Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2011, Article ID 645964, 14 pages
doi:10.1155/2011/645964

Research Article

A Dependent Multilabel Classification Method Derived from the k-Nearest Neighbor Rule

Zoulficar Younes (EURASIP Member),[1] Fahed Abdallah,[1] Thierry Denoeux,[1] and Hichem Snoussi[2]

[1] *Heudiasyc, UMR CNRS 6599, University of Technology of Compiègne, 60205 Compiègne, France*
[2] *ICD-LM2S, FRE CNRS 2848, University of Technology of Troyes, 10010 Troyes, France*

Correspondence should be addressed to Zoulficar Younes, zoulficar.younes@hds.utc.fr

Received 17 June 2010; Revised 9 January 2011; Accepted 21 February 2011

Academic Editor: Bülent Sankur

In multilabel classification, each instance in the training set is associated with a set of labels, and the task is to output a label set whose size is unknown *a priori* for each unseen instance. The most commonly used approach for multilabel classification is where a binary classifier is learned independently for each possible class. However, multilabeled data generally exhibit relationships between labels, and this approach fails to take such relationships into account. In this paper, we describe an original method for multilabel classification problems derived from a Bayesian version of the k-nearest neighbor (k-NN) rule. The method developed here is an improvement on an existing method for multilabel classification, namely multilabel k-NN, which takes into account the dependencies between labels. Experiments on simulated and benchmark datasets show the usefulness and the efficiency of the proposed approach as compared to other existing methods.

1. Introduction

Traditional *single-label classification* assigns an object to exactly one class, from a set of Q disjoint classes. *Multilabel classification* is the task of assigning an instance simultaneously to one or multiple classes. In other words, the target classes are not exclusive: an object may belong to an unrestricted set of classes, rather than to exactly one class. For multilabeled data, an instance may belong to more than one class not because of ambiguity (*fuzzy* membership), but because of multiplicity (*full* membership) [1]. Note that traditional supervised learning problems (binary or multi-class) can be regarded as special cases of the problem of multilabel learning, where instances are restricted to belonging to a single class.

Recently, multilabel classification methods have been increasingly required by modern applications where it is quite natural for some instances to belong to several classes at once. Typical examples of multilabel problems are text categorization, functional genomics, and scene classification. In text categorization, each document may belong to multiple topics, such as *arts* and *humanities* [2–5]; in gene functional analysis, each gene may be associated with a set of functional classes, such as *energy*, *metabolism,* and *cellular biogenesis* [6]; in natural scene classification, each image may belong to several image types at the same time, such as *sea* and *sunset* [1].

A common approach to a multilabel learning problem is to transform it into one or more single-label problems. The best known transformation method is the *binary relevance* (BR) approach [7]. This approach transforms a multilabel classification problem with Q possible classes into Q single-label classification problems. The qth single-label classification problem ($q \in \{1, \ldots, Q\}$) consists in separating the instances belonging to class ω_q from the others. This problem is solved by training a binary classifier (0/1 decision) where each instance in the training set is considered to be *positive* if it belongs to ω_q, and *negative* otherwise. The output of the multilabel classifier is determined by combining the decisions provided by the different binary classifiers. The BR approach tacitly assumes that labels can be assigned

independently: when one label provides information about another, the binary classifier fails to capture this effect. For example, if a news article belongs to a "music" category, it is very likely that it also belongs to an "entertainment" category. Although the BR approach is generally criticized for its assumption of label independencies [8, 9], it is a simple, intuitive approach that has the advantage of having low computational complexity.

In [10], the authors present a Bayesian multilabel k-nearest neighbor (MLkNN) approach where, in order to assign a set of labels to a new instance, a decision is made separately for each label by taking into account the number of neighbors containing the label to be assigned. This method therefore fails to take into account the dependency between labels.

In this paper, we present a generalization of the MLkNN-based approach to multilabel classification problems where the dependencies between classes are considered. We call this method DMLkNN, for dependent multilabel k-nearest Neighbor. The principle of the method is as follows. For each unseen instance, we identify its k-NNs in the training set. According to the class membership of neighboring instances, a *global* maximum *a posteriori* (MAP) principle is used in order to assign a set of labels to the new unseen instance. Note that unlike MLkNN, in order to decide whether a particular label should be included among the unseen instance's labels, the *global* MAP rule takes into account the numbers of different labels in the neighborhood, instead of considering only the number of neighbors having the label in question.

Note that this paper is an extension of a previously published conference paper [11]. Here, the method is more thoroughly interpreted and discussed. Extensive comparisons on several real world datasets and with some state-of-the-art methods are added in the experimental section. In addition, we provide an illustrative example on a simulated dataset, where we explain step by step the principle of our algorithm.

The remainder of the paper is organized as follows. Section 2 presents related work. Section 3 describes the principle of multilabel classification and the notion of label dependencies. Section 4 introduces the DMLkNN method and its implementation. Section 5 presents some experiments and discusses the results. Finally, Section 6 summarizes this work and makes concluding remarks.

2. Related Work

Several methods have been proposed in the literature for multilabel learning. These methods can be categorized into two groups. A first group contains the *indirect* methods that transform a multilabel classification problem into binary classification problems (a binary classifier for each class or pairwise classifiers) [1, 9] or into multi-class classification problem (each subset of classes is considered as a new class) [7]. A second group consists in extending common learning algorithms and making them able to manipulate multilabel data *directly* [12]. Some multilabel classification methods are briefly described below.

In [13], an adaptation of the traditional radial basis function (RBF) neural network for multilabel learning is presented. It consists of two layers of neurons: a first layer of hidden neurons representing basis functions associated with prototype vectors, and a second layer of output neurons related to all possible classes. The proposed method, named MLRBF, first performs a clustering of the instances corresponding to each possible class; the prototype vectors of the first-layer basis functions are then set to the centroids of the clustered groups. In a second step, the weights of the second-layer are fixed by minimizing a sum-of-squares error function. The output neuron of each class is connected to all input neurons corresponding to the prototype vectors of the different possible classes. Therefore, information encoded in the prototype vectors of all classes is fully exploited when optimizing the connection weights and predicting the label sets for unseen instances.

In [6], a multilabel ranking approach based on support vector machines (SVM) is presented. The authors define a cost function and a special multilabel margin and then propose an algorithm named RankSVM based on a ranking system combined with a label set size predictor. The set size predictor is computed from a threshold value that separates the relevant from the irrelevant labels. The value is chosen by solving a learning problem. The goal is to minimize a ranking loss function while having a large margin. RankSVM uses kernels rather than linear dot products, and the optimization problem is solved via its dual transformation.

In [12], an evidence-theoretic k-NN rule for multilabel classification is presented. This rule is based on an evidential formalism for representing uncertainties in the classification of multilabeled data and handling imprecise labels, described in detail in [14]. The formalism extends all the notions of Dempster-Shafer theory [15] to the multilabel case, with only a moderate increase in complexity as compared to the classical case. Under this formalism, each piece of evidence about an instance to be classified is represented by a pair of sets: a set of classes that surely apply to the unseen instance, and a set of classes that surely do not apply to this instance.

A distinction should be made between multilabel and *multiple-label* learning problems. Multiple-label learning [16] is a semisupervised learning problem for single-label classification where each instance is associated with a set of labels, but where only one of the candidate labels is the true label for the given instance. For example, this situation occurs when the training data is labeled by several experts and, owing to conflicts and disagreements between the experts, a set of labels, rather than exactly one label, is assigned to some instances. The set of labels of an instance contains the decision (the assigned label) made by each expert about this instance. This means that there is an ambiguity in the class labels of the training instances.

Another learning problem is *multi-instance multilabel* learning, where each object is described by a bag of instances and is assigned a set of labels [17]. Different real-world applications can be handled under this framework. For example, in text categorization, each document can be represented by a bag of instances, with each instance representing a section of

the document in question, while the document may deal with several topics at the same time, such as *culture* and *society*.

In [18], *dynamic conditional random fields* (DCRFs) are presented for representing and handling complex interaction between labels in sequence modeling, such as when performing multiple, cascaded labeling tasks on the same sequence. DCRFs are a generalization of conditional random fields. Inference in DCRFs can be done using approximate methods, and training can be done by maximum *a posteriori* estimation.

3. Multilabel Classification

3.1. Principle. Let $\mathbb{X} = \mathbb{R}^d$ denote the domain of instances and let $\mathcal{Y} = \{\omega_1, \omega_2, \ldots, \omega_Q\}$ be the finite set of labels. The multilabel classification problem can be formulated as follows. Given a set $\mathcal{D} = \{(\mathbf{x}_1, Y_1), (\mathbf{x}_2, Y_2), \ldots, (\mathbf{x}_n, Y_n)\}$ of n training examples, independently drawn from $\mathbb{X} \times 2^{\mathcal{Y}}$, and identically distributed, where $\mathbf{x}_i \in \mathbb{X}$ and $Y_i \subseteq \mathcal{Y}$, the goal of the learning system is to build a multilabel classifier $\mathcal{H} : \mathbb{X} \to 2^{\mathcal{Y}}$ in order to assign a label set to each unseen instance. As for standard classification problems, we can associate with the multilabel classifier $\mathcal{H}$ a scoring function $f : \mathbb{X} \times \mathcal{Y} \to \mathbb{R}$, which assigns a real number to each instance/label combination $(\mathbf{x}, \omega) \in \mathbb{X} \times \mathcal{Y}$. The score $f(\mathbf{x}, \omega)$ corresponds to the probability that instance $\mathbf{x}$ belongs to class ω. Given any instance $\mathbf{x}$ with its known set of labels $Y \subseteq \mathcal{Y}$, the scoring function f is assumed to give larger scores for labels in Y than it does for those not in Y. In other words, $f(\mathbf{x}, \omega_q) > f(\mathbf{x}, \omega_r)$ for any $\omega_q \in Y$ and $\omega_r \notin Y$. The scoring function f allows us to rank the different labels according to their scores. For an instance $\mathbf{x}$, the higher the rank of a label ω, the larger the value of the corresponding score $f(\mathbf{x}, \omega)$. Note that the multilabel classifier $\mathcal{H}(\cdot)$ can be derived from the function $f(\cdot, \cdot)$ via thresholding:

$$\mathcal{H}(\mathbf{x}) = \{\omega \in \mathcal{Y} \mid f(\mathbf{x}, \omega) \geq t\}, \tag{1}$$

where t is a threshold value.

