

Random Exemplar Hashing

Tao-Yen Tang, Tzu-Wei Huang, and Hwann-Tzong Chen

Department of Computer Science
National Tsing Hua University, Taiwan

Abstract. We present a new method that addresses the problem of approximate nearest neighbor search via partitioning the feature space. The proposed *random exemplar hashing* algorithm can be used to generate binary codes of data to facilitate nearest neighbor search within large datasets. Inspired by the idea of using an ensemble of classifiers for discriminative learning, we devise an unsupervised learning algorithm to explore the feature space with respect to randomly selected exemplars. Experimental results on three large datasets show that our method outperforms the state-of-the-art, especially on the cases of longer binary codes.

1 Introduction

Nearest neighbor criterion [6] is one of the most popular learning strategies for solving vision problems. Despite its simplicity, its effectiveness has been shown to be acceptable for various vision applications, especially when high-dimensional distinctive visual features are employed to represent the data [2, 13, 25, 30, 33, 34]. Approximation approaches such as quantization trees, dimensionality reduction, low-dimensional embedding, and hashing are often applied to the original high-dimensional data, and more economical coding schemes can be derived to make the search of nearest neighbors among a large number of data in high-dimensional space more feasible and efficient.

Hashing techniques, *e.g.* [10, 34] have been widely explored for approximate nearest neighbor search. They can achieve fast query speed and only require a limited amount of storage. For example, *locality-sensitive hashing* (LSH) [10, 15] can offer sub-linear time search via hashing highly similar instances together in a hash table. Regarding data representations, binary codes generated by hashing are often preferred for large-scale search. During query time, since the original feature vectors in the dataset have been converted to binary codes, we may use the Hamming distance to retrieve nearest neighbors. The computation of Hamming distance can be made very fast using bit-wise operations. Hashing-based algorithms are shown to perform well on various large datasets [17, 19, 20, 24, 31, 35, 37, 38].

Beyond unsupervised hashing techniques, modern discriminative classifiers have the capability to achieve good performance on supervised image categorization problems. In this work we aim to make use of the discriminating power of classifiers in an unsupervised setting. The mechanism of approximate nearest neighbor search by hashing can be viewed as partitioning the feature space into meaningful regions such that the locality property can be preserved. The main idea and contribution of our approach is to model and partition the feature space based on local data distributions. We propose an unsupervised learning method that explores the feature space with respect to randomly selected exemplars. The method is called *random exemplar hashing*, in which the randomly selected exemplars are employed to create an ensemble of classifiers for discriminative learning. We show that our method is able to generate representative binary hash codes for searching large datasets.

2 Related Work

The basic idea of locality-sensitive hashing (LSH) [10, 15] is to construct hash functions from random projections, which are shown to keep near neighbors together with high probability and thus preserve locality. Since the success of LSH, many hashing techniques have been proposed for approximate nearest neighbor search. Hashing techniques like LSH are often categorized as *data-independent* methods because they do not need to learn hash functions from training data. Another example is the work of Ji *et al.* [19]. The basic idea of their approach is to orthogonalize the random projection vectors in batches, and it is theoretically guaranteed that the orthogonalized random projections also provide an unbiased estimate of angular similarity. Yagnik *et al.* [37] propose the *winner-takes-all* hashing, which is a sparse embedding method that converts the input feature space into a Hamming space such that the Hamming distance closely resemble the rank similarity.

Data-independent hashing techniques are simple and general, but the majority of hashing techniques try to derive hash codes from data. Semantic hashing [31] uses *restricted Boltzmann machines* to learn compact binary codes representing the memory addresses so that semantically similar data are located at nearby addresses and can be easily compared using the Hamming distance. In a similar spirit, Weiss *et al.* [36] show that the learning

of compact binary codes can be formulated as an NP-hard graph partitioning problem. They relax the original problem, and present the *spectral hashing* algorithm to solve the relaxed problem. The spectral hashing algorithm can also handle the problem of out-of-sample extension of spectral methods, and thus can be applied to new input data easily. More recently, Weiss *et al.* [35] propose a new multidimensional formulation for learning binary codes that seek to reconstruct the affinities between data. Unlike spectral hashing whose performance may degrade as the number of bits increases, the optimal code produced by multidimensional spectral hashing is guaranteed to reproduce the affinities between data as the number of bits increases. Spectral hashing mainly assume that the input feature vectors are from a Euclidean space. When a better kernel function is available for measuring the similarities between feature vectors, *kernelized locality-sensitive hashing* (KLSH) [22] may be used to take advantage of the kernel-based similarity measure and derive more effective binary codes for the data.

