Random Fourier Features via Fast Surrogate Leverage Weighted Sampling

Fanghui Liu,[1,2] **Xiaolin Huang,**[2,3] **Yudong Chen,**[4] **Jie Yang,**[2,3] **Johan A.K. Suykens**[1]
[1]Department of Electrical Engineering (ESAT-STADIUS), KU Leuven, Belgium
[2]Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, China
[3]Institute of Medical Robotics, Shanghai Jiao Tong University, China
[4]School of Operations Research and Information Engineering, Cornell University, USA
fanghui.liu@kuleuven.be, {xiaolinhuang, jieyang}@sjtu.edu.cn, yudong.chen@cornell.edu, johan.suykens@esat.kuleuven.be

Abstract

In this paper, we propose a fast surrogate leverage weighted sampling strategy to generate refined random Fourier features for kernel approximation. Compared to the current state-of-the-art method that uses the leverage weighted scheme (Li et al. 2019), our new strategy is simpler and more effective. It uses kernel alignment to guide the sampling process and it can avoid the matrix inversion operator when we compute the leverage function. Given n observations and s random features, our strategy can reduce the time complexity for sampling from $\mathcal{O}(ns^2+s^3)$ to $\mathcal{O}(ns^2)$, while achieving comparable (or even slightly better) prediction performance when applied to kernel ridge regression (KRR). In addition, we provide theoretical guarantees on the generalization performance of our approach, and in particular characterize the number of random features required to achieve statistical guarantees in KRR. Experiments on several benchmark datasets demonstrate that our algorithm achieves comparable prediction performance and takes less time cost when compared to (Li et al. 2019).

1 Introduction

Kernel methods (Schölkopf and Smola 2003) are one of the most important and powerful tools in statistical learning. However, kernel methods often suffer from scalability issues in large-scale problems due to high space and time complexities. For example, given n observations in the original d-dimensional space $\mathcal{X}$, kernel ridge regression (KRR) requires $\mathcal{O}(n^3)$ training time and $\mathcal{O}(n^2)$ space to store the kernel matrix, which becomes intractable when n is large.

One of the most popular approaches for scaling up kernel methods is random Fourier features (RFF) (Rahimi and Recht 2007), which approximates the original kernel by mapping input features into a new space spanned by a small number of Fourier basis. Specifically, suppose a given kernel $k(\cdot,\cdot): \mathcal{X}\times\mathcal{X}\to\mathbb{R}$ satisfies 1) positive definiteness and 2) shift-invariance, i.e., $k(\boldsymbol{x},\boldsymbol{x}')=k(\boldsymbol{x}-\boldsymbol{x}')$. By Bochner's theorem (Bochner 2005), there exists a finite Borel measure $p(\boldsymbol{w})$ (the Fourier transform associated with k) such that

$$k(\boldsymbol{x}-\boldsymbol{x}')=\int_{\mathbb{R}^d} p(\boldsymbol{w})\exp\left(\mathrm{i}\boldsymbol{w}^\top(\boldsymbol{x}-\boldsymbol{x}')\right)\mathrm{d}\boldsymbol{w}\,. \quad (1)$$

(Typically, the kernel is real-valued and thus the imaginary part in Eq. (1) can be discarded.) One can then use Monte Carlo sampling to approximate $k(\boldsymbol{x},\boldsymbol{x}')$ by the low-dimensional kernel $\tilde{k}_p(\boldsymbol{x},\boldsymbol{x}')=\varphi_p(\boldsymbol{x})^\top\varphi_p(\boldsymbol{x}')$ with the explicit mapping

$$\varphi_p(\boldsymbol{x}):=\frac{1}{\sqrt{s}}\left[\exp(-\mathrm{i}\boldsymbol{w}_1^\top\boldsymbol{x}),\cdots,\exp(-\mathrm{i}\boldsymbol{w}_s^\top\boldsymbol{x})\right]^\top, \quad (2)$$

where $\{\boldsymbol{w}_i\}_{i=1}^s$ are sampled from $p(\boldsymbol{w})$ *independently* of the training set. For notational simplicity, here we write $z_p(\boldsymbol{w}_i,\boldsymbol{x}_j) := 1/\sqrt{s}\exp(-\mathrm{i}\boldsymbol{w}_i^\top\boldsymbol{x}_j)$ such that $\varphi_p(\boldsymbol{x}) = [z_p(\boldsymbol{w}_1,\boldsymbol{x}),\cdots,z_p(\boldsymbol{w}_s,\boldsymbol{x})]^\top$. Note that we have $k(\boldsymbol{x},\boldsymbol{x}') = \mathbb{E}_{\boldsymbol{w}\sim p}[\varphi_p(\boldsymbol{x})^\top\varphi_p(\boldsymbol{x}')] \approx \tilde{k}_p(\boldsymbol{x},\boldsymbol{x}') = \sum_{i=1}^s z_p(\boldsymbol{w}_i,\boldsymbol{x})z_p(\boldsymbol{w}_i,\boldsymbol{x}')$. Consequently, the original kernel matrix $\boldsymbol{K}=[k(\boldsymbol{x}_i,\boldsymbol{x}_j)]_{n\times n}$ on the n observations $\boldsymbol{X}=\{\boldsymbol{x}_i\}_{i=1}^n$ can be approximated by $\boldsymbol{K}\approx\tilde{\boldsymbol{K}}_p=\boldsymbol{Z}_p\boldsymbol{Z}_p^\top$, where $\boldsymbol{Z}_p=[\varphi_p(\boldsymbol{x}_1),\cdots,\varphi_p(\boldsymbol{x}_n)]^\top\in\mathbb{R}^{n\times s}$. With s random features, this approximation applied to KRR only requires $\mathcal{O}(ns^2)$ time and $\mathcal{O}(ns)$ memory, hence achieving a substantial computational saving when $s\ll n$.

Since RFF uses the Monte Carlo estimates that are independent of the training set, a large number of random features are often required to achieve competitive approximation and generalization performance. To improve performance, recent works (Sun, Gilbert, and Tewari 2018; Li et al. 2019) consider using the ridge leverage function (Bach 2017; Avron et al. 2017) defined with respect to the training data. For a given random feature $\boldsymbol{w}_i$, this function is defined as

$$l_\lambda(\boldsymbol{w}_i)=p(\boldsymbol{w}_i)\boldsymbol{z}_{p,\boldsymbol{w}_i}^\top(\boldsymbol{X})(\boldsymbol{K}+n\lambda\boldsymbol{I})^{-1}\boldsymbol{z}_{p,\boldsymbol{w}_i}(\boldsymbol{X})\,, \quad (3)$$

where λ is the regularization parameter in KRR and $\boldsymbol{z}_{p,\boldsymbol{w}_i}(\boldsymbol{X})\in\mathbb{R}^n$ is the i-th column of $\boldsymbol{Z}_p$ given by $(\boldsymbol{z}_{p,\boldsymbol{w}_i}(\boldsymbol{X}))_j := z_p(\boldsymbol{w}_i,\boldsymbol{x}_j)$. Observe that $q^*(\boldsymbol{w}) := \frac{l_\lambda(\boldsymbol{w})}{\int l_\lambda(\boldsymbol{w})\mathrm{d}\boldsymbol{w}}$ can be viewed as a probability density function, and hence is referred to as the *Empirical Ridge Leverage Score* (ERLS) distribution (Avron et al. 2017). Therefore, one can sample the features $\{\boldsymbol{w}_i\}_{i=1}^s$ according to $q^*(\boldsymbol{w})$, which is an importance weighted sampling strategy. Compared to standard Monte Carlo sampling for RFF, $q^*(\boldsymbol{w})$-based sampling requires fewer Fourier features and enjoys theoretical guarantees (Avron et al. 2017; Li et al. 2019).

