Random Projection with Filtering for Nearly Duplicate Search

Yue Lin[*] **Rong Jin**[†] **Deng Cai**[*] **Xiaofei He**[*]
[*]State Key Lab of CAD&CG, College of Computer Science, Zhejiang University, Hangzhou, China
[†]Dept. of Computer Science & Eng., Michigan State University, East Lansing, MI, U.S.A.
linyue29@gmail.com, rongjin@cse.msu.edu, dengcai@cad.zju.edu.cn, xiaofeihe@cad.zju.edu.cn

Abstract

High dimensional nearest neighbor search is a fundamental problem and has found applications in many domains. Although many hashing based approaches have been proposed for approximate nearest neighbor search in high dimensional space, one main drawback is that they often return many false positives that need to be filtered out by a post procedure. We propose a novel method to address this limitation in this paper. The key idea is to introduce a filtering procedure within the search algorithm, based on the compressed sensing theory, that effectively removes the false positive answers. We first obtain a sparse representation for each data point by the landmark based approach, after which we solve the nearly duplicate search that the difference between the query and its nearest neighbors forms a sparse vector living in a small ℓ_p ball, where $p \leq 1$. Our empirical study on real-world datasets demonstrates the effectiveness of the proposed approach compared to the state-of-the-art hashing methods.

Introduction

Nearest neighbor search is a fundamental problem and has found applications in many domains. A number of efficient algorithms, based on pre-built index structures (e.g. KD-tree (Robinson 1981) and R-tree (Arge et al. 2004)), have been proposed for nearest neighbor search. Unfortunately, these approaches perform worse than a linear scan when the dimension of the space is high (Gionis, Indyk, and Motwani 1999). Given the intrinsic difficulty of exact nearest neighbor search, a number of hashing algorithms were proposed for Approximate Nearest Neighbor (ANN) search (Kushilevitz, Ostrovsky, and Rabani 1998; Andoni and Indyk 2008; Eshghi and Rajaram 2008; Arya, Malamatos, and Mount 2009), which aim to preserve the distance between data points by random projection. The most notable hashing method is Locality Sensitive Hashing (LSH) (Gionis, Indyk, and Motwani 1999), which offers a sub-linear time search by hashing similar data points into the same entry of a hash table(s). Besides random projection, several learning based hashing methods, mostly based on machine learning techniques, were proposed recently (Goldstein, Platt, and Burges 2004; Weiss, Torralba, and Fergus 2008; Dong et al. 2008; Salakhutdinov and Hinton 2009; Kulis, Jain, and Grauman 2009; Wang, Kumar, and Chang 2010; B. and Darrell 2010; Yu, Cai, and He 2010; Xu et al. 2011b; Norouzi and Fleet 2011; Wu et al. 2012). Despite the efforts, one common drawback of these hashing methods is that they often return many objects that are far away from the query, requiring a post procedure to remove the irrelevant objects from the return list.

To overcome this shortcoming, we present a novel algorithm, termed *Random Projection with Filtering* (RPF), that effectively removes far away data points by introducing a filtering procedure. Based on the compressed sensing theory (Candès, Romberg, and Tao 2006), we show that most objects in the returned list of our method are within a short distance of the query. Specifically, we first learn a sparse representation for each data point using the landmark based method (Liu, He, and Chang 2010; Chen and Cai 2011; Xu et al. 2011a), after which we solve the nearly duplicate search that the difference between the query and its nearest neighbors forms a sparse vector living in a small ℓ_p ball, where $p \leq 1$. Experimental results on real-world datasets show that the proposed method outperforms the state-of-the-art approaches significantly.

The rest of the paper is organized as follows: Section 2 reviews related work on nearest neighbor search; Section 3 presents the proposed algorithm and the related analysis; Section 4 presents the empirical study, and Section 5 concludes this work.

Related Work

Many data structures were proposed for efficient nearest neighbor search in low dimensional spaces. A well known algorithm is KD-tree (Robinson 1981) and its variants (Silpa-Anan and Hartley 2008). The other popular data structures for spatial search are M-tree (Ciaccia, Patella, and Zezula 1997) and R-tree (Arge et al. 2004). All these methods, though work well in low dimensional space, fail as dimension increases (Gionis, Indyk, and Motwani 1999).

A number of hashing algorithms have been developed for approximate nearest neighbor search. The most notable ones are Locality Sensitive Hashing (LSH) (Gionis, Indyk, and Motwani 1999) and its variants (Charikar 2002; Datar et al. 2004; Lv et al. 2007; Eshghi and Rajaram 2008;

Kulis and Grauman 2009; Raginsky and Lazebnik 2009). These algorithms embed high dimensional vectors into a low dimensional space by random projection. In LSH, several hash tables are independently constructed in order to enlarge the probability that similar data points are mapped to the same bucket. However, due to the data independence, LSH always suffers from the severe redundancy of the hash codes as well as the redundancy of the hash tables (Xu et al. 2011b). It requires many hash tables to achieve a reasonable recall and should make the trade off between efficiency and accuracy. In addition to random projection, many learning based hashing methods were proposed, including spectral hashing (Weiss, Torralba, and Fergus 2008), semantic hashing (Salakhutdinov and Hinton 2009), laplacian co-hashing (Zhang et al. 2010a), self-taught hashing (Zhang et al. 2010b), minimal loss hashing (Norouzi and Fleet 2011) and semi-supervised hashing (Wang, Kumar, and Chang 2010). Most of them performed the spectral analysis of the data (Cai, He, and Han 2005; 2007b; 2007a; Cai et al. 2007) and achieved encouraging performances in the empirical studies. However, most of the learning based hashing algorithms only employ a single hash table. When higher recall is desired, they usually have to retrieve exponentially growing number of hash buckets containing the query, which may significantly drag down the precision (Xu et al. 2011b).

