1.4. Model trees

Breiman's early work on improving the robustness of decision trees showed great success concerning the application of decision trees on classification problems [1]. However, Breiman's focus on classification trees [5, 7] 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 classes.

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, such as a linear regression model or neural network. However, no technique comes 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 [9].

There was still a need for a learning model capable of approximating non-linear regression and providing its user with insights into the relationship of the data. 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 [10]. Model tree leaves can incorporate any multivariate model to better capture the patterns present in the training data.

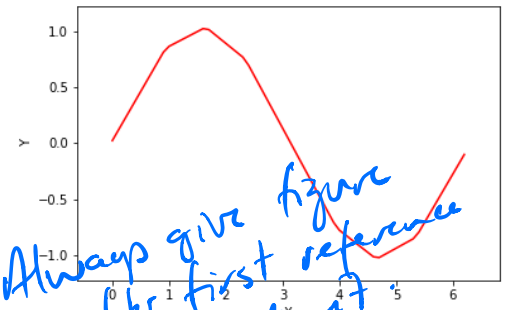

Figure 1.4: The continuous piecewise linear approximation of a sinusoidal wave using a model tree.

Figure 1.4 illustrates how a model tree, that comprises linear models, would approximate 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 over 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 $y = sin(x)$.

(1) What is a model tree problem. Rather refer to these as regression problems, because that is what they ultimately are.

(2) In complete sentence. What favoured?

(3) Avoid such long questions.

(4) For main sections, do not give such detailed discussions, suddenly followed by subsections. Provide a main section with a short paragraph to state the purpose of the section, followed by a section outline. The main text then all organized in subsections.

1.4.1. 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[1]. The first difference between the two methodologies is its splitting criterion used to evaluate a proposed split. Instead of selecting the split that maximises the reduction in variance or absolute deviation, the M5 induction algorithm selects the split which maximises an expected error reduction at a node, defined by the following formula [10]:

$$\Delta error = sd(T) - \sum_i \frac{|T_i|}{|T|} \times sd(T_i) \tag{1.1}$$

where T represents the collection of training samples for the data encapsulated by the node, and $sd(T)$ 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 linear models are constructed for non-leaf as well leaf nodes is to accommodate for *pruning* and *smoothing*.

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 [10]. Once the linear model is constructed, each parameter of the linear model is evaluated for simplification of the linear model. If excluding 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 each node has been constructed a simplified linear model, 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 reduce the complexity of the final model, subsequently reducing its variance.

Smoothing is applied when the model is tasked with predicting a test sample. Smoothing prevents discontinuities from forming between the linear models of neighbouring leaf nodes. Smoothing adjusts predicted values of the model tree as follows: starting with the predicted value at the leaf node, and passing it on along the path to the root node, the predicted

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

value is 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:

$$y_i = \frac{ny_{i-1} + ky}{n + k} \tag{1.2}$$

where n is the number of training samples at the previous node along the path, y_{i-1} is the adjusted prediction passed on 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.

1.4.2. Early M5 model tree performance

Quinlan compared the M5 model tree to MARS (multivariate adaptive regression spline) as this was a popular regression model that proved effective on non-linear data at the time of the M5 model tree's conception [10]. MARS is similar to M5 as it is also non-parametric[2] 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 would struggle past no more than twenty [10].

1.4.3. 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 skipped over. 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 [9]:

$$w_0 + w_1x_1 + w_2x_2 + ... + w_kx_k \tag{1.3}$$

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. Through experimentation Wang and Witten also deemed the simplification of the linear models compromising to the size of the model tree and opted to omit this induction step, as sometimes much smaller trees were obtained when all features were left in the linear models. The M5' model tree was shown performing better than the original M5 tree on the standard datasets for which results where available [9].

[2] 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.

1.4.4. Shortcomings of model trees comprising linear models

The M5 model has been shown to perform inadequately in comparison to other solutions on datasets that exhibit noisy patterns and highly non-linear relationships between its features. 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[3] 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 overpruning and has been shown to have negative affects on a model tree's performance [11].

