International Journal of Computer Trends and Technology
ISSN: 2231-2803/ https://doi.org/10.14445/22312803/IJCTT-V68I7P108

Volume 68 Issue 7, 49-55, July 2020
© 2020 Seventh Sense Research Group®

Original Article

Spherical Bitwise Weighted Hamming Distance Method

Zhen Wang[1], Baomin Shao[2]

[1,2]phd, School of computer science and technology, Shandong University of Technology
Zibo, Shandong Province, China

Received Date: 07 June 2020
Revised Date: 23 July 2020
Accepted Date: 25 July 2020

Abstract *- In recent years, hashing algorithms that can map floating point data into compact binary code have been adopted to achieve the approximate nearest neighbour (ANN) search task in the Hamming space. However, binary codes are discrete integer values, which makes the data pairs with different binary codes would share the same Hamming distance. To solve the above problem, the spherical bitwise weighted Hamming distance (SBWHD) method is proposed to assign different weight values to each binary bit. Thus, the weighted Hamming distance can be utilized to distinguish the ranking orders among the data points which have the same Hamming distance to the query sample. As the ANN search task mainly focuses on the samples at the top position of retrieval results, SBWHD learns the hashing and bitwise weight functions which obey a spherical distribution. During the training process, the similarity threshold is considered as radius, and the nearest neighbours are put inside the spherical. To further guarantee the ANN search performance, both the Hamming distance and the weighted Hamming distance are required to approximate the corresponding Euclidean distance. During the training process, SBWHD simultaneously learns the hashing functions and bitwise weight functions by an iterative optimization mechanism. When the algorithm converges, the bitwise weights can effectively improve the ANN search performance obtained based on the binary codes. The final comparative experiments in three large scale datasets prove that the ANN search performance of SBWHD is superior.*

Keywords **-** *approximate nearest neighbour search, binary code, hashing algorithm, bitwise weight method.*

I. INTRODUCTION

Generally, the high dimensional floating-point data is directly utilized to fulfil the approximate nearest neighbour (ANN) search task, and the nearest neighbour is returned according to the Euclidean distance. As the above ANN search mechanism has high time complexity, it cannot fast respond to the ANN search task in a large scale dataset. To fix this problem, hashing algorithms [1-4] are proposed to encode the raw data into compact binary code. Therefore, the ANN search task can be achieved based on the Hamming distance, which has low computational complexity. According to the training process, the existing hashing algorithms can be roughly divided into data-independent hashing [5] and data-dependent hashing [6, 7]. The data-independent hashing algorithm, such as local sensitive hashing (LSH) [5], randomly generates linear projection functions and encodes the data according to the projection results. As the training samples does not involve in the learning process of the LSH method, the generated binary codes may not be adaptive to data distribution. Thus, the ANN search performance cannot be obviously improved as the length of binary bits increases.

To achieve an excellent ANN search performance with compact binary code, the data-dependent hashing algorithms utilize a machine learning mechanism to generate hashing functions. Spectral hashing [6] and anchor graph hashing [8] establish a similarity graph and generate binary codes through the graph partition mechanism. However, both spectral hashing [6] and anchor graph hashing [8] demand the distribution of training samples should be uniform. In practice, the real datasets do not obey the above assumption. The iterative quantization (ITQ) method [7] maps the data into the vertices of a fixed hyper cubic, which leads the encoding results not adaptive to data distribution. To get rid of the restriction of data distribution, K-means hashing [9] learns encoding centres by minimizing quantization error and making the Hamming distances approximate the original Euclidean distances by minimizing the similarity loss. The above-mentioned hashing algorithms [5-9] directly utilize the data pair's Hamming distance to approximate their Euclidean distance. Recently, another kind of hashing algorithm which aims to preserve the Euclidean ordinal relationship in the Hamming space is proposed. Minimal loss hashing [10] defines listwise loss as an objective function that penalizes the similar data pairs with large Hamming distance and the dis-similar data pairs with minimal Hamming distance. Listwise loss hashing [11] defines the similarity loss function based on triplet elements, and it requires the Hamming distance of similar data pair should be minimal than that of dissimilar data pair. Order preserving hashing [12] divides data points into different categories according to their distances to the query sample. During the training process, order-preserving hashing [12] tries to make the categories in the Hamming space and Euclidean space be consistent with each other.

