DW-ML-kNN: A Dual Weighted Multi-label kNN Algorithm

Duoqian Miao, Zhifei Zhang*, Zhihua Wei, Chuanyan Wang

Department of Computer Science and Technology, Tongji University, No. 4800, Cao'an Highway Shanghai, 201804, China
{dqmiao, zhihua_wei}@tongji.edu.cn, zhifei.zzh@gmail.com, chunyanfriend@163.com

Multi-label learning is a continually developing field in machine learning and has wide applications in many domains. ML-kNN is a simple but effective method to learn from multi-label data. But when data are imbalanced, its performance is not encouraging. Therefore, we propose a dual weighted ML-kNN (for short, DW-ML-kNN) algorithm, which utilizes distances of neighbors as weights and considers the impact of neighbors without one certain label. The experiment results on three multi-label datasets show that the new algorithm achieves better performance than ML-kNN.

Keywords: Multi-label learning; k nearest neighbors; distance weight.

1. Introduction

Multi-label data are popular in our daily life. For example, an image may be classified into semantic classes *sunset* and *beach* simultaneously; a news report is about *politics*, as well as *military*. Recently, the issue of learning from these data, which is called multi-label learning, has drawn significant attention of researchers. Multi-label learning is widely used in many domains, such as text classification[1,2,3], semantic annotation of videos[4], image annotation[5,6] and functional genomics[7,8]. There are two major tasks in multi-label learning: multi-label classification and label ranking[9]. The methods of solving this problem are grouped into two categories: problem transformation and algorithm adaptation[10].

Problem transformation is to transform the multi-label learning problem into one or more single-label learning problem, for which traditional learning algorithms are used. Ranking by pairwise comparison proposed by Hüllermeier et al.[11] constructs multiple binary classifiers, and determines the label ranking of a new instance by voting. Calibrated label ranking[12] extends it by introducing an additional virtual label, which assigns the labels ranking before the virtual label to unseen example. Zhang et al.[13] gives an algorithm called INSDIF, in which each prototype for each label is computed with the all instances belonging to this label as the training set. This group of methods poses an important complexity problem, especially for large value of the number of labels. PPT[14] and RAkE[15] methods solve the problem to some extent.

* Corresponding Author

Algorithm adaptation is to extend single-label learning algorithms in order to deal with multi-label data. Clare and King[8] adopt C4.5 algorithm to solve this problem with a modified formula of entropy calculation. AdaBoost.MH and AdaBoost.MR, which are two extensions of AdaBoost, are proposed by Schapire and Singer[1]. They are used in multi-label classification and label ranking respectively. A modified SVM algorithm is proposed in Ref. 16, and then three improvements combining the binary relevance with SVM classifiers are given in Ref. 2. Thabtah et al.[17] propose MMAC algorithm based on the paradigm of associative classification, in which the labels are ranked according to the support of rules. Zhang and Zhou[5] propose ML-kNN algorithm based on k Nearest Neighbors (for short, kNN) lazy learning algorithm, using the maximum a posterior principle to determine the labels of unseen example, in which prior and posterior probabilities for the frequency of each label in the k nearest neighbors are calculated.

The paper focuses on ML-kNN algorithm. This algorithm is simple, but performs poor when data are imbalanced, and tends to assign some labels with high frequency, because of concerning the frequency of each label only. Thus, we proposed a dual weighted ML-kNN algorithm named DW-ML-kNN. It takes into account not only the distances between an instance and its neighbors for each label, but also the labels not appearing in the neighbors. The distances are converted to two weights where the labels are present and not present in the neighbors. This is why we call it "dual weighted". The experiment results on three benchmark datasets show that our algorithm achieves better performance than ML-kNN.

The remainder of this paper is organized as follows. Section 2 describes ML-kNN algorithm. The proposed algorithm is demonstrated in Section 3 in detail. Section 4 shows the experiment results. The conclusion is drawn in Section 5.

