Sparse Hashing for Fast Multimedia Search

XIAOFENG ZHU and ZI HUANG, The University of Queensland
HONG CHENG, The Chinese University of Hong Kong
JIANGTAO CUI, Xidian University
HENG TAO SHEN, The University of Queensland

Hash-based methods achieve fast similarity search by representing high-dimensional data with compact binary codes. However, both generating binary codes and encoding unseen data effectively and efficiently remain very challenging tasks. In this article, we focus on these tasks to implement approximate similarity search by proposing a novel hash based method named sparse hashing (SH for short). To generate interpretable (or semantically meaningful) binary codes, the proposed SH first converts original data into low-dimensional data through a novel nonnegative sparse coding method. SH then converts the low-dimensional data into Hamming space (i.e., binary encoding low-dimensional data) by a new binarization rule. After this, training data are represented by generated binary codes. To efficiently and effectively encode unseen data, SH learns hash functions by taking a-priori knowledge into account, such as implicit group effect of the features in training data, and the correlations between original space and the learned Hamming space. SH is able to perform fast approximate similarity search by efficient bit XOR operations in the memory of a modern PC with short binary code representations. Experimental results show that the proposed SH significantly outperforms state-of-the-art techniques.

Categories and Subject Descriptors: H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Hashing, indexing, sparse coding, multimedia search

ACM Reference Format:
Zhu, X., Huang, Z., Cheng, H., Cui, J., and Shen, H. T. 2013. Sparse hashing for fast multimedia search. ACM Trans. Inf. Syst. 31, 2, Article 9 (May 2013), 24 pages.
DOI: http://dx.doi.org/10.1145/2457465.2457469

1. INTRODUCTION

Similarity (or nearest neighbor) search is used to quickly find the most similar data points of a given data point [Huang et al. 2011]. Existing indexing techniques (e.g., kd-tree, R-tree and others) are effective in conducting exact similarity search within low-dimensional data. Recently, it has been shown that hash based methods perform fast approximate similarity search well for high-dimensional data [Kulis and Darrell 2009; Raginsky and Lazebnik 2009; Wang et al. 2010b] and were applied for many kinds of real applications, such as image retrieval [Baluja and Covell 2010; Kulis and Darrell 2009], document analysis [Zhang et al. 2010b], near-duplicate detection [Liu

Authors' addresses: X. Zhu, Z. Huang, and H. T. Shen, School of Information Technology and Electrical Engineering, University of Queensland, Brisbane, QLD 4072 Australia; email: {zhux, huang, shenht}@itee.uq.edu.au; H. Cheng, Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong; email: hcheng@se.cuhk.edu.hk; J. Cui, School of Computer Science and Technology, Xidian University, China; email: cuijt@xidian.edu.cn.

DOI: http://dx.doi.org/10.1145/2457465.2457469

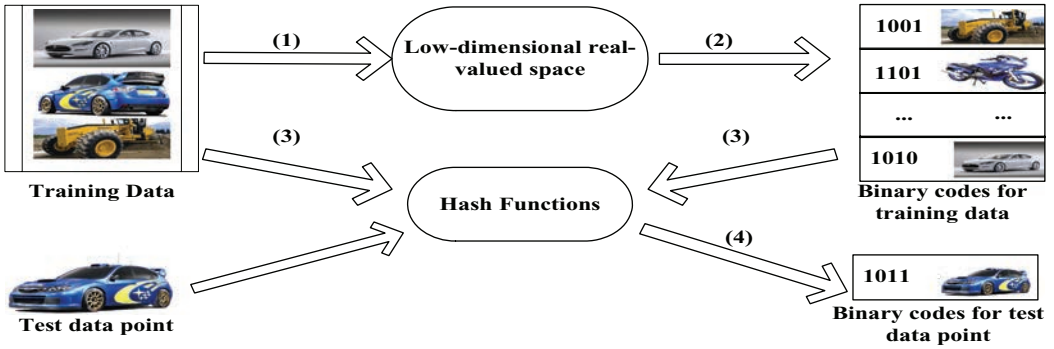


Fig. 1. The framework of hash based methods (including the proposed SH). The ordered numbers from (1) to (4) represent the process of mapping the inputs, generating binary codes, learning hash functions and encoding unseen data, respectively.

et al. 2013; Song et al. 2011; Shang et al. 2010], and so on. For example, spectral hashing [Weiss et al. 2008] encodes all the data points with compact binary codes as well as ensures similar data points to have similar codes such that approximate similarity search is performed on binary codes by quick bit XOR operations in the memory of a modern PC. Even for an exhaustive search on all binary codes, such approximate similarity search is very fast.[1]

The general framework of traditional hash based methods (e.g., [Weiss et al. 2008; Zhang et al. 2010b]) in Figure 1 includes four sequential steps: First, original data are mapped into a low-dimensional real-valued space by the machine learning techniques, for instance, manifold dimensionality reduction [Belkin et al. 2006; He and Niyogi 2003]. Second, the derived low-dimensional data are converted into binary codes by some approaches, such as thresholding method [Salakhutdinov and Hinton 2009; Zhang et al. 2010b]. Third, hash functions are learned for encoding unseen data. The used methods include SVM method [Zhang et al. 2010b], Nystrom method [Drineas and Mahoney 2005], Laplace-Beltrami eigenvalue function method [Weiss et al. 2008], among others. Finally, unseen data are encoded with binary codes by the learned hash functions.

Hash-based methods for approximate similarity search have been focused on learning effectively compact binary codes, that is, effectively and efficiently encoding high-dimensional data into binary codes. Generally, encoding process should meet the constraints [Norouzi and Fleet 2011; Weiss et al. 2008], namely, preserve similarity (i.e., similar data points in original space have similar codes in Hamming space), be efficient (i.e., fewer bits are used to represent a data point, and there is fast indexing and search speed for any a new input), be equivalent (i.e., each bit has equal chance to be 1 or 0), and be independent (i.e., different bits are independent of each other).

In this article, we propose a novel sparse hashing (SH) method to perform fast approximate similarity search. For generating interpretable (or semantically meaningful) binary codes, SH first converts original feature space of data into low-dimensional space, by proposing a nonnegative sparse coding method, namely, SH_LARS algorithm. This process makes each data point to be represented by short binary codes. More concretely, the SH_LARS converts original data into a low-dimensional space (aka basis

[1]Here the notion of "very fast" for exhaustive search of hash codes is relative. An exhaustive search has a linear time complexity, although Hamming distance can be computed efficiently. To achieve sub-linear complexity, one way is to further build indexing structures on hash codes so that only a small portion of hash codes are compared with the query.

space) spanned by the bases. In the basis space, each data point is sparsely represented by the bases, and the corresponding weight is sparse and nonnegative. At the same time, the local similarity of each data point in original space is preserved. SH then converts the low-dimensional data into equivalent-dimensional Hamming space (i.e., binarizing the low-dimensional data) by designing a simple encoding rule, by which each data point is represented by the bases. This process ensures that the local similarity of the data in the low-dimensional space is preserved as much as possible.

To effectively encode unseen data, SH learns hash functions between original data and the learned binary codes by an integrated regression method, namely ENCW algorithm, which is a combination of Elastic Net (EN) [Zou and Hastie 2005] and Curds and Whey (CW) methods [Breiman and Friedman 1997]. The proposed ENCW learns more effective hash functions by taking two types of a-priori knowledge into account, that is, the implicit group effect among training features via EN method, and the correlations between original space and the learned Hamming space via CW method. Existing methods (e.g., Weiss et al. [2008] and Zhang et al. [2010b]) do not consider any a-priori knowledge to generate hash functions. With the learned hash functions, encoding unseen data only involves a single matrix-vector multiplication. This makes the encoding process highly efficient.

The contributions of the proposed SH framework are presented as follows.

—The proposed SH method explores the characteristics of nonnegative sparse coding to generate highly interpretable binary codes. Most existing methods (e.g., spectral hashing [Weiss et al. 2008] and self-taught hashing [Weiss et al. 2008; Zhang et al. 2010b]) also generate negative values in the subspace which can hardly be interpreted for real-world data such as image color histograms.
—SH performs approximate similarity search effectively and efficiently. To effectively generate binary codes of training data, SH preserves the local similarity structures of training data as well as achieves minimal reconstruction error of training data from original space to low-dimensional space. To effectively encode unseen data, SH learns hash functions by taking two kinds of a-priori knowledge into account. To efficiently encode unseen data, SH only involves a single matrix-vector multiplication. This enables SH to be utilized in real applications.
—The experimental results on five real-world datasets show that SH significantly outperforms state-of-the-art techniques (e.g., the methods in [Weiss et al. 2008; Zhang et al. 2010b; Salakhutdinov and Hinton 2009; Zhang et al. 2010a]) in terms of the search quality.

In the remainder of the article, we review the related literature in Section 2. In the subsequent sections, we outline the motivation for our research, followed by details of the SH method. The experimental results are reported in Section 5. Section 6 concludes the article.

2. RELATED WORK

Methods for similarity search generally can be categorized into two types: exact similarity search methods and approximate similarity search methods.

Exact similarity search methods include kd-tree, M-tree, cover tree, metric tree, QUC-tree [Shen et al. 2009] and extended B^+ tree [Huang et al. 2009], among other related methods. These methods usually partition data space recursively to implement exact similarity search in low-dimensional feature space. For example, kd-tree method prebuilds space-partitioning index structures, and R-tree method pre-defines data-partitioning index structures. However, although some literatures (e.g., [Muja and Lowe 2009]) made kd-trees available for searching for approximate nearest neighbor, tree-based methods often leads their complexity to a linear scan in the worst case while

attempting to speed up the computation of similarity search. In this case, tree-based similarity search methods do not perform better than the naive method, that is, a linear scan of the entire dataset, when the dimensionality is slightly high (e.g., >10) [Song et al. 2011; Zhang et al. 2010b]. The dimensionality in real applications is usually very high. In such a scenario, approximate similarity search methods are good alternatives as they can dramatically speed up high-dimensional approximate similarity search into virtually constant time by utilizing hash based methods [Stein 2007; Zhang et al. 2011].

Hash based methods are designed to map original data to a low-dimension Hamming space for approximate similarity search, meanwhile preserving the semantic similarity structure of the data in original space as much as possible. The key point of hash based methods is to generate compact hash codes via developing a hashing method. Existing hash based methods include unsupervised method (e.g., Weiss et al. [2008] and Zhang et al. [2010b]), supervised method (e.g., Jain et al. [2008] and Mu et al. [2010]) and semisupervised method (e.g., Wang et al. [2010a]).

Since the proposed SH in this article belongs to unsupervised method, we give it a brief review in the following part of this section.

