A Ranking-based KNN Approach for Multi-Label Classification

Tsung-Hsien Chiang r97041@csie.ntu.edu.tw
Hung-Yi Lo hungyi@iis.sinica.edu.tw
Shou-De Lin sdlin@csie.ntu.edu.tw
Graduate Institute of Computer Science and Information Engineering
National Taiwan University

Editor: Steven C.H. Hoi and Wray Buntine

Abstract

Multi-label classification has attracted a great deal of attention in recent years. This paper presents an interesting finding, namely, being able to identify neighbors with trustable labels can significantly improve the classification accuracy. Based on this finding, we propose a k-nearest-neighbor-based ranking approach to solve the multi-label classification problem. The approach exploits a ranking model to learn which neighbor's labels are more trustable candidates for a weighted KNN-based strategy, and then assigns higher weights to those candidates when making weighted-voting decisions. The weights can then be determined by using a generalized pattern search technique. We collect several real-word data sets from various domains for the experiment. Our experiment results demonstrate that the proposed method outperforms state-of-the-art instance-based learning approaches. We believe that appropriately exploiting k-nearest neighbors is useful to solve the multi-label problem.

Keywords: Multi-Label classification, Nearest neighbor classification, Ranking, Optimization, Generalized pattern search

1. Introduction

1.1. Background

In a conventional classification task, given a set of m possible disjoint classes, each instance is associated with one and only one class. Different kinds of machine learning algorithms, such as the k-nearest neighbor, support vector machine, and logistic regression methods, have been proposed to resolve such classification problem, and have achieved a satisfactory level of success. However, in some real-world problems, each instance could be associated with multiple classes simultaneously, i.e., an instance could have a set of labels. For example, a song might be annotated as both R&B and Funk. Such prediction tasks are usually denoted as multi-label classification problems. In fact, a conventional classification problem is simply a special case of the multi-label classification problem where only one label can be correct. Multi-label classification problems exist in several domains. In text mining, a document can be associated with several topics, such as the arts and history; and in gene functional analysis in bioinformatics, each gene can belong to multiple functional classes, e.g., metabolism and transcription classes. Meanwhile, in image classification tasks, an image may contain several concepts simultaneously, such as mountain and sunset.

Because of the increase in online labeling/tagging services and the growth in the volume of data, multi-label classification has attracted a great deal of attention in recent years. Since each instance may be associated with multiple classes simultaneously, exploiting the relationships among labels effectively is obviously crucial to the success of a multi-label classification algorithm. For example, if a picture is annotated as *sea*, it should have a higher probability of also being annotated as *fish* than as *cow*. The results of the past study indicate that exploiting the correlations and interdependency between labels can improve the prediction accuracy (Tsoumakas and Katakis, 2007).

The k-nearest neighbor algorithm (KNN) is an intuitive yet effective machine learning method for solving conventional classification problems. It is generally regarded as an instance-based learning or lazy learning method because hypotheses are constructed locally and the computation is deferred until the test dataset is acquired. In a KNN-based method, an instance is usually classified by a majority vote of its k-nearest neighbor instances and assigned to the most common class among those neighbors.

Researchers have tried to extend the KNN concept to handle the multi-label classification problem. The Multi-Label K-Nearest Neighbors algorithm (MLKNN) is regarded as the first multi-label lazy learning algorithm (Zhang and Zhou, 2007). It is an extension of the KNN algorithm based on the maximum-a-posteriori principle. Subsequently, Cheng and Hüllermeier (2009) developed the Instance-Based Logistic Regression (IBLR) algorithm, which is a combination of instance-based learning and logistic regression techniques (Cheng and Hüllermeier, 2009). It includes the statistics of the k-nearest neighbors as features in logistic regression. Both IBLR and MLKNN are considered state-of-the-art multi-label classification algorithms that exploit instance-based learning, and can sometimes outperform state-of-the-art model-based approaches (Zhang and Zhou, 2007; Cheng and Hüllermeier, 2009). In this paper, we propose another kind of KNN-based learning algorithm for multi-label classification. Our objective is to exploit the dependency among labels by incorporating a ranking model into the selection process of trustable neighbors.

1.2. Motivation

We began by conducting some experiments to investigate how well a KNN-based algorithm can achieve in solving a classification problem. In these experiments, we used the Hamming loss between the label set and the gold standard as the evaluation metric. Hamming loss calculates the symmetric difference between the test instance label set and the classifier output label set: the lower the value, the better the classification performance.

Applying instance-based learning to such problems is based on the assumption that instances with similar features should have similar labels. Given that assumption, an intuitive yet simple way for a KNN approach to predict multi-labels for an instance is to assume that the labels of an instance are identical to those of its neighbors with the most similar features. We call the method the *feature-KNN approach* since the neighbors are found by using the feature set (i.e., we ignore the label set). We conduct experiments on six datasets to assess the effectiveness of such approach. The results are shown in the first row of Table 1. To improve the performance of the KNN algorithm, Dudani (1976) proposed using a weighted voting strategy to weight the votes of the top-k closest neighbors based on the closeness of their features to those of the instance to be predicted (Dudani, 1976). If a neighbor is closer to the instance to be predicted, it should be associated with a higher weight. We also performed experiments on this approach by identifying the top 10 neighbors of an instance and using weighted votes to determine the outputs, as shown in Table 1. We found that the feature-KNN approach performs significantly worse than the two state-of-the-art approaches (i.e. MLKNN and IBLR). The performance of the weighted version of feature-KNN is competitive with that of MLKNN, but it is still not as good as IBLR.

	yeast	scene	emotions	audio	genbase	medical
feature-KNN ($k = 1$)	0.2439	0.1108	0.2333	0.1143	0.0013	0.0197
feature-KNN ($k = 10$, weighted)	0.1945	0.0896	0.1879	0.0864	0.0028	0.0166
MLKNN ($k = 10$)	0.1944	0.0857	0.2615	0.0896	0.0046	0.0155
IBLR ($k = 10$)	0.1935	0.0839	0.1860	0.0840	0.0021	0.0187
Oracle-1 ($k = 10$)	0.0804	0.0156	0.0468	0.0603	0.0007	0.0054
Oracle-5 ($k = 10$)	0.1260	0.0443	0.1030	0.0678	0.0018	0.0104

Table 1: Hamming loss of 6 different methods on 6 different datasets. (10-fold cross-validation, 5-repeat).

