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

Alternating Circulant Random Features for Semigroup Kernels

Yusuke Mukuta
The University of Tokyo
mukuta@mi.t.u-tokyo.ac.jp

Yoshitaka Ushiku
The University of Tokyo
ushiku@mi.t.u-tokyo.ac.jp

Tatsuya Harada
The University of Tokyo and RIKEN AIP
harada@mi.t.u-tokyo.ac.jp

Abstract

The random features method is an efficient method to approximate the kernel function. In this paper, we propose novel random features called "alternating circulant random features," which consist of a random mixture of independent random structured matrices. Existing fast random features exploit random sign flipping to reduce the correlation between features. Sign flipping works well on random Fourier features for real-valued shift-invariant kernels because the corresponding weight distribution is symmetric. However, this method cannot be applied to random Laplace features directly because the distribution is not symmetric. The method proposed herein yields alternating circulant random features, with the correlation between features being reduced through the random sampling of weights from multiple independent random structured matrices instead of via random sign flipping. The proposed method facilitates rapid calculation by employing structured matrices. In addition, the weight distribution is preserved because sign flipping is not implemented. The performance of the proposed alternating circulant random features method is theoretically and empirically evaluated.

1 Introduction

Kernel methods are effective for classification using nonlinear relationships between data. However, the complexity of these methods increases as the dataset size grows. Thus, kernel approximation methods are necessary.

The random features method is an approximation method that first represents the given kernel function by the expectation of the product of the parameterized function values of two inputs and then approximates the kernel with the inner product of vectors composed of the values of randomly sampled functions. Various random features techniques have been proposed, with their construction depending on the structure of the given input feature and kernel function (Pennington, Felix, and Kumar 2015; Kar and Karnick 2012; Rahimi and Recht 2007; Yang et al. 2014).

When we apply the random features method on large-scale data, we need to consider the cost involved in calculating the approximate feature vectors. Naive approximation methods often require $O(Dd)$ computation, where d and D denote input and output dimensions, respectively. This is not negligible when d is large because in most cases, $D \geq d$. Thus, recent work has proposed methods to calculate the features with $O(D \log d)$. Most existing work approximates the Gaussian kernel (Felix et al. 2016; Le, Sarlós, and Smola 2013; Yu et al. 2015) or polynomial kernels (Pham and Pagh 2013). In this work, we propose a fast method that approximates semigroup kernels that is effective for positive vectors but where we cannot apply existing fast methods directly.

A wide range of features, including histograms, discrete probabilistic distributions, and activations of convolutional neural networks (CNNs), are associated with the space of positive vectors $\mathbb{R}_+^d$. The semigroup kernel is a kernel that exploits such positive definiteness. For two inputs $x, y \in \mathbb{R}_+^d$, the kernel value of the semigroup kernel depends only on the summation of the two inputs $x + y$. In such a case, Berg et al. (Berg, Christensen, and Ressel 1984) have proven that there exists a distribution $p(w)$ on $\mathbb{R}_+^d$ such that

$$k(x+y) = \int e^{-w^t(x+y)} p(w) dw. \tag{1}$$

Based on Eq. (1), Yang et al. (Yang et al. 2014) proposed random Laplace features, which use $\{\sqrt{\frac{1}{D}} e^{-w_j x}\}_{j=1}^D$ with $\{w_j\}_{j=1}^D$ sampled from $p(w)$ as a D-dimensional feature vector. Random Laplace features exhibit superior performance to existing commonly used kernels such as the Gaussian and χ^2 kernels with regard to the classification of histogram data. The computational complexity is $O(Dd)$. As the dimension of the input feature grows, this linear dependency on d becomes non-negligible.

In this paper, we propose a method to calculate the feature vector with $O(D \log d)$ complexity by exploiting structured matrices to approximate the semigroup kernel quickly. Structured matrices are matrices with special structures that enable us to calculate the matrix-vector products recursively with $O(d \log d)$ time complexity and $O(d)$ memory at the expense of independency between different feature elements. Existing fast Gaussian kernel approximation methods (Felix et al. 2016; Le, Sarlós, and Smola 2013; Yu et al. 2015) combine structured matrices with random sign flipping to reduce the correlation between different feature elements. For random Fourier features on a shift-invariant kernel such as Gaussian kernel, the sample distribution is symmetric. Thus, sign flipping does not change the output distribution. However, for random Laplace features, the support of the weight distribution $p(w)$ and the input x are positive. Thus, sign flipping changes the output distribution. Instead, we propose a novel method called "alternating circulant random features" that utilizes weights randomly sampled from multiple independent structured matrices to calculate the features. Because the proposed method uses structured matrices, the computation complexity depends on the $\log$ of the input dimension d. Further, the proposed method preserves the weight distribution by replacing random sign flipping with random choice from the variables sampled from the same distribution. Thus, this technique can be applied to semigroup kernels.

In addition, we demonstrate that the covariance for random circulant features without random sign flipping is as large as autocovariance, while the covariance for the proposed method becomes small with respect to the autocovariance as d increases. Thus, the proposed features exhibit comparable performance to random Laplace features.

We compare the performance of the proposed method to that of an approximate semigroup kernel for application to Bag of Visual Words features that can be regarded as a histogram, and to the CNN features. We demonstrate that the proposed method exhibits comparable classification performance to the random Laplace features method with a significantly shorter computation time.

In summary, the contributions of this paper are as follows:

- We propose a novel fast random features called "alternating circulant random features"; the proposed method can compute random features with computation complexity $O(D \log d)$ and $O(D)$ memory requirement.
- We theoretically evaluate that the approximation error of the proposed method is comparable to that of the random Laplace features method as the input dimension d grows.
- Experimental results using Bag of Visual Words features and CNN features show that the proposed method achieves similar accuracy to the random Laplace features method and requires less computation time.

2 Background

For a given kernel function $k(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ expressed as the expectation of the product of the parameterized function $\phi_w(\cdot) : \mathbb{R}^d \to \mathbb{R}$ with respect to the parameter w, denoted as $E_{w \sim p(w)}[\phi_w(\cdot)\phi_w(\cdot)]$, random features use $\{\sqrt{\frac{1}{D}}\phi_{w_j}(\cdot)\}_{j=1}^D$ with $\{w_j\}_{j=1}^D$ randomly sampled from p as a feature function to approximate the kernel value.

