Discrete Binary Coding based Label Distribution Learning

Ke Wang and **Xin Geng**[*]
MOE Key Laboratory of Computer Network and Information Integration, China
School of Computer Science and Engineering, Southeast University, Nanjing 210096, China
{k.wang, xgeng}@seu.edu.cn

Abstract

Label Distribution Learning (LDL) is a general learning paradigm in machine learning, which includes both single-label learning (SLL) and multi-label learning (MLL) as its special cases. Recently, many LDL algorithms have been proposed to handle different application tasks such as facial age estimation, head pose estimation and visual sentiment distributions prediction. However, the training time complexity of most existing LDL algorithms is too high, which makes them unapplicable to large-scale LDL. In this paper, we propose a novel LDL method to address this issue, termed *Discrete Binary Coding based Label Distribution Learning* (DBC-LDL). Specifically, we design an efficiently discrete coding framework to learn binary codes for instances. Furthermore, both the pair-wise semantic similarities and the original label distributions are integrated into this framework to learn highly discriminative binary codes. In addition, a fast approximate nearest neighbor (ANN) search strategy is utilized to predict label distributions for testing instances. Experimental results on five real-world datasets demonstrate its superior performance over several state-of-the-art LDL methods with the lower time cost.

1 Introduction

Learning with label ambiguity has attracted considerable attention in recent years, because it lays the foundation for many important fields, including machine learning, computer vision and data mining. Single-label learning (SLL) and multi-label learning (MLL) are two prevailing learning paradigms for solving the problem of label ambiguity. However, SLL and MLL only can model the ambiguity of "what describes the instance", but fail to model the more general ambiguity of "how to describe the instance" [Xing *et al.*, 2016]. To solve this problem, Geng (2016) proposed label distribution learning (LDL), which assumes that an instance is labeled by a label distribution over all labels. Each element of the label distribution is named the description degree. For example, $d_{\boldsymbol{x}}^{y}$ is the description degree of y to the instance $\boldsymbol{x}$. Without loss of generality, LDL assumes that $d_{\boldsymbol{x}}^{y} \in [0, 1]$ and $\sum_y d_{\boldsymbol{x}}^{y} = 1$. Different from SLL and MLL, LDL aims at learning a group of projection functions from an instance to its label distribution.

Many LDL methods have already been proposed. Some representative works include IIS-LDL [Geng *et al.*, 2013], conditional probability neural network (CPNN) [Geng *et al.*, 2013], label distribution support vector regressor (LDSVR) [Geng and Hou, 2015], BFGS-LDL [Geng, 2016], PT-Bayes [Geng, 2016], PT-SVM [Geng, 2016], AA-BP [Geng, 2016], LDLogitBoost [Xing *et al.*, 2016], AOSO-LDLogitBoost [Xing *et al.*, 2016], deep label distribution learning (DLDL) [Gao *et al.*, 2017], structural label distribution learning (SLDL)[Ren and Geng, 2017] and label distribution forests (LDLFs) [Shen *et al.*, 2017]. However, the training time complexity of most methods is high. How to design a LDL algorithm to achieve good accuracy performance with low training time complexity is still a challenging problem.

To address the above-mentioned problem, Geng (2016) put forward a LDL approach named AA-kNN, which is an adaptation algorithm of k-Nearest Neighbor (k-NN). Although AA-kNN does not need a training process, its testing process (linear search) usually cost much time. Wang and Geng (2018) proposed a scalable LDL algorithm called BC-LDL, which uses the binary coding techniques to deal with the large-scale LDL problem. Compared with AA-kNN, the testing time cost of BC-LDL can be greatly reduced, because the similarity of two instances can be efficiently measured via XOR operations. However, BC-LDL suffers two drawbacks. Firstly, the discrete constraints are discarded to simplify the optimization, and the binary codes of training instances are approximately obtained by quantization. The errors caused by quantization may lead to low-quality binary codes. Secondly, BC-LDL adopts an iterative optimization strategy in the training phase, and the iteration number is equal to the code length. BC-LDL requires relatively long binary codes (i.e., over 64 bits) to achieve acceptable accuracy. Therefore, the training phase of BC-LDL is still time-consuming.

Recently, the discrete binary coding techniques have attracted increasing interest and many discrete optimization based hashing methods have been proposed. The desirable binary codes can be generated by these discrete methods, because the binary codes of training instances are directly

[*]Corresponding author.

learned without any quantization loss during the optimization procedure. Since supervised discrete hashing methods take the useful label information into consideration when learning binary codes and hash functions, they usually yield higher accuracy than unsupervised ones. Representative supervised discrete hashing methods include supervised discrete hashing (SDH) [Shen *et al.*, 2015], column sampling based discrete supervised hashing (COSDISH) [Kang *et al.*, 2016], asymmetric discrete graph hashing (ADGH) [Shi *et al.*, 2017], fast supervised discrete hashing (FSDH) [Gui *et al.*, 2018] , fast scalable supervised hashing (FSSH) [Luo *et al.*, 2018a], scalable supervised discrete hashing (SSDH) [Luo *et al.*, 2018b] etc. However, most of them may suffer some drawbacks when using the pair-wise similarity matrix (denoted as $\boldsymbol{S}^{n\times n}$) in the training phase. For example, COSDISH and ADGH only use some points of $\boldsymbol{S}$ for training. FSSH and SSDH embed $\boldsymbol{S}$ into a small-size matrix in the pre-processing phase, but the complexity of this pre-processing phase is $O(n^2)$.

The motivations of this paper are three-fold: (1) addressing the issue of quantization loss in BC-LDL; (2) designing a linear discrete hashing algorithm with all the points of $\boldsymbol{S}$ utilized for instances labeled by a label distribution; (3) proposing an efficient LDL approach based on this discrete algorithm.

