Faster Kernel Ridge Regression Using Sketching and Preconditioning

Haim Avron
Department of Applied Math
Tel Aviv University, Israel

SIAM CS&E 2017
February 27, 2017

Brief Introduction to Kernel Ridge Regression

- Input domain: $\mathcal{X} \subseteq \mathbb{R}^d$, Output domain: $\mathcal{Y} \subseteq \mathbb{R}$.
  Non-deterministic dependency of output $y \in \mathcal{Y}$ on input $\mathbf{x} \in \mathcal{X}$.

- We are given n samples: $\{(\mathbf{x}_i, y_i)\}_{i=1}^n \subseteq \mathcal{X} \times \mathcal{Y}$.

- Goal: try to infer the connection between output and input.

- Regularized least-squares approach:

$$\arg\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2 + \lambda \|f\|_{\mathcal{H}}^2$$

  where $\mathcal{H}$ is an hypothesis space of functions.

Brief Introduction to Kernel Ridge Regression

- Define a symmetric positive definite *kernel* function: $k : \mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{R}$
  Examples:
  - Polynomial Kernel: $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z} + c)^q$
  - Gaussian Kernel: $k(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|_2^2 / \sigma^2)$
- k defines an Hilbert space $\mathcal{H}_k$ of functions from $\mathcal{X}$ to $\mathbb{R}$:

$$\mathcal{H}_k = \overline{\left\{ \sum_{i=1}^{m} \alpha_i k(\mathbf{z}_i, \cdot) \mid \mathbf{z}_i \in \mathcal{X},\, \alpha_i \in \mathbb{R},\, m \in \mathbb{Z}_+ \right\}}$$

- Use $\mathcal{H} = \mathcal{H}_k$.

Brief Introduction to Kernel Ridge Regression

- Representer theorem: the solution has the form
  $f^\star(x) = \sum_{i=1}^n \alpha_i k(\mathbf{x}_i, \mathbf{x})$
- Let $\mathbf{K} \in \mathbb{R}^{n \times n}$ be the *kernel matrix*: $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$
- The optimal coefficients $\alpha = (\alpha_1, \dots, \alpha_n)^T$ are the solution to

$$\arg\min_{\alpha \in \mathbb{R}^n} \|\mathbf{y} - \mathbf{K}\alpha\|_2^2 + \lambda n \alpha^T \mathbf{K} \alpha$$

Equalize the gradient to 0 to find that the solution satisfies

$$(\mathbf{K} + \lambda n \mathbf{I})\alpha = \mathbf{y}.$$

Problem reduces to solving a dense linear system!

Example Use (Advertisement)

- Consider a mis-specified model:

$$\frac{d}{dt}\mathbf{y}(t) = G(\mathbf{y}(t)) + F(\mathbf{y}(t))$$

  where G is known and F is unknown.

- Can recover F from measurements of $\mathbf{y}$ using kernel ridge regression.

- See talk on Friday in MS326 (Friday, 11:45-12:05).

Main Limiting Factor: Scalability

- Kernel ridge regression:

$$(\mathbf{K} + \lambda n \mathbf{I})\boldsymbol{\alpha} = \mathbf{y} \qquad \begin{array}{cc} O(n^2) & \text{storage} \\ O(n^3 + n^2 d) & \text{training} \\ O(nd) & \text{test speed} \end{array}$$

- Too expensive if n is even moderately big!

The Random Features Method

- Approximate low-dimensional feature map $\varphi : \mathbb{R}^d \to \mathbb{R}^s$ such that

$$k(\mathbf{x}, \mathbf{z}) \approx \varphi(\mathbf{x})^T \varphi(\mathbf{z})$$

- Use $\tilde{k}(\mathbf{x}, \mathbf{z}) = \varphi(\mathbf{x})^T \varphi(\mathbf{z})$ as a substitute kernel.

