Journal of Southeast University (English Edition) Vol. 23, No. 3, pp. 447 – 450 Sept. 2007 ISSN 1003—7985

New rank learning algorithm

Liu Huafu Pan Yi Wang Zhong

(Department of Computer Science and Technology, Changsha University, Changsha 410003, China)

Abstract: To overcome the limitation that complex data types with noun attributes cannot be processed by rank learning algorithms, a new rank learning algorithm is designed. In the learning algorithm based on the decision tree, the splitting rule of the decision tree is revised with a new definition of rank impurity. A new rank learning algorithm, which can be intuitively explained, is obtained and its theoretical basis is provided. The experimental results show that in the aspect of average rank loss, the ranking tree algorithm outperforms perception ranking and ordinal regression algorithms and it also has a faster convergence speed. The rank learning algorithm based on the decision tree is able to process categorical data and select relative features.

Key words: machine learning; rank learning algorithm; decision tree; splitting rule

Recently, the ranking problem has become an important research topic in the field of machine learning. In the task of ranking, the goal is to assign a rank to an instance, which is as close as possible to the instance's true rank. It is pointed out that rank learning is important in our daily lives[1], and it is indigenously difficult. Because of the similarity between ranking and classification, and the similarity between ranking and regression, naturally, researchers on machine learning want to convert rank learning problems to classification problems or regression problems. In the conversion to regression, it is very difficult to decide the real number substitute of an order because the learned rule is too sensitive to the real number representation of the order. In the conversion to classification, the ordering of the classes would be lost if the problem were dealt with as a simple classification problem.

All the algorithms in Refs. [1 – 3] cannot process complex data types with noun attributes, and they need a kernel mapping to map samples to attribute space when the data is linearly inseparable. Just like most kernel methods that are indigenously difficult, it is very difficult to select a good kernel mapping because the result is affected enormously by the kernel function. Ref. [4] focused on rank learning processing using the available tools of machine learning.

This paper presents a rank learning algorithm based on the decision tree[5] to overcome this difficulty. The decision tree is a very useful tool in machine learning and data mining. It brings us great accuracy and at the same time makes it easy to interpret the rules. It can accept continuous, discrete and noun attributes, and is robust in transforming data monotonically. It smartly avoids nonlinear problems by separating the original attribute space. Currently available decision tree methods can only solve classification problems and regression problems, so how to process rank learning using the decision tree becomes a problem. In this paper, we present an algorithm to compute the impurity level of a class in a set and revise the branching rule of a decision tree. As a result, the impurity level of the set and the difference of the middle classes in the set have been reduced with the increment in the number of levels of the tree. Thus, we acquire a decision tree algorithm which can be applied to ranking.

1 Rank Learning Based on Decision Tree

In this section, we present an impurity for rank learning, it can be easily applied to various decision tree learning algorithms.

1.1 Rank impurity

Gini impurity is very suitable for classification learning, but for rank learning, it is not suitable because of the order relations among class labels. Here, we define a new impurity.

Definition 1 Given a sample set T in which the label of every element is selected from a totally ordered set $L = \{L_1, \ldots, L_k\}$, let $N_i(T) = N(L_i \mid T)$ represent the number of elements in the set T which belongs to rank L_i, then rank impurity is defined as

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

Received 2007-05-18.

Foundation item: The Planning Program of Science and Technology of Hunan Province (No. 05JT1039).

Biography: Liu Huafu (1961—), male, associate professor, hfliu9063@163.com.

Class impurity can be interpreted as the class disorder of a set. Suppose that element $a_1 \in T$ belongs to class L_1 and another element $a_2 \in T$ belongs to class L_2. If these two elements are in the same set, then they cause a disorder. The more different the classes, the greater the disorder level. The disorder of one element in the set can be acquired by summing up the disorders caused by this element and all other elements. The rank impurity of set T can then be acquired by summing up the disorders of all the elements.

