Fast and Scalable Polynomial Kernels via Explicit Feature Maps *

Ninh Pham
IT University of Copenhagen
Copenhagen, Denmark
ndap@itu.dk

Rasmus Pagh
IT University of Copenhagen
Copenhagen, Denmark
pagh@itu.dk

ABSTRACT

Approximation of non-linear kernels using random feature mapping has been successfully employed in large-scale data analysis applications, accelerating the training of kernel machines. While previous random feature mappings run in $O(ndD)$ time for n training samples in d-dimensional space and D random feature maps, we propose a novel randomized tensor product technique, called *Tensor Sketching*, for approximating any polynomial kernel in $O(n(d + D \log D))$ time. Also, we introduce both *absolute* and *relative* error bounds for our approximation to guarantee the reliability of our estimation algorithm. Empirically, Tensor Sketching achieves higher accuracy and often runs orders of magnitude faster than the state-of-the-art approach for large-scale real-world datasets.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*

General Terms

Algorithms, Performance, Experimentation

Keywords

polynomial kernel; SVM; tensor product; Count Sketch; FFT

1. INTRODUCTION

Kernel machines such as Support Vector Machines (SVMs) have recently emerged as powerful approaches for many machine learning and data mining tasks. One of the key properties of kernel methods is the capability to efficiently find non-linear structure of data by the use of kernels. A kernel can be viewed as an *implicit* non-linear data mapping from original data space into high-dimensional feature space, where each coordinate corresponds to one feature of the data points. In that space, one can perform well-known data analysis algorithms without ever interacting with the coordinates of the data, but rather by simply computing their pairwise inner products. This operation can not only avoid the cost of explicit computation of the coordinates in feature space but also handle general types of data (such as numeric data, symbolic data).

While kernel methods have been used successfully in a variety of data analysis tasks, their scalability is a bottleneck. Kernel-based learning algorithms usually scale poorly with the number of the training samples (a cubic running time and quadratic storage for direct methods). This drawback is becoming more crucial with the rise of big data applications [12, 3]. Recently, Joachims [9] proposed an efficient training algorithm for *linear* SVMs that runs in time linear in the number of training examples. Since one can view *non-linear* SVMs as *linear* SVMs operating in an appropriate feature space, Rahimi and Recht [16] first proposed a random feature mapping to approximate shift-invariant kernels in order to combine the advantages of both linear and non-linear SVM approaches. This approach approximates kernels by an *explicit* data mapping into relatively low-dimensional random feature space. In this random feature space, the kernel of any two points is well approximated by their inner product. Therefore, one can apply existing fast linear learning algorithms to find data relations corresponding to non-linear kernel methods in the random feature space. That leads to a substantial reduction in training time while obtaining similar testing error.

Following up this line of work, many randomized approaches to approximate kernels are proposed for accelerating the training of kernel machines [10, 12, 20, 21]. While the training algorithm is linear, existing kernel approximation mappings require time proportional to the product of the number of dimensions d and the number of random features D. This means that the mapping itself is a bottleneck whenever dD is not small. In this paper we address this bottleneck, and present a near-linear time mapping for approximating any polynomial kernel.

Particularly, given any two points of a dataset S of n points, $x = \{x_1, \cdots, x_d\}, y = \{y_1, \cdots, y_d\} \in S \subset \mathbb{R}^d$ and an implicit *feature space* mapping $\phi : \mathbb{R}^d \mapsto \mathcal{F}$, the inner product between these points in the feature space $\mathcal{F}$ can be quickly computed as $\langle \phi(x), \phi(y) \rangle = \kappa(x, y)$ where $\kappa()$ is an easily computable kernel. An *explicit* random feature mapping $f : \mathbb{R}^d \mapsto \mathbb{R}^D$ can efficiently approximate a kernel

*This work is supported by the Danish National Research Foundation under the Sapere Aude program.

KDD'13, August 11–14, 2013, Chicago, Illinois, USA.

$\kappa()$ if it satisfies:

$$\mathbf{E}\left[\langle f(x), f(y)\rangle\right] = \langle \phi(x), \phi(y)\rangle = \kappa(x, y) \ .$$

So we can transform data from the original data space into a low-dimensional explicit random feature space and use any linear learning algorithm to find non-linear data relations.

Rahimi and Recht [16] introduced a random projection-based algorithm to approximate shift-invariant kernels (e.g. the Gaussian kernel $\kappa(x, y) = \exp(-\|x - y\|^2 / 2\sigma^2)$, for $\sigma > 0$). Vempati et al. [21] extended this work to approximate generalized radial-basic function (RBF) kernels (e.g. the exponential-χ^2 kernel $\kappa(x, y) = \exp(-\chi^2(x, y)/2\sigma^2)$, where $\sigma > 0$ and χ^2 is the Chi-squared distance measure). Recently, Kar and Karnick [10] made use of the Maclaurin series expansion to approximate inner product kernels (e.g. the polynomial kernel $\kappa(x, y) = (\langle x, y\rangle + c)^p$, for $c \geq 0$ and an integer p).

These approaches have to maintain D random vectors $\omega_1, \cdots \omega_D \in \mathbb{R}^d$ in $O(dD)$ space and need $O(ndD)$ operations for computing D random feature maps. That incurs significant (quadratic) computational and storage costs when $D = O(d)$ and d is rather large. When the decision boundary of the problem is rather smooth, the computational cost of random mapping might dominate the training cost. In addition, the *absolute* error bounds of previous approaches are not tight. Particularly, the Maclaurin expansion based approach [10] suffers from large error because it approximates the homogeneous polynomial kernel $\kappa(x, y) = \langle x, y\rangle^p$ by $\prod_{i=1}^p \langle \omega_i, x\rangle \prod_{i=1}^p \langle \omega_i, y\rangle$ where $\omega_i \in \{+1, -1\}^d$. Our experiments show that large estimation error results in either accuracy degradation or negligible reduction in training time.

In this work, we consider the problem of approximating the commonly used polynomial kernel $\kappa(x, y) = (\langle x, y\rangle + c)^p$ to accelerate the training of kernel machines. We develop a fast and scalable randomized tensor product technique, named *Tensor Sketching*, to estimate the polynomial kernel of any pair of points of the dataset. Our proposed approach works in $O(np(d + D \log D))$ time and requires $O(pd \log D)$ space for random vectors. The main technical insight is the connection between tensor product and fast convolution of Count Sketches [2, 14], which enables us to reduce the computational complexity and space usage. We introduce both *absolute* and *relative* error bounds for our approximation to guarantee the reliability of our estimation algorithm. The empirical experiments on real-world datasets demonstrate that Tensor Sketching achieves higher accuracy and often runs orders of magnitude faster than the state-of-the-art approach for large-scale datasets.

