Chapter 1

Literature Review

This chapter presents the background of model trees and ensemble methods required to implement the model tree forest presented in the following chapters. There is no single conceptualised model tree, but many different implementations of the idea of a model tree. This is because any predictive machine learning model can be incorporated within the leaf nodes of a model tree. Model trees are an extension over decision trees. Hence, decision trees are first discussed in this chapter. Understanding the concept of the bias-variance tradeoff is necessary to motivate the use of decision trees in ensembles.

This chapter is structured as follows: Section 1.1 provides a background to decision trees and the induction thereof. The bias-variance tradeoff and how the two ensemble methods, *bagging* and *boosting*, address the tradeoff is discussed in Section 1.2. Section 1.3 presents the differences a model tree has over a decision tree as well as a general discussion on the performance of the M5 model tree. Two different model trees, HTL and GPMCC, which incorporate non-linear models at their leaf nodes are discussed in Section 1.4. Finally, two existing model tree ensembles, RMT and MTE, which incorporate M5 model trees are introduced in Section 1.5

1.1. Decision trees

In this section the user is introduced to the concept of decision trees in the context of machine learning. Section 1.1.1 discusses decision trees in the application of classification and regression. Section 1.1.2 introduces CART and the portrayal of a decision tree. Finally, Section 1.1.3 enumerates the induction process employed by CART to grow both regression and/or classification trees.

1.1.1. Introduction to decision trees

Decision trees are tree-like predictive models used in machine learning on both classification or regression problems. Decision trees are induced on labelled datasets to model the relationship between input and output variables. The output variable represents the target value which the decision tree is tasked with predicting. Classification and regression applications of decision trees are characterised by their output variable type:

(1) Is it an extension of a decision tree, or is it a type of decision tree?

(2) An understanding of ...

noun

yours is a verb.

(3) Sentence not good:

... background of ... the induction of ... ?

Rather background of decision trees and discusses how decision trees are induced.

(4) ... differences between model trees and ...

(5) Check the full sentence:

.... presents. ... a general discussion ?

Rather:

..... and discusses ...

(6) Use the Latex acronym package.

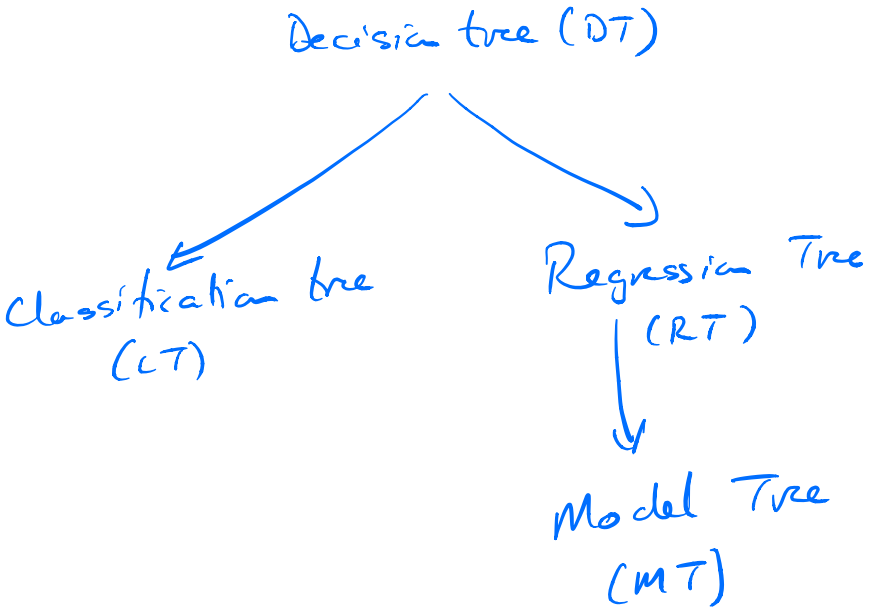
(7) This section introduces the concept of ...

(8) Does a different context exist for decision trees?

(9) ...as predictive models for classification and regression problems.

(10) Are you sure that this is what you mean? Provides a numbered list of the induction process? Does not make sense.

(11) Question on decision tree classification:



So CT is a DT

RT is a DT

MT is a RT ?

- classification trees are used to predict a categorical output; while
- regression trees predict a numerical output.

Categorical values are qualitative as they take on labelled values and numerical values are quantitative as they take on real values.

A decision tree organises data recursively through a hierarchical representation of tests on input features, giving it the appearance of a tree data structure. Through a series of tests, present in the nodes of the tree structure, the data is divided with branches representing the different outcomes of a test. Alternatively, decision trees are expressed as a collection of if-else statements to model decision boundaries in data. Decision trees were inspired by the *divide-and-conquer* paradigm, which entails recursively breaking an intricate problem down into smaller sub-problems which each can be solved more simply.

Before the conception of decision trees, many of the statistical techniques used for classification problems were developed with small datasets in mind [4]. Furthermore, the variables were all of the same type and the data was often assumed to be homogeneous i.e. the same relationships between these variables held throughout the entirety of the decision space. For that reason simple models that focused only on a few parameters to model the influence of a wide range of factors made up the majority of conceptualised models at the time [4]. The application of these simple models to larger datasets, with the assumption that the data retains its homogeneous structure, is flawed.

Datasets are not only subject to being large, but complex as well. The complexity of data is influenced by several factors; missing values, noise, outliers, high dimensionality, mixtures of data types, non-parametric distribution[1] and non-homogeneity[2]. "The curse of dimensionality" states the more features present, the higher the variance of the data points are. Hence the data is sparser and more spread out. The dimensionality of data can be reduced, however the accompanying drawbacks are unfavourable. The interpretability of data is significantly reduced with dimension reduction.

There was a need for a method in which the most important features of the data are accentuated, the background noise disregarded and the conclusions interpretable to the analyst [4], which brought rise to the conceptualisation of decision tree induction. The defining characteristic of decision trees is its ability to not only produce satisfactory results, but also give the user thoughtful information and insight into the data. For example, in classification problems, a decision tree helps uncover the predictive nature of a problem and helps the user better understand how specific features contribute to the outcome that is being observed [4].

[1] Data which does not fit a known distribution.
[2] Different relationships hold between variables within different parts of the decision space

(12) Are values categorical, or are variables/features categorical?

(13) a value takes on a value?

(14) Need to be consistent in term use: either feature or variable.

(15) This sentence says too much. Try using two, flowing sentences.

(16) ↝ means that there is no flow between these sentences.

(17) What is a factor?

(18) Is factors a correct term to use here?

(19) Not decision tree induction, but decision trees.

