Abstract

Kernel methods offer the flexibility to learn complex relationships in modern, large data sets while enjoying strong theoretical guarantees on quality. Unfortunately, these methods typically require cubic running time in the data set size, a prohibitive cost in the large-data setting. Random feature maps (RFMs) and the Nyström method both consider low-rank approximations to the kernel matrix as a potential solution. But, in order to achieve desirable theoretical guarantees, the former may require a prohibitively large number of features $J$, and the latter may be prohibitively expensive for high-dimensional problems. We propose to combine the simplicity and generality of RFMs with a data-dependent feature selection scheme to achieve desirable theoretical approximation properties of Nyström with just $O(\log J)$ features. Our key insight is to begin with a large set of random features, then reduce them to a small number of weighted features in a data-dependent, computationally efficient way, while preserving the statistical guarantees of using the original large set of features. We demonstrate the efficacy of our method with theory and experiments—including on a data set with over 50 million observations. In particular, we show that our method achieves small kernel matrix approximation error and better test set accuracy with provably fewer random features than state-of-the-art methods.

1 Introduction

Kernel methods are essential to the machine learning and statistics toolkit because of their modeling flexibility, ease-of-use, and widespread applicability to problems including regression, classification, clustering, dimensionality reduction, and one and two-sample testing [Holmann et al., 2008, Scholkopf and Smola, 2001, Chwialkowski et al., 2016, Gretton et al., 2012]. In addition to good empirical performance, kernel-based methods come equipped with strong statistical and learning-theoretic guarantees [Vapnik, 1998, Mendelson, 2003, Balcan et al., 2008, Boser et al., 1992, Vapnik et al., 1997, Sriperumbudur et al., 2010]. Because kernel methods are nonparametric, they are particularly attractive for large-scale problems, where they make it possible to learn complex, highly non-linear structure from data. Unfortunately, their time and memory costs scale poorly with data size. Given $N$ observations, storing the kernel matrix $K$ requires $O(N^2)$ space. Using $K$ for learning typically requires $O(N^3)$ time, as this often entails inverting $K$ or computing its singular value decomposition.

To overcome poor scaling in $N$, researchers have devised various approximations to exact kernel methods. A widely-applicable and commonly used tactic is to replace $K$ with a rank-$J$ approximation, which reduces storage requirements to $O(NJ)$ and computational complexity of inversion or singular value decomposition to $O(NJ^2)$ [Halko et al., 2011]. Thus, if $J$ can be chosen to be constant or slowly increasing in $N$, only (near-)linear time and space is required in the dataset size. Two popular approaches to constructing low-rank approximations are random feature maps (RFMs) [Kar and Karnick, 2012, Pennington et al., 2015, Daniely et al., 2017, Samo and Roberts, 2013]—particularly random Fourier features (RFFs) [Rahimi and Recht, 2007]—and Nyström-type approximations [Drineas and Mahoney, 2005]. The Nyström method is based on using $J$ randomly sampled columns from $K$, and thus is data-dependent. The data-dependent nature of Nyström methods can provide statistical guarantees even when $J \ll N$, but these results either apply only
to kernel ridge regression (El Alaoui and Mahoney, 2015; Yang et al., 2017; Rudi et al., 2015) or require burdensome recursive sampling schemes (Musco and Musco, 2017; Lim et al., 2018). Random features, on the other hand, are simple to implement and use $J$ random features that are data-independent. For problems with both large $N$ and number of covariates $p$, an extension of random features called Fast Food RFM has been successfully applied at a fraction of the computational time required by Nyström-type approximations, which are exponentially more costly in terms of $p$ (Le et al., 2013). The price for this simplicity and data-independence is that a large number of random features is often needed to approximate the kernel matrix well (Honorio and Li, 2017; Kar and Karnick, 2012; Rahimi and Recht, 2007; Yang et al., 2012; Huang et al., 2014).

The question naturally arises, then, as to whether we can combine the simplicity of random features and the ability to scale to large-$p$ problems with the appealing approximation and statistical properties of Nyström-type approaches. We provide one possible solution by making random features data-dependent, and we show promising theoretical and empirical results. Our key insight is to begin with a large set of random features, then reduce them to a small set of weighted features in a data-dependent, computationally efficient way, while preserving the statistical guarantees of using the original large set. We frame the task of finding this small set of features as an optimization problem, which we solve using ideas from the coreset literature (Campbell and Broderick, 2019; 2018). Using greedy optimization schemes such as the Frank–Wolfe algorithm, we show that a large set of $J_+$ random features can be compressed to an exponentially smaller set of just $O(\log J_+)$ features while still achieving the same statistical guarantees as using all $J_+$ features. We demonstrate that our method achieves superior performance to existing approaches on a range of real datasets—including one with over 50 million observations—in terms of kernel matrix approximation and classification accuracy.

