Bayesian Additive Modeling

STAT 220 Final Project

Zad Chin Jarell Cheong

Harvard University

May 8, 2023

Presentation Overview

Overview of the Project

Model Setup and Purpose of the Project

Suppose we have n datapoints $\{(x_i, y_i)\}_{i=1}^n$, where x_i has p features (put differently, x_i is p-dimensional). In Statistics 220, we learned about linear models of the form

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i, \quad i = 1, \ldots, n, \tag{1}$$

where $x_i = (x_{i1}, \ldots, x_{ip})^t$ and $\epsilon_i \sim N(0, \sigma^2)$ independently.

The primary limitations of the above-defined linear model arise from their parametric, model-driven nature, resulting in a high degree of dependence on the assumptions of the model rather than the data.

Overview of the Project

Model Setup and Purpose of the Project

The development of Additive Models (AM) serves to address the issue. In our project, we will focus on two main goals:

- Survey and study, in great detail, the Bayesian approach to AMs.
- Implement these approaches, as well as other benchmark models, to data.

AMs

The additive model (AM) is a model taking on the following form:

$$y_i = \beta_0 + f_1(x_{i1}) + \cdots + f_p(x_{ip}) + \epsilon_i, \;\; i = 1, \ldots, n, \tag{2}$$

where again $x_i = (x_{i1}, \ldots, x_{ip})^t$ and $\epsilon_i \sim N(0, \sigma^2)$ independently.

Here, $f_1, \ldots, f_p$ are smooth and nonparametric functions to be estimated from the data.

AMs vs Other Models

Linear Model

1. Biased
2. Interpretable

AMs

1. Flexible
2. Interpretable
3. Regularization

Machine Learning

1. Flexible
2. Not Interpretable

More on AMs

Advantages

- **Interpretability** From the output of AMs, we can determine the effect of any particular predictive variables
- **Flexibility** Predictor functions are automatically derived during model estimation, and one does not need to know upfront which ”type” of global functions they will need
- **Regularization** The AM framework allows for the control of smoothness of the predictor functions to prevent overfitting

Challenges

How do we represent smooth functions and choose how smooth they should be?

Simple AMs setup

For simplicity, consider a model containing only one function of one covariate, where

$$y_i = f(x_i) + \epsilon_i, \;\; i = 1, \ldots, n, \tag{3}$$

where f is a smooth function and $\epsilon_i \sim N(0, \sigma^2)$ independently.

In the penalized regression framework, we consider estimating the function f by choosing a basis.

Supppose first that we know what a basis for f is, and let $b_j(x)$ be the jth such basis function. Then, f has a representation

$$f(x) = \sum_{j=1}^{k} b_j(x)\beta_j, \tag{4}$$

for some values of the unknown parameters, β_j. Substituting (4) into (3) will yield the familiar linear model

$$y = X\beta + \epsilon, \tag{5}$$

where the vectors y, β, ϵ and the matrix X are defined in the following manner:

$$y = (y_1, \ldots, y_n)^t, \tag{6}$$
$$\beta = (\beta_1, \ldots, \beta_k)^t, \tag{7}$$
$$\epsilon = (\epsilon_1, \ldots, \epsilon_n)^t, \tag{8}$$
$$X_{ij} = b_j(x_i), \quad i = 1, \ldots, n, \quad j = 1, \ldots, k. \tag{9}$$

Two Questions

1. How to control the “smoothness” of the resulting model?
2. How to find suitable basis functions $b_j(x)$?

We are going to investigate how to tackle these two questions from both Frequentist and Bayesian approaches [1] [2].

[1] We note that the Frequentist approach to these questions, as well as the Bayesian approach to question (1) is well-studied in Wood’s textbook. Our main contribution is to study and implement the recent advancements in the Bayesian approach to Question (2).

[2] We also note that while we generally discuss AMs, the key ideas of this presentation/paper also generalize to fitting Generalized Additive Models as well, with a few caveats discussed in Subsection 1.4 of the paper.

Frequentist Approach

Q2: How to find the basis function $b_j(x)$ - Frequentist Approach

Consider some natural basis choices -
Piecewise Linear Functions, Natural Cubic Splines

Lagrange's Interpolation

Theorem

Given n datapoints $\{(x_i, y_i)\}_{i=1}^{n}$, where the x_i's are distinct, there is a polynomial P with real coefficients which satisfies $P(x_i) = y_i$ for each $i = 1, \ldots, n$. Moreover, if the condition $\deg(P) < n$ is imposed, such a polynomial P is unique.

The problem with Lagrange's Interpolation

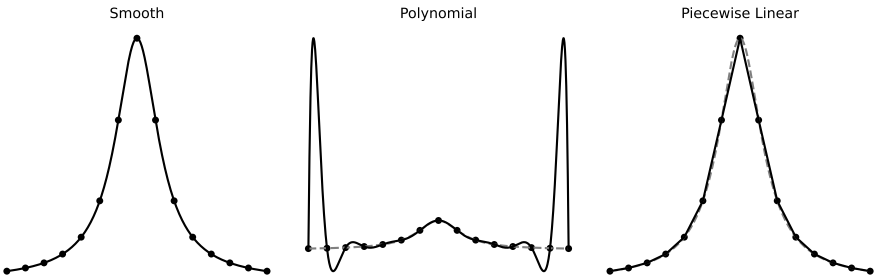


Figure: The left panel displays the smooth function $1/(1+x^2)$ and also some points (in black dots) on this function. The middle panel shows an attempt to reconstruct (in black) the function (dashed curve) by polynomial interpolation of the black dots. The rightmost panel shows the equivalent piecewise linear interpolant. The condition that polynomials should interpolate the data with all its derivatives continuous yields wilder oscillations.

Piecewise Linear Functions

Definition

A basis for piecewise linear functions of a univariate variable x is determined entirely by the locations of the function's derivative discontinuities; that is, the locations at which the linear pieces join up. These are called knots. Let them be

$$\{x_j^* \mid j = 1, \ldots, k\}. \tag{10}$$

Suppose additionally that $x_j^* > x_{j-1}^*$. For $b_1(x)$, the piecewise linear basis is given by

$$b_1(x) = \begin{cases} (x_2^* - x)/(x_2^* - x_1^*) & x < x_2^*, \\ 0 & \text{otherwise}, \end{cases} \tag{11}$$

Cont. on Piecewise Linear Functions

Following from above, for $j = 2, \ldots, k-1$:

$$b_j(x) = \begin{cases} (x - x^*_{j-1})/(x^*_j - x^*_{j-1}) & x^*_{j-1} < x \leq x^*_j, \\ (x^*_{j+1} - x)/(x^*_{j+1} - x^*_j) & x^*_j < x < x^*_{j+1}, \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

and

$$b_k(x) = \begin{cases} (x - x^*_{k-1})/(x^*_k - x^*_{k-1}) & x > x^*_{k-1}, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

Note that because of their shape, the b_j are often known as tent functions.

An illustration of tent function

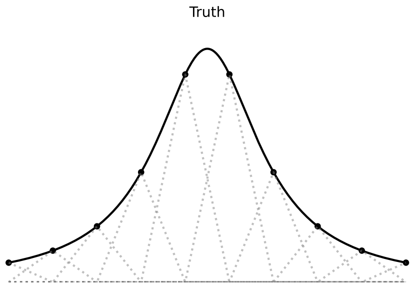


Figure: Fitting points on a smooth curve with a combination of tent functions.

Knots & Smoothness

Knots & Smoothness

- More knots - smoother prediction
- Less knots - rougher prediction

Challenges

How do we choose a suitable number of knots?

- Adding penalty term that regularizes towards smoothness and choose a smoothing parameter λ by cross-validation (more on this later)

Why Splines?

Claim

If we choose to represent our smooths in terms of spline bases, it is possible to get greatly reduced approximation error for our functions when the dimension of the smoothing basis is fixed.

We will prove our claim above by diving into the theoretical properties of cubic splines to explain why splines are so appealing for penalized regression.

Cubic Splines

Consider the set of points

$$\{(x_i, y_i) \mid i = 1, \ldots, n\}, \;\; x_i < x_{i+1}. \tag{14}$$

Then, the natural cubic spline, $g(x)$, interpolating these points, is a function that is made up of sections of cubic polynomial, one for each $[x_i, x_{i+1}]$, which are then joined together such that the whole spline is continuous to the second derivative and $g(x_i) = y_i$, $g''(x_1) = g''(x_n) = 0$ are both satisfied.

Cubic Splines produces Smoothest Interpolation

Theorem

Of all functions which are continuous on $[x_1, x_n]$ *having absolutely continuous first derivatives and interpolating* $\{(x_i, y_i)\}_{i=1}^n$, $g(x)$ *is the smoothest one in the sense that this natural cubic spline is the minimizer of the following integral:*

$$J(f) = \int_{x_1}^{x_n} f''(x)^2 \, dx. \tag{15}$$

Proof
A complete proof can be found in the paper.

Cubic Splines is the best smoother

Theorem

Of all functions f that are continuous on $[x_1, x_n]$ and have absolutely continuous first derivatives, $g(x)$ is the function that minimizes the expression below:

$$\sum_{i=1}^{n}[y_i - f(x_i)]^2 + \lambda \int f''(x)^2 \, dx. \tag{16}$$

Proof

A complete proof can be found in the paper.

