Structured Random Forest for Label Distribution Learning

Mengting Chen, Xinggang Wang, Bin Feng,, Wenyu Liu*

School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China

Abstract

Label distribution learning (LDL) has proven effective in many machine learning applications. Previous LDL methods have focused on learning a non-linear conditional probability mass function by maximizing entropy or minimizing the Kullback-Leibler (K-L) divergence. In order to make full use of the connection among different classes, a method called structured random forest (StructRF) regression is used which has been applied to semantic image labeling and edge detection. It is a general LDL model that treats the distribution as an integral whole. In StructRF, all label distributions are mapped to a discrete space at each split node in a random forest. In this way, standard information gain measures can be evaluated. Then the predicted distribution can be obtained directly without calculating the probability of every class individually during the test. StructRF is proved to be faster for training. Excellent performance is obtained with higher efficiency and lower standard deviation. Besides, we propose an adaptive variable step method that can speed up the training process by removing the most calculations of the information gain. It is suitable for the most decision tree based models.

Keywords: Structured random forest, Label distribution learning

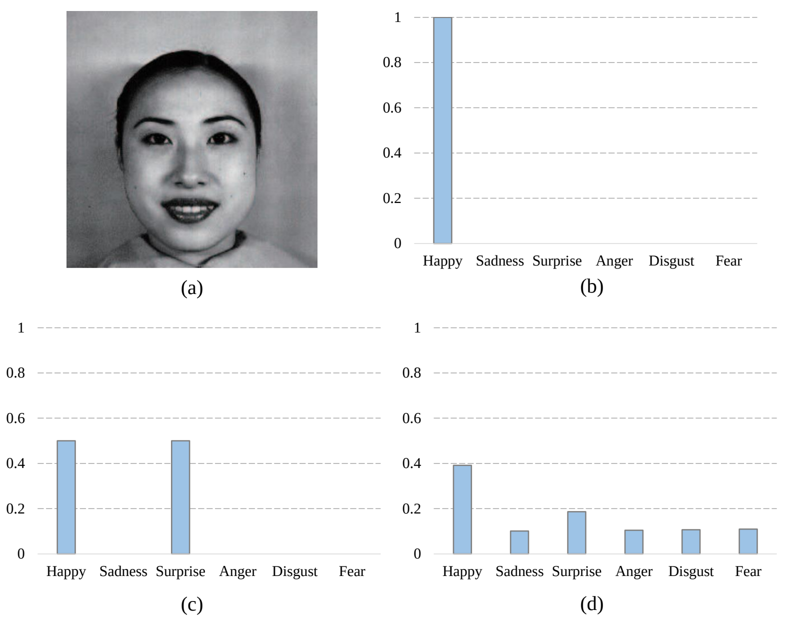

Figure 1: **Three different ways to label an instance.** (a) An instance from a facial expression database BU_3DFE. (b) Single label. (c) Multi-label. (d) Label distribution.

1. Introduction

Learning with ambiguity is a popular and significant topic in machine learning and data mining. When we focus on the mapping from the instances to the labels, labeling one instance with a label distribution can be more efficient and accurate in a lot of application scenarios such as facial expression estimation, age estimation, and crowd opinion prediction. As shown in Fig. 1, label distribution learning (LDL) is a more general label strategy that can directly deal with the problem "how much does each emotion describe the facial expression comes from BU_3DFE [1]" while single-label learning (SLL) and multi-label learning (MLL) [2] aim only to answer the question that "which emotion can describe the facial expression" [3].

*Corresponding author
Email address: `liuwy@hust.edu.cn` (Wenyu Liu)

The proposal of LDL was first formally introduced by Geng et al. [3], in which six working LDL algorithms were proposed in three strategies: problem transformation, algorithm adaptation, and specialized algorithm design. The two specialized algorithms SA-IIS and SA-BFGS stood out by the significant advantage of accuracy and efficiency. This illustrates that LDL requires the special designs to achieve good performance, which encourages us to follow the pace of specialized design. Therefore, SA-IIS and SA-BFGS are two major algorithms that we compared with. Both SA-IIS and SA-BFGS are represented by a maximum entropy model [4]. SA-IIS uses a strategy similar to Improved Iterative Scaling (IIS) [5]. IIS starts with an arbitrary set of parameters. Then for each step, it updates the current estimate of the parameters. SA-BFGS follows the idea of an effective quasi-Newton method BFGS, which performs much more efficiently than the standard line search Newton method. There are a lot of LDL algorithms which can fit some real applications well. For example, in order to settle the problem of lacking sufficient and complete training data, Geng et al. proposed two algorithms named IIS-LLD and CPNN for facial age estimation [6]. After that, Geng et al. observed that the facial appearance changes at different speed over different age period [7], so they further proposed an adaptive label distribution learning (ALDL) algorithm for better estimation. Geng and Hou combined LDL with SVR method and proposed an algorithm named Label Distribution Support Vector Regression (LDSVR) for pre-release prediction of movies [8]. Then label distribution was extended to multivariate label distribution by Geng and Xia [9]. They proposed two algorithms based on the Jeffery divergence for head pose estimation. LDL can also be adapted to crowd counting in public video surveillance [10] for the fact that the crowd image contains a similar number of people showing similar features. Xing et al. used a method called Logistic Boosting Regression (LogitBoost) [11], a combination of the boosting method and the logistic regression [12] to learn a general LDL model family [13]. Inspired by differentiable decision trees [14], Shen et al. proposed an end-to-end strategy label distribution learning forests (LDLFs) [15].

In previous studies, the most common learning approach of LDL is learning it by optimizing an energy function based on the maximum entropy model [3, 6, 7]. Such an approach is limited for that the exponential part of this model restricts the generality. In order to avoid making this assumption, Those papers [13, 10] extend the existing learning algorithms by boosting and support vector regression. To break the limitation in representation learning, LDLFs [15] was proposed to learn deep features in an end-to-end manner. Although considerable research has been devoted to optimizing strategy or feature representation, rather less attention has been paid to the connection between different classes which is quite common in real-world applications. For example, in facial expression recognition, some basic emotions often appear together such as disgust and fear, and some often conflict with each other such as happiness and sadness. This phenomenon of combining classes is also common in natural scene recognition. Sky and cloud often appear together, but rarely do we see both snow and desert in one image. So structured prediction is more applicable to solving this problem by considering the output as an integral whole. Therefore, a novel approach called structured random forest (StructRF) is proposed in this paper. It combines random forest and structured prediction for LDL. There are many structured prediction algorithms, among which the random forest is proved to be simple and effective. Constructing a decision tree is all about finding the split decision that returns the highest information gain. The information gain measures the level of impurity in a group. But the impurity of label distributions cannot be measured. In order to break down the barriers, training instances are clustered into two distinct groups based on label distributions at each split node of decision trees. Then, the instances are attached to the same cluster label if they are clustered to the same group. In this way, the information gain of each candidate threshold can be obtained directly. When a node reaches the condition of generating a leaf node of the tree, the mean of all instances label distributions is the prediction of this leaf node. So when a testing instance reaches a leaf node, the predicted distribution of one tree is formed directly, and the diversity of random forest can help to avoid overfitting.

Another finding of this paper is that the calculating procedure during training process of the decision tree is redundant. In traditional methods, the information gain at each candidate threshold needs to be calculated. But the information gain has the feature of smoothly varying at adjacent candidate thresholds. In other words, when the information gain of a threshold is small, it tends to be still small at nearby thresholds. Therefore, there is no need to compute the information gain at every candidate threshold. In light of this discovery, a fast training method with an adaptive variable step is proposed.

