# Chebyshev Approximations to the Histogram $\chi^2$ Kernel

Fuxin Li, Guy Lebanon
Georgia Institute of Technology
{fli,lebanon}@cc.gatech.edu

Cristian Sminchisescu
University of Bonn
cristian.sminchisescu@ins.uni-bonn.de

## Abstract

*The random Fourier features methodology can be used to approximate the performance of kernel classifiers in linear time in the number of training examples. However, there still exists a non-trivial performance gap between the approximation and the nonlinear kernel classifiers, especially for the exponential $\chi^2$ kernel, one of the most powerful models for histograms. Based on analogies with Chebyshev polynomials, this paper proposes an asymptotically convergent analytic series of the $\chi^2$ kernel that is used in the random Fourier approximation of the exponential $\chi^2$ kernel. The new series removes the need to use periodic approximations to the $\chi^2$ function, as typical in previous methods, and improves the classification accuracy. Besides, out-of-core principal component analysis (PCA) methods are introduced, which can reduce the dimensionality of the approximation and achieve better performance, with only an additional constant factor to the time complexity. Especially, when PCA is performed jointly on the training and unlabeled testing data, a further performance improvement is achieved. The proposed approaches are tested on the PASCAL VOC 2010 segmentation and the ImageNet ILSVRC 2010 datasets, and give statistically significant improvements over previous approximation methods.*

## 1. Introduction

Random Fourier (RF) features [23, 25, 24, 18] is a promising methodology for large-scale classification. It uses Monte Carlo sampling in the frequency domain to construct an embedding such that, linear functions on the embedding are asymptotically convergent approximations to the nonlinear functions attainable in kernel methods. The benefit of this transition is that now the time complexity of many learning methods will be linear in the number of examples $n$, compared to at least $O(n^{2.3})$ for the kernel methods. Therefore, RF makes possible to use complicated nonlinear learning models in the massive datasets that are increasingly common nowadays. RF also enjoys most of the learning rate and generalization results of kernel methods, for instance, local Rademacher bounds in [2]. These benefits raise the question whether the slower kernel formulation can be avoided while preserving its predictive power.

Unfortunately at least in the visual recognition community, the current answer is still no. In practice there seems to be a nontrivial performance difference between RF approaches and kernel approaches. Although this gap (0.5% – 4%) is usually not large [18], it is still too significant to ignore. Multi-stage methods still play a major role for object detection [20], where RF and more expensive kernel methods can be used as two consecutive stages [7].

This paper aims to reduce the approximation gap without losing the advantageous $O(n)$ time complexity. The two main contributions are: (a) we propose a new convergent analytic series for the $\chi^2$ distance commonly used for histogram features, and (b) we exploit principal component analysis (PCA) on the obtained random features, in order to improve performance without additional complexity.

The starting point of our exploration is the two-stage approximation of the exponential chi-square kernel (exp-$\chi^2$)

$$k(x, y) = \exp(-\chi^2(x, y)) \tag{1}$$

proposed in [24]. Empirically we found that this has the best performance in visual recognition, over all the RF kernel approximations that have been proposed so far. The two-stage method [24] first uses the Fourier transform on the non-negative orthant to approximate the $\chi^2$ distance as an inner product. Then another standard RF for the Gaussian kernel is used to approximate the final exp-$\chi^2$.

Previous inner-product approximation for the $\chi^2$ distance [25] relied on a periodic version of the function. The additional periodicity parameter is rather sensitive. Even if well-tuned, the approximation quality can deteriorate when the histograms are out of the periodic range [26]. In this paper derive an analytic recurrence formula to obtain asymptotically convergent approximations to the $\chi^2$ distance. Experiments show that the new convergent approximations obtain better performance than existing periodic methods.

In addition, in order to obtain more compact embeddings for large-scale learning when the data cannot fit into memory, we exploit an out-of-core version of PCA that adds little computational overhead to the RF approximation, especially when combined with least squares and other methods based on quadratic loss (e.g. group LASSO). PCA allows us to reduce the number of dimensions required for classification, and relaxes memory constraints when multiple kernels have to be approximated by RF. We also explore using the unlabeled test set to better estimate the covariance matrix in PCA, leading to a better selection of the frequency components and improved classification performance.

## 2. Related Work

Speed-ups to kernel methods based on low-rank approximations of the kernel matrix have been proposed before [13, 1]. These methods are effective, but applying the kernel machine on new data requires slow kernel computations between the test and training examples. An alternative is to use the Nyström methods [28] that sub-samples the training set and operate on a reduced kernel matrix. Although this works well in practice, the asymptotic convergence rate of this approximation is slow: $O(n^{-\frac{1}{4}})$ [11], where *n* is the number of examples used for the approximation.

A topic of recent interest is coding image features. The goal of such methods is to achieve good performance with linear classification or regression after coding the features [27, 15]. Hierarchical coding schemes with deeper structures have also been proposed [16]. Both sparse and dense coding schemes have proved successful. In fact, the supervector coding [19] and the Fisher kernels [21] have been the best performers in the ImageNet large-scale image classification challenge [10]. Comparing between coding methods and RF, we note that RF usually starts with bag-of-word vector quantization while coding schemes sometimes start with raw image features and therefore have an extra layer of processing freedom. Nevertheless, replacing hard clustering with a soft-assignment clustering may improve the performance of the histogram method to become on par with some coding schemes [8]. Alternatively, one can use a Gaussian matching kernel approximated with RF, instead of comparing bins independently [3].

