Fastfood — Approximating Kernel Expansions in Loglinear Time

Quoc Le QVL@GOOGLE.COM
Tamás Sarlós STAMAS@GOOGLE.COM
Alex Smola ALEX@SMOLA.ORG
Google Knowledge, 1600 Amphitheatre Pkwy, Mountain View 94043, CA USA

Abstract

Despite their successes, what makes kernel methods difficult to use in many large scale problems is the fact that computing the decision function is typically expensive, especially at prediction time. In this paper, we overcome this difficulty by proposing Fastfood, an approximation that accelerates such computation significantly. Key to Fastfood is the observation that Hadamard matrices when combined with diagonal Gaussian matrices exhibit properties similar to dense Gaussian random matrices. Yet unlike the latter, Hadamard and diagonal matrices are inexpensive to multiply and store. These two matrices can be used in lieu of Gaussian matrices in Random Kitchen Sinks (Rahimi & Recht, 2007) and thereby speeding up the computation for a large range of kernel functions. Specifically, Fastfood requires $O(n \log d)$ time and $O(n)$ storage to compute n non-linear basis functions in d dimensions, a significant improvement from $O(nd)$ computation and storage, without sacrificing accuracy. We prove that the approximation is unbiased and has low variance. Extensive experiments show that we achieve similar accuracy to full kernel expansions and Random Kitchen Sinks while being 100x faster and using 1000x less memory. These improvements, especially in terms of memory usage, make kernel methods more practical for applications that have large training sets and/or require real-time prediction.

1. Introduction

Kernel methods are successful techniques for solving many problems in machine learning, ranging from classification and regression to sequence annotation and feature extraction (Boser et al., 1992; Cortes & Vapnik, 1995; Vapnik et al., 1997; Taskar et al., 2004; Schölkopf et al., 1998). At their heart lies the idea that inner products in high-dimensional feature spaces can be computed in an implicit form via a kernel function k:

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

Here $\phi : \mathcal{X} \to \mathcal{F}$ maps elements of the observation space $\mathcal{X}$ into a high-dimensional feature space $\mathcal{F}$. Key to kernel methods is that as long as kernel algorithms have access to k, we do not need to represent $\phi(x)$ explicitly. Most often, that means although $\phi(x)$ can be high-dimensional or even infinite-dimensional, their inner products, can be evaluated in an inexpensive manner by k. This idea is known as the "kernel trick."

More concretely, to evaluate the decision function $f(x)$ on an example x, one typically employs the kernel trick as follows

$$f(x) = \langle w, \phi(x) \rangle = \left\langle \sum_{i=1}^{N} \alpha_i \phi(x_i), \phi(x) \right\rangle = \sum_{i=1}^{N} \alpha_i k(x_i, x)$$

This has been viewed as a strength of kernel methods, especially in the days that datasets consisted of ten thousands of examples. This is because the Representer Theorem (Kimeldorf & Wahba, 1970) states that such a function expansion in terms of finitely many coefficients must exist under fairly benign conditions even whenever the space is infinite dimensional. Hence we can effectively perform optimization in infinite dimensional spaces.

Unfortunately, on large amounts of data, this expansion turns into a significant limitation for computational efficiency. For instance, (Steinwart & Christmann, 2008) show that the number of nonzero α_i (i.e., N, also known as the number of "support vectors") in many estimation problems can grow linearly in the size of the training set. As a consequence, as the dataset grows, the expense of evaluating f also grows. This property makes kernel methods expensive in many large scale problems.

Random Kitchen Sinks (Rahimi & Recht, 2007; 2008)[1], the algorithm that our algorithm is based on,

Proceedings of the 30^{th} International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

[1](Rahimi & Recht, 2007) introduced the method, and

approximates the function f by means of multiplying the input with a Gaussian random matrix, followed by the application of a nonlinearity. If the expansion dimension is n and the input dimension is d (i.e., the Gaussian matrix is $n \times d$), it requires $O(nd)$ time and memory to evaluate the decision function f. For large problems with sample size $m \gg n$, this is typically much faster than the aforementioned "kernel trick" because the computation is independent of the size of the training set. Experiments also show that this approximation method achieves accuracy comparable to RBF kernels while offering significant speedup.

Our proposed approach, Fastfood, accelerates Random Kitchen Sinks from $O(nd)$ to $O(n \log d)$ time. The speedup is most significant when the input dimension d is larger than 1000, which is typical in most applications. For instance, a tiny 32x32x3 image in the CIFAR-10 (Krizhevsky, 2009) already has 3072 dimensions (and non-linear function classes have shown to work well for MNIST (Schölkopf & Smola, 2002) and CIFAR-10). Our approach relies on the fact that Hadamard matrices, when combined with Gaussian scaling matrices, behave very much like Gaussian random matrices. That means these two matrices can be used in place of Gaussian matrices in Random Kitchen Sinks and thereby speeding up the computation for a large range of kernel functions. The computational gain is achieved because unlike Gaussian random matrices, Hadamard matrices and scaling matrices are inexpensive to multiply and store.

We prove that the Fastfood approximation is unbiased, has low variance, and concentrates almost at the same rate as Random Kitchen Sinks. Moreover, extensive experiments with a wide range of datasets show that Fastfood achieves similar accuracy to full kernel expansions and Random Kitchen Sinks while being 100x faster with 1000x less memory. These improvements, especially in terms of memory usage, make it possible to use kernel methods even for embedded applications. Our experiments also demonstrate that Fastfood, thanks to its speedup in training, achieves state-of-the-art accuracy on the CIFAR-10 dataset (Krizhevsky, 2009) among permutation-invariant methods.

