A hyper-heuristic for descriptive rule induction

Tho Hoan Pham[1] and Tu Bao Ho[2]

[1]*Hanoi University of Education, 136 Xuan-Thuy, Cau-Giay, Hanoi, Vietnam*
[2]*Japan Advanced Institute of Science and Technology, Nomishi, Ishikawa, Japan*

Email: hoanpt@dhsphn.edu.vn, bao@jaist.ac.jp

ABSTRACT

Rule induction from examples is a machine learning technique that finds rules of the form *condition* → *class*, where *condition* and *class* are logic expressions of the form $variable_1 = value_1 \wedge variable_2 = value_2 \wedge \ldots \wedge variable_k = value_k$. There are in general three approaches to rule induction: exhaustive search, divide-and-conquer, and separate-and-conquer (or its extension as weighted covering). Among them, the third approach, according to different rule search heuristics, can avoid the problem of producing many redundant rules (limitation of the first approach) or non-overlapping rules (limitation of the second approach).

In this paper, we propose a hyper-heuristic to construct rule search heuristics for weighted covering algorithms that allows producing rules of desired generality. The hyper-heuristic is based on a *PN*-space, a new ROC-like tool for analysis, evaluation and visualization of rules. Well-known rule search heuristics such as entropy, Laplacian, weight relative accuracy, and others are equivalent to ones proposed by the hyper-heuristic. Moreover, it can present new non-linear rule search heuristics, some are especially appropriate for description tasks. The non-linear rule search heuristics have been experimentally compared with others on the generality of rules induced from UCI datasets and used to learn regulatory rules from microarray data.

INTRODUCTION

Rule induction from examples is a machine learning technique that has been successfully used as a support tool for knowledge acquisition and prediction. The induced rules are usually expressed as *condition* → *class*, where *condition* and *class* are logic expressions of the form ($variable_1 = value_1 \wedge variable_2 = value_2 \wedge \ldots \wedge variable_k = value_k$.

There are three kinds of rule inducting algorithms: association rule-based, decision tree-based, and covering. The first ones, association rule-based algorithms, use an *exhaustive search* strategy by exploring almost the whole search space (Kavsek *et. al*, 2003; Liu *et. al*, 1998). The basic idea is to use an association rule algorithm to gather all rules that predict the class attribute and also pass a minimum quality criterion. The seconds, decision tree-based algorithms, use a *divide-and-conquer* strategy (Quinland, 1986, 1987). Much of the popularity of these algorithms stems from their efficiency in learning and classification. A decision tree can be easily turned into a rule set by generating one rule for each path from the root to a leaf. Finally, covering algorithms make use of a *separate-and-conquer* strategy over the search space to learn a rule set (see Furnkranz, 1999 for an overview). This *separate-and-conquer* strategy searches for a rule that explains (covers) part of its training instances, separates (or reassigns with lower weight) these examples, and recursively conquers the remaining examples by learning more rules until no examples remain.

In all rule induction systems, rule evaluation measures are an important factor. Rule evaluation measures are functions with which we can infer how "good" or "interesting" a rule is when comparing to others. They are used as a rule search heuristic in learning algorithms (in this case they were referred to *search heuristics*), as well as for filtering out uninteresting rules and/or as a stopping criterion of the refinement process (referred to *filtering heuristics*). Many measures have been proposed in the literature: accuracy, entropy (Clark and Nibblet, 1989), Laplace (Clark and Boswell, 1991), *m*-estimate (Cestnik, 1990), weighted relative accuracy (Lavrac *et. al*, 2004), etc. However, they are given by different authors without any systematic guidance. The generality of induced rules is different according to different heuristics. Therefore, deciding which one is the best suited measure, especially for description tasks, is still an open problem (Furnkranz and Flach, 2005; Lavrac *et. al*, 2004).

In this paper, we propose a way (hyper-heuristic) to form search heuristics. Our hyper-heuristic is based on *PN*-space, a ROC-like tool for analysis, evaluation and visualization of rules. We will show that our hyper-heuristic can form linear heuristics that are equivalent to some existing search heuristics (i.e. having the same rule finding ability in induction systems). Besides these, it can be used to discover new search heuristics including non-linear ones.

BACKGROUND

Definitions

Let P and N be the total number of positive and negative examples in a training set, while $p(r)$ and $n(r)$ denote the respective number of examples covered by a rule r.

Definition 1: A rule evaluation measure of the rule r is a two-dimensional function of the form $h(p(r), n(r))$. For clarity, we will abridge $h(p(r), n(r))$ as $h(r)$, and omit the argument (r) from functions p, n, and h when it can be clearly deduced from the context.

