Information Processing and Management 49 (2013) 158–168

Contents lists available at SciVerse ScienceDirect

Information Processing and Management

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

Rank hash similarity for fast similarity search

Min Lu [a], YaLou Huang [a,b], MaoQiang Xie [b,c,*], Jie Liu [a]

[a] College of Information Technical Science, Nankai University, Tianjin, China
[b] College of Software, Nankai University, Tianjin, China
[c] Information Technology Research Base of Civil Aviation Administration of China, Civil Aviation University of China, China

ARTICLE INFO

Article history:
Received 16 September 2011
Received in revised form 12 February 2012
Accepted 10 July 2012
Available online 9 August 2012

Keywords:
Similarity search
Hash function
Similarity ranking

ABSTRACT

The paper is concerned with similarity search at large scale, which efficiently and effectively finds similar data points for a query data point. An efficient way to accelerate similarity search is to learn hash functions. The existing approaches for learning hash functions aim to obtain low values of Hamming distances for the similar pairs. However, these methods ignore the ranking order of these Hamming distances. This leads to the poor accuracy about finding similar items for a query data point. In this paper, an algorithm is proposed, referred to top *k RHS* (Rank Hash Similarity), in which a ranking loss function is designed for learning a hash function. The hash function is hypothesized to be made up of *l* binary classifiers. The issue of learning a hash function can be formulated as a task of learning *l* binary classifiers. The algorithm runs *l* rounds and learns a binary classifier at each round. Compared with the existing approaches, the proposed method has the same order of computational complexity. Nevertheless, experiment results on three text datasets show that the proposed method obtains higher accuracy than the baselines.

1. Introduction

Recently, similarity search (a.k.a. nearest neighbor search) has gained increasing attention in both fields of machine learning and information retrieval. Similarity search has the widespread applications, such as near-duplicate detection (Henzinger, 2006), content-based multimedia retrieval (Lew, Sebe, Djeraba, & Jain, 2006), collaborative filtering (Manning, Raghavan, & Schtze, 2008) and caching (Pandey et al., 2009). The target of fast similarity search is to find the similar data points for a query data point efficiently and effectively from a large dataset.

A common way to accelerate similarity search is to reduce the feature dimensionality. Hash functions are promising among all of the feature reduction methods (Berry, Dumais, & O'Brien, 1995; Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990; He, Cai, Liu, & Ma, 2004; He & Niyogi, 2003). The hash functions design binary codes for the data points, and map the similar data points to the similar codes within short Hamming distances. Several approaches (Andoni & Indyk, 2006; Baluja & Covell, 2008; Pandey et al., 2009; Shi & Malik, 2000; Shakhnarovich, Viola, & Darrell, 2003) have been proposed to learn hash functions. The methods (Andoni & Indyk, 2006; Shakhnarovich et al., 2003; Shi & Malik, 2000) minimize a linear combination of Hamming distances, the weights of which specify the similarity between two data points on the original feature space. The similar pairs have large weights for linear combination. As a result, these methods aim to obtain low values of Hamming distances for the similar pairs. This results in ignoring the ranking order of these Hamming distances. However, in fact similarity search does not care about the values of these Hamming distances but focuses on their ranking

* Corresponding author at: College of Software, Nankai University, Tianjin, China. Tel.: +86 13702194492.
E-mail addresses: lumin@mail.nankai.edu.cn (M. Lu), huangyl@nankai.edu.cn (Y. Huang), xiemq@nankai.edu.cn (M. Xie), jliu@nankai.edu.cn (J. Liu).

0306-4573/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.ipm.2012.07.003

order. The above statements are validated by the retrieval process for similarity search. For a query data point, the retrieval process for similarity search is generally summarized into the three steps:

- The binary codes for the query data point are computed by the hash function.
- The Hamming distances between the query data point and data points in the dataset are calculated.
- The predicted list of data points is returned by ascending order of Hamming distances. The data points with short Hamming distances are ranked top.

