An Ensemble Diversity Approach to Binary Hashing

Miguel Á. Carreira-Perpiñán **Ramin Raziperchikolaei**
EECS, University of California, Merced
http://eecs.ucmerced.edu

Introduction Information retrieval tasks such as searching for a query image or document in a database are essentially a nearest-neighbor search. When the dimensionality of the query and the size of the database is large, approximate search is necessary. We focus on binary hashing, where the query and database are mapped onto low-dimensional binary vectors, where the search is performed. This has two speedups: computing Hamming distances (with hardware support) is much faster than computing distances between high-dimensional floating-point vectors; and the entire database becomes much smaller, so it may reside in fast memory rather than disk (for example, a database of 1 billion real vectors of dimension 500 takes 2 TB in floating point but 8 GB as 64-bit codes).

Constructing hash functions that do well in retrieval measures such as precision and recall is usually done by optimizing an affinity-based objective function that relates Hamming distances to neighborhood information in a training set. Many such objective functions have the form of a sum of pairwise terms that indicate whether the training points $\mathbf{x}_n$ and $\mathbf{x}_m$ are neighbors. The output of the hash function is binary, which makes the objective function nonsmooth (implicitly discrete) and difficult to optimize. Furthermore, there is a large number of binary variables which are coupled in the objective function or the constraints. Most papers find the binary codes by relaxing the codes to real values, optimizing them and then truncating them. Some recent papers try to respect the binary nature of the codes during their optimization, using techniques such as alternating optimization and GraphCut [4, 6]. Most of these approaches are slow and limited to small datasets (a few thousand points) because of the quadratic number of pairwise terms in the objective. We propose a different, much simpler approach. Rather than coupling the hash functions into a single objective function, we train each hash function *independently from each other and using a single-bit objective function of the same form.*

Our proposed method: Independent Laplacian Hashing (ILH) We focus on the Laplacian loss:

$$E(\mathbf{Z}) = \sum_{n,m=1}^{N} w_{nm}^{+} \|\mathbf{z}_n - \mathbf{z}_m\|^2 - \sum_{n,m=1}^{N} w_{nm}^{-} \|\mathbf{z}_n - \mathbf{z}_m\|^2 \text{ where } \mathbf{z}_n = \mathbf{h}(\mathbf{x}_n),\ n = 1, \dots, N. \quad (1)$$

Here, $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ is the dataset of high-dimensional feature vectors (e.g. SIFT features of an image), $\mathbf{h}$: $\mathbb{R}^D \to \{-1, +1\}^b$ are b binary hash functions, $\mathbf{z} = \mathbf{h}(\mathbf{x})$ is the b-bit code vector for input $\mathbf{x} \in \mathbb{R}^D$, and $w_{nm}^{+} \geq 0$ and $w_{nm}^{-} \geq 0$ are the similarity values for pairs of similar and dissimilar points ("positive" and "negative" neighbors). By having negative neighbors, we avoid the trivial solution of mapping all points to the same code, i.e., $\mathbf{z}_1 = \cdots = \mathbf{z}_N$.

At first sight, optimizing (1) without constraints does not seem like a good idea: since $\|\mathbf{z}_n - \mathbf{z}_m\|^2$ separates over the b bits, we obtain b independent identical objectives, one over each hash function h_b. And, if all hash functions are equal, they are equivalent to using just one ($b = 1$), which will give a much lower precision/recall than using b different hash functions. In the literature of hashing (or of manifold learning, as in Laplacian eigenmaps [1]), this is avoided by adding terms to the objective or constraints that couple the b functions, as in Supervised Hashing with Kernels (KSH) [5].

We propose a different approach based on ensemble learning techniques [3]. As we will see, not only is this far simpler, but it even performs better. To construct an ensemble of classifiers, we have a training set of input vectors and output class labels, and want to train several classifiers whose outputs are then combined (usually by majority vote). If the classifiers are all equal, we gain nothing over a single classifier. Hence, it is necessary to introduce *diversity* among the classifiers so that they disagree in their predictions.

