Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 379718, 10 pages
http://dx.doi.org/10.1155/2013/379718

Research Article

Maximum Variance Hashing via Column Generation

Lei Luo,[1] Chao Zhang,[2] Yongrui Qin,[3] and Chunyuan Zhang[1]

[1] *College of Computer, National University of Defense Technology, Changsha, Hunan 410073, China*
[2] *School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China*
[3] *School of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia*

Correspondence should be addressed to Lei Luo; l.luo@nudt.edu.cn

Received 23 January 2013; Accepted 6 March 2013

Academic Editor: Shengyong Chen

With the explosive growth of the data volume in modern applications such as web search and multimedia retrieval, hashing is becoming increasingly important for efficient nearest neighbor (similar item) search. Recently, a number of data-dependent methods have been developed, reflecting the great potential of learning for hashing. Inspired by the classic nonlinear dimensionality reduction algorithm—maximum variance unfolding, we propose a novel unsupervised hashing method, named maximum variance hashing, in this work. The idea is to maximize the total variance of the hash codes while preserving the local structure of the training data. To solve the derived optimization problem, we propose a column generation algorithm, which directly learns the binary-valued hash functions. We then extend it using anchor graphs to reduce the computational cost. Experiments on large-scale image datasets demonstrate that the proposed method outperforms state-of-the-art hashing methods in many cases.

1. Introduction

Nearest neighbor search is a fundamental problem in many applications concerned with information retrieval, including content-based multimedia retrieval [1–3], object and scene recognition [4], and image matching [5]. Due to the exciting advancement of data acquisition techniques, more and more data have been produced in recent years, leading these applications to suffer from the expensive time and storage demand. Recently, hashing has become a popular method to address this issue in terms of storage and speed. These methods convert a high-dimensional data item, for example, an image, into a compact binary code so that more items can be loaded into the main memory and the distance between two items can be computed efficiently by using bit XOR operation of their binary codes, and therefore they have great potential to solve complex problems.

Seminal work of hashing, such as locality-sensitive hashing (LSH) [6], focuses on using random projection to generate random binary codes in the Hamming space. It is then extended to accommodate more distance metrics [7, 8] or kernelized to capture nonlinear relationships in the data space [9, 10]. Without using any training data, LSH and its variances can map close data samples to similar binary codes, and it is theoretically guaranteed that the original metrics are asymptotically preserved in the Hamming space as the code length increases. However, because of the random projection, they need very long codes to achieve good precision in practice.

Data-dependent hashing methods, instead, take advantage of the available data to learn more impact codes for specific tasks, leading to the recent endeavors in hashing. For instance, PCA hashing (PCAH) [11] generates linear hash functions through PCA projections of the training data and is suggested for producing more impact codes rather than random projections. PCAH can be considered as the simplest data-dependent hashing method and can not capture nonlinear similarity information that is available in the training data. Alternatively, spectral hashing (SH) [12] and self-taught hashing (STH) generate hash codes from the low-energy spectrums of data neighborhood graphs to seek nonlinear data representations. The difficulty is how to compute the code of an unseen item, which is known as the problem of *out-of-sample extension*. As a result, SH has to assume that the training data are uniformly distributed in a hyper rectangle, which limits its practicabilities. STH

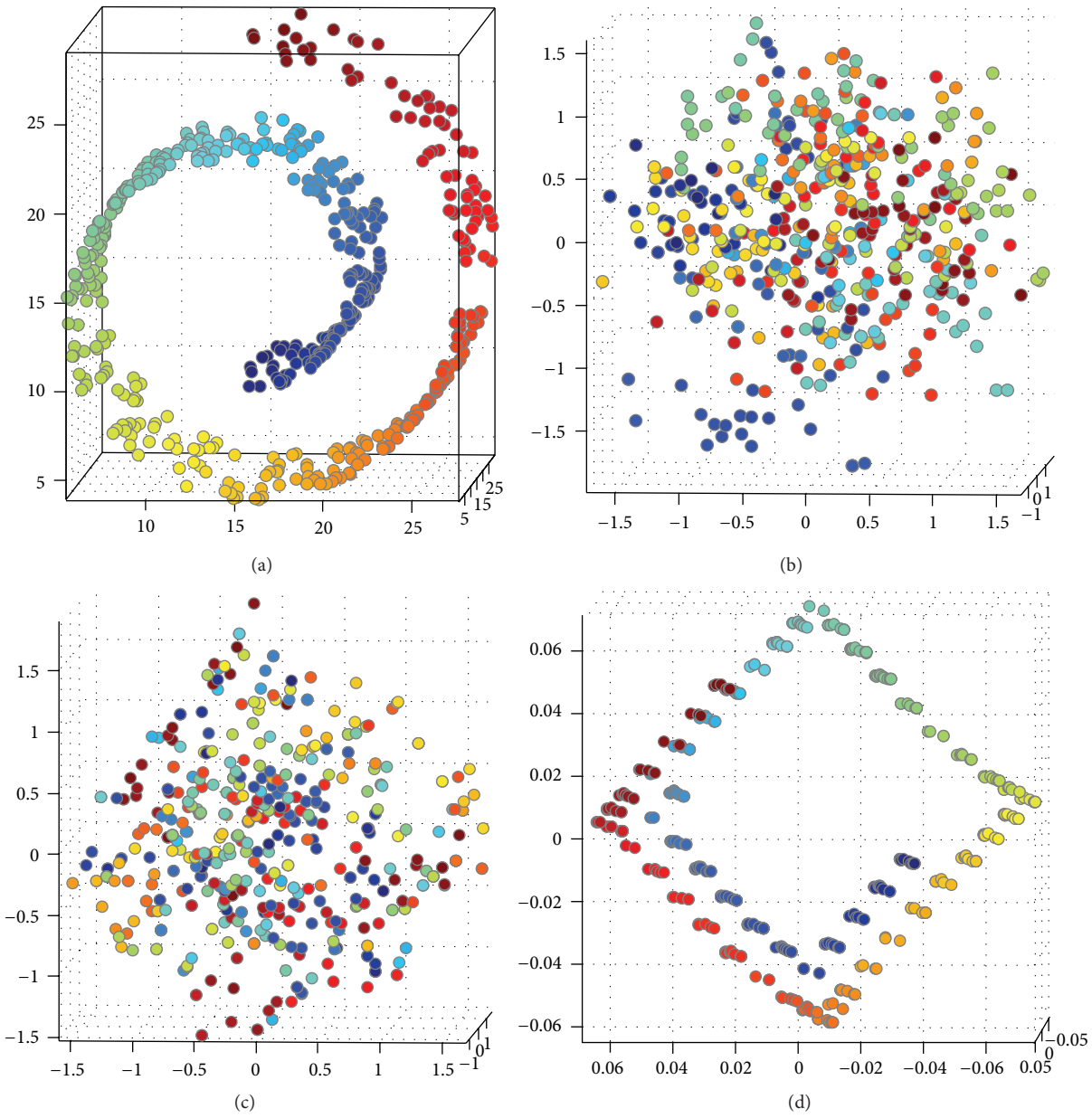

Figure 1: From (a) to (d), a Swiss roll and its hash codes (embedded to 3D by PCA) after applying SH, STH, and MVH-CG, respectively. MVH-CG can maintain the manifold of the Swiss roll in some sense. SH and STH fail to preserve the manifold.

addresses this problem in another way. By viewing the binary codes of the training data as pseudo-labels, it learns the hash functions via an extra pseudo-supervised learning stage. Nevertheless, learning errors in the self-taught stage may collapse the manifold structure of the learning data as illustrated in Figure 1.

