Ann. Data. Sci. (2016) 3(4):445–468
DOI 10.1007/s40745-016-0091-y

A Systematic Review on Minwise Hashing Algorithms

Jingjing Tang[1] · **Yingjie Tian**[2]

Received: 8 August 2016 / Revised: 28 August 2016 / Accepted: 29 September 2016 /
Published online: 26 October 2016

Abstract Similarity detection technology captures a host of researchers' attention. Minwise hashing schemes become the current researching hot spots in machine learning for similarity preservation. During the data preprocessing stage, the basic idea of minwise hashing schemes is to transfer the original data into binary codes which are good proxies of original data to preserve the similarity. Minwise hashing schemes can improve the computation efficiency and save the storage space without notable loss of accuracy. Thus, they have been studied extensively and developed rapidly for decades. Considering minwise hashing algorithm and its variants, a systematic survey is needed and beneficial to understand and utilize this kind of data preprocessing techniques more easily. The purpose of this paper is to review minwise hashing algorithms in detail and provide an insightful understanding of current developments. In order to show the application prospect of the minwise hashing algorithms, various algorithms have combined with linear Support Vector Machine for large-scale classification. Both theoretical analysis and experimental results demonstrate that these algorithms can achieve massive advantages in accuracy, efficiency and energy-consumption. Furthermore, their limitations, major opportunities and challenges, extensions and variants as well as potential important research directions have been pointed out.

Keywords Minwise hashing · Similarity estimation · Large-scale · Linear SVM

✉ Yingjie Tian
tyj@ucas.ac.cn

Jingjing Tang
tangjingjing13@mails.ucas.ac.cn

[1] School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China

[2] Research Center on Fictitious Economy and Data Science, Chinese Academy of Sciences, Beijing 100190, China

1 Introduction

With the booming development of information acquisition technology, we have witnessed an explosive growth in the scale of shared data collections that are characterized by a set of relevant features and represented as high-dimensional points. Since many similar information existing in these data collections highly consumes resources of index, a fundamental task is to examine data for similar items [1] in the real-world applications such as duplicated web page removal [2,3], advertising diversification [4], wireless sensor networks [5], graph sampling [6] and web spam [7]. When we take the distance metric between all data points as the similarity measurement, the storage and computation requirements are rigorous. Thus, the compact storage and efficient distance computation are indispensable for data representations.

Binary codes [8,9] are attractive data representations for search and retrieval purposes due to their efficiency in computation and storage capacity. The hashing technique [10–16] is a common method for assigning binary codes to data points. The binary codes serve as hash keys which are learned from the hash functions to preserve some notion of similarity in the original feature space. Suitable binary codes should maintain the general hash property of high collision probabilities for similar items. Furthermore, the storage needed to store the binary codes will be greatly decreased. The existing hashing methods can be mainly divided into two categories [13,17,18]: data-independent methods and data-dependent methods.

Representative data-independent methods include locality-sensitive hashing (LSH) [10,11,19] and its extensions [20–22], and shift invariant kernel hashing (SIKH) [23]. These data-independent methods need longer codes to achieve satisfactory performance [17], which will be inefficient due to the higher storage and computational cost. In contrast, data-dependent methods learn hash functions from the training data to overcome the shortcomings of the data-independent methods. Typical methods include Semantic hashing [24], Spectral hashing (SH) [25], Binary reconstruction embedding (BRE) [21], Semi-supervise hashing (SSH) [26], Self-taught hashing [27], Composite hashing [28], Minimal loss hashing (MLH) [29] and Iterative quantization (ITQ) [17].

In recent decades, approximation distance computation [30,31] for similarity searching is proposed by several researchers to avoid the running time bottleneck. This is due to the fact that approximate nearest neighbor is almost as good as the exact one in many cases. Locality-sensitive hashing (LSH) [22] was introduced as an approximate high-dimensional similarity search scheme with provably sub-linear time complexity dependence on the data size. The key idea is to compute randomized hash functions that guarantee a high probability of collision for similar samples. Then, one can determine near neighbors by hashing the query sample and retrieving the elements stored in the buckets containing that sample.

Minwise hashing is a standard algorithm for estimating the sets similarities. Then, several variants, i.e. b-bit minwise hashing [32–37], connected bit minwise hashing [38], f-fractional bit minwise hashing [39] and one permutation hashing [40], appear from diverse perspectives to improve the original minwise hashing algorithm for better performance.

While original minwise hashing method stores each hashed value using 40 bits [2] or 64 bits [41], the b-bit minwise hashing algorithm [33–36] stores the low-

est b bits of each hashed value and gains substantial advantages in storage space and computational efficiency for large-scale machine learning problems. To improve the effectiveness of document similarity detection, connected bit minwise hashing algorithm [38] is proposed for computing set similarities among high-dimensional vectors. As the extension of b-bit minwise hashing, f-fractional bit minwise hashing algorithm [39] is presented for a wider range of selectivity on accuracy and storage space requirements. The feasibility of f-fractional bit minwise hashing algorithm has investigated associated with the optimal fractional bit that makes the minimum variances estimator. Instead of $k(k \geq 100)$ permutations of existing minwise hashing algorithms, one permutation hashing algorithm merely applies one permutation, which not only preferably keeps the structure of original data set, but also avoids the disadvantages of the expensive preprocessing cost and the loss of classification accuracy.

Furthermore, experimental results in [42,43] have shown the effectiveness of these minwise hashing algorithms on large-scale data sets. Thus, these algorithms have been studied extensively and developed rapidly for decades. Considering minwise hashing algorithm and its variants, a systematic survey is needed and beneficial to understand and utilize this style of data preprocessing techniques more easily. The purpose of this paper is to review minwise hashing algorithms in detail and provide an insightful understanding of current developments. In order to show the application prospect of the minwise hashing algorithms, we combine various algorithms with linear Support Vector Machine (SVM) [44] in classification for large-scale data set. Both theoretical analysis and experimental results demonstrate that these algorithms can achieve massive advantages in accuracy, efficiency and energy-consumption. Furthermore, their limitations, major opportunities and challenges, as well as potential important research directions have been pointed out.

Section 2 of the paper reviews Locality-sensitive Hashing (LSH) and five representative minwise hashing algorithms, i.e. minwise hashing [2,3], b-bit minwise hashing [33–36], connected bit minwise hashing [38], f-fractional bit minwise hashing [39] and one permutation hashing [40]. Section 3 describes applications of these minwise hashing algorithms for large-scale classification problems. Section 4 introduces several extensions and variants of minwise hashing algorithms. Finally, concluding remarks and future research directions are provided in Sect. 5.

2 Minwise Hashing Algorithms

2.1 Locality-Sensitive Hashing

We begin by briefly reviewing Locality-sensitive Hashing (LSH). A binary hashing function defines a mapping $h(x)$ from R_d to the discrete set $\{0, 1\}$. In practice, each sample will be fed into a number of independent random hashing functions.

LSH is a special type of hashing algorithms with the locality-sensitive property, which basically states that if two samples are similar in the original feature space, their corresponding hash codes shall be alike. A key notion to quantify this property is collision probability, which is defined as $Pr(h(x_1) = h(x_2))$ (the expectation of

identical hash bit over all possible hashing functions). The locality-sensitive property also implies that the collision probability should be monotonically increasing with respect to the pairwise data similarity.

The basic idea behind locality-sensitive hashing (LSH) is to project the data into a low-dimensional binary space. Each data point is mapped to a l-bit binary vector termed as the hash key. With proper hash projections, the approximate nearest neighbors can be found in sublinear time with respect to the size of training samples. Define an LSH family $\mathcal{F}$ for a metric space $\mathcal{M} = (M, d)$, a threshold $R > 0$ and an approximation factor $c > 1$. $\mathcal{F}$ is a family of functions $h : \mathcal{M} \to \mathcal{S}$ satisfying the following conditions for any two samples $x_1, x_2 \in \mathcal{M}$ and a function h chosen uniformly at random from $\mathcal{F}$:

- if $d(x_1, x_2) \leq R$, then $h(x_1) = h(x_2)$ (i.e., p and q collide) with probability at least P_1,
- if $d(x_1, x_2) \geq cR$, then $h(x_1) = h(x_2)$ with probability at most P_2.

A family is interesting when $P_1 > P_2$. Such a family $\mathcal{F}$ is called (R, cR, P_1, P_2)-sensitive. For any sample, the hash key is constructed by applying l binary-valued hash functions, i.e., $h_1, \ldots, h_l$, from the LSH family to the sample.

2.2 Minwise Hashing

Minwise hashing [2,3] is a basic algorithmic tool for real-world problems related to set similarity and containment. Computing the size of set intersection is a crucial problem in information retrieval and machine learning. Minwise hashing applies the idea of Monte Carlo method, which transforms the problem of computing the size of set intersections into the probability of one case occurs. Due to large numbers of the permutations k, one can estimate the occurred probability of the case to achieve the document resemblance. It is worth noting that minwise hashing mainly works well with binary data represented via w-shingle, which can be viewed either as 0/1 vectors in high-dimension or as a set.

Consider two sets S_1 and S_2 where $S_1, S_2 \subseteq \Omega = \{0, 1, \ldots, D-1\}$. A generally used measurement of similarity is the resemblance

$$R = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = \frac{a}{f_1 + f_2 - a}, \tag{1}$$

where $f_1 = |S_1|$, $f_2 = |S_2|$, $a = |S_1 \bigcap S_2|$.