However, computing the ridge leverage scores and the ERLS distribution may be intractable when n is large, as we need to invert the kernel matrix in Eq. (3). An alternative way in (Sun, Gilbert, and Tewari 2018; Li et al. 2019) is to use the subset of data to approximate $\boldsymbol{K}$, but this scheme still needs $\mathcal{O}(ns^2 + s^3)$ time to generate random features. To address these computational difficulties, we design a simple but effective leverage function to replace the original one. For a given $\boldsymbol{w}$, our leverage function is defined as

$$L_\lambda(\boldsymbol{w}) = p(\boldsymbol{w})\boldsymbol{z}_{p,\boldsymbol{w}}^\top(\boldsymbol{X})\left(\frac{1}{n^2\lambda}\Big(\boldsymbol{y}\boldsymbol{y}^\top + n\boldsymbol{I}\Big)\right)\boldsymbol{z}_{p,\boldsymbol{w}}(\boldsymbol{X}), \tag{4}$$

where the matrix $\boldsymbol{y}\boldsymbol{y}^\top$ is an ideal kernel that directly fits the training data with 100% accuracy in classification tasks, and thus can be used to guide kernel learning tasks as in kernel alignment (Cortes, Mohri, and Rostamizadeh 2012). It can be found that, our *surrogate* function avoids the matrix inversion operator so as to further accelerate kernel approximation. Note that, we introduce the additional term $n\boldsymbol{I}$ and the coefficient $1/(n^2\lambda)$ in Eq. (4) to ensure, L_λ is a *surrogate* function that upper bounds l_λ in Eq. (3) for theoretical guarantees. This can be achieved due to $L_\lambda(\boldsymbol{w}_i) \geq l_\lambda(\boldsymbol{w}_i)$[1]. In this way, our method with the *surrogate* function requires less computational time while achieving comparable generalization performance, as demonstrated by our theoretical results and experimental validations.

Specifically, the main contributions of this paper are:

- We propose a surrogate ridge leverage function based on kernel alignment and derive its associated fast surrogate ERLS distribution. This distribution is simple in form and has intuitive physical meanings. Our theoretical analysis provides a lower bound on the number of random features that guarantees no loss in the learning accuracy in KRR.
- By sampling from the surrogate ERLS distribution, our *data-dependent* algorithm takes $\mathcal{O}(ns^2)$ time to generate random features, which is the same as RFF and less than the $\mathcal{O}(ns^2+s^3)$ time in (Li et al. 2019). We further provide theoretical guarantees on the generalization performance of our algorithm equipped with the KRR estimator.
- Experiments on various benchmark datasets demonstrate that our method performs better than standard random Fourier features based algorithms. Specifically, our algorithm achieves comparable (or even better) accuracy and uses less time when compared to (Li et al. 2019).

The remainder of the paper is organized as follows. Section 2 briefly reviews the related work on random features for kernel approximation. Our surrogate leverage weighted sampling strategy for RFF is presented in Section 3, and related theoretical results are given in Section 4. In section 5, we provide experimental evaluation for our algorithm and compare with other representative random features based methods on popular benchmarks. The paper is concluded in Section 6.

[1]It holds by $(\boldsymbol{K} + n\lambda\boldsymbol{I})^{-1} \preceq (n\lambda\boldsymbol{I})^{-1} \preceq \frac{1}{n^2\lambda}(\boldsymbol{y}\boldsymbol{y}^\top + n\boldsymbol{I})$, where the notation $0 \preceq \boldsymbol{A}$ denotes that $\boldsymbol{A}$ is semi-positive definite.

2 Related Works

Recent research on random Fourier features focuses on constructing the mapping

$$\varphi(\boldsymbol{x}) := \frac{1}{\sqrt{s}}\left[a_1\exp(-\mathrm{i}\boldsymbol{w}_1^\top\boldsymbol{x}), \cdots, a_s\exp(-\mathrm{i}\boldsymbol{w}_s^\top\boldsymbol{x})\right]^\top.$$

The key question is how to select the points $\boldsymbol{w}_i$ and weights a_i so as to uniformly approximate the integral in Eq. (1). In standard RFF, $\{\boldsymbol{w}_i\}_{i=1}^s$ are randomly sampled from $p(\boldsymbol{w})$ and the weights are equal, i.e., $a_i \equiv 1$. To reduce the approximation variance, Yu et al. (2016) proposes the orthogonal random features (ORF) approach, which incorporates an orthogonality constraint when sampling $\{\boldsymbol{w}_i\}_{i=1}^s$ from $p(\boldsymbol{w})$. Sampling theory Niederreiter (1992) suggests that the convergence of Monte-Carlo used in RFF and ORF can be significantly improved by choosing a deterministic sequence $\{\boldsymbol{w}_i\}$ instead of sampling randomly. Therefore, a possible middle-ground method is Quasi-Monte Carlo sampling (Avron et al. 2016), which uses a low-discrepancy sequence $\{\boldsymbol{w}_i\}$ rather than the fully random Monte Carlo samples. Other deterministic approaches based on numerical quadrature are considered in (Evans 1993). Bach (2017) analyzes the relationship between random features and quadrature, which allows one to use deterministic numerical integration methods such as Gaussian quadrature (Dao, De Sa, and Ré 2017), spherical-radial quadrature rules (Munkhoeva et al. 2018), and sparse quadratures (Gauthier and Suykens 2018) for kernel approximation.

The above methods are all *data-independent*, i.e., the selection of points and weights is independent of the training data. Another line of work considers *data-dependent* algorithms, which use the training data to guide the generation of random Fourier features by using, e.g., kernel alignment (Sinha and Duchi 2016), feature compression (Agrawal et al. 2019), or the ridge leverage function (Avron et al. 2017; Sun, Gilbert, and Tewari 2018; Li et al. 2019; Fanuel, Schreurs, and Suykens 2019). Since our method builds on the leverage function $l_\lambda(\boldsymbol{w})$, we detail this approach here. From Eq. (3), the integral of $l_\lambda(\boldsymbol{w})$ is

$$\int_{\mathbb{R}^d} l_\lambda(\boldsymbol{w})\mathrm{d}\boldsymbol{w} = \mathrm{Tr}\left[\boldsymbol{K}(\boldsymbol{K} + n\lambda\boldsymbol{I})^{-1}\right] =: d_{\boldsymbol{K}}^\lambda. \tag{5}$$

The quantity $d_{\boldsymbol{K}}^\lambda \ll n$ determines the number of independent parameters in a learning problem and hence is referred to as the *number of effective degrees of freedom* (Bach 2013). Li et al. (2019) provides the sharpest bound on the required number of random features; in particular, with $\Omega(\sqrt{n}\log d_{\boldsymbol{K}}^\lambda)$ features, no loss is incurred in learning accuracy of kernel ridge regression. Albeit elegant, sampling according to $q^*(\boldsymbol{w})$ is often intractable in practice. The alternative approach proposed in (Li et al. 2019) takes $\mathcal{O}(ns^2 + s^3)$ time, which is larger than $\mathcal{O}(ns^2)$ in the standard RFF.

3 Surrogate Leverage Weighted RFF

3.1 Problem setting

Consider a standard supervised learning setting, where $\mathcal{X}$ is a compact metric space of features, and $\mathcal{Y} \subseteq \mathbb{R}$ (in regression) or $\mathcal{Y} = \{-1, 1\}$ (in classification) is the label space.

