the probability of each class along the given branch. [29][30]. For regression tasks, you can use the `DecisionTreeRegressor` class. Similar to classification, you need to pass `X` and `y` arrays, but in this case, the `y` array should contain floating-point values instead of integers. [29][30]. The `fit()` and `predict()` methods can be used similarly to what was described above. [29][30]

HAC

For HAC (Hierarchical Agglomerative Clustering), you can use the `AgglomerativeClustering` class. When creating an instance, you can specify various parameters. The `n_clusters` parameter specifies the number of clusters after which the merging should stop. The `linkage` parameter allows you to choose different methods for calculating the distance between two clusters. In the library, the following methods are available: `ward`, `complete`, `average`, and `single`. [28][31]. In the `ward` method, the two clusters whose variation is least different are merged. In the `average` method, the distance is the average of the distances between all points in one cluster and all points in another cluster. The `complete` method uses the most distant points between two clusters, while the `single` method uses the closest points. [28][31]

Linear Regression/Polynomial Regression

For linear regression, you can use the `LinearRegression` class. This class creates a linear model with coefficients that minimize the sum of squared differences between the target values in the dataset and the predicted values. [32][33]. The `fit()` method creates a linear model that fits the dataset. The `X` array is a two-dimensional array with dimensions `n_samples` (number of samples) and `n_features` (number of features), containing the attributes, while the `y` array contains the target values corresponding to the attributes. [32][33]. Polynomial regression is actually a special case of linear regression, so you can still use the `LinearRegression` class, but you will need to transform

the data. The PolynomialFeatures class's fit_transform() method can be used to transform the dataset accordingly. It is important to specify the degree of the polynomial. It will then generate a matrix containing the polynomial combinations of the attributes up to the specified degree. [32][33]

Multiway splitting

Regression tree algorithm splits the dataset in a binary manner. This method is widely used because it has proven to be effective and reliable. However, in case of binary splitting, more splits are needed to build a tree than if we were using non-binary splits.

The ID3 algorithm does not split the dataset in a binary manner; instead, it splits based on what reduces entropy the most. Despite not being applicable to regression tasks, the model serves as an excellent example of non-binary splitting. As a result, the number of calculations is reduced, meaning the algorithm completes its assigned task in less time. Additionally, the tree will not be as deep, which reduces the depth of the tree. Consequently, since we build the tree recursively, reducing the tree's depth also reduces memory requirements, preventing issues where available memory is exhausted while constructing a specific branch.

Non-binary splitting thus not only saves time but also energy, which is especially costly today. Moreover, it reduces environmental impact, a crucial consideration. The growing volume of datasets further emphasizes the need to train models more efficiently and quickly.

Parameters add yet another layer of complexity to this topic. Unfortunately, we cannot predict how a specific parameter will affect the outcome, so we are compelled to test various parameters. This means that the model needs to be trained multiple times during the process.

Beyond this, the task at hand may also require the model to be trained multiple times. It may be necessary for the model to be adaptive or at least capable of learning from newly available or expanded datasets on a regular basis. To use the updated model as soon as possible, and potentially accelerate development, we need to train our model more efficiently and quickly than before.

As shown by the research above, regression trees remain relevant and widely used today. Moreover, as we have seen, they can address a broad range of problems. For these reasons, it is reasonable to develop methods that accelerate model training.

It is generally accepted that multiway splitting for regression tasks will result in overfitting, and it will fragment the available data too quickly. It means there will be insufficient data as in deeper levels.[39][40]

The problem we aim to solve is one of optimization and error reduction. To address this effectively, I propose a new method: a non-binary regression decision tree.

Proposed Regression Tree Model

Multiway Split Regression Tree with HAC based optimization (MSRTHAC)

Let T be our training data consisting of predictor vector and response variables $(\vec{x}, y) \in T, y \in R$

For each splitting variable h , where the splitting variable indicates which attribute is chosen, it split the data into k partitions based on HAC algorithm.[44]

Initially each point of the dataset is a cluster. For each cluster it fits a linear model, and then it combines the two which are the closest to each other based on the similarities of the linear models.[44]

The linear model:

$$\hat{y} = ax + b$$

The error can be measured for each cluster using MSE:

$$L = \sum_{i=1}^n [(ax + b) - y_i]^2$$

Using the partial derivatives of the error function with respect to a and b we get the following:

Partial derivative for a :

$$\frac{\partial}{\partial a} = \sum_{i=1}^n (2ax_i^2 + 2x_i b - 2x_i y_i)$$

$$\frac{\partial}{\partial a} = \sum_{i=1}^n (ax_i^2 + x_i b - x_i y_i)$$

$$\frac{\partial}{\partial a} = a \sum_{i=1}^n x_i^2 + b \sum_{i=1}^n x_i - \sum_{i=1}^n x_i y_i$$

Partial derivative for b :

$$\frac{\partial}{\partial b} = \sum_{i=1}^n (2ax_i + 2y_i - 2b)$$

$$\frac{\partial}{\partial b} = a \sum_{i=1}^n x_i + \sum_{i=1}^n b - \sum_{i=1}^n y_i$$

With these variables we can calculate the distances between the partitions.

$$S_1 = \sum_{i=1}^n x_i$$

$$S_2 = \sum_{i=1}^n x_i^2$$

$$S_3 = \sum_{i=1}^n x_i y_i$$

$$S_4 = \sum_{i=1}^n y_i$$

$$D_{i,j} = \left(\frac{S_1^i}{N} - \frac{S_1^j}{N}\right)^2 + \left(\frac{S_2^i}{N} - \frac{S_2^j}{N}\right)^2 + \left(\frac{S_3^i}{N} - \frac{S_3^j}{N}\right)^2 + \left(\frac{S_4^i}{N} - \frac{S_4^j}{N}\right)^2$$

Where $D_{i,j}$ is the distance between the i th and j th cluster, and N is the number of data. The merging process continues until the number of clusters equal to the number of partitions we want.[44]

Multiway Splitting Regression Tree with Hill Climbing optimization (MSRTHC)

Let R be our training data consisting of pairs of predictor vector and response variables $(\vec{x}, y) \in T, y \in R$.

