Chebyshev Approximations to the Histogram χ^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

April 17, 2012

Abstract

The random Fourier embedding methodology can be used to approximate the performance of non-linear 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 models, especially for the exponential χ^2 kernel, one of the most powerful models for histograms. Based on analogies with Chebyshev polynomials, we propose an asymptotically convergent analytic series that can be used in the random Fourier approximation of the exponential χ^2 kernel. The new series removes the need to use periodic approximations to the χ^2 function, as typical in previous methods, and improves the classification accuracy. Besides, out-of-core principal component analysis (PCA) methods are introduced to reduce the dimensionality of the approximation and achieve better performance at the expense of only an additional constant factor to the time complexity. Moreoever, when PCA is performed jointly on the training and unlabeled testing data, a further performance improvement can be obtained. The proposed approaches are tested on the PASCAL VOC 2010 segmentation and the ImageNet ILSVRC 2010 datasets, and give statistically significant improvements over alternative approximation methods.

1 Introduction

Random Fourier (RF) feature embeddings [25, 27, 26, 19] are a promising methodology for large-scale classification. By using Monte Carlo sampling in the frequency domain of a kernel, one can construct an embedding such that, linear functions on that embedding are asymptotically convergent approximations to the nonlinear kernel functions. The benefit of this transformation is that the time complexity of many learning methods becomes linear in the number of examples n, compared to at least $O(n^{2.3})$ for

the (non-approximated) kernel method. Therefore, RF makes possible to use complicated nonlinear learning models in the massive datasets that are increasingly common nowadays. RF also benefits from most of the learning rate and generalization results valid for kernel methods, for instance, local Rademacher bounds [2]. Sucha advantages raise the question whether a 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 methods. Although this gap (0.5% – 4%) is not huge [19], it is still too significant to ignore. Multi-stage methods still play a major role for object detection [22], where RF and more expensive kernel methods can be used as two consecutive stages [8].

This paper aims to reduce the approximation gap without losing the advantageous $O(n)$ time complexity. Our two main contributions are: (a) a new convergent analytic series for the χ^2 distance commonly used for histogram features, and (b) principal component analysis (PCA) methodologies on the random feature representations, 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-χ^2)

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

proposed in [26]. Empirically we found that this similarity measure has the best performance in visual recognition over most RF kernel approximations that have been proposed so far. The two-stage method [26] first uses the Fourier transform on the non-negative orthant to approximate the χ^2 distance as an inner product. Then another standard RF approximation for the Gaussian kernel is used to estimate exp-χ^2.

Existing inner-product approximations to the χ^2 distance [28] rely 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 [28]. In this paper we derive an analytic recurrence formula to obtain asymptotically convergent approximations to the χ^2 distance. Besides its intrinsic theoretical and methodological novelty, experiments show that such an approach tends to obtain slightly better performance than existing periodic methods.

In addition, in order to obtain more compact embeddings for large-scale learning when the data does not fit into memory, we exploit an out-of-core versions of PCA that add little computational overhead to the RF approximation, especially when combined with least squares and other methods based on quadratic losses (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 the use of unlabeled (test) data in order to better estimate the covariance matrix in PCA. This, turns out, better selects of the frequency components that are effective for classification performance.

2 Related Work

Speed-ups to kernel methods based on low-rank approximations of the kernel matrix have been proposed before [14, 1]. These methods are effective, but applying the kernel predictor on new data requires slow kernel computations between the test and training examples. An alternative is to use the Nyström methods [30] that sub-sample the training set to operate on a reduced kernel matrix. Although this works well in practice, the asymptotic convergence rate of this approximation is slow: $O(n^{-\frac{1}{4}})$ [12], where n is the number of sample datapoints in the approximation.

A topic of recent interest is on methods for coding image features. The goal of such methods is to achieve good performance with linear classification or regression following a feature embedding [29, 16]. Hierarchical coding schemes with deeper structures have also been proposed [17]. Both sparse and dense coding schemes have proven successful, with supervector coding [21] and the Fisher kernels [23] being some of the best performers in the ImageNet large-scale image classification challenge [11]. Contrasting coding methods and RF, we notice that often RF work on bag-of-word vector quantizations whereas coding schemes often operate on raw image features, therefore having an extra layer of processing freedom. Nevertheless, replacing hard clustering with a soft-assignment may fill the gap between the performance of the histogram methods and some of the coding schemes [9]. Alternatively, one can use a Gaussian match kernel approximated with RF, instead of comparing bins independently [3].

