CoRE Kernels

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

Abstract

The term "CoRE kernel" stands for *correlation-resemblance kernel*. In many real-world applications (e.g., computer vision), the data are often high-dimensional, sparse, and non-binary. We propose two types of (nonlinear) CoRE kernels for non-binary sparse data and demonstrate the effectiveness of the new kernels through a classification experiment. CoRE kernels are simple with no tuning parameters. However, training nonlinear kernel SVM can be costly in time and memory and may not be always suitable for truly large-scale industrial applications (e.g., search). In order to make the proposed CoRE kernels more practical, we develop basic probabilistic hashing (approximate) algorithms which transform nonlinear kernels into linear kernels.

1 INTRODUCTION

The use of high-dimensional data has become popular in practice, especially in search, natural language processing (NLP), and computer vision. For example, Winner of 2009 PASCAL image classification challenge [27] used 4 million (non-binary) features. [5, 25, 28] mentioned datasets with billions or even trillions of features.

For text data, the use of extremely high-dimensional representations (e.g., n-grams) is the standard practice. In fact, binary representations for text data could be sufficient if the order of n-grams is high enough. On the other hand, in current practice of computer vision, it is still more common to use non-binary feature representations, for example, *local coordinate coding (LCC)* [29, 27]. It is often the case that in practice high-dimensional non-binary features might be appropriately sparsified without hurting the performance of subsequent tasks (e.g., classification). However simply binarizing the features will often incur loss of accuracies, sometimes significantly so. See Table 1 for an illustration of such a phenomenon.

Our contribution in this paper is the proposal of two types of (nonlinear) "CoRE" kernels, where "CoRE" stands for "correlation-resemblance", for non-binary sparse data. Interestingly, using CoRE kernels leads to improvement in classification accuracies (in some cases significantly so) on a variety of datasets (see Table 2).

For practical large-scale applications, naive implementations of nonlinear kernels may be too costly (in time and/or memory), while linear learning methods (e.g., linear SVM or logistic regression) are extremely popular in industry. The proposed CoRE kernels would be facing the same challenge. To address this critical issue, we also develop basic hashing algorithms which approximate the CoRE kernels by linear kernels. These new hashing algorithms allow us to take advantage of highly efficient (batch or stochastic) linear learning algorithms, e.g., [15, 24, 1, 8].

In the rest of this section, we first review the definitions of correlation and resemblance, then we provide an experimental study to illustrate the loss of classification accuracies when sparse data are binarized.

1.1 Correlation

We assume a data matrix of size $n \times D$, i.e., n observations in D dimensions. Consider, without loss of generality, two data vectors $u, v \in \mathbb{R}^D$. The correlation is simply the normalized inner product defined as follows

$$\rho = \rho(u,v) = \frac{\sum_{i=1}^D u_i v_i}{\sqrt{\sum_{i=1}^D u_i^2 \sum_{i=1}^D v_i^2}} = \frac{A}{\sqrt{m_1 m_2}}, \quad (1)$$

$$\text{where } A = \sum_{i=1}^D u_i v_i, \;\; m_1 = \sum_{i=1}^D u_i^2, \;\; m_2 = \sum_{i=1}^D v_i^2$$

It is well-known that $\rho(u,v)$ constitutes a positive definite and linear kernel, which is one of the reasons why correlation is very popular in practice.

1.2 Resemblance

For binary data, the resemblance is commonly used:

$$R = R(u,v) = \frac{a}{f_1 + f_2 - a}, \tag{2}$$

$$\text{where } f_1 = \sum_{i=1}^{D} 1\{u_i \neq 0\}, \quad f_2 = \sum_{i=1}^{D} 1\{v_i \neq 0\},$$

$$a = \sum_{i=1}^{D} 1\{u_i \neq 0\} 1\{v_i \neq 0\}$$

It was shown in [22] that the resemblance defines a type of positive definite (nonlinear) kernel. In this study, we will combine correlation and resemblance to define two new types of nonlinear kernels.

1.3 Linear SVM Experiment

Table 1 lists the datasets, which are non-binary and sparse. The table also presents the test classification accuracies using linear SVM on both the original (non-binary) data and the binarized data. The results in the table illustrate the noticeable drop of accuracies by using only binarized data.[1]

Available at the UCI repository, *Youtube* is a multi-view dataset, and we choose the largest set of features (audio) for our experiment. *M-Basic*, *M-Rotate*, and *MNIST10k* were used in [18] for testing *abc-logitboost* and *abc-mart* [17] (and comparisons with deep learning [16]). For *RCV1*, we use a subset of the original testing examples (to facilitate efficient kernel computation later needed in the paper).

Table 1: Classification accuracies using linear SVM (LIBLINEAR [8]) on sparse non-binary data. As we always normalize data to unit norm, the correlation kernel ρ is naturally used in our study. We experiment with the l_2-regularized linear SVM (with a regularization parameter "C") and report the best test accuracies from a wide range of C values. Using binarized data (i.e., the last column), the test accuracies drop very noticeably in most datasets.

Dataset	#Train	#Test	Linear	Lin. Bin.
M-Basic	12,000	50,000	90.0%	88.9%
MNIST10k	10,000	60,000	90.0%	88.8%
M-Rotate	12,000	50,000	48.0%	44.4%
RCV1	20,242	60,000	96.3%	95.6%
USPS	7,291	2,007	91.8%	87.4%
Youtube	11,930	97,934	47.6%	46.5%

Figure 1 provides more detailed classification accuracy results for a wide range of C values, where C is the usual l_2-regularization parameter in linear SVM.