1.1.2. The CART methodology

One of the earliest documented work on the theory of decision tree induction was published in 1984 [4]. Breiman et al. was inspired by the deficiencies of existing tree-structured classifiers at the time to develop an improved methodology which provided a more flexible and accurate solution. Breiman labelled his proposed decision tree induction algorithm as CART (Classification and Regression Trees). CART grows a tree through a series of binary questions, which when answered sequentially, produces a solution to the problem at hand.

A simplistic classification tree is portrayed in Figure 1.1, capable of classifying an individual's gender based on two input features: weight and height. Nodes X_1 and X_3 each represents a test. Tests, also referred to as splits, comprise two parts: a feature on which the test is performed and a value indicating the decision boundary. Node X_1 is referred to as a *root* node because it is the starting node of the decision tree. Nodes X_2, X_4 and X_5 contain the label of the output class. These nodes are referred to as *leaf* nodes, as they terminate all possible *paths* (sequences of nodes) that can be followed in the tree from the root node to any leaf node.

Each node of the decision tree encapsulates a subset of the dataset; the data subset of a node is described by the tests that lead up to the node. When the data subsets each leaf node of the decision tree denotes is combined, the entirety of the instance space is accounted for [17]. The *size* of a decision tree refers to the number of nodes it consists of. Finally, *depth* is the number of nodes in the longest path the tree contains. Therefore a tree is regarded as being *shallow* if its depth is small.

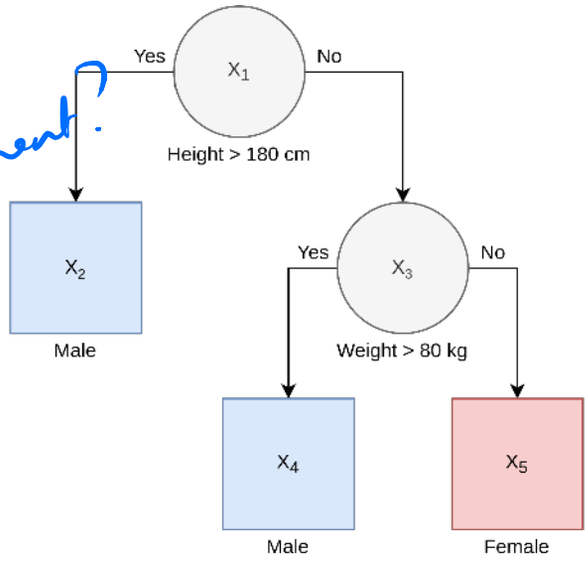


Figure 1.1: A hypothetical two-class tree-structured classifier.

(20) I will not say "described". Rather: the instances in a subset is covered by all of the conditions from the root node up to that node.

→ instances satisfy the conditions.

(21) Very verbose: The union of all leaf node subsets forms the entire training set.

1.1.3. CART's induction algorithm

CART induces a decision tree for classification, such as shown in Figure 1.1, from a set of labelled observations through three fundamental steps:

1. Determining the tests.
2. Evaluating stopping rules.
3. Assigning class labels to each leaf.

Starting at the root node, step 1 is repeated for each subsequent node until step 2 is satisfied for each path in the decision tree. To construct the ideal tree, each split is chosen to minimise the mixture of classes in the descendant nodes' subsets. A splitting function determines the feature as well as the value of a split; multiple splits are compared for a node by the splitting function. A metric often used to express the result of the splitting function is impurity; the larger the impurity, the higher the number of classes present in the subset is. CART produces binary splits from the value of a single input feature only. For that reason splits are always parallel to the decision space axes. CART allows input features to be both discrete and/or continuous.

At any node, the tree induction algorithm evaluates a total of $F \times N$ splits, where F is the number of features and N the number of observations in the dataset. A split is therefore proposed on every feature and the corresponding feature value for each observation in the dataset. The best performing split is selected as the one split which minimises the impurity function.

Once a node is reached where no significant decrease in impurity is produced, the node is declared a leaf node. This is an example stopping rule; stopping rules can also be explicit, such as fixing the depth of a tree. The class making up the majority of each leaf is assigned as the leaf's label. The decision tree growing sequence, i.e. the induction algorithm, is thus completed.

In the application of CART's induction algorithm to regression problems, little changes. The induction algorithm proceeds to partition the dataset into subsets via a series of tests on the input features. A different metric in place of impurity is incorporated; the expected reduction in either variance or absolute deviation for a proposed split is calculated. The splitting function of regression trees maximises the reduction in either variance or absolute deviation.

Changes are also present in the leaf nodes; instead of predicting which class an observation belongs to, the decision tree now predicts a numerical value associated with a class[3]. The numerical value of a leaf node is calculated as the sum of squares over the target values for all observations in the subset of data denoted by the path to the leaf node.

[3] As apposed to a classification tree which predicts a categorical value to define a class.

(22) i.e. the more heterogeneous the data subsets with respect to class distribution.

(23) Make sure that you do not overload the meaning of splits.

(24) Maximizes reduction in heterogeneity.

(25) Is it only C4.5 that induces the tree to overfit the training set?

(26) It is still an impurity measure, just a different impurity measure.

Fundamentally, the numerical value predicted by the regression tree is a dependent variable based on a multitude of independent variables[4]. The numerical values which represent the output in a regression tree are constant per leaf node. Subsequently, the regression tree models the relationship between the input and output variables through discrete piecewise approximation. Figure 1.2 illustrates the predicted constant values of a regression tree, induced on the values $y = sin(x)$ for $x \in [0, 2\pi]$, with the target value y being dependent on the input feature x.

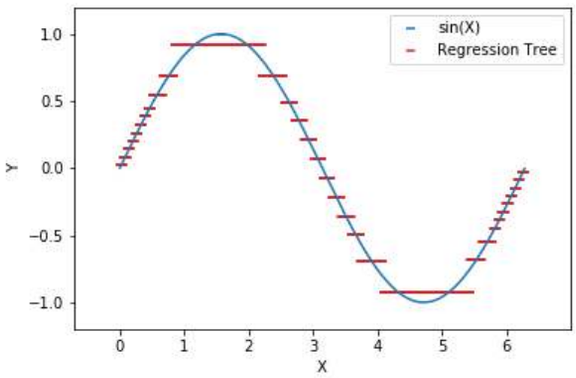


Figure 1.2: A regression tree's piecewise discrete approximation (red) of a continuous sinusoidal function (blue).

The advantages CART's induction algorithm had over statistical models at the time of its conception included [4]:

1. The induction algorithm handled both categorical and/or numerical input as well as output data. A decision tree can thus be grown from a dataset containing mixed feature types.

2. Once grown, the application of a decision tree on new incoming data is computationally efficient and interpretable.

3. Decision trees exploit the conditional relationships between variables present in non-homogeneous data to better model the observations.

4. The most beneficial information for identification is incorporated at each node, forming a subset of the dataset. Hence, automatic dimensionality reduction is effectively performed on the dataset.