Cubic Basis Function on f

Assume we are trying to estimate a smooth function f of a single variable, but now with a cubic spline basis that has k knots $x_1, \ldots, x_k$. Then, define the expressions

$$h_j = x_{j+1} - x_j, \tag{17}$$

$$a_j^-(x) = (x_{j+1} - x)/h_j, \tag{18}$$

$$a_j^+(x) = (x - x_j)/h_j, \tag{19}$$

$$c_j^-(x) = [(x_{j+1} - x)^3/h_j - h_j(x_{j+1} - x)]/6, \tag{20}$$

$$c_j^+(x) = [(x - x_j)^3/h_j - h_j(x - x_j)]/6, \tag{21}$$

$$D_{i,i} = 1/h_i, \tag{22}$$

$$D_{i,i+1} = -1/h_i - 1/h_{i+1}, \tag{23}$$

$$D_{i,i+2} = 1/h_{i+1}, \tag{24}$$

$$B_{i,i} = (h_i + h_{i+1})/3, \tag{25}$$

$$B_{i,i+1} = h_{i+1}/6, \tag{26}$$

$$B_{i+1,i} = h_{i+1}/6, \tag{27}$$

where $j = 1, \ldots, k$ and $i = 1, \ldots, k-2$ for D, $i = 1, \ldots, k-3$ for B.

Cont. Cubic Basis Function on f

Additionally, let $\beta_j = f(x_j)$ and $\delta_j = f''(x_j)$. Then, we can express

$$f(x) = a_j^-(x)\beta_j + a_j^+(x)\beta_{j+1} + c_j^-(x)\delta_j + c_j^+(x)\delta_{j+1}, \quad x_j \le x \le x_{j+1}, \tag{28}$$

where the basis functions $a_j^-, a_j^+, c_j^-, c_j^+$ are defined in (18), (19), (20), (21). We have that the cubic regression spline defined by (28) has value β_j and second derivative δ_j at knot x_j, and that value and second derivative are continuous across the knots.

Illustration of Cubic Basis Function

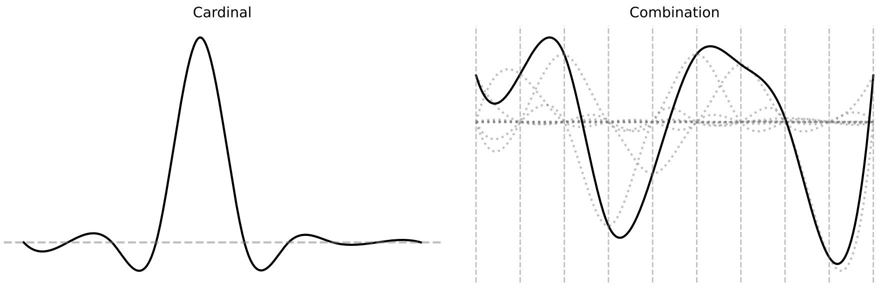


Figure: The left panel illustrates one basis function for a cubic regression spline of the type discussed in (28): this basis function takes the value one at one knot of the spline, and zero at all other knots (such basis functions are sometimes denoted as “cardinal basis functions”). The right panel shows how such basis functions are combined to represent a smooth curve. The various curves of medium thickness show the basis functions of a cubic regression spline, each multiplied by their associated coefficient: these scaled basis functions are summed to get the smooth curve illustrated by the thick continuous curve, and additionally, the vertical thin lines display the knot locations which were specified.

Q1: How to control the ”smoothness” of the resulting model - Frequantist Approach

Cross Validation

Cross Validation Setup

Given datapoints $\{((x_i, v_i), y_i)\}_{i=1}^{n}$, where the x_i's and v_i's are one-dimensional, our model shall take on the form

$$y_i = \alpha + f_1(x_i) + f_2(v_i) + \epsilon_i, \quad i = 1, \ldots, n, \tag{29}$$

where $\epsilon_i \sim N(0, \sigma^2)$ independently. This is the two-component special case.
Observe that the coefficient estimates of the model (29) are obtained by the minimization of the penalized least squares objective given, by the form

$$\|y - X\beta\|^2 + \lambda_1 \beta^t S_1 \beta + \lambda_2 \beta^t S_2 \beta, \tag{30}$$

where the smoothing parameters λ_1 and λ_2 control the weight to be given to the objective of ensuring that f_1 and f_2 are smooth, relative to the objective of closely fitting the response data.
Question: How to we estimate $\lambda = (\lambda_1, \lambda_2)$?

Unbiased Risk Estimator (UBRE)

When the variance of the data is known and constant, we can try to ensure that the expected mean square error is minimized, i.e. that

$$M = \mathbb{E}(\|\mathbb{E}(y) - X\hat{\beta}\|^2/n) = \mathbb{E}(\|y - Ay\|^2)/n - \sigma^2 + 2\text{tr}(A)\sigma^2/n \quad (31)$$

is minimized. This leads to the unbiased risk estimator (UBRE) introduced in 1979 by Craven and Wahba. The UBRE is given by the following expression below:

$$\mathcal{V}_u = \|y - Ay\|^2/n - \sigma^2 + 2\text{tr}(A)\sigma^2/n. \quad (32)$$

Question: What if the variance is not known?

Mean Square Prediction Error

An alternative is to base the smoothing parameter estimation on mean square prediction error; that is, on the average squared error in predicting some new observation $\tilde{y}$ using the fitted model. This expected mean square prediction error is simply

$$P = \sigma^2 + M. \tag{33}$$

The direct dependence on σ^2 suggests that basing criteria on P ensures that said criteria are more resistant to over-smoothing, which would inflate the σ^2 estimate, than are criteria based on M alone.

We will estimate P with cross validation.

Ordinary Cross Validation (OCV)

Theorem

The ordinary cross validation (OCV) estimate, $\mathcal{V}_o$, can be written as

$$\mathcal{V}_o = \frac{1}{n} \sum_{i=1}^{n} \frac{(y_i - \hat{\mu}_i)^2}{(1 - A_{ii})^2}, \tag{34}$$

where $\hat{\mu}_i$ is from the fit to the full y. Whence, $\mathcal{V}_o$ can be calculated from a single fit.

Proof A complete proof can be found in the paper.

Drawback: $\mathcal{V}_o$ has a slightly disturbing lack of invariance.

Drawbacks of OCV

Consider the additive model fitting objective

$$\|y - X\beta\|^2 + \lambda_1 \beta^t S_1 \beta + \lambda_2 \beta^t S_2 \beta. \tag{35}$$

Given smoothing parameters, all inferences about β made on the basis of minimizing the above are identical to the inferences that are made by using the alternate objective

$$\|Qy - QX\beta\|^2 + \lambda_1 \beta^t S_1 \beta + \lambda_2 \beta^t S_2 \beta, \tag{36}$$

where Q is some orthogonal matrix.

However, the two objectives above generally give rise to different OCV scores!

Generalized Cross Validation (GCV)

To fix this, we can instead consider the generalized cross validation (GCV) score:

$$\mathcal{V}_g = \frac{n\|y - \hat{\mu}\|^2}{[n - \text{tr}(A)]^2} \tag{37}$$

Here, $\hat{\mu}$ is the estimate of $\mathbb{E}(y)$ from the fit. Note that since the expected prediction error is unaffected by the rotation, and GCV is just OCV on the rotated problem, thus GCV must be as valid an estimate of prediction error as OCV, but of course, GCV has the nice property of being invariant under rotation by orthogonal matrices Q.

Double Cross Validation

Both OCV and GCV are not sensitive enough to overfit. Thus, we will do worse than we could with actual predictions.

Thus, we introduce Double Cross Validation (DCV) score:

$$\mathcal{V}_d = \frac{n\|y - \hat{\mu}\|^2}{[n - 1.5\mathrm{tr}(A)]^2}. \tag{38}$$

Numerical Method

Estimating the functions in AM (or GAMs) directly without the use of basis functions or smoothing penalties

Backfitting Algorithm Setup

Our model setup here will be the same as in (2); that is,

$$y_i = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}), \quad i = 1, \ldots, n, \tag{39}$$

where $x_i = (x_{i1}, \ldots, x_{ip})^t$ and $\epsilon_i \sim N(0, \sigma^2)$ independently for the data $\{(x_i, y_i)\}_{i=1}^{n}$.

Backfitting Algorithm

Algorithm 1: The Backfitting Algorithm for Additive Models

```
initialize $\beta_0 = \bar{y}$
initialize $f_j = 0$ for $j = 1, \ldots, p$
repeat
    for $j = 1, \ldots, p$ do
        $e_j \leftarrow y - \beta_0 - \sum_{k \neq j} f_k$
        $f_j \leftarrow S_j[e_j]$
        $f_j \leftarrow f_j - \sum_{i=1}^{n} f_j(x_{ij})/n$
    $\delta \leftarrow \sum_{i=1}^{n} [y_i - \beta_0 - \sum_{j=1}^{p} f_j(x_{ij})]^2$
until $\delta \leq \epsilon$
return $f_j$ for $j = 1, \ldots, p$
```

Convergence properties of Backfitting Algorithm

1. The backfitting algorithm converges for a restrictive and impractical set of smoothers.
2. The backfitting algorithm is mean square consistent for a slightly less restrictive class of smoothers in the following sense: if $(f_j)_n$ is the estimate of f_j applied to a finite sample with size n, and $(f_j)_\infty$ is the estimate of f_j applied to infinite samples (the distribution), then as $n \to \infty$, the convergence below holds:

$$\mathbb{E}[(f_j)_n(x) - (f_j)_\infty(x)]^2 \to 0. \tag{40}$$

Advantages and Disadvantages of Backfitting Algorithm

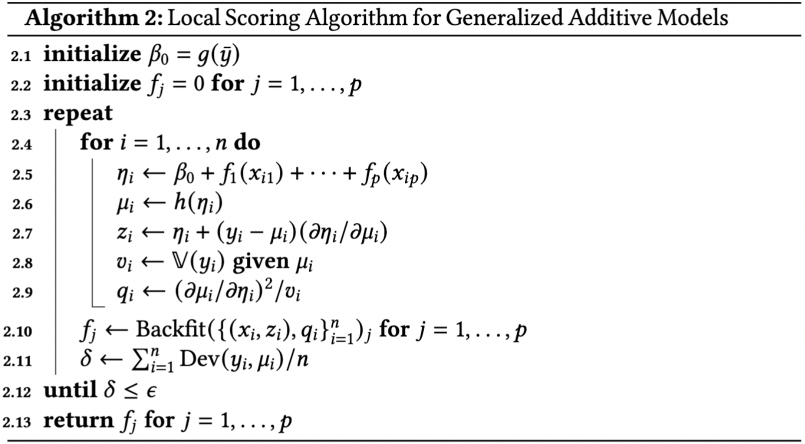
Advantages

1. High Flexibility

Disadvantages

1. Difficulty in estimating smoothness
2. Depends on the order in which the predictor variables are fit
3. Arbitrary Stopping Conditions

