On Handling Conflicts between Rules with Numerical Features

Tony Lindgren
Department of Computer and Systems Sciences,
Stockholm University and Royal Institute of Technology
Forum 100
164 40 Kista, Sweden
tony@dsv.su.se

ABSTRACT

Rule conflicts can arise in machine learning systems that utilise unordered rule sets. A rule conflict is when two or more rules cover the same example but differ in their majority classes. This conflict must be solved before a classification can be made. The standard methods for solving this type of problem are to use naive Bayes to solve the conflict or using the most frequent class (CN2). This paper studies the problem of rule conflicts in the area of numerical features. A novel family of methods, called distance based methods, for solving rule conflicts in continuous domains is presented. An empirical evaluation between a distance based method, CN2 and naive Bayes is made. It is shown that the distance based method significantly outperforms both naive Bayes and CN2.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms

Keywords

Rule Learning, Rule conflicts, Numerical features

1. INTRODUCTION

When inducing rules to create a classifier, two major strategies are used: Separate-And-Conquer (SAC) [7] or Divide-And-Conquer (DAC) [13].

One major difference between SAC and DAC is that the former can induce rules that overlap. DAC repeats two steps until a stop criterion is meet, the steps are the division of the examples and the conquering of these new subgroups of examples, hence the name Divide-And-Conquer, (DAC is also known as Top-Down Induction of Decision Trees (TDIDT)). As each attribute is exhaustively explored (in the case of a discrete attributes) at every step of the induction process, the leaves contain mutually exclusive example sets. Hence it is not possible for an example belong to multiple rules (leaf nodes) at the same time.

Top-Down SAC starts with an overly general rule (i.e., a rule that cover all examples) and selects one value of an attribute at each refinement step, until a stopping criterion is met. The induction process then starts again with the overly general rule. Repeated restarts with the overly general rule make it possible for different rules to cover the same examples, which in turn can give rise to rule conflicts. Rule conflicts are when two or more rules cover the same example that is to be classified, and the rules have different majority classes. This type of conflicts has not received very much attention in the past, but resent work have addressed the issue of rule conflicts, see [4, 11, 10, 9].

In this paper we have addressed the problem of rule conflicts in the special setting of having only continuous values in the attributes of the examples (i.e., numerical features) but still trying to predict a class. We will refer to this setting as the continuous setting in the rest of the paper.

To solve the problem of rule conflicts in the continuous setting, a novel family of methods that utilises one-dimensional distances, are introduced in this paper. The single best method from this family is compared with existing standard methods for solving rule conflicts, and it is empirically shown that the new method, with statistical significance, outperforms the standard methods.

The rest of this paper is organised as follows. Section 2 is devoted to the detailed explanation of the methods studied in this paper: the novel family of distance based methods, naive Bayes and CN2. In section 2 a comparison between the different distance based methods is also provided. In section 3 the experimental setting is described and in the section 4 the experimental results are presented. Finally, in the section 5 a discussion about the work is held and pointers to future work are given.

2. RULE CONFLICT RESOLVING METHODS

In Figure 1, an example scenario shows five rules that cover examples from two classes, the positive (+) and the negative class (-). Three of these rules are overlapping, namely R1, R2 and R3. The two former rules have the positive class as their majority class, while the latter has

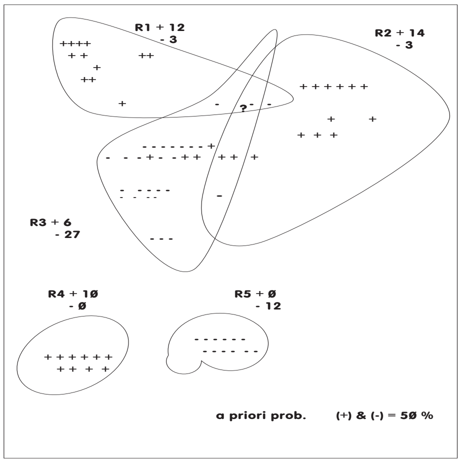

Figure 1: Three rules covering an example to be classified (marked with '?'). The training examples are labelled with their respective classes (+ and $-$).

the negative class as the majority class. In the intersection of the three rule is a question mark, this denotes an example that we want to classify. The question now is how should we classify it? This is the question that we are addressing in this paper, within the scope of continuous setting.