Semigroup kernels are the kernels whose values depend only on the summation $x + y$ of the two inputs $x, y \in \mathbb{R}_+^d$. For the semigroup kernel, there exists a distribution $p(w)$ on $\mathbb{R}_+^d$ that satisfies Eq. (1). Random Laplace features use $\{\sqrt{\frac{1}{D}}e^{-w_i x}\}_{i=1}^D$ with $\{w_i\}_{i=1}^D$ sampled from $p(w)$ as a feature. In this study, in addition to the assumption of semigroup kernel, we also assume that the kernel can be decomposed as the product of the kernels of each dimension k_1, as $k(x + y) = \prod_{j=1}^d k_1(x_j + y_j)$. Note that the exponential-semigroup kernel $e^{-\beta \sum_{j=1}^d \sqrt{x_j + y_j}}$ and reciprocal-semigroup kernel $\prod_{j=1}^d \frac{\lambda}{x_j + y_j + \lambda}$, to which Yang et al. applied random Laplace features, satisfy this assumption. When we denote the matrix $W = [w_1, w_2, \cdots w_D]^t \in \mathbb{R}^{D \times d}$, the features are expressed as $\sqrt{\frac{1}{D}}e^{-Wx}$.

We call the kernel function defined by two inputs $x, y \in \mathbb{R}^d$ a "shift-invariant kernel," if the kernel value depends on the difference in the input $x - y$ only. Bochner (Rudin 2011) has shown that the shift-invariant kernel k can be expressed with the measure $p(w)$ on $\mathbb{R}^d$ as

$$k(x - y) = \int e^{iw^t(x-y)} p(w) dw. \tag{2}$$

Random Fourier features methods (Rahimi and Recht 2007) randomly sample $\{w_j\}_{j=1}^D$ from $p(w)$ in Eq. (2) and use $\sqrt{\frac{1}{D}}e^{iWx}$ or $\sqrt{\frac{2}{D}}\cos(Wx + b)$, with b uniformly sampled from $[0, 2\pi]$, as features.

Recently, fast computation methods have been proposed for the Gaussian kernel, which is one of the shift-invariant kernels. These approaches use structured matrices where the matrix-vector product can be calculated with $O(d \log d)$ and random diagonal matrices where the matrix-vector product can be calculated with $O(d)$ (Felix et al. 2016; Le, Sarlós, and Smola 2013; Yu et al. 2015). These studies have allowed calculation of d-dimensional random features with $O(d \log d)$ time complexity and we achieve D-dimensional features with $O(D \log d)$ complexity by independently sampling the features D/d times. Such fast random features exhibit comparable or superior approximation performance to the original random Fourier features.

3 Related Work

Various approximation techniques have been proposed depending on the kernel structures (Pennington, Felix, and Kumar 2015; Pham and Pagh 2013; Rahimi and Recht 2007; Yang et al. 2014). In this section, we focus on fast random Fourier features using structured matrices. The properties of the existing fast approximation methods and the proposed method are summarized in Table 1. In the discussion that follows, we assume that d is a power of 2. In the general case, we pad 0's to set the dimension to a power of 2.

FastFood FastFood (Le, Sarlós, and Smola 2013) exploits the property for which the Hadamard matrix combined with diagonal Gaussian matrices exhibits similar behavior to a Gaussian matrix. For the Gaussian kernel $e^{-\frac{\|x-y\|^2}{2\sigma^2}}$, FastFood uses

$$\sqrt{2}\cos\left(\frac{1}{\sigma\sqrt{d}}SHG\Pi HBx + b\right) \tag{3}$$

as a global feature, where H is the Hadamard matrix whose elements are -1 or 1.This matrix is a constant multiple of orthogonal matrices, and the matrix-vector product can be calculated recursively with $O(d \log d)$ time complexity. Here, S, G, and B are diagonal matrices representing random scaling, a random Gaussian variable, and random sign flipping,

Table 1: Comparison of kernel approximation methods. Here, d and D denote the dimensions of the input and output features, respectively, and m is the number of mixed structured matrices (2 or $\log_2 d$ in this study). "Gaussian" and "Semigroup" indicate the method applicability to Gaussian and semigroup kernels, respectively. Note that the random circulant features method can be applied to semigroup kernels if we omit random sign flipping.

Method	Memory	Time	Gaussian	Semigroup
Random Fourier / Laplace Features	$O(Dd)$	$O(Dd)$	Yes	Yes
FastFood	$O(D)$	$O(D \log d)$	Yes	No
Circulant	$O(D)$	$O(D \log d)$	Yes	Partially Yes
Structured Orthogonal Random Features	$O(D)$	$O(D \log d)$	Yes	No
Alternating Circulant Random Features (ours)	$O(mD)$	$O(mD \log d)$	Yes	Yes

respectively. Further, Π is a random permutation. Thus, the complexity needed to calculate a d-dimensional random feature is $O(d \log d)$. We independently sample this feature D/d times to obtain the D-dimensional output feature.

Random Circulant Features Yu et al. (Yu et al. 2015) have proposed a random Fourier features that uses the circulant matrix instead of the Hadamard matrix. For the vector $r \in \mathbb{R}^d$, the circulant matrix $\mathrm{circ}(r) \in \mathbb{R}^{d \times d}$ is defined as

$$\mathrm{circ}(r) = \begin{bmatrix} r_1 & r_d & \cdots & r_2 \\ r_2 & r_1 & \cdots & r_3 \\ \vdots & r_2 & \ddots & \vdots \\ r_d & r_{d-1} & \cdots & r_1 \end{bmatrix}. \tag{4}$$

We can calculate the matrix-vector product using the fast Fourier transform as $\mathrm{circ}(r)x = F^{-1}(F(r) \circ F(x))$ with $O(d \log d)$ complexity, where $\circ$ denotes the element-wise product.

