Error-correcting Output Hashing in Fast Similarity Search

Zhou Yu Deng Cai Xiaofei He

The State Key Lab of CAD&CG, College of Computer Science
Zhejiang University, China
echoyuzhou@gmail.com, {dengcai,xiaofeihe}@cad.zju.edu.cn

ABSTRACT

Fast similarity search is one of the key techniques in many large scale learning and data mining applications. Recently, hashing-based methods, which create compact and efficient codes that preserve data distribution, have received considerable attention due to their promising theoretical and empirical results. An ideal hashing method 1) can naturally have out-of-sample extension; 2) has very low computational complexity; and 3) has significant improvement over linear search in the original space in terms of accuracy. However, most existing hashing methods failed to satisfy all the above three requirements. In this paper, we propose a new method called Error-correcting Output Hashing (ECOH) which meets all the above three requirements. ECOH first groups all the samples into clusters using a conventional clustering algorithm. Each cluster is assigned an Error-Correcting Output Code (ECOC) and the linear mappings from the sample vectors to the ECOC are then learned using linear regression models. In this way, ECOH learns both the binary code for each sample and the function which links the input vector and the output code. Experimental results on real world data sets demonstrate the effectiveness and efficiency of the proposed approach.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Experimentation, Performance

Keywords

Similarity Search, Semantic Hashing, Error-correcting output code

1. INTRODUCTION

ICIMCS'10, December 30-31, 2010, Harbin, China

Nearest neighbor search is one of fundamental problems in machine learning. It plays an important role in various areas, such as data compression, pattern recognition and especially similarity search. Similarity search is defined as finding the similar items of a given query in a certain data set. It has various applications in real world, for example: duplicate image detection [2], scene completion [5], plagiarism analysis [17] and so on.

In nearest neighbor search, typically, an item is represented as a point in R^d and a distance metric is used to measure the similarity (or, distance) between objects. For a database containing N objects, if the naive linear scan approach is used, the complexity for each query is $O(N)$. To speed up this process, the approximate nearest neighbor search was introduced [3][14][9]. Instead of finding the exact nearest neighbor of a query, approximate nearest neighbor search returns an approximate nearest neighbor within a certain error tolerance. Thus, this approach can have significant computational saving while the approximate neighbor is almost as good as the exact one. For example, tree based approximate nearest neighbor methods were proposed to claim sublinear $o(N)$ time or logarithmatic $O(\log(N))$ query time [7].

In recent years, hashing based methods [12][20] received considerable attention. These methods design compact binary codes for the data so that semantically similar data points are mapped to similar codes (within a short Hamming distance). Since the Hamming distance between two binary codes can be computed efficiently by using bit XOR operation and counting the number of set bits [8][18], performing similarity search over such binary codes can be extremely fast. Given a query, instead of computing its similarity to all the data points in the data base, a small number of the most similar data points are usually needed. In such situations, we can simply return all the data points that are hashed into a tight Hamming ball centered around the binary code of the query point, which can further reduce the computational cost.

An ideal hashing method 1) can naturally have out-of-sample extension; 2) has very low computational complexity; and 3) has significant improvement over linear search in the original space in terms of accuracy. However, most existing hashing methods can not satisfy all the above three requirements. In this paper, we propose a novel hashing method called Error-correcting Output Hashing (ECOH) which meets all the above three requirements. ECOH first groups all the samples into clusters using a conventional

clustering algorithm. Each cluster is assigned an Error-Correcting Output Code (ECOC) and the linear mappings from the sample vectors to the ECOC are then learned using linear regression models. In this way, ECOH learns both the binary code for each sample and the function which links the input vector and the output code. One of the advantages of our method lies in the error correcting ability of the error-correcting output code that we use to represent each data point. It can not only correct error bits in the test query but also tolerant certain noise in the training and testing phases. The usage of error-correcting output code in machine learning community can be traced to early 1960s [11]. It is extensively used in multi-class learning problems [4]. To the best of our knowledge, this work is the first one to use the error-correcting output code in learning semantic hashing functions. In the experiments we carried out in this paper, ECOH outperforms all the state-of-the-art methods on fast similarity search in large scale data sets.

