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

Abstract

The random Fourier embedding methodology can be used to approximate the performance of non-linear kernel classifiers in linear time on 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 of the χ^2 measure. 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 when used in the random Fourier approximation of the exponential χ^2 kernel. 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. Moreover, when PCA is performed jointly on the training and unlabeled testing data, further performance improvements can be obtained. The proposed approaches are tested on the PASCAL VOC 2010 segmentation and the ImageNet ILSVRC 2010 datasets, and shown to give statistically significant improvements over alternative approximation methods.

1. Introduction

Random Fourier (RF) feature embeddings [19, 25, 26, 28] 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. 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 exact 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 learning rate and generalization results valid for kernel methods such as local Rademacher bounds [2]. Such advantages raise the question whether the exact kernel calculation 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 [6].

In this paper we aim to reduce the approximation gap without losing the advantageous $O(n)$ time complexity of learning. Our two main contributions are: (a) a new convergent analytic series for the χ^2 distance, used for histogram features, and (b) principal component analysis (PCA) methodologies adapted for the random feature representations that can improve their performance without significant 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 approximations for different kernels. 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. In a second step, a standard RF approximation for the Gaussian kernel is used to estimate exp-χ^2.

Existing inner-product approximations to the χ^2 distance [27, 28] rely on a periodic version of the function. The approximation quality can fluctuate 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 theoretical novelty, in experiments we 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 out-of-core versions of PCA that add little

computational overhead to the RF approximation, especially when combined with least squares and other 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 to improve the performance by better selecting effective frequency components.

2. Related Work

Speed-ups to kernel methods based on low-rank approximations of the kernel matrix have been proposed before [1, 14]. These methods are effective, but applying the kernel predictor on new data requires slow kernel computations between the test datapoint and the training examples. An alternative is to use Nyström methods [30] to sub-sample the training set and operate on a reduced kernel matrix. Although this approach works well in practice, the asymptotic convergence rate of the approximation is slower than RF methods: $O(n^{-\frac{1}{4}})$ vs. $O(n^{-\frac{1}{2}})$ [12], where n is the number of sampled datapoints.

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

The dictionaries of some influential coding schemes are usually extremely large – both the Fisher kernel and supervector coding usually require more than 200k dimensions [9]) and the training of the dictionary is often time-consuming. RF is theoretically guaranteed to approximate the kernel model with a reasonable asymptotic convergence rate [25], and in practice requires neither dictionary training nor too many dimensions. Therefore it appears worth exploring as an alternative approach.

3. The Chebyshev Approximation

Throughout this paper we use $\mathbf{X}$ 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

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

(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 term $e^{-j\omega \log x} \mathrm{sech}(\pi\omega)$ is approximated with a periodic function, which is then approximated with finite Fourier coefficients (hereafter called the VZ approximation, 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 high when $|\log x|$ is larger than the period. Convergence is attained in [28] because the introduced aperiodic bias is diminished with the factor $\sqrt{xy}$ when x or y is small. However, uneven biases in different regions may impact classification 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 – more detail is given in our companion 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

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

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

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

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

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

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

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 two functions are symmetric/antisymmetric, half of the coefficients in either cosine series are zero. Therefore, we can combine the two series to form 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 algebraic derivations (see our TR [20]) we obtain the following analytic form for $c_n(x)$:

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

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 χ^2 kernel. Then, we follow [26] and use RF to estimate a Gaussian kernel on $c(x)$, and obtain the approximation for the exp-χ^2 kernel $k(\mathbf{x},\mathbf{y}) = \exp(-\gamma\chi^2(\mathbf{x},\mathbf{y}))$. The complete procedure is presented in Algorithm 1. We refer to 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, 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 directly (a topic of our current research).

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 $\mathbf{X} = [\mathbf{X}_1^\top, \mathbf{X}_2^\top, \ldots, \mathbf{X}_n^\top]^\top$. Parameters m, D.

output : The random Fourier feature $\mathbf{Z}$ 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 $\mathbf{X}_i$. Denote $\mathbf{c}(\mathbf{X}_i)$ the $md\times 1$ vector constructed by concatenating all $c_k(x_{ij}), j = 1, \ldots, d$.