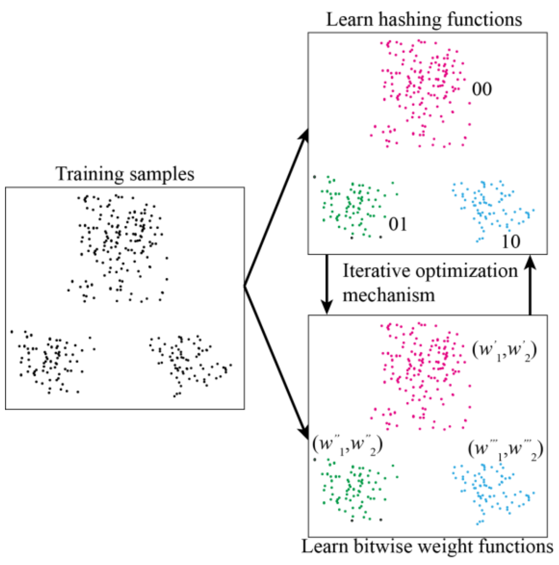


Fig. 1 SBWHD iteratively learns the hashing functions and bitwise weight functions

The binary code methods treat each binary bit equally and assign them the same weight value. However, the binary codes have a discrete integer value. As a result, many data pairs with different binary codes would share the same Hamming distance to the query sample. To further distinguish the ranking orders among the data points which have the same Hamming distance to the query sample, the bitwise weight methods, such as WhRank [13] and QRank [14], assign different weight values to each binary bit [3]. WhRank [13] utilizes the data pair's weighted Hamming distance to approximate their Euclidean distance. QRank [14] aims to learn the query sensitive bitwise weights.

The ANN search task pays more attention to the top position of the retrieval results, but the above hashing algorithms and bitwise weight methods rarely take the samples' position into consideration during the training process. To fix the above problem, a bitwise weight method termed the spherical bitwise weighted Hamming distance (SBWHD) method is proposed in this paper. SBWHD emphasizes preserving the similarity relationship among the data points at the top position. The learning process of SBWHD is shown in Fig. 1. SBWHD iteratively learns binary codes and bitwise weights. When the algorithm converges, SBWHD can get an excellent ANN search performance in both the Hamming and weighted Hamming spaces.

The main contributions of this paper can be concluded as follows.

1. The samples' positions are taken into consideration during learning hashing functions and bitwise weight functions. SBWHD can well preserve the similarity relationship among the data points at the top position, which is consistent with the nature of the ANN search task.
2. Different from the general two-step mechanism, SBWHD simultaneously learns the hashing functions and bitwise weight functions by an iterative optimization mechanism.
3. SBWHD demands the data pair's weighted Hamming distance can well approximate their original Euclidean distance. As a result, SBWHD can fulfil the ANN search task in the Hamming space.

II. SPHERICAL BITWISE WEIGHTED HAMMING DISTANCE METHOD

To fulfil the ANN search task, the nearest neighbours should have a minimal Hamming distance value to the query sample [9]. In contrast, the distance value between the dis-similar data point and the query sample should be relatively larger. To satisfy the above requirement, SBWHD learns the hashing and bitwise weight functions, which obey a spherical distribution. During the training process, the nearest neighbours are put inside of the spherical, and the dis-similar data points are set outside of the spherical [1]. Correspondingly, the similarity threshold is defined as the spherical radius.

A. The ANN Search Procedure in the Weighted Hamming Space

Given the floating-point data set $X=\{x_1, \cdots, x_n\} \in \mathbf{R}^{n \cdot d}$ (d is the dimension of data point), hashing algorithms map the data x_i into m-bit binary code $B=\{b_1, \cdots, b_m\}$ according to the projection sign as in Eq. (1). $\{f_1, \cdots, f_m\}$ represent the linear hashing functions.

$$B = \{\mathrm{sgn}(f_1(x_i)), \cdots, \mathrm{sgn}(f_m(x_i))\} \qquad (1)$$

After mapping the data into binary code, their Hamming distance can be computed by Eq. (2).

$$D_H(x_i, x_j) = \sum_{m=1}^{M} b_m(x_i) \otimes b_m(x_j) \qquad (2)$$

Based on the definition of binary code in Eq. (1), it is clearly known that the values of binary codes are discrete integers, and each binary bit has the same weight value. As a result, the data points with different binary codes would share the same Hamming distance to the query sample. A simple example is given below. If the data points x_1, x_2 and x_3 are separately encoded as 3-bit binary codes 001, 010 and 100, all the Hamming distances among them are 2. When querying, the nearest neighbours of x_1, x_2 and x_3 are simultaneously returned, and it's hard to distinguish their ranking orders in the ANN search results. To fix the above problem, SBWHD assigns different weight values $W=\{w_1, \cdots, w_M\}$ to each binary bit as in Eq. (3), and the bitwise weight function relates to the query sample.

