An Effective Tree-Based Algorithm for Ordinal Regression

Fen Xia, Wensheng Zhang, and Jue Wang, *Senior Member, IEEE*

***Abstract*— Recently ordinal regression has attracted much interest in machine learning. The goal of ordinal regression is to assign each instance a rank, which should be as close as possible to its true rank. We propose an effective tree-based algorithm, called Ranking Tree, for ordinal regression. The main advantage of Ranking Tree is that it can group samples with closer ranks together in the process of tree learning. This approach is compared with original decision tree. Experiments on some synthetic and real-world datasets show that Ranking Tree outperforms original decision tree in terms of speed and accuracy as well as robustness.**

***Index Terms*— Machine learning, ranking, decision tree, splitting rule.**

I. Introduction

CONSIDER the following stamp-rating scenario. As a stamp collector, Jack has already collected a lot of stamps in the past few years. However, he is still looking for new stamps. Whenever he gets a stamp, he would need to rate the stamp based on a 1-5 scale, with 5 representing the most valuable collection.

Jack's rating problem can be modeled as a supervised inductive learning task. Two most popular supervised inductive learning methods are classification and regression. In classification, unknown labels are estimated from a set of finite, *unordered* categories. In regression, numeric outputs take continuous values. However, Jack's rating problem cannot be directly solved by either of these two methods because labels in this case are chosen from a set of finite, *ordered* ratings. In the literature, Jack's problem is one that predicts instances of ordinal scale, i.e., the so-called ordinal regression [1].

Applications of ordinal regression frequently arise from domains where human-generated data play an important role. Examples of these domains include information retrieval, collaborative filtering, medicine, and psychology. When people assess objects of interest in these domains (e.g., in terms of their correctness, quality, or any other characteristics), they often resort to subjective evaluation and provide rating information that is typically imprecise. Also, rating results or scores given by different persons are usually not comparable. Therefore, ordinal labels are preferred to continuous scores. In practice, ordinal labels typically correspond to linguistic terms such as "very bad", "good", "very good".

Several approaches have been developed in the machine learning literature to deal with ordinal regression. One obvious idea is to convert the ordinal regression to the regular regression problem. For instance, [2] investigated the use of a regression tree learner by mapping rating results to real values. However, determining an appropriate mapping is often difficult because the true, underlying metric among the ordinal scales is unknown for most tasks. As a result, these regression algorithms are more sensitive to the representation of the ranks rather than the ordinal relationships. Another idea is to convert the ordinal regression to a multi-class classification problem [3]. In these approaches, the ordinal regression problems are converted into nested binary classification problems and the results of these binary classifications are combined to produce for rating prediction. It is also possible formulate the ordinal regression as a large augmented binary classification problem. [1] applied the principle of structural risk minimization to ordinal regression, leading to a new distribution-independent learning algorithm based on a loss function defined on pair items of different ranks. [7] considers general ranking problems in the form of preference judgments and presents a complexity gap between classification and ranking. [8] presents a formal framework for the general ranking problem in the form of preference judgments. However, these approach are time consuming as they operate on pre-processed datasets whose size is quadratic of that of the original dataset. As for on-line learning, [4] and [5] operate directly on ranks by associating each rank with a distinct sub-interval on the real line and those intervals are adapted in the process of learning. [6] generalizes the approach of [4] and [5] to deal with the ranking and re-ranking problem in natural language processing. The ranking algorithm in [6] searches dynamically for pairs of inconsistent objects with different margins and uses them to update the weight vector. Other methods have also been proposed. [9] presents a probabilistic kernel approach to ordinal regression based on Gaussian processes. [10] generalizes the formulation of support vector machines to ordinal regression. [11] uses gradient descent method for learning ranking functions based on the pairs items.

In this paper we develop an alternative approach that uses a decision tree [12], [13], [14] with a suitable splitting rule

This work was supported in part by the National Basic Research Program of China (2004CB318103), National Science Foundation of China (60033020), National Science Foundation (60575001) of China, and Overseas Outstanding Talent Research Program of Chinese Academy of Sciences(06S3011S01).

Fen Xia is with the Key Laboratory of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, P.R. China, and also with the Graduate School, Chinese Academy of Sciences, Beijing, P.R. China (e-mail: fen.xia@ia.ac.cn).