Several recent methods are also related to our work: The *isotropic hashing* (IsoHash) algorithm proposed by Kong and Li [20] is to learn an orthogonal matrix to rotate the PCA projection matrix. The *anchor graph hashing* (AGH) algorithm presented by Liu *et al.* [24] uses anchor graphs to estimate graph Laplacian eigenvectors, and the hash functions are learned by thresholding the lower eigenfunctions of the anchor graph Laplacian. The *iterative quantization* (ITQ) algorithm proposed by Gong and Lazebnik [11] generates binary hash codes by deciding a rotation of zero-centered data such that the quantization error of mapping the data to the vertices of binary hypercube can be minimized.

While most embedding algorithms seek to compute binary codes for both the query and the items in the database, some recent research results suggest that binarizing the query is not compulsory, *e.g.*, [3, 8, 12, 16]. These asymmetric algorithms show that to binarize the database items but not the query may provide superior accuracy. The cost of computing the asymmetric distances is generally comparable to the cost of computing symmetric Hamming distances.

Another way of exploring the feature spaces and encoding the data is to build decision trees based on random selection of features, such as *randomized trees* [1, 23] or *random forests* [4]. Apart from the popular strategies of tree-structured random feature selection, Rahimi and Recht [29] present two sets of randomized feature maps that can be used to approximate various radial basis (shift-invariant) kernels. Rahimi and Recht show that the combination of the random features with simple linear learning algorithms is comparable to applying kernel methods (*e.g.* SVMs) and thus advantageous to dealing with large datasets.

Our work is inspired by the image categorization method of Dai *et al.* [7]. They use *random walk* to generate a series of weak training sets. For each WT set, a max-margin classifier is trained to further partition the whole dataset to produce a proximity matrix. The final ensemble proximity matrix derived from the collection of classifiers is used to perform spectral clustering for image categorization. In our approach, we randomly choose some data as exemplars. We obtain the WT sets for the exemplars based on the affinities induced by Gaussian approximation to the feature space, using the method described in [9]. Furthermore, we use the predictions of classifiers to produce hash codes for searching rather than performing image clustering. Another related method is *self-taught hashing* (STH) proposed by Zhang *et al.* [38]. They treat binary codes of training data as binary class labels and try to learn a set of binary classifiers that can predict the binary codes for new queries. While our method is to partition the feature space defined by the data, the STH method, in a similar sense, is to partition the feature space through the proxy Hamming space defined by precomputed hash codes. Our method also shares similar spirits of boundary-based representations and hyperplane partitioning methods, of which a detailed survey is available in [32].

3 Random Exemplar Hashing

Many previous algorithms try to find projections that can approximate the original distance by Hamming distance in the Hamming space. We present a new method that addresses the unsupervised learning task of finding projections through solving small classification problems. The main idea of our method is to partition the high-dimensional feature space by a set of weak classifiers. Our method is fully unsupervised despite the process of constructing weak classifiers involves discriminative learning. We do not use the information of class labels or attributes. The weak classifiers are learned from the distributions of data. After combining the decision surfaces of weak classifiers, we can partition the original feature space into convex sets, and the affinities of feature vectors can be approximately measured according to the binary indexes associated with the convex sets.

Fig. 1 shows the pipeline of our method. At the training stage, we randomly select K ‘exemplars’ from the dataset and set up small classification problems to learn partition functions. We find ‘neighbors’ and ‘foreigners’ for each of the exemplars, and build K *weak training* (WT) sets. The idea of WT sets is similar to [7] but we do not use random walk to collect the WT sets. Each WT set contains M neighboring data points denoted as positive data and M distant data points denoted as negative data. We then train a weak linear classifier for each WT set and obtain K classifiers. The weak classifiers partition the feature space into convex regions indexed by binary codes.

