LEARNING SUPERVISED BINARY HASHING: OPTIMIZATION VS DIVERSITY

Ramin Raziperchikolaei and **Miguel Á. Carreira-Perpiñán**, EECS, UC Merced

1 Abstract

Binary hashing is a practical approach for fast, approximate retrieval in large image databases. The goal is to learn a hash function that maps images onto a binary vector such that Hamming distances approximate semantic similarities. The search is then fast by using hardware support for binary operations. Most hashing papers define a complicated objective function that couples the single-bit hash functions. A recent work has shown the surprising result that by learning the single-bit functions independently and making them diverse using ensemble learning techniques, one can achieve simpler optimization, faster training, and better retrieval results. In this paper, we study the interplay between optimization and diversity in learning good hash functions. We show that to achieve good hash functions, no matter how we optimize the objective, the diversity among the single-bit hash functions is a crucial element.

Work supported by NSF award IIS–1423515

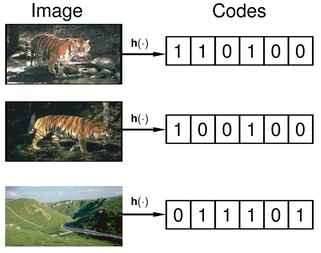
2 Binary hash functions for fast image retrieval

In K nearest neighbors problem, there are N training points in D-dimensional space (usually $D > 100$) $\mathbf{x}_i \in \mathbb{R}^D, i = 1, \dots, N$. The goal is to find 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 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 b-bit vector $\mathbf{z} = \mathbf{h}(\mathbf{x}) \in \{0,1\}^b$.

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

In supervised hashing, we try to preserve the semantic similarity between the images (e.g. images from different view points are similar, while they are far in the Euclidean space).



Finding K nearest neighbors in Hamming space needs $\mathcal{O}(Nb)$ in time and space. Distances can be computed efficiently using hardware operations.

$N = 10^9$, $D = 500$ and $b = 64$

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

3 Affinity-based objective functions

Most hashing papers try to minimize an affinity-based objective function, which directly tries to preserve the original similarities in the binary space:

$$\min \mathcal{L}(\mathbf{h}) = \sum_{n,m=1}^{N} L(\mathbf{h}(\mathbf{x}_n), \mathbf{h}(\mathbf{x}_m);\ y_{nm})$$

where $\mathbf{x}_i \in \mathbb{R}^D$ is the ith input data, $\mathbf{h}$ is the parameters of the hash function, $L(\cdot)$ is a loss function that compares the codes for two images with the ground-truth value y_{nm} that measures the affinity in the original space between the two images $\mathbf{x}_n$ and $\mathbf{x}_m$. Many such loss functions $L(\mathbf{z}_n, \mathbf{z}_m;\ y_{nm})$ exist, e.g.:

KSH: $(\mathbf{z}_n^T \mathbf{z}_m - b y_{nm})^2$ Laplacian: $(y_{nm} \|\mathbf{z}_n - \mathbf{z}_m\|^2)$ BRE: $(\frac{1}{b}\|\mathbf{z}_n - \mathbf{z}_m\|^2 - y_{nm})^2$

4 Ensemble-based approach in binary hashing

Optimizing $\mathcal{L}(\mathbf{h})$ is difficult because $\mathbf{h}$ is discrete. Many optimization-based methods have been considered in the binary hashing literature.

Limitations of the optimization-based methods:

- The hash function outputs binary values, hence the problem is nonconvex and nonsmooth. The underlying problem of finding the binary codes for the points is an NP-complete optimization over Nb variables.
- Most methods do not scale beyond a few thousand training points.
- The b single-bit functions are coupled (to avoid trivial solutions where all codes are the same).
- In the end, there is little practical difference between the different objective functions and optimization algorithms proposed.

Rather than coupling the b hash functions into a single objective function, a recent method, Independent Laplacian Hashing (ILH) (Carreira-Perpiñán and Raziperchikolaei, NIPS 2016), proposed to train each hash function independently from each other.

To get good retrieval results, the single-bit hash functions have to be different from each other. ILH uses the ensemble learning techniques to make the hash functions different from each other:

- Different initializations (ILHi): Each hash function is initialized randomly, which results in different local optima for different hash functions.
- Different training sets (ILHt): Each hash function uses a training set of N points that is different from the other hash functions.
- Different feature subsets (ILHf): Each hash function is trained on a random subset of $1 \le d \le D$ features.

In ILH, the best results were achieved when different training subsets are used. We consider this mechanism to make the functions diverse.

