Random Projections With Bayesian Priors

Keegan Kang

Singapore University Of Technology And Design, Singapore 487372,
keegan_kang@sutd.edu.sg

Abstract. The technique of random projection is one of dimension reduction, where high dimensional vectors in $\mathbb{R}^D$ are projected down to a smaller subspace in $\mathbb{R}^k$. Certain forms of distances or distance kernels such as Euclidean distances, inner products [10], and l_p distances [12] between high dimensional vectors are approximately preserved in this smaller dimensional subspace. Word vectors which are represented in a bag of words model can thus be projected down to a smaller subspace via random projections, and their relative similarity computed via distance metrics. We propose using marginal information and Bayesian probability to improve the estimates of the inner product between pairs of vectors, and demonstrate our results on actual datasets.

1 Introduction

Suppose we have N documents and D words in our language. One popular model used to represent these words is the bag of words model. Every document could be represented as a binary vector $\mathbf{x}_i \in \mathbb{R}^D$, where the i^{th} element in the vector represents whether the word is present or not. Instead of using binary vectors, term weighting could also occur where the i^{th} element is some measure of frequency of the i^{th} word appearing in the document, such as square root weighting, log weighting etc.

Having represented such documents as vectors in $\mathbb{R}^D$, appropriate distance metrics could be used to cluster such documents such as the inner product between each pair of vectors. Classification of these documents could also take place via kernels [15] representing higher dimensional features, with the kernel output fed into algorithms like support vector machines (SVMs).

For N documents, computing their pairwise distance under some distance metric would take $O(N^2D)$, which can be computationally intensive when D is large. With a bigram model or trigram model, the dimension of the word vector and eventual resultant computational cost would increase exponentially.

Random projections become a solution to this problem. Suppose the matrix $X_{n\times D}$ represents the matrix of word vectors, where each row is a document vector. Suppose we generate a random matrix $R_{D\times k}$, where each entry r_{ij} is i.i.d. Normal $N(0,1)$, and compute the matrix product $V = XR$. If we consider the row vectors

$$\mathbf{x}_i := (x_{i1}, x_{i2}, \ldots, x_{iD}) \tag{1}$$

$$\mathbf{x}_j := (x_{j1}, x_{j2}, \ldots, x_{jD}) \tag{2}$$

$$\mathbf{v}_i := (v_{i1}, v_{i2}, \ldots, v_{ik}) \tag{3}$$

$$\mathbf{v}_j := (v_{j1}, v_{j2}, \ldots, v_{jk}) \tag{4}$$

we note that

$$\mathbb{E}[v_{i1}v_{j1}] = \mathbb{E}\left[\left(\sum_{s=1}^{D} x_{is}r_{s1}\right)\left(\sum_{s=1}^{D} y_{is}r_{s1}\right)\right] = \mathbb{E}\left[\sum_{s=1}^{D} x_{is}y_{is}r_{s1}^2 + \sum_{s\neq t} x_{is}y_{jt}r_{s1}r_{t1}\right] = \langle \mathbf{x}_i, \mathbf{y}_i \rangle \quad (5)$$

Therefore, by the Law of Large Numbers we have that

$$\frac{\langle \mathbf{v}_i, \mathbf{v}_j \rangle}{k} \approx \langle \mathbf{x}_i, \mathbf{x}_j \rangle \quad (6)$$

and sharp probability bounds [18] on the error can be given, which we give here without proof

$$\mathbb{P}\left[(1-\varepsilon)\langle \mathbf{x}_i, \mathbf{x}_j \rangle \leq \frac{\langle \mathbf{v}_i, \mathbf{v}_j \rangle}{k} \leq (1+\varepsilon)\langle \mathbf{x}_i, \mathbf{x}_j \rangle\right] \leq 1 - 4\exp\left\{-(\varepsilon^2 - \varepsilon^3)k/4\right\} \quad (7)$$

This bound can therefore be used to choose a value of k, such that the error in the estimate is no more than $(1+\varepsilon)$ away from the true value, which is independent of D, the original dimensions of the data.

Furthermore, random projections can also be used with other distance metrics such as the l_p distance (for even p) [12], or even with kernels such as the polynomial kernel [15] to get close estimates as well.

The time taken under random projections is $O(NDk + N^2k)$, where $O(NDk)$ of time is taken to do the matrix multiplication XR, and N^2k to compute the respective distances between vectors. When $k \ll D$, the time saving is substantial.

Random projection can therefore be seen as a method to effectively deal with large vectors [2]. In natural language processing (NLP) which is computationally intensive, other concerns such as *accuracy*, *speed*, and *storage* arise.