2. RELATED WORKS

Many hashing based methods are proposed to handle the similarity search problem. If we loose the compactness principle of the code in semantic hashing [13], then it reduces to the basic idea of the locality sensitive hashing (LSH) [1]. In LSH [1], every bit of the code is calculated by a random linear projection followed by a random threshold. The Hamming distance between codes will approach the Euclidean distance between the original data vectors. However, this method usually leads to very ineffective code and low accuracy. Shift-invariant Kernel-based Hashing(SIKH) [10] is an extension of LSH with a shifted cosine function when doing hashing. Parameter-Sensitive Hashing (PSH) [15] is also an extension of LSH that uses supervised learning. It uses hash functions sensitive to the similarity in the parameter space, and then retrieves approximate nearest neighbors. Stacked Restricted Boltzman Machine (RBM) [12] is another supervised learning method which seeks to accelerate the retrieval speed by producing effective compact binary code. However, the above two supervised learning methods both fail to achieve short response time due to large computation complexity. Spectral hashing (SpH) [19] designs compact code that reveals the intrinsic geometric structure of the data set. It outperforms all the methods above. Self-taught hashing (STH) [20] is a revised version of SpH. It uses a supervised learning method to obtain the hashing function in order to solve the out-of-sample extension problem, instead of posing strick uniform distribution supposition like SpH.

3. ERROR-CORRECTING OUTPUT HASHING

Following the recently proposed self-taught hashing (STH) [20], our Error-correcting Output Hashing (ECOH) also adopts a two-step approach. In the first step, unsupervised learning is used to learn the binary codes for all the data points in the training data. In the second step, supervised learning is used to learn the mapping from the original sample vectors to the binary codes. Different from STH, where the spectral embedding method is utilized to learn the binary codes which is computationally expensive, our ECOH approach designs the Error-correcting output codes for data clusters (by using a conventional clustering algorithm) as the binary codes. The error correcting ability of the code makes it possible for hashing functions to tolerant noise.

Table 1: A 15-bit error-correcting output code for 10 classes

Class	Code Word														
0	1	1	0	0	0	0	1	0	1	0	0	1	1	0	1
1	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0
2	1	0	0	1	0	0	0	1	1	1	1	0	1	0	1
3	0	0	1	1	0	1	1	1	0	0	0	0	1	0	1
4	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1
5	0	1	0	0	1	1	0	1	1	1	0	0	0	0	1
6	1	0	1	1	1	0	0	0	0	1	0	1	0	0	1
7	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1
8	1	1	0	1	0	1	1	0	0	1	0	0	0	1	1
9	0	1	1	1	0	0	0	0	1	0	1	0	0	1	1

We begin with a short description on designing the Error-correcting output codes.

3.1 Error-correcting output codes design

Table 1 shows 15-bit error-correcting output codes for 10 classes. The length of a code is the number of columns and the row number is the number of classes. Effective error-correcting codes for multiple classes should maximize both the row and column differences of the code matrix in order to reduce the correlation among the projection matrix in the later step. This restriction separates the code in binary space, which leads to the large entropy in codes as well.

There are four popular error-correcting output codes designing methods [4]: (a) an exhaustive technique, (b) a method that selects columns from an exhaustive code, (c) a method based on a randomized hill-climbing algorithm, and (d) BCH codes. The choice of which method to use is based on the number of classes k. Finding a single method suitable for all values of k still remains an open problem. All these methods are described in detail in [4] and some well generated error-correcting output codes are also available online.[1]

3.2 Clusters generation

To discover the geometrical structure of the data, we need to perform a conventional clustering step. Among various clustering algorithms, spectral clustering (Normalized Cut)[16] is one of the most promising methods. In our work, we simply use spectral clustering.

Given the data set $\mathbf{x}_1, \cdots, \mathbf{x}_n$, we construct the $n \times n$ local similarity matrix W as

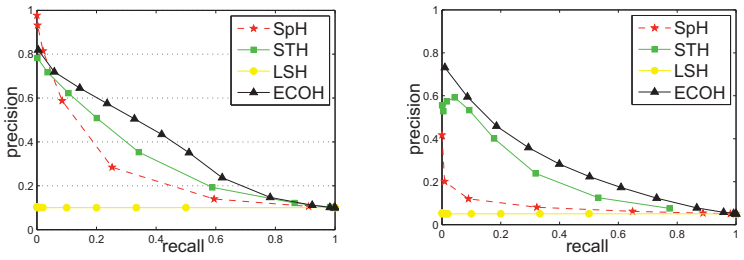
$$W_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_i \in N_k(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in N_k(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases}$$

where $N_k(\mathbf{x}_i)$ denotes the k-nearest neighbors of $\mathbf{x}_i$. Define the $n \times n$ matrix D whose entries are given by the sums of the columns (or the rows) of W. Spectral clustering needs to solve the following eigen-problem:

$$D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \mathbf{z} = \lambda \mathbf{z}$$