The dictionaries of the aforementioned coding schemes are usually extremely large (e.g., both the Fisher kernel and supervector coding usually require more than 200k dimensions [8]) and the generation of the dictionary is often extremely time-consuming. RF is theoretically guaranteed to approximate the kernel model with a reasonable asymptotic convergence rate [23]. It would neither require too many dimensions nor training for the dictionaries. Therefore it is well worth exploring as an alternative approach.

## 3. The Chebyshev Approximation

Throughout this paper we use $X$ to denote the training set with $n$ training examples and $d$ dimensions. $D$ denotes the number of random features after the RF embedding. The all-ones vector is described by $\mathbf{1}$, the all-zeros by $\mathbf{0}$, and the imaginary unit by $j$. $*$ is used for complex conjugate. All kernels are positive semi-definite kernels.

In [25], the class of $\gamma$-*homogeneous* kernels is introduced:

$$k(cx, cy) = c^{\gamma} k(x, y), \forall c \geq 0. \tag{2}$$

Choosing $c = \frac{1}{\sqrt{xy}}$, a $\gamma$-*homogeneous* kernel can be written as:

$$\begin{aligned} k(x,y) &= c^{-\gamma} k(cx, cy) = (xy)^{\frac{\gamma}{2}} k(\sqrt{\frac{y}{x}}, \sqrt{\frac{x}{y}}) \\ &= (xy)^{\frac{\gamma}{2}} \mathcal{K}(\log y - \log x) \end{aligned} \tag{3}$$

where $\mathcal{K}$ is an even function, i.e., $\mathcal{K}(-x) = \mathcal{K}(x)$.

Denoting $\Delta = \log y - \log x$, the $1 - \chi^2$ kernel is

$$k_0(x, y) = 1 - \sum_i \frac{(x_i - y_i)^2}{x_i + y_i} = \sum_i \frac{2x_i y_i}{x_i + y_i} \tag{4}$$

(assuming $\sum_i x_i = 1$). In each dimension we have

$$k_0(x, y) = \frac{2xy}{x + y} = \sqrt{xy} \frac{2}{\sqrt{\frac{x}{y}} + \sqrt{\frac{y}{x}}} = \sqrt{xy} \operatorname{sech}(\frac{\Delta}{2}), \tag{5}$$

where $\operatorname{sech}(x) = \frac{2}{e^x + e^{-x}}$ is the hyperbolic secant function whose Fourier transform is $\pi \operatorname{sech}(\pi\omega)$. Using the inverse Fourier transform to map $\pi \operatorname{sech}(\pi\omega)$ back to $k_0(x, y)$

$$\begin{aligned} k_0(x,y) &= \sqrt{xy} \int_{-\infty}^{\infty} e^{j\omega(\log x - \log y)} \operatorname{sech}(\pi\omega) d\omega \\ &= \int_{-\infty}^{\infty} \Phi_\omega(x)^* \Phi_\omega(y) d\omega \end{aligned} \tag{6}$$

where $\Phi_\omega(x) = \sqrt{x} e^{-j\omega \log x} \sqrt{\operatorname{sech}(\pi\omega)}$.

In [25], the function $e^{-j\omega \log x} \operatorname{sech}(\pi\omega)$ is approximated with a periodic function, which is then approximated with finite Fourier coefficients (hereafter called the VZ approximation as a shorthand for Vedaldi-Zisserman). However, $e^{-j\omega \log x} \operatorname{sech}(\pi\omega)$ is inherently aperiodic. As a consequence the approximation error is low when $|\log x|$ is small, but excessively high when $|\log x|$ is larger than the period. Convergence is attained in [25] because the introduced aperiodic bias is cancelled with the factor $\sqrt{xy}$ when $x$ or $y$ is small. However, uneven biases in different regions may impact performance in learning. Here we pursue an alternative derivation that is analytic and asymptotically convergent, even without the factor $\sqrt{xy}$. We describe the main ideas below and provide more details in the supplementary material.

Because the kernel is symmetric the imaginary part of the inverse Fourier transform is 0, leading to

$$\begin{aligned} k_0(x,y) &= \sqrt{xy} \int_{-\infty}^{\infty} \cos(\omega(\log x - \log y)) \operatorname{sech}(\pi\omega) d\omega \\ &= \sqrt{xy} \int_{-\infty}^{\infty} (\cos(\omega \log x) \cos(\omega \log y) \\ &\quad + \sin(\omega \log x) \sin(\omega \log y)) \frac{2}{e^{\pi\omega} + e^{-\pi\omega}} d\omega. \end{aligned} \tag{7}$$

Using a change of variable $z = 2\arctan e^{\pi\omega}$, the integral becomes

$$k_0(x,y) = \tag{8}$$
$$\frac{\sqrt{xy}}{\pi}\int_0^{\pi}(\cos(\frac{1}{\pi}\log|\tan\frac{z}{2}|\log x)\cos(\frac{1}{\pi}\log|\tan\frac{z}{2}|\log y)$$
$$+\sin(\frac{1}{\pi}\log|\tan\frac{z}{2}|\log x)\sin(\frac{1}{\pi}\log|\tan\frac{z}{2}|\log y))dz.$$

