2013

Towards Efficient Sparse Coding for Scalable Image Annotation

Junshi HUANG

Hairong LIU

Jialie SHEN
Singapore Management University, jlshen@smu.edu.sg

Shuicheng YAN

DOI: https://doi.org/10.1145/2502081.2502127

Towards Efficient Sparse Coding for Scalable Image Annotation

Junshi Huang[1], Hairong Liu[2], Jialie Shen[3], Shuicheng Yan[1]
[1]Department of Electrical and Computer Engineering, National University of Singapore, Singapore
[2]School of Mechanical Engineering, Purdue University, USA
[3]School of Information Systems, Singapore Management University, Singapore
junshi.huang@nus.edu.sg, lhrbss@gmail.com, jlshen@smu.edu.sg, eleyans@nus.edu.sg

ABSTRACT

Nowadays, content-based retrieval methods are still the development trend of the traditional retrieval systems. Image labels, as one of the most popular approaches for the semantic representation of images, can fully capture the representative information of images. To achieve the high performance of retrieval systems, the precise annotation for images becomes inevitable. However, as the massive number of images in the Internet, one cannot annotate all the images without a scalable and flexible (i.e., training-free) annotation method. In this paper, we particularly investigate the problem of accelerating sparse coding based scalable image annotation, whose off-the-shelf solvers are generally inefficient on large-scale dataset. By leveraging the prior that most reconstruction coefficients should be zero, we develop a general and efficient framework to derive an accurate solution to the large-scale sparse coding problem through solving a series of much smaller-scale subproblems. In this framework, an active variable set, which expands and shrinks iteratively, is maintained, with each snapshot of the active variable set corresponding to a subproblem. Meanwhile, the convergence of our proposed framework to global optimum is theoretically provable. To further accelerate the proposed framework, a sub-linear time complexity hashing strategy, e.g. *Locality-Sensitive Hashing*, is seamlessly integrated into our framework. Extensive empirical experiments on NUS-WIDE and IMAGENET datasets demonstrate that the orders-of-magnitude acceleration is achieved by the proposed framework for large-scale image annotation, along with zero/negligible accuracy loss for the cases without/with hashing speed-up, compared to the expensive off-the-shelf solvers.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models; I.2.6 [**Learning**]: Knowledge acquisition

MM'13, October 21–25, 2013, Barcelona, Spain.

http://dx.doi.org/10.1145/2502081.2502127.

Keywords

Hash-accelerated sparsity induced scalable optimization, Sparse coding, Large-scale image annotation

1. INTRODUCTION

With the integration of semantic information in image representation, content-based retrieval approach is still one of the most popular methods for image retrieval. Many efforts [16, 24, 25] have been devoted to improving the performance of retrieval methods. However, many retrieval methods cannot be performed unless the training process is completely operated. This becomes impracticable due to the massive number of images in the Internet. One possible way to improve the retrieval system is to focus on the semantic representation of images, and to this end, the automatic image annotation is thus inevitable.

In this paper, we aims to improve the retrieval system by proposing a scalable and flexible method for image annotation. Traditionally, the image annotation problem has been extensively studied for decades. The existing algorithms can be roughly divided into three categories, namely the classifier-based approaches, the probabilistic modeling-based approaches, and the graph-based approaches. The classifier-based approaches [4, 9, 10] are based on the foundation of discriminative classifiers, e.g. Support Vector Machine (SVM), and perform image annotation by encoding images as bags of localized features or other global features. The probabilistic modeling-based algorithms, e.g. [15, 32], attempt to infer the correlations or joint probabilities between images and annotation labels. Specifically, Feng et al. [15] proposed a multiple Bernoulli relevance model for image annotation. Wang et al. [32] used the probabilistic topic model to predict the labels of images. Besides, nonparametric graph-based algorithms [6, 21, 30], which construct informative graphs to propagate label information, are also widely explored. In particular, Tang et al. [30] proposed a graph-based semi-supervised sparse learning approach to harness both labeled and unlabeled data for image label propagation.

Due to the complex visual contents in images, calculating effective image signatures has become a long-standing challenge for image annotation task. Recently, the sparse coding-based image annotation, proposed by [33], shows that the semantic similarity between two images with overlapped labels can be well recovered in a reconstruction-based way. Chen et al. [7] adopted an exclusive LASSO model with overlapped groups for multi-labels image annotation. The

Area Chair: Roelof van Zwol

success of [7, 33] inspired us to study image annotation problem based on sparse coding in this work. Moreover, motivated by the fact that these works [7, 33] mainly focus on the algorithmic effectiveness, yet are generally quite unsatisfactory in efficiency aspect, we particularly focus on accelerating the sparse optimization for large-scale datasets. Note that most recent sparse optimization methods [13, 19, 23], which are based on gradient descent, are only practical for small-scale or medium-scale sparse optimization problems. When handling large-scale settings, the computational burden of gradient computation becomes unbearable. In such case, even the first order methods might become inefficient, since the gradient with respect to all variables needs to be calculated at every iterative step.

Therefore, the key challenge of efficient optimization for large-scale sparse coding comes from the large number of variables to be optimized. Fortunately, the sparsity itself implies that most of variables should be zero when reaching the optimal solution. By leveraging such a strong prior, we can eliminate these zero variables and simplify the original problem into a much smaller-scale problem if these irrelevant variables can be identified. Though we can theoretically solve a large-scale image annotation problem by transforming it into a smaller one, it is still difficult and time-consuming to detect non-zero variables from a large set of variables corresponding to all the data. To address this issue, an effective and efficient approach is proposed by embedding a sub-linear time complexity method, e.g. locality-sensitive hashing (LSH) [2, 11, 18, 30], for efficiently seeking non-zero variables, with the theoretic convergence guaranteed. As the sparsity property and high efficiency of our solution, our framework is particularly suitable for the problem of large-scale image annotation.

To the best of our knowledge, it is worthy to highlight the main contributions of this work:

- Inspired by the success of sparse coding on image annotation problem, we propose a scalable and efficient framework which drives the optimal solution to the original large-scale image annotation problem by solving a series of simpler and much smaller-scale subproblems.
- In order to efficiently maintain the active variable set, which constructs a small-scale subproblem at each iterative step, the LSH method is seamlessly integrated into our framework to search the nearest neighbors of the intermediate residue vector in sub-linear complexity. The two strategies together bring orders-of-magnitude speed-up in sparse coding optimization, yet with almost negligible accuracy sacrifice in image annotation.
- As the size of the aforementioned active variable set is usually small and controllable, and the solution to each subproblem is usually a good initialization to the successive subproblem, the intermediate small-scale subproblems can be efficiently solved by applying any off-the-shelf sparse optimization methods.

The remainder of this paper is organized as following. Section 2 introduces the sparse coding and the theoretic foundation of LSH. In Section 3, we elaborate the proposed framework in terms of problem formulation and algorithm analysis. Experiments which extensively evaluate the proposed framework for image annotation problem are conducted in Section 4. Finally, conclusions and future works are listed in Section 5.

