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 theoretical 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 associated hashing method named "generalized consistent weighted sampling (GCWS)" which linearizes this nonlinear kernel. We provide an extensive empirical evaluation of the RBF kernel and the GMM kernel on more than 50 publicly available datasets. For a majority of the datasets in our experiments, the (tuning-free) GMM kernel outperforms the best-tuned RBF kernel.

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

The empirical success of GCWS (compared to NRFF) can also be explained from a theoretical perspective. 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 are close to zero). Secondly, if we make a gentle model assumption on the data, 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.

Inspired by this work, [15] developed "tunable GMM kernels" which in many datasets considerably improve the (tuning-free) GMM kernel. In fact, kernel SVMs with tunable GMM kernels can be strong competitors to deep nets and boosted trees. [14] compared GMM with the normalized GMM kernel and the intersection kernel. [13] reported the experiments for linearizing GMM with the Nystrom method. [19] developed a theoretical framework for analyzing the convergence property of the GMM kernel using classical statistics, by making model assumptions.

We expect that GMM and GCWS (and their variants) will be adopted in practice for large-scale statistical learning and efficient near neighbor search (as GCWS generates discrete hash values).

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)" and compare it with GCWS.

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 correlation):

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

This normalization step is in general a recommended practice. For example, when using LIBLINEAR or LIBSVM packages [6], it is often suggested to first normalize the input data vectors to unit l_2 norm. In addition to packages such as LIBLINEAR which implement batch linear algorithms, methods based on stochastic gradient descent (SGD) become increasingly important especially for truly large-scale industrial applications [2].

In this paper, 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 [20, 9, 12] on the transformed data.

1.1 Data Transformation

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

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

For example, when $D = 2$ and $u = [-5 \ \ 3]$, the transformed data vector becomes $\tilde{u} = [0 \ \ 5 \ \ 3 \ \ 0]$.

1.2 Generalized Min-Max (GMM) Kernel

Given two data vectors $u, v \in \mathbb{R}^D$, we first transform them into $\tilde{u}, \tilde{v} \in \mathbb{R}^{2D}$ according to (2). Then the generalized min-max (GMM) similarity is defined as

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

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.

[1] This transformation can be generalized by considering a "center vector" μ_i, $i = 1$ to D, such that

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

In this paper, we always use $\mu_i = 0, \ \forall i$. Note that the same center vector μ should be used for all data vectors.

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 (e.g., via hashing). In this paper, we focus on the generalized consistent weighted sampling (GCWS).

1.3 Generalized Consistent Weighted Sampling (GCWS)

Algorithm 1 summarizes the "generalized consistent weighted sampling" (GCWS). Given two data vectors u and v, we transform them into nonnegative vectors $\tilde{u}$ and $\tilde{v}$ as in (2). We then apply the original "consistent weighted sampling" (CWS) [20, 9] to generate random tuples:

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

where $i^* \in [1,\ 2D]$ and t^* is unbounded. Following [20, 9], we have the basic probability result.

Theorem 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). Note that we slightly re-write the expression for a_i compared to [9].

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

With k samples, we can simply use the averaged indicator to estimate $GMM(u, v)$. By property of the binomial distribution, we know the expectation (E) and variance (Var) 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 $\frac{1}{k}(1 - GMM)GMM$, which vanishes as GMM approaches 0 or 1, or as the sample size $k \to \infty$.

1.4 0-bit GCWS for Linearizing GMM Kernel SVM

The so-called "0-bit" GCWS idea is that, based on intensive empirical observations [12], one can safely ignore t^* (which is unbounded) and simply use

$$\mathbf{Pr}\left\{i^*_{\tilde{u},j} = i^*_{\tilde{v},j}\right\} \approx GMM(u, v) \tag{8}$$

For each data vector u, we obtain k random samples $i^*_{\tilde{u},j}$, $j = 1$ to k. We store only the lowest b bits of i^*, based on the idea of [18]. 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 can 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 $b \geq 4$.

The natural competitor of the GMM kernel is the RBF (radial basis function) kernel, and the competitor of the GCWS hashing method is 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 beyond. In this study, for convenience (e.g., parameter tuning), we recommend the following version:

$$RBF(u, v; \gamma) = e^{-\gamma(1-\rho)} \tag{9}$$

where $\rho = \rho(u, v)$ is the correlation defined in (1) and $\gamma > 0$ is a crucial tuning parameter. Based on Bochner's Theorem [24], it is known [22] 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 $\|u\|_2 = \|v\|_2 = 1$, then we have

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

This provides a nice mechanism for linearizing the RBF kernel and the RFF method has become popular in machine learning, computer vision, and beyond, e.g., [21, 27, 1, 7, 5, 28, 8, 25, 4, 23].

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

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

$$E\left[\cos(\sqrt{\gamma}x)\cos(\sqrt{\gamma}y)\right] = \frac{1}{2}e^{-\gamma(1-\rho)} + \frac{1}{2}e^{-\gamma(1+\rho)} \tag{12}$$

$$Var\left[\sqrt{2}\cos(\sqrt{\gamma}x + w)\sqrt{2}\cos(\sqrt{\gamma}y + w)\right] = \frac{1}{2} + \frac{1}{2}\left(1 - e^{-2\gamma(1-\rho)}\right)^2 \tag{13}$$

The proof for (13) can also be found in [26]. One can see that the variance of RFF can be large. Interestingly, the variance can be substantially reduced if we normalize the hashed data, a procedure which we call "normalized RFF (NRFF)". The theoretical results are presented in Theorem 3.

Theorem 3 *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:*

$$\sqrt{k}\left(\frac{\sum_{j=1}^{k} X_j Y_j}{\sqrt{\sum_{j=1}^{k} X_j^2}\sqrt{\sum_{j=1}^{k} Y_j^2}} - e^{-\gamma(1-\rho)}\right) \xRightarrow{D} N\left(0, V_{n,\rho,\gamma}\right) \tag{14}$$

where

$$V_{n,\rho,\gamma} = V_{\rho,\gamma} - \frac{1}{4}e^{-2\gamma(1-\rho)}\left[3 - e^{-4\gamma(1-\rho)}\right] \tag{15}$$

$$V_{\rho,\gamma} = \frac{1}{2} + \frac{1}{2}\left(1 - e^{-2\gamma(1-\rho)}\right)^2 \tag{16}$$

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 (much) smaller than that of the original RFF. Figure 1 plots $\frac{V_{n,\rho,\gamma}}{V_{\gamma,\gamma}}$ to visualize the improvement due to normalization, which is most significant when ρ is close to 1.

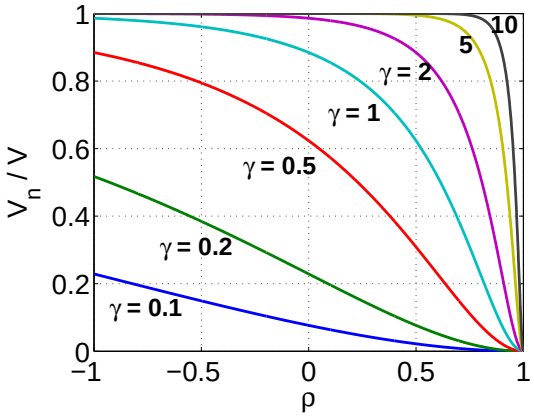

Figure 1: The ratio $\frac{V_{n,\rho,\gamma}}{V_{\gamma,\gamma}}$ from Theorem 3 for visualizing the improvement due to normalization.

Note that the theoretical results in Theorem 3 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 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 (15) becomes accurate for merely $k > 10$.

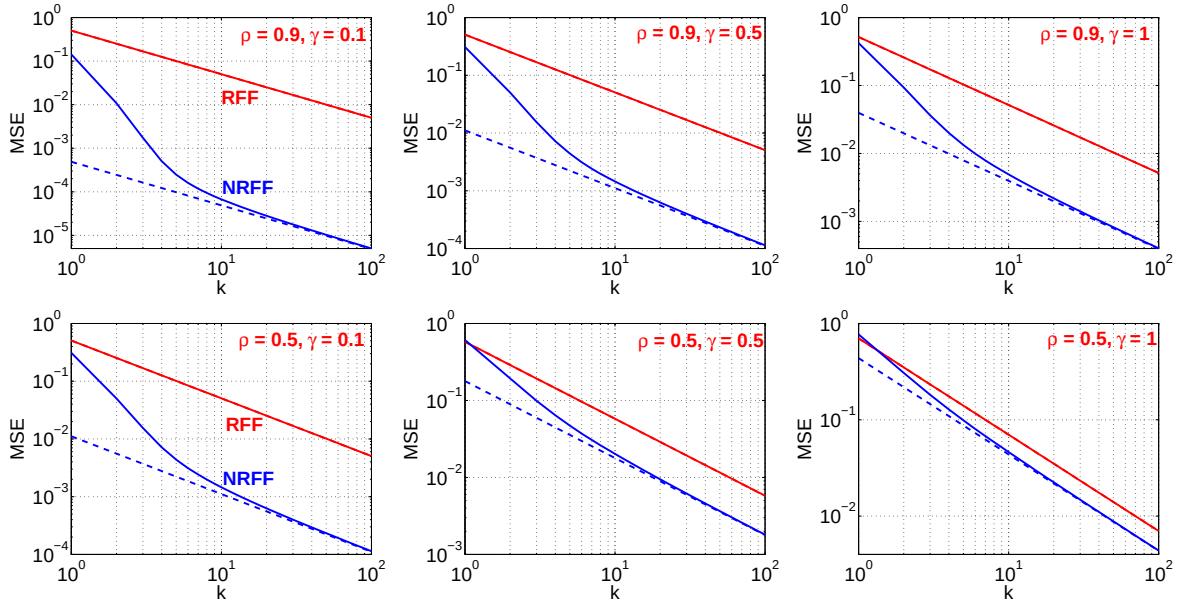

Figure 2: A simulation study to verify the asymptotic theoretical results in Theorem 3. With k samples, we estimate the RBF kernel $e^{-\gamma(1-\rho)}$, using both the original RFF and the normalized RFF (NRFF). With 10^5 repetitions at each k, we can compute the empirical mean square error: MSE = Bias^2+Var. 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 with GCWS. While ultimately we can rely on classification accuracy as a metric for performance, here we compare their variances (Var) relative to their expectations (E) in terms of Var/E^2, as shown in Figure 3. For GCWS, we know $Var/E^2 = E(1-E)/E^2 = (1-E)/E$. For the original RFF, we have $Var/E^2 = \left[\frac{1}{2} + \frac{1}{2}\left(1-E^2\right)^2\right]/E^2$, etc.

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

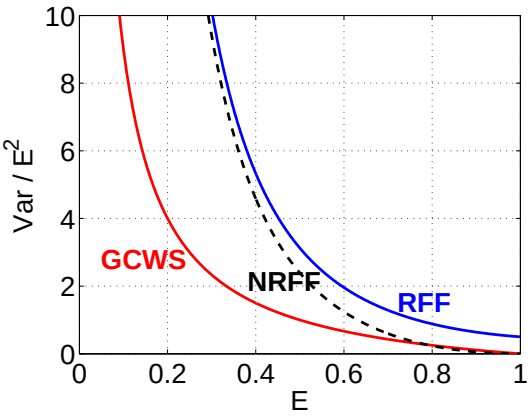

Figure 3: Ratio of the variance over the squared expectation, denoted as Var/E^2, for the convenience of comparing RFF/NRFF with GCWS. Smaller (lower) is better.

The results from Figure 3 provide one 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. Also see Section 3 and Figure 4 for a comparison from the perspective of estimating RBF using GCWS based on a model assumption.

In a sense, this drawback of RFF is expected, due to nature of random projections. For example, as shown in [16, 17], the linear estimator of the correlation ρ using random projections has variance $\frac{1+\rho^2}{k}$, where k is the number of projections. In order to make the variance small, one will have to use many projections (i.e., large k).

