Fast non-parametric regression using different approximation methods

Aniruddha R Rao

April 24, 2018

Introduction

Kernel ridge regression : Consider the supervised problem of learning a function given a training set of n examples (x_i, y_i), i =1,2.....,n , where $x_i \in$ X, X = R^d and $y_i \in$ R. Kernel methods are nonparametric approaches defined by a kernel K : X * X $\rightarrow$ R, that is a symmetric and positive definite (PD) function. A particular instance is kernel ridge regression given by,

$$f_\lambda(x) = \Sigma_{i=1}^{n} \alpha_i k(x_i, x)$$

The convex problem is,
$\hat{f} = argmin_{f \in \mathcal{H}} [1/N \Sigma_{i=1}^{n} (f(x_i) - y_i)^2 + \lambda ||f||_{\mathcal{H}}^2]$

Equivalently, $\hat{\alpha} = argmin_{\alpha \in R^n} [(y - k\alpha)^{'}(y - k\alpha) - \lambda \alpha^{'} k\alpha]$
Solution, $\hat{\alpha} = (k + \lambda I)^{-1} y$

Kernel Ridge Regression

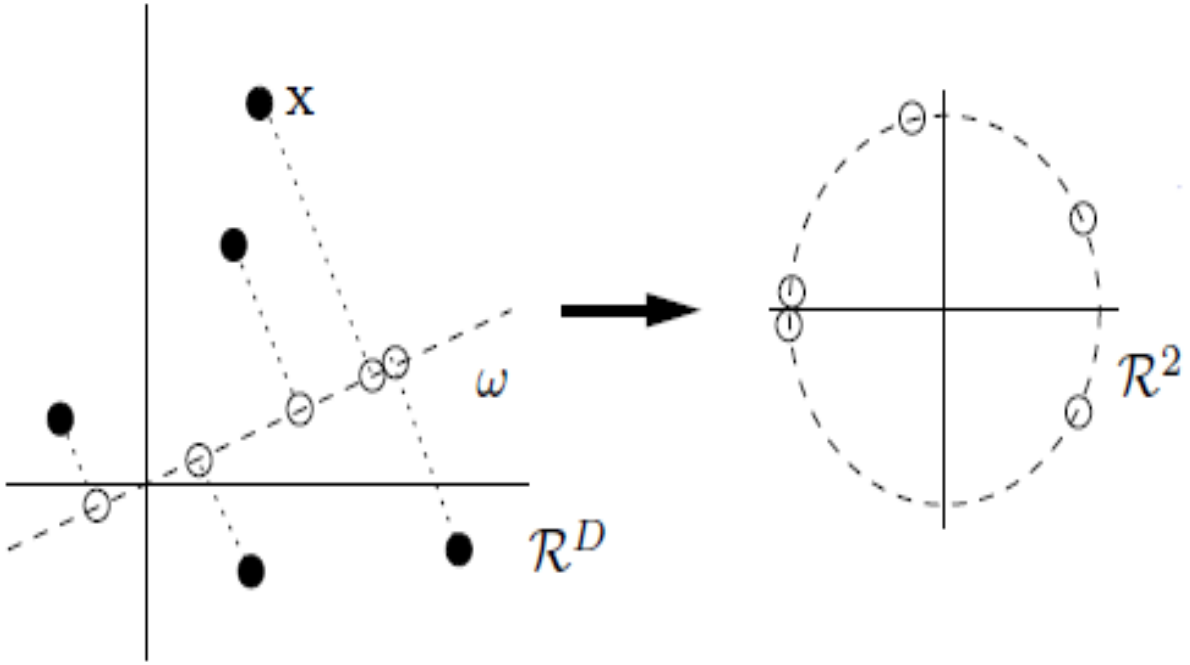
What is the Problem?

- Kernel matrix (Gram matrix) is fully dense and scales poorly with the size of the training dataset.
- k(x, y)= $< \phi(x), \phi(y) >$. Memory O(n^2) and computation time O(n^3).

Solution is to approximate the kernel matrix or approximates the kernel function directly.

Random Fourier Features

Random features consists of random Fourier bases z(x)=cos(w'x) where w$\in R^d$. Each component of the feature map z(x) projects x onto a random direction w drawn from the Fourier transform p(w) of $k(x, y)$ and wraps this line onto the unit circle in R^2. After transforming two points x and y in this way, their inner product is an unbiased estimator of k(x,y).



RFF Algorithm

- Select a positive definite shift-invariant kernel k(x, y) = $k(x - y)$.
- Compute the Fourier transform p of the kernel k:
  $p(w) = 1/(2\pi) \int e^{jw'\delta} K(\delta) d \triangle$.
- Draw D iid samples w_1,.....,$w_D \in R^d$ from p.
- Construct a randomized feature map z(x) : $R^d \to R^D$ so that
  $z(x)'z(y) \approx k(x - y)$.
  z(x)=$\sqrt{1/D}[cos(w_1'x), ..., cos(w_D'x), sin(w_1'x), ..., sin(w_D'x)]$

Note : z(x) takes O(nD^2 + D^3) in time, O(nD) in space, and and
k(x,y) =($e^{-s*||x-y||^2/2}$) follows p(w) =N_d(μ=0,Σ=s*I)

Sketching's Method

Sketching's method approximates of KRR based on m-dimensional randomized sketches (projections) of the kernel matrix.

