The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

On Data-Dependent Random Features for Improved Generalization in Supervised Learning

Shahin Shahrampour, Ahmad Beirami, Vahid Tarokh
School of Engineering and Applied Sciences, Harvard University
Cambridge, MA, 02138 USA

Abstract

The randomized-feature approach has been successfully employed in large-scale kernel approximation and supervised learning. The distribution from which the random features are drawn impacts the number of features required to efficiently perform a learning task. Recently, it has been shown that employing data-dependent randomization improves the performance in terms of the required number of random features. In this paper, we are concerned with the randomized-feature approach in supervised learning for good generalizability. We propose the Energy-based Exploration of Random Features (EERF) algorithm based on a data-dependent score function that explores the set of possible features and exploits the promising regions. We prove that the proposed score function with high probability recovers the spectrum of the best fit within the model class. Our empirical results on several benchmark datasets further verify that our method requires smaller number of random features to achieve a certain generalization error compared to the state-of-the-art while introducing negligible pre-processing overhead. EERF can be implemented in a few lines of code and requires no additional tuning parameters.

Introduction

At the heart of many machine learning problems, kernel methods (such as Support Vector Machine (SVM) (Cristianini and Shawe-Taylor 2000)) describe the nonlinear representation of data via mapping the features to a high-dimensional feature space. Even without access to the explicit form of the *feature maps*, one can still compute their inner products inexpensively using a kernel function, an idea known as the “kernel trick”. However, unfortunately, methods using kernel matrices are not applicable to large-scale machine learning as they incur a prohibitive computational cost scaling at least quadratically with data. This observation motivated (Rahimi and Recht 2007) to consider kernel approximation using *random features*, and extend the idea to train shallow architectures (Rahimi and Recht 2009). Replacing the optimization of nonlinearities by randomization, randomized shallow networks efficiently approximate the function describing the input-output relationship via random features. Nevertheless, a natural concern is the stochastic oracle from which the features are sampled. As noted in (Yang et al. 2012), the basis functions used by random Fourier features (Rahimi and Recht 2009) are sampled from a distribution that is *independent* of the training set, and hence, a large number of random features may be needed to learn the data subspace. Therefore, one can ask whether *data-dependent* sampling can improve the prediction accuracy in supervised learning.

Recently, (Sinha and Duchi 2016) proposed a data-dependent sampling scheme using an optimization perspective that can reduce the number of random features needed for effective learning. The generalization performance of their method, however, relies on the regularization parameter of the optimization problem, which requires an extra level of tuning (e.g., using cross-validation).

In this paper, within the framework of supervised learning, we develop a data-dependent sampling method, called Energy-based Exploration of Random Features (EERF), with the goal of better generalizability. Our algorithm operates based on a score function that is defined with respect to (a subsample of) the training samples. The algorithm explores the domain of random features, evaluates the score function in different regions, and outputs the promising random features for generalization. We prove that the score function mimics the spectrum of the best fit within the model class with high probability. We further apply our results to practical datasets, where we observe that our algorithm learns the subspace faster than the state-of-the-art as a function of the number of random features. Notably, our algorithm does not require additional parameter tuning.

Related literature on random features: Some previous works on random features have focused on kernel approximation as well as prediction in a supervised manner. It has been shown that a wide variety of kernels can be approximated efficiently using random features. Examples include shift-invariant kernels using Monte Carlo (Rahimi and Recht 2007) and Quasi Monte Carlo (Yang et al. 2014a) sampling, polynomial kernels (Kar and Karnick 2012), additive kernels (Vedaldi and Zisserman 2012), and many more. In particular, Gaussian kernel has received considerable attention (see e.g. (Felix et al. 2016) for a recent study on efficient Gaussian kernel approximation), and the error of random Fourier features has been analyzed in the context of kernel approximation (Sutherland and Schneider 2015). Additionally, the generalization property of the randomized-feature approach has been theoretically studied from the statistical learning theory viewpoint (Rudi, Camoriano, and Rosasco 2016).

Another line of research has focused on decreasing the time and space complexity of kernel approximation. In (Le, Sarlós, and Smola 2013), the Fast-food method has been developed to approximate kernel expansions in log-linear time. The underlying idea is that Hadamard matrices combined with diagonal Gaussian matrices exhibit properties similar to dense Gaussian random matrices. Using this approach a class of flexible kernels has been proposed in (Yang et al. 2015).

Data-dependent random features have been recently studied in (Yu et al. 2015; Oliva et al. 2016; Chang et al. 2017) for approximation of shift-invariant kernels. We consider a broader class of kernels (see Eq. (3)) and propose a sampling scheme that improves the generalization error, particularly when the number of random features is small. Our work is particularly relevant to that of (Sinha and Duchi 2016), where a data-dependent optimization approach is developed to sample features with promising generalization error for small to moderate number of bases. Their generalization performance relies on the regularization parameter in their optimization problem, which requires an extra level of tuning (e.g., using cross-validation). Our method, in contrast, does not need additional parameter tuning. Finally, the details of the benchmark algorithms used for comparison in this paper can be found in Table 1 in the empirical evaluations section.

Nyström method: We remark that Nyström method (Williams and Seeger 2001; Drineas and Mahoney 2005) adopts an alternative viewpoint for approximation of kernel by a low rank matrix. The method samples a subset of training data, approximates a kernel matrix, and then transforms the data using the approximated kernel. We refer the reader to (Yang et al. 2012) for a discussion on the fundamental differences between the Nyström method and random features.