## 2 Preliminaries and related work

Suppose we observe data $\{(x_n, y_n)\}_{n=1}^N$ with predictors $x_n \in \mathbb{R}^p$ and responses $y_n \in \mathbb{R}$. In a supervised learning task, we aim to find a model $f : \mathbb{R}^p \to \mathbb{R}$ among a set of candidates $\mathcal{F}$ that predicts the response well for new predictors. Modern data sets of interest often reach $N$ in the tens of millions or higher, allowing analysts to learn particularly complex relationships in data. Nonparametric kernel methods (Schölkopf and Smola, 2001) offer a flexible option in this setting: by taking $\mathcal{F}$ to be a reproducing kernel Hilbert space with positive-definite kernel $k : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$, they enable learning more nuanced details of the model $f$ as more data are obtained. As a result, kernel methods are widespread not just in regression and classification but also in dimensionality reduction, conditional independence testing, one and two-sample testing, and more (Schölkopf et al., 1997; Zhang et al., 2011; Gretton et al., 2008; 2012; Chwialkowski et al., 2016).

The problem, however, is that kernel methods become computationally intractable for large $N$. We consider kernel ridge regression as a prototypical example (Sauder et al., 1998). Let $K \in \mathbb{R}^{N \times N}$ be the kernel matrix consisting of entries $K_{nm} := k(x_n, x_m)$. Collect the responses into the vector $y \in \mathbb{R}^N$. Then kernel ridge regression requires solving

$$\min_{\alpha \in \mathbb{R}^N} -\frac{1}{2} \alpha^T (K + \lambda I) \alpha + \alpha^T y,$$

where $\lambda > 0$ is a regularization parameter. Computing and storing $K$ alone has $O(N^2)$ complexity, while computing the solution $\alpha^* = (K + \lambda I)^{-1} y$ further requires solving a linear system, with cost $O(N^3)$. Many other kernel methods have $O(N^3)$ dependence; see Table 1.

To make kernel methods tractable on large datasets, a common practice is to replace the kernel matrix $K$ with an approximate low-rank factorization $\hat{K} := ZZ^T \approx K$, where $Z \in \mathbb{R}^{N \times J}$ and $J \ll N$. This factorization can be viewed as replacing the kernel function $k$ with a finite-dimensional inner product $k(x_n, x_m) \approx z(x_n)^T z(x_m)$ between features generated by a feature map $z : \mathbb{R}^p \to \mathbb{R}^J$. Using this type of approximation significantly reduces downstream training time, as shown in the second column of Table 1. Previous results show that as long as $ZZ^T$ is close to $K$ in the Frobenius norm, the optimal model $f$ using $\hat{K}$ is uniformly close to the one using $K$ (Cortes et al., 2010); see the rightmost column of Table 1.

However, finding a good feature map is a nontrivial task. One popular method, known as random Fourier features (RFF) (Rahimi and Recht, 2007), is based on Bochner’s Theorem:

**Theorem 2.1** (Rudin, 1994: p. 19). A continuous, stationary kernel $k(x, y) = \phi(x - y)$ for $x, y \in \mathbb{R}^p$ is positive definite with $\phi(0) = 1$ if and only if there exists a probability measure $Q$ such that

$$\phi(x - y) = \int_{\mathbb{R}^p} e^{i\omega^T(x-y)} dQ(\omega) = \mathbb{E}_Q[\psi_\omega(x) \psi_\omega(y)^*], \quad \psi_\omega(x) := e^{i\omega^T x}. \quad (1)$$

Theorem 2.1 implies that $z_{\text{complex}}(x) := (1/\sqrt{J})[\psi_{\omega_1}(x), \ldots, \psi_{\omega_J}(x)]^T$, where $\omega_i \overset{i.i.d.}{\sim} Q$, provides a Monte-Carlo approximation of the true kernel function. As noted by Rahimi and Recht (2008), the real-valued feature map

---

**Note:** The content is a significant excerpt from a research paper discussing the use of random features, particularly in the context of kernel ridge regression, with a focus on the computational challenges and the effectiveness of approximating K in large datasets. The paper highlights the use of random Fourier features (RFF) as a method to make kernel methods more tractable for large-scale problems.
Table 1: A comparison of training time for PCA, SVM, and ridge regression using the exact kernel matrix $K$ versus a low-rank approximation $\hat{K} = ZZ^T$, where $Z$ has $J$ columns. Exact training requires either inverting or computing the SVD of the true kernel matrix $K$ at a cost of $O(N^3)$ time, as shown in the first column. The second column refers to training the methods using a low-rank factorization $Z$. For ridge regression and PCA, the low-rank training cost reflects the time to compute and invert the feature covariance matrix $Z^T Z$. For SVM, the time refers to fitting a linear SVM on $Z$ using dual-coordinate descent with optimization tolerance $\rho$ (Hsieh et al., 2008). The third column quantifies the uniform error between the function fit using $K$ and the function fit using $Z$. For specific details of how the bounds were derived, see Appendix D.