The standard methods for solving rule conflict is the method used in the machine learning system CN2 [2, 1] and the use of naive Bayes. Both of these methods can be used in all settings where rule conflicts can occur, not just the special setting considered in this paper.

The distance based methods that we propose in this paper is restricted to be used in the setting when only considering examples with numerical features. This is because they utilise the distances of these numerical features for classifying examples in conflict. This of course restricts these methods to be used only in such domains, but this restriction may be rewarded by better predictions.

In the following sections, all rule resolving methods are described in detail: CN2, naive Bayes and the novel distance based family of methods.

2.1 CN2

The system CN2 resolves classification conflicts between rules in the following way. Given the examples in Figure 1, the class frequencies of the rules that cover the example to be classified (marked with '?') are calculated:

$$C(+) = covers(R_1, +) + covers(R_2, +) + covers(R_3, +) = 32$$

$$C(-) = covers(R_1, -) + covers(R_2, -) + covers(R_3, -) = 33$$

Where $covers(R, C)$ gives the number of examples of class C that are covered by Rule R. This means that CN2 would classify the example as belonging to the negative class ($-$). More generally:

$$CN2 = argmax_{C_i \in Classes} \sum_{j=1}^{|CovRules|} covers(R_j, C_i)$$

Where $CovRules$ is the set of rules that cover the example to be classified, and $covers$ is the function defined above.

2.2 Naive Bayes classification

In the machine learning system of Rule Discovery System (RDS) [14], naive Bayes is used to resolve rule conflicts. Bayes theorem is as follows:

$$P(C|R_1 \wedge \ldots \wedge R_n) = P(C)\frac{P(R_1 \wedge \ldots \wedge R_n|C)}{P(R_1 \wedge \ldots \wedge R_n)}$$

Where C is a class label for the example to be classified and $R_1 \ldots R_n$ are the rules that cover the example. As usual, since $P(R_1 \wedge \ldots \wedge R_n)$ does not affect the relative order of different hypotheses according to probability, it is ignored. Assuming (naively) that $P(R_1 \wedge \ldots \wedge R_n|C) = P(R_1|C) \ldots P(R_n|C)$, the maximum a posteriori probable hypothesis (MAP) is:

$$h_{MAP} = argmax_{C_i \in Classes} P(C_i) \prod_{Rj \in Rules}^{|CovRules|} P(R_j|C_i)$$

Where $CovRules$ is the set of rules that covers the example to be classified. If we again consider the example shown in Figure 1, we get:

$$P(+|R_1 \wedge R_2 \wedge R_3) = P(+) * P(R_1|+) * P(R_2|+) * P(R_3|+) =$$

$$40/80 * 12/40 * 14/40 * 6/40 = 0.0079$$

$$P(-|R_1 \wedge R_2 \wedge R_3) = P(-) * P(R_1|-) * P(R_2|-) * P(R_3|-) =$$

$$40/80 * 3/40 * 3/40 * 27/40 = 0.0019$$

This means that the naive Bayes classification results in that the example with the unknown class label is classified as positive (+).

Note that if a rule involved in a conflict does not cover any examples of a particular class, this would eliminate the chances for that class to be selected, even if there are several other rules that cover the example with a high probability for that class. To overcome this problem, Laplace-1 correction (described in [8]) is used in the experiments.

2.3 Distance based methods

Distance based models utilise the difference between each rule's values (conditions) and the values of the example under consideration, for each specific attribute. Each rule is assigned a distance value. This value is then used together with the estimated class probability (ECP) of each rule in a weighted voting scheme to solve the conflict.

In a pre-processing step all training examples are examined to establish the maximum and the minimum values for each attribute, these values are then saved for each domain. Using the minimum and maximum value for an attribute a scale for the attribute can be obtained, $Scale(Attribute) = Max(Attribute) - Min(Attribute)$.

