SUPERVISED HASHING WITH JOINTLY LEARNING EMBEDDING AND QUANTIZATION

Hao Zhu[1]

3M Cogent
Beijing, China

Feng Wang[2], *Xiang Xiang*[2], *Trac D. Tran*[3]

[2]Dept. of Computer Science [3]Dept. of Electrical & Computer Engineering
Johns Hopkins University, 3400 N. Charles St, Baltimore, MD 21218, USA

ABSTRACT

Compared with unsupervised hashing, supervised hashing commonly illustrates better accuracy in many real applications by leveraging semantic (label) information. However, it is tough to solve the supervised hashing problem directly because it is essentially a discrete optimization problem. Some other works try to solve the discrete optimization problem directly using binary quadratic programming, but they are typically too complicated and time-consuming while some supervised hashing methods have to solve a relaxed continuous optimization problem by dropping the discrete constraints. However, these methods typically suffer from poor performance due to the errors caused by the relaxation manner. In this paper based on the general two-step framework: learning binary embedded codes and learning hash functions, we propose a new method to solve the problem introduced by relaxing the cost function. Inspired by the property of rotation invariance of learning embedding features, our method tries to jointly learn similarity-preserving representation and rotation transformation for better quantization alternatively. In experiments, our method shows significant improvement. Compared with the methods based on discrete optimization our methods obtains the competitive performance and even achieves the state-of-the-art performance in some image retrieval applications.

Index Terms— Learning to Hash, Supervised Hashing, Image Retrieval

1. INTRODUCTION

Approximation Nearest Neighbor (ANN) search plays a fundamental role in today's big data age. With the explosive growth of data on the Internet, it is difficult to employ traditional Nearest Neighbor (NN) search due to computational cost and network communication. Actually, in many real applications, the exact NN search is unnecessary for every possible query. So it makes us leverage hashing technique to improve speed and reduce memory capacity. For example, we only need 4MB memory to store 1 million points with 32bit code, and few millisecond to search all points. Thus learning to hash is an efficient way for this goal.

There are two main categories of methods for hashing methods: data-independent methods and data-dependent methods. Representative data-independent methods include locality sensitive hashing (LSH) [1, 2] and shift invariant kernel hashing (SIKH) [3]. Compared with the data-dependent approaches, data-independent ones usually need longer codes to achieve similar performances. Therefore, date-independent methods do not work well in relatively low bits quantization. To address the shortcoming of data independent methods, recent research pays a lot of attentions to data-dependent methods. Their hash functions are learned from training data via data-driven methods, which can be further divided into two categories: unsupervised methods [4, 5, 6, 7] and supervised hashing [8, 6, 9, 10, 11, 12, 13]. And the latter has attracted more and more attention in recent years because it has shown better accuracy than other hashing methods in many real world tasks. Supervised learning [8, 14, 15, 13] is also able to utilize supervised (label) information to learn the hash codes and thus make it more compact and semantic.

However, it is not easy to solve the learning to hash problem because it is essentially a discrete optimization problem. Most existing supervised hashing methods solve a relaxed continuous optimization problem after dropping the discrete constraints. Although the relaxed continuous optimization has some merits such as low computational complexity, these methods typically suffer from poor performances due to the errors caused by the relaxation. Some other methods [16, 15] directly solve the discrete optimization problem achieving the state-of-art performance. However, they are typically time-consuming and unscalable compared with hashing methods using relaxed continuous optimization.

In this paper, we argue the reason that relaxed continuous optimization may suffer a poor performance due to the neglect of the quantization error in the cost function. Without an appropriate cost term to penalize quantization error, the solution is likely to be that the dimension with larger variance typically carries more information, but in the quantization process, the same number of bits is used for each dimension, which yields a poor performance. To address this problem, we propose a novel algorithm which simultaneously takes both quantization and distance approximation into account. It solves the problem of performance loss caused by using the relaxed continuous optimization. Experiments on three different datasets with semantic demonstrate that the performance of our method is comparable with the state-of-the-art methods including the ones based on discrete optimization. The advantage is even huger for the dataset with a large number of categories (*e.g.*, CIFAR-100 *vs.* CIFAR-10).

2. METHODOLOGY

In this section, we will first discuss the learning binary embedding in the general two-step approach [12]. Then we describe the issues of relaxed continuous optimization, which can be interpreted under a probability model. Lastly, we present the details of our hashing model, including the model formulation and learning algorithms, and explain the advantage of our method.

