Linearized GMM Kernels and Normalized Random Fourier Features

Ping Li
Department of Statistics and Biostatistics
Department of Computer Science
Rutgers University
Piscataway, NJ 08854, USA
pingli@stat.rutgers.edu

ABSTRACT

The method of "random Fourier features (RFF)" has become a popular tool for approximating the "radial basis function (RBF)" kernel. The variance of RFF is actually large. Interestingly, the variance can be substantially reduced by a simple normalization step as we theoretically demonstrate. We name the improved scheme as the "normalized RFF (NRFF)", and we provide a technical proof of the asymptotic variance of NRFF, as validated by simulations.

We also propose the "generalized min-max (GMM)" kernel as a measure of data similarity, where data vectors can have both positive and negative entries. GMM is positive definite as there is an associate hashing method named "generalized consistent weighted sampling (GCWS)" which linearizes this (nonlinear) kernel. We provide an extensive empirical evaluation of the RBF and GMM kernels on more than 50 datasets. For a majority of the datasets, the (tuning-free) GMM kernel outperforms the best-tuned RBF kernel.

We then conduct extensive classification experiments for comparing the linearized RBF kernel using NRFF with the linearized GMM kernel using GCWS. We observe that, in order to reach a similar accuracy, GCWS typically requires substantially fewer samples than NRFF, even on datasets where the original RBF kernel outperforms the original GMM kernel. As the training, storage, and processing costs are directly proportional to the sample size, our experiments can help demonstrate that GCWS would be a more practical scheme for large-scale machine learning applications.

The empirical success of GCWS (compared to NRFF) can also be explained theoretically, from at least two aspects. Firstly, the relative variance (normalized by the squared expectation) of GCWS is substantially smaller than that of NRFF, except for the very high similarity region (where the variances of both methods approach zero). Secondly, if we are allowed to make a general model assumption on the data, then we can show analytically that GCWS exhibits much smaller variance than NRFF for estimating the same object (e.g., the RBF kernel), except for the very high similarity region.

KDD '17, August 13-17, 2017, Halifax, NS, Canada

DOI: 10.1145/3097983.3098081

1 INTRODUCTION

It is popular in machine learning practice to use linear algorithms such as logistic regression or linear SVM. It is known that one can often improve the performance of linear methods by using nonlinear algorithms such as kernel SVMs, if the computational/storage burden can be resolved. In this paper, we introduce an effective measure of data similarity termed "generalized min-max (GMM)" kernel and the associated hashing method named "generalized consistent weighted sampling (GCWS)", which efficiently converts this nonlinear kernel into linear kernel. Moreover, we will also introduce what we call "normalized random Fourier features (NRFF)".

We start the introduction with the basic linear kernel. Consider two data vectors $u, v \in \mathbb{R}^D$. It is common to use the normalized linear kernel (i.e., the "cosine"):

$$\rho = \rho(u,v) = \frac{\sum_{i=1}^D u_i v_i}{\sqrt{\sum_{i=1}^D u_i^2}\sqrt{\sum_{i=1}^D v_i^2}} \tag{1}$$

This normalization step is in general a recommended practice. For example, when using LIBLINEAR or LIBSVM packages [8], it is suggested to first normalize the input data vectors to unit l_2 norm.

The popularity of the cosine can be explained by the fact that, if (u_i, v_i), $i = 1$ to D, are iid samples from a bivariate zero-mean normal distribution with a true (population) correlation parameter $\bar{\rho}$, then as $D \to \infty$, ρ converges to $\bar{\rho}$. However, when data are not exactly normal, as shown in [18], this convergence may not even hold. See Section 3 for more explanation on this important matter.

The proposed GMM kernel is defined on general data types which can have both negative and positive entries. The basic idea is to first transform the original data into nonnegative data and then compute the min-max kernel [11, 14, 19] on the transformed data.

1.1 Generalized Min-Max (GMM) Kernel

Consider the data vector u_i, $i = 1$ to D. We define the following transformation, depending on whether an entry u_i is positive or negative:

$$\begin{cases} \tilde{u}_{2i-1} = u_i, \quad \tilde{u}_{2i} = 0 & \text{if } u_i > 0 \\ \tilde{u}_{2i-1} = 0, \quad \tilde{u}_{2i} = -u_i & \text{if } u_i \le 0 \end{cases} \tag{2}$$

For example, when $D = 2$ and $u = [-5 \;\; 3]$, the transformed data vector becomes $\tilde{u} = [0 \;\; 5 \;\; 3 \;\; 0]$. After this step of data transformation, the generalized min-max (GMM) similarity is defined as

$$GMM(u,v) = \frac{\sum_{i=1}^{2D} \min(\tilde{u}_i, \; \tilde{v}_i)}{\sum_{i=1}^{2D} \max(\tilde{u}_i, \; \tilde{v}_i)} \tag{3}$$

When the data are binary (0/1), GMM becomes the well-known "resemblance" similarity [4, 5, 16]. We will show in Section 4 that GMM is indeed an effective measure of data similarity through an extensive experimental study on kernel SVM classification. It is generally nontrivial to scale nonlinear kernels for large data [3]. In a sense, it is not practically meaningful to discuss nonlinear kernels without knowing how to compute them efficiently. This paper will focus on the "generalized consistent weighted sampling" (GCWS).

1.2 Generalized Consistent Weighted Sampling

Algorithm 1 summarizes the generalized consistent weighted sampling (GCWS). Given data vectors u and v, we transform them into nonnegative vectors $\tilde{u}$ and $\tilde{v}$ as in Eq. (2), then apply the original consistent weighted sampling (CWS) [11, 19] to generate tuples:

$$\left(i^*_{\tilde{u},j}, t^*_{\tilde{u},j}\right) \text{ and } \left(i^*_{\tilde{v},j}, t^*_{\tilde{v},j}\right), \quad j = 1, 2, ..., k \tag{4}$$

where $i^* \in [1,\ 2D]$ and t^* is unbounded. Following [11, 19], we have the basic probability result in Theorem 1.1.

Theorem 1.1.

$$\mathbf{Pr}\left\{\left(i^*_{\tilde{u},j}, t^*_{\tilde{u},j}\right) = \left(i^*_{\tilde{v},j}, t^*_{\tilde{v},j}\right)\right\} = GMM(u, v) \tag{5}$$

Algorithm 1 Generalized Consistent Weighted Sampling (GCWS).

Input: Data vector $u = (i = 1 \text{ to } D)$
Transform: Generate vector $\tilde{u}$ in $2D$-dim by (2)
Output: Consistent uniform sample (i^*, t^*)

For i from 1 to $2D$
$r_i \sim Gamma(2,1),\ c_i \sim Gamma(2,1),\ \beta_i \sim Uniform(0,1)$
$t_i \leftarrow \lfloor \frac{\log \tilde{u}_i}{r_i} + \beta_i \rfloor,\quad a_i \leftarrow \log(c_i) - r_i(t_i + 1 - \beta_i)$
End For
$i^* \leftarrow arg\min_i\ a_i, \qquad t^* \leftarrow t_{i^*}$

Note that for sparse data, one will only need to deal with nonzero entries. The same random variables $\{r_i, c_i, \beta_i\}$ will be used for all data vectors (not just u). Algorithm 1 describes the procedure for generating one sample (i^*, t^*). With k independent samples, one can simply use the averaged indicator to estimate $GMM(u, v)$. The expectation (E) and variance (Var) of the indicator are

$$E\left[1\{i^*_{\tilde{u},j} = i^*_{\tilde{v},j} \text{ and } t^*_{\tilde{u},j} = t^*_{\tilde{v},j}\}\right] = GMM(u, v), \tag{6}$$

$$Var\left[1\{i^*_{\tilde{u},j} = i^*_{\tilde{v},j} \text{ and } t^*_{\tilde{u},j} = t^*_{\tilde{v},j}\}\right] = (1 - GMM(u, v))GMM(u, v) \tag{7}$$

The estimation variance, given k samples, will be $\frac{1}{k}(1-GMM)GMM$.

Practical Implementation: Based on intensive empirical observations [14], one can safely ignore t^* (which is unbounded) and simply use $\mathbf{Pr}\left\{i^*_{\tilde{u},j} = i^*_{\tilde{v},j}\right\} \approx GMM(u, v)$. Specifically, for each data vector u, we obtain k random samples $i^*_{\tilde{u},j}$, $j = 1$ to k. Then we store only the lowest b bits of i^*, based on the idea of [17]. We need to view those k integers as locations (of the nonzeros) instead of numerical values. For example, when $b = 2$, we should view i^* as a vector of length $2^b = 4$. If $i^* = 3$, then we code it as [1 0 0 0]; if $i^* = 0$, we code it as [0 0 0 1]. We then concatenate all k such vectors into a binary vector of length $2^b \times k$, with exactly k 1's.

