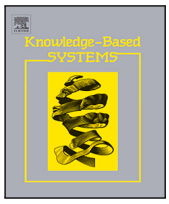
Knowledge-Based Systems 320 (2025) 113666

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



RG4LDL: Renormalization group for label distribution learning

Chao Tan [a,*], Sheng Chen [b], Jiaxi Zhang [a], Zilong Xu [a], Xin Geng [c], Genlin Ji [a]

[a] School of Computer and Electronic Information/School of Artificial Intelligence, Nanjing Normal University, Nanjing, 210023, China
[b] School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK
[c] School of Computer Science and Engineering, Southeast University, Nanjing, 210096, China

ARTICLE INFO

Keywords:
Label distribution learning
Unsupervised neural network
Renormalization group
Restricted Boltzmann machine

ABSTRACT

Label distribution learning (LDL) is an effective paradigm to address label ambiguity by modeling the relevance of multiple labels to an instance. However, existing LDL methods suffer from challenges such as high model complexity, slow convergence, and limited availability of label distribution-annotated training data. To tackle these issues, we propose RG4LDL, a novel framework that integrates the renormalization group (RG) principle with LDL for the first time. RG4LDL employs a restricted Boltzmann machine (RBM)-based neural network to iteratively extract relevant degrees of freedom, thereby optimizing feature learning and improving predictive accuracy. By combining unsupervised RG learning and supervised LDL prediction in an end-to-end manner, RG4LDL achieves both efficiency and effectiveness. Experimental results on 13 real-world datasets and a synthetic toy dataset demonstrate that RG4LDL significantly outperforms state-of-the-art LDL methods in terms of predictive accuracy and computational efficiency. These results highlight the potential of RG4LDL as a benchmark solution for label distribution learning tasks.

1. Introduction

As one of the core research fields of artificial intelligence, machine learning has gradually shifted from academic research to practical applications and has been widely used in multiple scientific and engineering fields, such as medical diagnosis [1], financial analysis [2], intelligent transportation [3], text analysis [4], and many others.

Machine learning problems are often inherently ambiguous, with the ambiguity manifesting not only in the examples but also in the labels. The problem of label ambiguity has long been an active research area in machine learning and data mining. Traditional methods are largely based on single-label learning (SLL) [5] or multi-label learning (MLL) [6], with the former assigning a single label to each example and the latter assigning multiple labels to each sample. In real life, things often contain multiple semantic meanings simultaneously. For instance, in image annotation [7], an image may contain multiple labels like 'human', 'ocean', 'cloud' at the same time.

Geng [8] proposed a novel learning paradigm called label distribution learning (LDL) to address label ambiguity. It assigns each instance a label distribution, whose elements are called label description degrees. Specifically, given an instance, LDL assigns each label a real-value to indicate the relevance of the label to the instance. For example, in image annotation [7], 'ocean' and 'cloud' are assigned label description degrees of 0.6 and 0.4, respectively, directly modeling their differing relevance. Essentially, MLL aims to determine 'which labels describe the instance', while LDL solves 'how much does each label describe it' [8]. Thus, LDL is more general than MLL. It has found wide applications, such as age estimation [9,10], emotion recognition [11] and noisy label learning [12,13].

Therefore, LDL inherently involves mining correlations between labels and can serve as a means for utilizing these correlations in SLL or MLL tasks. As an important baseline, the IIS-LLD algorithm [10] used the maximum entropy model as a primary assumption in the absence of additional information, although there is no specific evidence supporting it in the context of age estimation. Alternatively, by using a three-layer neural network (NN) for approximation, the CPNN [10] offers a way to eliminate this assumption. However, it leads to a large number of weights between the hidden and output layers. With a limited number of training samples, an excessive number of weights can cause difficulties for the learning algorithm to converge. In this paper we characterize the feature extractor of the LDL process by random variables called degrees of freedom. This aspect constitutes an important focus of this paper. To elaborate further, the LDL methods [10,14] utilize the correlations between adjacent labels during the machine learning stage but after the features are extracted, these correlations are ignored. The proposed method in this paper by contrast is an end-to-end deep learning method that takes advantage of this correlation information in both feature learning and classifier learning.

* Corresponding author.
E-mail addresses: 73022@njnu.edu.cn (C. Tan), sqc@ecs.soton.ac.uk (S. Chen), 222202040@njnu.edu.cn (J. Zhang), 222202021@njnu.edu.cn (Z. Xu), xgeng@seu.edu.cn (X. Geng), glji@njnu.edu.cn (G. Ji).

https://doi.org/10.1016/j.knosys.2025.113666
Received 29 December 2024; Received in revised form 9 April 2025; Accepted 28 April 2025
Available online 10 May 2025

Deep convolutional neural networks (CNNs) have a natural advantage in feature learning. Existing CNN frameworks can be divided into classification and regression models based on different optimization objectives. However, existing deep learning methods are unable to utilize label ambiguity information. Label distribution can naturally describe the fuzzy information between all possible labels by utilizing the correlations between adjacent labels [15]. To achieve this objective, a multi-layer NN may be constructed with samples and their corresponding labels as inputs, and the trained NN outputs the label distributions for the examples. This is another research focus of this paper.

Although the aforementioned advancements have opened up new avenues for studying machine learning algorithms, few works have used them to enhance learning algorithms, and there exists no study that utilizes information-theoretic NNs in LDL. This is because identifying relevant degrees of freedom is challenging, particularly in LDL predictive tasks. To combat this challenge, this paper introduces renormalization group (RG) to LDL process for the first time with an unsupervised NN that identifies relevant degrees of freedom in a spatial region iteratively. Concretely, for a physical system, e.g., feature extractor of the LDL process, represented by a set of degrees of freedom or random variables $\boldsymbol{X}=\{\boldsymbol{x}_i\}$, we consider a new smaller set of degrees of freedom $\boldsymbol{H}=\{\boldsymbol{h}_j\}$, the coarse-grained variables, whose dependence on $\boldsymbol{X}$ is found by maximizing an information theoretic measure, specifically, the mutual information of $\boldsymbol{X}$ and $\boldsymbol{H}$, and lends itself naturally to an NN implementation. The conditional probability distribution to predict for new samples is given by a Boltzmann probability distribution with the energy $E(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\eta})$ having the parameters $\boldsymbol{\eta}$, and the parameters to learn $\boldsymbol{\eta}$ include all the trainable parameters of the implementing NN. Since all the conditional probability distributions $d_{\boldsymbol{x}}^{y}=P(y|\boldsymbol{x})$ normalize, the mutual information is invariant under homeomorphic transformation of $\boldsymbol{X}$ or $\boldsymbol{H}$, which is the basis for training the NN with restricted Boltzmann machine (RBM) in this work.

To sum up, in this paper, we propose the RG for LDL (RG4LDL). Specifically, we utilize the RG, namely, RBM, approach to address the drawback of excessive degrees of freedom in LDL predictive tasks, thereby resolving the complexity and convergence challenges encountered when dealing with high-dimensional LDL models, as well as to tackle the main difficulty of lack of label distribution annotated training data in many real-world LDL problems. This significantly speeds up the LDL process as well as enhances the LDL predictive accuracy. To our best knowledge, this is the first time that the RBM theory is applied to enhance LDL. Our contributions are as follows.

- We introduce the RBM [16], a type of probabilistic graphical model that can be explained by stochastic NNs, into the LDL process to identify relevant degrees of freedom in the feature space iteratively.
- We verify our training algorithm based on evaluating the mutual information between separated regions in the label space to predict label distributions on the test set.
- Experiments demonstrate that our RG4LDL is significantly faster than the existing state-of-the-art LDL methods, and it outperforms these LDL methods in terms of LDL predictive accuracy.

The remainder of this paper is structured as follows. Section 2 introduces the related works, and Section 3 is dedicated to our proposed RG4LDL framework. Experiment design is presented in Section 4. Extensive experimental results are provided in Section 4, and a further discussion of the results is offered in Section 6. Section 7 concludes the paper. Appendix lists the main symbols used in the paper.

2. Related works

We review the existing studies related to the topics of ambiguity in learning, deep learning-based LDL, information-theoretic NN, and restricted Boltzmann machine (RBM), which form the foundation of our proposed framework.

2.1. Ambiguity in the learning process

Ambiguity in the learning process has emerged as a focal issue in the field of machine learning in recent years. Concept learning fundamentally involves establishing a mapping from instances to concept labels. Ambiguity can arise at either end of this mapping. To date, MLL has become a mainstream technique for addressing label-end ambiguity and has been successfully applied to various practical problems with conceptual ambiguity, including document classification [17], image classification and labeling [18], object recognition [19], gene analysis [17,18], web page classification [17], natural scene classification [17,18], audio labeling and retrieval [18,20], etc.

To dig deeper, in the traditional machine learning paradigm, there are two main ways of data annotation: (1) one example is assigned a label; and (2) one example is assigned with multiple labels. SLL assumes that all the examples in the training set are labeled in the first way, while MLL [6] allows the training examples to be labeled in the second way. Therefore, MLL can deal with the ambiguity that an example belongs to multiple categories. Both SLL and MLL aim to answer an essential question, that is, 'which labels can describe the example?'. However, they do not directly answer another deeper question, 'how does each label describe the example?', that is, 'how does each label describe the relative importance of the example?'.

In fact, the relevance or irrelevance of each label corresponding to an example is essentially relative in real-world tasks. If multiple tags represent an instance, the relative importance between the tags is more likely to be different, rather than exactly equal. LDL, which is a relatively new machine learning paradigm, explicitly characterizes the importance of different labels to the example with label distribution, and it has achieved good results in multiple application fields. For instance, in the studies investigating protein-tumor relationships [21], it has been observed that certain proteins may be associated with multiple tumors but exhibit distinct expression levels across them, with higher expression levels indicating stronger associations. Predicting protein-tumor associations and quantifying their differences based on variations in protein expression levels hold significant implications for early cancer detection and treatment.

The LDL framework [8] offers a way of resolving the label ambiguity. Since its proposal, LDL has received extensive attention. For example, Ren et al. [22] proposed an LDL algorithm by utilizing label-specific features, simultaneously learning the common features of all labels and specific features of each label to enhance LDL models. The authors of [23] applied a label correlation matrix based on low-rank approximation to capture global label correlations, and further modified the label correlation matrix using local label correlations between sample pairs. Jia et al. [24] proposed an algorithm to train LDL models by introducing the ranking loss function, which offers state-of-the-art performance.

LDL was initially applied to age estimation. The main difficulty in age estimation was the lack of sufficient datasets, as collecting facial images of the same person from childhood to old age requires a very long time span. Also, in the existing datasets, one facial image corresponded to one age label. Inspired by the LDL idea, Geng et al. [10] designed the IIS-LDL algorithm. One of the main assumptions of IIS-LDL is that the probability of label conditioned on example $p(y|\boldsymbol{x})$ is derived as a maximum entropy model [25]. Although this is a reasonable assumption without additional information, there is no particular evidence to support it in age estimation problems. Alternatively, using a three-layer NN to approximate $p(y|\boldsymbol{x})$ is a way to eliminate this assumption. The natural design of such an NN has input units receiving $\boldsymbol{x}$ with dimensionality q, which is typically very large, and c output units, each outputting the description degree of a label y. Geng et al. [10] introduced such NNs to age estimation, called CPNN. An NN based LDL model typically contains large number of weights. Each weight can be viewed as a degree of freedom or a variable. The higher the degrees of freedom and the more variables, the more complex the

model and the stronger its capability. However, the more the NN's weights, the easier it is to overfit and more sensitive to noise. Moreover, supervised training is required for the NN of CPNN. Gradient descent type algorithms are generally used in training NN based LDL models. With limited training samples, any learning algorithm will struggle to converge if the NN has too many weights.

The aforementioned issues of the existing LDL paradigm are generic and they are not restricted to the age estimation application. The LDL approach proposed in this paper is specifically designed to resolve the complexity and convergence challenges encountered by the existing LDL schemes as well as to address the problem of lack of label distribution annotated training data in real-world LDL tasks.

2.2. Deep learning-based LDL

With the advent of deep learning, LDL research has increasingly focused on leveraging advanced neural architectures. CNNs and recurrent neural networks have been widely used for feature extraction and label distribution prediction, particularly in tasks such as age estimation and emotion recognition [10,24]. These methods benefit from the strong representation learning capabilities of deep networks but often fail to fully exploit label correlations or address label ambiguity effectively.

More recently, contrastive learning techniques, as explored in [26], have been applied to LDL tasks to enforce label correlation by leveraging contrastive learning. These advancements have shown superior performance compared to traditional CNN-based methods, particularly in scenarios with limited labeled data. By integrating features and logical labels through contrastive learning and ensuring label attribute consistency, the paper [27] proposed a novel and effective method for recovering label distributions in LDL. A novel framework introduced in [28], called the DCLE, integrated dual-view descriptions and dual contrastive learning for LDL to produce high-quality label distributions.

The proposed RG4LDL framework builds upon these advancements by introducing the renormalization group (RG) principle and RBMs for feature extraction. Unlike traditional NNs or contrastive learning methods, which focus on leveraging attention mechanisms or representation learning, RG4LDL addresses the challenge of reducing degrees of freedom in high-dimensional feature spaces. This makes RG4LDL particularly effective in scenarios where computational efficiency and feature abstraction are critical.

2.3. Information-theoretic neural network

Hinton developed the Boltzmann machine [29], a type of NN, based on Hopfield network [30]. The Boltzmann machine randomly sets the values of neurons according to a probability distribution, ensuring that the network converges to a thermal equilibrium state fully determined by 'energy'. In the Hopfield network, each node (neuron) represents a function. Nodes are updated according to a deterministic rule, which guarantees that the network's defined 'energy' decreases monotonically towards a local minimum. Thus, training a Hopfield network involves minimizing this 'energy'. However, the Hopfield network may converge to local minima rather than the global minimum. To avoid this issue, Hinton introduced a stochastic update rule, leading to the development of the Boltzmann machine. NNs based on the Boltzmann machine can be considered powerful empirical tools for testing the Boltzmann distribution. In other words, where NNs are effective, the Boltzmann distribution is also applicable.

Many machine learning techniques still lack solid theoretical foundations, which hinders our understanding of their effects. Scholars have endeavored to describe some learning algorithms, such as RG [31], with methods from statistical physics. RG is an iterative coarse-graining scheme that allows relevant features to be extracted when physical systems are examined at different scales. Mehta and Schwab [32] constructed an exact mapping from the variational RG first introduced by Kadanoff [31] to the deep learning framework based on RBMs. Their research shows that deep learning algorithms can learn relevant features from data in a scheme similar to RG.

More specifically, RG and deep NNs (DNNs) are both NN models, but there are some differences. RG is more flexible than DNN, capable of modeling dependencies between variables rather than just feature extraction. Compared to DNN, RG can better represent dependencies between variables, making RG more suitable for structured inference. RG can explicitly express relationships between variables through defined potential functions, while relationships in DNN are more implicit. This enables RG to better explain the reasoning process and deal with uncertain knowledge, which is important for many LDL tasks. RG can represent various graph structures, suitable for knowledge graphs, molecular structures and other complex network topologies commonly seen in LDL. By contrast, DNN relies more on grid-like topologies. RG can perform inference and learning simultaneously, seamlessly integrating graph structure learning and inference. It can learn graph topologies and perform complex inference over graphs. The modeling flexibility, interpretability and ability to represent complex relationships make RG particularly suitable for graph model learning and LDL tasks.

For LDL tasks, the input layer of the NN inputs the feature vector of the example, and the output layer of the NN contains c output units, each of which outputting a label description, where c is the number of labels for each example. In machine learning and optimization combination problems, the most commonly used method is the gradient descent method, e.g., the back propagation (BP) algorithm [33]. The training goal of the BP NN algorithm is to make the c output values of each training example as close as possible to its label distributions after passing the example through the NN. In this task, the more neurons that the NN has, the larger the degree of freedom is or more adjustable weight variables are. It is well-known however that the more complex a model is, the more likely the model is to overfit and be too sensitive to noise. Moreover, the lack of supervision information makes it difficult to evaluate the training and generalization errors of unsupervised samples.

The RG method adopted in this paper is a representation of DNN, consisting NNs with fully-connected layer structures. The fully-connected networks are responsible for feature transformation and the final classification or regression of abstract feature outputs. The two complement each other to achieve end-to-end structured model learning. NN can learn local features of grid data, while RG can model non-Euclidean space structural data, which demonstrates the advantage of integration and evolution of NN and RG method.

2.4. Restricted Boltzmann machine

The two basic functions of NNs are feature transformation and classification/regression prediction. Each NN model encompasses at least one of these functions. RBM [16] can be employed for feature transformation, wherein the principle lies in variable transformation based on probability distribution. RBM consists of two layers: the visual layer representing the input variables and the hidden layer representing the transformed variables. Within each layer, the variables are disconnected, while between layers they are fully connected. Each node corresponds to a dimension, with independence among dimensions within the same layer. More specifically, RBM is a structure based on bipartite graphs, which are special models in graph theory. In an undirected graph, the vertices can be divided into two non-intersecting subsets, and each edge connects vertices from different subsets. RBM aims to find the mapping relationship between the input feature X and the transformed feature $\widehat{X}$ which is another representation of X. While traditional NN models typically use a decision function $Y = f(X)$ to represent this mapping relationship, RBM employs a joint probability distribution $P(X, \widehat{X})$ for this purpose. Both discriminant functions and joint probability distributions can describe the underlying variable relationships. Therefore, RBM provides an alternative approach to solving

feature extraction problems by describing variable relationships from a probabilistic perspective.