With randomly sampled w, the random circulant features approach yields

$$\sqrt{2} \cos\left(\mathrm{circ}(w)Bx + b\right) \tag{5}$$

as a global feature. Although Yu et al. did not analyze the performance of this method, in experiments, the random circulant features method exhibits equivalent performance to the random Fourier features method.

Choromanski & Sindhwani (Choromanski and Sindhwani 2016) generalized the fast Fourier features method to construct low-variance features via graph-theoretic concept.

Structured Orthogonal Random Features Felix et al. (Felix et al. 2016) have proposed the use of randomly sampled orthogonal matrices as the W for a Gaussian kernel, and have also proposed structured orthogonal random features that compute matrices similar to orthogonal matrices with $O(d \log d)$ time complexity. Although structured orthogonal random features are not an unbiased estimator of the Gaussian kernel, the approximation error converges to 0 with increasing d. Further, the correlation between features is small because orthogonal matrices are used. Structured orthogonal random features are calculated as

$$\sqrt{2} \cos\left(\frac{\sqrt{d}}{\sigma} HB_1HB_2HB_3 + b\right), \tag{6}$$

where the B_i terms correspond to independent random sign flipping.

4 Alternating Circulant Random Features

As discussed in the previous section, existing fast random features methods using structured matrices depend on sign flipping to reduce the covariance between the obtained features. Thus, we cannot apply these methods to the case of random Laplace features for which the weights are sampled from $\mathbb{R}_+^d$. The random circulant features approach does not use Hadamard matrices; thus, random circulant features can be applied if random sign flipping is omitted. However, as shown in the next section, the covariance of the features becomes very large. To overcome this problem, we propose the alternating circulant random features method, in which random sign flipping is replaced with random mixture, and we analyze the approximation performance.

4.1 Method

In the proposed method, we employ a random column mixture of m independent circulant matrices. We sample $\{w_j^{(l)}\}_{j=1,l=1}^{j=d,l=m}$ in advance and use $\sqrt{\frac{1}{D}}e^{-Wx}$ or $\sqrt{\frac{2}{D}}\cos(Wx+b)$, with each column of W uniformly sampled from $\mathrm{circ}(w^{(l)})_{:,j}$ with l ranging from 1 to m. As each column in $\mathrm{circ}(w)$ is composed of all the elements of w, only changing the column reduces the correlation of feature elements drastically. Thus, we can rapidly calculate the proposed features, as in the case of random circulant features. We denote $s^{(l)} \in \mathbb{R}^d$ as a vector, where $s_j^{(l)} = 1$ if and only if the j-th column is sampled from $\mathrm{circ}(w^{(l)})$ and is 0 otherwise. Then, Wx can be calculated as

$$\sum_{l=1}^{m} F^{-1}\left(F\left(w^{(l)}\right) \circ F\left(s^{(l)} \circ x\right)\right). \tag{7}$$

The computational complexity of Eq. (7) is $O(mD \log d)$ and the required memory is $O(mD)$. In the experiments discussed below, we considered the case in which $m = 2$ and $m = \log_2 d$. When $m = 2$, the computational complexity is identical to that for existing fast random Fourier features methods. Further, the proposed method is an unbiased estimator of the original kernel because for each selected row, each column element is independent and identically distributed (i.i.d.) from the original distribution.

4.2 Analysis

In this section, we evaluate the variance of the kernel functions approximated with random Laplace features. We first evaluate the original method, which randomly samples all the weights i.i.d. from p_1; then, we evaluate the case of random circulant features without random sign flipping. Finally, we calculate the variance for the proposed features. For two inputs x and y, we define $z = x + y \in \mathbb{R}^d$. As discussed in Section 1, we assume that $k(z) = \prod_{j=1}^{d} k_1(z_j)$. In this case, the corresponding distribution p can be decomposed as $p(w) = \prod_{j=1}^{d} p_1(w_j)$.

First, we evaluate the autocovariance of the approximate kernel value e^{-wz} for the original method.

Theorem 1. *$Var_{w \sim p(w)}[e^{-z^t w}] = k(2z) - k(z)^2$.*

Proof. It holds that

$$E_{w \sim p(w)}\left[\left(e^{-z^t w}\right)^2\right] = E_{w \sim p(w)}[e^{-(2z)^t w}] = k(2z). \tag{8}$$

Thus,

$$\begin{aligned} & \mathrm{Var}_{w \sim p(w)}[e^{-z^t w}] && (9) \\ = \; & E_{w \sim p(w)}\left[\left(e^{-z^t w}\right)^2\right] - \left(E_{w \sim p(w)}[e^{-z^t w}]\right)^2 && (10) \\ = \; & k(2z) - k(z)^2. && (11) \end{aligned}$$

□

As there is no covariance between elements with different dimensions, the variance of the approximated kernel with output dimension D is $\frac{k(2z)-k(z)^2}{D}$.

Next, we calculate the covariance for random circulant features without random sign flipping. For the vector $v = [v_1, v_2, \cdots, v_d]^t$, we denote ${}^{\uparrow n}v = [v_{n+1}, v_{n+2}, \cdots, v_d, v_1, \cdots, v_n]^t$, ${}^{\downarrow n}v = {}^{\uparrow -n}v$, and $\mathrm{rev}(v) = [v_d, v_{d-1}, \cdots, v_2, v_1]^t$. The autocovariance is identical to that for the original method. In this case, the elements for the different dimensions have non-zero covariance because the weights are shared in the different rows.

Theorem 2.

$$E_{w \sim p(w)}[e^{-(Wz)_{j_1}} e^{-(Wz)_{j_2}}] = k(z + {}^{\uparrow (j_1 - j_2)}z), \tag{12}$$

for $1 \le j_1, j_2 \le d$.