The most well-known unsupervised hash based method is probably locality-sensitive hashing (LSH) [Andoni and Indyk 2008; Charikar 2002] and its extensions, such as p-norms [Datar et al. 2004], learned metrics [Jain et al. 2008], and image kernels [Grauman 2007] and among the others [Tao et al. 2009; Raginsky and Lazebnik 2009]. LSH-based methods generate compact hash codes by employing random projection. That is, LSH-based methods map the close data in Euclidean space to with similar binary codes by employing linear random projection followed by a random thresholding method. LSH-based methods theoretically guarantee that approximate results may be found within sub-linear time to the total number of data points. However, while increasing the code length in LSH-based methods, Hamming distance between two binary codes will asymptotically approach to their Euclidean distances. It is not feasible in real applications [Weiss et al. 2008].

Recent, unsupervised hash based methods generate compact hash codes via employing the techniques on machine learning. That is, they improve the drawbacks of LSH-based methods by replace random projection with some novel methods. For example, Torralba et al. [2008] showed that both stacked-restricted boltzmann machine (stacked-RBM for short) method and similarity sensitive coding (SSC) method work significantly better than LSH-based methods while applying to real applications containing tens of millions of data points. He et al. [2011] developed a new hashing algorithm to explicitly optimize search accuracy as well as search time. Liu et al. [2011] proposed a novel anchor graph hashing method to automatically discover the neighborhood structure inherent in the data, aim at learning appropriate compact codes.

The most similar unsupervised hash based methods to the proposed SH are spectral hashing (SpH) [Weiss et al. 2008] and self-taught hashing (STH) [Zhang et al. 2010b]). They have been demonstrated to outperform over LSH, stacked-RBM and SSC for finding originally similar items. This is because they can usually perform real-time search for millions of data with a single modern PC by maintaining all the binary codes in the memory. To encode training data into binary codes, SpH tries to preserve the global similarity structures defined in original space. STH preserves the local similarity structures. To obtain binary codes of unseen data points, that is, encoding test data, SpH assumes that the data should follow uniform distribution, which is very restrictive in real applications. STH works well with any data distribution, and thus is more practical than SpH. However, STH generates hash functions without considering any a-priori knowledge. In contrast, the proposed SH preserves the local similarity structures of original space of the data, as well as achieves minimal reconstruction

error in the derived low-dimensional space. To encode unseen data, SH learns hash functions between original space and Hamming space, by considering the implicit group effect among the features as well as the correlations between training data and the learned Hamming space. The reasonable a-priori knowledge improves the effectiveness of approximate similarity search, as well as induces the efficient encoding unseen data.

3. MOTIVATION

3.1. The Limitations of Traditional Spectral Hashing

Traditional hash-based methods have limitations for improvement. These limitations are mostly related to the generation of interpretable (or semantically meaningful) binary codes, and the encoding of unseen data.

On the one hand, existing methods (e.g., SpH or STH) generate nonzero coefficients (i.e., nonzero coordinates) in low-dimensional real-valued space, then binarize the low-dimensional real-valued data by some methods, such as thresholding method. Although thresholding method is frequently used in practice (e.g., Zhang et al. [2010b] and Weiss et al. [2008]), it can be potentially misleading in various respects [Zass and Shashua 2006] because it is often difficult to interpret the derived results. For example, a negative value in the image color histogram can be hardly explained. Hence, it is desirable to generate interpretable binary codes.

On the other hand, since existing hash based methods do not provide explicit mapping functions for encoding unseen data, the "out of sample extension" problem occurs. To solve this issue, existing hash based methods directly learn mapping functions (i.e., hash functions) between original space and Hamming space. For example, STH learns hash functions via SVM without considering any a-priori knowledge. In this case, we expect to learn effective hash functions by making the full use of a-priori knowledge. Moreover, the efficiency of STH should be improved for encoding unseen data as SVM needs to first map unseen data into the spanned space of training data, in which hash functions are learned. This makes the encoding inefficient.

3.2. The Advantages of Sparse Coding

Sparse coding can be used to represent each image (including training data and unseen test data) with the common parts (i.e., the bases) learned from training images [Raina et al. 2007]. Such an image representation has several interesting advantages. First, the learned bases are high-level because they can be used to represent the images including unseen images (e.g., test data). This can avoid the issue of overfitting, which is often found in existing hash based methods. Second, such a representation is more abstract as the learned bases can describe all images. Third, the learned bases can be well extracted for other different data types [Dai et al. 2008; Lee et al. 2009], such as optical characters, speech audio, and document scripts. Last but not least, the bases are the common parts between training data and test data such that sparse coding can avoid the issue on underfitting or overfitting.

3.3. Motivation

According to this analysis, in this article we employ sparse coding with designed constraints (i.e., preserving the local similarity structure and generating nonnegative sparse codes) to deal with the limitations of traditional hash-based methods, then apply Elastic Net model [Zou and Hastie 2005] to learn hash functions and apply Curds and Whey method [Breiman and Friedman 1997] to strengthen the performance of the learned initial hash functions.

Both sparse representation and nonnegative data are desirable in real applications [Zass and Shashua 2006]. For example, in computer vision, derived coordinates may

correspond to pixels, and nonnegative sparse representation is related to the extraction of relevant parts from images. In machine learning, sparseness is closely related to feature selection, while nonnegativity may relate to probability distributions. In real applications, sparse representation usually makes the proposed algorithm efficiently on storage and speed [Lee et al. 2007; Ghosh 2011], while nonnegative data are with better semantic interpretation. For example, many real applications (e.g., absolute temperatures, light intensities, probabilities, sound spectra, among others) need to have a nonnegative weight.

3.4. Framework

The proposed SH framework (presented in Figure 1) for fast approximate similarity search includes four steps, namely, mapping the inputs, generating binary codes, learning hash functions and encoding unseen data. Of these four steps, the first two are proposed for generating effective codes, and the last two for efficiently and effectively encoding unseen data. The corresponding methods for the four steps are SH_ LARS algorithm, a proposed binarization rule, ENCW algorithm and the process of encoding unseen data respectively.

In the first step, SH converts original data into a low-dimensional space by learning a succinct (i.e., parsimonious) and high-level representation of the inputs. During the learning process, the constraints (i.e., similarity preservation and nonnegative sparse codes) are taken into account. Similarity preservation is necessary to ensure that semantically similar data points in original space are still similar in low-dimensional space. Nonnegative constraint is used for generating semantic binary codes. With the sparse coding model, each data point is succinctly represented by high-level bases. The high-level characteristics can avoid the issue of overfitting or underfitting. However, in traditional hash based methods (e.g., SpH or STH), when test data are with different distribution to training data, either the issue of overfitting or the issue of underfitting often occurs. Representing each data point parsimoniously makes the proposed SH more efficient. Unlike the traditional dimensionality reduction methods, such as PCA, which should satisfy orthogonality constraints among features, SH independently generates nonnegative sparse codes (see details in Section 4.1.2).

In the second step, SH converts the low-dimensional and nonnegative real-valued data into Hamming space by proposing a new binarization rule. That is, positive values in the low-dimensional space are converted to 1 in the Hamming space, and zeros are converted to 0. Note that SH does not have negative real-valued data in the low-dimensional space, which is different from existing methods.

In the third step, SH implements unvariate regression between original space and each bit vector in the learned Hamming space to learn one hash function for each bit vector, by proposing the ENCW algorithm. ENCW first employs EN model, which conducts regression by considering implicit group effect among the features of training data, to learn the relationship between original training data and each bit vector in Hamming space. After receiving initial regression results, CW model utilizes the correlations between original space and learned Hamming space to obtain a more accurate encoding. Thus, the proposed ENCW algorithm can more effectively encode unseen data by making the best use of a-priori knowledge.

Finally, unseen data can be easily encoded by the learned hash functions. The learned hash functions are more efficient because the encoding process is only a single matrix-vector multiplication.

3.5. Notation

The notation used in this article is represented as follows: feature space is in uppercase italic, a scalar is in lowercase italic, a column vector is in lowercase bold, a matrix is

in uppercase bold, transposes of a vector (or a matrix) are indicated by a superscript T, and the inverse of a matrix is represented by a superscript -1.

4. APPROACH

4.1. Mapping the Inputs

4.1.1. Objective Function. In order to maintain the data into the memory of a PC, we represent each original data point with short binary codes. Meanwhile, we expect that similar data in original space are still similar in low-dimensional space. In this article, the proposed SH utilizes nonnegative sparse coding method[2] to sparsely represent each data point, as well as adds similarity preservation constraint into nonnegative sparse coding model.

Sparse coding is a popular reconstruction technique due to the ability of the succinct representation to save time [Tibshirani 1994; Mairal et al. 2010]. The reconstruction estimation for sparse coding can be obtained by minimizing the loss function under a penalized constraint, that is,

$$\min_{\mathbf{s}} \quad \|\mathbf{x} - \mathbf{B}\mathbf{s}\|_2^2 + \lambda\|\mathbf{s}\|_1, \tag{1}$$

where $\|.\|_p$ is the ℓp vector norm, $\mathbf{x} \in \boldsymbol{R}^d$ represents the data point in original space, $\mathbf{B} \in \boldsymbol{R}^{d\times m}$ represents the bases, and $\mathbf{s} \in \boldsymbol{R}^m$ is sparse codes (i.e., the weights of the bases, or the coordinates of the data point in low-dimensional space) of $\mathbf{x}$ with respect to $\mathbf{B}$. m is the number of reduced dimensionality (or the number of bases, or the number of bits in Hamming space). $\|\mathbf{s}\|_1$ ensures the sparsity, that is, many of $s_i (i = 1, \ldots, m)$ are zeros while adjusting the values of λ (where $\lambda > 0$ is a regularization parameter).

SH first uses least square loss function to achieve minimal reconstruction error during the reconstruction process, see the first term in Equation (1). Usually, an efficient hash based method for fast approximate similarity search should preserve similarity in original space. That is, similar data points in original space should be mapped with similar codes.

In the existing hash based methods, SpH [Salakhutdinov and Hinton 2009] preserves the global similarity structures of training data to implement fast similarity search. Actually, local similarity structures are often more important than global one [Roweis and Saul 2000]. For example, Wu and Schölkopf [2007] reported that local learning algorithms are often better than global ones. Moreover, in real applications, we often prefer to index the top k similar data points to the query data point rather than retrieve all of its similar data points. Furthermore, the local similarity structures can be extended to global similarity structures by setting the value of k as the value of the number of training data, that is, $k = n$. Hence, in this article, we focus on preserving the local similarity structure by building a k-nearest-neighbor graph for each data point.

More specifically, following the idea in Belkin et al. [2006], we use a heat kernel $w_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma}}$ (σ is a tuning parameter, we set $\sigma = 1$ in this article) to build a weight matrix $\mathbf{W}$. The value of w_{ij} is used to measure the closeness of two points $\mathbf{x}_i$ and $\mathbf{x}_j$. We set $w_{ii} = 0$ to avoid the problem of scale in this article.