Our method is based on the compressed sensing theory (Donoho 2006; Candès, Romberg, and Tao 2006), which has been successfully applied to multiple domains, such as error control coding (Candes and Tao 2005), message-passing (Donoho, Maleki, and Montanari 2009), and signal reconstruction (Schniter 2010).

Algorithm

Our proposed algorithm contains two stages. In the first stage, we use the landmark based approach (Liu, He, and Chang 2010; Chen and Cai 2011) to obtain a sparse representation for each data point. In the second stage, we introduce a filtering procedure within the search algorithm based on the compressed sensing theory, that effectively removes the false positive answers. In order to reduce the storage space and speed up the query process, we extend the proposed algorithm to the binary codes finally.

Landmark Based Sparse Representation

The first stage of the proposed algorithm is to obtain the sparse representations for all data points, that can well preserve the structure of the data in the original space. Landmark based sparse representation (Liu, He, and Chang 2010; Chen and Cai 2011) is proved to be an effective way to discover the geometric structure of the data. The basic idea is to use a small set of points called *landmarks* to approximate each data point which leads a sparse representation of the data.

Suppose we have N data points $V = [\mathbf{v}_1, \cdots, \mathbf{v}_N], \mathbf{v}_i \in \mathbb{R}^D$ and let $\mathbf{a}_j \in \mathbb{R}^D, j \in [m]$ be the landmarks that sampled from the dataset. To represent the data points in the space spanned by the landmarks, we essentially find two matrices $A \in \mathbb{R}^{D\times m}$ and $X \in \mathbb{R}^{m\times N}$, whose product can approximate V,

$$V \approx AX.$$

where j-th column of A is the landmark $\mathbf{a}_j$ and each column of X is a m-dimensional representation of $\mathbf{v}$.

For any data point $\mathbf{v}_i \in \mathbb{R}^D$, its approximation $\hat{\mathbf{v}}_i$ can be written as

$$\hat{\mathbf{v}}_i = \sum_{j=1}^{m} x_{ji}\mathbf{a}_j \tag{1}$$

A natural assumption here is that x_{ji} should be larger if $\mathbf{v}_i$ is closer to $\mathbf{a}_j$. We can emphasize this assumption by setting the x_{ji} to zero if $\mathbf{a}_j$ is not among the $t(\leq m)$ nearest neighbors of $\mathbf{v}_i$. This restriction naturally leads to a sparse representation for $\mathbf{v}_i$ (Liu, He, and Chang 2010; Chen and Cai 2011). Let $E_{\langle i\rangle}$ denote the index set which consists t indexes of t nearest landmarks of $\mathbf{v}_i$, we compute x_{ji} as

$$x_{ji} = \frac{K(\mathbf{v}_i, \mathbf{a}_j)}{\sum_{j'\in E_i} K(\mathbf{v}_i, \mathbf{a}_{j'})}, j \in E_{\langle i\rangle}, \tag{2}$$

$x_{ji} = 0$ if $j \notin E_{\langle i\rangle}$, where $K(\centerdot)$ is a kernel function. In this paper, we use the Gaussian kernel.

$$K(\mathbf{v}_i, \mathbf{a}_j) = e^{-\frac{\|\mathbf{v}_i - \mathbf{a}_j\|}{2h^2}} \tag{3}$$

The sparse representations of data points enable us to use the Restricted Isometry Property (Donoho 2006), which is a key in the next stage.

Random Projection with Filtering

Let $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be the collection of N objects to search, where each object is represented by a sparse vector $\mathbf{x}_i \in \mathbb{R}^m$ as we discussed in the last section. We assume both the dimension m and the number of objects N are large.

Given a query vector $\mathbf{q} \in \mathbb{R}^m$, an object $\mathbf{x}$ is a (R, S)-nearly duplicate of $\mathbf{q}$ if (i) $|\mathbf{x} - \mathbf{q}|_2 \leq R$ (i.e., $\mathbf{x}$ is within distance R of $\mathbf{q}$), and (ii) $|\mathbf{x} - \mathbf{q}|_0 \leq S$ (i.e., $\mathbf{x}$ differs from $\mathbf{q}$ by no more than S attributes). It is the second requirement, referred to as the **sparsity** criterion, that makes it possible to design better algorithms for searching in high dimensional space. Because of the sparse coding step in the first stage, the conventional nearly duplicate search problem becomes a (R, S)-nearly duplicates search problem.

Given a query vector $\mathbf{q} \in \mathbb{R}^m$, the objective is to find the objects in $\mathcal{D}$ that are (R, S)-nearly duplicates of query $\mathbf{q}$. To motivate the proposed algorithm, we start with the random projection algorithms for high dimension nearest neighbor search, based on Johnson Lindenstrauss Theorem (Johnson and Lindenstrauss 1984).

