RW.KNN: A Proposed Random Walk KNN Algorithm for Multi-label Classification

Xin Xia , Xiaohu Yang, Shanping Li, Chao Wu and Linlin Zhou
Computer Science College, Zhejiang University
38 Zheda Road, Hangzhou, 310027, China
{xxkidd, yangxh, shan, wuchao}@zju.edu.cn, zll.splay@gmail.com

ABSTRACT

Multi-label classification refers to the problem that predicts each single instance to be one or more labels in a set of associated labels. It is common in many real-world applications such as text categorization, functional genomics and semantic scene classification. The main challenge for multi-label classification is predicting the labels of a new instance with the exponential number of possible label sets. Previous works mainly pay attention to transforming the multi-label classification to be single-label classification or modifying the existing traditional algorithm. In this paper, a novel algorithm which combines the advantage of the famous KNN and Random Walk algorithm (RW.KNN) is proposed. The KNN based link graph is built with the k-nearest neighbors for each instance. For an unseen instance, a random walk is performed in the link graph. The final probability is computed according to the random walk results. Lastly, a novel algorithm based on minimizing Hamming Loss to select the classification threshold is also proposed in this paper.

Categories and Subject Descriptors

I.2.6 **[Artificial Intelligence]:** Learning – *concept learning*, *induction*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval –*information filtering*

General Terms

Algorithms, Theory

Keywords

Multi-label classification, KNN, Link Graph, Random Walk

1. INTRODUCTION

In the supervised learning area, most of works have been devoted to single-label classification, where each instance is assigned to a single label λ from a set of disjoint labels L , $|L| > 1$. However, in many real-world applications, one instance may belong to more than one label simultaneously. For example, in text categorization research area, one piece of text may have one or more categorizes [1]. In functional genomics classification area, a piece of Yeast gene can be assigned to a subset of functional classes [2, 3]. In the music categorization area, a single song makes a person feel happy and existence at the same time, while another song makes the person sad and nervous [4].The above examples have some common aspects, each instance is associated with a set of labels. The learning task is to predict a new instance to be the proper labels by analyzing the training instance with known labels [5] .The multi-label classification task is associated with a subset $Y \subseteq L$ for each instance. The dataset is composed of n instances $(X_1, Y_1), (X_2, Y_2) \dots (X_n, Y_n)$, where X_i denotes the input space and Y_i is the proper labelsets.

A common trend for multi-label classification is by way of *problem transformation* [6], it transforms the multi-labels problems into a set of single-label problems. The *problem transformation* methods is based on binary relevance (BR) [7-11] or label powerset (LP) [12-14].

An alternative to the *problem transformation* method is *algorithm adaptation* methods [11], which extends specific learning algorithms in order to handle multi-label data directly. These algorithms include lazy algorithm (ML-KNN [15] and Mr-KNN [16]), AdaBoosting.MH [1], Rank-SVM [17], BP-MLL [18] and decision trees [3].

In this paper, we propose a novel algorithm called Random Walk KNN algorithm (RW.KNN), which introduces the KNN based link graph, and with random walking in the link graph, the ranking of each label is generated. The contribution of this paper is summarized as follows:

- Introduce the KNN based link graph, which provides the environment for random walk in the graph.
- Propose a new Random Walk KNN algorithm, this algorithm not only gives the classification results, but also provides the ranking of each label.
- Propose a new algorithm to compute the classification threshold based by minimizing Hamming Loss evaluation metrics.

The rest of this paper is organized as follows: the related works about KNN and random walk is briefly reviewed in Section 2. In Section 3, the detail of RW.KNN algorithm and the classification threshold algorithm are proposed. In Section 4, the conclusion and future work is provided.

2. RELATED WORK

In this section, we first give the formal notation of multi-label classification, and then briefly review the KNN [19] algorithm and Random Walk model [20].The following notation is used throughout this paper. Let χ denote the input space. Given the multi-label training dataset $D = \{(X_i, Y_i)\}_m^{i=1}$ where $X_i \in \chi \; and \; Y_i \subseteq Y$ .let $Y = \{1,2,3 \dots |L|\}$ denotes the $|L|$ finite set drawn from an unknown distribution D, the goal of multi-label classification is to learn a hypothesis $h: \chi \rightarrow 2^Y$ which is used to predict the proper label of the new instance.