Wensheng Zhang is with the Key Laboratory of complex system and Intelligence Science, Institute of Automation, Chinese Academy of Sciences, Beijing, P.R China (email: wensheng.zhang@mail.ia.ac.cn).

Jue Wang is with the Key Laboratory of Complex Systems and Intelligence Science, Institute of Automation, Chinese Academy of Sciences, Beijing, P.R China(email: jue.wang@mail.ia.ac.cn).

for ordinal regression. As a widely-used data mining and machine learning tool [17], [18], decision trees can achieve good prediction accuracy while producing an easy-to-interpret rule. It can accept continuous, discrete and categorical inputs. It is invariant under strictly monotone transformations of the individual inputs and performs internal feature selection as an integral part of the procedure. Therefore, it is quite desirable to use decision tree for ordinal regression. To our best knowledge, the use of tree learners in ordinal regression is largely under-explored. [2] investigated the indirect use of a regression tree learner to tackle ordinal regression problems. However, their method requires a proper mapping function, which in many cases can only be heuristically defined through trials-and-errors. Another possible use of the tree learner in ordinal regression is to formulate the ordinal regression problem as a multi-class classification problem. As is well known, splitting rule is a growth strategy which guides the learning of the tree. A major problem with this method is that the splitting rule in classification does not take the ordinal relationship into account. The key technical challenge with developing a tree-based ordinal regression method, in our view, is the development of a proper splitting rule that can make use of the ordinal relationship.

The splitting rule is based on the impurity measure of a set. Thus, development of a proper splitting rule is equal to seek a proper impurity measure. We present a new impurity measure motivated by the following intuition. The impurity of a set can be decided by the deviation of sample ratings in the set. A pair of irrelevant items should cause more impurity than a relevant or possibly relevant pair. Likewise, the more pairs with different ratings in a set, the more impure the set will be. We formalize this intuition by developing a new impurity measure on a set.

The reported research is based on this new impurity measure. We use it to construct the splitting rule for the ordinal regression problem. Based on the splitting rule, we train a decision tree, called Ranking Tree. This method is compared with the original classification tree using some synthetic and real-world datasets. Experiments show that Ranking Tree outperforms the classification tree in terms of speed and accuracy as well as robustness.

The remainder of this paper is organized as follows. In Section 2, we present two impurity measures: the gini impurity, a popular measure widely used in the classification tree literature and the base of comparison for our measure; the ranking impurity, our measure proposed in this paper; Section 3 presents a detailed analysis of these two measures. In Section 4, we experimentally compare the Ranking Tree with the classification tree and summarize the results. In Section 5, we conclude the paper and point some possible future research directions.

II. Two Impurity Measures

A. The Gini Impurity

One of the most commonly used impurity measures in classification problems is the gini impurity, defined as follows:

Definition 1: Given a sample set T, let $p_i = p(i|T)$ be the relative proportion of class i samples in the set T, where $i \in \{1, \ldots, k\}$ is the class label; the gini impurity (also known as the gini index) is defined as

$$I_{gini}(T) = \sum_i \sum_j {}_{j \neq i} p_i p_j$$

There are two interpretations of the gini impurity. If a sample belongs to class i with probability p_i, the loss of misclassifying it would be $p_i \sum_{j \neq i} p_j$. Therefore, the expected loss on all classes due to misclassifications is given by $\sum_i \sum_j {}_{j \neq i} p_i p_j$. In the second interpretation, if each sample is coded as 1 for the class i with probability p_i and zero otherwise, the variance of this code variable is $p_i(1 - p_i)$. Summing these variances over all classes produces the gini impurity.

With the impurity measure, sets can be compared. Also, the split associated with sets can be compared. A split is to divide a set T into two sets T_L and T_R, corresponding to the left child and the right child of T respectively. The splitting rule of gini impurity is to find the best split, which is the one that maximizes the quantity defined as

$$\Delta I = I_{gini}(T) - I_{gini}(T_L)p(T_L) - I_{gini}(T_R)p(T_R)$$

This objective can be interpreted as to minimize error of random rule in child nodes.

The gini index is well suitable for standard classification tasks. However, in ordinal regression, the gini index ignores the ordinal relationship among the class labels in that all class labels are treated equally. Furthermore, consider the first interpretation discussed above. Misclassifying a sample from class i to every other class produces an equal portion of loss. This is problematic in ordinal regression because ranking an item further away from its actual rank would be more harmful.