Theorem 1. *(Johnson Lindenstrauss Theorem) Let $U \in \mathbb{R}^{m\times d}$ be a random matrix, with each entry $U_{i,j}$ i.i.d. samples from a Gaussian distribution $\mathcal{N}(x|0, 1)$. Let $P_U : \mathbb{R}^m \mapsto \mathbb{R}^d$ be the projection operator that projects a vector in $\mathbb{R}^m$ into the subspace spanned by the column vectors in U. For any two fixed objects $\mathbf{x}, \mathbf{q} \in \mathbb{R}^m$ and for a given*

$\epsilon \in (0,1)$, let $D_P = |P_U(\mathbf{x}-\mathbf{q})|_2^2$ and $D_O = |\mathbf{x}-\mathbf{q}|_2^2$, we have

$$\Pr\Big(D_P \le \frac{d(1-\epsilon)}{m} D_O\Big) \le \exp\Big(-\frac{d\epsilon^2}{4}\Big)$$
$$\Pr\Big(D_P \ge \frac{d(1+\epsilon)}{m} D_O\Big) \le \exp\Big(-\frac{d\epsilon^2[1-\frac{2\epsilon}{3}]}{4}\Big)$$

Based on Johnson Lindenstrauss theorem, a straightforward approach is to project high dimensional vectors into the low dimensional space generated by a random matrix, and then explore methods, such as KD-tree, for efficient nearest neighbor search in the low dimensional space. The main problem with such an approach is that in order to ensure that all the objects within distance r of query $\mathbf{q}$ are returned with a large probability, d has to be $O(4[\epsilon^2(1-2\epsilon/3)]^{-1}\log N)$, which is too large for efficient nearest neighbor search.

An alternative approach is to first run multiple range searches, each in a very low dimensional space, and then assemble the results together by either anding or oring the results returned by each range search. This approach does guarantee that with a large probability, all the objects within distance r of query $\mathbf{q}$ are found in the union of the objects returned by multiple range searches. However, the problem is that many of the returned objects are far away from the query $\mathbf{q}$ and need to be removed. In case when one dimensional range search is used, according to Johnson Lindenstrauss Theorem, the probability of including any object $\mathbf{x}$ with $|\mathbf{x}-\mathbf{q}|_2 \ge r\sqrt{(1+\epsilon)/(1-\epsilon)}$ can be $\min(1, d\exp(-\epsilon^2/4))$. We evidently could make better tradeoff between retrieval precision and efficiency by being more careful in designing the combination strategy.

For instance, we could create multiple number (defined as K) of d-dimensional embedding: for each d-dimensional embedding, we merge the objects returned by every one-dimensional embedding by anding; we then combine the objects returned by each d-dimensional embedding by oring. The main shortcoming of this strategy is that it requires both d and K to be large numbers, leading to a large number (i.e. dK) of one-dimensional searching structures to be implemented and consequentially low efficiency in search.

Before we present the proposed algorithm, we first extend Johnson Lindenstrauss Theorem. Instead of using the projection operator P_U, the extended theorem directly uses a random matrix U to generate projection of data points, significantly simplifying the computation and analysis. To this end, first, we have the following concentration inequality for a Gaussian random matrix U, whose elements are randomly drawn from distribution $N(x|0, 1/m)$.

Theorem 2. *(Candes and Tao 2005) Let $U \in \mathbb{R}^{m\times d}$ be a Gaussian random matrix, with $m > d$. Let $\sigma_{\max}(U)$ and $\sigma_{\min}(U)$ be the maximum and minimum singular values for matrix U, respectively. We have*

$$\Pr\Big(\sigma_{\max}(U) > 1 + \sqrt{\frac{d}{m}} + \eta + t\Big) \le \exp(\frac{-mt^2}{2}) \quad (4)$$

$$\Pr\Big(\sigma_{\min}(U) < 1 - \sqrt{\frac{d}{m}} + \eta - t\Big) \le \exp(\frac{-mt^2}{2}) \quad (5)$$

where

$$\eta = \frac{1}{2m^{1/3}}\gamma^{1/6}(1+\sqrt{\gamma})^{2/3}$$

with $\gamma \in (0,1)$.

Corollary 3. *Let $U \in \mathbb{R}^{m\times d}$ be a Gaussian random matrix with $m > d$. Assume m and d are large enough. For fixed $\mathbf{x}$, with a high probability at least $1-2e^{-d/2}$, we have*

$$(1-\sqrt{d/m})|P_U\mathbf{x}|_2 \le |U^\top\mathbf{x}|_2 \le (1+3\sqrt{d/m})|P_U\mathbf{x}|_2$$

Proof. Using Theorem 2, for a fixed $\mathbf{x}$, with a probability at least $1-\delta$, for any vector $\mathbf{x}$, we have

$$\begin{aligned}|U^\top x|_2 &= \sqrt{x^\top UU^\top x} \ge \sigma_{\min}\sqrt{x^\top VV^\top x} \\ &= \sigma_{\min}\sqrt{x^\top VV^\top VV^\top x} = \sigma_{\min}|P_U\mathbf{x}|_2 \\ &\ge \left(1-\sqrt{\frac{d}{m}}+\eta-\sqrt{\frac{2\ln(2/\delta)}{m}}\right)|P_U\mathbf{x}|_2\end{aligned}$$

where V contains the eigenvectors of $UU^\top$. Similarly, we have

$$\begin{aligned}|U^\top x|_2 &\le \sigma_{\max}|P_U\mathbf{x}|_2 \\ &\le \left(1+\sqrt{\frac{d}{m}}+\eta+\sqrt{\frac{2\ln(2/\delta)}{m}}\right)|P_U\mathbf{x}|_2\end{aligned}$$

With sufficiently large d, we can replace η with $\sqrt{d/m}$. We complete the proof by setting $\delta = 2e^{-d/2}$. □

The following modified Johnson Lindenstrauss Theorem follows directly Theorem 1 and Corollary 3.

