Wrapper for Ranking Feature Selection*

Roberto Ruiz, Jesús S. Aguilar-Ruiz, and José C. Riquelme

Departamento de Lenguajes y Sistemas, Universidad de Sevilla
Avda. Reina Mercedes /N. 41012 Sevilla, España
{rruiz,aguilar,riquelme}@lsi.us.es

Abstract. We propose a new feature selection criterion not based on calculated measures between attributes, or complex and costly distance calculations. Applying a wrapper to the output of a new attribute ranking method, we obtain a minimum subset with the same error rate as the original data. The experiments were compared to two other algorithms with the same results, but with a very short computation time.

1 Introduction

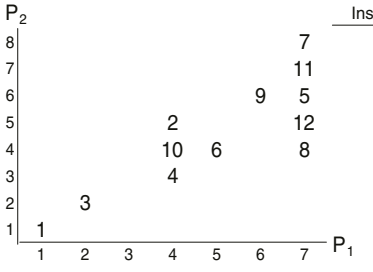
Feature selection methods can be grouped into two categories from the point of view of a method's output. One category is about ranking feature according to same evaluation criterion; the other is about choosing a minimum set of features that satisfies an evaluation criterion. There are several taxonomies of these evaluation measures in previous work, depending on different criterions: Langley [1] group evaluation functions into two categories: filter and wrapper. Blum y Langley [2] provide a classification of evaluation functions into four groups, depending on the relation between the selection and the induction process: embedded, filter, wrapper, weight. Another different classification, Doak [3] and Dash [4] provide a classification of evaluation measure based on their general characteristics more than in the relation with the induction process. The classification realized by Dash, shows five different types of measures: distance, information, dependence, consistency y accuracy.

In this paper, we propose a new feature selection criterion not based on calculated measures between attributes, or complex and costly distance calculations. This criterion is based on a unique value called NLC. It relates each attribute with the label used for classification. This value is calculated by projecting data set elements onto the respective axis of the attribute (ordering the examples by this attribute), then crossing the axis from the beginning to the greatest attribute value, and counting the Number of Label Changes (NLC) produced.

All filter methods use heuristics based on general characteristics to evaluate the merit of a feature or a features' subsets. However, the wrappers include the learning algorithm as a part of their evaluation function. Wrappers usually provide better accuracy but are computationally more expensive than the Filter

* This work has been supported by the Spanish Research Agency CICYT under grant TIC2001-1143-C03-02.

Z.R. Yang et al. (Eds.): IDEAL 2004, LNCS 3177, pp. 384–389, 2004.



Instances	P_1	P_2	Class
1	1	1	O
2	4	5	E
3	2	2	O
4	4	3	E
5	7	6	O
6	5	4	E
7	7	8	O
8	7	4	E
9	6	6	O
10	4	4	E
11	7	7	O
12	7	5	E

Fig. 1. Data set with twelve elements and two classes (Even,Odd)

schemes. In our algorithm, we want to take advantage of the two methods. The ranked list of features sorted according to each NLC is produced. The Starting point is the empty set, and we add features from the list sequentially in the same order. The features are added depending on whether or not they increase the performance of the learner.

2 Number of Label Changes (NLC)

2.1 Definitions

To describe the definitions below, let us consider the situation depicted in Figure 1, with twelve elements numbered and two labels (O-odd numbers and E-even numbers).

Definition 1. *Let the attribute P_j be a continuous or discrete variable that takes values in $A_j = [a_j, b_j] \subseteq \mathbb{R}$ if it is a continuous attribute, and if it is a discrete attribute, let A_j be a set of possible values. Let Ω be a set of m attributes.*

In Figure 1, P_1 and P_2 are continuous attributes, $A_1 = [1, 7]$, $A_2 = [1, 8]$ and $\Omega = \{P_1, P_2\}$.

Definition 2. *Let Λ be a set of discrete labels.*

In Figure 1, $\Lambda = \{E, O\}$ depending on the instance number.

