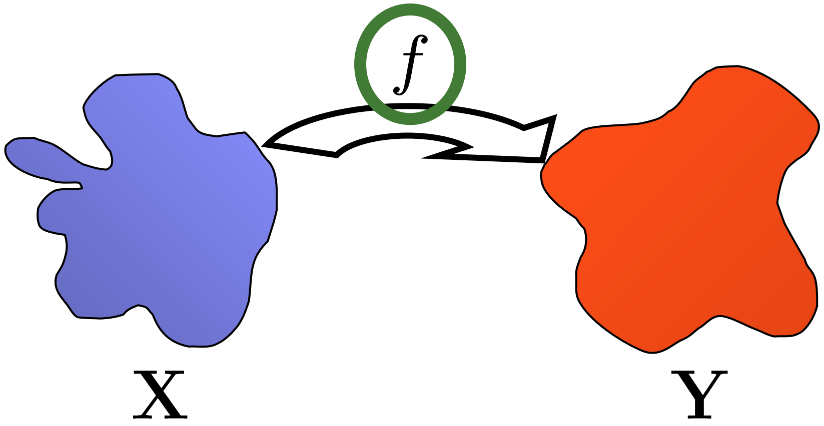
Uniform Approximation of Functions with Random Bases

Ali Rahimi
Intel Research

Ben Recht
UW Madison

Typically a list of example inputs



classes
covariate
state

$$\min_{f\in\mathcal{F}} \ \mathbf{fitness}(f, \text{data})$$

Which space of functions?

dictated by application

- **Goal:** Find a class $\mathcal{F}$ which is easy to search over, but can approximate complex behavior.

Approximation Schemes

- Approximate $f(\mathbf{x})$ by $f_n(\mathbf{x}) = \sum_{k=1}^{n} c_k \phi_k(\mathbf{x}; \theta_k)$
- Jones (1992), $\tilde{f} \in L_1(\mathbb{R}^d)$ $\phi(\mathbf{x};\, \mathbf{w}, b) = \cos(\mathbf{w}^* \mathbf{x} + b)$
- Barron (1993), $\nabla \tilde{f} \in L_1(\mathbb{R}^d)$ $\phi(\mathbf{x};\, \mathbf{w}, b) = \sigma(\mathbf{w}^* \mathbf{x} + b)$
- Girosi & Anzellotti (1995), $f \in W_{2,s}(\mathbb{R}^d)$ with $2s > d$ $\phi(\mathbf{x}; \mathbf{z}) = \exp(-\|x - z\|^2)$
- Using nearly identical analysis, all of these schemes achieve

$$\|f - f_n\|_2 = O\left(\frac{1}{\sqrt{n}}\right)$$

Approximation Schemes

- Approximate $f(\mathbf{x})$ by $f_n(\mathbf{x}) = \sum_{k=1}^{n} c_k \phi_k(\mathbf{x}; \theta_k)$

Simultaneously optimize

- Parameter tuning is tricky…
- (Can achieve $\|f - f_n\|_2 = O\left(\frac{1}{\sqrt{n}}\right)$ via a greedy "algorithm").

Randomize, don’t optimize

- Approximate $f(\mathbf{x})$ by $$f_n(\mathbf{x}) = \sum_{k=1}^{n} \underset{\text{optimize}}{c_k}\,\phi_k(\mathbf{x}; \underset{\text{sample}}{\theta_k})$$

- For which functions can we achieve $\|f - f_n\| = O\left(\frac{1}{\sqrt{n}}\right)$?

- How are these functions related to objects we already know and love?

