Preconditioning Kernel Matrices

By:
Divija Swetha Gadiraju (2018802001)

Prathyusha Akundi (2018701014)

Course Project
Topics in Applied Optimization
International Institute of Information Technology, Hyderabad, India

IIIT, HYDERABAD

December 5, 2018

Outline

1. Introduction
2. Motivation and Need for Preconditioning
   - Gaussian Processes (GP)
   - Preconditioned Conjugate Gradient Algorithm
   - Non-Gaussian Likelihoods
3. Preconditioning Kernal Matrices
   - Nystrom type approximation
   - Block Jacobi approximation
   - Regularization
4. Comparison of Preconditioners
5. Impact of preconditioning on GP learning
6. Conclusion Remarks
7. Important References

Introduction and Motivation

IIIT, HYDERABAD

December 5, 2018

Introduction

- **Kernel Machines** comprise of an important class of tools throughout machine learning and statistics, typically used in support vector machines and Gaussian Processes (GP).

- Need to solve linear systems involving Gram matrix

$$K = \{k(x_i, x_j|\theta)\}_{i,j=1,..,n} \tag{1}$$

where the kernel function k, parameterized by θ, data points x_i.

- Computational bottleneck
  - storing K is $O(n^2)$
  - solving a linear system with K is $O(n^3)$

Introduction

- Standard approaches to kernel machines involves factorization (Cholesky) of K
  - quadratic storage and cubic time costs
- Approximate methods exploit structure of kernel
  - a severe loss of accuracy
- Alternative to factorization is the **conjugate gradient (CG) method**
  - directly solve linear systems using a sequence of matrix vector products
  - run-time improvements and eliminates the storage burden with a good kernel structure
  - Otherwise, $O(n^3)$ degradation of run-time performance than factorization
- Apply **preconditioners** to improve the slow convergence of CG

Contribution

1. Apply a broad range of Kernel matrix approximations as preconditioners
2. Learn kernel parameters and make prediction in GP
3. Extend stochastic gradient learning for GP to allow any likelihood that factorizes over the data points
   - developing unbiased estimate of gradient of the approximate log-marginal likelihood
   - demonstrated by using PCG for GP classification
4. A trade-off between accuracy and computational effort
   - PCG has performance improvement beyond state-of-art approximation and factorization approaches

Motivating Example: Gaussian Process

- **GP:** Collection of random variables with property that any finite number of them is jointly Gaussian distributed
- $X = \{x_1, .., x_n\}$ are n input vectors and $y = \{y_1, ..y_n\}^T$ are their labels
- Kernel function determines the covariance of the random variables

$$\text{cov}(f(x), f'(x)) = k(x, x'|\theta) \tag{2}$$

- Radical Basis Function (RBF) Kernel

$$k(x, x'|\theta) = \sigma^2 \text{exp}[\frac{-1}{2} \sum_{r=1}^{d} \frac{(x_i - x_j)_r^2}{l_r^2}] \tag{3}$$

- Assume zero mean GP and $\boldsymbol{f}$=$(f_1, ..f_n)^T$. Observations are modeled through a transformation h of a set of GP-distributed latent variables

$$y_i \sim p(y_i|h(f_i)), \qquad \mathbf{f} \sim N(\mathbf{f}|\mathbf{0}, K) \tag{4}$$

The need for Preconditioning

- Log-marginal likelihood of GP model

$$\log[p(y|\theta, X)] = \frac{-1}{2}\log(|K_y|) - \frac{1}{2}y^T K_y^{-1} y + \text{const} \quad (5)$$

- derivative with respect to kernel parameter θ

$$g_i = \frac{-1}{2}\text{Tr}\frac{K_y^{-1}\partial K_y}{\partial \theta_i} + \frac{1}{2}y^T \frac{K_y^{-1}\partial K_y}{\partial \theta_i} K_y^{-1} y \quad (6)$$

  where $K_y = K + \lambda I$
- Traditional Approach: factorize $K_y = LL^T$ using Cholesky Algorithm ($O(n^3)$)
  - The solution of linear system is required to compute variance at every test point and not viable for large n
  - approaches to approximate the computations lead to approximate values
- Avoiding approximations: for parameter optimization, obtain unbiased estimate of g_i

The need for Preconditioning

- Using stochastic linear algebra result

$$\mathrm{Tr}\left(\frac{K_y^{-1}\partial K_y}{\partial \theta_i}\right) \approx \frac{1}{N_r}\sum_{i=1}^{N_r} r^{(i)^T}\frac{K_y^{-1}\partial K_y}{\partial \theta_i} r^{(i)} \tag{7}$$

- From (7), to calculate stochastic gradients, we need to efficiently solve linear systems
- Linear systems are iteratively solved using CG
  - need not store K_y and $O(n^2)$
- Convergence of CG depends on condition number $\kappa(K_y)$
- Preconditioning is used to improve the conditioning of a matrix, in turn improves the convergence
- Preconditioning matrix P
  - $P^{-1}K_y$ approximates the identity matrix I

