Mach Learn (2013) 91:1–42
DOI 10.1007/s10994-012-5312-9

Exploiting label dependencies for improved sample complexity

Lena Chekina · Dan Gutfreund · Aryeh Kontorovich · Lior Rokach · Bracha Shapira

Received: 7 October 2010 / Revised: 1 March 2012 / Accepted: 23 June 2012 /
Published online: 8 August 2012

Abstract Multi-label classification exhibits several challenges not present in the binary case. The labels may be interdependent, so that the presence of a certain label affects the probability of other labels' presence. Thus, exploiting dependencies among the labels could be beneficial for the classifier's predictive performance. Surprisingly, only a few of the existing algorithms address this issue directly by identifying dependent labels explicitly from the dataset. In this paper we propose new approaches for identifying and modeling existing dependencies between labels. One principal contribution of this work is a theoretical confirmation of the reduction in sample complexity that is gained from unconditional dependence. Additionally, we develop methods for identifying conditionally and unconditionally dependent label pairs; clustering them into several mutually exclusive subsets; and finally, performing multi-label classification incorporating the discovered dependencies. We compare these two notions of label dependence (conditional and unconditional) and evaluate their performance on various benchmark and artificial datasets. We also compare and analyze labels identified as dependent by each of the methods. Moreover, we define an ensemble framework for the new methods and compare it to existing ensemble methods. An empirical comparison of the new approaches to existing base-line and state-of-the-art methods on 12 various benchmark datasets demonstrates that in many cases the proposed single-classifier and ensemble methods outperform many multi-label classification algorithms. Perhaps surprisingly, we discover that the weaker notion of unconditional dependence plays the decisive role.

Editors: Grigorios Tsoumakas, Min-Ling Zhang, and Zhi-Hua Zhou.

L. Chekina (✉) · L. Rokach · B. Shapira
Department of Information Systems Engineering and Telekom Innovation Laboratories, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel
e-mail: lenat@bgu.ac.il

D. Gutfreund
IBM Research, Haifa, Israel

A. Kontorovich
Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel

Keywords Multi-label classification · Conditional and unconditional label dependence · Generalization bounds · Multi-label evaluation measures · Ensemble learning algorithms · Ensemble models diversity · Empirical experiment · Artificial datasets

1 Introduction and motivation

Conventional classification tasks deal with problems where each item should be assigned to exactly one category from a finite set of available labels. This type of classification is referred to in the literature as (single-label) multi-class. Conversely, in multi-label classification, an instance can be associated with several labels simultaneously. Multi-label classification has many applications in everyday life. For example, a news item about an assassination attempt in the course of a presidential election campaign can be classified simultaneously to *National Elections* and *Crime* concepts at the same time; a photograph can similarly belong to more than one conceptual class, such as *sunset* and *beaches*; and in music categorization, a song may belong to more than one genre. Multi-labeling is a very common problem in text classification: medical documents, Web pages, and scientific papers, for example, often belong simultaneously to a number of concept classes. Due to its increasing practical relevance as well as its theoretical interest, multi-label classification has received more attention from the machine learning community in recent years and many recent studies look for efficient and accurate algorithms for coping with this classification challenge.

In an exhaustive overview of existing approaches for multi-label classification, Tsoumakas et al. (2010) partition them into two main categories: problem transformation and algorithm adaptation. Problem transformation includes methods that transform the multi-label classification problem into one or more single-label classification problems. The main advantage of these methods is that they are suitable for use with any readily available single-label classifier. Algorithm adaptation embraces methods that extend specific learning algorithms in order to handle multi-label data directly. The main criticism of the adaptation methods is that their application requires changing known classification algorithms in order to adapt them to a specific problem. Algorithm adaptation methods are beyond the scope of this paper.

In the problem transformation category, the common methods used are the Label Powerset (LP) and Binary Relevance (BR) approaches. According to the LP approach, each distinct combination of labels that exists in the multi-label dataset is considered as a single class. The main problem of this method is that many of the created classes are associated with too few examples.

According to the BR approach, a multi-label classification problem is decomposed into multiple, independent binary classification problems and the final labels for each data point are determined by aggregating the classification results from all binary classifiers. The main criticism of this method is that possible dependencies among the labels are ignored.

Recently, many problem transformation methods addressing problems in the LP and BR approaches have been proposed. Some of these methods are discussed in the next section, Related Work.

The aim of this paper is to examine whether the dependencies (conditional or unconditional) among labels can be leveraged to improve the classification accuracy. To this end, we define a natural family of cost functions that interpolates between the 0–1 and the Hamming distances on the multi-label vectors. For each cost function in this family, we derive apparently novel generalization bounds. Furthermore, we give theoretical evidence that unconditional dependence reduces sample complexity. In addition, we propose new algorithms for explicitly identifying conditionally and unconditionally dependent labels and

multi-label classification incorporating the discovered dependencies. Heuristic analysis is used to demonstrate why and under what circumstances the proposed multi-label classification algorithm will be beneficial. Empirical evaluation of the proposed methods on a wide range of datasets confirms our theoretical findings.

The rest of the paper is organized as follows. In the next section, related work is discussed. In Sect. 3 we formally define the multi-label classification problem and analyze a number of measures commonly used for evaluating multi-label classification algorithms. In Sect. 4 some general theoretical results for multi-label learning are derived. Section 5 describes the proposed method and analyzes the circumstances in which it will be beneficial. Section 6 presents the setup of the empirical experiment conducted for evaluating the proposed approaches. And in Sect. 7 the results of the experiment are presented. Finally, Sect. 8 concludes the current work and outlines some further research directions.

2 Related work

In this section we briefly review several recently proposed algorithms for multi-label classification and then, in the light of these and other previous works, summarize what we believe are the original contributions of this paper.

The data sparseness problem of the LP approach was addressed in Read et al. (2008). The authors propose Pruned Sets (PS) and Ensemble of Pruned Sets (EPS) methods to concentrate on the most important correlations. This is achieved by pruning away examples with infrequently occurring label sets. Some of the pruned examples are then partially reintroduced into the data by decomposing them into more frequently occurring label subsets. Finally, a process similar to the regular LP approach is applied on the new dataset. The authors show empirically that the proposed methods are often superior to other multi-label methods. However, these methods are likely to be inefficient in domains with a large proportion of distinct label combinations (Read et al. 2008) and with an even distribution of examples over those combinations. Another limitation of the PS and EPS methods is the need to balance the trade-off between information loss (caused by pruning training examples) and adding too many decomposed examples with smaller label sets. For this purpose there is a need to choose some non-trivial parameter values before applying the algorithm or, alternatively, to perform calibration tests for parameters adjustment. And still another limitation is that the dependencies within the decomposed label sets are not considered.

Another approach for multi-label classification in domains with a large number of labels was proposed by Tsoumakas et al. (2008). The proposed algorithm (HOMER) organizes all labels into a tree-shaped hierarchy with a much smaller set of labels at each node. A multi-label classifier is then constructed at each non-leaf node, following the BR approach. The multi-label classification is performed recursively, starting from the root and proceeding into the child nodes only if their labels are among those predicted by the parent's classifier. One of the main HOMER processes is the clustering of the label set into disjoint subsets so that similar labels are placed together. This is accomplished by applying a balanced k-means clustering algorithm on the label part of the data. In this work, differently from HOMER we try to cluster labels based on the level of dependency among them and perform a flat multi-label classification (considering internal dependencies) into each one of the clusters.

A recent paper by Read et al. (2009) argues in defense of the BR method. It presents a method for chaining binary classifiers—Classifiers Chains (CC)—in a way that overcomes the label independence assumption of BR. According to the proposed method, a single binary classifier is associated with each one of the predefined labels in the dataset and all these

classifiers are linked in an ordered chain. The feature space of each classifier in the chain is extended with the 0/1 label associations of all previous classifiers. Thus, each classification decision for a certain label in the chain is augmented by all prior binary relevance predictions in the chain. In this way correlations among labels are considered. The CC method has been shown to improve classification accuracy over the BR method on a number of regular (not large-size) datasets. One of the disadvantages of this method, noted by authors, is that the order of the chain itself has an effect on accuracy. This can be solved either by a heuristic for selecting a chain order or by using an ensemble of chain classifiers. Any of these solutions increases the required computation time.

Recently, a probabilistic extension of the CC algorithm was proposed (Dembczynski et al. 2010a). According to the probabilistic classifier chains (PCC) approach, the conditional probability of each label combination is computed using the product rule of probability. For estimating the joint distribution of labels, a model is learned for each label on a feature space augmented by previous labels as additional attributes. The classification prediction is then derived from the calculated join distributions in an explicit way. Authors theoretically and empirically confirm expectations that PCC produces better estimates than original Classifier Chains. However, the price is paid in a much higher algorithm complexity. In fact, the main disadvantage of the PCC method is its applicability only on datasets with a small number of labels, not more than about 15.

An idea relatively close to that described in this research is presented in Tsoumakas and Vlahavas (2007). The authors propose an approach that constructs an ensemble of LP classifiers. Each LP classifier is trained using a different, small random subset of the set of labels. This approach (RAkEL) aims at taking into account label correlations and at the same time avoiding the LP limitations mentioned above. A comparison shows the superiority of RAkEL's performance over popular BR and LP methods on the full set of labels. Tsoumakas and Vlahavas note that the random nature of their method may lead to including models that affect the ensemble's performance in a negative way.

To the best of our knowledge, few works on multi-label learning have directly identified dependent labels explicitly from the dataset. One such method where the degree of label correlation is explicitly measured was presented in Tsoumakas et al. (2009). In this paper the authors use stacking (Wolpert 1992) of BR classifiers to alleviate the label correlations problem. The idea in stacking is to learn a second (or meta) level of models that consider as input the output of all first (or base) level models. In this way, correlations between labels are modeled by a meta-level classifier. To avoid the noise that may be introduced by modeling uncorrelated labels in the meta-level, the authors prune models participating in the stacking process by explicitly measuring the degree of label correlation using the phi coefficient. They showed, by exploratory analysis, that detected correlations are meaningful and useful. The main disadvantage of this method is that the identified correlations between labels are utilized by a meta-level classifier only. In this paper we show that direct exploration of label dependence (i.e. by a base-level classifier) is more beneficial for predictive performance.

Another recent paper by Zhang and Zhang (2010) exploited conditional dependencies among labels. For this purpose the authors utilize a Bayesian network representing the joint probability of all labels conditioned on the feature space, such that dependency relations among labels are explicitly expressed by the network structure. Zhang and Zhang learn approximate network structure from classification errors of independent binary models for all labels. On the next step, a new binary classifier is learned for each label by treating its parental labels in the network as additional input features. The labels of unseen examples are predicted by binary classifiers learned on the feature space augmented by its parental labels. The ordering of the labels is implied by the Bayesian network structure. Zhang and

Zhang (2010) showed empirically that their method is highly comparable to some of the state-of-the-art approaches over a range of datasets using 3 multi-label evaluation measures. Note that according to this method all parental labels, which a certain label is found to be dependent on, are added to the feature space. The main limitation of this method is in the complexity of Bayesian network learning. It can be efficiently learned with only a small (up to 20) number of variables. To handle cases where the number of variables is larger than 20, the authors switch the algorithm to approximate maximum a posterior (MAP) structure learning for which maximum running time and some other parameters should be specified.

Tenenboim et al. (2009) demonstrated that dividing the whole set of labels into several mutually exclusive subsets of dependent labels and applying a combination of BR and LP methods to these subsets provides in many cases higher predictive performance than regular LP and BR approaches.

A general detailed analysis of the label dependence issue was presented in Dembczynski et al. (2010b). The paper distinguishes and formally explains the differences and connections between two dependence types—conditional and unconditional. Also an overview of state-of-the-art algorithms for MLC and their categorization according to the type of label dependence they seek to capture is given. As well, authors analyze potential benefit of exploiting label dependencies in the light of 3 different loss functions.

Recently, some algorithm adaptation methods considering labels correlations in various ways were proposed. For example, Zhang et al. (2009) proposed an extension to the popular Naive Bayes classifiers for dealing with multi-label instances called MLNB. The authors incorporated feature selection techniques (principal component analysis and genetic algorithms) to mitigate the harmful effects caused by the classic Naive Bayes assumption of class conditional independence. Furthermore, correlations between different labels were also explicitly addressed through the specific fitness function used by the genetic algorithm. Authors experimentally demonstrated the effectiveness of utilized feature selection techniques in addressing the inter-label relationships and reported significant rise in the performance of MLNB due to these techniques.

Another example of algorithm adaptation for multi-label classification considering inter-label relationships is ML-SVDD, a fast multi-label classification algorithm based on support vector data description (Xu 2010). According to this algorithm, a k-label problem is divided into k sub-problems each of which consists of instances from a specific class. For each class a sub-classifier is learned using support vector data description method. For making an entire multi-label classification decision predictions of all sub-classifier are combined as follows. The classes which predicted pseudo posterior probability above some threshold are added to the set of predicted labels. To compensate missing correlations between labels linear ridge regression model is used when constructing a threshold function.

In this paper, we analyze the commonly used multi-label evaluation measures and demonstrate that classification accuracy is better suited than other measures for general evaluation of classifier performance for most regular multi-label classification problems. Thus, accuracy is the target evaluation measure that we aim to improve in the current research.

We propose to discover existing dependencies among labels in advance, before any classifiers are induced, and then to use the discovered dependencies to construct a multi-label classifier. We define methods estimating conditional and unconditional dependencies between labels from a training set and apply a new algorithm that combines the LP and BR methods on the results of each one of the dependence identification methods. The new algorithm is termed "LPBR". We then compare the contributions of both methods for label dependency identification on classifier predictive performance in terms of 4 evaluation measures. Moreover, an ensemble framework that we introduce for the proposed algorithm makes it possible to further improve the classifier's predictive performance.

We empirically evaluate the proposed methods on twelve multi-label datasets and show that the new approach for multi-label classification outperforms many existing methods.

The main contributions of this paper are:

- The development of the LPBR algorithm which combines the best features of the LP and BR methods while eliminating their inherent disadvantages.
- Theoretical confirmation of the reduction in sample complexity that is gained from unconditional dependence.
- Heuristic analysis of conditions for which the LPBR method is supposed to be beneficial.
- The formulation of novel algorithms—ConDep and ChiDep—for explicitly identifying conditionally and unconditionally dependent label pairs, clustering them into disjoint subsets and applying LPBR for modeling the specified dependencies.
- An ensemble framework for the ConDep and ChiDep algorithms.
- An extensive empirical evaluation experiment comparing the effectiveness of developed algorithms to nine existing multi-label algorithms on a wide range of various datasets.

3 Problem formulation and evaluation measures analysis

3.1 Formal definitions

Our learning model is the following standard extension of the binary case. We have a teacher generating examples $X \in \mathcal{X}$ iid according to some distribution P. Each example X_i is accompanied by its multi-label Y_i, for which we use subset notation $Y \subseteq [L]$ or vector notation $Y \in \{0,1\}^L$, as dictated by convenience.[1]

Assume $\mathcal{Y}$ is a given set of predefined binary labels $\mathcal{Y} = \{\lambda_1, \ldots, \lambda_L\}$. For a given set of labeled examples $D = \{X_1, X_2, \ldots, X_n\}$ the goal of the learning process is to find a classifier $h : \mathcal{X} \to \mathcal{Y}$, which maps an object $X \in \mathcal{X}$ to a set of its classification labels $Y \in \mathcal{Y}$, such that $h(X) \subseteq \{\lambda_1, \ldots, \lambda_L\}$ for all X in $\mathcal{X}$.

The main feature distinguishing multi-label classification from a regular classification task is that a number of labels have to be predicted simultaneously. Thus, exploiting potential dependencies between labels is important and may improve classifier predictive performance. In this paper we consider two types of label dependence, namely conditional and unconditional. The first type refers to dependencies between labels conditional to (i.e. given) a specific instance, while the second one refers to general dependencies existing in the whole set, independently of any concrete observation.

Both types of dependence are formally defined below.