Definition 2: A PN-space (coverage space) is a two-dimensional space of points (n, p), where $0 \le n \le N$ denotes the number of negative examples covered by a rule (false positives) and $0 \le p \le P$ denotes the number of positive examples covered by the rule (true positives).

PN-space is quite similar to ROC space, a two-dimensional plane in which the operating characteristics of classifiers are visualized (Furnkranz and Flach, 2004). Each rule is represented by a point in the PN-space, with the point $(0, P)$ (corresponding to the rule that covers all positive examples and none of the negative) being an ideal point that every learning system tries to reach.

Definition 3: An isometrics of a measure h is a line (or curve) in *PN*-space that connects, for some value c, all points for which $h(p, n) = c$. In other words, an isometric of a measure h is an equivalent class in the partition of *PN*-space based on measure h; all points in an isometric have the same value respect h; and an isometric divides the *PN*-space into two parts: one is above the isometric (points in this part have greater value than those in the isometric), and the other is under the isometric (with points having lesser value than those in the isometric). The former corresponds to rules better than rules in the isometric and the latter corresponds to worse ones with respect to the measure h.

Some Existing Measures and Their Isometrics

The following are some basic measures in the literature that are to use in search or filtering heuristics. Other measures can be found in (Furnkranz and Flach, 2005).

Accuracy (Furnkranz and Flach, 2003)	$h_{acc}(\mathrm{r}) = p - n$	(1)
Weighted relative accuracy (Lavrac *et. al*, 2004)	$h_{wra}(r) = \frac{p+n}{P+N}\left(\frac{p}{p+n} - \frac{P}{P+N}\right)$	(2)
Entropy (Furnkranz and Flach, 2003)	$h_{en}(r) = -\left(\frac{p}{p+n}\log_2\frac{p}{p+n} + \frac{n}{p+n}\log_2\frac{n}{p+n}\right)$	(3)
Laplacian (Clark and Boswell, 1991)	$h_{lap}(r) = \frac{p+1}{n+p+2}$	(4)
m-estimate (Clark and Boswell, 1991)	$h_m(r) = \frac{p + m\frac{P}{P+N}}{p+n+m}$	(5)
Confidence/precision	$h_{conf}(r) = \frac{p}{n+p}$	(6)
Positive_support	$h_{p\_\sup p}(r) = p$	(7)
Support	$h_{\sup p}(r) = p + n$	(8)

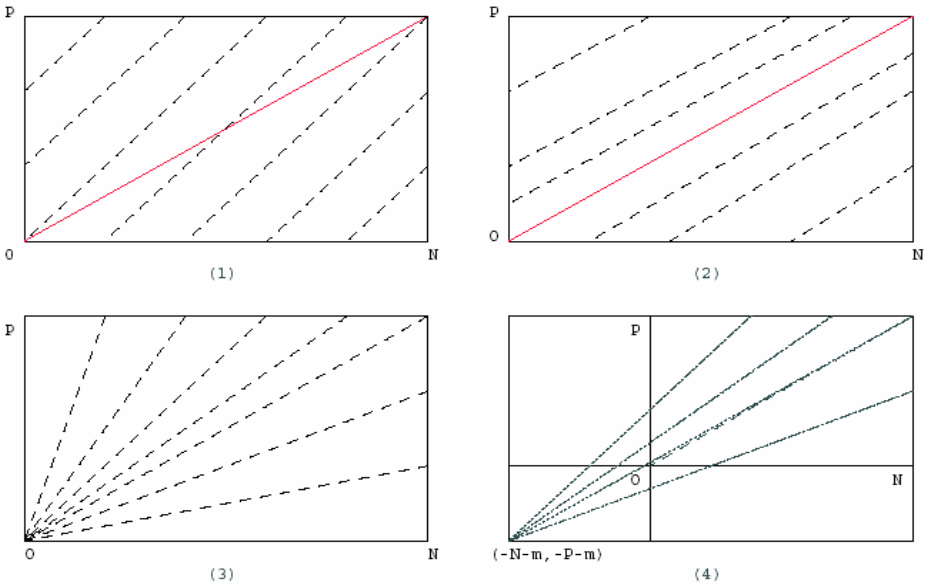

Figure 1 Isometrics of (1) h_{acc}, (2) h_{wra}, (3) h_{en}, and (4) h_m

The isometrics of the first four measures are shown in Figure 1. The measure *confidence* has isometrics similar to measure *entropy* (Figure 1 (3)).