<table>
<thead>
<tr>
<th>Method</th>
<th>Exact Training Cost</th>
<th>Low-Rank Training Cost</th>
<th>Approximation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>$O(N^3)$</td>
<td>$\Theta(NJ^2)$</td>
<td>$O\left(1 - \frac{1}{N}|\hat{K} - K|_F\right)$</td>
</tr>
<tr>
<td>SVM</td>
<td>$O(N^3)$</td>
<td>$\Theta(NJ \log \frac{1}{\epsilon})$</td>
<td>$O\left(|\hat{K} - K|_F\right)$</td>
</tr>
<tr>
<td>Ridge Regression</td>
<td>$O(N^3)$</td>
<td>$\Theta(NJ^2)$</td>
<td>$O\left(\frac{1}{N}|\hat{K} - K|_F\right)$</td>
</tr>
</tbody>
</table>

$z(x) := \frac{(1/\sqrt{J})[\cos(\omega_1^Tx + b_1), \ldots, \cos(\omega_J^Tx + b_J)]^T}{b_1 \sim [0, 2\pi]}$ also yields an unbiased estimator of the kernel function; we use this feature map in what follows unless otherwise stated. The resulting $N \times J$ feature matrix $Z$ yields estimates of the true kernel function with standard Monte-Carlo error rates of $O(1/\sqrt{J})$ uniformly on compact sets. The RRF methodology also applies quite broadly. There are well-known techniques for obtaining samples from $Q$ for a variety of popular kernels such as the squared exponential, Laplace, and Cauchy, as well as extensions to more general random feature maps (RFMs), which apply to many types of non-stationary kernels.

The major drawback of RFMs is the $O(NJp)$ time and $O(NJ)$ memory costs associated with generating the feature matrix $Z$. Although these are linear in $N$ as desired, recent empirical evidence suggests that $J$ needs to be quite large to provide competitive performance with other data analysis techniques. Recent work addressing this drawback has broadly involved two approaches: variance reduction and feature compression. Variance reduction techniques involve modifying the standard Monte-Carlo estimate of $k$, e.g. with control variates, quasi-Monte-Carlo techniques, or importance sampling. These approaches either depend poorly on the data dimension $p$ (in terms of statistical generalization error), or, for a fixed approximation error, reduce the number of features $J$ compared to RFM only by a constant. Feature compression techniques, on the other hand, involve two steps: (1) “up-projection,” in which the basic RFM methodology generates a large number $J_+ \gg J$ of features—followed by (2) “compression,” in which those features are used to find a smaller number $J_\ell$ of features while ideally retaining the kernel approximation error of the original $J_+$ features. Compact random feature maps (Hamid et al., 2014) represent an instance of this technique in which compression is achieved using the Johnson–Lindenstrauss (JL) algorithm (Johnson et al., 1986). However, not only is the generation of $J_+$ features prohibitively expensive for large datasets, JL compression is data-independent and leads to only a constant reduction in $J_\ell$ as we show in Appendix C (see summary in Table 2).

### 3 Random feature compression via coresets

In this section, we present an algorithm for approximating a kernel matrix $K \in \mathbb{R}^{N \times N}$ with a low-rank approximation $\hat{K} = ZZ^T$ obtained using a novel feature compression technique. In the up-projection step we generate $J_+$ random features, but only compute their values for a small, randomly-selected subset of $S \ll N^2$ datapoint pairs. In the compression step, we select a sparse, weighted subset of $J_\ell$ of the original $J_+$ features in a sequential greedy fashion. We use the feature values on the size-$S$ subset of all possible data pairs to decide, at each step, which feature to include and its weight. Once this process is complete, we compute the resulting weighted subset of $J_\ell$ features on the whole dataset. We use this low-rank approximation of the kernel in our original learning problem. Since we use a sparse weighted feature subset for compression—as opposed to a general linear combination as in previous work—we do not need to compute all $J_+$ features for
Data-dependent compression of random features