[1]For all datasets except *USPS*, we used "0" as the threshold to binarize the data. For *USPS*, since it contains many very small entries, we used a threshold which is slightly different from zero.

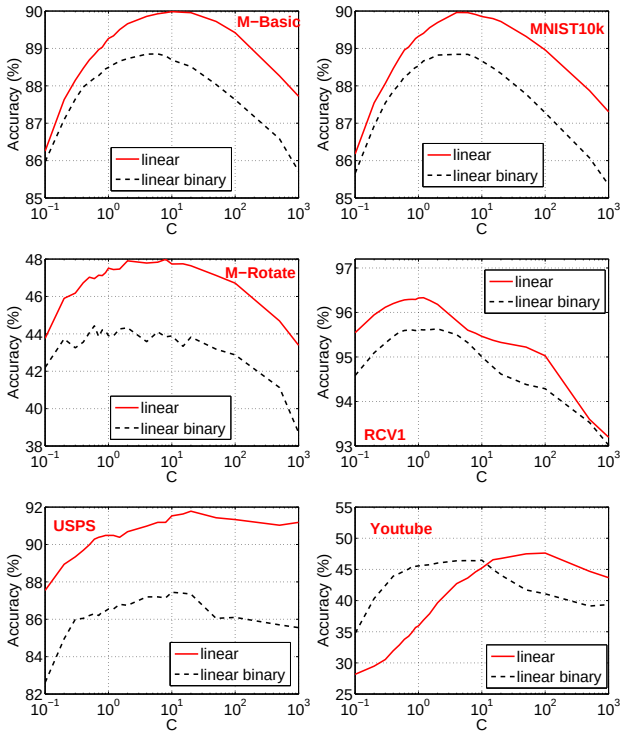

Figure 1: Test classification accuracies for both the original (non-binary, solid) and the binarized (dashed) data, using l_2-regularized linear SVM with a regularization parameter C. We present results for a wide range of C values. The best (highest) values are summarized in Table 1.

While linear SVM is extremely popular in industrial practice, it is often not as accurate. Our proposed CoRE kernels will be able to produce noticeably more accurate results.

2 CORE KERNELS

We propose two types of CoRE kernels, which combine resemblance with correlation, for sparse non-binary data. Both kernels are positive definite. We will demonstrate the effectiveness of the two CoRE kernels using the same datasets in Table 1 and Figure 1.

2.1 CoRE Kernel, Type 1

The first type of CoRE kernel is basically the product of correlation ρ and the resemblance R, i.e.,

$$K_{C,1} = K_{C,1}(u,v) = \rho R \tag{3}$$

Later in the paper we will express $K_{C,1}$ as an (expectation of) inner product, i.e., $K_{C,1}$ is obviously positive definite.

If the data are fully dense (i.e., no zero entries), then $R = 1$ and $K_{C,1} = \rho$. On the other hand, if the data are binary, then $\rho = \frac{a}{\sqrt{f_1 f_2}}$ and $K_{C,1} = \frac{a}{\sqrt{f_1 f_2}} \frac{a}{f_1 + f_2 - a}$. See (2) for the definitions of f_1, f_2, a.

2.2 CoRE Kernel, Type 2

The second type of CoRE kernel perhaps appears less intuitive than the first type:

$$K_{C,2} = K_{C,2}(u,v) = \rho \frac{\sqrt{f_1 f_2}}{f_1 + f_2 - a} = \frac{\rho R}{a/\sqrt{f_1 f_2}} \quad (4)$$

If the data are binary, then $K_{C,2} = R$. We will, later in the paper, also write $K_{C,2}$ as an expectation of inner product to confirm it is also positive definite.

2.3 Kernel SVM Experiment

Figure 2 presents the classification accuracies on the same 6 datasets as in Figure 1 and Table 1, using nonlinear kernel SVM with three different kinds of kernels: CoRE Type 1, CoRE Type 2, and resemblance. We can see that resemblance (which only uses binary information of the data) does not perform as well as CoRE kernels.

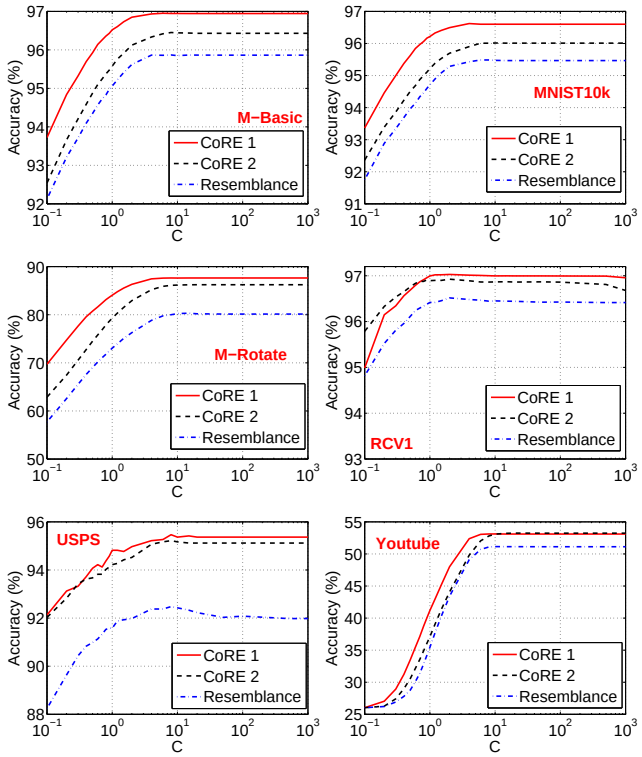

Figure 2: Test classification accuracies using nonlinear kernel SVM and three types of kernels: CoRE Type 1, CoRE Type 2, and resemblance. We use the LIBSVM precomputed kernel functionality. Compared with the results of linear SVM in Figure 1, we can see CoRE kernels and resemblance kernel perform better (or much better, especially on *M-Rotate* dataset). The best results (highest points on the curves) are summarized in Table 2.