3.2. Label Dependencies in Multilabel Applications. In multilabel classification, the assignment of class ω to an instance $\mathbf{x}$ may provide information about that instance's membership of other classes. Label dependencies exist when the probability of an instance belonging to a class depends on its membership of other classes. For example, a document with the topic *politics* is unlikely to be labeled as *entertainment*, but the probability that the document belongs to the class *economic* is high.

In general, relationships between labels are high order or even full order, that is, there is a relation between a label and all remaining labels, but these relations are more difficult to represent than second-order relations, that is, relations that exist between pairs of labels. The dependencies between labels can be represented in the form of a contingency matrix mat that allows us to express only *second-order* relations between labels. Let H_1^q denote the hypothesis that instance $\mathbf{x}$ belongs to class $\omega_q \in \mathcal{Y}$. Given a multilabeled dataset $\mathcal{D}$ with Q possible labels, $\mathrm{mat}[q]\,[r] = \Pr(\mathrm{H}_1^q \mid \mathrm{H}_1^r)$, where

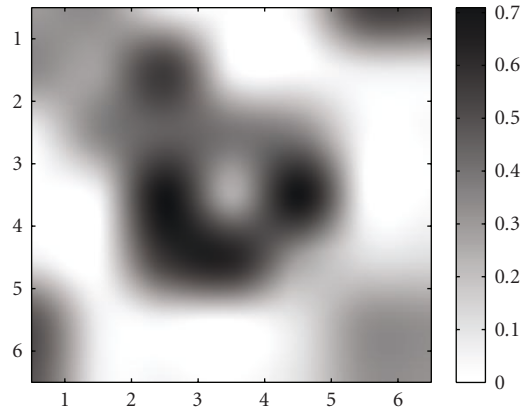


Figure 1: Contingency matrix for the emotion dataset.

q and $r \in \{1, \ldots, Q\}$ with $q \neq r$, indicates the second-order relationship between labels ω_q and ω_r. $\Pr(\mathrm{H}_1^q \mid \mathrm{H}_1^r)$ represents the proportion of data in $\mathcal{D}$ belonging to ω_r to which label ω_q is also assigned. $\mathrm{mat}[q]\,[q] = \Pr(\mathrm{H}_1^q)$ indicates the frequency of label ω_q in the dataset $\mathcal{D}$. Figure 1 shows the contingency matrix for the emotion dataset ($Q = 6$) used in the experiments in Section 5. In this dataset, each instance represents a song and is labeled by the emotions evoked by this song. We can see in Figure 1 that $\mathrm{mat}[1]\,[4] = \Pr(\mathrm{H}_1^1 \mid \mathrm{H}_1^4) = 0$, meaning that labels ω_1 and ω_4 cannot occur together. This is easily interpretable, as ω_1 corresponds to "amazed-surprised" while ω_4 corresponds to "quiet-still", and these two emotions are clearly opposite. We can also see that $\mathrm{mat}[5]\,[4] = \Pr(\mathrm{H}_1^5 \mid \mathrm{H}_1^4) = 0.6$, which means that ω_5, representing "sad-lonely", frequently coexists in the label sets with ω_4. We can deduce from these examples that labels in multilabeled datasets are often mutually dependent, and exploiting relationships between labels will be very helpful in improving classification performance.

4. The DMLkNN Method for Multilabel Classification

We use the same notation as in [10] in order to facilitate comparison with the MLkNN method. Given an instance $\mathbf{x}$ and its associated label set $Y \subseteq \mathcal{Y}$, let $\mathcal{N}_{\mathbf{x}}^k$ denote the set of the k closest training examples of $\mathbf{x}$ in the training dataset $\mathcal{D}$ according to a distance function $\mathrm{d}(\cdot, \cdot)$, and let $\mathbf{y}_{\mathbf{x}}$ be the Q-dimensional *category* vector of $\mathbf{x}$ whose qth component indicates whether $\mathbf{x}$ belongs to class ω_q:

$$\mathbf{y}_{\mathbf{x}}(q) = \begin{cases} 1, & \text{if } \omega_q \in Y, \\ 0, & \text{otherwise,} \end{cases} \quad \forall q \in \{1, \ldots, Q\}. \tag{2}$$

Let us represent by $\mathbf{c}_{\mathbf{x}}$ the Q-dimensional *membership counting* vector of $\mathbf{x}$, the qth component of which indicates how many examples amongst the k-NNs of $\mathbf{x}$ belong to class ω_q:

$$\mathbf{c}_{\mathbf{x}}(q) = \sum_{\mathbf{x}_i \in \mathcal{N}_{\mathbf{x}}^k} \mathbf{y}_{\mathbf{x}_i}(q), \quad \forall q \in \{1, \ldots, Q\}. \tag{3}$$

4.1. MAP Principle. Let $\mathbf{x}$ now denote an instance to be classified. Like in all k-NN based methods, when classifying a test instance $\mathbf{x}$, the set $\mathcal{N}_{\mathbf{x}}^{k}$ of its k nearest neighbors should first be identified. Under the multilabel assumption, the counting vector $\mathbf{c}_{\mathbf{x}}$ is computed. As mentioned before, let H_1^q denote the hypothesis that $\mathbf{x}$ belongs to class ω_q, and H_0^q the hypothesis that $\mathbf{x}$ should not be assigned to ω_q. Let E_j^q ($j \in \{0,1,\ldots,k\}$) denote the event that there are exactly j instances in $\mathcal{N}_{\mathbf{x}}^{k}$ belonging to class ω_q. To determine the qth component of the category vector $\mathbf{y}_{\mathbf{x}}$ for instance $\mathbf{x}$, the MLkNN algorithm uses the following MAP [10]:

$$\hat{\mathbf{y}}'_{\mathbf{x}}(q) = \arg\max_{b\in\{0,1\}} \Pr\left(\mathrm{H}_b^q \mid \mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}\right), \tag{4}$$

while for the DMLkNN algorithm, the following MAP is used:

$$\begin{aligned}\hat{\mathbf{y}}_{\mathbf{x}}(q) &= \arg\max_{b\in\{0,1\}} \Pr\left(\mathrm{H}_b^q \mid \bigwedge_{\omega_l\in\mathcal{Y}} \mathrm{E}^l_{\mathbf{c}_{\mathbf{x}}(l)}\right)\\ &= \arg\max_{b\in\{0,1\}} \Pr\left(\mathrm{H}_b^q \mid \mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l\in\mathcal{Y}\setminus\{\omega_q\}} \mathrm{E}^l_{\mathbf{c}_{\mathbf{x}}(l)}\right).\end{aligned} \tag{5}$$

In contrast to decision rule (4), we can see from (5) that the assignment of label ω_q to the test instance $\mathbf{x}$ depends not only on the event that there are exactly $\mathbf{c}_{\mathbf{x}}(q)$ instances having label ω_q in $\mathcal{N}_{\mathbf{x}}^{k}$, that is, $\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}$, but also on $\bigwedge_{\omega_l\in\mathcal{Y}\setminus\{\omega_q\}}\mathrm{E}^l_{\mathbf{c}_{\mathbf{x}}(l)}$, which is the event that there are exactly $\mathbf{c}_{\mathbf{x}}(l)$ instances having label ω_l in $\mathcal{N}_{\mathbf{x}}^{k}$, for each $\omega_l \in \mathcal{Y}\setminus\{\omega_q\}$. Thus, it is clear that label correlation is taken into account in (5), since all the components of the counting vector $\mathbf{c}_{\mathbf{x}}$ are involved in the assignment or not of label ω_q to $\mathbf{x}$, which is not the case in (4).

4.2. Posterior Probability Estimation. Regarding the counting vector $\mathbf{c}_{\mathbf{x}}$, the number of possible events $\bigwedge_{\omega_l\in\mathcal{Y}}\mathrm{E}^l_{\mathbf{c}_{\mathbf{x}}(l)}$ is upper bounded by $(k+1)^{\mathrm{Q}}$. This means that, in addition to the complexity problem, the estimation of (5) from a relatively small training set will not be accurate. To overcome this difficulty, we will adopt a fuzzy approximation for (5). This approximation is based on the event F_j^l, $j \in \{0,1,\ldots,k\}$, which is the event that there are *approximately* j instances in $\mathcal{N}_{\mathbf{x}}^{k}$ belonging to class ω_l, that is, F_j^l, denotes the event that the number of instances in $\mathcal{N}_{\mathbf{x}}^{k}$ that are assigned label ω_l is in the interval $[j-\delta; j+\delta]$, where $\delta \in \{0,\ldots,k\}$ is a *fuzziness* parameter. As a consequence, we can derive a fuzzy MAP rule

$$\hat{\mathbf{y}}_{\mathbf{x}}(q) = \arg\max_{b\in\{0,1\}} \Pr\left(\mathrm{H}_b^q \mid \bigwedge_{\omega_l\in\mathcal{Y}} \mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)}\right). \tag{6}$$

To remain closer to the initial formulation and for comparison with MLkNN, (6) will be replaced by the following rule:

$$\hat{\mathbf{y}}_{\mathbf{x}}(q) = \arg\max_{b\in\{0,1\}} \Pr\left(\mathrm{H}_b^q \mid \mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l\in\mathcal{Y}\setminus\{\omega_q\}} \mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)}\right). \tag{7}$$

For large values of δ, the results of our method will be similar to those of MLkNN. In fact, for $\delta = k$, the MLkNN algorithm is a particular case of the DMLkNN algorithm, where $\bigwedge_{\omega_l\in\mathcal{Y}\setminus\{\omega_q\}}\mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)}$ will be *certain*, because for each $\omega_l \in \mathcal{Y}\setminus\{\omega_q\}$, the number of instances in $\mathcal{N}_{\mathbf{x}}^{k}$ belonging to class ω_l will surely be in the interval $[j-k; j+k]$. For small values of δ, the assignment or not of label ω_q to test instance $\mathbf{x}$ will not only depend on the number of instances in $\mathcal{N}_{\mathbf{x}}^{k}$ that belong to label ω_q, but also on the number of instances in $\mathcal{N}_{\mathbf{x}}^{k}$ belonging to the remaining labels.

