Classification and Regression using AddiVortes: (Bayesian) Additive Voronoi Tessellations

Adam J. Stone

Classification and Regression using AddiVortes: (Bayesian) Additive Voronoi Tessellations

Adam J. Stone

A Thesis presented for the degree of

Doctor of Philosophy

Department of Mathematical Sciences
Durham University
United Kingdom

June 2026

Abstract

This thesis presents the development of AddiVortes, a novel algorithm that integrates Voronoi tessellations within a Bayesian additive ensemble framework. By leveraging the flexible spatial partitioning capabilities of Voronoi tessellations, the method is designed to capture the interactions between covariates more effectively, where traditional axis-parallel splits often prove suboptimal. Extensive empirical analysis on both real-world and simulated datasets demonstrates that AddiVortes performs competitively against leading "black-box" models, including random forests and Bayesian Additive Regression Trees (BART). To address the challenge of categorical data, the thesis introduces a new technique called permutation encoding for effectively incorporating nominal covariates. The framework is further extended to handle binary and nominal classification through probit link functions and data augmentation, as well as heteroscedastic regression by employing a secondary ensemble to model covariate-dependent noise. The work concludes with a discussion of the algorithm's properties and outlines potential directions for future research.

Keywords: AddiVortes, Voronoi tessellations, Bayesian methods, Nonparametric regression, Multivariate analysis.

Declaration

The work in this thesis is based on research carried out at the Department of Mathematical Sciences, Durham University, UK. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Acknowledgments

I would like to extend my sincere gratitude to John Paul Gosling, for his exceptional guidance and mentorship throughout this research. His academic expertise and rigorous feedback have been instrumental in shaping the direction and overall quality of this work. I am also deeply thankful to Emmanuel Ogundimu, for his critical insights and continuous support. Emmanuel's specialised knowledge and constructive advice proved invaluable in navigating the complexities of this project and refining my methodology.

I would also like to thank my examiners, Ioanna Manolopoulou and Andrew Golightly, for their insightful feedback and recommendations. Their comments have significantly helped to improve the quality of this work and will continue to shape future research arising from this thesis.

Beyond academia, I wish to express my deepest appreciation to my friends and family for their enduring support. They have been a constant source of motivation, and I remain profoundly grateful for their belief in my abilities throughout this undertaking.

Contents

Chapter 1

Introduction

Most modern predictive models focus on capturing high-dimensional, non-linear structures in real-world data. While classical parametric methods (like generalised linear models, Nelder and Wedderburn, 1972) offer interpretability, their rigid functional assumptions often result in high bias. To overcome this limitation, flexible, non-parametric "black-box" ensemble methods, particularly those based on decision trees (such as random forests, (Breiman, 2001), and gradient boosting machines, (Friedman, 2001)), have become the standard for accurately modelling relationships and interactions. While these non-parametric methods offer greater flexibility, decision tree algorithms frequently lack a coherent framework for uncertainty quantification, while alternatives like kernel smoothing (Wand and Jones, 1995) or splines (Wahba, 1990) often struggle with the curse of dimensionality.

The Bayesian Additive Regression Trees (BART) (Chipman et al., 2010) model excels by summing over a regularised ensemble of "weak" decision trees to produce a "strong learner" that can accurately capture more complex functions. Unlike many algorithmic ensemble methods, BART operates within a fully generative Bayesian framework, which allows it to provide robust uncertainty quantification through predictive distributions rather than just simple point estimates. To ensure stability and prevent any single tree from dominating the model, regularisation priors are used to intentionally constrain each tree to capture only a small portion of the overall relationship. BART has been highly successful in several real-world applications, including electricity demand forecasting (Chen et al., 2016), genomic prediction (Waldmann, 2016), and credit risk modelling (Zhang and Härdle, 2010).

However, the standard BART model is constrained by its use of axis-aligned splits, which create rigid, hyperrectangular decision boundaries that may not efficiently represent smooth or oblique data structures. The step-wise nature of the individual trees can produce sharp discontinuities in the predictive surface, and the framework can struggle to handle nominal categorical variables without significantly increasing the dimensionality of the covariate space. Therefore, many extensions of the BART model have been developed and a few of these are explained in Chapter 2.

This thesis presents AddiVortes (Additive Voronoi Tessellations) (Stone and Gosling, 2025a), a novel Bayesian ensemble method that replaces the rigid, axis-aligned boundaries of decision trees with the flexible, convex, polygonal cells of Voronoi tessellations. A Voronoi tessellation partitions space based on distances to centres, allowing its boundaries to assume arbitrary orientations. By defining the regression function as an additive sum of these flexible tessellations, AddiVortes uses the successful ensemble structure and Bayesian backfitting algorithm of BART but fundamentally improves how the input space is partitioned, enabling the model to capture spatial relationships and interactions more effectively.

A naive application of Voronoi tessellations in high dimensions can be computationally expensive, prone to overfitting and sensitive to noise. The complexity of tessellation grows exponentially and distances become less informative as the number of dimensions or cells increases. A key design principle in AddiVortes is therefore to construct each tessellation on a low-dimensional regularised subset of covariates with a regularised number of centres. Individual tessellations remain computationally tractable, while their additive ensemble allows the model to capture complex higher-order interactions. This low-dimensional ensemble strategy mitigates the curse of dimensionality that would otherwise make Voronoi-based modelling impractical.

In this thesis, we illustrate the superiority of AddiVortes and the use of Voronoi tessellations over decision tree based methods and other "black-box" models using several simulated and real-world applications. A range of metrics are used to compare methods within different settings.

Real-world data sets frequently contain nominal categorical covariates, which present a particular challenge for geometric partitioning methods. Voronoi tessellations rely on a notion of distance, but nominal covariates have no natural metric: there is no inherent sense in which one level is closer to another. Standard encodings such as one hot (dummy) vectors inflate the dimension and create arbitrary geometry, while arbitrary numeric labels impose artificial orderings that may not

reflect the underlying structure. This thesis introduces a permutation encoding framework that treats the ordering of categorical levels as a latent parameter. By placing a prior over permutations and updating them within the MCMC scheme, the method learns geometries that group levels with similar behaviour, effectively embedding nominal categories into a metric space suitable for Voronoi tessellations. In doing so, it allows AddiVortes to handle mixed data types and to recover latent ordinal structure where it is supported by the data.

The AddiVortes framework is further extended beyond standard regression to encompass classification tasks. For binary and multinomial outcomes, the model is adapted using a data augmentation strategy based on the probit link function. In the binary case, latent variables are introduced to map the discrete class labels to a continuous scale, allowing the standard regression AddiVortes model to be applied. For multinomial classification, where outcomes fall into one of several unordered categories, the challenge is greater due to the need to model a vector of latent utilities and the associated identifiability constraints on the error covariance matrix. Building on recent advances in augmentation samplers for multinomial probit models, this work implements efficient sampling schemes that ensure stable, rapid mixing of MCMC chains. This allows AddiVortes to generate probabilistic classifications that are robust to class imbalance and can capture complex, non-linear decision boundaries between multiple classes.

Another critical aspect of real-world modelling addressed in this thesis is heteroscedasticity. Standard regression models, including the original BART, typically assume homoscedasticity, where the error variance is constant across the input space. However, in many domains, such as finance or environmental science, the variability of the response often depends on the predictors. To accommodate this, the Heteroscedastic AddiVortes model is developed, which jointly models the conditional mean and the conditional variance of the response. The mean is modelled using the standard sum-of-tessellations approach, while the variance is modelled simultaneously using a multiplicative product-of-tessellations ensemble. This dual-ensemble structure allows the model to capture both the signal (the mean function) and noise (the variance function), providing more accurate posterior distributions.

The thesis structure moves logically from fundamentals to novel contributions. Chapter 2 reviews BART, establishing the theoretical baseline for regression and gives details on the developments. Chapter 3 then focuses on the geometric foundations by introducing Voronoi tessellations,

including their properties, efficient computational search algorithms (like k-d trees) Wang et al. (2021), and explores the impact of alternative metrics (such as Manhattan and Chebyshev) and dual structures (like Delaunay triangulations) on the resulting partitions.

Chapter 4 presents the core AddiVortes model. It details the mathematical formulation of the additive Voronoi ensemble, the specification of regularisation priors for tessellation structures and cell parameters, and the derivation of the Metropolis-Hastings acceptance ratios required for the MCMC sampler. This chapter demonstrates the model's performance on standard regression benchmarks, illustrating its competitive predictive accuracy and its ability to recover true functions in simulation studies. It also discusses the computational implementation, highlighting the strategies used to ensure scalability.

The thesis continues by developing the algorithm across three subsequent chapters, focusing on extensions to handle specific data types better. Chapter 5 introduces a Permutation Encoding technique to handle nominal categorical covariates by treating category ordering as a learnable parameter within a reversible-jump MCMC framework. Chapter 6 extends the framework to classification, detailing data augmentation strategies for probit models and showing its utility in applications, including the home mortgage outcome predictions (Stone et al., 2025). Chapter 7 is based on work in Stone and Gosling (2025b), which introduces the heteroscedastic AddiVortes model, formulating a joint model for mean and variance, deriving necessary MCMC updates, and applying it to real-world datasets like used car prices to capture non-constant variance.

Finally, Chapter 8 concludes the thesis by summarising the key findings and contributions. It discusses the implications of replacing trees with tessellations in ensemble learning and outlines potential opportunities for future research, such as developing soft tessellations to allow smoothness and extending the method to non-Euclidean manifolds. Through these chapters, this thesis aims to establish AddiVortes as a robust, flexible, and statistically rigorous alternative to tree-based ensembles, capable of addressing multiple challenges of modern data analysis.

Chapter 2

Bayesian Additive Regression Trees (BART): A Review and Current Work

Bayesian Additive Regression Trees (BART), introduced by Chipman et al. (2010), is a highly versatile and robust statistical model that continues to be a focal point of contemporary research due to its exceptional ability to model complex interactions and provide reliable uncertainty quantification. It integrates tree-based models within a Bayesian framework and models the unknown regression function as a sum of regression trees, with each tree capturing low-order structure and the ensemble representing a complex, non-linear response surface. This approach is similar to boosting methods (Friedman, 2001), but BART differs fundamentally by being fully generative and by placing priors on both the tree structures and their terminal node values.

BART uses a Bayesian backfitting MCMC algorithm to draw samples from the posterior distribution. This results in samples from predictive distributions rather than point estimates, making BART particularly valuable for applications that require calibrated measures of uncertainty. The model's regularisation priors prevent overfitting, while its ensemble structure allows for the modelling of interactions and non-linearities even in high-dimensional datasets.

The versatility of BART is further demonstrated by its methodological extensions. Heteroscedastic BART (Pratola et al., 2020) allows the model to accommodate varying noise levels across the input space, thereby enhancing performance in domains such as financial modelling and environmental data analysis, where homoscedasticity is unrealistic. For time-to-event data, BART has

been adapted for survival analysis through modifications to the likelihood and the introduction of censored-data mechanisms (Sparapani et al., 2016). Meanwhile, BART's non-parametric nature and ability to flexibly model counterfactual outcomes have made it a standard method for causal inference (Hill, 2011; Hahn et al., 2020; Caron et al., 2022).

This chapter provides a comprehensive account of BART, starting from its foundational formulation as a regression model and extending to its various adaptations for different environments such as classification, heteroscedastic modelling, survival analysis, and causal inference. Emphasis is placed on the mathematical formulation, and by the end of the chapter, readers will understand why BART has become a strong competitor in modern Bayesian non-parametric modelling.

2.1 Regression model

Many regression methods aim to model a conditional expectation function f that relates the continuous variable Y to some or all of p (potential) covariates $\boldsymbol{x} = (x_1, \ldots, x_p)$, such that

$$Y = f(\boldsymbol{x}) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2),$$

under the assumption that the noise is mean zero and homoscedastic. BART models the systematic part of this relationship, $f(\boldsymbol{x}) = \mathbb{E}(Y \mid \boldsymbol{x})$, the mean of Y given $\boldsymbol{x}$, using a sum of stepwise functions built upon decision trees.

2.1.1 Decision Trees

At the core of BART lies binary decision trees (also known as regression trees). A decision tree partitions the covariate space into distinct subgroups. Within each subgroup, generally the model's prediction remains constant; that is, all observations assigned to the same subgroup receive the same output value.

Consider the example illustrated in Figure 2.1, which shows a single decision tree, constructed using two covariates. The decision tree outlines a series of binary splitting rules located at the interior decision nodes (represented as boxes). Each observation is passed through the tree, following a path determined by these decision rules. BART uses "binary" decision trees; that is, the condition

is either true or false. In this decision tree, the rule is that, if the condition is satisfied, you follow the right-hand path.

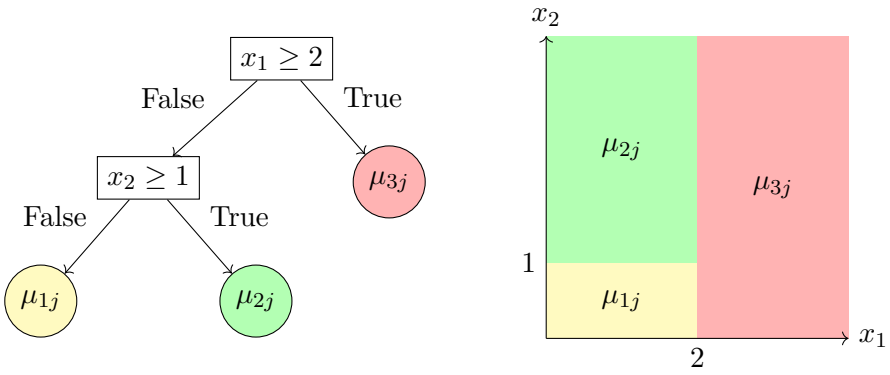


Figure 2.1: A binary decision tree is depicted on the left, and the partition that it produces is illustrated on the right.

At the end of each path, the observation reaches a terminal node, illustrated as circles at the bottom of the tree. Each terminal node represents a distinct subgroup and is associated with a specific parameter value that constitutes the tree's prediction for any observation falling into that subgroup. This parameter is referred to as the output value for a given observation.

The partition produced by the decision tree is displayed on the right of Figure 2.1. Notice that the partition is very rigid; all its boundaries are axis-aligned and orthogonal to each other. In Chapter 3, we see how Voronoi tessellations, which are what AddiVortes is built upon, can partition the covariate space with much more flexibility and therefore capture covariate interactions better.

BART uses an ensemble of decision trees, and T_j denotes the structure of the j^{th} tree, including the splitting rules of the interior nodes. The corresponding values for each terminal node are given by $\boldsymbol{M}_j = \{\mu_{1j}, \ldots, \mu_{b_j j}\}$, where b_j is the number of terminal nodes in the tree T_j. The output value for a sample $\boldsymbol{x}$ of tree T_j is given by the function $g(\boldsymbol{x} \mid T_j; \boldsymbol{M}_j)$. For example, the observation $\boldsymbol{x} = (1.1, 2.3)$ for the decision tree j in Figure 2.1, gives

$$g(\boldsymbol{x} \mid T_j; \boldsymbol{M}_j) = \mu_{2j}.$$

Traditionally, decision trees are constructed by trying to minimise a loss function, like the residual sum of squares. To achieve this, a greedy algorithm is employed that builds the tree by selecting

the locally optimal split at each step. In contrast, BART adopts a Bayesian approach where the model is defined as an additive ensemble of trees, this is explained in Section 2.1.2, with each tree contributing a small fraction to the overall prediction. During the MCMC inference scheme, posterior sampling is conducted by sequentially updating each tree via a backfitting algorithm, conditional on the current partial residuals of all other trees in the ensemble. This probabilistic framework is a significant advantage, as it inherently facilitates quantifying uncertainty, offering not just a single prediction but a credible range of possible outcomes.

2.1.2 Additive ensemble

A weak learner is an intentionally constrained model that is simple and has a small impact on the overall prediction. A core principle in ensemble learning is that combining many weak learners can produce a strong learner (Schapire, 1990). This underlies methods like boosting (Friedman, 2001), bagging (Breiman, 1996), random forests (Breiman, 2001), and BART. Weak learners typically have high bias and capture only a fraction of the overall relationship, but when their individual components are combined, they form a much stronger learner that can capture more complex relationships.

BART uses an additive ensemble referred to as the "sum-of-trees" model by Chipman et al. (2010), which models the function f as the sum of the output values from multiple decision trees, that is,

$$f(\boldsymbol{x}) = \sum_{j=1}^{m} g(\boldsymbol{x}; T_j, \boldsymbol{M}_j),$$

where m is the number of trees in the ensemble.

To avoid overfitting, BART uses regularisation priors (see Subsection 2.1.3) that constrain tree complexity and shrink terminal node parameters towards zero, ensuring each tree acts as a weak learner.

In BART, each shallow tree explains only a small part of the conditional expectation function. Fitting subsequent trees to the residuals progressively refines the model, but the regularisation priors encourage gradual improvement by limiting the influence of individual trees and reduces the risk of overfitting on the training data. As more trees are added, the ensemble can approximate

increasingly flexible functions. This behaviour is analogous in spirit to classical universal approximation results for nonparametric regression estimators (for example, Stone, 1982), where additive collections of simple basis functions can approximate a wide class of continuous functions. This additive structure also helps balance bias and variance since an individual shallow tree has high bias but low variance (it is stable). By adding many trees together, the bias of the ensemble is systematically reduced as the model becomes complex enough to fit the true signal.

2.1.3 Regularisation priors

Ensemble methods require some regularisation to keep the influence of individual components (i.e., trees for BART) constrained. BART does this using regularisation priors that limit the complexity of individual decision trees and restrict their influence on the overall fit.

The aim of BART is to sample from the posterior distribution with density

$$p((T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m), \sigma \mid \boldsymbol{x}_{train}, \boldsymbol{y}_{train}),$$

and so the prior distribution of the model has density

$$\pi((T_1, M_1), \ldots, (T_m, M_m), \sigma).$$

The prior is simplified by assuming that each tree structure is independent of other tree structures, of the output values, and of σ. It is also assumed that all output values are independent of each other and are independent of σ. These assumptions imply

$$\begin{aligned}\pi\left((T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m), \sigma\right) &= \prod_{j=1}^{m} \left[\pi(T_j, \boldsymbol{M}_j)\right] \pi(\sigma) \\ &= \prod_{j=1}^{m} \left[\pi\left(\boldsymbol{M}_j \mid T_j\right) \pi(T_j)\right] \pi(\sigma)\end{aligned}$$

and

$$\pi(\boldsymbol{M}_j \mid T_j) = \prod_{i=1}^{b_j} \pi\left(\mu_{ij} \mid T_j\right),$$

where b_j is the number of terminal nodes in the j^{th} decision tree. These independence restrictions

mean that the prior can be split into three parts: the T_j prior, the $\mu_{ij} \mid T_j$ prior and the σ prior.

The prior for T_j

The prior on the tree structures, $\pi(T_j)$, is designed to strongly favour shallow trees with terminal node parameters close to zero. Each tree is assigned identical and independent priors, which follow a form similar to that of the Classification and Regression Tree (CART) model introduced by Chipman et al. (1998). The probability that a node at depth d will split (that is, not terminal) is given by

$$\text{P(split at depth } d) = \alpha(1+d)^{-\beta}, \qquad \alpha \in (0,1), \quad \beta \in (0,\infty),$$

with default values $\alpha = 0.95$ and $\beta = 2$, as proposed by Chipman et al. (2010). This specification places strong prior weight on small trees. For example, it has the following prior probabilities for a tree having 1, 2, 3, 4, and more than 5 terminal nodes: 0.05, 0.55, 0.28, 0.09, and 0.03, respectively.

In the default specification, the splitting variable in each non-terminal node is drawn uniformly from the available covariates. Conditional on the variable, the splitting point is drawn uniformly either from a large set of quantiles over the range of the selected covariate (for continuous variables) or over all possible values (for binary or ordinal variables). Classically, for nominal categorical variables with q levels, the variable is typically expanded into q binary indicators; this method is called one-hot encoding. Further details on the use of nominal covariates are described in Section 2.2 and later in Chapter 5 we introduce a new way of nominal covariate inclusion that could be implemented for BART based on permutating covariates within the MCMC algorithm.

A drawback of selecting a specific observation from the training data as a split value is that it makes the model more susceptible to overfitting and creates artificial boundaries that don't generalise well to new data. When a decision tree selects a split, it learns a specific rule using the training data values, which can create a model tailored to the data it has seen and too rigid to handle new observations.

Also, splits create hard, axis-aligned boundaries, implying a fundamental difference between values just below and just above the split. To address this limitation, Section 2.6 explores Soft-BART (Linero and Yang, 2018), which utilises a probabilistic approach at interior nodes to create smoother, more flexible decision boundaries. Building on this, Chapter 8 discusses a potential re-

search direction involving "soft tessellations" designed to model continuous functional forms more naturally.

The prior for $\mu_{ij} \mid T_j$

To prevent overfitting, BART employs a shrinking prior as a form of regularisation. This prior pulls the terminal values of the trees towards zero, ensuring that no single tree can become too influential and forcing the model to rely on the cumulative effect of the entire ensemble. The parameters associated with the i^{th} terminal nodes and j^{th} tree, μ_{ij}, are assigned a conjugate normal prior of the form

$$\mu_{ij} \sim \mathcal{N}(\mu_\mu, \sigma_\mu^2).$$

This specification offers significant computational advantages by enabling the analytical marginalisation of μ_{ij} during posterior sampling.

Under the BART framework, the predicted response function is expressed as a sum over m regression trees, and consequently, the prior distribution induced on the expected response $\mathbb{E}(Y \mid \boldsymbol{x})$ is the distribution of a sum of m independent, identically distributed μ_{ij} values. That is,

$$\mathbb{E}(Y \mid \boldsymbol{x}) \sim \mathcal{N}(m\mu_\mu, m\sigma_\mu^2),$$

and a prior is constructed that places substantial probability mass within the plausible range of Y values observed in the data. Denoting the observed minimum and maximum of Y by $y_{\min}$ and $y_{\max}$ respectively, the prior is calibrated so that the distribution $\mathcal{N}(m\mu_\mu, m\sigma_\mu^2)$ assigns most of its mass to the interval $(y_{\min}, y_{\max})$.

To achieve this, the hyperparameters μ_μ and σ_μ^2 are selected so that

$$m\mu_\mu - k\sqrt{m}\sigma_\mu = y_{\min} \quad \text{and} \quad m\mu_\mu + k\sqrt{m}\sigma_\mu = y_{\max}$$

for a pre-specified value of k. Solving these equations gives

$$\mu_\mu = \frac{y_{\min} + y_{\max}}{2m}, \quad \sigma_\mu = \frac{y_{\max} - y_{\min}}{2k\sqrt{m}}.$$

In practice, to simplify this calibration and facilitate automation, the response variable Y is often rescaled linearly so that its transformed values fall within the interval $[-0.5, 0.5]$, thereby setting $y_{\min} = -0.5$ and $y_{\max} = 0.5$. With this transformation, the prior for each μ is centred at zero ($\mu_\mu = 0$), and the standard deviation becomes

$$\sigma_\mu = \frac{0.5}{k\sqrt{m}}.$$

Thus, the prior becomes

$$\mu_{ij} \sim \mathcal{N}\left(0, \frac{1}{4k^2 m}\right),$$

which ensures that the sum of the μ_{ij} values across the m trees gives a reasonable range for the estimation of the expectation function, that is, the overall fit $f(\boldsymbol{x})$, has prior variance approximately $1/(4k^2)$ and hence standard deviation $1/(2k)$. For example, choosing the default value $k = 2$ implies that the prior distribution on $f(\boldsymbol{x})$ places approximately 95% of its mass within the interval $[-0.5, 0.5]$ since approximately 95% of the probability mass in the distribution $\mathcal{N}(0, \frac{1}{16})$ falls in the interval $[-0.5, 0.5]$.

This calibration strategy provides an intuitive, data-informed approach to specifying the prior: it ensures that the prior belief about $\mathbb{E}(Y \mid \boldsymbol{x})$ lies within a reasonable range, avoids both overconcentration and overdispersion, and does not require subjective specification of hyperparameters. The transformation of Y does not affect the modelling of predictor variables, as BART's decision tree structure is invariant under monotonic transformations of $\boldsymbol{x}$. This property circumvents the need for predictor standardisation, unlike models like neural networks, which rely on linear combinations of inputs and are sensitive to their scales.

The prior shrinkage induced by small values of σ_μ has a similar effect to the learning rate or shrinkage parameter in gradient boosting machines. It weakens the influence of individual trees, encouraging them to contribute incrementally to the final prediction. As such, values of k between 1 and 3 are generally effective, with $k = 2$ serving as a well-performing default. However, k may be tuned via cross-validation if more precise control over shrinkage is desired. This prior structure, while mathematically simple, has proven effective in practice and enhances BART's robustness.

The σ Prior

For the residual standard deviation σ, a conjugate prior is placed in the form of a scaled inverse-chi-squared distribution

$$\sigma^2 \sim \frac{\nu\lambda}{\chi^2_\nu},$$

where $\nu > 0$ denotes the degrees of freedom and $\lambda > 0$ is a scale parameter. This prior is convenient for posterior computation and introduces a level of regularisation on the noise variance. The key objective in calibrating this prior is to ensure it reflects plausible beliefs about the data's noise level without imposing excessive concentration or dispersion. A data-driven approach is adopted in specifying the hyperparameter λ, using a conservative overestimate of the noise standard deviation as a reference point.

A natural and effective choice for this overestimate is the residual standard deviation obtained from an ordinary least squares (OLS) fit to the data, denoted by $\hat{\sigma}_{\text{OLS}}$. Since OLS fits tend to underfit in the presence of nonlinearity, the corresponding residual variance generally overestimates the true noise variance. The prior is then calibrated by choosing the scale parameter λ such that a specified quantile q of the prior distribution for σ is aligned with this overestimate, that is,

$$\Pr(\sigma < \hat{\sigma}_{\text{OLS}}) = q,$$

where q is typically set to 0.75, 0.90, or 0.99. This equation is solved numerically for λ given a fixed value of ν. The quantile level q determines the aggressiveness of the prior shrinkage, with larger values of q shifting more prior mass below $\hat{\sigma}_{\text{OLS}}$, thereby favouring lower values of σ.

The choice of degrees of freedom ν affects the shape and tail behaviour of the prior. Smaller values of ν give heavier tails and more diffuse priors, while larger values lead to tighter concentration around the scale parameter.

Chipman et al. (2010) notes that empirical evidence suggests that setting ν in the range of 3 to 10 provides a good trade-off between flexibility and stability. In particular, the setting $(\nu, q) = (3, 0.90)$ serves as a reliable default, producing a weakly informative prior that moderates overfitting while still allowing the data to inform the posterior distribution of σ. Care must be taken not to select values of ν that are too small (e.g., $\nu < 3$), as this can place excessive mass near zero and lead to

unrealistically small noise estimates, resulting in poor generalisation.

In practice, the hyperparameters (ν, q) can also be tuned via cross-validation if a more precise regularisation of σ is desired. However, in most cases, the default setting provides robust performance across a wide range of applications. In the heteroscedastic extensions considered later, this global variance parameter is replaced by a regression function $\sigma(\boldsymbol{x})$ that varies over the covariate space, and the prior on σ is replaced by a prior on the parameters of that variance function.

2.1.4 Markov chain Monte Carlo for single-tree models

Posterior inference in BART models uses Markov Chain Monte Carlo (MCMC) methods, adapting techniques originally developed for single-tree models and integrating them into Metropolis-within-Gibbs samplers. These methods leverage efficient exploration of tree space, which is essential for capturing data patterns and relationships.

Chipman et al. (1998), Denison et al. (1998) and Wu et al. (2007) discuss posterior sampling strategies for single-tree regression and classification models. These strategies employ a blocked Metropolis–Hastings approach to explore the posterior space. The method comprises two primary steps: first, proposing a new tree T^* based on a proposal distribution conditional on the current tree, followed by the computation of the Metropolis–Hastings acceptance ratio

$$\alpha(T_j, T_j^*) = \min\left\{1, \frac{q(T_j^*, T_j)}{q(T_j, T_j^*)} \cdot \frac{p(\boldsymbol{R}_j \mid T_j^*, \boldsymbol{M}_j, \sigma)}{p(\boldsymbol{R}_j \mid T_j, \boldsymbol{M}_j, \sigma)} \cdot \frac{p(T_j^*)}{p(T_j)}\right\},$$

where $q(T_j, T_j^*)$ is the proposal distribution for transitioning from T_j to T_j^*; $P(\boldsymbol{R}_j \mid T_j, \boldsymbol{M}_j, \sigma)$ is the residual likelihood and $P(T_j)$ is the prior probability of the tree structure. The acceptance of T_j^* occurs with probability α, whereas rejection results in retaining the current configuration. This approach ensures thorough exploration of potential tree structures, optimising predictive performance and model fit.

Following the tree update, the corresponding terminal node values $\boldsymbol{M}_j$ undergo a sampling process from their full conditional distribution $p(\boldsymbol{M}_j \mid T_j, \boldsymbol{y}_{\mathbf{train}})$, typically available in closed form due to the model's conjugate setting. This blocked updating efficiently manages the varying dimensionality inherent to the terminal node values, enhancing overall computational efficiency and stability within the MCMC framework.

Most BART implementations incorporate proposal distributions that modify the current tree using specific moves: grow (splitting a terminal node to introduce additional complexity), prune (merging sibling leaves to simplify tree structure), change (altering decision rules at interior nodes), and swap (exchanging decision rules between distinct interior nodes). Wu et al. (2007) and Pratola (2016) further discuss additional tree modification proposals, contributing to the enhancement and refinement of BART models.

2.1.5 Bayesian MCMC backfitting algorithm

The primary method for fitting BART models is an iterative Bayesian backfitting MCMC algorithm, which can be understood as a Metropolis-Hastings algorithm inside a Gibbs sampler. For notational convenience, let $T_{(j)}$ denote the set of all trees in the sum except T_j, and similarly define $\boldsymbol{M}_{(j)}$ as the set of associated terminal node parameters for these trees. The Gibbs sampler proceeds by iteratively drawing from the full conditional distributions of the model parameters. The chain is commonly initialised with m simple single-node trees with the terminal node value equal to $\text{mean}(\boldsymbol{y}_{\mathbf{train}})/m$.

The algorithm now involves m successive draws of each tree-parameter pair $(T_j, \boldsymbol{M}_j)$ conditional on the current values of the other trees and parameters $(T_{(j)}, \boldsymbol{M}_{(j)}, \sigma, \boldsymbol{y}_{\mathbf{train}})$,

$$(T_j, \boldsymbol{M}_j) \mid T_{(j)}, \boldsymbol{M}_{(j)}, \sigma, \boldsymbol{y}_{\mathbf{train}}, \quad j = 1, \ldots, m, \tag{2.1}$$

followed by a draw of the standard deviation σ from its full conditional distribution

$$\sigma \mid (T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m), \boldsymbol{y}_{\mathbf{train}}. \tag{2.2}$$

This general approach is a stochastic generalisation of the backfitting algorithm often used for additive models (Hastie and Tibshirani, 2000). The draw of σ in (2.2) is from an inverse Gamma distribution, which is conjugate to the Normal likelihood, thus σ can be efficiently sampled.

The conditional distribution $p(T_j, \boldsymbol{M}_j \mid T_{(j)}, \boldsymbol{M}_{(j)}, \sigma, \boldsymbol{y}_{\mathbf{train}})$ depends on $(T_{(j)}, \boldsymbol{M}_{(j)}, \boldsymbol{y}_{\mathbf{train}})$ only

through the partial residuals for the j-th tree,

$$\boldsymbol{R}_j = \boldsymbol{y} - \sum_{k \neq j} g(\boldsymbol{x}; T_k, \boldsymbol{M}_k). \tag{2.3}$$

Thus, the m draws in (2.1) are equivalent to m draws from

$$(T_j, \boldsymbol{M}_j) \mid \boldsymbol{R}_j, \sigma. \tag{2.4}$$

This formulation treats $\boldsymbol{R}_j$ as the effective data for updating $(T_j, \boldsymbol{M}_j)$, reducing the problem to that of a single-tree model $\boldsymbol{R_j} = g(\boldsymbol{x}; T_j, \boldsymbol{M}_j) + \varepsilon$.

Since a conjugate prior is used for output values of μ_{ij}, the parameters of $\boldsymbol{M}_j$ can be integrated out to obtain the marginal likelihood for T_j,

$$p(T_j \mid \boldsymbol{R}_j, \sigma) \propto p(T_j) \int p(\boldsymbol{R}_j \mid T_j, \boldsymbol{M}_j, \sigma) p(\boldsymbol{M}_j \mid T_j) d\boldsymbol{M}_j, \tag{2.5}$$

where $p(T_j)$ is the prior on the tree structure and $p(\boldsymbol{M}_j \mid T_j)$ is the prior on its terminal node parameters (σ is independent of T_j and $\boldsymbol{M_j}$ from independence assumptions in Section 2.1.3). This marginalisation is crucial, as it allows T_j to be sampled without sampling $\boldsymbol{M}_j$ simultaneously, simplifying moves that change the dimension of $\boldsymbol{M}_j$.

BART then iteratively updates each using the MH algorithm similar to the previous section. That is, the draw from (2.4) is performed in two steps:

1. Draw the tree structure $T_j \sim p(T_j \mid \boldsymbol{R}_j, \sigma)$. Propose a new tree T_j^* based on the current tree T_j using one of four moves with specified proposal probabilities: growing a terminal node (0.25), pruning a pair of terminal nodes (0.25), changing a non-terminal rule (0.40), or swapping a rule between a parent and child node (0.10). The acceptance probability depends on the prior probabilities of the trees and the integrated likelihoods.

2. Draw the terminal node parameters $\boldsymbol{M}_j \sim p(\boldsymbol{M}_j \mid T_j, \boldsymbol{R}_j, \sigma)$. Given the new tree structure T_j (if accepted, otherwise the old T_j) and the partial residuals $\boldsymbol{R}_j$, the parameters $\mu_{ij} \in \boldsymbol{M}_j$ associated with each terminal node are drawn independently from their respective full conditionals.

By integrating out $\boldsymbol{M}_j$ in step 1, complexities associated with reversible jump MCMC for changes in the number of terminal nodes are avoided. The draw of $\boldsymbol{M}_j$ in step 2 is straightforward and enables the calculation of subsequent residuals for the next tree update. Chipman et al. (2010) shows that this backfitting MCMC algorithm mixes much better than single-tree MCMC approaches, such as the one described in Section 2.1.4.

This stochastic backfitting scheme is also the blueprint for the MCMC algorithms developed later in the thesis, where the decision trees T_j are replaced by Voronoi tessellations and analogous local updates are used for the tessellation structures and their associated cell parameters.

However, while this original MCMC algorithm is often effective, Carnegie (2019) suggests using multiple chains (for example, between 10 and 12 chains) to help encourage convergence. More recently, alternative MCMC proposals have been developed. For example, Lakshminarayanan et al. (2015) introduced a particle Gibbs sampler that uses sequential Monte Carlo to construct full-tree proposals. He et al. (2019) proposed a recursive tree-growing proposal called XBART where split variables and cut-points are sampled stochastically using marginal likelihoods at each step. These advanced samplers aim to bias proposals towards trees with better posterior support, potentially improving mixing performance, especially for complex regression surfaces or high-dimensional covariates.

2.1.6 Posterior inference statistics

The Bayesian backfitting MCMC algorithm generates a sequence of draws of the model parameters $((T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m), \sigma)$ that converges in distribution to their joint posterior

$$p((T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m), \sigma \mid \boldsymbol{y}_{\mathbf{train}}).$$

Consequently, the induced sequence of sum-of-trees functions, where each function is denoted by

$$f^*(\cdot) = \sum_{j=1}^{m} g(\cdot; T_j^*, \boldsymbol{M}_j^*), \tag{2.6}$$

(based on a set of drawn tree structures T_j^* and parameters $\boldsymbol{M}_j^*$) converges to $p(f \mid \boldsymbol{y}_{\mathbf{train}})$, the posterior distribution of the true underlying function $f(\cdot)$. After a suitable burn-in period, the

sequence of K function draws, say $f_1^*(\cdot), \ldots, f_K^*(\cdot)$, can be treated as approximate samples from $p(f \mid \boldsymbol{y}_{\mathbf{train}})$.

This collection of posterior draws allows for estimating a range of Bayesian inferential quantities, enabling comprehensive statistical analysis of the underlying model. To estimate $f(\boldsymbol{x})$ or predict $\boldsymbol{Y}$ at a particular input $\boldsymbol{x}$ (either in-sample or out-of-sample), a natural choice is the average of the post-burn-in draws,

$$\hat{f}(\boldsymbol{x}) = \frac{1}{K} \sum_{k=1}^{K} f_k^*(\boldsymbol{x}), \tag{2.7}$$

which approximates the posterior mean $E[f(\boldsymbol{x}) \mid \boldsymbol{y}_{\mathbf{train}}]$. Alternatively, the median of $f_1^*(\boldsymbol{x}), \ldots, f_K^*(\boldsymbol{x})$ can approximate the posterior median of $f(\boldsymbol{x})$.

Posterior uncertainty about $f(\boldsymbol{x})$ is captured by the variability in these draws. For instance, a $(1-\alpha)\%$ posterior credible interval for $f(\boldsymbol{x})$ can be formed using the $\alpha/2$ and $1-\alpha/2$ quantiles of the $f_k^*(\boldsymbol{x})$ sample. These intervals tend to widen for $\boldsymbol{x}$ values far from the observed data, reflecting increased uncertainty.

2.2 Nominal categorical covariates

Most predictive models inherently require numerical input variables, creating a fundamental challenge when incorporating nominal categorical features into "black-box" algorithms. Tree-based methods, in particular, struggle with high-cardinality nominal covariates, as their splitting rules are natively designed for continuous or ordinal spaces. Historically, practitioners have relied on basic transformations to force categorical data into a numerical format. However, these foundational approaches often impose severe structural compromises on the model, artificially altering the geometry of the covariate space or introducing false mathematical relationships between independent categories.

As applications have grown more complex, the need for robust handling of categorical data has driven the development of more sophisticated, data-driven transformations. These intermediate methods attempt to map discrete levels to continuous values based on summary statistics of the target variable, offering a much more compact representation. Yet, they remain highly susceptible to overfitting and require strict regularisation to prevent the model from memorising statistical

noise.

Consequently, modern research has shifted towards directly incorporating the hierarchical or network structure of categorical levels within the algorithm itself. By utilising structured priors to capture inherent relationships between levels, modern Bayesian approaches avoid the pitfalls of external pre-processing. Specifically, recent innovations like flexBART (Deshpande, 2025) extend the classical BART framework by applying alternative priors on splitting rules for nominal covariates, enabling highly effective and computationally efficient tree structures.

2.2.1 Encoding techniques

Encoding techniques are a common pre-processing step used to convert qualitative, categorical data into a suitable quantitative format. The choice of encoding strategy is not trivial, as it can significantly influence model performance by altering the information available to the algorithm. Here, we define the most common encoding techniques for incorporating categorical variables into BART and other "black-box" methods.

Given a categorical predictor with L possible levels, **one-hot encoding** transforms it into L binary indicator variables, each taking the value 1 if the observation belongs to the corresponding category and 0 otherwise. This representation avoids imposing an artificial ordering on nominal covariates, but it increases the dimensionality of the covariate space.

Count encoding replaces each category with its frequency of occurrence in the dataset. While this efficiently captures a category's prevalence, it risks data leakage. Data leakage is when information from outside the training set is used to create the model, leading to overly optimistic performance evaluations. With count encoding, this occurs if frequencies are calculated using the entire dataset, as information from the test set would then improperly influence the training process. A further weakness is that it can assign the same value to different categories that happen to share the same frequency.

Target encoding is a supervised method that represents categories using information from the target variable, $\boldsymbol{y}$. It works by replacing each category with a measure of its conditional expectation, such as the mean of $\boldsymbol{y}$ for that specific category. The main advantage of this technique is that it converts each categorical feature into a single numerical variable, thereby not increasing the dataset's dimensionality. However, it imposes an artificial order on the categories.

An example of this is Bayesian encoding, discussed in Micci-Barreca (2001), a technique for handling categorical features derived from Empirical Bayes methods. Its core function is to combine a prior distribution with a posterior distribution to estimate the expected value of the dependent variable. For any given category level l, the feature is encoded as an estimate of the conditional expectation $E(\boldsymbol{y} \mid \boldsymbol{x} = l)$. The general formula for this estimation is a weighted average of the prior and posterior values,

$$\phi(l) = B_l \hat{\mu}_l + (1 - B_l)\mu \tag{2.8}$$

where μ is the prior, estimated as the mean of the target variable $\boldsymbol{y}$. The term $\hat{\mu}_l$ is the posterior, estimated as the mean of $\boldsymbol{y}$ for observations of level l. The balance between these is controlled by a shrinkage factor B_l, which is constrained between zero and one.

The simplest variant is **mean encoding**, which uses the conditional expectation directly without any prior regularisation. In the context of the general formula, this approach effectively sets the shrinkage factor B_l to one, thereby considering only the posterior value. The posterior is calculated as the target mean for the corresponding level. Consider the subset of the dataset where $x_i = l$ as D_l, with a size of m_l, the mean encoding for level l is expressed purely as the average of the target within that subset.

$$\phi(l) = \frac{1}{m_l} \sum_{i \in D_l} y_i.$$

While straightforward, this method can produce unreliable estimates for categories with few instances, making it prone to overfitting.

To address this unreliability, regularised Bayesian encodings introduce a dynamic shrinkage factor that depends on the number of instances m_l. The **S-shrink Encoder** implements the full logic of Equation 2.8 by defining B_l as a monotonic, S-shaped function of m_l. This ensures that for infrequent categories, where the posterior mean is unreliable, the encoding is 'shrunk' towards the more stable prior mean. The components are defined as

$$\mu = \frac{1}{m} \sum_{i=1}^{m} y_i, \qquad \hat{\mu}_l = \frac{1}{m_l} \sum_{i \in D_l} y_i, \qquad B_l = \frac{1}{1 + \exp\left(-\frac{m_l - S_1}{S_2}\right)}$$

where the hyperparameters S_1 and S_2 control the threshold and steepness of the shrinkage.

Similarly, the **M-estimate Encoder** offers a simpler regularisation mechanism, defining its

shrinkage factor as

$$\mu = \frac{1}{m}\sum_{i=1}^{m} y_i, \qquad \hat{\mu}_l = \frac{1}{m_l}\sum_{i \in D_l} y_i, \qquad B_l = \frac{m_l}{m_l + M}$$

where the hyperparameter M serves as a threshold. When the instance count m_l is small relative to M, the shrinkage factor B_l is also small, giving more weight to the prior value. Both of these methods provide a more robust encoding than the simple mean by systematically reducing the influence of noisy estimates from rare categories.

Other advanced target encoding techniques include the James-Stein encoder (James and Stein, 1960), which improves stability by applying a classical shrinkage formula to pull category means towards a central value; the quantile encoder (Mougan et al., 2021), which offers robustness to outliers by using quantiles of the target variable rather than its mean; and generalised linear mixed model encoding (Pargent et al., 2022), which provides a principled, model-based form of regularisation by treating categories as random effects.

The optimal encoding technique is not universal but depends on the specific characteristics of each covariate. Therefore, a combination of methods is often employed within a single analysis, applying the most suitable approach to each categorical variable. A detailed comparison of these encoders across a variety of "black-box" models can be found in Zhu et al. (2024).

Permutation encoding, introduced in Chapter 5, is a direct competitor to these methods when the underlying algorithm uses a Bayesian MCMC approach, and aims to overcome many of the limitations mentioned here. We use some of these encoding techniques to compare to permutation encoding on a range of datasets.

2.2.2 FlexBART

FlexBART directly incorporates categorical variables by allowing decision nodes to partition the input space based on arbitrary subsets of the possible levels. Formally, if a node v splits on a categorical variable X_t with domain $\mathcal{X}_t = \{1, \ldots, L_t\}$, it selects a subset $A_v \subset \mathcal{X}_j$ and applies the rule

$$x_i \in A_v \Rightarrow \text{left child of } v, \quad x_i \notin A_v \Rightarrow \text{right child of } v.$$

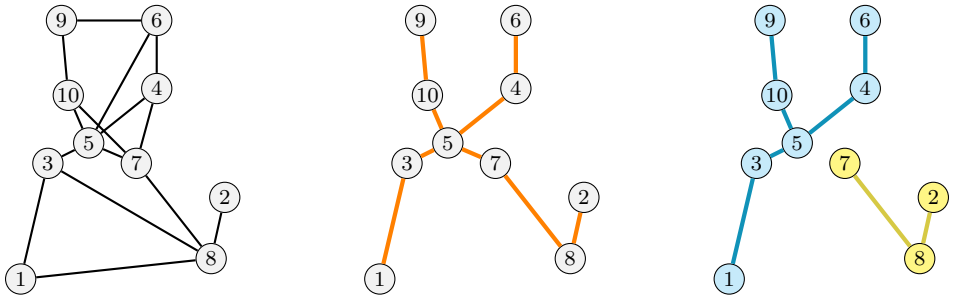
This approach permits multiple levels to be grouped into a single decision, allowing partial pooling across categories. The number of possible splits for a categorical feature with L_t levels (excluding the empty set and the full set) is given by

$$2^{L_t} - 2.$$

This dramatically increases the potential complexity of the partition space for covariates with many classes, making regularisation essential to avoid overfitting.

To determine the set A_v, a network G is formed whose vertices are the levels of X_t and whose edges are drawn between levels. Let G^* be a subgraph of G such that the edges between vertices represent the possibility of both levels being in A_v. A process for partitioning the connected subgraph G^* into two connected components is used to give the splitting rule. For example, Figure 2.2a is an example of the connected subgraph G^* with edges between available levels that can be in the same group, and Figure 2.2c is the partition of the network into two subgroups.

One popular method for partitioning the network from network analysis (Li et al., 2022) draws on the Fiedler vector, the eigenvector associated with the smallest non-zero eigenvalue of the network's Laplacian matrix. The vertices corresponding to positive entries of the Fiedler vector form a connected component of the original network (Fiedler, 1973).



a) Network of classes b) Spanning tree c) One edge removed

Figure 2.2: Partitioning the class set into two groups using networks.

An alternative approach, inspired by spatial clustering (Teixeira et al., 2019; Luo et al., 2021), involves partitioning the network defined by the covariate's class into connected components by removing edges from a randomly drawn spanning tree, as illustrated in Figure 2.2b. This can be achieved using the algorithm introduced in (Wilson, 1996), which generates uniform spanning trees

via loop-erased random walks. After drawing a spanning tree, a partition can be formed by either:

1. Selecting a uniformly random edge from the spanning tree and removing it, resulting in two connected subgraphs.

2. Selecting an edge with probability proportional to the size of the smaller resulting subgraph, promoting more balanced partitions. Formally, if an edge e_i separates subgraphs G_1 and G_2 with sizes n_1 and n_2, the probability of selecting this edge can be given by

$$P(e_i \text{ is selected}) = \frac{\min(n_1, n_2)}{\sum_j \min(n_j, n'_j)},$$

   where the denominator sums over all possible edges in the spanning tree.

3. Computing the Fiedler partition of the uniformly drawn spanning tree. To divide the network, you simply look at these assigned scores and group all the nodes with positive numbers into one community and all the nodes with negative numbers into the other, which naturally reveals the two most loosely connected sub-groups within the data.

These methods introduce a useful way to handle categorical variables without the need for exhaustive search over the full set of subsets, providing more scalable tree structures for complex data. This structured approach to tree construction and regularisation allows flexBART to efficiently incorporate nominal categorical features, while maintaining computational tractability, making it well-suited to high-dimensional, mixed-type data.

2.3 Binary and multi-class classification problems

For classification methods, early approaches were rooted in classical statistics, with linear discriminant analysis (LDA) (Fisher, 1936) and logistic regression (Cox, 1959) providing some of the first formal frameworks for binary and multi-class classification. These models assume linear decision boundaries and are based on strong parametric assumptions, making them effective for well-structured, low-dimensional data.

As computational power improved, decision trees emerged as a more flexible alternative, capable of modelling non-linear relationships by recursively partitioning the input space (Breiman

et al., 1984). At the same time, Support Vector Machines (SVMs) introduced maximum margin classification, providing a powerful framework for multi-dimensional, sparse data. Crucially, they leveraged the kernel trick, a mathematical shortcut that implicitly maps data into a higher-dimensional space where non-linear patterns become linearly separable, allowing the algorithm to efficiently draw decision boundaries without ever explicitly calculating the new coordinates (Cortes and Vapnik, 1995)

Bliss (1934) introduced the probit link, which connects the probability of success to predictors through the inverse cumulative distribution function (CDF) of the standard normal distribution, and later Albert and Chib (1993) introduced data augmentation techniques to allow Bayesian models to use a latent variable. These techniques allow BART to adapt to classification tasks.

2.3.1 Binary classification

Data augmentation (DA) is a foundational technique in Bayesian computation for models with latent structures, particularly in binary classification. The core idea is to introduce unobserved variables that simplify the conditional distributions of a model's parameters, enabling efficient Gibbs sampling.

For example, in the Bayesian probit model, the observed binary outcome $Y \in \{0, 1\}$ is linked to an underlying continuous latent variable Z, such that $Y = 1$ if $Z > 0$ and $Y = 0$ otherwise, with $Z \sim \mathcal{N}(f(\boldsymbol{x}), 1)$. This augmentation transforms the problem into sampling from conjugate Gaussian distributions, greatly improving computational tractability and mixing properties of the Markov chain (Tanner and Wong, 1987).

Zhang and Härdle (2010) extended BART to binary classification by using a probit link function and basic data augmentation techniques. For binary outcomes $Y \in \{0, 1\}$, the model specifies

$$P(Y = 1 \mid \boldsymbol{x}) = \Phi\left(f(\boldsymbol{x})\right),$$

where $\Phi(\cdot)$ denotes the CDF of the standard normal distribution, and $f(\boldsymbol{x})$ is the sum-of-trees model

$$f(\boldsymbol{x}) = \sum_{j=1}^{m} g(\boldsymbol{x}; T_j, \boldsymbol{M}_j).$$

To facilitate posterior computation, the latent variables $\boldsymbol{Z} = \{Z_1, \ldots, Z_n\}$ such that

$$Z = f(\boldsymbol{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1),$$

and the model sets $\sigma = 1$ to solve identifiability issues. This approach allows Gibbs sampling to draw from the posterior of f, treating z_i as observed in each iteration. The Gibbs sampler consists of drawing n successive draws of z_i (for $i = 1, ..., n$, where n is the number of observations in training set) from

$$p(\boldsymbol{Z} \mid (T_1, M_1), (T_2, M_2), ..., (T_m, M_m), \boldsymbol{y_{\text{train}}})$$

followed by m successive draws of (T_j, M_j) for $j = 1, ..., m$ from

$$p((T_j, M_j) \mid \{T_{j'}, M_{j'}\}_{j \neq j'}, \boldsymbol{Z}, \boldsymbol{y_{\text{train}}})$$

using the BART model.

Let $\hat{z}_i = \sum_{j=1}^{m} g(x_i; T_j, M_j)$ denote the fitted value for observation i from the sum of trees. Then, z_i can be independently generated from truncated normal distributions

$$z_i \sim \begin{cases} \mathcal{TN}_{(0,\infty)}(\hat{z}_i, 1) & \text{if } y_i = 1 \\ \mathcal{TN}_{(-\infty,0)}(\hat{z}_i, 1) & \text{if } y_i = 0, \end{cases}$$

where $\mathcal{TN}$ denotes a truncated normal distribution, which in this specific case restricts the underlying normal distribution with mean $\hat{z}_i$ and unit variance to strictly positive values $(0, \infty)$ when the observed outcome y_i is 1, and to strictly negative values $(-\infty, 0)$ when y_i is 0, discarding any values outside these boundaries and proportionally rescaling the remaining probability mass.

A prediction for $\boldsymbol{x}$ is determined by

$$y_i = \begin{cases} 1 & \text{if } z_i > 0, \\ 0 & \text{otherwise} \end{cases}$$

and given the k^{th} posterior draw, $\left(T_j^{(k)}, M_j^{(k)}\right)$ for $j = 1, \ldots, m$, the predicted probability that an

observation with input variables $\boldsymbol{x}$ belongs to class 1 is

$$P(Y = 1 \mid \boldsymbol{x}, (T_1, M_1), \ldots, (T_m, M_m), \sigma) = \Phi\left\{\sum_{j=1}^{m} g(\boldsymbol{x}, T_j^{(k)}, M_j^{(k)})\right\}.$$

This latent variable approach produces naturally calibrated probabilities that capture high-dimensional interactions without manual feature engineering. Unlike traditional classifiers, the Bayesian structure allows for the construction of credible intervals, providing a rigorous measure of predictive confidence.

2.3.2 Ordinal classification

This method can be extended to classification problems where Y is an ordered categorical variable, that is, Y has an inherent order, such as satisfactory ratings or letter grades. Lee and Hwang (2024) introduces the ordered probit BART (OPBART) model for ordered categorical outcomes by generalising the data augmentation and using the Gibbs sampling scheme from binary classification.

Suppose Y is an observed ordered categorical variable that can take one of L ordered categories. A latent variable $\boldsymbol{z}$ is introduced and is defined by

$$y_i = l \quad \Longleftrightarrow \quad u_{l-1} < z_i \leq u_l,$$

where $u_0 = -\infty$ and $u_L = \infty$. The restriction $u_1 = 0$ is imposed to address the identifiability issue. Then, the probabilities associated with each value of Y given the covariates $\boldsymbol{x}$ are defined by

$$P(Y = l \mid \boldsymbol{x}) = \Phi(u_l - f(\boldsymbol{x})) - \Phi(u_{l-1} - f(\boldsymbol{x})), \quad l = 1, \ldots, L$$

where Φ denotes the CDF of the standard normal distribution. The values from the joint posterior distribution $(T_1, M_1), \ldots, (T_m, M_m), \boldsymbol{u}, \boldsymbol{z} \mid \boldsymbol{y}$ can be sampled using a Gibbs sampling procedure. Then, u_l can be sampled from the full conditional distributions

$$u_j \mid \boldsymbol{z}, , \boldsymbol{u}^{(-j)}, \boldsymbol{y} \sim \text{Unif}\left(\max\left(\max(z_i : y_i = j), u_{j-1}\right), \min\left(\min(z_i : y_i = j+1), u_{j+1}\right)\right),$$

where $\boldsymbol{u}^{(-j)}$ denotes all u values except u_j, and again $\boldsymbol{z}$ is sampled from a truncated normal

distribution

$$\boldsymbol{z} \mid \boldsymbol{u}, y = l \sim \mathcal{N}_{(u_{l-1}, u_l)}(f(\boldsymbol{x}), 1).$$

$f(\boldsymbol{x})$ is estimated by treating $\boldsymbol{z}$ as the continuous outcome variable.

To determine the class for a new observation with covariates $\boldsymbol{x}$, the model estimates the posterior predictive probability that it belongs to each potential class l. This is achieved by averaging the results across K samples taken from the posterior distribution. For each of the K samples, the model provides a fitted latent value, $G^{(k)}(\boldsymbol{x})$, which is defined as the sum of the predictions from all trees in the ensemble for that specific draw, $G^{(k)}(\boldsymbol{x}) = \sum_{j=1}^{m} g(\boldsymbol{x}; T_j^{(k)}, M_j^{(k)})$. The probability for a single draw is then found by calculating the area under the standard normal curve that falls between two class-specific cut-points. This process is formally expressed by averaging these individual probabilities over all K posterior samples:

$$\hat{p}_l(\boldsymbol{x}) = \frac{1}{K} \sum_{k=1}^{K} \left\{ \Phi\left(u_l - G^{(k)}(\boldsymbol{x})\right) - \Phi\left(u_{l-1} - G^{(k)}(\boldsymbol{x})\right) \right\}.$$

Once this estimated probability, $\hat{p}_l(\boldsymbol{x})$, has been computed for each class, the final classification is made by simply assigning the observation to the class with the highest posterior probability.

The ordinal classification framework in BART extends the latent variable approach by partitioning the underlying Gaussian space into ordered intervals using a series of thresholds. This structure allows the model to preserve the natural ranking of categories while capturing the non-linear interactions that drive transitions between adjacent classes.

By providing full posterior distributions for both the thresholds and class probabilities, BART offers a rigorous way to quantify uncertainty in ordered rankings. This makes it a superior alternative to simpler methods that ignore the inherent order or require manual threshold tuning.

2.3.3 Nominal classification

Extending the data augmentation framework from binary and ordinal models to the nominal classification setting introduces significant new computational challenges. While the foundational principle of using latent variables to reframe the problem in terms of regression remains, the model must now accommodate a vector of latent utilities rather than a single one. This shift from a univari-

ate to a multivariate latent structure necessitates a more complex approach to handle statistical identifiability.

For a categorical outcome Y that takes values in the set $\{0, \ldots, L\}$, there are $L+1$ distinct categories. The general latent variable framework for multinomial models posits that Y is the observable result of an unobserved vector of latent utilities, $\boldsymbol{Z} = (Z_0, \ldots, Z_L)^T \in \mathbb{R}^{L+1}$. The outcome is determined by selecting the category with the highest utility, such that $Y = \arg\max_l Z_l$; that is, $Y = l$ if $Z_l > Z_k$ for all $k \neq l$.

