Preconditioning Kernel Matrices

K. Cutajar[1], M. A. Osborne[2], J. P. Cunningham[3], M. Filippone[1]

1 - EURECOM, Sophia Antipolis, France
2 - University of Oxford, Oxford, UK 3 - Columbia University, New York City, USA

Kernel Machines and Solving Linear Systems

- Operate in a high-dimensional, implicit feature space;
- Rely on the construction of an $n \times n$ Gram matrix K;
- Popular kernels:
  - RBF : $k(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \exp\left(-\frac{1}{2}d^2\right)$;
  - Matérn : $k_{v=\frac{3}{2}}(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2\left(1+\sqrt{3}d\right)\exp\left(-\sqrt{3}d\right)$;

    where $d^2 = (\mathbf{x}_i - \mathbf{x}_j)^\top \Lambda (\mathbf{x}_i - \mathbf{x}_j)$.
- Involve the solution of linear systems $K\mathbf{z} = \mathbf{v}$;
- Cholesky Decomposition:
  - $\mathcal{O}(n^2)$ space and $\mathcal{O}(n^3)$ time - unfeasible for large n.
- Conjugate Gradient (CG):
  - Numerical solution of linear systems constructed using matrix-vector multiplications;
  - $\mathcal{O}(tn^2)$ for t CG iterations - in theory $t = n$ (possibly worse).
- Preconditioned Conjugate Gradient (PCG):
  - Transforms linear system to be better conditioned, improving convergence;
  - Yields a new linear system of the form $P^{-1}K\mathbf{z} = P^{-1}\mathbf{v}$;
  - $\mathcal{O}(tn^2)$ for t PCG iterations - in practice $t \ll n$.

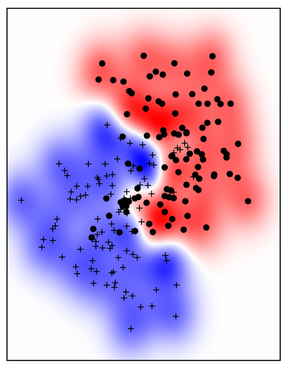

Fig. 1: Kernel machines enable non-linear separation of data.

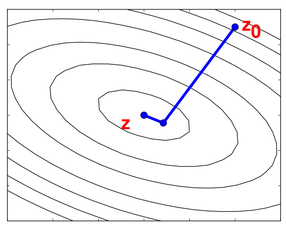

Fig. 2: CG

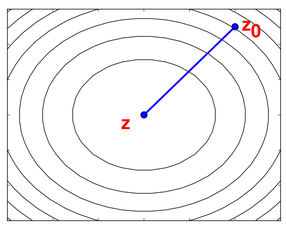

Fig. 3: Preconditioned CG

Preconditioning Approaches

- Suppose we want to precondition $K_\mathbf{y} = K + \lambda I$;
- Our choice of preconditioner should:
  - Approximate $K_\mathbf{y}$ as closely as possible;
  - Be easy to invert.
- For low-rank preconditioners we employ the Woodbury inversion lemma:

$K_\mathbf{y}$ = + P = +

P^{-1} = $^{-1}$ − $^{-1}$ [$^{-1}$ + $^{-1}$] $^{-1}$ $^{-1}$

- For other preconditioners we solve inner linear systems once again using CG!