Local Scoring Algorithm



Algorithm 2: Local Scoring Algorithm for Generalized Additive Models

```
2.1  initialize $\beta_0 = g(\bar{y})$
2.2  initialize $f_j = 0$ for $j = 1, \ldots, p$
2.3  repeat
2.4      for $i = 1, \ldots, n$ do
2.5          $\eta_i \leftarrow \beta_0 + f_1(x_{i1}) + \cdots + f_p(x_{ip})$
2.6          $\mu_i \leftarrow h(\eta_i)$
2.7          $z_i \leftarrow \eta_i + (y_i - \mu_i)(\partial\eta_i / \partial\mu_i)$
2.8          $v_i \leftarrow \mathbb{V}(y_i)$ given $\mu_i$
2.9          $q_i \leftarrow (\partial\mu_i / \partial\eta_i)^2 / v_i$
2.10     $f_j \leftarrow \text{Backfit}(\{(x_i, z_i), q_i\}_{i=1}^{n})_j$ for $j = 1, \ldots, p$
2.11     $\delta \leftarrow \sum_{i=1}^{n} \text{Dev}(y_i, \mu_i)/n$
2.12 until $\delta \leq \epsilon$
2.13 return $f_j$ for $j = 1, \ldots, p$
```

Details

1. The backfitting algorithm can be extended to include weights $\{q_i\}_{i=1}^{n}$: just transform the data to have predictors $\{q_i x_i\}_{i=1}^{n}$ and apply the backfitting algorithm to this data.
2. The GAM has link function g with inverse $h = g^{-1}$.
3. Dev is the deviance, which is usually defined in standard GLM classes, or in the textbook by McCullagh and Nelder.

Bayesian Approach

Q1: How to control the “smoothness” of the resulting model - Bayesian Approach

Controlling Smoothness by Putting a Prior on β

Wiggliness Penalty

Specifically, to control the smoothness, we are penalizing wiggliness. In the one-component model, rather than fitting the model by minimizing the classical penalty $\|y - X\beta\|^2$, we minimize

$$\|y - X\beta\|^2 + \lambda \beta^t S \beta \tag{41}$$

with respect to β. Here, S is a rank deficient matrix determined by the choice of basis functions $b_j(x)$. Then, note that

Theorem

The formal expression for the minimizer of (41), $\hat{\beta}$, *is given below:*

$$\hat{\beta} = (X^t X + \lambda S)^{-1} X^t y. \tag{42}$$

Prior on β

Consider a prior distribution on function wiggliness, $p(\beta)$:

$$p(\beta) \propto \exp(-\lambda \beta^t S \beta / \sigma^2), \tag{43}$$

an exponential prior. However, this prior is recognizable as being equivalent to an improper Multivariate Normal prior. Explicitly, this improper prior is simply

$$\beta \sim N(0, \sigma^2 S^- / \lambda), \tag{44}$$

where S^- is a pseudoinverse of S (because S is rank deficient by the dimension of the penalty null space, it does not necessarily have an inverse).

Bayesian Interpertation of Smoothing Penalty

Note that with the prior on β defined above, the MAP estimate of β is the solution 42 to Eq. 41 and , (i.e. the β that miniminizes the wiggliness). Note that the posterior distribution is

$$\beta \mid y \sim N(\hat{\beta}, (X^t X + \lambda S)^{-1} \sigma^2). \tag{45}$$

Q2: How to choose a basis - Bayesian Approach

Gaussian Process Prior, Splines and Difference Priors

We will go into detail about advancements in this aspect next week!

The Model For the Next Two Sections

Our data will now be

$$\{(x_i, y_i)\}_{i=1}^n, \;\; x_i = (x_{i1}, \ldots, x_{ip})^t, \tag{46}$$

where each x_{ij} is now allowed to be a vector. Then, our additive model shall be

$$y_i = f_1(x_{i1}, \beta_1) + \cdots + f_p(x_{ip}, \beta_p) + \epsilon_i, \;\; i = 1, \ldots, n, \tag{47}$$

where the f_j are smooth functions, x_{ij} are the input observations for the function f_j, β_j are the vectors of parameters for the function f_j, and the ϵ_i are measurement errors.

Differences Between This and Previous Model

1. We no longer have a constant term β_0, for simplicity.
2. We explicitly encode the parameters β_j on which f_j depends on.
3. The x_{ij}'s are now vectors, so each f_j could be multivariable.
4. A priori, we do not assume a Normal distribution for the ϵ_i's.

One More Restriction

We will require that the f_j's are such that the full model is linear with respect to the parameters $\beta = (\beta_1, \ldots, \beta_p)^t$ (but can be nonlinear with respect to their respective inputs x_{ij}). Then, we further impose Normal prior distributions on the unknown parameter values β, which we will want to estimate.

One More Restriction Cont.

We want the full model for the estimation of β to always be expressible in the following compact form that is shown below:

$$y = X\beta + \epsilon, \tag{48}$$

$$B\beta \sim \mathcal{N}(\mu_{\text{pr}}, \Gamma_{\text{pr}}), \tag{49}$$

where $y = (y_1, \dots, y_n)^t$, $\epsilon = (\epsilon_1, \dots, \epsilon_n)^t$, and X and B are matrices. If we further suppose that the measurement errors are Normal, i.e. $\epsilon \sim \mathcal{N}(0, \Gamma_{\text{obs}})$, we can then realize this model as a linear Normal system, and by Normal-Normal conjugacy,

$$\beta \mid y \sim \mathcal{N}(\mu_{\text{pos}}, \Gamma_{\text{pos}}), \tag{50}$$

$$\Gamma_{\text{pos}}^{-1} = X^t \Gamma_{\text{obs}}^{-1} X + B^t \Gamma_{\text{pr}}^{-1} B, \tag{51}$$

$$\mu_{\text{pos}} = \Gamma_{\text{pos}} (X^t \Gamma_{\text{obs}}^{-1} y + B^t \Gamma_{\text{pr}}^{-1} \mu_{\text{pr}}). \tag{52}$$

Basis Representation

For $i = 1, \dots, p$ and each index j for the jth component of f_i,

$$(f_i)_j = \sum_{k=1}^{K_i} \alpha_{ik} b_{ik}(x_{ij}), \tag{53}$$

where now the unknown parameters β_i related to the function f_i are the weights α_{ik}. At times, we will also add a translation to this equation to obtain the modified

$$(f_i)_j = m_i(x_{ij}) + \sum_{k=1}^{K_i} \alpha_{ik} b_{ik}(x_{ij}), \tag{54}$$

where m_i is the translation we will associate to the smooth function f_i. Here, each m_i should be a known function of the input data, and we can basically interpret this as a sort of “mean function.”

Basis Representation Cont.

Then, the design matrix X can be written in the following “stacked” form:

$$X = \begin{pmatrix} \overbrace{b_{11}(x_{11}) \quad \cdots \quad b_{1K_1}(x_{11})}^{f_1} & \cdots & \overbrace{b_{p1}(x_{p1}) \quad \cdots \quad b_{pK_p}(x_{p1})}^{f_p} \\ b_{11}(x_{12}) \quad \cdots \quad b_{1K_1}(x_{12}) & \cdots & b_{p1}(x_{p2}) \quad \cdots \quad b_{pK_p}(x_{p2}) \\ \vdots & \vdots & \vdots \\ b_{11}(x_{1n}) \quad \cdots \quad b_{1K_1}(x_{1n}) & \cdots & b_{p1}(x_{pn}) \quad \cdots \quad b_{pK_p}(x_{pn}) \end{pmatrix}. \tag{55}$$

Basis From Gaussian Process Prior

Set priors on each function f_i as follows:

$$f_i(x, \beta) \sim \mathcal{GP}(m_i(x), k_i(x, x')), \tag{56}$$

where we specify the model parameters β later. Discretize f_i on a fixed grid of input points, when we get $\mathrm{f}_i \sim \mathcal{N}(\mu_{f_i}, \Sigma_{f_i})$, where μ_{f_i} and Σ_{f_i} are defined by

$$\mu_{f_i} = (m_i(a_1), \ldots, m_i(a_L)), \tag{57}$$
$$(\Sigma_{f_i})_{jk} = k_i(a_j, a_k), \tag{58}$$

where $a_1, \ldots, a_L$ form the grid of input points we specify.

Basis From Gaussian Process Prior Cont.

Compute the eigenvalue decomposition for the matrix Σ_{f_i}:

$$\Sigma_{f_i} = \sum_{r=1}^{L} \lambda_r q_r q_r^t = PP^t, \tag{59}$$

where λ_r is the rth eigenvalue of Σ_{f_i} with corresponding eigenvector q_r, and the rth column of the matrix P, p_r, is the rth eigenvector scaled by the its eigenvalue:

$$p_r = \sqrt{\lambda_r} q_r, \;\; r = 1, \ldots, L. \tag{60}$$

Basis From Gaussian Process Prior Cont.