A key consideration in this framework is model identifiability, as the outcome Y remains unchanged by either a uniform translation or a positive scaling of the latent vector $\boldsymbol{Z}$. Without loss of generality, category 0 is designated as the reference, achieving normalisation by defining the outcome as a function of a reduced set of latent variables, $\boldsymbol{W} = (W_1, \ldots, W_L)^T \in \mathbb{R}^L$. These are defined by the difference relative to the reference category, $W_l = Z_l - Z_0$. Consequently, the decision rule, expressed in terms of $\boldsymbol{W}_i$, becomes $y_i = l$ if $W_{il} > 0$ and $W_{il} > W_{ik}$ for all $k \neq l$, and $y_i = 0$ if all elements of $\boldsymbol{W}_i$ are negative. Furthermore, by definition, the outcome Y is invariant to the multiplication of $\boldsymbol{W}$ by any positive constant. To address this, a normalisation for scale is imposed by placing a constraint on the covariance matrix, $\boldsymbol{\Sigma}$. Following the approach of Burgette and Nordheim (2012), this is achieved by ensuring that the trace of the covariance matrix is equal to its dimension, L, that is, $\text{trace}(\boldsymbol{\Sigma}) = L$.

This constraint, however, removes the conjugacy of standard prior distributions, specifically, the inverse-Wishart prior,

$$\boldsymbol{\Sigma} \sim \mathcal{W}^{-1}(\nu, \Psi),$$

which is typically applied to unconstrained covariance matrices, rendering the standard Gibbs sampler computationally inefficient. The MPBART algorithm, as proposed by Kindo et al. (2016), offers a strategic solution to this problem by employing Marginalised Data Augmentation (MDA). Standard Data Augmentation (DA) traditionally introduces missing data to simplify the posterior. In contrast, Liu and Wu (1999) established key theoretical results concerning the convergence rate of the MDA algorithm, which expands the parameter space. In parallel, Meng and Dyk (1999) introduced MDA in the context of two augmentation schemes, grouping and collapsing, which yield the same final distribution.

The fundamental idea distinguishing MDA from standard DA is the mathematical expansion of the original model. Instead of working with the original parameterisation defined by the probability distribution $f(Y, \phi)$, where f represents the joint density, Y denotes the observed data, and ϕ represents the original model parameters or latent variables, MDA introduces an overparameterised distribution $f(Y, \phi, \alpha)$. This new "expansion parameter" α typically corresponds to a mathematical transformation of the observed data, the original parameters, or both.

The initial step of the algorithm jointly samples from the distribution $p(\tilde{\boldsymbol{W}}, \alpha_1^2 \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, Y)$, where $\boldsymbol{\mu}$ represents the conditional mean vector of the latent variables as determined by the current sum-of-trees ensemble. This process begins by drawing the expansion parameter α_1^2 from its prior distribution, $p(\alpha_1^2 \mid \boldsymbol{\Sigma})$. Specifically, α_1^2 is sampled such that

$$\alpha_1^2 / \text{trace}(\Psi \boldsymbol{\Sigma}^{-1})$$

follows an inverse-chi-squared distribution with νL degrees of freedom.

Once this parameter is sampled, the algorithm computes $\tilde{\boldsymbol{W}} = \alpha_1 \boldsymbol{W}$. The matrix $\boldsymbol{W}$ is itself drawn from its conditional distribution $p(\boldsymbol{W} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, Y)$. Each element of $\boldsymbol{W}$ is sampled from a truncated normal distribution parametrised by the current mean and variance,

$$W_{il} \sim \begin{cases} \mathcal{TN}_{(-\infty, 0]}(\mu_{il}, \sigma_{il}^2) & \text{if } y_i = 0, \\ \mathcal{TN}_{[\max(0, \max_{k \neq l} W_{ik}), \infty)}(\mu_{il}, \sigma_{il}^2) & \text{if } y_i = l, \\ \mathcal{TN}_{(-\infty, W_{ik}]}(\mu_{il}, \sigma_{il}^2) & \text{if } y_i = k \text{ for } k \neq l \text{ and } k \neq 0. \end{cases}$$

This truncation is required because the observed outcome y_i places an interval constraint on the latent variable $\boldsymbol{W}_i$; for example, if y_i equals the reference level 0, the corresponding W_{il} values are right-truncated at 0.

The second step of the process updates the tree ensembles and their terminal node parameters via a Bayesian backfitting MCMC algorithm, operating temporarily within the expanded latent space. For each of the m trees, the algorithm isolates its specific contribution by calculating the partial residuals. This is achieved by subtracting the current predictions of all other trees from the

expanded latent utilities, such that

$$\tilde{\boldsymbol{R}}_j = \tilde{\boldsymbol{W}}^{(k)} - \sum_{t \neq j} G(\boldsymbol{x}; T_t, \tilde{\boldsymbol{M}}_t).$$

Treating these partial residuals as the effective response variable, the algorithm proposes a modification to the current tree structure T_j using standard perturbation moves, such as growing a new terminal node, pruning an existing one, or changing a split rule. This proposed structure is then evaluated using a Metropolis-Hastings acceptance ratio. Because conjugate regularising priors are employed, the terminal node parameters can be analytically integrated out to calculate the marginal likelihood of the proposed tree, greatly improving the efficiency of the stochastic search.

Once the tree structure is updated, its associated terminal node parameters, $\tilde{\boldsymbol{M}}_j$, are drawn directly from their multivariate normal full conditional distribution. This backfitting cycle is repeated sequentially for all m trees. Finally, the algorithm aggregates the updated trees to compute the total unnormalised mean vector, $\tilde{\boldsymbol{\mu}} = G(\boldsymbol{x}; \{T^{(k)}, \tilde{\boldsymbol{M}}^{(k)}\})$, and immediately projects it back into the stable, identified parameter space by calculating the normalised mean, $\boldsymbol{\mu} = \tilde{\boldsymbol{\mu}}/\alpha_1^{(k)}$.

The final step of the algorithm updates the covariance matrix through a straightforward sequence of calculations. First, it finds the model's residuals by subtracting the sum-of-trees predictions from the latent variables. Using these residuals, it then draws a temporary, unconstrained covariance matrix, $\tilde{\boldsymbol{\Sigma}}$, from an inverse-Wishart distribution. However, this newly sampled matrix will not naturally meet the strict requirement that its trace must equal L. To correct this and maintain model identifiability, the algorithm calculates a specific scaling factor, α_3, by dividing the trace of this temporary matrix by L. Finally, it divides the temporary matrix by this scaling factor to generate the correctly constrained covariance matrix for the next iteration,

$$\boldsymbol{\Sigma}^{(k)} = \frac{\tilde{\boldsymbol{\Sigma}}}{(\alpha_3^{(k)})^2}.$$

This rescaling step guarantees that the mathematical boundaries of the model are strictly enforced before the cycle repeats.

Algorithm 1 is an adapted version from the appendices of Xu et al. (2025) and gives the steps of nominal classification using BART.

Algorithm 1: Algorithm for Nominal Classification BART

Data: Initialise $\boldsymbol{W}^{(0)}, \boldsymbol{\Sigma}^{(0)}$, and tree ensembles $\{T, \boldsymbol{M}\}$.

for $k = 1, \ldots, max\_iter$ **do**

Step 1: Update latent variables $\boldsymbol{W}$ and scaling parameter α_1

Draw $(\alpha_1^{(k)})^2 \sim \text{trace}(\Psi(\boldsymbol{\Sigma}^{(k-1)})^{-1})/\chi^2_{\nu L}$.

for $i = 1, \ldots, n$ **do**

for $l = 1, \ldots, L$ **do**

Draw $W_{il}^{(k)} \sim \mathcal{TN}(\mu_{il}, \sigma_{il}^2)$,

where $\boldsymbol{W}_{i(-l)} = (W_{i1}^{(k)}, \ldots, W_{i(l-1)}^{(k)}, W_{i(l+1)}^{(k-1)}, \ldots, W_{iL}^{(k-1)})$.

Rescale: $\tilde{\boldsymbol{W}}^{(k)} = \alpha_1^{(k)} \boldsymbol{W}^{(k)}$.

Step 2: Update tree ensembles

Draw $\{T^{(k)}, \tilde{\boldsymbol{M}}^{(k)}\} \sim p(T, \tilde{\boldsymbol{M}} \mid \tilde{\boldsymbol{W}}^{(k)}, \boldsymbol{\Sigma}^{(k-1)})$;

Set $\tilde{\boldsymbol{\mu}}^{(k)} = G(\boldsymbol{x}; \{T^{(k)}, \tilde{\boldsymbol{M}}^{(k)}\})$ and compute the normalised mean $\boldsymbol{\mu}^{(k)} = \tilde{\boldsymbol{\mu}}^{(k)}/\alpha_1^{(k)}$.

Set the normalised terminal node parameters: $\boldsymbol{M}^{(k)} = \tilde{\boldsymbol{M}}^{(k)}/\alpha_1^{(k)}$.

Step 3: Update covariance matrix $\boldsymbol{\Sigma}$ and scaling factor α_3

Compute residuals $\boldsymbol{\epsilon} = \boldsymbol{W}^{(k)} - \boldsymbol{\mu}^{(k)}$.

Draw $\tilde{\boldsymbol{\Sigma}} \sim \mathcal{W}^{-1}\big(n + \nu, \Psi + \sum_{i=1}^n \boldsymbol{\epsilon}_i \boldsymbol{\epsilon}_i^\top\big)$.

Set $(\alpha_3^{(k)})^2 = \text{trace}(\tilde{\boldsymbol{\Sigma}})/L$.

Rescale $\boldsymbol{\Sigma}^{(k)} = \tilde{\boldsymbol{\Sigma}}/(\alpha_3^{(k)})^2$.

Prediction: For new $\boldsymbol{x}^*$, draw

$$\boldsymbol{W}^* \sim \mathcal{N}(G(\boldsymbol{x}^*; \{T^{(\text{max\_iter})}, \boldsymbol{M}^{(\text{max\_iter})}\}), \boldsymbol{\Sigma}^{(\text{max\_iter})}),$$

and classify by $\hat{y}^* = \arg\max_{0 \le l \le L} W_l^*(\boldsymbol{x}^*)$, where $W_0^* = 0$.

In the first step of the algorithm, the latent variable $\tilde{\boldsymbol{W}}$ is defined as a scaled version of $\boldsymbol{W}$ through the transformation $\tilde{\boldsymbol{W}} = \alpha_1 \boldsymbol{W}$. Fitting the BART model using this scaled variable is analogous to sampling the model parameters within an un-normalised space. A significant challenge arises, however, because conducting a stochastic search can be inefficient when the quantity being fitted, $\tilde{\boldsymbol{W}}$, is unstable. This means, fitting the decision trees to the expanded variable $\tilde{\boldsymbol{W}}$ is highly inefficient because its scale randomly fluctuates at every single iteration, making it nearly impossible for the MCMC sampler to smoothly converge on the true underlying patterns.

Recognising the potential instability of the MPBART sampler, Xu et al. (2025) proposed two new versions of BART for classification, which are designed to improve MCMC mixing by altering the data augmentation scheme. While these methods also create an expanded version of the latent utilities,

$$\tilde{\boldsymbol{W}} = \alpha_1 \boldsymbol{W},$$

the crucial innovation is to perform the BART backfitting update using the original, normalised

latent variables $\boldsymbol{W}$ as the response. This ensures the complex stochastic search for tree structures is conducted in a stable parameter space, yielding an updated mean function $\boldsymbol{\mu} = G(\boldsymbol{X}; \boldsymbol{\theta})$ in which regularising priors can effectively function.

The first of the two proposals introduced by Xu et al. (2025) then reverts to the expanded space only for the mathematically convenient update of the covariance matrix, calculating the residuals as

$$\tilde{\boldsymbol{\epsilon}}_i = \tilde{\boldsymbol{W}}_i - \alpha_1 \boldsymbol{\mu}_i,$$

before re-imposing the identifiability constraint through normalisation. This provides a significant advantage by segregating the tasks, reserving the stable space for the difficult BART trees update.

The second proposal by Xu et al. (2025) is a further simplification that takes a more direct procedure, which mostly removes the explicit data augmentation for the latent variables, instead sampling and operating on the normalised utilities $\boldsymbol{W}$ throughout the BART update to produce $\boldsymbol{\mu} = G(\boldsymbol{X}; \boldsymbol{\theta})$. The augmentation machinery is now confined entirely to the final step, where residuals calculated in the normalised space,

$$\boldsymbol{\epsilon}_i = \boldsymbol{W}_i - \boldsymbol{\mu}_i,$$

are used to update a working, unconstrained covariance matrix, which is then immediately projected back to the constrained space via trace normalisation. The strong performance of this streamlined sampler demonstrates that the key to improved mixing and accuracy is to update BART trees from the more stable constrained space.

In Chapter 6, similar modifications are employed in the MCMC algorithm for the AddiVortes model, demonstrating that this technique using Voronoi tessellations performs competitively against other "black-box" methods, both in simulated and real-world settings.

2.4 Heteroscedasticity

Heteroscedastic models have been developed alongside advances in regression modelling, reflecting the need to account for varying levels of noise across different regions of the input space. Early approaches focussed on weighted least squares, which adjusts for heteroscedasticity by assigning

weights to observations based on estimated error variances, thereby improving the efficiency of the parameter estimates (Aitken, 1936). Generalised least squares allows for more flexible error structures by accommodating correlated and heteroscedastic residuals, whereas generalised linear models (Nelder and Wedderburn, 1972) extend the linear framework to accommodate response variables from distributions other than the Gaussian distribution. The introduction of generalised additive models further broadened the scope of heteroscedastic modelling by capturing nonlinear mean structures while accommodating nonconstant variance (Hastie and Tibshirani, 1986).

More recently, Gaussian process regression (GPR) has become a popular choice for modelling heteroscedastic data, providing a probabilistic framework with fully flexible mean and variance functions (Le et al., 2005). Today, heteroscedastic models are widely used in fields ranging from finance and economics to environmental science and biomedicine, where accounting for varying levels of uncertainty is critical.

To model heteroscedasticity, Pratola et al. (2020) introduced HBART, which extends the standard BART framework by allowing the variance to depend on the predictors,

$$Y = f(\boldsymbol{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, s^2(\boldsymbol{x})).$$

This is achieved by specifying an additional ensemble of regression trees to model the variance, separate from those used to estimate the mean function $f(\boldsymbol{x})$. The variance component is represented through a multiplicative product-of-trees model,

$$s^2(\boldsymbol{x}) = \prod_{j=1}^{m'} h(\boldsymbol{x}; T'_j, \boldsymbol{M}'_j),$$

where each $h(\boldsymbol{x}; T'_j, \boldsymbol{M}'_j)$ corresponds to the contribution of the jth variance tree, parametrised by its structure T'_j and associated leaf parameters $\boldsymbol{M}'_j$. Since $h(\cdot) > 0$, this multiplicative formulation allows the variance to inflate and deflate naturally without the structural risk of sampling negative values, a fundamental limitation of additive models, whilst simultaneously capturing complex interactions between predictors through the product of simple tree-based components.

The inference procedure for HBART closely follows the standard MCMC backfitting algorithm used in BART, but is adapted to jointly estimate both the mean and variance functions. Posterior

sampling proceeds via a modified Gibbs sampler that alternates between updates to the mean and variance ensembles. Specifically, the mean function f is drawn conditional on current estimates of the variance function s and the observed data, and subsequently, the variance function s^2 is updated conditional on the current residuals from the mean model. This iterative scheme allows information to flow between the two components, with improvements in one leading to more accurate estimation in the other.

Priors for the variance trees are generally chosen to mirror those used for the mean trees, but are tuned to encourage smoothness and stability in the estimated noise process. Typically, the leaf parameters of the variance trees are given lognormal priors to maintain positivity and control variability across regions of the predictor space. This Bayesian regularisation prevents overfitting to local fluctuations and ensures that the estimated variance surface remains interpretable.

By jointly modelling both the conditional mean and variance functions, HBART provides a fully nonparametric and probabilistic treatment of heteroscedasticity. It flexibly captures complex patterns of varying noise across the input space and delivers improved uncertainty quantification compared with homoscedastic BART. This makes HBART particularly valuable in applications where the magnitude of uncertainty is itself of interest.

An approach similar to this is used in Chapter 7, where we also adapt the AddiVortes model to the heteroscedastic setting using a second set of Voronoi tessellations. We will see that when using Voronoi tessellations: the variance is generally modelled better and produces better uncertainty quantification than tree-based methods.

2.5 Oblique decision trees

Decision trees produce axis-aligned splits when partitioning the covariate space. There are limitations of axis-aligned splits in capturing complex, high-dimensional decision boundaries and early attempts to address this included the Linear Machine Decision Tree (LMDT), which replaced axis-aligned cuts with hyperplanes optimised using linear discriminant analysis (Bennett, 1992), and Oblique Decision Trees, which used a combination of random perturbations and hill-climbing to find effective oblique splits (Murthy et al., 1994). An example of a partition given by an oblique decision tree is given in Figure 2.3; notice that the partition is more natural then the one in

Figure 2.1.

Oblique decision trees extend traditional axis-aligned decision trees by allowing splits based on linear combinations of features, rather than restricting splits to individual variables. Several authors have found that oblique trees and ensembles often outperform their axis-aligned methods in terms of prediction accuracy. For instance, Bertsimas and Dunn (2017) reported that oblique decision trees can outperform CART by 7%, while Breiman (2001) noted that an oblique version of RF can improve RMSE performance by up to 8.5%. Beyond these empirical results, Cattaneo et al. (2024) recently demonstrated that oblique decision trees and their ensembles can sometimes obtain the same convergence rates as neural networks.

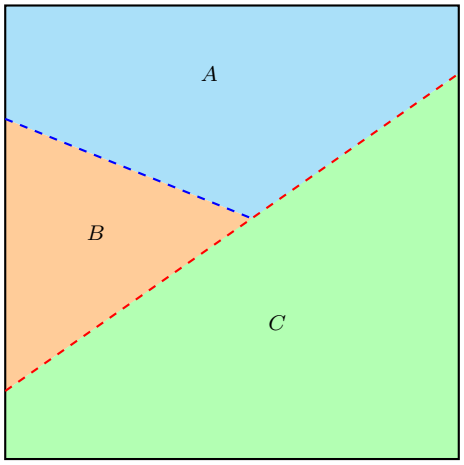


Figure 2.3: Partitioning of the space using an oblique decision tree.

These methods significantly improve model flexibility but suffer from high computational costs and instability due to the increased complexity of the search space. More recent approaches, such as Rotation Forests and Random Rotation Ensembles, have further extended these ideas by incorporating oblique splits within ensemble frameworks to improve accuracy and stability (Rodriguez et al., 2006).

Nguyen et al. (2024) introduces ObliqueBART, which extends the classical BART framework by allowing for decision rules based on linear combinations of predictor variables, rather than the axis-aligned splits used in classical BART. This modification significantly increases the model's flexibility, enabling it to partition the feature space along arbitrary hyperplanes and potentially capturing complex interactions between predictors.

Unlike classical BART, which restricts decision rules to axis-aligned splits of the form

$$X_t < c,$$

ObliqueBART instead allows decision rules of the form

$$\phi^\top X_t < c,$$

where $\phi \in \mathbb{R}^p$ is a vector that specifies a hyperplane. This form of decision rule is considerably more flexible, as it permits splits that depend on multiple covariates simultaneously.

The prior over the direction vector ϕ plays a critical role in controlling model complexity. ObliqueBART adopts a hierarchical spike-and-slab prior to encourage sparsity in the decision vectors, inspired by Breiman (2001) and Tomita et al. (2020). Formally, let $\phi = (\phi_1, \ldots, \phi_p)^\top$ and introduce binary indicators $\gamma = (\gamma_1, \ldots, \gamma_p)$ to denote whether each component of ϕ is active. Given a sparsity parameter $\theta \in [0, 1]$ and for $j = 1, \ldots, p$, draw

$$\gamma_j \mid \theta \sim \text{Bernoulli}(\theta), \quad \phi_j \mid \gamma_j \sim \mathcal{N}(0, 1) \text{ if } \gamma_i = 1,$$

and set $\phi_j = 0$ if $\gamma_i = 0$. The ϕ_j are conditionally independent given γ and then rescale ϕ to have unit norm

$$\phi \leftarrow \frac{\phi}{||\phi||},$$

where $||\cdot||$ denotes the Euclidean norm.

This construction encourages sparse decision vectors with a flexible number of non-zero entries. A further Beta prior is placed on θ to allow the level of sparsity to be learned from the data

$$\theta \sim \text{Beta}(a_\theta, b_\theta),$$

where the hyperparameters a_θ and b_θ are typically chosen to reflect prior beliefs about the degree of sparsity in the data.

Once a direction vector ϕ is drawn, the cutpoint c is sampled uniformly from the range of valid projections determined by the current node's ancestor constraints. That is, if the set of observations

reaching the current node is I_n, then the range of feasible cutpoints is given by

$$\min_{\boldsymbol{x} \in I_n} \phi^\top \boldsymbol{x} < c < \max_{\boldsymbol{x} \in I_n} \phi^\top \boldsymbol{x}.$$

This range is computed by solving two linear programs that respectively minimise and maximise the projection $\phi^\top \boldsymbol{x}$ over the polytope defined by the node's ancestor splits.

Posterior inference for ObliqueBART proceeds via a Metropolis-within-Gibbs sampler. At each iteration, the sampler updates the tree ensemble by proposing small local modifications, such as growing or pruning subtrees, with acceptance probabilities determined by a Metropolis-Hastings ratio. The full conditional posterior for the tree structure now extends Equation 2.5 by including θ and is given by

$$p(T_j \mid \boldsymbol{R}_j, \sigma, \theta) \propto p(T_j \mid \theta) \int p(\boldsymbol{R}_j \mid T_j, \boldsymbol{M}_j, \sigma) p(\boldsymbol{M}_j \mid T_j) d\boldsymbol{M}_j,$$

The MCMC backfitting algorithm is now performed in the same manner as the classical BART algorithm using this posterior distribution for the tree structure.

The flexibility of oblique decision rules comes at a computational cost, as the optimal projection direction and corresponding cutpoint must be recalculated at each decision node. This is mitigated by efficient linear programming algorithms that compute the feasible cutpoint range and by careful tuning of the sparsity-inducing priors.

This adaptation increases the flexibility of partitioning the covariate space, but this has some limitations. In Chapter 3 and 4, we see how we further increase flexibility by using Voronoi tessellations to partition the covariate space and continue to improve predictive performance.

2.6 Other notable developments

In this section, SoftBART is discussed, a popular variant of BART specifically adapted for modelling smooth functions. We also highlight research areas where BART excels, such as variable selection, survival analysis, and causal inference.

Smooth functions

A significant development to the BART framework is the SoftBART model, introduced by Linero and Yang (2018), which replaces standard hard regression trees with the soft regression trees described in İrsoy et al. (2012). This approach replaces the deterministic, hard splits of traditional trees with a smooth probabilistic rule, allowing an observation's influence to be partitioned across both left and right child nodes rather than following a single path. This modification has proven to be a flexible and powerful tool, leading to its adoption in various statistical applications.

For example, Liu et al. (2022) employed SoftBART as their algorithm of choice for addressing non-response in surveys. Demonstrating its scalability, Ran and Bai (2021) constructed a MAP-reduce algorithm to fit the model to massive datasets. Moreover, its predictive accuracy has been shown to be highly competitive; Bai et al. (2022) highlighted SoftBART as a very high-quality competitor to their own spike-and-slab group lasso GAM model.

To generalise a decision tree to a soft regression tree, note that for the j^{th} tree,

$$g(\boldsymbol{x}; T_j, \boldsymbol{M}_j) = \sum_{i=1}^{b_j} \mu_{ij} \phi_{ij}(\boldsymbol{x}; T_j, \boldsymbol{M}_j)$$

where $\phi_{ij}(\boldsymbol{x}; T_j, M_j)$ is the indicator function of the event that $\boldsymbol{x}$ is associated to terminal node μ_{ij} of tree T_j. Notice that $\phi_{ij}(\boldsymbol{x}; T_j, M_j)$ can be expressed in terms of the branch rules as

$$\phi_{ij}(\boldsymbol{x}; T_j, M_j) = \prod_{t \in A(\ell)} I(\boldsymbol{x} \leq C_t)^{L_t} I(\boldsymbol{x} > C_t)^{1-L_t},$$

where $A(\ell)$ is the set of internal nodes that are ancestral to terminal node i of the j^{th} tree and $L_t = 1$ if the path from the root to node i goes left at t and $L_t = 0$ if the path goes right. For example, the terminal node with μ_{1j} in Figure 2.1 has $A(\ell)$ consisting of both of the internal nodes and $L_{\text{root}} = 1$ since the path from the root goes left rather than right.

The indicator functions in this form are not ideal, as they encode a sharp jump from 0 to 1 as $\boldsymbol{x}$ is increased from $\boldsymbol{x} \leq C_t$ to $\boldsymbol{x} > C_t$. Linero and Yang (2018) generalise this by replacing the hard decision rules $I(\boldsymbol{x} \leq C)$ with soft decision rules using the function ψ, which is the CDF of a

symmetric random variable. The modified weights then become

$$\phi_{ij}(\boldsymbol{x}; T_j, M_j) = \prod_{t \in A(\ell)} \psi\left(\frac{\boldsymbol{x} - C_t}{\tau}\right)^{L_t} \left\{1 - \psi\left(\frac{\boldsymbol{x} - C_t}{\tau}\right)\right\}^{1-L_t},$$

where $\psi(x)$ is typically chosen as the logistic function

$$\psi(x) = \frac{1}{1 + \exp(-x)},$$

and each tree T_j has its own bandwidth parameter τ_j, which is assigned the prior

$$\tau_j \sim \exp(\text{scale} = 0.1),$$

allowing for flexibility in the smoothness of each individual tree in the ensemble. This formulation enables SoftBART to smooth over the decision boundaries typically associated with hard decision trees, resulting in more flexible and accurate function approximations.

Sparsity and Variable Selection

The challenge of variable selection, or identifying the most informative predictors from a large set, is a foundational problem in statistical learning. Classical approaches, such as stepwise selection (Zhang, 2016), have long been used but are known to suffer from high variance and issues with statistical inference.

A significant paradigm shift occurred with the introduction of penalised likelihood methods, most notably the LASSO (Least Absolute Shrinkage and Selection Operator) introduced in Tibshirani (1996), building on non-negative garrote (Breiman, 1995). LASSO uses an L_1 penalty to shrink many coefficients to exactly zero, simultaneously performing model regularisation and variable selection. This ability to produce "sparse" models, where only a handful of covariates are deemed active, became a main approach for high-dimensional analysis where the number of predictors p can far exceed the number of observations n (Bühlmann and van de Geer, 2011).

Bayesian tree ensemble variable selection methods are broadly categorised into three distinct groups: firstly, the permutation-based approaches (Bleich et al., 2014; Luo and Daniels, 2024) function by assessing variable importance, usually quantified by the variable inclusion proportion (VIP),

and establishing selection thresholds from null distributions which are constructed by permuting the response variable; secondly, the sparsity-inducing priors (Linero, 2016; Liu et al., 2021) involve modifying the underlying BART prior to explicitly encourage the formation of parsimonious ensembles, with variables then selected based on their marginal posterior inclusion probabilities using the median probability model threshold (Barbieri and Berger, 2004); and thirdly, clustering-based approaches (Ye and Li, 2025a) bypass the need for tuning or permutation by separating the relevant and irrelevant predictors through unsupervised clustering, typically applied to rank-based importance measures, a strategy known for yielding high recall. A review of many of these methods can be found in Ye and Li (2025b).

Here, we describe how BART is adapted for variable selection settings and sparse data to identify relevant covariates by incorporating sparsity-inducing priors. The classical BART model uses a uniform distribution to sample a covariate for a splitting rule. In situations where only a handful of those variables actually impact the outcome, this uniform assumption leads to "noise" variables entering the model purely by chance, which dilutes the importance of the true predictors and reduces the interpretability of the model.

Linero (2016) introduces Dirichlet Additive Regression Trees (DART) that applies a Dirichlet process prior over the covariate inclusion probabilities, effectively encouraging the model to sample the more influential predictors more frequently.

Formally, Linero (2016) replaced the default uniform prior for the splitting probability vector $\boldsymbol{s}$ with a sparsity-inducing Dirichlet prior. This prior is defined as,

$$\boldsymbol{s} \sim \text{Dirichlet}\left(\frac{\alpha}{p}, \ldots, \frac{\alpha}{p}\right),$$

where p is the total number of variables and $\alpha \in \mathbb{R}^+$ is a hyperparameter specifically designed to control the sparsity of the resulting model. When α is set to a small value, the Dirichlet distribution puts most of its density near the corners of the simplex. This mathematically enforces that the resulting vector $\boldsymbol{s}$ will have a few large entries (the important variables) and many entries that are practically zero (the irrelevant noise). This is distinct from standard BART, which effectively assumes α is infinity, resulting in a uniform distribution.

For a fully Bayesian specification, one can further place a hyperprior on α of the form,

$$\frac{\alpha}{\alpha + \rho} \sim \text{Beta}(a, b),$$

where the parameters (a, b, ρ) are chosen to provide a suitable sparsity level for most applications, typically the default $(a, b, \rho) = (0.5, 1, p)$ is used.

DART relies on the fact that the Dirichlet distribution is a conjugate prior to the Multinomial distribution. Since the prior for $\boldsymbol{s}$ is Dirichlet and the variable usage counts follow a Multinomial distribution (representing categorical choices of variables), the posterior distribution for $\boldsymbol{s}$ is mathematically straightforward to derive. When all predictors possess sufficient unique observations such that the possible splitting rules of the form $\{x_j \leq c\}$ versus $\{x_j > c\}$ are not exhausted, such as when dealing exclusively with continuous predictors, the observed counts are simply added to the prior parameters. Consequently, the posterior samples of $\boldsymbol{s}$ are drawn from the following distribution,

$$\boldsymbol{s} \mid \text{Trees} \sim \text{Dirichlet}\left(\frac{\alpha}{p} + k_1, \ldots, \frac{\alpha}{p} + k_p\right),$$

where α is the sparsity hyperparameter, p is the total number of variables, and k_j is the number of times variable j is currently used in the sum-of-trees model.

If the predictors include low cardinality or categorical features that may exhaust the available split values, additional computational steps are required for updating $\boldsymbol{s}$, such as employing data augmentation techniques to sample the posterior of $\boldsymbol{s}$.

Survival Analysis

The statistical analysis of time-to-event data, or survival analysis, was formalised in the mid-20th century, building on early work on life tables and survival estimation (Deevey, 1947; Berkson and Gage, 1950). This was largely driven by challenges in medical and industrial research. Foundational non-parametric methods were established to handle censored data, including the Kaplan-Meier estimator for survival functions (Kaplan and Meier, 1958) and the log-rank test for comparing survival curves (Mantel, 1966). Modern survival analysis is largely built upon the semi-parametric proportional hazards model (Cox, 1972), which incorporates covariate information into the estimation of

survival times.

More recently, the field has expanded to handle more complex scenarios, such as correlated data using frailty models (Vonta, 2009). In recent decades, the rise of high-dimensional data has improved machine learning techniques, leading to powerful predictive methods such as Random Survival Forests (Ishwaran et al., 2008), gradient boosting models (Binder and Schumacher, 2008), and deep learning approaches like DeepSurv (Katzman et al., 2018).

In the survival setting, BART has been adapted to handle right-censored time-to-event data, requiring a fundamentally different likelihood structure from the Gaussian assumption typically used in regression. Instead of modelling a continuous response directly, survival BART estimates the survival function $S(t \mid \boldsymbol{x}) = P(T > t \mid \boldsymbol{x})$ or the hazard function $h(t \mid \boldsymbol{x})$, which capture the probability of survival beyond a given time or the instantaneous risk of an event occurring, respectively.

One common approach is to model the log-hazard function as

$$\log h(t \mid \boldsymbol{x}) = \mu + f(\boldsymbol{x}),$$

where $f(\boldsymbol{x})$ is the sum-of-trees component that captures non-linear effects and interactions, while μ is a baseline hazard term. This structure ensures a positive hazard rate while retaining the flexibility of the additive tree model. Given this formulation, the survival function can be expressed as

$$S(t \mid \boldsymbol{x}) = \exp\left(-\int_0^t h(s \mid \boldsymbol{x})\, ds\right),$$

which can be directly estimated from the fitted trees.

A fully nonparametric survival model is presented in Sparapani et al. (2016) by assuming a discrete-time survival approach. Here, the observed failure and censoring times $t_1, \ldots, t_n$ span the entire support of the response. A sequence of binary indicators, Z_{ij}, is introduced such that

$$Z_{ij} = \begin{cases} 1, & \text{if } Y_i = t_j \\ 0, & \text{otherwise.} \end{cases}$$

This transformation effectively converts the survival analysis into a series of binary regression

problems. $Y_{ij} \sim \text{Bernoulli}(f(t_j, x_i))$ and the model is fit using the BART probit regression model.

Beyond this, the BART framework has been adapted to handle more complex time-to-event data structures. An extension is in the area of recurrent events, such as repeated hospitalisations, where subjects may experience events multiple times. Sparapani et al. (2018) proposed a method that avoids the restrictive assumptions of linearity and proportionality common in traditional recurrent event models.

Further developments have also been applied to applications with competing risks, where subjects are at risk from multiple, mutually exclusive event types. Sparapani et al. (2020) developed a non-parametric BART model to flexibly estimate the cumulative incidence function, directly modelling the complex relationships and non-proportional hazards that are often present in competing risks data.

Causal Inference

One of the most significant advancements of the BART framework has been its application to causal inference, an area of study that has seen extensive and rapid development. In this section, we provide a brief overview of how BART is adapted to estimate causal effects and address the specific challenges of this field.

A primary goal is to estimate causal effects under the Rubin Causal Model (Rubin, 1974), also known as the Potential Outcomes Framework. For each individual i, potential outcomes are defined as $Y_i(1)$ (under treatment) and $Y_i(0)$ (under control). Since only one outcome is observed,

$$Y_i = Z_i Y_i(1) + (1 - Z_i) Y_i(0),$$

the individual causal effect $\tau_i = Y_i(1) - Y_i(0)$ is never observed.

Hill (2011) first proposed BART as a non-parametric solution to estimate the response surfaces $f(z, x) = E[Y(z) \mid X = \boldsymbol{x}]$. A single BART model, $Y_i \sim \mathcal{N}(f(Z_i, \boldsymbol{x}_i), \sigma^2)$, flexibly captures non-linearities and interactions, avoiding the model misspecification bias common in simpler parametric models. Because BART is Bayesian, it provides a full posterior distribution for any counterfactual prediction, allowing for principled credible intervals for causal estimands.

Although this approach naturally captures interactions between the treatment and predictors,

it can be prone to model misspecification in complex settings. Specifically, the single-model BART may struggle to find the correct balance between the prognostic effects of the covariates and the heterogeneous treatment effects, sometimes leading to spurious results or biased estimates of the Conditional Average Treatment Effect (CATE). To improve on this, Hahn et al. (2020) proposed Bayesian Causal Forests. This method models the outcome as $Y_i = \mu(\boldsymbol{x}_i) + \tau(\boldsymbol{x}_i)Z_i + \epsilon_i$, using separate BART priors for the prognostic function $\mu(\boldsymbol{x}_i)$ and the CATE function $\tau(\boldsymbol{x}_i)$. A regularising prior is applied to $\tau(\boldsymbol{x}_i)$, shrinking it towards a simpler form and preventing the model from finding spurious heterogeneity.

BART models within the causal inference framework have demonstrated exceptional performance, particularly in the Atlantic Causal Inference Conference (ACIC) competition (Dorie et al., 2019). These models excel at estimating treatment effects in complex, high-dimensional datasets, thanks to their flexibility and robustness. BART models naturally capture non-linear relationships and interactions without the need for manual specification, a significant advantage for causal inference tasks. In the ACIC competitions, BART's ability to manage model uncertainty and provide accurate interval estimates has set it apart from other methods, consistently delivering high-quality causal estimates.

2.7 Discussion

The BART framework provides a robust and highly accurate approach to regression by combining an ensemble of regularised weak learners to capture non-linear dependencies within multi-dimensional datasets. This foundational success has led to its adaptation across diverse statistical settings, where it frequently outperforms alternative "black-box" models.

BART has several critical features that directly motivate the development of the AddiVortes framework. A primary strength is the use of regularisation priors on the tree structures, which ensures the ensemble remains a sum of simple components and facilitates stable convergence. This will be essential for AddiVortes since regularisation priors on the tessellations limit the influence but also help reduce computational costs of expensive Voronoi tessellations.

Additionally, the Bayesian backfitting MCMC algorithm allows for a principled exploration of the model space, providing a way to quantify uncertainty often absent in frequentist boosting

methods. Later, we see that the AddiVortes framework retains the core strengths of BART by employing a similar Bayesian backfitting algorithm.

Oblique BART has represented one of the first significant steps towards a more flexible partition space by replacing rigid, axis-aligned splits with linear combinations of multiple predictors. This transition allows the model to capture diagonal relationships and rotations that traditional trees struggle to approximate, effectively relaxing the structural constraints of grid-like boundaries. The improvement of this adaptation highlights the need for more natural partitions for the covariate space and motivates the use of the even more flexible Voronoi tessellations.

The original BART framework needs nominal covariates to be encoded using methods such as one-hot encoding or target encoding. Many of these encoding techniques have their own drawbacks and choosing a suitable encoding technique can be challenging. However, flexBART adapted the BART framework to efficiently include nominal covariates within the MCMC backfitting algorithm and its improved predicted performance motivates Chapter 5, which aims to improve the inclusion of categorical covariates in Bayesian predictive models.

BART effectively extends to classification tasks by mapping its ensemble of trees to discrete outcomes through a latent variable framework. In binary settings, a probit link function allows the model to capture complex, non-linear class boundaries while providing naturally calibrated uncertainty estimates. Similarly, for nominal classification with multiple unordered categories, the framework employs a multinomial structure to estimate class membership probabilities without assuming a natural ranking. Similar techniques are effectively employed in Chapter 6 to extend the AddiVortes method to classification tasks.

Heteroscedastic BART extends the standard framework by relaxing the assumption of constant error variance, instead modelling the variance as a function of the predictors. This is achieved by employing a second sum-of-trees ensemble to capture the log-variance, allowing the model to identify regions of the covariate space where the noise level is significantly higher or lower. This extension motivates Chapter 7, which uses a similar approach but instead uses a second set of Voronoi tessellations to model the variance.

BART demonstrates exceptional versatility across diverse statistical challenges, most notably through its capacity for variable selection by identifying influential predictors based on their inclusion frequency within tree splits. The framework further evolves with SoftBART, which replaces

hard binary splits with smoother transitions to better capture continuous functional forms and improve predictive stability. In the context of survival analysis, BART effectively models time-to-event data without the rigid proportional hazards assumptions of traditional methods, while in causal inference, it provides a robust mechanism for estimating heterogeneous treatment effects by uncovering complex interactions within the data.

Overall, BART's strength lies in its ability to manage diverse data challenges, making it a key tool for analysts. Its broad applicability, from regression to survival analysis and causal inference, demonstrates its versatility across domains. The main advantages of BART include its nonparametric foundation, its ability to capture nonlinearities and interactions, its robust uncertainty quantification, and its effective variable selection. These features make BART a reliable choice for exploring complex relationships in data.

Chapter 3

Voronoi Tessellations and partitioning methods

Voronoi tessellations are a powerful and widely used mathematical tool that partitions space into regions based on proximity to a set of given points. In the context of predictive modelling, Voronoi tessellations offer an intuitive way to represent relationships between data points, spatial structures, and decision boundaries. This chapter explores the concept of Voronoi tessellations, their mathematical properties, and their application in predictive modelling tasks. Finally, other tessellation structures such as Delaunay and Laguerre tessellations are introduced.

3.1 Introduction to Voronoi tessellations

Voronoi tessellations date back to Descartes in 1644, but their first notable use was by Dirichlet (1850), who applied two-dimensional and three-dimensional Dirichlet diagrams in research on quadratic forms. These tessellations were later named after Georgy Voronoi, who extended them to the general n-dimensional case in Voronoi (1908). Voronoi tessellations find applications across diverse fields, such as in robotics (for example, Galceran and Carreras, 2013), biology (for example, Bock et al., 2010; Li et al., 2012), geophysics and meteorology, where they are often referred to as Thiessen polygons, and epidemiology for analysing the spatial clustering of disease.

Voronoi tessellations are defined on a metric space $(\mathcal{S}, d)$, where $\mathcal{S}$ is a non-empty set and $d : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ is a distance function satisfying the usual properties of a metric: non-negativity,

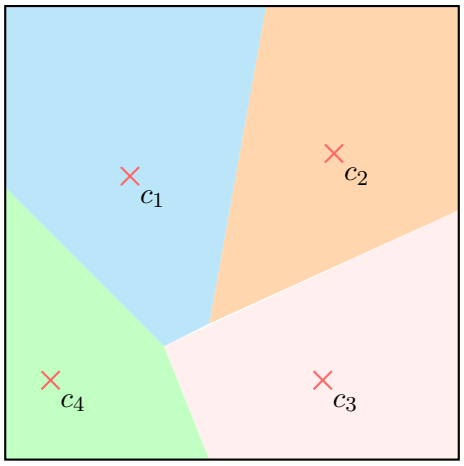


Figure 3.1: Illustration of a tessellation with crosses as centres.

symmetry, the triangle inequality, and that $d(\boldsymbol{x}, \boldsymbol{y}) = 0$ if and only if $\boldsymbol{x} = \boldsymbol{y}$.

Given a finite collection of distinct points $\boldsymbol{M} = \{c_1, c_2, \ldots, c_b\} \subset \mathcal{S}$, referred to as sites or centres, the Voronoi tessellation associated with $\boldsymbol{M}$ is the partition of $\mathcal{S}$ into b regions, called Voronoi cells. For each $i \in \{1, \ldots, b\}$, the Voronoi cell V_i corresponding to centre c_i is defined as

$$V_i = \{\boldsymbol{x} \in \mathcal{S} : d(\boldsymbol{x}, c_i) \leq d(\boldsymbol{x}, c_j) \quad \text{for all} \quad j \neq i\}.$$

Thus, the Voronoi cell V_i consists of all points in $\mathcal{S}$ that are closer to c_i than they are to any other centre. The collection of all Voronoi cells $\{V_i\}_{i=1}^{b}$ provides a complete partition of the space $\mathcal{S}$.

For any two distinct centres c_i and c_j, the set of points equidistant from both centres is called the bisector and is defined by

$$B(c_i, c_j) = \{\boldsymbol{x} \in \mathcal{S} : d(\boldsymbol{x}, c_i) = d(\boldsymbol{x}, c_j)\}.$$

The boundaries of Voronoi cells are subsets of such bisectors, and the overall structure of the tessellation emerges from the arrangement of these boundaries across all pairs of centres. An illustration of a partition produced by a set of 4 centres in the Euclidean space is illustrated in Figure 3.1.

The most common uses of Voronoi tessellations are in the Euclidean space $\mathbb{R}^d$ equipped with the standard Euclidean distance, in which the boundaries between cells are portions of hyperplanes, and each Voronoi cell is a convex polyhedral region.

However, as we see in Section 3.4, the geometric properties of the Voronoi cells depend strongly on the choice of the distance function d. For example, if the distance metric is the Manhattan distance, the boundaries consist of lines with slopes ± 1. Furthermore, if a more general metric is employed, cells may have curved or irregular boundaries.

While Chapter 2 showed that decision trees partition the covariate space into axis-aligned hyperrectangles via recursive splitting, Voronoi tessellations partition space according to proximity to a set of prototype points. Each cell contains all locations that are closer to one centre than to any other, which leads to convex regions whose boundaries can have arbitrary orientation. This geometry allows Voronoi tessellations to approximate rotated or smoothly curved structures more efficiently than a step function built from axis-aligned boxes. In the AddiVortes model developed in Chapter 4, these centre locations are treated as parameters, so the geometric properties of Voronoi partitions directly determine the flexibility and inductive bias of the regression function.

3.2 Computational speed of nearest neighbour search

Determining the nearest neighbour for a set of observations is a foundational task in computational geometry and data analysis. The ability to efficiently assign every observation point to its nearest centre is the critical step in defining the tessellation. As datasets grow in size and dimensionality, the computational cost of this search can become significant. Therefore, the choice of algorithm to perform this nearest neighbour search has an impact on the feasibility and speed of generating the final partition, leading to the development of various methods ranging from straightforward to highly optimised.

The brute-force method for assigning observations to their nearest centres is the most direct and conceptually simple approach. It operates as an exhaustive search: for each individual observation point, the algorithm calculates the squared distance to every single Voronoi centre. The centre that yields the minimum squared distance is then assigned to that observation. This process is repeated for all observations. The squared distance is calculated for points $\boldsymbol{p}$ and $\boldsymbol{q}$ in an n-dimensional space as

$$d(\boldsymbol{p}, \boldsymbol{q})^2 = ||\boldsymbol{p} - \boldsymbol{q}||^2.$$

While reliable, this method can be computationally expensive, especially with a large number of

observations or centres, as its complexity grows proportionally with the product of the two.

To significantly improve this method's performance, we can use vectorisation. Instead of iterating through each point one by one in slow, explicit loops, vectorisation leverages optimised linear algebra libraries to perform the distance calculations in parallel. The observation points and the Voronoi centres are represented as large matrices. The calculation of all pairwise squared Euclidean distances can be elegantly expressed as matrix operations. Let the observation points be represented by a matrix $\boldsymbol{X}$ of size $n \times p$ (n points, p dimensions) and the Voronoi centres by a matrix $\boldsymbol{C}$ of size $m \times d$ (m centres, d dimensions). The squared Euclidean distance between a single observation vector $\boldsymbol{x}_i$ (with dimensions not in the tessellation removed) and a centre vector $\boldsymbol{c}_j$ is

$$\boldsymbol{D}_{ij}^2 = ||\boldsymbol{x}_i - \boldsymbol{c}_j||^2 = (\boldsymbol{x}_i - \boldsymbol{c}_j) \cdot (\boldsymbol{x}_i - \boldsymbol{c}_j)^T = ||\boldsymbol{x}_i||^2 - 2\boldsymbol{x}_i \cdot \boldsymbol{c}_j^T + ||\boldsymbol{c}_j||^2,$$

where $|| \cdot ||$ denotes the Euclidean norm.

This can be expanded to compute the full $m \times n$ distance matrix $\boldsymbol{D}^2$ without loops. This involves calculating the squared norms of all rows of $\boldsymbol{X}$ and $\boldsymbol{C}$ and performing the matrix multiplication $\boldsymbol{X} \cdot \boldsymbol{C}^T$. After computing the entire distance matrix, a single 'argmin' operation can find the index of the minimum value for each row, instantly assigning each observation to its nearest centre.

While vectorisation dramatically speeds up the brute-force calculation, it still computes every possible distance. For even greater efficiency, it is possible to avoid this exhaustive comparison by using specialised data structures designed to partition the search space. A k-d tree (k-dimensional tree) is one such structure that organises points in a multi-dimensional space to significantly accelerate nearest neighbour searches Bentley (1975).

The tree is built by recursively partitioning the set of points. At the root, the points are split into two halves based on their median value along a specific dimension. Each of these halves then becomes a child node, which is subsequently split along the next dimension. This process continues, cycling through the dimensions, until each leaf node contains only a few points, creating a binary tree that spatially sorts the data.

Building a k-d tree for n points in k-dimensional space primarily involves recursively partitioning the data by finding the median value along alternating axes, which dictates its computational cost. The most efficient construction algorithms achieve a time complexity of $O(kn \log n)$ by pre-sorting

the points along all k dimensions prior to building the tree, rather than sorting at each recursive step, which would otherwise degrade the performance to $O(kn \log^2 n)$. Alternatively, applying a linear-time median-finding algorithm, such as Quickselect, at each node yields an average time complexity of $O(n \log n)$, though the hidden constant factors can sometimes make the pre-sorting method faster in practical implementations. Regarding space complexity, the k-d tree is highly efficient; because it partitions the space using the data points themselves without requiring vast auxiliary structures, the memory requirement scales linearly as $O(kn)$.

When searching for the nearest centre for a given observation point, the k-d tree avoids the need to compare against every single centre. The search algorithm traverses down the tree to find the leaf node where the observation point would belong, giving an initial best guess for the nearest neighbour (Friedman et al., 1977). The key to the method's efficiency is the backtracking step. As the algorithm moves back up the tree, it only explores the other branch at a given node if there is a chance it contains a closer point. This is determined by comparing the current shortest distance found to the distance between the observation point and the splitting hyperplane at that node. If the hyperplane is farther from the current best match, the entire region on the other side of the hyperplane can be pruned from the search, drastically reducing the number of distance calculations required.

Beyond k-d trees, other data structures and algorithms exist for efficient nearest neighbour searching. Ball trees, for instance, partition data within nested hyperspheres instead of the axis-aligned boxes used by k-d trees, which can offer better performance in higher-dimensional spaces where k-d trees become less effective (Uhlmann, 1991). In two or three dimensions, quadtrees and octrees are also common. Furthermore, some algorithms construct the Voronoi diagram directly rather than just assigning points. Methods such as Fortune's algorithm and a sweep-line approach construct the geometric boundaries of the Voronoi cells themselves (Fortune, 1987). The optimal choice of method ultimately depends on factors such as the data dimensionality, the number of points, and whether the set of centres is static or dynamic.

3.3 Gaussian process modelling using Voronoi tessellation

Historically, the application of Voronoi tessellations within classical statistical modelling has been limited. This is mostly due to the high computational cost associated with constructing and integrating these more complex geometric structures into statistical methods, particularly in higher dimensions. However, as computational efficiency has steadily increased, algorithms based on Voronoi partitions have become more viable and are beginning to see greater use. In this section, we describe one such modern method for Gaussian process modelling that uses Voronoi tessellations.

The method described in Pope et al. (2021) partitions the input space into disjoint regions using Voronoi tessellations, with each region being modelled by an independent Gaussian process (GP). This approach allows for an accurate representation of heterogeneous functions with discontinuities by relaxing the global smoothness assumption typically made in standard GP models.

The model defines a partition of the whole input space that is $\mathbb{R}^p$, where p is the number of covariates, by grouping the cells of an underlying Voronoi tessellation, T. This new partition is denoted as $\mathcal{R} = \{R_1, \ldots, R_r\}$, consisting of r disjoint regions. Each region $R_k \in \mathcal{R}$ is itself constructed as a disjoint union of one or more of the original Voronoi cells.

The partitioning is defined by a set of relationships between these cells, captured in the relationship vector $\boldsymbol{c}$, which specifies which cells are grouped into regions. The joint distribution over the tessellation parameters T is given by

$$\pi(T) = \pi(k, \boldsymbol{M})\pi(r \mid k)\pi(\boldsymbol{c} \mid k),$$

where k is the number of centres in the tessellation and $\boldsymbol{M}$ is the location of the centres, r is the number of regions, and $\boldsymbol{c}$ denotes the combination of cells grouped together to form regions.

The prior specifications are given by

$$\begin{aligned} k, \boldsymbol{M} &\sim \text{PoissonProcess}(\lambda), \\ r \mid k &\sim \text{DU}(1, k), \\ \boldsymbol{c} \mid k &\sim \text{DU}(1, B_k), \end{aligned}$$

where $\text{DU}(1, k)$ is a discrete uniform distribution on $\{1, \ldots, k\}$, B_k is the k-th Bell number, rep-

resenting the number of possible partitions of k items, and λ is the intensity parameter of the Poisson process controlling the number of centres. The Poisson process places centres randomly and independently across the input space, with the number of centres in any given area following a Poisson distribution, allowing for a flexible, spatially unbiased arrangement. Once the regions are defined, independent Gaussian processes are fit within each region, resulting in a piecewise GP model.

The core principle of Markov chain Monte Carlo (MCMC) methods is to construct a Markov chain whose stationary distribution matches a target distribution of interest, which in our context is the posterior distribution (Peters, 2006). A standard requirement for building such a chain is detailed balance, a mathematical condition that ensures the chain is reversible. If the transition probabilities satisfy detailed balance with respect to the target distribution, that target is guaranteed to be the invariant distribution of the chain. By designing transition kernels that preserve this reversibility, we can iteratively sample from complex distributions that lack a tractable analytical form.

However, conventional MCMC techniques, such as the Metropolis-Hastings algorithm, operate exclusively within a parameter space of fixed dimension. To estimate the posterior distribution over the space of tessellation configurations, where the dimension of the parameter space inherently fluctuates, a reversible-jump Markov chain Monte Carlo (RJMCMC) algorithm is employed (Green, 1995). This advanced framework extends the concept of detailed balance to accommodate trans-dimensional moves, enabling the Markov chain to jump between spaces of differing dimensions while strictly maintaining reversibility. For this model, the state of the Markov chain encompasses the set of Voronoi centres, the total number of centres k, and the partitioning of these cells into r regions.

At each iteration, one of four proposal types is selected with equal probability: (i) *birth*, which proposes adding a new centre drawn uniformly over the input space; (ii) *death*, which proposes removing a randomly selected existing centre; (iii) *move*, which perturbs the location of an existing centre using a Gaussian random walk; and (iv) *change*, which proposes altering the region assignment of a centre to either join a different existing region or form a new singleton region.

Similar to the BART algorithm, the proposed move is accepted with probability

$$\alpha = \min\left(1, \text{Likelihood ratio} \times \text{Prior ratio} \times \text{Adjustment factor}\right).$$

The adjustment factor accounts for asymmetric proposal probabilities in boundary cases. For instance, when only one centre is present, a *death* or *change* move is not permissible, and the acceptance probability of a *birth* move is multiplied by $1/2$ to ensure reversibility. Conversely, if a *death* move is proposed when there are exactly two centres, its acceptance probability is multiplied by 2. In all other cases, the adjustment factor is set to 1.

Each proposed tessellation is evaluated by independently fitting a Gaussian process model within each region. The likelihood contribution for each region is computed, and the overall posterior density is used to accept or reject the move.

This flexible Gaussian process modelling framework captures spatial heterogeneity and discontinuities using a data-driven Voronoi tessellation of the input space. By integrating reversible jump MCMC with region-specific Gaussian process priors, the model can adapt to both smooth and abrupt changes in the underlying response surface. The tessellation serves as a nonparametric partitioning mechanism, allowing distinct statistical behaviour across regions and thereby improving predictive performance in nonstationary settings.

Conceptually, Pope et al. (2021) and AddiVortes share the idea of updating tessellations via MCMC. However, the former applies Gaussian Processes inside regions of a single tessellation. In contrast, AddiVortes uses an additive ensemble of many simple tessellations. This design choice aligns AddiVortes closer to the BART philosophy (sum-of-weak-learners), enabling the efficient backfitting algorithms described in Chapter 4.

3.4 Other tessellation structures

In this section, we extend the concept of Voronoi tessellations by first exploring their construction under various metric spaces beyond the standard Euclidean case. We then introduce and define other fundamental geometric partitions that are closely related to Voronoi diagrams. These include Delaunay triangulations (Delaunay, 1934), which represent the dual structure, and Laguerre tessel-

lations Fejes Tóth (1977), which generalise the partitioning by incorporating site-specific weights. Each of these structures provides a different framework for spatial partitioning and is suited to different types of data and computational applications.

Changing the metric space

The structure of a Voronoi partition is fundamentally determined by its underlying metric space $(\mathcal{S}, d)$. The distance function d quantifies the notion of proximity, thereby axiomatically defining the geometric boundaries of the Voronoi cells. Consequently, any alteration to the metric will induce a new partition, fundamentally changing the shape, size, and topology of the cells, even for an identical set of centres.

The Minkowski distance is a generalisation of the Euclidean distance function, defined as

$$d_p(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p},$$

for $p \geq 1$. While the $p = 2$ case corresponds to the standard Euclidean distance, the $p = 1$ case defines the Manhattan distance, also known as the L_1-norm, which changes the geometry of the Voronoi cells.

The Manhattan distance between two points $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$ in $\mathbb{R}^n$ is given by

$$d_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} |x_i - y_i|.$$

In a 2D plane, this corresponds to movement along a grid, like a taxi in a city. The bisector between two points under the Manhattan distance is no longer a straight line perpendicular to the segment connecting the points; instead, it consists of line segments with slopes of ± 1. This results in Voronoi cells that are generally convex polygons, but with edges that are aligned with the coordinate axes or at 45° angles, giving them a more "blocky" or "diamond-like" appearance compared to their Euclidean counterparts, as illustrated in Figure 3.2.

As p approaches infinity, the Minkowski distance converges to the **Chebyshev distance**, also

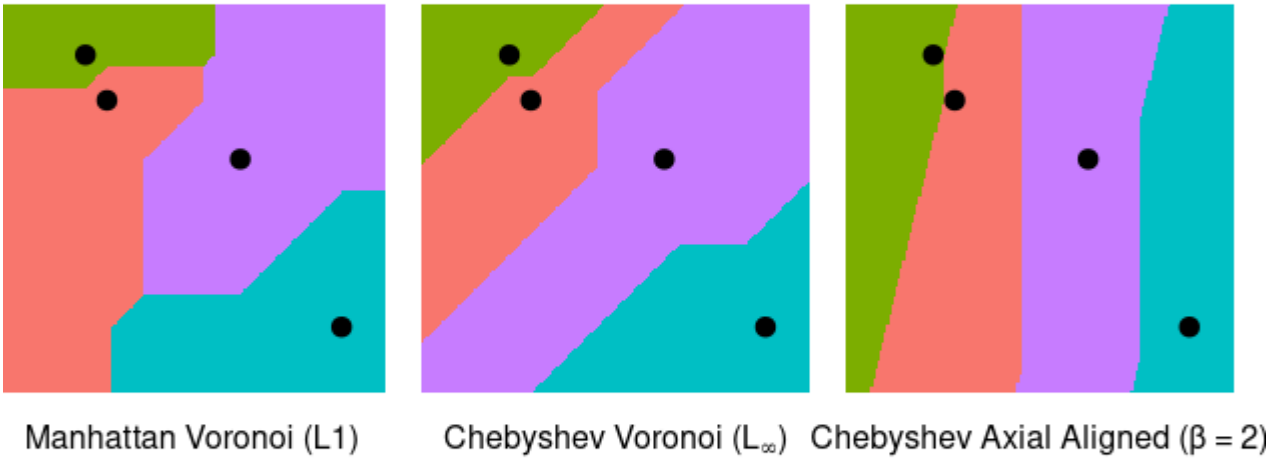


Figure 3.2: A comparison of Voronoi tessellations generated for the same set of points using three distinct distance metrics.

known as the L_∞-norm, which is defined as

$$d_\infty(\boldsymbol{x}, \boldsymbol{y}) = \max_{i=1,\ldots,n} |x_i - y_i|.$$

The bisectors are again composed of segments that are either horizontal, vertical, or at 45-degree angles. However, the entire tessellation has a more pronounced axis-aligned appearance, with cells that look more like "squares" and "rectangles" compared to the "diamond" shapes often seen in the Manhattan tessellation. An interactive website that illustrates Voronoi tessellations using these distance metrics can be found at http://www.sygreer.com/projects/voronoi/.

