

Computationally Efficient Nyström Approximation using Fast Transforms

Si Si

Department of Computer Science
University of Texas at Austin

ICML 2016

Joint work with Cho-Jui Hsieh and Inderjit S. Dhillon

Outline

- Background
- Motivation
- Fast Transforms for Nyström Approximation
 - Construct Structured Landmark Points
 - Using Fast Transforms to Speed up Kernel Value Evaluation
- Experimental Results on Fast Prediction

Background: Kernel Machines

- Popular kernel machines: training set $\{\mathbf{x}_i, y_i\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^d$,
 - Kernel SVM

$$\alpha^* \leftarrow \arg \min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - \mathbf{e}^T \alpha \quad \text{s.t. } 0 \leq \alpha \leq C,$$

- Kernel Ridge Regression

$$\alpha^* \leftarrow \arg \min_{\alpha} \alpha^T G \alpha + \lambda \alpha^T \alpha - 2 \alpha^T \mathbf{y},$$

$G \in \mathbb{R}^{n \times n}$ is the kernel matrix; $G_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$; $Q_{ij} = y_i y_j G_{ij}$.

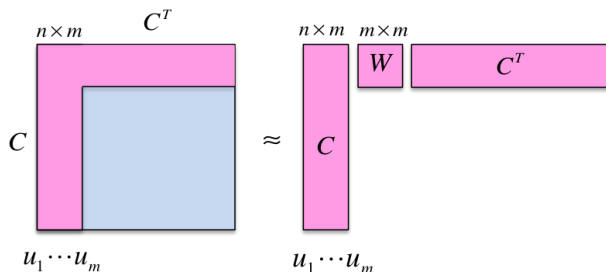
- Slow Training:
 - $O(n^2 d)$ time to form the kernel matrix.
 - $O(n^2)$ space to store the kernel matrix.
- Slow Prediction:
 - Kernel SVM: $y = \text{sign}(\sum_{i=1}^n \alpha_i K(\mathbf{x}, \mathbf{x}_i))$; $O(d(\#SV))$.
 - Kernel Ridge Regression: $y = \sum_{i=1}^n \alpha_i K(\mathbf{x}, \mathbf{x}_i)$; $O(nd)$.

Background: Nyström Kernel Approximation

- Nyström approximation (Williams and Seeger, 2001): based on m landmark points $\mathbf{u}_1, \dots, \mathbf{u}_m$:

$$G \approx \tilde{G} = CWC^T.$$

- Replace G with \tilde{G} in the objective and train the model:



- Speed up training and prediction of kernel machines.

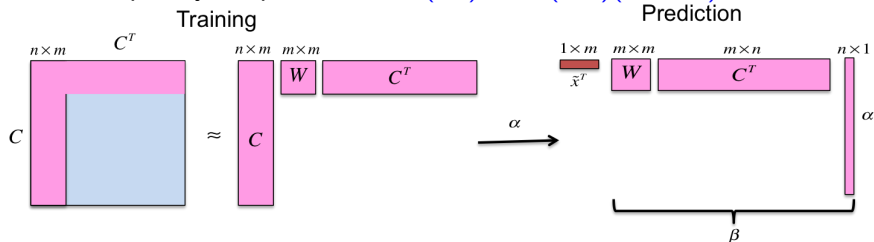
Background: Prediction Using Nyström Approximation

- Perform prediction on a new point \mathbf{x} given the model α :

$$\sum_{i=1}^n \alpha_i \tilde{K}(\mathbf{x}, \mathbf{x}_i) = \tilde{\mathbf{x}}^T W C^T \alpha = \tilde{\mathbf{x}}^T \beta$$

