Chapter 12
Applying James–Stein Estimation to b-Bit Minwise Hashing

Jing En Daniel Toh, Rui Xian Matthew Kan, and Keegan Kang

Abstract b-bit minwise hashing (bBMWH) is an efficient hashing algorithm used in machine learning. The James–Stein (JS) estimator paradoxically produces a lower mean square error (MSE) than the traditional maximum likelihood estimator. Using 1000 documents from the Bag of Words Datasets (KOS) in the UCI Machine Learning Repository, we computed the pairwise resemblance for all documents in the dataset. We compared the performance of bBMWH with b from 1 to 4 bits with and without JS estimation, by calculating the precision, recall, F_1-score, and MSE in classifying pairs with resemblance $\geq R0$, with $R0$ from 0.30 to 0.60. Our results for $R0 = 0.30$ demonstrated that for $b = 4$ with JS estimation, the precision was high at 0.9 for a small sample size $k < 100$ and was maximized at 1.0 for higher k, while recall was decreased to 0.8. For $b = 3$, JS estimation improved precision without a significant drop in recall. JS estimation decreased the MSE of bBMWH for all b values investigated and especially for small k where MSE is higher. Our findings may be useful when precision is optimized over recall, e.g., spam detection. In cases where we want to estimate the pairwise resemblances for machine learning, bBMWH with JS estimation requires a smaller k to achieve the same MSE as bBMWH alone, thus saving computational time and storage space.

Keywords James–Stein · b-bit minwise hashing · Machine learning

J. E. D. Toh (✉) · R. X. M. Kan
NUS High School of Mathematics and Science, Singapore, Singapore
e-mail: h1710142@nushigh.edu.sg

R. X. M. Kan
e-mail: h1810065@nushigh.edu.sg

K. Kang
Engineering Systems and Design, Singapore University of Technology and Design, Singapore, Singapore
e-mail: keegan_kang@sutd.edu.sg

12.1 Introduction

Many machine learning applications are faced with very large and high-dimensional datasets, resulting in challenges in scaling up training algorithms and storing the data [1]. Hashing algorithms such as minwise hashing [2, 3] and random projections [4, 5] reduce storage requirements and improve computational efficiency, without compromising on estimation accuracy. b-bit minwise hashing (bBMWH) [6, 7] is a recent progress for efficiently (in both time and space) computing resemblances among extremely high-dimensional binary vectors. bBMWH can be seamlessly integrated [1] with linear support vector machine and logistic regression solvers.

In traditional statistical theory, no other estimation is uniformly better than the observed average when applied to observations. The paradoxical element in James–Stein estimation is that it contradicts traditional statistical theory elemental law if there are three or more sets of data, even when the three sets are completely unrelated [8, 9]. For example, the unrelated datasets of the estimates of the average price of HDB flats in Singapore, the chance of rain in London, and the average height of Americans can be combined to obtain an estimate better than computing the estimates individually in terms of mean squared error. When first proposed, the James–Stein estimator seemed counterintuitive and illogical. However, it has been proven to have lower mean squared error than the traditional maximum likelihood estimator, when there are at least three parameters of interest [8, 9].

12.2 Hypothesis

In this study, we hypothesized that adding James–Stein estimation to bBMWH improves the precision, recall, and $F1$-score and decreases the mean square error of the estimate from the hashing algorithm.

12.3 Materials and Methods

We briefly review the following: James–Stein estimation [9], minwise hashing [2, 3], and b-bit minwise hashing [7].

12.3.1 James–Stein Estimation

Given a random vector $z\tilde{}\mathcal{N}_N(\boldsymbol{\mu}, I)$, the James–Stein estimator is defined to be

$$\hat{\boldsymbol{\mu}}^{(JS)} = \left(1 - \frac{N-2}{S}\right)z \tag{12.1}$$

where $S = |z|^2$.

N is the number of true means we want to estimate across datasets.

For $N = 3$, $\boldsymbol{\mu}$ could be a vector containing the true average price of HDB flats in Singapore, true chance of rain in London, and the true average height of Americans. Given some observations, we want to estimate $\boldsymbol{\mu}$ with $\hat{\boldsymbol{\mu}}$.

The maximum likelihood estimator (MLE) for $\boldsymbol{\mu}$, $\hat{\boldsymbol{\mu}}^{(\boldsymbol{MLE})}$ maximizes a likelihood function under an assumed statistical model, so that the observed data is most probable. The likelihood of a N-variate normal distribution has a closed form and thus can be maximized by using numerical methods such as the Newton–Raphson method to obtain the roots of its derivative.