Since the functions $\cos(\frac{1}{\pi}\log|\tan\frac{z}{2}|\log x)$ and $\sin(\frac{1}{\pi}\log|\tan\frac{z}{2}|\log x)$ are periodic and even, they can be represented using discrete-term Fourier cosine series

$$f_x(z) = \frac{a_0(x)}{2} + \sum_{n=1}^{N} a_n(x)\cos(nz). \tag{9}$$

Since for all integers $n$ and $m$,

$$\int_0^{\pi}\cos(nx)\cos(mx)dx = \begin{cases} 0 & n \neq m \\ \pi/2 & n = m \end{cases},$$

we have

$$\frac{1}{\pi}\int_0^{\pi} f_x(z)f_y(z)dz = \frac{a_0(x)a_0(y)}{4} + \frac{1}{2}\sum_i a_i(x)a_i(y) \tag{10}$$

which offers a natural orthogonal decomposition. A vector $a_x = \frac{1}{\sqrt{2}}[a_0(x)/\sqrt{2}, a_1(x), a_2(x), \ldots, a_n(x)]$ guarantees that $a_x^T a_y = \frac{1}{\pi}\int_0^{\pi} f_x(z)f_y(z)dz$.

Obtaining the coefficients require computing the integrals $\int_0^{\pi}\cos(\frac{1}{\pi}\log(\tan\frac{z}{2})\log x)$ and $\int_0^{\pi}\sin(\frac{1}{\pi}\log(\tan\frac{z}{2})\log x)$. Since these functions are symmetric/antisymmetric, half of the coefficients in either cosine series are zero. Therefore, we can combine the two series into a new one, $c_n(x)$, which contains the even Fourier coefficients from the cosine term and the odd coefficients from the sine term.

In fact, using integration-by-parts we can represent the coefficients from one series (either the cosine term or the sine term) by coefficients of the other series. After some algebraic derivations (see supplementary material) we obtain the following analytic form of $c_n(x)$:

$$c_k(x) = \begin{cases} \frac{1}{k}((-1)^k\frac{2\log x}{\pi}c_{k-1}(x) + (k-2)c_{k-2}(x)), & k > 1 \\ -\frac{\sqrt{2}\log x}{\pi}c_0(x), & k = 1 \\ \frac{2x}{x+1}, & k = 0 \end{cases} \tag{11}$$

with $k_0(x,y) = \sum_k c_k(x)c_k(y)$.

Applying the calculation for all dimensions yields the new Fourier embedding for the $\chi^2$ kernel. Then, we follow [24] and use RF to approximate a Gaussian kernel on $c(x)$, to obtain the approximation of the exp-$\chi^2$ kernel $k(x,y) = \exp(-\gamma\chi^2(x,y))$. The complete procedure is presented in Algorithm 1. We name the above algorithm as the Chebyshev approximation because it draws ideas from Chebyshev polynomials and the Clenshaw-Curtis quadrature [4]. A central idea in the Clenshaw-Curtis quadrature is to use the change of variable $\theta = \arccos(x)$ in order to convert an aperiodic integral into a periodic one, enabling the application of Fourier techniques. Our variable substitution $z = \arctan e^x$ serves a similar purpose. The same technique can be applied in principle to other kernels, such as the histogram intersection and the Jensen-Shannon kernel. However, the analytical approximation due to integration-by-parts may not extend easily. We plan to pursue these extensions in the future.

**Algorithm 1** Approximation of the exp-$\chi^2$ kernel based on the Chebyshev approximation of the $\chi^2$ distance.

**input** : $n \times d$ data matrix $X = [X_1^T, X_2^T, \ldots, X_n^T]^T$. Parameters $m, D$.

**output** : The random Fourier feature $Z$ of the exp-$\chi^2$ kernel.

1: Compute for $k = 0, \ldots, m-1$

$$c_k(x_{ij}) = \begin{cases} \frac{1}{k}((-1)^k\frac{2\log x_{ij}}{\pi}c_{k-1}(x_{ij}) \\ +(k-2)c_{k-2}(x_{ij})), & k > 1 \\ -\frac{\sqrt{2}\log x_{ij}}{\pi}c_0(x_{ij}), & k = 1 \\ \frac{2x_{ij}}{x_{ij}+1}, & k = 0 \end{cases}$$

for each dimension $j$ of each example $x_i$. Denote $c(X_i)$ the $md \times 1$ vector constructed by concatenating all $c_k(x_{ij}), j = 1, \ldots, d$.

2: Construct a $md \times D$ matrix $\Omega$, where each entry is sampled from a normal distribution $\mathcal{N}(0, 2\gamma)$.

3: Construct a $D \times 1$ vector $b$ which is sampled randomly from $[0, 2\pi]^D$.

4: $Z_i = \cos(c(X_i)\Omega + b)$ is the RF feature for $X_i$ [23].

## 4. Convergence Rate of the Chebyshev Approximation

In this section we present an analysis on the asymptotic convergence rate of the Chebyshev approximation. Since (11) is exact, we can apply standard results on Fourier series coefficients [4], which says the convergence rate depends on the smoothness of the function that is approximated.

