HAL
archives-ouvertes.fr

Random Maximum Margin Hashing

Alexis Joly, Olivier Buisson

▶ **To cite this version:**

Alexis Joly, Olivier Buisson. Random Maximum Margin Hashing. CVPR'11 - IEEE Computer Vision and Pattern Recognition, Jun 2011, Colorado springs, United States. IEEE, pp.873-880, 2011, <http://cvpr2011.org/>. <10.1109/CVPR.2011.5995709>. <hal-00642178>

HAL Id: hal-00642178
https://hal.inria.fr/hal-00642178

Submitted on 17 Nov 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Random Maximum Margin Hashing

Alexis Joly
INRIA
Domaine de Voluceau, 78153, France
http://www-rocq.inria.fr/~ajoly/

Olivier Buisson
INA
4 avenue de l'Europe, 94336, France
http://obuisson.free.fr/

Abstract

Following the success of hashing methods for multidimensional indexing, more and more works are interested in embedding visual feature space in compact hash codes. Such approaches are not an alternative to using index structures but a complementary way to reduce both the memory usage and the distance computation cost. Several data dependent hash functions have notably been proposed to closely fit data distribution and provide better selectivity than usual random projections such as LSH. However, improvements occur only for relatively small hash code sizes up to 64 or 128 bits. As discussed in the paper, this is mainly due to the lack of independence between the produced hash functions. We introduce a new hash function family that attempts to solve this issue in any kernel space. Rather than boosting the collision probability of close points, our method focus on data scattering. By training purely random splits of the data, regardless the closeness of the training samples, it is indeed possible to generate consistently more independent hash functions. On the other side, the use of large margin classifiers allows to maintain good generalization performances. Experiments show that our new Random Maximum Margin Hashing scheme (RMMH) outperforms four state-of-the-art hashing methods, notably in kernel spaces. [1]

1. Introduction

10 years after the first LSH [6] version, hashing methods are gaining more and more interest in the computer vision community. Embedding visual feature spaces in very compact hash indeed allow to drastically scale up many computer vision applications (from 10 to 1000 times larger datasets). One advantage of hashing methods over trees or other structures is that they allow simultaneously efficient indexing and data compression. Hash codes can indeed be used either to gather features into buckets but also to approximate exact similarity measures by efficient hash code comparisons (typically a hamming distance on binary codes). Memory usage and time costs can therefore be drastically reduced. Hashing methods can be classified across three main categories:

Data independent hashing functions: in these methods, the hashing function family is defined uniquely and independently from the data to be processed. We can distinguish the one based on randomized process, to which Locality Sensitive Hashing (LSH) functions belong (L_p stable [4], min-hash [3], random fourier features [17]), and the one based on a deterministic structuring, including grids [22], space filling curves [10, 16] or more recently, lattices [7, 19]. The randomized ones are usually considered as more adaptive to heterogeneous data distributions and are thus usually more efficient than deterministic hash functions. However, some recent works did show that using more complex lattices may be more effective [7, 19], at least under the L_2 metric and for the studied data distributions. Most recent research on randomized methods did focus more on new similarity measures, notably the work of Raginsky et al. who defined a hashing function sensitive to any Shift-Invariant Kernel [17].

Data dependent hashing functions: In that case, the hashing function family is defined uniquely only for a given training dataset and the hash functions usually involve similarity comparisons with some features of the training dataset. The objective of these methods is to closely fit the data distribution in the feature space in order to achieve a better selectivity while preserving locality as much as possible. Among the most popular methods, we can cite K-mean based hashing [15], Spectral Hashing (SH) [23] based on graph partitioning theory, subspaces product quantization [8] and Restricted Boltzmann Machine (RBM) [18] based on the training of a multilayer neural network. KLSH [11], is a slight different approach since its main objective was to generalize hashing to any Mercer kernel rather than outperforming data independent methods.

[1] Acknowledgement: This work was co-funded by the EU through the Integrated Project GLOCAL http://www.glocal-project.eu/ and by the French Agropolis foundation through the project Pl@ntNet http://www.plantnet-project.org/

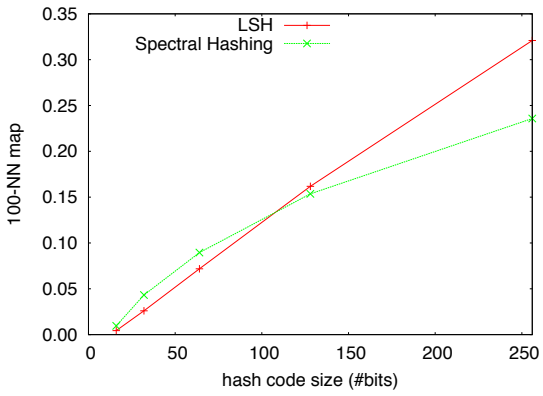

Figure 1. LSH vs Spectral Hashing for increasing hash code sizes

(Semi-)supervised data dependent hashing functions: In this last category, the training dataset contains additional supervised information, e.g. class labels [21, 12] or pairwise constraints [14]. These methods usually attempt to minimize a cost function on the hash functions set, combining an error term (to fit training data) and a regularization term (to avoid over-fitting). Our method being fully unsupervised, we did not consider these methods in our experiments.

