Learning Robust Graph Hashing for Efficient Similarity Search

Luyao Liu[1], Lei Zhu[1] and Zhihui Li[2]

[1]School of Information Technology and Electrical Engineering
[1]The University of Queensland
[2]Beijing Etrol Technologies Co., Ltd
luyao.liu@uq.edu.au, leizhu0608@gmail.com, zhihuilics@gmail.com

Abstract. Unsupervised hashing has recently drawn much attention in efficient similarity search for its desirable advantages of low storage cost, fast search speed, semantic label independence. Among the existing solutions, graph hashing makes a significant contribution as it could effectively preserve the neighbourhood data similarities into binary codes via spectral analysis. However, existing graph hashing methods separate graph construction and hashing learning into two independent processes. This two-step design may lead to sub-optimal results. Furthermore, features of data samples may unfortunately contain noises that will make the built graph less reliable. In this paper, we propose a Robust Graph Hashing (RGH) to address these problems. RGH automatically learns robust graph based on self-representation of samples to alleviate the noises. Moreover, it seamlessly integrates graph construction and hashing learning into a unified learning framework. The learning process ensures the optimal graph to be constructed for subsequent hashing learning, and simultaneously the hashing codes can well preserve similarities of data samples. An effective optimization method is devised to iteratively solve the formulated problem. Experimental results on publicly available image datasets validate the superior performance of RGH compared with several state-of-the-art hashing methods.

1 Introduction

Similarity search plays an important role in machine learning [5], data mining [10], and database [1]. However, for large-scale high-dimensional data, similarity search becomes dramatically time and memory consuming. Therefore, it is important to accelerate the similarity search while satisfying memory savings.

Hashing [2, 32, 31, 35–37, 30] is an effective indexing technique that can achieve the above objective. With binary embedding of hashing, the original time-consuming similarity computation is transformed to efficient bit operations. The similarity search process could be greatly accelerated with constant or linear time complexity [34]. And moreover, binary representation could significantly shrink the memory cost of data samples, and thus accommodate large-scale similarity search with only limited memory. Due to these desirable advantages, hashing has been widely explored for efficient large-scale similarity search [3, 19, 27, 25, 33].

Locality Sensitive Hashing (LSH) [6, 11, 14] is a typical data-independent hashing technique. It simply exploits random mapping to project the data samples into binary Hamming space. In practice, for its ignorance on underlying data characteristics, LSH generally requires more hashing tables and longer codes to achieve satisfactory performance [19, 23]. This requirement will unfortunately bring in more storage cost. To solve the limitation, learning-based hashing methods [29, 20, 12, 19, 23, 16, 18, 22, 17, 34] are proposed to generate the projected hashing codes with various advanced machine learning models. Their basic learning principal is that if two data samples are similar in the original space, they should have a small Hamming distance between their corresponding binary codes.

According to the learning dependence on semantic labels, existing learning-based hashing methods can be categorized into two major families: unsupervised hashing [29, 9, 12, 19, 16, 26] and supervised hashing [18, 22, 17, 34]. Supervised hashing learns effective binary codes based on large amounts of semantic labels. It usually achieves better performance than unsupervised hashing methods. However, high-quality labels are hardly or expensive to obtain in many applications. Unsupervised hashing is developed without this limitation. Graph hashing [29, 20, 12, 19, 23, 16] is one of the representative unsupervised hashing methods that achieve state-of-the-art performance. It generally operates with two subsequent steps: First, similarity graph is constructed by computing similarities of data samples. Second, hashing codes and functions are learned on the built graph with spectral analysis. Spectral Hashing (SPH) [29], as a pioneering graph hashing methods, extends spectral clustering to hashing. In SPH, hashing codes are computed by eigenvalue decomposition on Laplacian matrix computed from similarity graph. Hashing functions are constructed with an efficient Nystrom method. Anchor Graph Hashing (AGH) [20] utilizes low-rank matrix to approximate the similarity graph to reduce the complexity of SPH. In this method, hashing functions are learned by binarizing the eigen-functions. Inductive Manifolds Hashing (IMH) [23] considers the intrinsic manifold structure and generates non-linear hashing functions. Scalable Graph Hashing (SGH) [12] leverages feature transformation to approximate the similarity graph, and thus avoids explicit graph computing. In SGH, hashing functions are learned in a bit-wise manner with a sequential learning. Discrete Graph Hashing (DGH) [19] proposes a discrete optimization approach to learn the binary codes in binary Hamming space directly. Discrete Proximal Linearized Minimization (DPLM) [24] can also handle the discrete constraint imposed on graph hashing. In DPLM, hashing codes are solved by iterative procedures with each one admitting an analytical solution.