A note on multiple kernel learning (MKL): The goal of MKL is to learn a good kernel based on training data (see e.g. (Gönen and Alpaydın 2011) for a survey). For the supervised learning setup, various methods consider optimizing a convex, linear, or nonlinear combination of base kernels with respect to a measure (e.g. kernel alignment) to identify the ideal kernel (Kandola, Shawe-Taylor, and Cristianini 2002; Cortes, Mohri, and Rostamizadeh 2009; 2012). Another approach is to optimize the kernel and the empirical risk simultaneously (Kloft et al. 2011; Lanckriet et al. 2004). These methods enjoy various theoretical guarantees (Bartlett and Mendelson 2002; Cortes, Mohri, and Rostamizadeh 2010), but they involve costly computational steps, such as eigen-decomposition of the Gram matrix (see (Gönen and Alpaydın 2011) for details). The distinction of our work with this literature is that we do not consider a combination of base kernels. Instead, we propose to use the randomized-feature approach of (Rahimi and Recht 2009; 2007) with a data-dependent sampling scheme to avoid computational cost.

Notation: We denote by $\mathcal{N}(\mu, \sigma^2)$ the Gaussian distribution with mean μ and variance σ^2, by I_d the identity matrix of size d, and by $[N]$ the set of positive integers $\{1, \dots, N\}$, respectively. w_i is the i-th component of the vector w, whereas z^m is m-th sample among the batch $\{z^j\}_{j=1}^M$.

Problem Setup

Consider the supervised learning setup: we are given a training *input-output* set $\{(x^n, y^n)\}_{n=1}^N$, where the pairs are generated independently from an *unknown, fixed* distribution $P_{\mathcal{X}\mathcal{Y}}$, and for every $n \in [N]$, $x^n = [x_1^n \cdots x_d^n]^\top \in \mathcal{X} \subset \mathbb{R}^d$. For regression, the response variable $y^n \in \mathcal{Y} \subseteq [-1, 1]$, while for classification $y^n \in \{-1, 1\}$. The goal is to fit a function $f : \mathcal{X} \to \mathbb{R}$ to training data via risk minimization. As $P_{\mathcal{X}\mathcal{Y}}$ is not available, we consider minimizing the empirical risk $\widehat{\mathbf{R}}(f)$ in lieu of $\mathbf{R}(f)$,

$$\mathbf{R}(f) \triangleq \mathbb{E}_{P_{\mathcal{X}\mathcal{Y}}}[c(f(x), y)], \quad \widehat{\mathbf{R}}(f) \triangleq \frac{1}{N} \sum_{n=1}^N c(f(x^n), y^n), \tag{1}$$

where $c(\cdot, \cdot)$ is a task-dependent loss function (e.g., quadratic for regression, hinge loss for SVM), measuring the dissimilarity of the mapping $f(x)$ and the output y over training samples. In general, one often parametrizes the function $f(\cdot)$ to minimize $\widehat{\mathbf{R}}(f)$ over a parameter space. Kernel methods offer such solutions where $f(x) \approx \sum_{n=1}^N \alpha_n k(x^n, x)$, and the empirical risk $\widehat{\mathbf{R}}(f)$ is minimized over $\{\alpha_n\}_{n=1}^N$. However, an immediate drawback of this approach is its inapplicability to large-scale data, since the computational complexity scales at least with $O(N^2)$ for training the kernel matrix. To overcome this shortcoming, an elegant approach was proposed by (Rahimi and Recht 2009), where shallow networks are parametrized using *random features*. Let us represent a feature map by $\phi : \mathcal{X} \times \Omega \to \mathbb{R}$, where Ω is the support set for random features. Then, considering functions of the form

$$f(x) = \int_\Omega F(\omega) \phi(x, \omega) d_\omega, \tag{2}$$

one can approximate $f(x) \approx \sum_{m=1}^M \theta_m \phi(x, \omega^m)$ and minimize $\widehat{\mathbf{R}}(f)$ over $\{\theta_m, \omega^m\}_{m=1}^M$, where M is (hopefully) much smaller than N. The main issue would be the joint optimization of $\{\theta_m, \omega^m\}_{m=1}^M$, which results in a non-convex problem. (Rahimi and Recht 2009) showed that one can randomize over $\{\omega^m\}_{m=1}^M$ (the so-called randomized-feature approach) and minimize $\widehat{\mathbf{R}}(f)$ only on $\{\theta_m\}_{m=1}^M$, which is an efficiently solvable convex problem. The resulting solution was shown to be not much worse than the solution obtained by optimally tuning $\{\omega^m\}_{m=1}^M$. In this context, the feature map can be an eigenfunction of a positive-definite kernel, and for any $x^n, x^{n'} \in \mathcal{X}$, the kernel can be represented as

$$K_{P_\Omega}(x^n, x^{n'}) = \int_\Omega \phi(x^n, \omega) \phi(x^{n'}, \omega) P_\Omega(\omega) d_\omega, \tag{3}$$

where P_Ω is a distribution on the random features[1]. Using various feature maps and distributions one can recover commonly used kernels (e.g. Gaussian, Cauchy, Laplacian, arc-cosine, and linear) from (3). A list of possible choices can be found in Table 1 of (Yang et al. 2014b).