Many techniques have been developed to speed up decision tree learning, such as designing a fast tree-growing algorithm, parallelization, and data partitioning. Among them, a large amount of research work has been done on reducing the computational cost, such as SLIQ [16], SPRINT [17], or Rainforest [18]. Apparently, developing a fast tree-growing algorithm is more essential. There are basically two approaches to designing a fast tree-growing algorithm: searching in a restricted model space and using a powerful search heuristic. Different from above works, our method aim to removing redundant computation by adapting the step.

The first contribution of this paper is exploring the potential value of the connection between different classes and making full use of it which results in a better performance. The structured random forest model is fast for training, easily parallelized and can store any structured output in leaf nodes. Another contribution is that an adaptive variable step method is proposed to speed up the training rate without any performance loss. This method is not only adapted to our approach but also suitable for the most decision tree based models.

The rest of the paper is organized as follows. First, a brief review and discussion of the related work on LDL are given in Section 2. The algorithm structured random forest for LDL and the adaptive variable step method are proposed in Section 3. Then the details of the experiments are reported in Section 4. Finally, the conclusions are drawn in Section 5.

2. Related Work

2.1. Formulation of LDL

Suppose that we are given a set of m samples $\mathcal{S} = \{(\mathbf{x}_1, D_1), ..., (\mathbf{x}_m, D_m)\}$ for training, where $\mathbf{x} = \{x_1, x_2, ..., x_q\}$ is a q-dimentional vector. For each instance $\mathbf{x}_i$, its label distribution is denoted by $D_i = \{d_{\mathbf{x}_i}^{y_1}, d_{\mathbf{x}_i}^{y_2}, ..., d_{\mathbf{x}_i}^{y_c}\}$ where $y \in \mathcal{Y}$, $\mathcal{Y} = \{y_1, y_2, ..., y_c\}$ denotes the complete set of labels. The constant c is the number of possible label values and $d_{\mathbf{x}_i}^{y_j}$ is the description degree of the particular j-th label y_j to the particular i-th instance $\mathbf{x}_i$. According to the definition, each description degree should satisfy the constraints $d_{\mathbf{x}}^{y} \in [0, 1]$ and $\sum_y d_{\mathbf{x}}^{y} = 1$. For a better understanding, it is necessary to distinguish the label distributions clearly from the possible label. The description degree is represented sometimes by the form of conditional probability, i.e., $P(y|\mathbf{x})$. It is not the probability that the class y can label the instance correctly but the degree that the class y describes the instance.

2.2. Structured prediction

2.2.1. Random forest

To avoid the overfitting of decision trees [19], the random forest model is built. Random Forest is an ensemble learning method for classification, regression, and other tasks. The first algorithm for random forests was created by Tin Kam Ho [20] using the random subspace method [21]. Then the model has been wildly applied.

During training, randomness is injected into trees in order to give the trained trees better generalization power. Given a training set, random forest repeatedly selects a random subset of samples and a random subset of the features with the replacement of the training set and fits trees to these samples. The standard choice of ensemble model includes majority voting for classification and averaging for regression.

The random forest model offers many advantages such as extremely fast for training and classification, being easily parallelized, tending not to over-fit and robust to noisy labels. So it has been widely used in many fields [22, 23, 24].

2.2.2. Structured random forest

Structured forest model has been used in many computer vision applications such as semantic image labelling [25], object counting [26] and edge detection [27].

The structured random forest we applied in this paper is slightly different from the traditional structured prediction model. On the one hand, only the output space is deemed to be structured, and the input space is standard, on the other hand, the scoring function and efficient optimization procedure are required on common approaches for structured prediction. The inference used in the structured random forest, by contrast, is more general and straightforward.

3. Structured Random Forest for LDL

3.1. Training a single tree

Firstly, the StructRF model is trained on a training dataset $\mathcal{S} = \{X, D\} = \{(\mathbf{x}_1, D_1), (\mathbf{x}_2, D_2), ..., (\mathbf{x}_m, D_m)\}$.

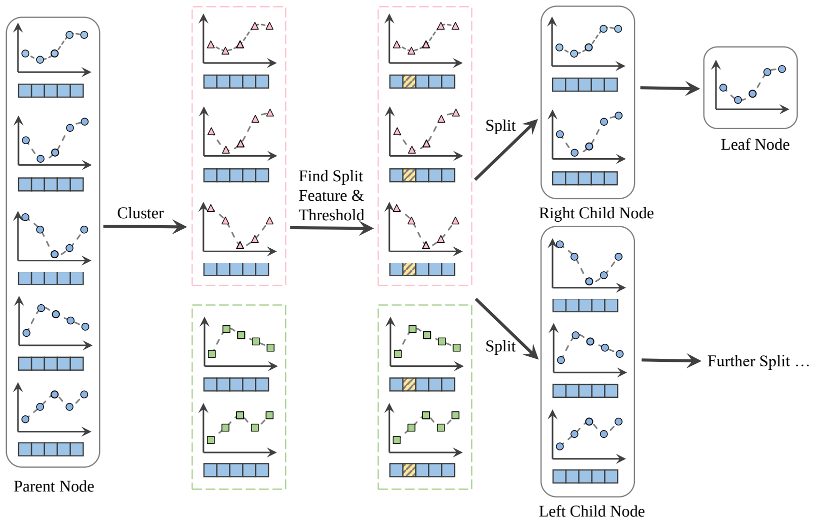

Figure 2: **The unit of the training process.** There are five samples at this parent node, features are denoted by blue rectangles, and scatter plots represent the label distributions. All samples at this parent node are clustered into two disjoint groups which are colored in pink and green respectively. The samples in the same group are assigned to the same cluster label. Based on cluster labels, the information gain can be attained. Then those samples are split by the split feature and threshold that can get the maximum information gain. The right child node is generated as a leaf node as it reaches preset condition, and the left child node will be further split.

A decision tree consists of a set of split nodes $\mathbf{n} \in \mathcal{N}^{Split}$ and leaf nodes $\mathbf{l} \in \mathcal{N}^{Leaf}$. Each split node can be viewed as a parent node. The nodes split from it are its child nodes. The unit of training process is shown in Fig. 2. At a given parent node $\mathbf{n}$, $\{X, D\}$ denotes of all the n training samples of it. Those samples will be split into two subsets $\{X^-, D^-\}$ and $\{X^+, D^+\}$, in which "$-$" and "$+$" respectively denote the left and right child node of the parent node. In order to formulate the split decision, we need to calculate a single feature whose index is denoted by f and a threshold is denoted by t, in which $f \in \{1, 2, .., q\}$. Since there are n samples in the current node, there are n numbers in the f-th dimension. Without the loss of generality, we suppose values of the n numbers are different. Then values are sorted in ascending order, denoted by $\left\{a_f^1, a_f^2, ..., a_f^n\right\}$. $T_f = \left\{\frac{a_f^i + a_f^{i+1}}{2} | 1 \leq i \leq n-1\right\}$ is the candidate set of the threshold (i.e., $t \in T_f$). The partition performs in the following way:

$$\begin{cases} \forall \mathbf{x} \in X : \mathbf{x} \in X_{f,t}^{-} \Leftrightarrow x_f < t \\ \forall \mathbf{x} \in X : \mathbf{x} \in X_{f,t}^{+} \Leftrightarrow x_f > t \end{cases} \tag{1}$$