Proof.

$$\begin{aligned} & E_{w \sim p(w)}[e^{-(Wz)_{j_1}} e^{-(Wz)_{j_2}}] && (13) \\ = \; & E_{w \sim p(w)}[e^{-{}^{\uparrow j_1}(\mathrm{rev}(w))^t z} e^{-{}^{\uparrow j_2}(\mathrm{rev}(w))^t z}] && (14) \\ = \; & E_{w \sim p(w)}[e^{-w^t \mathrm{rev}({}^{\downarrow j_1}z)} e^{-w^t \mathrm{rev}({}^{\downarrow j_2}z)}] && (15) \\ = \; & E_{w \sim p(w)}[e^{-w^t (\mathrm{rev}({}^{\downarrow j_1}z) + \mathrm{rev}({}^{\downarrow j_2}z))}] && (16) \\ = \; & k(\mathrm{rev}({}^{\downarrow j_1}z) + \mathrm{rev}({}^{\downarrow j_2}z)) && (17) \\ = \; & k({}^{\downarrow j_1}z + {}^{\downarrow j_2}z) = k(z + {}^{\uparrow (j_1 - j_2)}z). && (18) \end{aligned}$$

□

Thus, the random circulant features without random sign flipping have $k(z + {}^{\uparrow(j_1 - j_2)}z) - k(z)^2$ covariance. For example, when each element in z is identical, this value is the same as the autocovariance. Thus, the variance is significant.

Next, we analyze the variance for the proposed method. We need to sum over all the random sampling of $\mathrm{circ}(w^{(l)})_{:,j}, l = 1$–$m$; which makes this problem more complex than the previous cases. We first calculate the covariance between the first and d-th elements. In this case, we can calculate the exact solution.

Theorem 3.

$$\begin{aligned} & E_{w^{(l)} \sim p(w), l_j \sim unif\{1,2,\ldots,m\}}[e^{-(Wz)_1} e^{-(Wz)_d}] && (19) \\ = & \left(\prod_{j=1}^{d} (k_1(z_j + z_{j+1}) + (m-1)k_1(z_j)k_1(z_{j+1})) \right. && (20) \\ & \left. + (m-1)\prod_{j=1}^{d}(k_1(z_j + z_{j+1}) - k_1(z_j)k_1(z_{j+1})) \right) / m^d, && (21) \end{aligned}$$

where we write $z_{d+1} = z_1$.

The ratio of the covariance to the autocovariance is roughly $(\frac{k_1(2z)+(m-1)k_1(z)^2}{mk_1(2z)})^d$, which exponentially converges toward 0 as d increases. The convergence speed is high if m is large. In fact, we vary k_1 as d increases; thus, the convergence speed does not exactly follow this order. However, this value is considerably smaller than that for random circulant features.

Proof. When we denote $u^{(l)} = \mathrm{rev}(w^{(l)})$, it holds that

$$\begin{aligned} & E_{w \sim p(w), l_j \sim \mathrm{unif}\{1,\ldots,m\}}[e^{-(Wz)_1} e^{-(Wz)_d}] && (22) \\ = & E_{w \sim p(w), l_j \sim \mathrm{unif}\{1,\ldots,m\}}[e^{-\sum_{j=1}^{d} v_j^{(l_j)} z_j} e^{-\sum_{j=1}^{d} {}^{\downarrow 1}v_j^{(l_j)} z_j}] && (23) \\ = & E_{w \sim p(w), l_j \sim \mathrm{unif}\{1,\ldots,m\}}[e^{-\sum_{j=1}^{d}\left(v_j^{(l_j)} z_j + v_j^{({}^{\uparrow 1}l_j){}^{\uparrow 1}} z_j\right)}] && (24) \\ = & E_{l_j \sim \mathrm{unif}\{1,\ldots,m\}}[\prod_{j=1}^{d} E_{w \sim p(w)}[e^{-\left(v_j^{(l_j)} z_j + v_j^{(l_{j+1})} z_{j+1}\right)}]]. && (25) \end{aligned}$$

The last equality follows from the fact that $v_j^{(l)}$ corresponding to different j are independent regardless of the values of l_j. We define the matrices $C^j \in \mathbb{R}^{m \times m}$ such that $C^j_{o,p} = E_{w^{(l)} \sim p(w)}[e^{-\left(v_j^{(o)} z_j + v_j^{(p)} z_{j+1}\right)}]$. In other words, the C^j are matrices with diagonal and non-diagonal elements are $k_1(z_j + z_{j+1})$ and $k_1(z_j)k_1(z_{j+1})$, respectively. With the C^j terms, Eq. (25) can be expressed as

$$\begin{aligned} & E_{l_j \sim \mathrm{unif}\{1,\ldots,m\}}[\prod_{j=1}^{d} E_{w \sim p(w)}[e^{-\left(v_j^{(l_j)} z_j + v_j^{(l_{j+1})} z_{j+1}\right)}]] && (26) \\ = & E_{l_j \sim \mathrm{unif}\{1,\ldots,m\}}[\prod_{j=1}^{d} C^j_{l_j, l_{j+1}}] && (27) \\ = & \frac{\mathrm{trace}(\prod_{j=1}^{d} C^j)}{m^d}, && (28) \end{aligned}$$

and the equation is derived. □

We continue to the general case. The point of the previous proof is that when we calculate the correlation by first obtaining the expectation with respect to w and then calculating the average of each l^j ranging from 1 to m, the correlation is represented as the trace of the product of d matrices that can be simultaneously diagonalized. This is because when the difference of the dimension is 1, when we bridge the dimensions that share the same $w^{(l)}$, the graph is a cyclic such as $1 \to 2 \to \cdots \to d \to 1$. When the greatest common factor between $\delta_j = j_1 - j_2$ and d is 2^g, the graph becomes an independent 2^g cyclic graph of length $d/(2^g)$ such as $1 \to 1 + \delta j \to 1 + 2\delta j \to \cdots \to 1$. The calculation for one loop is the same as the previous proof. Thus, the ratio of one loop is roughly of the order $(\frac{k_1(2z)+(m-1)k_1(z)^2}{mk_1(2z)})^{d/2^g}$. When the 2^g products are calculated, the order is $(\frac{k_1(2z)+(m-1)k_1(z)^2}{mk_1(2z)})^d$. Thus, the order is roughly the same in the general case. Therefore, the covariance is small if d is sufficiently large. We now write the expectation of the product of i-th dimension and j-th dimension of e^{-Wz} as $g(z; i, j)$.

We also derive a uniform bound for the approximation error. We follow the analysis used in (Yang et al. 2014) and (Sutherland and Schneider 2015).

