Approximating Kernel Matrices Using Random Features

Ahmet Emre Belge, 24-942-864

1 Introduction

Statistical learning algorithms often apply a feature map that lifts the input data into a typically higher-dimensional space. This helps transform the raw data into representations that capture the aspects of the data most relevant to the task at hand. In kernelized models, such as support vector machines (SVMs) [1] and Gaussian processes [2], the data enters the problem only through the inner products of the mapped points. That is, given inputs $\mathbf{x}$ and $\mathbf{y}$, we only need $\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle = \phi(\mathbf{x})^\top \phi(\mathbf{y})$, where $\phi(\cdot)$ is our feature map.

This is where the kernel trick comes in. Suppose we have a function $k(\mathbf{x}, \mathbf{y})$, such that $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$. In that case, we could directly evaluate the kernel function for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ without explicitly computing the features and then their inner product. While the kernel trick has important advantages in the computation of very high- or infinite-dimensional features, it does not scale well as the size of the training dataset gets bigger, since given the kernel matrix $K \in \mathbb{R}^{N \times N}$ with entries $k(\mathbf{x}, \mathbf{y})$ (given $\mathbf{x}$ and $\mathbf{y}$ in the training set of size N), exact kernel methods must form and manipulate K, incurring $O(N^2)$ memory and $O(N^3)$ time.

In their 2007 paper *Random Features for Large-Scale Kernel Machines*, Rahimi and Recht propose two randomized algorithms to approximate kernels by using a randomized feature map $\mathbf{z} : \mathbb{R}^d \to \mathbb{R}^D$ that is typically lower-dimensional than the original feature map ϕ. More specifically, we have:

$$\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle = k(\mathbf{x}, \mathbf{y}) \approx \mathbf{z}(\mathbf{x})^\top \mathbf{z}(\mathbf{y})$$

Through the construction of these random maps, one can directly transform the input data using $\mathbf{z}$ and then take the inner product in order to approximate the kernel. In this report, we present the random Fourier and random binning features algorithms, discuss their empirical performance, and conclude with a discussion of the advantages and trade-offs of randomization.

2 Random Fourier Features

A kernel $k(\mathbf{x}, \mathbf{y})$ is called shift-invariant (or stationary) if there exists a function $\tilde{k}$ such that $\tilde{k}(\mathbf{x}-\mathbf{y}) = k(\mathbf{x}, \mathbf{y})$. These types of kernels only depend on the relative locations of the inputs and not their absolute locations. From now on, for a shift-invariant kernel, we will drop the tilde and just use $k(\mathbf{x} - \mathbf{y})$. Since one can express a shift-invariant kernel as a function in one variable, $(\mathbf{x} - \mathbf{y})$, we can compute its Fourier transform, which we will denote with $p(\boldsymbol{\omega})$ (also called the spectral density of k). This will lead us to the following theorem [3].

Theorem 1 (Bochner)**.** *A continuous shift-invariant kernel on $\mathbb{R}^d$ is positive-definite if and only if its Fourier transform $p(\boldsymbol{\omega})$ is non-negative.*

As the kernel can be scaled, given a positive-definite and shift-invariant kernel k we can turn $p(\boldsymbol{\omega})$ into a proper probability distribution, which will be the foundation of the randomness in the algorithm. We will use the distribution $p(\boldsymbol{\omega})$ in order to rewrite $k(\mathbf{x}-\mathbf{y})$ as an expectation, which we will then approximate using Monte Carlo. For example, for the Gaussian kernel $k(\mathbf{x}, \mathbf{y}) = \exp(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2})$, we find $p(\boldsymbol{\omega}) = (2\pi)^{-\frac{D}{2}} \exp(-\frac{\|\boldsymbol{\omega}\|^2}{2})$. We can see this procedure more clearly with the following notation, where the integral can be interpreted as an inverse Fourier transform to recover the kernel from $p(\boldsymbol{\omega})$:

$$k(\mathbf{x} - \mathbf{y}) = \int_{\mathbb{R}^d} p(\boldsymbol{\omega}) e^{i\boldsymbol{\omega}^\top(\mathbf{x}-\mathbf{y})} d\boldsymbol{\omega} = \mathbb{E}_{\boldsymbol{\omega}\sim p}[e^{i\boldsymbol{\omega}^\top(\mathbf{x}-\mathbf{y})}] = \mathbb{E}_{\boldsymbol{\omega}\sim p}[e^{i\boldsymbol{\omega}^\top \mathbf{x}} e^{-i\boldsymbol{\omega}^\top \mathbf{y}}]$$