Algorithm:

- Define $\alpha_{n*1} = S_{n*m} * W_{m*1}$ where S is a matrix defined by random sketches where m<<n.
- $\hat{W} = argmin_{W \in R^m}[(y - kSW)'(y - kSW) - \lambda W' S' kSW]$
  $\hat{W} = [S'(k + n\lambda I)S]^{-1} S' y$
- $f_\lambda(x) = \Sigma_{i=1}^{n} (SW)_i k(x_i, x)$

Note : The computation time gets reduced to O(n^2 log(m)+m^3) and storage space is O(nm+m^2)

Standard Nystrom

Algorithm:

- Decide m (<<n). Randomly sample m columns from k(x,y). Get W_{m*m}.
-

$$k(x,y) = \begin{bmatrix} W_{m*m} & k' \\ k_{(n-m)*m} & f(W,k) \end{bmatrix} \quad C_{n*m} = \begin{bmatrix} W \\ k \end{bmatrix}$$

- $k(x,y) \approx CW^{-}C^{T}$ & $(\mathrm{k}+\lambda^{*}\mathrm{I})^{-1}=(\mathrm{I}-\mathrm{C}[\lambda^{*}\mathrm{I}+W^{-}C^{T}C]^{-}W^{-}C^{T})/\lambda$
- Substitute above values in KRR method.

Note : The computation time gets reduced to $\mathrm{O}(m^3+\mathrm{n}m^2)$ and storage space is $\mathrm{O}(m^2+\mathrm{nm})$

Other methods

1. **Model Averaging**
   - Randomly partitions a dataset of size n into m subsets of equal size
   - Compute an independent kernel ridge regression model for each subset
   - Average the local solutions into a global predictor.

Note : The computation time gets reduced to $O(n^3/m^2)$ and storage space is $O(n^2/m^2)$

2. **Cholesky Decomposition**

Simulation

One Dimensional example:

y =sin(x)+cos(x) where x $\in$ (-6,6) and y $\in$ (-1.42,1.42)

Gaussian kernel : k(x,y)=exp(-s*$||x-y||^2$)

The bandwidth (s) and penalty (λ) are constant across the methods and the other tuning parameters are functions for n. (ϵ=0.03)

Multi Dimensional example:

y$\sim N_{d=5}(u=0, \Sigma=I)$ where I is identity matrix.

Gaussian kernel : k(x,y)=exp(-s*$||x-y||^2$)

All the parameters had to be tuned for every case of n in each of the approximate method.(ϵ=1*10^{-8})

Simulation result : 1-D example

Figure 1: Comparison plot of different methods for n=5000

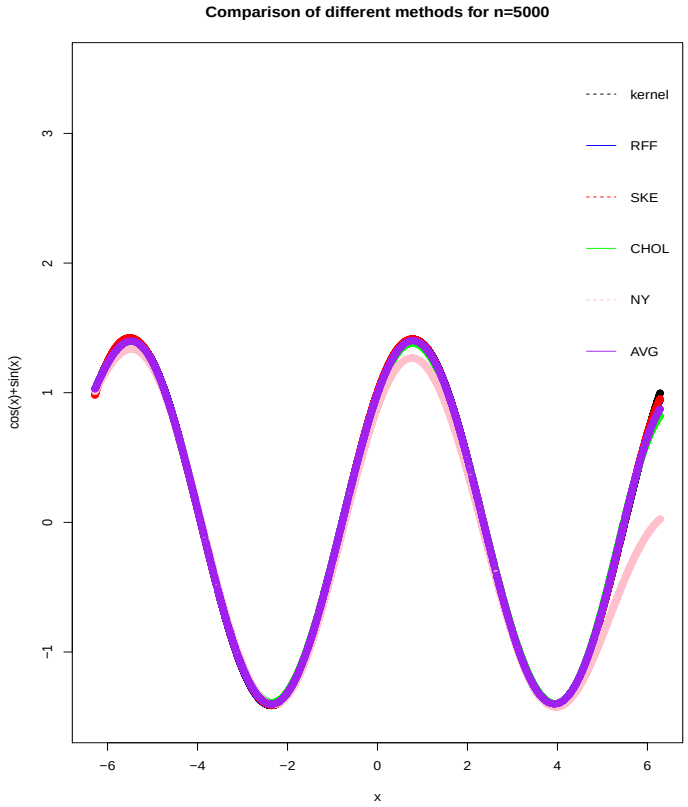
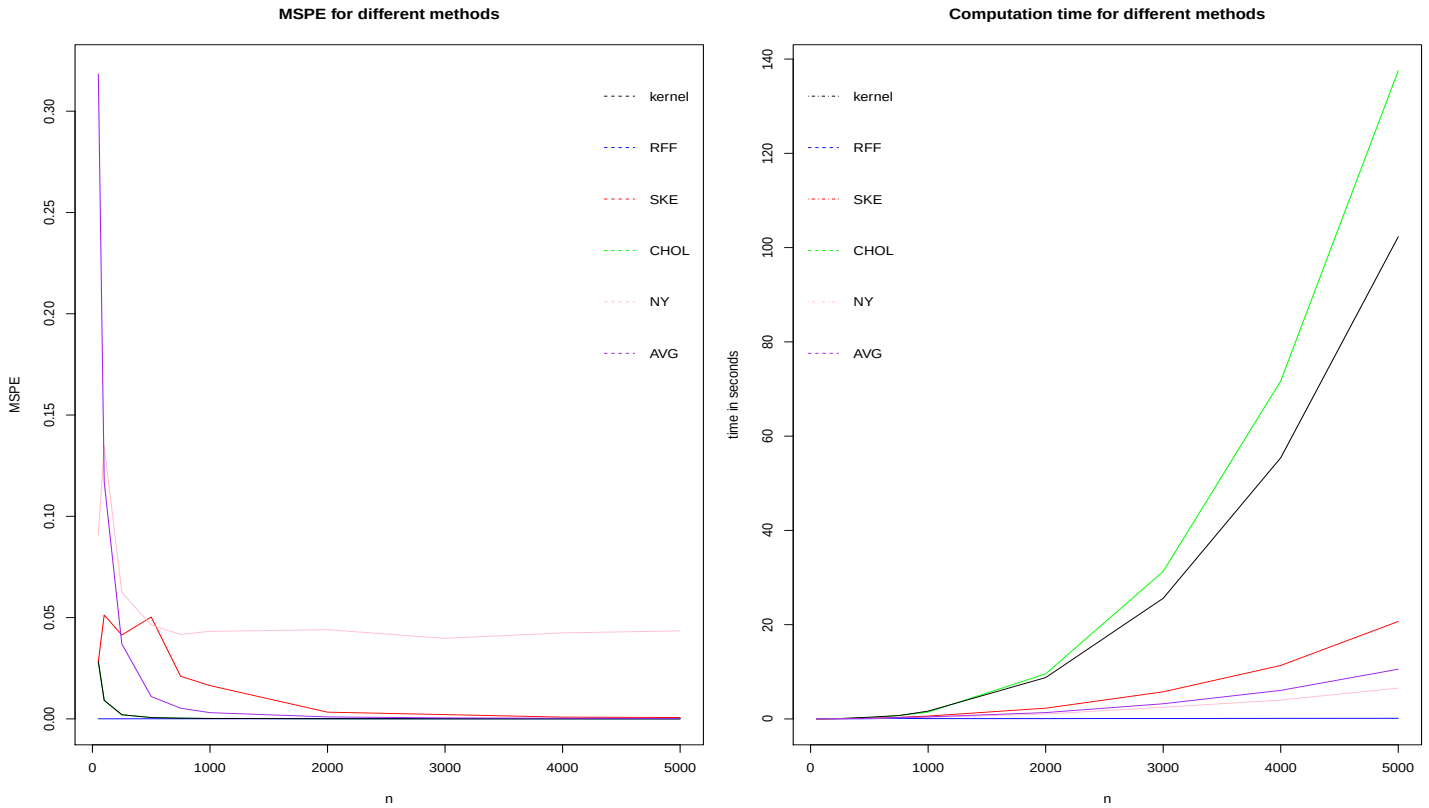