As can be seen in the figure, measures *confidence*, *entropy* (their isometrics are lines from the original of *PN*-space) and *accuracy* (its isometrics are lines that always form an angle 45^o with the axis *ON*) prefer rules with the low covering ($p+n$), while *weighted relative accuracy*-*WRA* (its isometrics are lines parallel with the main diagonal) prefer rules that is a trade-off between the covering and confidence. Measure *m-estimate* is more flexible than *entropy* and *WRA* according to value of the parameter *m*. When $m = 0$, *m-estimate* would become equivalent to *entropy* and when $m = \infty$, it is equivalent to *WRA*.

Filtering and Search Heuristics

When one or more rule evaluation measures are used to find or filter rules, they are called heuristics. As mentioned in Section 0, there are three approaches to rule induction: separate-and-conquer, exhaustive search, and divide-and-conquer. Among them, the first two approaches used some of the measures mentioned above as their search and filtering heuristics.

The separate-and-conquer approach learns a rule set by iteratively adding one rule at a time. The algorithm starts by performing a general-to-specific search to learn the best rule according to some measures. Examples covered by this rule are separated (or their weight is lowered) before learning the next rule. This is repeated until each example is covered by at least one rule in the rule set or some stopping criteria are satisfactory. In each iteration, the general-to-specific algorithm starts with a default rule (the rule that classifies all examples to be positive), and then searches the space of possible rules by successively specializing the current best rule. Rules are specialized by greedily adding the condition which promises the highest gain according to some evaluation measures, i.e. *search heuristics*. Some measures

(search heuristics) have been introduced to use in this context: *entropy* (Eq.3) (Furnkranz and Flach, 2003), *Laplacian* (Eq. 4) (Clark and Boswell, 1991), *weighted relative accuracy* (Eq.2) (Lavrac *et. al*, 2004), etc.

The exhaustive search approach explores almost the whole search space (Liu *et. al*, 1998; Kavsek *et. al*, 2003). The basic idea is to use an association rule algorithm to gather all rules that predict the class attribute and also pass some minimum quality criteria according to some measures, i.e. *filtering heuristics*. The most commonly used filtering heuristic is to use a pair of measures: *support* (Eq.7) and *confidence* (Eq.6).

A HYPER-HEURISTIC TO FORM SEARCH HEURISTICS

A hyper-heuristic

In subsection Filtering and Search Heuristics, we have distinguished two kinds of heuristics: search and filtering heuristics. While the former often uses only one measure to evaluate rules in learning systems, the latter can use more than one measures to filter redundancies out. In this section, we focus on how to construct measures to use in search heuristics, and we will refer these measures to heuristics.

Definition 4: Coverage space of a rule *r*(*n, p*), (or *pn*-space, note: small *p* and *n***)** is a subspace of *PN*-space that includes all points (n_1, p_1), where $0 \le n_1 \le n$ and $0 \le p_1 \le p$.

Definition 5: A rule $r_1(n_1, p_1)$ is a **descendant** of a rule $r(n, p)$, if r_1 is the result of adding one or more conditions to the rule *r*. In this case, rule *r* is called **an ancestor** of r_1

We can easily see that if rule $r_1(n_1, p_1)$ is a descendant of a rule $r(n, p)$, the point (n_1, p_1) corresponding to r_1 in *PN*-space must be in the coverage space of *r* (or *pn*-space, see Figure 2), i.e., $0 \le n_1 \le n$ and $0 \le p_1 \le p$. The point $(0, p)$ in the *pn*-space corresponds to the best specific rule of *r*, where the general-to-specific procedure tries to reach. Points in the main diagonal of the *pn*-space correspond to rules r_1 with class distribution similar to the class distribution of *r*.

Definition 6: A *h*-beam search space $bss_h(r)$ of a rule $r(n, p)$ with respect to an heuristic *h* is the part of the coverage space of *r* that includes all points (n_1, p_1), where $h(n_1, p_1) > h(n, p)$.

In other words, $bss_h(r)$ is the part of the coverage space of *r* that includes points upper the *h*-isometric going through the point (n, p). Figure 2 presents the beam search space of a rule respective to some heuristics: entropy h_{en}, *m*-estimate h_m, and weighted relative accuracy h_{wra}. We can easily see that: for every rule $r(n, p)$: $bss_{hwra}(r) \subseteq bss_{hm}(r) \subseteq bss_{\text{hen}}(r) = \text{coverage\_space}(r)/2$ with $m > 0$ (Figure 2). This is the reason why h_{wra} often produces too general rules (Lavrac *et. al*, 2004), while h_{en} produces too specific rules (Furnkranz and Flach, 2003).