The above results hint that not all neighbors are *trustable* for label prediction. Trustable neighbors form a subset of feature neighbors whose labels are similar to those of the gold standard. That is, the trustable neighbors of an instance are the ones that possess the most similar labels to those of the instance, instead of the most similar features. We call such neighbors *trustable neighbors*, in contrast to the feature-neighbors described above. Obviously, trustable neighbors are more useful than feature-neighbors for KNN-based multi-label classification because it is the label sets that we true care. To gain a better understanding of how much improvement trustable neighbors can provide, we conducted the following experiment. Here we assume there is an oracle that can reveal which neighbor among the k-nearest neighbors has the closest labels to the true label set of the instance to be predicted (i.e. we want to know which neighbor is the most trustable). We then label x using the labels of the most trustable instance. This method improves the performance dramatically, as shown by the results in Table 1 (i.e., Oracle-1). Usually, it reduces the Hamming loss to at least one-third from the state-of-the-art. We argue that this is the optimal performance that any KNN-based approach can achieve since the oracle has already identified the most trustable-neighbor based on the label information. In another similar experiment, we used majority voting on the top-5 trustable-neighbors (called Oracle-5), assuming they can be revealed as well. Although the results are not as good as those of the Oracle-1 method, they are still significantly better than the results of the state-of-the-art methods. The above experiments demonstrate an interesting finding, namely, being able to predict which instances are trustable from the training data is extremely useful for solving multi-label classification problems.

Unfortunately, in real-life situations, there is no such oracle to identify the top trustable neighbors because the label of each instance to be predicted is unknown. Therefore, to solve the multi-label classification problem, we propose to build a ranking model to learn what the oracle is supposed to learn. That is, we transform a multi-label classification problem into a ranking problem in order to predict the top trustable neighbors from the training data. Then, we integrate the labels of those neighbors to generate the final answers. Since the dependency between the labels is already embedded in each instance, our approach considers the dependency implicitly. The experiment results show that the approach outperforms state-of-the-art instance-based methods.

1.3. Problem Definition and Proposed Solution

In this paper, we propose a ranking-based KNN approach for multi-label classification. Given an unknown test instance x, the approach determines the final label set of the instance, as shown in Figure 1. First, similar to other KNN-based methods, we identify the k-nearest neighbors of x. Then the selected neighbors are re-ranked by a ranking model trained on their trustiness (i.e., how close their label sets are to the true label set). That is, the ranking model tries to promote the neighbors whose label set is the closest to the true label set of x. After deriving the re-ranked neighbors, we use a weighted voting strategy to produce the final prediction. The weights are learned by an optimization method that minimizes a given loss function. In the experiments, we use the Hamming loss as the performance metric. We discuss the ranking model and the weights in Section 3. It is worth noting that our strategy can be regarded as a ”complicated similarity function” that tries to learn the oracle distance (or label distance). It can be incorporated with any distance metric to provide some degree of improvement.

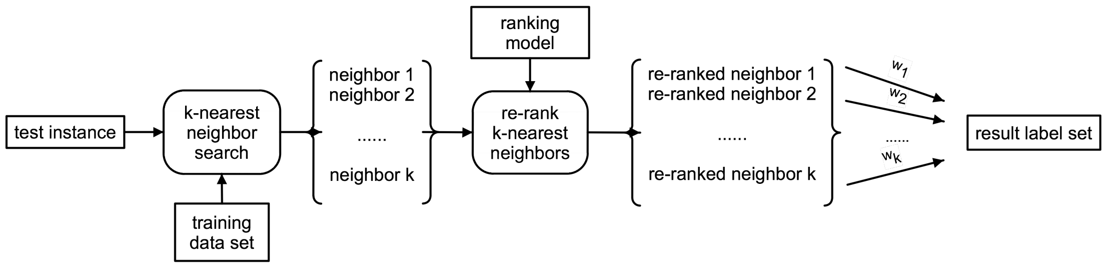

Figure 1: Testing overview

1.4. Contribution

In this paper, we propose a novel KNN-based approach to solve the multi-label classification problem. We transform a multi-label classification problem to a k-nearest neighbors ranking problem. In this way, the relations between labels can be considered. According to the experiment results, our method is competitive with IBLR, which could be considered a state-of-the-art multi-label classification algorithm via instance-based learning approach.

1.5. Paper Organization

The rest of this paper is organized as follows. The related work is in Chapter 2. Our proposed method is described in Chapter 3. The experiment results are presented in Chapter 4. Finally, the conclusion and future work are summarized in Chapter 5.

2. Related Work

Before introducing related works, we formally define the multi-label classification problem: Let X denote the domain of an instance and let $L = \{\lambda_1, ..., \lambda_m\}$ denote the complete set of labels. Assume we are given a multi-label training data set $T = \{(x_1, Y(x_1)), ..., (x_u, Y(x_u))\}$, whose instances are drawn identically and independently from an unknown distribution D. Each instance $x \in X$ is associated with a label set $Y(x) \subseteq L$. The goal of multi-label classification is to train a classifier $h : X \rightarrow 2^L$ that maps a feature vector to a set of labels, while optimizing some specific evaluation metrics.