By using this scale to normalise the values, the values can be compared with each other. Some domains also contain negative values that is why the subtraction of Min in the normalisation step needed. $Normalised(Value, Min, Scale) = (Value - Min)/Scale$. Here the $Value$ is either the value of a condition or the value of the example to be classified. The difference between the normalised conditions of the rule and the normalised values of the example are then used to get a distance value for each rule.

In our study, we tested three different methods for assigning distance values. These methods are called Voting Mean,

Table 1: A framework for the distance based methods.

```
Input:  R = rules in conflict,
example = example to be classified,
MIN = List of all attributes minimum values,
MAX = List of all attributes maximum values.
Output:  C = a class assigned to example


For each Rule in R
    For each Cond in Rule
        Get the Min value for Cond from MIN.
        Get the Max value for Cond from MAX.
        Scale = Max - Min
        Nor.Cond = normalise(Cond,Min,Scale)
           Get the example Value for Cond.
           Nor.Value = normalise(Value,Min,Scale)
           Nor.Dist = abs(Nor.Cond - Nor.Value)
           Update Mean, Best, and Worst Values.
        Use Nor.Dist in Mean, Best or Worst
        fashion together with the Maximum ECP
        of the rule to get RuleValue.
        Add RuleValue to Rulevalues.
    C = weighted_voting(RuleValues)
```

Voting Best and Voting Worst. Voting Mean takes the average from all (as many as the conditions in the rule) the normalised distances by each rule to the example to be classified. Together with the ECP of each rule the conflicting rules vote for their majority class according to their weights (i.e. distance * ECP), the votes for each class is summed up, the class which gets the highest vote are chosen for classification.

The machine learning system that we used in our experiments is called Rule Discovery System (RDS). One thing to note about the ECP values set by RDS is that they are set using a separate validation set. Hence this set has not been used to induce the rules, and because of this the ECP values can be regarded as quite reliable.

The voting scheme for the other two methods, Voting Best and Voting Worst, are the same as for Voting Mean Voting, but the distance used in the voting process are derived from the conflicting rules in another way. Voting Best selects a single condition for each rule in conflict, namely the condition that has the highest distance of all conditions. Similarly, Voting Worst selects the single condition with lowest distance for each rule in conflict.

All three methods are different in their way of how they try to asses the quality of the prediction of each rule (the weight of the vote). Voting worst is most conservative, as each vote is based on the single worst distance. Voting Mean consider all distances of each rule. While Voting Best consider only the single best distance value of each rule in the votes, this is quite opposite to the conservative approach of Voting Worst. Voting Mean is in-between these two extremes. Table 1 shows a framework for the distance-based methods.

For an illustrative example of the three different methods, consider the scenario in Figure 1 where three rules are in conflict. The estimated class probabilities are the following:

$ECP(R1,+) = 12/15 = 0.80$, $ECP(R1,-) = 3/15 = 0.2$
$ECP(R2,+) = 14/17 = 0.82$, $ECP(R2,-) = 3/17 = 0.18$
$ECP(R3,+) = 6/33 = 0.18$, $ECP(R3,-) = 27/33 = 0.82$

Let us consider a scenario where the examples have four different attributes and a class label. The example denoted by the question mark (?) has the following value, $example(A1 = 12, A2 = -0.5, A3 = 90, A4 = 8.5, Class = ?)$. Where the scales for the different attributes are:
$Scale(A1) = 50 - 0 = 50$, $Scale(A2) = 50 - (-50) = 100$
$Scale(A3) = 100 - 0 = 100$, $Scale(A4) = 10 - 0 = 10$

The normalised values for the example becomes:
$Normalised(A1, Min_{A1}, Scale_{A1}) = (12 - 0)/50 = 0.24$
$Normalised(A2, Min_{A2}, Scale_{A2}) = (-0.5 - (-50)/100 = 0.495$
$Normalised(A3, Min_{A3}, Scale_{A3}) = (90 - 0)/100 = 0.9$
$Normalised(A4, Min_{A4}, Scale_{A4}) = (8.5 - 0)/10 = 0.85$