Table 1: Comparison for n=5000

Method	MSPE
KRR	3.580697e-04
RFF	5.722305e-05
NYSTROM	4.166946e-02
SKETCHING	2.104542e-02
CHOLESKY	3.580697e-04
MODEL AVG	5.276929e-03

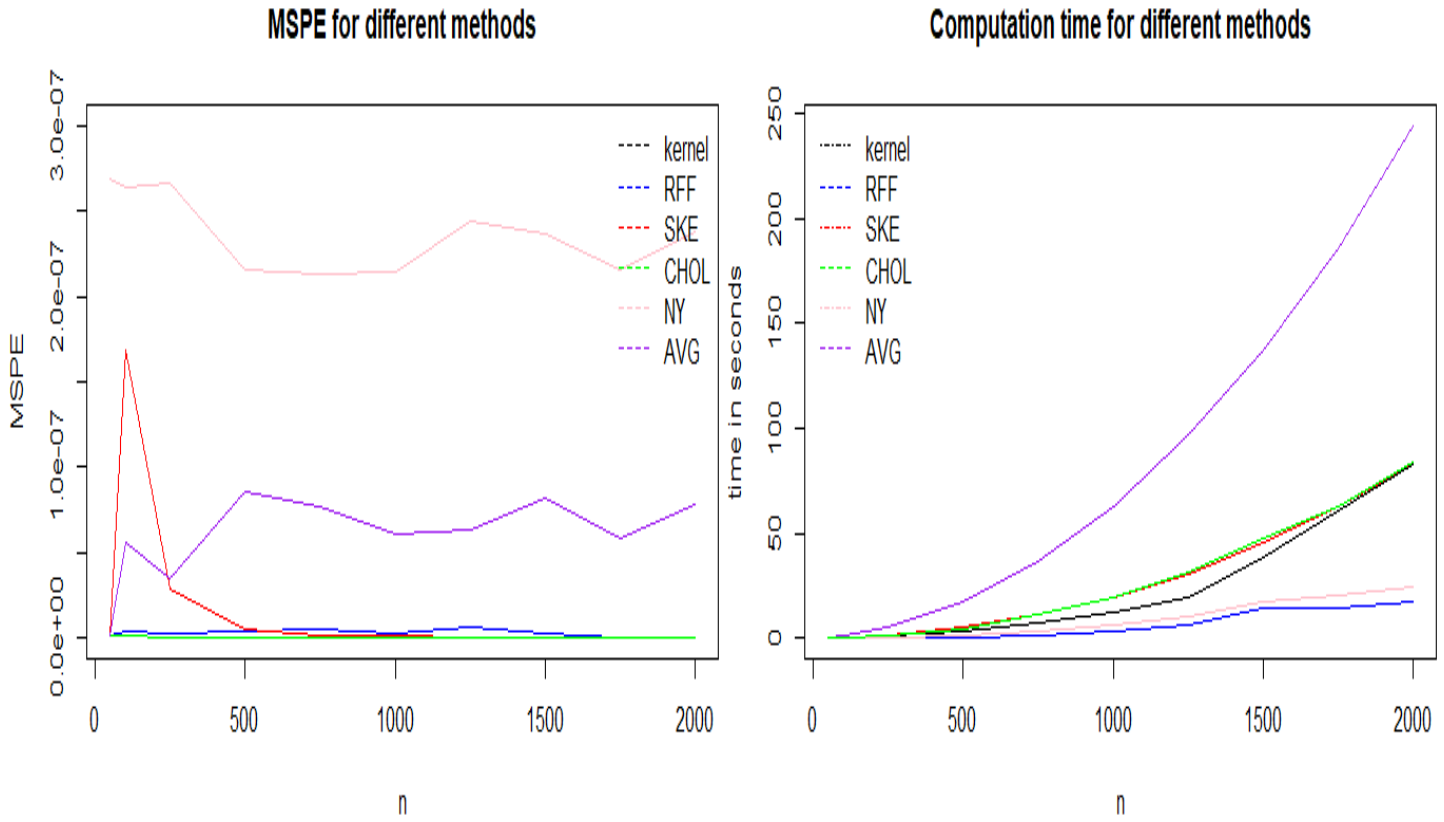
Simulation result : 1-D example

Figure 2: Plots for MSPE and Computation time



Simulation result : Multi-D example

Figure 3: Plots for MSPE and Computation time



Real Data Example

Parkinsons Telemonitoring Data Set from UCI Data Repository.
The dataset is composed of a range of biomedical voice measurements.
The goal is to predict UPDRS score from the different voice measures.

Table 2: Comparison wrt MSPE

Method	MSPE
KRR	409.8344
RFF	105.8395
NYSTROM	416.6946
SKETCHING	868.1353
CHOLESKY	409.8344
MODEL AVG	783.3628

n=5875, d=19
In all it took around 8 hours to run all the methods. I have not tuned it yet but the results look not so bad.

Problems and extension ideas

Problems:

- In the 1-D example, Nystrom method results in huge distance between the Matrices. Also could not apply direct method.
- Tuning all the methods was time consuming. Some methods had 3 tunning parameters.

Extension ideas:

- Applying different methods within Model Averaging.
- Better selection of subset for Nystrom method.
- Exploring different types of random sketches in sketching's method.

GLMM Lasso for High-dimensional Data

Presenter: Junjie Liang

College of IST

jul672@ist.psu.edu

GLMM – problem definition

Likelihood for GLM is:

$$f(y_i|\theta_i,\phi) = \exp\left(\phi^{-1}\left(y_i \cdot \theta_i - \xi(\theta_i)\right) + c(y_i,\phi)\right)$$

where ϕ and θ_i are model parameters. For GLMM:

$h(\theta_i) = \mu_i = \mathbb{E}[y_i|b_i] = g^{-1}(x_i\beta + z_i b_i)$ where $b_i \sim N(0, Q^{-1})$

For GLMM, since we have unobserved random effect, the augmented likelihood is:

$$f(y_i;\theta_i,\phi,Q) = \int f(y_i|\theta_i,\phi,b_i)\cdot f(b_i|Q)\mathrm{d}b_i$$

This is intractable because we have no close form solution to the integral!

GLMM – solutions

Bayesian inference:

- Variational Bayesian Inference (a.k.a. Variational Bayes) (VBI)
- MCMC with auxiliary variables (MCMC)