Existing multi-label classification algorithms can be divided into two categories: Problem Transformation and Algorithm Adaptation (Tsoumakas and Katakis, 2007). The former decomposes a multi-label classification task into one or more single-label classification tasks. Therefore, existing single-label classification algorithms can be applied to problems directly. Problem transformation approaches, such as Binary Relevance and Label Power-set, are widely used in classification tasks. Algorithm Adaptation tries to modify specific algorithms to handle multi-label data directly. MLKNN, IBLR are well-known Algorithm Adaptation approaches. In this paper, we only discuss non-instance-based approaches briefly, as we focus on two well known instance-based approaches, namely, MLKNN and IBLR.

2.1. MLKNN

MLKNN, the first multi-label lazy learning approach, is based on the traditional KNN algorithm (Zhang and Zhou, 2007). The rationale for the approach is that an instance's labels depend on the number of neighbors that possess identical labels. Given a to-be-labeled instance x with an unknown label set $Y(x) \subseteq L$, MLKNN first identifies the k nearest neighbors in the training data and counts the number of neighbors belonging to each class (i.e. a random variable Z). Then the maximum a posteriori principle is used to determine the label set for the test instance. The posterior probability of $\lambda_i \in L$ is given by

$$P(\lambda_i \in Y(x)|Z = z) = \frac{P(Z = z|\lambda_i \in Y(x)) \cdot P(\lambda_i \in Y(x))}{P(Z = z)}. \tag{1}$$

Then, for each label $\lambda_i \in L$, the algorithm builds a classifier h_i using the rule

$$h_i(x) = \begin{cases} 1 & P(\lambda_i \in Y(x)|Z = z) > P(\lambda_i \notin Y(x)|Z = z) \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

$h_i(x) = 1$ means λ_i is relevant to x, while 0 means it is not relevant. The prior and likelihood probabilities in Equation 1 are estimated from the training data set in advance. The reported experiment results show that MLKNN performed well on several real-world data sets (Zhang and Zhou, 2007). However, MLKNN is actually a binary relevance learner because it learns a single classifier h_i for each label independently. In other words, it does not consider the correlations between labels. The algorithm is often criticized because of this drawback.

2.2. IBLR

Instance Based Logistic Regression (IBLR) is a novel approach that combines instance-based learning and logistic regression (Cheng and Hüllermeier, 2009). Under this approach, the label statistics of neighboring instances are regarded as features by the logistic regression classifier. IBLR defines the posterior probability of $\lambda_i \in L$ as follows:

$$\pi_0^{(i)} = P(\lambda_i \in Y(x)|N_k(x)), \tag{3}$$

where $N_k(x)$ denotes the k-nearest neighbors of x. For each label $\lambda_i \in L$, IBLR trains a logistic regression classifier h_i derived from the model

$$log\left(\frac{\pi_0^{(i)}}{1-\pi_0^{(i)}}\right) = \omega_0^{(i)} + \sum_{j=1}^{m} \alpha_j^{(i)} \cdot \omega_{+j}^{(i)}(x) \tag{4}$$

where $\omega_{+j}^{(i)}(x)$ is defined by

$$\omega_{+j}^{(i)}(x) = \sum_{\widetilde{x} \in N_k(x)} y_j(\widetilde{x}), \tag{5}$$

which is a summary of the presence of the j-th label λ_j in $N_k(x)$. Here, $y_j(\widetilde{x}) = 1$ if λ_j is relevant to $\widetilde{x}$ and 0 otherwise; and $\omega_0^{(i)}$ is a bias term.

Obviously, this approach considers the correlation between labels. According the reported experiment results, IBLR's performance on several real-world problems compares favorably with that of some state-of-the-art multi-label model-based classification algorithms (Cheng and Hüllermeier, 2009).

3. Methodology

As mentioned earlier, we propose a ranking-based KNN framework for multi-label classification. Algorithm 1 shows the steps of our decision-making strategy for test instances. Given a test instance x, we begin by finding its k-nearest neighbors. Let $\{N_k(x,j)|j = 1,...,k\}$ be the k-nearest neighbors of the instance x. $N_k(x,j)$ is the j-th nearest feature-neighbor of the instance x. Then, we exploit a global ranking model to re-rank the k-nearest neighbors. Let $\{N'_k(x,j)|j = 1,...,k\}$ be the re-ranked neighbors of the instance x. $N'_k(x,j)$ is the j-th trustable neighbor of the instance x, decided by the ranking model. Finally, we apply a weighted voting strategy to obtain the final prediction. In the following sections, we will discuss the ranking model used to re-rank the k-nearest neighbors based on their trustiness, and explain how to determine the weights for the weighted voting strategy.

