

# ANTI-SPARSE CODING FOR APPROXIMATE NEAREST NEIGHBOR SEARCH

Hervé Jégou<sup>†</sup>, Teddy Furon<sup>†</sup> and Jean-Jacques Fuchs<sup>\*</sup>

<sup>†</sup>INRIA Rennes Bretagne Atlantique  
Campus de Beaulieu, Rennes, France

<sup>\*</sup> University of Rennes 1

## ABSTRACT

This paper proposes a binarization scheme for vectors of high dimension based on the recent concept of *anti-sparse* coding, and shows its excellent performance for approximate nearest neighbor search. Unlike other binarization schemes, this framework allows, up to a scaling factor, the *explicit* reconstruction from the binary representation of the original vector. The paper also shows that random projections which are used in Locality Sensitive Hashing algorithms, are significantly outperformed by regular frames for both synthetic and real data if the number of bits exceeds the vector dimensionality, i.e., when high precision is required.

**Index Terms**— sparse coding, spread representations, approximate neighbors search, Hamming embedding

## 1. INTRODUCTION

This paper addresses the problem of approximate nearest neighbor (ANN) search in high dimensional spaces. Given a query vector, the objective is to find, in a collection of vectors, those which are the closest to the query with respect to a given distance function. We focus on the Euclidean distance in this paper. This problem has a very high practical interest, since matching the descriptors representing the media is the most consuming operation of most state-of-the-art audio [1], image [2] and video [3] indexing techniques. There is a large body of literature on techniques whose aim is the optimization of the trade-off between retrieval time and complexity.

We are interested by the techniques that regard the memory usage of the index as a major criterion. This is compulsory when considering large datasets including dozen millions to billions of vectors [4, 5, 2, 6], because the indexed representation must fit in memory to avoid costly hard-drive accesses. One popular way is to use a Hamming Embedding function that maps the real vectors into binary vectors [4, 5, 2]: Binary vectors are compact, and searching the Hamming space is efficient (XOR operation and bit count) even if the comparison is exhaustive between the binary query and the database vectors. An extension to these techniques is the asymmetric scheme [7, 8] which limits the approximation done on the query, leading to better results for a slightly higher complexity.

We propose to address the ANN search problem with an *anti-sparse* solution based on the design of *spread* representations recently proposed by Fuchs [9]. Sparse coding has received in the last decade a huge attention from both theoretical and practical points of view. Its objective is to represent a vector in a higher dimensional space with a very limited number of non-zeros components. Anti-sparse coding has the opposite properties. It offers a robust representation of a vector in a higher dimensional space with all the components sharing evenly the information.

Sparse and anti-sparse coding admits a common formulation. The algorithm proposed by Fuchs [9] is indeed similar to path-following methods based on continuation techniques like [10]. The anti-sparse problem considers a  $\ell_\infty$  penalization term where the sparse problem usually considers the  $\ell_1$  norm. The penalization in  $\|\mathbf{x}\|_\infty$  limits the range of the coefficients which in turn tend to ‘stick’ their value to  $\pm\|\mathbf{x}\|_\infty$  [9]. As a result, the anti-sparse approximation offers a natural binarization method.

Most importantly and in contrast to other Hamming Embedding techniques, the binarized vector allows an explicit and reliable reconstruction of the original database vector. This reconstruction is very useful to refine the search. First, the comparison of the Hamming distances between the binary representations identifies some potential nearest neighbors. Second, this list is refined by computing the Euclidean distances between the query and the reconstructions of the database vectors.

We also provide a Matlab package to reproduce the analysis comparisons reported in this paper<sup>1</sup>. The paper is organized as follows. Section 2 introduces the anti-sparse coding framework. Section 3 describes the corresponding ANN search method which is evaluated in Section 4 on both synthetic and real data.

## 2. SPREAD REPRESENTATIONS

This section briefly describes the anti-sparse coding of [9]. We first introduce the objective function and provide the guidelines of the algorithm giving the spread representation of a given input real vector.