- Practical Implementations

Function Class

- Fix parameterized basis functions $\phi(\mathbf{x};\theta)$
- Fix a probability distribution $p(\theta)$
- Our target space will be:

$$\mathcal{F}_p \equiv \left\{ f = \int \alpha(\theta)\phi(\cdot;\theta)d\theta \;\middle|\; \sup_\theta \left|\frac{\alpha(\theta)}{p(\theta)}\right| < \infty \right\}$$

- With the convention that

$$\left|\frac{\alpha(\theta)}{0}\right| = \begin{cases} 0 & \alpha(\theta) = 0 \\ \infty & \text{otherwise} \end{cases}$$

Random Features: Example

- Fourier basis functions: $\phi(\mathbf{x};\omega,b) = \cos(\omega^*\mathbf{x} + b)$
- Gaussian parameters $\omega \sim \mathcal{N}(0,\sigma^2 I)$ $b \sim \mathrm{unif}([0,2\pi])$
- If $\tilde{f}(\omega) = \int_{\mathbb{R}^d} f(\mathbf{x}) e^{-i\omega^*\mathbf{x}} dx$, then $\sup_{\omega} \left| \frac{\tilde{f}(\omega)}{p(\omega)} \right| \leq \infty$ means

that the frequency distribution of *f* has subgaussian tails.

$$\mathcal{F}_p \equiv \left\{ f = \int \alpha(\theta)\phi(\cdot;\theta)d\theta \;\middle|\; \sup_\theta \left| \frac{\alpha(\theta)}{p(\theta)} \right| \leq \infty \right\}$$

- **Thm:** Let f be in $\mathcal{F}_p$ with $\|f\|_p = \sup_\theta \left|\frac{\alpha(\theta)}{p(\theta)}\right|$. Let θ_1,…, θ_n be sampled iid from *p*. Then with probability at least 1 - δ:

$$\min_{c_k} \left\| f - \sum_{k=1}^{n} c_k \varphi(\mathbf{x};\theta_k) \right\|_2 \leq \frac{\|f\|_p}{\sqrt{n}} \left(\sqrt{2} + \tfrac{\sqrt{2}}{2} \log(\tfrac{1}{\delta}) \right)$$

- If additionally, $\phi(x;\theta)=\phi(\theta' x)$, with $\phi:\mathbb{R}\rightarrow\mathbb{R}$ L-Lipschitz, $\phi(0)=0$, and $|\phi|<1$ and *p* has a finite second moment, then with probability at least 1- δ

$$\min_{c_k} \left\| f - \sum_{k=1}^{n} c_k \varphi(\mathbf{x};\theta_k) \right\|_\infty \leq \frac{\|f\|_p}{\sqrt{n}} \left(\sqrt{\log \tfrac{1}{\delta}} + 4LB\sqrt{\mathbb{E}\theta'\theta} \right)$$

where $B = \sup_{x\in X} \|x\|_2$

Reproducing Kernel Hilbert Spaces

- A symmetric function **k**:X×X $\to \mathbb{R}$ is a *positive definite kernel* if for all N

$$\sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

- Reproducing Kernel Hilbert Space: $\{f(\mathbf{x}) = \sum_{i=1}^{n} c_i \mathbf{k}(\mathbf{x}_i, \mathbf{x})\}$

$$\left\langle \sum_{j=1}^{n} c_j k(\mathbf{x}_j, \cdot), \sum_{k=1}^{m} d_k k(\mathbf{z}_k, \cdot) \right\rangle_k := \sum_{j=1}^{n} \sum_{k=1}^{m} c_j d_k \mathbf{k}(\mathbf{x}_j, \mathbf{z}_k)$$

- Extensive Applications: Support Vector Machines, Kernel Machines, etc.

$$k(\mathbf{x}, \mathbf{y}) = \int p(\theta)\phi(\mathbf{x};\theta)\phi(\mathbf{y};\theta)d\theta$$

- RKHS generated by k: $\mathcal{H} = \{f(\mathbf{x}) = \sum_{i=1}^{n} c_i \mathbf{k}(\mathbf{x}_i, \mathbf{x})\}$

$$\mathcal{F}_p \equiv \left\{ f = \int \alpha(\theta)\phi(\cdot;\theta)d\theta \;\middle|\; \sup_\theta \left| \frac{\alpha(\theta)}{p(\theta)} \right| \leq \infty \right\}$$

- $\mathcal{F}_p$ is *dense* in $\mathcal{H}$, and for any *f* $\in \mathcal{F}_p$

$$\|f\|_k \leq \sup_\theta \left| \frac{\alpha(\theta)}{p(\theta)} \right|$$

Gaussian RKHS vs Random Features

- **Representer Theorem**: for many applications, the optimal function in an RKHS is of the form