3.1 Algorithm derivation

Let $Z_+ \in \mathbb{R}^{N \times J_+}$, $J_+ > J$, be a fixed up-projection feature matrix generated by RFM. Our goal is to use $Z_+$ to find a compressed low-rank approximation $K = ZZ^T \approx K$, $Z \in \mathbb{R}^{N \times J}$. Our approach is motivated by the fact that spectral 2-norm bounds on $K$—provided uniform bounds on the difference between learned models using $K$ and $K$ (Cortes et al., 2010), as well as the fact that the Frobenius norm bounds the 2-norm. So we aim to find a $Z$ that minimizes the Frobenius norm error $\|K - ZZ^T\|_F$. By the triangle inequality,

$$\|K - ZZ^T\|_F \leq \|K - Z_+ Z_+^T\|_F + \|Z_+ Z_+^T - ZZ^T\|_F,$$

so constructing a good feature compression down to $J$ features amounts to picking $Z$ such that $Z_+ Z_+^T \approx ZZ^T$ in Frobenius norm. Let $Z_{+j} \in \mathbb{R}^N$ denote the $j$th column of $Z_+$. Then we would ideally like to solve the optimization problem

$$\arg\min_{w \in \mathbb{R}^{J_+}} \frac{1}{N^2} \|Z_+ Z_+^T - Z(w) Z(w)^T\|_F^2,$$

subject to $\|w\|_0 \leq J. (3)$

This problem is intractable to solve exactly for two main reasons. First, computing the objective function requires computing $Z_+$, which itself takes $\Omega(NJ_+p)$ time. But it is not uncommon for all three of $N$, $J_+$, and $p$ to be large, making this computation expensive. Second, the cardinality, or “0-norm,” constraint on $w$ yields a difficult combinatorial optimization. In order to address these issues, first note that

$$\frac{1}{N^2} \|Z_+ Z_+^T - Z(w) Z(w)^T\|_F^2 = \mathbb{E}_{i,j \sim \mathbb{P}} \left[ (z_{i+}^T z_{j+} - z_i(w)^T z_j(w))^2 \right],$$

where $\mathbb{P}$ is the uniform distribution on the integers $\{1, \ldots, N\}$, and $z_{i+}, z_i(w)$ are the $i$th rows of $Z_+, Z(w)$, respectively. Therefore, we can generate a Monte-Carlo estimate of the optimization objective by sampling $S$ pairs $i_s, j_s \sim \mathbb{P}$:

$$\frac{1}{N^2} \|Z_+ Z_+^T - Z(w) Z(w)^T\|_F^2 \approx \frac{1}{S} \sum_{s=1}^S (z_{i_s+}^T z_{j_s+} - z_{i_s}(w)^T z_{j_s}(w))^2$$

$$= (1 - w)^T R R^T (1 - w) \text{ s.t.}$$

$$R := \begin{bmatrix} z_{i_1+} \circ z_{j_1+}, & \cdots, & z_{i_S+} \circ z_{j_S+} \end{bmatrix} \in \mathbb{R}^{J_+ \times S},$$

where $\circ$ indicates a component-wise product. Denoting the $j$th row of $R$ by $R_j \in \mathbb{R}^S$ and the sum of the rows by $r = \sum_{j=1}^{J_+} R_j$, we can rewrite the Monte Carlo approximation of the original optimization problem in Eq. (3) as

$$\arg\min_{w \in \mathbb{R}^{J_+}} \|r - r(w)\|_2^2$$

subject to $\|w\|_0 \leq J, (5)$

where $r(w) := \sum_{j=1}^{J_+} w_j R_j$. Note that the $s$th component $r_s = z_{i_s+} \circ z_{j_s+}$ of $r$ is the Monte-Carlo approximation of $k(x_i, x_j)$ using all $J_+$ features, while $r(w)_s = (w \circ z_{i_s+})^T (w \circ z_{j_s+})$ is the sparse Monte-Carlo approximation using weights $w \in \mathbb{R}^{J_+}$. In other words, the difference between the full optimization in Eq. (3) and the reformulated optimization in Eq. (5) is that the former attempts to find a sparse, weighted set of features that approximates the full $J_+$-dimensional feature inner products for all data pairs, while the latter attempts to do so only for the subset of pairs $i_s, j_s$, $s \in \{1, \ldots, S\}$. Since a kernel matrix is symmetric and $k(x_n, x_n) = 1$ for any datapoint $x_n$, we only need to sample $(i, j)$ above the diagonal of the $N \times N$ matrix (see Algorithm 1).