Although graph hashing can achieve promising results, there still exist several problems that impede its performance.

- In graph hashing methods, hashing learning fully relies on a previously constructed similarity graph. The separation of graph construction and hashing learning may lead to sub-optimal result, as the manually constructed graph may not be optimal for the subsequent hashing learning.

– Graph is generally built with specific distance measures. Extra parameters, such as the bandwidth parameter for similarity computation and the number of nearest neighbours for graph construction, will inevitably be brought into graph construction. These parameters are highly dependent on experimental experience. They may not achieve the satisfactory performance due to the high complexity of graph modelling.
– The similarity graph may be unreliable as the data samples from the real world always contain adverse noise. These noises will damage the local manifold structure and bring negative impact on the hashing performance.

In this paper, we propose a Robust Graph Hashing (RGH) to address the aforementioned problems in existing graph hashing methods. Specifically, RGH automatically learns a robust graph based on self-representation of data samples to avoid the adverse noises. Moreover, it seamlessly integrates graph construction and hashing learning into a unified learning framework where two learning processes can mutually reinforce each other. An effective optimization method is devised to iteratively solve the formulated learning problem. Experimental results on two publicly available image datasets demonstrate the superior performance of RGH compared with several state-of-the-art hashing methods.

It is worthwhile to highlight the key contributions of our proposed approach:

– We formulate a unified learning framework that integrates graph construction and hashing learning. The framework ensures that the optimal graph can be automatically constructed for hashing learning, and simultaneously the hashing codes can effectively preserve discriminative information of original data samples.
– We exploit self-representation of data samples to construct the similarity graph. It can avoid parametric distance measures. In addition, we introduce an error matrix that mitigates the negative interferences and makes the constructed graph more robust.

The rest of the paper is structured as follows. Section 2 gives the details of the proposed approach. In Section 3, we present experimental results and analysis. Section 4 concludes the paper.

2 Methodology

2.1 Overall Objective Formulation

As indicated by the principal of subspace clustering [7, 8, 15, 21], a point set in a union of subspaces generally locates in an underlying low-dimension subspace instead of distributing equably in the entire space. In presentation, it means that a data sample could be expressed as a linear combination of other data samples in low-dimension subspace.

We construct the graph with the inspiration of above principal. For a given data set $X = \{x_1, x_2, \ldots, x_n\} \in \Re^{n \times d}$ (n is the number of data samples and d is the dimension of feature representation). Each sample can be represented

with the representation of the remaining samples in data set. Formally, X can be represented as $X = XZ$, where $Z = \{z_1, z_2, \ldots, z_n\} \in \Re^{n \times n}$ is the coefficient matrix, z_i is the representation of the raw data x_i based on the subspaces. Z actually characterizes the correlations of data samples and it can be exploited for affinity matrix construction of graph. On the other hand, real world data samples inevitably contain noises that are adverse for subsequent learning. In this paper, we resort to a outlying entries matrix $E = \{e_1, e_2, \ldots, e_n\} \in \Re^{n \times d}$ to alleviate them. Then, $X = XZ$ is rewritten as $X = XZ + E$. In optimization, we try to minimize the difference between X and $XZ + E$ and automatically learn robust Z. This process can be represented as

$$\min_{Z,E} \|X - XZ - E\|_F^2 + \lambda \|E\|_1 \quad s.t.\ diag(Z) = 0, Z^T \mathbf{1} = 1 \tag{1}$$

where $\lambda > 0$ plays the trade-off between two terms, the constraint $diag(Z) = 0$ avoids the trivial case that a data point x_i is represented with a combination of itself. The constraint $Z^T \mathbf{1} = 1$ indicates that the data point locates in a union of affine subspaces. Data sample can be linearly represented with other samples. E is introduced to strengthen the robustness of the model to the corrupted noise involved in the real world data. The l_1 norm on E is to promote the sparseness of the noises.