$$w_m = g_m(x_{query}) \qquad (3)$$

According to Eq. (3), the weighted Hamming distance of a data pair can be computed by Eq. (4).

$$D_H^W(x_i, x_j) = \sum_{m=1}^{M} g_m(x_i) \cdot (b_m(x_i) \otimes b_m(x_j)) \qquad (4)$$

With the assistance of bitwise weighted Hamming distance, the ANN search performance obtained based on binary codes can be further boosted, as shown in Fig. 2. Firstly, the neighbours with minimal Hamming distance to the query sample are retrieved. Then, the data points sharing the same Hamming distance to the query sample are re-sorted according to the weighted Hamming distances.

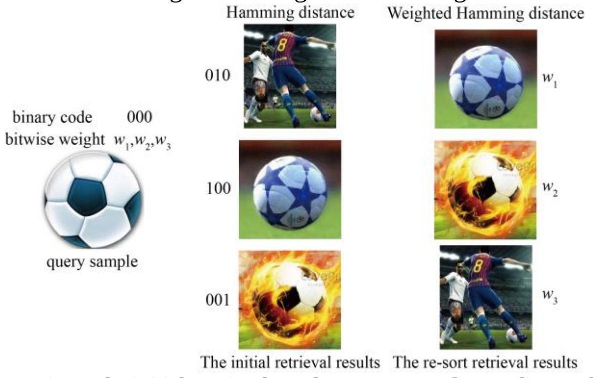

Hamming distance
Weighted Hamming distance
binary code 000
bitwise weight w_1, w_2, w_3
query sample
010
100
001
w_1
w_2
w_3
The initial retrieval results
The re-sort retrieval results

Fig. 2 The initial retrieval results are re-sorted according to the weighted Hamming distance

B. *The Similarity Preserving Restriction Based on Probability Statistic*

To achieve an excellent ANN search performance, the similarity relation should be well preserved in the weighted Hamming spaces [14, 15]. Similar to the SNE algorithm, the distance values are treated as the similarity degree, and the probability value p_{ij} and q_{ij} are separately utilized instead of the Euclidean distance and weighted Hamming distance. Therefore, the excellent hashing and bitwise weight functions can be generated by making the Hamming and Euclidean probability values resemble each other.

Here, the Kullback-Leibler is adopted to measure the mismatch between the Hamming probability values and the corresponding Euclidean probability values. The definition is shown in Eq. (5).

$$KL = \sum_{i,j}^{N} p_{ij} \log \frac{p_{ij}}{q_{ij}} \qquad (5)$$

During the training procedure, the Euclidean distance value is directly assigned to the probability p_{ij}. Then, the key problem is how to compute the value of probability q_{ij}, and it will be discussed in section 2.3.

C. *Spherical Binary Code and Bitwise Weight*

If directly utilizing the objective function in Eq. (5) to learn binary code and bitwise weights, there exists a problem as described below.

When x_i and x_j has a large Euclidean distance, the learnt bitwise weighted Hamming distance can well approximate the corresponding Euclidean distance by directly minimizing the value of the *KL* objective function. In contrast, when x_j is the nearest neighbour of x_i, and they have a small Euclidean distance value, the *KL* objective function always has a minimal value, and it fails to learn the bitwise weights. Therefore, the objective function should be re-defined according to the similarity relationship of the data pair (x_i, x_j) as in Eq. (6). For dis-similar data pairs, (d_T-p_{ij}) and (1-q_{ij}) are separately utilized instead of the original parameter.

$$KL = \sum_{i=1}^{N}\sum_{j=1}^{N} \begin{cases} p_{ij} \log \dfrac{p_{ij}}{q_{ij}} & d(x_i, x_j) > d_T \\ (d_T - p_{ij}) \log \dfrac{d_T - p_{ij}}{1 - q_{ij}} & d(x_i, x_j) < d_T \end{cases} \qquad (6)$$

When the data pairs are nearest neighbours, d_T-p_{ij} has a large value. To minimize the objective value in Eq. (6), the value of 1-q_{ij} should be enlarged. As a result, the value of q_{ij} is minimized to satisfy the requirement of preserving the similarity relationship.
During the training process, the two sub-functions in Eq. (6) are integrated as in Eq. (7).