By the representation of a Multivariate Normal,

$$\mathsf{f}_i = \mu_{f_i} + P\beta, \quad \beta \sim \mathcal{N}(0, I_L), \tag{61}$$

which can be recognized as the basis form given before. Explicitly, the parameters for our model, β, are the weights for the basis vectors (with the convenient i.i.d. Normal prior), the translation is given by μ_{f_i}, and the basis vectors would be encoded by P.

Problem 1

The number of basis vectors obtained (i.e. the basis dimension) is precisely the number L of grid points chosen. Thus, to have a sufficiently fine grid so that the function f_i looks continuous, we would end up with an enormously high basis dimension! Of course, this is undesirable as it would be computationally expensive (or in most cases, impossible) to store the resulting matrix P, and plus, such a model is very prone to overfitting anyway.

Solution to Problem 1

Use the “truncated SVD” method. Out of the L eigenvalues $\lambda_1, \ldots, \lambda_L$, we can select the first K eigenvalues $\lambda_1, \ldots, \lambda_K$ such that, for instance, we obtain

$$(\lambda_1 + \cdots + \lambda_K)/(\lambda_1 + \cdots + \lambda_L) > 0.9999. \tag{62}$$

Then, we can consider the corresponding eigenvalues $q_1, \ldots, q_K$, and their scaled

$$p_1 = \sqrt{\lambda_1} q_1, \ldots, p_K = \sqrt{\lambda_K} q_K \tag{63}$$

counterparts. The vectors $p_1, \ldots, p_K$ can then be used as the columns for a matrix P_K, and with this, we can approximate $\Sigma_{f_i} \approx P_K (P_K)^t$ and now, $\mathrm{f}_i = \mu_{f_i} + P_k \beta, \;\; \beta \sim \mathcal{N}(0, I_K)$.

Problem 2

The number of grid points grows as an exponential function with respect to the total number of variables in the input space. As an example, if the input space turns out to be three-dimensional for one of the functions in our AM and we use fifty grid points in each direction, we would need $50^3 = 125000$ points in total.

Storing the high-dimensional covariance matrices that arise is impossible for most computing systems, and what's even worse, one usually cannot employ methods from sparse matrix algebra: most of the typically used kernel functions result in dense covariance matrices.

Solution to Problem 2

Use separable kernel functions and build big covariance matrices out of smaller covariance matrices using the Kronecker product. For example if

$$k(x,x') = k_1(x_1,x_1')k_2(x_2,x_2'), \quad x = (x_1,x_2), \quad x' = (x_1',x_2'), \tag{64}$$

we can calculate the one-dimensional covariance matrices

$$\Sigma_1 \in \mathbb{R}^{m_1 \times m_1} \text{ and } \Sigma_2 \in \mathbb{R}^{m_2 \times m_2} \tag{65}$$

that are associated to the kernels k_1 and k_2, and then the covariance matrix in the joint space is simply the Kronecker product of the two individual covariance matrices, i.e.

$$\Sigma = \Sigma_1 \otimes \Sigma_2 \in \mathbb{R}^{m_1 m_2 \times m_1 m_2}, \tag{66}$$

under a suitable ordering of the variables.

Solution to Problem 2 Cont.

Definition

Let A be an $m \times n$ matrix and B a $p \times q$ matrix. Then, the Kronecker product of A and B is the $pm \times qn$ block matrix

$$A \otimes B = \begin{pmatrix} A_{11}B & \cdots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{m1}B & \cdots & A_{mn}B \end{pmatrix}. \tag{67}$$

Alternatively, suppose $\star$ and $\diamond$ denote truncating integer division and remainder, respectively. Then,

$$(A \otimes B)_{ij} = A_{i \star p, j \star q} B_{i \diamond p, j \diamond q}, \tag{68}$$

assuming that as a convention, we number the matrix elements starting from 0.

Solution to Problem 2 Cont.

We also see that if $\Sigma_1 = Q_1 \Lambda_1 Q_1^t$ and $\Sigma_2 = Q_2 \Lambda_2 Q_2^t$ are eigendecompositions for Σ_1 and Σ_2, respectively, then $\Sigma = \Sigma_1 \otimes \Sigma_2$ has the below eigendecomposition:

$$\Sigma = (Q_1 \Lambda_1 Q_1^t) \otimes (Q_2 \Lambda_2 Q_2^t) = (Q_1 \otimes Q_2)(\Lambda_1 \otimes \Lambda_2)(Q_1 \otimes Q_2)^t. \tag{69}$$

Thus, the eigenvectors of Σ are just Kronecker products of the individual eigenvectors of Σ_1 and Σ_2, while the eigenvalues of Σ are just products of the eigenvalues for Σ_1 and Σ_2 (after a suitable ordering of vectors and values).

An Example

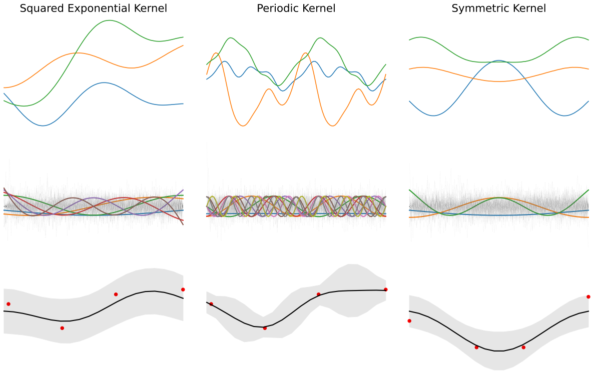


Figure: Random samples (top row), basis vectors (middle row), and simple fitting examples (bottom row), for $k_{\text{SE}}(x, x')$ (first column), $k_{\text{PER}}(x, x')$ (second column), $k_{\text{SYM}}(x, x')$ with respect to $k_{\text{SE}}(x, x')$ (third column). The dashed lines in the second row plots show the basis functions that are dropped out from the estimation. The final row compares the mean and $\pm$ two standard deviations calculated from the reduced basis vectors.

B-spline Basis

To define a B-spline basis with k dimensions, we first define $k+m+2$ knots, $x_1 < x_2 < \cdots < x_{k+m+1} < x_{k+m+2}$, where the interval for spline evaluation is $[x_{m+2}, x_{k+1}]$, and the first and last $m+1$ knot locations are essentially arbitrary. Then, an $(m+1)$-th order spline will be

$$f(x) = \sum_{i=1}^{k} B_i^m \beta_i, \tag{70}$$

where the B-spline basis functions are defined recursively in the following way:

$$B_i^{-1}(x) = \begin{cases} 1 & x_i \leq x < x_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{71}$$

$$B_i^m(x) = \frac{x - x_i}{x_{i+m+1} - x_i} B_i^{m-1}(x) + \frac{x_{i+m+2} - x}{x_{i+m+2} - x_{i+1}} B_{i+1}^{m-1}(x), \tag{72}$$

for $i = 1, \ldots, k$.

Difference Priors

Eilers and Marx (1996) employed B-splines along with difference priors. This setup is a low rank smoother using a B-spline basis, usually defined on evenly spaced knots, along with a difference prior applied directly to the parameters β_i to control function wiggliness.

Difference Priors Example

Suppose we decide to penalize the squared difference between adjacent β_i values, so that $\mathcal{P} = \sum_{i=1}^{k-1}(\beta_{i+1} - \beta_i)^2 = \beta^t P^t P \beta$ is the penalty, where the matrix P is defined so that the following equalities hold:

$$P = \begin{pmatrix} -1 & 1 & 0 & \cdot & \cdot \\ 0 & -1 & 1 & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}, \quad P\beta = \begin{pmatrix} \beta_2 - \beta_1 \\ \beta_3 - \beta_2 \\ \cdot \\ \cdot \end{pmatrix} \tag{73}$$

$$\mathcal{P} = \beta^t \begin{pmatrix} 1 & -1 & 0 & \cdot & \cdot \\ -1 & 2 & -1 & \cdot & \cdot \\ 0 & -1 & 2 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \beta. \tag{74}$$

Difference Priors in 1D

In the framework before, X will be a sparse linear interpolation matrix that sends the function values (at the chosen input grid) to the observation locations. An example is

$$X = \begin{pmatrix} 0 & \cdots & w_1 & 1-w_1 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & w_2 & 1-w_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}, \tag{75}$$

where w_i is the distance of the input observation to the first input grid value that it exceeds. In other words, if the input grid is $x_g = (x_{g,1}, \ldots, x_{g,L})$, then X is just

$$X_{ij} = \begin{cases} x_i - x_{g,j} & x_{g,j} \leq x_i \leq x_{g,j+1}, \\ 1 - (x_i - x_{g,j}) & x_{g,j-1} \leq x_i \leq x_{g,j}, \\ 0 & \text{otherwise.} \end{cases} \tag{76}$$

Difference Priors in 1D Cont.

If we approximate, in a grid, the derivatives of the functions, we will get sparse difference matrices. Examples for the first three orders of difference matrices for a one-dimensional input grid of length $L = 5$ is shown (as D_1, D_2, and D_3) below:

$$D_1 = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}, \tag{77}$$

$$D_2 = \begin{pmatrix} -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \end{pmatrix}, \quad D_3 = \begin{pmatrix} -1 & 3 & -3 & 1 & 0 \\ 0 & -1 & 3 & -3 & 1 \end{pmatrix}. \tag{78}$$

Choosing Difference Prior Orders

In general, while all choices of order shall give similar fits for the data range, the extrapolation behavior would be different. For instance, Figure 5 (next slide) displays some example using first, second, and third order differences for a simple function approximation task. Beyond the data range, the first order prior extrapolates to a constant, the second order prior extrapolates to a linear function, and the third order prior extrapolates to a quadratic function.

Choosing Difference Prior Orders Cont.