Both the kernel and the probability measure are real-valued, therefore, given the convergence of the integral, we can substitute the complex exponentials with cosine functions. Through further algebraic manipulations and trigonometric identities, we reach the following real-valued mapping satisfying our condition:

$$\mathbb{E}_{\boldsymbol{\omega},b}[z_{\boldsymbol{\omega},b}(\mathbf{x})^\top z_{\boldsymbol{\omega},b}(\mathbf{y})] = \mathbb{E}_{\boldsymbol{\omega}\sim p}[e^{i\boldsymbol{\omega}^\top(\mathbf{x}-\mathbf{y})}] = k(\mathbf{x}-\mathbf{y}) \quad \text{by setting} \quad z_{\boldsymbol{\omega},b}(\mathbf{x}) = \sqrt{2}\cos(\boldsymbol{\omega}^\top\mathbf{x}+b)$$

Where $\boldsymbol{\omega} \sim p(\boldsymbol{\omega})$ and $b \sim U[0, 2\pi]$. For the Monte Carlo estimation, we sample D instances of $\boldsymbol{\omega}$ and b, then for each instance j, we compute $z^{(j)}_{\boldsymbol{\omega},b}(\mathbf{x}) \cdot z^{(j)}_{\boldsymbol{\omega},b}(\mathbf{y})$, and then take their average:

$$k(\mathbf{x}-\mathbf{y}) = \mathbb{E}_{\boldsymbol{\omega},b}[z_{\boldsymbol{\omega},b}(\mathbf{x})^\top z_{\boldsymbol{\omega},b}(\mathbf{y})] \approx \frac{1}{D}\sum_{j=1}^{D} z^{(j)}_{\boldsymbol{\omega},b}(\mathbf{x}) z^{(j)}_{\boldsymbol{\omega},b}(\mathbf{y})$$

Given independent samples $\boldsymbol{\omega}, b$, we use the following notation:

$$\mathbf{z}(\mathbf{x})^\top \mathbf{z}(\mathbf{y}) = \frac{1}{D}\sum_{j=1}^{D} z^{(j)}_{\boldsymbol{\omega},b}(\mathbf{x}) z^{(j)}_{\boldsymbol{\omega},b}(\mathbf{y}) \quad \text{with} \quad \mathbf{z}(\mathbf{x}) = \frac{1}{\sqrt{D}}\left[z^{(1)}_{\boldsymbol{\omega},b}(\mathbf{x}),\ \ldots,\ z^{(D)}_{\boldsymbol{\omega},b}(\mathbf{x})\right]^\top$$

Rahimi and Recht (2007) show in Claim 1 that random Fourier features converge uniformly to the kernel. Their proof uses Hoeffding's inequality [4] as a starting point and then extends it via a covering-net argument to obtain uniform convergence over all $(\mathbf{x}, \mathbf{y}) \in \mathcal{M}$, where $\mathcal{M}$ is the subset of the input space over which we want the approximation guarantee to hold.

Claim 1: Let $\mathcal{M}$ be a compact subset of $\mathbb{R}^d$ with diameter $\text{diam}(\mathcal{M})$, then for a stationary kernel $k(\mathbf{x}, \mathbf{y})$, $\varepsilon > 0$, $\sigma_p^2 = E_p[\boldsymbol{\omega}^\top\boldsymbol{\omega}]$ and the scaled random feature vectors $z_{\boldsymbol{\omega},b}(\mathbf{x}), z_{\boldsymbol{\omega},b}(\mathbf{y})$ we have:

$$\Pr\left[\sup_{\mathbf{x},\mathbf{y}\in\mathcal{M}} \left|\mathbf{z}(\mathbf{x})^\top\mathbf{z}(\mathbf{y}) - k(\mathbf{x},\mathbf{y})\right| \geq \varepsilon\right] \leq 2^8\left(\frac{\sigma_p \,\text{diam}(\mathcal{M})}{\varepsilon}\right)^2 \exp\left(-\frac{D\varepsilon^2}{4(d+2)}\right)$$