- $\mathbf{Z} = [\varphi(\mathbf{x}_1) \ldots \varphi(\mathbf{x}_n)]^T$ and $\mathbf{w} = \mathbf{Z}^T \alpha$, we have $f^\star(\mathbf{x}) = \mathbf{w}^T \varphi(\mathbf{x})$ and

$$\mathbf{w} = \arg\min \left\| \begin{pmatrix} \mathbf{Z} \\ \sqrt{n\lambda}\mathbf{I} \end{pmatrix} \mathbf{w} - \begin{pmatrix} \mathbf{y} \\ 0 \end{pmatrix} \right\|_2 \qquad \begin{array}{cc} O(ns) & \text{storage} \\ O(ns^2) & \text{training} \\ O(s + maptime) & \text{test speed} \end{array}$$

Obviously, we want small s and good approx. to k (in some sense).

Random Fourier Features (Rahimi and Recht 2007)

- Shift-Invariant kernels:

$$k(\mathbf{x},\mathbf{z}) = \psi(\mathbf{x}-\mathbf{z}),$$

for some *positive definite function* ψ on $\mathbb{R}^d$.

- A consequence of Bochner's theorem (1932-1933): If k is shift-invariant there exist a probability distribution $p(\cdot)$ such that

$$k(\mathbf{x},\mathbf{z}) = \frac{1}{2\pi}\int_{\mathbb{R}^d}\int_0^{2\pi} \cos(\mathbf{x}^T\mathbf{w}+b)\cos(\mathbf{z}^T\mathbf{w}+b)p(\mathbf{w})db\,d\mathbf{w}$$

- Gaussian kernel: $k(\mathbf{x},\mathbf{z}) = e^{-\frac{\|\mathbf{x}-\mathbf{z}\|_2^2}{2\sigma^2}} \Longleftrightarrow p = \mathcal{N}(0,\sigma^{-2}\mathbf{I}_d)$.

Random Fourier Features (Rahimi and Recht 2007)

- Main idea: the integral can be approximated via Monte-Carlo.

- Draw $\mathbf{w}_1, \ldots, \mathbf{w}_s \sim p(\cdot)$ and $b_1, \ldots, b_s \sim U(0, 2\pi)$. Now,

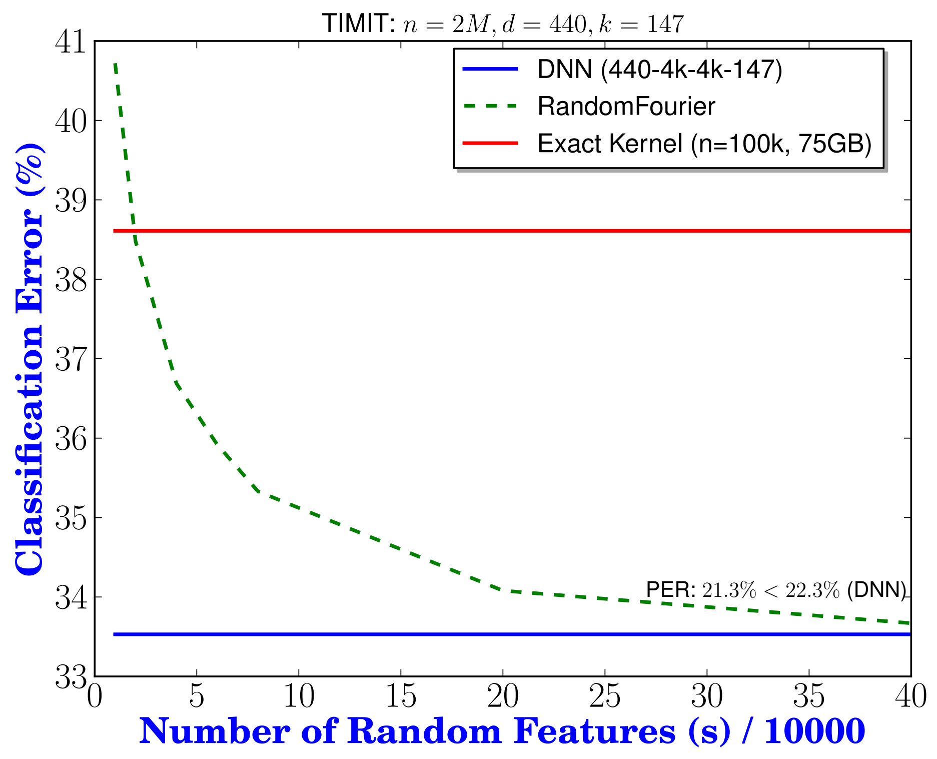
$$k(\mathbf{x}, \mathbf{z}) \approx \frac{1}{s} \sum_{j=1}^{s} \cos(\mathbf{x}^T \mathbf{w}_j + b_j) \cos(\mathbf{z}^T \mathbf{w}_j + b_j)$$

