Improving Hashing Algorithms for Similarity Search *via* MLE and the Control Variates Trick

Keegan Kang KEEGAN_KANG@SUTD.EDU.SG
Engineering Systems and Design Pillar, Singapore University of Technology and Design

Sergey Kushnarev SKUSHNA1@JHU.EDU
Department of Applied Mathematics and Statistics, Johns Hopkins University

Wei Pin Wong WEIPIN_WONG@SUTD.EDU.SG
Science, Mathematics, and Technology Cluster, Singapore University of Technology and Design

Rameshwar Pratap RAMESHWAR@IITMANDI.AC.IN
School of Computing and Electrical Engineering, IIT Mandi

Haikal Yeo YEOHAIKAL@GMAIL.COM
Independent

Yijia Chen YIJIA_CHEN@MYMAIL.SUTD.EDU.SG
Singapore University of Technology and Design

Editors: Vineeth N Balasubramanian and Ivor Tsang

Abstract

Hashing algorithms are continually used for large-scale learning and similarity search, with computationally cheap and better algorithms being proposed every year. In this paper we focus on hashing algorithms which involve estimating a distance measure $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ between two vectors $\boldsymbol{x}_i, \boldsymbol{x}_j$. Such hashing algorithms require generation of random variables, and we propose two approaches to reduce the variance of our hashed estimates: control variates and maximum likelihood estimates. We explain how these approaches can be immediately applied to a wide subset of hashing algorithms. Further, we evaluate the impact of these methods on various datasets. We finally run empirical simulations to verify our results.

Keywords: control variates, hashing algorithms, maximum likelihood estimation, statistical techniques

1. Introduction

The first ever locality sensitive hashing algorithm established was *minhash* (Broder, 1997), which looked at estimating the resemblance between any two documents. Since then, a wide variety of hashing algorithms has been proposed in the literature (see Wang et al. (2014); Chi and Zhu (2017) for most of the common hashing algorithms since 2017), including Shrivastava (2016); Li (2019); Pratap et al. (2019) in recent times. We focus on two broad classes of hashing algorithms which estimate some distance measure $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ between two vectors $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^p$. For ease of notation, we define a generic hash function h on a set of objects $X \subset \mathbb{R}^p$ where for all $\boldsymbol{x}_i \in X$, we have $h : \boldsymbol{x}_i \in \mathbb{R}^p \mapsto v_i \in \mathbb{R}, 1 \leq i \leq n$.

Name	**Distance**	$f^{(1)}(\rho_h(\boldsymbol{x}_i, \boldsymbol{x}_j))$	$f^{(2)}(v_i, v_j)$
Minwise hashing	Resemblance	$\rho_h(\boldsymbol{x}_i, \boldsymbol{x}_j)$	NA
Sign random projections	Angular distance	$1 - \pi\rho_h(\boldsymbol{x}_i, \boldsymbol{x}_j)/k$	NA
Random projections	Inner product	NA	$v_i v_j$

Table 1: Examples of $f^{(1)}(\cdot)$ and $f^{(2)}(\cdot, \cdot)$ in these two classes of hashing algorithms.

Our first class of algorithms has the property that for all $\boldsymbol{x}_i \in X$, we have

$$\mathbb{P}[h(\boldsymbol{x}_i) = h(\boldsymbol{x}_j)] = \mathbb{P}[v_i = v_j] = \rho_h(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{1}$$

where $\rho_h(\boldsymbol{x}_i, \boldsymbol{x}_j) \in [0, 1]$ is some similarity measure defined on X, and we can approximate $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ from some $f^{(1)}(\rho_h(\boldsymbol{x}_i, \boldsymbol{x}_j))$ where $f^{(1)}$ is linear. Some examples include *minwise hashing* (Broder, 1997; Ioffe, 2010; Li and König, 2010; Ji et al., 2013; Shrivastava, 2016) and *sign random projections* (Charikar, 2002; Ji et al., 2012; Kang and Wong, 2018).

Our second class of algorithms has the property that for all $\boldsymbol{x}_i \in X$, we have

$$\mathbb{E}[f^{(2)}(h(\boldsymbol{x}_i), h(\boldsymbol{x}_j))] = \mathbb{E}[f^{(2)}(v_i, v_j)] = d(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{2}$$

Examples of such hash functions include *random projections* (Indyk and Motwani, 1998; Li et al., 2006b, 2010; Li, 2019). We give examples of the two classes in Table 1.

In both classes, computing the actual distance between these vectors takes $O(p)$ time. Given N pairs of vectors, the total time taken is $O(Np)$. If we hash these vectors $k \ll p$ times, then by the Law of Large Numbers, we can get an estimate of $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ by computing either the estimator $f^{(1)}(\sum_{t=1}^{k} h_t(\boldsymbol{x}_i)h_t(\boldsymbol{x}_j)/k)$ or $\left(\sum_{t=1}^{k} f^{(2)}(h_t(\boldsymbol{x}_i), h_t(\boldsymbol{x}_j))\right)/k$. This reduces our computational cost down to $O(Nk)$.

If we think of $Y_t^{(1)} := h_t(\boldsymbol{x}_i)h_t(\boldsymbol{x}_j)$ in the first class, and $Y_t^{(2)} := f^{(2)}(h_t(\boldsymbol{x}_i), h_t(\boldsymbol{x}_j))$ in the second class as random variables, we immediately observe these three facts: First, $Y_t^{(1)}$ can be modelled as a Bernoulli random variable, regardless of the random process involved in $h(\boldsymbol{x})$. Second, the variance of these estimators are dependent on number of hashes, k, rather than on initial dimension p. Third, techniques such as maximum likelihood estimators (MLEs) (Casella and Berger, 2001) can be applied to $Y_t^{(1)}$, and control variates (Lavenberg and Welch, 1981) applied to $Y_t^{(2)}$ to further reduce the variance of our estimates.

Our goal is to achieve significant variance reduction in estimating the pairwise similarities without generating more i.i.d. copies of these random variables.

2. Related work

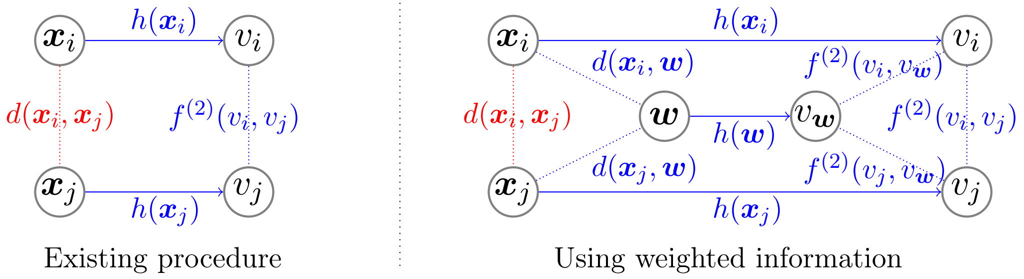
Both MLEs and control variates have been studied for variance reduction in similarity estimation algorithms using sketching techniques. We mention a few notable works as follows and contrast them with our contributions in this paper. Kang and Wong (2018) suggests variance reduction using the MLE method in similarity estimations via `SRP` (Charikar, 2002), as a contrast to antithetic sampling in Ji et al. (2013).