Theorem 4. *(Modified Johnson Lindenstrauss Theorem) Let $U \in \mathbb{R}^{m\times d}$ be a Gaussian random matrix. Assume m and d are sufficiently large. For any fixed $\mathbf{x}, \mathbf{q} \in \mathbb{R}^m$ and for a given $\epsilon \in (0,1)$, let $D_U = |U^\top\mathbf{x} - U^\top\mathbf{q}|_2^2$, $D_O = |\mathbf{x}-\mathbf{q}|_2^2$, $p_1 = (1+3\sqrt{d/m})^2$ and $p_2 = (1-\sqrt{d/m})^2$, we have*

$$\Pr\Big(D_U \le \frac{dp_1(1-\epsilon)D_O}{m}\Big) \le \exp\Big(-\frac{d\epsilon^2}{4}\Big)$$
$$\Pr\Big(D_U \ge \frac{dp_2(1+\epsilon)D_O}{m}\Big) \le \exp\Big(-\frac{d\epsilon^2[1-\frac{2\epsilon}{3}]}{4}\Big)$$

To reduce the number of false positives, we explore the sparsity criterion in (R,S)-nearly duplicate search by utilizing the Restricted Isometry Property (RIP) (Candes and Tao 2005; Candès, Romberg, and Tao 2006) for sparse vectors.

Theorem 5. *(Restricted Isometry Property) Assume $d < m$ and let $U \in \mathbb{R}^{m\times d}$ be a Gaussian random matrix whose entries are i.i.d. samples from $\mathcal{N}(x|0, 1/m)$. If S/m is small enough, then with a high probability there exists a positive constant $\delta_S < 1$ such that for any $\mathbf{c} \in \mathbb{R}^m$ with at most S non-zero elements, we have*

$$(1-\delta_S)|\mathbf{c}|_2 \le \sqrt{\frac{m}{d}}|U^\top\mathbf{c}|_2 \le (1+\delta_S)|\mathbf{c}|_2$$

Since in (R,S)-nearly duplicate search, the nearest neighbor $\mathbf{x}$ differs from a query $\mathbf{q}$ by at most S attributes, $\mathbf{x}-\mathbf{q}$ is a S-sparse vector and is eligible for the application of RIP condition. In the proposed algorithm, we utilize this property in the nearly duplicate search to filter out the false positive points.

We now state our algorithm for efficient (R, S)-nearly duplicate search. Let d be some modest integer so that (i) for given $\epsilon \in (0, 1)$, d is large enough to make $\exp(-d\epsilon^2/4)$ a small value, and (ii) d is small enough that allows for efficient range search in space of d dimensions. Choose K as $K = \lceil (CS \log m)/d \rceil$, where $C > 1$ is a parameter that needs to be adjusted empirically. We generate K random matrices $U_1, U_2, \ldots, U_K$ of size $m \times d$ whose entries are i.i.d. samples from $\mathcal{N}(x|0, 1/m)$. For each random matrix U_j, we map all the objects in $\mathcal{D}$ from $\mathbb{R}^m$ to $\mathbb{R}^d$ by $U_j^\top \mathbf{x}$, denoted by $\mathcal{D}_j = \{U_j^\top \mathbf{x}_1, U_j^\top \mathbf{x}_2, \ldots, U_j^\top \mathbf{x}_N\}$. We build a search structure for each $\mathcal{D}_j$ and assume range search over $\mathcal{D}_j$ can be done efficiently. We denote by $\mathcal{M}_j(\mathbf{q}, r) = \{i \in [N] : |U_j^\top(\mathbf{x}_i - \mathbf{q})|_2 \leq r\}$ the subset of objects whose projection by U_j is within distance r of query $\mathbf{q}$. We introduce a filtering procedure after merging the objects returned by the range search. It discards the objects whose cumulative distance to query (i.e., $\sqrt{\frac{m}{Kd}\sum_{j'=1}^{j} |U_{j'}^\top(\mathbf{x} - \mathbf{q})|_2^2}$) is larger than the threshold $(1 + \delta_S)R$, where the factor $\sqrt{m/[Kd]}$ is used to scale the variance $1/m$ used in generating the projection matrix.

