Neurocomputing 171 (2016) 283–292

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Linear unsupervised hashing for ANN search in Euclidean space

Jian Wang, Xin-Shun Xu*, Shanqing Guo, Lizhen Cui, Xiao-Lin Wang

School of Computer Science and Technology, Shandong University, 1500 Shunhua Road, Jinan 250101, China

ARTICLE INFO

Article history:
Received 21 January 2015
Received in revised form
9 May 2015
Accepted 22 June 2015
Communicated by X. Gao
Available online 30 June 2015

Keywords:
Hashing
ANN search
Unsupervised Hashing
Locality-Sensitive Hashing
Sequential learning
Euclidean space

ABSTRACT

Approximate nearest neighbors (ANN) search for large scale data has attracted considerable attention due to the fact that large amounts of data are easily available. Recently, hashing has been widely adopted for similarity search because of its good potential for low storage cost and fast query speed. Among of them, when semantic similarity information is available, supervised hashing methods show better performance than unsupervised ones. However, supervised hashing methods need explicit similarity information which is not available in some scenarios. In addition, they have the problems of difficult optimization and time consuming for training, which make them unpracticable to large scale data. In this paper, we propose an unsupervised hashing method – Unsupervised Euclidean Hashing (USEH), which learns and generates hashing codes to preserve the Euclidean distance relationship between data. Specifically, USEH first utilizes Locality-Sensitive Hashing (LSH) to generate pseudo labels; then, it adopts a sequential learning strategy to learn the hash functions, one bit at a time, which can generate very discriminative codes. Moreover, USEH avoids explicitly computing the similarity matrix by decomposing it into the product of a label matrix and its transposition, which makes the training complexity of USEH linear to the size of training samples when the number of training samples is much greater than the dimension of feature. Thus, it can efficiently work on large scale data. We test USEH on two large scale datasets – SIFT1M and GIST1M. Experimental results show that USEH is comparable to state-of-the-art unsupervised hashing methods.

1. Introduction

In recent years, there has been a massive explosion of data on the web, e.g., images and videos. For such data, we usually need to find the nearest neighbors of a query sample from a given large scale database. However, such databases contain even billions of samples; this requires retrieval methods that should be highly efficient. In addition, besides the infeasibility of exhaustive search problem, storage of the original data is also a problem for such data. To tackle such problems, approximate nearest neighbor (ANN) search techniques have been proposed; especially, hashing based ANN methods have attracted more and more attention. The main idea of hashing for ANN is to learn hash functions for converting high-dimension features into compact binary codes while preserving similarity in original space. By this way, hashing enables large gains in data storage and computation speed for similarity search. Thus, it has become a popular method applied to many communities having large scale data. Especially, for documents/multimedia/cross-modal/multimodal retrieval, hashing has shown its superiority for fast semantic similarity approximate nearest neighbors search [1–6].

Generally, exiting hashing methods can be divided into three main categories: unsupervised hashing, semi-supervised hashing and supervised hashing. Unsupervised hashing only uses unlabeled data to generate binary codes. In this paradigm, Locality-Sensitive Hashing (LSH) [7] is a well-known method, especially in computer vision. The basic idea of LSH is to compute randomized hash functions that guarantee a high probability of collision for similar samples. It offers probabilistic guarantees of retrieving items within $(1+\epsilon)$ times the optimal similarity, with sub-linear query time with respect to the number of samples in database [8,9].

Another famous unsupervised hashing method is Spectral Hashing (SH) which was proposed by Weiss et al. [10]. SH uses a separable Laplacian eigenfunction formulation that ends up assigning more bits to directions along which the data has a wider range. Some methods such as IsoHash [11], Bagging PCA [12] and ITQ [13] are based on PCA. Iterative quantization (ITQ) [13] tires to find a rotation of zeros-centered data to minimize the quantization error of mapping data to the vertices of a zero-centered binary hypercube. It has shown good performance; however, the code length of ITQ is limited by the dimension of original data. Similar to ITQ, ok-means and ck-means [14] generalize the ITQ algorithm and Product Quantization (PQ) [15], respectively, to minimize the quantization error. He et al. [16] proposed a non-orthogonal quantization method, K-means hashing, which simultaneously

* Corresponding author.
E-mail address: xuxinshun@sdu.edu.cn (X.-S. Xu).

http://dx.doi.org/10.1016/j.neucom.2015.06.036
0925-2312/© 2015 Elsevier B.V. All rights reserved.

performs k-means clustering and learning the binary indices of the clusters, is a non-orthogonal quantization method. In [17], Heo et al. propose spherical hashing method which is a hypersphere-based binary coding scheme and achieves balanced partitioning of data points for each hash function and independence between hashing functions. There are also some methods utilizing anchor points to approximate the data neighborhood structure, e.g., Anchor Graph Hashing [18] and Compressed Hashing [19].

Unsupervised hashing is promising to retrieve metric distance neighbors when there is no labeled information. However, in some circumstances, we have some pairwise labeled data, e.g., semantic similarity. In order to make use of such labeled data, many semi-supervised or supervised hashing methods have been proposed to generate codes that respect such similarity. For example, Semi-Supervised Hashing (SSH) [20] can leverage semantic similarity using labeled data while remaining robust to overfitting. In addition, it relaxes the orthogonality constraints to allow successive projections to capture more of the data variance. Considering that the sequential learning shows superiority to projection learning, Wang et al. [21] proposed a sequential projection learning version of SSH. Semi-Supervised Discriminant Hashing (SSDH) is also a semi-supervised method developed by Kim et al. [22], which is based on linear discriminant analysis.