The approach of (Rahimi and Recht 2009) is particularly appealing due to its computational tractability since preparing

[1] More accurately, P_Ω is a probability density function when Ω is continuous, whereas it is a probability mass function when Ω is discrete, but instead, we use the word distribution to refer to both.

the feature matrix during training requires $O(MN)$ computations, while evaluating a test sample needs $O(M)$ computations, which significantly outperforms the complexity of traditional kernel methods. However, the potential drawback is that random features are drawn from a distribution P_Ω, *independent* of the training set, and therefore, we may require a large number of random features before learning the data subspace (Yang et al. 2012).

More specifically, under mild assumptions, the algorithm proposed in (Rahimi and Recht 2009) outputs an approximation $\widehat{f}(\cdot)$, which given a sampling distribution P_Ω, with probability at least $1-\varepsilon$ satisfies,

$$\mathbf{R}(\widehat{f}\,) - \min_{f\in\mathcal{F}_{P_\Omega}} \mathbf{R}(f) \le O\left(\left(\frac{1}{\sqrt{M}}+\frac{1}{\sqrt{N}}\right) C\sqrt{\log \varepsilon^{-1}}\right), \tag{4}$$

where

$$\mathcal{F}_{P_\Omega} \triangleq \{f(x) = \textstyle\int_\Omega F(\omega)\phi(x,\omega)d_\omega : |F(w)| \le CP_\Omega(\omega)\}. \tag{5}$$

As noted by (Rahimi and Recht 2009), $\mathcal{F}_{P_\Omega}$ is a rich class consisting of functions whose weights decay faster than the given sampling distribution P_Ω. As an example, in the case of sinusoidal feature maps, the set comprises of functions whose Fourier transforms decay faster than CP_Ω. This intuitively implies that given a sampling distribution P_0 (e.g., Gaussian) and a large constant C_0, we can hope to (under mild technical assumptions on the target function) push the best candidate within the class $\mathcal{F}_{P_0}$ to the target function. Given such P_0 and C_0, let us assume that the best function fit within the model class $\mathcal{F}_{P_0}$ is

$$f_0(x) \triangleq \text{argmin}_{f\in\mathcal{F}_{P_0}} \mathbf{R}(f) = \int_\Omega F_0(\omega)\phi(x,\omega)d_\omega. \tag{6}$$

While using the pair (C_0, P_0), $f_0(\cdot)$ can eventually be recovered (due to (4)), we may as well try to modify the initial sampling distribution P_0 according to the shape of $F_0(\cdot)$. In other words, if we knew $F_0(\cdot)$ precisely, we could set $P_\Omega(\omega) = \frac{|F_0(\omega)|}{\int_\Omega |F_0(\omega')| d_{\omega'}}$ and $C = \int_\Omega |F_0(\omega')|\, d_{\omega'}$, respectively, to sample the randomized features that weight more in the spectrum $F_0(\cdot)$. In the next section, we propose an algorithm that exploits the training data to find a "good" set of random features; thereby, improving the approximation of $f_0(\cdot)$ using finitely many random features for better generalization with small number of random features. The key is to use a *data-dependent* score function that approximately mimics the shape of $F_0(\cdot)$ with some error.

Proposed Method: Energy-based Exploration of Random Features (EERF)

In this section, we propose an algorithm to choose random features that maintain a low generalization error in supervised learning. The algorithm employs a *score* function to explore the domain of random features and retains the samples with the highest score. The key is to use a proper score function $S : \Omega \to \mathbb{R}$, which we define to be

$$S(\omega) \triangleq \mathbb{E}_{P_{\mathcal{XY}}}[y\phi(x,\omega)] \qquad \widehat{S}(\omega) \triangleq \frac{1}{N}\sum_{n=1}^{N} y^n\phi(x^n,\omega), \tag{7}$$

where $\widehat{S}(\cdot)$ denotes its empirical estimate. The score function can mimic kernel polarization (Baram 2005) asymptotically in the limit of large M. In particular, $\frac{1}{M}\sum_{m=1}^{M}\widehat{S}^2(\omega^m)$ for an i.i.d. sequence $\{\omega^m\}_{m=1}^M$ amounts to kernel polarization, which aims to polarize the data in the associated feature space to draw correspondence between the proximity of the points in the high-dimensional feature space and their responses. Roughly speaking, $S^2(\omega)$ can be considered to be an energy spectral density, after which the proposed algorithm is named. As is formally stated in Theorem 1, the proposed score function $S(\omega)$ is aligned to the spectrum $F_0(\omega)$ (up to an inevitable projection error). Given this result, we can use $\widehat{S}(\cdot)$, the empirical version of the true score, to re-weight features and modify the initial data-independent sampling distribution. Before further investigation of the behavior of the score function, we describe our algorithm.

Algorithm: Our algorithm works as follows: it draws M_0 samples from an initial distribution P_0, evaluates them in the empirical score given in (7), and selects the top M samples in the sense of maximizing $|\widehat{S}(\cdot)|$. The pseudo-code is given in Algorithm 1.