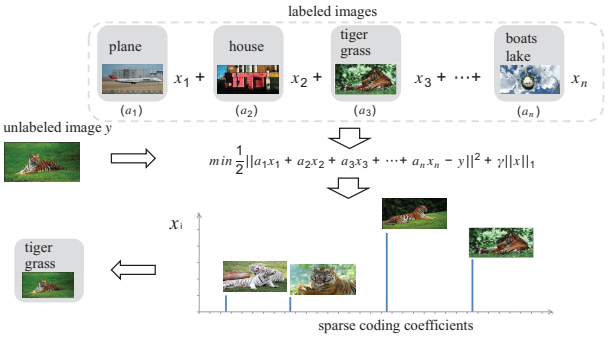

Figure 1: An illustration of sparse coding for scalable image annotation. Given an unlabeled testing image, its feature vector is sparsely reconstructed by a dictionary of labeled training images by solving a sparse coding problem. Then, the label(s) of this testing image are propagated from the labeled training images via the derived sparse reconstruction coefficients.

2. PRELIMINARIES

In this section, we will briefly introduce sparse coding and LSH, which together lay down the technical foundation of our proposed framework for accelerating image annotation.

2.1 Sparse Coding

Given a set of over-complete bases, sparse coding aims to find a succinct representations of stimuli. In the image annotation field, each stimulus, namely, unlabeled testing sample, can be approximately described as a weighted linear combination of a small subset of labeled training samples from the over-complete image set. Those training samples involved in the reconstruction usually well preserve the semantic information of the testing sample [6].

Concretely, given a set of n labeled training images $A = [a_1, \cdots, a_n]$, an unlabeled testing sample y usually can be reconstructed by using a sparse vector of weights $x \in \mathcal{R}^n$, i.e., $y = Ax$, if the set A is over complete. The sparse vector x, which is also known as reconstruction coefficient vector, is assumed to be zero for most of its components. The obtain x, it is generally to solve the following minimization problem

$$\min \| x \|_0 \quad \text{s.t. } y = Ax. \tag{1}$$

Unfortunately, problem (1) requires combinatorial search, and thus is usually intractable. Some greedy algorithms, such as the *matching pursuit* [27] and *orthogonal matching pursuit* [28], are proposed. Another approximate method, called *basis pursuit* [5], suggests using the l_1-norm to replace the l_0-norm, and converting the problem (1) into

$$\min_x \frac{1}{2} \| Ax - y \|_2^2 + \gamma \| x \|_1. \tag{2}$$

Problem (2), which is proved to give the same solution to problem (1) under certain assumptions, is convex and thus easier to be solved [12]. Figure 1 illustrates the idea of how to perform image annotation based on problem (2).

Many recent works [1, 14, 20, 26] focus on efficiently solving the problem (2). However, the proposed algorithms are

based on direct iterative minimization, which is usually computationally expensive, especially when n is large. Therefore, reducing the computational complexity is valuable and essential, especially for large-scale problems.

2.2 Locality-Sensitive Hashing

This subsection briefly introduces LSH, which is a key module of our schema to improve the efficiency of large-scale sparse coding. Generally speaking, LSH is a approximate hashing method which aims to perform probabilistic dimension reduction on high-dimensional space [18]. It can search the approximate nearest neighbors in sub-linear time complexity. Particularly, LSH generates several randomized hash functions to guarantee that the collision probability of two samples is inversely proportional to their "distance", where the specific meaning of "distance" is task dependent. In other words, the larger the "distance" is, the smaller the collision probability is.

Formally, let $d = (\cdot, \cdot)$ be a distance function. For any $p \in S$, where S is the dataset to be indexed, denote $R(p, r)$ as the set of samples from S with distance to p less than r.

DEFINITION 1. *Let h denotes a random choice of hash function from a family H. The family H is called (r_1, r_2, p_1, p_2)-sensitive for $d = (\cdot, \cdot)$ if for any $p, q \in S$*

- *If $p \in R(q, r_1)$, then $Pr[h_H(q) = h_H(p)] \geq p_1$,*
- *If $p \notin R(q, r_2)$, then $Pr[h_H(q) = h_H(p)] \leq p_2$.*

In order to utilize the family of functions, it must satisfy $p_1 > p_2$ and $r_1 < r_2$. For a B-bits LSH, it defines a bucket family $G = \{g : S \to U^b\}$ so that

$$g(p) = (h_1(p), h_2(p), \cdots, h_B(p)),$$

where $h_b \in H$, $1 \leq b \leq B$. Supposing M is the number of hash tables, LSH chooses K groups of functions $g_1, g_2, \cdots, g_K$ from G independently, uniformly and randomly. During the preprocessing stage, all the samples in dataset are mapped into the B hash tables which are indexed by

$$g_k(p) = (h_{k_1}(p), h_{k_2}(p), \cdots, h_{k_B}(p)),$$

with $k = 1, 2, \cdots, K$. To search a query p, LSH only needs to traverse the samples among the union of hash buckets indexed by p exhaustively.

Particularly, Datar et al. [11] proposed a novel LSH schema which works on samples in Euclidean space under l_2-norm without embedding. In this approach, the query can be retrieved in time $O(log(n))$. Furthermore, this hash method works more effectively if the data is extremely high-dimensional but sparse. In this paper, we utilize LSH to retrieve approximate nearest neighbors, so as to improve the overall efficiency of our framework on large-scale datasets.

3. OUR PROPOSED FRAMEWORK

In this section, we will elaborate our proposed framework for the problem of sparse coding based image annotation. First of all, we present some basic concepts and theoretic foundations. Then, a general framework, Sparsity Induced Scalable Optimization (SISO), is proposed to efficiently solve the sparse coding problem. Due to the seamless conversion between the expansion of active variable set and retrieval result of LSH, we integrate LSH into our framework, i.e., the Hash-accelerated SISO, to further improve the efficiency of our approach on large-scale dataset. Finally, the algorithmic convergence and computational complexity of Hash-accelerated SISO are carefully analyzed.

3.1 Problem Formulation and Analysis

Denote a set of n training images $A = [a_1, \cdots, a_n]$ where $a_i \in \mathcal{R}^m$ is the i-th sample of A. The corresponding label set of training samples is represented by matrix $L = [l_1, \cdots, l_n]$, where l_{ij} is set as 1 if a_i is annotated with the j-th label; otherwise 0. Given a testing sample $y \in \mathcal{R}^m$, we assume that y can be approximately reconstructed by A by the following way:

$$y \approx Ax = \sum_{i=1}^{n} a_i x_i, \tag{3}$$

where $x = [x_1, x_2, \cdots, x_n]^T$ is the reconstruction coefficients. Therefore, we can roughly predict the labels l_y of the testing sample y by

$$l_y = \sum_{i=1}^{n} l_i x_i. \tag{4}$$

Note that the number of images with similar label information may be huge on a large-scale image dataset, a "tiger" object in a testing sample, for example, can be semantically reconstructed by various training samples with "tiger" label. However, there are usually discrepancies between the "tiger" object in the training samples and the reconstructed testing sample. Meanwhile, those training samples, which are used to reconstruct the testing sample, may contain many irrelevant labels. For a good performance, we need to find certain training samples, which can not only precisely reconstruct the information of "tiger" object in the testing sample, but also contain as less irrelevant information as possible. On the other hand, as it is often the case that the dimension of features is typically smaller than the sample number in large-scale dataset, the problem (3) may be underdetermined, and the solution becomes ambiguous. Fortunately, the sparse coding, owing to its convexity property, can properly handle those problems, and thus demonstrates good performance in reconstructing label information of the testing sample. Particularly, the image annotation problem is reformulated as a LASSO problem [31] in problem (2).

