HASHING WITH BINARY AUTOENCODERS

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

EECS, School of Engineering, University of California, Merced

BayLearn
Bay Area Machine Learning Symposium

1 Abstract

An attractive approach for fast search in image databases is binary hashing, where each high-dimensional, real-valued image is mapped onto a low-dimensional, binary vector and the search is done in this binary space. Finding the optimal hash function is difficult because it involves binary constraints, and most approaches approximate the optimization by relaxing the constraints and then binarizing the result. Here, we focus on the binary autoencoder model, which seeks to reconstruct an image from the binary code produced by the hash function. We show that the optimization can be simplified with the method of auxiliary coordinates. This reformulates the optimization as alternating two easier steps: one that learns the encoder and decoder separately, and one that optimizes the code for each image. Image retrieval experiments show the resulting hash function outperforms or is competitive with state-of-the-art methods for binary hashing.

Funded by NSF award IIS–1423515.

2 Binary hash functions

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.
- Hamming distance computed using XOR and then counting.
- Time complexity is $\mathcal{O}(NL)$ instead of $\mathcal{O}(ND)$ with smaller constants because of efficient hardware operations with bits.
- Space complexity is $\mathcal{O}(NL)$ instead of $\mathcal{O}(ND)$. If we have $N = 1\,000\,000, D = 300$ and $L = 32$, we need 4 MB instead of 1.2 GB memory to store the data.

3 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 methods use a suboptimal, "filter" approach to find the binary codes:

1. Relax the binary constraints and solve a continuous problem.
2. Binarize the continuous codes using approaches such as:
   - Truncate the real values using threshold zero.
   - Find the best threshold for truncation.
   - Rotate the real vectors to minimize the quantization loss.
3. Fit a mapping 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 $\mathbf{h}$.

4 Our hashing model: Binary Autoencoder

We consider binary autoencoders as our hashing model:

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

- The encoder $\mathbf{h}\colon \mathbf{x} \to \mathbf{z}$ 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 $\mathbf{f}\colon \mathbf{z} \to \mathbf{x}$ maps $\mathbf{z}$ back to $\mathbb{R}^D$ in an effort to reconstruct $\mathbf{x}$.

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

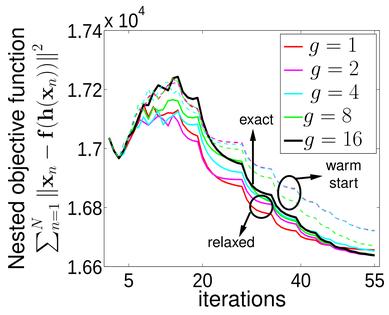
$$\min_{\mathbf{h},\mathbf{f},\mathbf{Z}} \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{f}(\mathbf{z}_n)\|^2 \quad \text{s.t.} \quad \mathbf{z}_n = \mathbf{h}(\mathbf{x}_n),\ \mathbf{z}_n \in \{0,1\}^L,\ n = 1, \dots, N$$

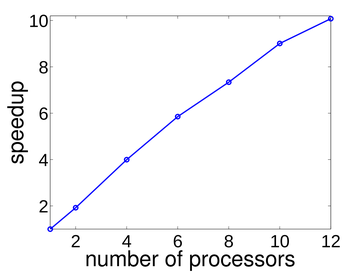
that is not nested, where $\mathbf{z}_n$ are the auxiliary coordinates for the output of $\mathbf{h}(x_n)$. Now we apply the quadratic-penalty method:

$$E_Q(\mathbf{h}, \mathbf{f}, \mathbf{Z}; \mu) = \sum_{n=1}^{N} \left(\|\mathbf{x}_n - \mathbf{f}(\mathbf{z}_n)\|^2 + \mu \|\mathbf{z}_n - \mathbf{h}(\mathbf{x}_n)\|^2 \right) \text{ s.t. } \mathbf{z}_n \in \{0,1\}^L,\ n = 1, \dots, N$$

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 $(\mathbf{X}, \mathbf{Z}_{:l})$ using the number of misclassified patterns as loss function.
- 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 over the 2^L codes. For larger L, we use alternating optimization over groups of g bits.





We have used the following 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. Using small g (≈ 1) is fastest and gives good optima.

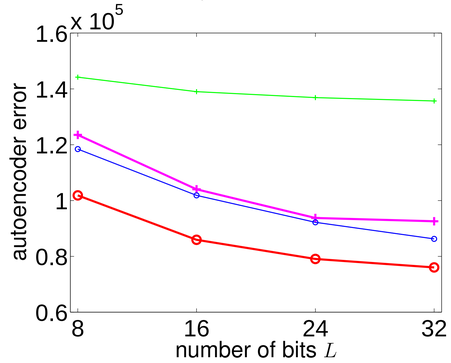
The algorithm is highly parallel:

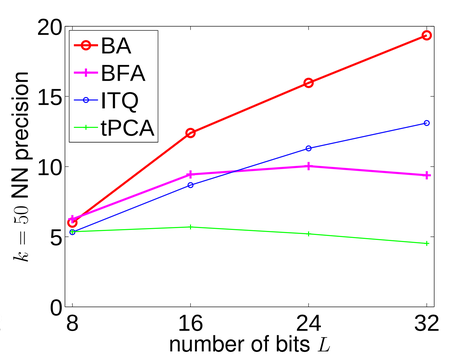
- For fixed $\mathbf{Z}$ we have $L + 1$ independent problems for each of the L single-bit hash functions, and for $\mathbf{f}$.
- For fixed $\mathbf{h}$ and $\mathbf{f}$ we have N independent optimization problems each over L variables.

5 Experiments

1. Optimizing Binary Autoencoders Improves Precision

NUS-WIDE-LITE dataset, $N = 27\,807$ training/ $27\,808$ test images.
BA achieves lower reconstruction error and also better precision/recall using MAC than using a suboptimal, "filter" optimization as in ITQ (which first estimates continuous codes, then binarizes them, then fits the hash function to them).





2. Comparison with other hashing algorithms

NUS-WIDE dataset: $269\,648$ high resolution color images in 81 concepts; training/test $N = 161\,789/107\,859$, $D = 128$ wavelet features.
A well-optimized (using MAC) binary autoencoder with a linear hash function consistently beats state-of-the-art methods using other objectives/(nonlinear) hash functions.