Algorithm 1 $Predict(x, k, M, T, w)$

```
Input:
    $x$: The test instance
    $k$: The number of neighbors
    $M$: The ranking model
    $T$: The training data set $\{(x_1, Y(x_1)), ..., (x_u, Y(x_u))\}$
    $w$: The weight scores for re-ranked $k$-nearest neighbors
Output:
    $Y(x)$: The result label set
1: $\{N_k(x,j)|j = 1,...,k\} = KnnSearch(x, k, T)$
2: $\{N'_k(x,j)|j = 1,...,k\} = ReRank(x, k, \{N_k(x,j)|j = 1,...,k\}, M)$
3: $Y(x) = WeightedVoting(\{N'_k(x,j)|j = 1,...,k\}, w)$
```

3.1. Trust-based Ranking Model

We train a ranking model to determine which neighbor's label set tends to be closer to the true label set. The training process is shown in Figure 2, and Algorithm 2 details the training pseudo code. Let x denote the instance's feature vector, and let $Y(x)$ denote the

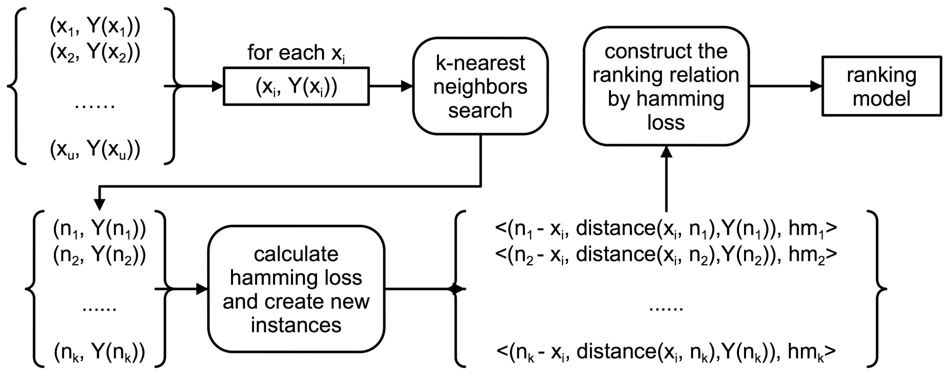
(x1, Y(x1))
(x2, Y(x2))
......
(xu, Y(xu))
for each xi
(xi, Y(xi))
k-nearest neighbors search
construct the ranking relation by hamming loss
ranking model
(n1, Y(n1))
(n2, Y(n2))
......
(nk, Y(nk))
calculate hamming loss and create new instances
<(n1 - xi, distance(xi, n1),Y(n1)), hm1>
<(n2 - xi, distance(xi, n2),Y(n2)), hm2>
......
<(nk - xi, distance(xi, nk),Y(nk)), hmk>

Figure 2: The training process building the ranking model

label set associated with x. First, for each instance x_i, we identify its k-nearest neighbors (line 2). Then, for the j-th nearest neighbor $N_k(x_i, j)$, we create a new instance q_j by using the features related to $N_k(x_i, j)$ and x_i (line 6). The new instance q_j contains the following features:

- The difference between each feature value of $N_k(x_i, j)$ and x_i (size $= dim(X)$)
- The Euclidean distance between $N_k(x_i, j)$ and x_i (size $= 1$)
- The cosine distance between $N_k(x_i, j)$ and x_i (size $= 1$)
- The label set of $N_k(x_i, j)$ (size $= |L|$)

Note that $dim(X)$ is the dimension of X. Hence, using the training instance x_i and its k-nearest neighbors, we create k instances and form the set $Q_i = \{q_1, q_2, ..., q_k\}$. Since the goal is to train a ranking model that can learn the trustiness of an instance's neighbors, we propose using a difference function (e.g., Hamming loss) between $N_k(x_i, j)$'s label set and x_i's label set to determine the quality of each new instance q_j (line 7). Based on this value, a pair-wise comparison can be made and a ranking-based classifier can be trained. In the experiment, we employ the Hamming loss because it is one of the final evaluation criteria used to evaluate the multi-label classification task. If another criterion is chosen, then it is reasonable to use it to rank new instances. Hamming loss is defined as follows:

$$HammingLoss(L_1, L_2) = \frac{1}{|L|} |L_1 \Delta L_2| . \tag{6}$$

Given two label sets L_1 and L_2, Hamming loss calculates the percentage of labels that are not consistent. The lower Hamming loss, the higher will be the rank assigned to a new instance q. In line 10, we use Hamming loss to define the ranking relation R_i over $Q_i \times Q_i$

$$q_a \leq_{R_i} q_b \text{ for all pairs } (q_a, q_b) \in Q_i \times Q_i, \text{ with } hm_a \leq hm_b, \tag{7}$$

where $q_a \leq_{R_i} q_b$ means that the rank of q_a is higher than the one of q_b.

Algorithm 2 *BuildKnnRankingModel*(k, T)

```
Input:
    $k$: The number of neighbors
    $T$: The training data set $\{(x_1, Y(x_1)), ..., (x_u, Y(x_u))\}$
Output:
    $M$: The ranking model
 1: for $i = 1$ to $n$ do
 2:     $\{N_k(x_i, j) | j = 1, ..., k\} = KnnSearch(x_i, k, T)$
 3:     $Q_i = \{\}$
 4:     for $j = 1$ to $k$ do
 5:         $\widetilde{x} = N_k(x_i, j)$
 6:         $q_j = (\widetilde{x} - x_i, EuclideanDistance(\widetilde{x}, x_i), CosineDistance(\widetilde{x}, x_i), Y(\widetilde{x}))$
 7:         $hm_j = HammingLoss(Y(\widetilde{x}), Y(x_i))$
 8:         add $q_j$ to $Q_i$
 9:     end for
10:     Construct the ranking relation $R_i$ over $Q_i \times Q_i$
11: end for
12: $M = BuildRankingModel(\{R_i | i = 1, ..., n\})$
```

Finally, we exploit all the relations to train a global ranking model (line 12). From Figure 2, one would think that every instance is used to train a ranking model, however, actually only one global ranking model would be trained in the training process. Note that in this paper we propose a framework to solve the multi-label classification problem. Any off-the-shelf ranking algorithms can be used in line 12. In our experiment, we choose Ranking SVM as the ranking algorithm (Joachims, 2002).

