Control Variates for Similarity Search

Jeremy Chew and Keegan Kang

Singapore University of Technology and Design
{jeremy_chew, keegan_kang}@sutd.edu.sg

Abstract. We present an alternative technique for similarity estimation under locality sensitive hashing (LSH) schemes with discrete output. By utilising control variates and extra information, we are able to achieve better theoretical variance reductions compared to maximum likelihood estimation with extra information. We show that our method obtains equivalent results, but slight modifications can provide better empirical results and stability at lower dimensions. Finally, we compare the various methods' performances on the MNIST and Gisette dataset, and show that our model achieves better accuracy and stability.

Keywords: Similarity search · control variates · variance reduction

1 Introduction

Suppose we are given a dataset $X_{n\times p}$ where we want to estimate some similarity measure $\rho(\vec{x}_i, \vec{x}_j) := p_{ij}$ between any two observations (thus denoted as row vectors, $\vec{x}_i$ and $\vec{x}_j$). Computing all pairwise similarities would take at least $O(n^2p)$ time, which is costly when n, p are large.

Locality sensitive hashing (LSH) schemes (such as random projections [1,8], sign random projections [3], and minwise hashing [2]) allow for efficient dimensional reduction of the original dataset X from p-dimensional vectors to k-dimensional vectors [16], $k \ll p$. We construct a hash function $h : \mathbb{R}^p \to \mathbb{R}$ (involving random variables) and compute $h(\vec{x}_i) = v_i, 1 \leq i \leq n$. $\mathbb{P}[v_i = v_j]$ is used to estimate $\rho(\vec{x}_i, \vec{x}_j)$. In practice, we hash k times, and compute $\frac{\sum_{s=1}^{k} 1_{\{v_{is}=v_{js}\}}}{k}$ to find an estimate of $\rho(\vec{x}_i, \vec{x}_j)$. This lowers the computational time for obtaining an estimate of each pairwise similarity from $O(n^2p)$ to $O(n^2k)$, plus the additional pre-processing time required to compute the hashed value of each vector.

Using computational statistics, one can use techniques such as maximum likelihood estimation [13] to estimate pairwise similarities with sign random projections with extra information [11], or control variates to estimate pairwise similarities with random projections [9,10] with extra information. These methods keep to the same order of pre-processing time as the original LSH schemes, but obtain a lower variance than the ordinary estimates using the same number of samples.

2 Our Contributions

Both [11,13] use the MLE technique for variance reduction for a class of LSH schemes similar to sign random projections, where the estimator relies on computing some form of $\frac{\sum_{s=1}^{k} 1_{\{v_{is}=v_{js}\}}}{k}$. However, we show that we can adapt the control variate technique in [9,10], using extra vectors to work with these LSH schemes, to come up with a control variate estimator for our similarity estimates. We show that this estimator obtains the same theoretical variance reduction as the MLE with the additional benefit of increased numerical stability. We also provide a generalised framework that one can use to generate estimators for any arbitrary number of control variates, allowing one to generate additional extra vectors for further variance reduction. Finally, we demonstrate our results via empirical simulations on the MNIST [12] and Gisette datasets [6,14].

3 Review of Preliminary Concepts

We first review the control variate technique. Control variates are used for variance reduction in Monte Carlo simulations [15]. By exploiting the difference between a known estimator and its observed result, the original estimator can be corrected to reduce the error between the observed value and the actual value.

Let X, Y be random variables. Suppose we want to estimate $\mu_X = \mathbb{E}[X]$. If we can find a random variable Y such that $\mu_Y = \mathbb{E}[Y]$ can be analytically calculated, then, $Z = X + c(Y - \mu_Y)$ is also an unbiased estimator for X as $\mathbb{E}[Z] = \mathbb{E}[X] + c(\mathbb{E}[Y] - \mu_Y) = \mu_X$ for any choice of the coefficient c. By choosing $c = -\frac{\mathrm{Cov}(X,Y)}{\mathrm{Var}(Y)}$, we get an expression for the variance of Z as $\mathrm{Var}(Z) = \mathrm{Var}(X) - \frac{(\mathrm{Cov}(X,Y))^2}{\mathrm{Var}(Y)}$ which always results in a reduced variance as long as $\mathrm{Cov}(X, Y)$ is non-zero. This choice of c is the optimal coefficient to minimise the variance [15].

We now explain the extra information idea in [11,10] using Figure 1.

