Spike-and-Slab Generalized Additive Models and Fast Algorithms for High-Dimensional Data

Boyi Guo

Department of Biostatistics
University of Alabama at Birmingham

August 8th, 2021

Outline

Outline

- Background
- Objectives
- Spike-and-slab GAM
  - Natural parameterization
  - Spike-and-slab Spline Prior
  - Algorithms
  - Simulations
- R packge: `BHAM`

Background

Non-linear Effect Modeling

- Simple solutions
  - Variable categorization
    - Unrealistic assumption, loss of power
  - Polynomial regression
    - Numerically unstable, inflexible
- Machine learning methods
  - Classification and regression tree, random forests, neural network
  - Lack of interpretation
  - Hard to extend to high-dimensional setting

Generalized Additive Model

Firstly proposed by Hastie and Tibshirani (1987)

- Formulation

$$y_i \overset{\text{i.i.d.}}{\sim} EF(\mu_i, \phi), \quad i = 1, \ldots, n$$

$$\mu_i = g^{-1}(a + \sum_{j=1}^{p} f_j(x_{ij}))$$

where $g(\cdot)$ is a known link function, $f_j(\cdot)$ is a smoother function

- Objective: to estimate smoother functions $f_j(\cdot)$
- We limit the smoother functions $f_j(x)$ in the class of spline functions

Smoothing Spline Model

Given a univariate spline model $y_i \overset{\text{i.i.d.}}{\sim} N(\beta^T \boldsymbol{B}(x_i), \sigma^2)$[1], we are interested in estimate β via penalized least square estimation

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^K} \sum_{i=1}^{n} \left[y_i - \beta^T \boldsymbol{B}(x_i) \right]^2 + \lambda \int f''(\boldsymbol{X})^2 dx$$

- Smoothing penalty $\lambda \int f''(X)^2 dx = \lambda \beta^T S \beta$
- The smoothing parameter λ is a tuning parameter, selected via cross-validation
- The matrix S is referred as wiggliness matrix
  - Given the data matrix $\boldsymbol{X}$, S is known
  - S differs for different spline representations
  - S is symmetric and positive semi-definite

[1] Generalizable to GLM

High-dimensional Data Analysis

High-dimensional data analysis refers to the analysis when the number of predictors p is equal or greater than the sample size n

- Classic statistics methods are infeasible due to ill-posed model fitting algorithm
- Conventional solutions
  - Penalized Models

$$Q_{Penalized}(\beta) = (y - \boldsymbol{X}\beta)^T(y - \boldsymbol{X}\beta) + P_\lambda(\beta).$$

  - Bayesian Hierarchical Models
    - Spike-and-slab priors

Spike-and-Slab Priors

- First coined by Mitchell and Beauchamp (1988)
- A mixture distribution conditioning on a latent binary indicator $\gamma \in \{0, 1\}$

$$\beta|\gamma \sim (1-\gamma)f_{spike}(\beta) + \gamma f_{slab}(\beta)$$

  - $f_{spike}(x)$: a spike density concentrate around 0 for small effects
  - $f_{slab}(x)$: a flat density for large effects
  - γ follows a Bernoulli distribution with probability θ
- Advantages:
  - Simultaneous variable selection and prediction
  - Robust estimation
  - Local adaptive
- Examples:
  - Stochastic search variable selection (George and McCulloch 1993)
  - Spike-and-slab lasso (Ročková and George 2018)

HD Spline Model

- HD Spline model inherits the statistical difficulties from HDA
  - Function selection instead of variable selection
  - Computational burden
- Unique challenges
  - Balance between smoothing penalty and sparse penalty
  - Global adaption VS Local adaption
  - Grouped nature of spline bases
  - Linear VS non-linear effect
- Previous solutions
  - Grouped Penalized Model
  - Bayesian Group models
    - Spike-and-slab normal-mixture-of-inverse gamma prior (Scheipl, Fahrmeir, and Kneib 2012)
    - Spike-and-slab group lasso (Bai 2020)

Objective

- To develop statistical models that allow non-linear effect modeling in high-dimensional data analysis
  - Focus on simultaneous variable selection and prediction
- To develop fast computing algorithms for proposed models
- To develop statistical software for proposed models