Speeding up the computation of the vectors $\mathbf{v} \in \mathbb{R}^k$ has been done in the last decade, by changing the structure of the random projection matrices. The r_{ij} in the random projection matrix R could come from i.i.d. distributions (binary random projections [1], very sparse random projections [11]), or correlated r_{ij} (fast Johnson Lindenstrauss transform [3], subsampled randomized Hadamard transform [5]).

Reducing the storage of the vectors $\mathbf{v} \in \mathbb{R}^k$ has also been done with sign random projections [10]. The output matrix V would consist only of 1s and 0s, where each entry takes up 1 bit of space. Otherwise, storing the entries in $\mathbf{v}$ as doubles would take up 64 bits of space per entry.

The accuracy of the random projection estimates is not just limited to computing the probability bounds of the estimates. Statistical theory can also help to sharpen these probability bounds, and hence get better estimates. For example, previous work by Li [10] used marginal information about the data, while our previous work [8] used variance reduction techniques in Monte Carlo methods.

Our method proposes using a Bayesian estimator of the inner product by assuming a prior distribution on the data. We show how to compute this estimator, and also demonstrate its effectiveness.

2 Related Work

Consider the row vectors $\mathbf{v}_i$, $\mathbf{v}_j$ in (3) and (4). Without loss of generality, suppose we normalize our word vectors $\mathbf{x}_i$ where $\|\mathbf{x}_i\|_2^2 = 1$, and we let $\{(v_1, v_2)\}_{s=1}^k$ denote the tuples (v_{is}, v_{js}). Then each tuple can be represented as

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & a \\ a & 1 \end{pmatrix}\right) \tag{8}$$

where a denotes the true inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$. Here, the tuple is seen as being drawn from a bivariate normal. In statistical theory, this can be seen as estimating an unknown correlation of a bivariate normal when both variances are known [6].

In 2006, Li [10] used a maximum likelihood estimator to improve the estimate of the inner product. Given observations of k tuples $(v_1, v_2)^{(1)}, (v_1, v_2)^{(2)}, \ldots, (v_1, v_2)^{(k)}$, Li derived the likelihood equation in terms of the inner product a, and found the value of $\hat{a}$ which maximized the likelihood.

More recently in 2016, our previous work [8] used control variates to improve the estimate of quadratic forms, one of which consisted the inner product. By using random variables where the true mean θ was known, we used a control variate correction to reduce the error of the estimate of the inner product.

Both our work and Li achieved the same improvement of the estimate of the inner product. With normalized data, the ordinary inner product estimate (6) had a variance

$$\text{Var}[a] = 1 + a^2 \tag{9}$$

but under our estimate and Li's estimate, the reduced variance was

$$\text{Var}[a] = \frac{(1-a^2)^2}{1+a^2} \tag{10}$$

We use the variance of the inner product here instead of probability bounds, since the derivation of the bounds make use of the variance, and the variance can be used as a proxy to how tight the probability bounds are.

Figure 1 shows the relationship between the original variance, our variance, and Li's variance. We see that substantial variance reduction occurs when vector pairs are correlated with $|a|$ near 1, but otherwise little variance reduction when a is near 0.

3 Inherent Challenges With High Dimensional Word Vectors

Geometry theory [4] states that as the dimension p increases, any two randomly drawn vectors are with high probability almost mutually orthogonal. While word vectors (and data in general) usually have some structure and are not seen as randomly drawn, most pairwise inner products a tend to be small in magnitude but non-zero in extremely high dimensions.

Based on Figure 1, both our technique and Li's technique would work well on datasets where a high proportion of vector pairs $\mathbf{x}_i, \mathbf{x}_j$ are heavily correlated. Otherwise, there would be no advantage is using their techniques.