5. CART proved to be quite robust when subjected to outliers and missclassified observations.

[4]The independent variables represent the input features and the dependent variable the target value

(27) A value is not a variable, but a variable is assigned a value.

(28) I do not see how this flows from the previous sentence.

1.2. The bias-variance tradeoff

This section introduces the dilemma of the bias-variance tradeoff which is present in all predictive machine learning models. This section also aims to discuss how ensemble learning can be an effective way to mitigate the problems associated with the bias-variance tradeoff. Section 1.2.1 starts by discussing the bias-variance tradeoff and how it causes a model to overfit or underfit. Next, Section 1.2.2 describes how the bias-variance tradeoff is specifically present in decision trees. Finally, Sections 1.2.3 and 1.2.4 discuss two popular ensemble methods and how each addresses the bias-variance tradeoff.

1.2.1. Overfitting and underfitting

The bias-variance dilemma stems from the degree of inaccuracy present in machine learning models. Both bias and variance are degrees of prediction error present in the model, negatively affecting its prediction accuracy. Ideally, a machine learning model should exhibit perfect prediction accuracy, but perfect prediction accuracy is infeasible because models are exposed to unseen and often noisy data during testing. Therefore, during training, the focus is on mitigating the prediction error, rather than trying to remove it altogether. To better mitigate the prediction error, its two components (bias and variance) must be examined.

Bias error is defined as the average difference in prediction a model makes from the target value; variance error is defined as the amount of variability or spread a model is subject to when predicting a certain data value. Figure 1.3 illustrates the predicted output of three models as follows: the first model exhibits a high variance error, the second a high bias error and finally, the third has a balance of low bias as well as low variance error. Each model is trained on the same set of observations, shown in blue.

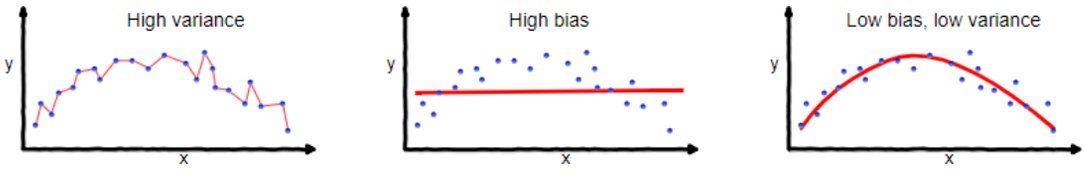


Figure 1.3: Three separate models' predicted output (red) and target output (blue) of a simple regression problem in a two-dimensional space. Reproduced from [20]

Models that are simple in their composition, such as the second model in Figure 1.3, are incapable of capturing any intricate patterns in the training data. These models produce an oversimplified interpretation of the data. Testing overly simple models on unseen data likely results in a high bias error [13]. A model with high bias error *underfits* the training data, i.e. it cannot capture the complex relationships within the data. Hence, a model that underfits will produce poor prediction accuracy.

(29) First need to discuss ensemble learning before you can do this.

(30) The tradeoff cannot cause overfitting / underfitting. Tradeoff can explain....

(31) I suggest that you provide background on ensemble learning in a separate subsection prior to section 1.2.

(32) stems from implies that it is due to...
BV-dilemma is as a result of the problem to balance model complexity with the desire to have accurate models.

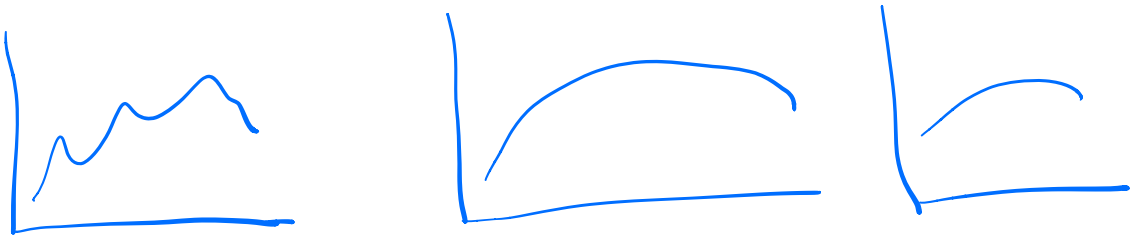
(33) How does examination of the two BV components help to mitigate prediction error?

(34) Only one value? "The" implies one.

(35) Here you mean a specific target value of the target feature.

(36) One-dimensional inputs.

(3) Use the subfigure environment



(a) caption (b) caption (c) caption.

Fig 1.3 . Caption . . .

In contrast, models that incorporate complex algorithms are more vulnerable to noise in the training data. Subsequently a large variance error is present and the model referred to as *overfitting* the training data. Although the prediction error on the training data of complex models may be low, such as the first model in Figure 1.3, the model still generalises poorly to unseen data [10]. The overly complex model fallaciously captures noisy patterns in the training data that are not associated with the desired output [13]. Hence, poor prediction accuracy is obtained, similar to models with large bias error.

Thus producing favourable prediction accuracy requires both bias and variance errors to be minimised. However, it is important to note that a model cannot simultaneously be less complex and more complex. Therefore, bias and variance, being respectively proportional to the complexity of a model, are error functions increasing in opposite directions. This tradeoff between bias and variance is portrayed in Figure 1.4. Models are designed with the strategy of minimising the tradeoff and finding the point of optimal balance between the two errors.

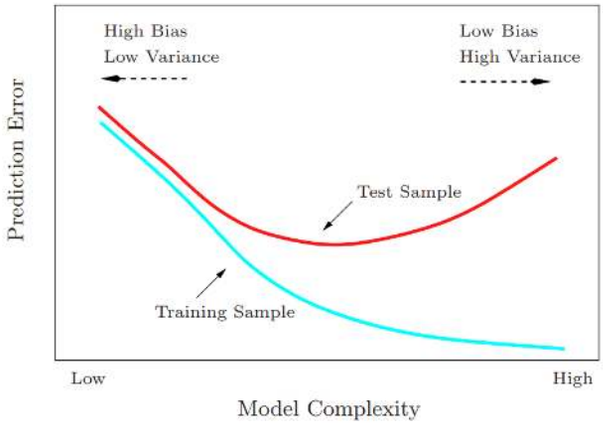


Figure 1.4: The change in a model's prediction error on the training set (red) and generalisation set (blue) as the complexity of the model is varied. Reproduced from [11].

1.2.2. In the context of decision trees

Decision trees are classified as supervised learning models, as they are trained with the task of mapping an observed output variable to multiple input variables. Unsupervised learning models are instead tasked with clustering unlabelled data into homogeneous regions. In the case of a supervised learning model, the criterion used in assessing its prediction accuracy is the ability of the model to generalise to unseen data [10].