The following theorem is taken from [9] and restated here:

Theorem 1 *For $N \geq 3$, the James–Stein estimator dominates the MLE $\boldsymbol{\mu}$ in terms of expected total squared error that is*

$$\boldsymbol{E}_{\mu}\left\{\hat{\boldsymbol{\mu}}^{(\boldsymbol{JS})} - \boldsymbol{\mu}^2\right\} < \boldsymbol{E}_{\mu}\left\{\hat{\boldsymbol{\mu}}^{(\boldsymbol{MLE})} - \boldsymbol{\mu}^2\right\} \tag{12.2}$$

for every choice of $\boldsymbol{\mu}$.

12.3.2 Minwise Hashing

Computing the size of set intersections is a fundamental problem in information retrieval, databases, and machine learning. Given two sets, S_1 and S_2, where

$$S_1, S_2 \subseteq \Omega = \{0, 1, 2, \ldots, D-1\},$$

a basic task is to compute the joint size $a = |S_1 \cap S_2|$, which measures the (unnormalized) similarity between S_1 and S_2.

The Jaccard similarity or resemblance, denoted by R, provides a normalized similarity measure:

$$R = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = \frac{a}{f_1 + f_2 - a} \quad \text{where } f_1 = |S_1|,\ f_2 = |S_2|$$

Computation of all pairwise resemblances takes $\mathcal{O}\big(N^2 D\big)$ time, as one would need to iterate over all $\binom{N}{2}$ pairs of vectors and for each pair of vectors, over all D elements in the set.

In most cases, D is sufficiently big to make direct computation infeasible.

The original minwise hashing method [2, 3] has become a standard technique for estimating set similarity (e.g., resemblance). We briefly restate the algorithm here as follows:

Suppose a random permutation π is performed on Ω, i.e.,

$$\pi : \Omega \to \Omega, \quad \text{where } \Omega = \{0, 1, \ldots, D-1\}$$

A simple probability argument shows that

$$\mathbf{Pr}(\min(\pi(S_1)) = \min(\pi(S_2))) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = R \tag{12.3}$$

After k minwise independent permutations, $\pi_1, \pi_2, \ldots, \pi_k$, one can estimate R without bias, as a binomial probability,

$$\hat{R}_M = \frac{1}{k}\sum_{j=1}^{k} 1\left\{\min\left(\pi_j(S_1)\right) = \min\left(\pi_j(S_2)\right)\right\} \tag{12.4}$$

$$\mathrm{Var}\left(\hat{R}_M\right) = \frac{1}{k}R(1-R). \tag{12.5}$$

This reduces the time complexity to $\mathcal{O}\left(N^2 k\right)$ where k is the number of permutations, thus reducing the time taken while sacrificing some accuracy.

12.3.3 b-Bit Minwise Hashing

By only storing the lowest b bits of each (minwise) hashed value (e.g., $b = 1$ or 2), b-bit minwise hashing can gain substantial advantages in terms of computational efficiency and storage space [7].

The following theorem is taken from the paper on b-bit minwise hashing by Li and Konig [7], which we restate here as follows:

Theorem 2 *Define the minimum values under π to be z_i and z_2:*

$$z_1 = \min(\pi(S_1)), z_2 = \min(\pi(S_2)).$$

Define $e_{1,i}$ and $e_{2,i}$ to be the i th lowest bit of z_1 and z_2, respectively.

$$E_b = \boldsymbol{Pr}\left(\prod_{i=1}^{b} 1\ \{\ e_{1,i} = e_{2,i}\} = 1\right). \tag{12.6}$$

Assuming D is large,

$$\boldsymbol{Pr}\left(\prod_{i=1}^{b} 1\ \{\ e_{1,i} = e_{2,i}\} = 1\right) = C_{1,b} + \left(1 - C_{2,b}\right)R \tag{12.7}$$

where

$$r_1 = \frac{f_1}{D}, r_2 = \frac{f_2}{D}, \tag{12.8}$$

$$C_{1,b} = A_{1,b}\frac{r_2}{r_1 + r_2} + A_{2,b}\frac{r_1}{r_1 + r_2}, \tag{12.9}$$

$$C_{2,b} = A_{1,b}\frac{r_1}{r_1 + r_2} + A_{2,b}\frac{r_2}{r_1 + r_2} \tag{12.10}$$

$$A_{1,b} = \frac{r_1[1 - r_1]^{2^b-1}}{1 - [1 - r_1]^{2^b}}, A_{2,b} = \frac{r_2[1 - r_2]^{2^b-1}}{1 - [1 - r_2]^{2^b}} \tag{12.11}$$