Although the sparse coding is effective in image annotation problem, its inefficiency greatly discourages its applications in large-scale dataset. Before introducing our approach for speed-up, some definitions are firstly provided as followings.

DEFINITION 2. *For the reconstruction coefficient vector $x \in \mathcal{R}^n$ in problem (1), the indices of all non-zero components of x, $\sigma(x) = \{i | x_i \neq 0, x_i \in x\}$, are called as its support.*

DEFINITION 3. *Let $I = \{1, 2, \cdots, n\}$ denotes the index set for all samples in A, and P_I denotes the problem (2), which contains the whole set of samples as bases. Given any subset $C \subseteq I$, we define the subproblem P_C by instantiating the i-th reconstruction coefficient x_i as zero in problem P_I if $i \in I/C$.*

DEFINITION 4. *Let $|C|$ denotes the cardinality of index set C. All the $|C|$ reconstruction coefficients in subproblem*

P_C, which are presented by a vector x_C, constitute the active variable set of P_C. Particularly, x_i is called an active variable if $x_i \in x_C$.

Obviously, C is the index set of active variable set x_C in subproblem P_C, and denote A_C as the corresponding bases in dictionary A. In order to transform x_C into a corresponding x, we can simply copy the values of x_C into x, and set the rest values of x as zero. On the other hand, we can directly discard the variable x_i if $i \in I/C$ and keep the rest to transform x into x_C.

Let $f(x) = \frac{1}{2}\| Ax - y \|_2^2 + \gamma \| x \|_1$, and according to the properties of convexity, the subproblem P_C inherits the convexity property of P_I:

THEORY 1. *In the problem P_I, if $f(x)$ is a convex function and the value field of x is a convex region, then for any subset $C \subseteq I$, the subproblem P_C is also convex.*

This theorem can be easily proved based on the definition of convexity and we omit the proof here.

THEORY 2. *If x^* is the optimal solution to the problem P_I, and let the index set C contains the support of x^*, i.e., $\sigma(x^*) \subseteq C$, then x_C^* is the solution to the subproblem P_C.*

Formally, Theorem 2 points out that if

$$x^* = \arg\min_x \frac{1}{2}\| Ax - y \|_2^2 + \gamma \| x \|_1, \tag{5}$$

i.e., x^* is the global optimal solution to the problem P_I. Then, for any index set C containing all the non-zero elements of x^*, namely, $\sigma(x^*) \subseteq C$, the following equation is always satisfied:

$$x_C^* = \arg\min_{x_C} \frac{1}{2}\| A_C x_C - y \|_2^2 + \gamma \| x_C \|_1. \tag{6}$$

As aforementioned, in the large-scale image annotation task, we need to solve the sparse coding problem (2). Since $|\sigma(x^*)| \ll n$ in this task, Theorem 2 indicates implicitly that we can solve the problem P_I quickly by solving a proper subproblem. So far, the key challenge lies on the determination of a proper subproblem P_C, or a proper index set C.

3.2 Sparsity Induced Scalable Optimization

Though it is difficult to construct a proper C directly, a proper strategy for solving the problem is to iterative deduction. Most existing sparse optimization methods are based on the gradient descent strategy. Specifically, starting from an initialization x_0, this strategy moves along a path $\{x(0), x(1), \cdots, x(t), x(t+1), \cdots, x^*\}$ with $f(x)$ monotonically decreased. Due to the sparse property of x^*, this process can be greatly accelerated if we can always move along a path with a small index set C. In other words, we can iteratively solve and update the much smaller subproblems P_C, until getting the optimal solution x^*. Based on this idea, we propose the SISO approach, which iterates between two phases, namely, *shrinkage phase* and *expansion phase*.

3.2.1 Shrinkage Phase

In the shrinkage phase, assuming the current active variable set is $x_{C(t)}$, we solve the subproblem $P_{C(t)}$ firstly if $x_{C(t)}$ is not the solution to $P_{C(t)}$. When solving the subproblem $P_{C(t)}$, the initialization is set as $x_{C(t)}$. In the optimization process of solving $P_{C(t)}$, x moves along a segment of path with $|\sigma(x_{C(t)})| \leq |C(t)|$, and $f(x)$ is monotonically decreasing. Obviously, some components in $x_{C(t)}$ may become zero, and thus the active variable set shrinks when setting $C(t) = \sigma(x_{C(t)})$, i.e., all zero components in the active variable set are discarded. Therefore, this phase is called shrinkage phase, whose time complexity is $O(|C(t)|)$.

3.2.2 Expansion Phase

In the expansion phase, $x_{C(t)}$ is already the solution to the subproblem $P_{C(t)}$. Then, we need to check whether $x(t)$, which is transformed from $x_{C(t)}$, is the solution of P_I or not. If yes, we can conclude that $x^* = x(t)$; Otherwise, some variables outside $C(t)$ should be activated and added into $C(t+1)$.

Denote the Lagrangian function of the subproblem P_C as

$$L_C(x_C) = \frac{1}{2}\| A_C x_C - y \|_2^2 + \gamma \| x_C \|_1. \tag{7}$$

Usually, $L_C(x_C)$ is non-differentiable, but has sub-differential $\frac{\partial}{\partial x_C} L_C(x_C)$, which is the set of all sub-gradients of $L_C(x_C)$. According to the generalized KKT condition [17], if x_C^* is the solution to P_C, then

$$0 \in \frac{\partial}{\partial x_i} L_C(x_C^*), \forall i \in C. \tag{8}$$

Formula (8) serves as a criterion to determine whether or not the solution x_C^* to the subproblem P_C can be transformed into the global optimal solution x^* to the problem P_I. If not, Formula (8) can also provide us a tool to guide the expansion of active variable set.

Note that if x^* is the solution to the problem P_I, according to the generalized KKT condition, we also have

$$0 \in \frac{\partial}{\partial x_i} L(x^*), \forall i \in I, \tag{9}$$

where $L(x)$ is the Lagrangian function of P_I.

Given a subproblem $P_{C(t)}$ and its derived solution $x_{C(t)}^*$, we have $\frac{\partial}{\partial x_i} L(x(t)) = \frac{\partial}{\partial x_i} L_{C(t)}(x_{C(t)}^*) = 0$ if $i \in C(t)$, where the vector $x(t)$ is transformed from $x_{C(t)}^*$. Therefore, we only need to consider the partial derivative of $L(x(t))$ with respect to x_i, where $i \notin C(t)$. Let $S = \{i | 0 \notin \frac{\partial}{\partial x_i} L(x(t))\}$ be the expansion set. According to the generalized KKT condition, if S is empty, then $x(t)$ is already the solution to P_I. Otherwise, we only need to add variables according to S into the active variable set $x_{C(t)}$ to generate $x_{C(t+1)}$, i.e., setting $C(t+1) = \sigma(x_{C(t)}^*) \cup S$.