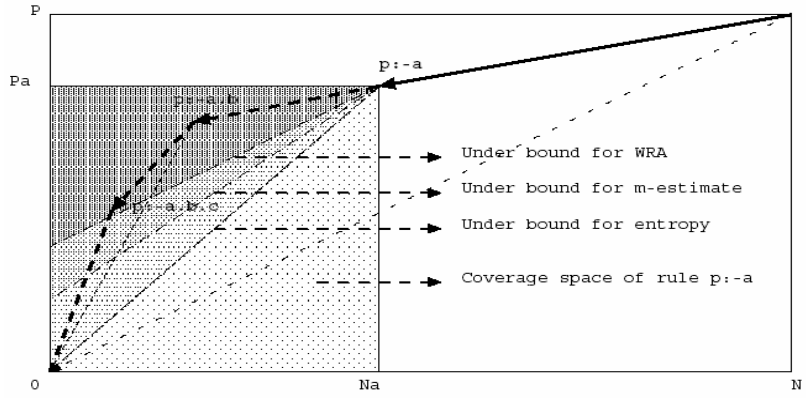

Figure 2 Beam search space of rule $p \leftarrow a$ with different heuristics: entropy (*en*), *m*-estimate and weighted relative accuracy (*WRA*)

Hyper-heuristic:

Returning to the main task in this section: how to form a search heuristic? As described in subsection Filtering and Search Heuristics, search heuristics are often used in general-to-specific algorithms. From the default rule (rule with body that is always true), a general-to-specific algorithm iteratively specializes the current best rule $r(n, p)$ by adding the condition that would produce the best rule according to a specific heuristic *h*, i.e. it searches if there is a descendant of the best current rule $r(n, p)$ in the *h*-beam search space of *r*. If there is, this descendant will replace the current best rule $r(n, p)$. Therefore, the smaller the rate between the area of the *h*-beam search space and that of the coverage space of *r*, the sooner the specialization process is stooped; and hence the more general the learned rule is. In other words, the purpose of the heuristic *h* is to restrict the search space in a specialization process to find rules better than the current one.

To form a search heuristic, we first state requirements that it needs to meet. From the above analysis, we can draw the following common properties of search heuristics:

(1) The rule covering *P* positive examples and *0* negative example (corresponding to the point $(0, P)$), if existed, is always the best rule; and the rule covering *N* negative and *0* positive examples (corresponding to the point $(N, 0)$) is the worst one respect to every search heuristic.
(2) If a point (n, p) of rule $r(n, p)$ is "nearer" to the point $(0, P)$ (the ideal point), it is better with respect to every heuristic.
(3) The purpose of a measure (heuristic) is to order (or rank) rules. Two measures h_1 and h_2 are equivalent if they order all the rules in the same way (for example, measures *entropy* and *confidence*). If the set of all isometrics of h_1 is equal to that of h_2, two measures h_1 and h_2 would be equivalent.
(4) The rate between the area of a *h*-beam search space and that of the coverage space of *r* is an important factor to control the generality of learned rules.

From four above properties, we can image a visualization way to construct a search heuristic based on the *PN*-space. The visualization tool *PN*-space allows us to

form a new search heuristic with desired properties and that would not be equivalent to the previously known ones. We can also easily check that the new heuristic would have a new "ordering" ability or the same as the existing ones.

In the next two subsections, we will introduce two kinds of search heuristics: linear and quadratic measures. With the linear heuristics, their mathematical formula is simple, but we can show that they are equivalent to many previously known ones according to an appropriate value of the parameter. The quadratic heuristics are non-linear and have new strong "ordering" abilities that linear measures do not have.

Linear Heuristics

All existing search heuristics are linear. Although they have been proposed by different authors at different time, Furnkranz and Flach (2003) proved that some of them have the same ordering/ranking ability since they have the same set of isometrics in the *PN*-space. In this subsection, we will form a general linear heuristic based on the desired characteristics of its isometrics in the visualization tool *PN*-space. With this approach, we can list all possible linear heuristics without the identical ordering ability.

The characteristics of a linear heuristic must include: (1) its isometrics should be linear (of course), (2) to meet requirements 1 and 2 mentioned above. Therefore, the linear measure at the rule $r(n, p)$ has the following formula:

$$h_{k\_linear}(r) = p - kn = n(\frac{p}{n} - k) \tag{9}$$

where k is the parameter, which defines the angle between the isometric and the horizontal axis.