B. The Ranking Impurity

We now present our new impurity measure named ranking impurity.

Definition 2: Given a sample set T labeled by a totally ordered set $L = \{L_1, \ldots, L_k\}$, let $N_i(T)$ be the number of elements in T that have label L_i; the ranking impurity is given by:

$$I_{rank}(T) = \sum_{j=1}^{k} \sum_{i=1}^{j} (j - i) N_j(T) N_i(T)$$

The ranking impurity can be interpreted as the maximum potential number of miss-ranked pairs in the set. Imagine a rater who always makes a mistake when he evaluates a pair of objects. For example, if one sample a_1 belongs to rating L_1, and another sample a_2 belongs to rating L_2, he will always give a wrong order and rank a_1 after a_2. To measure the extent of such mistakes, we weigh the pair by the difference of the ratings, that is, $L_2 - L_1$. Since a set can be decomposed into many pairs, the maximum mistakes that the rater will make are our ranking impurity.

The splitting rule of the ranking impurity is then to find the best split, which is the one that maximizes the quantity defined as

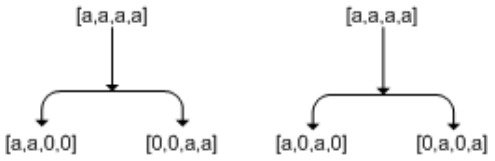

Fig. 1. Two splits of the decision tree.

$$\Delta I = I_{rank}(T) - I_{rank}(T_L) - I_{rank}(T_R) \qquad (1)$$

The objective can be interpreted as to minimize the maximum potential number of miss-ranked pairs in both T_L and T_R.

It is easy to verify that the ΔI in (1) is positive whenever neither T_L nor T_R is empty. So it prevents the creation of degenerate trees.

Roughly speaking, the ranking impurity emphasizes the role of individual samples while the gini impurity emphasizes the role of the individual classes. Meanwhile, the former takes the order relationship into account while the latter not.

III. Ranking Impurity Based Decision Tree Evaluation

In this section, we analyze the ranking impurity and describe its capacity in expressing ordinal relationships.

Consider for instance the two splits in Fig.1.

In both splits, the parent nodes have four ratings (1, 2, 3, 4 with 1 as the first element) and each rating has the same number of a samples. The split in the left tree sends all samples with rating equal 1 and all samples with rating equal 2 to its left child node. Then the remainder is sent to its right child node. On the other hand, the split in the right tree sends all samples with rating equal 1 and all samples with rating equal 3 to its left child node. Then the others are sent to its right child node.

Now we evaluate these two splits using the gini and ranking impurity measures. The child nodes have the same weighted average gini impurity in both splits. In contrast, the left split leaves a ranking impurity of $2a^2$ while the right split $4a^2$. Therefore, ranking impurity prefers the left split to the right split.

Comparing the two splits, we observe that the samples of closer ratings are bundled together in the left split but not in the right split. We omit the theoretical proof due to the lack of space and state that partitioning with rank impurity can group samples with closer ratings together in each splitting step. Consider a case where there are $N_1(T)$ samples of rating 1, $N_2(T)$ samples of rating 2 and $N_3(T)$ samples of rating 3 at a node T. If $N_1(T)$, $N_2(T)$, and $N_3(T)$ are equal, the split with ranking impurity will never separate out those $N_2(T)$ samples of rating 2. On the other hand, since the split with gini impurity ignores the ordinal relationship and it may separate out the samples of rating 2. If $N_2(T) \leq 2N_1(T)$, or $N_2(T) \leq 2N_3(T)$, then the splitting with ranking impurity will avoid separating out the samples of rating 2.

In ordinal regression, it is important to group the samples with closer ratings together in each splitting step for the following reasons: Firstly, it might lead to a fast error convergence rate measured by the deviation from the true rank in the process of the partitioning. Secondly, it provides a robust method to deal with noises. The ratings given by users often contain noise; for instance, the rater often is unsure about which one of the adjacent ratings to assign. Partitions aimed to preserve the ranking of samples may be less affected by these noises than simple partitioning since they would tend to put samples of adjacent ratings together in one node. Computationally, the two impurity measures share the same goal, that is, making the leaf nodes pure. However, the process of splitting can be very different because of the greedy nature of the tree-based algorithms. We argue that splitting with ranking impurity is more suitable than splitting with gini impurity in ordinal regression. The next section reports experimental findings that support this argument.