The non-zero elements in z_i only correspond to the points from the same subspace. With Z, affinity matrix of graph can be defined as

$$W = \frac{|Z| + |Z^T|}{2} \tag{2}$$

As long as we have the affinity matrix, a conceptive graph hashing can be performed on such a self-representation based affinity matrix. The basic principal is that close data samples in original space should have small Hamming distance between their corresponding hashing codes. Formally, the hashing codes can be obtained by solving

$$\min_{Y} Tr(Y^T(D - W)Y) \quad s.t.\ Y^T Y = I, Y \in \{-1, 1\}^{n \times c} \tag{3}$$

where D is the diagonal matrix with diagonal elements and the i_{th} entry is defined as $d_i = \sum_j \frac{|z_{ij}| + |z_{ji}|}{2}$, $Y \in \{-1, 1\}^{n \times c}$ is the hashing codes of data samples. The constraint $Y^T Y = I$ ensures the different hashing bits to be independent with each other.

Conventional graph hashing methods separate graph construction and hashing learning in two subsequent process, which may lead to suboptimal results. Different from them, in this paper, we integrate them into a unified learning framework. We derive the overall objective formulation as

$$\begin{aligned} &\min_{Z,E,Y} \|X - XZ - E\|_F^2 + \beta Tr(Y^T(D - W)Y) + \lambda \|E\|_1 \\ &s.t.\ diag(Z) = 0, Z^T \mathbf{1} = 1, Y^T Y = I, Y \in \{-1, 1\}^{n \times c} \end{aligned} \tag{4}$$

where $\beta > 0$ balances robust graph construction and hashing learning.

2.2 Optimization

Directly solving hashing codes is NP-hard. In this paper, we first relax the discrete constraint to continues ones and then propose an iterative algorithm to compute the approximate results. Specifically, we iteratively optimize each variable by fixing the others.

Update matrix Z. By fixing Y, E, the optimization for Z could be represented as the following problem:

$$\begin{aligned} \min_{Z} \|X - XZ - E\|_F^2 + \beta Tr(Y^T(D - \frac{|Z| + |Z^T|}{2})Y) \\ s.t.\ diag(Z) = 0, Z^T\mathbf{1} = 1. \end{aligned} \tag{5}$$

As derived in [8], the solution of above problem could be rewritten as

$$z_k = \begin{cases} v_k - \frac{\beta p_k}{4}, & if\ v_k > \frac{\beta p_k}{4}, \\ v_k + \frac{\beta p_k}{4}, & if\ v_k < -\frac{\beta p_k}{4}, \\ 0, & otherwise. \end{cases} \tag{6}$$

where z^T presents the i-th row of Z, if $k = i, z_k = z_i = 0$ and $v = \frac{X-(XZ-xz^T)-E)^T x}{x^T x}$, p presents the i-th row of P and $P_{ij} = \|y^i - y^j\|_F^2$ where y^i presents the i-th row of matrix Y.

Update matrix E. By fixing Z, Y, the optimization for E can be represented as

$$\min_{E} \|X - XZ - E\|_F^2 + \lambda\|E\|_1 \tag{7}$$

The solution of the above problem can be obtained as [8]:

$$E_{ij} = \begin{cases} (X - XZ)_{ij} - \frac{\lambda}{2}, & if\ (X - XZ)_{ij} > \frac{\lambda}{2}, \\ (X - XZ)_{ij} + \frac{\lambda}{2}, & if\ (X - XZ)_{ij} < -\frac{\lambda}{2}, \\ 0, & otherwise. \end{cases} \tag{8}$$