2.1. Problem Definition

Suppose there are n images $X = \{x_i\}_{i=1}^n$ where x_i is the feature vector of point i. x_i can be the hand-crafted features or the raw images in a retrieval problem. Besides the feature vectors, the training set also contains a set of pairwise labels $S = \{s_{ij}\}$ with $s_{ij} \in \{1, -1\}$, where $s_{ij} = 1$ if x_i and x_j belong to the same class or $s_{ij} = -1$ otherwise.

The supervised hashing with pairwise labels is to learn a binary code $b_i \in \{-1, 1\}^c$ for each point x_i, where c is the code length. Instances with similar labels should have similar binary codes $B = \{b_i\}_{i=1}^n$ which preserve the similarity in S. More specifically, if $s_{ij} = 1$, the hamming distance between binary codes b_i and b_j should be low. Otherwise if $s_{ij} = -1$, the hamming distance between binary codes b_i and b_j should be high. The goal of supervised hashing is to learn a binary code matrix $B = [b_1, ..., b_i, b_j, ..., b_n]$ and then we can learn a function for the binary code as $b_i = h(x_i) = [h_1(x_i), h_2(x_i), ..., h_c(x_i)]^T$, where $h(x_i)$ is the hash function to learn.

2.2. Supervised Hashing with Pairwise Labels

In most of the existing supervised hashing methods, the side information is in the form of pairwise labels that indicate the semantic similarities or dissimilarities on image pairs. The loss functions in these methods are thus designed to preserve the pairwise similarities of images. Let $\theta_{ij} \in \Theta$ denote half of the inner product between two binary codes $b_i, b_j \in \{-1, 1\}^c$, where

$$\theta_{ij} = \frac{1}{2} b_i^T b_j, \tag{1}$$

commonly we can define the likelihood of the observed similarity labels S as follows

$$P(S|B) = \prod_{s_{ij} \in S} p(s_{ij}|\sigma(\theta_{ij})). \tag{2}$$

It is easy to see that we can get different $P(S|B)$ by using different $\sigma(\theta_{ij})$ functions with different prior probability. If we define $\sigma(\theta_{ij}) = \frac{\theta_{ij}}{c}$ and assume that the $P(S_{ij}|\sigma(\theta_{ij}))$ has normal distribution conditional on the binary codes B, then maximizing the likelihood of S in Eq. 2 would be equivalent to optimize the following objective function [11, 12, 16, 15]:

$$\begin{aligned} &\max_{B \in \{-1,1\}^{N \times c}} \sum_{s_{ij} \in S} \log P(s_{ij}|\sigma(\theta_{ij})) \\ = &\min_{B \in \{-1,1\}^{N \times c}} \sum_{s_{ij} \in S} (s_{ij} - \theta_{ij})^2 \end{aligned} \tag{3}$$

The KSH loss function is based on hamming affinity using L_2 loss function, which also be used in MDSH [17]. Alternatively, if we define $P(s_{ij}|\sigma(\theta_{ij}))$ with

$$P(s_{ij}|\sigma(\theta_{ij})) = \frac{1}{1 + e^{-s_{ij}\theta_{ij}}} \tag{4}$$

then maximizing the likelihood of S in Eq. 2 would be equivalent to optimize the following objective function:

$$\begin{aligned} &\max_{B \in \{-1,1\}^{N \times c}} \sum_{s_{ij} \in S} \log P(s_{ij}|\sigma(\theta_{ij})) \\ = &\min_{B \in \{-1,1\}^{N \times c}} \sum_{s_{ij} \in S} \log(1 + e^{-s_{ij}\theta_{ij}}) \end{aligned} \tag{5}$$

It is obvious that these both two object functions are essentially a discrete optimization problem which is hard to optimize. Therefore, some algorithms try to relax $B \in \{-1, 1\}^{N \times c}$ to be a real valued matrix $U \in \mathbb{R}^{N \times c}$, then the $p(S|B)$ can be turned to $P(S|U)$ and the $\theta_{ij} = \frac{1}{2} u_i u_j^T, u \in R^c$. The optimal B is obtained through Single-Bit Quantization (SBQ), where the real values of each dimension are quantized to $\{-1, 1\}$ by thresholding.