The reformulated optimization problem in Eq. (5)—i.e., approximating the sum of $r$ of a collection $(R_j)_{j=1}^{J_+}$ of vectors in $\mathbb{R}^S$ with a sparse weighted linear combination—is precisely the Hilbert coreset construction problem studied in previous work (Campbell and Broderick, 2019, 2018). There exist a number of efficient algorithms to solve this problem approximately; in particular, the Frank–Wolfe-based method of Campbell and Broderick (2019) and “greedy iterative geodesic ascent” (GIGA) (Campbell and Broderick, 2018) both provide an exponentially decreasing objective value as a function of the compressed number of features $J$. Note that it is also possible to apply other more general-purpose methods for cardinality-constrained convex optimization (Chen et al., 1998; Candès and Tao, 2007; Tibshirani, 1994), but these techniques are often too computationally expensive in the large-dataset setting. Our overall algorithm for feature compression is shown in Algorithm 1.
Algorithm 1 Random Feature Maps Compression (RFM-FW / RFM-GIGA)

**Input:** Data \((x_n)_{n=1}^N \in \mathbb{R}^p\), RFM distribution \(Q\), number of starting random features \(J_+\), number of compressed features \(J\), number of data pairs \(S\)

**Output:** Weights \(w \in \mathbb{R}^d\) with at most \(J\) non-zero entries

1. \((i_s,j_s)_{s=1}^S \overset{i.i.d.}{\sim} \text{Unif} (\{(i,j) : i < j, 2 \leq j \leq N\})
2. Sample \((\omega_j)_{j=1}^J \overset{i.i.d.}{\sim} Q\)
3. Sample \(b_j \overset{\text{unif}}{\sim} [0, 2\pi], 1 \leq j \leq J_+\)
4. for \(s = 1 : S\) do
5. Compute \(z_{j+i} \leftarrow (1/\sqrt{J_+}) \cos(\omega_j^T x_i + b_j), \ldots, \cos(\omega_J^T x_i + b_J))^T\); same for \(z_{j+i}\)
6. Compute \(R \leftarrow \left[z_{1+i}, \ldots, z_{J+i}, \ldots, z_{J+i} \right]\)
7. \(R_j \leftarrow \text{row}\ j\ of\ \ R; \ r_j \leftarrow \sum_{j=1}^{J_+} R_j\)
8. \(w \leftarrow \text{solution to Eq. (5) with FW}\) Campbell and Broderick [2019] or GIGA Campbell and Broderick [2018]
9. \(Z(w) = \sqrt{w} T Z + \cdots + \sqrt{w} T Z + J_+\)
10. return \(Z(w)\)

3.2 Theoretical results

In order to employ Algorithm 1 we must choose the number \(S\) of data pairs, the up-projected feature dimension \(J_+\), and compressed feature dimension \(J\). Selecting these three quantities involves a tradeoff between the computational cost of using Algorithm 1 and the resulting low-rank kernel approximation Frobenius error, but it is not immediately clear how to perform that tradeoff. Theorem 3.2 and Corollary 3.3 provide a remarkable resolution to this issue: roughly, if we fix \(J_+\) such that the basic random feature maps provide kernel approximation error \(\epsilon > 0\) with high probability, then choosing \(S = \Omega(J_+^2 \log J_+^2)\) and \(J = \Omega(\log J_+)\) suffices to guarantee that the compressed feature kernel approximation error is also \(O(\epsilon)\) with high probability. In contrast, previous feature compression methods required \(J = \Omega(J_+)\) to achieve the same result; see Table 2. Note that Theorem 3.2 assumes that the compression step in Algorithm 1 is completed using the Frank–Wolfe-based method from Campbell and Broderick [2019]. However, this choice was made solely to simplify the theory; as GIGA Campbell and Broderick [2018] provides stronger performance both theoretically and empirically, we expect a stronger result than Theorem 3.2 and Corollary 3.3 to hold when using GIGA. The proof of Theorem 3.2 is given in Appendix B and depends on the following assumptions.

**Assumption 3.1.** (a) The cardinality of the set of vectors \({x_i - x_j, x_i + x_j, 1 \leq i < j \leq N}\) is \(\frac{N(N-1)}{2}\), i.e., all vectors \(x_i - x_j, x_i + x_j, 1 \leq i < j \leq N\) are distinct.
(b) \(Q(\omega)\) for \(\omega \in \mathbb{R}^p\) has strictly positive density on all of \(\mathbb{R}^p\), where \(Q\) is the measure induced by the kernel \(k\); see Theorem 2.1

Assumption 3.1(a-b) are sufficient to guarantee that the compression coefficient \(\nu_{J_+}\) provided in Theorem 3.2 does not go to 1. If \(\nu_{J_+} \to 1\) as \(J_+ \to \infty\), the amount of compression could go to zero asymptotically. When the \(x_i\)'s contain continuous (noisy) measurements, Assumption 3.1(b) is satisfied by most kernels used in practice (e.g. radial basis function, Laplace kernel, etc.).