The organization of the paper is as follows. In Section 2, we briefly review related work. Section 3 describes background and preliminaries. The proposed approach is presented and analyzed in Section 4 and 5. In Section 6, we show experimental evaluations of our proposed approach on real world datasets. Section 7 concludes the paper.

2. RELATED WORK

Traditional approaches for solving non-linear SVMs on large datasets are *decomposition methods* [1, 13]. These methods divide the training set into two sets, named working set and fixed set; and iteratively solve the optimization problem with respect to the working set while *freezing* the fixed set. In other words, they iteratively update a subset of kernel methods' coefficients by performing coordinate ascent on subsets of the training set until KKT conditions have been satisfied to within a certain tolerance. Although such approaches can handle the memory restrictions involving the dense kernel matrix, they still involve numerical solutions of optimization subproblems and therefore can be problematic and expensive to large-scale datasets.

In order to apply kernel methods to large-scale datasets, many approaches have been proposed for quickly approximating the kernel matrix, including the Nyström methods [5, 23], sparse greedy approximation [19] and low-rank kernel approximation [7]. These approximation schemes can reduce the computational and storage costs of operating on a kernel matrix while preserving the quality of results. An assumption of these approaches is that the kernel matrix has many zero eigenvalues. This might not be true in many datasets. Furthermore, there is a lack of experiments to illustrate the efficiency of these approaches on large-scale datasets [16].

Instead of approximating the kernel matrix, recent approaches [10, 12, 16, 20, 21, 24] approximate the kernels by explicitly mapping data into a relatively low-dimensional random feature space. The explicit mapping transforms data into a random feature space where the pairwise inner products of transformed data points are approximately equal to kernels in feature space. Therefore, we can apply existing fast linear learning algorithms [6, 9, 18] to find non-linear data relations in that random feature space. While previous such approaches can efficiently accelerate the training of kernel machines, they incur significant computational cost (quadratic in the dimensionality of data). That results in performance degradation on large-scale high-dimensional datasets.

3. BACKGROUND AND PRELIMINARIES

3.1 Count Sketch

Charikar et al. [2] described and analyzed a sketching approach, called Count Sketch, to estimate the frequency of all items in a stream. Recently, the machine learning community has used Count Sketch as a feature hashing technique for large-scale multitask learning [22] because Count Sketches can preserve the pairwise inner products within an arbitrarily small factor. In our work, we view Count Sketch as a specific random projection technique in high-dimensional space because it maintains linear projections of a vector with the number of random vectors defined *implicitly* by simple independent hash functions.

DEFINITION 1. *Given two 2-wise independent hash functions $h : [d] \mapsto [k]$ and $s : [d] \mapsto \{+1, -1\}$. Count Sketch of a point $x = \{x_1, \cdots, x_d\} \in \mathbb{R}^d$ is denoted by $Cx = \{(Cx)_1, \cdots, (Cx)_k\} \in \mathbb{R}^k$ where $(Cx)_j = \sum_{i:h(i)=j} s(i)x_i$.*

Note that the two Count Sketches $C^{(1)}x, C^{(2)}x$ of a point x are different if they use different hash functions $h_1 \neq h_2$ and $s_1 \neq s_2$. The following lemma provides the bias and variance of the pairwise inner product of Count Sketches.

LEMMA 2. *Given two points $x, y \in \mathbb{R}^d$, we denote by $Cx, Cy \in \mathbb{R}^k$ the respective Count Sketches of x, y on the*

same hash functions h, s.

$$\mathbf{E}[\langle Cx, Cy\rangle] = \langle x, y\rangle ,$$

$$\mathbf{Var}[\langle Cx, Cy\rangle] = \frac{1}{k}\left(\sum_{i\neq j} x_i^2 y_j^2 + \sum_{i\neq j} x_i y_i x_j y_j\right) .$$

PROOF. See [22, Appendix A]. □

We derive an upper bound of variance of any pairwise inner product of Count Sketches as follows:

LEMMA 3. *Given two points $x, y \in \mathbb{R}^d$, we denote by $Cx, Cy \in \mathbb{R}^k$ the respective Count Sketches of x, y on the same hash functions h, s.*

$$\mathbf{Var}[\langle Cx, Cy\rangle] \leq \frac{1}{k}\left(\langle x, y\rangle^2 + \|x\|^2 \|y\|^2\right).$$

PROOF. Given any two points $x = \{x_1, \cdots, x_d\}, y = \{y_1, \cdots, y_d\}$, we have:

$$\|x\|^2\|y\|^2 = \sum_i x_i^2 y_i^2 + \sum_{i\neq j} x_i^2 y_j^2 \ ,$$

$$\langle x, y\rangle^2 = \sum_i x_i^2 y_i^2 + \sum_{i\neq j} x_i y_i x_j y_j \ .$$

By the lemma 2, we have:

$$\begin{aligned}\mathbf{Var}[\langle Cx, Cy\rangle] &= \frac{1}{k}\left(\sum_{i\neq j} x_i^2 y_j^2 + \sum_{i\neq j} x_i y_i x_j y_j\right) \\ &= \frac{1}{k}\left(\langle x, y\rangle^2 + \|x\|^2\|y\|^2\right) - \frac{2}{k}\sum_i x_i^2 y_i^2 \\ &\leq \frac{1}{k}\left(\langle x, y\rangle^2 + \|x\|^2\|y\|^2\right).\end{aligned}$$

□

It is worth noting that Count Sketch might not distort a sparse vector. This is due to the fact that non-zero elements will always be hashed into a cell of Count Sketch. In other words, they are retained after sketching with high probability. In addition, Count Sketch requires $O(nd)$ operations for n points in d-dimensional space. Therefore, Count Sketch might provide better performance than traditional random projections in applications dealing with sparse vectors.

3.2 Tensor product

Given a vector $x = \{x_1, \cdots, x_d\} \in \mathbb{R}^d$, the 2-level tensor product or outer product $x^{(2)} = x \otimes x$ is defined as follows:

$$x^{(2)} = x \otimes x = \begin{bmatrix} x_1x_1 & x_1x_2 & \cdots & x_1x_d \\ x_2x_1 & x_2x_2 & \cdots & x_2x_d \\ \vdots & \vdots & \ddots & \vdots \\ x_dx_1 & x_dx_2 & \cdots & x_dx_d \end{bmatrix} \in \mathbb{R}^{d^2} \ .$$

Given an integer p, we consider a p-level tensor product $\Omega^p : \mathbb{R}^d \mapsto \mathbb{R}^{d^p}$ given by

$$x \mapsto x^{(p)} = x \underbrace{\otimes \cdots \otimes}_{p \text{ times}} x \ .$$

The following lemma justifies that tensor product is an explicit feature mapping for the homogeneous polynomial kernel.