Efficiency improvements of data dependent methods over independent ones have been shown in several studies [8, 23, 18]. But this acts only for limited hash code sizes, up to 64 or 128. Indeed, the drawback of data dependent hash functions is that their benefit degrades when increasing the number of hash functions, due to a lack of independence between the hash functions. This is illustrated by Figure 1 showing the performance of a standard LSH function compared to the popular Spectral Hashing method [23], known to outperform several other data dependent methods. This conclusion is confirmed by [17] who did show that their Shift-Invariant Kernel hashing function (data independent) dramatically outperforms Spectral Hashing for all hash code sizes above 64 bits.
Our new method answers to the two limitations of previous data dependent methods: (i) It is usable for any Mercer Kernel (ii) it produces more independent hashing functions.

2. Hashing in kernel spaces

Let us first introduce some notations. We consider a dataset $\mathbf{X}$ of N feature vectors $\mathbf{x}_i$ lying in a Hilbert space $\mathbb{X}$. For any two points $\mathbf{x}, \mathbf{y} \in \mathbb{X}$, we denote as $\mathbf{x}.\mathbf{y}$ the inner product associated with $\mathbb{X}$ and $\|\mathbf{x}\| = \sqrt{\mathbf{x}.\mathbf{x}}$ represents the norm of any vector $\mathbf{x}$. We generally denote as $\mathcal{H}$, a family of binary hash functions $h : \mathbb{X} \rightarrow \{-1, 1\}$.
If we consider hash function families based on random hyperplanes we have

$$h(\mathbf{x}) = sgn\left(\mathbf{w}.\mathbf{x} + b\right) \tag{1}$$

where $\mathbf{w} \in \mathbb{X}$ is a random variable distributed according to p_w and b is a scalar random variable distributed according to p_b. When working in the Euclidean space $\mathbb{X} = \mathbb{R}^d$ and choosing $p_w = \mathcal{N}(0, \mathbf{I})$ and $b = 0$, we get the popular LSH function family sensitive to the inner product [2, 11]. In that case, for any two points $\mathbf{q}, \mathbf{v} \in \mathbb{R}^d$ we have:

$$Pr\left[h(\mathbf{q}) = h(\mathbf{v})\right] = 1 - \frac{1}{\pi} cos^{-1}\left(\frac{\mathbf{q}.\mathbf{v}}{\|\mathbf{q}\|\,\|\mathbf{v}\|}\right) \tag{2}$$

Unfortunately, this hashing function family can not be generalized in arbitrary kernalized spaces. Let $\kappa : \mathbb{X}^2 \rightarrow \mathbb{R}$ denote a symmetric kernel function satisfying Mercer's theorem, so that κ can be expressed as an inner product in some unknown Hilbert space through a mapping function Φ such as $\kappa(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}).\Phi(\mathbf{y})$. We can still define a kernalized hashing function family as:

$$h(\mathbf{x}) = sgn\left(\kappa(\mathbf{w}, \mathbf{x}) + b\right) = sgn\left(\Phi(\mathbf{w}).\Phi(\mathbf{x}) + b\right) \tag{3}$$

But in that case, Φ being usually unknown, it is not possible to draw $\Phi(\mathbf{w})$ from a normal distribution.
Recently, Raginsky et al. [17], did introduce a new hashing scheme for the specific case of shift invariant kernels, i.e Mercer kernels verifying $\kappa(\mathbf{x}, \mathbf{y}) = \kappa(\mathbf{x} - \mathbf{y})$. They notably define the following family sensitive to the RBF kernel:

$$h(\mathbf{x}) = sgn\left(cos\left(\mathbf{w}.\mathbf{x} + b\right)\right) \tag{4}$$

where $\mathbf{w}$ is drawn from $p_w = \mathcal{N}(0, \gamma\mathbf{I})$, γ being the kernel band width, and b is uniformly distributed in $[0, 2\pi]$. Although it is proved that a unique distribution p_w may be found for any shift invariant kernel, other shift invariant kernels have not been addressed for now. The proposed method is therefore limited to the RBF kernel.
The only method proposing a solution for any Mercer kernel is KLSH [11]. In this work, the authors suggest to approximate a normal distribution in the kernel space thanks to a data dependent hashing function using only kernel comparisons. The principle is based on the central limit theorem which states that the mean of a sufficiently large number of independent random variables will be approximately normally distributed. The authors suggest to average p samples selected at random from $\mathbf{X}$ and to use a Kernel-PCA like strategy to whiten the resulting data. More formally, they define the following hashing function family:

$$h(\mathbf{x}) = sgn\left(\sum_{i=1}^{p} w_i \kappa(\mathbf{x}, \mathbf{x}_i)\right) \tag{5}$$

$$\mathbf{w} = \mathbf{K}^{-\frac{1}{2}} \mathbf{e}_t$$

where $\mathbf{K}$ is a $p \times p$ kernel matrix computed on the p training samples x_i, and e_t is a random vector containing t ones

