Instance-Ranking: A New Perspective to Consider the Instance Dependency for Classification

Xin Xia, Xiaohu Yang, Shanping Li, and Chao Wu

College of Computer Science and Technology, Zhejiang University
38 Zheda Road, Hangzhou, 310027, China
{xxkidd,yangxh,shan,wuchao}@zju.edu.cn

Abstract. Single-label classification refers to the task to predict an instance to be one unique label in a set of labels. Different from single-label classification, for multi-label classification, one instance is associated with one or more labels in a set of labels simultaneously. Various works have focused on the algorithms for those two types of classification. Since the ranking problem is always coexisting with the classification problem, and traditional researches mainly assume the uniform distribution for the instances, in this paper, we propose a new perspective for the ranking problem. With the assumption that the distribution for the instance is not uniform, different instances have different influences for the distribution, the Instance-Ranking algorithm is presented. With the Instance-Ranking algorithm, the famous K-nearest-neighbors (KNN) algorithm is modified to confirm the validity of our algorithm. Lastly, the Instance-Ranking algorithm is combined with the ML.KNN algorithm for multi-label classification. Experiment with different datasets show that our Instance-Ranking algorithm achieves better performance than the original state-of-art algorithm such as KNN and ML.KNN.

Keywords: Instance Ranking, KNN, ML.KNN, Multi-label Classification.

1 Introduction

Supervised learning problems are the one of the core topics in the machine learning related area. Generally speaking, in traditional supervised learning area, one instance is assigned to an unique single label λ from a set of disjoint labels $\mathcal{L}, |\mathcal{L}| > 1$ [1].However, single-label classification can't satisfy the requirements of many real-world applications. For example, a piece of DNA [2] from Yeast Gene can have the function of transcription and protein synthesis simultaneously; For text categorization [3], one news about the England Riots may have more than one categories, such as the economy, politics and ethics; For music categorization [4], a single song makes a person feel happy and exciting at the same time, while another song makes the person sad and nervous. The above examples show some common aspects, a single instance can belong to more than one label simultaneously. This phenomenon is called multi-label classification [5].

More formally, let χ denote the input space and labels. Given the multi-label training dataset $D = \{(X_i, Y_i)\}_m^{i=1}$ where $X_i \in \chi \, and \; Y_i \subseteq \mathcal{L}$. Let $\mathcal{L} = \{1,2,3 \dots |L|\}$ denote the $|L|$ finite associated labels, the goal of multi-label classification is to learn a hypothesis $h: \chi \to 2^Y$ which is used to predict the proper label set for a new instance.

Various methods have been proposed to solve the multi-label classification. Generally to say, those methods can be divided into two camps, *problem transformation method* and *algorithm adaptation method* [5]. The *problem transformation method* [6-8] transforms the multi-label classification task into a subset of multiple single-label classification tasks, usually it is based on the on binary relevance (BR) or label powerset (LP). *Algorithm adaptation method* extends specific learning algorithms in order to handle multi-label data directly. Those algorithms include lazy algorithm (ML-KNN [9] and Mr-KNN [10]), AdaBoosting.MH [3], Rank-SVM [11], BP-MLL[12] and decision trees [13].

Ranking problem is always coexisting with the classification problem. For single-label classification, the probability of the instance belong to each label in the associated label set is computed, and the instance is considered to belong to the label with the highest probability. For multi-label classification, after the probability is computed, the ranking threshold is computed; for each label of the instance, if the probability is above the threshold, then this label is added into the proper label set. Especially, for multi-label classification, one particular evaluation metrics called ranking loss is provided to evaluate average fraction of label pairs that are reversely ordered for the instance.

The traditional ranking analysis is built with the assumption the instance is uniform distributed, and each instance affects the labels distribution equally. In this paper, we propose another view to see the ranking and classification problem, which assumes that the instances are not uniformly distributed. Inspired by the PageRank which rank the web pages on the Internet, we rank the instances in the training datasets to generate the ranking scores. The ranking scores are used to describe the distribution for the instances in the datasets. The instance with higher scores will have deeper influence for the proper label sets. To validate the effective and feasibility of our Instance-Ranking algorithm, we modify the K-Nearest-Neighbors (KNN) [1] algorithm with the ranking scores (IR.KNN), and compare the IR.KNN with KNN. With the experiment showing that IR.KNN achieves a slightly better performance than KNN, we go further to apply the Instance-Ranking algorithm to multi-label classification. The ML.KNN algorithm is enhanced with our Instance-Ranking algorithm (IR.MLKNN). Experiments show that our IR.MLKNN achieves better than MLKNN.

