MULTI-FEATURE HASHING BASED ON SNR MAXIMIZATION

Honghai Yu †∗ and Pierre Moulin †∗

†ECE Dept., University of Illinois at Urbana-Champaign, USA
∗Advanced Digital Sciences Center, Singapore

ABSTRACT
Hashing algorithms which encode signal content into compact binary codes to preserve similarity, have been extensively studied for applications such as large-scale visual search. However, most existing hashing algorithms work with a single feature type, while combining multiple features is helpful in many vision tasks. In this paper, we propose two multi-feature hashing algorithms based on signal-to-noise ratio (SNR) maximization, where a globally optimal solution is obtained by solving a generalized eigenvalue problem. The first one jointly considers all feature correlations and learns uncorrelated hash functions by solving a generalized eigenvalue problem. The proposed algorithms perform favorably compared to other state-of-the-art multi-feature hashing algorithms on several benchmark datasets.

Index Terms— Hashing, multi-feature, signal-to-noise ratio

1. INTRODUCTION
The explosive growth of vision data in applications such as web-scale image retrieval demands fast similarity search and efficient indexing techniques. This has motivated the recent studies on robust hashing algorithms, where visual content is encoded into compact binary hash codes which allows real-time search. The hash codes must be robust to various content-preserving distortions, while being discriminative enough to distinguish between perceptually different signals.

Various hashing algorithms have been proposed based on hand-crafted image features [1, 2, 3, 4]. Recently, there has been a surge of interest in learning-based similarity preserving hashing algorithms. Improving on the well-known locality sensitive hashing (LSH) [5], many hashing algorithms have been successfully applied to learn binary codes that preserve similarity in a metric space [6, 7, 8] or preserve semantic similarity [9, 10, 11, 12]. Despite this progress, most existing algorithms only utilize one type of feature. However, different features extracted from the same underlining signal can be complementary to each other and boost system performance. For instance, combining raw image pixels with patch-based features gives better face recognition performance [13]. In near-duplicate video retrieval, combining features such as color histogram and local binary pattern yields a significant performance improvement [14]. Recently, fusion of RGB and depth features has been shown to improve system performance. For instance, combining raw image pixels with patch-based features gives better face recognition performance [13]. In near-duplicate video retrieval, combining features such as color histogram and local binary pattern yields a significant performance improvement [14].

The literature on multi-feature hashing is very limited. In [15], the final hash function is the convex combination of individual linear hash functions on different features. The algorithm of [17] learns nonlinear hash function on each feature using the kernel trick. In [16], features concatenated and binary codes are generated using learned hyperplanes to simultaneously preserve similarities in each individual feature space. In [18], hash bits are selected from a pool of hash bits generated by different hashing algorithms on different features. The more recent works on multi-feature kernel hashing (MFKH) [17] and hash bit selection (HBS) [18] have shown superior performance over previous art [16, 14].

In this paper, we propose two multi-feature hashing algorithms based on signal-to-noise ratio (SNR) maximization. This work is an extension of our recent paper [19], which shows that maximizing SNR is equivalent to minimizing hashing error probability under a Gaussian model. The first algorithm concatenates different features as one and jointly learns uncorrelated hash functions that maximize SNR. We call this algorithm SNR joint hashing (SNR-JH). The second algorithm separately learns hash functions on each feature based on SNR maximization and the overall hash functions are selected according the SNR associated with each hash function. We call this selection procedure SNR selection hashing (SNR-SH). Both SNR-JH and SNR-SH outperform the state-of-the-art MFKH and HBS significantly on several benchmark datasets.

2. BACKGROUND AND RELATED WORK
In multi-feature hashing, the basic task is to learn a mapping \( h(x) = \{h_1(x), \ldots, h_k(x)\} \in \{\pm 1\}^k \) that projects an input \( x = \{x^{(1)}, x^{(2)}, \ldots, x^{(M)}\} \) consisting of \( M \) features vectors of respective dimensions \( d_1, \ldots, d_M \), onto \( K \)-dimensional binary codes, while preserving some notion of similarity. A simple way to learn this mapping is to concatenate different features, treat them as one, and apply a single-feature hashing algorithm. However, without considering the different statistical properties from different features, most existing single-feature hashing algorithms would perform poorly (often worse than the best single feature). In this section, we briefly summarize two state-of-the-art multi-feature hashing algorithms, MFKH and HBS. They are used as the benchmark algorithms in our experiments.