LEMMA 4. *Given any pair of points x, y and an integer p, we have:*

$$\left\langle x^{(p)}, y^{(p)}\right\rangle = \langle x, y\rangle^p \ .$$

PROOF. See [17, Proposition 2.1]. □

By taking $y = x$ on the lemma 4, we have:

LEMMA 5. *Given any point x and an integer p, we have:*

$$\|x^{(p)}\|^2 = \|x\|^{2p} .$$

It is obvious that the tensor product requires d^p dimensions to comprise the polynomial feature space. Therefore, it fails for realistically sized applications.

4. TENSOR SKETCHING APPROACH

As elaborated above, it is infeasible to directly perform any learning algorithm in the polynomial feature space. In this section, we introduce an efficient approach to randomly project the images of data without ever computing their coordinates in that polynomial feature space. The proposed approach runs in $O(np(d + D\log D))$ time for n training examples in d-dimensional space and D random projections; and outputs unbiased estimators of the degree-p polynomial kernel of any pair of data points.

4.1 The Convolution of Count Sketches

Recently, Pagh [14] has introduced a fast algorithm to compute Count Sketch of an outer product of two vectors. Instead of directly computing the outer product, the approach compresses these vectors into their Count Sketches and then computes the Count Sketch of their outer product by those sketches. Due to the fact that the outer product of two different Count Sketches can be efficiently computed by the polynomial multiplication (using FFT), we can compute the Count Sketch of an outer product of any two vectors in time near-linear in the dimensionality of the sketches.

More precisely, given a vector $x \in \mathbb{R}^d$, we denote by $C^{(1)}x, C^{(2)}x \in \mathbb{R}^D$ its two different Count Sketches using *2-wise* independent hash functions $h_1, h_2 : [d] \mapsto [D]$ and $s_1, s_2 : [d] \mapsto \{+1, -1\}$. We consider the outer product $x \otimes x \in \mathbb{R}^{d^2}$ and its Count Sketch $Cx^{(2)} \in \mathbb{R}^D$ using independent and *decomposable* hash functions $H : [d^2] \mapsto [D]$ and $S : [d^2] \mapsto \{+1, -1\}$. We decompose H and S as follows:

$$H(i, j) = h_1(i) + h_2(j) \bmod D \text{ and } S(i, j) = s_1(i)s_2(j).$$

We note that the hash functions H and S are *2-wise* independent [15]. We then represent a Count Sketch in D-dimensional space as a polynomial of degree $D - 1$ where each coordinate corresponds to one term of the polynomial. For example, we consider two degree-$(D - 1)$ polynomials representing for $C^{(1)}x, C^{(2)}x$:

$$P_x^{(1)}(\omega) = \sum_{i=1}^d s_1(i)x_i\omega^{h_1(i)} \text{ and } P_x^{(2)}(\omega) = \sum_{j=1}^d s_2(j)x_j\omega^{h_2(j)}.$$

We can fast compute the degree-$(D-1)$ polynomial for $Cx^{(2)}$ using hash functions H and S:

$$\begin{aligned}P_{x^{(2)}}(\omega) &= \sum_{i,j=1}^d S(i, j)x_ix_j\omega^{H(i,j)} \\ &= \mathrm{FFT}^{-1}(\mathrm{FFT}(P_x^{(1)}) * \mathrm{FFT}(P_x^{(2)})),\end{aligned}$$

where $(*)$ is the component-wise product operator and FFT uses D interpolation points. In other words, the Count Sketch $Cx^{(2)}$ of $x \otimes x$ can be efficiently computed by Count Sketches $C^{(1)}x, C^{(2)}x$ in $O(d + D\log D)$ time.

Inspired by the fast convolution of Count Sketches, we are able to efficiently compute the the polynomial $P_{x^{(p)}}(\omega)$ for the Count Sketch in D-dimensional space, $Cx^{(p)}$, of the tensor product $x^{(p)}$ of any point $x \in \mathbb{R}^d$ by using independent and *decomposable* hash functions $H : [d^p] \mapsto [D]$ and $S : [d^p] \mapsto \{+1, -1\}$. We decompose H and S as follows:

$$H(i_1, \cdots, i_p) = \sum_{k=1}^{p} h_k(i_k) \bmod D,$$

$$S(i_1, \cdots, i_p) = \prod_{k=1}^{p} s_k(i_k),$$

where $h_1, \cdots, h_p : [d] \mapsto [D]$ and $s_1, \cdots, s_p : [d] \mapsto \{+1, -1\}$ are chosen independently from *2-wise* independent family.

The proposed approach works in $O(p(d + D\log D))$ time by using $2p$ different and independent hash functions as elaborated above. This idea motivates the intuition for Tensor Sketching approach to approximate polynomial kernels.

4.2 Tensor Sketching Approach

We exploit the fast computation of Count Sketches on tensor domains to introduce an efficient algorithm for approximating the polynomial kernel $\kappa(x, y) = (\langle x, y\rangle + c)^p$, for an integer p and $c \geq 0$. It is obvious that we can avoid the constant c by adding an extra dimension of value $\sqrt{c}$ to all data points. So, for simplicity, we solely consider the homogeneous polynomial kernel $\kappa(x, y) = \langle x, y\rangle^p$ for the proposed algorithm and theoretical analysis.

For each point $x \in S \subset \mathbb{R}^d$, Tensor Sketching returns the Count Sketch of size D of the tensor product $x^{(p)}$ as random feature maps in $\mathbb{R}^D$ for the polynomial kernel. The pseudocode in Algorithm 1 shows how Tensor Sketching works. We maintain $2p$ independent hash functions (lines 2 - 3), where $h_1, \cdots, h_p$ and $s_1, \cdots, s_p$ are 2-wise independent. For each point x, we create p different Count Sketches of size D using these $2p$ different and independent hash functions (line 5). We then compute the Count Sketch of $x^{(p)}$ by the usage of polynomial multiplication (using FFT) (lines 6-8). As a result, we have obtained a random feature mapping f which provides unbiased estimators for the polynomial kernel.

Now, we analyze the complexity of Tensor Sketching. It requires $O(pd\log D)$ space usage to store $2p$ hash functions. For each point, the running time of computing the Count Sketch of its p-level tensor product is $O(pd + pD\log D)$ due to applying FFT. Therefore, the total running time of Tensor Sketching is $O(np(d + D\log D))$. To increase the accuracy of estimates, we choose $D = O(d)$; therefore, we need $O(npd\log d)$ operations compared to $O(nd^2)$ of the previous approaches [10, 16].