Having the definition of the class disorder level, the branching rule of the decision tree is to select a separation from all the possible separations to maximize

$$\max\Delta I = I_{\text{rank}}(T) - I_{\text{rank}}(T_L) - I_{\text{rank}}(T_{\text{R}}) \quad (2)$$

Eq. (2) can be interpreted as searching a separation to minimize the sum of impurity of the left child set T_{L} and that of the right child set T_{R}.

1.2 Theoretical proof

This section presents some theoretical bases of rank impurity (1), through these theoretical proofs one can see how to guide the decision tree to finish ranking learning.

Theorem 1 is given in order to prove that the generated decision tree is not degenerated following the branching rule in Eq. (2), that is, for every separation ΔI is nonnegative.

Theorem 1 ΔI in Eq. (2) is nonnegative, ΔI is zero if and only if the impurity level of set T is zero before separation.

Proof Since $N_i(T) = N_i(L_{\text{L}}) + N_i(L_{\text{R}})$,

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

$$\sum_{j=1}^{k}\sum_{i=1}^{j}(j-i)(N_j(L_{\text{L}}) + N_j(L_{\text{R}}))(N_i(L_{\text{L}}) +$$

$$N_i(L_{\text{R}})) \geqslant \sum_{j=1}^{k}\sum_{i=1}^{j}(j-i)(N_j(T_{\text{L}})N_i(T_{\text{L}}) +$$

$$N_j(T_{\text{R}})N_i(T_{\text{R}})) \geqslant I_{\text{rank}}(T_{\text{L}}) + I_{\text{rank}}(T_{\text{R}})$$

Therefore, ΔI is nonnegative, as long as the impurity level of the set is not zero before separation, the separation process will go on until the elements in every set have the same class.

Lemma 1 Let $\boldsymbol{A}_k = (a_{ij})_{k\times k}$, with $a_{ij} = a_{ji} \neq 0$, $a_{ii} = 0$, $i, j \in \{1, 2, \ldots, k\}$, then $\det(\boldsymbol{A}_k) \neq 0$ ($\det(\boldsymbol{A}_k)$ is the determinant of $\boldsymbol{A}_k$).

Proof Let $\boldsymbol{A}_{i1} = \{a_{i1}, a_{i2}, \ldots, a_{i(i-1)}\}$ and $\boldsymbol{A}_{1i} = \{a_{1i}, a_{2i}, \ldots, a_{(i-1)i}\}^{\text{T}}$.

Proof by the mathematical induction: when $i = 2$ $\det(\boldsymbol{A}_2) = -a_{12}a_{21} \neq 0$, suppose that $\det(\boldsymbol{A}_i) \neq 0$, then

$$\det(\boldsymbol{A}_{i+1}) = \det(\boldsymbol{A}_i)\det(a_{i+1,i+1} - \boldsymbol{A}_{i1}\boldsymbol{A}_i^{-1}\boldsymbol{A}_{1i}) \quad (3)$$

Because $a_{i+1,i+1} = 0, \boldsymbol{A}_{i1} = \boldsymbol{A}_{1i}^{\text{T}}$. With Eq. (3) and $\boldsymbol{A}_{i1} = \boldsymbol{A}_{1i}^{\text{T}}$, we have

$$\det(\boldsymbol{A}_{i+1}) = -(\boldsymbol{A}_{i1}\boldsymbol{A}_i^{-1}\boldsymbol{A}_{1i})\det(\boldsymbol{A}_i) \quad (4)$$

Because $\det(\boldsymbol{A}_i) \neq 0$, $\det(\boldsymbol{A}_i^{-1}) \neq 0$. $\boldsymbol{A}_i^{-1}$ is a real symmetric matrix. Apply SVD decomposition on $\boldsymbol{A}_i^{-1}$, $\boldsymbol{A}_i^{-1} = \boldsymbol{U}^{\text{T}}\boldsymbol{\Lambda}\boldsymbol{U}$, provided $\boldsymbol{U}$ is a unitary matrix, $\boldsymbol{\Lambda}$ is a diagonal matrix with real-numbered diagonal elements. With $\boldsymbol{A}_i^{-1} = \boldsymbol{U}^{\text{T}}\boldsymbol{\Lambda}\boldsymbol{U}$, we have

$$\boldsymbol{A}_{i1}\boldsymbol{A}_i^{-1}\boldsymbol{A}_{1i} = (\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{U}\boldsymbol{A}_{1i})^{\text{T}}(\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{U}\boldsymbol{A}_{1i}) \quad (5)$$