Denote $z(x)$ the sub-gradient of $|x|$, we have

$$z(x) = \frac{\partial}{\partial x}|x| = \begin{cases} 1, & \text{if } x > 0 \\ z, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{cases}, \tag{10}$$

where z is any value within the closed interval [-1,1].

Let $e(x) = \frac{1}{2}\| Ax - y \|_2^2$, then

$$\frac{\partial}{\partial x} L(x) = e'(x) + \gamma * z(x), \tag{11}$$

where $e'(x) = A^T(Ax - y)$, and the generalized KKT condition $0 \in \frac{\partial}{\partial x} L(x)$ can be expressed as

$$e_i{}'(x) = \begin{cases} -\gamma, & \text{if } i \in I, \text{and } x_i > 0 \\ -\gamma * z, & \text{if } i \in I, \text{and } x_i = 0 \\ \gamma, & \text{if } i \in I, \text{and } x_i < 0 \end{cases}, \tag{12}$$

where $e_i{}'(x) = a_i^T(Ax - y)$.

In the same way, if $x^*_{C(t)}$ is the solution to the subproblem $P_{C(t)}$, then $0 \in \frac{\partial}{\partial x^*_{C(t)}} L_{C(t)}(x^*_{C(t)})$, that is,

$$e_i{}'(x^*_{C(t)}) = \begin{cases} -\gamma, & \text{if } i \in C(t), \text{and } x_i > 0 \\ -\gamma * z, & \text{if } i \in C(t), \text{and } x_i = 0 \\ \gamma, & \text{if } i \in C(t), \text{and } x_i < 0 \end{cases}, \quad (13)$$

Since $|z| \leq 1$, we can summarize the index set of the expansion set as

$$S = \{i \mid |e_i{}'(x^*_{C(t)})| \geq \gamma\}. \quad (14)$$

Meanwhile, as

$$e_i{}'(x^*_{C(t)}) = \frac{\partial}{\partial x_i^*} e(x^*_{C(t)}) = a_i^T(A_{C(t)}x^*_{C(t)} - y), \quad (15)$$

where $a_i \in A_{C(t)}$. Thus, we can obtain that

$$S = \{i \mid |a_i^T(A_{C(t)}x^*_{C(t)} - y)| \geq \gamma\}. \quad (16)$$

Here, γ is a natural threshold to select the expansion set, and is empirically set as $\alpha \| A^T y \|_\infty$ in our experiments to make the solution sparse, where $\alpha = 0.002$.

Since $\sigma(x^*_{C(t)}) \subset C(t+1)$ and $0 \notin \frac{\partial}{\partial x_i} L(x(t))$ where $i \in C(t+1)/\sigma(x^*_{C(t)})$, the new subproblem $P_{C(t+1)}$ definitely can be further optimized such that $f(x(t+1)) < f(x(t))$, where the vectors $x(t+1)$ and $x(t)$ are derived from $x^*_{C(t+1)}$ and $x^*_{C(t)}$ respectively. Eventually, the function $f(x)$ will always arrive at the global optimum. If $f(x)$ does not converge, there must exist at least one sample x_i such that $0 \notin \frac{\partial}{\partial x_i} L(x(t))$, which conflicts with the convergent condition. To control the complexity, when $|S|$ is too large, we usually select k variables in S, corresponding to the top-k largest values of $|a_i^T(A_{C(t)}x^*_{C(t)} - y)|$, where k is set as 100-200 in our experiments.

In conclusion, we solve a subproblem containing a much smaller number of variables than original problem in the shrinkage phase. Usually, some active variables in the derived solution become inactive, and thus the active variable set shrinks. In the expansion phase, we expand the active variable set to form a new subproblem for the next shrinkage phase. Therefore, the iteration of these two phases leads to an efficient iterative procedure, and reaches the global optimal solution to the original problem eventually.

3.3 Hash Acceleration for SISO

Obviously, the expansion set casts an important role in our proposed approach. However, its computational complexity is heavily dependent on n, which is the number of bases and is extremely large in large-scale dataset. As aforementioned, with the current residue vector $r_C = y - A_C x_C$, the expansion set can be obtained by

$$\begin{aligned} S &= \{i \mid |a_i^T(A_C x^*_C - y)| \geq \gamma\} \\ &= \{i \mid |a_i^T(-r_C)| \geq \gamma\} \end{aligned}, \quad (17)$$

i.e.,

$$S = \{i \mid a_i^T r_C \geq \gamma \text{ or } a_i^T r_C \leq -\gamma\}. \quad (18)$$

Assuming each basis a_i is l_2-normalized as in general implementation, we can obtain

$$\begin{aligned} S = \{i \mid &-\frac{1}{2}\| a_i - \frac{r_C}{\| r_C \|} \|^2 + 1 \geq \frac{\gamma}{\| r_C \|} \\ &\text{or } \frac{1}{2}\| a_i - \frac{-r_C}{\| r_C \|} \|^2 - 1 \leq -\frac{\gamma}{\| r_C \|}\} \end{aligned}. \quad (19)$$

Let $d(p,q) = \| p - q \|$ be the l_2-distance between vector p and q, we have

$$\begin{aligned} S = \{i \mid &d(a_i, \frac{r_C}{\| r_C \|}) \leq \sqrt{2(1 - \frac{\gamma}{\| r_C \|}} \\ &\text{or } d(a_i, -\frac{r_C}{\| r_C \|}) \leq \sqrt{2(1 - \frac{\gamma}{\| r_C \|}}\} \end{aligned}. \quad (20)$$

To make the formula (20) valid, $\gamma \leq \| r_C \|$ must satisfy. If we use l_2-norm to measure similarity between training samples and the intermediate residue, the value of $\sqrt{2(1 - \frac{\gamma}{\|r_C\|})}$ is the threshold for finding expansion set. For the consideration of efficiency, one does not always want to find the exact similar training samples to the intermediate residue, but use the approximately similar samples instead. Based on this idea, to further enhance the efficiency of SISO, we utilize LSH to retrieve the approximate nearest neighbors of vector $\frac{r_C}{\|r_C\|}$ from A, as well as the approximate nearest neighbors of $-\frac{r_C}{\|r_C\|}$. Compared with Hash-based KNN, the LSH in our framework aims to search the approximate nearest neighbors of normalized residue r_C, instead of reconstructed sample y.

A proper type of hashing function in [11], with $O(log(n))$ computational complexity, is adopted in our experiments. Specifically, this hashing function can directly perform on samples in Euclidean space with l_p-norm, where $p \in (0, 2]$. Obviously, we can directly use the LSH with l_2-norm-based hash function for our implementation.