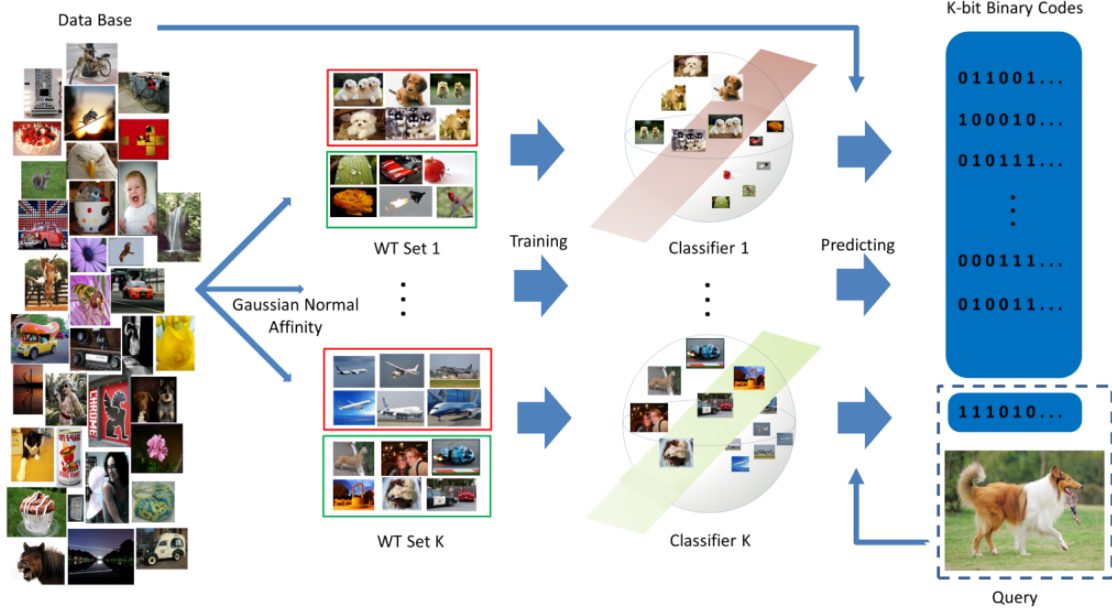


Fig. 1. The work flow of our method. From the training data we build K weak training (WT) sets. Each WT set contains M neighboring data points denoted as positive data and M distant data points denoted as negative data. We then train a weak linear classifier for each WT set and obtain K classifiers. The weak classifiers partition the feature space into convex regions indexed by binary codes. All training data in the dataset are assigned to one of the convex regions. Given a query, we compute its binary code according to classifiers' predictions and use the binary code to retrieve similar training data.

The K learned partitions associated with the K exemplars yield a way of K -bit indexing for the feature space. All training data in the dataset can then be assigned to one of the convex regions. As mentioned earlier, training is carried out in a totally unsupervised manner, despite the need of learning local classifiers for random exemplars. Given a query, we use the learned partition functions, *i.e.*, the weak classifiers, to decide the region to which the query belongs and get the binary code of the query for search.

An intuitive way to create the required K WT sets is to select K exemplars randomly from the training data and define a measure of affinity with respect to each exemplar. We choose the neighbors of each exemplar as the positive data of WT set, and select farthest data of that exemplar as the negative data of WT set. We may train a linear support vector machine (SVM) to separate the WT set and obtain a weak classifier.

The affinity measure for each exemplar plays an important role in collecting the members of WT set. It is easy to imagine that the neighbors of an exemplar are close to each other. On the other hand, data points that are far away from the exemplar may scatter everywhere in the high-dimensional feature space. It would be inadequate and ambiguous if we seek to learn a linear classifier that separates so distributed data. The classifier may be confused by the diversely distributed data and thus cannot find a meaningful partition.

To resolve the problem, we adopt the idea of Gaussian approximation of feature space [9]. It has been shown that a high-dimensional feature space can be approximated by the covariance matrix on training data. Given the covariance matrix, the normal vector at a data point is computed by multiplying the inverse of covariance matrix. An affinity measure called *Gaussian normal affinity* can thus be derived from the normal vector: We use the normal vector to define the nearest and farthest points when we choose the 'neighbors' and 'foreigners' for each of the exemplars. Data points are projected onto the approximated Gaussian normal at the location of the exemplar, and therefore the farthest points are more likely to be close to each other. In this case, it is easier to separate the two groups by a linear decision boundary.

Selecting suitable WT sets is important to our method. Gaussian normal affinity provides an effective measure for identifying suitable farthest groups as negative training data. We describe the computation of Gaussian normal affinity in detail in the next section.