For linear methods, the computational cost is largely determined by the number of nonzeros in each data vector, i.e., the k in our case. For the other parameter b, we recommend to use $b \geq 4$. When using batch algorithms (such as LIBLINEAR), a larger b will in general increase the training time but typically not much. Note that, with online learning [2], it would be obvious that the training time is determined by the number of nonzeros (and number of epoches).

The natural competitor of the GMM kernel is the RBF (radial basis function) kernel, and one competitor of the GCWS hashing method could be the RFF (random Fourier feature) algorithm.

2 RBF KERNEL AND NORMALIZED RANDOM FOURIER FEATURES (NRFF)

The radial basis function (RBF) kernel is widely used in machine learning and many fields. In this study, for convenience (e.g., parameter tuning), we recommend the following version:

$$RBF(u, v; \gamma) = e^{-\gamma(1-\rho)} \tag{8}$$

where $\rho = \rho(u, v)$ is the cosine defined in Eq. (1) and $\gamma > 0$ is a tuning parameter. Based on Bochner's Theorem [23], it is known [21] that, if we sample $w \sim uniform(0, 2\pi)$, $r_i \sim N(0, 1)$ i.i.d., and let $x = \sum_{i=1}^D u_i r_{ij}$, $y = \sum_{i=1}^D v_i r_{ij}$, where $\|u\|_2 = \|v\|_2 = 1$, we have

$$E\left(\sqrt{2}\cos(\sqrt{\gamma}x + w)\sqrt{2}\cos(\sqrt{\gamma}y + w)\right) = e^{-\gamma(1-\rho)} \tag{9}$$

This provides an elegant mechanism for linearizing the RBF kernel and the RFF method has become very popular in machine learning, computer vision, and beyond, e.g., [1, 6, 7, 9, 10, 20, 22, 24, 27, 28].

The result in Theorem 2.1 on the variance of RFF was known, for example, see [25]. In this paper, we provide a proof with general intermediate results which are needed for the proof of Theorem 2.2.

Theorem 2.1. *Given $x \sim N(0,1)$, $y \sim N(0,1)$, $E(xy) = \rho$, and $w \sim uniform(0, 2\pi)$, we have*

$$E\left[\sqrt{2}\cos(\sqrt{\gamma}x + w)\sqrt{2}\cos(\sqrt{\gamma}y + w)\right] = e^{-\gamma(1-\rho)} \tag{10}$$

$$Var\left[\sqrt{2}\cos(\sqrt{\gamma}x + w)\sqrt{2}\cos(\sqrt{\gamma}y + w)\right] = \frac{1}{2} + \frac{1}{2}\left(1 - e^{-2\gamma(1-\rho)}\right)^2 \tag{11}$$

From Theorem 2.1, one can see that the variance of RFF might be large. Interestingly, the variance can be substantially reduced if we normalize the generated data, a procedure which we call "normalized RFF (NRFF)", as shown in Theorem 2.2.

Theorem 2.2. *Consider k iid samples (x_j, y_j, w_j) where $x_j \sim N(0,1)$, $y_j \sim N(0,1)$, $E(x_j y_j) = \rho$, $w_j \sim uniform(0, 2\pi)$, $j = 1, 2, ..., k$. Let $X_j = \sqrt{2}\cos\left(\sqrt{\gamma}x_j + w_j\right)$ and $Y_j = \sqrt{2}\cos\left(\sqrt{\gamma}y_j + w_j\right)$. As $k \to \infty$, the following asymptotic normality holds:*

$$\sqrt{k}\left(\frac{\sum_{j=1}^k X_j Y_j}{\sqrt{\sum_{j=1}^k X_j^2}\sqrt{\sum_{j=1}^k Y_j^2}} - e^{-\gamma(1-\rho)}\right) \xRightarrow{D} N\left(0, V_{n,\rho,\gamma}\right) \tag{12}$$

where

$$V_{n,\rho,\gamma} = V_{\rho,\gamma} - \frac{1}{4}e^{-2\gamma(1-\rho)}\left[3 - e^{-4\gamma(1-\rho)}\right] \tag{13}$$

$$V_{\rho,\gamma} = \frac{1}{2} + \frac{1}{2}\left(1 - e^{-2\gamma(1-\rho)}\right)^2 \tag{14}$$

Obviously, $V_{n,\rho,\gamma} < V_{\rho,\gamma}$ (in particular, $V_{n,\rho,\gamma} = 0$ at $\rho = 1$), i.e., the variance of the normalized RFF is smaller than that of the original RFF. Figure 1 plots $\frac{V_{n,\rho,\gamma}}{V_{\rho,\gamma}}$ to visualize the improvement.

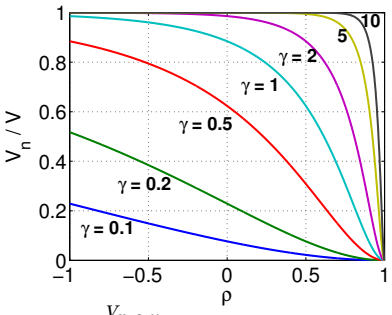

Figure 1: The ratio $\frac{V_{n,\rho,\gamma}}{V_{\rho,\gamma}}$ from Theorem 2.2 for visualizing the improvement due to the normalization step.

Note that the theoretical results in Theorem 2.2 are asymptotic (i.e., for larger k). With k samples, the variance of the original RFF is exactly $\frac{V_{\rho,\gamma}}{k}$, however the variance of the normalized RFF (NRFF) is written as $\frac{V_{n,\rho,\gamma}}{k} + O\left(\frac{1}{k^2}\right)$. It is thus important to understand the behavior when k is not large. For this purpose, Figure 2 presents the simulated mean square error (MSE) results for estimating the RBF kernel $e^{-\gamma(1-\rho)}$, confirming that a): the improvement due to normalization can be substantial, and b): the asymptotic variance formula (13) becomes accurate for merely $k > 10$.

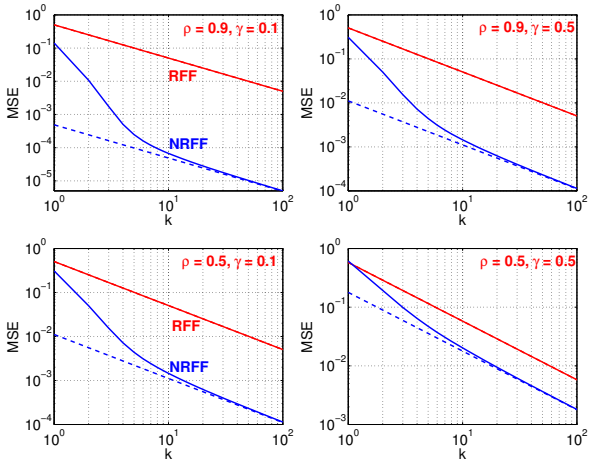

Figure 2: A simulation study to verify the asymptotic theoretical results in Theorem 2.2. With k samples, we estimate the RBF kernel $e^{-\gamma(1-\rho)}$, using both the original RFF and the normalized RFF (NRFF). With 10^5 repetitions at each k, we can compute the empirical mean square error: MSE = Bias^2+Var. Each panel presents the MSEs (solid curves) for a particular choice of (ρ, γ), along with the theoretical variances: $\frac{V_{\rho,\gamma}}{k}$ and $\frac{V_{n,\rho,\gamma}}{k}$ (dashed curves). The variance of the original RFF (curves above, or red if color is available) can be substantially larger than the MSE of the normalized RFF (curves below, or blue). When $k > 10$, the normalized RFF provides an unbiased estimate of the RBF kernel and its empirical MSE matches the theoretical asymptotic variance.

Next, we attempt to compare RFF (and NRFF) with GCWS. While ultimately we can rely on specific machine learning accuracy (e.g., classification, regression, or clustering) as a metric of performance, here we compare their variances (Var) relative to their expectations (E) in terms of Var/E^2, as shown in Figure 3. For example, for GCWS, we know $Var/E^2 = E(1-E)/E^2 = (1-E)/E$.

Figure 3 shows that the relative variance of GCWS is substantially smaller than that of the original RFF and NRFF, especially when E is not large. For the very high similarity region (i.e., $E \to 1$), both NRFF and GCWS exhibit substantially smaller relative variances than the original RFF. Note that in the very high similarity region, the variances of both NRFF and GCWS approach zero.

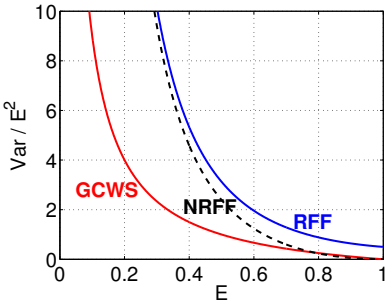

Figure 3: Ratio of the variance over the squared expectation, denoted as Var/E^2, for the convenience of comparing RFF & NRFF with GCWS. Also, see Section 3 and Figure 4.

