Data-dependent compression of random features for large-scale kernel approximation

Raj Agrawal MIT
Trevor Campbell UBC
Jonathan Huggins Harvard
Tamara Broderick MIT

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.

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

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 (Hofmann et al., 2008; Schölkopf 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, 2015)—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 (Saunders 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 $\alpha^\star = (K + \lambda I)^{-1}y$ 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*

$$\begin{aligned}\phi(x-y) &= \int_{\mathbb{R}^p} e^{i\omega^T(x-y)}\mathrm{d}Q(\omega) \\ &= \mathbb{E}_Q[\psi_\omega(x)\psi_\omega(y)^*], \quad \psi_\omega(x) := e^{i\omega^T x}.\end{aligned} \tag{1}$$

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

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^TZ. For SVM, the time refers to fitting a linear SVM on Z using *dual-coordinate descent* with optimization tolerance ρ (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.

Method	Exact Training Cost	Low-Rank Training Cost	Approximation Error
PCA	$O(N^3)$	$\Theta(NJ^2)$	$O\left((1-\frac{\ell}{N})\|\hat{K}-K\|_F\right)$
SVM	$O(N^3)$	$\Theta(NJ\log\frac{1}{\rho})$	$O\left(\|\hat{K}-K\|_F^{\frac{1}{2}}\right)$
Ridge Regression	$O(N^3)$	$\Theta(NJ^2)$	$O\left(\frac{1}{N}\|\hat{K}-K\|_F\right)$

$z(x) := (1/\sqrt{J})[\cos(\omega_1^T x + b_1), \cdots, \cos(\omega_J^T x + b_J)]^T$, $b_j \overset{\text{unif.}}{\sim} [0, 2\pi]$ 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 (Rahimi and Recht, 2007; Sutherland and Schneider, 2015). The RFF 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 (Rahimi and Recht, 2007), as well as extensions to more general *random feature maps* (RFMs), which apply to many types of non-stationary kernels (Kar and Karnick, 2012; Pennington et al., 2015; Daniely et al., 2017).

The major drawback of RFMs is the $O(NJp)$ time and $O(NJ)$ memory costs associated with generating the feature matrix Z.[1] Although these are linear in N as desired, recent empirical evidence (Huang et al., 2014) 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 (Avron et al., 2016; Chang et al., 2017; Shen et al., 2017; Yu et al., 2016; Avron et al., 2017). 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_+ of features—followed by (2) "compression," in which those features are used to find a smaller number J 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 and storage of J_+ features prohibitively expensive for large datasets, JL compression is *data-independent* and leads to only a constant reduction in J_+ as we show in Appendix C (see summary in Table 2).

[1] *Fast Food RFM* can reduce the computational cost of generating the feature matrix to $O(NJ\log p)$ by exploiting techniques from sparse linear algebra. For simplicity, we focus on RFM here, but we note that our method can also be used on top of Fast Food RFM in cases when p is large.

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 $K \approx \hat{K} = ZZ^T$ 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 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 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

the whole dataset. This circumvents the expensive $O(NJ_+p)$ up-projection computation typical of past feature compression methods. In addition, we show that our greedy compression algorithm needs to output only $J = O(\log J_+)$ features—as opposed to past work, where $J = O(J_+)$ was required—while maintaining the same kernel approximation error provided by RFM with J_+ features. These results are summarized in Table 2 and discussed in detail in Section 3.2.

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 $\hat{K} = ZZ^T \approx K$, $Z \in \mathbb{R}^{N \times J}$. Our approach is motivated by the fact that spectral 2-norm bounds on $K - \hat{K}$ provide uniform bounds on the difference between learned models using K and $\hat{K}$ (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,

$$\begin{aligned}&\|K - ZZ^T\|_F \\ &\leq \|K - Z_+{Z_+}^T\|_F + \|Z_+{Z_+}^T - ZZ^T\|_F, \end{aligned} \tag{2}$$

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 jth column of Z_+. Then we would ideally like to solve the optimization problem

$$\begin{aligned}\underset{w \in \mathbb{R}_+^{J_+}}{\text{argmin}} \quad & \frac{1}{N^2}\|Z_+{Z_+}^T - Z(w)Z(w)^T\|_F^2 \\ \text{s.t.} \quad Z(w) &:= \begin{bmatrix} \sqrt{w_1}Z_{+1} & \cdots & \sqrt{w_{J_+}}Z_{+J_+} \end{bmatrix} \\ & \|w\|_0 \leq J. \end{aligned} \tag{3}$$