Other related work Speeding up kernel methods has been a research focus for many years. Early work compresses function expansions after the problem was solved (Burges, 1996) by means of reduced-set expansions. Subsequent work aimed to reduce memory footprint and complexity by finding subspaces to expand functions (Smola & Schölkopf, 2000; Fine & Scheinberg, 2001; Williams & Seeger, 2001). They typically require $O(n^3 + mnd)$ steps to process m observations and to expand d dimensional data into an n-dimensional function space. Moreover, they require $O(n^2)$ storage at least at preprocessing time to obtain suitable basis functions. Despite these efforts, these costs are still expensive for practical applications.

Along the lines of Rahimi & Recht (2007; 2008)'s work, fast multipole expansions (Lee & Gray, 2009; Gray & Moore, 2003) offer another interesting avenue for efficient function expansions. While this idea is attractive when the dimensionality of the input dimension d is small, they become computationally intractable for large d's due to the curse of dimensionality in terms of partitioning.

2. Random Kitchen Sinks

We start by reviewing some basic tools from kernel methods (Schölkopf & Smola, 2002) and then analyze key ideas behind Random Kitchen Sinks.

2.1. Mercer's Theorem and Expansions

At the heart of kernel methods is the theorem of (Mercer, 1909) which guarantees that kernels can be expressed as an inner product in some Hilbert space.

Theorem 1 (Mercer) *Any kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ satisfying $\int k(x, x')f(x)f(x')dxdx' \geq 0$ for all $L_2(\mathcal{X})$ measurable functions f can be expanded into*

$$k(x, x') = \sum_j \lambda_j \phi_j(x)\phi_j(x') \tag{2}$$

Here $\lambda_j > 0$ and the ϕ_j are orthonormal on $L_2(\mathcal{X})$.

The key idea of (Rahimi & Recht, 2007; 2008) is to use sampling to approximate the sum in (2). In other words, they draw[2]

$$\lambda_i \sim p(\lambda) \text{ where } p(\lambda_i) \propto \lambda_i \tag{3}$$

$$\text{and } k(x, x') \approx \frac{\sum_j \lambda_j}{n} \sum_{i=1}^{n} \phi_{\lambda_i}(x)\phi_{\lambda_i}(x') \tag{4}$$

Note that the basic connection between random basis functions was well established, e.g., by Neal (1994) in proving that the Gaussian Process is a limit of an infinite number of basis functions. The expansion (3) is possible whenever the following conditions hold:

1. An inner product expansion of the form (2) is known for a given kernel k.
2. The basis functions ϕ_j are sufficiently inexpensive to compute.
3. The sum $\sum_j \lambda_j < \infty$ converges, i.e., k corresponds to a trace class operator (Kreyszig, 1989).

(Rahimi & Recht, 2008) generalized it further. We refer to it with the title of the latter instead of the ambiguous phrase "random features" to better differentiate from our own.

[2] The normalization arises from the fact that it is an n-sample from the distribution over basis functions.

Although condition 2 is typically difficult to achieve, there exist special classes of expansions that are computationally attractive. Specifically, whenever the kernels are invariant under an action of a symmetry group, we can use the eigenfunctions of its representation to diagonalize the kernel.

For instance, for the translation group the Fourier basis diagonalizes its action because translations can be represented by multiplications in Fourier space. Likewise, the rotation group $\mathrm{SO}(n)$ leads to spherical harmonics as the matching representation. For the symmetric group (i.e., permutations) we obtain corresponding invariants. In particular, a major focus of Random Kitchen Sinks is the class of translation invariant kernels that can be written as a function of $x - x'$ and have the properties

$$k(x, x') = k(x - x', 0).$$

Here the eigenfunctions are given by the Fourier basis

$$k(x, x') = \int_z \overline{\phi_z(x)} \phi_z(x') \lambda(z) \text{ and } \phi_z(x) = e^{i\langle z, x\rangle}. \tag{5}$$

Here $\lambda(z) \geq 0$ is a kernel-specific weight that quantifies how much high frequency components are penalized. By construction the function $\lambda(z)$ is quite easily obtained by applying the Fourier transform to $k(x, 0)$ — in this case the above expansion is simply the inverse Fourier transform:

$$\lambda(z) = (2\pi)^{-d} \int e^{i\langle x, z\rangle} k(x, 0) dx. \tag{6}$$

This technique allows us to obtain explicit Fourier expansions for a wide class of kernel functions (Gaussian RBF, Laplace, Matern, etc.). For instance, for Gaussian RBF kernel it is a Gaussian with the inverse covariance structure. For the Laplace kernel it yields the damped harmonic oscillator spectrum and for the Matern kernel, i.e., Bessel functions, this yields the convolutions of the unit ball (Schölkopf & Smola, 2002).

2.2. Random Kitchen Sinks for Gaussians

Rahimi & Recht (2008) use this property of $\lambda(z)$ to generate approximations to the Gaussian RBF kernel, $k(x, x') = \exp(-\|x - x'\|^2 / (2\sigma^2))$, by drawing values z_i from a normal distribution:

input Scale σ^2, n, d
Sample entries in $Z \in \mathbb{R}^{n \times d}$ i.i.d. from $\mathcal{N}(0, \sigma^{-2})$.
for all x **do**
  Use (6) and compute empirical feature map
  $\phi_j(x) = \frac{1}{\sqrt{n}} \exp(i[Zx]_j)$
end for

As derived above, the associated feature map converges in expectation to the Gaussian RBF kernel. In fact, convergence occurs with high probability and at the rate of independent empirical averages (Rahimi & Recht, 2007; 2008). This allows one to use primal space methods for training, and thus prevents the cost of computing decision function from growing as the dataset grows.

This approach is still limited by the fact that we need to store Z and, more importantly, we need to compute Zx for each x. That is, each observation costs $O(nd)$ operations and we need $O(nd)$ storage. In the next section, we propose Fastfood that improves Random Kitchen Sinks further by approximating the random matrix using a set of simple transforms.

