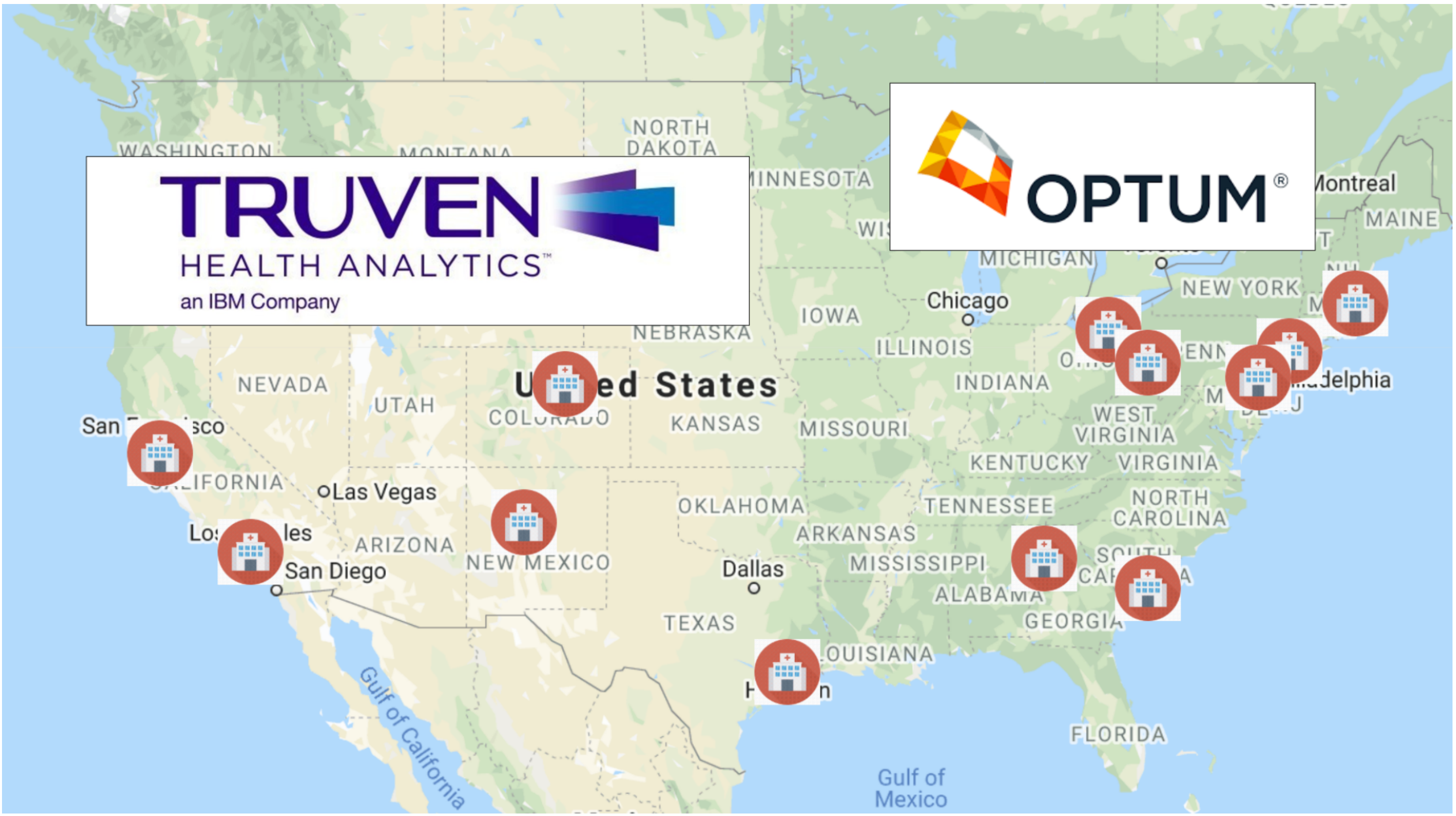
Large-scale Bayesian sparse regression for OHDSI network studies