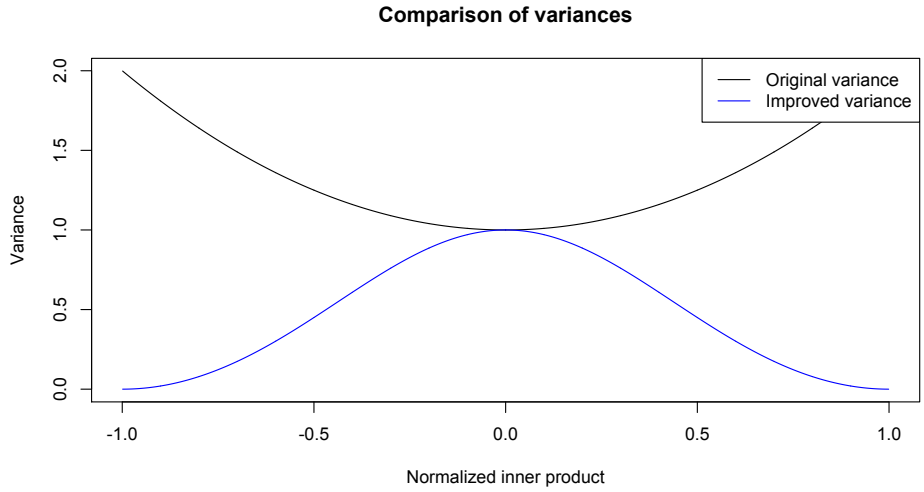

Fig. 1. Comparison of original variance and improved variance under both our estimate and Li's estimate. Both our estimate and Li's estimate have the same variance (in blue). Recall that that if the inner product between two vectors is 1, they are in the same direction. If the inner product is -1, there are in opposite directions. If the inner product is 0, they are orthogonal.

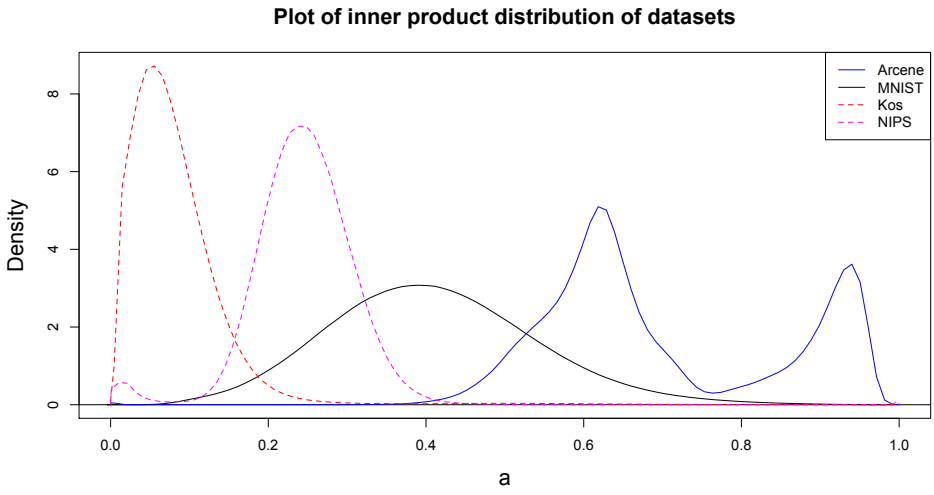

Fig. 2. Distribution of pairwise inner products for Arcene, MNIST, Kos blog posts, NIPS conference papers datasets.

For example, consider the following four high dimensional datasets, two of which are document-word datasets (*Kos blogposts* [13] , *NIPS conference papers* [17]), and two non document-word datasets (*Arcene* [7], *MNIST* [9]). We use log term weighting for the *Kos blogposts* and *NIPS conference papers* datasets.

Figure 2 shows the distribution of true pairwise inner products for these datasets. The document-word datasets have a significantly higher proportion of pairwise inner products near 0 compared to the non document-word datasets, hence our algorithm and Li's algorithm would not be useful at all. Relying on basic principal component analysis (PCA) to reduce the dimension of data would also not be of much help. We show the following scree plots of the same four datasets.

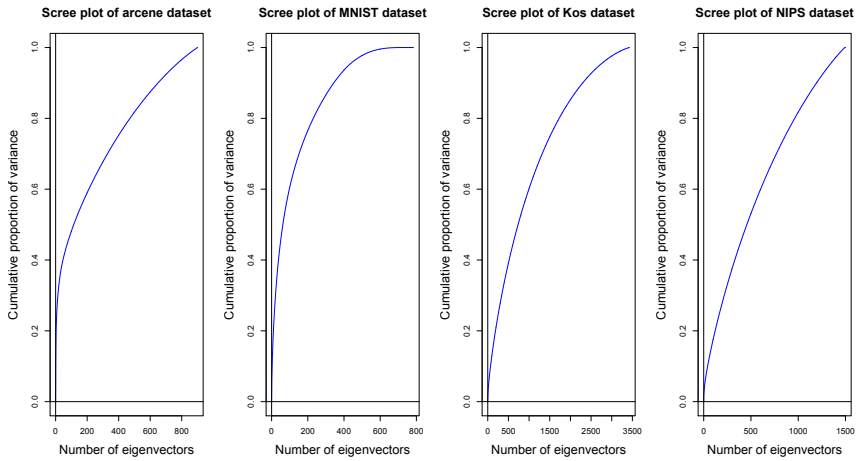

Fig. 3. Scree plots of arcene, MNIST, Kos blogposts, and NIPS conference papers datasets.