In this paper, we generalise the MLE technique for a wide class of discrete hashing algorithm [1] such as MinHash (Broder et al., 1998), Winner-takes-all (`WTA`) (Yagnik et al., 2011). The result of Kang and Wong (2018) can be thought of as a special case of this work.

For the control variate technique, we extend upon the special case of random projections in Kang (2021) which can be applied to a wide class of random projection algorithms such as signed full random projection (Li, 2019), sparse and very sparse random projection (Li et al., 2006b; Kane and Nelson, 2014; Cohen et al., 2018), stable random projection (Li and Hastie, 2008), as well as frequency estimators (Pratap and Kulkarni, 2021). While the control variate technique can be seen as an analogue to the MLE technique for random projections in Li et al. (2020), the control variate technique is much cleaner.

3. Our Contributions

Our work goes beyond literature that looks at statistical properties of hashing algorithms (Rusu and Dobra, 2007; Ahfock et al., 2017) by also incorporating structural information of the dataset $\{\boldsymbol{x}_i\}$. We propose creating weighted vectors $\boldsymbol{w}_s$ which point in the direction of groups or clusters of vectors in the dataset, and hash them to get $v_{\boldsymbol{w}_s}$. More explicitly, these weighted vectors $\boldsymbol{w}_s$ should have small distances $d(\boldsymbol{w}_s, \boldsymbol{x}_i)$, between vectors $\boldsymbol{x}_i$ in the dataset. We defer the discussion on how to create these vectors to Section 6. We propose conditioning upon these weighted vectors to improve our estimates, by making use of the true values $d(\boldsymbol{w}_s, \boldsymbol{x}_i), d(\boldsymbol{w}_s, \boldsymbol{x}_j)$ which we pre-compute in advance, and the estimated values $\sum_{t=1}^k 1_{\{h_t(\boldsymbol{w}_s)=h_t(\boldsymbol{x}_i)\}}/k, \sum_{t=1}^k 1_{\{h_t(\boldsymbol{w}_s)=h_t(\boldsymbol{x}_j)\}}/k$ (or similarly $f^{(2)}(v_{\boldsymbol{w}_s}, v_i), f^{(2)}(v_{\boldsymbol{w}_s}, v_j)$). Figure 1 shows the process pictorially for the second case.



$h(\boldsymbol{x}_i)$
$d(\boldsymbol{x}_i, \boldsymbol{x}_j)$
$f^{(2)}(v_i, v_j)$
$h(\boldsymbol{x}_j)$
Existing procedure
$h(\boldsymbol{x}_i)$
$d(\boldsymbol{x}_i, \boldsymbol{w})$
$f^{(2)}(v_i, v_{\boldsymbol{w}})$
$d(\boldsymbol{x}_i, \boldsymbol{x}_j)$
$h(\boldsymbol{w})$
$f^{(2)}(v_i, v_j)$
$d(\boldsymbol{x}_j, \boldsymbol{w})$
$f^{(2)}(v_j, v_{\boldsymbol{w}})$
$h(\boldsymbol{x}_j)$
Using weighted information

Figure 1: Using one weighted vector $\boldsymbol{w}$ for which we know relevant information (shown in blue) to estimate unknown $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ (shown in red).

Our contributions are the following: 1) We outline the MLE and control variates approaches which lead to improved variance estimates of $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$. 2) We discuss the cases where these two methods improve the variance reduction substantially, as well as potential tradeoffs and limitations. 3) We validate our approach via empirical simulations.

1. We refer a hashing algorithm as discrete if it outputs integer valued hash values.

	A	B	C	D	E
v_{is}	equal	different	equal	equal	distinct
v_{js}	equal	equal	different	equal	distinct
$v_{\boldsymbol{w}s}$	equal	equal	equal	different	distinct

Table 2: Table of possible clashes between the hashed values v_{is}, v_{js}, $v_{\boldsymbol{w}s}$.

In the next two sections, we assume that values $d(\boldsymbol{w}_s, \boldsymbol{x}_i)$ (for $s = 1, \ldots, S$) have been computed and stored for all $\boldsymbol{x}_i \in X$, which is a once-off computational cost of $O(npS)$ and a storage cost of $O(nS)$.

4. Improving The First Class Of Hashing Algorithms: Maximum Likelihood Estimators (MLE)

The MLE trick has been known since the 1940s (Deming and Stephan, 1940), and has been used in Church et al. (2006); Li and Church (2007); Kang and Wong (2018) for specific hashing algorithms. We extend this to classes of hashing algorithms where $\mathbb{P}[v_i = v_j] = \rho_h(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

There are two subcases. The first is where the hashed values are binary (e.g. *sign random projections* (Charikar, 2002), *min-max hash* (Ji et al., 2013), *super-bit locality sensitive hashing* (Ji et al., 2012)), and the other where the hashed values are discrete (e.g. *minhash* (Broder, 1997), *weighted minhash* (Shrivastava, 2016)). We outline the discrete case here, which extends the work in Kang and Wong (2018) to the general discrete case.

Suppose we have k hashes, and define $v_{ik}, v_{jk}, v_{\boldsymbol{w}k}$ to be the respective values of $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{w}$ under the k^{th} hash. Consider the k^{th} triple given by $(v_{ik}, v_{jk}, v_{\boldsymbol{w}k})$. There are only five possible sets of triples: a) all elements are equal, b) two elements are equal and the third is different, and c) all elements are distinct. Table 2 shows the different types of triples.

Suppose we count the triples in each set and denote this as n_l, where $l \in \{A, B, C, D, E\}$, and p_l the probability of observing a triple falling in the set l. We note that $\sum_l n_l = k$.

Since we have pre-computed $d(\boldsymbol{x}_i, \boldsymbol{w})$ and $d(\boldsymbol{x}_j, \boldsymbol{w})$, and $f^{(1)}$ is linear, we can invert $f^{(1)}$ to find $\rho_h(\boldsymbol{x}_i, \boldsymbol{w}), \rho_h(\boldsymbol{x}_j, \boldsymbol{w})$. Moreover, we can write $p_A + p_C = \rho_h(\boldsymbol{x}_i, \boldsymbol{w})$, $p_A + p_B = \rho_h(\boldsymbol{x}_j, \boldsymbol{w})$, $p_A + p_D = \rho_h(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and $\sum_l p_l = 1$. Finally, since $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is given by $p_A + p_D$, we write the log likelihood function $\ell(p_A, p_B, p_C, p_D, p_E)$ in terms of p_A and p_D, and get

$$\begin{aligned}\ell(p_A, p_D) = n_A \log(p_A) + n_B \log(\rho_h(\boldsymbol{x}_j, \boldsymbol{w}) - p_A) + n_C \log(\rho_h(\boldsymbol{x}_i, \boldsymbol{w}) - p_A) + n_D \log(p_D) \\ + n_E \log(1 - \rho_h(\boldsymbol{x}_i, \boldsymbol{w}) - \rho_h(\boldsymbol{x}_j, \boldsymbol{w}) + p_A - p_D). \qquad (3)\end{aligned}$$

as we want to compute the MLE of $\hat{p}_A + \hat{p}_D$ to give an estimate of $\rho_h(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

The log-likelihood function can be reduced to a cubic in $\hat{p}_A$, and can be solved to give us a value of $\hat{p}_D$. The following theorem tells us that the obtained MLE of $\rho_h(\boldsymbol{x}_i, \boldsymbol{x}_j)$ will be more accurate than the estimate obtained without the MLE. Thus we can get a more precise estimate of $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as well.