The Preconditioned CG Algorithm [1]

Algorithm 1 The Preconditioned CG Algorithm, adapted from (Golub & Van Loan, 1996)

Require: data X, vector $\mathbf{v}$, convergence threshold ϵ, initial vector $\mathbf{x}_0$, maximum no. of iterations T

$\mathbf{r}_0 = \mathbf{v} - K_{\mathbf{y}}\mathbf{x}_0$; $\mathbf{z}_0 = P^{-1}\mathbf{r}_0$; $\mathbf{p}_0 = \mathbf{z}_0$

for i = 0 : T **do**

$\alpha_i = \frac{\mathbf{r}_i^T \mathbf{z}_i}{\mathbf{r}_i^T K_{\mathbf{y}} \mathbf{z}_i}$

$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{p}_i$

$\mathbf{r}_{i+1} = \mathbf{r}_i + \alpha_i K_{\mathbf{y}} \mathbf{p}_i$

if $\|\mathbf{r}_{i+1}\| < \epsilon$ **then**

return $\mathbf{x} = \mathbf{x}_{i+1}$

end if

$\mathbf{z}_{i+1} = P^{-1}\mathbf{r}_{i+1}$

$\beta_i = \frac{\mathbf{r}_{i+1}^T \mathbf{r}_{i+1}}{\mathbf{r}_i^T \mathbf{r}_i}$

$\mathbf{p}_{i+1} = \mathbf{p}_{i+1} + \beta_i \mathbf{p}_i$

end for

Non-Gaussian Likelihoods

- The likelihood $p(y_i|f_i)$ is not Gaussian
- For non-Gaussian likelihood, choose Laplace approximation
- Preconditioning and stochastic gradient approximation within Laplace approximation to compute stochastic gradients for non-conjugate models
- Define a diagonal matrix W, $W = -\bigtriangledown_f \bigtriangledown_f \log[p(y|f)]$ and the linear systems solved involve the matrix B, $B = I + W^{\frac{1}{2}} K W^{\frac{1}{2}}$ is solved using CG or PCG
- Laplace approximation yields the mode $\tilde{f}$ of the posterior over the latent variables and log-marginal likelihood:

$$\log[\tilde{p}(y|\theta, X)] = \frac{-1}{2} \log|B| - \frac{1}{2}\tilde{f}^T K^{-1}\tilde{f} + \log[p(y|\tilde{f})] \quad (8)$$

- Unbiasedly estimate using the stochastic approximation of trace.

Preconditioning Kernel Matrices

IIIT, HYDERABAD

December 5, 2018

Preconditioning Kernel Matrices

Preconditioners for K_y

Apply left preconditioning to solve $K_y Z = v$, the formulation becomes $P^{-1} K_y z = P^{-1} v$

Nystrom Approximation

- To approximate eigen decomposition of kernel matrices and obtains a low rank approximation of K
- Selects a set, U with $m << n$ data (inducing points) to approximate the spectrum of K: $\tilde{K} = K_{XU} K_{UU}^{-1} K_{UX}$
- The preconditioner $P = \tilde{K} = K_{XU} K_{UU}^{-1} K_{UX} + \lambda I$ which is inverted using matrix inversion lemma

$$P^{-1} v = \lambda^{-1} [I - \tilde{K} = K_{XU} (K_{UU} + K_{UX} K_{XU})_{UU}^{-1} K_{UX}] v \quad (9)$$

- Requires $O(m^3)$

Fully and Partially Independent Training Conditional

- The use of subset data for approximating GP Kernel in FITC and PITC
- Covariance of the approximation for FITC:

$$P = K_{XU}K_{UU}^{-1}K_{UX} + \mathbf{diag}(K - K_{XU}K_{UU}^{-1}K_{UX}) + \lambda I \quad (10)$$

- PITC method, no dependence between inducing points in different blocks

$$P = K_{XU}K_{UU}^{-1}K_{UX} + \mathbf{bldiag}(K - K_{XU}K_{UU}^{-1}K_{UX}) + \lambda I \quad (11)$$

Approximate factorization of kernel matrices

- Approximations to K that factorize as $\tilde{K} = \phi\phi^T$

$$P^{-1}v = (\phi\phi^T + \lambda I)^{-1}v = \lambda^{-1}[I - \phi(I + \phi^T\phi)^{-1}\phi^T]v \quad (12)$$

- Explore different ways of determining ϕ
  - P can be inverted at lower cost than the original Kernel matrix K
- Next, we review methods to approximate K in the form $\phi\phi^T$