The main contribution of this paper is summarized as follows:

- We propose a new perspective to view the instance dependency, and address a feasible instance ranking algorithm.
- We enhance the previous KNN and MLKNN algorithm with our instance ranking algorithm, and experiments show that our enhanced algorithm achieves better than the original ones.

The rest of this paper is organized as follows: the related works about PageRank, KNN and MLKNN are briefly reviewed in Section 2. In Section 3, the detail of

Instance-Ranking Algorithm is proposed. The Instance-Ranking Based KNN (IR.KNN) and the Instance-Ranking Based MLKNN (IR.MLKNN) are addressed in Section 4. In Section 5, the experiments and the results are presented. The final conclusion and future work are provided in Section 6.

2 Related Works

In this section, we briefly review the PageRank algorithm, the KNN and MLKNN algorithm. PageRank algorithm is most related to our Instance-Ranking algorithm, while the KNN and MLKNN algorithm will be enhanced later in the paper.

2.1 PageRank

The PageRank [14, 15] algorithm properly takes advantage of primitive matrix's convergence feature and the assumption of random surfer model perfectly. The basic idea of PageRank is that if page u has a link to page v, then the author of u is implicitly conferring some importance to page v. Therefore, for each page v that page u links to, the Rank of page v will be:

$$Rank_{i+1}(v) = \sum_{u \in B_v} Rank_i(u) / N_u \tag{1}$$

i is the iteration number. B_v is the set of pages that link to page v. N_u is the total number of pages u links to.

According this idea, a $n \times n$ transition matrix P is constructed (n is the number of pages). Each element P_{ij} represents the probability that random surfer move from $Page_i$ to $Page_j$.To Avoid the Rank Sink, the jump probabilities are added to dispatch rank scores of the dangling links and transfer transition matrix P to be a primitive one ($\bar{P}$). However, matrix $\bar{P}$ may not be an irreducible one as zero factors might exist in matrix $\bar{P}$. After dispatching some scores to pages not directly link to, transition matrix $\bar{\bar{P}}$ finally be transferred as an irreducible primitive matrix.

$$P \to \bar{P} \to \bar{\bar{P}} \tag{2}$$

$$\bar{\bar{P}} = \alpha\bar{P} + (1-\alpha)ee^T/N \tag{3}$$

2.2 KNN and MLKNN

K-nearest neighbor (KNN) [1] algorithm has a long history in the data mining area for single-label classification. Simply to say, KNN finds a group of nearest k instance in the training set, and bases the assignment of a label on the predominance of a particular label in its neighborhood.

ML-KNN [9] considers the labels for a new instance according to the labels of KNN , and then based on statistical information from the neighboring instance, maximum a posteriori (MAP) principle is utilized to determine the label set for the new instance.

3 Instance-Ranking Algorithm

In this section, we propose the details of the Instance-Ranking algorithm, which is an extension of our previous work [17].Unlike to the web pages which have different types of links, the instances in the datasets don't have the inherent link relationship. To rank the instances in the dataset, two jobs are required: The link model for the instances and the ranking algorithm for the instances.

3.1 Instance Ranking Graph

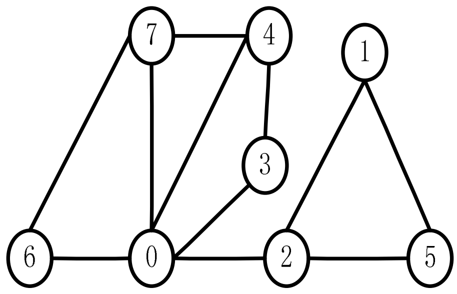
To build the instance ranking graph, we need to map the instances into the graph. For each instance X_i belongs to the dataset D $(X_i \in D)$, there is an unique vertex v_i representing the instance. The edges among the vertexes are defined as : for each instance X_i, find its K-nearest-neighbors in D, denote as $X_{i_1}, X_{i_2}, X_{i_3} \dots X_{i_k}$, and there is an edge between v_i and v_m $(m \in \{i_1, i_2, i_3 \dots i_k\})$.