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

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

4: $\mathbf{Z}_i = \cos(\mathbf{c}(\mathbf{X}_i)\mathbf{\Omega} + \mathbf{b})$ is the RF feature for $\mathbf{X}_i$ [25].

4. Convergence Rate of the Chebyshev Approximation

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

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

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

and consequently

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

Using Lemma 1 it is easy to prove the following result:

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

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

Albeit slower than the VZ approximation, the convergence of our method is independent of the factors $\sqrt{x_i y_i}$. When x_i or y_i are small, the VZ approximation can only guarantee $\frac{k_0(x_i,y_i)}{\sqrt{x_i y_i}} \leq C_1$ where C_1 is a constant close to 1. In contrast, we can guarantee $\frac{k_0(x_i,y_i)}{\sqrt{x_i y_i}} \leq \frac{C}{m}$, which could 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 offer an advantage for such data distributions.

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.

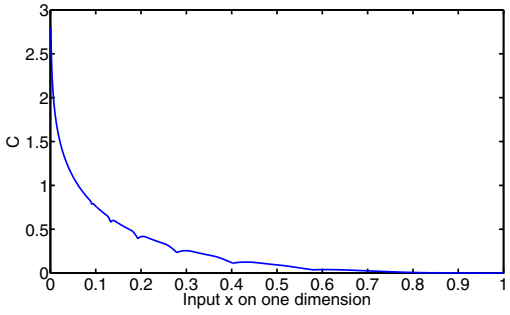

Figure 1. A plot of the C in Theorem 1 for different input values. The L_1 error of the kernel approximation is decided by C (Theorem 1). The value C is large when the histogram value is small, which can be offset by the $\sqrt{x}$ factor multiplying it.

5. Principal Component Analysis of Random Features on Multiple Descriptors

Another complementary strategy we pursue to improve efficiency 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 one 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 degrades 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 a flavor of semi-supervised learning, as 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 within the PCA calculation, frequencies that help discriminate test data will more likely be selected. In our experiments this strategy is shown to improve performance over PCA representations computed only on the training data.

Algorithm 2 Out-of-Core Principal Component Analysis.

input : $n \times d$ data matrix $\mathbf{X} = [\mathbf{X}_1^\top, \mathbf{X}_2^\top, \ldots, \mathbf{X}_n^\top]^\top$. Output vector $\mathbf{y}$. Number of dimension D to retain after PCA.

1: Divide the data into k chunks, called $\mathbf{X}_{(1)}$, $\mathbf{X}_{(2)}, \ldots,$ $\mathbf{X}_{(k)}$.
2: $\mathbf{H} = \mathbf{0}, \mathbf{m} = \mathbf{0}, \mathbf{v} = \mathbf{0}$
3: **for** $i = 1 \rightarrow k$ **do**
4: Load the i-th chunk $\mathbf{X}_{(i)}$ into memory.
5: Use Algorithm 1 to compute the RF feature $\mathbf{Z}_{(i)}$ for $\mathbf{X}_{(i)}$.
6: $\mathbf{H} = \mathbf{H} + \mathbf{Z}_{(i)}^\top \mathbf{Z}_{(i)}$, $\mathbf{m} = \mathbf{m} + \mathbf{Z}_{(i)}^\top 1$, $\mathbf{v} = \mathbf{v} + \mathbf{Z}_{(i)}^\top \mathbf{y}$
7: **end for**
8: $\mathbf{H} = \mathbf{H} - \frac{1}{n}\mathbf{m}\mathbf{m}^\top$.
9: Compute eigen-decomposition $\mathbf{H} = \mathbf{U}\mathbf{D}\mathbf{U}^\top$. Output the first D columns of $\mathbf{U}$ as $\bar{\mathbf{U}}$, the diagonal matrix $\mathbf{D}$, and the input-output product $\mathbf{v}$.