Theorem 6. *Let $\mathcal{M}(\mathbf{q}, R, S)$ be the set of (R, S)-nearly duplicate objects in $\mathcal{D}$ for query $\mathbf{q}$, and $\mathcal{M}$ be the set of objects returned by Algorithm 1 for query $\mathbf{q}$. We have (i) with a high probability that $\mathcal{M}$ includes all the objects in that are (R, S)-nearly duplicates of query $\mathbf{q}$ i.e.,*

$$\Pr(\mathcal{M}(\mathbf{q}, R, S) \subseteq \mathcal{M}) \geq 1 - N\exp(-\frac{dK\epsilon^2[1 - \frac{2\epsilon}{3}]}{4}) \quad (6)$$

and (ii) all the returned objects are within a short distance from $\mathbf{q}$, i.e.,

$$\max_{i \in \mathcal{M}} |\mathbf{x}_i - \mathbf{q}|_2 \leq \frac{1 + \delta_S}{1 - \delta_S} R \quad (7)$$

Proof. We denote by $\mathcal{M}(\mathbf{q}, R) = \{i \in [N] : |\mathbf{x}_i - \mathbf{q}|_2 \leq R$. For inequality in (6), it is sufficient to bound $\Pr(\mathcal{M}(\mathbf{q}, R) \subseteq \mathcal{M})$. If we don't take into account the filtering steps, using Theorem 4, for fixed $\mathbf{x} \in \mathcal{M}(\mathbf{q}, R)$, the probability that $\mathbf{x}$ is not returned by a d-dimensional range search is less than $\exp(-d\epsilon^2[1 - 2\epsilon/3]/4)$. This probability is reduced to $\exp(-dK\epsilon^2[1 - 2\epsilon/3]/4)$ after the union of data points returned by K independent d-dimensional range searches. By taking the union bound over all N data points we have (6). Note that the filtering steps do not affect the result because of the RIP condition in Theorem 5. The second inequality directly follows Theorem 5 because of the filtering steps. □

As indicated by Theorem 6, the returned list $\mathcal{M}$ contains almost all the nearly duplicates, and most of the returned objects are within distance of $(1 + \delta_S)/(1 - \delta_S)R$. It is the inequality in (7) that allows us to reduce the false positives.

Algorithm 1 Nearly Duplicate Search by Random Projection with Filtering (**RPF**)

```
 1: Input
      • $R_{Ham}$: Hamming distance threshold
      • $r_{Ham}$: Hamming radius
      • $m$: the number of landmarks
      • $t$: the number of nearest neighbors
      • $\mathbf{q}$: a query
 2: Initialization: $\mathcal{M} = \emptyset$ and $d_q = 0$
 3: Obtain the sparse representations using Eq.(1)-(3).
 4: Get the $K$ hash tables $\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_K$ using Eq.(8).
 5: for $j = 1, 2, \ldots, K$ do
 6:   Run efficient search over $\mathcal{T}_j$ with Hamming radius $r_{Ham}$
 7:   Return $\mathcal{M}_j(\mathbf{q}, r)$ along with the distance information, i.e., for $i \in \mathcal{M}_j(\mathbf{q}, r)$, we compute $d_i^j = |\mathcal{T}_j(\mathbf{x}_i) - \mathcal{T}_j(\mathbf{q})|_1$.
 8:   for $i \in \mathcal{M}$ do
 9:     if $i \notin \mathcal{M}_j$ then
10:       $d_i = d_i + r_{Ham}$
11:     else
12:       $d_i = d_i + d_i^j$, $\mathcal{M}_j = \mathcal{M}_j \setminus \{i\}$
13:     end if
14:     $\mathcal{M} = \mathcal{M} \setminus \{i\}$ if $d_i \geq R_{Ham}$
15:   end for
16:   for $i \in \mathcal{M}_j$ do
17:     $d_i = d_q + d_i^j$
18:     $\mathcal{M} = \mathcal{M} \cup \{i\}$ if $d_i < R_{Ham}$
19:   end for
20:   $d_q = d_q + r_{Ham}$
21: end for
22: Return $\mathcal{M}$
```

Scalable Extension to Binary Codes

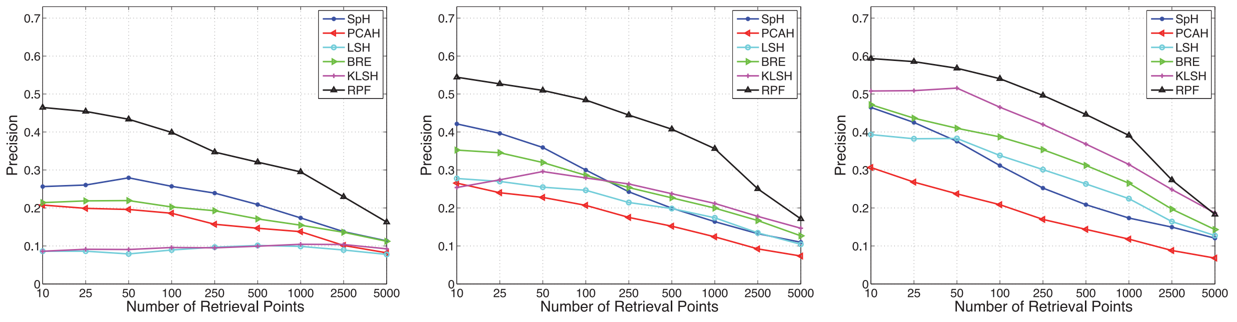
There are two practical issues of the above algorithm. First, the storage space needed to record the final representations is too large, especially while dealing with the large scale datasets. Second, the search process for each $\mathcal{D}$ is time consuming, even using the KD-tree structure. In this subsection, we extend our algorithm to the binary codes (*i.e.*, Hamming space). In the Hamming space, the encoded data are highly compressed so that they can be loaded into the memory. What's more, the Hamming distance between two binary codes can be computed very efficiently by using bit XOR operation (Zhang et al. 2010b).