$$KL = \sum_{i=1}^{N}\sum_{j=1}^{N} r(d_{ij} \log \frac{1}{q_{ij}} + (d_T - d_{ij}) \log \frac{1}{1-q_{ij}}) \quad (7)$$

The constant terms are omitted, and the coefficient r is employed to balance the effects of similar data pairs and dis-similar ones. In this paper, r represents the ratio value between the number of similar data pairs and that of dis-similar data pairs.
Usually, the value of q_{ij} is defined as in Eq. (8).

$$q_{ij} = \frac{1}{1 + \exp(M - 2d_h^w(x_i, x_j))} \qquad (8)$$

In Eq. (8), M is the number of binary bits, and $d_h^w(x_i, x_j)$ computes the weighted Hamming distance between x_i and x_j.

Here, the bitwise weighted Hamming distance is adopted to distinguish the similarity degree among the data pairs which have the same Hamming distance value. The weighted Hamming distance between the nearest neighbour and the query sample should be minimal than the threshold. In contrast, the distance value among dis-similar data points should be larger than the threshold value. To satisfy the above requirement, the distribution of the data points should be spherical in the Hamming space, and the similarity threshold value is defined as the radius. Therefore, the probability q_{ij} can be rewritten as in Eq. (9).

$$q_{ij} = \begin{cases} \dfrac{1}{1 + \max(0, d_h^w(x_i, x_j) - d_T^H)}, & d(x_i, x_j) < d_T \\ \dfrac{1}{1 + \max(d_T^H, d_h^w(x_i, x_j))}, & d(x_i, x_j) > d_T \end{cases} \qquad (9)$$

In Eq. (9), max(·,·) returns the relative larger value, and d_T^H is the threshold value in the weighted Hamming space. Generally, the value of d_T^H is set as 2.

D. *The Objective Function and Optimization Mechanism*

During the process of learning spherical bitwise weights and binary codes, the definition of q_{ij} in Eq. (9) is utilized to rewrite the objective function as below.

$$\begin{aligned} L = &\sum_{i,j}^{N} r \cdot d_{ij} \log(1 + \max(0, d_h^w(x_i, x_j) - d_T^H)) \\ &+ \sum_{i,j}^{N} r(d_T - d_{ij}) \log(1 + \frac{1}{\max(d_T^H, d_h^w(x_i, x_j))}) \end{aligned} \qquad (10)$$

In Eq. (10), the weighted Hamming distance is computed based on the binary code, which has discrete integer values. As a result, it's NP hard to directly optimize the objective function. To fix this problem, the discrete encoding function is relaxed to a continuous form, as shown in Eq. (11).

$$\begin{aligned} b_m(x) &= \operatorname{sgn}(u_m^T x) \\ &\approx \tanh(u_m^T x) \end{aligned} \quad (11)$$

Based on the definition of binary encoding in Eq. (11), the weighted Hamming distance is rewritten as in Eq. (12).

$$\begin{aligned} d_w^h(x_i, x_j) &= \frac{1}{2}\Big(\sum_{m=1}^{M} w_m(x_j) - \\ &\sum_{m=1}^{M} w_m(x_j)\cdot\tanh(u_m^T x_i)\cdot\tanh(u_m^T x_j)\Big) \end{aligned} \quad (12)$$

Thanks to the above relaxation mechanism, the gradient descent algorithm can be utilized to learn the hashing functions and bitwise weight functions.

III. EXPERIMENTS AND RESULTS

This section introduces a comparative experimental setting and the ANN search results.

A. Data Sets and Evaluation Standard

The comparative experiments are conducted in three widely used large scale image datasets, including NUS-WIDE [16], 22K LableMe [17] and ImageNet 100. To train bitwise weight hashing functions and evaluate the ANN search performance, each data set is divided into three sub-sets, the training dataset, the query dataset and the test dataset. The training dataset is utilized to learn the hashing and bitwise weight functions. The query dataset includes the query samples, and their nearest neighbours are stored in the test dataset. During the evaluation process, the nearest neighbours of the query samples are retrieved from the corresponding test dataset.

The total number of images in the NUS-WIDE dataset [16] is 270 thousand, and the images are randomly selected from the Flickr image dataset. For the NUS-WIDE dataset [16], 190 thousand images are randomly picked as the test dataset. Correspondingly, 50 thousand images are considered as the query samples, and the remaining 30 thousand images are utilized for training the bitwise weight functions and hashing functions in the NUS-WIDE dataset [16]. 22 thousand images without labels are stored in the 22K LabelMe dataset, and the number of images in the test dataset is 20 thousand. 2 thousand images are utilized as query samples, and 5 thousand images are randomly picked to learn the bitwise weight functions. 100 kinds of images in ImageNet are randomly picked to form the ImageNet 100 dataset. The number of the training images is 30 thousand, and 130 thousand images are considered as the test dataset. 10 thousand images are utilized as the query dataset.