The Laplacian objective (1) takes the following form for a single bit:

$$E(\mathbf{z}) = \sum_{n,m=1}^{N} w_{nm}^{+}(z_n - z_m)^2 - \sum_{n,m=1}^{N} w_{nm}^{-}(z_n - z_m)^2 \text{ where } \begin{cases} z_n = h(\mathbf{x}_n) \in \{-1, 1\}, \\ n = 1, \dots, N. \end{cases} \quad (2)$$

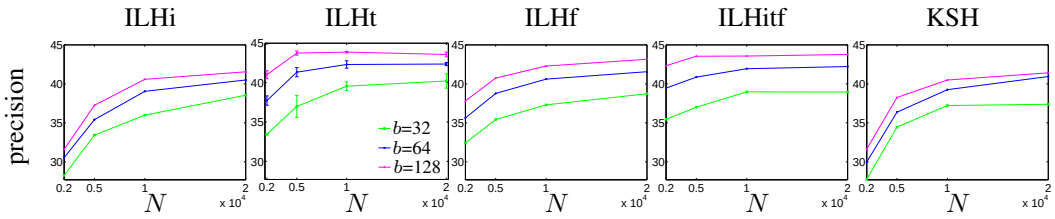
We consider three types of diversity mechanism (as well as their combination). *ILHi* uses different initializations: each hash function is initialized from a random N-bit vector $\mathbf{z}$. *ILHt* uses different training sets: each hash function uses a training set of N points that is different and disjoint from that of other hash functions. *ILHf* uses different feature subsets: each hash function is trained on a random subset of $1 \leq d \leq D$ features sampled without replacement (so the d features are distinct).

To optimize (2), we use a two-step approach, where we first optimize (2) over the N bits and then learn the hash function by fitting to it a binary classifier. (It is also possible to optimize over the hash function directly with the method of auxiliary coordinates [2, 6], which essentially iterates over optimizing (2) and fitting the classifier.) The Laplacian objective (2) over the N bits is NP-complete if we have negative neighbors (i.e., some $w_{nm}^{-} > 0$). We approximately optimize it using the GraphCut algorithm applied in alternating fashion to submodular blocks as described in [4]. The approximate optimizer found depends on the initial $\mathbf{z} \in \{-1, 1\}^N$.

Training the hash functions independently has some important advantages beyond its obvious simplicity. (1) Training the b functions can be parallelized perfectly. Coupled objective functions such as KSH do not exhibit obvious parallelism. (2) Even in a single processor, doing b binary optimizations over N variables each is much more effective than doing one binary optimization over bN variables. (3) The solution exhibits "nesting", that is, to get the solution for $b + 1$ bits we just need to take a solution with b bits and add one more bit (as happens with PCA).

Experiments We show results on the CIFAR dataset, which contains 58 000 images for training and 2 000 images for test in 10 classes. As hash functions (for each bit), we use linear SVMs. Ground-truth for a query contains all training points with the same label as the query and the retrieved set contains the k nearest neighbors of the query point in the Hamming space.

We evaluate the 3 mechanisms ILHi, ILHt and ILHf, and their combination ILHitf, over a range of number of bits b and training set size N. As baseline coupled objective function, we use KSH optimized using GraphCut [4, 5], which is one of the state-of-the-art hashing methods. All our diversity mechanisms beat or are comparable with KSH. The consistently best diversity mechanism is ILHt, which is significantly better than KSH. Intuitively, besides the effect introduced by diversity, ILHt has the distinct advantage of effectively using b times as much training data, because each hash function has its own disjoint dataset. Using bN training points in KSH would be far slower.



Acknowledgments Work supported by NSF award IIS–1423515.

References

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, June 2003.

[2] M. Á. Carreira-Perpiñán and R. Raziperchikolaei. Hashing with binary autoencoders. In *CVPR*, 2015.

[3] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, second edition, 2014.

[4] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *CVPR*, 2014.

[5] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, 2012.

[6] R. Raziperchikolaei and M. Á. Carreira-Perpiñán. Learning hashing with affinity-based loss functions using auxiliary coordinates. arXiv:1501.05352 [cs.LG], Jan. 21 2015.