2. Multi-label Learning

2.1. *Problem Statement*

In multi-label learning, the task is to predict the true label set for an unseen instance with the model, which is established according to the training instances associated with a set of labels.

Let X denote the domain of instances and $Y = \{1, 2, ..., q\}$ denotes the finite set of labels. A training set $S = \{(x_i, Y_i)\}, i = 1, ..., m$ is given, in which $x_i \in X$ and $Y_i \subseteq Y$. The learning algorithm is to output a multi-label classifier $h: X \to 2^Y$ for multi-label classification or a label ranking mapping $f: X \times Y \to R$. In a word, the goal of multi-label learning is to get the classification function or the ranking function.

2.2. *ML-kNN Algorithm*

ML-kNN is an effective method of handling multi-label data, which combines Bayesian probability theory and single-label kNN algorithm.

Let $\vec{y}_x$ be the label vector for an instance $x = x_i$, where its l-th component $y_x(l)$ takes the value of 1 if $l \in Y_i$ and 0 otherwise. Let $N(x)$ be the set of K nearest neighbors of the instance x in the training set, and $C_x(l)$ be the number of neighbors with label l of the instance x, calculated by the following formula.

$$C_x(l) = \sum_{a \in N(x)} y_a(l) \tag{1}$$

For an instance t, ML-kNN firstly indentifies the set of K nearest neighbors $N(t)$ in the training set. Let H_b^l denote the hypothesis of the instance t with the label $l(b=1)$ and without the label $l(b=0)$. Let $E_j^l (j = 0,1,...,K)$ denote the event that there are exactly j instances with the label l in the K nearest neighbors. Therefore, the label vector of instance t is predicted using the following MAP principle:

$$y_t(l) = \arg\max_{b \in \{0,1\}} P(H_b^l \mid E_{C_t(l)}^l) \tag{2}$$

By the Bayesian Rule, Eq. (2) can be rewritten as:

$$y_t(l) = \arg\max_{b \in \{0,1\}} P(H_b^l) P(E_{C_t(l)}^l \mid H_b^l) \tag{3}$$

The priori probabilities $P(H_b^l)$ and the posterior probabilities $P(E_{C_t(l)}^l \mid H_b^l)$ can be directly estimated based on frequency counting from the training set.

3. DW-ML-kNN Algorithm

In this section, the main idea of our algorithm is described. And we concretely illustrate DW-ML-kNN algorithm with some comments.

3.1. *Main Idea*

The performance of ML-kNN algorithm is poor when data are imbalanced. We propose DW-ML-kNN algorithm based on the following two considerations:

- Usually, the instances with the same label are concentrated, while the instances with different labels are decentralized. We assign higher weights to closer neighbors and lower weights to farther neighbors.
- Neighbors without a certain label have an impact on determining the label set of an unseen instance. If there is a higher probability with a label for an instance but a lower without this label, the instance can be assigned this label to a large extent.

Two weights are assigned to neighbors where a certain label is present and absent respectively. This is the reason why our algorithm is named dual weighted ML-kNN.

Let $d(x,a), a \in N(x)$ denote the distance between the instance x and one of its K nearest neighbors. The distance is regularized as:

$$d_{reg}(x,a)=\frac{d(x,a)-\min_{u\in N(x)} d(x,u)}{\max_{u\in N(x)} d(x,u)-\min_{u\in N(x)} d(x,u)} \tag{4}$$

The regularized distance always takes a value within the interval [0,1]. In our implementation, a small positive constant is added to the denominator in order to avoid the denominator of 0. This could happen if all the K neighbors' distances are the same.