Definition 1 A set of labels $Y \subseteq [L]$ is called unconditionally L-independent if

$$P(Y) = \prod_{i=1}^{L} p^{(i)}(Y_i),$$

where $p^{(i)}(Y_i)$ is the marginal distribution of λ_i.

[1] Actually, most of our results continue to hold in the agnostic setting (Kearns et al. 1994), where there is no "teacher" and the probability distribution P is over example-label pairs (X, Y).

Definition 2 A set of labels $Y \subseteq [L]$ is called conditionally L-independent given X if

$$P_x(Y) = \prod_{i=1}^{L} p_x^{(i)}(Y_i),$$

where $p_x^{(i)}(Y_i)$ is the marginal distribution of λ_i given X.

We refer to these definitions when talking about conditional and unconditional dependencies in Sect. 5.1, Label Dependence Identification.

3.2 Evaluation measures analysis

In this paper we consider the most commonly used multi-label evaluation measures from Tsoumakas and Vlahavas (2007), namely multi-label example-based classification accuracy, subset accuracy, Hamming loss, and label-based micro-averaged F-measure. Their formal definition and analysis are presented below.

Let D be a multi-label evaluation dataset, consisting of $|D|$ multi-label examples (X_i, Y_i), $i = 1 \ldots |D|$, $Y_i \subseteq [L]$. Let h be a multi-label classifier.

Hamming loss computes the percentage of labels whose relevance is predicted incorrectly. For the two label subsets $A, B \subseteq [L]$, their Hamming distance is

$$\ell_{Ham}(A, B) = \frac{1}{L} \sum_{i=1}^{L} 1_{\{i \in A\}} \neq 1_{\{i \in B\}}. \tag{1}$$

Over all dataset examples the Hamming loss is averaged as follows:

$$\textit{Hamming loss}(h, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{Y_i \Delta h(X_i)}{L}$$

where Δ stands for the symmetric difference between two sets.

Hamming loss is very sensitive to the label set size L. It measures the percentage of incorrectly predicted labels both positive and negative. Thus, in cases where the percentage of positive labels is low relative to L, the low values of the Hamming loss measure do not give an indication of high predictive performance. Thus, as the empirical evaluation results demonstrate below, the accuracy of the classification algorithm on two datasets with similar Hamming loss values may vary from about 30 to above 70 percent (as, for example, in the "bibtex" and "medical" datasets). However, the Hamming loss measure can be useful for certain applications where errors of all types (i.e., incorrect prediction of negative labels and missing positive labels) are equally important.

Subset accuracy computes the number of exact predictions, i.e., when the predicted set of labels exactly matches the true set of labels. This measure is the opposite of the zero-one loss, which for the two binary vectors $A, B \subseteq [L]$ is defined as follows:

$$\ell_{01}(A, B) = \mathbf{1}_{\{A \neq B\}}. \tag{2}$$

Over all dataset examples the subset accuracy is averaged as follows:

$$\textit{Subset accuracy}(h, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} I\big(Y_i = h(X_i)\big).$$

It should be noted also that subset accuracy is a very strict measure since it requires the predicted set of labels to be an exact match of the true set of labels, and equally penalizes

predictions that may be almost correct or totally wrong. However, it can be useful for certain applications where classification is only one step in a chain of processes and the exact performance of the classifier is highly important. For example, assume a LEDs classification problem where each combination of turned on LEDs specifies a certain number. In this case, an incorrect prediction even of a single LED makes the whole classification decision absolutely incorrect and useless.

Accuracy computes the percentage of correctly predicted labels among all predicted and true labels. Accuracy averaged over all dataset examples is defined as follows:

$$Accuracy(h, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{Y_i \cap h(X_i)}{Y_i \cup h(X_i)}.$$

Accuracy seems to be a more balanced measure and better indicator of an actual algorithm's predictive performance for most standard classification problems than Hamming loss and subset accuracy. However, it should be noted that it also is relatively sensitive to dataset label cardinality (average number of labels per example). This means that for two classification problems (i.e., datasets) of the same complexity, accuracy values would be lower in the dataset with the higher label cardinality. The empirical evaluation experiment below supports this conclusion (consider, for example accuracy and F-measure values on "emotions", "scene" and "yeast" datasets).

The F-measure is the harmonic mean between precision (π) and recall (ρ) and is commonly used in information retrieval. Precision and recall are defined as follows:

$$\pi_\lambda = \frac{TP_\lambda}{TP_\lambda + FP_\lambda}, \qquad \rho_\lambda = \frac{TP_\lambda}{TP_\lambda + FN_\lambda},$$

where TP_λ, FP_λ and FN_λ stands for the number of true positives, false positives and false negatives correspondingly after binary evaluation for a label λ.

The micro-averaged precision and recall are calculated by summing over all individual decisions:

$$\pi = \frac{\sum_{\lambda=1}^{L} TP_\lambda}{\sum_{\lambda=1}^{L} (TP_\lambda + FP_\lambda)}, \qquad \rho = \frac{\sum_{\lambda=1}^{L} TP_\lambda}{\sum_{\lambda=1}^{L} (TP_\lambda + FN_\lambda)},$$

where L is the number of labels. The micro-averaged F-measure score of the entire classification problem is then computed as:

$$F(\textit{micro-averaged}) = \frac{2\pi\rho}{\pi + \rho}.$$

Note that micro-averaged F-measure gives equal weight to each document and is therefore considered as an average over all the document/label pairs. It tends to be dominated by the classifier's performance on common categories and is less influenced by the classifier's performance on rare categories.

Of the various measures that are discussed here, the micro-averaged F-measure seems to be the most balanced and the least dependent on dataset properties. Thus it could be the most useful indicator of classifier general predictive performance for various classification problems. However it is more difficult for human interpretation, as it combines two other measures (precision and recall).

Summarizing the above analysis of some of the most commonly used evaluation measures, we conclude that accuracy and micro-averaged F-measure are better suited for general evaluation of algorithm performance for most regular multi-label classification problems while the Hamming loss and subset accuracy measures may be more appropriate for some specific multi-label classification problems.

Actually the accuracy measure, where the number of true labels among the predicted ones is important, can be considered as a "golden mean" between Hamming loss where all labels are equally important and subset accuracy where only the whole set of positive labels is important. Thus, in this research we aim at improving the accuracy measure.

Some other evaluation measures, such as one-error, coverage, ranking loss and average precision, which are specially designed for multi-label ranking do exist (Schapire and Singer 2000). This category of measures, known as ranking-based, is often used in the literature (although not directly related to multi-label classification), and is nicely presented in Tsoumakas et al. (2010) among other publications. These measures are tailored for evaluation of specific-purpose ranking problems and are of a less interest for our research.

4 Generalization bounds for multi-label learning

In this section, we derive some general theoretical results for multi-label learning.

4.1 General theory

Following Eqs. (1) and (2) we can interpolate between the Hamming and the 0–1 distance via the family of distances ℓ_k, for $k = 1, \ldots, L$:

$$\ell_k(A, B) = \binom{L}{k}^{-1} \sum_{\substack{E \subseteq [L] \\ |E|=k}} \mathbf{1}_{\{A \cap E \neq B \cap E\}}. \tag{3}$$

Note that $k = 1$ corresponds to ℓ_{Ham} and $k = L$ corresponds to ℓ_{01}.

Given Definition 1 our goal is to guarantee that any hypothesis $h : \mathcal{X} \to \{0, 1\}^L$, in some class $\mathcal{H}$ achieves, with high probability,

$$\mathbf{E}\big[\ell_k\big(\mathbf{h}(X), Y\big)\big] \leq \frac{1}{n} \sum_{i=1}^{n} \ell_k\big(\mathbf{h}(X_i), Y_i\big) + \varepsilon. \tag{4}$$

We denote by $\mathcal{H}$ the space of all admissible hypotheses and make the following structural assumption on $\mathcal{H}$: there is some fixed concept class $\mathcal{C} \subset 2^{\mathcal{X}}$ such that each $\mathbf{h} \in \mathcal{H}$ is of the form

$$\mathbf{h}(x) = \big(h_1(x), h_2(x), \ldots, h_L(x)\big),$$

where $h_j \in \mathcal{C}$.

Let us define the auxiliary function $f : \mathcal{X} \times \{0, 1\}^L \to [0, 1]$ as

$$f(X, Y) = \ell_k\big(\mathbf{h}(X), Y\big)$$

where $\mathbf{h} \in \mathcal{H}$; we denote by $\mathcal{F}$ the space of all functions f that may be obtained in this way. In words, f combines the prediction of a hypothesis with the loss that it suffers. Note that the function class $\mathcal{F}$ is determined entirely by $\mathcal{C}$ and ℓ_k.

For any class of real valued functions $\mathcal{F}$, its pseudo-dimension is defined as follows (Pollard 1984; Vapnik 1995). For $t \in \mathbb{R}$, define the operator $\text{Binary}_t : \mathbb{R}^{\mathcal{X}} \to \{0, 1\}^{\mathcal{X}}$, which maps a function $f : \mathcal{X} \to \mathbb{R}$ to the function $\mathrm{g} : \mathcal{X} \to \{0, 1\}$, given by

$$\mathrm{g}(\mathrm{z}) = 1_{\mathrm{g}(\mathrm{z}) > t}.$$

For a class of functions: $\mathcal{F} \subset \mathbb{R}^{\mathcal{X}}$, its pseudo-dimension is defined by

$$\text{Pdim}(\mathcal{F}) = \text{VCdim}\big(\big\{\text{Binary}_t(f) : f \in \mathcal{F}, t \in \mathbb{R}\big\}\big).$$

(We will occasionally abuse notation by writing $\mathrm{VCdim}(\mathcal{H})$ as a shorthand for $\mathrm{Pdim}(\mathcal{F})$, where $\mathcal{F}$ is the auxiliary class defined above, under the 0–1 loss ℓ_L.)

In order to bound $\mathrm{Pdim}(\mathcal{F})$, we will need the following result. Define the step function $\theta : \mathbb{R} \to \{0, 1\}$ by $\theta(x) = 1_{\{x \geq 0\}}$. For any concept class $\mathcal{C}$ over $\mathcal{X}$ and any $T \in \mathbb{N}$, define the concept class of thresholded linear combinations of concepts in $\mathcal{C}$:

$$\Theta_T(\mathcal{C}) = \left\{ x \mapsto \theta\left(\sum_{t=1}^{T} a_t h_t - b \right) : b, a_t \in \mathbb{R},\ h_t \in \mathcal{C} \right\}.$$

The techniques developed in Blumer et al. (1989) and Baum and Haussler (1989) allow us to obtain tight (see Eisenstat and Angluin 2007; Eisenstat 2009) bounds on $\mathrm{VCdim}(\Theta_T(\mathcal{C}))$ in terms of $\mathrm{VCdim}(\mathcal{C})$:

Lemma 1 (Baum and Haussler 1989)

$$\mathrm{VCdim}\big(\Theta_T(\mathcal{C})\big) \leq 2(d+1)(T+1)\log_2\big(e(T+1)\big) = O(Td\log T)$$

where $d = \mathrm{VCdim}(\mathcal{C})$.

Corollary 1 *Define* $\mathcal{C}^{\cup k}$ *to be the set of all* k*-fold unions of over* $\mathcal{C}$:

$$\mathcal{C}^{\cup k} = \{h_1 \cup h_2 \cup \cdots \cup h_k : h_i \in \mathcal{C}\}.$$

Then

$$\begin{aligned} \mathrm{VCdim}\big(\mathcal{C}^{\cup k}\big) &\leq 2(d+1)(k+1)\log_2\big(e(k+1)\big) \\ &= O(kd\log k). \end{aligned}$$

Proof Any function $h \in \mathcal{C}^{\cup k}$ may be represented as

$$h = \theta\left(\sum_{i=1}^{k} h_i - 1 \right)$$

with $h_i \in \mathcal{C}$. Thus, the claimed bounds follow immediately from Lemma 1. □

Theorem 2 *Let* $\mathcal{F}$ *be the family of functions induced by* $\mathcal{C}$ *and* ℓ_k*, as above. Then*

$$\mathrm{Pdim}(\mathcal{F}) = O\left(\begin{cases} Ld\log L, & k = 1 \\ Ld, & k = L \\ \binom{L}{k} kd\log k \log\binom{L}{k}, & 1 < k < L \end{cases} \right)$$

where $d = \mathrm{VCdim}(\mathcal{C})$.

Proof Consider the case $1 < k < L$. Each function $f : \mathcal{X} \times \{0,1\}^L \to \mathbb{R}$ in $\mathcal{F}$ may be expressed as

$$f(x, y) = \binom{L}{k}^{-1} \sum_{\substack{E \subseteq [L] \\ |E| = k}} f_E(x, y)$$

where for $x \in \mathcal{X}$, $y \in \{0,1\}^L$ and $E \subseteq [L]$,

$$f_E(x, y) = 1_{\{h_E \neq y_E\}}$$

where $\mathbf{h} \in \mathcal{H}$ and the subscript E indicates the restriction of the L coordinates to the set $E \subseteq [L]$. Denote by $\mathcal{F}_k$ the set of all $f_E : \mathcal{X} \times \{0,1\}^L$ that can be obtained in this way; then Lemma 1 implies that

$$\mathrm{VCdim}(\mathcal{F}) = O\left(\binom{L}{k} \mathrm{VCdim}(\mathcal{F}_E) \log \binom{L}{k}\right). \tag{5}$$

Now we apply Lemma 1 again to $\mathcal{F}_k$:

$$\mathrm{VCdim}(\mathcal{F}_k) = O\big(k\,\mathrm{VCdim}(\mathcal{C}) \log k\big). \tag{6}$$

The theorem follows by combining (5) and (6); the analysis for $k = 1$ and $k = L$ is a specialization of the arguments above. □

Our desired generalization bound (4) is now immediate:

Theorem 3 *Let (X_i, Y_i), $i = 1, \ldots, n$ be an iid sample, with $X_i \in \mathcal{X}$ and $Y_i \in \{0,1\}^L$. Suppose the hypothesis $\mathbf{h} : \mathcal{X} \to \{0,1\}^L$ is of the form*

$$\mathbf{h}(x) = \big(h_1(x), h_2(x), \ldots, h_L(x)\big),$$

where $h_j \in \mathcal{C}$ for some concept class $C \subset 2^{\mathcal{X}}$ with $\mathrm{VCdim}(\mathcal{C}) = d$. Then for any $\delta > 0$ we have with probability at least $1 - \delta$, for any $\mathbf{h} \in \mathcal{H}$

$$\mathbf{E}\big[\ell_k\big(\mathbf{h}(X), Y\big)\big] \le \frac{1}{n}\sum_{i=1}^{n} \ell_k\big(\mathbf{h}(X_i), Y_i\big) + \sqrt{\frac{2D \log \frac{en}{D}}{n}} + \sqrt{\frac{\log \frac{1}{\delta}}{2n}}$$

where $D = O((\frac{L}{k})kd \log k \log(\frac{L}{k}))$ and ℓ_k is the interpolated loss defined above, for $k \in [L]$.

Proof Immediate from the pseudo-dimension estimate in Theorem 2 and the standard generalization bounds (Pollard 1984; Vapnik 1995). □

4.2 Tighter bounds via unconditional dependence

To illustrate the advantage conferred by unconditionally dependent labels, let us consider the following toy example. Take $\mathcal{X} = \{0,1\}^n$ and let $\mathcal{C}$ be the set of all monotone conjunctions over the n Boolean variables. That is, each member of $\mathcal{C}$ is an AND of some fixed subset of the n variables, without negations.

We will consider the 0–1 distance over multi-labels (corresponding to $k = L$). Since $|\mathcal{C}| = 2^n$, we have $d = \mathrm{VCdim}(\mathcal{C}) \le n$. In fact, $d = n$, since the $\mathcal{C}$ shatters the set $\{10^{n-1}, 010^{n-2}, \ldots, 0^{n-1}1\} \subset \{0,1\}^n$. Similarly, it is easy to see that $\mathcal{H} = \mathcal{C}^L$ has VC-dimension Ln. Thus, a sample of size $\Omega(Ln/\varepsilon)$ is necessary in order to achieve a distribution-free generalization error bounded by ε with high probability (Blumer et al. 1989).