ML estimation:

- Monte Carlo EM (MC-EM)
- Laplace approximation (LA)

Variational Bayesian Inference

- We want to know $f(\boldsymbol{\theta}|y) = \frac{f(y|\boldsymbol{\theta})f(\boldsymbol{\theta})}{f(y)}$, but:
  - don't know the normalizing constant $f(y)$
  - Computing $f(y|\boldsymbol{\theta})$ is also very difficult (e.g., GLMM)
- Let's use a proposal distribution $q(\boldsymbol{\theta})$ to approximate $f(\boldsymbol{\theta}|y)$.
- A good proposal distribution would be to minimize the KL divergence

$$q^*(\boldsymbol{\theta}) = \min_{q(\boldsymbol{\theta})} KL\big(q(\boldsymbol{\theta})||f(\boldsymbol{\theta}|y)\big)$$

- The best solution is when $q(\boldsymbol{\theta}) = f(\boldsymbol{\theta}|y)$.
- Simplifying the KL divergence, we find that

$$q^*(\boldsymbol{\theta}) = \min_{q(\boldsymbol{\theta})} KL\big(q(\boldsymbol{\theta})||f(\boldsymbol{\theta}|y)\big) = \min_{q(\boldsymbol{\theta})} KL\big(q(\boldsymbol{\theta})||f(\boldsymbol{\theta}, y)\big)$$

- Therefore, the best solution is converted to when $q(\boldsymbol{\theta}) = f(y|\boldsymbol{\theta})f(\boldsymbol{\theta})$, but this is still intractable.

Variational Bayesian Inference (cont.)

- What if we assume $q(\boldsymbol{\theta}) = \prod_i q(\theta_i)$ (mean field theory)
- Then we have

$$q^*(\theta_i) = \min_{q(\theta_i)} KL\big(q(\boldsymbol{\theta}) || f(\boldsymbol{\theta}, y)\big)$$

where:

$$\begin{aligned}
&\min_{q(\theta_i)} KL\big(q(\boldsymbol{\theta}) || f(\boldsymbol{\theta}, y)\big) \\
&= \min_{q(\theta_i)} \int \prod_i q(\theta_i) \log \frac{\prod_i q(\theta_i)}{f(\boldsymbol{\theta}, y)} d\boldsymbol{\theta} \\
&= \min_{q(\theta_i)} \int \prod_i q(\theta_i) \log \prod_i q(\theta_i)\, d\boldsymbol{\theta} - \int \prod_i q(\theta_i) \log f(\boldsymbol{\theta}, y)\, d\boldsymbol{\theta} \\
&= \min_{q(\theta_i)} \int q(\theta_i) \log q(\theta_i)\, d\theta_i - \int q(\theta_i) \mathbb{E}_{q(\theta_{-i})}[\log f(\boldsymbol{\theta}, y)] d\theta_i \\
&= \min_{q(\theta_i)} KL\big(q(\theta_i) || \exp\big(\mathbb{E}_{q(\theta_{-i})}[\log f(\boldsymbol{\theta}, y)]\big)\big)
\end{aligned}$$