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

$$\begin{aligned}&\frac{1}{N^2}\|Z_+{Z_+}^T - Z(w)Z(w)^T\|_F^2 = \\ &\mathbb{E}_{i,j \overset{\text{i.i.d.}}{\sim} \pi}\left[(z_{+i}^T z_{+j} - z_i(w)^T z_j(w))^2\right], \end{aligned}$$

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

$$\begin{aligned}&\frac{S}{N^2}\|Z_+{Z_+}^T - Z(w)Z(w)^T\|_F^2 \\ &\approx \sum_{s=1}^{S}(z_{+i_s}^T z_{+j_s} - z_{i_s}(w)^T z_{j_s}(w))^2 \\ &= (1-w)^T RR^T(1-w) \text{ s.t.} \\ &R := \begin{bmatrix} z_{+i_1} \circ z_{+j_1}, & \cdots, & z_{+i_S} \circ z_{+j_S} \end{bmatrix} \in \mathbb{R}^{J_+ \times S}, \end{aligned} \tag{4}$$

where $\circ$ indicates a component-wise product. Denoting the jth 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

$$\begin{aligned}\underset{w \in \mathbb{R}_+^{J_+}}{\text{argmin}} \quad & \|r - r(w)\|_2^2 \\ \text{s.t.} \quad & \|w\|_0 \leq J, \end{aligned} \tag{5}$$

where $r(w) := \sum_{j=1}^{J_+} w_j R_j$. Note that the s^{th} component $r_s = z_{+i_s}^T z_{+j_s}$ of r is the Monte-Carlo approximation of $k(x_{i_s}, x_{j_s})$ using all J_+ features, while $r(w)_s = (\sqrt{w} \circ z_{+i_s})^T(\sqrt{w} \circ z_{+j_s})$ 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 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; Candes 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 $w \in \mathbb{R}^{J_+}$ with at most J non-zero entries

$(i_s, j_s)_{s=1}^S \overset{\text{i.i.d.}}{\sim} \text{Unif}(\{(i,j) : i < j, 2 \le j \le N\})$.

Sample $(\omega_j)_{j=1}^{J_+} \overset{\text{i.i.d.}}{\sim} Q$

Sample $b_j \overset{\text{unif.}}{\sim} [0, 2\pi], 1 \le j \le J_+$

for $s = 1 : S$ **do**

  Compute $z_{+i_s} \leftarrow (1/\sqrt{J_+})[\cos(\omega_1^T x_{i_s} + b_1), \cdots, \cos(\omega_{J_+}^T x_{i_s} + b_{J_+})]^T$; same for z_{+j_s}

Compute $R \leftarrow \left[z_{+i_1} \circ z_{+j_1}, \ \cdots, \ z_{+i_S} \circ z_{+j_S}\right]$

$R_j \leftarrow$ row j of R; $r \leftarrow \sum_{j=1}^{J_+} R_j$

$w \leftarrow$ solution to Eq. (5) with FW (Campbell and Broderick, 2019) or GIGA (Campbell and Broderick, 2018)

$Z(w) = \begin{bmatrix} \sqrt{w_1} Z_{+1} & \cdots & \sqrt{w_{J_+}} Z_{+J_+} \end{bmatrix}$

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 features method provides 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 $\{x_i - x_j, x_i + x_j\}_{1 \le i < j \le N}$ is $\frac{N(N-1)}{2}$, i.e., all vectors $x_i - x_j, x_i + x_j, 1 \le i < j \le N$ 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 ν_{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_j's contain continuous (noisy) measurements, Assumption 3.1(a) is very mild since the difference or sum between two datapoints is unlikely to equal the difference or sum between two other datapoints. 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. (5) 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 by $r(w)$ 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 3.4.