To be efficient, the size of expansion set should be small, and is usually set as 100 to 200 in our experiments. Overall, as $|C| \ll |I|$ and because of the limited number of iterations before convergence, the time for calculating the solution of subproblems can be ignored. Thus, the main computational complexity of our approach is on the part of searching approximate nearest neighbor.

Formally, let m be the dimension of features. The gradient strategy, which thoroughly goes through each training sample and calculates its sub-differential, requires $O(nm)$ time complexity. Comparatively, the hash-based approach, which can quickly obtain the approximate nearest neighbors by directly mapping the residue vector into similar buckets, is proved of the same time complexity with hashing search, i.e., $O(log(n))$ [11].

In conclusion, our proposed framework, i.e., Hash-accelerated SISO, is summarized in Algorithm 1. Since there is no restriction on the methods to solve the subproblem P_C, any appropriate off-the-shelf sparse optimization method can be applied. Meanwhile, the LSH can also be substituted by many other appropriate approximated nearest neighbors searching methods. Therefore, Algorithm 1 essentially provides a very general and effective framework to accelerate sparse optimization on large-scale problems. Note that in each round of shrinkage and expansion, the overall objective function value shall not be increased, and obviously be lower bounded

Algorithm 1 Hash-accelerated Sparsity Induced Scalable Optimization

Input: The large-scale sparse optimization problem P_I, including the training samples A, the initialization $x(0)$, and the parameter k.
Initialization: $C(0) = \sigma(x(0))$, transform $x(0)$ into $x_{C(0)}$, and normalize each basis in A;
Preprocessing: Create LSH tables and map all samples in A to the buckets;
while $x(t+1)$ is not the solution to P_I **do**
  Solve the sub-problem $P_{C(t)}$ with initialization of $x_{C(t)}$, and update $x_{C(t)}$ as the new solution;
  Set $C(t) = \sigma(x_{C(t)})$; {shrinkage}
  Calculate $r_{C(t)} = y - A_{C(t)}x_{C(t)}$;
  Construct the expansion set S according to the approximate nearest neighbours of $\frac{r_{C(t)}}{\| r_{C(t)} \|}$ and $-\frac{r_{C(t)}}{\| r_{C(t)} \|}$ by LSH;
  if S is empty **then**
    $x_{C(t+1)}$ is the solution to P_I, break;
  else
    Add S into $C(t+1)$ and update $x_{C(t+1)}$; {expansion}
  end if
end while
Output: The solution to P_I.

by zero. Thus, the algorithmic convergence can be naturally guaranteed.

4. EXPERIMENTS

This section presents a set of extensive experiments to evaluate the efficiency and effectiveness of Hash-accelerated SISO on large-scale dataset, including NUS-WIDE [8], its subset NUS-WIDE-LITE, and ImageNet [3]. The experiment results indicate that our proposed framework outperform most of competitors with less running time. Compared with LASSO and SISO, Hash-accelerated SISO proves of significant acceleration with negligible accuracy loss. All experiments are conducted on a PC equipped with Intel X5472 CPU of 3.00 GHz and 32.0 GB RAM.

4.1 Datasets

The first experiment are conducted on a large-scale real-world image dataset NUS-WIDE-LITE, whose images are randomly crawled from some popular image sharing websites, such as Flickr.com. The diversity and complexity of images in this dataset make it a good test-bed for large-scale image annotation problem [6, 8]. Meanwhile, the larger-scale image annotation experiments are conducted on the superset of NUS-WIDE-LITE, known as NUS-WIDE. To further highlight the efficiency of our proposed framework, we also conduct the experiments on ImageNet. For the experiments on ImageNet, we use the provided 1,000-D features (bag of visual words) to represent images and evaluate the efficiency of our framework on about 1.2 million training images and 50,000 testing images with 1,000 labels. For the evaluation of accuracy, we use the flat cost by predicting 5 labels per testing image.

NUS-WIDE-LITE consists of 55,615 images in total, within which 27,808 images are selected as the training set, and the rest of them are used as testing set. Each image is combined with a 81-D label vector to indicate its relationship to 81 distinct concepts. For the convenience, multiple types of local visual features for these images are provided, including 225-D blockwise color moments, 128-D wavelet texture, and 75-D edge direction histogram. To comprehensively demonstrate the performance of algorithms, we use variant percentages of labeled data. The percentage, which is denoted as $\tau \in (0, 1]$ in our experiments, ranges from 10% to 100% with a step of 10%.

NUS-WIDE contains 269,648 images. For each image, an 81-D label vector is maintained. Multiple types of local visual features, including 225-D blockwise color moments, 128-D wavelet texture, and 75-D edge direction histogram, are also provided in this dataset. The training image set consists of 161,789 images, and the rest of them are used for testing. Analogously, the same sampling strategy in the experiment of NUS-WIDE-LITE is adopted on this dataset.

ImageNet contains about 1.2 million training images with 1,000 object labels and one label per image. For each image, a 1,000-D bag of visual words through vector quantization, including vector quantized SIFT features, are provided. In the experiments, we randomly select several different numbers of training images from the training image set, ranging from 200,000 to 1.2 million with a step of 200,000. For different training set, we mainly evaluate the average running time of our framework on 50,000 testing images.

Table 1: The comparison algorithms

Name	Methods
KNN	k-Nearest Neighbors
SVM [10]	Support Vector Machine
RRML	Ridge Regression for Multi-Label Annotation
LNP [34]	Linear Neighborhood Propagation
EGSSC [29]	Entropic Graph Semi-Supervised Classification
LSMP [6]	Large-Scale Multi-label Propagation

4.2 Evaluation Criteria and Algorithms

In the experiments on NUS-WIDE-LITE and NUS-WIDE, we compare our approach with five state-of-the-art algorithms as reported in Table 1. Generally, those algorithms can be divided into three categories, namely classifier-based algorithms,reconstruction-based algorithms, and graph-based algorithms.

Classifier-based Algorithms: In this category, we select the traditional classification algorithm SVM [10] as the representative. In the experiments, SVM is implemented as multi-class classifier with RBF kernel by adopting One-vs-All approach. Particularly, we train several SVMs for each category by using different parameters C and γ, and select the optimal models for evaluation.

Reconstruction-based Algorithms: The main idea of the approaches in this category is to use training dataset as bases, and calculate the reconstruction coefficients for testing data on these bases. Then, the labels of testing data are calculated by multiplying the labels of training data and the reconstruction coefficients. As well-known approaches, KNN and RRML with different parameters are selected in this category. Note that sparse coding, e.g. LASSO, also belongs to this category. In the experiments, the implementation of LASSO is provided by the SLEP [22] package.