- This defines the feature map:

$$\varphi(\mathbf{x}) \equiv \frac{1}{\sqrt{s}} \left[\cos(\mathbf{w}_1^T \mathbf{x} + b_1) \ldots \cos(\mathbf{w}_s^T \mathbf{x} + b_s)\right]^T \in \mathbb{R}^s \ .$$

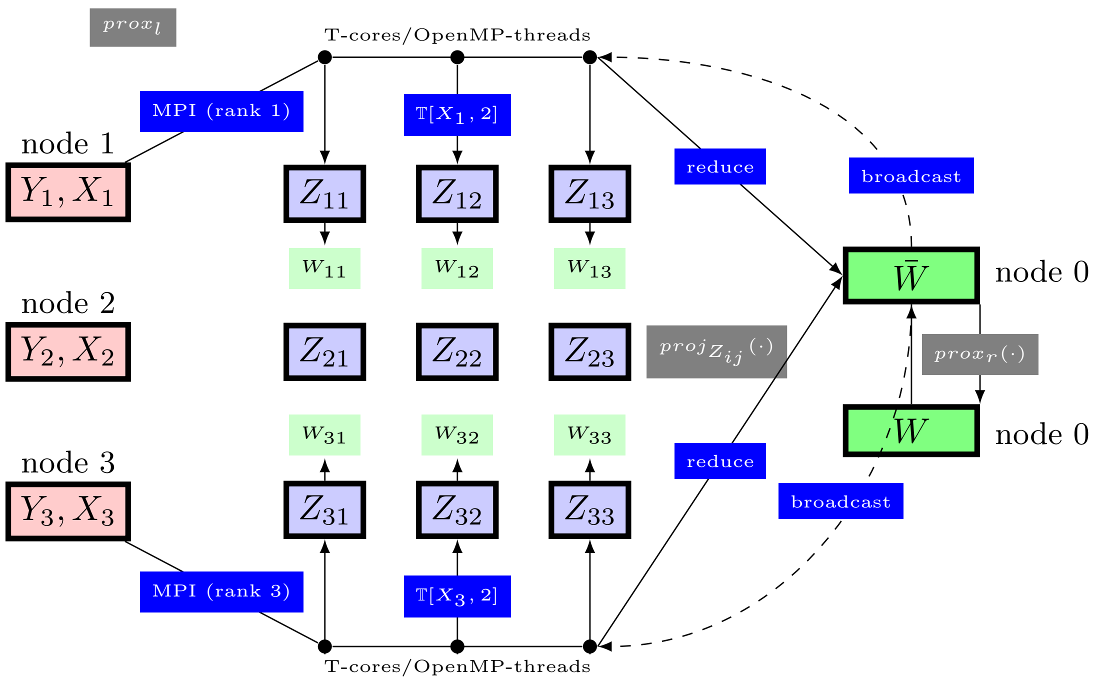
The Success of Random Fourier Features