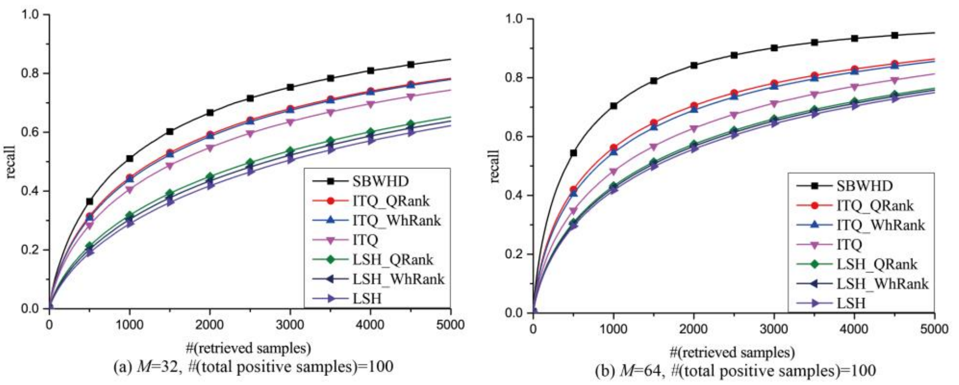
To evaluate the ANN search performance, the metric evaluation *recall* is adopted to measure the ratio of the retrieved positive samples to the total amount of the true nearest neighbours. The definition of *recall* is shown in Eq. (13).

$$recall = \frac{\#(retrieved\ \ positive\ samples)}{\#(total\ \ positive\ samples)} \quad (13)$$

#(*retrieved positive samples*) represents the number of retrieved positive samples. #(*total positive samples*) means the amount of the nearest neighbours to the query sample in the test dataset.

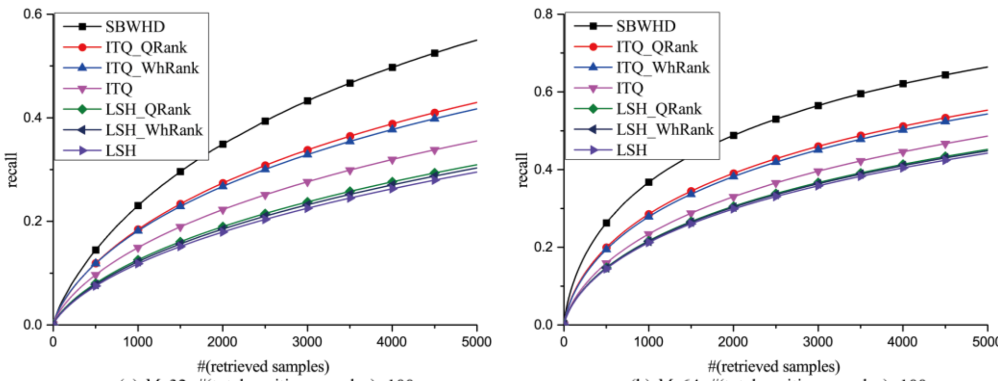
B. Experiments and Experimental Results

The performance of hashing algorithms and bitwise weight methods is evaluated by performing the ANN search of images in three widely large scale image datasets. The experimental results are shown in Figs.3-8.



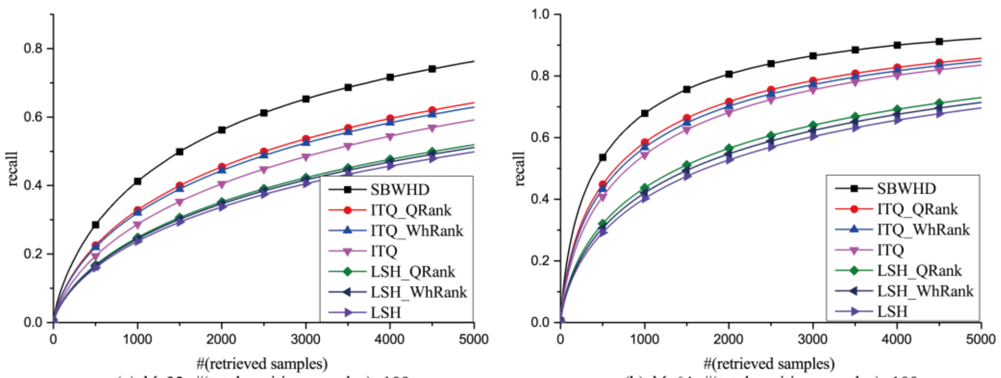
(a) M=32, #(total positive samples)=100 (b) M=64, #(total positive samples)=100

Fig. 3 The recall curves in the 22K Labelme dataset and 100 samples with minimal distance are defined as the true nearest neighbours.