Spectral Approximation

- Uses Fourier features for deriving a sparse approximation of a GP
- The elements of K are approximated as:

$$\tilde{K}_{i,j} = \frac{\sigma_0^2}{m}\phi(x_i)^T\phi(x_j) = \frac{\sigma_0^2}{m}\sum_{r=1}^{m}\cos[2\pi s_r^T(x_i - X_j)] \quad (13)$$

- s_r are spectral points which are sampled values of RBF kernel

Partial SVD and Structured Kernel Interpolation (SKI)

Partial SVD

- SVD factorizes K into $A \wedge A^T$, A is unitary and $\wedge$ is a diagonal matrix
- Randomized truncated SVD constructs low rank factors of K using random sampling to accelerate computing

Structured Kernel Interpolation (SKI)

- Exploits Kronecker matrix-vector multiplications
- A grid of inducing points, U and the covariance between the training data and U is $K_{XU} = WK_{UU}$. W is a sparse interpolation matrix.
- Preconditioner is $P = WK_{UU}W^T + \lambda I$
- Let $V = W/\sqrt{\lambda}$ then, $P^{-1} = \lambda^{-1}(VK_{UU}V^T + I)^{-1}$
- Solve inner loop (linear system) by CG, all within one iteration of outer loop PCG
- Less than $O(n^2)$

Block Jacobi and Regularization

Block Jacobi

- Instead of using a single subset of data, construct local GPs over segments of data
- Preconditioner, $P = \textbf{bldiag}(K_y + \lambda I)$
- Inverse is computaionally cheap
- Information is covariance matrix is ignored

Regularization

- Adding noise to diagonal of K_y makes it better conditioned
- $P = K_y + \delta I$
- Condition number decreases with increasing δ
- Uses right preconditioning $K_y P^{-1}(Px) = v$
- Linear system is solved using CG at every outer iteration of PCG

Comparision of Preconditioners

IIIT, HYDERABAD

December 5, 2018

Comparison of Preconditioners

- The quality of preconditioners based on how many matrix-vector products they require
- Convergence threshold: $\epsilon^2 = n * 10^{-10}$, accepting an average error of 10^{-5} on each element of solution
- Nystrom-type method: $m = \sqrt{n}$ inducing points, preconditioner in $O(m^3) = O(n^{3/2})$
- SKI: equal number of elements on grid for each dimension, Kronecker products cost $O(dn^{\frac{d+1}{d}})$, preconditioner in $O(n^{3/2})$
- Regularization: Diagonal offset δ is two orders of magnitude greater than the noise process
- RBF Kernel with variance, $\sigma^2 = 1$
- Length parameter l and noise variance λ are plotted in $\log_{10}$ scale and maximum iterations are 100,000

$$\log_{10}\left(\frac{\#\text{PCG iterations}}{\#\text{CG iterations}}\right) \tag{14}$$