Aki Nishimura

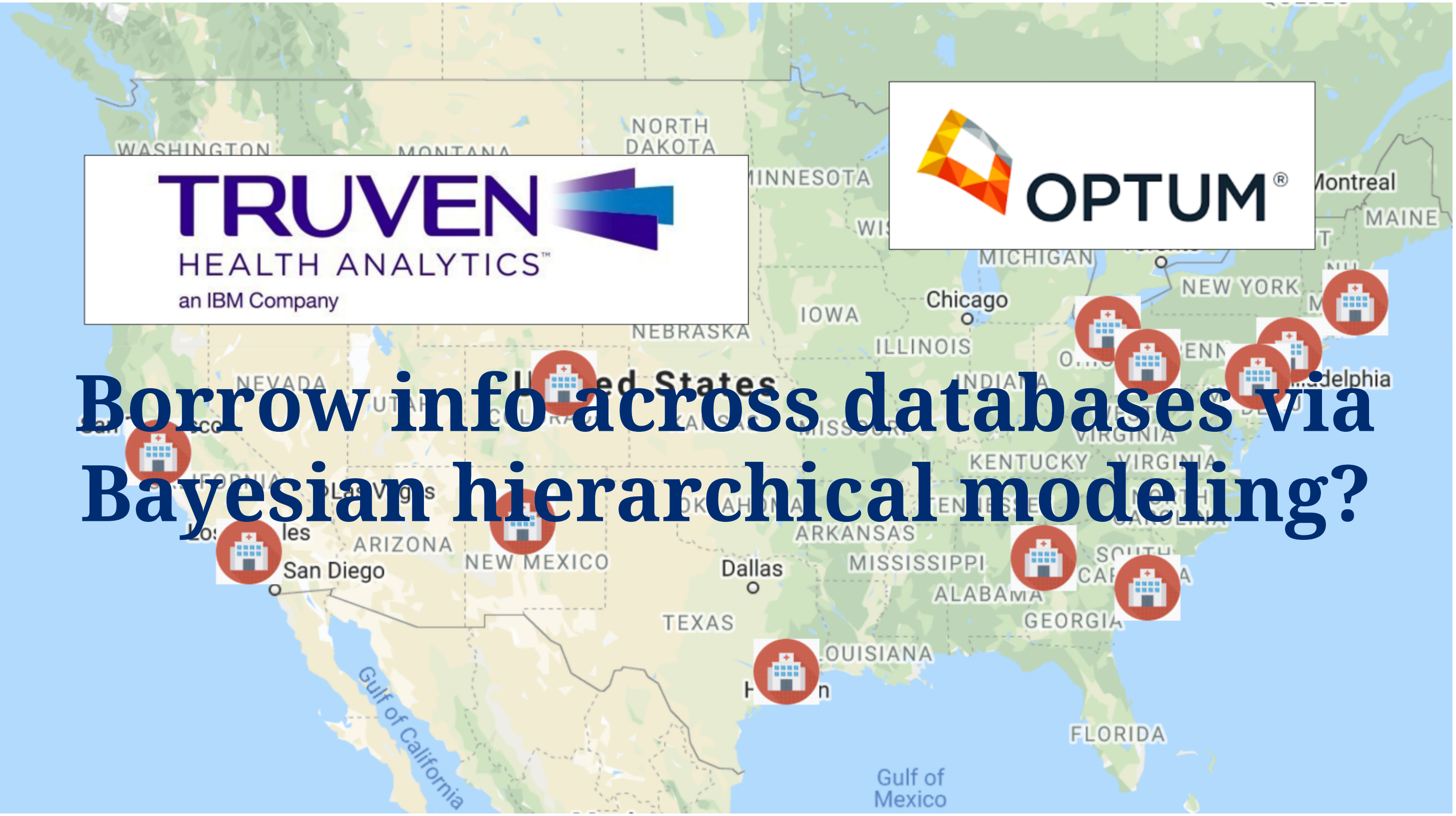
~~Problem~~ Opportunity for OHDSI

Many health databases are too small & too heterogeneous.



~~Problem~~ Opportunity for OHDSI

Many health databases are too small & too heterogeneous.



Large-scale L^1-penalized regression: a statistical engine behind OHDSI studies

Large-scale L^1-penalized regression: a statistical engine behind OHDSI studies

International Journal of Epidemiology

IEA International Epidemiological Association

Evaluating large-scale propensity score performance through real-world and synthetic data experiments FREE

Yuxi Tian ✉, Martijn J Schuemie, Marc A Suchard

International Journal of Epidemiology, Volume 47, Issue 6, December 2018, Pages 2005–2014, https://doi.org/10.1093/ije/dyy120

Published: 22 June 2018 **Article history** ▾

Large-scale L^1-penalized regression: a statistical engine behind OHDSI studies

Cyclops

R-CMD-check passing | codecov 67% | CRAN 3.1.2 | downloads 933/month

Cyclops is part of the HADES.

Introduction

Cyclops (Cyclic coordinate descent for logistic, Poisson and survival analysis) is an R package for performing large scale regularized regressions.