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 $0 \le c_\delta^* < \infty$, which depends only on δ, such that if*

$$J = \Omega\left(-\frac{\log J_+}{\log \nu_{J_+}}\right) \text{ and } S = \Omega\left(\frac{c_\delta^*}{\epsilon^2}\left[\frac{\log \frac{1}{\epsilon}}{\log \nu_{J_+}}\right]^4 \log J_+\right),$$

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

$$\frac{1}{N^2}\|Z_+ Z_+^T - ZZ^T\|_F^2 \le \epsilon.$$

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

$$0 < \limsup_{J_+ \to \infty} \nu_{J_+} < 1. \tag{6}$$

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 $J_+ = {}^1/_\epsilon \log {}^1/_\epsilon$ 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.

Method	# Compressed Features J	Cost of Computing Z	PCA/Ridge Reg. Cost
RFM	$O(J_+)$	$O(NJ_+)$	$O\left(NJ_+^2\right)$
RFM-JL	$O(J_+)$	$O(NJ_+ \log J_+)$	$O\left(NJ_+^2\right)$
RFM-FW	$O(\log J_+)$	$O(SJ_+ \log J_+ + N \log J_+)$	$O\left(N(\log J_+)^2\right)$

Proof. Claim 1 of Rahimi and Recht (2007) implies that $\frac{1}{N^2}\|K - Z_+{Z_+}^T\|_F^2 = O(\epsilon)$ if we set $J_+ = \Omega({}^1/_\epsilon \log {}^1/_\epsilon)$. 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^TZ 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 ϵ. 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 ν_{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* $\{(i,j) : 1 \le i < j \le N\}$ *are sampled in Algorithm 1. Then,*

$$0 < \limsup_{J_+ \to \infty} \nu_{J_+} < 1 - \frac{\left(1 - \frac{\|K\|_F}{c_Q}\right)^2}{2} < 1, \tag{7}$$

where K *is the exact kernel matrix and*

$$\begin{aligned} c_Q &:= \frac{1}{N}\mathbb{E}_{\omega\sim Q, b\sim Unif[0,2\pi]}\|u(\omega,b)\|_2, \text{ with} \\ u(\omega,b) &:= (\cos(\omega^T x_i + b)\cos(\omega^T x_j + b))_{i,j\in[N]}. \end{aligned} \tag{8}$$

By Theorem 2.1, $\|K\|_F = \frac{1}{N}\|\mathbb{E}_{\omega,b}u(\omega,b)\|_2$, so $\|K\|_F \le c_Q$ 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 λ_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,

Table 3: All datasets are taken from `LIBSVM`.

Dataset	# Samples	Dimension	# Classes
Adult	48,842	123	2
Human	10,299	561	6
MNIST	70,000	780	10
Sensorless	58,000	9	11
Criteo	51,882,752	1,000,000	2

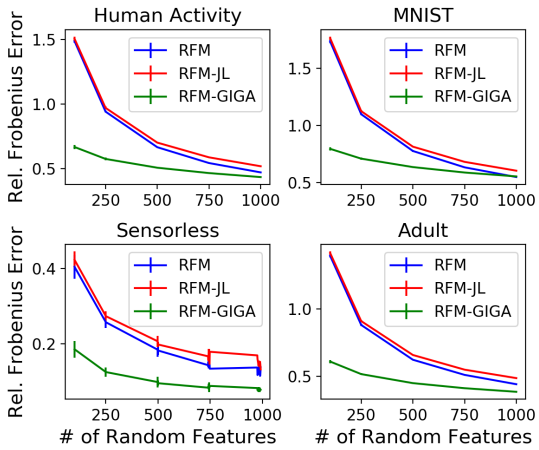

Figure 1: Kernel matrix approximation error. Lower is better. Points average 20 runs; error bar is one standard deviation.

2007), RFM with Johnson-Lindenstrauss compression (RFM-JL) (Hamid et al., 2014), and our proposed algorithm with compression via greedy iterative geodesic ascent (Campbell and Broderick, 2018) (RFM-GIGA). We note that there are many other random feature methods, such as Quasi-Monte-Carlo random features (Avron et al., 2016), that one might consider besides RFM-JL. A strength of our method is that it can be used as an additional compression step with these methods and is thus complementary with them; we discuss this idea and demonstrate the resulting improvements in Appendix E. In this section, we focus on Johnson-Lindenstrauss as the current state-of-the-art random features compression method.