Objectives

Spike-and-Slab GAM

Model

Given the data $\{\boldsymbol{X}_i, y_i\}_{i=1}^n$ where $\boldsymbol{X}_i \in \mathbb{R}^p$ $y_i \in \mathbb{R}$ and $p >> n$, we have the genralized additive model

$$y_i \overset{\text{i.i.d.}}{\sim} EF(\mu_i, \phi),$$

$$g(\mu_i) = \sum_{j=1}^{p} f_j(x_{ij}), \quad i = 1, \dots, n.$$

We express the linear predictor in the matrix form using natural parameterization

$$g(\mu_i) = \sum_{j=1}^{p} \left[{\beta_j^0}^T X_{ij}^0 + {\beta_j^{pen}}^T X_{ij}^{pen} \right].$$

We propose two sets of spike-and-slab priors

- Mixture normal priors
- Mixture double exponential priors for ultra-high dimension

Natural Parameterization

- Linear space of the spline will not be penalized
  - the second derivative $x^{''} = 0^2$:
- Eigen-decomposition of S
  - $\boldsymbol{S} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T$
  - $U \equiv [\boldsymbol{U}^{pen} : \boldsymbol{U}^0]$ and $\boldsymbol{D} \equiv [\boldsymbol{D}^+ : \boldsymbol{0}]$
  - $\boldsymbol{X}\beta = \boldsymbol{X}\boldsymbol{U}\boldsymbol{U}^T\beta = X_0\beta_0 + X_{pen}\beta_{pen}$
  - $\beta^{pen} \sim MVN_{K^*}(\boldsymbol{0}, \boldsymbol{I})^3$
- Benefits
  - Isolate linear part from the polynomial part of the spline functions
  - Independent prior for the penalized part

[2] For simplicity we assume the null space is 1-dimensional. It is trivial to generalized to multi-dimensional

[3] requires extra step of scaling based on $\boldsymbol{D}^+$

Mixture Normal Priors

For the coefficients of spline matrices X_j^0 and X_j^β of the predictor x_j

$$
\begin{aligned}
\beta_j^0|\gamma_j^0, s_0, s_1 &\sim N(0, (1-\gamma_j^0)s_0 + \gamma_j^0 s_1) \\
\beta_{jk}^{pen}|\gamma_j^{pen}, s_0, s_1 &\sim N(0, (1-\gamma_j^{pen})s_0 + \gamma_j^{pen} s_1), k = 1, \ldots, K_j^*, \\
\gamma_j^0|\theta_j^0 &\sim Bin(\gamma_j^0|1, \theta_j^0) \\
\gamma_j^{pen}|\theta_j &\sim Bin(\gamma_j^{pen}|1, \theta_j), \\
\theta_j^0 &\sim Beta(a, b) \\
\theta_j &\sim Beta(a, b).
\end{aligned}
$$

Mixture Normal Priors

- All coefficients of penalized space β_{jk}^{pen} share the same binary indicator γ_j^{pen}
- Each predictor has two binary indicators γ_j^0 and γ_j^{pen} controlling the inclusion of linear effects and non-linear effects respectively
  - $\gamma_j^0 = 1$, $\gamma_j^{pen} = 1$: non-linear effect exists
  - $\gamma_j^0 = 1$, $\gamma_j^{pen} = 0$: only linear effect exists
  - $\gamma_j^0 = 0$, $\gamma_j^{pen} = 0$: no effects
- θ_j^0 and θ_j are the corresponding inclusion probability
  - The hyper-prior of θs allow for local adaption of shrinkage
  - θ_j controls the smoothness of the polynomial function
  - θ_j^0 controls the sparsity of signals in the model

Mixture Double Exponential Priors

Similarly, we replace the mixture normal prior of βs with the mixture double exponential prior for sparse solution

$$
\begin{aligned}
\beta_j^0 | \gamma_j^0, s_0, s_1 &\sim \boxed{DE(0, (1-\gamma_j^0)s_0 + \gamma_j^0 s_1)} \\
\beta_{jk}^{pen} | \gamma_j^{pen}, s_0, s_1 &\sim \boxed{DE(0, (1-\gamma_j^{pen})s_0 + \gamma_j^{pen} s_1)}, k = 1, \dots, K_j^* \\
\gamma_j^0 | \theta_j^0 &\sim Bin(\gamma_j^0 | 1, \theta_j^0) \\
\gamma_j^{pen} | \theta_j &\sim Bin(\gamma_j^{pen} | 1, \theta_j), \\
\theta_j^0 &\sim Beta(a, b) \\
\theta_j &\sim Beta(a, b).
\end{aligned}
$$