Example on a Speech Recognition Dataset



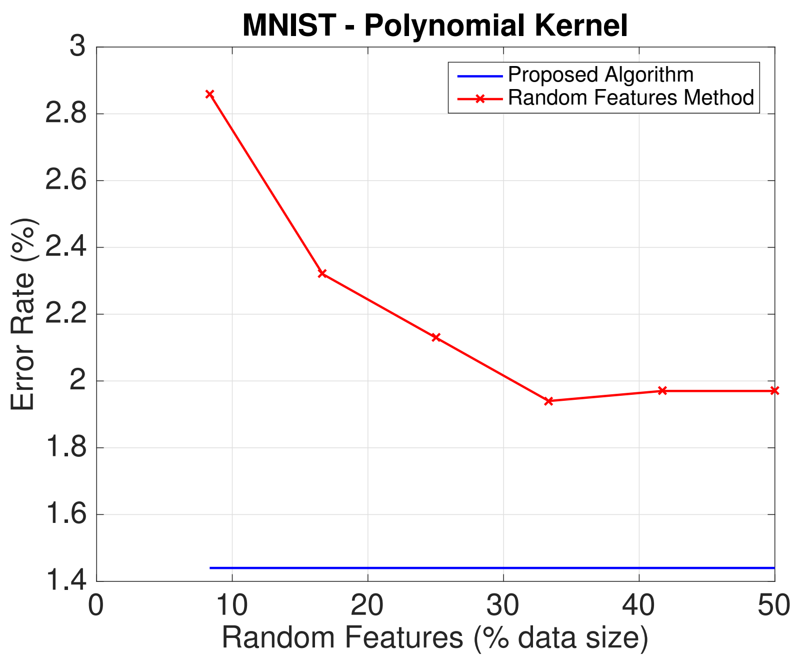
How do you learn with so many features (400K!!!) ?

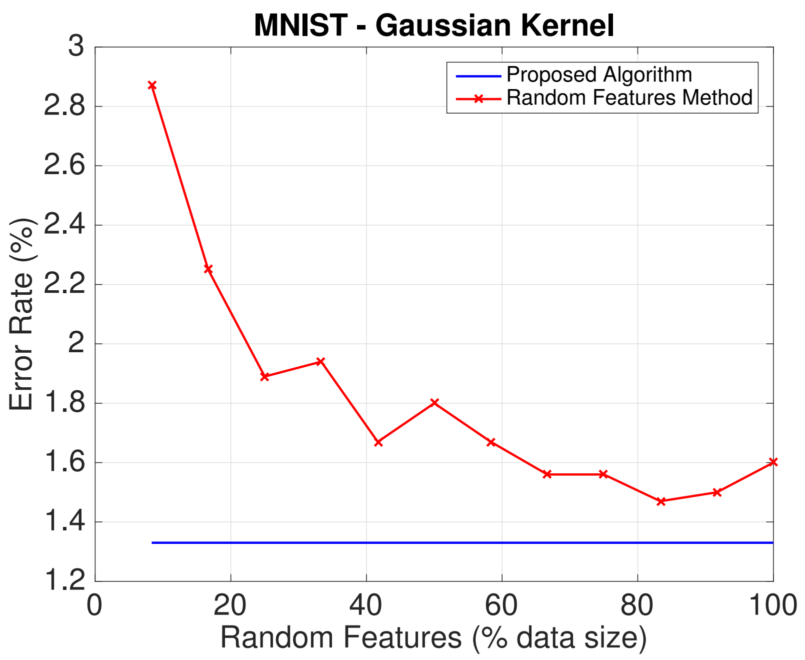
Scalable Kernel Learning Using Random Features: ADMM+Implicit Distributed Optimization