The dictionaries of some of the influential coding schemes are usually extremely large (e.g., both the Fisher kernel and supervector coding usually require more than 200k dimensions [9]) 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 [25], and requires neither too many dimensions, in practice, nor training dictionaries. Therefore it appears 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.

In [28], the class of γ-*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 γ-*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

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

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

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

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

In [28], the function $e^{-j\omega\log x}\mathrm{sech}(\pi\omega)$ 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 [28]). However, $e^{-j\omega\log x}\mathrm{sech}(\pi\omega)$ 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 [28] because the introduced aperiodic bias is cancelled with the factor $\sqrt{xy}$ when x or y are small. However, uneven biases in different regions may impact learning performance. 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 a technical report [20].

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

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

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

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

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 $a_x^T a_y = \frac{1}{\pi}\int_0^\pi f_x(z)f_y(z)dz$.

Now, to determine the coefficients which are

$$\begin{aligned} a_k(x) &= \frac{2}{\pi}\int_0^\pi \cos(\frac{1}{\pi}\log\tan(\frac{z}{2})\log x)\cos(kz)dz \\ b_k(x) &= \frac{2}{\pi}\int_0^\pi \sin(\frac{1}{\pi}\log\tan(\frac{z}{2})\log x)\cos(kz)dz \end{aligned} \tag{11}$$

we try to derive an analytical recurrence relation. The idea is to use integration by parts twice. First we list some useful properties.

Lemma 1. *Let* $u(z) = \frac{1}{\pi}\log\tan(\frac{z}{2})\log x$*, then,*

(1) $u(\pi) = 0; \cos(u(\frac{\pi}{2})) = 1; \sin(u(\frac{\pi}{2})) = 0$

(2) $u'(z) = \frac{\log x}{\pi}\frac{1}{\sin(z)}, \frac{1}{u'(z)} = \frac{\pi}{\log x}\sin(z)$

(3) $\frac{1}{u'(0)} = 0, \frac{1}{u'(\pi)} = 0$

First of all we concern a_0 and b_0. With Mathematica we can compute $a_0 = 2\text{sech}(\frac{\log x}{2}) = \frac{4\sqrt{x}}{x+1}$ and $b_0 = 0$. For the rest of the series, we can immediately observe that for k even, $b_k = 0$ because $\sin(u(z))$ is antisymmetric at $\frac{\pi}{2}$ and $\cos(kz)$ is symmetric for even k and antisymmetric for odd k. Same argument gets us for k odd, $a_k = 0$. Therefore we only need to solve the coefficients b_k with odd k, and a_k with

even k. Therefore, we start with the integration:

$$\begin{aligned}
\frac{\pi}{4}b_k(x) &= \int_0^{\frac{\pi}{2}} \sin(\frac{1}{\pi}\log\tan(\frac{z}{2})\log x)\cos(kz)dz \\
&= \int_0^{\frac{\pi}{2}} \sin(u(z))\cos(kz)dz \\
&= -\int_0^{\frac{\pi}{2}} \cos(kz)\frac{1}{u'(z)}d(\cos(u(z))) \\
&= \int_0^{\frac{\pi}{2}} \cos(u(z))d(\cos(kz)\frac{\pi}{\log x}\sin(z)) \\
&= \frac{\pi}{\log x}\int_0^{\frac{\pi}{2}} \cos(u(z))(-k\sin(kz)\sin(z)+\cos(kz)\cos(z))dz \\
&= \frac{\pi}{\log x}\int_0^{\frac{\pi}{2}} \cos(u(z))(\cos((k+1)z)-(k-1)\sin(kz)\sin(z))dz \\
&= \frac{\pi}{\log x}\int_0^{\frac{\pi}{2}} \cos(u(z))(\cos((k+1)z)-\frac{k-1}{2}(\cos((k-1)z)-\cos((k+1)z)))dz \\
&= \frac{\pi}{4}\frac{\pi}{\log x}(\frac{k+1}{2}a_{k+1}(x)-\frac{k-1}{2}a_{k-1}(x))
\end{aligned}$$