How might we exploit dependence between the labels? Let us consider the case of 2 labels ($L = 2$). Thus, the target concept is a function $g : \mathcal{X} \to \{0,1\}^2$, given by $\boldsymbol{g}(x) = (g_1(x), g_2(x))$, with $g_1, g_2 \in \mathcal{C}$. Without any assumption on the dependence between g_1 and g_2, we have $\mathrm{VCdim}(\mathcal{H}) = 2n$. However, let us suppose that g_1 and g_2 are not independent in the following sense: they can only differ by at most τ variables. That is, if $A_1, A_2 \subseteq [n]$ are the two sets of variables corresponding to g_1 and g_2, then $|A_1 \Delta A_2| \le \tau$. For a fixed set of variables $S \subseteq [n]$, there are not more than

$$\left(\sum_{i=0}^{\tau} \binom{n}{i}\right)^2 \le (en/\tau)^{2\tau} = O\big(n^{2\tau}\big)$$

subsets $A_1, A_2 \subseteq [n]$ *s.t.* $A_1 \cap A_2 = S$ and $|A_1 \Delta A_2| \leq \tau$. This upper-bounds the number of such "τ-close" conjunctions by $2^n (en/\tau)^{2\tau}$, implying $\mathrm{VCdim}(\mathcal{H}) \leq n + 2\tau \log_2(en/\tau)$—a clear improvement over the independent case. One can easily extend this analysis to the case that there are $L > 2$ conjunctions all intersecting on a large subset of the variables, leaving only τ variables for each conjunction outside the intersection. The above analysis implies that in this case ($L > 2$ conjunctions) $\mathrm{VCdim}(\mathcal{H}) \leq n + L\tau \log_2(en/\tau)$, as opposed to the Ln in the independent case.

In greater generality, we may define a dependence ratio as follows. Let $\mathcal{X}$ be an instance space and $\mathcal{C}$ a concept class over $\mathcal{X}$. For some fixed L, let $\mathcal{H}$ be a collection of functions $\mathcal{X} \to \{0,1\}^L$, such that $h(x) = (h_1(x), h_2(x), \ldots, h_L(x))$, with $h_j \in \mathcal{C}$. Define the *dependence ratio*

$$\rho(\mathcal{C}, \mathcal{H}) = \frac{\mathrm{VCdim}(\mathcal{C}^L)}{\mathrm{VCdim}(\mathcal{H})} = L \frac{\mathrm{VCdim}(\mathcal{C})}{\mathrm{VCdim}(\mathcal{H})}$$

(recall our convention of writing $\mathrm{VCdim}(\mathcal{H})$ as a convenient shorthand for $\mathrm{Pdim}(\mathcal{F})$, where $\mathcal{F}$ is the auxiliary class defined above, under the 0–1 loss ℓ_L).

Then, combining the sample complexity lower bound (Blumer et al. 1989) with the VC dimension upper bound (Vapnik and Chervonenkis 1971), we obtain the following result:

Theorem 4 *Let $\mathcal{C}^L$ be the collection of all functions $h : \mathcal{X} \to \{0,1\}^L$ of the form $h(x) = (h_1(x), h_2(x), \ldots, h_L(x))$, with $h_j \in \mathcal{C}$ and let $\mathcal{H} \subseteq \mathcal{C}$. Let $m_{\mathcal{C}}(\epsilon, \delta)$ be the sample complexity corresponding to $\mathcal{C}^L$—i.e., the smallest number of examples required for a consistent learner to achieve an expected* 0–1 *loss of not more than ϵ with probability at least $1 - \delta$. Let $m_{\mathcal{H}}(\epsilon, \delta)$ be the analogous sample complexity for $\mathcal{H}$. Then*

$$m_{\mathcal{H}}(\epsilon, \delta) \leq m_{\mathcal{C}}(\epsilon, \delta) / \tilde{\Theta}\big(\rho(\mathcal{C}, \mathcal{H})\big)$$

where the $\tilde{\Theta}(\cdot)$ notation suppresses logarithmic factors.

Putting computational issues aside (e.g., how to efficiently find a consistent pair of τ-close conjunctions), this analysis implies that dependence among the labels can be exploited to significantly reduce the sample complexity.

5 Methods

This section describes the proposed methods for multi-label classification.

Our approach comprises two main steps. The aim of the first step is preliminary identification of dependencies and the clustering of all labels into several independent subsets. In this paper we examine two methods for handling this step, namely, identifying unconditional and conditional label dependencies.

The second step is multi-label classification incorporating the category dependencies discovered in the previous step. For this we apply a combination of standard BR and LP approaches to the independent groups of labels that have been defined.

Following is a description of the methods applied for each of the steps.

5.1 Label dependence identification

5.1.1 Unconditional label dependence

The proposed method is based on analyzing the number of instances in each category. We apply the chi-square test for independence to the number of instances for each possible

Table 1 General contingency table for labels λ_i and λ_j

	$\boldsymbol{\lambda}_i$	$\neg\boldsymbol{\lambda}_i$	total
$\boldsymbol{\lambda}_j$	a	b	$a+b$
$\neg\boldsymbol{\lambda}_j$	c	d	$c+d$
total	$a+c$	$b+d$	$a+b+c+d=N$

Unconditional label dependence identification

input: D – training set; $\mathcal{Y}$ – a set of labels $\{\lambda_1, \ldots, \lambda_L\}$
output: *orderedScoresList* – ordered list of dependence score for each label pair
1. FOR each label pair (λ_i, λ_j), where i=0,…L-1 and j=i+1,…L
2. *score* = compute chi-square score of the labels pair (λ_i, λ_j)
3. add new pair $[(\lambda_i, \lambda_j)$, *score*] to the *dependencyScoresList*
4. ENDFOR
5. *orderedScoresList* = sort the *dependencyScoresList* in descending order of the *score*
6. RETURN *orderedScoresList*

Fig. 1 A pseudo-code for the unconditional label dependence identification algorithm

combination of two categories. The level of dependence between each two labels in the dataset is thus identified.

Given two labels, λ_i and λ_j, and the frequency counts of their co-occurrences as presented in Table 1, the χ^2 score is computed as follows:

$$\chi^2 = \frac{(ad-bc)^2(a+b+c+d)}{(a+b)(c+d)(b+d)(a+c)}.$$

The label pairs with χ^2 score higher than a critical value of χ^2 are considered as dependent.

A pseudo-code for the unconditional label dependence identification algorithm is given in Fig. 1. This algorithm along with the label clustering and LPBR approaches described below will subsequently be referred to as ChiDep.

5.1.2 Conditional label dependence

Conditional label dependence is hard to identify since it is specific for each instance x. We try to estimate the conditional dependencies between each pair of labels by evaluating the advantage gained by exploiting this dependence for classification.

For two conditionally independent labels λ_1 and λ_2, the following holds:

$$P(\lambda_1|\lambda_2, X) = P(\lambda_1|X).$$

The above equation means that if labels λ_1 and λ_2 are conditionally independent, then the predictions (of λ_1) by probability-based classification models trained once on a regular features space x and second on the features space x augmented by label λ_2 should be at least very similar.

Following this condition, for each pair of labels λ_i and λ_j, we train two binary classifiers for predicting λ_i. One—unconditional on λ_j—is trained on a regular features space x; the second one—conditional on λ_j—is trained on the features space x augmented by label λ_j as additional feature. Then we compare the accuracy of both classifiers using 5×2 fold cross-validation. If the accuracy of the conditional model is significantly higher than that of the unconditional one, we conclude that the labels are conditionally dependent. The statistical significance of the difference between the resultant vectors of both classifiers is determined using two-sample t-test. Since both models were evaluated on the same folds of data a paired

Conditional label dependency identification

input: D – training set; $\mathcal{Y}$ – a set of labels $\{\lambda_1, \ldots, \lambda_L\}$
output: *orderedScoresList* – ordered list of dependence score for each label pair
1. FOR each label pair (λ_i, λ_j), where i=0,…L-1 and j=i+1,…L
2. Create N data folds → *trainDataArray*, *testDataArray*
3. FOR each *fold*
4. train binary modelA: h(x) → { λ_i }on the *trainDataArray[fold]*
5. train binary modelB: h(x, λ_j) → { λ_i } *trainDataArray[fold]*
6. *accuracyArrayA*[*fold*] = correctPredictions(modelA.evaluate(*testDataArray[fold]*)
7. *accuracyArrayB*[*fold*] = correctPredictions(modelB.evaluate(*testDataArray[fold]*)
8. ENDFOR each *fold*
9. *avgA* = averageValues(*accuracyArrayA*)
10. *avgB* = averageValues(*accuracyArrayB*)
11. *t-value* = applyPairedT-test(*accuracyArrayA* , *accuracyArrayB*)
12. IF (*avgB* > *avgA*) AND (*t-value* > *t-critical*) THEN *val*(λ_i, λ_j)= *t-value*
13. ELSE *val*(λ_i, λ_j)=0
14. REPEAT STEPS 2-13 for the labels pair (λ_j, λ_i) to get *val*(λ_j, λ_i)
15. *score* = maximum of *val*(λ_i, λ_j) and *val*(λ_j, λ_i)
16. add new pair [(λ_i, λ_j), *score*] to the *dependencyScoresList*
17. ENDFOR each two labels
18. *orderedScoresList* ← sort the *dependencyScoresList* in descending order of the *score*
19. RETURN *orderedScoresList*

Fig. 2 A pseudo-code for the conditional label dependence identification algorithm

t-test is applied. To comply with the t-test assumptions each of the two compared populations should follow a normal distribution. We checked the normality assumption using the Shapiro-Wilk test and find out that in most of the cases this assumption holds.

The label pairs with a t-value higher than t-critical are considered as dependent. We perform this procedure for all possible label pairs considering the labels order in the pair.[2] Among the two pairs with the same labels, the pair with maximal t-statistic value is added to the resulting list of dependent pairs (for the clustering algorithm the order of labels in the pairs does not matter). Finally, we sort the resultant label pairs according to their t-statistic value in descending order (i.e., from the most to the least dependent pairs).

A pseudo-code for the conditional label dependence identification algorithm is given in Fig. 2. This algorithm for dependent label identification along with the label clustering and LPBR approaches described below will subsequently be referred to as ConDep.

Note that even such approximate estimation of conditional dependencies is very computationally expensive. Indeed, the approach becomes very time consuming and sometimes not feasible at all for large datasets.

5.1.3 Dependent labels clustering

In this paper we implemented a straightforward greedy clustering approach. Initially we assume all labels to be independent and perform a multi-label classification according to the BR approach. We then get the list of label pairs ordered according to their dependence score value (χ^2 or t for unconditional and conditional dependence respectively), and start to group

[2]Theoretically, based on the symmetry rule of conditional independence we could skip the test of a pair (λ_j, λ_i) if the pair (λ_i, λ_j) was found to be independent. However the described procedure only approximately estimates conditional independence. Thus in these circumstances, the symmetry rule may not hold for some pairs.

the most dependent labels together. First, the two most dependent labels are clustered into a group and a multi-label classification model (LPBR) is induced on the new set of labels as described in Sect. 5.2. Then, the next most dependent label pair is analyzed. If both labels are single, they will be joined into a new group. If one of the labels already belongs to a group, the second label will be appended to this group. If both labels belong to two different groups those groups are unified. If both labels belong to the same group, the procedure proceeds to the next pair of labels. In the next step, a multi-label classification model will be constructed using the new set of grouped labels. After each classification model is constructed, we compare its accuracy (in terms of the configurable target evaluation measure) to that of the model from the previous step. The process of grouping labels continues as long as the accuracy improves. However, we allow a number of steps without seeking any concomitant improvement in the accuracy. This allows the algorithm to escape from a local-minimum when it encounters a "non-useful" pair of labels. Hence, a "non-useful" combination is filtered out and the algorithm continues to evaluate subsequent pairs of dependent labels until one of the stop conditions is reached. The possible stop conditions are:

– no more label pairs to consider
– all labels are clustered into one single group
– pair dependence score (chi-square value or t-value) is below some threshold value t
– the number of allowed "non-improving" label pairs n is exceeded.

While the first two stop conditions are beyond our control, the third and fourth could be configured and used to control model complexity. Intuition suggests that if there are a number of consequent reductions in accuracy, it is not "by chance" and further clustering of less dependent labels can cause model over-fitting to the training data. Thus, appropriate values for the number of "non-improving" pairs n are very small and may vary from 1 to 10. To achieve the best results, this parameter can be tuned specifically for each dataset since it may vary depending on different dataset properties. In case when specific configuration tuning is impossible we would suggest setting this parameter to 10, for the reason that when a significant number (such as 10) of concurrent pairs does not improve classifier performance, there are very few chances that any of the following less dependent pairs will make a significant improvement. For dependence score threshold values, we used the χ^2 *critical*[3] and *t-critical*[4] values for significance level *0.01* according to the tested dependence type.

A pseudo-code for this clustering algorithm is presented in Fig. 3.

5.2 LPBR method

We defined and examined a method that combines BR and LP approaches for multi-label classification to the defined independent groups of labels. One of the main disadvantages of the regular BR approach commonly applied to multi-label classification is that dependencies among labels are ignored. The main limitation of the LP approach is data sparseness and the large number of class combinations. Our approach involves a combination of these common methods and eliminates disadvantageous elements. Once we have identified the dependent labels and clustered them into independent groups, we divide the classification responsibility in the following way:

[3]Critical value of χ^2 statistic with one degree of freedom is 6.635 for the significance level 0.01.

[4]Critical value of t statistic with nine degrees of freedom is 3.25 for the significance level 0.01.

Dependent labels clustering

input: D – training set; $\mathcal{Y}$ – a set of labels $\{\lambda_1, \ldots, \lambda_L\}$; *pairScoreList* – ordered list of label pairs dependence scores; F - single-label classification algorithm

output: *clusters* – label set partitioning

1. *clusters* ← $\{\lambda_0\}, \{\lambda_1\}, \ldots, \{\lambda_L\}$
2. *acc* ← LPBRclassifier(*clusters*, F).evaluate(D)
3. *i*=0
4. REPEAT
5. *labelsPair* ← *pairScoreList*.getPair(i)
6. *newClusters* ← **updateCluster(*clusters, labelsPair*)**
7. *newAcc* ← LPBRclassifier(*newClusters*, F).evaluate(D)
8. IF *newAcc* >= *acc* THEN *clusters* ← *newClusters*
9. i=i+1
10. UNTIL (stop condition)
11. RETURN *clusters*

updateCluster(*clusters, labelsPair*)

input: *clusters* – label set partitioning; *labelsPair* – a pair of labels to be clustered

output: *newClusters* – new label set partitioning

1. *i* ← *labelsPair*.getFirstLabel
2. *j* ← *labelsPair*.getSecondLabel
3. IF *i* and *j* are single labels THEN *newClusters* ← *clusters* + $\{\lambda_i, \lambda_j\} - \{\lambda_i\} - \{\lambda_j\}$
4. IF *i* ∈ *groupA* and *j* is single label THEN *newClusters* ← add label *j* to *groupA*
5. IF *j* ∈ *groupA* and *i* is single label THEN *newClusters* ← add label *i* to *groupA*
6. IF *i* ∈ *groupA* and *j* ∈ *groupB* THEN *newClusters* ← join *groupA* and *groupB* into one cluster
7. RETURN *newClusters*

Fig. 3 A pseudo-code for the dependent labels clustering algorithm

- the BR approach to the independent groups of labels is applied without the limitation of ignored dependencies;
- the LP approach to classification into labels within the groups of dependent labels is applied without incurring the problem of a large number of class combinations (since it is applied to a group with a limited, potentially small, number of classes).