Applying a random permutation π on two sets S_1 and S_2 and storing the smallest elements under π as $\min(\boldsymbol{\pi}(S_1))$ and $\min(\boldsymbol{\pi}(S_2))$, the collision probability is simply

$$P_r(\min(\pi(S_1)) = \min(\pi(S_2))) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}. \tag{2}$$

Then, repeat the permutation k times to estimate R without bias as the following binomial probability associated with its variance:

$$\hat{R}_M = \frac{1}{k}\sum_{j=1}^{k} 1\{\min\left(\pi_j\left(S_1\right)\right) = \min\left(\pi_j\left(S_2\right)\right)\},$$
$$Var(\hat{R}_M) = \frac{1}{k}R(1-R). \tag{3}$$

The common practice is to store each hashed value, e.g., $\min(\pi(S_1))$ and $\min(\pi(S_2))$, using 40 bits [2] or 64 bits [41]. The cost of storage and computation will be formidable in large-scale application [45]. In order to demonstrate the comprehensive promotion degree with respect to the estimations of variance, storage space and sample size, the storage-factor for minwise hashing can be constructed with 64 bits as follows:

$$M(R) = 64 \times Var(\hat{R}_M) \times k,$$
$$= 64R(1-R). \tag{4}$$

However, it is worth noting that the storage-factor cannot be regarded as the measurement of precision. Moreover, minwise hashing requires the storage space of $64mk$ bits, where m is the size of data set and 64 is the bit number for each hashed binary code.

2.3 b-Bit Minwise Hashing

For the sake of computing resemblances, it is costly in time, storage space and energy-consumption for large-scale applications. Recently, the development of b-bit minwise hashing [33–36] recommend storing only the lowest b bits instead of 40 bits [2,3] or 64 bits [41] of each hashed value. By only storing b bits, b-bit minwise hashing gains substantial advantages in terms of storage space and the speed of computation.

Given two sets $S_1, S_2 \subseteq \Omega = \{0, 1, \ldots, D-1\}$. Define the minimum hashed values of S_1 and S_2 under a random permutation π to be $Z_1 = \min(\pi(S_1))$ and $Z_2 = \min(\pi(S_2))$; $Z_1^{(b)}$ ($Z_2^{(b)}$) the lowest b bits for the hashed value Z_1 (Z_2). Define $e_{1,i}$ ($e_{2,i}$) the i-th lowest bit for $Z_1^{(b)}$ ($Z_2^{(b)}$). Theorem 1 [33] provides the main probability formulations. Its proof assumes that D is large and the random permutation repeats many times, which is invariably satisfied in practice.

Theorem 1 *Assume D is large.*

$$P_b = \Pr\left(Z_1^{(b)} = Z_2^{(b)}\right)$$
$$= \Pr\left(\prod_{i=1}^{b} 1\{e_{1,i} = e_{2,i}\} = 1\right)$$
$$= C_{1,b} + (1 - C_{2,b})R, \tag{5}$$
$$r_1 = \frac{f_1}{D}, r_2 = \frac{f_2}{D}, f_1 = |S_1|, f_2 = |S_2|, \tag{6}$$

$$C_{1,b} = A_{1,b}\frac{r_2}{r_1+r_2} + A_{2,b}\frac{r_1}{r_1+r_2}, \tag{7}$$

$$C_{2,b} = A_{1,b}\frac{r_1}{r_1+r_2} + A_{2,b}\frac{r_2}{r_1+r_2}, \tag{8}$$

$$A_{1,b} = \frac{r_1[1-r_1]^{2^b-1}}{1-[1-r_1]^{2^b}}, \tag{9}$$

$$A_{2,b} = \frac{r_2[1-r_2]^{2^b-1}}{1-[1-r_2]^{2^b}}. \tag{10}$$

For a fixed r_j(where $j \in \{1, 2\}$), $A_{j,b}$ is a monotonically decreasing function of $b = 1, 2, 3 \ldots$.

For a fixed b, $A_{j,b}$ is a monotonically decreasing function of $r_j \in [0, 1]$ reaching a limit: $\lim_{r_j \to 0} A_{j,b} = \frac{1}{2^b}$.

From Theorem 1, the desired probability equation (5) is determined by R and the ratios $r_1 = \frac{f_1}{D}$ and $r_2 = \frac{f_2}{D}$ for a fixed b. Meanwhile, $A_{j,b}$ converges to zero in a rapid speed with the increasing of b. $A_{j,b}$ is closed to zero when $b \geq 32$. If $R = 1$, then $P_b = 1$ since we have $r_1 = r_2$ and $C_{1,b} = C_{2,b}$ in this case.

Theorem 1 suggests an unbiased estimator $\hat{R}_b$ for R:

$$\hat{R}_b = \frac{\hat{P}_b - C_{1,b}}{1 - C_{2,b}}, \hat{P}_b = \frac{1}{k}\sum_{j=1}^{k}\left\{\prod_{i=1}^{b} 1\left\{e_{1,i,\pi_j} = e_{2,i,\pi_j}\right\} = 1\right\}, \tag{11}$$

where $e_{1,i,\pi_j}(e_{2,i,\pi_j})$ is i-th lowest bit of Z_1 (Z_2) under the permutation π_j. The variance is

$$Var(\hat{R}_b) = \frac{1}{k}\frac{\left[C_{1,b} + \left(1 - C_{2,b}\right)R\right]\left[1 - C_{1,b} - \left(1 - C_{2,b}\right)R\right]}{\left(1 - C_{2,b}\right)^2} \tag{12}$$

From (3) and (11), $\hat{R}_b$ converges to the variance of $\hat{R}_M$ for large b, namely $\lim_{b\to\infty} Var(\hat{R}_b) = Var(\hat{R}_M)$. In fact, for the purpose of practice, $Var(\hat{R}_b)$ and $Var(\hat{R}_M)$ are numerically indistinguishable when b is 64 bits. Intuitively, compared to (12), at the same sample size k, the estimation variance will be increasing when we use fewer bits per sample so that the accuracy is contaminated. Hence, we need increase the value of k to maintain the same accuracy. In brief, b-bit minwise hashing not only preferably improves the accuracy, but also significantly reduces the storage and computational requirements.

With the same size k, the space required for storing each sample will be smaller with the decrease of b. Unfortunately, the estimated variance (12) will increase according to the b-bit minwise hashing theory. The storage factor $B(b; R, r_1, r_2)$ [33–35] is proposed to accurately measure the variance-space trade-off as follows:

$$B(b; R, r_1, r_2) = b \times Var\left(\hat{R}_b\right) \times k,$$
$$= \frac{b\left[C_{1,b} + \left(1 - C_{2,b}\right) R\right]\left[1 - C_{1,b} - \left(1 - C_{2,b}\right) R\right]}{\left(1 - C_{2,b}\right)^2}. \quad (13)$$

From that we know the lower $B(b; R, r_1, r_2)$ is better. The ratio $\frac{B(b_1; R, r_1, r_2)}{B(b_2; R, r_1, r_2)}$ measures the improvement of using $b = b_2$ over using $b = b_1$. Moreover, b-bit minwise hashing requires the storage space of bmk bits, where m is the size of data set.

2.4 Connected Bit Minwise Hashing

Based on the b-bit minwise hashing theory, connected bit minwise hashing [38] is proposed which connects bits obtained by b-bit minwise hashing algorithm. As an efficient and feasible method for similarity estimation, it can greatly reduce the number of comparisons so that the efficiency of similarity estimation is improved merely with a minor loss of accuracy. Furthermore, connected bit is convenient to be built and the performance increases with a far-reaching practical significance in large-scale data environment.

Again, consider two sets $S_1, S_2 \subseteq \Omega = \{0, 1, 2, \ldots, D-1\}$ and a random permutation group $\boldsymbol{\pi}$, where $\boldsymbol{\pi} = \{\pi_1, \pi_2, \ldots, \pi_k\}$, $\pi_j : \Omega \to \Omega$ and $j \in \{1, 2, \ldots, k\}$. Define the minimum hashed value under $\boldsymbol{\pi}$ to be $\boldsymbol{Z_h} = \min(\boldsymbol{\pi}(S_h))$; $\boldsymbol{Z_h^{(b)}}$ the lowest b bits for each hashed value of $\boldsymbol{Z_h}$; e_{h,i,π_j} the i-th lowest bit for $\boldsymbol{Z_h^{(b)}}$ under the permutation π_j and $h \in \{1, 2\}$. Experimental number k corresponds to k random independent permutations. For the connected bits b and the connected number n, define the variables:

$$\begin{aligned} x_1 &= e_{1,1,\pi_1} e_{1,2,\pi_1} \cdots e_{1,b,\pi_1} e_{1,1,\pi_2} e_{1,2,\pi_2} \cdots e_{1,b,\pi_2} e_{1,1,\pi_3} \\ &\quad \cdots e_{1,b,\pi_3} \cdots e_{1,1,\pi_n} \cdots e_{1,b,\pi_n}, \\ x_2 &= e_{2,1,\pi_1} e_{2,2,\pi_1} \cdots e_{2,b,\pi_1} e_{2,1,\pi_2} e_{2,2,\pi_2} \cdots e_{2,b,\pi_2} e_{2,1,\pi_3} \\ &\quad \cdots e_{2,b,\pi_3} \cdots e_{2,1,\pi_n} \cdots e_{2,b,\pi_n} \end{aligned}$$

If and only if $e_{1,1,\pi_j} \cdots e_{1,b,\pi_j} = e_{2,1,\pi_j} \cdots e_{2,b,\pi_j}$, where $\pi_j \in \hat{\boldsymbol{\pi}} = \{\pi_1, \pi_2, \cdots, \pi_n\}$, we obtain $x_1 = x_2$. Note that the permutation group $\hat{\boldsymbol{\pi}}$ consists of n permutations selected from a k-size sequential random independent permutation group $\boldsymbol{\pi}$ retaining its original order ($k \gg n$).