3. Fastfood

Our main contribution is to show strategies for accelerating Zx from $O(nd)$ to $O(n \log d)$ time and how this can be used for constructing kernels using arbitrary spectral distributions $\lambda(z)$ provided that they are spherically invariant, i.e., they must only depend on $\|z\|_2$. In a nutshell, the approach relies on the fact that Hadamard matrices, when combined with Gaussian scaling matrices, behave very much like Gaussian random matrices. The adaptation to distributions other than Gaussians then occurs via rescaling by coefficients drawn from the equivalent radial distribution.

3.1. Gaussian RBF Kernels

We begin with the Gaussian RBF case and extend it to more general spectral distributions subsequently. Without loss of generality assume that $d = 2^l$ for some $l \in \mathbb{N}$.[3] For the moment assume that $d = n$. The matrices that we consider instead of Z are parameterized by a product of diagonal and simple matrices:

$$V = \frac{1}{\sigma\sqrt{d}} SHG\Pi HB. \tag{7}$$

Here $\Pi \in \{0, 1\}^{d \times d}$ is a permutation matrix and H is the Walsh-Hadamard matrix.[4] S, G and B are all *diagonal* random matrices. More specifically, B has random $\{\pm 1\}$ entries on its main diagonal, G has random Gaussian entries, and S is a random scaling matrix. V is then used to compute the feature map.

The coefficients for S, G, B are computed once and stored. The Walsh-Hadamard matrix is given by

$$H_2 := \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \text{ and } H_{2d} := \begin{bmatrix} H_d & H_d \\ H_d & -H_d \end{bmatrix}.$$

[3] If this is not the case, we can trivially pad the vectors with zeros until $d = 2^l$ holds.

[4] We conjecture that H can be replaced by any matrix $T \in \mathbb{R}^{d \times d}$, such that $T/\sqrt{d}$ is orthonormal, $\max_{ij} |T_{ij}| = O(1)$, i.e. T is smooth, and Tx can be computed in $O(d \log d)$ time. A natural candidate is the Discrete Cosine Transform (DCT); we defer its analysis to the journal version of the paper.

The fast Hadamard transform, a variant of the FFT, allows us to compute $H_d x$ in $O(d \log d)$ time.

When $n > d$, we replicate (7) for n/d independent random matrices V_i and stack them via $V^T = [V_1, V_2, \dots V_{n/d}]^T$ until we have enough dimensions. The feature map for Fastfood is then defined as

$$\phi_j(x) = n^{-\frac{1}{2}} \exp(i[Vx]_j). \tag{8}$$

In the next section, we will prove that this feature map approximates the RBF kernel. The rest of this section will focus on its attractiveness in terms of computational efficiency.

Lemma 2 (Computational Efficiency) *The features of (8) can be computed at $O(n \log d)$ cost using $O(n)$ permanent storage for $n \geq d$.*

Proof Storing the matrices S, G, B costs $3n$ entries and $3n$ operations for a multiplication. The permutation matrix Π costs n entries and n operations. The Hadamard matrix itself requires no storage since it is only implicitly represented. Furthermore, the fast Hadamard transforms costs $O(n \log d)$ operations to carry out (we have $O(d \log d)$ per block and n/d blocks). Computing the Fourier basis for n numbers is an $O(n)$ operation. Hence the total CPU budget is $O(n \log d)$ and the storage is $O(n)$. ∎

Note that the construction of V is analogous to that of (Dasgupta et al., 2011). We will use these results in establishing a sufficiently high degree of decorrelation between rows of V. Also note that multiplying with a longer chain of Walsh-Hadamard matrices and permutations would yield a distribution closer to independent Gaussians. However, as we shall see, two matrices provide a sufficient amount of decorrelation.

3.2. Basic Properties

Now that we showed that the above operation is *fast*, let us give some initial indication why it is also *useful* and how the remaining matrices S, G, B, Π are defined.

Binary scaling matrix B: It is a diagonal matrix with $B_{ii} \in \{\pm 1\}$ drawn iid. The initial $HBd^{-\frac{1}{2}}$ acts as an isometry that makes the input dense (Ailon & Chazelle, 2009).

Permutation Π: This ensures that the rows of the two Walsh-Hadamard matrices are incoherent relative to each other. Π can be stored efficiently as a lookup table at $O(d)$ cost and it can be generated by sorting random numbers.

Gaussian scaling matrix G: This is a diagonal matrix whose elements $G_{ii} \sim \mathcal{N}(0, 1)$ are drawn iid from a Gaussian. The next Walsh-Hadamard matrices H will allow us to 'recycle' n Gaussians to make the resulting matrix closer to an iid Gaussian. The goal of the preconditioning steps above is to guarantee that no single G_{ii} can influence the output too much and hence provide near-independence.

Scaling matrix S: Note that the length of all rows of $HG\Pi HB$ are constant as equation (11) shows below. In the Gaussian case S ensures that the length distribution of the row of V are independent of each other. In the more general case, one may also adjust the capacity of the function class via a suitably chosen scaling matrix S. That is, large values in S_{ii} correspond to high complexity basis functions whereas small S_{ii} relate to simple functions with low total variation. For the RBF kernel we choose

$$S_{ii} = s_i \|G\|_{\text{Frob}}^{-\frac{1}{2}} \tag{9}$$

$$s_i \sim (2\pi)^{-\frac{d}{2}} A_{d-1}^{-1} r^{d-1} e^{-\frac{r^2}{2}}. \tag{10}$$

Here the normalization constant A_{d-1} denotes the surface volume of the d-dimensional unit ball. Thus s_i matches the radial part of a normal distribution and we rescale it using the Frobenius norm of G.

We now analyze the distribution of entries in V.