The best results in Figure 2 are summarized in Table 2. It is interesting to compare them with the test accuracies of linear SVM in Table 1 and Figure 1. We can see that CoRE kernels perform very well, without using additional tuning parameters. In fact, if we compare the best results in [16, 18] (e.g., RBF SVM, abc-boosting, or deep learning) on MNIST10k, M-Rotate, and M-Basic, we will see that CoRE kernels (with no tuning parameters) can achieve the same (or similar) performance.

Table 2: Best test classification accuracies (in %) for five different kernels. The first two columns (i.e., "linear" and "linear binary") are already shown in Table 1.

Dataset	Lin.	Lin. Bin.	Res.	CoRE1	CoRE2
M-Basic	90.0	88.9	95.9	**97.0**	96.5
MNIST10k	90.0	88.8	95.5	**96.6**	96.0
M-Rotate	48.0	44.4	80.3	**87.6**	86.2
RCV1	96.3	95.6	96.5	**97.0**	96.9
USPS	91.8	87.4	92.5	**95.5**	95.2
Youtube	47.6	46.5	51.1	53.1	**53.2**

We shall mention that our experiments can be fairly easily reproduced because all datasets are public and we use standard SVM packages (LIBSVM and LIBLINEAR) without any modifications. We also provide the results for a wide range of C values in Figure 1 and Figure 2. Note that, because we use pre-computed kernel functionality of LIBSVM (which consumes very substantial memory to store the kernel matrix), we only experiment with training data of moderate sizes, to ensure repeatability (by other researchers without access to machines with large memory).[2]

2.4 Challenges with Nonlinear Kernel SVM

[2, Section 1.4.3] mentioned three main computational issues of kernels summarized as follows:

1. *Computing kernels is very expensive*.
2. *Computing a full kernel matrix is wasteful*, because not all pairwise kernel values are used during training.
3. *The kernel matrix does not fit in memory*. The cost of storing the full kernel matrix in the memory is $O(n^2)$, which is not realistic for most PCs even for merely 10^5, while the industry has used training data with billions of examples. Thus, kernel evaluations are often conducted *on the fly*, which means the computational cost is dominated by kernel evaluations.

In fact, evaluating kernels on-demand would encounter another serious (and often common) issue if the dataset itself is too big for the memory.

All these crucial issues motivate us to develop hashing algorithms to approximate CoRE kernels by linear kernels.

[2]At the time this paper was written, the implementation of LIBSVM restricted the maximum size of the kernel matrix. The LIBSVM team recently has made effort on this issue and it is expected such a restriction will be removed in the new release. We highly appreciate Dr. Chih-Jen Lin and his team for the efforts.

2.5 Benefits of Hashing

Our goal is to develop good probabilistic hashing algorithms to (approximately) transform our proposed nonlinear CoRE kernels into linear kernels. Once we have the new data representations (i.e., the hashed data), we can use highly efficient batch or stochastic linear methods for training SVM (or logistic regression) [15, 24, 1, 8].

Another benefit of hashing would be in the context of approximate near neighbor search because probabilistic hashing provides a (often good) strategy for space partitioning (i.e., bucketing) which will help reduce the search time (i.e., no need to scan all data points). Our proposed hashing methods can be modified to become an instance of *locality sensitive hashing (LSH)* [13] in the space of CoRE kernels.

At this stage, we will focus on developing hashing algorithms for CoRE kernels based on the standard *random projection* and *minwise hashing* methods. There will be plenty of room for improvement which we leave for future work.

We first provide a review of the two basic building blocks.

3 REVIEW OF RANDOM PROJECTIONS AND MINWISE HASHING

Typically, the method of random projections is used for dense high-dimensional data, while the method of minwise hashing is very useful for sparse (often binary) data. The proposed hashing algorithms for CoRE kernels combine random projections and minwise hashing.

3.1 Random Projections

Consider two vectors $u, v \in \mathbb{R}^D$. The idea of random projection is simple. We first generate a random vector of i.i.d. entries r_i, $i = 1$ to D, and then compute the inner products as the hashed values:

$$P(u) = \sum_{i=1}^{D} u_i r_i, \qquad P(v) = \sum_{i=1}^{D} v_i r_i \tag{5}$$

For the convenience of theoretical analysis, we adopt the choice of $r_i \sim N(0, 1)$, which is a typical choice in the literature. Several variants of random projections like [21, 28] are essentially equivalent, as analyzed in [22].

In this study, we always assume the data are normalized, i.e., $\sum_{i=1}^{D} u_i^2 = \sum_{i=1}^{D} v_i^2 = 1$. Note that computing the l_2 norms of all the data points only requires scanning the data once which is anyway needed during data collection/processing. For normalized data, it is known that $E[P(u)P(v)] = \rho$. In order to estimate ρ, we need to use k random projections to generate $P_j(u), P_j(v), j = 1$ to k, and estimate ρ by $\frac{1}{k}\sum_{j=1}^{k} P_j(u)P_j(v)$, which is also an inner product. This means we can directly use the projected data to build a linear classifier.