Proof of Theorem 2: The following three integrals will be useful in our proof:

$$\int_{-\infty}^{\infty} \cos(cx) e^{-x^2/2} dx = \sqrt{2\pi} e^{-c^2/2}$$

$$\begin{aligned}\int_{-\infty}^{\infty} \cos(c_1 x)\cos(c_2 x) e^{-x^2/2} dx =& \frac{1}{2}\int_{-\infty}^{\infty} \left[\cos((c_1+c_2)x) + \cos((c_1-c_2)x)\right] e^{-x^2/2} dx \\ =& \frac{\sqrt{2\pi}}{2}\left[e^{-(c_1+c_2)^2/2} + e^{-(c_1-c_2)^2/2}\right]\end{aligned}$$

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

Firstly, we consider integers $b_1, b_2 = 1, 2, 3, ...$, and evaluate the following general integral:

$$
\begin{aligned}
&E\left(\cos(c_1x + b_1w)\cos(c_2y + b_2w)\right)\\
=&\frac{1}{2\pi}\int_0^{2\pi} E(\cos(c_1x + b_1t)\cos(c_2y + b_2t))dt\\
=&\frac{1}{2\pi}\int_0^{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(\cos(c_1x + b_1t)\cos(c_2y + b_2t))\frac{1}{2\pi}\frac{1}{\sqrt{1-\rho^2}}e^{-\frac{x^2+y^2-2\rho xy}{2(1-\rho^2)}}dxdydt\\
=&\frac{1}{2\pi}\int_0^{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(\cos(c_1x + b_1t)\cos(c_2y + b_2t))\frac{1}{2\pi}\frac{1}{\sqrt{1-\rho^2}}e^{-\frac{x^2+y^2-2\rho xy+\rho^2x^2-\rho^2x^2}{2(1-\rho^2)}}dxdydt\\
=&\frac{1}{2\pi}\int_0^{2\pi}\int_{-\infty}^{\infty}\frac{1}{2\pi}\frac{1}{\sqrt{1-\rho^2}}e^{-\frac{x^2}{2}}\cos(c_1x + b_1t)dx\int_{-\infty}^{\infty}\cos(c_2y + b_2t)e^{-\frac{(y-\rho x)^2}{2(1-\rho^2)}}dydt\\
=&\frac{1}{2\pi}\int_0^{2\pi}\int_{-\infty}^{\infty}\frac{1}{2\pi}e^{-\frac{x^2}{2}}\cos(c_1x + b_1t)dx\int_{-\infty}^{\infty}\cos(c_2y\sqrt{1-\rho^2} + c_2\rho x + b_2t)e^{-y^2/2}dydt\\
=&\frac{1}{2\pi}\int_0^{2\pi}\int_{-\infty}^{\infty}\frac{1}{2\pi}e^{-\frac{x^2}{2}}\cos(c_1x + b_1t)\cos(c_2\rho x + b_2t)dx\int_{-\infty}^{\infty}\cos(c_2y\sqrt{1-\rho^2})e^{-y^2/2}dydt\\
=&\frac{1}{2\pi}\int_0^{2\pi}\int_{-\infty}^{\infty}\frac{1}{2\pi}e^{-\frac{x^2}{2}}\cos(c_1x + b_1t)\cos(c_2\rho x + b_2t)\sqrt{2\pi}e^{-\frac{c_2^2(1-\rho^2)}{2}}dxdt\\
=&\frac{1}{2\pi}\frac{1}{\sqrt{2\pi}}e^{-\frac{c_2^2(1-\rho^2)}{2}}\int_0^{2\pi}\int_{-\infty}^{\infty}e^{-\frac{x^2}{2}}\cos(c_1x + b_1t)\cos(c_2\rho x + b_2t)dxdt
\end{aligned}
$$

Note that

$$
\begin{aligned}
&\int_0^{2\pi}\cos(c_1x + b_1t)\cos(c_2\rho x + b_2t)dt\\
=&\int_0^{2\pi}\cos(c_1x)\cos(b_1t)\cos(c_2\rho x)\cos(b_2t)dt + \int_0^{2\pi}\sin(c_1x)\sin(b_1t)\sin(c_2\rho x)\sin(b_2t)dt\\
-&\int_0^{2\pi}\cos(c_1x)\cos(b_1t)\sin(c_2\rho x)\sin(b_2t)dt - \int_0^{2\pi}\sin(c_1x)\sin(b_1t)\cos(c_2\rho x)\cos(b_2t)dt
\end{aligned}
$$

When $b_1 \neq b_2$, we have

$$
\begin{aligned}
\int_0^{2\pi}\cos(b_1t)\cos(b_2t)dt &= \frac{1}{2}\int_0^{2\pi}\cos(b_1t - b_2t) + \cos(b_1t + b_2t)dt = 0\\
\int_0^{2\pi}\sin(b_1t)\sin(b_2t)dt &= \frac{1}{2}\int_0^{2\pi}\cos(b_1t - b_2t) - \cos(b_1t + b_2t)dt = 0
\end{aligned}
$$

If $b_1 = b_2$, then

$$
\int_0^{2\pi}\cos(b_1t)\cos(b_2t)dt = \int_0^{2\pi}\sin(b_1t)\sin(b_2t)dt = \pi
$$

In addition, for any $b_1, b_2 = 1, 2, 3, ...$, we always have

$$
\int_0^{2\pi}\sin(b_1t)\cos(b_2t)dt = \frac{1}{2}\int_0^{2\pi}\sin(b_1t - b_2t) + \sin(b_1t + b_2t)dt = 0
$$

Thus, only when $b_1 = b_2$ we have

$$\int_0^{2\pi} \cos(c_1 x + b_1 t)\cos(c_2\rho x + b_2 t)dt = \pi\cos(c_1 x)\cos(c_2\rho x) + \pi\sin(c_1 x)\sin(c_2\rho x) = \pi\cos((c_1 - c_2\rho)x)$$

Otherwise, $\int_0^{2\pi} \cos(c_1 x + b_1 t)\cos(c_2\rho x + b_2 t)dt = 0$. Therefore, when $b_1 = b_2$, we have

$$\begin{aligned}
&E\left(\cos(c_1 x + b_1 w)\cos(c_2 y + b_2 w)\right)\\
=&\frac{1}{2\pi}\frac{1}{\sqrt{2\pi}}e^{-\frac{c_2^2(1-\rho^2)}{2}}\int_0^{2\pi}\int_{-\infty}^{\infty} e^{-\frac{x^2}{2}}\cos(c_1 x + b_1 t)\cos(c_2\rho x + b_2 t)dxdt\\
=&\frac{1}{2\pi}\frac{1}{\sqrt{2\pi}}e^{-\frac{c_2^2(1-\rho^2)}{2}}\int_{-\infty}^{\infty} e^{-\frac{x^2}{2}}\pi\cos((c_1 - c_2\rho)x)dx\\
=&\frac{1}{2\pi}\frac{1}{\sqrt{2\pi}}e^{-\frac{c_2^2(1-\rho^2)}{2}}\pi\sqrt{2\pi}e^{-(c_1 - c_2\rho)^2/2}\\
=&\frac{1}{2}e^{-\frac{c_1^2 + c_2^2 - 2c_1 c_2\rho}{2}}\\
=&\frac{1}{2}e^{-c^2(1-\rho)}, \qquad \text{when } \; c_1 = c_2 = c
\end{aligned}$$

This completes the proof of the first moment. Next, using the following fact

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

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

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

Finally, we prove the first moment without the "w" random variable:

$$\begin{aligned}
E\left(\cos(cx)\cos(cy)\right) =& \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cos(cx)\cos(cy)\frac{1}{2\pi}\frac{1}{\sqrt{1-\rho^2}}e^{-\frac{x^2+y^2-2\rho xy+\rho^2x^2-\rho^2x^2}{2(1-\rho^2)}}dxdy \\
=& \int_{-\infty}^{\infty}\frac{1}{2\pi}\frac{1}{\sqrt{1-\rho^2}}e^{-\frac{x^2}{2}}\cos(cx)dx\int_{-\infty}^{\infty}\cos(cy)e^{-\frac{(y-\rho x)^2}{2(1-\rho^2)}}dy \\
=& \int_{-\infty}^{\infty}\frac{1}{2\pi}e^{-\frac{x^2}{2}}\cos(cx)dx\int_{-\infty}^{\infty}\cos(cy\sqrt{1-\rho^2}+c\rho x)e^{-y^2/2}dy \\
=& \int_{-\infty}^{\infty}\frac{1}{2\pi}e^{-\frac{x^2}{2}}\cos(cx)\cos(c\rho x)dx\int_{-\infty}^{\infty}\cos(cy\sqrt{1-\rho^2})e^{-y^2/2}dy \\
=& \int_{-\infty}^{\infty}\frac{1}{2\pi}e^{-\frac{x^2}{2}}\cos(cx)\cos(c\rho x)\sqrt{2\pi}e^{-c^2\frac{1-\rho^2}{2}}dx \\
=& \frac{1}{\sqrt{2\pi}}e^{-c^2\frac{1-\rho^2}{2}}\int_{-\infty}^{\infty}e^{-\frac{x^2}{2}}\cos(cx)\cos(c\rho x)dx \\
=& \frac{1}{\sqrt{2\pi}}e^{-c^2\frac{1-\rho^2}{2}}\frac{\sqrt{2\pi}}{2}\left[e^{-c^2\frac{(1-\rho)^2}{2}}+e^{-c^2\frac{(1+\rho)^2}{2}}\right] \\
=& \frac{1}{2}e^{-c^2(1-\rho)}+\frac{1}{2}e^{-c^2(1+\rho)}
\end{aligned}$$

This completes the proof of Theorem 2. □

Proof of Theorem 3: We will use some of the results from the proof of Theorem 2. Define

$$X_j=\sqrt{2}\cos(\sqrt{\gamma}x_j+w_j),\quad Y_j=\sqrt{2}\cos(\sqrt{\gamma}y_j+w_j),\quad Z_k=\frac{\sum_{j=1}^k X_jY_j}{\sqrt{\sum_{j=1}^k X_j^2}\sqrt{\sum_{j=1}^k Y_j^2}}$$

From Theorem 2, it is easy to see that, as $k\to\infty$, we have

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

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

We express the deviation Z_k-Z_∞ as

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

Note that if $a \approx 1$ and $b \approx 1$, then

$$1 - ab = 1 - (1 - (1 - a))(1 - (1 - b)) = (1 - a) + (1 - b) - (1 - a)(1 - b)$$

and we can ignore the higher-order term.

Therefore, to analyze the asymptotic variance, it suffices to study the following expectation

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

which can be obtained from the results in the proof of Theorem 2. In particular, if $b_1 = b_2$, then

$$E\left(\cos(c_1 x + b_1 w)\cos(c_2 y + b_2 w)\right) = \frac{1}{2} e^{-\frac{c_1^2 + c_2^2 - 2c_1 c_2 \rho}{2}}$$

Otherwise $E\left(\cos(c_1 x + b_1 w)\cos(c_2 y + b_2 w)\right) = 0$. We can now compute

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

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

$$E\left[\cos(cx + w)\right]^4 = \frac{1}{8} + \frac{1}{4} = \frac{3}{8}$$

$$\begin{aligned}
V_{n,\rho,\gamma} =&E\left(XY - Z_\infty + Z_\infty \frac{1 - X^2}{2} + Z_\infty \frac{1 - Y^2}{2}\right)^2 \\
=&E(X^2Y^2) + Z_\infty^2 E(X^4 + Y^4 + 2X^2Y^2)/4 - Z_\infty E(X^3Y) - Z_\infty E(XY^3) \\
=&\frac{1}{2} e^{-4c^2(1-\rho)} + 1 + e^{-2c^2(1-\rho)}\left(\frac{3}{8} + \frac{3}{8} + \frac{1}{4} e^{-4c^2(1-\rho)} + \frac{1}{2}\right) - e^{-c^2(1-\rho)}\left(\frac{3}{2} e^{-c^2(1-\rho)} + \frac{3}{2} e^{-c^2(1-\rho)}\right) \\
=&\frac{1}{2} e^{-4c^2(1-\rho)} + 1 + e^{-2c^2(1-\rho)}\left(\frac{5}{4} + \frac{1}{4} e^{-4c^2(1-\rho)}\right) - 3e^{-2c^2(1-\rho)} \\
=&\frac{1}{2} e^{-4c^2(1-\rho)} + 1 + \frac{1}{4} e^{-6c^2(1-\rho)} - \frac{7}{4} e^{-2c^2(1-\rho)} \\
=&V_{\rho,\gamma} - \frac{1}{4} e^{-2c^2(1-\rho)}\left[3 - e^{-4c^2(1-\rho)}\right]
\end{aligned}$$

where $V_{\rho,\gamma}$ is the corresponding variance factor without using normalization:

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

This completes the proof of Theorem 3. □

3 Another Comparison Based on Asymptotic of GMM

As proved in a technical report following this paper [19], under mild model assumption, as the dimension D becomes large, the GMM kernel converges to a function of the true data correlation:

$$GMM \rightarrow \frac{1-\sqrt{(1-\rho)/2}}{1+\sqrt{(1-\rho)/2}} = g \tag{17}$$

The convergence holds almost surely for data with bounded first moment. Using the expression of g we can express RBF $e^{-\gamma(1-\rho)}$ in terms of g:

$$\rho = 1 - 2\left(\frac{1-g}{1+g}\right)^2, \qquad e^{-\gamma(1-\rho)} = e^{-2\gamma\left(\frac{1-g}{1+g}\right)^2} \tag{18}$$

For the convenience of conducting theoretical analysis, we assume $GMM = \frac{1-\sqrt{(1-\rho)/2}}{1+\sqrt{(1-\rho)/2}} = g$, exactly instead of asymptotically. Then we have another estimator of the RBF kernel from GCWS. Note that with k hashes, the estimate of GMM follows a binomial distribution $binomial(k, g)$.