Algorithm 1 Tensor Sketching(S, p, D)

Require: A dataset S of size n, the number of random features D and the degree of polynomial kernel p
Ensure: Return Count Sketches of the point set S as a random feature mapping f for the polynomial kernel $\kappa(x, y) = \langle x, y\rangle^p$
1: $f(S) \leftarrow \emptyset$
2: Pick p independent hash functions $h_1, \cdots, h_p : [d] \mapsto [D]$, each from a 2-universal family
3: Pick p independent hash functions $s_1, \cdots, s_p : [d] \mapsto \{+1, -1\}$, each from a 2-universal family
4: **for** each data point $x \in S$ **do**
5: Create p different Count Sketches: $C^{(1)}x, \cdots, C^{(p)}x$
6: $(C^{(1)}x, \cdots, C^{(p)}x) \leftarrow \mathrm{FFT}(C^{(1)}x, \cdots, C^{(p)}x)$
7: Obtain $f(x)$ in frequency domain by the component-wise multiplication $C^{(1)}x * \cdots * C^{(p)}x$
8: $f(x) \leftarrow \mathrm{FFT}^{-1}(f(x))$
9: Insert $f(x)$ into $f(S)$
10: **end for**
11: **return** Return $f(S)$

5. ERROR ANALYSIS

In this section we analyze the precision of estimate of the kernel $\kappa(x, y) = \langle x, y\rangle^p$, where $x, y \in \mathbb{R}^d$ and p is an integer, showing bounds on the number of random features (D) to achieve a given *absolute* or *relative* precision ϵ. It is worth noting that the previous approaches [10, 16, 20, 12] only introduced bounds of an absolute error estimate. Often, however, the kernel has small value and a good absolute error approximation is typically a poor relative error estimate. Large errors of estimate might result in either performance degradation or negligible reduction in computational cost.

5.1 Relative error bound

In contrast to the previous techniques, our approach can be viewed as a specific random projection technique applied to images of data in the explicit polynomial feature space. In fact, Tensor Sketching maintains random projections of images of data in the feature space via independent hash functions of Count Sketches. Therefore, its estimators are unbiased and have tight error bounds.

Given two points $x, y \in \mathbb{R}^d$, we denote by $Cx^{(p)}, Cy^{(p)} \in \mathbb{R}^D$ the Count Sketches of $x^{(p)}$, $y^{(p)} \in \mathbb{R}^{d^p}$, respectively. The lemma 4 and 5 guarantee that

$$\left\langle x^{(p)}, y^{(p)} \right\rangle = \langle x, y\rangle^p, \quad \|x^{(p)}\| = \|x\|^p, \quad \|y^{(p)}\| = \|y\|^p \ .$$

So applying the lemma 2 and 3, we have:

Lemma 6.

$$\mathbf{E}\left[\left\langle Cx^{(p)}, Cy^{(p)} \right\rangle\right] = \langle x, y\rangle^p,$$
$$\mathbf{Var}\left[\left\langle Cx^{(p)}, Cy^{(p)} \right\rangle\right] \leq \frac{1}{D}\left(\langle x, y\rangle^{2p} + \|x\|^{2p}\|y\|^{2p}\right).$$

While previous works on random feature mappings do not provide bounds on the variance of estimates, the variance of our estimate can be bounded. We make use Chebyshev's inequality to bound the relative error, which depends on the cosine of the angle θ_{xy} between x and y.

Lemma 7.

$$\mathbf{P}\left[\left|\left\langle Cx^{(p)}, Cy^{(p)} \right\rangle - \langle x, y\rangle^p\right| \geq \epsilon \langle x, y\rangle^p\right] \leq \frac{2}{D\epsilon^2}\left(\frac{1}{\cos\theta_{xy}}\right)^{2p}.$$

PROOF. Consider the random variable $X = \left\langle Cx^{(p)}, Cy^{(p)} \right\rangle$, Chebyshev's inequality guarantees that:

$$\mathbf{P}[|X - \mathbf{E}[X]| \geq \epsilon \mathbf{E}[X]] \leq \frac{\mathbf{Var}[X]}{\epsilon^2 \mathbf{E}[X]^2}$$
$$\leq \frac{1}{D\epsilon^2} \frac{\langle x, y\rangle^{2p} + \|x\|^{2p} \|y\|^{2p}}{\langle x, y\rangle^{2p}}$$
$$= \frac{1}{D\epsilon^2} \left(\frac{1}{(\cos\theta_{xy})^{2p}} + 1 \right) \leq \frac{2}{D\epsilon^2} \left(\frac{1}{\cos\theta_{xy}} \right)^{2p}.$$

□

It is obvious that we need more random features to approximate polynomial kernels of large degree p. In addition, the relative error depends on the pairwise angles of data points. So we have to use large D for almost orthogonal data points to achieve a good approximation.

5.2 Absolute error bound

Following up on the work of Kar and Karnick [10], we assume that the 1-norm of any point of data can be bounded, such that $\|x\|_1 \leq R$ for any point x and a nonnegative real R. It is clear that $\|x^{(p)}\|_1 = \|x\|_1^p \leq R^p$. So we first establish the bound of $\left|\left\langle Cx^{(p)}, Cy^{(p)} \right\rangle\right|$ for any pair of points x, y as follows:

LEMMA 8.

$$\left|\left\langle Cx^{(p)}, Cy^{(p)} \right\rangle\right| \leq R^{2p} \ .$$

PROOF. The Hölder inequality says that $\left|\left\langle Cx^{(p)}, Cy^{(p)} \right\rangle\right| \leq \|Cx^{(p)}\|_1 \|Cy^{(p)}\|_\infty$. So it suffices to prove that $\|Cx^{(p)}\|_1 \leq R^p$ for any x due to $\|Cx^{(p)}\|_\infty \leq \|Cx^{(p)}\|_1$. By applying the Cauchy-Schwarz inequality, we have: $\|Cx^{(p)}\|_1 = \sum_{i=1}^{D} |Cx_i^{(p)}| \leq \sum_{i=1}^{d^p} |x_i^{(p)}| \leq R^p$. That proves the claim. □

For any pair of points x, y, we use t different pairs of Count Sketches $\left(C^{(1)}x^{(p)}, C^{(1)}y^{(p)}\right), \cdots, \left(C^{(t)}x^{(p)}, C^{(t)}y^{(p)}\right)$. By Hoeffding's inequality, we achieve a tighter absolute error bound than the previous approach [10] as follows:

LEMMA 9. *Let $X = \frac{1}{t}\sum_{i=1}^{t} X_i$ be an average of the sum of independent random variables $X_i = \left\langle C^{(i)}x^{(p)}, C^{(i)}y^{(p)} \right\rangle$ for each $i \in [t]$, $X_i \in [-R^{2p}, R^{2p}]$ for any nonnegative real R. For any $\epsilon > 0$,*

$$\Pr[|X - \mathbf{E}[X]| \geq \epsilon] \leq 2\exp\left(\frac{-t\epsilon^2}{2R^{4p}}\right) \ .$$

Our absolute error bound depends on the largest value taken by the polynomial kernel in the data space (e.g. R^{2p}). In fact, no algorithm guaranteeing an absolute error can avoid this dependence due to the unbounded nature of the polynomial kernel.