(a) M=32, #(total positive samples)=100 (b) M=64, #(total positive samples)=100

Fig. 4 The recall curves in the NUS-WIDE dataset and 100 samples with minimal distance are defined as the true nearest neighbours.

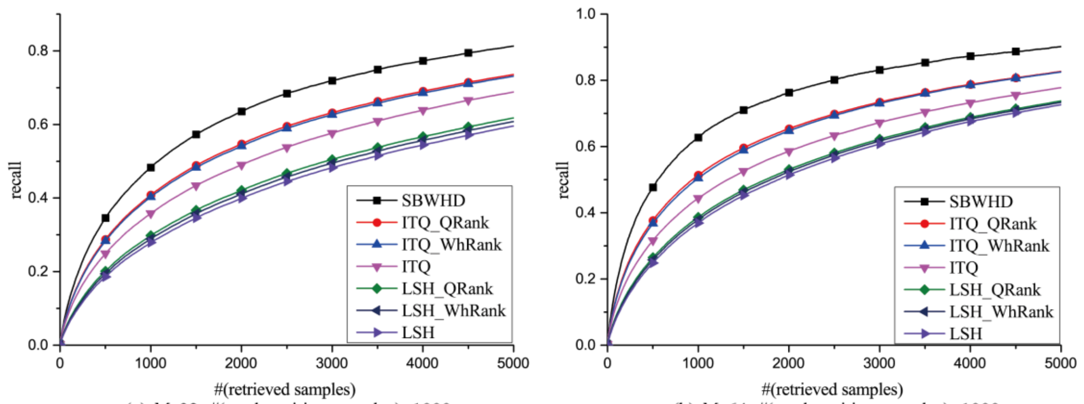


(a) M=32, #(total positive samples)=100 (b) M=64, #(total positive samples)=100

Fig. 5 The recall curves in ImageNet 100 dataset and 100 samples with minimal distance are defined as the true nearest neighbours.

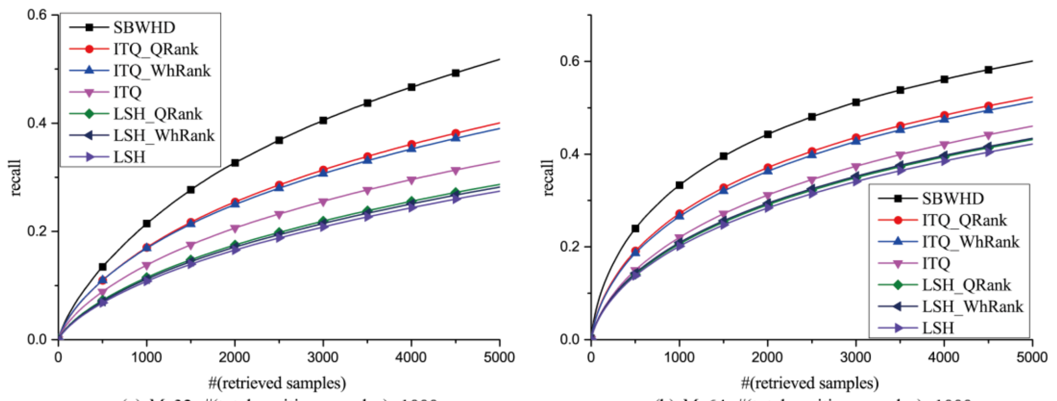
In Figs. 3, 4 and 5, the 100 samples with the nearest Euclidean distance to the query image are defined as the nearest neighbours.

To further prove the stability of the ANN search performance of SBWHD, the nearest neighbours are re-defined as the 1000 samples with minimal Euclidean distance values to the query sample. The recall cures of the comparative ANN search results are shown in Figs. 6, 7, 8.