When Breiman started his initial work on decision trees, it was clear that arguably the biggest shortcoming was a product of the high variance present in decision trees [4]. Trees would often deliver satisfactory results on training sets, but generalised poorly on test sets. During induction, classification trees continuously optimise the classification boundaries by adding additional decision nodes. Hence, decision trees are prone to capturing noise in

(38) Make sure that when you refer to prediction error that the reader knows to which you refer: training error or generalization.

(39) To what do you refer? Also, the noise is actually associated with the target.

(40) No similar. High bias results in poor training and generalization error.
High variance results in only poor generalization error.

(41) ... trained with the task... Does not make sense.

(42) The other way around:

$$f: \mathbb{R}^n \to \mathbb{R}^m$$

input space to output space.

(43) Be careful. Not all unsupervised models do clustering. SOM, e.g. does not do clustering, but nonlinear topological mapping.

(44) One looks at both training error and generalization, of which the latter is the most important.

(45) Note: Breiman et al. Covered throughout.

the training set. This meant that trees were prone to overfitting and require something to remedy this.

The answer Breiman put forth was a fundamental shift in focus. Instead of adjusting the stopping criterion to combat overfitting, the tree was grown with lenient stopping rules and once fully grown, selectively *pruned*. Pruning works as follows: starting at the leaf nodes and following the paths upwards into the tree, nodes are evaluated through cross-validation and the subtree below removed if warranted. Each node along the path is temporarily pruned to a leaf node and this temporary pruned tree's performance on unseen data compared to that of the original unpruned tree. If the tree's prediction error is found to have decreased, the node remains pruned. Otherwise, the tree is reverted to its state before the concerning node was pruned. Pruning proved to lower the variance in classification trees and significantly improve its performance [4].

As for regression trees, the tradeoff is more intricate. Because of its tree structure, regression trees are prone to overfitting. However, the constant output value at a leaf node can be argued to underfit the subset of data denoted by the path to the leaf node. Figure 1.2 shows how poorly constant values fit the shape of a sinusoidal wave. This underfitting is only applicable to the subset of data though. If the regression tree is evaluated as a whole, a high variance error remains observed. This is due to the splits being highly sensitive to changes in data, especially in the case of noisy data. Therefore regression trees are pruned similarly to classification trees to reduce the variance error.

An alternative method to pruning with the intent of minimising the variance of decision trees is the use of ensemble learning. The premise of ensemble learning is to develop multiple models in unison, which together produces a better result than the result an individual model is capable of producing.

1.2.3. Bagging

One of the simplest tree ensemble methods is bootstrap aggregation, referred to as *bagging* [5]. Several decision trees are induced in parallel, each on a subset of the training data. A subset is determined through random selection. For regression applied bagging, the final prediction of the ensemble is derived through averaging the predictions of each tree within the ensemble. In the case of classification, the majority vote is taken instead of the average predicted value.

The randomness in the data subsets on which the decision trees are induced contributes to minimising the variance of the collective model. The total variation in the prediction a bagged model produces for a given set of observations[5] is decomposed into two terms and

[5]The set of observations is assumed to be a set of independent and identically distributed random variables

(46) Also, less complex rule sets are extracted.

(47) Not the tree structure, but the condition used to stop splitting. Splitting is stopped if deviation over target value is below a threshold. If threshold is too small

$\Rightarrow$ overfit

If too large $\Rightarrow$ underfit.

(48) But, we work with fixed data sets.

(49) Biggest problem is outliers.

(50) You can only say this once ensemble learning has been discussed.

(51) So, how does this address pruning?

(52) So, how is bias-variance balanced?

(53) You first need to define what an ensemble is.

(54) Sampling with replacement? Without replacement?

expressed as [13]

$$Var(\boldsymbol{x}) = \rho(\boldsymbol{x})\sigma^2(\boldsymbol{x}) + \frac{(1-\rho(\boldsymbol{x}))\sigma^2(\boldsymbol{x})}{N} \tag{1.1}$$

where ρ is the sampling correlation between the prediction of any two decision trees in the ensemble, σ is the sampling variance of any single, randomly drawn, decision tree and N is the number of decision trees in the ensemble. Finally, $\boldsymbol{x}$ represents a set of observations. Due to the injection of randomness in the tree induction algorithm, it can be assumed that trees are divergent from one another, i.e. $\rho(\boldsymbol{x}) < 1$. The stronger random effects are the more ρ tends to 0, reducing Equation 1.1 to only $\frac{\sigma^2(\boldsymbol{x})}{N}$. It is evident that as the number of decision trees in the ensemble is increased, the total variance decreases further. The variance of the ensemble is thus strictly less than the variance of an individual tree. Furthermore, combining randomised models does not affect the bias error [11]. Therefore, decision trees are ideal for bagging as they exhibit a very low bias. Bagging decision trees produces a model that is both low in bias and variance, improving its accuracy significantly over that of a single tree.

Building on the success achieved by bagging, Breiman conceptualised the *random forest* method as an extension over bagging [6]. In a random forest, each tree is once again trained on a randomly selected subset of the data. The difference random forest incorporates is a change to the induction algorithm of the individual trees that make up the ensemble to strengthen random effects. A subset of input features is randomly selected at each node of the tree to determine the best split for that node. Hence, the splitting function only evaluates splits on the randomly selected features. The decision trees within the ensemble are further decorrelated as a result, decreasing the collective model's variance. Random forest has the benefit of performing well on higher dimensionality data, due to the dimension reduction-like induction algorithm [6]. Note that individual trees are left unpruned in random forest. Therefore, each decision tree within the ensemble is *fully grown*, retaining the low bias of decision trees. Random forests were shown to outperform many competing classifiers whilst being robust to overfitting [6].

1.2.4. Boosting

In contrast to bagging is the *boosting* method. Boosting is fundamentally different from bagging; boosting employs a sequential learning strategy as opposed to the parallel induction of bagging. The models which make up a boosted ensemble are developed with a higher bias than variance in mind and referred to as weak learners. A weak learner's error rate is only slightly better than a random guess [11]. In the context of a decision tree, an example strategy of inducing a high bias decision tree is a growing a shallow tree. Weak learners are incapable of individually modelling complex observed relationships. However, as a collective model, weak learners form a strong learner capable of adequately modelling more complicated patterns in data.

(55) So, $\mathbf{x}$ is a data subset?

Conventions are:

sets in uppercase, not bold

matrices in uppercase, bold

vectors in lowercase, bold

scalars in lowercase, not bold.

(56) A reference that shows this?

(57) What makes a random effect stronger?

(58) Individually, or as an ensemble?

(59) Can you add an illustration to illustrate the differences between bagging and boosting?

(60) What is an error rate vs an error? Also, you refer here to individual weak learners.

(61) Decision stumps are used.