The results from Figure 3 provide one plausible explanation why later we will observe that, in the classification experiments, GCWS typically needs substantially fewer samples than the normalized RFF, in order to achieve similar classification accuracies. Note that for practical data, the similarities among most data points are usually small (i.e., small E) and hence it is not surprising that GCWS may perform substantially better, according to Figure 3.

While Var/E^2 (known as the squared coefficient of variation) is a standard term in probability and statistics, there might be a concern that we are comparing two different methods which estimate different similarities. Interestingly, it is possible to connect the GMM kernel with cosine (and hence the RBF kernel) if we are allowed to make a gentle model assumption on the data. See Section 3 and Figure 4, which deliver a similar conclusion as Figure 3.

In a sense, this drawback of RFF is expected, due to nature of random projections. For example, as shown in [15], the linear estimator of ρ using random projections has variance $\frac{1+\rho^2}{k}$, where k is the number of projections. Understandably, in order to make the variance small, one will have to use a large number of projections.

Sketch Proof of Theorem 2.1: Firstly, we make use of a known integral: $\int_{-\infty}^{\infty} \cos(cx) e^{-x^2/2} dx = \sqrt{2\pi} e^{-c^2/2}$ to derive two basic integrals:

$$\int_{-\infty}^{\infty} \cos(c_1 x)\cos(c_2 x) e^{-x^2/2} dx = \frac{\sqrt{2\pi}}{2}\left[e^{-(c_1+c_2)^2/2} + e^{-(c_1-c_2)^2/2}\right]$$

$$\int_{-\infty}^{\infty} \sin(c_1 x)\sin(c_2 x) e^{-x^2/2} dx = \frac{\sqrt{2\pi}}{2}\left[e^{-(c_1-c_2)^2/2} - e^{-(c_1+c_2)^2/2}\right]$$

Next, we consider integers $b_1, b_2 = 1, 2, 3, ...$, and, after some tedious algebra, we derive

$$
\begin{aligned}
&E\left(\cos(c_1x + b_1w)\cos(c_2y + b_2w)\right)\\
=&\frac{1}{2\pi}\int_0^{2\pi} E(\cos(c_1x + b_1t)\cos(c_2y + b_2t))dt\\
=&\frac{1}{2\pi}\frac{1}{\sqrt{2\pi}}e^{-\frac{c_2^2(1-\rho^2)}{2}}\int_0^{2\pi}\int_{-\infty}^{\infty} e^{-\frac{x^2}{2}}\cos(c_1x + b_1t)\cos(c_2\rho x + b_2t)dxdt\\
=&\frac{1}{2\pi}\frac{1}{\sqrt{2\pi}}e^{-\frac{c_2^2(1-\rho^2)}{2}}\int_{-\infty}^{\infty} e^{-\frac{x^2}{2}}\pi\cos((c_1 - c_2\rho)x)dx, \quad \text{when } b_1 = b_2\\
=&\frac{1}{2\pi}\frac{1}{\sqrt{2\pi}}e^{-\frac{c_2^2(1-\rho^2)}{2}}\pi\sqrt{2\pi}e^{-(c_1-c_2\rho)^2/2}\\
=&\frac{1}{2}e^{-\frac{c_1^2+c_2^2-2c_1c_2\rho}{2}} = \frac{1}{2}e^{-c^2(1-\rho)}, \qquad \text{when } c_1 = c_2 = c
\end{aligned}
$$

The above integral is otherwise 0 if $b_1 \neq b_2$. This completes the proof of the first moment. Now, using the following fact

$$
\begin{aligned}
E\cos(2cx + 2w) =&\frac{1}{2\pi}\int_0^{2\pi}\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\cos(2cx + 2t)e^{-x^2/2}dxdt\\
=&\frac{1}{2\pi}\int_0^{2\pi}\frac{1}{\sqrt{2\pi}}\frac{1}{2}\sin 2t\int_{-\infty}^{\infty}\cos(2cx)e^{-x^2/2}dxdt\\
=&\frac{1}{4\pi}e^{-2c^2}\int_0^{2\pi}\sin 2tdt = 0
\end{aligned}
$$

we are ready to compute the second moment

$$
\begin{aligned}
&E\left[\cos(cx + w)\cos(cy + w)\right]^2\\
=&\frac{1}{4}E\left[\cos(2cx + 2w)\cos(2cy + 2w) + \cos(2cx + 2w) + \cos(2cy + 2w)\right] + \frac{1}{4}\\
=&\frac{1}{4}E\left[\cos(2cx + 2w)\cos(2cy + 2w)\right] + \frac{1}{4}\\
=&\frac{1}{8}e^{-4c^2(1-\rho)} + \frac{1}{4}
\end{aligned}
$$

and the variance

$$
Var\left[\cos(cx + w)\cos(cy + w)\right] = \frac{1}{8}e^{-4c^2(1-\rho)} + \frac{1}{4} - \frac{1}{4}e^{-2c^2(1-\rho)}
$$

This completes the proof of Theorem 2.1. □

Sketch Proof of Theorem 2.2: Define

$$
X_j = \sqrt{2}\cos(\sqrt{\gamma}x_j + w_j), \qquad Y_j = \sqrt{2}\cos(\sqrt{\gamma}y_j + w_j),
$$

$$
Z_k = \frac{\sum_{j=1}^k X_jY_j}{\sqrt{\sum_{j=1}^k X_j^2}\sqrt{\sum_{j=1}^k Y_j^2}}
$$

From Theorem 2.1, it is easy to see that, as $k \to \infty$, we have

$$
\frac{1}{k}\sum_{j=1}^k X_j^2 \to E\left(X_j^2\right) = e^{-\gamma(1-1)} = 1, \ \ a.s. \qquad \frac{1}{k}\sum_{j=1}^k Y_j^2 \to 1, \ \ a.s.
$$

$$
Z_k = \frac{\frac{1}{k}\sum_{j=1}^k X_jY_j}{\sqrt{\frac{1}{k}\sum_{j=1}^k X_j^2}\sqrt{\frac{1}{k}\sum_{j=1}^k Y_j^2}} \to e^{-\gamma(1-\rho)} = Z_\infty, \ \ a.s.
$$

We express the deviation $Z_k - Z_\infty$ as

$$
\begin{aligned}
Z_k - Z_\infty =&\frac{\frac{1}{k}\sum_{j=1}^k X_jY_j - Z_\infty + Z_\infty}{\sqrt{\frac{1}{k}\sum_{j=1}^k X_j^2}\sqrt{\frac{1}{k}\sum_{j=1}^k Y_j^2}} - Z_\infty\\
=&\frac{\frac{1}{k}\sum_{j=1}^k X_jY_j - Z_\infty}{\sqrt{\frac{1}{k}\sum_{j=1}^k X_j^2}\sqrt{\frac{1}{k}\sum_{j=1}^k Y_j^2}} + Z_\infty\frac{1 - \sqrt{\frac{1}{k}\sum_{j=1}^k X_j^2}\sqrt{\frac{1}{k}\sum_{j=1}^k Y_j^2}}{\sqrt{\frac{1}{k}\sum_{j=1}^k X_j^2}\sqrt{\frac{1}{k}\sum_{j=1}^k Y_j^2}}\\
=&\frac{1}{k}\sum_{j=1}^k X_jY_j - Z_\infty + Z_\infty\frac{1 - \frac{1}{k}\sum_{j=1}^k X_j^2\frac{1}{k}\sum_{j=1}^k Y_j^2}{2} + O_P(1/k)\\
=&\frac{1}{k}\sum_{j=1}^k X_jY_j - Z_\infty + Z_\infty\frac{1 - \frac{1}{k}\sum_{j=1}^k X_j^2}{2} + Z_\infty\frac{1 - \frac{1}{k}\sum_{j=1}^k Y_j^2}{2} + O_P(1/k)
\end{aligned}
$$