Therefore, following the mean field theory, we want to find $\theta_i, i = 1, \dots, |\theta|$:

$$q(\theta_i) \propto \exp\big(\mathbb{E}_{q(\theta_{-i})}[\log f(\boldsymbol{\theta}, y)]\big)$$

where $q(\theta_i)$ should be a valid distribution.

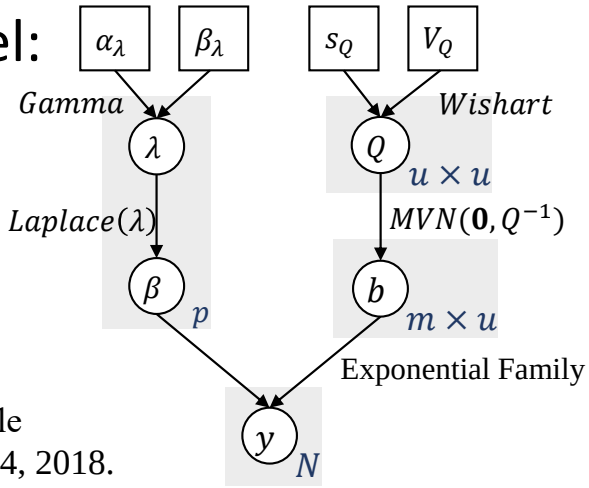
Variational Bayesian Inference (cont.)

General guidance for finding $q(\theta_i)$:

- Following $q^*(\theta_i) = \min_{q(\theta_i)} KL\big(q(\boldsymbol{\theta})||f(\boldsymbol{\theta}, y)\big)$. Find $q^*(\theta_i)$ directly using conjugate distribution.
- Following $q^*(\theta_i) \propto \exp\big(\mathbb{E}_{q(\theta_{-i})}[\log f(\boldsymbol{\theta}, y)]\big)$. Find $q^*(\theta_i)$ using some tricks.
- In BI framework for GLMM, [1] gives a graphical model:

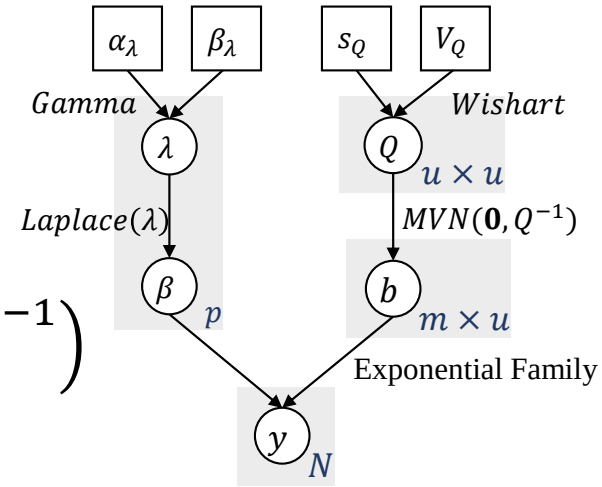
Therefore, our goal is:

$$q^*(\lambda, \beta, Q, b) = \min_{q} KL\big(q(\lambda, \beta, Q, b)||f(\lambda, \beta, Q, b, y)\big)$$



[1] D. T. Tung, M.-N. Tran, and T. M. Cuong, "Bayesian adaptive lasso with variational Bayes for variable selection in high-dimensional generalized linear mixed models," *Commun. Stat.-Simul. Comput.*, pp. 1–14, 2018.

Solving GLMM with VBI



Following mean field theory, we have:

- $q^*(Q) \sim Wishart\left(s_Q + m, \left(V_Q^{-1} + \sum_{i=1}^{m} \mathbb{E}_{q(b)}\left[b_i \cdot b_i^T\right]\right)^{-1}\right)$
- $q^*(\lambda) \sim Gamma\left(\alpha_\lambda + 1, \beta_\lambda + \mathbb{E}_{q(\beta)}[|\beta|]\right)$
- $q^*(b) \propto \exp \mathbb{E}_{-q(b)}[\log f(y|b,\beta) f(b|Q)]$
- $q^*(\beta) \propto \exp \mathbb{E}_{-q(\beta)}[\log P(y|b,\beta) P(\beta|\lambda)]$

For $q(b)$ and $q(\beta)$, I use gaussian approximation, thus:

$$q(b_i) \sim MVN\left(b_i^*, -\left(Z_i^T \cdot \mathrm{Diag}\left\{\zeta''\left(\eta_i(b_i^*)\right) \times_e \eta_i''(b_i^*)\right\} \cdot Z_i - \mathbb{E}_{q(Q)}[Q]\right)^{-1}\right)$$

$$q(\beta) \sim MVN\left(\beta^*, -\left(X^T \cdot \mathrm{Diag}\left\{\zeta''\left(\eta_i(\beta^*)\right) \times_e \eta_i''(\beta^*)\right\} \cdot X\right)^{-1}\right)$$