It is easy to see $\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{U}\boldsymbol{A}_{1i} \neq \boldsymbol{0}$; therefore, $\det(\boldsymbol{A}_{i+1}) \neq 0$. So $\det(\boldsymbol{A}_k) \neq 0$.

Theorem 2 Given a sample set T, with the label of every element selected from a totally ordered set $L = \{L_1, \ldots, L_k\}$, let $N_i(T) = N(L_i \mid T)$ represent the number of samples with rank label L_i in the set T; s represents one separation; T_{L} represents the set of left node after separation; T_{R} represents the set of right node after separation. If $N_1(T) = N_2(T) = \ldots = N_k(T) = \omega$, and samples can go freely to left node or right node, then the best separation is

$$\max\Delta I(T, s) = I_{\text{rank}}(T) - I_{\text{rank}}(T_{\text{L}}) - I_{\text{rank}}(T_{\text{R}}) \quad (6)$$

and its maximum value can be reached under the following conditions (without losing generality, supposing $N_1(T_L) \neq 0$):

When the number of ranks is even,

$$N_1(T_{\text{L}}) = N_2(T_{\text{L}}) = \ldots = N_{\frac{k}{2}}(T_{\text{L}}) = \omega$$

$$N_{\frac{k}{2}+1}(T_{\text{L}}) = N_{\frac{k}{2}+2}(T_{\text{L}}) = \ldots = N_k(T_{\text{L}}) = 0$$

$$N_1(T_{\text{R}}) = N_2(T_{\text{R}}) = \ldots = N_{\frac{k}{2}}(T_{\text{R}}) = 0$$

$$N_{\frac{k}{2}+1}(T_{\text{R}}) = N_{\frac{k}{2}+2}(T_{\text{R}}) = \ldots = N_k(T_{\text{R}}) = \omega$$

When the number of classes is odd,

$$N_1(T_{\text{L}}) = N_2(T_{\text{L}}) = \ldots = N_{\frac{k\pm1}{2}}(T_{\text{L}}) = \omega$$

$$N_{\frac{k\pm1}{2}+1}(T_{\text{L}}) = N_{\frac{k\pm1}{2}+2}(T_{\text{L}}) = \ldots = N_k(T_{\text{L}}) = 0$$

$$N_1(T_{\text{R}}) = N_2(T_{\text{R}}) = \ldots = N_{\frac{k\pm1}{2}}(T_{\text{R}}) = 0$$

$$N_{\frac{k\pm1}{2}+1}(T_{\text{R}}) = N_{\frac{k\pm1}{2}+2}(T_{\text{R}}) = \ldots = N_k(T_{\text{R}}) = \omega$$

Proof From Eq. (1),

$$I_{\text{rank}}(T) = \sum_{j=1}^{k}\sum_{i=1}^{j}(j-i)N_j(T)N_i(T) \quad (7)$$

$$I_{\text{rank}}(T_{\text{L}}) = \sum_{j=1}^{k}\sum_{i=1}^{j}(j-i)N_j(T_{\text{L}})N_i(T_{\text{L}}) \quad (8)$$

$$I_{\text{rank}}(T_{\text{R}}) = \sum_{j=1}^{k}\sum_{i=1}^{j}(j-i)N_j(T_{\text{R}})N_i(T_{\text{R}}) \quad (9)$$

Because $N_1(T) = N_2(T) = \ldots = N_k(T) = \omega$, and $N_i(L_{\text{L}}) + N_i(L_{\text{R}}) = \omega$ ($\forall i \in \{1, 2, \ldots, k\}$).

