Hashing with Binary Autoencoders

Miguel Á. Carreira-Perpiñán, Ramin Raziperchikolaei
EECS, University of California, Merced.

Introduction. We consider the problem of binary hashing, where given a high-dimensional vector $\mathbf{x} \in \mathbb{R}^D$, we want to map it to an L-bit vector $\mathbf{z} = \mathbf{h}(\mathbf{x}) \in \{0,1\}^L$ using a hash function $\mathbf{h}$, while preserving the neighbors of $\mathbf{x}$ in the binary space. Binary hashing has emerged in recent years as an effective technique for fast search on image (and other) databases. While the search in the original space would cost $\mathcal{O}(ND)$ in both time and space, using floating point operations, the search in the binary space costs $\mathcal{O}(NL)$ where $L \ll D$ and the constant factor is much smaller. This is because the hardware can compute binary operations very efficiently and the entire dataset (NL bits) can fit in the main memory of a workstation.

Many different hashing approaches have been proposed in the last few years. They formulate an objective function of the hash function $\mathbf{h}$ or of the binary codes that tries to capture some notion of neighborhood preservation. All these approaches have in common two things: $\mathbf{h}$ performs dimensionality reduction ($L < D$) and, as noted, it outputs binary codes ($\mathbf{h}: \mathbb{R}^D \to \{0,1\}^L$). The latter implies a step function or binarization applied to a real-valued function of the input $\mathbf{x}$. Optimizing this is difficult. In practice, most approaches follow a **two-step procedure**: first they learn a real hash function ignoring the binary constraints and then the output of the resulting hash function is binarized. This procedure can be seen as a "**filter**" approach and is suboptimal. To obtain the optimal solution, we must optimize the objective function jointly over mappings and thresholds, respecting the binary constraints while learning $\mathbf{h}$; this is a "**wrapper**" approach. In this paper we show that this joint optimization can actually be carried out reasonably efficiently.

Our hashing models. We consider a well-known model for continuous dimensionality reduction, the (continuous) autoencoder, defined in a broad sense as the composition of an encoder $\mathbf{h}(\mathbf{x})$ which maps a real vector $\mathbf{x} \in \mathbb{R}^D$ onto a real code vector $\mathbf{z} \in \mathbb{R}^L$ (with $L < D$), and a decoder $\mathbf{f}(\mathbf{z})$ which maps $\mathbf{z}$ back to $\mathbb{R}^D$ in an effort to reconstruct $\mathbf{x}$. For hashing, the encoder maps continuous inputs onto *binary* code vectors with L bits, $\mathbf{z} \in \{0,1\}^L$, and we call it a ***binary autoencoder (BA)***. Our desired hash function will be the encoder $\mathbf{h}$, and it should minimize the following function, given a dataset of high-dimensional patterns $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$:

$$E_{\text{BA}}(\mathbf{h}, \mathbf{f}) = \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{f}(\mathbf{h}(\mathbf{x}_n))\|^2$$

which is the usual least-squares error but where the code layer is binary. We will also consider a related model:

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

where $\mathbf{f}$ is linear and we optimize over the decoder $\mathbf{f}$ and the binary codes $\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_N)$ of each input pattern. We call this model (least-squares) *binary factor analysis (BFA)*. A hash function $\mathbf{h}$ can be obtained from BFA by fitting a binary classifier of the inputs to each of the L code bits. It is a filter approach, since it first learns $\mathbf{Z}$ and then $\mathbf{h}$, while the BA is an optimal (wrapper) approach, since it optimizes jointly over $\mathbf{f}$ and $\mathbf{h}$.

We use the recently proposed ***method of auxiliary coordinates (MAC)*** for optimization of BA and BFA. The idea is to break nested functional relationships judiciously by introducing variables as equality constraints, turning them into penalties and applying alternating optimization. We introduce as auxiliary coordinates the outputs of $\mathbf{h}$, i.e., the codes for each of the N input patterns, and obtain the following equality-constrained problem:

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

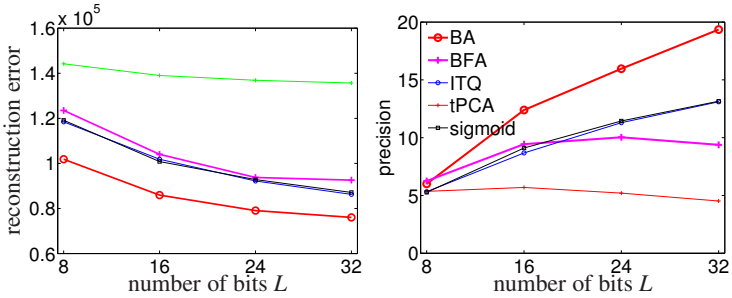

Figure 1: Wrapper vs filter optimization. *Left*: BA objective function. *Right*: precision returning $k = 50$ nearest neighbors.

Note the codes are binary. We now apply the quadratic-penalty method and minimize the following objective function while progressively increasing μ, so the constraints are eventually satisfied:

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

Now we apply alternating optimization over $\mathbf{Z}$ and $(\mathbf{h}, \mathbf{f})$. This results in the following two steps:

- Over $\mathbf{Z}$ for fixed $(\mathbf{h}, \mathbf{f})$, the problem separates for each of the N codes. The optimal code vector for pattern $\mathbf{x}_n$ tries to be close to the prediction $\mathbf{h}(\mathbf{x}_n)$ while reconstructing $\mathbf{x}_n$ well. This binary optimization has the form of a binary proximal operators over few variables (L), so it can be solved exactly by enumeration, or approximately by alternating optimization.
- Over $(\mathbf{h}, \mathbf{f})$ for fixed $\mathbf{Z}$, we obtain $L + 1$ independent problems for each of the L single-bit hash functions and for $\mathbf{f}$.

We can now see the advantage of the auxiliary coordinates: the individual steps are (reasonably) easy to solve, and besides they exhibit significant parallelism. The resulting algorithm alternates a step over the encoder (L classifications) and decoder (one regression) with a step over the codes (N binary proximal operators).

Experiments. First, we investigate the **speedup** that can be achieved in training of BA. We use the Matlab Parallel Processing Toolbox with up to 12 processors and simply replace "for" with "parfor" loops so each iteration (over points in the $\mathbf{Z}$ step, over bits in the $\mathbf{h}$ step) is run in a different processor. We observe a **nearly perfect scaling** in training of BA.

Second, We focus purely on the **BA objective function** (reconstruction error) and study the gain obtained by the MAC optimization, which **respects the binary constraints**, over the suboptimal, "filter" approaches. We compared BA with four relevant methods: tPCA relaxes the constraints (i.e., PCA) and then binarizes the result by thresholding at 0, ITQ finds the optimal rotation before truncating the result of PCA, and sigmoid trains the autoencoder with backpropagation where the sign function is replaced by sigmoid. We train different methods on the NUS-WIDE-LITE dataset, which contains $N = 27\,807$ images for training and $27\,808$ images for test. Fig. 1 shows that **BA dominates all other methods in reconstruction error, as expected, and also in precision**. This demonstrates that a better optimization of the objective significantly improves the hash function learned.

Finally, We compare BA with state-of-the-art methods like ITQ, AGH, SH, SPH and KLSH on large image datasets using $b = 8$ to 32 bits. It is known that the retrieval performance of a given algorithm depends strongly on the size of the neighbor set used, so we report experiments with **small and large number of points in the ground truth set**. Results show that BA outperforms other methods, often by a large margin.

This is an extended abstract. The full paper is available at the Computer Vision Foundation webpage.