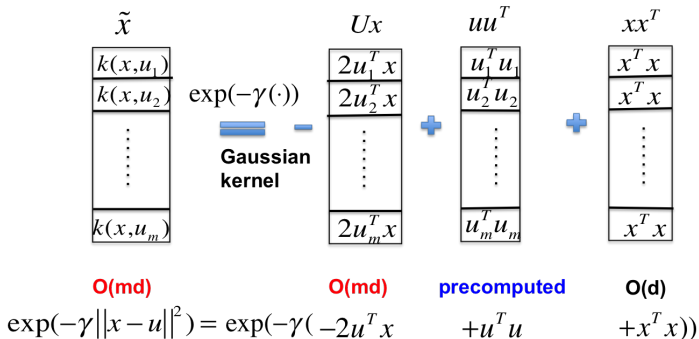
$$\text{where } \tilde{\mathbf{x}} = [K(\mathbf{x}, \mathbf{u}_1), \dots, K(\mathbf{x}, \mathbf{u}_m)]^T.$$

- Compute $\tilde{\mathbf{x}} = [k(\mathbf{x}, \mathbf{u}_1), \dots, k(\mathbf{x}, \mathbf{u}_m)]^T$: $O(md)$.
- Compute $\tilde{\mathbf{x}}^T \beta$: $O(d)$.
- Time complexity for prediction: $O(nd) \rightarrow O(md)(n \gg m)$.



Motivation

- Consider the kernel form: $K(\mathbf{x}_i, \mathbf{x}_j) = f(\mathbf{x}_i)f(\mathbf{x}_j)g(\mathbf{x}_i^T \mathbf{x}_j)$.
- Gaussian kernel: $f(\mathbf{x}) = e^{-\|\mathbf{x}\|^2}$, $g(z) = e^{2z}$.



- On mnist dataset, with $n = 60K$, $m = 160$
total time is 11.15 seconds and $T_{U\mathbf{x}}$ takes 10.81 seconds.

The dominant term is $T_{U\mathbf{x}}$ — usually takes $O(md)$.

Construct Landmark Points

- How to speed up Ux ?

Our solution: construct structured landmark points based on fast transforms.

- The form of landmark points U :

$$U = [H_d V_1, H_d V_2, \dots, H_d V_m].$$

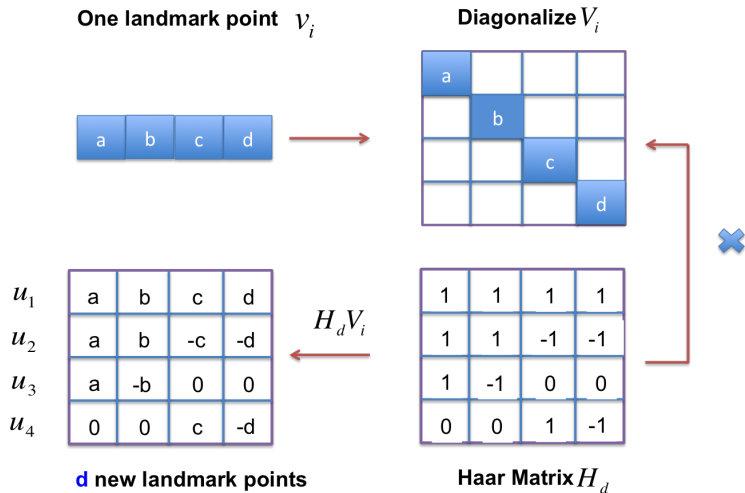
- Given initial landmark point v_i , V_i is a diagonal matrix:

$$V_i = \text{diag}(v_{i1}, \dots, v_{id}).$$

- H_d contains the sign pattern of the Haar or Hadamard matrix:

$$H_{\text{haar}} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix}, \quad H_{\text{hadamard}} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}.$$

Fast Transforms based Landmark Points



Construct Structured Landmark Points

- Initially m landmark points v_1, \dots, v_m .
- The form of new landmark points U (**md landmark points**):