In the case of non-binary splits, we need multiple split points. Let k denote the number of partitions. It means that for k partitions there should be $k-1$ number of split points $p_1, p_2, \dots, p_{k-1}$. Regarding the initial position of the points, they are placed at an equal distance in the value range.

By initializing the splitting points, we also determined the base partitions, which are determined by the value of p_i

$$D_i^{base} = \{(\bar{x}, y) \mid p_{i-1} \leq x_h < p_i\}$$

Where D_i^{base} is the base partition for p_i .

For quality measurement we use the mean squared error as a lost function.

$$L(D) = \frac{1}{N} \times \sum_{i=1}^N (\hat{y}_i - y_i)^2, y_i \in y$$

The goal is to minimize the error. To lower the error of the partitions by moving the splitting points which will optimize the partitions. Multiway splits can be achieved by a series of binary splits [], which we want to avoid. Instead, we optimize the position of each individual split point for a given splitting variable, so we can find a partitioning that has a lower error.

To find the best splitting points we apply hill climbing heuristics.

It is important to note that unlike when we only have one split point, we cannot find the optimal position for each p_i one by one since the adjacent splitting points influence the partitions. Each p_i influence two partitions. Which means we must adjust them simultaneously.

For each p_i we add the learning value l based on the values of the given attribute. It will result in two new partitions for p_i .

When the l value is added it means expansion, and when it is subtracted it means shrinking.

The new candidate partitions can be defined as follows:

$$D_i^{base} = \{(\bar{x}, y) \mid p_{i-1} \leq x_h < p_i + l\}$$

After measuring whether the expansion or shrinking will result in lower error, we update the splitting point then, do the same for the rest of p_i . The base partitions are also updated. To measure if the new partitions are better or not, we compare them to the base partitions or in other words to the previous ones:

$$Error\ expansion = L(D_i^{base} \cup D_{i+1}^{base}) - \left(\frac{|D_i^+|}{|D_i|} L(D_i^+) + \frac{|D_i^-|}{|D_i|} L(D_{i+1}^-) \right)$$

$$Error\ shrink = L(D_i^{base} \cup D_{i+1}^{base}) - \left(\frac{|D_i^-|}{|D_i|} L(D_i^-) + \frac{|D_i^+|}{|D_i|} L(D_{i+1}^+) \right)$$

We continue this optimization until the error cannot be lowered furthermore.

Hybrid tree (hybrid splitting)

The hybrid tree is the variation of push tree. It combines the push tree algorithm and binary splitting. The test performed shows that there are some issues and as stated in this book [39] multiway split is not always preferable. But with hybrid trees we could get the best of both worlds. That is why initially the algorithm starts with multi-way split, and when a certain criterion is met, it switches to binary split.

Multiway splitting is generally better while there is a larger amount of data. Using multiple split points means that the model can find better partitions since it can take advantage of having several local optimums which can outweigh one local optimum for all the given values. But as there is less and less data, there is also less opportunity to have an edge over binary splitting. With low enough number of data there is simply not enough information for multiple split points. It makes sense that with low enough number of data just use one split point, since in that case it is similar to being one of the multiple split points. But since there is so few data it is worth trying all the split points, or in other words do binary splitting.

Test results

Datasets

Windmill

Source:[41]<https://www.kaggle.com/datasets/mubashirrahim/wind-power-generation-data-forecasting>

This dataset is a unique compilation of field-based meteorological observations and wind power generation data, collected directly from one of our company's operational sites. The data set represents a detailed hourly record, starting from January 2, 2017. This rich data set provides real-world insights into the interplay between various weather conditions and wind energy production.

Context and Inspiration: The data set was conceived out of the necessity to understand the dynamic relationship between meteorological variables and their impact on wind power generation. By collecting data directly from the field and the wind turbine installations, we aim to provide a

comprehensive and authentic data set that can be instrumental for industry-specific research, operational optimization, and academic purposes.

Data Collection: Data was meticulously gathered using state-of-the-art equipment installed at the site. The meteorological instruments measured temperature, humidity, dew point, and wind characteristics at different heights, while power generation data was recorded from the wind turbines' output. This dataset is a unique compilation of field-based meteorological observations and wind power generation data, collected directly from one of our company's operational sites. The dataset represents a detailed hourly record, starting from January 2, 2017. This rich dataset provides real-world insights into the interplay between various weather conditions and wind energy production.

The data set is made up of four parts, but we used only the first one since it contain 43800 elements.

Auto-Mpg Data

Source [42]<https://archive.ics.uci.edu/dataset/9/auto+mpg>

This dataset is a slightly modified version of the dataset provided in the StatLib library. In line with the use by Ross Quinlan (1993) in predicting the attribute "mpg", 8 of the original instances were removed because they had unknown values for the "mpg" attribute. The original dataset is available in the file "auto-mpg.data-original". "The data concerns city-cycle fuel consumption in miles per gallon, to be predicted in terms of 3 multivalued discrete and 5 continuous attributes." (Quinlan, 1993)

Graduate Admission

Source[43] <https://www.kaggle.com/datasets/mohansacharya/graduate-admissions>

This dataset was built with the purpose of helping students in shortlisting universities with their profiles. The predicted output gives them a fair idea about their chances for a particular university.