The general formula is very simple, but many existing linear measures are equivalent (the same ordering ability) to some specials of this general linear measure. For example, the measure *accuracy* (Eq.1) and *confidence* (Eq.6) are the special or equivalence of h_{k_linear} when $k=1$; the *weighted relative accuracy* (*WRA*) (Eq.3) is equivalent to the special of h_{k_linear} with $k=P/N$; and the measure *positive_support* (Eq.7) is the special of h_{k_linear} with $k=0$.

Non-linear Heuristics

Now we propose completely new search heuristics that have non-linear isometrics. Similar to the formation of linear heuristics mentioned above, we first state some characteristics of these new search heuristics as follows: (1) their isometrics are quadratic curves; (2) they meet requirements 1 and 2 as described above. We can easily form the two following kinds of heuristics *quad_1* (Eq. 10) and *quad_2* (Eq. 11) with quadratic isometrics, quarters of nested ellipses centered at (0, P) and (N, 0) respectively.

$$h_{quad\_1}(r) = \frac{1}{\frac{(p-P)^2}{P_1^2} + \frac{n^2}{N_1^2}} \tag{10}$$

$$h_{quad\_2}(r) = \frac{p^2}{P_1^2} + \frac{(n-N)^2}{N_1^2} \tag{11}$$

where P_1 and N_1 are parameters.

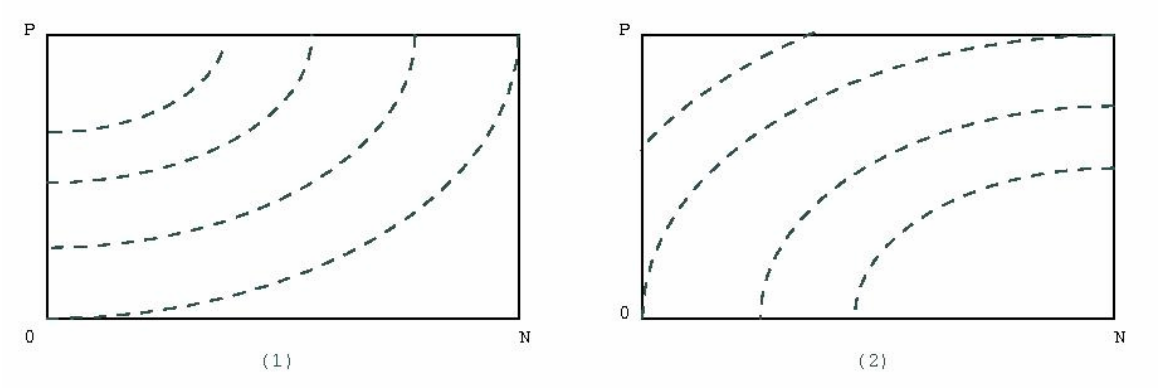

Figure 3 Isometrics of (1) h_{quad-1}, and (2) h_{quad-2} with P_1=P and N_1=N

When P_1 and N_1 are equal to P and N respectively, isometrics of these measures are shown in Figure 3; and the beam search space of these two heuristics at a rule $p \leftarrow a$ (Figure 4) are more flexible than that of linear ones.

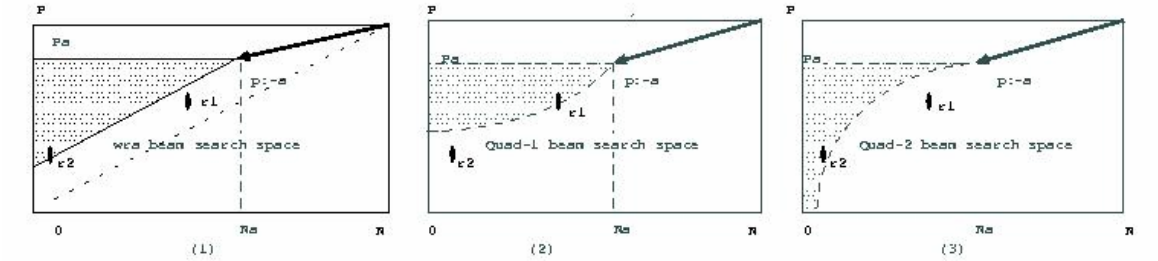

Figure 4 Beam search space of rule $p \leftarrow a$ with heuristics:
(1) *wra*, (2) *quad_1* and (3) *quad_2*

As can be seen in Figure 4, one of non-linear heuristics, *quad_1*, should be better than weighted relative accuracy (*wra*), which was previously known the best heuristic for descriptive rule induction (Kavsek *et. al*, 2003; Lavrac *et. al*, 2004). Similar to *wra*, the *quad_1* also provides a tradeoff between the rule covering and confidence. Moreover, *quad_1* is more flexible to prefer rules nearer the "ideal" point (*0, P*), which every search heuristic should try to reach. For example, rule r_1 is out and r_2 (Figure 4) is in the *wra*-beam search space of rule $p \leftarrow a$, this may not be what we want to. But these events are inversively happen when one uses *quad_1* instead of *wra* heuristic.