We can immediately observe that the error probability decreases exponentially in the dimension of the Fourier feature space D. Moreover, if $D = \Omega\left(\frac{d}{\varepsilon^2}\log\left(\frac{\sigma_p \,\text{diam}(\mathcal{M})}{\varepsilon}\right)\right)$, then $\sup_{\mathbf{x},\mathbf{y}\in\mathcal{M}} \left|\mathbf{z}(\mathbf{x})^\top\mathbf{z}(\mathbf{y}) - k(\mathbf{x},\mathbf{y})\right| \leq \varepsilon$. That is, we approximate $k(\mathbf{x}, \mathbf{y})$ to within ε with any constant probability. Below, we find the pseudocode that wraps up our first randomized 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 $\mathbf{z}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^D$ so that $\mathbf{z}(\mathbf{x})^\top\mathbf{z}(\mathbf{y}) \approx k(\mathbf{x} - \mathbf{y})$
Compute the Fourier transform p of the kernel k: $p(\boldsymbol{\omega}) = \frac{1}{2\pi}\int e^{-i\boldsymbol{\omega}^\top(\mathbf{x}-\mathbf{y})}k(\mathbf{x}-\mathbf{y})\,d(\mathbf{x}-\mathbf{y})$.
Draw D independent samples $\boldsymbol{\omega_1}, \ldots, \boldsymbol{\omega_D} \in \mathbb{R}^d$ from p (given $\int p(\boldsymbol{\omega})d\boldsymbol{\omega} = 1$) , and D independent samples $b_1, \ldots, b_D \in \mathbb{R}$ from the uniform distribution on $[0, 2\pi]$.
Let $\mathbf{z}(\mathbf{x}) \equiv \sqrt{\frac{2}{D}}\left[\cos(\boldsymbol{\omega}^{(1)^\top}\mathbf{x} + b^{(1)}),\ \ldots,\ \cos(\boldsymbol{\omega}^{(D)^\top}\mathbf{x} + b^{(D)})\right]^\top$.

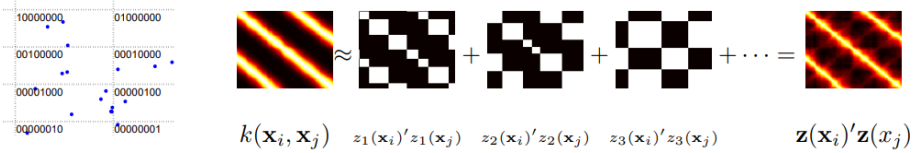

Figure 1: From Rahimi and Recht (2007), visual representation of random binning features

3 Random Binning Features

The intuition behind the second randomized algorithm presented in Rahimi and Recht (2007), which is particularly suited for approximating kernels that depend only on the L1 distance between input points, is as follows: Using rectilinear grids with random resolutions (sizes) and random shifts, we partition the input space (see Figure 1). These grids are constructed so that the probability of two input points, $\mathbf{x}$ and $\mathbf{y}$, falling into the same grid depends on the value of $k(\mathbf{x}, \mathbf{y})$. In each trial p, we construct the map $z_p(\mathbf{x})$ as a one-hot vector indicating which bin $\mathbf{x}$ falls into. Therefore, when taking the inner product of the random map for inputs $\mathbf{x}$ and $\mathbf{y}$, we get:

$$z_p(\mathbf{x})^\top z_p(\mathbf{y}) = \begin{cases} 1, & \text{if they share the same bin,} \\ 0, & \text{otherwise,} \end{cases} \qquad \text{and thus} \quad \mathbb{E}[z_p(\mathbf{x})^\top z_p(\mathbf{y})] = k(\mathbf{x}, \mathbf{y})$$

The construction of these random features starts with a randomized mapping to approximate the hat kernel on a segment of $\mathbb{R}$, where $k_{\text{hat}}(x, y; \delta) = \max\Big(0,\, 1 - \frac{|x-y|}{\delta}\Big)$. We partition the segment into intervals $[u + n\delta,\, u + (n+1)\delta]$ with grids of length δ and a random shift $u \sim U[0, \delta]$. If we number the bins so that a given point x falls into bin $\hat{x} = \lfloor \frac{x-u}{\delta} \rfloor$, then $\mathbb{P}_u[\hat{x} = \hat{y} | \delta] = k_{\text{hat}}(x, y; \delta)$ [5]. So, if we encode x in a one-hot vector indicating membership to a bin, then, analogously to what we have shown in the previous paragraph, we can approximate $k_{\text{hat}}(x, y; \delta)$ with $\mathbb{E}_u[z(x)^\top z(y) | \delta]$, making z a random map for approximating k_{hat}.