**Lemma 1.** *$|k_0(x_i,y_i) - \sum_{k=1}^m c_k(x_i)c_k(y_i)| \leq \frac{C}{m}\sqrt{x_i y_i}$ where $C$ is a constant.*

*Proof.* Since $\frac{c_m(x_i)}{\sqrt{x_i}}$ represents Fourier series for $\cos(\frac{1}{\pi}\log|\tan\frac{z}{2}|\log x_i)$ and $\sin(\frac{1}{\pi}\log|\tan\frac{z}{2}|\log x_i)$, which are both absolutely continuous but not continuously differentiable (it oscillates at $z = 0$), we have:

$$0 < mc_m(x_i) \leq \sqrt{C}\sqrt{x_i} \tag{12}$$

and consequentially

$$|k_0(x_i,y_i) - c(x_i)^T c(y_i)| \leq \sum_{k>m}\frac{C}{m^2}\sqrt{x_i y_i} \leq \frac{C}{m}\sqrt{x_i y_i} \quad \square$$

Using Lemma 1 it is straightforward to prove that

**Theorem 1.** *$|k_0(x,y) - \sum_i \sum_{k=1}^m c_k(x_i)c_k(y_i)| \leq \frac{C}{m}$ when $\sum_i x_i = \sum_i y_i = 1$.*

*Proof.* Use Cauchy-Schwarz inequality,
$|k_0(x,y) - \sum_i \sum_{k=1}^m c_k(x_i)c_k(y_i)| \leq \frac{C}{m}\sum_i \sqrt{x_i y_i}$
$\leq \frac{C}{m}\sqrt{\sum_i x_i \sum_i y_i} = \frac{C}{m}$. ☐

Although our method converges slower than the VZ approximation, our convergence is independent on the factors $\sqrt{x_i y_i}$. When $x_i$ or $y_i$ is small, the VZ approximation can only guarantee $\frac{k_0(x_i,y_i)}{\sqrt{xy}} \leq C_1$ where $C_1$ is a constant close to 1, but our approximation can guarantee $\frac{k_0(x_i,y_i)}{\sqrt{xy}} \leq \frac{C}{m}$. In this case we perform much better than VZ. Since the image histograms considered in this work often consist many small values instead of a few large ones, our alternative approximation is expected to work slightly better in general.

We can numerically simulate the constant $C$ for different $x$ values by computing the empirical bound $\max_m \frac{mc_m}{\sqrt{x}}$. The simulation results with $100,000 \leq m \leq 500,000$ are presented in Figure 1. It can be seen that the approximation is more accurate if the input values are larger, however, the error on the smaller input values can be offset by the $\sqrt{x}$ factor, making the effective constant small in all cases.

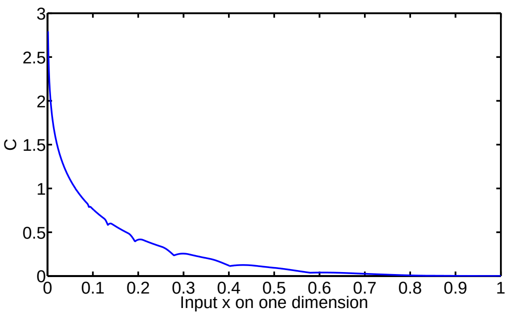


Figure 1. A plot of the $C$ in Theorem 1 for different input values. The $L_1$ error of the kernel approximation is decided by $C$ (Theorem 1). The value $C$ is large when the histogram value is small, which can be offset by the $\sqrt{x}$ factor multiplied to it.

## 5. Principal Component Analysis of Random Features on Multiple Descriptors

Another rather orthogonal strategy we pursue is principal component analysis after obtaining random features. This is useful for reducing the memory footprint when multiple image descriptors are used (common in computer vision, e.g. [14]) and RF embeddings are computed for each of them. It is known that the performance of RF improves when more random dimensions are used. However, when the RF of multiple kernels are concatenated: e.g. with 7 kernels and 7,000 RF dimensions for each kernel, the learning phase following RF needs to operate on a 49,000 dimensional feature vector. In most cases, the speed of learning algorithms deteriorates quickly when the data cannot be load in memory. PCA is a natural choice to use fewer dimensions for an approximation of the same quality. In fact, it is one of the very few possible choices in high dimensions, since many other techniques like quasi-Monte Carlo suffer from the curse of dimensionality – the convergence rate decreases exponentially with the number of dimensions [5], which makes them unsuitable for RF where many dimensions are needed.

Another interesting aspect of RF-PCA is it can bring an unexpected flavor of semi-supervised learning, in that one can use unlabeled test data to improve classification accuracy. RF-PCA amounts to selecting the relevant dimensions in the frequency space. By considering both the training and testing data during PCA, frequencies that help discriminate test data will more likely be selected. In the experiments such a strategy will be shown to improve performance over the computation of PCA only on training data.

**Algorithm 2** Out-of-Core Principal Component Analysis.

**input** : $n \times d$ data matrix $X = [X_1^T, X_2^T, \ldots, X_n^T]^T$. Output vector $y$. Number of dimension $D$ to retain after PCA.

Divide the data into $k$ chunks, called $X_{(1)}, X_{(2)}, \ldots, X_{(k)}$.
$H = \mathbf{0}, m = \mathbf{0}, v = \mathbf{0}$
**for** $i = 1 \to k$ **do**
  Load the $i$-th chunk $X_{(i)}$ into memory.
  Use Algorithm 1 to compute the RF feature $Z_{(i)}$ for $X_{(i)}$.
  $H = H + Z_{(i)}^T Z_{(i)}, m = m + Z_{(i)}^T 1, v = v + Z_{(i)}^T y$
**end for**
$H = H - \frac{1}{n}mm^T$.
Compute eigen-decomposition $H = U\mathbf{D}U^T$. Output the first $D$ columns of $U$ as $\bar{U}$, the diagonal matrix $\mathbf{D}$, and the input-output product $v$.