This dataset is inspired by the UCLA Graduate Dataset. The test scores and GPA are in the older format. The dataset is owned by Mohan S Acharya.

The dataset consists of 500 elements

House

The House dataset was created by us. It consists of four features and 10.000 elements. The attributes are randomly generated with a certain rule for each of them. The first attribute is size which values is in the range of 0 and 100. We randomly generate a number in the previously stated

range with one decimal precision and we calculate it's influence on the predictor value int the following way:

$$P_1 = \frac{\sqrt{size} * 10 + (0.3 * size)}{10}$$

The first attribute is distance which values is in the range of 0 and 100. We randomly generate a number in the previously stated range, and we calculate it's influence on the predictor value int the following way:

$$P_2 = \frac{1000}{distance} + 5$$

The Second attribute is supporting which values is in the range of 0 and 10. We randomly generate a number in the previously stated range, and we calculate it's influence on the predictor value int the following way:

$$P_3 = \text{support}$$

The first attribute is salary which values is in the range of 0 and 100. We randomly generate a number in the previously stated range, and we calculate it's influence on the predictor value int the following way:

$$P_4 = \text{salary}^3$$

We calculate the predictor value in the following way: The datasets we divided into two part with 80:20 ratio. We used the 80% for training and the 20% for testing. We used the train_test_split function from sckiti learn. The stopping criteria is related to leaf size. When the number of data point gets below a pre-determined number it stops splitting.

Summarizing table of the datasets

Dataset name	Number of data	Training data	Test data	Number of attributes	Task
windmill	43800	80%	20%	8	Regression
Auto-Mpg	392	80%	20%	7	Regression
Graduate Admission	500	80%	20%	7	Regression
House	10000	80%	20%	4	Regression

Implementation environment

To implement this new algorithm, I used the python language for various reasons. The python programming language is great for prototyping, since less code has to be written. A large number of libraries are available which saves a lot of time in development, not to mention these libraries are usually solves the given problem more effectively than most programmers could do. Not to mention that many algorithms are already implemented which could be used for testing as a baseline or use it as part of my own implementation.

Python 3.10.12 were used with various packages. Numpy for to handle data and execute operation on it, Sklearn for regression tree which serves as a base line. Matplotlib for visualization, and others like time for time measurements.

The python environment was handled by anaconda, and the underlying operating system was Ubuntu. The code was written in visual studio code.

Implementation

The dataset was the same for all methods and variations. The datasets were split into training and test sets, as described above. The termination criteria is a threshold for the minimum sample size that, when reached, that particular node becomes a terminal node. An error threshold was established to zero, so unless this value is reached (which is unlikely), no matter the error, splitting continues. If the termination criteria is met, further splitting stops regardless of sample size, as this provides a satisfactory result for our purposes. Finally, I use the test set to evaluate how accurately the resulting model performs on previously unseen data. I record the training time and error rate in a simple text file.

First, I created a class named Node, which stores all relevant information for each node. This includes the split points and split variable. The Node also contains the child Nodes in an array in sequential order.

The following functions are important in my implementation, which implements the mosot important parts of the algorithm.

Best split: For all attributes in x, it finds the optimal split points using hill climbing heuristics. After each split points are at an optimal value it evaluates that split then start over with the following attribute. After all attribute tested, it returns the best partitions with the corresponding attribute.

Sortf f: For the given attribute it sorts x and y , where y is corresponding to x .

Generate mask: Initializes the split points using the given x_h values. It takes the maximum and minimum values of x_h , calculates its differences and divide it by the number of split points. This gives the distance between each split points. The split points are stored in an array. An iteration from zero to the number of split points +1, we add the minimum value + iteration number* times the distance value. In the end it will contain both the minimal and maximum value, which will be ignored. They are only used for boundaries.

Get partitions: Based on the split points (split values), and the x_h , it partitions the given dataset which is ordered based on x_h .

Get loss: Calculates whether the expansion or shrinking results in lower error rate for the given two partitions

Predict: For the given attribute vector, it goes down the tree and returns the predicted value at the terminal node it reached.

The learning value is determined by the chosen x_h values. It is the difference between the maximum and minimum value of x_h divided by 200.

Parameters

In each implementation, I define a threshold value indicating the error rate at which further splitting stops because the accuracy achieved is satisfactory. I also specify a minimum sample size for a leaf node. This means that if a node's sample size is below this value, the branch will not continue, and that node is designated as a leaf. Additionally, I set a maximum depth; if this depth is exceeded, the decision tree stops growing, and nodes at that level are designated as leaves. I only set this depth limit when necessary, as recursion might otherwise exhaust the available memory, preventing the computer from completing the model. The learning value can be tweaked.

The hybrid tree switches to binary split if the amount of data at the given Node is below 30. Other restrictions could be used but testing showed that around this number of data multiway splits has some problem, and hybrid tree could offer a solution.

Measurements

I ran each algorithm five times on the same dataset and then took the average of the error rates and run times. Additionally, for algorithms using non-binary splitting, I tested several different

numbers of splits. The method I presented was designed to split the dataset non-binary, perform binary splits as well would perform poorly because of the locality.