The dichotomous (probabilistic) graph represents the dependency between variables, but cannot describe the specific form of $P(X, \hat{X})$. If the specific form $P(X, \hat{X})$ is known, the parameters of the probability distribution can be easily solved by the maximum likelihood function. When the specific form $P(X, \hat{X})$ is unknown but the dependency between variables is known, the probability distribution can be constructed by means of the energy based model [29]. Hence according to the probability graph, by first defining an energy function, the corresponding probability distribution can be found according to this energy function. On the other hand, RBM is a stochastic NN rooted in statistical mechanics [29]. The learning of RBM is unsupervised. Inspired by statistical mechanics, an energy function is introduced. This energy function is a measure to describe the state of the entire system. The minimum value of the energy function corresponds to the most stable state of the system. The conclusion of statistical mechanics shows that any probability distribution can be transformed into an energy-based model, and distribution learning can exploit the unique properties and learning process of this energy model.

Inspired by the aforementioned related works, this paper proposes an unsupervised NN for LDL, called RG for LDL (RG4LDL), which models $P(y|\boldsymbol{x})$ using the RBM type NN, learns the latent representation $\boldsymbol{H}$ from $\boldsymbol{X}$ that retains the maximum label-related information, and predicts the label distribution of new samples based on the latent representation.

3. Proposed RG4LDL

This section details the proposed RG4LDL framework, which includes the problem definition, the unsupervised RG learning process, and the supervised LDL process.

3.1. Definition of LDL problem

Denote the ith example by $\boldsymbol{x}_i$, its jth label by y_j^i, and the description degree of y_j^i to $\boldsymbol{x}_i$ by $d_{\boldsymbol{x}_i}^{y_j^i}$. We have $d_{\boldsymbol{x}_i}^{y_j^i} \in [0,1]$ and $\sum_j d_{\boldsymbol{x}_i}^{y_j^i} = 1$, which means that the label set fully describes an example and all the labels are correct labels for describing the example. The label distribution of $\boldsymbol{x}_i$ is denoted by $\boldsymbol{d}_i = \left[d_{\boldsymbol{x}_i}^{y_1^i}, d_{\boldsymbol{x}_i}^{y_2^i}, \ldots, d_{\boldsymbol{x}_i}^{y_{\bar{c}}^i}\right]$, where $\bar{c}$ is the number of possible labels.

Borrowing from statistical theory, the description degree of label y to example $\boldsymbol{x}$ can be represented as a conditional probability $d_{\boldsymbol{x}}^{y} = P(y|\boldsymbol{x})$. Let $\mathcal{X} \in \mathbb{R}^q$ denote the feature space of instances and $\mathcal{Y}$ the label space. Given a training set $S = \{(\boldsymbol{x}_i, \boldsymbol{d}_i), 1 \le i \le n\}$, the goal of LDL is to learn from S a conditional probability mass function $P(y|\boldsymbol{x})$. We further assume that the parametric model of $P(y|\boldsymbol{x})$ is $P(y|\boldsymbol{x};\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the parameter matrix. Given S, the goal of LDL is then to find a $\boldsymbol{\theta}$ such that given an instance $\boldsymbol{x}_i$, $P(y|\boldsymbol{x};\boldsymbol{\theta})$ can generate a label distribution as close as possible to the true label distribution $\boldsymbol{d}_i$ of $\boldsymbol{x}_i$. If Kullback–Leibler divergence is used to measure the distance between two distributions, the optimal parameter matrix $\boldsymbol{\theta}^\star$ is obtained as:

$$\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \sum_{j=1}^{\bar{c}} d_{\boldsymbol{x}_i}^{y_j^i} \ln P\left(y_j^i \middle| \boldsymbol{x}_i; \boldsymbol{\theta}\right). \tag{1}$$

3.2. Unsupervised RG learning

Probability labels are mainly used in cases where the true labels of instances are uncertain. In practice, determining the probability (or confidence) of label occurrence is often challenging. In most cases, it relies on prior knowledge from human experts, which is a highly subjective and varying process. Therefore, it is necessary to determine the relevant degrees of freedom in the space concerned. The problem of learning from probability labels has not been widely studied so far, which is often a challenging conceptual step.

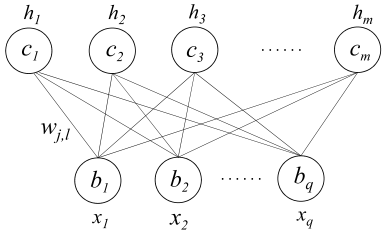


Fig. 1. Structure of RBM.

This paper introduces an NN algorithm to iteratively identify relevant degrees of freedom in the feature space. We consider the underlying system represented by a set of degrees of freedom or random variables $\mathcal{X} = \{\boldsymbol{x}_i\} \in \mathbb{R}^q$, where q denotes the feature dimension of instances. As discussed in Sections 2.3 and 2.4, there exists a new smaller set of degrees of freedom $\mathcal{H} = \{\boldsymbol{h}_i\}$, i.e., coarse-grained variables, whose dependence on $\mathcal{X}$ can be found through an algorithm that maximizes an information-theoretic metric. This algorithm is suitable for an NN implementation. The conditional probability distribution for predicting new samples is then given by the Boltzmann distribution with the energy $E(\boldsymbol{x}, \boldsymbol{h}|\boldsymbol{\eta})$, where $\boldsymbol{\eta}$ denotes the parameters of the energy model, and all the conditional probability distributions $d_{\boldsymbol{x}}^{y_j} = P(y_j|\boldsymbol{x})$ are invariant under any homeomorphic degrees of freedom in $\mathcal{X}$ or $\mathcal{H}$. This is the principle for using RBMs in NN training.

Concretely, given a dataset $\boldsymbol{X} = \left[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\right] \in \mathbb{R}^{q\times n}$ of n samples, we define a joint probability distribution $P(\boldsymbol{X}, \boldsymbol{H})$ that describes how the relevant degrees of freedom $\boldsymbol{H} \in \mathbb{R}^{m\times n}$ (called degree of freedom outputs) depend on 'random' variables $\boldsymbol{X}$, where m is the number of hidden layer neurons, and typically $m \le q$. This is found by maximizing information-theoretic measures using an algorithm that lends itself naturally to NN implementation. The conditional probability distribution to predict new samples is given by a Boltzmann probability distribution with the energy $E(\boldsymbol{x}, \boldsymbol{h}|\boldsymbol{\eta})$. By taking a probabilistic view, treating $\boldsymbol{X}$ and $\boldsymbol{H}$ as random variables, we extract relevant degrees of freedom $\boldsymbol{H}$ from $\boldsymbol{X}$, i.e., learn latent representations $\boldsymbol{H}$ from $\boldsymbol{X}$ that retain maximum label correlation information, and make label distribution predictions based on the joint representations. We employ RBM which is well-suited to approximating arbitrary data distributions. RBM is a stochastic NN with a two-layer structure, specifically, a 'visible layer' of q nodes (corresponding to the feature dimension of the input data) and a 'hidden layer' of m nodes (corresponding to local degrees of freedom). There are full symmetric connections between the two layers but no connections within a layer and no feedback.

Fig. 1 depicts the structure of RBM, where $w_{j,l}$ is the connection weight between the jth visible node and the lth hidden node, b_j is the bias term of the jth visible node and c_l is the bias term of the lth hidden node. The degrees of freedom outputs are computed according to

$$h_l = \sigma\left(\sum_{j=1}^{q} w_{j,l} x_j + c_l\right),\ 1 \le l \le m, \tag{2}$$

where $\sigma(\cdot)$ is the sigmoid function. As the connections between the two layers are symmetric, the visible variables are related to h_l, $1 \le l \le m$, by

$$x_j = \sigma\left(\sum_{l=1}^{m} w_{j,l} h_l + b_j\right),\ 1 \le j \le q. \tag{3}$$

After the RBM has been trained, the 'output' vector $\boldsymbol{h} = \left[h_1, \ldots, h_m\right]^{\mathrm{T}}$ of the RBM for the given 'input' vector $\boldsymbol{x} = \left[x_1, \ldots, x_q\right]^{\mathrm{T}}$ can be written as

$$\boldsymbol{h} = \sigma\left(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{c}\right) = \sigma\left(f(\boldsymbol{x}, \boldsymbol{\eta})\right), \tag{4}$$

which is the extracted feature vector for sample $\boldsymbol{x}$, where $\boldsymbol{W} = \left[w_{j,l}\right] \in \mathbb{R}^{q\times m}$, $\boldsymbol{b} = \left[b_1, \ldots, b_q\right]^{\mathrm{T}}$, $\boldsymbol{c} = \left[c_1, \ldots, c_m\right]^{\mathrm{T}}$, and the parameters $\boldsymbol{\eta}$ include $\boldsymbol{W}$, $\boldsymbol{b}$ and $\boldsymbol{c}$.

Algorithm 1: RBM training process

Data: Input set $\boldsymbol{X} = \{\boldsymbol{x}_t\}_{t=1}^n$ of n samples, number of hidden layer neurons m, number of maximum iterations $I_{\max}$, learning rate ε;
Result: Weight matrix $\boldsymbol{W}$, bias vectors $\boldsymbol{b}$ and $\boldsymbol{c}$;
Initialization: Randomly assign $\boldsymbol{W}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ with small values;
for $k = 1, 2, \ldots, I_{\max}$ **do**
  for $t = 1, 2, \ldots, n$ **do**
    Given sample $\boldsymbol{x}_t$, compute degrees of freedom outputs $\boldsymbol{h}_t$:
    for $l = 1, 2, \ldots, m$ **do**
      $[\boldsymbol{h}_t]_l = P([\boldsymbol{h}_t]_l|\boldsymbol{x}_t) = \sigma\left(c_l + \sum_{j=1}^{q} w_{j,l}[\boldsymbol{x}_t]_j\right)$;
    Given $\boldsymbol{h}_t$, compute visible variables $\boldsymbol{x}_t^*$:
    for $j = 1, 2, \ldots, q$ **do**
      $[\boldsymbol{x}_t^*]_j = P([\boldsymbol{x}_t^*]_j|\boldsymbol{h}_t) = \sigma\left(b_j + \sum_{l=1}^{m} w_{j,l}[\boldsymbol{h}_t]_l\right)$;
    Given $\boldsymbol{x}_t^*$, update degrees of freedom outputs $\boldsymbol{h}_t^*$:
    for $l = 1, 2, \ldots, m$ **do**
      $[\boldsymbol{h}_t^*]_l = P([\boldsymbol{h}_t^*]_l|\boldsymbol{x}_t^*) = \sigma\left(c_l + \sum_{j=1}^{q} w_{j,l}[\boldsymbol{x}_t^*]_j\right)$;
    Update parameters:
    $\boldsymbol{W} \leftarrow \boldsymbol{W} + \varepsilon\left(\boldsymbol{x}_t\boldsymbol{h}_t^{\mathrm{T}} - \boldsymbol{x}_t^*(\boldsymbol{h}_t^*)^{\mathrm{T}}\right)$;
    $\boldsymbol{b} \leftarrow \boldsymbol{b} + \varepsilon\left(\boldsymbol{x}_t - \boldsymbol{x}_t^*\right)$;
    $\boldsymbol{c} \leftarrow \boldsymbol{c} + \varepsilon\left(\boldsymbol{h}_t - \boldsymbol{h}_t^*\right)$;

To train this RBM, we exploit the fact that the interactions between the two layers are defined by an energy function [29]

$$E(\boldsymbol{x}, \boldsymbol{h}|\boldsymbol{\eta}) = -\sum_{j=1}^{q} b_j x_j - \sum_{l=1}^{m} c_l h_l - \sum_{j=1}^{q}\sum_{l=1}^{m} x_j w_{j,l} h_l. \tag{5}$$

The joint probability distribution for visible variables and degrees of freedom outputs is given by a Boltzmann weight, and the parameters $\boldsymbol{\eta}$ can be learned by maximizing the log-likelihood of the RBM over the training set $\boldsymbol{X}$, which is

$$\begin{aligned} L(\boldsymbol{\eta}) &= \frac{1}{n}\sum_{i=1}^{n} \log \sum_{k=1}^{n} P\left(\boldsymbol{x}_i, \boldsymbol{h}_k|\boldsymbol{\eta}\right) \\ &= \frac{1}{n}\sum_{i=1}^{n} \log \frac{\sum_{k=1}^{n} \exp\left(-E\left(\boldsymbol{x}_i, \boldsymbol{h}_k|\boldsymbol{\eta}\right)\right)}{\sum_{i=1}^{n}\sum_{k=1}^{n} \exp\left(-E\left(\boldsymbol{x}_i, \boldsymbol{h}_k|\boldsymbol{\eta}\right)\right)}, \end{aligned} \tag{6}$$

where $P(\boldsymbol{x}, \boldsymbol{h}|\boldsymbol{\eta})$ defines the probabilistic relationship between $\boldsymbol{x}$ and $\boldsymbol{h}$. This is an unsupervised learning, involving the input $\boldsymbol{X}$ only. The algorithm of [29] can be adopted and extended to train this RBM,[1] which is presented in Algorithm 1, where the jth element of vector $\boldsymbol{x}$ is denoted by $[\boldsymbol{x}]_j$.

It is worth pointing out that the CPNN proposed in [10] has a different NN structure. Specifically, in the CPNN, the input to the NN is $\boldsymbol{x}$, the activation functions for the hidden layer and output layer are sigmoid function and exponential function, respectively. The output of the NN is given by

$$P(\boldsymbol{x}; \boldsymbol{\eta}') = \exp\left(c(\boldsymbol{\eta}') + \sigma\left(f(\boldsymbol{x}; \boldsymbol{\eta}')\right)\right). \tag{7}$$

However, unlike our unsupervised RG learning, the training of this NN, i.e., learning of $\boldsymbol{\eta}'$, is in a supervised manner, requiring the true label distributions.

[1] In [29], the visible variables and degrees of freedom outputs are binary variables. But the visible variables and degrees of freedom outputs in our RBM are not binary variables.

3.3. Supervised LDL

Our RG4LDL assumes the parametric conditional probability model $p(y|\boldsymbol{x}; \boldsymbol{\theta})$, and this probability distribution is given by a Boltzmann weight, which is a Boltzmann probability distribution with energy $E(\boldsymbol{x}, \boldsymbol{h}|\boldsymbol{\theta})$. Specifically,

$$p(y_j|\boldsymbol{x}; \boldsymbol{\theta}) = \frac{1}{Z}\exp\left(\sum_{l=1}^{m} \theta_{j,l}\left[\sigma(f(\boldsymbol{x}; \boldsymbol{\eta}))\right]_l\right), \tag{8}$$

where the parameter matrix $\boldsymbol{\theta} = [\theta_{j,l}] \in \mathbb{R}^{\bar{c}\times m}$, $\sigma(f(\boldsymbol{x}; \boldsymbol{\eta})) \in \mathbb{R}^m$ is given by (4), and the normalization term is given by

$$Z = \sum_{j=1}^{\bar{c}} \exp\left(\sum_{l=1}^{m} \theta_{j,l}\left[\sigma(f(\boldsymbol{x}; \boldsymbol{\eta}))\right]_l\right). \tag{9}$$

Plugging (8) into (1), we obtain the objective function:

$$\begin{aligned} J(\boldsymbol{\theta}) = &\sum_{i=1}^{n}\sum_{j=1}^{\bar{c}} d_{\boldsymbol{x}_i}^{y_j}\sum_{l=1}^{m} \theta_{j,l}\left[\sigma\left(f\left(\boldsymbol{x}_i; \boldsymbol{\eta}\right)\right)\right]_l \\ &- \sum_{i=1}^{n} \ln\left(\sum_{j=1}^{\bar{c}} \exp\left(\sum_{l=1}^{m} \theta_{j,l}\left[\sigma\left(f\left(\boldsymbol{x}_i; \boldsymbol{\eta}\right)\right)\right]_l\right)\right). \end{aligned} \tag{10}$$

The goal is to find the optimal parameters $\boldsymbol{\theta}^\star$ by maximizing $J(\boldsymbol{\theta})$. We use the BFGS algorithm [8] to perform this optimization. BFGS approximates the Hessian matrix of objective to achieve faster convergence compared to simple gradient descent, while avoiding expensive computation of the true Hessian.

Specifically, consider the second-order Taylor series expansion of $T(\boldsymbol{\theta}) = -J(\boldsymbol{\theta})$ around the current estimate $\boldsymbol{\theta}^{(l)}$

$$T(\boldsymbol{\theta}^{(l+1)}) \approx T(\boldsymbol{\theta}^{(l)}) + \nabla T(\boldsymbol{\theta}^{(l)})^{\mathrm{T}}\boldsymbol{\Delta} + \frac{1}{2}\boldsymbol{\Delta}^{\mathrm{T}}\boldsymbol{M}(\boldsymbol{\theta}^{(l)})\boldsymbol{\Delta}, \tag{11}$$

where $\boldsymbol{\Delta} = \boldsymbol{\theta}^{(l+1)} - \boldsymbol{\theta}^{(l)}$, $\nabla T(\boldsymbol{\theta}^{(l)})$ is the gradient, and $\boldsymbol{M}(\boldsymbol{\theta}^{(l)})$ is the Hessian. The minimum of (11) is

$$\boldsymbol{\Delta}^{(l)} = -\boldsymbol{M}^{-1}(\boldsymbol{\theta}^{(l)})\nabla T(\boldsymbol{\theta}^{(l)}), \tag{12}$$

which can be obtained through a Newton method based on line search with the search direction $\boldsymbol{d}^{(l)} = \boldsymbol{\Delta}^{(l)}$, and the parameter vector is updated by

$$\boldsymbol{\theta}^{(l+1)} = \boldsymbol{\theta}^{(l)} + \alpha^{(l)}\boldsymbol{d}^{(l)}, \tag{13}$$

where the step size $\alpha^{(l)}$ obtained by line search satisfies the strong Wolfe condition [34]:

$$T(\boldsymbol{\theta}^{(l)} + \alpha^{(l)}\boldsymbol{d}^{(l)}) \le T(\boldsymbol{\theta}^{(l)}) + \rho_1\alpha^{(l)}\nabla T(\boldsymbol{\theta}^{(l)})^{\mathrm{T}}\boldsymbol{d}^{(l)}, \tag{14}$$

$$|\nabla T(\boldsymbol{\theta}^{(l)} + \alpha^{(l)}\boldsymbol{d}^{(l)})^{\mathrm{T}}\boldsymbol{d}^{(l)}| \le \rho_2|\nabla T(\boldsymbol{\theta}^{(l)})^{\mathrm{T}}\boldsymbol{d}^{(l)}|, \tag{15}$$

where $0 < \rho_1 < \rho_2 < 1$.