In (Charikar 2002), the authors proposed an algorithm to generate binary codes based on rounding the output of a product with a random hyperplane $\mathbf{w}$:

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

where $\mathbf{w}$ is a vector generated from a zero-mean multivariate Gaussian $\mathcal{N}(0, \mathbf{I})$ of the same dimension as the input $\mathbf{x}$. In this case, for any two points $\mathbf{x}_i, \mathbf{x}_j$, we have the following property:

$$Pr[h(\mathbf{x}_i) = h(\mathbf{x}_j)] = 1 - \frac{1}{\pi}\cos^{-1}(\frac{\mathbf{x}_i^T\mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}) \quad (9)$$



(a) At 8 bits (b) At 16 bits (c) At 32 bits

Figure 1: Comparison of the performance on the patches dataset. (a)-(c) are the performances for the hash codes of 8 bits, 16 bits and 32 bits respectively.

It is important to note that our proposed algorithm also utilizes the random projection method in the second stage, hence, we can easily extend our algorithm to generate the binary codes. We define the new hash tables as $\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_K$. In each hash table, we retrieve the points within a specific Hamming radius and we filter out the points whose accumulate Hamming distances are larger than a threshold before returning the answers. It is reasonable to consider that the Hamming radius is related to the reduced dimensions, and the threshold is related to both the number of hash tables and the reduced dimensions. Therefore, we assume the Hamming radius $r_{Ham} = \alpha d$ and the threshold $R_{Ham} = \beta K d$.

Algorithm 1 presents the detailed steps of our approach. There are several important features. First, we obtain the sparse representations in step 3. Second, the filtering procedure takes place in step 12. In this case, we remove the data points whose accumulative distances are larger than R_{Ham}. Third, in step 10, for $\mathbf{x} \in \mathcal{M} \setminus \mathcal{M}_j$, we lower bound its distance to query $\mathbf{q}$ by $d_i = d_i + r_{Ham}$. Finally, we introduce d_q, the minimum distance to query $\mathbf{q}$ for any object $\mathbf{x} \notin \mathcal{M}$. This quantity is used to bound the distance between query $\mathbf{q}$ and any $\mathbf{x} \in \mathcal{M}_j \setminus \mathcal{M}$, as shown in step 17.

Experiments

Compared Algorithms

We compare the proposed Random Projection with Filtering (RPF) algorithm with the following state-of-the-art hashing methods:

- Spectral Hashing (**SpH**) (Weiss, Torralba, and Fergus 2008)
- PCA Hashing (**PCAH**) (Wang et al. 2006)
- Learning to Hash with Binary Reconstructive Embeddings (**BRE**) (B. and Darrell 2010)
- Locality Sensitive Hashing (**LSH**) (Charikar 2002)
- Kernelized Locality Sensitive Hashing (**KLSH**) (Kulis and Grauman 2009)

Datasets

The following three datasets are used in our experiment:

- **patches**[1]: It contains $59,500$ 20x20 grey-level motorcycle images. Each image in this database is represented by a vector of 400 dimensions.
- **LabelMe**[2]: It contains $22,019$ images and each of them is represented by a GIST descriptor, which is a vector of 512 dimensions.
- **GIST1M**[3]: We collect 1 million images from the Flickr and use the code provided on the web[4] to extract the GIST descriptors from the images. Each GIST descriptor is represented by a 512-dimensional vector.

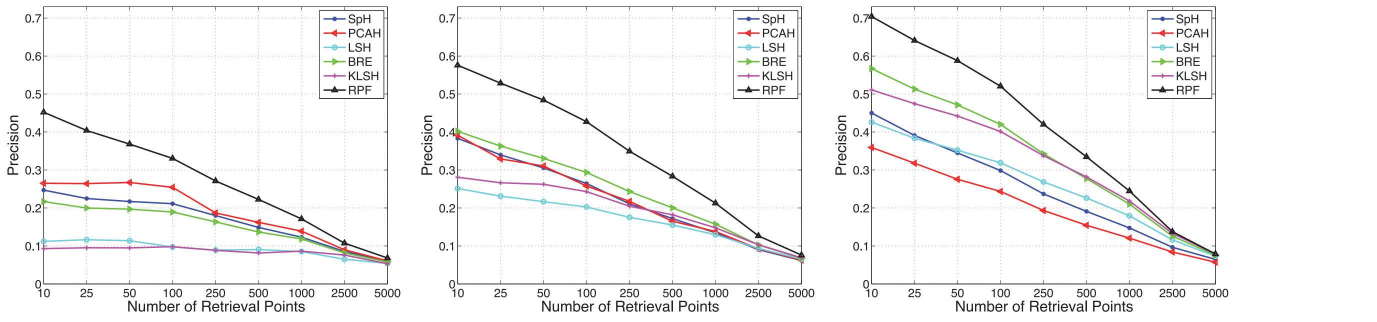
For each dataset, we randomly select 1k data points as queries and use the remaining to form the gallery database. Similar to the criterion used in (Wang, Kumar, and Chang 2010), a returned point is considered as a true neighbor if it lies in the top 2 percentile points closest to a query. In this paper, we use the Hamming ranking method to evaluate the performances of all techniques. We first calculate the Hamming distance between the hash codes of the query and each point in the dataset. The points are then ranked according to the corresponding Hamming distances, and a certain number of top ranked points are retrieved (Xu et al. 2011b).