Following is a precise description of the whole process:

For each independent group of labels, one single LP classifier is independently created to determine the relatedness of each instance to the labels from a group. In the case of a group including only one category, the classifier is binary. However, if a group consists of k labels, the classifier is, actually, a single-label multi-class classifier with 2^k labels. Note that k is the number of maximum labels within one group and can be controlled by the model designer. Eventually, the final classification prediction is determined, similarly to the BR approach, by combining labels generated by each single LP and BR classifier.

On the one hand, this approach is more powerful than the regular BR approach, because it does not make the independence assumption which is wrong at times and, on the other, allows simple multi-label classification using any readily available single-label classifier.

A pseudo-code for LPBR model construction and labels prediction processes is presented in Fig. 4.

5.3 LPBR discussion

Dembczynski et al. (2010a) showed that the BR approach estimates marginal label probabilities and thus is tailored for the Hamming loss measure; the LP approach estimates the joint

LPBR method – model construction

input: Ω - label set partitioning (i.e. list of label subsets); F - single-label classification algorithm

output: *modelsArray* - LPBR model (an array of classification models for each label *subset)*

1. FOR each label *subset* in Ω
2. IF *subset* contain a single label *i*
3. THEN *modelsArray*[*subset*] = build BR model for the label *i* using the F algorithm
4. ELSE *modelsArray*[*subset*] = build LP model for all *subset* labels using the F algorithm
5. ENDFOR
6. RETURN *modelsArray*

LPBR method – classification prediction for unseen example x

input: *x* – unseen example; Ω - label set partitioning; *modelsArray* - LPBR model
output: *Y* – array of predicted labels

1. FOR each *subset* of labels in Ω
2. *prediction* = *modelsArray*[*subset*].classify(*x*)
3. FOR each *label* in the *subset*
4. *Y*[*label*] = *prediction*[*label*]
5. ENDFOR each *label*
6. ENDFOR each *subset*
7. RETURN *Y*

Fig. 4 A pseudo-code for the LPBR method

distribution of the whole label subset and thus is tailored for the subset accuracy measure. The authors also show that in case of conditionally independent labels, both BR and LP approaches are supposed to perform equally well.

In this paper we claim that the LPBR approach can improve the classification performance of both the LP and BR methods in terms of the accuracy measure on datasets where part of the labels are dependent. We claim as well that LPBR might be beneficial in terms of subset accuracy, also, in cases where there is a small training set without sufficient examples for all labels combinations.

In this section two reasons for the above conjectures are presented.

We believe that the most common case in practice is datasets where the partial dependencies among labels exist. In this case, given a dataset and a label set partitioning into groups of dependent labels, LPBR algorithm applies BR method on each independent label and LP on each group of dependent labels. For the conditionally independent labels, maximizing the marginal probabilities (by BR) is equivalent to maximizing their joint probability. And for conditionally dependent labels their joint probability is directly maximized by applying LP. Thus, the maximal conditional joint probability of the whole label set is estimated. At the same time applying LP (or BR) separately on the whole set of labels will maximize the conditional joint (or marginal) probability of both dependent and independent labels. This can lead to classification prediction different from that estimated by LPBR. An example of such a case is presented in Fig. 5.

Assuming that label dependencies supplied to LPBR were identified correctly, the LPBR prediction is expected to be more accurate than that of LP and BR separately.

The cases when all labels are independent or oppositely all labels are dependent are the borderline cases of LPBR in which it will apply the BR or LP strategy correspondingly for all labels and will predict the results same to BR or LP.

Consider a problem in which three labels $\lambda_1, \lambda_2, \lambda_3$ are to be predicted. Assume that: (1) labels λ_1 and λ_2 are conditionally dependent on X and label λ_3 is conditionally independent on X and (2) the joint distribution $p(X, \lambda_1, \lambda_2, \lambda_3)$ on $X \times Y$ is given as in the following table (a):

(a)

λ_1	λ_2	λ_3	Px(Y)
0	0	0	0.15
0	0	1	0.1
0	1	0	0.15
0	1	1	0.2
1	0	0	0.05
1	0	1	0.25
1	1	0	0
1	1	1	0.1

(b)

λ_1	λ_2	Px(Y)
0	0	0.25
0	1	0.35
1	0	0.3
1	1	0.1

λ_1	Px(Y)
0	0.6
1	0.4

λ_2	Px(Y)
0	0.55
1	0.45

λ_3	Px(Y)
0	0.35
1	0.65

Based on the given the joint distribution the prediction by each classification method would be as following: LP→(1,0,1); BR→(0,0,1); LPBR→(0,1,1). See the derived distribution tables on the side (b).

Fig. 5 Example of LPBR vs. LP and BR methods prediction

Although, the above justification of the LPBR method refers to conditional label dependence, in this paper we consider unconditional label dependence as well, and compare between both types.

The next reason for the superiority of LPBR refers to the targeted evaluation measures. As shown in Sect. 3, the accuracy measure can be considered as a kind of "golden mean" between subset accuracy and Hamming loss measures. Thus, the LPBR method, which interpolates between these two end-point measures, should perform well for the accuracy measure. The power of the LP classifier can suffer in the case of a few, if any, training examples for some label combinations. In such cases, separating the labels of these combinations into different subsets, which are treated independently of each other, is an advantage of the LPBR approach, allowing it to provide more accurate results. These hypotheses were tested by empirical experiments discussed below.

5.4 Computational complexity

The complexity of the dependent labels identification step varies between the ChiDep and ConDep methods. Comparison of computational time analysis for these methods is presented in Sect. 7.1.3. This section analyzes the computational complexity of the ChiDep\ConDep approaches after the dependent labels identification step.

The proposed clustering approach searches the space between the BR and LP models in an optimized method such that if there are no dependencies found, the result of ChiDep\ConDep is the same as the BR result. If all labels are interdependent, then the result is the same as that of LP. Thus, the complexity of the ChiDep\ConDep algorithm depends on the number of dependent label pairs that have been identified within the dataset and on the complexity of the underlying learner. If the complexity of the single-label base classifier is $O(f(L, |A|, |D|))$, where L is the number of labels, |A| is the number of attributes and |D| is the number of examples in training set. The ChiDep\ConDep's 'best case' complexity is equal to that of BR, that is $O(L^* f(2, |A|, |D|))$. The 'worst case' complexity is equal to that of LP, that is $O(f(2^L, |A|, |D|))$. Note, that value of 2^L is bounded by |D|, as there cannot be more label combinations than training examples.

We assume that in most regular cases there are partial dependencies among labels within the dataset and ChiDep\ConDep will stop much earlier before it reaches the 'worst case'. Thus, ChiDep\ConDep's complexity is approximated as follows: at each step, when two groups are clustered and a new set of labels is created, an evaluation of the new model is performed. It is important to note that the difference between the new model and the previous one is only in the newly clustered group. This allows us to optimize the calculation time by reusing models of the unchanged subsets, which were constructed at previous steps. Thus, reducing each step computation time to $O(f(2^{k_s}, |A|, |D|))$, where k_s is the size of the new cluster at step s which increases from 2 to L as the algorithm proceeds. This results in a total complexity of $O(L^* f(2, |A|, |D|)) + O(sf(2^{k_s}, |A|, |D|))$, where s is the number of clustering steps actually performed by the algorithm. The whole expression can be replaced by a general one $O((L + s) f(2^{k_s}, |A|, |D|))$.

Note that number of classes to be considered at each step (2^{k_s}) is actually limited by the number of distinct label combinations in dataset L_{DC} (see Sect. 6.1 and Table 2) and typically is relatively small. Also, it is possible to limit both parameters k and s at the application stage.

5.5 Ensemble framework for ChiDep and ConDep algorithms

Ensemble methods are well known for their capability to improve the prediction performance over a single classifier (Rokach and Maimon 2005). We follow this approach and gather several different ChiDep\ConDep classifiers into a composite ensemble model which is expected to further improve overall classification accuracy. For this purpose, a large number (i.e. 50000) of possible label sets partitions is randomly generated and a score for each partition is computed according to the dependence (i.e. χ^2 or t-value) score of all label pairs within the partition. The top m label set partitions are selected as members of the ensemble. The dependence scores of pairs of labels (a, b) are normalized, such that:

$$ns_{(a,b)} = score_{(a,b)} - score_{critical}$$

This normalization assures that all pairs of dependent labels receive a positive score and pairs of independent labels a negative score. The overall partition score is then calculated by summing up the scores of all pairs whose labels are in the same group and subtracting the scores of all pairs whose labels are in different groups of the set, such that:

$$Partition\ \text{“}dependence\text{”}\ score = \sum ns_{(i,j)} - \sum ns_{(q,r)},$$

where i, j labels are in the same group; and q, r labels are in different groups. For example the "dependence" score of the label set partition $\{\{a, c\}; \{b, d, e\}\}$ is:

$$\begin{aligned} Total = {} & ns_{(a,c)} + ns_{(b,d)} + ns_{(d,e)} + ns_{(b,e)} - ns_{(a,b)} \\ & - ns_{(a,d)} - ns_{(a,e)} - ns_{(c,b)} - ns_{(c,d)} - ns_{(c,e)} \end{aligned}$$

Finally, we choose m distinct sets with the highest scores to be included in the ensemble. The m is a user-specified parameter defining the number of classification models in the ensemble.

A pseudo-code for this algorithm is presented in Fig. 6.

For classification of a new instance, binary decisions of all models for each label are averaged and the final decision is taken. All labels whose average is greater than a user-specified threshold t are returned as a final classification result.

Ensemble of ChiDep/ConDep algorithm

input: D – training set; $\mathcal{Y}$ – a set of labels $\{\lambda_1, \ldots, \lambda_L\}$; F - single-label classification algorithm; m – number of models for ensemble; (for the diverse version: N-number of high scored partition to consider; k-percent of most different partitions to consider)

output: *model*s - array (ensemble) of LPBR classification models

1. create R random label set partitions (R should be large, i.e. 50000)
2. *weightsMatrix* ← compute normalized dependence sores of all label pairs
3. FOR each *partition*
4. *totalScore* ← **computePartitionScore(*partition*, *weightsMatrix*)**
5. ENDFOR
6. *labelSetPartitions* ← selec*t m* partitions with highes*t totalScore*
   OR for the diverse version:
   labelSetPartitions ← **selectDiversePartitions (*m*, *N*, *k*)**
7. *i*=0
8. FOR each *partition* from the *labelSetPartitions*
9. *models*[*i*] ← LPBRclassifier(*partition*, F).build(D) // the algorithm is presented on Fig. 4
10. *i=i+1*
11. ENDFOR
12. RETURN *models*

computePartitionScore(*partition*, *weightsMatrix*)

output: *w* – partition "dependence" score

1. *w*=0
2. FOR each label pair (λ_i, λ_j), where i=0,…L-1 and j=i+1,…L
3. IF λ_i and λ_j are in the same group THEN *w= w+weightsMatrix.getScore(i,j)*
4. IF λ_i and λ_j are in different groups THEN *w= w-weightsMatrix.getScore(i,j)*
5. ENDFOR
6. RETURN *w*

selectDiversePartitions(m, N, k)

output: *selectedPartitions* – array of label set partitions for ensemble classifier

7. *candidatePartitions* ← select *N* partitions with highest *totalScore*
8. *distanceMatrix* ← compute distance between each two partitions from *candidateModels*
9. *selectedModels*[0] ← select from *candidateModels* a partition with highest *totalScore*
10. FOR i=1 to *m*
11. FOR each partition *c* in *candidateModels*
12. FOR each partition *s* in *selectedModels*
13. *dist*[*s*] ← compute distance between c and s
14. ENDFOR each *s*
15. $minDist_c$ ← getMinimalValue(*dist*)
16. ENDFOR each *c*
17. *bestCandidateModels* ← select *k* percents of *candidateModels* with highest *minDist*
18. *selectedModels*[i] ← select from *bestCandidateModels* a partition with highest *totalScore*
19. ENDFOR
20. RETURN *selectedModels*

Fig. 6 A pseudo-code for the ChiDep/ConDep ensemble algorithm

Ensemble model diversity Accuracy of the ensemble classifier might be further improved by selecting the most different from the highly scored models. This property has been demonstrated in other ensemble settings (Rokach 2010). Thus we have defined a strategy for selecting more diverse models for participation in the ensemble than simply selecting the m models with the highest score.

For this purpose we utilize the distance function defined in Rokach (2008). For each pair of labels l_i, l_j the distance function adds "1" to the total distance if both labels belong to

the same subset in one partitioning structure and to different subsets in the other partitioning structure. We defined the following procedure for selecting the diverse models from among the highly scored ones.

1. Compute the distance matrix between all pairs of N partitions with highest "*dependency*" scores. Later we will refer to these N high-scored partitions as the set of "*candidate*" models.
2. Select the label set partition with the highest dependence score and add it to the set of "*selected*" models for participation in the ensemble.
3. Find the minimal distance $d_{i,min}$ from each one of the "*candidate*" models to all "*selected*" models.
4. Sort all "*candidate*" models in descending order of their d_{min} value (in step 3) and select k percent of the partitions with the highest d_{min}. This step is intended to choose k percents of the "*candidate*" models that are most different from the "*selected*" models. We refer to this set of models as "*best candidates*".
5. From the "*best candidates*" set, select the partition with the highest dependence score and add it to the set of "*selected*" models.
6. Repeat steps 3–5 until the number of the models in the set of "selected" models reaches m.
7. Return the "*selected*" set of models for participating in the ensemble.

A pseudo-code for the "diverse" version of the algorithm is presented in the procedure SelectDiversePartitions of Fig. 6.

This procedure allows us to trade-off between "diversity" and "dependency" scores among the selected models. Let us clarify how parameter values N and k influence the model selection process and level of "diversity"–"dependency" among the ensemble models. Parameter N defines the number of high-scored partitions which would be considered as "candidates" for ensemble. As higher this number as more diverse but less "dependent" partitions would be selected. For example for a dataset with 6 labels there are 172 possible distinct label-set partitions which dependence score may vary from high positive to high negative numbers. Thus, when defining $N = 100$ more than half of all possible partitions will be considered and some of them will probably have negative "dependency" scores. However, for a dataset with 14 labels there are above 6300 possible distinct label-set partitions and most probably all of the 100 high scored ones will have high positive "dependence" scores. Thus, for datasets with small number of labels and\or low dependency level among the label, relatively small values (between 20 and 100) for N should be considered. However, for datasets with large number of labels and\or higher dependency level among the labels, higher values of N (100 and above) are likely to perform better. Parameter k allows defining dynamically a threshold value for models which are "different enough" from already selected ones. For example, given that $N = 100$ setting k to *0.2* means that all 20 (20 percent of 100) of the most different from all the currently selected models will be considered as enough different and finally one of them having the highest dependency score will be added to ensemble. Larger values of k are expected to reduce the level of diversity among the selected models. Clearly, the "best" values for these parameters are dependent on dataset properties. Thus, in order to achieve the best performance, we recommend calibrating these parameters for each dataset specifically.

In this research we carried out a variety of global calibration experiments in order to define the appropriate default values which would allow parameters to perform sufficiently for most datasets. The selected values are presented in the following section.

Table 2 Datasets used in the experiment: information and statistics

Name	Domain	Labels	Train	Test	Attributes	L_{CARD}	L_{DC}
Emotions	Music	6	391	202	72*	1.869	27
Scene	Image	6	1211	1196	294*	1.074	15
Yeast	Biology	14	1500	917	103*	4.237	198
Genbase	Biology	27	463	199	1186	1.252	32
Medical	Text	45	645	333	1449	1.245	94
Enron	Text	53	1123	579	1001	3.378	753
Slashdot	Text	22	2348	1434	1079	1.18	156
Ohsumed	Text	23	8636	5293	1002	1.66	1147
tmc2007[a]	Text	22	21519	7077	500	2.158	1341
rcv1(subset1)	Text	101	3000	3000	944*	2.88	1028
Mediamill	Video	101	30993	12914	120*	4.376	6555
Bibtex	Text	159	4880	2515	1836	2.402	2856

[a]tmc2007-500 version of dataset was used

*Notes numeric attributes

6 Empirical evaluation

This section presents the setup of empirical experiments we conducted to evaluate the proposed approaches. It describes the datasets and learning algorithms used during the experiments, and presents the measures used for evaluation.