The rows of $HG\Pi HB$ have the same length. To compute their length we take

$$\begin{aligned} l^2 &:= \left[HG\Pi HB(HG\Pi HB)^\top\right]_{jj} \\ &= [HG^2H]_{jj} d = \sum_i H_{ij}^2 G_{ii}^2 d = \|G\|_{\text{Frob}}^2 d \end{aligned} \tag{11}$$

Rescaling with $\|G\|_{\text{Frob}}^{-\frac{1}{2}} d^{-\frac{1}{2}}$ yields length 1 rows.

Any given row of $HG\Pi HB$ is iid Gaussian. Each entry $[HG\Pi HB]_{ij} = B_{jj} H_i^T G \Pi H_j$ is zero-mean Gaussian as it consists of a sum of zero-mean independent Gaussian random variables. Sign changes retain Gaussianity. Also note that $\text{Var}\,[HG\Pi HB]_{ij} = d$. B ensures that different entries in $[HG\Pi HB]_{i\cdot}$ have 0 correlation. Hence they are iid Gaussian (checking first and second order moments suffices).

The rows of $SHG\Pi HB$ are Gaussian. Rescaling the length of a Gaussian vector using (10) retains Gaussianity. Hence the rows of $SHG\Pi HB$ are Gaussian (albeit not independent).

Lemma 3 *The expected feature map recovers the Gaussian RBF kernel, i.e.,*

$$\mathbf{E}_{S,G,B,\Pi}\left[\overline{\phi(x)}^\top \phi(x')\right] = e^{-\frac{\|x-x'\|^2}{2\sigma^2}}.$$

Moreover, the same holds for $V' = \frac{1}{\sigma\sqrt{d}} HG\Pi HB$.

Proof We already saw above that any given row in V is a random Gaussian vector with distribution

$\mathcal{N}(0, \sigma^{-2} I_d)$, hence we can directly appeal to the construction of (Rahimi & Recht, 2008). This also holds for V'. The main difference being that the rows in V' are considerably more correlated. ■

3.3. Approximation Guarantees

In this section we prove that the approximation that we incur relative to a Gaussian random matrix is mild.

3.3.1. Low Variance

Theorem 4 shows that when approximating the RBF kernel with n features the variance of Fastfood (even without the scaling matrix S) is at most the variance of straightforward Gaussian features, the first term in (12), plus $O(1/n)$. In fact, we will see in experiments that our approximation works as well as an exact kernel expansion and Random Kitchen Sinks.

Since the kernel values are real numbers, let us consider the real version of the complex feature map ϕ for simplicity. Set $y_j = [V(x' - x)]_j$ and recall that

$$\overline{\phi_j(x)}\phi_j(x') = n^{-1}\exp(iy_j) = n^{-1}(\cos(y_j) + i\sin(y_j)).$$

Thus we can replace $\phi(x) \in \mathbb{C}^n$ with $\phi'(x) \in \mathbb{R}^{2n}$, where $\phi'_{2j-1}(x) = n^{-1/2}\cos([Vx]_j)$ and $\phi'_{2j}(x) = n^{-1/2}\sin([Vx]_j)$, see (Rahimi & Recht, 2007).

Theorem 4 *Let $C(\alpha) = 6\alpha^4\left[e^{-\alpha^2} + \frac{\alpha^2}{3}\right]$ and $v = (x - x')/\sigma$. Then for the feature map $\phi' : \mathbb{R}^d \to \mathbb{R}^{2n}$ obtained by stacking n/d i.i.d. copies of matrix $V' = \frac{1}{\sigma\sqrt{d}} HG\Pi HB$ we have that*

$$\mathrm{Var}\left[\phi'(x)^\top \phi'(x')\right] \leq \frac{2\left(1 - e^{-\|v\|^2}\right)^2}{n} + \frac{C(\|v\|)}{n}. \tag{12}$$

Moreover, the same holds for $V = \frac{1}{\sigma\sqrt{d}} SHG\Pi HB$.

Proof $\phi'(x)^\top \phi'(x')$ is the average of n/d independent estimates, each arising from $2d$ features. Hence it's sufficient to prove the claim for a single block, i.e. when $n = d$. We show the latter for V' in Theorem 5 and omit the near identical argument for V. ■

Theorem 5 *Let $v = (x - x')/\sigma$ and let $\psi_j(v) = \cos(d^{-\frac{1}{2}}[HG\Pi HBv]_j)$ denote the estimate of the kernel value that comes from the jth pair of random features for each $j \in \{1 \dots d\}$. Then for each j we have*

$$\mathrm{Var}\left[\psi_j(v)\right] = \frac{1}{2}\left(1 - e^{-\|v\|^2}\right)^2, \text{ and} \tag{13}$$

$$\mathrm{Var}\left[\sum_{j=1}^{d} \psi_j(v)\right] \leq \frac{d}{2}\left(1 - e^{-\|v\|^2}\right)^2 + dC(\|v\|) \tag{14}$$

where $C(\alpha) = 6\alpha^4\left[e^{-\alpha^2} + \frac{\alpha^2}{3}\right]$.

Proof Since $\mathrm{Var}(\sum X_j) = \sum_{j,t} \mathrm{Cov}(X_j, X_t)$ for any random variable X_j, our goal is to compute $\mathrm{Cov}(\psi(v), \psi(v)) = \mathbf{E}\left(\psi(v)\psi(v)^T\right) - \mathbf{E}(\psi(v))\mathbf{E}(\psi(v))^T$. Let $w = \frac{1}{\sqrt{d}} HBv$, $u = \Pi w$, and $z = HGu$ and hence $\psi_j(v) = \cos(z_j)$. Now condition on the value of u. Then it follows that $\mathrm{Cov}(z_j, z_t|u) = \rho_{jt}(u)\|v\|^2$, where $\rho_{jt}(u) \in [-1, 1]$ is the correlation of z_j and z_t.