Theorem 4 *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*

$$E\left(e^{-2\gamma\left(\frac{1-\bar{X}}{1+\bar{X}}\right)^2}\right) = e^{-\gamma(1-\rho)} + O\left(\frac{1}{k}\right) \tag{19}$$

$$Var\left(e^{-2\gamma\left(\frac{1-\bar{X}}{1+\bar{X}}\right)^2}\right) = \frac{V_{g,\gamma}}{k} + O\left(\frac{1}{k^2}\right) \tag{20}$$

$$where \;\; V_{g,\gamma} = e^{-2\gamma(1-\rho)}\frac{g(1-g)^3}{(1+g)^6}64\gamma^2 \tag{21}$$

Proof of Theorem 4: 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 $\mu = \frac{1-g}{1+g}$, $\sigma^2 = \frac{1}{k}\frac{4g(1-g)}{(1+g)^4}$. Thus, it suffices to compute

$$\begin{aligned}
E\left(e^{X^2 t}\right) &= \int_{-\infty}^{\infty} e^{x^2 t}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}dx = \int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2-2\sigma^2x^2t}{2\sigma^2}}dx \\
&= \int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2-2\sigma^2x^2t}{2\sigma^2}}dx = \int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(1-2\sigma^2t)x^2-2\mu x+\mu^2}{2\sigma^2}}dx \\
&= \int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{x^2-2\mu/c^2x+\mu^2/c^2}{2\sigma^2/c^2}}dx, \qquad \text{where } c^2 = 1-2\sigma^2t \\
&= \frac{1}{c}\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}\sigma/c}e^{-\frac{(x-\mu/c^2)^2-\mu^2/c^4+\mu^2/c^2}{2\sigma^2/c^2}}dx \\
&= \frac{1}{c}e^{\frac{\mu^2(1-c^2)}{2\sigma^2c^2}} = \frac{1}{c}e^{\frac{\mu^2}{c^2}t} = \frac{1}{\sqrt{1-2\sigma^2t}}e^{\frac{\mu^2t}{1-2\sigma^2t}}
\end{aligned}$$

from which we can compute the variance (letting $\sigma^2 = \frac{1}{k}\lambda^2$)

$$
\begin{aligned}
Var\left(e^{X^2 t}\right) =& E\left(e^{X^2 2t}\right) - E^2\left(e^{X^2 t}\right) = \frac{1}{\sqrt{1-2\sigma^2 2t}} e^{\frac{\mu^2 2t}{1-2\sigma^2 2t}} - \frac{1}{1-2\sigma^2 t} e^{\frac{\mu^2 2t}{1-2\sigma^2 t}} \\
=& \left(1 + \frac{2\lambda^2 t}{k} + O\left(\frac{1}{k^2}\right)\right) e^{2\mu^2 t\left(1+\frac{4\lambda^2 t}{k}+O\left(\frac{1}{k^2}\right)\right)} - \left(1 + \frac{2\lambda^2 t}{k} + O\left(\frac{1}{k^2}\right)\right) e^{2\mu^2 t\left(1+\frac{2\lambda^2 t}{k}+O\left(\frac{1}{k^2}\right)\right)} \\
=& \left(1 + O\left(\frac{1}{k}\right)\right) e^{2\mu^2 t}\left(1 + \frac{8\mu^2\lambda^2 t^2}{k} + O\left(\frac{1}{k^2}\right)\right) - \left(1 + O\left(\frac{1}{k}\right)\right) e^{2\mu^2 t}\left(1 + \frac{4\mu^2\lambda^2 t^2}{k} + O\left(\frac{1}{k^2}\right)\right) \\
=& \frac{4\mu^2\lambda^2 t^2}{k} e^{2\mu^2 t} + O\left(\frac{1}{k^2}\right)
\end{aligned}
$$

Plugging in $t = -2\gamma$, $\mu = \frac{1-g}{1+g}$, and $\lambda^2 = \frac{4g(1-g)}{(1+g)^4}$, yields

$$
Var\left(e^{-2\gamma\left(\frac{1-\bar{X}}{1+\bar{X}}\right)^2}\right) = \frac{64\gamma^2}{k}\frac{g(1-g)^3}{(1+g)^6} e^{-4\gamma\left(\frac{1-g}{1+g}\right)^2} + O\left(\frac{1}{k^2}\right) = \frac{64\gamma^2}{k}\frac{g(1-g)^3}{(1+g)^6} e^{-2\gamma(1-\rho)} + O\left(\frac{1}{k^2}\right)
$$

□

This theoretical result provides a direct comparison of GCWS with NRFF for estimating the same object, by visualizing the variance ratio: $\frac{V_{n,\rho,\gamma}}{V_{g,\gamma}}$, using results from Theorem 3. As shown in Figure 4, for 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 later in the paper.

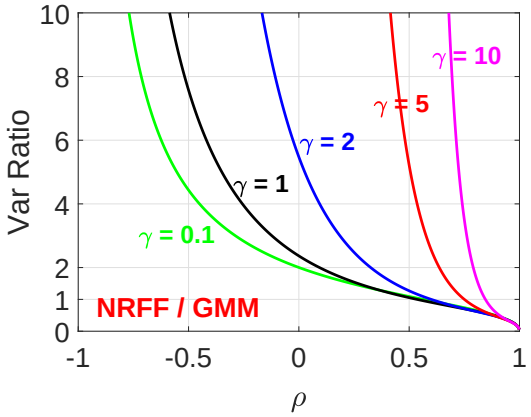

Figure 4: The variance ratio: $\frac{V_{n,\rho,\gamma}}{V_{g,\gamma}}$ provides another comparison of GCWS with NRFF. $V_{g,\gamma}$ is derived in Theorem 4 and $V_{n,\rho,\gamma}$ is derived in Theorem 3. The ratios are significantly larger than 1 except for the very high similarity region (where the variances of both methods are close to zero).

4 An Experimental Study on Kernel SVMs

Table 1 lists datasets from the UCI repository. Table 2 presents datasts from the LIBSVM website as well as datasets which are fairly large. Table 3 contains datasets used for evaluating deep learning and trees [10, 11]. Except for the relatively large datasets in Table 2, we also report the classification accuracies for the linear SVM, kernel SVM with RBF, and kernel SVM with GMM, at the best l_2-regularization C values. More detailed results (for all regularization C values) are available in Figures 5, 6, 7, and 8. To ensure repeatability, we use the LIBSVM pre-computed kernel functionality. This also means we can not (easily) test nonlinear kernels on larger datasets.

For the RBF kernel, we exhaustively experimented with 58 different values of $\gamma \in \{0.001, 0.01, 0.1{:}0.1{:}2, 2.5, 3{:}1{:}20\ 25{:}5{:}50, 60{:}10{:}100, 120, 150, 200, 300, 500, 1000\}$. Basically, Tables 1, 2, and 3 reports the best RBF results among all γ and C values in our experiments.

The classification results indicate that, on these datasets, kernel (GMM and RBF) SVM classifiers improve over linear classifiers substantially. For more than half of the datasets, the GMM kernel (which has no tuning parameter) outperforms the best-tuned RBF kernel. For a small number of datasets (e.g., "SEMG1"), even though the RBF kernel performs better, we will show in Section 5 that the GCWS hashing can still be substantially better than the NRFF hashing.

Table 1: **Public (UCI) classification datasets and l_2-regularized kernel SVM results**. We report the 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
Car	864	864	6	71.53	94.91 (100)	**98.96**
Covertype25k	25000	25000	54	62.64	**82.66** (90)	82.65
CTG	1063	1063	35	60.59	**89.75** (0.1)	88.81
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 group 1 and group 3 are from the LIBSVM website. Datasets in group 2 are from the UCI repository. Datasets in group 2 and 3 are too large for LIBSVM pre-computed kernel functionality and are thus only used for testing hashing methods.

Group	Dataset	# train	# test	# dim	linear	RBF (γ)	GMM
1	Letter	15000	5000	16	61.66	**97.44** (11)	97.26
	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		
3	IJCNN	49990	91701	22	92.56		
	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		

Table 3: Datasets from [10, 11]. See the technical report [15] on "tunable GMM kernels" for substantially improved results, by introducing tuning parameters in the GMM kernel.

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

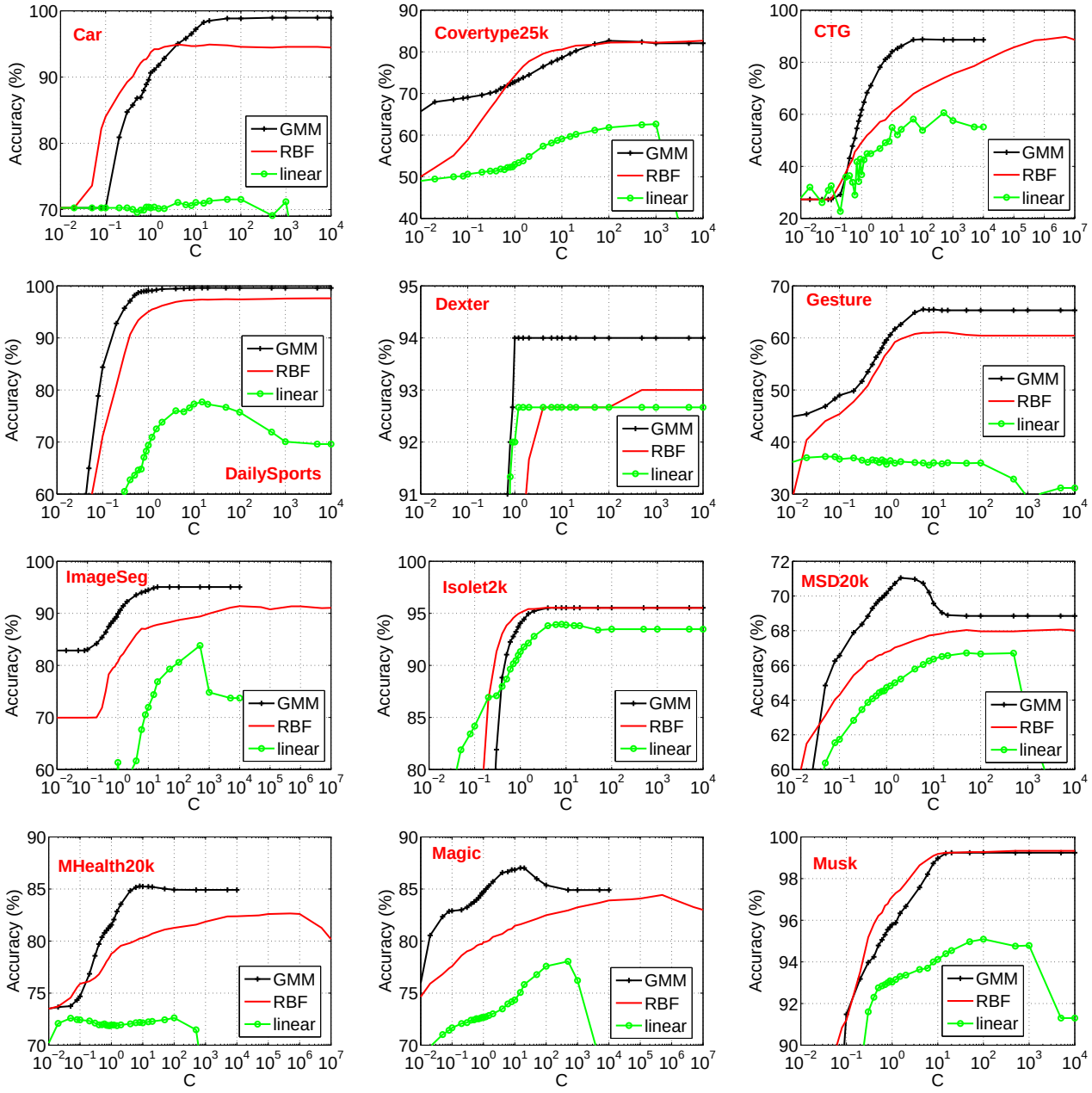

Figure 5: **Test classification accuracies using kernel SVMs**. Both the GMM kernel and RBF kernel substantially improve linear SVM. C is the l_2-regularization parameter of SVM. For the RBF kernel, we report the result at the best γ value for every C value.