The **Chebyshev distance with axial scale** introduces a specific form of anisotropy, or directional dependence, into the metric space. This is achieved by applying a positive scaling factor, β, to differentially weight the axes. Let $\boldsymbol{p_0} = (x_0, y_0)$ and $\boldsymbol{p_1} = (x_1, y_1)$ be two points in $\mathbb{R}^2$, then the distance function in two dimensions is defined as

$$d(\boldsymbol{p_0}, \boldsymbol{p_1}) = \max(\beta|x_1 - x_0|, \frac{1}{\beta}|y_1 - y_0|).$$

The β parameter directly controls the geometric properties of the cells. When $\beta > 1$, the horizontal component $|x_1 - x_0|$ is penalised more heavily, effectively making horizontal distance "more

expensive" than vertical distance. This results in vertically elongated Voronoi cells, forming tall, thin rectangles. Conversely, when $\beta < 1$, the vertical component is penalised more heavily, leading to cells elongating horizontally into wide, short rectangles.

Beyond modifying the distance function within the Euclidean space, the foundational manifold ($\mathcal{S}$) on which the tessellation is constructed can itself be changed. This substitution, for instance, replacing a flat plane with a curved spherical or hyperbolic surface, introduces a new intrinsic geometry. Consequently, the very definition of 'distance' is altered, as straight-line paths are replaced by geodesics. This change in the underlying spatial framework naturally leads to structurally and topologically distinct Voronoi partitions, whose cell boundaries are no longer hyperplanes but are instead curves defined by the manifold's geometry.

In spherical geometry, the boundaries of Voronoi tessellation cells are arcs of great circles that act as perpendicular bisectors. In three dimensions, the geometry is defined by the equation of a sphere, $x^2 + y^2 + z^2 = R^2$, and usually the distance between any two points is defined as the arc length $d = R\gamma$, where γ is the angular separation between them. This distance metric is fundamental to constructing the Voronoi cells and contrasts with Euclidean geometry in that all resulting cells are finite and bounded.

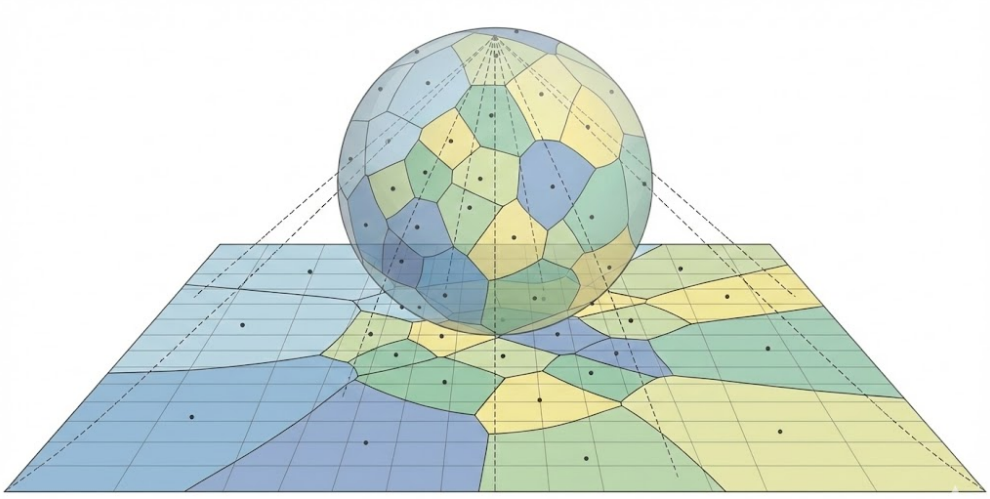


Figure 3.3: Illustration of a stereographic projection, a Voronoi tessellation on a sphere and the partition it produces in Euclidean space. Image created using generative AI.

Bijections between the Euclidean and Spherical spaces exist, and it is often useful to map the sphere's curved surface onto a flat Euclidean plane. Stereographic projection maps points on a

sphere to a plane by casting rays from a single point, the pole of projection, through every other point on the sphere's surface and onto the plane. This process creates a map that is conformal: that is, it preserves the angles at which curves intersect, although it does not preserve areas or boundaries of Voronoi tessellations. Figure 3.3 is an illustration of Voronoi tessellation on a sphere being stereographically projected onto a Euclidean plane from the north pole. The partition that is produced is displayed on the plane and it shows that the boundaries are now curved.

Alternatively, hyperbolic space is a non-Euclidean geometry characterised by constant negative curvature, in which any given line has infinitely many parallel lines passing through an external point. This geometry is often visualised using the Poincaré disk model (Poincaré, 1882), where the entire infinite space is contained within a unit circle, and distances grow exponentially as they approach the boundary. The shortest path between two points is a geodesic, which appears as a circular arc orthogonal to the disk's boundary. Consequently, a Voronoi tessellation in hyperbolic space is composed of hyperbolic polygons, whose boundaries are these geodesic arcs. This structure is particularly suited for representing hierarchical data, as the exponential nature of the space effectively models tree-like relationships.

Delaunay Triangulations

Delaunay triangulations can be traced back to Delaunay (1934), who first introduced duals of the Voronoi diagram. This work, extending beyond the familiar two-dimensional Euclidean plane, explores the triangulations of the sphere and other surfaces. Delaunay triangulations have found applications in numerous fields, such as in computer vision (for example, Labatut et al., 2007), medical imaging (for example, Pennisi et al., 2016) and robotics (for example, Schwab and Lunze, 2023).

For a given set of points P in d-dimensional Euclidean space, a Delaunay triangulation, denoted $\mathrm{DT}(P)$, is a specific triangulation of the points. The vertices of the triangles are points in the set P. Its defining property is the "empty circle" or "empty hypersphere" condition. This states that for any simplex in the triangulation (a triangle in 2D, a tetrahedron in 3D), the unique circum-hypersphere that passes through its vertices contains no other point from P in its interior. This structure is the direct geometric dual of the Voronoi tessellation; two points in P are connected by an edge in the Delaunay triangulation if and only if their corresponding Voronoi cells share a

common boundary. This duality implies a close relationship between these two types of partitions, and they are often used together to solve complex problems in computational geometry.

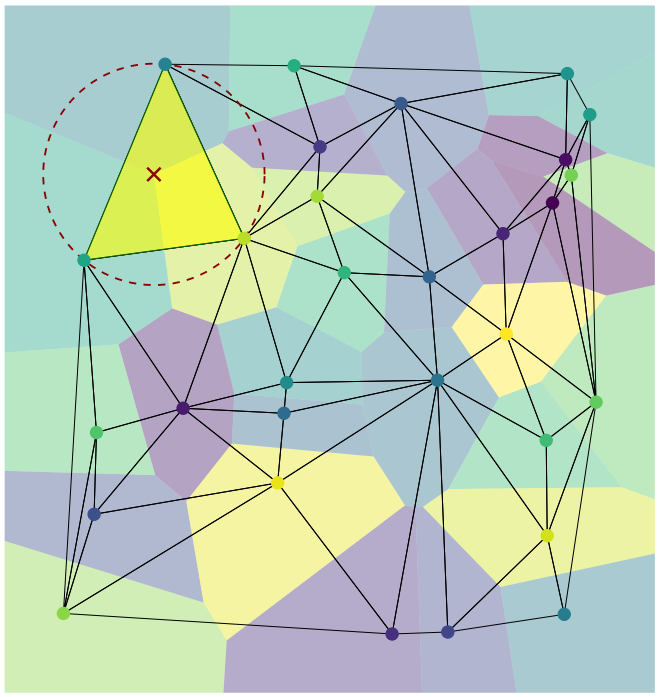


Figure 3.4: Illustration of a Delaunay triangulation given a set of points. The black lines represent the partition using Delaunay triangulation. The points are the centres of the Voronoi tessellation, with coloured cells representing the partition of the tessellation.

This geometric structure is illustrated in Figure 3.4. The triangulation itself is depicted by the black lines connecting the points in P. One such simplex, a triangle, is highlighted in yellow to demonstrate the empty circle property. Its validity as a Delaunay triangle is confirmed by its red circumcircle, which passes through its three vertices and, crucially, contains no other point from P within its interior. Furthermore, the background displays the corresponding Voronoi tessellation, visually reinforcing the dual relationship between the two structures.

A unique Delaunay triangulation exists as long as the points in P are in "general position". In this context, the general position primarily means that no $d + 2$ points lie on the same d-hypersphere. For example, in a 2D plane, it means no four points lie on the same circle. If this condition is violated (for a 2D example, four points do lie on a circle), the triangulation is not unique because the interior edge connecting them can be "flipped" to form two different valid

triangulations.

Delaunay triangulations are constructed using several algorithms, each optimised for different applications. The Bowyer-Watson algorithm, for instance, incrementally adds points and re-triangulates affected areas (Bowyer, 1981; Watson, 1981), while the divide-and-conquer method handles large point sets efficiently by recursively merging smaller triangulations (Guibas and Stolfi, 1985).

In scenarios where the metric deviates from the standard Euclidean distance, such as in weighted or anisotropic distances, the properties of Delaunay triangulations change accordingly. For example, in the presence of anisotropy, where distance is dependent on direction, the fundamental empty circle property is redefined using a metric tensor. As explored by Labelle and Shewchuk (2003), this effectively transforms the circumcircles into circum-ellipses, where the shape and orientation of the ellipses are dictated by the local directional properties of the space. The resulting triangulation naturally produces triangles that are stretched and oriented along the directions of higher correlation, thus providing a mesh that is far more adapted to and representative of the underlying anisotropic spatial structure of the data than a standard Euclidean triangulation could be.

Laguerre Tessellations

Aurenhammer (1987) credits the definition of power distance to work by mathematicians Edmond Laguerre and Georgy Voronoi, in the 19th century. This distance gives the foundations of Laguerre tessellations or Power diagrams, which extends the traditional Voronoi concept by attributing a specific weight to each cell centre. This generalisation allows the partitioning to account for the varying influence of each site, creating boundaries that shift according to the associated weights rather than relying solely on geometric distance.

Borgwardt and Frongillo (2019) demonstrates how power diagrams can be used to solve information elicitation problems by characterising the regions where specific reports are optimal for a given scoring rule. Arslan and Koditschek (2016) presents a framework for achieving collision-free robot navigation by utilising power diagrams to define a globally navigation-guaranteed vector field that directs a robot toward a target while maintaining an exact distance from multiple obstacles. Furthermore, Alpers et al. (2015) uses generalised balanced power diagrams to accurately represent the complex three-dimensional geometry of polycrystals by optimising cell weights and positions to

match experimental data or grain statistics.

Given a set $\mathcal{P} = \{(p_1, w_1), (p_2, w_2), \ldots, (p_n, w_n)\}$, where each p_i is a point in a metric space $(\mathcal{S}, d)$ and each w_i is a real-valued weight, the Laguerre tessellation is defined based on the weighted distance, known as the power distance. For a point $\boldsymbol{x} \in \mathcal{S}$ and a centre (p_i, w_i), this distance is given by

$$\text{PowerDistance}(\boldsymbol{x}, (p_i, w_i)) = d(\boldsymbol{x}, p_i)^2 - w_i.$$

The Laguerre cell, or Power cell, V_i associated with the site (p_i, w_i), is defined as

$$V_i = \{\boldsymbol{x} \in \mathcal{S} : \text{PowerDistance}(\boldsymbol{x}, (p_i, w_i)) \leq \text{PowerDistance}(\boldsymbol{x}, (p_j, w_j)) \quad \text{for all} \quad j \neq i\}.$$

The weighted nature of these tessellations allows them to accommodate varying influences or densities across the spatial field. When applied to Euclidean space $\mathbb{R}^d$ with the standard distance, and when centres are in general position, the boundaries between cells are segments of hyperplanes, leading to polytopal regions that depend on both location and weight.

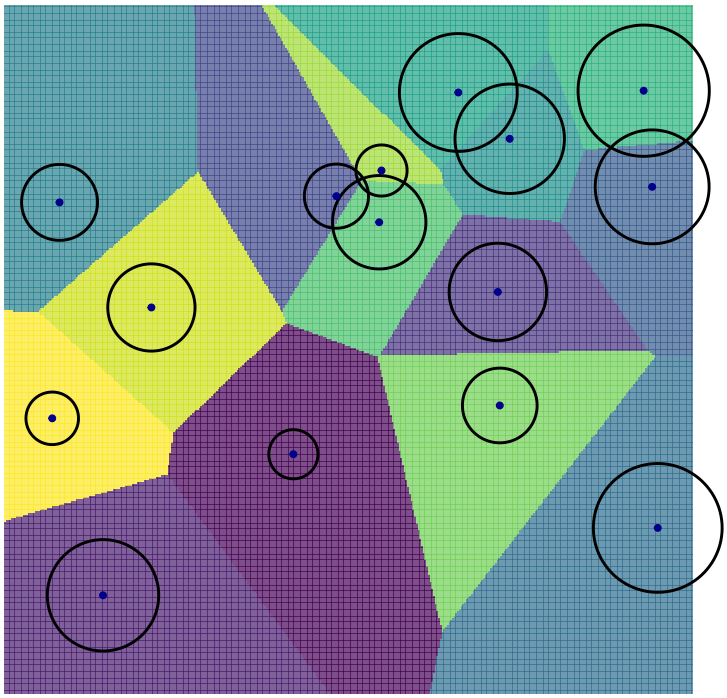


Figure 3.5: Illustration of a Laguerre tessellation given a set of points.

Figure 3.5 is an illustration of a Laguerre tessellation. The space is partitioned into coloured

convex cells, which consists of a centre point (blue dot) and an associated circle. The radius of each circle acts as a weight, influencing the size of its corresponding cell. A larger radius gives more weight to that cell, resulting in a larger area. A key property, visible here, is that the boundaries between all cells remain straight lines. This is true even for overlapping circles, as seen in the upper-centre cluster. In such cases, the boundary is the unique straight line, known as the radical axis, that passes through the two points of intersection of the circles.

In scenarios where weights vary significantly, the properties of these tessellations change, allowing for cell shapes which model the underlying environment or structure more accurately. As with classical Voronoi diagrams and Delaunay triangulations, Laguerre tessellations can be computed by adapting algorithms for unweighted tessellations to incorporate weight in determining cell boundaries.

3.5 Discussion

Voronoi tessellations provide a simple yet powerful framework for partitioning space, offering a natural method to capture non-linear relationships that contrast sharply with the rigid, axis-aligned splits of traditional decision trees. While these tessellations historically incurred a higher computational cost, we have discussed how this can be effectively managed through specialised algorithms, ranging from vectorised brute-force methods to advanced structures like k-d trees. These innovations, paired with rising computational availability, have made the derivation of Voronoi tessellations significantly more efficient and accessible for practical algorithmic use.

Despite these advancements, current predictive models that employ ensembles of Voronoi tessellations, such as in Pope et al. (2021), often face limitations by incorporating the full set of covariates into every partition. Given the inherent mathematical complexity of Voronoi partitioning, this approach frequently results in prohibitive computational costs, particularly when applied to high-dimensional data where the calculation of cell boundaries becomes exponentially more demanding.

Addressing this challenge, Chapter 4 introduces a novel ensemble approach where Voronoi tessellations are dynamically updated, akin to the decision trees in Bayesian Additive Regression Trees (BART). By adopting the concept of weak learners described in Section 2.1, this method utilises

only a subset of covariates for each tessellation. This strategy not only improves computational efficiency but also allows for a more flexible and precise partitioning of the covariate space compared to traditional methods.

The generalisation of the standard Euclidean Voronoi diagram enables the modelling of a diverse array of real-world statistical scenarios. By altering the underlying metric to include Manhattan or Chebyshev distances, or by shifting the space to spherical or hyperbolic spaces, the definition of proximity can be tailored to the specific constraints of fields ranging from urban planning to global climate science. Additionally, the incorporation of weights within Laguerre tessellations permits the modelling of sites with varying degrees of influence, while the dual structure of Delaunay triangulations offers an optimal foundation for mesh generation and interpolation.

Building on this theoretical versatility, Chapter 8 outlines future research directions for the AddiVortes algorithm that exploit these alternative partitioning techniques. We mention the potential for integrating different distance metrics and using Laguerre diagrams to define the partition. Furthermore, the discussion extends to adapting the model for non-Euclidean geometries, such as spherical spaces, which would allow AddiVortes to naturally accommodate data structures with intrinsic curvature.

Chapter 4

AddiVortes: (Bayesian) Additive Voronoi Tessellations

This chapter is based on the paper 'AddiVortes: (Bayesian) Additive Voronoi Tessellations', published in the Journal of Computational and Graphical Statistics (Stone and Gosling, 2025a). The content has been adapted and expanded for inclusion in this thesis.

In Chapter 2, the BART algorithm was introduced that uses a Bayesian backfitting algorithm and regularisation priors applied to an additive ensemble. However, a clear limitation of BART is the use of rigid decision trees that are unable to partition the covariate space effectively. Developments such as oblique BART have tried to increase the flexibility of the partition but tessellations described in Chapter 3 are far more natural than decision trees in partitioning the covariate space.

AddiVortes draws on ideas underpinning BART, particularly the notions of additive structure and iterative posterior sampling. It combines multiple low-dimensional tessellations to approximate functions in high-dimensional spaces, offering an alternative to traditional tree-based ensembles such as BART. Each tessellation contributes additively to the predicted outcome, with its own set of covariates and cell-specific output values. This sum-of-tessellations framework enables the model to capture both marginal and interaction effects by partitioning the covariate space effectively.

The algorithm employs a similar Metropolis–Hastings-based MCMC backfitting strategy used in BART to iteratively update each tessellation while conditioning on the other tessellations. This means AddiVortes inherits the foundational benefits of the BART framework, ensuring that the

new model remains computationally stable and maintains calibrated uncertainty quantification.

Furthermore, the use of regularisation priors and the selection of only subsets of covariates in each tessellation mitigates overfitting by discouraging overly complex tessellations with many centres or dimensions. This approach not only allows for flexible, non-linear function estimation but also makes the algorithm computationally more efficient than global Voronoi models, which tend to be more expensive in high-dimensional settings.

Importantly, the AddiVortes framework is designed to be extendable. In subsequent chapters we demonstrate its adaptability to a variety of modelling contexts, including classification tasks in Chapter 6, and heteroscedastic regression in Chapter 7.

To support the application of the AddiVortes methods discussed in this chapter, a dedicated package is available on Github. This package allows users to run the algorithm and perform comprehensive diagnostics. It can be accessed at https://johnpaulgosling.github.io/AddiVortes/index.html. Additionally, the AddiVortes package can be found as a CRAN package at https://cran.rstudio.com/web/packages/AddiVortes/index.html which allows easy installation in RStudio.

The remainder of the chapter is organised as follows. In Section 4.1, the fundamental concepts upon which AddiVortes is built are outlined. In Section 4.2, a Bayesian backfitting MCMC algorithm and methods for inference are described. In Sections 4.4 and 4.5, we illustrate the potential of AddiVortes through a diverse range of examples, encompassing both simulated scenarios and real-world data. Section 4.7 concludes the chapter with a discussion.

4.1 Fundamentals of the AddiVortes model

This section outlines the fundamentals of the AddiVortes framework, beginning with the definition of tessellation components and essential data preprocessing. It then details the construction of the additive ensemble model and introduces the regularisation priors that control model complexity.

4.1.1 Tessellation setup and data integration

For AddiVortes, each tessellation incorporates a (non-fixed) subset of covariates as its dimensions, which can change when updating the tessellations, as discussed in Section 4.2. That is, if the j^{th} tessellation involves three covariates at a given iteration, then it is three-dimensional. Each cell in

the j^{th} tessellation is assigned a parameter $\mu_{ij} \in \mathbb{R}$ for $i = 1, \ldots, b_j$, where b_j denotes the number of cells. This parameter, μ_{ij}, represents the output value associated with all samples located within the corresponding cell.

For instance, Figure 4.1 illustrates a two-dimensional tessellation based on covariates x_1 and x_2, where the crosses denote the centres and the cells are labelled with their respective output values $(\mu_1, \ldots, \mu_8)$. Observations are represented by points; each observation is associated with the output value of the cell in which it falls. The complete set of output values for the j^{th} tessellation is denoted by $\boldsymbol{M}_j = \{\mu_{1j}, \ldots, \mu_{b_j j}\}$.

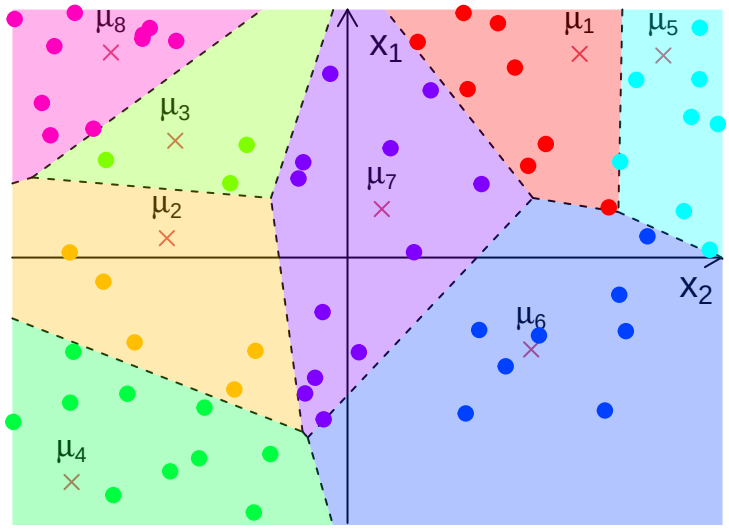


Figure 4.1: An illustration of predictive modelling using a two-dimensional Voronoi tessellation. Centres (crosses) are labelled with output values $(\mu_1, \ldots, \mu_8)$. Observations (points) inherit the output value of the cell to which they belong.

To facilitate computation and improve numerical stability, we apply an affine transformation to the dependent variable Y. Specifically, following the approach in Chipman et al. (2010), we scale and shift the training outcomes $\boldsymbol{y}_{\text{train}}$ such that their minimum and maximum values are -0.5 and 0.5, respectively.

Similarly, each covariate is rescaled using one of two methods. The first approach linearly maps the minimum and maximum values of each covariate for the training data to -0.5 and 0.5, respectively. This ensures that all training inputs are transformed to lie within a p-dimensional

hypercube $[-0.5, 0.5]^p$, where p denotes the number of covariates. Alternatively, each covariate can be standardised, so its distribution has approximately zero mean and unit variance. In this thesis, the first method is used unless otherwise stated.

This transformation enables the prior distribution for the coordinates of the tessellation centres to be specified independently of the original data scale, ensuring that centre locations can be sampled independently, that is, within the suitable range of the hypercube, without depending on the particular distribution or range of the observed covariates. Moreover, these transformations promote comparability between covariates and simplify the formulation of priors across different dimensions.

4.1.2 Sum-of-tessellations

Similar to the sum-of-trees model, a sum-of-tessellations model comprises a sum of m distinct tessellations and the value of m acts as a hyperparameter of the method (see Section 4.1.4). Each tessellation is indexed by $j \in \{1, \ldots, m\}$ and is denoted by T_j, which defines both the subset of covariates involved (that is, the dimensions of the tessellation) and the configuration of the centres that generate the corresponding Voronoi cells.

For the j^{th} tessellation, the associated cell output values are collected in the set

$$\boldsymbol{M}_j = \{\mu_{1j}, \ldots, \mu_{b_j j}\},$$

where b_j is the number of cells partitioned by T_j. Given an input vector $\boldsymbol{x} \in \mathbb{R}^p$, the prediction contribution from tessellation T_j is represented by the function $g(\boldsymbol{x} \mid T_j, \boldsymbol{M}_j)$, that is the output value μ_{ij} of the cell containing $\boldsymbol{x}$. This definition of g is similar to the BART model described earlier in Section 2.1.

If the model consists of a single tessellation ($m = 1$), then the predictive model takes the form

$$Y = g(\boldsymbol{x} \mid T_1, \boldsymbol{M}_1) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2),$$

where $g(\boldsymbol{x} \mid T_1, \boldsymbol{M}_1)$ is output value of the cell that $\boldsymbol{x}$ belongs to. In this case, the tessellation serves as a piecewise-constant approximation to the regression function.

In the general case of $m > 1$, the AddiVortes model approximates the conditional expectation $\mathrm{E}[Y \mid \boldsymbol{x}, \boldsymbol{y}_{\text{train}}, \boldsymbol{X}_{\text{train}}]$ by summing the contributions across all tessellations, that is,

$$Y = \sum_{j=1}^{m} g(\boldsymbol{x} \mid T_j, \boldsymbol{M}_j) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2).$$

Thus, each tessellation captures a portion of the regression surface's structure, allowing the model to capture covariate interactions to approximate functions.

When tessellations are constructed over a single covariate, the corresponding outputs μ_{ij} represent main effects, since $g(\boldsymbol{x} \mid T_j, \boldsymbol{M}_j)$ depends on only one dimension of $\boldsymbol{x}$. In contrast, multi-dimensional tessellations introduce interaction effects between the involved covariates, as the partitioning and corresponding outputs depend jointly on multiple covariate values. Consequently, the additive structure of AddiVortes naturally enables the model to capture both main effects and interactions.

An additional computational advantage of the additive construction is the reduced dimensionality of each tessellation. Since each T_j typically involves only a subset of the covariates, the associated nearest-neighbour search to determine the Voronoi cell for a given input can be performed efficiently. Specifically, we employ the `FNN`: Fast Nearest Neighbour Search Algorithms and Applications package from Beygelzimer et al. (2023) to efficiently assign each new observation $\boldsymbol{x}$ to its corresponding cell in each tessellation using k-d trees, described in Section 3.2.

4.1.3 Regularisation priors

The AddiVortes algorithm samples from the posterior distribution

$$p\left((T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m), \sigma \mid \boldsymbol{y}_{\text{train}}, \boldsymbol{x}_{\text{train}}\right),$$

where $(T_j, \boldsymbol{M}_j)$ denotes the structure and output values associated with the j^{th} tessellation. Therefore, we define a prior density

$$\pi\left((T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m), \sigma\right),$$

to allow for posterior sampling.

Following the approach taken in Chipman et al. (2010), we simplify the specification of regularisation priors by imposing independence assumptions. Specifically, we assume that

$$\begin{aligned}\pi\left(\left(T_1, \boldsymbol{M}_1\right), \ldots, \left(T_m, \boldsymbol{M}_m\right), \sigma\right) &= \prod_{j=1}^{m} \left[\pi(T_j, \boldsymbol{M}_j)\right] \pi(\sigma) \\ &= \prod_{j=1}^{m} \left[\pi\left(\boldsymbol{M}_j \mid T_j\right) \pi(T_j)\right] \pi(\sigma)\end{aligned} \tag{4.1}$$

and

$$\pi(\boldsymbol{M}_j \mid T_j) = \prod_{i=1}^{b_j} \pi\left(\mu_{ij} \mid T_j\right), \tag{4.2}$$

where $\mu_{ij} \in \boldsymbol{M}_j$ is the output value associated with the i^{th} cell in tessellation T_j.

These assumptions imply that the tessellation structures are independent of one another and of σ *a priori*, and that, within each tessellation, the output values are independent given the tessellation structure. It is important to note, however, that although independence is assumed *a priori*, dependence is induced *a posteriori* through the joint likelihood, particularly since the acceptance probability of proposed tessellation updates in the MCMC algorithm (described in Section 4.2) depends on the current configuration of all tessellations.

The independence assumptions greatly simplify prior specification. Instead of requiring a complex joint prior over all tessellations and parameters, it suffices to specify priors for the tessellation structure T_j, for the output values $\mu_{ij} \mid T_j$, and for σ. Moreover, for practical implementation, we adopt identical prior forms across tessellations and across cells within each tessellation, further reducing complexity.

The prior forms we adopt are similar to those proposed by Chipman et al. (1998) for Bayesian CART models. The prior distributions are designed to favour simpler models, encouraging tessellations with fewer dimensions and fewer centres. This regularisation controls model complexity, preventing any single tessellation from dominating the prediction and mitigating the risk of overfitting, which would otherwise affect both predictive performance and computational efficiency.

To facilitate implementation, we propose default prior specifications based on a small number of interpretable hyperparameters used to determine the distributions of T_j, $\boldsymbol{M}_j$, and σ. These defaults have been found to be effective across a wide range of examples, as illustrated in Sections 4.4 and 4.5. Our recommended defaults, given in Section 4.3, are calibrated using the observed variability in

the response Y, providing a reasonable choice of hyperparameters. Alternatively, users may specify plausible ranges for hyperparameters, guided by theoretical considerations, and use cross-validation to select appropriate values. However, it should be noted that cross-validation will incur additional computational cost, which may not justify the potential gains in accuracy.

T_j prior

The prior distribution for the tessellation structure T_j is determined by several components: the number of covariates involved, the number of centres, the selection of covariates included in the tessellation, and the coordinates of the centres themselves.

For the number of covariates included in tessellation T_j, we specify a shifted binomial distribution. Specifically, if d_j denotes the number of covariates included, then

$$d_j - 1 \propto \text{Binomial}(p-1, \omega/p),$$

where p is the total number of available covariates and $\omega < p$ is a hyperparameter controlling the expected complexity in terms of the number of dimensions of the tessellation. This choice of distribution reflects a modelling preference towards lower-dimensional tessellations, thereby regularising model complexity. In practice, by setting ω relatively small compared to p, we favour simpler tessellations that capture lower-dimensional marginal structures, reserving higher-dimensional tessellations for cases strongly supported by the data. Note that ω must be less than p, otherwise the probability of inclusion is greater than 1 and the distribution becomes invalid.

The number of cells, or equivalently the number of centres, within a tessellation T_j is assigned a shifted Poisson prior. Specifically,

$$b_j - 1 \propto \text{Poisson}(\lambda_c),$$

where b_j denotes the number of cells and λ_c is a rate hyperparameter. The rationale for this choice is that Poisson distributions naturally model counts and offer flexibility in the expected number of centres. Small values of λ_c encourage tessellations with few centres, avoiding over-partitioning of the covariate space, which could otherwise lead to overfitting.

The selection of which covariates are included in T_j is modelled as uniform among all remaining covariates not already selected. This prior is computationally easy to sample from and avoids biases that favour particular variables before observing the data. However, one could modify this in a similar way to DART (Linero, 2016) by introducing a Dirichlet prior on the inclusion of variables. This approach is suggested for future work in Chapter 8 as it allows the model to adapt to sparsity and be used for variable selection.

The coordinates of the centres within each tessellation are assigned independent normal distributions. Specifically, we model the coordinates using a normal distribution $\mathcal{N}(0, \sigma_c^2)$. A zero-mean prior is a natural choice, as the covariates are pre-processed before modelling by either standardisation or rescaling linearly so that their values in the training data fall within the range of $[-0.5, 0.5]$. Consequently, for most covariates, the empirical mean after transformation is close to zero. The normal prior then places the mass of the distribution around the observed centre of the covariate space, while the variance parameter σ_c^2 determines the spread of the centres. We choose σ_c such that the centres have a high probability of falling within a suitable range of the $(-0.5, 0.5)$ interval, thereby aligning with the observed covariate range.

The tails of the normal distribution allow sampling centres slightly outside the observed data range, which is advantageous. By allowing centres beyond the training data, the tessellation can adapt to potential extrapolation requirements near the boundaries, and helps avoid edge effects where model predictions might otherwise be excessively constrained.

Each of these distributional choices for specifying the prior on T_j is made deliberately to balance model complexity and regularisation. The binomial prior on the number of covariates encourages low-dimensional structures, the Poisson prior on the number of centres controls partition complexity, the uniform choice of covariates maintains neutrality between input dimensions, and the normal prior on centre locations enables local adaptability while respecting the observed data range. Collectively, these choices regularise the tessellation structures in an adaptable but controlled manner.

$\mu_{ij} \mid T_j$ prior

The parameters μ_{ij}, which define the output values associated with each cell in a tessellation, are modelled using a conjugate normal prior distribution, $\mu_{ij} \mid T_j \sim \mathcal{N}(\mu_\mu, \sigma_\mu^2)$, following the approach adopted in Chipman et al. (2010). The use of a conjugate prior has notable computational advan-

tages, particularly by facilitating closed-form updates and enabling straightforward marginalisation over μ_{ij} during posterior sampling.

Since our model expresses the conditional expectation $\mathrm{E}[Y \mid \boldsymbol{x}, \boldsymbol{y}_{\text{train}}, \boldsymbol{X}_{\text{train}}]$ as the sum of the outputs of all tessellations that the input $\boldsymbol{x}$ falls into, and each μ_{ij} is independently and identically distributed, the induced prior distribution on $\mathrm{E}[Y \mid \boldsymbol{x}, \boldsymbol{y}_{\text{train}}]$ is itself normal,

$$\mathrm{E}[Y \mid \boldsymbol{x}, \boldsymbol{y}_{\text{train}}] \sim \mathcal{N}(m\mu_\mu, m\sigma_\mu^2),$$

where m denotes the total number of tessellations in the model.

To ensure that the prior for the model output predominantly falls within the observed range of the training output variable, specifically the interval $(y_{\min}, y_{\max})$, we choose the hyperparameters μ_μ and σ_μ accordingly. In particular, we wish to centre the prior for $\mathrm{E}[Y \mid \boldsymbol{x}, \boldsymbol{y}_{\text{train}}]$ at zero and constrain its variability such that the majority of its mass lies within the rescaled interval $(-0.5, 0.5)$. This is achieved by fixing the prior mean to zero, $\mu_\mu = 0$, and choosing σ_μ such that

$$m\mu_\mu \pm k\sqrt{m}\sigma_\mu = \pm 0.5,$$

where k is a user-specified hyperparameter controlling the strength of the shrinkage. Solving this relationship gives

$$\sigma_\mu = \frac{0.5}{k\sqrt{m}}.$$

This formulation ensures that, under the prior, the model output is unlikely to deviate far from zero, and that the degree of shrinkage is increased either by increasing the number of additive tessellations m or by choosing a larger value of k. In practice, this shrinkage effect has the beneficial consequence of regularising the model and preventing overfitting by discouraging excessively large contributions from individual tessellations.

σ prior

For σ, we use the same prior as BART described in Section 2.1.3, and adopt a conjugate prior in the form of the inverse-chi-squared distribution,

$$\sigma^2 \sim \text{Inv-}\chi^2(\nu, \lambda),$$

where ν denotes the degrees of freedom and λ is the scale parameter. This choice is motivated by both its conjugacy in normal models and its interpretability, allowing us to encode prior beliefs about the plausible scale of the noise in the response variable Y.

To inform the specification of the hyperparameters ν and λ, we employ a data-driven calibration strategy that centres the prior around a conservative estimate $\hat{\sigma}$ of the residual standard deviation. The rationale is to assign high prior probability to values of σ that are consistent with the observed variability in the training data, while still allowing sufficient flexibility to accommodate alternative values through appropriate dispersion.

To calibrate λ, two plausible strategies for giving a rough overestimate of $\hat{\sigma}$ are considered. The first is a naive estimator, namely the sample standard deviation of the transformed response values in the training set. The second strategy involves fitting a linear regression model of the scaled response Y on the rescaled covariates $\boldsymbol{X}$, and taking the residual standard deviation from this model as the estimate $\hat{\sigma}$. This latter approach typically gives a better estimate, as it accounts for linear dependencies in the covariates and better reflects the unexplained variance. However, this approach cannot be used if the number of covariates exceeds the number of observations, as the system of equations becomes underdetermined.

Once $\hat{\sigma}$ has been estimated, a hyperparameter is selected for the specific quantile q of the prior distribution for σ that matches $\hat{\sigma}$, that is,

$$\Pr(\sigma < \hat{\sigma}) = q,$$

for a chosen quantile level $q \in (0.7, 0.99)$.

Increasing λ shifts the distribution towards larger values of σ^2, while decreasing λ concentrates

it around smaller values. Specifically, when $\nu > 2$, the mean is given by

$$\mathbb{E}[\sigma^2] = \frac{\lambda}{\nu - 2}.$$

Thus, λ directly determines the expected scale of the variance, and λ also influences the overall spread of the distribution in a similar way; that is, larger values give a greater spread.

The degrees of freedom ν are chosen from a range typically between 3 and 10 to balance the trade-off between prior concentration and dispersion. A lower ν results in a heavier-tailed distribution, allowing for higher values of σ, while a higher ν implies more confidence in the scale parameter.

This strategy enables an interpretable and practical specification of the prior for σ, which reflects both prior regularisation and the empirical characteristics of the observed data. By tuning the prior through simple scalar quantities, this approach maintains computational efficiency while maintaining a good prior form for σ. Figure 4.2 illustrates priors corresponding to three (ν, q) settings when the rough overestimate is $\hat{\sigma} = 1$. We refer to these three settings, $(\nu, q) = (10, 0.75)$, $(3, 0.90)$, $(3, 0.99)$, as conservative, default and aggressive, respectively.

4.1.4 Choice of m

We assume a fixed number of tessellations and an iterative backfitting algorithm are used to cycle through these tessellations. Determining an appropriate value for the number of tessellations m presents a challenge. Two common strategies to address this challenge are to treat m as an unknown parameter and assign a prior to it or to use cross-validation to select the 'best' value for m from a range of reasonable choices. However, both of these strategies can incur significant computational costs.

Typically, as illustrated in Section 4.5, increasing m from 1 leads to considerable improvements in predictive performance until it levels off. Beyond this point, for larger m values, performance slowly degrades, because each tessellation contributes such a small fraction of the expected outcome and there is uncertainty within each tessellation.

To avoid computational overhead, a practical approach is to start with a default value of $m = 200$. In numerous scenarios, this default value has demonstrated excellent predictive performance,

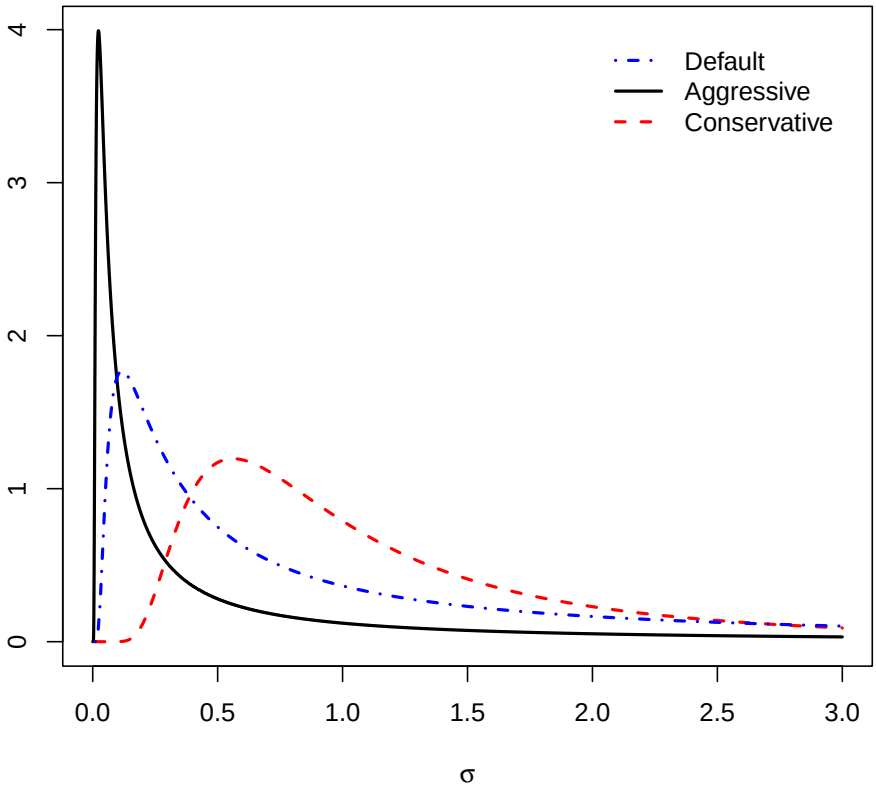


Figure 4.2: Priors for σ for the conservative, default and aggressive cases when $\hat{\sigma} = 1$.

and this is illustrated in Section 4.4 and 4.5. However, other considerations might come into play, particularly when using AddiVortes for variable selection, where the choice of m could be influenced by additional factors.

4.2 MCMC backfitting algorithm

An objective of AddiVortes is to sample from the posterior distribution of all unknown parameters in the sum-of-tessellations model,

$$\pi\big((T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m), \sigma \mid \boldsymbol{y}\big), \tag{4.3}$$

using a Markov chain Monte Carlo (MCMC) backfitting algorithm, similar to the one described in Chapter 2. To simplify notation, throughout this section we denote the training outputs and covariates by $\boldsymbol{y}$ and $\boldsymbol{X}$, respectively.

The Gibbs sampler proceeds by drawing σ from its full conditional distribution, followed by sequentially drawing $(T_j, \boldsymbol{M}_j)$ conditional on $(\{T_{j'}, \boldsymbol{M}_{j'}\}_{j' \neq j}, \sigma, \boldsymbol{y}, \boldsymbol{X})$ for $j = 1, \ldots, m$. Hastie and Tibshirani (2000) previously investigated a similar Gibbs sampling approach for additive and generalised additive models, with σ held fixed. They showed that the method constitutes a stochastic generalisation of the classical backfitting algorithm,(Hastie, 1990).

The draw of σ from its full conditional is straightforward, as the likelihood

$$\pi(\boldsymbol{y} \mid (T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m), \sigma^2) \propto N\left(\sum_{j=1}^{m} g(\boldsymbol{x} \mid T_j, \boldsymbol{M}_j), \sigma^2\right),$$

and the scaled inverse chi squared distribution selected as our prior are conjugate, thus, we sample from the inverse-gamma distribution,

$$\begin{aligned}\pi\left(\sigma^2 \mid (T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m), \boldsymbol{y}, \boldsymbol{X}\right) &\propto \pi(\boldsymbol{y} \mid (T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m), \sigma^2)\pi(\sigma^2) \\ &\propto \mathrm{IG}\left(\frac{\nu + n}{2}, \frac{\nu\lambda + \sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{m} g(\boldsymbol{x_i} \mid T_j, \boldsymbol{M}_j)\right)^2}{2}\right),\end{aligned}$$

which is trivial using established methods.

Sampling $(T_j, \boldsymbol{M}_j)$ conditionally on the other tessellations and σ is more involved. The conditional probability

$$\pi(T_j, \boldsymbol{M}_j \mid \{T_{j'}, \boldsymbol{M}_{j'}\}_{j' \neq j}, \sigma, \boldsymbol{y})$$

depends on the data only through the partial residual vector

$$\boldsymbol{R}_j = \boldsymbol{y} - \sum_{k \neq j} g(\boldsymbol{x} \mid T_k, \boldsymbol{M}_k), \tag{4.4}$$

which represents the residuals from excluding the j^{th} tessellation. Thus, sampling $(T_j, \boldsymbol{M}_j)$ given

the other tessellations and $\boldsymbol{y}$ reduces to drawing from the distribution

$$(T_j, \boldsymbol{M}_j) \mid \boldsymbol{R}_j, \sigma. \tag{4.5}$$

Equation (4.5) is equivalent to the posterior distribution in the single-tessellation model $\boldsymbol{R}_j = g(\boldsymbol{x}; T_j, \boldsymbol{M}_j) + \epsilon$, where $\boldsymbol{R}_j$ plays the role of the data. Because we have used a conjugate prior for $\boldsymbol{M}_j$, the marginal posterior for T_j can be written (up to a normalising constant) as

$$\begin{aligned} \pi(T_j \mid \boldsymbol{R}_j, \sigma) &\propto \pi(\boldsymbol{R}_j \mid T_j, \sigma)\pi(T_j) \\ &\propto \pi(T_j) \int \pi(\boldsymbol{R}_j, \boldsymbol{M}_j \mid T_j, \sigma)\, d\boldsymbol{M}_j, \\ &\propto \pi(T_j) \int \pi(\boldsymbol{R}_j \mid \boldsymbol{M}_j, T_j, \sigma)\pi(\boldsymbol{M}_j \mid T_j, \sigma)\, d\boldsymbol{M}_j. \end{aligned} \tag{4.6}$$

since T_j and σ are independent a priori. Note that, here, we use proportionality because, as in the MH algorithm, the normalising constant cancels.

This permits two-stage sampling:

$$\begin{aligned} &T_j \mid \boldsymbol{R}_j, \sigma, \\ &\boldsymbol{M}_j \mid T_j, \boldsymbol{R}_j, \sigma. \end{aligned} \tag{4.7}$$

The sampling of T_j in Equation (4.7) is performed using a Metropolis-Hastings algorithm similar to that described by Chipman et al. (2010). To change the current tessellation, we propose six possible moves, each associated with a specific proposal probability. The algorithm can propose either adding or removing a centre, with each action having a probability of 0.2. Similarly, it can propose adding or removing a covariate, with probability 0.2 per move. The final two moves are to swap a covariate or to change the position of a centre, each with a proposal probability of 0.1. Following the proposal of a new tessellation, it is either accepted or rejected based on the acceptance probability derived in Section 4.2.1.

The most critical proposal moves in the AddiVortes method are the add/remove dimension and add/remove centre steps, as they directly control the structural complexity of the model; therefore, the default specification is to apply a slightly higher probability to these proposals. The former determines which covariates are actively interacting by altering the dimensionality of individual tessellations, while the latter adjusts the granularity of the partitions by varying the

number of centres. In contrast, the swap and change moves serve a more exploratory role, helping the Metropolis–Hastings algorithm traverse the posterior space more effectively and avoid becoming trapped in local modes; thus, we apply these steps with a slightly lower probability.

To add a centre, new coordinates are drawn from a normal distribution (see Section 4.1.3). To remove a centre, one is selected uniformly at random. Adding a covariate entails selecting one not currently in the tessellation and sampling coordinates for each centre. Removing a covariate involves randomly selecting one of the currently present covariates and discarding its associated coordinates. Changing a centre resamples its coordinates, and swapping involves replacing one covariate with another not yet included, using resampled coordinates.

Note that the acceptance probability must be adjusted when certain moves are not possible, for example, one cannot remove the last remaining centre. See Section 4.2.1 for details on these adjustments.

By marginalising over $\boldsymbol{M}_j$ in Equation (4.6), we avoid the need for reversible-jump MCMC between continuous spaces of varying dimension (Green, 1995). That is, instead of sampling both the tessellation structure T_j and its continuous cell output values $\boldsymbol{M}_j$ at the same time. $\boldsymbol{M}_j$ can be integrated out (or marginalised over). This is possible because a conjugate prior distribution is used for the cell output values. Thus, the marginal likelihood of the data can be calculated given only the tessellation structure.

After selecting T_j, the vector $\boldsymbol{M}_j$ is drawn by independently sampling each μ_{ij} from its tractable normal full conditional. This enables the computation of the next residual vector $\boldsymbol{R}_{j+1}$ for subsequent tessellation updates.

Let $\boldsymbol{R}_{ij} = (R_{ij1}, \ldots, R_{ijn_i})^T$ be a subset from $\boldsymbol{R}_j$ where n_i is the number of R_{ijh} values allocated to the i^{th} cell with parameter μ_{ij}. We note that $R_{ijh} \mid T_j, \mu_{ij}, \sigma \sim \mathcal{N}(\mu_{ij}, \sigma^2)$ and

$$\mu_{ij} \mid T_j \sim \mathcal{N}(\mu_\mu, \sigma_\mu^2).$$

Then, the full conditional density of μ_{ij} is given by

$$\begin{aligned} p(\mu_{ij} \mid T_j, \sigma, \boldsymbol{R}_j) &\propto p(\boldsymbol{R}_{ij} \mid T_j, \mu_{ij}, \sigma) p(\mu_{ij} \mid T_j) \\ &\propto \exp\left[-\frac{\sum_h (R_{ijh} - \mu_{ij})^2}{2\sigma^2}\right] \exp\left[-\frac{(\mu_{ij} - \mu_\mu)^2}{2\sigma_\mu^2}\right] \\ &\propto \exp\left[-\frac{(n_i\sigma_\mu^2 + \sigma^2)\mu_{ij}^2 - 2(\sigma_\mu^2 \sum_h R_{ijh} + \sigma^2\mu_\mu)\mu_{ij}}{2\sigma^2\sigma_\mu^2}\right] \\ &\propto \exp\left[-\frac{\left(\mu_{ij} - \frac{\sigma_\mu^2 \sum_h R_{ijh} + \sigma^2\mu_\mu}{n_i\sigma_\mu^2 + \sigma^2}\right)^2}{\frac{2\sigma^2\sigma_\mu^2}{n_i\sigma_\mu^2 + \sigma^2}}\right], \end{aligned} \tag{4.8}$$

where $\sum_h (R_{ijh} - \mu_{ij})^2$ is the summation of the squared difference between the parameter μ_{ij} and each R_{ijh} value allocated to the cell with parameter μ_{ij}. Hence,

$$\mu_{ij} \mid T_j, \sigma \sim \mathcal{N}\left(\frac{\sigma_\mu^2 \sum_h R_{ijh} + \sigma^2\mu_\mu}{n_i\sigma_\mu^2 + \sigma^2}, \frac{2\sigma^2\sigma_\mu^2}{n_i\sigma_\mu^2 + \sigma^2}\right).$$

Note that, this is simplified in practice because in Section 4.1.3, we set $\mu_\mu = 0$.

The algorithm is initialised with m simple, single-centre, one-dimensional tessellations. Iterations are run through a burn-in period, typically 200 iterations, as convergence is rapid (see Section 4.5).

Algorithm 2: Bayesian backfitting MCMC for posterior inference in AddiVortes

Data: Training data $(\boldsymbol{X}_{\text{train}}, \boldsymbol{y}_{\text{train}})$, AddiVortes hyperparameters $(m, \nu, q, k, \sigma_c, \omega, \lambda_c)$

Find λ ; `// Using naïve or linear method`

for $j = 1$ **to** m **do**

  Initialise tessellation T_j ; `// Create` m `single-centre tessellations`

  Assign μ_{1j} ; `//` $\mu_{1j} = \texttt{mean}(\boldsymbol{y}_{\texttt{train}})/m$

for $i = 1$ **to** *number of iterations* **do**

  Sample σ^2 ; `// From inverse-gamma distribution`

  for $j = 1$ **to** m **do**

    Compute residuals $\boldsymbol{R}_j$; `//` $R_{ij} = y_i - \sum_{k \neq j} g(x_i \mid T_k, \boldsymbol{M}_k)$

    Propose a new tessellation ;

    Accept or reject proposed tessellation ;

    Sample μ_{ij} ; `// From conjugate normal distribution`

Section 4.5 demonstrates that the backfitting MCMC algorithm performs well, even in challenging scenarios. The algorithm tends to be consistent across repeated runs, but averaging over

several chains is recommended to ensure convergence. The algorithm exhibits strong mixing and adaptability, making it a powerful tool for complex modelling tasks. Algorithm 2 outlines the steps of AddiVortes.

4.2.1 Deriving the Metropolis–Hastings ratio

To sample T_j, we use a reversible jump move, and the acceptance probability is given by

$$\alpha(T_j, T_j^*) = \min\left\{1, \frac{q(T_j^*, T_j)}{q(T_j, T_j^*)} \cdot \frac{P(\boldsymbol{R}_j \mid T_j^*, \sigma)}{P(\boldsymbol{R}_j \mid T_j, \sigma)} \cdot \frac{P(T_j^*)}{P(T_j)}\right\},$$

where $q(T_j, T_j^*)$ is the proposal distribution for transitioning from T_j to T_j^*, $P(\boldsymbol{R}_j \mid T_j, \boldsymbol{M}_j)$ is the likelihood, and $P(T_j)$ is the prior probability of the tessellation structure. The acceptance ratio follows a similar structure to the BART algorithm; however, the components differ considerably given the use of tessellations in AddiVortes. Here we present the derivations for the three resulting ratios.

Likelihood Ratio

The likelihood of the residuals is

$$\pi(\boldsymbol{R}_j \mid T_j, \sigma) = \int \pi(\boldsymbol{R}_j \mid \boldsymbol{M}_j, T_j, \sigma)\pi(\boldsymbol{M}_j \mid T_j)\, d\boldsymbol{M}_j, \tag{4.9}$$

and since all the output values are independent of each other,

$$\pi(\boldsymbol{R}_j \mid \boldsymbol{M}_j, T_j, \sigma)\pi(\boldsymbol{M}_j \mid T_j, \sigma) = \prod_{i=1}^{b_j} \pi(\boldsymbol{R}_j \mid \mu_{ij}, T_j, \sigma)\pi(\mu_{ij} \mid T_j).$$

Recalling that b_j is the number of centres in the j^{th} tessellation, our goal is to find the marginal likelihood of the partial residuals $\boldsymbol{R}_j$ by integrating out the cell-specific output values μ_{ij}. We begin with the joint probability of the residuals and the parameters, which is the product of the likelihood and the prior. For a given cell i in tessellation j, this is given by

$$\pi(\boldsymbol{R}_{ij}, \mu_{ij} \mid T_j, \sigma) = \pi(\boldsymbol{R}_{ij} \mid \mu_{ij}, T_j, \sigma)\pi(\mu_{ij} \mid T_j).$$

Given the conjugate normal likelihood and prior, where $\boldsymbol{R}_{ij}$ has n_i observations, we have:

$$\pi(\boldsymbol{R}_{ij}, \mu_{ij} \mid T_j, \sigma) \propto \left(\frac{1}{\sigma^2}\right)^{n_i/2} \exp\left(-\frac{1}{2\sigma^2}\sum_{h=1}^{n_i}(R_{ijh} - \mu_{ij})^2\right)\left(\frac{1}{\sigma_\mu^2}\right)^{1/2} \exp\left(-\frac{1}{2\sigma_\mu^2}\mu_{ij}^2\right).$$

We can now combine the terms and focus on the exponent. We expand the quadratic term and collect terms involving μ_{ij},

$$\begin{aligned}
\text{Exponent} &= -\frac{1}{2\sigma^2}\sum_{h=1}^{n_i}(R_{ijh}^2 - 2R_{ijh}\mu_{ij} + \mu_{ij}^2) - \frac{1}{2\sigma_\mu^2}\mu_{ij}^2 \\
&= -\frac{1}{2}\left[\frac{1}{\sigma^2}\sum_{h=1}^{n_i}R_{ijh}^2 - \frac{2\mu_{ij}}{\sigma^2}\sum_{h=1}^{n_i}R_{ijh} + \frac{n_i\mu_{ij}^2}{\sigma^2} + \frac{\mu_{ij}^2}{\sigma_\mu^2}\right] \\
&= -\frac{1}{2}\left[\frac{\sum R_{ijh}^2}{\sigma^2} + \mu_{ij}^2\left(\frac{n_i}{\sigma^2} + \frac{1}{\sigma_\mu^2}\right) - 2\mu_{ij}\left(\frac{\sum R_{ijh}}{\sigma^2}\right)\right].
\end{aligned}$$

The next step is to complete the square for the terms involving μ_{ij}. This allows us to rewrite the exponent in terms of the kernal of a normal distribution for μ_{ij}, simplifying the integration. The expression inside the brackets can be recognised as the kernel with mean $\bar{\mu}$ and variance V_μ, given by

$$V_\mu = \left(\frac{n_i}{\sigma^2} + \frac{1}{\sigma_\mu^2}\right)^{-1} = \frac{\sigma^2\sigma_\mu^2}{n_i\sigma_\mu^2 + \sigma^2} \quad \text{and} \quad \bar{\mu} = V_\mu\left(\frac{\sum R_{ijh}}{\sigma^2}\right) = \frac{\sigma_\mu^2\sum R_{ijh}}{n_i\sigma_\mu^2 + \sigma^2}.$$

Substituting these back into the expression allows us to separate the terms that depend on μ_{ij} from those that do not

$$\pi(\boldsymbol{R}_{ij}, \mu_{ij} \mid T_j, \sigma) \propto \exp\left(-\frac{\sum R_{ijh}^2}{2\sigma^2}\right)\exp\left(\frac{(\sum R_{ijh})^2\sigma_\mu^2}{2\sigma^2(n_i\sigma_\mu^2 + \sigma^2)}\right) \times \exp\left(-\frac{(\mu_{ij} - \bar{\mu})^2}{2V_\mu}\right).$$

Now we can perform the integration over μ_{ij}. The expression has been factored into terms that form the marginal likelihood and the kernel of the posterior distribution of μ_{ij}. Hence,

$$\begin{aligned}
\pi(\boldsymbol{R}_{ij} \mid T_j, \sigma) &= \int \pi(\boldsymbol{R}_{ij}, \mu_{ij} \mid T_j, \sigma)\, d\mu_{ij} \\
&\propto \underbrace{\left(\frac{1}{\sigma^{2n_i}\sigma_\mu^2}\right)^{\frac{1}{2}}}_{\text{non exponent term}} \underbrace{\exp\left(-\frac{\sum R_{ijh}^2}{2\sigma^2} + \frac{(\sum R_{ijh})^2\sigma_\mu^2}{2\sigma^2(n_i\sigma_\mu^2 + \sigma^2)}\right)}_{\text{Terms not dependent on } \mu_{ij}} \times \underbrace{\int \exp\left(-\frac{(\mu_{ij} - \bar{\mu})^2}{2V_\mu}\right) d\mu_{ij}}_{\text{Integral of a Gaussian kernel}}.
\end{aligned}$$

The integral on the right is proportional to the normalising constant of a Gaussian, which is $\sqrt{2\pi V_\mu}$.