We can see from Figure 3 that the first few principal components of *arcene* and *MNIST* account for a significant proportion of the variance. However, the proportion of variance for the *Kos blogposts*, *NIPS conferences* datasets increases linearly with the number of principal components. Therefore, it is difficult to adopt a strategy similar to our previous work (using a control variate of a known quantity) or Li (using marginal information) with knowledge of the first few eigenvectors. Even if PCA produces good results (several principal components accounting for most of the variance), there is usually a loss of actual distance information.

We therefore proceed in an orthogonal direction to PCA as well as our method and Li's method and look at Bayes Rule for inspiration.

4 Our Contributions

In this paper, we propose a method which estimates the inner product of vector word pairs which has lower variance than either Li's method and our previous method. We analyze the time taken for our method. We also evaluate and demonstrate our method on actual datasets.

4.1 Bayes Rule and Information Updating

Theorem 1. *Bayes Rule*
Let A, B be any two events, and $\mathbb{P}[B] = 0$. Then

$$\mathbb{P}[A|B] = \frac{\mathbb{P}[B|A]\mathbb{P}[A]}{\mathbb{P}[B]} \tag{11}$$

Intuitively, suppose we had some prior belief about A, which we encode as $\mathbb{P}[A]$. In our NLP framework, this can be how we believe the (true) inner product a between pairs of vectors are distributed. Then for some observations B which we see, we can update our initial belief via $\mathbb{P}[B|A]$. Similarly, our observations B in this context would be our tuples $\{(v_1, v_2)\}_{s=1}^{k}$. The posterior $\mathbb{P}[A|B]$ describes our final belief, based on our observations. We want this to be our inner product estimate $\hat{a}$.

4.2 Our Proposed Algorithm

Using Bayes Rule, our estimate for the inner product a can be written as

$$\mathbb{E}[a \mid \{(v_1, v_2)\}_{s=1}^{k}] = \frac{1}{K} \int_0^1 a \; p(a) \; L(a) \; da \tag{12}$$

where

$$p(a) = \text{prior belief of how inner product is distributed} \tag{13}$$

$$L(a) = \text{likelihood based on observations } \{(v_1, v_2)\}_{s=1}^{k} \tag{14}$$

$$\frac{1}{K} = \text{the normalizing constant to make } \int_0^1 \frac{p(a)L(a)}{K} \; da$$
$$\text{be a probability density function (pdf) and integrate to 1} \tag{15}$$

We can express the likelihood function of the bivariate normal as

$$L(a) \propto \left(1 - a^2\right)^{-k/2} \exp\left\{ - \sum_{s=1}^{k} \frac{\begin{pmatrix} v_{1s} & v_{2s} \end{pmatrix} \begin{pmatrix} 1 & -a \\ -a & 1 \end{pmatrix} \begin{pmatrix} v_{1s} \\ v_{2s} \end{pmatrix}}{2(1-a^2)} \right\} \tag{16}$$

$$\propto \left(1 - a^2\right)^{-k/2} \exp\left\{ - \frac{\sum_{s=1}^{k} v_{1s}^2 + v_{2s}^2 - 2a v_{1s} v_{2s}}{2(1-a^2)} \right\} \tag{17}$$

We do not need the exact form of $L(a)$, since the constant can be part of $\frac{1}{K}$. We now consider the prior distribution $p(a)$.

4.3 Computing the Integral $\int_0^1 a\ p(a)\ L(a)\ da$

If we treat the distribution of pairwise inner products (Figure 2) as a probability distribution $p(a)$, statistical theory tells us that we can get a good idea of the distribution by sampling from it. In this paper, we recommend sampling $\left\lfloor \sqrt{\frac{N(N-1)}{2}} \right\rfloor$ pairwise inner products out of $\frac{N(N-1)}{2}$ total pairwise inner products to get convergence to a good prior.

We do not place any assumptions on $p(a)$ being a commonly known distribution like the beta distribution (Kos blog posts, NIPS conference papers datasets), or a bivariate mixture model distribution (Arcene dataset). Rather, we use kernel density estimators [16] such as the Nadaraya Watson estimator to evaluate $p(a)$ at several equally spaced points from the interval $[0,1]$.

The motivation for doing so is that we only pass the value of $p(a)$ into a numerical integration algorithm to compute $\int_0^1 a\ p(a)\ L(a)\ da$. However, numerical integration algorithms work by evaluating the integral at equally spaced points. Therefore, we do not need a closed form expression of $p(a)$, and the evaluation of $p(a)$ at these equally spaced points suffices.