Assume that the rules have the following conditions:
$R1 = A1 < 20 \wedge A2 < 0$
The normalised conditions for R1:
$Normalised(A1, Min_{A1}, Scale_{A1}) = (20 - 0)/50 = 0.4$
$Normalised(A4, Min_{A2}, Scale_{A2}) = (0 - (-50))/100 = 0.5$
$R2 = A1 < 14$
The normalised conditions for R2:
$Normalised(A1, Min_{A1}, Scale_{A1}) = (14 - 0)/50 = 0.28$
$R3 = A3 > 84 \wedge A4 > 6$
The normalised conditions for R3:
$Normalised(A3, Min_{A3}, Scale_{A3}) = (84 - 0)/100 = 0.84$
$Normalised(A4, Min_{A4}, Scale_{A4}) = (6 - 0)/10 = 0.6$

Using the above values Voting Mean would produce the following classification:
$MeanValue(R1) = (abs(0.4 - 0.24) + abs(0.5 - 0.495))/2 = 0.0825$
$MeanValue(R2) = abs(0.28 - 0.24)/1 = 0.04$
$MeanValue(R3) = (abs(0.84 - 0.9) + abs(0.6 - 0.85))/2 = 0.155$
$Voting(+) = 0.8 * 0.0825 + 0.82 * 0.04 + 0.18 * 0.155 = 0.1267$
$Voting(-) = 0.2 * 0.0825 + 0.18 * 0.04 + 0.82 * 0.155 = 0.1508$
Thus the negative class would be used for classification.

Using the above values Voting Best would produce the following classification:
$BestValue(R1) = abs(0.4 - 0.24) = 0.16$
$BestValue(R2) = abs(0.28 - 0.24)/1 = 0.04$
$BestValue(R3) = abs(0.6 - 0.85) = 0.25$
$Voting(+) = 0.8 * 0.16 + 0.82 * 0.04 + 0.18 * 0.25 = 0.1648$
$Voting(-) = 0.2 * 0.16 + 0.18 * 0.04 + 0.82 * 0.25 = 0.2352$
Thus the negative class would be used for classification.

Using the above values Voting Worst would produce the following classification:
$WorstValue(R1) = abs(0.5 - 0.495)/2 = 0.05$
$WorstValue(R2) = abs(0.28 - 0.24)/1 = 0.04$
$WorstValue(R3) = abs(0.84 - 0.9)/1 = 0.06$
$Voting(+) = 0.8 * 0.05 + 0.82 * 0.04 + 0.18 * 0.06 = 0.0836$
$Voting(-) = 0.2 * 0.05 + 0.18 * 0.04 + 0.82 * 0.06 = 0.0664$
Thus the positive class would be used for classification.

2.3.1 *Comparison of the weighting schemes*

The assumption that lies behind all the three distance based methods presented is that the further the distance, the more likely the rule. To see if this assumption is valid or not we tested the three different methods, Voting Mean, Voting Best and Voting Worst, on eleven domains, exploring the correlation between the distance and the probability (on an interval) of a correct classification.

Figure 2 shows the correlation between the probability of a correct classification (the y-axis) and the normalised distances between the conditions of the tree methods and the classified example (the x-axis). Figure 2 shows the average values from all the domains run (11 domains) the probability values for a correct classification (y-axis) are produced by splitting the classifications into intervals based on their normalised distances.

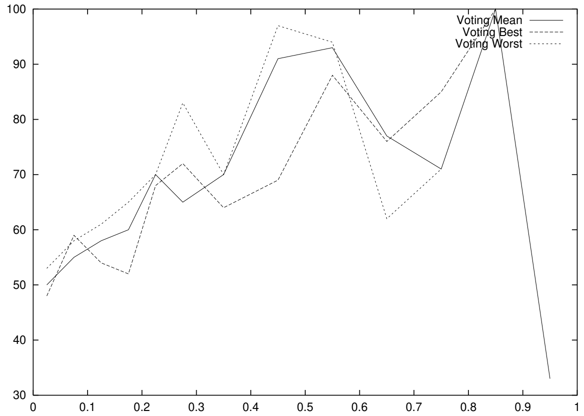

Figure 2: The y-axis shows the probability of a correct classification and the x-axis shows the normalised distance.