Table 2: The running time (unit: hour) of algorithms on NUS-WIDE-LITE dataset

Algorithm	Construction Time[1]	Prediction Time	Total Time	Ratio[2]
SVM	7.27	2.95	10.22	29.20
RRML	0.00	3.43	3.43	9.80
KNN	2.82	0.12	2.94	8.40
LNP	2.69	0.03	2.72	7.77
EGSSC	1.54	0.04	1.58	4.51
LSMP	0.68	0.03	0.72	2.06
Hash-accelerated SISO	0.01	0.34	0.35	1

1. Construction Time is the time for constructing graph or calculating the reconstruction coefficients.
2. $Ratio = \frac{Total\ Time\ of\ One\ Method}{Total\ Time\ of\ Hash\text{-}accelerated\ SISO}$

Graph-based Algorithms: Graph-based algorithms are extensively used for image annotation. In the experiments, three graph-based algorithms are selected as competitors. LNP [34], which is based on a linear-construction criterion, aims to propagate the supervision information by a local propagation and updating process. In our implementation, Local and Global Consistency (LGC) [36] is used as the inference algorithm of LNP. As an entropic graph-regularized semi-supervised classification method, EGSSC [29] is on the foundation of minimizing a Kullback-Leibler divergence over the graph created by k-NN Gaussian similarity. LSMP [6], which is also founded on the Kullback-Leibler divergence, uses hash strategies to accelerate the construction of l_1-graph.

The criteria used for comparing performance on NUS-WIDE-LITE and NUS-WIDE is the mean average precision (MAP) among all the labels. More importantly, the running time for model construction and label prediction among all algorithms are also compared. Particularly, the experiments on computational time and performance of LASSO and SISO are also conducted on NUS-WIDE-LITE and NUS-WIDE to highlight the significant acceleration of our framework. The experiment results indicate that Hash-accelerated SISO outperforms most other competitors in terms of MAP with least running time. Note that some competitors are not completely performed on NUS-WIDE due to the immense running time. For example, we only select 1000 testing samples from NUS-WIDE at random for the evaluation of LASSO.

As the huge size of ImageNet, we only conduct the full-scale experiment on ImageNet with Hash-accelerated SISO. For the LASSO and SISO, we only randomly select 1,000 unlabeled samples for testing. To evaluate the performance, we predict 5 labels for each testing image, and calculate the flat cost. Experiment results show that Hash-accelerated SISO can even achieve orders-of-magnitude acceleration when compared with LASSO and SISO.

Table 3: The running time (unit: hour) of LASSO, SISO and Hash-accelerated SISO on NUS-WIDE-LITE dataset

Algorithm	Total Time	Ratio[1]
LASSO	133.03	380.08
SISO	2.52	7.20
Hash-accelerated SISO	0.35	1

1. $Ratio = \frac{Total\ Time\ of\ One\ Method}{Total\ Time\ of\ Hash\text{-}accelerated\ SISO}$

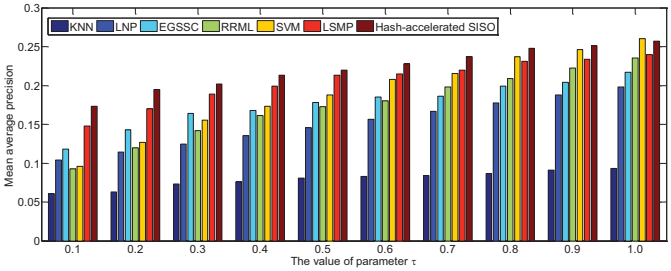

Figure 2: The MAPs of competitors and Hash-accelerated SISO on NUS-WIDE-LITE with various percentages of training dataset.

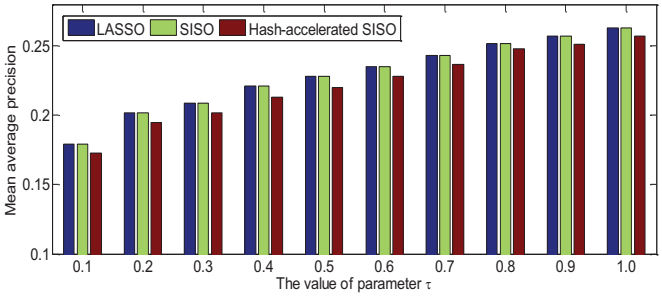

Figure 3: The MAPs of LASSO, SISO and Hash-accelerated SISO on NUS-WIDE-LITE with various percentages training dataset

4.3 Experiments on NUS-WIDE-LITE

In this experiment, we compare Hash-accelerated SISO with six state-of-the-art algorithms as reported in Table 2, including KNN, SVM, RRML, LNP, EGSSC and LSMP. Generally, we use various percentages, ranging from 10% to 100%, of labeled images as training data, and carefully tune the optimal parameters for each algorithm. Specifically 1) The number of nearest neighbors in KNN is set as 500 2) RBF kernel is adopted in SVM, and the searching ranges of its two parameters are $\gamma \in \{0.1, 0.2, ..., 1.0\}$ and $C \in \{0.5, 1, 2\}$. The optimal values of γ and C are set as 0.6 and 1 respectively. 3) The single parameter of RRML, denoted as k, is searched within $\{0.1, 0.2, ..., 1.0, 2.0, ..., 20.0\}$, and the optimal value of k is set as 8 in our experiments. 4) For LNP, LGC is selected as its inference algorithm. The fraction of useful label information that one image receives from its neighbors in LNP is denoted as α, and $\alpha \in \{0.8, 0.9, 0.95, 0.99\}$. The optimal value of α is 0.95. The number of nearest neighbors in LNP is searched among $K \in$

Table 4: The running time (unit: hour) of algorithms on NUS-WIDE dataset.

Algorithm	Construction Time[1]	Prediction Time	Total Time	Ratio[2]
SVM	153.1	15.5	168.6	26.76
KNN	143.6	0.7	144.3	22.90
LNP	56.4	0.9	57.3	9.09
EGSSC	35.4	1.4	36.8	5.84
LSMP	21.6	0.5	22.1	3.51
Hash-accelerated SISO	0.1	6.2	6.3	1

1. Construction Time is the time for constructing graph or calculating the reconstruction coefficients.
2. $Ratio = \frac{Total\ Time\ of\ One\ Method}{Total\ Time\ of\ Hash\text{-}accelerated\ SISO}$

$\{5, 10, 20, ..., 100\}$, and is set as 10. 5) In EGSSC, the values of its three parameters $\mu \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$, $\nu \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$, and $\beta \in \{1, 2, 3\}$ are set as 0.1, 1, and 2 respectively. To be specific, μ and ν are used for weighting the divergence term of Kullback-Leiber and Shannon entropy term respectively; β ensures the convergence of those two similar probability measures. 6) For LSMP, the two parameters, namely, $\mu \in \{1, 2, 5, 10\}$ and $\beta \in \{1, 2, 5, 10\}$, which are used to leverage the balance between terms, are set as 10 and 5 respectively. In our proposed framework, the threshold γ for expansion set is set as $\alpha \| A^T y \|_\infty$, where α is empirically set as 0.002. The maximal number of expansion set, denoted by k, is set as 100 in this experiment. The results of all algorithms are presented in Figure 2 and Table 2 in terms of MAP and running time respectively.

We have the following observations from Figure 2 and Table 2:

- As the number of training samples increases, the overall performances of all methods generally rise. Overall, the MAP of Hash-accelerated SISO outperforms KNN, LNP, EGSSC, RRML, SVM, and LSMP for most settings. Especially when $\tau < 0.6$, Hash-accelerated SISO maintains much higher MAP than LNP, EGSSC, and RRML.