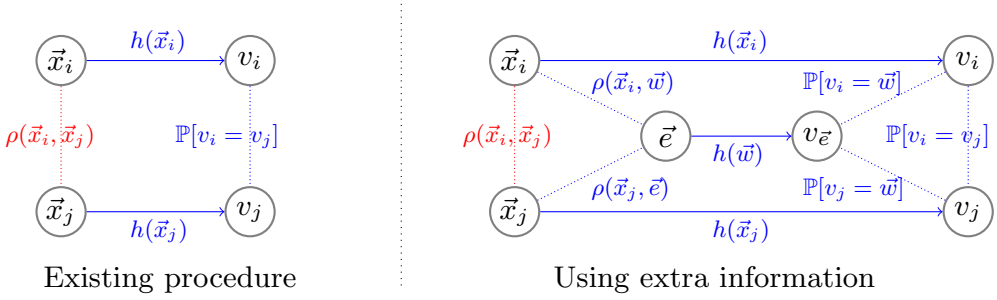

Fig. 1. Using one extra vector $\vec{e}$ for which we know relevant information (shown in blue) to estimate unknown $\rho(\vec{x}_i, \vec{x}_j)$ (shown in red).

In the original LSH scheme, we know the hash function h and its relation to $\mathbb{P}[v_i = v_j]$ (blue). We condition on these information to find $\rho(\vec{x}_i, \vec{x}_j)$ (red).

In the extra information case, we generate an extra vector $\vec{e} \in \mathbb{R}^p$, then compute and store $\rho(\vec{x}_i, \vec{e}), 1 \leq i \leq n$. This takes $O(np)$ time and requires $O(n)$ space. We then compute the hashed values $v_i, 1 \leq i \leq n$ and $v_{\vec{e}}$. Finally, we condition on all of the known information (blue) to estimate $\rho(\vec{x}_i, \vec{x}_j)$ (red).

We now describe what happens in practice where we hash k times. The hashed results are stored in a matrix $Y_{n+1,k}$, where the last row corresponds to the k hashed values $h(\vec{e})$. We denote the last row as Y_e.

We remark that the hashed values can be either discrete [2] or binary [3], depending on the LSH scheme. The MLE approach [11] then considers the following sets for binary and discrete-typed hashes where:

$$A := \{s \mid Y_{is} \neq Y_{js}, Y_{js} = Y_{es}\} \qquad B := \{s \mid Y_{is} = Y_{js}, Y_{js} \neq Y_{es}\} \tag{1}$$
$$C := \{s \mid Y_{is} = Y_{js}, Y_{js} = Y_{es}\} \qquad D := \{s \mid Y_{is} \neq Y_{js}, Y_{js} \neq Y_{es}\} \tag{2}$$

is the collection of indices for binary hashes, $1 \leq s \leq k$ and

$$A := \{s \mid Y_{is} = Y_{js}, Y_{js} = Y_{es}\} \tag{3}$$
$$B := \{s \mid Y_{is} \neq Y_{js}, Y_{is} \neq Y_{es}, Y_{js} = Y_{es}\} \tag{4}$$
$$C := \{s \mid Y_{is} \neq Y_{js}, Y_{is} = Y_{es}, Y_{js} \neq Y_{es}\} \tag{5}$$
$$D := \{s \mid Y_{is} = Y_{js}, Y_{is} \neq Y_{es}, Y_{js} \neq Y_{es}\} \tag{6}$$
$$E := \{s \mid Y_{is} \neq Y_{js}, Y_{is} \neq Y_{es}, Y_{js} \neq Y_{es}\} \tag{7}$$

is the collection of indices for discrete hashes, $1 \leq s \leq k$.

Then, let n_i denote the cardinality of the set i, with $n_A + n_B + n_C + n_D = k$ for binary hashes, and $n_A + n_B + n_C + n_D + n_E = k$ for discrete hashes. Also, let p_i be the probability that an observed element falls in set i.

3.1 Binary hashes

For any given observed cardinalities n_A, n_B, n_C, n_D, the likelihood of such an event, given parameters p_A, p_B, p_C, p_D, would be

$$\mathcal{L}(p_A, p_B, p_C, p_D) = \frac{k!}{n_A! n_B! n_C! n_D!} p_A^{n_A} p_B^{n_B} p_C^{n_C} p_D^{n_D} \tag{8}$$

However, as $\rho(\vec{x}_i, \vec{e})$ have been stored and computed, then we also have the following constraints

$$p_A + p_C = \rho(\vec{x}_j, \vec{e}) := p_{je} \tag{9}$$
$$p_C + p_D = \rho(\vec{x}_i, \vec{e}) := p_{ie} \tag{10}$$
$$p_A + p_B + p_C + p_D = 1 \tag{11}$$

Taking the log-likelihood and substituting in the above constraints, we obtain

$$\begin{aligned} l(p_C) = K &+ n_A \log(p_{je} - p_C) + n_B \log(1 + p_C - p_{ie} - p_{je}) \\ &+ n_C \log(p_C) + n_D \log(p_{ie} - p_C) \end{aligned} \tag{12}$$

$\hat{p}_C$ can thus be expressed as a root of a cubic, as described in [11], and found via numerical methods.