In this paper, we take an initial attempt to apply the discrete hashing techniques to large-scale LDL and further present a scalable LDL method, named *Discrete Binary Coding based Label Distribution Learning* (DBC-LDL). Different from the previous discrete hashing methods, the binary coding part of DBC-LDL is specially designed for instances labeled by a label distribution. The main contributions of this work are briefly summarized as follows:

- A novel supervised discrete binary coding method is proposed for instances annotated by a label distribution. In our algorithm, we consider both the pair-wise similarities of different training instances and the original label distributions, leading to high-quality binary codes. Moreover, the similarity matrix $\boldsymbol{S}$ is avoided directly utilizing during the optimization procedure. The time and storage cost of our method can be reduced significantly.
- An iterative optimization strategy is proposed to learn the binary codes for training instances and the binary coding functions. The global non-convex optimization problem can be decomposed into several tractable subproblems, and each variable has a closed-form solution.
- Integrating discrete binary coding and ANN search into a joint framework, a scalable LDL approach is proposed to deal with the large-scale LDL. Compared with BC-LDL, the desirable binary codes of training instances can be directly learned in the training phase, thus the quantization errors are avoided in DBC-LDL.
- Extensive experiments are conducted on five benchmark datasets to evaluate the proposed DBC-LDL. Experimental results verify the effectiveness of our approach.

The remainder of this paper is organized as follows. In Section 2, the details of the DBC-LDL algorithm are presented. In Section 3, we report the experimental results on five public datasets, followed by the conclusion in Section 4.

Symbols	Descriptions
$\boldsymbol{X}, \overline{\boldsymbol{X}}$	the feature matrix, $\overline{\boldsymbol{X}} = [\boldsymbol{X}; \boldsymbol{1}_n]$
Y	the complete set of labels
$\boldsymbol{D}$	the label distributions matrix
$\overline{\boldsymbol{D}}$	the adjusted label distributions matrix
$\boldsymbol{P}, \boldsymbol{W}$	the projection matrices
$\boldsymbol{e}, \overline{\boldsymbol{P}}$	the bias vector, $\overline{\boldsymbol{P}} = [\boldsymbol{P}, \boldsymbol{e}]$
$\boldsymbol{S}$	the $n \times n$ pair-wise similarity matrix
$\boldsymbol{B}$	the the binary code matrix
l, T	the code length, the maxiter number

Table 1: Important notations used in this paper.

2 The Proposed Method

2.1 Notations and Problem Definition

Suppose that $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n] \in \mathbb{R}^{m\times n}$ is the feature space, and $Y = \{y_1, y_2, \cdots, y_c\}$ is the complete set of labels. c denotes the number of the labels. Here, we use $d_{\boldsymbol{x}}^{y}$ to represent the description degree of the label $y \in Y$ to the instance $\boldsymbol{x} \in \boldsymbol{X}$. $Z = \{(\boldsymbol{x}_1, \boldsymbol{d}_1), (\boldsymbol{x}_2, \boldsymbol{d}_2,), \cdots, (\boldsymbol{x}_n, \boldsymbol{d}_n)\}$ denotes the training set, where $\boldsymbol{x}_i \in \mathbb{R}^{m\times 1}$ is a training instance, and $\boldsymbol{d}_i = [d_{\boldsymbol{x}_i}^{y_1}; d_{\boldsymbol{x}_i}^{y_2}; \cdots; d_{\boldsymbol{x}_i}^{y_c}]$ is the corresponding label distribution of $\boldsymbol{x}_i$. n is the number of training instances. Without loss of generality, we consider that the instances are zero-centered, i.e., $\sum_{i=1}^{n} \boldsymbol{x}_i = 0$. Table 1 shows the important notations used in this paper.

In this paper, the target of DBC-LDL is to learn the binary codes for training instances and binary coding functions for future instances. Then, the same ANN search strategy as that in [Wang and Geng, 2018] is adopted to generate label distributions for future instances. The binary coding function $h(\boldsymbol{x})$ is defined as follows:

$$h(\boldsymbol{x}) = sign(\boldsymbol{P}\boldsymbol{x} + \boldsymbol{e}), \tag{1}$$

where $\boldsymbol{P} \in \mathbb{R}^{l\times m}$ is the projection matrix, and $\boldsymbol{e} \in \mathbb{R}^{l\times 1}$ is the bias vector. Here, l represents the length of binary codes.

2.2 Pair-wise Similarity Matrix

For most existing supervised discrete hashing approaches [Kang *et al.*, 2016; Shi *et al.*, 2017; Li *et al.*, 2017; Luo *et al.*, 2018a; 2018b; Jiang *et al.*, 2018], the semantic similarity of two instances can be obtained via modeling the label consistency between them. However, it is not appropriate to apply this idea to LDL, because an instance is labeled by a distribution consisting of some real numbers in LDL, rather than annotated by a binary label vector.

In this paper, we employ the adjusted cosine similarity (ACS) metric [Sarwar *et al.*, 2001; Wang and Geng, 2018] to build the similarity matrix $\boldsymbol{S} \in [-1, 1]^{n\times n}$. Suppose there are two instances $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ from the training set, the similarity between them is defined as follows:

$$\begin{aligned} s_{ij} &= \frac{(\boldsymbol{d}_i - \epsilon)^T \cdot (\boldsymbol{d}_j - \epsilon)}{||\boldsymbol{d}_i - \epsilon||_2 ||\boldsymbol{d}_j - \epsilon||_2} \\ &= \overline{\boldsymbol{D}}_{i*}^T \overline{\boldsymbol{D}}_{j*}, \end{aligned} \tag{2}$$

where "$\cdot$" represents the inner product between two vectors, $||\cdot||_2$ represents the L_2 norm of a vector, ϵ is the mean of all the label distributions, and $\overline{\boldsymbol{D}}_{i*} = \frac{\boldsymbol{d}_i - \epsilon}{||\boldsymbol{d}_i - \epsilon||_2}$.