Therefore, suppose for any i, $N_i(T_{\text{L}}) = x_i$, then $N_i(T_{\text{R}}) = \omega - x_i$, $0 \leqslant x_i \leqslant \omega$. Substituting Eqs. (7), (8) and (9) into Eq. (2), we obtain

$$\Delta I(T, s) = -\sum_{j=1}^{k}\sum_{i=1}^{j}(j-i)(2x_i x_j - \omega x_i - \omega x_j)$$
$$0 \leqslant x_i \leqslant \omega \qquad (10)$$

This is a continuous quadratic function in a closed interval, so its extreme point can only be seen on the boundary or is a singular point. The Hessian matrix $\boldsymbol{H}$ of function $\Delta I(T, s)$ satisfies the form of lemma 1. Based on lemma 1, $\det(\boldsymbol{H}) \neq 0$ and $\mathrm{tr}(\boldsymbol{H}) = 0$, so $\boldsymbol{H}$ must have positive and negative eigenvalues, so it is an indefinite matrix, and, therefore, it is impossible that the singular point is an extreme point. Eq. (10) obtains its extreme point on the boundary. After adding a boundary condition $x_j = 0$ to Eq. (10), the new form of the function also satisfies the conditions of lemma 1. By repeating this process we can obtain the peaks of the extreme points in the domain. The number of peaks is limited, so the maximum can be found. From theorem 2 we see that if the number of samples of every rank in the set is the same, every step makes the samples of closer ranks even closer.

2 Experimental Results and Discussion

We use the same simulated data as in Ref. [2] to test the algorithm. In the experiment, the CART decision tree algorithm is used, with a different separation rule, namely, Eq. (2) that we have proposed.

First, some random points are generated evenly in a unit square $[0, 1] \times [0, 1]$, then follow the following rules: $y = \max_r \{r: 10(x_1 - 0.5)(x_2 - 0.5) + \varepsilon > b_r\}$, $b = \{-\infty, -1, -0.1, -0.25, 1\}$, ε obeys the normal distribution with average 0 and variance 0.125, every sample point is assigned a class label. The same as in Ref. [2], we do a Monte-Carlo sample collection 20 times, every time using 50 000 samples to train and 1 000 samples to test.

For comparison, we use the same evaluation standard as that of Ref. [2]. The average class deviation loss is $\frac{1}{m}\sum_{t=1}^{m}|\hat{y}_t - y_t|$, where m is the number of test samples. Likewise, we present CART's result regarding the gini impurity level. We use cross testing to choose the depth of the tree. Tab. 1 shows the results of our algorithm, CART's results, and the results in Ref. [2].

Tab. 1 lists the results of every algorithm on the test set of simulated data, using average class deviation loss $\frac{1}{m}\sum_{t=1}^{m}|\hat{y}_t - y_t|$. The 95% confidence interval of the loss has been presented with a student t distribution. As shown in Ref. [2], OAP-VP represents online aggregate prank-voted perception; OAP-Bagg represents online aggregate prank-bagging; OAP-BPM represents online aggregate prank-bayes point machine; τ is the parameter; RT represents our ranking tree; CT represents classification tree; Prank-VP represents prank with voted perception; and WH represents the Widrow-Hoff algorithm.

Tab. 1 Algorithms comparison

Algorithm	Rank loss
WH with $\eta = 0.1$	0.30 ± 0.02
Prank	0.37 ± 0.07
Prank-VP	0.31 ± 0.00
OAP-VP with $\tau = 0.3$	0.32 ± 0.01
OAP-VP with $\tau = 0.6$	0.31 ± 0.02
OAP-VP with $\tau = 0.9$	0.32 ± 0.03
OAP-BPM with $\tau = 0.3$	0.22 ± 0.01
OAP-BPM with $\tau = 0.6$	0.24 ± 0.03
OAP-BPM with $\tau = 0.9$	0.25 ± 0.03
OAP-Bagg with $\tau = 0.3$	0.34 ± 0.01
OAP-Bagg with $\tau = 0.6$	0.32 ± 0.02
OAP-Bagg with $\tau = 0.9$	0.33 ± 0.03
RT(Depth = 9)	0.17 ± 0.01
CT(Depth = 9)	0.18 ± 0.01

The results in Tab. 1 show that our sorting tree algorithm is the most effective on the test set. The results by the kernel mapping algorithm are not so good mainly because it is a difficult matter to choose a good kernel mapping.