3.2 Discrete hashes

The procedure for discrete hashes is similar. We have the same constraints, but since we now have five variables (instead of four in the binary case), we write our log-likelihood in terms of two variables, say p_A and p_D to obtain

$$\begin{aligned} l(p_A, p_D) = K &+ n_A \log(p_A) + n_B \log(p_{je} - p_A) + n_C \log(p_{ie} - p_A) \\ &+ n_D \log(p_D) + n_E \log(1 - p_{ie} - p_{je} + p_A - p_D) \end{aligned} \tag{13}$$

By calculus, we can express

$$p_D = \frac{n_D(1 + p_A - p_{ie} - p_{je})}{n_D + n_E} \tag{14}$$

and find p_A expressed as a root of a cubic via numerical methods.

However, finding the root of these cubics are not numerically stable for low values of k, which leads us to consider a new estimator.

4 Our Control Variate Estimator: Binary Hashes

Suppose we have generated a random vector $\vec{e} \in \mathbb{R}^p$, and computed and stored the similarities $\rho(\vec{x}_i, \vec{e})$ for each observation, similar to the setup in [11,10]

Now, suppose we want to estimate $p_{ij} = \rho(\vec{x}_i, \vec{x}_j)$ for any pair of observations $\vec{x}_i, \vec{x}_j$, and we have k hashes. We define the following random variables

$$A := \frac{\sum_{s=1}^{k} 1_{\{Y_{is} = Y_{js}\}}}{k} \quad B := \frac{\sum_{s=1}^{k} 1_{\{Y_{is} = Y_{es}\}}}{k} \quad C := \frac{\sum_{s} 1_{\{Y_{js} = Y_{es}\}}}{k} \tag{15}$$

Then, $\mathbb{E}[A] = p_{ij}$, $\mathbb{E}[B] = \rho(\vec{x}_i, \vec{e})$ and $\mathbb{E}[C] = \rho(\vec{x}_j, \vec{e})$. We know $\mathbb{E}[B], \mathbb{E}[C]$ from pre-calculating $\rho(\vec{x}_i, \vec{e})$ and $\rho(\vec{x}_j, \vec{e})$.

We define the following new estimator:

$$A' = A + c_1(B - \mathbb{E}[B]) + c_2(C - \mathbb{E}[C]) \tag{16}$$

A' is an unbiased estimator of p_{ij} and guaranteed to have a lower variance than A as long as either (or both of) $\mathrm{Cov}(A, B)$ and $\mathrm{Cov}(A, C)$ are non-zero.

By finding the partial derivative of $\mathrm{Var}(A')$ and solving for optimal $\hat{c}_1, \hat{c}_2$, we can get the optimal control variate corrections. Thus, we have

$$\begin{aligned} \mathrm{Var}(A') = &\mathrm{Var}(A) + c_1^2\mathrm{Var}(B) + c_2^2\mathrm{Var}(C) + 2c_1\mathrm{Cov}(A, B) + 2c_2\mathrm{Cov}(A, C) \\ &+ 2c_1c_2\mathrm{Cov}(B, C) \end{aligned} \tag{17}$$

The partial derivatives are

$$\frac{\partial \text{Var}(A')}{\partial c_1} = 2c_1\text{Var}(B) + 2\text{Cov}(A,B) + 2c_2\text{Cov}(B,C) \tag{18}$$

$$\frac{\partial \text{Var}(A')}{\partial c_2} = 2c_2\text{Var}(C) + 2\text{Cov}(A,C) + 2c_1\text{Cov}(B,C) \tag{19}$$

Denoting the covariance matrix between B and C as Σ, and packing the covariances $\text{Cov}(A,B), \text{Cov}(A,C)$ as a column vector Σ_A, we can write the equations forming the optimal values of c_1 and c_2 as follows: $[c_1 \;\; c_2]^T = \Sigma^{-1}\Sigma_A$.

Now, the control variate corrections c_1 and c_2 involve the true value of p_{ij} in the calculations of Σ and Σ_A, which we do not have. To get around this, one solution is to obtain an initial estimate for p_{ij} using $\frac{1}{k}\sum_{s=1}^{k} A$ similar to [10]. Another solution involves finding analytic expressions for Σ and Σ_A, collecting the p_{ij} terms on one side, and finally solving the resultant cubic.