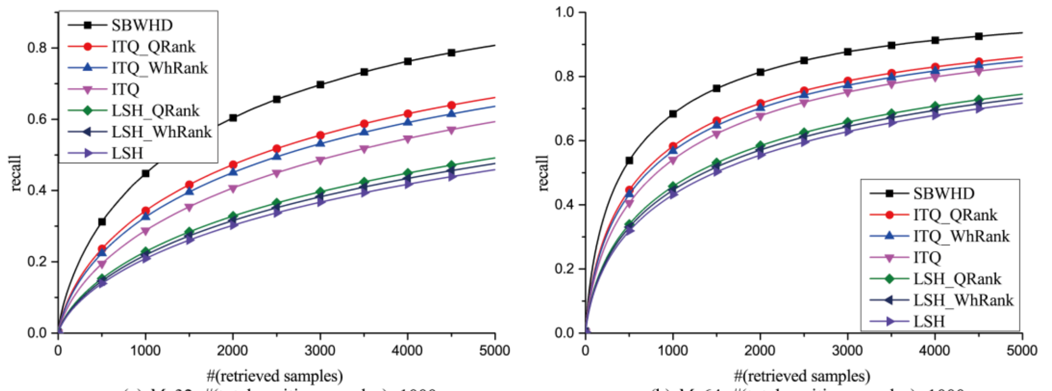
(a) M=32, #(total positive samples)=1000 (b) M=64, #(total positive samples)=1000

Fig. 6 The recall curves in the 22K Labelme dataset and 1000 samples with minimal distance are defined as the true nearest neighbours.



(a) M=32, #(total positive samples)=1000 (b) M=64, #(total positive samples)=1000

Fig. 7 The recall curves in the NUS-WIDE dataset and 1000 samples with minimal distance are defined as the true nearest neighbours.



(a) M=32, #(total positive samples)=1000 (b) M=64, #(total positive samples)=1000

Fig. 8 The recall curves in ImageNet 100 dataset and 1000 samples with minimal distance are defined as the true nearest neighbours.

The above comparative ANN search results have shown that the proposed spherical bitwise weighted Hamming distance (SBWHD) method achieves the best ANN search performance. Furthermore, two kinds of ANN search experiments with a different number of nearest neighbours tell that the ANN search performance of the SBWHD method is stable.

For the comparative experiments, the baseline methods include the binary code methods (LSH and ITQ) and the bitwise weight methods (WhRank and QRank).
The classical method, local sensitive hashing (LSH) [5], does not own the training process. LSH [5] randomly generates the hashing functions, which are independent of the training dataset. As a result, LSH has an inferior ANN search performance, and its ANN search performance cannot obviously improve as the length of binary code increases [6]. To fix the above problem, ITQ [7] and the proposed SBWHD method learn hashing functions by machine learning mechanism and the similarity loss is required to minimize. ITQ method encodes the floating-point data into binary code by assigning the data to the vertices of a hyper cubic. However, ITQ adopts a fixed also learns the query sensitive bitwise weights to make the weighted Hamming distance well approximate the original Euclidean similarity degree. In addition, SBWHD emphasizes preserving the similarity relationship Among the data points, which has a small Hamming distance to the hyper cubic, which leads the encoding results not adaptive to the data distribution [9]. Therefore, the encoding results would undermine the ANN search performance. To get rid of the restriction of the fixed encoding centres, SBWHD takes the data distribution into consideration and demands the nearest neighbours should have the same binary code.
As the binary bits are discrete integer values and have the same weight value, the data pairs with different binary codes may share the same Hamming distance value. In the Hamming space, binary code methods randomly return the samples which have the same Hamming distance value to the query sample. As a result, the ANN search performance of binary code methods, LSH [5] and ITQ [7], are relative inferior. To fix the above problem, the bitwise weights methods WhRank [13] and QRank [14] are utilized to improve the ANN search performance of the binary code methods. The proposed methods SBWHD, WhRank [13] and QRank [14] learn bitwise weights and utilize the weighted Hamming distance to further distinguish the ranking orders of the data points. WhRank [13] learns the weighted Hamming distance according to the data distribution. QRank [14] demands the bitwise weights should be sensitive to the query sample. SBWHD query sample. The final experimental results also show that SBWHD achieves the best performance. Furthermore, the comparative experiments with a different number of true nearest neighbours prove that the ANN search performance of the proposed SBWHD method is stable.