The three methods share the same basic shape of their curves. The overall shape is starting from 50 percent probability of a correct classification at the first interval of the x-axis. The probability then rises (with a few drops) until its peak around a normalised distance of 0.45 to 0.55 here the probability for a correct classification is around 85 to 95 percent. Then the probability for a correct classification drops between 60 to 75 percent within the normalised values 0.65 to 0.75. Then the probability of a correct classification picks up again to reach 100 percent at 0.85 of the normalised value, and finally with higher x-values the probability drops dramatically.

This shape thus supports the assumption made, i.e. that a greater distance results in a higher probability of a correct classification.

To get a more comprehensive view of the methods Figure 3 presents the number of examples in each interval of the x-axis (the normalised differences between the rules conditions and the examples). The figure is based on all the eleven domains examined. Both the total number of examples and the number of correctly classified examples are showed at the y-axis in Figure 3. Here the number of examples are great at low distances and drops the greater the normalised difference.

If we have Figure 3 in mind and look at Figure 2 again we see that the method that dominates the other methods up until the interval 0.5 - 0.6 on the x-axis, is Voting Worst. This is important because the amount of examples that are on the left of this interval, (including the examples in the interval), account for 99 percent of all examples in the Voting

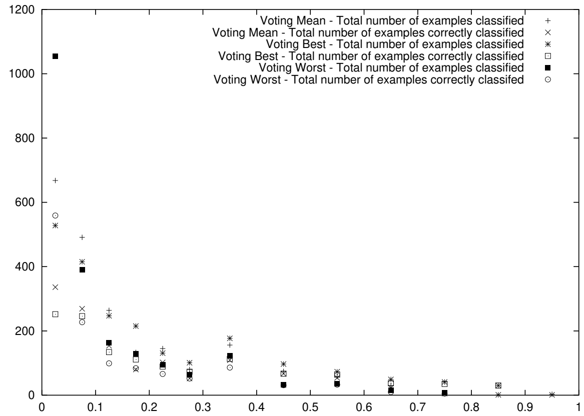

Figure 3: The x-axis shows the number of examples and the y-axis shows the normalised distance.

Worst case. The same figure for Voting Best is 93 percent and for Voting Mean 98. In this light it seems clear as which of the methods to choose as the basis for a comparison of other baseline methods.

3. EXPERIMENTAL SETTING

Ten-fold cross validation was used in the experiments. Exactly the same conflicts where solved by all the different methods. Information gain was used as the search heuristic. Incremental reduced error pruning are used to prune the rules [6]. The machine learning system used in the experiment is RDS. RDS incrementally (during the learning process) introduces cut-points in the numerical attributes to create discrete attributes. RDS usually does this in the standard fashion, i.e. creates cut points in the middle of two examples of different classes [5]. During our experiments we changed this behaviour, with the aim of increasing the number of conflicts between the rules, by creating cut points in-between all examples.

4. EXPERIMENTAL RESULTS

Table 2: Comparing wins and losses for all domains

	naive Bayes	CN2	Voting W.
naive Bayes	-	5,2	1,9*
CN2	2,5	-	1,8*
Voting W.	9,1*	8,1*	-

In Table 2, a comparison of the number of wins and losses in terms of higher or lower accuracy between the three methods is given. For each row in the table, the first digit denote the number of wins against the method in the column and the second digit denote the number of losses against the method in the column. If an asterisk (*) is shown after these figures, it denotes a statistical significant difference (P-value lower than 0.05) according to a sign test. Here we see that Voting Worst performs significantly better than both naive Bayes and CN2. But no significant difference is noted between CN2 and naive Bayes, even though naive Bayes wins five times and only looses two times.

Table 3: Accuracies of the different rule conflict resolving methods.