Indeed, all these mentioned data-dependent methods aim at hashing the high-dimensional features of the training data with low-dimensional binary codes while preserving the underlying data structure. These methods normally suffer from loss of local geometric structure of the training data. However, by viewing this problem from a different angle and removing the constraint of Hamming space, it can be seen as a variation of the traditional dimensionality reduction problem. Among the large number of dimensionality reduction methods (see [13] for a survey), maximum variance unfolding (MVU) [14] can almost faithfully preserve the local geometric structure of the training data (e.g., the distances and angles between nearby samples in details).

Meanwhile, Liu et al. [15] recently proposed a scalable graph-based hashing method, named anchor graph hashing (AGH). They approximated the origin data by a small set of *anchors* and learned the hash functions by using the Nyström extension [16]. But the generalized eigenfunctions

are derived only for the Laplacian eigenmaps embedding and their performance may decline rapidly when their number increases.

In summary, the main contributions of this work can be described as follows. (i) Inspired by MVU, we propose maximum variance hashing (MVH), which directly embeds the high-dimensional data into a specified low-dimensional Hamming space and preserve the geometric properties of local neighborhoods. The idea is maximizing the total variance of the hash codes subject to the constraints imposed by the rigid rods between k nearest neighbors (kNN). (ii) To address the out-of-example extension difficulty, we propose a column generation-based solution of the derived optimization problem, named MVH-CG. As the size of training data increases, the construction of neighborhood graphs become infeasible. (iii) On the other hand, since the outputs of MVH-CG are a set of binary-valued functions, we can learn the hash functions on the anchor set and then apply them to any unseen data items directly. This motivates us to propose the anchor version of MVH (referred as MVH-A) to reduce the computational cost.

We put forward our algorithms and present the main results on several large-scale image datasets in the next sections.

2. Methodology

2.1. Notation. The following notations will be used throughout this paper:

(i) a bold lower-case letter ($\mathbf{x}$): a column vector;

(ii) a bold upper-case letter ($\mathbf{X}$): a matrix;

(iii) a calligraphic style upper-case letter ($\mathcal{X}$): a set;

(iv) $|\mathcal{X}|$: the number of elements in $\mathcal{X}$;

(v) $h(\cdot)$ or $d(\cdot,\cdot)$: a function with one or two inputs;

(vi) $\mathbb{R}^m$: the m-dimensional real number space;

(vii) (i, j): a pair of order numbers for two data samples.

2.2. Problem Definition. Given a set of samples $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{R}^m\}_{i=1}^n$, we would like to map each point to a low-dimensional binary code for fast nearest neighbor searching. Suppose that the desired number of dimensions of the embedded binary space is ℓ, the goal is to seek a transformation $f(\cdot) : \mathbb{R}^m \to \{0,1\}^\ell$, where the pairwise relationship of $\mathbf{x}_i, \mathbf{x}_j$ in $\mathbb{R}^m$ is kept, in some sense, with their counterpart $\mathbf{y}_i, \mathbf{y}_j$ in $\{0,1\}^\ell$. Here each code $\mathbf{y}_i = [h_1(\mathbf{x}_i); \ldots; h_\ell(\mathbf{x}_i)]$ is an ℓ-dimensional binary vector projected from $\mathbf{x}_i$ using a set of ℓ binary-valued functions $\mathcal{H} = \{h_k(\cdot)\}_{k=1}^\ell$.

Formally, we denote the relationship of $\mathbf{x}_i, \mathbf{x}_j$ as their Euclidean distance (Any other metric can be chosen based on the nature of $\mathcal{X}$, though here we use Euclidean distance as a normal setting.) $d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\|\mathbf{x}_i - \mathbf{x}_j\|^2}$. Meanwhile, the relationship of $\mathbf{y}_i$, $\mathbf{y}_j$ can be defined as their Hamming distance $d_{\text{Hamming}}(\mathbf{y}_i, \mathbf{y}_j)$ naturally. We minimize the following objective to keep the pairwise relationship:

$$\mathcal{O}(h_1, \ldots, h_\ell) = \sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}} \eta_{ij} \left(d(\mathbf{x}_i, \mathbf{x}_j) - C d_{\text{Hamming}}(\mathbf{y}_i, \mathbf{y}_j) \right)^2, \tag{1}$$

where η_{ij}, depending on the specific application, is the weight of how important the relationship of $\mathbf{x}_i$ and $\mathbf{x}_j$ should be kept during the transformation and C is a constant scale factor. Typically, it is reasonable to use the 0-1 adjacency matrix of the training data's kNN graph to define η_{ij} in order to preserve the local structure of $\mathcal{X}$. That is, the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ will be kept if and only if $\mathbf{x}_i$ is a kNN of $\mathbf{x}_j$ or the other way around. The kNN graph has been successfully used in STH to represent the local similarity structure, and the sparse nature of it greatly reduces the computational demand of the next optimization process. In addition, Weinberger and Saul [14] proved that if we add a small number of edges over the kNN graph, both the distances along the edges and the angles between edges in the original graph are preserved. Accordingly, we define η_{ij} as:

$$\eta_{ij} = \begin{cases} 1, & (i, j) \in \mathcal{N}, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

where $(i, j) \in \mathcal{N}$ if and only if $\mathbf{x}_i$ and $\mathbf{x}_j$ are k nearest neighbors themselves or common k nearest neighbors of another sample.