We obtain the exponential compression in Theorem 3.2 for the following reason: Frank-Wolfe and GIGA converge linearly when the minimizer of Eq. (6) belongs to the relative interior of the feasible set of solutions (Marguerite and Philip [1956]), which turns out to occur in our case. With linear convergence, we need to run only a logarithmic number of iterations (which upper bounds the sparsity of \(w\)) to approximate \(r(w)\) for a given level of approximation error. For fixed \(J_+\), Lemma A.5 from Campbell and Broderick [2019] immediately implies that the minimizer belongs to the relative interior. As \(J_+ \to \infty\) (that is, as we represent the kernel function exactly), we show that the minimizer asymptotically belongs to the relative interior, and we provide a lower bound on its distance to the boundary of the feasible set. This distance lower bound is key to the asymptotic worst-case bound on the compression coefficient given in Theorem 3.2 and Theorem 3.3.

**Theorem 3.2.** Fix \(\epsilon > 0\), \(\delta \in (0, 1)\), and \(J_+ \in \mathbb{N}\). Then there are constants \(\nu_{J_+} \in (0, 1)\), which depends only on \(J_+\), and \(0 < c_0 < \infty\), which depends only on \(\delta\), such that if

\[
J = \Omega \left( \frac{-\log J_+}{\log \nu_{J_+}} \right) \quad \text{and} \quad S = \Omega \left( \frac{c_0^2}{c^2} \left[ \log \frac{1}{\delta} \right]^4 \log J_+ \right),
\]

then with probability at least \(1 - \delta\), the output \(Z\) of Algorithm 1 satisfies

\[
\frac{1}{N^2} \|Z_+ Z^T - Z Z^T\|_F^2 \leq \epsilon.
\]

Furthermore, the compression coefficient is asymptotically bounded away from 1. That is,

\[
0 < \limsup \nu_{J_+} < 1.
\]

**Corollary 3.3.** In the setting of Theorem 3.2, if we let \(J_+ = \Omega(1/\epsilon \log 1/\epsilon)\), then

\[
\frac{1}{N^2} \|K - ZZ^T\|_F^2 = O(\epsilon).
\]
Table 2: A comparison of the computational cost of basic random feature maps (RFM), RFM with JL compression (RFM-JL), and RFM with our proposed compression using FW (RFM-FW) for $N$ datapoints and $J_+ = \frac{1}{\epsilon} \log^2 \frac{1}{\epsilon}$ up-projection features. The first column specifies the number of compressed features $J$ needed to retain the $O(\epsilon)$ high probability kernel approximation error guarantee of RFM. The second and third columns list the complexity for computing the compressed features and using them for PCA or ridge regression, respectively. Theoretically, the number of datapoint pairs $S$ should be set to $\Omega(J_+^2 (\log J_+)^2)$ in Algorithm 1 (see Theorem 3.2) but empirically we find in Section 4 that $S$ can be set much smaller. See Appendix C for derivations.

<table>
<thead>
<tr>
<th>Method</th>
<th># Compressed Features $J$</th>
<th>Cost of Computing $Z$</th>
<th>PCA/Ridge Reg. Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFM</td>
<td>$O(J_+)$</td>
<td>$O(NJ_+)$</td>
<td>$O(NJ_+^2)$</td>
</tr>
<tr>
<td>RFM-JL</td>
<td>$O(J_+)$</td>
<td>$O(NJ_+ \log J_+)$</td>
<td>$O(NJ_+^2)$</td>
</tr>
<tr>
<td>RFM-FW</td>
<td>$O(\log J_+)$</td>
<td>$O(SJ_+ \log J_+ + N \log J_+)$</td>
<td>$O(N(\log J_+)^2)$</td>
</tr>
</tbody>
</table>

Proof. Claim 1 of Rahimi and Recht (2007) implies that $\frac{1}{\epsilon} \left\| K - Z_+ Z_+^T \right\|_F = O(\epsilon)$ if we set $J_+ = \Omega(1/\epsilon \log \frac{1}{\epsilon})$. The result follows by combining Theorem 3.2 and Eq. (2).

Table 2 builds on the results of Theorem 3.2 and Corollary 3.3 to illustrate the benefit of our proposed feature compression technique in the settings of kernel principal component analysis (PCA) and ridge regression. Since random features and random features with JL compression both have $J = \Omega(J_+)$, the $O(NJ_+^2)$ cost of computing the feature covariance matrix $Z^T Z$ dominates when training PCA or ridge regression. In contrast, the dominant cost of random features with our proposed algorithm is the compression step; each iteration of Frank-Wolfe has cost $O(J_+ S)$, and we run it for $O(\log J_+)$ iterations.