Update matrix Y. By fixing Z, E, the optimization for Y can be formulated as

$$\min_{Y} Tr(Y^T(D - \frac{|Z| + |Z^T|}{2})Y) \quad s.t.\ Y^T Y = I. \tag{9}$$

Let $A = D - \frac{|Z|+|Z^T|}{2}$, Eq.(9) becomes:

$$\min_{Y} Tr(Y^T A Y),\ s.t. Y^T Y = I, \tag{10}$$

The optimal solution of Y are the eigenvectors corresponding to the smallest k eigenvalue of the Laplacian matrix A.

We iteratively conduct the above procedures and obtain the optimal relaxed results of hashing codes.

Algorithm 1 Solving the relaxed hashing codes

Input: Training data $X = \{x_1, x_2, \ldots, x_n\} \in \Re^{n \times d}$, code length c, parameters λ, β.
Output: Relaxed hashing codes Y.

Initialize Y as sparse matrix, $E = 0$, parameters λ, β.
repeat
Update Z by Eq.(6):
For $k = i, z_k = 0$; For $k \neq i$,

$$z_k = \begin{cases} v_k - \frac{\beta p_k}{4}, & if\ v_k > \frac{\beta p_k}{4}, \\ v_k + \frac{\beta p_k}{4}, & if\ v_k < -\frac{\beta p_k}{4}, \\ 0, & otherwise. \end{cases}$$

Update E by Eq.(8):

$$E_{ij} = \begin{cases} (X - XZ)_{ij} - \frac{\lambda}{2}, & if\ (X - XZ)_{ij} > \frac{\lambda}{2}, \\ (X - XZ)_{ij} + \frac{\lambda}{2}, & if\ (X - XZ)_{ij} < -\frac{\lambda}{2}, \\ 0, & otherwise. \end{cases}$$

Update Y by Eq.(10):

$$\min_{Y^T Y = I} Tr(Y^T A Y),$$

Solve it by eigenvalue decomposition of A.
until Converges or N iteration steps

2.3 Iterative Rotation

Directly binarizing the relaxed hashing codes (continues values) will lead to quantization errors. In this paper, we apply a iterative rotation on the relaxed hashing codes Y to minimize the quantization loss. This process can be formulated as:

$$\begin{aligned} &\min_{B,Q} \|B - YQ\|_F^2, \\ &s.t.\ B \in \{-1, 1\}^{n \times c}, Q^T Q = I. \end{aligned} \tag{11}$$

$Q \in \Re^{c \times c}$ is an arbitrary orthogonal matirx for rotation.

This optimization problem can be solved by applying iteratively alternative minimization as [9].

2.4 Hashing Function Learning

With the hashing codes, we construct the hashing functions. In this paper, we leverage linear projection to achieve the aim for its high efficiency. The objective is to minimize the loss between the hashing codes and the projected ones. The formulation is

$$\min_H ||Y - XH||_F^2 + \eta ||H||_F \tag{12}$$

where $\eta > 0$ has the same function with the aforementioned β and λ, $H \in \Re^{d \times c}$ denotes the projection matrix. The optimal H is calculated as

$$H = \left(X^T X + \eta I\right)^{-1} X^T Y \tag{13}$$

With H, hashing functions can be constructed as

$$F(x) = \frac{\mathtt{sgn}(xHQ) + 1}{2} \tag{14}$$

2.5 Online Search

In online search, we first leverage hashing functions to generate hashing codes for queries. Then, the similarities between queries and database samples are calculated with Hamming distance computation. Finally, similarities are ranked according to the distance ascending or descending, and their corresponding data samples are returned.

3 Experiment

3.1 Experimental Dataset

In this paper, two widely used benchmark datasets, *CIFAR-10* [13] and *NUS-WIDE* [4], are applied to evaluate the performance of our method on similarity search.