	Formulation	Strategy
Nyström	$P = K_{XU}K_{UU}^{-1}K_{UX} + \lambda I$ where $U \subset X$	Woodbury
FITC	$P = K_{XU}K_{UU}^{-1}K_{UX} + \text{diag}\left(K - K_{XU}K_{UU}^{-1}K_{UX}\right) + \lambda I$	Woodbury
PITC	$P = K_{XU}K_{UU}^{-1}K_{UX} + \text{bldiag}\left(K - K_{XU}K_{UU}^{-1}K_{UX}\right) + \lambda I$	Woodbury
Spectral	$P_{ij} = \frac{\sigma^2}{m}\sum_{r=1}^{m}\cos\left[2\pi\mathbf{s}_r^\top(\mathbf{x}_i - \mathbf{x}_j)\right] + \lambda I_{ij}$	Woodbury
Partial SVD	$K = A\Lambda A^\top \quad \Rightarrow \quad P = A_{[\cdot,1:m]}\Lambda_{[1:m,1:m]}A^\top_{[1:m,\cdot]} + \lambda I$	Woodbury
Block Jacobi	$P = \text{bldiag}(K) + \lambda I$	Block Inverse
SKI	$P = WK_{UU}W^\top + \lambda I$ where K_{UU} is Kronecker	Inner CG
Regularization	$P = K + \delta I + \lambda I$	Inner CG