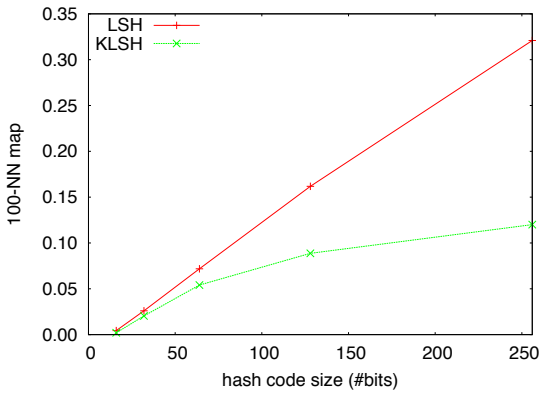

Figure 2. LSH vs KLSH for increasing hash code sizes

at random positions (in order to randomly select t indices among p). The authors show that interesting results may be achieved on diversified kernels. The performance of KLSH are however usually far from what we could expect with a real normal distribution. The convergence of the central limit theorem is indeed usually weak and depends on the input data distribution. A good way to show how this weak convergence affects the hashing quality, is to study KLSH in the linear case (i.e. by using $\kappa(\mathbf{x}, \mathbf{y}) = \mathbf{x}.\mathbf{y}$ in KLSH algorithm) and to compare to a real normal distribution (or at least the normal distribution produced by a standard Gaussian generator). Figure 2 presents such result on **ImageNet-BOF** dataset (see section 5), by comparing the mean average precision of the exact 100-NN within the hash codes produced by both methods (see section 5 for details). It shows that the performance of KLSH is quickly degrading when the number of hash functions increases. For a hash code size of 256 bits, the mean average precision is several times lower.

3. Random Maximum Margin Hashing

Our claim is that the lack of independence between hash functions is the main issue affecting the performance of data dependent hashing methods compared to data independent ones. Indeed, the basic requirement of any hashing method is that the hash function provide a **uniform** distribution of hash values, or at least as uniform as possible. Non-uniform distributions do increase the overall expected number of collisions and therefore the cost of resolving them. For Locality Sensitive Hashing methods, we argue that this uniformity constraint should not be relaxed too much even if we aim at maximizing the collision probability of close points. More formally, let us denote as $\mathbf{h}_p = [h_1, ..., h_p]$ a binary hash code of length p, lying in $\mathbb{B}^p = \{-1, 1\}^p$, where the hash functions h_i are built from a hash function family $\mathcal{H}$. For data independent hashing methods, the resulting collision probability follows:

$$Pr_p(\mathbf{q}, \mathbf{v}) = Pr\left[\mathbf{h}_p(\mathbf{q}) = \mathbf{h}_p(\mathbf{v})\right] = \left[f\left(d(\mathbf{q}, \mathbf{v})\right)\right]^p$$

where $f(.)$ is the sensitivity function of the family $\mathcal{H}$ for a given metric $d(.)$, i.e the collision probability function of a **single** hash function.

Data dependent hash functions usually aim at providing a better sensitivity function than data independent ones. They are indeed built to boost the collision probability of close points while reducing the collision probability of irrelevant point pairs. But when the hash functions are dependent from each other, we have:

$$\frac{Pr_p(\mathbf{q}, \mathbf{v})}{Pr_{p-1}(\mathbf{q}, \mathbf{v})} = Pr\left[h_p(\mathbf{q}) = h_p(\mathbf{v}) | \mathbf{h}_{p-1}(\mathbf{q}) = \mathbf{h}_{p-1}(\mathbf{v})\right]$$

Without independence, the second term is usually increasing with p and more and more diverging from the initial sensitivity function. At a certain point, the number of **irrelevant** collisions might even be not reduced anymore.

Following these remarks, we consider uniformity of produced hash codes as a primary constraint for building an efficient data dependent hash function family. For a dataset drawn from a probability density function p_x defined on $\mathbb{X}$, an ideal hash function should respect:

$$\forall p \in \mathbb{N}^*, \quad \forall \mathbf{h}_i \in \mathbb{B}^p \qquad \int_{\mathbf{h}(x)=\mathbf{h}_i} p_x(x)dx = c \qquad (6)$$

where c is a constant (equal to $\frac{1}{2^p}$). From this follows that (i) each individual hash function should be balanced (when $p = 1$):

$$\int_{h(x)=1} p_x(x)dx = \int_{h(x)=0} p_x(x)dx = \frac{1}{2} \qquad (7)$$

and (ii) all hash functions must be independent from each others.