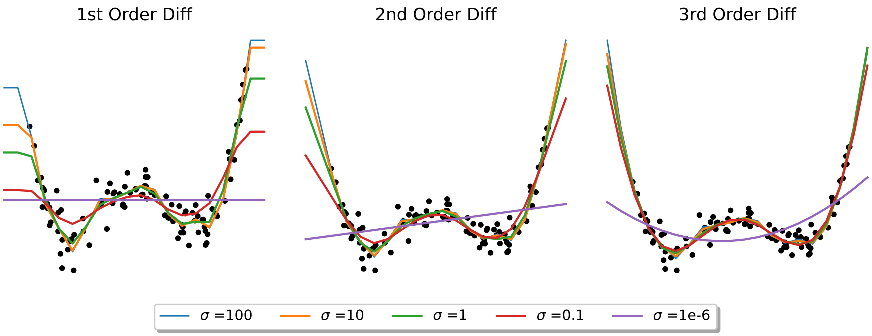


Figure: Fitting data generated around the true function $f(x) = 3x^4 - 6x^2 + 2$ using 3 different orders for the difference priors and 5 different prior variances (see the legend).

High Dimensional Fits

Solution: use tensor product basis! If

$$f_x(x) = \sum_{i=1}^{I} \alpha_i a_i(x),\ \ f_z(z) = \sum_{\ell=1}^{L} \delta_\ell d_\ell(z),\ \ f_v(v) = \sum_{k=1}^{K} \beta_k b_k(v)$$

are basis representations, then the three-variable function $f_{xzv}(x, z, v)$ can be represented by

$$f_{xzv}(x, z, v) = \sum_{i=1}^{I} \sum_{\ell=1}^{L} \sum_{k=1}^{K} \beta_{i\ell k} b_k(v) d_\ell(z) a_i(x).$$

High Dimensional Fit Example

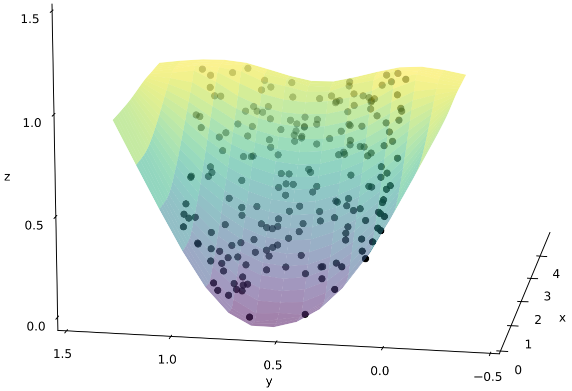


Figure: Fitting data generated around the true function $f(x, y) = 0.5x + 4(y - 0.5)^2/(1 + 2x)$ using a second order difference prior for x and a third order difference prior for y.

Spatially Varying Smoothness

One benefit of this local basis approach is that it is possible to tune prior variances of the difference priors as a way to achieve “spatially varying smoothness.” By this term, we mean that when estimating a function which has nonconstant smoothness, specifying a prior variance that is suitable for one region of the function might work poorly for different regions of the function. The solution is to “spatially vary” the smoothness by assigning different prior variances for different parts of the function.

Spatially Varying Smoothness Cont.

This phenomenon can be seen clearly, for example, in the three plots of Figure 7 that are shown below.

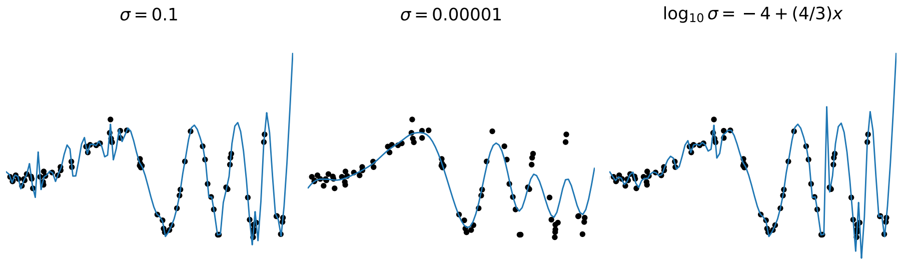


Figure: Fitting data generated around the true function $f(x) = \sin(x^3)$ using 3 different ways of defining the prior variance. For the first two plots, the prior variance is fixed at a constant value, while in the third/last plot, the prior variance is made a function of x.

Spatially Varying Smoothness Cont.

In the figure above, the goal was to fit points around the true function given by

$$f(x) = \sin(x^3), \tag{79}$$

which clearly has nonconstant smoothness. We see that increasing the variance of the prior via $\log_{10} \sigma = -4 + (4/3)x$ as a function of x yielded a better fit to the data compared with keeping the prior variance fixed, as expected. Of course, one thing of note is that we have chosen the behavior of the prior variance manually such that the fit looks good, and automatically estimating how the prior variance should vary with each dimension is impossible in general.

Sudden Jumps

Moreover, one particular case where spatially varying smoothness will dramatically improve the resulting fit of the additive model is when there is a sudden jump in the function. This type of behavior can be induced by setting the prior variance to be high in the vicinity of the discontinuity. For an example, consider the function f with sudden jump

$$f(x) = \sin(x) + xI(x > 2) \tag{80}$$

at $x = 2$.

Sudden Jumps Cont.

We can fit two additive models with B-spline bases as well as difference priors, the first one with constant smoothness and the second one with varying smoothness. The results of this fit are displayed below in Figure 8.

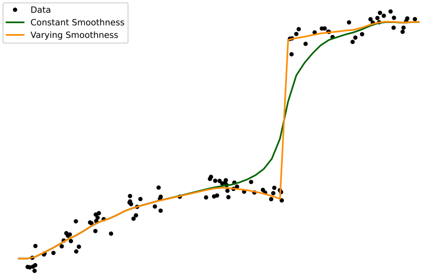


Figure: Fitting data generated around the true function $f(x) = \sin(x) + xI(x > 2)$ in 2 different ways: fixing the prior variance vs. increasing the prior variance around $x = 2$.

Experimentation

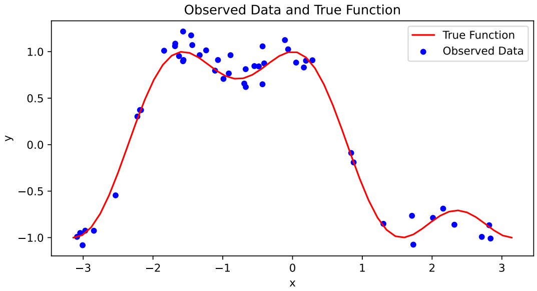
For our application, we we will implement a spline model from Section 2, as well as special cases for the models introduced in Section 3 and Section 4, along with benchmark models (in our case, linear regression and random forest regression) to 3 different datasets:

- Toy Function
- Univariate Data from a Simulated Motorcycle Accident
- Multivariate Data from the Beijing Multi-Site Air-Quality Data Set

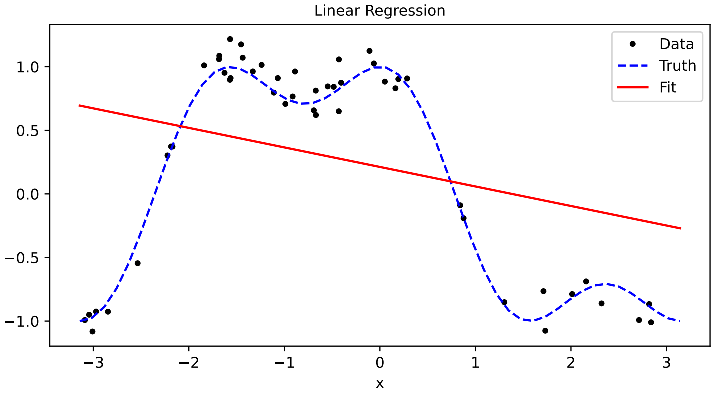
Toy Function

Introduction

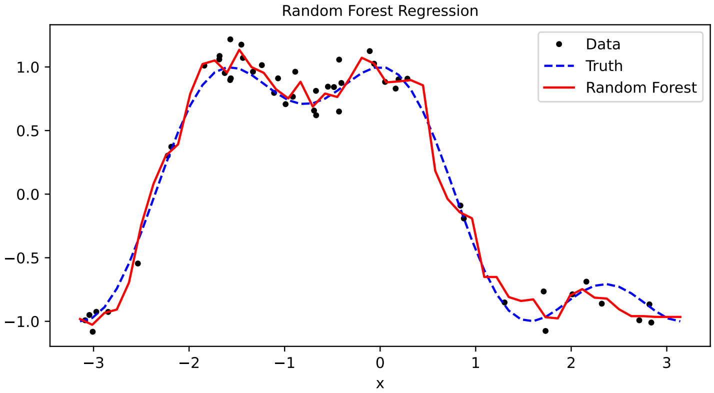
The function we are interested in is $-\sin^3(x) + \cos^3(x)$.
We train our models on 50 noised observations. The exact function and the “noised” data points are visualized below:



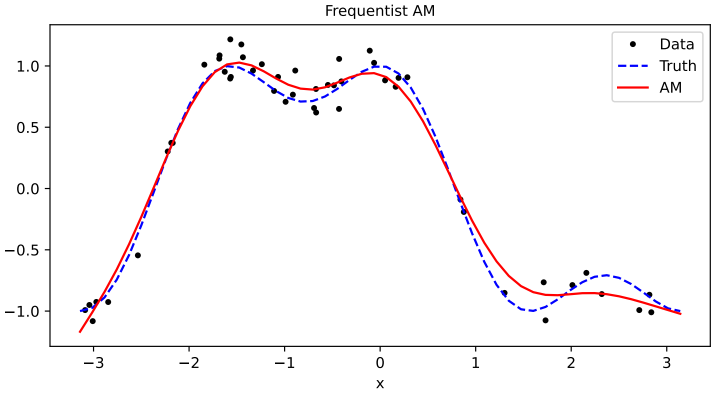
Linear Regression Result



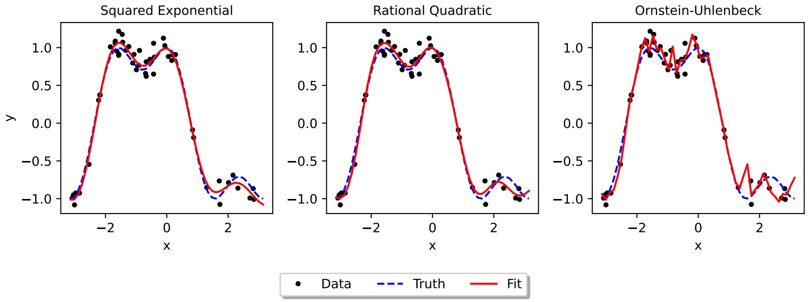
Random Forest Result



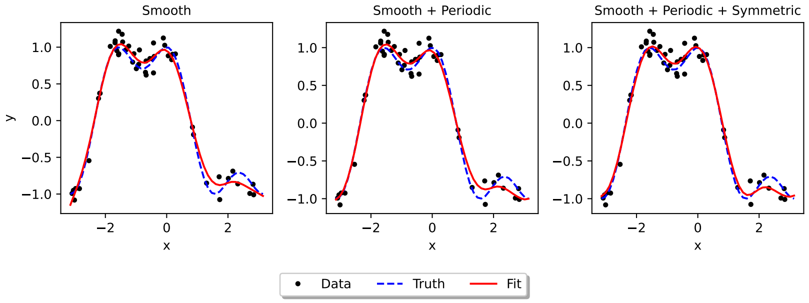
Frequentist Penalized GAM Result



Gaussian Process Priors with Different Kernels Result



Difference Priors Result



Overall Result

Model	MSE	RMSE	MAE	R^2
LR	0.56602	0.75235	0.66527	0.10453
AM	0.00902	0.09496	0.07896	0.98573
GP-SE	0.00301	0.05483	0.04816	0.99524
GP-RQ	0.00401	0.06329	0.05389	0.99366
GP-OU	0.01922	0.13864	0.09894	0.96959
DP-S	0.00674	0.08209	0.06725	0.98934
DP-P	0.00690	0.08304	0.06522	0.98909
DP-M	0.00431	0.06568	0.05368	0.99318
RF	0.01208	0.10993	0.09154	0.98088

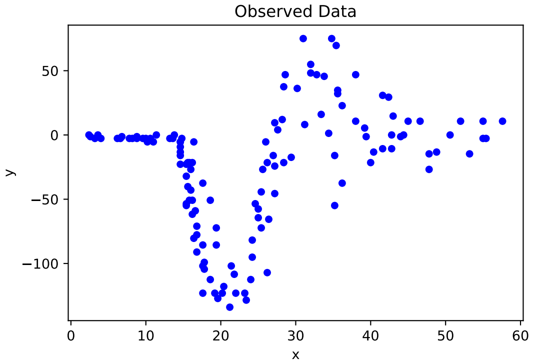
Analysis

- As expected, linear regression performs the worse out of all the models we tested since the data is nonlinear.
- Gaussian process prior with squared exponential kernel and rational quadratic kernel are among the best performing model, which makes sense since the true function is smooth.
- Interestingly, random forest is the second worse performing algorithm, although it is already able to capture the shape of the function really well, followed by Frequentist AM. This may be caused by the generality of the methods, which makes them perform worse than models that use infinitely differentiable functions (Gaussian process priors).
- Difference priors perform pretty well, the second best after Gaussian process priors.

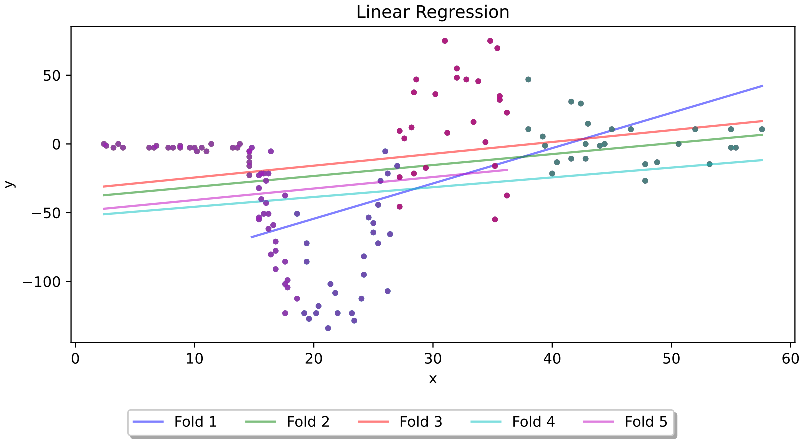
Univariate Data from a Simulated Motorcycle Accident

Overview

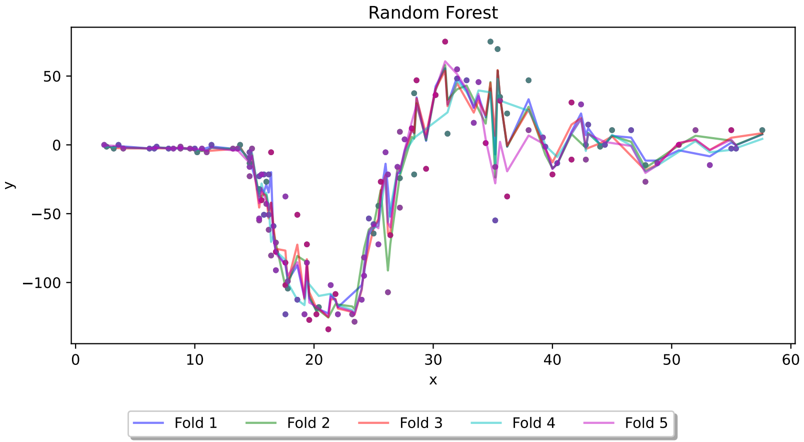
Here, we are trying to fit a function through the following points:



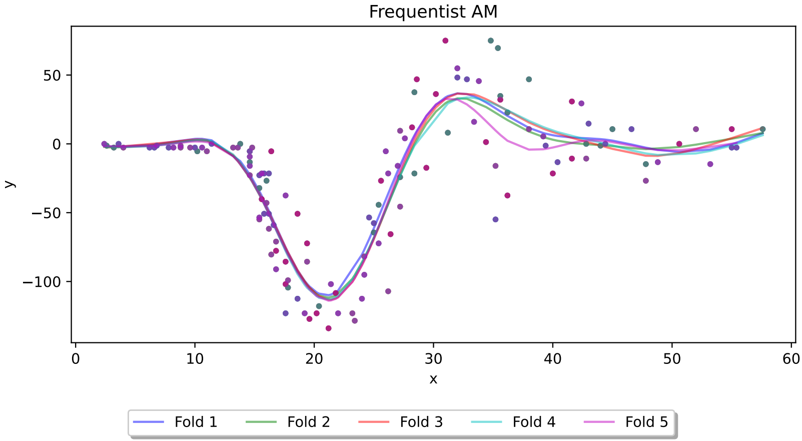
Linear Regression Fit



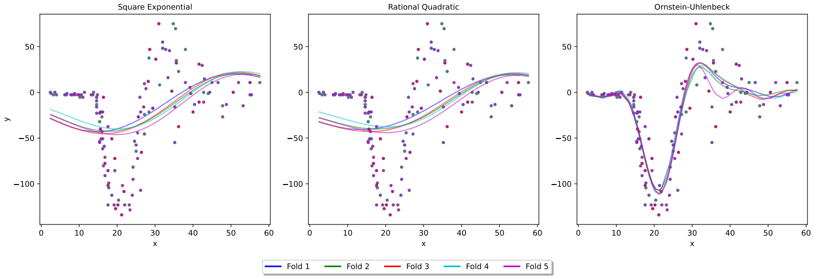
Random Forest Fit



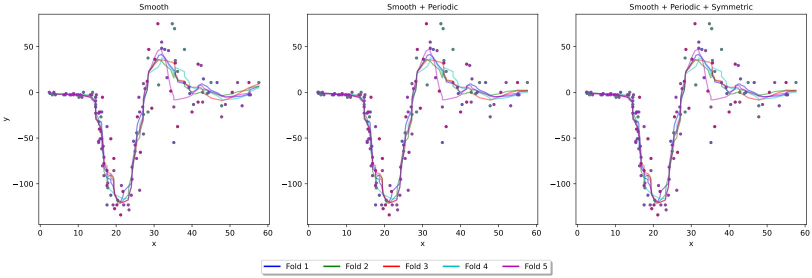
Frequentist Penalized GAM Title



Gaussian Process Priors with Different Kernels Result



Difference Priors Result



Overall Result

Model	MSE	RMSE	MAE	R^2
LR	1962.3170	44.05878	35.97921	0.10951
AM	476.69544	21.81958	16.57895	0.79347
GP-SE	1762.6589	41.96837	34.64343	0.23659
GP-RQ	1826.0479	42.71584	35.01356	0.20919
GP-OU	487.54859	22.06440	16.87112	0.78877
DP-S	379.33748	19.45626	13.71573	0.83592
DP-P	379.99529	19.47301	13.78276	0.83563
DP-M	379.99529	19.47301	13.78276	0.83563
RF	205.31152	14.32060	9.777730	0.91103

Analysis

- As expected, linear regression performs the worst out of all the models we tested because this dataset is again nonlinear.
- Random Forest is the best-performing model, followed by Difference prior models, whose local flexibility allowed for a decent fitting of the data.
- For Difference priors, we note that the periodic and symmetric properties do not influence much of the evaluation result, indicating a smooth function is enough to fit the data.