Theorem 1 *Suppose we have a hashing algorithm where the estimate of interest is given by* $\rho_h(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathbb{E}[f(Y^{(1)})]$, *where* $Y^{(1)}$ *is a Bernoulli random variable,* f *is a linear function, and the output of the hashing algorithm takes discrete values. Suppose we add a weighted* *vector $\boldsymbol{w}$, and compute the maximum likelihood estimate via (3). Then: a) this estimator is unbiased, and b) the asymptotic variance of this estimator $\text{Var}[\rho_h(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{MLE}$ is always lower than or equal to the variance of the estimator $\text{Var}[\rho_h(\boldsymbol{x}_i, \boldsymbol{x}_j)]$ without the MLE.*

We prove this theorem in the supplementary material.

We give two remarks on our method. For our method to work well, we need to choose a weighted vector that is appropriate for the hashing algorithm we are using. In order to do so, we need to examine our proof of part b) more carefully. The proof involves showing that $\text{Var}[\rho_h(\boldsymbol{x}_i, \boldsymbol{x}_j)] - \text{Var}[\rho_h(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{\text{MLE}} \geq 0$, and we have shown that this is equivalent to

$$\frac{p_A^2(p_B - p_C)^2((p_A p_B + p_A p_C + p_B p_C)(p_D + p_E) + p_A p_B p_C)}{(p_A + p_B)(p_A + p_C)(p_B + p_C)} \geq 0 \tag{4}$$

with equality when $p_B = p_C$ and $p_D = p_A(1 - p_A - p_B)/(p_A + p_B)$. Equation (4) allows us to “reverse engineer" and pick a choice of weighted vector to get the most variance reduction based on the hashing algorithm we use.

To give an example, we can see that in Table 1, we estimate our angular distance between $\boldsymbol{x}_i, \boldsymbol{x}_j$ with sign random projections using the estimate $1 - \pi(p_A + p_D)/k$ using our notation. Since we want to avoid the case where $p_B = p_C$ (with no variance reduction), this implies our weighted vector must have high angular similarity with both $\boldsymbol{x}_i, \boldsymbol{x}_j$, to avoid the case of $p_B \approx p_C$. In general, choosing the best weighted vector for each hashing algorithm is not easy, but can be done via some heuristics.

Our second remark is that we necessarily need to run a root finding process to get a better estimate, and implementing such a root finding process in high level programming languages such as Matlab can take up a considerable amount of time, unless the code is written in a low level programming language like C++.

5. Improving The Second Class Of Hashing Algorithms: Control Variates

We briefly review control variates (see Lavenberg and Welch (1981) for full treatment). Suppose we have a random number generator that generates a random variable Y, and we want to estimate $\mathbb{E}[Y]$. Suppose we use the same random numbers from the generator to generate a random variable Z, but we know the true mean $\mathbb{E}[Z] = \mu_Z$. For any c, we have that $(Y + c(Z - \mu_Z))$ is an unbiased estimator of Y. The variance is given by $\text{Var}[Y + c(Z - \mu_Z)] = \text{Var}[Y] + c^2 \text{Var}[Z] + 2c \,\text{Cov}(Y, Z)$.

We can find the value of $\hat{c} = -\frac{\text{Cov}(Y,Z)}{\text{Var}[Z]}$ which minimizes the variance by completing the square, and hence get the lowest variance $\text{Var}[Y + c(Z - \mu_Z)] = \text{Var}[Y] - \frac{\text{Cov}(Y,Z)^2}{\text{Var}[Z]}$.

In this case, Z is called a *control variate*, and $\hat{c}$ is a *control variate correction*. We observe that the theoretical variance of $(Y + c(Z - \mu_Z))$ is always lower than the theoretical variance of Y, with equality if there is no correlation between Y and Z.

For our second class of hashing algorithms, we thus set $Y = f^{(2)}(v_i, v_j)$ which is an approximation of $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that we want to find. We set Z to be a linear combination of estimates we already know and can compute, and choose the number of weighted vectors $\boldsymbol{w}_s$, of which the general case is given by

$$\begin{aligned} Z = \alpha_1 f^{(2)}(v_i, v_i) + \alpha_2 f^{(2)}(v_j, v_j) + \sum_s \alpha_{3,s} f^{(2)}(v_i, \boldsymbol{w}_s) + \sum_s \alpha_{4,s} f^{(2)}(\boldsymbol{w}_s, v_i) \\ + \sum_s \alpha_{5,s} f^{(2)}(v_j, \boldsymbol{w}_s) + \sum_s \alpha_{6,s} f^{(2)}(\boldsymbol{w}_s, v_j) + \sum_{s,t} \alpha_{7,s,t} f^{(2)}(\boldsymbol{w}_s, \boldsymbol{w}_t). \end{aligned} \tag{5}$$

If $f^{(2)}$ is symmetric, the expression for Z will have only five terms.

As control variates always guarantee a variance reduction, the goal is to find the values of α which maximizes this variance reduction. In practice, we do not consider all α terms, as we may omit the terms $f^{(2)}(\cdot, \cdot)$ which involve values of $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ which is being estimated.

To compute the control variate correction $\hat{c}$, we need to find an expression for an arbitrary $\mathbb{E}[f^{(2)}(v_i, v_j) f^{(2)}(v_k, v_l)]$, which will be used to compute $\text{Cov}(Y, Z)$ and $\text{Var}[Z]$. With these values we then can find the values of αs which maximize the variance reduction $\dfrac{\text{Cov}(Y,Z)^2}{\text{Var}[Z]}$.

In the next two subsections we give two examples of how to compute $\hat{c}$, both which are dependent on the form of $f^{(2)}(v_i, v_j)$: one where $(f^{(2)}(v_i, v_j), f^{(2)}(v_k, v_l))$ converges to a bivariate normal, and the case where $(f^{(2)}(v_i, v_j), f^{(2)}(v_k, v_l))$ do not.

5.1. Control variates where $(f^{(2)}(v_i, v_j), f^{(2)}(v_k, v_l))$ converge to a bivariate normal distribution

In this section, we look at both *random projection* (Indyk and Motwani, 1998; Li et al., 2006a) and *very sparse random projection* (Li et al., 2006b). For the *random projection case*, given two real-valued vectors $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^p$ normalized to length 1, we generate a random matrix $R_{p \times k}$ with entries in R being i.i.d $N(0,1)$. We then compute vectors $\boldsymbol{v}_i$, $\boldsymbol{v}_j$ given by $\boldsymbol{v}_i = \boldsymbol{x}_i^T R, \boldsymbol{v}_j = \boldsymbol{x}_j^T R$.

For the *very sparse random projection* case, the only modification is that the entries in the random matrix $R_{p \times k}$ are drawn independently from the scaled Sparse Bernoulli distribution,

$$r_{i,j} = \sqrt{s} \begin{cases} 1, & \text{with probability } \frac{1}{2s}, \\ 0, & \text{with probability } 1 - \frac{1}{s}, \\ -1, & \text{with probability } \frac{1}{2s}, \end{cases} \tag{6}$$

with $s \geq 1$ being a parameter of our choice.

We note that the control variate terms $v_{\boldsymbol{w}_s} v_{\boldsymbol{w}_t}$ always remain the same regardless of the pair we want to estimate, and only requires stored values of $\alpha_{s,t}$ for each pair.