Same trick applies to the a_k series with even coefficients:

$$\begin{aligned}
\frac{\pi}{4}a_k(x) &= \int_0^{\frac{\pi}{2}} \cos(\frac{1}{\pi}\log\tan(\frac{z}{2})\log x)\cos(kz)dz \\
&= \int_0^{\frac{\pi}{2}} \cos(u(z))\cos(kz)dz \\
&= \int_0^{\frac{\pi}{2}} \cos(kz)\frac{1}{u'(z)}d(\sin(u(z))) \\
&= -\int_0^{\frac{\pi}{2}} \sin(u(z))d(\cos(kz)\frac{\pi}{\log x}\sin(z)) \\
&= -\frac{\pi}{\log x}\int_0^{\frac{\pi}{2}} \sin(u(z))(-k\sin(kz)\sin(z)+\cos(kz)\cos(z))dz \\
&= -\frac{\pi}{\log x}\int_0^{\frac{\pi}{2}} \sin(u(z))(\cos((k+1)z)-(k-1)\sin(kz)\sin(z))dz \\
&= -\frac{\pi}{\log x}\int_0^{\frac{\pi}{2}} \sin(u(z))(\cos((k+1)z)-\frac{k-1}{2}(\cos((k-1)z)-\cos((k+1)z)))dz \\
&= -\frac{\pi}{4}\frac{\pi}{\log x}(\frac{k+1}{2}b_{k+1}(x)-\frac{k-1}{2}b_{k-1}(x))
\end{aligned}$$

For $k = 0$ it's slightly different as:

$$\begin{aligned}
\frac{\pi}{4} a_0(x) &= \int_0^{\frac{\pi}{2}} \cos(u(z)) dz \\
&= \int_0^{\frac{\pi}{2}} \frac{1}{u'(z)} d(\sin(u(z))) \\
&= -\int_0^{\frac{\pi}{2}} \sin(u(z)) d(\frac{\pi}{\log x} \sin(z)) \\
&= -\frac{\pi}{\log x} \int_0^{\frac{\pi}{2}} \sin(u(z)) \cos(z) dz \\
&= -\frac{\pi}{4} \frac{\pi}{\log x} b_1(x)
\end{aligned}$$

Now we can combine the nonzero entries for the two series and write it as c_k, and the recurrence relation can also be written out for c_k as:

$$c_k(x) = \begin{cases} \frac{1}{k}((-1)^k \frac{2 \log x}{\pi} c_{k-1}(x) + (k-2) c_{k-2}(x)), & k > 1 \\ -\frac{\sqrt{2} \log x}{\pi} c_0(x), & k = 1 \\ \frac{2x}{x+1}, & k = 0 \end{cases} \tag{12}$$

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 χ^2 kernel. Then, we follow [26] and use RF to approximate a Gaussian kernel on $c(x)$, to obtain the approximation of the exp-χ^2 kernel $k(x, y) = \exp(-\gamma \chi^2(x, y))$. The complete procedure is presented in Algorithm 1. We refer to the above algorithm as

Algorithm 1 Approximation of the exp-χ^2 kernel based on the Chebyshev approximation of the χ^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-χ^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 Ω, 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 [25].

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, making possible to apply 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 integration by parts used to derive the analytical approximation may not extend straightforward (this is a topic of our current research).

4 Convergence Rate of the Chebyshev Approximation

In this section we present an analysis on the asymptotic convergence rate of the Chebyshev approximation. Since (12) is exact, we can apply standard results on Fourier series coefficients [4], which state the convergence rate depends on the smoothness of the function that is approximated.

Lemma 2. $|k_0(x_i, y_i) - \sum_{k=1}^{m} c_k(x_i)c_k(y_i)| \leq \frac{C}{m}\sqrt{x_i y_i}$ *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 (oscillate at $z = 0$), we have:

$$0 < m c_m(x_i) \leq \sqrt{C}\sqrt{x_i} \tag{13}$$

and consequently

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

Using Lemma 2 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. We use Cauchy-Schwarz inequality,
$|k_0(x, y) - \sum_i \sum_{k=1}^{m} c_k(x_i)c_k(y_i)| \leq \frac{C}{m}\sum_i \sqrt{x_i y_i}$
$\leq \frac{C}{m}\sqrt{\sum_i x_i \sum_i y_i} = \frac{C}{m}$. $\square$

Although our method converges slower than the VZ's approximation, our convergence is independent on the factors $\sqrt{x_i y_i}$. When x_i or y_i are 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. In contrast, we can guarantee $\frac{k_0(x_i,y_i)}{\sqrt{xy}} \leq \frac{C}{m}$, which in turn would be superior to VZ. Since the image histograms considered in this work often consist of many small values instead of a few large ones, our approximation can be expected to work slightly better.