$$U = [H_d V_1, H_d V_2, \dots, H_d V_m].$$

m initial landmark points



md new landmark points

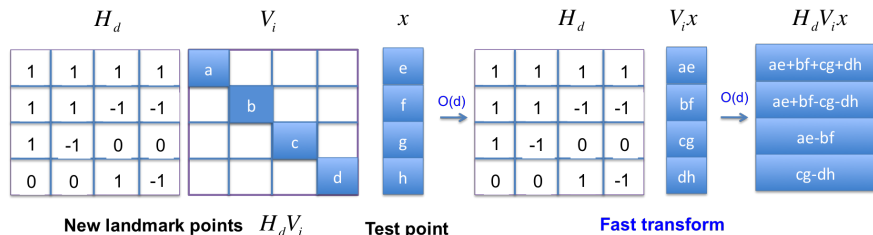
$$\{v_1, \dots, v_m\} \rightarrow \left\{ \left[H_d \begin{pmatrix} v_{11} & 0 & \dots & 0 \\ 0 & v_{12} & \dots & \vdots \\ \vdots & \dots & \ddots & 0 \\ 0 & \dots & 0 & v_{1d} \end{pmatrix} \right], \dots, \left[H_d \begin{pmatrix} v_{m1} & 0 & \dots & 0 \\ 0 & v_{m2} & \dots & \vdots \\ \vdots & \dots & \ddots & 0 \\ 0 & \dots & 0 & v_{md} \end{pmatrix} \right] \right\}$$

Benefit of U

- $U\mathbf{x}$ will become:

$$U\mathbf{x} = [(H_d V_1 \mathbf{x}), (H_d V_2 \mathbf{x}), \dots, (H_d V_m \mathbf{x})].$$

- Time for computing one block $H_d V_i \mathbf{x}$:
 - Haar landmark points: $O(d)$.
 - Hadamard landmark points: $O(d \log(d))$.



Prediction Cost

$$U\mathbf{x} = [(H_d V_1 \mathbf{x}), (H_d V_2 \mathbf{x}), \dots, (H_d V_m \mathbf{x})]$$

- Time complexity analysis (U has md landmark points):

$$\left\{ \left[\underbrace{H_d \begin{pmatrix} v_{11} & 0 & \cdots & 0 \\ 0 & v_{12} & \cdots & \vdots \\ \vdots & \cdots & \ddots & 0 \\ 0 & \cdots & 0 & v_{1d} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix}}_{O(d)} \right], \dots, \left[\underbrace{H_d \begin{pmatrix} v_{m1} & 0 & \cdots & 0 \\ 0 & v_{m2} & \cdots & \vdots \\ \vdots & \cdots & \ddots & 0 \\ 0 & \cdots & 0 & v_{md} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix}}_{O(d)} \right] \right\}_{O(md)}$$

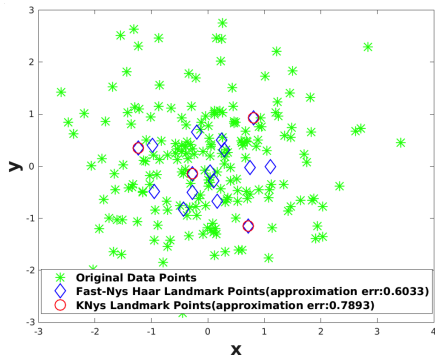
- Prediction time for md landmark points:
 - Haar landmark points: $O(md)$.
 - Hadamard landmark points: $O(md \log(d))$.

Time Complexity Analysis

Table : Time complexity analysis. d is the dimension of the instance.

	# of initial landmark points	# of new landmark points	Prediction time
Standard Nyström	m	0	$O(md)$
Haar landmark points	m	md	$O(md)$
Hadamard landmark points	m	md	$O(md \log d)$

- $d = 4$; $m = 4$; initial landmark points are kmeans centroids



Setting:

4 dimensional data points

4 initial landmark points

16 new landmark points

Results:

Fast-Nys App. Error: 0.6033

Kmeans-Nys App. Error: 0.7893

Learning the Structure

- How to learn U automatically?
 - Satisfy the structural constraints $U = [H_d V_1, H_d V_2, \dots, H_d V_m]$.
 - Minimize the upper bound of the kernel approximation error.

$$\arg \min_{U \in \mathcal{S}} \sum_{i=1}^n \left(\min_{t(i)} \|x_i - u_{t(i)}\|^2 \right)$$

Structural constraints **Kmeans objective**

$$U = [H_d V_1, \dots, H_d V_m]$$

- How to do optimization? **Kmeans-like way**
 - Update indicator $t(i)$: the landmark point $u_{t(i)}$ that x_i is closest to.
 - Update landmark points U : recompute the new centroids under structural constraints.
- Details are in the paper.