Definition 1: the Instance Ranking Graph is defined as:

$$G = (V, E)$$
$$V = \{v_i | X_i \in D\} \tag{4}$$
$$E = \{(v_i, v_j) | v_i, v_j \in V, v_i \in KNN(v_j) \; or \; v_j \in KNN(v_i), i \neq j\}$$

ID	Input Space	2NN
0	2.5,3,3.5	{6,7}
1	6.5,7,8	{2,5}
2	3.5,4.5,5.5	{0,5}
3	6.5,3.5,2.5	{0,4}
4	4,1.5,2	{0,7}
5	5.5,6.5,7	{1,2}
6	1,3,2	{0,7}
7	2,2,2	{0,6}

(a)



(b)

Fig. 1. (a) an example dataset in the three-dimensional coordinate space; (b) the 2NN Instance Rank Graph with the dataset in (a)

The ***Instance Ranking Graph*** is a fully connected graph without any separate components as for each vertex v_i, as its K-Nearest-Neighbors can be always found, which avoids the existing of dangling vertex. Figure 1 (a) shows one example dataset with 8 instances in the three-dimensional coordinate space, Figure 1(b) shows the Instance Ranking Graph with 2NN. For example, the 2NN of the vertex 0 is vertex 6 and 7, so there are edges between vertex 0 and 6, vertex 0 and 7 in Figure 1(b); for the vertex 6, the 2NN is vertex 0 and 7, as there is already between vertex 0 and 6, only the edge between vertex 6 and 7 is added in Figure 1(b). With the ***Instance Ranking Graph*** built, the definition of the weights between two connected vertexes is required.

In this paper, we use the Euclid distance to describe the distance between two instances v_i and v_j(denoted as $distance(v_i, v_j)$).

Definition 2: the weight matrix W ***of the Instance Ranking Graph is:***

$$W_{ij} = \begin{cases} 0, i = j \ or \ (v_i, v_j) \notin E \\ \frac{1/distance(v_i, v_j)}{\sum_{v_p \in KNN(v_i)} 1/distance(v_i, v_p)}, (v_i, v_j) \in E \end{cases} \tag{5}$$

W is a Normalized matrix with $\sum_{j=1}^{m} W_{ij} = 1$. From the definition of weight matrix W, one can conclude that weight matrix is not a symmetrical matrix.

3.2 The Procedure of Instance Ranking

The random walk model [16] is used to rank the instances in the dataset. The general idea for instance ranking is: Visiting the instance rank graph from anyone of the vertexes v_0 . In anyone of the vertex v, it has the probability α to visit the neighbor vertex of v from the instance ranking graph, or it may also random teleport to any other vertex with the probability $(1 - \alpha)$, where α controls the priority jump to the vertex's neighbor as opposed to teleport to a random vertex in the graph. The whole process can be summarized in formula (6):

$$\boldsymbol{\pi}^{(k)T} = \boldsymbol{\pi}^{(k-1)T}(\alpha \cdot \mathbf{P} + (\mathbf{1} - \alpha) \cdot \mathbf{et}^{T}) \tag{6}$$

where $\boldsymbol{\pi}^{(k)}$ denotes the output ranking scores for the instances after k-th iteration, **P** denotes the adjacent probability matrix, α denotes the random teleport parameter (0<α<1), $\mathbf{t}$ denotes the probability of random walk to anyone of the vertexes. As the adjacent probability matrix **P** satisfies the property the sum of each row equals 1 ($\sum_{j=1}^{m} P_{ij} = 1$) , and the weight matrix W satifies $\sum_{j=1}^{m} W_{ij} = 1$. The formula (6) is rewritten as formula (7):

$$\boldsymbol{\pi}^{(k)T} = \boldsymbol{\pi}^{(k-1)T}(\alpha \cdot \mathbf{W} + (\mathbf{1} - \alpha) \cdot \mathbf{et}^{T}) \tag{7}$$