- *CIFAR-10* has 60,000 32x32 color images which are separated into 10 classes. It has 6,000 images for per class and each images is represented by 512-dimensional GIST feature. For evaluation, we randomly select 1,000 images as our query set and 1,000 images as training images. The rest images are determined as the database images to be retrieved. In this dataset, images are considered to be relevant only if they belong to the same category.
- *NUS-WIDE* is a web image database which contains 269,648 images. It provides ground-truth of 81 concepts. In experiment, we prune the original *NUS-WIDE* to construct a new dataset consisting of 195,834 images by preserving the images that belong to one of the 21 most frequent concepts. Each image is described with 500-D bag-of-words[1]. The dataset partition for evaluation in *NUS-WIDE* is the same with *CIFAR-10*. In *NUS-WIDE*, as images are labelled into multiple concepts, they are determined as relevant if they share at least one concept.

3.2 Evaluation Metrics

In experiment, mean average precision (mAP) [29, 9, 12, 19, 16, 26] is adopted as the evaluation metric for effectiveness. It is defined as the average precision (AP) of all queries. Larger mAP indicates the better retrieval performance. For a given query, AP is calculated as

$$AP = \frac{1}{NR} \sum_{r=1}^{R} pre(r) rel(r) \tag{15}$$

[1] SIFT is employed as local feature.

where R is the total number of retrieved images, NR is the number of relevant images in retrieved set, $pre(r)$ denotes the precision of top r retrieval images, which is defined as the ratio between the number of the relevant images and the number of retrieved images r, and $rel(r)$ is indicator function which equals to 1 if the r_{th} image is relevant to query, and 0 vice versa. In experiments, we set R as 100 to collect experimental results. Furthermore, *Precision-Scope* curve is also reported to illustrate the retrieval performance variations with respect to the number of retrieved images.

Table 1. mAP of all approaches on two datasets The best result in each column is marked with bold.

Methods	*CIFAR-10*				*NUS-WIDE*			
	16	32	64	128	16	32	64	128
SH	0.2434	0.2880	0.3201	0.3360	0.4425	0.4547	0.4551	0.4491
AGH	0.2755	0.2995	0.3039	0.3024	0.4321	0.4479	0.4493	0.4559
PCAH	0.2694	0.2989	0.3125	0.3048	0.4525	0.4581	0.4532	0.4420
SGH	0.2713	0.3186	0.3498	0.3838	0.4448	0.4596	0.4621	0.4655
DPLM	0.2603	0.2702	0.2992	0.3011	0.4173	0.4292	0.4331	0.4315
RGH	**0.2809**	**0.3244**	**0.3547**	**0.3871**	**0.4547**	**0.4712**	**0.4818**	**0.4965**

Table 2. Effects of robust graph construction and one-step hashing learning. RGH* is the variant method that removes E in Eq.(4). RGH† is the approach which separates graph construction and hashing learning into two steps.

Methods	*NUS-WIDE*			
	16	32	64	128
RGH*	0.4408	0.4631	0.4733	0.4857
RGH†	0.4500	0.4612	0.4767	0.4897
RGH	**0.4547**	**0.4712**	**0.4818**	**0.4965**

3.3 Compared Approaches

We compare RGH with several state-of-the-art unsupervised hashing approaches. They include Spectral Hashing (SPH) [29], Anchor Graph Hashing (AGH) [20], PCA based Hashing (PCAH) [28], Scalable Graph Hashing (SGH) [12], and Discrete Proximal Linearized Minimization (DPLM) [24].

3.4 Implementation Setting

RGH has two parameters, λ and β, in hashing objective function Eq.(4). They are used to play the balance between the formulated regularization terms. In