However, there is a very important problem: higher-variance dimension carry more information, using the same number of bits for each dimension yields poor performance. In LFH [14], the Gaussian prior is $P(U) = \prod_{i=1}^{c} N(u_{*j}|0, \beta\mathbf{I})$ (u_{*j} is j-th dimension of U), but it cannot be used to solve the problem efficiently. Instead, we extend $P(B|S)$ as the following formula:

$$P(B|S) = P(S|U)P(U|B) \tag{6}$$

In this paper, we propose a new method which can solve the problem more efficiently by balancing the embedding results with real value and quantization error. Please note that we only consider the objective function Eq. 5 due to space limitation. The extension of the proposed model such as Eq. 3 is beyond the scope of this paper.

2.3. Jointly Learning Embedding and Quantization (JLEQ)

Using Iterative Quantization [6] is a direct way for unsupervised hashing methods but only a few works discuss its application in supervised hashing method (CCA-ITQ in [18]). So far as we know, no works discuss the application of using ITQ in hashing methods based on embedding approaches. In the empirical analysis, we found the ITQ cannot improve the performance of two-stepped methods as simple as the CCA-ITQ. Thus based on the property of rotation invariance of embedding techniques, we propose a novel way to jointly learn representation and quantization based on the two-stepped framework [12]. More specifically, we can define the $P(B|U)$ in Eq.6 as the following:

$$\begin{aligned} P(B|U) &= \prod_{i=1}^{n} P(b_i \mid u_i; R) \\ &= \prod_{i=1}^{n}\prod_{j=1}^{c} \frac{1}{\sqrt{2\sigma^2\pi}}\, e^{-\frac{(b_{ij}-R_j u_{ij})^2}{2\sigma^2}} \end{aligned} \tag{7}$$

and then the negative log posterior of $P(B|S)$ can then be derived as:

$$\begin{aligned} L &= -logP(B|S) \\ &= \sum \log(1+e^{-s_{ij}\theta_{ij}}) + \sum_{i=1}^{n} \|sign(u_i) - Ru_i\|_2^2 \end{aligned} \tag{8}$$

the penal term $\|sign(u_i) - Ru_i\|_2^2$ is able to control the space distribution of u_i without changing distance. So for the embedding problem in Eq. 5, the smaller the quantization loss such as $\|sign(u_i) - Ru_i\|^2$, the better the result will be an approximation of binary codes. So we call the Eq.6 using Eq.7 as JLEQ (Jointly Learning Embedding and Quantization). Therefore, we can minimize U and R in an alternating procedure. Please note this procedure need many iterations.

Fixing U, optimize R. This reduces to the Orthogonal Procrustes Problem (OPP):

$$\sum_{i=1}^{n} \|sign(u_i) - Ru_i\|_2^2 \quad s.t. R^T R = I_c \tag{9}$$

where an optimum of OPP can be obtained as same as the method in [18], and then U is updated as RU.

Fixing R, optimize U. Since directly optimizing the whole B and U is impossible when the dataset is too large because it would be very time-consuming, so a simple strategy more feasible is to optimize each row of U at a time with its other rows fixed. We adopt the second order stochastic gradient decent algorithm [19, 14] to optimize each row $u_i \in U$. The gradient vector and the Hessian matrix of the object function $-\log(P(S|U))$ defined in Eq. 5 can be derived as:

$$\frac{\partial L}{\partial u_i^T} = \sum_{i=1}^{n}\sum_{j=1}^{n} -s_{ij}\sigma(-s_{ij}\theta_{ij}) u_j^T \tag{10}$$

$$\frac{\partial^2 L}{\partial u_i^T \partial u_i} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \sigma(-s_{ij}\theta_{ij})(1-\sigma(-s_{ij}\theta_{ij})) u_j^T u_j \tag{11}$$

where $\sigma(-s_{ij}\theta_{ij})(1-\sigma(-s_{ij}) \leq \frac{1}{4}$, the and the thus hessian matrix **H** can be defined as [20]:

$$H = \frac{1}{8}\sum_{j=1}^{n} u_j^T u_j \tag{12}$$

2.4. Out-of-sample Extension

In the above section, we have presented the detail of our method to learn a binary code. However, for new query u not in the training set, we still need to compute the binary code b. Many supervised hashing methods based on the two-step strategy in [12] which first learns binary code and then trains a linear transformation from the original input to the binary code. Because of leveraging this strategy, our method like [14, 16, 15] still need to train a matrix $W \in R^{D\times C}$ that maps x to u in the following ways:

$$u = W^T x \tag{13}$$

Then the binary code can be easily obtained by $sign(u)$. Commonly, the W can be learnt using linear regression with regularization like:

$$\min_{W} \|U - XW\|_F^2 + \lambda\|W\|_F^2 \tag{14}$$

where $U = [u_1, ..., u_n]$ is the embedding result of the original input X. The optimal W can be represented as a closed form:

$$W = (X^T X + \lambda I)^{-1} X^T U \tag{15}$$

where I is an identity matrix.