The ignorant about ranking order of these Hamming distances decreases the accuracy about finding similar items for a query data point. In addition, the ranking order of the top k data points for similarity search is very crucial. This phenomenon is generated by the users from the similarity search system. The users prefer viewing the top data points on the predicted list to examining the whole predicted list.

To tackle the problems above, an algorithm is proposed, referred to top *k RHS* (Rank Hash Similarity). The proposed method designs a ranking loss function for similarity search. The ranking loss function emphasizes the ranking order of the top k similar data points. A hash function is learned by minimization of the ranking loss function. The hash function is hypothesized to be made up of l binary classifiers. The issue of learning a hash function degenerates to the task of learning l binary classifiers. Since the binary classifiers are not related to each other, a binary classifier can be learned separately at each round. The algorithm runs l rounds and learns l binary classifiers.

Compared with the existing approaches, the top *k RHS* is efficient and more effective. On one hand, the proposed method and the existing approaches have the same order $O(kN)$ of computational complexity where N denotes the size of the dataset. On the other hand, experiment results on three text datasets demonstrate that the proposed method achieves high accuracy than the state-of-the-art techniques.

2. Related work about similarity search

2.1. Overview of similarity search

Similarity search aims to efficiently and effectively find the k nearest neighbors for a query data point from a dataset $\mathcal{X}^{(N)} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_N}\}$. N is the number of data points in the dataset. $\boldsymbol{x_i} \in \mathcal{R}^D$ denotes the i^{th} data point in the dataset where D is the feature dimensionality of data points. In most scenarios, the computational complexity of the retrieval process is order of $O\ (ND)$ since the computational complexity of similarity calculation between two data points is order of $O\ (N)$. In the small scale of a dataset, it efficiently performs the retrieval process because of small values of N and D. Nevertheless, the large values of N and D in the large dataset makes similarity search a huge time-consuming task.

For a large dataset, the way to accelerate similarity search is reducing feature dimensionality. Several state-of-the-arts techniques (Berry et al., 1995; Deerwester et al., 1990) only conduct feature reduction for the already known data points. However, the query data point is usually unseen in many applications. In order to work out for the unknown data points, some methods are proposed to learn a feature mapping function $\phi : \mathcal{X} \mapsto \mathcal{Y}$. $\mathcal{X}$ denotes the original feature space; $\mathcal{Y}$ is the output feature space. The underlying assumptions of the function ϕ are: (i) the dimensionality of the output feature space is lower than that of the original feature space, (ii) the high speed of performing mapping operation and (iii) the similar pairs on the original feature space should be still similar on the output feature space.

There has been great efforts paid by the researchers in learning the feature mapping function ϕ. Based on types of the output feature space $\mathcal{Y}$, the existing approaches can be summarized into two categories: continuous methods and discrete methods. In the continuous methods, the output feature space is real. The representative methods are Locality Preserving Indexing (LPI) (He et al., 2004; He & Niyogi, 2003) and Latent Semantic Indexing (LSI) (Berry et al., 1995). In the discrete methods, the output feature space is sequences of binary codes. In this scenario, the feature mapping function is expressed as $\phi : \mathcal{R}^D \mapsto \{0,1\}^l (l \ll D)$, also known as a hash function.

2.2. Similarity search based on hash functions

Learning a hash function becomes a new trend for similarity search due to the high speed of hash functions in similarity search, which owes to two factors. One factor is that the binary codes of the data points are highly compressed and can be loaded into the main memory. The other factor is that the Hamming distance between the two binary codes is efficiently computed by using bit XOR[1] operation. Millions of bit XOR operation can be performed in a few milliseconds for an ordinary personal computer.

Several approaches have been proposed, including Laplacian Co-Hashing (LCH) (Zhang, Wang, Cai, & Lu, 2010a), Similarity Sensitive Coding (SSC) (Pandey et al., 2009), Forgiving Hashing (FgH) (Baluja & Covell, 2008) and Self Taught Hashing (STH) (Zhang, Wang, Cai, & Lu, 2010b). Their object functions are expressed as a linear combination of Hamming distances, defined as

[1] XOR is performed on two bit patterns of equal length. The result in each position is 1 if the two bits are different, and 0 if they are the same.