$\hat{R}_b$ is an unbiased estimator of R:

$$\hat{R}_b = \frac{\hat{E}_b - C_{1,b}}{1 - C_{2,b}}, \tag{12.12}$$

$$\hat{E}_b = \frac{1}{k}\sum_{j=1}^{k}\left\{\prod_{i=1}^{b} 1\{\ e_{1,i} = e_{2,i}\} = 1\right\}, \tag{12.13}$$

where e_{1,i,π_j} and e_{2,i,π_j} denote the i th lowest bit of z_1, z_2 under the permutation π_j, respectively.

Following property of binomial distribution, we obtain

$$\begin{aligned}\mathrm{Var}\left(\hat{R}_b\right) &= \frac{\mathrm{Var}\left(\hat{E}_b\right)}{\left[1 - C_{2,b}\right]^2} = \frac{1}{k}\frac{E_b(1 - E_b)}{\left[1 - C_{2,b}\right]^2}\\ &= \frac{1}{k}\frac{[C_{1,b} + \left(1 - C_{2,b}\right)R][1 - C_{1,b} - \left(1 - C_{2,b}\right)R]}{\left[1 - C_{2,b}\right]^2}\end{aligned} \tag{12.14}$$

12.3.4 Experiment

We used Python 3.7.10 with vectorization to implement bBMWH and James–Stein estimation. We also used it to plot our graphs of the results. We computed the precision, recall, $F1$-score, and MSE at various R_0 values, using bBMWH with $b = 1,2,3,4$

Table 12.1 Document pairs in the dataset with $R \geq R_0$

R_0	Number of pairs
0.3	4230
0.4	2529
0.5	1020
0.6	353

bits with and without James–Stein estimation and also the original minwise hashing. We aimed to determine the smallest bit possible to save storage space and improve computational efficiency, while maintaining good levels of precision and recall.

Our experiment adopted a similar methodology as the Experiment 3 in the landmark b-bit minwise hashing paper by Li and Konig [7].

The word dataset used is a collection of the first 1000 documents (499,500 pairs) from the Bag of Words Dataset (KOS) in the UCI Machine Learning Repository [10].

We represented the i-th document as a binary vector X_i of size w, the total number of distinct words in the dataset. For this dataset, $w = 6906$. The j-th element of X_i will be 1 if the word occurs in the document and 0 otherwise.

We then computed the true pairwise resemblances for all documents in the dataset using the binary vectors and counted the number of pairs with $R \geq R_0$ (Table 12.1).

We conducted our experiment for $R_0 \in \{0.3, 0.4, 0.5, 0.6\}$ to represent the range covered in the abovementioned experiment. We ran bBMWH to compute the estimate of pairwise resemblances between vectors in X, represented as a square matrix. We then took the upper triangular portion of this matrix and flattened it to get a vector of $\binom{N}{2}$ elements *res*, representing the list of pairwise resemblances. We then ran James–Stein estimation on this vector which shrank the results toward 0, to obtain another vector *jsres*.

We compared these estimates to the true resemblances by selecting all elements with $R \geq R_0$ and computing the precision and recall of these estimates in identifying pairs with $R \geq R_0$.

Using the precision and recall, we calculated the $F1$-score. We also calculated the mean squared error (MSE) of these estimates with the true resemblance. This was done for $k = 500$ permutations and averaged over 100 iterations.

12.4 Results and Discussion

We present some findings for $R_0 = 0.30$ which are more significant here.

All experiments were done for $k = 500$ permutations and averaged over 100 iterations.

We plotted all graphs with error bars to represent the variance in our obtained values instead of relying on point estimates.