In the decision making (or testing) phase, when an un-labeled instance x is found, the algorithm combines the instance's k-nearest neighbors to create k new instances in the same way. Then the instances are sent to the trained ranking model for re-ranking. The ranking outputted by the ranking model can represents the ranking of corresponding neighboring instances. According to the ranking, we re-rank the original k-nearest neighbors $\{N_k(x, j) | j = 1, ..., k\}$ and get re-ranked neighbors $\{N'_k(x, j) | j = 1, ..., k\}$. A weighted voting process is then applied to the labels of the re-ranked neighbors to obtain the final predictions.

3.2. Determining Voting Weights

In this section, we explain how the weights for voting can be generated. Given the ranked list of label sets obtained in the previous stage, each neighbor is assigned a weight score. Then, the weighted voting strategy aggregates the information from different label sets. We model the task as an optimization problem in which the objective is to minimize certain loss function (i.e. the Hamming loss in our experiment).

Let $(w_1, ..., w_k)$ denote weight scores of the re-ranked neighbors $\{N'_k(x, j) | j = 1, ..., k\}$. For each label $\lambda_i \in L$, it is possible to produce an accumulated score as the weighted sum of k scores from each re-ranked neighbor

$$f_i(x) = \frac{\sum_{j=1}^{k} w_j \cdot y_i(N'_k(x,j))}{\sum_{j=1}^{k} w_j}. \tag{8}$$

The denominator is regarded as the normalization factor. We predict whether the test instance x should be associated with λ_i by the rule

$$h_i(x) = \begin{cases} 1 & f_i(x) \geq 0.5 \\ 0 & f_i(x) < 0.5. \end{cases} \tag{9}$$

The final prediction of the label set of x is defined as

$$H(x) = \{\lambda_i | h_i(x) = 1\}. \tag{10}$$

Equation 8, 9, and 10 essentially capture the idea that if the majority of the neighbors' votes are positive, the instance should be regarded as positive; otherwise, it should be considered negative.

To determine the optimal weights, the optimization problem is formulated as follows:

$$\underset{w_1,\ldots,w_k}{\text{minimize}} \quad \sum_{x \in T'} HammingLoss(Y(x), H(x)) \tag{11}$$

$$\text{subject to} \quad 1 \geq w_1 \geq w_2 \geq \ldots \geq w_k \geq 0. \tag{12}$$

For the fairness, we additionally use the held-out set T' to handle the optimization task, instead of the original training set used to train the ranking model. Note that in the KNN search process, neighboring instances are still searched in the training set. Some constraints are added to this objective function. First, we consider all weights are between 0 and 1. Second, the neighbor with a higher rank should be associated with a higher weight. Since Equation 9 is not a continuous function, finding an analytical solution for Equation 11 is a non-trival task; therefore, we exploit the direct search techniques to solve this problem.

Direct search methods have been known since the 1950s (Kolda et al., 2003). Although they were developed heuristically, they remain an effective option, especially for difficult optimization problems (Lewis and Torczon, 1999; Audet, 2004). Generalized pattern search, the algorithm adopted in this paper, is one of the most popular direct search methods. Studies in the literature report that when the objective function is not continuous, the generalized pattern search approach usually performs well (Audet, 2004).

Given an objective function $f : \mathbb{R}^r \rightarrow \mathbb{R}$, the generalized pattern search algorithm first defines a pattern set V, the set of unit vectors, used to determine the points to examine at each iteration. To find a point that improves the objective function, at each iteration, the algorithm searches a set of points, called a mesh. It then generates a set of vectors by multiplying each pattern vector $v_i \in V$ by a scalar Δ, which is called the mesh size. From the derived vectors and the current point, we can obtain new points, called mesh points, which may yield better solutions.

Algorithm 3 *GeneralizedPatternSearch*$(f, x_0, \Delta_0, \Delta_{tol})$

```
Input:
    $f$: The objective function
    $x_0$: The initial guess point
    $\Delta_0$: The initial mesh size
    $\Delta_{tol}$: The tolerance of the mesh size
Output:
    $x_{opt}$: The solution for the optimization problem
 1: Let $V$ be the set of coordinate directions $\{\pm e_i | i = 1, ..., r\}$, where $e_i$ is the $i$-th unit
    vector in $\mathbb{R}^r$.
 2: for $i = 0, 1, 2, ...$ do
 3:     if there exists $v_i \in V$ such that $f(x_i + \Delta_i v_i) < f(x_i)$ then
 4:         $x_{i+1} = x_i + \Delta_i v_i$
 5:         $\Delta_{i+1} = \Delta_i$
 6:     else
 7:         $x_{i+1} = x_i$
 8:         $\Delta_{i+1} = \frac{1}{2}\Delta_i$
 9:         if $\Delta_{i+1} < \Delta_{tol}$ then
10:             $x_{opt} = x_{i+1}$
11:             return $x_{opt}$
12:         end if
13:     end if
14: end for
```

After determining the mesh points, the algorithm performs the polling step. Specifically, it polls current mesh points by computing their objective function values. When it finds a mesh point whose objective function value is less than that of the current point, it stops the polling process. This poll is considered a success and the newly discovered point becomes the current point at the next iteration. If the algorithm fails to find a mesh point to improve the objective function, the poll is deemed a failure. The current point will not be changed at the next iteration, but the mesh size will be reduced by half. The mesh point search step and the polling step are repeated until the mesh size is less than a threshold. The pseudo code of generalized pattern search algorithm is shown in Algorithm 3. Note that different initial points yield different solutions, so random restart is needed to perform a generalized pattern search.