- Table 2 shows that Hash-Accelerated SISO is faster than any other methods on NUS-WIDE-LITE. Particularly, compared with SVM, RRML, and LNP, our framework achieves much higher computational speed-up on NUS-WIDE-LITE dataset by several times. For example, SVM spends about 7.27 hours on training and about 2.95 hours on prediction. However, Hash-accelerated SISO only requires 0.35 hour for prediction with negligible construction time. It is almost orders-of-magnitude saving in terms of time.

- Note that the relatively low MAPs of other LASSO related algorithms, e.g. LSMP, result from the simplification in the LASSO step for efficiency consideration, To be specific, LSMP only considers a small number of nearest neighbors before performing LASSO, which causes the inaccuracy of initialization.

Additionally, we also compare the MAP and running time of LASSO, SISO and Hash-accelerated SISO for image annotation problem on NUS-WIDE-LITE with varying numbers of training samples. The results are reported in Figure 3 and Table 3. It is worthwhile to note that SISO has zero MAP loss, and achieves about 7 times of acceleration when compared with LASSO. The experiment result also shows that Hash-accelerated SISO is the most efficient approach, though LASSO and SISO do not require construction time (i.e., training-free). Compared with LASSO, Hash-accelerated SISO is much (more than 300 times) faster, with only less than 1% loss of MAP.

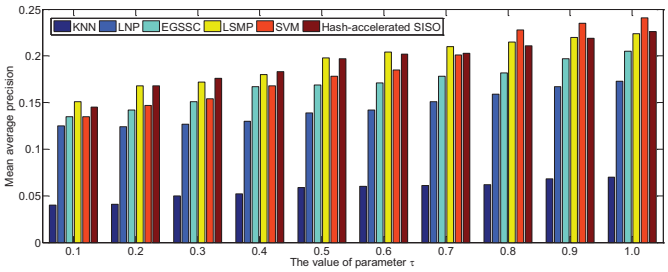

Figure 4: The MAPs of competitors and Hash-accelerated SISO on NUS-WIDE with various percentages training dataset.

4.4 Experiments on NUS-WIDE

In this experiment, we run five image annotation algorithms as competitors which are presented in Table 4, including KNN, SVM, LNP, EGSSC and LSMP. The experiment of RRML is omitted due to the lack of memory. The MAPs of each algorithm with various percentages of training data are presented in Figure 4. Simultaneously, the running time of each algorithm is reported in Table 4.

The parameters of these algorithms are set as following: 1) The number of nearest neighbors in KNN is 1000. 2) For SVM, RBF kernel are also adopted, and its two parameters γ and C are set as 0.8 and 2. 3) For LNP, the optimal fraction of label information is set as 0.99 in this experiment, and the number of nearest neighbors is 20. 4) EGSSC has three parameters, including μ, ν, and β which are set as 0.5, 1, and 1 respectively. 5) Two parameters of LSMP, μ and η, are set as 10 and 8 respectively. In our proposed algorithm, γ is just the same as the experiment on NUS-WIDE-LITE, and the maximal number of expansion set is set as 200 in this experiment.

From Figure 4 and Table 4, we may have the following observations:

- With the increasing number of training images, the MAP of each algorithm varies and monotonically increases. Generally, our proposed algorithm consistently performs better than most other methods. However, the MAP of SVM is higher than Hash-accelerated SISO

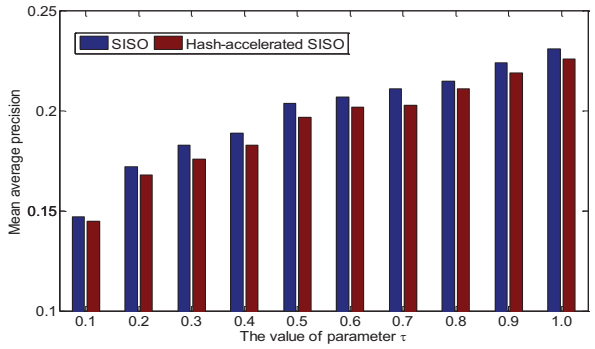

Figure 5: The MAPs of SISO and Hash-accelerated SISO on NUS-WIDE with various percentages training dataset

when $\tau > 0.7$. This situation may be caused by the inaccuracy of approximated nearest neighbors returned by LSH when the number of training samples increases.

- As we can see from Table 4, Hash-accelerated SISO requires nearly negligible time for constructing LSH on 161,789 training images, and spends only 6.2 hours on predicting the labels of 107,859 testing images. Compared with the competitors, our framework achieves significant speed-up by about 4-26 times. Note that our framework spends only 0.14 second on average on predicting the labels of each image when using the whole set of training samples. This is a huge improvement and indicates that Hash-accelerated SISO can provide significant improvement on handling large-scale dataset.

Additionally, we compare the MAP only between SISO and Hash-accelerated SISO on NUS-WIDE due to the unrealistic running time of image annotation by LASSO. The experiment result, which is presented in Figure 5, indicates that Hash-accelerated SISO can achieve comparable MAP when compared with SISO. Meanwhile, we should note that the MAP of SISO is higher than the corresponding MAP of SVM. For example, the MAP of SISO is 23.1% when $\tau = 1.0$; While the corresponding MAP of SVM is 23.0%.

Table 5: The average time (unit: second) of LASSO, SISO and Hash-accelerated SISO on NUS-WIDE dataset

Algorithm	Average Time	Ratio[1]
LASSO	124.05	886.07
SISO	1.78	12.71
Hash-accelerated SISO	0.14	1

1. $Ratio = \frac{Total\ Time\ of\ One\ Method}{Total\ Time\ of\ Hash\text{-}accelerated\ SISO}$

To estimate the prediction speed of LASSO, we run the LASSO on part of NUS-WIDE by randomly selecting 1000 testing samples from testing set, and using the whole training dataset of NUS-WIDE for reconstruction. The result is presented in Table 5 in terms of time cost. Experiment result shows that Hash-accelerated SISO performs much faster than SISO and LASSO. Especially when compared with LASSO, Hash-accelerated SISO significantly accelerates by more than 800 times.

4.5 Experiments on ImageNet

In this part, we only use Hash-accelerated SISO to run the experiment on the whole training dataset with original 1,000-D features. Due to the inefficiency of implementing other competitors, we mainly focus on the efficiency and speed-up achieved by Hash-accelerated SISO. For robustness, we randomly select various numbers of images, ranging from 200,000 to 1.2 million, from the training dataset of ImageNet for training. Meanwhile, 50,000 images are used as testing samples. We evaluate the performance of Hash-accelerated SISO on two aspects, i.e., flat cost and average time.

The parameters of Hash-accelerated SISO are set as following: $k = 200$, $\gamma = \alpha \| A^T y \|_\infty$, where $\alpha = 0.002$. As Table 6 demonstrates, the larger the number of training images is, the better the performance is. Specifically, when the number of training images is larger than 800,000, the performance of Hash-accelerated SISO is better than the baseline in [3], whose flat cost is 0.80. Note that many participants of ImageNet Large Scale Visual Recognition Challenge 2010 [3] achieve better performance than ours, which may owe to the additional features extracted by themselves. Importantly, the running time of our proposed framework increases slowly when the number of training images rises. Even when the number of training images reaches up to 600,000, the average time to predict the label of one image is only 1.24 second.