The marginal likelihood for all cells in the tessellation is the product of the marginal likelihoods for each individual cell. Substituting this result back into Equation 4.9, we obtain the final expression for the marginal likelihood of the residuals for the entire tessellation T_j

$$\pi(\boldsymbol{R}_j \mid T_j, \sigma) \propto \prod_{i=1}^{b_j} \left(\frac{1}{\sigma^2}\right)^{\frac{n_i}{2}} \left(\frac{\sigma^2}{n_i\sigma_\mu^2 + \sigma^2}\right)^{1/2} \exp\left(-\frac{\sum R_{ijh}^2}{2\sigma^2} + \frac{(\sum R_{ijh})^2\sigma_\mu^2}{2\sigma^2(n_i\sigma_\mu^2 + \sigma^2)}\right)$$

To derive the Metropolis-Hastings acceptance probability, we need the likelihood ratio. This is the ratio of the marginal likelihood of the data under the proposed tessellation, T_j^*, to the marginal likelihood under the current tessellation, T_j. We will use a star notation (for example, n_i^*, b_j^*) to denote values corresponding to the proposed tessellation.

Using the final expression for the marginal likelihood derived in the previous section, the full ratio is

$$\frac{\pi(\boldsymbol{R}_j \mid T_j^*, \sigma)}{\pi(\boldsymbol{R}_j \mid T_j, \sigma)} = \frac{\prod_{i=1}^{b_j^*} \left(\frac{\sigma^2}{n_i^*\sigma_\mu^2+\sigma^2}\right)^{1/2} \exp\left(-\frac{\sum_h (R_{ijh})^2}{2\sigma^2} + \frac{(\sum_h R_{ijh})^2\sigma_\mu^2}{2\sigma^2(n_i^*\sigma_\mu^2+\sigma^2)}\right)}{\prod_{i=1}^{b_j} \left(\frac{\sigma^2}{n_i\sigma_\mu^2+\sigma^2}\right)^{1/2} \exp\left(-\frac{\sum_h R_{ijh}^2}{2\sigma^2} + \frac{(\sum_h R_{ijh})^2\sigma_\mu^2}{2\sigma^2(n_i\sigma_\mu^2+\sigma^2)}\right)}.$$

We can simplify this expression considerably. The term $\exp(-\frac{\sum_h R_{ijh}^2}{2\sigma^2})$ is taken over the product of all cells. This means the sum in the exponent is over all partial residuals. Since the set of partial residuals $\boldsymbol{R}_j$ is the same for both the current and proposed tessellations (they are partitioned differently, but the values themselves are fixed for this step), this entire exponential term cancels out from the numerator and denominator.

After this cancellation, we are left with

$$\frac{\pi(\boldsymbol{R}_j \mid T_j^*, \sigma)}{\pi(\boldsymbol{R}_j \mid T_j, \sigma)} = \frac{\prod_{i=1}^{b_j^*} \left(\frac{\sigma^2}{n_i^*\sigma_\mu^2+\sigma^2}\right)^{1/2} \exp\left(\frac{(\sum_h R_{ijh})^2\sigma_\mu^2}{2\sigma^2(n_i^*\sigma_\mu^2+\sigma^2)}\right)}{\prod_{i=1}^{b_j} \left(\frac{\sigma^2}{n_i\sigma_\mu^2+\sigma^2}\right)^{1/2} \exp\left(\frac{(\sum_h R_{ijh})^2\sigma_\mu^2}{2\sigma^2(n_i\sigma_\mu^2+\sigma^2)}\right)}.$$

We can separate the pre-factor terms from the exponential. The term $(\sigma^2)^{1/2} = \sigma$ appears b_j^* times in the numerator's product and b_j times in the denominator's product. For moves that do not change the number of centres (change centre position, add/remove/swap covariates), we have

$b_j^* = b_j$, so all σ cancels. This gives the following simplified likelihood ratio

$$\frac{\prod_{i=1}^{b_j} \left(n_i\sigma_\mu^2 + \sigma^2\right)^{1/2}}{\prod_{i=1}^{b_j^*} \left(n_i^*\sigma_\mu^2 + \sigma^2\right)^{1/2}} \times \exp\left[\frac{\sigma_\mu^2}{2\sigma^2}\left(\sum_{i=1}^{b_j^*} \frac{(\sum_h R_{ijh})^2}{n_i^*\sigma_\mu^2 + \sigma^2} - \sum_{i=1}^{b_j} \frac{(\sum_h R_{ijh})^2}{n_i\sigma_\mu^2 + \sigma^2}\right)\right]. \tag{4.10}$$

However, for moves that do change the number of centres, the σ terms do not fully cancel. When adding a centre, $b_j^* = b_j + 1$, so an extra factor of σ remains in the numerator. Conversely, when removing a centre, $b_j^* = b_j - 1$, and we are left with an extra factor of $(\sigma)^{-1}$ in the denominator. Therefore, the likelihood ratio is calculated using Equation (4.10) and then multiplied by σ for an ADD move, or divided by σ for a REMOVE move.

Transition Ratio and Tessellation Structure Ratio

Now, we derive the transition and prior probability ratios required for the Metropolis-Hastings acceptance step. We will use the add dimension move as an illustrative example, though the ratios for all other moves, such as removing a dimension or adding a centre, can be derived in a similar fashion by making the appropriate adjustments.

Let D be the number of covariates in the data; d be the number of dimensions in the original tessellation T_j, and c^* be the coordinates of the newly added dimension for all centres. Terms with the star notation correspond to the proposed tessellation. Here, we also denote 'AD' as choosing the add dimension move and 'RD' as the remove dimension move.

The proposal probability of transitioning from a tessellation T_j to a new tessellation T_j^* by adding a dimension can be formally expressed. It is the product of selecting the 'add dimension' move, choosing a specific dimension from the $D - d$ available covariates, and sampling new coordinate values for that dimension from their prior distribution, $p(c^*)$, that is,

$$p(T_j^* \mid T_j) = p(\text{AD}) \cdot \frac{1}{D-d} \cdot p(c^*).$$

The reverse move, transitioning from T_j^* back to T_j, corresponds to removing the dimension that was added. Its proposal probability is the product of selecting the 'remove dimension' move and choosing to remove the specific dimension that was just added, which is now one of $d + 1$

dimensions in the tessellation, so

$$p(T_j \mid T_j^*) = p(\text{RD}) \cdot \frac{1}{d+1}.$$

Therefore, the transition ratio required for the acceptance probability is

$$\frac{p(T_j \mid T_j^*)}{p(T_j^* \mid T_j)} = \frac{p(\text{RD}) \cdot \frac{1}{d+1}}{p(\text{AD}) \cdot \frac{1}{D-d} \cdot p(c^*)} = \frac{p(\text{RD})}{p(\text{AD})} \frac{D-d}{d+1} \frac{1}{p(c^*)}.$$

Next, we must consider the ratio of the prior probabilities, $\frac{p(T_j^*)}{p(T_j)}$. The prior for a tessellation structure, $p(T_j)$, is composed of priors on the number of centres, the number of dimensions, the specific set of covariates, and the coordinate values. Since most components remain unchanged, the ratio simplifies significantly. It becomes the ratio of the priors for the number of dimensions, the choice of covariates, and the coordinate values. This gives the complete prior ratio as

$$\frac{p(T_j^*)}{p(T_j)} = \frac{p(d+1)}{p(d)} \cdot \frac{d+1}{D-d} \cdot p(c^*),$$

where $p(d+1)$ and $p(d)$ denote the probability of a tessellation having $d+1$ and d dimensions, respectively.

The product of the transition and prior ratios gives a simpler expression, as several terms conveniently cancel out.

$$\frac{p(T_j \mid T_j^*)}{p(T_j^* \mid T_j)} \frac{p(T_j^*)}{p(T_j)} = \left(\frac{p(\text{RD})}{p(\text{AD})} \frac{D-d}{d+1} \frac{1}{p(c^*)} \right) \left(\frac{p(d+1)}{p(d)} \frac{d+1}{D-d} p(c^*) \right) = \frac{p(\text{RD})}{p(\text{AD})} \frac{p(d+1)}{p(d)}.$$

This final expression correctly balances the probability of proposing a move with the prior belief in the resulting model structure, ensuring the MCMC sampler converges to the correct posterior distribution.

Adjustments

To ensure the model's validity, several structural constraints are imposed on each tessellation within the AddiVortes framework. A tessellation must always have at least one centre and be defined by at least one dimension. Furthermore, the number of dimensions in a tessellation cannot exceed

the total number of available covariates, since we do not allow repeating covariates as dimensions. These constraints ensure that the model components remain well-defined throughout the sampling process.

These rules, however, create asymmetries in the MCMC proposal mechanism that must be addressed to maintain detailed balance and ensure the sampler converges to the correct posterior distribution. For instance, it is impossible to propose removing a centre if only one exists, or to remove a dimension if the tessellation is only one-dimensional. Similarly, adding or swapping a dimension is not possible if all available covariates are already included in the tessellation. To correct for these asymmetries, specific adjustments are made to the Metropolis-Hastings acceptance ratio in these boundary cases.

These corrections are applied as multiplicative factors. When proposing to add a centre to a tessellation that currently has only one, the acceptance ratio is multiplied by $1/2$. Conversely, when proposing to remove a centre from a tessellation with two centres, the ratio is multiplied by 2. The root cause of this $1/2$ correction factor is the strict structural boundary at exactly one centre. At this absolute minimum limit, it is impossible to propose a "remove centre" move without collapsing the model. Because the removal option is completely eliminated, the algorithm is forced into a deterministic choice: it has a probability of 1 of proposing an "add centre" move.

If this proposed addition is evaluated and accepted, the tessellation successfully expands to two centres. In this new state, the algorithm is no longer at the boundary, meaning the standard symmetry of choices returns. The sampler now has an equal $1/2$ probability of proposing to add another centre or to remove one.

Consequently, the probability of proposing the exact reverse move, removing the newly added centre to return to the original one-centre state, is exactly $1/2$. The Metropolis-Hastings mechanism calculates the proposal ratio by dividing this reverse probability by the forward probability. Because the forward move was forced by the boundary constraints (probability 1) while the reverse move remains symmetric (probability $1/2$), the resulting ratio is mathematically fixed at $(1/2)/1 = 1/2$.

A similar logic applies to the number of dimensions. If a tessellation is one-dimensional and an 'add dimension' move is proposed, the acceptance ratio is multiplied by $1/2$. If a tessellation has two dimensions and a 'remove dimension' move is proposed, the factor is 2. Finally, if all covariates are in use and a 'remove dimension' move is proposed, the ratio is multiplied by $1/2$,

while proposing to add a dimension to a tessellation that is missing only one covariate requires multiplying the ratio by 2. In all other, non-boundary scenarios, this adjustment factor is set to 1.

4.2.2 Inference for the model parameters

The backfitting algorithm generates a Markov chain whose stationary distribution is the posterior distribution

$$\pi((T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m), \sigma \mid \boldsymbol{y}, \boldsymbol{X}).$$

This Markov chain is ergodic, meaning that the distribution of the chain converges to the target posterior distribution as the number of iterations tends to infinity. After discarding an initial burn-in period and running the chain for a sufficiently long number of iterations, we obtain a sample from the posterior distribution.

The corresponding sequence of sum-of-tessellations functions is given by

$$f(\cdot) = \sum_{j=1}^{m} g(\cdot; T_j^*, \boldsymbol{M}_j^*), \tag{4.11}$$

where $(T_1^*, \boldsymbol{M}_1^*), \ldots, (T_m^*, \boldsymbol{M}_m^*)$ are posterior samples. This defines a random function whose distribution approximates the posterior distribution $\pi(f \mid \boldsymbol{y})$, where $f\colon \mathbb{R}^p \to \mathbb{R}$ denotes the unknown regression function. The sequence of sampled functions $f_1^*, \ldots, f_K^*$ constitutes a draw from $\pi(f \mid \boldsymbol{y})$, where each $f_k^*(\boldsymbol{x}) = \sum_{j=1}^{m} g(\boldsymbol{x}; T_{j,k}, \boldsymbol{M}_{j,k})$ for $k = 1, \ldots, K$.

Point estimates of $f(\boldsymbol{x})$, such as the posterior mean or median, can be obtained from the posterior samples as

$$\widehat{f}_{\text{mean}}(\boldsymbol{x}) = \frac{1}{K} \sum_{k=1}^{K} f_k^*(\boldsymbol{x}), \quad \text{or} \quad \widehat{f}_{\text{median}}(\boldsymbol{x}) = \text{median}\left(f_1^*(\boldsymbol{x}), \ldots, f_K^*(\boldsymbol{x})\right).$$

To quantify uncertainty in these estimates, one can compute posterior credible intervals. A naive $(1-\alpha)\%$ pointwise credible interval for $f(\boldsymbol{x})$ is given by the empirical quantiles

$$\left[Q_{\alpha/2}(\boldsymbol{x}), Q_{1-\alpha/2}(\boldsymbol{x})\right],$$

where $Q_\alpha(\boldsymbol{x})$ denotes the empirical α-quantile of $\{f_k^*(\boldsymbol{x})\}_{k=1}^K$. However, this approach does not

account for observation noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

A more appropriate method is to construct *posterior predictive intervals*, which incorporate both the posterior uncertainty in f and the noise variance σ^2. For each posterior draw $f_k^*(\boldsymbol{x})$, we generate a predictive draw

$$\tilde{y}_k = f_k^*(\boldsymbol{x}) + \epsilon_k, \quad \epsilon_k \sim \mathcal{N}(0, \sigma_k^2),$$

where σ_k^2 is the posterior draw of the noise variance corresponding to iteration k. The empirical quantiles of the set $\{\tilde{y}_k\}_{k=1}^K$ yield a predictive $(1-\alpha)\%$ interval for future observations. These intervals naturally widen in regions of sparse data, reflecting greater epistemic and aleatoric uncertainty.

In practice, the Markov chain may exhibit autocorrelation, particularly when the posterior is highly multimodal or the proposal mechanism mixes slowly. To mitigate the effects of autocorrelation, one may apply *thinning*, whereby only every k^{th} sample is retained from the chain. As discussed in Owen (2017), thinning aims to reduce the dependence between retained samples. However, it should be applied cautiously, as discarding samples reduces the total number of observations and may not always yield efficiency gains unless storage or memory constraints are binding.

4.3 Default hyperparameters

Careful selection of hyperparameters in the prior distribution is crucial for controlling the influence of individual tessellation components and ensuring robust predictive performance in the AddiVortes model. This regularisation strategy improves generalisation and allows the model to flexibly adapt to a wide range of data structures without overfitting. The default hyperparameter values were selected based on empirical evaluation across both simulated and real-world datasets. The default values are summarised in Table 4.1.

In particular, the sparsity-controlling parameter ω controls the prior distribution over the number of covariates used in each tessellation. A value of $\omega = 3$ places most prior mass on tessellations involving two to three covariates, which strikes a balance between capturing main effects and allowing for moderate interactions. Importantly, ω has been found to be relatively insensitive: the model's performance is robust to small changes in its value.

The parameter λ_c, which regulates the number of centres within a tessellation, is more sensitive.

While a default of $\lambda_c = 25$ works well in many settings, it can sometimes encourage overly fine partitions in tessellations, especially in high-noise or low-sample-size contexts. In such cases, excessive partitioning may lead to overfitting as individual tessellations become too influential. Adjusting λ_c downward helps mitigate this risk by favouring simpler partitions.

For the centre location prior, the variance parameter σ_c is chosen to reflect the rescaled covariate space. Since all covariates are transformed to lie approximately in $[-0.5, 0.5]$, we set $\sigma_c = 0.8$ to ensure that centre coordinates are likely to fall within a suitable range of this region, with progressively less probability outside.

The shrinkage applied to the cell means μ_{ij} is controlled by both m, the number of tessellations, and the shrinkage factor k in the $\mu_{ij} \mid T_j$ prior. As k or m increases, the prior becomes tighter, placing stronger regularisation on individual tessellations. This ensures that each tessellation contributes only modestly to the overall fit, encouraging an ensemble-driven solution. We find that setting $k \in [1, 3]$ performs well in practice, and use $k = 3$ as the default.

The hyperparameters ν and q, which influence the inverse-χ^2 prior over the noise variance σ^2, appear to have less pronounced impact on out-of-sample RMSE. As a default, we use $(\nu, q) = (6, 0.85)$. Alternative settings such as $(10, 0.75)$ and $(3, 0.99)$ are also reasonable: the former imposes stronger shrinkage and may be viewed as an "aggressive" prior that encourages smaller noise variances, potentially at the risk of overfitting. The latter is more "conservative", favouring larger noise variances and thus promoting smoother fits.

We set the default number of tessellations to $m = 200$ since this specification provides robust predictive performance across diverse data generating processes. While increasing m may marginally reduce the out-of-sample error, the resultant gains in accuracy are typically insufficient to justify the increased cost of posterior sampling. In contrast, a reduction in m can be advantageous for variable selection and sparse settings. By restricting the size of the ensemble, the model is forced to concentrate posterior mass on the most relevant predictors.

4.4 Performance on regression data sets

We conducted a comprehensive evaluation of the predictive performance of AddiVortes by comparing it with several benchmark algorithms across a selection of data sets obtained from Loh et al.

Parameter	Description	Default	Effect of Increasing Value
ω	Sparsity-controlling parameter for number of covariates	3	Places more prior mass on a higher number of covariates per tessellation.
λ_c	Regulates the number of centres per tessellation	25	Encourages more cells and finer partitioning of the space.
σ_c	Variance of centre coordinates	0.8	Increases probability of absolute values of coordinates of centres being larger.
k	Shrinkage factor for cell means	3	Tightens the prior on cell output values and enforces stronger regularisation.
(ν, q)	Inverse-χ^2 prior hyperparameters	$(6, 0.85)$	$(\nu, q) = (10, 0.75)$ is aggressive and encourages smaller values of σ, while $(3, 0.99)$ is conservative and encourages larger values.
m	Number of tessellations	200	Uses a larger ensemble of tessellations, increasing computational time.

Table 4.1: Summary of model parameters and prior specifications

(2007). Owing to availability constraints, we conducted our analysis on a subset of the full list of datasets; the included datasets are summarised in Table 4.2. These data sets vary in both sample size and dimensionality, with the number of observations ranging from 96 to 4,177, and each data set has a single output variable and between 4 and 21 covariates. To accommodate categorical covariates, we applied one-hot encoding, since each covariate had fewer than 5 levels, and this did not significantly increase dimensionality.

Name	$\boldsymbol{n}$	**Name**	$\boldsymbol{n}$	**Name**	$\boldsymbol{n}$	**Name**	$\boldsymbol{n}$	**Name**	$\boldsymbol{n}$
Abalone	4177	Baskball	96	Boston	506	Edu	1400	Enroll	258
Fat	252	Hatco	100	Labor	2953	Medicare	4406	Mpg	392
Ozone	330	Price	159	Rate	144				

Table 4.2: A table showing the data sets used in our analysis.

For each data set, we generated 20 independent train/test splits, allocating $\frac{4}{5}$ of the data to

the training set in each case. Two variants of the AddiVortes model were evaluated. The first, referred to as *AddiVortes-CV*, selected the prior hyperparameters $(m, \nu, q, k, \sigma_c, \omega, \lambda_c)$ via five-fold cross-validation. The second, denoted *AddiVortes-Def*, employed a fixed configuration based on extensive empirical analysis of both simulated and real-world data sets

$$(m, \nu, q, k, \sigma_c, \omega, \lambda_c) = (200, 6, 0.85, 3, 0.8, 3, 25),$$

as discussed in Section 4.3. For all runs, we applied a burn-in period of 200 iterations, which was found to be sufficient to ensure convergence in most cases.

As competitors, we evaluated four "black-box" methods: random forests (Breiman, 2001, implemented as `randomforest` in R), gradient boosting (Friedman, 2001, implemented as `gbm` in R), BART (Chipman et al., 2010, implemented as `bart` from the BayesTree R package) and SoftBART (Linero and Yang, 2018, implemented as `SoftBart` in R). These models were selected for their robust multivariate regression capabilities, interpretability and comparability.

Method	**Parameters**	**Values considered**
Random forests	Number of trees	500
	% variables sampled to grow each node	10, 25, 50, 100
Gradient boosting	Number of trees	50, 100, 200
	Shrinkage (multiplier of each tree added)	0.01, 0.05, 0.10, 0.25
	Max depth permitted for each tree	1, 2, 3, 4
BART	σ prior: (ν, q) combinations	(3, 0.90), (3, 0.99), (10, 0.75)
	Number trees, m	50, 200
	μ prior: k value for σ_μ	1, 2, 3, 5
SoftBART	Number trees, m	50, 200
	μ prior: k value for σ_μ	1, 2, 3, 5
AddiVortes	# Tessellations: m	50, 200
	σ prior: (ν, q)	(6,0.85)
	μ prior: k value for σ_μ	1, 3
	Standard deviation of centre location: σ_c	0.255, 0.8
	Probability weight for # covariates: ω	3
	Poisson rate for # centres: λ_c	5, 25

Table 4.3: A table showing the cross-validation values for competing methods.

All methods, except AddiVortes-def, underwent 5-fold cross-validation to select hyperparameters. The specific values considered for each method are listed in Table 4.3. For AddiVortes-CV in particular, we explored the following cross-validation values

- $k \in \{1, 3\}$, reflecting moderate to strong shrinkage for the μ prior,
- $\sigma_\mu \in \{0.255, 0.8\}$, representing low to high variability in the coordinates of centres,
- $\lambda_c \in \{5, 25\}$, assessing the effect of a smaller versus larger number of centres per tessellation, and
- $m \in \{50, 200\}$, specifying the number of tessellations in the ensemble,

giving a total of $2^4 = 16$ combinations of $(k, \sigma_\mu, \lambda_c, m)$ for evaluation. All other hyperparameters were held at their default values, as they were found to be less influential on performance, and expanding the search over these would significantly increase computational cost.

In our analysis, cross-validation using RMSE metric was performed using a brute-force method, in which every possible combination of hyperparameters was considered. However, for computational efficiency, we could have used a different method for cross-validation that considered the most likely combinations, avoiding unnecessary algorithm runs.

To precisely define how prediction error is calculated, we evaluate the models using the RMSE based on the posterior predictive mean. Specifically, we do not fix the parameter values at a single point estimate, such as a posterior mode or mean of the components. Instead, for a new test observation $\boldsymbol{x}^*$, the predicted value $\hat{y}^*$ is obtained by averaging the output of the sum-of-tessellations model over the S post-burn-in MCMC samples, given by

$$\hat{y}^* = \frac{1}{S} \sum_{s=1}^{S} G(\boldsymbol{x}^*; \boldsymbol{T}^{(s)}, \boldsymbol{M}^{(s)}).$$

The RMSE is then computed by comparing these averaged predictions against the true observed values in the test set.

To facilitate performance comparisons across datasets, we used the relative RMSE, defined as the RMSE divided by the minimum RMSE achieved by any method for each train/test split. This normalisation allows for fair and interpretable comparisons across data sets, regardless of differences in the scale or location of the response variable. Boxplots of the RRMSE values for each method, aggregated across all train/test splits, are presented in Figure 4.3. Additionally, the 50% and 75% RRMSE quantiles are reported in Table 4.4.

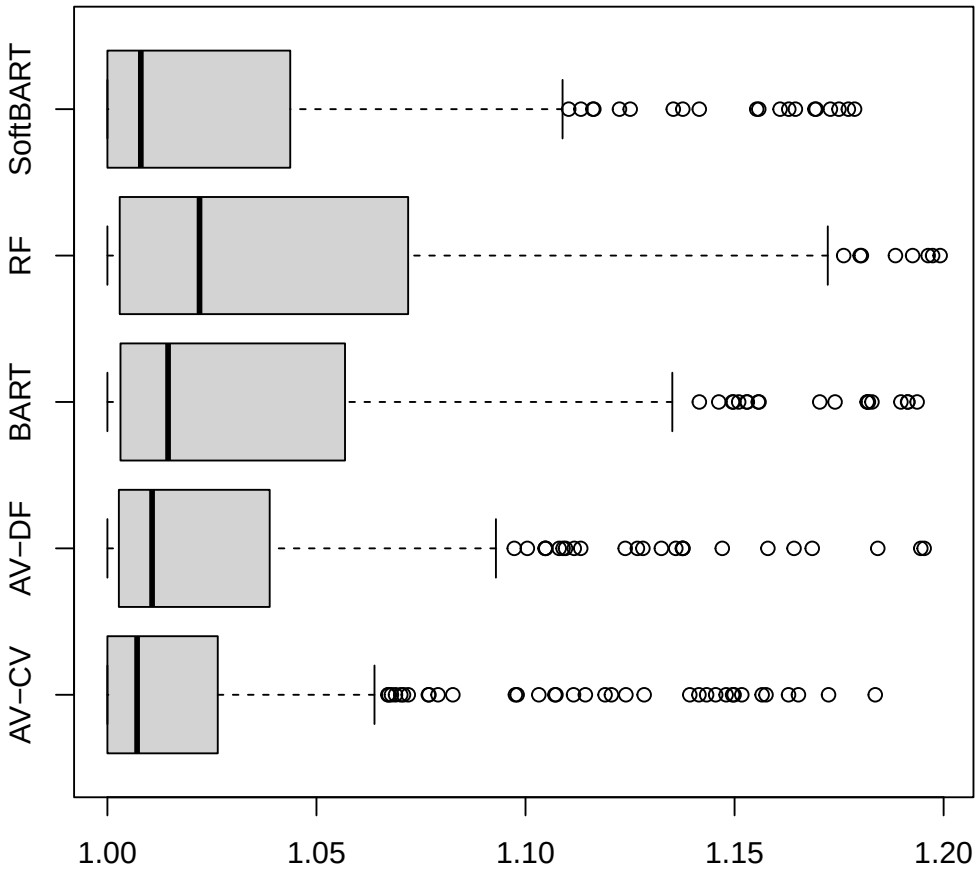


Figure 4.3: Boxplot of RRMSE for competing methods on 13 data sets.

Note that gradient boosting was excluded from the boxplot in Figure 4.3, as its RRMSE values were substantially higher than those of the other methods and would have obscured meaningful comparisons. Additionally, we removed all RRMSE values exceeding 1.5 to improve the figure's interpretability. This filtering did not affect AddiVortes-CV or BART, while only a single value was removed for AddiVortes-Def. In contrast, four values were removed for SoftBART and nine for random forests.

While Figure 4.3 illustrates variation in relative performance across data sets, the overall distribution of RRMSE values indicates that AddiVortes-CV most frequently achieved the lowest

Method	**(50%, 75%)**
AddiVortes-CV	(1.007635, 1.026602)
AddiVortes-Def	(1.011348, 1.028736)
SoftBART	(1.007709, 1.026700)
BART	(1.013905, 1.056225)
Random Forests	(1.036218, 1.100439)
Gradient Boosting	(1.068354, 1.184328)

Table 4.4: (50%, 75%) quantiles of relative RMSE values for each method.

RMSE among the competing methods. Notably, AddiVortes-Def also performed competitively, despite relying solely on fixed hyperparameter settings. This is especially significant given that random forests, SoftBART, and BART all underwent cross-validated hyperparameter tuning, whereas AddiVortes-Def achieved strong results without such optimisation.

4.5 Friedman dataset

We use simulated data to evaluate the performance of AddiVortes against known ground truth and adopt the experimental setup originally proposed in Friedman (1991). The covariate vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$ is generated by drawing each component independently from the standard uniform distribution. The response variable Y is then defined as

$$Y = f(\boldsymbol{x}) + \epsilon = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \epsilon, \tag{4.12}$$

where $\epsilon \sim \mathcal{N}(0, 1)$. Importantly, the true regression function depends only on the first five covariates, $x_1, \ldots, x_5$, while the remaining variables $x_6, \ldots, x_p$ are inactive. The presence of these irrelevant predictors introduces additional complexity, providing a useful test of each model's ability to identify and utilise informative covariates while discarding noise.

4.5.1 Illustrations of findings

To facilitate comparison between the AddiVortes model and BART, we adopt a simulation setup with a similar number of MCMC iterations to those used in Chipman et al. (2010). We begin by illustrating the core features of AddiVortes using a single simulated data set generated from the Friedman function, with $p = 10$ predictors and $n = 150$ observations. To implement AddiVortes,

we apply the default hyperparameter settings, $(m, \nu, q, k, \sigma_c, \omega, \lambda_c) = (200, 6, 0.85, 3, 0.8, 3, 25)$ and generate a total of 1,800 posterior samples of f^* using the MCMC algorithm, discarding the initial 200 iterations as burn-in.

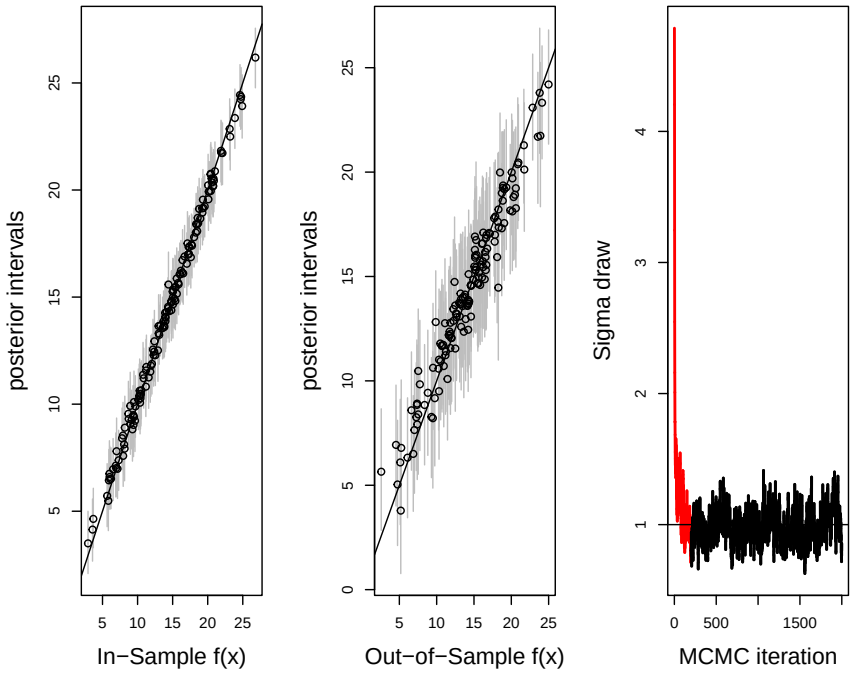


Figure 4.4: Inference about Friedman's $f(\boldsymbol{x})$ for p = 10 dimensions.

For each unique input $\boldsymbol{x}$, we compute posterior mean estimates $\hat{f}(\boldsymbol{x})$ by averaging the 1,800 posterior samples $f^*(\boldsymbol{x})$. To quantify uncertainty, we construct "naive" 90% posterior intervals for each $f(\boldsymbol{x})$ by taking the 5th and 95th percentiles of the posterior samples. Figure 4.4(a) displays the estimated values $\hat{f}(\boldsymbol{x})$ against the true function values $f(\boldsymbol{x})$ for the $n = 150$ in-sample $\boldsymbol{x}$ values used to generate the corresponding Y values via Equation (4.12). Vertical lines indicate the associated 90% posterior intervals. As shown, the estimates $\hat{f}(\boldsymbol{x})$ closely track the true values, with the vast majority of true values falling well within the corresponding intervals.

Figure 4.4(b) extends this comparison to 150 randomly sampled out-of-sample $\boldsymbol{x}$ values. While the correlation between the posterior means $\hat{f}(\boldsymbol{x})$ and the true values $f(\boldsymbol{x})$ is slightly weaker than in the in-sample case, the estimated function still closely approximates the true function over a wide range of the input space. As expected, the posterior intervals are somewhat wider, reflecting the increased uncertainty. This result is particularly encouraging given the limited training data

size ($n = 150$) and the presence of several irrelevant covariates. It highlights the model's capacity to avoid overfitting and to concentrate predictive power on the relevant structure, even when noisy or redundant predictors are present.

It is important to note that the nominal 90% posterior intervals do not necessarily correspond to 90% frequentist coverage. In this experiment, the intervals covered approximately 92% of the true function values for the training data and 98% for the testing data. This suggests that the posterior intervals are generally conservative, particularly in out-of-sample settings. In applications involving real-world data, where the true regression function f is unknown, alternative approaches such as bootstrap resampling or cross-validation may offer empirical assessments of interval coverage.

Additionally, for input values $\boldsymbol{x}$ lying near the edges of the covariate space, the influence of the prior distribution becomes more pronounced. In such cases, posterior estimates tend to shrink more strongly towards the prior mean (that is, zero), which can reduce the empirical coverage of the intervals. This effect underscores the need for careful prior specification and highlights the role of model structure in shaping predictive uncertainty at the boundaries of the input domain.

Figure 4.4(c) displays the sampled values of σ across MCMC iterations, with the true value $\sigma = 1$ indicated by a horizontal reference line. The plot illustrates that the Markov chain quickly converges to an equilibrium, with the sampled σ values fluctuating around the true value. This behaviour provides evidence that the model is not overfitting the data as the posterior uncertainty in σ is consistent with the known noise level.

Figure 4.5 presents the average number of dimensions (left) and the average number of centres (right) per tessellation across MCMC iterations for the simulation described above. The plots indicate rapid convergence of the Markov chain, with the average number of dimensions stabilising around 2.4 and the average number of centres around 3.5. However, the continued oscillation around these values suggests that the sampler is still actively exploring the posterior space. This behaviour suggests good mixing and supports the notion that the model is not prematurely settling into a restricted region of the parameter space, thereby allowing for better posterior sampling.

Figure 4.6 illustrates the percentage of accepted tessellations per iteration. The acceptance rate is exceptionally high during the initial iterations. This occurs because the initial sigma estimation is an large overestimation and the tessellations are initialised with a very simple structure, consisting of a single dimension and a centre. Therefore, any early proposal that increases the model's

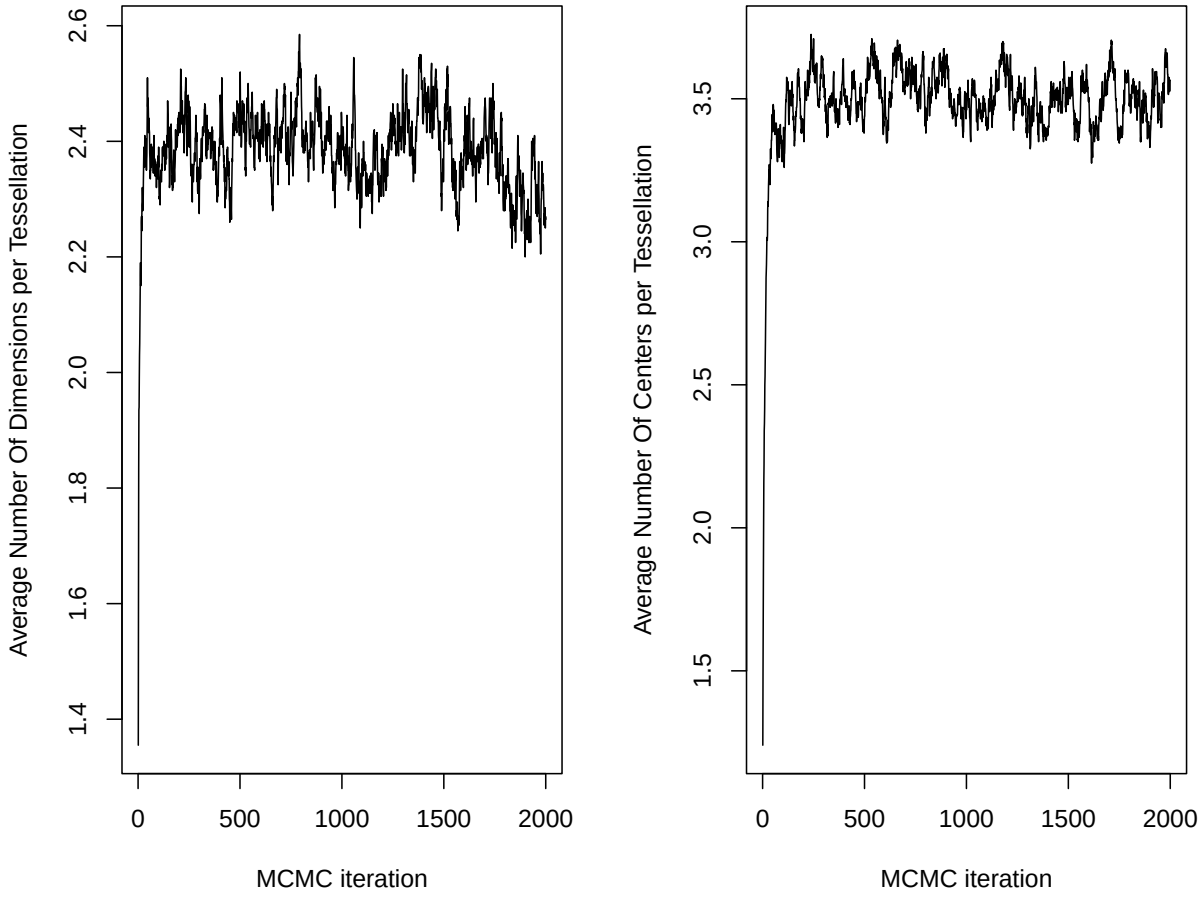


Figure 4.5: The average number of dimensions (left) and centres (right) per tessellation for each iteration.

complexity has a high probability of improving its predictive performance and is consequently accepted. As the algorithm proceeds, the acceptance rate stabilises at approximately 30%.

This behaviour compares favourably with the standard BART algorithm, as shown in Inglis et al. (2022). Although the average acceptance rate is similar, the original BART algorithm often has iterations in which no proposed trees are accepted. The consistent acceptance rate in our method, however, may suggest that the algorithm is more efficiently exploring the parameter space, then the original BART model. That is, the proposed modifications to the tessellation are more consistently constructive, resulting in fewer wasted iterations and a more effective search for the optimal model configuration.

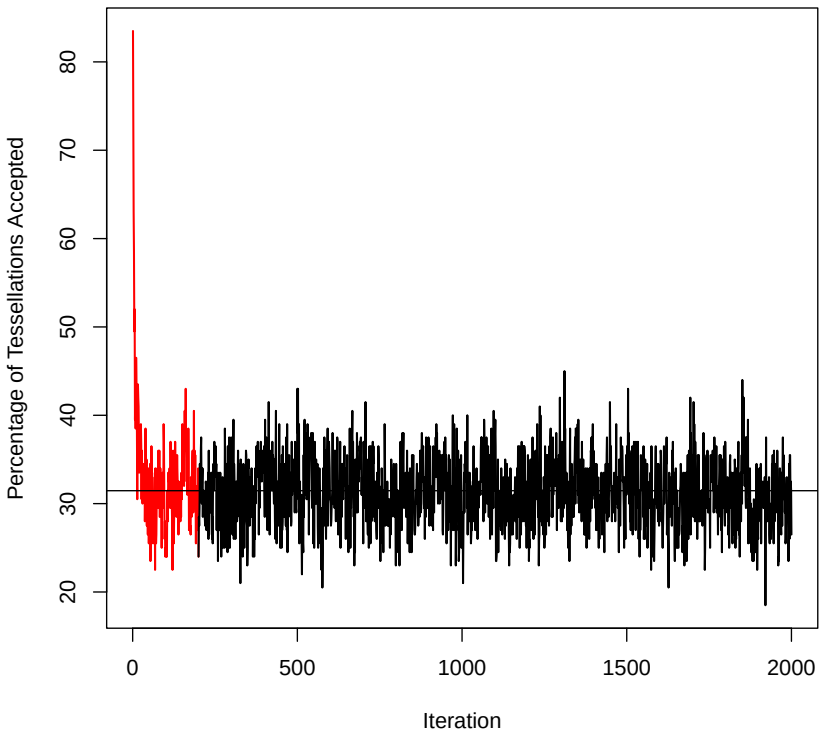


Figure 4.6: Percentage of tessellations accepted per iteration with red section illustrating burn-in period of 200 iterations.

The diagnostic plots presented in Figure 4.7 provide strong evidence validating the AddiVortes model's fit to the simulated data. An examination of the predicted values plotted against their corresponding residuals reveals a random, unstructured scatter of points. The absence of any discernible pattern, confirms that the error variance is constant across all predicted values, a property known as homoscedasticity. This result strongly indicates that the AddiVortes model provides an unbiased fit, having successfully captured the underlying data-generating process without introducing systematic errors.

Furthermore, the distribution of the residuals was assessed to ensure it meets key statistical assumptions. A histogram of the residuals displays a symmetric, bell-shaped curve, which provides a clear visual indication of normality. This observation is rigorously corroborated by the Quantile-Quantile (QQ) plot, where the residual quantiles align almost perfectly with the $y = x$ theoretical reference line. This close conformity confirms that the error distribution is normal. Collectively,

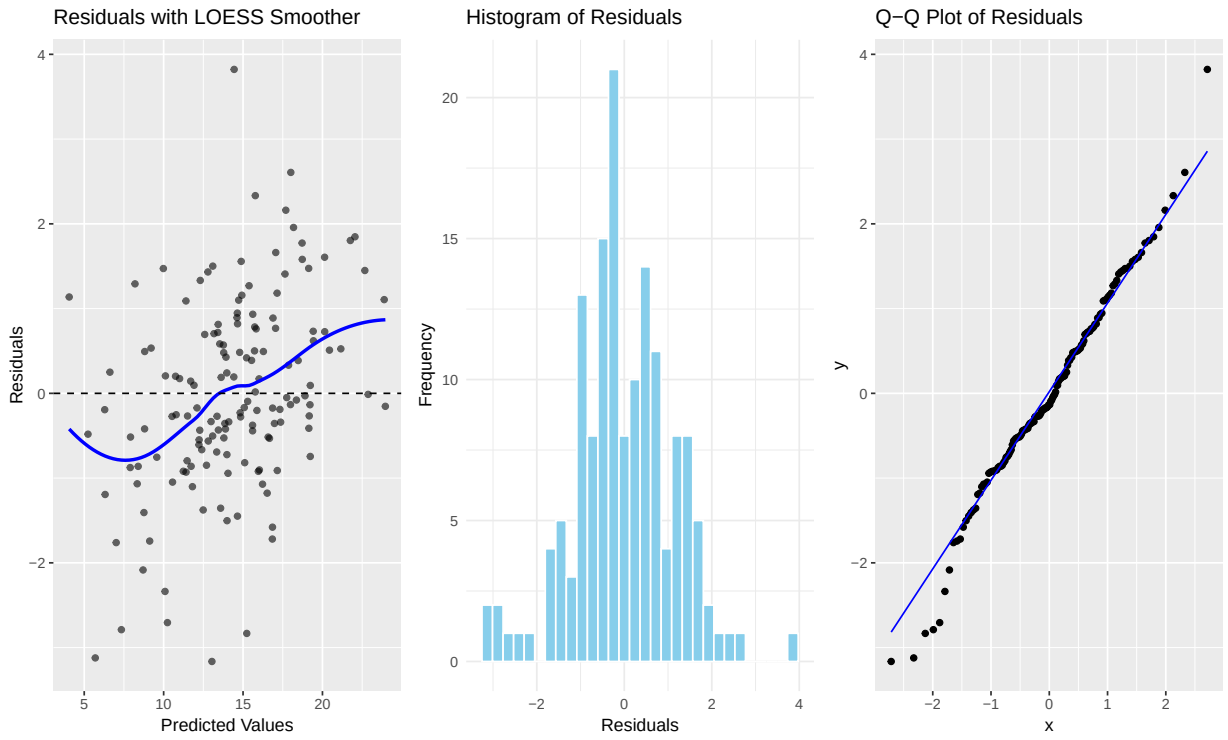


Figure 4.7: Residual Graphs.

these diagnostic checks demonstrate that the model is well-specified and that the foundational assumptions for valid statistical inference are satisfied.

The partial dependence function (Friedman, 2001) is a valuable tool for interpreting complex models. It isolates and summarises the marginal effect of a specific covariate, $\boldsymbol{x_j}$, on the predicted response by averaging out the effects of all other covariates, denoted as $\boldsymbol{x_{-j}}$. Mathematically, if $G(\boldsymbol{x})$ is the prediction function, the partial dependence on x_j is defined as the expected value over the marginal distribution of $\boldsymbol{x}_{-j}$,

$$f_j(x_j) = \mathbb{E}_{\boldsymbol{x}_{-j}}[G(\boldsymbol{x}_j, \boldsymbol{x}_{-j})].$$

In practice, because the true joint distribution of the covariates is unknown, this integral is approximated using the empirical distribution of the training data. For a specific evaluation point x_j, the model's prediction is calculated across all n observations by fixing the target covariate at x_j while leaving the complementary variables exactly as they appear in the dataset. The estimated

partial dependence is then computed as the sample average,

$$\hat{f}_j(x_j) = \frac{1}{n}\sum_{i=1}^{n} G(x_j, \boldsymbol{x}_{i,-j}).$$

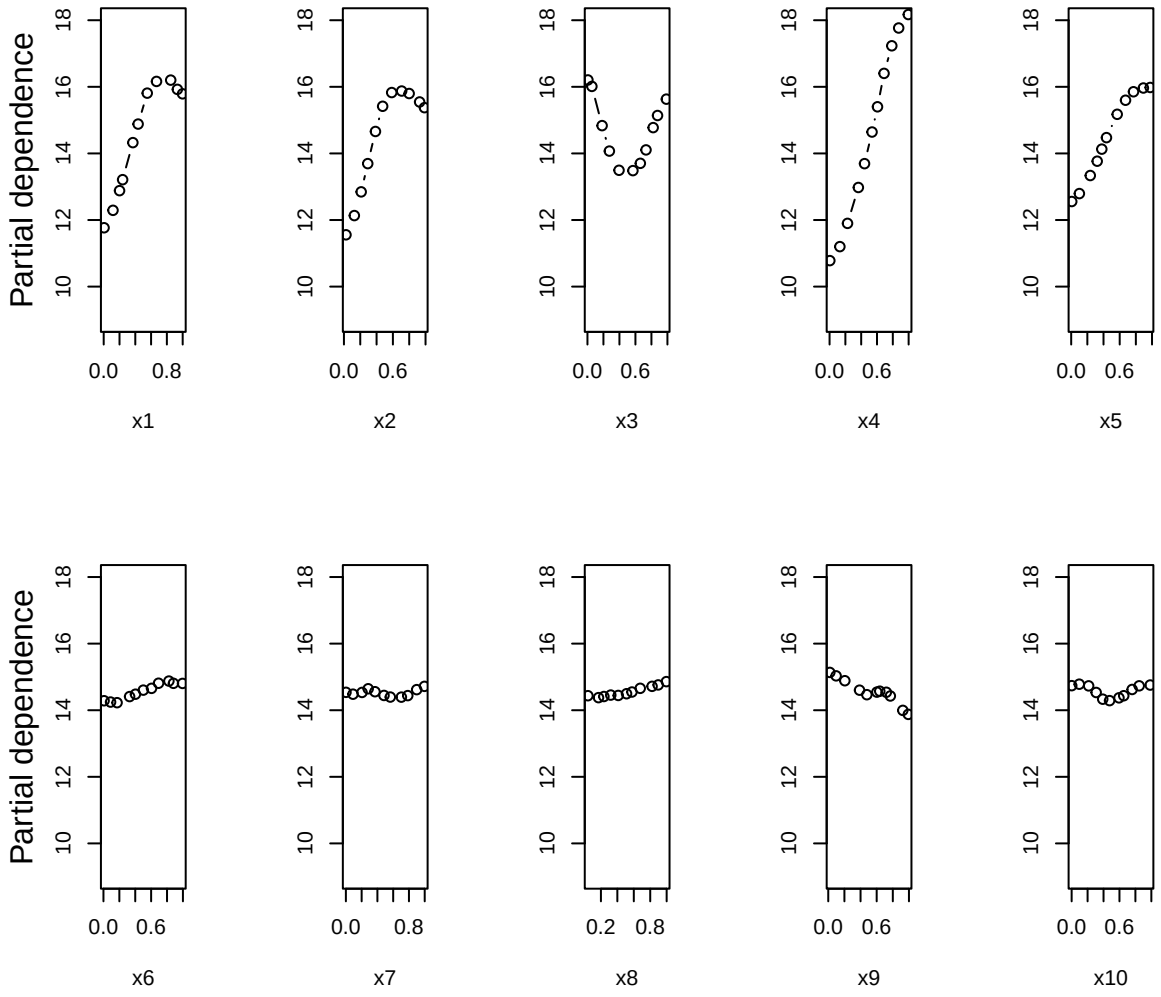


Figure 4.8: Partial dependence plots for the 10 predictors in the Friedman function.

AddiVortes can be used to estimate the partial dependence function, which summarises the marginal effect of an individual covariate x_i on the response Y. Figure 4.8 displays point estimates of the partial dependence functions for $x_1, \ldots, x_{10}$, based on 5,000 posterior samples. The plots clearly reveal strong marginal effects for the active variables $x_1, \ldots, x_5$, consistent with the true data-generating mechanism. In contrast, the estimated effects for the inactive variables $x_6, \ldots, x_{10}$

are essentially flat, indicating that the model has successfully identified the irrelevance of these covariates. This highlights the model's ability to capture meaningful structure while remaining robust to noise.

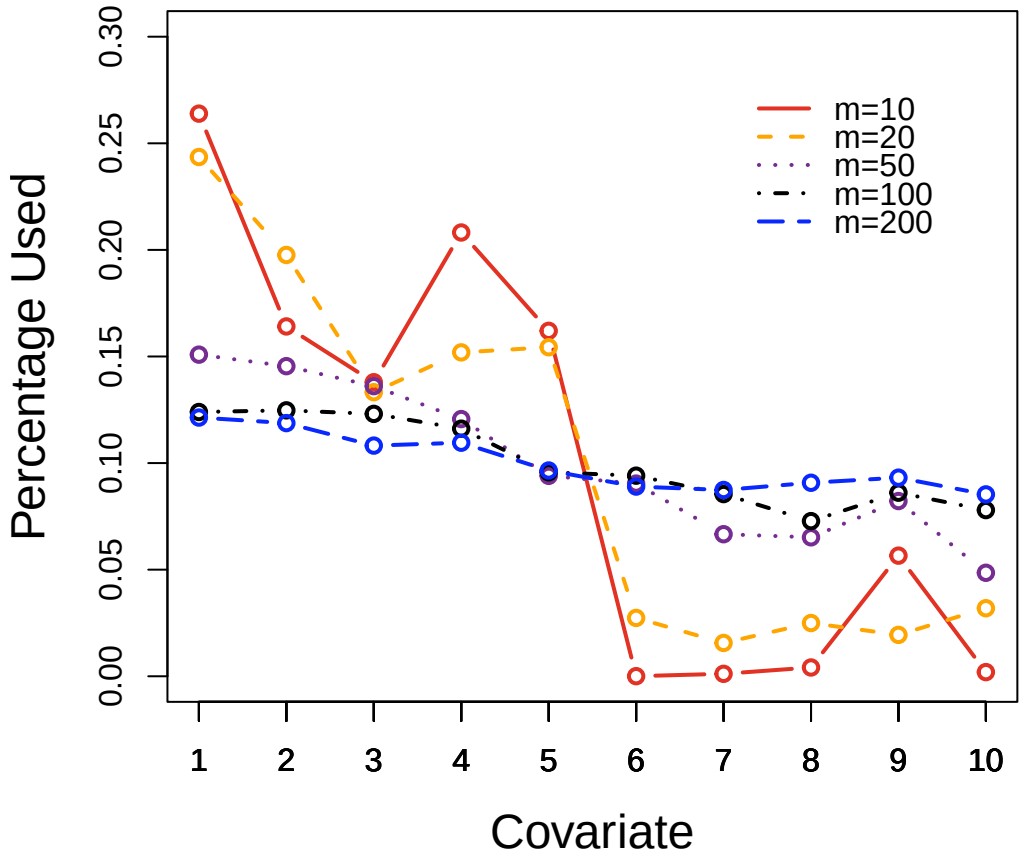


Figure 4.9: A graph showing the percentage of the covariates, $x_1, \ldots, x_{10}$, that are most frequently included in accepted tessellations.

In addition to its predictive capabilities, AddiVortes can be employed as a variable selection tool by tracking the covariates most frequently included in accepted tessellations. Figure 4.9 illustrates the percentage of inclusion for each covariate $x_1, \ldots, x_{10}$ across 5,000 posterior draws f^*, evaluated for a range of ensemble sizes $m \in \{10, 20, 50, 100, 200\}$. The data set consists of $n = 500$ simulated observations from the Friedman function (4.12) with $p = 10$, aligning with the experimental setup used in the original BART paper.

As the number of tessellations m decreases, the sum-of-tessellations ensemble becomes more selective, increasingly concentrating on the truly relevant covariates $x_1, \ldots, x_5$ that drive the variation in Y. This behaviour occurs naturally without imposing any prior knowledge of the true functional form of f. These results demonstrate that AddiVortes not only achieves strong predictive performance but also provides an effective, data-driven mechanism for identifying the underlying structure of the regression function.

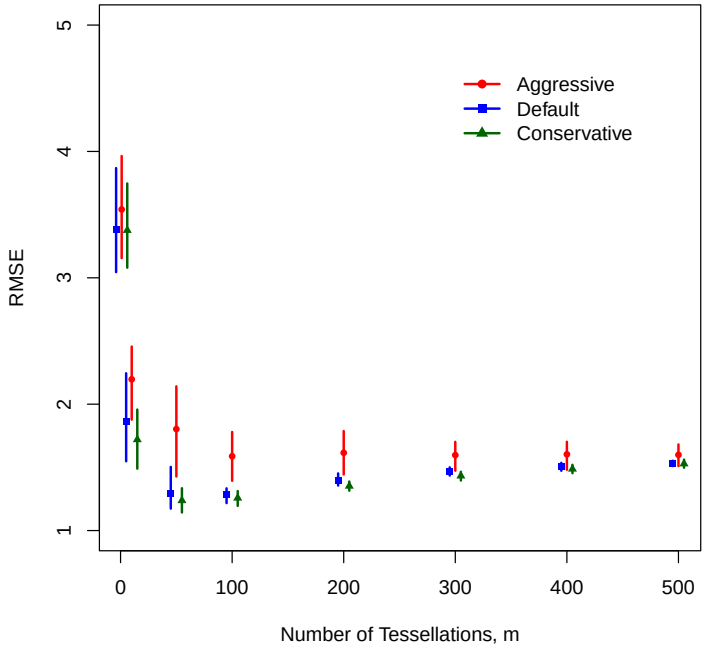


Figure 4.10: RMSE values and 90% intervals as m increases from 1 to 500 under three hyperparameter settings: aggressive (circle), default (square), and conservative (triangle). Points are slightly offset for visual clarity.

AddiVortes exhibits robustness to changes in its hyperparameter settings. To illustrate this, we simulate 150 observations from the Friedman function with $p = 10$, and compute the RMSE on 150 out-of-sample test points. AddiVortes is run 100 times under three distinct hyperparameter configurations: *aggressive*, *default*, and *conservative*.

The aggressive configuration, $(\nu, q, k, \sigma_c, \omega, \lambda_c) = (10, 0.75, 3, 0.8, 1, 5)$, imposes stronger shrinkage towards zero and favours lower-dimensional tessellations with fewer centres. In contrast, the conservative setting, $(\nu, q, k, \sigma_c, \omega, \lambda_c) = (6, 0.95, 1, 1.5, 3, 25)$, reflects weaker shrinkage and encour-

ages the use of more complex tessellations involving higher dimensionality and more centres. The default configuration, $(6, 0.85, 3, 0.8, 3, 25)$, lies between these two extremes.

Figure 4.10 shows the RMSEs across these three configurations for varying numbers of tessellations $m \in [1, 500]$, along with the associated 90% intervals. Across all values of m, the RMSEs remain relatively close across settings, demonstrating the model's stability. The aggressive setting performs noticeably worse for small values of m, where limited model capacity exacerbates the effects of over-regularisation.

The figure also shows a trend in RMSE with the number of tessellations. As m increases from 1 to 50, predictive accuracy improves significantly. Beyond $m = 200$, however, RMSE begins to degrade slightly, which may reflect increasing uncertainty due to over-partitioning within individual tessellations. Notably, once $m > 100$, the 90% intervals narrow significantly, indicating that repeated algorithm runs give consistent results. This suggests the posterior variability across runs is small for larger m.

Next, we investigate the performance of AddiVortes in higher-dimensional settings, using the same underlying function f from Equation (4.12), which depends only on the first five covariates $x_1, \ldots, x_5$. Building on our earlier analysis with $p = 10$ and $n = 150$ observations, we now consider larger values of p while keeping n fixed. These simulations are designed to assess AddiVortes' ability to detect low-dimensional structure in the presence of many irrelevant covariates.

We replicate the illustrations presented in Figure 4.4 for $p = 20$, 100, and 1000. These examples demonstrate that, even in extremely high-dimensional settings with a small sample size, AddiVortes can isolate the relevant covariates and produce accurate predictions without overfitting to noise introduced by inactive variables.