Generalized EM algorithm

In EM algorithm, for iteration t, we do the following:

$$f\left(y|\theta^{(t)}\right) = \int f\left(y, z|\theta^{(t)}\right)dz = \int \frac{f\left(y, z|\theta^{(t)}\right)}{q^{(t)}(z)} q^{(t)}(z)dz$$

where $q^{(t)}(z) = f\left(z|y, \theta^{(t)}\right)$. Let's continue to use $q(z)$, then we have:

$$f\left(y|\theta^{(t)}\right) = \mathbb{E}_{q^{(t)}(z)}\left[\frac{f\left(y, z|\theta^{(t)}\right)}{q^{(t)}(z)}\right]$$

Assume that we want to maximize the log-likelihood, then:

$$l\left(\theta^{(t)}; y\right) = \max_{\theta} \log f\left(y|\theta^{(t)}\right) \geq \mathbb{E}_{q^{(t)}(z)}\left[\log \frac{f\left(y, z|\theta^{(t)}\right)}{q^{(t)}(z)}\right]$$

$$= \int q^{(t)}(z) \log \frac{f\left(z|y, \theta^{(t)}\right) f\left(y|\theta^{(t)}\right)}{q^{(t)}(z)} dz$$

$$= \log f\left(y|\theta^{(t)}\right) - KL\left(q^{(t)}(z)||f\left(z|y, \theta^{(t)}\right)\right)$$

Therefore, when $q^{(t)}(z) = f\left(z|y, \theta^{(t)}\right)$, we have $KL\left(q^{(t)}(z)||f\left(z|y, \theta^{(t)}\right)\right) = 0$, which is the optimal solution.

Generalized EM algorithm

For the E-step, what we exactly want is to compute:

$$f\left(y|\theta^{(t)}\right) = \mathbb{E}_{q^{(t)}(z)}\left[\frac{f\left(y,z|\theta^{(t)}\right)}{q^{(t)}(z)}\right]$$

This can be achieved by two ways:

1. If we can draw samples from $q^{(t)}(z)$, then we don't need to know the form of $q^{(t)}(z)$. This results in MC-EM algorithm.
2. If we need the close form distribution of $q^{(t)}(z)$, then our goal is:

$$q^{(t)}(z) = \min_{q(z)} KL\left(q(z)||f\left(z|y,\theta^{(t)}\right)\right)$$

   This results in VBI.

Generalized EM algorithm:

E-step: solve $q^{(t)}(z) = \min_{q(z)} KL\left(q(z)||f\left(z|y,\theta^{(t)}\right)\right)$

M-step: solve $\theta^{(t+1)} = \max_{\theta} \mathbb{E}_{q^{(t)}(z)}\left[\frac{f\left(y,z|\theta^{(t)}\right)}{q^{(t)}(z)}\right]$

Solving GLMM with MCEM [2]

We want to optimize the following objective function:

$$\hat{\theta} = \max_{\theta} \sum_{i=1}^{m} \log \int_{\mathbb{R}^q} f(y_i|\beta, b_i, Q) f(b_i|Q) \mathrm{d}b_i - \lambda \left|\left|\beta\right|\right|_1$$

E-step: we want to solve $\mathbb{E}_{b_i}\left[\log \frac{f(y_i|\theta_i, b_i) f(b_i|Q)}{f\left(b_i|\theta_i^{(t)}, y_i\right)} |\theta_i^{(t)}, y_i\right]$

M-step: $\hat{\theta}^{(t+1)} = \max_{\theta} \sum_{i=1}^{m} \mathbb{E}_{b_i}\left[\log \frac{f(y_i|\theta_i, b_i) f(b_i|Q)}{f\left(b_i|\theta_i^{(t)}, y_i\right)} |\theta_i^{(t)}, y_i\right] - \lambda \left|\left|\beta\right|\right|_1$

For E step, I apply the Metropolis-Hasting algorithm, where

$$f\left(b_i|\beta^{(t)}, Q^{(t)}, y_i\right) = \frac{f\left(y_i|b_i, \beta^{(t)}\right) f\left(b_i|Q^{(t)}\right) f\left(\beta^{(t)}\right)}{f\left(y_i, \beta^{(t)}, Q^{(t)}\right)}$$