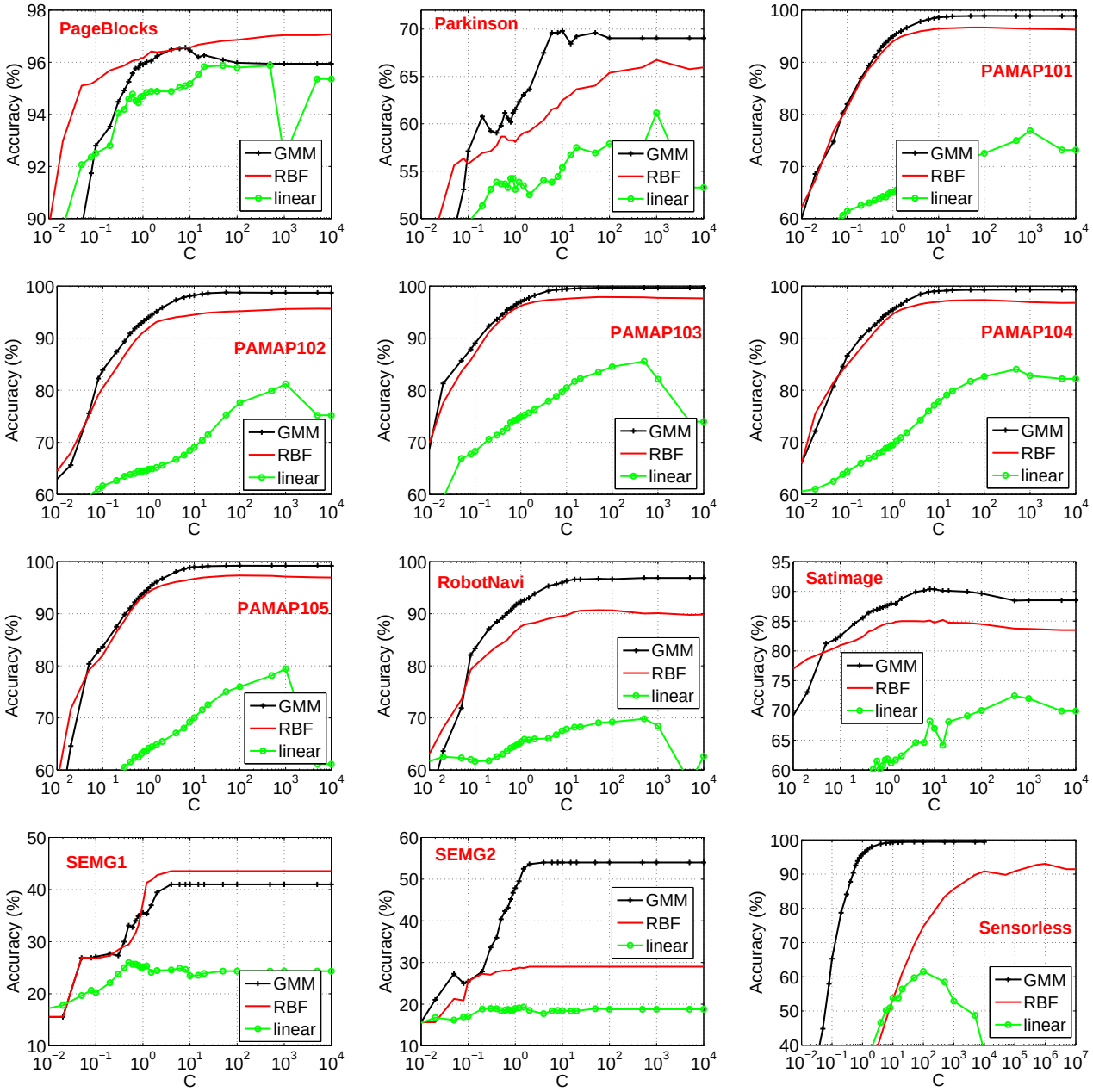

Figure 6: **Test classification accuracies using kernel SVMs**. Both the GMM kernel and RBF kernel substantially improve linear SVM. C is the l_2-regularization parameter of SVM. For the RBF kernel, we report the result at the best γ value for every C value.

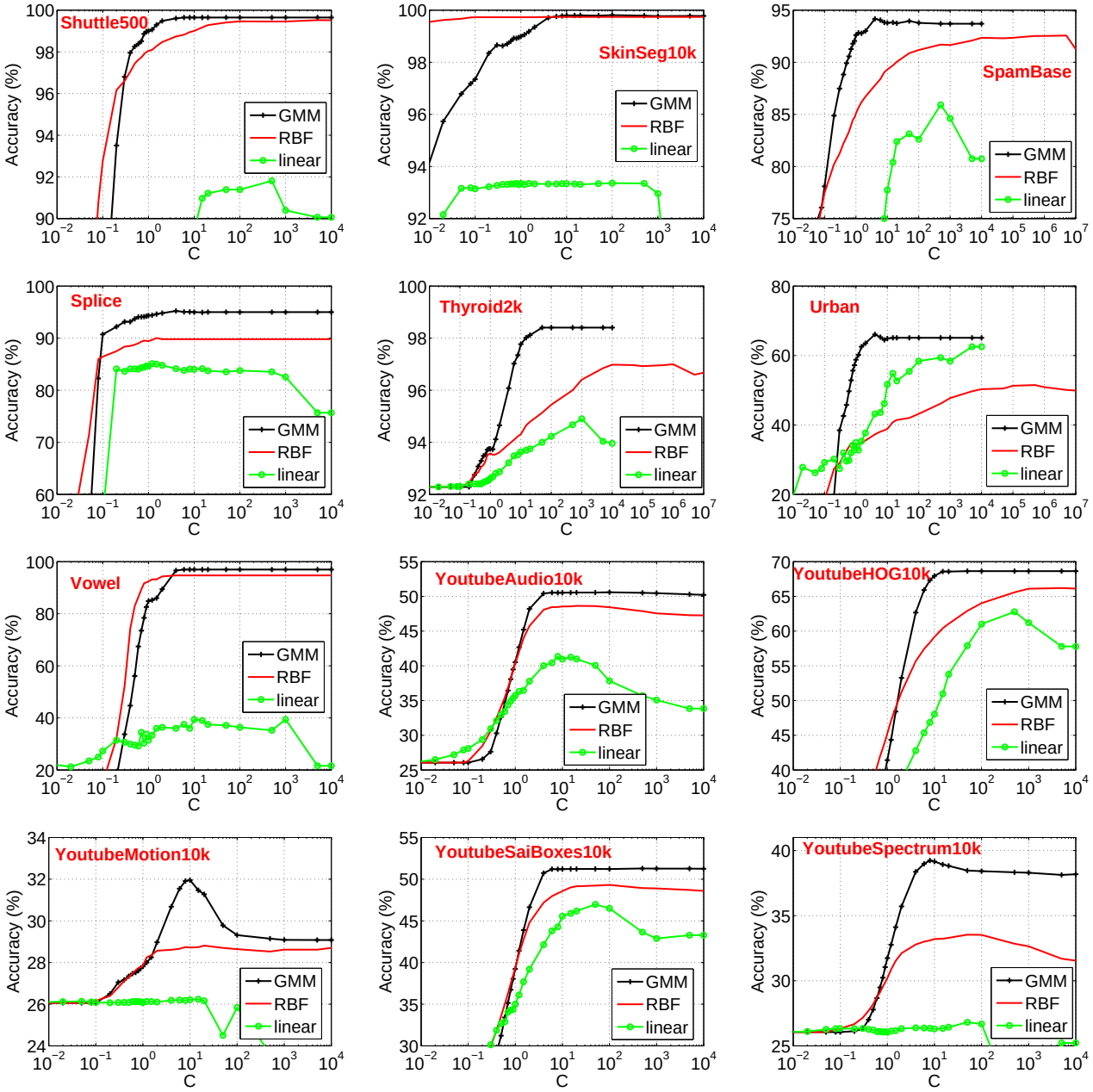

Figure 7: **Test classification accuracies using kernel SVMs**. Both the GMM kernel and RBF kernel substantially improve linear SVM. C is the l_2-regularization parameter of SVM. For the RBF kernel, we report the result at the best γ value for every C value.