The main problem is in large-scale datasets, the data cannot be fully loaded into memory. Therefore PCA needs to be performed out-of-core, a high-performance computing term depicting this situation (unable to load data into memory). As have been discussed extensively in the high-performance computing literature (e.g., [22]), the way to perform out-of-core PCA in linear time is not by singular value decomposition on the RF features $Z$, but rather by performing eigenvalue decomposition for the centered covariance matrix $Z^T(I - \frac{1}{n}\mathbf{11}^T)Z$, which can be computed out-of-core by just loading a chunk of $X_i$ into memory at a time, compute their RF feature $Z$, compute the covariance matrix and then delete the RF features from memory. Then an eigen-decomposition gives the transformation matrix $U$ for PCA. We denote $\bar{U}$ as the matrix obtained by selecting the first $D$ dimensions of $U$ corresponding to the largest eigenvalues (Algorithm 2). Denote the mean vector of the input matrix $\bar{Z} = \frac{1}{n}Z^T\mathbf{1}$, then

$$\tilde{Z} = (Z - \mathbf{1}\bar{Z}^T)\bar{U} = (I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)Z\bar{U} \tag{13}$$

is the feature vector obtained after PCA projection.

It is very convenient to perform regression with a quadratic loss after PCA, since only the Hessian is needed for optimization. This applies not only to traditional least squares regression, but also to the LASSO, group LASSO, and other composite regularization approaches. In this case the projections need not be performed explicitly. Instead, notice that only $\tilde{Z}^T\tilde{Z}$ and $\tilde{Z}^T y$ are needed for regression:

$$\begin{aligned}\tilde{Z}^T\tilde{Z} &= \bar{U}^T Z^T (I - \frac{1}{n}\mathbf{1}\mathbf{1}^T) Z\bar{U} \\ \tilde{Z}^T y &= \bar{U}^T Z^T (I - \frac{1}{n}\mathbf{1}\mathbf{1}^T) y\end{aligned} \tag{14}$$

It follows that only $Z^TZ$, $Z^T\mathbf{1}$ and $Z^Ty$ have to be computed. All terms can be computed out-of-core simultaneously. Algorithm 3 depicts this scenario. Under this PCA approach the data is loaded only once to compute the Hessian. Additional complexity of $O(D^3)$ is necessary for matrix decomposition on $H$. If ridge regression is used, the $H'$ after decomposition is diagonal therefore only $O(D)$ is needed to obtain the regression results. The bottleneck of this algorithm for large-scale problems is undoubtedly the computation of the initial Hessian, which involves reading multiple chunks from disk.

**Algorithm 3** Learning after PCA with Quadratic Loss.

**input** : $n \times d$ data matrix $X = [X_1^T, X_2^T, \ldots, X_n^T]^T$. Output vector $y$. Number of dimension $D$ to retain after PCA.
1: Perform out-of-core PCA using Algorithm 2.
2: $H' = \bar{U}^T H \bar{U} = \bar{\mathbf{D}}$, the first $D$ rows and columns of the diagonal matrix $\mathbf{D}$.
3: $v' = \bar{U}^T v - \frac{1}{n}(\mathbf{1}^T y)\bar{U}^T m$.
4: Perform learning on $\bar{\mathbf{D}}, v'$, e.g., for linear ridge regression where the optimization is $\arg\min_w \|w^T\tilde{Z} - y\|^2 + \lambda\|w\|^2$, the solution is $w = (\bar{\mathbf{D}} + \lambda I)^{-1}v'$.
5: Use $\bar{U}^T w$ instead of $w$ as a function of the original inputs: $f(x) = w^T\bar{U}x - \frac{1}{n}w^T\bar{U}m$, in order to avoid the projection for the testing examples.

The more sophisticated case is when PCA needs to be performed separately on multiple different kernel approximators, i.e., $Z = [Z^{(1)} Z^{(2)} \ldots Z^{(l)}]$, where each $Z^{(i)}$ is the RF feature embedding of each kernel. This time, the need to compute ${Z^{(i)}}^T Z^{(j)}$ rules out tricks for simple computation. The data needs to be read in twice (Algorithm 4), first to perform the PCA, and then use $U$ to transform $X$ in chunks in order to obtain $Z$ and $Z^TZ$. But the full computation is still linear in the number of training examples.

In both cases, the projection is not required for the testing examples. Because whenever $w$ is obtained, $w^T\tilde{Z} = w^T\bar{U}(Z - \frac{1}{n}\bar{Z}1^T)$, then $\bar{U}w$ can be the weight vector for the original input, with the addition of a constant term.