6.1 Datasets

We empirically evaluated the proposed approach by measuring its performance on twelve benchmark multi-label datasets[5] from different domains and variable sizes. All datasets along with their properties are listed in Table 2.

Besides the regular classification properties, such as label set and feature set size and the number of train and test examples, we present specific statistical information for multi-label classification. This information includes: (1) Label Cardinality (L_{CARD})—a measure of "multi-labeled-ness" of a dataset introduced by Tsoumakas et al. (2010) that quantifies the average number of labels per example in a dataset; and (2) Label Distinct Combinations (L_{DC})—a measure representing the number of distinct combinations of labels found in the dataset.

As shown in Table 2, six small to medium size datasets and six large size datasets are included in the experiment. The datasets are ordered by their approximate complexity and roughly divided (by horizontal line) between regular and large size datasets. From the rcv1 corpus only the subset 1 dataset was used. In addition, dimensionality reduction has been performed on this dataset as in Zhang and Zhang (2010), such that the top 944 features with highest document frequency have been retained.

[5]The datasets are available at http://mlkd.csd.auth.gr/multilabel.html, http://meka.sourceforge.net/#datasets, http://davis.wpi.edu/~xmdv/datasets/ohsumed.html.

6.2 Procedure

We implemented the ChiDep and ConDep methods and their ensemble versions (the implementation is written in Java using Weka[6] and Mulan[7] open-source Java libraries). The new algorithms have been integrated into the Mulan library. The tests were performed using original train and test dataset splits. The internal selection of a model label set was carried out using 3-fold cross-validation over the large training sets (namely, slashdot, ohsumed, tmc2007, rcv1, mediamill and bibtex) and 10-fold over the rest training sets. The overall cross-validation process was repeated ten times for the training sets with less than 500 examples (namely, emotions and genbase). For the rest of training sets we followed the approach proposed by Kohavi and John (1997). According to this approach the number of repetitions was determined on the fly by looking at the standard deviation of the accuracy estimate. If the standard deviation of the accuracy estimate was above 1 % and ten cross-validations have not been executed, another cross-validation run was executed. Although this is only a heuristic, Kohavi and John claim that this heuristic seems to work well in practice and avoids multiple cross-validation runs for large datasets.

First, we observed the results achieved by ChiDep and ConDep algorithms and compared the level of performance achieved by modeling unconditional vs. conditional label dependencies. We noticed that there is no benefit from modeling conditional dependencies (as presented on Sect. 7.1). Taking into consideration that modeling conditional dependencies is very computationally expensive and not feasible for large datasets, we continued the evaluation with the ChiDep (modeling unconditional dependencies) method.

We compared the results achieved by the ChiDep (denoted CD) approach to those of the standard multi-label classification methods such as the BR and LP approaches, and, also to some state-of-the-art methods addressed in the Related Work section: HOMER (HO), MLStacking (2BR), Pruned Sets (PS) and Classifier Chains (CC).

The RAkEL (RA) method is an ensemble algorithm combining votes from a number of multi-label classifiers to a single classification decision. We thus compare it to the ensemble versions of the ChiDep (CDE), Classifier Chains (ECC) and Pruned Sets (EPS) methods. Due to the random nature of the RAkEL and ECC algorithms and the partially random nature of CDE, results vary between runs. Thus, we averaged the results of each one of these algorithms over five distinct runs on each dataset. Consecutive numbers from 1 to 5 were used as initialization seed for the random number generator allowing reproducibility of the experimental results.

All methods were evaluated using the Mulan library. All the algorithms were supplied with Weka's J48 implementation of a C4.5 tree classifier as a single-label base learner. We compare the results using the evaluation measures presented in Sect. 3. The statistical significance of differences in algorithm results was determined by Friedman test (Demsar 2006) and post-hoc Holm's procedure for controlling the family-wise error in multiple hypothesis testing.

6.3 Parameter configuration

All configurable parameters of the participating algorithms were set to their optimal values as reported in the relevant papers. For HOMER, its balanced k means version with $k = 3$

[6]Software is available at http://www.cs.waikato.ac.nz/ml/weka/.

[7]Software is available at http://mulan.sourceforge.net/.

was set. The 2BR was supplied with J48 for both base-level and meta-level binary classification algorithms. The PS's p and s parameters require tuning for each dataset. We used the parameters chosen by the authors for datasets presented in the PS paper (Read et al. 2008). For other datasets we set $p = 1$ (as the dominant value among chosen values in the PS paper) and s was computed by PS's utility (from Meka[8] library) according to label cardinality and the number of labels in the dataset as recommended by the authors. BR, LP and CC do not require parameters. For the ChiDep and ConDep algorithms, we set the n parameter to 10 for all datasets, for the reason mentioned in Sect. 5.1.3. The target evaluation measure was set accordingly to each one of the considered measures, namely accuracy, subset accuracy, micro-averaged F-measure and Hamming loss measure. The χ^2 and t *critical* values were set to 6.635 and 3.25 respectively, as described in Sect. 5.

Ensemble methods configuration The number of models participating in the ensemble classifier is supposed to influence the predictive accuracy of the classifier. For the sake of a fair comparison, we wanted to evaluate ensemble models of an equivalent complexity. To achieve this, we configured all of the ensemble algorithms in the experiment to construct the same number of distinct models. ChiDep\ConDep ensemble algorithms construct a varying number of models for each label set partition according to the number of dependent labels groups at each partition. Thus, for the ChiDep\ConDep ensemble we set the number of label set partitions m to 10 (as frequently used for the number of classifiers in an ensemble) and averaged the number of distinct models constructed by the ensemble across all the random runs. This number, presented in Table 13, was taken as the base number of models for all ensemble methods. RAkEL, ECC and EPS were configured to construct the same number of distinct models. RAkEL and EPS allow supplying the number of desired models via the constructor. However, ECC aggregates the number of CC classifiers and each one of them constructs a number of models equal to the number of labels in the dataset. Thus, for ECC we divided the desired number of models by L and rounded it up. The result was set as the number of CC classifiers participating in ECC. For all ensemble methods the majority voting threshold was set to a commonly used intuitive value of *0.5*.

Other parameters of ensemble methods were configured as follows. The RAkEL's k parameter was set to 3. ECC does not require additional (to the number of models and threshold) parameters. For the EPS, at each dataset p and s, parameters were set to the same values as those used for the PS algorithm. The diverse version of the ChiDep ensemble (CDE-d) was supplied with $N = 100$ and $k = 0.2$ (i.e. twenty percents), accordingly to the results of calibration experiments described in Sect. 7.2.

7 Experimental results

This section presents the results of the evaluation experiments that we conducted. Initially, we compare results of the algorithms utilizing unconditional (ChiDep) vs. conditional (ConDep) dependence identification and select the best one for further comparison to other baseline and state-of-the-art multi-label algorithms. Then we compare the selected algorithm and its ensemble version to other single-classifier and ensemble algorithms accordingly.

[8]Software is available at http://meka.sourceforge.net/.

Table 3 Predictive performance of ChiDep and ConDep algorithms and their ensemble versions on various datasets

Dataset	Accuracy				micro-avg. F-measure			
	Single-classifier		Ensemble		Single-classifier		Ensemble	
	ChiDep	ConDep	ChiDep	ConDep	ChiDep	ConDep	ChiDep	ConDep
Emotions	43.28	**43.84**	**51.05**	44.87	**60.31**	59.25	**63.87**	59.66
Scene	**57.71**	56.76	**59.91**	59.29	60.4	60.4	**66.06**	65.72
Yeast	42.88	42.88	**49.53**	44.73	**56.92**	56.72	**62.31**	58.9
Genbase	**99.16**	98.66	98.66	98.66	**98.97**	98.77	98.77	98.77
Medical	72.52	**72.9**	**71.34**	71.17	**79.36**	77.99	**78.97**	78.92
Enron	**41.14**	38.73	**42.91**	40.46	**51.94**	50.63	**54.81**	53.58
Slashdot	38.73	**38.78**	38.42	**38.69**	48.93	**49.56**	47.81	**49.**59
Dataset	Subset accuracy				Hamming loss			
	Single-classifier		Ensemble		Single-classifier		Ensemble	
	ChiDep	ConDep	ChiDep	ConDep	ChiDep	ConDep	ChiDep	ConDep
Emotions	**22.28**	17.82	**26.24**	13.86	**25.41**	25.99	**22.77**	25.66
Scene	**53.68**	50.59	**54.26**	50.42	14.51	**14.35**	**11.4**	12.12
Yeast	**14.94**	12.00	**14.76**	10.45	26.59	26.59	**21.5**	24.39
Genbase	**98.49**	97.49	97.49	97.49	**0.09**	0.11	0.11	0.11
Medical	**65.47**	64.26	**62.93**	62.76	1.17	1.17	**1.11**	1.12
Enron	**12.95**	11.92	**12.55**	10.71	**5.38**	5.40	**5.08**	5.17
Slashdot	**33.47**	33.12	**33.66**	33.1	4.36	**4.20**	4.4	**4.21**

7.1 Conditional vs. unconditional dependencies

In this section the results of the new algorithm utilizing two different methods for dependence identification, are presented, namely, ChiDep using the chi-square test for unconditional dependence identification and ConDep for estimating conditional label dependencies.

Table 3 presents the results of the ChiDep and ConDep algorithms and their ensemble versions on small and medium size datasets for each evaluation measure. On the rest of datasets, the identification of conditional dependencies takes more than a week of computation and thus is generally not feasible under existing time constraints and available resources (see footnote 9 for the characteristics of the utilized hardware). The best result for each measure on a particular dataset is separately marked in bold for single and ensemble methods.

Consider first the single algorithms. It can be observed that the ChiDep algorithm outperforms ConDep on all datasets for subset accuracy measure and on the most datasets for the F-measure. However, for the accuracy and Hamming loss measures the ConDep outperforms the ChiDep on 3 and 2 datasets correspondingly. For the ensemble algorithms we observe that ChiDep outperforms ConDep for all measures on almost all datasets.

The above results indicate that among the two methods of modeling dependencies, the modeling of unconditional dependencies (ChiDep) is superior in terms of subset accuracy and F-measure. In addition, its ensemble version is superior in terms of accuracy and Hamming loss as well.

To verify these quite surprising results, we performed some experiments on artificial datasets where various types of dependencies are simulated. The results of these simulation

Fig. 7 Data patterns used for introducing conditional dependence

	features								*labels*							
#	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	1	0	1	.	.	.	.	.	.	1	.	1	.	.	.	[1]
2	.	1	.	0	.	.	.	1	.	.	.	.	0	1	[1]	.
3	.	1	.	1	.	.	0	.	1	0	[1]	.	.	.	.	.
4	.	.	0	1	1	.	.	.	.	1	[0]	.	.	.	.	.
5	0	.	.	.	0	1	.	.	0	.	.	1	.	[1]	.	.

experiments (presented in the next section) correspond to our findings on the benchmark datasets.

Furthermore, these results correspond with the results presented in Ghamrawi and McCallum (2005), where two models were evaluated, one of which parameterizes conditional dependencies, the CMLF, and another which parameterizes unconditional dependencies, the CML. The results presented in their work show that the CMLF model is not better and in many cases even worse (in terms of micro and macro-averaged F-measure and subset accuracy) than the CML model.

Taking into consideration that the computational cost of conditional dependencies is also much higher than that of unconditional dependencies (as can be observed from the times comparison reported in Sect. 7.1.3), we conclude that modeling unconditional dependencies is sufficient enough for solving multi-label classification problems. The rest of the comparisons are performed with ChiDep and an ensemble of ChiDep algorithms.

7.1.1 Experiments on artificial datasets

In this section the methods for conditional and unconditional label's dependence identification are compared on artificially created multi-label datasets.

We experimented with two types of artificially generated datasets. In the first type of datasets, a predefined pattern was introduced to a set of randomly generated sequences. In the second type, artificial datasets were generated according to the models defined by Dembczynski et al. (2010a).

We hypothesize that in cases where label's dependence is accurately expressed, the conditional method might perform better than the unconditional. However, in non-ideal cases where various levels of noise and less dependence examples are present, the unconditional dependence identification would perform better than the conditional. These hypotheses are tested in the experiments below.

Experiments on the predefined patterns data In this experiment we generated 100,000 instances with 8 attributes and 8 labels. The instances were randomly drawn from the set of binary vectors $E = \{0, 1\}^{16}$. For introducing conditional dependence, the five patterns presented in Fig. 7 were introduced to the data. If a record matches the defined pattern (point means that any value 0 or 1 is allowed), the specified label, written in the quadratic parenthesis, is set to the defined value. Each one of the defined patterns was introduced to the data separately from others to avoid the effects of mutual interaction between the patterns. In total five datasets, one for each pattern, were created.

For each one of these datasets, two distinct experiments were conducted. In one of the experiments (referred as experiment-1), 9,000 random records (i.e., not matching the pattern) and 1,000 records matching the pattern were selected from the full set of generated records. These 10,000 instances were used as a training set. For the remaining experiment (referred as experiment-2), 500 "pattern" records were removed from the experiment-1 dataset, thus

Table 4 Dependent label pairs identified by ChiDep and ConDep algorithms on training data

Pattern number	Experiment-1				Experiment-2			
	ChiDep		ConDep		ChiDep		ConDep	
	pair	p-value	pair	p-value	pair	p-value	pair	p-value
1	**[2, 8]**	**1.2E-24**	**[2, 8]**	**1.0E-06**	**[2, 8]**	**2.6E-09**	**[2, 8]**	**9.0E-08**
	[4, 8]	**1.6E-21**	**[4, 8]**	**3.9E-05**	**[4, 8]**	**1.9E-07**	**[4, 8]**	**4.7E-06**
	[2, 4]	**1.9E-12**	[2, 1]	4.2E-03			[8, 3]	1.9E-03
			[5, 6]	8.1E-03				
			[8, 3]	9.7E-03				
2	**[5, 7]**	**2.8E-24**	**[5, 7]**	**1.8E-06**	**[5, 7]**	**3.6E-09**	**[5, 7]**	**1.8E-05**
	[6, 7]	**1.8E-17**	**[6, 7]**	**1.7E-04**	**[6, 7]**	**2.2E-05**	[7, 1]	1.0E-03
	[5, 6]	**6.0E-09**					**[6, 7]**	**1.3E-03**
3	**[2, 3]**	**9.8E-26**	**[2, 3]**	**5.8E-07**	**[2, 3]**	**5.9E-10**	**[1, 3]**	**5.7E-07**
	[1, 3]	**4.0E-14**	**[1, 3]**	**4.4E-04**	**[1, 3]**	**1.4E-03**	**[2, 3]**	**1.3E-05**
	[1, 2]	**4.3E-12**	[5, 7]	1.6E-03			[6, 3]	4.4E-04
							[6, 2]	6.1E-03
							[5, 4]	6.8E-03
4	**[2, 3]**	**2.6E-18**	**[2, 3]**	**4.0E-08**	**[2, 3]**	**2.3E-05**	**[2, 3]**	**2.5E-07**
							[2, 5]	2.9E-03
5	**[4, 6]**	**7.0E-24**	**[1, 6]**	**1.1E-05**	**[4, 6]**	**1.3E-08**	**[1, 6]**	**5.6E-06**
	[1, 6]	**2.1E-22**	**[4, 6]**	**2.6E-05**	**[1, 6]**	**6.5E-08**	**[4, 6]**	**7.2E-05**
	[1, 4]	**6.5E-12**	[1, 8]	7.6E-03			[8, 4]	6.2E-03
							[7, 6]	6.4E-03
							[8, 2]	8.3E-03

reducing the number of dependency examples. The remaining 9,500 instances were used for training. In both experiments, the two algorithms, ChiDep, using the chi-square test for unconditional dependence identification, and ConDep, estimating conditional label dependencies, were tested on 1,000 records of the corresponding pattern, which were generated using another set of random vectors $E' = \{0, 1\}^{16}$.