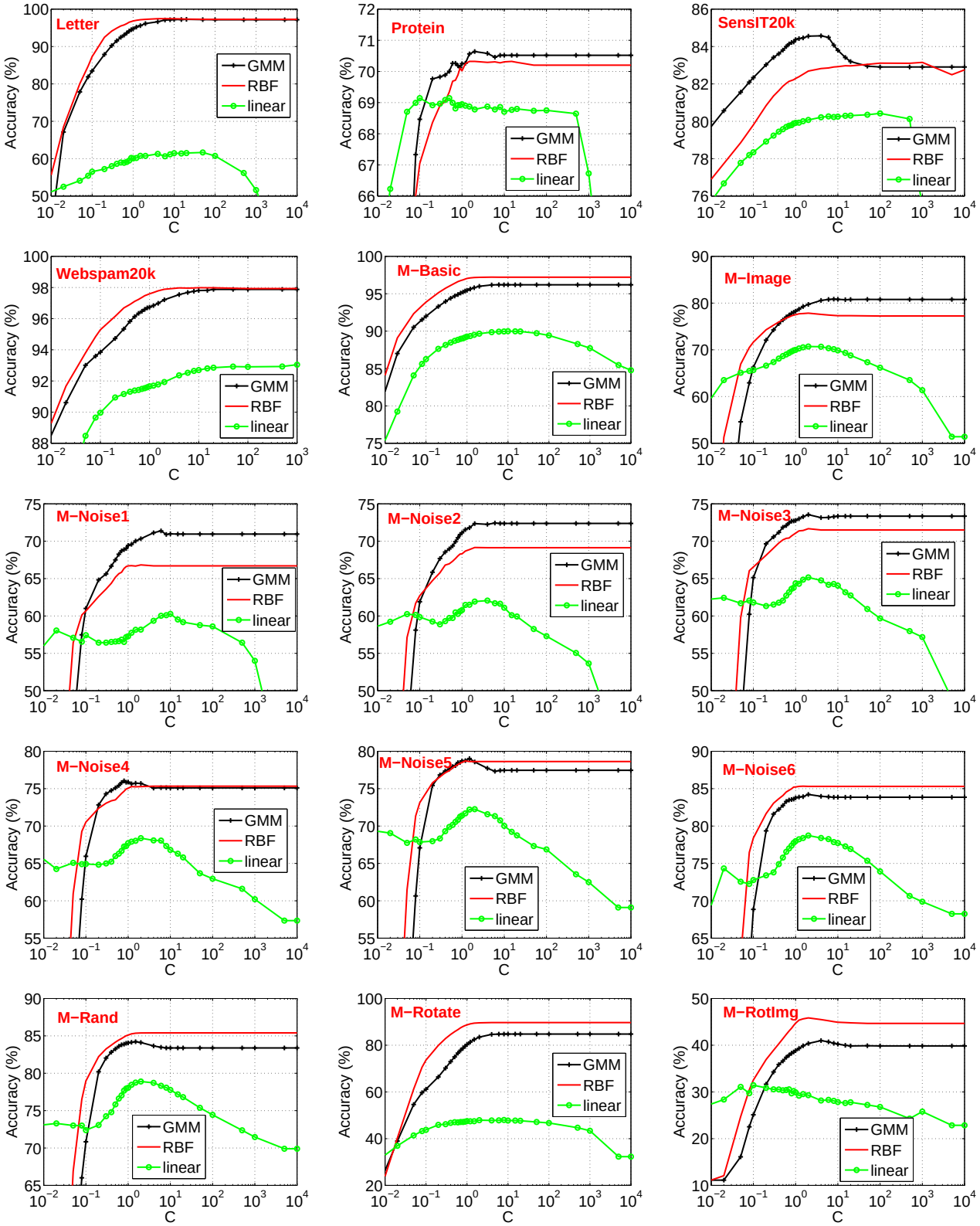

Figure 8: **Test classification accuracies using kernel SVMs**. Both the GMM kernel and RBF kernel substantially improve linear SVM. C is the l_2-regularization parameter of SVM. For the RBF kernel, we report the result at the best γ value for every C value.

For the datasets in Table 3, since [10] also conducted experiments on the RBF kernel, the polynomial kernel, and neural nets, we assembly the (error rate) results in Figure 9 and Table 4.

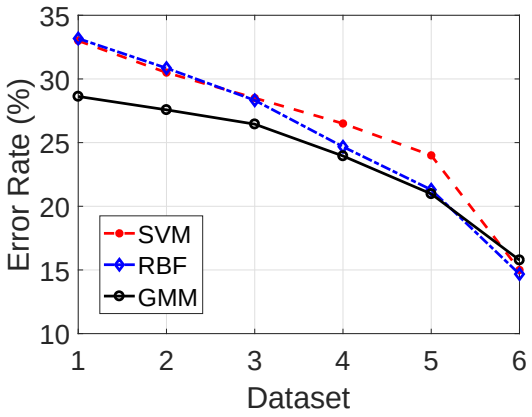

Figure 9: Error rates on 6 datasets: M-Noise1 to M-Noise6 as in Table 3. In this figure, the curve labeled as "SVM" represents the results on RBF kernel SVM conducted by [10], while the curve labeled as "RBF" presents our own experiments. The small discrepancies might be caused by the fact that we always use normalized data (i.e., ρ).

Table 4: Summary of test error rates of various algorithms on other datasets used in [10, 11]. Results in group 1 are reported by [10] for using RBF kernel, polynomial kernel, and neural nets. Results in group 2 are from our own experiments. Also, see the technical report [15] on "tunable GMM kernels" for substantially improved results, by introducing tuning parameters in the GMM kernel.

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

5 Hashing for Linearizing Nonlinear Kernels

It is known that a straightforward implementation of nonlinear kernels can be difficult for large datasets [3]. For example, for a small dataset with merely $100,000$ data points, the $100,000 \times 100,000$ kernel matrix has 10^{10} entries. In practice, being able to linearize nonlinear kernels becomes very beneficial, as that would allow us to easily apply efficient linear algorithms especially online learning [2]. Randomization (hashing) is a popular tool for kernel 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 (i.e., the normalized RFF) requires substantially more samples (i.e., larger k). Note that in our SVM experiments, we always normalize the input features to the unit l_2 norm (i.e., we will always use NRFF instead of RFF).

We will report detailed experimental results on 6 datasets. As shown in Table 5, on the first two datasets, the original RBF and GMM kernels perform similarly; in the second group, the GMM kernel noticeably outperforms the RBF kernel; in the last group, the RBF kernel noticeably outperforms the GMM kernel. We will show on all these 6 datasets, the GCWS hashing is substantially more accurate than the NRFF hashing at the same number of sample size (k). We will then present less detailed results on other datasets.

Table 5: 6 datasets used for presenting detailed experimental results on GCWS and NRFF.

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	DailySports	4560	4560	5625	77.70	97.61 (4)	**99.61**
	RobotNavi	2728	2728	24	69.83	90.69 (10)	**96.85**
3	SEMG1	900	900	3000	26.00	**43.56** (4)	41.00
	M-Rotate	12000	50000	784	47.99	**89.68** (5)	84.76

Figure 10 reports the test classification accuracies on the **Letter** dataset, for both linearized GMM kernel with GCWS and linearized RBF kernel (at the best γ) with NRFF, using LIBLINEAR. From Table 5, we can see that the original RBF kernel slightly outperforms the GMM kernel. Obviously, the results obtained by GCWS hashing are noticeably better than the results of NRFF hashing, especially when the number of samples (k) is not too large (i.e., the left panels).

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 the left panels of Figure 10, 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.

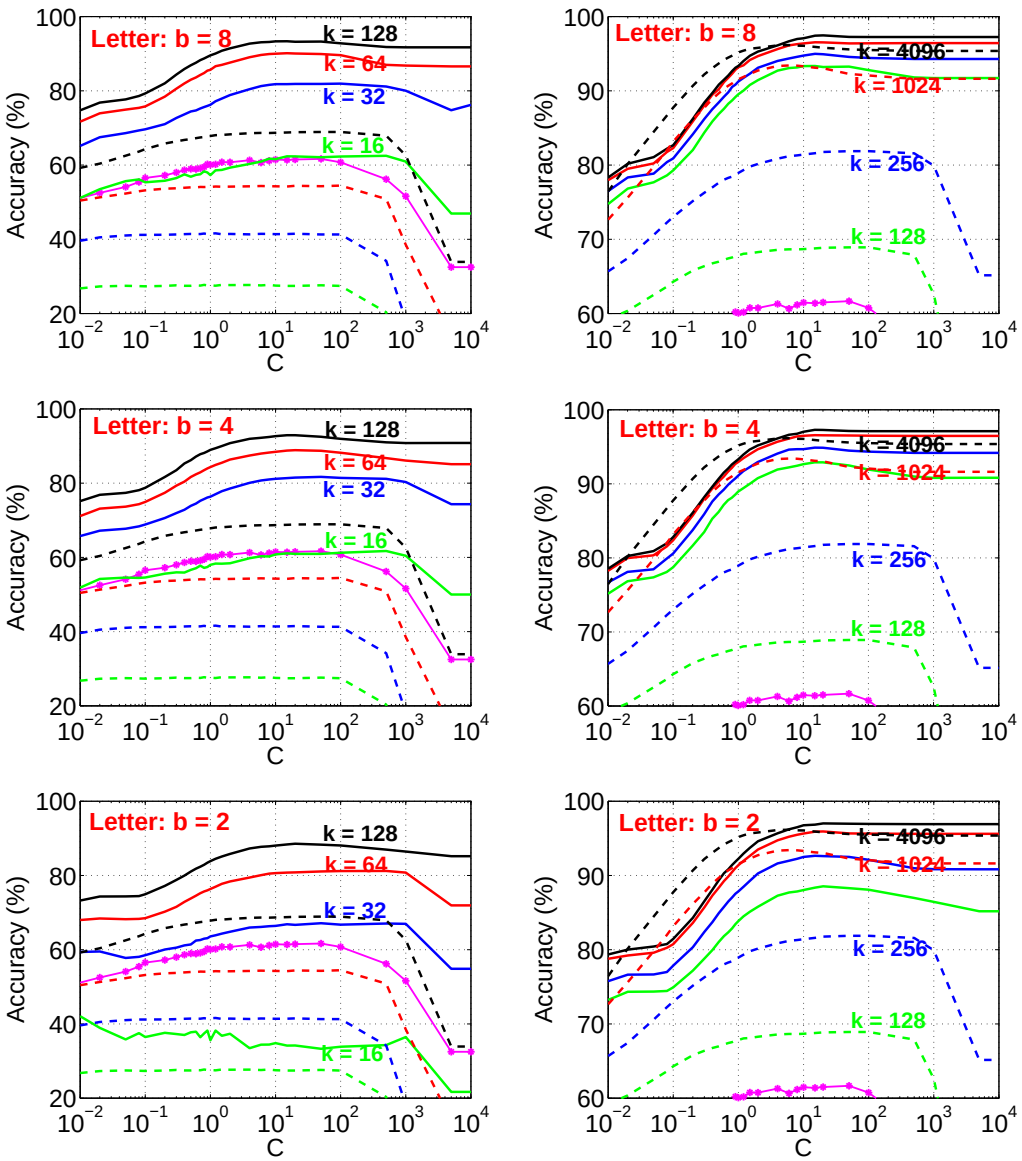

Figure 10: **Letter**: Test classification accuracies of the linearized GMM kernel (solid, GCWS) and linearized RBF kernel (dashed, NRFF), using LIBLINEAR, averaged over 10 repetitions. In each panel, we report the results on 4 different k (sample size) values: 128, 256, 1024, 4096 (right panels), and 16, 32, 64, 128 (left panels). We can see that the linearized q RBF (using NRFF) would require substantially more samples in order to reach the same accuracies as the linearized GMM kernel (using GCWS). Two interesting points: (i) Although the original (best-tuned) RBF kernel slightly outperforms the original GMM kernel, the results of GCWS are still more accurate than the results of RFF even at $k = 4096$, which is very large, considering the original data dimension is merely 16. (ii) With merely $k = 16$ samples ($b \geq 4$), GCWS already produces better results than linear SVM based on the original dataset (the solid curve marked by *).