To simplify the notation, in what follows we write ρ instead of $\rho_{jt}(u)$. Observe that the marginal distribution of each z_j is $\mathcal{N}(0, \|v\|^2)$ as $\|u\| = \|v\|$ and each element of H is ± 1. Thus the joint distribution of z_j and z_t is a Gaussian with mean 0 and covariance

$$\mathrm{Cov}\left[[z_j, z_t]|u\right] = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \|v\|^2 = L \cdot L^T,$$

$L = \begin{bmatrix} 1 & 0 \\ \rho & \sqrt{1-\rho^2} \end{bmatrix} \|v\|$ is its Cholesky factor. Hence

$$\begin{aligned} &\mathrm{Cov}(\psi_j(v), \psi_t(v)|u) \\ =&\mathbf{E}_g\left[\cos([Lg]_1)\cos([Lg]_2)\right] - \mathbf{E}_g[\cos(z_j)]\mathbf{E}_g[\cos(z_t)] \end{aligned} \tag{15}$$

where $g \in \mathbb{R}^2$ is drawn from $\mathcal{N}(0, \mathbf{1})$. From the trigonometric identity

$$\cos(\alpha)\cos(\beta) = \frac{1}{2}\left[\cos(\alpha - \beta) + \cos(\alpha + \beta)\right]$$

it follows that we can rewrite

$$\begin{aligned} \mathbf{E}_g\left[\cos([Lg]_1)\cos([Lg]_2)\right] =& \frac{1}{2}\mathbf{E}_h\left[\cos(a_- h) + \cos(a_+ h)\right] \\ =& \frac{1}{2}\left[e^{-\frac{1}{2}a_-^2} + e^{-\frac{1}{2}a_+^2}\right] \end{aligned}$$

$h \sim \mathcal{N}(0, 1)$ and $a_\pm^2 = \left\|L^\top [1, \pm 1]\right\|^2 = 2\|v\|^2(1 \pm \rho)$. That is, after applying the addition theorem we explicitly computed the now one-dimensional Gaussian integrals.

Likewise, since by construction z_j and z_j have zero mean and variance $\|v\|^2$ we have that

$$\mathbf{E}_g[\cos(z_j)]\mathbf{E}_g[\cos(z_t)] = \mathbf{E}_h[\cos(\|v\| h)]^2 = e^{-\|v\|^2}$$

Combining both terms we obtain that the covariance can be written as

$$\mathrm{Cov}[\psi_j(v), \psi_t(v)|u] = e^{-\|v\|^2}\left[\cosh[\|v\|^2 \rho] - 1\right] \tag{16}$$

To prove the first claim realize that here $j = t$ and correspondingly $\rho = 1$. Plugging this into the above covariance expression and simplifying terms yields our first claim (13).

To prove our second claim, observe that from the Taylor series of cosh with remainder in Lagrange form, it

follows that there exists $\eta \in [-\|v\|^2 |\rho|, \|v\|^2 |\rho|]$ such

$$\begin{aligned}\cosh(\|v\|^2 \rho) =& 1 + \frac{1}{2}\|v\|^4 \rho^2 + \frac{1}{6}\sinh(\eta)\|v\|^6 \rho^3 \\ \leq& 1 + \frac{1}{2}\|v\|^4 \rho^2 + \frac{1}{6}\sinh(\|v\|^2)\|v\|^6 \rho^3 \\ \leq& 1 + \rho^2 \|v\|^4 B(\|v\|),\end{aligned}$$

where $B(\|v\|) = \frac{1}{2} + \frac{\sinh(\|v\|^2)\|v\|^2}{6}$. From (16) we have

$$\mathrm{Cov}[\psi_j(v), \psi_t(v)|u] \leq \rho^2 \|v\|^4 B(\|v\|).$$

Note that we still conditioned on u. What remains is to bound $\mathbf{E}_u[\rho^2]$, which is small if $\mathbf{E}[\|u\|_4^4]$ is small. The latter is ensured by HB, which acts as a randomized preconditioner. These calculations are fairly standard and can be found in Appendix A.1 of the supplementary material. ■

3.3.2. Concentration

The following theorem shows that given error probability δ, the approximation error of a $d \times d$ block of Fastfood is at most $O(\sqrt{\log(d/\delta)})$ times larger than the error of Random Kitchen Sinks. We believe that this bound is not tight and could be further improved. We defer analyzing the concentration of $n > d$ stacked Fastfood features to future work.

Theorem 6 *For all $x, x' \in \mathbb{R}^d$ let $\hat{k}(x, x') = \sum_{j=1}^d \cos(d^{-\frac{1}{2}}[HG\Pi HB(x - x')/\sigma]_j)/d$ denote our estimate of the RBF kernel $k(x, x')$ that arises from a $d \times d$ block of Fastfood. Then we have that*

$$\mathbf{P}\left[\left|\hat{k}(x, x') - k(x, x')\right| \geq \sqrt{\frac{\log(2/\delta)}{d}}\alpha\right] \leq 2\delta$$

for all $\delta > 0$ where $\alpha = \frac{2\|x - x'\|}{\sigma}\sqrt{\log(2d/\delta)}$.

Theorem 6 demonstrates almost sub-Gaussian convergence Fastfood kernel for a fixed pair of points x, x'. A standard ϵ-net argument then shows uniform convergence over any compact set of $\mathbb{R}^d$ with bounded diameter (Rahimi & Recht, 2007)[Claim 1]. Also, the small error of the approximate kernel does not significantly perturb the solution returned by wide range of learning algorithms (Rahimi & Recht, 2007)[Appendix B] or affect their generalization error.