The label's dependencies identified by each one of the methods on the training data are presented in Table 4. Pairs of the correctly identified dependent labels are marked in bold.

Consider the algorithm's results on pattern number 1, where labels 2, 4, and 8 are dependent. We can see that in the experiment-1, the "unconditional" (ChiDep) method identifies all three pairs of dependent labels. In the same experiment, the "conditional" (ConDep) method identifies only two pairs correctly. Additionally, the ConDep method misidentifies three other label pairs as dependent. In the experiment-2, both methods identify the same two (out of three) label pairs as dependent. However, ConDep again misidentifies another pair as dependent labels. The results on all the other patterns are quite similar. For experiment-1, the ChiDep method correctly identifies all pairs of dependent labels on all patterns, while the ConDep identifies only two out of three dependent pairs on all patterns except pattern number 4. In pattern 4, only one pair of labels (2 and 3) is dependent and is correctly identified by both methods. In experiment-2, both methods identify the same label pairs as dependent, however, the ConDep misidentifies some other pairs as dependent labels on all the patterns.

The results of these experiments demonstrate that on "noisy" datasets with different levels of dependence representations, the "unconditional" ChiDep method identifies a label's

Table 5 Predictive performance of ChiDep and ConDep algorithms on predefined patterns data

Measure	Experiment-1		Experiment-2					
	Pattern5		Pattern2		Pattern3		Pattern5	
	ChiDep	ConDep	ChiDep	ConDep	ChiDep	ConDep	ChiDep	ConDep
Accuracy	**0.5735**	0.5647	**0.5606**	0.5588	**0.5888**	0.5720	**0.5789**	0.5712
Subset Accuracy	**0.0340**	0.0250	**0.0270**	0.0220	**0.0260**	0.0250	**0.0240**	0.0220
Micro F-measure	**0.7213**	0.7136	**0.7110**	0.7091	**0.7362**	0.7210	**0.7284**	0.7212
Hamming Loss	**0.3140**	0.3160	0.3199	**0.3185**	**0.3113**	0.3116	0.3165	**0.3154**

dependencies more accurately than the "conditional" ConDep method. One of the problems with the ConDep method is the identification of spurious dependencies. We hypothesize that the origin of these misidentifications is the false positive of the statistical t-test, known in statistics as the Type 1 error. As can be seen from the results, the p-value of the misidentified pairs is relatively high and almost always much higher than that of the actual dependent pairs. An additional glitch of the ConDep method is that the actual dependent pairs are not always identified.

The results of the ChiDep and ConDep algorithms on the test data are presented in Table 5. In some cases the results of the algorithms are the same despite the fact that not all pairs were identified as dependent by the method for a conditional label's dependence identification. This is due to the clustering procedure (described in Sect. 5.1.3) in which two pairs are clustered in a group of three labels, if the classification performance is improved. The experiments where both methods performed exactly the same are omitted from the results in Table 5.

It can be seen that the ChiDep method was outperformed by ConDep on two experiments for the Hamming loss measure only. On the other hand, the ChiDep outperforms the ConDep in four experiments on almost all measures.

Experiments on independent, dependent, and combined datasets In Dembczynski et al. (2010a), two models for generating artificial three-label datasets are defined and used. In the first dataset, all the labels are conditionally independent, however the unconditional dependence exists between labels 2 and 3. In the second dataset, all the labels are conditionally dependent. We utilized these models for our experiments. The models, defined in Dembczynski et al. (2010a) are briefly presented below for reading convenience.

For each dataset, 10,000 instances were generated. The first (conditionally independent) dataset was generated by uniformly drawing instances from the square $\boldsymbol{x} \in [-0.5, 0.5]^2$. The label distribution is given by the product of the marginal distributions defined by $\boldsymbol{P}_{\boldsymbol{x}}(y_i) = 1/(1 + \exp(-f_i(\boldsymbol{x})))$, where the f_i are linear functions: $f_1(\boldsymbol{x}) = x_1 + x_2$, $f_2(\boldsymbol{x}) = -x_1 + x_2$, $f_3(\boldsymbol{x}) = x_1 - x_2$. This model generates conditionally independent labels. However, labels 2 and 3 are dependent marginally (unconditionally) as $f_2(\boldsymbol{x}) = -f_3(\boldsymbol{x})$. This dataset will subsequently be referred to as independent or conditionally independent.

The second (dependent) dataset was generated by drawing the instances from an univariate uniform distribution $\boldsymbol{x} \in [-0.5, 0.5]$. The label distribution is given by the product rule: $\boldsymbol{P}_{\boldsymbol{x}}(\boldsymbol{Y}) = \boldsymbol{P}_{\boldsymbol{x}}(y_1)\boldsymbol{P}_{\boldsymbol{x}}(y_2|y_1)\boldsymbol{P}_{\boldsymbol{x}}(y_3|y_1, y_2)$, where the probabilities are modeled by linear functions similarly as before: $f_1(x) = x$, $f_2(y_1, x) = -x - 2y_1 + 1$, $f_3(y_2, y_1, x) = x + 12y_1 - 2y_2 - 11$. This dataset will subsequently be referred to as dependent.

Additionally, for a simulation of "noisy" dependencies, we created a third dataset combined of the first two. The third dataset consists of two features and six labels. The feature's

Table 6 Dependence identification results on dependent (*left*) and independent (*right*) datasets. In *bold* mark the label pairs identified as dependent

Dependent dataset				Independent dataset			
ChiDep		ConDep		ChiDep		ConDep	
pair	p-value	pair	p-value	pair	p-value	pair	p-value
[1, 3]	**0.0E+00**	**[1, 3]**	**6.5E-17**	**[2, 3]**	**1.1E-06**	[2, 3]	2.3E-01
[2, 3]	**0.0E+00**	**[2, 3]**	**4.0E-15**	[1, 3]	6.3E-01	[1, 2]	3.2E-01
[1, 2]	**0.0E+00**	**[1, 2]**	**6.4E-15**	[1, 2]	8.8E-01	[3, 1]	5.3E-01

Table 7 Dependence identification results on combined datasets

10000 training examples				1000 training examples			
ChiDep		ConDep		ChiDep		ConDep	
pair	p-value	pair	p-value	pair	p-value	pair	p-value
[4, 6]	**0.0E+00**	**[4, 6]**	**1.1E-15**	**[4, 6]**	**5.6E-102**	**[4, 6]**	**2.3E-10**
[5, 6]	**0.0E+00**	**[5, 6]**	**5.1E-15**	**[5, 6]**	**3.9E-68**	**[5, 4]**	**7.9E-10**
[5, 4]	**0.0E+00**	**[5, 4]**	**2.1E-14**	**[5, 4]**	**2.9E-51**	**[5, 6]**	**1.5E-08**
[2, 3]	**1.1E-06**			**[2, 3]**	**1.1E-03**		
[1, 6]	1.2E-04						
[1, 5]	2.4E-04						
[2, 6]	2.2E-03						
[2, 4]	4.9E-03						

attributes and first three labels were generated according to the conditionally independent model and the next three labels were generated according to the dependent model, considering $x = x_2$. This dataset will subsequently be referred to as combined. On the combined dataset, we experimented with various training set sizes, namely, 10,000, 1,000, 500 and 300 instances.

The results of the label's dependence identification for the independent and dependent datasets are shown in Table 6 and for the combined dataset in Table 7.

As can be seen in Table 6, in the dependent dataset, both methods correctly identified all label pairs as dependent. In the case of independent data, the identification results are correct as well. The ConDep method identifies all the labels as independent. The ChiDep method identifies pairs [1, 3] and [1, 2] as independent and pair [2, 3] as dependent. Note that labels 2 and 3 are indeed unconditionally dependent. For the combined dataset sets with 500 and 300 train instances, both methods equally identified the three label pairs from the dependent model as dependent. Thus, these results are omitted from Table 7. The results presented are for the training sets of 10,000 and 1,000 instances. It can be seen that on both these training sets, the ConDep method correctly identifies the three label pairs from the dependent model as dependent. The ChiDep method identifies the same three pairs as the most dependent. It correctly identifies the pair of labels [2, 3] as depended, as well. Additionally, on the training set with 10,000 examples, four other pairs are identified as unconditionally dependent by the ChiDep method. These dependencies are caused by a mutual interaction between the two underling data models.

The classification performance of the ChiDep and ConDep methods was evaluated on all the generated datasets using a 3-fold cross-validation. The average results for each evaluation measure over test instances are reported in Table 8. Note that due to the cross-validation, the dependencies identification for each dataset was performed on two thirds of the instances for each fold.

Table 8 Classification results on independent, dependent, and combined datasets

Measure	Independent		Dependent		Combined (10000 train)		Combined (300 train)	
	ChiDep	ConDep	ChiDep	ConDep	ChiDep	ConDep	ChiDep	ConDep
Accuracy	**0.43**	0.4265	0.4795	0.4795	0.4155	**0.4166**	**0.3363**	0.3194
Subset Accuracy	0.1936	**0.1939**	0.4024	0.4024	**0.0795**	0.0786	**0.0267**	0.0233
Micro F-measure	**0.5836**	0.5774	0.5336	0.5336	0.5549	**0.5559**	**0.492**	0.4714
Hamming Loss	0.4231	**0.4217**	0.4127	0.4127	0.4168	0.4168	**0.4828**	0.49

Considering the classification results of the ChiDep and ConDep algorithms, we can see that on dependent data both methods perform equally for all evaluated measures. On the other hand, the results of ConDep in the independent dataset are slightly better for the subset accuracy and Hamming loss measures, while utilizing the unconditional dependence discovered between labels 2 and 3 by the ChiDep method results in a better performance in terms of accuracy and micro-averaged F-measure.

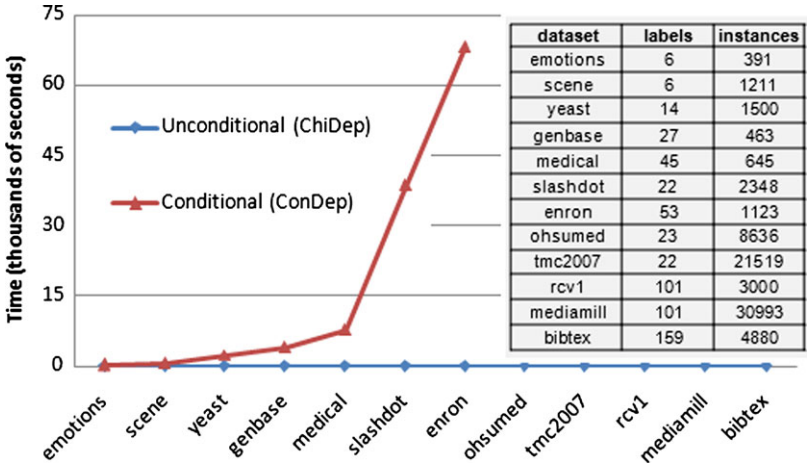
In the case of the combined dataset, we can see that on the large training set of 10,000 instances, the results of the two algorithms are very close. The ConDep slightly outperforms the ChiDep for accuracy and micro-averaged F-measures. Alternatively, ChiDep slightly outperforms the ConDep for the subset accuracy measure. However, on the small training set of 300 instances, the ChiDep significantly outperforms the ConDep on all evaluated measures. On the training sets of 1,000 and 500 instances, both methods perform equally for all evaluated measures, thus these results are omitted from Table 8.

Experiments conclusions Summarizing the experiments on artificial datasets where various dependence types and conditions were simulated, we draw the following conclusions. Both methods for conditional and unconditional dependence identification correctly identify the existing dependencies in datasets with well-expressed dependencies and on datasets with many examples for dependence. In such cases the classification results of both methods are either the same or very close to each other. However, on small or with a limited number of dependence examples datasets, the unconditional method is more accurate in dependence identification and consequently in classification predictions. In most of such cases, the conditional method does not identify all existing dependencies and additionally suffers from Type 1 errors, causing identification of spurious dependencies. These results correspond to our results on benchmark datasets and confirm that in practice, modeling of unconditional dependencies provides either very similar or more effective results than modeling of conditional dependencies.

7.1.2 Comparison of computational time for unconditional vs. conditional label dependence identification

As we already noted the computation of conditional dependencies is very time consuming. This section presents comparison of computational time required by the methods for conditional and unconditional label dependence identification. The time required for constructing the ordered list of dependent pairs by both methods is considered. Figure 8 presents those times in thousand of seconds (see footnote 9 for the characteristics of the utilized hardware).

It can be seen from the graph that time required for approximation of conditional label dependencies increases exponentially with the number of training examples in the dataset.



dataset	labels	instances
emotions	6	391
scene	6	1211
yeast	14	1500
genbase	27	463
medical	45	645
slashdot	22	2348
enron	53	1123
ohsumed	23	8636
tmc2007	22	21519
rcv1	101	3000
mediamill	101	30993
bibtex	159	4880

Fig. 8 Comparison of computational time (in thousands of seconds) for unconditional vs. conditional label dependence identification

For each one of the five largest datasets (namely, ohsumed, tmc2007, rcv1, mediamill and bibtex) this approximation took more than one week and thus could not be completed. On the other hand the time required by chi-square test applied on labels data for identification of unconditional label dependencies took about 4–5 seconds even on the largest datasets.

7.1.3 Identified label dependencies

Although, the ConDep algorithm concede to the ChiDep in predictive performance, we felt that it might be of interest to compare the pair of labels identified as dependent by each of the methods on benchmark datasets. Below we present and compare dependencies identified by both algorithms for each dataset. Number of label pairs identified as dependent by each one of the methods is summarized in Table 9. The "NF" mark indicates that execution of an algorithm was Not Feasible (i.e. took more than a week) under existing resources constraints. The dependence is identified at the significance level of $p = 0.01$ wherever it is not stated differently.

For the emotions dataset, the unconditional label dependence identification using the chi-square test identified 14 (from 15) label pairs as dependent. The only pair of independent labels is {amazed-surprised, happy-pleased}. According to our conditional dependence identification procedure, there are two dependent pairs of labels with a 0.01 significance level, and four more pairs are dependent at a 0.05 significance level. The two most conditionally dependent label pairs are {relaxing-calm, angry-aggressive} and {happy-pleased, angry-aggressive}, the next four (at $p = 0.05$) are {happy-pleased, sad-lonely}, {happy-pleased, quiet-still}, {amazed-surprised, quiet-still} and {amazed-surprised, relaxing-calm}. The {relaxing-calm, angry-aggressive} pair is also the most unconditionally dependent pair. For the scene dataset, all labels were found to be unconditionally dependent. But only five were found to be conditionally dependent at a $p = 0.01$ significance level; three more pairs were conditionally dependent at $p = 0.05$. The first five conditionally dependent label pairs are {Mountain, Urban}, {Beach, Urban}, {Sunset, Fall Foliage}, {Beach, Mountain} and {Beach, Field}. The {Mountain, Urban} pair is also the most unconditionally dependent pair. For the yeast dataset, 53 unconditionally dependent pairs and 16 conditionally dependent pairs were found from a total of 91 pairs. Also in this dataset, the first most dependent

Table 9 Number of label pairs identified as dependent by the methods for conditional and unconditional label dependence identification at the significance level of $p = 0.01$

Dataset	Total label pairs	Number of *unconditionally* dependent label pair	Number of *conditionally* dependent label pairs
Emotions	15	14	2
Scene	15	15	5
Yeast	91	53	16
Genbase	351	42	0
Medical	990	29	2
Enron	1378	136	6
Slashdot	231	50	1
Ohsumed	253	110	NF
tmc2007	231	178	NF
rcv1(subset1)	5050	582	NF
Mediamill	5050	1532	NF
Bibtex	1566	1157	NF

pair of labels is the same for both methods. As well all of the 16 conditionally dependent labels were also found to be unconditionally dependent. For the slashdot dataset, 50 unconditionally dependent pairs and only one {Idle, Games} conditionally dependent pair were found from a total of 231 pairs. Three more pairs {Games, Technology}, {Idle, Mobile} and {Science, Technology} were conditionally dependent at $p = 0.05$. Also in this dataset, the first most dependent pair of labels that was identified by both methods is the same. The situation was very similar for the genbase, medical and enron dataset. Details for these datasets can be seen in Table 9.