We can numerically simulate the constant C for different x values by computing the empirical bound $\max_m \frac{m c_m}{\sqrt{x}}$. 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.

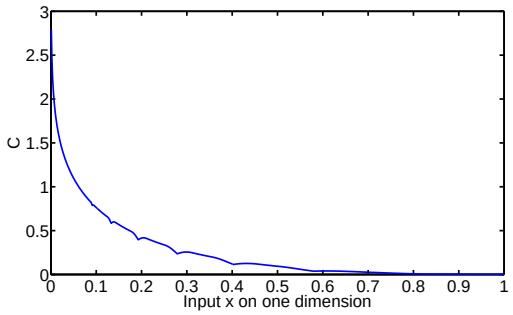

Figure 1: A plot of the C in Theorem 1 for different input values. The L_1 error of the kernel approximation is decided by C (Theorem 1). The value C is large when the histogram value is small, which can be offset by the $\sqrt{x}$ factor multiplying it.

5 Principal Component Analysis of Random Features on Multiple Descriptors

Another orthogonal strategy we pursue is principal component analysis on random features. This is useful for reducing the memory footprint when multiple image descriptors are used and RF embeddings are computed for each of them (this is common in computer vision, see e.g. [15]). 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. for 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 loaded into memory. PCA appears to be a natural choice in order to obtain fewer dimensions without significant loss of approximation quality. In fact, it is one of the very few possible choices in high dimensions, since many other techniques like quasi-Monte Carlo face the curse of dimensionality – as convergence rate decreases exponentially with the number of dimensions [5], they would be unsuitable for RF, when many dimensions are needed.

Another interesting aspect of RF-PCA is that it can bring a 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 domain. By considering both the training and the testing data during PCA, frequencies that help discriminate test data will more likely be selected. In the experiments this strategy is shown to improve performance over working with PCA only on training data.

The main problem in large-scale datasets is that data cannot be fully loaded into memory. Therefore PCA needs to be performed out-of-core – a high-performance computing terminology describing this situation (unable to load data into memory). As discussed extensively in the high-performance computing literature (e.g., [24]), the approach to out-of-core PCA in linear time would not be by singular value decomposition on the RF features Z, but by performing eigenvalue decomposition for the centered covariance matrix $Z^T(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)Z$, which can be computed out-of-core by

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.

1: Divide the data into k chunks, called $X_{(1)}, X_{(2)}, \ldots, X_{(k)}$.
2: $H = \mathbf{0}, m = \mathbf{0}, v = \mathbf{0}$
3: **for** $i = 1 \rightarrow k$ **do**
4: Load the i-th chunk $X_{(i)}$ into memory.
5: Use Algorithm 1 to compute the RF feature $Z_{(i)}$ for $X_{(i)}$.
6: $H = H + Z_{(i)}^T Z_{(i)}$, $m = m + Z_{(i)}^T 1$, $v = v + Z_{(i)}^T y$
7: **end for**
8: $H = H - \frac{1}{n} m m^T$.
9: 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.

just loading a chunk of X_i into memory at a time, computing their RF feature Z, computing the covariance matrix and then deleting 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

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

is the feature vector obtained after PCA projection.

It is very convenient to perform regression with a quadratic loss following 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 necessary for regression:

$$\begin{aligned} \tilde{Z}^T \tilde{Z} &= \bar{U}^T Z^T (I - \frac{1}{n}\mathbf{1}\mathbf{1}^T) Z \bar{U} \\ \tilde{Z}^T y &= \bar{U}^T Z^T (I - \frac{1}{n}\mathbf{1}\mathbf{1}^T) y \end{aligned} \tag{15}$$

It follows that only $Z^T Z$, $Z^T \mathbf{1}$ and $Z^T y$ 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 work 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 the disk.