On the other hand, the Hamming distance used in $\{0,1\}^\ell$ may not be as descriptive as their counterpart in $\mathbb{R}^m$, especially when ℓ is small (which is desirable in practice). We therefore relax the discrete Hamming distance to the real weighted Hamming distance $\tilde{d}(\mathbf{y}_i, \mathbf{y}_j) = \sum_{k=1}^\ell w_k |h_k(\mathbf{x}_i) - h_k(\mathbf{x}_j)| = \sum_{k=1}^\ell w_k h_k^{ij}$, where w_k is a nonnegative weight factor associated with $h_k(\cdot)$ and h_k^{ij} is the shorthand symbol of $|h_k(\mathbf{x}_i) - h_k(\mathbf{x}_j)|$. Let $d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$, $\tilde{d}_{ij} = \tilde{d}(\mathbf{y}_i, \mathbf{y}_j)$, and $\xi_{ij} = \sum_{k=1}^\ell w_k h_k^{ij} - d_{ij}$ be the slack between them (we remove the constant scale factor C by merging it into the weight factors); the objective then is

$$\mathcal{O}(h_1, \ldots, h_\ell, w_1, \ldots, w_\ell) = \sum_{(i,j) \in \mathcal{N}} \xi_{ij}^2. \tag{3}$$

As discussed in [14], by preserving the pairwise distances in the extended kNN constraint set $\mathcal{N}$, we faithfully preserve the local manifold of $\mathcal{X}$. Direct optimization of (3), however, tends to crowd points together in one location due to the absence of pairs other than kNNs in the constraint, which is also a problem in the research of manifold learning. Various methods have been proposed in the literature to overcome this problem. t-SNE [17], for instance, uses the long-tailed Student's-t-distribution to make all pairwise information in use. But the constraint set is not sparse any more in t-SNE. Weinberger and Saul [14], instead, proposed to maximize the variance of the embedded codes. That is, the mapping

codes will be pulled apart as far as possible subject to the kNN distance constraints in the embedded space. The maximum variance formulation has two advantages—it is (i) a global formulation and (ii) economical in computation. The variance of $\{\mathbf{y}_i\}_{i=1}^n$ measured by weighted Hamming distance is $\sum_{1\le i,j\le n} \widetilde{d}_{ij} = \sum_{1\le i,j\le n} \sum_{k=1}^{\ell} w_k h_k^{ij} = \sum_{k=1}^{\ell} w_k a_k$. Here, $a_k = \sum_{1\le i,j\le n} h_k^{ij}$ is the variance of the kth bit. It is easy to see that $a_k = (n-s)s$, where $s = \sum_{i=1}^n h_k(\mathbf{x}_i)$ is the count of "1"s $h_k(\cdot)$ produced on $\mathcal{X}$. Combining them together, the optimization object can be written as:

$$\begin{aligned} &\min_{h_1(\cdot),\ldots,h_\ell(\cdot),w_1,\ldots,w_\ell} -\sum_{k=1}^{\ell} w_k a_k + \frac{\gamma}{2}\sum_{(i,j)\in\mathcal{N}} \xi_{ij}^2, \quad w_k \ge 0, \\ &\text{s.t.} \quad \xi_{ij} = \sum_{k=1}^{\ell} w_k h_k^{ij} - d_{ij}, \end{aligned} \tag{4}$$

where γ is the balancing parameter of the two terms. The introduced variable ξ_{ij} here is nontrivial, which will play a critical role in the derivation of (4)'s dual.

2.3. Column Generation Algorithm. There are two sets of variables to be optimized in (4)—the binary-valued hash functions $h_1(\cdot),\ldots,h_\ell(\cdot)$ and their weights $w_1,\ldots,w_\ell$. It is normally difficult to optimize them simultaneously, and the fact that the former one is a group of functions, even adds to the difficulty of solving the problem. We can use the column generation (CG) technique to find an approximate solution of it iteratively. It has been successfully applied in boosting algorithms [18, 19], which also have to generate a series of binary-valued functions and optimize their weights at the same time. Similar to the well-known expectation-maximization (EM) algorithm, column generation has a two-step iteration framework, where one set of variables are treated as constant in each step. The aim of column generation is to reduce the gap between the primal and the dual solutions iteratively.

We first consider $h_k(\cdot)$ in (4) as a known function and ξ_{ij} as a variable to be optimized. Then, the Lagrangian of it is

$$\begin{aligned} &\mathscr{L}\left(w_k, \xi_{ij}, v_k, u_{ij}\right) \\ &\quad = -\sum_{k=1}^{\ell} w_k a_k + \frac{\gamma}{2}\sum_{(i,j)\in\mathcal{X}} \xi_{ij}^2 - \sum_{k=1}^{\ell} v_k w_k \\ &\qquad + \sum_{(i,j)\in\mathcal{X}} u_{ij}\left(\sum_{k=1}^{\ell} w_k h_k^{ij} - d_{ij} - \xi_{ij}\right), \end{aligned} \tag{5}$$

where $v_k \ge 0$ and u_{ij} are Lagrange multipliers. At optimum, the first derivation of the Lagrangian w.r.t. the primal variables must vanish as follows:

$$\begin{aligned} \frac{\partial \mathscr{L}}{\partial w_k} &= -a_k - v_k + \sum_{(i,j)\in\mathcal{N}} u_{ij} h_k^{ij} \\ &= 0 \longrightarrow \sum_{(i,j)\in\mathcal{N}} u_{ij} h_k^{ij} - a_k = v_k \ge 0, \end{aligned} \tag{6a}$$

$$\frac{\partial \mathscr{L}}{\partial \xi_{ij}} = \gamma\xi_{ij} - u_{ij} = 0 \longrightarrow \xi_{ij} = \gamma^{-1} u_{ij}. \tag{6b}$$

The Lagrange dual function is

$$\begin{aligned} \inf_{w_k,\xi_{ij}} \mathscr{L} &= \inf_{w_k,\xi_{ij}} \sum_{k=1}^{\ell} w_k \left(-a_k - v_k + \sum_{(i,j)\in\mathcal{N}} u_{ij} h_k^{ij}\right) \\ &\quad + \sum_{(i,j)\in\mathcal{N}} \left(\frac{\gamma}{2}\xi_{ij}^2 - u_{ij}\xi_{ij}\right) - \sum_{(i,j)\in\mathcal{N}} u_{ij} d_{ji} \\ &= \inf_{\xi_{ij}} \sum_{(i,j)\in\mathcal{N}} \left(\frac{\gamma}{2}\xi_{ij}^2 - u_{ij}\xi_{ij}\right) - \sum_{(i,j)\in\mathcal{N}} u_{ij} d_{ji} \\ &= -\frac{\gamma^{-1}}{2}\sum_{(i,j)\in\mathcal{N}} u_{ij}^2 - \sum_{(i,j)\in\mathcal{N}} u_{ij} d_{ij}. \end{aligned} \tag{7}$$