We compare performance on the task of kernel SVM classification (Vapnik et al., 1997). We consider five real, large-scale datasets, summarized in Table 3. We assess performance via two quality metrics—Frobenius error of the kernel approximation and test set classification error. We also measure overall computation time—including both random feature projection and SVM training. We use the radial basis kernel $k(x, y) = e^{-\gamma\|x-y\|^2}$; we pick both γ and the SVM regularization strength for each dataset by randomly sampling 10,000 datapoints, training an exact kernel SVM on those datapoints, and using 5-fold cross-validation.

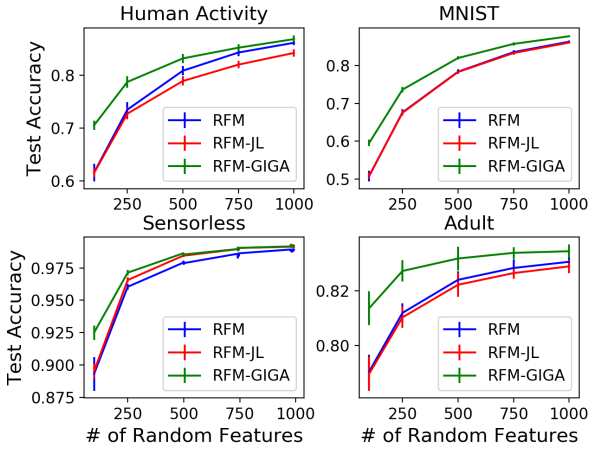

Figure 2: Classification accuracy. Higher is better. Points average 20 runs; error bar is one standard deviation.

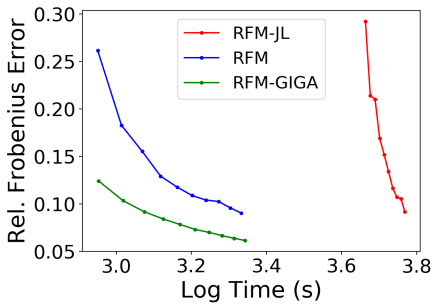

Figure 3: Log clock time vs. kernel matrix approximation quality on the Criteo data. Lower is better.

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 $\|ZZ^T - K\|_F / \|K\|_F$ 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 (Li et al., 2006) to reduce the input dimensionality to 250 dimensions (due to memory constraints). We set $J_+ = 5000$ 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. 3 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 5×10^8 by 5×10^3 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 ± 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 $\Omega(J_+^2(\log J_+)^2))$ in practice and (2) if we can get an exponential compression of J_+ in practice as Theorem 3.2 and Theorem 3.4 guarantee.

To test the impact of S on performance, we fixed $J_+ =$ 5,000, and we let S vary between 10^2 and 10^6. Figure 4 shows what the results in Fig. 1 would have looked like had we chosen a different S. We clearly see that after around only $S = 10{,}000$ 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 $S = 20{,}000$ as before. We examined the HIGGS dataset consisting of 1.1×10^7 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 8,600 after 10^4 iterations in Fig. 5. Fig. 5 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. 5 shows that we are able to compress J_+ by around two orders of magnitude.

Finally, since our proofs of Theorem 3.2 and Theorem 3.4 assume Step 8 of Algorithm 1 is run using Frank-Wolfe instead of GIGA, we compare in Fig. 6 how the results in Fig. 1 change by using Frank-Wolfe instead. Fig. 6 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 Campbell and Broderick (2018), where GIGA is motivated specifically for the case of high compression.

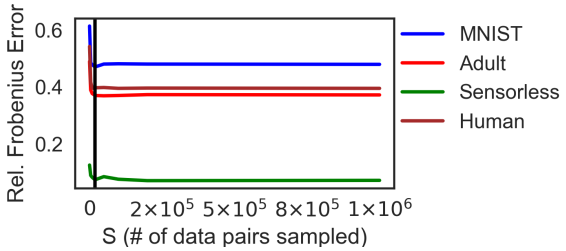

Figure 4: We plot the relative Frobenius norm error against S for J_+ fixed at 5,000. The solid black line corresponds to the results found in Fig. 1.