Table 1. The instances of Datasets D and its 2NN

	Input space	Labelsets Y	2NN
1	(1,1)	$\{\lambda_1, \lambda_2\}$	{2,7}
2	(2,1)	$\{\lambda_1\}$	{1,3}
3	(2,2)	$\{\lambda_2, \lambda_3\}$	{2,4}
4	(3,2)	$\{\lambda_1, \lambda_3\}$	{2,3}
5	(1,3)	$\{\lambda_2\}$	{6,7}
6	(2,3)	$\{\lambda_3\}$	{3,5}
7	(1,2)	$\{\lambda_1, \lambda_2, \lambda_3\}$	{1,5}

2.1 KNN Algorithm

K-nearest neighbor (KNN) [19] algorithm has a long history in the data mining area for single-label classification. Simply to say, KNN finds a group of nearest k instances in the training set, and bases the assignment of a label on the predominance of a particular label in its neighborhood. It seems easy to modify the KNN algorithm for multi-label classification, just divide the labelsets into |L| binary labels. For each label, KNN algorithm is used to decide whether the instance belongs to this label. Although KNN for multi-label classification is easy, but it ignores the most important information: the labels correlation.

ML-KNN [15] considered 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.

2.2 Random Walk Model

Random walk model [20] has successfully applied in the area of data mining and Internet application. The most famous application for random walk model is Google's PageRank [21, 22]. The basic idea for Random Walk is: Suppose someone visits the graph from one or a set of vertexes. In anyone of the vertex v, he has the probability α to visit the neighbor vertex of v, and he 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.

3. RW.KNN : Algorithm Description

RW.KNN has two components: A KNN based multi-label link graph and a modified random walk algorithm in the KNN based link graph.

3.1 A KNN based link graph

RW.KNN first maps the Training dataset D into a multi-dimensionality multi-label link graph. For each training instance $(X_i, Y_i) \in D$, there is a unique vertex v_i representing it. The links between vertexes represent the correlation between the instances of training dataset, and the weights of the links measure the correlation degree of the instances.

The core idea of multi-label link graph is : for each instance $(X_i, Y_i) \in D$, found its KNN (denoted them as $\{(X_{i_1}, Y_{i_1}), (X_{i_2}, Y_{i_2}), \dots (X_{i_k}, Y_{i_k})\} \subseteq D$), then there is a link between pair of vertexes $(v_i, v_{i_p}), p = \{1,2, \dots k\}$.

Definition 1: the KNN based link graph is defined as :

$G = (V, E)$

$V = \{v_i | (X_i, Y_i) \in D\}$

$E = \{(v_i, v_j) | v_i, v_j \in V, v_i \in KNN(v_j) \ or \ v_j \in KNN(v_i), i \neq j\}$

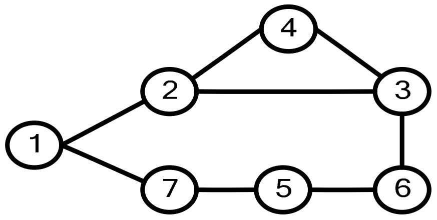

Figure 1. The 2NN based link graph with the training set

The remaining parts of the paper will use v_i instead of the training instance $(X_i, Y_i) \in D$ in the link graph. The KNN based link graph is a fully connected graph without any separate components as for each vertex v_i, as its K-Nearest-Neighbor can be always found.

Definition 2: the weight matrix W of the KNN based link graph is :

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

W is a Normalized matrix with $\sum_{j=1}^{m} W_{ij} = 1$. The Euclid distance is used to describe the $distance(v_i, v_j)$. from the definition of weight matrix W, one can conclude that weight matrix is not a symmetrical matrix.

For example, the labelsets are$Y = \{\lambda_1, \lambda_2, \lambda_3\}$, and the input space χ is a 2-dimension coordinate. Table 1 shows the training set D, with $K = 2$. For vertex 1, its 2-Nearest-Neoghbor is vertex 2 and vertex 7 with distance 1, so in the link graph, vertex 1 and vertex 2 , vertex 1 and vertex 7 will have a undirected edge. For vertex 2 , its 2-Nearest-neighbor is vertex 1 and 3, as vertex 1 and vertex 2 has already connected, an edge between vertex 2 and vertex 3 will be added. Figure 1 shows the 2NN based link graph of the example.