In the next step, we consider shift-invariant kernels that can be written as a convex combination of hat kernels, that is: $k(x, y) = \int_0^\infty k_{\text{hat}}(x, y; \delta) p(\delta)\, d\delta$. As shown in Lemma 1 [6], $p(\delta)$ can be recovered from k as $p(\delta) = \delta \ddot{k}(\delta)$. Here we note that, for some kernels, such as the Gaussian kernel, we do not obtain a valid random map with this construction, as $\delta \ddot{k}(\delta)$ is not non-negative. Given $\delta \sim p(\delta)$, and $u \sim U[0, \delta]$, then z provides an unbiased random map for k as follows:

$$\mathbb{E}_{\delta,u}\big[z(x)^\top z(y)\big] = \mathbb{E}_\delta\Big[\mathbb{E}_u\big[z(x)^\top z(y) \mid \delta\big]\Big] = \mathbb{E}_\delta\big[k_{\text{hat}}(x, y; \delta)\big] = k(x, y).$$

The final step of our construction extends the random map to separable multivariate shift-invariant kernels of the form $k(\mathbf{x} - \mathbf{y}) = \prod_{m=1}^d k_m(|x_m - y_m|)$. If each k_m can be written as a convex combination of hat kernels, then we can apply the 1-dimensional binning process described in the previous paragraph over each dimension d independently. In this case, the probability that x_m and y_m are in the same bin in dimension m is $k_m(|x_m - y_m|)$, and since the binning is independent across dimensions, the probability that $\mathbf{x}$ and $\mathbf{y}$ are binned together in every dimension is $\prod_{m=1}^d k_m(|x_m - y_m|) = k(\mathbf{x} - \mathbf{y})$. In this multivariate construction, $z(\mathbf{x})$ theoretically encodes the integer vector $[\hat{\mathbf{x}}_1, \cdots, \hat{\mathbf{x}}_d]$ corresponding to each bin of the d-dimensional grid as a binary indicator vector [6]. However, in practice, to avoid overflow in computing $z(\mathbf{x})$ when d is large, as the total number of grid cells is the product of the number of bins along each axis, growing exponentially in d and quickly exceeding available memory, we remove unoccupied bins from the representation.

To estimate the expectation, we follow the usual Monte Carlo approach by constructing P independent random binning features and then taking their average as follows: $\mathbf{z}(\mathbf{x})^\top \mathbf{z}(\mathbf{y}) = \sum_{p=1}^P z_p(\mathbf{x})^\top z_p(\mathbf{y})$.

Regarding the convergence of the estimate using random binning features to the kernel, Rahimi and Recht (2007) state a bound that is analogous to the uniform bound derived for random Fourier features.

Claim 2: Let $\mathcal{M}$ be a compact subset of $\mathbb{R}^d$ with diameter $\text{diam}(\mathcal{M})$. Let $\alpha = \mathbb{E}[1/\delta]$ and L_k the Lipschitz

constant of k w.r.t. the L1 norm. With $\mathbf{z}$ constructed as above and any $\varepsilon > 0$, we have:

$$\Pr\Big[\sup_{\mathbf{x},\mathbf{y}\in\mathcal{M}} \big|\, \mathbf{z}(\mathbf{x})^\top \mathbf{z}(\mathbf{y}) \;-\; k(\mathbf{x},\mathbf{y})\big| \;\leq \varepsilon\Big] \;\geq\; 1 - 36dP\alpha\,\mathrm{diam}(\mathcal{M})\exp\Big(-\frac{\frac{P\varepsilon^2}{8} \;+\; \ln\frac{\varepsilon}{L_k}}{d+1}\Big)$$

Below, we find the pseudocode for our second randomized algorithm.

Algorithm 2 Random Binning Features

Require: A point $\mathbf{x} \in \mathbb{R}^d$, a kernel function $k(\mathbf{x},\mathbf{y}) = \prod_{m=1}^{d} k_m(|x_m - y_m|)$ so that $p(\Delta) \equiv \Delta\ddot{k}_m(\Delta)$ is a probability distribution on $\Delta \geq 0$

Ensure: A randomized feature map $\mathbf{z}(\mathbf{x})$ so that $\mathbf{z}(\mathbf{x})^\top \mathbf{z}(\mathbf{y}) \approx k(\mathbf{x}-\mathbf{y})$

for $p = 1, ..., P$ **do**

Draw $\delta, \mathbf{u} \in \mathbb{R}^d$ with $\delta_d \sim p_m$ and $u_m \sim U[0, \delta_m]$

Let z return the coordinate of the bin containing $\mathbf{x}$ as a binary indicator vector $z_p(\mathbf{x}) \equiv \mathrm{hash}(\lceil\frac{x_1-u_1}{\delta_1}\rceil, ..., \lceil\frac{x_d-u_d}{\delta_d}\rceil)$

end for

$\mathbf{z}(\mathbf{x}) \equiv \sqrt{\frac{1}{P}}[z_1(\mathbf{x}), ..., z_p(\mathbf{x})]^\top$.