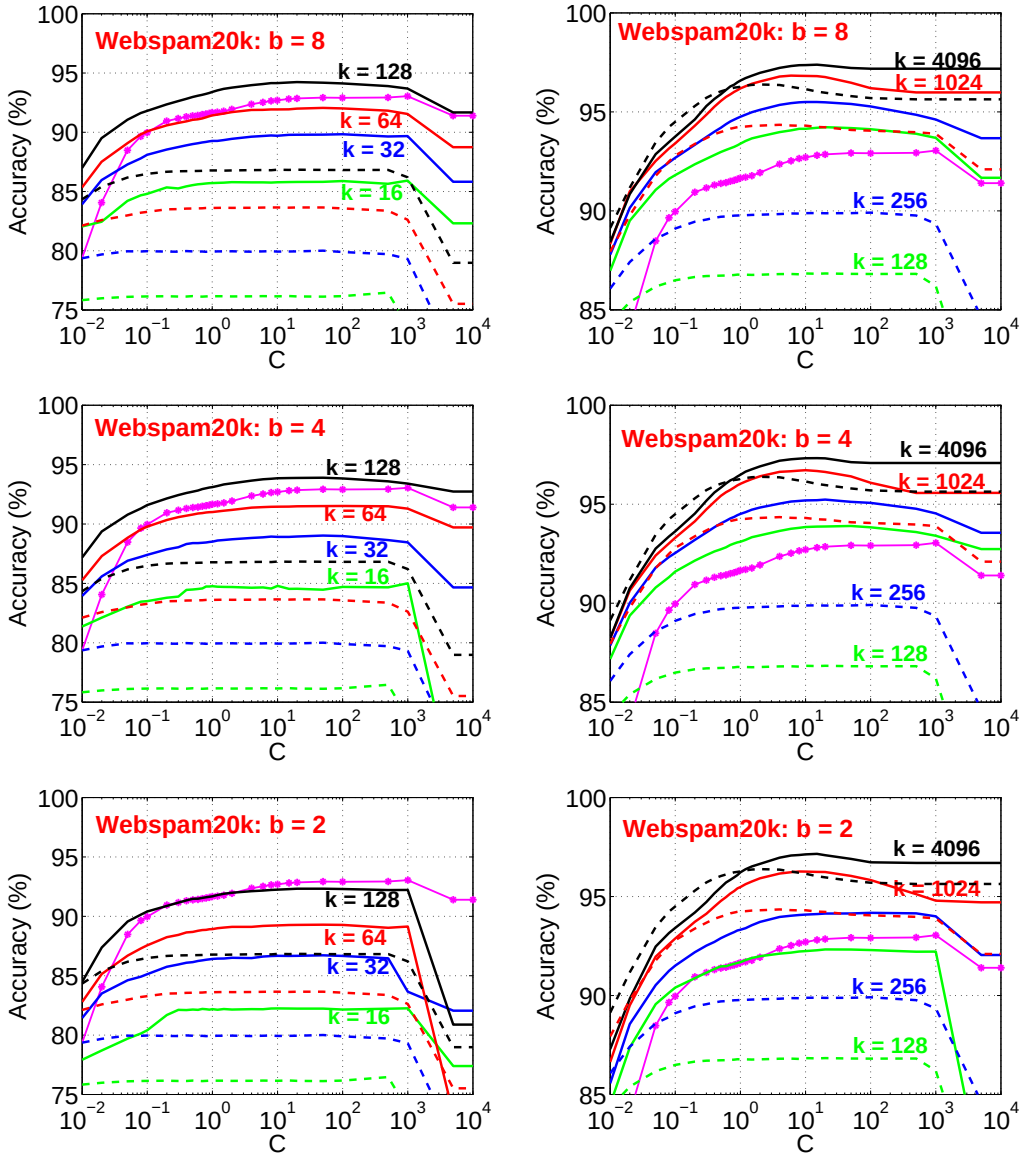

Figure 11: **Webspam20k**: Test classification accuracies of the linearized GMM kernel (solid, GCWS) and linearized RBF kernel (dashed, NRFF), using LIBLINEAR, averaged over 10 repetitions. In each panel, we report the results on 4 different k (sample size) values: 128, 256, 1024, 4096 (right panels), 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). The linear SVM results are represented by solid curves marked by *.

Figure 11 reports the test classification accuracies on the **Webspam20k** dataset. Again, the results obtained by GCWS hashing and linear classification are noticeably better than the results of NRFF hashing and linear classification, especially when the number of samples (k) is not too large (i.e., the left panels). For this dataset, the original dimension is 254. With GCWS hashing and merely $k = 128$, we can achieve higher accuracy than using linear classifier on the original data. However, with NRFF hashing, we need almost $k = 1024$ in order to outperform linear classifier on the original data. Also, note that it is sufficient to use $b = 4$ for GCWS hashing on this dataset.

Figure 12 and Figure 13 report the test classification accuracies on the **DailySports** dataset and the **RobotNavi** dataset, respectively. For both datasets, the original GMM kernel noticeably outperforms the original RBF kernel. Not surprisingly, NRFF hashing requires substantially more samples in order to reach similar accuracy as GCWS hashing, on both datasets. The results also illustrate that the parameter b (i.e., the number of bits we store for each GCWS hashed value i^*) does matter, but nevertheless, as long as $b \geq 4$, the results do not differ much.

Figure 14 and Figure 15 report 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 hashing still needs substantially more samples than GCWS hashing on both datasets. Again, for GCWS, the results do not differ much once we use $b \geq 4$. These results again confirm the advantage of GCWS hashing.

Figure 16 reports the test classification accuracies on more datasets, only for $b = 8$ and $k \geq 128$. Figure 17 presents the hashing results on 6 larger datasets for which we can not directly train kernel SVMs. We report only for $b = 8$ and k up to 1204. All these results confirm that linearization via GCWS works well for the GMM kernel. In contrast, the normalized random Fourier feature (NRFF) approach typically requires substantially more samples (i.e., much larger k). This phenomenon can be largely explained by the theoretical results in Theorem 3 and Theorem 4, which conclude that GCWS hashing is more (considerably) accurate than NRFF hashing, unless the similarity is high. At high similarity, the variances of both hashing methods become very small.

We should mention that the original (tuning-free) GMM kernel can be modified by introducing tuning parameters. The original GCWS algorithm can be slightly modified to linearize the new (and tunable) GMM kernel. As shown in [15], on many datasets, the tunable GMM kernel can be a strong competitor compared to computationally expensive algorithms such as deep nets or trees.

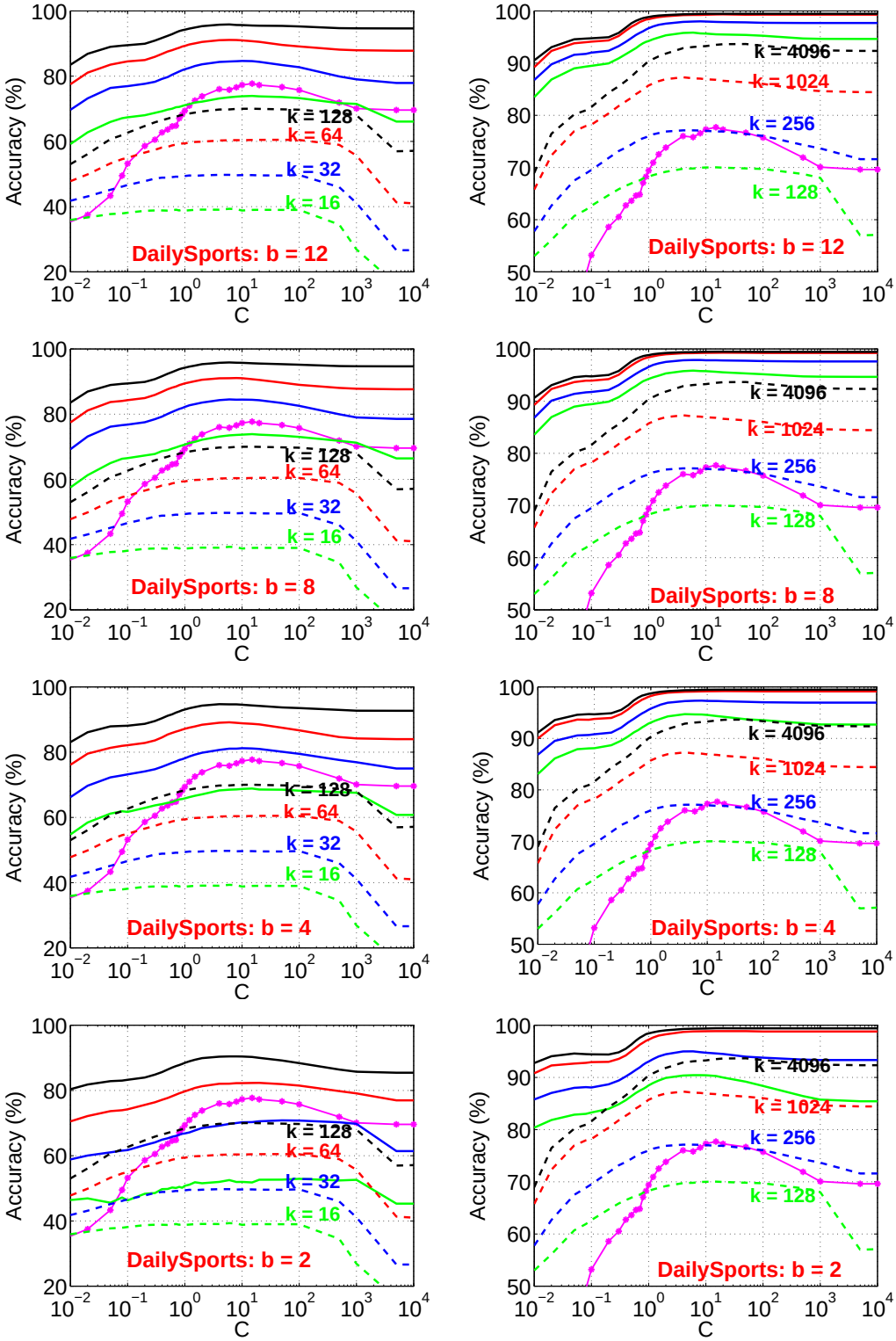

Figure 12: **DailySports**: Test classification accuracies of the linearized GMM kernel (solid) and linearized RBF kernel (dashed) , using LIBLINEAR. In each panel, we report the results on 4 different k (sample size) values: 128, 256, 1024, 4096 (right panels), and 16, 32, 64, 128 (left panels). We 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).

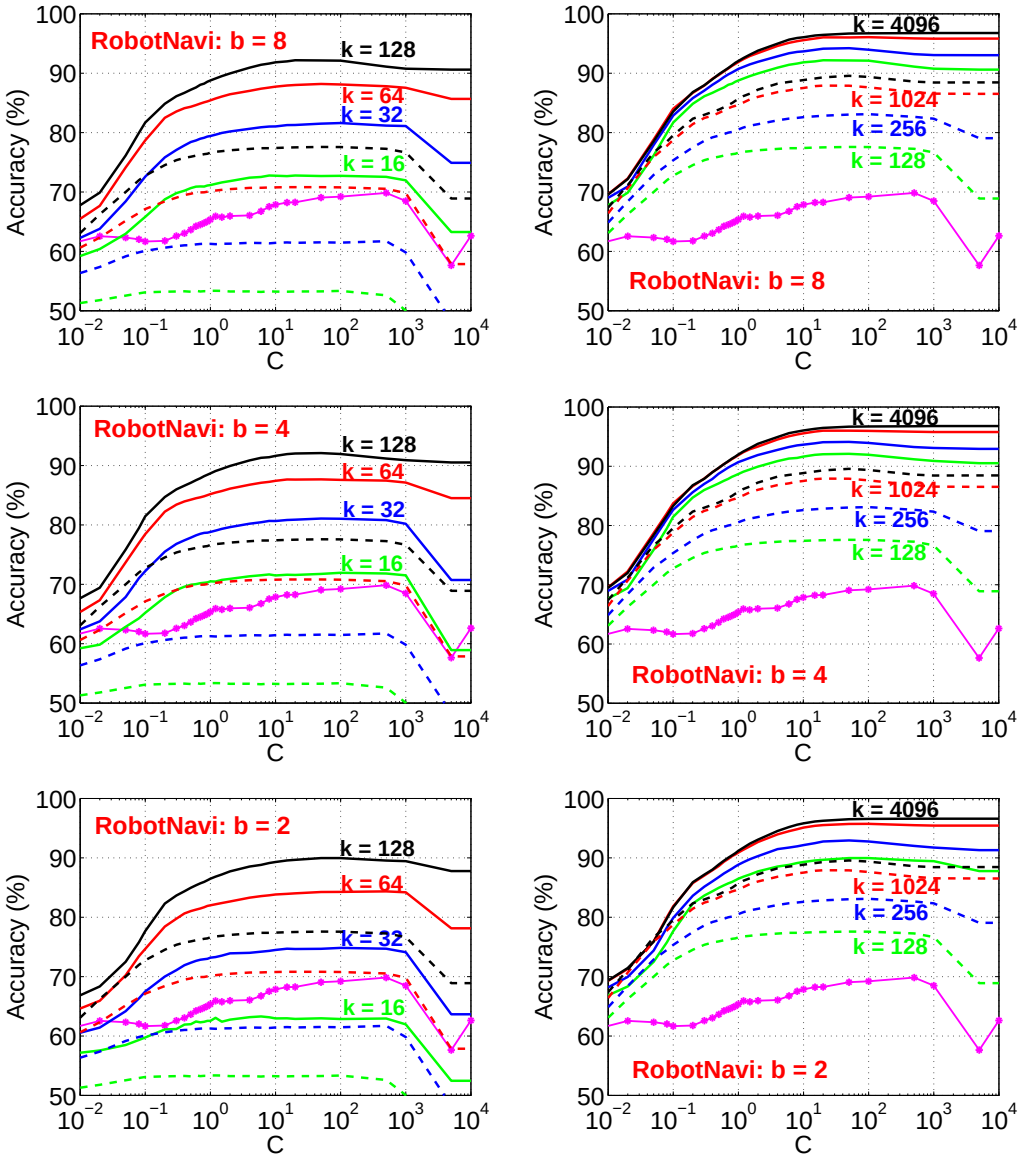

Figure 13: **RobotNavi**: Test classification accuracies of the linearized GMM kernel (solid) and linearized RBF kernel (dashed), using LIBLINEAR. In each panel, we report the results on 4 different k (sample size) values: 128, 256, 1024, 4096 (right panels), and 16, 32, 64, 128 (left panels). We can 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).