We refer the interested reader to Appendix A.2. in the supplementary material for the proof of the theorem. Our key tool is concentration of Lipschitz continuous functions under the Gaussian measure (Ledoux, 1996). We ensure that Fastfood construct has a small Lipschitz constant using Lemma 7.

Lemma 7 *(Ailon & Chazelle, 2009) Let $x \in \mathbb{R}^d$ and $t > 0$. Let $H \in \mathbb{R}^{d \times d}$ and $B \in \mathbb{R}^{d \times d}$ denote the Hadamard and the binary random diagonal matrices in our construct. Then for any $\delta > 0$ we have that $\mathbf{P}\left[\left\|d^{-\frac{1}{2}}HBx\right\|_\infty \geq \|x\|_2 \sqrt{\log\left(\frac{2d}{\delta}\right)\frac{2}{d}}\right] \leq \delta$.*

3.4. Changing the Spectrum

Changing the kernel from a Gaussian RBF to any other radial basis function kernel is straightforward. After all, $HG\Pi HB$ provides almost spherically uniformly distributed random vectors with common length. Rescaling each direction of projection separately costs only $O(n)$ space and computation. Consequently we are free to choose different coefficients S_{ii} rather than (9). Instead, we may use

$$S_{ii} \sim c^{-1} r^{d-1} A_{d-1}^{-1} \lambda(r).$$

Here c is a normalization constant and $\lambda(r)$ is the radial part of the spectral density function of the regularization operator associated with the kernel.

A key advantage over a conventional kernel approach is that we are not constrained by the requirement that the spectral distributions (5) be analytically computable. Even better, the spectra only need to be computable by some procedure (rather than have a closed-form representation).

For concreteness consider the Matern kernel. Its spectral properties are discussed, e.g. in (Schölkopf & Smola, 2002). In a nutshell, given data in $\mathbb{R}^d$ denote by $\nu := \frac{d}{2}$ a dimension calibration and let $t \in \mathbb{N}$ be a fixed parameter, which is usually set experimentally. Moreover, denote by $J_\nu(r)$ the Bessel function of the first kind of order ν. Then the kernel given by

$$k(x, x') := \|x - x'\|^{-t\nu} J_\nu^t(\|x - x'\|) \text{ for } t \in \mathbb{N}$$

has as its associated Fourier transform

$$\mathcal{F}k(\omega) = \bigotimes_{i=1}^{t} \chi_{S_d}[\omega].$$

Here χ_{S_d} is the characteristic function on the unit ball in $\mathbb{R}^d$ and $\bigotimes$ denotes convolution. In words, the Fourier transform of k is the t-fold convolution of χ_{S_d}. As convolutions of distributions arise from adding independent random variables this yields a simple algorithm for computing the Matern kernel:

for each S_{ii} **do**
  Draw t iid samples ξ_i uniformly from S_d.
  Use $S_{ii} = \left\|\sum_{i=1}^t \xi_i\right\|$ as scale.
end for

While this may appear costly, it only needs to be carried out once at initialization time. After that we can

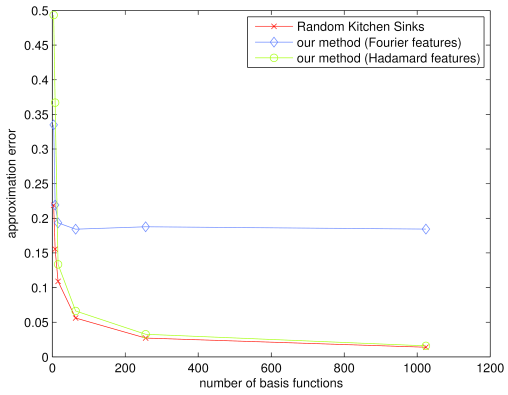

Figure 1. Kernel approximation errors of different methods with respect to number of basis functions n.

store the coefficients S_{ii}. Also note that this addresses a rather surprising problem with the Gaussian RBF kernel — in high dimensional spaces draws from a Gaussian are strongly concentrated on the surface of a sphere. That is, we only probe the estimation problem with a fixed characteristic length. The Matern kernel, on the other hand, spreads its capacity over a much larger range of frequencies.

4. Experiments

In the following we assess the performance of Random Kitchen Sinks and Fastfood. The results show that Fastfood performs as well as Random Kitchen Sinks in terms of accuracy. Fastfood, however, is orders of magnitude faster and exhibits a significantly lower memory footprint. For simplicity, we focus on penalized least squares regression since in this case we are able to compute exact solutions and are independent of any other optimization algorithms. We also benchmark Fastfood on CIFAR-10 (Krizhevsky, 2009) and observe that it achieves state-of-the-art accuracy. This advocates for the use of non-linear expansions even when d is large.

4.1. Approximation quality

We begin by investigating how well our features can approximate the exact kernel computation as n increases. For that purpose, we uniformly sample 4000 vectors from $[0, 1]^{10}$. We compare the exact kernel values to Random Kitchen Sinks and Fastfood.

The results are shown in Figure 1. We used the absolute difference between the exact kernel and the approximation to quantify the error (the relative difference also exhibits similar behavior and is thus not shown due to space constraints). The results are presented as averages, averaging over 4000 samples. As can be seen, as n increases, both Random Kitchen Sinks and Fastfood converge quickly to the exact kernel values. Their performance is indistinguishable, as expected from the construction of the algorithm.

Note, though, that fidelity in approximating $k(x, x')$ does not imply generalization performance (unless the bounds are very tight). To assess this, we carried out experiments on all *regression* datasets from the UCI repository (Frank & Asuncion, 2010) that are not too tiny, i.e., that contained at least $4,000$ instances.