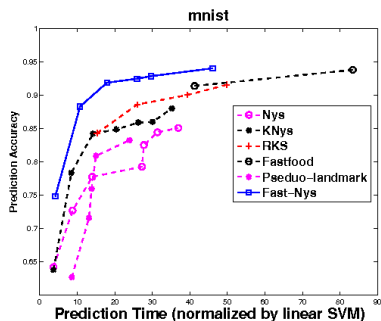
The Experimental Setting

- Methods compared in the experiments:
 - 1 The standard Nyström method(Nys)(Williams and Seeger, 2001);
 - 2 Kmeans Nyström (KNys)(Zhang et al. 2008);
 - 3 Random Kitchen Sinks(RKS)(Rahimi and Recht, 2007);
 - 4 Fastfood with “Hadamard features”(fastfood)(Le et al. 2013);
 - 5 Pseudo Landmark points (Pseudo) (Hsieh et al. 2014);
 - 6 The Local Deep Kernel Learning method (LDKL) (Jose et al. 2013);
 - 7 Divide-and-Conquer based fast Prediction (DC-Pred++) (Hsieh et al. 2014);
 - 8 Fast transform based Nyström Approximation (**Fast-Nys**) (Si et al. 2016).
- Data set statistics (n : number of samples):

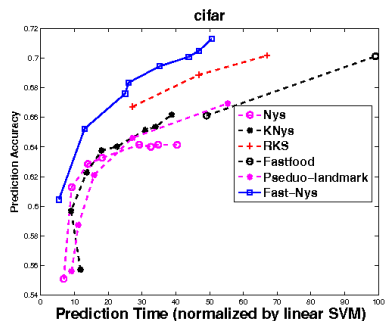
Dataset	n	d	Dataset	n	d
usps	9298	256	webspam	350,000	254
a9a	48,842	123	mnist	60,000	784
letter	18,000	16	cifar	60,000	400

Results on Fast Prediction

- Prediction Time vs. Prediction Accuracy.
- The unit of prediction time is linear SVM's prediction time.
- Both Haar and Hadamard landmark points yield similar results.



(a) mnist



(b) cifar

More Results on Fast Prediction

- Combine Fast-Nys with divide-and-conquer framework: **DC-Fast-Nys**.
- The unit in prediction time is linear SVM prediction time.
- DC-Pred++ and LDKL are two state-of-the-art fast prediction algorithms.

Dataset	Metric	DC-Fast-Nys	DC-Pred++	LDKL	KNys	RKS	Fastfood	Liblinear
letter	Prediction Time	7.6x	12.8x	29x	140x	61x	50x	1x
	Accuracy	95.4%	95.90%	95.78%	87.58%	89.93%	89.9%	73.08%
usps	Prediction Time	7.31x	14.4x	12.01x	200x	72.5x	80x	1x
	Accuracy	94.9%	95.56%	95.96%	92.53%	91.33%	94.39%	83.65%
webspam	Prediction Time	11.21x	20.5x	23x	200x	34.5x	80x	1x
	Accuracy	98.0%	98.4%	95.15%	95.01%	96.4%	96.7%	93.10%

Conclusions

- Observation: Computing Ux is the bottleneck.
- Fast-Nys: Fast Transforms for Nyström Approximation.
 - Construct Structured Landmark Points.
 - Using Fast Transforms to Speed up Kernel Value Evaluation.
- Experimental Results on Fast Prediction.