Theorem 4. *Let $\mathcal{M}$ be the set of the input vector in $\mathbb{R}$ that satisfies $\|x\|_2 \leq R$ and $x_i \geq r \geq 0, i = 1, ..., d$. Then provided that $L_{p,r} = E_{w\sim p(w)}[e^{-2wr}w^2] \leq \infty$ and $\sigma^2_{\mathcal{M}} = \sup_{x,y\in\mathcal{M}} \left[\frac{1}{d^2}\sum_{i=1}^d \sum_{j=1}^d g(x+y; i, j) - k(x+y)^2\right]$, for the mapping Ψ sampled in our algorithm, we have*

$$P\left[\sup_{x,y\in\mathcal{M}} |\langle\Psi(x), \Psi(y)\rangle - k(x+y)| \geq \epsilon\right] \tag{29}$$

$$\leq \left(\left(\frac{d}{2}\right)^{\frac{2}{d+2}} + \left(\frac{2}{d}\right)^{\frac{d}{d+2}}\right)(16R)^{\frac{2d}{d+2}} e^{-\frac{\epsilon^2 D/d}{\alpha_{\epsilon,\mathcal{M}}(d+2)}}\left(\frac{4dL_{p,r}}{\epsilon^2}\right)^{\frac{d}{d+2}} \tag{30}$$

where $\alpha_{\epsilon,\mathcal{M}} = \min(1, 4\sigma^2_{\mathcal{M}} + 2\epsilon/3)$.

Proof. Let $z = x + y$, $s = D/d$, and $c(z) = \frac{1}{s}\sum_{i=1}^s g_i(z)$ where $g_i(z) = \frac{1}{d}\sum_{j=di+1}^{(d+1)i} e^{-z^t w_j}$. From the definition, $c(z) = \langle\Psi(x), \Psi(y)\rangle$. We also write $f(z) = c(z) - k(z)$. Let the range of z written as $\mathcal{M}_z = \{x + y | x, y \in \mathcal{M}\}$.

Like (Yang et al. 2014), We need to obtain high probability bound for $f(z)$ and $\sup_{z\in\mathcal{M}_z} \|\nabla f(z)\|$.

Since $c(z)$ is the mean of i.i.d. $g_i(z)$s and $0 \leq g_i(z) \leq 1$, we obtain

$$P\left[|f(z) - k(z)| > \frac{\epsilon}{2}\right] \leq 2e^{-\epsilon^2 s/2}, \tag{31}$$

by using the Hoeffding's inequality.

Since the variance of $g_i(z)$ is bounded as

$$E\left[\left(\frac{1}{d}\sum_{j=di+1}^{(d+1)i} e^{-z^t w_j}\right)^2\right] - E\left[\frac{1}{d}\sum_{j=di+1}^{(d+1)i} e^{-z^t w_j}\right]^2 \leq \sigma^2_{\mathcal{M}}, \tag{32}$$

we can obtain

$$P\left[|f(z) - k(z)| > \frac{\epsilon}{2}\right] \leq 2e^{-\frac{\epsilon^2 s/2}{4\sigma^2_{\mathcal{M}}+2\epsilon/3}}, \tag{33}$$

using Bernstein inequality. By combining above two inequality, we get

$$P\left[|f(z) - k(z)| > \frac{\epsilon}{2}\right] \leq 2e^{-\frac{\epsilon^2 s}{2\alpha_{\epsilon,\mathcal{M}}}}. \tag{34}$$

We can also get

$$P\left(\sup_{z\in\mathcal{M}_z} \|\nabla f(z)\|^2 \geq \epsilon^2/4\gamma^2\right) \leq 4\gamma^2 dL_{p,r}/\epsilon^2. \tag{35}$$

by following the same discussion as (Yang et al. 2014) because the corresponding discussion in (Yang et al. 2014) requires only the independence of w per each column that also holds true for our method.

Combining these results, we get the uniform bound with probability at least $1 - 2\left(\frac{16R}{\gamma}\right)^d e^{-\frac{\epsilon^2 s}{2\alpha_{\epsilon,\mathcal{M}}}} - 4\gamma^2 dL_{p,r}/\epsilon^2$ for positive γ. By optimizing γ, we get the bound. ☐

5 Experiments

In this section, we experimentally evaluate the accuracy and computation time of the proposed method. In Section 5.1, we evaluate the approximation error of the gram matrix. In Section 5.2 and 5.3, we evaluate the classification accuracy on image recognition datasets. In Section 5.4, the computation time for feature encoding using synthesized data is compared.

5.1 Approximation Error of Gram Matrix

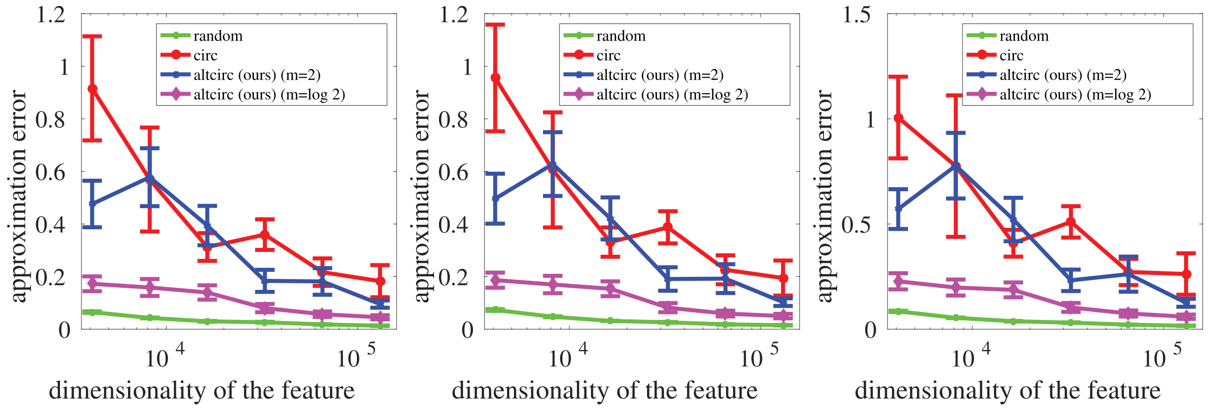
We first applied the proposed method to CNN features and evaluated the approximation error of kernel values. The method was tested on three object recognition datasets (CUB-200, Stanford Dogs, and Caltech256). CUB-200 (Welinder et al. 2010) is a standard fine-grained object recognition dataset that consists of 200 bird species with 60 images per class. Stanford Dogs (Khosla et al. 2011) consists of approximately 20,000 images of 120 dog classes, and Caltech256 (Griffin, Holub, and Perona 2007) consists of approximately 30,600 images of 256 object classes.