Thus, the pair-wise similarity matrix can be explicitly calculated by $\boldsymbol{S} = \overline{\boldsymbol{D}}^T \overline{\boldsymbol{D}}$. We can find that the complexity of both storage and computational time to construct this similarity matrix is $O(n^2)$. Fortunately, the high storage and time cost can be avoided in DBC-LDL, because the similarity matrix will not be directly used in the optimization phase.

2.3 Graph Term

Graph based supervised hashing methods aim at transforming the high-dimensional original features into compact binary codes with semantic similarities between different instances best maintained. Similar to some existing graph based hashing algorithms [Weiss *et al.*, 2009; Liu *et al.*, 2011; 2014], the graph term is defined as follows:

$$\min_{\boldsymbol{B}} \sum_{i,j=1}^{n} s_{ij} ||\boldsymbol{b}_i - \boldsymbol{b}_j||_2^2, \tag{3}$$

where $\boldsymbol{B} \in \{-1, 1\}^{l \times n}$ is the binary code matrix.

Actually, the element of the similarity matrix $\boldsymbol{S}$ is a constant, and $\boldsymbol{b}_i^T \boldsymbol{b}_i = l (\forall i)$. Eq. (3) can be rewritten as follows:

$$\begin{aligned} &\min_{\boldsymbol{B}} \sum_{i,j=1}^{n} -2 s_{ij} \boldsymbol{b}_i^T \boldsymbol{b}_j + const \\ &= \min_{\boldsymbol{B}} -2tr(\boldsymbol{B}\boldsymbol{S}\boldsymbol{B}^T). \end{aligned} \tag{4}$$

where $tr(\cdot)$ denotes the trace operator.

The above graph term is defined on the directly learned binary codes $\boldsymbol{B}$. However, the binary codes of testing instances are generated by using binary coding functions (i.e., quantization). For training instances, we expect $\boldsymbol{B}$ and the binary codes generated by quantization are as similar as possible. This is because it can help testing instances to reduce the latent quantization loss. Thus, we put the constraints of semantic similarity preserving on $\boldsymbol{B}$ and $(\boldsymbol{P}\boldsymbol{X} + \boldsymbol{E})$, and further reformulate the graph term by adding a new constraint as:

$$\begin{aligned} &\min_{\boldsymbol{B},\boldsymbol{P},\boldsymbol{E}} -2tr((\boldsymbol{P}\boldsymbol{X} + \boldsymbol{E})\boldsymbol{S}\boldsymbol{B}^T) + \alpha ||\boldsymbol{B} - (\boldsymbol{P}\boldsymbol{X} + \boldsymbol{E})||_F^2 \\ &= \min_{\boldsymbol{B},\overline{\boldsymbol{P}}} -2tr(\overline{\boldsymbol{P}}\,\overline{\boldsymbol{X}}\,\boldsymbol{S}\boldsymbol{B}^T) + \alpha ||\boldsymbol{B} - \overline{\boldsymbol{P}}\,\overline{\boldsymbol{X}}\,||_F^2, \end{aligned} \tag{5}$$

where $\boldsymbol{E} = [\boldsymbol{e}, \boldsymbol{e}, \cdots, \boldsymbol{e}] \in \mathbb{R}^{l \times n}$, $\overline{\boldsymbol{X}} = [\boldsymbol{X}; \boldsymbol{1}_n]$ and $\overline{\boldsymbol{P}} = [\boldsymbol{P}, \boldsymbol{e}]$. α is a tradeoff parameter.

2.4 Label Distribution Embedding Term

To further improve the quality of the learned binary codes, the original label distributions are taken into consideration when constructing the objective function. By assuming an explicit projection from label distributions to binary codes, we can define the following optimization problem:

$$\min_{\boldsymbol{W}} ||\boldsymbol{B} - \boldsymbol{W}\boldsymbol{D}||_F^2, \tag{6}$$

where $\boldsymbol{D} \in \mathbb{R}^{c \times n}$ is the label distributions matrix, and $\boldsymbol{W} \in \mathbb{R}^{l \times c}$ is the corresponding mapping matrix.

2.5 Overall Objective Function

The overall objective function, consisting of the graph term in (5), the label distribution embedding term in (6) and an additional regularization term, is written as follows:

$$\begin{aligned} &\min_{\boldsymbol{B},\overline{\boldsymbol{P}},\boldsymbol{W}} \Theta(\boldsymbol{B}, \overline{\boldsymbol{P}}, \boldsymbol{W}) \\ &= -2tr(\overline{\boldsymbol{P}}\,\overline{\boldsymbol{X}}\,\boldsymbol{S}\boldsymbol{B}^T) + \alpha ||\boldsymbol{B} - \overline{\boldsymbol{P}}\,\overline{\boldsymbol{X}}\,||_F^2 \\ &+ \beta ||\boldsymbol{B} - \boldsymbol{W}\boldsymbol{D}||_F^2 + \gamma R(\overline{\boldsymbol{P}}, \boldsymbol{W}), \end{aligned} \tag{7}$$

where the regularization term $R(\cdot) = ||\cdot||_F^2$ is introduced to resist overfitting. α, β, γ are balance parameters.

2.6 Optimization

The optimization problem stated in (7) is hard to be directly solved due to its non-convexity with three matrix variables $\boldsymbol{B}$, $\overline{\boldsymbol{P}}$, $\boldsymbol{W}$. In this paper, we use an iterative strategy to solve this optimization problem, and each variable can obtain a closed-form solution according to this strategy. The detailed optimization procedure is presented below.

Step 1: Fix $\boldsymbol{B}$, $\boldsymbol{W}$, let $\frac{\partial \Theta}{\partial \overline{\boldsymbol{P}}} = 0$, then obtain:

$$\overline{\boldsymbol{P}} = \boldsymbol{B}(\overline{\boldsymbol{X}} + \frac{1}{\alpha}\overline{\boldsymbol{X}}\boldsymbol{S})(\overline{\boldsymbol{X}}\,\overline{\boldsymbol{X}}^T + \frac{\gamma}{\alpha}\boldsymbol{I}_m)^{-1}, \tag{8}$$

where $\boldsymbol{I}_m$ is a $m \times m$ identity matrix.