Table 6: The time (unit: hour) and flat cost of Hash-accelerated SISO on ImageNet dataset

Training Image Number	Total Time	Flat Cost
200,000	3.84	0.869
400,000	10.60	0.835
600,000	15.58	0.808
800,000	17.55	0.791
1,000,000	19.14	0.776
1,200,000	21.07	0.764

For comparison, we again randomly select 100 testing images and predict their labels by applying the LASSO and SISO over the whole training image dataset. Table 7 indicates that the average time of LASSO and SISO for prediction are about 3445.20 seconds and 45.82 seconds respectively. When compared with Hash-accelerated SISO, our proposed framework can significantly accelerate the speed of original LASSO and SISO by more than 2,000 and 30 times, respectively.

Table 7: The average time (unit: second) of LASSO, SISO and Hash-accelerated SISO on ImageNet dataset

Algorithm	Average Time	Ratio[1]
LASSO	3445.20	2266.58
SISO	45.82	30.14
Hash-accelerated SISO	1.52	1

1. $Ratio = \frac{Total\ Time\ of\ One\ Method}{Total\ Time\ of\ Hash\text{-}accelerated\ SISO}$

5. CONCLUSION AND FUTURE WORK

This paper aims to improve the retrieval system via the precise annotations of images. To effectively and efficiently annotate images, this paper turns to accelerate sparse coding optimization in the context of large-scale image annotation. Distinguished from the previous approaches, our proposed solution well utilizes the fact that the reconstruction coefficient vector is sparse in large-scale dataset, and solves the large-scale sparse coding problem by solving a series of much smaller-scale subproblems. The solution includes two phases, i.e., the shrinkage phase and the expansion phase. As the main computational burden is in the expansion phase, to further accelerate the optimization, we utilize the LSH technique to select new active variables more efficiently. Extensive experiments of large-scale image annotation problem show that the proposed solution is superior in both effectiveness and efficiency.

One of the promising future directions is to further evaluate the efficiency of our proposed solution on even larger-scale web multimedia dataset, and also evaluate whether other graph-based algorithms may benefit from the efficiently derived l_1-graph over the complete data by using the proposed solution instead of approximate l_1-graph over KNN as in [30]. It is also noted that LSH can only return the approximate nearest neighbors, thus it is valuable to investigate whether we can further improve the effectiveness of the proposed solution through incorporating other more advanced hashing approaches, e.g. spectral hashing [35].

6. ACKNOWLEDGMENTS

This research is partially supported by the Singapore National Research Foundation under its International Research Centre @Singapore Funding Initiative and administered by the IDM Programme Office.

7. REFERENCES

[1] M. Aharon, M. Elad, and A.M. Bruckstein. K-svd and its non-negative variant for dictionary design. In *SPIE*, 2005.

[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, 2006.

[3] A. Berg, Deng J., and Li F.F. *ImageNet Large Scale Visual Recognition Challenge 2010 (ILSVRC2010)*. Stanford Vision Lab, 2010.

[4] G. Carneiro, A.B. Chan, P.J. Moreno, and N. Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *TPAMI*, 29(3):394–410, 2007.

[5] S.S. Chen, D.L. Donoho, and M.A. Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[6] X. Chen, Y. Mu, S. Yan, and T. Chua. Efficient large-scale image annotation by probabilistic collaborative multi-label propagation. In *MM*, 2010.

[7] X. Chen, X. Yuan, Q. Chen, S. Yan, and T. Chua. Multi-label visual classification with label exclusive context. In *ICCV*, 2011.

[8] T.S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. Nus-wide: A real-world web image database from national university of singapore. In *CIVR*, 2009.

[9] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Large scale transductive svms. *JMLR*, 7:1687–1712, 2006.

[10] C. Cusano, G. Ciocca, and R. Schettini. Image annotation using svm. In *SPIE*. Citeseer, 2004.

[11] M. Datar, N. Immorlica, P. Indyk, and V.S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SoCG*, 2004.

[12] D. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization. In *Proc. Natl. Acad. Sci. USA*, 2003.

[13] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *ICML*, 2008.

[14] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

[15] S. Feng, R. Manmatha, and V. Lavrenko. Multiple bernoulli relevance models for image and video annotation. In *CVPR*, 2004.

[16] J.F. He, J.Y. Feng, X.L. Liu, T. Cheng, T.H. Lin, H. Chung, and S.F. Chang. Mobile product search with bag of hash bits and boundary reranking. In *CVPR*, 2012.

[17] J. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of convex analysis*. Springer Verlag, 2001.

[18] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*, 1998.

[19] K. Koh, S. Kim, and S. Boyd. An interior-point method for large-scale l1-regularized logistic regression. *JMLR*, 8(8):1519–1555, 2007.

[20] H. Lee, A. Battle, R. Raina, and A.Y. Ng. Efficient sparse coding algorithms. In *NIPS*, 2007.

[21] D. Liu, X.S. Hua, L. Yang, M. Wang, and H.J. Zhang. Tag ranking. In *WWW*, 2009.

[22] J. Liu, S. Ji, and J. Ye. *SLEP: Sparse Learning with Efficient Projections*. Arizona State University, 2009.

[23] J. Liu and J. Ye. Efficient euclidean projections in linear time. In *ICML*, 2009.

[24] S. Liu, Z. Song, G.C. Liu, C.S. Xu, H.Q. Lu, and S.C. Yan. Street-to-shop: Cross-scenario clothing retrieval via parts alignment and auxiliary set. In *CVPR*, 2012.

[25] J. Lu, J. Zhou, J. Wang, T. Mei, X. Hua, and S. Li. Image search results refinement via outlier detection using deep contexts. In *CVPR*, 2012.

[26] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative learned dictionaries for local image analysis. In *CVPR*, 2008.

[27] S.G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Tran. Signal Process.*, 41(12):3397–3415, 1993.

[28] Y.C. Pati, R. Rezaiifar, and PS Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Asilomar Conference on Signals, Systems and Computers*, pages 40–44, 1993.

[29] A. Subramanya and J. Bilmes. Entropic graph regularization in non-parametric semi-supervised classification. In *NIPS*, 2009.

[30] J. Tang, S. Yan, R. Hong, G.J. Qi, and T.S. Chua. Inferring semantic concepts from community-contributed images and noisy tags. In *ACM MM*, 2009.

[31] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58:267–288, 1996.

[32] C. Wang, D. Blei, and F.F. Li. Simultaneous image classification and annotation. In *CVPR*, 2009.

[33] C. Wang, S. Yan, L. Zhang, and H. Zhang. Multi-label sparse coding for automatic image annotation. In *CVPR*, 2009.

[34] F. Wang, C. Zhang, H.C. Shen, and J. Wang. Semi-supervised classification using linear neighborhood propagation. In *CVPR*, 2006.

[35] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008.

[36] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, 2004.