The above method requires computing the inverse Hessian matrix at each iteration, which is computationally expensive. The BFGS algorithm addresses this problem by iteratively updating the matrix $\boldsymbol{B}$ to approximate $\boldsymbol{M}^{-1}$ [34]

$$\begin{aligned} \boldsymbol{B}^{(l+1)} = &\left(\boldsymbol{I} - \boldsymbol{d}^{(l)}\boldsymbol{s}^{(l)}\left(\boldsymbol{u}^{(l)}\right)^{\mathrm{T}}\right)\boldsymbol{B}^{(l)}\left(\boldsymbol{I} - \boldsymbol{d}^{(l)}\boldsymbol{u}^{(l)}\left(\boldsymbol{s}^{(l)}\right)^{\mathrm{T}}\right) \\ &+ \boldsymbol{d}^{(l)}\boldsymbol{s}^{(l)}\left(\boldsymbol{s}^{(l)}\right)^{\mathrm{T}}, \end{aligned} \tag{16}$$

where $\boldsymbol{s}^{(l)} = \boldsymbol{\theta}^{(l+1)} - \boldsymbol{\theta}^{(l)}$ and $\boldsymbol{u}^{(l)} = \nabla T\left(\boldsymbol{\theta}^{(l+1)}\right) - \nabla T\left(\boldsymbol{\theta}^{(l)}\right)$. To optimize $T(\boldsymbol{\theta})$ using BFGS, the calculations mainly involve the first-order derivative given by

$$\begin{aligned} \frac{\partial T(\boldsymbol{\theta})}{\partial \theta_{j,k}} = &\sum_i \frac{\exp\left(\sum_k \theta_{j,k}\left[\sigma\left(f\left(\boldsymbol{x}_i; \boldsymbol{\eta}\right)\right)\right]_k\right)\left[\sigma\left(f\left(\boldsymbol{x}_i; \boldsymbol{\eta}\right)\right)\right]_k}{\sum_j \exp\left(\sum_k \theta_{j,k}\left[\sigma\left(f\left(\boldsymbol{x}_i; \boldsymbol{\eta}\right)\right)\right]_k\right)} \\ &- \sum_i d_{\boldsymbol{x}_i}^{y_j}\left[\sigma\left(f\left(\boldsymbol{x}_i; \boldsymbol{\eta}\right)\right)\right]_k. \end{aligned} \tag{17}$$

Our proposed RG4LDL is presented in Algorithm 2 and its whole architecture is shown in Fig. 2.

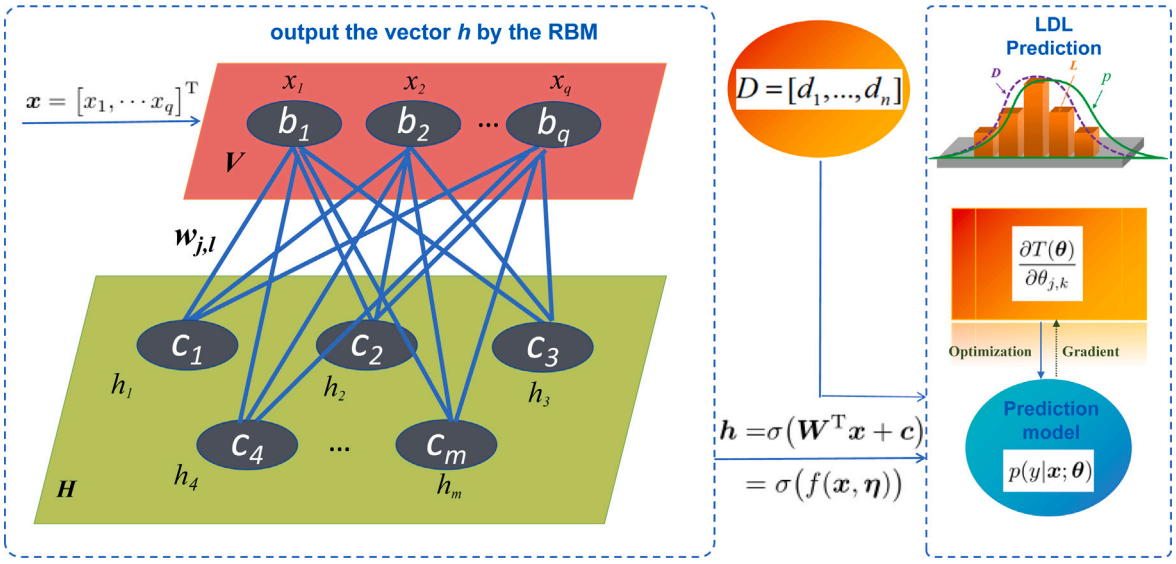


Fig. 2. Flowchart of the proposed RG4LDL. Given the training dataset $S = \{\boldsymbol{x}_i, \boldsymbol{d}_i\}_{i=1}^n$, the unsupervised RG learning transforms the input feature vectors into the degree of freedom outputs, which together with the true labels are used by the supervised LDL to produce the parametric conditional probability model $p(y|\boldsymbol{x};\boldsymbol{\theta})$. In the top-right figure, D illustrates true label distributions, L denotes true logic labels, and p indicates predicted label distributions.

4. Experiment design

We evaluate the proposed RG4LDL framework through extensive experiments conducted on a variety of datasets. The experiments are designed to assess the effectiveness, efficiency, and robustness of RG4LDL compared to existing state-of-the-art LDL methods. The key aspects of the experimental setup includes the choice of datasets, evaluation metrics, and comparison benchmarks.

4.1. Datasets

Our experiments involve 13 real-world datasets and an artificial toy dataset [8]. The 13 real-world datasets cover a variety of domains including yeast biology experiments, facial expression image databases, natural scene recognition, human gene, movie user ratings, etc. The SJAFFE facial expression dataset is collected from facial expression images, and the natural scene dataset is collected from natural scene images and movies, while the datasets of Yeast-alpha to Yeast-spo5 are collected from the records of 9 biological experiments on the budding yeast genes. These datasets are chosen because they all provide the ground-truth label distributions. Table 1 summarizes the features of these datasets, where n is the number of examples, q is the feature dimension, and $\bar{c}$ is the number of class labels.

Table 1
Multilabel datasets with ground-true label distributions from [8] used in experimental evaluation with LDL metrics.

Dataset	Examples (n)	Features (q)	Labels ($\bar{c}$)
Yeast-alpha	2465	24	18
Yeast-cdc	2465	24	15
Yeast-elu	2465	24	14
Yeast-diau	2465	24	7
Yeast-heat	2465	24	6
Yeast-spo	2465	24	6
Yeast-cold	2465	24	4
Yeast-dtt	2465	24	4
Yeast-spo5	2465	24	3
SJAFFE	213	243	6
Natural_Scene	2000	294	9
Human_Gene	30 542	36	68
Movie	7755	1869	5

Algorithm 2: RG4LDL

Data: Training set $S = \{\boldsymbol{x}_i, \boldsymbol{d}_i\}_{i=1}^n$, stop criterion ε;
Result: $p(y|\boldsymbol{x};\boldsymbol{\theta})$;
Initialize $\boldsymbol{\theta}^{(0)}$ and $\boldsymbol{B}^{(0)}$, compute RBM feature matrix $\boldsymbol{H}$ by (4), compute $\nabla T(\boldsymbol{\theta}^{(0)})$ by (17);
$l \leftarrow 0$;
repeat
  Compute search direction $\boldsymbol{d}^{(l)} \leftarrow -\boldsymbol{B}^{(l)}\nabla T(\boldsymbol{\theta}^{(l)})$;
  Compute $\alpha^{(l)}$ by line search to meet (14) and (15);
  $\boldsymbol{\theta}^{(l+1)} \leftarrow \boldsymbol{\theta}^{(l)} + \alpha^{(l)}\boldsymbol{d}^{(l)}$;
  Compute $\nabla T(\boldsymbol{\theta}^{(l+1)})$ by (17);
  Compute $\boldsymbol{B}^{(l+1)}$ by (16);
  $l \leftarrow l + 1$;
until $\left\|\nabla T(\boldsymbol{\theta}^{(l)})\right\| < \varepsilon$;
$p(y_j|\boldsymbol{x};\boldsymbol{\theta}) \leftarrow \frac{1}{Z}\exp\left(\sum_l \theta_{j,l}\left[\sigma(f(\boldsymbol{x};\boldsymbol{\eta}))\right]_l\right)$.

The toy dataset which contains 500 training instances is artificially generated to visualize the mapping from the instance to the label distribution. The instance $\boldsymbol{x}$ is three dimensional and there are three labels. The ground-truth label distribution $\boldsymbol{d} = \left[d_{\boldsymbol{x}}^{y_1}, d_{\boldsymbol{x}}^{y_2}, d_{\boldsymbol{x}}^{y_3}\right]$ of $\boldsymbol{x} = [x_1, x_2, x_3]^{\mathrm{T}}$ is created as

$$t_i = a' x_i + b' x_i^2 + c' x_i^3 + d', \; i = 1, 2, 3, \tag{18}$$

$$\begin{cases} \psi_1 = \left(\bar{\boldsymbol{h}}_1^{\mathrm{T}}\boldsymbol{t}\right)^2, \\ \psi_2 = \left(\bar{\boldsymbol{h}}_2^{\mathrm{T}}\boldsymbol{t} + \beta_1\psi_1\right)^2, \\ \psi_3 = \left(\bar{\boldsymbol{h}}_3^{\mathrm{T}}\boldsymbol{t} + \beta_2\psi_2\right)^2, \end{cases} \tag{19}$$

$$d_{\boldsymbol{x}}^{y_i} = \frac{\psi_i}{\psi_1 + \psi_2 + \psi_3}, \; i = 1, 2, 3, \tag{20}$$

where $\boldsymbol{t} = \left[t_1, t_2, t_3\right]^{\mathrm{T}}$, $x_i \in [-1, 1]$, $a' = 1$, $b' = 0.5$, $c' = 0.2$ and $d' = 1$, while $\bar{\boldsymbol{h}}_1 = [4, 2, 1]^{\mathrm{T}}$, $\bar{\boldsymbol{h}}_2 = [1, 2, 4]^{\mathrm{T}}$, $\bar{\boldsymbol{h}}_3 = [1, 4, 2]^{\mathrm{T}}$ and $\beta_1 = \beta_2 = 0.01$. To visualize the results of LDL algorithms, the test examples are selected from a certain manifold in the feature space. The first two components of $\boldsymbol{x}$, x_1 and x_2, are located at a grid of the interval 0.01 within the range $[-1, 1]$, and there are a total of $201 \times 201 = 40,401$ instances. The

third component x_3 is calculated by

$$x_3 = \sin\big(\pi(x_1 + x_2)\big). \tag{21}$$

Then the label distribution of each test instance is transformed into a color. Thus the ground-truth or predicted label distributions of the test instances can be compared visually through the color pattern on the manifold.

4.2. Evaluation measures

Six different metrics are used to assess the predictive performance of LDL algorithms, and they are: Chebyshev distance (Cheb), Clark distance (Clark), Canberra metric (Canber), Kullback–Leibler divergence (KL), cosine coefficient (Cosine) and intersection similarity (Intersec) [8]. For the first four metrics, which are distance-based metrics, lower values mean better predictive accuracy, and we use '↓' to indicate this. For the last two metrics, which are similarity-based metrics, higher values mean better performance, and we use '↑' to indicate this.

We now elaborate why choosing these six metrics. Cheb metric measures the largest local difference between distributions by capturing the maximum deviation, making it particularly suitable for the scenarios that emphasize extreme errors. Clark metric is based on the weighted sum of squared relative differences, effectively accounting for the overall deviation of the two distributions. Canber metric, with its careful weighting of the absolute differences across dimensions, sensitively reflects subtle shifts in the distributions. KL metric quantifies the information loss between the predicted and true distributions, making it especially relevant for the scenarios involving probabilistic approximations. Cosine metric evaluates the similarity of distribution shapes by measuring the consistency of vector directions, thereby reducing the impact of differences in magnitude. Intersec metric focuses on the overlapping areas of distributions, directly reflecting the degree of shared information retention. Together, these six metrics provide an evaluation framework that covers multiple perspectives, including sensitivity to extreme values, global bias, local details, differences in information entropy, shape similarity, and overlap measurement. This framework not only highlights the distinct differences across various dimensions in label distribution learning but also aligns with diverse optimization objectives in real-world applications. As such, it establishes a multi-scale and multi-perspective evaluation system for effectively quantifying the degree of similarity or approximation between predicted and true label distributions.

Additionally, the runtime is also used to measure the complexity. Obviously, the runtime ↓ is a metric that is the smaller the better.

4.3. Comparison benchmarks

The proposed RG4LDL is compared with 8 existing state-of-the-art LDL methods, and they are AA-BP [8], BFGS-LDL [8], CPNN [10], IIS-LDL [10], LDL-LRR [24], LDLLC [35], LDLSF [22], and LDL-LCLR [23]. For AA-BP, a hidden layer with 60 neurons and 6000 iterations is set for NN training. For BFGS-LDL, the maximum number of iterations is 300 to guarantee convergence. For CPNN, the number of training iterations is 100. For IIS-LDL, the maximum number of iterations for training is 50. The last four more recent competing LDL algorithms are known to provide state-of-the-art performance, and their parameters are tuned following the recommendations in their original papers to ensure their optimal performance, with 100 to 400 iterations in training. For our RG4LDL, the numbers of hidden units are selected from $\{10, 20, 50, 100, 200, 500, 800\}$ and the maximum number of iterations is 10.

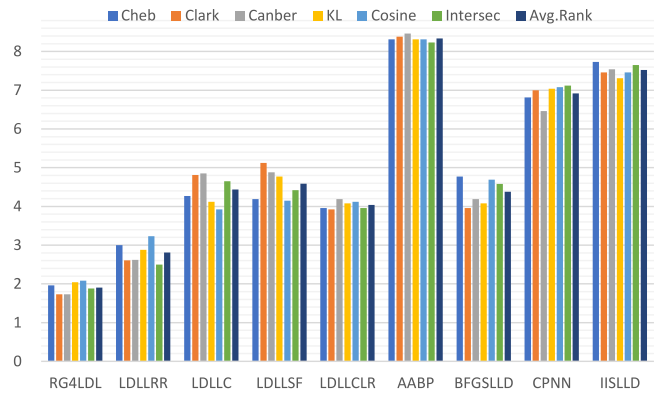


Fig. 3. Average ranking on predictive performance of each algorithm across all 13 datasets for each evaluation metric as well as average ranking over all the evaluation metrics, where the smaller the value the higher the ranking.

5. Experimental results

This section presents the experimental results, including comparative LDL predictive performance analysis, an ablation study of the proposed RG4LDL, and statistical validation of the experimental results as well as label distribution visualization. For each algorithm, we perform 10 runs of 5-fold cross-validation on each dataset. Specifically, we randomly split the dataset into 5 folds, using one fold as the test set and the rest as the training set, repeated for 5 folds. This process is repeated 10 times. All the experiments are carried out on Matlab 2019b, running on a PC with i5-12400 2.50 GHz processor of 12th Gen Intel(R) Core(TM).

5.1. Comparison of LDL predictive performance

For the 13 real-world datasets, the results obtained are shown in Tables 2 to 7, which are presented in the format of mean ±std (rank) for each dataset and each metric. The ranking achieved by each algorithm for each dataset and each evaluation metric is included in the brackets, where the smaller the value the higher the ranking, and the boldface value indicates the best performance. The results of Tables 2 to 7 clearly show that our RG4LDL overall outperforms the other 8 benchmark LDL algorithms, in terms of LDL predictive accuracy. To visually show that overall our RG4LDL outperforms the other 8 benchmarks, we aggregate the average ranking of each algorithm across all the 13 datasets for each evaluation metric as well as the average ranking across all the six metrics. The results obtained are depicted in Fig. 3, where it can be seen that our RG4LDL has the best average ranking across all the 6 metrics.

In the runtime statistics of Table 8, it can be seen that our RG4LDL is extremely time-efficient. The average runtime of our algorithm over 13 datasets is only 1.791 s, and it is more than 10 times faster than the second-best performing LDL-LRR. Experimental results therefore convincingly demonstrate that our RG4LDL is significantly faster than the existing state-of-the-arts LDL methods and offers superior LDL predictive accuracy over these existing methods.

