LETTER

Efficient Anchor Graph Hashing with Data-Dependent Anchor Selection

Hiroaki TAKEBE[†a)], *Member*, Yusuke UEHARA[†], *Nonmember, and* Seiichi UCHIDA[††], *Senior Member*

SUMMARY Anchor graph hashing (AGH) is a promising hashing method for nearest neighbor (NN) search. AGH realizes efficient search by generating and utilizing a small number of points that are called anchors. In this paper, we propose a method for improving AGH, which considers data distribution in a similarity space and selects suitable anchors by performing principal component analysis (PCA) in the similarity space.

key words: *nearest neighbor search, anchor graph hashing, similarity space, principal component analysis*

1. Introduction

When a set of data in a metric space is given, nearest neighbor (NN) search is the problem of finding a pattern that is the nearest from a query. This is one of the most fundamental problems in computer science, and NN search plays an important role in handling massive data, such as by pattern recognition (e.g., [1]).

Since recent NN search tasks deal with a high-dimensional space having massive data, approximate NN search methods (ANN) are often used. Among ANN search methods, hashing-based NN searches are becoming popular. These methods enable high-speed search by converting each data into binary codes called hash vectors.

One of the best-known hash-based methods is Locality Sensitive Hashing (LSH) [2], [3]. Although LSH has succeeded in many cases, it has the problem of needing huge memory according to the increase of training data, because LSH generates hash tables independent of data and thus has to maintain multiple hash tables in order to achieve good recall.

Recently, data-dependent hash generation approaches have attracted a great deal of attention [4]–[6]. This approach enables efficient ANN search by mapping high-dimensional data on the low-dimensional space, based on the idea that data are distributed on a low-dimensional subspace. Spectral Hashing (SH) [4] and Anchor Graph Hashing (AGH) [5] are well-known methods of the data-dependent hash generation approach. These methods are based upon a Laplacian eigenmap [7] that reduces the dimension of data by estimating a gram matrix (similarity matrix) of data.

SH computes hash vectors for unknown data by solving an eigenfunction problem under the strong assumption that the data will be uniformly distributed over the hyperrectangle spanned by eigenvectors, which are decided by principal component analysis (PCA). However, the assumption cannot be relied on to be suitable for real data.

On the other hand, AGH can compute hash vectors for unknown data by calculating the similarities between the data and the small number of points that are called *anchors*. AGH is promising because it does not assume any specific data distribution. AGH, however, relies on the simple K-means for anchor generation and thus its performance is degraded when data distribution does not fit to K-means as described later.

In this paper, we propose a method of generating suitable anchors so that the performance of AGH improves. We firstly show that ideal anchors can be obtained by mapping data to a similarity space, in which each data is represented by a vector whose elements are similarities with the other data, and performing PCA for the data distribution in the similarity space. We secondly show the algorithm of selecting data as anchors. Specifically, data oriented near the eigenvector with a large eigenvalue are selected as anchors. As a result, AGH with these selected anchors can be improved more than the conventional AGH.

2. Anchor Graph Hashing (AGH) [5]

2.1 Outline of AGH

In AGH, m anchors $\{\mathbf{a}_j \mid j = 1, \ldots, m\}$ are generated from n training data $\{\mathbf{x}_i \mid i = 1, \ldots, m\}$, where $m \ll n$. As detailed later, AGH defines similarity between a pair of data $\mathbf{x}_i$ and $\mathbf{x}_{i'}$ by using similarities between $\mathbf{x}_i$ and its neighboring anchors $\mathbf{a}_j$ $(j \in \langle i \rangle)$ and those between $\mathbf{x}_{i'}$ and $\mathbf{a}_{j'}$ $(j' \in \langle i' \rangle)$, where $\langle i \rangle$ represents a set of indices of the s nearest anchors for $\mathbf{x}_i$. Figure 1 illustrates a graph where each data is shown as a white circle and an anchor is shown as a black circle.

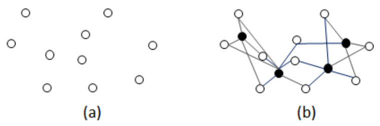

Fig. 1 (a) Data distribution. (b) Data distribution and anchors. Black circles are anchors. The graph shows data connected with their s nearest anchors.