In the traditional random forest algorithm, f and t depend on the maximum information gain that each candidate threshold can achieve. In order to calculate the maximum information gain easily, the samples $\{X, D\}$ are divided into two clusters by k-means at each split node. After that all the instances are relabelled as $\{X, C\} = \{(\mathbf{x}_1, c_1), ..., (\mathbf{x}_n, c_n)\}$, in which $c_i \in \{0, 1\}$. In our method, Shannon entropy $Ent(X) = -\sum_{c \in \{0,1\}} p_c log(p_c)$ is used to measure the impurity, where p_c is the fraction of samples in X with label c. The information gain for the f-th dimension feature split on t is:

$$Gain(X, X_{f,t}^{-}, X_{f,t}^{+}) = Ent(X) - \sum_{\lambda \in \{-,+\}} \frac{\left|X_{f,t}^{\lambda}\right|}{|X|} Ent(X_{f,t}^{\lambda}) \tag{2}$$

where $X_{f,t}^{\lambda}$, $\lambda \in \{-, +\}$ denotes the two parts split from X based on f, t and Eq. (1). Since the information gain can be calculated, the best choices of f and t will be obtained as follows:

$$f, t = \underset{f \in \{1,2,...,q\}, t \in T_f}{\arg\max} \; Gain(X, X_{f,t}^{-}, X_{f,t}^{+}) \tag{3}$$

The process of training a single tree is shown in Algorithm 1, and the overall algorithm is presented in Algorithm 2.

3.2. Merge trees to a forest

Algorithm 1: Generate a Single Tree

```
Input   : The training set: $(X, D)$
Default: The maximum depth of tree: $Dep_{max}$;
         The minimum number of samples allowed to be split: $Num_{min}$;
Output: Tree structure
Function GenerateTree (X,D)
    $Num$ = number of samples in the current node;
    $Dep$ = depth of the current node;
    if $(Num < Num_{min})||(Dep > Dep_{max})$ then
        Generate a leaf node $Leaf = mean(D)$;
        return
    end
    Cluster the label set $D$ into 2 clusters by k-means;
    Relabel $\mathbf{x}_i \in X$ with $c_i \in \{0, 1\}$;
    $f, t = BestSplit(X, C)$ as Eq. (3);
    Split $(X, D)$ into $(X^+_{f,t}, D^+_{f,t})$ and $(X^-_{f,t}, D^-_{f,t})$ based on Eq. (1);
    $GenerateTree(X^+_{f,t}, D^+_{f,t})$;
    $GenerateTree(X^-_{f,t}, D^-_{f,t})$;
    return

Function BestSplit (X,C)
    $g_{max} = 0, f_{best} = 1, t_{best} = 0$;
    for all feature dimensions $f = 1, 2, ..., q$ do
        for all possible split thresholds $t \in T_f$ do
            Split $X$ into $X^+_{f,t}, X^-_{f,t}$ based on $f$, $t$ and Eq. (1);
            $g = Gain(X, X^+_{f,t}, X^-_{f,t})$ as Eq. (2);
            if $g > g_{max}$ then
                $g_{max} = g$; $f_{best} = f$; $t_{best} = t$;
            end
        end
    end
    return $f_{best}$, $t_{best}$
```

Algorithm 2: Structured Random Forest Regression

```
Input   : The training set $(X, D)$
Default: The number of decision trees $T$
Output: The learned structRF
1 for $\mu = 1, 2, ..., T$ do
2     Randomly sample, with replacement, $(0.8 \times m)$ training examples
        from $(X, D)$ to form a new training set $(X_\mu, D_\mu)$;
3     $tree_\mu = GenerateTree(X_\mu, D_\mu)$;
4 end
5 Merge all the trees into a learned structRF;
6 return
```

Whether random forests can achieve robust results depends on how we combine the output of multiple decorrelated trees. We can consider it as a regression problem. Then the averaging can be used to get the ensemble output:

$$H(\mathbf{x}, D) = \frac{1}{T} \sum_{i=1}^{T} tree_i(\mathbf{x}_i, D_i) \tag{4}$$

where $(\mathbf{x}_i, D_i)$ is the subset of training data for the i-th decision tree.

3.3. Testing

Given a test image, it fist pass a feature extractor to get a feature x. Then the feature is fed to each decision tree to get n primary predictions. The final prediction is gotten by merging all the primary predictions.

3.4. Adaptive variable step method

During the process of selecting the best split feature f and threshold t, all the candidates are traversed in the traditional method. To avoid unnecessary computation, the step is variable in our method instead of always being set to 1. Since the information gain varies smoothly at adjacent candidate thresholds, the step should be short when the current information gain is comparatively large

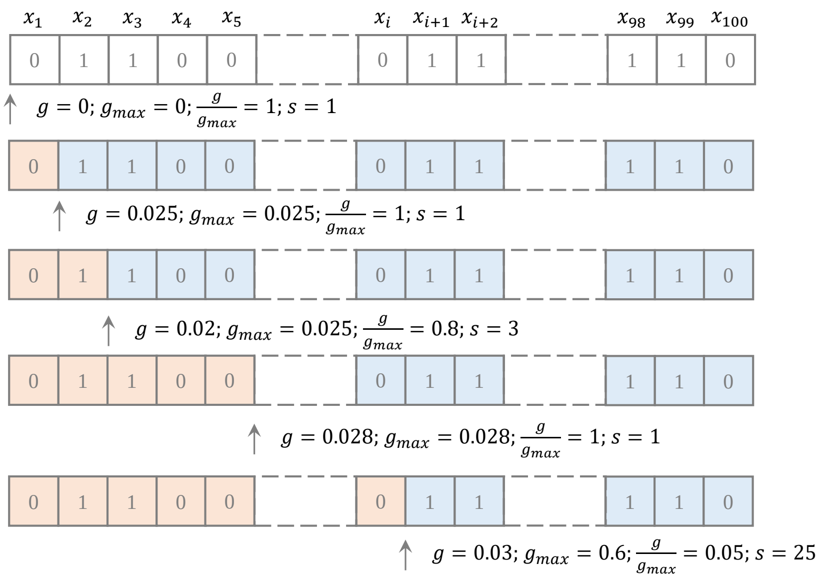
x_1 x_2 x_3 x_4 x_5 x_i x_{i+1} x_{i+2} x_{98} x_{99} x_{100}
0 1 1 0 0 0 1 1 1 1 0
$g = 0; g_{max} = 0; \frac{g}{g_{max}} = 1; s = 1$
$g = 0.025; g_{max} = 0.025; \frac{g}{g_{max}} = 1; s = 1$
$g = 0.02; g_{max} = 0.025; \frac{g}{g_{max}} = 0.8; s = 3$
$g = 0.028; g_{max} = 0.028; \frac{g}{g_{max}} = 1; s = 1$
$g = 0.03; g_{max} = 0.6; \frac{g}{g_{max}} = 0.05; s = 25$

Figure 3: **Adaptive variable step method.** There are 100 samples in ascending order on a particular feature dimension. Step s is initialized to 1 at first and the highest information gain g_{max} is set to 0. At first, g_{max} shows quite near to current information gain g, so the step s is relatively small. But it may be larger in the middle if relatively high information gain has been gotten in the previous process and the current gain g decreases to a comparatively low level.

with the best information gain, and vice versa. Suppose there are N examples, g is the current information gain and g_{max} is the current best information gain. The next step s is:

$$s = \frac{\alpha N}{1 + e^{\beta(g/g_{max} - 0.5)}} \tag{5}$$

where α and β are two constants control the intensity of the variation. In this paper, α is set to 0.25, and β is set to 8.