5.2. Ablation study

To evaluate the contribution of each component of our RG4LDL, we conduct an ablation study to assess the impact of removing or isolating individual components on the overall performance of the framework. The results obtained are given in Table 9. The complete RG4LD (the top row), which combines unsupervised RG learning and supervised LDL, is the full RG4LDL framework. RG4LDL without RBM (the middle row), i.e., the RG learning step is removed, is directly applied to the

Table 2
Label distribution recovery performance measured by Chebyshev distance ↓.

Dataset	RG4LDL	LDL-LRR	LDLLC	LDLSF	LDL-LCLR	AA-BP	BFGS-LDL	CPNN	IIS-LDL
Yeast-alpha	**0.0134 ± 0.0003(3)**	**0.0134 ± 0.0001(3)**	0.0135 ± 0.0002(6)	**0.0134 ± 0.0001(3)**	**0.0134 ± 0.0002(3)**	0.0361 ± 0.019(9)	**0.0134 ± 0.0064(3)**	0.0144 ± 0.0065(7)	0.0169 ± 0.0072(8)
Yeast-cdc	**0.0162 ± 0.0004(3)**	**0.0162 ± 0.0002(3)**	0.0163 ± 0.0004(6)	**0.0162 ± 0.0005(3)**	**0.0162 ± 0.0002(3)**	0.0371 ± 0.0199(9)	**0.0162 ± 0.0082(3)**	0.0176 ± 0.0083(7)	0.0200 ± 0.0090(8)
Yeast-elu	0.0162 ± 0.0001(2)	**0.0161 ± 0.0004(1)**	0.0163 ± 0.0163(4.5)	0.0163 ± 0.0001(4.5)	0.0163 ± 0.0002(4.5)	0.0370 ± 0.0197(9)	0.0163 ± 0.0075(4.5)	0.0170 ± 0.0075(7)	0.0202 ± 0.0086(8)
Yeast-diau	**0.0369 ± 0.0006(1)**	0.0371 ± 0.0009(6)	0.0370 ± 0.0008(3.5)	0.0370 ± 0.0006(3.5)	0.0370 ± 0.0008(3.5)	0.0482 ± 0.0236(9)	0.0370 ± 0.0181(3.5)	0.0401 ± 0.0186(7)	0.0412 ± 0.0187(8)
Yeast-heat	**0.0418 ± 0.0006(1)**	0.0421 ± 0.0007(2)	0.0423 ± 0.0007(4.5)	0.0423 ± 0.0004(4.5)	0.0423 ± 0.001(4.5)	0.0515 ± 0.0241(9)	0.0423 ± 0.0198(4.5)	0.0425 ± 0.0199(7)	0.0465 ± 0.0202(8)
Yeast-spo	**0.0583 ± 0.0020(2)**	0.0585 ± 0.0011(6)	**0.0583 ± 0.0013(2)**	0.0584 ± 0.0012(4.5)	**0.0583 ± 0.0008(2)**	0.0662 ± 0.0333(9)	0.0584 ± 0.0312(4.5)	0.0595 ± 0.0324(7)	0.0617 ± 0.0315(8)
Yeast-cold	0.0510 ± 0.0008(2)	**0.0509 ± 0.0010(1)**	0.0511 ± 0.0011(3.5)	0.0512 ± 0.0011(5.5)	0.0511 ± 0.0016(3.5)	0.0573 ± 0.0312(9)	0.0512 ± 0.0280(5.5)	0.0523 ± 0.0286(7)	0.0567 ± 0.0300(8)
Yeast-dtt	0.0359 ± 0.0006(2)	**0.0358 ± 0.0008(1)**	0.0361 ± 0.0009(5)	0.0361 ± 0.0014(5)	0.0360 ± 0.0010(3)	0.0439 ± 0.0257(9)	0.0361 ± 0.0212(5)	0.0368 ± 0.0215(7)	0.0433 ± 0.0243(8)
Yeast-spo5	**0.0910 ± 0.0020(1)**	0.0913 ± 0.0008(3)	0.0913 ± 0.0021(3)	0.0914 ± 0.0008(5.5)	0.0913 ± 0.0011(3)	0.0936 ± 0.0571(8)	0.0915 ± 0.0566(7)	0.0914 ± 0.0563(5.5)	0.0946 ± 0.0579(9)
SJAFFE	**0.0859 ± 0.0057(1)**	0.0882 ± 0.0039(3)	0.1196 ± 0.0075(8)	0.0871 ± 0.0069(2)	0.1163 ± 0.0053(6)	0.1352 ± 0.0603(9)	0.1043 ± 0.0552(4)	0.1105 ± 0.0476(5)	0.1170 ± 0.0567(7)
Natural_Scene	0.2976 ± 0.0036(2)	**0.2949 ± 0.0038(1)**	0.3097 ± 0.0096(3)	0.3207 ± 0.0060(6)	0.3190 ± 0.0072(5)	0.3375 ± 0.2090(7)	0.3220 ± 0.0170(9)	0.3148 ± 0.2160(4)	0.3413 ± 0.2238(8)
Human_Gene	**0.0533 ± 0.0009(2.5)**	0.0536 ± 0.0012(7)	0.0534 ± 0.0012(5.5)	**0.0533 ± 0.001(2.5)**	**0.0533 ± 0.0013(2.5)**	0.0600 ± 0.0754(8)	**0.0533 ± 0.0779(2.5)**	0.3148 ± 0.0780(9)	0.0534 ± 0.0779(5.5)
Movie	0.1153 ± 0.0010(3)	0.1139 ± 0.0015(2)	**0.1136 ± 0.0015(1)**	0.1264 ± 0.0026(5)	0.1327 ± 0.0022(8)	0.1228 ± 0.0704(4)	0.1292 ± 0.0750(6)	0.1340 ± 0.0764(9)	0.1311 ± 0.0721(7)
Average rank	**1.9615(1)**	3.0000(2)	4.2692(5)	4.1923(4)	3.9615(3)	8.3077(9)	4.7692(6)	6.8077(7)	7.7308(8)

Table 3
Label distribution recovery performance measured by Clark distance ↓.

Dataset	RG4LDL	LDL-LRR	LDLLC	LDLSF	LDL-LCLR	AA-BP	BFGS-LDL	CPNN	IIS-LDL
Yeast-alpha	**0.2096 ± 0.0029(1)**	0.2097 ± 0.0027(2)	0.2114 ± 0.0034(6)	0.2102 ± 0.0024(4.5)	0.2102 ± 0.0034(4.5)	0.7169 ± 0.4157(9)	0.2101 ± 0.0823(3)	0.2273 ± 0.0819(7)	0.2605 ± 0.0866(8)
Yeast-cdc	0.2158 ± 0.0045(2)	**0.2151 ± 0.0021(1)**	0.2165 ± 0.0041(6)	0.2162 ± 0.0053(5)	0.2160 ± 0.0022(3.5)	0.5670 ± 0.3341(9)	0.2160 ± 0.0987(3.5)	0.2358 ± 0.1014(7)	0.2589 ± 0.1008(8)
Yeast-elu	0.1986 ± 0.0012(2)	**0.1975 ± 0.0040(1)**	0.1998 ± 0.1998(6)	0.1992 ± 0.0017(4.5)	0.1992 ± 0.0012(4.5)	0.5109 ± 0.3022(9)	0.1991 ± 0.0778(3)	0.2091 ± 0.0788(7)	0.2406 ± 0.0830(8)
Yeast-diau	**0.2005 ± 0.0032(1)**	0.2016 ± 0.0055(6)	0.2010 ± 0.0042(4)	0.2010 ± 0.0021(4)	0.2010 ± 0.0038(4)	0.2650 ± 0.1355(9)	0.2009 ± 0.1033(2)	0.2161 ± 0.1049(7)	0.2223 ± 0.1031(8)
Yeast-heat	**0.1810 ± 0.0034(1)**	0.1821 ± 0.0026(2)	0.1830 ± 0.0032(6)	0.1827 ± 0.0007(3)	0.1828 ± 0.0041(4.5)	0.2258 ± 0.1063(9)	0.1828 ± 0.0830(4.5)	0.1842 ± 0.0836(7)	0.2005 ± 0.0842(8)
Yeast-spo	0.2498 ± 0.0086(3)	0.2504 ± 0.0042(6)	**0.2497 ± 0.0054(1.5)**	0.2500 ± 0.0067(5)	**0.2497 ± 0.0038(1.5)**	0.2884 ± 0.1405(9)	0.2499 ± 0.1265(4)	0.2541 ± 0.1280(7)	0.2637 ± 0.1257(8)
Yeast-cold	0.1395 ± 0.0023(3.5)	**0.1385 ± 0.0036(1)**	0.1394 ± 0.0029(2)	0.1397 ± 0.0029(5.5)	0.1395 ± 0.0044(3.5)	0.1563 ± 0.0874(9)	0.1397 ± 0.0793(5.5)	0.1424 ± 0.0804(7)	0.1532 ± 0.0825(8)
Yeast-dtt	0.0979 ± 0.0017(2)	**0.0976 ± 0.0024(1)**	0.0984 ± 0.0026(6)	0.0983 ± 0.0040(4.5)	0.0982 ± 0.0033(3)	0.1196 ± 0.0734(9)	0.0983 ± 0.0618(4.5)	0.1001 ± 0.0626(7)	0.1163 ± 0.0674(8)
Yeast-spo5	**0.1836 ± 0.0048(1)**	0.1838 ± 0.0018(2)	0.1840 ± 0.0050(3)	0.1843 ± 0.0016(5.5)	0.1841 ± 0.0029(4)	0.1885 ± 0.1238(8)	0.1843 ± 0.1234(5.5)	0.1845 ± 0.1230(7)	0.1902 ± 0.1243(9)
SJAFFE	**0.3278 ± 0.0108(1)**	0.3296 ± 0.0118(2)	0.4258 ± 0.0158(8)	0.3638 ± 0.0261(3)	0.4143 ± 0.0184(6)	0.5545 ± 0.2210(9)	0.4082 ± 0.1575(5)	0.3983 ± 0.1194(4)	0.4169 ± 0.1175(7)
Natural_Scene	**2.3791 ± 0.0086(1)**	2.4297 ± 0.0109(4)	2.4501 ± 0.0137(5)	2.4745 ± 0.0090(9)	2.4286 ± 0.0063(3)	2.4603 ± 0.2990(6)	2.4110 ± 0.0230(2)	2.4679 ± 0.3021(8)	2.4606 ± 0.3099(7)
Human_Gene	**2.1060 ± 0.0212(1)**	2.1129 ± 0.0212(5)	2.1196 ± 0.0181(7)	2.1119 ± 0.0131(4)	2.1087 ± 0.0196(2)	3.3666 ± 0.8462(9)	2.1092 ± 1.2453(3)	2.4679 ± 1.2453(8)	2.1166 ± 1.2442(6)
Movie	0.5200 ± 0.0041(3)	**0.5117 ± 0.0041(1)**	0.5123 ± 0.0044(2)	0.6154 ± 0.0104(9)	0.5787 ± 0.0098(7)	0.5591 ± 0.2700(5)	0.5604 ± 0.2874(6)	0.5795 ± 0.2946(8)	0.5585 ± 0.2751(4)
Average rank	**1.7308(1)**	2.6154(2)	4.8077(5)	5.1154(6)	3.9231(3)	8.3846(9)	3.9615(4)	7.0000(7)	7.4615(8)

Table 4
Label distribution recovery performance measured by Canberra metric ↓.

Dataset	RG4LDL	LDL-LRR	LDLLC	LDLSF	LDL-LCLR	AA-BP	BFGS-LDL	CPNN	IIS-LDL
Yeast-alpha	0.6806 ± 0.0097(2)	**0.6800 ± 0.0074(1)**	0.6868 ± 0.0110(6)	0.6825 ± 0.0080(4)	0.6828 ± 0.0101(5)	2.3640 ± 1.3499(9)	0.6823 ± 0.2602(3)	0.7409 ± 0.2586(7)	0.8595 ± 0.2849(8)
Yeast-cdc	0.6469 ± 0.0137(2)	**0.6446 ± 0.0095(1)**	0.6492 ± 0.0103(6)	0.6486 ± 0.0143(5)	0.6478 ± 0.0082(3.5)	1.7160 ± 0.9886(9)	0.6478 ± 0.2719(3.5)	0.7134 ± 0.2878(7)	0.7869 ± 0.2889(8)
Yeast-elu	0.5821 ± 0.0051(2)	**0.5784 ± 0.0117(1)**	0.5862 ± 0.5862(6)	0.5833 ± 0.0042(3)	0.5835 ± 0.0052(5)	1.5030 ± 0.8639(9)	0.5834 ± 0.2128(4)	0.6157 ± 0.2176(7)	0.7148 ± 0.2368(8)
Yeast-diau	**0.4304 ± 0.0058(1)**	0.4331 ± 0.0117(6)	0.4316 ± 0.0087(5)	0.4314 ± 0.0038(4)	0.4313 ± 0.0085(3)	0.5710 ± 0.2854(9)	0.4311 ± 0.2149(2)	0.4654 ± 0.2223(7)	0.4808 ± 0.2169(8)
Yeast-heat	**0.3610 ± 0.0053(1)**	0.3633 ± 0.0045(2)	0.3650 ± 0.0061(6)	0.3643 ± 0.0021(3)	0.3645 ± 0.0079(4.5)	0.4540 ± 0.2089(9)	0.3645 ± 0.1579(4.5)	0.3681 ± 0.1597(7)	0.4035 ± 0.1632(8)
Yeast-spo	0.5132 ± 0.0178(2)	0.5140 ± 0.0074(6)	**0.5131 ± 0.0110(1)**	0.5138 ± 0.0124(5)	0.5133 ± 0.0085(3)	0.5910 ± 0.2863(9)	0.5136 ± 0.2585(4)	0.5236 ± 0.2626(7)	0.5417 ± 0.2562(8)
Yeast-cold	0.2401 ± 0.0037(3.5)	**0.2386 ± 0.0057(1)**	0.2399 ± 0.0051(2)	0.2404 ± 0.0058(5)	0.2401 ± 0.0071(3.5)	0.2693 ± 0.1497(9)	0.2406 ± 0.1355(6)	0.2457 ± 0.1382(7)	0.2647 ± 0.1429(8)
Yeast-dtt	**0.1668 ± 0.0032(1)**	0.1678 ± 0.0037(2)	0.1692 ± 0.0043(6)	0.1691 ± 0.0066(4.5)	0.1690 ± 0.0056(3)	0.2060 ± 0.1225(9)	0.1691 ± 0.1014(4.5)	0.1720 ± 0.1026(7)	0.2011 ± 0.1131(8)
Yeast-spo5	**0.2819 ± 0.0067(1)**	0.2824 ± 0.0026(2)	0.2826 ± 0.0072(3)	0.2831 ± 0.0024(6)	0.2828 ± 0.0041(4)	0.2895 ± 0.1846(8)	0.2831 ± 0.1837(6)	0.2831 ± 0.1828(6)	0.2923 ± 0.1863(9)
SJAFFE	**0.6668 ± 0.0215(1)**	0.6775 ± 0.0271(2)	0.8886 ± 0.0368(8)	0.7214 ± 0.0517(3)	0.8644 ± 0.0394(6)	1.1329 ± 0.4539(9)	0.8360 ± 0.3320(5)	0.8310 ± 0.2642(4)	0.8698 ± 0.2744(7)
Natural_Scene	**6.5057 ± 0.0183(1)**	6.6337 ± 0.0420(3)	6.7152 ± 0.0555(5)	6.7756 ± 0.0414(7)	6.6686 ± 0.0372(4)	6.7783 ± 1.2586(8)	6.6200 ± 0.0970(2)	6.8549 ± 1.2275(9)	6.7649 ± 1.3250(6)
Human_Gene	14.4166 ± 0.1696(2)	14.4701 ± 0.1815(6)	14.5114 ± 0.1390(8)	14.4508 ± 0.1050(5)	14.4345 ± 0.1463(3)	22.9427 ± 7.0070(9)	14.4388 ± 9.8155(4)	**6.8549 ± 9.8147(1)**	14.4915 ± 9.8117(7)
Movie	0.9954 ± 0.0089(3)	0.9805 ± 0.0085(2)	**0.9795 ± 0.0098(1)**	1.1516 ± 0.0236(9)	1.1161 ± 0.0220(7)	1.0622 ± 0.5492(4)	1.0805 ± 0.5968(6)	1.1187 ± 0.6138(8)	1.0730 ± 0.5676(5)
Average rank	**1.7301(1)**	2.6154(2)	4.8462(5)	4.8846(6)	4.1923(3.5)	8.4615(9)	4.1923(3.5)	6.4615(7)	7.5385(8)

Table 5
Label distribution recovery performance measured by Kullback–Leibler divergence ↓.