**Algorithm 4** Two-stage Principal Component Analysis when learning with multiple kernels.

**input** : $n \times d$ data matrix $X = [X_1^T, X_2^T, \ldots, X_n^T]^T$. Output vector $y$. Number of dimension $D$ to retain after PCA.
1: Perform out-of-core PCA using Algorithm 2.
2: **for** $i = 1 \to k$ **do**
3: Load the $i$-th chunk $X_{(i)}$ into memory.
4: Use Algorithm 1 to compute the RF feature $Z_{(i)}$ for $X_{(i)}$, with the same randomization vectors $w$ as before.
5: $\tilde{Z} = (Z_{(i)} - \frac{1}{n}\mathbf{1}m^T)\bar{U}$.
6: $H' = H' + \tilde{Z}^T\tilde{Z}$, $v' = v' + \tilde{Z}^T y$
7: **end for**
8: Perform learning on $H', v'$. E.g., for linear least squares where the optimization is $\arg\min_w \|w^T Z - y\|^2$, the solution is $w = H'^{-1}v'$.
9: Use $\bar{U}^T w$ instead of $w$ as a function of the original inputs: $f(x) = w^T\bar{U}x - \frac{1}{n}w^T\bar{U}m$, in order to avoid the projection step for the testing examples.

We note that out-of-core least squares or ridge regression scales extremely well with the number of output dimensions $c$, which can be used to solve one-against-all classification problems with $c$ classes. In Algorithm 2 or 4, $Z^Ty$ will be computed in $O(nDc)$ time along with the Hessian. After the inverse of Hessian is obtained, only a matrix-vector multiplication costing $O(D^2c)$ is needed to obtain all the solutions, without any dependency on $n$. Thus the total time of this approach with $c$ classes is $O(nDc + D^2c)$ which scales very nicely with $c$. Especially compared with other algorithms that need to perform the full training procedure on each class. Although the $L_2$ loss is not optimal for classification, in large-scale problems (e.g. ImageNet) with $1,000-10,000$ classes, the out-of-core ridge regression can still be used to generate a fairly good baseline result quickly.

## 6. Experiments

Our experiments are conducted on two extremely challenging datasets, the PASCAL VOC 2010 [12] and the ImageNet [10] ILSVRC 2010 (http://www.image-net.org/challenges/LSVRC/2010/) . These benchmarks reveal the different performance among approximation methods, which would otherwise be difficult to observe in simple datasets. We conduct most experiments on the medium-scale PASCAL VOC data in order to compare against kernel methods. For this dataset, we use exclusively the `train` and `val` datasets, which have 964 images and around 2100 objects each. Classification results are also shown on the ImageNet dataset to demonstrate the efficiency of the kernel approximation. The experiments are conducted using an Intel Xeon E5520 2.27GHz with 8 cores and 24GB memory. The algorithm 1 is parallelized using OpenMP to take advantage of all cores.

### 6.1. Results on the Chebyshev Approximation

To test the Chebyshev approximation, we take a small-scale problem from the PASCAL VOC dataset. For training, we use segments that best match each ground truth segment in terms of overlap (called best-matching segments) in the `train` set, plus the ground truth segments. The best-matching segments in the `val` set are used as test. This creates a problem with 5100 training and 964 test segments.

The methods tested are `Chebyshev`, `PCA-Chebyshev` and `VZ`. The kernel approximation accuracies for each method are shown in the supplementary materials. For reference, we also report classification results on the $\chi^2$ kernel without exponentiating as `Chi2`, as well as the skewed $\chi^2$ kernel proposed in [18] as `Chi2-Skewed`. Because of the randomness in the Monte Carlo approximation, different random seeds lead to quite significant performance variations. Therefore the experiments are all averaged over 20 trials of random seeds. Within each trial, the same random seeds are used for all methods. For `PCA-Chebyshev`, the initial sampling is done using three times the final approximating dimensions, and PCA is performed to reduce the dimensionality to the same level as the other two methods. We test the classification performance of these kernels with two different types of features: a bag of SIFT words (BOW) feature of 300 dimensions, and a histogram of gradient (HOG) feature of 1700 dimensions. The classification is done via a linear SVM using the LIBSVM library (empirically we find the LIBLINEAR library produces worse results in this case for dense features). The $C$ parameter in LIBSVM is set to 50, the kernel to be approximated is a exp-$\chi^2$ kernel with $\beta = 1.5$. For `VZ`, the period parameter is set to the optimal one specified in [25]. For each kernel, $10$ dimensions are used to approximate the $\chi^2$ distance in each dimension. More dimensions have been tested but they did not improve the performance; therefore those results are not included.

The results are shown in Tables 1 and 2. It can be seen that the `Chebyshev` approximation almost always gives a slight performance edge over the `VZ` approximation. And `PCA-Chebyshev` is always significantly better than the other two. This should not be surprising since `PCA-Chebyshev` takes advantage of three times the dimensions than the other methods (before the dimensionality reduction). With $7000$ approximating dimensions and good random seeds, the `PCA-Chebyshev` method is able to match the performance of the kernel methods, a non-trivial achievement for the exp-$\chi^2$ kernel.

### 6.2. Results with Multiple Kernels on the PASCAL VOC Segmentation Challenge

In this section the image segmentation task from PASCAL VOC is considered, where we need to both recognize objects in images, and generate pixel-wise segmentations for these objects. Ground truth segments of objects paired with their category labels are available for training.

A recent state-of-the-art approach trains a scoring function for each class on many putative figure-ground segmentation hypotheses, obtained using the constrained parametric min-cut method [6]. This creates a large-scale learning task even if the original image database has moderate size: with $100$ segments in each image, training on $964$ images creates a learning problem with around $100,000$ training examples. This training set size is still tractable for kernel approaches, thus we can directly compare against them.