Multi-Feature Kernel Hashing (MFKH): Inspired by multiple-kernel learning, MFKH [17] formulates the hashing problem as a similarity preserving-hashing with linearly combined multiple kernels. In particular, each input \( x^{(m)} \) is implicitly mapped to an element of the high-dimensional (possibly infinite dimensional) feature space \( \mathcal{F} \) by an embedding function \( \phi_m : \mathbb{R}^{d_m} \to \mathcal{F} \), and the overall embedding \( \phi(x) = [\phi_1^T(x^{(1)})^T, \ldots, \phi_M^T(x^{(M)})^T]^T \) is a weighted concatenation of \( \{\phi_m(x^{(m)})\}_m \). Therefore, the kernel function \( K_{ij} = \phi(x)^T \phi(x_i) = \sum_{m=1}^{M} \alpha_m K^{(m)}_{ij} \) is a linear combination of the kernels on different features. With the

This work was supported by NSF grant CCF 12-19145 and by A*STAR Graduate Academy, Singapore.
embedding function \( \phi(\cdot) \), the \( k \)-th hash function is defined as
\[
h_k(x) = \text{sgn}(\gamma_k^T \phi(x) + b_k),
\]
where \( \gamma_k \) is a projection vector in the high-dimensional feature space and \( b_k \) is the bias. By using the kernel trick, one only needs to evaluate kernel functions instead of working with the high dimensional feature embedding \( \phi(\cdot) \).

Similarly to spectral hashing (SH) [6], MFKH learns the hash codes for \( N \) training data to preserve a similarity matrix \( S \in \mathbb{R}^{N \times N} \) while satisfying the balance and uncorrelated constraints. For hashing with multiple features, \( S \) is constructed using semantic labels because labels are consistent across different features. As the \( S \) function is nondifferentiable and nonconvex, MFKH replaces it with the identity function to obtain an solution by alternative minimization between \( \mu \) and \( (V, b) \). Though approximating the \( S \) function with the identity function makes the learning problem tractable, it obviously introduces large approximation error when the magnitude is large.

Hash Bit Selection (HBS): Unlike MFKH where bits are generated from the combination of all features, each bit in HBS [18] is derived from only one type of feature. In particular, HBS selects \( K \) hash bits (corresponding to \( K \) hash functions) from a pool of candidate bits generated by a given hashing algorithm using different features. In HBS, the selection criteria are similarity preservation and independence. As in spectral hashing and MFKH, similarity preservation means the binary codes should preserve the original similarity measure \( S \) between training data points in the Hamming space. HBS uses a loss function based on spectral embedding loss [6]. In [6, 11, 17, 18], the independence of hash bits is considered a desirable property for generating compact codes, and HBS uses pairwise mutual information between hash bits as a regularizer to penalize hash functions generating highly correlated hash bits. Combining these two criteria, HBS formulates the bit selection problem as a quadratic programming problem. By relaxing the discrete constraint, the minimization problem can be solved using replicator dynamics [18].

3. SNR Maximization Hashing

Signal-to-noise ratio (SNR) has been traditionally used as the performance measure in many applications, such as lossy compression [20], matched filter [21], relay functionality in memoryless relay networks [22], and beamforming in narrowband sensor arrays [23, 24]. The usefulness of SNR in hashing algorithms has recently been established in [19], where maximizing SNR in the projection direction was shown to be equivalent to minimizing the hashing error probability in a Gaussian model. Motivated by the analysis, SNR maximization hashing (SNR-MH) was proposed and demonstrated significant performance gain over existing hashing algorithms on both synthetic and real datasets [19]. In this section, we summarize the SNR-MH algorithm (the analysis of SNR-MH can be found in [19]). Building on SNR-MH, we present two multi-feature extensions in the next two sections.

Similarly to many other hashing algorithms [9, 11, 25, 12, 26], the training dataset for SNR-MH requires weakly supervised information in the form of \( N \) similar feature pairs \( \{(x_i, \hat{x}_i)\}_{i=1}^N \) where \( x_i, \hat{x}_i \in \mathbb{R}^d \). Without loss of generality, we assume that feature vectors are centered, i.e., \( \sum_{i=1}^N x_i = \mathbf{0} \) and \( \sum_{i=1}^N \hat{x}_i = \mathbf{0} \). A pair \( (x_i, \hat{x}_i) \) is said to be similar if \( \hat{x}_i \) is a distorted version of \( x_i \), or \( x_i \) and \( \hat{x}_i \) share the same class label. SNR-MH assumes \( \hat{x}_i \) is the sum of \( x_i \) and independent noise \( z_i \):
\[
\hat{x}_i = x_i + z_i.
\]

Dissimilar \( x_i \) and \( \hat{x}_i \) are assumed to be independent. Operating under this assumption, SNR-MH does not require dissimilar pairs for training unlike algorithms in [9, 11, 25, 12, 26].