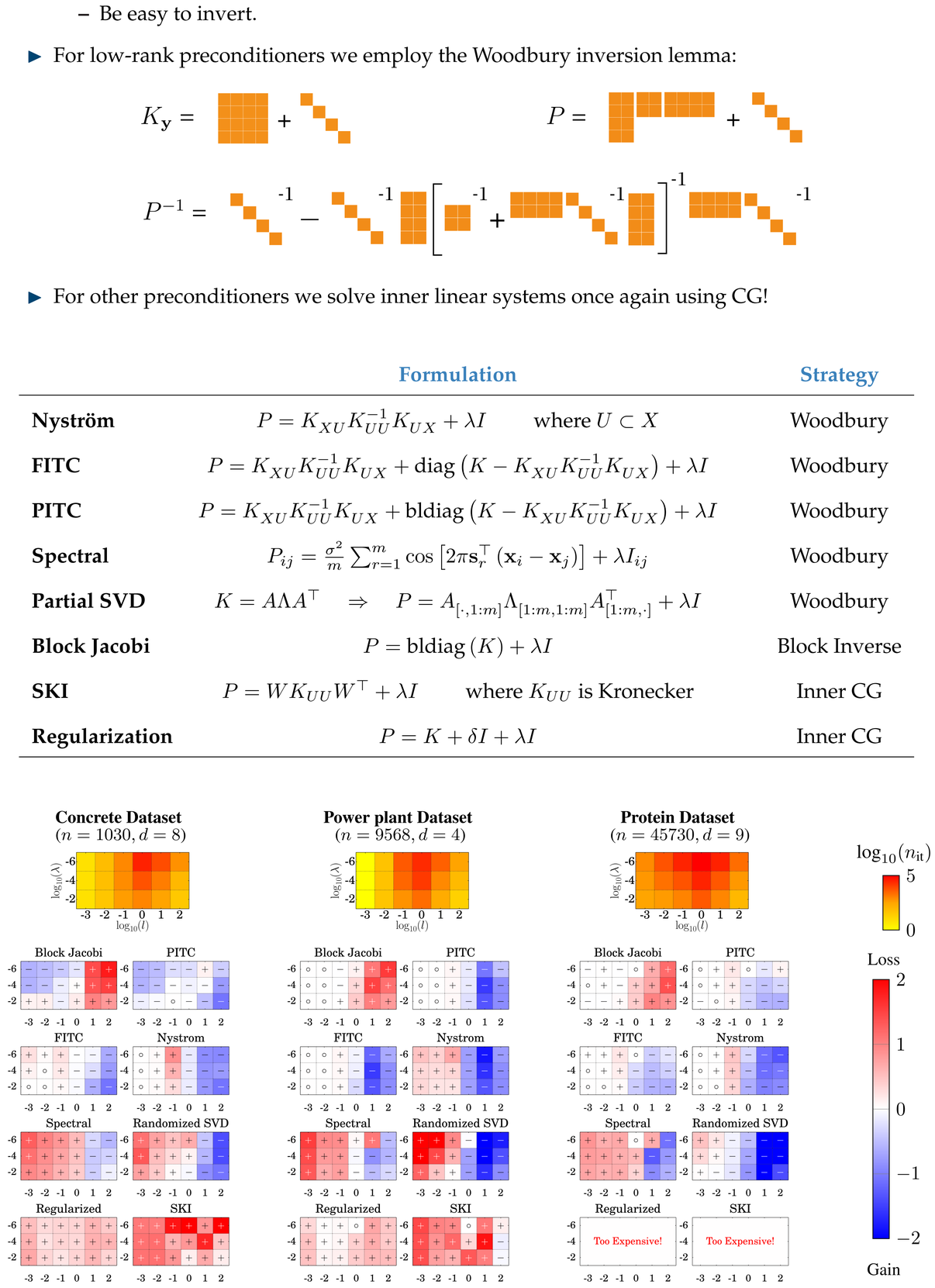

Fig. 4: Comparison of preconditioners for different settings of kernel parameters across multiple datasets. Top: Number of iterations required to solve the corresponding linear system using CG. Bottom: Rate of improvement (blue) or degradation (red) achieved by using PCG to solve the same linear system.

Motivating Example - Gaussian Processes

- Marginal likelihood:

$$\log[p(\mathbf{y}|\text{par})] = -\frac{1}{2}\log|K_\mathbf{y}| - \frac{1}{2}\mathbf{y}^\top K_\mathbf{y}^{-1}\mathbf{y} + \text{const.}$$

- Derivatives wrt par:

$$\frac{\partial \log[p(\mathbf{y}|\text{par})]}{\partial \text{par}_i} = -\frac{1}{2}\text{Tr}\left(K_\mathbf{y}^{-1}\frac{\partial K_\mathbf{y}}{\partial \text{par}_i}\right) + \frac{1}{2}\mathbf{y}^\top K_\mathbf{y}^{-1}\frac{\partial K_\mathbf{y}}{\partial \text{par}_i}K_\mathbf{y}^{-1}\mathbf{y}$$

- Stochastic estimate of the trace - assuming $\mathrm{E}[\mathbf{r}\mathbf{r}^\top] = I$, then:

$$\text{Tr}\left(K_\mathbf{y}^{-1}\frac{\partial K_\mathbf{y}}{\partial \text{par}_i}\right) = \text{Tr}\left(K_\mathbf{y}^{-1}\frac{\partial K_\mathbf{y}}{\partial \text{par}_i}\mathrm{E}[\mathbf{r}\mathbf{r}^\top]\right) = \mathrm{E}\left[\mathbf{r}^\top K_\mathbf{y}^{-1}\frac{\partial K_\mathbf{y}}{\partial \text{par}_i}\mathbf{r}\right] \approx \frac{1}{N_\mathbf{r}}\sum_{i=1}^{N_\mathbf{r}}\mathbf{r}^{(i)\top}K_\mathbf{y}^{-1}\frac{\partial K_\mathbf{y}}{\partial \text{par}_i}\mathbf{r}^{(i)}$$

- Linear systems only!
- Laplace approximation for non-Gaussian likelihoods may be formulated in a similar way!

Experimental Setup and Results

- Exact gradient-based optimization using Cholesky decomposition (CHOL);
- Stochastic gradient-based optimization using `ADAGRAD` - using CG and PCG;
- GP Approximations:
  - Variational learning of inducing variables (VAR);
  - Fully Independent Training Conditional (FITC);
  - Partially Independent Training Conditional (PITC).