Compute $G_{b,n} = P_r(x_1 = x_2) = P_b^n = [C_{1,b} + (1 - C_{2,b})R]^n$. Then, an unbiased estimator $\hat{R}_{b,n}$ for R from k independent permutations can be obtained as follows:

$$\hat{R}_{b,n} = \frac{\hat{G}_{b,n}^{\frac{1}{n}} - C_{1,b}}{1 - C_{2,b}}, \quad (14)$$

$$\hat{G}_{b,n} = \frac{\sum_{j=1}^{\lfloor \frac{k}{n} \rfloor} \left\{ \prod_{i=1}^{n} 1 \left\{ \begin{matrix} e_{1,1,\pi_{n(j-1)+i}} \ldots e_{1,b,\pi_{n(j-1)+i}} \\ = e_{2,1,\pi_{n(j-1)+i}} \ldots e_{2,b,\pi_{n(j-1)+i}} \end{matrix} \right\} = 1 \right\}}{\lfloor \frac{k}{n} \rfloor}. \tag{15}$$

When $n = 1$, namely $G_{b,n} = G_{b,1} = P_b$, we obtain that the resemblance estimator $\hat{R}_b$ of b-bit minwise hashing is the special case for $\hat{R}_{b,n}$. Following the property of binomial distribution and the delta method [46] in statistics, the variance of $\hat{R}_{b,n}$ is

$$Var(\hat{R}_{b,n}) = \frac{1}{k} \times \frac{G_{b,n}\left(1 - G_{b,n}\right)}{\left(1 - C_{2,b}\right)^2 \times n G_{b,n}^{\frac{2(n-1)}{n}}}. \tag{16}$$

It can be derived that the variance of connected bit minwise hashing is larger than that of b-bit minwise hashing which has some slight effects on accuracy. Fortunately, the connected bit minwise hashing can greatly reduce the number of comparisons and preferably improve the performance.

Similarly, the storage factor $G(b, n; R, r_1, r_2)$ for connected bit minwise hashing [38] quantifies the variance-space trade-off as follows:

$$\begin{aligned} G(b, n; R, r_1, r_2) &= b \times n \times \left(\frac{k}{n}\right) \times Var(\hat{R}_{b,n}), \\ &= \frac{b G_{b,n}\left(1 - G_{b,n}\right)}{\left(1 - C_{2,b}\right)^2 \times n G_{b,n}^{\frac{2(n-1)}{n}}}. \end{aligned} \tag{17}$$

From that, we know the lower $G(b, n; R, r_1, r_2)$ is better. Although connected bit minwise hashing has the same storage space as b-bit minwise hashing with bmk bits, where m is the size of data set, the comparisons to achieve its resemblance has substantially reduced. However, it is worth noting that the storage-factor only demonstrates the degree of comprehensive promotion of the estimation variance, storage space and sample size but cannot indicate the effect of precision.

2.5 f-Fractinal Bit Minwise Hashing

Although the higher integer bit value decreases the variance estimator and improves the accuracy, it will consume lots of time. Furthermore, the integer bit determined by b-bit minwise hashing method could not meet fine-grained requirements in terms of accuracy, computational efficiency and storage space. Thus, f-fractional bit minwise hashing algorithm [39] is proposed for a wider range of selectivity on these aspects. The key idea of this algorithm is the continuous selectivity of bit instead of the discrete integer value represented by a linear combination of bits obtained by b-bit minwise hashing method. Correspondingly, the similarity and the optimal fractional bit that makes the minimum variance estimator can be estimated. F-fractional bit minwise hashing algorithm not only enriches the theoretical system of b-bit minwise hashing

algorithm, but also satisfies the various needs of accuracy and storage space in the practical system.

Consider two sets $S_1, S_2 \subseteq \Omega = \{0, 1, 2, \cdots, D-1\}$ and a random permutation group $\boldsymbol{\pi}$, where $\boldsymbol{\pi} = \{\pi_1, \pi_2, \cdots, \pi_k\}$, $\pi_j : \Omega \to \Omega$ and $j \in \{1, 2, \cdots, k\}$. Define the minimum hashed value under $\boldsymbol{\pi}$ as $\boldsymbol{Z_h} = \min(\boldsymbol{\pi}(S_h)) \in R^k$; the lowest b_l bits for each dimension of $\boldsymbol{Z_h}$ as $\boldsymbol{Z_h^{(b_l)}} \in R^k$; the i-th lowest bit of the j-th dimension for $\boldsymbol{Z_h^{(b_l)}}$ as e_{h,i,π_j}, where $h \in \{1, 2\}$ and $i \in \{1, 2, \cdots, b_l\}$.

Let b_1, b_2 be integer bit with $b_1 < b_2$ and w_l be the proportion of b_l with $w_l = \frac{k_l}{k}$, where $l \in \{1, 2\}$ and $k_1 + k_2 = k$.

Define f:

$$f = w_1 b_1 + w_2 b_2, w_1 + w_2 = 1, \quad b_1 < b_2. \tag{18}$$

Define the j-th dimensional value of $\boldsymbol{Z_1^{(b_l)}}$ and $\boldsymbol{Z_2^{(b_l)}}$ as the following variables respectively: $X_{1,j} = e_{1,1,\pi_j} e_{1,2,\pi_j} \cdots e_{1,b_l,\pi_j}$ and $X_{2,j} = e_{2,1,\pi_j} e_{2,2,\pi_j} \cdots e_{2,b_l,\pi_j}$, where $b_l \in \{b_1, b_2\}$ and $\pi_j \in \{\pi_1, \pi_2, \cdots, \pi_k\}$.

Then

$$\begin{aligned} P_f &= P_r(X_{1,j} = X_{2,j}) \\ &= P_r(b = b_1) P_r\left(\prod_{i=1}^{b_1} 1\left\{e_{1,i,\pi_j} = e_{2,i,\pi_j}\right\} = 1\right) \\ &+ P_r(b = b_2) P_r\left(\prod_{i=1}^{b_2} 1\left\{e_{1,i,\pi_j} = e_{2,i,\pi_j}\right\} = 1\right) \\ &= w_1\left[C_{1,b_1} + (1 - C_{2,b_1}) R_f\right] + w_2\left[C_{1,b_2} + (1 - C_{2,b_2}) R_f\right]. \end{aligned} \tag{19}$$

The unbiased estimator $\hat{R}_f$ for R_f from k independent permutations is:

$$\hat{R}_f = \frac{\hat{P}_f - (w_1 C_{1,b_1} + w_2 C_{1,b_2})}{1 - (w_1 C_{2,b_1} + w_2 C_{2,b_2})}, \tag{20}$$

$$\begin{aligned} \hat{P}_f = \frac{1}{k}\Bigg(&\sum_{j=1}^{w_1 k}\left\{\prod_{i=1}^{b_1} 1\left\{e_{1,i,\pi_j} = e_{2,i,\pi_j}\right\} = 1\right\} \\ &+ \sum_{j=1}^{w_2 k}\left\{\prod_{i=1}^{b_2} 1\left\{e_{1,i,\pi_{w_1 k + j}} = e_{2,i,\pi_{w_1 k + j}}\right\} = 1\right\}\Bigg). \end{aligned} \tag{21}$$

Note that $\hat{R}_b$ and $\hat{P}_b$ are special case of $\hat{R}_f$ and $\hat{P}_f$ with $w_1 = 0$, $w_2 = 1$ and $b_2 = 1$ or $w_1 = 1$, $w_2 = 0$ and $b_1 = 1$.

Following the property of binomial distribution and the delta method [46] in statistics, the variance of $\hat{R}_f$ is

$$Var(\hat{R}_f) = \frac{1}{k} \times \frac{{w_1}^2 P_{b_1}(1 - P_{b_1}) + {w_2}^2 P_{b_2}(1 - P_{b_2})}{\left[1 - (w_1 C_{2,b_1} + w_2 C_{2,b_2})\right]^2}, \tag{22}$$

where $P_{b_1} = C_{1,b_1} + (1 - C_{2,b_1})R_f$ and $P_{b_2} = C_{1,b_2} + (1 - C_{2,b_2})R_f$.

The variance decreases with a larger f-bit and lies between that of its proximate integer bits. In order to satisfying various accuracy and storage space requirements, diverse combination of b_1 and b_2 with respective w_1 and w_2 can be made to constitute fractional bit f. Similarly, the storage factor $F(w_1, w_2, b_1, b_2; R, r_1, r_2)$ for f-fractional bit minwise hashing [47] can be presented as:

$$\begin{aligned} F(w_1, w_2, b_1, b_2; R, r_1, r_2) &= f \times k \times Var(\hat{R}_f) \\ &= (w_1 b_1 + w_2 b_2) \\ &\quad \times \frac{{w_1}^2 P_{b_1}(1 - P_{b_1}) + {w_2}^2 P_{b_2}(1 - P_{b_2})}{\left[1 - (w_1 C_{2,b_1} + w_2 C_{2,b_2})\right]^2}. \end{aligned} \tag{23}$$

According to the accuracy and storage requirements, we select an appropriate fractional bit f. It is apparent that the variance is the decreasing function of bit which simultaneously determines the size of storage space. If f satisfies $b_1 < f < b_2$, we conclude that $Var(b_1) > Var(f) > Var(b_2)$ and $\{storage(b_1) < storage(f) < storage(b_2)\}$.

In fact, there exists multifarious combinations for any fixed fractional bit f. From (18), $w_1 = \frac{b_2 - f}{b_2 - b_1}$, $w_2 = \frac{f - b_1}{b_2 - b_1}$. For example, when choosing $f = 2.5$, $w_1 = 0.5$, $w_2 = 0.5$, $b_1 = 2$, $b_2 = 3$ or $w_1 = 0.75$, $w_2 = 0.25$, $b_1 = 2$, $b_2 = 4$ can be selected to compose f. For any given integer bit b_1 and b_2, $f = f_0$, $1 \le b_1 < f_0 < b_2 \le 32$, $Var(\hat{R}_f) = Var(b_1, b_2, f_0)$. Because the integer values b_1 and b_2 are discrete and limited, the number of the combinations for fractional bit f are finite. Thus, the variance under the numbered b_1 and b_2 can be calculated.