Algorithm 1 Energy-based Exploration of Random Features (EERF)

```
Input: $\{(x^n, y^n)\}_{n=1}^N$, the feature map $\phi(\cdot,\cdot)$, integers $M_0$
and $M$ where $M \le M_0$, initial sampling distribution $P_0$.

1: Draw samples $\{\tilde{\omega}^m\}_{m=1}^{M_0}$ independently from $P_0$.
2: Evaluate the samples in $\widehat{S}(\cdot)$, the empirical score in (7).
3: Sort $|\widehat{S}(\cdot)|$ for all $M_0$ samples in descending order, and
   let $\{\omega^m\}_{m=1}^M$ be the top $M$ arguments, i.e., the ones that
   give the top $M$ values in the sorted array.
Output: $\{\omega^m\}_{m=1}^M$.
```

Once we have the "good" M features $\{\omega^m\}_{m=1}^M$, we can solve the following empirical risk minimization (Rahimi and Recht 2009)

$$\widehat{\theta} = \underset{\theta:\|\theta\|_\infty \le \frac{C}{M}}{\text{argmin}} \left\{\frac{1}{N}\sum_{n=1}^{N} c\left(\frac{1}{\sqrt{M}}\sum_{m=1}^{M}\theta_m\phi(x^n,\omega^m), y^n\right)\right\}, \tag{8}$$

to approximate the underlying model. We remark that the EERF algorithm requires $O(dNM_0)$ computations to calculate the empirical score and $O(M_0 \log M_0)$ time on average to sort the M_0 obtained scores. One can often use a subsample of the training set instead of the entire N samples, and the value of M_0 should be set to an integer multiple of M. The initial distribution P_0 is either trivial to choose (e.g. uniform for the linear kernel, or standard Gaussian for the arccosine kernel), or can be selected using some rules-of-thumb. We elaborate on these issues in the experiments.

Theoretical results: The key to understanding Algorithm 1 is to analyze the empirical score $\widehat{S}(\cdot)$. It is immediate from McDiarmid's inequality that with probability at least $1-\delta$, the empirical score is concentrated around the true score as

$$\left|\widehat{S}(\omega) - S(\omega)\right| \leq O\left(\sqrt{\frac{2\log\frac{M_0}{\delta}}{N}}\right),$$

for all samples $\{\tilde{\omega}^m\}_{m=1}^{M_0}$. Restricting our attention to the regression model in Theorem 1, we show that $F_0(\cdot)$ and $S(\cdot)$ (when normlizad) exhibit similar behavior. That is, we can use the empirical version of $S(\cdot)$ in lieu of the *unknown* spectrum $F_0(\cdot)$ to better approximate $f_0(\cdot)$. An informal version of our result can be stated as follows

Theorem 1. *For the regression model where* $\mathbb{E}_{P_{\mathcal{Y}|\mathcal{X}}}[y] = f^\star(x)$*, under some technical assumptions, we have*

$$\frac{|S(\omega) - err_p(\omega)|}{\int_\Omega |S(u) - err_p(u)|\, d_u} \approx \frac{|F_0(\omega)|}{\int_\Omega |F_0(u)|\, d_u}, \tag{9}$$

where $err_p(\cdot)$ *is bounded by the sup-norm of* $f^\star(\cdot) - f_0(\cdot)$.

The exact statement of the theorem (including assumptions) and its proof and consequences are given in the extended version of the paper (Shahrampour, Beirami, and Tarokh 2017). Note that the projection error is inevitable and is a defect of the model class, and not the algorithm. The content of the theorem is that the score function aligns with the "spectrum" of the best model in the model class (5) (up to some projection error). Following the discussion after (6), recall that the right-hand side of (9) is precisely what we are seeking for the reconstruction of $f_0(\cdot)$, and our algorithm calculates an empirical version of $S(\cdot)$ to approximate the right-hand side of (9). Moreover, as we shall find in the supplementary material, $err_p(\cdot)$ is a decreasing function of C in (5), i.e., by increasing C, we make the class $\mathcal{F}_{P_\Omega}$ richer and decrease the projection error.

We remark that although the focus of Theorem 1 is on the regression model, the same approach intuitively applies to the logistic regression model for classification. In logistic regression, it can be shown that $\mathbb{E}_{P_{\mathcal{Y}|\mathcal{X}}}[y] = \tanh(f^\star(x)/2)$. Observe that $|\tanh(z/2)|$ is a monotonic function of $|z|$, and hence, the selection of random features based on the score function still aligns with the spectrum of the best model within the model class (5). Thus, the underlying intuition used to describe polarization (alignment) after (7) still holds.

Empirical Results

Gaussian Kernel

We apply our proposed method to several datasets from the UCI Machine Learning Repository. Since all of our baseline algorithms are applicable to Gaussian kernels, we first compare our method to the state-of-the-art within that framework, and next we show the applicability of our method to linear and arccosine kernels.

Benchmark algorithms: We use the algorithms in Table 1 as baselines for comparison. Notice that in Table 1, we have also reported the prior work used by each baseline for comparison. The following comments are in order:

- The ORF algorithm involves a QR decomposition step ($O\left(d^3\right)$ time) which can be side-stepped using the companion algorithm SORF in (Felix et al. 2016). The main advantage of SORF, which combines Walsh-Hadamard matrices and diagonal "sign-flipping" matrices, is computational, thus when the prediction accuracy is concerned, SORF and ORF are shown to have similar performance, while SORF performs worse than ORF for $d < 32$ (Felix et al. 2016).
- LKRF introduces a pre-processing optimization to reweight random features. The algorithm initially samples M_0 random features, forms the optimization with $O\left(dM_0N\right)$ computations, and requires $O\left(M_0 \log \epsilon^{-1}\right)$ time to find an ϵ-optimal solution. Also, the optimization involves a hyper-parameter balancing the trade-off between an alignment measure versus the f-divergence of solution with the uniform distribution. We run the algorithm multiple times with the hyperparameter in the set $\{10^{-5}, 10^{-4}, \ldots, 10^5\}$ and report the best result.
- The SES algorithm also re-weights random features by solving an optimization problem using sketching techniques. Letting T_S be the time cost of sketching, the optimization problem costs $O\left(rd^2 + T_S\right)$, where r is the number of samples included in the sketching matrix. The main purpose of SES is kernel approximation, but when applied to supervised learning on a number of datasets, SES has proven to be competitive to Monte Carlo and Quasi Monte Carlo methods (see Table 1).