4. Experiment

We compare the proposed methods with other KNN-based multi-label classification algorithms on several data sets. In the following sections, we describe the experiment setup including the data sets, the compared algorithms, and the evaluation criteria, and then discuss the experiment results.

4.1. Data Sets

We conduct experiments on six commonly used data sets belonging to different domains. The statistics of the data sets are shown in Table 2. Note that the cardinality is defined as the average number of labels per instance, while the density is the same number divided by $|L|$. The "distinct" column shows the number of distinct label sets. We describe each dataset below.

	instances	features	labels	cardinality	density	distinct
yeast	2417	103	14	4.237	0.303	198
scene	2407	294	6	1.074	0.179	15
emotions	593	72	6	1.869	0.311	27
audio	2472	177	45	4.119	0.092	1553
genbase	662	1186	27	1.252	0.046	32
medical	978	1449	45	1.245	0.028	94

Table 2: Statistics of the multi-label data sets

The *yeast* data set is used to predict the gene functional classes of the Yeast Saccharomyces cerevisiae (Elisseeff and Weston, 2002). The features of the data set correspond to the micro-array expression data and phylogenetic profiles of the genes. The *scene* data set is used to predict the semantics of the scenes in the pictures (Boutell et al., 2004). Each picture is associated with a set of six classes: beach, sunset, foliage, field, mountain, and urban. The features contain spatial color moments in the LUV space. It has been shown that color and spatial information is effective in distinguishing between outdoor scenes. The *emotions* data set contains 593 instances, each of which is a music clip that is described by 72 extracted features (Trohidis et al., 2008). The *audio* data set was obtained from the MajorMinor's music labeling game (Lo et al., 2010). In the game, players can listen to 10-second music clips and tag them. The features of the data set are divided into five types: dynamics, rhythm, timbre, pitch, and tonality. The *medical* data set is used to predict the categories of clinical free text. The data was obtained from the Medical NLP challenge, sponsored by the Computational Medical Center.

4.2. Evaluation Metrics

A number of metrics have been proposed for evaluating the performance of multi-label classification algorithms (Tsoumakas and Katakis, 2007). For a classifier H, let $H(x)$ denote the label set of instance x predicted by the classifier, and let $Y(x)$ be the answer.

In most cases, the classifier produces a scoring function $f(x, \lambda)$ denoting the score assigned to label λ for the instance x, and $rank_f(x, \lambda)$ denoting the position of the label λ for the instance x in the ordering induced by f. We use four evaluation metrics for multi-label classification: the Hamming loss, one error, average precision and root-mean-square-error. The definitions of the metrics are described as follows:

Hamming loss: as mentioned previously, the Hamming loss calculates the percentage of labels whose relevance is predicted incorrectly. The value is between 0 and 1.

$$HammingLoss(h) = \frac{1}{|T|} \sum_{i=1}^{|T|} \frac{1}{|L|} \left| H(x_i) \Delta Y(x_i) \right| , \tag{13}$$

where Δ is the symmetric difference between two sets.

One error: the metric computes how many times the top-ranked label is not relevant.

$$OneError(f) = \frac{1}{|T|} \sum_{i=1}^{|T|} I\left(\mathrm{argmax}_{\lambda \in L} f(x_i, \lambda) \notin Y(x_i)\right), \tag{14}$$

where $I(\pi)$ equals 1 if π holds and 0 otherwise.

Average precision: this is the average of the per-instance average precision over all test instances. The value is between 0 and 1. The higher value, the better performance:

$$AvgPrec(f) = \frac{1}{|T|} \sum_{i=1}^{|T|} \left(\frac{1}{|Y(x_i)|} \sum_{\lambda \in Y(x_i)} \frac{|\lambda' | rank_f(x_i, \lambda') \leq rank_f(x_i, \lambda), \lambda' \in Y(x_i)|}{rank_f(x_i, \lambda)} \right). \tag{15}$$

Finally, we use root mean square error (RMSE) to evaluate the average performance on each label:

$$RMSE(f) = \sqrt{\frac{\sum_{j=1}^{|L|} \sum_{i=1}^{|T|} (f_j(x_i) - y_j(x_i))^2}{|T|\,|L|}}, \tag{16}$$

where $f_j(x)$, described in Equation 8, is the accumulated score of λ_j.

4.3. Experiment Setup and Results

We compare the proposed method's performance with that of two other multi-label instance-based learning algorithms: MLKNN and IBLR, which are regarded as state-of-the-art approaches. Both algorithms are parameterized by the size of the neighborhood k. Following their experiment setup (Zhang and Zhou, 2007; Cheng and Hüllermeier, 2009), we set the value of $k = 10$, and use the Euclidean metric as the distance function. We choose two baseline methods by using binary relevance learning with two different classifiers: logistic regression (BR-LR), and KNN where k=10 (BR-KNN). Another multi-label algorithm, RAKEL, is also used in our experiment (Tsoumakas et al., 2011). Although RAKEL is not an instance-based learning method, it is an effective multi-label algorithm, according to a recent study (Madjarov et al., 2012). The base classifier of RAKEL is L2-loss linear SVM, and the parameters are determined by cross-validation. Note that all the algorithms are implemented in the MULAN package (Tsoumakas and Katakis, 2007), and the logistic regression algorithm in IBLR and BR-LR is implemented in the Weka package. Since LP performance is not better than BR performance on several datasets (Zhang and Zhou, 2007; Cheng and Hüllermeier, 2009), we only use two BR algorithms as our baseline. In our proposed framework, we have tried several ranking algorithms: Ranking SVM (RS) (Joachims, 2002), Passive-Aggressive Perceptron (PA) (Crammer et al., 2006), and Perceptron with Margins (PM) (Sculley, 2009). For the generalized pattern search algorithm, we randomly generate 300 points for the initial points, and adopt the values $\Delta_0 = 1$ and $\Delta_{tol} = 10^{-6}$.