Comparison of Preconditioners

$n = 1030$ and $d = 8$

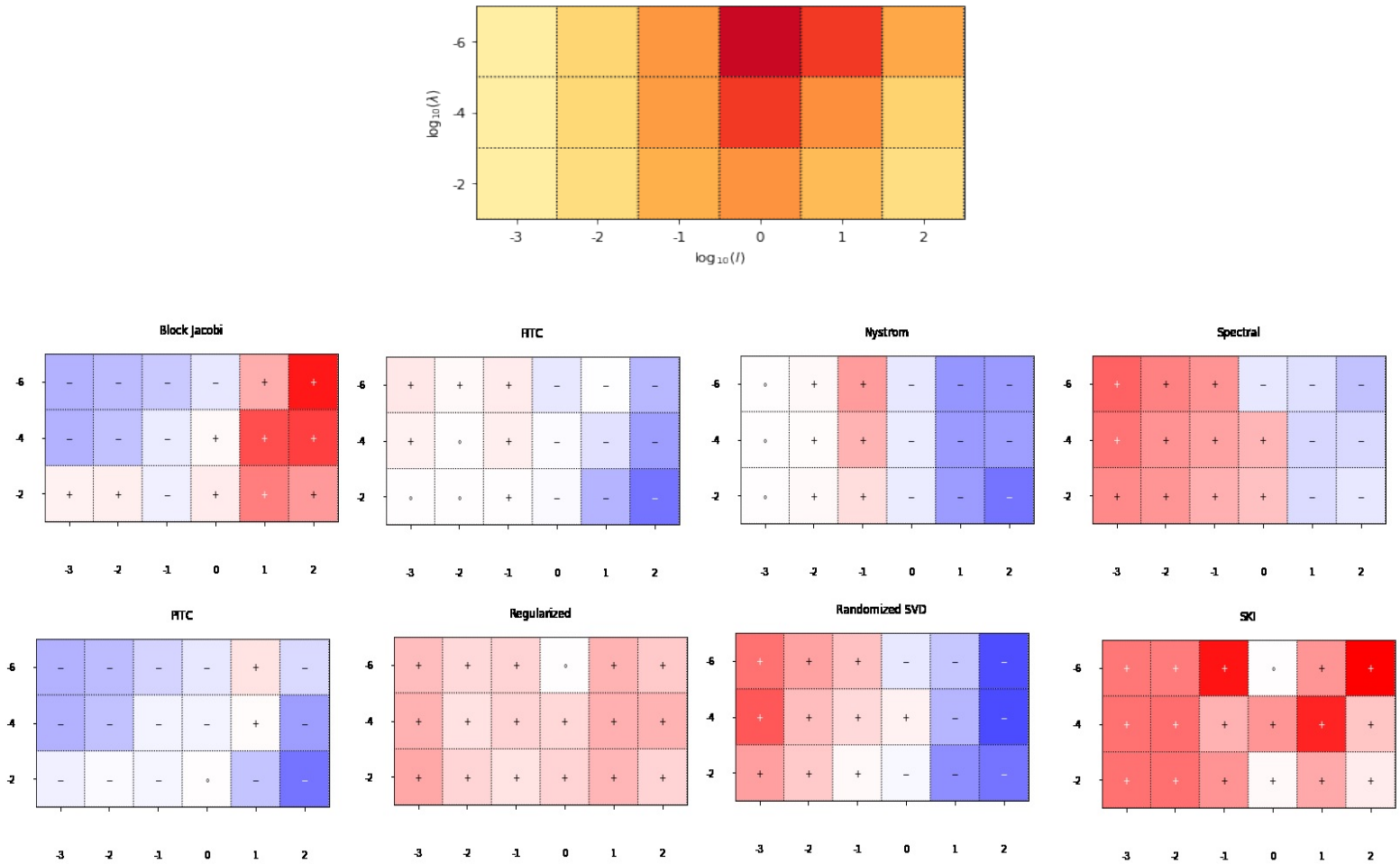

Figure: Concrete Dataset

Comparison of Preconditioners

$n = 9568$ and $d = 4$

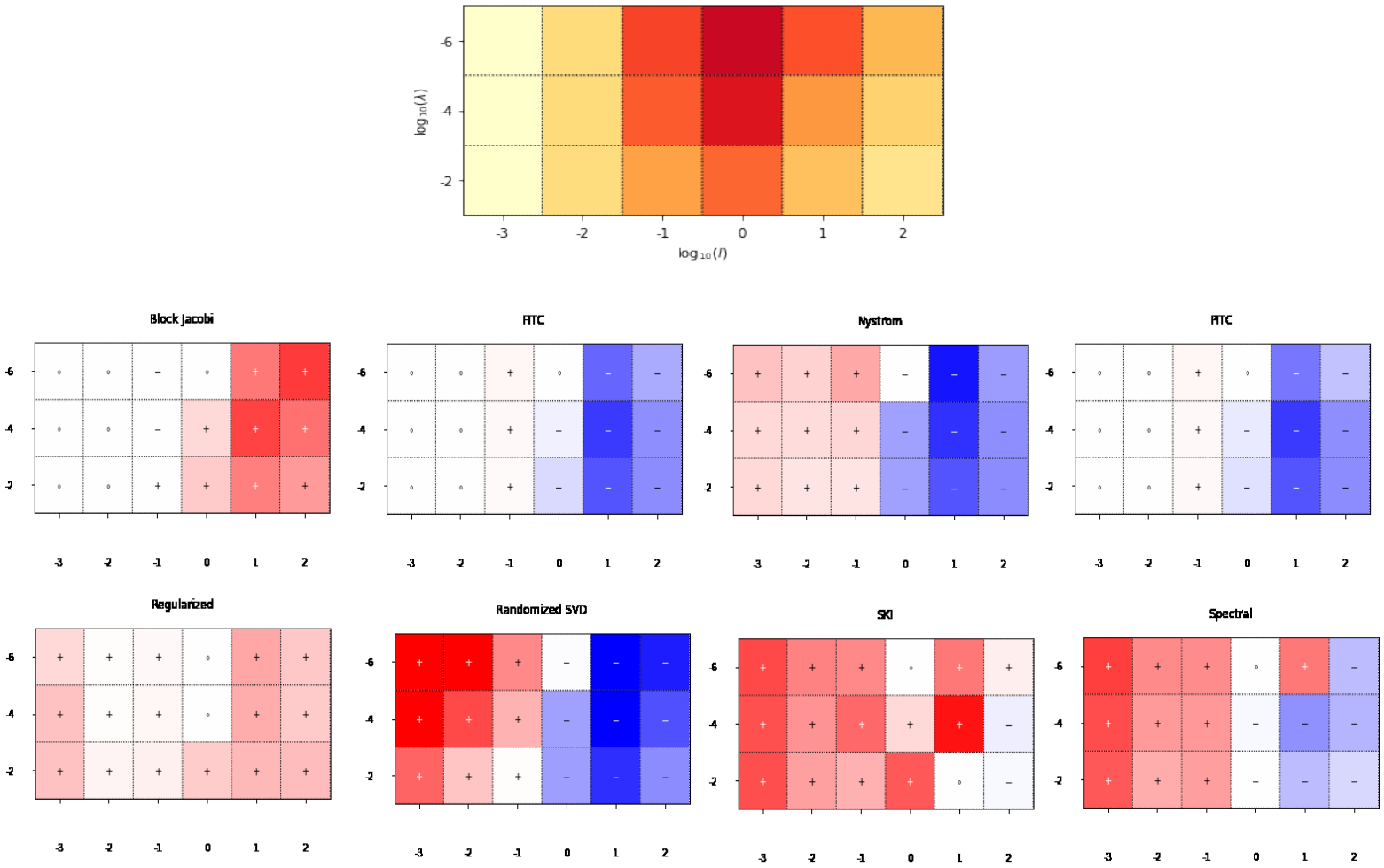

Figure: Power Plant Dataset

Comparison of Preconditioners

$n = 45730$ and $d = 9$