We give the following example using Simpson's Rule as our numerical integration algorithm. Suppose we consider the interval $[0,1]$ with equally spaced points $a_0 := 0, a_1, a_2, \ldots, a_{2s} := 1$. Suppose we let α denote the length of the interval $[a_i, a_{i+1}]$. Further suppose we use the Nadaraya Watson estimator to evaluate $p(a)$ at the points $a_0, a_1, \ldots, a_{2s}$. Using Simpson's Rule gives

$$\begin{aligned}\int_0^1 a\ p(a)\ L(a)\ da &= \int_0^{a_2} a\ p(a)\ L(a)\ da + \int_{a_2}^{a_4} a\ p(a)\ L(a)\ da + \ldots + \int_{a_{2s-2}}^{1} a\ p(a)\ L(a)\ da \\ &= \frac{2\alpha}{6}\left[f(0) + 4f(a_1) + f(a_2)\right] + \ldots + \frac{2\alpha}{6}\left[f(a_{2s-2}) + 4f(a_{2s-1}) + f(1)\right] \\ &= \frac{2\alpha}{6}\sum_{t=1}^{s}\left[f(a_{2t-2}) + 4f(a_{2t-1}) + f(a_{2t})\right] \end{aligned} \tag{18}$$

and do likewise for $\int_0^1 p(a)\ L(a)\ da$.

The only other assumption we make is that the distribution of inner products is relatively smooth, so we need a small number of equally spaced points.

Theorem 2. *The expected variance of the inner product estimate given by our algorithm in (12) is smaller than the original variance of the inner product estimate.*

Proof. From the law of total variance, we have

$$\mathrm{Var}[a] = \mathbb{E}\left[\mathrm{Var}[a \mid \{(v_1, v_2)\}_{s=1}^{k}]\right] + \mathrm{Var}\left[\mathbb{E}[a \mid \{(v_1, v_2)\}_{s=1}^{k}]\right] \tag{19}$$

and rearranging this, we have

$$\mathrm{Var}\left[\mathbb{E}[a \mid \{(v_1, v_2)\}_{s=1}^{k}]\right] = \mathrm{Var}[a] - \mathbb{E}\left[\mathrm{Var}[a \mid \{(v_1, v_2)\}_{s=1}^{k}]\right] \tag{20}$$

$$\Rightarrow \mathrm{Var}\left[\mathbb{E}[a \mid \{(v_1, v_2)\}_{s=1}^{k}]\right] < \mathrm{Var}[a] \tag{21}$$

This does not imply that the actual variance of our inner product estimate is always lower than the variance of the original estimate, but in practice this is usually the case if we sample enough inner products.

4.4 Numerical Analysis of Algorithm and Time Taken

To compute our improved estimate of the inner product, this is equivalent to computing the quotient of two integrals

$$\frac{\int_0^1 a\ p(a)\ L(a)\ da}{\int_0^1 p(a)\ L(a)\ da} \tag{22}$$

where the denominator is the normalizing constant K. We need to use numerical integration tools like Simpson's Rule or Gauss-Hermite Quadrature as shown in the previous subsection.

The time taken for this algorithm thus depends on the number of inner products sampled, as well as the number of equally spaced points to compute the kernel density estimates.

We first consider the pre-processing period. Sampling $\left\lfloor \sqrt{\frac{N(N-1)}{2}} \right\rfloor$ inner products is of order $O(ND)$. Evaluating $p(a)$ at s equally spaced points is of order $O(Ns)$. Computing the matrix product $V = XR$ is of order $O(NDk)$. The overall pre-processing cost is therefore $O(NDk + ND + Ns)$.

We need to do numerical integration to compute the inner product estimate after random projection. For each inner product pair, it takes $O(k)$ time (and $O(1)$ space) to compute and store the constant terms in (17). We then take $O(s)$ time to compute the integral for our estimate, as in (17). The total time taken for all pairwise estimates is therefore $O(N^2(k+s))$.

The overall time taken for our algorithm is $O(NDk + ND + Ns + N^2(k+s)) = O(NDk + N^2(k+s))$. Ordinary random projections, Li's method, and Kang's method all take $O(NDk + N^2k)$, and are therefore faster.

However, we will show in our experiments that while our algorithm takes an extra $O(N^2s)$ time, the tradeoff in accuracy is worth the decrease in speed.