Thus, to analyze the asymptotic variance, it suffices to study:

$$
\begin{aligned}
&E\left(XY - Z_\infty + Z_\infty\frac{1-X^2}{2} + Z_\infty\frac{1-Y^2}{2}\right)^2\\
=&E\left(XY - Z_\infty(X^2 + Y^2)/2\right)^2\\
=&E(X^2Y^2) + Z_\infty^2E(X^4 + Y^4 + 2X^2Y^2)/4 - Z_\infty E(X^3Y) - Z_\infty E(XY^3)
\end{aligned}
$$

which can be obtained from the intermediate results in the proof of Theorem 2.1. In particular, if $b_1 = b_2$, then

$$
E\left(\cos(c_1x + b_1w)\cos(c_2y + b_2w)\right) = \frac{1}{2}e^{-\frac{c_1^2+c_2^2-2c_1c_2\rho}{2}}
$$

which is otherwise 0 if $b_1 \neq b_2$. We can now compute

$$
\begin{aligned}
E\left[\cos(cx + w)^3\cos(cy + w)\right] &= \frac{3}{8}e^{-c^2(1-\rho)}\\
E\left[\cos(cx + w)\cos(cy + w)\right]^2 &= \frac{1}{8}e^{-4c^2(1-\rho)} + \frac{1}{4}\\
E\left[\cos(cx + w)\right]^4 &= \frac{1}{8} + \frac{1}{4} = \frac{3}{8}
\end{aligned}
$$

$$
\begin{aligned}
&V_{n,\rho,\gamma} = E\left(XY - Z_\infty + Z_\infty\frac{1-X^2}{2} + Z_\infty\frac{1-Y^2}{2}\right)^2\\
=&E(X^2Y^2) + Z_\infty^2E(X^4 + Y^4 + 2X^2Y^2)/4 - Z_\infty E(X^3Y) - Z_\infty E(XY^3)\\
=&\frac{1}{2}e^{-4c^2(1-\rho)} + 1 + e^{-2c^2(1-\rho)}\left(\frac{3}{8} + \frac{3}{8} + \frac{1}{4}e^{-4c^2(1-\rho)} + \frac{1}{2}\right)\\
&- e^{-c^2(1-\rho)}\left(\frac{3}{2}e^{-c^2(1-\rho)} + \frac{3}{2}e^{-c^2(1-\rho)}\right)\\
=&V_{\rho,\gamma} - \frac{1}{4}e^{-2c^2(1-\rho)}\left[3 - e^{-4c^2(1-\rho)}\right]
\end{aligned}
$$

where $V_{\rho,\gamma}$ is the corresponding variance factor without the normalization step: $V_{\rho,\gamma} = \frac{1}{2} + \frac{1}{2}\left(1 - e^{-2c^2(1-\rho)}\right)^2$. □

3 COMPARING NRFF WITH GCWS USING ASYMPTOTIC OF GMM

The comparison might be more convincing if we can use both NRFF and GCWS to estimate the same object. This task would not be possible unless we can connect GMM with cosine in Eq. (1):

$$
\rho = \rho(u, v) = \frac{\sum_{i=1}^D u_iv_i}{\sqrt{\sum_{i=1}^D u_i^2}\sqrt{\sum_{i=1}^D v_i^2}}
$$

It is well understood that if (u_i, v_i), $i = 1$ to D, are iid samples from a bivariate zero-mean normal distribution with a true correlation parameter $\bar{\rho}$, then as $D \to \infty$, ρ converges to $\bar{\rho}$. However, when data are not exactly normal, as shown in [18], this convergence may not even hold.

For example, when (u_i, v_i), $i = 1$ to D, are iid samples from a bivariate centered t-distribution with ν degrees of freedom and a covariance matrix $\begin{bmatrix} 1 & \bar{\rho} \\ \bar{\rho} & 1 \end{bmatrix}$, the convergence $\rho \to \bar{\rho}$ holds only if $\nu \geq 2$. [18] provided a much more general result by assuming the data follow an "elliptical distribution" (which includes the t distribution). Simply speaking, the use of cosine ρ makes sense only when data have bounded second moments and it becomes reliable when data have bounded fourth (or higher) moments.

Under the same t-distribution assumption, [18] proved an interesting convergence result for the GMM kernel:

$$GMM(u,v) \to \frac{1-\sqrt{(1-\bar{\rho})/2}}{1+\sqrt{(1-\bar{\rho})/2}} = g, \qquad \text{as } D \to \infty \tag{15}$$

The convergence holds as long as $\nu \geq 1$, which is a much weaker assumption than that required for the convergence of cosine. The result was also generalized to the elliptical distribution family.

These theoretical results provide a mechanism to compare GMM with cosine. For example, if one uses both GMM and cosine to estimate the true correlation $\bar{\rho}$, it is more preferable to use GMM when $\nu < 8$. In fact, even when the data are perfectly normal, using GMM will not lose much statistical efficiency compared to using cosine. Since real-world data are virtually always heavy-tailed, the results in [18] appear to suggest one should essentially always use GMM.

In this section, we utilize the results in [18] to compare NRFF with GCWS. To simplify the discussion, we assume that the data dimension D is large enough and the data satisfy the convergence conditions for both GMM and cosine. This way, it is reasonable to assume

$$\rho = \bar{\rho}, \qquad GMM = g = \frac{1-\sqrt{(1-\bar{\rho})/2}}{1+\sqrt{(1-\bar{\rho})/2}} \tag{16}$$

Using the expression of g we can re-write RBF as

$$\rho = 1 - 2\left(\frac{1-g}{1+g}\right)^2, \qquad e^{-\gamma(1-\rho)} = e^{-2\gamma\left(\frac{1-g}{1+g}\right)^2} \tag{17}$$

This result provides us another estimator of the RBF kernel from GCWS hashing, which generates iid samples from a binomial distribution: $binomial(k,g)$, where g is the probability parameter.

Theorem 3.1. *Assume* $g = \frac{1-\sqrt{(1-\rho)/2}}{1+\sqrt{(1-\rho)/2}}$ *and* $X \sim binomial(k,g)$. *Then, denoting* $\bar{X} = \frac{1}{k}\sum_{i=1}^{k} X_i$, *we have*

$$E\left(e^{-2\gamma\left(\frac{1-\bar{X}}{1+\bar{X}}\right)^2}\right) = e^{-\gamma(1-\rho)} + O\left(\frac{1}{k}\right) \tag{18}$$

$$Var\left(e^{-2\gamma\left(\frac{1-\bar{X}}{1+\bar{X}}\right)^2}\right) = \frac{V_{g,\gamma}}{k} + O\left(\frac{1}{k^2}\right) \tag{19}$$

$$V_{g,\gamma} = e^{-2\gamma(1-\rho)}\frac{g(1-g)^3}{(1+g)^6}64\gamma^2 \tag{20}$$

Proof sketch: For an asymptotic analysis with large k, it suffices to consider $Z = \frac{1-\hat{X}}{1+\hat{X}}$ as a normal random variable, whose mean and variance can be calculated to be $\mu = \frac{1-g}{1+g}$, $\sigma^2 = \frac{1}{k}\frac{4g(1-g)}{(1+g)^4}$. A direct calculation shows that $E\left(e^{Z^2 t}\right) = \frac{1}{\sqrt{1-2\sigma^2 t}} e^{\frac{\mu^2 t}{1-2\sigma^2 t}}$. Then, by letting $\sigma^2 = \frac{1}{k}\lambda^2$, we have

$$\begin{aligned} Var\left(e^{X^2 t}\right) =& E\left(e^{X^2 2t}\right) - E^2\left(e^{X^2 t}\right) \\ =& \frac{1}{\sqrt{1-2\sigma^2 2t}} e^{\frac{\mu^2 2t}{1-2\sigma^2 2t}} - \frac{1}{1-2\sigma^2 t} e^{\frac{\mu^2 2t}{1-2\sigma^2 t}} \\ =& \left(1 + \frac{2\lambda^2 t}{k} + O\left(\frac{1}{k^2}\right)\right) e^{2\mu^2 t\left(1+\frac{4\lambda^2 t}{k}+O\left(\frac{1}{k^2}\right)\right)} \\ & - \left(1 + \frac{2\lambda^2 t}{k} + O\left(\frac{1}{k^2}\right)\right) e^{2\mu^2 t\left(1+\frac{2\lambda^2 t}{k}+O\left(\frac{1}{k^2}\right)\right)} \\ =& \frac{4\mu^2\lambda^2 t^2}{k} e^{2\mu^2 t} + O\left(\frac{1}{k^2}\right) \end{aligned}$$

The result follows by letting $t = -2\gamma$, $\mu = \frac{1-g}{1+g}$, and $\lambda^2 = \frac{4g(1-g)}{(1+g)^4}$. □

Theorem 3.1 and Theorem 2.2 provide a more direct comparison of GCWS with NRFF by visualizing $\frac{V_{n,\rho,\gamma}}{V_{g,\gamma}}$. As shown in Figure 4, when estimating the RBF kernel, the variance of GCWS is substantially smaller than the variance of NRFF, except for the very high similarity region (depending on γ). At high similarity, the variances of both methods approach zero. This provides another explanation for the superb empirical performance of GCWS compared to NRFF, as will be reported in Section 5.