For the CNN model, we used the VGG-16 (Simonyan and Zisserman 2014) network pretrained with ImageNet. We employed the l_1-normalized activation of the last fully connected layer with a dimension of 4,096. The activation was positive definite because we applied the ReLU function.

As a kernel function, we used an exponential-semigroup kernel that was reported to exhibit good performance in (Yang et al. 2014), using the kernel parameter $\beta = 0.1$. The kernel function had the form $e^{-\beta\sum_{j=1}^d \sqrt{x_j+y_j}}$ and the corresponding weight distribution was a Lévy distribution $p(w) = \frac{\beta}{2\sqrt{\pi}}w^{-3/2}e^{\frac{-\beta^2}{4w}}$.

We compared random Laplace features (random), random circulant features excluding random sign flipping (circ), and the proposed method (altcirc) with $m = 2$ and $\log_2 d$. Then, we varied the output feature dimension to $4,096 \times \{1, 2, 4, 8, 16, 32\}$.

We randomly sampled 2,000 data from each dataset and computed the mean and standard error of $\frac{\|K_{\text{true}}-K_{\text{approx}}\|_F}{\|K_{\text{true}}\|_F}$,



(a) CUB (b) StanfordDogs (c) Caltech256

Figure 1: Comparison of the approximation error for the gram matrix using VGG-16 last activation.

where K_{true} is the true gram matrix, and K_{approx} denotes gram matrices calculated from random features for 10 trials.

Figure 1 shows the results. The approximation error is roughly 'random' $<$ 'altcirc ($m = \log_2 d$)' $<$ 'altcirc ($m = 2$) $<$ 'circ'. Random circulant features have a much larger error and the performance is unstable. Though the error of "random" is smaller than the error of "altcirc ($m = \log_2 d$)," "altcirc ($m = \log_2 d$)" has a small error, and its performance increases in stable fashion as D increases.

5.2 Semigroup Kernel on Bag of Visual Words

Next, we applied the proposed method to Bag of Visual Words features and evaluated its accuracy on image recognition datasets. As in Section 5.1, the method was tested on CUB-200, Stanford Dogs, and Caltech256.

We used the default train/test splits for the CUB-200 and Stanford Dogs datasets. For the Caltech256 dataset, we randomly sampled 25 images per class as training data and 30 images per class as test data.

For all datasets, we extracted dense 128-dimensional scale-invariant feature transform (SIFT) features with a step size of 2 and scales of 4, 6, 8, and 10. We used "vl_phow" implemented in VLFeat (Vedaldi and Fulkerson 2008) for extraction and then encoded the extracted SIFT to 4,096-dimensional Bag of Visual Words features. We used 250,000 local features to learn the codebooks.

As a kernel function, we used an exponential-semigroup kernel like Section 5.1 with the kernel parameter $\beta = 0.01$. We compared random Laplace features (random), random circulant features excluding random sign flipping (circ), and the proposed method (altcirc) with $m = 2$ and $\log_2 d$. Then, we varied the output feature dimension to $4,096 \times \{1, 2, 4, 8, 16\}$. We applied linear SVM implemented in LIBLINEAR (Fan et al. 2008) with $C = 100$ and evaluated the mean and standard error of the accuracy over 10 trials.

The results are listed in Table 2 and indicate that random circulant features without random sign flipping exhibited poor performance with a large standard error. Thus, the random circulant features are unstable. On the other hand, the proposed method with $m = 2$ exhibited a very similar performance to the random Laplace features. Further, the proposed method exhibited comparable performance to the random Laplace features method for $m = \log_2 d$, although the standard error was slightly larger for the former. Thus, the proposed method is a good choice for histogram feature.

5.3 Semigroup Kernel on the CNN feature

Next, we applied the proposed method to features calculated from the CNNs. We used the VGG-16 (Simonyan and Zisserman 2014) network pretrained with ImageNet. We employed the l_1-normalized activation of the last fully connected layer with a dimension of 4,096 and a 1,000-dimensional output for the softmax layer, which was embedded into a 1,024-dimensional vector with the additional dimensions filled with 0's. The activation was positive. The softmax layer output was also positive. Thus, both features satisfied the requirement for positive definiteness.

We compared the random Laplace features, random circulant features, and the proposed method with $m = 2$ and $m = \log_2 d$, corresponding to the exponential-semigroup kernel. We also evaluated the results obtained using the methods for original features (linear) and by applying random Fourier features corresponding to a Gaussian kernel (gauss) so as to verify the performance of the exponential-semigroup kernel on CNN features. As kernel parameters, we used $\beta = 0.1$ and 0.01 for the outputs of the last fully-connected and softmax layers, respectively. We used the mean of the 50-th l_2 nearest-neighbor distances of 1,000 data sampled from the training data for the Gaussian kernel, following (Yu et al. 2015). In addition, we employed the same dataset, evaluation metric, and classifier setting used in Section 5.2.

Tables 3 and 4 show the results. Table 3 indicates that, while the Gaussian kernel did not work well for the softmax output, the exponential-semigroup kernel contributed significantly to a performance improvement. The results indicate that the semigroup kernel was effective not only for histogram data but also for probability score of the softmax output. Furthermore, the proposed method with $m = \log_2 d$

Table 2: Comparison of accuracy on image recognition datasets using Bag of Visual Words.