A hash function \( h_k \) is parameterized by a projection vector \( w_k \in \mathbb{R}^d \)
\[
h_k(x) = \text{sgn}(w_k^T x).
\]
The SNR in the direction \( w_k \) is defined as
\[
\text{SNR}_k \triangleq \frac{w_k^T C_X w_k}{w_k^T C_Z w_k}
\]
where \( C_X = \frac{1}{N} \sum_{i=1}^N x_i x_i^T \) and \( C_Z = \frac{1}{N} \sum_{i=1}^N (\hat{x}_i - x_i)(\hat{x}_i - x_i)^T \) are the empirical signal covariance and empirical noise covariance matrices, respectively.

The goal is to learn a \( d \times K \) transformation matrix \( W = [w_1, \ldots, w_K] \) such that the transformed feature vector is uncorrelated and the SNR at each projection is maximized. Mathematically, the projection vectors \( w_k, k = 1, 2, \ldots, K \), are sequentially learned via the following optimization:
\[
w_k = \arg \max_w \frac{w_k^T C_X w_k}{w_k^T C_Z w_k}
\]
subject to
\[
w_k^T C_X w_j = 0, \quad \forall j < k
\]
\[
w_k^T C_Z w_j = 0, \quad \forall j < k
\]
\[
w_k^T C_Z w_k = 1,
\]
where the last constraint is to normalize the projected noise to unit power so the solution is unique. To ensure \( C_Z \) is invertible, a small constant is often added to the diagonal entries of \( C_Z \), i.e., \( C_Z \) is replaced with \( C_Z + \alpha I \). The solution of (4) is given by the \( K \) eigenvectors corresponding to the first \( K \) largest eigenvalues of the generalized eigenproblem [27]
\[
C_X w = \gamma C_Z w,
\]
where \( \gamma \) is the eigenvalue (to be interpreted as the SNR in the direction \( w \)). Bits generated from learned high-SNR projections are very informative and have shown superior performance on both synthetic and real datasets [19].

4. SNR Joint Hashing

We propose the first SNR-based multi-feature hashing algorithm, SNR joint hashing (SNR-JH), to learn hash functions from the combination of all features. Let \( d = \sum_{m=1}^M d_m \). In a multiple feature setting, \( d \)-dimensional feature vectors \( x = (x^{(1)}, x^{(2)}, \ldots, x^{(M)}) \) and \( \hat{x} = (\hat{x}^{(1)}, \hat{x}^{(2)}, \ldots, \hat{x}^{(M)}) \), both centered, are formed by concatenating \( M \) different feature vectors. Similarly to SNR-MH, the goal of SNR-JH is to learn \( K \) projection vectors from \( M \) different features such that projected feature vector components are uncorrelated and SNR at each projection is maximized.

Following composite hashing in [19] and MFKH, hash function \( h_k \) in SNR-JH is parameterized by projection vectors \( w_k^{(m)} \) and non-negative weights \( \mu_m \), for \( m = 1, \ldots, M \). The different features are then linearly combined, and the sign of the weighted sum is the output:
\[
h_k(x) = \text{sgn}\left(\sum_{m=1}^M \mu_m w_k^{(m)} x^{(m)}\right).
\]
In both composite hashing and MFKH, learning \( \{ w_{\gamma}^{(m)} \}_{m=1}^{M} \) and \( \{ \mu_m \}_{m=1}^{M} \) is done via alternating optimization as the objective functions are nonconvex with respective to \( \{ w_{\gamma}^{(m)} \}_{m=1}^{M} \) and \( \{ \mu_m \}_{m=1}^{M} \) jointly. However in SNR-JH, we show that the weights \( \{ \mu_m \}_{m=1}^{M} \) are redundant and can be incorporated into \( \{ w_{\gamma}^{(m)} \}_{m=1}^{M} \). Denote by

\[
w_k = [\mu_1 w_1^{(1)^T}, \mu_2 w_2^{(2)^T}, \ldots, \mu_M w_M^{(M)^T}]^T \in \mathbb{R}^d
\]

the \( k \)-th projection direction. Then we can rewrite (6) as

\[
h_k(x) = \text{sgn} (w_k^T x).
\]