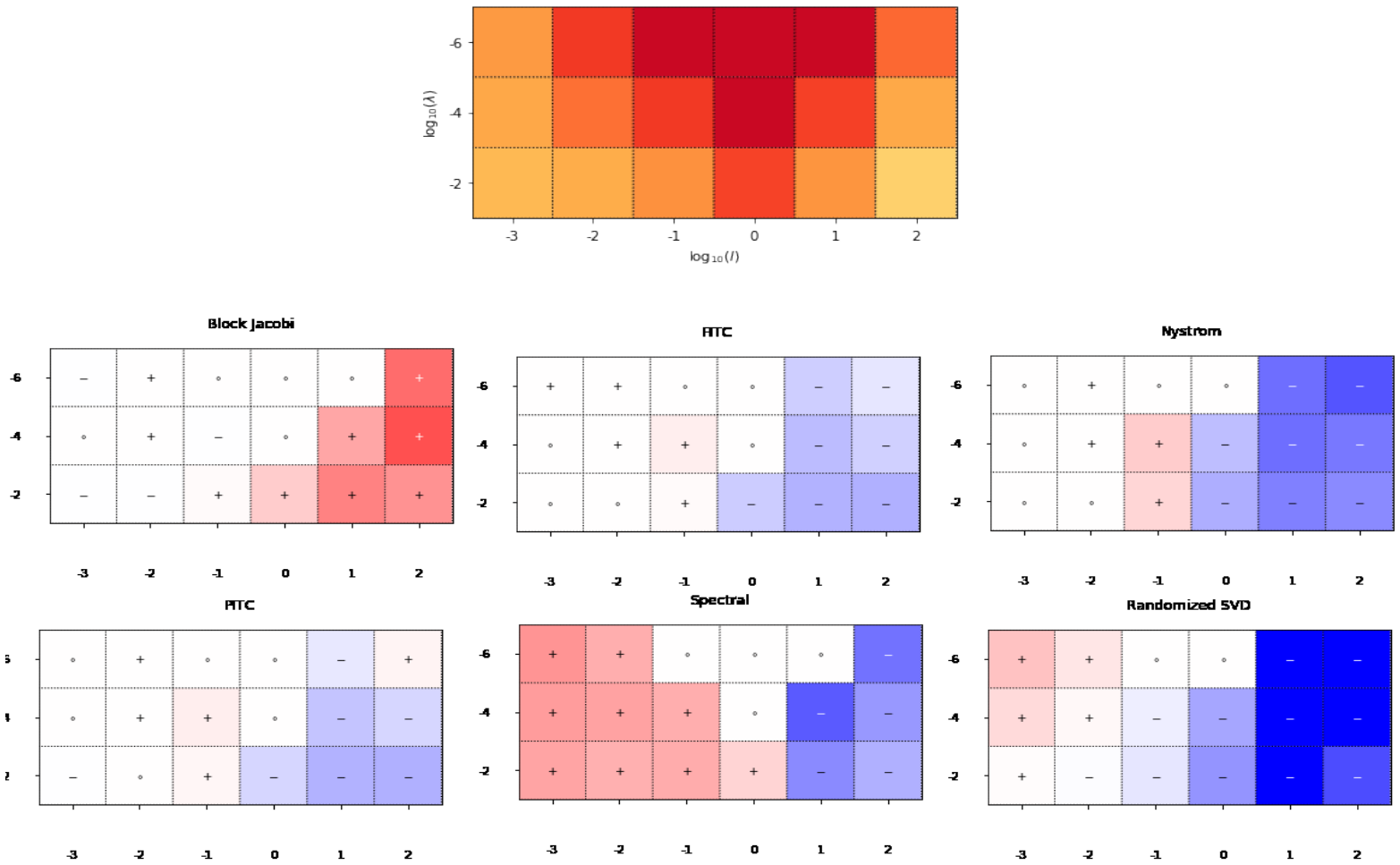

Figure: Protein Dataset

Impact of Preconditioning on GP learning

IIIT, HYDERABAD

December 5, 2018

Impact of Preconditioning on GP learning

- Preconditioning is employed in GP regression and classification
- Given predictive mean and variance for the test points, two errors are measured