Two experiments are conducted using multiple kernel approximations of exp-$\chi^2$ kernels. The first one considers only SIFT on the foreground and background, in order to compare against a sparse coding method EMK [3] which works only on SIFT. The second one is on 7 different image descriptors, which include 3 HOGs at different scales, BOW on SIFT for the foreground and background, and BOW on color SIFT for the foreground and background [7]. The VOC segmentation measure is used to compare the approaches. This measure is the average of pixel-wise average precision on the 20 classes plus background. To avoid complications and for a fair comparison, the post-processing step [7] is not performed and the result is obtained by only outputing one segment with the highest s-core in each image. The method used for nonlinear estimation is one-against-all support vector regression (SVR) as in [17], and the method for linear estimation is one-against-all ridge regression. The latter is used since fast solutions for linear SVR problems are not yet available for out-of-core dense features. We want to avoid stochastic gradient methods (e.g., [19]) since these are difficult to tune to ful-l convergence, which can potentially bias the results. We average over 5 trials of different random seeds.

For the first experiment, we use 1,000 dimensions for each descriptor. EMK is performed with a 1000-words BOW without a spatial pyramid, and RF is performed by expanding a 300-words BOW to 1,000 RF dimensions. The result is shown in the upper part of Table 3, where one can see that `EMK` is vastly inferior to the RF approach in `Chebyshev-BOW-only`. It seems without the spatial pyramid, this feature encoding approach is not performing as well as approximations to the exp-$\chi^2$ kernel.

For the second experiment, the result of `Chebyshev`, `VZ` and `PCA-Chebyshev` is shown. Here `PCA-Chebyshev` takes the principal components on both the training and the test set. Additionally we show results taking PCA on the training set only, under

| Number of Dimensions | 3000 | 5000 | 7000 |
|---|---|---|---|
| Chi2 | 29.15% | 30.50% | 31.22% |
| Chi2-Skewed | 30.08% $\pm$ 0.74% | 30.37 % $\pm$ 0.63% | 30.51 % $\pm$ 0.35 % |
| Chebyshev | 31.26% $\pm$ 0.62% | 32.75% $\pm$ 0.71% | 33.03% $\pm$ 0.87% |
| PCA-Chebyshev | **32.74**%$\pm$ 0.62% | **33.35**%$\pm$ 0.68% | **33.49**%$\pm$ 0.45% |
| VZ | 31.37% $\pm$ 0.77% | 32.19 % $\pm$ 0.83% | 32.66% $\pm$ 0.78% |
| Exact exp-$\chi^2$ | 34.34% | | |

Table 1. Classification accuracy of exp-$\chi^2$ kernel when the $\chi^2$ function is approximated with different approximations, on a HOG descriptor. Results for the `Chi2` and `Chi2-Skewed` kernels are also shown for a reference.

| Number of Dimensions | 3000 | 5000 | 7000 |
|---|---|---|---|
| Chi2 | 41.91% | 42.32% | 42.12% |
| Chi2-Skewed | 39.82% $\pm$ 0.73% | 40.79% $\pm$ 0.55% | 40.90% $\pm$ 0.82% |
| Chebyshev | 41.48% $\pm$ 0.95% | 42.52% $\pm$ 0.88% | 42.65% $\pm$ 0.47% |
| PCA-Chebyshev | **42.80**%$\pm$ 0.74% | **43.25**%$\pm$ 0.55% | **43.42**%$\pm$ 0.42 % |
| VZ | 41.08% $\pm$ 1.22% | 42.06 % $\pm$ 0.92% | 42.46% $\pm$ 0.72 % |
| Exact exp-$\chi^2$ | 44.19% | | |

Table 2. Classification accuracy of exp-$\chi^2$ kernel when the $\chi^2$ function is approximated with different approximations, on a BOW-SIFT descriptor. Results for the `Chi2` and `Chi2-Skewed` kernels are also shown for a reference.

`PCA-training-Chebyshev`. For `Chebyshev` and `VZ`, we take 4,000 RF dimensions for each kernel, which totals 28,000 dimensions (the largest number that can fit in our computer memory). For PCA, we retain a total of 19,200 dimensions, particularly since additional dimensions do not seem to improve the performance. In addition, we compare to the `Nyström` method [28] by taking 28,000 random training examples and evaluating the combined kernel of each example against them for the feature vector.

The results in this experiment are computed using the pixel average precision measure of VOC, and are shown in the latter part of Table 3. The trend resembles the last experiment, as `PCA-Chebyshev` is better than `Chebyshev`, which is slightly better than `VZ`. Interestingly, `PCA-Chebyshev` is slightly better than `PCA-training-Chebyshev`, which shows the benefit of a semi-supervised approach to PCA. Interestingly, while being very different techniques to approximate the kernel, the performance of `Nyström` is comparable with `PCA-Chebyshev`. This may hint further improvements by combining the two techniques together. However, `PCA-Chebyshev` still displays a non-trivial performance gap with respect to Kernel SVR. This could partially be accounted to the difference between SVR and ridge regression, but still shows that the overall prediction model can be further improved.