3. EXPERIMENT

3.1. Datasets

We compare our model with several baselines on three widely used benchmark datasets: CIFAR-10, CIFAR-100 [21], and NUS-WIDE [22]. The CIFAR-10 dataset consists of 60,000 32×32 color images which are categorized into ten classes (6000 images per class). It is a single-label dataset in which each image belongs to one of the ten categories. The CIFAR-100 dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each. The NUS-WIDE dataset has nearly 270,000 images collected from the Internet. It is a multi-label dataset in which each image is annotated with one or multiple class labels from 81 classes. Following [11], There also exist some images without any label, which are not suitable for our evaluation. After removing those images without any label, we get 209,347 images for our experiment. We consider two images to be semantically similar if they share at least one common label. Otherwise, they are semantically dissimilar.

To be independent of deep feature's representation power, this paper has not touched upon deep features. Instead, conventional hand-crafted features such as GIST are used for all hashing methods during evaluation. We represent each image in CIFAR-10 and CIFAR-100 by a 512-dimensional GIST vector. We represent each image in NUS-WIDE by an 1134 dimensional low-level feature vector, including 64-D color histogram, 144-D color correlogram, 73-D edge direction histogram, 128-D wavelet texture, 225-D block-wise color moments and 500-D bag of words based on SIFT descriptions.

3.2. Setting

Experiments in [14, 16] showed that supervised methods can outperform unsupervised methods, thus we only compare our method with several state-of-the-art supervised hashing methods including SPLH [9], KSH [11], TSH [12], FastH [16], LFH [14], SDH [8], COSDISH [15] and CCA-ITQ [6, 18]. All the baselines are implemented using the source code provided by the corresponding authors. For LFH and COSDISH, we use the stochastic learning version with 50 iterations and each iteration samples q columns of the semantic similarity matrix as in [14, 15]. For SPLH, KSH and TSH, using the entire training set for training is impractical due to high time complexity. Same with [11], we randomly sample 2000 points as the training set for CIFAR-10 and 5000 points for NUS-WIDE. TSH [12] can use different loss functions for training. To be fair, the loss function in KSH is used for training TSH and the SVM with RBF-kernel is used for out-of-sample-extension in TSH. For KSH, the number of support vectors is 300 for CIFAR-10, and 1000 for NUS-WIDE. For FastH, boosted decision trees are used for out-of-sample extension rather than linear regression. We use the entire training set for FastH training on CIFAR-10, and randomly sample 100,000 points for FastH training on NUS-WIDE. As most existing hashing methods, the mean average precision (MAP) is used to measure the accuracy of our proposed method and other baselines.

3.3. Result

The mean average precision (MAP) is a widely used metric for evaluating the accuracy of hashing. Table 1, table 2 and table 3 show the MAP of our method and baselines with different code lengths on the three datasets, respectively. In table 1, compared with other methods using relaxed constraints to optimize hash functions, our method outperforms all of them in CIFAR-10. Although the gap between the proposed method and FastH is 64-bits, our method still has a significant advantage in the setting of shorter code lengths. However, the FastH baseline exploits random forest rather than linear regression as the classifier that commonly works better than linear regression in this cases. In Table 2, our method gets the pretty good performance of 8-bits and 16 bits settings in NUS-WIDE and it is even superior to the methods based on discrete optimization. Although it is still not as good as COSDISH in 32, 64-bits. Compared with CIFAR-10, CIFAR-100 has more classes (100 categories) and fewer samples for each category. Thus it is a large challenge for hashing methods. As shown in Table 3, as the code length increased, the performance of the proposed method consistently increases, but other methods just have slightly perturbation in different code lengths. We only select three methods in this dataset because the LFH is the state-of-arts of hashing method using relaxed optimizations.COSDISH works better than FastH.