IV. Experiments and Discussion

To compare the Ranking Tree algorithm with the classification tree algorithm, we use one synthetic dataset and several real-world datasets. In our experiments the CART decision tree algorithm was used, with the splitting rule specified either by the gini or ranking impurity measure. The implementation of CART was based on the *rpart* package in R, which can be found at *http://www.r-project.org*.

A. Evaluation using a synthetic dataset

We generated a synthetic dataset using the same data generation process as specified in [1], [4], [5]. Firstly, we generated random points according to the uniform distribution on the unit square $[0, 1] \times [0, 1]$. Then we assigned each point with the rank chosen from set $\{1, \ldots, 5\}$ using the following ranking rule, $y = \max_r\{r : 10((x_1 - 0.5)(x_2 - 0.5)) + \varepsilon > b_r\}$ where $b = \{-\infty, -1, -0.1, -0.25, 1\}$ and ϵ was normally distributed with zero mean and standard deviation of 0.125. We used the measure, which quantified the accuracy of predictive ordinal ranks with respect to true ranks, i.e., the average rank loss $\frac{1}{T}\sum_{t=1}^{T} |\widehat{y}_t - y_t|$, where T is the number of samples in the test set.

We used 20 Monte-Carlo trials with $50,000$ training samples and a separate test set of $1,000$ samples to compare the performance of the two algorithms in the large training datasets. Cross-validation was used to choose the depth of the tree. Table I shows the results of Ranking Tree and classification tree.

TABLE I
The average rank loss $\frac{1}{T}\sum_{t=1}^{T} |\widehat{y}_t - y_t|$ with their corresponding 95 percent confidence intervals with the Student's t-distribution produced by separate test samples with different algorithm in the synthetic dataset, where T is the test set size. RT refers to our Ranking Tree. CT refers to the Classification Tree. Depth refers to the depth of the tree.

Algorithm	Rank loss
RT with depth = 9	0.16±0.01
CT with depth = 9	0.17±0.01

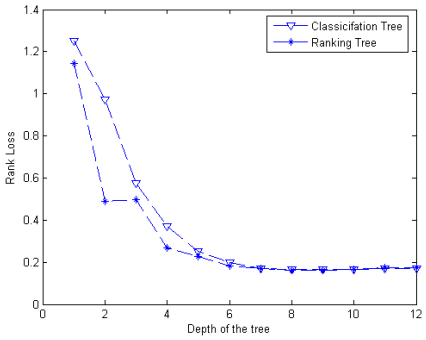

Fig. 2. The average 5-fold cross-validated ranking loss of classification tree and Ranking Tree, with respect to the depth of the tree.

From Table I, we note that the performances of the RT and CT algorithms are very close. It is shown that given enough training samples the RT and CT algorithms can achieve almost the same overall performance.

To compare the convergence rates of the two algorithms, we used $50,000$ training samples and recorded their 5-fold cross-validation results in the process of partitioning. Fig. 2 shows the results of the two algorithms with respect to the depth of the tree. Ranking Tree exhibits a much faster convergence rate than classification tree. This observation supports our analysis, which predicted that Ranking Tree would create better partitions than the classification tree. We also notice the closely-matched performance of classification tree and Ranking Tree as the tree depth increases. We suspect that this is due to the fact that both algorithms are able to find a partition that every node in the tree is very " pure ", resulting in similar performance.

To model noises in the data, we defined a noise level σ , and assumed that each rating could be "misranked" to its adjacent ratings with probability σ.

We used 20 Monte-Carlo trials to test the two algorithms in different size of the training samples. All the results were produced by 5-fold cross validation and were shown in Fig. 3.

From Fig. 3 we can see that Ranking Tree algorithm achieves lower rank loss and delivers much tighter confidence intervals than the classification tree algorithm in all conditions, especially when the size of training samples is small and the noise level is high. This supports our claim that the Ranking Tree is more robust than the classification tree algorithm.

B. Ranking with real-world collaborative filtering datasets

For testing purposes, we chose two real-world collaborative filtering datasets; both of them were used for ordinal regression research [5]: Cystic Fibrosis [15] and MovieLens dataset [16]. The original datasets are composed of the items where each entry is given by a query-document-rating triple. We constructed the dataset in the following way. We randomly chose a target rank y_t on one item and then used the remaining ratings as the dimensions of the instance vector x_t. The detailed experimental setup for each dataset is described below.