Dataset	Method	$D = d$	$D = 2d$	$D = 4d$	$D = 8d$	$D = 16d$
CUB	random	11.20 ± 0.12	12.70 ± 0.11	14.06 ± 0.08	**14.81 ± 0.05**	15.12 ± 0.05
	circ	9.37 ± 0.90	12.15 ± 0.55	13.77 ± 0.35	14.57 ± 0.19	**15.36 ± 0.08**
	altcirc (ours) ($m = 2$)	10.95 ± 0.39	12.59 ± 0.14	14.03 ± 0.19	14.73 ± 0.11	15.27 ± 0.09
	altcirc (ours) ($m = \log_2 d$)	**11.26 ± 0.21**	**12.77 ± 0.11**	**14.09 ± 0.13**	14.71 ± 0.09	15.23 ± 0.07
Stanford Dogs	random	16.65 ± 0.10	19.23 ± 0.12	21.02 ± 0.05	22.23 ± 0.08	23.06 ± 0.08
	circ	12.50 ± 2.10	18.73 ± 0.89	**21.49 ± 0.28**	**22.49 ± 0.26**	**23.37 ± 0.13**
	altcirc (ours) ($m = 2$)	16.68 ± 0.79	**19.51 ± 0.29**	21.30 ± 0.22	22.18 ± 0.21	23.22 ± 0.15
	altcirc (ours) ($m = \log_2 d$)	**16.83 ± 0.20**	18.91 ± 0.21	21.22 ± 0.15	22.38 ± 0.10	23.03 ± 0.09
Caltech 256	random	**30.12 ± 0.15**	**32.72 ± 0.09**	34.50 ± 0.08	35.55 ± 0.08	36.06 ± 0.10
	circ	25.82 ± 1.88	31.42 ± 0.93	34.23 ± 0.35	**35.56 ± 0.23**	36.23 ± 0.12
	altcirc (ours) ($m = 2$)	29.28 ± 0.81	32.46 ± 0.23	34.51 ± 0.24	35.45 ± 0.23	**36.30 ± 0.16**
	altcirc (ours) ($m = \log_2 d$)	30.06 ± 0.27	32.55 ± 0.31	**34.60 ± 0.17**	**35.56 ± 0.16**	36.09 ± 0.16

Table 3: Comparison of accuracy on image recognition datasets using VGG-16 softmax output.

Dataset	Method	$D = d$	$D = 2d$	$D = 4d$	$D = 8d$	$D = 16d$
CUB	random	**36.07 ± 0.30**	**39.48 ± 0.11**	**40.91 ± 0.12**	**42.03 ± 0.08**	**42.55 ± 0.07**
	circ	33.29 ± 2.55	35.26 ± 2.18	37.88 ± 2.03	40.35 ± 1.14	41.20 ± 0.92
	altcirc (ours) ($m = 2$)	33.76 ± 2.08	36.95 ± 1.95	39.11 ± 1.09	40.77 ± 0.89	42.10 ± 0.63
	altcirc (ours) ($m = \log_2 d$)	36.05 ± 0.68	37.50 ± 1.09	**40.91 ± 0.43**	41.86 ± 0.21	42.46 ± 0.20
	gauss	23.07 ± 0.06	23.34 ± 0.05	23.38 ± 0.06	23.39 ± 0.03	23.44 ± 0.04
	linear	22.83				
Stanford Dogs	random	78.81 ± 0.04	**79.13 ± 0.02**	**79.22 ± 0.04**	79.28 ± 0.04	79.29 ± 0.04
	circ	78.50 ± 0.30	78.93 ± 0.13	79.02 ± 0.11	79.31 ± 0.04	79.33 ± 0.03
	altcirc (ours) ($m = 2$)	78.75 ± 0.19	79.00 ± 0.12	79.21 ± 0.07	79.29 ± 0.02	**79.35 ± 0.03**
	altcirc (ours) ($m = \log_2 d$)	**78.86 ± 0.08**	79.08 ± 0.06	79.18 ± 0.05	**79.34 ± 0.04**	79.34 ± 0.02
	gauss	78.07 ± 0.03	78.03 ± 0.04	78.02 ± 0.02	78.00 ± 0.03	78.03 ± 0.01
	linear	78.12				
Caltech 256	random	**70.04 ± 0.21**	**71.52 ± 0.18**	**72.25 ± 0.21**	**72.57 ± 0.15**	**72.74 ± 0.14**
	circ	67.35 ± 1.19	69.20 ± 1.02	70.47 ± 0.96	71.84 ± 0.51	72.32 ± 0.36
	altcirc (ours) ($m = 2$)	68.54 ± 0.95	69.94 ± 0.93	71.45 ± 0.51	72.10 ± 0.36	72.69 ± 0.20
	altcirc (ours) ($m = \log_2 d$)	69.78 ± 0.35	70.73 ± 0.32	72.01 ± 0.21	72.53 ± 0.16	72.71 ± 0.18
	gauss	58.48 ± 0.20	58.72 ± 0.22	58.65 ± 0.23	58.65 ± 0.24	58.69 ± 0.24
	linear	57.90				

worked as well as the random Laplace features approach. In Table 4, when the output feature dimension was small, the linear method and the Gaussian kernel worked well. However, as the dimension grew, the approximation method for the semigroup kernel exhibited better performance. Thus, we can state that the semigroup kernel is superior to the Gaussian kernel for encoding the activation output of the CNNs when the approximation is sufficiently accurate.

These results demonstrate that the proposed method is effective for encoding CNN features.

5.4 Computation Time

Finally, we compared the feature-encoding computation time using synthesized data, with each element sampled from a uniform distribution ranging from 0 to 1. We varied the dimension of the input feature d to 1,024, 2,048, 4,096, 8,192 and 16,384, and set the dimension of the output feature $D = d$. We compared the computation time required to encode one input vector using an Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz. We implemented each method using Matlab with the "-singleCompThread" option.

Table 5 shows that the overall performance roughly follows the order given in Table 1. The proposed method requires significantly less computation time than the random Laplace features method. Although the proposed method with $m = \log_2 d$ is slower than the proposed method with $m = 2$, even with $m = \log_2 d$, it is approximately 100 times as fast as the random Laplace features method when the dimension is 16,384. Thus, the proposed method is efficient.

