Random Feature Maps for Dot Product Kernels

Purushottam Kar and Harish Karnick
{purushot,hk}@cse.iitk.ac.in

Indian Institute of Technology, Kanpur, UP, INDIA

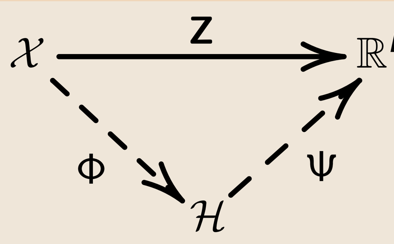
Introduction

- Proliferation of kernel learning techniques in diverse domains
- Kernel trick : working in high dimensional spaces via feature maps $\Phi : \mathcal{X} \to \mathcal{H}$
- High dimensionality necessitates working with implicit representations
  - SVM Classification : $h(\mathbf{x}) = \text{sgn}\left(\langle \mathbf{W}, \Phi(\mathbf{x})\rangle\right) = \text{sgn}\left(\sum_{\mathbf{x}' \in \mathcal{S}} \alpha_{\mathbf{x}'} K(\mathbf{x}, \mathbf{x}')\right)$
  - Kernel PCA : $P_k(\mathbf{x}) = \langle \mathbf{V}_k, \Phi(\mathbf{x})\rangle = \sum_{\mathbf{x}' \in \mathcal{S}^k} \alpha_{\mathbf{x}'} K(\mathbf{x}, \mathbf{x}')$
  - *Support vector effect* : slow test routines if support sets are large
- **Goal** : Circumvent the support vector effect
- **Our Contributions** :
  - develop random feature maps for the class of dot product kernels
  - provide theoretical guarantees of performance
  - empirically demonstrate speedups for kernel SVMs

Plan of attack

- Since high dimensionality of RKHS is the problem - reduce it !
  - Inner product preserving map from RKHS to small dimensions $\Psi : \mathcal{H} \to \mathbb{R}^D$
  - Reduces kernel problems to linear ones eg. linear SVM, linear PCA
  - Test times become independent of support set sizes
- Motivation : existence of such maps predicted by Johnson-Lindenstrauss lemma

Definition 1. (Approximate feature maps for kernels)



- A map $\mathbf{Z} : \mathcal{X} \to \mathbb{R}^D$ is an ϵ-approximate feature map for K if for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$, $|K(\mathbf{x}, \mathbf{y}) - \langle \mathbf{Z}(\mathbf{x}), \mathbf{Z}(\mathbf{y})\rangle| < \epsilon$

- Existing Work (among others)
  - [1] : maps for translation invariant kernels $K(\mathbf{x}, \mathbf{y}) = f(\mathbf{x} - \mathbf{y})$
  - [2] : maps for homogeneous kernels $K(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} (\mathbf{x}_i \mathbf{y}_i)^\alpha f(\log \mathbf{x}_i - \log \mathbf{y}_i)$
- **This paper** : maps for dot product kernels $K(\mathbf{x}, \mathbf{y}) = f(\langle \mathbf{x}, \mathbf{y}\rangle)$