4 Empirical Evidence

We now briefly discuss the experiments presented in Rahimi and Recht (2007). Below, we compare, in three standard large-scale datasets and their respective tasks, the training time and performance of ridge regression with random Fourier features and random binning features with core vector machines [7] [1] and state-of-the-art exact support vector machine algorithms.

Table 1: Comparison of testing error and training time between ridge regression with random features, CVM, and various state-of-the-art exact methods reported in the literature. For classification tasks, the percentage of incorrect predictions is reported. For regression tasks, the root mean squared error normalized by the norm of the ground truth is reported.

Dataset	Fourier+LS	Binning+LS	CVM	Exact SVM
Census regression 18,000 instances, 119 dims $D = 500,\ P = 30$	5% 36 secs	7.5% 19 mins	8.8% 7.5 mins	9% 13 mins (SVMTorch)
Adult binary classification 32,000 instances, 123 dims $D = 500,\ P = 30$	14.9% 9 secs	15.3% 1.5 mins	14.8% 73 mins	15.1% 7 mins (SVMlight)
Forest Cover multiclass classification 522,000 instances, 54 dims $D = 5000,\ P = 50$	11.6% 71 mins	2.2% 25 mins	2.3% 7.5 hrs	2.2% 44 hrs (libSVM)

We can observe that ridge regression with random features, despite its simplicity, significantly accelerated training time while still providing a similar, if not better, performance with respect to CVM and exact SVM methods. An important observation is that while random Fourier features performed better in tasks that rely on interpolation and with smoother decision boundaries, such as regression and binary classification, random binning features perform better on tasks that require memorization, which are tasks where the optimal predictor essentially has to "remember" lots of individual training examples as the decision boundary or regression

[1] Core vector machines speed up SVM training by reformulating it as finding the smallest enclosing ball around the data in feature space and then using only a small "coreset" of key points.

surface is highly non-smooth or piecewise. This is because random binning features explicitly preserve locality in the input space. We can observe this in the remarkable advantage of random binning features on Fourier features in the Forest Cover task where the decision boundary is non-smooth.

5 Discussion & Conclusion

Since the random map $\mathbf{z}$ yields a fixed, lower-dimensional design matrix, training reduces to a standard linear problem. This also helps to decouple the choice of solver from the kernel, allowing us to use any off-the-shelf optimizer, such as stochastic gradient descent (SGD), without any modifications.

In the table below, we compare the time and memory complexity during training of a general kernel method versus random feature approaches.

	Time	**Memory**
Exact Kernel	$O(N^3)$	$O(N^2)$
Random Fourier Features	$O(ND^2 + D^3)$	$O(ND + D^2)$
Random Binning Features	$O(NP^2 + P^3)$	$O(NP + P^2)$

We can see that in cases where $D << N$ (and $P << N$), our algorithms provide a significant computational advantage. It is important to note that these costs assume the implementation of a closed-form solution and can be improved with iterative solvers (such as SGD).

The correctness and power of these randomized algorithms are rooted in the principles of Monte Carlo estimation, and in particular, the law of large numbers. The (strong) law of large numbers states that, under certain conditions, we can approximate an expectation $\mathbb{E}_p[f(x)]$ with $\frac{1}{n}\sum_{i=1}^{n} f(x_i)$, $x_i \overset{\text{iid}}{\sim} p$, and that this approximation will converge almost surely to the expectation as the sample size n goes to infinity. Furthermore, thanks to bounds derived from Hoeffding's inequality in Claims 1-2, we have important guarantees for the approximation errors of these random maps.

Even with the above guarantees, methods that make use of randomness through Monte Carlo estimation of expectations can suffer from high variance, an unlucky draw can inflate error. Moreover, another point to keep in mind with randomized algorithms is that the concentration bounds are only high-probability, they offer no worst-case guarantee. Therefore, a small residual failure probability must be tolerated or otherwise mitigated in safety-critical settings.

Our conclusion is that random features trade an acceptable loss in approximation accuracy for orders-of-magnitude speed-ups, making kernel methods viable on today's large-scale data while retaining strong theoretical guarantees.

References

[1] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[2] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[3] Walter Rudin. *Fourier Analysis on Groups*. Wiley Classics Library. Wiley-Interscience, New York, reprint edition edition, 1994.

[4] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[5] Piotr Indyk and Narayanan Thaper. Fast image retrieval via embeddings. In *Proceedings of the International Workshop on Statistical and Computational Theories of Vision*, 2003.

[6] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. 2007.

[7] Ivor Tsang, James Kwok, and Pak-Ming Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 04 2005.