We assume that a sample set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ is drawn from a non-degenerate Borel probability measure ρ on $\mathcal{X} \times \mathcal{Y}$. The *target function* of ρ is defined by $f_\rho(\boldsymbol{x}) := \int_{\mathcal{Y}} y \mathrm{d}\rho(y|\boldsymbol{x})$ for each $\boldsymbol{x} \in \mathcal{X}$, where $\rho(\cdot|\boldsymbol{x})$ is the conditional distribution of ρ at $\boldsymbol{x}$. Given a kernel function k and its associated reproducing kernel Hilbert space (RKHS) $\mathcal{H}$, the goal is to find a hypothesis $f : \mathcal{X} \to \mathcal{Y}$ in $\mathcal{H}$ such that $f(\boldsymbol{x})$ is a good estimate of the label $y \in \mathcal{Y}$ for a new instance $\boldsymbol{x} \in \mathcal{X}$. By virtue of the representer theorem (Schölkopf and Smola 2003), an empirical risk minimization problem can be formulated as

$$\hat{f}^\lambda := \operatorname*{argmin}_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \ell\left(y_i, f(\boldsymbol{x}_i)\right) + \lambda \boldsymbol{\alpha}^\top \boldsymbol{K} \boldsymbol{\alpha}\,, \quad (6)$$

where $f = \sum_{i=1}^n \alpha_i k(\boldsymbol{x}_i, \cdot)$ with $\boldsymbol{\alpha} \in \mathbb{R}^n$ and the convex loss $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ quantifies the quality of the estimate $f(\boldsymbol{x})$ w.r.t. the true y. In this paper, we focus on learning with the squared loss, i.e., $\ell(y, f(\boldsymbol{x})) = (y - f(\boldsymbol{x}))^2$. Hence, the expected risk in KRR is defined as $\mathcal{E}(f) = \int_{\mathcal{X}\times\mathcal{Y}} (f(\boldsymbol{x}) - y)^2 \mathrm{d}\rho$, with the corresponding empirical risk defined on the sample, i.e., $\hat{\mathcal{E}}(f) = \frac{1}{n} \sum_{i=1}^n \left(f(\boldsymbol{x}_i) - y_i\right)^2$. In standard learning theory, the optimal hypothesis f_ρ is defined as $f_\rho(\boldsymbol{x}) = \int_Y y \mathrm{d}\rho(y|\boldsymbol{x}), \quad \boldsymbol{x} \in \mathcal{X}$, where $\rho(\cdot|\boldsymbol{x})$ is the conditional distribution of ρ at $\boldsymbol{x} \in \mathcal{X}$. The regularization parameter λ in Eq. (6) should depend on the sample size; in particular, $\lambda \equiv \lambda(n)$ with $\lim_{n\to\infty} \lambda(n) = 0$. Following (Rudi, Camoriano, and Rosasco 2017; Li et al. 2019), we pick $\lambda \in \mathcal{O}(n^{-1/2})$.

As shown in (Li et al. 2019), when using random features, the empirical risk minimization problem (6) can be expressed as

$$\boldsymbol{\beta}_\lambda := \operatorname*{argmin}_{\boldsymbol{\beta} \in \mathbb{R}^s} \frac{1}{n} \left\|\boldsymbol{y} - \boldsymbol{Z}_q \boldsymbol{\beta}\right\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2\,, \quad (7)$$

where $\boldsymbol{y} = [y_1, y_2, \cdots, y_n]^\top$ is the label vector and $\boldsymbol{Z}_q = [\varphi_q(\boldsymbol{x}_1), \cdots, \varphi_q(\boldsymbol{x}_n)]^\top \in \mathbb{R}^{n\times s}$ is the random feature matrix, with $\varphi_q(\boldsymbol{x})$ as defined in Eq. (2) and $\{\boldsymbol{w}_i\}_{i=1}^s$ sampled from a distribution $q(\boldsymbol{w})$. Eq. (7) is a linear ridge regression problem in the space spanned by the random features (Suykens et al. 2002; Mall and Suykens 2015), and the optimal hypothesis is given by $\boldsymbol{f}_{\boldsymbol{\beta}}^\lambda = \boldsymbol{Z}_q \boldsymbol{\beta}_\lambda$, with

$$\boldsymbol{\beta}_\lambda = (\boldsymbol{Z}_q^\top \boldsymbol{Z}_q + n\lambda \boldsymbol{I})^{-1} \boldsymbol{Z}_q^\top \boldsymbol{y}\,. \quad (8)$$

Note that the distribution $q(\boldsymbol{w})$ determines the feature mapping matrix and hence has a significant impact on the generalization performance. Our main goal in the sequel is to design a good $q(\boldsymbol{w})$, and to understand the relationship between $q(\boldsymbol{w})$ and the expected risk. In particular, we would like to characterize the number s of random features needed when sampling from $q(\boldsymbol{w})$ in order to achieve a certain convergence rate of the risk.

3.2 Surrogate leverage weighted sampling

Let $q(\boldsymbol{w})$ be a probability density function to be designed. Given the points $\{\boldsymbol{w}_i\}_{i=1}^s$ sampled from $q(\boldsymbol{w})$, we define the mapping

$$\varphi_q(\boldsymbol{x}) = \frac{1}{\sqrt{s}} \left(\sqrt{\frac{p(\boldsymbol{w}_1)}{q(\boldsymbol{w}_1)}} e^{-\mathrm{i}\boldsymbol{w}_1^\top \boldsymbol{x}}, \cdots, \sqrt{\frac{p(\boldsymbol{w}_s)}{q(\boldsymbol{w}_s)}} e^{-\mathrm{i}\boldsymbol{w}_s^\top \boldsymbol{x}} \right)^\top . \quad (9)$$

We again have $k(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{\boldsymbol{w}\sim q}[\varphi_q(\boldsymbol{x})^\top \varphi_q(\boldsymbol{x}')] \approx \tilde{k}_q(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i=1}^s z_q(\boldsymbol{w}_i, \boldsymbol{x}) z_q(\boldsymbol{w}_i, \boldsymbol{x}')$, where $z_q(\boldsymbol{w}_i, \boldsymbol{x}_j) := \sqrt{p(\boldsymbol{w}_i)/q(\boldsymbol{w}_i)} z_p(\boldsymbol{w}_i, \boldsymbol{x}_j)$. Accordingly, the kernel matrix $\boldsymbol{K}$ can be approximated by $\boldsymbol{K}_q = \boldsymbol{Z}_q \boldsymbol{Z}_q^\top$, where $\boldsymbol{Z}_q := [\varphi_q(\boldsymbol{x}_1), \cdots, \varphi_q(\boldsymbol{x}_n)]^\top \in \mathbb{R}^{n\times s}$. Denoting by $\boldsymbol{z}_{q,\boldsymbol{w}_i}(\boldsymbol{X})$ the i-th column of $\boldsymbol{Z}_q$, we have $\boldsymbol{K} = \mathbb{E}_{\boldsymbol{w}\sim p}[\boldsymbol{z}_{p,\boldsymbol{w}}(\boldsymbol{X}) \boldsymbol{z}_{p,\boldsymbol{w}}^\top(\boldsymbol{X})] = \mathbb{E}_{\boldsymbol{w}\sim q}[\boldsymbol{z}_{q,\boldsymbol{w}}(\boldsymbol{X}) \boldsymbol{z}_{q,\boldsymbol{w}}^\top(\boldsymbol{X})]$. Note that this scheme can be regarded as a form of importance sampling.