The Gaussian kernel is utilized to transform the distance between x and a to the probability that they have the same label, which is implemented with

$$p(a\mid x)=\frac{1}{\sqrt{2\pi}}e^{-d_{reg}(x,a)^2/2} \tag{5}$$

The weighted value of the instance x with the label l according to all its K neighbors is calculated by the normalization of Eq. (5).

$$w_x(l)=\frac{\sum_{a\in N(x)\wedge y_a(l)=1} p(a\mid x)}{\sum_{a\in N(x)} p(a\mid x)} \tag{6}$$

Then, we construct the classification function for our algorithm, including two posterior probabilities:

$$y_t(l)=\arg\max_{b\in\{0,1\}}\{\lambda P(H_b^l\mid E^l_{C_t(l)})+(1-\lambda)P(H_b^l\mid \tilde{E}^l_{K-C_t(l)})\} \tag{7}$$

$\tilde{E}^l_{K-C_t(l)}$ is the event that there are exactly $K-C_t(l)$ instances without the label l in the K nearest neighbors, and λ is a tuning factor, measuring the significance level of these two probabilities.

Using the Bayesian Rule, Eq. (7) can be rewritten as:

$$y_t(l)=\arg\max_{b\in\{0,1\}}\{P(H_b^l)[\lambda P(E^l_{C_t(l)}\mid H_b^l)+(1-\lambda)P(\tilde{E}^l_{K-C_t(l)}\mid H_b^l)]\} \tag{8}$$

3.2. *Algorithm Description*

The process of our algorithm is listed below:

(i) To compute the prior probabilities from the given training set. A smoothing parameter is used to control the strength of uniform prior.

(ii) To compute the posterior probabilities from the given training set. Different from ML-kNN algorithm, our algorithm replaces simple frequency counting by distance weighting. The word "weighted" in the name embodies here.

(iii) To compute the label set and label ranking for the unseen instance. A tuning factor is used to control the significant level between the two probabilities of assigning one certain label and not assigning the certain label. The word "dual" in the name embodies here.

Fig. 1 describes the pseudo code of DW-ML-kNN algorithm. Steps from (1) to (4) are corresponding to (i) of the process. m is the number of instances in the training set. And the parameter s is set to be 1 which yields the Laplace smoothing.

Input: Training set S, unseen instance t, number of nearest neighbors K, smoothing parameter s, tuning factor λ.
Output: Label set $\vec{y}_t$ and ranking $\vec{r}_t$.
% Computing the prior probabilities
(1) $m = |S|$;
(2) **For** $l \in \{1,...,q\}$
(3) $P(H_1^l) = \dfrac{s + \sum_{i=1}^{m} y_i(l)}{s \times 2 + m}$;
(4) $P(H_0^l) = 1 - P(H_1^l)$;
% Computing the posterior probabilities
(5) **Indentify** $N(x_i)$, denoted by $N(x)$;
(6) **Transform** distance using Eq. (4);
(7) **For** $l \in Y$
(8) Computing the weight $w_x(l)$ using Eq. (5) and Eq. (6);
(9) $\tilde{w}_x(l) = 1 - w_x(l)$;
(10) **For** $j = 0$ to K
(11) $c[j] = c'[j] = \tilde{c}[j] = \tilde{c}'[j] = 0$;
(12) **For** $i = 1$ to m
(13) $\delta = C_x(l)$;
(14) **If** $y_x(l) == 1$ **Then** $c[\delta] += w_x(l)$; $\tilde{c}[K - \delta] += \tilde{w}_x(l)$;
(15) **Else** $c'[\delta] += w_x(l)$; $\tilde{c}'[K - \delta] += \tilde{w}_x(l)$;
(16) **For** $j = 0$ to K
(17) $P(E_j^l \mid H_1^l) = \dfrac{s + c[j]}{s \times (K+1) + \sum_{p=0}^{K} c[p]}$; $P(E_j^l \mid H_0^l) = \dfrac{s + c'[j]}{s \times (K+1) + \sum_{p=0}^{K} c'[p]}$;
(18) $P(\tilde{E}_j^l \mid H_1^l) = \dfrac{s + \tilde{c}[j]}{s \times (K+1) + \sum_{p=0}^{K} \tilde{c}[p]}$; $P(\tilde{E}_j^l \mid H_0^l) = \dfrac{s + \tilde{c}'[j]}{s \times (K+1) + \sum_{p=0}^{K} \tilde{c}'[p]}$;
% Computing the label set and ranking
(19) **Indentify** $N(t)$;
(20) **For** $l \in Y$
(21) $C_t(l) = \sum_{a \in N(t)} y_a(l)$;
(22) Computing $y_t(l)$ by Eq. (8);
(23) $r_t(l) = \dfrac{P(H_1^l)[\lambda P(E_{C_t(l)}^l \mid H_1^l) + (1-\lambda) P(\tilde{E}_{K-C_t(l)}^l \mid H_1^l)]}{\sum_{b \in \{0,1\}} P(H_b^l)[\lambda P(E_{C_t(l)}^l \mid H_b^l) + (1-\lambda) P(\tilde{E}_{K-C_t(l)}^l \mid H_b^l)]}$;