The whole algorithm of Instance Ranking is in Figure 2. The algorithm receives the parameters dataset D and the random teleport parameter **α**, after iterate formula (10) enough times, the algorithm will converge to a unique ranking scores. For different α, the algorithm will have different results. If α → 0, the iteration times are huge, and if α → 1, the Rank Matrix **RM** become much more volatile, and fluctuates noticeably for even small changes of Rank Matrix **RM**. The final ranking result for the example in Figure 1 (a) is Rank(0)=0.245, Rank(1)=0.067, Rank(2)=0.055, Rank(3)=0.019, Rank(4)=0.028, Rank(5)=0.083, Rank(6)=0.240, Rank(7)=0.264.

4 IR.KNN and IR.MLKNN

To verify the performance of our Instance-Ranking algorithm, we modify the two state-of-art algorithms for single-label and multi-label classification, KNN and

MLKNN. In this section, we present the modified KNN (IR.KNN) and MLKNN (IR.MLKNN) algorithms, and in the next section, the experiments for IR.KNN and IR.MLKNN are addressed.

```
Algorithm 1:The Instance Ranking Algorithm
Input: Dataset D, the random teleport parameter α
Output: Ranking Scores for each instance in D

Instance-Ranking (D, α)
1. Build the KNN based link graph G from D using (4);
2. Compute the weight matrix W using (5)
3. Random choose one of the rows in W as the initial ranking scores, $\boldsymbol{\pi}^{(0)} = \mathbf{W}_i^T$
4. Set $\mathbf{t} = (\frac{1}{m}.\frac{1}{m},\frac{1}{m} \ldots \frac{1}{m})^T$.
5. Compute the Rank Matrix $\mathbf{RM} = \boldsymbol{\alpha} \cdot \mathbf{W} + (\mathbf{1} - \boldsymbol{\alpha}) \cdot \mathbf{et}^T$
6. Iterate $\boldsymbol{\pi}^{(k)T} = \boldsymbol{\pi}^{(k-1)T} * \mathbf{RM}$ until $\boldsymbol{\pi}^{(k)}$ get convergent.
7. Return the final convergences $\boldsymbol{\pi}^{(\mathbf{final})}$
```

Fig. 2. The details of the Instance Ranking Algorithm

4.1 IR.KNN

The basic idea of IR.KNN is simple. Suppose current there are |L| different labels, for a new instance X_{new}, found the KNN of X_{new} (denote them as $X_{new_1}, X_{new_2}, \ldots X_{new_k}$). For each label $Label_i$, compute the probabilities:

$$P_{label_i} = \sum\nolimits_{X \in \{X_{new_1}, X_{new_2}, \ldots X_{new_k}\} \&\& X \in Label_i} Rank(X) \tag{8}$$

X_{new} is considered to belong to the label with the highest probability. Figure 3 presents the detail of IR.KNN.

4.2 IR.MLKNN

As discussed in Section 2.2, the MLKNN algorithm can be transformed into the problem of computing the prior probability $P(H_b^j)$ and the posterior ty $P\left(E_{\vec{C}_t(j)}^j \middle| H_b^j\right)$. In the MLKNN paper, the prior probability is simply count the total number of instances which belong to the particular label divide by the total number of instances. More formally, the prior probability of MLKNN algorithm is:

$$P(H_1^j) = (s + \sum_{i=1}^m y_{X_i}(j))/(s * 2 + m), \tag{9}$$

$$P(H_0^j) = 1 - P(H_1^j), j \in \mathcal{L} \tag{10}$$

where s is a smoothing parameter controlling the strength of uniform prior.