Dataset	RG4LDL	LDL-LRR	LDLLC	LDLSF	LDL-LCLR	AA-BP	BFGS-LDL	CPNN	IIS-LDL
Yeast-alpha	0.0055 ± 0.0002(3.5)	**0.0054 ± 0.0002(1)**	0.0056 ± 0.0002(6)	0.0055 ± 0.0001(3.5)	0.0055 ± 0.0002(3.5)	0.0819 ± 0.1032(9)	0.0055 ± 0.0043(3.5)	0.0064 ± 0.0046(7)	0.0084 ± 0.0055(8)
Yeast-cdc	0.0070 ± 0.0003(4)	**0.0069 ± 0.0002(1)**	0.0070 ± 0.0003(4)	0.0070 ± 0.0003(4)	0.0070 ± 0.0001(4)	0.0603 ± 0.0841(9)	0.0070 ± 0.0062(4)	0.0085 ± 0.0069(7)	0.0099 ± 0.0073(8)
Yeast-elu	0.0062 ± 0.0001(4)	**0.0061 ± 0.0003(1)**	0.0062 ± 0.0062(4)	0.0062 ± 0.0001(4)	0.0062 ± 0.0001(4)	0.0526 ± 0.0763(9)	0.0062 ± 0.0048(4)	0.0068 ± 0.0051(7)	0.0091 ± 0.0062(8)
Yeast-diau	**0.0131 ± 0.0004(2)**	0.0133 ± 0.0007(6)	**0.0131 ± 0.0005(2)**	0.0132 ± 0.0002(4.5)	0.0132 ± 0.0004(4.5)	0.0242 ± 0.0334(9)	**0.0131 ± 0.0133(2)**	0.0152 ± 0.0152(7)	0.0159 ± 0.0144(8)
Yeast-heat	**0.0124 ± 0.0004(1)**	0.0126 ± 0.0003(3)	0.0127 ± 0.0004(5.5)	0.0126 ± 0.0001(3)	0.0126 ± 0.0006(3)	0.0203 ± 0.0264(9)	0.0127 ± 0.0114(5.5)	0.0129 ± 0.0116(7)	0.0151 ± 0.0124(8)
Yeast-spo	**0.0246 ± 0.0014(2)**	0.0247 ± 0.0009(6)	**0.0246 ± 0.0009(2)**	**0.0246 ± 0.0011(2)**	**0.0246 ± 0.0008(2)**	0.0332 ± 0.0360(9)	**0.0246 ± 0.0240(2)**	0.0257 ± 0.0251(7)	0.0271 ± 0.0250(8)
Yeast-cold	**0.0120 ± 0.0006(1)**	0.0122 ± 0.0005(4)	0.0122 ± 0.0006(4)	0.0122 ± 0.0006(4)	0.0122 ± 0.0010(4)	0.0154 ± 0.0207(9)	0.0122 ± 0.0154(4)	0.0127 ± 0.0158(7)	0.0146 ± 0.0170(8)
Yeast-dtt	**0.0061 ± 0.0003(1)**	0.0062 ± 0.0005(2)	0.0063 ± 0.0005(4.5)	0.0063 ± 0.0006(4.5)	0.0063 ± 0.0005(4.5)	0.0095 ± 0.0154(9)	0.0063 ± 0.0101(4.5)	0.0065 ± 0.0102(7)	0.0087 ± 0.0120(8)
Yeast-spo5	**0.0291 ± 0.0017(1)**	0.0292 ± 0.0006(2.5)	0.0292 ± 0.0014(2.5)	0.0294 ± 0.0005(6.5)	0.0293 ± 0.0013(4.5)	0.0305 ± 0.0361(8)	0.0293 ± 0.0351(4.5)	0.0294 ± 0.0352(6.5)	0.0312 ± 0.0368(9)
SJAFFE	**0.0427 ± 0.0052(1)**	0.0435 ± 0.0024(2)	0.0735 ± 0.0064(8)	0.0486 ± 0.0080(3)	0.0682 ± 0.0047(6)	0.1345 ± 0.1304(9)	0.0667 ± 0.0560(5)	0.0629 ± 0.0386(4)	0.0693 ± 0.0440(7)
Natural_Scene	0.7404 ± 0.0142(2)	**0.7083 ± 0.0155(1)**	0.7609 ± 0.0230(3)	1.1981 ± 0.0718(9)	0.8199 ± 0.0342(4)	0.8901 ± 0.4918(7)	0.8540 ± 0.0620(5)	0.9194 ± 0.6027(8)	0.8697 ± 0.4453(6)
Human_Gene	**0.2355 ± 0.0061(1)**	0.2372 ± 0.0066(6)	0.2374 ± 0.0053(7)	0.2370 ± 0.0037(5)	0.2359 ± 0.0049(2)	0.5058 ± 0.4012(8)	0.2360 ± 0.3623(3)	0.9194 ± 0.3651(9)	0.2367 ± 0.3627(4)
Movie	0.0986 ± 0.0019(3)	0.0963 ± 0.0028(2)	**0.0958 ± 0.0029(1)**	0.2111 ± 0.0132(9)	0.1339 ± 0.0052(7)	0.1112 ± 0.1217(4)	0.1241 ± 0.1512(6)	0.1356 ± 0.1669(8)	0.1206 ± 0.1395(5)
Average rank	**2.0384(1)**	2.8846(2)	4.1154(5)	4.7692(6)	4.0769(3.5)	8.3077(9)	4.0769(3.5)	7.0385(7)	7.3077(8)

Table 6
Label distribution recovery performance measured by Cosine coefficient ↑.

Dataset	RG4LDL	LDL-LRR	LDLLC	LDLSF	LDL-LCLR	AA-BP	BFGS-LDL	CPNN	IIS-LDL
Yeast-alpha	**0.9946 ± 0.0002(3)**	**0.9946 ± 0.0001(3)**	0.9945 ± 0.0002(6)	**0.9946 ± 0.0001(3)**	**0.9946 ± 0.0002(3)**	0.9494 ± 0.0457(9)	**0.9946 ± 0.0041(3)**	0.9937 ± 0.0044(7)	0.9915 ± 0.0055(8)
Yeast-cdc	**0.9934 ± 0.0003(1)**	0.9934 ± 0.0002(4)	0.9933 ± 0.0002(4)	0.9933 ± 0.0003(4)	0.9933 ± 0.0001(4)	0.9595 ± 0.0391(9)	0.9933 ± 0.0055(4)	0.9918 ± 0.0062(7)	0.9902 ± 0.0070(8)
Yeast-elu	**0.9941 ± 0.0001(1.5)**	**0.9941 ± 0.0002(1.5)**	0.9940 ± 0.9940(4.5)	0.9940 ± 0.0001(4.5)	0.9940 ± 0.0001(4.5)	0.9637 ± 0.0358(9)	0.9940 ± 0.0043(4.5)	0.9934 ± 0.0046(7)	0.9909 ± 0.0059(8)
Yeast-diau	**0.9879 ± 0.0003(3)**	0.9878 ± 0.0007(6)	**0.9879 ± 0.0005(3)**	**0.9879 ± 0.0002(3)**	0.9879 ± 0.0004(3)	0.9791 ± 0.0204(9)	**0.9879 ± 0.0113(3)**	0.9860 ± 0.0123(7)	0.9851 ± 0.0126(8)
Yeast-heat	**0.9882 ± 0.0003(1)**	0.9880 ± 0.0003(4)	0.9880 ± 0.0004(4)	0.9880 ± 0.0002(4)	0.9880 ± 0.0006(4)	0.9816 ± 0.0169(9)	0.9880 ± 0.0100(4)	0.9878 ± 0.0102(7)	0.9854 ± 0.0113(8)
Yeast-spo	0.9769 ± 0.0013(4.5)	0.9769 ± 0.0008(4.5)	**0.9770 ± 0.0008(1.5)**	0.9769 ± 0.0009(4.5)	0.9770 ± 0.0007(2)	0.9701 ± 0.0268(9)	0.9769 ± 0.0208(4.5)	0.9759 ± 0.0219(7)	0.9744 ± 0.0220(8)
Yeast-cold	0.9886 ± 0.0004(3)	**0.9887 ± 0.0005(1)**	0.9886 ± 0.0005(3)	0.9885 ± 0.0005(5.5)	0.9886 ± 0.0008(3)	0.9857 ± 0.0160(9)	0.9885 ± 0.0128(5.5)	0.9880 ± 0.0133(7)	0.9861 ± 0.0146(8)
Yeast-dtt	**0.9942 ± 0.0002(1)**	0.9941 ± 0.0003(4)	0.9941 ± 0.0003(4)	0.9941 ± 0.0005(4)	0.9941 ± 0.0004(4)	0.9912 ± 0.0112(9)	0.9941 ± 0.0077(4)	0.9939 ± 0.0079(7)	0.9916 ± 0.0098(8)
Yeast-spo5	**0.9742 ± 0.0013(2)**	**0.9742 ± 0.0005(2)**	**0.9742 ± 0.0010(2)**	0.9740 ± 0.0004(7)	0.9741 ± 0.0008(5)	0.9730 ± 0.0288(8)	0.9741 ± 0.0279(5)	0.9741 ± 0.0278(5)	0.9723 ± 0.0298(9)
SJAFFE	**0.9603 ± 0.0061(1)**	0.9590 ± 0.0025(3)	0.9308 ± 0.0061(8)	0.9593 ± 0.0057(2)	0.9355 ± 0.0042(6)	0.9016 ± 0.0692(9)	0.9404 ± 0.0522(4)	0.9400 ± 0.0368(5)	0.9345 ± 0.0428(7)
Natural_Scene	0.7479 ± 0.0045(2)	**0.7614 ± 0.0061(1)**	0.7393 ± 0.0088(3)	0.7241 ± 0.0046(4)	0.7193 ± 0.0103(5)	0.6928 ± 0.1749(7)	0.7100 ± 0.0170(9)	0.6774 ± 0.2008(8)	0.6976 ± 0.1567(6)
Human_Gene	**0.8351 ± 0.0030(1)**	0.8341 ± 0.0037(6)	0.8336 ± 0.0025(7)	0.8347 ± 0.0025(3.5)	0.8348 ± 0.0022(2)	0.7219 ± 0.1642(8)	0.8347 ± 0.1768(3.5)	0.6774 ± 0.1778(9)	0.8342 ± 0.1767(5)
Movie	0.9351 ± 0.0013(3)	0.9365 ± 0.0019(2)	**0.9369 ± 0.0018(1)**	0.9227 ± 0.0034(5)	0.9161 ± 0.0031(8)	0.9280 ± 0.0758(4)	0.9206 ± 0.0884(7)	0.9119 ± 0.0968(9)	0.9212 ± 0.0831(6)
Average rank	**2.0769(1)**	3.2308(2)	3.9231(3)	4.1538(5)	4.1154(4)	8.3077(9)	4.6923(6)	7.0769(7)	7.4615(8)

Table 7
Label distribution recovery performance measured by Intersec↑.

Dataset	RG4LDL	LDL-LRR	LDLLC	LDLSF	LDL-LCLR	AA-BP	BFGS-LDL	CPNN	IIS-LDL
Yeast-alpha	0.9624 ± 0.0005(2)	**0.9625 ± 0.0004(1)**	0.9621 ± 0.0006(6)	0.9623 ± 0.0004(4)	0.9623 ± 0.0005(4)	0.8767 ± 0.0652(9)	0.9623 ± 0.0142(4)	0.9591 ± 0.0142(7)	0.9520 ± 0.0160(8)
Yeast-cdc	0.9574 ± 0.0009(3)	**0.9576 ± 0.0007(1)**	0.9573 ± 0.0007(5.5)	0.9573 ± 0.0009(5.5)	0.9574 ± 0.0006(3)	0.8915 ± 0.0581(9)	0.9574 ± 0.0175(3)	0.9530 ± 0.0186(7)	0.9476 ± 0.0190(8)
Yeast-elu	0.9589 ± 0.0004(2.5)	**0.9592 ± 0.0008(1)**	0.9586 ± 0.9586(6)	0.9589 ± 0.0003(2.5)	0.9588 ± 0.0004(4.5)	0.8975 ± 0.0546(9)	0.9588 ± 0.0147(4.5)	0.9565 ± 0.0151(7)	0.9489 ± 0.0169(8)
Yeast-diau	**0.9403 ± 0.0008(1.5)**	0.9400 ± 0.0016(6)	0.9402 ± 0.0012(4)	0.9402 ± 0.0006(4)	0.9402 ± 0.0012(4)	0.9210 ± 0.0379(9)	**0.9403 ± 0.0287(1.5)**	0.9353 ± 0.0299(7)	0.9328 ± 0.0293(8)
Yeast-heat	**0.9408 ± 0.0008(1)**	0.9404 ± 0.0008(2)	0.9401 ± 0.0009(6)	0.9402 ± 0.0005(4)	0.9402 ± 0.0013(4)	0.9256 ± 0.0331(9)	0.9402 ± 0.0252(4)	0.9396 ± 0.0256(7)	0.9333 ± 0.0265(8)
Yeast-spo	**0.9155 ± 0.0029(2.5)**	0.9154 ± 0.0012(5.5)	**0.9155 ± 0.0018(2.5)**	0.9154 ± 0.0018(5.5)	**0.9155 ± 0.0014(2.5)**	0.9031 ± 0.0455(9)	**0.9155 ± 0.0413(2.5)**	0.9138 ± 0.0422(7)	0.9105 ± 0.0413(8)
Yeast-cold	0.9409 ± 0.0008(3)	**0.9412 ± 0.0013(1)**	0.9409 ± 0.0012(3)	0.9408 ± 0.0015(5)	0.9409 ± 0.0017(3)	0.9336 ± 0.0358(9)	0.9407 ± 0.0323(6)	0.9394 ± 0.0331(7)	0.9345 ± 0.0346(8)
Yeast-dtt	0.9584 ± 0.0008(2)	**0.9586 ± 0.0009(1)**	0.9583 ± 0.0010(4.5)	0.9583 ± 0.0015(4.5)	0.9583 ± 0.0013(4.5)	0.9492 ± 0.0290(9)	0.9583 ± 0.0237(4.5)	0.9575 ± 0.0240(7)	0.9500 ± 0.0272(8)
Yeast-spo5	**0.9090 ± 0.0020(1)**	0.9087 ± 0.0008(3)	0.9087 ± 0.0021(3)	0.9086 ± 0.0008(5.5)	0.9087 ± 0.0011(3)	0.9064 ± 0.0571(8)	0.9085 ± 0.0566(7)	0.9086 ± 0.0563(5.5)	0.9054 ± 0.0579(9)
SJAFFE	**0.8886 ± 0.0049(1)**	0.8854 ± 0.0043(2)	0.8485 ± 0.0069(8)	0.8844 ± 0.0092(3)	0.8529 ± 0.0065(6)	0.8144 ± 0.0722(9)	0.8611 ± 0.0602(4)	0.8577 ± 0.0473(5)	0.8519 ± 0.0521(7)
Natural_Scene	**0.5821 ± 0.0041(1)**	0.5725 ± 0.0032(2)	0.5462 ± 0.0079(4)	0.5397 ± 0.0056(5)	0.5482 ± 0.0094(3)	0.4924 ± 0.1678(6)	0.5480 ± 0.0170(9)	0.4819 ± 0.1861(8)	0.4869 ± 0.1733(7)
Human_Gene	**0.7849 ± 0.0026(1)**	0.7841 ± 0.0030(5)	0.7833 ± 0.0021(7)	0.7844 ± 0.0017(4)	0.7846 ± 0.0020(2)	0.6685 ± 0.1187(8)	0.7845 ± 0.1467(3)	0.4819 ± 0.1471(9)	0.7838 ± 0.1466(6)
Movie	0.8354 ± 0.0016(3)	0.8380 ± 0.0019(2)	**0.8384 ± 0.0019(1)**	0.8192 ± 0.0043(5)	0.8136 ± 0.0039(8)	0.8248 ± 0.0948(4)	0.8189 ± 0.1058(6.5)	0.8104 ± 0.1101(9)	0.8189 ± 0.1003(6.5)
Average rank	**1.8846(1)**	2.5000(2)	4.6538(6)	4.4231(4)	3.9615(3)	8.2308(9)	4.5769(5)	7.1154(7)	7.6538(8)

Table 8
Experimental results on 13 real-world datasets measured by runtime performance [s] ↓.

Dataset	RG4LDL	LDL-LRR	LDLLC	LDLSF	LDL-LCLR	AA-BP	BFGS-LDL	CPNN	IIS-LDL
Yeast_alpha	**0.135**	4.710	9.478	10.740	325.731	36.856	0.271	30.142	7.285
Yeast_cdc	**0.129**	3.252	6.622	9.309	272.557	35.151	0.150	28.828	6.540
Yeast_elu	**0.198**	2.948	5.994	9.338	247.768	35.967	0.128	27.758	5.677
Yeast_diau	**0.213**	1.453	3.367	6.931	2.997	33.618	0.063	20.102	2.798
Yeast_heat	**1.067**	1.215	3.208	5.925	1.783	35.083	0.053	19.285	2.440
Yeast_spo	**1.223**	1.050	3.358	6.779	1.748	34.914	0.053	8.359	2.440
Yeast_cold	**0.135**	0.790	2.885	4.632	1.031	32.196	0.036	4.435	2.005
Yeast_dtt	**0.181**	0.913	2.920	4.030	1.006	32.028	0.035	4.880	1.673
Yeast_spo5	**0.257**	0.762	2.675	4.176	0.840	31.752	0.030	4.144	1.443
SJAFFE	**0.170**	0.210	1.693	114.562	26.958	7.098	16.397	2.419	15.144
Natural_Scene	**1.161**	1.853	30.075	403.482	983.672	33.144	0.312	27.643	45.901
Human_Gene	**1.534**	294.397	154.992	3131.411	1206.775	174.411	86.704	465.243	80.173
Movie	16.886	**5.851**	210.769	2122.874	973.160	154.443	9.400	189.726	226.575
MEAN	**1.791**	24.569	33.695	448.784	311.233	52.051	8.741	64.074	30.776

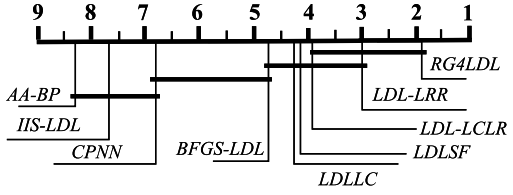
raw features for LDL. RG4LDL without BFGS (the bottom row) is used as the baseline.