Cont. Analysis

- Interestingly, Gaussian process prior models are the second worst-performing algorithm, although it is the best-performing algorithm in the toy function case.
- This can be explained by the fact that the underlying true function is probably nonlinear and nonsmooth, hence it is not differentiable at some point, causing it unable to estimate the function well.
- However, we note that Gaussian process with Ornstein-Uhlenbeck kernel performs exceptionally well among other kernels, which makes sense because $k_{\mathsf{OU}}(x, x')$ kernel does not assume as much smoothness of the function, compared to squared exponential kernel and rational quadratic kernel.

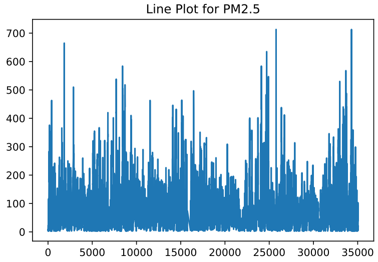
Multivariate Data from the Beijing Multi-Site Air-Quality Data Set

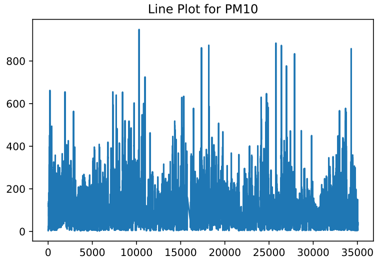
Overview

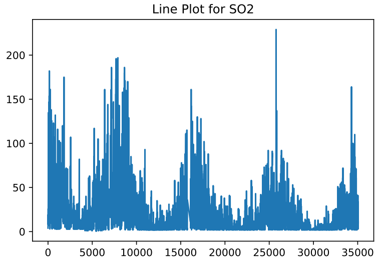
The given dataset has 12 subdatasets, corresponding to the 12 air quality collection sites in Beijing. In this experiment, we are considering data collected at Aoti Zhongxin (or commonly known as Olympic Park) in 2013 - 2017.

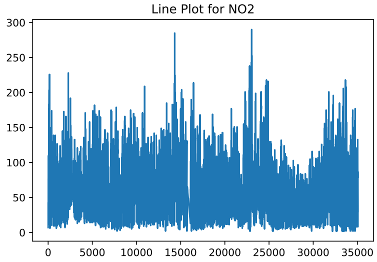
- The AotiZhongxin dataset contains 35064 rows and 9 columns.
- The columns are: PM2.5, PM10, SO2, NO2, CO, O3, PRES, RAIN, WSPM.
- There are several null/NaN values in the dataset: 925 in PM2.5m, 718 in PM10, 935 IN SO2, 1023 in NO3, 1776 in CO, 1719 in O3, 20 in PRES, 20 in RAIN, and 14 in WSPM.
- We dropped all rows with null/NaN values, and dropping time indicator columns and string columns, resulted in a dataframe of 31815 rows and 9 columns.

Exploratory Data Analysis- Line Plots

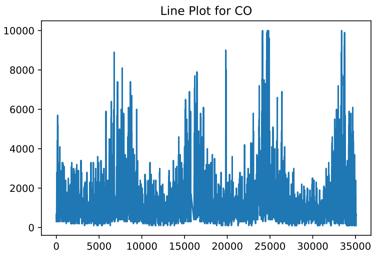


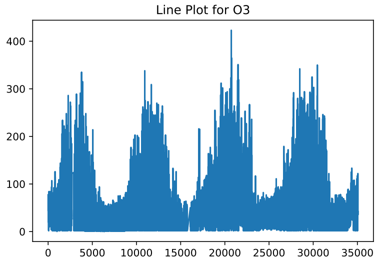


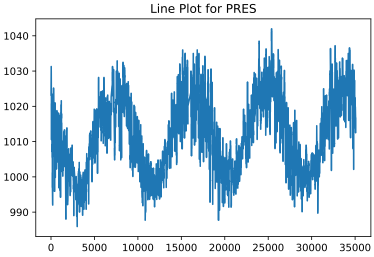


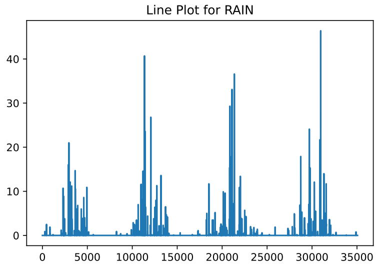


Cont. Exploratory Data Analysis- Line Plots

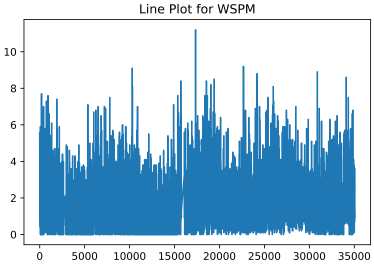


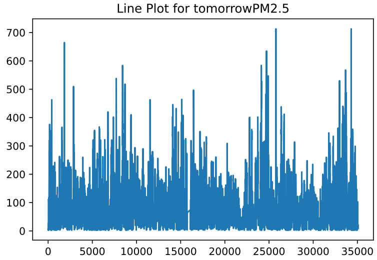




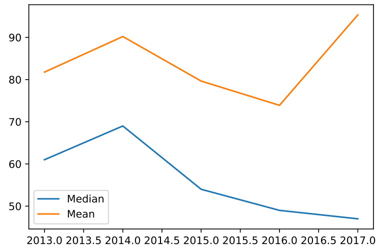


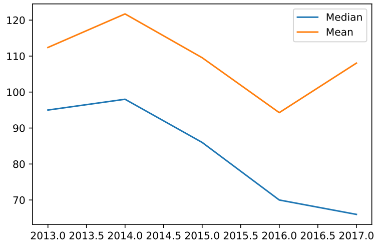
Cont. Exploratory Data Analysis- Line Plots

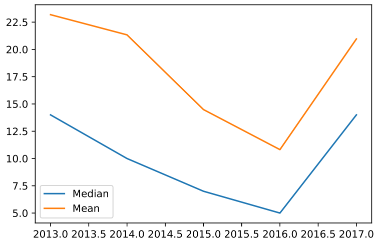


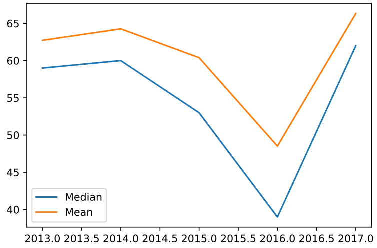


(2) Exploratory Data Analysis- Mean & Median

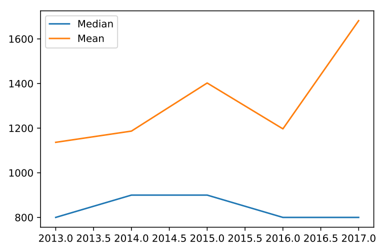


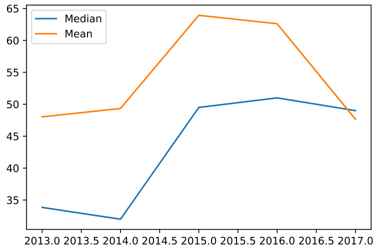


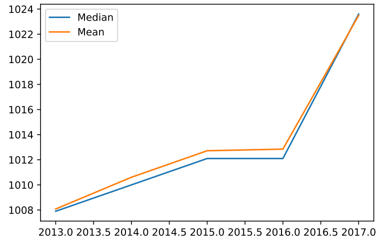


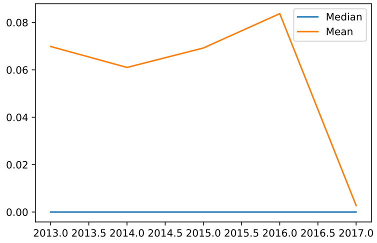


Cont. Exploratory Data Analysis- Mean & Median

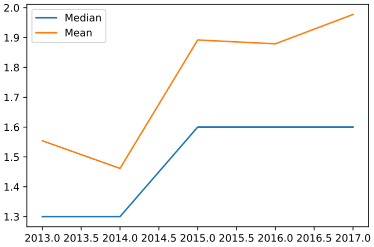


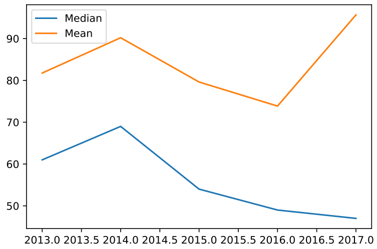




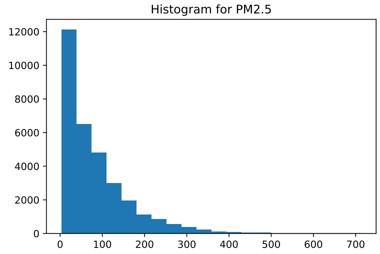


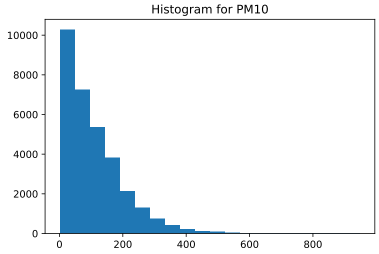
Cont. Exploratory Data Analysis- Mean & Median