In this work, we propose to approximate this ideal objective by training balanced and independent binary partitions of the feature space. For each hash function, we pick up M training points selected at random from the dataset $\mathbf{X}$ and we **randomly** label half of the points with -1 and the other half with 1. We denote as x_j^+ the resulting $\frac{M}{2}$ positive training samples and as $\mathbf{x}_j^-$ the $\frac{M}{2}$ negative training samples. The hash function is then computed by training a binary classifier $h_\theta(\mathbf{x})$ such as:

$$h(\mathbf{x}) = \underset{h_\theta}{\operatorname{argmax}} \sum_{j=1}^{\frac{M}{2}} h_\theta(\mathbf{x}_j^+) - h_\theta(\mathbf{x}_j^-) \qquad (8)$$

Now, the remaining question is how to choose the best type of classifier. Obviously, this choice may be guided by the nature of the targeted similarity measure. For non-metric or non-vectorial similarity measures for instance, the choice

may be very limited. In such context, a KNN classifier might be very attractive in the sense that it is applicable in all cases. Using a 1-NN classifier for kernalized feature spaces would for exemple define the following hash function family:

$$h(\mathbf{x}) = sgn\left(\max_j \kappa(\mathbf{x}, \mathbf{x}_j^+) - \max_j \kappa(\mathbf{x}, \mathbf{x}_j^-)\right) \quad (9)$$

Interestingly, it is easy to show that such family is indeed sensitive to the expected number of shared neighbors. Shared neighbors information has already been proved to be a consistent similarity measure for clustering purposes.
Better classifiers may however be found for kernel spaces. In this way, let us now consider the second main requirement of an ideal Locality Sensitive Hashing function family, that is preserving locality. Maximizing the collision probability of *close points* is indeed the primary principle of classical LSH methods. Within our balanced training strategy, we should thus minimize the probability that a point close to one of the training sample *spill over* the boundary between the two classes. In this context, maximizing the margin between positive and negative samples appear to be very well appropriated. This will indeed maximize the distance of all training samples to the boundary and guaranty that neighbors with a distance lower than the half margin do not spill over. This remark is closely related to Vapnik & Chervonenkis theory which states that large margin classifiers have low capacities and thus provide better generalization. We therefore propose to define our hash function family by the set of hyperplanes maximizing the margin between random balanced samples:

$$h(\mathbf{x}) = sgn\left(\mathbf{w}_m.\mathbf{x} + b_m\right) \quad (10)$$

$$(\mathbf{w}_m, b_m) = \underset{\mathbf{w},b}{\operatorname{argmax}} \frac{1}{\|\mathbf{w}\|} \min\left[\min_j(\mathbf{w}.\mathbf{x}_j^+ + b), \min_j(-\mathbf{w}.\mathbf{x}_j^- - b)\right] \quad (11)$$

We refer to the proposed method as **RMMH**, for **R**andom **M**aximum **M**argin **H**ashing. In practice, optimal hyperplanes $\mathbf{w}_m$ can be computed easily by a Support Vector Machine (SVM). For kernel spaces, $\mathbf{w}_m$'s can only be expressed as a weighted sum over support vectors, so that the hash function becomes:

$$h(\mathbf{x}) = sgn\left(\sum_{i=1}^{m} \alpha_i^* \kappa(\mathbf{x}_i^*, \mathbf{x}) + b_m\right) \quad (12)$$

where $\mathbf{x}_i^*$ are the m support vectors selected by the SVM ($\mathbf{x}_i^* \in \{\mathbf{x}_j^+, \mathbf{x}_j^-\}$).

4. Parameter M

The number M of samples selected for each hash function is the only parameter of RMMH. Deriving a theoretical optimal value for M unfortunately appears to be a tricky task. It would require to formally model the distribution p_w of $\mathbf{w}_m$ which is an open problem to the best of our knowledge. Some interesting logical guidelines can however be discussed according to three constraints: hashing effectiveness, hashing efficiency and training efficiency.
Let us first discuss **efficiency** concerns. SVM training being based on quadratic programming, an acceptable training cost implies that $M << N$ (even if it is an offline process). But hashing efficiency is even more critical: hash functions usually need to be computed online and the resulting cost is part of the overall search cost. In the linear case, this is obviously not a problem since a single projection on w_m needs to be computed, making our method as efficient as normal projections. In kernel spaces however, the hashing cost is higher since we need to compute as much kernel values as the number of support vectors, for each of the p hash functions. Worst case hashing cost complexity is therefore $O(pM)$. So that an important requirement is that:

$$M << \frac{N}{p} \quad (13)$$

Let us now discuss **effectiveness** concerns related to the two ideal objectives discussed above: *uniformity* and *locality preservation*. The larger the training size M and the better the uniformity. For an extreme value $M = N$ (supposing that the capacity of the classifier is large enough to separate any training set of size N), we would get a perfect uniformity and the probability of irrelevant collisions would be minimal. In other words, the data would be perfectly *shattered*, to re-use Vapnik-Chervonenkis terminology. But this would also lead to overfitting, since close pairs would be shattered as well. On the other extreme, too small training data would increase the error expectation of the classifier and thus degrade the expected uniformity. Data would be not shattered enough. The optimal value for M is thus guided by the a tradeoff between uniformity (data shattering) and locality preservation (generalization). To estimate an approximate max bound on M, we can at least control the risk that close points might be split in the training data itself. Let us consider k-nearest neighbors as relevant matches and any other pair of point as irrelevant. In that case, the expected number of relevant pairs in a random training set of M points is equal to $\frac{M^2 k}{N}$. If we want to have this expected number lower than 1, we get:

$$M < \sqrt{\frac{N}{k}} \quad (14)$$

Interestingly, this value is sub-linear in dataset size N. With this max bound value, the hashing cost complexity $O(pM)$ becomes $O(p\sqrt{\frac{N}{k}})$ which guaranties that it does not become preeminent for very large datasets.