In this paper, we reconsider the prior probability $P(H_b^j)$, and we embed our Instance-Ranking algorithm to enhance the prior probability. More formally, the prior probability of IR.MLKNN is:

$$P\left(H_1^j\right) = \sum_{i=1}^{m} Rank(X_i) * y_{X_i}(j), \tag{11}$$

$$P\left(H_0^j\right) = \sum_{i=1}^{m} Rank(X_i) * (1 - y_{X_i}(j)), j \in \mathcal{L} \tag{12}$$

```
Algorithm 2:The IR.KNN Algorithm
Input: Training Dataset D ( $X_i, L_i$), the random teleport parameter $\alpha$, the new instance $X_{new}$
Output: The label of the Instance $X_{new}$

IR.KNN (D, $\alpha$, $X_{new}$)
1   Compute the Rank(X,L) ((X, L) $\in$ D) using the Instance Ranking Algorithm
2   $\{(X_{new_1}, L_{new_1}), (X_{new_2}, L_{new_2}) \ldots (X_{new_k}, L_{new_k})\} = KNN(X_{new}, D)$
3   For each Label $label_i$ in the labelset
        $P_{label_i} = \sum_{(X,L) \in \{(X_{new_1}, L_{new_1}), (X_{new_2}, L_{new_2}) \ldots (X_{new_k}, L_{new_k})\} \&\& L \in Label_i} Rank(X, L)$
4   Return the Label with the highest probability.
```

Fig. 3. The IR.KNN Algorithm

The detail of IR.MLKNN is in Figure 4. We mainly change the way of the Computing of Prior Probability. The array C[p] and $C'[p]$ is used to describe that with p neighbors of the instance belong to the label, whether the instance belongs to the label. If it is, C[p] will add one; else $C'[p]$ will add one.

5 Experiments and Results

In this section, we focus on the experiments on IR.KNN and IR.MLKNN, to verify our Instance-Ranking algorithm. The section includes two subsections: the experiments and results for IR.KNN comparing with KNN, and the experiments andresults for IR.MLKNN comparing with MLKNN. All the experiments are based on Weka [16] and Mulan [5].

5.1 Experiments and Results for IR.KNN

The dataset heart-statlog in the UCI[1] database is used to complete the experiment with IR.KNN. Heart-statlog dataset contains 270 instances, 13 attributes of the type

[1] `http://sourceforge.net/projects/weka/files/datasets/datasets-UCI/datasets-UCI.jar`

numeric, and two class “absent” and “present”. We compare IR.KNN and KNN from K=5 to 50 using 10-fold cross-validation with $\alpha = 0.85$ and three evaluation metrics are provided: the correctly classified instances, mean absolute error, and relative absolute error.

```
Algorithm 3:The IR.MLKNN Algorithm
Input: Training Dataset D ( $X_i, L_i$), the random teleport parameter $\alpha$, the new instance $X_{new}$
Output: The label set of the Instance $X_{new}$

IR.MLKNN (D, $\alpha, X_{new}$)
--Compute the prior probability $P(H_b^j)$
1       Compute the Rank(X,L) ($(X, L) \in$ D) using the Instance Ranking Algorithm
        For each label $\ell \in \mathcal{L}$
2           $P(H_1^j) = \sum_{i=1}^{m} Rank(X_i) * y_{X_i}(j)$
3           $P(H_0^j) = \sum_{i=1}^{m} Rank(X_i) * (1 - y_{X_i}(j))$
-- Posterior Probability $P\left(E_{\vec{C}_t(j)}^j \middle| H_b^j\right)$
For each label $\ell \in \mathcal{L}$
        For each Instance $X_i \in$ D,
4           Identity KNN($X_i$).
5           Compute the number of instances which belong to the label $\ell$. $\delta = \vec{C}_{X_i}(j) = \sum_{b \in KNN(X_i)} y_b(\ell)$
6           If($y_{X_i}(\ell)$==1) C[$\delta$] + +; else $C'[\delta]$ + +;
        For each $j \in \{0,1,2, \ldots, K\}$
7           $P(E_j^\ell | H_1^\ell) = C[j] / \sum_{p=0}^{K} C[p]$;  $P(E_j^\ell | H_0^\ell) = C'[j] / \sum_{p=0}^{K} C'[p]$
--Prediction Period
For each label $\ell \in \mathcal{L}$
8           $\vec{C}_{X_{new}}(\ell) = \sum_{b \in KNN(X_i)} y_b(\ell)$
9           $y_{X_{new}}(\ell) = \arg\max_{b \in \{0,1\}} P(H_b^\ell)\, P\left(E_{\vec{C}_{X_{new}}(\ell)}^\ell \middle| H_b^\ell\right)$
Return $y_{X_{new}}(\ell)$ $\ell \in \mathcal{L}$
```