Both *random projection* and *very sparse random projection* aim to estimate the inner product $d(\boldsymbol{x}_i, \boldsymbol{x}_j) := \boldsymbol{x}_i^T \boldsymbol{x}_j$ between any two vectors $\boldsymbol{x}_i, \boldsymbol{x}_j$. Denote $v_{i,t}$ and $v_{j,t}$ to be the t^{th}

element of $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ respectively. For both cases, for all t, it can be shown that

$$\mathbb{E}[v_{i,t}v_{j,t}] = \mathbb{E}\left[\sum_{k=1}^{p} x_{i,k}r_{k,t}\sum_{k'=1}^{p} x_{j,k'}r_{k',t}\right] \tag{7}$$

$$= \mathbb{E}\left[\sum_{k=1}^{p} x_{i,k}x_{j,k}(r_{k,t})^2 + \sum_{k\neq k'} x_{i,k}x_{j,k'}(r_{k,t}r_{k',t})\right]$$

$$= \sum_{k=1}^{p} x_{i,k}x_{j,k}\mathbb{E}[{r_{k,t}}^2] + \sum_{k\neq k'} x_{i,k}x_{j,k'}\mathbb{E}[r_{k,t}r_{k',t}] \tag{8}$$

$$= \sum_{k=1}^{p} x_{i,k}x_{j,k} = d(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{9}$$

where $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the dot product between $\boldsymbol{x}_i, \boldsymbol{x}_j$. The above equality holds as $\mathbb{E}[r_{k,t}^2] = 1$, and $\mathbb{E}[r_{k,t}r_{k',t}] = 0$ for $k \neq k'$. Similarly, $\mathbb{E}[v_{i,t}^2] = \sum_{k=1}^{p} x_{i,k}^2 = \|\boldsymbol{x}_i\|$.

For *ordinary random projections*, we have that for $1 \leq t \leq k$ the tuple is bivariate normal:

$$\begin{pmatrix} v_{i,t} \\ v_{j,t} \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & d(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ d(\boldsymbol{x}_i, \boldsymbol{x}_j) & 1 \end{pmatrix}\right). \tag{10}$$

This allows us to make use of the multivariate normal distribution (Li et al., 2020). We can keep on adding extra weighted vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_s$, where $\boldsymbol{v}_{\boldsymbol{w}_i} = \boldsymbol{w}_i^T R$, and the hashes will be distributed as multivariate normal (by themselves or in the limit)

$$\begin{pmatrix} v_{i,t} \\ v_{j,t} \\ v_{\boldsymbol{w}_1,t} \\ \vdots \\ v_{\boldsymbol{w}_s,t} \end{pmatrix} \sim N\left(\boldsymbol{0}, \begin{pmatrix} 1 & d(\boldsymbol{x}_i, \boldsymbol{x}_j) & d(\boldsymbol{x}_i, \boldsymbol{w}_1) & \ldots & d(\boldsymbol{x}_i, \boldsymbol{w}_s) \\ d(\boldsymbol{x}_i, \boldsymbol{x}_j) & 1 & d(\boldsymbol{x}_j, \boldsymbol{w}_1) & \ldots & d(\boldsymbol{x}_j, \boldsymbol{w}_s) \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ d(\boldsymbol{x}_i, \boldsymbol{w}_s) & d(\boldsymbol{x}_j, \boldsymbol{w}_s) & d(\boldsymbol{w}_1, \boldsymbol{w}_s) & \ldots & 1 \end{pmatrix}\right) \tag{11}$$

and we can find the control variate correction for special weighted vectors, which we state as a theorem below.

Theorem 2 Suppose $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_s$ are chosen to be pairwise orthogonal. With the estimator $v_i v_j$, we can write the control variate

$$Z = \sum_{s,t} \alpha_{s,t} v_{\boldsymbol{w}_s} v_{\boldsymbol{w}_t} \tag{12}$$

with $\alpha_{s,t} = d(\boldsymbol{x}_i, \boldsymbol{w}_s)d(\boldsymbol{x}_j, \boldsymbol{w}_t) + d(\boldsymbol{x}_i, \boldsymbol{w}_t)d(\boldsymbol{x}_j, \boldsymbol{w}_s)$, with control variate correction $\hat{c} = -\frac{1}{2}$.

We prove this in the supplementary material. In fact, so long $(v_{i,t}, v_{j,t})$ converges to a bivariate normal in distribution, such as *very sparse random projections*, or *Tug-of-War sketch* (Alon et al., 1999; Pratap et al.), we can still apply our result in Theorem 2.

Moreover, we can now apply the following corollary from Kang (2021) which we restate slightly for the case where $(f^{(2)}(v_i, v_j), f^{(2)}(v_k, v_l))$ converge to a bivariate normal distribution to exactly quantify our variance reduction.

Corollary 3 The control variate correction given our choice of control variates $Z = \sum_{s,t} \alpha_{s,t} v_{\boldsymbol{w}_s} v_{\boldsymbol{w}_t}$ gives a variance reduction of $\frac{1}{2}\,\mathrm{Cov}(Y, Z)$.

5.2. Control variates for other distributions of $(f^{(2)}(v_i, v_j), f^{(2)}(v_k, v_l))$

We now look at sign full random projections (Li, 2019) as an example. Sign full random projections aims to estimate the inner product $d(\boldsymbol{x}_i, \boldsymbol{x}_j) := \boldsymbol{x}_i^T \boldsymbol{x}_j$ between two vectors $\boldsymbol{x}_i, \boldsymbol{x}_j$. Suppose we compute and store values $\theta_{i,\boldsymbol{w}}$, $d(\boldsymbol{x}_i, \boldsymbol{w})$, the angle and inner product between all vectors $\boldsymbol{x}_i$ and $\boldsymbol{w}$. Theorem 3 in Li (2019) leads to the estimator $\mathbb{E}[\operatorname{sgn}(v_i)v_j] = \sqrt{\frac{2}{\pi}} d(\boldsymbol{x}_i, \boldsymbol{x}_j) \equiv \mathbb{E}[f^{(2)}(v_i, v_j)]$.

Lemma 4 *Suppose our vectors are normalized. Denote $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to be the dot product between $\boldsymbol{x}_i, \boldsymbol{x}_j$, and $\theta_{i,j}$ to be the angle between $\boldsymbol{x}_i, \boldsymbol{x}_j$. Then we have that*

$$\begin{aligned} &\mathbb{E}[\operatorname{sgn}(v_i)\operatorname{sgn}(v_j) v_k v_l] \\ &= \begin{cases} d(\boldsymbol{x}_k, \boldsymbol{x}_l), & \text{when } i = j, k \neq l, \\ 1, & \text{when } i = j, k = l, \\ d(\boldsymbol{x}_k, \boldsymbol{x}_l)\left(1 - \frac{2\theta_{i,j}}{\pi}\right), & \text{when } i \neq j, k \neq l, \\ \left(1 - \frac{2\theta_{i,j}}{\pi}\right), & \text{when } i \neq j, k = l. \end{cases} \end{aligned} \tag{13}$$

Proof It is easy to see that when $i = j$, $\mathbb{E}[\operatorname{sgn}(v_i)\operatorname{sgn}(v_j) v_k v_l]$ reduces to $\mathbb{E}[v_k v_l]$, which we know the value of from ordinary random projections. When $i \neq j$, we can decompose

$$\begin{aligned} &\mathbb{E}[\operatorname{sgn}(v_i)\operatorname{sgn}(v_j) v_k v_l] = \mathbb{E}[\mathbb{P}[\operatorname{sgn}(v_i) = \operatorname{sgn}(v_j)] v_k v_l] \\ &- \mathbb{E}[\mathbb{P}[\operatorname{sgn}(v_i) \neq \operatorname{sgn}(v_j)] v_k v_l] = \left(1 - \frac{2\theta_{i,j}}{\pi}\right)\mathbb{E}[v_k v_l]. \end{aligned} \tag{14}$$

■

We demonstrate how to find acceptable control variates for sign full random projection using Lemma 4. In the context of Li (2019), only the signs of v_is are stored.