The error was measured with mean squared error. To compare the different algorithms to each other the termination criteria was reaching below a certain number of data. So, if the stopping criteria is 50 it means that after splitting the data if the partition has less than 50 samples it stops splitting.

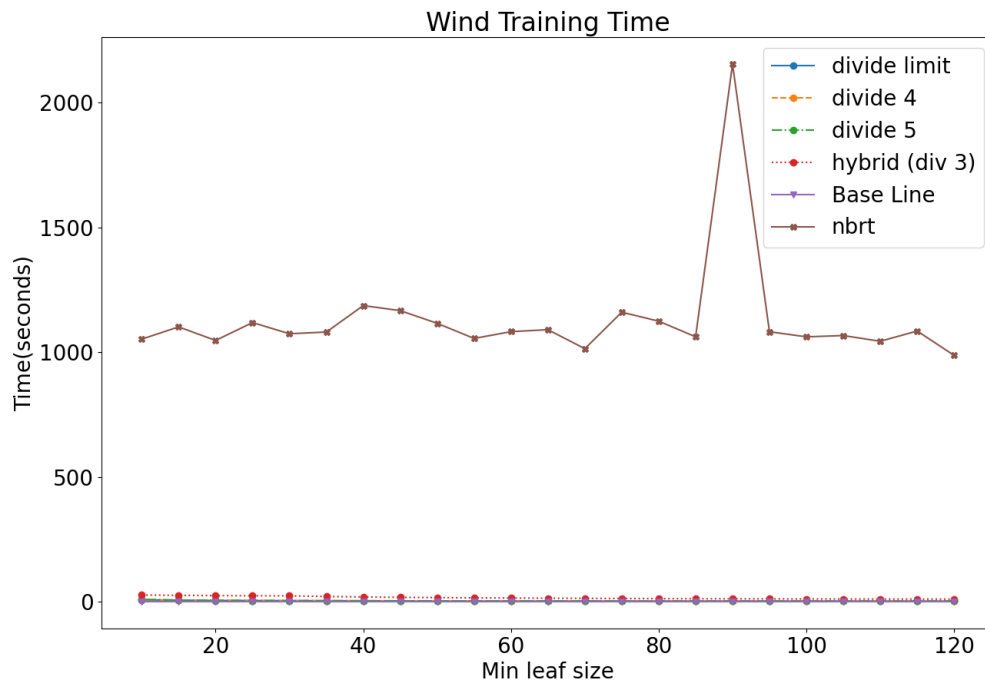
The time was measured in seconds with a precision of 15 decimal places.

The different algorithms tested

Model	Number of child nodes	Notation
Regression tree	2	Base Line
MSRTHC	3	Divide 3
MSRTHC	4	Divide 4Divi
MSRTHC	5	Divide 5
MSRTHC	3(2)	Hybrid
MSRTHAC	3	nbrt

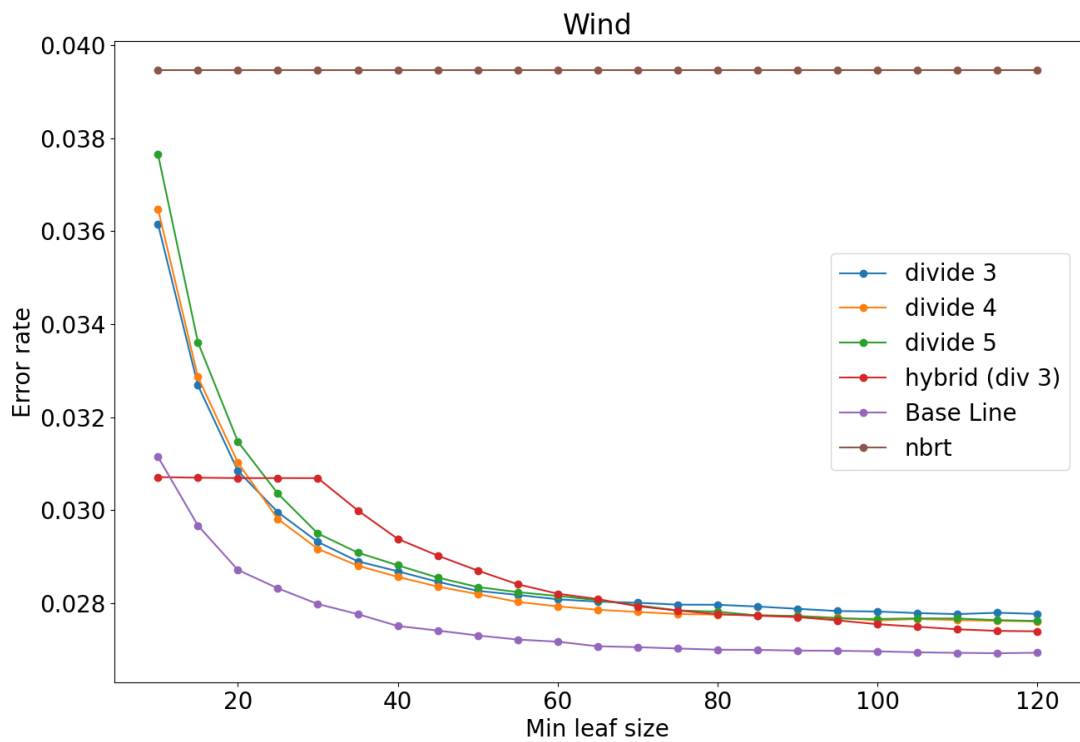
Results

Windmill



2. plot showing the elapsed time of training the tree on the wind dataset measured in seconds