3.2 The procedure of Random Walk on KNN based link graph

3.2.1 Procedure of Random Walk

The building of random walk for RW.KNN needs four arguments: the adjacent probability matrix $\mathbf{P}$, the parameter α, the initial probability distribution $\boldsymbol{\pi}^{(0)}$, and the probability vector of random teleporting to any other vertex $\mathbf{t}$. Assume after k-th random walk, $\boldsymbol{\pi}^{(k)}$ denote the output probability. Then the random walk process can be written:

$$\boldsymbol{\pi}^{(k)T} = \boldsymbol{\pi}^{(k-1)T}(\alpha \cdot \mathbf{P} + (\mathbf{1} - \alpha) \cdot \mathbf{et}^{T}) \quad (2)$$

Fortunately, the *weight matrix W* can replace the adjacent probability matrix $\mathbf{P}$, since $\sum_{j=1}^{m} W_{ij} = 1$.Then the random walk process can be rewritten as :

$$\boldsymbol{\pi}^{(k)T} = \boldsymbol{\pi}^{(k-1)T}(\alpha \cdot \mathbf{W} + (\mathbf{1} - \alpha) \cdot \mathbf{et}^{T}) \quad (3)$$

3.2.2 Random Walk on KNN based link graph

After building the KNN based link graph such as Figure 1 shows, when a new unseen instance arrives (denote it as vertex v_0), RW.KNN start with the vertex v_0, random walk around until $\boldsymbol{\pi}^{(k)}$ get convergent. In the appendix, we prove that this random walk model has a unique convergent point.

Definition 3: the KNN based Random Walk graph is defined as :

$G_r = (V_r, E_r)$

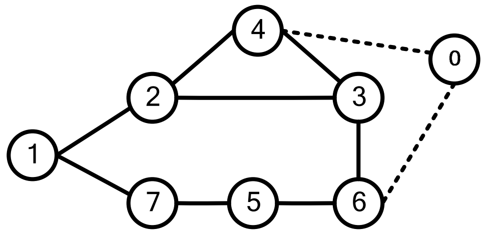

Figure 2 The 2NN based Random Walk graph

$V_r = V \cup \{v_0\}$

$E_r = E \cup \{v_j \in V, v_j \in KNN(v_0), j \neq 0\}$

Figure 2 shows the KNN based Random Walk graph with a new instance with input space $X_0 = (3,3)$. The 2NN for the new instance is vertex 4 and vertex 6. With the KNN based Random Walk graph, the procedure of the RW.KNN is as follows:

- Compute the initial probability distribution π^0. π^0 is computed with a similar ways to computer *weight matrix* $\boldsymbol{W}$. π^0 satisfies $\sum_{i=1}^{m} \pi^0(i) = 1$.

$$\pi^0(i) = \begin{cases} \frac{distance\ (v_0, v_i)}{\sum_{v \in KNN(v_0)} distance\ (v_0, v)}, (v_0, v_i) \in E_r \\ 0, \text{other else} \end{cases} \quad (4)$$

- Set the probability vector of random teleporting to any other vertex $\mathbf{t}$. In RW.KNN, suppose that the probability of teleport to any vertexes is the same. That is

$$\mathbf{t} = (\frac{1}{\mathbf{m}}, \frac{1}{\mathbf{m}}, \frac{1}{\mathbf{m}} \dots \frac{1}{\mathbf{m}})^{\mathbf{T}} \quad (5)$$

- Choose the proper parameter α, $\alpha \in (0,1)$. There are a lot of literature discussing the best practice of α. The choosing of α need to make a balance between efficiency and effectiveness. If $\alpha \to 0$, then the iteration times is huge, while if $\alpha \to 1$, the the adjacent probability matrix $\mathbf{P}$ become much more volatile, and fluctuate noticeably for even small changes of $\mathbf{P}$. To make the problem simple, α is set to 0.85, which is the same value for Google's PageRank.
- Compute the adjacent probability matrix $\mathbf{P}$, which is the same as the *weight matrix* W.
- Iterate (3) until the π^k get convergent. Then the final probability from the new vertex v_0 to all the other vertex v_i is generated, denote it as π^{final}.

π^{fianl} builds the relationship between the new instance and the other instance in the training data set. If $\pi^{\text{final}}(i)$ is large, it means the new instance has a close relationship with the i-th instance in the training set, That is to say, the i-th instance takes the deeper impact on the predicting for the new instance's labesets. From this perspective, the probability of a label belongs to a new instance is:

$$\mathbf{P}(\boldsymbol{\lambda}_i \in \mathbf{Y}_{\text{new}}) = \sum_{p=1}^{m} \boldsymbol{\pi}^{\text{final}}(\mathbf{p})\, \mathbf{I}(\boldsymbol{\lambda}_i \in \mathbf{Y}_p) \quad (6)$$

Where $\mathbf{I}(\text{true}) = 1$ and $\mathbf{I}(\text{false}) = 0$ .The whole algorithm for RW.KNN is in Figure 3

3.3 The classification threshold algorithm

RW.KNN provides a probability distribution of labelsets for the new instance, but it doesn't address whether the new instance belong to a particular label. The simplest way is setting a user-specific threshold p_0, if $P(\lambda_i \in Y_{new}) > p_0$ then this label belong to the instance; else otherwise.