Let  $A = [\mathbf{a}_1 | \dots | \mathbf{a}_m]$  be a  $d \times m$  ( $d < m$ ) full rank matrix. For any  $\mathbf{y} \in \mathbb{R}^d$ , the system  $A\mathbf{x} = \mathbf{y}$  admits an infinite number of solutions. To single out a unique solution, one add a constraint as for instance seeking a minimal norm solution. Whereas the case of the Euclidean norm is trivial, and the case of the  $\ell_1$ -norm stems in the vast literature of sparse representation, Fuchs recently studied the case of the  $\ell_\infty$ -norm. Formally, the problem is:

$$\mathbf{x}^* = \min_{\mathbf{x}: A\mathbf{x}=\mathbf{y}} \|\mathbf{x}\|_\infty, \quad (1)$$

with  $\|\mathbf{x}\|_\infty = \max_{i \in \{1, \dots, m\}} |x_i|$ . Interestingly, by minimizing the range of the components,  $m - d + 1$  of them are stuck to the limit, i.e.  $x_i = \pm\|\mathbf{x}\|_\infty$  (a proof will be presented in a journal version). Fuchs also exhibits an efficient way to solve (1). He proposes to solve the series of simpler problems  $\mathbf{x}_h^* = \min_{\mathbf{x} \in \mathbb{R}^m} J_h(\mathbf{x})$  with  $J_h(\mathbf{x}) = \|A\mathbf{x} - \mathbf{y}\|_2^2/2 + h\|\mathbf{x}\|_\infty$  for some decreasing values of  $h$ . As  $h \rightarrow 0$ ,  $\mathbf{x}_h^* \rightarrow \mathbf{x}^*$ .

The reader may refer to the simple Matlab implementation we provide for an algorithmic presentation of the method.

<sup>1</sup>This work was realized as part of the Quaero Project, funded by OSEO, French State agency for innovation.

<sup>1</sup><http://gforge.inria.fr/projects/antisparse>

## 2.1. The sub-differential set

For a fixed  $h$ ,  $J_h$  is not differentiable due to  $\|\cdot\|_\infty$ . Therefore, we need to work with sub-differential sets. The sub-differential set  $\partial f(\mathbf{x})$  of function  $f$  at  $\mathbf{x}$  is the set of gradients  $\mathbf{v}$  s.t.  $f(\mathbf{x}') - f(\mathbf{x}) \geq \mathbf{v}^\top(\mathbf{x}' - \mathbf{x})$ ,  $\forall \mathbf{x}' \in \mathbb{R}^m$ . For  $f \equiv \|\cdot\|_\infty$ , we have:

$$\partial f(\mathbf{0}) = \{\mathbf{v} \in \mathbb{R}^m : \|\mathbf{v}\|_1 \leq 1\}, \quad (2)$$

$$\begin{aligned} \partial f(\mathbf{x}) &= \{\mathbf{v} \in \mathbb{R}^m : \|\mathbf{v}\|_1 = 1, \\ &v_i x_i \geq 0 \text{ if } |x_i| = \|\mathbf{x}\|_\infty, \\ &v_i = 0 \text{ else}\}, \text{ for } \mathbf{x} \neq \mathbf{0} \end{aligned} \quad (3)$$

Since  $J_h$  is convex,  $\mathbf{x}_h^*$  is solution iff  $\mathbf{0}$  belongs to the sub-differential set  $\partial J_h(\mathbf{x}_h^*)$ , i.e. iff there exist  $\mathbf{v} \in \partial f(\mathbf{x}_h^*)$  s.t.

$$A^\top(A\mathbf{x}_h^* - \mathbf{y}) + h\mathbf{v} = \mathbf{0} \quad (4)$$

## 2.2. Initialization and first iteration

For  $h_0$  large enough,  $J_{h_0}(\mathbf{x})$  is dominated by  $\|\mathbf{x}\|_\infty$ , and the solution writes  $\mathbf{x}_{h_0}^* = \mathbf{0}$  and  $\mathbf{v} = h_0^{-1}A^\top\mathbf{y} \in \partial f(\mathbf{0})$ . (2) shows that this solution no longer holds for  $h < h_1$  with  $h_1 = \|A^\top\mathbf{y}\|_1$ .