We further note that our algorithm does not necessarily need to be used for estimating the inner product. We can also estimate other distances such as l_p distances, but modify $p(a), L(a)$ respectively.

5 Our Experiments

We look at three word datasets for our experiments. They are the *Kos blogposts* [13] ($N = 3,430$, $D = 6,906$, $\tilde{p} = 5,880,735$), *NIPS conference papers* [13] ($N = 1,500$, $D = 12,419$, $\tilde{p} = 1,124,250$), and the *MSN web crawl* dataset [11] ($N = 2,702$, $D = 65,536$, $\tilde{p} = 3,649,051$), where $\tilde{p}$ is the number of pairwise inner products. We use log term weighting for our data, before normalizing each word vector to have norm 1.

For each dataset, we compute the average root mean square error (RMSE) of the inner product estimates over all pairs for 1000 iterations. We project down from D dimensions to $k = \{10, 20, \ldots, 100\}$ with our random projection matrix. For each iteration, we randomly sample an increasing number of pairwise inner products up to the nearest power of 2 greater than $\sqrt{\tilde{p}}$ to show how this affects our kernel density estimate of $p(a)$ and hence our inner product estimate. We evaluate our integral at 101 equally

spaced points. We compare the estimates of our RMSE against ordinary random projections, Li's method, and our control variate method, and compute error bounds of 3 standard deviations for each RMSE estimate.

Since we normalize our data to have norm of 1, the RMSE becomes a number between 0 and 1. If the RMSE is near 0, it means that most estimates of pairwise inner products are close to the truth. Otherwise, if the RMSE is near 1, it means that most estimates of the inner product are far from the truth.

The code for our experiments can be found at `https://github.com/erwin4d/research_projects/`.

Figures 4, 5, and 6 show the results of our experiments. The left plots in these figures shows the average RMSE as the number of columns k increases. The right plots in these figures shows the average RMSE at $k = 100$, but with increasing number of inner products sampled to compute the kernel density estimate $p(a)$ as a proportion of $\lfloor\sqrt{\tilde{p}}\rfloor$. Note that the scale on the x-axis is logarithmic. The horizontal dotted lines and the crosses represent 3 standard deviations around the RMSE. The red vertical dotted line represents $\lfloor\log\tilde{p}\rfloor$ and the blue vertical dotted line represents $\lfloor\sqrt{\tilde{p}}\rfloor$.

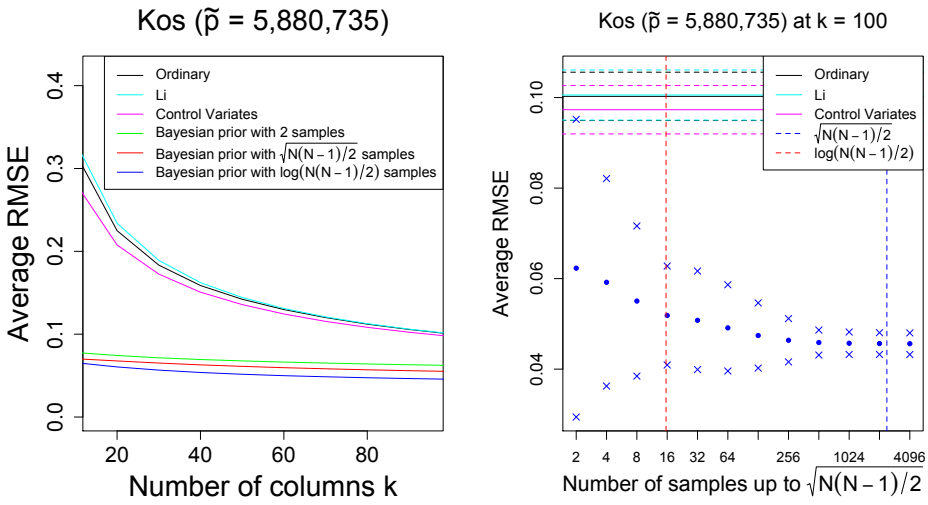

Fig. 4. The plots show the average RMSE of the inner product estimates over all 5,880,735 pairs for the Kos blogposts dataset.

From our results, we can see that Li's method and our previous control variate method performs similarly to ordinary random projections, our Bayesian prior method has a significantly lower RMSE as k increases. This is not surprising, as the proportion of highly correlated pairwise inner products in these datasets is small compared to the proportion of weakly correlated pairwise inner products.