5.3 Normalization

Empirically, it has been shown that normalizing a kernel may improve the performance of SVMs. A way to do so is to normalize the data such as $\|x\| = 1$ so that the exact kernel is properly normalized, i.e. $\kappa(x, x) = \langle x, x\rangle^p = 1$. The following lemma shows that Count Sketches can preserve the normalization of kernels.

LEMMA 10. *Given fixed constants $\epsilon, \delta < 1$ and a point x such that $\|x\| = 1$, we denote by $Cx^{(p)} \in \mathbb{R}^D$ the Count Sketch of $x^{(p)}$. If $\|x^{(p)}\|_\infty \leq \frac{\epsilon}{18\sqrt{\log(1/\delta)\log(D/\delta)}}$ and $D \geq 72\log(1/\delta)/\epsilon^2$, we have that*

$$\mathbf{Pr}\left[\left|\left\langle Cx^{(p)}, Cx^{(p)} \right\rangle - 1\right| \geq \epsilon\right] \leq 2\delta.$$

PROOF. See [22, Appendix B]. □

It is obvious that our kernel approximation can maintain the normalization of kernels within an arbitrarily small factor. In contrast, the Maclaurin expansion based approach [10] does not satisfy this property.

6. EXPERIMENTAL RESULTS

We implemented random feature mappings in Matlab-7.11.0 and conducted experiments in a 2.67 GHz core i7 Windows platform with 3GB of RAM. We compared the performance of random feature mappings, including Tensor Sketching (TS) and Random Maclaurin (RM) [10] with non-linear SVMs on 4 real world datasets: Adult [8], Mnist [11], Gisette [1], and Covertype[1] [8]. We used LIBSVM-3.14 [1] for non-linear kernels and LIBLINEAR-1.92 [6] for random feature mappings for classification task. All averages and standard deviations are over 5 runs of the algorithms.

6.1 Accuracy of Estimation

This subsection presents the accuracy experiments to evaluate the reliability of our estimation algorithm. We carried out experiments to compare the accuracy of estimators based on the number of random features (D) on two random feature mappings: Tensor Sketching (TS) and Random Maclaurin (RM). We measured the relative error of the approximation of the homogeneous and inhomogeneous polynomial kernels of degree $p = 2, 3, 4$. We took D in ranges [500, 3000] and conducted experiments on Adult dataset with size $n = 48,842$ and dimensionality $d = 123$. Figure 1 displays the relative error (ϵ) from expectation of the two approaches on different polynomial kernels.

It is obvious that TS provides a smaller error than the RM approach on those polynomial kernels. The difference is most dramatic on the homogeneous kernels because of the use of Rademacher vectors $\omega_i \in \{+1, -1\}^d$ in RM. In fact, it estimates $\langle x, y\rangle^p$ as $\prod_{i=1}^{p} \langle \omega_i, x\rangle \prod_{i=1}^{p} \langle \omega_i, y\rangle$, which incurs very large variance, especially for large p. Due to the fact that we have to normalize data before applying any kernel method, RM gives small error on inhomogeneous kernels. In this case, the value of Maclaurin expansion concentrates on some low order terms that have small variance of estimate. When the accuracy of kernel machines depends on higher order terms, RM either suffers from low accuracy or needs large D due to large variance of estimate. In contrast, TS is a specific random projection in the polynomial feature space. So it greatly outperforms RM and does not require a large number of random features to achieve a small error. For example, on the inhomogeneous kernels, TS only needs $D = 500$ to achieve $\epsilon < 1$ while RM requires more than 3000 random features.

[1]We sample 100,000 points for Covertype datasets due to the limit of RAM

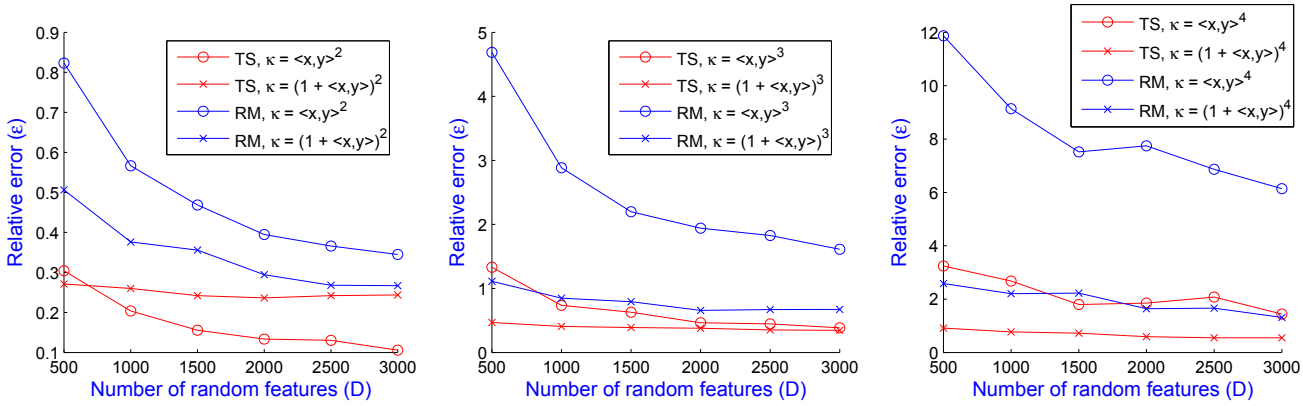

Figure 1: Comparison of relative errors between Tensor Sketching (TS) and Random Maclaurin (RM) estimators on the Adult dataset (n = 48,842, d = 123) using different polynomial kernels. (Figures best viewed in color.)

6.2 Efficiency

This subsection compares the random feature construction time of the two approaches, TS and RM, on two large high-dimensional datasets: Adult ($d = 123$) and Mnist ($d = 780$). As analyzed above, TS requires $O(np(d + D\log D))$ time while RM demands $O(ndD)$ time and much randomness. It is obvious that the running time of TS is faster and less dependent on the original dimensionality of data, a very desirable property since random feature mapping often contributes a significant computational cost in training large-scale high-dimensional datasets.