Using Bayes' rule, (4) and (7) can be written as follows:

$$\begin{aligned}\hat{\mathbf{y}}'_{\mathbf{x}}(q) &= \arg\max_{b\in\{0,1\}} \frac{\Pr\left(\mathrm{H}_b^q\right)\Pr\left(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)} \mid \mathrm{H}_b^q\right)}{\Pr\left(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}\right)}\\ &= \arg\max_{b\in\{0,1\}} \Pr\left(\mathrm{H}_b^q\right)\Pr\left(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)} \mid \mathrm{H}_b^q\right).\end{aligned} \tag{8}$$

$$\begin{aligned}\hat{\mathbf{y}}_{\mathbf{x}}(q) &= \arg\max_{b\in\{0,1\}} \frac{\Pr\left(\mathrm{H}_b^q\right)\Pr\left(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l\in\mathcal{Y}\setminus\{\omega_q\}}\mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)} \mid \mathrm{H}_b^q\right)}{\Pr\left(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l\in\mathcal{Y}\setminus\{\omega_q\}}\mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)}\right)}\\ &= \arg\max_{b\in\{0,1\}} \Pr\left(\mathrm{H}_b^q\right)\Pr\left(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l\in\mathcal{Y}\setminus\{\omega_q\}} \mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)} \mid \mathrm{H}_b^q\right).\end{aligned} \tag{9}$$

To rank labels in $\mathcal{Y}$, a Q-dimensional real-valued vector $\mathbf{r}_{\mathbf{x}}$ can be calculated. The qth component of $\mathbf{r}_{\mathbf{x}}$ is defined as the posterior probability $\Pr(\mathrm{H}_1^q \mid \mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l\in\mathcal{Y}\setminus\{\omega_q\}}\mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)})$

$$\begin{aligned}\mathbf{r}_{\mathbf{x}}(q) &= \Pr\left(\mathrm{H}_1^q \mid \mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l\in\mathcal{Y}\setminus\{\omega_q\}} \mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)}\right)\\ &= \frac{\Pr\left(\mathrm{H}_1^q\right)\Pr\left(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l\in\mathcal{Y}\setminus\{\omega_q\}}\mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)} \mid \mathrm{H}_1^q\right)}{\Pr\left(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l\in\mathcal{Y}\setminus\{\omega_q\}}\mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)}\right)}\\ &= \frac{\Pr\left(\mathrm{H}_1^q\right)\Pr\left(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l\in\mathcal{Y}\setminus\{\omega_q\}}\mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)} \mid \mathrm{H}_1^q\right)}{\sum_{b\in\{0,1\}}\Pr\left(\mathrm{H}_b^q\right)\Pr\left(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l\in\mathcal{Y}\setminus\{\omega_q\}}\mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)} \mid \mathrm{H}_b^q\right)}.\end{aligned} \tag{10}$$

For comparison, the real-valued vector $\mathbf{r}'_{\mathbf{x}}$ for MLkNN has the following expression:

$$\begin{aligned}\mathbf{r}'_{\mathbf{x}}(q) &= \Pr\left(\mathrm{H}_1^q \mid \mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}\right)\\ &= \frac{\Pr\left(\mathrm{H}_1^q\right)\Pr\left(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)} \mid \mathrm{H}_1^q\right)}{\Pr\left(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}\right)}\\ &= \frac{\Pr\left(\mathrm{H}_1^q\right)\Pr\left(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)} \mid \mathrm{H}_1^q\right)}{\sum_{b\in\{0,1\}}\Pr\left(\mathrm{H}_b^q\right)\Pr\left(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)} \mid \mathrm{H}_b^q\right)}.\end{aligned} \tag{11}$$

$[\mathbf{y}_{\mathbf{x}}, \mathbf{r}_{\mathbf{x}}] = \text{DML}k\text{NN}(\mathcal{D}, \mathbf{x}, k, s, \delta)$

%Computing the prior probabilities and the number of instances belonging to each class

(1) **For** $q = 1, \ldots, Q$

(2) $\Pr(\mathrm{H}_1^q) = (\sum_{i=1}^{m} \mathbf{y}_{\mathbf{x}_i}(q))/(n)$; $\Pr(\mathrm{H}_0^q) = 1 - \Pr(\mathrm{H}_1^q)$;

(3) $\mathbf{u}(q) = \sum_{i=1}^{n} \mathbf{y}_{\mathbf{x}_i}(q)$; $\mathbf{u}'(q) = n - u(q)$;

EndFor

%For each test instance $\mathbf{x}$

(4) **Identify** $N(\mathbf{x})$ and $\mathbf{c}_{\mathbf{x}}$

%Counting the training instances whose membership counting vectors satisfy the constraints (15)

(5) **For** $q = 1, \ldots, Q$

(6) $\mathbf{v}(q) = 0$; $\mathbf{v}'(q) = 0$

EndFor

(7) **For** $i = 1, \ldots, n$

(8) **Identify** $N(\mathbf{x}_i)$ and $\mathbf{c}_{\mathbf{x}_i}$

(9) **If** $\mathbf{c}_{\mathbf{x}}(q) - \delta \leq \mathbf{c}_{\mathbf{x}_i}(q) \leq \mathbf{c}_{\mathbf{x}}(q) + \delta, \forall q \in \mathcal{Y}$ **Then**

(10) **For** $q = 1, \ldots, Q$

(11) **If** $\mathbf{c}_{\mathbf{x}_i}(q) == \mathbf{c}_{\mathbf{x}}(q)$ **Then**

(12) **If** $\mathbf{y}_{\mathbf{x}_i}(q) == 1$ **Then** $\mathbf{v}(q) = \mathbf{v}(q) + 1$;
Else $\mathbf{v}'(q) = \mathbf{v}'(q) + 1$;

EndFor

EndFor

%Computing $\mathbf{y}_{\mathbf{x}}$ and $\mathbf{r}_{\mathbf{x}}$

(13) **For** $q = 1, \ldots, Q$

(14) $\Pr(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l \in \mathcal{Y} \setminus \{\omega_q\}} \mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)} \mid \mathrm{H}_1^q) = (s + \mathbf{v}(q))/(s \times Q + \mathbf{u}(q))$;

(15) $\Pr(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l \in \mathcal{Y} \setminus \{\omega_q\}} \mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)} \mid \mathrm{H}_0^q) = (s + \mathbf{v}'(q))/(s \times Q + \mathbf{u}'(q))$;

(16) $\mathbf{y}_{\mathbf{x}}(q) = \arg\max_{b \in \{0,1\}} \Pr(\mathrm{H}_b^q) \Pr(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l \in \mathcal{Y} \setminus \{\omega_q\}} \mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)} \mid \mathrm{H}_b^q)$

(17) $$\mathbf{r}_{\mathbf{x}}(q) = \frac{\Pr(\mathrm{H}_1^q) \Pr(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l \in \mathcal{Y} \setminus \{\omega_q\}} \mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)} \mid \mathrm{H}_1^q)}{\sum_{b \in \{0,1\}} \Pr(\mathrm{H}_b^q) \Pr(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l \in \mathcal{Y} \setminus \{\omega_q\}} \mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)} \mid \mathrm{H}_b^q)}$$

EndFor

Algorithm 1: DMLkNN algorithm.

In order to determine the category vector $\hat{\mathbf{y}}_{\mathbf{x}}$ and the real-valued vector $\mathbf{r}_{\mathbf{x}}$ of instance $\mathbf{x}$, we need to determine the prior probabilities $\Pr(\mathrm{H}_b^l)$ and the likelihoods $\Pr(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l \in \mathcal{Y} \setminus \{\omega_q\}} \mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)} \mid \mathrm{H}_b^q)$, for each $q \in \{1, \ldots, Q\}$, and $b \in \{0, 1\}$. These probabilities are estimated from a training dataset $\mathcal{D}$.

Given an instance $\mathbf{x}$ to be classified, the output of the DMLkNN method for multilabel classification is determined as follows:

$$\begin{aligned} \mathcal{H}(\mathbf{x}) &= \left\{\omega_q \in \mathcal{Y} \mid \hat{\mathbf{y}}_{\mathbf{x}}(q) = 1\right\}, \\ f\left(\mathbf{x}, \omega_q\right) &= \mathbf{r}_{\mathbf{x}}(q), \quad \text{for each } \omega_q \in \mathcal{Y}. \end{aligned} \tag{12}$$

Algorithm 1 shows the pseudocode of the DMLkNN algorithm. The value of δ may be selected through cross-validation and provided as input to the algorithm. The prior probabilities $\Pr(\mathrm{H}_b^q)$, $b = \{0, 1\}$, for each class ω_{q} are first calculated and the instances belonging to each label are counted (steps (1) to (3)):

$$\begin{aligned} \Pr\left(\mathrm{H}_1^q\right) &= \frac{1}{n} \sum_{i=1}^{n} \mathbf{y}_{\mathbf{x}_i}(q), \\ \Pr\left(\mathrm{H}_0^q\right) &= 1 - \Pr\left(\mathrm{H}_1^q\right). \end{aligned} \tag{13}$$

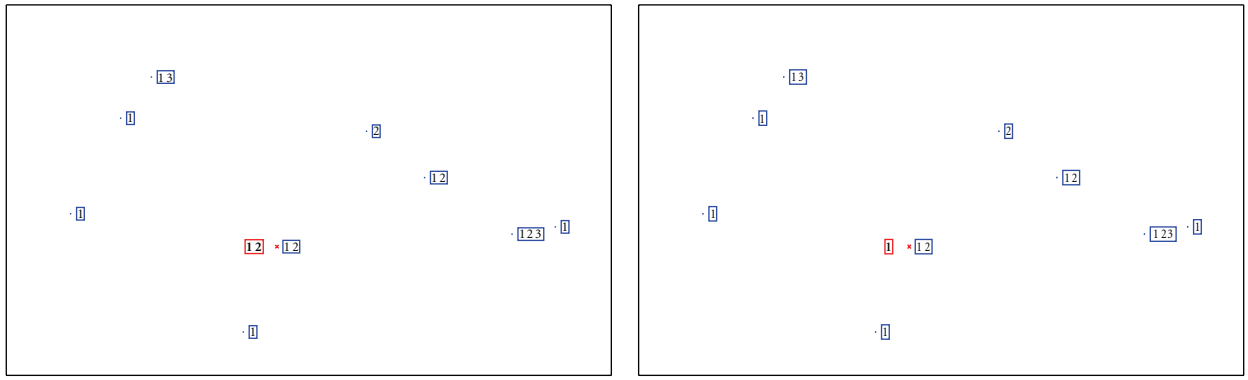
Recall that n is the number of training instances. $\mathbf{u}(q)$ is the number of instances belonging to class ω_q, and $\mathbf{u}'(q)$ indicates the number of instances not having ω_q in their label sets:

$$\begin{aligned} \mathbf{u}(q) &= \sum_{i=1}^{n} \mathbf{y}_{\mathbf{x}_i}(q), \\ \mathbf{u}'(q) &= n - \mathbf{u}(q). \end{aligned} \tag{14}$$