The more sophisticated case is when PCA has 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 twice (Algorithm 4), first

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.

to perform the PCA, and then use U to transform X in chunks in order to obtain Z and $Z^T Z$. 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, $Z^T y$ will be computed in $O(nDc)$ time along with the Hessian. After the inverse of Hessian is obtained, only a matrix-vector multiplication costing $O(D^2 c)$ is needed to obtain all the solutions, without any dependency on n. Thus the total time of this approach with c classes is $O(nDc + D^2 c)$ which scales very well in 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 [13] and the ImageNet [11] 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 consider a small-scale problem from the PASCAL VOC segmentation dataset. For training, we use image 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` [28]. The kernel approximation accuracies for each method are shown in our accompanying TR [**?**]. For reference, we also report classification results on the χ^2 kernel without exponentiating as `Chi2`, as well as the skewed χ^2 kernel proposed in [19] as `Chi2-Skewed`. Due to the Monte Carlo approximation, different random seeds can 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-χ^2 kernel with $\beta = 1.5$. For `VZ`, the period parameter is set to the optimal one specified in [28]. For each kernel, 10 dimensions are used to approximate the χ^2 distance in each dimension. More dimensions have been tested but they did not improve performance (hence those results are not included).

The results are shown in Tables 1 and 2. It can be seen that the `Chebyshev` approximation almost always has 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 moe dimensions than the other methods (before dimensionality reduction). With 7000 approximating dimensions and good random seeds, the `PCA-Chebyshev` method is able to match the performance of the kernel methods, arguably a non-trivial achievement for the exp-χ^2.

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	$\mathbf{32.74}\%\pm$ 0.62%	$\mathbf{33.35}\%\pm$ 0.68%	$\mathbf{33.49}\%\pm$ 0.45%
VZ	31.37% $\pm$ 0.77%	32.19 % $\pm$ 0.83%	32.66% $\pm$ 0.78%
Exact exp-χ^2	34.34%		

Table 1: Classification accuracy of exp-χ^2 kernel when the χ^2 function is estimated with different approximations, on a HOG descriptor. Results for the `Chi2` and `Chi2-Skewed` kernels are also shown for 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	$\mathbf{42.80}\%\pm$ 0.74%	$\mathbf{43.25}\%\pm$ 0.55%	$\mathbf{43.42}\%\pm$ 0.42 %
VZ	41.08% $\pm$ 1.22%	42.06 % $\pm$ 0.92%	42.46% $\pm$ 0.72 %
Exact exp-χ^2	44.19%		

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

6.2 Results for 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 [7]. 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 train-

ing set is still tractable for kernel approaches, thus we can directly compare against them.

Two experiments are conducted using multiple kernel approximations for the exp-χ^2 kernels. We use 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 [18, 6]. The VOC segmentation measure is used to compare the different 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 [6] is not performed and the result is obtained by only reporting one segment with the highest score in each image. The method used for nonlinear estimation is one-against-all support vector regression (SVR) as in [18], 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., [21]) since these are difficult to tune to full convergence, and this can potentially bias the results. We average over 5 trials of different random seeds.

The result of applying `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 by PCA on the training set only, under `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 [30] by taking 28,000 random training examples and evaluating the combined kernel of each example against them on the feature vector.

The results for 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 one in the previous experiment, with `PCA-Chebyshev`'s accuracy better than `Chebyshev`, in turn, slightly higher than `VZ`. Interestingly, `PCA-Chebyshev` gives slightly better results than `PCA-training-Chebyshev`, which shows the benefit of a semi-supervised approach to PCA. While a very different techniques to approximate the kernel, the performance of `Nyström` is comparable with `PCA-Chebyshev`. This may indicate that further improvements can be achieved by combining ideas from the two techniques. However, `PCA-Chebyshev` still leaves a performance gap with respect to the full Kernel SVR. This could partly be accounted to the difference between SVR and ridge regression, but shows that the prediction model can be further improved.

6.3 Results on ImageNet

The ImageNet ILSVRC 2010 is a challenging classification dataset where 1 million images have 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 should improve the results significantly (the running time of fea-

Method	Performance
Chebyshev	$26.25\% \pm 0.41\%$
VZ	$25.50\% \pm 0.54\%$
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.

ture extraction is the main limiting factor). A calibration is done on the output scores to make the 500th highest score for each class the same.