Fast Computing Algorithms

We are interested in estimate $\Theta = \{\boldsymbol{\beta}, \boldsymbol{\theta}, \phi\}$

- Prevous Bayesian methods relies on computationally intensive MCMC algorithms
  - Infeasible in high-dimensional setting
- Fast computing EM-based algorithms are proposed
  - Previously used in high-dimensional variable selection
  - Shows great success
- We expand the algorithms to fit the aforementioned models
  - EM - Iterative weighted least square
    - Uncertainty inference
  - EM - Coordinate descent algorithm
    - Sparse Solution and faster computation

EM algorithm

EM algorithm is an iterative algorithm to find local maximum a posterori estimates

- Treat nuisance parameters as “missing values”
- Calculate the expectation of the posterior density with respect to the missing values
- Maximize the expectation to estimate the parameters
- Iterate the process until convergence

EM algorithm

We aim to maximize the log posterior density of Θ by treating binary indivators γ as "missing"

$$
\begin{aligned}
Q(\Theta, \gamma) &\equiv \log p(\Theta, \gamma | \mathbf{y}, \mathbf{X}) \\
&= \log p(\mathbf{y}|\beta, \phi) + \log p(\phi) + \sum_{j=1}^{p}\left[\log p(\beta_j^0|\gamma_j^0) + \sum_{k=1}^{K_j} \log p(\beta_{jk}^{pen}|\gamma_{jk}^{pen})\right] \\
&+ \sum_{j=1}^{p}\left[(\gamma_j^0)\log\theta_j^0 + (1-\gamma_j^0)\log(1-\theta_j) + (\gamma_j^{pen})\log\theta_j + (1-\gamma_j^{pen})\log(1-\theta_j)\right] \\
&+ \sum_{j=1}^{p}\log p(\theta_j) + \sum_{j=1}^{p}\log p(\theta_j^0)
\end{aligned}
$$

EM algorithms

- E-step
  - Formulate $E_{\gamma|\Theta^{(t)}}\left[Q(\Theta,\gamma)\right]$
  - Calcualte $E(\gamma_j^0)$ and $E(\gamma_j^{pen})$ by Bayes' theorem
- M-step:
  - maximize $E\left[Q(\Theta,\gamma)\right]$ to find $\Theta^{(t+1)}$

$$\hat{\Theta}^{(t+1)} = \text{argmax}_{\Theta} E\left[Q(\Theta,\gamma)\right]$$

  - Iterative weighted least square
  - Coordinate descent
- Repeat E- and M-steps until convergent

E Step

We decompose the expected log posterior density to two parts

$$E\left[Q(\Theta)\right] = E(Q_1) + E(Q_2)$$

where

$$E(Q_1) = \log p(\mathbf{y}|\beta,\phi) + \log p(\phi) + \sum_{j=1}^{p}\left[E(S_j^{0^{-1}})\beta_j^{0^2} + \sum_{k=1}^{K_j} E(S_j^{-1})\beta_{jk}^2\right]$$

$$E(Q_2) = \sum_{j=1}^{p}\left[E(\gamma_j^0)\log\theta_j^0 + (1-E(\gamma_j^0))\log(1-\theta_j^0)\right] + \sum_{j=1}^{p}\log p(\theta_j^0)$$
$$+ \sum_{j=1}^{p}\left[E(\gamma_j^{pen})\log\theta_j + (1-E(\gamma_j^{pen}))\log(1-\theta_j)\right] + \sum_{j=1}^{p}\log p(\theta_j)$$

E-step (Cont.)