Figure 2.a shows the CPU time requirements in seconds of the two approaches on the kernel $\kappa = (1+\langle x, y\rangle)^4$ when varying the number of random features D in ranges $[0, 4000]$ and fixing the number of training samples $n = 10,000$. It is clear that the running time of TS approach is almost independent from the dimensionality of data d when using large D. On both Adult ($d = 123$) and Mnist ($d = 779$) datasets, TS approach scales well when increasing D compared to RM on Adult dataset. In contrast, RM shows a linear dependence with d, as depicted on Mnist dataset ($d = 780$). When the dataset (e.g. Mnist) has a smooth decision boundary, RM feature construction time dominates the training time. This property might limit the use of RM.

When the dimensionality of data d increases, we need to increase the number of random features $D = O(d)$ to boost the accuracy. Figure 2.b demonstrates a quadratic running time of RM in terms of dimensionality of data on the synthetic dataset with setting $d = D$ and $n = 10,000$. This means that RM will be a bottleneck of kernel machines on high-dimensional datasets. The next section will show a significant domination of RM feature mapping when training on the Gisette dataset ($d = 5000$).

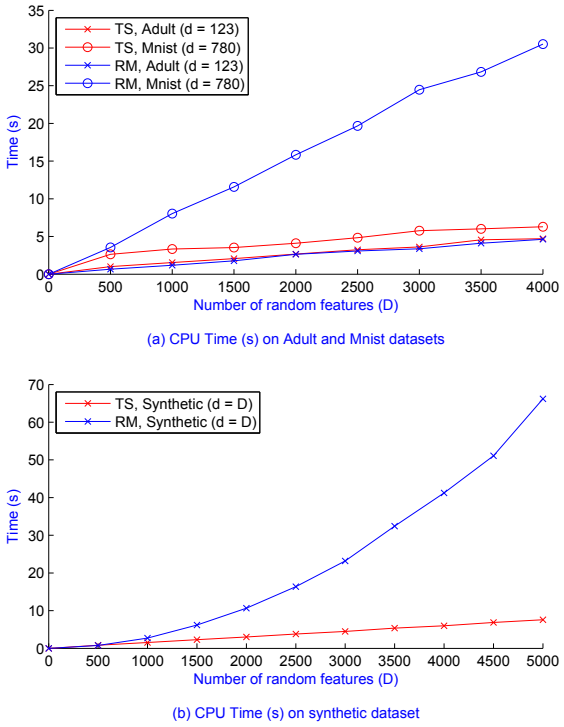

Figure 2: Comparison of CPU time (s) between Tensor Sketching (TS) and Random Maclaurin (RM) approaches on 3 datasets: (a) Adult (d = 123) and Mnist (d = 780); (b) Synthetic ($d = D$) using $\kappa = (1 + \langle x, y\rangle)^4$. (Figures best viewed in color.)

6.3 Scalability

In this experiment, we compare the performance of random feature mappings (TS, RM) along with LIBLINEAR [6] and non-linear kernel mapping along with LIBSVM [1] for classification tasks on 4 large-scale datasets. We measured the training accuracy and time of these approaches on a variety of polynomial kernels. We note that training time of random feature mapping approaches include time for feature construction and linear SVMs training.

Figure 3 demonstrates a comparison of accuracy between TS, RM and non-linear SVMs on degree-2 polynomial kernels. The results impressively show that TS provides higher accuracy than RM on 4 datasets. The most dramatic dif-

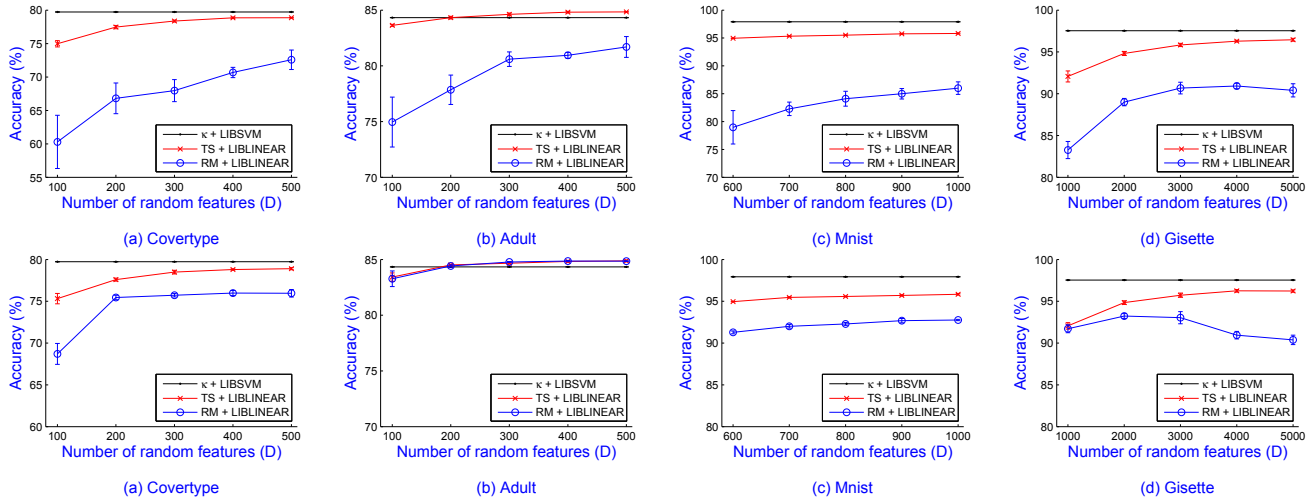

Figure 3: Comparison of accuracy of Tensor Sketching (TS), Random Maclaurin (RM) with LIBLINEAR and non-linear kernels with LIBSVM on 4 datasets with the homogeneous kernel $\kappa = \langle x, y \rangle^2$ (upper row) and the inhomogeneous kernel $\kappa = (1 + \langle x, y \rangle)^2$ (lower row). (Figures best viewed in color.)