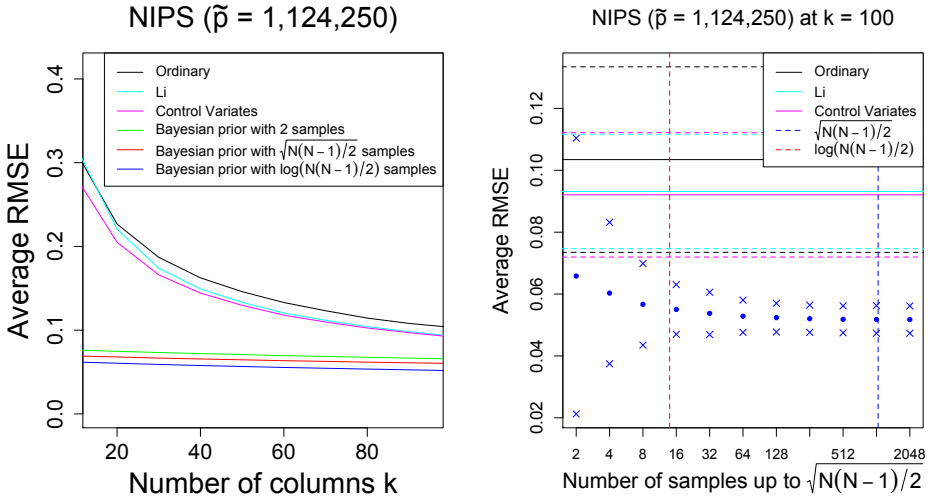

Fig. 5. The plots show the average RMSE of the inner product estimates over all 1,124,250 pairs for the NIPS conference papers dataset.

Furthermore, we see that sampling $\lfloor\sqrt{\tilde{p}}\rfloor$ inner products for the kernel density estimate of our prior $p(a)$ can be too conservative, as there is "convergence" in the average pairwise RMSE for much fewer samples. Even before this "convergence", taking $\lfloor\log\tilde{p}\rfloor$ samples gives a significantly lower RMSE with smaller standard deviations than ordinary random projections, Li's method, and the control variate method. Nevertheless, we keep the number of samples fixed at $\lfloor\sqrt{\tilde{p}}\rfloor$ as a rule of thumb just to be safe, as other datasets might not have a smooth distribution of inner products.

Intuitively the performance of our algorithm on these datasets can be seen as an updating process based on the observation of the tuples (v_{is}, v_{js}), using Bayes' Rule. Thus, we end up updating our beliefs based on our observations, hence we get more accurate estimates and lower RMSE provided we started with the right beliefs.

6 Conclusion and Future Work

NLP deals with extremely high dimensional sparse data which can be computationally intensive. Dimension reduction techniques are essential, but current dimension reduction techniques may not work well for word vectors. Figures 1 and 2 show how random projection techniques may not work well for word vectors due to the distribution of inner products. Figure 3 also shows why PCA may also not be useful in reducing the number of dimensions with certain word datasets.

We have demonstrated an algorithm using Bayes Rule which performs well in estimating the inner product between all pairwise word vectors rather than "good correlated pairs" for small K, provided we have some notion of the underlying distribution of the

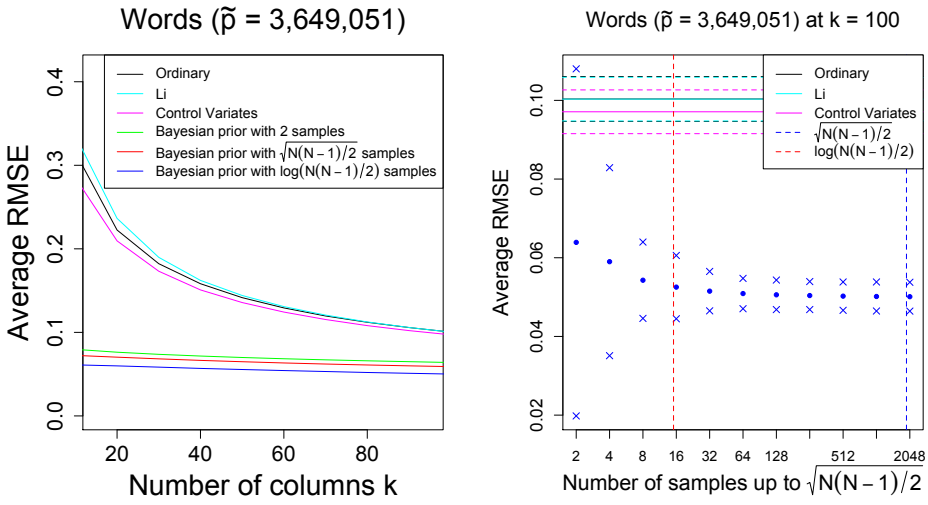

Fig. 6. The plots show the average RMSE of the inner product estimates over all 3,649,051 pairs for the words dataset.

inner product. While our algorithm takes $O(N^2(k+s))$ time in estimating all pairwise inner products compared to $O(N^2k)$ time, the average RMSE is significantly lower for the same value of k. In fact, if we picked some k for ordinary random projections, we can set $\hat{k} = k - s$ for our Bayesian prior algorithm to keep to the same runtime, and yet get smaller error. From our results, using $k = 10$ for our Bayesian prior algorithm still yields a much lower RMSE than $k = 100$ for Li's algorithm, our control variate algorithm, or ordinary random projection estimates.