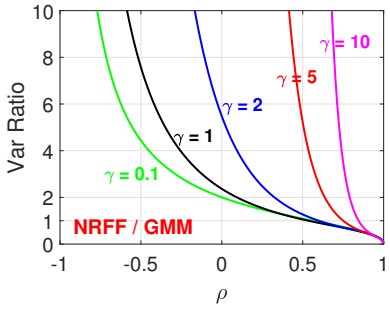

Figure 4: The variance ratio: $\frac{V_{n,\rho,\gamma}}{V_{g,\gamma}}$ provides another comparison of GCWS with NRFF. $V_{g,\gamma}$ is derived in Theorem 3.1 and $V_{n,\rho,\gamma}$ is derived in Theorem 2.2. The ratios are significantly larger than 1 except for the very high similarity region (where the variances of both methods are close to zero).

4 AN EXPERIMENTAL STUDY ON KERNELS

Table 1 lists datasets from the UCI repository. Table 2 presents datasets from the LIBSVM website as well as datasets which are fairly large. Table 3 contains datasets used in an empirical study on deep learning [12] and later for comparing deep nets with tree methods [13]. Except for the relatively large datasets in Table 2, we report the classification accuracies for using kernel SVM with RBF and kernel SVM with GMM, at their the best l_2-regularization

C values. More detailed results (for all regularization C values) are available in Figure 6. To ensure repeatability, we use the LIBSVM pre-computed kernel functionality. This also means we can not (easily) test nonlinear kernels on larger datasets (but then we can resort to hashing methods, e.g., GMM or NRFF).

Table 1: Public (UCI) classification datasets and l_2-regularized kernel SVM results. We report test classification accuracies for the linear kernel, the best-tuned RBF kernel (and the best γ), and the GMM kernel, at their individually- best SVM regularization C values.

Dataset	# train	# test	# dim	linear	RBF (γ)	GMM
Covertype25k	25000	25000	54	62.64	**82.66** (90)	82.65
CTG	1063	1063	35	60.59	**89.75** (0.1)	88.81
Car	864	864	6	71.53	94.91 (100)	**98.96**
DailySports	4560	4560	5625	77.70	97.61 (4)	**99.61**
Dexter	300	300	19999	92.67	93.00 (0.01)	**94.00**
Gesture	4937	4936	32	37.22	61.06 (9)	**65.50**
ImageSeg	210	2100	19	83.81	91.38 (0.4)	**95.05**
Isolet2k	2000	5797	617	93.95	**95.55** (3)	95.53
MSD20k	20000	20000	90	66.72	68.07 (0.1)	**71.05**
MHealth20k	20000	20000	23	72.62	82.65 (0.1)	**85.28**
Magic	9150	9150	10	78.04	84.43 (0.8)	**87.02**
Musk	3299	3299	166	95.09	**99.33** (1.2)	99.24
PageBlocks	2737	2726	10	95.87	**97.08** (1.2)	96.56
Parkinson	520	520	26	61.15	66.73(1.9)	**69.81**
PAMAP101	20000	20000	51	76.86	96.68 (15)	**98.91**
PAMAP102	20000	20000	51	81.22	95.67 (1.1)	**98.78**
PAMAP103	20000	20000	51	85.54	97.89 (19)	**99.69**
PAMAP104	20000	20000	51	84.03	97.32 (19)	**99.30**
PAMAP105	20000	20000	51	79.43	97.34 (18)	**99.22**
RobotNavi	2728	2728	24	69.83	90.69 (10)	**96.85**
Satimage	4435	2000	36	72.45	85.20 (200)	**90.40**
SEMG1	900	900	3000	26.00	**43.56** (4)	41.00
SEMG2	1800	1800	2500	19.28	29.00 (6)	**54.00**
Sensorless	29255	29254	48	61.53	93.01 (0.4)	**99.39**
Shuttle500	500	14500	9	91.81	99.52 (1.6)	**99.65**
SkinSeg10k	10000	10000	3	93.36	99.74 (120)	**99.81**
SpamBase	2301	2300	57	85.91	92.57 (0.2)	**94.17**
Splice	1000	2175	60	85.10	90.02 (15)	**95.22**
Thyroid2k	2000	5200	21	94.90	97.00 (2.5)	**98.40**
Urban	168	507	147	62.52	51.48 (0.01)	**66.08**
Vowel	264	264	10	39.39	94.70 (45)	**96.97**
YoutubeAudio10k	10000	11930	2000	41.35	48.63 (2)	**50.59**
YoutubeHOG10k	10000	11930	647	62.77	66.20 (0.5)	**68.63**
YoutubeMotion10k	10000	11930	64	26.24	28.81 (19)	**31.95**
YoutubeSaiBoxes10k	10000	11930	7168	46.97	49.31 (1.1)	**51.28**
YoutubeSpectrum10k	10000	11930	1024	26.81	33.54 (4)	**39.23**

Table 2: Datasets in groups 1 and 3 are from the LIBSVM website. Datasets in group 2 are from the UCI repository. Datasets in group 2 and 3 are fairly large and are used for testing hashing methods.

	Dataset	# train	# test	# dim	linear	RBF (γ)	GMM
	Letter	15000	5000	16	61.66	**97.44** (11)	97.26
1	Protein	17766	6621	357	69.14	70.32 (4)	**70.64**
	SensIT20k	20000	19705	100	80.42	83.15 (0.1)	**84.57**
	Webspam20k	20000	60000	254	93.00	**97.99** (35)	97.88
2	PAMAP101Large	186,581	186,580	51	79.19		
	PAMAP105Large	185,548	185,548	51	83.35		
	IJCNN	49990	91701	22	92.56		
3	RCV1	338,699	338,700	47,236	97.66		
	SensIT	78,823	19,705	100	80.55		
	Webspam	175,000	175,000	254	93.31		

When experimenting with the RBF kernel, we searched 58 different values of $\gamma \in \{0.001, 0.01, 0.1{:}0.1{:}2, 2.5, 3{:}1{:}20\ 25{:}5{:}50, 60{:}10{:}100, 120, 150, 200, 300, 500, 1000\}$, where for example 60:10:100 means the values are from 60 to 100 spaced at 10. Basically, Tables 1, 2, and 3 report the best RBF results among all γ and C values.

Table 3: Datasets from [12] for comparing deep learning methods with many learning algorithms such as kernel SVM. These datasets were later used by [13] to compare deep nets with tree methods.

Dataset	# train	# test	# dim	linear	RBF (γ)	GMM
M-Basic	12000	50000	784	89.98	**97.21** (5)	96.20
M-Image	12000	50000	784	70.71	77.84 (16)	**80.85**
M-Noise1	10000	4000	784	60.28	66.83 (10)	**71.38**
M-Noise2	10000	4000	784	62.05	69.15 (11)	**72.43**
M-Noise3	10000	4000	784	65.15	71.68 (11)	**73.55**
M-Noise4	10000	4000	784	68.38	75.33 (14)	**76.05**
M-Noise5	10000	4000	784	72.25	78.70 (12)	**79.03**
M-Noise6	10000	4000	784	78.73	**85.33** (15)	84.23
M-Rand	12000	50000	784	78.90	**85.39** (12)	84.22
M-Rotate	12000	50000	784	47.99	**89.68** (5)	84.76
M-RotImg	12000	50000	784	31.44	**45.84** (18)	40.98

The classification results indicate that, on these datasets, kernel (GMM and RBF) SVM classifiers often substantially improve over linear classifiers. Interestingly, For more than half of the datasets, the (tuning-free) GMM kernel outperforms the best-tuned RBF kernel. For a small number of datasets (e.g., "SEMG1" or "M-Rotate"), even though the RBF kernel performs better, we will show in Section 5 that GCWS can still be substantially better than NRFF.

For the datasets in Table 3, since [12] also conducted experiments on the RBF kernel, the polynomial kernel, and neural nets, we assembly the (error rate) results in Figure 5 and Table 4.

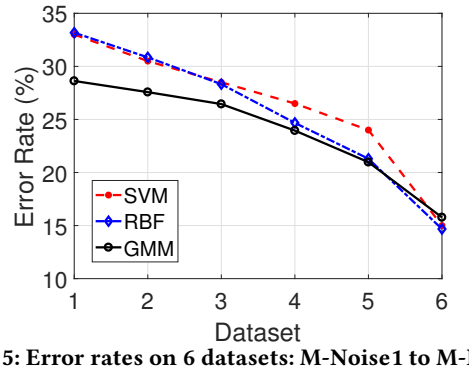

Figure 5: Error rates on 6 datasets: M-Noise1 to M-Noise6 as in Table 3. In this figure, the curve labeled as "SVM" represents the results on RBF kernel SVM conducted by [12], while the curve labeled as "RBF" presents our own experiments. The small discrepancies might be caused by the fact that we always use normalized data (i.e., ρ).

Table 4: Summary of test error rates of various algorithms on several datasets used in [12]. Results in group 1 are reported by [12] for using RBF kernel, polynomial kernel, and neural nets. Results in group 2 are from our own experiments.