For $p = 20$, we used hyperparameter values similar to the default setting, except with a reduced number of tessellations: $m = 50$ instead of 200. For the higher-dimensional cases $p = 100$ and $p = 1000$, we used the configuration $(m, \nu, q, k, \sigma_c, \omega, \lambda_c) = (50, 6, 0.99, 3, 0.2, 3, 25)$. In these settings, we anchored the prior on the noise standard deviation σ using the naive estimator $\hat{\sigma}$ (i.e., the sample standard deviation of Y), rather than the least squares estimate. This adjustment avoids complications when $p \geq n$, where the linear regression model becomes undefined.

Figure 4.11 illustrates the performance of AddiVortes on both in-sample and out-of-sample data as the dimensionality p increases. Corresponding coverage rates for the 90% posterior intervals

are reported in Table 4.5. Across all values of p, the in-sample predictions and their associated uncertainty intervals remain highly accurate. In the out-of-sample setting, we observe a mild shrinkage effect: extreme values of $f(\boldsymbol{x})$ are drawn slightly toward the mean as p increases. Despite this, even with a small training sample and a large number of irrelevant covariates, the model maintains good prediction quality and effective uncertainty quantification.

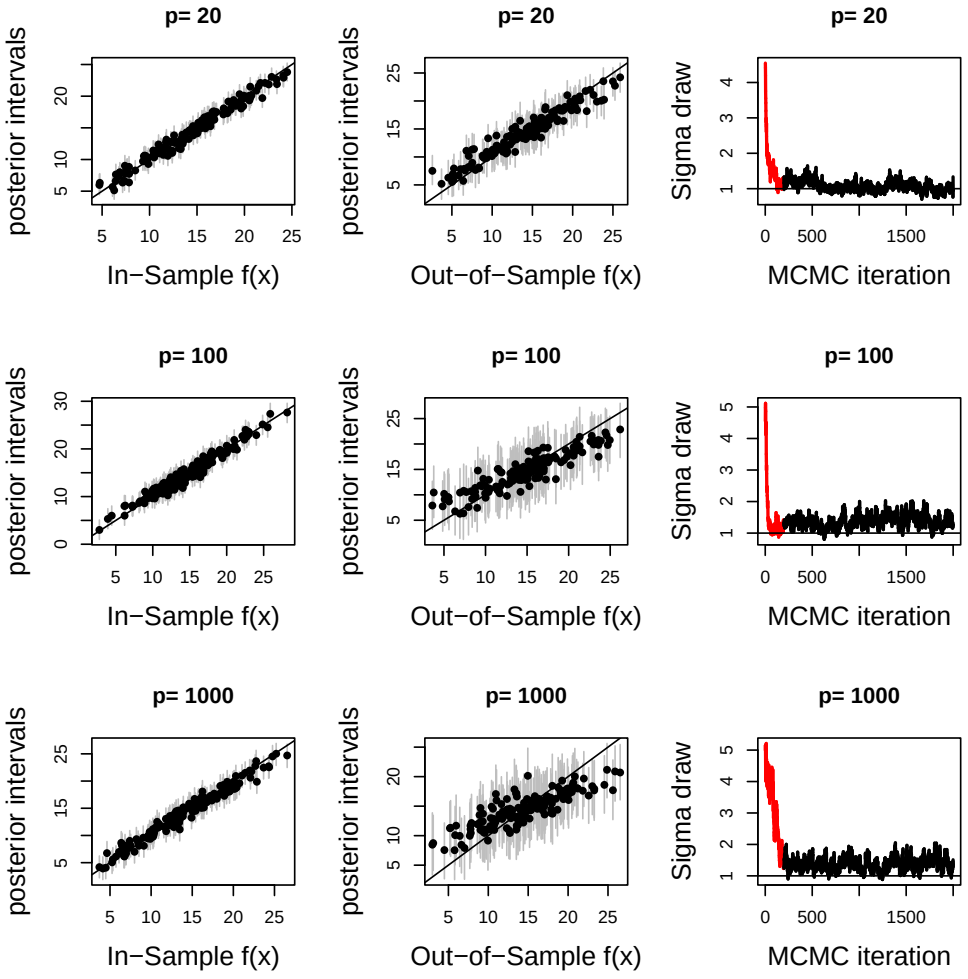


Figure 4.11: Inference about Friedman's function for $p = 20$, 100, 1000 dimensions.

The third column of Figure 4.11 presents the MCMC trace plots of the sampled values of σ, with a horizontal line marking the true value $\sigma = 1$. These plots show that the posterior draws of σ frequently fluctuate around the true value. However, as the dimensionality p increases, the estimates tend to drift toward larger values, reflecting increased posterior uncertainty. This behaviour is expected in high-dimensional settings, where the model must account for greater

p	In-Sample	Out-of-Sample
20	95%	92%
100	97%	98%
1000	93%	90%

Table 4.5: Coverage for the posterior intervals with increasing number of inactive covariates.

variability and potential noise introduced by the large number of inactive covariates.

A particularly appealing feature of AddiVortes is its robustness to the effects of inactive covariates. To investigate this, we simulated $n = 150$ observations under a null model with $f \equiv 0$, for three levels of dimensionality: $p = 10$, 100, and 1000. Under these settings, AddiVortes produced posterior intervals for f that, in both in-sample and out-of-sample settings, were centred around zero for $p = 10$ and $p = 100$, correctly indicating the absence of any signal.

In the most challenging case, $p = 1000$, where the data contain no predictive structure and the number of covariates greatly exceeds the sample size, some in-sample intervals deviated from zero, reflecting a modest influence of the prior. Nevertheless, the out-of-sample 90% posterior intervals consistently included zero, demonstrating that AddiVortes remains cautious in its predictions when no signal is present, and effectively avoids overfitting in ultra-sparse settings.

To illustrate the flexible partitions created by Voronoi tessellations, we conduct a simulated experiment using a simplified version of the Friedman function. The covariate vector $\boldsymbol{x} = (x_1, x_2)$ is generated by drawing each component independently from the standard uniform distribution, $U(0, 1)$. We then define the response variable Y as

$$Y = f(\boldsymbol{x}) + \epsilon = \sin(\pi x_1 x_2) + \epsilon, \tag{4.13}$$

where $\epsilon \sim \mathcal{N}(0, 1)$. This problem is constrained to two dimensions to allow for straightforward visualisation of the partitions. Figure 4.12 illustrates three randomly chosen tessellations from iterations 20, 50, and 100 of the AddiVortes algorithm.

These plots demonstrate the highly flexible, non-axis-aligned partitions produced by Voronoi tessellations. A notable characteristic observed in the figure is that many of the Voronoi centres are located outside the $[-0.5, 0.5]^2$ data domain. This phenomenon is likely a consequence of the MCMC sampling procedure. Minor perturbations to centres located outside the data range result in

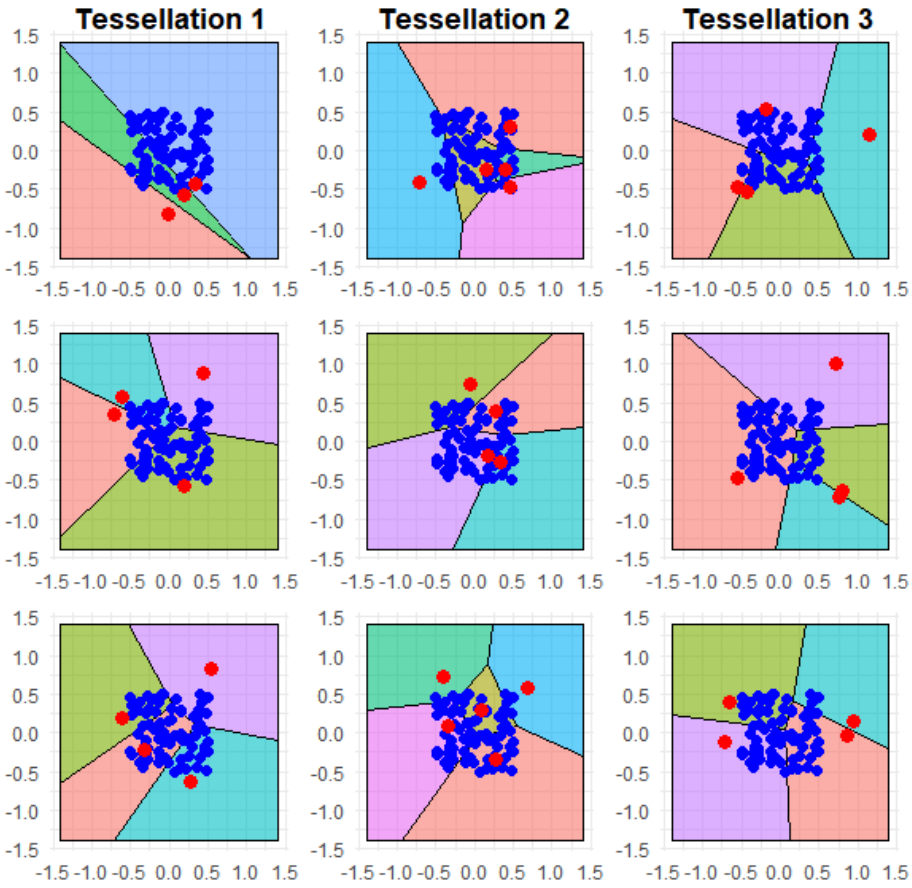


Figure 4.12: Partitions by 3 tessellations at iterations 20, 50 and 100 (top, middle and bottom rows, respectively) for simplified simulated data. The red points are the centres, and the blue points represent the training data.

only small adjustments to the partitions within the data's active range, leading to stable acceptance rates. Conversely, small movements of centres located deep within the data domain could induce dramatic changes to the partitions, increasing the likelihood of rejection.

4.5.2 Evaluating out-of-sample performance across competing approaches

To assess the effectiveness of AddiVortes within the Friedman setup, we conducted a simulation study comparing its performance to that of random forests, BART, gradient boosting, and Soft-BART. The goal was to estimate the underlying function f using $n = 200$ observations with $p = 10$ covariates.

For the AddiVortes model, we ran 1,000 MCMC iterations, discarding the first 200 draws as burn-in. Hyperparameter selection was performed using five-fold cross-validation over the candidate values specified in Table 4.3. This setup enables a fair comparison with the competing methods, all of which also underwent cross-validation to select their tuning parameters.

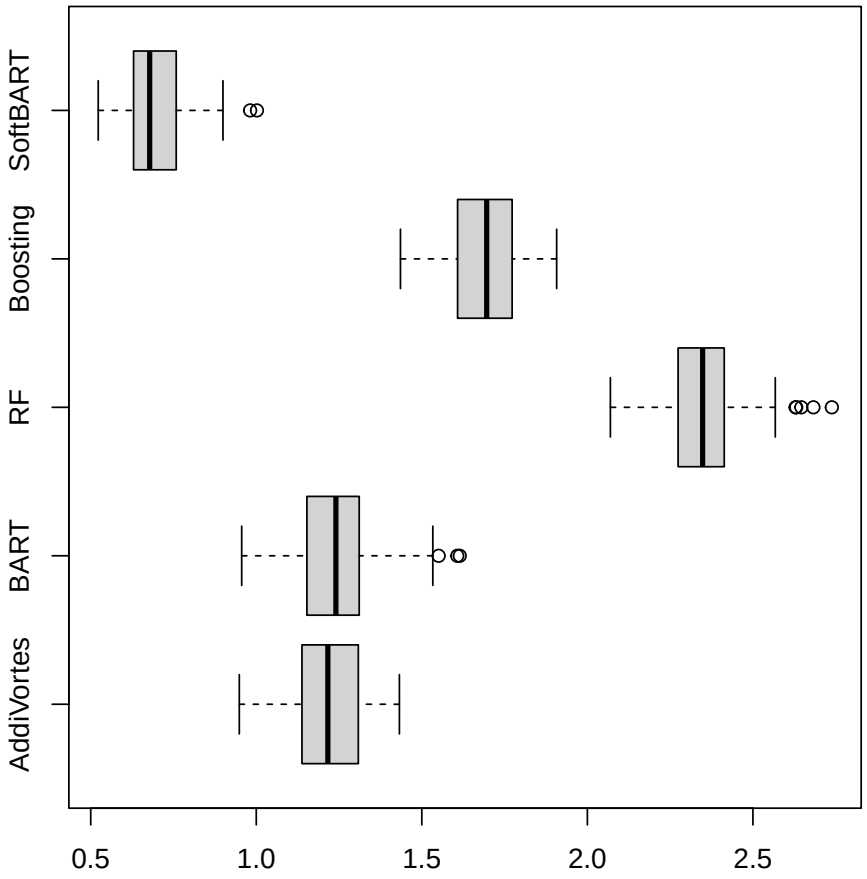


Figure 4.13: RMSE values for Friedman's function for $p = 10$ covariates.

We simulated 100 independent data sets, each consisting of $n = 200$ observations. Since the data-generating process is fully known, there was no need to simulate a separate test set. Instead, for each fitted function $\hat{f}$ produced by a given method on each data set, we generated 800 independent covariate vectors $\boldsymbol{x}$ and used them to evaluate predictive accuracy. The root mean squared error

(RMSE) was computed as

$$\text{RMSE} = \sqrt{\frac{1}{800}\sum_{i=1}^{800}\left(\hat{f}(\boldsymbol{x}_i) - f(\boldsymbol{x}_i)\right)^2}.$$

This gives 100 RMSE values for each method, enabling a robust comparison of predictive performance across the different models.

Figure 4.13 presents the results of this experiment using a boxplot of RMSE values (note that, unlike Figure 4.3, this figure displays absolute RMSE values rather than relative RMSE). As expected, SoftBART achieves the strongest performance among the competing methods. This can be attributed to its use of a sparsity-inducing prior for variable selection within the tree structure, as proposed by Linero (2016), a modification that could also be integrated into the AddiVortes framework.

AddiVortes performs slightly better than BART, with both methods substantially outperforming gradient boosting and random forests. These results highlight AddiVortes' capacity to effectively identify relevant covariates and produce accurate predictions, even in the presence of potential noise. With further methodological development, such as incorporating sparsity priors or improving proposal mechanisms, AddiVortes has the potential to match or exceed the performance of state-of-the-art methods like SoftBART.

4.6 Computational time

We are continuously developing the AddiVortes package to improve computational efficiency and to introduce methodological variants that perform better under different modelling scenarios. The most computationally intensive task in the algorithm is determining the index of the nearest centre for each observation. This operation is required m times per iteration to identify the centres of the newly proposed tessellation and is subsequently executed $m \times \#$(posterior samples) times during training, when allocating observations to their corresponding cells. As the number of observations increases, the nearest-centre search can become a major computational bottleneck, with its cost scaling linearly in the number of data points.

To effectively mitigate the computational cost, the algorithm employs several complementary

strategies designed to enhance efficiency, with the primary method involving the use of k-d trees to accelerate the nearest-centre computation, as described in Section 3.2. This approach exploits the inherent independence of each observation's nearest-centre search, replacing the need for brute-force vectorisation by efficiently partitioning the data space. By constructing and leveraging the k-d tree structure, the algorithm significantly reduces the number of distance calculations required, thereby achieving substantial runtime reductions.

The predict function, which is designed to generate predictions for new observations, is capable of leveraging parallel processing efficiency by automatically detecting the total number of available processing cores, which allows it to distribute the computational workload. The process of predicting the outcome for any single observation is non-sequential and entirely independent of all other observations. This inherent independence allows the workload to be parallelised, eliminating data dependencies that would otherwise constrain computation.

Running multiple MCMC chains simultaneously is beneficial because it allows for better assurance of convergence to the true posterior distribution and provides a check on the sampler's performance. If multiple chains, started from dispersed initial points, all converge to the same distribution, it strengthens the confidence that the target distribution has been adequately reached. Additionally, using multiple chains effectively increases the total number of samples drawn from the posterior in a given time, which is particularly useful for improving the effective sample size and providing a more robust estimate of summary statistics and uncertainty measures, as suggested by Carnegie (2019). This approach inherently benefits from parallel processing, as the computations for each chain are independent and can be run across multiple cores to maximise throughput.

Furthermore, the most computationally critical components of the algorithm are implemented in C++. This allows seamless integration with R while maintaining the low-level efficiency of compiled code. All unnecessary memory allocations and object copies have been eliminated, and preallocation of key data structures ensures minimal overhead per iteration. The use of templated C++ functions enables efficient handling of different data types without redundant code duplication, while modern compiler optimisations further enhance performance.

Taken together, these design choices substantially reduce computational cost, making AddiVortes scalable to larger datasets and allowing the user to run multiple chains. The resulting implementation achieves a balance between flexibility and speed, enabling practical Bayesian infer-

ence for models that would otherwise be prohibitively expensive to compute.

Despite the implementation of several previous optimisations to the underlying code, the AddiVortes framework is currently undergoing continued refinement and optimisation with the explicit goal of further reducing and minimising its computational overhead, ensuring its efficiency and practical scalability are continually improved.

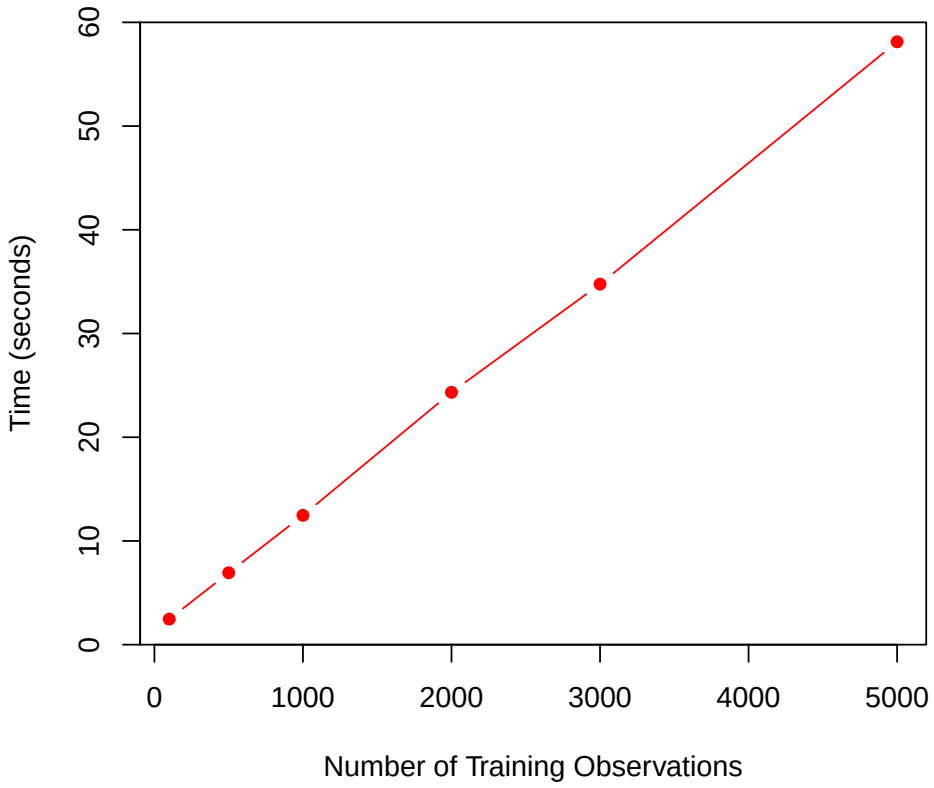


Figure 4.14: Time to train AddiVortes on Friedman set-up with varying number of observations.

The latest version of the AddiVortes framework's code is used to analyse the computational cost using the established Friedman function simulation setup, specifically by training the model on a varying number of observations ranging from 100 to 5,000. For this task, the model employed five active predictors and five inactive predictors. The model was run with a burn-in period of 1,000 iterations followed by the collection of 1,000 post-burn-in samples for precise inference. All executions were performed on a Galaxy Book2 Pro 360, which runs a Windows operating system and features an Intel Core i7-1260P processor with a hybrid 12-core architecture and 16GB of

high-speed LPDDR5 RAM allowing the study to properly assess performance under typical user conditions. The primary finding from this experiment, which is visually represented in Figure 4.14, demonstrated that the algorithm has a near-linear scaling of computational time with respect to the increasing number of observations in the training set.

The fact that the algorithm, even in its current state of ongoing refinement, already maintains a commendably low computational overhead suggests its future scalability. Further optimisations within the code mean that a final runtime performance is highly likely to become competitive with established, optimised tree-based methods such as BART. This suggests AddiVortes is well-positioned for practical scalability across larger datasets in real-world applications.

4.7 Discussion

The AddiVortes framework is based on the novel idea of substituting the axis-aligned partitions of conventional decision trees with the flexible, non-orthogonal cell boundaries of Voronoi tessellations as the core basis function. This means AddiVortes can capture more complex, high-order interactions and relationships that are poorly represented by rectangular, axis-parallel splits created by decision trees.

The model is built as a sum-of-tessellations, maintaining the robust, regularising nature of Bayesian ensemble methods. Each tessellation contributes incrementally to the overall prediction, ensuring that the model remains a collection of "weak learners", thereby preventing overfitting. The ability of each tessellation to incorporate only a low-dimensional subset of covariates allows the model to isolate and approximate individual marginal and interaction effects, which collectively capture the regression function.

The success of the AddiVortes model is mainly due to the highly efficient Bayesian backfitting MCMC algorithm. The use of conjugate normal priors on the cell output values, μ_{ij}, which enables the analytical marginalisation of these parameters during the Metropolis-Hastings acceptance step for the tessellation structure. This allows us to reduce the computationally complexity required of reversible jump MCMC for changes in model dimensionality, making the sampler fast and effective even for high-dimensional problems. The algorithm consistently gave reliable posterior mean and credible interval estimates in simulation experiments, such as the Friedman dataset, accurately cap-

turing the true regression function while demonstrating robustness to hyperparameter specification across various configurations.

Empirically, the model's success is demonstrated across a range of evaluations. When benchmarked against numerous diverse datasets, AddiVortes consistently provided out-of-sample predictive RMSE that was better to that of established methods such as gradient boosting, BART, and random forests. This predictive excellence is complemented by its use as a variable selection tool, where the influence of a covariate is measured by its posterior inclusion frequency across the ensemble. By adjusting the number of tessellations, the model naturally pushes the inclusion proportion to concentrate on the truly influential covariates. Furthermore, the framework successfully addresses the common challenge of computational scalability often associated with Voronoi models, as demonstrated by the near-linear growth of computational time observed in simulations as the sample size increased.

Finally, the chapter lays the groundwork for further theoretical expansions to address current limitations and expand the model's application range. As discussed in Chapter 8, further research includes integrating smoothing mechanisms to overcome the piecewise-constant nature of the tessellation output, and on enabling fundamental partition changes by exploring non-Euclidean metric spaces, such as the Manhattan distance or spherical metrics. These innovations are crucial for further enhancing the model's capacity to flexibly partition the covariate space in a manner best suited to the underlying geometry of complex data structures. The consistent performance metrics and strong theoretical foundation confirm AddiVortes as a significant advancement in non-parametric Bayesian regression.

Chapter 5

Permutation encoding

Nominal covariates, categorical variables without a natural ordering of their levels, are common in many areas of applied statistics, including the social sciences, biomedical research, environmental modelling, and marketing analytics. Examples include treatment groups in clinical trials, product categories in consumer studies, and geographical regions in spatial analyses. While such variables often contain important predictive information, they present a challenge in the context of modern "black-box" statistical models such as random forests (Breiman, 2001), gradient boosting machines (Friedman, 2001), and BART (Chipman et al., 2010). These methods are designed to learn flexible relationships between predictors and responses, but their performance can be sensitive to the way nominal covariates are encoded.

A key difficulty is that nominal variables lack a canonical numerical representation. As seen in Section 2.2, standard encoding strategies, such as one-hot (dummy variable) encoding, treat each level as independent, which can lead to high-dimensional feature spaces, especially when the number of categories is large. This can result in increased variance, reduced interpretability, and slower computation. Conversely, imposing an arbitrary integer ordering on the levels risks introducing spurious ordinal structure, potentially biasing the model. In practice, the choice of encoding can have a substantial impact on predictive accuracy and on the model's ability to detect meaningful patterns Potdar et al. (2017).

In parametric models, solutions such as hierarchical modelling and partial pooling are well established for handling nominal predictors (Efron and Morris, 1977). However, in flexible, non-parametric, and "black-box" models, principled approaches for adaptively leveraging similarities

across categories are less developed. Existing techniques often fail to fully exploit the potential benefits of pooling information across levels, or they require manual pre-specification of category groupings. This gap is particularly consequential when some levels of a nominal covariate are statistically indistinguishable in their effect on the response, as pooling can improve estimation efficiency without sacrificing flexibility.

The aim of this chapter is to develop and evaluate methods for incorporating nominal covariates into statistical "black-box" models that respect their unordered nature while allowing the model to discover and exploit structure among the levels. We introduce a permutation-encoding framework that enables the model to explore alternative category orderings within MCMC algorithm, coupled with an adaptive weighting scheme that favours orderings that improve predictive performance. This approach integrates naturally into existing MCMC-based algorithms for Bayesian ensemble models and preserves the theoretical validity of the sampling scheme.

Through a series of real-data experiments, we demonstrate that the proposed method can recover meaningful groupings of nominal levels, improve predictive accuracy, and adapt to diverse datasets. Our results highlight the importance of carefully handling nominal covariates in "black-box" models and provide a general framework for doing so without requiring model-specific modifications.

5.1 Motivation

If there is a relationship or inherent ordering between classes that affects the output variable and is unknown to the user, a "black-box" algorithm must learn it to improve predictive accuracy.

Partition techniques used by ensemble methods usually perform better when categories with similar influence on the output variable are adjacent in the encoding. For example, for ordinal categorical variables, one would encode the covariate with respect to the order of the covariate; for example, satisfactory rating with categories {bad, neutral, good} could intuitively be encoded as 1,2,3, respectively, as there is an inherent order and categories adjacent to each other are more likely to have similar effects.

Most commonly used encoding strategies for nominal covariates are stationary, such as target or label encoding, in which each category is assigned a fixed numerical representation throughout

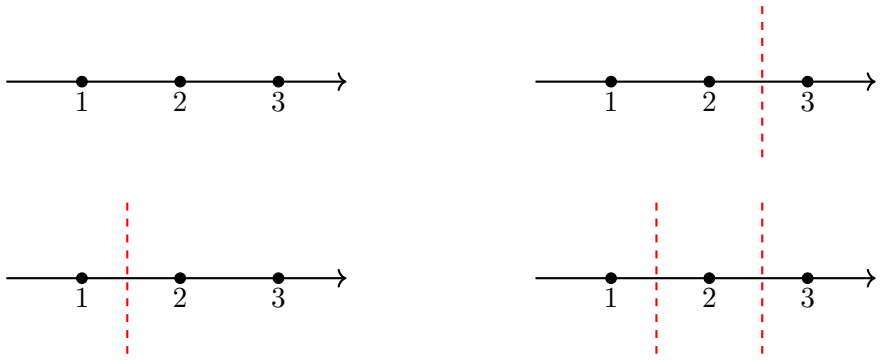


Figure 5.1: Illustration of partition groups.

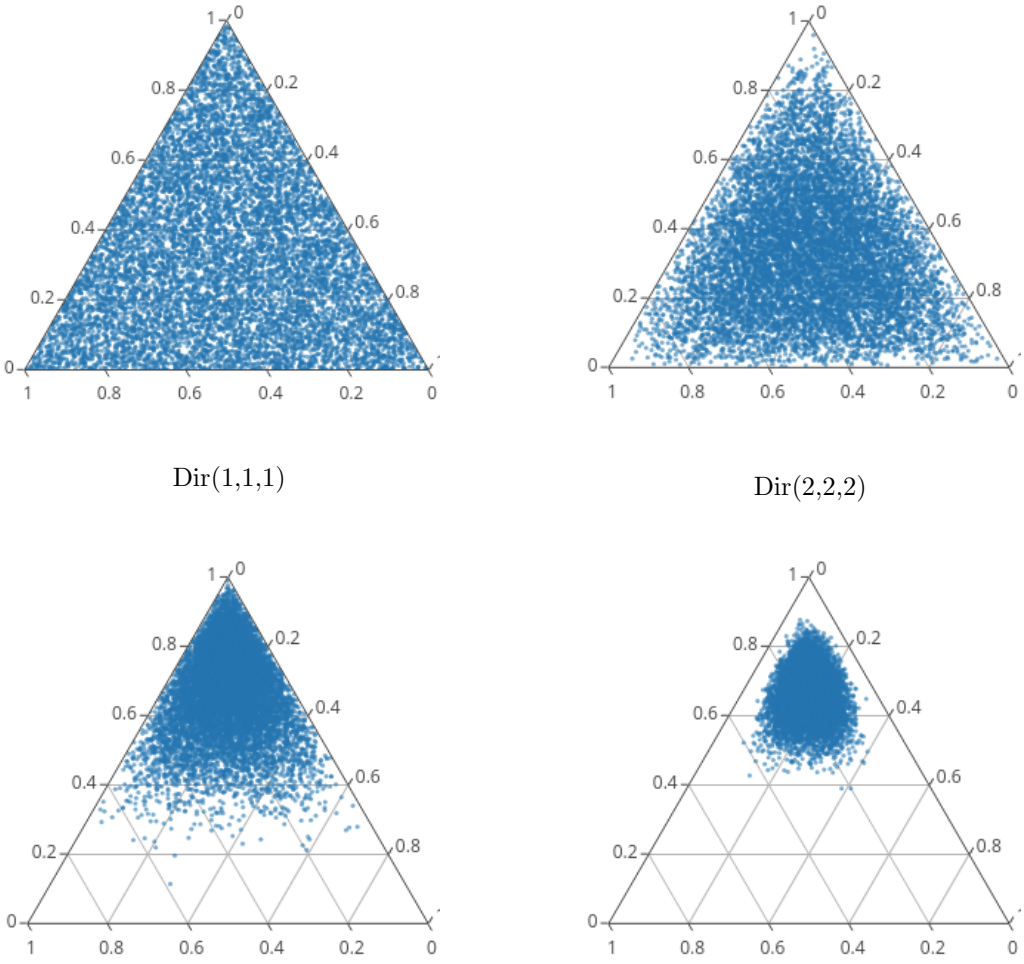
model fitting. If this imposed ordering does not reflect the true structure of the data, important interactions between categories may be missed, thereby reducing predictive accuracy.

For instance, consider Figure 5.1, where a nominal covariate with categories $\{1, 2, 3\}$ is to be partitioned using methods such as decision trees or Voronoi tessellations, techniques that underpin many competitive "black-box" algorithms. Under a stationary encoding, possible partitions include: all three categories in the same cell; $\{1, 2\}$ in one cell with 3 in another; $\{2, 3\}$ in one cell with 1 in another; or each category in its own cell. However, it is not possible to obtain a partition in which $\{1, 3\}$ form one cell and 2 forms another, even if such a grouping reflects an important relationship in the data. As the number of categories in a nominal covariate increases, the number of potentially relevant groupings grows combinatorially, making the limitations of stationary encoding increasingly restrictive.

It is important to note that permutations for binary covariates with only two categories are not considered because the categories are always adjacent and the order does not affect their relationship.

Determining this order can be difficult; for a covariate with K classes, there are $K!/2$ distinct orderings for the encoding of a nominal covariate if reversible orderings are counted as a single ordering, as they both have the same effect in most algorithms. For nominal covariates with a very small number of categories, there are few different orderings; for example, a nominal covariate with 3 categories has 3 possible encodings. However, as the number of categories increases, the number of potential orderings increases rapidly: with just 5 categories, there are 60 distinct orderings, and for 7 categories, this number increases to 2520.

Our approach focuses on deciding which covariates should be adjacent. This significantly reduces our search space, as there are only $\sum_{i=1}^{K-1} i$ distinct pairs of categories for nominal covariates; for example, there are only 21 pairs for a 7-categorical variable.



Dir(1,1,1) Dir(2,2,2)

Dir(8,2,2) Dir(32,8,8)

Figure 5.2: Dirichlet distribution with varying components.

We apply a Dirichlet prior to the probabilities of pairs being grouped together in an encoding scheme that is updated to reflect the pairings that frequently appear in models across the ensemble. The Dirichlet distribution is a multivariate generalisation of the Beta distribution and is commonly used to model probability vectors $(p_1, \ldots, p_K)$ that lie on the $(K-1)$-dimensional simplex, satisfying $p_k \geq 0$ and $\sum_{k=1}^{K} p_k = 1$. It is parametrised by a vector of concentration parameters

$\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$, with the prior density given by

$$\pi(p_1, \ldots, p_K) \propto \prod_{k=1}^{K} p_k^{\alpha_k - 1}.$$

The magnitude and relative sizes of the α_k control the spread of probability mass: when all $\alpha_k > 1$, the distribution favours more uniform allocations, whereas when $\alpha_k < 1$, the distribution encourages sparsity by concentrating mass near the vertices of the simplex. This latter property is particularly relevant in our setting, as small concentration parameters naturally favour a small subset of pairings having high probability, while the remainder receive negligible weight.

An illustration of Dirichlet distributions with varying component values is provided in Figure 5.2. The sparsity-inducing behaviour for small concentration parameters has been widely exploited in the literature, for example, in constructing priors achieving minimax posterior convergence rates in Bayesian convex aggregation Yang and Dunson (2014), in optimal shrinkage priors for variable selection in linear models Bhattacharya et al. (2015), and in anisotropic function estimation Bhattacharya et al. (2014). In our framework, this property allows the prior to naturally adapt towards pairings that are consistently favoured by the ensemble, while downweighting rarely occurring combinations without requiring hard thresholding.

5.2 Permutation encoding

To initialise a nominal covariate for permutation encoding, each category is randomly assigned a distinct integer in the range $1, \ldots, K$, where K denotes the total number of categories. This assignment serves as a temporary numerical representation of the unordered categorical levels.

We then construct an undirected combinatorial graph over the K categories, where each node represents a category and is labelled with its assigned integer. A complete graph is one in which every pair of nodes (categories in our case) is connected by an edge. The total number of edges in this complete graph is given by

$$N = \sum_{i=1}^{K-1} i = \frac{K(K-1)}{2},$$

which is the binomial coefficient $\binom{K}{2}$. For example, if the nominal covariate has $K = 5$ categories, the complete graph contains $\binom{5}{2} = 10$ edges, corresponding to all possible pairwise connections

between categories.

Consider a nominal covariate consisting of colours {blue, red, green, yellow, orange} that are encoded as {1,2,3,4,5}, respectively. The complete graph is illustrated in Figure 5.3.

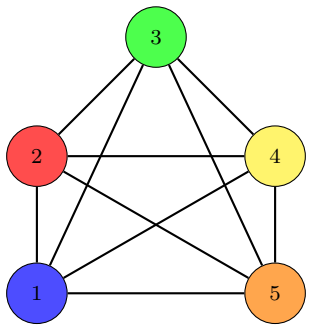


Figure 5.3: A complete graph for encoding a nominal covariate with 5 categories.

The edges are ordered using the lexicographic ordering, the unordered pairs (i, j) where $i < j$, arranged in the sequence

$$(1, 2), (1, 3), \ldots, (1, K), (2, 3), \ldots, (2, N), \ldots, (K - 1, K),$$

are listed in lexicographic order based on the first element, and then the second. The position in the order is the edge's index. For example, the edge index between the red and yellow nodes or 2^{nd} and 4^{th} nodes is 6.

Each edge in the graph is associated with a probability weight $\boldsymbol{w} = (w_1, ..., w_N)$ that governs how likely the corresponding pair of categories is to be placed adjacent to each other in an encoded ordering. These weights are adapted during inference to reflect the usefulness of certain orderings in improving model fit.

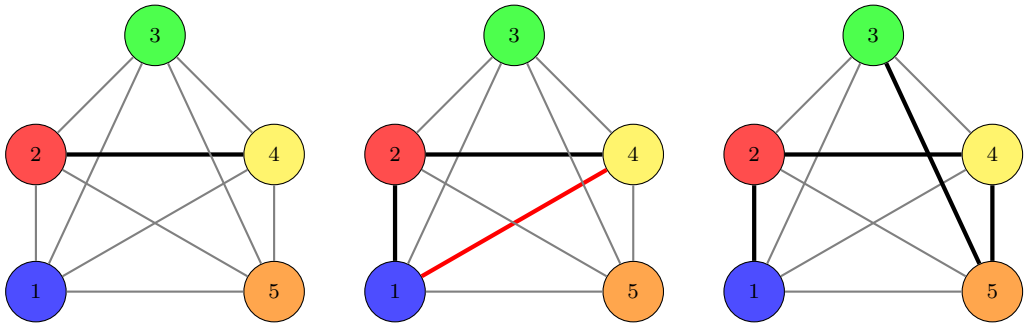
The goal of permutation encoding is to vary the ordering of categories each time the nominal covariate is sampled for inclusion in a model. To achieve this, we sample a constrained spanning tree from the complete graph. The constraint is that no node in the spanning tree may have more than two edges connected to it. This ensures that the resulting structure is a path (that is, a linear ordering) over the categories.

To construct such a spanning tree, we proceed iteratively. At each step, we sample an edge

from a Dirichlet distribution defined over the N possible edges

$$e \sim \text{Dir}(w_1, \ldots, w_N) = \text{Dir}\left(\frac{\alpha}{P} + m_1 \cdot \tau, \ldots, \frac{\alpha}{P} + m_{K-1} \cdot \tau\right),$$

where $\alpha > 0$ and $\tau > 0$ are hyperparameters, and $\boldsymbol{M} = (m_1, \ldots, m_N)$ is the number of times the edge is included in other models generated at each iteration of the sampler. Initially, all $m_i = 0$, and the Dirichlet distribution is uniform. As the MCMC proceeds, edges that appear more frequently in accepted models receive higher probabilities and are thus more likely to be selected again.



a) Sampling an edge. b) Rejected edge. c) Sampled spanning tree.

Figure 5.4: Illustration of sampling the spanning tree.

Once an edge has been sampled, the edge weight for the sampled edge is temporarily assigned 0 until a spanning tree for this inclusion of the nominal covariate is created. For example, returning to the previous example, as illustrated in Figure 5.4a, if the edge between the red and yellow nodes or 2^{nd} and 4^{th} is sampled, then w_6 is set to zero, so that the edge can not be sampled again.

Then, we check that the sampled edge does not create a cycle. If it does, we discard it and sample another edge; if this condition is satisfied, we add the edge to an "inclusion set". By adding this edge, if one of the nodes now has two edges, then all the edge weights are temporarily set to 0 so that the condition that no node has more than two connected edges is satisfied. The process is repeated until the inclusion set contains exactly $K - 1$ edges, which guarantees that the structure is a path connecting all K nodes.

For instance, if the red edge is sampled in Figure 5.4b, then it is rejected since a cycle {1-2-4} is created, and the edge weight for this edge is set to zero, $w_3 = 0$. Also note that since node 2 already has two edges, all edges connected to it will have a weight of 0.

The resulting path defines a permutation of the categories: the two terminal nodes (that is,

those of degree one) are assigned positions 1 and K, and the remaining nodes are ordered according to their positions along the path. This ordering is then used as the model's encoding. Figure 5.4c gives an example of this, and encoding would now be {blue, red, yellow, orange, green}={1,2,4,5,3}, or the reverse of this, but in most algorithms, the direction of the order does not influence the algorithm.

It is important to emphasise that while the encoding varies across individual models, the original labelling of the complete graph remains fixed. This ensures that the probability weights always refer to the same pairs of categories, allowing them to be updated meaningfully across iterations.

If a model is accepted in the MH algorithm, the values of $\boldsymbol{M}$ corresponding to the edges in the inclusion set are incremented by 1, effectively increasing the associated Dirichlet parameters by τ in future proposals. Conversely, if a nominal covariate is removed from a model, the corresponding entries in $\boldsymbol{M}$ are decremented by 1. This dynamic adjustment encourages the algorithm to reinforce orderings that have been useful in the past while still allowing exploration of new permutations.

Over time, this adaptive weighting increases the probability of sampling edges that contribute to effective orderings, thereby supporting convergence to a suitable or optimal encoding of the nominal covariate. Intuitively, edges that consistently improve model fit will become more dominant in the Dirichlet distribution, leading to greater prevalence in future encodings.

The hyperparameters α and τ are central to controlling the levels of exploration and adaptation. A larger value of α results in more uniform initial weights and makes the effect of each τ update relatively smaller, thus encouraging exploration. In contrast, increasing τ amplifies the reinforcement of successful edges, promoting faster convergence but possibly reducing diversity in explored orderings.

The hyperparameters (α, τ) can be selected via cross-validation, or treated as unknowns and assigned hyper-priors in a fully Bayesian approach. However, such strategies can be computationally demanding. In practice, we find that default values such as $\alpha = 10$ and $\tau = 0.1$ perform well, especially if paired with a slightly extended burn-in period. It is often sufficient to conduct a brief sensitivity analysis to assess whether other choices yield meaningful improvements in predictive accuracy or convergence speed.

Algorithm 3 gives the steps of sampling a new permutation of categories when a nominal covariate is selected and shows that if a proposed tessellation then the edge weights are updated.

Algorithm 3: Sampling an encoding.

```
Data: Training data (X_train, y_train), hyperparameters (α, τ)
for i = 1 to (K − 1) do
    Sample edge
    if no cycles created then
        Add edge to inclusion set
        if Either of the nodes has more than two edges then
            Set all edge weights to 0 for nodes with more than two edges
    else
        Set weight of edge to 0
        return to 2
Accept or reject proposed tessellation;
if Accepted then
    Update edge weights
```

5.3 Implementation in reversible jump algorithms

We consider a general Bayesian ensemble framework in which the model is composed of multiple components, denoted by parameters θ. For instance, in an additive ensemble, θ may represent a set of functions $\{f_1, \ldots, f_M\}$, where the overall predictive model takes the form

$$f(x) = \sum_{m=1}^{M} f_m(x).$$

The data likelihood given the ensemble configuration θ is denoted by $p(y \mid \theta)$, and the prior over the full set of ensemble components is $p(\theta)$. In the Metropolis-Hastings (MH) algorithm, a proposal $\theta' \sim q(\theta' \mid \theta)$ is generated, and the acceptance probability is given by the standard MH rule

$$\alpha(\theta, \theta') = \min\left(1,\ \frac{p(y \mid \theta')\, p(\theta')\, q(\theta \mid \theta')}{p(y \mid \theta)\, p(\theta)\, q(\theta' \mid \theta)}\right).$$

In practical ensemble implementations, it is common to update one component f_m at a time while conditioning on the remaining components $\{f_j\}_{j \neq m}$. Letting θ represent the entire ensemble and θ' represent an updated ensemble differing only in f_m, the acceptance probability becomes

$$\alpha = \min\left(1,\ \frac{p\left(y \mid f_1, \ldots, f'_m, \ldots, f_M\right)\ p(f'_m)\, q(f_m \mid f'_m)}{p\left(y \mid f_1, \ldots, f_m, \ldots, f_M\right)\ p(f_m)\, q(f'_m \mid f_m)}\right).$$

Assuming additive Gaussian noise, the model can be written as

$$y = \sum_{m=1}^{M} f_m(\boldsymbol{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2),$$

so that the likelihood becomes a multivariate normal. When only f_m is modified, it is computationally efficient to define the residual excluding f_m as

$$r^{(-m)} = y - \sum_{j \neq m} f_j(\boldsymbol{x}),$$

leading to the conditional likelihood of f_m being

$$p(y \mid \theta) = \mathcal{N}(r^{(-m)} \mid f_m(\boldsymbol{x}), \sigma^2).$$

This residual-based formulation is particularly important in MCMC schemes such as Bayesian backfitting and underpins the efficiency of ensemble samplers such as BART and additive Gaussian processes. By isolating f_m in this way, updates are made locally while preserving global model structure and ensuring the detailed balance condition of MH.

Our proposed permutation encoding for nominal covariates integrates seamlessly with this framework. Because the encoding affects only the representation of nominal inputs, not the underlying distributional assumptions or model parameters, its influence on the MH acceptance probability is nullified.

Specifically, when a nominal covariate is encoded via a different permutation, both the proposal and the target ensemble incorporate the same class-dependent encoding transformation. If a nominal covariate with K levels is included or removed, the transition probability associated with the tessellation structure is scaled by $1/K$ in both the forward and reverse directions. These terms cancel in the MH ratio, and thus the acceptance probability remains unchanged

$$\frac{1/K}{1/K} = 1.$$

Hence, the permutation encoding mechanism does not interfere with the detailed balance of the MH algorithm. The model's likelihood and prior remain correctly specified under any categorical

ordering, and the proposal symmetry is preserved by design. This enables adaptive learning of good permutations without altering the theoretical foundations of the MCMC algorithm.

Over time, the Dirichlet-based weighting mechanism described previously encourages the sampler to favour orderings that contribute to more accurate models, without requiring any change to the MH formulation.

This property is essential for incorporating nominal covariates into Bayesian ensembles in a principled way. It preserves the theoretical correctness of the sampling algorithm while introducing a flexible and data-driven mechanism for learning category orderings in the presence of nominal inputs.

5.4 Benchmark data sets

We compared the predictive performance of AddiVortes against several competing algorithms on benchmark datasets obtained from Deshpande (2025) and Zhu et al. (2024). These datasets span a wide range of applications and sample sizes, with the number of observations ranging from 202 (Ais) to 22,272 (Spouse) with between 3 and 60 covariates.

For each dataset, we created 20 independent train/test splits, with 3/4 of the data allocated to the training set. Three encoding techniques are used in our analysis for AddiVortes: permutation encoding (PE), one-hot encoding (OH) and target encoding (TE) based on the target mean. We also considered BART with one-hot encoding and FlexBART from Deshpande (2025).

To maintain computational efficiency, the default hyperparameter configurations were employed for all algorithms throughout the analysis. The nominal AddiVortes algorithm was initialised with the standard default values of $(m, \nu, q, k, \sigma_c, \omega, \lambda_c) = (200, 6, 0.85, 3, 0.8, 3, 25)$ alongside the settings $(\alpha, \tau) = (10, 0.1)$, which have consistently demonstrated strong performance across various simulations. While cross-validation could have been utilised to fine-tune these hyperparameters for specific datasets, the selected defaults provided a robust and practical balance between predictive accuracy and total processing time.

We use the Standardised Mean Squared Error (SMSE) to compare methods, which evaluates predictive performance by scaling the Mean Squared Error (MSE) relative to the sample variance

of the response variable:

$$\text{SMSE} = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}.$$

This metric facilitates performance comparisons across different datasets, where y_i represents the observations, $\hat{y}_i$ the model predictions, and $\bar{y}$ the sample mean. A value of one indicates that the model provides no improvement over a simple baseline prediction of the mean, whereas a value of zero denotes a perfect fit.

To precisely define how prediction error is calculated, we evaluate the models using the Standardised Mean Squared Error (SMSE) based on the posterior predictive mean. Specifically, we do not fix the parameter values at a single point estimate, such as a posterior mode or mean of the components. Instead, for a new test observation $\boldsymbol{x}^*$, the predicted value $\hat{y}^*$ is obtained by averaging the output of the sum-of-tessellations model over the S post-burn-in MCMC samples, given by

$$\hat{y}^* = \frac{1}{S}\sum_{s=1}^{S} G(\boldsymbol{x}^*; \boldsymbol{T}^{(s)}, \boldsymbol{M}^{(s)}).$$

The SMSE is then computed by taking the mean squared error of these averaged predictions against the true observed values in the test set, and standardising it by dividing by the variance of the observed responses. This standardisation yields a scale-free metric, allowing for fair and interpretable comparisons across different datasets regardless of the scale of the target variable.

Dataset	$\boldsymbol{n}$	**AV-PE**	**AV-OH**	**AV-TE**	**BART-OH**	**FlexBART**
Abalone	4,177	43.0	**42.8**	43.0	44.4	44.4
Ais	202	**11.0**	11.2	11.4	11.8	11.9
Alcohol	2,467	96.8	97.1	97.1	97.1	**96.6**
Amenity	3,044	**29.7**	29.8	29.8	29.7	29.8
Cpu	209	**29.4**	31.3	31.0	30.1	30.1
Insur	2,182	**2.1**	2.4	2.3	3.1	**2.1**
Mpg	392	**10.9**	11.0	11.1	11.7	11.6
Servo	167	**15.7**	17.4	16.4	17.7	16.1
Spouse	22,272	**17.0**	17.1	17.2	17.1	17.1
Strike	625	**30.0**	31.5	30.4	32.3	**30.0**
Cholesterol	303	**31.0**	35.7	40.2	89.7	35.9

Table 5.1: A table comparing the SMSE for competing methods on several benchmark datasets. AV denotes the use of the AddiVortes algorithm; where PE is using permutation encoding; OH denotes the use of one-hot encoding and TE indicates the use of mean target encoding.

Table 5.1 details the SMSE values calculated for each method across the benchmark datasets. The comprehensive results underscore the robust predictive capability of the AddiVortes model

when paired with permutation encoding. Specifically, the AV-PE approach achieved the lowest error rate in nine of the eleven datasets analysed, either securing the top position outright or tying with the best performing alternative. This trend is consistent across varying sample sizes, from the smaller *Ais* dataset to the much larger *Spouse* dataset, suggesting that the permutation encoding strategy successfully adapts to different data densities and structures.

A closer examination of the internal comparisons between encoding strategies reveals significant disparities in performance. In datasets characterised by higher complexity, such as *Cpu* and *Servo*, the permutation encoding method provided a substantial improvement over the standard one-hot encoding approach. For instance, in the *Cpu* dataset, AV-PE achieved an SMSE of 29.4 compared to 31.3 for AV-OH. Similarly, in the *Servo* dataset, AV-PE recorded an error of 15.7 against 17.4 for AV-OH. This performance gap suggests that statically expanding categorical variables into binary columns may dilute the signal in settings with limited data or high cardinality, whereas finding an optimal ordering allows the model to partition the covariate space more efficiently.

The comparison with target encoding further highlights the stability of the permutation approach. While target encoding is a popular strategy for handling high-cardinality features, it appears to struggle in certain regression contexts presented here. The *Cholesterol* dataset provides the clearest evidence of this, where AV-TE resulted in a notably higher error of 40.2 compared to 31.0 for AV-PE. This indicates that replacing categories with their unconditional mean may result in a loss of information required to capture conditional dependencies, a limitation that the dynamic partition-based nature of permutation encoding is able to overcome. For the *Cholesterol* dataset, the BART-OH method performed particularly poorly. We attribute this to the underlying signal being diluted by the increased dimensionality. In contrast, AddiVortes benefits from a more natural variable selection mechanism, which allows the model to isolate and capture the true underlying signal much more effectively, even when one hot encoding is applied.

When benchmarked against external algorithms, AV-PE consistently outperformed standard BART with one-hot encoding and remained highly competitive with the state-of-the-art FlexBART model. The standard BART implementation struggled severely with the *Cholesterol* dataset, returning an SMSE of 89.7, which is nearly triple that of the AV-PE result. This suggests that standard BART trees may have difficulty selecting relevant splits when the feature space is artificially inflated by one-hot encoding. Against FlexBART, AV-PE performed comparably, achieving

identical results in the *Insur* and *Strike* datasets and outperforming it in *Ais* and *Servo*. Although FlexBART proved marginally superior in the *Alcohol* dataset, the overall results show permutation-encoded AddiVortes as a leading method for regression with nominal inputs.

5.5 Discussion

The integration of nominal covariates into geometric partitioning methods presents a unique theoretical challenge, as Voronoi tessellations rely fundamentally on the existence of a metric space which is absent in unordered categorical data. The permutation encoding framework introduced in this chapter successfully resolves this incompatibility by transforming the discrete nominal space into a latent ordinal structure that is dynamically learned during the inference process. Unlike stationary techniques such as one-hot or target encoding, which fix the representation of categories prior to model fitting, this approach treats the proximity of categories as an unknown parameter. By allowing the model to explore the space of possible permutations, AddiVortes gains the flexibility to group arbitrary subsets of categories within a single Voronoi cell, thereby recovering complex structural relationships that static encodings often obscure.

The empirical results from the benchmark datasets provide robust evidence for the efficacy of this method. The consistent outperformance of permutation encoding (AV-PE) compared to one-hot (AV-OH) and target encoding (AV-TE) indicates that the rigid structures imposed by standard techniques can be detrimental to predictive performance. Specifically, the stark contrast in performance on datasets such as Cholesterol and Servo highlights the limitations of one-hot encoding, which increases dimensionality and sparsity, making it difficult for the sampler to identify meaningful partitions. Conversely, while target encoding avoids dimensionality issues, it risks discarding valuable conditional information by compressing categories into a single scalar summary. Permutation encoding strikes an effective balance by maintaining the distinct identity of categories while enabling the model to learn an order for nominal covariates that improves predictive performance.

Furthermore, the comparison with FlexBART is particularly illuminating regarding the nature of categorical data modelling. FlexBART represents the current state-of-the-art in this domain by explicitly modelling the connectivity of categories using graph-based priors. The fact that AddiVortes with permutation encoding performs competitively with, and frequently outperforms,

FlexBART suggests that inducing a linear ordering via a constrained spanning tree is a sufficient and highly effective strategy for capturing categorical structure. The adaptive Dirichlet prior on the graph edges plays a critical role in this success; by accumulating evidence on which categories are frequently grouped together, the algorithm effectively prunes the vast combinatorial space of permutations, guiding the sampler towards optimal encodings without requiring manual intervention.

To further validate the effectiveness of the permutation encoding method, comprehensive empirical evaluations encompassing a broader spectrum of synthetic scenarios and diverse real-world datasets are required. While initial results are promising, a more exhaustive sensitivity analysis is necessary to ascertain the stability of the encoding across varying levels of covariate cardinality and degrees of sparsity.

The theoretical framework suggests that this technique possesses broader applicability and could be seamlessly integrated into other Bayesian ensemble frameworks, such as BART.

Moreover, it is important to acknowledge the computational trade-offs inherent in this approach. While the reversible jump MCMC formulation elegantly ensures that the acceptance probabilities remain tractable, the process of sampling a spanning tree and deriving a permutation for every nominal covariate in every iteration introduces computational overhead. For variables with moderate cardinality, this cost is negligible compared to the gains in accuracy. However, for categorical variables with extremely high cardinality, the quadratic growth of the underlying complete graph could pose scalability challenges.

Future research could investigate hierarchical or approximate graph sampling methods to extend this framework to massive categorical domains. Nevertheless, the results presented here establish permutation encoding as a powerful, general-purpose extension to the AddiVortes framework, ensuring its applicability extends beyond continuous regression to the full spectrum of mixed-type data problems.

Chapter 6

Binary and multinomial classification

Binary classification, a fundamental task in supervised learning, plays a key role in extracting meaningful insights from complex datasets. Accurate models are crucial for applications such as medical diagnosis (see, for example, Allison, 2024), fraud detection (see, for example, Leevy et al., 2023), and predicting equipment failure (see, for example, Ferreira et al., 2021).

While binary classification is important, many real-world problems naturally involve more than two outcomes. Ordinal classification arises when outcomes are categorical but possess an inherent order, as in credit ratings, disease severity, or customer satisfaction scores. Multinomial (or nominal) classification, by contrast, is appropriate when outcomes are categorical without any natural ordering, such as predicting a patient's disease type.

In this chapter, we extend the AddiVortes method with a probit-based modelling framework that accommodates binary and multinomial outcomes. The probit model is particularly appealing in this context due to its interpretability and established statistical properties (Albert and Chib, 1993). For binary data, the latent-variable probit model provides a natural way to link continuous underlying processes to observed discrete responses. For multinomial outcomes, the multinomial probit model generalises this construction by considering multiple latent variables, with the observed class determined by the maximum latent variable.

To assess performance, we evaluate AddiVortes across a range of real benchmark datasets. These datasets span diverse application areas and outcome structures, allowing us to demonstrate the adaptability and robustness of AddiVortes under different classification challenges.

Alongside real-world data, we conduct carefully designed simulation studies to examine the

behaviour of AddiVortes under controlled conditions. Simulation experiments allow us to assess predictive performance when the true data-generating process is known, providing insights into model calibration, uncertainty quantification, and scalability. In particular, we explore settings with varying class separations, noise levels, and dimensionality, ensuring that AddiVortes is tested across a variety of difficulty levels. By combining benchmark datasets with simulation studies, we obtain a comprehensive understanding of how the method performs across practical applications and theoretical cases.

The developments of the algorithm have significant implications for real-world applications. We illustrate the value of AddiVortes for binary classification with a study on loan approval prediction using data from the Home Mortgage Disclosure Act (HMDA). Established in 1975 in the United States, HMDA promotes transparency in mortgage lending by requiring financial institutions to report loan-level information. The dataset includes loan type, property value, applicant demographics, and application outcomes, and is widely used in research on lending practices and potential discrimination (Popick, 2022; Lewis-Faupel and Tenev, 2024; Cyree and Winters, 2023). Its breadth makes it valuable for predictive modelling, where accurate approval predictions benefit both financial institutions by improving decision-making and efficiency and borrowers by clarifying factors influencing outcomes. On a broader scale, predictive insights inform policies promoting fair lending and equitable access to credit.

The rest of the chapter is organised as follows. Sections 6.1 and 6.2 detail the probit model setup for AddiVortes, including the mathematical formulation and implementation details. Section 6.3 highlights the use of Voronoi tessellations in classification through simulation experiments. Section 6.4 demonstrates AddiVortes' potential for binary classification using real-world data, comparing its performance with other methods to highlight its strengths and weaknesses. Section 6.5 presents a more in-depth analysis of predicting loan outcomes using HMDA data. Finally, Section 6.6 concludes with a discussion on findings.

