Learning Supervised Binary Hashing Without Binary Code Optimization

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

1 Abstract

The goal of binary hashing is to learn a hash function that maps high-dimensional points to bit codes, which can be used to speed up search on large databases. Most papers use optimization approaches based on a suitable objective function with a difficult and inexact optimization. Recently, it has been shown that the hash function for a code bit may be learned independently from that of the other code bits. One simply optimizes a single-bit objective function defined on a random data sample, and then fits a binary classifier to the resulting codes. We show that it is even possible to dispense with the single-bit optimization, by assigning binary codes to the points based on their similarity to a randomly chosen seed point. This procedure is very simple, scalable, and is competitive with the state-of-the-art methods in retrieval metrics.

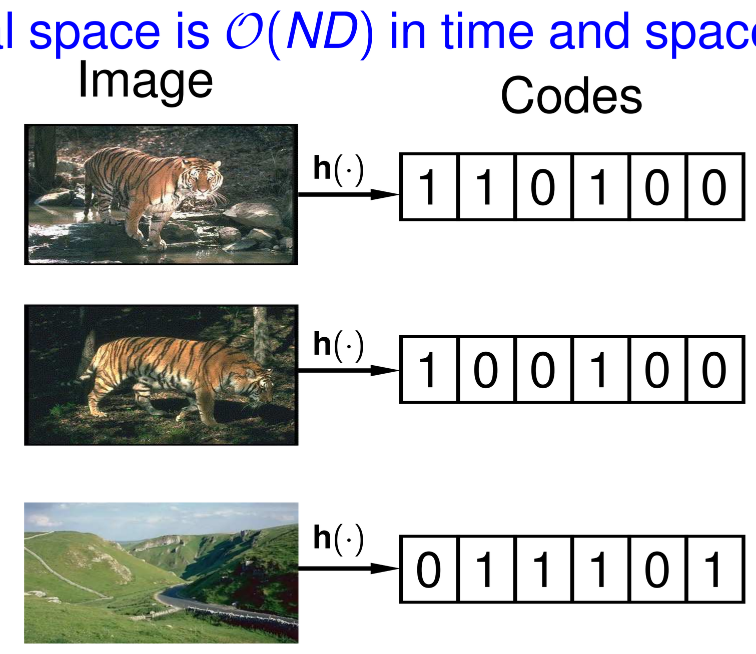
Work supported by NSF award IIS–1423515

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 $\mathbf{x}_q \in \mathbb{R}^D$. 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 binary 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 Previous works: learning the hash function

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

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

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.:

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

Optimizing $\mathcal{L}(\mathbf{h})$ is difficult because $\mathbf{h}$ is discrete.

Optimization-based approach: Many optimization-based methods have been proposed to optimize the objective approximately over the b-bit hash function. These methods have several limitations:

- The hash function outputs binary values, hence the problem is nonconvex and nonsmooth. An NP-complete problem over Nb variables has to be solved.
- They do not scale beyond a few thousand training points.
- In the end, there is little practical difference between the different objective functions and optimization algorithms proposed.

Ensemble-based approach: Rather than coupling the b hash functions, 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, like using different training subsets or different initializations for each single-bit hash function, to make the hash functions different from each other.

ILH minimizes the objective $\mathcal{L}(\cdot)$ over a single-bit hash function h:

$$\min_{\mathbf{h}} P(\mathbf{h}) = \mathbf{h}(\mathbf{X})\,\mathbf{Y}\,\mathbf{h}(\mathbf{X})^T = \sum_{n,m=1}^{N} y_{nm}\, h(\mathbf{x}_n)\, h(\mathbf{x}_m)$$

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, $h(\mathbf{x}_n) = \mathcal{J}(\mathbf{w}^T\mathbf{x}_n)$, and $\mathcal{J}(t) = +1$ if $t \geq 0$ and -1 if $t < 0$.

4 Advantages of the ensemble diversity approach

The ensemble-based approach gives several advantages:

- It is better or comparable to the optimization-based methods in terms of retrieval performance.
- 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.

In this paper we show that by guessing the binary codes it is even possible to dispense with the single-bit optimization.

5 Independent Supervised Hashing (ISH)

While in the b-bit case the binary code space can have 2^b different codes, with $b = 1$ there are just two possible codes, $+1$ and -1.