Theorem 1. *Suppose we set $g(p_C)$ to be the cubic found after calculus in (12) and $f(p_C)$ to be the cubic we find after our control variate approach. We have that $\frac{f(p_A)}{g(p_A)} = -2$.*

Proof. We can write A, B, C in our estimator as

$$A = \frac{n_B + n_C}{k} \qquad B = \frac{n_C + n_D}{k} \qquad C = \frac{n_A + n_C}{k} \tag{20}$$

We set $A' = p_{ij}$, which we rewrite in terms of p_C by using the result presented in (9)-(11). Furthermore, as A, B, C are Bernoulli random variables, we can easily calculate their expectations, variances and covariances as follows

$$\mathbb{E}[A] = p_{ij} = 1 + 2p_C - p_{ie} - p_{je} \tag{21}$$

$$\text{Var}(A) = p_{ij}(1-p_{ij}) = (1 + 2p_C - p_{ie} - p_{je})(p_{ie} + p_{je} - 2p_C) \tag{22}$$

$$\mathbb{E}[B] = p_{ie} \qquad \mathbb{E}[C] = p_{je} \tag{23}$$

$$\text{Var}(B) = p_{ie}(1-p_{ie}) \qquad \text{Var}(C) = p_{je}(1-p_{je}) \tag{24}$$

$$\text{Cov}(A,B) = p_C - p_{ie}(1 + 2p_C - p_{ie} - p_{je}) \qquad \text{Cov}(B,C) = p_C - p_{ie}p_{je} \tag{25}$$

$$\text{Cov}(A,C) = p_C - p_{je}(1 + 2p_C - p_{ie} - p_{je}) \tag{26}$$

We can then substitute all these rewritten expressions back into (16). By bringing A' over, and cross-multiplying to remove the denominator, we obtain the following equation, which we call $f(p_C)$:

$$\begin{aligned} f(p_C) = \; & -2n_1p_C^3 + 4n_1p_C^2p_{ie} + 2n_1p_C^2p_{je} - 2n_1p_C^2 - 2n_1p_Cp_{ie}^2 - 2n_1p_Cp_{ie}p_{je} \\ & + 2n_1p_Cp_{ie} - 2n_2p_C^3 + 2n_2p_C^2p_{ie} + 2n_2p_C^2p_{je} - 2n_2p_Cp_{ie}p_{je} - 2n_3p_C^3 \\ & + 4n_3p_C^2p_{ie} + 4n_3p_C^2p_{je} - 2n_3p_C^2 - 2n_3p_Cp_{ie}^2 - 6n_3p_Cp_{ie}p_{je} + 2n_3p_Cp_{ie} \\ & - 2n_3p_Cp_{je}^2 + 2n_3p_Cp_{je} + 2n_3p_{ie}^2p_{je} + 2n_3p_{ie}p_{je}^2 - 2n_3p_{ie}p_{je} - 2n_4p_C^3 \\ & + 2n_4p_C^2p_{ie} + 4n_4p_C^2p_{je} - 2n_4p_C^2 - 2n_4p_Cp_{ie}p_{je} - 2n_4p_Cp_{je}^2 + 2n_4p_Cp_{je} \end{aligned} \tag{27}$$

which is exactly equivalent to the $\frac{1}{2}$ of cubic in [11].

5 Our Estimator: Discrete Hashes

Unlike the MLE approach for discrete hashes where we define a new collection of sets, our estimator uses the same random variables defined in the binary case.

We find that we require the true value of p_{ij} to calculate c_1 and c_2. As such, we are presented with the same two options as in the binary case. We can either use $\frac{1}{k}\sum_s^k A$ as an initial estimate for p_{ij}, or we can collect the p_{ij} terms on one side, and solve the root for the resultant cubic. Similarly, this result is equivalent to the cubic for discrete hashes.

Theorem 2. *Suppose we set $g(p_A)$ to be the cubic found after calculus in (13) and $f(p_A)$ to be the cubic we find after our control variate approach. We have that $\frac{f(p_A)}{g(p_A)} = -2n_D - n_E$.*

The equivalency of maximum likelihood estimation and control variates have been explored in [4] and [17], where they show how solving the "constrained Monte Carlo" with non-parametric maximum likelihood estimation coincides with the method of control variates. This implies that the variance reduction for our control variate method is asymptotically equivalent to the variance reduction for maximum likelihood estimation method [11].

However, the control variate method allows us to substitute in a proxy for p_{ij} in our coefficients c_1, c_2 (or even compute the empirical covariance), rather than resorting than solving for a root of a cubic. This is less computationally costly than running Newton-Raphson. Moreover, Theorem 2 gives some (theoretical) insight why a naive implementation of the MLE estimator may not be numerically stable at low values of k, due to the values of n_D and n_E, which may be zero for small k.