Although various combinations can be made for the fixed fractional bit f, the one which makes the minimum estimator of variance is optimal. Thus, the theoretical system of minwise hashing algorithms gets further development and ultimately attaches profound significance in large-scale data environment.

For (20), if $r_1 = \frac{f_1}{D} = r_2 = \frac{f_2}{D}$, then $C_{1,b_1} = C_{2,b_1} = C_{b_1}$ and $C_{1,b_2} = C_{2,b_2} = C_{b_2}$. In particular, C_{b_1} and C_{b_2} can be approximated at zero if b_1 and b_2 are large enough. In this case, the variance of $\hat{R}_{f_{opt}}$ can be yielded as

$$Var\left(\hat{R}_{f_{opt}}\right) = \frac{R_f(1 - R_f)\left[(b_2 - f)^2 + (f - b_1)^2\right]}{k(b_2 - b_1)^2}. \tag{24}$$

Computing the partial derivative of (24) on the components b_1, b_2 and f and let them be zero, $b_1 = f, b_2 = f$ and $f = \frac{b_1 + b_2}{2}$. Because b_1 and b_2 are integer bits, f is a fractional bit and $1 \le b_1 < f_0 < b_2 \le 32$, the optimal fractional bit is achieved with $b_1 = \lfloor f \rfloor$, $b_2 = \lceil f \rceil$, $w_1 = \frac{\lceil f \rceil - f}{\lceil f \rceil - \lfloor f \rfloor}$ and $w_2 = \frac{f - \lfloor f \rfloor}{\lceil f \rceil - \lfloor f \rfloor}$, which make the minimum estimator of variance as $Var(\lfloor f \rfloor, \lceil f \rceil, f)$. Thus, the formula of optimal fractional bit is $f_{opt} = \frac{\lceil f \rceil - f}{\lceil f \rceil - \lfloor f \rfloor} \times \lfloor f \rfloor + \frac{f - \lfloor f \rfloor}{\lceil f \rceil - \lfloor f \rfloor} \times \lceil f \rceil$ concerning the certain fractional bit f. Moreover, the storage factor of f_{opt} is

$$
\begin{aligned}
&F_{opt}\left(\frac{\lceil f\rceil - f}{\lceil f\rceil - \lfloor f\rfloor}, \frac{f-\lfloor f\rfloor}{\lceil f\rceil - \lfloor f\rfloor}, \lfloor f\rfloor, \lceil f\rceil; R, r_1, r_2\right)\\
&= f \times k \times Var(\hat{R}_f)\\
&= \left(\frac{\lceil f\rceil - f}{\lceil f\rceil - \lfloor f\rfloor}\lfloor f\rfloor + \frac{f-\lfloor f\rfloor}{\lceil f\rceil - \lfloor f\rfloor}\lceil f\rceil\right)\\
&\quad \times \frac{(\frac{\lceil f\rceil - f}{\lceil f\rceil - \lfloor f\rfloor})^2 P_{\lfloor f\rfloor}(1-P_{\lfloor f\rfloor}) + (\frac{f-\lfloor f\rfloor}{\lceil f\rceil - \lfloor f\rfloor})^2 P_{\lceil f\rceil}(1-P_{\lceil f\rceil})}{[1-(\frac{\lceil f\rceil - f}{\lceil f\rceil - \lfloor f\rfloor}C_{2,\lfloor f\rfloor} + \frac{f-\lfloor f\rfloor}{\lceil f\rceil - \lfloor f\rfloor}C_{2,\lceil f\rceil})]^2}.
\end{aligned}
\tag{25}
$$

Besides, f-fractional bit minwise hashing and the optima fractional bit minwise hashing require the storage space of fmk and $f_{opt}mk$ bits respectively, where m is the size of data set.

2.6 One Permutation Hashing

For the sake of similarity retrieval [1], the computation of similarity between sets is a core mission. Algorithms such as minwise hashing [2,3], b-bit minwise hashing [33–36], connected bit minwise hashing[38] and f-fractional bit minwise hashing [39] apply k independent random permutations on the entire data set leading to expensive cost of time, storage space and energy-consumption for similarities computation. However, one permutation hashing [40] merely utilizes one permutation without notable replacement of samples and preserves the matrix sparsity. By doing so, it not only preferably preserves the structure of original data set, but also avoids expensive preprocessing cost.

At first, the shingled binary data vector for each sample is viewed as a set consisting of the locations of the nonzero elements. Consider sets $S_i \subseteq \Omega = \{0, 1, 2, \cdots, D-1\}$, where D is the size of the space. One permutation hashing algorithm permutes each set once and convert it into a binary vector where 1 represents the new location of the nonzero element. Then, the D-dimensional binary vector is divided into r bins and the smallest nonzero element's location of each bin is stored for each data vector.

For the theoretical analysis, define the number of "jointly empty bins" and the number of "matched bins" as respectively:

$$
N_{emp} = \sum_{j=1}^{k} I_{emp,j}, \quad N_{mat} = \sum_{j=1}^{k} I_{mat,j}, \tag{26}
$$

where $I_{emp,j}$ and $I_{mat,j}$ are defined for the j-th bin, as

$$
I_{emp,j} = \begin{cases} 1 \text{ if both } \pi(S_1) \text{ and } \pi(S_2) \text{ are empty in the j} - \text{th bin} \\ 0 \; otherwise \end{cases} \tag{27}
$$

$$
I_{mat,j} = \begin{cases} 1 \text{ if both } \pi(S_1) \text{ and } \pi(S_2) \text{ are not empty and the smallest element} \\ \quad \text{of } \pi(S_1) \text{ matches the smallest element of } \pi(S_2), \text{ in the j} - \text{th bin} \\ 0 \; otherwise \end{cases} \tag{28}
$$

Recall the notation: $f_1 = |S_1|$, $f_2 = |S_2|$, $a = |S_1 \cap S_2|$. We also use $f = |S_1 \cap S_2| = f_1 + f_2 - a$.

Theorem 2 $\hat{R}_{mat} = \frac{N_{mat}}{k - N_{emp}}, \quad E(\hat{R}_{mat}) = R,$

$$Var\left(\hat{R}_{mat}\right) = R(1-R)\left(E\left(\frac{1}{k-N_{emp}}\right)\left(1+\frac{1}{f-1}\right) - \frac{1}{f-1}\right),$$

$$E\left(\frac{1}{k-N_{emp}}\right) = \sum_{j=0}^{k-1} \frac{Pr(N_{emp}=j)}{k-j} \geq \frac{1}{k-E(N_{emp})}.$$

The fact that $E(\hat{R}_{mat}) = R$ may seem surprising as in general ratio estimators are not unbiased. Note that $k - N_{emp} > 0$, because we assume the original data vectors are not completely empty (all-zero). As expected, when $k \ll f = f1 + f2 - a$, N_{emp} is essentially zero and hence $Var(\hat{R}_{mat}) \approx \frac{R(1-R)}{k}$. In fact, $Var(\hat{R}_{mat})$ is a bit smaller than $\frac{R(1-R)}{k}$, especially for large k.

It is probably not surprising that our one permutation scheme (slightly) outperforms the original k-permutation scheme (at merely $1/k$ of the preprocessing cost), because one permutation hashing, which is "sampling-without-replacement", provides a better strategy for matrix sparsification.

Actually, the hashed values are aligned identically since they are generated from the same permutation. From the perspective of energy-consumption, the number of permutations reduces from k to just one which is much more computationally efficient and makes the storage realizable. Moreover, in consideration of the scheme converting many nonzero elements into zero without the destruction of the original data set, it provides a relatively satisfactory strategy for sparsity. Furthermore, two strategies are adopted for dealing with the empty bins, i.e. the Zero Coding Scheme and the m-Permutation Scheme.

3 Applications of Minwise Hashing Algorithms

In this section, we show the combinations of various minwise hashing algorithms with linear support vector machine (SVM) for large-scale document classification [48]. Both the theoretical analysis and the experimental results demonstrate that this kind of combinations can achieve massive advantages in accuracy, efficiency and energy-consumption.

Document similarity detection technology [49–52] is an important topic in the information processing field, and it is a powerful tool to protect the author's intellectual property and to improve the efficiency of information retrieval. Generally, the problem of similarity computation is transformed into finding sets with a relatively large intersection by the technique known as "shingling". Representing documents as w-shingle sets generates lower-dimensional data sets. The value of w depends on the length of typical documents and the size of set with typical characters, where $w \geq 5$ in prior studies [2,41]. Since word frequency distributions with documents approximately follow a power-law [53] and most shingle terms occur rarely, it is reasonable and adequate to view w-shingle data sets as the sets of 0/1 (absence/presence) vectors in high-dimensional space. Compared with the decimal data, binary-quantized data performs well in experiments. In practice, the problem of insufficient memory capacity

often appears. Thus, minwise hashing algorithms are beneficial to make the original data represented compactly.

Connected bit minwise hashing [38], f-fractional bit minwise hashing [39] and one permutation hashing [40] extend b-bit minwise hashing [33–36] to improve the efficiency of similarity estimation without notable loss of accuracy. The hashed bit is convenient to be built and the performance increases with a strong practical significance in the environment of massive amounts of data.

This part compares the integrations of linear SVM with b-bit minwise hashing [35,36], connected bit minwise hashing [43], f-fractional bit minwise hashing [42] and one permutation hashing [40] practically. In theoretical analysis, the positive definiteness of resemblance matrices generated by these minwise hashing algorithms guarantees the converting of the nonlinear SVM problem [54] into linear SVM for large-scale classification. This property makes the kernel matrices decompose into matrices inner product to be the foundation for the integration as Theorems 5, 6 and 7 shown. Through solving large-scale linear SVM with many representative software packages such as LIBLINEAR [55], SVM^{perf} [56], SGD [57,58] and Pegasos [59], these combinations yield better performance and possess their own merits.