Algorithm 1 :The Random Walk KNN algorithm

Input: training Dataset D, the labelsets Y, the random walk probability α, the new instance X_0

Output: the probability distribution of labelsets for the new instance X_0

```
RW.KNN(D,Y, α, X_0)
1. Build the KNN based link graph G from D;
2. Based on G, integrate the new instance $X_0$ to form the KNN based Random Walk graph $G_r$;
3. Compute initial probability distribution $\pi^0$ using (4);
4. Compute the probability vector of random teleporting to any other vertex t using (5)
5. Compute the weight matrix W using (1)
6. Iterate (3) until the $\pi^k$ get convergent.
7. Using (6) to provide the final output
```

Figure 3. The Random Walk KNN algorithm

But the user-specific threshold p_0 omits how to define the threshold p_0 to make a good prediction. In this section, we propose a new novel algorithm, which based on the principle of minimization the Hamming Loss.

The Hamming loss is defined as :

$$Hamming\ Loss = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{|Y|}|h(X_i)\Delta Y_i| \quad (7)$$

Here, Δ represents the exclusive OR (XOR) Boolean logic operation. It is used to compute the average binary classification error. The smaller the Hamming Loss, the better prediction performance it achieves.

With the principle of minimization of Hamming Loss, the classification threshold algorithm is as follows:

- Random sample a small data set D' from the original training data set D. For each $(X_{i'}, Y_{i'}) \in D'$, using RW.KNN algorithm to get the probability distribution of the labels. Then there are $|D'|$ instances with the predicted labels probability distribution and the true labelsets $Y_{i'}$.

  Table 2 shows an example, with $Y = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\}$, and $|D'| = 3$. If the threshold is set to be 0.3, then for instance 1, the predicted labelset is $\{\lambda_1, \lambda_2\}$; for instance 2, the predicted labelset is $\{\lambda_4\}$; for instance 3,the predicted labelset is $\{\lambda_4\}$. Then the Hamming Loss for D' is $1/3$. But if the threshold is set to 0.15, then the Hamming Loss is 0.
- Found the threshold t that minimize the Hamming Loss in the dataset D':

$$t = \arg\min_{p(\lambda)} \frac{1}{|D'|}\sum_{i=1}^{|D'|}\frac{1}{|Y|}|h(X_i)\Delta Y_i| \quad (8)$$

Figure 4 shows a heuristic algorithm for the approximate optimize threshold t.

3.4 Complexity Analysis

For multi-label classification with |L| labels, and m training

Table 2.True and Predicted labelsets probability for D'

	True labelsets	Predicted labels probability distribution
1	$\{\lambda_1, \lambda_2, \lambda_3\}$	$\mathbf{P}(\lambda_1) = 0.4, \mathbf{P}(\lambda_2) = 0.3, \mathbf{P}(\lambda_3) = 0.1,$ $\mathbf{P}(\lambda_4) = 0.05, \mathbf{P}(\lambda_5) = 0.05$
2	$\{\lambda_2, \lambda_3, \lambda_4, \lambda_5\}$	$\mathbf{P}(\lambda_4) = 0.5, \mathbf{P}(\lambda_2) = 0.2, \mathbf{P}(\lambda_3) = 0.2,$ $\mathbf{P}(\lambda_5) = 0.3, \mathbf{P}(\lambda_1) = 0$
3	$\{\lambda_3, \lambda_4\}$	$\mathbf{P}(\lambda_4) = 0.7, \mathbf{P}(\lambda_3) = 0.2, \mathbf{P}(\lambda_1) = 0.1,$ $\mathbf{P}(\lambda_2) = 0, \mathbf{P}(\lambda_5) = 0$

Algorithm 2: Heuristic algorithm for the approximate optimize threshold t

```
Input: KNN based Random Walk graph $G_r$, the labelsets Y, The random sampling dataset $D'$
Output: the approximate optimize threshold t
OptThreshold($G_r$,Y, $D'$)
Initial the 2-Dimension probability array P
1.  For each instance $(X_i, Y_i) \in D'$
      $P(i) =$ the result of instance $X_i$ of algorithm RW.KNN
2.  Set min_threshold=1, hamming_loss = $\infty$
3.  Ranking all the item in P(i), store it in the array R
4.  For each item r in R
      tmp_hamloss = $\frac{1}{|D'|}\sum_{i=1}^{|D'|}\frac{1}{|Y|}|h(X_i)\Delta Y_i|$ , with threshold r
      if(tmp_hamloss <=hamming_loss)
          min_threshold= r
          hamming_loss=tmp_hamloss
5.  return min_threshold
```

Figure 4. Heuristic algorithm for approximate optimize threshold.

instances, the time complexity for build KNN based link graph is $O(m^2)$ without optimization. For the random walk in the graph, the time complexity to the convergent point is $O(\log|E|) = O(\log m)$. So the total time complexity for RW.KNN is $O(m^2 + \log m)$.