Fig. 4. The IR.MLKNN Algorithm

Figure 5 (a)-(c) presents the experiments results for heart-statlog dataset. The correctly classified instances metric has some disturbance. For K =15,40 and 50, the correctly classified instances of KNN is better than IR.KNN. But with the whole processes from K=5 to 50, IR.KNN achieves a slight better prediction than KNN, the average correctly classified instance percentage for IR.KNN is 83.03%, while for KNN is 82.37%. Although IR.KNN and KNN achieve almost the same correctly classified instance percentages, for other evaluation metrics, IR.KNN shows much better performance. The average mean absolute error for IR.KNN is 0.17669, while for KNN is 0.27161, IR.KNN achieves 53% better than KNN for the mean absolute error average; the average relative absolute error for IR.KNN is 35.77136%, while for KNN is 54.99154%, IR.KNN achieves 53.7% better than KNN for the relative absolute error average.

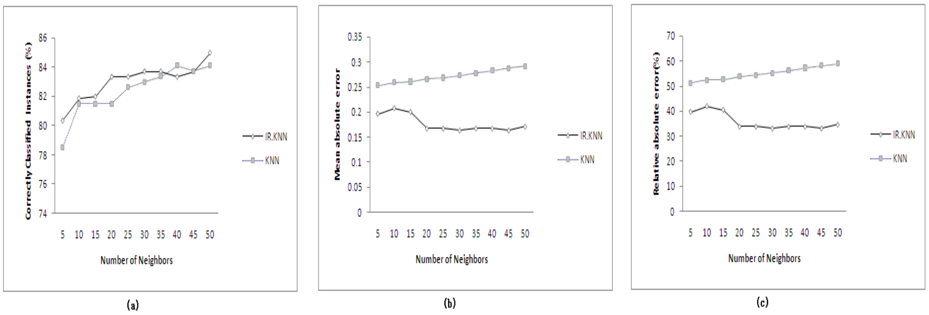


Fig. 5. The experiments results for IR.KNN and KNN for UCI heart-statlog dataset. (a) presents the correctly classified instances; (b) presents the mean absolute error; (c) presents the relative absolute error.

Table 1 presents our comparison of our IR.KNN with SVM, Naive Bayes, and decision tree [1]. We choose K=20 for IR.KNN, and the default parameters for the other algorithms as in Weka [16]. The experiment results show that our IR.KNN achieves better performance than Naive Bayes, and decision tree, and similar performance of SVM.

Table 1. Comparison of IR.KNN with SVM, Naive Bayes and Decision Tree

Evaluation Metrics	**IR.KNN**	**SVM**	**Naive Bayes**	**Decision Tree**
Accuracy	83.84%	84.07%	83.70%	76.67%
Mean Absolute Error	0.1635	0.1593	0.1835	0.2740
Relative Absolute Error	33.745%	32.25%	37.16%	55.48%

Table 2. The detail Description of the multi-label datasets emotions and CAL500

Name	**Domain**	**Instances**	**Attributes**	**Labels**	**Cardinality**	**Density**	**Distinct**
emotions	music	593	72	6	1.869	0.311	27
CAL500	music	502	68	174	26.044	0.150	502

5.2 Experiments and Results for IR.MLKNN

To verify the performance of IR.MLKNN, we used two multi-label datasets emotions and CAL500[2]. The description of those two datasets is in table 2. We compare

[2] `http://mulan.sourceforge.net/datasets.html`

IR.MLKNN with MLKNN from K=5 to 40 using 10-fold cross-validation with $\alpha = 0.85$. About four evaluation metrics [5] are provided: Hamming Loss, Average Precision, Ranking Loss, and One Error. As the evaluation metrics for multi-label classification is a bit different from single-label classification, we would firstly briefly introduce those evaluation metrics.

Results. Figure 6 and Figure 7 present the experiments results for IR.MLKNN with datasets emotions and CAL500 respectively. Table 3 summaries the average performance for those two datasets from the four evaluation metrics. Table 4 presents the results of the two datasets from the view of the maximum difference between IR.MLKNN and MLKNN. Generally speaking, IR.MLKNN achieves better performance than MLKNN consider the results of Figure 6, Figure 7, Table 3 and Table 4.