6 Application to Sign Random Projections

We demonstrate how our estimator can be used for sign random projections (SRP). In SRP, we want an estimate for the angle θ_{ij} between any two vectors $\vec{x}_i$ and $\vec{x}_j$ [5]. Given a data matrix $X_{n\times p}$, we generate a random matrix $R_{p\times k}$ with entries i.i.d from $N(0,1)$. We next compute $V = \text{sgn}(XR)$ where we define $\text{sgn}(x) = 1_{\{x\geq 0\}}$. The estimate of θ_{ij} is given by $\hat{\theta} = \sum_{s=1}^k \frac{\pi}{k} \cdot 1_{\{V_{is}\neq V_{js}\}}$.

Suppose we now generate an extra vector $\vec{e}$, similar to [11]. Then we only need to make minor modifications to our estimator to have it calculate estimates for the angles between two vectors. Specifically, we have $\mathbb{E}[A] = 1 - \frac{\theta_{ij}}{\pi}, \mathbb{E}[B] = 1 - \frac{\theta_{ie}}{\pi}$, and $\mathbb{E}[C] = 1 - \frac{\theta_{je}}{\pi}$.

Computing $\mathbb{E}[AB] = \mathbb{E}[AC] = \mathbb{E}[BC]$ involves finding the "three-way" probability of A, B, and C. In the case of estimating angles, this is equivalent to having the three vectors $\vec{x}_i$, $\vec{x}_j$, and $\vec{e}$ fall on one side of some hyperplane. This probability is given by $1 - \frac{\theta_{ij}+\theta_{ie}+\theta_{je}}{2\pi}$, with the proof given in [11].

In general, in order to obtain estimates for any LSH scheme, we need to first obtain an expression for the "three-way" probability of A, B, and C. In other

words, we need to be able to calculate an expression for the expectation of the statistic $1_{\{Y_{is}=Y_{js}=Y_{es}\}}$. We discuss more on these in Section 8.

With the above modifications, our estimator is able to directly output an estimate of the angle between any two vectors.

7 Experimental Results

We demonstrate our estimators by using the MNIST test dataset [12] and the Gisette dataset [7,14]. The MNIST test dataset has $n = 10,000$ observations with $p = 784$ parameters. The Gisette dataset has $n = 13,500$ observations with $p = 5000$ parameters. We normalise the vectors in both datasets to unit length. To ensure reproducibility, we use a constant seed for our experiments. Furthermore, we set the extra vector $\vec{e}$ to be the mean vector across all observations in each dataset. The simulations are written in Python, and make use of GPU-accelerated computations through the library PyTorch.

In order to prevent possible biases in picking "good pairs" of vectors, we choose to calculate all possible pairwise estimates for the full dataset. We then calculate the mean-squared error between each estimate and the actual value.

We vary k over the range of $\{10, 20, \ldots, 100\}$, and for k, we calculate and average the results over 100 iterations. For each individual simulation run, we calculate 4 different estimates, namely `CV_SUB`, `CV_CUBIC`, `MLE`, and `SRP`. Respectively, they refer to: using an initial estimate for the default control variate procedure, solving the cubic derived from control variates, solving the cubic derived from MLEs [11], and sign random projections [5].

We compute the average of all mean-squared errors of $49,995,000$ pairwise angular similarity estimates for the MNIST test dataset, as well as the average mean-squared errors of $91,118,250$ pairwise angular similarity estimates for the Gisette dataset. These are displayed in Figures 2 and 3 respectively.

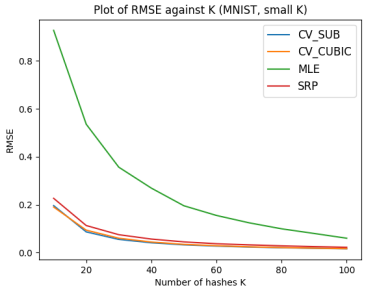

Fig. 2. Averaged MSE for MNIST test dataset

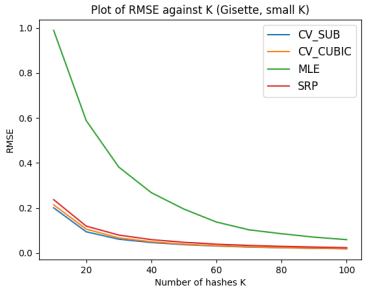

Fig. 3. Averaged MSE for Gisette dataset

In general, we note that `CV_SUB` and `CV_CUBIC` both consistently outperform `SRP` and `MLE`. Furthermore, we also note that `CV_SUB`, that is, simply improv-

ing an initial estimate using control variates, outperforms solving the cubic in `CV_CUBIC`. Finally, although `CV_CUBIC` and `MLE` have been shown to be equivalent in the previous sections, empirical results show that the control variates approach offers greater stability and accuracy compared to the MLE approach.