3.1 Integrating Linear SVM with Connected Bit Minwise Hashing Kernel

Connected bit minwise hashing algorithm can improve the efficiency of similarity estimation since the half of comparisons is greatly reduced without notable loss of accuracy. The positive definiteness of the resemblance matrices generated by connected bit minwise hashing and optimal fractional bit minwise hashing serve as the theoretical foundation for integrating linear SVM with them.

Definition 3 [36] A symmetric $m \times m$ matrix K satisfying $\sum_{i.j} c_i c_j K_{ij} \geq 0$ for all real vectors c is called positive definite (PD).

Lemma 4 [36] *Consider m sets $S_1, \cdots, S_m \in \Omega = \{0, 1, \cdots, D-1\}$. Apply one permutation π to each set. Define $Z_i = \min(\pi(S_i))$ and $Z_i^{(b)}$ be the lowest b bits of Z_i. The following three matrices are PD.*

1. *The resemblance matrix $R \in R^{m \times m}$, whose (i, j)-th entry is the resemblance between set S_i and set S_j: $R_{ij} = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} = \frac{|S_i \cap S_j|}{|S_i| + |S_j| - |S_i \cap S_j|}$.*
2. *The minwise hashing matrix $M \in R^{m \times m}$: $M_{ij} = 1\{Z_i = Z_j\}$.*
3. *The b-bit minwise hashing matrix $M^{(b)} \in R^{m \times m}$: $M_{ij}^{(b)} = 1\{Z_i^{(b)} = Z_j^{(b)}\}$.*

Consequently, consider k independent permutations and denote $M_{(s)}^{(b)}$ the b-bit minwise hashing matrix generated by the s-th permutation. Then, the summation $\sum_{s=1}^{k} M_{(s)}^{(b)}$ is also PD.

Theorem 5 [43] *Consider m sets* $S_1, \cdots, S_m \in \Omega = \{0, 1, \cdots, D-1\}$. *Apply one permutation group* $\boldsymbol{\pi}$ *of size n to each set* S_i. *Define the minimum value under* $\boldsymbol{\pi}$ *as* $\boldsymbol{Z_i} = \min(\boldsymbol{\pi}(S_i)) = \{\min(\pi_1(S_i)), \min(\pi_2(S_i)), \cdots, \min(\pi_n(S_i))\}$, $\boldsymbol{Z_i^{(b)}}$ *be the lowest b bits for each dimension of* $\boldsymbol{Z_i}$ *and* $\boldsymbol{Z_i}, \boldsymbol{Z_i^{(b)}} \in R^n$, ${Z_i}^{(n,b)}$ *with the length of nb derived from sequentially connecting n dimensions of* $\boldsymbol{Z_i^{(b)}}$, ${Z_i}^{(n,b)} \in R^1$ *and n the number of connected bits. The matrix generated by connected bit minwise hashing is PD.*

Instead of just one permutation group $\boldsymbol{\pi}$ consist of n permutations, we use k ($k \gg n$) permutations and thus there will be $\lfloor k/n \rfloor$ connected bit minwise hashing matrices. Define $M_{(s)}^{(n,b)}$ as a connected bit minwise hashing matrix generated by the s-th permutation group with the size of n, where $s = 1, 2, \cdots, \lfloor k/n \rfloor$. Note that the summation $\sum_{s=1}^{\lfloor k/n \rfloor} M_{(s)}^{(n,b)}$ is still PD, since $c^\top \Big[\sum_{s=1}^{\lfloor k/n \rfloor} M_{(s)}^{(n,b)} \Big] c = \sum_{s=1}^{\lfloor k/n \rfloor} c^\top M_{(s)}^{(n,b)} c \geq 0$ for arbitrary vector c by the reason that $M_{(s)}^{(n,b)}$ is PD.

Seemingly, the positive definiteness of $M_{(s)}^{(n,b)}$ is not beneficial enough for efficient linear SVM training since it is a nonlinear operation. A concise strategy can be provided to construct a matrix B_s to decompose the resemblance matrix as an inner product satisfying $M_{(s)}^{(n,b)} = B_s^\top B_s$, where B_s has dimensions $2^{nb} \times 2^{nb}$. It is high-effective to connect the value of b-bit minwise hashing. Meanwhile, the decomposition of resemblance matrix can be taken as the transformation of kernel matrix for SVM model.

Consider a data set $\{(x_i, y_i)\}_{i=1}^m$, where a binary data vector $x_i \in R^D$ and $y_i \in \{-1, 1\}$. Apply k random permutations on each feature vector x_i and store the lowest b bits of each hashed value. Then, connect every successive n b-bit and obtain a new data set using mbk bits in total. Later, expand each new data point into a $\lfloor k/n \rfloor$-dimensional vector and convert the above vector into a $2^{nb} \times \lfloor k/n \rfloor$-length vector with exactly $\lfloor k/n \rfloor$ 1's at run-time.

For example, suppose $k = 6$ and the original hashed values are $\{4, 3, 3, 3, 1, 1\}$ whose binary digits are $\{100, 011, 011, 011, 001, 001\}$. Consider $b = 1$. The binary digits are stored as $\{0, 1, 1, 1, 1, 1\}$. Then, set connected bit number $n = 2$ and connect consecutive 2 bits of the stored digits as $\{01, 11, 11\}$ corresponding to $\{1, 3, 3\}$ in decimals. At run-time, expand it into a 12-length ($2^{nb} \times \lfloor k/n \rfloor = 2^{2\times 1} \times \lfloor 6/2 \rfloor = 12$) vector, to be $\{0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0\}$ and it is a new feature vector applicable to the "LIBLINEAR" solver. The expansion is directly and reasonably based on Theorem 5.

3.2 Integrating Linear SVM with F-Fractional Bit Minwise Hashing Kernel

The positive definiteness of the resemblance matrices generated by f-fractional bit minwise hashing and optimal fractional bit minwise hashing has been proved as the theoretical foundation for integrating linear SVM with them.

Theorem 6 [42] *Consider m sets $S_1, \cdots, S_m \in \Omega = \{0, 1, \cdots, D-1\}$ and a permutation group $\boldsymbol{\pi}$ ($\boldsymbol{\pi} = \{\pi_1, \pi_2, \cdots, \pi_k\}$). Apply the permutation group $\boldsymbol{\pi}$ to each set. Define the minimum value under the permutation group $\boldsymbol{\pi}$ as $\boldsymbol{Z_h} = \min(\boldsymbol{\pi}(S_h))$; the lowest b_l bits for each dimension of $\boldsymbol{Z_h}$ as $\boldsymbol{Z_h^{(b_l)}}$ and $\boldsymbol{Z_h}$, $\boldsymbol{Z_h^{(b_l)}} \in R^k$, where $h \in \{1, 2\}$ and $b_l \in \{b_1, b_2\}$ Define $f = w_1 b_1 + w_2 b_2$, $w_1 + w_2 = 1, b_1, b_2$ be integer bit with $b_1 < b_2$ and w_l be the proportion of b_l with $w_l = \frac{k_l}{k}$, where $l \in \{1, 2\}$. Retaining the order of original permutation group, k_1 is corresponding to the former w_1 percent permutations and k_2 is corresponding to the latter w_2 percent permutations. Thus, define $\boldsymbol{Z^{(b_l)}_{(h;1,w_1k)}}$ be comprised of the former $w_1 k$ dimension of the k-dimensional vector $\boldsymbol{Z_h^{(b_l)}}$ and $\boldsymbol{Z^{(b_l)}_{(h;1,w_1k)}} \in R^{w_1 k}$; $\boldsymbol{Z^{(b_l)}_{(h;w_1k+1,k)}}$ be comprised of the latter $w_2 k$ dimension of the k-dimensional vector $\boldsymbol{Z_h^{(b_l)}}$ and $\boldsymbol{Z^{(b_l)}_{(h;w_1k+1,k)}} \in R^{w_2 k}$. Let $b_1 = \lfloor f \rfloor$, $b_2 = \lceil f \rceil$, $w_1 = \frac{\lceil f \rceil - f}{\lceil f \rceil - \lfloor f \rfloor}$ and $w_2 = \frac{f - \lfloor f \rfloor}{\lceil f \rceil - \lfloor f \rfloor}$ and thus the formula of optimal fractional bit combination is $f_{opt} = \frac{\lceil f \rceil - f}{\lceil f \rceil - \lfloor f \rfloor} \times \lfloor f \rfloor + \frac{f - \lfloor f \rfloor}{\lceil f \rceil - \lfloor f \rfloor} \times \lceil f \rceil$. The following two matrices are PD:*

1. *The f-fractional minwise hashing matrix $M^{(f)} \in R^{m \times m}$, where (i, j)-th entry is the resemblance between the sets S_i and S_j: $M^{(f)}_{ij} = 1\{\boldsymbol{Z^{(b_1)}_{(i;1,w_1k)}} = \boldsymbol{Z^{(b_1)}_{(j;1,w_1k)}}\} \times 1\{\boldsymbol{Z^{(b_2)}_{(i;w_1k+1,k)}} = \boldsymbol{Z^{(b_2)}_{(j;w_1k+1,k)}}\}$.*
2. *The optimal fractional minwise hashing matrix $M^{(f_{opt})} \in R^{m \times m}$, where (i, j)-th entry is the resemblance between the sets S_i and S_j: $M^{(f_{opt})}_{ij} = 1\{\boldsymbol{Z^{(\lfloor f \rfloor)}_{(i;1,w_1k)}} = \boldsymbol{Z^{(\lfloor f \rfloor)}_{(j;1,w_1k)}}\} \times 1\{\boldsymbol{Z^{(\lceil f \rceil)}_{(i;w_1k+1,k)}} = \boldsymbol{Z^{(\lceil f \rceil)}_{(j;w_1k+1,k)}}\}$, where $w_1 = \frac{\lceil f \rceil - f}{\lceil f \rceil - \lfloor f \rfloor}$.*

However, it seems that the PD of $M^{(f)}$ and $M^{(f_{opt})}$ are not beneficial enough for efficient linear SVM training since they are nonlinear operations. In spite of this, a concise strategy can be obtained to construct matrices B and C satisfying $M^{(f)} = B^\top B$ and $M^{(f_{opt})} = B^\top_{opt} B_{opt}$.