$$\text{RMSE} = \sqrt{\frac{1}{N_{test}} \sum_{i=1}^{N_{test}} (m_i - y_i)^2} \tag{15}$$

- Nystrom Preconditioning method is used
- $m = 4\sqrt{n}$ points are randomly selected from input data
- Also evaluated the performance of approximate GP methods (FITC, PITC)

Impact of Preconditioning on GP learning

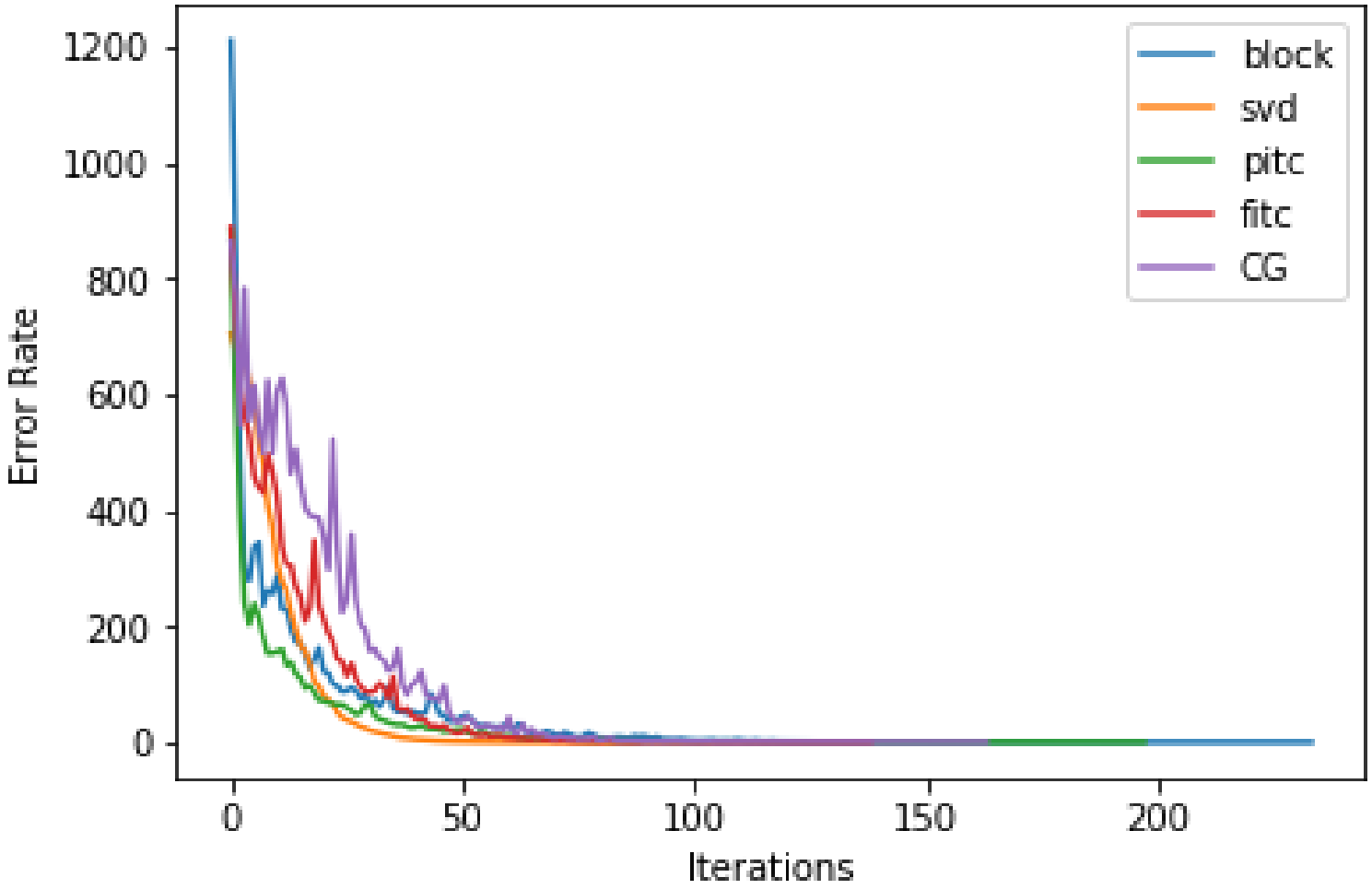

Figure: Concrete Dataset

Impact of Preconditioning on GP learning

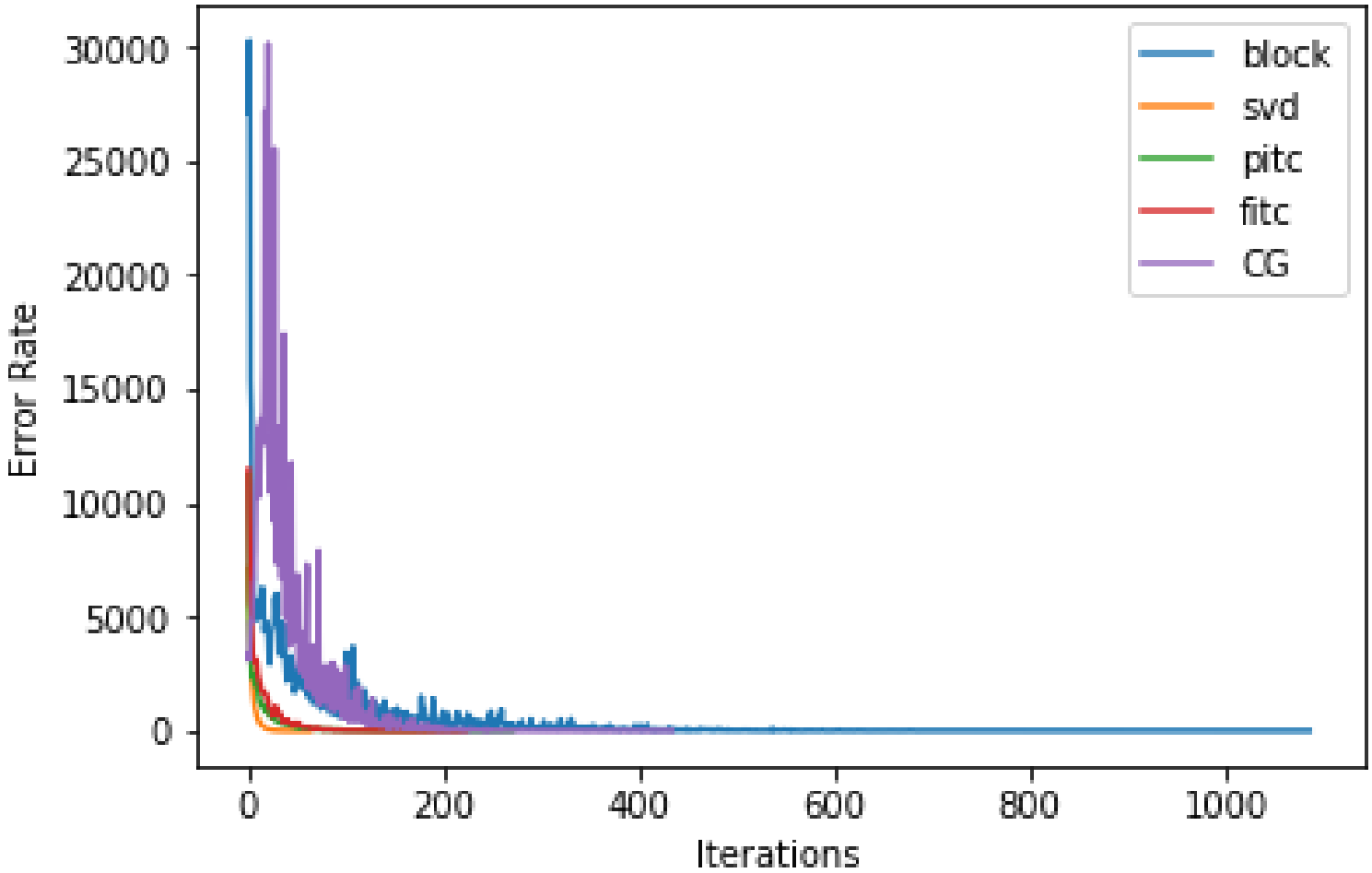

Figure: Power Plant Dataset

Impact of Preconditioning on GP learning

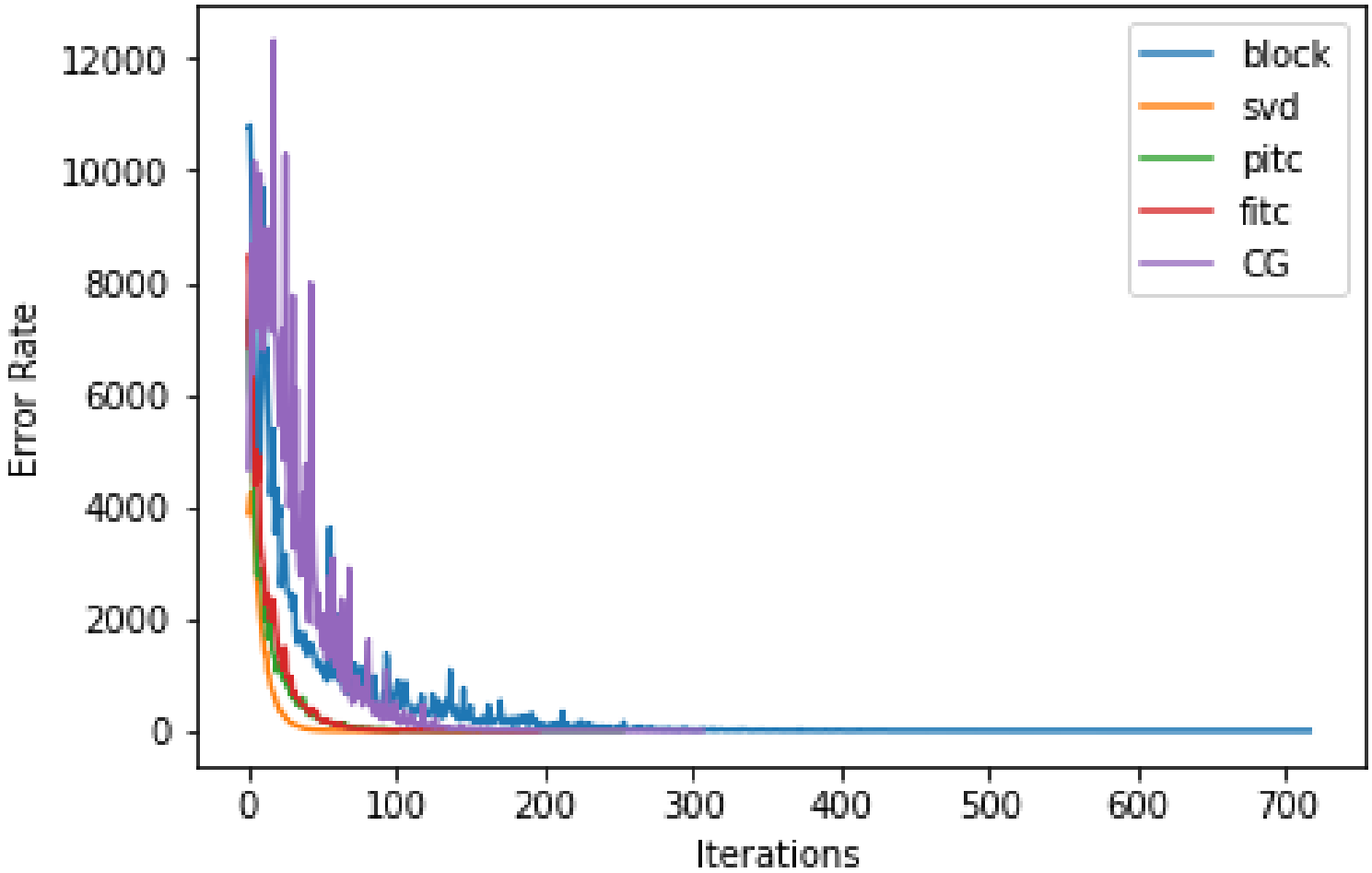

Figure: Protein Dataset

Discussion and Conclusions

IIIT, HYDERABAD