6.1 Binary classification

In this section, we extend AddiVortes from modelling continuous responses to handling binary classification problems, where $Y \in \{0, 1\}$, using a probit formulation, similar to the approach

used to modify BART in Zhang and Härdle (2010) as described in Section 2.3.1. The goal is to estimate the probability that an observation belongs to the target class $Y = 1$, given covariates $\boldsymbol{x}$. Specifically, the model estimates a function $p : \mathbb{R}^p \to [0, 1]$ defined as

$$p(\boldsymbol{x}) \equiv P[Y = 1 \mid \boldsymbol{x}] = \Phi\big(G(\boldsymbol{x})\big), \tag{6.1}$$

where

$$G(\boldsymbol{x}) = \sum_{j=1}^{m} g(\boldsymbol{x}; T_j, M_j),$$

and $\Phi(\cdot)$ denotes the standard normal cumulative distribution function. The classification probability $p(\boldsymbol{x})$ is thus derived from the sum-of-tessellations structure of AddiVortes, offering a coherent and interpretable probabilistic framework distinct from aggregate ensemble classifiers that average over hard decision rules.

To extend AddiVortes using Equation (6.1), we place a regularisation prior on the latent function $G(\boldsymbol{x})$ and implement the Bayesian backfitting algorithm for posterior computation, analogous to that used in the original AddiVortes model. We assume prior independence similar to Equations (4.1) and (4.2), but the probit model implicitly assumes a fixed variance $\sigma = 1$ to solve identifiability issues. As a result, the prior is specified only over the tessellation components $(T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m)$.

In the probit model, this identifiability problem occurs with two "unknowns": the value of the latent function, $G(\mathbf{x})$, and the standard deviation of the unobserved noise, σ. The model's goal is to find the probability of $Y = 1$, which it calculates using the equation $P(Y = 1 \mid \mathbf{x}) = \Phi(G(\mathbf{x})/\sigma)$. This means the final probability depends only on the ratio of $G(\mathbf{x})$ to σ, not on their individual values. If the model determines that this ratio must be 2 to fit the data, it cannot tell the difference between scenarios like $G(\mathbf{x}) = 2$ and $\sigma = 1$, or $G(\mathbf{x}) = 4$ and $\sigma = 2$, or $G(\mathbf{x}) = 20$ and $\sigma = 10$. All these pairs produce the exact same ratio and therefore the exact same probability, leaving the model with no basis to prefer one solution over the others.

To make the model identifiable and thus solvable, we must force it to have only one unknown. We achieve this by fixing the scale of one of the parameters. Similar to Carriero et al. (2015), we simply set the noise standard deviation σ to 1. By setting $\sigma = 1$, the model's equation simplifies to

$P = \Phi(G(\mathbf{x}))$. Now, if the model needs the ratio to be 2, there is only one possible solution: $G(\mathbf{x})$ must be 2. The ambiguity is gone; the model can find a unique estimate for the function $G(\mathbf{x})$, which is why it implicitly assumes $\sigma = 1$.

For the conditional prior distribution $p(\mu_{ij} \mid T_j)$, we use the same conjugate normal form as AddiVortes,

$$\mu_{ij} \sim \mathcal{N}(0, \sigma_\mu^2), \tag{6.2}$$

but with $\sigma_\mu = \frac{3.0}{k\sqrt{m}}$, so that a shrinkage factor k can be chosen to ensure that the sum-of-tessellations function $G(\boldsymbol{x})$ lies with high probability within the interval $(-3.0, 3.0)$. This encourages posterior values of $\Phi(G(\boldsymbol{x}))$ to fall meaningfully within $[0, 1]$, while avoiding saturation at the boundaries. The prior, therefore, shrinks the tessellation output values μ_{ij} towards zero, constraining the influence of individual tessellations.

By shrinking $G(\boldsymbol{x})$ toward zero, the prior in Equation (6.2) has the effect of shrinking the classification probability $p(\boldsymbol{x}) = \Phi[G(\boldsymbol{x})]$ toward 0.5. If shrinkage toward a different baseline probability $p_0 \in (0, 1)$ is desired, this can be achieved by redefining the latent function as $G_c(\boldsymbol{x}) = G(\boldsymbol{x}) + c$ in Equation (6.1), where the offset $c = \Phi^{-1}(p_0)$.

Additionally, to concentrate $p(\boldsymbol{x})$ within an interval other than $(\Phi(-3.0), \Phi(3.0))$, the prior in Equation (6.2) can be modified accordingly. In particular, adjusting the scale parameter σ_μ allows direct control over the expected range of $G(\boldsymbol{x})$, and hence the spread of the resulting classification probabilities.

For posterior computation, the MCMC backfitting algorithm described in Hastie and Tibshirani (2000) can be adapted to the classification setting via a data augmentation approach, as proposed by Albert and Chib (1993). This involves introducing a set of latent variables $\boldsymbol{z} = \{z_1, \ldots, z_n\}$, which govern the observed binary responses Y_i through the rule

$$Y_i = \begin{cases} 1 & \text{if } z_i > 0, \\ 0 & \text{if } z_i \leq 0. \end{cases}$$

The latent variables $\boldsymbol{z}$ are linked to the predictors through an additive structure, where each additive term corresponds to a tessellation component based on the covariates. Specifically, for a

given observation $\boldsymbol{x}_i$, the model takes the form

$$z_i = G(\boldsymbol{x_i}) + \epsilon_i,$$

where $\epsilon_i \sim \mathcal{N}(0,1)$, consistent with the probit specification. This formulation enables efficient posterior sampling of both the latent variables and the tessellation components via standard Gibbs steps.

A Gibbs sampler is then used to generate samples from the joint posterior distribution

$$\pi\left((T_1, \boldsymbol{M}_1), (T_2, \boldsymbol{M}_2), \ldots, (T_m, \boldsymbol{M}_m), \boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{y}\right).$$

Each iteration of the Gibbs sampler consists of two main steps. First, we draw n values z_i for $i = 1, \ldots, n$ from the full conditional distribution

$$\pi\left(\boldsymbol{z} \mid (T_1, \boldsymbol{M}_1), (T_2, \boldsymbol{M}_2), \ldots, (T_m, \boldsymbol{M}_m), \boldsymbol{x}, \boldsymbol{y}\right).$$

Second, we sequentially update each tessellation component $(T_j, \boldsymbol{M}_j)$ for $j = 1, \ldots, m$ by sampling from the conditional distribution

$$\pi\left((T_j, \boldsymbol{M}_j) \mid \{T_{j'}, \boldsymbol{M}_{j'}\}_{j' \neq j}, \boldsymbol{z}, \boldsymbol{x}, \boldsymbol{y}\right),$$

using the standard AddiVortes MCMC update step.

Let $\hat{z}_i = G(\boldsymbol{x}_i)$ denote the current fitted value for observation i based on the sum of all tessellations. Then, each z_i can be sampled independently from a truncated normal distribution

$$z_i \mid y_i, (T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m) \sim \begin{cases} \mathcal{TN}_{(0,\infty)}(\hat{z}_i, 1) & \text{if } y_i = 1 \\ \mathcal{TN}_{(-\infty,0)}(\hat{z}_i, 1) & \text{if } y_i = 0 \end{cases}$$

This augmentation facilitates efficient inference in the probit AddiVortes model by decoupling the binary likelihood into conditionally Gaussian updates.

After discarding a suitable burn-in period to ensure convergence, the subsequent K posterior draws are used for inference. Let these draws be denoted as $\{(T_1^{(k)}, M_1^{(k)}), \ldots, (T_m^{(k)}, \boldsymbol{M}_m^{(k)})\}_{k=1}^{K}$. For

the k^{th} posterior sample, the estimated probability that an observation with covariates $\boldsymbol{x}$ belongs to class 1 is given by

$$\Phi\left\{\sum_{j=1}^{m} g(\boldsymbol{x}; T_j^{(k)}, \boldsymbol{M}_j^{(k)})\right\}.$$

Averaging over all posterior samples, the posterior mean estimate of the classification probability is

$$\frac{1}{K}\sum_{k=1}^{K}\Phi\left\{\sum_{j=1}^{m} g(\boldsymbol{x}; T_j^{(k)}, \boldsymbol{M}_j^{(k)})\right\}. \tag{6.3}$$

We use the probability estimate from Equation (6.3) for classification. An observation is classified as belonging to class 1 if the posterior mean probability exceeds 0.5 (or an alternative threshold, if specified); otherwise, it is classified as class 0.

6.2 AddiVortes for multinomial classification

We extend the AddiVortes model to the multinomial classification setting in a similar way to Kindo et al. (2016) as described in Section 2.2, where the response variable $Y \in \{1, 2, \ldots, L\}$ indicates membership in one of L unordered categories. To model this, we adopt the multinomial probit framework, which introduces a vector of latent variables and defines the observed outcome based on which latent variable is largest.

That is, for each observation $i \in \{1, \ldots, n\}$, we introduce a latent vector $\boldsymbol{z}_i = (z_{i1}, \ldots, z_{i(L-1)})^\top$. The observed class label Y_i is determined by

$$Y_i = \begin{cases} l, & \text{if } z_{il} = \max\left\{z_{i1}, \ldots, z_{i(L-1)}\right\} \geq 0, \\ L, & \text{if } \max\left\{z_{i1}, \ldots, z_{i(L-1)}\right\} < 0. \end{cases}$$

Category L serves as the baseline or reference class, ensuring identifiability by fixing the baseline class latent utility to zero. This constraint is necessary because the choice of class only depends on the differences between the latent variables, not their absolute values; without a fixed reference, adding any constant C to all latent variables would produce identical probabilities, making a unique set of parameters impossible to identify.

Each latent variable vector $\boldsymbol{z}_i$ is modelled using a sum-of-tessellations structure,

$$\boldsymbol{z}_i = \boldsymbol{G}(\boldsymbol{x}_i; \boldsymbol{T}, \boldsymbol{M}) + \boldsymbol{\epsilon}_i, \quad \boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}),$$

where $\boldsymbol{G}(\boldsymbol{x}_i; \boldsymbol{T}, \boldsymbol{M}) = (G_1(\boldsymbol{x}_i; \boldsymbol{T}_1, \boldsymbol{M}_1), \ldots, G_{L-1}(\boldsymbol{x}_i; \boldsymbol{T}_{L-1}, \boldsymbol{M}_{L-1}))^\top$ is a vector of predicted values from the $L-1$ sum-of-tessellations, and $\boldsymbol{\epsilon}_i = (\epsilon_{i1}, \ldots, \epsilon_{i(L-1)})^\top \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is an $(L-1) \times (L-1)$ positive-definite covariance matrix capturing correlations between the latent variables. Each class $l = 1, \ldots, L-1$ is modelled with its own AddiVortes ensemble of m tessellation components.

Similar to the binary case, the model specification exhibits a well-documented identifiability issue. For example, the multiplication of both sides of the latent equation by a positive constant does not alter the implied choice outcome. To illustrate, suppose there are latent variables $\tilde{\boldsymbol{z}}$ such that

$$\tilde{\boldsymbol{z}} = \boldsymbol{G}(\boldsymbol{x}; \tilde{\boldsymbol{T}}, \tilde{\boldsymbol{M}}) + \tilde{\boldsymbol{\epsilon}}, \quad \tilde{\boldsymbol{\epsilon}} \sim \mathcal{N}(\boldsymbol{0}, \tilde{\boldsymbol{\Sigma}}),$$

where

$$\tilde{\boldsymbol{z}} = \bar{\alpha}\boldsymbol{z}, \qquad \boldsymbol{G}(\boldsymbol{x}; \tilde{\boldsymbol{T}}, \tilde{\boldsymbol{M}}) = \bar{\alpha}\boldsymbol{G}(\boldsymbol{x}; \boldsymbol{T}, \boldsymbol{M}), \qquad \tilde{\boldsymbol{\Sigma}} = \bar{\alpha}^2\boldsymbol{\Sigma}, \qquad \bar{\alpha} > 0.$$

By the decision rule, $\tilde{\boldsymbol{z}}$ and $\boldsymbol{z}$ yield the same categorical outcome Y. To circumvent this issue, Burgette and Nordheim (2012), among others, restrict the trace of $\boldsymbol{\Sigma}$ to equal $L-1$.

Following Imai and van Dyk (2005), the prior distribution of Σ is given by

$$\pi(\boldsymbol{\Sigma}) \;=\; \int p(\boldsymbol{\Sigma}, \alpha^2)\, d\alpha^2 \;\propto\; |\boldsymbol{\Sigma}|^{-(\nu+L)/2} \Big\{ \operatorname{tr}\big(\Psi\boldsymbol{\Sigma}^{-1}\big) \Big\}^{-\nu(L-1)/2},$$

with the restriction $\operatorname{tr}(\boldsymbol{\Sigma}) = L-1$. This prior is not set directly but is constructed using an auxiliary scaling parameter, α^2. The prior $\pi(\boldsymbol{\Sigma})$ is obtained using marginalisation, which means integrating out (or averaging over) all possible values of this temporary α^2 parameter. It is a constrained inverse-Wishart distribution induced by

$$\tilde{\boldsymbol{\Sigma}} \sim \mathcal{W}^{-1}\big(\nu,\ \Psi\big), \qquad \alpha^2 \mid \boldsymbol{\Sigma} \;\sim\; \frac{\operatorname{tr}\big(\Psi\boldsymbol{\Sigma}^{-1}\big)}{\chi^2_{\nu(L-1)}},$$

where ν and the positive-definite matrix $\Psi \in \mathbb{R}^{(L-1)\times(L-1)}$ are hyperparameters. Setting $\nu = L+1$ and Ψ to be an identity matrix is equivalent to sampling the corresponding correlations of $\boldsymbol{\Sigma}$ from

a uniform distribution. When $\nu > L$, the expectation of the unconstrained matrix has a closed form,

$$\mathbb{E}(\tilde{\boldsymbol{\Sigma}}) = \frac{\Psi}{\nu - L}.$$

Posterior sampling is carried out using a Gibbs sampler similar to the one used in AddiVortes, which alternates between updates of the latent variables $\boldsymbol{z}$, the tessellation structures and cell means (using a backfitting Markov chain Monte Carlo algorithm), and the covariance matrix $\boldsymbol{\Sigma}$, with identifiability maintained through an auxiliary scaling parameter α.

First, we sample the expansion parameter $(\alpha_1^{(k)})^2$ from its prior distribution,

$$(\alpha_1^{(k)})^2 \sim \text{tr}(\Psi(\boldsymbol{\Sigma}^{(k-1)})^{-1})/\chi^2_{\nu(L-1)},$$

where $\chi^2_{\nu(L-1)}$ denotes a chi-squared distribution with $\nu(L-1)$ degrees of freedom. This auxiliary parameter allows the algorithm to navigate the unconstrained parameter space while maintaining conjugacy for the covariance matrix update. Drawing a sample for $\boldsymbol{z}_i \mid Y_i, \boldsymbol{T}, \boldsymbol{M}, \boldsymbol{\Sigma}$ involves iterative random draws from a univariate truncated normal distribution,

$$z_{il} \mid Y_i, \boldsymbol{z}_{i(-l)}, \boldsymbol{T}, \boldsymbol{M}, \boldsymbol{\Sigma} \sim \begin{cases} \mathcal{TN}_{[\max(0,\max(\boldsymbol{z}_{i(-l)})),\infty)}(m_{il}, \tau_{il}) & \text{if } Y_i = l, \\ \mathcal{TN}_{(-\infty,\max(0,\max(\boldsymbol{z}_{i(-l)}))]}(m_{il}, \tau_{il}) & \text{if } Y_i \neq l, \end{cases}$$

that is, $\max\{0, \max(\boldsymbol{z}_{i(-l)})\}$ is the lower truncation point if $Y_i = l$, and acts as an upper truncation point if $Y_i \neq l$. The conditional mean and variance are

$$m_{il} = G_l(\boldsymbol{x}_i; \boldsymbol{T}, \boldsymbol{M}) + \boldsymbol{\Sigma}_{l(-l)}\boldsymbol{\Sigma}^{-1}_{(-l)(-l)}\big(\boldsymbol{z}_{i(-l)} - \boldsymbol{G}_{(-l)}(\boldsymbol{x}_i; \boldsymbol{T}, \boldsymbol{M})\big),$$

and

$$\tau_{il} = \boldsymbol{\Sigma}_{ll} - \boldsymbol{\Sigma}_{l(-l)}\boldsymbol{\Sigma}^{-1}_{(-l)(-l)}\boldsymbol{\Sigma}_{(-l)l},$$

where $\boldsymbol{\Sigma}_{l(-l)}$ is the l^{th} row of $\boldsymbol{\Sigma}$ with the l^{th} element removed, and $\boldsymbol{\Sigma}_{(-l)(-l)}$ is the sub-matrix of $\boldsymbol{\Sigma}$ with the l^{th} row and column removed.

We can now sample $\{\boldsymbol{T}_{lj}, \boldsymbol{M}_{lj}\}$ for $l = 1, \ldots, L-1$ and $j = 1, \ldots, m$ using the following construction. Given all tessellations and their output values except the j^{th} tessellation in the l^{th}

ensemble, along with $\boldsymbol{\Sigma}$ and $\boldsymbol{z}_{i(-l)}$, we observe that

$$z_{il}^{\dagger} = g(\boldsymbol{x}_i; \boldsymbol{T}_{lj}, \boldsymbol{M}_{lj}) + \epsilon_{il}, \quad \epsilon_{il} \sim \mathcal{N}(0, \tau_{il}),$$

where

$$z_{il}^{\dagger} = z_{il} - \sum_{k \neq j} g(\boldsymbol{x}_i; \boldsymbol{T}_{lk}, \boldsymbol{M}_{lk}) - \boldsymbol{\Sigma}_{l(-l)} \boldsymbol{\Sigma}_{(-l)(-l)}^{-1} \big(\boldsymbol{z}_{i(-l)} - \boldsymbol{G}_{(-l)}(\boldsymbol{x}_i; \boldsymbol{T}, \boldsymbol{M}) \big).$$

Now, the left-hand side of this equation can be considered as a partial residual $\boldsymbol{R}_j$, similar to Equation 4.4, associated with the j^{th} tessellation in the sum-of-tessellations. Hence, the usual Metropolis-Hastings algorithm in the AddiVortes method can be run with this residual.

Note that τ_{il} is constant for a given l, and so the condition that the variance of the error term is constant is satisfied in this case, meaning the AddiVortes model can be run almost identically to the original method.

After updating the tessellations and cell means, the temporary covariance matrix $\tilde{\boldsymbol{\Sigma}}$ is sampled from its full conditional distribution. Because the algorithm temporarily operates within the unconstrained parameter space, we can exploit the conjugacy of the inverse-Wishart prior with the multivariate normal likelihood of the latent residuals. By updating the prior scale matrix Ψ with the sample scatter matrix of these residuals and adding the sample size n to the prior degrees of freedom, $\tilde{\boldsymbol{\Sigma}}$ is updated from an inverse-Wishart distribution,

$$\tilde{\boldsymbol{\Sigma}} \sim \mathcal{W}^{-1} \left(\nu + n,\, \Psi + \sum_{i=1}^{n} (\boldsymbol{z}_i - \boldsymbol{G}(\boldsymbol{x}_i; \boldsymbol{T}, \boldsymbol{M}))(\boldsymbol{z}_i - \boldsymbol{G}(\boldsymbol{x}_i; \boldsymbol{T}, \boldsymbol{M}))^{\top} \right),$$

which is followed by rescaling under the trace restriction $\operatorname{tr}(\boldsymbol{\Sigma}) = L - 1$ to ensure identifiability.

Finally, a scaling step is performed so that the strictly identified model parameters can be used in the next iteration. We set $(\alpha_3^{(k)})^2 = \operatorname{tr}(\tilde{\boldsymbol{\Sigma}})/(L-1)$ and rescale the matrix as $\boldsymbol{\Sigma}^{(k)} = \tilde{\boldsymbol{\Sigma}}/(\alpha_3^{(k)})^2$, guaranteeing the trace constraint is met.

By iterating these steps, posterior draws of the latent variables, tessellation structures, ensemble means, and covariance matrix are obtained. For a new predictor $\boldsymbol{x}^*$, posterior predictive probabilities are estimated by the proportion of posterior draws in which each class maximises the latent

variable $\boldsymbol{z}^*$, that is,

$$p(Y^* = l \mid \boldsymbol{x}^*, \boldsymbol{X}_{\text{train}}) \approx \frac{1}{S}\sum_{s=1}^{S} \mathbb{I}\left\{ z_l^{*(s)} = \max_{k\in\{1,\ldots,L\}} z_k^{*(s)} \right\},$$

where S is the number of post burn-in samples of the posterior. Algorithm 4 gives the steps of this algorithm broken down into three stages.

Algorithm 4: Algorithm for Nominal Classification AddiVortes

Data: Initialise $\boldsymbol{z}^{(0)}, \boldsymbol{\Sigma}^{(0)}$, and tessellation ensembles $\{\boldsymbol{T}, \boldsymbol{M}\}$.
for $k = 1, \ldots, max\_iter$ **do**
  Step 1: Update latent variables $\boldsymbol{z}$ and scaling parameter α_1
  Draw $(\alpha_1^{(k)})^2 \sim \text{tr}(\Psi(\boldsymbol{\Sigma}^{(k-1)})^{-1})/\chi^2_{\nu(L-1)}$.
  for $i = 1, \ldots, n$ **do**
    for $l = 1, \ldots, L-1$ **do**
      Draw $z_{il}^{(k)} \sim \mathcal{TN}(m_{il}^{(k-1)}, \tau_{il}^{(k-1)})$,
      where $\boldsymbol{z}_{i(-l)} = (z_{i1}^{(k)}, \ldots, z_{i(l-1)}^{(k)}, z_{i(l+1)}^{(k-1)}, \ldots, z_{i(L-1)}^{(k-1)})$.
  Rescale: $\tilde{\boldsymbol{z}}^{(k)} = \alpha_1^{(k)}\boldsymbol{z}^{(k)}$.
  Step 2: Update tessellation ensembles
  for $l = 1, \ldots, L-1$ **do**
    for $j = 1, \ldots, m$ **do**
      Compute partial residuals $z_{il}^{\dagger}$ `// Equation 6.2`
      Propose a new tessellation T_{lj}^*
      Accept or reject proposed tessellation
      Sample $\boldsymbol{M}_{lj}$ `// From conjugate normal distribution`
  Set $\tilde{\boldsymbol{\mu}}^{(k)} = \boldsymbol{G}(\boldsymbol{x}; \{\boldsymbol{T}^{(k)}, \tilde{\boldsymbol{M}}^{(k)}\})$ and compute the normalised mean $\boldsymbol{\mu}^{(k)} = \tilde{\boldsymbol{\mu}}^{(k)}/\alpha_1^{(k)}$.
  Set the normalised cell means: $\boldsymbol{M}^{(k)} = \tilde{\boldsymbol{M}}^{(k)}/\alpha_1^{(k)}$.
  Step 3: Update covariance matrix $\boldsymbol{\Sigma}$ and scaling factor α_3
  Compute residuals $\boldsymbol{\epsilon} = \boldsymbol{z}^{(k)} - \boldsymbol{\mu}^{(k)}$.
  Draw $\tilde{\boldsymbol{\Sigma}} \sim \mathcal{W}^{-1}\big(n + \nu, \Psi + \sum_{i=1}^{n} \boldsymbol{\epsilon}_i\boldsymbol{\epsilon}_i^{\top}\big)$.
  Set $(\alpha_3^{(k)})^2 = \text{tr}(\tilde{\boldsymbol{\Sigma}})/(L-1)$.
  Rescale $\boldsymbol{\Sigma}^{(k)} = \tilde{\boldsymbol{\Sigma}}/(\alpha_3^{(k)})^2$.
Prediction: For new $\boldsymbol{x}^*$, draw

$$\boldsymbol{z}^* \sim \mathcal{N}(\boldsymbol{G}(\boldsymbol{x}^*; \{\boldsymbol{T}^{(\text{max\_iter})}, \boldsymbol{M}^{(\text{max\_iter})}\}), \boldsymbol{\Sigma}^{(\text{max\_iter})}),$$

and classify by $\hat{y}^* = \arg\max_{1\le l\le L} z_l^*(\boldsymbol{x}^*)$, where $z_L^* = 0$.

Given the results in Xu et al. (2025), we can update the algorithm in a manner similar to the two proposals presented in the paper. The crucial innovation in both proposals is to perform the computationally difficult AddiVortes backfitting update (Step 2) using the original, normalised latent variables, $\boldsymbol{z}$, as the response. This ensures the complex stochastic search for the tessellation

structures is conducted in a stable parameter space, allowing the regularising priors to function more effectively.

In the first proposal, step 2 is modified to update $\boldsymbol{G}(\boldsymbol{x})$ using the normalised $z_{il}^{\dagger}$ residuals. The algorithm reverts to the expanded, or unnormalised, space only for the covariance update in Step 3, calculating the residuals as $\tilde{\boldsymbol{\epsilon}}_i = \tilde{\boldsymbol{z}}_i - \alpha_1 \boldsymbol{G}(\boldsymbol{x}_i)$.

In the second proposal, there is a further simplification. Again, it updates the tessellations in the stable, normalised space during step 2. However, it also confines the augmentation entirely to the covariance update in Step 3. The residuals for this step are calculated directly in the normalised space as $\boldsymbol{\epsilon}_i = \boldsymbol{z}_i - \boldsymbol{G}(\boldsymbol{x}_i)$. This working covariance matrix is then rescaled to ensure the trace constraint is met.

We found through both simulation studies and applications to real-world data that the new proposals performed significantly better. This superior performance was observed in both posterior accuracy and parameter sampling. Therefore, in our analysis, we will present only the results from the new proposed algorithms.

6.3 Simulation studies

In this section, we use simulation studies to evaluate the AddiVortes classification models under controlled conditions where the true data-generating process is known. This allows us to rigorously assess performance in ways not possible with real-world data. We first investigate the binary classification case, using synthetic functions to compare AddiVortes's ability to capture oblique and highly non-linear patterns against traditional BART. We then assess the model's finite-sample performance in the nominal classification setting, focusing on its ability to recover class boundaries and predictive probabilities for a multi-category, non-linear problem.

6.3.1 Simulation study binary classification

In this section, the strengths of AddiVortes in comparison to tree-based models such as BART are illustrated through two simulation experiments using synthetic functions: the *rotated axis function* and the *sinusoid function* from Nguyen et al. (2024). These functions were selected to evaluate the model's ability to handle both oblique decision boundaries and highly non-linear patterns, which

are common in classification tasks.

The **rotated axis function** is designed to test a model's capability to approximate decision boundaries that are not aligned with the coordinate axes. For a given point $\boldsymbol{x} = (x_1, x_2)$, the function first applies a counter-clockwise rotation by an angle θ and then classifies the point based on its location relative to the rotated axes. Mathematically, the rotation is defined as

$$u(\boldsymbol{x}) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

where u_1 and u_2 are the rotated coordinates of $\boldsymbol{x}$. The rotated axis function $f(\boldsymbol{x}; \theta)$ is then defined as

$$f(\boldsymbol{x}; \theta) = \begin{cases} 1 & \text{if } u_1 \cdot u_2 > 0, \\ 0 & \text{otherwise,} \end{cases}$$

where $\theta \in [0, \pi/4]$ controls the angle of rotation. This function is depicted in Figure 6.1 (left) for $\theta = \pi/6$.

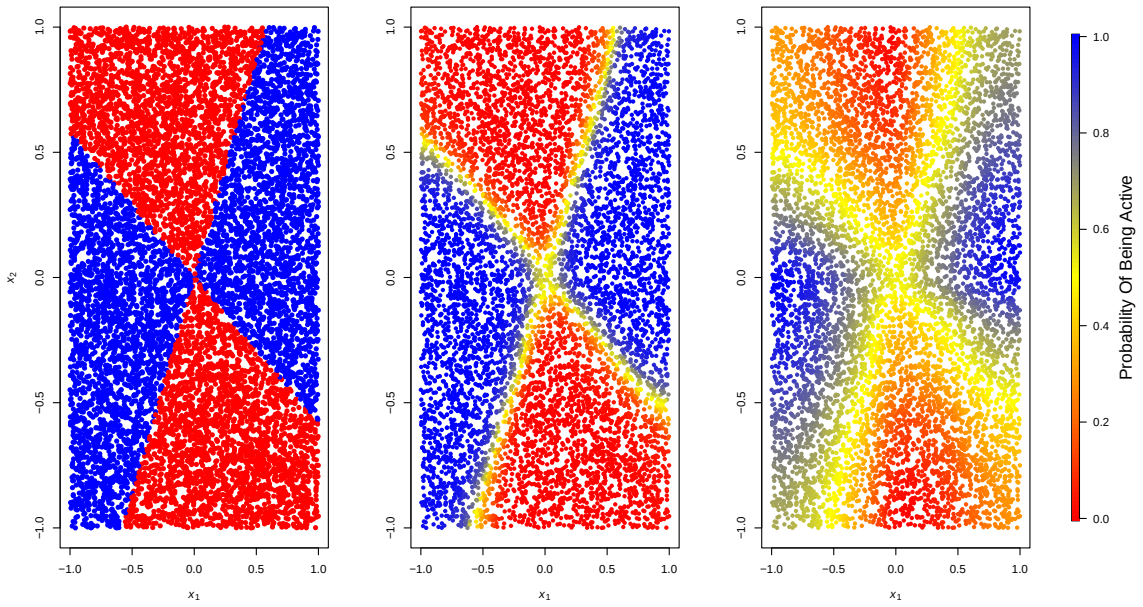


Figure 6.1: The rotational axis function with $\theta = \pi/6$ (left); the predicted probability of the rotated axis function for AddiVortes (middle) and BART (right).

The **sinusoid function** evaluates the model's ability to capture complex, non-linear decision

boundaries. The classification boundary is defined by a sinusoidal curve, with the outcome depending on whether the point lies above or below it. The sinusoid function $f(\boldsymbol{x}; \alpha)$ is given by

$$f(\boldsymbol{x}; \alpha) = \begin{cases} 1 & \text{if } x_2 > \alpha \sin(10x_1), \\ 0 & \text{otherwise,} \end{cases}$$

where α represents the amplitude of the sinusoidal curve. This function is shown in Figure 6.2 (left) for $\alpha = 0.5$.

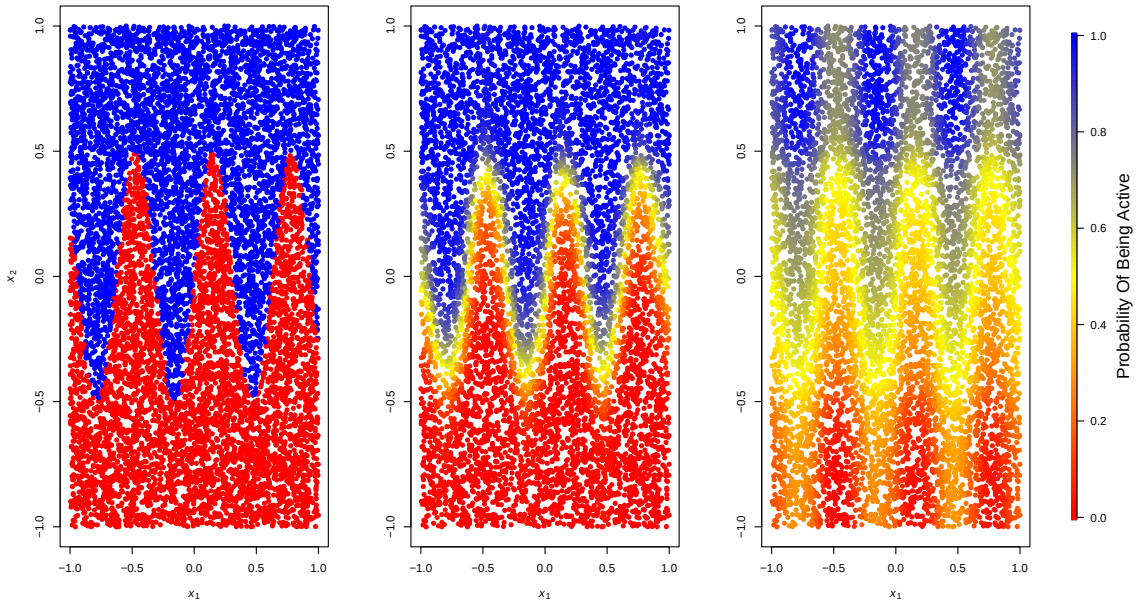


Figure 6.2: The sinusoid function with $\alpha = 0.5$ (left); the predicted probability of sinusoid for AddiVortes (middle) and BART (right).

To demonstrate the ability of the models to approximate these functions, we sampled n two-dimensional observations $\boldsymbol{x}_i \sim \text{Uniform}([0, 1]^2)$, and generated the dependent variables y_i based on the rotated axis function $f(\boldsymbol{x}; \theta)$ or the sinusoid function $f(\boldsymbol{x}; \alpha)$. For both functions, we applied the AddiVortes and BART models for binary classification.

Figures 6.1 (middle) and 6.2 (middle) illustrate the predicted probabilities for $Y = 1$ using AddiVortes, while Figures 6.1 (right) and 6.2 (right) present the results for BART. As shown, AddiVortes demonstrates greater certainty near the boundaries of the decision functions, whereas BART struggles to maintain consistency in these regions. You can see this by BART predicting

values much closer to 0.5 near the boundaries whereas AddiVortes correctly distinguishes between the classes better.

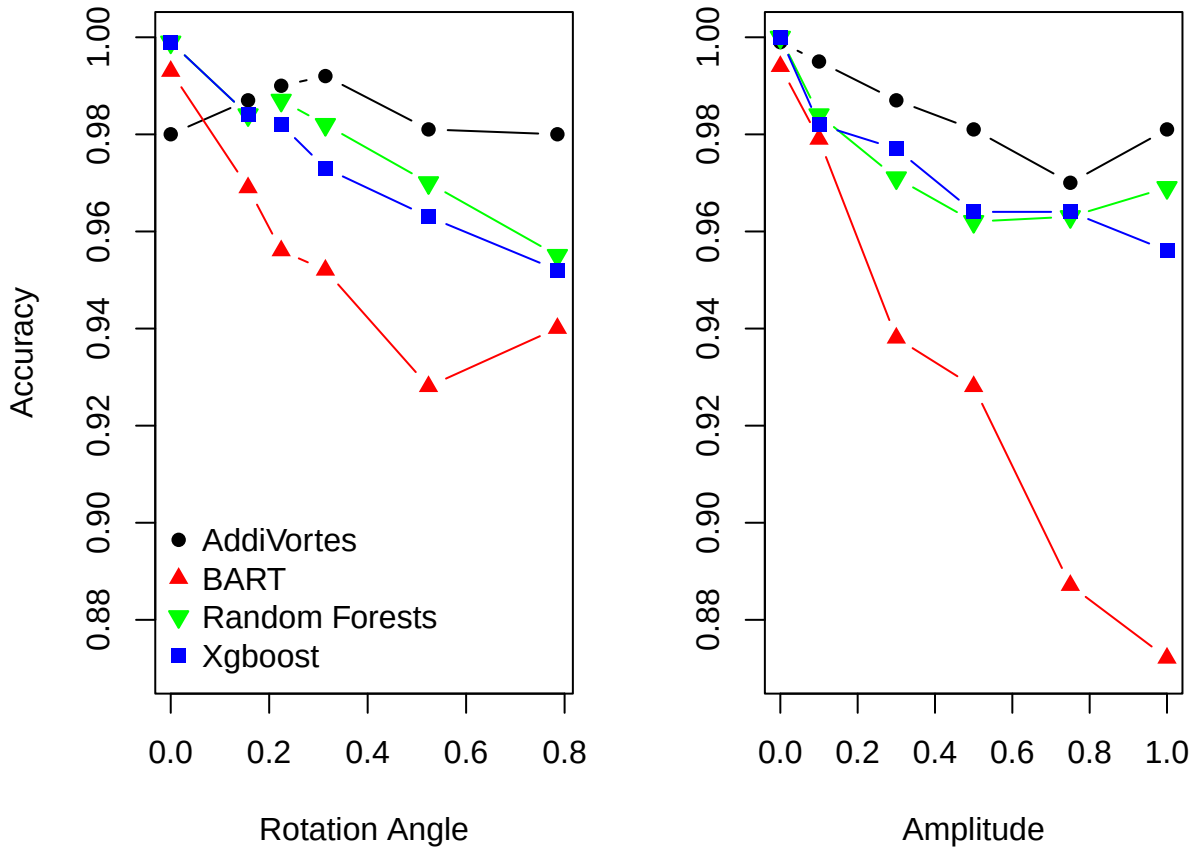


Figure 6.3: The accuracy of prediction for AddiVortes and competing methods for the rotated axis function for varying θ (left) and the sinusoid function for varying α.

To further illustrate the performance of the models, Figure 6.3 shows the accuracy for varying values of the parameters θ (for the rotated axis function) and α (for the sinusoid function). 1,000 observations were generated for each of the training and testing sets. The left panel of Figure 6.3 shows the accuracy as a function of the rotation angle θ, while the right panel displays the accuracy as a function of the amplitude α.

As competitors, we evaluated four "black-box" methods: random forests (Breiman, 2001, implemented as `randomforest` in R), (extreme) gradient boosting (Chen and Guestrin, 2016, implemented as `XGboost` in R) and BART (Chipman et al., 2010, implemented as `bart` from the

BayesTree R package). These models were selected for their robust binary classification capabilities, interpretability and comparability.

In the case of the rotated axis function, AddiVortes consistently achieves higher accuracy across all values of θ, demonstrating its robustness in capturing oblique decision boundaries. Notably, BART's performance decreases as θ increases, highlighting its difficulty in modelling decision boundaries that deviate significantly from the coordinate axes. This result underscores the advantage of the additive Voronoi tessellations in handling rotated feature spaces.

For the sinusoid function, AddiVortes again outperforms BART across the range of α values. The accuracy of AddiVortes remains stable even as the non-linearity of the boundary increases with larger values of α. In contrast, BART shows noticeable performance degradation for higher amplitudes, indicating its limited ability to adapt to highly non-linear decision boundaries.

6.3.2 Simulation study nominal classification

To facilitate comparison of nominal classification to the BART algorithm, we use a simulation study found in Xu et al. (2025). A training set and a test set are created, both of size 5000. The data generation process is based on a multinomial probit model with two latent utilities, corresponding to three distinct outcome categories. The model covariates consist of a set (U, V), where U is a vector of five independent variables $U = (U_1, \ldots, U_5)$ drawn from a Uniform(0,1) distribution, and V is a single variable drawn from a Uniform(0,2) distribution.

The mean function for the first utility, z_1, is given by

$$G_1(u) = 15\sin(\pi U_1 U_2) + (U_3 - 0.5)^2 - 10U_4 - 5U_5.$$

To produce relatively balanced outcome categories, with a distribution of (0.45, 0.25, 0.30), the second mean function is defined as

$$G_2(u, v) = (U_3 - 0.5)^3 - 20U_4U_5 + 4V.$$

The latent variables Z are generated by adding noise from a multivariate normal distribution, such that $z \sim MN(G(u, v), \Sigma)$, where $G(u, v) = (G_1(u), G_2(u, v))^T$. This noise is drawn using a

true covariance matrix of

$$\Sigma = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix},$$

and the final observed categorical outcome Y is then determined from Z according to the MNP framework. That is,

$$Y_i = \begin{cases} 1, & \text{if } z_{i1} = \max\{z_{i1}, z_{i2}\} \geq 0, \\ 2, & \text{if } z_{i2} = \max\{z_{i1}, z_{i2}\} \geq 0, \\ 3, & \text{if } \max\{z_{i1}, z_{i2}\} < 0. \end{cases}$$

We use random forests (Breiman, 2001, implemented as `randomforest` in R) and BART (Sparapani et al., 2021, implemented as `mbart` in R) for competitors for nominal classification in this simulation study. To reduce computational expense we used the default values from the packages for the hyperparameters. Matching the study found in Xu et al. (2025), for BART we collected 3,000 posterior draws after a 5,000 iteration burn-in period.

Similarly, to allow for easy comparison of BART, for AddiVortes, 3,000 posterior draws were collected for each algorithm after a 5,000-iteration burn-in. Each mean component G_l is parametrised as a sum of 200 tessellations, using the standard AddiVortes priors and default hyperparameters, $(m, k, \sigma_c, \omega, \lambda_c) = (200, 3, 0.8, 3, 25)$. To sample the latent covariance $\tilde{\Sigma} \sim \text{Inv-Wishart}(\nu, \Psi)$, we use the default hyperparameters $\psi_{11} = \psi_{22} = 1$, and consider the uniform prior $(\nu = C + 1, \psi_{12} = 0)$. We only use the first proposal from Section 6.2 of the updated versions of the AddiVortes algorithm for nominal classification, since the results from the second proposal are similar.

In addition to accuracy as a metric to access the performance of the models, we also used the F_1 score and balanced accuracy.

The F_1 score is defined as the harmonic mean of precision and recall, expressed as

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

The F_1 score ranges from 0 to 1, where a value closer to 1 indicates a better model that successfully captures the majority of actual cases while maintaining high predictive accuracy.

Balanced Accuracy is defined as the mean of sensitivity and specificity, calculated as

$$\text{Balanced Accuracy} = \frac{\text{Sensitivity} + \text{Specificity}}{2}.$$

This metric accounts for class imbalance by ensuring that performance on smaller groups is weighted equally to the majority class. A score of 1 represents perfect prediction, while 0.5 indicates random guessing.

Model	**Accuracy**	F_1 **Score**	**Balanced Accuracy**
AddiVortes	0.9244	0.9187	0.9393
BART	0.9134	0.9073	0.9300
Random Forest	0.8916	0.8850	0.9113

Table 6.1: Summary of performance metrics averaged over 20 runs for classification models.

We ran each algorithm 20 times and the average value of predictive performance metrics are given in table 6.1. We see that, AddiVortes has the highest Accuracy of 92.44%, surpassing the BART model's 91.34% and the Random Forest model's 89.16%, establishing it as an effective classifier for this simulation study.

In the simulation study, the AddiVortes model achieves the highest F_1 scores across all categories, with a mean of 0.9187 ((class 1, class 2 class 3) = $(0.949, 0.897, 0.910)$). These results consistently outperform the BART model, which has a mean of 0.9073 $(0.939, 0.884, 0.899)$, as well as the Random Forest model, which is slightly lower with a mean of 0.8850 $(0.918, 0.860, 0.877)$. This comparison confirms that AddiVortes is an effective an algorithm for balancing sensitivity with precision, making it a reliable tool for classification problems.

Across all three classes, the Balanced Accuracy for the AddiVortes model remains consistently high $(0.9545, 0.9315, 0.9316)$ with a mean of 0.9393, suggesting its performance is not only due to the class imbalance but a true reflection of its ability to classify instances across different categories.

AddiVortes

	C1	C2	C3
P1	2159	59	80
P2	50	1112	73
P3	43	73	1351

BART

	C1	C2	C3
P1	2142	84	85
P2	46	1084	78
P3	64	76	1341

Random Forest

	C1	C2	C3
P1	2118	111	136
P2	68	1041	69
P3	66	92	1299

Table 6.2: Side-by-side comparison of confusion matrices for AddiVortes, BART, and random forest models.

The confusion matrices shown in Tables 6.2, provide insight into the specific types of errors made by the models. We see that AddiVortes recorded the fewest total misclassification among the three models. In contrast, the Random Forest model struggled more to distinguish classes, particularly when Class 1 samples were incorrectly assigned to Class 2 and Class 3, resulting in 111 and 136 errors, respectively. This pattern of misclassification suggests that while all models are competent, the use of Voronoi tessellation in the AddiVortes model is better at delineating the boundaries between the three target classes.

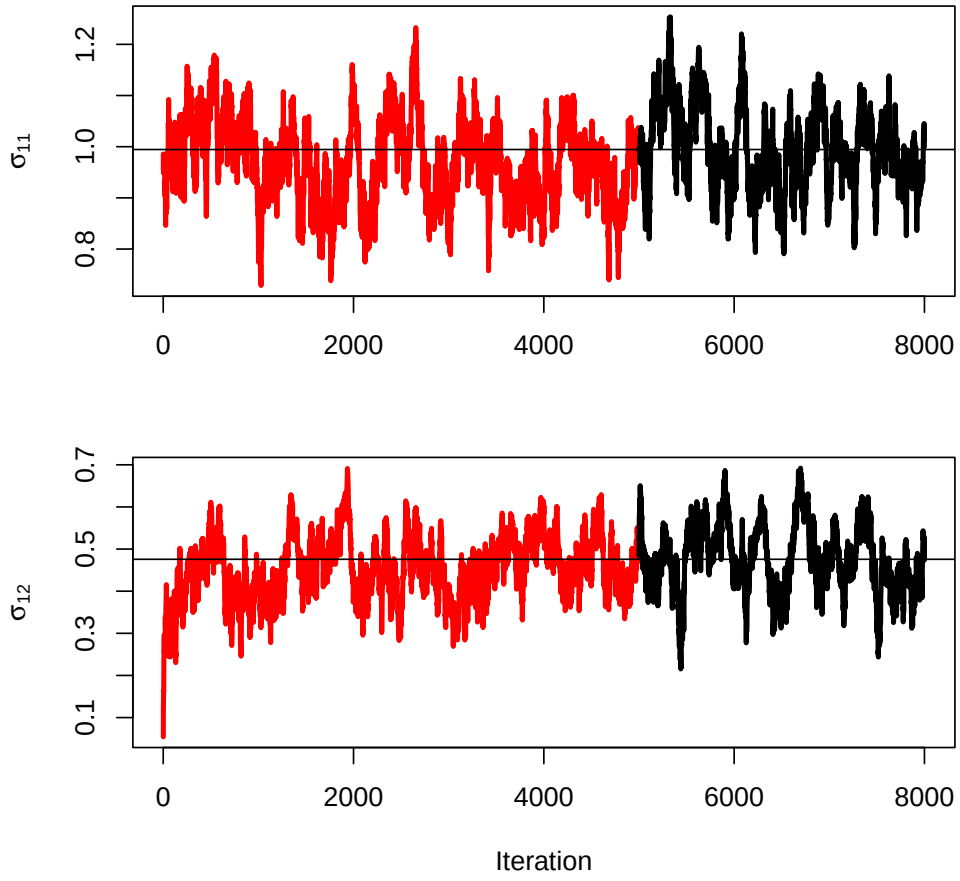


Figure 6.4: Trace of σ_{11} (top) and σ_{12} for a burn-in period of 5000 iterations in red and a post burn-in period of 3000 in black. The horizontal line indicates the mean of the posterior samples.

In addition to predictive performance, the reliability of the model can be assessed through the trace plots of the posterior samples for the variance-covariance parameters, as illustrated in Figure 6.4. For this simulation study, the true underlying values for σ_{11} and σ_{12} are known to be

1 and 0.5, respectively. The trace plots demonstrate that the AddiVortes algorithm successfully explores the parameter space, with the sampled values for σ_{11} and σ_{12} oscillating consistently around their respective true targets. The posterior means for both parameters are remarkably close to these true values, indicating that the algorithm not only achieves stable convergence but also accurately recovers the latent structure of the data-generating process.

6.4 Benchmark data sets

We compared the predictive performance of AddiVortes against several competing algorithms for six benchmark datasets, five from the UCI Machine Learning Repository: Breast Cancer, Ionosphere, Sonar, Heart Failure and German credit and one from the IEEEdataPort on digital sensors detecting oil or water. These data sets exhibit varying sample sizes, ranging from 208 to 4,475 observations, and each includes a binary output variable and between 9 and 60 covariates.

As metrics, we use the accuracy as defined in Section 6.3 and the Area Under the Curve (AUC) values. The AUC is a metric used to evaluate the performance of binary classification models by measuring the area under the Receiver Operating Characteristic (ROC) curve. The ROC curve plots the true positive rate against the false positive rate across different classification thresholds. An AUC value of 0.5 indicates random guessing, while a value of 1 represents perfect classification.

Name	n	**TC(%)**	**# Cov**
Sonar	208	53.4%	60
Heart Failure	299	32.1%	12
German Credit	1000	30.0%	24
Breast Cancer	683	35.0%	9
Digital Sensors	4475	57.4%	10
Ionosphere	351	64.1%	32

Table 6.3: A table showing the data sets used in our analysis, the percentage of observations in the target class and the number of covariates in the datasets.

For each dataset, we generated 20 independent train/test split samples (sampling without replacement), allocating 80% of the data to the training set and 20% to the test set. All models were subjected to five-fold cross-validation to optimise hyperparameters, with the ranges of values considered for each method detailed in Table 6.4.

Tables 6.5 and 6.6 present the accuracy and AUC values of AddiVortes (AV) compared to

Method	**Parameters**	**Values considered**
Random forests	Number of trees	500
	% variables sampled to grow each node	10, 25, 50, 100
XGBoost	Number of trees	50, 100, 200
	Shrinkage (multiplier of each tree added)	0.01, 0.05, 0.10, 0.25
	Max depth permitted for each tree	3, 5, 7
BART	Number trees, m	50, 200
	μ prior: k value for σ_μ	1, 2, 3, 4, 5
AddiVortes	# Tessellations: m	200
	μ prior: k value for σ_μ	3
	Standard deviation of centre location: σ_c	0.2, 0.4
	Probability weight for # covariates: ω	3, 5
	Poisson rate for # centres: λ_c	15, 30, 45

Table 6.4: A table showing the cross-validation values for competing methods.

competing methods, namely BART, Random Forest (RF), and XGBoost. These results highlight the performance and robustness of AddiVortes in binary classification tasks.

Dataset	AV	BART	RF	XGBoost
Sonar	**86.4**	75.7	81.3	83.8
Heart Failure	**85.8**	80.3	85.3	83.8
Breast Cancer	**97.2**	96.6	**97.2**	96.3
German Credit	76.4	76.1	**76.5**	75.9
Ionosphere	**93.7**	88.7	93.2	91.8
Digital Sensor	**99.7**	96.0	99.0	99.7

Table 6.5: A table showing the Accuracy(%) of each method for the datasets.

Table 6.5 shows that AddiVortes achieves the highest accuracy in four of the six datasets, demonstrating its adaptability to diverse data structures. Even when not achieving the top accuracy, as in the Breast Cancer and German Credit datasets, its performance remains highly competitive. This ability to maintain strong results across varied scenarios underscores its robustness and versatility against established methods.

Dataset	AV	BART	RF	XGBoost
Sonar	**95.2**	92.1	94.4	93.4
Heart Failure	**91.8**	91.5	91.6	89.7
Breast Cancer	**99.4**	99.3	99.1	99.2
German Credit	**79.3**	78.6	78.9	78.3
Ionosphere	**98.0**	97.6	97.4	96.7
Digital Sensor	**100.0**	99.8	99.9	99.9

Table 6.6: A table showing the AUC of each competitor on the datasets.

Table 6.6 provides additional insights into the discriminative ability of the models. AddiVortes achieves the highest AUC across all six datasets, confirming its superior ability to correctly rank positive and negative classes across various thresholds. This strong performance can be attributed to its ability to partition the covariate space more flexibly than competitors that use standard decision trees. These results further confirm the reliability of AddiVortes in datasets with varying complexities.

The combined results from Tables 6.5 and 6.6 demonstrate the strength of AddiVortes in delivering high accuracy and robust classification performance. While other methods exhibited more variability in their rankings, AddiVortes consistently delivered competitive, top-tier results, making it a reliable choice for binary classification. These findings establish it as a powerful and generalisable tool for predictive modelling, particularly in scenarios that demand the ability to model complex data structures effectively.

6.5 Home mortgage acceptance predictions

Enacted in 1975, HMDA was established in America to promote transparency in mortgage lending practices. It requires financial institutions to maintain, report, and publicly disclose loan-level information about mortgages. The primary goal of HMDA is to provide the public with sufficient data to help identify potential discriminatory lending practices, ensure that financial institutions are meeting the housing needs of their communities, and facilitate the allocation of public and private investment in housing.

The data collected under HMDA is extensive, encompassing various attributes of both loan applications and the applicants themselves. This data includes information on the loan type, amount, property value, applicant demographics (such as race, gender, and income), and the outcome of the application (whether it was approved or denied). The comprehensive nature of this data allows for detailed analysis of lending patterns and practices.

The Home Mortgage Disclosure Act (HMDA) data has been extensively utilised in research to explore various aspects of mortgage lending practices. For instance, it has been used to analyse whether minority applicants face adverse lending outcomes in the mortgage market, as discussed by (Popick, 2022); study racial and ethnic disparities in mortgage lending (Lewis-Faupel and Tenev,

2024) and to investigate potential discrimination by banks in lending practices (Cyree and Winters, 2023).

Given the vast amount of information in HMDA data, it is particularly valuable for predictive analytics. Predicting whether a loan application will be accepted is not only beneficial for financial institutions but also for prospective borrowers and policymakers.

Variable	Description
derived_loan_product_type	Derived loan product type from Loan Type and Lien Status fields for easier querying of specific records
derived_dwelling_category	Derived dwelling type from Construction Method and Total Units fields for easier querying of specific records
derived_race	Single aggregated race categorisation derived from applicant
derived_sex	Single aggregated sex categorisation derived from applicant
purchaser_type	Type of entity purchasing a covered loan from the institution
conforming_loan_limit	Indicates whether the reported loan amount exceeds the GSE (government-sponsored enterprise) conforming loan limit
preapproval	Whether the covered loan or application involved a request for a preapproval of a home purchase loan under a preapproval program
loan_type	The type of covered loan or application
loan_purpose	The purpose of the covered loan or application
lien_status	Lien status of the property
open-end_line_of_credit	Whether the covered loan or application is for an open-end line of credit
business_or_commercial_purpose	Whether the covered loan or application is primarily for a business or commercial purpose
loan_amount	The amount of the covered loan, or the amount applied for
combined_loan_to_value_ratio	The ratio of the total amount of debt secured by the property to the value of the property relied on in making the credit decision
hoepa_status	Whether the covered loan is a high-cost mortgage
interest_only_payment	Whether the contractual terms include, or would have included, interest-only payments
balloon_payment	Whether the contractual terms include, or would have included, a balloon payment
other_nonamortizing_features	Whether the contractual terms include, or would have included, any non-standard term that would allow for payments other than fully amortising payments during the loan term
property_value	The value of the property securing the covered loan relied on in making the credit decision
occupancy_type	Occupancy type for the dwelling
total_units	The number of individual dwelling units related to the property
income	The gross annual income, in thousands of dollars
debt_to_income_ratio	The ratio, as a percentage, of the applicant's total monthly debt to the total monthly income
applicant_credit_score_type	The name and version of the credit scoring model used to generate the credit score
applicant_ethnicity	Ethnicity of the applicant
applicant_race	Race of the applicant
applicant_race_observed	Whether the race of the applicant was collected on the basis of visual observation or surname
applicant_sex_observed	Whether the sex of the applicant was collected on the basis of visual observation or surname
submission_of_application	Whether the applicant submitted the application directly to the financial institution
initially_payable_to_institution	Whether the obligation was initially payable to the financial institution
aus	The automated underwriting system (AUS) used by the financial institution to evaluate the application
tract_population	Total population in tract
tract_minority_population_percent	Percentage of minority population to total population for tract, rounded to two decimal places
ffiec_msa_md_median_family_income	FFIEC Median family income in dollars for the MSA/MD in which the tract is located
tract_to_msa_income_percentage	Percentage of tract median family income compared to MSA/MD median family income
tract_owner_occupied_units	Number of dwellings, including individual condominiums, that are lived in by the owner
tract_one_to_four_family_homes	Dwellings that are built to houses with fewer than 5 families
tract_median_age_of_housing_units	Tract median age of homes

Table 6.7: Table of the features of the home mortgage dataset.

Financial institutions can use predictive models to strengthen decision-making processes, reduce processing times, and improve customer service. For borrowers, understanding the factors that influence loan approval can significantly improve their chances of success, enabling them to tailor their applications accordingly. On a broader scale, insights from predicting loan outcomes using

HMDA data can inform policy decisions to promote fair lending practices and ensure equitable access to credit.

We use the "One Year National Loan Level Dataset" for 2022 and 2023, published under the Home Mortgage Disclosure Act by the FFIEC (Federal Financial Institutions Examination Council), which contains Mortgage information for over a million home mortgage applicants in America. We train the models on the 2022 dataset to predict whether a single-person applicant will be accepted or rejected for a mortgage, based on a set of features listed in Table 6.7, and test them on the 2023 mortgage data. We have preprocessed the data by removing any applicants with missing data and all features relevant only to joint applicants.

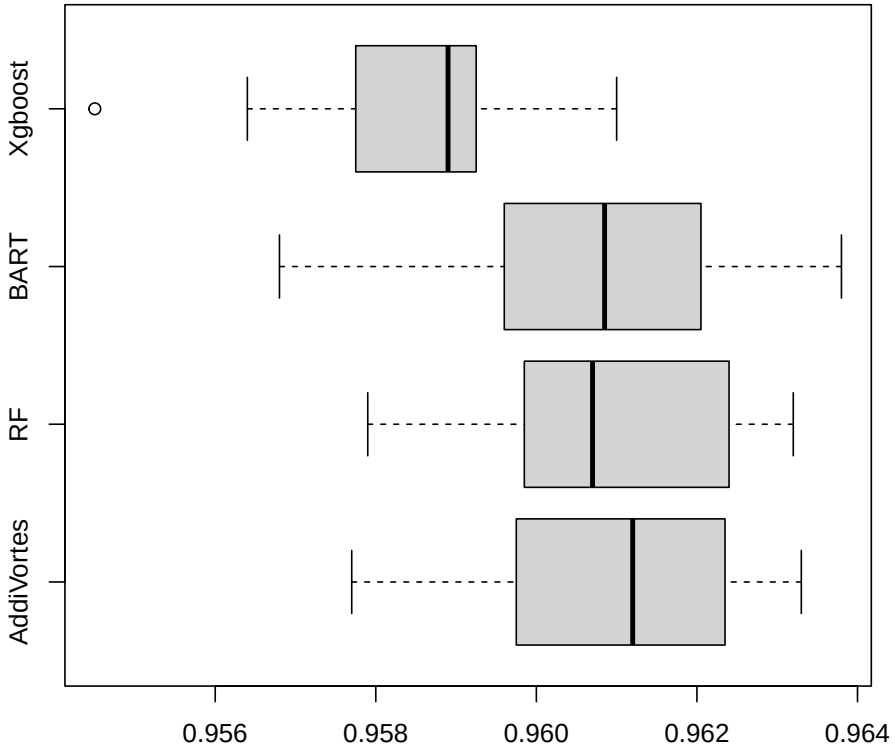


Figure 6.5: A boxplot of the accuracy values for each competing method on the home mortgage dataset.