To find acceptable control variates Z, we necessarily need to check that the terms we use in Z result in computable $\operatorname{Cov}(Y, Z)$ and $\operatorname{Var}[Z]$.

Consider the following **cross term**:

$$\operatorname{Cov}(f^{(2)}(v_i, v_i), f^{(2)}(v_j, v_j)) = d(\boldsymbol{x}_i, \boldsymbol{x}_j)\left(1 - \frac{2\theta_{\boldsymbol{x}_i, \boldsymbol{x}_j}}{\pi}\right) \tag{15}$$

which consists of the term $\theta_{\boldsymbol{x}_i, \boldsymbol{x}_j}$ which **we do not know**, and are unable to find (if we could find this, we might as well spend the effort to find $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ instead).

Hence any acceptable control variate Z can never contain **both** of $f^{(2)}(v_i, v_i)$ or $f^{(2)}(v_j, v_j)$ in any scalar multiple, since $\operatorname{Var}[Z]$ requires using the expression $\operatorname{Cov}(f^{(2)}(v_i, v_i), f^{(2)}(v_j, v_j))$.

In fact, if we cycle though the terms in Z, and omit terms which give us an expression of $\operatorname{Cov}(Y, Z)$ or $\operatorname{Var}[Z]$ which we do not know, or unable to compute, we get two potential control variates

$$Z_1 = \alpha_1 f^{(2)}(v_i, v_{\boldsymbol{w}}) + \alpha_2 f^{(2)}(v_{\boldsymbol{w}}, v_j) + \alpha_3 f^{(2)}(v_{\boldsymbol{w}}, v_{\boldsymbol{w}}) \tag{16}$$

$$Z_2 = f^{(2)}(v_{\boldsymbol{w}}, v_{\boldsymbol{w}}) \tag{17}$$

Since variance and covariance are linear operators, we repeatedly apply Lemma 4 to get

$$\begin{aligned}\mathrm{Cov}(Y,Z_1)=&\alpha_1\left(d(\boldsymbol{x}_j,\boldsymbol{w})-\frac{2d(\boldsymbol{x}_i,\boldsymbol{w})d(\boldsymbol{x}_i,\boldsymbol{x}_j)}{\pi}\right)+\alpha_2\left(\left(1-\frac{2\theta_{i,\boldsymbol{w}}}{\pi}\right)-\frac{2d(\boldsymbol{x}_j,\boldsymbol{w})d(\boldsymbol{x}_i,\boldsymbol{x}_j)}{\pi}\right)\\&+\alpha_3\left(\left(1-\frac{2\theta_{i,\boldsymbol{w}}}{\pi}\right)d(\boldsymbol{x}_j,\boldsymbol{w})-\frac{2d(\boldsymbol{x}_i,\boldsymbol{x}_j)}{\pi}\right),\end{aligned} \tag{18}$$

$$\begin{aligned}\mathrm{Var}[Z_1]=&\alpha_1^2\left(1-\frac{2d(\boldsymbol{x}_i,\boldsymbol{w})^2}{\pi}\right)+\alpha_2^2\left(1-\frac{2d(\boldsymbol{x}_j,\boldsymbol{w})^2}{\pi}\right)+\alpha_3^2\left(1-\frac{2}{\pi}\right)\\&+2\alpha_1\alpha_2\left(\left(1-\frac{2\theta_{i,\boldsymbol{w}}}{\pi}\right)d(\boldsymbol{x}_j,\boldsymbol{w})-\frac{2d(\boldsymbol{x}_i,\boldsymbol{w})d(\boldsymbol{x}_j,\boldsymbol{w})}{\pi}\right)\\&+2\alpha_1\alpha_3\left(1-\frac{2\theta_{i,\boldsymbol{w}}}{\pi}\right)+2\alpha_2\alpha_3 d(\boldsymbol{x}_j,\boldsymbol{w}),\end{aligned} \tag{19}$$

and similarly

$$\mathrm{Cov}(Y,Z_2)=\left(\left(1-\frac{2\theta_{i,\boldsymbol{w}}}{\pi}\right)d(\boldsymbol{x}_j,\boldsymbol{w})-\frac{2d(\boldsymbol{x}_i,\boldsymbol{x}_j)}{\pi}\right),\qquad \mathrm{Var}[Z_2]=1-\frac{2}{\pi}. \tag{20}$$

We thus have two control variate estimators of the form $(Y+c(Z_t-\mu_{Z_t}))$

$$\hat{d}_{\text{C.V.}}(\boldsymbol{x}_i,\boldsymbol{x}_j)=\sqrt{\frac{\pi}{2}}\,\mathrm{sgn}(v_i)v_j-\frac{\mathrm{Cov}(Y,Z_t)}{\mathrm{Var}[Z_t]}\,(Z_t-\mathbb{E}[Z_t]) \tag{21}$$

with our variance reduction is still given by $\frac{\mathrm{Cov}(Y,Z_t)^2}{\mathrm{Var}[Z]}$.

We further note that while the term $d(\boldsymbol{x}_i,\boldsymbol{x}_j)$ occurs in $\hat{c}=-\frac{\mathrm{Cov}(Y,Z_t)}{\mathrm{Var}[Z_t]}$, we can can think of some $-\frac{\mathrm{Cov}(Y,Z_1)}{\mathrm{Var}[Z_1]}\equiv g(d(\boldsymbol{x}_i,\boldsymbol{x}_j))$ and hence we can substitute the naive estimate of $\hat{d}(\boldsymbol{x}_i,\boldsymbol{x}_j)$ into $\hat{c}$ as a proxy for the true value of $d(\boldsymbol{x}_i,\boldsymbol{x}_j)$, and use this to compute $\hat{d}_{\text{C.V.}}(\boldsymbol{x}_i,\boldsymbol{x}_j)$.

This means that instead of computing the naive estimate of the k hashed values

$$\hat{d}(\boldsymbol{x}_i,\boldsymbol{x}_j)\approx\sqrt{\frac{\pi}{2}}\left(\sum_{t=1}^{k}\frac{\mathrm{sgn}(v_{it})v_{jt}}{k}\right) \tag{22}$$

we can instead use the approximation

$$\hat{d}_{\text{C.V.}}(\boldsymbol{x}_i,\boldsymbol{x}_j)\approx\sqrt{\frac{\pi}{2}}\left(\frac{\sum_{t=1}^{k}\mathrm{sgn}(v_{it})v_{jt}}{k}\right)+g\left(\sum_{t=1}^{k}\frac{\mathrm{sgn}(v_{it})v_{jt}}{k}\right)(Z-\mu_Z) \tag{23}$$

to give a better estimate of the inner product between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

In the case of $t=1$, we need to find $\underset{\alpha_1,\alpha_2,\alpha_3}{\arg\max}\frac{\mathrm{Cov}(Y,Z_1)^2}{\mathrm{Var}[Z_1]}$ and substitute the estimated value $\sqrt{\frac{\pi}{2}}\,\mathrm{sgn}(v_i)v_j$ into $d(\boldsymbol{x}_i,\boldsymbol{x}_j)$ in the expression Y_1.