For test instance $\mathbf{x}$, the k-NNs are identified and the membership counting vector $\mathbf{c}_{\mathbf{x}}$ is determined (step (4)). To decide whether or not to assign the label ω_q to $\mathbf{x}$, we must determine the likelihoods $\Pr(\mathrm{E}^q_{\mathbf{c}_{\mathbf{x}}(q)}, \bigwedge_{\omega_l \in \mathcal{Y} \setminus \{\omega_q\}} \mathrm{F}^l_{\mathbf{c}_{\mathbf{x}}(l)} \mid \mathrm{H}_b^q)$, $b \in \{0, 1\}$, using the training instances such that their corresponding membership counting vectors satisfy the following constraints:

$$\begin{aligned} \mathbf{c}_{\mathbf{x}_i}(q) &= \mathbf{c}_{\mathbf{x}}(q), \\ \mathbf{c}_{\mathbf{x}}(l) - \delta \leq \mathbf{c}_{\mathbf{x}_i}(l) &\leq \mathbf{c}_{\mathbf{x}}(l) + \delta, \quad \text{for each } \omega_l \in \mathcal{Y} \setminus \left\{\omega_q\right\}. \end{aligned} \tag{15}$$

This is illustrated in steps (5) to (12). The number of instances from the training set satisfying these constraints and belonging to class ω_q is stored in $\mathbf{v}(q)$. The number of



(a) (b)

Figure 2: Estimated label set (in bold) for a test instance using the DMLkNN (a) and MLkNN (b) methods.

remaining instances satisfying the previous constraints and not having ω_q in their sets of labels is stored in $\mathbf{v}'(q)$. The likelihoods $\Pr(\mathrm{E}^q_{\mathbf{c}_\mathbf{x}(q)}, \bigwedge_{\omega_l \in \mathcal{Y}\setminus\{\omega_q\}} \mathrm{F}^l_{\mathbf{c}_\mathbf{x}(l)} \mid \mathrm{H}^q_b)$, $b \in \{0,1\}$, are then computed

$$\begin{aligned}
\Pr\left(\mathrm{E}^q_{\mathbf{c}_\mathbf{x}(q)}, \bigwedge_{\omega_l \in \mathcal{Y}\setminus\{\omega_q\}} \mathrm{F}^l_{\mathbf{c}_\mathbf{x}(l)} \mid \mathrm{H}^q_1\right) &= \frac{s + \mathbf{v}(l)}{s \times Q + \mathbf{u}(l)}, \\
\Pr\left(\mathrm{E}^q_{\mathbf{c}_\mathbf{x}(q)}, \bigwedge_{\omega_l \in \mathcal{Y}\setminus\{\omega_q\}} \mathrm{F}^l_{\mathbf{c}_\mathbf{x}(l)} \mid \mathrm{H}^q_0\right) &= \frac{s + \mathbf{v}'(l)}{s \times Q + \mathbf{u}'(l)},
\end{aligned} \tag{16}$$

where s is a smoothing parameter [19]. Smoothing is commonly used to avoid zero probability estimates. When $s = 1$, it is called Laplace smoothing. Finally, the category vector $\mathbf{y}_\mathbf{x}$ and the real-valued vector $\mathbf{r}_\mathbf{x}$ to rank labels in $\mathcal{Y}$ are calculated using (9) and (10), respectively (steps (13) to (17)).

Note that, in the MLkNN algorithm, only the first constraint in (15) is considered when computing the likelihoods $\Pr(\mathrm{E}^q_{\mathbf{c}_\mathbf{x}(q)} \mid \mathrm{H}^q_b)$, $b \in \{0,1\}$. As a result, the number of examples in the learning set satisfying this constraint is larger than the number of examples satisfying (15). MLkNN and DMLkNN should therefore not necessarily be compared using the same smoothing parameter.

4.3. Illustration on a Simulated Dataset. In this subsection, we illustrate the behavior of the DMLkNN and MLkNN methods using simulated data.

The simulated dataset contains 1019 instances in $\mathbb{R}^2$ belonging to three possible classes, $\mathcal{Y} = \{\omega_1, \omega_2, \omega_3\}$. The data were generated from seven Gaussian distributions with means $(0,0)$, $(1,0)$, $(0.5,0)$, $(0.5,1)$, $(0.25,0.6)$, $(0.75,0.6)$, $(0.5,0.5)$, respectively, and equal covariance matrix $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$. The number of instances in each class is chosen arbitrarily (see Table 1). Taking into account the geometric distribution of the Gaussian data, the instances of each set were, respectively, assigned to label(s) $\{\omega_1\}, \{\omega_2\}, \{\omega_1, \omega_2\}, \{\omega_3\}, \{\omega_1, \omega_3\}, \{\omega_2, \omega_3\}, \{\omega_1, \omega_2, \omega_3\}$.

Figure 2 shows the neighboring training instances and the estimated label set for a test instance $\mathbf{x}$ using DMLkNN and MLkNN. For both methods, k was set to 8, and Laplace smoothing ($s = 1$) was used. For DMLkNN, δ was set to 1. Below we describe the different steps in the estimation of the label set of $\mathbf{x}$ using the DMLkNN and MLkNN algorithms applied to the test data. For the sake of clarity we recall some definitions of events already given above. The membership counting vector of the test instance is $\mathbf{c}_\mathbf{x} = (7,3,2)$. Using the DML$k$NN method, in order to estimate the label set of $\mathbf{x}$, the following probabilities need to be computed from (9):

$$\begin{aligned}
\hat{\mathbf{y}}_\mathbf{x}(1) &= \arg\max_{b\in\{0,1\}} \Pr\left(\mathrm{H}^1_b\right)\Pr\left(\mathrm{E}^1_7, \mathrm{F}^2_3, \mathrm{F}^3_2 \mid \mathrm{H}^1_b\right), \\
\hat{\mathbf{y}}_\mathbf{x}(2) &= \arg\max_{b\in\{0,1\}} \Pr\left(\mathrm{H}^2_b\right)\Pr\left(\mathrm{E}^2_3, \mathrm{F}^1_7, \mathrm{F}^3_2 \mid \mathrm{H}^2_b\right), \\
\hat{\mathbf{y}}_\mathbf{x}(3) &= \arg\max_{b\in\{0,1\}} \Pr\left(\mathrm{H}^3_b\right)\Pr\left(\mathrm{E}^3_2, \mathrm{F}^1_7, \mathrm{F}^2_3 \mid \mathrm{H}^3_b\right).
\end{aligned} \tag{17}$$

We recall that E^1_7 is the event that there are seven instances in $\mathcal{N}^k_\mathbf{x}$ which have label ω_1, and F^2_3 is the event that the number of instances in $\mathcal{N}^k_\mathbf{x}$ belonging to label ω_2 is in the interval $[3-\delta; 3+\delta] = [2,4]$. In contrast, for estimating the label set of the unseen instance using the MLkNN method, the following probabilities must be computed from (8):

$$\begin{aligned}
\hat{\mathbf{y}}'_\mathbf{x}(1) &= \arg\max_{b\in\{0,1\}} \Pr\left(\mathrm{H}^1_b\right)\Pr\left(\mathrm{E}^1_7 \mid \mathrm{H}^1_b\right), \\
\hat{\mathbf{y}}'_\mathbf{x}(2) &= \arg\max_{b\in\{0,1\}} \Pr\left(\mathrm{H}^2_b\right)\Pr\left(\mathrm{E}^2_3 \mid \mathrm{H}^2_b\right), \\
\hat{\mathbf{y}}'_\mathbf{x}(3) &= \arg\max_{b\in\{0,1\}} \Pr\left(\mathrm{H}^3_b\right)\Pr\left(\mathrm{E}^3_2 \mid \mathrm{H}^3_b\right).
\end{aligned} \tag{18}$$

First, the prior probabilities are computed from the training set according to (13):

$$\begin{aligned}
\Pr\left(\mathrm{H}^1_1\right) &= 0.4527, \qquad \Pr\left(\mathrm{H}^1_0\right) = 0.5473, \\
\Pr\left(\mathrm{H}^2_1\right) &= 0.5038, \qquad \Pr\left(\mathrm{H}^2_0\right) = 0.4962, \\
\Pr\left(\mathrm{H}^3_1\right) &= 0.4396, \qquad \Pr\left(\mathrm{H}^3_0\right) = 0.5604.
\end{aligned} \tag{19}$$

Secondly, the posterior probabilities for the DMLkNN and MLkNN algorithms are calculated using the training set (For

DMLkNN, this is done according to steps (7) to (15), as shown in Algorithm 1 and explained in Section 4.2.)

$$\begin{aligned}
&\Pr\left(\mathrm{E}_7^1, \mathrm{F}_3^2, \mathrm{F}_2^3 \mid \mathrm{H}_1^1\right) = 0.0478, \quad \Pr\left(\mathrm{E}_7^1, \mathrm{F}_3^2, \mathrm{F}_2^3 \mid \mathrm{H}_0^1\right) = 0.0139,\\
&\Pr\left(\mathrm{E}_3^2, \mathrm{F}_7^1, \mathrm{F}_2^3 \mid \mathrm{H}_1^2\right) = 0.0237, \quad \Pr\left(\mathrm{E}_3^2, \mathrm{F}_7^1, \mathrm{F}_2^3 \mid \mathrm{H}_0^2\right) = 0.0218,\\
&\Pr\left(\mathrm{E}_2^3, \mathrm{F}_7^1, \mathrm{F}_3^2 \mid \mathrm{H}_1^3\right) = 0.0394, \quad \Pr\left(\mathrm{E}_2^3, \mathrm{F}_7^1, \mathrm{F}_3^2 \mid \mathrm{H}_0^3\right) = 0.1161,\\
&\Pr\left(\mathrm{E}_7^1 \mid \mathrm{H}_1^1\right) = 0.1108, \quad \Pr\left(\mathrm{E}_7^1 \mid \mathrm{H}_0^1\right) = 0.0431,\\
&\Pr\left(\mathrm{E}_3^2 \mid \mathrm{H}_1^2\right) = 0.1231, \quad \Pr\left(\mathrm{E}_3^2 \mid \mathrm{H}_0^2\right) = 0.1746,\\
&\Pr\left(\mathrm{E}_2^3 \mid \mathrm{H}_1^3\right) = 0.0655, \quad \Pr\left(\mathrm{E}_2^3 \mid \mathrm{H}_0^3\right) = 0.0593.
\end{aligned} \tag{20}$$