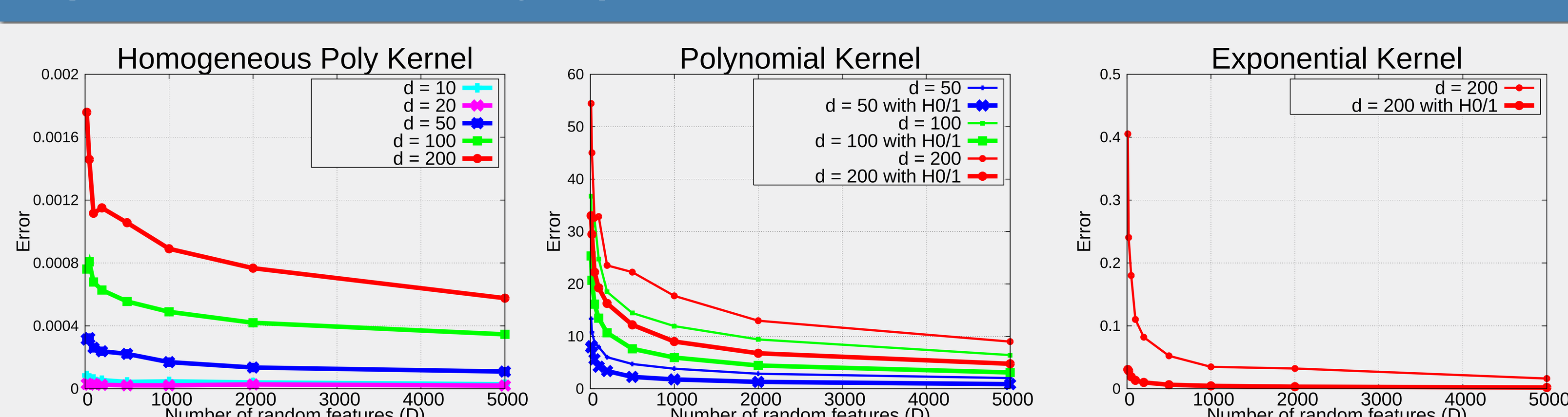
Dot product kernels

Theorem 2. (Characterization of dot product kernels)

A function $f : \mathbb{R} \to \mathbb{R}$ constitutes a positive definite kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, $K : (\mathbf{x}, \mathbf{y}) \mapsto f(\langle \mathbf{x}, \mathbf{y}\rangle)$ for all $d > 0$ iff f is an analytic function having a Maclaurin expansion with only non-negative coefficients i.e. $f(x) = \sum_{n=0}^{\infty} a_n x^n$, $a_n \geq 0$.

- Proof proceeds in two steps
  - Show that p.d.-ness over all $\mathbb{R}^d, d > 0 \Leftrightarrow$ p.d.-ness over Hilbert spaces
  - Characterize kernels that are p.d. over Hilbert spaces (based on [3])
- Examples
  - Polynomial Kernels : homogeneous $(\langle \mathbf{x}, \mathbf{y}\rangle)^p$, non-homogeneous $(1 + \lambda \langle \mathbf{x}, \mathbf{y}\rangle)^p$
  - Exponential Kernels : $\exp\left(\frac{\langle \mathbf{x}, \mathbf{y}\rangle}{\sigma^2}\right)$
  - Vovk's Kernels : polynomial $\frac{1 - \langle \mathbf{x}, \mathbf{y}\rangle^p}{1 - \langle \mathbf{x}, \mathbf{y}\rangle}$, infinite polynomial $\frac{1}{1 - \langle \mathbf{x}, \mathbf{y}\rangle}$

Experimental results : Toy experiments



- Sample 100 points from $\mathbb{R}^d$ and create kernel matrices
- Use random feature maps with/out **H0/1** and reconstruct the kernel matrices
- **H0/1** offers sharper drop in average reconstruction error

Random Feature Maps

Algorithm 3. (Feature map construction algorithm)

- **Given** : Kernel $K(\mathbf{x}, \mathbf{y}) = f(\langle \mathbf{x}, \mathbf{y}\rangle)$ over $\mathcal{X} \subseteq \mathbb{R}^d$, target dimensionality D
- Obtain Maclaurin expansion of $f(x) = \sum_{n=0}^{\infty} a_n x^n$ by setting $a_n = \frac{f^{(n)}(0)}{n!}$
- Choose a small constant $p > 1$
- Create a unidimensional feature map $Z : \mathcal{X} \to \mathbb{R}$
  1. Sample $N \in \mathbb{N} \cup \{0\}$ with probability $\mathbb{P}[N = n] = \frac{1}{p^{n+1}}$
  2. Sample N independent Rademacher vectors $\omega_1, \ldots, \omega_N \in \{-1, +1\}^d$
  3. Create a feature map $Z : \mathbf{x} \mapsto \sqrt{a_N p^{N+1}} \prod_{j=1}^{N} \omega_j^T \mathbf{x}$
- Create D independent unidimensional feature maps $Z_1, \ldots, Z_D$
- **Output** : $\mathbf{Z} : \mathbf{x} \mapsto \frac{1}{\sqrt{D}}(Z_1(\mathbf{x}), \ldots, Z_D(\mathbf{x})) \in \mathbb{R}^D$

Theoretical analysis

Theorem 4. (Approximation guarantee)

Suppose $\mathcal{X} \subseteq \mathcal{B}_1(\mathbf{0}, R)$ is a compact subset of $\mathbb{R}^d$, and $K(\mathbf{x}, \mathbf{y}) = f(\langle \mathbf{x}, \mathbf{y}\rangle)$. Let $C = f(pR^2)$ and $L = f'(pR^2) \cdot d$. Then if $D = \Omega\left(\frac{dC^2}{\epsilon^2} \log\left(\frac{RL}{\epsilon\delta}\right)\right)$, then the feature map $\mathbf{Z} : \mathcal{X} \to \mathbb{R}^D$ constructed above is an ϵ-approximate feature map for K with probability $1 - \delta$.