On the other hand, instead of substituting the naive estimate of $d(\boldsymbol{x}_i,\boldsymbol{x}_j)$ into our control variate as a proxy for the actual $d(\boldsymbol{x}_i,\boldsymbol{x}_j)$, if we had a closed form version of g^{-1}, then we could instead rearrange Equation (23) and treat $\hat{d}_{\text{C.V.}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \equiv \hat{d}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and express $\hat{d}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in terms of all the other constants we know.

For example, if we do this when $t = 2$, our estimator simplifies to

$$\hat{d}(\boldsymbol{x}, \boldsymbol{x}_j) = \frac{\left(1 - \frac{2}{\pi}\right) \operatorname{sgn}(x) y - \left(1 - \frac{2\theta_{i,\boldsymbol{w}}}{\pi}\right) d(\boldsymbol{x}_j, \boldsymbol{w}) \operatorname{sgn}(\boldsymbol{w}) \boldsymbol{w} - \sqrt{\frac{2}{\pi}}}{\sqrt{\frac{2}{\pi}} - \frac{2}{\pi}\sqrt{\frac{2}{\pi}} - \frac{2}{\pi}\left(\operatorname{sgn}(\boldsymbol{w})\boldsymbol{w} - \sqrt{\frac{2}{\pi}}\right)} \tag{24}$$

We note that there is no guarantee that rearranging the general form of Equation (23) will work for all hashing algorithms, but this is a heuristic which works for most cases.

We give a remark on our control variate estimators. In both the sign full random projection case and the random projection case, we had to look at all possible $v_i v_j$ terms, and compute the covariance between each possible pair. This would allow us to build our weighted control variate estimator which would allow us to “optimize cleanly", in order to get a better variance reduction. Unlike our first MLE technique which allowed us to focus on the five sets and to always get the same cubic always, our weighted control variate estimator will depend on the algorithm involved.

6. Analysis of Estimators

We first note that the weighted vector technique we use rely **on the same hashed information for the weighted vector(s)** for all pairs. In the original hashing method without any weighted vector(s), we would look at k hashed values for $\boldsymbol{x}_i$, and k hashed values for $\boldsymbol{x}_j$. Using our method, while we are further looking at an extra k hashed values for our weighted vector, **these hashed values are always the same regardless of the pair we look at**. Hence the computational time (or additional length) of the hashed weighted vectors just need to be computed once, and is negligible.

The estimators we have presented rely on computing the actual distances between the weighted vectors $\boldsymbol{w}_s$ and $\boldsymbol{x}_i$s in our original dataset. Choosing these weighted vectors carefully are required, to avoid using too many weighted vectors.

For the MLE trick, the choice of weighted vectors depends on the type of hashing algorithm - in particular how $f^{(1)}(\rho_h(\boldsymbol{x}_i, \boldsymbol{x}_j))$ is computed. The analysis of (4) would determine how these weighted vectors are chosen.

Conversely, for the control variate trick, the choice of our control variate terms first depends on the joint distribution of $(f^{(2)}(v_i, v_j), f^{(2)}(v_k, v_l))$. If this distribution converges to the bivariate normal, we can directly apply results from Kang (2021) to choose appropriate weighted vectors, control variate correction, as well as quantify the variance reduction.

On the other hand, if $(f^{(2)}(v_i, v_j), f^{(2)}(v_k, v_l))$ does not, then we have to construct the control variates on our own, similar to Equation 5. To do so, we need to check which terms in Equation 5 we are able to find, and this requires us to compute the expression $\mathbb{E}[f^{(2)}(v_i, v_j) f^{(2)}(v_k, v_l)]$ for any arbitrary i, j, k, l. This allows us to quickly build our control variate expression, and omit terms that we cannot compute. However, this is dependent on the algorithm itself, as we have shown in the sign full random projections case.

Both these cases will allow us to get the variance reduction of $\frac{\text{Cov}(Y,Z)^2}{\text{Var}[Z]}$.

The additional computational time to find the hashed estimates involves either generating the values of Table 2 for binary / discrete data, or adding the control variate correction to each estimate. This could take up to an additional $O(k)$ time in the discrete case, or additional $O(1)$ time (reading stored values of αs from a table) for the control variate case.

7. Our Experiments

We perform our experiments on two publicly available datasets, the MNIST (Lecun et al., 1998) test dataset, as well as the NIPS dataset (Lichman, 2013; Perrone et al., 2016). The MNIST test dataset consists of $n = 10,000$ observations in $\mathbb{R}^{784}$, and the NIPS dataset consists of $n = 5811$ observations in $\mathbb{R}^{11463}$. We center the MNIST dataset, and normalize each vector to have unit length. We convert the NIPS dataset to be binary. We choose our t weighted vectors to correspond to the top t singular vectors for the MNIST dataset, and heuristically choose the weighted vector for the NIPS dataset to be a vector with high resemblance to most vectors. The technical specifications for the workstation we used for our simulations are as follows: Intel E5-2690v3 (2.60GHz, 12 cores) and we run our simulations on parallel cores.

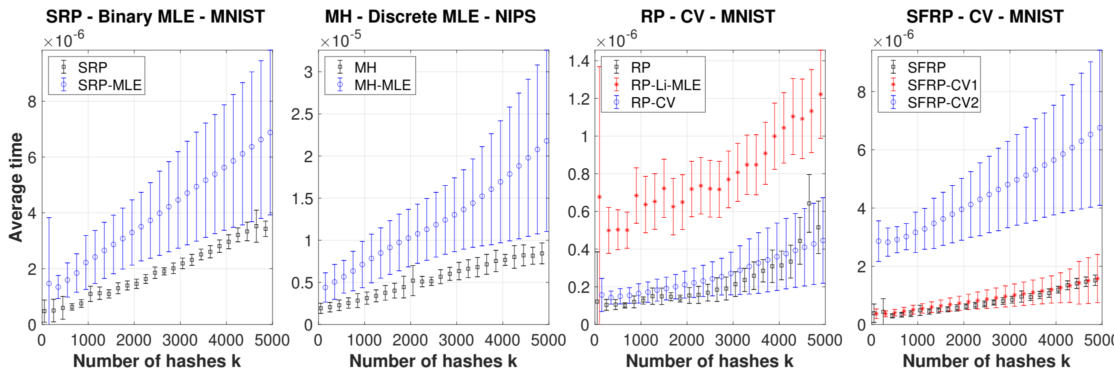


Figure 2: Average time taken for MLE and control variate techniques with selected hashing algorithms.

7.1. Experiment 1 (sanity check): Running time

Proposed methods use the weighted vector to improve estimates of distances given a hashing algorithm, and necessarily will take up additional computational time. This additional computational time is independent of the hashing algorithm, as we assume that we have the stored hashed values of the weighted vector, and the stored known distances between the weighted vectors and the pairs of vectors we want to find the estimate of.