Our surrogate empirical ridge leverage score distribution $L_\lambda(\boldsymbol{w})$ is given by Eq. (4). The integral of $L_\lambda(\boldsymbol{w})$ is

$$\int_{\mathbb{R}^d} L_\lambda(\boldsymbol{w}) \mathrm{d}\boldsymbol{w} = \frac{1}{n^2\lambda} \operatorname{Tr}\left[\left(\boldsymbol{y}\boldsymbol{y}^\top + n\boldsymbol{I}\right)\boldsymbol{K}\right] := D_{\boldsymbol{K}}^\lambda\,. \quad (10)$$

Combining Eq. (4) and Eq. (10), we can compute the surrogate empirical ridge leverage score distribution by

$$q(\boldsymbol{w}) := \frac{L_\lambda(\boldsymbol{w})}{\int_{\mathbb{R}^d} L_\lambda(\boldsymbol{w}) \mathrm{d}\boldsymbol{w}} = \frac{L_\lambda(\boldsymbol{w})}{D_{\boldsymbol{K}}^\lambda}\,. \quad (11)$$

The random features $\{\boldsymbol{w}_i\}_{i=1}^s$ can then be sampled from the above $q(\boldsymbol{w})$. We refer to this sampling strategy as *surrogate leverage weighted RFF*. Compared to the standard l_λ and its associated ERLS distribution, the proposed $L_\lambda(\boldsymbol{w})$ and $D_{\boldsymbol{K}}^\lambda$ are simpler: it does not require inverting the kernel matrix and thus accelerates the generation of random features.

Since the distribution $q(\boldsymbol{w})$ involves the kernel matrix $\boldsymbol{K}$ that is defined on the entire training dataset, we need to approximate $\boldsymbol{K}$ by random features, and then calculate/approximate $q(\boldsymbol{w})$. To be specific, we firstly sample $\{\boldsymbol{w}_i\}_{i=1}^l$ with $l \geq s$ from the spectral measure $p(\boldsymbol{w})$ and form the feature matrix $\boldsymbol{Z}_l \in \mathbb{R}^{n\times l}$. We have $\boldsymbol{K} \approx \tilde{\boldsymbol{K}}_p = \boldsymbol{Z}_l \boldsymbol{Z}_l^\top$, and thus the distribution $q(\boldsymbol{w})$ can be approximated by

$$\tilde{q}(\boldsymbol{w}) = \frac{p(\boldsymbol{w}) \boldsymbol{z}_{p,\boldsymbol{w}}^\top(\boldsymbol{X}) \left(\boldsymbol{y}\boldsymbol{y}^\top + n\boldsymbol{I}\right) \boldsymbol{z}_{p,\boldsymbol{w}}(\boldsymbol{X})}{\|\boldsymbol{y}^\top \boldsymbol{Z}_l\|_2^2 + n\|\boldsymbol{Z}_l\|_{\mathrm{F}}^2}\,. \quad (12)$$

Hence, we can then sample from $\tilde{q}(\boldsymbol{w})$ to generate the refined random features by importance sampling.

Note that the term $n\boldsymbol{I}$ in Eq. (4) and Eq. (12) is independent of the sample set $\boldsymbol{X}$. If we discard this term in our algorithm implementation, $L_\lambda(\boldsymbol{w})$ in Eq. (4) can be transformed as

$$L'_\lambda(\boldsymbol{w}) = p(\boldsymbol{w}) \boldsymbol{z}_{p,\boldsymbol{w}}^\top(\boldsymbol{X}) \left(\frac{1}{n^2\lambda} \boldsymbol{y}\boldsymbol{y}^\top\right) \boldsymbol{z}_{p,\boldsymbol{w}}(\boldsymbol{X})\,, \quad (13)$$

and further $\tilde{q}(\boldsymbol{w})$ in Eq. (12) can be simplified to

$$\tilde{q}'(\boldsymbol{w}) = \frac{p(\boldsymbol{w}) \boldsymbol{z}_{p,\boldsymbol{w}}^\top(\boldsymbol{X}) \left(\boldsymbol{y}\boldsymbol{y}^\top\right) \boldsymbol{z}_{p,\boldsymbol{w}}(\boldsymbol{X})}{\|\boldsymbol{y}^\top \boldsymbol{Z}_l\|_2^2}\,. \quad (14)$$

For each feature $\boldsymbol{w}_i \sim \tilde{q}'(\boldsymbol{w})$, its re-sampling probability p_i is associated with the approximate empirical ridge leverage score in Eq. (13). To be specific, it can be represented as

$$p_i \propto \left(\boldsymbol{y}^\top (\boldsymbol{Z}_l)_i\right)^2 = \left|\sum_{j=1}^n y_j z_p(\boldsymbol{w}_i, \boldsymbol{x}_j)\right|^2, \quad i = 1, 2, \cdots, l\,. \quad (15)$$

Algorithm 1: The Surrogate Leverage Weighted RFF Algorithm in KRR

Input: the training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, the shift-invariant kernel k, the number of random features s, and the regularization parameter λ

Output: the random feature mapping $\varphi(\cdot)$ and the optimization variable $\boldsymbol{\beta}_\lambda$ in KRR

1 Sample random features $\{\boldsymbol{w}_i\}_{i=1}^l$ from $p(\boldsymbol{w})$ with $l \geq s$, and form the feature matrix $\boldsymbol{Z}_l \in \mathbb{R}^{n\times l}$.

2 associate with each feature $\boldsymbol{w}_i$ a real number p_i such that p_i is proportional to

$$p_i \propto \left(\boldsymbol{y}^\top (\boldsymbol{Z}_l)_i\right)^2 , \quad i = 1, 2, \cdots, l\,.$$

3 Re-sample s features from $\{\boldsymbol{w}_i\}_{i=1}^l$ using the multinomial distribution given by the vector $(p_1/L, p_2/L, \cdots, p_l/L)$ with $L \leftarrow \sum_{i=1}^l p_i$.

4 Create the feature mapping $\varphi(\boldsymbol{x})$ for an example $\boldsymbol{x}$ by Eq. (9).

5 Obtain $\boldsymbol{\beta}_\lambda$ solved by Eq. (8).

It has intuitive physical meanings. From Eq. (15), it measures the correlation between the label y_j and the mapping output $z_p(\boldsymbol{w}_i, \boldsymbol{x}_j)$. Therefore, p_i quantifies the contribution of $\boldsymbol{w}_i$, which defines the i-th dimension of the feature mapping $\varphi(\cdot)$, for fitting the training data. In this view, if p_i is large, $\boldsymbol{w}_i$ is more important than the other features, and will be given the priority in the above importance sampling scheme. Based on this, we re-sample s features from $\{\boldsymbol{w}\}_{i=1}^l$ to generate the refined random features. Our surrogate leverage weighted RFF algorithm applied to KRR is summarized in Algorithm 1.

Also note that if the following condition holds

$$\frac{\boldsymbol{y}^\top \sum_{j=1}^n (\boldsymbol{Z}_s)_j (\boldsymbol{Z}_s)_j^\top \boldsymbol{y}}{\boldsymbol{y}^\top (\boldsymbol{Z}_s)_i (\boldsymbol{Z}_s)_i^\top \boldsymbol{y}} \approx \frac{\sum_{j=1}^n \|(\boldsymbol{Z}_s)_j\|_2^2}{\|(\boldsymbol{Z}_s)_i\|_2^2}\,,$$

then sampling from $\tilde{q}(\boldsymbol{w})$ or $\tilde{q}'(\boldsymbol{w})$ does not have distinct differences. This condition is satisfied if $\|(\boldsymbol{Z}_s)_i\|_2$ does not dramatically fluctuate for each column. in which sampling from $\tilde{q}(\boldsymbol{w})$ or $\tilde{q}'(\boldsymbol{w})$ may be used.

The method in (Li et al. 2019) samples $\{\boldsymbol{w}_i\}_{i=1}^s$ from $q^*(\boldsymbol{w}) := l_\lambda(\boldsymbol{w})/d_{\boldsymbol{K}}^\lambda$, while ours samples from $q(\boldsymbol{w}) := L_\lambda(\boldsymbol{w})/D_{\boldsymbol{K}}^\lambda$. In comparison, our surrogate ERLS distribution is much simpler as it avoids inverting the matrix $\boldsymbol{Z}_s^\top \boldsymbol{Z}_s$. Hence, generating s random features by Algorithm 1 takes $\mathcal{O}(ns^2)$ time to do the sampling. It is the same as the standard RFF and less than the $\mathcal{O}(ns^2 + s^3)$ time needed by (Li et al. 2019) which requires $\mathcal{O}(ns^2)$ for the multiplication of $\boldsymbol{Z}_s^\top \boldsymbol{Z}_s$ and $\mathcal{O}(s^3)$ for inverting $\boldsymbol{Z}_s^\top \boldsymbol{Z}_s$.