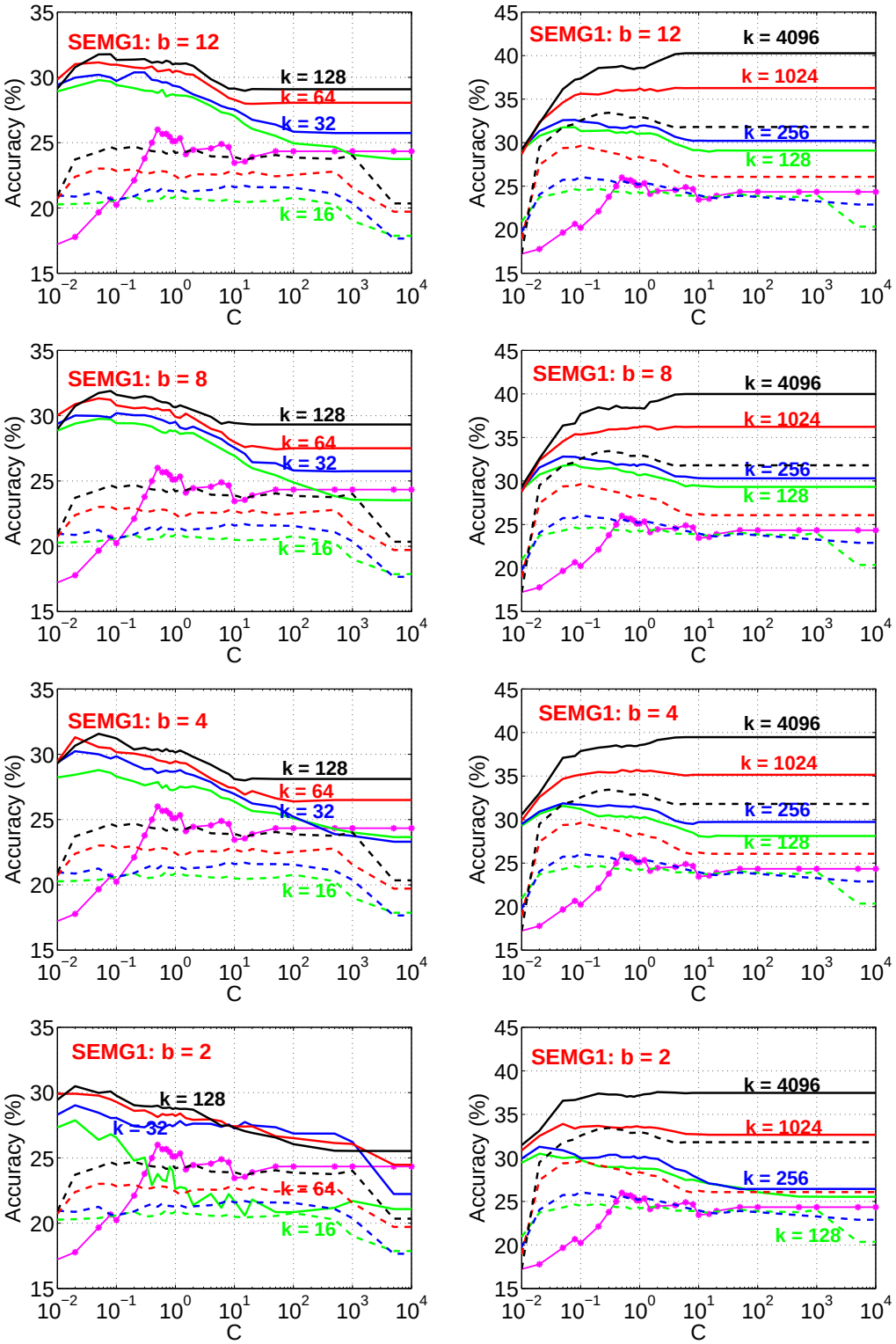

Figure 14: **SEMG1**: Test classification accuracies of the linearized GMM kernel (solid) and linearized RBF kernel (dashed), using LIBLINEAR. Again, we can see that the linearized RBF would require substantially more samples in order to reach the same accuracies as the linearized GMM kernel. Note that, for this dataset, the original RBF kernel actually outperforms the original GMM kernel as shown in Table 1.

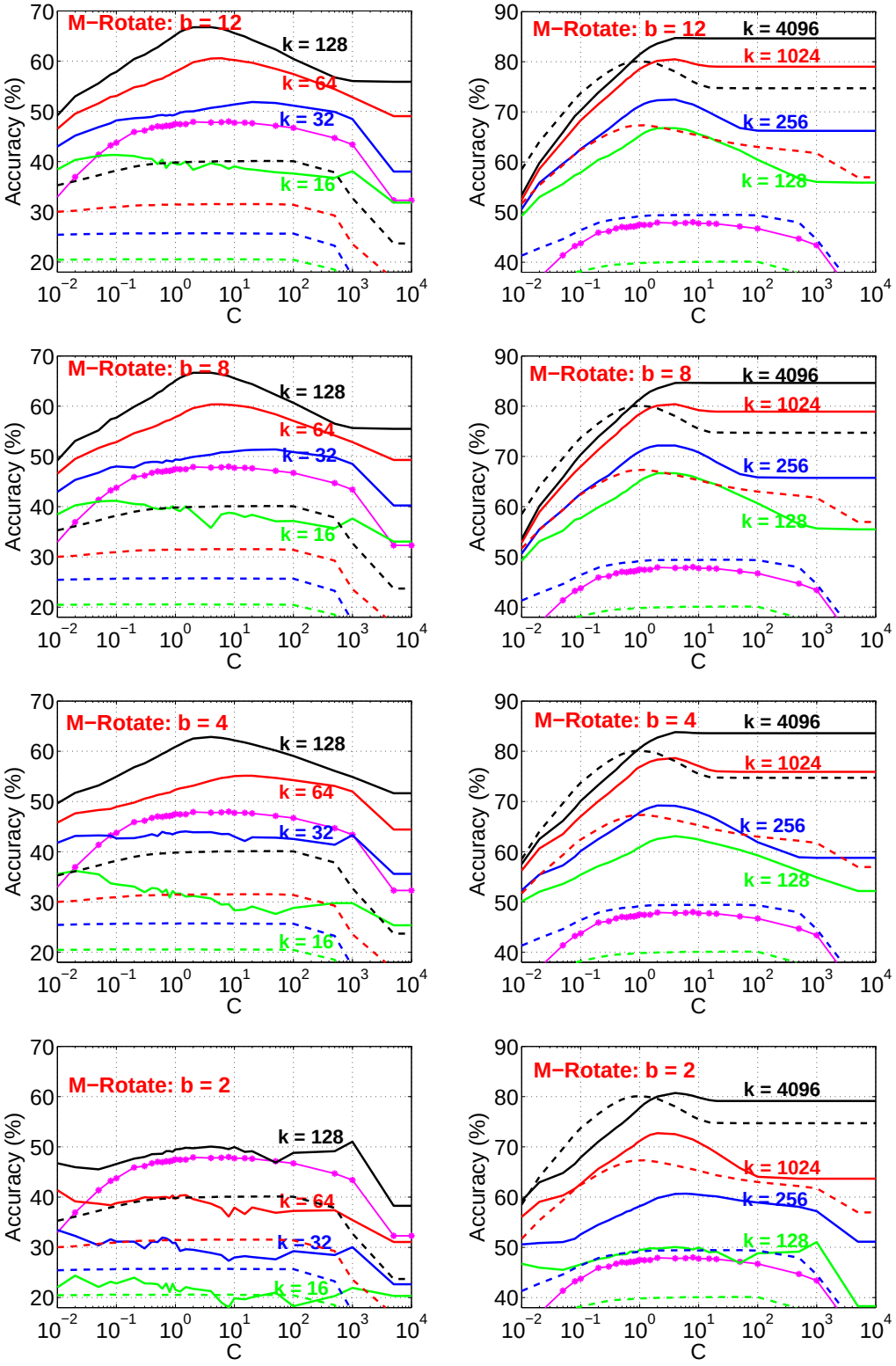

Figure 15: **M-Rotate**: Test classification accuracies of the linearized GMM kernel (solid) and linearized RBF kernel (dashed) , using LIBLINEAR. Again, we can see that the linearized RBF would require substantially more samples in order to reach the same accuracies as the linearized GMM kernel. For M-Rotate, the original RBF kernel actually outperforms the original GMM kernel as shown in Table 3.

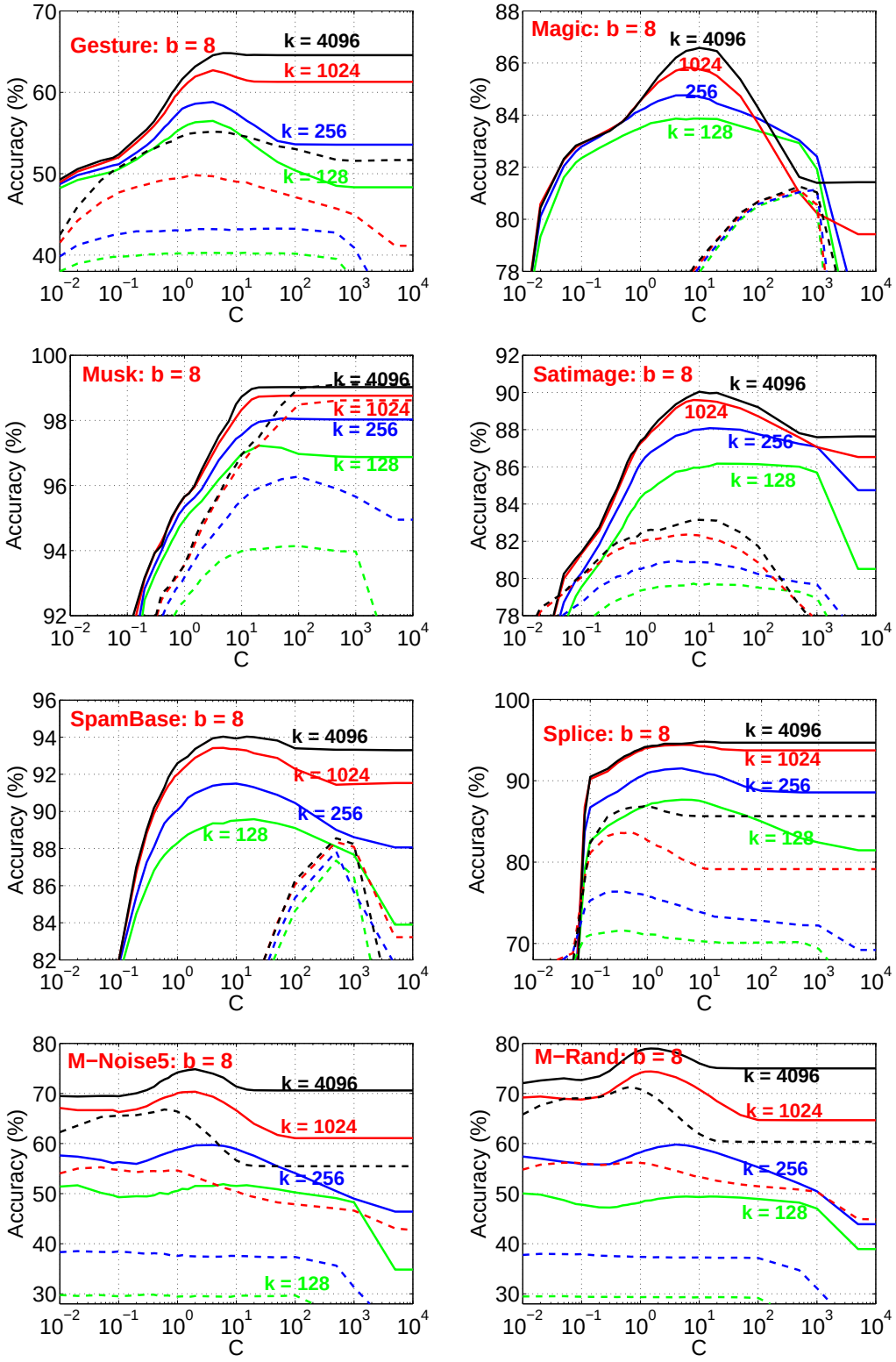

Figure 16: **More Datasets**: Test classification accuracies of the linearized GMM kernel (solid) and linearized RBF kernel (dashed) , using LIBLINEAR. Typically, the linearized RBF would require substantially more samples in order to reach the same accuracies as the linearized GMM kernel.