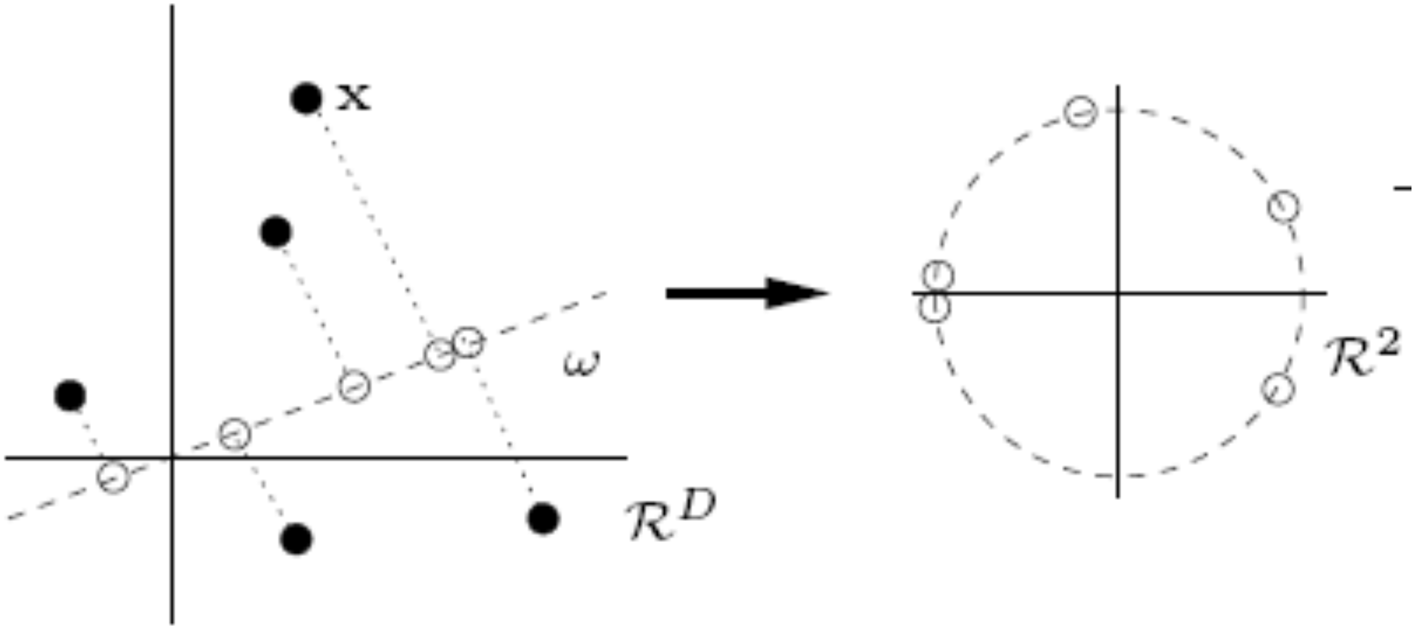
$$\sum_{i=1}^{n} c_i \mathbf{k}(\mathbf{x}_i, \mathbf{x})$$

given data set

- **RKHS form is preferred:** when number of data points is small or the function is not smooth

- **Random Features are preferred:** when number of data points is very large or the Representer theorem doesn’t apply