Following (Rahimi and Recht 2009), we replace the infinity-norm constraint of (8) by a quadratic regularizer in practice. We then tune the regularization parameter by trying different values from $\{10^{-5}, 10^{-4}, \ldots, 10^5\}$. For all methods (including ours) in the Gaussian case, we sample random features from $\frac{1}{\sigma}\mathcal{N}(0, I_d)$. The value of σ for each dataset is chosen to be the mean distance of the 50th ℓ_2 nearest neighbor, which is shown to result in good classification[2] (Felix et al. 2016). It is also important to note that:

- RKS, ORF, and SES draw M samples from the Gaussian distribution. RKS directly uses the samples, ORF "orthogonalizes" them to another set of M vectors, and SES re-weights them using an optimization.
- In contrast, LKRF and EERF (our work) draw $M_0 > M$ samples from the Gaussian distribution, process them, and use the most promising M samples. The value of M_0 for each dataset along with the pre-processing overhead for both algorithms are reported in Table 5.

All codes are written in MATLAB and run on a machine with CPU 2.9 GHz and 16 GB memory.

Datasets: Table 2 represents the number of training samples (N_{train}) and test samples (N_{test}) used for each dataset. If training and test sets are provided explicitly, we use them accordingly; otherwise, we split the dataset randomly. The features in all datasets are scaled to be empirically zero mean

[2] This choice is a rule-of-thumb and further tuning may result in improved generalization. However, we use the same choice for all of the methods for a fair comparison.

Table 1: We compare our work to the baselines in the left-most column. The right-most column lists the prior art (on random features) that was compared against in the baseline paper.

Baseline	Data-dependent	Class of kernels	Prior work used for comparison by the baseline
RKS (Rahimi and Recht 2009)	No	Eq. (3)	Adaboost
ORF (Felix et al. 2016)	No	Gaussian	RFF (Rahimi and Recht 2007), Fastfood (Le, Sarlós, and Smola 2013), QMC (Yang et al. 2014a), Circulant (Yu et al. 2015)
LKRF (Sinha and Duchi 2016)	Yes	Eq. (3)	RKS (Rahimi and Recht 2009)
SES (Chang et al. 2017)	Yes	Gaussian	RFF (Rahimi and Recht 2007), BQ (Huszár and Duvenaud 2012), QMC (Yang et al. 2014a)

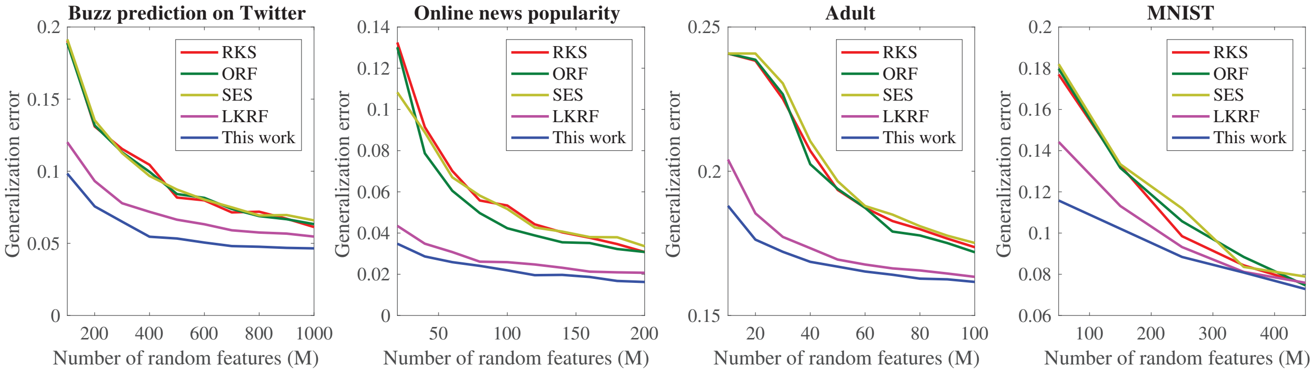

Figure 1: Performance on practical datasets: we compare the generalization error of our method (EERF) with the baselines RKS, ORF, SES, and LKRF. In all cases, for a fixed M, our algorithm achieves a smaller generalization error.

and unit variance and the responses in regression tasks are normalized to $[-1, 1]$.

Table 2: The description of the datasets used for Gaussian kernel: the number of features, training samples, and test samples are denoted by d, N_{train}, and N_{test}, respectively.

Dataset	Task	d	N_{train}	N_{test}
Buzz prediction on Twitter	Regression	77	93800	46200
Online news popularity	Regression	58	26561	13083
Adult	Classification	122	32561	16281
MNIST	Classification	784	60000	10000

Performance: The results on datasets in Table 2 are reported in Fig. 1: for each dataset, by pre-processing random features in the score function (7), our method learns the subspace faster compared to state-of-the-art, i.e., we require smaller number of random features M to achieve a given generalization error threshold. As the number of samples increases, all methods tend to generalize better, which is not surprising, since they eventually sample the "good" random features for learning the data model. In the regime of moderate M, LKRF closely competes with our algorithm due to its data-dependent pre-processing phase. We will elaborate on the performance of our method versus LKRF in the next section, after experiments on linear and arccosine kernels are also presented.