We investigate estimation accuracy via Gaussian process regression using approximated kernel computation methods and we compare this to exact kernel computation whenever the latter is feasible. For completeness, we compare the following methods:

Exact RBF uses the exact RBF kernel. This is possible on all but the largest datasets where the kernel matrix does not fit into memory.

Nystrom uses the Nystrom approximation of the kernel matrix (Williams & Seeger, 2001). These methods have received recent interest due to the improved approximation guarantees of (Jin et al., 2011) which indicate that approximation rates faster than $O(n^{-\frac{1}{2}})$ are achievable. Hence, theoretically, the Nystrom method could have a significant accuracy advantage over Random Kitchen Sinks and Fastfood when using the same number of basis functions, albeit at exponentially higher cost (d vs. $\log d$) per function. We set $n = 2,048$.

Random Kitchen Sinks uses the the Gaussian random projection matrices described by (Rahimi & Recht, 2007). We use $n = 2,048$ basis functions.

Fastfood ("Hadamard features") uses the random matrix given by $SHG\Pi HB$, again with $n = 2,048$ dimensions.

FFT Fastfood ("Fourier features") uses a variant of the above construction. Instead of combining two Hadamard matrices, a permutation and Gaussian scaling, we use a permutation in conjunction with a Fourier Transform matrix F: the random matrix given by $V = \Pi FB$. The motivation is the Subsampled Random Fourier Transform (Tropp, 2011): by picking a random subset of columns from a (unitary) Fourier matrix, we end up with vectors that are almost spatially isotropic, albeit with slightly more dispersed lengths than in Fastfood. We use this heuristic for comparison purposes.

The results of the comparison are given in Table 2. As can be seen, there is virtually no difference between the exact kernel, the Nystrom approximation, Random Kitchen Sinks and Fastfood. Somewhat surprisingly the Fourier features work very well. This indicates that the concentration of measure effects impacting Gaussian RBF kernels may actually be counterproductive at their extreme.

In Figure 2, we show regression performances as a function of number of basis functions n on the CPU dataset and demonstrates that it is necessary to have a large n in order to learn highly nonlinear functions. Interestingly, although Fourier features do not seem

Table 2. Test set RMSE of different kernel computation methods. We can see Fastfood methods perform comparably with Exact RBF, Nystrom or Random Kitchen Sinks. m and d are the size of the training set the dimension of the input.

Dataset	m	d	Exact RBF	Nystrom RBF	Random Kitchen Sinks (RBF)	Fastfood FFT	Fastfood RBF	Exact Matern	Fastfood Matern
Insurance Company	5,822	85	0.231	0.232	0.266	0.266	0.264	0.234	0.235
Wine Quality	4,080	11	0.819	0.797	0.740	0.721	0.740	0.753	0.720
Parkinson Telemonitor	4,700	21	0.059	0.058	0.054	0.052	0.054	0.053	0.052
CPU	6,554	21	7.271	6.758	7.103	4.544	7.366	4.345	4.211
Location of CT slices (axial)	42,800	384	n.a.	60.683	49.491	58.425	43.858	n.a.	14.868
KEGG Metabolic Network	51,686	27	n.a.	17.872	17.837	17.826	17.818	n.a.	17.846
Year Prediction MSD	463,715	90	n.a.	0.113	0.123	0.106	0.115	n.a.	0.116
Forest	522,910	54	n.a.	0.837	0.840	0.838	0.840	n.a.	0.976

Table 1. Runtime, speed and memory improvements of Fastfood relative to Random Kitchen Sinks

d	n	Fastfood	RKS	Speedup	RAM
1,024	16,384	0.00058s	0.0139s	24x	256x
4,096	32,768	0.00136s	0.1224s	90x	1024x
8,192	65,536	0.00268s	0.5360s	200x	2048x

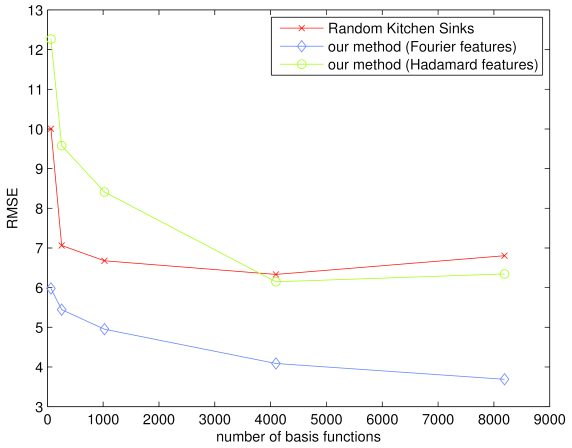

Figure 2. Test RMSE on CPU dataset with respect to the number of basis functions. As number of basis functions increases, the quality of regression generally improves.

to approximate the Gaussian RBF kernel, they perform well compared to other variants and improve as n increases. This suggests that learning the kernel by direct spectral adjustment might be a useful application of our proposed method.

4.2. Speed of kernel computations

In the previous experiments, we observe that Fastfood is on par with exact kernel computation, the Nystrom method, and Random Kitchen Sinks. The key point, however, is to establish whether the algorithm offers computational savings.

For this purpose we compare Random Kitchen Sinks using Eigen[5] and our method using Spiral[6]. Both are highly optimized numerical linear algebra libraries in C++. We are interested in the time it takes to go from raw features of a vector with dimension d to the label prediction of that vector. On a small problem with $d = 1,024$ and $n = 16,384$, performing prediction with Random Kitchen Sinks takes 0.07 seconds. Our method is around 24x faster, taking only 0.003 seconds to compute the label for one input vector. The speed gain is even more significant for larger problems, as is evident in Table 1. This confirms experimentally the $O(n \log d)$ vs. $O(nd)$ runtime and $O(n)$ vs. $O(nd)$ storage of Fastfood relative to Random Kitchen Sinks.