Given a weight matrix $\mathbf{W}$, we use the Euclidean distance to measure the smoothness between $\mathbf{s}_i$ and $\mathbf{s}_j$ (where $\mathbf{s}_i$ (or $\mathbf{s}_j$) is sparse codes of $\mathbf{x}_i$ (or $\mathbf{x}_j$) respectively in

[2]Unlike the approach in [Hoyer 2004] and others, in which both the bases and sparse codes are nonnegative, SH only requires nonnegative sparse codes. Hence, the proposed method can also be one of the methods on positive Lasso [Efron et al. 2004].

low-dimensional space), that is,

$$\begin{aligned}\frac{1}{2}\sum_{i,j}\|\mathbf{s}_i-\mathbf{s}_j\|^2 w_{ij} &= \sum_{i,j}\mathbf{s}_i D_{ii}\mathbf{s}_i^T - \sum_{i,j}\mathbf{s}_i\mathbf{s}_j^T w_{ij}\\ &= tr(\mathbf{SDS}^T) - tr(\mathbf{SWS}^T)\\ &= tr(\mathbf{SLS}^T). \end{aligned} \quad (2)$$

We denote $\mathbf{D}$ as a diagonal matrix. The entries of $\mathbf{D}$ are the column (or row, since $\mathbf{W}$ is symmetric) sum of $\mathbf{W}$, that is, $D_{ii} = \sum_j w_{ij}$. Obviously $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is a Laplacian matrix.

In this article, both local similarity structures of training data presented in Equation (2) and nonnegative constraint are added into sparse coding model in Equation (1). More concretely, given a training data matrix $\mathbf{X} = (x_{ij}) \in \boldsymbol{R}^{d\times n}$ (each column represents a data point), we want to learn the bases $\mathbf{B} \in \boldsymbol{R}^{d\times m}$ and the corresponding sparse codes $\mathbf{S} \in \boldsymbol{R}^{m\times n}$ from training data $\mathbf{X}$ with the constraints, such as preserving local similarity structure, and generating nonnegative sparse codes. The objective function of the proposed SH is defined as:

$$\min_{\{\mathbf{B},\mathbf{S}\}} \|\mathbf{X} - \mathbf{BS}\|_F^2 + \alpha tr(\mathbf{SLS}^T) + \lambda\sum_{i=1}^n \|\mathbf{s}_i\|_1 \qquad s.t.\ \|\mathbf{B}_j\|^2 \le 1,\ \mathbf{S} \succeq 0. \quad (3)$$

$\|.\|_F$ means Frobenius matrix norm. $\mathbf{S} \succeq 0$ (or $\mathbf{s} \succeq 0$) indicates each element in matrix $\mathbf{S}$ (or vector $\mathbf{s}$) is nonnegative. $\|\mathbf{B}_j\|^2 \le 1, j = 1, \ldots, m$ is to prevent $\mathbf{B}$ from having arbitrarily large values which would lead to very small values of $\mathbf{S}$.

For generating semantic binary codes, the proposed objective function in Equation (3) reconstructs each data point to achieve the objectives, such as obtaining minimal reconstruction error (i.e., first term in Equation (3)), preserving the local similarity structure (i.e., second term in Equation (3)), and generating nonnegative sparse codes (i.e., $\mathbf{S} \succeq 0$). The constraint on minimal reconstruction error is to achieve minimal loss after the reconstruction process. The constraint on local similarity preservation is to ensure that each data point with its k-nearest neighbors is still similar in low-dimensional space. The constraint of nonnegative sparse codes is for generating semantic binary codes. Moreover, the nonnegative sparse coding model usually imposes fewer constraints than Lasso model [Tibshirani 1994]. This enables the objective function in Equation (3) to be solved quickly. However, SpH only preserves global similarity and STH only preserves local similarity for mapping the inputs to low-dimensional real-valued space. Moreover, when the value of the parameter α reaches a larger value, the objective function in Equation (3) can be regarded as only considering the constraint on preserving local similarity structure of the data. This indicates that STH is a special case of the proposed SH method. In this way, SpH is also a special case of the proposed SH method when the value of k is set as the value of the number of training data and the parameter α reaches a larger value.

4.1.2. Implementation of the Objective Function. Two variables (i.e., $\mathbf{B}$, and $\mathbf{S}$) in Equation (3) should be optimized. The objective function in Equation (3) is not jointly convex for both $\mathbf{B}$ and $\mathbf{S}$. Hence, following the work in [Lee et al. 2007; Gao et al. 2010; Zheng et al. 2011], we optimize $\mathbf{B}$ and $\mathbf{S}$ alternatively in an iterative process.

When $\mathbf{S}$ is fixed in Equation (3), we learn the bases $\mathbf{B}$ by optimizing the following objective function with a conjugate gradient decent method presented in Lee et al. [2007],

$$\min_{\mathbf{B}} \|\mathbf{X} - \mathbf{BS}\|_F^2 \qquad s.t.\ \|\mathbf{B}_j\|^2 \le 1, j = 1, \ldots, m. \quad (4)$$

Note that we initialize sparse codes $\mathbf{S}$ as the results of the general sparse coding.

When fixing $\mathbf{B}$ to learn $\mathbf{S}$, we optimize each column vector $\mathbf{s}$ in $\mathbf{S}$ one by one rather than optimize all the vectors in $\mathbf{S}$ simultaneously because different bits should be independently generated according to the constraints of hashing. Thus, the objective function in Equation (3) on one column vectors $\mathbf{s}$ becomes

$$\min_{\mathbf{s}} \quad \|\mathbf{x} - \mathbf{B}\mathbf{s}\|_2^2 + \alpha(2\mathbf{s}^T(\mathbf{S}\mathbf{l}_i) - \mathbf{s}^T l_{ii}\mathbf{s}) + \lambda \sum_{j=1}^{m} \|s_j\|_1 \;\; s.t.\; \mathbf{s} \succeq 0. \tag{5}$$

where $\mathbf{l}_i$ is the i-th column of $\mathbf{L}$, and l_{ii} is the element in the i-th column and i-th row of $\mathbf{L}$.

To solve $\mathbf{s}$ in Equation (5), unlike the works in Lee et al. [2007] which is focused on the general sparse coding framework[3], SH generates nonnegative sparse codes for each data point. In this article, we first convert Equation (5) to standard form of Lasso [Efron et al. 2004; Olshausen and Field 1996] by Theorem 1 and then obtain the optimal $\mathbf{s}$ by Algorithm 1.

THEOREM 4.1. *The objective function in Equation (5) is equivalent to:*

$$\min_{\boldsymbol{s}} \quad \|\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{B}}\boldsymbol{s}\|_2^2 + \lambda \sum_{j=1}^{m} \|s_j\|_1 \quad s.t.\; \boldsymbol{s} \succeq 0., \tag{6}$$

where $\tilde{\boldsymbol{B}}^T\tilde{\boldsymbol{B}} = (\boldsymbol{B}^T\boldsymbol{B} + \alpha l_{ii}\boldsymbol{I})$, $\tilde{\boldsymbol{x}} = (\tilde{\boldsymbol{B}}^T)^{-1}(\boldsymbol{B}^T\boldsymbol{x} + \alpha\boldsymbol{S}\boldsymbol{w}_i)$, $\tilde{\boldsymbol{x}} \in \mathbf{R}^d$.

PROOF. We denote $\mathbf{S}_{-i} = (\mathbf{s}_1, \ldots, \mathbf{s}_{i-1}, \mathbf{s}_{i+1}, \ldots, \mathbf{s}_n)$ as $\mathbf{S}$ without the column vector $\mathbf{s}_i$, and $\mathbf{l}_{-ii}$ ($\mathbf{d}_{-ii}$ and $\mathbf{w}_{-ii}$) as the column vector $\mathbf{l}_i$ ($\mathbf{d}_i$ and $\mathbf{w}_i$) without the element l_{ii} (d_{ii} and w_{ii}). Due to $w_{ii} = 0$, we know $\mathbf{l}_{-ii} = \mathbf{d}_{-ii} - \mathbf{w}_{-ii} = -\mathbf{w}_{-ii}$, then $\mathbf{S}_{-i}\mathbf{l}_{-ii} = -\mathbf{S}_{-i}\mathbf{w}_{-ii} = -\mathbf{S}_{-i}\mathbf{w}_{-ii} - \mathbf{s}w_{ii} = -\mathbf{S}\mathbf{w}_i$. Thus

$$\begin{aligned} 2\mathbf{s}^T(\mathbf{S}\mathbf{l}_i) - \mathbf{s}^T l_{ii}\mathbf{s} &= 2\mathbf{s}^T(\mathbf{s} \quad \mathbf{S}_{-i})\begin{pmatrix} l_{ii} \\ \mathbf{l}_{-ii} \end{pmatrix} - \mathbf{s}^T l_{ii}\mathbf{s} \\ &= \mathbf{s}^T l_{ii}\mathbf{s} + 2\mathbf{s}^T\mathbf{S}_{-i}\mathbf{l}_{-ii} \\ &= \mathbf{s}^T l_{ii}\mathbf{s} - 2\mathbf{s}^T(\mathbf{S}\mathbf{w}_i). \end{aligned}$$

Hence,

$$\begin{aligned} &\min_{\mathbf{s}} \quad \|\mathbf{x} - \mathbf{B}\mathbf{s}\|_2^2 + \alpha(2\mathbf{s}^T(\mathbf{S}\mathbf{l}_i) - \mathbf{s}^T l_{ii}\mathbf{s}) \\ \Longleftrightarrow\; &\min_{\mathbf{s}} \quad \mathbf{s}^T(\mathbf{B}^T\mathbf{B} + \alpha l_{ii}\mathbf{I})s - 2\mathbf{s}^T(\mathbf{B}^T\mathbf{x} + \alpha\mathbf{S}\mathbf{w}_i) \\ \Longleftrightarrow\; &\min_{\mathbf{s}} \quad \|\tilde{\mathbf{x}} - \tilde{\mathbf{B}}\mathbf{s}\|_2^2. \end{aligned}$$

Note that, $\tilde{\mathbf{B}}^T\tilde{\mathbf{B}}$ is a positive semi-definite matrix. In this article we employ Cholesky decomposition method to obtain $\tilde{\mathbf{B}} \in R^{m\times m}$.

According to Lykou and Whittaker [2010], Equation (6) is a convex problem and the KKT conditions are necessary and sufficient conditions for a global optimum. In this article, we solve the nonnegative Lasso in Equation (6) by designing a revised least angle regression (LARS) algorithm (referred to as SH_ LARS algorithm) presented in Algorithm 1.

[3]In this article, general sparse coding means that the generated sparse codes can be any real-valued.