We have also given a recipe on how to use our algorithm with different estimates of distances, not just the inner product. The likelihood function $L(a)$ necessarily needs to change based on the distance estimate.

We believe that there is potential synthesis of this random projection algorithm together with existing and future NLP works, such as [2, 14], leading to better and more accurate results.

7 Acknowledgements

We thank the reviewers who provided us with many helpful comments. We hope we have addressed most of these comments in this version of the paper where possible. This research was supported by the SUTD Faculty Fellow Award.

References

1. Achlioptas, D.: Database-friendly Random Projections: Johnson-Lindenstrauss with Binary Coins. J. Comput. Syst. Sci. 66(4), 671–687 (Jun 2003), http://dx.doi.org/10.1016/

S0022-0000(03)00025-4
2. Agustí, P., Traver, V.J., Pla, F.: Bag-of-words with aggregated temporal pair-wise word co-occurrence for human action recognition. Pattern Recogn. Lett. 49(C), 224–230 (Nov 2014), http://dx.doi.org/10.1016/j.patrec.2014.07.014
3. Ailon, N., Chazelle, B.: The Fast Johnson-Lindenstrauss Transform and Approximate Nearest Neighbors. SIAM J. Comput. 39(1), 302–322 (May 2009), http://dx.doi.org/10.1137/060673096
4. Ball, K.: An elementary introduction to modern convex geometry. Flavors of geometry 31, 1–58 (1997), http://library.msri.org/books/Book31/files/ball.pdf
5. Boutsidis, C., Gittens, A.: Improved matrix algorithms via the subsampled randomized hadamard transform. CoRR abs/1204.0062 (2012), http://arxiv.org/abs/1204.0062
6. Fosdick, B.K., Raftery, A.E.: Estimating the correlation in bivariate normal data with known variances and small sample sizes. The American Statistician 66(1), 34–41 (2012), http://EconPapers.repec.org/RePEc:taf:amstat:v:66:y:2012:i:1:p:34-41
7. 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), http://papers.nips.cc/paper/2728-result-analysis-of-the-nips-2003-feature-selection-challenge.pdf
8. Kang, K., Hooker, G.: Random projections with control variates. In: Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM,. pp. 138–147. INSTICC, ScitePress (2017)
9. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE. pp. 2278–2324 (1998)
10. Li, P., Hastie, T., Church, K.W.: Improving Random Projections Using Marginal Information. In: Lugosi, G., Simon, H.U. (eds.) COLT. Lecture Notes in Computer Science, vol. 4005, pp. 635–649. Springer (2006)
11. Li, P., Hastie, T.J., Church, K.W.: Very Sparse Random Projections. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 287–296. KDD '06, ACM, New York, NY, USA (2006), http://doi.acm.org/10.1145/1150402.1150436
12. Li, P., Mahoney, M.W., She, Y.: Approximating higher-order distances using random projections. CoRR abs/1203.3492 (2012), http://arxiv.org/abs/1203.3492
13. Lichman, M.: UCI machine learning repository (2013), http://archive.ics.uci.edu/ml
14. Maqueda, A.I., Ruano, A., del Blanco, C.R., Carballeira, P., Jaureguizar, F., Garca, N.: Novel multi-feature bag-of-words descriptor via subspace random projection for efficient human-action recognition. In: 2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). pp. 1–6 (Aug 2015)
15. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. The MIT Press (2012)
16. Nadaraya, E.A.: On estimating regression. Theory of Probability & Its Applications 9(1), 141–142 (1964)
17. Perrone, V., Jenkins, P.A., Spano, D., Teh, Y.W.: Poisson random fields for dynamic feature models (2016), arXiv e-prints: 1611.07460
18. Vempala, S.S.: The Random Projection Method, DIMACS series in discrete mathematics and theoretical computer science, vol. 65. Providence, R.I. American Mathematical Society (2004), http://opac.inria.fr/record=b1101689, appendice p.101-105