3.1 Gaussian Normal Affinity

The idea of Gaussian normal affinity is introduced by Gharbi *et al.* [9], which is aimed to provide a re-interpretation of the exemplar-based SVM approach [26]. Gharbi *et al.* show that data points distributed in a

high-dimensional feature space can be approximated by a Gaussian, and the normal to the Gaussian at a given point can be used to rank the affinities between the given point and other points. The normal $\mathbf{w}_x$ to the Gaussian at the location of exemplar $\mathbf{x}$ is thus computed by

$$\mathbf{w}_x = \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}), \quad (1)$$

where $\boldsymbol{\mu}$ and Σ are the mean and the covariance matrix of the approximated Gaussian. The mean $\boldsymbol{\mu}$ is trivially obtained by $\boldsymbol{\mu} = \sum_i \mathbf{v}_i / N$, where N is the number of training data and $\{\mathbf{v}_i | \mathbf{v}_i \in \mathbb{R}^D, i = 1, \dots, N\}$ are the feature vectors in the training set. The covariance matrix Σ , which is a D -by- D symmetric matrix, can be estimated as $\Sigma = \sum_i (\mathbf{v}_i - \boldsymbol{\mu})(\mathbf{v}_i - \boldsymbol{\mu})^T / N$. Note that we only need to compute $\boldsymbol{\mu}$ and Σ once, and they are applicable for new exemplars as long as the exemplars belong to the same feature space.

We may define the Gaussian normal affinity as the closeness measured along the direction of the Gaussian normal $\mathbf{w}_x$. That is, we project feature vectors onto the Gaussian normal $\mathbf{w}_x$, and use the projected locations to decide the neighbors and foreigners with respect to the exemplar $\mathbf{x}$. Gaussian normal affinity is an exemplar-specific function. It measures the distance with respect to the ‘shape’ of feature space and the location of exemplar in the feature space.

Gaussian normal affinity helps to define meaningful negative data for WT sets so that linear classifiers can be trained to partition the feature space. All of our experimental results confirm that using Gaussian normal affinity to choose positive and negative data for weak training is better than using Euclidean distance.

It is worth mentioning that Gaussian normal affinity is only used to build the WT sets during training. After we perform weak training tasks and obtain the ensemble of weak classifiers, we directly apply the classifiers to the input feature vectors since the classifiers have learned how to partition the feature space.

3.2 Single and Dyadic Exemplar Selection Schemes

Gharbi *et al.* [9] have shown that, in the high-dimensional feature space, feature vectors tend to lie at the outer hypersurface of the set of all feature vectors. Empirically we also find that the group of neighbors and the group of foreigners chosen by an exemplar according to Gaussian normal affinity often lie far away at the opposite ends of the outer hypersurface. As a result, the decision plane learned by a linear SVM tends to pass the center of the set of feature vectors. To explore other possibilities of partitioning the feature space, we also try another way to establish WT sets. Instead of selecting one exemplar for each weak training task, we randomly choose two exemplars and find the neighbors of each exemplar according to its Gaussian normal affinity function. We then arbitrarily assign the neighbors of one exemplar as positive data and the neighbors of the other exemplar as negative data, and use such a WT set to train a weak classifier. We call this process the dyadic exemplar selection scheme [27] in contrast to our original single exemplar selection scheme. We compare these two schemes in latter experiments and show that they might perform differently on different datasets.

Besides the single and dyadic exemplar selection schemes, we also come up with a very simple version of our method: Given a WT set of positive and negative data found by the single exemplar selection scheme, we compute the mean of positives and the mean of negatives, and take the hyperplane that is orthogonal to the Gaussian normal and passes the middle point of the two means as the decision plane. By doing so we can obtain a weak classifier without training linear SVM.

4 Experiments

We evaluate the proposed method on three commonly used large datasets, and compare our method with several state-of-the-art hashing techniques by the performance on those datasets.

4.1 Datasets

Three benchmark datasets are used in our experiments:

1. The *MIR-Flickr* dataset [14] consists of 25,000 images downloaded from Flickr through its public API. We represent each image by a 540-dimensional Gist descriptor [28]. We randomly choose 1,000 images as queries, and the remaining 24,000 images are considered as the database. The ‘ground truth’ for each query is defined by the top 5% neighbors measured in Euclidean distance, which is the same setting presented in [19].

2. The *CIFAR-10* dataset [21] consists of 60,000 images, each of which has a size of 32×32 pixels. We represent each image as a 320-dimensional Gist descriptor. The dataset contains 10 classes including different kinds of vehicles and animals. In addition to the true labels provided by the dataset, we also test it with ground truth defined by Euclidean distance. We follow the setting of [20], which sets the average distance of the 50th nearest neighbor as a threshold, and every image whose distance to the query is smaller than the threshold is considered a true positive.
3. The *GIST1M* dataset used in [18] contains 500,000 Gist features for learning, 1,000 for query, and 1,000,000 as the database to be searched. The learning set consists of the images extracted from the tiny-image dataset of [34]. The database for search is a medley of the INRIA Holidays images and the Flickr1M used in [17]. The ground truth is defined by the first 100 nearest neighbors of each query found by Euclidean distance.