ALGORITHM 1: SH_ LARS algorithm

Input: $\mathbf{x}$ and $\mathbf{B}$
Output: $\mathbf{s}$
Normalize $\mathbf{B}$; center $\mathbf{x}$;
$s_1, \ldots, s_m = 0$;
$\mathbf{c} = \mathbf{B}^T\mathbf{x}$; $C = max_j\{c_j\}$; $j = argmax_j\{c_j\}$; $AS = \{j\}$;
repeat
  $\mathbf{G}_A = (\mathbf{B}^T\mathbf{B})^{-1}$;$A_A = (\mathbf{1}_A^T\mathbf{G}_A\mathbf{1}_A)^{-\frac{1}{2}}$;$\mathbf{u}_A = \mathbf{B}(A_A\mathbf{G}_A\mathbf{1}_A)$;
  $\mathbf{a} = \mathbf{B}^T\mathbf{u}_A$; $\mathbf{c} = \mathbf{B}^T(\mathbf{x} - \mathbf{u}_A)$; $C = max_j\{c_j\}$;
  if |AS| $< k$ **then**
    $\gamma = min_{\{j \notin AS\}}\{\frac{C - c_j}{A_A - a_j}\}$
  else
    $\gamma = \frac{C}{A_A}$. *// γ: the maximal length of the next step updated*
  end
  Lasso Modification *//same as in* [Efron et al. 2004];
  Update AS *//same as in* [Efron et al. 2004];
until *meeting the pre-defined conditions*;
return $\mathbf{s}$;

As a forward selection method, the SH_LARS starts with all the coefficients $s_i = 0$ $(i = 1, \ldots, m)$ (line 2) after finishing the pre-processing (line 1), and the finds the first predictor $\bar{s}_p$ (where $\bar{s}_p \in \{s_1, \ldots, s_m\}$) which is most correlated to $\mathbf{B}$ (line 3). Furthermore the SH_LARS takes the largest step possible in the direction of $\bar{s}_p$ until another predictor $\bar{s}_q$ has as much correlation with the current residual. Instead of continuing along $\bar{s}_p$, the SH_LARS proceeds in the equiangular direction between $\bar{s}_p$ and $\bar{s}_q$, until a third predictor $\bar{s}_r$ earns its way into the most correlated set AS. The SH_LARS then proceeds equiangularly between $\bar{s}_p$, $\bar{s}_q$ and $\bar{s}_r$, that is, along the least angle direction, until a fourth predictor enters, and so on (Repeat from line 4 to line 11). The SH_ LARS stops after meeting the predefined conditions, that is, reaching convergence.

After the alternative optimization we have illustrated, we get the bases $\mathbf{B}$ and the sparse codes $\mathbf{S}$ of training data.

4.2. Generating Binary Codes

In the step of mapping the inputs, SH maps original data to low-dimensional real-valued space to obtain the bases and sparse codes of training data. Meanwhile, the local similarity structure for each data point is preserved from original space to low-dimensional real-valued space.

We need to further represent each data point with binary codes for the purpose of saving storage and facilitating efficient search later. We also need to design the binarization rule to generate interpretable binary codes. In this step, SH converts the low-dimensional data into compact Hamming space by a simple binarization rule: encoding each positive value in low-dimensional nonnegative real-valued space into 1, and zero into 0. For example, a data point A(0.4,0,0.1,0.7) (top row in Figure 2) is encoded as '1011' (bottom row in Figure 2) in Hamming space.

The proposed binarization rule can be naturally interpreted for real-world data. In low-dimensional real-valued space, each data point is sparsely represented by the bases. The positive weight (i.e., sparse codes) in the i-th dimension means that the data point is composed by the i-th basis. The value of the weight means the important degree of the i-th basis to the data point. The zero weight in the i-th dimension indicates that the data point does not contain the i-th basis. For example, given a data point A (0.4, 0,

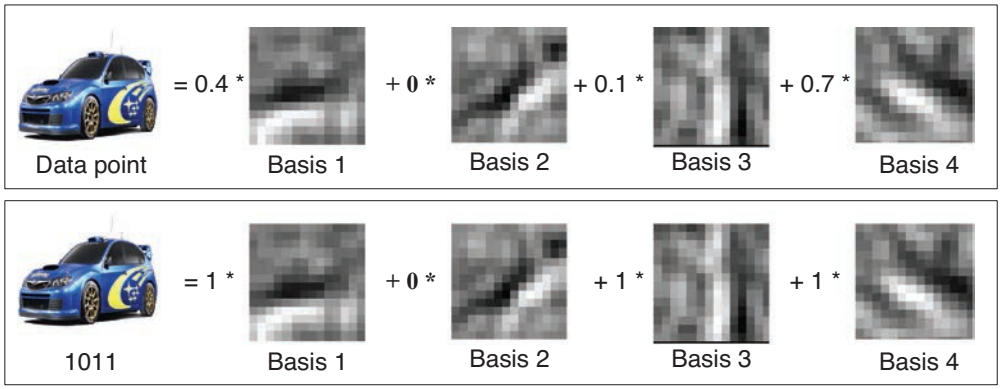


Fig. 2. An illustrated example of the proposed binarization rule. Top row: the data point is sparsely represented by four bases in low-dimensional real-valued space. Bottom row: the data point is represented in low-dimensional Hamming space.

0.1, 0.7), it is represented by three bases (i.e., first, third and fourth basis) and does not contain the second basis as the corresponding weight in the second dimension is zero. The important degrees of three bases are 0.4, 0.1 and 0.7. Hence, we can say that the data point A is composed by three bases in low-dimensional nonnegative space. Note that existing hashing methods (e.g., STH) which transfers the values less than the median to 0 and the values greater than the median to 1 do not make such semantic interpretation, because their negative values in the low-dimensional space cannot naturally explain the 'contain' relationship. According to the proposed binarization rule, the data point A (i.e., '1011', presented in the bottom row in Figure 2) is still composed of the same three bases in Hamming space. We can also easily know that the data composed by the same bases are similar in low-dimensional nonnegative space. For example, data point B (0.51, 0.51, 0.51, 0.51) and C (0.49, 0.49, 0.49, 0.49) are similar in low-dimensional nonnegative space because they are composed of all four bases. They are encoded as '1111' in Hamming space and still similar in Hamming space because their Hamming distance is zero.

Although the SH method does not satisfy all the constraints of hash-based methods (e.g., each bit has equal chance to be 1 or 0), SH can achieve better performance and generate more semantic codes by relaxing such a constraint into the case in which each bit has an arbitrary probability.

4.3. Learning Hash Functions

4.3.1. The "Out of Sample Extension" Issue. After converting training data into binary codes, hash based methods need to learn hash functions for encoding binary codes for unseen data, that is, test data. That is because training process does not generate explicit mapping functions for encoding unseen data. This problem is referred to as "*out-of-sample extension*" , whose existing solutions include Nystrom method [Drineas and Mahoney 2005], Laplace-Beltrami eigenvalue function method [Salakhutdinov and Hinton 2009], SVM method [Zhang et al. 2010b], among others.

Nystrom method for encoding unseen data is computationally expensive since an exhaustive similarity search is performed over the whole training data. This makes it impractical in real applications. Laplace-Beltrami eigenvalue function method is faster than Nystrom method, but it assumes the data should follow uniform distribution. SVM method overcomes these limitations and is faster than Laplace-Beltrami eigenvalue

function method for encoding unseen data. However, the learning process for generating hash functions by SVM method does not consider any a-priori knowledge.

SH can generate explicit functions by mapping test data into the bases space via the propose objective function in Equation (6). Yet such a process requires the running of iterative minimization algorithm, which is always accompanied by expensive computation cost. Hence, an alternative is necessary. To efficiently encode unseen data, we learn hash functions to encode unseen data. Note that the proposed SH can generate explicit mapping functions for encoding unseen data, that is, via Equation (6). In this article, to efficiently encoding unseen data, we design a new method (namely the ENCW algorithm) to learn hash functions.

Unlike SVM method for solving the "out of sample extension" issue, SH first employs Elastic Net (EN) model [Zou and Hastie 2005], which can take into account implicit group effect among the features of training data, to conduct regression between training data and each column (i.e., bit vector) in Hamming space, for generating hash function of each bit vector. Then the initial result is fed into CW model [Breiman and Friedman 1997] for improving initial encoding accuracy.

4.3.2. ENCW Method. To solve the issue of "*out-of-sample extension*," we learn explicit hash functions between original training features and binary codes learned from the first two steps in the proposed framework presented in Figure 1. Note that the proposed method is still unsupervised because the label information (i.e., the binary codes of training data) is learned rather than given in advance.

More specifically, given d-dimensional training data and the learned m-dimensional binary codes, we build m explicit hash functions (i.e., classifiers) via the proposed ENCW method, which first performs unvariate regression (or classification) m times via EN model and then implements Curds and Whey (CW) method for improving encoding accuracy. The EN model learns one classifier between the d-dimensional training data and one vector (i.e., column) of the learned m-dimensional binary codes once. Then it outputs m hash functions. The CW method uses canonical correlation analysis (CCA) model to improve the encoding performance of m hash functions.

Both EN model and Lasso model are usually used for performing unvariate regression [Breiman and Friedman 1997; Ghosh 2011]. In this article, we select EN model rather than Lasso [Efron et al. 2004] because, first, EN model can simultaneously achieve accuracy and sparsity. Second, EN model encourages grouping effect, where strongly correlated predictors (or encoders) tend to be in or out of the model together [Ghosh 2011]. However, Lasso tends to select only one variable from implicit group and does not care which one is selected. Third, EN model is particularly useful when the number of predictors (or encoders) (d) is much bigger than the number of observations (n) [Zou and Hastie 2005]. In such a case, Lasso selects at most n variables before it saturates, because of the nature of the convex optimization problem. However, EN selects all of them for reconstruction process. All the scenarios are often found in real applications. Hence, employing EN model to use such a-priori knowledge is reasonable.

In the proposed EN model, for any fixed nonnegative λ_1 and λ_2, denoting the learned m-dimensional binary codes as $\mathbf{Y}$, where $\mathbf{Y} \in \boldsymbol{R}^{n\times m}$ is centered, that is, $\sum_{i=1}^{n} y_i = 0$, denoting d-dimensional standardized training data as $\mathbf{X} \in \boldsymbol{R}^{d\times n}$, that is, $\sum_{i=1}^{n} x_{j,i} = 0$, for each column $\mathbf{y} \in R^n$ in $\mathbf{Y}$, the EN model is defined as:

$$\min_{\beta_i} \|\mathbf{y}_i - \mathbf{X}^T\beta_i\|_2^2 + \lambda_1\|\beta_i\|_1 + \lambda_2\|\beta_i\|^2, \tag{7}$$

where β_i ($\beta_i \in \boldsymbol{R}^d, i = 1, \ldots, m$) is the elastic net estimator (or coefficient), that is, the transformation coefficient of the i-th hash function (or classifier). Therefore, given

unseen data point $\mathbf{x}_t$, we obtain its binary codes of the i-th hash function by $\mathbf{y}_i^t = sign(\mathbf{x}_t\beta_i)$. Denoting $\beta = (\beta_1, \ldots, \beta_m)$, m binary codes of $\mathbf{x}_t$ can be encoded by $sign(\mathbf{x}_t^T\beta)$.

The Curds and Whey method (CW) [Breiman and Friedman 1997] employs canonical correlation analysis (CCA) to make full use of information in correlated responses, for improving the encoding (or classification) of the learned hash functions via the EN model. The CW method has been shown (e.g., Breiman and Friedman [1997]) that encoding errors of the hash functions are substantially reduced when the responses are correlated, while maintaining accuracy if they are uncorrelated. Actually, the responses $\mathbf{y}_i$ (where $i = 1, \ldots, m$) are correlated because they are dependent on the variables $\mathbf{X}$. Therefore, it is reasonable for using CW method to boost encoding performance given initial results derived from EN model.