[5] http://eigen.tuxfamily.org/index.php?title=Main_Page

[6] http://spiral.net

4.3. Random features for CIFAR-10

To understand the importance of nonlinear feature expansions for a practical application, we benchmarked Fastfood, Random Kitchen Sinks on the CIFAR-10 dataset (Krizhevsky, 2009) which has 50,000 training images and 10,000 test images. Each image has 32x32 pixels and 3 channels ($d = 3072$). In our experiments, linear SVMs achieve 42.3% accuracy on the test set. Non-linear expansions improve the classification accuracy significantly. In particular, Fastfood FFT ("Fourier features") achieve 63.1% while Fastfood ("Hadamard features") and Random Kitchen Sinks achieve 62.4% with an expansion of $n = 16,384$. These are also best known classification accuracies using permutation-invariant representations on this dataset. In terms of speed, Random Kitchen Sinks is 5x slower (in total training time) and 20x slower (in predicting a label given an image) compared to Fastfood and Fastfood FFT. This demonstrates that non-linear expansions are needed even when the raw data is high-dimensional, and that Fastfood is more practical for such problems.

In particular, in many cases, linear function classes are used because they provide fast training time, and especially test time, but not because they offer better accuracy. The results on CIFAR-10 demonstrate that Fastfood can overcome this obstacle.

Summary We demonstrated that it is possible to compute n nonlinear basis functions in $O(n \log d)$ time, a significant speedup over the best competitive algorithms. This means that kernel methods become more practical for problems that have large datasets and/or require real-time prediction. In fact, Fastfood can be used to run on cellphones because not only it is fast, but it also requires only a small amount of storage.

Acknowledgments We thank John Langford and Ravi Kumar for fruitful discussions.

References

Ailon, N. and Chazelle, B. The fast Johnson–Lindenstrauss transform and approximate nearest neighbors. *SICOMP*, 2009.

Aizerman, M. A., Braverman, A. M., and Rozonoér, L. I. Theoretical foundations of the potential function method in pattern recognition learning. *Autom. Remote Control*, 25:821–837, 1964.

Aronszajn, N. La théorie générale des noyaux réproduisants et ses applications. *Proc. Cambridge Philos. Soc.*, 39:133–153, 1944.

Boser, B., Guyon, I., and Vapnik, V. A training algorithm for optimal margin classifiers. *COLT* 1992.

Burges, C. J. C. Simplified support vector decision rules. *ICML*, 1996

Cortes, C. and Vapnik, V. Support vector networks. *Machine Learning*, 20(3):273–297, 1995.

Dasgupta, A., Kumar, R., and Sarlós, T. Fast locality-sensitive hashing. *SIGKDD*, pp. 1073–1081, 2011.

Fine, S. and Scheinberg, K. Efficient SVM training using low-rank kernel representations. *JMLR*, 2001.

Frank, A. and Asuncion, A. UCI machine learning repository. http://archive.ics.uci.edu/ml.

Girosi, F. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10(6):1455–1480, 1998.

Girosi, F., Jones, M., and Poggio, T. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

Gray, A. G. and Moore, A. W. Rapid evaluation of multiple density models. *AISTATS*, 2003.

Jin, R., Yang, T., Mahdavi, M., Li, Y.F., and Zhou, Z.H. Improved bound for the Nystrom's method and its application to kernel classification, 2011. URL http://arxiv.org/abs/1111.2262.

Kimeldorf, G. S. and Wahba, G. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *Annals of Mathematical Statistics*, 41:495–502, 1970.

Kreyszig, E. *Introductory Functional Analysis with Applications*. Wiley, 1989.

Krizhevsky, A. Learning multiple layers of features from tiny images. TR, U Toronto, 2009.

Ledoux, M. Isoperimetry and Gaussian analysis. Lectures on probability theory and statistics, 1996.

Lee, D. and Gray, A. G. Fast high-dimensional kernel summations using the Monte Carlo multipole method. *NIPS*, 2009.

MacKay, D. J. C. *Information Theory, Inference, and Learning Algorithms*. Cambridge 2003.

Mercer, J. Functions of positive and negative type and their connection with the theory of integral equations. *Royal Society London*, A 209:415–446, 1909.

Micchelli, C. A. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constr. Approximation*, 2:11–22, 1986.

Neal, R. Priors for infinite networks. CRG-TR-94-1, U Toronto, 1994.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. *NIPS 20*, 2007.

Rahimi, A. and Recht, B. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. *NIPS 21*, 2008.

Schölkopf, B., Smola, A. J., and Müller, K.-R. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10:1299–1319, 1998.

Schölkopf, Bernhard and Smola, A. J. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

Smola, A. J. and Schölkopf, B. Sparse greedy matrix approximation for machine learning. *ICML*, 2000.

Smola, A. J., Schölkopf, B., and Müller, K.-R. The Connection between Regularization Operators and Support Vector Kernels. *Neur. Networks*, 1998

Steinwart, Ingo and Christmann, Andreas. *Support Vector Machines*. Springer, 2008.

Taskar, B., Guestrin, C., and Koller, D. Max-margin Markov networks. *NIPS 16*, 2004.

Tropp, J. A. Improved analysis of the subsampled randomized Hadamard transform Adv. Adapt. Data Anal., 2011.

Vapnik, V., Golowich, S., and Smola, A. Support vector method for function approximation, regression estimation, and signal processing. *NIPS 9*, 1997.

Wahba, G. *Spline Models for Observational Data*, *CBMS-NSF*, vol. 59, SIAM, Philadelphia, 1990.

Williams, C. K. I. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In Jordan, M. I. (ed.), *Learning and Inference in Graphical Models*, Kluwer, 1998.

Williams, C. K. I. and Seeger, M. Using the Nystrom method to speed up kernel machines. *NIPS 13*, 2001.