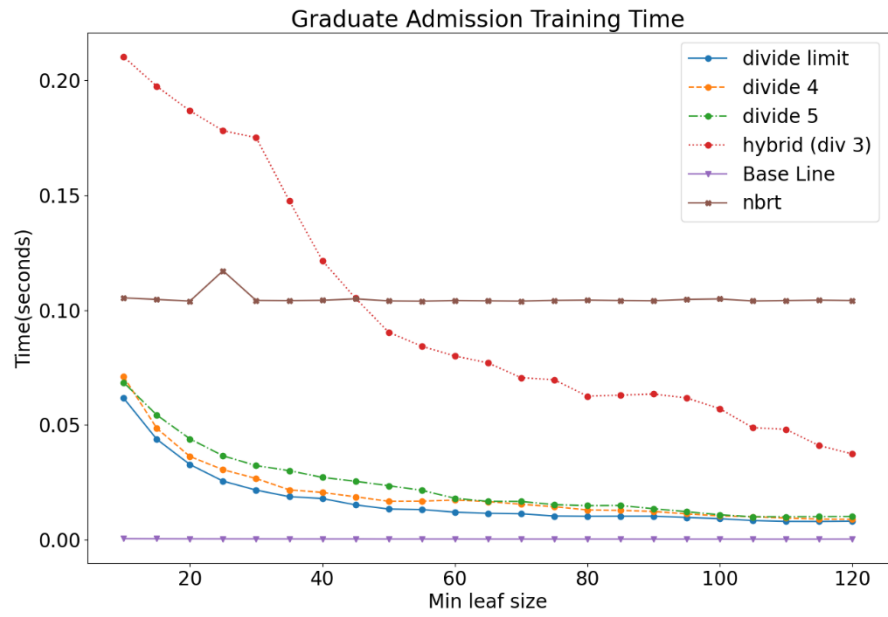
The windmill dataset is large with a lot of attributes. As we can see the base line performed the best for both the training time and error rate which is not a good start. Although it is not that far if we stop early when there is still a relatively large amount of data in a terminal node. Not to mention that using multiway split has a similar characteristic as CART. As the number of elements decrease at a given node the model performs worse. The interesting part is that in case of using a hybrid tree which utilizes both multiway splits and binary split it stabilizes the model and does not shoot off.



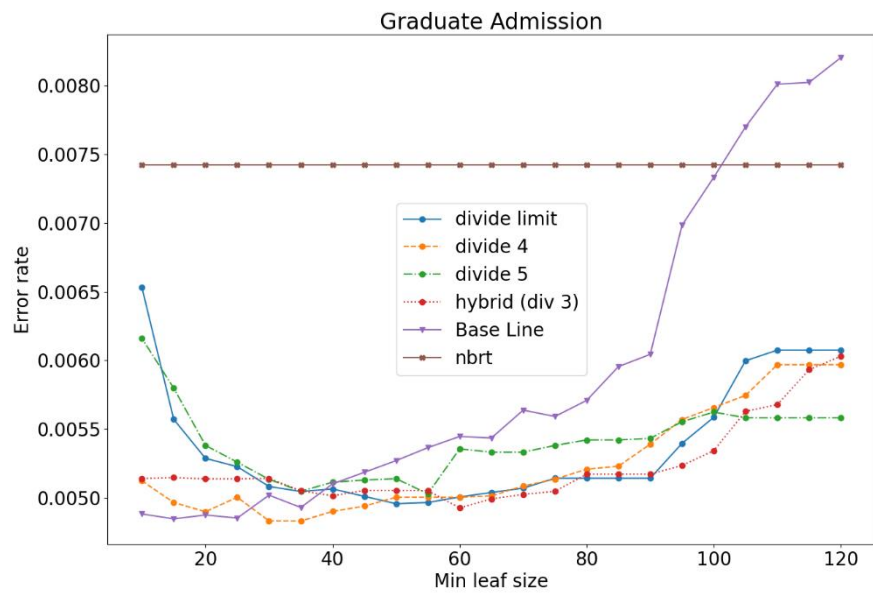
3. plot showing the error of the tree on the wind dataset measured with MSE

Graduate Admission

Compared to the previous dataset it shows much more promising results. With much less splits the push tree algorithm performs way better and as the number of data decrease it still improves. Although as the number of data points goes below 30 the model starts to perform worse and worse. But again, the hybrid tree stabilizes the model.