Consider a data set $\{(x_i, y_i)\}_{i=1}^m$, where $x_i \in R^D$ is a D-dimension binary data vector and $y_i \in \{-1, 1\}$. For a certain fractional bit f, choose k random permutations acting on each feature vector x_i and then store the lowest b_1 bits as $\boldsymbol{Z^{(b_1)}}$ and b_2 bits as $\boldsymbol{Z^{(b_2)}}$ of each binary hashed value. Then, select the former $w_1 k$ dimensions of $\boldsymbol{Z^{(b_1)}}$ and the latter $w_2 k$ dimensions of $\boldsymbol{Z^{(b_2)}}$ and combine them as a new k-dimensional vector retaining their original order. Thus, a new binary data set is achieved with fkm bits, where $f = w_1 b_1 + w_2 b_2$. Later, convert each new binary data point into a k-dimensional vector in decimals and expand the above vector into a $(2^{b_1} w_1 k + 2^{b_2} w_2 k)$-length vector with exactly k 1's at run-time.

For example, suppose $k = 6$ and the original hashed values are $\{4, 3, 3, 3, 1, 1\}$ whose binary digits are $\{100, 011, 011, 011, 001, 001\}$. Considering $f = 2.5$ with its optimal combination as $f_{opt} = \frac{\lceil f \rceil - f}{\lceil f \rceil - \lfloor f \rfloor} \times \lfloor f \rfloor + \frac{f - \lfloor f \rfloor}{\lceil f \rceil - \lfloor f \rfloor} \times \lceil f \rceil = \frac{1}{2} \times 2 + \frac{1}{2} \times 3$, thus, the binary digits are stored as $\{00, 11, 11, 011, 001, 001\}$ corresponding to $\{0, 3, 3, 3, 1, 1\}$ in decimals. At run-time, expand it into a 36-length $(2^{b_1} w_1 k + 2^{b_2} w_2 k = 2^2 \times \frac{1}{2} \times 6 + 2^3 \times \frac{1}{2} \times 6 = 36)$ vectors, to be $\{0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,$

0, 0, 1} and it is a new feature vector applicable to the "LIBLINEAR" solver. The expansion is directly and reasonably based on Theorem 6.

3.3 Integrating Linear SVM with One Permutation Hashing Kernel

Based on the assumption that the shingled data vectors are binary, relatively sparse and high-dimensional, the resemblance matrix generated by one permutation hashing can be regarded as the kernel matrix of SVM for large-scale text classification. The positive definiteness of the resemblance matrix provides a solid theoretical foundation for the integration. According to the superior decomposability of the positive definite resemblance matrix, the desired performance can be achieved.

Theorem 7 *Consider m sets $S_1, \cdots, S_m \in \Omega = \{0, 1, \cdots, D-1\}$. Apply one permutation hashing scheme and suppose the space Ω is divided evenly into r bins. Assume the number of zero elements is small compared to r and "$*$" represents empty bin coped with "zero coding" strategy. The one permutation hashing matrix $M^{(o)} \in R^{m \times m}$ is PD, where (x, y)-th entry is the resemblance between the sets S_x and S_y.*

Although $M^{(o)}$ is a nonlinear operation corresponding to the kernel matrix, the positive definiteness of it provides a concise strategy to construct a matrix B satisfying $M^{(o)} = B^\top B$ and ensures the rationality of the transformation from a non-linear operation into a linear operation. Thus, similar model of standard linear SVM can be yielded and solved by "LIBLINEAR".

Consider a data set $\{(x_i, y_i)\}_{i=1}^m$, where the data vector $x_i \in R^D$ and $y_i \in \{-1, 1\}$. Choose a random permutation acting on each feature vector x_i and view it as a binary vector, where "1" represents the new location of the existing elements. Divide the D-dimensional space evenly into r bins. In each bin for one data vector, select the location of the smallest nonzero element and represent it as a binary data. Then, store the lowest b bits of each binary value and obtain a new data set using rbm bits. Later, convert each new binary data into a r-dimensional vector in decimals and expand it into a $2^b r$-length vector with exactly r 1's at most."$*$" denotes the empty bin processed by "zero coding" strategy.

For example, suppose $D = 25$ and $r = 5$. The original hashed values under the permutation π are $\{2, 4, 7, 10, 12, 17, 19\}$. Take the smallest element for each bin and store them as $\{2, 7, 10, 17, *\}$, whose binary digits are $\{10, 111, 1010, 10001, *\}$. Select $b = 2$ and thus the binary digits are stored as $\{10, 11, 10, 01, *\}$ corresponding to $\{2, 3, 2, 1, *\}$ in decimals. At run-time, expand them into a 20-length ($2^b r = 2^2 \times 5 = 20$) vector with "zero coding" strategy to be $\frac{1}{\sqrt{5-1}}\{0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0\}$ and regard it as a new feature vector applied to linear SVM solver. The expansion is reasonably based on the positive definiteness of one permutation hashing matrix in Theorem 7. Moreover, binary data performs well compared with the original decimal data in the experiments. Therefore, the goal of compressing the original data set is realized for linear SVM training.

Table 1 Experimental results *accuracy* (*time*) of the BBMH SVM and CBMH SVM algorithms on *webspam*

Permutations	Algorithm	$b = 2$	$b = 4$	$b = 6$	$b = 8$
$k = 100$	BBMH SVM	94.37 (451.77)	97.96 (59.01)	98.99 (52.34)	99.25 (55.63)
	CBMH SVM	95.47 (256.04	98.56 (17.18)	99.13 (23.71)	99.29 (46.00)
$k = 300$	BBMH SVM	97.71 (542.13)	99.11 (95.28)	99.45 (92.92)	99.56 (129.30)
	CBMH SVM	99.08 (88.57)	99.26 (51.00)	99.51 (64.44)	99.56 (90.20)
$k = 600$	BBMH SVM	98.64 (258.42)	99.43 (212.53)	99.61 (193.58)	99.66 (236.89)
	CBMH SVM	98.59 (102.78)	99.47 (108.04)	99.61 (134.82)	99.64 (192.78)
$k = 900$	BBMH SVM	99.31 (300.10)	99.66 (312.97)	99.71 (368.44)	99.73 (335.10)
	CBMH SVM	99.10 (152.30)	99.56 (172.14)	99.66 (210.28)	99.67 (315.21)

3.4 Experimental Comparisons of the Integrations with Minwise Hashing Algorithms

In this subsection, we compare the performance of the integrations of linear SVM with b-bit minwise hashing, connected bit minwise hashing, f-fractional bit minwise hashing and one permutation hashing experimentally on large-scale data set *webspam* with 350,000 samples, each of which has 16,609,143 features. We split the set to 80 % for training and the remaining 20 % for testing and repeat every experiment 50 times. LIBLINEAR is chosen as the workhorse and all the experiments are conducted on the workstation with Intel(R) Xeon(R) CPU (E5-2630@2.30GHz) and 32GB RAM, under Centos 7 System. A series of penalty parameter C are set ranging from 0.01 to 10 and each sample is normalized to an unite vector. Besides, b-bit minwise hashing SVM (BBMH SVM for short) sets $k \in \{100, 300, 600, 900\}$ and $b \in \{2, 4, 6, 8\}$; connected bit minwise hashing SVM (CBMH SVM for short) sets $k \in \{100, 300, 600, 900\}$, $b \in \{2, 4, 6, 8\}$ and $n \in \{1, 2\}$; f-fractional bit minwise hashing SVM (FBMH SVM for short) sets $k \in \{100, 200, 300, 400, 500\}$ and f varying from 1 to 8 with the interval of 0.25 and one permutation hashing SVM (One-perm SVM for short) sets $r \in \{128, 256, 512, 1024\}$ and $b \in \{2, 4, 6, 8\}$ (r is the bins' number and b is the bits' number). The best experimental results from the perspective of accuracy (%) and time (second) on current experimental setup are highlighted in bold letters.

The experimental results in Table. 1 illustrate that the CBMH SVM algorithm reduces the processing time with a minor loss of accuracy compared with the BBMH SVM algorithm. It means that both the CBMH SVM and BBMH SVM algorithms have their own advantages. The experimental results in Table. 2 show that the FBMH SVM algorithm with the optimal combination of fractional bit yields better performance with less time-consumption. From the results in Table 3, the data compression by one permutation hashing scheme results in the significant reduction of the storage space and the data loading time in large-scale classification. Due to the better preservation of the original data structure, the One-perm SVM algorithm can be applied to the machines with low configuration keeping the similar experimental accuracy to that of the original data set.