Group	Method	M-Basic	M-Rotate	M-Image	M-Rand	M-RotImg
	SVM-RBF	3.05%	11.11%	22.61%	14.58%	55.18%
1	SVM-POLY	3.69%	15.42%	24.01%	16.62%	56.41%
	NNET	4.69%	18.11%	27.41%	20.04%	62.16%
	Linear	10.02%	52.01%	29.29%	21.10%	68.56%
2	RBF	**2.79%**	**10.30%**	22.16%	**14.61%**	**54.16%**
	GMM	3.80%	15.24%	**19.15%**	15.78%	59.02%

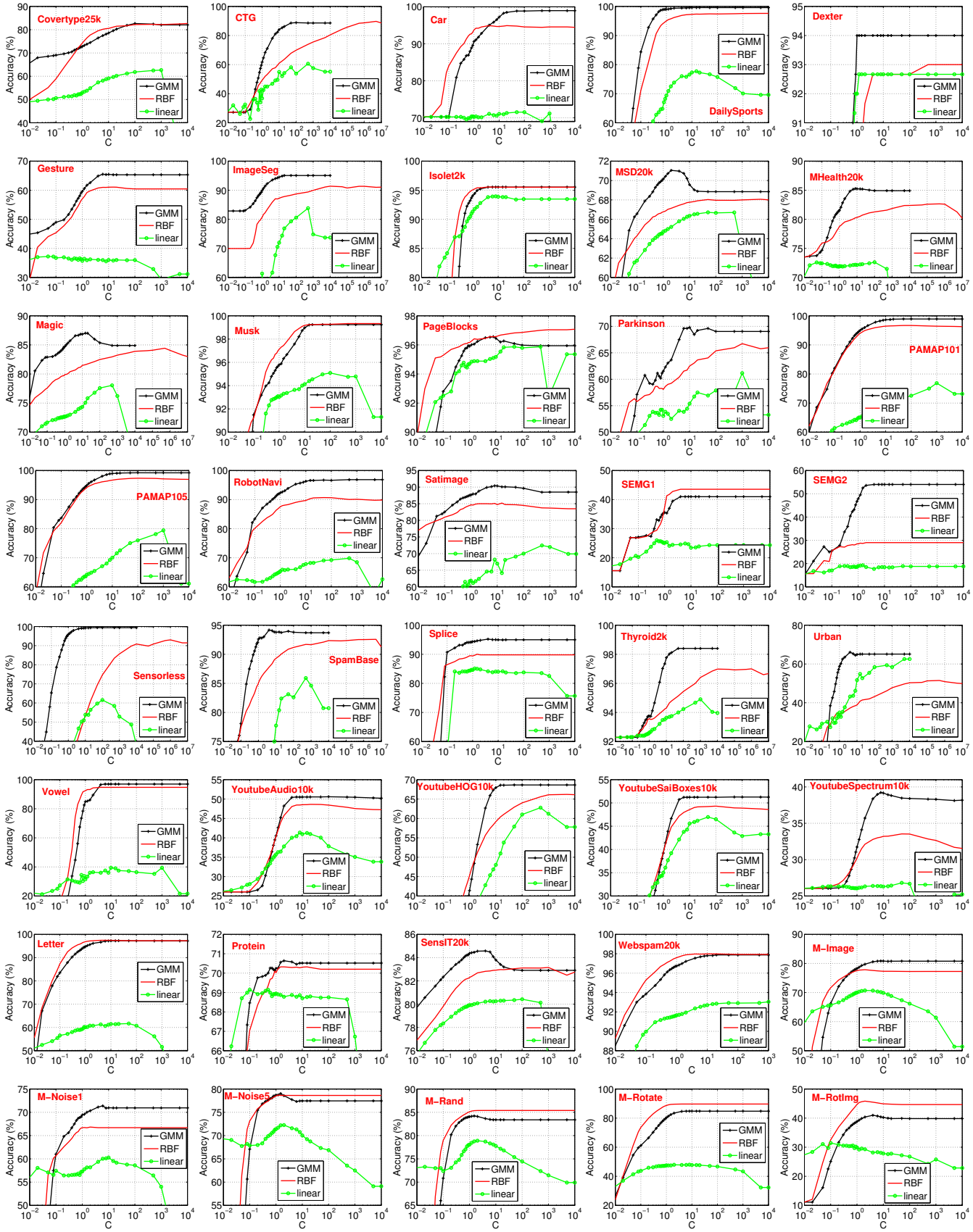

Figure 6: Test classification accuracies using kernel SVMs. Both the GMM kernel and RBF kernel substantially improve linear SVM. C is the l_2-regularization parameter of SVM. For the RBF kernel, we report the result at the best γ value for every C value.

5 HASHING FOR LINEARIZING KERNELS

It is known that a straightforward implementation of nonlinear kernels is difficult for large datasets [3]. For example, for a small dataset with merely $60,000$ data points, the $60,000 \times 60,000$ kernel matrix has 3.6×10^9 entries. In practice, being able to linearize nonlinear kernels becomes highly beneficial, as that would allow us to easily apply efficient linear algorithms especially online learning [2]. Randomization (hashing) is a popular tool for linearization.

In the introduction, we have explained how to linearize both the RBF kernel and the GMM kernel. From practitioner's perspective, while the kernel classification results in Tables 1, 2, and 3 are informative, they are not sufficient for guiding the choice of kernels. For example, as we will show, for some datasets, even though the RBF kernel outperform the GMM kernel, the linearization algorithm (NRFF) requires substantially more samples (i.e., larger k).

We will report detailed experimental results on 4 datasets. As shown in Table 5, on the first two datasets, the RBF and GMM kernels perform similarly. In the second group, the RBF kernel noticeably outperforms the GMM kernel. We will show on all these 4 datasets, the GCWS hashing is substantially more accurate than NRFF at the same number of sample size (k). We will then present less detailed results on more datasets.

Table 5: 4 datasets for presenting detailed experimental results.

Group	Dataset	# train	# test	# dim	linear	RBF (γ)	GMM
1	Letter	15000	5000	16	61.66	**97.44** (11)	97.26
	Webspam20k	20000	60000	254	93.00	**97.99** (35)	97.88
2	SEMG1	900	900	3000	26.00	**43.56** (4)	41.00
	M-Rotate	12000	50000	784	47.99	**89.68** (5)	84.76

Figure 7 reports the test classification accuracies on the **Letter** dataset and **Webspam20k** dataset, for both linearized GMM kernel using GCWS and linearized RBF kernel (at the best γ) using NRFF, using the LIBLINEAR package. Obviously, the results obtained by GCWS hashing are noticeably better than the results of NRFF, especially when the number of samples (k) is not too large.

For the **Letter** dataset, the original dimension is merely 16. It is known that, for modern linear algorithms, the computational cost is largely determined by the number of nonzeros. Hence, the number of samples (i.e., k) is a crucial parameter which directly controls the training complexity. From Figure 7, we can see that with merely $k = 16$ samples, GCWS already produces better results than the original linear method. This phenomenon is exciting, because in industrial practice, the goal is often to produce better results than linear methods without consuming much more resources.

Figure 8 reports the test classification accuracies on the **SEMG1** dataset and **M-Rotate** dataset, respectively. For both datasets, the original RBF kernel considerably outperforms the original GMM kernel. Nevertheless, NRFF still needs substantially more samples than GCWS hashing on both datasets. Again, for GCWS, the results do not differ much once we use $b \geq 4$. Once again, these results confirm the advantage of GCWS hashing for large-scale learning.

Figure 9 reports test classification accuracies on more datasets, only for $b = 8$ and $k \geq 128$, which again confirm that linearization via GCWS works well for the GMM kernel. In contrast, the NRFF approach requires substantially more samples (i.e., larger k).