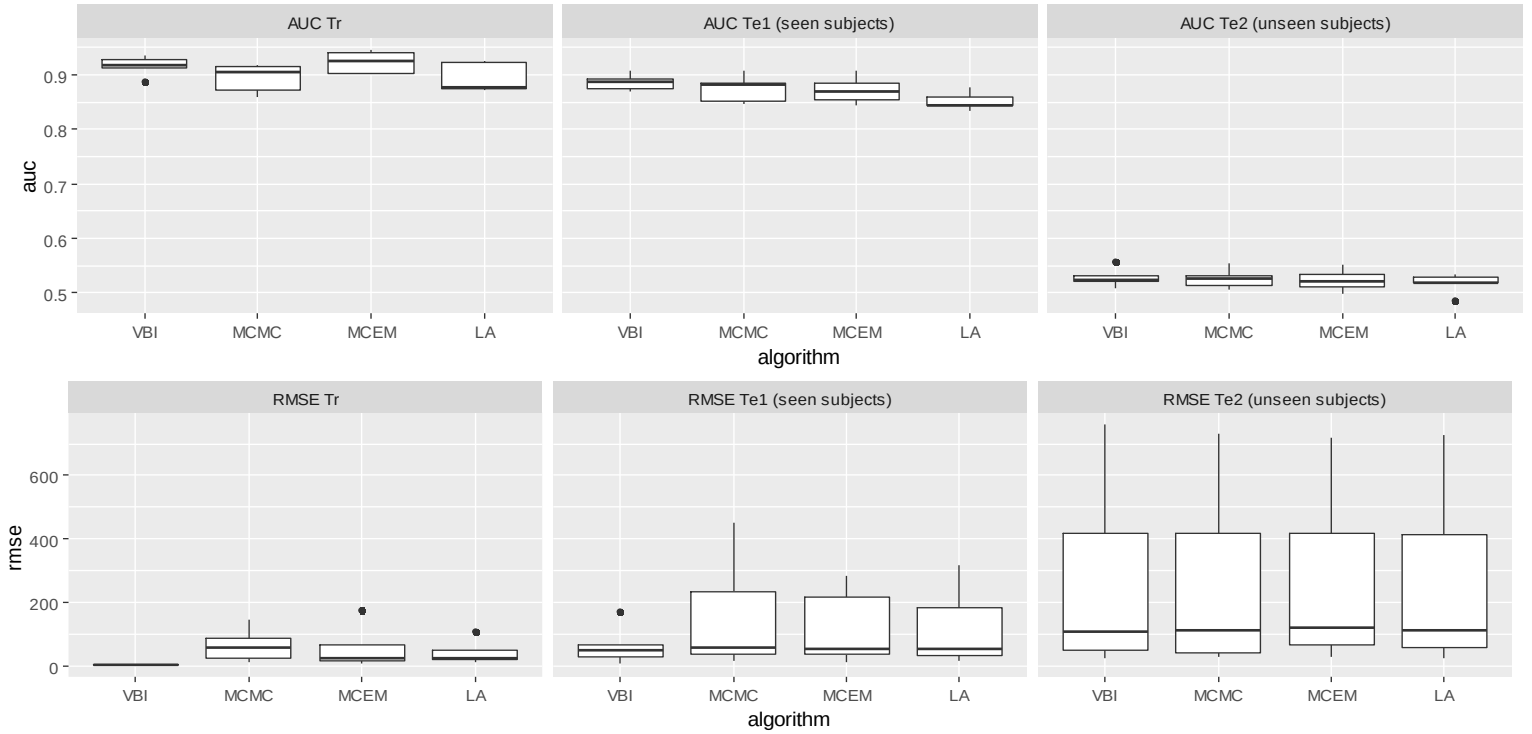
C. E. McCulloch, “Maximum likelihood algorithms for generalized linear mixed models,” *J. Am. Stat. Assoc.*, vol. 92, no. 437, pp. 162–170, 1997.

Simulation study

- Logistic regression and Poisson regression.
- BI: VBI and MCMC; MLE: MC-EM and LA
- Simulation process:
  - Generate β, X, Z.
  - For each subject, generate $b_i \sim N(0, Q^{-1})$
  - Compute $\mu_{it} = g^{-1}(x_{it}\beta + z_{it}b_i)$
  - Simulate y_{it} with mean μ_{it}
  - Simulate two test sets. One is to reuse b_i, assuming predicting the outcome for existing subjects; one is only reusing β and re-simulate b_i, assuming predicting the outcome for new subjects.
- Evaluation metrics:
  - AUC for Logistic Reg, RMSE for Poisson Reg
  - Runtime
  - Overall coverage rate/coverage rate for sparsity

Results

Algorithm	Runtime/ iteration	AUC for Tr	AUC for Te_1	AUC for Te_2	Coverage	Coverage sparse
VBI	93.03	0.977	0.824	0.572	0.397	1.000
MCMC	224.74/100	1.000	0.879	0.573	0.064	0.089
MCEM	215.23	1.000	0.825	0.536	1.000	1.000
LA	932.46	0.962	0.801	0.550	1.000	1.000



Conclusion

- In all cases, algorithms based on BI model is faster than MLE
- For low-dimensional case, VBI > MC-EM > MCMC > LA
- For high-dimensional case, algorithms based on BI model outperform MLE
- MLE usually has larger variance.
- Difficulty of derivation and implementation:

VBI > MC-EM > LA > MCMC

Simulation study of Stochastic Gradient Descent Algorithms

Balaji Kumar

STAT 540

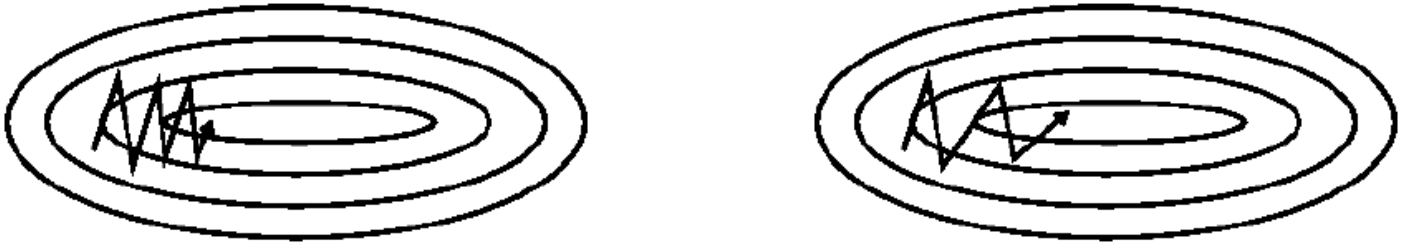
24 April 2018

Stochastic Gradient Descent

- Randomly pick one or more data points to compute $\nabla J(\theta, x^*)$;
- Update: $\theta_{i+1} = \theta_i - \eta \nabla J(\theta, x^*)$
- Challenges:
  - No convergence guaranteed
  - Choosing “ideal” learning rate can be difficult.
  - Learning rate schedule needs to be fine tuned according to data
  - Having same learning rate for all parameters is not optimal
  - SGD gets trapped in saddle points in non-convex optimization (Dauphin et al, 2015)