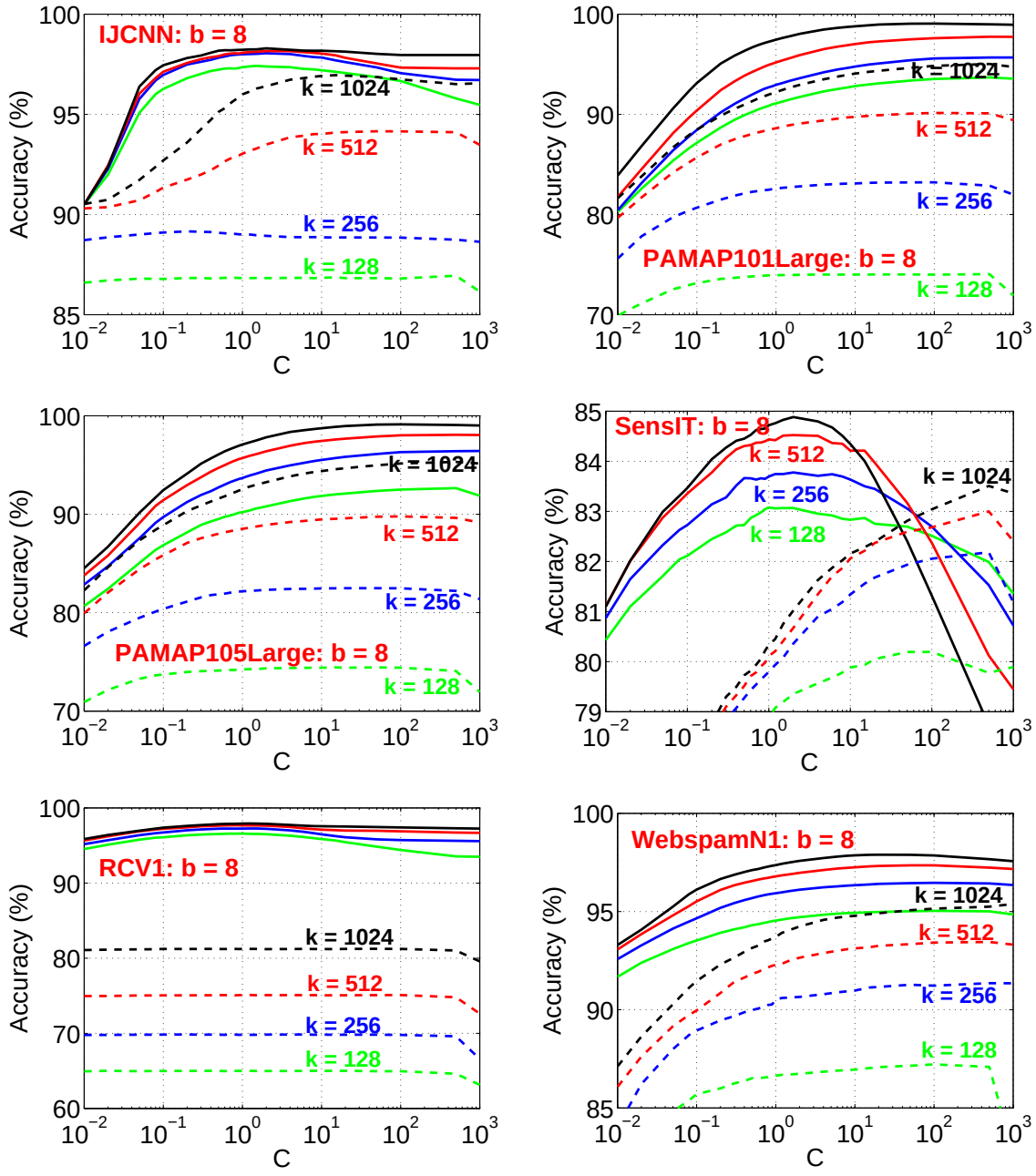

Figure 17: **Larger Datasets**: Test classification accuracies of the linearized GMM kernel with GCWS (solid) and linearized RBF kernel with NRFF (dashed), using LIBLINEAR, on 6 larger datasets which we can not directly compute kernel SVM classifiers. The experiments again confirm that GCWS hashing is substantially more accurate than NRFF hashing at the same sample size k.

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 (such as LIBINEAR), a larger b will increase the training time but not much. See Figure 18 for an example, which actually shows that NRFF will consume more time at high C (for achieving a good accuracy). Note that, with online learning, it would be more obvious that the training time is determined by the number of nonzeros and number of epoches. For industrial practice, typically only one epoch or a few epoches are used.

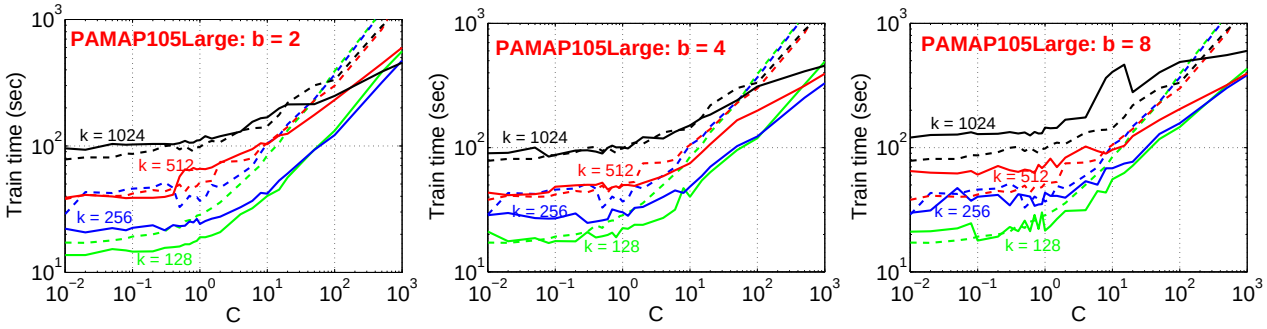

Figure 18: **PAMAP105Large**: Training times of GCWS (solid curves) and NRFF (dashed curves), for four sample sizes $k \in \{128, 256, 512, 1024\}$, and $b \in \{2, 4, 8\}$.

Data storage: For GCWS, the storage cost per data vector is $b \times k$ while the cost for NRFF would be $k\times$ number of bits per hashed value (which might be a large value such as 32). Therefore, at the same sample size k, GCWS will likely need less space to store the hashed data than NRFF.

6 Conclusion

Large-scale machine learning has become increasingly important in practice. For industrial applications, it is often the case that only linear methods are affordable. It is thus practically beneficial to have methods which can provide substantially more accurate prediction results than linear methods, with no essential increase of the computation 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 vision, and beyond, e.g., [21, 27, 1, 7, 5, 28, 8, 25, 4, 23]. In this paper, we rigorously prove that a simple normalization step (i.e., NRFF) can substantially improve the original RFF procedure by reducing the estimation variance.

In this paper, we also propose 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 to reach the same accuracy as the normalized random Fourier feature (NRFF) method. This is practically important, because the training (and testing) cost and storage cost are determined by the number of nonzeros (which is the number of samples in NRFF or GCWS) per data vector of the dataset. The superb empirical performance of GCWS can be largely explained by our theoretical analysis that the estimation variance of GCWS is typically much smaller than the variance of NRFF (even though NRFF has improved the original RFF).

By incorporating tuning parameters, [15] demonstrated that the performance of the GMM kernel and GCWS hashing can be further improved, in some datasets remarkably so. See [15] for the comparisons with deep nets and trees. Lastly, we should also mention that GCWS can be naturally applied in the context of efficient near neighbor search, due to the discrete nature of the samples, while NRFF or samples from Nystrom method can not be directly used for building hash tables.

References

[1] R. H. Affandi, E. Fox, and B. Taskar. Approximate inference in continuous determinantal processes. In *NIPS*, pages 1430–1438. 2013.

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

[3] L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors. *Large-Scale Kernel Machines*. The MIT Press, Cambridge, MA, 2007.

[4] K. P. Chwialkowski, A. Ramdas, D. Sejdinovic, and A. Gretton. Fast two-sample testing with analytic representations of probability measures. In *NIPS*, pages 1981–1989. 2015.

[5] B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M.-F. F. Balcan, and L. Song. Scalable kernel methods via doubly stochastic gradients. In *NIPS*, pages 3041–3049. 2014.

[6] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[7] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *NIPS*, pages 918–926. 2014.

[8] C.-J. Hsieh, S. Si, and I. S. Dhillon. Fast prediction for large-scale kernel machines. In *NIPS*, pages 3689–3697. 2014.

[9] S. Ioffe. Improved consistent sampling, weighted minhash and L1 sketching. In *ICDM*, pages 246–255, Sydney, AU, 2010.

[10] H. Larochelle, D. Erhan, A. C. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*, pages 473–480, Corvalis, Oregon, 2007.

[11] P. Li. Robust logitboost and adaptive base class (abc) logitboost. In *UAI*, 2010.

[12] P. Li. 0-bit consistent weighted sampling. In *KDD*, Sydney, Australia, 2015.

[13] P. Li. Nystrom method for approximating the gmm kernel. Technical report, arXiv:1605.05721, 2016.

[14] P. Li. Generalized intersection kernel. Technical report, arXiv:1612.09283, 2017.

[15] P. Li. Tunable gmm kernels. Technical report, arXiv:1701.02046, 2017.

[16] P. Li, T. J. Hastie, and K. W. Church. Improving random projections using marginal information. In *COLT*, pages 635–649, Pittsburgh, PA, 2006.

[17] P. Li, T. J. Hastie, and K. W. Church. Very sparse random projections. In *KDD*, pages 287–296, Philadelphia, PA, 2006.

[18] P. Li, A. Shrivastava, J. Moore, and A. C. König. Hashing algorithms for large-scale learning. In *NIPS*, Granada, Spain, 2011.

[19] P. Li and C.-H. Zhang. Theory of the gmm kernel. Technical report, arXiv:1608.00550, 2016.

[20] M. Manasse, F. McSherry, and K. Talwar. Consistent weighted sampling. Technical Report MSR-TR-2010-73, Microsoft Research, 2010.

[21] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, pages 1509–1517. 2009.

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

[23] E. Richard, G. A. Goetz, and E. J. Chichilnisky. Recognizing retinal ganglion cells in the dark. In *NIPS*, pages 2476–2484. 2015.

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

[25] A. Shah and Z. Ghahramani. Parallel predictive entropy search for batch global optimization of expensive objective functions. In *NIPS*, pages 3330–3338. 2015.

[26] D. J. Sutherland and J. Schneider. On the error of random fourier features. In *UAI*, Amsterdam, The Netherlands, 2015.

[27] T. Yang, Y.-f. Li, M. Mahdavi, R. Jin, and Z.-H. Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. In *NIPS*, pages 476–484. 2012.

[28] I. E.-H. Yen, T.-W. Lin, S.-D. Lin, P. K. Ravikumar, and I. S. Dhillon. Sparse random feature algorithm as coordinate descent in hilbert space. In *NIPS*, pages 2456–2464. 2014.