Manuscript received March 10, 2015.
Manuscript revised July 30, 2015.
Manuscript publicized August 17, 2015.

[†]The authors are with FUJITSU LABORATORIES LTD., Kawasaki-shi, 211–8588 Japan.
[††]The author is with Kyushu University, Fukuoka-shi, 819–0395 Japan.
a) E-mail: takebe.hiroaki@jp.fujitsu.com

DOI: 10.1587/transinf.2015EDL8060

No data pairs are directly connected by an edge, but they are indirectly connected via one (or more) anchors. For deriving a $n \times n$ matrix $\mathbf{A}$ whose elements are the similarity by the above principle, we first calculate a $n \times m$ matrix $\mathbf{Z}$ called a truncated similarity matrix whose elements are calculated as follows:

$$\mathbf{Z}_{ij} = \begin{cases} \dfrac{\exp(-\|\mathbf{x}_i - \mathbf{a}_j\|^2/T)}{\sum\limits_{j' \in \langle i \rangle} \exp(-\|\mathbf{x}_i - \mathbf{a}_{j'}\|^2/T)}, & \text{if } j \in \langle i \rangle, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Here, T is a bandwidth parameter. Note that the matrix $\mathbf{Z}$ is sparse, because almost all similarities other than the s nearest are truncated. By using $\mathbf{Z}$, the approximate similarity matrix $\mathbf{A}$ can be derived as $\mathbf{A} = \mathbf{Z}\mathbf{\Lambda}^{-1}\mathbf{Z}^T$ where $\mathbf{\Lambda} = \mathrm{diag}(\mathbf{1}^T\mathbf{Z})$ [8]. The multiplication by the term $\mathbf{\Lambda}^{-1}$ is for normalization, and in fact both row and column sums of $\mathbf{A}$ are units.

In AGH, data are mapped to a low dimensional space. The mapping can be roughly described as follows: firstly data is nonlinearly transformed to a vector by computing its similarities to anchors as shown in Fig. 2 (a); then the similarity vectors are projected onto vectors calculated from the largest eigenvectors of the graph Laplacian matrix defined by $\mathbf{I} - \mathbf{A}$, where $\mathbf{I}$ is an identity matrix. For more details, see [5].

2.2 Drawback of AGH

When classification by NN search is considered, performance of AGH depends greatly upon what anchors are generated. The dependency of AGH performance on the anchors is illustrated well by using a similarity space. As described in Sect. 2.1, data mapping to a low dimensional space in AGH can be considered as composition of a nonlinear mapping to the similarity space and a linear mapping from it to the low dimensional space. Thus, we can roughly estimate classification ability of AGH by observing the distribution of mapped data in the similarity space.

Here, we estimate AGH performance based upon a space spanned by distances from anchors, which can be substituted for the similarity space. Some examples of distribution in the original space and the distribution of mapped data in the space spanned by distances are shown in Fig. 2. (a) and (b) show the case of original AGH, which uses K-means. In this case, the distribution can be represented well in (a) but not in (b). Specifically, data from different class are separated in the space spanned by the distances in (a) but not in (b). This means that the original AGH performance is good in (a) but not in (b). On the other hand, if anchors are generated as shown by (c), the distribution in the space spanned by distances would have a larger variance than (b). Then, it will become easier for NN search to classify unknown data.

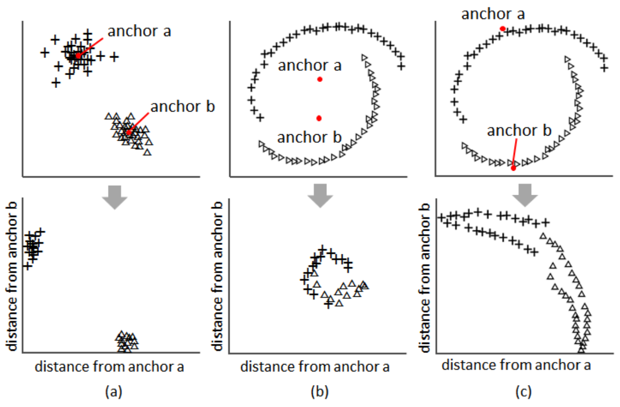

Fig. 2 Toy examples of data distribution and its distribution of mapped data in a space spanned by distances from anchors. (a) Anchors are generated by K-means. In this case, the distribution of mapped data is easy to classify. (b) Anchors are generated by K-means. In this case, the distribution of mapped data is difficult to classify. (c) An example of the distribution of mapped data becoming easier to classify because of configuration of anchors.