3.2 Minwise Hashing

The method of minwise hashing [3] is very popular for computing set similarities, especially for industrial applications, for example, [3, 9, 12, 26, 14, 7, 11, 23, 4].

Consider the space of the column numbers: $\Omega = \{1, 2, 3, ..., D\}$. We assume a random permutation $\pi : \Omega \longrightarrow \Omega$ and apply π on the coordinates of both vectors u and v. For example, consider $D = 4$, $u = [0,\ 0.45,\ 0.89,\ 0]$ and $\pi : 1 \to 3,\ 2 \to 1,\ 3 \to 4,\ 4 \to 2$. Then the permuted vector becomes $\pi(u) = [0.45,\ 0,\ 0,\ 0.89]$. In this example, the first nonzero column of $\pi(u)$ is 1, and the corresponding value of the coordinate is 0.45. For convenience, we introduce the following notation:

$$L(u) = \text{location of first nonzero entry of } \pi(u) \tag{6}$$

$$V(u) = \text{value of first nonzero entry of } \pi(u) \tag{7}$$

In this example, we have $L(u) = 1$ and $V(u) = 0.45$.

The well-known *collision probability*

$$\mathbf{Pr}\left(L(u) = L(v)\right) = R(u, v) = R \tag{8}$$

can be used to estimate the resemblance R. To do so, we need to generate k permutations π_j, $j = 1$ to k.

4 HASHING CORE KERNELS

The goal is to develop unbiased linear estimators of CoRE Kernels $K_{C,1}$ and $K_{C,2}$. Linear estimators can be written as inner products. We assume that we have already conducted random projections and minwise hashing k times. In other words, for each data vector u, we have the hashed values $P_j(u)$, $L_j(u)$, $V_j(u)$, $j = 1$ to k. Recall the definitions of P_j, L_j, V_j in (5), (6), and (7), respectively.

4.1 Hashing Type 1 CoRE Kernel

Our proposed estimator of $K_{C,1}$ is

$$\hat{K}_{C,1}(u, v) = \sum_{j=1}^{k} P_j(u)P_j(v)1\{L_j(u) = L_j(v)\} \tag{9}$$

The following Theorem 1 shows $\hat{K}_{C,1}$ is an unbiased estimator and provides its variance.

Theorem 1

$$E\left(\hat{K}_{C,1}\right) = K_{C,1} \tag{10}$$

$$Var\left(\hat{K}_{C,1}\right) = \frac{1}{k}\left\{\left(1 + 2\rho^2\right)R - \rho^2 R^2\right\} \tag{11}$$

Proof: *See Appendix A.* □

A simple argument can show that $\hat{K}_{C,1}$ could be written as an inner product and hence $K_{C,1}$ is positive definite. Although this fact is obvious since $K_{C,1}$ is a product of two positive definite kernels, we would like to present a constructive proof because the construction is basically the same procedure for expanding the hashed data before feeding them to a linear SVM solver.

Recall, L_j is the location of the first nonzero after minwise hashing. Basically, we can view $L_j(u)$ equivalently as a vector of length D whose coordinates are all zero except the $L_j(u)$-th coordinate. The value of the only nonzero coordinate will be $P_j(u)$. For example, suppose $D = 4$, $L_j(u) = 2$, $P_j(u) = 0.1$. Then the equivalent vector would be $[0,\ 0.1,\ 0,\ 0]$. With k projections and k permutations, we can have k such vectors. This way, we can write $\hat{K}_{C,1}$ as an inner product of two $D \times k$-dimensional sparse vectors.

Note that the input data format of standard SVM packages is the sparse format. For linear SVM, the cost is essentially determined by the number of nonzeros (in this case, k), not much to do with the dimensionality (unless it is too high). If D is too high, then we can adopt the standard trick of *b-bit minwise hashing* [22] by only using the lowest b bits of $L_j(u)$. This will lead to an efficient implementation.

4.2 Hashing Type 2 CoRE Kernel

Our proposed estimator for Type 2 CoRE Kernel is

$$\hat{K}_{C,2} = \frac{\sqrt{f_1 f_2}}{k} \sum_{j=1}^{k} V_j(u) V_j(v) 1\{L_j(u) = L_j(v)\} \quad (12)$$

Recall that we always assume the data $(u,\ v)$ are normalized. For example, if the data are binary, then we have $u_i = \frac{1}{\sqrt{f_1}}$, $v_i = \frac{1}{\sqrt{f_2}}$. Hence the values $V_j(u)$ and $V_j(v)$ are small (and we need the term $\sqrt{f_1 f_2}$).

This estimator is again unbiased. Theorem 2 proves the mean the variance of $\hat{K}_{C,2}$.

Theorem 2

$$E\left(\hat{K}_{C,2}\right) = K_{C,2} \quad (13)$$

$$Var\left(\hat{K}_{C,2}\right) \quad (14)$$
$$= \frac{1}{k} \frac{f_1 f_2}{f_1 + f_2 - a} \left(\sum_{i=1}^{D} u_i^2 v_i^2 - \frac{\left(\sum_{i=1}^{D} u_i v_i\right)^2}{(f_1 + f_2 - a)} \right)$$

Proof: *See Appendix B.* □

Once we understand how to express $\hat{K}_{C,1}$ as an inner product, it should be easy to see that $\hat{K}_{C,2}$ can also be written as an inner product. Again, suppose $D = 4$, $L_j(u) = 2$, and $V_j(u) = 0.05$. We can consider an equivalent vector $[0,\ 0.05\sqrt{f_1},\ 0,\ 0]$. In other words, the difference between $\hat{K}_{C,1}$ and $\hat{K}_{C,2}$ is what value we should put in the nonzero location. Compared to $\hat{K}_{C,1}$, one advantage of $\hat{K}_{C,2}$ is that it only requires the permutations and thus eliminates the cost for conducting random projections.