4 Theoretical Analysis

In this section, we analyze the generalization properties of kernel ridge regression when using random Fourier features sampled from our $q(\boldsymbol{w})$. Our analysis includes two parts. We first study how many features sampled from $q(\boldsymbol{w})$ are needed to incur no loss of learning accuracy in KRR. We then characterize the convergence rate of the expected risk of KRR when combined with Algorithm 1. Our proofs follow the framework in (Li et al. 2019) and in particular involve the same set of assumptions.

4.1 Expected risk for sampling from $q(\mathbf{w})$

The theorem below characterizes the relationship between the expected risk in KRR and the total number of random features used.

Theorem 1. *Given a shift-invariant and positive definite kernel function k, denote the eigenvalues of the kernel matrix $\boldsymbol{K}$ as $\lambda_1 \geq \cdots \geq \lambda_n$. Suppose that the regularization parameter λ satisfies $0 \leq n\lambda \leq \lambda_1$, $|y| \leq y_0$ is bounded with $y_0 > 0$, and $\{\boldsymbol{w}_i\}_{i=1}^s$ are sampled independently from the surrogate empirical ridge leverage score distribution $q(\boldsymbol{w}) = L_\lambda(\boldsymbol{w})/D_{\boldsymbol{K}}^\lambda$. If the unit ball of $\mathcal{H}$ contains the optimal hypothesis f_ρ and*

$$s \geq 5 D_{\boldsymbol{K}}^\lambda \log\left(16 d_{\boldsymbol{K}}^\lambda\right)/\delta\,,$$

then for $0 < \delta < 1$, with probability $1-\delta$, the excess risk of $f_{\boldsymbol{\beta}}^\lambda$ can be upper bounded as

$$\mathcal{E}\left(f_{\boldsymbol{\beta}}^\lambda\right) - \mathcal{E}\left(f_\rho\right) \leq 2\lambda + \mathcal{O}(1/\sqrt{n}) + \mathcal{E}\left(\hat{f}^\lambda\right) - \mathcal{E}\left(f_\rho\right)\,, \quad (16)$$

where $\mathcal{E}\left(\hat{f}^\lambda\right) - \mathcal{E}\left(f_\rho\right)$ is the excess risk for the standard kernel ridge regression estimator.

Theorem 1 shows that if the total number of random features sampled from $q(\boldsymbol{w})$ satisfies $s \geq 5 D_{\boldsymbol{K}}^\lambda \log\left(16 d_{\boldsymbol{K}}^\lambda\right)/\delta$, we incur no loss in the learning accuracy of kernel ridge regression. In particular, with the standard choice $\lambda = \mathcal{O}(n^{-1/2})$, the estimator $f_{\boldsymbol{\beta}}^\lambda$ attains the minimax rate of kernel ridge regression.

To illustrate the lower bound in Theorem 1 on the number of features, we consider three cases regarding the eigenvalue decay of $\boldsymbol{K}$: i) the exponential decay $\lambda_i \propto ne^{-ci}$ with $c > 0$, ii) the polynomial decay $\lambda_i \propto ni^{-2t}$ with $t \geq 1$, and iii) the slowest decay with $\lambda_i \propto n/i$ (see (Bach 2013) for details). In all three cases, direct calculation shows

$$D_{\boldsymbol{K}}^\lambda = \frac{1}{n^2\lambda} \operatorname{Tr}\left[(\boldsymbol{y}\boldsymbol{y}^\top + n\boldsymbol{I})\boldsymbol{K}\right] \leq \frac{2}{n\lambda} \operatorname{Tr}(\boldsymbol{K}) \in \mathcal{O}(\sqrt{n})\,.$$

Moreover, $d_{\boldsymbol{K}}^\lambda$ satisfies $d_{\boldsymbol{K}}^\lambda \in \mathcal{O}(\log n)$ in the exponential decay case, $d_{\boldsymbol{K}}^\lambda \in \mathcal{O}(n^{1/(4t)})$ in the polynomial decay case, and $d_{\boldsymbol{K}}^\lambda \in \mathcal{O}(\sqrt{n})$ in the slowest case. Combining these bounds gives the number s of random features sufficient for no loss in the learning accuracy of KRR; these results are reported in Tab. 1. It can be seen that sampling from $q^*(\boldsymbol{w})$ (Li et al. 2019) sometimes requires fewer random features than our method. This is actually reasonable as the design of our surrogate ERLS distribution follows in a simple fashion and we directly relax D_K^λ to $\mathcal{O}(\sqrt{n})$. It does not strictly follow with the continuous generalization of the leverage scores used in the analysis of linear methods (Alaoui and Mahoney 2015; Cohen, Musco, and Musco 2017; Avron et al. 2017). Actually, with a more careful argument, this bound can be further improved and made tight, which we leave to future works. Nevertheless, our theoretical analysis actually provides the worst case estimation for the lower bound of

Table 1: Comparisons of the number s of features required by two sampling schemes.

Eigenvalue decay	(Li et al. 2019)	Ours
$\lambda_i \propto ne^{-ci}, c > 0$	$s \geq \log^2 n$	$s \geq \sqrt{n}\log\log n$
$\lambda_i \propto ni^{-2t}, t \geq 1$	$s \geq n^{1/(4t)}\log n$	$s \geq \sqrt{n}\log n$
$\lambda_i \propto n/i$	$s \geq \sqrt{n}\log n$	$s \geq \sqrt{n}\log n$

s. In practical uses, our algorithm would not require the considerable number of random features to achieve a good prediction performance. Specifically, our experimental results in Section 5 demonstrate that when using the same s, there is no distinct difference between (Li et al. 2019) and our method in terms of prediction performance. But our approach costs less time to generate the refined random features, achieving a substantial computational saving when the total number of random features is relatively large.

To prove Theorem 1, we need the following lemma.

Lemma 1. *Under the same assumptions from Theorem 1, let* $\epsilon \geq \sqrt{m/s} + 2L/3s$ *with constants* m *and* L *given by*

$$m := D^{\lambda}_{\boldsymbol{K}} \frac{\lambda_1}{\lambda_1 + n\lambda} \quad L := \sup_i \frac{l_\lambda(\boldsymbol{w}_i)}{q(\boldsymbol{w}_i)},\ \forall i = 1, 2, \cdots, s\,.$$

If the number of random features satisfies

$$s \geq D^{\lambda}_{\boldsymbol{K}} \left(\frac{1}{\epsilon^2} + \frac{2}{3\epsilon}\right) \log \frac{16 d^{\lambda}_{\boldsymbol{K}}}{\delta}\,, \tag{17}$$

then for $0 < \delta < 1$*, with probability* $1 - \delta$*, we have*

$$-\epsilon \boldsymbol{I} \preceq (\boldsymbol{K}+n\lambda\boldsymbol{I})^{-\frac{1}{2}}(\tilde{\boldsymbol{K}}_q - \boldsymbol{K})(\boldsymbol{K}+n\lambda\boldsymbol{I})^{-\frac{1}{2}} \preceq \epsilon \boldsymbol{I}\,. \tag{18}$$

Proof. Following the proof of Lemma 4 in (Li et al. 2019), by the matrix Bernstein concentration inequality (Tropp and others 2015) and $l_\lambda(\boldsymbol{w}) \leq L_\lambda(\boldsymbol{w})$, we conclude the proof. □

Based on Lemma 1, as well as the previous results including Lemma 2, Lemma 5, Lemma 6, Theorem 5 in (Li et al. 2019), we can immediately prove Theorem 1.