5. Experiments

We used the 3 following datasets to conduce our experiments:

- **SIFT**: A set of about 11 M SIFT features [13] extracted from OxfordBuilding dataset[2] (d=128).
- **ImageNet-BOF**: A set of 1.2 M **B**ags **O**f SIFT **F**eatures (d=1000) provided within ImageNet/PASCAL VOC Large Scale Visual Recognition Challenge (ILSVRC[3]).
- **Wikipedia-DescX**: $X \in [1:5]$, 5 datasets of 237 K global visual features extracted from ImageClef wikipedia dataset [4]. The 5 global visual features are the following: Desc1=HSV histogram (d=120), Desc2=Fourier histogram ([5], d=49), Desc 3=Hough histogram ([5], d=40), Desc4=Weighted color histogram ([20], d=216).

From these initial data we derived different subsets, either to study data size factor or to comply with implementation constraints of some of the state-of-the-art methods experimented in the paper. For example, **SIFT-1M** and **BOF-100K** correspond respectively to subsamples of 1M SIFT and 100K BOF.

All experiments are based on a leave-one-out procedure: 1000 queries are randomly selected from the dataset and removed one by one before being searched. Hash codes are compared with the Hamming distance and ranked accordingly. Performance is measured by the Mean Average Precision of the produced ranked list of results using the exact top k nearest neighbors as ground truth (k=100 when not specified). The metric used for generating the exact k nearest neighbors depends on the experiment and is discussed respectively. Quantization effects related to the Hamming space have to be considered when computing the Mean Average Precision: to compute the precision for a given neighbor v in the ground truth, we first compute the Hamming distance between its hash code and the query hash code. We then consider in the precision's calculation all items having a Hamming distance lower or equal to this value.

For a better understanding of the results, we notice that low MAP values can still provide very interesting performances. Hash codes are indeed usually used only to **filter** the data, either within a hash table or by a direct scanning (as done in VA-file [22] for example). Retrieved results can still be refined or re-ranked afterwards with the original metric. For example, a map of 0.1 for $k = 100$ nearest neighbors means that on average 1000 points would need to be re-ranked.

[2] http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/
[3] http://www.image-net.org/challenges/LSVRC/2010/
[4] http://www.imageclef.org/2010/wiki

5.1. Stability of parameter M

We first conduced an empirical study of M parameter. Figure 3 shows MAP curves when varying M value, for several data configuration. The first conclusion is that M always reaches an empirical maximum, which confirms our discussion of section 4 regarding the tradeoff between uniformity and locality preservation. It also shows that M is rather stable around its maximum and that it evolves only slightly for varying data sizes (from 10K to 1M) and varying number of neighbors (from 10 to 1000). The max bound we introduced in Equation 14 is not always respected by the empirical optimum, but the order of magnitude is correct. In the following experiments of this paper, we used a fixed value $M = 32$.

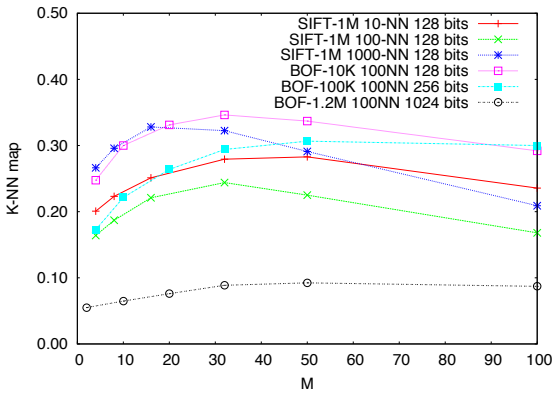

Figure 3. Impact of M parameter for various number of neighbors and various dataset sizes