The main problem in large-scale datasets is that data cannot be completely loaded in memory. Therefore PCA needs to be performed *out-of-core* (we follow the high-performance computing terminology describing this memory constrained situation). As discussed extensively in the high-performance computing literature e.g. [24], but to our knowledge not well known in computer vision, the approach to out-of-core PCA in linear time would not be by singular value decomposition on the RF features $\mathbf{Z}$, but by performing eigenvalue decomposition on the centered covariance matrix $\mathbf{Z}^\top(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)\mathbf{Z}$, which can be computed out-of-core by just loading a chunk of $\mathbf{X}_i$ into memory at a time, computing their RF feature $\mathbf{Z}$, computing the covariance matrix and then removing the RF features from memory. Then, an eigen-decomposition gives the transformation matrix $\mathbf{U}$ for PCA. We denote $\bar{\mathbf{U}}$ the matrix obtained by selecting the first D dimensions of $\mathbf{U}$ corresponding to the largest eigenvalues (Algorithm 2). Denote the mean vector

of the input matrix $\bar{\mathbf{Z}} = \frac{1}{n}\mathbf{Z}^\top \mathbf{1}$, then

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

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 $\tilde{\mathbf{Z}}^\top\tilde{\mathbf{Z}}$ and $\tilde{\mathbf{Z}}^\top\mathbf{y}$ are necessary for regression:

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

Algorithm 3 Learning after PCA with Quadratic Loss.

input : $n \times d$ data matrix $\mathbf{X} = [\mathbf{X}_1^\top, \mathbf{X}_2^\top, \ldots, \mathbf{X}_n^\top]^\top$. Output vector $\mathbf{y}$. Number of dimension D to retain after PCA.

1: Perform out-of-core PCA using Algorithm 2.
2: $\mathbf{H}' = \bar{\mathbf{U}}^\top\mathbf{H}\bar{\mathbf{U}} = \bar{\mathbf{D}}$, the first D rows and columns of the diagonal matrix $\mathbf{D}$.
3: $\mathbf{v}' = \bar{\mathbf{U}}^\top\mathbf{v} - \frac{1}{n}(\mathbf{1}^\top\mathbf{y})\bar{\mathbf{U}}^\top\mathbf{m}$.
4: Perform learning on $\bar{\mathbf{D}}, \mathbf{v}'$, e.g., for linear ridge regression where the optimization is $\arg\min_{\mathbf{w}} \|\mathbf{w}^\top\tilde{\mathbf{Z}} - \mathbf{y}\|^2 + \lambda\|\mathbf{w}\|^2$, the solution is $\mathbf{w} = (\bar{\mathbf{D}} + \lambda\mathbf{I})^{-1}\mathbf{v}'$.
5: Use $\bar{\mathbf{U}}^\top\mathbf{w}$ instead of $\mathbf{w}$ as a function of the original inputs: $f(\mathbf{x}) = \mathbf{w}^\top\bar{\mathbf{U}}\mathbf{x} - \frac{1}{n}\mathbf{w}^\top\bar{\mathbf{U}}\mathbf{m}$, in order to avoid the projection for the testing examples.

It follows that only $\mathbf{Z}^\top\mathbf{Z}$, $\mathbf{Z}^\top\mathbf{1}$ and $\mathbf{Z}^\top\mathbf{y}$ 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 the matrix decomposition of $\mathbf{H}$. If ridge regression is used, the $\mathbf{H}'$ 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., $\mathbf{Z} = [\mathbf{Z}^{(1)}\mathbf{Z}^{(2)}\ldots\mathbf{Z}^{(l)}]$, where each $\mathbf{Z}^{(i)}$ is the RF feature embedding of each kernel. This time, the need to compute $\mathbf{Z}^{(i)^\top}\mathbf{Z}^{(j)}$ rules out tricks for simple computation. The data needs to be read twice (Algorithm 4), first to perform PCA, and then use $\mathbf{U}$ to transform $\mathbf{X}$ in chunks in order to obtain $\mathbf{Z}$ and $\mathbf{Z}^\top\mathbf{Z}$. The full computation is however still linear in the number of training examples.