3. The Proposed Method

For improving the performance of AGH, we propose a new anchor generation method. In fact, if we can have better anchors like those in Fig. 2 (c), we can avoid the case in Fig. 2 (b). Then we use the criterion of preserving the shape of data distribution as much as possible on transforming data from the original space into the similarity space.

We notice that the truncated similarity matrix $\mathbf{Z}$ can be considered as expressing distribution of mapped data in the similarity space. Specifically, the truncated similarity matrix $\mathbf{Z}_a$ in which all data are made to be anchors can be considered as the original distribution in the similarity space, as shown in Fig. 3 (a). Therefore, our strategy can be described as being that anchors should be generated so that the reduction from $\mathbf{Z}_a$ to $\mathbf{Z}$ can preserve the shape of the original distribution as much as possible. Thus, we can generate anchors by realizing this reduction by PCA, as shown in Figs. 3 (b) and (c).

PCA is performed for a data distribution that corresponds to the truncated similarity matrix $\mathbf{Z}_a$. Then, the eigenvalues and eigenvectors of the covariance matrix $\mathbf{Z}_a\mathbf{Z}_a^T$ can be obtained. Here, the eigenvectors corresponding to the k large eigenvalues are described as $\mathbf{v}_t$ and σ_t $(t = 1, \ldots, k)$, respectively, and ith element of $\mathbf{v}_t$ is described as v_i^t $(i = 1, \ldots, n)$. The eigenvectors indicate ideal anchors, which

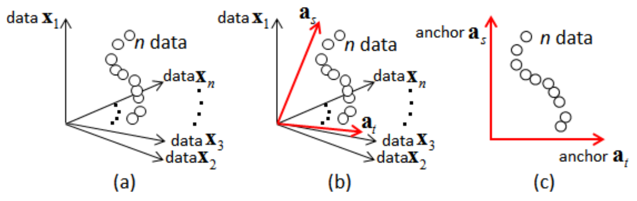

Fig. 3 (a) Data distribution in the similarity space obtained by making all the n data be anchors. (b) The principal components of the data distribution. (c) The data distribution transformed to the reduced similarity space spanned by the principal components.

means that points in the feature space whose similarities for each data $\mathbf{x}_i$ equal v_i^t are ideal anchors. Here, we can say that directions corresponding to ideal anchors are obtained by kernel PCA from the fact that the eigenvectors of $\mathbf{Z}_a\mathbf{Z}_a^T$ equal those of $\mathbf{Z}_a$.

The anchors are selected based on the obtained eigenvectors. It is difficult to compute a point in the feature space that realizes an ideal anchor, so the approximate solutions are applied by selecting data oriented near the eigenvectors. Multiple anchors are extracted from each eigenvector. Specifically, several elements of largest weights are extracted according to their absolute values for each eigenvector $\mathbf{v}_t$, and the training data corresponding to those extracted elements are selected as anchors. Here, the index i_u of anchors that should be selected can be described as (2).

$$i_u = \mathop{\arg\max}_{i \in \{1,2,\ldots,n\} \setminus \{i_1,\ldots,i_{u-1}\}} (|v_i^t|) \tag{2}$$

4. Experiment

4.1 Results of Anchor Selection

We show images of anchors selected by the proposed method. Here, we use a private font image dataset, which is provided by a commercial site[†]. Their shape variation is about 7000 different font images, like those shown in Fig. 4 (a). The proposed method selects data as an anchor corresponding to an element with a large absolute value of eigenvector. Then we show the anchors associated with elements of eigenvectors in Fig. 5.

First of all, 500 images of "A" were selected randomly. Then, the feature vectors with 288 dimensions were extracted from the character images based on the weighted direction index histogram method [9]. The truncated similarity matrix and the covariance matrix were calculated from these vectors, and the eigenvectors were obtained. In Fig. 5, we show 1st, 8th and 23rd eigenvectors in the graphs and add the images of the selected anchors on the corresponding elements of the eigenvectors. From these images, it is clear that the anchors obtained from the same eigenvector are similar. Moreover, it is confirmed that the anchors obtained from a higher eigenvector are often usually used fonts, and those from a lower eigenvector are deformed or decorated fonts.