For Hamming Loss, we notice that for K=5, the IR.MLKNN is higher than the MLKNN for CAL500 in Figure 7(a), this issue happens since the neighbors for the Instance-Ranking is so small, the finally ranking cannot be used to represent the true ranking sequence for the instance of CAL500. Another interesting phenomenon is that the curves for our IR.MLKNN are closer to the curves for MLKNN in Figure 6 than those in Figure 7.

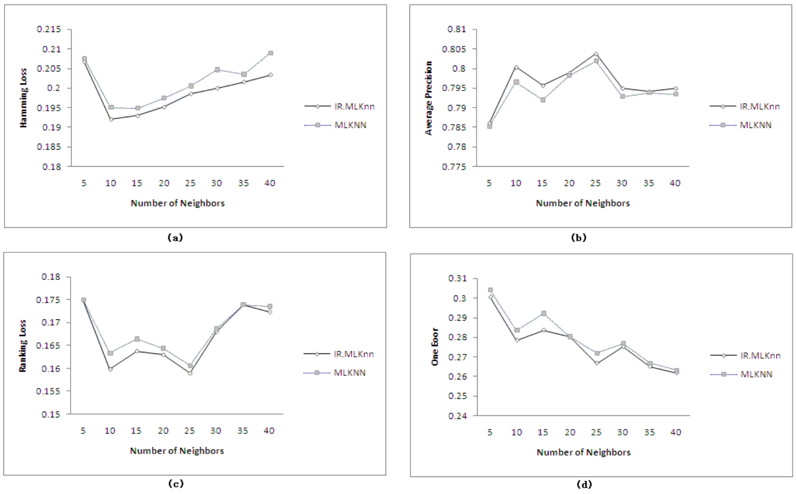


Fig. 6. The experiments results for IR.MLKNN and MLKNN for emotions dataset. (a) Hamming Loss; (b) Average Precision; (c) Ranking Loss; (d) One Error.

This is because the emotions dataset only has 6 labels, while CAL500 dataset has 174 labels. The prior probability enhancement for CAL500 is more significantly than emotions dataset. That is to say, for the datasets with more labels, the performance of our IR.MLKNN is much better.

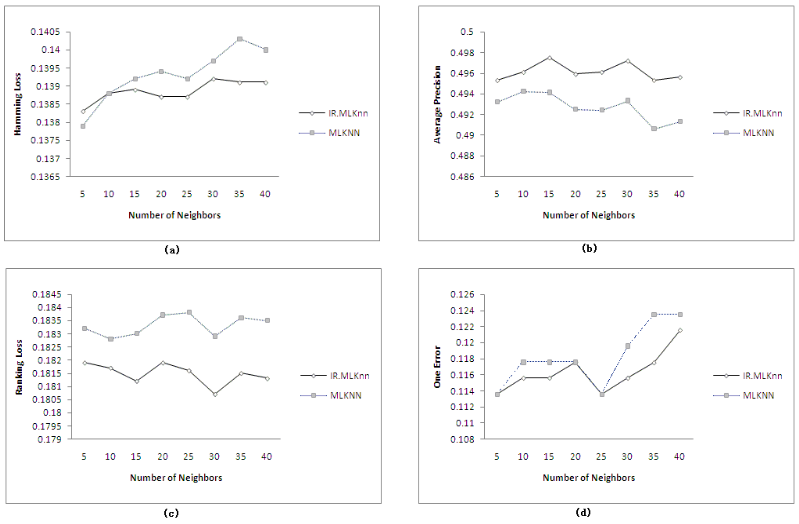


Fig. 7. The experiments results for IR.MLKNN and MLKNN for CAL500 dataset. (a) presents the Hamming Loss; (b) presents the Average Precision; (c) presents the Ranking Loss; (d) presents the One Error.