- Proof exploits Lipschitz properties of the kernel and the feature map
- $D = \tilde{\mathcal{O}}\left(d/\epsilon^2\right)$: near optimal dependence on ϵ, quasi-linear dependence on d
- Dot product kernels are unbounded : stronger kernel-specific dependence
  - C is the largest value taken by K in the region $p\mathcal{X}$
  - L encodes the rate of growth of K in the region $p\mathcal{X}$

Extension to compositional kernels

- Kernels of the form $f(K_{\text{inner}}(\mathbf{x}, \mathbf{y}))$ for arbitrary p.d kernel K_{inner}
  - Dot product kernels are a special cases with $K_{\text{inner}}(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y}\rangle$
- Assume access to a (randomized) feature map $W : \mathcal{X} \to \mathbb{R}$ for K_{inner}
  - W should give an unbiased estimate for K_{inner} over $\mathcal{X}$
  - W should be bounded and Lipschitz on expectation
- Feature map construction algorithm : identical to Algorithm 3 except
  - In step 2, request N independent copies of W : $W_1, \ldots, W_N$
  - In step 3, create a feature map $Z : \mathbf{x} \mapsto \sqrt{a_N p^{N+1}} \prod_{j=1}^{N} W_j(\mathbf{x})$
- Approximation guarantee : similar to that in Theorem 4

Practical considerations

- Randomness reduction : truncate the Maclaurin expansion
  - Truncation error ϵ_1 uniform by properties of Maclaurin series
  - Gives us $(\epsilon + \epsilon_1)$-approximate feature maps
- **H0/1**: heuristic for more accurate feature maps
  - Maclaurin expansion : first term is constant, second is linear
  - No need to estimate these - append the original features to **Z**
  - *Advantages* : variance reduction, more accuracy
  - *Disadvantages* : feature dimensionality goes up, mapping time goes up
  - Offers best results with small to medium values of D

References

[1] Ali Rahimi and Benjamin Recht. Random Features for Large-Scale Kernel Machines. In *21st Annual Conference on Neural Information Processing Systems*, 2007.

[2] Andrea Vedaldi and Andrew Zisserman. Efficient Additive Kernels via Explicit Feature Maps. In *23rd IEEE Conference on Computer Vision and Pattern Recognition*, pages 3539–3546, 2010.

[3] Isaac Jacob Schoenberg. Positive Definite Functions on Spheres. *Duke Mathematical Journal*, 9(1):96–108, 1942.