4.2 Results of Recognition Experiment

We conducted recognition experiments for evaluating the performance of the proposed method on two datasets: MNIST, which is a well-known dataset of handwritten digits from "0" to "9" as shown by Fig. 4 (b), and the large font image dataset, which was used at Sect. 4.1. The font image dataset includes various patterns for a character, so it is considered as an example of a dataset that has a more complex distribution than MNIST. The reason for using this dataset for our evaluation is that we can expect it to reveal the weakness of the original AGH and the strength of the proposed AGH. This expectation comes from its non-Gaussian distribution, where k-means results in the case like in Fig. 2 (b).

We compared the proposed method (AGH-P) with the conventional methods in a character recognition experiment. The conventional methods are AGH that uses K-means (AGH-K), AGH which uses anchors selected randomly (AGH-R), and NN search by full search (ALL-NN).

We randomly extracted 500 images for training and 1000 images for the test of each class. Then, we experimented on the character recognition for the test set. In ALL-NN, full NN search was performed based on Euclidean distance. In each kind of AGH, we made the hash dimensions be 16. In addition, the class of data with the smallest Ham-

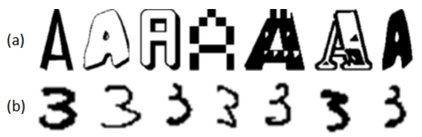

Fig. 4 (a) Example images of the font image dataset. (b) Example images of MNIST.

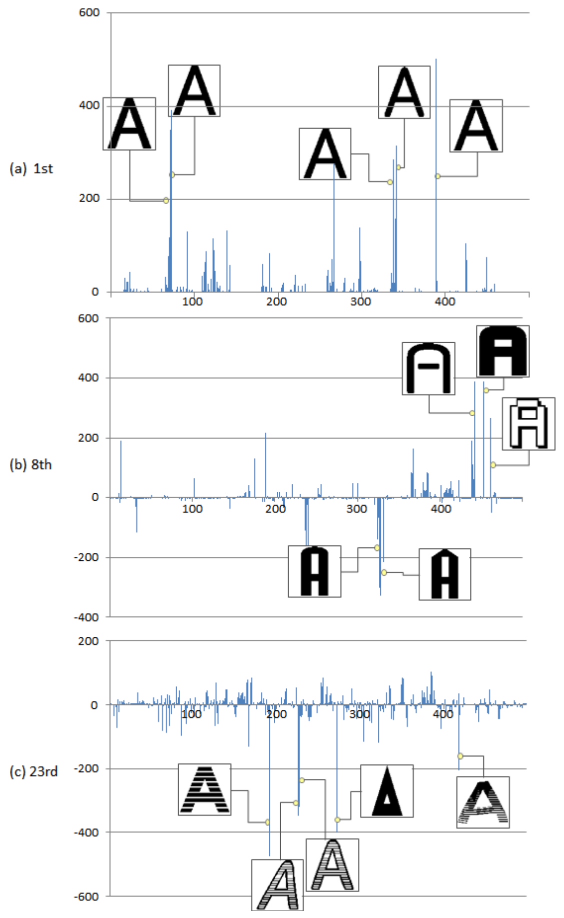

Fig. 5 Examples of element representation of eigenvectors and the images of anchors obtained by the proposed method. The horizontal axis means elements of an eigenvector and the vertical axis means weight of each element.