EXPERIMENTS

In this section, we will show some characteristics of rules that are learned from using some typical linear and quadratic measures described above as rule search heuristics: (1) *laplacian* (Eq. 4), which is a special case of m-estimate (Eq.5) and

was previously the best performance heuristic for predictive rule induction (Clark and Boswell, 1991); (2) some specials of our linear heuristic *k-linear*; (3) *weighted relative accuracy (wra)* (Eq. 2), which was previously the best heuristic for descriptive rule induction; (4) *quad_1* (Eq. 10), which is expected to be better than *wra* heuristic; and (5) *quad_2* (Eq. 11).

We first improve a rule induction system CN2-SD (Lavrac *et. al*, 2004) in two aspects: (1) implementing the new search heuristics: k-linear and quadratic heuristics; and (2) implementing different methods for applying rules to predict new examples.

Rule Induction CN2-SD

CN2 (Clark and Boswell, 1991; Clark and Nibblet, 1989) is a rule learning system originally designed for classification and prediction tasks. Recently, it has also been improved for description tasks by using a weighted covering strategy (an extension to the separate-and-conquer strategy in CN2, see Section 3.2), combined with a rule search heuristic (weighted relative accuracy) that favors rules with higher generality (Lavrac *et. al*, 2004). We further improve this version (CN2-SD) that implements our new search heuristics: *k-linear*, *quad_1* and *quad_2*. We also add two methods for applying rules to predict new examples in the case of induced rules are un-ordered: the probability as in (Lavrac *et. al*, 2004), and relative probability (related to the prior probability) that can overcome the biased classes problem. The improvement of CN2-SD can be obtained in the web site *http://user.dhsphn.edu.vn/ThoHoan/cn2-sd2*.

UCI datasets

We conducted some experiments to show the properties of rules produced by some search heuristics including the new ones: *quad_1* and *quad_2*. We use the rule learning systems CN2-SD (Lavrac *et. al*, 2004) with some improvements described in Section 4.1. Ten UCI data sets were selected for the evaluation: Audiology, Balance, Wine, Heart-Cleveland diseases, Heart-Hungarian diseases, Echocardiogram, Hepatitis, Iris, Wine, Heart-Cleveland, Heart-Hungarian, Hepatitis, Ionosphere, Iris, and Breast. We report the average number of induced rules, the average number of conditions (*#cond*) of rules, the average number of covered examples (*#cover*) of rules, and the predictive accuracy with two rule applying methods: major probability (*acc-1*) and relative probability (*acc-2*) by 10×10-fold cross-validations (Table 1).

We confirmed that each rule search heuristic is suitable for some datasets according to the criterion accuracy (Lavrac *et. al*, 2004) . We also confirmed the expectation as analysis in Section 0 that heuristic *quad_1* would be better than *wra* in descriptive tasks: the *quad_1* often produces “better” rule set (although the average number of learned rules from *quad_1* is often smaller, the average coverage of rules are greater and the accuracy performance is competitive) (see Table 1).