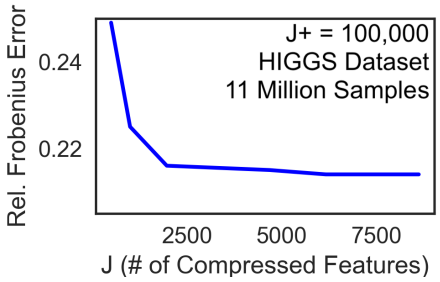

Figure 5: Let $S = 20{,}000$, $J_+ = 10^5$. We plot the relative Frobenius norm error vs. J from 500 to 10^4.

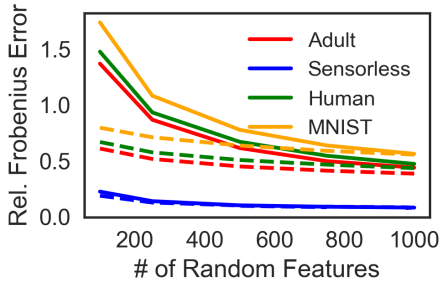

Figure 6: The performance of GIGA versus Frank-Wolfe for the experiment described in Fig. 1. Solid lines correspond to Frank-Wolfe and dashed with GIGA.

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 (Williams and Seeger, 2001), 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

H. Avron, V. Sindhwani, J. Yang, and M. W. Mahoney. Quasi-Monte Carlo feature maps for shift-invariant kernels. *Journal of Machine Learning Research*, pages 1–38, 2016.

H. Avron, M. Kapralov, C. Musco, C. Musco, A. Velingker, and A. Zandieh. Random Fourier features for kernel ridge regression: Approximation bounds and statistical guarantees. In *International Conference on Machine Learning*, 2017.

M. Balcan, A. Blum, and S. Vempala. On kernels, margins, and low-dimensional mappings. In *Algorithmic Learning Theory*, pages 1–12, 2008.

B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Workshop on Computational Learning Theory*, pages 144–152, 1992.

T. Campbell and T. Broderick. Bayesian coreset construction via greedy iterative geodesic ascent. In *International Conference on Machine Learning*, 2018.

T. Campbell and T. Broderick. Automated scalable Bayesian inference via Hilbert coresets. *Journal of Machine Learning Research*, 2019.

E. Candes and T. Tao. The Dantzig selector: Statistical estimation when p is much larger than n. *The Annals of Statistics*, pages 2313–2351, 2007.

W. Chang, C. Li, Y. Yang, and B. Poczos. Data-driven random Fourier features using Stein effect. In *International Joint Conference on Artificial Intelligence*, pages 1497–1503, 2017.

S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, pages 33–61, 1998.

K. Chwialkowski, H. Strathmann, and A. Gretton. A kernel test of goodness of fit. In *International Conference on Machine Learning*, 2016.

C. Cortes, M. Mohri, and A. Talwalkar. On the impact of kernel approximation on learning accuracy. In *International Conference on Artificial Intelligence and Statistics*, 2010.

A. Daniely, R. Frostig, V. Gupta, and Y. Singer. Random features for compositional kernels. *arXiv:1703.07872*, 2017.

P. Drineas and M. Mahoney. On the Nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, pages 2153–2175, 2005.

A. El Alaoui and M. Mahoney. Fast randomized kernel methods with statistical guarantees. In *Advances in Neural Information Processing Systems*, 2015.

A. Gretton, K. Fukumizu, C. H., L. Song, B. Schölkopf, and A. Smola. A kernel statistical test of independence. In *Advances in Neural Information Processing Systems*, pages 585–592, 2008.

A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, pages 723–773, 2012.

N. Halko, P. Martinsson, and J. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, pages 217–288, 2011.

R. Hamid, Y. Xiao, A. Gittens, and D. DeCoste. Compact random feature maps. In *International Conference on International Conference on Machine Learning*, 2014.

T. Hofmann, B. Schölkopf, and A. Smola. Kernel methods in machine learning. *The Annals of Statistics*, pages 1171–1220, 2008.

J. Honorio and Y.-J. Li. The error probability of random Fourier features is dimensionality independent. *arXiv:1710.09953*, 2017.

C. Hsieh, K. Chang, C. Lin, S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *International Conference on Machine Learning*, pages 408–415, 2008.

P. Huang, H. Avron, T. Sainath, V. Sindhwani, and B. Ramabhadran. Kernel methods match deep neural networks on TIMIT. In *International Conference on Acoustics, Speech and Signal Processing*, pages 205–209, May 2014.