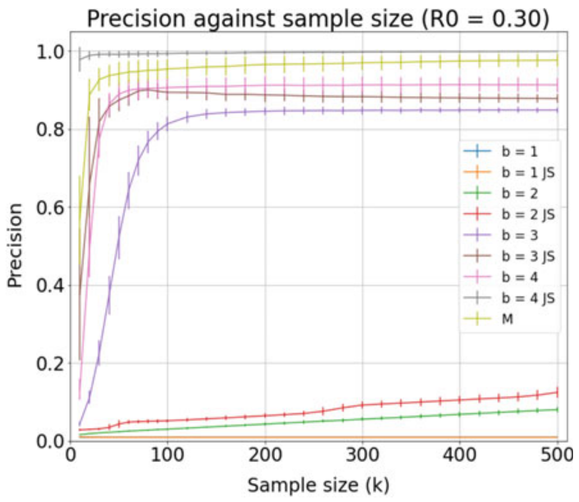

Fig. 12.1 Precision of bBMWH with and without James–Stein estimation with $R_0 = 0.30$ for $b = 1, 2, 3, 4$ and original minwise hashing

12.4.1 Precision

For $b \leq 2$, the precision for both bBMWH and bBMWH combined with James–Stein estimation was low at less than 0.2. This agrees with previous research [7] where using $b = 1$ bit per hashed value yields a sufficiently high precision only when $R_0 \geq 0.5$.

At $b = 3$, the precision for bBMWH increased to 0.8 even for a small $k = 100$. Adding James–Stein estimation to bBMWH further increased the precision.

At $b = 4$, the precision for bBMWH increased to 0.9 for $k < 100$. Adding James–Stein estimation to bBMWH further increased the precision to near 1.0 (Fig. 12.1).

12.4.2 Recall

A high recall of more than 0.9 was obtained for bBMWH alone and bBMWH with James–Stein estimation at the lowest bit of $b = 1$. This result is in accordance with previous research [7] where recall values are all very high even when very few bits are used and R_0 is low. The recall values are all very high and do not well differentiate between bBMWH with or without James–Stein estimation (except for low values of k).

However, for $b = 4$, the recall value decreased to 0.8 when James–Stein estimation was added. An increase in precision with James–Stein estimation results in a decrease in recall (Fig. 12.2).

Adding James–Stein estimation to bBMWH would be useful for applications where precision needs to be optimized over recall, for example, in spam detection where important emails wrongly classified as spam will not be seen by users.

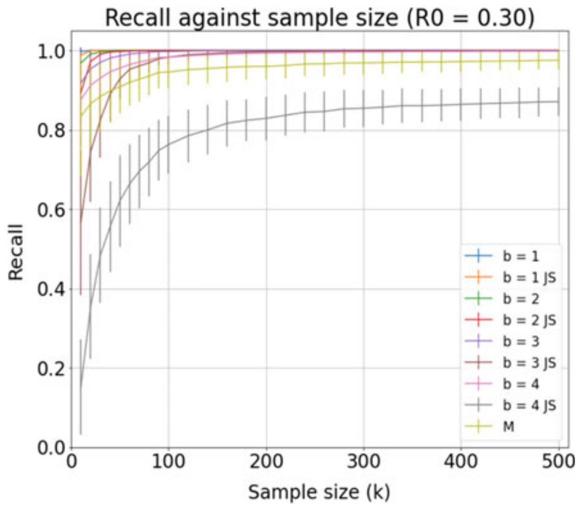

Fig. 12.2 Recall of bBMWH with and without James–Stein estimation with $R_0 = 0.30$ for b = 1, 2, 3, 4 and original minwise hashing

12.4.3 Mean Square Error

MSE is not affected by the threshold value R_0 chosen and is the same for all R_0 values.

Adding James–Stein estimation to bBMWH decreased the MSE for all b values. At small sample sizes of k, where MSE is higher, adding James–Stein estimation to bBMWH was especially useful in reducing the MSE as compared to bBMWH alone (Fig. 12.3).

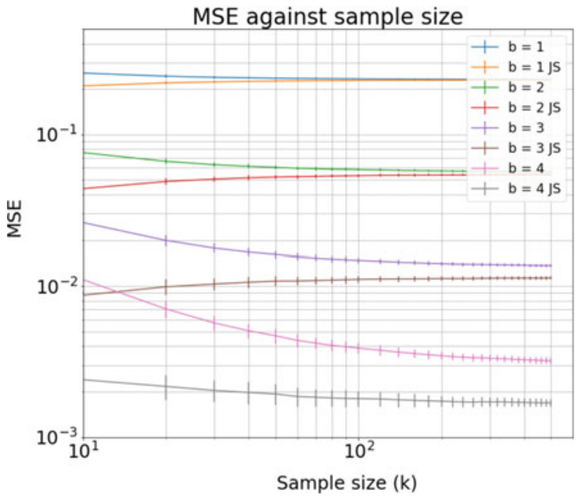

Fig. 12.3 MSE of bBMWH with and without James–Stein estimation for $b = 1, 2, 3, 4$

bBMWH (for small values of b) does require more permutations than the original minwise hashing; for example, by increasing k by a factor of 3 when using $b = 1$, the resemblance threshold is $R_0 = 0.5$. In the context of machine learning and b-bit minwise hashing, in some datasets, k has to be fairly large, e.g., $k = 500$ or even more, and this can be expensive [11]. This is because machine learning algorithms use all similarities, not just highly similar pairs.