4. CONCLUSIONS AND FUTURE WORKS

In this paper, a novel algorithm RW.KNN is proposed for multi-label classification. RW.KNN combines the advantage of KNN and the Radom Walk model, which provides a new perspective for multi-label learning. The KNN based link graph makes the random walk in the multi-label datasets possible.

Currently, the research of RW.KNN is ongoing. The experiment of RW.KNN has not completed yet. The experiment is based on the open source software Mulan [5].

In the future, a complete version of this paper will generate, which contains the experiment of RW.KNN, and the comparison of the state-of-art algorithms, such like MLKNN,ECC and RAKEL.

5. ACKNOWLEDGMENTS

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

REFERENCES

[1] R. E. Schapire and Y. Singer, "BoosTexter: A boosting-based system for text categorization," Machine Learning, vol. 39, pp. 135-168, 2000.

[2] X. Chen, M. Liu, and R. Ward, "Protein function assignment through mining cross-species protein-protein interactions," PLoS One, vol. 3, p. e1562, 2008.

[3] A. Clare and R. King, "Knowledge discovery in multi-label phenotype data," Principles of Data Mining and Knowledge Discovery, pp. 42-53, 2001.

[4] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas, "Multilabel classification of music into emotions," 2008.

[5] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," International Journal of Data Warehousing and Mining, vol. 3, pp. 1-13, 2007.

[6] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," Data Mining and Knowledge Discovery Handbook, pp. 667-685, 2010.

[7] J. Furnkranz, E. Hullermeier, E. Loza Mencia, and K. Brinker, "Multilabel classification via calibrated label ranking," Machine Learning, vol. 73, pp. 133-153, 2008.

[8] N. Ghamrawi and A. McCallum, "Collective multi-label classification," in Proceedings of the 14th ACM international conference on Information and knowledge management, 2005, pp. 195-200.

[9] E. Hullermeier, J. Furnkranz, W. Cheng, and K. Brinker, "Label ranking by learning pairwise preferences," Artificial Intelligence, vol. 172, pp. 1897-1916, 2008.

[10] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," Machine Learning and Knowledge Discovery in Databases, pp. 254-269, 2009.

[11] M. L. Zhang and K. Zhang, "Multi-label learning by exploiting label dependency," in Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 2010, pp. 999-1008.

[12] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification," Advances in Knowledge Discovery and Data Mining, pp. 22-30, 2004.

[13] J. Read, "A pruned problem transformation method for multi-label classification," in Proceedings of the New Zealand Computer Science Research Student Conference 2008, 2008, p. 143¨C150.

[14] G. Tsoumakas and I. Vlahavas, "Random k-labelsets: An ensemble method for multilabel classification," Machine Learning: ECML 2007, pp. 406-417, 2007.

[15] M. L. Zhang and Z. H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," Pattern Recognition, vol. 40, pp. 2038-2048, 2007.

[16] X. Lin and X. Chen, "Mr. KNN: soft relevance for multi-label classification," in Proceedings of the 19th ACM International Conference on Information and Knowledge Management, 2010, pp. 349-358.

[17] A. Elisseeff and J. Weston, "A Kernel Method for Multi-Labelled Classification," in Advances in Neural Information Processing Systems 14, 2002, pp. 681-687.

[18] M. L. Zhang and Z. H. Zhou, "Multilabel neural networks with applications to functional genomics and text categorization," IEEE transactions on Knowledge and Data Engineering, pp. 1338-1351, 2006.

[19] X. Wu and V. Kumar, The top ten algorithms in data mining: Chapman & Hall/CRC, 2009.

[20] L. Lovasz, "Random walks on graphs: A survey," Combinatorics, Paul Erdos is Eighty, vol. 2, pp. 1-46, 1993.

[21] A. N. Langville and C. D. Meyer, Google page rank and beyond: Princeton Univ Pr, 2006.

[22] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," 1999.