4.2 Evaluation details

We compare our method with several state-of-the-art hashing techniques including LSH [10], SH [36], IsoHash [20], ITQ [11], MDSH [35], and AGH [24]. In all comparisons, we run the available program provided by those authors. For the sake of speed, we replace the k-means step in AGH by random selection of data as anchors. This modification does not influence the performance much, but hugely accelerates their method. For AGH on CIFAR-10 and GIST1M datasets, we set the parameter of the number of anchors as 500 and the parameter of how many nearest anchors used as 100. For AGH on MIR-Flickr dataset we set these two parameters as 500 and 120, respectively. The bandwidth parameter σ of MDSH for computing the affinity matrix is fixed to 0.4.

In our implementation of the proposed method, we use LIBSVM [5] with linear kernel and set the penalty parameter C of the error term equal to 350 for both the single and dyadic schemes. Since the number of training data is small, it should be adequate to enforce a larger penalty weight on the error term without increasing the training time. The issue of overfitting is also not a concern in our case because our goal of weak training is deriving partition functions rather than solving real classification problems. We have tried different settings on the parameter M regarding the size of WT set (containing M positive data and M negative data). We test the value of M from $M = 10$ to $M = 40$. We find that changes in the value of M within this range do not affect the final precision much, and so we simply keep $M = 20$ for all experiments.

We compute the mean average precision (mAP) and plot the precision-recall curves for the evaluations on MIR-Flickr and CIFAR-10 datasets. For GIST1M dataset, we draw the curve that depicts the number of true positives versus R nearest neighbors. We evaluate all the methods on each dataset using different bit lengths, from 32 bits to 512 bits, except the CIFAR-10 dataset. Since ITQ and IsoHash cannot create more bits than the original feature dimensions, we set the largest length to 320 bits for CIFAR-10 dataset.

Table 1. Results on MIR-Flickr and GIST1M datasets. The rank of each method on each category is included (ranked from 1 to 9).

method	mAP of MIR-Flickr					recall @ $R = 500$ of GIST1M				
# bits	32	64	128	256	512	32	64	128	256	512
AGH	0.347 ₄	0.431 ₂	0.442 ₈	0.432 ₈	0.375 ₈	0.069 ₄	0.120 ₃	0.159 ₅	0.168 ₇	0.189 ₇
LSH	0.292 ₈	0.371 ₈	0.447 ₆	0.500 ₅	0.538 ₅	0.033 ₈	0.079 ₇	0.150 ₆	0.232 ₅	0.319 ₄
SH	0.219 ₉	0.251 ₉	0.306 ₉	0.333 ₉	0.279 ₉	0.034 ₇	0.058 ₈	0.070 ₉	0.106 ₉	0.128 ₉
IsoHash	0.363 ₃	0.410 ₆	0.454 ₅	0.478 ₆	0.498 ₆	0.109 ₂	0.168 ₂	0.228 ₁	0.282 ₃	0.323 ₃
ITQ	0.372 ₂	0.414 ₄	0.443 ₇	0.465 ₇	0.480 ₇	0.111 ₁	0.169 ₁	0.223 ₂	0.261 ₄	0.293 ₅
MDSH	0.415 ₁	0.504 ₁	0.564 ₁	0.585 ₂	0.581 ₃	0.107 ₃	0.095 ₅	0.147 ₇	0.184 ₆	0.200 ₆
Our-single	0.307 ₅	0.427 ₃	0.542 ₂	0.630 ₁	0.706 ₁	0.040 ₆	0.092 ₆	0.203 ₃	0.337 ₂	0.504 ₁
Our-dyadic	0.306 ₆	0.411 ₅	0.500 ₃	0.558 ₃	0.615 ₂	0.048 ₅	0.107 ₄	0.194 ₄	0.343 ₁	0.466 ₂
Our-simple	0.296 ₇	0.380 ₇	0.460 ₄	0.515 ₄	0.572 ₄	0.023 ₉	0.041 ₉	0.076 ₈	0.115 ₈	0.152 ₈

4.3 Results

Every experimental result presented here is an average of the performances over five runs. The left part of Table. 1 shows the mAP of MIR-Flickr and the corresponding precision-recall curves are shown in Fig. 2. As we can see, our method generally outperforms other methods on the cases of using longer binary codes. The precision of our method improves faster than other methods as the number of bits increases, which suggests that each additional bit generated by our method provides more information that can further partition the feature space in an efficient way.