Boosting sequentially induces a weak learner on repeatedly modified versions of the data. Data is modified through weights that are assigned to each training observation. Weights represent the probability of an observation being chosen as part of a subset on which a weak learner is induced [14]. After each additional weak learner is added to the ensemble, larger weights are assigned to the observations which have the highest contribution to the prediction error. Each successive weak learner thereby prioritises improving the prediction error of the larger weighted observations. Through this sequential optimisation, the collective model's bias is reduced compared to that of a single weak learner. Once induction of the entire ensemble is complete, weights are assigned to the outputs of each weak learner based on their prediction error.

Bagging and boosting has now been shown to have similarities, such as subsampling of the dataset but these methods have fundamentally different approaches to addressing the bias-variance tradeoff. To conclude, boosting reduces bias with an already low variance whilst bagging reduces variance with an already low bias.

1.3. Model trees

This section discusses how model trees are an extension over decision trees conceptualised to produce increased performance on regression problems. The section is outlined as follows. Section 1.3.2 elaborates on the need that brought rise to the conceptualisation of model trees. The induction algorithm of the M5 model tree is described in Section 1.3.2. Section 1.3.3 discusses the first published performance results of the M5 model tree. The linear models which the M5' model tree incorporates is described in Section 1.3.4. Finally, Section 1.3.5 discusses the drawbacks associated with model trees that incorporate strictly linear models.

1.3.1. The limited capability of regression trees

Breiman's early work on improving the robustness of decision trees showed great success concerning the application of decision trees to classification problems [4]. However, Breiman's focus on classification trees [5, 6] meant that regression applications of decision trees still left much to be desired. A big drawback of regression trees remained the discontinuous output it produced, as shown in Figure 1.2. This meant that regression trees, developed through the CART methodology, had limited capability of adequately predicting the target value for continuous target features.

Before the conception of model trees, problems where the predicted value took on a continuous numeric value favoured the use of learning techniques capable of producing continuous numerical predictions. Linear regression or neural networks are two example models capable of producing a continuous numerical output. However, no technique comes

(62) You have to reorganize the flow of the sections.
At this point you have the following flow:

- individual decision trees
- bias variance
- ensembles
- bias variance
- mode trees
- ensembles (model trees)

Rather

- decision trees
- model trees
- ensembles → classification
  → regression
  → model trees
- bias-variance
  → define & discuss
  → for classification trees
  → for ensembles.

without its drawbacks and both linear regression and neural networks are no exception to this. Linear regression has limited capability because it imposes a linear relationship on data, whilst neural networks do not provide the user with an insight into the relationship of the data, due to its problem of opacity [24].

There was still a need for a learning model capable of performing non-linear regression and providing its user with insights into the relationship among descriptive features that results in specific predictions. Model trees were first developed by Quinlan in 1992 as an extension over Breiman's regression trees with the fundamental difference that Quinlan's M5 model tree's leaf nodes are not limited to having a constant prediction value [18]. Model tree leaves can incorporate any multivariate model to better capture the patterns present in the training data. These multivariate models are used to predict a continuous target feature.

Figure 1.5 illustrates how a model tree, that comprises linear models, approximates the continuous sinusoidal function $y = sin(x)$ for $x \in [0, 2\pi]$, as opposed to the regression tree in Figure 1.2. The use of linear models instead of constant values at the leaves allows a model tree to predict a numeric target value without producing discontinuities and thus better approximate the non-linear function, $y = sin(x)$. The linear models also allows profiling, i.e. describing the conditions on the input features that will result in a specific linear trend.

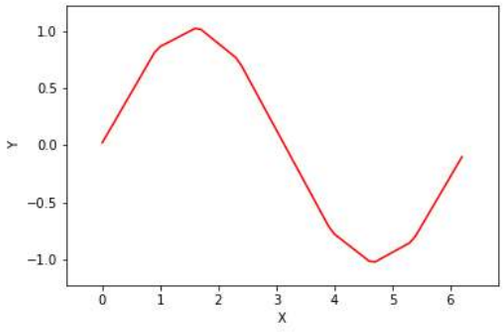


Figure 1.5: The continuous piecewise linear approximation of a sinusoidal wave using a model tree.

1.3.2. M5 model tree induction

The induction of an M5 model tree is quite similar to that of CART, as M5 model tree induction also employs a greedy approach[6]. The first difference between the two methodologies is the splitting criterion used to evaluate a proposed split. Instead of selecting the split that maximises the reduction in variance or absolute deviation, the M5

[6]A greedy approach is a heuristic approach that makes choices resulting in local optima, with the intentions of reaching an global optimum.

(63) Note that approaches do exist to extract rules from neural networks.

induction algorithm selects the split which maximises an expected error reduction at a node, defined by the following formula [18]

$$\Delta error = sd(T) - \sum_i \frac{|T_i|}{|T|} \times sd(T_i) \tag{1.2}$$

where T represents the collection of training samples for the data encapsulated by the node, and $sd(T)$ is the standard deviation of the target values in T. Every potential split is evaluated through the subset of samples it produces. T_i denotes the subset of samples that belong to outcome i of a split and $std(T_i)$ the standard deviation of the target values in T_i. Therefore, the M5 induction algorithm chooses splits that are locally optimal for each node, similar to the induction of regression trees. After the model tree is grown, linear models are constructed for each node. The reason why linear models are constructed for non-leaf in addition to leaf nodes is to accommodate for *pruning*. The linear model of a leaf node's parent node is required to calculate the alternate performance of the tree if that leaf node were to be pruned away.

A multivariate linear regression model, at any given node, is constructed with only the features defined within the splits of that node and its subtree nodes. This ensures that the performance of a non-leaf node linear model is compared to the performance of the subtree on a level playing field during the pruning process [18]. Once the linear model is constructed, each parameter of the linear model is evaluated for simplification of the linear model. If the exclusion of a variable minimises the error of the linear model, it is removed. This means that all of the variables of a linear model can be removed, leaving only a constant behind.

Once a simplified linear model is constructed for each node, the tree is pruned by examining all non-leaf nodes as if they were leaf nodes, starting at the bottom of the tree. If the error of the final model is reduced by regarding the specified node as a leaf node, the subtree, encapsulated by the node, is withdrawn from the model and the specified node pruned to a leaf node. The pruning step together with the simplification of the linear models at the nodes of the model tree helps to reduce the complexity of the final model, subsequently reducing its variance.