Fourier Random Features



$\omega \sim \mathcal{N}(0, 1)$

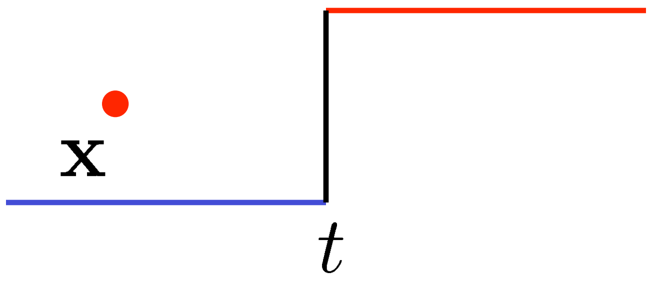
$b \sim \text{unif}[-\pi, \pi]$

$$\phi(\mathbf{x};\, \omega, b) = \cos(\omega' \mathbf{x} + b)$$

$$k(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2)$$

RKHS is dense in continuous functions

Random Decision Stumps



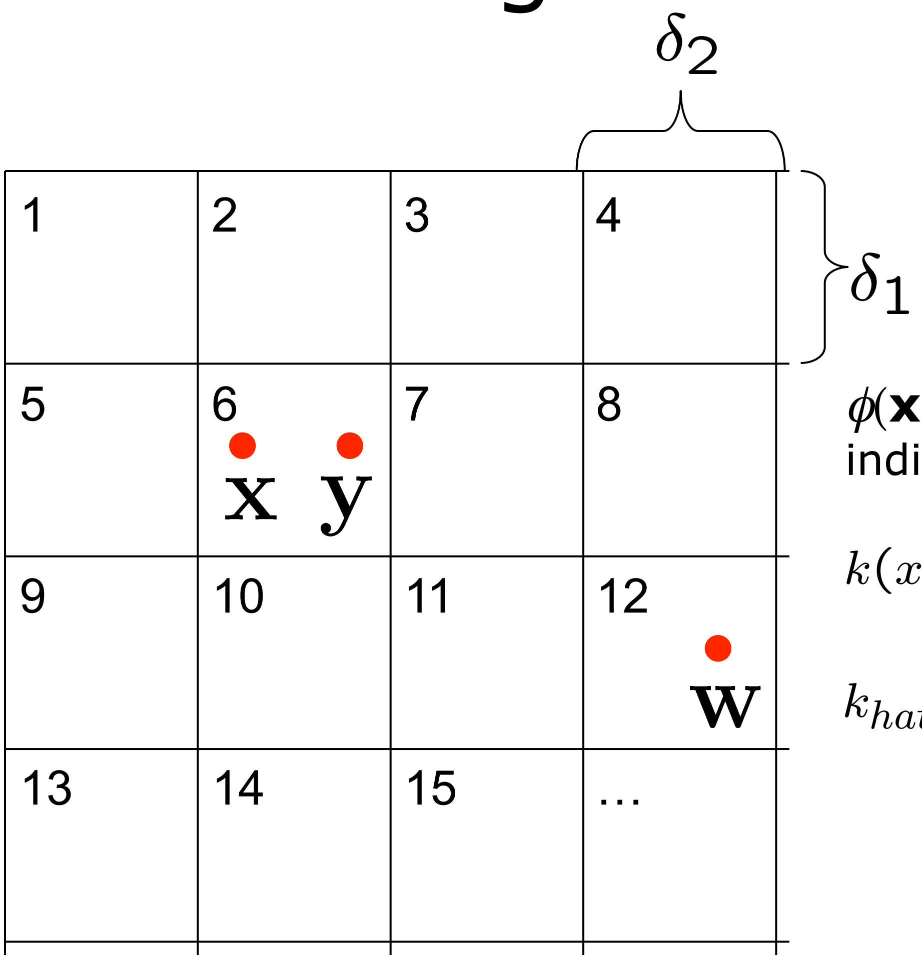
$$t \sim \mathrm{unif}([a,b])$$
$$i \sim \mathrm{unif}(\{1,\ldots,d\})$$

$$\phi(\mathbf{x}; t, i) = \mathrm{sign}(x_i - t)$$

Boosting Features

$$k(\mathbf{x},\mathbf{y}) = 1 - 2\frac{\|\mathbf{x}-\mathbf{y}\|_1}{b-a}$$

Binning Random Features

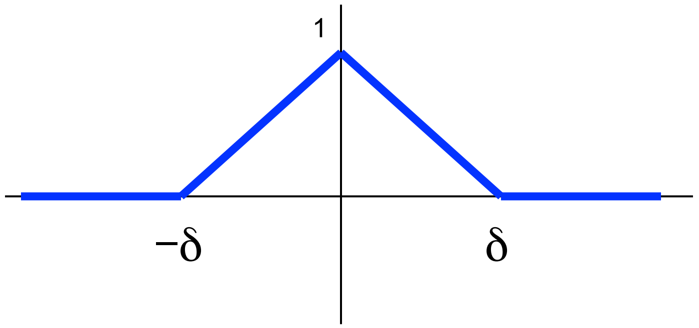


Lay a random grid so that for any **x** and **y**
*Pr[**x** and **y** are binned together] = k(**x**,**y**)*

