HASHING WITH BINARY AUTOENCODERS

Miguel Á. Carreira-Perpiñán and **Ramin Raziperchikolaei**

EECS, School of Engineering, University of California, Merced

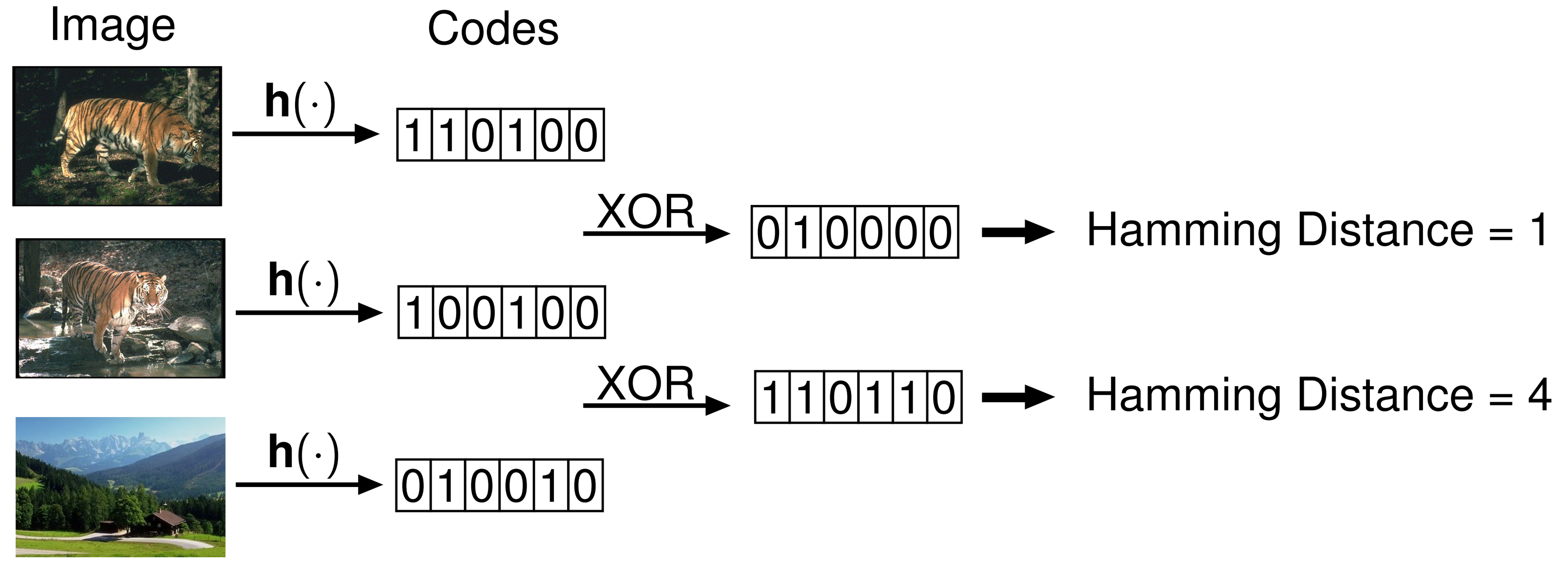
1 Binary hash functions for fast image retrieval

In K nearest neighbors problem, there are N training points in D-dimensional space (usually $D > 100$) $\mathbf{x}_i \in \mathbb{R}^D, i = 1, \dots, N$ and the goal is finding the K nearest neighbors of a query point $\mathbf{x}_q \in \mathbb{R}^D$.

- Exact search in the original space is $\mathcal{O}(ND)$ in both time and space.

A binary hash function $\mathbf{h}$ takes as input a high-dimensional vector $\mathbf{x} \in \mathbb{R}^D$ and maps it to an L-bit vector $\mathbf{z} = \mathbf{h}(\mathbf{x}) \in \{0,1\}^L$. The search is done in this low-dimensional, binary space.

- The main goal is preserving the neighborhood, i.e., assign (dis)similar codes to (dis)similar patterns.