Table 3 tabulates the time cost and the generalization error for our method and RKS used with a fixed Gaussian kernel. For each dataset, the statistics are reported for the largest value of M used in the experiment. We randomly subsample N_0 data points of the dataset to calculate the empirical score (e.g., for "Buzz prediction on Twitter" we use 10% of the training samples) and generate an initial M_0 random features to evaluate the score function. Then, the most promising M samples with the highest scores are selected following Algorithm 1, and the performance is compared to the case where M Monte-Carlo samples are generated by RKS. As an example, for the "Buzz prediction on Twitter" dataset, our method reduces the test error of RKS by 23.63%. However, this accuracy comes at a computational cost in the pre-processing stage. Table 3 also tabulates the pre-processing, training, and testing time of our algorithm. Except for the "Online news popularity" dataset, the pre-processing time is always less than the training time, and most notably, for "Adult" dataset it is only 10.2% of the training time. In general, the comparison between the two may not be immediate: our pre-processing requires $O(M_0 N_0 d)$ computations to evaluate the score function, followed by the time cost of sorting an array of size M_0 (on average $O(M_0 \log M_0)$). On the other hand, while training requires $O(MNd)$ computations to build the feature matrix, the training time can be largely affected by the choice of regularization parameter used in lieu of the norm-infinity

Table 3: Comparison of the time cost and performance of our algorithm versus RKS. t_{pp}, t_{train}, and t_{train} represent pre-processing, training, and testing time (seconds). N_0 is the number of samples we use for pre-processing, and M_0 is the number of random features we initially generate. M is the number of random features used by both algorithms for eventual prediction. The standard errors are reported in parentheses.

Dataset	M	M_0	N_0/N	Our t_{pp}	Our t_{train}	Our t_{test}	RKS t_{train}	RKS t_{test}	Our error (%)	RKS error (%)
Buzz prediction on Twitter	1000	10000	10%	0.84	1.96	1.76	2.11	1.78	**4.65** (4e-2)	6.09 (7e-2)
Online news popularity	200	20000	5%	0.53	0.15	0.07	0.13	0.04	**1.63** (4e-2)	3.08 (5e-2)
Adult	100	2000	5%	0.19	1.78	0.05	1.61	0.06	**16.16** (2e-2)	17.37 (6e-2)
MNIST	450	10000	20%	5.20	16.17	8.58	19.45	10.65	**7.28** (3e-2)	7.53 (1.8e-1)

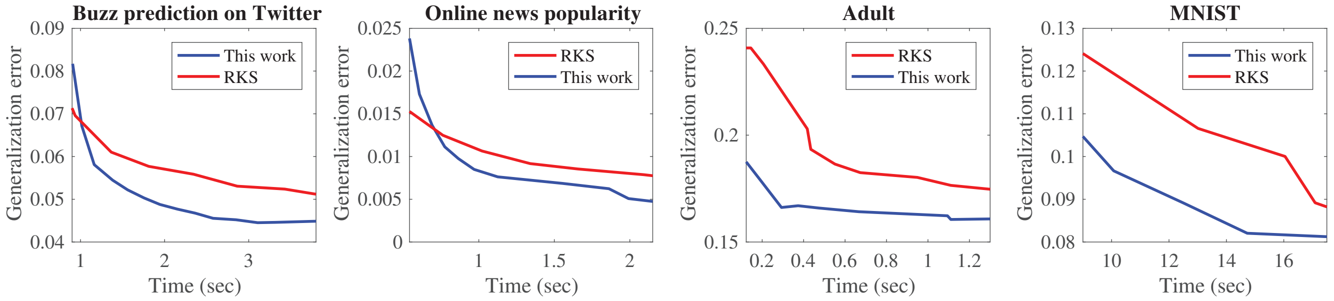

Figure 2: Generalization error versus time for our method against RKS. The time for our method is the summation of training and pre-processing time, whereas for RKS it is the training time.

Table 4: The description of the datasets used for linear and arccosine kernels: the number of features, training samples, and test samples are denoted by d, N_{train}, and N_{test}, respectively. $H(\cdot)$ denotes the Heaviside step function ($H(x) = 0.5 + 0.5\text{sgn}(x)$, where $\text{sgn}(\cdot)$ is the sign function).

Dataset	Task	$\phi(x, \omega)$	d	N_{train}	N_{test}
Years prediction	Regression	$(\omega^\top x)H(\omega^\top x)$	90	463715	51630
Letter recognition	Classification	$(\omega^\top x)^2 H(\omega^\top x)$	16	15000	5000
Hand movements	Classification	x_ω	561	7352	2947

constraint in (8). For instance, in regression, the parameter directly governs the condition number of the $M \times M$ matrix that is to be inverted. As a rule of thumb, one can select M_0 in the range $5M$ to $20M$ and set N_0 to $0.1N$.

Since our method adds a computational overhead versus RKS, we also plot the generalization error versus time for both methods in Fig. 2. The time for RKS represents the training time, whereas for our method it is the sum of the training and pre-processing time. For example, in Adult dataset, where our computational overhead is quite negligible, our method has superior accuracy versus RKS for any computational time. However, in general the trend is as follows: for very small number of random features (or relatively bad accuracy), our method is inferior, but past a certain computational time threshold, we outperform RKS. This is not surprising: for very small number of features, training is fast, but our method still calculates the empirical score, adding additional cost to training. Once we have more random features, training tends to take more time and the preprocessing time becomes less significant compared to the training time.