Domain	naive Bayes	CN2	Voting Worst	Upper-bound
Breast cancer wis.	95.44	94.37	96.38	0.12
Bupa	67.20	65.29	68.47	0.50
Cleveland heart disease	57.25	57.25	57.61	0.33
Glass	89.22	89.22	89.22	0.0
Haberman	74.19	74.55	74.55	0.10
Image segmentation	76.96	76.96	76.96	0.19
Ionosphere	90.31	87.50	86.56	0.24
Iris	83.21	83.21	83.94	0.05
New thyroid	88.76	88.7	89.80	0.05
Sonar	72.11	72.63	75.26	0.32
Spectral flare	76.10	75.47	76.73	0.20
Wine	87.04	86.42	87.65	0.06

When analysing the results, it seems clear that Voting Worst method is performing good. It is only in one domain that it has lower accuracy than both naive Bayes and CN2. In all other cases it has higher or equal accuracy. It is worth noting that the upper-bound differs a lot in different domains from 50 percent to 0 (no conflicts).

In Table 3, the accuracy of the methods on each domain is presented. In the first column, the name of the data set is written. In the second column, the accuracy for naive Bayes is presented. Columns three and four present the accuracy for CN2 and Voting Worst respectively. The last column shows the upper-bound for improvement (in percent), i.e. the number of conflicts in a domain.

5. CONCLUSIONS

In this article, we have presented a novel family of methods that solve rule conflicts in continuous setting. The most promising of these methods was compared to two standard methods. In the comparison it was shown, with statistical significance, that the novel method outperforms the standard methods.

The novel family of methods that utilises the distances to solve rule conflicts is shown to outperform the usual methods for solving this type of conflict. One of the drawbacks of the method is of course that it is restricted to numerical features. The computational complexity is approximately the same as naive Bayes and CN2.

An area that has received quite a lot of attention is discretization. For a nice overview see [3, 12]. The reason for the attention is because some machine learning algorithms require discretization of the continuous attributes, either as a pre-processing step of the data or incrementally during learning.

One possible future application of the distance-based methods is to use the measure for discretization purposes. By using a distance based method the split points could be placed not only where the examples change class (as is usually done), but where there is a "big" distance between the examples. This could hopefully aid in handling outliers.

6. ACKNOWLEDGEMENTS

The author would like to thank Henrik Boström for his help with this paper.

7. REFERENCES

[1] P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Proceedings of the Fifth European Working Session on Learning*, pages 151–163, Berlin, 1991. Springer-Verlag.

[2] P. Clark and T. Niblett. The CN2 Induction Algorithm. *Machine Learning, 3, 261-283*, 1989.

[3] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *International Conference on Machine Learning*, pages 194–202, 1995.

[4] Tom Fawcett. Using rule sets to maximize roc performance. In *ICDM*, pages 131–138, 2001.

[5] U. M. Fayyad and K. B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8:87–102, 1992.

[6] J. Fürnkranz and G. Widmer. Incremental Reduced Error Pruning. In *Proceedings of the 11th International Conference on Machine Learning*, 1994.

[7] Johannes Fürnkranz. Separate-and-Conquer Rule Learning. *Artificial Intelligence Review*, 1999.

[8] R. Kohavi, B. Becker, and D. Sommerfield. Improving simple Bayes. In *Proceedings of the European Conference on Machine Learning*, 1997.

[9] T. Lindgren and H. Boström. Resolving rule conflicts with double induction. *Intelligent Data Analysis - An International Journal, Volume 8, Number 5*, 2004.

[10] Tony Lindgren. Methods for Rule Conflict Resolution. In *Proceedings of the 15th European Conference on Machine Learning (ECML-04)*, pages 262–273. Springer, 2004.

[11] Tony Lindgren and Henrik Boström. Classification with Intersecting Rules. In *Proceedings of the 13th International Conference on Algorithmic Learning Theory (ALT'02)*, pages 395–402. Springer, 2002.

[12] Huan Liu, Farhad Hussain, Chew Lim Tan, and Manoranjan Dash. Discretization: An enabling technique. *Data Min. Knowl. Discov.*, 6(4):393–423, 2002.

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

[14] RDS. Rule Discovery System (RDS) – 1.0, Compumine AB, 2003. www.compumine.com.