It can be seen that the RG4LDL performance without RBM (RG learning) is significantly degraded, suggesting that RG learning is critical in reducing and extracting effective features. The performance of RG4LDL without BFGS is also significantly weaker than that of the full framework, indicating the central role of BFGS in optimizing the objective function. Combining the two components allows the RG4LDL framework to achieve state-of-the-art performance, indicating the importance of integrating both unsupervised RG learning and supervised LDL.

5.3. Friedman test and critical difference diagram

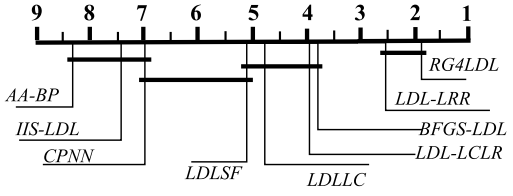
Friedman test statistically compares relative performance among multiple algorithms over multiple datasets [36]. We use this test to validate the statistical significance of the performance of various algorithms given in Tables 2 to 7. Table 10 shows the Friedman statistic F_F and the critical value on each evaluation metric at a significance level of 0.05, among the 9 comparing algorithms and 13 datasets.

As confirmed in Table 10, the F_F values on all the evaluation metrics are greater than the critical value. Therefore, Bonferroni-Dunn test [36] can be adopted as a post hoc test to show the algorithms' relative performances. Specifically, based on Table 10, we use Nemenyi test [36] to check the average ordering comparison between two algorithms. Figs. 4 to 9 represent these results with a critical difference (CD) graph for each evaluation metric, respectively. When the significance level is 0.05, the number of comparison algorithms is 9, and the number of datasets is 13, the CD value is CD = 2.1870 for Nemenyi test. In the CD diagram, the average ordering of each algorithm is marked on the same coordinate axis. If the difference between the average order of the two algorithms is less than the CD value, then there exists no significant difference between the two algorithms and they are connected by a line segment in the CD graph. Algorithms not connected with the RG4LDL in the CD diagrams are considered to have significant performance difference from the control algorithm, given the CD value of 2.1870 at a significance level of 0.05. From the CD diagrams of Nemenyi tests for the six estimation accuracy metrics depicted in Figs. 4 to 9, it can be seen that only the LDL-LRR has line segments connected with the RG4LDL in the tests for Clark and Canberra distances. Thus the



(a) Chebyshev distance

Fig. 4. CD diagrams given CD = 2.1870 of Nemenyi tests on the 9 algorithms for Chebyshev distance evaluation metric.



(b) Clark distance

Fig. 5. CD diagrams given CD = 2.1870 of Nemenyi tests on the 9 algorithms for Clark distance evaluation metric.

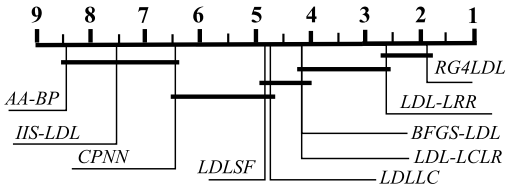
Table 9
Ablation study on LDL predictive performance comparison between RG4LDL (the top row), RG4LDL without RBM (the middle row) and RG4LDL without BFGS (the bottom row) on each of 13 datasets measured by 6 metrics shown as 'mean ± std (rank)'.

Dataset	Cheb↓	Clark↓	Canber↓	KL↓	Cosine↑	Intersec↑
Yeast_alpha	**0.0134 ± 0.0003**	**0.2096 ± 0.0029**	**0.6806 ± 0.0097**	0.0055 ± 0.0002	0.9946 ± 0.0002	**0.9624 ± 0.0005**
	0.0135 ± 0.0002	0.2102 ± 0.0028	0.6824 ± 0.0092	0.0055 ± 0.0001	0.9946 ± 0.0001	0.9623 ± 0.0005
	0.0140 ± 0.0001	0.2200 ± 0.0028	0.7220 ± 0.0092	0.0060 ± 0.0001	0.9938 ± 0.0001	0.9589 ± 0.0005
Yeast_cdc	0.0162 ± 0.0004	**0.2158 ± 0.0045**	**0.6469 ± 0.0137**	0.0070 ± 0.0003	**0.9934 ± 0.0003**	0.9574 ± 0.0009
	0.0162 ± 0.0004	0.2160 ± 0.0036	0.6479 ± 0.0090	0.0070 ± 0.0003	0.9933 ± 0.0002	0.9574 ± 0.0006
	0.0170 ± 0.0004	0.2200 ± 0.0035	0.7000 ± 0.0030	0.0080 ± 0.0003	0.9920 ± 0.0002	0.9540 ± 0.0005
Yeast_elu	**0.0162 ± 0.0001**	**0.1986 ± 0.0012**	**0.5821 ± 0.0051**	0.0062 ± 0.0001	**0.9941 ± 0.0001**	**0.9589 ± 0.0004**
	0.0163 ± 0.0004	0.1994 ± 0.0040	0.5839 ± 0.0117	0.0062 ± 0.0002	0.9940 ± 0.0002	0.9588 ± 0.0008
	0.0170 ± 0.0004	0.2100 ± 0.0040	0.6290 ± 0.0117	0.0070 ± 0.0002	0.9930 ± 0.0002	0.9550 ± 0.0008
Yeast_diau	**0.0369 ± 0.0006**	**0.2005 ± 0.0032**	**0.4304 ± 0.0058**	0.0131 ± 0.0004	0.9879 ± 0.0003	0.9403 ± 0.0008
	0.0370 ± 0.0007	0.2006 ± 0.0043	0.4306 ± 0.0088	0.0131 ± 0.0005	0.9879 ± 0.0004	0.9403 ± 0.0011
	0.0380 ± 0.0007	0.2010 ± 0.0043	0.4400 ± 0.0088	0.0140 ± 0.0005	0.9870 ± 0.0004	0.9390 ± 0.0011
Yeast_heat	**0.0418 ± 0.0006**	**0.1810 ± 0.0034**	**0.3610 ± 0.0053**	**0.0124 ± 0.0004**	**0.9882 ± 0.0003**	**0.9408 ± 0.0008**
	0.0422 ± 0.0009	0.1825 ± 0.0041	0.3639 ± 0.0074	0.0126 ± 0.0006	0.9880 ± 0.0005	0.9403 ± 0.0012
	0.0430 ± 0.0009	0.1850 ± 0.0041	0.3700 ± 0.0074	0.0130 ± 0.0006	0.9870 ± 0.0005	0.9390 ± 0.0012
Yeast_spo	**0.0583 ± 0.0020**	**0.2498 ± 0.0086**	**0.5132 ± 0.0178**	0.0247 ± 0.0014	0.9769 ± 0.0013	**0.9155 ± 0.0029**
	0.0584 ± 0.0007	0.2500 ± 0.0031	0.5138 ± 0.0062	**0.0246 ± 0.0007**	0.9769 ± 0.0006	0.9154 ± 0.0010
	0.0600 ± 0.0007	0.2540 ± 0.0031	0.5200 ± 0.0062	0.0250 ± 0.0007	0.9760 ± 0.0006	0.9140 ± 0.0010
Yeast_cold	**0.0510 ± 0.0008**	**0.1395 ± 0.0023**	**0.2401 ± 0.0037**	0.0122 ± 0.0005	**0.9886 ± 0.0004**	**0.9409 ± 0.0008**
	0.0512 ± 0.0011	0.1398 ± 0.0026	0.2406 ± 0.0040	0.0122 ± 0.0005	0.9885 ± 0.0004	0.9407 ± 0.0010
	0.0530 ± 0.0011	0.1410 ± 0.0026	0.2441 ± 0.0040	0.0130 ± 0.0005	0.9881 ± 0.0004	0.9399 ± 0.0010
Yeast_dtt	**0.0359 ± 0.0006**	**0.0979 ± 0.0017**	**0.1686 ± 0.0032**	**0.0062 ± 0.0003**	0.9941 ± 0.0002	**0.9584 ± 0.0008**
	0.0361 ± 0.0005	0.0983 ± 0.0021	0.1692 ± 0.0035	0.0063 ± 0.0004	0.9941 ± 0.0003	0.9583 ± 0.0007
	0.0370 ± 0.0005	0.0100 ± 0.0021	0.1755 ± 0.0035	0.0070 ± 0.0004	0.9939 ± 0.0003	0.9555 ± 0.0007
Yeast_spo5	**0.0910 ± 0.0020**	**0.1836 ± 0.0048**	0.2819 ± 0.0067	**0.0291 ± 0.0017**	**0.9742 ± 0.0013**	**0.9090 ± 0.0020**
	0.0914 ± 0.0023	0.1842 ± 0.0045	0.2819 ± 0.0072	0.0293 ± 0.0014	0.9741 ± 0.0010	0.9086 ± 0.0023
	0.0920 ± 0.0023	0.1865 ± 0.0045	0.2869 ± 0.0072	0.0300 ± 0.0014	0.9735 ± 0.0010	0.9077 ± 0.0023
SJAFFE	**0.0859 ± 0.0057**	**0.3278 ± 0.0108**	**0.6668 ± 0.0215**	**0.0427 ± 0.0052**	**0.9603 ± 0.0061**	**0.8886 ± 0.0049**
	0.1178 ± 0.0123	0.4683 ± 0.0219	0.9681 ± 0.0530	0.0885 ± 0.0105	0.9238 ± 0.0116	0.8401 ± 0.0123
	0.1170 ± 0.0123	0.4190 ± 0.0219	0.8750 ± 0.0530	0.0770 ± 0.0105	0.9338 ± 0.0116	0.8501 ± 0.0123
Natural_Scene	**0.2976 ± 0.0036**	**2.3791 ± 0.0086**	**6.5057 ± 0.0183**	**0.7404 ± 0.0142**	**0.7479 ± 0.0045**	**0.5821 ± 0.0041**
	\	\	\	\	\	\
	0.3400 ± 0.0016	2.4210 ± 0.0026	6.6005 ± 0.0113	0.8601 ± 0.0042	0.7089 ± 0.0085	0.4889 ± 0.0041
Human_Gene	0.0533 ± 0.0009	**2.1060 ± 0.0212**	**14.4166 ± 0.1696**	**0.2355 ± 0.0061**	**0.8351 ± 0.0030**	**0.7849 ± 0.0026**
	0.0533 ± 0.0015	2.1090 ± 0.0276	14.4363 ± 0.2215	0.2359 ± 0.0073	0.8348 ± 0.0041	0.7846 ± 0.0034
	0.0534 ± 0.0015	2.1199 ± 0.0276	14.5413 ± 0.2215	0.2370 ± 0.0073	0.8330 ± 0.0041	0.7835 ± 0.0034
Movie	**0.1153 ± 0.0010**	**0.5200 ± 0.0041**	**0.9954 ± 0.0089**	**0.0986 ± 0.0019**	**0.9351 ± 0.0013**	**0.8354 ± 0.0016**
	0.1329 ± 0.0017	0.5784 ± 0.0072	1.1156 ± 0.0141	0.1344 ± 0.0037	0.9159 ± 0.0021	0.8135 ± 0.0023
	0.1415 ± 0.0017	0.5850 ± 0.0072	1.1376 ± 0.0141	0.1388 ± 0.0037	0.9110 ± 0.0021	0.8050 ± 0.0023

The RG4LDL without RBM does not work on Natural_Scene dataset.

Table 10
Friedman statistics F_F for each evaluation metric with critical value at significance level 0.05 (comparing algorithms 9, datasets 13).

Evaluation metric	F_F	Critical value
Chebyshev distance	19.5996	2.036
Clark distance	24.3199	
Canberra distance	19.8939	
Kullback–Leibler divergence	14.0494	
cosine coefficient	17.7629	
intersectional similarity	23.3283	



(c) Canberra distance

Fig. 6. CD diagrams given CD = 2.1870 of Nemenyi tests on the 9 algorithms for Canberra distance evaluation metric.

conclusion can be led to that the RG4LDL consistently achieves the significant good estimation accuracy statistically.

5.4. Statistical validation of LDL predictive results

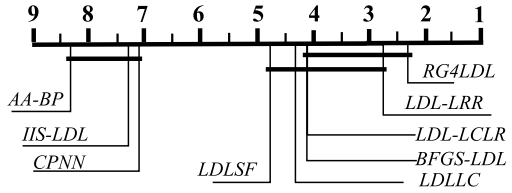
To show the statistical relationships among the 9 LDL algorithms on the 13 real-world datasets, Wilcoxon signed-rank test [36] is employed to validate whether our RG4LDL performs significantly better than the other 8 benchmarks, in terms of each evaluation metric. Table 11 summarizes the Wilcoxon signed-rank statistical test results, where the p-values for the corresponding tests are shown in the brackets. Validated by the Wilcoxon signed-rank test results of Table 11, it is statistically significant that our RG4LDL outperforms all the benchmarks with the exception of LDL-LRR, in terms of all six metrics. The

Table 11
Wilcoxon signed-rank test for LDL predictive performance of RG4LDL vs. benchmarks (significance level $\alpha = 0.05$, p-values shown in the brackets).

RG4LDL	Evaluation metric					
versus	Cheb	Clark	Canber	KL	Cosine	Intersec
LDL-LRR	TIE[0.587890625]	TIE[0.533492435410]	TIE[0.497314453125]	TIE[0.752669134351]	TIE[0.811070129333]	TIE[0.375732421875]
LDLLC	WIN[0.021484375]	WIN[0.022647175233]	WIN[0.021484375]	WIN[0.096614208447]	WIN[0.159369547355]	WIN[0.032423418011]
LDLSF	WIN[0.000244140625]	WIN[0.005062032126]	WIN[0.000244140625]	WIN[0.012772494550]	WIN[0.011513828076]	WIN[0.001925774646]
LDL-LCLR	WIN[0.002183044737]	WIN[0.007632441648]	WIN[0.002830577835]	WIN[0.012772494550]	WIN[0.024353118662]	WIN[0.004753229214]
AA-BP	WIN[0.000244140625]	WIN[0.000244140625]	WIN[0.000244140625]	WIN[0.000244140625]	WIN[0.000244140625]	WIN[0.000244140625]
BFGS-LDL	WIN[0.000244140625]	WIN[0.005033508200]	WIN[0.000244140625]	WIN[0.017290280592]	WIN[0.011310671074]	WIN[0.004948477274]
CPNN	WIN[0.021484375]	WIN[0.000244140625]	WIN[0.000244140625]	WIN[0.000244140625]	WIN[0.000244140625]	WIN[0.000244140625]
IIS-LDL	WIN[0.000244140625]	WIN[0.000244140625]	WIN[0.000244140625]	WIN[0.000244140625]	WIN[0.000244140625]	WIN[0.000244140625]

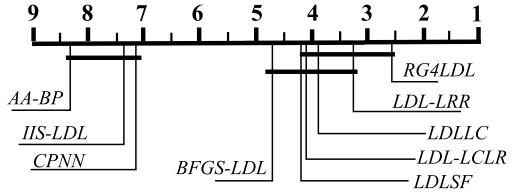
Table 12
Bayesian signed-rank test for LDL performance of RG4LDL vs. benchmarks (significance level rope = 0.0001, default prior strength is 0.6).

RG4LDL	Evaluation metric					
versus	Cheb	Clark	Canber	KL	Cosine	Intersec
LDL-LRR	[0.73598, 0.0, 0.26402]	[0.63526, 0.1578, 0.20694]	[0.7721, 0.00044, 0.22746]	[0.2334, 0.41096, 0.35564]	[0.12528, 0.71822, 0.1565]	[0.77086, 0.01618, 0.21296]
LDLLC	[0.99416, 0.0, 0.00584]	[0.9047, 0.07446, 0.02084]	[0.99482, 0.0, 0.00518]	[0.6368, 0.33618, 0.02702]	[0.37708, 0.59578, 0.02714]	[0.96124, 0.02388, 0.01488]
LDLSF	[1.0, 0.0, 0.0]	[0.94664, 0.05336, 0.0]	[1.0, 0.0, 0.0]	[0.85806, 0.14194, 0.0]	[0.6471, 0.3529, 0.0]	[0.86692, 0.13308, 0.0]
LDL-LCLR	[1.0, 0.0, 0.0]	[0.74908, 0.25092, 0.0]	[0.99996, 4e−05, 0.0]	[0.7762, 0.2238, 0.0]	[0.51996, 0.48004, 0.0]	[0.86766, 0.13234, 0.0]
AA-BP	[1.0, 0.0, 0.0]	[1.0, 0.0, 0.0]	[1.0, 0.0, 0.0]	[1.0, 0.0, 0.0]	[1.0, 0.0, 0.0]	[1.0, 0.0, 0.0]
BFGS-LDL	[1.0, 0.0, 0.0]	[0.94736, 0.05264, 0.0]	[1.0, 0.0, 0.0]	[0.77368, 0.22632, 0.0]	[0.56966, 0.43034, 0.0]	[0.92356, 0.07644, 0.0]
CPNN	[0.98592, 0.0, 0.01408]	[1.0, 0.0, 0.0]	[1.0, 0.0, 0.0]	[1.0, 0.0, 0.0]	[1.0, 0.0, 0.0]	[1.0, 0.0, 0.0]
IIS-LDL	[1.0, 0.0, 0.0]	[1.0, 0.0, 0.0]	[1.0, 0.0, 0.0]	[1.0, 0.0, 0.0]	[1.0, 0.0, 0.0]	[1.0, 0.0, 0.0]



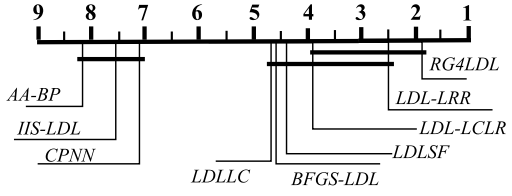
(d) Kullback-Leibler distance

Fig. 7. CD diagrams given CD = 2.1870 of Nemenyi tests on the 9 algorithms for Kullback–Leibler divergence metric.