A sample graph is shown in Fig. 3. The adaptive variable step method makes the training process slow-moving when there may be the best split threshold, and takes a big step forward if the current information gain g is in a comparatively low level. That is the key why the method can speed up training with almost no loss of performance.

4. Experiments

4.1. Datasets

Table 1: **Statistics of the 15 datasets used in the experiments.**

No.	Dataset	Examples(n)	Features(q)	Labels(c)
1	Yeast-alpha	2,465	24	18
2	Yeast-cdc	2,465	24	15
3	Yeast-elu	2,465	24	14
4	Yeast-diau	2,465	24	7
5	Yeast-heat	2,465	24	6
6	Yeast-spo	2,465	24	6
7	Yeast-cold	2,465	24	4
8	Yeast-dtt	2,465	24	4
9	Yeast-spo5	2,465	24	3
10	Yeast-spoem	2,465	24	2
11	Human Gene	30,542	36	68
12	Natural Scene	2,000	294	9
13	s-JAFFE	213	243	6
14	s-BU_3DFE	2,500	243	6
15	Movie	7,755	1,869	5

There are 15 real-world datasets employed in the experiments in total. The statistics are shown in Table 1. The datasets from the first to the tenth are collected from ten biological experiments on the budding yeast *Saccharomyces cerevisiae*[28]. Each dataset includes 2,465 yeast genes, and an associated phylogenetic profile vector of the length 24 is utilized to represent each gene. During one biological experiment, the gene expression level is usually disparate at each discrete time point. So the labels correspond to the time point, and the label distributions are measured by the description degrees of all the labels.

The eleventh dataset *Human Gene* is much larger than the other datasets used in this experiment. This dataset is collected from the biological research on the relationship between human genes and diseases. Each of the 30,542 human genes is represented by the 36 numerical descriptors for a gene sequence proposed in[29] and 68 different diseases are corresponding to each label. The gene expression level of different diseases provides a measure of the description degree of the label for every human gene. Then the description degrees constitute a label distribution.

The twelfth dataset *Natural Scene* is collected from 2,000 natural scene images that are ranked inconsistently by ten human rankers. Each image is represented by a 294-dimensional feature vector extracted by [30] and is associated with a multi-label selected from 9 possible labels, i.e., sun, sky, water, cloud, mountain, snow, desert, building, and plant. Then rankers are required to rank the relevant labels in descending order of relevance. As expected, the rankings for the same image from different rankers are highly inconsistent. So, a nonlinear programming process[31] is applied to get the label distribution.

The thirteenth and fourteenth datasets *JAFFE* and *BU_3DFE* are two widely used facial expression image datasets. There are 213 gray-scale expression images in JAFFE dataset while BU_3DFE contains 2,500 facial expression images. The images in JAFFE are scored by 60 persons on the six primary emotion labels with a 5-level scale, i.e., fear, disgust, happiness, anger, sadness, surprise, and the images in BU_3DFE are scored by 23 persons in the same way in JAFFE. Both two datasets are represented by a 243-dimensional feature vector extracted by the method of Local Binary Patterns (LBP)[32]. The score for each emotion is regarded as the description degree, and the description degrees (normalized gene expression level) of all the six emotions constitute a label distribution for a particular facial expression image.

Finally, the fifteenth dataset *Movie* includes 7,755 movies. There are 54,243,292 ratings from 478,656 different users on a scale from 1 to 5 integral stars from Netflix. The percentage of each rating level is regarded as the label distribution. There are numeric and categorical attributes in the dataset such as genre, director, country, year, budget and so on. After transforming the categorical attributes into binary vectors, the final feature vector of each movie is of 1,869-dimensional.

4.2. Experimental Setup

Several parameter configurations are tested and the parameter setting with the best average performance is selected by 10-fold cross validation. In detail, the dataset is split into training, test and validation subsets, yielding an 8:1:1 ratio. Then the model is trained on the train set and tested on the validation set to select the best parameters. After that, the validation set is merged into the training set. The model is retrained with the selected parameters setting on the updated training set and tested on the test set. This procedure is repeated in 10 times (each time with different train set) and the average performance is recorded.

4.3. Measures

Table 2: **Evaluation measure for LDL algorithms.**

Name	Formula
Chebyshev(Cheby)↓	$Dis\left(D,\hat{D}\right) = max_j \left\|d_j - \hat{d}_j\right\|$
Clark↓	$Dis\left(D,\hat{D}\right) = \sqrt{\sum_{j=1}^{c} \frac{(d_j-\hat{d}_j)^2}{(d_j+\hat{d}_j)^2}}$
Canberra(Can)↓	$Dis\left(D,\hat{D}\right) = \sum_{j=1}^{c} \frac{\|d_j-\hat{d}_j\|^2}{d_j+\hat{d}_j}$
Kullback-Leibler(KL)↓	$Dis\left(D,\hat{D}\right) = \sum_{j=1}^{c} d_j \ln \frac{d_j}{\hat{d}_j}$
Cosine(Cos)↑	$Sim\left(D,\hat{D}\right) = \frac{\sum_{j=1}^{c} d_j \hat{d}_j}{\sqrt{\sum_{j=1}^{c} {d_j}^2}\sqrt{\sum_{j=1}^{c} \hat{d}_j^2}}$
Intersection(Inter)↑	$Sim\left(D,\hat{D}\right) = \sum_{j=1}^{c} min\left((d_j, \hat{d}_j\right)$

Six measurements are selected following four principles[3], i.e., Chebyshev Distance, Clark Distance, Canberra Metric, Kullback-Leibler Divergence, Cosine Coefficient and Intersection Similarity as shown in Table 2. These measures come from a different syntactic family summarized in the paper[33] and are relatively widely used in the related areas. Thus, they are believed to represent a good chance to reflect different aspects of those algorithms.

4.4. Results on LDL

First of all, the running time on a modern 14-core Intel Xeon E5-2683 v3 server CPU (35M Cache, 2.00 GHz) of all the algorithms is given in Table 3.

Table 3: **Running time(s).** *With the adaptive variable step method. T_a:training time. T_e:test time. (1)Yeast-alpha. (11)Human Gene. (12)Natural Scene. (13)s-JAFFE. (14)s-BU_3DFE. (15)Movie.

Data	SA-IIS		SA-BFGS		RF		RF*		StructRF		StructRF*	
	T_a	T_e	T_a	T_e	T_a	T_e	T_a	T_e	T_a	T_e	T_a	T_e
(1)	244	0.060	22	0.009	1.6	0.006	1.2	0.006	5.0	0.028	4.5	0.029
(11)	1818	0.011	367	0.014	9.9	0.047	7.6	0.054	340	0.201	304	0.197
(12)	1714	0.001	82	0.001	8.5	0.006	6.6	0.007	10.1	0.291	6.5	0.020
(13)	660	0.001	93	0.002	0.7	0.003	0.5	0.003	1.3	0.139	0.7	0.006
(14)	755	0.002	120	0.005	6.5	0.007	5.1	0.008	10.2	0.029	9.2	0.029
(15)	5422	0.003	455	0.008	121	0.024	91	0.025	145.6	0.113	129.3	0.112

There are six representative datasets. Six algorithms for comparisons are implemented in Matlab. The maximum step in SA-BFGS is 300 and in SA-IIS is 200. The numbers of trees in the random forest (RF) and StructRF are 50. In RF, the task is treated as a single label problem in training. In testing, the class with the maximum is taken as the label, and the probability distributions of each class in the leaf nodes are assumed as the predictions of the label distribution. Since SA-BFGS and SA-IIS share the same parametric model, the test time is analogous. And we can see that the training speed of SA-BFGS is much faster than SA-IIS, which is consistent with the remarks in the paper [34]. RF and StructRF are quicker than the other two algorithms that need to give credit to the parallelism property. Compared with StructRF, RF is much faster. This occurs because there is no cluster computation in RF. This also explains why it is time-consuming of StructRF on *Human Gene*. It takes quite a while to update cluster centers when handing large volumetric datasets. And with the adaptive variable step method, both RF* and StructRF* have a noticeable improvement in training speed. Since we must wait until the predictions of all the 50 trees are calculated to get the final output, which is a limitation of the ensemble method, so the test time of StructRF is relatively long. Nonetheless, each tree's predictions are independent of each other and the testing time for a single tree is similar than the other two algorithms. So if the condition for parallel computing is available, test time can be further reduced. Besides, the code is unoptimized Matlab code. The speed can also be improved by engineering means.