### 6.3. Results on ImageNet

The ImageNet ILSVRC 2010 is a challenging classification dataset where 1 million images needs to be classified into 1,000 different categories. Here we only show preliminary experiments performed using the original BOW feature provided by the authors. Our goal is primarily to compare among different approximations, hence we did not generate multiple image descriptors or a spatial pyramid, which are compatible with our framework and will improve the results significantly (the running time of feature extraction is the main limiting factor). A calibration is done on the output scores to make the 500th highest score on each class the same.

| Method | Performance |
|---|---|
| EMK | $8.52\%$ |
| Chebyshev-BOW-only | $14.95\% \pm 0.67\%$ |
| Chebyshev | $26.25\% \pm 0.41\%$ |
| VZ | $25.57\% \pm 0.57\%$ |
| PCA-Chebyshev | $27.57\% \pm 0.44\%$ |
| PCA-training-Chebyshev | $26.95\% \pm 0.35\%$ |
| Nyström | $27.55\% \pm 0.49\%$ |
| Kernel SVR | $30.47\%$ |

Table 3. VOC Segmentation Performance on the `val` set, measured by pixel AP with one segment output per image (no post-processing). averaged over 5 random trials. The upper part shows results on only BOW-SIFT features for the foreground and background, in order to compare RF methods with the feature coding method EMK. The lower part shows results using 7 different descriptors.

In Table 4, the performance obtained using `Linear` kernel [9] is shown along with the RF results. It can be seen that among the tested RF methods, `PCA-Chebyshev` is still superior. Interestingly, different random seeds seem to have a much smaller effect on ImageNet, a fact for which we currently lack an explanation. In any case, one could see that RF improves the accuracy by at least 6% over the linear kernel, with very little computational overhead: for `VZ` and `Chebyshev`, each run would finish in 3

| Number of Dimensions | 3000 | 5000 | 7000 |
|---|---|---|---|
| Chebyshev | 16.30% ± 0.04% | 17.11% ± 0.04% | 17.63% ± 0.09% |
| PCA-Chebyshev | **16.66**%± 0.08% | **18.05**%± 0.08% | **18.85**%± 0.10 % |
| VZ | 16.05% ± 0.04% | 16.97 % ± 0.08% | 17.46% ± 0.09% |
| Linear | 11.6% ([9]) | | |

Table 4. Performance on ImageNet ILSVRC 2010 data

hours on a single machine. For the most-time consuming `PCA-Chebyshev`, each run still finishes in 7 hours. Especially, after collecting the Hessian matrix, training each regressor would only take 0.1-1 seconds, which would make this approach scale easily to 10,000 or more classes.

## 7. Conclusion

This paper introduces two techniques to improve the performance of random Fourier features in the context of approximating large-scale kernel machines. First, based on analogy to Chebyshev polynomials, an exact analytic series is proposed to the $\chi^2$ kernel. Second, out-of-core PCA on joint training and testing data is proposed and applied after extracting the random Fourier features. Empirical results show that these steps increase the performance of RF significantly for the state-of-the-art exponentiated $\chi^2$ kernel. In the meanwhile, the method is still linear in the number of training examples. Moreover, in combination with an $L_2$ loss function in the training objective and a ridge regression model, the methods are shown to scale extremely well with large number of classes.

## References


[1] F. Bach and M. I. Jordan. Predictive low-rank decomposition for kernel methods. In *ICML*, 2005. 2

[2] P. L. Bartlett, O. Bousquet, and S. Mendelson. Local rademacher complexities. *Annals of Statistics*, 33:1497–1537, 2005. 1

[3] L. Bo and C. Sminchisescu. Efficient match kernels between sets of features for visual recognition. 2009. 2, 6

[4] J. P. Boyd. *Chebyshev and Fourier Spectral Methods (second ed.)*. Dover, 2001. 3

[5] R. Caflisch. Monte carlo and quasi-monte carlo methods. *Acta Mumerica*, 7:1–49, 1998. 4

[6] J. Carreira and C. Sminchisescu. Constrained parametric min cuts for automatic object segmentation. In *CVPR*, 2010. 6

[7] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. *PAMI*, To appear. 1, 6

[8] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011. 2

[9] J. Deng, A. C. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *ECCV*, 2010. 7, 8

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2, 5

[11] P. Drineas and M. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *JMLR*, 6:2153–2175, 2005. 2

[12] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88:303–338, 2010. 5

[13] S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representation. *JMLR*, 2:243–264, 2001. 2

[14] P. V. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *ICCV*, 2009. 4

[15] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *NIPS*, pages 801–808, 2007. 2

[16] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, pages 609–616, 2009. 2

[17] F. Li, J. Carreira, and C. Sminchisescu. Object recognition as ranking holistic figure-ground hypotheses. In *CVPR*, 2010. 6

[18] F. Li, C. Ionescu, and C. Sminchisescu. Random Fourier approximations for skewed multiplicative histogram kernels. In *DAGM*, 2010. 1, 6

[19] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. S. Huang. Large-scale image classification: Fast feature extraction and svm training. In *CVPR*, 2011. 2, 6

[20] M. Pedersoli, A. Vedaldi, and J. Gonzalez. A coarse-to-fine approach for fast deformable object detection. In *CVPR*, 2011. 1

[21] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010. 2

[22] Y. Qu, G. Ostrouchov, N. Samatova, and A. Geist. Principal component analysis for dimension reduction in massive distributed data sets. In *ICDM*, 2002. 4

[23] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, 2007. 1, 2, 3

[24] V. Sreekanth, A. Vedaldi, C. V. Jawahar, and A. Zisserman. Generalized rbf feature maps for efficient detection. In *BMVC*, 2010. 1, 3

[25] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010. 1, 2, 6

[26] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *PAMI*, To appear. 1

[27] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 2

[28] C. K. I. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *NIPS*, 2001. 2, 7