5.2. Comparison to state-of-the-art

5.2.1 Euclidean space

We first evaluate RMMH in $\mathbb{R}^d$ to allow comparisons to state-of-the-art methods. We used a dataset of 1 M SIFT features (**SIFT-1M**) normalized according to $L2$-norm, so that the exact k-nearest neighbors according to L_2 are equivalent to the exact top k items according to the inner product, the triangular L2 kernel or the RBF kernel. This allowed us to compare a quite large range of methods on this dataset: RMMH was experimented with 3 different kernels: linear, triangular L2 and RBF. For the RBF kernel, we estimated γ on real k-nn samples. We did compare RMMH to two data dependent methods (KLSH [11] and spectral hashing [23]) and two data independent methods (LSH and Random Fourier Features, the RBF-sensitive method of Raginsky et al. [17] discussed in section 2). For Spectral Hashing and KLSH, we used the same number of training samples than the one required by RMMH ($p \times M$). For KLSH we used the L_2 triangular kernel, since we got the best performances with it. For LSH, we used the family sensitive to the inner product (equation 1). For Raginsky's method, we

used the same RBF kernel parameter γ than for RMMH. Results are provided in Figure 4. They show that RMMH clearly outperforms the two other data dependent methods, whatever the used kernel, even the linear one. Thanks to the better independence of RMMH hash functions, the performance are indeed less degrading when increasing the hash code size. Comparisons to data independent methods show that RMMH performs better for a wide range of useful hash code sizes from 1 to about 800 bits which covers many hashing applications. Beyond the quite slight effectiveness gain, the most important point is that RMMH succeed in producing independent enough hash functions. This means that we can expect a good independence as well in kernel spaces (that cannot be addressed by data independent methods).

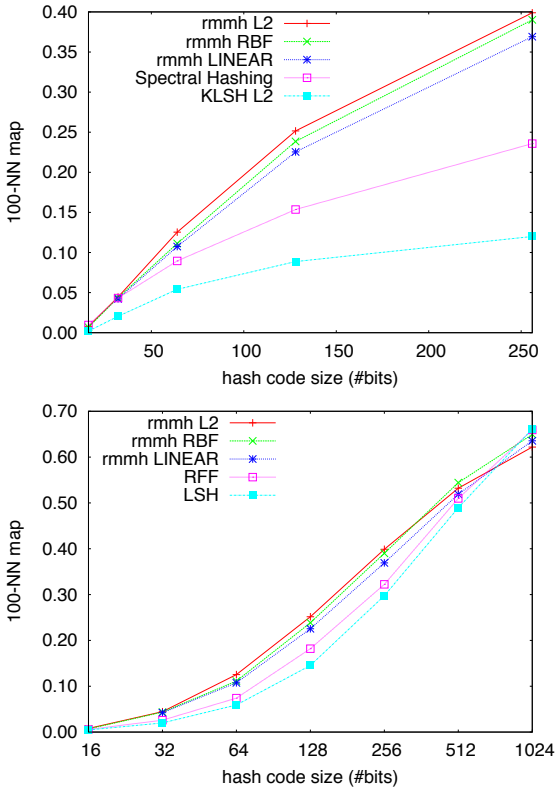

Figure 4. Comparison of RMMH to state-of-the-art methods **(top)** comparison to data dependent methods **(bottom)** comparison to data independent methods

5.2.2 Kernel spaces

We now evaluate the performance of RMMH in other kernel spaces. We compare only to KLSH [11], since it is the only one dealing with any Mercer kernel. We used a 10K subset of **ImageNet-BOF** with a Chi Square kernel and **Wikipedia-HSV** dataset with a Generalized Histogram Intersection kernel (GHI, [1]). Results of Figure 5 confirm that RMMH outperforms KLSH.

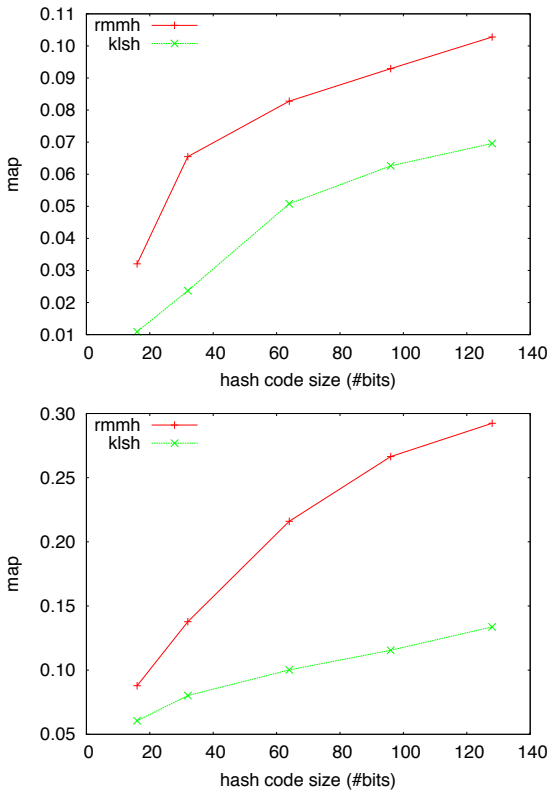

Figure 5. Comparison between RMMH and KLSH in 2 different kernel space: **(top)** GHI kernel (WIKIPEDIA-HSV dataset) **(bottom)** Chi2 kernel (ImageNet-BOF-10K)