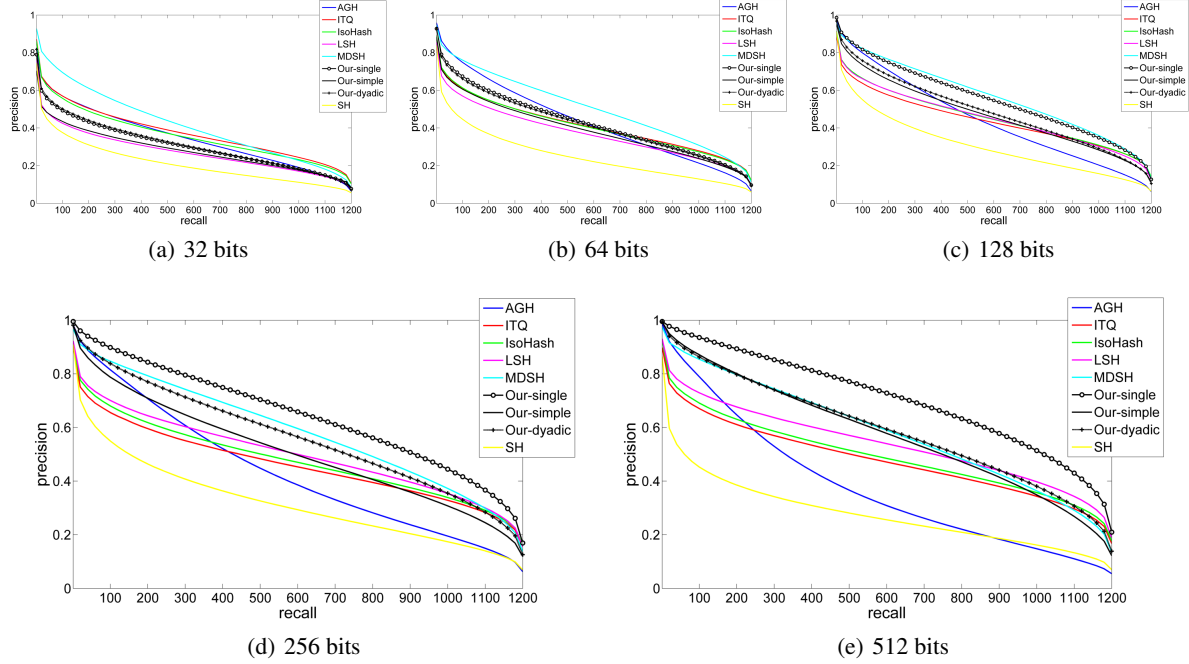


Fig. 2. The precision recall curves of MIR-Flickr dataset. (a)-(e) The comparisons on different bit-lengths.

Table 2. Results on CIFAR-10. The rank of each method on each category is included (ranked from 1 to 9).

method	mAP of CIFAR 50 th nearest neighbors					Precision @ recall=500 for CIFAR true labels				
# bits	32	64	128	256	320	32	64	128	256	320
AGH	0.193 ₃	0.234 ₄	0.263 ₆	0.270 ₈	0.259 ₈	0.213 ₃	0.214 ₅	0.205 ₈	0.185 ₈	0.175 ₈
LSH	0.120 ₈	0.184 ₈	0.249 ₈	0.310 ₆	0.345 ₄	0.182 ₈	0.196 ₈	0.221 ₆	0.236 ₆	0.247 ₅
SH	0.114 ₉	0.148 ₉	0.180 ₉	0.213 ₉	0.207 ₉	0.169 ₉	0.174 ₉	0.165 ₉	0.160 ₉	0.157 ₉
IsoHash	0.202 ₂	0.259 ₂	0.307 ₂	0.330 ₄	0.354 ₃	0.218 ₂	0.230 ₂	0.239 ₃	0.246 ₄	0.249 ₄
ITQ	0.209₁	0.266₁	0.307 ₃	0.331 ₃	0.343 ₅	0.231₁	0.244₁	0.253₁	0.260 ₂	0.263 ₂
MDSH	0.155 ₄	0.241 ₃	0.293 ₅	0.309 ₇	0.324 ₇	0.196 ₆	0.210 ₇	0.213 ₇	0.213 ₇	0.212 ₇
Our-single	0.136 ₇	0.207 ₆	0.304 ₄	0.407₁	0.440₁	0.183 ₇	0.217 ₄	0.232 ₄	0.254 ₃	0.262 ₃
Our-dyadic	0.140 ₅	0.221 ₅	0.308₁	0.381 ₂	0.408 ₂	0.211 ₄	0.223 ₃	0.249 ₂	0.264₁	0.270₁
Our-simple	0.139 ₆	0.197 ₇	0.261 ₇	0.323 ₅	0.334 ₆	0.204 ₅	0.213 ₆	0.228 ₅	0.239 ₅	0.242 ₆