4.2 Expected risk for Algorithm 1

In the above analysis, our results are based on the random features $\{\boldsymbol{w}_i\}_{i=1}^s$ sampled from $q(\boldsymbol{w})$. In Algorithm 1, $\{\boldsymbol{w}_i\}_{i=1}^s$ are actually drawn from $\tilde{q}(\boldsymbol{w})$ or $\tilde{q}'(\boldsymbol{w})$. In this section, we present the convergence rates for the expected risk of Algorithm 1.

Theorem 2. *Under the same assumptions from Theorem 1, denote by* $\tilde{f}^{\lambda^*}$ *the KRR estimator obtained using a regularization parameter* λ^* *and the features* $\{\boldsymbol{w}_i\}_{i=1}^s$ *sampled via Algorithm 1. If the number of random features satisfies*

$$s \geq \max\left\{\frac{7z_0^2 \log\left(16 d^{\lambda}_{\boldsymbol{K}}\right)}{\lambda\delta}, 5D^{\lambda^*}_{\boldsymbol{K}} \frac{\log\left(16 d^{\lambda^*}_{\boldsymbol{K}}\right)}{\delta}\right\},$$

with $|z_p(\boldsymbol{w}, \boldsymbol{x})| < z_0$*, then for* $0 < \delta < 1$*, with probability* $1 - \delta$*, the excess risk of* $\tilde{f}^{\lambda^*}$ *can be estimated by*

$$\mathcal{E}(\tilde{f}^{\lambda^*}) - \mathcal{E}\left(f_\rho\right) \leq 2\lambda + 2\lambda^* + \mathcal{O}(1/\sqrt{n})\,. \tag{19}$$

Proof. According to Theorem 1 and Corollary 2 in (Li et al. 2019), if the number of random features satisfies $s \geq 7z_0^2 \log\left(16 d^{\lambda}_{\boldsymbol{K}}\right)/(\lambda\delta)$, then for any $0 < \delta < 1$, with confidence $1 - \delta$, the excess risk of $f^{\lambda}_{\boldsymbol{\alpha}}$ can be bounded by

$$\mathcal{E}(f^{\lambda}_{\boldsymbol{\alpha}}) - \mathcal{E}\left(f_\rho\right) \leq 2\lambda + \mathcal{O}(1/\sqrt{n})\,. \tag{20}$$

Let $f_{\tilde{\mathcal{H}}}$ be the function in $\tilde{\mathcal{H}}$ spanned by the approximated kernel that achieves the minimal risk, i.e., $\mathcal{E}(f_{\tilde{\mathcal{H}}}) = \inf_{f \in \tilde{\mathcal{H}}} \mathcal{E}(f)$. Hence, we re-sample $\{\boldsymbol{w}_i\}_{i=1}^s$ according to $q(\boldsymbol{w})$ as defined in Eq. (11), in which the kernel matrix is indicated by the actual kernel $\tilde{k}$ spanned in $\tilde{\mathcal{H}}$. Denote our KRR estimator with the regularization parameter λ^* and the learning function $\tilde{f}^{\lambda^*}$, and according to Theorem 1, if the number of random features s satisfies $s \geq 5D^{\lambda^*}_{\boldsymbol{K}} \frac{\log\left(16 d^{\lambda^*}_{\boldsymbol{K}}\right)}{\delta}$, then for $0 < \delta < 1$, with confidence $1 - \delta$, the excess risk of $\tilde{f}^{\lambda^*}$ can be estimated by

$$\mathcal{E}\left(\tilde{f}^{\lambda^*}\right) - \mathcal{E}\left(f_{\tilde{\mathcal{H}}}\right) \leq 2\lambda^* + \mathcal{O}(1/\sqrt{n})\,. \tag{21}$$

Since $f_{\tilde{\mathcal{H}}}$ achieves the minimal risk over $\mathcal{H}$, we can conclude that $\mathcal{E}(f_{\tilde{\mathcal{H}}}) \leq \mathcal{E}(f^{\lambda}_{\boldsymbol{\alpha}})$. Combining Eq. (20) and Eq. (21), we obtain the final excess risk of $\mathcal{E}(\tilde{f}^{\lambda^*})$. □

Theorem 2 provides the upper bound of the expected risk in KRR estimator over random features generated by Algorithm 1. Note that, in our implementation, the number of random features used to approximate the kernel matrix is also set to s for simplicity, which shares the similar way with the implementation in (Li et al. 2019).

5 Experiments

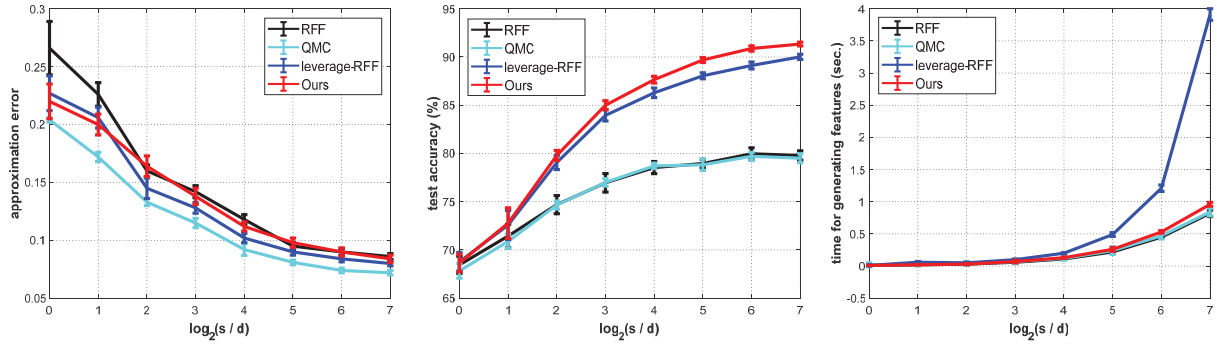
In this section, we empirically evaluate the performance of our method equipped with KRR for classification tasks on several benchmark datasets. All experiments are implemented in MATLAB and carried out on a PC with Intel® i5-6500 CPU (3.20 GHz) and 16 GB RAM. The source code of our implementation can be found in http://www.lfhsgre.org.

5.1 Experimental settings

We choose the popular shift-invariant Gaussian/RBF kernel for experimental validation, i.e., $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2/\sigma^2)$. Following (Avron et al. 2017), we use a fixed bandwidth $\sigma = 1$ in our experiments. This is without loss of generality since we can rescale the points and adjust the bounding interval. The regularization parameter λ is tuned via 5-fold inner cross validation over a grid of $\{0.05, 0.1, 0.5, 1\}$.

Datasets: We consider four classification datasets including *EEG*, *cod-RNA*, *covtype* and *magic04*; see Tab. 2 for an overview of these datasets. These datasets can be downloaded from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/ or the UCI Machine Learning Repository[2]. All datasets are normalized to $[0, 1]^d$ before the experiments. We use the given training/test partitions on the *cod-RNA* dataset. For the other three datasets, we randomly pick half of the data for training and the rest for testing. All experiments

[2]https://archive.ics.uci.edu/ml/datasets.html.



(a) Relative error (b) Test accuracy (c) Time cost

Figure 1: Comparison of various algorithms on approximation error in (a), test accuracy in (b), and time cost for generating random features in (c) versus the number of random features s on the *EEG* dataset.

are repeated 10 times and we report the average classification accuracy as well as the time cost for generating random features.