For comparison, we also use the realistic data in Ref. [2] to test the algorithm. The data includes two cooperative filtering data sets: Cystic fibrosis[6] and MovieLens[7]. Every piece of the original data consists of three items: query, test, and class. The original data is processed as follows: for every piece of data, a rank is randomly selected from the ranks labeled as rank y_i and the rest of the ranks as eigenvector x_i, to form a sample. The performance of the algorithm is measured by the average of 500 Monte-Carlo sample collections.

It can be seen from the experimental results that on the cystic fibrosis data set, obviously the ranking tree algorithm is better than the rest of algorithms, including the classification algorithm. All the algorithms using kernel mapping give poor results.

The tree classifier is a natural attribute selection method; usually a decision tree uses only a very small part of the attributes to make decisions, so there is a huge difference between the model space of a decision tree and that of the kernel algorithm. We can see from the experiments that the results of the decision tree algorithm are much better than those of the kernel algo-

rithm on many real ranking problems.

3 Conclusion

The effective rank learning algorithm based on the decision tree is presented, according to the new definition of class impurity in this paper. The advantage of the algorithm is proved both in theory and in experiments.

We compared the results to the data provided in Ref. [2]. The results indicate that the ranking tree algorithm is obviously better than the other algorithms in Ref. [2]. In order to show the differences between classification and ranking, we tested the ranking tree and the classification tree respectively. The results indicate that the ranking tree is much more robust than the classification tree and has a faster convergence speed.

A decision tree has many advantages in practice. It can process continuous, discrete and noun attributes; it can process data with part of the attributes missing; it can make decisions using only part of the attributes. By applying a new separation rule, our decision tree has kept the above advantages and can be applied to real ranking tasks.

References

[1] Cohen W W, Schapire R E, Singer Y. Learning to order things[J]. *Journal of Artificial Intelligence Research*, 1999, **10**(5): 243 - 275.

[2] Edward F H. Online ranking/collaborative filtering using the perception algorithm[C]//*Proceedings of the* 20*th International Conference on Machine Learning* (*ICML*-2003). Washington DC, 2003: 115 - 132.

[3] Shen Libin, Aravind K J. Ranking and reranking with perception[J]. *Machine Learning*, 2005, **60**(3): 73 - 96.

[4] Freund Y, Iyer R, Schapire R E, et al. An efficient boosting algorithm for combining preference[J]. *Journal of Machine Learning Research*, 2003, **3**(4): 933 - 969.

[5] Quinlan R. Induction of decision trees[J]. *Machine Learning*, 1986, **18**(1): 81 - 106.

[6] Shaw W M, Wood J B, Wood, R. E, et al. The cystic fibrosis database: content and research opportunities[J]. *Library and Information Science Research*, 1991, **13**(5): 347 - 367.

[7] GroupLens Research Project. MovieLens data sets [EB/OL]. (2005-09-18) [2007-04-20]. http://www.grouplens.org/data.

一种新的排序学习算法

刘华富　潘　怡　王　仲

(长沙大学计算机科学与技术系,长沙 410003)

摘要:为了克服排序学习算法不能处理包括名词性特征的复杂数据类型的局限性,设计一种新的排序学习算法.在决策树学习算法中,采用新的等级不纯度定义,修改决策树的分裂规则,得到具有直观解释的排序算法,并给出了相关理论基础.实验结果表明:排序树的平均等级损失明显优于感知机类算法和序回归类算法,且具有较快的收敛速度.基于决策树的排序学习算法,可以处理名词性数据和选择相关的特征.

关键词:机器学习;排序学习算法;决策树;分裂规则

中图分类号:TP181