Fig.1 Pseudo code of DW-ML-kNN algorithm

Steps from (5) to (18) are corresponding to (ii) of the process. Among them, step (8) is to measure the weighted value $w_x(l)$ of the instance x with the label l, $\tilde{w}_x(l)$ in step (9) is the weighted value of the instance x without the label l. Steps from (10) to (15) are to calculate $c[j], c'[j], \tilde{c}[j], \tilde{c}'[j]$, where $c[j]$ counts the weighted sum of training instances with the label l whose K nearest neighbors contain j instances with

the label l, $c'[j]$ counts the weighted sum of training instances without l whose K nearest neighbors contain j instances with l, $\tilde{c}[j]$ counts the weighted sum of training instances with l whose K nearest neighbors contain j instances without l, and $\tilde{c}'[j]$ counts the weighted sum of training instances without l whose K nearest neighbors contain j instances without l. Here, weighted sum is different from that of ML-kNN algorithm, in which simple frequency counting is adopted. Steps from (16) to (18) are to estimate the posterior probabilities; step (17) is the same with that of ML-kNN algorithm, but step (18) is added to compute the posterior probabilities of not assigning one certain label.

Steps from (19) to (23) are corresponding to (iii) of the process. The formula of computing the label set of an unseen instance in step (22) is the classification function in our algorithm, as shown in Eq. (8). Correspondingly, the formula of computing the label ranking in step (23) is modified. It's worth mentioning that the tuning factor λ is set to be 0.5 which yields the equal significant level of the dual posterior possibilities.

4. Experiments

4.1. *Data Sets*

Three multi-label datasets from three different application domains are collected for our experiments, which are Yeast dataset, Scene dataset and Emotions dataset. Yeast is a biological dataset about protein function classification, Scene is an image dataset about semantic indexing of still scenes and Emotions dataset is a music dataset about song classification according to the emotions.

The statistics of the three datasets are shown in Table 1, such as the number of examples $|S|$, the number of attributes $D(S)$, the number of labels $L(S)$, the label cardinality $LC(S)$ and the label density $LD(S)$ [10]. Label cardinality is the average number of labels per example, and label density is the same number divided by the total number of labels.

Table 1. Datasets and their statistics

Data set	Domain	$\|S\|$	$D(S)$	$L(S)$	$LC(S)$	$LD(S)$
Yeast	Biology	2417	103	14	4.237	0.303
Scene	Multimedia	2712	294	6	1.074	0.179
Emotions	Music	593	72	6	1.869	0.311

4.2. *Evaluation Criteria*

In our experiments, we compare DW-ML-kNN algorithm with ML-kNN algorithm. To evaluate their performance, six criteria commonly-used in multi-label learning are chosen. Among them, Hamming loss, Precision and Recall are classification-based,

One-error, Coverage and Average precision are ranking-based. Their expressions and meanings are described in Table 2.