The goal of the single-bit objective function: if image $\mathbf{x}_n$ is similar (dissimilar) to the image $\mathbf{x}_m$ then ideally $\mathbf{x}_n$ and $\mathbf{x}_m$ should have the same (different) binary code(s).

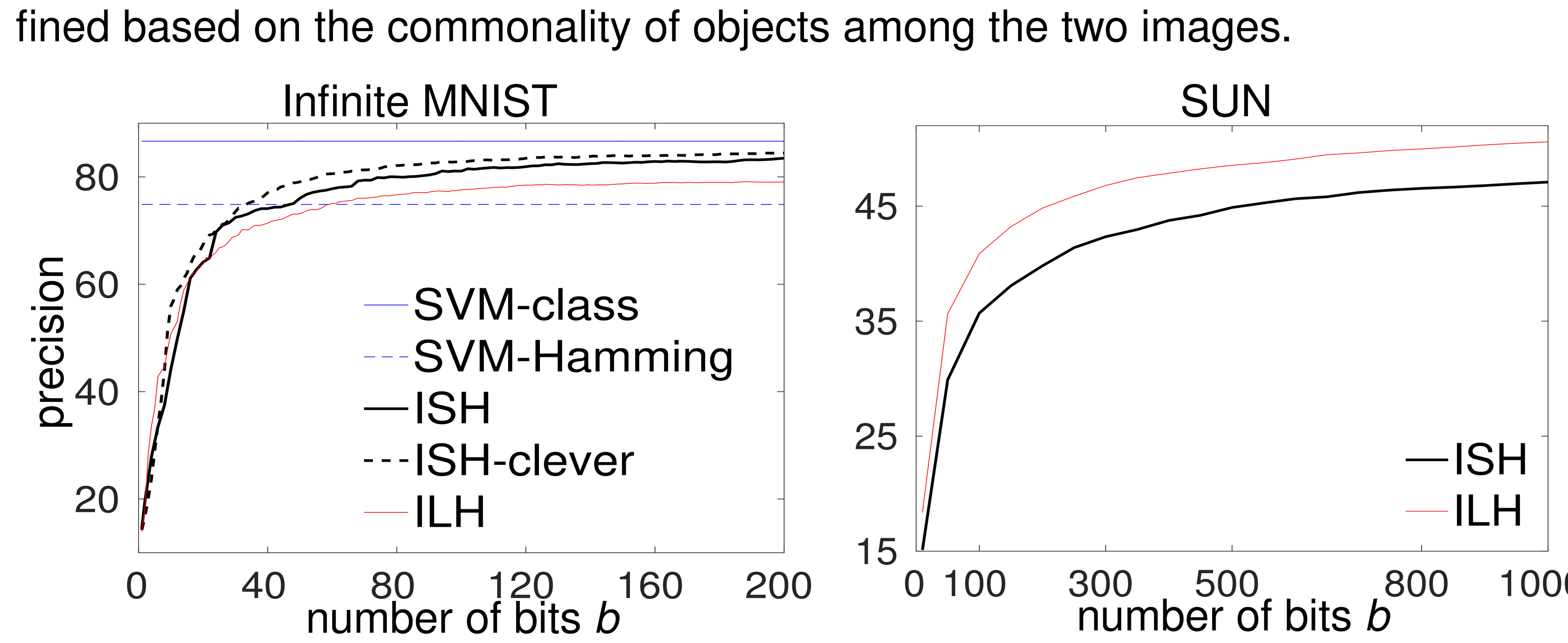
ISH first picks a point $\mathbf{x}_n$ (the "seed") and finds a sample $\mathcal{S}_+$ of points that are similar to $\mathbf{x}_n$ ($y_{nm} > 0$) and a sample $\mathcal{S}_-$ of points that are dissimilar to $\mathbf{x}_n$ ($y_{nm} < 0$). This defines a two-class problem on the training set $\mathcal{S}_+ \cup \mathcal{S}_-$, on which ISH trains a linear SVM or any other classifier to use as single-bit hash function.