Table 4: **Results on Yeast-alpha.**

Measures	RF	SA-IIS	SA-BFGS	StructRF
Cheby↓	0.1657±0.0071	0.0148±0.001	0.0135±0.0009	**0.0133±0.0008**
Clark↓	1.6332±0.0534	0.2334±0.012	0.2108±0.0133	**0.2079±0.0134**
Can↓	5.5901±0.2407	0.7630±0.042	0.6848±0.0455	**0.6745±0.0461**
KL↓	0.3635±0.0301	0.0067±0.001	0.0062±0.0007	**0.0054±0.0007**
Cos↑	0.7373±0.0157	0.9934±0.001	0.9945±0.0007	**0.9947±0.0007**
Inter↑	0.6888±0.0137	0.9577±0.002	0.9620±0.0025	**0.9628±0.0025**

Table 5: **Results on Yeast-cdc.**

Measures	RF	SA-IIS	SA-BFGS	StructRF
Cheby↓	0.1287±0.004	0.0178±0.001	0.0163±0.0009	**0.0161±0.0010**
Clark↓	1.6482±0.0189	0.2352±0.012	0.2161±0.0138	**0.2139±0.0139**
Can↓	5.2399±0.0839	0.7088±0.036	0.6492±0.0416	**0.6403±0.0427**
KL↓	0.4108±0.0117	0.0082±0.001	0.0071±0.0009	**0.0068±0.0009**
Cos↑	0.7782±0.0077	0.9921±0.001	0.9931±0.0008	**0.9934±0.0008**
Inter↑	0.6787±0.0067	0.9531±0.002	0.9572±0.0027	**0.9579±0.0028**

Table 6: **Results on Yeast-elu.**

Measures	RF	SA-IIS	SA-BFGS	StructRF
Cheby↓	0.1262±0.0042	0.0178±0.001	0.0164±0.0006	**0.0160±0.0006**
Clark↓	1.0117±0.0458	0.2160±0.007	0.1992±0.0058	**0.1961±0.0058**
Can↓	3.032±0.1301	0.6392±0.019	0.5838±0.0172	**0.5756±0.0171**
KL↓	0.1709±0.0173	0.0073±0.0005	0.0063±0.0004	**0.0061±0.0004**
Cos↑	0.8486±0.0084	0.9929±0.0005	0.9939±0.0004	**0.9941±0.0004**
Inter↑	0.7804±0.0079	0.9547±0.001	0.9588±0.0012	**0.9593±0.0012**

The results are presented on Table 4 to Table 18. Because those algorithms are tested via ten-fold cross-validation, the performance is represented by "mean±standard deviation". In each table, we highlight the best performance by boldface. As it can be seen from those tables, the overall performance of each algorithm on all the six measures is consistent, i.e., StructRF > SA-BFGS > SA-IIS > RF. Both SA-IIS and SA-BFGS are represented by a maximum

Table 7: **Results on Yeast-diau.**

Measures	RF	SA-IIS	SA-BFGS	StructRF
Cheby↓	0.1981±0.0047	0.0386±0.001	0.0370±0.0015	**0.0358±0.0018**
Clark↓	1.3546±0.0134	0.209±0.007	0.2008±0.0082	**0.1941±0.0085**
Can↓	2.9506±0.0339	0.449±0.017	0.4309±0.0193	**0.4164±0.0201**
KL↓	0.6327±0.0246	0.014±0.001	0.0131±0.0011	**0.0124±0.0012**
Cos↑	0.7849±0.0065	0.987±0.001	0.9879±0.0011	**0.9884±0.0011**
Inter↑	0.6479±0.0058	0.938±0.002	0.9401±0.0028	**0.9421±0.0029**

Table 8: **Results on Yeast-heat.**

Measures	RF	SA-IIS	SA-BFGS	StructRF
Cheby↓	0.1653±0.005	0.0430±0.001	0.0423±0.0009	**0.0406±0.0009**
Clark↓	0.7145±0.0155	0.1881±0.003	0.1828±0.0032	**0.1764±0.0032**
Can↓	1.4808±0.0344	0.3772±0.005	0.3647±0.0067	**0.3526±0.0068**
KL↓	0.2088±0.0114	0.0133±0.0004	0.0127±0.0005	**0.0118±0.0005**
Cos↑	0.8743±0.0044	0.987±0.0004	0.9880±0.0005	**0.9887±0.0005**
Inter↑	0.7694±0.005	0.938±0.001	0.9401±0.0011	**0.9422±0.0011**

Table 9: **Results on Yeast-spo.**

Measures	RF	SA-IIS	SA-BFGS	StructRF
Cheby↓	0.2649±0.0087	0.0600±0.004	0.0584±0.0039	**0.0575±0.0036**
Clark↓	0.8367±0.0332	0.255±0.017	0.2504±0.0175	**0.2461±0.0154**
Can↓	1.7739±0.0777	0.523±0.034	0.5134±0.0355	**0.5044±0.0298**
KL↓	0.2725±0.0216	0.0254±0.003	0.0246±0.0031	**0.0240±0.0027**
Cos↑	0.7919±0.011	0.976±0.003	0.9769±0.0027	**0.9774±0.0023**
Inter↑	0.699±0.0115	0.914±0.005	0.9154±0.0056	**0.9170±0.0047**

Table 10: **Results on Yeast-cold.**

Measures	RF	SA-IIS	SA-BFGS	StructRF
Cheby↓	0.1948±0.0091	0.0530±0.002	0.0512±0.0018	**0.0498±0.0018**
Clark↓	0.4804±0.0171	0.144±0.005	0.1398±0.0059	**0.1361±0.0059**
Can↓	0.8348±0.0323	0.249±0.009	0.2406±0.0104	**0.2348±0.0107**
KL↓	0.1416±0.0103	0.013±0.001	0.0122±0.0012	**0.0118±0.0012**
Cos↑	0.8955±0.0074	0.988±0.001	0.9885±0.001	**0.9891±0.0010**
Inter↑	0.7934±0.0087	0.938±0.002	0.9407±0.0024	**0.9422±0.0025**