Then, it is easy to obtain the Lagrange dual as follows:

$$\begin{aligned} &\max_{u_{ij}} \quad -\frac{\gamma^{-1}}{2}\sum_{(i,j)\in\mathcal{N}} u_{ij}^2 - \sum_{(i,j)\in\mathcal{N}} u_{ij} d_{ij} \\ &\text{s.t.} \quad \sum_{(i,j)\in\mathcal{N}} u_{ij} h_k^{ij} \ge a_k. \end{aligned} \tag{8}$$

The idea of CG is to iteratively add a variable by selecting the most violated constraint of the dual, and then optimize the related variables by solving a restricted version of the origin optimization problem. It works on the basis that the sequence of restrict primal problems all have the same dual in which the most violated constraint indicates the steepest ascent direction of the dual. For (8), the subproblem for generating the most violated constraint is as follows:

$$\begin{aligned} h^*(\cdot) &= \arg\min_{h(\cdot)\in\mathcal{B}} \sum_{(i,j)\in\mathcal{N}} u_{ij} h_{h(\cdot)}^{ij} - a_{h(\cdot)} \\ &= \arg\min_{h(\cdot)\in\mathcal{B}} \sum_{(i,j)\in\mathcal{N}} u_{ij} \left|h\left(\mathbf{x}_i\right) - h\left(\mathbf{x}_j\right)\right| \\ &\quad - \sum_{1\le i,j\le n} \left|h\left(\mathbf{x}_i\right) - h\left(\mathbf{x}_j\right)\right|, \end{aligned} \tag{9}$$

where $\mathcal{B}$ is the class of the base binary-valued hash function. Since there can be infinitely many functions in $\mathcal{B}$, we restrict it to be the decision stumps (Since decision stump is a deterministic model, the column generation process will converge when all the constraints in the dual are satisfied, which means that no new hash function can be generated. This convergence, however, usually happens after the required ℓ (typically less than 128) iterations in practice. Moreover, if it is a nondeterministic model here, column generation can produce new hash functions even after satisfying all constraints. We, therefore, do not mention the convergence in Algorithm 1.) [20], a machine learning model widely used in ensemble learning techniques, on the training set $\mathcal{X}$. By the restriction of decision stump, (9) can be solved by exhaustive search within reasonable time.

Input: Training data $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$; balancing parameter $\gamma > 0$; length of hash codes ℓ.
(1) **Initialize**: $\mathbf{w} = \mathbf{0}$; $\mathcal{H} = \emptyset$; assign a positive constant to each element of $\mathbf{u}$.
(2) **for** $t = 1, \ldots, \ell$ **do**
(3) Find a new binary hash function $h_t(\cdot)$ by solving (9); Add $h_t(\cdot)$ to $\mathcal{H}$ and the restricted primal problem (10); Solve (10) by Mosek to obtain the updated $\mathbf{w}$; update $\mathbf{u}$ by (11).
(4) Map $\mathcal{X}$ to $\mathcal{Y} = \{\mathbf{y}_i\}_{i=1}^n$ using $\mathcal{H}$.
Output: The learnt hash functions $\mathcal{H}$, their weights $\mathbf{w}$ and the binary codes $\mathcal{Y}$.

Algorithm 1: MVH-CG: Column generation for maximum variance hashing.

Input: Training data $\mathcal{X} = \{\mathrm{x}_i\}_{i=1}^n$; balancing parameter $\gamma > 0$; length of hash codes ℓ; number of anchors $c \ll n$.
(1) Generate the anchor set $\mathcal{A}$ by a clustering method (e.g. K-means) from $\mathcal{X}$;
(2) Find $\mathcal{H}$ via running Algorithm 1 on $\mathcal{A}$ with parameters γ and ℓ.
(3) Map $\mathcal{X}$ to $\mathcal{Y} = \{\mathbf{y}_i\}_{i=1}^n$ using $\mathcal{H}$.
Output: The learnt hash functions $\mathcal{H}$ and the binary codes $\mathcal{Y}$.

Algorithm 2: MVH-A: Maximum variance hashing with anchors.

We summarize our MVH-CG framework in Algorithm 1. In the tth iteration, we add a new function $h_t(\cdot) = h^*(\cdot)$ to the restricted primal problem. Let $r = |\mathcal{X}|$; we use $\mathbf{w} = [w_k]_{t\times 1}$, $\mathbf{a} = [a_k]_{t\times 1}$, $\mathbf{d} = [d_{ij}]_{r\times 1}$, and $\mathbf{u} = [u_{ij}]_{r\times 1}$ to, respectively, gather the corresponded scalars, and let $\mathbf{H} \in \mathbb{R}^{r\times t}$ to denote the learned hash functions' response on $\mathcal{N}$ such that the kth column of it is the gather of h_k^{ij}, $(i, j) \in \mathcal{N}$. Then, the restricted primal problem (without the introduced variables ξ_{ij}) can be written as:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\mathbf{w}^\top \mathbf{H}^\top \mathbf{H}\mathbf{w} - \left(\gamma^{-1}a^\top + d^\top \mathbf{H}\right)\mathbf{w} \\ \text{s.t.} \quad & \mathbf{w} \succeq 0. \end{aligned} \tag{10}$$

As a quadratic programming problem, (10) can be solved efficiently by the off-the-shelf solver Mosek [21]. The KKT condition (6b) establishes the connection between the primal variables $\mathbf{w}^*$ and the dual variables $\mathbf{u}^*$ at optimality as follows:

$$\mathbf{u}^* = \gamma\left(\mathbf{H}\mathbf{w}^* - \mathbf{d}\right). \tag{11}$$

The outputs of Algorithm 1 are the learnt binary-valued hash functions $\mathcal{H}$, their weights $\mathbf{w}$, and the binary codes $\mathcal{Y}$ of the training set $\mathcal{X}$. Given a new observation $\mathbf{x} \in \mathbb{R}^m$, $\mathcal{H}$ is used to obtain its ℓ-bit binary code as follows:

$$\mathbf{y} = f(\mathbf{x}) = \left[h_1(\mathbf{x}); \ldots; h_\ell(\mathbf{x})\right]. \tag{12}$$

The weight vector $\mathbf{w}$ is a result of the relaxation of the difficult discrete problem. We simply abandon it and use $\mathbf{y}$ only in hash applications.