Smoothing is applied when the model is tasked with predicting a test sample. Thereby, discontinuities are prevented from forming between the linear models of adjacent leaf nodes. Smoothing adjusts the predicted values of the model tree as follows: starting with at the leaf node, the predicted value is passed on along the path to the root node and adjusted at each node such that it better resembles each predicted value produced by the linear model of a given node on the path. The formula used to adjust the predicted value, y_i, at the i^{th} node along the path to the root is

$$y_i = \frac{ny_{i-1} + ky}{n + k} \tag{1.3}$$

(64) Make sure that

1. each symbol has one meaning only
2. each meaning is indicated with the same, one, unique symbol.

where n is the number of training samples at the previous node along the path, y_{i-1} is the adjusted prediction from the previous node, y is the predicted value of node i, and k is the specified numeric smoothing constant. The value of y_i produced by the root node at the end of the path is the predicted value of the model tree given the test sample.

1.3.3. Early M5 model tree performance

Quinlan compared the M5 model tree to MARS (multivariate adaptive regression spline), which was a popular regression model that proved effective on non-linear data at the time of the M5 model tree's conception [18]. MARS is similar to M5 as it is also non-parametric[7] and utilises piecewise linear approximation. M5 and MARS produced similar accuracy results, but what set M5 apart from MARS was the computational requirement. The computational requirements of MARS grew aggressively with an increase in dimensionality, severely limiting its applicability. M5 was able to handle tasks with up to hundreds of input features, whereas MARS struggled past no more than twenty [18]. The results Quinlan published proved that model trees are a viable choice for solving regression problems. Model trees also compare better to alternative regression techniques as opposed to regression trees which struggle to compete against linear regression models or neural networks.

1.3.4. The M5' model tree

Quinlan's work on the M5 model tree was not readily available and some design decisions, such as how missing values are handled, were not addressed. This prompted Wang and Witten to refine the induction steps to create the M5' model tree. One important aspect of the M5' model tree is that it specifies the linear model implemented in a node. The linear model, named the $(k+1)$-parameter model, is denoted by the formula [24]

$$w_0 + w_1 x_1 + w_2 x_2 + ... + w_k x_k \tag{1.4}$$

with x_k representing the k^{th} input feature, w_k the respective weight of that input feature in the linear model, and w_0 the bias term. Fundamentally, Equation (1.4) is a linear polynomial with weights that represents its coefficients. Through experimentation Wang and Witten deemed the simplification step of M5 (the simplification step of M5 removes terms from the linear model) compromising to the size of the model tree. Sometimes much smaller trees were obtained when all features were left in the linear models. Therefore, the simplification step was omitted from the induction algorithm. The M5' model tree was shown to perform better than the original M5 tree on the standard datasets for which results where available [24].

[7] A non-parametric model has no preconceived notion regarding the number of parameters it is tasked with learning nor the distribution that the data follows.

(65) Should discuss this when you discuss MS.

1.3.5. Shortcomings of model trees comprising linear models

The M5 model has been shown to perform inadequately in comparison to other regression techniques on datasets that contain noisy patterns and highly non-linear relationships between its features [1]. The M5 model tree's susceptibility to initially overfitting noisy patterns can be attributed to the already high variance exhibited by tree based models, together with the model tree's increased complexity over the regression tree. To combat this, the model tree is pruned during induction. However, excessively post-pruning[8] a decision tree, with the intent of increasing its resilience to overfitting noisy patterns, limits the complexity of the model and in turn its ability to capture highly non-linear patterns. Within the tree multiple leaf nodes, each with linear models, are pruned away and consequently these linear models simplified into one. This is referred to as over-pruning and has been shown to have negative affects on a model tree's performance [1].

A study published in 2016 [12] hypothesised that the reason for the M5 model tree's inferior results on highly non-linear data is its piecewise linear functions' limited capability to fit the training data. In this study, only six quantitative water quality parameters were used in predicting the monthly chemical oxygen demand of a river. The relationship between these parameters is highly non-linear. A MARS model was able to achieve, on average, an accuracy of 19.1% better than the competing M5 model tree. It is important to note that the M5 model trees were designed with large datasets in mind, making it the preferred choice for problems with a large number of input parameters [18].

Quinlan recommended researching the application of non-linear models at the leaf nodes [18] to improve the ability of a model tree to adequately capture highly non-linear patterns present in complex datasets. This would also allow for a model tree to be grown smaller, as a single non-linear model can approximate the same function which demands multiple linear models to approximate. The number of splits of the training data is reduced to produce a smaller tree, and should therefore decrease variance in the model making it less sensitive to noise and less prone to overfitting.

Careful consideration should be given to the increase in computational cost that comes with the implementation of non-linear models. A model has to justify its increased computation time with a clear and substantial improvement in its accuracy or even interpretability. Ockham's razor expresses the fallacy in expecting that the increase in complexity of a model will guarantee an improvement in its performance. More often than not a simpler approach is the favoured solution to a problem [3]. These factors are hypothesised as contributing to the lack of abundant research on model trees that incorporate non-linear models.

[8] Post-pruning refers to pruning of the decision tree after it has been grown to overfit on the data, as opposed to stopping the growth of the tree prematurely to effectively prune it.

(66) Between the input features, or do you mean between input features and target features?

(67) Discuss the impact of the models on bias-variance.

(68) Pruning during induction is not post-pruning.

(69) that underfits the corresponding mapping?

(70) This is also something to avoid. Rewrite, e.g.

--- reason for the inferior results of --- limited capability of ---

1.4. Non-linear solutions

This section discusses two existing model trees that incorporate non-linear models within their leaf nodes. For the model tree forest that will later be proposed in this paper it is important to first evaluate the performance of existing model trees that comprise non-linear models. Inducing model tree comprising non-linear models is already computationally expensive, and growing an entire ensemble may be inapproachable. In Section 1.4.1 the non-linear model present in PLT is broken down and the performance thereof examined. Next, Section 1.4.2 discusses GPMCC, a model tree which is induced via a genetic programming approach.

1.4.1. Kernel regression

Torgo developed the hybrid regression tree (HTL) that, amongst other proposed models, incorporated kernel regression at its leaf nodes [23]. A kernel function empowers a linear model to interpret the relationships between the observations as non-linear by mapping the observations to a higher-dimensional feature space. The kernel regression that HTL incorporates is known as a lazy learner, because the system only generalizes the training data once a prediction is made[9].

The HTL structure is induced using the CART methodology of selecting splits that minimizes the mean square error (MSE)

$$MSE = \frac{1}{N}\sum_{i}^{N}(y_i - y_i')^2 \tag{1.5}$$

where N is the number of observations over which the MSE is computed, y_i is the actual target value of the i^{th} observation, and y_i' is the predicted value for the observation. The predicted value for each observation is chosen as the average target value in the subtree where the split is proposed[10]. A kernel regression model is thereafter used to calculate o_i when making predictions. This gives rise to the hybrid nature of HTL. Torgo states that using the kernel regression's calculations to determine the MSE for each proposed split within the tree structure is computationally too expensive [23], and therefore opts to use the CART methodology to induce the tree structure.