While Corollary 3.3 says how large $S$ must be for a given $J_+$, it does not say how to pick $J_+$, or equivalently how to choose the level of precision $\epsilon$. As one would expect, the amount of precision needed depends on the downstream application. For example, recent theoretical work suggests that both kernel PCA and kernel ridge regression require $J_+$ to scale only sublinearly with the number of datapoints $N$ to achieve the same statistical guarantees as an exact kernel machine trained on all $N$ datapoints (Sriperumbudur and Sterge, 2017; Avron et al., 2017; Rudi and Rosasco, 2017). For kernel support vector machines (SVMs), on the other hand, Sutherland and Schneider (2015) suggest that $J_+$ needs to be larger than $N$. Such a choice of $J_+$ would make random features slower than training an exact kernel SVM. However, since Sutherland and Schneider (2015) do not provide a lower bound, it is still an open theoretical question how $J_+$ must scale with $N$ for kernel SVMs.

For $J_+$ even moderately large, setting $S = \Omega(J_+^2 (\log J_+)^2)$ to satisfy Theorem 3.2 will be prohibitively expensive. Fortunately, in practice, we find $S \ll J_+^2$ suffices to provide significant practical computational gains without adversely affecting approximation error; see the results in Section 4. We conjecture that we see this behavior since we expect even a small number of data pairs $S$ to be enough to guide feature compression in a data-dependent manner. We empirically verify this intuition in Fig. 4 of Section 4.

Finally, we provide an asymptotic upper bound for the compression coefficient $\nu_{J_+}$. We achieve greater compression when $\nu_{J_+} \downarrow 0$. Hence, the upper bound below shows the asymptotic worst-case rate of compression.

**Theorem 3.4.** Suppose all $\{(i, j) : 1 \leq i < j \leq N\}$ are sampled in Algorithm 1. Then,

$$0 \leq \limsup_{J_+ \to \infty} \nu_{J_+} < 1 - \left(1 - \frac{\|K\|_F}{c_Q}\right)^2 < 1,$$

where $K$ is the exact kernel matrix and

$$c_Q := \frac{1}{N} \mathbb{E}_{\omega \sim Q, b \sim \mathcal{N}(0, 2\pi)} \left\| u(\omega, b) \right\|_2,$$

$$u(\omega, b) := (\cos(\omega^T x_i + b) \cos(\omega^T x_j + b))_{i,j \in [N]}.$$

By Theorem 2.1, $\|K\|_F = \frac{1}{N} \mathbb{E}_{\omega, b} \|u(\omega, b)\|_2$, so $\|K\|_F \leq c_Q$ by Jensen’s inequality. In Appendix A we show this inequality holds strictly. Hence the term squared in Eq. (7) lies in $(0, 1]$. Recall $\|K\|_F^2 = \sum_{i=1}^N \lambda_i$, for $\lambda_i$ the eigenvalues of $K$. With these observations, Theorem 3.4 says that the asymptotic worst-case rate of compression improves if $K$’s eigenvalue sum is smaller.

As rough intuition: If the sum is small, then $K$ may be nearly low-rank and thus easier to approximate via a low-rank approximation. Since we subsample only $S$ of all pairs in Theorem 3.2 the upper bound in Theorem 3.4 does not necessarily apply. Nonetheless, for $S$ moderately large, this upper bound roughly characterizes the worst-case compression rate for Algorithm 1.

## 4 Experiments

In this section we provide an empirical comparison of basic random feature maps (RFM) (Rahimi and Recht, 2007) with JL compression (RFM-JL) and our proposed compression using FW (RFM-FW) for $N$ datapoints and $J_+ = \frac{1}{\epsilon} \log^2 \frac{1}{\epsilon}$ up-projection features.
Table 3: All datasets are taken from LIBSVM.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Samples</th>
<th>Dimension</th>
<th># Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>48,842</td>
<td>123</td>
<td>2</td>
</tr>
<tr>
<td>Human</td>
<td>10,299</td>
<td>561</td>
<td>6</td>
</tr>
<tr>
<td>MNIST</td>
<td>70,000</td>
<td>780</td>
<td>10</td>
</tr>
<tr>
<td>Sensorless</td>
<td>58,000</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>Criteo</td>
<td>51,882,752</td>
<td>1,000,000</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 1: Kernel matrix approximation error. Lower is better. Points average 20 runs; error bar is one standard deviation.

Figure 2: Classification accuracy. Higher is better. Points average 20 runs; error bar is one standard deviation.

Figure 3: Log clock time vs. kernel matrix approximation quality on the Criteo data. Lower is better.

For both RFM-JL and RFM-GIGA we set $J = 5,000$, and for RFM-GIGA we set $S = 20,000$.