Our results have potential applications in machine learning algorithms by achieving a low MSE without a need to increase the number of permutations k, thus saving on computational time and cost.

12.5 Conclusion and Implications

To our knowledge, we are the first to study the effect of adding James–Stein estimation to the b-bit minwise hashing algorithm.

At a low b-bit value of $b = 4$, the precision for bBMWH was high at 0.9 for a small sample size $k < 100$. Adding James–Stein estimation to bBMWH further increased the precision to 1.0 and decreased the recall to 0.8. For $b = 3$, James–Stein estimation increased the precision while not significantly decreasing the recall value. Adding James–Stein estimation to bBMWH would be useful for applications where precision needs to be optimized over recall, for example, in spam detection.

Detecting similar pairs of documents represented in a Bag of Words format is useful for search engines seeking to omit duplicate results in searches. For search engines, precision is important as a webpage wrongly marked as duplicate and omitted from search results will experience a decrease in site traffic, and users would be unable to obtain a complete search result. On the other hand, a small drop in recall that results in some duplicate pages not being omitted from search results will not significantly impact users' experience. Thus, improving the precision of classification of similar pairs while sacrificing slight recall is useful.

Adding James–Stein estimation to bBMWH decreased the MSE for all b values in the experiment. At small values of k, where MSE is typically higher, adding James–Stein estimation to bBMWH was especially useful in reducing the MSE. Our results have potential applications in machine learning algorithms by achieving a low MSE without a need to increase the number of permutations k, thus saving on computational time and cost.

References

1. Li, P., Shrivastava, A., Moore, J., & König, A. C. (2011). Hashing algorithms for large-scale learning. In *Proceedings of the 24th international conference on neural information processing systems* (pp. 2672–2680), December 2011.

2. Broder, A. Z. (1997). On the resemblance and containment of documents. In *Proceedings of the compression and complexity of sequences* (pp. 21–29), June 1997.
3. Broder, A. Z., Glassman, S. C., Manasse, M. S., & Zweig, G. (1997). Syntactic clustering of the web. *Computer Networks and ISDN Systems, 29*(8–13), 1157–1166.
4. Kang, K., & Hooker, G. (2017). Random projections with control variates. In *Proceedings of 6th international conference on pattern recognition application methods (ICPRAM)* (pp. 138–147).
5. Kang, K. (2017). Random projections with Bayesian priors. In *Proceedings of national CCF conference on natural language processing and Chinese computing (NLPCC)*, Dalian, China, pp. 170–182, Nov 2017.
6. Li, P., & König, A. C. (2011). Theory and applications of b-Bit minwise hashing. *Communications of The ACM—CACM, 54*(8), 101–109.
7. Li, P., & König, A. C. (2010). b-bit minwise hashing. In *Proceedings of 10th international conference on WWW*, Raleigh, NC, pp. 671–680, Apr 2010.
8. Efron, B., & Morris, C. (1977). Stein's paradox in statistics. *Scientific American, 236*(5), 119–127.
9. Efron, B. (2010). Empirical Bayes and the James—Stein Estimator. In *Large-scale inference: Empirical Bayes methods for estimation, testing, and prediction (Institute of Mathematical Statistics Monographs)* (pp. 1–14). Cambridge: Cambridge University Press.
10. UCI Machine Learning Repository Centre for Machine Learning and Intelligent Systems. Bag of Words Dataset (KOS). https://archive.ics.uci.edu/ml/datasets/bag+of+words. Irvine, CA: University of California, School of Information and Computer Science. Last retrieved, 2 January 2021.
11. Li, P., Shrivastava, A., & König, A. C. (2013). b-Bit minwise hashing in practice. In *Proceedings of 5th Asia-Pacific symposium on internetware* (no. 13, pp. 1–10), Oct 2013.