2.4. Anchor Hashing Using MVH. In MVH-CG, we use the kNN set $\mathcal{N}$ to preserve the manifold structure in $\mathcal{X}$. The sparse nature of kNN matrix can reduce the number of variables in (8). Yet the size of training set $\mathcal{X}$ can be large, for example, the image dataset CIFAR-10 (http://www.cs.toronto.edu/~kriz/cifar.html) has 60,000 images, and the digits recognition dataset MNIST (http://yann.lecun.com/exdb/mnist/) has 70,000 samples. To solve the hashing problem more efficiently, Liu et al. [15] proposed to represent a data point by a set of *anchors*, which are the cluster centers obtained by running K-means on the whole (or a random selected small subsample of the) database. As the number of anchors are sufficiently small, the effective Laplacian eigenvector-based hashing method [12] can be processed on it in linear time. The main difficulty is how to generate hash codes for unseen points, which is known as *out-of-sample extension* problem. For this reason, [15] has to use the Nyström method [16] to learn eigenfunctions for a kernel matrix. Instead, our MVH-CG method learns a set of binary-valued hash functions, which can be directly applied to any data points. As a result, we only need to run the MVH-CG algorithm on the anchor set, and then apply the learnt binary-valued functions to hash the whole dataset. The anchor version of MVH (referred as MVH-A) is summarized in Algorithm 2.

3. Evaluation of the Algorithm

In this section, we evaluate the hashing behavior of MVH-CG and the influence of the parameters.

We first evaluate the hashing behavior of MVH-CG on a Swiss roll toy data. The Swiss roll is a 2D submanifold embedded in a 3D space and can be thought of as curling a piece of rectangular paper. We apply MVH-CG, spectral hashing [12] (SH) and self-taught hashing (STH) [22] on it with the same code length $\ell = 32$, respectively. To visualize the results, we embed the obtained hash codes into $\mathbb{R}^3$ by PCA. The results are illustrated in Figure 1. It is shown that all three methods tend to keep the neighborhood relationship during the mapping. MVH-CG maps the Swiss roll into

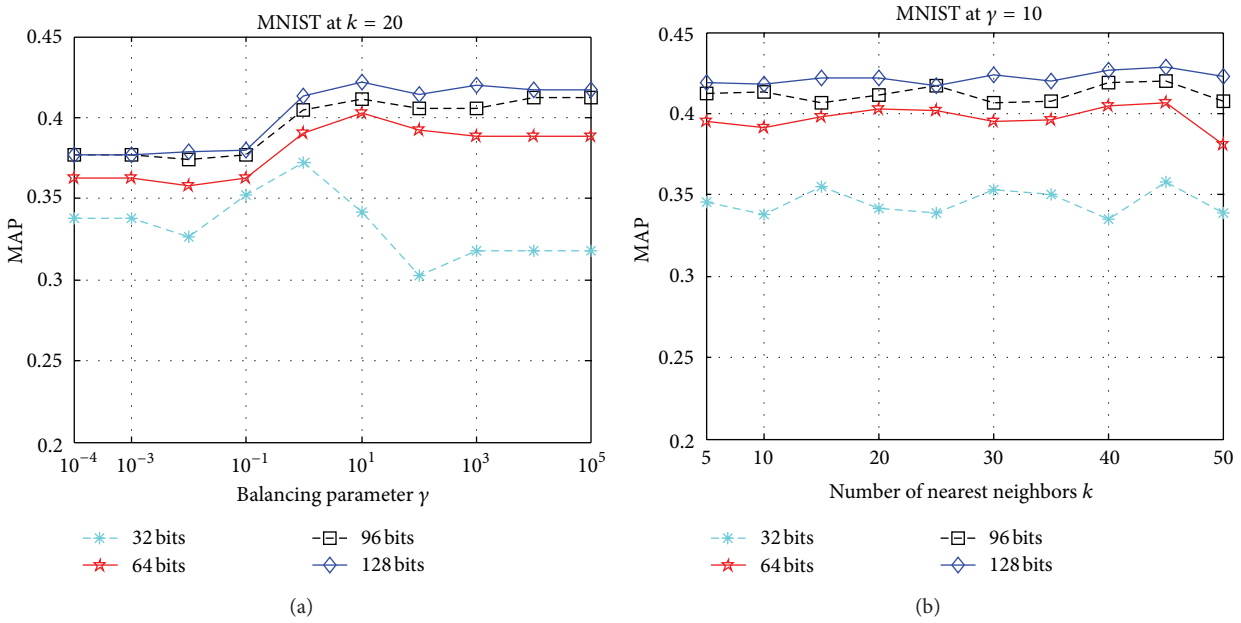

Figure 2: MAP results versus varying balancing parameter γ ((a), fixing $k = 20$) and number of nearest neighbors k ((b), fixing $\gamma = 10$) for MVH-CG. The comparison is conducted on a subset of the MNIST dataset.

a cube and can maintain the submanifold of it in some sense. SH and STH fail in preserving the manifold. For SH, one reason may be that it attempts to keep all pairwise relationships during the mapping. Studies in dimensionality reduction point out that kNN can achieve good approximation of the original manifold, and a method built on kNN kernel (used in MVH-CG) can analyze data that lies on a low-dimensional submanifold more faithfully than a predefined global kernel (used in SH) [23]. For STH, the failure may be due to the learning errors in its self-taught stage.

We then take the MNIST dataset as an example to evaluate the influence of the parameters. The MNIST dataset consists of 70,000 images of handwritten digits divided into 10 classes of 28×28 pixel image. We use the original 784-dimension pixel representation for the MNIST dataset. There are two parameters, k and γ, in MVH-CG. k is the kNN size, and γ is the balancing parameter between the two terms of object function (4). To eliminate the scale difference of these two terms, γ is multiplied with a constant n^2/r in experiments. We randomly generate 4,000 samples of the MNIST dataset, half for training and the rest for test, to evaluate the influences of k and γ. The results are summarized in Figure 2. From (a), we can see that the MAP curves rise after $\gamma = 1$, which indicates that the second term of (4) is somewhat more important; from (b), we see that the performance of MVH-CG does not change significantly with the number of nearest neighbors k. Based on the above observation, we set $\gamma = 10$ and $k = 20$ for the remainder of our experiments.

We also run an experiment to evaluate the influence of anchor set size c of MVH-A based on the MNIST dataset. We randomly select 1000 samples as test set and the others (which are 69,000 samples) for training. As described in Algorithm 2, in this experiment, we first reduce the 69,000 training samples into c anchors by K-means clustering and then run MVH-CG on the anchor set. The resulted MAP curves in Figure 3 basically remain stable from $c = 250$ to $c = 2000$. We, therefore, set $c = 1000$ for MVH-A.