W. Johnson, J. Lindenstrauss, and G. Schechtman. Extensions of Lipschitz maps into Banach spaces. *Israel Journal of Mathematics*, pages 129–138, 1986.

S. Kakade and G. Shakhnarovich. Lecture notes in large scale learning, 2009. URL `http://ttic.uchicago.edu/~gregory/courses/LargeScaleLearning/lectures/jl.pdf`.

P. Kar and H. Karnick. Random feature maps for dot product kernels. In *International Conference on Artificial Intelligence and Statistics*, pages 583–591, 2012.

Q. Le, T. Sarlos, and A. Smola. Fastfood - approximating kernel expansions in loglinear time. In *International Conference on Machine Learning*, 2013.

P. Li, T. Hastie, and K. Church. Very sparse random projections. In *International Conference on Knowledge Discovery and Data Mining*, pages 287–296, 2006.

W. Lim, R. Du, B. Dai, K. Jung, L. Song, and H. Park. Multi-scale Nystrom method. In *International Conference on Artificial Intelligence and Statistics*, 2018.

F. Marguerite and W. Philip. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, pages 95–110, 1956.

S. Mendelson. On the performance of kernel classes. *Journal of Machine Learning Research*, pages 759–771, 2003.

C. Musco and C. Musco. Recursive sampling for the Nyström method. In *Advances in Neural Information Processing Systems*, 2017.

J. Pennington, F. Yu, and S. Kumar. Spherical random features for polynomial kernels. In *Advances in Neural Information Processing Systems*, pages 1846–1854, 2015.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Neural Information Processing Systems*, 2007.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2008.

A. Rudi and L. Rosasco. Generalization properties of learning with random features. In *Advances in Neural Information Processing Systems*, 2017.

A. Rudi, R. Camoriano, and L. Rosasco. Less is more: Nyström computational regularization. In *Advances in Neural Information Processing Systems*, 2015.

W. Rudin. *Fourier Analysis on Groups*, chapter The Basic Theorems of Fourier Analysis. Wiley, 1994.

Y. Samo and S. Roberts. Generalized spectral kernels. *arXiv:1506.02236*, 2015.

C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *International Conference on Machine Learning*, pages 515–521, 1998.

B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

B. Schölkopf, A. Smola, and K. Müller. Kernel principal component analysis. In *Artificial Neural Networks*, pages 583–588, 1997.

W. Shen, Z. Yang, and J. Wang. Random features for shift-invariant kernels with moment matching. In *Association for the Advancement of Artificial Intelligence Conference*, 2017.

B. Sriperumbudur and N. Sterge. Approximate kernel PCA using random features: Computational vs. statistical trade-off. *arXiv:1706.06296*, 2017.

B. Sriperumbudur, A. Gretton, K. Fukumizu, B. Schölkopf, and G. Lanckriet. Hilbert space embeddings and metrics on probability measures. *Journal of Machine Learning Research*, pages 1517–1561, 2010.

D. Sutherland and J. Schneider. On the error of random Fourier features. In *Conference on Uncertainty in Artificial Intelligence*, pages 862–871, 2015.

A. Talwalkar. *Matrix Approximation for Large-scale Learning*. PhD thesis, New York University, 2010.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, pages 267–288, 1994.

V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.

V. Vapnik, S. Golowich, and A. Smola. Support vector method for function approximation, regression estimation and signal processing. In *Advances in Neural Information Processing Systems*, pages 281–287, 1997.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pages 682–688, 2001.

T. Yang, Y. Li, M. Mahdavi, R. Jin, and Z. Zhou. Nyström method vs random Fourier features - a theoretical and empirical comparison. In *Advances in Neural Information Processing Systems*, 2012.

Y. Yang, M. Pilanci, and M. J. Wainwright. Randomized sketches for kernels: Fast and optimal nonparametric regression. *The Annals of Statistics*, pages 991–1023, 2017.

F. Yu, A. Suresh, K. Choromanski, D. Holtmann-Rice, and S. Kumar. Orthogonal random features. In *Advances in Neural Information Processing Systems*, pages 1975–1983, 2016.

K. Zhang, J. Peters, D. Janzing, and B. Schölkopf. Kernel-based conditional independence test and application in causal discovery. In *Conference on Uncertainty in Artificial Intelligence*, pages 804–813, 2011.