$\phi(\mathbf{x})$ is the bin ID, encoded as a binary indicator vector.

$$k(x-y) = \prod_d \int_0^\infty k_{hat}(x-y;\delta)p(\delta)\, d\delta$$

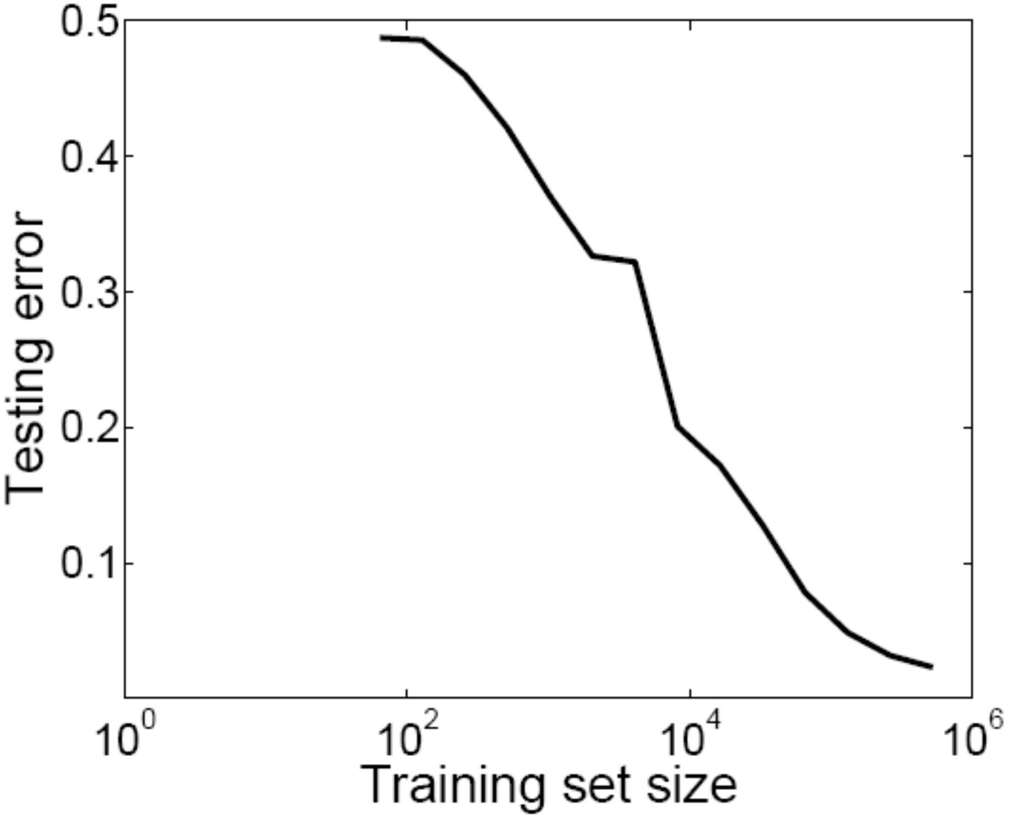
$$k_{hat}(x-y;\delta) = \max\left(0, 1-\delta^{-1}|x-y|\right)$$

