A Review of Random Features for
Large-Scale Kernel Machines
Wenying Deng

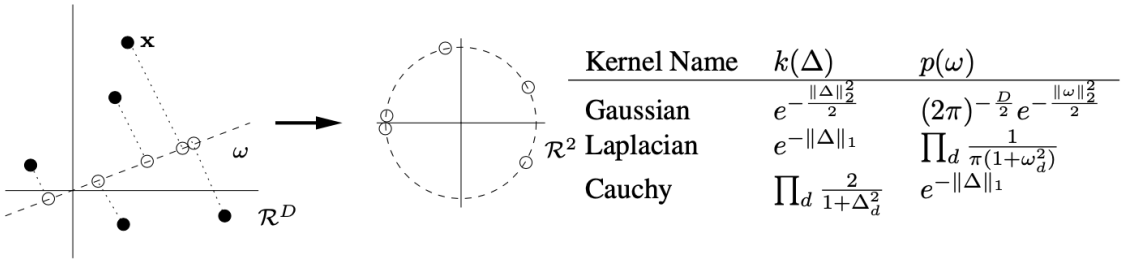
We know that methods that operate on the kernel matrix of the data scale poorly with the size of the training dataset. On the other hand, specialized algorithms for linear Support Vector Machines and regularized regression run much more quickly when the dimensionality of the data is small. Therefore, they propose a way to combine the advantages of the linear and nonlinear approaches.

The idea is mapping the data to a low-dimensional Euclidean inner product space using a randomized feature map $\mathbf{z} : \mathcal{R}^d \to \mathcal{R}^D$ so that the inner product between a pair of transformed points approximates their kernel evaluation:

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle \approx \mathbf{z}(\mathbf{x})'\mathbf{z}(\mathbf{y}). \tag{1}$$

And then apply fast linear learning methods to approximate the answer of the corresponding nonlinear kernel machine.

1 Random Fourier Features



Kernel Name	$k(\Delta)$	$p(\omega)$
Gaussian	$e^{-\frac{\|\Delta\|_2^2}{2}}$	$(2\pi)^{-\frac{D}{2}} e^{-\frac{\|\omega\|_2^2}{2}}$
Laplacian	$e^{-\|\Delta\|_1}$	$\prod_d \frac{1}{\pi(1+\omega_d^2)}$
Cauchy	$\prod_d \frac{2}{1+\Delta_d^2}$	$e^{-\|\Delta\|_1}$

Figure 1: Random Fourier Features. Each component of the feature map $\mathbf{z}(\mathbf{x})$ projects $\mathbf{x}$ onto a random direction ω drawn from the Fourier transform $p(\omega)$ of $k(\Delta)$, and wraps this line onto the unit circle in $\mathcal{R}^2$. After transforming two points $\mathbf{x}$ and $\mathbf{y}$ in this way, their inner product is an unbiased estimator of $k(\mathbf{x}, \mathbf{y})$. The mapping $z(\mathbf{x}) = \cos(\omega'\mathbf{x} + b)$ additionally rotates this circle by a random amount b and projects the points onto the interval $[0, 1]$. The table lists some popular shift-invariant kernels and their Fourier transforms. To deal with non-isotropic kernels, we can first whiten the data and apply one of these kernels

Theorem 1. *(Bochner [1]). A continuous kernel* $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$ *on* $\mathcal{R}^D$ *is positive definite if and only if* $k(\delta)$ *is the Fourier transform of a non-negative measure.*

If a shift-invariant kernel $k(\delta)$ is properly scaled, Bochner's theorem guarantees that its Fourier transform $p(\omega)$ is a proper probability distribution. Defining $\zeta_\omega(\mathbf{x}) = e^{j\omega'\mathbf{x}}$, we have

$$k(\mathbf{x} - \mathbf{y}) = \int_{\mathcal{R}^d} p(\omega) e^{j\omega'(\mathbf{x}-\mathbf{y})} d\omega = E_\omega[\zeta_\omega(\mathbf{x})\zeta_\omega(\mathbf{y})^*], \tag{2}$$

so $\zeta_\omega(\mathbf{x})\zeta_\omega(\mathbf{y})^*$ is an unbiased estimate of $k(\mathbf{x}, \mathbf{y})$ when ω is drawn from p.

i. Since both the probability distribution $p(\omega)$ and the kernel $k(\Delta)$ are real, so we may obtain a real-valued mapping that satisfies the condition $E_\omega[z_\omega(\mathbf{x})z_\omega(\mathbf{y})] = k(\mathbf{x}, \mathbf{y})$ by setting $z_\omega(\mathbf{x}) = \sqrt{2}\cos(\omega'\mathbf{x} + b)$, where ω is drawn from $p(\omega)$ and b is drawn uniformly from $[0, 2\pi]$.