Step 2: Fix $\boldsymbol{B}$, $\overline{\boldsymbol{P}}$, let $\frac{\partial \Theta}{\partial \boldsymbol{W}} = 0$, then obtain:

$$\boldsymbol{W} = \boldsymbol{B}\boldsymbol{D}^T(\boldsymbol{D}\boldsymbol{D}^T + \frac{\gamma}{\beta}\boldsymbol{I}_c)^{-1}. \tag{9}$$

Step 3: Fix $\overline{\boldsymbol{P}}$, $\boldsymbol{W}$, then obtain the following problem:

$$\begin{aligned} &\min_{\boldsymbol{B}} -2tr(\overline{\boldsymbol{P}}\,\overline{\boldsymbol{X}}\,\boldsymbol{S}\boldsymbol{B}^T) + \alpha tr((\boldsymbol{B} - \overline{\boldsymbol{P}}\,\overline{\boldsymbol{X}})(\boldsymbol{B} - \overline{\boldsymbol{P}}\,\overline{\boldsymbol{X}})^T) \\ &\quad + \beta tr((\boldsymbol{B} - \boldsymbol{W}\boldsymbol{D})(\boldsymbol{B} - \boldsymbol{W}\boldsymbol{D})^T). \end{aligned} \tag{10}$$

It is easy to see that $tr(\boldsymbol{B}\boldsymbol{B}^T)$, $tr((\overline{\boldsymbol{P}}\,\overline{\boldsymbol{X}})(\overline{\boldsymbol{P}}\,\overline{\boldsymbol{X}})^T)$ and $tr((\boldsymbol{W}\boldsymbol{D})(\boldsymbol{W}\boldsymbol{D})^T)$ are constants in (10). Thus, the above optimization problem can be reformulated as follows:

$$\min_{\boldsymbol{B}} -2tr((\overline{\boldsymbol{P}}\,\overline{\boldsymbol{X}}\,\boldsymbol{S} + \alpha\overline{\boldsymbol{P}}\,\overline{\boldsymbol{X}} + \beta\boldsymbol{W}\boldsymbol{D})\boldsymbol{B}^T). \tag{11}$$

Finally, a closed-form solution of $\boldsymbol{B}$ can be obtained:

$$\boldsymbol{B} = sign(\overline{\boldsymbol{P}}\,\overline{\boldsymbol{X}}\,\boldsymbol{S} + \alpha\overline{\boldsymbol{P}}\,\overline{\boldsymbol{X}} + \beta\boldsymbol{W}\boldsymbol{D}). \tag{12}$$

It can be seen that the time complexity of $\overline{\boldsymbol{X}}\boldsymbol{S}$ in Eq. (8) and (12) is $O(mn^2)$. However, $\overline{\boldsymbol{X}}\boldsymbol{S} = (\overline{\boldsymbol{X}}\,\overline{\boldsymbol{D}}^T)\overline{\boldsymbol{D}}$, and the time complexity of the latter is only $O(cmn)$. By doing so, all the pair-wise similarities can be utilized, and the consumption of storage and time can be reduced remarkably in our algorithm. The whole optimization algorithm is summarized in Algorithm 1. T is set to 10 in this paper.

2.7 Label Distribution Generation

Inspired by BC-LDL [Wang and Geng, 2018], the proposed DBC-LDL employs the same strategy to generate a label distribution for a future instance. Specifically, the binary codes of a future instance can be generated via the binary coding function in Eq. (1). Then, we use the ANN search strategy to find k nearest training instances in the Hamming space. Benefit from the fast XOR operations, this search procedure is quite efficient. Finally, the mean of the label distributions of these training instances is computed as the predicted label distribution for this future instance.

Algorithm 1 DBC-LDL

```
1: Input: Feature matrix $\boldsymbol{X}$; label distributions matrix $\boldsymbol{D}$; Adjusted label distributions matrix $\overline{\boldsymbol{D}}$; Code length $l$.
2: Output: Projection matrix $\boldsymbol{P}$; Bias vector $\boldsymbol{e}$; Binary codes matrix $\boldsymbol{B}$.
3: $\overline{\boldsymbol{X}} \leftarrow [\boldsymbol{X}; \mathbf{1}_n]$.
4: Initialize $\boldsymbol{B}$, $\overline{\boldsymbol{P}}$, $\boldsymbol{W}$ by randomization.
5: repeat
6:     Fix $\boldsymbol{B}$, $\boldsymbol{W}$, update $\overline{\boldsymbol{P}}$ by Eq. (8);
7:     Fix $\boldsymbol{B}$, $\overline{\boldsymbol{P}}$, update $\boldsymbol{W}$ by Eq. (9);
8:     Fix $\overline{\boldsymbol{P}}$, $\boldsymbol{W}$, update $\boldsymbol{B}$ by Eq. (12);
9: until convergence or reaching the maxiter $T$
10: $\boldsymbol{P} \leftarrow \overline{\boldsymbol{P}}(:, 1 : end - 1)$; $\boldsymbol{e} \leftarrow \overline{\boldsymbol{P}}(:, end)$.
```

2.8 Complexity Analysis

In this subsection, we discuss the time complexity of DBC-LDL. DBC-LDL adopts an iterative optimization strategy to learn variables, where each iteration contains three main steps. The time cost of updating $\overline{\boldsymbol{P}}$, $\boldsymbol{W}$, and $\boldsymbol{B}$ is $O(cmn+lmn+nm^2+m^3)$, $O(cln+nc^2+lc^2+c^3)$, and $O(cmn+lmn+cln)$, respectively. Therefore, the whole time complexity is $O((cmn+lmn+nm^2+m^3+cln+nc^2+lc^2+c^3)t)$, where t is the number of iterations. Since $c, m \ll n$, the training time complexity of DBC-LDL is linear to n.