We hence choose to use the following hashing algorithms with one weighted vector: sign random projections (`SRP`) (Charikar, 2002) (binary case MLE), minwise hashing (`MH`) (Broder, 1997) (discrete case MLE), random projection (`RP`) (Indyk and Motwani, 1998) (control variates), and sign full random projection (`SFRP`) (Li, 2019) (control variates) with both control variate estimators discussed above. We compare the time taken for the original estimate without our techniques, and the time taken using our techniques. We also use *Li's MLE algorithm* (`Li-MLE`) for random projections (Li et al., 2006a) as a baseline for the random projection case, as this is an algorithm in hashing literature which also makes use of numerical methods to solve a cubic.

We repeatedly compare the time taken to compute distance estimates of two vectors from the MNIST dataset and the NIPS dataset for 1000 iterations over every k hashes from $k = \{100, 300, \ldots, 4900\}$. Each error bar in Figure 2 was computed at the intervals $k = \{100, 300, \ldots, 4900\}$, but shifted slightly for better comparisons.

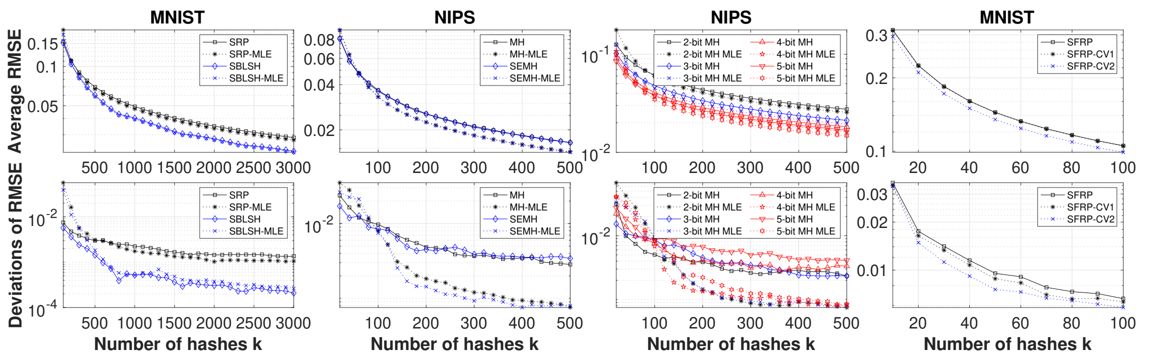


Figure 3: Experiment 2: Average RMSE and deviations for all pairwise estimates with selected hashing algorithms. The average RMSE is of all 49,995,000 pairs for the MNIST test dataset, and 16,880,955 pairs for the NIPS dataset.

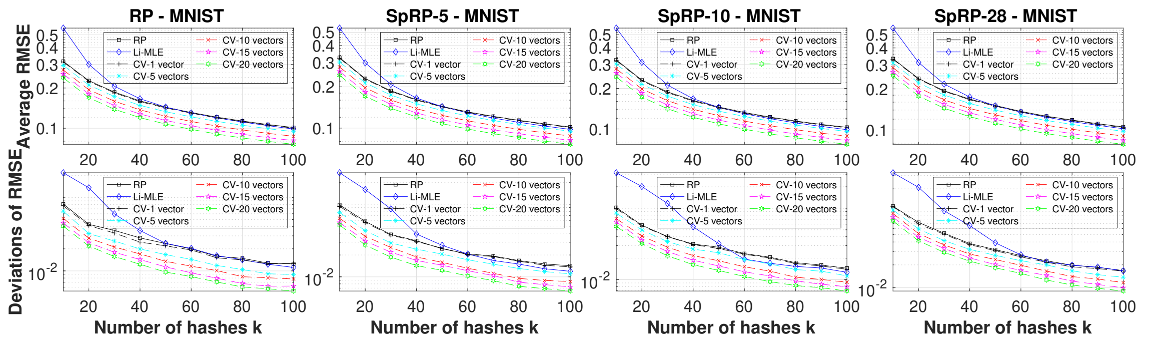


Figure 4: Experiment 3: Average RMSE and deviations for all 49,995,000 pairwise estimates with *random projections* and *very sparse random projections* on the MNIST test dataset. We use $t = \{1, 5, 10, 15, 20\}$ weighted vectors.

It can be seen that while our techniques take more time than the original algorithm, the order of time is roughly the same. We note that our MLE algorithm for the first class of hashing algorithms takes slightly more time, perhaps similar to how Li's MLE algorithm works in the random projection case, but our control variate estimators are faster compared to the MLEs.

7.2. Experiment 2 and 3: Average RMSE with weighted vectors

Experiment 2 looks at the RMSE of all pairwise estimates of distances using a sample of these hashing algorithms from 1997-2019 with one weighted vector: *minwise hashing* (`MH`) (Broder, 1997), *sign random projections* (`SRP`) (Charikar, 2002), *b-bit minwise hashing* (`b-bit`) (Li and König, 2010), *super-bit LSH* (Ji et al., 2012), *simple and efficient weighted minwise hashing* (`SEMH`) (Shrivastava, 2016), and *sign full random projections* (`SFRP`) (Li, 2019). We use dotted lines to denote our techniques. We use different hashes depending on the hashing algorithm. For example, the output of `SRP` is 1 bit, hence each hash value corresponds to 1 bit of information.

Experiment 3 looks at the RMSE of all pairwise estimates of distances for *random projections* (`RP`) (Indyk and Motwani, 1998) and *very sparse random random projections* with scale factor s from $s = \{5, 10, 28\}$ (`SpRP-s`) (Li et al., 2006b) using multiple weighted vectors and to illustrate Theorem 2. We additionally compare our results with Li's MLE (`Li-MLE`) (Li et al., 2006a).

In both experiments, we plot the error bars / standard deviations of the RMSE in separate plots due to difference in magnitudes. In general, our techniques always improve our estimates with lower RMSE or do no worse, albeit perhaps in the binary case where the deviations in our RMSE is almost similar to the original estimate as in Figure 3. Figure 4 shows that one weighted vector does marginally worse than Li's MLE in the long run, but multiple control variates always improves the estimate. If we know the distribution the random variables come from and can come up with appropriate control variate corrections, we can get better and better estimates with more weighted vectors.

8. Discussion and Future Work

We have shown that by treating the estimates of distances in these hashing algorithms as random variables, we can improve these estimates by conditioning on weighted vectors and using statistical techniques.

To reiterate, our work is not to "prove one variance bound for all algorithms", but rather to show that we can **characterize** the types of hashing algorithms in order to allow us to use these statistical techniques. We have shown how we can use the form of $f^{(1)}(\rho_h(\boldsymbol{x}_i, \boldsymbol{x}_j))$, $f^{(2)}(v_i, v_j)$, the distribution of $(f^{(2)}(v_i, v_j)\ f^{(2)}(v_k, v_l))$, and $\mathbb{E}[f^{(2)}(v_i, v_j) f^{(2)}(v_k, v_l)]$ to choose appropriate weighted vectors for variance reduction, as well as to quantify this reduction..

Control variates and MLEs are very generalizable and should new hashing algorithms surface in the literature, these characterizations allow us to apply these techniques to yield better performance (Pratap et al.; Pratap and Kulkarni, 2021) in both computational time and the reduced variance of estimators.