Examples

```
library(Cyclops)
cyclopsData <- createCyclopsDataFrame(formula)
cyclopsFit <- fitCyclopsModel(cyclopsData)
```

Penalized vs. Bayesian sparse regression

Under Bayes, data $\boldsymbol{y}$ and $\boldsymbol{X}$ inform unknown $\boldsymbol{\beta}$ via:

$$\pi_{\text{post}}(\boldsymbol{\beta} \mid \boldsymbol{y}, \boldsymbol{X}) \propto L(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\beta})\, \pi_{\text{prior}}(\boldsymbol{\beta}).$$

Penalized vs. Bayesian sparse regression

Using prior is analogous to placing *penalty* on $\boldsymbol{\beta}$:

$$\hat{\boldsymbol{\beta}} = \mathrm{argmin}_{\boldsymbol{\beta}} \{ -\log L(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\beta}) + \mathrm{pen}(\boldsymbol{\beta}) \}$$

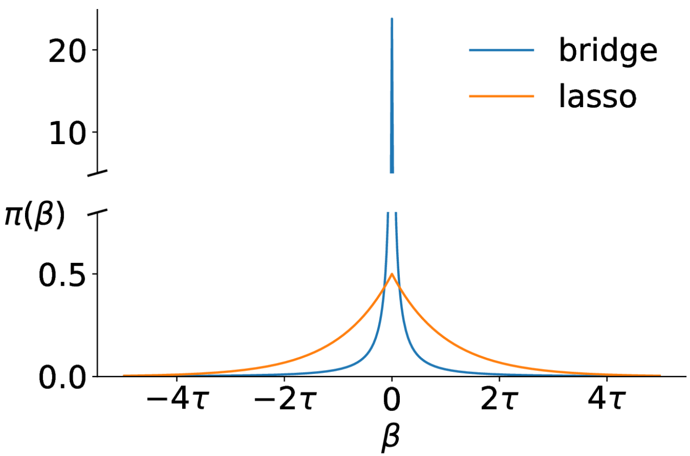
where $\mathrm{pen}(\boldsymbol{\beta})$ "=" $-\log \pi_{\mathrm{prior}}(\boldsymbol{\beta})$.

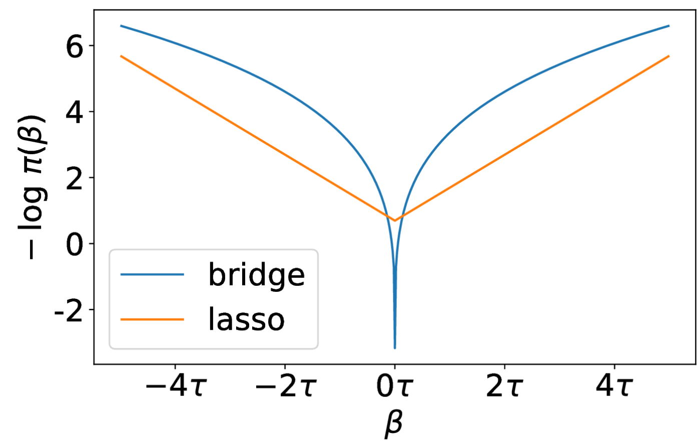
Penalized vs. Bayesian sparse regression

Using prior is analogous to placing *penalty* on $\boldsymbol{\beta}$:

$$\hat{\boldsymbol{\beta}} = \mathrm{argmin}_{\boldsymbol{\beta}} \{ -\log L(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\beta}) + \mathrm{pen}(\boldsymbol{\beta}) \}$$

where $\mathrm{pen}(\boldsymbol{\beta})$ "$=$" $-\log \pi_{\mathrm{prior}}(\boldsymbol{\beta})$.





Example: Bridge prior $\pi_{\mathrm{prior}}(\beta_j \mid \tau) \propto \tau^{-1} \exp\big(-|\beta_j/\tau|^{\alpha} \big)$

"Bayes doesn't scale"?

Bayesians often rely on *Monte Carlo* simulation, drawing

$$\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(M)} \sim \pi_{\text{post}}(\, \cdot \mid \boldsymbol{y}, \boldsymbol{X}),$$

and use $M^{-1} \sum_m \delta_{\boldsymbol{\beta}^{(m)}}(\cdot)$ to quantify the posterior.

"Bayes doesn't scale"?

Bayesians often rely on *Monte Carlo* simulation, drawing

$$\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(M)} \sim \pi_{\text{post}}(\,\cdot \mid \boldsymbol{y}, \boldsymbol{X}),$$

and use $M^{-1} \sum_m \delta_{\boldsymbol{\beta}^{(m)}}(\cdot)$ to quantify the posterior.

This computation can be **prohibitively expensive**.