Table 3. The summarization information for IR.MLKNN and MLKNN for the datasets Emotions and CAL500 with four evaluation metrics from an average view

	Emotions		**CAL500**	
Evaluation Metrics	IR.MLKNN	MLKNN	IR.MLKNN	MLKNN
Hamming Loss	0.1987	0.2015	0.1388	0.1393
Average Precision	0.7960	0.7942	0.4961	0.4927
One Error	0.1668	0.1682	0.1814	0.1833
Ranking Loss	0.2618	0.2632	0.1215	0.1235

Table 4. The summarization information for IR.MLKNN and MLKNN for the datasets Emotions and CAL500 with four evaluation metrics from the view of maximum the margin

Evaluation Metrics	**Emotions**			**CAL500**		
	K	IR.MLKNN	MLKNN	K	IR.MLKNN	MLKNN
Hamming Loss	10	0.1920	0.1951	35	0.1391	0.1403
Average Precision	10	0.8003	0.7965	35	0.4953	0.4906
One Error	10	0.1599	0.1633	35	0.1814	0.1836
Ranking Loss	10	0.2784	0.2835	35	0.1175	0.1235

6 Conclusions

In this paper, we propose a new perspective to consider the instance dependency for classification. A novel algorithm called Instance-Ranking is presented to mining the instance dependency information. With Instance-Ranking algorithm, we modify the

KNN and MLKNN algorithm. The experiments show that the enhanced KNN and MLKNN (IR.KNN and IR.MLKNN) algorithm achieve better than the original algorithms. In the future, we will focus on the improvement of the Instance-Ranking algorithm, and apply it into more data mining applications, not only classification, but also cluster and semi-supervised problems.

Acknowledgment. This work is supported by the Ministry of Industry and Information Technology of China (No. 2010ZX01042-002-003-001).

References

1. Wu, X., Kumar, V.: The top ten algorithms in data mining. Chapman & Hall/CRC (2009)
2. Chen, X., Liu, M., Ward, R.: Protein function assignment through mining cross-species protein-protein interactions. PLoS One 3, e1562 (2008)
3. Schapire, R.E., Singer, Y.: BoosTexter: A boosting-based system for text categorization. Machine Learning 39, 135–168 (2000)
4. Trohidis, K., Tsoumakas, G., Kalliris, G., Vlahavas, I.: Multilabel classification of music into emotions. In: ISMIR (2008)
5. Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. International Journal of Data Warehousing and Mining 3, 1–13 (2007)
6. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier Chains for Multi-label Classification. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part II. LNCS, vol. 5782, pp. 254–269. Springer, Heidelberg (2009)
7. Tsoumakas, G., Vlahavas, I.: Random *k*-Labelsets: An Ensemble Method for Multilabel Classification. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 406–417. Springer, Heidelberg (2007)
8. Zhang, M.L., Zhang, K.: Multi-label learning by exploiting label dependency. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 999–1008 (2010)
9. Zhang, M.L., Zhou, Z.H.: ML-KNN: A lazy learning approach to multi-label learning. Pattern Recognition 40, 2038–2048 (2007)
10. Lin, X., Chen, X.: Mr. KNN: soft relevance for multi-label classification. In: Proc. of the 19th ACM CIKM, pp. 349–358 (1999)
11. Weston, J.: A Kernel Method for Multi-Labelled Classification. Advances in Neural Information Processing Systems 14, 681–687 (2002)
12. Zhang, M.L., Zhou, Z.H.: Multilabel neural networks with applications to functional genomics and text categorization. IEEE Transactions on Knowledge and Data Engineering, 1338–1351 (2006)
13. Clare, A., King, R.D.: Knowledge Discovery in Multi-label Phenotype Data. In: Siebes, A., De Raedt, L. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, pp. 42–53. Springer, Heidelberg (2001)
14. Langville, A.N., Meyer, C.D.: Google page rank and beyond. Princeton Univ. Pr. (2006)
15. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web (1999)
16. Bouckaert, R.R., Frank, E., Hall, M.A., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: WEKA–experiences with a java opensource project. Journal of Machine Learning Research 11, 2533–2541 (2010)
17. Xia, X., Yang, X., Li, S., Wu, C., Zhou, L.: RW.KNN: A proposed random walk knn algorithm for multi-label classification. In: Proceedings of the 4th Workshop on Workshop for Ph. D. Students in Information & Knowledge Management, pp. 87–90. ACM (2011)