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, accuracy 19.1% higher 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 given there are a higher number of input parameters [10].

Quinlan recommended researching the application of non-linear models at the leaf nodes [10], 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 in 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 computation 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. There is also the fallacy in expecting the increase in complexity of a model to guarantee an improvement in its performance. Often a simpler approach is the favoured solution to a problem [13]. These factors are hypothesised as contributing to the lack of abundant research on model trees that incorporate non-linear models.

[3] Post-pruning is the 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.

1.5. Non-linear solutions

1.5.1. Kernel Regression

Torgo developed the hybrid regression tree (HTL) that, amongst other proposed models, incorporated kernel regression at its leaf nodes [14]. 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 HTL incorporates is known as a lazy learner as the system only generalizes the training data once a prediction is made[4].

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

$$MSE = \frac{1}{N}\sum_{i}^{N}(y_i - o_i)^2 \tag{1.4}$$

where N is the number of instances over which the MSE is computed, y_i the actual target value of the i^{th} instance, and o_i its predicted value. The predicted value for each training instance is chosen as the average target value in the subtree where the split is proposed[5]. 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 [14] and therefore opts to use CART methodology to induce the tree structure.

The kernel regression model comprises a k-nearest neighbours model and Gaussian kernel function. The kernel regression model calculates predictions using only the training samples most similar to a given query. This similarity based on a distance metric between two instances. The Gaussian kernel function increases the weight, based on the distance metric, of the neighbours the nearer they are. Given the query point $\boldsymbol{q}$, the prediction is calculated through the following function [14]:

$$o(\boldsymbol{q}) = \frac{1}{SK}\sum_{i}^{k} y_i \times K\left(\frac{D(\boldsymbol{x_i}, \boldsymbol{q})}{h}\right) \tag{1.5}$$

with

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

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

[5]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.

and

$$K(d) = e^{-d^2} \tag{1.7}$$

where $D(\cdot)$ is a distance function between two vectors, h the bandwidth parameter, and x_i the i^{th} instance of the 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 (1.5) portrays how the computational complexity grows as the value of k increases due to the added terms in both the kernel function and encapsulated distance function. For experimentation, Torgo chose $k = 3$ without specifying the reasoning behind [14].

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[6]. Although Torgo mentions the M5 model tree incorporating similar linear models to the linear HTL variant, Torgo does not compare the performance of HTL directly to that of M5 [14].

HTL was further improved on by imposing the kernel regression model, described by Equation (1.5), on the linear model in Equation (1.3). The resulting model now incorporating the weight vector, $\boldsymbol{w}$, of the linear model is described by the equation;

$$o(\boldsymbol{q}) = \boldsymbol{w}\boldsymbol{q} - \frac{1}{SK}\sum_{i}^{k} e_i \times K\left(\frac{D(\boldsymbol{x_i}, \boldsymbol{q})}{h}\right) \tag{1.8}$$

where e_i is the error of the linear model calculated as $e_i = \boldsymbol{w}\boldsymbol{x_i} - y_i$. This form 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 [15].

PLT was compared to MARS as well as a commercial version of M5, Cubist, on 12 separate benchmarking datasets producing highly competitive results, though Torgo described the use of PLT as being computationally heavy for the same reasons HTL is considered computationally expensive [15].

Torgo recommends 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 [15].

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

1.5.2. 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]:

$$f(\boldsymbol{x}) = \sum_{\tau=0,\Sigma_{j=1}^{m}\lambda_j=\tau}^{n} \left(w_{(\lambda_1,\lambda_2,\ldots,\lambda_m)} \prod_{q=1}^{m} x_q^{\lambda_q} \right) \tag{1.9}$$

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 instances 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 replicates natural selection mechanisms to evolve an optimal solution from multiple candidates [17]. "Survival of the fittest" is simulated across numerous generations, each generation having multiple candidate solutions [18]. The fitness function of a genetic algorithm dictates which candidate solutions correspond to better solutions in the problem domain [2]. Only the candidates deemed fit are allowed to reproduce or 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. 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 getting 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 [19]. This is due to genetic approaches seeking the 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 instance.