Definition 3. *An example e is a tuple of values $(p_1, p_2, \ldots, p_m, l) \in A_1 \times \cdots \times A_m \times \Lambda$ where p_j is the value of the attribute P_j. If P_j is continuous $p_j \in [a_j, b_j]$. If P_j is discrete $p_j \in A_j$. $l \in \Lambda$ is the label of e. Let E be the set of all examples in the training set. Therefore, $E \subseteq A_1 \times \cdots \times A_m \times \Lambda$. Let n be the cardinal of E.*

In Figure 1, E is the table on the right, and $e_5 = (7, 6, O)$.

Definition 4. *We define the functions π_j, $\forall j : 1..m$ and lab.*

$$\pi_j : E \to A_j \; \pi_j(e) = p_j \quad \wedge \tag{1}$$

$$lab : E \to \Lambda \quad lab(e) = l \tag{2}$$

In Figure 1,

$$\pi_1(e_1) = 1 \quad \forall i : 1..12\ \pi_1(e_i) = \{1, 4, 2, 4, 7, 5, 7, 7, 6, 4, 7, 7\}$$
$$lab(e_1) = O\ \forall i : 1..12\ lab(e_i) = \{O, E, O, E, O, E, O, E, O, E, O, E\}$$

Definition 5. *Let $\sigma_j(E) =< e_1, \ldots, e_n >$ be an ordered sequence of E by the attribute j, then we say that there is a multiple ordered subsequence (m.o.s.) and we denote S if*

$$\exists\ i, k > 0\ /\ e_{i-1} <_j e_i =_j \cdots =_j e_{i+k} <_j e_{i+k+1} \tag{3}$$

being the m.o.s. $S =< e_i, \ldots, e_{i+k} >$ (if $i = 1$ the first condition is removed) A majority label of an m.o.s., $ml(S)$, is the mode of de set $\{lab(e_j)\}_{j=i+1}^{k}$.

m.o.s. in Figure 1 are:
$S =< e_4, e_{10}, e_2 >$ is an m.o.s. by the attribute P_1, because the examples e_4, e_{10} and e_2 have the same value (4) for P_1. $ml(S) = E$ because all the examples are even numbers.

Definition 6. *Let $\sigma_j(E) =< e_1, \ldots, e_n >$ be an ordered sequence of E by the attribute j, then we say that there is a simple ordered subsequence (s.o.s.) and we denote S if*

$$\exists\ i, k > 0\ /\ e_{i-1} =_j e_i <_j e_{i+1} <_j \cdots <_j e_{i+k-1} <_j e_{i+k} =_j e_{i+k+1} \tag{4}$$

being the s.o.s. $S =< e_{i+1}, e_{i+2}, \ldots, e_{i+k-1} >$ (if $i = 1$ the first condition is removed) The instance e_{i+1} is called first example, $fe(S)$, and e_{i+k-1} last example $le(S)$.

s.o.s. in Figure 1 are:
$S =< e_1, e_3 >$ is an s.o.s. by the attribute P_1, because the examples e_1 and e_3 have different values (1 and 2) for P_1. $fe(S) = e_1$ and $le(S) = e_3$.

Theorem 1. *Given an ordered sequence of examples $\sigma_j(E)$ by an attribute j, it can be divided into subsequences where there is an s.o.s., or a m.o.s., or a sequence of the subsequences of one after the other. We take into account that one m.o.s. always comes after an s.o.s., but after an m.o.s. an s.o.s. or an m.o.s. can come In general, $\sigma_j(E) = S_1, S_2, \ldots, S_r$ where if S_i is an s.o.s. then S_{i+1} is m.o.s., and if S_i is an m.o.s., then S_{i+1} can be an s.o.s. or m.o.s.*

Definition 7. *Given an ordered sequence S of k examples $S =< e_1, e_2, \ldots, e_k >$ we define the function $ch : S - \{e_k\} \rightarrow \{0, 1\}$*

$$ch(e_i) = \begin{cases} 1\ si & lab(e_i) \neq lab(e_{i+1}) \\ 0\ si & lab(e_i) = lab(e_{i+1}) \end{cases} \quad \forall i : 1 \ldots (k–1) \tag{5}$$