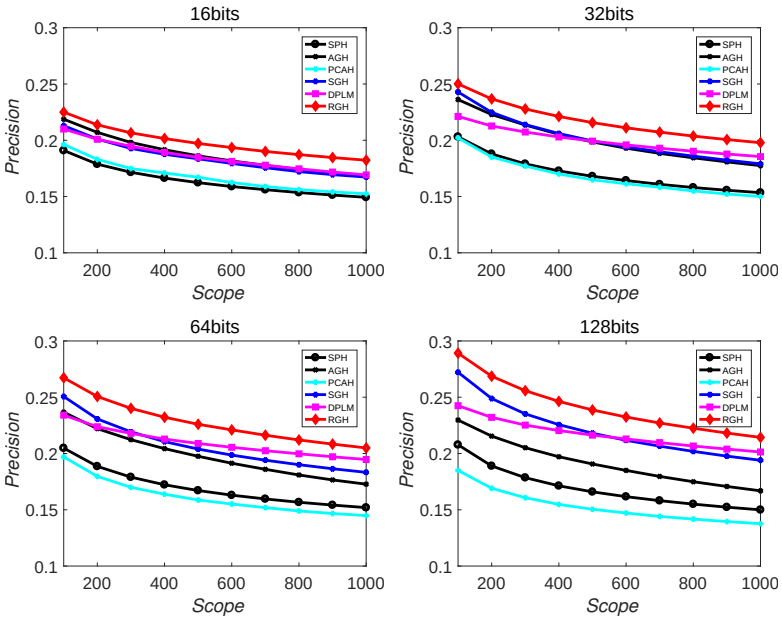

Fig. 1. *Precision-Scope* curves on *CIFAR-10* varying code length.

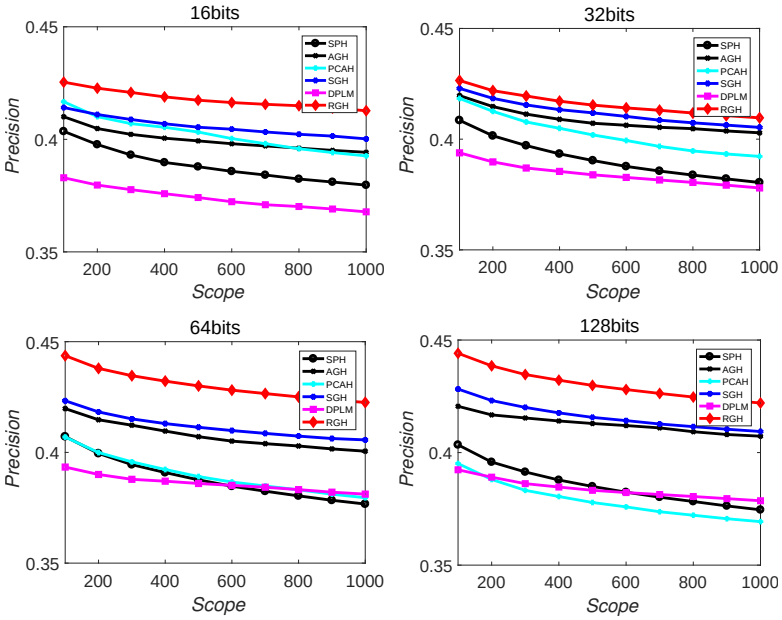

Fig. 2. *Precision-Scope* curves on *NUS-WIDE* varying code length.

experiment, we set the value of parameters as $\{\lambda = 10^{-3}, \beta = 10^{-1}\}$ and $\{\lambda = 10^{-1}, \beta = 10^{-2}\}$ to achieve the best performance on *CIFAR-10* and *NUS-WIDE* respectively. The maximum number of iterations to solve Eq.(4) and ITQ is set to 20 and 50 respectively.

In experiments, hashing code length on all datasets is varied in the range of [16, 32, 64, 128] to observe the performance. Further, The retrieval scope on two datasets is set from 100 to 1000 with step size 100. In the first step of Algorithm 1, the initial Y is a sparse matrix, values of E are set to 0. When minimizing the quantization error, we initialize Q with an arbitrary orthogonal matrix.

3.5 Experimental Results

We report mAP results of all compared methods in Table 1. The *Precision-Scope* curves on *CIFAR-10* and *NUS-WIDE* are presented in Figure 1 and 2 respectively. We can easily find that RGH outperforms the competitors on all cases. As shown in Table 1, the largest mAP gain of RGH over the second best approach is about 3.5% and 6.7%, on 16bits of *CIFAR-10* and 128bits of *NUS-WIDE* respectively. Moreover, Figure 2 shows that, on *NUS-WIDE* with 16, 64, 128 bits, the performance of RGH is much better than the second best approach with the increasing of code length.