5.3. Image retrieval performances

We now evaluate the performance of RMMH for image retrieval. The concern here is not to retrieve the k-nearest neighbors in the original feature space but the most relevant images. We therefore used the full **ImageNet-BOF** dataset with associated labels (1000 categories). We still used a leave-on-out procedure over 1000 random queries but we now run a k-nn classifier on the top k results retrieved by each evaluated method (i.e. the score of each category is determined by the number of instances retrieved in the top k hash codes. k was set up to 1000). As suggested in ILSVRC challenge, we relaxed the classification tolerance to the five best retrieved classes (recognition rate@5). We first did evaluate RMMH with 3 different kernels: Linear, Chi Square and Triangular L2. Results of table 1 show that the best kernel is the Chi Square one. However the gain over the linear kernel is very slight whereas the hashing complexity is much larger (as discussed in section 4). Overall, the linear kernel appears to be the best choice. Furthermore, it allows an interesting comparison between RMMH and the LSH family sensitive to the inner product (equ. 1).
In this way, Figure 6 presents the classification rate of RMMH and LSH for varying hash code sizes. The horizon-

Kernel	Linear	Chi Square	Triang L_2
recognition rate@5	0.149	**0.150**	0.135

Table 1. Classification performance of RMMH with 3 different kernels (**ImageNet** dataset, 128 bits)

tal line corresponds to the classification rate obtained with the exact inner product in the original feature space. We can first remark that the gain of RMMH over LSH is much larger than previous experiments (when searching approximate k-nearest neighbors). That means that RMMH provides a better embedding of the underlying data structure, whereas LSH only converges to the original metric. RMMH is even better than the exact distances for hash code sizes larger than 600 bits. Finally, with only 512 bits (64 bytes), RMMH hash codes provide equivalent performances than the original 1000 dimensional bag-of-features.

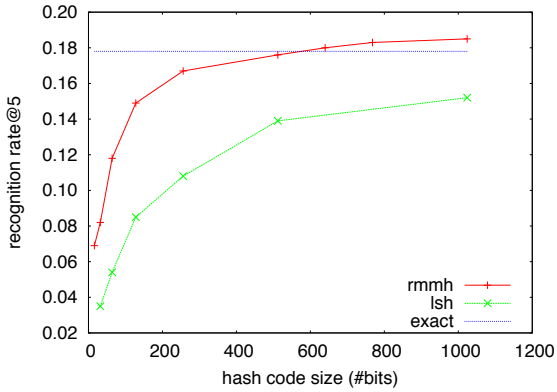

Figure 6. Classification performances on **ImageNet-BOF**

5.4. Indexing performances

We now evaluate RMMH in terms of indexing performances, using a **multi-probe** Locality Sensitive Hashing framework to reduce memory usage. We used the a posteriori multi-probe LSH method of Joly at al. [9] (AMP-LSH). It allows using any hash function whereas other multi-probe methods are focused on L_2. In this experiment, we only used RMMH to construct the hash table and we kept the original data in the buckets. Of course, we can also replace the original features by RMMH hash codes to achieve even better speed-up and memory saving (as presented below). But our first goal here is to study the partition induced by RMMH for constructing hash tables. We present only the results on **ImageNet-BOF** which is more challenging than the **SIFT** dataset due to the higher dimension. It is important to notice that the sparsity of this dataset is rather weak (about 2/3 of null components), which means that an inverted list would fail to provide consistent efficiency gains (up to about 3 compared to exhaustive scan).

The used metric is the inner product and we did vary the dataset size from 100 features to 1 M features. We used the following AMP-LSH settings: quality control parameter $\alpha = 0.7$, number of hash tables $L = 1$, number of searched nearest neighbors $k = 100$. The depth p of each table (i.e. the hash code size) depends on the dataset size thanks to the following empirical formula $p = log_2(N) + 5$ (values ranged from 11 to 25). RMMH was used with the linear kernel and $M = 40$. It was compared to the LSH family sensitive to the inner-product. For hardware independent comparisons, performances of the exhaustive scan are reported as well.

Results are reported in Figure 7 and table 2. The plot shows that both LSH and RMMH achieve sub-linear search time in data size, providing consistent efficiency gains over the linear scan (which is not trivial with a dimension equal to 1000). But RMMH clearly outperforms LSH (as much as LSH outperforms the exhaustive scan). The sub-linearity coefficient of RMMH is indeed higher, leading to increasing efficiency gains when the size increases. That confirms again that RMMH closely fit the data distribution while keeping a good independence between the hash functions. For the full dataset of 1M BoF (see table 2), RMMH is finally 37 times faster than exhaustive scan and 5 times faster than LSH.

In table 2, we finally report the performances obtained on the full dataset when replacing the original features by RMMH hash codes. We used 1024 bits for the hash codes and returned the 10K nearest hash codes (before re-ranking with the exact inner product). This combined strategy allows to divide the search time by 5 and the memory usage by 13.