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 trees. Potgieter and Engelbrecht attributed the slow computational speed to the recursive procedures that perform fitness evaluation, crossover, and mutation of genetic candidates [2].

1.6. Model tree ensembles

The use of model trees in an ensemble is something not widely researched. The challenges of developing ensembles of model trees are the increased computation cost and the difficulty of interpreting the models [18]. However, these challenges are rewarded by the improved performance ensemble approaches have to offer. There are various approaches to ensemble modelling, those applicable in the context of this paper were discussed in Section 1.3.

Authors Aleksovski and Pfahringer are two of the few who have applied ensemble approaches to model trees for regression problems, with each author incorporating a unique approach [11, 20]. 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. Introducing randomness into an ensemble increases the diversity present in the learners that comprise the ensemble, allowing for greater prediction accuracy [7].

1.6.1. Pfahringer's Random Model Trees

Pfahringer modified the M5 algorithm to grow balanced trees within an ensemble approach (RMT), largely based on Breiman's Random Forest with little deviation other than the use of model trees as base learners. 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). 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 [20].

1.6.2. Aleksovski's Model Tree Ensembles

Aleksovski was tasked with developing models for dynamic systems, whose data was not evenly distributed in the feature space. Aleksovski, therefore, opted not to incorporate balanced trees for base learners as to avoid poor approximations of the data [11]. 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 subsets of randomly selected features, with replacement[7], are created for each tree in the ensemble. The standard induction of

[7]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.

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 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 used instead to combat this [11]. 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 instance, $o(\boldsymbol{x})$, is accordingly calculated as:

$$o(\boldsymbol{x}) = \mu(x_j, c, \alpha) f_1(\boldsymbol{x}) + \mu(x_j, c, \alpha) f_2(\boldsymbol{x}) \tag{1.10}$$

with

$$\mu(x_j, c, \alpha) = \frac{1}{1 + e^{-\alpha(x_j - c)}} \tag{1.11}$$

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. It is worth noting 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 it demanded [11].

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 the ensemble is evaluated both with and without each tree[8] it comprises. If a particular tree contributes to the increase of the prediction error, it is removed from the ensemble. The prediction of the ensemble is calculated through the uniformly weighted average of its trees.

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 produce a low variance error [11].

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

Bibliography

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[2] G. Potgieter and A. P. Engelbrecht, "Evolving model trees for mining data sets with continuous-valued classes," *Expert Systems with Applications*, vol. 35, no. 4, pp. 1513 – 1532, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417407003673

[3] P. Geurts, "Contributions to decision tree induction: bias/variance tradeoff and time series classification," Ph.D. dissertation, University of Liège Belgium, 2002.

[4] S. Singh, "Understanding the Bias-Variance Tradeoff," 2018. [Online]. Available: https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229. [Accessed: 22- Mar- 2020].

[5] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[6] A. Nagpal, "Decision Tree Ensebmles- Bagging and Boosting," 2017. [Online]. Available: https://towardsdatascience.com/decision-tree-ensembles-bagging-and-boosting-266a8ba60fd9. [Accessed: 23- Mar- 2020].

[7] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[8] G. Moisen, "Classification and regression trees," *In: Jørgensen, Sven Erik; Fath, Brian D.(Editor-in-Chief). Encyclopedia of Ecology, volume 1. Oxford, UK: Elsevier. p. 582-588.*, pp. 582–588, 2008.

[9] Y. Wang and I. Witten, "Induction of model trees for predicting continuous classes," *Induction of Model Trees for Predicting Continuous Classes*, 01 1997.

[10] J. R. Quinlan *et al.*, "Learning with continuous classes," in *5th Australian joint conference on artificial intelligence*, vol. 92. World Scientific, 1992, pp. 343–348.

[11] D. Aleksovski, J. Kocijan, and S. Džeroski, "Model-tree ensembles for noise-tolerant system identification," *Advanced Engineering Informatics*, vol. 29, no. 1, pp. 1–15, 2015.