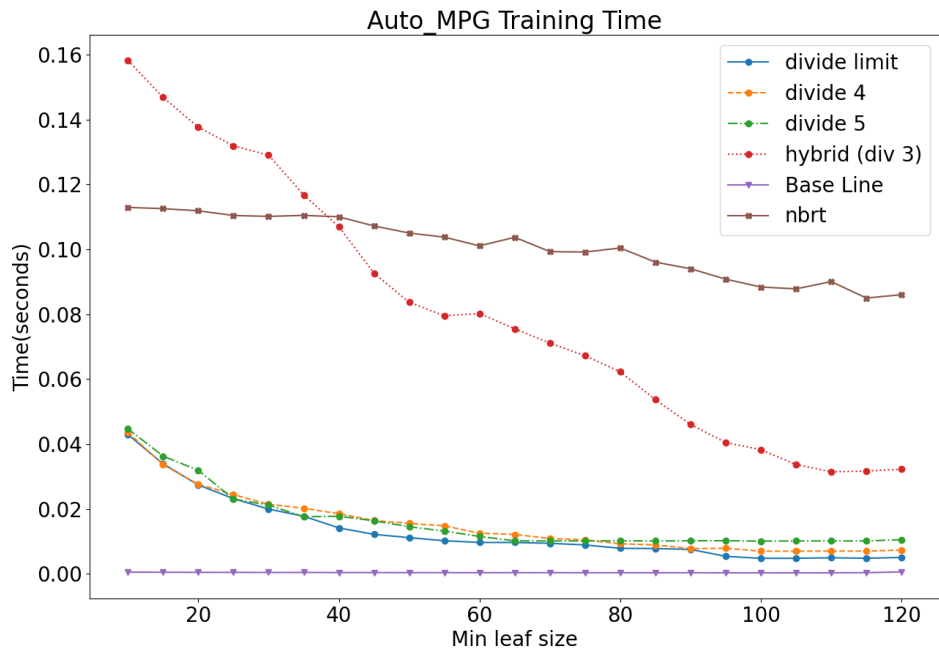


4. plot showing the elapsed time of training the tree on the Graduate Admission dataset measured in seconds

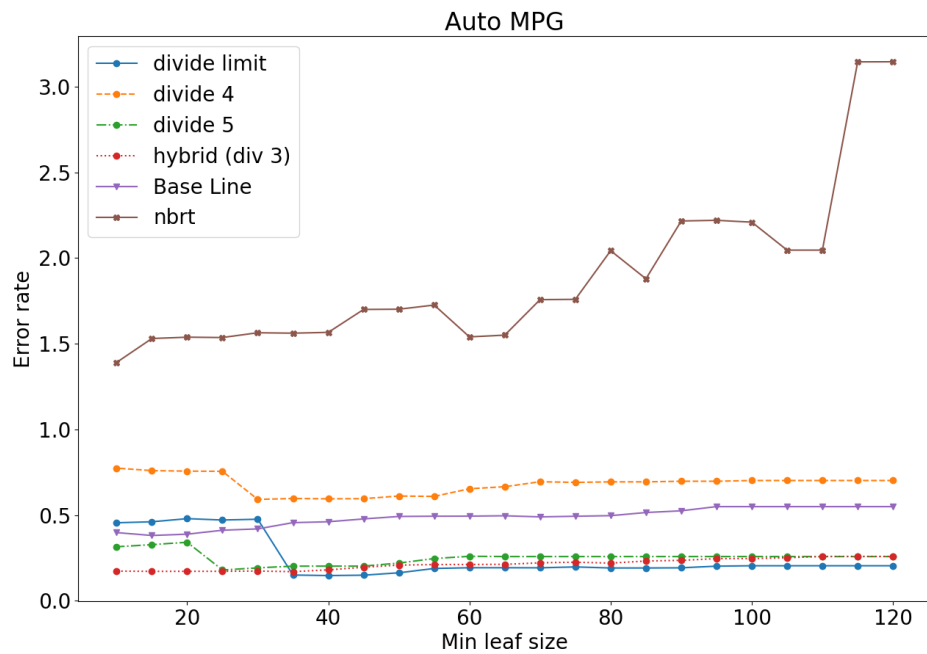


5. plot showing the error of the tree on the Graduate Admission dataset measured with MSE

Auto MPG



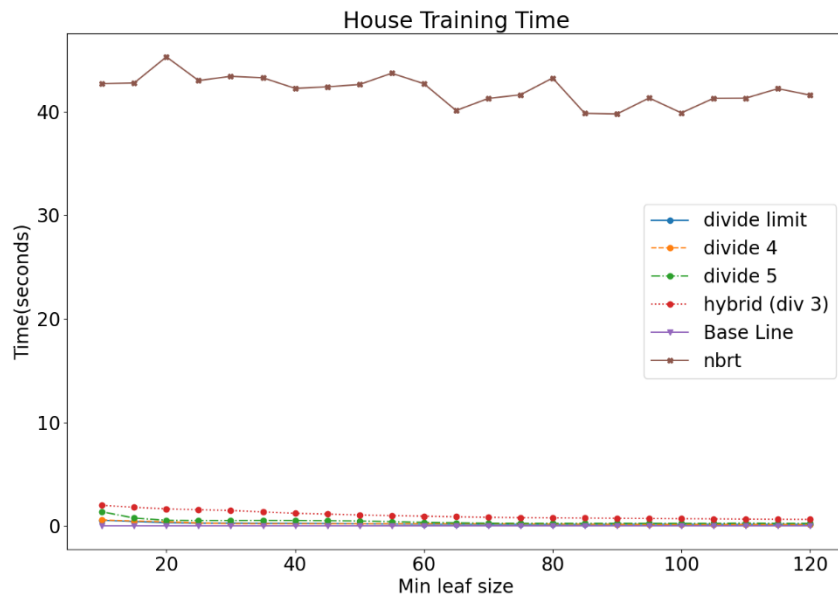
6. plot showing the elapsed time of training the tree on the Auto MPG dataset measured in seconds