3 Experiment

We conduct our experiments on five real-world datasets, namely M^{2}B (Multi-Modality Beauty) [Nguyen *et al.*, 2012], s-BU_3DFE (scores-Binghamton University 3D Facial Expression) [Zhou *et al.*, 2015], Twitter_LDL [Yang *et al.*, 2017], Flickr_LDL [Yang *et al.*, 2017], and Ren-CECps [Quan and Ren, 2010], to evaluate our algorithm in terms of both accuracy and efficiency.

M^{2}B and s-BU_3DFE are two regular-scale image datasets. The application task of the former is the facial beauty sense and the task of the latter is the facial expression recognition. Twitter_LDL and Flickr_LDL are two large-scale image datasets. The application task of them is the visual sentiment distributions prediction. Ren-CECps is a large-scale text dataset, and its application task is the text emotion distributions prediction. Due to the page limitation, here we only show statistics of the five real-world datasets in Table 2.

All the experiments are implemented using Matlab on a standard PC with a 2.30GHz Intel CPU and 12GB memory.

3.1 Baselines Evaluation and Protocols

In this paper, the baselines contain one adaptation algorithm AA-kNN, and five representative specialized LDL algorithms, i.e., IIS-LDL, BFGS-LDL, CPNN, LDSVR and BC-LDL. For the proposed DBC-LDL, we empirically set the parameter $\alpha = 10^4$, $\beta = 10^4$ and $\gamma = 10^{-2}$. The code length in DBC-LDL and BC-LDL is same (i.e., 128 bits) for making a fair comparison, and k in DBC-LDL, BC-LDL and AA-kNN is chosen from $\{10, 20, \cdots, 50\}$. For the other baselines, we set the corresponding parameters by following the authors' suggestions. All the results are averaged over 10-fold cross validation in terms of both accuracy and time cost.

Dataset	#Instances	#Features	#Labels
M^{2}B	1,240	250	5
s-BU_3DFE	2,500	243	6
Twitter_LDL	10,045	200	8
Flickr_LDL	10,700	200	8
Ren-CECps	35,096	100	8

Table 2: Statistics of the five real-world datasets.

According to the suggestion in [Geng, 2016], six prevailing metrics are adopted to evaluate the accuracy performance of different algorithms, including Chebyshev distance (Cheb), Clark distance (Clark), Canberra metric (Canber), Kullback-Leibler divergence (K-L), cosine coefficient (Cosine) and intersection similarity (Intersec). Since the first four are used to measure the distance between two distributions, they are the smaller the better. The last two are similarity metrics, so they are the larger the better.

3.2 Experimental Results

The detailed experimental results of the proposed DBC-LDL and other comparing algorithms on the five evaluated datasets are reported in Table 3. To show the accuracy, efficiency and comprehensive performance of these approaches, three rank measurements are introduced, including Accuracy Rank (Acc. Rank), Time Rank and Average Rank (Avg. Rank). Moreover, the ranks of six evaluation metric values are presented in the parentheses right after the corresponding measure values. The average value of these six ranks is denoted as Acc. Rank. In addition, Avg. Rank is the average value between Acc. Rank and Time Rank, which can effectively reflect the comprehensive performance of an algorithm.

From Table 3, it can be seen that DBC-LDL achieves the best accuracy performance on four datasets, and achieves the second best performance on other datasets. Moreover, DBC-LDL can yield the best efficiency performance and comprehensive performance on all the evaluated datasets. Note that the results of LDSVR on Ren-CECps are not reported due to the high memory consumption. Overall, we have the following five observations based on the experimental results:

- On the five evaluated datasets, DBC-LDL ranks *1st* in 63.3% cases and ranks *2st* in 13.3% cases across all the distance and similarity metrics, which can demonstrate its superiority.
- Compared with BC-LDL, DBC-LDL can yield the better accuracy performance on four datasets (i.e., M^{2}B, s-BU_3DFE, Twitter_LDL and Flickr_LDL) and the same accuracy performance on Ren-CECps. The better performance of DBC-LDL can be attributed to its ability to learn good binary codes for training instances by a discrete optimization strategy without quantization loss.
- AA-kNN achieves the desirable accuracy performance on Twitter_LDL and Flickr_LDL, which is also observed in [Wang and Geng, 2018]. This is because that the high-quality features of these datasets are extracted via VGGNet [Simonyan and Zisserman, 2014]. AA-kNN can