Using the prior and the posterior probabilities, the category vectors associated to the test instance by the DMLkNN and MLkNN algorithms can be calculated

$$\begin{aligned}
&\hat{\mathbf{y}}_{\mathbf{x}}(1) = 1, \quad \hat{\mathbf{y}}'_{\mathbf{x}}(1) = 1,\\
&\hat{\mathbf{y}}_{\mathbf{x}}(2) = 1, \quad \hat{\mathbf{y}}'_{\mathbf{x}}(2) = 0,\\
&\hat{\mathbf{y}}_{\mathbf{x}}(3) = 0, \quad \hat{\mathbf{y}}'_{\mathbf{x}}(3) = 0.
\end{aligned} \tag{21}$$

Thus, the estimated label set for instance $\mathbf{x}$ given by the DMLkNN method is $\hat{Y} = \{\omega_1, \omega_2\}$, while that given by ML$k$NN is $\hat{Y}' = \{\omega_1\}$. The true label set for $\mathbf{x}$ is $Y = \{\omega_1, \omega_2\}$. Here, we can see that no error has occurred when estimating the label set of $\mathbf{x}$ using the DMLkNN method, while for MLkNN the estimated label set is not identical to the ground truth label set. Seven training instances in $\mathcal{N}_{\mathbf{x}}^k$ have class ω_1 in their label sets, while only three instances belong to ω_2. In fact, the existence of class ω_1 in the neighborhood of $\mathbf{x}$ gives some information about the existence or not of class ω_2 in the label set of $\mathbf{x}$. If we take a look at the training dataset, we remark that 14.7% of instances belong to ω_1, 15.9% to ω_2, and 29.8% to ω_1 and ω_2 simultaneously. Thus, the probability that an instance belongs to both classes ω_1 and ω_2 is approximately twice the probability that it belongs to only one of the two classes. DMLkNN is able to capture the relationship between classes ω_1 and ω_2, while MLkNN is not able to capture this relation. This example shows that the DMLkNN method, which takes into account the dependencies between labels, may improve the classification performance and estimate the label sets of test instances with greater accuracy.

5. Experiments

In this section, we report a comparative study between DMLkNN and some state-of-the-art methods on several datasets collected from real world applications, and using different evaluation metrics.

5.1. Evaluation Metrics. There exist a number of evaluation criteria that evaluate the performance of a multilabel learning system, given a set $\mathcal{D} = \{(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_n, Y_n)\}$ of n test examples. We now describe some of the main evaluation criteria used in the literature to evaluate a multilabel learning system [3, 7]. The evaluation metrics can be divided into two groups: *prediction-based* and *ranking-based* metrics. Prediction-based metrics assess the *correctness* of the label sets predicted by the multilabel classifier $\mathcal{H}$, while ranking-based metrics evaluate the label ranking quality depending on the scoring function f. Since not all multilabel classification methods compute a scoring function, prediction-based metrics are of more general use.

Table 1: Summary of the simulated data set.

Label set	Number of instances
$\{\omega_1\}$	150
$\{\omega_2\}$	162
$\{\omega_1, \omega_2\}$	304
$\{\omega_3\}$	262
$\{\omega_1, \omega_3\}$	43
$\{\omega_2, \omega_3\}$	78
$\{\omega_1, \omega_2, \omega_3\}$	20

5.1.1. Prediction-Based Metrics

Accuracy. The accuracy metric is an average degree of similarity between the predicted and the ground truth label sets of all test examples:

$$\mathrm{Acc}\,(\mathcal{H}, \mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|Y_i \cap \hat{Y}_i\right|}{\left|Y_i \cup \hat{Y}_i\right|}, \tag{22}$$

where $\hat{Y}_i = \mathcal{H}(\mathbf{x}_i)$ denotes the predicted label set of instance $\mathbf{x}_i$.

F1-Measure. The F1-measure is defined as the harmonic mean of two other metrics known as precision (Prec) and recall (Rec) [20]. The former computes the proportion of correct positive predictions while the latter calculates the proportion of true labels that have been predicted as positives. These metrics are defined as follows:

$$\begin{aligned}
\mathrm{Prec}\,(\mathcal{H}, \mathcal{D}) &= \frac{1}{n}\sum_{i=1}^{n}\frac{\left|Y_i \cap \hat{Y}_i\right|}{\left|\hat{Y}_i\right|},\\
\mathrm{Rec}\,(\mathcal{H}, \mathcal{D}) &= \frac{1}{n}\sum_{i=1}^{n}\frac{\left|Y_i \cap \hat{Y}_i\right|}{|Y_i|},\\
F1(\mathcal{H}, \mathcal{D}) &= \frac{2 \cdot \mathrm{Prec} \cdot \mathrm{Rec}}{\mathrm{Prec} + \mathrm{Rec}} = \frac{1}{n}\sum_{i=1}^{n}\frac{2\left|Y_i \cap \hat{Y}_i\right|}{|Y_i| + \left|\hat{Y}_i\right|}.
\end{aligned} \tag{23}$$

Hamming Loss. This metric counts prediction errors (an incorrect label is predicted) and missing errors (a true label is not predicted)

$$\mathrm{HLoss}\,(\mathcal{H}, \mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{Q}\left|Y_i \Delta \hat{Y}_i\right|, \tag{24}$$

Table 2: Characteristics of datasets.

Dataset	Domain	Number of instances	Feature vector dimension	Number of labels	Label cardinality	Label density	Distinct label sets
Emotion	Music	593	72	6	1.868	0.311	27
Scene	Image	2407	294	6	1.074	0.179	15
Yeast	Biology	2417	103	14	4.237	0.303	198
Medical	Text	978	1449	45	1.245	0.028	94
Enron	Text	1702	1001	53	3.378	0.064	753

Table 3: Characteristics of the webpage categorization dataset.

	Number of instances	Feature vector dimension	Number of labels	Label cardinality	Label density	Distinct label sets
Arts and Humanities	5000	462	26	1.636	0.063	462
Business and Economy	5000	438	30	1.588	0.053	161
Computers and Internet	5000	681	33	1.508	0.046	253
Education	5000	550	33	1.461	0.044	308
Entertainment	5000	640	21	1.420	0.068	232
Health	5000	612	32	1.662	0.052	257
Recreation and Sports	5000	606	22	1.423	0.065	322
Reference	5000	793	33	1.169	0.035	217
Science	5000	743	40	1.451	0.036	398
Social and Science	5000	1047	39	1.283	0.033	226
Society and Culture	5000	636	27	1.692	0.063	582

where Δ stands for the symmetric difference between two sets.

Note that the values of the prediction-based evaluation criteria are in the interval $[0,1]$. Larger values of the first four metrics correspond to higher classification quality, while for the Hamming loss metric, the smaller the symmetric difference between predicted and true label sets, the better the performance [7, 20].

5.1.2. Ranking-Based Metrics. As stated before, this group of criteria is based on the scoring function $f(\cdot,\cdot)$ and evaluates the ranking quality of the different possible labels [6, 10]. Let $\text{rank}_f(\cdot,\cdot)$ be the ranking function derived from f and taking values in $\{1,\ldots,Q\}$. For each instance $\mathbf{x}_i$, the label with the highest scoring value has rank 1, and if $f(\mathbf{x}_i,\omega_q) > f(\mathbf{x}_i,\omega_r)$, then $\text{rank}_f(\mathbf{x}_i,\omega_q) < \text{rank}_f(\mathbf{x}_i,\omega_r)$.

One-Error. The one-error metric evaluates how many times the top-ranked label, that is, the label with the highest score, is not in the true set of labels of the instance:

$$\text{OErr}(f,\mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n}\left\langle\left[\arg\max_{\omega\in Y} f(\mathbf{x}_i,\omega)\right] \notin Y_i\right\rangle, \quad (25)$$

where for any proposition H, $\langle H\rangle$ equals to 1 if H holds and 0 otherwise. Note that, for single-label classification problems, the one-error is identical to ordinary classification error.

Coverage. The coverage measure is defined as the average number of steps needed to move down the ranked label list in order to cover all the labels assigned to a test instance:

$$\text{Cov}(f,\mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n}\max_{\omega\in Y_i}\text{rank}_f(\mathbf{x}_i,\omega) - 1. \quad (26)$$

Ranking Loss. This metric calculates the average fraction of label pairs that are reversely ordered for an instance:

$$\begin{aligned}\text{RLoss}(f,\mathcal{D}) &= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|Y_i|\left|\overline{Y}_i\right|} \\ &\quad\times\left|\left\{\left(\omega_q,\omega_r\right)\in Y_i\times\overline{Y}_i \mid f\left(\mathbf{x}_i,\omega_q\right)\le f(\mathbf{x}_i,\omega_r)\right\}\right|,\end{aligned} \quad (27)$$

where $\overline{Y}_i$ denotes the complement of Y_i in $\mathcal{Y}$.

Average Precision. This criterion was first used in information retrieval and was then adapted to multilabel learning problems in order to evaluate the effectiveness of label ranking. This metric measures the average fraction of labels

Table 4: Experimental results (mean ± std) on the emotion dataset.

	DMLkNN	MLkNN	BRkNN	MLRBF	RankSVM
Acc^{+}	**0.562 ± 0.029**	0.536 ± 0.032$^{\bullet}$	0.551 ± 0.030$^{\bullet}$	0.548 ± 0.029$^{\bullet}$	0.476 ± 0.027$^{\bullet}$
Prec^{+}	**0.691 ± 0.032**	0.674 ± 0.033$^{\bullet}$	0.689 ± 0.033°	0.686 ± 0.037°	0.601 ± 0.031$^{\bullet}$
Rec^{+}	**0.653 ± 0.030**	0.622 ± 0.041$^{\bullet}$	0.637 ± 0.031$^{\bullet}$	0.639 ± 0.032$^{\bullet}$	0.589 ± 0.032$^{\bullet}$
F1^{+}	**0.671 ± 0.028**	0.648 ± 0.033$^{\bullet}$	0.663 ± 0.029$^{\bullet}$	0.662 ± 0.031$^{\bullet}$	0.592 ± 0.027$^{\bullet}$
HLoss^{-}	**0.189 ± 0.015**	0.197 ± 0.015$^{\bullet}$	0.190 ± 0.016°	0.191 ± 0.015°	0.221 ± 0.016$^{\bullet}$
OErr^{-}	0.266 ± 0.033$^{\bullet}$	0.285 ± 0.035$^{\bullet}$	0.261 ± 0.036$^{\bullet}$	**0.255 ± 0.045**	0.313 ± 0.039$^{\bullet}$
Cov^{-}	**1.762 ± 0.111**	1.803 ± 0.115$^{\bullet}$	1.789 ± 0.125$^{\bullet}$	1.765 ± 0.120°	1.875 ± 0.117$^{\bullet}$
RLoss^{-}	0.161 ± 0.019$^{\bullet}$	0.167 ± 0.021$^{\bullet}$	0.190 ± 0.017$^{\bullet}$	**0.159 ± 0.021**	0.181 ± 0.021$^{\bullet}$
AvPrec^{+}	0.804 ± 0.019°	0.794 ± 0.022$^{\bullet}$	0.798 ± 0.020$^{\bullet}$	**0.809 ± 0.024**	0.779 ± 0.020$^{\bullet}$
AvRank	1.4	4	2.5	2.1	5

$^{+(-)}$: the higher (smaller) the value, the better the performance.
$\bullet^{(\circ)}$: statistically significant (nonsignificant) difference of performance as compared to the best result in bold, based on two-tailed paired t-test at 5% significance.