This is because the calculation for `MLE` relies on n_A, n_B, n_C, n_D. If any of them are zero due to few observations, this may result in a division-by-zero error. Cross-multiplying can help prevent this error, but the numerical stability would still be affected. On the other hand, `CV_SUB` uses the empirical value of the control variate coefficient c. This takes the correlation of the already observed vectors into account, thus giving more accurate and stable results.

We also plot the standard deviation of the mean-squared errors in Figures 4 and 5. We note that `CV_SUB` tends to have lower standard deviations compared to other estimates. We also note that for larger values of k, we can observe lower variances and higher accuracies compared to the baseline of `SRP`.

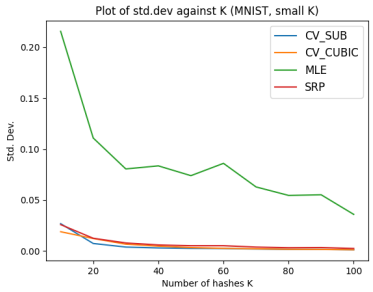

Fig. 4. Standard deviations for MNIST test dataset

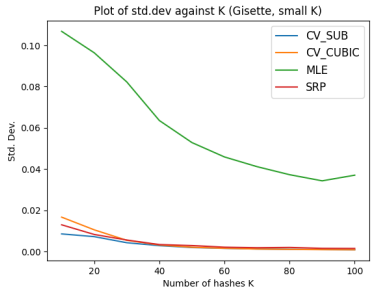

Fig. 5. Standard deviations for Gisette dataset

We also run another set of simulations for a wider range of $k = \{200, 250, \ldots 1000\}$, to show the asymptotic behaviour of the estimators for larger values of k. This time, the results are obtained by averaging over 25 iterations. Again, we calculate these values from all possible pairs of vectors in both the MNIST and Gisette dataset. The results are graphed and displayed in Figures 6 and 7 for the MNIST and Gisette dataset respectively.

Table 1. Average MSE for $k = 50$ and $k = 1000$

	MNIST ($k = 50$)	**Gisette** ($k = 50$)	**MNIST** ($k = 1000$)	**Gisette** ($k = 1000$)
`SRP`	0.0448 ± 0.0153	0.0477 ± 0.00867	0.00216 ± 0.000381	0.002400 ± 0.000498
`MLE`	0.195 ± 0.222	0.195 ± 0.159	0.00165 ± 0.000205	0.00192 ± 0.000107
`CV_SUB`	$\mathbf{0.0324 \pm 0.00731}$	$\mathbf{0.0372 \pm 0.00578}$	$\mathbf{0.00163 \pm 0.000180}$	$\mathbf{0.00192 \pm 0.000119}$
`CV_CUBIC`	0.0345 ± 0.00974	0.0393 ± 0.00621	0.00164 ± 0.00974	0.00192 ± 0.000107

We observe that `MLE` only outperforms `SRP` for larger values of k, but does not match the performance of control variate. We also observe that as k increases, `MLE` asymptotically approaches the same performance as `CV_CUBIC`. This verifies the equivalency of the two approaches as proven in Theorems 1 and 2. Table 1

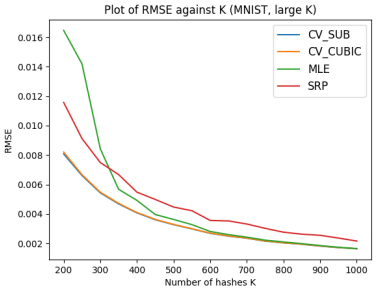

Fig. 6. Averaged MSE for MNIST test dataset

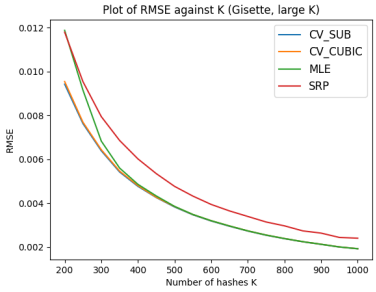

Fig. 7. Averaged MSE for Gisette dataset

displays the averaged MSE (with bounds of 3 standard deviations) for $k = 50$ and $k = 1000$ for both the MNIST and Gisette datasets. We can see that for both datasets, the MSE for `MLE` and `CV_CUBIC` are approximately equal at $k = 1000$, verifying that our two methods are asymptotically identical.

We note that although large values of k give higher accuracies, it does not make sense to use values of k which are similar in magnitude to the original dimensions of the data, p. This is because the computational complexities of our estimators are in the order of $O(n^2k)$, while the computational complexity of calculating each pairwise angle is $O(n^2p)$. Using larger values of k would result in a slower running time, due to the additional overhead of preprocessing the hash values. In this sense, we are only interested in the results for small values of k, but nevertheless, our CV approach still matches/outperforms the MLE approach even for large k.