We perform 10-fold 5-repeat cross-validation on the above data sets. A separate ranking model was learned on each training set of the cross-validation, however, we pick out 20% instances of the training set and use them as the held-out set to perform the optimization task described earlier. All methods use exactly the same training and testing sets, hence all KNN-based methods use the same set of neighbors. In this way, given the same set of neighbors, we can distinguish which KNN-based method can obtain better results. The experiment results are shown in Table 3. The numbers in parentheses represent the rank of the algorithms among the compared algorithms. In general, IBLR outperforms the BR-LR algorithm significantly, which implies that exploiting the frequency of neighbors' label features is effective. Meanwhile, MLKNN and BR-KNN do not perform as well as IBLR. This implies that exploiting neighbor information alone is not sufficient, and the correlations between labels should also be considered. RAKEL has the best performance on the data sets *audio* and *medical*, however, it does not perform well on the other data sets. In our proposed framework, choosing a suitable ranking algorithm is important. Ranking SVM outperforms Passive-Aggressive Perceptron and Perceptron with Margins.

Overall, our proposed framework with Ranking SVM (RS) outperform the compared methods on each measure. The average ranking of our method on 6 datasets using four different metrics is (1.67, 2.17, 1.67, 2.33), while the second best algorithm, IBLR, only achieves (3.00, 2.83, 2.83, 2.67). It is worth noting that, although our training of the ranked-based model and weight determination focus on minimizing the Hamming loss, the proposed methods not only perform the best (average rank=1.33) on the Hamming loss but also perform very well in terms of the other three metrics. We performed the significance test to check whether our algorithm is significantly better than other algorithms. We conducted the t-test between RS and IBLR (2nd best algorithm in general). Out of the 24 experiments, 15 of them has p-value smaller than or equal to 0.05. Out of these 15 cases, our algorithm wins in 9 cases.

Table 3: Experiment Results

	BR-KNN	BR-LR	IBLR	MLKNN	RAKEL	PM	PA	RS	P-value
Hamming Loss									
yeast	0.198 (6)	0.206 (8)	0.194 (2)	0.194 (3)	0.202 (7)	0.196 (5)	0.195 (4)	0.191 (1)	0.05
scene	0.093 (7)	0.140 (8)	0.084 (2)	0.086 (3)	0.091 (5)	0.092 (6)	0.089 (4)	0.082 (1)	0.11
emotions	0.194 (3)	0.217 (7)	0.186 (1)	0.261 (8)	0.212 (6)	0.195 (5)	0.194 (4)	0.187 (2)	0.45
audio	0.089 (6)	0.102 (8)	0.084 (2)	0.090 (7)	0.079 (1)	0.087 (5)	0.086 (4)	0.086 (3)	0.00
genbase	0.003 (5)	0.002 (3)	0.002 (4)	0.005 (7)	0.007 (8)	0.003 (6)	0.001 (2)	0.001 (1)	0.00
medical	0.017 (5)	0.026 (8)	0.019 (7)	0.015 (4)	0.010 (1)	0.017 (6)	0.013 (3)	0.012 (2)	0.00
AvgRank	5.33	7.00	3.00	5.33	4.67	5.50	3.50	1.67	
One Error									
yeast	0.233 (4)	0.242 (7)	0.228 (1)	0.230 (2)	0.253 (8)	0.239 (6)	0.236 (5)	0.233 (3)	0.17
scene	0.258 (7)	0.367 (8)	0.224 (2)	0.224 (3)	0.239 (5)	0.248 (6)	0.236 (4)	0.220 (1)	0.22
emotions	0.262 (4)	0.280 (6)	0.254 (1)	0.380 (8)	0.333 (7)	0.271 (5)	0.261 (2)	0.262 (3)	0.25
audio	0.460 (5)	0.643 (8)	0.428 (2)	0.550 (7)	0.369 (1)	0.462 (6)	0.459 (4)	0.458 (3)	0.00
genbase	0.006 (3)	0.014 (8)	0.009 (5)	0.010 (6)	0.012 (7)	0.008 (4)	0.005 (2)	0.005 (1)	0.03
medical	0.289 (5)	0.324 (8)	0.307 (6)	0.245 (4)	0.136 (1)	0.307 (7)	0.195 (3)	0.174 (2)	0.00
AvgRank	4.67	7.50	2.83	5.00	4.83	5.67	3.33	2.17	
Average Precision									
yeast	0.760 (6)	0.755 (7)	0.768 (1)	0.764 (3)	0.713 (8)	0.761 (5)	0.763 (4)	0.767 (2)	0.41
scene	0.846 (7)	0.767 (8)	0.867 (2)	0.866 (3)	0.848 (6)	0.851 (5)	0.858 (4)	0.867 (1)	0.46
emotions	0.802 (3)	0.794 (6)	0.814 (1)	0.715 (8)	0.753 (7)	0.799 (5)	0.802 (4)	0.804 (2)	0.06
audio	0.528 (4)	0.489 (7)	0.557 (1)	0.448 (8)	0.519 (6)	0.526 (5)	0.530 (3)	0.531 (2)	0.00
genbase	0.992 (3)	0.988 (7)	0.989 (5)	0.988 (6)	0.980 (8)	0.991 (4)	0.992 (2)	0.992 (1)	0.03
medical	0.776 (5)	0.748 (8)	0.759 (7)	0.811 (4)	0.871 (1)	0.766 (6)	0.839 (3)	0.849 (2)	0.00
AvgRank	4.67	7.17	2.83	5.33	6.00	5.00	3.33	1.67	
RMSE									
yeast	0.376 (3)	0.385 (7)	0.372 (1)	0.374 (2)	0.426 (8)	0.378 (6)	0.376 (5)	0.376 (4)	0.00
scene	0.261 (5)	0.344 (8)	0.249 (2)	0.251 (3)	0.276 (7)	0.262 (6)	0.257 (4)	0.248 (1)	0.46
emotions	0.369 (2)	0.398 (6)	0.364 (1)	0.420 (7)	0.424 (8)	0.373 (5)	0.371 (4)	0.370 (3)	0.02
audio	0.258 (4)	0.286 (8)	0.252 (1)	0.266 (7)	0.262 (6)	0.260 (5)	0.258 (3)	0.257 (2)	0.00
genbase	0.049 (5)	0.041 (3)	0.041 (4)	0.054 (7)	0.073 (8)	0.051 (6)	0.031 (1)	0.031 (2)	0.00
medical	0.118 (5)	0.157 (8)	0.126 (7)	0.112 (4)	0.090 (1)	0.119 (6)	0.103 (3)	0.100 (2)	0.00
AvgRank	4.00	6.67	2.67	5.00	6.33	5.67	3.33	2.33	