Table 5: Experimental results (mean ± std) on the scene dataset.

	DMLkNN	MLkNN	BRkNN	MLRBF	RankSVM
Acc^{+}	**0.676 ± 0.015**	0.668 ± 0.020$^{\bullet}$	0.658 ± 0.018$^{\bullet}$	0.631 ± 0.016$^{\bullet}$	0.521 ± 0.016$^{\bullet}$
Prec^{+}	**0.704 ± 0.017**	0.695 ± 0.021$^{\bullet}$	0.684 ± 0.019$^{\bullet}$	0.652 ± 0.017$^{\bullet}$	0.505 ± 0.019$^{\bullet}$
Rec^{+}	**0.677 ± 0.015**	0.669 ± 0.022°	0.661 ± 0.018$^{\bullet}$	0.644 ± 0.018$^{\bullet}$	0.660 ± 0.017$^{\bullet}$
F1^{+}	**0.692 ± 0.016**	0.683 ± 0.023$^{\bullet}$	0.672 ± 0.019$^{\bullet}$	0.649 ± 0.017$^{\bullet}$	0.526 ± 0.017$^{\bullet}$
HLoss^{-}	**0.084 ± 0.004**	0.087 ± 0.003°	0.092 ± 0.005$^{\bullet}$	0.086 ± 0.003°	0.135 ± 0.004$^{\bullet}$
OErr^{-}	0.219 ± 0.017$^{\bullet}$	0.228 ± 0.016$^{\bullet}$	0.245 ± 0.018$^{\bullet}$	**0.206 ± 0.015**	0.279 ± 0.017$^{\bullet}$
Cov^{-}	0.461 ± 0.035°	0.476 ± 0.035$^{\bullet}$	0.558 ± 0.042$^{\bullet}$	**0.451 ± 0.041**	0.939 ± 0.041$^{\bullet}$
RLoss^{-}	**0.071 ± 0.007**	0.077 ± 0.009°	0.110 ± 0.009$^{\bullet}$	0.072 ± 0.008°	0.118 ± 0.009$^{\bullet}$
AvPrec^{+}	0.869 ± 0.010°	0.865 ± 0.009$^{\bullet}$	0.843 ± 0.011$^{\bullet}$	**0.876 ± 0.009**	0.801 ± 0.011$^{\bullet}$
AvRank	1.3	2.5	3.5	2.5	5

$^{+(-)}$: the higher (smaller) the value, the better the performance.
$\bullet^{(\circ)}$: statistically significant (nonsignificant) difference of performance as compared to the best result in bold, based on two-tailed paired t-test at 5% significance.

ranked above a particular label $\omega_q \in Y_i$ which actually are in Y_i:

$$\begin{aligned}\text{AvPrec}\left(f,\mathcal{D}\right) &= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|Y_i|} \\ &\times \sum_{\omega_q\in Y_i}\frac{\left|\left\{\omega_r \in Y_i \mid \text{rank}_f(\mathbf{x}_i,\omega_r) \le \text{rank}_f\left(\mathbf{x}_i,\omega_q\right)\right\}\right|}{\text{rank}_f\left(\mathbf{x}_i,\omega_q\right)}.\end{aligned} \tag{28}$$

For the ranking-based metrics, smaller values of the first three metrics correspond to better label ranking quality, while AvPrec$(f,\mathcal{D}) = 1$ means that the labels are perfectly ranked for all test examples [6].

5.2. Benchmark Datasets. Given a multilabeled dataset $\mathcal{D} = \{(\mathbf{x}_i, Y_i),\ i = 1,\ldots,n\}$ with $\mathbf{x}_i \in \mathbb{X}$ and $Y_i \subseteq \mathcal{Y}$, the following measures give some statistics about the "label multiplicity" of the dataset $\mathcal{D}$ [7]:

(i) The *label cardinality* of $\mathcal{D}$, denoted by LCard$(\mathcal{D})$, indicates the average number of labels per instance:

$$\text{LCard}\left(\mathcal{D}\right) = \frac{1}{n}\sum_{i=1}^{n}|Y_i|. \tag{29}$$

(ii) The *label density* of $\mathcal{D}$, denoted by LDen$(\mathcal{D})$, is defined as the average number of labels per instance divided by the number of possible labels Q:

$$\text{LDen}\left(\mathcal{D}\right) = \frac{\text{LCard}(\mathcal{D})}{Q}. \tag{30}$$

(iii) DL$(\mathcal{D})$ counts the number of *distinct label sets* appeared in the dataset $\mathcal{D}$:

$$\text{DL}(\mathcal{D}) = \left|\left\{Y_i \subseteq \mathcal{Y} \mid \exists \mathbf{x}_i \in \mathbb{X} : (\mathbf{x}_i, Y_i) \in \mathcal{D}\right\}\right|. \tag{31}$$

Several real datasets were used in our experiments. The datasets used are from different application domains, namely, text categorization, bioinformatics and multimedia applications (music and image). These datasets can be downloaded from http://mlkd.csd.auth.gr/multilabel.html.

(i) The *emotion dataset*, presented in [21], consists of 593 songs annotated by experts according to the emotions they generate. The emotions are: amazed-surprised, happy-pleased, relaxing-calm, quiet-still, sad-lonely, and angry-fearful. Each emotion corresponds to a class. Consequently there are 6 classes, and each song was labeled as belonging to one or several classes. Each song was also described by 8 rhythmic features and 64 timbre features, resulting in a total of 72 features. The number of distinct label sets is equal to 27, the label cardinality is 1.868, and the label density is 0.311.

(ii) The *scene dataset* consists of 2407 images of natural scenery. For each image, spatial color moments are used as features. Images are divided into 49 blocks using a 7×7 grid. The mean and variance of each band are computed corresponding to a low-resolution image and to computationally inexpensive texture features, respectively [1]. Each image is then transformed into a $49 \times 3 \times 2 = 294$-dimensional feature vector. A label set is manually assigned to each image. There are 6 different semantic types: *beach*, *sunset*, *field*, *fall-foliage*, *urban,* and *mountain*. The average number of labels per instance is 1.074, thus the label density is 0.179. The number of distinct sets of labels is equal to 15.

(iii) The *yeast dataset* contains data regarding the gene functional classes of the yeast *Saccharomyces cerevisiae* [6]. It includes 2417 genes, each of which is represented by 103 features. A gene is described by the concatenation of microarray expression data and a phylogenetic profile and is associated with a set of functional classes. There are 14 possible classes and there exist 198 distinct label combinations. The label cardinality is 4.237, and the label density is 0.303.

(iv) The *medical dataset* consists of 978 examples, each example represented by 1449 features. This dataset was provided by the *Computational Medicine Center* as part of a challenge task involving the automated processing of free clinical text, and is the dataset used in [8]. The average cardinality is 1.245, and the label density is 0.028 with 94 distinct label sets.

(v) The *Enron email dataset* consists of 1702 examples, each represented by 1001 features. It comprises email messages belonging to users, mostly senior management of the *Enron Corp*. This dataset was used in [8]. 753 distinct label combinations exist in the dataset. The label cardinality is 3.378, and the label density is 0.064.

Table 2 summarizes the characteristics of the emotion, scene, yeast, medical, and Enron datasets.

(vi) The *webpage categorization dataset* was investigated in [10, 22]. The data were collected from the "http://www.yahoo.com/" domain. Eleven different webpage categorization subproblems are considered, corresponding to 11 different categories: Arts and Humanities, Business and Economy, Computers and Internet, Education, Entertainment, Health, Recreation and Sports, Reference, Science, Social and Science, and Society and Culture. Each subproblem consists of 5000 documents. Over the 11 subproblems, the number of categories varies from 21 to 40 and the instance dimensionality varies from 438 to 1.047. Table 3 shows the statistics of the different subproblems within the webpage dataset.

5.3. Experimental Results. The DMLkNN method was compared to two other binary relevance-based approaches, namely, MLkNN and BRkNN. The model parameters for DMLkNN are the number of neighbors k, the *fuzziness* parameter δ, and the smoothing parameter s. Parameter tuning can be done via cross-validation. For fair comparison, k was set to 10 for the three methods, and s was set to 1, as in [10]. Note that as stated in Section 4.2, the parameter δ should be set to a small value. When k is set to 10, extensive experiments have shown that the value $\delta = 2$ generally gives good classification performances for DMLkNN.

In addition to the two k-NN based algorithms, our method was compared to two other state-of-the-art multi-label classification methods that have been shown to have good performances: MLRBF [13], derived from radial basis function neural networks, and RankSVM [6], based on the traditional support vector machine. As used in [13], the fraction parameter for MLRBF was set to 0.01 and the scaling factor to 1. For RankSVM, polynomial kernel was used as reported in [6].

For all k-NN based algorithms, the Euclidean distance was used. Note that usually, when feature variables are numeric and are not of comparable units and scales, that is, there are large differences in the ranges of values encountered (such as in the emotion dataset), the distance metric implicitly assigns greater weight to features with wide ranges than to those with narrow ranges. This may affect the nearest neighbors search. In such cases, feature normalization is recommended to approximately equalize the ranges of features so that they will have the same effect on distance computation [23]. In addition, we may remark that in the cases of the medical, and Enron datasets, the dimensions of feature vectors are very large as compared to the number of training instances (see Table 2). We applied the χ^2 statistic approach for feature selection on these two datasets, and we retained 20% of the most relevant features [24].

Five iterations of ten-fold cross-validation were performed on each dataset. Tables 4, 5, 6, 7, and 8 report the detailed results in terms of the different evaluation metrics for the emotion, scene, yeast, medical and Enron datasets, respectively. On the webpage dataset, ten-fold cross validation was performed on each subproblem, and Table 9 reports the average results.