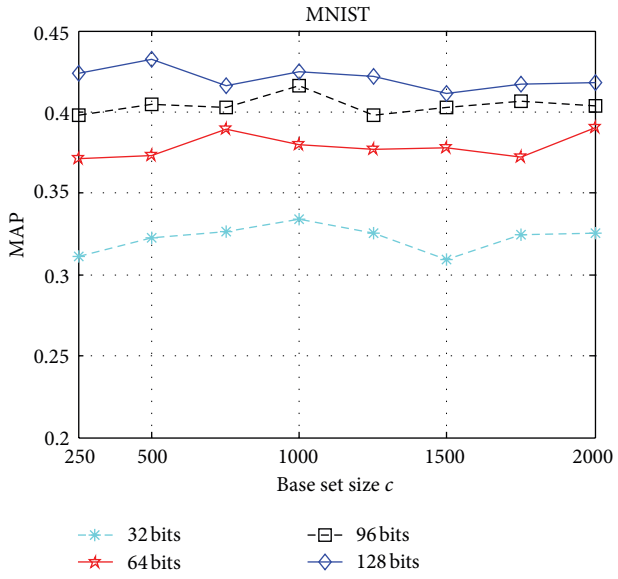

Figure 3: MAP results versus anchor set size c on the MNIST dataset.

4. Experiments

In this section, we evaluate the proposed hashing algorithm on the large-scale image datasets MNIST and CIFAR-10. The MNIST dataset consists of 70,000 images of handwritten

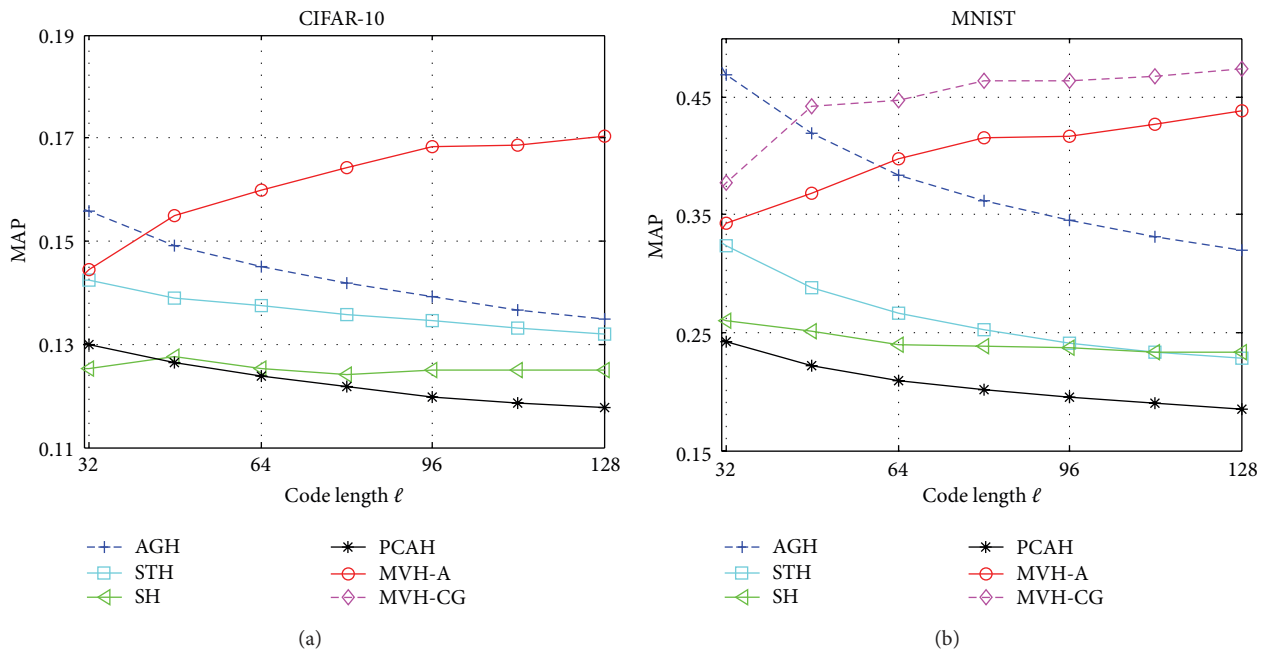

Figure 4: Comparison of different methods using MAP for varying code lengths on CIFAR-10 (a) and MNIST (b).

digits. The CIFAR-10 dataset consists of 60,000 images of 10 classes, which means that there are 6,000 samples for each class. In our experiments, we use the original 784-dimension pixel representation for the MNIST dataset and a 512-dimension GIST [24] feature for the CIFAR-10 dataset. Both of them are split into a test set with 1,000 images and a training set with all other samples. Since the proposed MVH method is fully unsupervised, we compare it with several unsupervised hashing algorithms including PCA-based hashing (PCAH) [11], spectral hashing (SH) [12], self-taught hashing (STH) [22], and anchor graph hashing (AGH) [15]. The performance of the comparison methods is measured by Mean Average Precision (MAP) or precision-recall curves for Hamming ranking.

4.1. Results on the MNIST Dataset. We report the experimental results based on MAP for Hamming ranking with code length from 32 to 128 bits in Figure 4(b). We can see that AGH obtains a high score at very short code length. Its performance, however, declines rapidly as ℓ increases and is inferior to MVH-CG after $\ell = 48$. The performance of PCAH and STH, similar to AGH, also drops down with longer bit lengths. By contrast, MVH-A and MVH-CG consistently improve their performance as code length grows. This property is important in very large-scale problems, when a short hash code, with a length of 32 bits, for example, is not enough to describe the whole dataset. MVH-CG is consistently superior to MVH-A as more data are used in the learning process. Yet, MVH-A also catches up with AGH at $\ell = 64$. We then plot the precision-recall curves for the compared methods in Figure 5. It can be seen that the curves of AGH are relatively high at the beginning, but they drop rapidly when more samples are returned. We also see that our MVH methods perform better at larger ℓ, which confirms the observation in Figure 4. PCAH performs worst in this case since it simply generates the hash hyperplanes by linear projects, which cannot capture the nonlinear similarity information behind the training data. SH is slightly better, but much worse than others, because it relies upon the strict uniform data assumption.

4.2. Results on the CIFAR-10 Dataset. The CIFAR-10 dataset is a manually labeled subset of the well-known 80 million tiny images dataset [4]. It consists of 60,000 images from 10 classes as in the examples shown in the top row of Figure 6. Each image is represented by a 512-dimension GIST [24] feature and then hashed by MVH-A. MVH-CG is not run since decision stump on the whole dataset is expensive. The bottom row of Figure 6 shows the returning list of the example query, where the first 8 results are correct and the last 2 are false positives. The MAP scores against code lengths are plotted in Figure 4(a). On this dataset, we can see that MVH-A yields rising performance as the number of bits increases. It outperforms all its competitors from $\ell = 48$ onward and achieves the highest MAP score at $\ell = 128$. PCAH and SH perform worst again. Figure 7 shows the precision-recall curves of hamming ranking for the compared methods with different code lengths. When $\ell = 32$, MVH-A is inferior to AGH. As the code length grows, the areas under the precision-recall curves of MVH-A are much broader than AGH and other methods. This trend is consistent with the MAP results.