(e) Cosine coefficient

Fig. 8. CD diagrams given CD = 2.1870 of Nemenyi tests on the 9 algorithms for cosine coefficient evaluation metric.



(f) Intersection similarity

Fig. 9. CD diagrams given CD = 2.1870 of Nemenyi tests on the 9 algorithms for intersectional similarity evaluation metric.

performance of RG4LDL and LDL-LRR are statistically tied judged by Wilcoxon signed-rank test.

To further discover to what extent our RG4LDL outperforms the other 8 algorithms, we use Bayesian sign ranking test [37] as the statistical test. Compared to Wilcoxon signed-rank test, Bayesian sign ranking test provides more statistical details/information. Table 12 summarizes the results of this statistical test, where the values in parentheses [a,b,c] represent the probabilities of the control algorithm (our RG4LDL in this case) being better, comparable, or worse than the comparison algorithm [WIN,TIE,LOSE]. The default assumption is that the performance of the two algorithms is the same. The prior strength is the strength of the null hypothesis, which means the probability of the null hypothesis being true is 0.6. If the difference between the results of the two algorithms is less than rope = 0.0001, their performance is considered similar. Verified by Bayesian sign ranking test results of Table 12, our RG4LDL statistically outperforms the other comparison algorithms. In particular, compared with the second best LDL-LRR, our RG4LDL 'WIN's in Cheb, Clark, Canber and Intersec metrics, and 'TIE's in KL and Cosine metric.

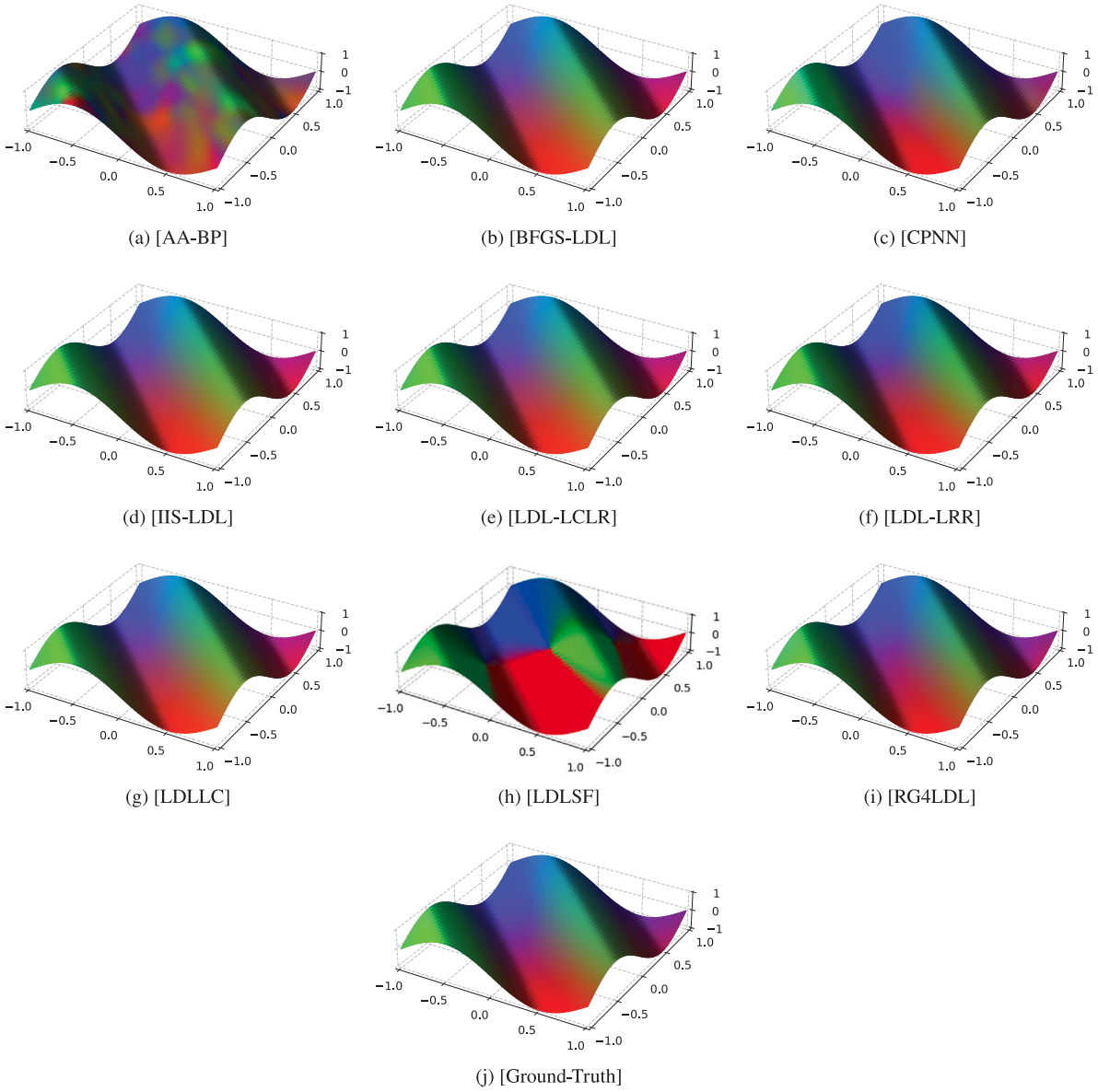
Therefore, validated by these two statistical tests, we can confidently state that our RG4LDL outperforms the 8 existing state-of-the-art LDL schemes.

5.5. Visualization of label distribution

The artificial toy dataset is used to compare how good the mappings from the instance to the label distribution generated by various LDL algorithms. In order to visually show the result of an LDL algorithm, the description degrees of the three labels are regarded as the three color channels of the RGB color space, respectively. In this way, the color of a point in the instance space will visually represent its label distribution. Thus the predictions made by various LDL algorithms can be compared with the ground-truth label distributions through observing the color patterns on the manifold where the test examples lie on. For easier comparison, the images are visually enhanced by applying a decorrelation stretch process. Fig. 10 compares the predicted label distributions by various LDL algorithms with the ground-truth, where the three-dimensional label distributions of the RGB color channels are displayed separately, and the three axises correspond to the three components, x_1, x_2 and x_3, of the example. The closer the color pattern of the predicted label distribution is to the ground truth, the better the prediction performance. It can be seen that BFGSLDL, CPNN, IIS-LLD, LDL-LCLR, LDL-LRR, LDLLC and our RG4LDL are able to obtain color distributions similar to the ground truth pattern, while AA-BP and LDLSF fail to achieve good prediction performance on this artificial dataset. A close examination of the visualization results in Fig. 10 shows that our RG4LDL is the closest to the ground-truth.

5.6. Case study for label ranking relationship

To intuitively illustrate the role of label ranking relationships, we select some representative samples from four different datasets. Figs. 11 to 14 show the predicted results of the different algorithms on



(a) [AA-BP] (b) [BFGS-LDL] (c) [CPNN]

(d) [IIS-LDL] (e) [LDL-LCLR] (f) [LDL-LRR]

(g) [LDLLC] (h) [LDLSF] (i) [RG4LDL]

(j) [Ground-Truth]

Fig. 10. Comparison between the ground-truth and predicted label distributions (as RBG color) on the artificial data test manifold.

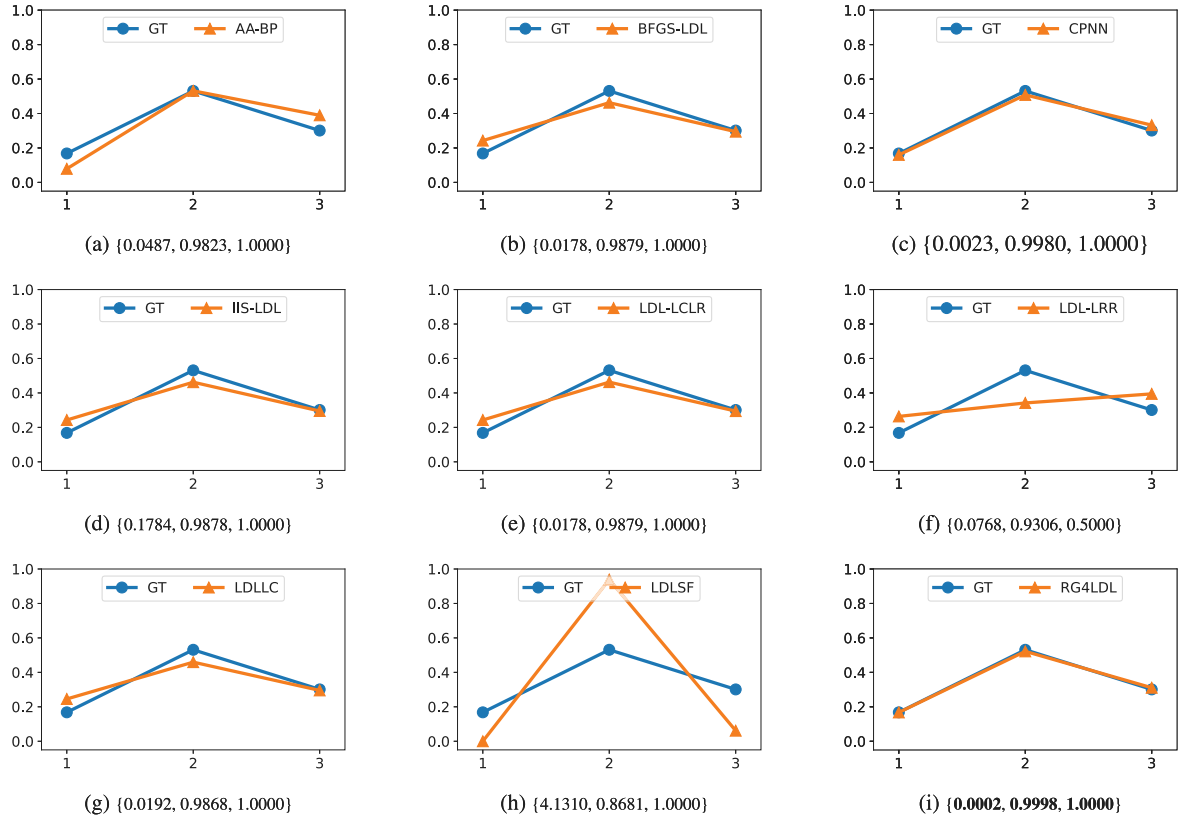
four typical test instances with 3, 4, 7 and 14 labels, respectively. Each figure shows the corresponding predicted distributions of one LDL algorithm. The numbers below each predicted label distributions correspond to three evaluation measures, and they are KL, Cosine, and Spearman's rank. It can be clearly seen that the label distribution fitting curves of our RG4LDL are the closest to the ground-truth curves, which is consistent with its three evaluation metrics.

6. Discussion

The key differences between our RG4LDL and other existing state-of-the-art LDL methods are as follows. Unlike traditional LDL methods such as CPNN and IIS-LDL, which rely on fixed feature representations, our RG4LDL employs an unsupervised RG-based feature extraction mechanism. This ensures that the most relevant degrees of freedom are retained, leading to better feature representations for LDL tasks. Unlike our RG4LDL, the LDL methods like LDL-LRR and LDL-LCLR focus on label correlations but do not explicitly optimize feature extraction in an unsupervised manner. Furthermore, our RG4LDL significantly reduces the complexity of high-dimensional LDL models by iteratively coarse-graining the feature space. This makes RG4LDL more efficient and scalable, particularly compared to the methods like CPNN, which suffer from overfitting and require extensive training iterations due to the large number of neural network parameters. RG4LDL seamlessly combines unsupervised RG feature learning with supervised LDL prediction in an end-to-end manner. This integration distinguishes it from the approaches like IIS-LDL and BFGS-LDL, which operate solely in a supervised learning paradigm. Last but not least, RG4LDL is significantly faster than the existing LDL methods, which is evidently demonstrated in the runtime experimental results.

The experimental results of Section 5 convincingly demonstrate that our RG4LDL outperforms the existing state-of-the-art LDL methods in terms of label distribution prediction accuracy as measured by multiple evaluation metrics, while offering the most efficient runtime efficiency with the average runtime an order of magnitude faster than the second-best LDL-LRR method. These significant improvements are owing to the following reasons.

RG4LDL leverages the RG principle and implements the RBM to iteratively reduce the degrees of freedom in the feature space. This approach ensures efficient feature extraction while preserving label distribution information, making it particularly suitable for high-

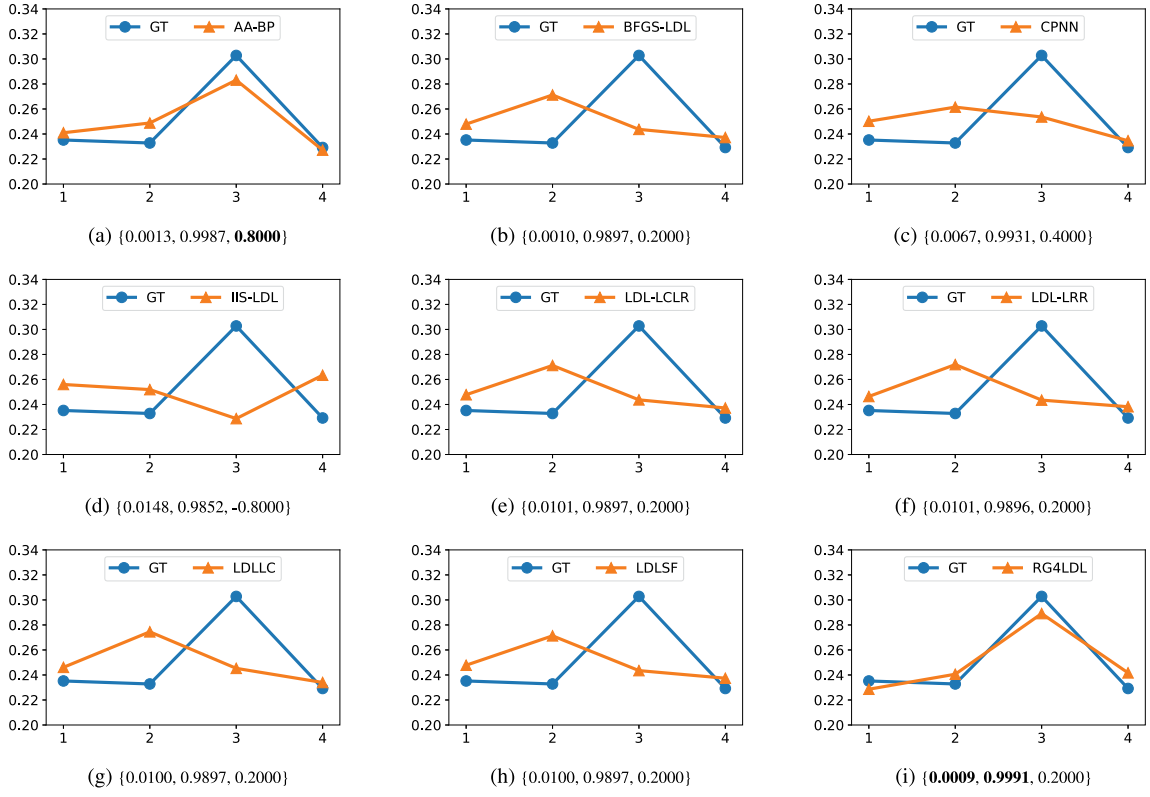


(a) {0.0487, 0.9823, 1.0000} (b) {0.0178, 0.9879, 1.0000} (c) {0.0023, 0.9980, 1.0000}

(d) {0.1784, 0.9878, 1.0000} (e) {0.0178, 0.9879, 1.0000} (f) {0.0768, 0.9306, 0.5000}

(g) {0.0192, 0.9868, 1.0000} (h) {4.1310, 0.8681, 1.0000} (i) {**0.0002**, **0.9998**, **1.0000**}

Fig. 11. The role of the label ranking relation in fitting the ground-truth label distribution illustrated by typical examples sampled from Artificial dataset (3 labels).