In Figure 1, for P_2, in the s.o.s. $S =< e_1, e_3, e_4 >$:

$$ch(e_1) = 0 \quad because\ lab(e_1) = lab(e_3)$$
$$ch(e_3) = 1 \quad because\ lab(e_3) \neq lab(e_4)$$

Table 1. Main Algorithm

```
Input: E training (n examples, m attributes)
Output: E reduced (n examples, k attributes)
  for each attribute P_j ∈ Ω
     QuickSort(E,j)
     NLC(j)
  Attribute Ranking
  S ← "first-f (first ranked att.)"
  F ← "initial set of n features - first-f"
  Calculate ER (Error Rate) with S
  repeat until F = "empty set"
     if ER with S ⋃ first-f > √ ER with S
        S ← S ⋃ first-f
     F ← F - first-f
```

Definition 8. *Given a set E of examples, $\sigma_j(E)$ the ordered sequence by the attribute j and $S_1 \dots S_r$, we define the function $NLC : \Omega \to \mathbb{N}$*

$$NLC(j) = \sum_{i=1}^{r} nch(S_i) \tag{6}$$

In Figure 1, $\sigma_1(E) = S_1, S_2, S_3, S_4$, where $S_1 =< e_1, e_3 >$, $S_2 =< e_4, e_{10}, e_2 >$, $S_3 =< e_6, e_9 >$, $S_4 =< e_8, e_{12}, e_5, e_{11}, e_7 >$ with $nch(S_1) = 1$, $nch(S_2) = 0$, $nch(S_3) = 1$, $nch(S_4) = 4 \Rightarrow NLC(P_1) = 6$. We also obtain $NLC(P_2) = 2$.

3 Algorithm

The algorithm is very simple and fast (Table 1). It has the capacity to operate with continuous and discrete variables as well as with databases which have two classes or multiple classes.

There are two phases in the algorithm: Firstly, the attributes are ranked. For each attribute, the training-set is ordered and we count the NLC throughout the ordered projected sequence. In the second place, we deal with the list of attributes one time. We obtain the Naive Bayes [5] error rate with the first attribute in the list and it is marked as selected. We obtain the Naive Bayes [5] error rate again with the first and the second attributes. The second will be marked as selected depending on the accuracy obtained is significatively better ($> \surd$). Next step is classify again with the marked attributes and the next attribute on the list, and it will be marked depending on the accuracy obtained. Repeat the process until the last attribute on the ranked list is reached.

4 Experiments

In this section we compare the quality of selected attributes by the NCL measure with the selected attributes by the other two methods: Information Gain (IG)

Table 2. Accuracy of attribute selection with C4.5, 1NN and naive Bayes

Data set	C4.5			1NN			Naive		
	NLC	RLF	IG	NLC	RLF	IG	NLC	RLF	IG
anneal	87.99 × ×	93.65	94.31	88.09 × ×	94.54	94.88	86.86 × ×	91.64	92.87
balance	78.39	78.39	78.39	86.88	86.88	86.88	88.81	88.81	88.81
g_credit	70.30	71.20	71.70	63.60 ×	69.70	70.60	69.80 ×	71.20	72.40
diabet.	74.87	75.65	75.65	69.80	68.76	68.76	76.03	76.55	76.55
glass	58.07	63.51	61.75	55.63	60.80	57.08	51.06	49.16	47.25
glass2	78.49 √	64.85	77.83	76.54	67.32	74.89	73.60 √	62.50	73.64
heart-s	74.44	76.67	72.22	71.48	73.70	69.26	73.33	75.93	71.48
ionosph	87.16	90.02	89.73	87.45	88.02	87.44	87.73	89.15	88.88
iris	92.00	93.33	93.33	92.67	94.67	91.33	92.67	94.67	92.67
kr-vs	94.27	94.27	94.27	94.27	94.27	94.27	94.27	94.27	94.27
lymph	74.95	71.62	74.95	74.29	70.95	74.95	74.95	72.29	74.95
segment	96.75 √	95.89	96.54	96.71	97.01	96.84	88.61 √	85.71	89.48
sonar	70.67	68.83	72.10	62.45	65.83	65.31	72.05	72.17	72.52
splice-2	93.92	93.70	93.95	87.99	88.12	87.59	94.14	94.08	93.89
vehicle	64.87	59.57	63.23	60.27	54.84	56.62	47.53 √	50.83	45.52
vowel	77.37	74.85	78.38	94.04 √	87.58	94.75	63.23 √	60.30	63.64
wave	77.78 √	76.38	77.52	78.26 √	75.74	78.14	81.84 √	80.62	81.96
zoo	85.18	87.09	88.18	85.18	87.09	88.18	84.18	86.18	84.18