As one would expect, the variance of $\hat{K}_{C,2}$ would be large if the data are heavy-tailed. However, when the data are appropriately normalized (e.g., via the TF-IDF transformation, or simply binarized), $Var\left(\hat{K}_{C,2}\right)$ is actually quite small. Consider the extreme case when the data are binary, i.e., $u_i = \frac{1}{\sqrt{f_1}}$, $v_i = \frac{1}{\sqrt{f_2}}$, we have $Var\left(\hat{K}_{C,2}\right) = \frac{1}{k}\left(R - R^2\right)$, which is (considerably) smaller than $Var\left(\hat{K}_{C,1}\right) = \frac{1}{k}\left\{\left(1 + 2\rho^2\right) R - \rho^2 R^2\right\}$.

4.3 Experiment for Validation

To validate the theoretical results in Theorem 1 and Theorem 2, we provide a set of experiments in Figure 3. Two pairs of word vectors are selected: "A–THE" and "HONG–KONG", from a chuck of web crawls. For example, the vector "HONG" is a vector whose i-th entry is the number of occurrences of the word "HONG" in the i-th document. For each pair, we apply the two proposed hashing algorithms to estimate $K_{C,1}$ and $K_{C,2}$. With sufficient repetitions (i.e., k), we can empirically compute the mean square errors (MSE = Var + Bias^2), which should match the theoretical variances if the estimators are indeed unbiased and the variance formulas, (11) and (14), are correct.

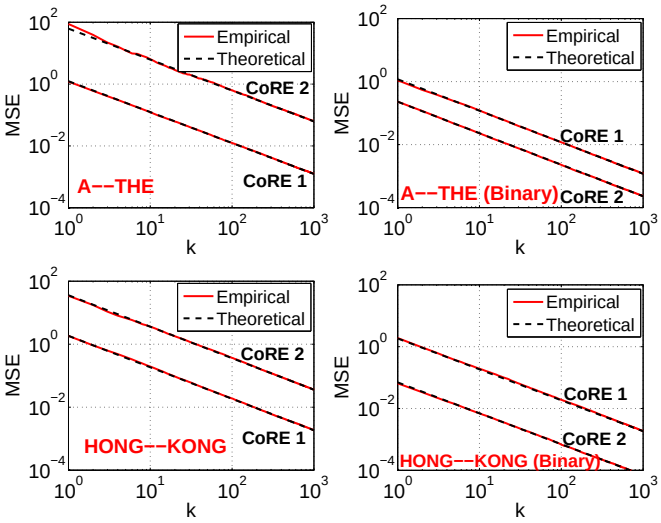

Figure 3: Mean square errors (MSE = Var + Bias^2) on two pairs of word vectors for validating Theorems 1 and 2. The empirical MSEs (solid curves) essentially overlap the theoretical variances (dashed curves), (11) and (14). When using the raw counts (left panels), the MSEs of $\hat{K}_{C,2}$ is significantly higher than the MSEs of $\hat{K}_{C,1}$. However, when using binarized data (right panels), the MSEs of $\hat{K}_{C,2}$ become noticeably smaller, as expected.

The number of word occurrences is a typical example of highly heavy-tailed data. Usually when text data are used in machine learning tasks, they have to be appropriately weighted (e.g., TF-IDF) or simply binarized. Figure 3 presents the results on the original data (raw counts) as well as the binarized data, to verify the formulas in Theorem 1 and Theorem 2, for $k = 1$ to 1000.

Indeed, the plots show that the empirical MSEs essentially overlap the theoretical variances. In addition, the MSEs of $\hat{K}_{C,2}$ is significantly larger than the MSEs of $\hat{K}_{C,1}$ on the raw data, as expected. Once the data are binarized, the MSEs of $\hat{K}_{C,2}$ become smaller, also as expected.

5 HASHING CORE KERNELS FOR SVM

In this section, we provide a set of experiments for using the hashed data as input for a linear SVM solver (LIBLINEAR). Our goal is to approximate the (nonlinear) CoRE kernels with linear kernels. In Section 4, we have explained how to express the estimators $\hat{K}_{C,1}$ and $\hat{K}_{C,2}$ as inner products by expanding the hashed data. With k permutations and k random projections, the number of nonzeros of the expanded data is precisely k. To reduce the dimensionality, we use only the lowest b bits of the locations [22]. In this study, we experiment with $b = 1, 2, 4, 8$.

Figure 4 presents the results on the **M-Rotate** dataset. As shown in Figure 1 and Table 1, using linear kernel can only achieve a test accuracy of 48%. This means, if we use random projections (or the variants, e.g., [21, 28]), which approximate inner products, then the best accuracy we can achieve would be about 48%. For this dataset, the performance of CoRE kernels (and resemblance kernel) is astonishing, as shown in Figure 2 and Table 2. Thus, we choose this dataset to demonstrate our proposed hashing algorithms combined with linear SVM can also approach the performance of (nonlinear) CoRE kernels.