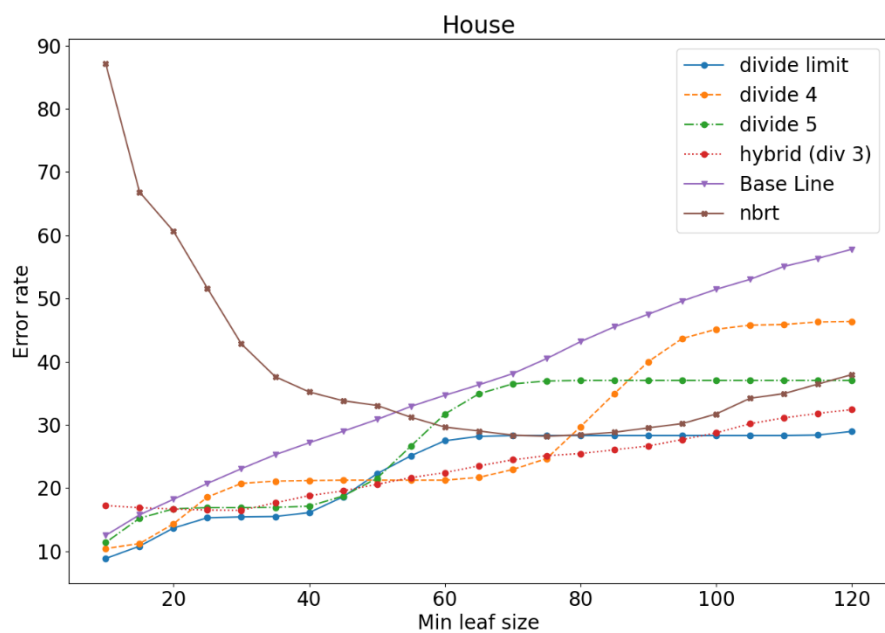
7. plot showing the error of the tree on the Auto MPG dataset measured with MSE

Multi way splits perform better than binary split. The usual problem with low number of data occurs again, but for example if we split the data into 3 or 5 parts it always performs better. Even if we would stop building the tree when they reach data below 120 it would still have higher accuracy than the baseline.

House



8. plot showing the elapsed time of training the tree on the House dataset measured in seconds



9. plot showing the error of the tree on the House dataset measured with MSE

At this dataset multiway split just performs better for each number of datapoints. Not to mention that in this case when the number of data points reach below 30, the error does not shoot off. This time using hybrid tree makes the model performance worse.

Times

The newly proposed algorithms runs slower every time. The difference is not large, but still the question remains, why is that. The answer is probably that the implementation itself is slower. Using another language like C might decrease the training time with better implementation.

Conclusion

In the field of artificial intelligence, decision trees have a long history and remain a method still in use today. As we can see, despite the first article on decision trees being published back in 1963 [7], this method is still relevant today, as indicated by the above researches, thus validating its continued use. Decision trees can be used for solving classification problems, for which the ID3 algorithm provides a solution. This algorithm was further improved, and thus C4.5 [4] does not have all the shortcomings of ID3. If we want to solve a regression problem, several algorithms can be used, including CART, which performs binary splits. Numerous methods are available for data analysis, such as linear regression and HAC. These methods, which have various implementations (e.g., from scikit-learn), can support the further development and optimization of other algorithms. The function-class-based decision tree, for example, aims to achieve more accurate results by using a linear model, which it successfully does [27]. The method I am presenting is a similar enhancement.

The purpose of the Multiway Splitting Regression Tree with Hill Climbing optimization method is to further optimize existing methods.

I perform the splits using a Hill-climbing based approach. As the tests show, the multiway splitting has higher accuracy until it reaches a low number of data. Naturally, a single method cannot perform best across all datasets. However, the test results clearly indicate that algorithms utilizing non-binary splitting consistently performed better or very similar. Therefore, it can be said that the non-binary tree method has successfully optimized existing solutions.

The key to multiway split is to find the right number of splits where the algorithm can take advantage in finding multiple local optimal split points compared to one global. The multiway splits with a small sample of data does not work well. The solution is to simply stop splitting earlier,

since we have seen that it can have better results with less splitting than binary trees. Another solution is to use the hybrid tree variation.

Notations

T-training data

D-partition

$\bar{x}$ - attribute vector

y – dependent variable

E_n -entropy

pr -probability

I_g -information gain

S_i -split info

G_r -gain ratio

s -split point

s' -candidate split point

MSE -.Mean Squared Error

h -splitting variable

L - loss function

$\hat{y}_i$ -predicted value

Literature

- [1] Kok, Joost N., et al. "Artificial intelligence: definition, trends, techniques, and cases." *Artificial intelligence* 1 (2009): 270-299.
- [2] Choubin, Bahram, et al. "An ensemble prediction of flood susceptibility using multivariate discriminant analysis, classification and regression trees, and support vector machines." *Science of the Total Environment* 651 (2019): 2087-2096.
- [3] AKINCI, T. Çetin, and H. Selçuk NOĞAY. "Application of decision tree methods for wind speed estimation." *European Journal of Technique (EJT)* 9.1 (2019): 74-83.
- [4] Quinlan, J. Ross. "Induction of decision trees." *Machine learning* 1 (1986): 81-106.
- [5] Andreas C. Muller and Sarah Guido (2016) *Introduction to Machine Learning with Python* O'Reilly Media,
- [6] Morgan, James N., and John A. Sonquist. "Problems in the analysis of survey data, and a proposal." *Journal of the American statistical association* 58.302 (1963): 415-434.
- [7] Loh, Wei-Yin. "Fifty years of classification and regression trees." *International Statistical Review* 82.3 (2014): 329-348.
- [8] Breiman, Leo, et al. "Cart." *Classification and regression trees* (1984).
- [9] Quinlan, J. Ross. *C4. 5: programs for machine learning*. Elsevier, 2014.
- [10] Romeo, Luca, et al. "Machine learning-based design support system for the prediction of heterogeneous machine parameters in industry 4.0." *Expert Systems with Applications* 140 (2020): 112869.
- [11] Zhang, Chen, et al. "Road condition recognition in self-driving cars based on classification and regression tree." *ICIC Express Letters, Part B: Applications* 10.12 (2019): 1115-1122.
- [12] Ktenioudaki, Anastasia, et al. "Blueberry supply chain: Critical steps impacting fruit quality and application of a boosted regression tree model to predict weight loss." *Postharvest Biology and Technology* 179 (2021): 111590.
- [13] http://project.mit.bme.hu/mi_almanach/books/aima/ch18s03