The kernel regression model comprises a k-nearest neighbours model and a Gaussian kernel function. The kernel regression model calculates predictions using only the training samples most similar to a given query. The similarity is based on a distance metric between two observations. The Gaussian kernel function increases the influence based on the

[9] For this reason, lazy learners are also a popular choice for models where the training data is regularly updated.

[10] This means that effectively the deviation is being calculated; it is showcased this way to emphasise the potential of interchanging o_i with the kernel regression prediction.

distance metric, neighbours have on the prediction the nearer they are to the query in the feature space. Given the query point $\boldsymbol{q}$, the prediction is calculated using the following function [23]:

$$o(\boldsymbol{q}) = \frac{1}{SK} \sum_{i}^{k} y_i \times K\left(\frac{D(\boldsymbol{x_i}, \boldsymbol{q})}{h}\right) \tag{1.6}$$

with

$$SK = \sum_{i}^{k} K\left(\frac{D(\boldsymbol{x_i}, \boldsymbol{q})}{h}\right) \tag{1.7}$$

and

$$K(d) = e^{-d^2} \tag{1.8}$$

where $D(\cdot)$ is a distance function between two vectors, h is the bandwidth parameter, and $\boldsymbol{x}_i$ is a vector of input variables with target value y_i for the i^{th} observation of k nearest neighbours. Only the training samples that fall into the same leaf node as $\boldsymbol{q}$ is evaluated to be one of its nearest neighbours. HTL is computationally more expensive than other tree models, as the nearest neighbours have to be re-evaluated for each query. Equation (1.6) portrays how the computational complexity grows as the value of k increases due to the added terms in both the kernel function and the encapsulated distance function. For experimentation, Torgo chose $k = 3$ without specifying the reasoning behind that choice [23].

The performance of HTL was evaluated on eleven different benchmarking datasets. Torgo concluded that, compared to linear models, the use of kernel regression models at the HTL leaf nodes resulted in significantly better performance for most cases and never significantly worse[11]. Although Torgo mentioned the M5 model tree incorporating similar linear models to the linear HTL variant, Torgo did not compare the performance of HTL directly to that of M5 [23].

HTL was further improved on by imposing the kernel regression model, described by Equation (1.6), on the linear model in Equation (1.4). The resulting model now incorporates the weight vector of Equation (1.4), $\boldsymbol{w}$, and is described by the equation:

$$o(\boldsymbol{q}) = \boldsymbol{wq} - \frac{1}{SK} \sum_{i}^{k} \epsilon_i \times K\left(\frac{D(\boldsymbol{x_i}, \boldsymbol{q})}{h}\right) \tag{1.9}$$

where e_i is the error of the linear model calculated as $\epsilon_i = \boldsymbol{wx_i} - y_i$. The vector $\boldsymbol{w}$ is a set of weights that represent the coefficients of a linear polynomial, equivalently expressed in Equation (1.4). For a set of M data points $\{(\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_m}, y_m)\}$, the weights of $\boldsymbol{w}$ are

[11] With the exception of one dataset which Torgo claimed as being biased towards linear models.

calculated by minimising a least squares error criterion:

$$\epsilon = \sum_{i=1}^{m} [y_i - \boldsymbol{w}\boldsymbol{x}_i]^2 \tag{1.10}$$

This continuation of HTL was named the partial linear tree (PLT). PLT was designed with the goal of maintaining the accuracy of linear models while improving its comprehensibility of non-linear patterns [22]. PLT was compared to MARS as well as a commercial version of M5, Cubist which incorporates only linear models, on 12 separate benchmarking datasets. PLT produced highly competitive results, though Torgo described the use of PLT as being computationally heavy for the same reasons HTL is considered computationally expensive [22].

Torgo recommended further research be done on the use of an alternative splitting criterion when inducing the tree structure as PLT did not change the procedure HTL follows, which is the standard CART methodology of selecting splits that minimise deviation. A criterion which incorporates the models used at the leaf nodes, such as the one used in M5, can improve the fit of the models in leaf nodes to be more accurate. However, this also increases the computation time and therefore demands careful consideration [22].

1.4.2. Higher-order polynomial regression

Another approach to modelling non-linear data is through the use of higher-order polynomial expressions, rather than piecewise linear polynomials. However, estimating the structure of a multivariate polynomial as well as its parameters is a difficult task. The formula of a multivariate polynomial function incorporating all possible terms can be expressed as [16]:

$$f(\boldsymbol{x}) = \sum_{\tau=0,\sum_{j=1}^{m}\lambda_j=\tau}^{n} \left(w_{(\lambda_1,\lambda_2,\ldots,\lambda_m)} \prod_{q=1}^{m} x_q^{\lambda_q} \right) \tag{1.11}$$

where m is the dimensionality of the feature space, n the degree of the polynomial order and $w_{(\lambda_1,\lambda_2,\ldots,\lambda_m)}$ the polynomial coefficients, i.e. the weights when used as a regression model. For a 3^{rd} order polynomial describing a 10-dimensional feature space, there are 286 terms in total. Terms can be dropped to change the characteristics of the polynomial function, meaning there are 2^{286} different possible combinations of those terms. Even with the coefficients already estimated, iteratively testing all 2^{286} possible functions on a set of observations is already computationally demanding. Genetic algorithms, however, can be implemented to perform these tasks that are otherwise considered computationally too expensive [16].

A genetic algorithm is a heuristic search method that models aspects of natural selection mechanisms to evolve an optimal solution from multiple candidates [8]. “Survival

of the fittest" is simulated across numerous generations, each generation having multiple candidate solutions [19]. The fitness function of a genetic algorithm dictates which candidate solutions correspond to better solutions in the problem domain [17]. Only the candidates deemed fit are allowed to reproduce or to survive to a next generation. The genetic algorithm thereby optimises the fitness function to produce a globally optimum solution.

Potgieter and Engelbrecht developed a hybrid genetic algorithm (GASOPE) capable of evolving polynomial expressions that are structurally optimal. The authors defined optimality as the shortest polynomial with the best possible function approximation. GASOPE was tested by approximating several non-linear functions and evaluating its prediction error. GASOPE was shown to be significantly faster than a neural network approach, whilst producing comparable results [16]. Note that GASOPE is regarded as a polynomial regression model in the context of this paper.

Building on the success of GASOPE, Potgieter and Engelbrecht employed GASOPE as the model used at the leaf nodes of a model tree. The proposed model tree (GPMCC) made use of a genetic program to evolve its tree structure [17]. Hitherto discussed decision trees all employed greedy approaches to induce its structure. The use of a greedy approach can be problematic as they are susceptible to becoming stuck in locally optimal solutions. This problem is attributed to the short-sighted nature of a greedy approach. Genetic approaches to inducing the structure of a decision tree outperform greedy approaches in the size of the induced tree, but at the expense of computational cost [2]. This is due to genetic approaches seeking a globally optimal solution as opposed to iteratively favouring solutions that yield local optima.