To explain the procedure, we use the same examples as in Section 4. Suppose we apply k minwise hashing and k random projections on the data and we consider without loss of generality the data vector u. For the j-th projection and j-th minwise hashing, suppose $L_j(u) = 2, V_j(u) = 0.05, P_j(u) = 0.1$. Recall L_j and V_j are, respectively, the location and the value of the first nonzero entry after minwise hashing. P_j is the projected value obtained from random projection.

In order to use linear SVM to approximate kernel SVM with Type 1 CoRE kernel, for the above example, we expand the j-th hashed data as a vector $[0,\ 0.1,\ 0,\ 0]$ if $b = 2$, or $[0,\ 0.1]$ if $b = 1$. We then concatenate k such vectors to form a vector of length $2^b \times k$ (with exactly k nonzeros). Before we feed the expanded hashed data to LIBLINEAR, we normalize the vectors to have unit norm. The experimental results are presented in the left panels of Figure 4.

To approximate Type 2 CoRE kernel, we expand the j-th hashed data of u as $[0,\ 0.05\sqrt{f_1},\ 0,\ 0]$ if $b = 2$, or $[0,\ 0.05\sqrt{f_1}]$ if $b = 1$, where f_1 is the number of nonzero entries in the original data vector u. Again, we concatenate k such vectors. The experimental results are presented in the middle panels of Figure 4.

To approximate resemblance kernel, we expand the j-th hashed data of u as $[0,\ 1,\ 0,\ 0]$ if $b = 2$ or $[0,\ 1]$ if $b = 1$ and we concatenate k such vectors.

The results in Figure 4 are exciting because linear SVM on the original data can only achieve an accuracy of 48%. Our proposed hashing methods + linear SVM can achieve $> 86\%$. In comparison, using only the original b-bit minwise hashing, the accuracy can still reach about 80%. Again, we should mention that other hashing algorithms which aim at approximating the inner product (such as random projections and variants) can at most achieve the same result as using linear SVM on the original data. This is the significant advantage of CoRE kernels.

6 DISCUSSIONIS

There is a line of related work called *Conditional Random Sampling (CRS)* [19, 20] which was also designed for sparse non-binary data. Basically, the idea of CRS is to keep the first (smallest) k nonzero entries after applying one permutation on the data. [19, 20] developed the trick to construct an (essentially) equivalent random sample for each pair. CRS is naturally applicable to non-binary data and is capable of estimating any (linear) summary statistics, in static as well as dynamic (streaming) settings. In fact, the estimators developed for CRS can be (substantially) more accurate than the estimator for minwise hashing.

The major drawback of CRS is that the samples are not appropriately aligned. Consequently, CRS is not suitable for training linear SVM (or other applications which require the input data to be in a metric space). Our method has overcome this drawback. Of course, CRS can still be used in important scenarios such as estimating similarities during the re-ranking stage in LSH.

Why do we need to two types of CoRE kernels? While hashing Type 2 CoRE kernel is simpler because it requires only the random permutations, Table 2 shows that Type 1 CoRE kernel can often achieve better results than Type 2 CoRE kernel. Therefore, we develop hashing methods for both CoRE kernels, to provide users with more choices.

There are many promising extensions. For example, we can construct new kernels based on CoRE kernels (which currently do not have tuning parameters), by using the exponential function and introducing an additional tuning parameter γ, just like RBF kernel. This will allow more flexibility and potentially further improve the performance.