Table 2. Six evaluation Criteria

Criterion	Expression	Meaning
Hamming loss↓	$Hamming-loss=\frac{1}{m}\sum_{i=1}^{m}\frac{\lvert h(x_i)\Delta Y_i\rvert}{\lvert Y\rvert}$	Average times that an instance-pair is misclassified.
Precision↑	$Precision=\frac{1}{m}\sum_{i=1}^{m}\frac{\lvert h(x_i)\cap Y_i\rvert}{\lvert h(x_i)\rvert}$	Average fraction the truly predicted labels of the predicted labels.
Recall↑	$Recall=\frac{1}{m}\sum_{i=1}^{m}\frac{\lvert h(x_i)\cap Y_i\rvert}{\lvert Y_i\rvert}$	Average fraction the truly predicted labels of the true labels.
One-error↓	$One-error=\frac{1}{m}\sum_{i=1}^{m}\arg\max_{l\in Y} f(x_i,l)\notin Y_i$	Average times that the top-ranked label is not in the true label set.
Coverage↓	$Coverage=\frac{1}{m}\sum_{i=1}^{m}\left\lvert\{l\mid f(x_i,l)\geq\min_{l'\in Y_i} f(x_i,l')\}\right\rvert-1$	Average steps that we need go down the list of labels in order to cover the true label set.
Average precision↑	$AvgPrec=\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\lvert Y_i\rvert}\sum_{l\in Y_i}\frac{\lvert\{l'\in Y_i\mid f(x_i,l)\geq f(x_i,l')\}\rvert}{\lvert\{l'\in Y\mid f(x_i,l)\geq f(x_i,l')\}\rvert}$	Average fraction of labels ranked above a particular label in the true label set.

Note: ↑tells higher value better performance, ↓ tells lower value, better performance.

4.3. *Experiment Results*

There is a key parameter in our algorithm, the number of nearest neighbors K. The algorithm is executed with a varying number of nearest neighbors. The parameter K ranges from 5 to 15. The following experiment results include three respective results on three datasets and the overall result.

The comparison results on Yeast dataset are illustrated in Fig. 2. The horizontal coordinate is the number of nearest neighbors and the vertical coordinate is the value of the criterion.

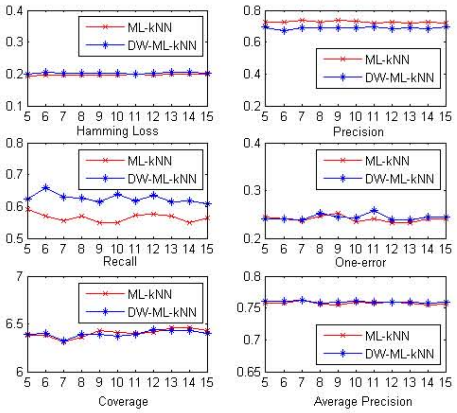


Fig. 2. Comparison results on Yeast dataset

As shown in Fig. 2, the number of nearest neighbors K has small impact on the performance. The reason is that the label cardinality of Yeast dataset is large, data are comparatively balanced and the distances are little changed. When $K = 7$, the performance is approximately up to the best. Recall, Average precision and Coverage of DW-ML-kNN obviously outperform ML-kNN, but Precision and Hamming loss are worse than that of ML-KNN. DW-ML-kNN tends to assign more labels to the instance, resulting in higher recall but lower precision.

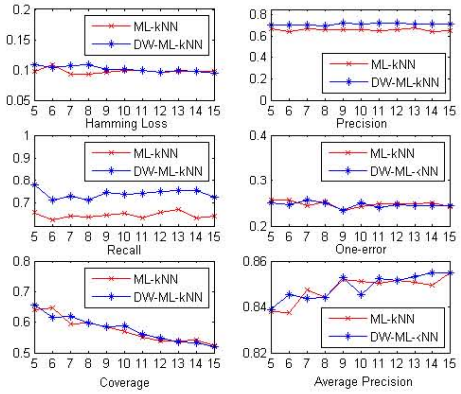


Fig. 3. Comparison results on Scene dataset

Fig. 3 gives the comparison results on Scene dataset. The meaning of the coordinates is the same with that in Fig. 2. We can see that Coverage and Average precision are better when K is larger. K has small impact on other criteria. DW-ML-kNN is better than ML-kNN on Coverage, Average precision, especially on Precision and Recall, but worse on One-error and Hamming loss. The label cardinality of Scene dataset is 1.074, that is, many instances are with the single label. DW-ML-kNN truly predicts the rare labels when the distances of nearest neighbors with these labels are short.