In both cases, the projection is not required for the testing examples. Because whenever $\mathbf{w}$ is obtained, $\mathbf{w}^\top\tilde{\mathbf{Z}} = \mathbf{w}^\top\bar{\mathbf{U}}(\mathbf{Z} - \frac{1}{n}\bar{\mathbf{Z}}\mathbf{1}^\top)$, then $\bar{\mathbf{U}}\mathbf{w}$ 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 $\mathbf{X} = [\mathbf{X}_1^\top, \mathbf{X}_2^\top, \ldots, \mathbf{X}_n^\top]^\top$. Output vector $\mathbf{y}$. 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 $\mathbf{X}_{(i)}$ into memory.
4: Use Algorithm 1 to compute the RF feature $\mathbf{Z}_{(i)}$ for $\mathbf{X}_{(i)}$, with the same normal matrix $\mathbf{\Omega}$ as before.
5: $\tilde{\mathbf{Z}} = (\mathbf{Z}_{(i)} - \frac{1}{n}\mathbf{1}\mathbf{m}^\top)\bar{\mathbf{U}}$.
6: $\mathbf{H}' = \mathbf{H}' + \tilde{\mathbf{Z}}^\top\tilde{\mathbf{Z}}$, $\mathbf{v}' = \mathbf{v}' + \tilde{\mathbf{Z}}^\top\mathbf{y}$
7: **end for**
8: Perform learning on $\mathbf{H}', \mathbf{v}'$. E.g., for linear least squares where the optimization is $\arg\min_{\mathbf{w}} \|\mathbf{w}^\top\mathbf{Z} - \mathbf{y}\|^2$, the solution is $\mathbf{w} = \mathbf{H}'^{-1}\mathbf{v}'$.
9: Use $\bar{\mathbf{U}}^\top\mathbf{w}$ instead of $\mathbf{w}$ as a function of the original inputs: $f(\mathbf{x}) = \mathbf{w}^\top\bar{\mathbf{U}}\mathbf{x} - \frac{1}{n}\mathbf{w}^\top\bar{\mathbf{U}}\mathbf{m}$, in order to avoid the projection step for the testing examples.

We note that out-of-core least squares or ridge regression scale extremely well in the number of output dimensions c, and this property can be exploited to solve one-against-all classification problems with c classes. In Algorithm 2 or 4, $\mathbf{Z}^\top\mathbf{y}$ will be computed in $O(nDc)$ time along with the Hessian. After the Hessian inverse is obtained, only a matrix-vector multiplication of at most $O(D^2c)$ is needed to obtain all solutions without any dependency on n. The total time of this approach for c classes is $O(nDc + D^2c)$ 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 good baseline result quickly.

6. Experiments

Our experiments are conducted on two challenging datasets: PASCAL VOC 2010 [13] and ImageNet [11] ILSVRC 2010 (http://www.image-net.org/challenges/LSVRC/2010/). These benchmarks reveal the performance differences 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 exact 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 our kernel approximations. 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 (obtained using the constrained parametric min-cuts algorithm, CPMC [6]) 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 small-scale problem with 5100 training and 964 test segments.

The methods tested in experiments are `Chebyshev`, `PCA-Chebyshev` and `VZ` [28]. For reference, we also report classification results for 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 on 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 found the LIBLINEAR library produced less good results in this context, for dense features). The C parameter in LIBSVM is validated to 50, the kernel to be approximated is exp-χ^2, 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 significantly affect performance.

The results are shown in Tables 1 and 2. It can be seen that `Chebyshev` approximation slightly improves over the `VZ` approximation, and `PCA-Chebyshev` is always significantly better than the other two. This is not surprising as `PCA-Chebyshev` takes advantage of three times more 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 exact kernel methods, arguably a non-trivial achievement for the exp-χ^2.

6.2. Results for Multiple Kernels on the PASCAL VOC Segmentation Challenge

In this section, we consider the semantic segmentation task from PASCAL VOC, 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 CPMC [7, 8]. This creates a large-scale learning task even if the original image database has moderate size: with 100 segments in each image, training for 964 images creates a learning problem with around $100,000$ training examples. This training set is still tractable for exact kernel approaches and 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 distraction 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 avoided stochastic gradient methods (e.g., [21]) since these are difficult to tune to convergence, and such effects 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 in Table 3. In this case, the `PCA-Chebyshev` method computes 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 use 4,000 RF dimensions to approximate each kernel lifting, which totals 28,000 dimensions (the largest size we 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