To reduce computational time, we sampled 20,000 applicants for the training and testing sets.

To compare methods on the home mortgage dataset, we use the same competitors as in Section 6.4, tuning the hyperparameters by sampling five distinct sets of 20,000 samples from the 2022 mortgage data to train the models and 20,000 samples from the 2023 data, and choosing the hyperparameters from Table 4.3 that give the best accuracy values.

To produce the boxplots in Figures 6.5 and 6.6, we sampled 24 sets of 20,000 samples from the 2022 mortgage data to train the models and 20,000 samples from the 2023 data to obtain 24 accuracy and AUC values for each method.

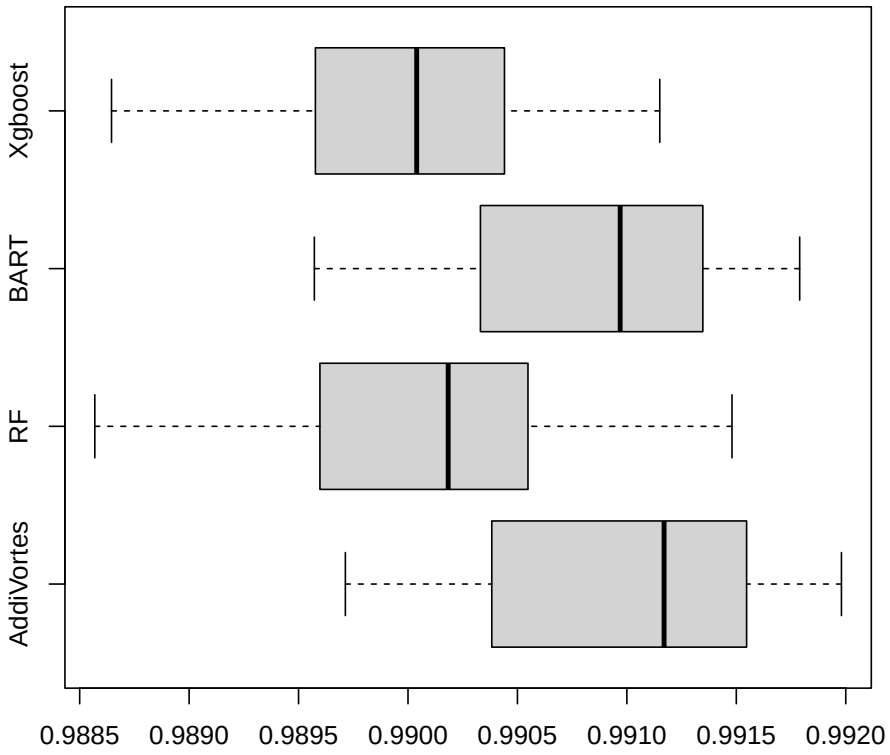


Figure 6.6: A boxplot of the AUC values for each competing method on the home mortgage dataset.

As shown in Figure 6.5, AddiVortes consistently achieves the highest accuracy among all competing methods on the Home Mortgage dataset. The boxplot illustrates the variation in accuracy across multiple runs, highlighting the robustness of AddiVortes in mortgage approval prediction. The median accuracy of AddiVortes exceeds that of Random Forest, BART, and XGBoost, demon-

strating its ability to capture relationships in the data. Note that the accuracy is much higher than the median base accuracy of 79% for all methods.

Similarly, Figure 6.6 presents the area under the curve (AUC) values for the same dataset. AddiVortes achieves the highest median AUC, indicating superior discriminative power in classifying approved and rejected loan applications. The narrow spread of the AUC values for AddiVortes suggests that it maintains consistent performance across different training and testing samples. This reliability is crucial in financial decision-making, where model stability is essential.

These results further reinforce the advantages of AddiVortes in binary classification tasks, particularly in mortgage lending applications. Its ability to outperform traditional "black-box" models while maintaining interpretability makes it a valuable tool for financial institutions seeking to enhance their predictive analytics frameworks.

Figure 6.7 illustrates the predicted probabilities $P[Y = 1 \mid x]$ for a test set of size 200, derived from a model trained on 20,000 observations. The graph includes 90% posterior intervals, which convey the uncertainty in the predictions, along with markers indicating whether each loan was accepted ($Y = 1$) or rejected ($Y = 0$). The posterior intervals widen for observations far from the majority of the training data, reflecting increased uncertainty. Notably, the majority of misclassified observations have posterior intervals that include $P[Y = 1 \mid x] = 0.5$, indicating ambiguity in these predictions. The overall test set accuracy was 97.5%, significantly exceeding the 70% base rate, highlighting the effectiveness of AddiVortes in predicting loan approval outcomes.

Figure 6.8 provides further insight into how AddiVortes constructs its predictive model by illustrating the percentage of tessellations in which each variable is included across 100 runs. Variables with higher inclusion rates are considered more influential in determining mortgage approval outcomes. Key predictors such as debt-to-income ratio, credit score type, and loan-to-value ratio are among the most frequently included, aligning with established financial risk assessment criteria. Additionally, tract population, minority population percentage, and applicant demographics appear frequently, suggesting that AddiVortes captures both individual financial attributes and broader socioeconomic factors.

Our evaluation in this section focuses on overall predictive performance, as measured by accuracy and AUC, rather than on formal fairness criteria. A comprehensive fairness analysis would require group-specific error rates or disparity measures (e.g. across race and sex groups), which

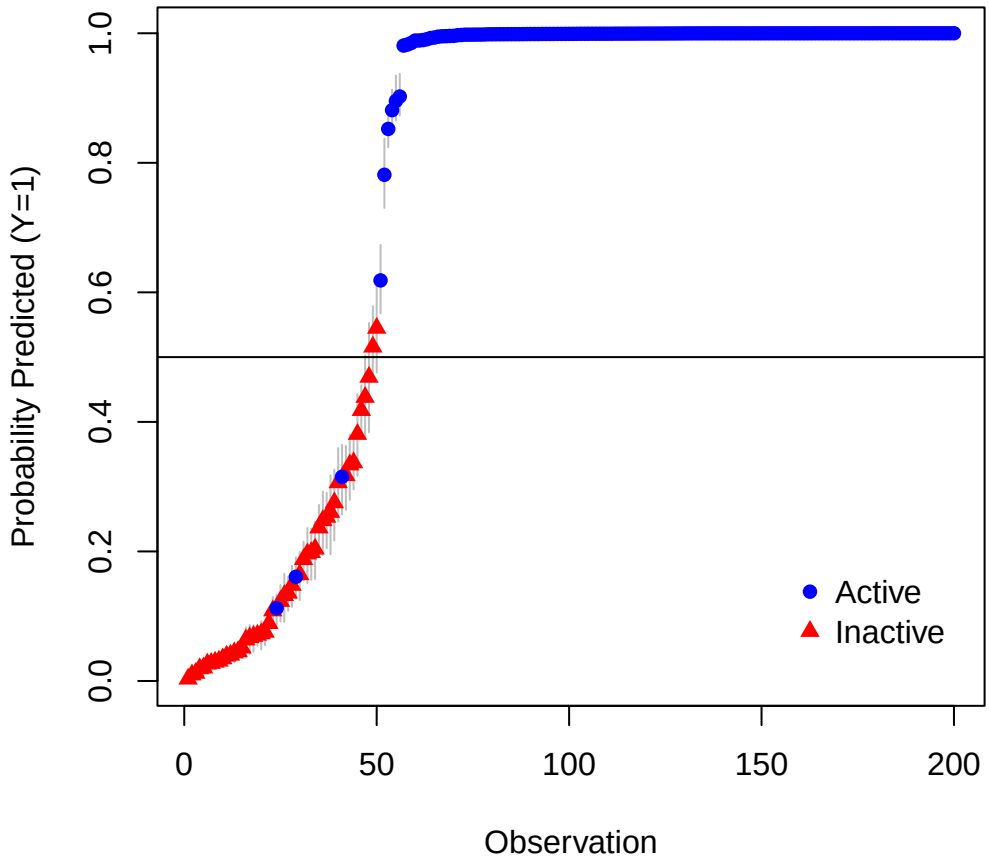


Figure 6.7: A graph of the predicted probabilities of an observation being active with the 90% confidence intervals for the test set and the true status of activity.

we view as an important direction for future work. Nonetheless, the variable inclusion frequencies in Figure 6.8 highlight the extent to which protected attributes and neighbourhood characteristics influence predictions, providing a starting point for such analyses.

6.6 Discussion

This chapter has presented a comprehensive framework for extending the AddiVortes methodology to binary and multinomial classification tasks. By leveraging a data augmentation strategy based on the probit link function, we have successfully mapped discrete categorical outcomes to continuous latent variables, thereby allowing the core sum-of-tessellations machinery to be applied effectively to

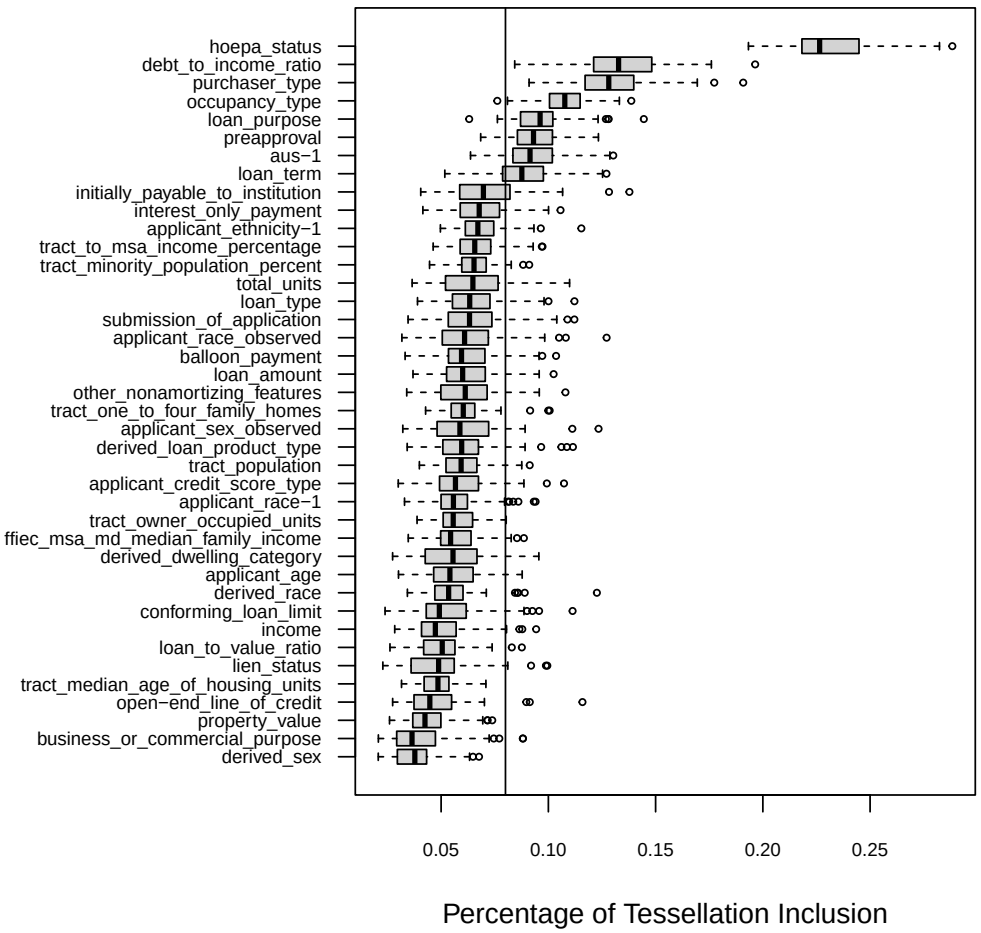


Figure 6.8: Boxplots showing the percentage of times a variable is included in a tessellation. A vertical line is drawn at 8% inclusion percentage.

classification problems. The formulation for binary classification, where the probability is modelled as

$$P(Y = 1 \mid \boldsymbol{x}) = \Phi(G(x)),$$

and the subsequent extension to the multinomial setting via a vector of latent utilities, provides a statistically rigorous foundation that preserves the benefits of Bayesian uncertainty quantification. A key theoretical advantage identified in this work is the ability of the Voronoi partitions to capture complex, oblique decision boundaries more naturally than the axis-aligned splits inherent to traditional tree-based models like BART.

The empirical evaluations conducted through simulation studies and benchmark datasets have validated the practical efficacy of this approach. In controlled simulations, particularly the rotated axis and sinusoid experiments, AddiVortes demonstrated superior adaptability to non-linear and rotated geometric structures where standard tree ensembles often struggle to approximate the true decision boundary. This flexibility translated into consistently strong performance across a suite of real-world benchmark datasets, where the model frequently achieved competitive or superior accuracy and AUC scores compared to established "black-box" methods such as Random Forests and Gradient Boosting. The results confirm that the added geometric flexibility of the tessellation structure yields tangible gains in discriminative power, particularly in settings where the separation between classes is not orthogonal to the feature axes.

Furthermore, the application of the model to substantial real-world problems, specifically the Home Mortgage Disclosure Act (HMDA) dataset and the dry bean classification task, highlights the method's utility in diverse domains. In the mortgage analysis, AddiVortes not only delivered high predictive accuracy but also offered crucial interpretability regarding feature importance, a vital requirement for transparency in financial modelling. These findings establish AddiVortes as a versatile and powerful tool for classification, offering a distinct and effective alternative to tree-based ensembles capable of modelling the intricate decision boundaries found in modern data analysis.

Chapter 7

Heteroscedastic AddiVortes

The task of developing flexible regression methods capable of capturing relationships between multidimensional predictors $\boldsymbol{x}$ and a continuous response Y remains a central challenge in modern statistics and machine learning. While several powerful approaches have been proposed, such as Gaussian process regression (Schulz et al., 2018), random forests (Breiman, 2001), gradient boosting machines (Friedman, 2001), and deep neural networks for regression (DNNR) (Chen et al., 2023), many of these methods primarily focus on modelling the conditional mean $\mathbb{E}[Y \mid \boldsymbol{x}]$ under the assumption of constant error variance. That is, they typically assume homoscedasticity, whereby the conditional variance of Y given $\boldsymbol{x}$ remains fixed across the input space and does not depend on the covariates.

However, the assumption of homoscedasticity is often unrealistic in practical applications. Real-world datasets from many different industries, including medical data (Zhang et al., 2024), financial markets (Nguyen et al., 2021), and climate science (Lee and Liu, 2023), frequently exhibit heteroscedasticity, where the variance of the response variable varies systematically with the covariates. Ignoring this structure can adversely affect both predictive accuracy and the reliability of uncertainty quantification. Models that explicitly account for heteroscedasticity provide a better representation of the data-generating process by allowing the conditional variance of Y to depend on $\boldsymbol{x}$, thereby improving both interpretability and predictive performance.

As we saw in Chapter 2, BART is adapted to deal with heteroscedastic data by modelling the error variance as a function of the covariates, rather than assuming a constant global noise parameter (Pratola et al., 2020). This is achieved by introducing a second sum-of-trees component

to estimate the variance.

This chapter builds upon the heteroscedastic BART framework and introduces the heteroscedastic AddiVortes model, an extension of the AddiVortes framework designed to jointly model both the conditional mean $\mathbb{E}[Y \mid \boldsymbol{x}]$ and conditional variance $\mathbb{V}\text{ar}[Y \mid \boldsymbol{x}]$. The methodology uses separate ensembles of multidimensional tessellations to capture these two components. The conditional mean is modelled using a "sum-of-tessellations" ensemble, which aggregates additive effects across components, while the conditional variance is represented via a "product-of-tessellations" ensemble that defines a flexible multiplicative structure for the variance surface, thereby ensuring that the resulting variance estimates remain strictly positive. This dual-ensemble formulation enables modelling of heteroscedastic data and supports efficient computation via conditional conjugacy, thereby facilitating intuitive and coherent prior specification.

In multidimensional settings, this enhanced AddiVortes framework enables simultaneous inference on both the mean and variance structures, capturing the posterior distribution more accurately than conventional homoscedastic models. Its flexibility in capturing interactions and heteroscedastic patterns makes it a powerful and versatile tool for a wide range of data analysis applications.

The remainder of the chapter is organised as follows. Section 7.1 extends the homoscedastic AddiVortes framework to the heteroscedastic setting, allowing the noise of the variance to be dependent on the covariates. In Section 7.2, we demonstrate the capabilities of heteroscedastic AddiVortes through a series of simulated experiments. Sections 7.3 and 7.4 present comparisons between heteroscedastic AddiVortes, heteroscedastic BART, and the original homoscedastic AddiVortes on real-world datasets, using a variety of performance metrics. Finally, Section 7.5 concludes the chapter with a summary and discussion.

7.1 Heteroscedastic AddiVortes

In this section, we extend the AddiVortes model to accommodate data with non-constant variance. The original AddiVortes model assumes a homoscedastic error structure, in which the variance of the response Y remains constant regardless of the predictor values. However, practical data frequently exhibit a covariate-dependent variance function. Similar to heteroscedastic BART, our heteroscedastic AddiVortes approach jointly models the conditional mean function $\mathbb{E}[Y \mid \boldsymbol{x}] = f(\boldsymbol{x})$

and the conditional variance function $\text{Var}[Y \mid \boldsymbol{x}] = s^2(\boldsymbol{x})$, leading to a more realistic modelling framework. Specifically,

$$Y = f(\boldsymbol{x}) + \epsilon, \quad \epsilon \sim N(0, s^2(\boldsymbol{x})),$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$ is a p-dimensional vector of predictors. While we assume for simplicity that the same predictors influence both $f(\boldsymbol{x})$ and $s^2(\boldsymbol{x})$, this assumption is not strictly necessary.

Our methodology uses two distinct ensembles of Voronoi tessellations: an additive tessellation model for the mean function, denoted by $f(\boldsymbol{x}) = \sum_{j=1}^{m} g(\boldsymbol{x}; T_j, M_j)$, and a multiplicative regression tessellation model for the variance component,

$$s^2(\boldsymbol{x}) = \prod_{l=1}^{m'} h(\boldsymbol{x}; T_l', M_l').$$

Here, as seen in previous chapters, $g(\boldsymbol{x}; T_j, M_j)$ represents the output value of the j^{th} tessellation for the mean, while $h(\boldsymbol{x}; T_l', M_l')$ represents the output value of the l^{th} tessellation for the variance.

In this formulation, T_j denotes the structure of the j^{th} tessellation for the mean, with $\boldsymbol{M}_j = \{\mu_{1j}, \ldots, \mu_{b_j j}\}$ representing the $b_j = |\boldsymbol{M}_j|$ scalar cell output values. Similarly, T_l' denotes the structure of the l^{th} tessellation for the variance, and $\boldsymbol{M}_l' = \{s^2_{1l}, \ldots, s^2_{b_l' l}\}$ represents the $b_l' = |\boldsymbol{M}_l'|$ scalar cell output values. This approach allows $f(\boldsymbol{x})$ to be modelled additively, capturing its effects via a sum of tessellations, while $s^2(\boldsymbol{x})$ is modelled multiplicatively, capturing its scale via a product of tessellations.

7.1.1 Mean Model

The mean is modelled in a manner nearly identical to the original AddiVortes, except that the variance of the error term is no longer assumed constant. Similar to the homoscedastic setting, we sample from $(T_j, \boldsymbol{M}_j) \mid \boldsymbol{R}_j, s^2(\boldsymbol{x})$, where

$$\boldsymbol{R}_j = \boldsymbol{y} - \sum_{q \neq j} g(\boldsymbol{x}; T_q, \boldsymbol{M}_q).$$

Recall that this is equivalent to the posterior of the single tessellation model $\boldsymbol{R}_j = g(\boldsymbol{x}; T_j, \boldsymbol{M}_j) + \epsilon$, where $\boldsymbol{R}_j$ plays the role of the data $\boldsymbol{y}$. Thus, if we let the set Z_{ij} be the set of all observations

that fall in the cell associated with μ_{ij} and, to ease notation, set $s^2(\boldsymbol{x}_z) = s_z^2$, then when viewed as a function of μ_{ij}, the heteroscedastic AddiVortes likelihood of μ_{ij} in the i^{th} cell of the j^{th} mean tessellation is given by

$$L(\mu_{ij} \mid \cdot) = p(\boldsymbol{R}_{ij} \mid T_j, \mu_{ij}, \sigma) = \prod_{z \in Z_{ij}} \frac{1}{\sqrt{2\pi s_z^2}} \exp\left(-\frac{(R_{ijz} - \mu_{ij})^2}{2s_z^2}\right). \tag{7.1}$$

As in the original AddiVortes model, we assume a conjugate prior distribution for the cell output values, where $\pi(\mu_{ij}) \sim \mathcal{N}(0, \sigma_\mu^2)$. For a given cell i in tessellation j, the full conditional distribution for its output value, μ_{ij}, is proportional to the likelihood of the partial residuals in that cell multiplied by the prior for μ_{ij}, that is,

$$p(\mu_{ij} \mid T_j, \boldsymbol{s}^2, \boldsymbol{R}_j) \propto \pi(\boldsymbol{R}_{ij} \mid T_j, \mu_{ij}, \boldsymbol{s}^2)\pi(\mu_{ij} \mid T_j).$$

If we let Z_{ij} be the set of indices for the data points that fall into cell i in tessellation T_j, then given the normal likelihood (with heteroscedastic variances s_z^2) and the normal prior, this becomes

$$p(\mu_{ij} \mid \cdot) \propto \exp\left[-\sum_{z \in Z_{ij}} \frac{(R_{ijz} - \mu_{ij})^2}{2s_z^2}\right] \exp\left[-\frac{\mu_{ij}^2}{2\sigma_\mu^2}\right].$$

To determine the form of this distribution, we combine the exponents and collect terms involving μ_{ij}, that is,

$$\begin{aligned} p(\mu_{ij} \mid \cdot) &\propto \exp\left[-\frac{1}{2}\left(\sum_{z \in Z_{ij}} \frac{R_{ijz}^2 - 2R_{ijz}\mu_{ij} + \mu_{ij}^2}{s_z^2} + \frac{\mu_{ij}^2}{\sigma_\mu^2}\right)\right] \\ &= \exp\left[-\frac{1}{2}\left(\mu_{ij}^2\left(\frac{1}{\sigma_\mu^2} + \sum_{z \in Z_{ij}} \frac{1}{s_z^2}\right) - 2\mu_{ij}\sum_{z \in Z_{ij}} \frac{R_{ijz}}{s_z^2} + \sum_{z \in Z_{ij}} \frac{R_{ijz}^2}{s_z^2}\right)\right] \\ &\propto \exp\left[-\frac{1}{2}\left(\mu_{ij}^2\left(\frac{1}{\sigma_\mu^2} + \sum_{z \in Z_{ij}} \frac{1}{s_z^2}\right) - 2\mu_{ij}\sum_{z \in Z_{ij}} \frac{R_{ijz}}{s_z^2}\right)\right]. \end{aligned}$$

Now, by completing the square and removing terms that don't contain μ_{ij}, we obtain

$$p(\mu_{ij} \mid \cdot) \propto \exp\left[-\frac{1}{2}\left(\frac{1}{\sigma_\mu^2} + \sum_{z \in Z_{ij}} \frac{1}{s_z^2}\right)\left(\mu_{ij}^2 - \left(\frac{1}{\sigma_\mu^2} + \sum_{z \in Z_{ij}} \frac{1}{s_z^2}\right)^{-1} \sum_{z \in Z_{ij}} \frac{R_{ijz}}{s_z^2}\right)^2\right],$$

and this is the kernel of a normal distribution.

Thus, we sample μ_{ij} from its full conditional posterior distribution, that is,

$$\mu_{ij} \mid \cdot \sim \mathcal{N}\left(\left(\frac{1}{\sigma_\mu^2} + \sum_{z \in Z_{ij}} \frac{1}{s_z^2}\right)^{-1} \sum_{z \in Z_{ij}} \frac{R_{ijz}}{s_z^2}, \left(\frac{1}{\sigma_\mu^2} + \sum_{z \in Z_{ij}} \frac{1}{s_z^2}\right)^{-1}\right).$$

To obtain the integrated likelihood used in the Metropolis-Hastings acceptance probability, we integrate the joint distribution with respect to μ_{ij}. By factoring out the terms that do not depend on μ_{ij}, we find that the marginal likelihood is proportional to the normalising constant of the posterior. After simplifying and ignoring terms that will cancel in the acceptance ratio, the integrated likelihood for cell i is,

$$\int p(\boldsymbol{R}_{ij} \mid \mu_{ij}) \pi(\mu_{ij})\, d\mu_{ij} \propto \left(\frac{1}{\sigma_\mu^2} + \sum_{z \in Z_{ij}} \frac{1}{s_z^2}\right)^{-\frac{1}{2}} \exp\left(\frac{1}{2} \frac{\left(\sum\limits_{z \in Z_{ij}} \frac{R_{ijz}}{s_z^2}\right)^2}{\frac{1}{\sigma_\mu^2} + \sum\limits_{z \in Z_{ij}} \frac{1}{s_z^2}}\right).$$

These forms are nearly identical to those in the original AddiVortes model, with the main difference being that sums over a constant variance $\frac{n_i}{\sigma^2}$ are replaced by sums of the individual inverse variances, $\sum\limits_{z \in Z_{ij}} \frac{1}{s_z^2}$.

7.1.2 Variance Model

We assume that

$$\boldsymbol{Y} \sim \mathcal{N}(f(\boldsymbol{x}), s^2(\boldsymbol{x})),$$

where $s^2(\boldsymbol{x}) = \prod_{l=1}^{m'} h(\boldsymbol{x}; T_l', M_l')$. Therefore, for any $i = 1, \ldots, n$ we have

$$\begin{aligned} p(\boldsymbol{Y} \mid f(\boldsymbol{x}_i), s^2(\boldsymbol{x}_i)) &\propto \frac{1}{\sqrt{2\pi s^2(\boldsymbol{x}_i)}} \exp\left(-\left(\frac{(y_i - f(\boldsymbol{x}_i))^2}{2s^2(\boldsymbol{x}_i)}\right)\right) \\ &\propto \frac{1}{\sqrt{2\pi s^2(\boldsymbol{x}_i)}} \exp\left(-\frac{\frac{(y_i - f(\boldsymbol{x}_i))^2}{s_{-l}^2(\boldsymbol{x}_i)}}{2s_{kl}^2}\right) \end{aligned}$$

where s_{kl} is the output value of tessellation T_l' for $\boldsymbol{x}_i$ for the variance model and $s_{-l}^2(\boldsymbol{x}_i) =$

$\prod_{q \neq l} h(\boldsymbol{x}_i; T'_q, M'_q)$.

The Heteroscedastic AddiVortes likelihood function for the variance component in the k^{th} cell of the l^{th} variance tessellation, if we let the set Z'_{kl} be the set of all observations that fall in the cell associated with s_{kl}, is given by

$$p(\boldsymbol{Y} \mid f(\boldsymbol{x}_i), s^2(\boldsymbol{x}_i)) = \prod_{z \in Z'_{kl}} \frac{1}{\sqrt{2\pi s_{kl}}} \exp\left(-\frac{e_z^2}{2s_{kl}^2}\right), \tag{7.2}$$

where

$$e_i^2 = \frac{\left(y_i - \sum_{j=1}^{m} g(\boldsymbol{x}_i; T_j, \boldsymbol{M}_j)\right)^2}{s_{-l}^2(\boldsymbol{x}_i)}.$$

A conjugate prior distribution is specified for the variance component,

$$s_{kl}^2 \sim \chi^{-2}(\nu', \lambda'),$$

for all l and k. The full conditional distribution for s_{kl}^2 then becomes

$$\begin{aligned} p(s_{kl}^2 \mid \boldsymbol{Y}, \boldsymbol{x}, T', M') &\propto p(Y \mid f(\boldsymbol{x}), s^2(\boldsymbol{x}))p(s_{kl}^2) \\ &\propto \prod_{z \in Z'_{kl}} \frac{1}{\sqrt{2\pi s_{kl}}} \exp\left(-\frac{e_z^2}{2s_{kl}^2}\right) \frac{\exp\left(\frac{-\nu'\lambda'}{2s_{kl}^2}\right)}{2s_{kl}^2} \\ &\propto \frac{1}{(s_{kl}^2)^{1+\frac{\nu'+|Z'_{kl}|}{2}}} \exp\left(-\frac{\nu'\lambda' + \sum_{z \in Z'_{kl}} e_z^2}{2s_{kl}^2}\right) \end{aligned} \tag{7.3}$$

Hence,

$$s_{kl}^2|\cdot \sim \chi^{-2}\left(\nu' + |Z'_{kl}|, \frac{\nu'\lambda' + \sum_{z \in Z'_{kl}} e_i^2}{\nu' + n}\right).$$

This allows for efficient updates of the cells in the m' variance component tessellations within the product decomposition using Gibbs sampling.

The integrated likelihood for s_{kl}^2 is also available in closed form as

$$\int L(s_{kl}^2 \mid \cdot)\pi(s_{kl}^2)ds_{kl}^2 = \frac{\Gamma\left(\frac{\nu'+n}{2}\right)\left(\frac{\nu'\lambda'}{2}\right)^{\nu'/2}}{(2\pi)^{n/2}\left(\nu'\lambda' + \sum\limits_{z\in Z'_{kl}} e_i^2\right)^{\frac{\nu'+n}{2}}}. \tag{7.4}$$

This form depends on the data through the sufficient statistic $\sum\limits_{z\in Z'_{kl}} e_i^2$, enabling the use of Metropolis–Hastings steps to explore the structure of the variance tessellations $T'_1, \ldots, T'_{m'}$ in a manner analogous to the mean model tessellations. That is, the conjugate inverse chi-squared prior leads to a Gibbs step drawing from an inverse chi-squared full conditional when updating the components of M' using Equation 7.3. Sampling the tessellation structure T'_j is performed via Metropolis–Hastings steps using the marginal likelihood given in Equation (7.4). This is possible because the integrated likelihood is analytically tractable under the specified heteroscedastic model.

The interpretation of this model follows from the mean being factored into a sum of weakly informative components (as usual in AddiVortes), while the variance is factored into the product of weakly informative components.

Note that the number of variance component tessellations, m', need not equal the number of mean component tessellations, m. A default value that has worked well in the examples explored is $m' = 40$. Since the tessellations making up the mean model are different from those making up the variance model, the number of cells in the l^{th} variance component tessellation is unrelated to the number of cells in the j^{th} mean component tessellation. This means, for instance, that the complexity of the variance function may differ from that of the mean function, and the predictors that are important for the variance function may also differ from those that are important for the mean.

7.1.3 Calibrating the variance prior

This section demonstrates a straightforward strategy for selecting prior parameters that yield reasonable, practical results.

As outlined in Section 7.1.2, the prior for the variance is given by $s^2(\boldsymbol{x}) \sim \prod_{l=1}^{m'} s_l^2$, with $s_l^2 \sim \chi^{-2}(\nu', \lambda')$, where the s_l^2 are independent and identically distributed. To determine the prior parameters (ν', λ') for the variance components, we begin by considering the prior for constant variance in the context of the homoscedastic AddiVortes model $Y = f(x) + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2)$, where

$\sigma^2 \sim \chi^{-2}(\nu, \lambda)$. Stone and Gosling (2025a) proposed strategies for selecting (ν, λ) in this scenario. To ensure consistency between the priors in the heteroscedastic and homoscedastic models, we set the prior means of the variance components in a consistant way. We have that

$$\mathbb{E}[\sigma^2] = \frac{\nu\lambda}{\nu - 2},$$

and

$$\mathbb{E}\left[s^2(\boldsymbol{x})\right] = \prod_{l=1}^{m'} E\left[s_l^2\right] = \lambda^{m'} \left(\frac{\nu'}{\nu' - 2}\right)^{m'}.$$

By equating these means, we ensure compatibility between the heteroscedastic and homoscedastic priors. To achieve this, we separately match the "λ component" and the "ν component", leading to the following expressions for the parameters of the heteroscedastic prior,

$$\lambda' = \lambda^{\frac{1}{m'}}, \quad \text{and} \quad \nu' = \frac{2}{1 - \left(1 - \frac{2}{\nu}\right)^{\frac{1}{m'}}}.$$

7.1.4 Model Evaluation Metrics

Assessing model performance requires more than evaluating predictive accuracy alone, particularly when modelling both the mean and variance of the response. Traditional metrics such as the RMSE offer a straightforward measure of prediction error but may overlook crucial aspects of distributional fit. For models that aim to capture the full predictive distribution, it is equally important to evaluate properties such as interval coverage and the precision of uncertainty estimates.

To address these considerations, we introduce graphical diagnostics and distributional metrics. The *H-evidence plot* is a graph of $\hat{s}(\boldsymbol{x}_i)$ versus the estimated posterior intervals of $\hat{s}(\boldsymbol{x}_i)$ for $i = 1, \ldots, n$, where n is the number of observations. This plot offers an intuitive way to detect patterns in variance that depend on the predictors, highlighting areas where heteroscedasticity is evident.

Another key diagnostic is the *predictive quantile-quantile (QQ) plot*, which extends classical QQ plots to compare the empirical quantiles of observed data against those predicted by the model's full predictive distribution $Y \mid \boldsymbol{x}$. If the model captures the data distribution well, the plotted points should approximate a straight line. Deviations from linearity indicate potential misfits, such as underestimating or overestimating variability.

In addition to graphical tools, we employ the *e-statistic*, a measure designed to quantify the similarity between two distributions F_1 and F_2 based on their respective random samples $\{U_i\}$ and $\{V_j\}$. This statistic provides a single scalar summary that captures both location and scale discrepancies. It is defined as

$$e = \frac{2}{n_1 n_2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} ||U_i - V_j|| - \frac{1}{n_1^2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_1} ||U_i - U_j|| - \frac{1}{n_2^2} \sum_{i=1}^{n_2} \sum_{j=1}^{n_2} ||V_i - V_j||, \tag{7.5}$$

where $||\cdot||$ denotes the Euclidean norm, and n_1 and n_2 are the sample sizes of $\{U_i\}$ and $\{V_j\}$, respectively.

The *e-statistic* works by comparing the pairwise distances among points in the two sample sets. The first term in the equation calculates the average pairwise distance between samples in F_1 and F_2,

$$\frac{2}{n_1 n_2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} ||U_i - V_j||.$$

This term quantifies the distance between the two sets of samples in Euclidean space. If the two distributions are similar, the distances between samples from F_1 and F_2 will tend to be small.

The second and third terms measure the pairwise distances within each sample set. Specifically,

$$\frac{1}{n_1^2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_1} ||U_i - U_j|| \quad \text{and} \quad \frac{1}{n_2^2} \sum_{i=1}^{n_2} \sum_{j=1}^{n_2} ||V_i - V_j||.$$

These terms reflect the spread or dispersion within the two distributions. By subtracting the internal distances from the between-sample distances, the e-statistic provides a measure of the dissimilarity between F_1 and F_2. A smaller value of e indicates closer agreement between the two distributions, while a larger value reflects greater discrepancy.

When comparing the *uniform distribution* to the *quantiles* of the sampled posterior distributions, the e-statistic can evaluate how closely the quantiles align with the uniform distribution. In this case, the uniform distribution on $[0, 1]$ serves as the reference, representing the ideal case in which data points are evenly distributed across the interval. The quantiles, derived from the empirical distribution of the observed data, form the second set of samples.

The e-statistic serves as a rigorous measure for comparing two distributions. When applied

to uniform distributions and empirical quantiles, it provides a clear assessment of how evenly the quantiles are spread, making it particularly useful for evaluating model calibration and fit.

For hyperparameter tuning, such as selecting the smoothing parameter κ in the prior mean model, cross-validation using the e-statistic ensures that both first- and second-moment properties are matched. This approach avoids relying solely on RMSE, which can be insufficient when modelling heteroscedasticity.

Overall, this combination of graphical diagnostics and quantitative metrics provides a comprehensive framework for evaluating and comparing the performance of the heteroscedastic AddiVortes model, ensuring that both predictive accuracy and uncertainty estimation are thoroughly assessed.

7.2 Illustrating heteroscedastic AddiVortes on simulated data

We use simulated data to evaluate the performance of AddiVortes against known values and consider a similar set-up to the one originally explored in Friedman (1991). We generate data by simulating the values of $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$, with $x_1, x_2, \ldots, x_d$ being independently drawn from the standard uniform distribution and the response variable Y given by

$$Y = f(\boldsymbol{x}) + \epsilon = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \epsilon, \tag{7.6}$$

where $\epsilon \sim \mathcal{N}(0, s^2(\boldsymbol{x}))$. Notably, Y is solely dependent on $x_1, \ldots, x_5$, so the predictors $x_6, \ldots, x_d$ are inactive variables. Similar to previous examples, this introduction of these extraneous variables injects a layer of complexity. In our simulated experiments, we set $d = 10$, that is, there are 10 covariates, of which only 5 are active.

To evaluate the predictive ability of the heteroscedastic AddiVortes model of the variance component $s^2(\boldsymbol{x})$, we consider three cases,

1. $s(\boldsymbol{x}) = 1$,
2. $s(\boldsymbol{x}) = 5x_1 + 2x_2$,
3. $s(\boldsymbol{x}) = \frac{1}{2}(10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 20x_4 + 5x_5)$.

The first case corresponds to the homoscedastic scenario, where the noise variance remains

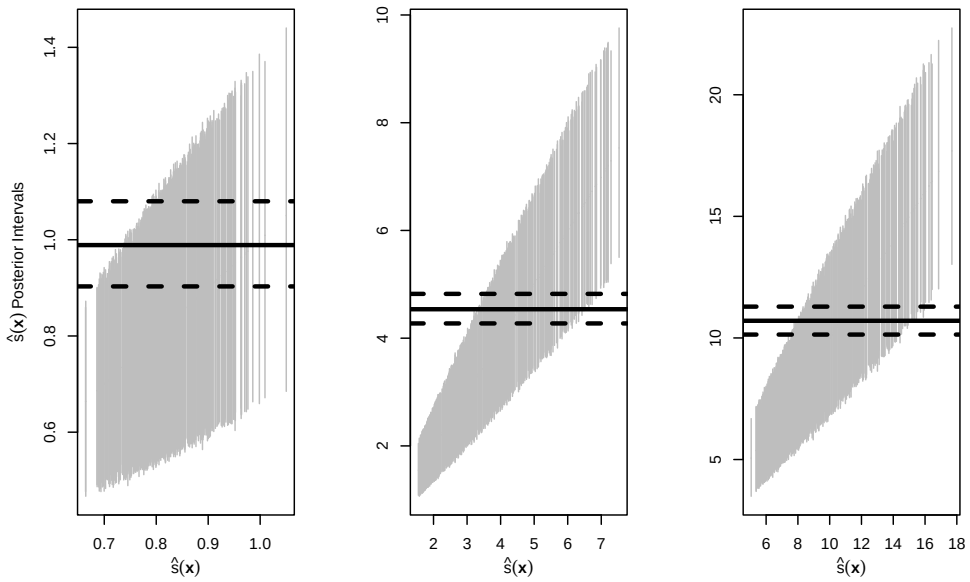


Figure 7.1: H-evidence plot for Heteroscedastic AddiVortes for case 1 (left), case 2 (middle) and case 3 (right). The solid horizontal line represents the estimate of σ obtained from the homoscedastic AddiVortes fit, with the posterior 90% interval depicted as dashed lines.

constant across the data. In contrast, the second and third cases involve heteroscedasticity, with non-constant noise variance. In the second case, the variance follows a simple linear relationship, while in the third case, it is determined by a more complex functional form.

Figure 7.1 displays the H-evidence plot for each of the cases. Observations are sorted based on the values of $\hat{s}(\boldsymbol{x})$, with $\hat{s}(\boldsymbol{x})$ plotted along the horizontal axis and the posterior intervals for $s(\boldsymbol{x})$ shown on the vertical axis. A solid horizontal line is drawn at the estimate of σ obtained from the homoscedastic version of AddiVortes. In the homoscedastic case (case 1), all the intervals overlap, suggesting that the data is indeed not heteroscedastic. However, in the other cases, the separation of the posterior intervals from the horizontal line clearly indicates that heteroscedasticity is "significant". This plot allows us to assess the practical significance of deviations from constant variance and has proven useful across a range of applications.

Since the data in this study is simulated, the true standard deviation function $s(\boldsymbol{x})$ is known for the testing data. Figure 7.2 compares the actual standard deviation $s(\boldsymbol{x})$ to the predicted standard deviation $\hat{s}(\boldsymbol{x})$. The heteroscedastic AddiVortes model demonstrates excellent predictive accuracy for the standard deviation in both cases, despite being trained on a limited number of observations.

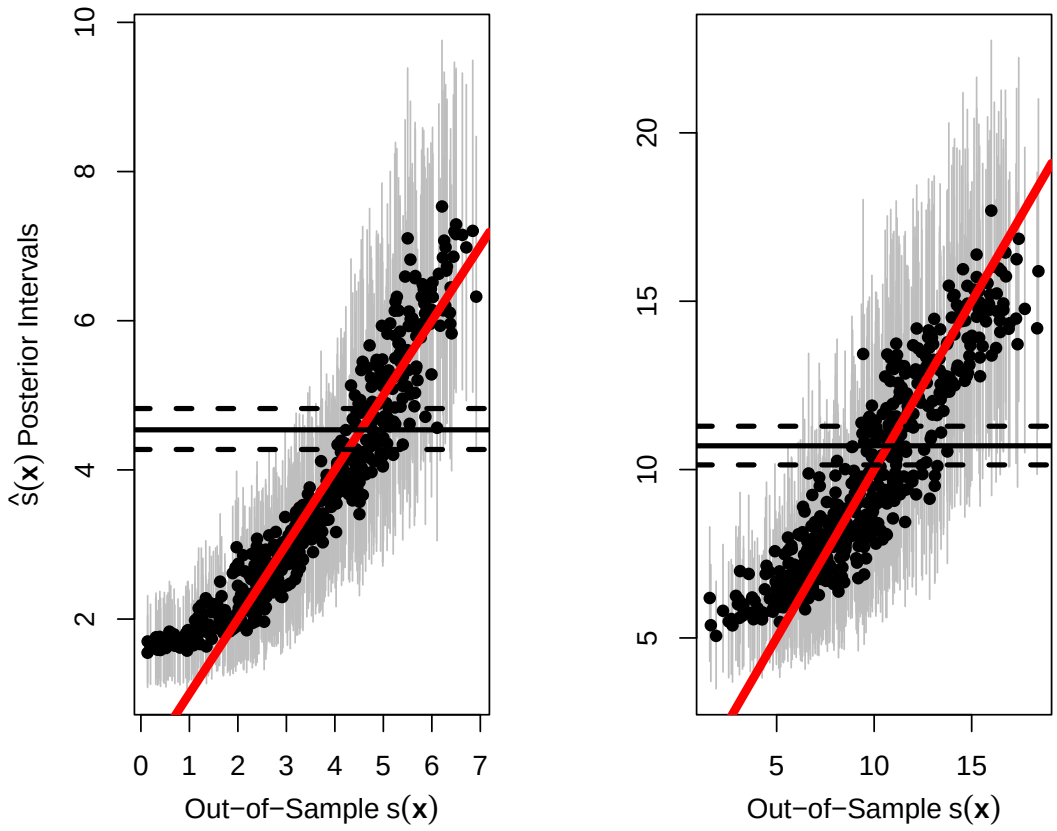


Figure 7.2: Comparison of true variance $s(x)$ and predicted variance $\hat{s}(x)$ when using the heteroscedastic AddiVortes model for case 2 (left) and case 3 (right).

Notably, the second case exhibits larger confidence intervals, reflecting increased uncertainty in the predictions, while in the first case, as the standard deviation decreases, the model shows a slight tendency to overestimate the standard deviation.

Furthermore, Figure 7.3 shows the relationship between the true mean function $f(\boldsymbol{x})$ and the predicted mean function $\hat{f}(\boldsymbol{x})$. The close alignment of the predicted and actual values indicates that the heteroscedastic AddiVortes model effectively captures the mean response, demonstrating strong predictive capabilities for both the mean and the standard deviation components of the data.

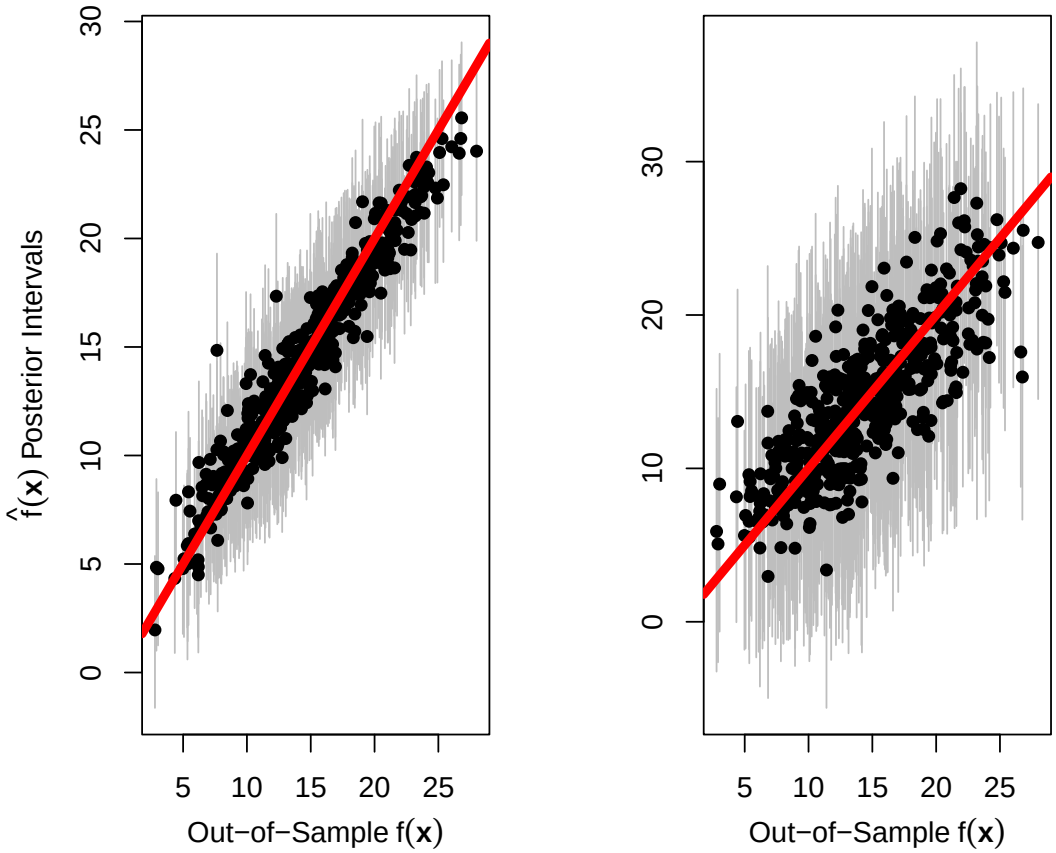


Figure 7.3: Comparison of the true mean function $f(x)$ and predicted mean function $\hat{f}(x)$ when using the heteroscedastic AddiVortes model for case 2 (left) and case 3 (right).

7.3 Cars dataset

The heteroscedastic AddiVortes model was applied to a dataset of used-car sales, comprising 1,000 observations collected from 1994 to 2013. This dataset, which spans the 2007–2008 financial crisis and subsequent recovery, includes both continuous predictors, mileage (in miles) and year of manufacture, and four categorical predictors: trim, colour, displacement, and ownership status. After encoding the categorical variables using one-hot encoding described in Section 2.2.1, the dataset contained 14 predictors.

The goal was to explore the relationship between car prices and the predictors, with particular focus on potential heteroscedasticity in price variability. The heteroscedastic AddiVortes model

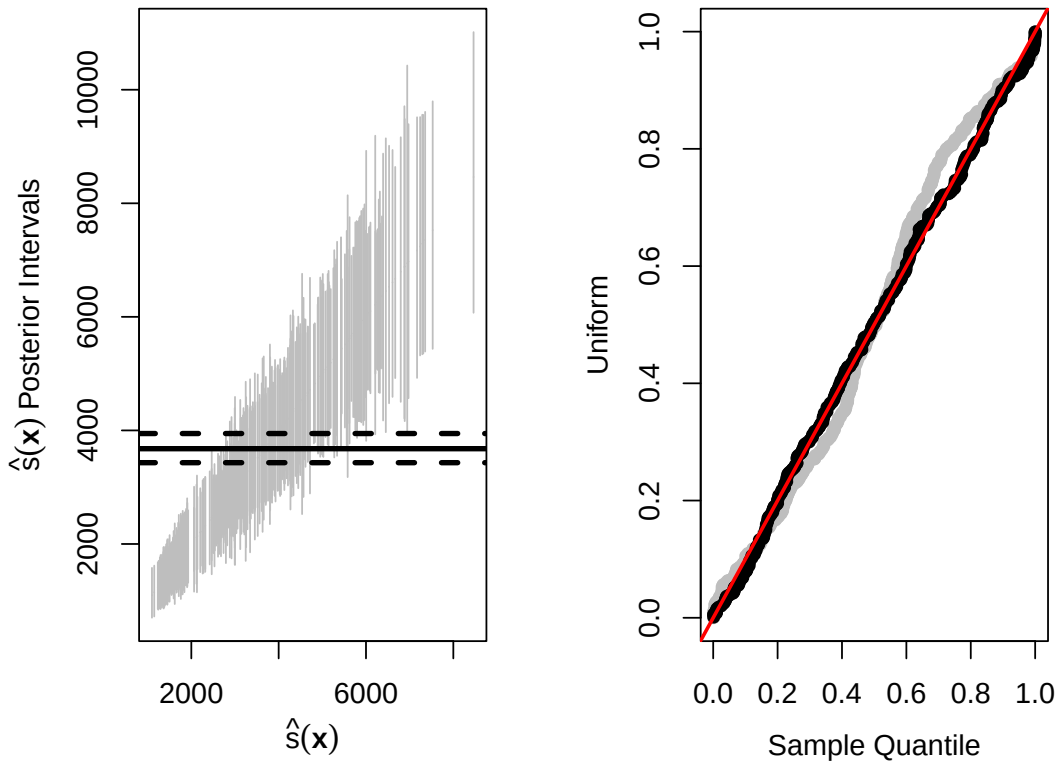


Figure 7.4: H-evidence plot (left) and Predictive qq-plot (right) for Heteroscedastic AddiVortes.

captured changes in variance that could not be explained by mean trends alone.

To tune the heteroscedastic AddiVortes model, cross-validation was performed using the e-statistic, ensuring that both mean and variance structures were accurately captured. This approach revealed that optimal settings for the hyperparameter κ differ between the heteroscedastic and standard AddiVortes models, with the former favouring greater smoothing to account for variance heterogeneity. The resulting analysis provided an understanding of the predictors driving both the expected value and the variability of car prices, highlighting heteroscedastic AddiVortes's ability to uncover meaningful, data-driven insights in regression problems.

The corresponding predictive out-of-sample qq-plots for the homoscedastic model trained on 80% of the full dataset with $\kappa = 0.75$ and the heteroscedastic model with $\kappa = 1.5$ are shown in Figure 7.4. This provides empirical graphical validation of the higher-level prior smoothing

suggested by the cross-validated tuning of κ. The qq-plot for the heteroscedastic model is closer to a straight line than for the homoscedastic version, suggesting the target distribution is better captured by heteroscedastic AddiVortes.

The H-evidence plot in Figure 7.4 serves as an additional check for evidence of heteroscedasticity: clearly, the posterior 90% credible intervals do not cover the estimate of standard deviation from the homoscedastic model for a majority of the data. Even though there is considerable uncertainty about $s(\boldsymbol{x})$ with larger values, we have strong evidence that some observations have $s(\boldsymbol{x}) > 7000$ and some observations have $s(\boldsymbol{x}) < 2500$. These differences are practically significant.

Table 7.1: A table showing the cross-validation values for competing methods.

Method	**Parameters**	**Values considered**
Homo-AV	Sigma prior: (ν, q)	(3, 0.90), (3, 0.99), (10, 0.75)
	μ prior: k value for σ_μ	0.5, 1, 5
	Poisson rate for # centres: λ_c	5, 25
Hetero-AV	Sigma prior: (ν, q)	(3, 0.90), (3, 0.99), (10, 0.75)
	μ prior: k value for σ_μ	0.5, 1, 5
	Poisson rate for # centres: λ_c	5, 25
BART	Sigma prior: (ν, q)	(3, 0.90), (3, 0.99), (10, 0.75)
	μ prior: k value for σ_μ	0.25, 0.5, 1, 2, 5, 10
HBART	Sigma prior: (ν, q)	(3, 0.90), (3, 0.99), (10, 0.75)
	μ prior: k value for σ_μ	0.25, 0.5, 1, 2, 5, 10

Now, we compare the performance of heteroscedastic AddiVortes to HBART (Pratola et al. (2020), implemented using `hbart` package), BART (Chipman et al. (2010), implemented using BayesTree package) and the homoscedastic AddiVortes model. To compare the performance of heteroscedastic AddiVortes to homoscedastic AddiVortes and the equivalent BART models, we used cross-validation on the e-statistic using values in Table 7.1, with all other hyperparameters taking their default value. 20 random train/test samples, 80%/20% split, and boxplots of the e-statistic and RMSE values for each method across all the splits are depicted in Figure 7.5 and the (50%, 75%) RRMSE quantiles are provided in Table 7.2.

The box plots reveal that the RMSE values for all methods are comparable, indicating similar levels of accuracy in point predictions. However, heteroscedastic AddiVortes outperforms competitors on the e-statistic, which measures the similarity between the predicted and true distributions. This substantial improvement in the e-statistic suggests that AddiVortes is not only capturing the data's central tendency but also providing a better overall representation of the predictive distribu-

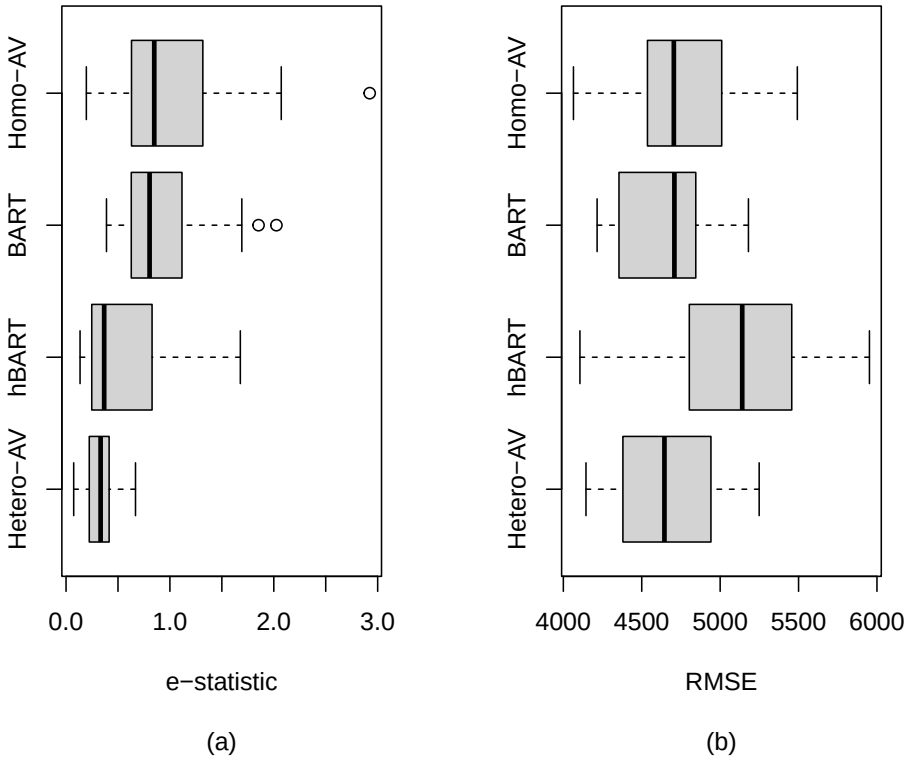


Figure 7.5: Box plot of e-statistic (left) and RMSE values (right) for all competing methods of the 20 train/test splits of the car dataset.

tion, particularly in scenarios with non-constant variance. This demonstrates its ability to model uncertainty and variability more effectively, offering a significant advantage in applications where understanding the full distribution is critical.

Note that, capturing the conditional distribution better does not guarantee that the prediction is better, as seen in the slightly worse performance in hBART despite having a much better e-statistic. This worse performance in RMSE compared to standard BART values could be down to differences is the packages used to implement the algorithms.

7.4 Million song dataset

To evaluate the heteroscedastic AddiVortes model in a more complex setting, we applied it to the year-prediction task in the Million Song Dataset (MSD). This dataset comprises 515,345 observa-

Table 7.2: (50%, 75%) quantiles of e-statistics and RMSE values for each method.

Method	e-statistic **(50%, 75%)**	RMSE **(50%, 75%)**
Heteroscedastic AddiVortes	(0.3223, 0.4095)	(4644.7, 4926.2)
Homoscedastic AddiVortes	(0.8492, 1.2699)	(4704.7, 5006.8)
HBART	(0.3670, 0.8278)	(5140.7, 5390.0)
BART	(0.8493, 1.2699)	(4693.7, 4842.8)

tions spanning 28,223 artists from 1922 to 2011, each with 90 predictor variables encoding audio features, including timbre averages (12 predictors) and timbre covariances (78 predictors). Predicting a song's release year from its audio features poses a challenging regression problem in a large n, p setting, making it an excellent test case for our model.