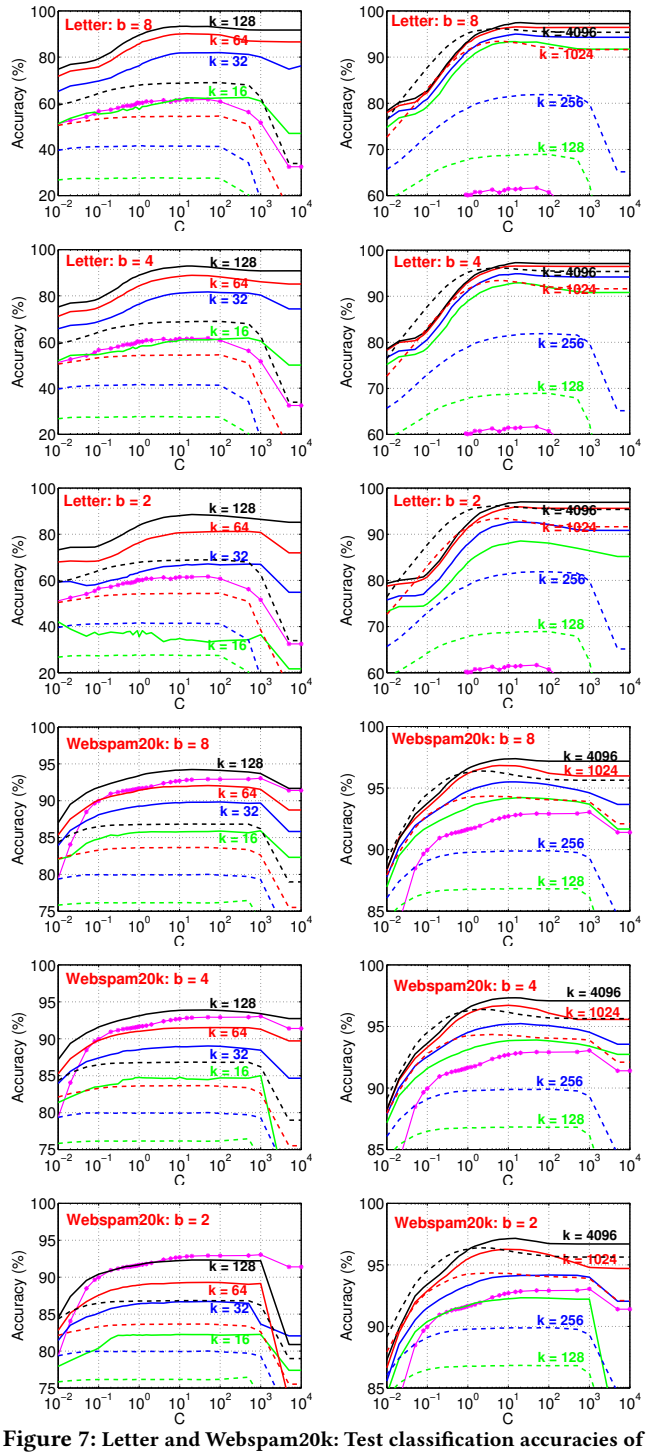

Figure 7: Letter and Webspam20k: Test classification accuracies of the linearized GMM kernel (solid, GCWS) and linearized RBF kernel (dashed, NRFF), using LIBLINEAR, averaged over 10 repetitions. In each panel, we report the results on 4 different k (sample size) values: 128, 256, 1024, 4096 (right panels), and 16, 32, 64, 128 (left panels). We can see that the linearized RBF (using NRFF) would require substantially more samples in order to reach the same accuracies as the linearized GMM kernel (using GCWS). In each panel, the solid curve marked by $*$ presents linear SVM results on the original data.

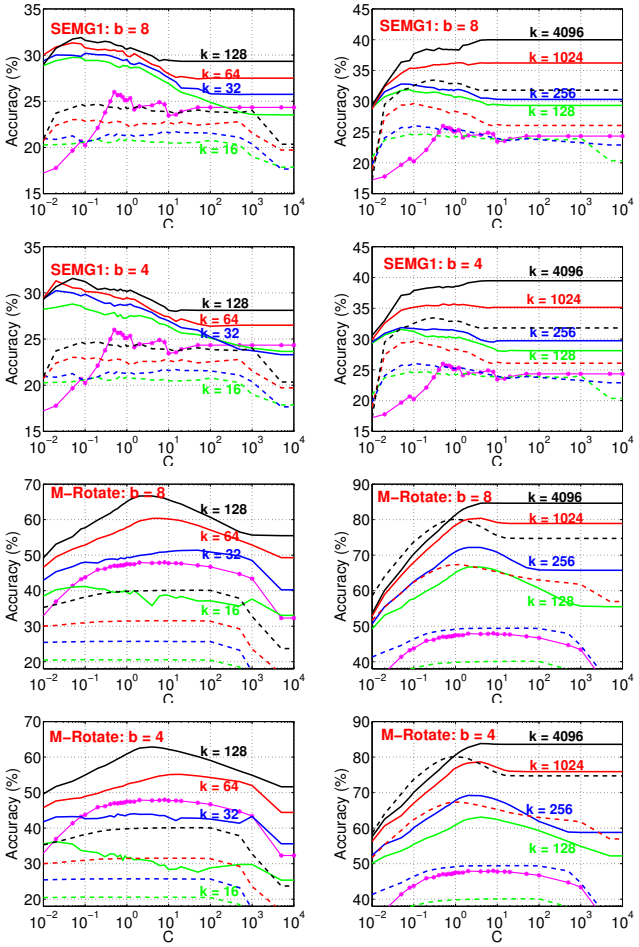

Figure 8: SEMG1 and M-Rotate: Test classification accuracies of the linearized GMM kernel (solid) and linearized RBF (dashed) kernel, using LIBLINEAR. Again, the linearized RBF would require substantially more samples in order to reach the same accuracies as the linearized GMM kernel. On these two datasets, the original RBF kernel outperforms the original GMM kernel as shown in Tables 1 and 3.

Figure 10 presents the hashing results on 6 larger datasets for which we cannot directly train kernel SVMs. We report only for $b = 8$ and k up to 1204. All these results confirm NRFF typically requires substantially more samples than GCWS. This phenomenon can be largely explained by the theoretical results in Theorem 2.2 and Theorem 3.1, which conclude that GCWS is considerably more accurate than NRFF, unless the similarity level is very high.

Training time: For linear algorithms, the training cost is largely determined by the number of nonzero entries per input data vector. In other words, at the same k, the training times of GCWS and NRFF will be roughly comparable. For GCWS and batch algorithms (e.g., LIBLINEAR), a larger b might increase the training time but often not much. See Figure 11 for an example, which in fact shows that NRFF consumed more time at high C (for achieving a good accuracy). Of course, with online learning, it would be more obvious that the training time is determined by the number of nonzeros and number of epoches (it is now common to use only one epoch).

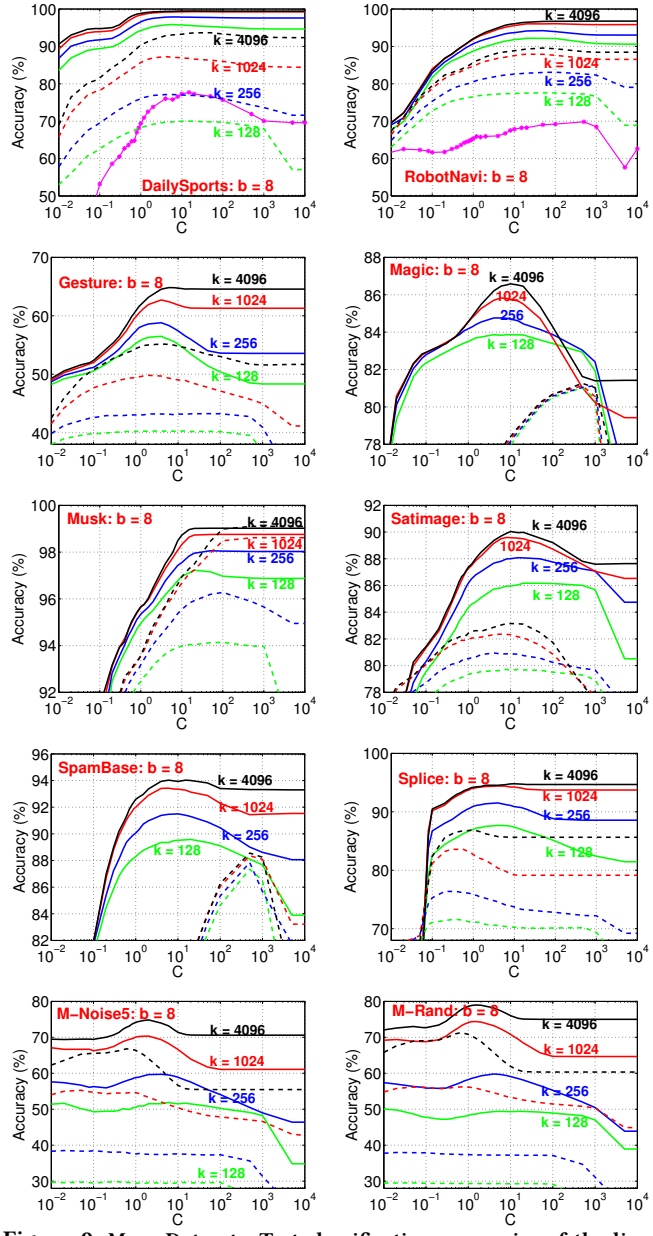

Figure 9: More Datasets: Test classification accuracies of the linearized GMM kernel (solid) and linearized RBF (dashed) kernel. Typically, the linearized RBF would require substantially more samples in order to reach the same accuracies as the linearized GMM kernel.