We hope that our work can pave the way for combining future statistical techniques with hashing algorithms, leading to improved accuracy with a slight trade-off in computational speed. Future work can include coming up with a generalizable formula for multiple weighted vectors in the MLE case, or designing the "best" control variates for other hashing algorithms, or even modify them not just based on the distribution involved for the hashing algorithm, but of the data itself.

Acknowledgments

This work is funded by the Singapore Ministry of Education Academic Research Fund Tier 2 Grant MOE2018-T2-2-013. The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

References

Daniel Ahfock, William J Astle, and Sylvia Richardson. Statistical properties of sketching algorithms. arXiv preprint arXiv:1706.03665, 2017.

Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. Journal of Computer and system sciences, 58(1):137–147, 1999.

Andrei Z Broder. On the resemblance and containment of documents. In Compression and Complexity of Sequences 1997. Proceedings, pages 21–29. IEEE, 1997.

Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations (extended abstract). In Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998, pages 327–336, 1998. doi: 10.1145/276698.276781. URL https://doi.org/10.1145/276698.276781.

George Casella and Roger Berger. Statistical Inference. Duxbury Resource Center, June 2001. ISBN 0534243126.

Moses S Charikar. Similarity estimation techniques from rounding algorithms. In Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, pages 380–388. ACM, 2002.

Lianhua Chi and Xingquan Zhu. Hashing techniques: A survey and taxonomy. ACM Computing Surveys (CSUR), 50(1):1–36, 2017.

Kenneth W. Church, Ping Li, and Trevor J. Hastie. Conditional random sampling: A sketch-based sampling technique for sparse data. In In NIPS, pages 873–880, 2006.

Michael B. Cohen, T. S. Jayram, and Jelani Nelson. Simple analyses of the sparse johnson-lindenstrauss transform. In 1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA, pages 15:1–15:9, 2018. doi: 10.4230/OASIcs.SOSA.2018.15. URL https://doi.org/10.4230/OASIcs.SOSA.2018.15.

W Edwards Deming and Frederick F Stephan. On a least squares adjustment of a sampled frequency table when the expected marginal totals are known. The Annals of Mathematical Statistics, 11(4):427–444, 1940.

Piotr Indyk and Rajeev Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM. ISBN 0-89791-962-9. doi: 10.1145/276698.276876. URL http://doi.acm.org/10.1145/276698.276876.

Sergey Ioffe. Improved consistent sampling, weighted minhash and l1 sketching. In Data Mining (ICDM), 2010 IEEE 10th International Conference on, pages 246–255. IEEE, 2010.

Jianqiu Ji, Jianmin Li, Shuicheng Yan, Bo Zhang, and Qi Tian. Super-bit locality-sensitive hashing. In Advances in Neural Information Processing Systems, pages 108–116, 2012.

Jianqiu Ji, Jianmin Li, Shuicheng Yan, Qi Tian, and Bo Zhang. Min-max hash for jaccard similarity. In Data Mining (ICDM), 2013 IEEE 13th International Conference on, pages 301–309. IEEE, 2013.

Daniel M. Kane and Jelani Nelson. Sparser johnson-lindenstrauss transforms. J. ACM, 61 (1):4:1–4:23, 2014. doi: 10.1145/2559902. URL https://doi.org/10.1145/2559902.

Keegan Kang. Correlations between random projections and the bivariate normal. Data Mining and Knowledge Discovery, pages 1–32, 2021.

Keegan Kang and Wei Pin Wong. Improving Sign Random Projections With Additional Information. In Jennifer Dy and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 2484–2492, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/kang18b.html.

S. S. Lavenberg and P. D. Welch. A Perspective on the Use of Control Variables to Increase the Efficiency of Monte Carlo Simulations. Management Science, 27(3):322–335, 1981. ISSN 00251909, 15265501. URL http://www.jstor.org/stable/2631207.

Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In Proceedings of the IEEE, pages 2278–2324, 1998.

Ping Li. Sign-full random projections. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 4205–4212, 2019.

Ping Li and Kenneth W. Church. A Sketch Algorithm for Estimating Two-Way and Multi-Way Associations. Comput. Linguist., 33(3):305–354, 2007. ISSN 0891-2017. doi: 10.1162/coli.2007.33.3.305. URL http://dx.doi.org/10.1162/coli.2007.33.3.305.

Ping Li and Trevor J Hastie. A unified near-optimal estimator for dimension reduction in l-alpha (0<alpha <2) using stable random projections. In Advances in Neural Information Processing Systems, pages 905–912, 2008.

Ping Li and Christian König. b-Bit minwise hashing. In Proceedings of the 19th international conference on World wide web, pages 671–680. ACM, 2010.

Ping Li, Trevor Hastie, and Kenneth Ward Church. Improving Random Projections Using Marginal Information. In Gábor Lugosi and Hans-Ulrich Simon, editors, COLT, volume 4005 of Lecture Notes in Computer Science, pages 635–649. Springer, 2006a. ISBN 3-540-35294-5.

Ping Li, Trevor J. Hastie, and Kenneth W. Church. Very Sparse Random Projections. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, pages 287–296, New York, NY, USA, 2006b. ACM. ISBN 1-59593-339-5. doi: 10.1145/1150402.1150436. URL http://doi.acm.org/10.1145/1150402.1150436.

Ping Li, Michael W Mahoney, and Yiyuan She. Approximating higher-order distances using random projections. In Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, pages 312–321. AUAI Press, 2010.

Y. Li, Z. Kuang, J. Y. Li, and K. Kang. Improving random projections with extra vectors to approximate inner products. IEEE Access, 8:78590–78607, 2020.

M. Lichman. UCI Machine Learning Repository, 2013. URL http://archive.ics.uci.edu/ml.

V. Perrone, P. A. Jenkins, D. Spano, and Y. W. Teh. Poisson Random Fields for Dynamic Feature Models. ArXiv e-prints: 1611.07460, 2016.

Rameshwar Pratap and Raghav Kulkarni. Variance reduction in frequency estimators via control variates method. In Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence, UAI 2020, virtual online, July 27-30, 2021, volume 124 of Proceedings of Machine Learning Research, pages 799–808. AUAI Press, 2021. URL https://auai.org/uai2021/pdf/uai2021.92.pdf.

Rameshwar Pratap, Bhisham Dev Verma, and Raghav Kulkarni. Improving *Tug-of-War* sketch using Control-Variates method, pages 66–76. doi: 10.1137/1.9781611976830.7. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611976830.7.

Rameshwar Pratap, Debajyoti Bera, and Karthik Revanuru. Efficient sketching algorithm for sparse binary data. In Jianyong Wang, Kyuseok Shim, and Xindong Wu, editors, 2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019, pages 508–517. IEEE, 2019. doi: 10.1109/ICDM.2019.00061. URL https://doi.org/10.1109/ICDM.2019.00061.

Florin Rusu and Alin Dobra. Statistical analysis of sketch estimators. In Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pages 187–198. ACM, 2007.

Anshumali Shrivastava. Simple and efficient weighted minwise hashing. In Advances in Neural Information Processing Systems, pages 1498–1506, 2016.

Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for similarity search: A survey. arXiv preprint arXiv:1408.2927, 2014.

Jay Yagnik, Dennis Strelow, David A. Ross, and Ruei-Sung Lin. The power of comparative reasoning. In IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011, pages 2431–2438, 2011. doi: 10.1109/ICCV.2011.6126527. URL https://doi.org/10.1109/ICCV.2011.6126527.