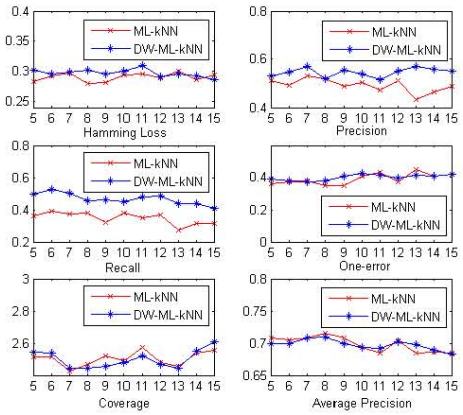


Fig. 4. Comparison results on Emotions dataset

The comparison results on Emotions dataset are described in Fig. 4. The meaning of the coordinates is the same with that in Fig. 2. The performance of DW-ML-kNN on Precision and Recall is better than ML-kNN, but not obviously on other criteria, especially worse on Coverage. The reason is similar to that of Scene dataset.

The overall result is obtained from the above three results. Table 3 reports the average performance of the two algorithms across 11 values of the parameter K for each dataset. The best result on each criterion is shown with bold typeface. The last line means the number of criteria on which each algorithm achieves the best result.

Table 3. Overall Comparison Results of Two Algorithms

Criterion	Yeast		Scene		Emotions	
	ML-kNN	DW-ML-kNN	ML-kNN	DW-ML-kNN	ML-kNN	DW-ML-kNN
Hamming loss	**0.1973**	0.2028	**0.0978**	0.1013	**0.2900**	0.2981
Precision	**0.7273**	0.6880	0.6583	**0.7106**	0.4928	**0.5468**
Recall	0.5653	**0.6275**	0.6471	**0.7625**	0.3510	**0.4781**
One-error	0.2405	**0.2398**	0.2481	**0.2462**	**0.3896**	0.4012
Coverage	6.4015	**6.3956**	**0.5804**	0.5828	2.4980	**2.4888**
Average precision	0.7576	**0.7595**	0.8472	**0.8490**	0.6898	**0.6994**
Win(s)	2(6)	4(6)	2(6)	4(6)	2(6)	4(6)

The results show that DW-ML-kNN is better than ML-kNN on more than half of the six criteria on three benchmark datasets. Average precision and Recall are always improved, while Hamming loss is still worse. The performance on other three criteria is unstable, but close. In a word, the dual weighted ML-kNN (DW-ML-kNN) algorithm achieves promising and better performance.

5. Conclusion

To solve the problem of imbalanced data, we propose a new algorithm-dual weighted multi-label k nearest neighbors, named DW-ML-kNN. The experiment results show that our algorithm is better than ML-kNN. We will carry out research on the criterion of Hamming loss, and try to improve its performance further. What's more, the tuning factor in our algorithm is also a further research topic.

Acknowledgments

The work is partially supported by the National Natural Science Foundation of China (No. 60970061, No. 61075056, and No. 61103067), the Opening Project of Shanghai Key Laboratory of Digital Media Processing and Transmission (No.2011KF03), and the Fundamental Research Funds for the Central Universities.