For  $\|\mathbf{x}\|_\infty$  small enough compared to the norm of  $A^\top\mathbf{y}$ ,  $J_h(\mathbf{x})$  is dominated by  $\|\mathbf{y}\|^2 - \mathbf{x}^\top A^\top\mathbf{y} + h\|\mathbf{x}\|_\infty$  whose minimizer is  $\mathbf{x}_h^* = \|\mathbf{x}\|_\infty \text{sign}(A^\top\mathbf{y})$ . In this case,  $\partial f(\mathbf{x})$  is the set of vectors  $\mathbf{v}$  s.t.  $\text{sign}(\mathbf{v}) = \text{sign}(\mathbf{x})$  and  $\|\mathbf{v}\|_1 = 1$ . Multiplying (4) by  $\text{sign}(\mathbf{v})^\top$  on the left and substituting  $\mathbf{x}_h^*$ , we have

$$h = h_1 - \|\text{Asign}(A^\top\mathbf{y})\|^2 \|\mathbf{x}\|_\infty. \quad (5)$$

This shows that i)  $\mathbf{x}_h^*$  can be a solution for  $h < h_1$ , and ii)  $\|\mathbf{x}\|_\infty$  increases as  $h$  decreases. Yet, Eq. (4) also imposes that  $\mathbf{v} = \nu_1 - \mu_1 \|\mathbf{x}\|_\infty$ , with  $\nu_1 \triangleq h^{-1}A^\top\mathbf{y}$  and  $\mu_1 \triangleq h^{-1}A^\top\text{Asign}(A^\top\mathbf{y})$ . But, the condition  $\text{sign}(\mathbf{v}) = \text{sign}(\mathbf{x})$  from (3) must hold, i.e. no change of sign is allowed in  $\mathbf{v}$ . This limits  $\|\mathbf{x}\|_\infty$  by  $\rho_{i_2}$  where  $\rho_i = \nu_{1,i}/\mu_{1,i}$  and  $i_2 = \arg \min_{i:\rho_i > 0}(\rho_i)$ , which in turn translates to a lower bound  $h_2$  on  $h$  via (5).

## 2.3. Index partition

For the sake of simplicity, we introduce  $\mathcal{I} \triangleq \{1, \dots, m\}$ , and the index partition  $\tilde{\mathcal{I}} \triangleq \{i : |x_i| = \|\mathbf{x}\|_\infty\}$  and  $\check{\mathcal{I}} \triangleq \mathcal{I} \setminus \tilde{\mathcal{I}}$ . The restriction of vectors and matrices to  $\tilde{\mathcal{I}}$  (resp.  $\check{\mathcal{I}}$ ) are denoted alike  $\bar{\mathbf{x}}$  (resp.  $\check{\mathbf{x}}$ ). For instance, Eq. (3) translates in  $\text{sign}(\bar{\mathbf{v}}) = \text{sign}(\bar{\mathbf{x}})$ ,  $\|\bar{\mathbf{v}}\|_1 = 1$  and  $\check{\mathbf{v}} = \mathbf{0}$ . The index partition splits (4) into two parts:

$$\check{A}^\top(\check{A}\check{\mathbf{x}} + \bar{A}\text{sign}(\bar{\mathbf{v}})\|\mathbf{x}\|_\infty) = \check{A}^\top\mathbf{y} \quad (6)$$

$$\bar{A}^\top(\check{A}\check{\mathbf{x}} + \bar{A}\text{sign}(\bar{\mathbf{v}})\|\mathbf{x}\|_\infty - \mathbf{y}) = -h\bar{\mathbf{v}} \quad (7)$$

For  $h_2 \leq h < h_1$ , we've seen that  $\bar{\mathbf{x}} = \mathbf{x}$ ,  $\bar{\mathbf{v}} = \mathbf{v}$ , and  $\bar{A} = A$ . Their 'tilde' versions are empty. For  $h < h_2$ , the index partition  $\tilde{\mathcal{I}} = \mathcal{I}$  and  $\check{\mathcal{I}} = \emptyset$  can no longer hold. Indeed, when  $v_{1,i_2}$  is null at  $h = h_2$ , the  $i_2$ -th column of  $A$  moves from  $\bar{A}$  to  $\check{A}$  s.t. now,  $\check{A} = [\mathbf{a}_{i_2}]$ .