Table 2 Experimental results *accuracy* (*time*) of the FBMH SVM algorithm on *webspam*

f-fractional	Combinations	$k = 100$	$k = 200$	$k = 300$	$k = 400$	$k = 500$
$f = 3.5$	$50\ \% \times \mathbf{3} + 50\ \% \times \mathbf{4}$	95.96 (60.04)	97.14 (95.72)	98.64 (135.24)	98.84 (163.27)	99.02 (203.78)
	$50\% \times \mathbf{2} + 50\ \% \times \mathbf{5}$	95.50 (60.49)	96.76 (96.04)	98.59 (135.62)	98.61 (163.28)	98.81 (204.55)
$f = 4.5$	$50\ \% \times \mathbf{4} + 50\ \% \times \mathbf{5}$	98.05 (61.09)	98.66 (96.82)	99.12 (136.78)	99.27 (165.13)	99.37 (205.03)
	$50\ \% \times \mathbf{3} + 50\ \% \times \mathbf{6}$	97.96 (61.22)	98.53 (97.45)	99.11 (137.80)	99.23 (165.26)	99.37 (206.13)
$f = 5.5$	$50\% \times \mathbf{5} + 50\ \% \times \mathbf{6}$	98.58 (61.72)	98.85 (97.78)	99.26 (138.34)	99.42 (166.12)	99.53 (209.49)
	$50\ \% \times \mathbf{4} + 50\ \% \times \mathbf{7}$	98.51 (61.79)	98.73 (97.82)	99.25 (138.41)	99.41 (166.52)	99.52 (210.64)
$f = 6.5$	$50\ \% \times \mathbf{6} + 50\ \% \times \mathbf{7}$	99.00 (62.22)	99.26 (99.36)	99.45 (138.75)	99.53 (167.44)	99.60 (211.39)
	$50\ \% \times \mathbf{5} + 50\ \% \times \mathbf{8}$	98.99 (62.39)	99.25 (99.70)	99.44 (138.76)	99.51 (167.80)	99.59 (212.61)

Table 3 Experimental results $accuracy$ $(time)$ of the One-perm SVM algorithm on *webspam*

Bins	$b = 2$	$b = 4$	$b = 6$	$b = 8$
$r = 128$	95.11 (47.63)	98.16 (34.43)	99.01 (36.02)	99.26 (30.24)
$r = 256$	97.25 (54.66)	98.95 (46.53)	99.38 (49.38)	99.49 (52.16)
$r = 512$	98.57 (96.63)	99.36 (83.77)	99.59 (85.04)	99.65 (89.69)
$r = 1024$	99.26 (134.39)	99.60 (137.18)	99.71 (159.5)	99.73 (162.05)
$r = \infty$	99.53 (2067)	99.53 (2068)	99.53 (2069)	99.53 (2070)

In summary, one can select appropriate processing algorithms on account of the characteristics of the data set and the system requirements.

4 Extensions and Variants of Minwise Hashing Algorithms

For the sake of similarity retrieval [1], the computation of similarity between sets is a core mission. Minwise hashing algorithm [2,3,60] stores the data set compactly and computes the distance to characterize similarity between data representations efficiently. B-bit minwise hashing algorithm [33–36,61] reduces bits of traditional minwise hashing from 64-bit to b-bit and saves both storage space and computing time. Connected bit minwise hashing[38] and f-fractional bit minwise hashing[39] improve the previous b-bit minwise hashing on time-consumption and accuracy. However, these algorithms require applying k independent random permutations on the entire data set leading to expensive cost of time, storage space and energy-consumption for the computation of similarities. However, one permutation hashing [40] appears without notable replacement of samples and preserves the matrix sparsity. Recently, many extensions and variants of minwise hashing algorithms have been proposed [62].

Li [63] developed 0-bit consistent weighted sampling (CWS) for efficiently estimating min-max kernel, which is a generalization of the resemblance kernel originally designed for binary data. Because the estimator of 0-bit CWS constitutes a positive definite kernel, this method can be naturally applied to large-scale data mining problems. By feeding the sampled data from 0-bit CWS to a highly efficient linear classifier, a nonlinear classifier can be trained effectively and approximately based on the min-max kernel. The min-max kernel often provides an effective measure of similarity for nonnegative data through an extensive classification study using kernel machines. Although the min-max kernel is nonlinear and might be difficult to be used for large-scale industrial applications, 0-bit CWS is a simplification of the original CWS method to build linear classifiers to approximate min-max kernel classifiers.

[64] focused on a simple 2-bit coding scheme and develop accurate nonlinear estimators of data similarity based on the 2-bit strategy. In the task of near neighbor search, a crucial step is to compute or estimate data similarities once a set of candidate data points have been identified by hash table techniques. The 2-bit coding scheme appears to be overall a good choice for building hash tables in near neighbor search and developing accurate nonlinear estimators.

Shrivastava and Li [65–67] proposed asymmetric minwise hashing (MH-ALSH) to provide a solution for estimating set resemblance. The new scheme utilizes asymmetric

transformations to cancel the bias of traditional minwise hashing towards smaller sets, making the final "collision probability" monotonic in the inner product. The theoretical comparisons show that MH-ALSH is provably better than traditional minwise hashing and other recently proposed hashing algorithms for the task of retrieving with binary inner products.

Weighted minwise hashing (WMH) [68] is one of the fundamental subroutine, required by many celebrated approximation algorithms, commonly adopted in industrial practice for large-scale search and learning. The resource bottleneck of the algorithms is the computation of multiple (typically a few hundreds to thousands) independent hashes of the data. Exact weighted minwise hashing broke the expensive barrier and showed an expected constant amortized time algorithm with only a few bits of storage per hash value.

The query complexity of locality sensitive hashing (LSH) is dominated by the number of hash evaluations, and this number grows with the data size. [69] presented a hashing technique to generate all the necessary hash evaluations for similarity search using one single permutation. The key of the proposed hash function is a "rotation" scheme which is the sparse sketches of one permutation hashing in an unbiased fashion thereby maintaining the LSH property.

[70] studied large-scale regression analysis where both the number of variables and observations may be large and in the order of millions or more. In order to make progress, one must seek a compromise between statistical and computational efficiency. For dealing with this large-scale problem, the proposed min-wise hash random-sign mapping (MRS mapping) is a dimensionality reduction technique for a sparse binary design matrix. It allows for the construction of variable importance measures, and is more amenable to statistical analysis. For both linear and logistic models, finite-sample bounds were given on the prediction error of procedures which perform regression in the new lower-dimensional space after applying MRS mapping.

In [71], Li et al. developed a parallelization scheme using GPUs, which reduced the processing time. Reducing the preprocessing time is highly beneficial in practice, for example, for duplicate web page detection (where minwise hashing is a major step in the crawling pipeline) or for increasing the testing speed of online classifiers (when the test data are not preprocessed). The identification of compound-protein interactions plays key roles in the drug development toward discovery of new drug leads and new therapeutic protein targets. The paper [72] developed a novel chemogenomic method to make a scalable prediction of compound-protein interactions from heterogeneous biological data using minwise hashing. In [73], the twisted tabulation was invented to yield Chernoff-style concentration bounds and high probability amortized performance bounds for linear probing when using minwise for similarity estimation to reduce variance.

5 Remarks and Future Directions

Minwise hashing schemes can improve the computation efficiency and save the storage space without notable loss of accuracy. This paper has offered a systematic review of minwise hashing algorithms and the variants mainly including five basic algorithms:

minwise hashing, b-bit minwise hashing, connected bit minwise hashing, f-fractional bit minwise hashing and one permutation hashing. Based on the five algorithms, the extensions and variants of minwise hashing algorithms are presented. Some of these algorithms have already been used in the real-life applications, such as large-scale regression and classification and scalable prediction of compound-protein interactions etc. Researchers in data mining, especially in SVMs can benefit from this survey in better understanding the essence of the minwise hashing algorithms. Furthermore, their limitations, major opportunities and challenges, as well as potential important research directions have been pointed out.

As we can see, the extensions or variants of minwise hashing algorithms were mostly based on the data-independent hashing functions, so there is a great space for developing methods based on the data-dependent hashing functions. These functions should also have the same desirable properties such as good generalization, scalability, simple and easy implementation of algorithm robustness, as well as local sensitivity. From this respect, we believe that minwise hashing algorithms can produce better results and worth to be considered.

Acknowledgments This work has been partially supported by Grants from National Natural Science Foundation of China (No. 11271361, No.71331005), Major International (Ragional) Joint Research Project (No. 71110107026).

References

1. Andoni A, Indyk P (2006) Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In Foundations of Computer Science, pp 459–468
2. Broder AZ (1997) On the resemblance and containment of documents. Compression Complex Seq 1997:21–29
3. Broder AZ, Glassman SC, Manasse MS, Zweig G (Sep 1997) Syntactic clustering of the web. In: Proceedings of the 6th international conference on World Wide Web, Vol 29, p 1157–1166
4. Gollapudi S, Sharma A (2009) An axiomatic approach for result diversification. In: Proceedings of the 18th international conference on World Wide Web, p 381–390
5. Kalpakis K, Tang S (2008) Collaborative data gathering in wireless sensor networks using measurement co-occurrence. Comput Commun 31(10):19791992
6. Najork M, Gollapudi S, Panigrahy R (2009) Less is more: sampling the neighborhood graph makes salsa better and faster. Web Search and Data Mining, p 242–251
7. Urvoy T, Chauveau E, Filoche P, Lavergne T (2008) Tracking web spam with html style similarities. ACM Trans Web 2(1):1–28
8. Gong Y, Kumar S, Verma V, Lazebnik S (2012) Angular quantization-based binary codes for fast similarity search. In: Advances in neural information processing systems, p 1196–1204
9. He J, Chang S, Radhakrishnan R. Bauer C (2011) Compact hashing with joint optimization of search accuracy and time. In IEEE Conference on Computer Vision and Pattern Recognition, p 753–760, IEEE
10. Gionis A, Indyk P, Motwani R (1999) Similarity search in high dimensions via hashing. VLDB 99:518–529
11. Andoni A, Indyk P (2006) Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In 2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), p 459–468,IEEE
12. Torralba A, Fergus R, Weiss Y (2008) Small codes and large image databases for recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, p 1–8. IEEE
13. Liu W, Wang J, Kumar S, Chang S (2011) Hashing with graphs. In: Proceedings of the 28th international conference on machine learning, p 1–8