For each method, the mean values of the different evaluation criteria, as well as the standard deviations (std)

Table 6: Experimental results (mean ± std) on the yeast dataset.

	DMLkNN	MLkNN	BRkNN	MLRBF	RankSVM
Acc^{+}	**0.511 ± 0.011**	0.508 ± 0.014°	0.510 ± 0.010°	0.510 ± 0.011$^{\bullet}$	0.492 ± 0.014$^{\bullet}$
Prec^{+}	**0.726 ± 0.014**	0.714 ± 0.015$^{\bullet}$	0.693 ± 0.014$^{\bullet}$	0.703 ± 0.013$^{\bullet}$	0.585 ± 0.021$^{\bullet}$
Rec^{+}	0.586 ± 0.012°	0.578 ± 0.017$^{\bullet}$	**0.599 ± 0.014**	0.594 ± 0.012°	0.547 ± 0.019$^{\bullet}$
F1^{+}	**0.623 ± 0.011**	0.612 ± 0.014$^{\bullet}$	0.615 ± 0.014$^{\bullet}$	0.616 ± 0.011$^{\bullet}$	0.556 ± 0.015$^{\bullet}$
HLoss^{-}	**0.192 ± 0.005**	0.194 ± 0.005°	0.199 ± 0.005$^{\bullet}$	0.197 ± 0.005$^{\bullet}$	0.202 ± 0.008$^{\bullet}$
OErr^{-}	**0.226 ± 0.021**	0.230 ± 0.017°	0.243 ± 0.019$^{\bullet}$	0.239 ± 0.019$^{\bullet}$	0.240 ± 0.023$^{\bullet}$
Cov^{-}	**6.240 ± 0.104**	6.275 ± 0.100$^{\bullet}$	6.631 ± 0.152$^{\bullet}$	6.489 ± 0.136$^{\bullet}$	6.997 ± 0.368$^{\bullet}$
RLoss^{-}	**0.165 ± 0.007**	0.167 ± 0.006°	0.210 ± 0.009$^{\bullet}$	0.175 ± 0.008$^{\bullet}$	0.183 ± 0.011$^{\bullet}$
AvPrec^{+}	**0.770 ± 0.010**	0.765 ± 0.010$^{\bullet}$	0.754 ± 0.011$^{\bullet}$	0.758 ± 0.011$^{\bullet}$	0.753 ± 0.014$^{\bullet}$
AvRank	1.2	2.4	3.5	2.6	4.8

$^{+(-)}$: the higher (smaller) the value, the better the performance.
$\bullet^{(\circ)}$: statistically significant (non-significant) difference of performance as compared to the best result in bold, based on two-tailed paired t-test at 5% significance.

Table 7: Experimental results (mean ± std) on the medical dataset.

	DMLkNN	MLkNN	BRkNN	MLRBF	RankSVM
Acc^{+}	0.634 ± 0.039$^{\bullet}$	0.609 ± 0.052$^{\bullet}$	0.565 ± 0.049$^{\bullet}$	**0.689 ± 0.029**	0.501 ± 0.041$^{\bullet}$
Prec^{+}	0.692 ± 0.037$^{\bullet}$	0.667 ± 0.048$^{\bullet}$	0.628 ± 0.048$^{\bullet}$	**0.709 ± 0.031**	0.522 ± 0.040$^{\bullet}$
Rec^{+}	**0.724 ± 0.041**	0.628 ± 0.053$^{\bullet}$	0.574 ± 0.048$^{\bullet}$	0.701 ± 0.025$^{\bullet}$	0.556 ± 0.038$^{\bullet}$
F1^{+}	**0.708 ± 0.037**	0.646 ± 0.050$^{\bullet}$	0.599 ± 0.051$^{\bullet}$	0.703 ± 0.027°	0.531 ± 0.036$^{\bullet}$
HLoss^{-}	0.015 ± 0.001$^{\bullet}$	0.015 ± 0.002$^{\bullet}$	0.016 ± 0.002$^{\bullet}$	**0.011 ± 0.002**	0.019 ± 0.002$^{\bullet}$
OErr^{-}	0.212 ± 0.044$^{\bullet}$	0.220 ± 0.052$^{\bullet}$	0.271 ± 0.048$^{\bullet}$	**0.153 ± 0.048**	0.215 ± 0.028$^{\bullet}$
Cov^{-}	2.454 ± 0.567$^{\bullet}$	2.514 ± 0.538$^{\bullet}$	3.218 ± 0.763$^{\bullet}$	**1.449 ± 0.296**	3.310 ± 0.781$^{\bullet}$
RLoss^{-}	0.035 ± 0.010$^{\bullet}$	0.037 ± 0.009$^{\bullet}$	0.099 ± 0.028$^{\bullet}$	**0.022 ± 0.009**	0.049 ± 0.019$^{\bullet}$
AvPrec^{+}	0.831 ± 0.026$^{\bullet}$	0.826 ± 0.033$^{\bullet}$	0.799 ± 0.029$^{\bullet}$	**0.898 ± 0.038**	0.791 ± 0.028$^{\bullet}$
AvRank	1.7	3	4.2	1.3	4.7

$^{+(-)}$: the higher (smaller) the value, the better the performance.
$\bullet^{(\circ)}$: statistically significant (non-significant) difference of performance as compared to the best result in bold, based on two-tailed paired t-test at 5% significance.

are given in the tables. A two-tailed paired t-test at 5% significance level was performed in order to determine the statistical significance of the obtained results in comparison with the best performances indicated in bold. In addition, for each dataset, the methods were ranked in decreasing order of performance. The average ranks over the different evaluation criteria are reported in the tables.

To give some idea about the computational complexity of the different algorithms, Table 10 provides the corresponding runtime statistics (in seconds) on the different datasets, using train/test experiments. All the algorithms were implemented in Matlab 7.4 and executed on the same computer (Intel Core Duo 2.13 GHz, 2 Go RAM).

Using the Sign test, a statistical comparison between the classifiers was made over the different datasets cited above. Table 11 reports the average ranking on each evaluation metric.

From the experimental results presented, the following observations can be made:

(i) DMLkNN performs better than MLkNN with respect to all evaluation metrics and on all datasets. In addition, DMLkNN performs better than BRkNN and is very competitive with the remaining methods that are based on more sophisticated classifiers (SVM and RBF). When the results obtained on the different datasets are averaged, DMLkNN gives the best performances with respect to all evaluation metrics apart from *One-error* and *Average-precision*. The next best results are obtained from MLRBF.

(ii) The experimental results show that, generally, DMLkNN performs better with respect to predicted-based metrics than with respect to ranking-based metrics, as for example on the emotion and scene datasets. In fact, for each instance to be classified, the output of DMLkNN is determined by combining binary decisions made about that instance's membership of the different classes. Thus, this method is concerned more with the pertinence of the predicted sets of labels than with the ranking of all labels. A ranking-based method, such as RankSVM, on the other hand, will normally tend to be more competitive with other methods as regards ranking-based metrics. This may be explained by the fact that ranking-based methods operate by ranking the labels

Table 8: Experimental results (mean ± std) on the Enron dataset.

	DMLkNN	MLkNN	BRkNN	MLRBF	RankSVM
Acc$^+$	**0.348 ± 0.033**	0.305 ± 0.035$^\bullet$	0.346 ± 0.025°	0.340 ± 0.031°	0.298 ± 0.041$^\bullet$
Prec$^+$	**0.646 ± 0.041**	0.572 ± 0.043$^\bullet$	0.602 ± 0.020$^\bullet$	0.587 ± 0.039$^\bullet$	0.525 ± 0.033$^\bullet$
Rec$^+$	0.378 ± 0.029°	0.343 ± 0.034$^\bullet$	0.382 ± 0.028°	**0.386 ± 0.038**	0.342 ± 0.041$^\bullet$
F1$^+$	**0.477 ± 0.034**	0.428 ± 0.038$^\bullet$	0.470 ± 0.027°	0.464 ± 0.040 $^\circ$	0.418 ± 0.030$^\bullet$
HLoss$^-$	**0.051 ± 0.001**	0.052 ± 0.001°	0.053 ± 0.002°	0.052 ± 0.001°	0.062 ± 0.006$^\bullet$
OErr$^-$	**0.270 ± 0.017**	0.271 ± 0.018°	0.304 ± 0.019$^\bullet$	0.281 ± 0.028°	0.324 ± 0.026$^\bullet$
Cov$^-$	**13.571 ± 0.308**	13.507 ± 0.342°	19.838 ± 0.467$^\bullet$	16.318 ± 0.689$^\bullet$	18.133 ± 0.671$^\bullet$
RLoss$^-$	**0.095 ± 0.004**	0.097 ± 0.004°	0.228 ± 0.014$^\bullet$	0.113 ± 0.009$^\bullet$	0.178 ± 0.012$^\bullet$
AvPrec$^+$	0.638 ± 0.008°	0.635 ± 0.009°	0.598 ± 0.015$^\bullet$	**0.642 ± 0.018**	0.586 ± 0.019$^\bullet$
AvRank	1.3	3	3.1	2.4	4.7

$^{+(-)}$: the higher (smaller) the value, the better the performance.
$\bullet^{(\circ)}$: statistically significant (non-significant) difference of performance as compared to the best result in bold, based on two-tailed paired t-test at 5% significance.

Table 9: Experimental results (mean ± std) on the webpage dataset.

	DMLkNN	MLkNN	BRkNN	MLRBF	RankSVM
Acc$^+$	0.296 ± 0.204$^\bullet$	0.285 ± 0.184$^\bullet$	0.286 ± 0.179$^\bullet$	**0.398 ± 0.146**	0.234 ± 0.171$^\bullet$
Prec$^+$	0.351 ± 0.257$^\bullet$	0.340 ± 0.227$^\bullet$	0.341 ± 0.211$^\bullet$	**0.462 ± 0.171**	0.228 ± 0.212$^\bullet$
Rec$^+$	0.308 ± 0.205$^\bullet$	0.291 ± 0.189$^\bullet$	0.296 ± 0.195$^\bullet$	**0.407 ± 0.153**	0.276 ± 0.186$^\bullet$
F1$^+$	0.319 ± 0.219$^\bullet$	0.304 ± 0.198$^\bullet$	0.317 ± 0.203$^\bullet$	**0.421 ± 0.156**	0.249 ± 0.195$^\bullet$
HLoss$^-$	0.041 ± 0.014$^\bullet$	0.043 ± 0.015$^\bullet$	0.052 ± 0.021$^\bullet$	**0.039 ± 0.013**	0.043 ± 0.014$^\bullet$
OErr$^-$	0.466 ± 0.165$^\bullet$	0.474 ± 0.157$^\bullet$	0.565 ± 0.184$^\bullet$	**0.375 ± 0.120**	0.440 ± 0.143$^\bullet$
Cov$^-$	**4.069 ± 1.255**	4.097 ± 1.237°	8.455 ± 1.887$^\bullet$	4.689 ± 1.403°	7.508 ± 2.396$^\bullet$
RLoss$^-$	**0.099 ± 0.046**	0.102 ± 0.045°	0.312 ± 0.101$^\bullet$	0.107 ± 0.039°	0.193 ± 0.065$^\bullet$
AvPrec$^+$	0.630 ± 0.120°	0.625 ± 0.116°	0.522 ± 0.151$^\bullet$	**0.688 ± 0.092**	0.601 ± 0.117$^\bullet$
AvRank	2	3.4	4.1	1.2	4.1

$^{+(-)}$: the higher (smaller) the value, the better the performance.
$\bullet^{(\circ)}$: statistically significant (non-significant) difference of performance as compared to the best result in bold, based on two-tailed paired t-test at 5% significance.