Linear and Arccosine Kernels

The feature map x_ω results in linear kernel when ω is sampled uniformly from $[d]$, and $(\omega^\top x)^n H(\omega^\top x)$ with $\omega \sim \mathcal{N}(0, I_d)$ gives the arccosine kernel[3] of order n ($H(\cdot)$ denotes the Heaviside step function, i.e., $H(x) = 0.5 + 0.5\text{sgn}(x)$, where $\text{sgn}(\cdot)$ is the sign function). These two kernels are among many others that conform to (3). In this part, we focus on these two kernels and compare our method with LKRF (Sinha and Duchi 2016). ORF and SES are designed for Gaussian kernels and are not applicable in this setting. RKS is data-independent, and as we saw in the previous part, for small number of random features, it is outperformed by EERF and LKRF. Table 4 describes the datasets as well as their corresponding feature map used for the experiment. We follow the previous section in data standardization and tuning the regularization parameters.

Performance of EERF versus LKRF: In Fig. 3, we compare our performance with LKRF (Sinha and Duchi 2016) in terms of the generalization error on several datasets. Our method slightly outperforms LKRF on "Year prediction" and "Letter recognition", while significantly improving the gener-

[3] The constraints in Theorem 1 do not hold for the feature map associated to the arccosine kernel, but we still observed improvement in the generalization error in practice.

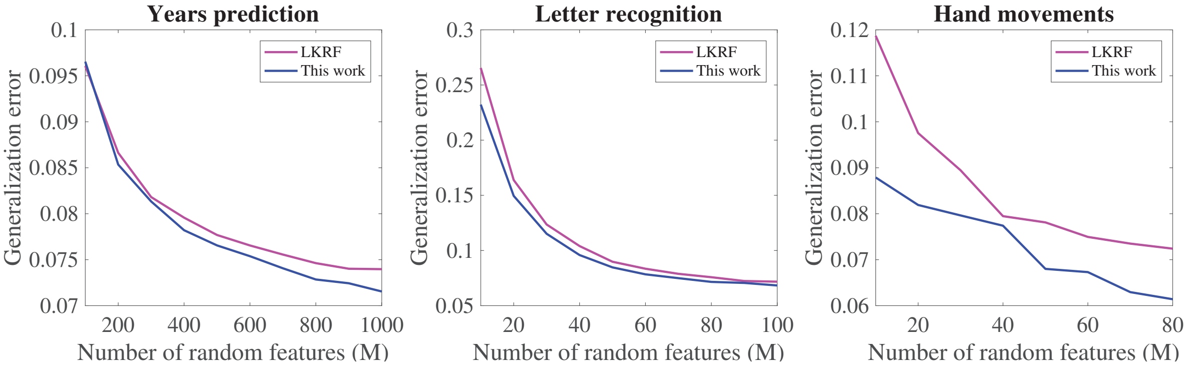

Figure 3: Performance on three datasets: we compare the generalization error of our method (EERF) against LKRF.

Table 5: Comparison of the time cost and performance of our algorithm versus LKRF. t_{pp}, t_{train}, and t_{train} represent pre-processing, training, and testing time (sec). N_0 is the number of samples both algorithms use for pre-processing, and M_0 is the number of random features they initially generate. M is the number of random features used by both algorithms for eventual prediction. The standard errors are reported in parentheses.

Dataset	M	M_0	N_0/N	Our t_{pp}	LKRF t_{pp}	Our t_{train}	LKRF t_{train}	Our error (%)	LKRF error (%)
Buzz prediction on Twitter	1000	10000	10%	0.84	0.97	1.96	1.88	**4.65** (4e-2)	5.23 (6e-2)
Online news popularity	200	20000	5%	0.53	0.50	0.15	0.14	**1.63** (4e-2)	2.07 (5e-2)
Adult	100	2000	5%	0.19	0.08	1.78	1.35	**16.16** (2e-2)	16.34 (2e-2)
MNIST	450	10000	20%	5.20	6.13	15.97	16.17	**7.28** (3e-2)	7.59 (1.7e-1)
Year prediction	1000	4000	10%	6.23	7.24	32.31	53.96	**7.15** (1.9e-2)	7.40 (1.4e-2)
Letter recognition	100	500	100%	4.55	5.44	11.33	12.95	**6.83** (7e-2)	7.17 (8e-2)
Hand movement	80	561	100%	0.18	0.04	0.83	0.98	**6.14** (2.7e-3)	7.24 (1.2e-2)

alization error on “Hand movement”. Further, Table 5 shows the time cost and the generalization error for the largest value of M in the plots. For the pre-processing stage, both algorithms sample an initial distribution M_0 times and incur $O\left(dN_0M_0\right)$ computational cost. Our algorithm sorts an array of size M_0 with average $O(M_0 \log M_0)$ time, while LKRF solves an optimization with $O\left(M_0 \log \epsilon^{-1}\right)$ time to reach the ϵ-optimal solution. Therefore, depending on the tolerance ϵ, the processing time may vary for LKRF.

The main advantage of our method over LKRF is that EERF is parameter-free and does not require tuning. LKRF solves an optimization problem to re-weight random features, which depends on a regularization parameter. The new weights can range from a uniform to a degenerate delta distribution, depending on the regularization parameter, which needs to be tuned. We observed that this brings forward two issues: i) a validation step for tuning the regularization parameter is needed, (ii) the obtained parameter works well only for a range of values for M and needs to be re-tuned for others.