To perform Hamming ranking for the hashing methods with multiple hash tables, i.e., LSH, KLSH, we compute the Hamming distance of a point and the query by the minimal Hamming distance found in all the hash tables. In RPF, we rank the retrieval points according to the accumulate Hamming distance. LSH, KLSH and RPF employ five hash tables in all the experiments.

Parameter Selection

In the first stage, for the purpose of efficiency, we use *random sampling* to select several data points as landmarks. In all the experiments, the number of landmarks m, the nearest number t and the bandwidth h are set to 200, 40 and 0.8 respectively. We do not use a very large number of landmarks due to the consideration of efficiency. In the second stage,

[1]http://ttic.uchicago.edu/∼gregory/download.html
[2]http://labelme.csail.mit.edu/instructions.html
[3]http://www.zjucadcg.cn/dengcai/Data/NNSData.html
[4]http://www.vision.ee.ethz.ch/∼zhuji/felib.html



(a) At 8 bits (b) At 16 bits (c) At 32 bits

Figure 2: Comparison of the performance on the LabelMe dataset. (a)-(c) are the performances for the hash codes of 8 bits, 16 bits and 32 bits respectively.

Table 1: The *precision*, *training time* and *test time* of all the algorithms at 32 bits on the GIST1M dataset. Both the *training time* and *test time* are in second.

Number of Retrieval Points	100	250	500	1000	2500	5000	Training time	Test time
SpH	0.3576	0.3278	0.3013	0.2749	0.2369	0.2082	39.9546	0.0796
PCAH	0.3558	0.3199	0.2901	0.2614	0.2204	0.1918	29.6332	0.0022
LSH	0.4577	0.4303	0.4082	0.3832	0.3489	0.3183	5.7340	0.0019
BRE	0.5770	0.5400	0.5074	0.4719	0.4182	0.3720	138.3070	0.0208
KLSH	0.5021	0.4786	0.4575	0.4326	0.3940	0.3591	8.5742	0.0098
RPF	**0.7349**	**0.6903**	**0.6507**	**0.6072**	**0.5387**	**0.4783**	96.5935	0.0118

we set $\alpha = 0.5$ and $\beta = 0.8$ according to the empirical study based on the least squares cross-validation.

Results

Experiment on the patches dataset We first compare the performance of our RPF method with other hashing methods on patches dataset. The performances of all the methods, in terms of the precision versus the number of retrieved points, are illustrated in Figure 1. On one hand, we can observe that the precision decreases in all hashing approaches when more data points are retrieved. On the other hand, all the hashing algorithms improve their performances as the code length increases. KLSH algorithm achieves a significant improvement from 8 bits to 32 bits. By introducing the filtering steps based on the techniques of sparse coding and compressed sensing, the proposed RPF method achieves promising performance on this dataset and consistently outperforms its competitors for all code lengths.

Experiment on the LabelMe dataset Figure 2 presents the performances of all the methods on LabelMe dataset. Similar to the results on patches dataset, with the effective filtering steps, the proposed RPF algorithm achieves the best performance for all the code lengths.

Experiment on the GIST1M dataset Table 1 presents the performances of different methods at 32 bits on GIST1M dataset. Considering the precision versus the number of retrieved points, we observe that the RPF algorithm greatly outperforms other methods, which justifies the effectiveness of the filtering approach. Considering the training time, we can find that BRE is the most expensive to train, while LSH need the least training time. KLSH needs to compute a sampled kernel matrix which slows down its computation. Most of the training time in the proposed RPF is spent on the process to obtain the sparse representations. Although our method is somewhat slower than others, it is still very fast and practical in absolute terms. In terms of the test time, all the hashing algorithms are efficient.

Conclusions

In this paper, we design a randomized algorithm for nearly duplicate search in high dimensional space. The proposed algorithm addresses the shortcomings of many hashing algorithms that tend to return many false positive examples for a given query. The key idea is designing a sparse representation for the data and exploring the sparsity condition of nearly duplicate search by introducing a filtering procedure into the search algorithm, based on the theory of compressed sensing. Empirical studies on three real-world datasets show the promising performance of the proposed algorithm compared to the state-of-the-art hashing methods for high dimensional nearest neighbor search. In the future, we plan to further explore the sparse coding methods that can effective and computationally efficient for large data sets.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China (Grant Nos: 61125203, 90920303), National Basic Research Program of China (973 Program) under Grant 2009CB320801, and US Army Research (ARO Award W911NF-11-1-0383).

References

Andoni, A., and Indyk, P. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM* 51(1):117–122.

Arge, L.; de Berg, M.; Haverkort, H. J.; and Yi, K. 2004. The priority r-tree: A practically efficient and worst-case-optimal r-tree. In *Cache-Oblivious and Cache-Aware Algorithms*.

Arya, S.; Malamatos, T.; and Mount, D. M. 2009. Space-time tradeoffs for approximate nearest neighbor searching. *J. ACM* 57(1).