Following the literature [Breiman and Friedman 1997], CW method searches for an optimal matrix $\mathbf{C}$ ($\mathbf{C} \in \boldsymbol{R}^{m\times m}$ and m is the number of hash functions) to make the use of the disparity of m classifiers learned from the EN model. The pseudo of ENCW method is presented in Algorithm 2.

ALGORITHM 2: ENCW algorithm

Input: $\mathbf{X}, \mathbf{Y}, \mathbf{x}_t, \lambda_1, \lambda_2$
Output: $\hat{\mathbf{y}}_t$
Perform CCA on X and Y;
Calculate matrix $\mathbf{W}$ by Equation (9);
for *each* $\boldsymbol{y}_i$ *in* $\boldsymbol{Y}$ **do**
  Compute vector β_i by Equation (7),
end
Compute $\hat{\mathbf{y}}_t = \mathbf{x}_t\beta\mathbf{C}^T$;

More specifically, first, CW method performs CCA [Hotelling 1936] between $\mathbf{Y}$ and $\mathbf{X}$ to obtain $r_i = \max corr(\mathbf{Yu}_i, \mathbf{X}^T\mathbf{v}_i), i = 1, \ldots, m$ (m = min(m,n), actually, $m \ll n$), $\mathbf{U} = \{\mathbf{u}_1, \ldots, \mathbf{u}_m\}$ and $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$. $\mathbf{u}_i$ and $\mathbf{v}_i$ are the vectors such that the correlation between the linear combinations of the response $\mathbf{Yu}_i$ and $\mathbf{X}_t^T\mathbf{v}_i$ is maximized.

Second, we set $\mathbf{W} = diag\{w_1, \ldots, w_m\}$ as an $m\times m$ diagonal matrix of shrinkage factors with

$$w_i = max\left\{\frac{(1-f)(r_i^2 - f)}{(1-f)^2 r_i^2 + f^2(1-r_i^2)}, 0\right\}, \quad i = 1, \ldots m, \tag{8}$$

where r_i is the canonical correlations between $\mathbf{y}_i$ and $\mathbf{X}$, and $f = d/n$.

Third, according to Breiman and Friedman [1997], the optimal matrix $\mathbf{C}$ is obtained by:

$$\mathbf{C} = \mathbf{U}^{-1}\mathbf{WU}. \tag{9}$$

Finally, given unseen data point $\mathbf{x}_t$ and $\beta = (\beta_1, \ldots, \beta_m)$ learned from the EN model, m binary codes of $\mathbf{x}_t$ can be encoded by $sign(\mathbf{x}_t\beta\mathbf{C}^T)$.

4.4. Encoding Unseen Data

SH generates the binary codes of training data by Algorithm 1, and hash functions by Algorithm 2. Given a test data point $\mathbf{x}_t$, SH maps it into Hamming space and obtains its m-bit binary codes by $sign(\mathbf{x}_t^T\beta(\mathbf{U}^{-1}\mathbf{WU})^T)$.

4.5. Complexity

The complexity of SH is mainly decided by the processes, that is, constructing kNN graph, the LARS algorithm, the binarization rule, and the encoding process.

Building kNN graph takes $O(n^2k)$ (where n is training size) [Cormen et al. 2001]. The complexity of LARS algorithm is relevant to the dimensionality d (or the number of bits m), and the number of training size n. If $d < n$, the complexity of LARS is $O(d^3 + d^2n)$ which is linear to the number of training size. If $d > n$, its complexity is $O(d^3)$ which is irrelevant to n [Efron et al. 2004]. Moreover, according to Theorem 1 and Algorithm 2, the proposed algorithms (i.e., SH_LARS and ENCW) are relevant to m rather than d. Actually, we know $d \gg m$ in the real applications and $m \ll n$ in our experiments, for example, the value of m is 64. Hence, the proposed LARS method in this article is $O(m^3 + m^2n)$ for ENCW algorithm and $O(m^3 + m^2d)$ for SH_LARS algorithm, which is at most linear to the number of training data. The proposed binarization rule encodes the real value to the binary codes with constant time. However, the theresholding method (e.g., the median-based binarization in SpH and STH) usually takes O(mn) time. Therefore the overall computational complexity of the training process is roughly quadratic to the number of training size. That is, the complexity of the training process is focused on constructing the kNN graph, which is equivalent to STH.

The encoding process of SH for a given query is simply to classify the test data using those m learned classifiers and then to assemble the output m binary labels into m-bit binary codes. Actually, the encoding process of SH is a single matrix-vector multiplication. However, for the STH method with linear SVM, the overall computational complexity of the encoding process for each test case is linear to the size of the test case.

5. EXPERIMENTAL ANALYSIS

In this section, we empirically evaluate the proposed SH method and compare it with recent methods, including LSH [Charikar 2002], STH [Zhang et al. 2010b], LSI [Salakhutdinov and Hinton 2009], LCH [Zhang et al. 2010a], MLH [Norouzi and Fleet 2011], and SpH [Weiss et al. 2008]. To separately test the reconstruction process and the process of learning hash functions, we use two versions of SH, namely, SH_ENCW and SH_SVM. SH_ENCW implements Algorithm 1 and Algorithm 2 together. SH_SVM implements Algorithm 1 and uses SVM for learning hash functions.

First, we evaluate the score of F1-measure of the proposed method (i.e., SH_ENCW) on different Hamming ball radius, for testing the stability of the proposed SH_ENCW algorithm. Then we compare the proposed methods (i.e., SH_ENCW and SH_SVM) with the comparison algorithms in terms of precision-recall curves, on the different bits for representing each data point. Furthermore, we test the individual effects of the SH_LARS algorithm (i.e., Algorithm 1) and the ENCW algorithm (i.e., Algorithm 2). Finally, we compare the proposed method to the comparison algorithms on encoding time, aim at testing the efficiency of encoding unseen data.

5.1. Experimental Setting

5.1.1. Data. We conduct experiments on four real datasets, that is, USPS,[4] Reuters21578,[5] 20Newsgroups,[6] and MNIST.[7]

The Reuters21578 corpus is a set of documents that appeared on the Reuters newswire in 1987. In our experiments, we discard the documents appearing in more than one category, and keep only the largest 10 categories. We use 7285 documents in

[4]http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/multiclass.html#usps.
[5]http://www.daviddlewis.com/resources/testcollections/reuters21578/.
[6]http://people.csail.mit.edu/jrennie/20Newsgroups/.
[7]http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

total, then employ the ModeApte split method to generate 5228 documents (72%) as training data and 2057 documents (28%) as test data. The 20Newsgroups corpus was collected and originally used for document analysis. In this article, it contains 18846 documents, evenly distributed across 20 categories, and 11314 documents (60%) for training data and 7532 documents (40%) for test data. In our experiments, both documents datasets are represented by a bag-of-words with a 500-word vocabulary. Hence, the size of original datasets Reuters21578 and 20Newsgroups are 7285 and 18846, and the number of features is 500.

USPS and MNIST are image databases. Both of them are handwritten digit databases. USPS contains 9298 images in 10 categories in which 7291 images are training data and 2007 are test data, and the raw feature of images has 16×16 pixel intensities. The raw feature of images in MNIST has 14×14 pixel intensities. We selected 10000 samples as training data and 10000 samples as test data in our experiments.

5.1.2. Parameters' Setting. In this article, we should set the parameters, such as α, λ and m in Algorithm 1, λ_1 and λ_2 in Algorithm 2. The code length is set from 4-bit to 64-bit (i.e., 4-bit, 8-bit, 16-bit, 32-bit and 64-bit respectively). We perform fivefold cross validation to find the best parameter pair between α ($\alpha = [0.1, 0.3, 0.5, 0.7]$) and λ ($\lambda = [0.001, 0.1, 10, 100]$), and between λ_1 ($\lambda_1 = [0.001, 0.1, 10, 100]$) and λ_2 ($\lambda_2 = [0.001, 0.1, 10, 100]$) in Algorithm 2. We set $k = 25$ to construct the k nearest neighbor graph.

5.1.3. Evaluation. Given a dataset in each experiment, the whole dataset is mapped into the Hamming space by the learned hash functions. Given a test data points, its binary codes are firstly generated by the learned hash functions, and then compared with the binary codes for all the data points in the dataset. The top k ($k = 25$) data points in the dataset with the smallest Hamming distances to the test data point are retrieved. We then compute standard retrieval performance measured by standard retrieval performance measures: precision, recall, and harmonic mean (i.e., F1-measure).

$$precision = \frac{the\ number\ of\ retrieved\ relevant\ data\ points}{the\ number\ of\ all\ retrieved\ data\ points} \tag{10}$$

$$recall = \frac{the\ number\ of\ retrieved\ relevant\ data\ points}{the\ number\ of\ all\ relevant\ data\ points}. \tag{11}$$

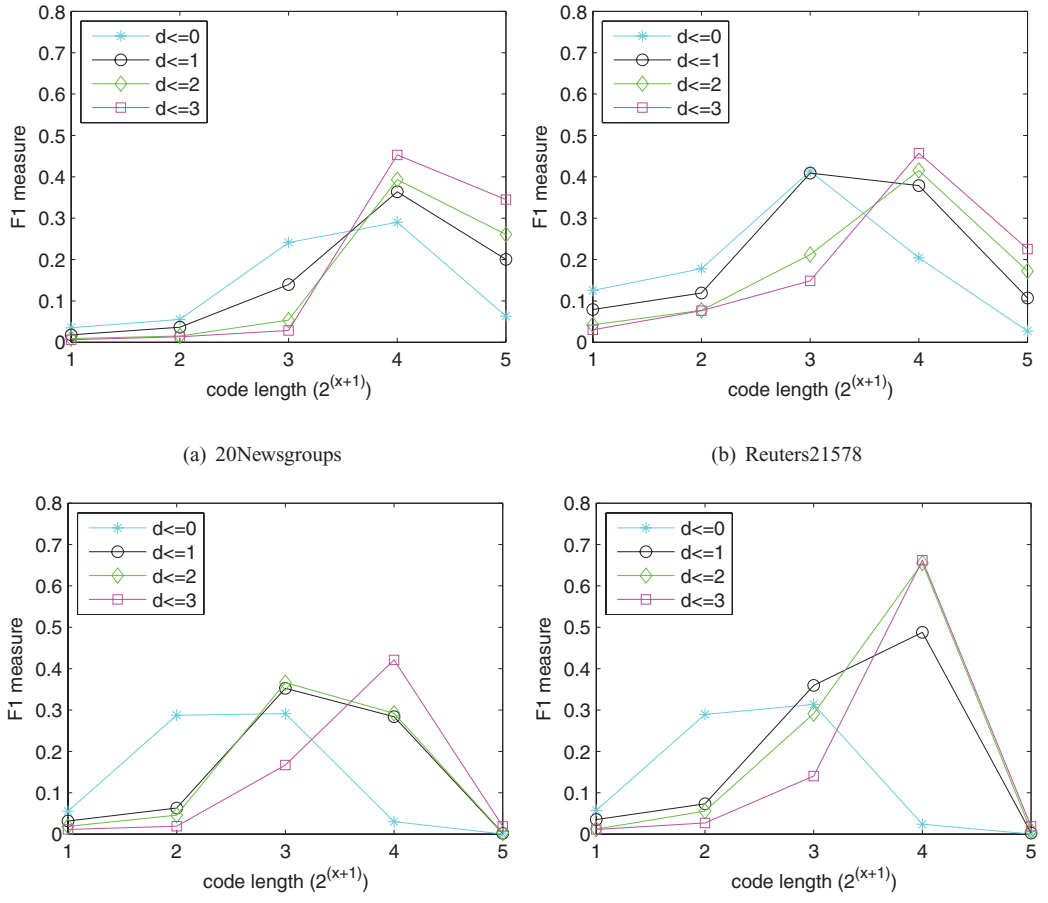
Following the setting in [Zhang et al. 2010b], to determine whether or not a retrieved data point is relevant to the given test data points, we adopt the following two evaluation methodologies:

(1) retrieving original nearest neighbors—the k most similar data points in training data for each test data point. That is, we select 25 of the most similar data points in training data as the ground truth.
(2) retrieving same-topic data points—the k most similar data points in training data with the same category as test data points.