Concluding Remarks

In this paper, we studied data-dependent random features for supervised learning. We proposed an algorithm called Energy-based Exploration of Random Features (EERF), which is based on a data-dependent scoring rule for sampling random features. We proved that under mild conditions, the proposed score function with high probability recovers the spectrum of best model fit within the postulated model class. We further empirically showed that our proposed method outperforms the state-of-the-art data-independent and data-dependent algorithms based on the randomized-feature approach. The EERF algorithm introduces a small computational pre-processing overhead and requires no additional tuning parameters in contrast to other data-dependent methods for generation of random features. Our method is particularly designed to reduce generalization error in regression and classification. Inspired by the recent results on the application of random features in matrix completion (cf. (Si et al. 2016)), an interesting future direction is to adapt our score function to improve generalization in this setup.

Acknowledgements

This work was supported in part by NSF ECCS Award No. 1609605. We thank the anonymous reviewers for their constructive feedback and the suggestion to provide time-vs-accuracy plots.

References

Baram, Y. 2005. Learning by kernel polarization. *Neural Computation* 17(6):1264–1275.

Bartlett, P. L., and Mendelson, S. 2002. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research* 3(Nov):463–482.

Chang, W.-C.; Li, C.-L.; Yang, Y.; and Poczos, B. 2017. Data-driven random fourier features using stein effect. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*.

Cortes, C.; Mohri, M.; and Rostamizadeh, A. 2009. Learning non-linear combinations of kernels. In *Advances in neural information processing systems*, 396–404.

Cortes, C.; Mohri, M.; and Rostamizadeh, A. 2010. Generalization bounds for learning kernels. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 247–254.

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

Cristianini, N., and Shawe-Taylor, J. 2000. An introduction to support vector machines.

Drineas, P., and Mahoney, M. W. 2005. On the nyström method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research* 6(Dec):2153–2175.

Felix, X. Y.; Suresh, A. T.; Choromanski, K. M.; Holtmann-Rice, D. N.; and Kumar, S. 2016. Orthogonal random features. In *Advances in Neural Information Processing Systems*, 1975–1983.

Gönen, M., and Alpaydın, E. 2011. Multiple kernel learning algorithms. *Journal of Machine Learning Research* 12(Jul):2211–2268.

Huszár, F., and Duvenaud, D. 2012. Optimally-weighted herding is bayesian quadrature. *UAI*.

Kandola, J.; Shawe-Taylor, J.; and Cristianini, N. 2002. Optimizing kernel alignment over combinations of kernel.

Kar, P., and Karnick, H. 2012. Random feature maps for dot product kernels. In *International conference on artificial intelligence and statistics*, 583–591.

Kloft, M.; Brefeld, U.; Sonnenburg, S.; and Zien, A. 2011. Lp-norm multiple kernel learning. *Journal of Machine Learning Research* 12(Mar):953–997.

Lanckriet, G. R.; Cristianini, N.; Bartlett, P.; Ghaoui, L. E.; and Jordan, M. I. 2004. Learning the kernel matrix with semidefinite programming. *Journal of Machine learning research* 5(Jan):27–72.

Le, Q.; Sarlós, T.; and Smola, A. 2013. Fastfood-approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning*, volume 85.

Oliva, J. B.; Dubey, A.; Wilson, A. G.; Póczos, B.; Schneider, J.; and Xing, E. P. 2016. Bayesian nonparametric kernel-learning. In *Artificial Intelligence and Statistics*, 1078–1086.

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

Rahimi, A., and Recht, B. 2009. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems (NIPS)*, 1313–1320.

Rudi, A.; Camoriano, R.; and Rosasco, L. 2016. Generalization properties of learning with random features. *arXiv preprint arXiv:1602.04474*.

Shahrampour, S.; Beirami, A.; and Tarokh, V. 2017. On data-dependent random features for improved generalization in supervised learning. *arXiv preprint*.

Si, S.; Chiang, K.-Y.; Hsieh, C.-J.; Rao, N.; and Dhillon, I. S. 2016. Goal-directed inductive matrix completion. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1165–1174. ACM.

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

Sutherland, D. J., and Schneider, J. 2015. On the error of random fourier features. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, 862–871.

Vedaldi, A., and Zisserman, A. 2012. Efficient additive kernels via explicit feature maps. *IEEE transactions on pattern analysis and machine intelligence* 34(3):480–492.

Williams, C., and Seeger, M. 2001. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*.

Yang, T.; Li, Y.-F.; Mahdavi, M.; Jin, R.; and Zhou, Z.-H. 2012. Nyström method vs random fourier features: A theoretical and empirical comparison. In *Advances in neural information processing systems*, 476–484.

Yang, J.; Sindhwani, V.; Avron, H.; and Mahoney, M. 2014a. Quasi-monte carlo feature maps for shift-invariant kernels. In *Proceedings of The 31st International Conference on Machine Learning (ICML-14)*, 485–493.

Yang, J.; Sindhwani, V.; Fan, Q.; Avron, H.; and Mahoney, M. W. 2014b. Random laplace feature maps for semigroup kernels on histograms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 971–978.

Yang, Z.; Wilson, A.; Smola, A.; and Song, L. 2015. A la carte–learning fast kernels. In *Artificial Intelligence and Statistics*, 1098–1106.

Yu, F. X.; Kumar, S.; Rowley, H.; and Chang, S.-F. 2015. Compact nonlinear maps and circulant extensions. *arXiv preprint arXiv:1503.03893*.