References

1. R. E. Schapire and Y. Singer. *Boostexter: a boosting-based system for text categorization*, Machine Learning, **39**(2-3), pp. 135-168, 2000.
2. S. Godbole and S. Sarawagi. *Discriminative methods for multi-labeled classification*, In Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 22-30, 2004.
3. Z. H. Wei, H. Y. Zhang, Z. F. Zhang, W. Li and D. Q. Miao. *A Naive Bayesian multi-label classification algorithm with application to visualize text search results*, International Journal of Advanced Intelligence, **3**(2), pp. 173-188, 2011.
4. G. J. Qi, X. S. Hua, Y. Rui, J. H. Tang, T. Mei, and H. J. Zhang. *Correlative multi-label video annotation*, In Proceedings of the 15th International Conference on Multimedia, pp. 17-26, 2007.
5. M. L. Zhang and Z. H. Zhou. *ML-kNN: A lazy learning approach to multi-label learning*, Pattern Recognition, **40**(7), pp. 2038-2048, 2007.
6. X. T. Lin and X. W. Chen. *Mr.KNN: Soft relevance for multi-label classification*, In Proceedings of the 19th ACM International Conference on Information and Knowledge Management, pp. 349-358, 2010.
7. H. Blockeel, L. Schietgat, J. Struyf, S. Dzeroski and A. Clare. *Decision trees for hierarchical multilabel classification: A case study in functional genomics*, In Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, **4213**, Lecture Notes in Computer Science, pp. 18-29, 2006.
8. A. Clare and R. King. *Knowledge discovery in multi-label phenotype data*, In Proceedings of the 5th European Conference on Principles and Practice of Knowledge Discovery in Databases, **2168**, Lecture Notes in Computer Science, pp. 42-53, 2001.
9. G. Tsoumakas, I. Katakis and I. Vlahavas. *Mining multi-label data*, Data Mining and Knowledge Discovery Handbook (2nd edition), Springer, pp. 667-685, 2010.
10. G. Tsoumakas and I. Katakis. *Multi-label classification: An overview*, International Journal of Data Warehousing and Mining, **3**, pp. 1-13, 2007.
11. E. Hüllermeier, J. Fürnkranz, W. W. Cheng and K. Brinker. *Label ranking by learning pairwise preferences*, Artificial Intelligence, **172**(16-17), pp. 1897-1916, 2008.
12. J. Fürnkranz, E. Hüllermeier, E. L. Mencia and K. Brinker. *Multilabel classification via calibrated label ranking*, Machine Learning, **73**(2), pp. 133-153, 2008.
13. M. L. Zhang and Z. H. Zhou. *Multi-label learning by instance differentiation*, In Proceedings of the 22nd AAAI Conference on Artificial Intelligence, pp. 669-674, 2007.
14. J. Read. *A pruned problem transformation method for multi-label classification*, In Proceedings of New Zealand Computer Science Research Student Conference, pp. 143-150, 2008.
15. G. Tsoumakas and I. Vlahavas. *Random k-labelsets: An ensemble method for multilabel classification*, In Proceedings of the 18th European Conference on Machine Learning, pp. 406-417, 2007.
16. A. Elisseeff and J. Weston. *A kernel method for multi-labelled classification*, In Advanced in Neural Information Processing Systems, **14**, pp. 681-687, 2002.
17. F. Thabtah, P. Cowling and Y. H. Peng. *MMAC: A new multi-class, multi-label associative classification approach*, In Proceedings of the 4th IEEE International Conference on Data Mining, pp. 217-224, 2004.

Duoqian Miao

He received the Ph.D. degree in Pattern Recognition and Intelligent Systems from Institute of Automation, Chinese Academy of Science in 1997. He is currently a professor at Tongji University. His research interests include machine learning, rough set theory, granular computing and natural language processing.

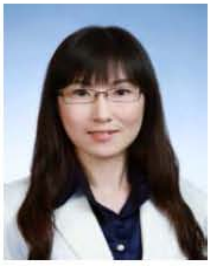
Zhifei Zhang

He received the B.E. degree in Computer Science and Technology from Tongji University in 2008. He is currently a Ph.D. candidate in Pattern Recognition and Intelligent Systems at Tongji University. His research interests include text mining, machine learning and natural language processing.

Zhihua Wei

She received the Ph.D. degree in Pattern Recognition and Intelligent Systems from Tongji University in 2010. She is currently a lecturer at Tongji University. Her research interests include text mining, machine learning and natural language processing.

Chunyan Wang



She received the B.E. degree in Computer Science and Technology from Tongji University in 2009. She is currently a master student in Pattern Recognition and Intelligent Systems at Tongji University. Her research direction is machine learning.