## 2.4. General iteration

The general iteration consists i) in determining on which interval  $[h_{k+1}, h_k]$  an index partition holds, ii) giving the expression of the solution  $\mathbf{x}_h^*$  on this interval, iii) and proposing a new index partition to the next iteration. Provided  $\check{A}$  is full rank, (6) gives

$$\check{\mathbf{x}} = \check{\xi}_k + \check{\zeta}_k \|\mathbf{x}\|_\infty, \quad \text{with} \quad (8)$$

$\check{\xi}_k = (\check{A}^\top\check{A})^{-1}\check{A}^\top\mathbf{y}$  and  $\check{\zeta}_k = -(\check{A}^\top\check{A})^{-1}\check{A}\text{sign}(\bar{\mathbf{v}})$ . (7) gives:

$$\bar{\mathbf{v}} = \nu_k - \mu_k \|\mathbf{x}\|_\infty, \quad \text{with} \quad (9)$$

$\mu_k = \bar{A}^\top(I - \bar{A}^\top\check{A}(\check{A}^\top\check{A})^{-1})\bar{A}\text{sign}(\bar{\mathbf{v}})/h$  and  $\nu_k = (\check{A}^\top\mathbf{y} - \check{\xi}_k)/h$ . Left multiplying (7) by  $\text{sign}(\bar{\mathbf{v}})$ , we get:

$$h = \eta_k - v_k \|\mathbf{x}\|_\infty \quad (10)$$

with  $v_k = (\bar{A}\text{sign}(\bar{\mathbf{v}}))^\top(I - \check{A}(\check{A}^\top\check{A})^{-1}\check{A}^\top)\bar{A}\text{sign}(\bar{\mathbf{v}})$ , and  $\eta_k = -\text{sign}(\bar{\mathbf{v}})^\top\bar{A}^\top(\check{A}\check{\mathbf{x}} - \mathbf{y})$ . Note that  $v_k > 0$  so that  $\|\mathbf{x}\|_\infty$  increases when  $h$  decreases.

These equations extend a solution  $\mathbf{x}_h^*$  to the neighborhood of  $h$ . However, we must check that this index partition is still valid as we decrease  $h$  and  $\|\mathbf{x}\|_\infty$  increases. Two events can break the validity:

- Like in the first iteration, a component of  $\bar{\mathbf{v}}$  given in (9) becomes null, which breaks  $\text{sign}(\bar{\mathbf{v}}) = \text{sign}(\bar{\mathbf{x}})$ . This index moves from  $\tilde{\mathcal{I}}$  to  $\check{\mathcal{I}}$ .
- A component of  $\check{\mathbf{x}}$  given in (8) sees its amplitude equalling  $\pm\|\mathbf{x}\|_\infty$ . This index moves from  $\check{\mathcal{I}}$  to  $\tilde{\mathcal{I}}$ , and the sign of this component will be the sign of the new component of  $\bar{\mathbf{x}}$ .

The value of  $\|\mathbf{x}\|_\infty$  for which one of these two events first happens is translated in  $h_{k+1}$  thanks to (10).

## 2.5. Stopping condition and output

If the goal is to minimize  $J_{h_t}(\mathbf{x})$  for a specific target  $h_t$ , then the algorithm stops when  $h_{k+1} < h_t$ . The real value of  $\|\mathbf{x}_{h_t}^*\|_\infty$  is given by (10), and the components not stuck to  $\pm\|\mathbf{x}_{h_t}^*\|_\infty$  by (8).