Finding K nearest neighbors in Hamming space is more efficient:

- Both time and space complexities would be $\mathcal{O}(NL)$ instead of $\mathcal{O}(ND)$.
- Hamming Distance can be computed very efficiently using hardware operations.

Suppose that $N = 10^9$, $D = 500$ and $L = 64$

Search in	Space	Time
original space	2 TB	1 hour
Hamming space	8 GB	10 seconds

2 Previous works on binary hashing

Optimizing the objective functions that have been used in dimensionality reduction algorithms is difficult because the codes are binary. Most of the hashing methods use a suboptimal, "filter" approach:

1. Relax the binary constraints and solve a continuous problem.
2. Binarize the continuous codes by finding a threshold or a rotation matrix.
3. Fit L classifiers to (patterns $\mathbf{x}$,codes $\mathbf{z}$) to obtain the hash function $\mathbf{h}$.

We seek an optimal, "wrapper" approach: optimize the objective function jointly over linear mappings and thresholds, respecting the binary constraints while learning $\mathbf{h}$.

3 Our hashing model: Binary Autoencoder

We consider binary autoencoders as our hashing model:

$$E_{\mathrm{BA}}(\mathbf{h},\mathbf{f}) = \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{f}(\mathbf{h}(\mathbf{x}_n))\|^2 \quad \text{s.t.} \quad \mathbf{h}(\mathbf{x}_n) \in \{0,1\}^L.$$

- The encoder $\mathbf{h}\colon \mathbf{x} \rightarrow \mathbf{z}$ maps a real vector $\mathbf{x} \in \mathbb{R}^D$ onto a low-dimensional binary vector $\mathbf{z} \in \{0,1\}^L$ (with $L < D$).
- The decoder $\mathbf{f}\colon \mathbf{z} \rightarrow \mathbf{x}$ maps $\mathbf{z}$ back to $\mathbb{R}^D$ in an effort to reconstruct $\mathbf{x}$.

We use the method of auxiliary coordinates (MAC), a generic approach to optimize nested functions. First, we convert the problem for $E_{\mathrm{BA}}(\mathbf{h},\mathbf{f})$ into an equivalent constrained problem:

$$\min_{\mathbf{h},\mathbf{f},\mathbf{Z}} \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{f}(\mathbf{z}_n)\|^2 \quad \text{s.t.} \quad \begin{array}{l} \mathbf{z}_n = \mathbf{h}(\mathbf{x}_n) \in \{0,1\}^L \\ n = 1,\dots,N. \end{array}$$

that is not nested, where $\mathbf{z}_n$ are the auxiliary coordinates for the output of $\mathbf{h}(\mathbf{x}_n)$. Now we apply the quadratic-penalty method:

$$E_Q(\mathbf{h},\mathbf{f},\mathbf{Z};\mu) = \sum_{n=1}^{N} \left(\|\mathbf{x}_n - \mathbf{f}(\mathbf{z}_n)\|^2 + \mu \|\mathbf{z}_n - \mathbf{h}(\mathbf{x}_n)\|^2 \right)$$

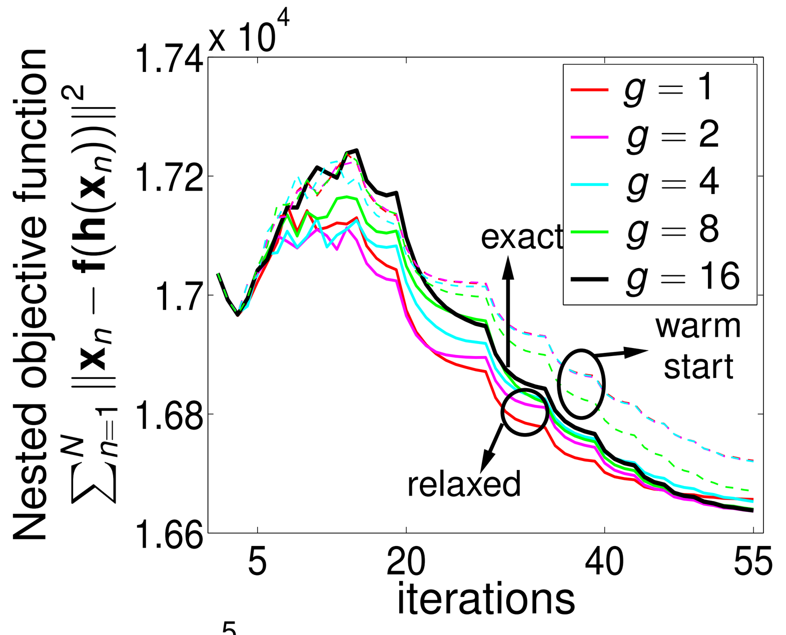
$$\text{s.t. } \mathbf{z}_n \in \{0,1\}^L, \; n = 1,\dots,N$$