The eigenvectors $\mathbf{z}$ corresponding to the largest eigenvalues are then be put together to form the low dimensional representation of the data. Clusters are then obtained by applying k-means on the low dimensional representation.

[1] http://www.princeton.edu/∼kung/ele571/571−MatLab/571svm/



(a) MNIST (b) 20Newsgroups

Figure 1: The precision recall curve of MNIST and 20NG data set for retrieving the same topic items with code length of 64

In real-world situations, we do not know how many clusters we should group the data into so that we can obtain the best results. It seems that more clusters will lead to better performance since the local geometric structure can be better preserved. However, a large number of clusters also leads to large computational cost. In our work, we treat the number of clusters as a parameter that may differ in different situations. We will further illustrate this in Section 4.

3.3 Supervised learning of hashing functions

The length of the error-correcting output codes is another parameter. Basically, the longer the code, the more error bits it corrects. However, the longer the code, the more space it occupy take in memory and the longer process time it will take when computing the hamming distance. Detailed analysis on the parameter selection will be presented in Section 4. After preparing the inputs (the original sample vectors) and responses (the error-correcting output codes), we perform a ridge regression to obtain the projection matrix. The projection matrix is considered as our hashing function. It is also possible to use support vector machines (SVM) [6] to learn the mapping (hashing) functions. Using this hashing function, we can encode all the items by simple multiplications and thresholds on 0.

3.4 Out-of-sample extension

The problem of out-of-sample extension is of great importance in real-world application. Our work provides a natural way to solve this problem. When a new query (a data point) comes, we simply use our hashing function to encode the query (We multiply the data point with the projecting matrix we obtained in the supervised learning step and threshold the result with 0.) After the code is computed, we retrieve the item with the smallest hamming difference with the query. Our method avoids the uniform distribution assumption imposed in spectral hashing [19] and provides a more reasonable and more efficient solution.

4. EXPERIMENTS

We compare our ECOH approach with Locality Sensitive Hashing (LSH), Spectral Hashing (SpH) and Self Taught Hashing (STH) on two real world data sets. We set the number of nearest neighbors as 5 throughout the experiments.

4.1 Data

The first data set is the MNIST which is available on line[2]. It is a database of handwritten digits which has a training set of 60,000 examples, and a testing set of 10,000 examples. Digits from 0 to 9 are all centered in a 28x28 image. The gray-scale intensity values of images are directly used as features which can be transformed into a 784-dimension vector.

The second data set is 20 Newsgroups (20NG) which is a collection of newsgroup documents and is also available online[3]. We use the popular "bydate" version which contains 18846 documents, evenly distributed across 20 categories. There are 11314 (around 60%) documents in the training set and 7532 (around 40%) documents in the test set. We further preprocessed the data by removing common stopwords, stemming, and then only considering the 2000 most frequent words in the training data set. As a result, each document is represented as a 2000-dimensional vector.

4.2 Evaluation

The standard retrieval performance measures, precision and recall, are used to evaluate the results.

$$precision = \frac{the\,number\,of\,retieved\,relevant\,item}{the\,number\,of\,all\,retrieved\,item}$$

$$recall = \frac{the\,number\,of\,retrieved\,relevant\,item}{the\,number\,of\,all\,relevant\,item}$$

In our experiments we define that an image or a document is relevant to the query if and only if the retrieved item has the same label as the query.

4.3 Results

Figure 1 shows the average precision-recall curves of different methods on the MNIST data set and 20NG data set. We set the code length as 64 for all methods. We can clearly see that ECOH outperforms its competitors on both data sets. The error correcting ability of the error-correcting output code contributes greatly to the good performance in term of precision and recall.