Algorithm	Accuracy							Time Cost(s)			Avg. Rank
	Cheb↓	Clark↓	Canber↓	K-L↓	Cosine↑	Intersec↑	Acc. Rank	Train	Test	Sum(Time Rank)	
M^{2}B											
DBC-LDL	**0.3467(1)**	**1.1452(1)**	**2.2768(1)**	0.5855(4)	0.7036(2)	**0.5958(1)**	**1.67**	0.0585	0.0066	**0.0651(1)**	**1.34**
BC-LDL	0.3511(2)	1.1783(2)	2.3433(2)	0.5937(5)	0.6976(5)	0.5866(2)	3.00	0.5192	0.0073	0.5265(3)	3.00
BFGS-LDL	0.3844(6)	1.2857(5)	2.5798(5)	0.7332(6)	0.6463(6)	0.5399(6)	5.67	5.3490	0.0036	5.3526(5)	5.34
IIS-LDL	0.3736(5)	1.3342(7)	2.6718(7)	0.5760(3)	0.6983(4)	0.5457(5)	5.17	330.37	0.0011	330.37(7)	6.09
CPNN	0.3912(7)	1.3289(6)	2.6592(6)	0.9542(7)	0.6363(7)	0.5439(7)	6.67	18.417	0.0124	18.429(6)	6.34
LDSVR	0.3613(3)	1.2700(3)	2.5450(3)	**0.5360(1)**	**0.7233(1)**	0.5782(3)	2.33	0.5435	0.0120	0.5555(4)	3.17
AA-kNN	0.3709(4)	1.2796(4)	2.5678(4)	0.5753(2)	0.7023(3)	0.5649(4)	3.50	0	0.2993	0.2993(2)	2.75
s-BU_3DFE											
DBC-LDL	**0.1003(1)**	**0.3131(1)**	**0.6411(1)**	0.0553(2)	0.9441(4)	**0.8825(1)**	**1.67**	0.0686	0.0245	**0.0931(1)**	**1.34**
BC-LDL	0.1005(2)	0.3176(2)	0.6515(2)	**0.0541(1)**	0.9451(3)	0.8811(2)	2.00	0.5360	0.0246	0.5606(2)	2.00
BFGS-LDL	0.1058(3)	0.3545(4)	0.7417(4)	0.0559(4)	**0.9457(1)**	0.8692(4)	3.33	90.969	0.0040	90.973(6)	4.17
IIS-LDL	0.1211(5)	0.3756(5)	0.8049(5)	0.0654(5)	0.9359(5)	0.8561(5)	5.00	366.47	0.0015	366.47(7)	6.00
CPNN	0.1064(4)	0.3462(3)	0.7288(3)	0.0554(3)	0.9456(2)	0.8703(3)	3.00	38.888	0.0223	38.910(5)	4.00
LDSVR	0.1260(7)	0.3834(6)	0.8261(7)	0.0726(6)	0.9289(6)	0.8518(6)	6.33	1.3039	0.0251	1.3290(4)	5.17
AA-kNN	0.1249(6)	0.3877(7)	0.8217(6)	0.0734(7)	0.9280(7)	0.8517(7)	6.67	0	1.2826	1.2826(3)	4.84
Twitter_LDL											
DBC-LDL	0.2688(4)	**1.9757(1)**	**4.4457(1)**	**0.4447(1)**	0.8350(4)	**0.6740(1)**	**2.00**	0.2299	0.4079	**0.6378(1)**	**1.50**
BC-LDL	0.2673(3)	2.0846(2)	4.8567(2)	0.4741(3)	0.8441(2)	0.6678(3)	2.50	0.7112	0.4030	1.1142(2)	2.25
BFGS-LDL	0.4597(6)	2.4097(5)	6.2895(5)	1.1421(6)	0.6370(6)	0.4325(6)	5.67	48.729	0.0041	48.733(4)	4.84
IIS-LDL	0.4842(7)	2.4037(4)	6.2672(4)	1.1503(7)	0.6102(7)	0.4056(7)	6.00	520.65	0.0025	520.65(7)	6.50
CPNN	0.3058(5)	2.4471(7)	6.4213(7)	0.7539(5)	0.7992(5)	0.6207(5)	5.67	163.34	0.1060	163.45(5)	5.34
LDSVR	0.2661(2)	2.4196(6)	6.3380(6)	0.5531(4)	0.8403(3)	0.6649(4)	4.17	245.38	0.2721	245.65(6)	5.09
AA-kNN	**0.2608(1)**	2.0900(3)	4.8778(3)	0.4645(2)	**0.8538(1)**	0.6739(2)	**2.00**	0	16.261	16.261(3)	2.50
Flickr_LDL											
DBC-LDL	0.2561(4)	**2.0695(1)**	**4.9239(1)**	0.5004(2)	0.8152(4)	0.6608(2)	2.67	0.2521	0.4905	**0.7426(1)**	**1.84**
BC-LDL	0.2548(2)	2.0988(3)	5.0498(3)	0.5041(4)	0.8222(3)	0.6569(3)	3.00	0.8119	0.4860	1.2979(2)	2.50
BFGS-LDL	0.3487(6)	2.2019(5)	5.4894(5)	0.8124(6)	0.7242(5)	0.5421(6)	5.50	55.955	0.0043	55.959(4)	4.75
IIS-LDL	0.3692(7)	2.1904(4)	5.4339(4)	0.8221(7)	0.7030(7)	0.5185(7)	6.67	536.23	0.0027	536.23(6)	6.34
CPNN	0.3240(5)	2.2626(7)	5.7674(7)	0.7903(5)	0.7229(6)	0.5642(5)	5.83	178.62	0.1042	178.72(5)	5.42
LDSVR	0.2556(3)	2.2022(6)	5.5249(6)	0.5016(3)	0.8260(2)	0.6548(4)	4.00	599.93	0.3610	600.29(7)	5.50
AA-kNN	**0.2451(1)**	2.0869(2)	4.9946(2)	**0.4707(1)**	**0.8369(1)**	**0.6679(1)**	**1.33**	0	23.665	23.665(3)	2.17
Ren-CECps											
DBC-LDL	**0.5576(1)**	**1.9447(1)**	**4.2308(1)**	**1.0468(1)**	0.5173(6)	**0.3977(1)**	**1.83**	0.4841	2.1890	**2.6731(1)**	**1.42**
BC-LDL	0.5605(2)	2.2985(2)	5.6406(2)	1.2018(2)	**0.5734(1)**	0.3757(2)	**1.83**	0.7957	2.2011	2.9968(2)	1.92
BFGS-LDL	0.6037(4)	2.6492(5)	7.3189(4)	1.4993(5)	0.5455(2)	0.3041(4)	4.00	202.06	0.0041	202.06(4)	4.00
IIS-LDL	0.6245(6)	2.6544(6)	7.3573(6)	1.5643(6)	0.5187(5)	0.2724(6)	5.83	424.74	0.0032	424.74(5)	5.42
CPNN	0.6204(5)	2.6478(4)	7.3203(5)	1.4759(4)	0.5337(4)	0.2866(5)	4.50	484.69	0.2703	484.96(6)	5.25
LDSVR	—	—	—	—	—	—	—	—	—	—	—
AA-kNN	0.5933(3)	2.3934(3)	6.0730(3)	1.3424(3)	0.5355(3)	0.3283(3)	3.00	0	157.21	157.21(3)	3.00

Table 3: Experimental results on the five evaluated datasets. The best results are highlighted in bold face.

obtain the accurate search results by using these high-quality features.