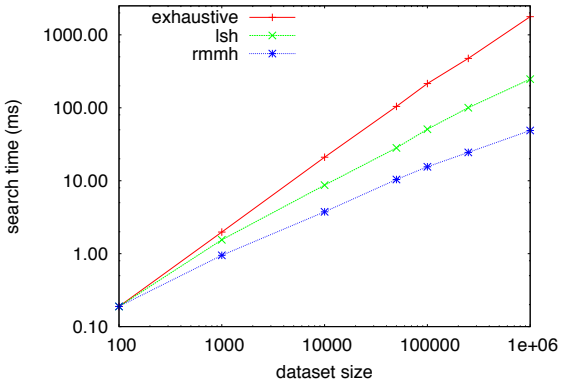

Figure 7. search time vs data size - comparison of RMMH to LSH and exhaustive scan

5.5. Conclusion

We introduced a new hashing function family suitable for any kernel space and providing excellent performances

method	time (ms)	NN recall	Mem (Gb)
Exhaustive	1777	1.0	5.05
LSH index	247	0.67	5.11
RMMH index	**49**	0.69	5.11
RMMH index + sketch	**10**	0.62	**0.39**

Table 2. Indexing and Search statistics on **ImageNet-BOF**

in Euclidean space. Contrary to previous data dependent methods, we did not focus on boosting the collision probability of close points. We rather try to minimize the collision probability of irrelevant pairs by boosting the scattering of the data. We therefore suggest to train purely random splits of the data, regardless the closeness of the training samples or any kind of supervision. Experiments did confirm that the resulting hash functions are consistently more independent than other data dependent methods. On the other side, the use of large margin classifiers prevents from overfitting and provides good generalization capacities for neighboring data. Image retrieval experiments did show that no more than 64 bytes are enough to achieve similar performances than exact distances in a 1000-dimensional feature space. Indexing performances finally confirmed that RMMH produces better partitions than purely random projections.

In future works, we will continue investigating a theoretical modeling of RMMH. This would help defining accurate bounds for M parameter and provide a better understanding on how data structures are embedded (density, symmetry, etc.). We believe that RMMH could be useful for many other goals including dimensionality reduction, independent component analysis or feature selection.

References

[1] S. Boughorbel, J.-P. Tarel, and N. Boujemaa. Generalized Histogram Intersection Kernel for Image Recognition. In *IEEE Int. Conf. on Image Processing (ICIP'05)*, 2005. 878

[2] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *ACM STOC*, pages 380–388, 2002. 874

[3] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *Proceedings of the British Machine Vision Conference*, 2008. 873

[4] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Symposium on Computational geometry*, pages 253–262, 2004. 873

[5] M. Ferecatu. Image retrieval with active relevance feedback using both visual and keyword-based descriptors. In *PhD thesis, University of Versailles*, 2005. 877

[6] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Int. Conf. on Very Large Data Bases*, pages 518–529, 1999. 873

[7] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In A. Z. David Forsyth, Philip Torr, editor, *European Conf. on Computer Vision*, volume I of *LNCS*, pages 304–317. Springer, oct 2008. 873

[8] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2010. to appear. 873, 874

[9] A. Joly and O. Buisson. A posteriori multi-probe locality sensitive hashing. In *MM '08: Proceeding of the 16th ACM Int. Conf. on Multimedia*, pages 209–218, New York, NY, USA, 2008. ACM. 879

[10] A. Joly, C. Frélicot, and O. Buisson. Feature statistical retrieval applied to content-based copy identification. In *Int. Conf. on Image Processing*, 2004. 873

[11] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *IEEE Int. Conf. on Computer Vision (ICCV*, 2009. 873, 874, 877, 878

[12] R.-S. Lin, D. Ross, and J. Yagnik. Spec hashing: Similarity preserving algorithm for entropy-based coding. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 848 –854, june 2010. 874

[13] D. G. Lowe. Object recognition from local scale-invariant features. In *Int. Conf. on Computer Vision*, pages 1150–1157, 1999. 877

[14] Y. Mu, J. Shen, and S. Yan. Weakly-supervised hashing in kernel space. In *Computer Vision and Pattern Recognition*, pages 3344 –3351, june 2010. 874

[15] L. Paulevé, H. Jégou, and L. Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognition Letters*, 31(11):1348 – 1358, 2010. 873

[16] S. Poullot, O. Buisson, and M. Crucianu. Z-grid-based probabilistic retrieval for scaling up content-based copy detection. In *CIVR '07: Proceedings of the 6th ACM Int. Conf. on Image and video retrieval*, pages 348–355, 2007. 873

[17] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, 2009. 873, 874, 877

[18] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML '07: Proceedings of the 24th Int. Conf. on Machine learning*, pages 791–798, New York, NY, USA, 2007. ACM. 873, 874

[19] G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. MIT Press, 2006. 873

[20] C. Vertan and N. Boujemaa. Upgrading color distributions for image retrieval can we do better? In *Advances in Visual Information Systems*, volume 1929, pages 597–606, 2000. 877

[21] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. *Computer Vision and Pattern Recognition*, 0:3424–3431, 2010. 874

[22] R. Weber, H. J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Int. Conf. on Very Large Data Bases*, pages 194–205, 1998. 873, 877

[23] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008. 873, 874, 877