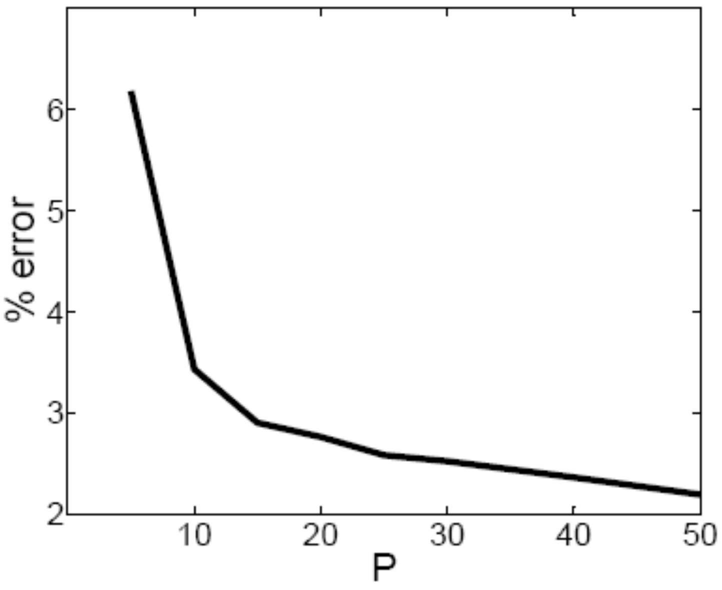


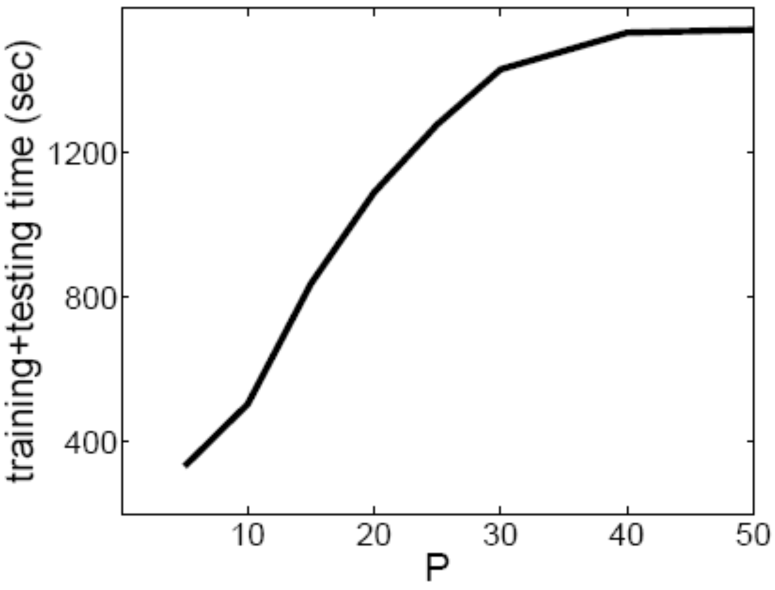
```
% Approximates Gaussian Process regression
%   with Gaussian kernel of variance gamma
% lambda: regularization parameter
% dataset: X is dxN, y is 1xN
% test: xtest is dx1
% D: dimensionality of random feature

% training
  w = randn(D, size(X,1));
  b = 2*pi*rand(D,1);
  Z = cos(sqrt(gamma)*w*X + repmat(b,1,size(X,2)));
  alpha = (lambda*eye(size(X,2)+Z*Z')\(Z*y);
% testing
  ztest = alpha(:)’*cos( sqrt(gamma)*w*xtest(:) + …
                   + repmat(b,1,size(X,2)) );
```

Dataset	**Fourier+LS**	**Binning+LS**	**Exact SVM**
`CPU` regression 6500 instances 21 dims	3.6% 20 secs $D = 300$	5.3% 3 mins $P = 350$	11% 31 secs ASVM
`Census` regression 18,000 instances 119 dims	5% 36 secs $D = 500$	7.5% 19 mins $P = 30$	9% 13 mins SVMTorch
`Adult` classification 32,000 instances 123 dims	14.9% 9 secs $D = 500$	15.3% 1.5 mins $P = 30$	15.1% 7 mins $\text{SVM}^{\text{light}}$
`Forest Cover` classification 522,000 instances 54 dims	11.6% 71 mins $D = 5000$	2.2% 25 mins $P = 50$	2.2% 44 hrs libSVM


0.5
0.4
0.3
0.2
0.1
Testing error
10^0
10^2
10^4
10^6
Training set size


6
5
4
3
2
% error
10
20
30
40
50
P


1200
800
400
training+testing time (sec)
10
20
30
40
50
P

$$f(\mathbf{x}) = \sum_{k=1}^{n} \underset{\substack{\uparrow \\ \textbf{Optimize}}}{c_k}\,\sigma(\underset{\nwarrow}{\mathbf{w}_k^*}\mathbf{x} + \underset{\uparrow}{b_k})$$

Randomize