5.2. Results

5.2.1. F1-Measures of SH_ENCW on Different Hamming Distance. We first test the score of F1-measure of SH_ENCW for retrieving the original nearest neighbors and same-topic data respectively. We set the Hamming ball radius (i.e., the maximum Hamming distance between any retrieved data point and test data point) from 0 to 3. The results are presented in Figure 3 and Figure 4.

As can be seen, the score of F1-measure for each curve (i.e., fixed Hamming ball radius with different code length from 4 bits to 64 bits) first increases to its peak value, and then begins to decrease. Before reaching to peak value, the score of F1-measure



(a) 20Newsgroups (b) Reuters21578

(c) MNIST (d) USPS

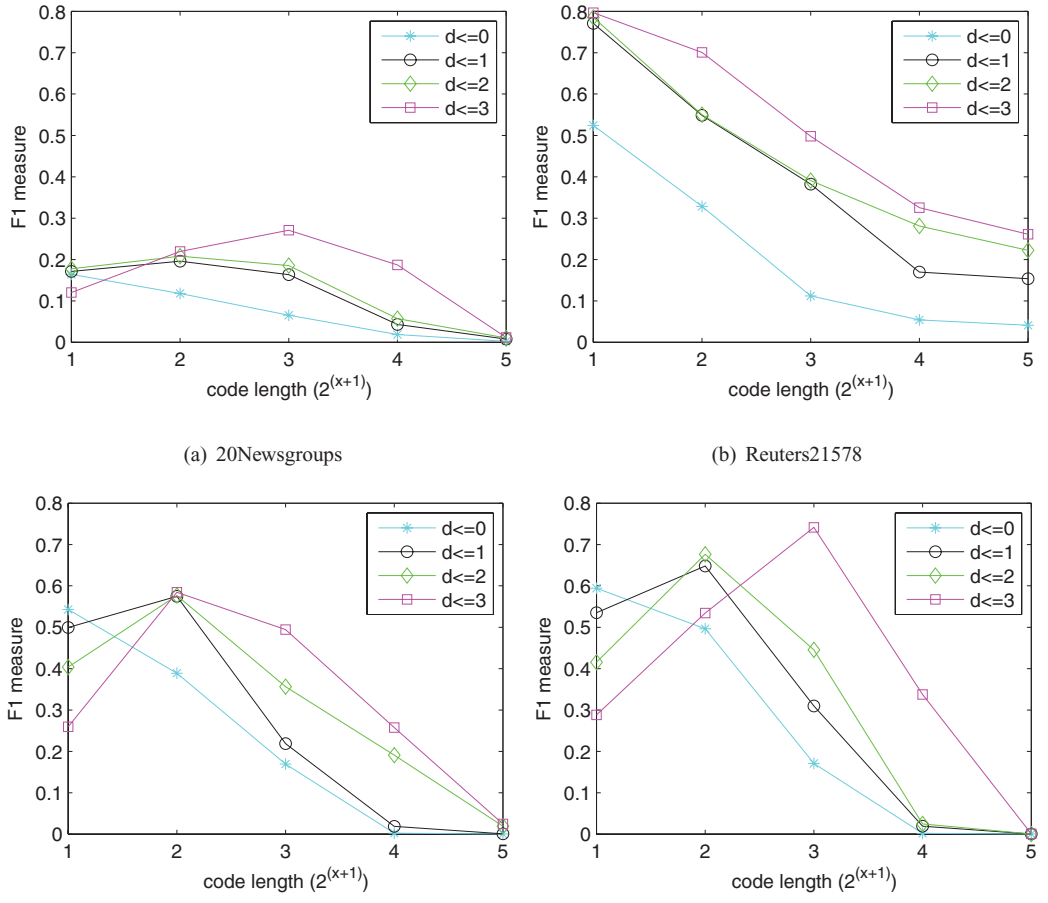
Fig. 3. The score of F_1 measure of SH_ENCW for retrieving original nearest neighbors. Note that, the value of horizontal axis indicates code length. For example, '3' means the code length is $2^{(1+3)} = 16$ bits.

in the curves with the larger Hamming ball radius is smaller than in the ones with a smaller Hamming ball radius. After decreasing from the peak value, the score of F1-measure in the curves with larger Hamming ball radius is bigger than in the ones with a smaller Hamming ball radius.

When the data are represented by shorter binary codes, there are many data points with the same binary codes. That is, their Hamming distance is zero. The top k similar data points to each query are almost all found in the data points with small Hamming distance to the query data point. Such a scenario leads to the result that the score of F1-measure in the curves with smaller Hamming distances is larger than in the curves with larger Hamming distance. After achieving their corresponding peak values, the case is contrary.

Hence, according to the presented experimental results, we know the following.

—When the code length and the Hamming ball radius increase, SH is able to achieve higher scores of F1-measure. However, more computational cost is required. Thus we set Hamming ball radius as 1 in the following experiments.



(a) 20Newsgroups (b) Reuters21578

(c) MNIST (d) USPS

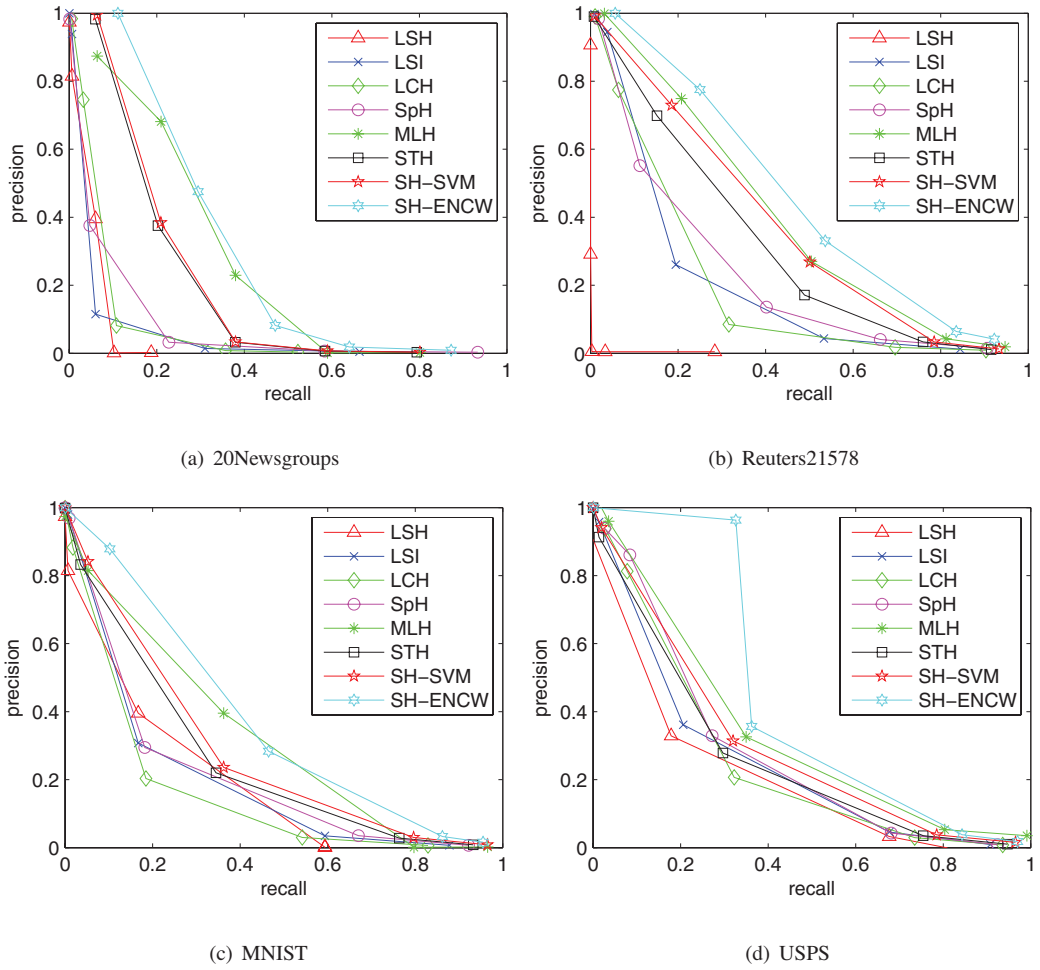
Fig. 4. The score of F_1 measure of SH_ENCW for retrieving same-topic data. Note that, the value of horizontal axis indicates code length. For example, '3' means code length is $2^{(1+3)} = 16$ bits.

—The peak value of each curve shows that we only need to represent original data with reasonably short codes. For example, the best score of F1 measure in Figure 3 and Figure 4 is found in the case with moderate bit length, that is, 16 and 32 bits, respectively.

5.2.2. Precision-Recall Comparison on Different Algorithms. Figure 5 and Figure 6 compare SH with existing hash based methods in terms of their precision-recall curves, for retrieving original nearest neighbors and same-topic data points respectively, while fixing Hamming ball radius as 1.

There are five points in each curve. Each point means a precision-recall pair on a fixed bit length (i.e., from 4 bits to 64 bits). Usually, the top upper point represents the precision-recall pair of the algorithm with 64-bit representation and the lowest point with 4 bits. The higher the place of the point, the more effective the method is.