We obtain the spread representation  $\mathbf{x}$  of the input vector  $\mathbf{y}$ . The vector  $\mathbf{x}$  has many of its components equal to  $\pm\|\mathbf{x}\|_\infty$  ( $m - d + 1$  indeed if  $h_t \rightarrow 0$ ). An approximation of the original vector  $\mathbf{y}$  is

$$\hat{\mathbf{y}} = A\mathbf{x}. \quad (11)$$

**Remark:** Alternately, the termination rule based on the pre-defined target value  $h_t$  may be replaced by another one, such as using a fixed number of iterations. Empirically, this choice leads to comparable results with respect to the approximate search problem.

## 3. INDEXING AND SEARCH MECHANISMS

This section describes how Hamming Embedding functions are used for approximate search, and in particular how the anti-sparse coding framework described in Section 2 is exploited.

### 3.1. Problem statement

Let  $\mathcal{Y}$  be a dataset of  $n$  real vectors,  $\mathcal{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\}$ , where  $\mathbf{y}_i \in \mathbb{R}^d$ , and consider a query vector  $\mathbf{q} \in \mathbb{R}^d$ . We aim at finding the  $k$  vectors in  $\mathcal{Y}$  that are closest to the query, with respect to the Euclidean distance. For the sake of exposure, we consider without loss of generality the nearest neighbor problem, i.e., the case  $k = 1$ . The nearest neighbor of  $\mathbf{q}$  in  $\mathcal{Y}$  is defined as

$$\text{NN}(\mathbf{q}) = \arg \min_{\mathbf{y} \in \mathcal{Y}} \|\mathbf{q} - \mathbf{y}\|^2. \quad (12)$$

The goal of approximate search is to find this nearest neighbor with high probability and using as less resources as possible. The performance criteria are the following:

- The quality of the search, i.e., to which extent the algorithm is able to return the true nearest neighbor ;

- The search efficiency, typically measured by the query time ;
- The memory usage, i.e., the number of bytes used to index a vector  $\mathbf{y}_i$  of the database.

In our paper, we assess the search quality by the recall@R measure: over a set of queries, we compute the proportion for which the system returns the true nearest neighbor in the first R positions.

### 3.2. Approximate search with binary embeddings

A class of ANN methods is based on embedding [4, 5, 2]. The idea is to map the input vectors to a space where the representation is compact and the comparison is efficient. The Hamming space offers these two desirable properties. The key problem is the design of the embedding function  $e : \mathbb{R}^d \rightarrow \mathbb{B}^m$  mapping the input vector  $\mathbf{y}$  to  $\mathbf{b} = e(\mathbf{y})$  in the  $m$ -dimensional Hamming space  $\mathbb{B}^m$ , here defined as  $\{-1, 1\}^m$  for the sake of exposure. Once this function is defined, all the database vectors in  $\mathcal{Y}$  are mapped to  $\mathbb{B}^m$ , and so is the query vector  $\mathbf{q}$ . The search problem is translated into the Hamming space based on the Hamming distance, or, equivalently:

$$\text{NN}_b(e(\mathbf{q})) = \arg \max_{\mathbf{y} \in \mathcal{Y}} e(\mathbf{q})^\top e(\mathbf{y}). \quad (13)$$

$\text{NN}_b(e(\mathbf{q}))$  is returned as the approximate  $\text{NN}(\mathbf{q})$ .

**Binarization with anti-sparse coding.** Given an input vector  $\mathbf{y}$ , the anti-sparse coding of Sect. 2 (with fixed parameters  $A$  and  $h_i$ ) produces  $\mathbf{x}$  with many components equal to  $\pm \|x\|_\infty$ . We consider a “pre-binarized” version  $\hat{x}(\mathbf{y}) = \mathbf{x}/\|\mathbf{x}\|_\infty$ , and the binarized version  $e(\mathbf{y}) = \text{sign}(\mathbf{x})$ .