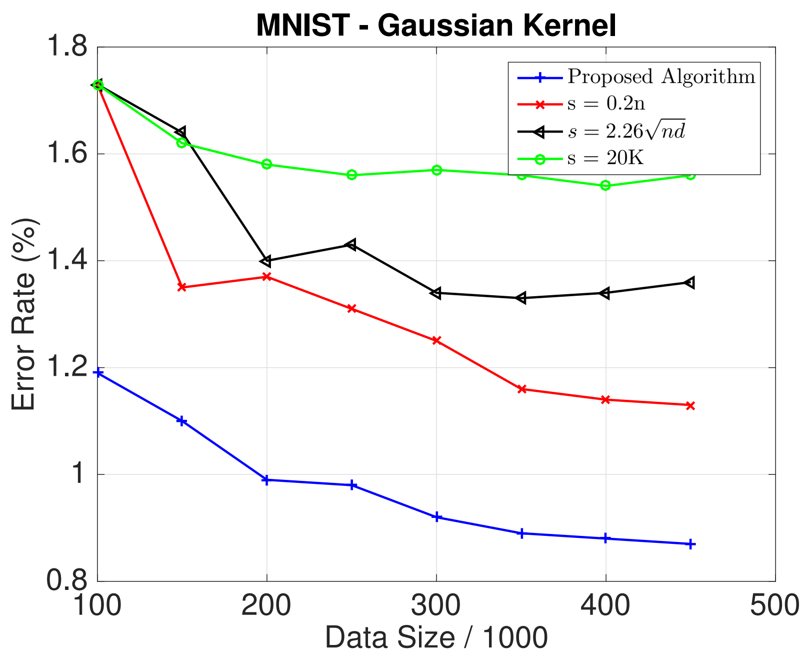
Avron and Sindhwani, High-Performance Kernel Machines With Implicit Distributed Optimization and Randomization", Technometrics 68 (3) 2016

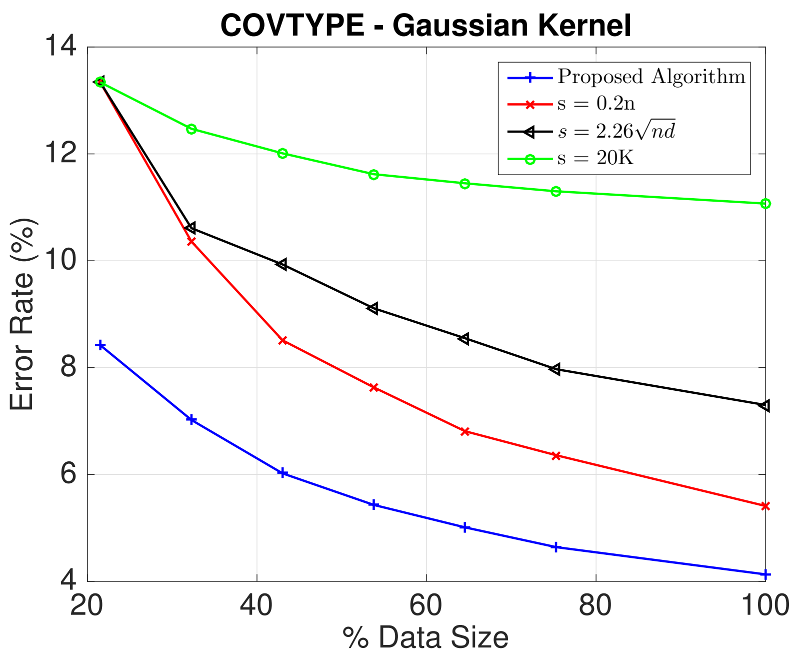
The Price for Scalability?





Is It Really Scalable?





A Better Use for Approximation: Preconditioning

- **Main Idea:** Use random features to accelerate the solution of the linear system, not to approximate it.

- Let $\mathbf{Z} = \left[\varphi(\mathbf{x}_1) \ldots \varphi(\mathbf{x}_n)\right]^T$. We have

$$\mathbf{K} \approx \mathbf{Z}\mathbf{Z}^T \, .$$

- So, for $\lambda \geqslant 0$: $\mathbf{K} + \lambda n \mathbf{I}_n \approx \mathbf{Z}\mathbf{Z}^T + \lambda n \mathbf{I}_n \, ..$
  - And the approximation improves as λ gets larger.