Table 11: **Results on Yeast-dtt.**

Measures	RF	SA-IIS	SA-BFGS	StructRF
Cheby↓	0.159±0.0037	0.0388±0.001	0.0361±0.0014	**0.0350±0.0014**
Clark↓	0.3959±0.0087	0.105±0.004	0.0984±0.0041	**0.0953±0.0038**
Can↓	0.7063±0.0175	0.181±0.005	0.1693±0.0065	**0.1636±0.0056**
KL↓	0.0965±0.0038	0.0070±0.001	0.0063±0.0007	**0.0059±0.0007**
Cos↑	0.921±0.0031	0.9933±0.0004	0.9940±0.0005	**0.9944±0.0005**
Inter↑	0.8248±0.0045	0.9552±0.001	0.9580±0.0015	**0.9597±0.0012**

Table 12: **Results on Yeast-spo5.**

Measures	RF	SA-IIS	SA-BFGS	StructRF
Cheby↓	0.1795±0.0067	0.0928±0.006	0.0914±0.0052	**0.0867±0.0048**
Clark↓	0.3747±0.0155	0.187±0.013	0.1842±0.0116	**0.1751±0.0103**
Can↓	0.575±0.0232	0.287±0.019	0.283±0.0169	**0.269±0.0154**
KL↓	0.1105±0.0079	0.03007±0.003	0.0293±0.0028	**0.0268±0.0023**
Cos↑	0.9187±0.005	0.9733±0.003	0.9741±0.0024	**0.9763±0.002**
Inter↑	0.8205±0.0067	0.9072±0.006	0.9086±0.0052	**0.9133±0.0048**

Table 13: **Results on Yeast-spoem.**

Measures	RF	SA-IIS	SA-BFGS	StructRF
Cheby↓	0.135±0.0096	0.0891±0.005	0.0868±0.0049	**0.0830±0.0042**
Clark↓	0.2068±0.0148	0.1321±0.007	0.1293±0.0073	**0.1240±0.006**
Can↓	0.2851±0.0203	0.1840±0.010	0.1798±0.0101	**0.1723±0.0084**
KL↓	0.0624±0.0084	0.025±0.003	0.0245±0.0024	**0.0223±0.0021**
Cos↑	0.9519±0.0056	0.9780±0.002	0.979±0.002	**0.9806±0.0017**
Inter↑	0.8651±0.0096	0.9109±0.005	0.9132±0.0049	**0.9170±0.0042**

Table 14: **Results on Human Gene.**

Measures	RF	SA-IIS	SA-BFGS	StructRF
Cheby↓	0.0798±0.0067	0.0534±0.004	0.0533±0.0038	**0.0532±0.0039**
Clark↓	3.8951±0.1679	2.123±0.088	2.1111±0.0864	**2.1026±0.0912**
Can↓	27.282±1.1435	14.541±0.653	14.4532±0.6455	**14.387±0.6685**
KL↓	0.6146±0.0607	0.238±0.019	0.2365±0.0194	**0.2336±0.0196**
Cos↑	0.6488±0.0139	0.833±0.011	0.8343±0.0107	**0.8359±0.0116**
Inter↑	0.6052±0.0122	0.783±0.010	0.7842±0.0097	**0.7858±0.0103**

Table 15: **Results on Natural Scene.**

Measures	RF	SA-IIS	SA-BFGS	StructRF
Cheby↓	0.3477±0.0093	0.3411±0.017	0.322±0.017	**0.2651±0.0148**
Clark↓	2.5521±0.0151	2.461±0.025	2.411±0.023	**2.408±0.0251**
Can↓	7.1564±0.0628	6.765±0.104	6.620±0.097	**6.5025±0.1081**
KL↓	1.5477±0.2001	0.870±0.026	0.854±0.062	**0.6142±0.0292**
Cos↑	0.6102±0.011	0.698±0.008	0.710±0.017	**0.8026±0.0121**
Inter↑	0.4602±0.0095	0.487±0.012	0.548±0.017	**0.6045±0.0131**

Table 16: **Results on s-JAFFE.**

Measures	RF	SA-IIS	SA-BFGS	StructRF
Cheby↓	0.2019±0.0425	0.1175±0.015	0.1184±0.0145	**0.1047±0.0137**
Clark↓	0.8755±0.0783	0.419±0.034	0.4657±0.0633	**0.3709±0.0399**
Can↓	1.7105±0.1829	0.875±0.086	0.9565±0.1269	**0.7726±0.092**
KL↓	0.2977±0.0658	0.070±0.012	0.0861±0.0231	**0.0544±0.0117**
Cos↑	0.8277±0.0488	0.9340±0.012	0.9277±0.0113	**0.9483±0.0117**
Inter↑	0.7371±0.0373	0.851±0.016	0.8426±0.0152	**0.8690±0.0165**

Table 17: **Results on s-BU_3DFE.**

Measures	RF	SA-IIS	SA-BFGS	StructRF
Cheby↓	0.219±0.0165	0.1113±0.004	**0.1089±0.0051**	0.1123±0.0056
Clark↓	0.8688±0.0337	0.416±0.009	0.3676±0.0121	**0.3476±0.0166**
Can↓	1.8266±0.0654	0.934±0.022	0.7631±0.0239	**0.7451±0.0337**
KL↓	0.2907±0.0349	0.068±0.004	0.0603±0.0043	**0.0581±0.0052**
Cos↑	0.8182±0.0162	0.935±0.004	0.9413±0.0043	**0.9425±0.0052**
Inter↑	0.71±0.0123	0.862±0.004	0.8647±0.0046	**0.8661±0.0062**

Table 18: **Results on Movie.**

Measures	RF	SA-IIS	SA-BFGS	StructRF
Cheby↓	0.263±0.0112	0.1502±0.008	0.1317±0.0062	**0.1108±0.005**
Clark↓	1.0681±0.0524	0.591±0.028	0.5665±0.0246	**0.5042±0.0235**
Can↓	2.0538±0.1017	1.137±0.057	1.0948±0.0511	**0.9629±0.0452**
KL↓	0.4403±0.0302	0.137±0.013	0.1273±0.0109	**0.0921±0.0066**
Cos↑	0.8402±0.0069	0.905±0.008	0.9183±0.0063	**0.9393±0.0042**
Inter↑	0.6849±0.0048	0.800±0.010	0.8155±0.0083	**0.8421±0.0062**

entropy model, and the optimization process of SA-IIS is similar to Improved Iterative Scaling (IIS) while SA-BFGS is optimized by effective quasi-Newton optimization. StructRF performs better than the other two algorithms because it makes full use of the inherent structure. The prediction of StrucRF is derived from the original training data as a whole, rather than a separate calculation of the conditional probability of each class.

The performance of StructRF on the Chebyshev distance is slightly weak compared with the other measures. This might result from the fact that the Chebyshev distance only cares about the worst match over the whole label distribution, and the random forest focuses on the overall structure D_i instead of each description degree $d_{x_i}^{y_j}$.

StructRF shows a significant advantage over Clark Distance and Canberra Metric. It is caused by the definition of these two measures. As noticed in [35], the Clark distance and Canberra metric are sensitive to small changes near zero. Nevertheless, the prediction of StructRF in each leaf node comes from the mean value of all the samples' distribution that stored in this leaf node which results in fewer near-zero outputs than SA-IIS and SA-BFGS.

The performance is improved with different degrees on StructRF for different datasets, among which the most obvious is for *JAFFE* that can be seen from Table 16. *JAFFE* has only 213 samples. It seems to be insufficient compared with the model size of SA-IIS and SA-BFGS. But forest model doesn't have the problem. The depth of each tree is connected with the data scale. The capability of self-adaptation makes it performs well no matter how large or how small the dataset is. And the variety of trees reduces the risk of overfitting.