The Cystic Fibrosis dataset is a set of 100 queries with the respective relevant documents. There are $1,239$ documents published from 1974 to 1979 discussing Cystic Fibrosis. In each query-document pair, there are three ratings of highly relevant, marginally relevant and not relevant, which we used the ranks of 3, 2, 1 to represent respectively. There are four ratings for each query-document pair. In the end, we have three dimensions of the feature vector and a target rank. The training set and test set sizes were $4,247$ and 572, respectively.

The MovieLens dataset consists of $100,000$ ratings $(1-5)$ from 943 users on $1,682$ movies, with each user rating at least 20 movies. We considered only those people who had rated over 300 movies. There are 54 persons in total in this category; as such the dimension of the instance vectors is 53. Firstly, we randomly chose a target person from the 54 people. Then we looked for the first 300 movies rated by him and formed an instance by using his ratings as the target rank. While doing so, the ratings from the remaining 53 people about the same movie forms the feature vector. If one of those 53 people had not seen a selected movie, we assigned rank 3 to that movie for the people. In this set of experiments, 210 random items were selected to form the training set and the remaining 90 movies served as the test set.

We tested our Ranking Tree and classification tree in the two collaborative filtering datasets. As in the case of the synthetic dataset, We also used the averaged rank loss $\frac{1}{T}\sum_{t=1}^{T}|\widehat{y}_t - y_t|$, where T is the test set size. The results were averaged on 500 Monte-Carlo trials and are given in Table II.

TABLE II

TEST SET PERFORMANCE ON COLLABORATIVE FILTERING. THE PERFORMANCE MEASURE IS THE AVERAGED RANK LOSS $\frac{1}{T}\sum_{t=1}^{T}|\widehat{y}_t - y_t|$, WHERE T IS THE TEST SET SIZE. THE RESULT IS REPRESENTED WITH THEIR CORRESPONDING 95 PERCENT CONFIDENCE INTERVALS WITH THE STUDENT'S T-DISTRIBUTION.

Algorithm	Cystic Fibrosis	MovieLens
RT	0.27±0.00 (Depth = 6)	0.79±0.02 (Depth = 2)
CT	0.39±0.00 (Depth = 4)	0.80±0.02 (Depth = 1)

From Table II we observe that on the Cystic Fibrosis dataset Ranking Tree significantly outperforms the classification tree. Interestingly, on the MovieLens dataset both classification tree and Ranking Tree prefer trees with fewer nodes. Also, it turns out that stumps (trees with depth 1) perform rather well on that dataset. This might imply that if given enough number of recommenders, one's recommendation would nearly always be similar to some other recommender's.

V. CONCLUSION

In this paper, we have presented an effective approach to ordinal regression based on decision tree embedding a new splitting rule based on rank impurity. We have experimentally validated this approach, demonstrating its performance and robustness, relative to an existing approach based on the gini impurity.

Decision tree algorithms have many practical merits. They can handle continuous, discrete and categorical features, fill missing values and select relevant features to produce simple rules. By applying the ranking impurity metric on decision tree, Ranking Tree preserves those merits.