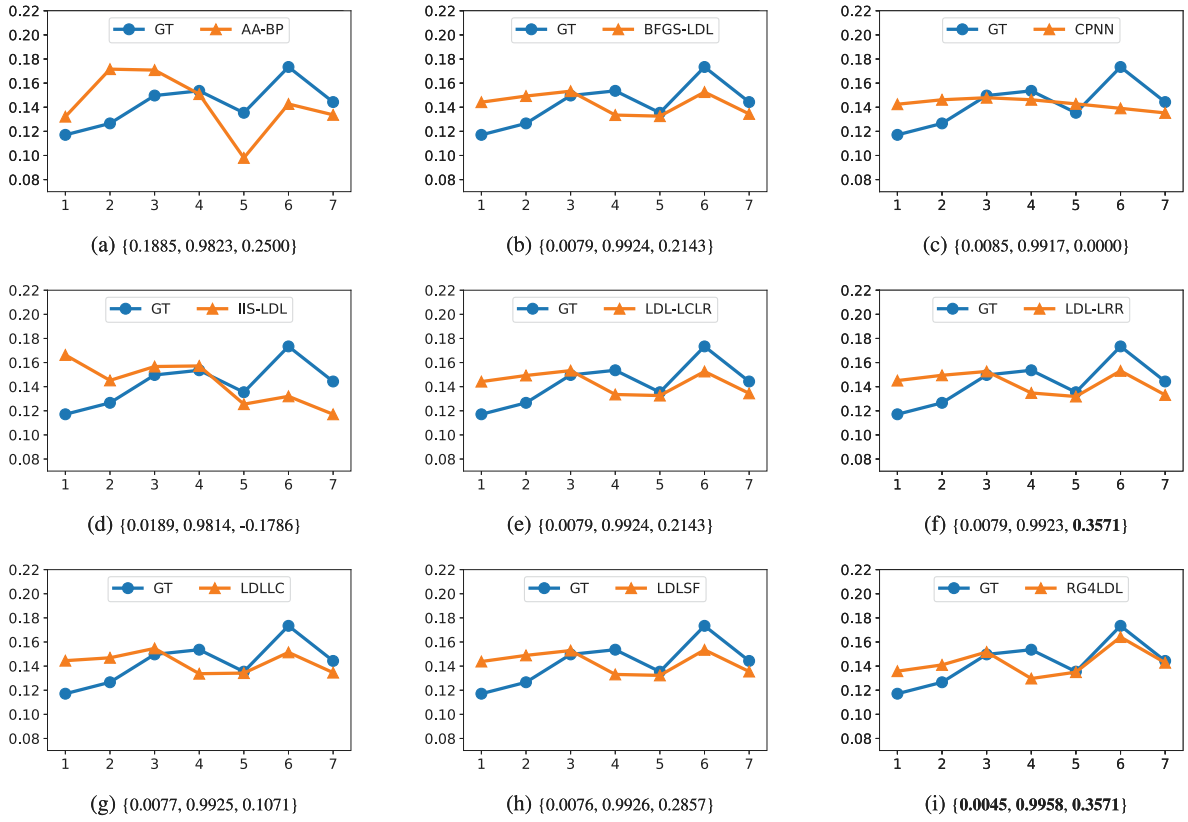


(a) {0.0013, 0.9987, **0.8000**} (b) {0.0010, 0.9897, 0.2000} (c) {0.0067, 0.9931, 0.4000}

(d) {0.0148, 0.9852, **-0.8000**} (e) {0.0101, 0.9897, 0.2000} (f) {0.0101, 0.9896, 0.2000}

(g) {0.0100, 0.9897, 0.2000} (h) {0.0100, 0.9897, 0.2000} (i) {**0.0009**, **0.9991**, 0.2000}

Fig. 12. The role of the label ranking relation illustrated by typical examples sampled from Yeast_cold dataset (4 labels).

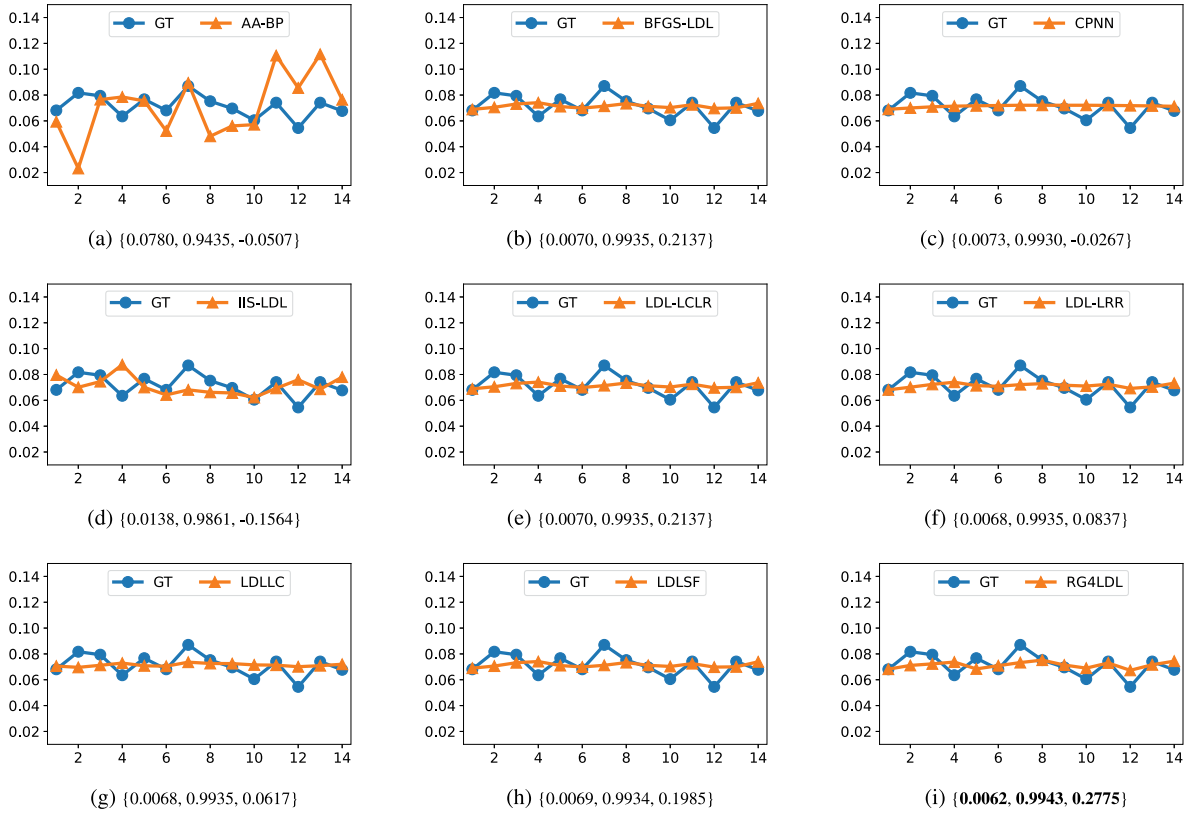


(a) {0.1885, 0.9823, 0.2500} (b) {0.0079, 0.9924, 0.2143} (c) {0.0085, 0.9917, 0.0000}

(d) {0.0189, 0.9814, -0.1786} (e) {0.0079, 0.9924, 0.2143} (f) {0.0079, 0.9923, **0.3571**}

(g) {0.0077, 0.9925, 0.1071} (h) {0.0076, 0.9926, 0.2857} (i) {**0.0045**, **0.9958**, **0.3571**}

Fig. 13. The role of the label ranking relation illustrated by typical examples sampled from Yeast_diau dataset (7 labels).



(a) {0.0780, 0.9435, -0.0507} (b) {0.0070, 0.9935, 0.2137} (c) {0.0073, 0.9930, -0.0267}

(d) {0.0138, 0.9861, -0.1564} (e) {0.0070, 0.9935, 0.2137} (f) {0.0068, 0.9935, 0.0837}

(g) {0.0068, 0.9935, 0.0617} (h) {0.0069, 0.9934, 0.1985} (i) {**0.0062**, **0.9943**, **0.2775**}

Fig. 14. The role of the label ranking relation illustrated by typical examples sampled from Yeast_elu dataset (14 labels)..

Table 13
The main symbols involved in the paper and their dimensions.

Symbol	Description	Dimensionality
$\boldsymbol{x}_t$	Current visible layer vector (tth input feature vector)	q
$\boldsymbol{h}_t$	Current hidden layer vector (tth extracted feature vector)	m
$\boldsymbol{x}_t^*$	Reconstructed visible layer vector (approximated tth input feature vector from hidden layer)	q
$\boldsymbol{h}_t^*$	Reconstructed hidden layer vector (extracted tth feature vector from reconstructed visible layer)	m
$\boldsymbol{W}$	Weight matrix between visible and hidden layers of RBM	$q \times m$
$\boldsymbol{b}$	Visible layer bias vector of RBM	q
$\boldsymbol{c}$	Hidden layer bias vector of RBM	m
$\boldsymbol{\theta}$	Parameter matrix of condition probability (label distribution) model	$\bar{c} \times m$

dimensional datasets. By combining unsupervised RG-based feature learning with supervised LDL prediction in an end-to-end manner, RG4LDL achieves significantly faster convergence in the feature optimization process and results in better label distribution predictive model, compared to traditional LDL methods. It can be seen that by effectively addressing the key challenges of LDL, RG4LDL not only improves the predictive performance of LDL tasks but also enhances the practicality of LDL in real-world scenarios.

Although RG4LDL is designed for LDL tasks, its underlying principles and framework can be extended to broader machine learning problems, such as multi-label classification and recommendation systems. Specifically, the RBM-based feature extraction in RG4LDL can naturally handle dependencies between multiple labels. By modifying the output layer to predict binary relevance scores for each label, RG4LDL can be adapted for multi-label classification tasks. This extension can be particularly useful in domains like image annotation, where instances are associated with multiple labels. The RG principle, which identifies relevant degrees of freedom, aligns well with the needs of recommendation systems that require effective dimensionality reduction and feature extraction. RG4LDL can be extended to model user-item interactions by treating users and items as separate sets of features, while the label distributions can represent the predicted relevance scores of items for each user.

7. Conclusions

In this paper, we have proposed RG4LDL, a novel framework that integrates the renormalization group (RG) principle with label distribution learning (LDL). To the best of our knowledge, this study is the first to incorporate the RG principle, implemented as a restricted Boltzmann machine (RBM), into the LDL process. RG4LDL addresses the major challenges faced by existing LDL methods, such as high model complexity, slow convergence, and the scarcity of label distribution-annotated data, by iteratively identifying relevant degrees of freedom and optimizing feature representation. Extensive experimental results have demonstrated that the proposed RG4LDL outperforms all the existing state-of-the-art LDL methods in terms of label distribution prediction accuracy, while imposing dramatically lower computational complexity than these existing benchmarks.

While our RG4LDL have demonstrated significant improvements, the current framework is specifically designed for LDL tasks. To remove this limitation, it requires further adaptations to generalize to broader tasks, such as multi-label classification and recommendation systems. Our future work will focus on exploring hybrid architectures that combine RG4LDL with other deep learning paradigms, to improve scalability, and extending the framework to broader machine learning tasks.

CRediT authorship contribution statement

Chao Tan: Writing – review & editing, Writing – original draft, Software, Project administration, Methodology, Investigation, Data curation, Conceptualization. **Sheng Chen:** Writing – review & editing, Supervision, Formal analysis. **Jiaxi Zhang:** Visualization, Validation, Software. **Zilong Xu:** Visualization, Validation, Software. **Xin Geng:** Resources, Funding acquisition, Data curation, Conceptualization. **Genlin Ji:** Resources, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by National Natural Science Foundation of China under Grant 62476135, 61702270, 62125602, U24A20324, and 62076063. This work was also supported by the Jiangsu Science Foundation (BK20243012, BG2024036), and by Industry Projects in Jiangsu S & T Pillar Program under Grant BE2023089. Dr Tan would like to thank the sponsorship of Chinese Scholarship Council for funding her research at School of Electronics and Computer Science, University of Southampton, UK.

Appendix

Table 13 describes the main symbols and their dimensions adopted in the paper.

Data availability

Data will be made available on request.

References

[1] K. Zhang, et al., Artificial intelligence in drug development, Nature Med. 31 (2025) 45–59.
[2] S. Li, et al., CausalStock: Deep end-to-end causal discovery for news-driven multi-stock movement prediction, in: Proc. NeurIPS 2024 (Vancouver, Canada), 2024, pp. 47432–47454.
[3] M. Haider, et al., NetworkGym: Reinforcement learning environments for multi-access traffic management in network simulation, 2024, arXiv preprint arXiv:2411.04138.
[4] Z. Li, et al., TEG-DB: A comprehensive dataset and benchmark of textual-edge graphs, in: Proc. NeurIPS 2024 (Vancouver, Canada), 2024, pp. 60980–60998.
[5] M.-L. Zhang, et al., Leveraging implicit relative labeling-importance information for effective multi-label learning, IEEE Trans. Knowl. Data Eng. 33 (5) (2021) 2057–2070.
[6] G. Tsoumakas, I. Katakis, Multi-label classification: An overview, Int. J. Data Warehous. Min. 3 (3) (2007) 1–13.
[7] S. He, et al., Reinforced multi-label image classification by exploring curriculum, in: Proc. AAAI 2018 (New Orleans, la, USA), Vol. 2–7, 2018, pp. 3183–3190.
[8] X. Geng, Label distribution learning, IEEE Trans. Knowl. Data Eng. 28 (7) (2016) 1734–1748.
[9] P. Li, et al., Deep label refinement for age estimation, Pattern Recognit. 100 (2020) 107178, 1–12.
[10] X. Geng, C. Yin, Z.-H. Zhou, Facial age estimation by learning from label distributions, IEEE Trans. Pattern Anal. Mach. Intell. 35 (10) (2013) 2401–2412.
[11] X. Zhou, et al., Facial depression recognition by deep joint label distribution and metric learning, IEEE Trans. Affect. Comput. 13 (3) (2022) 1605–1618.
[12] J. Zhang, et al., Safe incomplete label distribution learning, Pattern Recognit. 125 (2022) 108518, 1–14.
[13] W. Qian, et al., Partial label feature selection based on noisy manifold and label distribution, Pattern Recognit. 156 (2024) 110791, 1–15.

[14] X. Geng, X. Qian, Z. Huo, Y. Zhang, Head pose estimation based on multivariate label distribution, IEEE Trans. Pattern Anal. Mach. Intell. 44 (4) (2022) 1974–1991.
[15] B.-B. Gao, et al., Deep label distribution learning with label ambiguity, IEEE Trans. Image Process. 26 (6) (2017) 2825–2838.
[16] G. Montúfar, Restricted Boltzmann machines: Introduction and review, in: N. Ay, P. Gibilisco, F. Matúš (Eds.), Information Geometry and Its Applications, Springer International Publishing, 2018, pp. 75–115.
[17] X. Kong, M.K. Ng, Z.-H. Zhou, Transductive multilabel learning via label set propagation, IEEE Trans. Knowl. Data Eng. 25 (3) (2013) 704–719.
[18] M.-L. Zhang, L. Wu, Lift: Multi-label learning with label-specific features, IEEE Trans. Pattern Anal. Mach. Intell. 37 (1) (2015) 107–120.
[19] S.S. Bucak, R. Jin, A.K. Jain, Multi-label learning with incomplete class assignments, in: Proc. CVPR 2011 (Colorado Springs, CO, USA), vol. 20–25, 2011, pp. 2801–2808.
[20] H.-Y. Lo, J.-C. Wang, H.-M. Wang, S.-D. Lin, Cost-sensitive multi-label learning for audio tag annotation and retrieval, IEEE Trans. Multimed. 13 (3) (2011) 518–529.
[21] J.D. Wulfkuhle, L.A. Liotta, E.F. Petricoin, Proteomic applications for the early detection of cancer, Nat. Rev. Cancer 3 (4) (2003) 267–275.
[22] T. Ren, et al., Label distribution learning with label-specific features, in: Proc. IJCAI 2019 (Macao, China), Vol. 10–16, 2019, pp. 3318–3324.
[23] T. Ren, X. Jia, W. Li, S. Zhao, Label distribution learning with label correlations via low-rank approximation, in: Proc. IJCAI 2019 (Macao, China), Vol. 10–16, 2019, pp. 3325–3331.
[24] X. Jia, et al., Label distribution learning by maintaining label ranking relation, IEEE Trans. Knowl. Data Eng. 35 (2) (2023) 1695–1707.
[25] A.L. Berger, S.A.D. Pietra, V.J.D. Pietra, A maximum entropy approach to natural language processing, Comput. Linguist. 22 (1) (1996) 39–71.
[26] T. Qian, et al., Contrastive learning from label distribution: A case study on text classification, Neurocomputing 507 (2022) 208–220.
[27] Y. Wang, et al., Contrastive label enhancement, in: Proc. IJCAI 2023 (Macao, S.a.R., China), Vol. 19–25, 2023, pp. 4353–4361.
[28] R. Guana, et al., Dual contrastive label enhancemen, Pattern Recognit. 160 (2025) 111183, 1–11.
[29] G.E. Hinton, A practical guide to training restricted Boltzmann machines, in: G. Montavon, G.B. Orr, K.-R. Müller (Eds.), Neural Networks: Tricks of the Trade, in: Lecture Notes in Computer Science, vol. 7700, Springer, Berlin, Heidelberg, 2012, pp. 599–619.
[30] J.J. Hopfield, Hopfield network, Scholarpedia 2 (5) (1977).
[31] L.P. Kadanoff, Scaling laws for ising models near T_c, Phys. Phys. Fiz. 2 (6) (1966) 263–272.
[32] P. Mehta, D.J. Schwab, An exact mapping between the variational renormalization group and deep learning, 2014, arXiv:1410.3831.
[33] M. Buscema, Back propagation neural networks, Subst. Use Misuse 33 (2) (1998) 233–270.
[34] J. Nocedal, S.J. Wright, Numerical Optimization, second ed., Spinger, New York, NY, USA, 2006.
[35] X. Jia, W. Li, J. Liu, Y. Zhang, Label distribution learning by exploiting label correlations, in: Proc. AAAI 2018 (New Orleans, la, USA), Vol. 2–7, 2018, pp. 3310–3317.
[36] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (1) (2006) 1–30.
[37] A. Benavoli, G. Corani, J. Demšar, M. Zaffalon, Time for a change: A tutorial for comparing multiple classifiers through Bayesian analysis, J. Mach. Learn. Res. 18 (77) (2017) 1–36.