In addition, the performance of structured random forest is also affected by the number of decision trees T, sampling ratio and max depth of each tree. Fig. 4 shows the influence of different experiment parameters to the results. We can see from Fig. 4(a) that the Chebyshev distance decreases as the max depth and sampling ratio increase, but when the max depth increases to 20 and sampling ratio increases to 0.8, the distance does not decrease significantly. So in the experiment, max depth is set to 20 and sampling ratio is set to 0.8. From

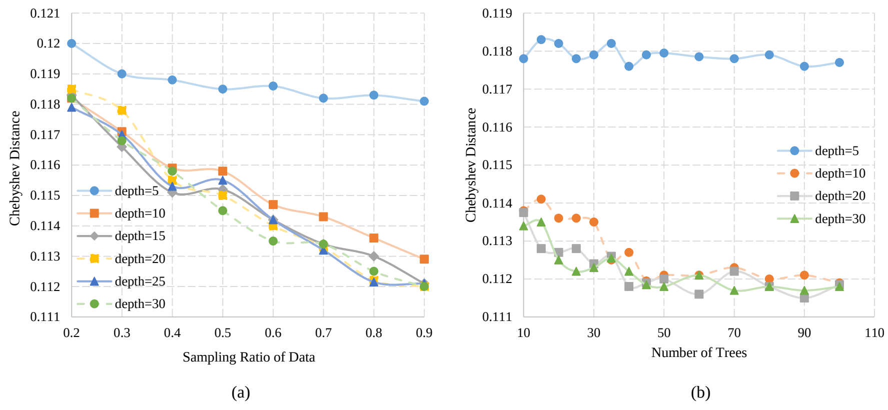

Figure 4: **The influence of experiment parameters to the results.** (a) The Chebyshev distance on dataset s-BU_3DFE with different maxdepth and different sampling ratio when there are 20 trees. (b) The Chebyshev distance on dataset s-BU_3DFE with different maxdepth and different number of trees when the sampling ratio is 0.8.

Fig. 4(b), we can observe that when the depth is large enough, the three curves have very similar characteristic. The performance of StructRF gets saturated when the number of trees reaches 50. When the depth is too small, e.g., 5, the performance is poor due to under-fitting.

Table 19: **Comparison with other LDL methods on dataset Movie.**

Method	KL↓	Euclidean↓	Sørensen↓
structRF	0.0921±0.0040	0.1726±0.0069	0.1712±0.0068
AOSO-LDLogitBoost	**0.0855±0.0037**	**0.1547±0.0030**	**0.1521±0.0032**
LDLogitBoost	0.0900±0.0038	0.1585±0.0032	0.1552±0.0031
Method	SquaredX^2 ↓	Fidelity↑	Intersection↑
structRF	**0.0548±0.0023**	0.9763±0.0019	0.8421±0.0039
AOSO-LDLogitBoost	0.0837±0.0034	**0.9778±0.0009**	**0.8478±0.0031**
LDLogitBoost	0.0875±0.0034	0.9767±0.0009	0.8448±0.0031

In Table 19, structRF is compared with the recent LDLogitBoost methods on Movie dataset. The results show that structRF method closely matches the performance of LDLogitBoost but has slightly worse accuracy. Both structRF and LDLogitBoost using decision trees are non-linear functions but have different learning strategy. LDLogitBoost utilizes end-to-end optimization to obtain better accuracies. It requires to optimize according to different distance measurements separately, i.e., for the six measurements, LDLogitBoost is trained for six times. While the proposed structRF is more general and only trained for one time. As for why the KL and Euclidean measures give a worse performance, it is probably due to the distribution of training and testing data in the Movie dataset, since we do not optimize according to a specific distance measure.

4.5. Results of the adaptive variable step method.

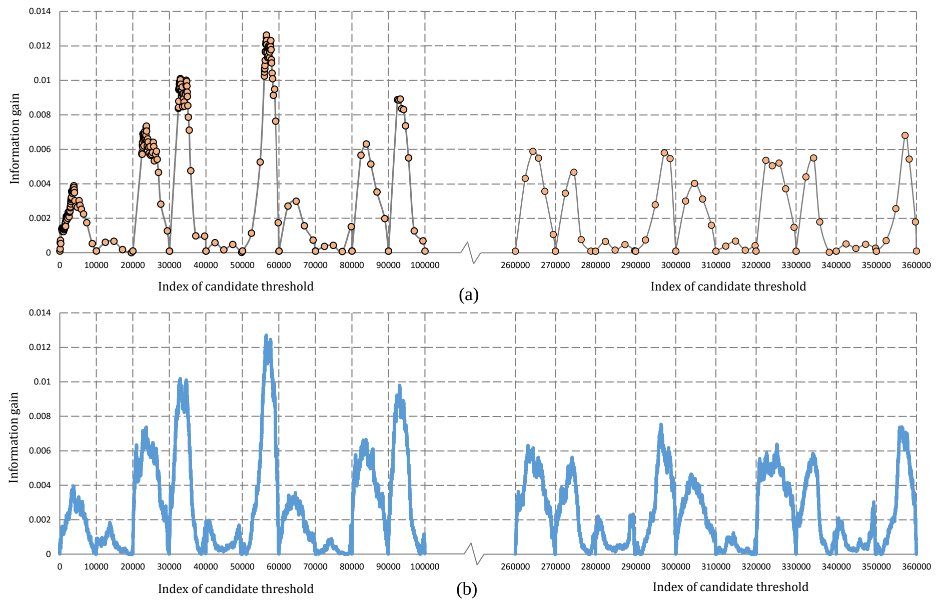

Figure 5: **The process of information gain computation.** (a) Adaptive variable step method. (b) Existing method.

To get more facts about the adaptive variable step method, an additional experiment is made. The task is to find out the best split decision that can make the information gain as higher as possible. The dataset is a subset of *Human Gene*. It contains only 10000 samples. The feature dimension of each sample is

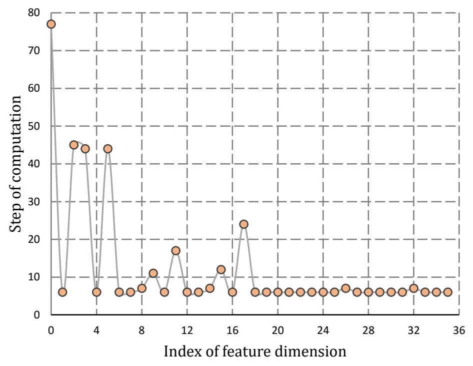

Figure 6: **The number of computations for each feature dimension.**

36. So there are $36 \times 10000 = 360000$ candidate thresholds at all. Fig. 5 shows the detailed process. Fig. 5(b) is the the existing method (step $s = 1$) which computes the information gain 360000 times. What we can see in Fig. 5(a) is that the computation is dense when it close to a peak and sparse when the current information gain is relatively low. After it passes the global maximum information gain, the step still keeps long nearby lower peaks. Fig. 6 shows the numbers of computation for each feature dimension. It shows the advantages of the adaptive variable step method quantificationally. The number of computation times reduced to only 447. Such expenditure reduction scarcely affects the performance. It can be proved from Table 20. The adaptive variable step method is ten times faster than the existing method, but the best information gains they find are very close.

Table 20: **The results of the variable step and the existing methods.**

Method	Feature dimension	Split threshold	Information gain	Time(ms)
Invariable step	3	0.5034	0.0085	25.36
Existing method	3	0.5038	0.0082	2.34

As shown in Table 21, the difference between StructRF and StructRF with the adaptive variable step method (StructRF*) is quite modest. It means that the adaptive variable step method can speed up the training process without

Table 21: **The difference between StructRF and StructRF*.** *With the adaptive variable step method.