6 Conclusion

In this study, we proposed alternating circulant random features for application to semigroup kernels. The proposed method randomly mixes the circulant random features. We analyzed the covariance of the proposed features and showed that this value is small when the dimension of the global feature is large. From experiments on image recognition datasets using histogram and CNN features, we demonstrated that the semigroup kernel is effective on a wide range of positive definite features, and that the proposed method exhibits comparable performance to, with much lower computation time than, the original random Laplace features. The proposed method enabled us to apply the corresponding semigroup kernels very efficiently.

Table 4: Comparison of accuracy on image recognition datasets using VGG-16 last activation.

Dataset	Method	$D = d$	$D = 2d$	$D = 4d$	$D = 8d$	$D = 16d$
CUB	random	55.00 ± 0.16	56.89 ± 0.12	57.94 ± 0.07	58.43 ± 0.09	58.74 ± 0.04
	circ	54.14 ± 0.94	56.80 ± 0.36	57.68 ± 0.19	58.45 ± 0.08	58.72 ± 0.06
	altcirc (ours) ($m = 2$)	54.85 ± 0.34	56.63 ± 0.22	57.75 ± 0.16	$\mathbf{58.57 \pm 0.07}$	$\mathbf{58.94 \pm 0.05}$
	altcirc (ours) ($m = \log_2 d$)	54.92 ± 0.14	56.75 ± 0.12	$\mathbf{57.81 \pm 0.06}$	58.39 ± 0.09	58.68 ± 0.05
	gauss	$\mathbf{56.88 \pm 0.11}$	$\mathbf{57.60 \pm 0.08}$	57.75 ± 0.06	57.84 ± 0.03	57.87 ± 0.05
	linear	56.47				
Stanford Dogs	random	75.45 ± 0.06	76.59 ± 0.09	77.20 ± 0.05	77.52 ± 0.06	$\mathbf{77.73 \pm 0.03}$
	circ	75.75 ± 0.27	$\mathbf{76.90 \pm 0.08}$	$\mathbf{77.44 \pm 0.07}$	$\mathbf{77.65 \pm 0.06}$	77.72 ± 0.04
	altcirc (ours) ($m = 2$)	75.70 ± 0.13	76.75 ± 0.07	77.24 ± 0.07	77.47 ± 0.05	77.71 ± 0.05
	altcirc (ours) ($m = \log_2 d$)	75.45 ± 0.11	76.48 ± 0.07	77.20 ± 0.07	77.53 ± 0.04	77.64 ± 0.03
	gauss	76.02 ± 0.05	76.23 ± 0.06	76.37 ± 0.05	76.39 ± 0.03	76.44 ± 0.04
	linear	**77.73**				
Caltech 256	random	75.42 ± 0.17	76.48 ± 0.16	77.15 ± 0.15	$\mathbf{77.54 \pm 0.15}$	77.65 ± 0.16
	circ	73.07 ± 1.64	76.03 ± 0.57	76.81 ± 0.40	77.45 ± 0.17	77.60 ± 0.14
	altcirc (ours) ($m = 2$)	75.01 ± 0.48	76.25 ± 0.28	$\mathbf{77.16 \pm 0.14}$	77.44 ± 0.16	77.66 ± 0.16
	altcirc (ours) ($m = \log_2 d$)	75.33 ± 0.18	76.53 ± 0.15	77.11 ± 0.13	77.44 ± 0.18	$\mathbf{77.70 \pm 0.16}$
	gauss	$\mathbf{76.32 \pm 0.20}$	$\mathbf{76.60 \pm 0.21}$	76.78 ± 0.19	76.83 ± 0.18	76.85 ± 0.19
	linear	75.50				

Table 5: Computation time (second) on synthesized data.

Method	1,024	2,048	4,096	8,192	16,384
random	3.4e-3	1.5e-2	6.2e-2	2.5e-1	1.0
altcirc (ours) ($m = 2$)	**3.6e-4**	**3.7e-4**	**4.8e-4**	**9.6e-4**	**1.7e-3**
altcirc (ours) ($m = \log_2 d$)	6.3e-4	8.5e-4	1.8e-3	4.0e-3	8.6e-3

We have applied our method to semigroup kernels. However, the concept of mixing random features can be applied to a wider domain. Thus, we will extend this concept to a broader range of kernels in future work.

Acknowledgements

This work was supported by CREST, JST and JSPS KAKENHI Grant Number JP17J05725.

References

Berg, C.; Christensen, J. P. R.; and Ressel, P. 1984. Harmonic analysis on semigroups.

Choromanski, K., and Sindhwani, V. 2016. Recycling randomness with structure for sublinear time kernel expansions. In *ICML*.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. Liblinear: A library for large linear classification. *JMLR* 9(Aug):1871–1874.

Felix, X. Y.; Suresh, A. T.; Choromanski, K. M.; Holtmann-Rice, D. N.; and Kumar, S. 2016. Orthogonal random features. In *NIPS*.

Griffin, G.; Holub, A.; and Perona, P. 2007. Caltech-256 object category dataset.

Kar, P., and Karnick, H. 2012. Random feature maps for dot product kernels. In *AISTATS*.

Khosla, A.; Jayadevaprakash, N.; Yao, B.; and Fei-Fei, L. 2011. Novel dataset for fine-grained image categorization. In *CVPR workshop on FGVC*.

Le, Q.; Sarlós, T.; and Smola, A. 2013. Fastfood–approximating kernel expansions in loglinear time. In *ICML*.

Pennington, J.; Felix, X. Y.; and Kumar, S. 2015. Spherical random features for polynomial kernels. In *NIPS*.

Pham, N., and Pagh, R. 2013. Fast and scalable polynomial kernels via explicit feature maps. In *KDD*.

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

Rudin, W. 2011. *Fourier analysis on groups*. John Wiley & Sons.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Sutherland, D. J., and Schneider, J. 2015. On the error of random fourier features. In *UAI*.

Vedaldi, A., and Fulkerson, B. 2008. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/.

Welinder, P.; Branson, S.; Mita, T.; Wah, C.; Schroff, F.; Belongie, S.; and Perona, P. 2010. Caltech-ucsd birds 200. Technical Report CNS-TR-201, Caltech.

Yang, J.; Sindhwani, V.; Fan, Q.; Avron, H.; and Mahoney, M. W. 2014. Random laplace feature maps for semigroup kernels on histograms. In *CVPR*.

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