Dataset		*Lap.*	*0.2_Lin*	*0.5_Lin*	*1.0_Lin*	*WRA*	*Quad_1*	*Quad_2*
	#rule	78.9	53.4	42.6	44.8	79.2	74.3	77.8
	#cond.	3.3	2.7	2.3	1.6	3.3	4.1	2.2
Audiolog	#cover	10.0	9.2	7.6	3.7	6.8	9.1	8.9
	Acc-1	66.5	65.2	67.3	56.5	71.6	68.8	60.5
	Acc-2	75.1	76.4	69.3	56.7	**77.2**	74.3	72.9
	#rule	165.7	45.6	38.6	41.1	62.9	29.9	59.3
	#cond.	3.5	2.9	2.7	2.7	3.0	2.9	2.7
Balance	#cover	20.9	84.4	70.2	48.7	45.9	121.3	29.1
	Acc-1	80.9	82.2	79.7	79.7	79.8	**82.5**	74.6
	Acc-2	79.9	82.2	79.7	79.7	79.4	**82.4**	74.3
	#rule	22.5	19.4	20.2	18.7	20.5	16.5	19.9
	#cond.	2.2	3.0	2.6	2.3	2.5	3.5	2.0
Wine	#cover	32.2	46.2	41.1	37.1	40.4	49.7	32.1
	Acc-1	95.3	95.6	95.5	95.1	95.5	95.2	94.6
	Acc-2	95.2	95.9	95.9	95.2	95.9	95.8	94.5
	#rule	39.0	18.5	18.6	20.5	20.9	26.3	24.2
Heart-	#cond.	3.2	4.0	3.4	3.0	3.1	4.9	1.9
Clevel-	#cover	24.8	91.0	62.0	40.1	39.0	63.8	28.2
And	Acc-1	79.9	78.6	78.7	77.8	78.4	**81.1**	78.4
	Acc-2	79.9	78.6	78.7	77.8	78.4	**81.1**	78.4
	#rule	49.6	20.4	18.1	18.9	19.3	21.7	22.5
Heart-	#cond.	3.5	3.7	3.2	3.1	3.1	4.1	2.1
Hunga-	#cover	20.7	75.2	66.5	48.0	43.4	63.2	29.7
Rian	Acc-1	79.9	77.3	80.1	79.2	80.3	81.3	79.4
	Acc-2	79.9	80.1	80.8	79.5	80.5	**82.4**	80.2
	#rule	34.4	10.1	17.9	19.7	20.4	20.4	23.8
Echocar-	#cond.	3.3	2.7	2.9	2.9	3.0	3.4	2.3
Diagram	#cover	11.1	35.9	27.3	18.8	16.6	29.4	10.9
	Acc-1	70.7	70.6	69.3	66.8	68.0	68.5	68.7
	Acc-2	**71.8**	69.4	68.5	67.7	68.5	**71.3**	69.8
	#rule	29.8	18.3	17.3	18.7	21.9	21.4	20.8
	#cond.	3.5	3.7	3.3	3.2	3.2	3.6	2.4
Hepatitis	#cover	21.7	44.9	41.7	34.4	25.7	40.0	21.8
	Acc-1	81.1	79.3	80.5	82.6	80.6	81.9	82.0
	Acc-2	81.8	82.9	**83.5**	82.8	80.7	82.7	79.7
	#rule	40.9	20.1	24.4	26.9	24.4	22.2	25.6
Iono-	#cond.	2.5	3.3	2.9	2.5	2.4	3.8	2.0
Sphere	#cover	43.7	92.6	62.9	51.5	47.9	77.7	40.9
	Acc-1	92.1	93.5	93.5	92.6	89.9	91.9	91.5
	Acc-2	92.6	**93.6**	**93.6**	92.6	90.0	92.0	91.0
	#rule	12.4	9.1	10.5	8.6	10.3	10.2	8.4
	#cond.	2.1	1.6	1.7	1.8	1.7	1.9	1.5
Iris	#cover	27.8	38.7	35.5	33.1	35.7	43.4	31.8
	Acc-1	95.4	**95.7**	95.4	91.4	95.5	94.5	94.2
	Acc-2	95.4	**96.0**	95.8	91.5	95.8	94.7	95.1
	#rule	62.6	35.5	36.5	37.8	36.6	42.7	53.9
	#cond.	1.8	1.2	1.4	1.5	1.4	1.4	1.5
Breast	#cover	51.8	102.1	91.6	83.6	97.8	84.3	63.4
	Acc-1	**95.8**	79.2	88.7	94.0	95.3	88.1	**95.5**
	Acc-2	**95.8**	84.2	89.8	94.2	95.4	89.2	**95.5**

Table 1 The average number of induced rules, number of conditions, coverage of rules and accuracy in the rule set on 10 datasets by different heuristics.

		Gasch00's	Gasch01's	Spellman98's
WRA	#rules	25.154	27.251	12.299
	#redandant rules	19.534	22.710	10.566
QUAD_1	#rules	19.196	23.825	11.285
	#redandant rules	13.909	18.822	9.386

Table 2. The number of transcriptional regulatory rules learned by heuristics *wra* and *quad_1*; and that of redundant rules on three microarray datasets

Microarray dataset

We used rule CN2-SD (see subsection Rule Induction CN2-SD) with two search heuristics *wra* and *quad_1* to find transcriptional regulatory rules, a problem in bioinformatics (Pham *et. al*, 2005). Transcriptional regulatory rules are induced from three microarray datasets: Gasch00's (Gasch *et. al*, 2000), Gasch01's (Gasch *et. al*, 2001), and Spellman1998's (Spellman *et. al*, 1998). In this work, we used a filtering measure consistency (Pham *et. al*, 2005) to remove redundant regulatory rules that the system found. Table 2 shows the number of transcriptional regulatory rules induced by two heuristics weighted relative accuracy *wra* and *quad_1* together the number of removed rules (for more details, see Pham *et. al*, 2005). We found that *quad_1* produces fewer redundancies and many interesting rules in the final set of transcriptional regulatory rules (after filtered) from heuristic *quad_1* are not found in the set of rules from heuristic *wra*. This once gain suggested that *quad_1* is better than *wra* for descriptive tasks.