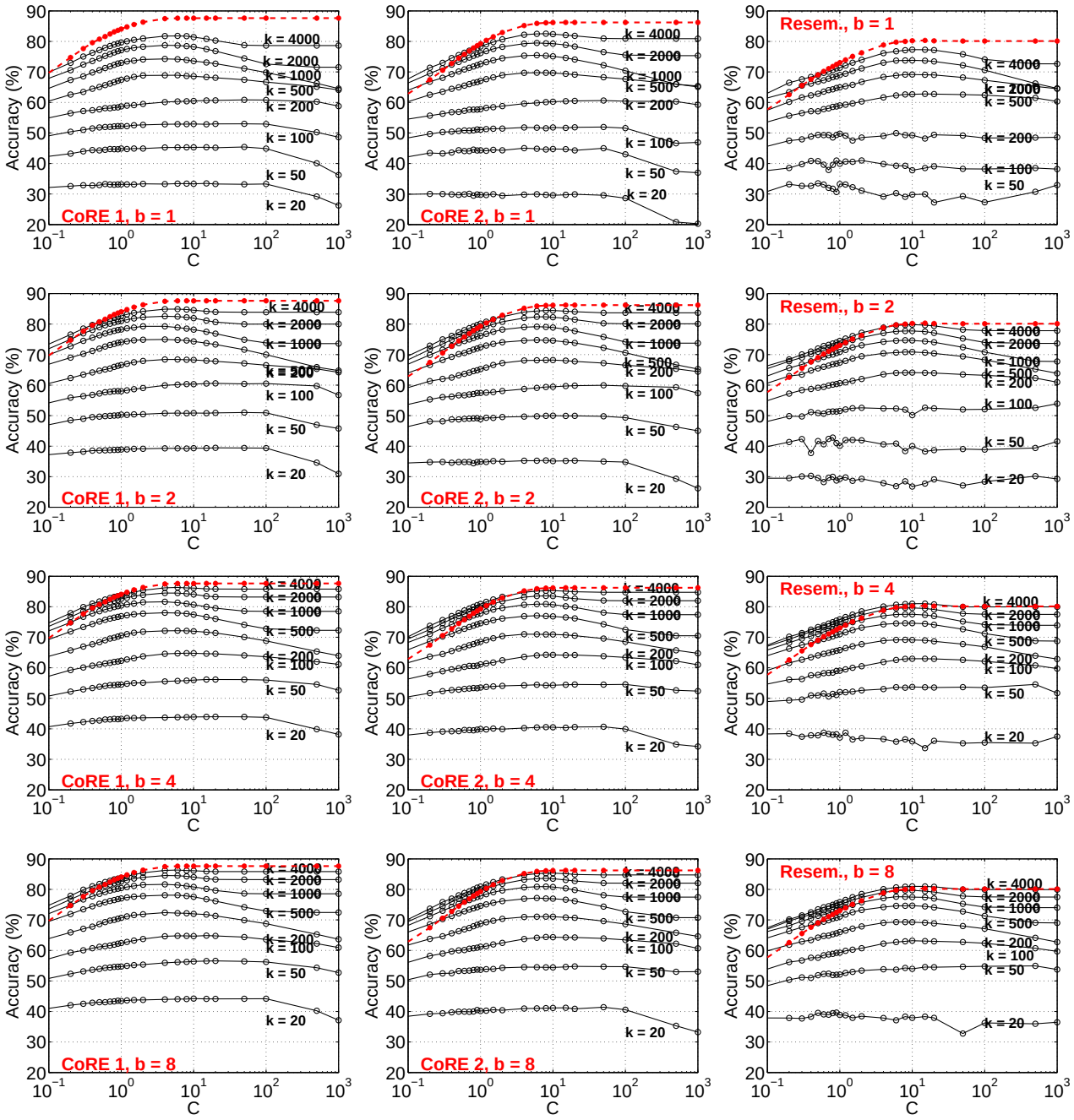

Figure 4: Test classification accuracies on the *M-Rotate* dataset using our proposed hashing methods and linear SVM (LIBLINEAR). The red (if color is available) dot curves are the results of kernel SVM on the original data (i.e., the same curves from Figure 2), using Type 1 CoRE kernel (left panels), Type 2 CoRE kernel (middle panels), and resemblance kernel (right panels), respectively. We apply both b-bit minwise hashing (with $b = 1, 2, 4, 8$) and random projections k times and feed the (expanded) hashed data to linear SVM.

Another interesting line of extensions would be applying other hashing algorithms on top of our generated hashed data. This is possible again because we can view our estimators as inner products and hence we can apply other hashing algorithms which approximate inner products on top of our hashed data. The advantage is the potential further data compression. Another advantage would be in the context of sublinear time approximate near neighbor search (when the target similarity is the CoRE kernels).

For example, we can apply another layer of random projections on top of the hashed data and then store the signs of the new projected data [6, 10]. These signs, which are bits, provide good indexing & space partitioning capability to allow sublinear time approximate near neighbor search under the framework of *locality sensitive hashing (LSH)* [13]. This way, we can search for near neighbors in the space of CoRE kernels (instead of the space of inner products).

In addition, we expect that our work will inspire new research on the development of more efficient (b-bit) minwise hashing methods when the size of the space (i.e., D) is not too large and the data are not necessarily extremely sparse. Traditionally, minwise hashing has been used as a data size/dimensionality reduction tool, typically for very large D (e.g., 2^{64}). Readers perhaps have noticed that, in our paper, (b-bit) minwise hashing could be utilized as a *data expansion* tool in order to apply efficient linear algorithms. When D is not very large, many aspects of the algorithms such as pseudo-random number generation would be quite different and new research may be necessary.

7 CONCLUSION

Current popular hashing methods, such as random projections and variants, often focus on approximating inner products and large-scale linear classifiers (e.g., linear SVM). However, linear kernels often do not achieve good performance. In this paper, we propose two types of nonlinear CoRE kernels which outperform linear kernels, sometimes by a large margin, on sparse non-binary data (which are common in practice). Because CoRE kernels are nonlinear, we accordingly develop basic hash methods to approximate CoRE kernels with linear kernels. The hashed data can be fed into highly efficient linear classifiers. Our experiments confirm the findings. We expect this work will inspire a new line of research on kernel learning, hashing algorithms, and large-scale learning.

ACKNOWLEDGEMENT

The research of Ping Li is partially supported by NSF-III-1360971, NSF-Bigdata-1419210, ONR-N00014-13-1-0764, and AFOSR-FA9550-13-1-0137.

A Proof of Theorem 1

To compute the expectation and variance of the estimator $\hat{K}_{C,1} = \frac{1}{k}\sum_{j=1}^{k} P_j(u)P_j(v)1\{L_j(u) = L_j(v)\}$, we need the first two moments of $P_j(u)P_j(v)1\{L_j(u) = L_j(v)\}$. The first moment is

$$\begin{aligned}&E\left[P_j(u)P_j(v)1\{L_j(u) = L_j(v)\}\right]\\ =&E\left[P_j(u)P_j(v)\right]\mathbf{Pr}\left(L_j(u) = L_j(v)\right) = \rho R\end{aligned}$$

which implies that $E\left(\hat{K}_{C,1}\right) = K_{C,1} = \rho R$. The second moment is

$$\begin{aligned}&E\left[P_j^2(u)P_j^2(v)1\{L_j(u) = L_j(v)\}\right]\\ =&E\left[P_j^2(u)P_j^2(v)\right]\mathbf{Pr}\left(L_j(u) = L_j(v)\right)\\ =&\left(1+2\rho^2\right)\rho R\end{aligned}$$

Here, we have used the result in the prior work [21]: $E\left[P_j^2(u)P_j^2(v)\right] = 1 + 2\rho^2$. Therefore, the variance is

$$Var\left(\hat{K}_{C,1}\right) = \frac{1}{k}\left\{\left(1+2\rho^2\right)R - \rho^2 R^2\right\}$$

This completes the proof.