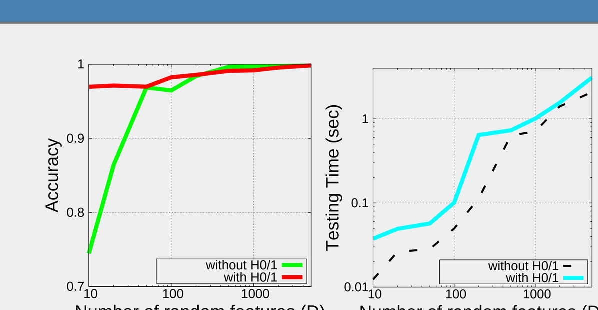
Experimental results : UCI datasets

Dataset	**K** + LIBSVM	**RF** + LIBLINEAR	**H0/1** + LIBLINEAR
Nursery N = 13000 $d = 8$	acc = 99.9% trn = 18.6s tst = 3.37s	acc = 99.7% trn = 3.96s (4.7×) tst = 0.63s (5.3×) D = 500	acc = 98.2% trn = 0.49s (38×) tst = 0.1s (33×) D = 100
Cod-RNA N = 60000 $d = 8$	acc = 95.2% trn = 144.1s tst = 28.6s	acc = 94.9% trn = 12.1s (12×) tst = 2.8s (10×) D = 500	acc = 93.77% trn= 0.63s (229×) tst = 0.51s (56×) D = 50
Adult N = 49000 $d = 123$	acc = 84.2% trn = 179.6s tst = 60.6s	acc = 84.7% trn = 21.2s (8.5×) tst = 15.6s (3.9×) D = 500	acc = 84.7% trn = 6.9s (26×) tst = 7.26s (8.4×) D = 100
IJCNN N=141000 $d = 22$	acc = 98.4% trn = 164.1s tst = 33.4s	acc = 97.3% trn = 36.5s (4.5×) tst = 23.3s (1.4×) D = 1000	acc = 92.3% trn= 4.98s (33×) tst = 7.5s (4.5×) D = 200
Covertype N=581000 $d = 54$	acc = 77.4% trn = 160.95s tst = 1653.9s	acc = 77.04% trn = 186.1s (—) tst = 236.8s (7×) D = 1000	acc = 75.5% trn = 3.9s (41×) tst = 70.3s (23×) D = 100

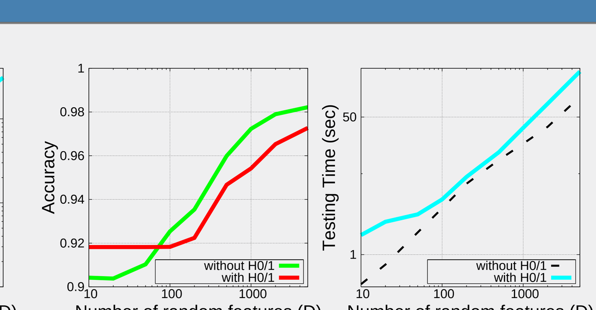
(a) Polynomial Kernel, $K(\mathbf{x}, \mathbf{y}) = (1 + \langle \mathbf{x}, \mathbf{y}\rangle)^{10}$

Dataset	**K** + LIBSVM	**RF** + LIBLINEAR	**H0/1** + LIBLINEAR
Nursery N = 13000 $d = 8$	acc = 99.8% trn = 10.8s tst = 1.7s	acc = 99.6% trn = 2.52s (4.3×) tst = 0.6s (2.8×) D = 500	acc = 97.96% trn = 0.4s (27×) tst = 0.18s (9.4×) D = 100
Cod-RNA N = 60000 $d = 8$	acc = 95.2% trn = 91.5s tst = 17.1s	acc = 94.9% trn = 11.5s (8×) tst = 2.8s (6.1×) D = 500	acc = 93.8% trn= 0.67s (136×) tst = 1.4s (12×) D = 50
Adult N = 49000 $d = 123$	acc = 83.7% trn = 263.3s tst = 33.4s	acc = 82.9% trn = 39.8s (6.6×) tst = 14.3s (2.3×) D = 500	acc = 84.8% trn = 7.18s (37×) tst = 9.4s (3.6×) D = 100
IJCNN N=141000 $d = 22$	acc = 98.4% trn = 135.8s tst = 29.98s	acc = 97.2% trn = 24.9s (5.5×) tst = 23.4s (1.3×) D = 1000	acc = 92.2% trn = 5.2s (26×) tst = 9.1s (3.3×) D = 200
Covertype N=581000 $d = 54$	acc = 80.6% trn = 194.1s tst = 695.8s	acc = 76.2% trn = 21.4s (9×) tst = 207s (3.6×) D = 1000	acc = 75.5% trn = 3.7s (52×) tst = 80.4s (8.7×) D = 100

(b) Exponential Kernel, $K(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{\langle \mathbf{x}, \mathbf{y}\rangle}{\sigma^2}\right)$



(c) Nursery with K_{poly}



(d) IJCNN with K_{exp}

- Random Features :
  - Useful in speeding up training and test routines for SVM
  - Experiments on other kernel learning tasks ?
- Using **H0/1** :
  - Competitive accuracies even with small values of D
  - Increased mapping time with larger values of D
  - Eventually, overheads prevent **H0/1** from being useful

15th International Conference on Artificial Intelligence and Statistics (**AISTATS 2012**) **Full Paper** : http://home.iitk.ac.in/~purushot/dpfeat.pdf