ILH minimizes $\mathcal{L}(\mathbf{h})$ over a single-bit hash function which gives the following optimization problem:

$$\min_{\mathbf{h}} P(\mathbf{h}) = \mathbf{h}(\mathbf{X})\,\mathbf{Y}\,\mathbf{h}(\mathbf{X})^T = \sum_{n,m=1}^{N} y_{nm}\, h(\mathbf{x}_n)\, h(\mathbf{x}_m) \quad \text{s.t.} \quad h(\mathbf{x}_n) = \sqcap(\mathbf{w}^T \mathbf{x}_n)$$

where $\mathbf{h}(\mathbf{X}) = (h(\mathbf{x}_1), \dots, h(\mathbf{x}_N)) \in \{-1,+1\}^N$ is a row vector of N bits and $\sqcap(t) = +1$ if $t \ge 0$ and -1 if $t < 0$.

The ensemble-based approach proposed by ILH gives several advantages:

- In terms of retrieval performance (precision/recall), the ensemble-based approach is better or comparable to optimization-based methods.
- Much simpler optimization: ILH deals with b independent problems each over N binary codes rather than 1 problem with Nb binary codes.
- Hence, faster training and better accuracy, because ILH deals with optimization problems of a smaller size.
- Training the b functions can be parallelized: this helps to learn a large number of single-bit functions very fast.

In the paper, we investigate the role of diversity vs optimization to see which one matters more.

6 Experiments

Our main goal is to find out the importance of diversity vs optimization in preserving neighborhoods and making the hash functions differ. In the paper, we design several experiments to investigate this:

- We train the 1-bit hash functions on different training subsets. We show that diversity is crucial in combining the single bit functions and achieving a reasonable retrieval performance.
- We train t-bit hash functions (instead of the 1-bit ones) on different training subsets and combine them to learn the final b-bit hash function. We repeat this experiment for different objective functions. We show that we may need both optimization technique and independent sets to make the hash functions diverse.

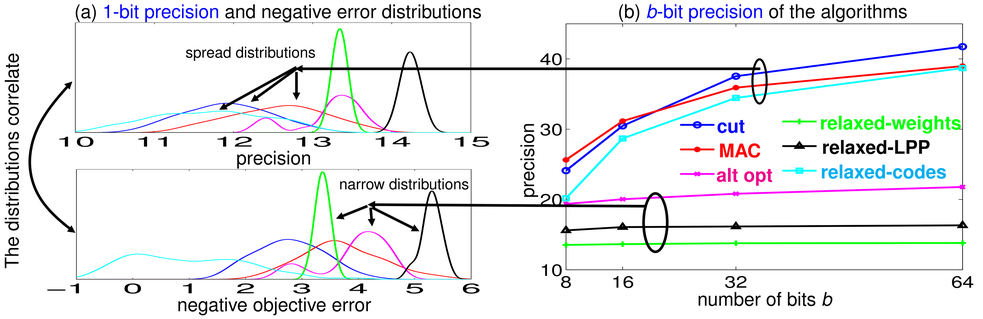
We conclude that while both optimization and diversity are important, diversity is crucial to get significant improvement as we use groups of bits.

Optimization vs diversity, which one is more important?

we investigate the importance of diversity by experimentally controlling the quality of the optimization of the 1-bit objective function with different algorithms and observing its effect on the learned b-bit hash function.

To optimize the 1-bit objective, we use the following optimization algorithms:

1. **cut**. It first optimizes the objective over the binary codes and then fits the hash function to these codes.
2. **MAC**. It optimizes the objective over the codes and the hash function in alternation, which gives better optimization results than the two-step approach.
3. **alt opt**. It optimizes each element of the weight vector in an alternation, while the rest of them are fixed (coordinate descent).
4. **relaxed-weights**. It ignores the sign function of the $\mathbf{h}$, adds $\mathbf{w}^T\mathbf{w} = 1$ constraint, and solves an eigenproblem.
5. **relaxed-LPP**. Same as the relaxed-weights, except that it uses $\mathbf{w}^T(\mathbf{X}\mathbf{X}^T)\mathbf{w} = 1$ as the constraint.
6. **relaxed-codes**. It relaxes the constraint $\mathbf{z} \in \{0,1\}^N$ to the $\mathbf{z}\mathbf{z}^T = N$ and solves an eigenproblem to learn $\mathbf{z}$.



(a) We optimize the 1-bit objective function on 64 training subsets of 5000 points of CIFAR dataset, using different optimization algorithms. We plot the distribution of the (negative) objective function error (bottom) and the distribution of the precision (top) of the 1-bit hash functions for each of the algorithms. By decreasing the value of the objective function (better optimization), we achieve a better precision.

(b) We put the 1-bit hash functions together, create groups of 8 to 64 bits and report their precisions. For a b-bit hash function, a better 1-bit optimization only improves the precision significantly if there is sufficient diversity: the more spread the error and precision distributions, the higher the diversity, and the better combined precision.