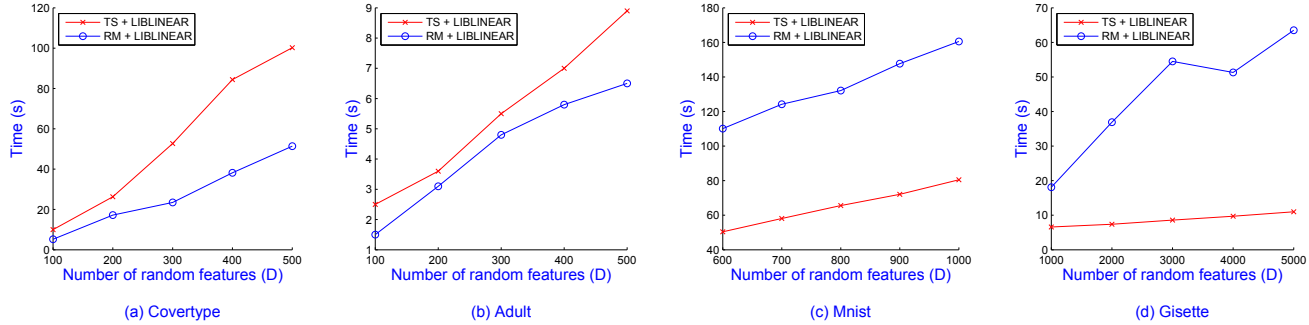

Figure 4: Comparison of training time between Tensor Sketching (TS) and Random Maclaurin (RM) with LIBLINEAR on 4 datasets with the inhomogeneous polynomial kernel $\kappa = (1 + \langle x, y \rangle)^2$. (Figures best viewed in color.)

ference is on the homogeneous kernels due to large error of estimate of RM. Moreover, the accuracy of TS converges faster than RM to that of non-linear kernels when increasing the number of random features D. RM even decreases the accuracy on Gisette dataset because it requires a significantly large number of random features for approximating higher order terms of Maclaurin expansion well.

Figure 4 shows the CPU time requirements in seconds of the two approaches in training linear SVMs using LIBLINEAR on the kernel $\kappa = (1 + \langle x, y \rangle)^2$. It is obvious that TS provides performance benefits on high-dimensional datasets, such as Mnist and Gisette. On Covertype and Adult datasets, RM is slightly faster than TS. This is because RM generates more features for the low order terms of Maclaurin expansion which do not require high computational cost. When p is large, TS significantly outperforms RM, as illustrated in Table 1.

It is obvious that RM performs quite poorly on homogeneous kernels on the 4 datasets. Due to the large error of estimate in homogeneous kernels, RM provides low accuracy on 4 datasets, especially in the kernel $\kappa = \langle x, y \rangle^4$. In fact, the large error of estimate produces meaningless results of training linear SVMs (e.g. 41.45 % of accuracy on the Mnist dataset). In contrast, TS shows stronger results than both RM and non-linear SVMs because it requires rather small time for feature construction and linear SVMs training while obtaining similar accuracy. TS performs exceptionally well on datasets of non-smooth decision boundaries, including Covertype and Adult, where it can achieve speed-ups of 50 and 1600 times, respectively, compared to non-linear SVMs on the kernel $\kappa = (1 + \langle x, y \rangle)^4$.

RM works better on inhomogeneous polynomial kernels because the value of Maclaurin expansion concentrates on some low order terms. However it suffers from large computational cost of random mapping in high-dimensional datasets (e.g. Gisette and Mnist). Because these datasets have smooth decision boundaries, their training time is dominated by the random feature construction time. So RM gives similar performance to non-linear SVMs on the Gisette dataset. When RM suffers from large error of estimate, it can influence the smoothness of decision boundaries of linear SVMs algorithm and therefore require more training time. This explains the inefficiency of RM compared to non-linear SVMs on Mnist dataset on the kernel $\kappa = (1 + \langle x, y \rangle)^4$.

Table 1: Comparison of Tensor Sketching (TS), Random Maclaurin (RM) feature mappings with LIBLINEAR and non-linear kernels with LIBSVM on 4 datasets on many polynomial kernels.

Kernel	κ+libsvm	TS+liblinear	RM+liblinear
$\langle x,y\rangle^2$	79.73% 11.3 mins	78.87±0.06% 1.6 mins	72.58±1.46% 3.7 secs
$(1+\langle x,y\rangle)^2$	79.73% 11.5 mins	78.90±0.12% 1.7 mins	75.96±0.45% 0.8 mins
$\langle x,y\rangle^4$	84.01% 1 hour	79.39±0.13% 1.6 mins	**58.55±2.75%** 3.3 secs
$(1+\langle x,y\rangle)^4$	84.20% **1.5 hours**	79.36±0.19% 1.8 mins	76.76±0.42% 1.6 mins

(a) Covertype (n = 100,000, d = 54, D = 500)

Kernel	κ+libsvm	TS+liblinear	RM+liblinear
$\langle x,y\rangle^2$	97.92% 4.7 mins	95.81±0.08% 1.3 mins	86.00±1.12% 0.5 mins
$(1+\langle x,y\rangle)^2$	97.93% 4.7 mins	95.84±0.10% 1.3 mins	92.76±0.08% 2.7 mins
$\langle x,y\rangle^4$	97.17% 5 mins	92.49±0.22% 2.1 mins	**41.45±4.81%** 0.5 mins
$(1+\langle x,y\rangle)^4$	97.31% 5 mins	92.44±0.04% 2.1 mins	90.07±0.65% **17.2 mins**

(b) Mnist (n = 60,000, d = 780, D = 1000)

Kernel	κ+libsvm	TS+liblinear	RM+liblinear
$\langle x,y\rangle^2$	84.33% 0.5 hours	84.33±0.12% 3.6 secs	77.85±1.32% < 1 sec
$(1+\langle x,y\rangle)^2$	84.34% 0.5 hours	84.51±0.07% 3.8 secs	84.42±0.10% 3.4 secs
$\langle x,y\rangle^4$	79.34% 2 hours	81.09±0.63% 4.3 secs	**58.04±2.37%** < 1 sec
$(1+\langle x,y\rangle)^4$	79.31% **2 hours**	81.89±0.24% 4.5 secs	84.04±0.46% 14.8 secs

(c) Adult (n = 48,842, d = 123, D = 200)

Kernel	κ+libsvm	TS+liblinear	RM+liblinear
$\langle x,y\rangle^2$	97.54% 1.4 mins	96.46±0.17% 10.6 secs	90.40±0.79% 1 min
$(1+\langle x,y\rangle)^2$	97.54% 1.4 mins	96.23±0.14% 10.6 secs	90.38±0.56% 1.1 mins
$\langle x,y\rangle^4$	97.91% 1.8 mins	95.11±0.15% 13.6 secs	**78.89±0.68%** 1.5 mins
$(1+\langle x,y\rangle)^4$	97.91% 1.8 mins	95.21±0.33% 13.5 secs	88.86±0.57% **2.9 mins**

(d) Gisette (n = 7000, d = 5000, D = 5000)

In contrast, the TS approach gives more stable and better performance than RM and non-linear SVMs approaches on 4 datasets. In particular, it has a slightly lower accuracy but runs much faster than non-linear SVMs. It not only achieves higher accuracy (up to 7%) but also runs faster (up to 13 times) than RM on the Mnist and Gisette datasets. Table 2 shows the speedup of TS compared to RM and non-linear SVMs on 4 datasets on the kernel $\kappa = (1+\langle x,y\rangle)^4$.

Table 2: Speedup of Tensor Sketching compared to Random Maclaurin and non-linear SVMs on $\kappa = (1+\langle x,y\rangle)^4$.