14. Shi QF, Petterson J, Dror G, Langford J, Smola A, Vishwanathan SVN (2009) Hash kernels for structured data. J Mach Learn Res 10:2615–2637
15. Mu YD, Hua G, Fan W, Chang SF (2014) Hash-SVM: scalable kernel machines for large-scale visual classification. In: IEEE Conference on Computer Vision and Pattern Recognition, p 979–986
16. Litayem S, Joly A, Boujemaa N (2012) Hash-based support vector machines approximation for large scale prediction. Br Mach Vis Conf 34(6):1092–1104
17. Gong Y, Lazebnik S (2011) Iterative quantization: a procrustean approach to learning binary codes. In: IEEE Conference on Computer Vision and Pattern Recognition, p 817–824. IEEE
18. Liu W, Wang J, Ji R, Jiang Y, Chang (2012) Supervised hashing with kernels. In: IEEE Conference on Computer Vision and Pattern Recognition, p 2074–2081. IEEE
19. Joly A, Buisson O (2008) A posteriori multi-probe locality sensitive hashing. In: Proceedings of the 16th ACM international conference on Multimedia, p 209–218. ACM
20. Datar M, Immorlica N, Indyk P, Mirrokni VS (2004) Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th annual symposium on Computational geometry*, p 253–262. ACM, 2004
21. Kulis B, Darrell T (2009) Learning to hash with binary reconstructive embeddings. In: Advances in neural information processing systems, p 1042–1050
22. Kulis B, Grauman K (2012) Kernelized locality-sensitive hashing. IEEE Trans Pattern Anal Mach Intell 34(6):1092–1104
23. Raginsky M, Lazebnik S (2009) Locality-sensitive binary codes from shift-invariant kernels. In: Advances in neural information processing systems, p 1509–1517
24. Salakhutdinov R, Hinton G (2009) Semantic hashing. Int J Approx Reason 50(7):969–978
25. Weiss Y, Torralba A, Fergus R (2009) Spectral hashing. In: Advances in neural information processing systems, p 1753–1760
26. Wang J, Kumar S, Chang S (2010) Sequential projection learning for hashing with compact codes. In: Proceedings of the 27th international conference on machine learning, p 1127–1134
27. Zhang D, Wang J, Cai D, Lu J (2010) Self-taught hashing for fast similarity search. In: Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, p 18–25. ACM
28. Zhang D, Wang F, Si L (2011) Composite hashing with multiple information sources. In: Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, p 225–234. ACM
29. Norouzi M, Blei DM (2011) Minimal loss hashing for compact binary codes. In: Proceedings of the 28th international conference on machine learning, p 353–360
30. Pandey S, Broder A, Chierichetti F, Josifovski V, Kumar R, Vassilvitskii S (2009) Nearest-neighbor caching for content-match applications. In: Proceedings of the 18th international conference on World Wide Web, p 441–450. ACM
31. Shrivastava A, Li P (2012) Fast near neighbor search in high-dimensional binary data. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, p 474–489. Springer
32. Li P, Knig AC, Gui WH (2010) b-bit minwise hashing for estimating three-way similarities. In: Advances in Neural Information Processing Systems
33. Li P, Knig AC (2010) b-bit minwise hashing. In: Proceedings of the 19th international conference on World Wide Web, p 671–680. ACM
34. Li P, Knig AC (2011) Theory and applications of b-bit minwise hashing. Commun ACM 54(8):101–109
35. Li P, Shrivastava A, Moore J, Knig AC (2011) b-bit minwise hashing for large-scale learning. In: Advances in Neural Information Processing Systems. Neural Information Processing Foundation
36. Li P, Shrivastava A, Moore J, Knig AC (2011) Hashing algorithms for large-scale learning. In: Advances in neural information processing systems, p 2672–2680
37. Li P, Moore JL, Knig AC (2011) b-bit minwise hashing for large-scale linear svm. Technical report
38. Yuan XP, Long J, Zhang ZP, Luo YY, Zhang H, Gui WH (2013) Connected bit minwise hashing. J Comput Res Dev 50(4):883–890
39. Yuan XP, Long J, Zhang ZP, Luo YY, Zhang H, Gui WH (2012) f-fractional bit minwise hashing. JSoftw 7(1):228–236
40. Li P, Owen A, Zhang CH (2012) One permutation hashing. In: Advances in Neural Information Processing Systems, p 3113–3121

41. Fetterly D, Manasse M, Najork M, Wiener J (2003) A large-scale study of the evolution of web pages. In: Proceedings of the 12th international conference on World Wide Web, p 669678, New York, NY, USA. Proceedings of the 12th International World Wide Web Conference, ACM
42. Tang JJ, Tian YJ (2015) f-fractional bit minwise hashing for large-scale learning. In: 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), Vol 3, p 60–63. IEEE
43. Tang JJ, Tian YJ, Liu DL (2015) Connected bit minwise hashing for large-scale linear svm. In: Fuzzy Systems and Knowledge Discovery (FSKD), 2015 12th International Conference on, p 995–1002. IEEE
44. Tian YJ, Ping Y (2014) Large-scale linear nonparallel support vector machine solver. Neural Netw 50:166–174
45. Manku GS, Jain A, Sarma AD (2007) Detecting near-duplicates for web crawlings. Proceedings of the 16th international conference on World Wide Web, (10):141–150
46. Tashev I, Acero A (2010) Statistical modeling of the speech signal. In: International Workshop on Acoustic, Echo, and Noise Control (IWAENC)
47. Yuan XP, Long J, Zhang ZP, Luo YY, Zhang H, Gui WH (August 2012) Research on optimal fractional bit minwise hashing. Computer Science, 39(8)
48. Hsieh CJ, Chang KW, Lin CJ, Keerthi SS, Sundararajan S (2008) A dual coordinate descent method for large-scale linear svm. In: International Conference on Machine Learning, p 408–415
49. Cherkasova L, Eshghi K, Morrey CB, Tucek J, Veitch A (2009) Applying syntactic similarity algorithms for enterprise information management. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, p 1087–1096. ACM
50. Forman G, Eshghi K, Suermondt J (2009) Efficient detection of large-scale redundancy in enterprise file systems. ACM SIGOPS Op Sys Rev 43(1):84–91
51. Bendersky M, Croft WB (2009) Finding text reuse on the web. In: Proceedings of the 2nd ACM International Conference on Web Search and Data Mining, p 262–271. ACM
52. Biggio B, Fumera G, Roli F (2014) Security evaluation of pattern classifiers under attack. IEEE Trans Knowl Data Eng 26(4):984–996
53. Cox RV, Kamm CA, Rabiner L, Schroeter J, Wilpon JG (2000) Speech and language processing for next-millennium communications services. Proc IEEE 88(8):1314–1337
54. Camastra F, Verri A (2005) A novel kernel method for clustering. IEEE Trans Pattern Anal Mach Intell 27(5):801–805
55. Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ (2008) Liblinear: a library for large linear classification. J Mach Learn Res 9(4):1871–1874
56. Joachims T (2006) Training linear svms in linear time. Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (10):217–226. KDD '06
57. Border AZ, Bottou L, Gallinari P (2009) Sgd-qn: careful quasi-newton stochastic gradient descent. J Mach Learn Res 10:1737–1754
58. Zhang T (July 4-8 2004) Solving large scale linear prediction problems using stochastic gradient descent algorithms. International Conference on Machine Learning
59. Singer Y, Srebro N (2007) Pegasos: Primal estimated sub-gradient solver for svm. International Conference on Machine Learning, pages 807–814
60. Indyk P (2001) A small approximately min-wise independent family of hash functions. J Algorithms 38(1):84–90
61. Li P, Konig C (2011). Accurate estimators for improving minwise hashing and b-bit minwise hashing. arXiv preprint arXiv:1108.0895
62. Li P, Shrivastava A, Konig C (2011) Training logistic regression and svm on 200gb data using b-bit minwise hashing and comparisons with vowpal wabbit (vw). arXiv preprint arXiv:1108.3072
63. Li P (2015) 0-bit consistent weighted sampling. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p 665–674. ACM
64. Li P, Mitzenmacher M, Shrivastava A (2016) 2-bit random projections, nonlinear estimators, and approximate near neighbor search. arXiv preprint arXiv:1602.06577
65. Shrivastava A, Li P (2014) Asymmetric minwise hashing. arXiv preprint arXiv:1411.3787
66. Shrivastava A, Li P (2015) Asymmetric minwise hashing for indexing binary inner products and set containment. In: Proceedings of the 24th International Conference on World Wide Web, p 981–991. ACM

67. Shrivastava A, Li P (2014) Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In: Advances in Neural Information Processing Systems, p 2321–2329
68. Shrivastava A (2016) Exact weighted minwise hashing in constant time. arXiv preprint arXiv:1602.08393
69. Shrivastava A, Li P (2014) Densifying one permutation hashing via rotation for fast near neighbor search. In: International Conference on Machine Learning, p 557–565
70. Shah RD, Meinshausen N (2013) Min-wise hashing for large-scale regression and classification with sparse data. arXiv preprint arXiv:1308.1269
71. Li P, Shrivastava A, Konig CA (2012) Gpu-based minwise hashing: Gpu-based minwise hashing. In: Proceedings of the 21st International Conference on World Wide Web, p 565–566. ACM
72. Tabei Y, Yamanishi Y (2013) Scalable prediction of compound-protein interactions using minwise hashing. BMC Syst Biol 7(6):1
73. Dahlgaard S, Thorup M (2014) Approximately minwise independence with twisted tabulation. In: Scandinavian Workshop on Algorithm Theory, p 134–145. Springer