- **We can use $\mathbf{Z}\mathbf{Z}^T + \lambda n \mathbf{I}_n$ to solve $(\mathbf{K} + \lambda n \mathbf{I}_n)\alpha = \mathbf{y}$ faster!**

Random Features Preconditioning

- **Use $\mathbf{Z}\mathbf{Z}^T + \lambda n \mathbf{I}_n$ as a preconditioner for $(\mathbf{K} + \lambda n \mathbf{I}_n)\alpha = \mathbf{y}$.**
- Efficiently applying the preconditioner is easy since:

$$\begin{aligned}(\mathbf{Z}\mathbf{Z}^T + \lambda n \mathbf{I}_n)^{-1} &= n^{-1}\lambda^{-1}(\mathbf{I}_n - \mathbf{Z}(\mathbf{Z}^T\mathbf{Z} + \lambda n \mathbf{I}_n)^{-1}\mathbf{Z}^T) \\ &= n^{-1}\lambda^{-1}(\mathbf{I}_n - \mathbf{U}^T\mathbf{U})\end{aligned}$$

  where $\mathbf{L}\mathbf{L}^T = \mathbf{Z}^T\mathbf{Z} + \lambda n \mathbf{I}_s$ and $\mathbf{U} = \mathbf{L}^{-T}\mathbf{Z}^T$.

- Cost: $O\left(ns^2 + n^2 \cdot \sqrt{\kappa((\mathbf{Z}\mathbf{Z}^T + \lambda n \mathbf{I}_n)^{-1}(\mathbf{K} + \lambda n \mathbf{I}_n))}\right)$
  (s - number of random features, n - data-points, d - data dimension)
- **How big should s be?**

Theoretical Results

- $O(\lambda^{-1}\log(1/\lambda))$ random Fourier features suffice.
  Analysis is via matrix concentration inequalities.

- Statistical learning theory says that λ should grow with n^{-1}, but a slower rate, so there is a provable gain.

- The bound is tight even for one dimensional datasets.
  This can be shown using Fourier analysis.

- Using a modified random Fourier features, can replace λ^{-1} with

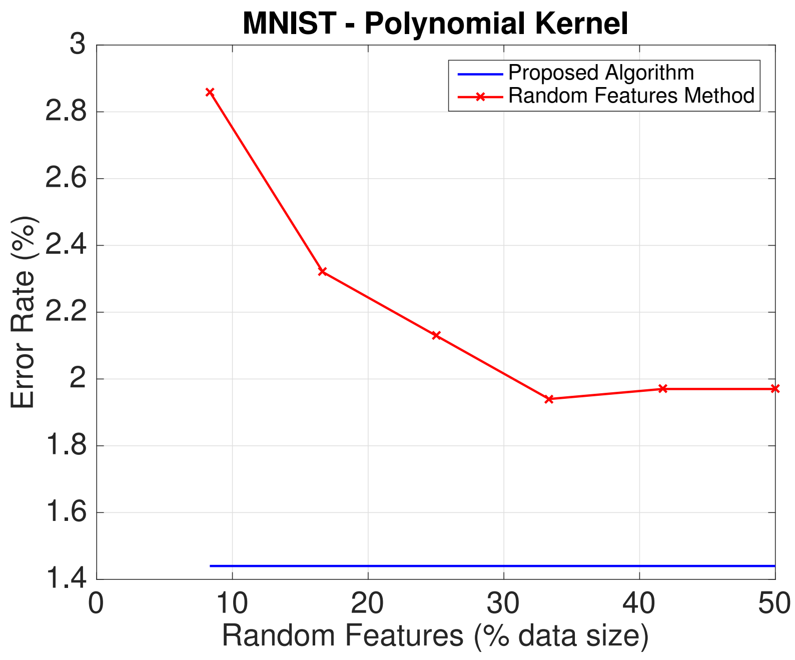
$$s_\lambda(\mathbf{K}) \equiv \mathbf{Tr}((\mathbf{K} + \lambda n \mathbf{I}_n)^{-1}\mathbf{K})$$