**Hash function design** The locality sensitive hashing (LSH) algorithm is mainly based on random projection, though different kinds of hash functions have been proposed for the Euclidean space [11]. Let  $A = [\mathbf{a}_1 | \dots | \mathbf{a}_m]$  be a  $d \times m$  matrix storing the  $m$  projection vectors. The most simple way is to take the sign of the projections:  $\mathbf{b} = \text{sign}(A^\top \mathbf{y})$ . Note that this corresponds to the first iteration of our algorithm (see Sect. 2.2).

We also try  $A$  as an uniform frame. A possible construction of such a frame consists in performing a QR decomposition on a  $m \times m$  matrix. The matrix  $A$  is then composed of the  $d$  first rows of the  $Q$  matrix, ensuring that  $A \times A^\top = \mathbb{I}_d$ . Sect. 4 shows that such frames significantly improve the results compared with random projections, for both LSH and anti-sparse coding embedding methods.

### 3.3. Asymmetric schemes

As recently suggested in the literature, a better search quality is obtained by avoiding the binarization of the query vector. Several variants are possible. We consider the simplest one derived from (13), where the query is not binarized in the inner product. For our anti-sparse coding scheme, this amounts to performing the search based on the following maximization:

$$\text{NN}_a(e(\mathbf{q})) = \arg \max_{\mathbf{y} \in \mathcal{Y}} \hat{x}(\mathbf{q})^\top e(\mathbf{y}). \quad (14)$$

The estimate  $\text{NN}_a$  is better than  $\text{NN}_b$ . The memory usage is the same because the vectors in the database  $\{e(\mathbf{y}_i)\}$  are all binarized. For better efficiency, the search (14) is done using look-up tables computed for the query and prior to the comparisons. Doing so, the asymmetric scheme remains a bit slower than the pure bit-based comparison: comparing two optimized implementations, Jain et al. report [8] that the comparison of binary vectors is 1.7 times faster than the corresponding asymmetric scheme on a typical computer

architecture. This is slightly slower than computing the Hamming distances in (13). This asymmetric scheme is interesting for any binarization scheme (LSH or anti-sparse coding) and any definition of  $A$  (either random projections or a frame).

### 3.4. Explicit reconstruction

The anti-sparse binarization scheme explicitly minimizes the reconstruction error, which is traded in (1) with the  $\ell_\infty$  regularization term. Eq. (11) gives an explicit approximation of the database vector  $\mathbf{y}$  up to a scaling factor:  $\hat{\mathbf{y}} \propto \frac{Ae(\mathbf{y})}{\|Ae(\mathbf{y})\|_2}$ . The approximate nearest neighbors  $\text{NN}_e$  are obtained by computing the exact Euclidean distances  $\|\mathbf{q} - \hat{\mathbf{y}}_i\|_2$ . This is slow compared to the Hamming distance computation. That is why, it is used to operate, like in [6], a re-ranking of the first hypotheses returned based on the Hamming distance (on the asymmetric scheme described in Sect. 3.3). The main difference with [6] is that no extra-code has to be retrieved: the reconstruction  $\hat{\mathbf{y}}$  solely relies on  $e(\mathbf{y})$ .

## 4. SIMULATIONS AND EXPERIMENTS

This section evaluates the search quality on synthetic and real data. In particular, we measure the impact of:

- The Hamming embedding technique: LSH and binarization based on anti-sparse coding. We also compare to the spectral hashing method of [5], using the code available online.
- The choice of matrix  $A$ : random projections or frame for LSH. For the anti-sparse coding, we always assume a frame.
- The search method: 1)  $\text{NN}_b$  of (13) 2)  $\text{NN}_a$  of (14) and 3)  $\text{NN}_e$  as described in Sect. 3.4.