6 CONCLUSION

Large-scale machine learning has become increasingly important in practice. For many industrial applications, typically only linear methods are affordable, and it is practically beneficial to have methods which can provide substantially more accurate prediction results than linear methods, with no substantial increase of computational cost. The method of "random Fourier features" (RFF) has been a popular tool for linearizing the radial basis function (RBF) kernel, with numerous applications in machine learning, computer

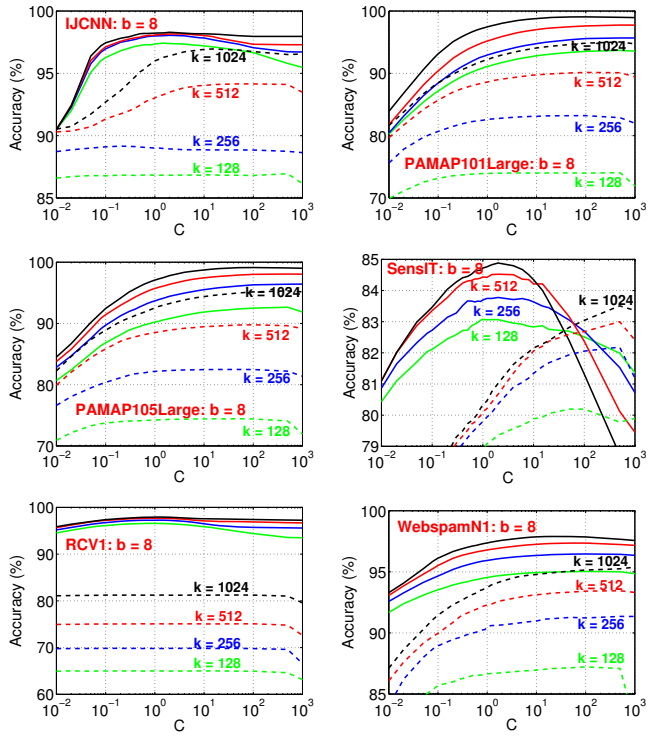

Figure 10: Larger Datasets: Test classification accuracies of the linearized GMM kernel (solid) and linearized RBF (dashed) kernel, on six larger datasets for which we cannot directly compute kernel SVM classifiers on the original data.

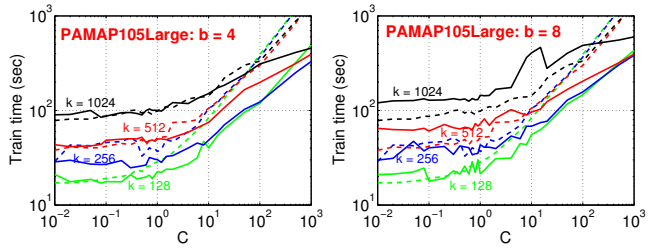

Figure 11: PAMAP105Large: Training times of GCWS (solid curves) and NRFF (dashed curves), for four sample sizes $k \in \{128, 256, 512, 1024\}$, and $b \in \{4, 8\}$.

vision, and beyond, e.g., [1, 6, 7, 9, 10, 20, 22, 24, 27, 28]. In this paper, we rigorously prove that a simple normalization step (i.e., NRFF) can substantially improve the original RFF procedure by considerably reducing the estimation variance.

This paper also proposes the "generalized min-max" (GMM) kernel as a measure of data similarity, to effectively capture data nonlinearity. The GMM kernel can be linearized via the "generalized consistent weighted sampling" (GCWS). Our experimental study demonstrates that usually GCWS does not need too many samples in order to achieve good accuracies. In particular, GCWS typically requires substantially fewer samples than NRFF, in order to reach a similar accuracy. This is practically important, because the training (and testing) cost is largely determined by the number of nonzeros (which is the same as the number of samples in NRFF or GCWS) per data vector of the dataset. The empirical success of GCWS can be explained by our theoretical analysis that the estimation variance of GCWS is usually much smaller than the variance of NRFF.

Given the popularity of the RBF kernel and RFF method, we expect that the GMM kernel and GCWS hashing will also be adopted in a broad range of applications, for example, computer vision [26].

Acknowledgement: NSF-Bigdata-1419210 and NSF-III-1360971.

REFERENCES

[1] Raja Hafiz Affandi, Emily Fox, and Ben Taskar. 2013. Approximate Inference in Continuous Determinantal Processes. In *NIPS*. Lake Tahoe, NV, 1430–1438.

[2] Leon Bottou. http://leon.bottou.org/projects/sgd.

[3] Léon Bottou, Olivier Chapelle, Dennis DeCoste, and Jason Weston (Eds.). 2007. *Large-Scale Kernel Machines*. The MIT Press, Cambridge, MA.

[4] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. 1998. Min-Wise Independent Permutations. In *STOC*. Dallas, TX, 327–336.

[5] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. 1997. Syntactic clustering of the Web. In *WWW*. Santa Clara, CA, 1157 – 1166.

[6] Kacper P Chwialkowski, Aaditya Ramdas, Dino Sejdinovic, and Arthur Gretton. 2015. Fast Two-Sample Testing with Analytic Representations of Probability Measures. In *NIPS*. Montreal, Canada, 1981–1989.

[7] Bo Dai, Bo Xie, Niao He, Yingyu Liang, Anant Raj, Maria-Florina F Balcan, and Le Song. 2014. Scalable Kernel Methods via Doubly Stochastic Gradients. In *NIPS*. Montreal, Canada, 3041–3049.

[8] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research* 9 (2008), 1871–1874.

[9] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. 2014. Predictive Entropy Search for Efficient Global Optimization of Black-box Functions. In *NIPS*. Montreal, Canada, 918–926.

[10] Cho-Jui Hsieh, Si Si, and Inderjit S Dhillon. 2014. Fast Prediction for Large-Scale Kernel Machines. In *NIPS*. Montreal, Canada, 3689–3697.

[11] Sergey Ioffe. 2010. Improved Consistent Sampling, Weighted Minhash and L1 Sketching. In *ICDM*. Sydney, AU, 246–255.

[12] Hugo Larochelle, Dumitru Erhan, Aaron C. Courville, James Bergstra, and Yoshua Bengio. 2007. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*. Corvalis, Oregon, 473–480.

[13] Ping Li. 2010. Robust LogitBoost and Adaptive Base Class (ABC) LogitBoost. In *UAI*. Catalina Island, CA.

[14] Ping Li. 2015. 0-Bit Consistent Weighted Sampling. In *KDD*. Sydney, Australia, 665–674.

[15] Ping Li, Trevor J. Hastie, and Kenneth W. Church. 2006. Improving Random Projections Using Marginal Information. In *COLT*. Pittsburgh, PA, 635–649.

[16] Ping Li and Arnd Christian König. 2010. b-Bit Minwise Hashing. In *WWW*. Raleigh, NC, 671–680.

[17] Ping Li, Anshumali Shrivastava, Joshua Moore, and Arnd Christian König. 2011. Hashing Algorithms for Large-Scale Learning. In *NIPS*. Granada, Spain, 2672–2680.

[18] Ping Li and Cun-Hui Zhang. 2017. Theory of the GMM Kernel. In *WWW*. Perth, Australia, 1053–1062.

[19] Mark Manasse, Frank McSherry, and Kunal Talwar. 2010. *Consistent Weighted Sampling*. Technical Report MSR-TR-2010-73. Microsoft Research.

[20] Maxim Raginsky and Svetlana Lazebnik. 2009. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*. Vancouver, Canada, 1509–1517.

[21] A. Rahimi and B. Recht. 2007. Random features for large-scale kernel machines. In *NIPS*. Vancouver, Canada, 1177–1184.

[22] Emile Richard, Georges A Goetz, and E. J. Chichilnisky. 2015. Recognizing retinal ganglion cells in the dark. In *NIPS*. Montreal, Canada, 2476–2484.

[23] Walter Rudin. 1990. *Fourier Analysis on Groups*. John Wiley & Sons, New York, NY.

[24] Amar Shah and Zoubin Ghahramani. 2015. Parallel Predictive Entropy Search for Batch Global Optimization of Expensive Objective Functions. In *NIPS*. Montreal, Canada, 3330–3338.

[25] Dougal J. Sutherland and Jeff Schneider. 2015. On the Error of Random Fourier Features. In *UAI*. Amsterdam, The Netherlands.

[26] Jing Wang, Jie Shen, and Ping Li. 2017. Object proposal with kernelized partial ranking. *Pattern Recognition* 69 (2017), 299 – 309.

[27] Tianbao Yang, Yu-feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. 2012. Nyström Method vs Random Fourier Features: A Theoretical and Empirical Comparison. In *NIPS*. Lake Tahoe, NV, 476–484.

[28] Ian En-Hsu Yen, Ting-Wei Lin, Shou-De Lin, Pradeep K Ravikumar, and Inderjit S Dhillon. 2014. Sparse Random Feature Algorithm as Coordinate Descent in Hilbert Space. In *NIPS*. Montreal, Canada, 2456–2464.