We are interested to calculate $p_j^0 \equiv E(\gamma_j^0)$, $p_j \equiv E(\gamma_j^{pen})$, $E({S_j^0}^{-1})$ and $E(S_j^{-1})$ via Bayes' Theorem

$$p_j = \frac{Pr(\gamma_j^{pen}=1|\theta_j)\prod_{k=1}^{K_j} f(\beta_{jk}|\gamma_j^{pen}=1,s_1)}{Pr(\gamma_j^{pen}=1|\theta_j)\prod_{k=1}^{K_j} f(\beta_{jk}|\gamma_j^{pen}=1,s_1) + Pr(\gamma_j^{pen}=0|\theta_j)\prod_{k=1}^{K_j} f(\beta_{jk}|\gamma_j^{pen}=0,s_0)}.$$

Similarly, we have can have p_j^0 and

$$E({S_j^0}^{-1}) = E\left[(1-\gamma_j^0)s_0 + \gamma_j^0 s_1\right] = \frac{1-p_j^0}{s_0} + \frac{p_j^0}{s_1}$$

$$E(S_j^{-1}) = \frac{1-p_j}{s_0} + \frac{p_j}{s_1}.$$

M Step

We maximize $E(Q_1)$ and $E(Q_2)$ separately

- Maximize $E(Q_1)$ for β, ϕ
  - Iterative weighted least square algorithm
- Maximize $E(Q_2)$ for $\boldsymbol{\theta}$
  - Beta conjugate prior
  - Closed form solution

$$\theta_j^0 = \frac{p_j^0 + a - 1}{a + b - 1} \quad \theta_j = \frac{p_j + a - 1}{a + b - 1}$$

EM - Iterative Weighted Least Square

- Update β by maximizing the linear approximation to the normal likelihood using weighted least square (Yi and Ma 2012).

Equivalently, running the augmented weighted normal linear regression

$$z_* \approx N(X_*\beta, \phi\Sigma_*),$$

where $z_* = \begin{pmatrix} z \\ 0 \end{pmatrix}$, $X_* = \begin{pmatrix} X \\ I_{p+1} \end{pmatrix}$, $\Sigma_* = diag(w_1^{-1}, \dots, w_n^{-1}, \tau_0^2/\phi, \dots, \tau_p^2/\phi)$

EM - Coordinate Descent

When using mixture double exponential prior, $E(Q_1)$ can be written as a L_1 penalized likelihood function

$$E(Q_1) = \log p(\mathbf{y}|\beta,\phi) + \log p(\phi) + \sum_{j=1}^{p}\left[E(S_j^{0^{-1}})|\beta_j^0| + \sum_{k=1}^{K_j} E(S_j^{-1})|\beta_{jk}|\right],$$

The log likelihood function can be easily solved using coordinate descent algorithm.

Tuning Parameter Selection

- s_0 and s_1 are tuning parameters
- Empirically, s_1 have extremely small effect on changing the estimates
- Focusing on tuning s_0
- Instead of the 2-D grid, We consider a sequence of L ordered values $\{s_0^l\} : 0 < s_0^1 < s_0^2 < \cdots < s_0^L < s_1$
- Cross-validation to choose optimal value for s_0

Simulation Study

- Follow the logistic regression simulation introduced in Bai (2020).
- $n_{train} = 500, \; n_{test} = 1000$
- $p = 4, 10, 50$

$$\log\left(\frac{\mathbb{E}(Y)}{1 - \mathbb{E}(Y)}\right) = 5\sin(2\pi x_1) - 4\cos(2\pi x_2 - 0.5) + 6(x_3 - 0.5) - 5(x_4^2 - 0.3),$$

- $f_j(x_j) = 0$ for $j = 5, \ldots, p$.
- 200 Iterations
- Splines are constructed using cubic spline with 25 knots

Metric

- Area under the curve
- Misclassification rate
- MSE

$$\text{MSE} = n_{train}^{-1} \sum_{i=1}^{n_{train}} (y_{i,new} - \hat{y}_{i,new})^2.$$

Performance

Reproducibility

Reproducibility receipt

```
## [1] "2021-08-02 12:16:33 CDT"

## Local:    master C:/Users/boyiguo1/Documents/GitHub/Talk_JSM2021-BHAM
## Remote:   master @ origin (https://github.com/boyiguo1/Talk_JSM2021-BHA
## Head:     [a86000e] 2021-08-02: Initial commit

## R version 4.0.5 (2021-03-31)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17763)
##
## Matrix products: default
##
## locale:
```