It is clear that on all data sets, the SH (including SH_ENCW and SH_SVM) outperform STH, LSI, LSH, LCH, and SpH (that has already been shown to outperform LSH



(a) 20Newsgroups (b) Reuters21578

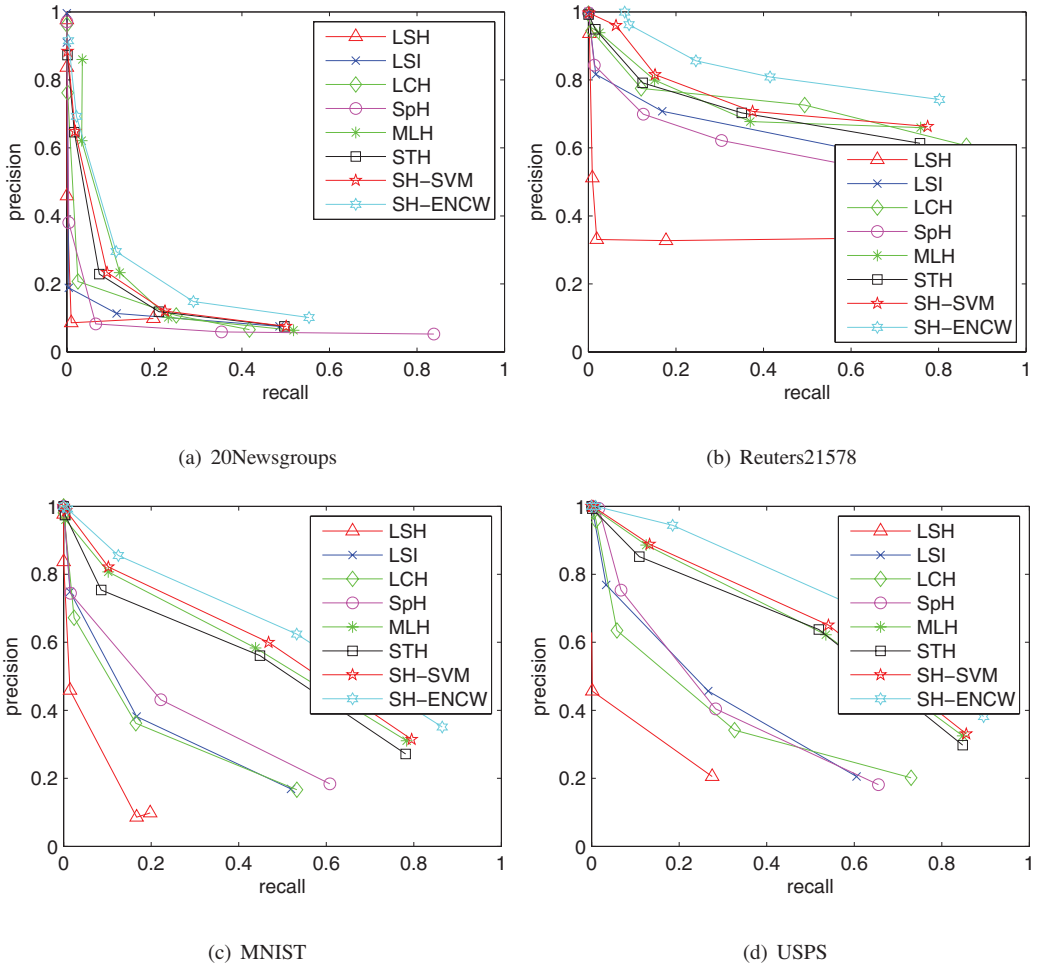
(c) MNIST (d) USPS

Fig. 5. The precision-recall curves of for retrieving original nearest neighbors.

[Charikar 2002; Andoni and Indyk 2008] and stacked-RBM [Torralba et al. 2008; Weiss et al. 2008]). We also found that the SH_SVM is a litter worse than MLH, which is a supervised hashing method. But the SH_ENCW outperforms MLH.

We believe that our SH_ENCW performs better than the comparison algorithms for the following reasons.

—SH_ENCW satisfies the local similarity preservation criterion as well as achieves the minimal reconstructor error, thus loses less information than other algorithms.
—The learned hash functions for encoding test instances avoid the issue of overfitting.
—The proposed framework make the best use of reasonable a-priori knowledge. Although utilizing a-priori knowledge makes the learning process more informative, the learning process can easily induce noise. However, reasonable utilization of a-priori knowledge (e.g., the two kinds of a-priori knowledge used in the article) always improves the effectiveness of learning process.



(a) 20Newsgroups (b) Reuters21578

(c) MNIST (d) USPS

Fig. 6. The precision-recall curves for retrieving same-topic data.

5.2.3. Comparison of SpH, STH, SH_SVM and SH_ENCW. As can be seen from Figure 5 and Figure 6, algorithm SH_SVM outperforms STH method, and SH_ENCW is better than SH_SVM, SpH and STH.

Actually, SH_SVM employs Algorithm 1 (i.e., SH_LARS) in training process, and SVM (same as STH) for learning hash functions. Hence, the experimental results of SH_SVM, STH, and SpH show that the proposed sparse hashing by employing Algorithm 1 is better than the existing hashing methods which only take one constraint (i.e., global similarity for SpH and local similarity for STH) into account. The reason is that the proposed sparse hashing can lose less information and generate more efficient models than the existing techniques, due to adding more constraints into the training process.

Comparing SH_SVM to SH_ENCW, SH_ENCW employs ENCW method (i.e., Algorithm 2) to learn hash functions. The experimental results show that ENCW method is more effective than SVM method. That is because that ENCW method takes into account a-priori knowledge (i.e., implicit group effect within the encoders and the correlations between original data and learned binary codes of training data).

Table I.
Time cost (seconds) for generating 64-bit (or 128-bit) binary codes for the dataset USPS (or GIST) on the comparison algorithms. The illustrated values are the overall time for encoding 2,007 (or 1,000) test data points in dataset USPS (or GIST).

Algorithms	USPS's Training Cost	USPS's Test Cost	GIST's Training Cost	GIST's Test Cost
SH_ENCW	111.363	0.013	155.433	0.006
STH	67.489	1.342	1945.773	7.462
MLH	4578.602	0.023	13451.628	0.007
SpH	1.137	0.099	114.902	0.074
LCH	10.719	0.039	47.864	0.018
LSI	4.556	0.019	28.665	0.008
LSH	**0.524**	**0.011**	**4.975**	**0.004**

Moreover, we also demonstrate that adding such a-priori knowledge is reasonable. However, SVM method does not consider any prior knowledge to learn hash functions.

5.2.4. Encoding Time. SH for approximate similarity search is fast on an ordinary PC with 2.93GHz CPU and 16GB RAM. The time cost of our Matlab implementation of 64-bit on dataset USPS is presented in the second column and the third column of Table I.

As can be seen, SH takes the second most training cost. That is because SH needs to implement LARS algorithm twice (i.e., SH_LARS algorithm and EN method respectively) and CCA method. Fortunately, training process is usually off-line. Moreover, in real applications, training cost of SH can be reduced by some improvements. For example, we can learn initial bases in Algorithm 1 and implement CCA in advance. We can also employ other fast algorithms to solve LARS algorithm, such as online learning method [Mairal et al. 2010].

Due to a single matrix-vector multiplication for encoding unseen data, SH incurs less test cost. Hence, SH method can be utilized in real applications.

5.3. Results on a Large-Scale Dataset

In this section, we use a large dataset ANN-GIST 1M dataset from [Jégou et al. 2011] (referred to as GIST) to evaluate the quality of approximate nearest neighbors search. In GIST, each image is represented by a single global GIST descriptor. GIST provides 1 million images in the base set and 1,000 images in the test set. Since the training process for most compared methods cannot scale to large training datasets, we use a sampling method to generate the training set.

Following the setting in Song et al. [2011], we randomly sampled 10,000 images from the base set as the training set to learn the model. Then the remaining images in the base set and the test images are transferred into binary codes via the learned hash functions. We repeat each algorithm ten times and report the median result in the ten results.

In the experimental setting, the code length is set as $\{16, 32, 48, 64, 80, 96, 112, 128\}$. According to the literature in Jégou et al. [2011], we set $k = 100$ to construct the k nearest neighbor graph. We set the Hamming ball radius as $\{0, 8, 16, 32\}$. The rest settings are the same as the ones in Section 5.1.2.

We first test the score of F1-measure of SH_ENCW for retrieving the original nearest neighbors and present the results in Figure 7. We then compare the proposed SH with the comparison algorithms and present the results in terms of their precision-recall curves for retrieving original nearest neighbors in Figure 8, while fixing Hamming ball radius as 16. Furthermore, we show the impact of different sample size on the evaluation $F1$ measure on all algorithms and report the result in Figure 9. We also

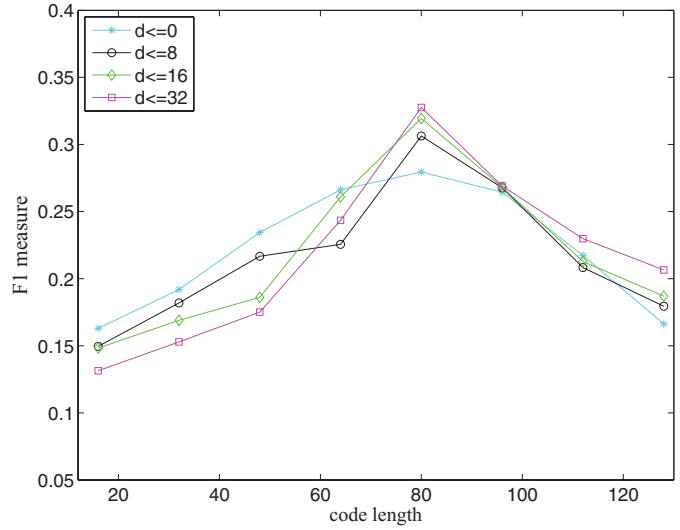


Fig. 7. The score of F_1 measure of SH_ENCW for retrieving original nearest neighbors. Note that the value of horizontal axis indicates code length.

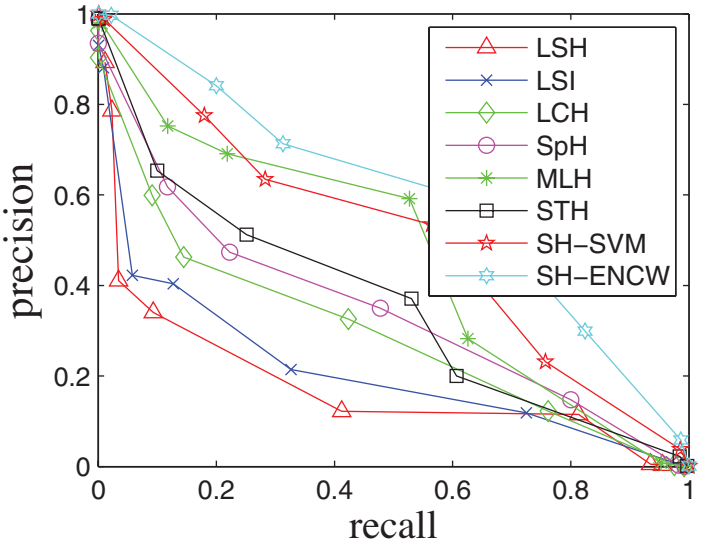


Fig. 8. The precision-recall curves on the large scale dataset GIST.

show the time cost for all comparison algorithms at GIST in the last two columns of Table I.