Advantages of the Independent Supervised Hashing (ISH):

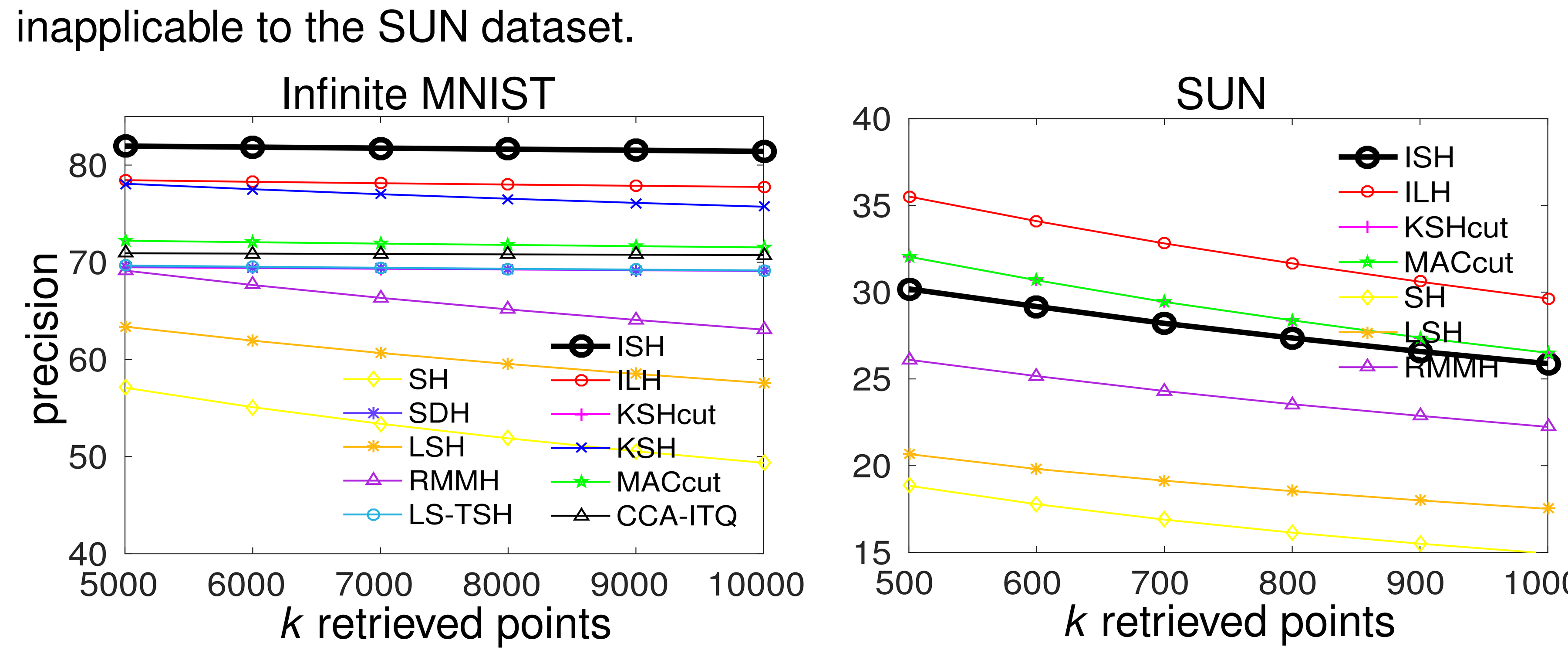
- It is very simple, requiring only the similarities and training binary classifiers.
- Like ILH, ISH is embarrassingly parallel over the b bits, and suitable for implementation in a distributed-data setting.
- It is faster than ILH, because it eliminates the NP-complete optimization of the single-bit objective function, and much faster than approaches that optimize over the b bits for the entire dataset jointly.
- It scales to bigger datasets, as big as long as we are able to train a binary classifier on them, while ILH is limited because of the NP-complete problem.
- ISH learns hash functions comparable to the state-of-the-art. The precision of ISH consistently increases as more bits are added over a range of b values.

6 Experiments

Infinite MNIST contains 1 000 000/2 000 images for training/test, in 10 classes. The groundtrurh is defined based on the labels of the images.
Sun dataset contains 23 537/2 000 images as training/test. The groundtruth is defined based on the commonality of objects among the two images.



We report precision as a function of number of bits. ISH-clever selects seeds by cycling over the C classes in the labeled datasets and achieves better results than randomly selecting the seeds (done by ISH). SVM-class trains C one-vs-all classifiers and reports the classification accuracy. When the ground-truth is given by the class label, SVM-class gives better precision than the hashing methods, but is inapplicable to the SUN dataset.



ILH and ISH outperform optimization-based methods. They are comparable in different datasets.