Also denote by

\[
C_X = \begin{pmatrix} C^{(11)}_X & \cdots & C^{(1M)}_X \\
\vdots & \ddots & \vdots \\
C^{(M1)}_X & \cdots & C^{(MM)}_X
\end{pmatrix} \in \mathbb{R}^{d \times d}
\]

and

\[
C_Z = \begin{pmatrix} C^{(11)}_Z & \cdots & C^{(1M)}_Z \\
\vdots & \ddots & \vdots \\
C^{(M1)}_Z & \cdots & C^{(MM)}_Z
\end{pmatrix} \in \mathbb{R}^{d \times d}
\]

the full covariance matrices for the \( M \) feature vectors and the noise vectors respectively, where \( C^{(mn)}_{\gamma} = \frac{1}{N} \sum_{i=1}^{N} \langle x^{(m)}_i, x^{(n)}_i \rangle \) and \( C^{(mn)}_{\tilde{\gamma}} = \frac{1}{N} \sum_{i=1}^{N} (\tilde{x}^{(m)}_i - x^{(m)}_i)(\tilde{x}^{(n)}_i - x^{(n)}_i)^T \), \( m, n = 1, 2, \ldots, M \).

Therefore, \( \mu_k, k = 1, 2, \ldots, K \) can be learned by solving (4) with \( C_X \) and \( C_Z \) given by (8) and (9) respectively. Clearly, SNR-JH does not need to learn the weights \( \{ \mu_m \}_{m=1}^{M} \) explicitly. The optimal linear combination of different features is automatically determined in SNR-JH. In Section 6, we illustrate how SNR-JH allocates different weights to different features on a real dataset.

5. SNR SELECTION HASHING

The second SNR-based multi-feature hashing is a selection procedure, termed SNR selection hashing (SNR-SH). In SNR-SH, \( K \) projection directions are learned separately from each feature by solving (4). Unlike HBS where the selection criteria are similarity preservation and independence, SNR-SH uses the SNR in the projection direction as the only selection criterion.

On the \( m \)-th feature, we extract \( K \) projections \( \{ w_{\gamma}^{(m)} \}_{\gamma=1}^{K} \) as the top \( K \) eigenvectors of the generalized eigenproblem

\[
C^{(m)}_X w^{(m)} = \gamma^{(m)} C^{(m)}_Z w^{(m)},
\]

where \( \gamma^{(m)}_{\gamma} \) is the SNR in the direction \( w^{(m)}_{\gamma} \), in descending order \( \gamma^{(m)}_{1} \geq \gamma^{(m)}_{2} \geq \ldots \geq \gamma^{(m)}_{K} \). In the candidate pool of projection directions \( w^{(m)}_{\gamma}, k = 1, \ldots, K, m = 1, \ldots, M \), SNR-SH selects \( K \) projections corresponding to the \( K \) largest \( \gamma^{(m)}_{\gamma}, k = 1, \ldots, K, m = 1, \ldots, M \).

It follows directly from (4) that \( \{ w^{(m)}_{\gamma} \}_{\gamma=1}^{K} \) learned on the \( m \)-th feature are uncorrelated projection directions, i.e., \( w^{(m)}_{\gamma} C^{(m)}_X w^{(m)}_{\gamma} = 0 \) for \( i \neq j \). However, \( w^{(m)}_{\gamma} C^{(m)}_X w^{(m)}_{\gamma} \) is not zero for \( m \neq n \) in general unless features \( m \) and \( n \) are uncorrelated, i.e., \( C^{(mn)}_X \) is the zero matrix. Therefore, bits generated from different features are in general correlated. In contrast, projection directions learned by SNR-JH are uncorrelated.

To see the connection between SNR-SH and SNR-JH, let \( \tilde{C}_X \) and \( \tilde{C}_Z \) be the covariances obtained by forcing all off-diagonal sub-matrices of \( C_X \) and \( C_Z \) to zero. Then SNR-SH is equivalent to finding the top \( K \) eigenvectors of the following generalized eigenproblem

\[
\tilde{C}_X w = \gamma \tilde{C}_Z w.
\]

As SNR-JH jointly considers all the correlation structures among different features, more high SNR projections can be obtained than SNR-SH. On the other hand, SNR-JH needs to estimate considerably more parameters (all \( M \times M \) sub-matrices \( C_{\gamma}^{(mn)} \) and \( C_{\tilde{\gamma}}^{(mn)} \)), which makes SNR-JH computationally less attractive than SNR-SH.

Projection learning in SNR-JH and SNR-SH is carried out by solving generalized eigenproblems, which can be done in less than one minute from a training dataset of 100,000 feature vectors of 1,000 dimension on a standard office desktop. Moreover, unlike MFKH and HBS where one needs to tune multiple parameters, both SNR-JH and SNR-SH are parameter-free, which makes training much easier.