- Compared with IIS-LDL, BFGS-LDL can achieve higher accuracy results with the lower training time cost on most datasets. The training time of LDSVR growing rapidly with the size of datasets increasing, because its time complexity is $O(n^2)$.

- Benefit from the efficient optimization strategy and the fast binary code based ANN search approach, the total time cost of DBC-LDL is lowest on all the evaluated datasets, which verifies the efficiency of our method.

3.3 Parameter Analysis

There are three parameters in our objective function, including α, β and γ. Due to the limitation of space, we only report the Cheb and Intersec results and use them to study the performance variation with respect to the different parameter values of α, β and γ. The results of other metrics are similar. The experiments are conducted by varying the value of one parameter while fixing the other two to the settings mentioned in Subsection 3.1. Here, we vary α in the range of $\{10^{-2}, 10^{-1}, \cdots, 10^{5}\}$, β in the range of $\{0, 1, \cdots, 10^{6}\}$ and γ in the range of $\{10^{-4}, 10^{-3}, \cdots, 10^{3}\}$. Figure 1 shows the results on the testing dataset (i.e., s-BU_3DFE). We can find that our method can achieve satisfactory results, when $\alpha \in [10^3, 10^4]$, $\beta \in [10^4, 10^5]$ and $\gamma \in [10^{-2}, 10^{-1}]$.

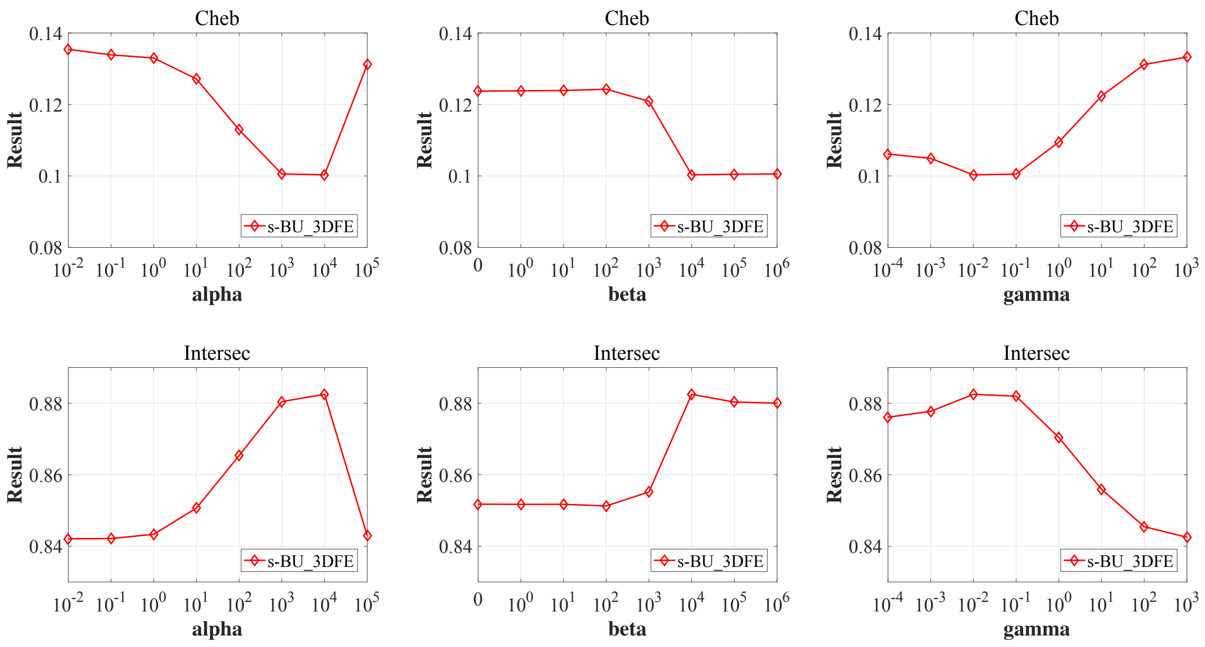

Figure 1: The parameter analysis of DBC-LDL on s-BU_3DFE with Cheb and Intersec metrics.

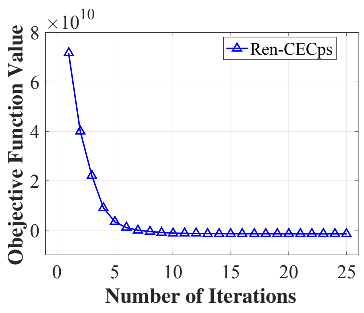

Figure 2: The convergence curve of DBC-LDL on Ren-CECps.

s-BU_3DFE						
Code Length	Cheb↓	Clark↓	Canber↓	K-L↓	Cosine↑	Intersec↑
32 bits	0.1075	0.3350	0.6881	0.0619	0.9378	0.8738
64 bits	0.1022	0.3198	0.6551	0.0566	0.9430	0.8802
128 bits	0.1003	0.3131	0.6411	0.0553	0.9441	0.8825
256 bits	0.0990	0.3085	0.6311	0.0540	0.9453	0.8843
512 bits	0.0978	0.3044	0.6226	0.0528	0.9464	0.8857
Ren-CECps						
Code Length	Cheb↓	Clark↓	Canber↓	K-L↓	Cosine↑	Intersec↑
32 bits	0.5749	2.0239	4.4613	1.1372	0.4960	0.3807
64 bits	0.5646	1.9849	4.3655	1.0991	0.5078	0.3898
128 bits	0.5576	1.9447	4.2308	1.0468	0.5173	0.3977
256 bits	0.5514	1.9312	4.1904	1.0325	0.5254	0.4037
512 bits	0.5448	1.9022	4.1231	1.0182	0.5328	0.4099

Table 4: Experimental results on s-BU_3DFE and Ren-CECps with respect to different code lengths.