Summarizing these results, we conclude that in general there are many more label pairs identified as unconditionally dependent, than those identified as conditionally dependent. Although the results of both methods are very different, some correspondence between the identified dependent label pairs can be noticed. So, for example, in many cases the most dependent label pairs are the same for both methods. As well most of the conditionally dependent pairs are also identified as unconditionally dependent.

7.2 ChiDep vs. single-classifier algorithms

In this section, the results of the ChiDep algorithm are compared to other multi-label single-classifier algorithms. The results are presented in Table 10. The 'OOM' mark indicates that the algorithm did not finish the run because of Java OutOfMemoryException.[9] We verified the statistical significance of the results by a comparison of ChiDep to other methods using a Friedman test and post-hoc Holm's procedure at $p = 0.01$. The rank of the ChiDep algorithm result relative to all other algorithms for a specific dataset can be seen in the 'ChiDep rank' column. The average rank of each algorithm over all datasets for a certain evaluation measure is presented at the last row of each sub table.

[9]The tests were performed on 64bit Intel Core Quad CPU Q6600@2.40 GHz machine providing 4GB of RAM for each algorithm.

Table 10 Comparing ChiDep to other single-classifier algorithms on various datasets. [+], [-]-statistically significant improvement or degradation vs. ChiDep

Dataset	ChiDep rank	Accuracy						
		ChiDep	BR	LP[-]	HO[-]	2BR[-]	PS[-]	CC
Emotions	2	43.28	**43.84**	42.15	42.34	41.7	41.49	42.77
Scene	2.5	57.71	51.34	57.71	46.98	47.07	57.04	**59.45**
Yeast	4	42.88	42.26	39.84	43.27	**45.62**	40.76	43.17
Genbase	1	**99.16**	98.66	98.41	97.91	98.74	98.41	98.66
Medical	3	72.52	71.17	71.92	69.07	66.12	**73.22**	73.17
Enron	1	**41.14**	36.71	30.97	40.19	37.25	33.18	39.41
Slashdot	3	38.73	38.69	41.87	34.26	37.58	**42.28**	38.43
Ohsumed	1	**36.46**	35.84	32.14	34.37	35.19	32.18	35.89
tmc2007	1	**75.71**	75.12	75.38	73.75	59.56	75.08	74.16
rcv1(subset1)	2	6.88	**6.94**	6.26	6.49	6.8	6.41	6.7
Mediamill	2	36.66	**36.89**	33.72	36.46	35.08	33.66	36.47
Bibtex	2	29.62	**29.92**	25.17	24.23	25.92	OOM	28.79
Avg. rank	2.0	**2.0**	3.1	5.0	5.1	4.6	4.8	3.1

Dataset	ChiDep rank	Subset accuracy						
		ChiDep	BR[-]	LP	HO[-]	2BR[-]	PS	CC
Emotions	1.5	**22.28**	12.87	**22.28**	12.38	19.8	18.81	16.34
Scene	1.5	**53.68**	40.13	**53.68**	32.61	42.31	51.92	53.34
Yeast	1	**14.94**	6.43	11.89	6.98	9.49	12.43	14.29
Genbase	1	**98.49**	97.49	96.98	96.48	97.49	96.98	97.49
Medical	2	**65.47**	62.76	63.66	59.46	58.56	**65.47**	**65.47**
Enron	1	**12.95**	8.64	9.67	10.36	3.8	11.23	11.57
Slashdot	3	33.47	33.12	**36.89**	25.94	32.71	36.75	32.64
Ohsumed	4	17.48	18.04	16.36	13.02	**21.12**	16.78	18.12
tmc2007	3	53.82	52.18	**64.45**	47.25	29.31	63.71	53.45
rcv1(subset1)	2.5	**0.13**	0.03	**0.13**	0	**0.13**	**0.13**	0.03
Mediamill	4	6.90	5.35	7.22	5.68	4.89	7.43	**8.42**
Bibtex	1	**14.19**	13.32	13.96	9.62	12.6	OOM	13.96
Avg. rank	2.1	**2.1**	5.0	3.1	6.3	4.8	3.2	3.2

Consider first the base-line BR and LP algorithms. It can be observed that for accuracy, subset accuracy and F-measure the ChiDep algorithm outperforms BR and LP in 25 and 28 (from a total of 36) cases and achieves the same results in 1 and 3 other cases, respectively. For the Hamming loss measure, the ChiDep algorithm is more accurate than LP on all datasets. However it is outperformed by BR in 9 of 12 cases. In general, ChiDep is significantly better than LP in respect to the accuracy, F-measure and the Hamming loss; and is significantly better than BR in respect to the subset accuracy measure.

Considering the recently developed state-of-the-art methods, we notice that ChiDep is significantly better than HOMER, 2BR and PS in respect to the accuracy measure. It is also

Table 10 (*Continued*)

Dataset	ChiDep rank	micro-avg. F-measure						
		ChiDep	BR	LP[-]	HO[-]	2BR	PS[-]	CC
Emotions	1	**60.31**	59.25	52.59	56.75	55.78	52.69	55.63
Scene	3	60.4	60.86	58.75	56.13	58.36	58.59	**61.53**
Yeast	3	56.92	56.9	52.65	57.6	**60.04**	53.98	55.84
Genbase	1	**98.97**	98.77	98.35	98.14	98.77	98.35	98.77
Medical	2	79.36	78.94	74.54	75.78	75.82	75.74	**79.46**
Enron	1	**51.94**	50.42	39.53	50.38	51.39	42.35	51.45
Slashdot	3	48.93	**49.64**	42.26	44.24	47.68	42.98	49.28
Ohsumed	3	48.04	**48.4**	37.52	45.54	46.64	37.56	48.36
tmc2007	2	83.31	**83.42**	78.79	81.2	70.87	78.43	81.85
rcv1(subset1)	2	12.48	12.3	10.09	11.46	11.75	10.72	**12.68**
Mediamill	2	50.44	**50.55**	45.39	48.66	48.82	46.08	49.41
Bibtex	2	39.09	**39.67**	28.97	29.37	36.09	OOM	38.18
Avg. rank	2.1	**2.1**	2.2	6.4	4.8	4.0	5.7	2.6

Dataset	ChiDep rank	Hamming loss						
		ChiDep	BR	LP[-]	HO[-]	2BR	PS[-]	CC
Emotions	2	25.41	25.99	30.2	30.94	**25.25**	30.53	28.96
Scene	4	14.51	13.89	14.72	17.49	**12.63**	14.91	13.92
Yeast	4	26.59	25.88	28.7	28.91	**21.43**	27.18	26.38
Genbase	1	**0.09**	0.11	0.15	0.17	0.11	0.15	0.11
Medical	3	1.17	**1.11**	1.39	1.29	1.23	1.31	**1.11**
Enron	2	5.38	5.4	7.41	6.86	5.59	6.45	**5.3**
Slashdot	3.5	4.36	**4.2**	5.96	5.29	4.36	5.88	4.25
Ohsumed	4	6.59	6.49	8.78	8.98	**5.89**	8.5	6.49
tmc2007	2	3.13	**3.11**	4.2	3.97	5.65	4.21	3.4
rcv1(subset1)	4	4.6	4.56	5.04	5.59	**4.29**	4.82	4.45
Mediamill	2.5	3.82	3.82	4.7	5.03	**3.71**	4.59	4.01
Bibtex	3.5	1.49	1.48	2.08	2.72	**1.34**	OOM	1.49
Avg. rank	3.0	3.0	**2.2**	5.9	6.3	2.4	5.3	2.7

significantly better than HOMER and 2BR in respect to the subset accuracy measure and is significantly better than HOMER and PS for the F-measure and the Hamming loss measures.

The overall comparison of the algorithms indicates that the ChiDep algorithm is successful for accuracy and subset accuracy with four first ranked places at each one of measures; and sharing the first rank with other algorithms for subset accuracy measure in three additional cases. For the Hamming loss measure, ChiDep is outperformed by the BR, 2BR and CC methods. However, the difference is not statistically significant.

Summarizing this comparison, we notice that, as expected, the ChiDep algorithm is mainly beneficial for accuracy, subset accuracy and F-measure providing the best average rank for these measures. Even when ChiDep is outperformed by other methods, its value of

Table 11 Comparing LP superiority vs. BR in terms of subset accuracy measure as function of dependent labels percentage in dataset

Dataset	Subset accuracy		LP vs. BR superiority (diff. in subset accuracy values)	Unconditionally dependent label pairs (%)
	BR	LP		
Scene	40.13	53.68	13.55	100
tmc2007	52.18	64.45	12.27	77
Emotions	12.87	22.28	9.41	93
Yeast	6.43	11.89	5.46	58
Slashdot	33.12	36.89	3.77	22
Mediamill	5.35	7.22	1.87	30
Enron	8.64	9.67	1.03	10
Medical	62.76	63.66	0.9	3
Bibtex	13.32	13.96	0.64	9
rcv1(subset1)	0.03	0.13	0.1	12
Genbase	97.49	96.98	−0.51	12
Ohsumed	18.04	16.36	−1.68	43

a predictive quality measure is still relatively high and is among the three highest in 34 of 36 prediction cases for the above measures.

Also, we notice that the 2BR method demonstrates high performance in terms of the Hamming loss measure. It provided the best (the lowest) Hamming loss in 7 of 12 datasets and in four other datasets it was only slightly outperformed by BR and\or ChiDep. However, on the tmc2007 dataset it is outperformed by all other algorithms. In general, the BR method is the most successful in terms of the Hamming loss measure achieving the best average rank over all datasets, although having the best rank on three datasets only. For the subset accuracy measure, the LP algorithm performs best on 5 of 12 datasets however, its average rank is much worse than that of the ChiDep method. Note that LP achieves results higher than ChiDep algorithm mainly on datasets with relatively large number of examples per distinct label combination (i.e. $|Train|/L_{\mathrm{DC}}$).

7.2.1 LP vs. BR considering percentage of dependent label pairs

Additionally we noted that LP superiority vs. BR in terms of subset accuracy measure is almost proportional to the percent of (unconditionally) dependent label pairs discovered in the dataset. The only exceptions are “genbase” and “ohsumed” datasets, where the BR algorithm slightly outperforms LP. For more convenient comparison, we summarize all the related information in Table 11. The datasets are ordered by difference in subset accuracy measure between LP and BR algorithms (the third column in the table). It can be easily observed that the LP results with subset accuracy values much higher from those of BR for datasets with large percentage of dependent labels. So, we conclude that the superiority of the LP approach compared to BR in respect to subset accuracy is growing with the percentage of dependent labels in dataset.

7.3 Ensemble diversity

In this section we compare the predictive performance of the “base” version of the ChiDep Ensemble (CDE) method according to which m best (i.e. with highest dependence score)

models are selected for participation in the ensemble to the "diverse" version (CDE-d). According to the "diverse" version of the ChiDep ensemble we try to select the most different (at least in k percent) models among the N-high scored ones. The k and N are configured parameters of the algorithm and might be specific for each dataset.

We performed several calibration experiments to examine if the "diverse" version of the CDE algorithm with some default values for the configured parameters can improve the predictive performance of the "base" version. The calibration experiments were run on scene, emotions, yeast and medical training sets using 10-fold cross-validation with parameter k varying from 0.1 to 0.9 with step 0.1 and parameter N varying from 100 to 500 with step 50. In the result analysis, we found that combination of values $k = 0.2$ and $N = 100$ performed well and was the only combination appearing among the 25 best results on all evaluated datasets. The test of CDE-d with the selected parameters on all the datasets showed that indeed the "diverse" version of the CDE algorithm, even with default parameters, improves the predictive performance of the ensemble.

Table 12 presents the results of the CDE-d algorithm with selected default parameters on all datasets and compares them to those of the base CDE version. The results for each algorithm are obtained by averaging five distinct runs on each dataset, with an initialization seed for random number generator varying from 1 to 5 consecutively.

The results show that model diversity, even with default parameters, leads to an improvement of prediction accuracy (in terms of all considered evaluation measures) of the ensemble classifier on almost all datasets. Note that higher predictive performance can be achieved by specifically calibrating the parameters for each dataset.

7.4 ChiDep vs. ensemble algorithms

In this section the results of the "diverse" version of ChiDep Ensemble algorithms are compared to other multi-label ensemble algorithms.

Table 13 presents the results of the compared ensemble classifiers. The results of CDE-d, RAkEL and ECC algorithms are averaged over five distinct runs on each dataset. The averaged number of distinct classification models constructed by the ChiDep Ensemble for each dataset is also presented. The results of the RAkEL algorithm on the emotions and scene datasets are for the 20 models ensemble as it is the maximal number of possible distinct subset combinations of three labels out of total six. On the tmc2007 and rcv1 datasets the EPS algorithm with a configured number of models failed because of insufficient memory for computation (i.e. with Java OutOfMemory Exception). Thus, the result presented is for the 10 models ensemble allowing at least for an estimate of the algorithm's performance. OOM for EPS indicates that the algorithm caused Java OutOfMemoryException also when configured to construct 10 models for the ensemble.

The differences between algorithms were found statistically significant in terms of accuracy and F-measure scores by Friedman test at $p = 0.02$. The followed post-hoc Holm's procedure indicate that there is no significant case where RAkEL, ECC or EPS is more accurate than CDE-d. On the other hand, CDE-d is significantly more accurate than EPS for accuracy and F-measure. In addition, the CDE-d accuracy, subset accuracy and F-measure values are higher than those of ECC and RAkEL algorithms in most cases. However these differences are not statistically significant. In general, the CDE-d method obtains the best average rank at the accuracy, subset accuracy and F-measure scores. From a detailed comparison we can observe that CDE-d algorithm achieves best results on 7, 5 and 6 datasets (from a total of 12) respectively for accuracy, subset accuracy and F-measure scores.

Table 12 Comparing ChiDep Ensemble base and "diverse" versions

Dataset	Accuracy		micro-avg. F-measure	
	CDE	CDE-d	CDE	CDE-d
Emotions	51.05	**53.92**	63.87	**66.84**
Scene	59.91	**60.05**	66.06	**67.29**
Yeast	49.04	**49.7**	62.42	**62.91**
Genbase	98.56	**98.66**	98.6	**98.77**
Medical	71.48	**71.56**	**79.12**	79.02
Enron	43.14	**43.26**	55.21	**55.29**
Slashdot	**38.57**	38.37	**47.55**	47.47
Ohsumed	39.69	**39.74**	**51.22**	51.1
tmc2007	**83.7**	83.55	**88.92**	88.85
rcv1(subset1)	7.2	7.2	**12.81**	12.78
Mediamill	42.54	**43.11**	56.17	**56.63**
Bibtex	30.06	**30.17**	40.03	**40.23**

Dataset	Subset accuracy		Hamming loss	
	CDE	CDE-d	CDE	CDE-d
Emotions	26.24	**28.22**	22.77	**21.62**
Scene	54.26	**54.93**	11.4	**10.76**
Yeast	14.92	**15.01**	21.66	**21.3**
Genbase	97.09	**97.49**	0.13	**0.11**
Medical	63.06	**63.18**	**1.11**	1.12
Enron	12.23	**12.57**	5.03	**5.02**
Slashdot	33.67	**33.77**	4.49	**4.45**
Ohsumed	22.49	**22.79**	5.87	**5.84**
tmc2007	**67.39**	67.12	**2.11**	2.12
rcv1(subset1)	0.09	**0.11**	4.42	4.42
Mediamill	9.52	**10.07**	3.23	**3.18**
Bibtex	13.63	**13.74**	1.45	**1.44**

In respect to Hamming loss, the RAkEL method seems to perform best achieving 4 best results and having the best average rank. Generally, the Hamming loss values for all algorithms are very close in many cases and the differences between them are not significant.

7.5 Algorithms computational time comparison

This section presents results of train and test computational time required by single-classifier and ensemble methods. We present the results for the two largest datasets, where the computational time becomes the important issue and the difference between algorithms is most perceptible. Figure 9 depicts times (in hours) required by single-classifier and ensemble methods algorithms for classifier training; and Fig. 10 presents the times (in minutes) required for classifiers testing.