ii. To lower the variance of the estimate of the kernel, we can randomly chosen z_ω D times and stack them in vector $\mathbf{z}$. Use the inner product $\mathbf{z}(\mathbf{x})'\mathbf{z}(\mathbf{y}) = \frac{1}{D}\sum_{j=1}^{D} z_{\omega_j}(\mathbf{x}) z_{\omega_j}(\mathbf{y})$ to estimate $k(\mathbf{x}-\mathbf{y})$.

iii. Uniform convergence of Fourier features: Let $\mathcal{M}$ be a compact subset of $\mathcal{R}^d$ with diameter $diam(\mathcal{M})$. Then for the mapping $\mathbf{z}$ defined in Algorithm 1, we have

$$Pr[\sup_{x,y\in\mathcal{M}} |\mathbf{z}(\mathbf{x})'\mathbf{z}(\mathbf{y}) - k(\mathbf{x},\mathbf{y})| \geqslant \epsilon] \leqslant 2^8 (\frac{\sigma_p\, diag(\mathcal{M})}{\epsilon})^2 \exp(-\frac{D\epsilon^2}{4(d+2)}), \tag{3}$$

where $\sigma_p^2 \equiv E_p[\omega'\omega]$ is the second moment of the Fourier transform of k.

2 The Algorithm

Algorithm 1 Random Fourier Features.

Require: A positive definite shift-invariant kernel $k(\mathbf{x},\mathbf{y}) = k(\mathbf{x}-\mathbf{y})$.
Ensure: A randomized feature map $\mathbf{z}(\mathbf{x}) : \mathcal{R}^d \to \mathcal{R}^D$ so that $\mathbf{z}(\mathbf{x})'\mathbf{z}(\mathbf{y}) \approx k(\mathbf{x}-\mathbf{y})$.
Compute the Fourier transform p of the kernel k: $p(\omega) = \frac{1}{2\pi}\int e^{-j\omega'\delta} k(\delta)\, d\Delta$.
Draw D iid samples $\omega_1, \cdots, \omega_D \in \mathcal{R}^d$ from p and D iid samples $b_1, \ldots, b_D \in \mathcal{R}$ from the uniform distribution on $[0, 2\pi]$.
Let $\mathbf{z}(\mathbf{x}) \equiv \sqrt{\frac{2}{D}}\left[\cos(\omega_1'\mathbf{x}+b_1) \cdots \cos(\omega_D'\mathbf{x}+b_D)\right]'$.

3 Experiments

Dataset	Fourier+LS	Binning+LS	CVM	Exact SVM
CPU regression 6500 instances 21 dims	3.6% 20 secs $D = 300$	5.3% 3 mins $P = 350$	5.5% 51 secs	11% 31 secs ASVM
Census regression 18,000 instances 119 dims	5% 36 secs $D = 500$	7.5% 19 mins $P = 30$	8.8% 7.5 mins	9% 13 mins SVMTorch
Adult classification 32,000 instances 123 dims	14.9% 9 secs $D = 500$	15.3% 1.5 mins $P = 30$	14.8% 73 mins	15.1% 7 mins SVM^{light}
Forest Cover classification 522,000 instances 54 dims	11.6% 71 mins $D = 5000$	2.2% 25 mins $P = 50$	2.3% 7.5 hrs	2.2% 44 hrs libSVM
KDDCUP99 (see footnote) classification 4,900,000 instances 127 dims	7.3% 1.5 min $D = 50$	7.3% 35 mins $P = 10$	6.2% (18%) 1.4 secs (20 secs)	8.3% < 1 s SVM+sampling

Table 1: Comparison of testing error and training time between ridge regression with random features, Core Vector Machine, and various state-of-the-art exact methods reported in the literature. For classification tasks, the percent of testing points incorrectly predicted is reported. For regression tasks, the RMS error normalized by the norm of the ground truth is reported.

The experiments show that least squares regression on random Fourier features is a fast way to approximate the training of supervised kernel machines. They focus on the CVM (core vector machine) because it was shown in [2] to be both faster and more accurate than other known approaches. In this case, they trained by solving the least squares problem

$$\min_{\mathbf{w}} ||\mathbf{Z}'\mathbf{w} - \mathbf{y}||_2^2 + \lambda||\mathbf{w}||_2^2, \tag{4}$$

where $\mathbf{Z}$ denotes the matrix of random features. To evaluate the resulting machine on a datapoint $\mathbf{x}$, we can compute $\mathbf{w}'\mathbf{z}(\mathbf{x})$.

References

[1] Functions of Fourier Transforms. In *Fourier Analysis on Groups*, pages 131–155. John Wiley & Sons, Ltd, 2011.

[2] Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core Vector Machines: Fast SVM Training on Very Large Data Sets. *Journal of Machine Learning Research*, 6(Apr):363–392, 2005.