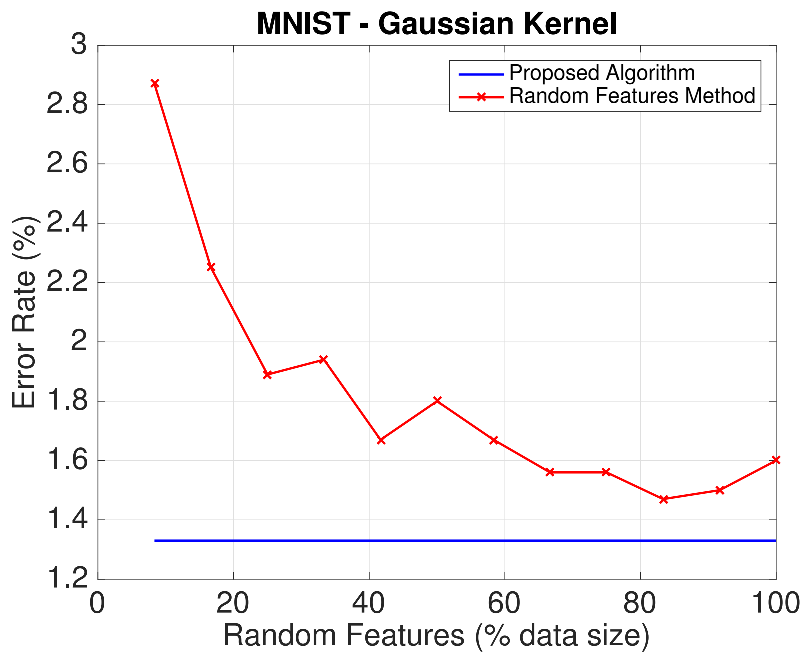
  ("statistical dimension"). Grows much slower with n than λ^{-1}.

Additional Results, Not Discussed Here (only a 20 minute talk!)

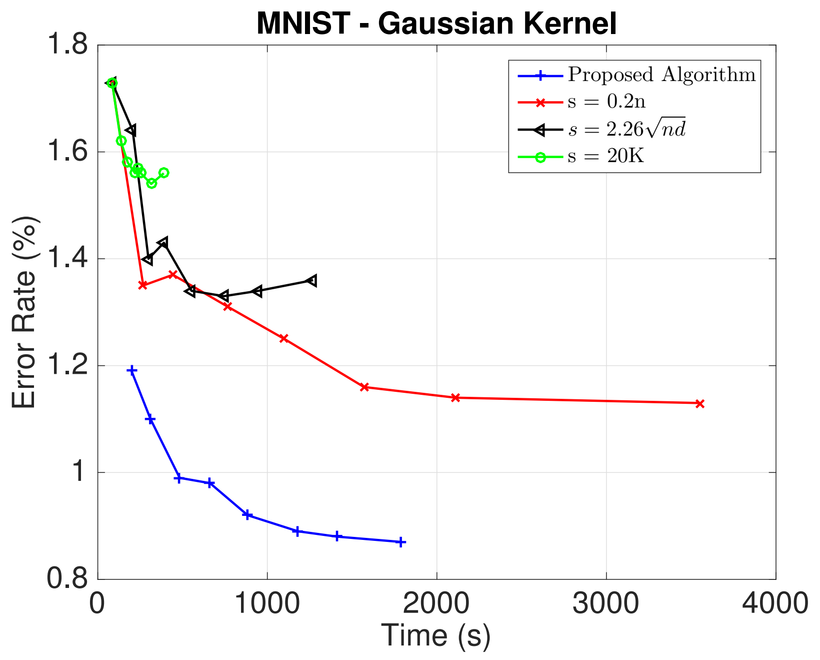
- Tight analysis for the polynomial kernel.
- Further dimensionality reduction using Randomized NLA.
- Testing preconditioners and adaptively setting their size.