Momentum

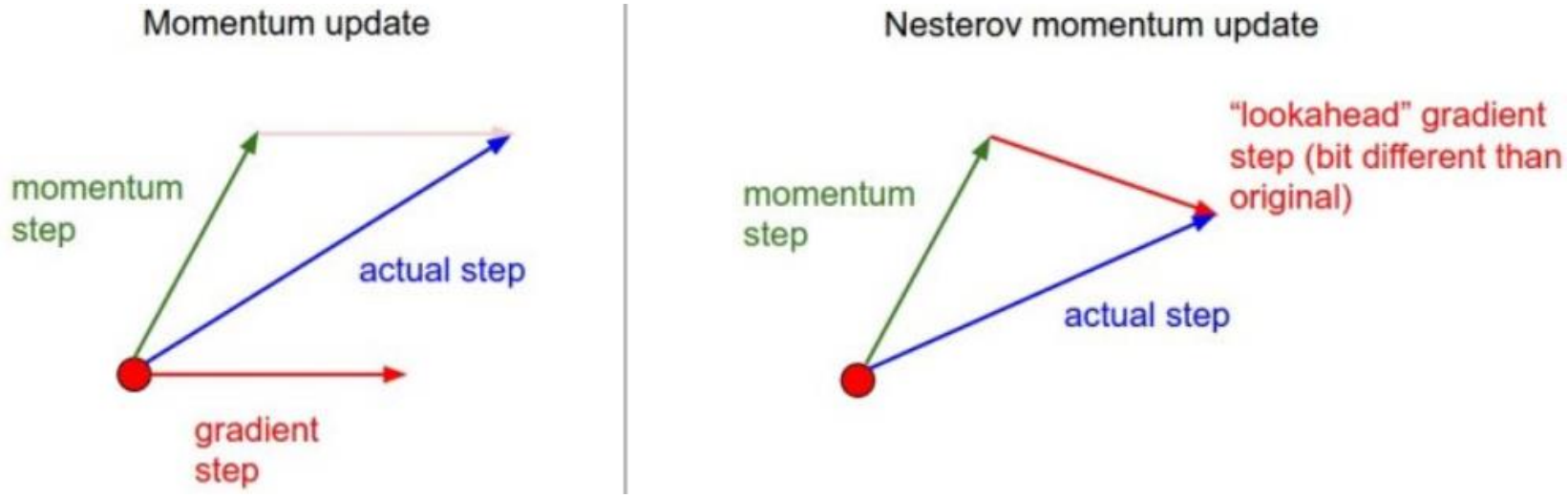
- Ning Qian, 1999
- $v_t = \gamma v_{t-1} + \eta \nabla J(\theta, x^*)$
- $\theta_{i+1} = \theta_i - v_t$
- SGD oscillates in ravines (Sutton et al, 1986)



- Figure left without momentum and right with momentum

Nesterov Accelerated Gradient

- Yurii Nesterov, 1983: Less blind than Momentum
- $v_t = \gamma v_{t-1} + \eta \nabla J(\theta - \gamma v_{t-1}, x^*)$
- $\theta_{i+1} = \theta_i - v_t$
-



Adaptive Learning Methods

- Adagrad (Singer et al, 2011): Per-parameter η
  - $\theta_{t,i+1} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} \nabla J\left(\theta_{t,i}, x^*\right)$
  - Dean et al 2006: Adagrad more robust than SGD for sparse data
  - No need to tune learning rate.
- Adadelta (Zeiler 2012): More adaptive and less monotonic-decreasing
  - $E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$
  - $E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1-\gamma) \Delta\theta_t^2$
  - $\theta_{t,i+1} = \theta_{t,i} - \frac{\sqrt{E[\Delta\theta^2]_{t-1} + \varepsilon}}{\sqrt{E[g^2]_t + \varepsilon}} \nabla J\left(\theta_{t,i}, x^*\right)$

Adaptive Learning Methods Contd.

- Adaptive Moment Estimation (Kingma and Lei Ba, 2015):
- $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$
- $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
- $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$
- $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$
- $\theta_{t+1} = \theta_t - \frac{\eta}{\varepsilon + \sqrt{\hat{v}_t}} \hat{m}_t$
- Default values for hyper-parameters work well in practice

Studying Update step

- Finding minimum of Beale function:
- $(1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$
- Multimodal, saddle points

Algorithm	x1,x2	F(x)	# Steps
SGD	(-2.51,1.30)	0.97	78
Momentum	(-2.5,1.3)	1	496
NAG	(-2.5,1.3)	**0.9**	**56**
Adadelta	(-1.4,1.4)	1.3	3148
Adam	(-1.6,1.5)	1.7	6362

Simulations

- $Y = \beta_0 + \mathrm{X}\beta_1 + \mathrm{N}(0, \sigma^2)$
- 1000 data points: 75-25 training-test

Algorithm	MSE	# Steps
SGD	852	12449
Momentum	8.2	2411
NAG	**2.78**	**2756**
Adadelta	4e+9	Stopped
Adam	**1.57**	20976

Conclusions

- Tuning hyper-parameters is not as easy as just backtracking
- Adaptive Learning algorithms sometimes can’t be tuned with backtracking
- NAG is fast and robust. Adam is more expensive and slower to converge and cumbersome to tune.
- Something seems wrong with Adadelta. It is the one algorithm doesn’t need tuning!

Future Work

- Fix Adadelta
- Apply Stochastic Gradient Descent Algorithms to real data