B., K., and Darrell, T. 2010. Learning to hash with binary reconstructive embeddings. In *NIPS*.

Cai, D.; He, X.; Zhang, W. V.; and Han, J. 2007. Regularized locality preserving indexing via spectral regression. In *CIKM*.

Cai, D.; He, X.; and Han, J. 2005. Using graph model for face analysis. Technical report, Computer Science Department, UIUC, UIUCDCS-R-2005-2636.

Cai, D.; He, X.; and Han, J. 2007a. Isometric projection. In *AAAI*.

Cai, D.; He, X.; and Han, J. 2007b. Spectral regression for dimensionality reduction. Technical report, Computer Science Department, UIUC, UIUCDCS-R-2007-2856.

Candes, E., and Tao, T. 2005. Decoding by linear programming. *IEEE Transactions on Information Theory* 51:4203–4215.

Candès, E. J.; Romberg, J. K.; and Tao, T. 2006. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory* 52(2):489–509.

Charikar, M. 2002. Similarity estimation techniques from rounding algorithms. In *STOC*.

Chen, X., and Cai, D. 2011. Large scale spectral clustering with landmark-based representation. In *AAAI*.

Ciaccia, P.; Patella, M.; and Zezula, P. 1997. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*.

Datar, M.; Immorlica, N.; Indyk, P.; and Mirrokni, V. S. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Symposium on Computational Geometry 2004*, 253–262.

Dong, W.; Wang, Z.; Josephson, W.; Charikar, M.; and Li, K. 2008. Modeling lsh for performance tuning. In *CIKM*.

Donoho, D.; Maleki, A.; and Montanari, A. 2009. Message-passing algorithms for compressed sensing. In *PNAS*.

Donoho, D. 2006. Compressed sensing. *IEEE Transactions on Information Theory* 52:1289–1306.

Eshghi, K., and Rajaram, S. 2008. Locality sensitive hash functions based on concomitant rank order statistics. In *KDD*.

Gionis, A.; Indyk, P.; and Motwani, R. 1999. Similarity search in high dimensions via hashing. In *VLDB*.

Goldstein, J.; Platt, J. C.; and Burges, C. J. C. 2004. Redundant bit vectors for quickly searching high-dimensional regions. In *Deterministic and Statistical Methods in Machine Learning*.

Johnson, W., and Lindenstrauss, J. 1984. Extensions of lipschitz mappings into a hilbert space.

Kulis, B., and Grauman, K. 2009. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*.

Kulis, B.; Jain, P.; and Grauman, K. 2009. Fast similarity search for learned metrics. *IEEE Trans. Pattern Anal. Mach. Intell.* 31(12):2143–2157.

Kushilevitz, E.; Ostrovsky, R.; and Rabani, Y. 1998. Efficient search for approximate nearest neighbor in high dimensional spaces. In *STOC*.

Liu, W.; He, J.; and Chang, S.-F. 2010. Large graph construction for scalable semi-supervised learning. In *ICML*.

Lv, Q.; Josephson, W.; Wang, Z.; Charikar, M.; and Li, K. 2007. Multi-probe lsh: Efficient indexing for high-dimensional similarity search. In *VLDB*.

Norouzi, M., and Fleet, D. J. 2011. Minimal loss hashing for compact binary codes. In *ICML*.

Raginsky, M., and Lazebnik, S. 2009. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*.

Robinson, J. T. 1981. The K-D-B-tree: A search structure for large multidimensional dynamic indexes. In *SIGMOD*.

Salakhutdinov, R., and Hinton, G. E. 2009. Semantic hashing. *International Journal of Approximate Reasoning* 50(7):969–978.

Schniter, P. 2010. Turbo reconstruction of structured sparse signals. In *CISS*.

Silpa-Anan, C., and Hartley, R. 2008. Optimised kd-trees for fast image descriptor matching. In *CVPR*.

Wang, X.; Zhang, L.; Jing, F.; and Ma, W. 2006. Annosearch: Image auto-annotation by search. In *CVPR*.

Wang, J.; Kumar, S.; and Chang, S.-F. 2010. Sequential projection learning for hashing with compact codes. In *ICML*.

Weiss, Y.; Torralba, A.; and Fergus, R. 2008. Spectral hashing. In *NIPS*.

Wu, C.; Zhu, J.; Cai, D.; Chen, C.; and Bu, J. 2012. Semi-supervised nonlinear hashing using bootstrap sequential projection learning. *IEEE Transactions on Knowledge and Data Engineering*.

Xu, B.; Bu, J.; Chen, C.; Cai, D.; He, X.; Liu, W.; and Luo, J. 2011a. Efficient manifold ranking for image retrieval. In *SIGIR*.

Xu, H.; Wang, J.; Li, Z.; Zeng, G.; Li, S.; and Yu, N. 2011b. Complementary hashing for approximate nearest neighbor search. In *ICCV*.

Yu, Z.; Cai, D.; and He, X. 2010. Error-correcting output hashing in fast similarity search. In *The 2nd International Conference on Internet Multimedia Computing and Service*.

Zhang, D.; Wang, J.; Cai, D.; and Lu, J. 2010a. Laplacian co-hashing of terms and documents. In *ECIR*.

Zhang, D.; Wang, J.; Cai, D.; and Lu, J. 2010b. Self-taught hashing for fast similarity search. In *SIGIR*.