Table 2: Dataset statistics.

datasets	#feature dimension	#traing examples	#test examples
EEG	14	7,490	7,490
cod-RNA	8	59,535	157,413
covtype	54	290,506	290,506
magic04	10	9510	9510

Compared methods: We compare the proposed surrogate leverage weighted sampling strategy with the following three random features based algorithms:

- RFF (Rahimi and Recht 2007): The feature mapping $\varphi_p(\boldsymbol{x})$ is given by Eq. (2), in which the random features $\{\boldsymbol{w}_i\}_{i=1}^s$ are sampled from $p(\boldsymbol{w})$.
- QMC (Avron et al. 2016): The feature mapping $\varphi_p(\boldsymbol{x})$ is also given by Eq. (2), but the random features $\{\boldsymbol{w}_i\}_{i=1}^s$ are constructed by a deterministic scheme, e.g., a low-discrepancy Halton sequence.
- leverage-RFF (Li et al. 2019): The data-dependent random features $\{\boldsymbol{w}_i\}_{i=1}^s$ are sampled from $q^*(\boldsymbol{w})$. The kernel matrix in $q^*(\boldsymbol{w})$ is approximated using random features pre-sampled from $p(\boldsymbol{w})$.

In our method, we consider sampling from $\tilde{q}'(\boldsymbol{w})$ in Algorithm 1 for simplicity.

5.2 Comparison results

High-level comparison: We compare the empirical performance of the aforementioned random features mapping based algorithms. In Fig. 1, we consider the *EEG* dataset and plot the relative kernel matrix approximation error, the test classification accuracy and the time cost for generating random features versus different values of s. Note that since we cannot compute the kernel matrix $\boldsymbol{K}$ on the entire dataset, we randomly sample 1,000 datapoints to construct the feature matrix $\boldsymbol{Z}_s\boldsymbol{Z}_s^\top$, and then calculate the relative approximation error, i.e., $err = \frac{\|\boldsymbol{K}-\boldsymbol{Z}_s\boldsymbol{Z}_s^\top\|_2}{\|\boldsymbol{K}\|_2}$.

Fig. 1(a) shows the mean of the approximation errors across 10 trials (with one standard deviation denoted by error bars) under different random features dimensionality. We find that QMC achieves the best approximation performance when compared to RFF, leverage-RFF, and our proposed method. Fig. 1(b) shows the test classification accuracy. We find that as the number of random features increases, leverage-RFF and our method significantly outperform RFF and QMC .

From the above experimental results, we find that, admittedly, QMC achieves lower approximation error to some extent, but it does not translate to better classification performance when compared to leverage-RFF and our method. The reason may be that the original kernel derived by the point-wise distance might not be suitable, and the approximated kernel is not optimal for classification/regression tasks, as discussed in (Avron et al. 2017; Munkhoeva et al. 2018; Zhang et al. 2019). As the ultimate goal of kernel approximation is to achieve better prediction performance, in the sequel we omit the approximation performance of these methods.

In terms of time cost for generating random features, Fig. 1(c) shows that leverage-RFF is quite time-consuming when the total number of random features is large. In contrast, our algorithm achieves comparable computational efficiency with RFF and QMC. These results demonstrate the superiority of our surrogate weighted sampling strategy, which reduces the time cost.

Detailed comparisons: Tab. 3 reports the detailed classification accuracy and time cost for generating random features of all algorithms on the four datasets. Observe that by using a *data-dependent* sampling strategy, leverage-RFF and our method achieve better classification accuracy than RFF and QMC on the *EEG* and *cod-RNA* dataset when the dimensionality of random features increases. In particular, on the *EEG* dataset, when s ranges from $2d$ to $128d$, the test accuracy of leverage-RFF and our method is better than RFF and QMC by around 1% to nearly 11%. On the *cod-RNA* dataset, the performance of RFF and QMC is worse than our method

Table 3: Comparison results of various algorithms for varying s in terms of classification accuracy (mean±std. deviation %) and time cost for generating random features (mean±std. deviation sec.). The higher test accuracy means better. Notation "•" indicates that leverage-RFF and our method are significantly better than the other two baseline methods via paired t-test.

Dataset	s	RFF	QMC	leverage-RFF	Ours
		Acc:% (time:sec.)	Acc:% (time:sec.)	Acc:% (time:sec.)	Acc:% (time:sec.)
EEG	d	68.45±0.89 (0.01±0.00)	67.83±0.73 (0.02±0.03)	68.78±0.97 (0.01±0.01)	68.62±0.89 (0.01±0.00)
	$2d$	71.44±1.22 (0.02±0.00)	70.89±0.72 (0.03±0.03)	72.59±1.51 (0.03±0.01)	72.72±1.35 (0.02±0.00)
	$4d$	74.70±0.94 (0.03±0.01)	74.66±0.42 (0.04±0.03)	79.06±0.73 (0.05±0.01)•	79.72±0.58 (0.03±0.01)•
	$8d$	76.96±0.96 (0.06±0.02)	77.01±0.40 (0.07±0.03)	83.95±0.58 (0.10±0.01)•	84.97±0.50 (0.07±0.02)•
	$16d$	78.54±0.63 (0.11±0.00)	78.71±0.29 (0.12±0.03)	86.29±0.50 (0.20±0.01)•	87.23±0.41 (0.13±0.01)•
	$32d$	78.96±0.44 (0.22±0.02)	78.83±0.59 (0.24±0.06)	88.05±0.31 (0.49±0.03)•	89.38±0.32 (0.26±0.03)•
	$64d$	79.97±0.62 (0.45±0.01)	79.71±0.40 (0.47±0.05)	89.12±0.36 (1.21±0.05)•	90.36±0.31 (0.53±0.02)•
	$128d$	79.79±0.49 (0.82±0.05)	79.51±0.47 (0.84±0.06)	90.01±0.27 (3.91±0.09)•	91.02±0.32 (0.96±0.03)•
cod-RNA	d	87.02±0.29 (0.06±0.01)	87.20±0.00 (0.07±0.03)	88.62±0.92 (0.09±0.02)	89.64±0.87 (0.07±0.01)•
	$2d$	87.12±0.19 (0.12±0.01)	87.65±0.00 (0.16±0.02)	90.42±1.15 (0.17±0.01)•	90.12±0.95 (0.13±0.01)•
	$4d$	87.19±0.08 (0.24±0.01)	87.44±0.00 (0.25±0.02)	92.65±0.38 (0.35±0.02)•	92.83±0.33 (0.27±0.01)•
	$8d$	87.27±0.11 (0.47±0.02)	87.29±0.00 (0.49±0.02)	93.41±0.07 (0.69±0.02)•	93.49±0.15 (0.53±0.02)•
	$16d$	87.29±0.08 (0.91±0.02)	87.30±0.00 (0.94±0.04)	93.71±0.06 (1.39±0.05)•	93.74±0.05 (0.99±0.02)•
	$32d$	87.27±0.05 (1.80±0.02)	87.33±0.00 (1.77±0.01)	93.76±0.02 (2.82±0.08)•	93.71±0.07 (1.95±0.03)•
	$64d$	87.30±0.03 (3.48±0.15)	87.32±0.00 (3.54±0.10)	93.73±0.03 (6.54±0.53)•	93.99±0.06 (4.05±0.08)•
	$128d$	87.30±0.03 (6.79±0.39)	87.32±0.00 (6.62±0.08)	93.66±0.03 (13.3±0.23)•	93.48±0.04 (7.78±0.09)•
covtype[1]	d	73.70±0.79 (1.90±0.03)	74.71±0.07 (1.88±0.11)	73.99±0.85 (2.96±0.09)	73.99±0.63 (2.00±0.05)
	$2d$	77.09±0.25 (3.31±0.21)	77.04±0.06 (3.37±0.29)	77.04±0.35 (5.25±0.09)	77.02±0.29 (3.44±0.10)
	$4d$	79.10±0.13 (6.27±0.35)	79.07±0.07 (6.12±0.17)	79.18±0.17 (10.2±0.15)	79.05±0.14 (6.58±0.19)
	$8d$	81.04±0.12 (12.3±0.71)	80.90±0.05 (12.1±0.45)	81.09±0.07 (21.1±0.72)	80.79±0.11 (13.2±0.34)
	$16d$	82.42±0.10 (24.5±1.02)	82.37±0.07 (24.3±1.56)	82.90±0.12 (46.5±2.20)	82.18±0.10 (28.6±1.58)
magic04	d	73.62±0.68 (0.01±0.00)	71.74±0.40 (0.02±0.04)	73.62±0.68 (0.01±0.01)	73.61±0.68 (0.01±0.00)
	$2d$	75.89±0.80 (0.01±0.01)	75.98±0.36 (0.03±0.03)	75.91±0.77 (0.03±0.01)	75.88±0.77 (0.02±0.00)
	$4d$	77.78±0.45 (0.03±0.01)	77.27±0.33 (0.04±0.03)	77.78±0.45 (0.05±0.01)	77.77±0.43 (0.03±0.00)
	$8d$	78.97±0.34 (0.05±0.00)	79.07±0.17 (0.07±0.03)	79.15±0.40 (0.09±0.01)	79.12±0.34 (0.06±0.01)
	$16d$	80.04±0.34 (0.10±0.01)	79.95±0.37 (0.11±0.03)	80.80±0.40 (0.19±0.01)	80.74±0.42 (0.11±0.01)
	$32d$	80.61±0.43 (0.19±0.01)	80.65±0.31 (0.21±0.04)	82.00±0.32 (0.41±0.03)•	82.02±0.32 (0.22±0.01)•
	$64d$	80.91±0.28 (0.38±0.03)	80.85±0.27 (0.41±0.05)	82.39±0.32 (0.93±0.05)•	82.37±0.25 (0.44±0.03)•
	$128d$	81.10±0.37 (0.73±0.03)	81.08±0.29 (0.76±0.04)	82.59±0.29 (2.61±0.15)•	82.55±0.55 (0.87±0.02)•