Previous studies, including those by Bertin-Mahieux et al. (2011), explored prediction models for this dataset, such as k-nearest neighbours and the Vowpal Wabbit algorithm. We compared standard main-effects linear regression, AddiVortes, and heteroscedastic AddiVortes models, using default priors for the Bayesian approaches. Table 7.3 summarises the Root Mean Squared Error (RMSE) for year prediction on the test set, showing that the linear and k-NN models performed worse than the AddiVortes and heteroscedastic AddiVortes models. Notably, both AddiVortes and heteroscedastic AddiVortes demonstrated competitive predictive accuracy, with the latter excelling due to its ability to model heteroscedasticity.

Table 7.3: The RMSE values of all the competing methods for the test set of the million song dataset.

Model	Hetero-AV	Homo-AV	HBART	Linear Regression	1-NN	50-NN	Vowpal Wabbit
RMSE	8.97	9.10	9.08	9.51	13.99	10.20	8.76

Convergence diagnostics, including trace plots of the variance $s^2(\boldsymbol{x})$ and posterior samples of σ, confirmed reasonable convergence for both models. The heteroscedastic AddiVortes model revealed substantial heteroscedasticity within the dataset, as evidenced by the posterior distribution of $s^2(\boldsymbol{x})$, which ranged from 1.21 to 31.4, compared to a constant posterior mean of 10.3 for σ in the AddiVortes model. The dataset's inherent imbalance, with most songs released post-1980 and relatively few observations from the early decades, does not imply a different variance process, but the sparsity leads to greater uncertainty.

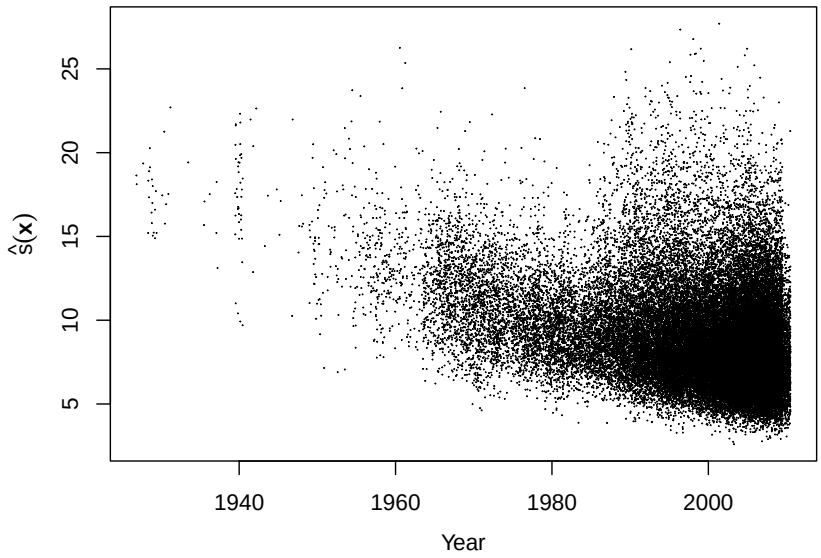


Figure 7.6: Plot of $\hat{s}(\boldsymbol{x})$ versus year for the million song test set. Note that the values of year have been perturbed to more clearly convey how sample size varies with year.

An analysis of the posterior average standard deviation $\hat{s}(\boldsymbol{x})$ indexed by year showed that high predictive uncertainty is not solely due to sparse data in early years; heteroscedastic AddiVortes identified regions of high uncertainty even in recent years with abundant data. This highlights the model's ability to differentiate between uncertainty driven by data sparsity and intrinsic complexity in the predictor-response relationship. Thus, heteroscedastic AddiVortes serves as a valuable tool for exploring data patterns, pinpointing areas where predictive uncertainty is inherent and guiding further model refinement.

7.5 Discussion

This chapter introduced the heteroscedastic AddiVortes model, an extension of the Bayesian additive Voronoi tessellations framework, designed to simultaneously model both the conditional mean and variance of a response variable. By using a "sum-of-tessellations" structure for the mean and a "product-of-tessellations" structure for the variance, the model effectively isolates signal from noise in datasets where the error variance depends on the covariates. This dual-ensemble approach

allows for more precise uncertainty quantification and improved interval coverage compared to homoscedastic models, which often gives biased predictive intervals in the presence of non-constant variance. Furthermore, the decoupling of the mean and variance components through an iterative backfitting algorithm ensures that the model can identify heteroscedastic patterns without compromising the accuracy of the primary regression surface.

The heteroscedastic AddiVortes model's key innovation is its ability to account for predictor-dependent variability, providing richer inferences than traditional homoscedastic models. The additive representation of the mean function enables intuitive modelling of main effects and interactions, while the multiplicative variance structure allows for precise, localised adjustments to the dispersion that reflect changes in scale driven by specific predictors. This dual-layer approach ensures that both central tendency and variability are accurately modelled, offering a comprehensive view of the predictive distribution.

Empirical results demonstrate the model's effectiveness in capturing data patterns across diverse applications, ranging from simulated examples to real-world datasets. The ability to adaptively identify heteroscedasticity and to provide interpretable insights into the sources of variation makes the heteroscedastic AddiVortes model a powerful tool for data exploration and prediction. Moreover, the use of conjugate priors and efficient MCMC sampling facilitates practical implementation, even in high-dimensional settings.

Chapter 8

Conclusion

BART (Chipman et al., 2010) is a powerful regression method that has high predictive accuracy and is able to naturally allow for uncertainty quantification due to Bayesian foundations. It can be effectively adapted and is a strong competitor amongst other "black-box" models in a variety of statistical settings. However, BART primarily relies on decision trees that produce axis-aligned partitions within the covariate space. This rigid framework often struggles to model complex interactions between variables efficiently. Findings in Bertsimas and Dunn (2017) indicate that oblique decision trees can outperform standard classification and regression trees by roughly 7%, while Breiman (2001) noted that an oblique variation of Random Forests has been shown to enhance performance by as much as 8.5%. To address these limitations, recent advancements such as obliqueBART (Nguyen et al., 2024) have been introduced, which refine the partitioning process by incorporating oblique decision trees to better capture the underlying relationships between covariates.

While these improved methods successfully enhance the predictive performance of BART, the work presented in Chapter 3 demonstrates that Voronoi tessellations and their variants offer a far more natural approach to partitioning a space. Voronoi tessellations have become increasingly relevant in predictive modelling, as demonstrated by Pope et al. (2021) and described in Section 3.3, who use them to represent discontinuities via Gaussian processes. Villagran et al. (2016) introduced a non-parametric sampling method that partitions the sample space into constant-density Voronoi regions, enabling efficient approximation of complex distributions without a predefined functional form. However, such methods frequently encounter significant computational challenges because

they tend to incorporate all covariates simultaneously when proposing a tessellation.

8.1 Innovations in this thesis

By adapting the powerful MCMC Bayesian backfitting algorithm that serves as the foundation for BART, we introduced AddiVortes (Stone and Gosling, 2025a), which integrates Voronoi tessellations to facilitate a more flexible partitioning of the covariate space. The model employs regularisation properties similar to those found in BART, which have proven successful in mitigating overfitting. The success of this approach lies in the combination of low-dimensional tessellations that maintain computational efficiency with an additive structure capable of recovering global nonlinearities.

The empirical evaluation of AddiVortes consistently demonstrated its superiority over established methods. In benchmark regression tasks, AddiVortes achieved a consistently lower RMSE compared to alternative models, including BART and other "black-box" methods, illustrating the benefits of replacing rigid axis-aligned partitions with flexible Voronoi cells. Through the use of synthetic datasets, we showed that AddiVortes is able to capture the posterior distribution of the underlying regression function effectively, thereby highlighting the theoretical advantages of the algorithm in a practical setting.

One problem with AddiVortes and other predictive "black-box" models is the inclusion of nominal covariates. Since most models require numerical inputs, nominal covariates must be transformed into a numerical format, yet their lack of a natural ordering makes it difficult to encode them optimally for use in algorithms. To address this limitation, encoding techniques mentioned in Section 2.2.1 convert the nominal structure of the covariate into a numerical representation. In Chapter 5, we introduce permutation encoding that deals with nominal categorical covariates by dynamically permutating the encoding of the covariate within the MCMC algorithm and tries to learn the best ordering. In real-world benchmark studies, the permutation encoding technique for AddiVortes provided a significant improvement in predictive accuracy compared to standard one-hot encoding or predefined ordinal structures. This suggests that the method is able to recover meaningful metric structures within nominal feature spaces without requiring any prior user prespecification.

Although the original AddiVortes algorithm was designed to model continuous dependent variables, the research in Chapter 6 extends the framework through the implementation of a probit link function and a latent variable structure, enabling both binary and multinomial classification. In binary classification tasks, AddiVortes demonstrated competitive performance across synthetic and real-world datasets, consistently achieving superior accuracy when compared to other leading "black-box" models. A comprehensive analysis using the Home Mortgage Disclosure Act (HMDA) dataset further showcased the robust capabilities of the model in predicting loan approval status, where it maintained reliable calibration and competitive area under the curve (AUC) scores. Furthermore, in the context of multinomial classification, AddiVortes successfully captured the posterior distribution in synthetic studies, while results from the dry bean classification study confirmed the ability to model multi-class environments involving highly correlated features.

Finally, the Heteroscedastic AddiVortes model, which is introduced in Chapter 7, captures non-constant residual variance by employing a second ensemble of tessellations designed to independently sample the variance conditional on the covariates. Simulation studies demonstrated the model's capacity to accurately estimate both the conditional mean and the conditional variance functions concurrently. Applied to real-world examples, such as the analysis of used car prices, the heteroscedastic formulation provided more accurate uncertainty quantification by correctly identifying regions of the feature space associated with higher price volatility. This advancement is critical for applications requiring reliable confidence intervals, proving that the Voronoi framework can effectively model not only the point estimations but also the complex uncertainty structure of the response variable.

8.2 Future work

The successful development and application of AddiVortes across diverse modelling tasks establishes a robust foundation for the use of Voronoi tessellation in Bayesian ensemble methods. While this thesis has demonstrated the effectiveness of replacing traditional decision trees with Voronoi tessellations, the inherent flexibility of the proposed additive framework opens up numerous modifications for further methodological advancement and applications. In this section, we detail the potential research directions that will help improve the predictive accuracy of AddiVortes.

8.2.1 Smoothed AddiVortes

A significant challenge in using standard Voronoi tessellations for regression is the nature of the predictive surface they create. In the standard AddiVortes model, the partition is piecewise-constant, meaning the predicted value is uniform within each cell and changes abruptly at the cell boundaries. This "step-wise" jump is a direct consequence of the "winner-takes-all" hard assignment, where an observation's output is determined solely by its single nearest centre. An observation near a boundary, but just inside one cell, can give very different predicted values than an observation just on the other side of that boundary. This behaviour is often unrealistic for modelling functions that are, in reality, smooth. As discussed in Section 2.6, this issue is analogous to the one in hard decision trees, which is addressed in BART by using soft decision trees.

Here, we discuss two distinct but related methods that can be adapted for the AddiVortes model to mitigate this problem and produce a smoother predictive function.

The first method introduces a probabilistic smoothing mechanism, conceptually similar to the 'soft decision trees' used in SoftBART Linero and Yang (2018). Instead of an observation $\boldsymbol{x}$ belonging deterministically to a single 'winning' cell, it is assigned a probability of belonging to every cell i in the tessellation. This probability, $p_i(\boldsymbol{x})$, is a decreasing function of the distance from $\boldsymbol{x}$ to the cell's centre $\boldsymbol{c}_i$. A function such as the softmax,

$$p_i(\boldsymbol{x}) \propto \exp(-k \cdot d(\boldsymbol{x}, \boldsymbol{c}_i)),$$

can be used to ensure these probabilities are highest for the nearest centre and sum to one. A large k recovers the original hard tessellation, while a smaller k creates a softer transition between cells. To reduce the need to calculate distances to each centre, one can find only the t closest centres for each observation; usually, $t = 2$ or 3 is sufficient to achieve smoother boundaries.

The second method is a non-probabilistic but related approach based on kernel smoothing, similar to inverse distance weighting (IDW) (Wong, 2017). The final prediction is a weighted average of all cell output values, $g(\boldsymbol{x} \mid T_j, \boldsymbol{M}_j) = \sum_i w_i(\boldsymbol{x})\mu_{ij}$. The weights $w_i(\boldsymbol{x})$ are defined directly as a function of normalised inverse distance. For example, a weight

$$w_i(\boldsymbol{x}) = 1/d(\boldsymbol{x}, \boldsymbol{c}_i)^p$$

could be used, where p is a power parameter. These weights are then normalised to sum to one. This formulation also smooths the predictive surface, as the influence of any single centre's output value μ_{ij} wanes gradually with distance rather than dropping to zero at the boundary. The choice of the power parameter p dictates the steepness of this fall-off and thus the smoothness of the resulting function.

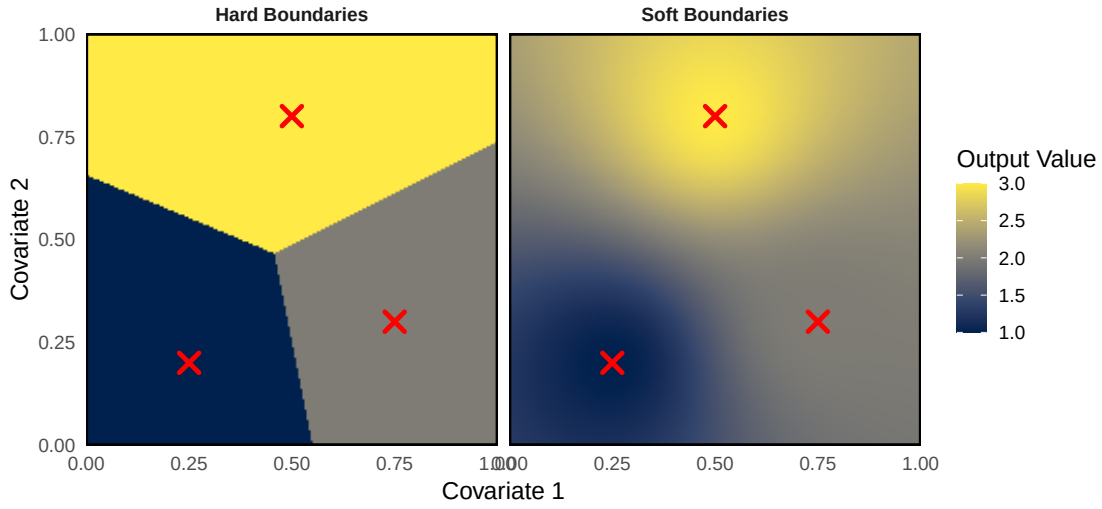


Figure 8.1: An illustration of a Voronoi tessellation with hard boundaries and output values {1,2,3}, and a Voronoi tessellation with soft boundaries using the inverse distance approach with $p = 2$.

Figure 8.1 provides a clear visual comparison between the standard tessellation model and the proposed smoothing adaptation. The left panel, "Hard Boundaries", illustrates the standard Voronoi tessellation. Here, the predictor space is partitioned into distinct cells with sharp, linear boundaries. All observations within a single cell are assigned the constant output value of that cell's centre, resulting in a piecewise-constant surface where the predicted value jumps abruptly at the cell borders. The right panel, "Soft Boundaries", demonstrates the effect of the inverse-distance weighting method with $p = 2$. In this version, the rigid cell boundaries are eliminated and this creates a smooth gradient across the space, effectively mitigating the unrealistic step-wise jumps

inherent in the standard model.

A drawback of these smoothing methods is the increase in computational overhead. The standard "hard" tessellation only requires finding the single nearest centre for each observation, a process that can be highly optimised using k-d trees. In contrast, the probabilistic approach increases the number of distance calculations by t, and the inverse distance weighting approach requires calculating the distance from every observation to every centre in the tessellation to compute the full set of weights. If the number of observations is high, this can significantly increase computational cost.

A more fundamental challenge lies in fully integrating these soft partitions into the AddiVortes MCMC framework. Currently, while the final prediction is "soft" and based on a weighted average, the full conditional distribution used to sample the μ_{ij} parameters is still derived as if the partitions were "hard". An approach would be to reformulate the full conditional for μ_{ij} to account for the probabilistic or weighted contributions from all observations, thereby better aligning the parameter estimation step with the final predictive model.

8.2.2 Fundamental partition changes

Most "black-box" statistical learning models implicitly assume a Euclidean feature space. However, many real-world applications involve data that reside on non-Euclidean manifolds. For instance, diffusion tensor imaging (DTI) data in neuroimaging are modelled on the manifold of symmetric positive definite matrices (Fillard et al., 2005); shape analysis in computer vision often requires statistical modelling on shape manifolds; directional data such as wind or ocean currents are naturally represented on the unit sphere; and pose estimation in robotics frequently involves data on Lie groups (Galceran and Carreras, 2013).

Modelling relationships in such settings requires methods that respect the underlying geometry of the data. For instance, environmental modelling often occurs on the curved surface of the Earth; neuroscience analyses data on cortical surfaces; robotics and social network data frequently benefit from hyperbolic or manifold-based embeddings. Some progress has been made in this direction, such as the extension of Gaussian processes to manifold settings (Borovitskiy et al., 2023), but the area remains underdeveloped.

A promising avenue for future research lies in relaxing this assumption by adopting alternative

distance metrics, such as the Manhattan (L_1) or Chebyshev (L_∞) norms. For example, using the Manhattan distance would alter the geometry of the cell boundaries to form regions defined by linear constraints similar to Lasso regularisation, potentially offering better robustness in high-dimensional settings where the contrast between Euclidean distances diminishes.

Furthermore, the flexibility of the tessellation can be significantly enhanced through the use of power diagrams, described in Section 3.4. Unlike standard Voronoi diagrams, which partition space based solely on the location of generator points, power diagrams introduce an additive weight to the distance function, defined as

$$d_{\text{power}}(\mathbf{x}, \mathbf{c}_k) =\mid \mathbf{x} - \mathbf{c}_k \mid^2 - w_k.$$

By placing a prior distribution on these cell weights, w_k, the model gains an additional degree of freedom to control the size and influence of each cell independently of its centre's location. This allows for a more dynamic allocation of probability mass, enabling the model to expand or contract specific regions based on the density of the data or the complexity of the response surface. Crucially, if a weight becomes sufficiently small relative to its neighbours, the corresponding cell can become empty, effectively providing a mechanism for adaptive pruning of the generators within the MCMC process itself.

Finally, the definition of the metric space need not be restricted to standard vector spaces. The additive nature of the Voronoi definition allows AddiVortes to be adapted to non-Euclidean domains, such as Riemannian manifolds or graph-based structures, simply by substituting the Euclidean norm with a geodesic or shortest-path distance. This would enable the application of the method to highly complex data types, such as phylogenetic trees or social network graphs, where the notion of linearity is ill-defined but a valid metric exists.

A simple method is using Voronoi partitions for spherical spaces after conversion through a stereographic projection. Let the unit n-sphere be embedded in $\mathbb{R}^{n+1}$ as

$$\mathbb{S}^n = \left\{ \boldsymbol{X} = (X_1, \ldots, X_{n+1}) \in \mathbb{R}^{n+1} :\mid \boldsymbol{X} \mid^2 = 1 \right\}.$$

Let $N = (0, 0, \ldots, 0, 1)$ denote the north pole of the sphere. The stereographic projection maps a

point $\boldsymbol{x} \in \mathbb{R}^n$ to a point on the sphere $\boldsymbol{X} \in \mathbb{S}^n \setminus \{N\}$ by projecting from the north pole onto the hyperplane $X_{n+1} = 0 \subset \mathbb{R}^{n+1}$. Given a point $\boldsymbol{x} \in \mathbb{R}^n$, the stereographic projection onto $\mathbb{S}^n$ is given by

$$\pi(\boldsymbol{x}) = \left(\frac{2\boldsymbol{x}}{|\ \boldsymbol{x}\ |^2 + 1}, \frac{|\ \boldsymbol{x}\ |^2 - 1}{|\ \boldsymbol{x}\ |^2 + 1} \right),$$

where $\pi(\boldsymbol{x}) \in \mathbb{R}^{n+1}$ lies on the unit sphere. This map is conformal and bijective between $\mathbb{R}^n$ and $\mathbb{S}^n \setminus \{N\}$.

The AddiVortes algorithm can operate within the spherical space by using the corresponding distance metric to define the Voronoi tessellation. As illustrated in Figure 3.3, this method produces a more natural partition of the Euclidean space by allowing for curved cell boundaries.

A framework could be developed for inference on general manifolds, modifying the existing AddiVortes MCMC method to accommodate more complex spaces. The objective would be to run the AddiVortes model on any space as long as there is a bijection to Euclidean space and a defined metric. Computational challenges will be mitigated by employing approximate geodesics, efficient spatial indexing and scalable MCMC techniques.

8.2.3 Other promising development areas

As shown in Chapter 2, BART is widely recognised for its ability to adapt to diverse statistical modelling settings. AddiVortes shares this versatility but arguably offers superior predictive power, as it partitions the covariate space more naturally to capture spatial interactions. However, despite these similarities, the underlying algorithms differ significantly; consequently, standard BART extensions require theoretical refinement to be effectively applied to AddiVortes. The following section outlines a few examples of such adaptations and proposes possible theoretical modifications for these specific settings.

Variable selection and sparsity-inducing priors

Analogous to the strategies applied to DART outlined in Section 2.6 and Linero (2016), variable selection and sparsity can be induced within the AddiVortes framework by modifying the prior on the tessellation structure. Recall that DART uses a Dirichlet prior to sample the splitting

probability vector $\boldsymbol{s}$ and the posterior samples of $\boldsymbol{s}$ are then drawn from the following distribution,

$$\boldsymbol{s} \mid \text{Trees} \sim \text{Dirichlet}\left(\frac{\alpha}{p} + k_1, \ldots, \frac{\alpha}{p} + k_p\right), \tag{8.1}$$

where α is the sparsity hyperparameter, p is the total number of variables, and k_j is the number of times variable j is currently used in the sum-of-trees model.

However, a fundamental methodological distinction between BART and AddiVortes lies in how covariates are utilised within a single learner. While BART permits a decision tree to split on the same covariate multiple times, AddiVortes typically restricts each covariate to contribute only a single dimension to the construction of a tessellation. To address this constraint and enable effective variable selection, we propose two potential extensions.

The first approach involves a straightforward algorithmic modification: relaxing the constraint to allow covariates to be selected multiple times within a single tessellation. This would necessitate only minor adjustments to the Metropolis-Hastings (MH) acceptance ratio to account for the non-uniform probability of covariate inclusion. In this case, when selecting a covariate as a dimension, all the covariates will be available for selection. Therefore, $\boldsymbol{s}$ can be a probability vector of choosing a covariate as a dimension in a tessellation and can be sampled from (8.1), but with k_j as the number of dimensions that the j^{th} covariate corresponds to in the ensemble.

However, a potential pitfall is that having covariates as more dimensions may not be an indication of importance in the model since including additional covariates of an important variable does not guarantee a positive effect. Consequently, an important variable may be underrepresented and its importance not captured correctly.

Beyond variable selection, this extension offers an interesting idea for the general regression tasks. Allowing covariates to appear multiple times within a single tessellation would enable the model to capture higher-order interactions at the level of the individual learner, rather than relying solely on the ensemble average. This modification would allow more flexibility in the geometric partitioning of AddiVortes, allowing the model to construct more complex decision boundaries and potentially improving convergence rates in scenarios with highly non-linear response functions.

The second approach employs the data augmentation technique described in the appendix of Linero (2016), while maintaining the restriction that each covariate appears at most once per

tessellation. In this setting, a data augmentation strategy is proposed that enables us to recover an update similar to (8.1). When drawing $\boldsymbol{s}$ from the prior to sample a covariate for the tessellation, we sample with replacement from $\boldsymbol{s}$ until an eligible covariate is selected. Had we observed this sequence of selections, rather than solely the final covariate chosen, we could apply (8.1), except that k_j represents the number of attempted inclusions of covariate j, rather than the count of actual inclusions.

The strategy, therefore, is to sample this latent history conditional on $\boldsymbol{s}$. For a given dimension, the full conditional for the total number of attempted covariate inclusions, W, follows a geometric distribution with success probability

$$1 - \sum_{j \notin \varepsilon_t} s_j,$$

where ε_t denotes the set of variables already included in tessellation t. Given W, each of the $W-1$ attempted covariate inclusions equals j with probability proportional to $s_j \mathbb{I}(j \notin \varepsilon_t)$. Once these quantities are sampled, we apply the update in (8.1), where the k_j values represent the number of attempted inclusions of covariate j rather than the number of actual inclusions.

A limitation is that a covariate's usage is strictly capped by the total number of tessellations, m, whereas in DART, there is no such cap. Due to regularisation, this ceiling may be too low to capture signals of importance for the covariates. Furthermore, if the sparsity prior is too aggressive, relevant variables may be excluded entirely rather than simply restricted in dimensionality.

Finally, the AddiVortes algorithm could be adapted using techniques such as permutation-based approaches (Bleich et al., 2014; Luo and Daniels, 2024) and clustering-based approaches (Ye and Li, 2025a), as these methods are less sensitive to the specific tessellation structure. However, these approaches typically rely on running the algorithm several times or using multiple ensembles of tessellations that massively increase computational costs but might not lead to much improvement.

Survival analysis

Given the predictive performance of AddiVortes within the binary classification setting, there is an indication that this method will function as a powerful predictive model within the survival analysis framework. This is because we can adopt an approach similar to that described in Section 2.6 and Sparapani et al. (2016), where the survival problem is essentially treated as a sequence of conditional

binary classification problems.

To adapt the AddiVortes model for survival analysis, the approach mirrors the strategy employed in BART, that is, fundamentally restructuring the data rather than altering the core tessellation algorithm. The primary obstacle in survival data, handling right-censored observations where the event time is unknown, prevents the direct application of standard additive regression. Consequently, the continuous time-to-event problem is recast as a discrete-time binary classification task. This involves discretising the time scale and expanding the dataset so that each subject is represented by multiple rows, corresponding to the time intervals they survived, with a binary outcome indicating whether the event occurred in that specific interval.

Once the data is transformed into this long binary format, the AddiVortes algorithm is applied using a probit link function to model the probability of the event occurring at a given time point, conditional on survival up to that point. In this framework, time is treated simply as an additional covariate within the input space. Unlike decision trees, which partition time in a stepwise, axis-aligned fashion, AddiVortes partitions the joint space of time and covariates using Voronoi tessellations. The algorithm can more naturally model non-linear interactions between time and predictors, effectively modelling non-proportional hazards.

Crucially, the output of this classification model, the discrete-time hazard $h(t \mid \boldsymbol{x})$, allows for the direct reconstruction of the survival function. For a given individual with covariates $\boldsymbol{x}$, the survival probability at time t is derived using the product limit estimator, calculated as

$$S(t \mid \boldsymbol{x}) = \prod_{j=1}^{t} (1 - h(j \mid \boldsymbol{x})).$$

A significant advantage of the Bayesian framework is that this calculation is performed for every posterior draw of the sum-of-tessellations. Consequently, the model yields a full posterior distribution for the survival curve of every subject, enabling the straightforward quantification of uncertainty and the construction of pointwise credible intervals, which are often difficult to obtain in standard non-parametric survival models.

It is important to note that this data augmentation strategy significantly increases the computational burden, as the effective sample size scales with the number of time intervals. While decision trees handle this expansion relatively efficiently, the distance calculations required for Voronoi tes-

sellations may become computationally expensive on larger datasets. Implementation of sparse matrix techniques or down-sampling strategies for the non-event intervals would ensure the model remains scalable to high-dimensional survival data.

AddiVortes offers a robust framework for analysing recurrent events by moving beyond the restrictive linearity assumptions typical of traditional survival models. Drawing on the work of Sparapani et al. (2018), the method accommodates repeated, non-absorbing incidents without enforcing rigid dependencies between covariates and risk. This allows for a more accurate representation of how hazards evolve over time across multiple event occurrences, capturing complex non-linear patterns that standard approaches often fail to detect.

In the context of competing risks, the model provides a powerful alternative to proportional hazards frameworks by enabling the direct estimation of cumulative incidence functions. Adopting the non-parametric strategy outlined by Sparapani et al. (2020), AddiVortes naturally accounts for non-proportional hazards and intricate covariate interactions inherent in multi-event settings. This capability ensures that the estimated probabilities of specific outcomes remain reliable even when the dominant risk factors shift significantly over the observation period.

Causal inference

Given the strong predictive capabilities of AddiVortes in non-linear regression settings, there is a strong indication that the method will function as a powerful tool within the causal inference framework. This is because we can adopt an approach similar to that famously described by Hill (2011) for BART, where the estimation of causal effects is essentially treated as a missing data problem within a potential outcomes framework.

To adapt the AddiVortes model for causal inference, the approach mirrors the "response surface" strategy employed in standard BART: modelling the outcome conditional on both the covariates and the treatment assignment simultaneously. The primary obstacle in causal inference is the fundamental problem that for any given observation, we only observe the outcome associated with the treatment they actually received, leaving the counterfactual outcome unobserved. Consequently, the causal problem is recast as a prediction task. The treatment assignment variable, Z, is simply included as an additional covariate in the design matrix, allowing the model to learn the relationship between the treatment, the confounders, and the outcome.

Once the treatment indicator is integrated into the input space, the AddiVortes algorithm is applied to model the response surface $f(\boldsymbol{x}, z)$. Unlike decision trees, which capture treatment heterogeneity by splitting on Z in a stepwise fashion (effectively dividing the population into treated and control subgroups within rectangular regions), AddiVortes partitions the joint space of treatment and covariates using Voronoi tessellations. The model can better capture non-linear interactions between the treatment and the covariates without the need for manual specification of interaction terms, as the tessellation cells can adapt locally to regions where the treatment effect varies.

The output of this model allows for the direct reconstruction of the Conditional Average Treatment Effect (CATE). For a given individual with covariates $\boldsymbol{x}$, the estimated treatment effect $\tau(\boldsymbol{x})$ is derived by predicting the outcome under both treatment conditions ($z = 1$ and $z = 0$) and taking the difference:

$$\tau(\boldsymbol{x}) = f(\boldsymbol{x}, 1) - f(\boldsymbol{x}, 0).$$

A significant advantage of the Bayesian framework is that this subtraction is performed for every posterior draw of the sum-of-tessellations. Consequently, the model yields a full posterior distribution for the treatment effect of every subject, enabling the straightforward quantification of uncertainty and the construction of pointwise credible intervals for the causal effect, a critical feature for personalised decision-making.

It is important to note that while this "single-surface" strategy is easily implemented, it is susceptible to what is known as regularisation-induced confounding. As noted by Hahn et al. (2020), if the prognostic signal (the effect of $\boldsymbol{x}$ on Y) is strong and the treatment effect is weak, a unified Bayesian regularisation prior may accidentally shrink the treatment estimate to zero.

To address this, the AddiVortes framework could be adapted to mimic the Bayesian Causal Forest (BCF) approach proposed by Hahn et al. (2020). This approach avoids the risk of regularisation bias by structurally separating the prognostic term from the treatment effect. Instead of a single model, the outcome is modelled as

$$Y = \mu(\boldsymbol{x}) + \tau(\boldsymbol{x})Z + \epsilon,$$

where $\mu(\boldsymbol{x})$ and $\tau(\boldsymbol{x})$ are estimated by separate sums-of-tessellations (or separate AddiVortes en-

sembles) with independent priors. Adopting a similar decoupled strategy for AddiVortes would allow for distinct regularisation of the baseline response and the treatment heterogeneity, making it particularly suitable for observational studies where selection bias is a significant concern.

The application of Bayesian ensemble methods to causal inference is a rapidly evolving field that extends well beyond the foundational approaches discussed here. The versatility of the sum-of-trees framework has already facilitated advanced techniques for complex scenarios, such as instrumental variable analysis and sensitivity analysis. These ongoing innovations within the broader BART literature provide a clear roadmap for further developing the AddiVortes framework. Thus, the adaptation of Voronoi tessellations for causal estimation represents an active area of research with significant potential to incorporate these advanced strategies for more robust causal discovery.

Bibliography

Aitken, A. C. (1936). IV.—On Least Squares and Linear Combination of Observations. *Proceedings of the Royal Society of Edinburgh*, 55:42–48.

Albert, J. H. and Chib, S. (1993). Bayesian Analysis of Binary and Polychotomous Response Data. *Journal of the American Statistical Association*, 88(422):669–679.

Allison, E. (2024). Binary Classification of Medical Images by Symbolic Regression. In Naik, N., Jenkins, P., Grace, P., Yang, L., and Prajapat, S., editors, *Advances in Computational Intelligence Systems*, pages 516–527, Cham. Springer Nature Switzerland.

Alpers, A., Brieden, A., Gritzmann, P., Lyckegaard, A., and Poulsen, H. F. (2015). Generalized Balanced Power Diagrams for 3D Representations of Polycrystals. *Philosophical Magazine*, 95(9):1016–1028.

Arslan, O. and Koditschek, D. E. (2016). Exact Robot Navigation Using Power Diagrams. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8.

Aurenhammer, F. (1987). Power Diagrams: Properties, Algorithms and Applications. *SIAM Journal on Computing*, 16(1):78–96.

Bai, R., Moran, G. E., Antonelli, J. L., Chen, Y., and Boland, M. R. (2022). Spike-and-Slab Group Lassos for Grouped Regression and Sparse Generalized Additive Models. *Journal of the American Statistical Association*, 117(537):184–197.

Barbieri, M. M. and Berger, J. O. (2004). Optimal Predictive Model Selection. *The Annals of Statistics*, 32(3):870–897.

Bennett, K. (1992). Decision Tree Construction via Linear Programming. *Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference, Utica, Illinois.*

Bentley, J. L. (1975). Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, 18(9):509–517.

Berkson, J. and Gage, R. P. (1950). Calculation of Survival Rates for Cancer. *Proceedings of the staff meetings. Mayo Clinic*, 25 11:270–286.

Bertin-Mahieux, T., Ellis, D., Whitman, B., and Lamere, P. (2011). The Million Song Dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, pages 591–596.

Bertsimas, D. and Dunn, J. (2017). Optimal Classification Trees. *Machine Learning*, 106(7):1039–1082.

Beygelzimer, A., Kakadet, S., John Langford, S. A., Mount, D., and Li, S. (2023). Fast Nearest Neighbor Search Algorithms and Applications.

Bhattacharya, A., Pati, D., and Dunson, D. (2014). Anisotropic Function Estimation Using Multi-Bandwidth Gaussian Processes. *The Annals of Statistics*, 42(1):352–381.

Bhattacharya, A., Pati, D., Pillai, N. S., and Dunson, D. B. (2015). Dirichlet–Laplace Priors for Optimal Shrinkage. *Journal of the American Statistical Association*, 110(512):1479–1490. PMID: 27019543.

Binder, H. and Schumacher, M. (2008). Allowing for Mandatory Covariates in Boosting Estimation of Sparse High-Dimensional Survival Models. *BMC bioinformatics*, 9:14.

Bleich, J., Kapelner, A., George, E. I., and Jensen, S. T. (2014). Variable Selection for BART: An Application to Gene Regulation. *The Annals of Applied Statistics*, 8(3):1750–1781.

Bliss, C. I. (1934). The Method of Probits. *Science*, 79(2037):38–39.

Bock, M., Tyagi, A. K., Kreft, J.-U., and Alt, W. (2010). Generalized Voronoi Tessellation as a Model of Two-Dimensional Cell Tissue Dynamics. *Bulletin of Mathematical Biology*, 72(7):1696–1731.

Borgwardt, S. and Frongillo, R. M. (2019). Power Diagram Detection with Applications to Information Elicitation. *Journal of Optimization Theory and Applications*, 181(1).

Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth, M. P. (2023). Matérn Gaussian Processes on Riemannian Manifolds.

Bowyer, A. (1981). Computing Dirichlet Tessellations. *The Computer Journal*, 24(2):162–166.

Breiman, L. (1995). Better Subset Regression Using the Nonnegative Garrote. *Technometrics*, 37(4):373–384.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, 24:123–140.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth Statistics/Probability Series. Wadsworth Advanced Books and Software, Belmont, CA.

Bühlmann, P. and van de Geer, S. (2011). *Variable Selection with the Lasso*, pages 183–247. Springer Berlin Heidelberg, Berlin, Heidelberg.

Burgette, L. F. and Nordheim, E. V. (2012). The Trace Restriction: An Alternative Identification Strategy for the Bayesian Multinomial Probit Model. *Journal of Business & Economic Statistics*, 30(3):404–410.

Carnegie, N. B. (2019). Comment: Contributions of Model Features to BART Causal Inference Performance Using ACIC 2016 Competition Data. *Statistical Science*, 34(1):90–93.

Caron, A., Baio, G., and Manolopoulou, I. (2022). Shrinkage Bayesian Causal Forests for Heterogeneous Treatment Effects Estimation. *Journal of Computational and Graphical Statistics*, 31(4):1202–1214.

Carriero, A., Kapetanios, G., and Marcellino, M. (2015). Credit Risk Modelling with Bayesian Additive Regression Trees. *Journal of Empirical Finance*, 32:129–143.

Cattaneo, M. D., Chandak, R., and Klusowski, J. M. (2024). Convergence Rates of Oblique Regression Trees for Flexible Function Libraries. *The Annals of Statistics*, 52(2):678–703.

Chen, C.-H., Lai, J.-P., Chang, Y.-M., Lai, C.-J., and Pai, P.-F. (2023). A Study of Optimization in Deep Neural Networks for Regression. *Electronics*, 12(14).

Chen, T. and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM.

Chen, T., Lehr, J., Lavrova, O., and Martinez-Ramonz, M. (2016). Distribution-Level Peak Load Prediction Based on Bayesian Additive Regression Trees. In *2016 IEEE Power and Energy Society General Meeting (PESGM)*, pages 1–5.

Chipman, H. A., George, E. I., and McCulloch, R. E. (1998). Bayesian CART Model Search. *Journal of the American Statistical Association*, 93(443):935–948.

Chipman, H. A., George, E. I., and McCulloch, R. E. (2010). BART: Bayesian Additive Regression Trees. *The Annals of Applied Statistics*, 4(1):266–298.

Cortes, C. and Vapnik, V. N. (1995). Support-Vector Networks. *Machine Learning*, 20:273–297.

Cox, D. R. (1959). The Regression Analysis of Binary Sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 21(1):238–238.

Cox, D. R. (1972). Regression Models and Life-Tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202.

Cyree, K. and Winters, D. (2023). Investigating Bank Lending Discrimination in the US Using CRA-Rated Banks' HMDA Loan Data. *Public Choice*, 197:1–25.

Deevey, E. S. (1947). Life Tables for Natural Populations of Animals. *The Quarterly Review of Biology*, 22(4):283–314. PMID: 18921802.

Delaunay, B. (1934). Sur la sphère vide. A la mémoire de Georges Voronoi. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et naturelles*, 6:793–800.

Denison, D. G. T., Mallick, B. K., and Smith, A. F. M. (1998). A Bayesian CART Algorithm. *Biometrika*, 85(2):363–377.

Deshpande, S. K. (2025). flexBART: Flexible Bayesian Regression Trees with Categorical Predictors. *Journal of Computational and Graphical Statistics*, 34(3):1117–1126.

Dirichlet, G. L. (1850). Über die Reduction der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen. *Crelle's Journal*, 40:209–227.

Dorie, V., Hill, J., Shalit, U., Scott, M., and Cervone, D. (2019). Automated versus Do-It-Yourself Methods for Causal Inference: Lessons Learned from a Data Analysis Competition. *Statistical Science*, 34(1):43–68.

Efron, B. and Morris, C. (1977). Stein's Paradox in Statistics. *Scientific American*, 236:119–127.

Fejes Tóth, L. (1977). Illumination of Convex Discs. *Acta Math. Acad. Sci. Hungar.*, 29(3-4):355–360.

Ferreira, L., Pilastri, A., Sousa, V., Romano, F., and Cortez, P. (2021). Prediction of Maintenance Equipment Failures Using Automated Machine Learning. In Yin, H., Camacho, D., Tino, P., Allmendinger, R., Tallón-Ballesteros, A. J., Tang, K., Cho, S.-B., Novais, P., and Nascimento, S., editors, *Intelligent Data Engineering and Automated Learning – IDEAL 2021*, pages 259–267, Cham. Springer International Publishing.

Fiedler, M. (1973). Algebraic Connectivity of Graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305.

Fillard, P., Arsigny, V., Ayache, N., and Pennec, X. (2005). A Riemannian Framework for the Processing of Tensor-Valued Images. In Fogh Olsen, O., Florack, L., and Kuijper, A., editors, *Deep Structure, Singularities, and Computer Vision*, pages 112–123, Berlin, Heidelberg. Springer Berlin Heidelberg.

Fisher, R. A. (1936). The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7(2):179–188.

Fortune, S. (1987). A Sweepline Algorithm for Voronoi Diagrams. *Algorithmica*, 2(1):153–174.

Friedman, J. H. (1991). Multivariate Adaptive Regression Splines. *Ann. Statist.*, 19(1):1–141. With discussion and a rejoinder by the author.

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232.

Friedman, J. H., Bentley, J. L., and Finkel, R. A. (1977). An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transactions on Mathematical Software*, 3(3):209–226.

Galceran, E. and Carreras, M. (2013). A Survey on Coverage Path Planning for Robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276.

Green, P. J. (1995). Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination. *Biometrika*, 82(4):711–732.

Guibas, L. and Stolfi, J. (1985). Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi. *ACM Trans. Graph.*, 4(2):74–123.

Hahn, P. R., Murray, J. S., and Carvalho, C. M. (2020). Bayesian Regression Tree Models for Causal Inference: Regularization, Confounding, and Heterogeneous Effects (with Discussion). *Bayesian Analysis*, 15(3):965–1056.

Hastie, T. and Tibshirani, R. (1986). Generalized Additive Models. *Statistical Science*, 1(3):297–310.

Hastie, T. and Tibshirani, R. (2000). Bayesian Backfitting. *Statist. Sci.*, 15(3):196–223.

Hastie, T. J. (1990). *Generalized Additive Models*. Routledge, 1st edition.

He, J., Yalov, S., and Hahn, P. R. (2019). XBART: Accelerated Bayesian Additive Regression Trees.

Hill, J. L. (2011). Bayesian Nonparametric Modeling for Causal Inference. *Journal of Computational and Graphical Statistics*, 20(1):217–240.

Imai, K. and van Dyk, D. A. (2005). A Bayesian Analysis of the Multinomial Probit Model Using Marginal Data Augmentation. *Journal of Econometrics*, 124(2):311–334.

Inglis, A., Parnell, A., and Hurley, C. (2022). Visualizations for Bayesian Additive Regression Trees.

Ishwaran, H., Kogalur, U. B., Blackstone, E. H., and Lauer, M. S. (2008). Random Survival Forests. *The Annals of Applied Statistics*, 2(3):841–860.

James, W. and Stein, C. (1960). Estimation with Quadratic Loss. In *Proc. 4th Berkeley Sympos. Math. Statist. and Prob., Vol. I*, pages 361–379. Univ. California Press, Berkeley-Los Angeles, Calif.

Kaplan, E. L. and Meier, P. (1958). Nonparametric Estimation from Incomplete Observations. *Journal of the American Statistical Association*, 53(282):457–481.

Katzman, J., Shaham, U., Cloninger, A., Bates, J., Jiang, T., and Kluger, Y. (2018). DeepSurv: Personalized Treatment Recommender System Using a Cox Proportional Hazards Deep Neural Network. *BMC Medical Research Methodology*, 18.

Kindo, B. P., Wang, H., and Peña, E. A. (2016). Multinomial Probit Bayesian Additive Regression Trees. *Stat*, 5(1):119–131.

Labatut, P., Pons, J.-P., and Keriven, R. (2007). Efficient Multi-View Reconstruction of Large-Scale Scenes Using Interest Points, Delaunay Triangulation and Graph Cuts. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8.

Labelle, F. and Shewchuk, J. R. (2003). Anisotropic Voronoi Diagrams and Guaranteed-Quality Anisotropic Mesh Generation. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, SCG '03, pages 191–200, New York, NY, USA. Association for Computing Machinery.

Lakshminarayanan, B., Roy, D. M., and Teh, Y. W. (2015). Particle Gibbs for Bayesian Additive Regression Trees. *ArXiv*, abs/1502.04622.

Le, Q. V., Smola, A. J., and Canu, S. (2005). Heteroscedastic Gaussian Process Regression. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pages 489–496, New York, NY, USA. Association for Computing Machinery.

Lee, J. and Hwang, B. S. (2024). Ordered Probit Bayesian Additive Regression Trees for Ordinal Data. *Stat*, 13(1):e643.

Lee, T. and Liu, J. S. (2023). Multi-Response Heteroscedastic Gaussian Process Models and Their Inference.

Leevy, J., Hancock, J., Khoshgoftaar, T., and Abdollah Zadeh, A. (2023). Investigating the Effectiveness of One-Class and Binary Classification for Fraud Detection. *Journal of Big Data*, 10.

Lewis-Faupel, S. and Tenev, N. (2024). Racial and Ethnic Disparities in Mortgage Lending: New Evidence from Expanded HMDA Data.

Li, H., Li, K., Kim, T., Zhang, A., and Ramanathan, M. (2012). Spatial Modeling of Bone Microarchitecture. In Baskurt, A. M. and Sitnik, R., editors, *Three-Dimensional Image Processing (3DIP) and Applications II*, volume 8290, page 82900P. International Society for Optics and Photonics, SPIE.

Li, T., Lei, L., Bhattacharyya, S., den Berge, K. V., Sarkar, P., Bickel, P. J., and Levina, E. (2022). Hierarchical Community Detection by Recursive Partitioning. *Journal of the American Statistical Association*, 117(538):951–968.

Linero, A. (2016). Bayesian Regression Trees for High Dimensional Prediction and Variable Selection. *Journal of the American Statistical Association*, 113.

Linero, A. R. and Yang, Y. (2018). Bayesian Regression Tree Ensembles that Adapt to Smoothness and Sparsity. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 80(5):1087–1110.

Liu, J. S. and Wu, Y. N. (1999). Parameter Expansion for Data Augmentation. *Journal of the American Statistical Association*, 94(448):1264–1274.

Liu, Y., Gelman, A., and Chen, Q. (2022). Inference from Nonrandom Samples Using Bayesian Machine Learning. *Journal of Survey Statistics and Methodology*, 11(2):433–455.

Liu, Y., Ročková, V., and Wang, Y. (2021). Variable Selection with ABC Bayesian Forests. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 83(3):453–481.

Loh, W.-Y., Shih, Y.-S., and Chaudhuri, P. (2007). Visualizable and Interpretable Regression Models with Good Prediction Power. *IIE Transactions*, 39:565–579.

Luo, C. and Daniels, M. J. (2024). Variable Selection Using Bayesian Additive Regression Trees. *Statistical Science*, 39(2):286–304.

Luo, Z. T., Sang, H., and Mallick, B. (2021). A Bayesian Contiguous Partitioning Method for Learning Clustered Latent Variables. *Journal of Machine Learning Research*, 22(37):1–52.

Mantel, N. (1966). Evaluation of Survival Data and Two New Rank Order Statistics Arising in Its Consideration. *Cancer chemotherapy reports*, 50(3):163–170.

Meng, X.-L. and Dyk, D. A. V. (1999). Seeking Efficient Data Augmentation Schemes via Conditional and Marginal Augmentation. *Biometrika*, 86(2):301–320.

Micci-Barreca, D. (2001). A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems. *SIGKDD Explor. Newsl.*, 3(1):27–32.

Mougan, C., Masip, D., Nin, J., and Pujol, O. (2021). Quantile Encoder: Tackling High Cardinality Categorical Features in Regression Problems. In Torra, V. and Narukawa, Y., editors, *Modeling Decisions for Artificial Intelligence*, pages 168–180, Cham. Springer International Publishing.

Murthy, S. K., Kasif, S., and Salzberg, S. (1994). A System for Induction of Oblique Decision Trees. *Journal of Artificial Intelligence Research*, 2:1–32.

Nelder, J. A. and Wedderburn, R. W. M. (1972). Generalized Linear Models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3):370–384.

Nguyen, C. P., Su, T. D., Bui, T. D., Dang, V. T. B., and Nguyen, B. Q. (2021). Financial Development and Energy Poverty: Global Evidence. *Environmental Science and Pollution Research*, 28(26):35188–35225.

Nguyen, P.-H. V., Yee, R., and Deshpande, S. K. (2024). Oblique Bayesian Additive Regression Trees.

Owen, A. B. (2017). Statistically Efficient Thinning of a Markov Chain Sampler. *Journal of Computational and Graphical Statistics*, 26(3):738–744.

Pargent, F., Pfisterer, F., Thomas, J., and Bischl, B. (2022). Regularized Target Encoding Outperforms Traditional Methods in Supervised Machine Learning with High Cardinality Features. *Computational Statistics*, 37(5):2671–2692.

Pennisi, A., Bloisi, D. D., Nardi, D., Giampetruzzi, A. R., Mondino, C., and Facchiano, A. (2016). Skin Lesion Image Segmentation Using Delaunay Triangulation for Melanoma Detection. *Computerized Medical Imaging and Graphics*, 52:89–103.

Peters, G. (2006). Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference (2nd edn). Dani Gamerman and Hedibert F. Lopes, Chapman & Hall/CRC, Boca Raton, FL. *Statistics in Medicine*, 27(16):3213–3214.

Poincaré, H. (1882). Théorie des groupes fuchsiens. *Acta Mathematica*, 1:1–62.

Pope, C. A., Gosling, J. P., Barber, S., Johnson, J. S., Yamaguchi, T., Feingold, G., and Blackwell, P. G. (2021). Gaussian Process Modeling of Heterogeneity and Discontinuities Using Voronoi Tessellations. *Technometrics*, 63(1):53–63.

Popick, S. (2022). Did Minority Applicants Experience Worse Lending Outcomes in the Mortgage Market? A Study Using 2020 Expanded HMDA Data. *SSRN Electronic Journal*.

Potdar, K., Pardawala, T. S., and Pai, C. D. (2017). A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers. *International Journal of Computer Applications*, 175(4):7–9.

Pratola, M. T. (2016). Efficient Metropolis–Hastings Proposal Mechanisms for Bayesian Regression Tree Models. *Bayesian Analysis*, 11(3):885–911.

Pratola, M. T., Chipman, H. A., George, E. I., and McCulloch, R. E. (2020). Heteroscedastic BART via Multiplicative Regression Trees. *Journal of Computational and Graphical Statistics*, 29(2):405–417.

Ran, H. and Bai, Y. (2021). Distributed Soft Bayesian Additive Regression Trees.

Rodriguez, J., Kuncheva, L., and Alonso, C. (2006). Rotation Forest: A New Classifier Ensemble Method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630.

Rubin, D. B. (1974). Estimating Causal Effects of Treatments in Randomized and Nonrandomized Studies. *Journal of Educational Psychology*, 66:688–701.

Schapire, R. E. (1990). The Strength of Weak Learnability. *Machine Learning*, 5(2):197–227.

Schulz, E., Speekenbrink, M., and Krause, A. (2018). A Tutorial on Gaussian Process Regression: Modelling, Exploring, and Exploiting Functions. *Journal of Mathematical Psychology*, 85:1–16.

Schwab, A. and Lunze, J. (2023). Formation Control Over Delaunay Triangulation Networks With Guaranteed Collision Avoidance. *IEEE Transactions on Control of Network Systems*, 10:419–429.

Sparapani, R., Logan, B. R., McCulloch, R. E., and Laud, P. W. (2020). Nonparametric Competing Risks Analysis Using Bayesian Additive Regression Trees. *Statistical Methods in Medical Research*, 29(1):57–77. PMID: 30612519.

Sparapani, R., Spanbauer, C., and McCulloch, R. (2021). Nonparametric Machine Learning and Efficient Computation with Bayesian Additive Regression Trees: The BART R Package. *Journal of Statistical Software*, 97(1):1–66.

Sparapani, R. A., Logan, B. R., McCulloch, R. E., and Laud, P. W. (2016). Nonparametric Survival Analysis Using Bayesian Additive Regression Trees (BART). *Statistics in Medicine*, 35(16):2741–2753.

Sparapani, R. A., Rein, L. E., Tarima, S. S., Jackson, T. A., and Meurer, J. R. (2018). Non-Parametric Recurrent Events Analysis with BART and an Application to the Hospital Admissions of Patients with Diabetes. *Biostatistics*, 21(1):69–85.

Stone, A. J. and Gosling, J. P. (2025a). AddiVortes: (Bayesian) Additive Voronoi Tessellations. *Journal of Computational and Graphical Statistics*, 34(3):859–871.

Stone, A. J. and Gosling, J. P. (2025b). H-AddiVortes: Heteroscedastic (Bayesian) Additive Voronoi Tessellations.

Stone, A. J., Ogundimu, E., and Gosling, J. P. (2025). Binary AddiVortes: (Bayesian) Additive Voronoi Tessellations for Binary Classification with an Application to Predicting Home Mortgage Application Outcomes.

Stone, C. J. (1982). Optimal Global Rates of Convergence for Nonparametric Regression. *The Annals of Statistics*, 10(4):1040–1053.

Tanner, M. A. and Wong, W. H. (1987). The Calculation of Posterior Distributions by Data Augmentation. *Journal of the American Statistical Association*, 82(398):528–540.

Teixeira, L. V., Assunção, R. M., and Loschi, R. H. (2019). Bayesian Space-Time Partitioning by Sampling and Pruning Spanning Trees. *Journal of Machine Learning Research*, 20(85):1–35.

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

Tomita, T. M., Browne, J., Shen, C., Chung, J., Patsolic, J. L., Falk, B., Priebe, C. E., Yim, J., Burns, R., Maggioni, M., and Vogelstein, J. T. (2020). Sparse Projection Oblique Randomer Forests. *Journal of Machine Learning Research*, 21(104):1–39.

Uhlmann, J. K. (1991). Satisfying General Proximity/Similarity Queries with Metric Trees. *Information Processing Letters*, 40(4):175–179.

Villagran, A., Huerta, G., Vannucci, M., Jackson, C. S., and Nosedal, A. (2016). Non-Parametric Sampling Approximation via Voronoi Tessellations. *Communications in Statistics - Simulation and Computation*, 45(2):717–736.

Vonta, F. (2009). The Frailty Model. *Journal of Applied Statistics*, 36(8):927–928.

Voronoi, G. (1908). Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier mémoire. Sur quelques propriétés des formes quadratiques positives parfaites. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1908(133):97–102.

Wahba, G. (1990). *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics.

Waldmann, P. (2016). Genome-Wide Prediction Using Bayesian Additive Regression Trees. *Genetics Selection Evolution*, 48.

Wand, M. P. and Jones, M. C. (1995). *Kernel Smoothing*, volume 60 of *Monographs on Statistics and Applied Probability*. Chapman and Hall, Ltd., London.

Wang, M., Xu, X., Yue, Q., and Wang, Y. (2021). A Comprehensive Survey and Experimental Comparison of Graph-Based Approximate Nearest Neighbor Search. *Proc. VLDB Endow.*, 14(11):1964–1978.

Watson, D. F. (1981). Computing the n-Dimensional Delaunay Tessellation with Application to Voronoi Polytopes. *The Computer Journal*, 24(2):167–172.

Wilson, D. B. (1996). Generating Random Spanning Trees More Quickly Than the Cover Time. In *Symposium on the Theory of Computing*.

Wong, D. W. (2017). *Interpolation: Inverse-Distance Weighting*, pages 1–7. John Wiley & Sons, Ltd.

Wu, Y., Tjelmeland, H., and West, M. (2007). Bayesian CART: Prior Specification and Posterior Simulation. *Journal of Computational and Graphical Statistics*, 16(1):44–66.

Xu, Y., Hogan, J., Daniels, M., Kantor, R., and Mwangi, A. (2025). Augmentation Samplers for Multinomial Probit Bayesian Additive Regression Trees. *Journal of Computational and Graphical Statistics*, 34(2):498–508.

Yang, Y. and Dunson, D. B. (2014). Minimax Optimal Bayesian Aggregation.

Ye, S. and Li, M. (2025a). Ab Initio Nonparametric Variable Selection for Scalable Symbolic Regression with Large p.

Ye, S. and Li, M. (2025b). Posterior Summarization for Variable Selection in Bayesian Tree Ensembles.

Zhang, J. L. and Härdle, W. K. (2010). The Bayesian Additive Classification Tree Applied to Credit Risk Modelling. *Computational Statistics & Data Analysis*, 54(5):1197–1205.

Zhang, X., Pak, D. H., Ahn, S. S., Li, X., You, C., Staib, L. H., Sinusas, A. J., Wong, A., and Duncan, J. S. (2024). Heteroscedastic Uncertainty Estimation Framework for Unsupervised Registration. In *Proceedings of Medical Image Computing and Computer Assisted Intervention – MICCAI 2024*, volume LNCS 15002, pages 651–661. Springer Nature Switzerland.

Zhang, Z. (2016). Variable Selection with Stepwise and Best Subset Approaches. *Annals of Translational Medicine*, 4:136–136.

Zhu, W., Qiu, R., and Fu, Y. (2024). Comparative Study on the Performance of Categorical Variable Encoders in Classification and Regression Tasks.

İrsoy, O., Yıldız, O. T., and Alpaydın, E. (2012). Soft Decision Trees. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 1819–1822.