Table 1. Results of MAP on CIFAR 10

Method	8-bits	16-bits	32-bits	64-bits
Ours (JLEQ)	0.5153	0.5811	0.6147	0.6353
LFH	0.2908	0.4098	0.5446	0.6038
SDH	0.2642	0.3994	0.4145	0.4346
TSH	0.2365	0.3080	0.3455	0.3663
KSH	0.2334	0.2662	0.2923	0.3128
SPLH	0.1588	0.1635	0.1701	0.1730
COSDISH	0.4986	0.5768	0.6191	0.6371
FastH	0.4230	0.5216	0.5970	0.6446

Table 2. Results of MAP on NUS-WIDE

Method	8-bits	16-bits	32-bits	64-bits
Ours (JLEQ)	0.5892	0.6134	0.6023	0.5940
SDH	0.4739	0.4674	0.4908	0.4944
LFH	0.5437	0.5929	0.6025	0.6136
TSH	0.4593	0.4784	0.4857	0.4955
KSH	0.4275	0.4546	0.4645	0.4688
SPLH	0.3769	0.4077	0.4147	0.4071
COSDISH	0.5454	0.5940	0.6218	0.6329
FastH	0.5014	0.5296	0.5541	0.5736

Table 3. Results of MAP on CIFAR 100

Method	8-bits	16-bits	32-bits	64-bits
Ours (JLEQ)	0.6029	0.6055	0.6298	0.6268
LFH	0.6007	0.6012	0.6086	0.6009
COSDISH	0.5638	0.5564	0.5826	0.5818
CCA-ITQ	0.5053	0.5039	0.5076	0.5080

4. CONCLUSION

In this paper, we propose a novel strategy of jointly learning embedding and quantization for supervised hashing. It only needs relaxed continuous optimization so that it is easy to be solved without sophisticated optimization methods. Hence, it is also easy to handle large-scale problems. Experimental results on three datasets with semantic labels show that our methods achieve competitive performance compared with the state-of-the-art.

5. REFERENCES

[1] Alexandr Andoni and Piotr Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*. IEEE, 2006, pp. 459–468.

[2] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al., "Similarity search in high dimensions via hashing," in *VLDB*, 1999, vol. 99, pp. 518–529.

[3] Maxim Raginsky and Svetlana Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Advances in neural information processing systems*, 2009, pp. 1509–1517.

[4] Ruslan Salakhutdinov and Geoffrey Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[5] Yair Weiss, Antonio Torralba, and Rob Fergus, "Spectral hashing," in *Advances in neural information processing systems*, 2008, pp. 1753–1760.

[6] Yunchao Gong and Svetlana Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 817–824.

[7] Weihao Kong and Wu-Jun Li, "Isotropic hashing," in *Advances in Neural Information Processing Systems*, 2012, pp. 1646–1654.

[8] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen, "Supervised discrete hashing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 37–45.

[9] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang, "Sequential projection learning for hashing with compact codes," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 1127–1134.

[10] Mohammad Norouzi and David M Blei, "Minimal loss hashing for compact binary codes," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 353–360.

[11] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang, "Supervised hashing with kernels," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2074–2081.

[12] Guosheng Lin, Chunhua Shen, David Suter, and Anton Hengel, "A general two-step approach to learning-based hashing," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2552–2559.

[13] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen, "Deep supervised hashing for fast image retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2064–2072.

[14] Peichao Zhang, Wei Zhang, Wu-Jun Li, and Minyi Guo, "Supervised hashing with latent factor models," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 2014, pp. 173–182.

[15] Wang-Cheng Kang, Wu-Jun Li, and Zhi-Hua Zhou, "Column sampling based discrete supervised hashing," in *AAAI*, 2016.

[16] Guosheng Lin, Chunhua Shen, Qinfeng Shi, Anton Hengel, and David Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1963–1970.

[17] Yair Weiss, Rob Fergus, and Antonio Torralba, "Multi-dimensional spectral hashing," in *European Conference on Computer Vision*. Springer, 2012, pp. 340–353.

[18] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 12, pp. 2916–2929, 2013.

[19] Léon Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. 2010, pp. 177–186, Springer.

[20] Kenneth Lange, David R Hunter, and Ilsoon Yang, "Optimization transfer using surrogate objective functions," *Journal of computational and graphical statistics*, vol. 9, no. 1, pp. 1–20, 2000.

[21] Alex Krizhevsky and Geoffrey Hinton, "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[22] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng, "Nus-wide: a real-world web image database from national university of singapore," in *Proceedings of the ACM international conference on image and video retrieval*. ACM, 2009, p. 48.