It can be observed from the graphs that ChiDep train times are relatively long; however its test times are short and comparable to that of BR. On the other hand, both train and test times

Table 13 Comparing ChiDep Ensemble (CDE-d) to other ensemble algorithms on various datasets. [+], [-]-statistically significant improvement or degradation vs. CDE-d

Dataset	CDE-d ModelsNum	CDE-d rank	Accuracy			
			CDE-d	RA	ECC	EPS[-]
Emotions	22	2	53.92	51.28	**54.02**	53.63
Scene	21	3	60.05	60.87	**62.24**	58.5
Yeast	37	1	**49.7**	48.39	45.63	49.14
Genbase	111	1.5	**98.66**	**98.66**	98.63	98.32
Medical	239	3	71.56	71.14	73.82	**75.03**
Enron	249	1	**43.26**	41.89	42.26	34.65
Slashdot	95	2	38.37	**38.54**	36.20	37.22
Ohsumed	86	1	**39.74**	37.90	39.36	32.56
tmc2007	77	1	**83.55**	80.90	72.19	75.69
rcv1(subset1)	473	1	**7.2**	7.04	6.662	6.94
Mediamill	459	2	43.11	**44.52**	40.89	OOM
Bibtex	761	1	**30.17**	29.88	29.37	OOM
Avg. rank		1.6	**1.6**	2.4	2.8	3.1

Dataset	CDE-d ModelsNum	CDE-d rank	Subset accuracy			
			CDE-d	RA	ECC	EPS
Emotions	22	2	28.22	24.75	24.95	**29.21**
Scene	21	2	54.93	54.60	51.97	**55.77**
Yeast	37	2	15.01	12.52	14.42	**16.9**
Genbase	111	1.5	**97.49**	**97.49**	97.39	96.48
Medical	239	3	63.18	62.70	65.29	**67.57**
Enron	249	2	12.57	11.54	**13.47**	11.57
Slashdot	95	1	**33.77**	32.98	31.52	33.26
Ohsumed	86	1	**22.79**	21.58	19.48	21.46
tmc2007	77	1	**67.12**	62.44	48.28	58.08
rcv1(subset1)	473	1.5	**0.11**	**0.11**	0.01	0.07
Mediamill	459	3	10.07	**11.38**	10.98	OOM
Bibtex	761	2.5	13.74	13.74	**15.79**	OOM
Avg. rank		1.9	**1.9**	2.7	2.9	2.2

of the ChiDep Ensemble method are comparable to those of other ensemble methods. The CDE train times are shorter than RAkEL's train times and are slightly longer than those of ECC. The test times of all ensemble methods are almost equal. Recall that the EPS algorithm on these datasets resulted with OutOfMemory Exception, thus its computational times are not presented.

7.6 Discussion

Comparing the two methods for label dependence identification indicates that estimating conditional dependencies is much more computationally expensive than estimating uncon-

Table 13 (*Continued*)

Dataset	CDE-d ModelsNum	CDE-d rank	micro-avg. F-measure			
			CDE-d	RA	ECC	EPS[-]
Emotions	22	2	66.84	64.93	66.17	**67.21**
Scene	21	3	67.29	**68.62**	66.57	67.61
Yeast	37	1	**62.91**	62.08	58.78	62.41
Genbase	111	1.5	**98.77**	**98.77**	98.73	98.34
Medical	239	2	79.02	78.91	**79.7**	77.79
Enron	249	1	**55.29**	54.87	53.48	45.29
Slashdot	95	3	47.47	**49.63**	47.53	45.92
Ohsumed	86	2	51.1	50.04	**52.08**	41.99
tmc2007	77	1	**88.85**	87.37	80.35	81.3
rcv1(subset1)	473	1.5	**12.78**	12.45	**12.78**	12.42
Mediamill	459	2	56.63	**57.21**	53.82	OOM
Bibtex	761	1	**40.23**	40.03	38.61	OOM
Avg. rank		1.8	**1.8**	2.2	2.7	3.2

Dataset	CDE-d ModelsNum	CDE-d rank	Hamming loss			
			CDE-d	RA	ECC	EPS
Emotions	22	2	21.62	22.19	23.48	**20.05**
Scene	21	3	10.76	10.45	12.53	**9.87**
Yeast	37	2	21.3	21.81	23.60	**19.71**
Genbase	111	2	**0.11**	**0.11**	**0.11**	0.15
Medical	239	2.5	1.12	1.12	**1.1**	1.18
Enron	249	2.5	5.02	**4.90**	5.05	5.02
Slashdot	95	4	4.45	**4.19**	4.21	4.43
Ohsumed	86	1	**5.84**	5.91	6.33	5.85
tmc2007	77	1	**2.12**	2.37	3.85	3.56
rcv1(subset1)	473	4	4.42	4.34	**4.04**	4.05
Mediamill	459	2	3.18	**3.03**	3.2	OOM
Bibtex	761	3	1.44	1.41	**1.32**	OOM
Avg. rank		2.4	2.4	**2.1**	2.8	2.4

ditional dependencies. The former method is also inferior from the predictive performance point of view on regular size datasets.

Analysis of identified dependent label pairs by both methods on benchmark datasets showed that many more label pairs were identified as unconditionally dependent, than those identified as conditionally dependent.

The results of the empirical experiments evaluating the ChiDep algorithm support our conjectures from Sect. 5.3 that modeling partial dependencies by combining the BR and LP approaches (as the LPBR method does) can be beneficial, compared to each algorithm separately, in terms of (1) accuracy measure and (2) subset accuracy measure for small and medium training sets.

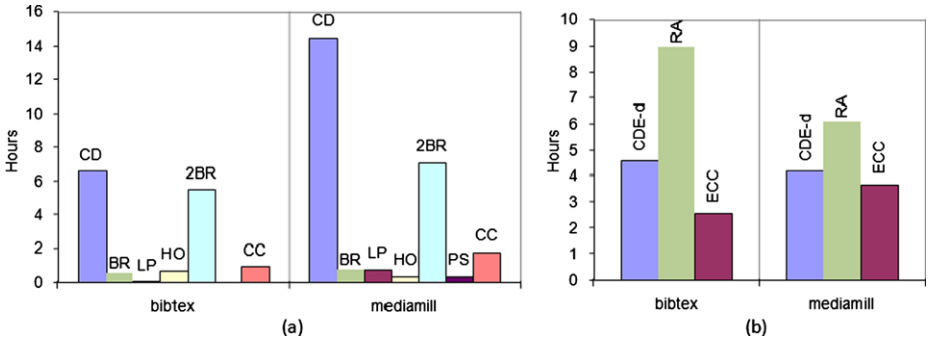


Fig. 9 Train times (in hours) required by (**a**) single-classifier methods and (**b**) ensemble methods

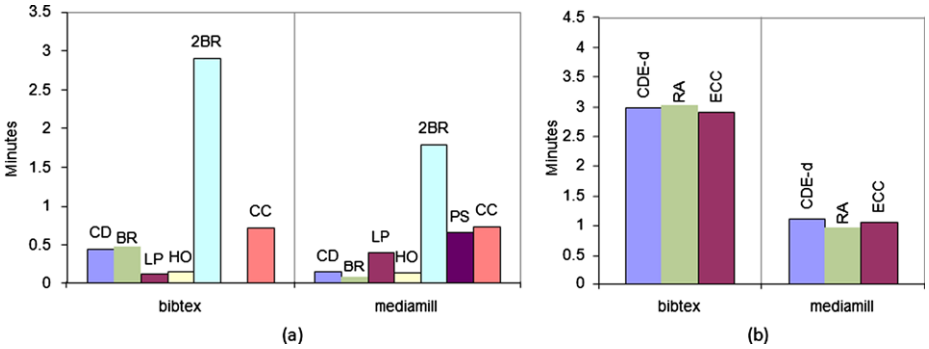


Fig. 10 Test times (in minutes) required by (**a**) single-classifier methods and (**b**) ensemble methods

Indeed, the empirical results demonstrate that ChiDep algorithm significantly outperforms BR in terms of the subset accuracy measure and LP in terms of the accuracy, micro-averaged F-measure and Hamming loss measures. In addition, for the subset accuracy measure, ChiDep outperforms LP on datasets with a limited number of training examples or with a small average number of examples per distinct label combination (i.e. $|Train|/L_{\mathrm{DC}}$ is comparatively small). As for F-measure, ChiDep outperforms LP on all datasets and as well outperforms BR on most of the regular size datasets. However on datasets having sufficient number of training examples BR performs with highest F-measure values.

Moreover, we found that the ChiDep algorithm has the highest average rank among all the compared algorithms in terms of accuracy, subset accuracy and micro-averaged F-measure scores. As well, the ChiDep Ensemble has the highest average rank among all the compared ensemble algorithms in terms of the same measures.

Summarizing the above we conclude that ChiDep and ChiDep Ensemble methods are especially beneficial when accuracy is the target measure of a classification problem. ChiDep is also beneficial in respect to subset accuracy and F-measure on datasets with a limited number of training examples.

Another finding reveals that the BR and 2BR methods demonstrate the highest performance in terms of Hamming loss measure among the single-classifier algorithms, whereas ensemble models make it possible to reduce the Hamming loss even further. It was also found that in terms of subset accuracy measure, the superiority from utilizing LP over BR is growing directly with the level of label interdependence in dataset.

A general conclusion of this wide range experiment is that different algorithms could be beneficial for various evaluation measures depending on the dataset properties. We believe that guidelines for practitioner for selection the algorithm for a specific problem might be indeed very useful. We can already draw some preliminary rules from the results of the current experiment:

- "*Among the baseline methods, use LP algorithm if you are interested in highest Subset accuracy on dataset with many interdependent labels and enough training examples for the existing label combinations*."
- "*If Accuracy or F-measure is the target of a classification problem, BR algorithm is able to provide highest results (among the single-classifier methods) on most datasets with large number of training examples*."
- "*For highest Accuracy values on datasets with few training examples choose the ChiDep algorithm (among the single-classifier methods).*"
- "*For best Hamming loss measure values use BR or 2BR method (among the single-classifier methods). To further reduce the values use one of the ensemble methods.*"

However, these rules yet should be refined and confirmed by specifically designed experiments. Much more investigation is needed to reach a comprehensive set of valuable rules. This is one of our future research directions.

8 Conclusions

In this paper we have presented a novel algorithm for label set partitioning into several subsets of dependent labels and for applying a combination of LP and BR methods on these subsets. The basic idea is to decompose the original set of labels into several subsets of dependent labels, build an LP classifier for each subset, and then combine them as in the BR method.

To evaluate the new algorithm we first compared methods for identifying conditional vs. unconditional label dependencies on various benchmark and artificial datasets. The results confirm that modeling unconditional dependencies is good enough for solving multi-label classification problems and modeling conditional dependencies does not improve the predictive performance of the classifier. Then, we evaluated and compared the new algorithm and its ensemble version to nine other multi-label algorithms with four different measures, utilizing 12 datasets of various complexities to give as wide as possible a picture of the algorithm's effectiveness.

Summarizing the results of this evaluation experiment, we conclude that the multi-label classification method presented in this paper is able to improve prediction accuracy and in some cases also subset accuracy compared to other known multi-label classification algorithms.

In addition, we presented generalization bounds for a flexible family of multi-label penalty functions. Our analysis yields a theoretical understanding of the reduction in sample complexity that is gained from unconditional label independence. The present analysis is a worst-case model, and we intend to examine average-case models in future work.

Among the additional issues to be further studied are: development of more efficient labels clustering procedure for improvement of the ChiDep time performance; exploration of better methods for identification of conditional label dependence; and development of rules and tools that could help in selection the optimal algorithm for multi-label classification according to specific dataset and classification problem properties.

References

Baum, E. B., & Haussler, D. (1989). What size net gives valid generalization? *Neural Computation*, *1*(1), 151–160.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. K. (1989). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, *36*(4), 929–965.

Dembczynski, K., Cheng, W., & Hullermeier, E. (2010a). Bayes optimal multilabel classification via probabilistic classifier chains. In *Proc. ICML 2010*, Haifa, Israel.

Dembczynski, K., Waegeman, W., Cheng, W., & Hüllermeier, E. (2010b). On label dependence in multi-label classification. Working notes of the 2nd international workshop on learning from multi-label data, Haifa, Israel.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Eisenstat, D., & Angluin, D. (2007). The VC dimension of k-fold union. *Information Processing Letters*, *101*(5), 181–184.

Eisenstat, D. (2009). k-fold unions of low-dimensional concept classes. *Information Processing Letters*, *109*(23–24), 1232–1234.

Ghamrawi, N., & McCallum, A. (2005). Collective multi-label classification. In *CIKM 2005* (pp. 195–200).

Kearns, M. J., Schapire, R. E., & Sellie, L. (1994). Toward efficient agnostic learning. *Machine Learning*, *17*(2–3), 115–141.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, *97*(1–2), 273–324.

Pollard, D. (1984). *Convergence of stochastic processes*. New York: Springer.

Read, J., Pfahringer, B., & Holmes, G. (2008). Multi-label classification using ensembles of pruned sets. In *Proceedings of eighth IEEE international conference on data mining* (pp. 995–1000).

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2009). Classifier chains for multi-label classification. In *Proceedings of 20th European conference on machine learning and knowledge discovery in databases* (Vol. 2, pp. 254–269).

Rokach, L. (2008). Genetic algorithm-based feature set partitioning for classification problems. *Pattern Recognition*, *41*(5), 1693–1717. doi:10.1016/j.patcog.2007.10.013.

Rokach, L. (2010). *Pattern classification using ensemble methods*. *Series in machine perception and artificial intelligence: Vol. 75*. Singapore: World Scientific.

Rokach, L., & Maimon, O. (2005). Feature set decomposition for decision trees. *Journal of Intelligent Data Analysis*, *9*(2), 131–158.

Schapire, R. E., & Singer, Y. (2000). Boostexter: a boosting-based system for text categorization. *Machine Learning*, *39*(2–3), 135–168.

Tenenboim, L., Rokach, L., & Shapira, B. (2009). Multi-label classification by analyzing labels dependencies. In G. Tsoumakas, M. L. Zhang, & Z. H. Zhou (Eds.), *Proceedings of the 1st international workshop on learning from multi-label data*, Bled, Slovenia (pp. 117–132).

Tsoumakas, G., & Vlahavas, I. (2007). Random k-labelsets: an ensemble method for multilabel classification. In *Proceedings of 18th European conference on machine learning*, Warsaw, Poland (pp. 406–417).

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2008). Effective and efficient multilabel classification in domains with large number of labels. In *Proceedings of ECML/PKDD 2008 workshop on mining multidimensional data* (pp. 30–44).

Tsoumakas, G., Dimou, A., Spyromitros, E., Mezaris, V., Kompatsiaris, I., & Vlahavas, I. (2009). Correlation-based pruning of stacked binary relevance models for multi-label learning. In G. Tsoumakas, M. L. Zhang, & Z. H. Zhou (Eds.), *Proceedings of the 1st international workshop on learning from multi-label data*, Bled, Slovenia (pp. 101–116).

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2010). Mining multi-label data. In O. Maimon & L. Rokach (Eds.), *Data mining and knowledge discovery handbook* (2nd ed., pp. 667–686). New York: Springer.

Vapnik, V. N., & Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, *16*, 264–279.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. New York: Springer.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*, 241–259.

Xu, J. (2010). Constructing a fast algorithm for multi-label classification with support vector data description. In *IEEE international conference on granular computing* (pp. 817–821).

Zhang, M. L., Peña, J. M., & Robles, V. (2009). Feature selection for multi-label naive Bayes classification. *Information Sciences*, *179*(19), 3218–3229.

Zhang, M., & Zhang, K. (2010). Multi-label learning by exploiting label dependency. In *Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining*, Washington, DC, USA (pp. 999–1008). http://doi.acm.org/10.1145/1835804.1835930.