December 5, 2018

Conclusion

- Preconditioning enables the use of iterative approaches for optimization of kernel parameters in GPs.
- When data and thus the kernel size becomes large, PCG is the optimal choice
- Future Research Direction: Computing the elements of K matrix on the fly so that we no longer need to store any objects.

Important References

1. Golub, G. H. and Van Loan, C. F. Matrix computations. The Johns Hopkins University Press, 3rd edition, 1996.
2. Anitescu, M., Chen, J., and Wang, L. A Matrix-free Approach for Solving the Parametric Gaussian Process Maximum Likelihood Problem. SIAM Journal on Scientific Computing, 34(1):A240–A262, 2012.
3. Davies, A. Effective Implementation of Gaussian Process Regression for Machine Learning. PhD thesis, University of Cambridge, 2014.
4. Filippone, M. and Engler, R. Enabling scalable stochastic gradient-based inference for Gaussian processes by employing the Unbiased LInear System SolvEr (ULISSE). In Blei, D. and Bach, F. (eds.), Proceedings of The 32nd International Conference on Machine Learning, volume 37 of JMLR Proceedings, pp. 1015–1024, 2015.
5. Gibbs, M. N. Bayesian Gaussian processes for regression and classification. PhD thesis, University of Cambridge, 1997.

THANK YOU!

IIIT, HYDERABAD

December 5, 2018