Number of Dimensions	3000	5000	7000
Chi2	29.15%	30.50%	31.22%
Chi2-Skewed	30.08% ± 0.74%	30.37 % ± 0.63%	30.51 % ± 0.35 %
Chebyshev	31.68% ± 0.63%	32.75% ± 0.71%	32.69% ± 0.83%
PCA-Chebyshev	**32.80**%± 0.70%	**33.35**%± 0.68%	**33.49**%± 0.59%
VZ	31.40% ± 1.23%	32.39 % ± 0.89%	32.72% ± 0.79%
Exact exp-χ^2	34.34%		

Table 1. Classification accuracy for the 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% ± 0.73%	40.79% ± 0.55%	40.90% ± 0.82%
Chebyshev	41.48% ± 0.95%	42.52% ± 0.88%	42.89% ± 0.65%
PCA-Chebyshev	**42.80**%± 0.74%	**43.25**%± 0.55%	**43.63**%± 0.55 %
VZ	41.29% ± 0.77%	42.33 % ± 0.78%	42.86% ± 0.69 %
Exact exp-χ^2	44.19%		

Table 2. Classification accuracy for the 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.

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 technique 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, in this case even `PCA-Chebyshev` shows a performance decrease with respect to the exact kernel SVR. This could partly be accounted to the difference between the SVR and ridge regression losses, and shows that the prediction model can be further refined.

Method	Performance
Chebyshev	26.25% ± 0.41%
VZ	25.50% ± 0.54%
PCA-Chebyshev	27.57% ± 0.44%
PCA-training-Chebyshev	26.95% ± 0.35%
Nyström	27.55% ± 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), and averaged over 5 random trials. The upper part of the table shows results on only BOW-SIFT features extracted on foreground and background. The lower part shows results based on combining 7 different descriptors.

6.3. Results on ImageNet

The ImageNet ILSVRC 2010 is a challenging classification dataset where 1 million images have to be separated 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 Since regression is used, the resulting scores are not well-calibrated across categories. Therefore we perform a calibration of the output scores to make the 500th highest score of each class the same.

In Table 4, the performance obtained using the `Linear` kernel [10] is shown alongside RF results. It can be seen that among the tested RF methods, `PCA-Chebyshev` achieves the highest score. 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 finishes in 3 hours on a single machine. For the more time-consuming `PCA-Chebyshev`, each run still finishes in 7 hours. After collecting the Hessian matrix, training each regressor takes 0.1-1 seconds, making this approach scale easily to 10,000 or more classes.

7. Conclusions

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 for the χ^2 kernel. Second, out-of-core PCA on joint training and testing data is proposed and applied on the random Fourier representation. Empirical results show that these complementary methods increase the performance of RF significantly for the exponentiated χ^2 kernel – a state of the art similarity

Number of Dimensions	3000	5000	7000
Chebyshev	16.30% $\pm$ 0.04%	17.11% $\pm$ 0.04%	17.63% $\pm$ 0.09%
PCA-Chebyshev	**16.66**%$\pm$ 0.08%	**17.85**%$\pm$ 0.08%	**18.65**%$\pm$ 0.10 %
VZ	16.10% $\pm$ 0.04%	16.95 % $\pm$ 0.08%	17.48% $\pm$ 0.09%
Linear	11.6% ([10])		

Table 4. Performance of linear classifiers as well as non-linear approximation methods on ImageNet ILSVRC 2010 data. Notice the significant boost provided by the non-linear approximations (exact non-linear calculations are intractable in this large scale setting).

measure in computer vision and machine learning – while being 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 decision problems with a large number of classes.

Acknowledgements: This work was supported in part by the DFG SM 317/1-1, CNCS-UEFISCDI, PNII-RU-RC-2/2009, and the European Commission, under MCEXT-025481.

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. In *NIPS*, 2009. 2

[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, F. Li, and C. Sminchisescu. Object recognition by sequential figure-ground ranking. *IJCV*, 98, 2012. 1, 6

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

[8] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. *PAMI*, 34, 2012. 6

[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. 2

[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. 7, 8

[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. 2, 5

[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. 2

[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. 5

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

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

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

[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. 2

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

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

[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. 2, 3

[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. 2, 6

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

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

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

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

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

[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. 1, 2, 6

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

[30] C. K. I. Williams and M. Seeger. Using the Nyström Method to Speed Up Kernel Machines. In *NIPS*, 2001. 2, 6