We also compare RGH with two variants of our approach: RGH* and RGH†. RGH* learns hashing codes without any specific sample noise accommodation. In implementation, we remove the variable E in objective function Eq.(4) and conduct hashing learning. RGH† separates graph construction and hashing learning into two subsequent steps. The experimental results are presented in Table 2. From it, we observe that RGH improves the performance from 1% to 3% than other two variants. These results clearly validate the effects of robust graph construction and our one-step graph hashing learning.

4 Conclusion

In this paper, we propose a Robust Graph Hashing (RGH) to solve efficient similarity search. We formulate a unified learning framework that integrates graph construction and hashing learning. The framework guarantees that the optimal graph can be automatically constructed for hashing learning, and the hashing codes can well preserve the original data similarity. RGH automatically learns a robust graph based on self-representation to avoid the parameterized similarity graph construction. And an error matrix is introduced to mitigate the negative interferences of noises and enhance the robustness of the constructed graph. An effective optimization method is devised to iteratively solve the formulated problem. Experimental results demonstrate that the proposed method can achieve superior performance than several state-of-the-art methods.

References

1. Agrawal, R., Faloutsos, C., Swami, A.N.: Efficient similarity search in sequence databases. In: Proc. International Conference on Foundations of Data Organization and Algorithms. pp. 69–84. FODO '93, Springer-Verlag, London, UK (1993)
2. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM 51(1), 117 – 122 (2008)
3. Andoni, A., Razenshteyn, I.: Optimal data-dependent hashing for approximate near neighbors. In: Proc. Annual ACM Symposium on Theory of Computing. pp. 793–801. STOC '15, ACM (2015)
4. Chua, T.S., Tang, J., Hong, R., Li, H., Luo, Z., Zheng, Y.: Nus-wide: A real-world web image database from national university of singapore. In: Proc. ACM International Conference on Image and Video Retrieval. pp. 48:1–48:9. CIVR '09, ACM (2009)
5. Cost, S., Salzberg, S.: A weighted nearest neighbor algorithm for learning with symbolic features. Mach. Learn. 10(1), 57–78 (1993)
6. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proc. Annual Symposium on Computational Geometry. pp. 253–262. SCG '04, ACM (2004)
7. Elhamifar, E., Vidal, R.: Sparse subspace clustering. In: Proc. 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2790–2797 (2009)
8. Gao, H., Nie, F., Li, X., Huang, H.: Multi-view subspace clustering. In: Proc. 2015 IEEE International Conference on Computer Vision (ICCV). pp. 4238–4246 (2015)
9. Gong, Y., Lazebnik, S., Gordo, A., Perronnin, F.: Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. IEEE Trans. Pattern Anal. Mach. Intell. 35(12), 2916–2929 (2013)
10. Hastie, T., Tibshirani, R.: Discriminant adaptive nearest neighbor classification. IEEE Trans. Pattern Anal. Mach. Intell. 18(6), 607–616 (1996)
11. Indyk, P., Motwani, R.: Approximate nearest neighbors: Towards removing the curse of dimensionality. In: Proc. Annual ACM Symposium on Theory of Computing. pp. 604–613. STOC '98, ACM (1998)
12. Jiang, Q.Y., Li, W.J.: Scalable graph hashing with feature transformation. In: Proc. International Conference on Artificial Intelligence. pp. 2248–2254. IJCAI'15, AAAI Press (2015)
13. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009)
14. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing. IEEE Trans. Pattern Anal. Mach. Intell. 34(6), 1092–1104 (2012)
15. Li, C.G., Vidal, R.: Structured sparse subspace clustering: A unified optimization framework. In: Proc. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 277–286 (2015)
16. Li, X., Hu, D., Nie, F.: Large graph hashing with spectral rotation (2017)
17. Liong, V.E., Lu, J., Wang, G., Moulin, P., Zhou, J.: Deep hashing for compact binary codes learning. In: Proc. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2475–2483 (2015)
18. Liu, W., Wang, J., Ji, R., Jiang, Y.G., Chang, S.F.: Supervised hashing with kernels. In: Proc. 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2074–2081 (2012)
19. Liu, W., Mu, C., Kumar, S., Chang, S.F.: Discrete graph hashing. In: Proc. International Conference on Neural Information Processing Systems. pp. 3419–3427. NIPS'14, MIT Press (2014)