Datasets	Random Maclaurin		κ + libsvm
	Mapping	Training	
Adult (D = 200)	—	8×	1600×
Covertype (D = 500)	—	—	50×
Mnist (D = 1000)	2×	9×	2×
Gisette (D = 5000)	9×	25×	8×

The TS random mapping does not show any speedup on low-dimensional datasets (e.g. Covertype, Adult) compared to RM, except for achieving smaller error. However, TS runs 8 times faster than RM in training the Adult dataset due to smaller estimation error. For high-dimensional datasets (e.g. Mnist and Gisette), TS shows speedup on both random mapping and training time. Compared to non-linear kernels, TS achieves significant speedup on Adult and Covertype which have non-smooth decision boundaries.

6.4 Comparison with Heuristic H0/1

In the previous work, the authors [10] introduce a heuristic named H0/1 for fast training. Due to the fact that we have to normalize data before applying any SVM-based learning algorithms, the value of Maclaurin expansion often concentrates on the low order terms. Therefore, we can precompute the first and second terms of the Maclaurin expansion to achieve higher accuracy. For example, consider a Maclaurin expansion of a degree-4 polynomial kernel as follows: $\kappa = (1+\langle x,y\rangle)^4 = 1+4\langle x,y\rangle+6\langle x,y\rangle^2+4\langle x,y\rangle^3+\langle x,y\rangle^4$. We can easily compute $1+4\langle x,y\rangle$ in advance and use D' random features to estimate $6\langle x,y\rangle^2+4\langle x,y\rangle^3+\langle x,y\rangle^4$. This means that H0/1 needs $D = d + D'$ random features and is able to achieve higher accuracy due to the use of D' random features for approximating higher order terms.

However, H0/1 shows some disadvantages: (1) it cannot be used for homogeneous kernels; (2) it is not a dimensionality reduction technique because of using $d + D'$ random features and (3) H0/1 requires longer feature construction times due to the use of more randomness. When d is large, the feature construction time is even larger and often dominates the training time. Table 3 shows the comparison between Tensor Sketching and Random Maclaurin with H0/1. Note that we do not use H0/1 on the Gisette dataset because of the large computational cost of random feature construction.

Table 3: Comparison of Tensor Sketching and Random Maclaurin with H0/1 on $\kappa = (1+\langle x,y\rangle)^4$.

Datasets	Tensor Sketching	Random Maclaurin with H0/1
Adult D = 200	81.89±0.24% 4.5 secs	84.79±0.09% **5.7 secs**
Covertype D = 500	79.36±0.19% 1.8 mins	78.88±0.12% **2.2 mins**
Mnist D = 1000	92.44±0.04% 2.1 mins	89.19±0.74% **7.8 mins**

Although RM with H0/1 can offer better accuracy than plain RM, its accuracy is still lower than TS, except on the Adult dataset. In fact, the Adult dataset works well and achieves higher accuracy (84.92%) on the kernel $\kappa = 1 + 4\langle x, y\rangle$ than with $\kappa = (1 + \langle x, y\rangle)^4$ (79.31%). That explains why the accuracy of RM with H0/1 is exceptionally high. Due to the use of more randomness, the feature construction time of RM with H0/1 is much longer than TS on 3 datasets. In general, H0/1 is only suitable for low-dimensional datasets and works well when the value of polynomial kernel highly concentrates on the first and second terms of Maclaurin expansion.

7. CONCLUSION

In this paper, we have introduced a fast and scalable randomized tensor product technique for approximating polynomial kernels, accelerating the training of kernel machines. By exploiting the connection between tensor product and fast convolution of Count Sketches, our approximation algorithm works in time $O(n(d+D\log D))$ for n training samples in d-dimensional space and D random features. We present a theoretical analysis of the quality of approximation to guarantee the reliability of our estimation algorithm. We show empirically that our approach achieves higher accuracy and often runs orders of magnitude faster than the state-of-the-art approach on large-scale real-world datasets.

An interesting research direction is analyzing and evaluating Tensor Sketching on other learning tasks, such as clustering [4] and multitask learning [22] on large-scale datasets. We also intend to apply Tensor Sketching on other kernels (e.g. Gaussian kernel, sigmoid kernel) by exploiting Taylor-series approximations of these kernels. By applying Tensor Sketching on Taylor-series approximations, we might achieve a substantial speedup in training these kernel machines.

8. ACKNOWLEDGMENTS

We thank P. Kar for useful discussion and released source code in the early state of the project. We also thank the anonymous reviewers for their constructive comments and suggestions.

9. REFERENCES

[1] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[2] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *Proceedings of ICALP'02*, pages 693–703, 2002.

[3] R. Chitta, R. Jin, T. C. Havens, and A. K. Jain. Approximate kernel k-means: solution to large scale kernel clustering. In *Proceedings of KDD'11*, pages 895–903, 2011.

[4] R. Chitta, R. Jin, and A. K. Jain. Efficient kernel clustering using random fourier features. In *Proceedings of ICDM'12*, pages 161–170, 2012.

[5] P. Drineas and M. W. Mahoney. On the Nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

[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] S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.

[8] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[9] T. Joachims. Training linear SVMs in linear time. In *Proceedings of KDD'06*, pages 217–226, 2006.

[10] P. Kar and H. Karnick. Random feature maps for dot product kernels. In *Proceedings of AISTATS'12*, pages 583–591, 2012.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.

[12] S. Maji and A. C. Berg. Max-margin additive classifiers for detection. In *Proceedings of ICCV'09*, pages 40–47, 2009.

[13] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Proceedings of NNSP'97*, pages 276–285, 1997.

[14] R. Pagh. Compressed matrix multiplication. In *Proceedings of ICTS'12*, pages 442–451, 2012.

[15] M. Pătraşcu and M. Thorup. The power of simple tabulation hashing. In *Proceedings of STOC'11*, pages 1–10, 2011.

[16] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in NIPS'08*, pages 1177–1184, 2007.

[17] B. Schökopf and A. J. Smola. *Learning with kernels: Support vector machines, regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[18] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of ICML'07*, pages 807–814, 2007.

[19] A. J. Smola and B. Schökopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of ICML'00*, pages 911–918, 2000.

[20] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *Proceedings of CVPR'10*, pages 3539–3546, 2010.

[21] S. Vempati, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Generalized RBF feature maps for efficient detection. In *Proceedings of BMVC'10*, pages 1–11, 2010.

[22] K. Q. Weinberger, A. Dasgupta, J. Langford, A. J. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of ICML'09*, pages 1113–1120, 2009.

[23] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in NIPS'01*, pages 682–688, 2001.

[24] 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 *Advances in NIPS'12*, pages 485–493, 2012.