To verify this, we display the average running time of the various algorithms for $k = \{50, 500, 1000\}$ on the MNIST dataset in Table 2. As expected, for small k, all estimators are faster than directly calculating the angles, but this changes as k increases. We note that `SRP` is the fastest, with our estimator, `CV_SUB` coming in close second. There is a tradeoff between accuracy and efficiency, but because both algorithms keep to the same order of complexity, the time tradeoff is not too significant. On the other hand, due to the need to numerically solve a cubic (through Newton-Raphson), `CV_CUBIC` and `MLE` have a much more noticeable accuracy-efficiency tradeoff. Overall, our results shows that our control variate estimator, not only offers an increase in accuracy from the baseline of `SRP`, but also consistently outperforms the MLE approach proposed by [11].

Table 2. Average time taken (in seconds) for $k = \{50, 500, 1000\}$ on MNIST dataset

	$k = 50$	$k = 500$	$k = 1000$
Direct Calculation	135.3		
`SRP`	11.28	138.4	287.4
`MLE`	99.05	881.5	1805
`CV_SUB`	15.20	146.7	294.9
`CV_CUBIC`	47.38	181.1	335.3

8 Discussion on Control Variates

An added benefit of using control variates is the ability to easily extend to include additional extra vectors. For example, we can generate further extra vectors $\vec{e}_1, \vec{e}_2, \ldots \vec{e}_j$, which can also be used to obtain even more accurate estimates.

For each extra vector e_t, we first pre-compute and store $\rho(\vec{x}_i, \vec{e}_t)$ for all observations. Then we have the following random variables for each extra vector:

$$A := \frac{\sum_s 1_{\{Y_{is}=Y_{js}\}}}{k} \qquad B_t := \frac{\sum_s 1_{\{Y_{is}=Y_{e_t s}\}}}{k} \qquad C_t := \frac{\sum_s 1_{\{Y_{js}=Y_{e_t s}\}}}{k} \tag{28}$$

We can disregard the additional random variables of the form $1_{\{Y_{e_m s}=Y_{e_n s}\}}$ because their covariance with A is 0. In general, for any two random variables to have a non-zero covariance, they must share at least one vector in common. Because we are interested in reducing the variance of A, any random variable we consider should at least share a common vector with A.

Thus, we can define the new estimator of A' of p_{ij} as follows:

$$A' = A + \sum_{t=0}^{j} c_{t,1}(B_t - \mathbb{E}[B_t]) + c_{t,2}(C_t - \mathbb{E}[C_t]) \tag{29}$$

Minimising $\mathrm{Var}(A')$, we obtain the following system of equations for $\vec{c}$:

$$\begin{bmatrix} c_{1,1} & c_{1,2} & \ldots & c_{j,1} & c_{j,2} \end{bmatrix}^T = \Sigma^{-1}\Sigma_A \tag{30}$$

where Σ refers to the covariance matrix of the random vector $\mathbf{X}$ whose elements are $(B_1, C_1, \ldots B_j, C_j)^T$, and Σ_A refers to the column vector whose elements are the covariances of $\mathbf{X}$ with respect to A.

In order to solve the system of equations above, we would need the covariance matrices. Since $\mathrm{Cov}(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$, and we already have $\mathbb{E}[X]$ and $\mathbb{E}[Y]$, we would only need to find expressions for $\mathbb{E}[XY]$.

Now, B_t and C_t are random variables that relate how two vectors are similar to one another. Thus, the expectation $\mathbb{E}[B_i C_j]$ for any i, j rely on which vectors are being referred to. If they share a common vector, such that the three vectors in question are $\vec{x}$, $\vec{y}$ and $\vec{z}$, then the product becomes finding the expectation of the random variable $1_{\{Y_{xs}=Y_{ys}=Y_{zs}\}}$. This is the "three-way" similarity between the three vectors. For sign random projections, this is the probability that the three vectors lie on the same side of a hyperplane.

If the random variables do not share any common vectors, e.g. B_1 and C_2, then, their covariance would equal to 0. This would mean that in the covariance matrix Σ above, there are several entries which we would know to be 0. This makes our covariance matrix sparse, and hence computationally easier to compute the control variate coefficients.