B Proof of Theorem 2

We need the first two moments of the estimator $\hat{K}_{C,2} = \frac{1}{k}\sum_{j=1}^{k} V_j(u)V_j(v)1\{L_j(u) = L_j(v)\}\sqrt{f_1 f_2}$

Because

$$\begin{aligned}&E\left[V_j(u)V_j(v)1\{L_j(u) = L_j(v)\}\right]\\ =&E\left[V_j(u)V_j(v)1\{L_j(u) = L_j(v)\}|L_j(u) = L_j(v)\right]\\ &\times\mathbf{Pr}\left(L_j(u) = L_j(v)\right)\\ =&\frac{\sum_{i=1}^{D} u_i v_i}{a}R = \rho\frac{1}{f_1+f_2-a}\end{aligned}$$

we know

$$E\left(\hat{K}_{C,2}\right) = \frac{1}{k}\sum_{j=1}^{k}\rho\frac{\sqrt{f_1 f_2}}{f_1+f_2-a} = K_{C,2}$$

and

$$\begin{aligned}&E\left[V_j^2(u)V_j^2(v)1\{L_j(u) = L_j(v)\}\right]\\ =&E\left[V_j^2(u)V_j^2(v)\right]\mathbf{Pr}\left(L_j(u) = L_j(v)\right)\\ =&\frac{\sum_{i=1}^{D} u_i^2 v_i^2}{a}R = \frac{\sum_{i=1}^{D} u_i^2 v_i^2}{f_1+f_2-a}\end{aligned}$$

Therefore,

$$\begin{aligned}&Var\left(\hat{K}_{C,2}\right)\\ =&\frac{1}{k}\frac{f_1 f_2}{f_1+f_2-a}\left(\sum_{i=1}^{D} u_i^2 v_i^2 - \frac{\left(\sum_{i=1}^{D} u_i v_i\right)^2}{(f_1+f_2-a)}\right)\end{aligned}$$

This completes the proof.

References

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

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

[3] A. Z. Broder. On the resemblance and containment of documents. In *the Compression and Complexity of Sequences*, pages 21–29, Positano, Italy, 1997.

[4] G. Buehrer and K. Chellapilla. A scalable pattern mining approach to web graph compression with communities. In *WSDM*, pages 95–106, Stanford, CA, 2008.

[5] T. Chandra, E. Ie, K. Goldman, T. L. Llinares, J. McFadden, F. Pereira, J. Redstone, T. Shaked, and Y. Singer. Sibyl: a system for large scale machine learning. Technical report, 2010.

[6] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, Montreal, Quebec, Canada, 2002.

[7] F. Chierichetti, R. Kumar, S. Lattanzi, M. Mitzenmacher, A. Panconesi, and P. Raghavan. On compressing social networks. In *KDD*, pages 219–228, Paris, France, 2009.

[8] 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.

[9] D. Fetterly, M. Manasse, M. Najork, and J. L. Wiener. A large-scale study of the evolution of web pages. In *WWW*, pages 669–678, Budapest, Hungary, 2003.

[10] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of ACM*, 42(6):1115–1145, 1995.

[11] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *WWW*, pages 381–390, Madrid, Spain, 2009.

[12] M. R. Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *SIGIR*, pages 284–291, 2006.

[13] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, Dallas, TX, 1998.

[14] N. Jindal and B. Liu. Opinion spam and analysis. In *WSDM*, pages 219–230, Palo Alto, California, USA, 2008.

[15] T. Joachims. Training linear svms in linear time. In *KDD*, pages 217–226, Pittsburgh, PA, 2006.

[16] 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.

[17] P. Li. Abc-boost: Adaptive base class boost for multi-class classification. In *ICML*, pages 625–632, Montreal, Canada, 2009.

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

[19] P. Li and K. W. Church. Using sketches to estimate associations. In *HLT/EMNLP*, pages 708–715, Vancouver, BC, Canada, 2005.

[20] P. Li, K. W. Church, and T. J. Hastie. Conditional random sampling: A sketch-based sampling technique for sparse data. In *NIPS*, pages 873–880, Vancouver, BC, Canada, 2006.

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

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

[23] M. Najork, S. Gollapudi, and R. Panigrahy. Less is more: sampling the neighborhood graph makes salsa better and faster. In *WSDM*, pages 242–251, Barcelona, Spain, 2009.

[24] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *ICML*, pages 807–814, Corvalis, Oregon, 2007.

[25] S. Tong. Lessons learned developing a practical large scale machine learning system. http://googleresearch.blogspot.com/2010/04/lessons-learned-developing-practical.html, 2008.

[26] T. Urvoy, E. Chauveau, P. Filoche, and T. Lavergne. Tracking web spam with html style similarities. *ACM Trans. Web*, 2(1):1–28, 2008.

[27] J. Wang, J. Yang, K. Yu, F. Lv, T. S. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, pages 3360–3367, San Francisco, CA, 2010.

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

[29] K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. In *NIPS*, Vancouver, BC, Canada, 2009.