[1] Due to the memory limit, we cannot conduct the experiment on the *covtype* dataset when $s \geq 32d$.

by over 6% when $s \geq 4d$. On the *covtype* dataset, all four methods achieve similar the classification accuracy. Instead, on the *magic04* dataset, our algorithm and leverage-RFF perform better than RFF and QMC on the final classification accuracy if more random features are considered.

In terms of computational efficiency on these four datasets, albeit *data-dependent*, our method still takes about the similar time cost with the *data-independent* RFF and QMC for generating random features. Specifically, when compared to leverage-RFF, our method achieves a substantial computational saving.

6 Conclusion

In this work, we have proposed an effective *data-dependent* sampling strategy for generating fast random features for kernel approximation. Our method can significantly improve the generalization performance while achieving the same time complexity when compared to the standard RFF. Our theoretical results and experimental validation have demonstrated the superiority of our method when compared to other representative random Fourier features based algorithms on several classification benchmark datasets.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (No.61572315, 61876107, 61977046), in part by the National Key Research and Development Project (No. 2018AAA0100702), in part by National Science Foundation grants CCF-1657420 and CCF-1704828, in part by the European Research Council under the European Union's Horizon 2020 research and innovation program / ERC Advanced Grant E-DUALITY (787960). This paper reflects only the authors' views and the Union is not liable for any use that may be made of the contained information; Research Council KUL C14/18/068; Flemish Government FWO project GOA4917N; Onderzoeksprogramma Artificiele Intelligentie (AI) Vlaanderen programme. Jie Yang and Xiaolin Huang are corresponding authors.

References

Agrawal, R.; Campbell, T.; Huggins, J.; and Broderick, T. 2019. Data-dependent compression of random features for large-scale kernel approximation. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, 1822–1831.

Alaoui, A., and Mahoney, M. W. 2015. Fast randomized

kernel ridge regression with statistical guarantees. In *Proceedings of Advances in Neural Information Processing Systems*, 775–783.

Avron, H.; Sindhwani, V.; Yang, J.; and Mahoney, M. W. 2016. Quasi-Monte Carlo feature maps for shift-invariant kernels. *Journal of Machine Learning Research* 17(1):4096–4133.

Avron, H.; Kapralov, M.; Musco, C.; Musco, C.; Velingker, A.; and Zandieh, A. 2017. Random Fourier features for kernel ridge regression: Approximation bounds and statistical guarantees. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 253–262.

Bach, F. 2013. Sharp analysis of low-rank kernel matrix approximations. In *Proceedings of Conference on Learning Theory*, 185–209.

Bach, F. 2017. On the equivalence between kernel quadrature rules and random feature expansions. *Journal of Machine Learning Research* 18(1):714–751.

Bochner, S. 2005. *Harmonic Analysis and the Theory of Probability*. Courier Corporation.

Cohen, M. B.; Musco, C.; and Musco, C. 2017. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1758–1777.

Cortes, C.; Mohri, M.; and Rostamizadeh, A. 2012. Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research* 13(2):795–828.

Dao, T.; De Sa, C. M.; and Ré, C. 2017. Gaussian quadrature for kernel features. In *Proceedings of Advances in neural information processing systems*, 6107–6117.

Evans, G. 1993. *Practical numerical integration*. Wiley New York.

Fanuel, M.; Schreurs, J.; and Suykens, J. A. 2019. Nyström landmark sampling and regularized Christoffel functions. *arXiv preprint arXiv:1905.12346*.

Gauthier, B., and Suykens, J. 2018. Optimal quadrature-sparsification for integral operator approximation. *SIAM Journal on Scientific Computing* 40(5):A3636–A3674.

Li, Z.; Ton, J.-F.; Oglic, D.; and Sejdinovic, D. 2019. Towards a unified analysis of random Fourier features. In *Proceedings of the 36th International Conference on Machine Learning*, 3905–3914.

Mall, R., and Suykens, J. A. 2015. Very sparse lssvm reductions for large-scale data. *IEEE Transactions on Neural Networks and Learning Systems* 26(5):1086–1097.

Munkhoeva, M.; Kapushev, Y.; Burnaev, E.; and Oseledets, I. 2018. Quadrature-based features for kernel approximation. In *Proceedings of Advances in Neural Information Processing Systems*, 9165–9174.

Niederreiter, H. 1992. *Random number generation and quasi-Monte Carlo methods*, volume 63. SIAM.

Rahimi, A., and Recht, B. 2007. Random features for large-scale kernel machines. In *Proceedings of Advances in Neural Information Processing Systems*, 1177–1184.

Rudi, A.; Camoriano, R.; and Rosasco, L. 2017. Generalization properties of learning with random features. In *Proceedings of Advances in Neural Information Processing Systems*, 3215–3225.

Schölkopf, B., and Smola, A. J. 2003. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT Press.

Sinha, A., and Duchi, J. C. 2016. Learning kernels with random features. In *Proceedins of Advances in Neural Information Processing Systems*. 1298–1306.

Sun, Y.; Gilbert, A.; and Tewari, A. 2018. But how does it work in theory? Linear SVM with random features. In *Proceedings of Advances in Neural Information Processing Systems*, 3383–3392.

Suykens, J. A.; Van Gestel, T.; De Brabanter, J.; De Moor, B.; and Vandewalle, J. 2002. *Least Squares Support Vector Machines*. World Scientific.

Tropp, J. A., et al. 2015. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning* 8(1-2):1–230.

Yu, F. X.; Suresh, A. T.; Choromanski, K.; Holtmannrice, D.; and Kumar, S. 2016. Orthogonal random features. In *Proceedings of Advances in Neural Information Processing Systems*, 1975–1983.

Zhang, J.; May, A.; Dao, T.; and Re, C. 2019. Low-precision random fourier features for memory-constrained kernel approximation. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, 1264–1274.