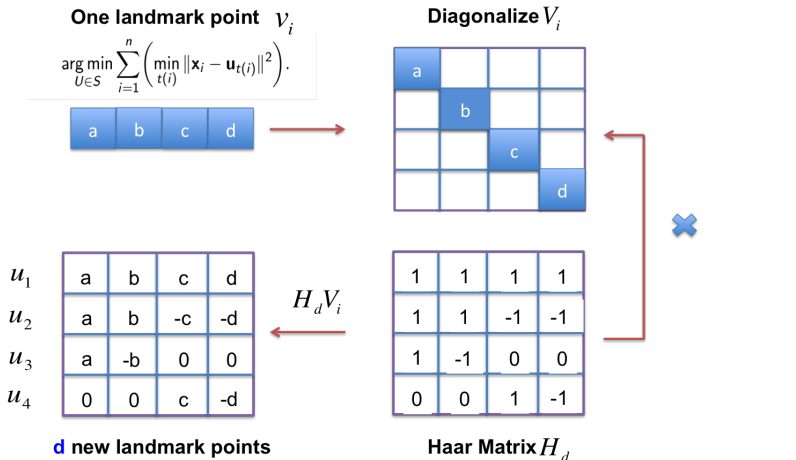
Our poster # 66 — Tuesday 10:00am to 1:00pm

References

- [1] Williams, Christopher and Seeger, Matthias *Using the Nyström Method to Speed Up Kernel Machines*. NIPS, 2001.
- [2] Hsieh, C.-J., Si, S., and Dhillon, I. S. *Fast Prediction for Large-Scale Kernel Machines*. NIPS, 2014.
- [3] Jose, C., Goyal, P., Aggrwal, P., and Varma, M. *Local deep kernel learning for efficient non-linear SVM prediction*. ICML, 2013.
- [4] Le, Q. V., Sarlos, T., and Smola, A. J. *Fastfood – Approximating Kernel Expansions in Loglinear Time*. ICML, 2013.
- [5] Rahimi, A. and Recht, Benjamin. *Random Features for Large-scale Kernel Machines*. NIPS, 2007.
- [6] Si, S., Hsieh, C.-J., and Dhillon, I. S. *Computationally Efficient Nyström Approximation using Fast Transforms*. ICML, 2016.
- [7] Zhang, K, Tsang, I. W., and Kwok, J. T. *Improved Nyström Low Rank Approximation and Error Analysis*. ICML, 2008.

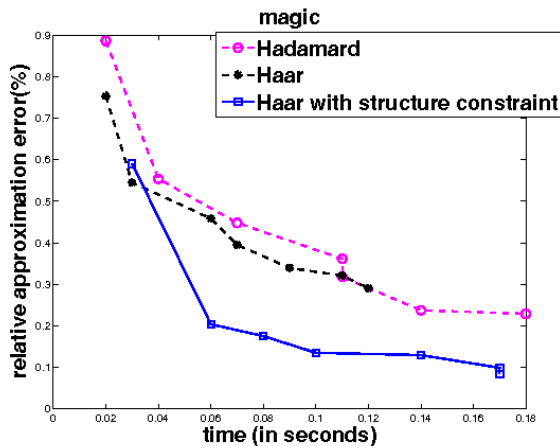
Algorithm

- Learn the initial m landmark points.
- Construct md new landmark points.
- Use md new landmark points during training and prediction.



Learning Structure of Landmark Points

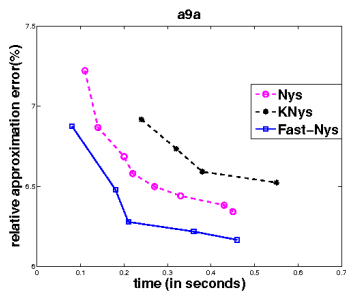
- Learning vs. not learning



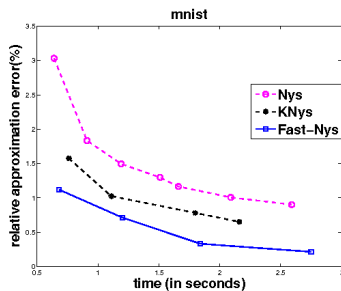
Results on Kernel Approximation

Relative kernel approximation error $\|G - \tilde{G}\|_F / \|G\|_F$.

- Time vs. kernel approximation error:



(c) a9a, polynomial.



(d) mnist, Gaussian.