Table 10: Run time of the compared algorithms on the different datasets in seconds.

	DMLkNN	MLkNN	BRkNN	MLRBF	RankSVM
Emotion	0.506	0.268	0.126	0.696	3.635
Scene	9.984	5.963	3.067	3.851	22.319
Yeast	11.966	4.096	1.696	12.224	248.532
Medical	3.674	2.216	1.275	4.519	233.549
Enron	20.009	11.422	4.173	28.193	1781.644

according to their relevance to a given instance to be classified, and then splitting the ordered set of labels into subsets of relevant and irrelevant labels for that instance.

(iii) DMLkNN has good performances and is more competitive with the other algorithms on datasets with high label-density, such as on the emotion and yeast datasets. The proposed method is intended primarily to take into account the dependencies between labels, and DMLkNN tends to perform better on datasets with high label multiplicity, in which labels may be potentially more interdependent.

Table 11: Comparisons of the classifiers over the different datasets.

Acc$^+$	DMLkNN $\succ$ MLRBF > BRkNN $\succ$ MLkNN > RankSVM
Prec$^+$	DMLkNN $\succ$ MLRBF > MLkNN $\succ$ BRkNN > RankSVM
Rec$^+$	DMLkNN $\succ$ MLRBF > BRkNN $\succ$ MLkNN > RankSVM
F1$^+$	DMLkNN $\succ$ MLRBF > BRkNN $\succ$ MLkNN > RankSVM
HLoss$^-$	DMLkNN $\succ$ MLRBF $\succ$ MLkNN $\succ$ BRkNN > RankSVM
OErr$^-$	MLRBF $\succ$ DMLkNN > MLkNN $\succ$ BRkNN > RankSVM
Cov$^-$	DMLkNN $\succ$ MLkNN > MLRBF $\succ$ BRkNN > RankSVM
RLoss$^-$	DMLkNN $\succ$ MLRBF > MLkNN > BRkNN > RankSVM
AvPrec$^+$	MLRBF $\succ$ DMLkNN > MLkNN > BRkNN > RankSVM

>: statistically significant difference of performance based on Sign test;
$\succ$: non-significant difference of performance.

(iv) MLkNN is computationally faster than DMLkNN. In fact, in the MLkNN method, the likelihoods $\Pr(\mathrm{E}^q_{\mathbf{c}_\mathbf{x}(q)} \mid \mathrm{H}^q_b)$, $b \in \{0,1\}$, are calculated from the training set, stored, and then used only when predicting the label set of each query instance. In contrast, when DMLkNN is used, the number of likelihoods $\Pr(\mathrm{E}^q_{\mathbf{c}_\mathbf{x}(q)}, \bigwedge_{\omega_l \in \mathcal{Y}\setminus\{\omega_q\}} \mathrm{F}^l_{\mathbf{c}_\mathbf{x}(l)} \mid \mathrm{H}^q_b)$, $b \in \{0,1\}$, is far greater; consequently, unlike MLkNN,

it is impractical to calculate these probabilities in advance and then store them. There is therefore not a training process for DMLkNN. The probabilities are computed locally for each instance to be classified. Regarding the different methods, BRkNN is the fastest algorithm. Apart from the number of possible labels, the computing time of BRkNN depends primarily on the computing time of the nearest neighbors search. There are no prior and posterior probabilities to compute for BRkNN, as there are for DMLkNN and MLkNN. The RankSVM method requires the greatest computing time. For RankSVM, the resolution of the quadratic problem and the choice of the support vectors is known to be very hard [25]. The complexity of MLRBF depends primarily on the complexity of the K-means algorithm used for clustering the instances belonging to each possible class. MLRBF therefore has a linear complexity with respect to the number of classes, the number of clusters, the number of instances, and the dimensionality of the corresponding features vectors.

6. Conclusion

In this paper, we proposed an original multilabel learning algorithm derived from the k-NN rule, where the dependencies between labels are taken into account. Our method is based on the binary relevance approach, that is frequently criticized for ignoring possible correlations between labels [8], but here, this disadvantage is overcome. The classification of an instance is accomplished through the use of local information given by the k nearest neighbors, and by using the maximum a posteriori rule. This method, referred to as DMLkNN, generalizes the MLkNN algorithm presented in [10].

The proposed method is particularly useful in practical situations where data are *significantly* multilabeled. Using a variety of benchmark datasets and different evaluation criteria, the experimental results clearly demonstrate this and confirm the usefulness and the effectiveness of DMLkNN as compared to other state-of-the-art multilabel classification methods.

Note that when the number of classes grows, more data are required in order to compute the posterior probabilities for DMLkNN. On limited datasets, it will be hard to capture relationships between labels. In addition, the performances of the proposed method depend on the distance computation metric used to determine the nearest neighbors.

References

[1] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[2] A. McCallum, "Multi-label text classification with a mixture model trained by EM," in *Proceedings of the Working Notes of the AAAI Workshop on Text Learning*, 1999.

[3] R. E. Schapire and Y. Singer, "BoosTexter: a boosting-based system for text categorization," *Machine Learning*, vol. 39, no. 2, pp. 135–168, 2000.

[4] S. T. Dumais, J. Platt, D. Heckerman, and M. Sahami, "Inductive learning algorithms and representation for text categorization," in *Proceedings of the 7th ACM International Conference on Information and Knowledge Management*, pp. 148–155, 1998.

[5] S. Gao, W. Wu, C.-H. Lee, and T.-S. Chua, "A maximal figure-of-merit learning approach to text categorization," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrievel*, pp. 174–181, 2003.

[6] A. Elisseeff and J. Weston, "Kernel methods for Multi-labelled classification and categorical regression problems," *Advances in Neural Information Processing Systems*, vol. 14, pp. 681–687, 2002.

[7] G. Tsoumakas and I. Katakis, "Multi-label classification: an overview," *International Journal of Data Warehousing and Mining*, vol. 3, no. 3, pp. 1–13, 2007.

[8] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," in *Proceedings of the 20th European Conference on Machine Learning and Knowledge Discovery in Databases*, vol. 5782 of *Lecture Notes in Computer Science*, pp. 254–269, Bled, Slovenia, 2009.

[9] W. Cheng and E. Hüllermeier, "Combining instance-based learning and logistic regression for multilabel classification," *Machine Learning*, vol. 76, no. 2-3, pp. 211–225, 2009.

[10] M. L. Zhang and Z. H. Zhou, "ML-KNN: a lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[11] Z. Younes, F. Abdallah, and T. Denoeux, "Multi-label classification algorithm derived from k-nearest neighbor rule with label dependencies," in *Proceedings of the 16th European Signal Processing Conference*, pp. 25–29, Eusipco, 2008.

[12] Z. Younes, F. Abdallah, and T. Denœux, "An evidence-theoretic k-nearest neighbor rule for multi-label classification," in *Proceedings of the 3rd International Conference on Scalable Uncertainty Management (SUM '09)*, vol. 5785 of *Lecture Notes in Computer Science*, pp. 297–308, Springer, 2009.

[13] M. L. Zhang, "Ml-rbf: RBF neural networks for multi-label learning," *Neural Processing Letters*, vol. 29, no. 2, pp. 61–74, 2009.

[14] T. Denoeux, Z. Younes, and F. Abdallah, "Representing uncertainty on set-valued variables using belief functions," *Artificial Intelligence*, vol. 174, no. 7-8, pp. 479–499, 2010.

[15] P. Smets, "Combination of evidence in the transferable belief model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 5, pp. 447–458, 1990.

[16] R. Jin and Z. Ghahramani, "Learning with multiple labels," *Advances in Neural Information Processing Systems*, vol. 1540, no. 7, pp. 879–904, 2003.

[17] C. Shen, J. Jiao, Y. Yang, and B. Wang, "Multi-instance multi-label learning for automatic tag recommendation," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC '09)*, pp. 4910–4914, October 2009.

[18] C. Sutton, A. McCallum, and K. Rohanimanesh, "Dynamic conditional random fields: factorized probabilistic models for labeling and segmenting sequence data," *Journal of Machine Learning Research*, vol. 8, pp. 693–723, 2007.

[19] F. Peng, D. Schuurmans, and S. Wang, "Augmenting naive Bayes classifiers with statistical language models," *Information Retrieval*, vol. 7, no. 3-4, pp. 317–345, 2004.

[20] Y. Yang, "An evaluation of statistical approaches to text categorization," *Journal of Information Retrieval*, vol. 1, pp. 78–88, 1999.

[21] T. Konstantinos, G. Tsoumakas, G. Kalliris, and I. Vlahavas, "Multilabel classification of music into emotions," in *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR '08)*, Philadelphia, Pa, USA, 2008.

[22] N. Ueda and K. Saito, "Parametric mixture models for multi-label text," *Advances in Neural Information Processing Systems*, vol. 15, pp. 721–728, 2003.

[23] S. A. Dudani, "The distance-weighted k-nearest neighbor rule," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 6, no. 4, pp. 325–327, 1976.

[24] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Proceedings of the 4th International Conference on Machine Learning*, pp. 412–420, Nashville, Tenn, USA, 1997.

[25] M. Petrovskiy, "Paired comparisons method for solving multi-label learning problem," in *Proceedings of the 6th International Conference on Hybrid Intelligent Systems and Fourth Conference on Neuro-Computing and Evolving Intelligence (HIS-NCEI '06)*, IEEE, Auckland, New Zealand, 2006.