Table. 2 shows the mAP and the precision at recall equal to 500 for CIFAR-10 with respect to *i*) the ground truth defined by Euclidean distance and *ii*) the true label. The results of CIFAR-10 with ground truth defined by Euclidean distance is similar to MIR-Flickr. Our method performs well when the bit-length is longer than 128 bits. In general, the single exemplar scheme performs more stably than the dyadic scheme, even though in some cases the dyadic scheme can achieve slightly better performance. The performance of the simple version of our method (without SVM training) is acceptable but is not comparable to the other two settings of our method that involve SVM training.

In the last set of experiments we evaluate the scalability of our method using the GIST1M dataset. Fig. 3 shows the experimental results on the number of true positives versus the first R retrieved nearest neighbors. The right part of Table. 1 summarizes the performance of different methods with respect to the recall at $R = 500$. Our method is better than other hashing techniques on the cases of 256-bit and 512-bit codes. Unlike in the previous two experiments, this time the simple version of our method performs poorly on the GIST1M dataset. This might imply that, for finding the decision boundary, the large-margin criterion is better than separating at the middle point. The drawback of inadequate partitioning of the feature space becomes significant when the size of dataset increases.

4.4 Computation Time of Querying

The training stage of our method is very efficient because we only use a small number of data to train the weak classifiers. Regarding the query time, it is worth noting that, the number of support vectors does not cause additional computational burden because we use linear kernel. All support vectors can be merged into a single vector

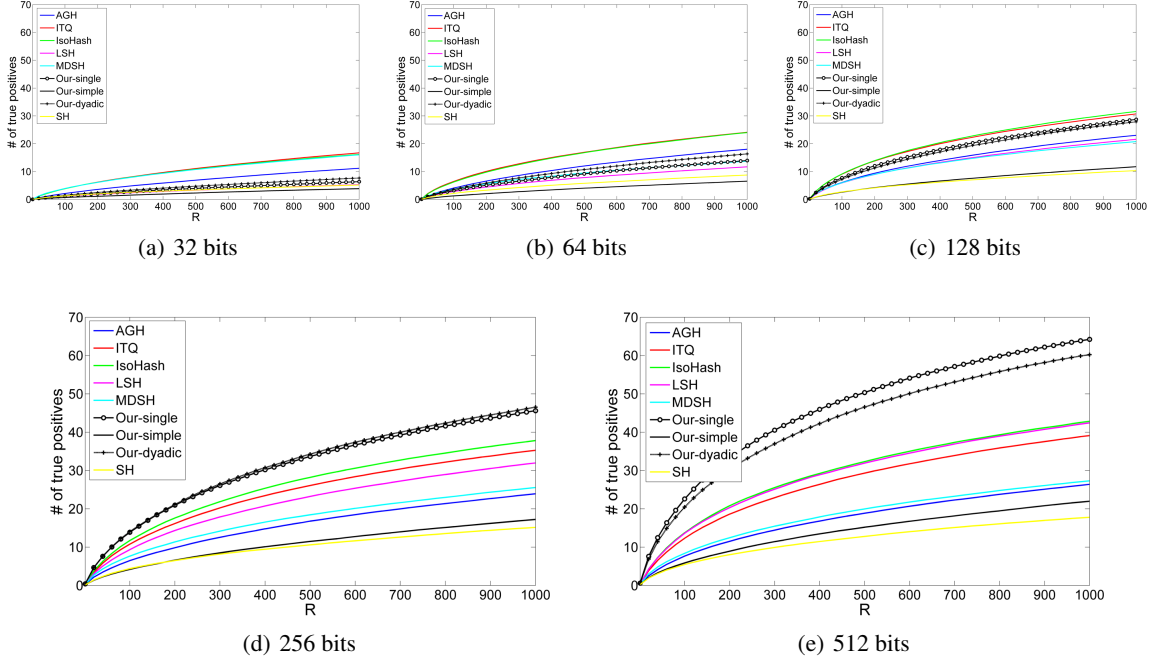


Fig. 3. The results on GIST1M dataset. Each curve depicts the performance evaluated by the number of true positives versus the R nearest neighbors. (a)-(e) The performances using different bit-lengths.