As can be seen from Figure 7 and Figure 8, the results on the large-scale dataset are similar to the ones on the small datasets, which confirm that the proposed SH methods are also effective for large datasets.

In the last experiment, we also test the effect of sampling size in the training process. We set sample size as [2000, 5000, 10000, 20000, 50000] while fixing Hamming radius as 8 and code length as 80. According to the results in Figure 9, we can see that the $F1$ results increase quickly as the sample size grows from 2000 to 10000, and get marginal

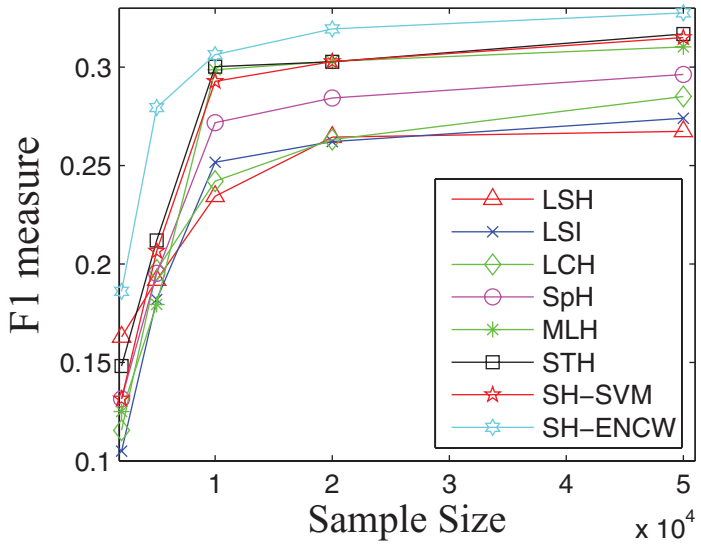


Fig. 9. The score of F_1 measure of all algorithms for different sample size on the large scale dataset GIST.

improvements when the sample size becomes greater than 10000. It is clear that a larger sample set leads to higher accuracy. However, given the high training cost, a reasonably small sample size achieving satisfactory results is preferred. In GIST dataset, the sample size of 10000 seems a good choice, since larger sample sizes do not improve the results significantly.

6. CONCLUSION

The article proposed a novel sparse hashing framework for fast approximate similarity search. The proposed SH framework explores the properties of nonnegative sparse coding to generate interpretably binary codes as well as to achieve minimal reconstruct error. Moreover, SH adds the similarity preservation constraint into sparse coding model for preserving local similarity restructures of the data. Furthermore, SH provides an effective and efficient method for encoding unseen data. The experimental results on five real-world datasets show that SH outperforms state-of-the-art techniques significantly. In future, we will focus on generating approximate hash functions during training process with the aim to make the "out of sample extension" issue redundant. We are also interested in applying the proposed method for the application of near-duplicate detection.

REFERENCES

ANDONI, A. AND INDYK, P. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Comm. ACM 51*, 1, 117–122.

BALUJA, S. AND COVELL, M. 2010. Beyond "near-duplicates": Learning hash codes for efficient similar-image retrieval. In *Proceedings of the 20th International Conference on Pattern Recognition*.

BELKIN, M., NIYOGI, P., AND SINDHWANI, V. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res. 7*, 2399–2434.

BREIMAN, L. AND FRIEDMAN, J. H. 1997. Predicting multivariate responses in multiple linear regression. *J. R. Statist. Soc. Series B 59*, 1, 3–54.

CHARIKAR, M. S. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the ACM Symposium on Theory of Computing*. ACM, 380–388.

CORMEN, T. H., STEIN, C., RIVEST, R. L., AND LEISERSON, C. E. 2001. *Introduction to Algorithms* 2nd Ed. McGraw-Hill Higher Education.

DAI, W., YANG, Q., RONG XUE, G., AND YU, Y. 2008. Self-taught clustering. In *Proceedings of the International Conference on Machine Learning*. 200–207.

DATAR, M., IMMORLICA, N., INDYK, P., AND MIRROKNI, V. S. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Annual Symposium on Computational Geometry*. 253–262.

DRINEAS, P. AND MAHONEY, M. W. 2005. On the nystrm method for approximating a gram matrix for improved kernel-based learning. *J. Mach. Learn. Res. 6*, 2153–2175.

EFRON, B., HASTIE, T., JOHNSTONE, L., AND TIBSHIRANI, R. 2004. Least angle regression. *Ann. Statist. 32*, 407–499.

GAO, S., TSANG, I. W.-H., CHIA, L.-T., AND ZHAO, P. 2010. Local features are not lonely - laplacian sparse coding for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3555–3561.

GHOSH, S. 2011. On the grouped selection and model complexity of the adaptive elastic net. *Statist. Comput. 21*, 3, 451–462.

GRAUMAN, K. 2007. Pyramid match hashing: Sub-linear time indexing over partial correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1–8.

HE, J., CHANG, S.-F., RADHAKRISHNAN, R., AND BAUER, C. 2011. Compact hashing with joint optimization of search accuracy and time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 753–760.

HE, X. AND NIYOGI, P. 2003. Locality preserving projections. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 197–204.

HOTELLING, H. 1936. Relations between two sets of variates. *Biometrika 28*, 3/4, 321–377.

HOYER, P. O. 2004. Non-negative matrix factorization with sparseness constraints. *J. Mach. Learn. Res. 5*, 1457–1469.

HUANG, Z., SHEN, H. T., LIU, J., AND ZHOU, X. 2011. Effective data co-reduction for multimedia similarity search. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1021–1032.

HUANG, Z., SHEN, H. T., SHAO, J., ZHOU, X., AND CUI, B. 2009. Bounded coordinate system indexing for real-time video clip search. *ACM Trans. Inf. Syst. 27*, 3.

JAIN, P., KULIS, B., AND GRAUMAN, K. 2008. Fast image search for learned metrics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1–8.

JÉGOU, H., DOUZE, M., AND SCHMID, C. 2011. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell. 33*, 1, 117–128.

KULIS, B. AND DARRELL, T. 2009. Learning to hash with binary reconstructive embeddings. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 1042–1050.

LEE, H., BATTLE, A., RAINA, R., AND NG, A. Y. 2007. Efficient sparse coding algorithms. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 801–808.

LEE, H., RAINA, R., TEICHMAN, A., AND NG, A. Y. 2009. Exponential family sparse coding with applications to self-taught learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 1113–1119.

LIU, J., HUANG, Z., CAI, H., SHEN, H. T., NGO, C.-W., AND WANG, W. 2013. Near-duplicate video retrieval: Current research and future trends. *ACM Comput Surv*. To appear.

LIU, W., WANG, J., KUMAR, S., AND CHANG, S.-F. 2011. Hashing with graphs. In *Proceedings of the International Conference on Machine Learning*. 1–8.

LYKOU, A. AND WHITTAKER, J. 2010. Sparse cca using a lasso with positivity constraints. *Comput. Stat. Data Anal. 54*, 3144–3157.

MAIRAL, J., BACH, F., PONCE, J., AND SAPIRO, G. 2010. Online learning for matrix factorization and sparse coding. *J. Mach. Learn. Res. 11*, 19–60.

MU, Y., SHEN, J., AND YAN, S. 2010. Weakly-supervised hashing in kernel space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3344–3351.

MUJA, M. AND LOWE, D. 2009. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proceedings of the International Conference on Computer Vision Theory and Applications*. 331–340.

NOROUZI, M. E. AND FLEET, D. J. 2011. Minimal loss hashing for compact binary codes. In *Proceedings of the International Conference on Machine Learning*. 353–360.

OLSHAUSEN, B. A. AND FIELD, D. J. 1996. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature 381*, 6583, 607–609.

RAGINSKY, M. AND LAZEBNIK, S. 2009. Locality-sensitive binary codes from shift-invariant kernels. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 1509–1517.

Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. Y. 2007. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the International Conference on Machine Learning*. 759–766.

Roweis, S. T. and Saul, L. K. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 2323–2326.

Salakhutdinov, R. and Hinton, G. 2009. Semantic hashing. *Int. J. Approximate Reasoning 50*, 969–978.

Shang, L., Yang, L., Wang, F., Chan, K.-P., and Hua, X.-S. 2010. Real-time large scale near-duplicate web video retrieval. In *Proceedings of the ACM International Conference on Multimedia*. 531–540.

Shen, J., Tao, D., and Li, X. 2009. Quc-tree: Integrating query context information for efficient music retrieval. *IEEE Trans. Multimedia 11*, 2, 313–323.

Song, J., Yang, Y., Huang, Z., Shen, H. T., and Hong, R. 2011. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *Proceedings of the ACM International Conference on Multimedia*. 423–432.

Stein, B. 2007. Principles of hash-based text retrieval. In *Proceedings of the ACM Special Interest Group on Information Retrieval*. 527–534.

Tao, Y., Yi, K., Sheng, C., and Kalnis, P. 2009. Quality and efficiency in high dimensional nearest neighbor search. In *Proceedings of the ACM Special Interest Group on Management of Data*. 563–576.

Tibshirani, R. 1994. Regression shrinkage and selection via the lasso. *J. R. Statist. Soc. Series B 58*, 267–288.

Torralba, A., Fergus, R., and Weiss, Y. 2008. Small Codes and Large Image Databases for Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1–8.

Wang, J., Kumar, S., and Chang, S.-F. 2010a. Semi-supervised hashing for scalable image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3424–3431.

Wang, J., Kumar, S., and Chang, S.-F. 2010b. Sequential projection learning for hashing with compact codes. In *Proceedings of the International Conference on Machine Learning*. 1127–1134.

Weiss, Y., Torralba, A., and Fergus, R. 2008. Spectral hashing. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 1753–1760.

Wu, M. and Schölkopf, B. 2007. Transductive classification via local learning regularization. *J. Mach. Learn. Res. 2*, 628–635.

Zass, R. and Shashua, A. 2006. Nonnegative sparse PCA. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 1561–1568.

Zhang, D., Wang, F., and Si, L. 2011. Composite hashing with multiple information sources. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. 225–234.

Zhang, D., Wang, J., Cai, D., and Lu, J. 2010a. Laplacian co-hashing of terms and documents. In *Proceedings of the European Conference on Information Retrieval*. 577–580.

Zhang, D., Wang, J., Cai, D., and Lu, J. 2010b. Self-taught hashing for fast similarity search. In *Proceedings of the ACM Special Interest Group on Information Retrieval*. 18–25.

Zheng, M., Bu, J., Chen, C., Wang, C., Zhang, L., Qiu, G., and Cai, D. 2011. Graph regularized sparse coding for image representation. *IEEE Trans. Image Process. 20*, 5, 1327–1336.

Zou, H. and Hastie, T. 2005. Regularization and variable selection via the elastic net. *J. R. Statist. Soc. Series B 67*, 2, 301–320.

Received March 2012; revised August 2012, December 2012; accepted February 2013