5. Conclusion

This paper has proposed a novel unsupervised hashing method based on manifold learning theories, which can

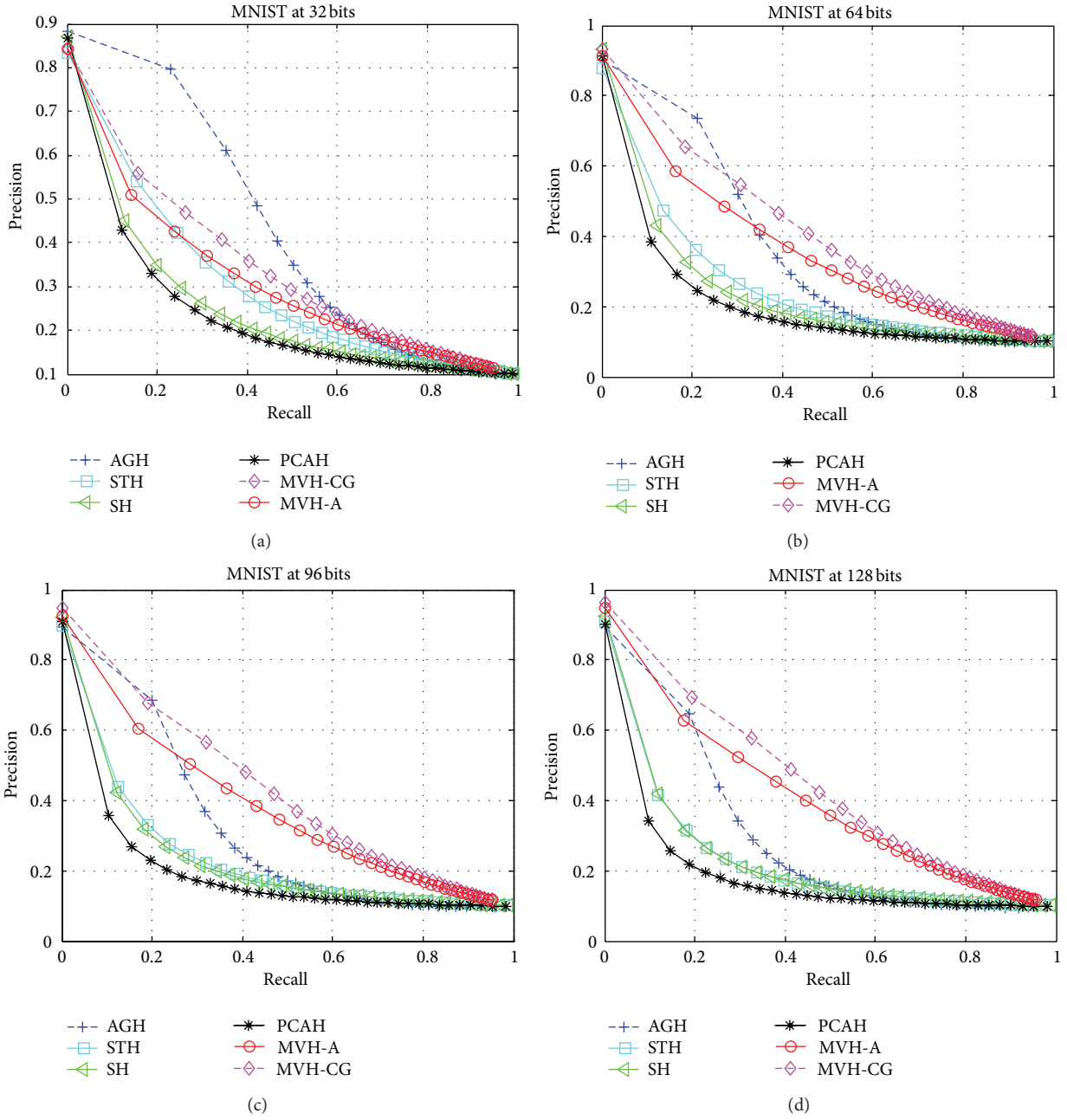

Figure 5: Precision-recall curves for competing methods on the MNIST dataset for different code lengths.

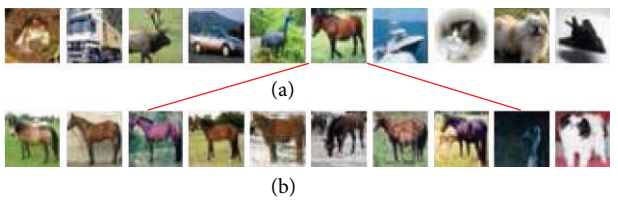

Figure 6: (a) Samples from the CIFAR-10 dataset, one for each category. (b) The results for a query of "horse" image returned by MVH-A with 128 bits. The last two returns are false positive.