The genetic heuristic of GPMCC generates multiple tree candidates by randomly choosing a node to expand on. The feature on which to split, as well as the splitting value was randomly selected. Specialized mutation and crossover operators were introduced. Mutation ensured that new genetic material, utilising domain-specific knowledge, would be injected into the population. This was done to improve the quality of a candidate solution. The crossover operator was used to splice together two candidates. The fitness function combined the complexity of a tree, in terms of its size and number of polynomial terms it encapsulated, as well as the difference in the predicted and actual output of an observation.

GPMCC was compared to Cubist on 13 benchmarking/artificial datasets producing significantly smaller tree structures, whilst being competitive with the accuracy of its predictions. For the majority of datasets, the order of the polynomials produced by GASOPE at the leaf nodes of GPMCC never exceeded three. This meant that cubic surfaces were sufficient in describing the non-linear relationships within these datasets. The disadvantage of GPMCC proved to be the speed at which the genetic algorithm induces a model tree. Potgieter and Engelbrecht attributed the slow computational speed

(71) Are you sure about this? Splits were done to maximize classification accuracy? True for the initial trees?

to the recursive procedures that perform fitness evaluation, crossover, and mutation of genetic candidates [17].

1.5. Model tree ensembles

In the final section of this chapter, model tree ensembles are discussed. Model tree ensemble research is not widespread and the computational cost associated with model tree ensembles is hypothesised as the main factor contributing to the lack of abundant research. Section 1.5.1 motivates the reason why further research into model tree ensembles can be beneficial and introduces two successful ensembles which comprise the M5 model tree. Section 1.5.2 discusses the performance of RMT. Finally, MTE is examined in Section 1.5.3.

1.5.1. M5 ensembles

The challenges of developing ensembles of model trees are the increased computational cost and the difficulty of interpreting the models [19]. One of the advantages decision trees have over competing supervised learning techniques is the interpretability it exhibits. Decision trees not only produce adequate performance, but also gives insight into the relationships between the input and output variables for a given problem. When ensembled, decision trees lose its interpretability and mores so when random effects are introduced. The performance increase of ensembling decision trees has to be adequate to justify the loss of interpretability. However, model trees have an even greater variance than decision trees. Ensemble methods have proven effective in minimizing the variance of model trees [19]. There are various approaches to ensemble modelling, those applicable in the context of this paper were discussed in Section 1.2.

Aleksovski and Pfahringer are two of the few who have applied ensemble methods to model trees for regression problems, with each author incorporating a unique approach [1, 15]. The common denominator between the approaches these authors employed was the use of the M5 model tree as the base learner. Each of the model tree ensemble approaches discussed here also incorporated an aspect of randomness in the induction step. The authors reference the success Breiman achieved with random forests as justification. Incorporation of randomness into an ensemble increases the diversity present in the learners that comprise the ensemble, allowing for greater prediction accuracy [6].

Both Aleksovski and Pfahringer opted to only incorporate model trees that comprise linear models. To the best knowledge of this paper, there is no research published yet on ensembles of model trees that comprise non-linear models.

(72) You have not discussed bias-variance for model trees.

(73) paper → thesis
↳ does not have knowledge.

1.5.2. Pfahringer's random model trees

Pfahringer modified the M5 algorithm to grow balanced trees within an ensemble (RMT), largely based on Breiman's random forest with little deviation other than the use of model trees as base learners [15]. Balanced trees are defined as trees incorporating the same number of observations in each leaf node. Pfahringer developed balanced trees by always selecting for the split value the median of a feature.

Pfahringer did not directly compare RMT to a single M5 tree. Instead, RMT was compared to a simple linear regression model, Gaussian process regression (GPR) and additive groves (AG) [15]. At the time, GPR and AG were considered competing state-of-the-art regression methods. RMT outperformed both GPR and AG in terms of computational efficiency, whilst having competitive performance regarding predictive accuracy [15].

1.5.3. Aleksovski's model tree ensembles

Aleksovski was tasked with developing models for dynamic systems, whose data was not evenly distributed in the feature space [1]. Aleksovski, therefore, opted not to incorporate balanced trees for base learners to avoid poor approximations of the data [1]. Aleksovski's approach to growing a model tree within an ensemble (MTE) included three key features: randomised splitting features, *fuzzification*, and ensemble pruning.

MTE's induction is based on bagging, meaning that subsets of randomly selected features with replacement[12] are created for each tree in the ensemble. The standard induction of M5 follows on each subset, i.e. growth that favours a reduction in standard deviation and pruning nodes to reduce prediction error.

MTE omits the smoothing process that M5 incorporated and instead implemented fuzzification to prevent the discontinuities each split within the model tree introduced. Aleksovski stated that the smoothing procedure of M5 produced low performing models, and fuzzification is used instead to combat this [1]. Fuzzification removes discontinuities from the model's prediction by creating a smooth transition between two local models. Splits are transformed into fuzzy splits via the implementation of a sigmoid function. For a tree comprising a single split, the prediction of a given observation, $o(\boldsymbol{x})$, is accordingly calculated as

$$o(\boldsymbol{x}) = \mu(x_j, c, \alpha) f_1(\boldsymbol{x}) + \mu(x_j, c, \alpha) f_2(\boldsymbol{x}) \tag{1.12}$$

with

$$\mu(x_j, c, \alpha) = \frac{1}{1 + e^{-\alpha(x_j - c)}} \tag{1.13}$$

where, f_1 and f_2 are the linear models of two adjacent leaf nodes divided by a split on the j^{th} feature with value c. Finally, α is a fuzzy parameter calculated using cross-validation.

[12] Subsets of features are always chosen from the entire pool of available features and the act of choosing a feature does not remove it from this pool.

It is important to note that Aleksovski did not incorporate fuzzification into the induction of the tree structure as it proved to decrease the efficiency of the M5 algorithm due to the added computational cost it demanded [1].

Once the ensemble is fully grown, i.e. a specified number of learners have been induced on the dataset, the ensemble is pruned as a whole via a greedy selection procedure. As is the case with individual tree pruning, the output error of MTE is evaluated both with and without each tree[13] in the ensemble. If a particular tree contributes to an increase in the prediction error, it is removed from the ensemble. The prediction of MTE is calculated through the uniformly weighted average of the predictions each tree within the ensemble makes.

MTE performed competitively against other state-of-the-art methods, including neural networks, in terms of its prediction error, whilst having the advantage of being quite robust to noise. Aleksovski described the MTE as being resilient to data that exhibited up to 20% noise. This resilience to noise is contributed to injection of randomness in the induction of MTE and helped MTE to produce a low variance error [1].

[13] In the case of individual tree pruning, nodes are removed.