IV. CONCLUSION

As the discrete integer binary code makes many data points share the same Hamming distance to the query sample, their ranking orders are ambiguous in the initial retrieval results. To solve the above problem, the spherical bitwise weighted Hamming distance (SBWHD) method is proposed to assign different weight values to each binary bit in this paper. Usually, a two-step learning mechanism is adopted to learn the bitwise weights, which firstly employs the existing hashing algorithms to generate binary code and learns bitwise weights according to the fixed binary codes. Different from the two-step mechanism, SBWHD simultaneously learns hashing functions and bitwise weight functions by an iterative optimization mechanism. When the algorithm converges, the bitwise weights and binary codes can well cooperate in boosting the ANN search performance. In this paper, the bitwise weights are required to be sensitive to query samples and adaptive to data distribution. The ANN search task considers the samples with minimal distance value to the query data as the nearest neighbours. To satisfy the above requirement of the ANN search task, SBWHD defines a spherical and sets the similarity threshold as a spherical radius. During the training process, the nearest neighbours are required to locate inside of the spherical, and the dis-similar data points are demanded to locate outside of the spherical. With the assistance of the above measure, SBWHD can well preserve the similarity relationship among the data points at the retrieval results' top position. When searching the nearest neighbours, the initial retrieval results obtained based on Hamming distances will be further re-sorted according to the weighted Hamming distances. The final experimental results show that the ANN search performances of SBWHD are superior.

ACKNOWLEDGMENT

This work is supported by the National Nature Science Foundation of China (No. 61841602) and the Nature Science Foundation of Shandong Province (No. ZR2018PF005).

REFERENCES

[1] R. Kang, Y. Cao, M. Long, J. Wang and P. S. Yu, Maximum-Margin Hamming Hashing, in Proc. ICCV'19, (2019)8251.

[2] Z. Wang, F. Sun, L. Zhang and P. P. Liu, Minimal residual ordinal loss hashing with an adaptive optimization mechanism, EURASIP Journal on Image and Video Processing, (2020)1-11.

[3] Z. Wang, F. Sun, L. Zhang and L. Wang, Top position-sensitive ordinal relation preserving bitwise weight for image retrieval, Algorithms, 13(2020)18.

[4] Z. Wang, L. Zhang, F. Sun, L. Wang and S. Liu, Relative similarity preserving bitwise weights generated by an adaptive mechanism, Multimedia Tools and Applications, 78(2019) 24453-24472.

[5] M. Datar, N. Immorlica, P. Indyk and V. S. Mirrokni, Locality-sensitive hashing scheme based on p-stable distributions in Proc. [C]. SCG'20, (2004) 253-262.

[6] Y. Weiss, A. Torralba and R. Fergus, Spectral hashing, in Proc. NIPS, (2008) 1753-1760.

[7] Y. Gong and S. Lazebnik, Iterative Quantization: A procrustean approach to learning binary codes, in Proc. CVPR, (2011) 817-824.

[8] W. Liu, J. Wang, S. Kumar and S. F. Chang, Hashing with graphs, in Proc. ICML' 28 (2011) 1-8.

[9] K. He, F. Wen and J. Sun, K-means hashing: an affinity-preserving quantization method for learning compact binary codes, in Proc. CVPR, (2013) 2938-2945.

[10] M. Norouzi and D. J. Fleet, Minimal loss hashing for compact binary codes, in Proc. ICML'28, (2011) 353-360.

[11] J. Wang, W. Liu, A. X. Sun and Y. G. Jiang, Learning hash codes with listwise supervision, in Proc. ICCV, (2013) 3032-3039.

[12] J. Wang, J. Wang, N. Yu, and S. Li, Order preserving hashing for approximate nearest neighbour search, in Proc. ICM'21, (2013) 133-142.

[13] L. Zhang, Y. Zhang, J. Tang, K. Lu and Q. Tian, Binary code ranking with weighted hamming distance, in Proc. CVPR, (2013) 1586-1593.

[14] T. Ji, X. Liu, C. Deng, L. Huang, and B. Lang, Query-Adaptive Hash Code Ranking for Fast Nearest Neighbor Search, in Proc. ICM, (2014) 1005-1008.

[15] H. Fu, X. Kong, and Z. Wang, Binary code reranking method with weighted hamming distance, Multimedia Tools and Applications, 75 (2016) 1391-1408.

[16] X. Wang, Y. Shi, and K. M. Kitani, Deep Supervised Hashing with Triplet Labels, in Proc. ACCV, (2016) 70-84.

[17] F. Cakir and S. Sclaroff, Adaptive Hashing for Fast Similarity Search, in Proc. CVPR, (2015) 1044-1052.

[18] Dr M E Purushoththaman, Dr Bhavani Buthtkuri, Effective Multiple Verification Process Ensuring Security And Data Accuracy In Cloud Environment Storage SSRG International Journal of Computer Science and Engineering 6(7)(2019)