There are also some sophisticated supervised hashing methods such as RBM [23], BRE [24] MLSH [25], MFH [26], IMH [27] and MLH [9] which have shown higher search accuracy than unsupervised hashing approaches. However, they all require difficult optimization and are time consuming in training phase. Another supervised hashing method, KSH [28], utilizes the equivalence between optimizing the code inner products and the hamming distances, which also achieves good performance.

From the above, we can see that unsupervised hashing cannot make use of labeled information and supervised hashing has the problems of difficult optimization and time consuming for training. In addition, we can usually get some useful information from unlabeled data. Thus, if a model can not only make use of such information, but also is easy for optimization and efficient in training, it is expected to have good performance and potential for large scale problems. Motivated by this, in this paper, we propose an unsupervised hashing technique in Euclidean space, namely Unsupervised Euclidean Hashing (USEH), which works on unlabeled data; however, it utilizes useful information got from unlabeled data. USEH consists of two phases. In the first phase, USEH uses LSH to generate pseudo labels of training data; then in the second phase, it adopts a sequential learning strategy to learn hash functions, one bit at a time. Moreover, we show that the similarity matrix in USEH is not needed explicitly and can be substituted by other formula. Due to this, when the number of training samples is much greater than the dimension of feature, the time complexity of USEH is dramatically reduced from $O(n^2)$ to $O(n)$ (n is the number of samples in training data), which makes it practicable and efficient on large-scale data.

The remainder of this paper is organized as follows: Section 2 gives a brief introduction of Locality Sensitive Hashing. Section 3 presents the detailed formulations and solutions to the optimization problem in USEH. Experimental results and analysis on two large-scale datasets are presented in Section 4. Finally, Section 5 concludes this paper.

2. Related work

2.1. Locality sensitive hashing

Locality-Sensitive Hashing (LSH) [7,29] is a famous hashing algorithm. It hashes the data to ensure that the probability of collision is much higher for objects that are close to each other than for those that are far apart. For our purpose, we adopt the following LSH scheme.

Given a feature space $\mathbb{M}$, $\mathcal{H}$ is a LSH family, for arbitrary $x, y \in \mathbb{M}$, we have

$$Pr_{h \in \mathcal{H}}[h(x) = h(y)] = s_{\mathcal{H}}(x, y), \tag{1}$$

where $s_{\mathcal{H}}$ is a similarity measure of $\mathbb{M}$. Given an locality sensitive hash function $h \in \mathcal{H}$, we can define the corresponding similarity measure as follows:

$$s_h(x, y) = \begin{cases} 1 & \text{if } h(x) = h(y), \\ 0 & \text{if } h(x) \neq h(y). \end{cases} \tag{2}$$

For a point $p \in \mathbb{R}^d$, an LSH function $h \in \mathcal{H}$ is defined as

$$h(x) = \left\lfloor \frac{\tilde{a} \cdot x + b}{\tilde{r}} \right\rfloor \bmod 2 \tag{3}$$

where $\tilde{a} \in \mathbb{R}^d$ is a random vector with each dimension sampled independently from the standard Gaussian distribution $N(0,1)$, $\tilde{r}$ is the window size and $b \in R$ is sampled from the uniform distribution $U[0, \tilde{r})$. Here, we adopt a simpler form of LSH family

$$h(x) = \frac{\text{sgn}(\tilde{a}x + b) + 1}{2}. \tag{4}$$

As $h(x) \in \{0, 1\}$ and closer data in original space have similar LSH codes, thus we could regard the LSH codes as the pseudo labels of original data.

3. Approach

In this section, we describe the proposed USEH approach in detail, including the formulations, how to learn the project matrix, e.g., learning the orthogonal projection matrix and the non-orthogonal project matrix. In addition, we analyze its computational complexity.

3.1. Notation and problem

Given a training set $X = \{x_1, \ldots, x_n | x_i \in \mathbb{R}^d\} \in \mathbb{R}^{n \times d}$, without loss of generality, we assume the data points are zero-centered, i.e., $\sum_{i=1}^n x_i = 0$. Our goal is to learn the projection matrix $W \in \mathbb{R}^{d \times c}$, where c denotes the code length and every column of W is a projection vector. Then, the binary code matrix $B \in \mathbb{R}^{n \times c}$ could be got by encoding the data set X as $B = \text{sgn}(XW)$, where $\text{sgn}(\cdot)$ is the element-wise sign function, $\text{sgn}(x) = +1$ if $x \geq 0$ and $\text{sgn}(x) = -1$ otherwise.

3.2. Unsupervised Euclidean hashing

3.2.1. Objective function

Given a Euclidean similarity matrix $S \in [-1, +1]^{n \times n}$, we try to reconstruct the similarity matrix by the learned hash codes so that the hash functions could preserve the similarity between the data points. Then, the objective function of the proposed model can be formulated to minimize the following squared error:

$$\min_W E(W) = \min_W \sum_{i,j} \left(\frac{1}{c} \text{sgn}(x_i W) \text{sgn}(x_j W)^\top - S_{ij} \right)^2 \tag{5}$$

Intuitively, if point x_i and x_j are close to each other in original space, then S_{ij} is supposed to be large (e.g., close to 1), in order to minimize the objective function, their corresponding codes should be also similar.

The objective function, i.e., Eq. (5), can also be rewritten in the following matrix form:

$$\min_W E(W) = \min_W \| \text{sgn}(XW) \text{sgn}(XW)^\top - cS \|_F^2 \tag{6}$$