where we start with a small μ and increase it slowly. To optimize E_Q we apply alternating optimization:

- Over $\mathbf{f}$ for fixed $\mathbf{Z}$: $\sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{f}(\mathbf{z}_n)\|^2$. With a linear decoder this is a straightforward linear regression with data $(\mathbf{Z},\mathbf{X})$.
- Over $\mathbf{h}$ for fixed $\mathbf{Z}$: $\min_{\mathbf{h}} \sum_{n=1}^{N} \|\mathbf{z}_n - \mathbf{h}(\mathbf{x}_n)\|^2$. This separates for each bit $l = 1 \dots L$. The subproblem for each bit is a binary classification problem with data $(\mathbf{X},\mathbf{Z}_{\cdot l})$.
- Over $\mathbf{Z}$ for fixed $(\mathbf{h},\mathbf{f})$: $\min_{\mathbf{z}_n} e(\mathbf{z}_n) = \|\mathbf{x} - \mathbf{f}(\mathbf{z}_n)\|^2 + \mu \|\mathbf{z}_n - \mathbf{h}(\mathbf{x})\|^2$. This is a binary optimization on NL variables, but it separates into N independent optimizations each on only L variables. With $L \lesssim 16$ we can afford an exhaustive search and for larger L, we use alternating optimization.

Advantages of optimizing BA using MAC: It respects the binary constraints and introduces significant parallelism in optimization. Furthermore, the individual steps in alternating optimization are (reasonably) easy to solve.

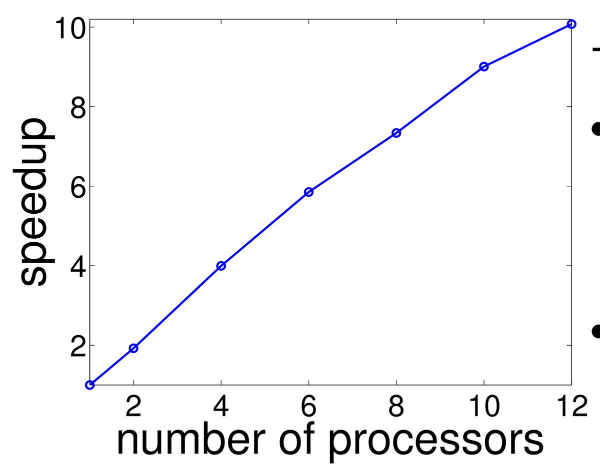
4 Experiments



Two approaches to initialize $\mathbf{z}_n$ in the $\mathbf{Z}$ step:

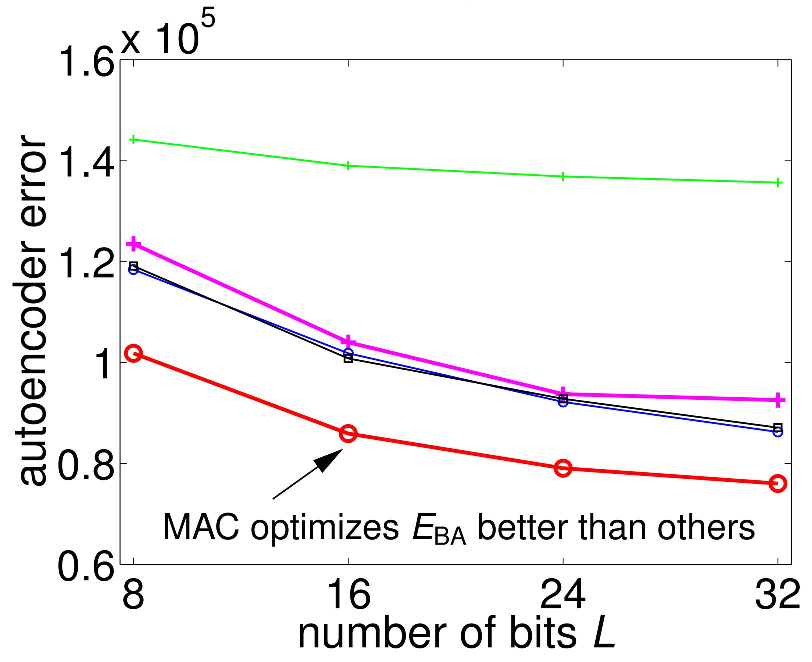
- Warm start: Initialize $\mathbf{z}_n$ to the code found in the previous iteration's $\mathbf{Z}$ step.
- Solve the relaxed problem on $\mathbf{z}_n \in [0,1]^L$ and then truncate it.

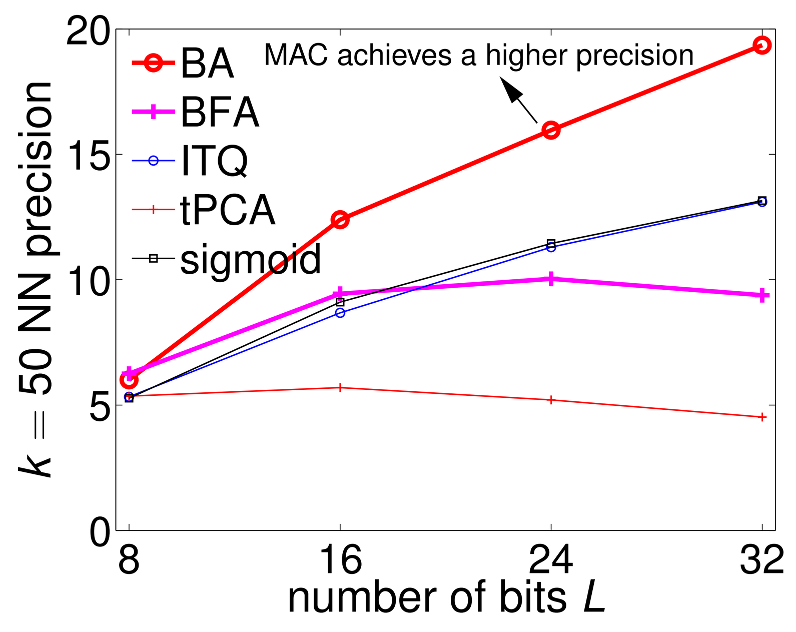
The latter achieves better local optima than using warm starts.



The algorithm is highly parallel:

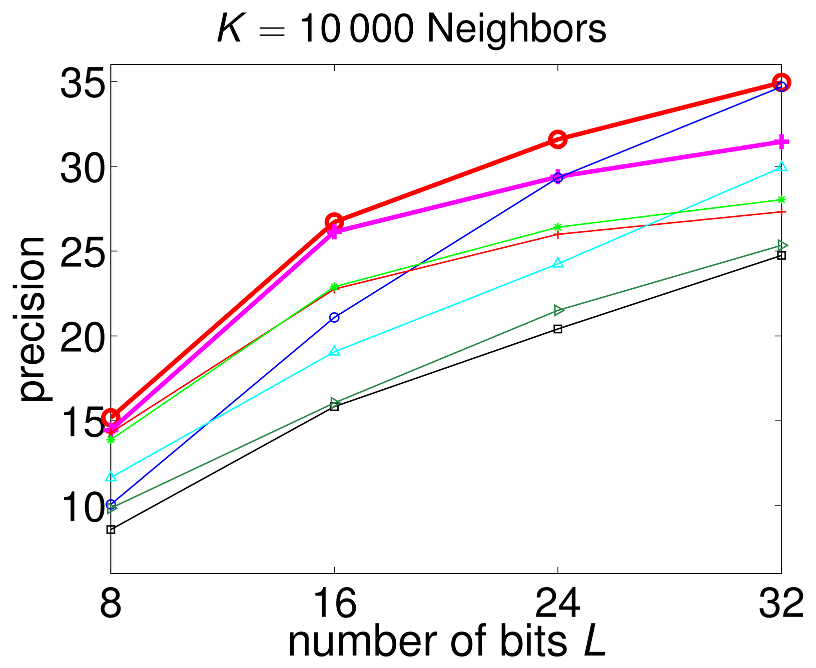
- For fixed $\mathbf{Z}$ we have $L+1$ independent problems for each of the L single-bit hash functions, and for $\mathbf{f}$.
- For fixed $\mathbf{h}$ and $\mathbf{f}$ we have N independent optimization problems each over L variables.

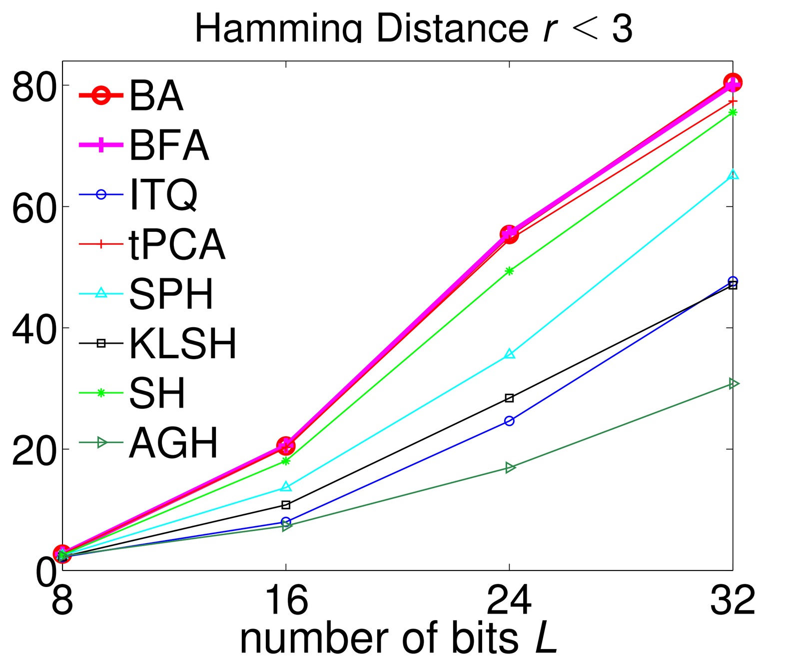




NUS-WIDE-LITE dataset, $N = 27\,807$ training/ 27 808 test images. Comparison between the methods that minimize the binary autoencoder objective function

BA achieves lower reconstruction error and also better precision/recall using MAC than using a suboptimal optimization as in tPCA (truncates codes at zero), ITQ (finds the best rotation matrix), and sigmoid (relaxes the step function to a sigmoid in training using backpropagation).





SIFT1M dataset, $N = 1\,000\,000$ training/ 10 000 test images.

We compare our proposed method (BA) with other state-of-the art methods. To report precision, we consider groundtruth $K = 10\,000$ neighbors and set of retrieved neighbors for $k = 10\,000$ and $r \leq 3$. Generally BA beats all other methods.

We compare our BA that uses a linear hash function and simply minimizes the reconstruction error with several hashing methods that learn nonlinear hash functions and use more sophisticated error functions . Results show that BA outperforms other methods, often by a large margin.

Acknowledgments

Work supported by NSF award IIS1423515 and UC Merced's School of Engineering.