Number of Dimensions	3000	5000	7000
Chebyshev	16.30% $\pm$ 0.04%	17.11% $\pm$ 0.04%	17.63% $\pm$ 0.09%
PCA-Chebyshev	$\mathbf{16.66}\%\pm$ 0.08%	$\mathbf{18.05}\%\pm$ 0.08%	$\mathbf{18.85}\%\pm$ 0.10 %
VZ	16.05% $\pm$ 0.04%	16.97 % $\pm$ 0.08%	17.46% $\pm$ 0.09%
Linear	11.6% ([10])		

Table 4: Performance on ImageNet ILSVRC 2010 data

In Table 4, the performance obtained using `Linear` kernel [10] is shown along with the RF results. It can be seen that among the tested RF methods, `PCA-Chebyshev` performs the best. Interestingly, different random seeds seem to have a much smaller effect on ImageNet. One could also see that RF improves accuracy by at least 6% over the linear kernel, with very little computational overhead: for methods like `VZ` and `Chebyshev`, each run would finish in 3 hours on a single machine. For the most-time consuming `PCA-Chebyshev`, each run still finishes in 7 hours. 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 novel techniques to improve the performance of random Fourier methodology (RF) in the context of approximating large-scale kernel machines. First, based on analogy to Chebyshev polynomials, an exact analytic series is proposed to the χ^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 methods increase the performance of RF significantly for the exponentiated χ^2 kernel, a state of the art similarity measure in computer vision and machine learning,

while the method is still linear in the number of training examples. Moreover, in conjunction with an L_2 loss training objective and a ridge regression model, the methods are shown to scale extremely well for large number of classes.

Acknowledgements: This work was supported in part by the DFG SM 317/1-1 and the EC under MCEXT-025481.

References

[1] F. Bach and M. I. Jordan. Predictive low-rank decomposition for kernel methods. In *ICML*, 2005. 3

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

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

[4] J. P. Boyd. *Chebyshev and Fourier Spectral Methods (second ed.)*. Dover, 2001. 8

[5] R. Caflisch. Monte Carlo and quasi-Monte Carlo Methods. *Acta Mumerica*, 7:1–49, 1998. 9

[6] J. Carreira, F. Li, and C. Sminchisescu. Object recognition by sequential figure-ground ranking. *IJCV*, 98, 2012. 14

[7] J. Carreira and C. Sminchisescu. Constrained parametric min cuts for automatic object segmentation. In *CVPR*, 2010. 13

[8] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. *PAMI*, 34, 2012. 2

[9] 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. 3

[10] 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. 15

[11] 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. 3, 12

[12] P. Drineas and M. Mahoney. On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning. *JMLR*, 6:2153–2175, 2005. 3

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

[14] S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representation. *JMLR*, 2:243–264, 2001. 3

[15] P. V. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *ICCV*, 2009. 9

[16] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *NIPS*, pages 801–808, 2007. 3

[17] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009. 3

[18] F. Li, J. Carreira, and C. Sminchisescu. Object recognition as ranking holistic figure-ground hypotheses. In *CVPR*, 2010. 14

[19] F. Li, C. Ionescu, and C. Sminchisescu. Random Fourier approximations for skewed multiplicative histogram kernels. In *DAGM*, 2010. 1, 2, 12

[20] F. Li, G. Lebanon, and C. Sminchisescu. The Chebyshev Approximation for the Histogram χ^2 Kernel and Principal Components of Fourier Features. Technical report, Computational Science and Engineering, Georgia Institute of Technology and Institute for Numerical Simulation, University of Bonn, June 2012. 4

[21] 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. 3, 14

[22] M. Pedersoli, A. Vedaldi, and J. Gonzalez. A coarse-to-fine approach for fast deformable object detection. In *CVPR*, 2011. 2

[23] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010. 3

[24] Y. Qu, G. Ostrouchov, N. Samatova, and A. Geist. Principal component analysis for dimension reduction in massive distributed data sets. In *ICDM*, 2002. 9

[25] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, 2007. 1, 3, 7

[26] V. Sreekanth, A. Vedaldi, C. V. Jawahar, and A. Zisserman. Generalized rbf feature maps for efficient detection. In *BMVC*, 2010. 1, 2, 7

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

[28] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *PAMI*, 34:480–492, 2012. 2, 3, 4, 12

[29] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 3

[30] C. K. I. Williams and M. Seeger. Using the Nyström Method to Speed Up Kernel Machines. In *NIPS*, 2001. 3, 14