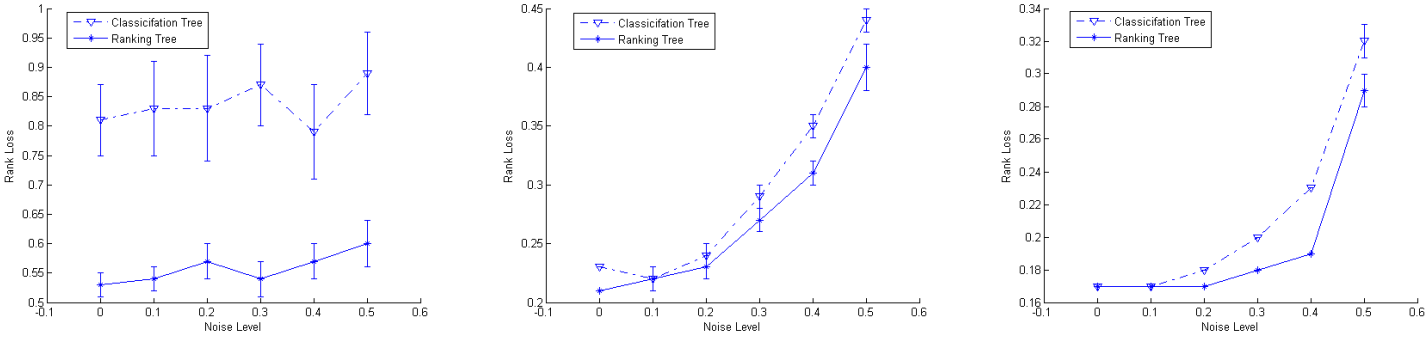

Fig. 3. Learning curves for Classification Tree (dashed-dotted line) and Ranking Tree (solid line) if we measure average rank loss. The error bars indicate the 95 confidence intervals of the estimated rank loss. (a) The size of training set is 100; (b) The size of training set is 1000; (c) The size of training set is 10000.

Decision trees are known to be instable. Techniques like bagging and boosting can be applied to greatly reduce the instability of decision trees. However, as these algorithms were originally defined on the classification or regression case, extending them to the ordinal regression problem will be a challenge. Our current research is addressing this challenge.

The reported work deals with totally ordered ratings only. In many applications, the sample set might have several subsets, with a different order defined on each one. We are working on investigating whether Ranking Trees can be extended to tackle these generalized ordinal regression problems.

REFERENCES

[1] R. Herbrich, T. Graepel, and K. Obermayer “Large margin rank boundaries for ordinal regression,” *Advance in Large Margin Classifiers*, pp 115–132, 2000.

[2] S. Kramer, G. Widmer, B. Pfahringer, and M. DeGroeve, “Prediction of ordinal classes using regression trees,” *Fundamenta Informaticae, 47*, pp. 1–13, 2001

[3] E. Frank and M. Hall, “A simple approach to ordinal classification” *Proceedings of the European Conference on Machine Learning*, pp. 145–165, 2001.

[4] K. Crammer and Y. Singer, “ Pranking with ranking,” *Proceedings of the conference on Neural information Processing Systems(NIPS)*, 2001.

[5] Edward F. Harrington, ”Online Ranking/Collaborative filtering using the Perceptron Algorithm,” *In Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, Washington DC, 2003.

[6] Libin Shen and Aravind K. Joshi, ”Ranking and Reranking with Perceptron,” *Maching Learning*, vol.60, pp. 73-96, 2005.

[7] W.W. Cohen, R.E. Schapire, and Y. Singer, ” Learning to order things,” *Journal of Artificial Intelligence Research(JAIR)*, 10:243-270, 1999.

[8] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer, ”An efficient boosting algorithm for combining preferences,” *Journal of Machine Learning Research*, 4:933-969, 2003.

[9] W. Chu and Z. Ghahramani, ”Gaussian processes for ordinal regression,” *Journal of Machine Learning Research*, 6:1019-1041, 2005.

[10] A. Shashua and A Levin, “Ranking with Large Margin Principle: Two Approaches,” *Proceedings of the conference on Neural information Processing Systems. (NIPS)*14* , 2003.

[11] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton and Greg Hullender, ”Learning to Ranking using Gradient Descent,” *In Proceedings of the 22nd International Conference on Maching Learning(ICML-2005)*, Bonn, Germany, 2005.

[12] Leo Breiman, Jerome H. Friedman, Richard A. Olshen and Charles J. Stone, *Classification and Regression Trees*, Wadsworth, 1984.

[13] J. R. Quinlan, ”Induction of decision trees”, *Machine Learning*, 1:81-106, 1986.

[14] J. R. Quinlan, ”C4.5: Programs for Machine Learning”, Morgan Kaufmann, 1993.

[15] Shaw, W.M, Wood, J.B, Wood, R.E and Tibbo, H.R, *The Cystic Fibrosis Database: Content and Research Opportunities. LISR 13*, pp. 347-366, 1991.

[16] GroupLens Research Project, ”MovieLens data sets,” *http://www.grouplens.org/data/*

[17] Shichao Zhang, Xindong Wu and Chengqi Zhang ”Multi-Database Mining”, *The IEEE Intelligent Informatics Bulletin*, Vol.2, No.1, June, 2003.

[18] Xindong Wu, ”Data Mining: An AI Perspective”, *The IEEE Intelligent Informatics Bulletin*, Vol.4, No.2, Dec, 2004.