- [14] Peng, Wei, Juhua Chen, and Haiping Zhou. "An implementation of ID3-decision tree learning algorithm." From web. arch. usyd. edu. au/wpeng/DecisionTree2. pdf Retrieved date: May 13 (2009).
- [15]<https://medium.com/analytics-vidhya/decision-trees-for-classification-id3-machine-learning-6844f026bf1a>
- [16]<https://levelup.gitconnected.com/c4-5-decision-tree-explained-from-bottom-up-67468c1619a7>
- [17] Hssina, Badr, et al. "A comparative study of decision tree ID3 and C4. 5." International Journal of Advanced Computer Science and Applications 4.2 (2014): 13-19.
- [18]<https://medium.com/deeplearning-concepts-and-implementation/independent-and-dependent-variables-in-machine-learning-210b82f891db>
- [19] Sebastian Raschka (2015) Python Machine Learning Packt Publishing
- [20]<https://towardsdatascience.com/cart-classification-and-regression-trees-for-clean-but-powerful-models-cc89e60b7a85>
- [21]<https://medium.com/analytics-vidhya/regression-trees-decision-tree-for-regression-machine-learning-e4d7525d8047>
- [22]<https://www.ibm.com/topics/linear-regression>
- [23]<https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/>
- [24]<https://towardsdatascience.com/polynomial-regression-bbe8b9d97491>
- [25]<https://www.analyticsvidhya.com/blog/2021/10/understanding-polynomial-regression-model/>
- [26]<https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21>
- [27] Quinlan, John R. "Learning with continuous classes." 5th Australian joint conference on artificial intelligence. Vol. 92. 1992.
- [28]<https://towardsdatascience.com/machine-learning-algorithms-part-12-hierarchical-agglomerative-clustering-example-in-python-1e18e0075019>
- [29]<https://scikit-learn.org/stable/modules/tree.html>

- [30]<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>
- [31]<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html>
- [32]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
- [33]<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html>
- [34]<https://medium.com/analytics-vidhya/incremental-online-learning-9868861db880>
- [35]French, Robert M. "The Turing Test: the first 50 years." Trends in cognitive sciences 4.3 (2000): 115-122.
- [36] <https://www.analyticsvidhya.com/blog/2023/12/hill-climbing-algorithm/>
- [37]<https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/hill-climbing-algorithm-in-ai>
- [38]<https://images.javatpoint.com/tutorial/ai/images/hill-climbing-algorithm-in-ai.png>
- [39] The Elements of Statistical Learning: Data Mining, Inference, and Prediction Springer series in statistics, ISSN 0172-7397
- [40]Machine Learning A Probabilistic Perspective Kevin P. Murphy
- [41] <https://www.kaggle.com/datasets/mubashirrahim/wind-power-generation-data-forecasting>
- [42] <https://archive.ics.uci.edu/dataset/9/auto+mpg>
- [43] <https://www.kaggle.com/datasets/mohansacharya/graduate-admissions>
- [44]Nembináris Regressziós Fák vizsgálata: <https://ojs.unimiskolc.hu/index.php/psaie/article/view/2806>

Table of content

Introduction	3
Related results	4
Supervised Learning	4
Decision Trees	4
ID3 Algorithm	5
C4.5 Algorithm	7
Regression	8
Regression Tree	8
Linear Regression	8
Polynomial Regression	9
Overfitting and Underfitting	10
M5 Model Regression Tree	10
Hierarchical Agglomerative Clustering (HAC)	10
Hill Climbing	11
Existing Implementations	12
Decision Tree	12
HAC	12
Linear Regression/Polynomial Regression	12
Multiway splitting	13
Proposed Regression Tree Model	14
Multiway Split Regression Tree with HAC based optimization (MSRTHAC)	14
Multiway Splitting Regression Tree with Hill Climbing optimization (MSRTHC)	15
Hybrid tree (hybrid splitting)	17
Test results	17
Datasets	17
Windmill	17
Auto-Mpg Data	18
Graduate Admission	18
House	18
Implementation environment	20
Implementation	20
Parameters	21
Measurements	21
Results	23
Windmill	23
.....	24

Graduate Admission.....	24
Auto MPG	26
Times	28
Conclusion.....	28

Eredetiség nyilatkozat

"Alulírott XY, a Miskolci Egyetem ... Karának hallgatója büntetőjogi és fegyelmi felelősségem tudatában kijelentem és aláírással igazolom, hogy ezt a dolgozatot saját magam készítettem, a benne leírt vizsgálatokat – ha ezt külön nem jelzem – magam végeztem el, és az ismertett eredményeket magam értem el. Adatokat, információkat csak az irodalomjegyzékben felsorolt forrásokból használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Miskolc,2024. 11. 05.....

.....
a hallgató aláírása

A konzulens nyilatkozata

Alulírott dr. Kovács László a Miskolci Egyetem GÉIK Informatika Intézetének oktatója a Analysis of multi-way split regression trees című TDK dolgozatot beadásra alkalmasnak ítélem.

Miskolc,2024. 11. 05.....

.....
a konzulens aláírása