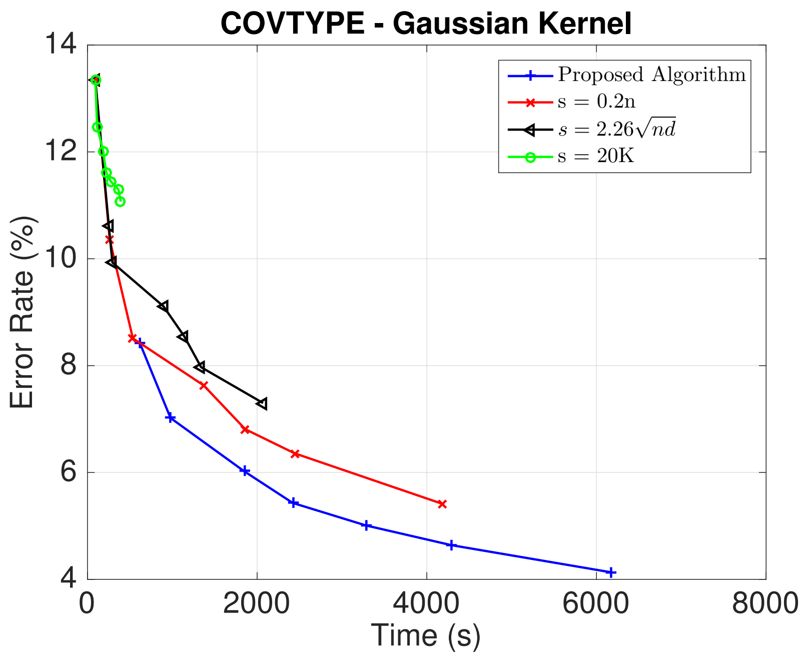
Experimental Results -
Comparison to Random Features (fixed size dataset)





Experimental Results - Comparison to Random Features (growing dataset)





Experimental Results - Training High Quality Models on Cloud Based Clusters

Dataset	n	d	Resources	s	Running Time (sec)	Error Rate
GISETTE	6,000	5,000	1 c4.large	500	8.1	3.50%
ADULT	32,561	123	1 c4.4xlarge	5,000	19.8	14.99%
IJCNN1	49,990	22	1 c4.8xlarge	10,000	55.5	1.39%
MNIST	60,000	780	1 c4.8xlarge	10,000	76.3	1.33%
MNIST-400K	400,000	780	8 r3.8xlarge	40,000	1060	0.89%
MNIST-1M	1,000,000	780	42 r3.8xlarge	40,000	1210	0.72%
EPSILON	400,000	2,000	8 r3.8xlarge	10,000	469	10.21%
COVTYPE	464,809	54	8 r3.8xlarge	40,000	2960	4.13%
YEARMSD	463,715	90	8 r3.8xlarge	15,000	312	4.58×10^{-3}

Experimental Results - Comparison to a Scalable ADMM Based Solver

Dataset	Resources	ADMM - s	ADMM - time	ADMM - error	Precond - time	Precond - error
MNIST	1 c4.8xlarge	15,000	102	1.95%	76.3	1.33%
MNIST-400K	8 r3.8xlarge	100,000	1017	1.10%	1060	0.89%
EPSILON	8 r3.8xlarge	100,000	1823	11.58%	469	10.21%
COVTYPE	8 r3.8xlarge	115,000	6640	5.73%	2960	4.13%
YEARMSD	8 r3.8xlarge	115,000	958	5.01×10^{-3}	312	4.58×10^{-3}

Summary and Limitations

- Summary:
  - Theoretically, usually running time is between $O(n^2)$ and $O(n^3)$.
  - Empirically, it often behaves like $O(n^2)$.
  - Simple and as such parallelizes well even on cloud platforms.

  Highly effective on datasets with as many as one million training examples.

- Limitations:
  - $O(n^2)$ memory usage - blows up memory usage quickly.
  - Many parameters, large prediction time.

Thank You!

Acknowledgments:

- Co-authors:
  - Random Features Preconditioning: Ken Clarkson, David Woodruff.
  - Other paper mentioned: Po-Sen Huang, Michael Kapralov, Cameron Musco, Christoper Musco, Huy Nguyen, Bhuvana Ramabhadran, Tara N. Sainath, Vikas Sindhwani, Ameya Velingker, Amir Zandieh.

- Funding: XDATA program of the Defense Advanced Research Projects Agency (DARPA), administered through Air Force Research Laboratory contract FA8750-12-C-0323.