~~"Bayes doesn't scale"~~

Home ▶ All Journals ▶ Journal of the American Statistical Association ▶

Journal of the American Statistical Association ›

Latest Articles

Submit an article | Journal homepage

Theory and Methods

Prior-Preconditioned Conjugate Gradient Method for Accelerated Gibbs Sampling in "Large *n*, Large *p*" Bayesian Sparse Regression

Akihiko Nishimura & Marc A. Suchard

Received 11 Feb 2020, Accepted 18 Mar 2022, Accepted author version posted online: 27 Mar 2022, Published online: 09 May 2022

Download citation https://doi.org/10.1080/01621459.2022.2057859 Check for updates

~~"Bayes doesn't scale"~~

Example: Compare alt. treatments for atrial-fibrillation, blood anti-coagulants *dabigatran* and *warfarin*.

Objective: Study relative risk of *gastrointestinal bleeding*.

- $n = 72{,}489$ patients, 27.3% dabigatran users
- $p = 22{,}175$ covariates

~~"Bayes doesn't scale"~~

Example: Compare alt. treatments for atrial-fibrillation, blood anti-coagulants *dabigatran* and *warfarin*.

Objective: Study relative risk of *gastrointestinal bleeding*.

- $n =$ 72,489 patients, 27.3% dabigatran users
- $p =$ 22,175 covariates

Computing time: With the previous state-of-the-art,

- Propensity score model
  — 106 hours for 5,500 iterations,
- Outcome model with subgroup-effect interactions
  — 212 hours for 11,000 iterations.

~~"Bayes doesn't scale"~~

Example: Compare alt. treatments for atrial-fibrillation, blood anti-coagulants *dabigatran* and *warfarin*.

Objective: Study relative risk of *gastrointestinal bleeding*.

- $n = 72{,}489$ patients, 27.3% dabigatran users
- $p = 22{,}175$ covariates

Computing time: With the new algorithm,

- Propensity score model
  — 11.4 hours (9.3-fold speedup) for 5,500 iterations,
- Outcome model with subgroup-effect interactions
  — 11.3 hours (18.8-fold speedup) for 11,000 iterations.

~~"Bayes doesn't scale"~~

Example: Compare alt. treatments for atrial-fibrillation, blood anti-coagulants *dabigatran* and *warfarin*.

Objective: Study relative risk of *gastrointestinal bleeding*.

- $n = 72{,}489$ patients, 27.3% dabigatran users
- $p = 22{,}175$ covariates

Computing time: With the new algorithm + GPU,

- Propensity score model
  — 0.62 hours (171-fold speedup) for 5,500 iterations,
- Outcome model with subgroup-effect interactions
  — 0.61 hours (347-fold speedup) for 11,000 iterations.

New algorithm in Python's BayesBridge

bayesbridge
latest

Search docs

DOCUMENTATION

Demo

Main Class and Method

Read the Docs v: latest

BayesBridge

Python package for Bayesian sparse regression, implementing the standard (Polya-Gamma augmented) Gibbs sampler as well as the CG-accelerated sampler of Nishimura and Suchard (2018). The latter algorithm can be orders of magnitudes faster for a large and sparse design matrix.

Installation

```
pip install bayesbridge
```

Background

The Bayesian bridge is based on the following prior on the regression coefficients β_j's:

$$\pi(\beta_j \,|\, \tau) \propto \tau^{-1} \exp\left(-|\beta_j/\tau|^{\alpha}\right) \quad \text{for} \quad 0 < \alpha \leq 1$$

The Bayesian bridge recovers the the Bayesian lasso when $\alpha = 1$ but can provide an improved separation of the significant coefficients from the rest when $\alpha < 1$.

Usage

```
from bayesbridge import BayesBridge, RegressionModel, RegressionCoefPrior

model = RegressionModel(y, X, family='logit')
prior = RegressionCoefPrior(bridge_exponent=.5)
bridge = BayesBridge(model, prior)
mcmc_output = bridge.gibbs(
    n_burnin=100, n_post_burnin=1000, thin=1,
    coef_sampler_type='cholesky' # Try 'cg' for large and sparse X
)
coef_samples = mcmc_output['samples']['coef']
```

bayesbridger: R wrapper based on reticulate

Set up Python environments,

```r
library(bayesbridger)
configure_python(envname = "bayesbridge")
```

instantiate BayesBridge with data y and X,

```r
model <- create_model(y, X)
prior <- create_prior(bridge_exponent=.25)
bridge <- instantiate_bayesbridge(model, prior)
```

and sample from the posterior!

```r
gibbs_output <- gibbs(bridge, n_iter = 1000L,
                      coef_sampler_type = "cg")
mcmc_samples <- gibbs_output$samples
```

Thank you!