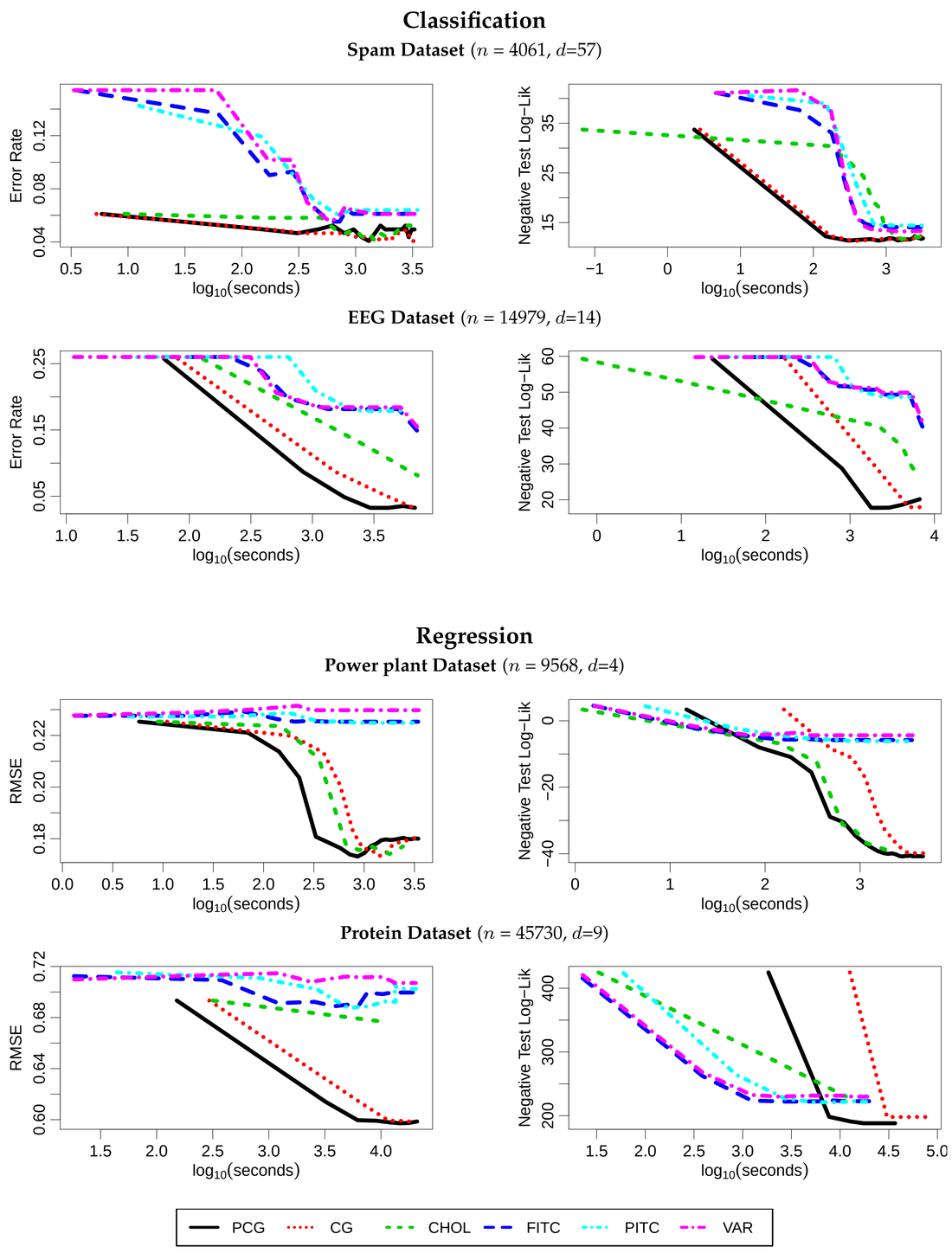

Fig. 5: Error and negative log likelihood on $\sqrt{n}$ held out test data over time. Curves are averaged over multiple repetitions.

Conclusions

- Our solution:
  - ✓ Exact in the limit of iterations;
  - ✓ Straightforward to construct and easy to tune;
  - ✓ Scalable to large datasets - no need to store K;
  - ✓ Competitive with exact Cholesky decomposition;
  - ✓ Superior to approximate methods.
- Ongoing work:
  - Extending this work to other kernel functions and models;
  - Implementation on a distributed framework;
  - Exploiting PCG in the solution of $f(K)\,\mathbf{z} = \mathbf{v}$.