4.4. Ranking vs. Weighted Voting

Since the proposed solution is comprised of two components, KNN re-ranking and voting weight optimization, we want to determine each component's contribution to the overall performance. We compare six types of set-up, as shown in Table 4. The results are shown in Table 5.

	Re-ranked KNN	KNN
Generalized Pattern Search	RK+GPS	K+GPS
Linear Weighting	RK+LW	K+LW
Logistic Regression	RK+LR	K+LR

Table 4: Six experiment setups for different component combinations

We compare our ranking-based method with the original KNN approach (without re-ranking). For weight determination, besides the proposed method, we tried assigning monotonically decreasing weights manually (i.e. the top-ranked neighbor was assigned weight k, and the lowest-ranked neighbor was assigned weight 1).

We also use another approach to determine the weights. In this approach, we concatenate all neighbors' labels as the features of training instances. Then for each label, we train a classifier such that the probability outputted by the classifier can be used as the weight score. Logistic regression is used in this method. The results demonstrate that either re-ranking and weight-voting contributes to the overall performance in its own aspect.

	RK+GPS	RK+LW	RK+LR	K+GPS	K+LW	K+LR
yeast	**0.1910**	0.1912	0.2065	0.1949	0.1960	0.2077
scene	**0.0817**	0.0846	0.0835	0.0902	0.0909	0.0869
emotions	0.1866	**0.1860**	0.2153	0.1918	0.1915	0.2157
audio	**0.0857**	0.0858	0.1541	0.0864	0.0872	0.1526
genbase	**0.0011**	0.0022	0.0027	0.0014	0.0023	0.0041
medical	**0.0117**	0.0143	0.0417	0.0157	0.0162	0.0521

Table 5: Hamming Loss results of different combination approaches

5. Conclusion

We have proposed a k-nearest-neighbor-based ranking approach to solve the multi-label classification problem. The major contributions of this paper are as follows.

1. We observe an interesting phenomenon from the data, namely, it is possible to improve the accuracy of state-of-the-art multi-label classification approaches if the trustable neighbors of instances can be identified. Our finding on the surprisingly good upper-bound performance (i.e. oracle-1 and oracle-5) also confirms the legitimacy of exploiting an instance-based learning method for multi-label classification. It suggests that the KNN-based approaches could enjoy more benefits to multi-label tasks than to single-label task.

2. Based on the above finding, we present a method that combines weighted KNN and ranked learning methods to solve the multi-label classification problem. The experiment results demonstrate the efficacy of our approach.

References

Charles Audet. Convergence results for generalized pattern search algorithms are tight. *Optimization and Engineering*, 5:101–122, 2004. ISSN 1389-4420.

Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757 – 1771, 2004. ISSN 0031-3203.

Weiwei Cheng and Eyke Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76:211–225, September 2009. ISSN 0885-6125.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585, December 2006. ISSN 1532-4435.

Sahibsingh A. Dudani. The distance-weighted k-nearest-neighbor rule. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-6(4):325 –327, april 1976. ISSN 0018-9472.

André Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14 (NIPS-01)*, pages 681–687, 2002.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X.

Tamara G. Kolda, Robert Michael Lewis, and Virginia Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45: 385–482, 2003.

Robert Michael Lewis and Virginia Torczon. Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization*, 9:1082–1099, April 1999. ISSN 1052-6234.

Hung-Yi Lo, Ju-Chiang Wang, and Hsin-Min Wang. Homogeneous segmentation and classifier ensemble for audio tag annotation and retrieval. In *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, pages 304 –309, july 2010.

Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and SašO Džeroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recogn.*, 45(9):3084–3104, September 2012. ISSN 0031-3203.

D. Sculley. Large Scale Learning to Rank. In *NIPS 2009 Workshop on Advances in Ranking*, December 2009.

Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, and Ioannis P. Vlahavas. Multi-label classification of music into emotions. In Juan Pablo Bello, Elaine Chew, and Douglas Turnbull, editors, *ISMIR*, pages 325–330, 2008. ISBN 978-0-615-24849-3.

Grigorios Tsoumakas and Ioannis Katakis. Multi label classification: An overview. *International Journal of Data Warehouse and Mining*, 3(3):1–13, 2007.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Random k-labelsets for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079–1089, 2011.

Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40:2038–2048, July 2007. ISSN 0031-3203.