Method	Cheby↓	Clark↓	Can↓	KL↓	Cos↑	Inter↑
Yeast-alpha	0	0.0001	0.0002	0	0.0001	-0.0002
Human Gene	0.0001	0	0	0.0001	-0.0002	-0.0001
Natural Scene	-0.0001	0	-0.0002	0	0.0003	0.0002
s-JAFFE	0	0	0.0001	0	-0.0001	0
s-BU_3DFE	0	0.0001	0	0	0	0
Movie	0	0	0	0	0.0001	0

degrading the performance.

5. Conclusion

In this paper, the label distribution learning problem is considered as a structured prediction problem and a novel algorithm StructRF is proposed. StructRF takes full account of the connection between the different classes and maps the distributions into a discrete space at every split node which is entirely different from other LDL methods. Being examined on various LDL benchmarks, StructRF has obtained superior accuracy compared with traditional LDL solvers. The encouraging performance suggests that our solution for LDL is a promising direction to be explored. And an adaptive variable step method for decision tree models is proved to be general and efficient to speed up the raining process without any loss of performance.

Acknowledgement

We thank anonymous reviewers for their very useful comments and suggestions. This work was supported in part by the National Natural Science Foundation of China under Grant 61733007 and 61572207.

References

[1] L. Yin, X. Wei, Y. Sun, J. Wang, M. J. Rosato, A 3d facial expression database for facial behavior research, in: International Conference on Automatic Face and Gesture Recognition, 2006, pp. 211–216.

[2] G. Tsoumakas, I. Katakis, Multi-label classification: An overview, International Journal of Data Warehousing & Mining 3 (3) (2010) 1–13.

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

[4] A. L. Berger, V. J. D. Pietra, S. A. D. Pietra, A maximum entropy approach to natural language processing, Computational linguistics 22 (1) (1996) 39–71.

[5] S. Della Pietra, V. Della Pietra, J. Lafferty, Inducing features of random fields, IEEE transactions on pattern analysis and machine intelligence 19 (4) (1997) 380–393.

[6] X. Geng, Q. Wang, Y. Xia, Facial age estimation by adaptive label distribution learning, in: International Conference on Pattern Recognition, 2014, pp. 4465–4470.

[7] X. Geng, Q. Wang, Y. Xia, Facial age estimation by adaptive label distribution learning, in: International Conference on Pattern Recognition, 2014, pp. 4465–4470.

[8] G. Xin, H. Peng, Pre-release prediction of crowd opinion on movies by label distribution learning, Proceedings of the International Joint Conference on Artificial Intelligence (2015) 3511–3517.

[9] G. Xin, X. Yu, Head pose estimation based on multivariate label distribution, in: Computer Vision and Pattern Recognition, 2014, pp. 1837–1842.

[10] Z. Zhang, M. Wang, X. Geng, Crowd counting in public video surveillance by label distribution learning, Neurocomputing 166 (C) (2015) 151–163.

[11] J. Friedman, T. Hastie, R. Tibshirani, et al., Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors), The annals of statistics 28 (2) (2000) 337–407.

[12] M. Collins, R. E. Schapire, Y. Singer, Logistic regression, adaboost and bregman distances, Machine Learning 48 (1-3) (2002) 253–285.

[13] C. Xing, X. Geng, H. Xue, Logistic boosting regression for label distribution learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4489–4497.

[14] P. Kontschieder, M. Fiterau, A. Criminisi, S. Rota Bulo, Deep neural decision forests, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1467–1475.

[15] W. Shen, K. Zhao, Y. Guo, A. Yuille, Label distribution learning forests, arXiv preprint arXiv:1702.06086.

[16] M. Mehta, R. Agrawal, J. Rissanen, Sliq: A fast scalable classifier for data mining, in: International Conference on Extending Database Technology, Springer, 1996, pp. 18–32.

[17] J. Shafer, R. Agrawal, M. Mehta, Sprint: A scalable parallel classi er for data mining, in: Proc. 1996 Int. Conf. Very Large Data Bases, Citeseer, 1996, pp. 544–555.

[18] J. Gehrke, R. Ramakrishnan, V. Ganti, Rainforesta framework for fast decision tree construction of large datasets, Data Mining and Knowledge Discovery 4 (2-3) (2000) 127–162.

[19] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer,, 2001.

[20] T. K. Ho, Random decision forests, in: International Conference on Document Analysis and Recognition, 1995, pp. 278–282 vol.1.

[21] T. K. Ho, The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis & Machine Intelligence 20 (8) (1998) 832–844.

[22] C. Strobl, A.-L. Boulesteix, A. Zeileis, T. Hothorn, Bias in random forest variable importance measures: Illustrations, sources and a solution, BMC bioinformatics 8 (1) (2007) 25.

[23] Y. Qi, J. Klein-Seetharaman, Z. Bar-Joseph, Random forest similarity for protein-protein interaction prediction from multiple sources, in: Biocomputing 2005, World Scientific, 2005, pp. 531–542.

[24] M. Khalilia, S. Chakraborty, M. Popescu, Predicting disease risks from highly imbalanced data using random forest, BMC medical informatics and decision making 11 (1) (2011) 51.

[25] P. Kontschieder, S. R. Bulo, H. Bischof, M. Pelillo, Structured class-labels in random forests for semantic image labelling, Proceedings 24 (4) (2011) 2190–2197.

[26] L. Fiaschi, U. Köthe, R. Nair, F. A. Hamprecht, Learning to count with regression forest and structured labels, in: Pattern Recognition (ICPR), 2012 21st International Conference on, IEEE, 2012, pp. 2685–2688.

[27] P. Dollár, C. L. Zitnick, Structured forests for fast edge detection, in: IEEE International Conference on Computer Vision, 2013, pp. 1841–1848.

[28] M. B. Eisen, P. T. Spellman, P. O. Brown, D. Botstein, Botstein d: Cluster analysis and display of genome-wide expression patterns, Proceedings of the National Academy of Sciences of the United States of America 95 (25) (1998) 14863–14868.

[29] J. F. Yu, D. K. Jiang, K. Xiao, Y. Jin, J. H. Wang, X. Sun, Discriminate the falsely predicted protein-coding genes in aeropyrum pernix k1 genome based on graphical representation, Match Communications in Mathematical & in Computer Chemistry 67 (3) (2012) 845–866.

[30] M. R. Boutell, J. Luo, X. Shen, C. M. Brown, Learning multi-label scene classification , Pattern Recognition 37 (9) (2004) 1757–1771.

[31] X. Geng, L. Luo, Multilabel ranking with inconsistent rankers, in: IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 3742–3747.

[32] T. Ahonen, A. Hadid, M. Pietikinen, Face description with local binary patterns: application to face recognition., IEEE Transactions on Pattern Analysis & Machine Intelligence 28 (12) (2006) 2037–2041.

[33] S. H. Cha, Comprehensive survey on distance/similarity measures between probability density functions, International Journal of Mathematical Models & Methods in Applied Sciences 1 (4).

[34] R. Malouf, A comparison of algorithms for maximum entropy parameter estimation, in: proceedings of the 6th conference on Natural language learning-Volume 20, Association for Computational Linguistics, 2002, pp. 1–7.

[35] A. D. Gordon, Classification (2nd edition, Chapman & Hall, London-New York.