[†]http://www.ultimatefontdownload.com/

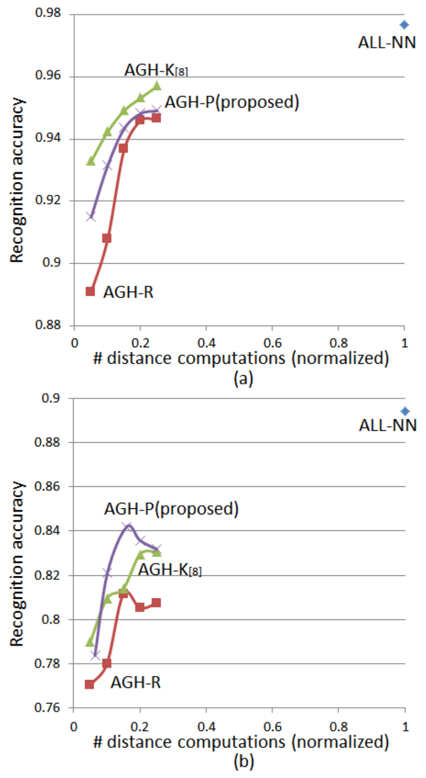

Fig. 6 Effect of computation reduction on MNIST (a) and font image dataset (b).

ming distance was output as a recognition result. When there were multiple classes of data with the smallest distance, we made a decision by a majority. Moreover, when we evaluated each kind of AGH, the number of anchors was changed. We set the number of eigenvectors and anchors (#eigenvectors, #anchors) used in the proposed method as follows: (13,26), (25,50), (38,76), (25,100), (32,128) on MNIST, and (16,32), (17,51), (20,80), (25,100), (25,125) on the font image dataset.

The results of recognition experiments are shown in Fig. 6. These horizontal axes represent the normalized number of Euclidean distance calculations in the original feature space. For AGH, these values mean the ratios of the number of anchors to all training data. From these, we can find that AGH methods could make computation 5 to 10 times more efficient with only slightly lower accuracy than ALL-NN. The result for MNIST is shown at Fig. 6 (a). The proposed method (AGH-P) performed worse than AGH-K, though it performed better than AGH-R. From this, we can find that the original AGH using K-means is very useful for a dataset that has a simple distribution such as handwritten digits. The result for the font image set is shown in Fig. 6 (b). AGH-P achieved the second highest accuracy after ALL-NN. Specifically, the accuracy of AGH-P was better than that of AGH-K. From this we can confirm the original AGH using K-means loses the effectiveness. On the other hand, the proposed method demonstrates its effectiveness for a dataset that has a complex distribution.

5. Conclusion

We propose a method for generating anchors so that data distribution in a similarity space could be preserved as much as possible. In this method, ideal anchors can be obtained as the directions of the principal components of data distribution in the similarity space, and data oriented near the principal components are selected as anchors. Experimental results show that AGH with the proposed method is effective for data that have a complex distribution.

References

[1] M. Iwamura, T. Tsuji, and K. Kise, “Memory-based recognition of camera-captured characters,” Proc. 8th IAPR International Workshop on Document Analysis Systems (DAS2010), pp.89–96, 2010.

[2] P. Indyk and R. Motwani, “Approximate nearest neighbors: Towards removing the curse of dimensionality,” Proc. 30th ACM Symposium on Theory of Computing (STOC ’98), pp.604–613, 1999.

[3] M. Datar, N. Immorlica, P. Indyk, and V.S. Mirrokni, “Locality-sensitive hashing scheme based on p-stable,” Proc. 20th Annual Symposium on Computational Geometry (SCG2004), pp.253–262, 2004.

[4] Y. Weiss, A. Torralba, and R.N. Fergus, “Spectral hashing,” Advances in Neural Information Processing Systems (NIPS), vol.21, pp.1753–1760, 2008.

[5] W. Liu, J. Wang, S. Kumar, and S.F. Chang, “Hashing with graphs,” Proc. 28th International Conference on Machine Learning (ICML), pp.1–8, 2011.

[6] D. Zhang, J. Wang, D. Cai, and J. Lu, “Self-taught hashing for fast similarity search,” Proc. 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.18–25, 2010.

[7] M. Belkin and P. Niyogi, “Laplacian eigenmaps for dimensionality reduction and data representation,” Neural Comput., vol.15, no.6, pp.1373–1396, 2003.

[8] W. Liu, J. He, and S.F. Chang, “Large graph construction for scalable semi-supervised learning,” Proc. 26th International Conference on Machine Learning (ICML), pp.679–686, 2010.

[9] F. Kimura, T. Wakabayashi, S. Tsuruoka, and Y. Miyake, “Improvement of handwritten Japanese character recognition using weighted direction code histogram,” Pattern Recognit., vol.30, no.7, pp.1329–1337, 1997.