6. EXPERIMENTS

6.1. Datasets and Features

We have evaluated our algorithms on five benchmark datasets: (1) the University of Kentucky (UK) object recognition dataset [28]; (2) the MNIST digit dataset [29]; (3) CIFAR-10 [30]; (4) Scene-15 [31]; (5) Caltech-101 [32]. Our algorithms consistently outperform MFKH and HBS on all datasets. Due to space limitation, we only discuss results for the UK object recognition dataset.

There are 2,550 different objects in the UK object recognition dataset, each of which contains four images taken under different viewpoint, orientation, scale or lighting conditions. Images in the dataset are 640 \times 480 pixels. Each image is represented by a 512-dimensional GIST feature [33] and 512-dimensional bag-of-SIFT-features (BoSF) [34]. GIST features are computed at 8 orientations and 4 different scales, resulting in 512-dimensional feature vectors.

In BoSF, SIFT descriptors are first extracted from every 480 \times 480 pixel block, and then the best one is kept. For MFKH, we consider three different kernels: linear, Gaussian and Chi-Square.

6.2. Protocols and Baseline Algorithms

We randomly take one image from each object as query and the rest are used as database and training set. For evaluation, we compute the Recall@K for each query, where \( K \) is the number of top retrieved images based on the Hamming distance between the query and database images, and we report the average over all queries. We also compute the mean average precision (mAP), or the area under the precision-recall curve, for different code lengths.

We compare SNR-JH and SNR-SH to two state-of-the-art multi-feature hashing algorithms MFKH and HBS using code provided by the authors. We also compare with other well-known single-feature hashing algorithms, iterative quantization (ITQ) [8] and semi-supervised hashing (SSH) [11], where different features are concatenated as one and treated as a single feature. ITQ and SSH are also served as the base algorithms for HBS, where a candidate pool of hash function are generated from each feature by ITQ and SSH, and later chosen by HBS. Whenever there are parameters in these baseline algorithms, we try a few candidate choices and report the best one. For MFKH, we consider three different kernels: linear, Gaussian and Chi-Square.
6.3. Results and Discussions

Fig. 1 compares all the multi-feature hashing algorithms based on the recall at different numbers of top retrieved samples. First notice that both ITQ and SSH perform poorly on this multi-feature setting, but combining with the selection procedure HBS, we see a large performance improvement. Our proposed SNR-JH and SNR-SH both outperform the next best MFKH algorithm by a large margin.

Fig. 2 shows mAP as a function of code size. We see a strong upward trajectory for all algorithms. Again, our proposed SNR-JH and SNR-SH algorithms consistently outperform the other algorithms, and the performance gap to the next best algorithm widens as more bits are used. Moreover, SNR-SH consistently outperforms SNR-JH. This may due to the following two reasons: (i) as SNR-JH jointly considers all the correlation structures among different features, more high SNR projections can be obtained than SNR-SH; (ii) the SNR-JH is able to learn uncorrelated projection directions, while projections from different features are correlated in SNR-SH.

We also compare our multi-feature SNR-JH and SNR-SH algorithms with the single-feature counterpart SNR-MH on the UK object recognition dataset. As shown in Fig. 3, performance with multiple features is much better than with single feature, which indicates that our multiple feature algorithms help improve retrieval performance by exploiting the complementary information between features.

To illustrate how SNR-JH allocates different weights to different features, Fig. 4 shows the ratio between energy allocated to feature 1 and the total energy of projection $w_k$, where $\{w_k\}$‘s are learned from the UK object recognition dataset. As shown in Fig. 3, using feature 1 (the BoSF feature) alone performs much better than feature 2 (the GIST feature) for up to 128 bits, which explains why SNR-JH allocates more than 90% of the energy to feature 1 for the first few hundreds of projections. After around $k = 700$, the ratio falls rapidly, which indicates that low-SNR projections comprises of mostly feature 2. Moreover, SNR-SH selects 47 projections from BoSF and 17 projections from GIST for the 64-bit codes.

7. CONCLUSIONS

We have proposed SNR-JH and SNR-SH, two multi-feature hashing algorithms based on SNR maximization. SNR-JH jointly learns projection directions from all features, while SNR-SH selects projection directions from a pool of candidate projections generated from each individual feature. Despite the simple linear model (1) and the simple training procedure (solving generalized eigenproblems and parameter-free), SNR-JH and SNR-SH have demonstrated superior retrieval performance over the state-of-the-art MFKH and HBS on a large-scale object retrieval task with 2,350 different objects.
8. REFERENCES