and the ReliefF method [6]. The quality of each selected attribute was tested by means of three classifiers: the Naive Bayes [5], C4.5 [7] and 1-NN [8].

The implementation of the induction algorithms and the others selectors was done using the Weka library [9] and the comparison was performed with eighteen databases of the University from California Irvine [10]. The measures were estimated taking the mean of a ten-fold cross validation, and the same folds were used for each algorithm training-sets. To asses the obtained results, two paired t statistical tests with a confidence level of 95% were realized.

In order to establish the number of attributes in each case, we deal with the list of attributes once. We obtain the Naive Bayes error rate with the first attribute in the list and it is marked as selected. We obtain the Naive Bayes error rate again with the first and the second attributes. The second will be marked as selected depending on whether the accuracy obtained is significantly better. The next step is to classify again, with the marked attributes, and the next attribute on the list. Then it will be marked depending on the accuracy obtained. Repeat the process until the last attribute on the ranked list is reached.

Table 2 shows a summary of the results of the classification using C4.5, 1NN and NB. The table shows how often each method performs significantly better (denoted by √) or worse (denoted by ×) than ReliefF (RLF) and Information Gain (IG). In ten of the one hundred and eight cases, the set of attributes selected by the NLC measure yields better accuracy than the two other methods. In ninety they are equal, and in eight they are worse than the other. Only in ANNEAL and GERMAN_CREDIT the accuracy is lower than the other methods, but the number of attributes selected is significantly better in both cases.

Algorithms reach a similar percentage of the original features retained (24% NLC, 22% RLF and 25%IG).

5 Conclusions

In this paper we present a deterministic attribute selection algorithm. It is a very efficient and simple method used in the preprocessing phase. A considerable reduction of the number of attributes is produced. It does not need distance nor statistical calculations, which could be very costly in time (correlation, gain of information, etc.). The computational cost to obtain the ranked list is lower than other methods O(m × n × log n). NLC takes 0.792 seconds in reducing 18 data sets whereas ReliefF takes 566 seconds and IG 2.19 seconds.

We conclude that by applying NLC, the knowledge attained in the original training file is conserved into the reduced training file, and the dimensionality of data is reduced significantly. We obtain similar results with the three methods, but needing much less time with NLC.

References

1. Langley, P.: Selection of relevant features in machine learning. In: Procs. Of the AAAI Fall Symposium on Relevance. (1994) 140–144
2. Blum, A., Langley, P.: Selection of relevant features and examples in machine learning. In: Artificial Intelligence. (1997) 245–271
3. Doak, J.: An evaluation of search algorithms for feature selection. Technical report, Los Alamos National Laboratory (1994)
4. Dash, M., Liu, H.: Feature selection for classification. Intelligent Data Analisys **1** (1997)
5. Duda, R., P.Hart: Pattern Classification and Scene Analysis. John Willey and Sons (1973)
6. Kononenko, I.: Estimating attributes: Analysis and estensions of relief. In: European Conference on Machine Learning. (1994) 171–182
7. Quinlan, J.: Induction of decision trees. Machine Learning **1** (1986) 81–106
8. Aha, D., Kibler, D., Albert, M.: Instance-based learning algorithms. Machine Learning **6** (1991) 37–66
9. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools with Java implementations. Morgan Kaufmann (1999)
10. Blake, C., Merz, E.K.: Uci repository of machine learning databases (1998)