Table 3. Number of support vectors for different datasets and bit-lengths.

# bits	MIR-Flickr	CIFAR	GIST1M
32	1099	1029	1036
64	2196	2040	2015
128	4418	4029	4084
256	8797	8014	8236
512	17630	16110	16279

via linear combination. Therefore, it takes only one inner-product to derive one hash bit, and the overall computational complexity of querying only depends on the bit-length of hash code. As a result, the query time of our method is faster than ITQ [11] and comparable to LSH [10]. Table. 3 lists the number of support vectors we obtain in each experiment for reference. Our method can be viewed as a two-level modeling of the original data: from the original data points to the support vectors, and from the support vectors to the partition functions for hashing. Our method can scale-up very well to deal with nearest-neighbor search problems.

5 Conclusion

This paper presents a new method to make use of the discriminating power of classifiers in an unsupervised way. Our method is able to model and partition the feature space according to the Gaussian normal affinity measure. We have shown that with the efficient mechanism of partitioning the feature space, our method performs very well on large datasets, especially when we use longer binary codes.

References

1. Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588, 1997.
2. O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008.
3. J. Brandt. Transform coding for fast approximate nearest neighbor search in high dimensions. In *CVPR*, pages 1815–1822, 2010.
4. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
5. C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

6. T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.
7. D. Dai, M. Prasad, C. Leistner, and L. J. V. Gool. Ensemble partitioning for unsupervised image categorization. In *ECCV (3)*, pages 483–496, 2012.
8. W. Dong, M. Charikar, and K. Li. Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces. In *SIGIR*, pages 123–130, 2008.
9. M. Gharbi, T. Malisiewicz, S. Paris, and F. Durand. A gaussian approximation of feature space for fast image similarity. Technical report, Massachusetts Institute of Technology, 2012.
10. A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.
11. Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, pages 817–824, 2011.
12. A. Gordo and F. Perronnin. Asymmetric distances for binary embeddings. In *CVPR*, pages 729–736, 2011.
13. K. Grauman and T. Darrell. Pyramid match hashing: Sub-linear time indexing over partial correspondences. In *CVPR*, 2007.
14. M. J. Huiskes and M. S. Lew. The mir flickr retrieval evaluation. In *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*, New York, NY, USA, 2008. ACM.
15. P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, 1998.
16. M. Jain, H. Jégou, and P. Gros. Asymmetric hamming embedding: taking the best of our bits for large scale image search. In *ACM Multimedia*, pages 1441–1444, 2011.
17. H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV (1)*, pages 304–317, 2008.
18. H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 33(1):117–128, jan 2011.
19. J. Ji, J. Li, S. Yan, B. Zhang, and Q. Tian. Super-bit locality-sensitive hashing. In *NIPS*, pages 108–116, 2012.
20. W. Kong and W.-J. Li. Isotropic hashing. In *NIPS*, pages 1655–1663, 2012.
21. A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Toronto University, 2009.
22. B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, pages 2130–2137, 2009.
23. V. Lepetit, P. Laguerre, and P. Fua. Randomized trees for real-time keypoint recognition. In *CVPR (2)*, pages 775–781, 2005.
24. W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, pages 1–8, 2011.
25. D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999.
26. T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, pages 89–96, 2011.
27. B. Moghaddam and G. Shakhnarovich. Boosted dyadic kernel discriminants. In *NIPS*, pages 745–752, 2002.
28. A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.
29. A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, 2007.
30. R. Salakhutdinov and G. E. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. *Journal of Machine Learning Research - Proceedings Track*, 2:412–419, 2007.
31. R. Salakhutdinov and G. E. Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978, 2009.
32. H. Samet. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
33. G. Shakhnarovich, T. Darrell, and P. Indyk. Nearest-neighbor methods in learning and vision. *IEEE Transactions on Neural Networks*, 19(2):377, 2008.
34. A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1958–1970, 2008.
35. Y. Weiss, R. Fergus, and A. Torralba. Multidimensional spectral hashing. In *ECCV (5)*, pages 340–353, 2012.
36. Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.
37. J. Yagnik, D. Strelow, D. A. Ross, and R.-S. Lin. The power of comparative reasoning. In *ICCV*, pages 2431–2438, 2011.
38. D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *SIGIR*, pages 18–25, 2010.