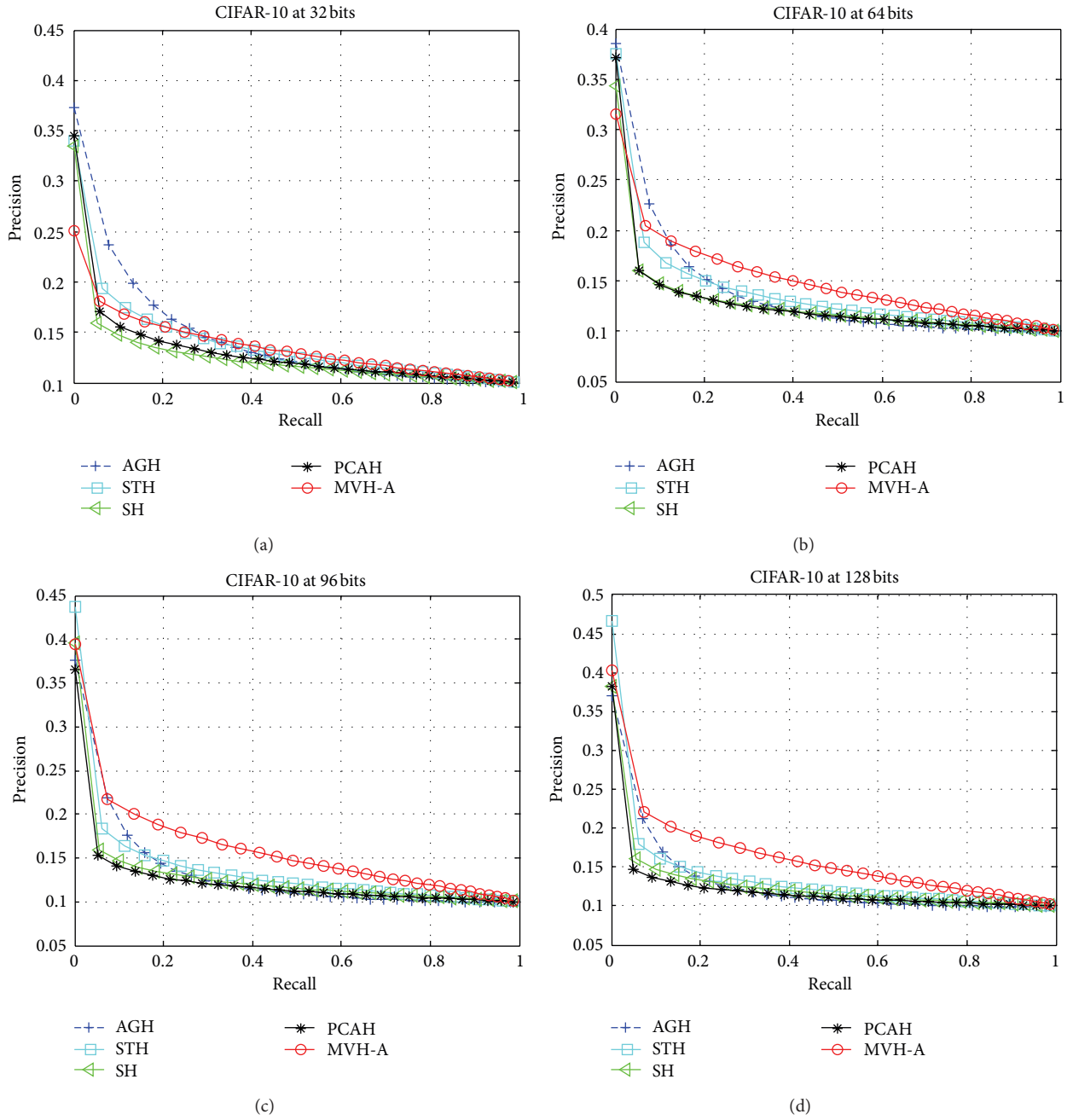

Figure 7: Precision-recall curves for competing methods on the CIFAR-10 dataset for different code lengths.

maximize the total variance of the hash codes while preserving the local structure of the training data. Two algorithms, MVH-CG and MVH-A, have been proposed to solve the derived optimization problem. Both of them can embed the input data into binary space while maintaining the submanifold with very short hash codes. The training process of MVH-A is faster than MVH-CG, but the anchor representation of MVH-A may degrade the retrieval performance of the resulted hash codes. Experimental results on large-scale image datasets show that, in the case of image retrieval, the proposed algorithms are consistently superior to the state-of-the-art unsupervised methods such as PCAH, SH, and STH and outperform AGH with relatively longer codes. The idea of manifold learning has a great potential for large-scale hashing. We are going to develop more efficient hashing method based on other manifold learning approaches.

Acknowledgments

The authors gratefully acknowledge the kind help from the Academic Editor Shengyong Chen. This work was supported by the National Nature Science Foundation of China under

NSFC nos. 61033008, 61272145, 60903041, and 61103080, Research Fund for the Doctoral Program of Higher Education of China under SRFDP no. 20104307110002, Hunan Provincial Innovation Foundation For Postgraduate under no. CX2010B028, Fund of Innovation in Graduate School of NUDT under nos. B100603 and B120605.

References

[1] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain, "Content-based multimedia information retrieval: state of the art and challenges," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 2, no. 1, pp. 1–19, 2006.

[2] S. Chen, J. Zhang, Y. Li, and J. Zhang, "A hierarchical model incorporating segmented regions and pixel descriptors for video background subtraction," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 1, pp. 118–127, 2012.

[3] L. Luo, C. Shen, C. Zhang, and A. van den Hengel, "Shape similarity analysis by self-tuning locally constrained mixeddiffusion," *IEEE Transactions on Multimedia*, 2013.

[4] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: a large data set for nonparametric object and scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1958–1970, 2008.

[5] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua, "LDAHash: improved matching with smaller descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 1, pp. 66–78, 2012.

[6] A. Gionis, P. Indyk, R. Motwani et al., "Similarity search in high dimensions via hashing," in *Proceedings of the International Conference on Very Large Data Bases*, pp. 518–529, 1999.

[7] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the 20th Annual Symposium on Computational Geometry (SCG '04)*, pp. 253–262, June 2004.

[8] B. Kulis, P. Jain, and K. Grauman, "Fast similarity search for learned metrics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2143–2157, 2009.

[9] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 6, pp. 1092–1104, 2012.

[10] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," *Advances in Neural Information Processing Systems*, vol. 22, 2009.

[11] J. Wang, S. Kumar, and S. Chang, "Semi-supervised hashing for large scale search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2393–2406, 2012.

[12] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," *Advances in Neural Information Processing Systems*, 2008.

[13] L. Van der Maaten, E. Postma, and H. Van Den Herik, "Dimensionality reduction: a comparative review," *Journal of Machine Learning Research*, vol. 10, pp. 1–41, 2009.

[14] K. Weinberger and L. Saul, "Unsupervised learning of image manifolds by semidefinite programming," *International Journal of Computer Vision*, vol. 70, no. 1, pp. 77–90, 2006.

[15] W. Liu, J. Wang, S. Kumar, and S. Chang, "Hashing with graphs," in *Proceedings of the International Conference on Machine Learning*, 2011.

[16] Y. Bengio, O. Delalleau, N. Le Roux, J. F. Paiement, P. Vincent, and M. Ouimet, "Learning eigenfunctions links spectral embedding and kernel PCA," *Neural Computation*, vol. 16, no. 10, pp. 2197–2219, 2004.

[17] L. Van Der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2625, 2008.

[18] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor, "Linear programming boosting via column generation," *Machine Learning*, vol. 46, no. 1–3, pp. 225–254, 2002.

[19] C. Shen and H. Li, "On the dual formulation of boosting algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2216–2231, 2010.

[20] W. Iba and P. Langley, "Induction of one-level decision trees," in *Proceedings of the International Conference on Machine Learning*, pp. 233–240, 1992.

[21] Mosek, "The MOSEK interior point optimizer," http://www.mosek.com.

[22] D. Zhang, J. Wang, D. Cai, and J. Lu, "Self-taught hashing for fast similarity search," in *Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*, pp. 18–25, 2010.

[23] L. Saul, K. Weinberger, J. Ham, F. Sha, and D. Lee, "Spectral methods for dimensionality reduction," *Semisupervised Learn*, pp. 293–308, 2006.

[24] A. Oliva and A. Torralba, "Modeling the shape of the scene: a holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.