20. Liu, W., Wang, J., Kumar, S., fu Chang, S.: Hashing with graphs. In: Getoor, L., Scheffer, T. (eds.) Proc. International Conference on Machine Learning (ICML-11). pp. 1–8. ACM (2011)
21. Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing 17(4), 395–416 (2007)
22. Shen, F., Shen, C., Liu, W., Shen, H.T.: Supervised discrete hashing. In: Proc. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 37–45 (2015)
23. Shen, F., Shen, C., Shi, Q., van den Hengel, A., Tang, Z.: Inductive hashing on manifolds. In: Proc. 2013 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1562–1569 (2013)
24. Shen, F., Zhou, X., Yang, Y., Song, J., Shen, H.T., Tao, D.: A fast optimization method for general binary code learning. IEEE Trans. Image Process. 25(12), 5610–5621 (2016)
25. Wang, J., Kumar, S., Chang, S.F.: Semi-supervised hashing for scalable image retrieval. In: Proc. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 3424–3431 (2010)
26. Wang, J., Kumar, S., Chang, S.F.: Semi-supervised hashing for large-scale search. IEEE Trans. Pattern Anal. Mach. Intell. 34(12), 2393–2406 (2012)
27. Wang, J., Xu, X.S., Guo, S., Cui, L., Wang, X.L.: Linear unsupervised hashing for {ANN} search in euclidean space. Neurocomputing 171, 283 – 292 (2016)
28. Wang, J., Kumar, S., Chang, S.F.: Semi-supervised hashing for large-scale search. IEEE Trans. Pattern Anal. Mach. Intell. 34(12), 2393–2406 (2012)
29. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (eds.) Proc. Advances in Neural Information Processing Systems 21, pp. 1753–1760. Curran Associates, Inc. (2009)
30. Xie, L., Shen, J., Zhu, L.: Online cross-modal hashing for web image retrieval. In: Proc. AAAI Conference on Artificial Intelligence (AAAI). pp. 294–300 (2016)
31. Xie, L., Zhu, L., Chen, G.: Unsupervised multi-graph cross-modal hashing for large-scale multimedia retrieval. Multimedia Tools Appl. 75(15), 9185–9204 (2016)
32. Xie, L., Zhu, L., Pan, P., Lu, Y.: Cross-modal self-taught hashing for large-scale image retrieval. Signal Process. 124, 81–92 (2016)
33. Xu, H., Wang, J., Li, Z., Zeng, G., Li, S., Yu, N.: Complementary hashing for approximate nearest neighbor search. In: Proc. 2011 International Conference on Computer Vision. pp. 1631–1638 (2011)
34. Zhang, P., Zhang, W., Li, W.J., Guo, M.: Supervised hashing with latent factor models. In: Proc. International ACM SIGIR Conference on Research 38; Development in Information Retrieval. pp. 173–182. SIGIR '14, ACM (2014)
35. Zhu, L., Shen, J., Xie, L., Cheng, Z.: Unsupervised topic hypergraph hashing for efficient mobile image retrieval. IEEE Trans. Cybern. PP(99), 1–14 (2016)
36. Zhu, L., She, J., Liu, X., Xie, L., Nie, L.: Learning compact visual representation with canonical views for robust mobile landmark search. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence. pp. 3959–3965 (2016)
37. Zhu, L., Shen, J., Xie, L., Cheng, Z.: Unsupervised visual hashing with semantic assistant for content-based image retrieval. IEEE Trans. on Knowl. and Data Eng. 29(2), 472–486 (2017)