CONCLUSIONS ANS FUTURE WORK

In this work, we have proposed a general approach (a hyper-heuristic) to the formation of rule search heuristics in existing rule induction systems. We have also constructed two classes of rule search heuristics: linear and quadratic by the proposed hyper-heuristic. Many existing heuristics are special cases of the new linear heuristic class. One of our quadratic heuristics (*quad_1*) has exhibited to be more suitable than the previously best known heuristic *weighted relative accuracy* for descriptive rule induction. However, more experiments on real problems are needed to confirm this conclusion.

ACKNOWLEDGEMENTS

This work was partly supported by a Grant-in-Aid for Scientific Research on Priority Areas (C) "Genome Information Science" from the Ministry of Education, Culture, Sports, Science, and Technology of Japan; the COE project JCP KS1 from Japan Advanced Institute of Science and Technology (JAIST). The first author has been supported by a Vietnamese government scholarship from the Ministry of Education and Training of Vietnam. We would like to express our sincere thanks to Professor Kenji Satou, Head of Genetic Knowledge System Laboratory for his support and cooperation.

REFERENCES

Cestnik, B. (1990). Estimating probabilities: a crucial task in machine learning. *In Proceedings of the Ninth European Conference on Artificial Intelligence*, pages 147-149.

Clark, P. and Boswell, R. (1991). Rule induction with CN2: Some recent improvements. *In Proceedings of the 5th European Working Sessions on Learning*, pages 151-163, Porto, Portugal.

Clark, P. and Nibblet, T. (1989). The CN2 induction algorithm. *Machine Learning*, 3(4):261-283.

Furnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):03-54.

Furnkranz, J. and Flach, P. (2003). An analysis of rule evaluation metrics. *In 20th International Conference on Machine Learning (ICML-03)*, Washington.

Furnkranz, J. and Flach, P. (2004). An analysis of stopping and filtering criteria for rule learning. *In 15th European Conference on Machine Learning (ECML-04)*, Italy.

Furnkranz, J. and Flach, P. (2005). Roc 'n' rule learning - towards a better understanding of covering algorithms. *Machine Learning*, 58(1):39-77.

Gasch, A., Huang, M., Metzner, S., Botstein, D., Elledge, S., and Brown, P. O. (2001). Genomic expression responses to DNA-damaging agents and the regulatory role of the yeast atr homolog mec1p. *Mol Biol Cell*, 12(10):2987-3003.

Gasch, A., Spellman, P., Kao, C., Carmel-Harel, O., Eisen M.B., Storz, G., Botstein, D., and Brown, P. (2000). Genomic expression programs in the response of yeast cells to environmental changes. *Mol Biol Cell*, 11(12):4241-4257.

Kavsek, B., Lavrac, N., and Javanoski, V. (2003). Apriori-sd: Adapting association rule learning to subgroup discovery. *In Intelligent Data Analysis IDA 2003*.

Lavrac, N., Flach, P., and Zupan, B. (1999). Rule evaluation measures: A unifying view. *In Ninth International Workshop on Inductive Logic Programming (ILP'99)*.

Lavrac, N., Kavsek, B., Flach, P., and Todorovski, L. (2004). Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5:153-188.

Liu, B., Hsu, W., and Ma, Y. (1998). Integrating classification and association rule mining. *In 4th International Conference on Knowledge Discovery and Data Mining KDD 98*.

Pham, T.H., Clemente, J.C, Satou, K. and Ho, T.B. (2005). Computational discovery of transcriptional regulatory rules. *Bioinformatics*, 21 (suppl.2):ii101-107.

Quinland, J. (1986). Induction of decision trees. *Machine Learning*, 1:81-106.

Quinland, J. (1987). Generating production rules from decision trees. *In 10th International Joint Conference on Artificial Intelligence (IJCAI-87)*.

Spellman, P.T, Sherlock, G., Zhang, M. Q., Iyer, V. R., Anders, K., Eisen, M. B., Brown, P. O., Botstein, D., and Futcher, B. (1998). Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mol Biol Cell*, 9(12):3273-3297.