We give an example of the covariance matrix Σ for $j = 4$ extra vectors:

$$\begin{array}{c|cccccccc} & B_1 & C_1 & B_2 & C_2 & B_3 & C_3 & B_4 & C_4 \\ \hline B_1 & v & c & c & 0 & c & 0 & c & 0 \\ C_1 & c & v & 0 & c & 0 & c & 0 & c \\ \hline B_2 & c & 0 & v & c & c & 0 & c & 0 \\ C_2 & 0 & c & c & v & 0 & c & 0 & c \\ \hline B_3 & c & 0 & c & 0 & v & c & c & 0 \\ C_3 & 0 & c & 0 & c & c & v & 0 & c \\ \hline B_4 & c & 0 & c & 0 & c & 0 & v & c \\ C_4 & 0 & c & 0 & c & 0 & c & c & v \end{array} \tag{31}$$

where v, c are the respective variances and covariances to be calculated.

9 Conclusion

We have shown how to use control variates to construct estimators for similarity estimation under LSH schemes. We have also demonstrated how our estimator could be used for sign random projections. The empirical results also show that our control variates estimator outperforms other estimators that use extra vectors to improve accuracy. Furthermore, we have shown how our estimator can easily be extended to include greater numbers of extra vectors, which would otherwise require redefining of contingency tables in other approaches.

We believe that this strategy of using control variates can help improve estimates of vector similarities. The stability and accuracy at even low values of k is an added improvement over other similar approaches. Hence, we believe that our framework of using control variates to achieve variance reduction could be beneficial when both fast computation and high accuracy is wanted.

10 Acknowledgements

This work is funded by the Singapore Ministry of Education Academic Research Fund Tier 2 Grant MOE2018-T2-2-013, as well as with the support of the Singapore University of Technology and Design's Undergraduate Research Opportunities Programme.

The authors also thank the anonymous reviewers for their comments and suggestions for improvement, which has helped to enhance the quality of the paper.

References

1. Achlioptas, D.: Database-friendly Random Projections: Johnson-Lindenstrauss with Binary Coins. J. Comput. Syst. Sci. **66**(4), 671–687 (Jun 2003). https://doi.org/10.1016/S0022-0000(03)00025-4, http://dx.doi.org/10.1016/S0022-0000(03)00025-4
2. Broder, A.Z.: On the resemblance and containment of documents. In: Compression and Complexity of Sequences 1997. Proceedings. pp. 21–29. IEEE (1997)
3. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing. pp. 380–388. ACM (2002)
4. Glynn, P.W., Szechtman, R.: Some new perspectives on the method of control variates. In: Monte Carlo and Quasi-Monte Carlo Methods 2000, pp. 27–49. Springer (2002)
5. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM (JACM) **42**(6), 1115–1145 (1995)
6. Guyon, I., Gunn, S., Ben-Hur, A., Dror, G.: Result Analysis of the NIPS 2003 Feature Selection Challenge. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems 17, pp. 545–552. MIT Press (2005)
7. Guyon, I., Gunn, S.R., Ben-Hur, A., Dror, G.: Result analysis of the nips 2003 feature selection challenge. In: NIPS. vol. 4, pp. 545–552 (2004)
8. Indyk, P., Motwani, R.: Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing. pp. 604–613. STOC '98, ACM, New York, NY, USA (1998). https://doi.org/10.1145/276698.276876, http://doi.acm.org/10.1145/276698.276876
9. Kang, K.: Using the Multivariate Normal to Improve Random Projections. In: Intelligent Data Engineering and Automated Learning – IDEAL 2017: 18th International Conference, Guilin, China, October 30 – November 1, 2017, Proceedings. pp. 397–405. Springer International Publishing, Cham (2017)
10. Kang, K.: Correlations between random projections and the bivariate normal. Data Mining and Knowledge Discovery (May 2021). https://doi.org/10.1007/s10618-021-00764-6, https://doi.org/10.1007/s10618-021-00764-6
11. Kang, K., Wong, W.P.: Improving sign random projections with additional information. In: International Conference on Machine Learning. pp. 2479–2487. PMLR (2018)
12. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
13. Li, P., Hastie, T.J., Church, K.W.: Improving random projections using marginal information. In: International Conference on Computational Learning Theory. pp. 635–649. Springer (2006)
14. Lichman, M.: UCI Machine Learning Repository (2013), http://archive.ics.uci.edu/ml
15. Rubinstein, R.Y., Marcus, R.: Efficiency of multivariate control variates in monte carlo simulation. Operations Research **33**(3), 661–677 (1985)
16. Slaney, M., Casey, M.: Locality-sensitive hashing for finding nearest neighbors [lecture notes]. IEEE Signal processing magazine **25**(2), 128–131 (2008)
17. Szechtman, R., Glynn, P.W.: Constrained monte carlo and the method of control variates. In: Proceeding of the 2001 Winter Simulation Conference (Cat. No. 01CH37304). vol. 1, pp. 394–400. IEEE (2001)