Figs. 1 and 2 show the relative kernel matrix approximation error $\| Z Z^T - K \|_F / \| K \|_F$ and test classification accuracy, respectively, as a function of the number of compressed features $J$. Note that, since we cannot actually compute $K$, we approximate the relative Frobenius norm error by randomly sampling $10^4$ datapoints. We ran each experiment 20 times; the results in Figs. 1 and 2 show the mean across these trials with one standard deviation denoted with error bars. RFM-GIGA outperforms RFM and RFM-JL across all the datasets, on both metrics, for the full range of number of compressed features that we tested. This empirical result corroborates the theoretical results presented earlier in Section 3.2; in practice, RFM-GIGA requires approximately an order of magnitude fewer features than either RFM or RFM-JL.

To demonstrate the computational scalability of RFM-GIGA, we also plot the relative kernel matrix approximation error versus computation time for the Criteo
dataset, which consists of over 50 million data points. Before random feature projection and training, we used sparse random projections \cite{Li2018} to reduce the input dimensionality to 250 dimensions (due to memory constraints). We set \( J_0 = 5000 \) and \( S = 2 \times 10^4 \) as before, and let \( J \) vary between \( 10^2 \) and \( 10^3 \). The results of this experiment in Fig.\ref{fig:performance} suggest that RFM-GIGA provides a significant improvement in performance over both RFM and RFM-JL. Note that RFM-JL is very expensive in this setting—the up-projection step requires computing a \( 5 \times 10^8 \) by \( 5 \times 10^5 \) feature matrix—explaining its large computation time relative to RFM and RFM-GIGA. For test-set classification, all the methods performed the same for all choices of \( J \) (accuracy of \( 0.74 \pm 0.001 \)), so we do not provide the runtime vs. classification accuracy plot. This result is likely due to our compressing the \( 10^6 \)-dimensional feature space to 250 dimensions, making it hard for the SVM classifier to properly learn.

Given the empirical advantage of our proposed method, we next focus on understanding (1) if \( S \) can be set much smaller than \( \Omega(J_0^2(\log J_+)^2) \) in practice and (2) if we can get an exponential compression of \( J_+ \) in practice as Theorem \ref{thm:expansion} and Theorem \ref{thm:compression} guarantee.

To test the impact of \( S \) on performance, we fixed \( J_+ = 5000 \), and we let \( S \) vary between \( 10^2 \) and \( 10^6 \). Figure \ref{fig:performance} shows what the results in Fig.\ref{fig:results} would have looked like had we chosen a different \( S \). We clearly see that after around only \( S = 10000 \) there is a phase transition such that increasing \( S \) does not further improve performance.

To better understand if we actually see an exponential compression in \( J_+ \) in practice, as our theory suggests, we set \( J_+ = 10^5 \) (i.e. very large) and fixed \( S = 20000 \) as before. We examined the Higgs dataset consisting of \( 1.1 \times 10^6 \) samples, and let \( J \) (the number of compressed features) vary between 500 and \( 10^4 \). Since GIGA can select the same random feature at different iterations (i.e. give a feature higher weight), \( J \) reached 8600 after \( 10^5 \) iterations in Fig.\ref{fig:results}. Fig.\ref{fig:performance} shows that for \( J \approx 2 \times 10^3 \), increasing \( J \) further has negligible impact on kernel approximation performance—only 0.001 difference in relative error. Fig.\ref{fig:results} shows that we are able to compress \( J_+ \) by around two orders of magnitude.

Finally, since our proofs of Theorem \ref{thm:expansion} and Theorem \ref{thm:compression} assume Step 8 of Algorithm \ref{alg:GIGA} is run using Frank-Wolfe instead of GIGA, we compare in Fig.\ref{fig:results} how the results in Fig.\ref{fig:performance} change by using Frank-Wolfe instead. Fig.\ref{fig:results} shows that for \( J \) small, GIGA has better approximation quality than FW but for larger \( J \), the two perform nearly the same. This behavior agrees with the theory and empirical results of \cite{Campbell2018}, where GIGA is motivated specifically for the case of high compression.

5 Conclusion

This work presents a new algorithm for scalable kernel matrix approximation. We first generate a low-rank approximation. We then find a sparse, weighted subset of the columns of the low-rank factor that minimizes the Frobenius norm error relative to the original low-rank approximation. Theoretical and empirical results suggest that our method provides a substantial improvement in scalability and approximation quality over past techniques. Directions for future work include investigating the effects of variance reduction techniques for the up-projection, using a similar compression technique on features generated by the Nyström method \cite{Williams2001}, and transfer learning of feature weights for multiple related datasets.
Acknowledgments

We thank Justin Solomon for valuable discussions. This research was supported in part by an ARO YIP Award, ONR (N00014-17-1-2072), an NSF CAREER Award, the CSAIL-MIT Trustworthy AI Initiative, Amazon, and the MIT-IBM Watson AI Lab.

References