Our comparison focuses on the case  $m \geq d$ . In the anti-sparse coding method, the regularization term  $h$  controls the trade-off between the robustness of the Hamming embedding and the quality of the reconstruction. Small values of  $h$  favors the quality of the reconstruction (without any binarization). Bigger values of  $h$  gives more components stuck to  $\|\mathbf{x}\|_\infty$ , which improves the approximation search with binary embedding. Optimally, this parameter should be adjusted to give a reasonable trade-off between the efficiency of the first stage (methods  $\text{NN}_b$  or  $\text{NN}_a$ ) and the re-ranking stage ( $\text{NN}_e$ ). Note however that, thanks to the algorithm described in Section 2, the parameter is stable, i.e., a slight modification of this parameter only affects a few components. We set  $h = 1$  in all our experiments. Two datasets are considered for the evaluation:

- A database of 10,000 16-dimensional vectors uniformly drawn on the Euclidean unit sphere (normalized Gaussian vectors) and a set of 1,000 query vectors.
- A database of SIFT [12] descriptors available online<sup>2</sup>, comprising 1 million database and 10,000 query vectors of dimensionality 128. Similar to [5], we first reduce the vector dimensionality to 48 components using principal component analysis (PCA). The vectors are not normalized after PCA.

**The comparison of LSH and anti-sparse.** Figures 1 and 2 show the performance of Hamming embeddings for synthetic data. On Fig. 1, the quality measure is the recall@10 (proportion of true NN ranked in first 10 positions) plotted as a function of the number of bits  $m$ . For LSH, observe the much better performance obtained by the proposed frame construction compared with random projections. The same conclusion holds for anti-sparse binarization.

<sup>2</sup><http://corpus-texmex.irisa.fr>

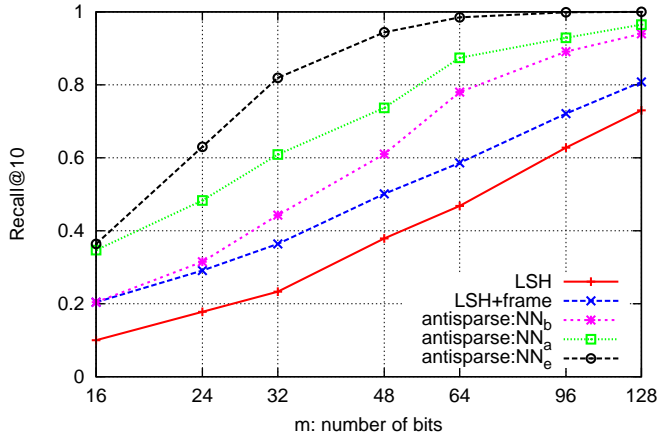


Fig. 1. Anti-sparse coding vs LSH on synthetic data. Search quality (recall@10) as a function of the number of bits of the representation.

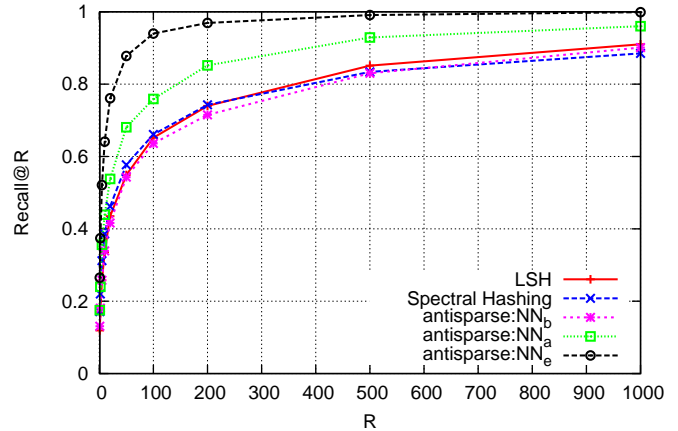


Fig. 3. Approximate search in a SIFT vector set of 1 million vectors.