3.4 Convergency Analysis

In this part, we evaluate the convergence of the proposed DBC-LDL. Figure 2 plots the convergence curve of the objective function during training on Ren-CECps. It can be observed that our method can successfully converge within 10 iterations and the value of the objective function decrease slowly after 4 iterations. The time consumption for optimization of DBC-LDL is not expensive, because it can converge quite fast.

3.5 Binary Code Length Analysis

In this subsection, we study the accuracy performance of our method regarding different code lengths. Table 4 summarizes the evaluation results on s-BU_3DFE and Ren-CECps. We can learn that DBC-LDL achieves better accuracy performance with longer binary codes. It means that more useful information can be preserved by DBC-LDL when learning longer binary codes.

4 Conclusion

In this paper, we have presented a novel LDL method (DBC-LDL) to tackle the task of large-scale LDL. In DBC-LDL, the pair-wise semantic similarities and original label distributions are taken into consideration for learning precise binary codes. Moreover, an efficiently discrete optimization strategy is proposed to learn binary codes directly for training instances. In addition, a binary code based ANN search method is employed to predict label distributions for testing instances. Experimental results on five benchmark LDL datasets show that DBC-LDL outperforms several state-of-the-art LDL approaches. The binary coding part of DBC-LDL is an one-step method. In future work, we plan to develop a two-step discrete method to solve the large-scale LDL problem.

Acknowledgments

This research was supported by the National Key Research & Development Plan of China (No. 2017YFB1002801), the National Science Foundation of China (61622203), the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Collaborative Innovation Center of Wireless Communications Technology.

References

[Gao *et al.*, 2017] Bin-Bin Gao, Chao Xing, Chen-Wei Xie, Jianxin Wu, and Xin Geng. Deep label distribution learning with label ambiguity. *IEEE Transactions on Image Processing*, 26(6):2825–2838, 2017.

[Geng and Hou, 2015] Xin Geng and Peng Hou. Pre-release prediction of crowd opinion on movies by label distribution learning. In *IJCAI*, pages 3511–3517, 2015.

[Geng *et al.*, 2013] Xin Geng, Chao Yin, and Zhi-Hua Zhou. Facial age estimation by learning from label distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(10):2401–2412, 2013.

[Geng, 2016] Xin Geng. Label distribution learning. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1734–1748, 2016.

[Gui *et al.*, 2018] Jie Gui, Tongliang Liu, Zhenan Sun, Dacheng Tao, and Tieniu Tan. Fast supervised discrete hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(2):490–496, 2018.

[Jiang *et al.*, 2018] Qing-Yuan Jiang, Xue Cui, and Wu-Jun Li. Deep discrete supervised hashing. *IEEE Transactions on Image Processing*, 27(12):5996–6009, 2018.

[Kang *et al.*, 2016] Wang-Cheng Kang, Wu-Jun Li, and Zhi-Hua Zhou. Column sampling based discrete supervised hashing. In *AAAI*, pages 1230–1236, 2016.

[Li *et al.*, 2017] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. Deep supervised discrete hashing. In *Advances in Neural Information Processing Systems*, pages 2482–2491, 2017.

[Liu *et al.*, 2011] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Hashing with graphs. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 2011.

[Liu *et al.*, 2014] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang. Discrete graph hashing. In *Advances in Neural Information Processing Systems*, pages 3419–3427. 2014.

[Luo *et al.*, 2018a] Xin Luo, Liqiang Nie, Xiangnan He, Ye Wu, Zhen-Duo Chen, and Xin-Shun Xu. Fast scalable supervised hashing. In *SIGIR*, pages 735–744, 2018.

[Luo *et al.*, 2018b] Xin Luo, Ye Wu, and Xin-Shun Xu. Scalable supervised discrete hashing for large-scale search. In *Proceedings of the 2018 World Wide Web conference on World Wide Web*, pages 1603–1612. ACM, 2018.

[Nguyen *et al.*, 2012] Tam V Nguyen, Si Liu, Bingbing Ni, Jun Tan, Yong Rui, and Shuicheng Yan. Sense beauty via face, dressing, and/or voice. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 239–248, Nara, Japan, 2012.

[Quan and Ren, 2010] Changqin Quan and Fuji Ren. Sentence emotion analysis and recognition based on emotion words using ren-cecps. *International Journal of Advanced Intelligence*, 2(1):105–117, 2010.

[Ren and Geng, 2017] Yi Ren and Xin Geng. Sense beauty by label distribution learning. In *IJCAI*, pages 2648–2654, 2017.

[Sarwar *et al.*, 2001] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[Shen *et al.*, 2015] 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*, pages 37–45, 2015.

[Shen *et al.*, 2017] Wei Shen, Kai Zhao, Yilu Guo, and Alan L Yuille. Label distribution learning forests. In *Advances in Neural Information Processing Systems*, pages 834–843. 2017.

[Shi *et al.*, 2017] Xiaoshuang Shi, Fuyong Xing, Kaidi Xu, Manish Sapkota, and Lin Yang. Asymmetric discrete graph hashing. In *AAAI*, pages 2541–2547, 2017.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Wang and Geng, 2018] Ke Wang and Xin Geng. Binary coding based label distribution learning. In *IJCAI*, pages 2783–2789, 2018.

[Weiss *et al.*, 2009] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009.

[Xing *et al.*, 2016] Chao Xing, Xin Geng, and Hui Xue. Logistic boosting regression for label distribution learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4489–4497, 2016.

[Yang *et al.*, 2017] Jufeng Yang, Ming Sun, and Xiaoxiao Sun. Learning visual sentiment distributions via augmented conditional probability neural network. In *AAAI*, pages 224–230, 2017.

[Zhou *et al.*, 2015] Ying Zhou, Hui Xue, and Xin Geng. Emotion distribution recognition from facial expressions. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1247–1250. ACM, 2015.