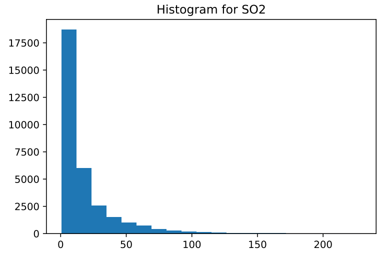


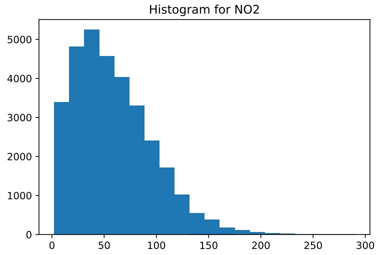


(3) Exploratory Data Analysis- Histogram

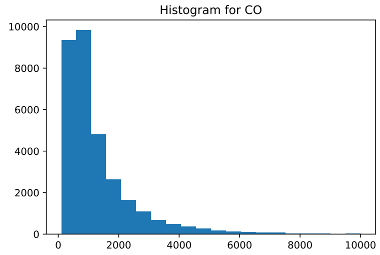


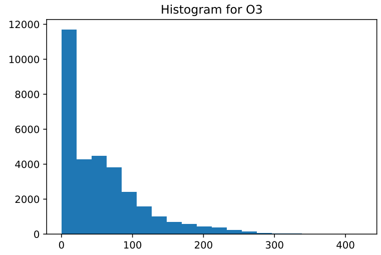


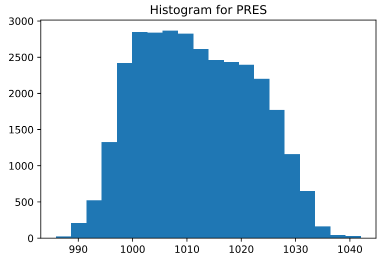


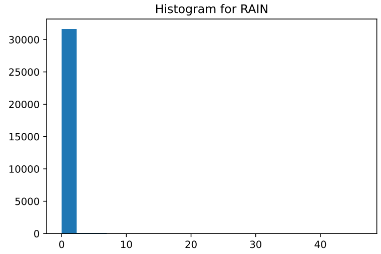


Cont. Exploratory Data Analysis- Histogram

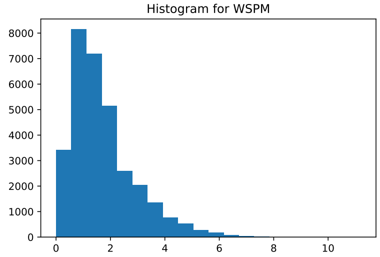


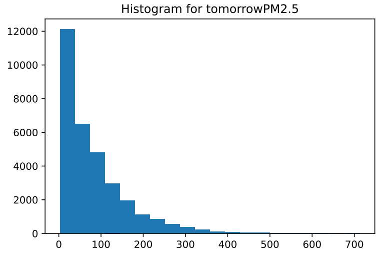




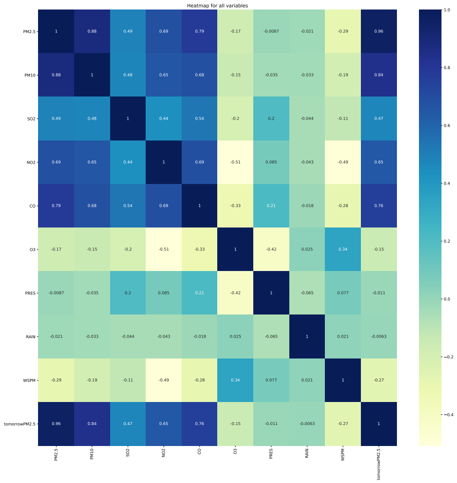


Cont. Exploratory Data Analysis- Histogram





Heatmap of different results



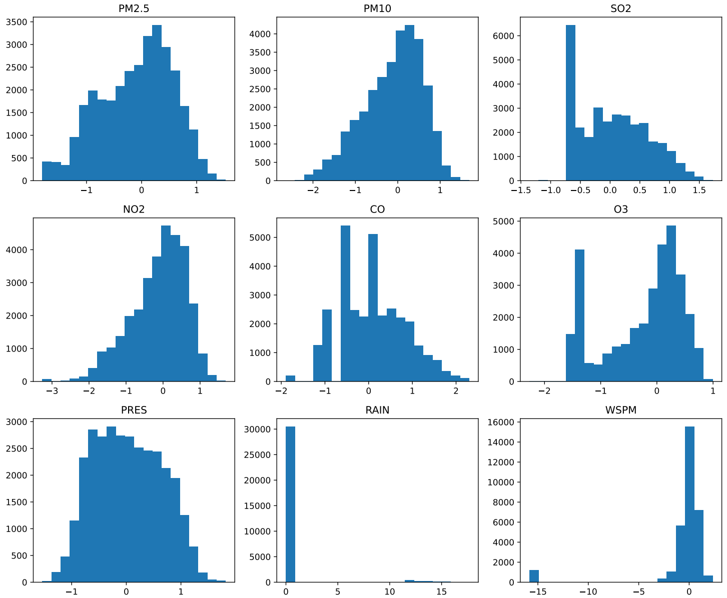
Miscellaneous

- As observed from the histogram, PM2.5 tomorrow (the predictor) is very right-skewed. Hence, we performed a log normalization to transform the data.
- As observed from the histogram, we note the wide range between each covariate, as well as outliers. Hence we standardize our data with Robust scaler, which has formula

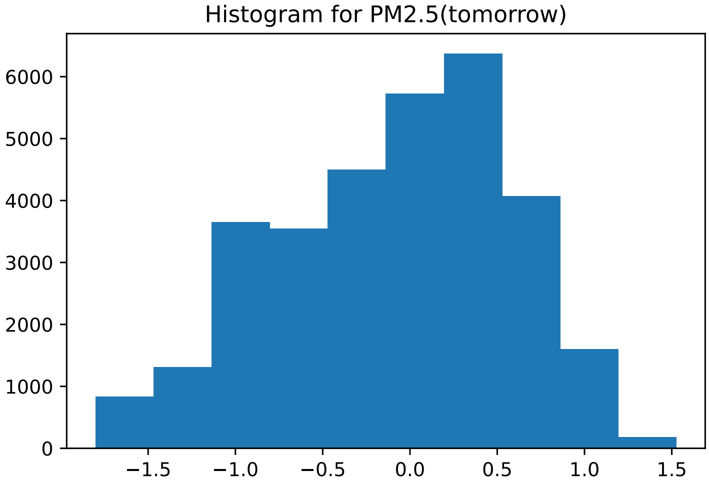
$$\frac{x_i - Q_{50}(x)}{Q_{75}(x) - Q_{25}(x)}, \quad i = 1, \dots, n, \quad x = (x_1, \dots, x_n).$$

- We perform a train-test on the dataset, assigning 30% of the dataset as a test set, resulting in a training set with a shape of (22270, 10) and a testing set with a shape of (9545,10).

Scaled & Standardized Data



Scaled & Standardized Data - Predictor



Overall Result on Training Data

Model	MSE	RMSE	MAE	R^2
LR	0.041791	0.204429	0.124798	0.906273
AM	0.039135	0.197827	0.120681	0.912229
GP-ES	0.039246	0.198106	0.121153	0.911982
GP-RQ	0.039189	0.197963	0.121193	0.912108
GP-OU	0.039210	0.198014	0.121232	0.912063
DP-S	0.038189	0.195420	0.121443	0.914352
DP-P	0.038189	0.195420	0.121439	0.914352
DP-SP	0.038190	0.195422	0.121457	0.914350
RF	0.031482	0.177432	0.111289	0.929393

Overall Result on Testing Data

Model	MSE	RMSE	MAE	R^2
LR	0.043389	0.208299	0.125495	0.903937
AM	0.040531	0.201322	0.121427	0.910265
GP-ES	0.040964	0.202397	0.122669	0.909305
GP-RQ	0.040988	0.202456	0.122855	0.909252
GP-OU	0.040953	0.202369	0.122749	0.909330
DP-S	0.041526	0.203780	0.125011	0.908060
DP-P	0.041588	0.203932	0.125091	0.907924
DP-M	0.041505	0.203728	0.124995	0.908108
RF	0.040227	0.200567	0.121238	0.910937

Analysis

- As predicted, linear regression performs the worst both on the train and the test dataset since the data is nonlinear. But since it does not lag too far behind the other models, the data might just be "slightly" nonlinear.
- From the train to the test dataset, we observe that Random Forest with Bayesian optimization overfitted by the most, achieving 0.031 MSE in training, and drastically increased to 0.040 in testing.
- The three Gaussian process prior models seemed to perform almost equally, with GP with rational quadratic kernel best in training but Ornstein-Ulhenback kernel best in testing.

Cont. Analysis

- For Difference priors with a mixture of Smooth + Periodic functions, we handpicked some covariates to be assigned a smooth difference prior and others to be assigned a periodic difference prior. Here, our reasoning for these assignments are from the exploratory data analysis.
- Overall, difference prior models perform better than Gaussian process prior models in training, but worse in testing. Hence, this indicates that the local approach may be more prone to overfitting compared to a global approach.

Conclusion

Conclusion

Throughout the analysis, we observe **clear evidence the initial three advantages of AMs**:

- **Flexibility**: We can select different kernels and functions to fit each of our covariates based on exploratory data analysis, achieving better results.
- **Interpretability**: Based on our selection of kernels/functions, we can very well understand the effect of our model based on explicit mathematical notation as described, unlike machine learning model.
- **Regularization**: We are able to control smoothness explicitly to fit the data.

Cont. Conclusion

In conclusion, we have proposed and discussed a few additive models, built upon frequentist and Bayesian methods, that achieves impressive fits on highly nonlinear data all the while still maintaining high levels of interpretability relative to other popular machine learning models used today.

Acknowledgements

This paper and presentation are written as a final project for Statistics 220: Bayesian Data Analysis, a class both authors are enrolled in under the instruction of Professor Jun Liu and Teaching Fellow Buyu Lin.

Thank you