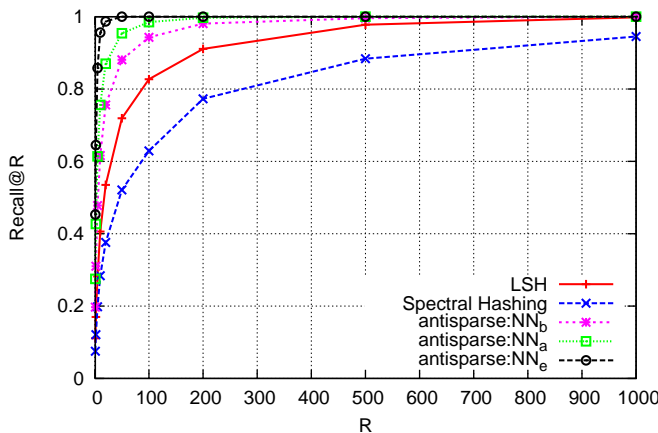


Fig. 2. Anti-sparse coding vs LSH on synthetic data ( $m = 48$ ).

The anti-sparse coding offers similar search quality as LSH for  $m = d$  when the comparison is performed using  $NN_b$  of (13). The improvement gets significant as  $m$  increases. The spectral hashing technique [5] exhibits poor performance on this synthetic dataset.

**The asymmetric comparison  $NN_a$**  leads a significant improvement, as already observed in [7, 8]. The interest of anti-sparse coding becomes obvious by considering the performance of the comparison  $NN_e$  based on the explicit reconstruction of the database vectors from their binary-coded representations. For a fixed number of bits, the improvement is huge compared to LSH. It is worth using this technique to re-rank the first hypotheses obtained by  $NN_b$  or  $NN_a$ .

**Experiments on SIFT descriptors.** As shown by Figure 3, LSH is slightly better than anti-sparse on real data when using the binary representation only (here  $m = 128$ ), which might be solved by tuning  $h$ , since the first iteration of antisparse leads to the binarization as LSH. However, the interest of the explicit reconstruction offered by  $NN_e$  is again obvious. The final search quality is significantly better than that obtained by spectral hashing [5]. Since we do not specifically handle the fact that our descriptors are not normalized after PCA, our results could probably be improved by taking care of the  $\ell_2$  norm.

## 5. CONCLUSION AND OPEN ISSUES

In this paper, we have proposed anti-sparse coding as an effective Hamming embedding, which, unlike concurrent techniques, offers an explicit reconstruction of the database vectors. To our knowledge, it outperforms all other search techniques based on binarization. There are still two open issues to take the best of the method. First, the computational cost is still a bit high for high dimensional vectors. Second, if the proposed codebook construction is better than random projections, it is not yet specifically adapted to real data.

## 6. REFERENCES

- [1] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-based music information retrieval: Current directions and future challenges," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, April 2008.
- [2] H. Jégou, M. Douze, and C. Schmid, "Improving bag-of-features for large scale image search," *IJCV*, vol. 87, no. 3, pp. 316–336, February 2010.
- [3] J. Law-To, L. Chen, A. Joly, I. Laptev, O. Buisson, V. Gouet-Brunet, N. Boujemaa, and F. Stentiford, "Video copy detection: a comparative study," in *CIVR*, New York, NY, USA, 2007, pp. 371–378, ACM.
- [4] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large databases for recognition," in *CVPR*, June 2008.
- [5] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *NIPS*, 2008.
- [6] H. Jégou, R. Tavenard, M. Douze, and L. Amsaleg, "Searching in one billion vectors: re-rank with source coding," in *ICASSP*, Prague Czech Republic, 2011.
- [7] W. Dong, M. Charikar, and K. Li, "Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces," in *SIGIR*, July 2008.
- [8] M. Jain, H. Jégou, and P. Gros, "Asymmetric hamming embedding," in *ACM Multimedia*, November 2011.
- [9] J-J. Fuchs, "Spread representations," in *ASILOMAR Conference on Signals, Systems, and Computers*, November 2011.
- [10] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Statist.*, vol. 32, no. 2, pp. 407–499, 2004.
- [11] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for near neighbor problem in high dimensions," in *Proceedings of the Symposium on the Foundations of Computer Science*, 2006, pp. 459–468.
- [12] D. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.