There are two parameters in our ECOH approach. The first one is the length of the codes and the other is the number of clusters that the original data should be grouped into. It is straightforward to see that the longer the error-correcting output codes, the more error bits it can correct. However, due to the limitation of hardware in computation speed and memory storage, only limited code length is possible to implement. In order to find the balance of the two,

[2]http://yann.lecun.com/exdb/mnist/

[3]http://people.csail.mit.edu/jrennie/20Newsgroups/

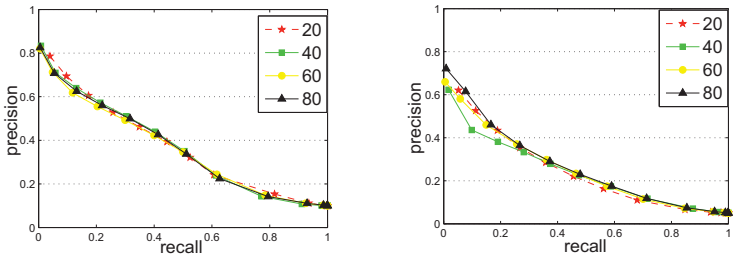


Figure 2: The precision recall curve of ECOH approach on MNIST and 20NG for retrieving the same topic items with different cluster numbers

we choose 64 as the code length in our experiment. Another parameter is more difficult to choose. In Figure 2 we demonstrate the precision recall curve of ECOH on the two data sets with different number of clusters. We can see that the performance of ECOH is very stable with respect to the number of clusters. This demonstrates the robustness of ECOH which makes it more applicable in real applications.

5. CONCLUSIONS

We propose a new type of semantic hashing which is called Error-correcting Output Hashing (ECOH). ECOH first groups all the samples into clusters using a conventional clustering algorithm. Each cluster is assigned an Error-Correcting Output Code (ECOC) and the linear mappings from the sample vectors to the ECOC are then learned using linear regression models. In this way, ECOH learns both the binary code for each sample and the function which links the input vector and the output code. Experimental results on real world data sets demonstrate the effectiveness and efficiency of the proposed approach.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China under Grants 60905001 and 90920303, National Basic Research Program of China (973 Program) under Grant 2011CB302206 and Fundamental Research Funds for the Central Universities.

6. REFERENCES

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *IEEE Symposium on Foundations of Computer Science*, 2006.

[2] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *CIVR '07: Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 549–556, New York, NY, USA, 2007. ACM.

[3] C.Silpa-Anan and R.Hartley. Optimised kd-trees for fast image descriptor matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[4] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. In *Journal of Artificial Intelligence Research*, 1995.

[5] J. Hays and A. A. Efros. Scene completion using millions of photographs. *Commun. ACM*, 51(10):87–94, 2008.

[6] T. Joachims. Learning to classify text using support vector machines. In *Kluwer*, 2002.

[7] J.Philbin, O.Chum, M. Isard, and A. J.Sivic. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 2007.

[8] D. Knuth. *The Art of Computer Programming*. Addison-Wesley, 3rd edition edition, 1997.

[9] M.MujaandD and G.Lowe. Fast approximate nearest neighbors with automatic algorithm confguration. In *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2009.

[10] M.Raginsky and S.Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *The Neural Information Processing Systems*, 2009.

[11] R.O.Duda, J.W.Machanik, and R.C.Singleton. Function modeling experiments. Technical report, Stanford Research Institute, 1963.

[12] R. Salakhutdinov and G. Hinton. Semantic hashing. In *International Journal of Approximate Reasoning*, 2009.

[13] R. R. Salakhutdinov and G. E. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *AISTATS*, 1993.

[14] S.Arya, D.M.Mount, N.S.Netanyahu, R.Silverman, and A.Y.Wu. An optimal algorithm for approximate nearest neighbor searching. In *Journal of ACM*, 1998.

[15] G. Shakhnarovich, P. A. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV)*, 2003.

[16] J. Shi and J.Malik. Normalized cuts and image segmentation. In *IEEE Transactions on pattern analysis and machine intelligence*, 2000.

[17] B. Stein, S. M. zu Eissen, and M. Potthast. Strategies for retrieving plagiarized documents. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2007.

[18] P. Wegner. A technique for counting ones in a binary computer. In *Communications of the ACM (CACM)*, 1960.

[19] Y. Weiss, A. B. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[20] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval(SIGIR)*, 2010.