Supervised Instance-based Learning Using Patterns in a Trie like Structure

2 authors:

Patricio Serendero
Universidade do Algarve
4 PUBLICATIONS **18** CITATIONS

SEE PROFILE

Miguel Toro
Universidad de Sevilla
210 PUBLICATIONS **1,642** CITATIONS

SEE PROFILE

Supervised Learning Using Instance-based Patterns

Patricio Serendero[1], Miguel Toro[2]
pserende@ualg.pt, mtoro@lsi.us.es
[1]Faculty of Sciences and Technology, U. of Algarve, Faro, Portugal
[2]Dept. of Languages and Information Systems, U. of Seville, Seville, Spain

Abstract: This paper introduces a new classification algorithm of the instance-based learning type. Training records are converted into patterns associated with a known class label, and stored permanently into a trie[1]-like tree structure along with other helpful information. Classifying new records is done selecting from the trie two best patterns as solutions hypotheses. Best pattern selection is done using standard distance metrics, a strength function and an exclusive values concept. Classification tests done on several data files have shown very accurate results.

Keywords: supervised learning, instance-based, pattern strength, exclusive values, trie.

1 Introduction

Supervised learning techniques, known also as classification, belong to the areas of Machine Learning and Knowledge Data Discovery. In general, their goal is to build up algorithms able to predict class labels in data files or whole databases. Training records of known class labels are used in the creation of the algorithm. Later, these are used to predict unknown classes previously unseen records. A plethora of methods and tools exists for this purpose. Among them: classification tree induction [23], [16], Bayesian classification methods [5], [9], neural networks [26], genetic algorithms [13], [17], and others[2]. Motivation for this work is around the following topics:

a) To treat data in such a way as to make them independent from attribute types.

[1] Pronounced "try", taken from the middle part of word "retrieve" [12]

[2] See also a comparison of Data Mining Tools in [10].

b) Have not one general hypothesis model but many hypotheses functions for records to be classified. c) To use data structures that soften somehow problems of attribute selection and threshold value typical of binary decision trees. [32].

2 Problem Definition

Let us consider the closed universe formed by a data file *R* composed of a finite set of *m* records *r*.

$$R = \{r_1, r_2,..,r_m\} \quad (1)$$

We consider set *R* as formed by a training subset R_T and a test subset R_X, such that $R = R_T \grave{E} R_X$. The first one will be used to *learn* data patterns and to help developing the classification algorithm. R_X is used for testing and algorithm verification. In a given record *r*, we find a finite sequential set of attributes *S*.

$$S = \{A_1, A_2, Ai_{...} A_n\}, \text{ a nonempty set.} \quad (2)$$

Every attribute $A_i \in S$ can take v_i values belonging to a set T_i, where T_i is the domain value. Every record can be associated with class labels l_1, l_2, etc. belonging to a set *L*.[3]

$$L = \{l_1, l_2, l_k\} \quad (3)$$

Each record *r* is formed by the Cartesian product of attribute values A|V and a label *l*, such that:

$$r = \{ v_1, v_2, ..., v_n, l \} \quad v_i \hat{I} T_i, l \hat{I} L. \quad (4)$$

From these definitions, we can define the following basic functions:

$$\text{val}_i(r) = v_i, \text{ the value associated to attribute } i \text{ from record } r$$
$$\text{label}(r) = l, \text{ the label associate with record } r. \quad (5)$$

3 Pattern Definition

Data attributes can have different data types: qualitative or non-numerical (symbolic, linguistic); quantitative or numerical (continuous, discrete). We would like to treat them equally, in order to create patterns where all attribute values are represented by integers. For non-numerical or categorical data, we do an arbitrary enumeration, if not already ordered by the domain expert. In addition, we apply *discretization* to all continuous-valued record's attributes. Discretization is the process of transforming the domain of a continuous attribute or feature into a finite number of intervals. For all A_i, user-defined intervals with lower and upper value-limits are established in T_i. Thus, for each attribute A_i corresponds a partition of the domain T_i. Intervals are represented by integers with

[3] In this article we use indistinctively the words class label, class or just label.

values from 1 to s_i. Thus, we convert every attribute A_i into a pattern value. Function ord() does this conversion:

$$p_i = \text{ord}_i(v_i),\ v_i \in T_i,\ p_i \in 1,..\ s_i\ . \tag{6}$$

This function returns p_i, an integer value representing one interval value for attribute A_i. We can define a pattern p as the sequence formed by n values.

$$p = <p_1, p_2,.., p_i,..,p_n>\ \ p_i = \{1..s_i\} \tag{7}$$

To each record $r = <v_1,v_2,..,v_n, l>$ corresponds a pattern obtained by applying function $\text{ord}_i()$. Hence, for every record r we obtain the corresponding pattern p associated with a given label as follows:

$$\text{pat}(r) = p = < p_1, p_2,..,p_n> = \acute{a}\text{ord}_1(v_1), \text{ord}_2(v_2),..,\text{ord}_n(v_n)\tilde{n} \tag{8}$$

For every pattern p we can define n sub-patterns q_i, which are the prefix portion of a pattern.

$$q_i = <p_1, p_2,.., p_i >,\quad q_i \text{ subsequence of } p,\ \ i = \{1..i\}. \tag{9}$$

Function freq(p) returns the number of records in R_T with pattern p.

$$\text{freq}(p) = \left|\{\ r\ \hat{I}\ R_T\ \left|\ \text{pat}(r) = p\ \}\right|\right. \tag{10}$$

In addition from above we can define

$$l_i = \text{freq}(q_i), \text{ the frequency of sub-patterns } q_i. \tag{11}$$

Labels are attached to patterns. Function labels(p) returns the set of labels associate to the subset of records with pattern p.

$$\text{labels}(p) = \{\ l \in L\ \left|\ \$\ r \in R_T \cdot \text{label}(r) = l\ \grave{U}\ \text{pat}(r) = p\ \}\right. \tag{12}$$

From this, we define function $\text{nlabels}(p) = \left|\ \text{labels}(p)\ \right|$, $\text{nlabels}(p) = \{1..L\ \}$

We can extend this concept to sub-patterns. Hence,

$$(\text{nlq})_i = \text{nlabels}(q_i) \tag{12a}$$

In any dataset R_T we find that generally function nlabels (p_n) = 1, meaning that a given full pattern $p \in R_T$ is associated with one label, making data consistent. As for sub-patterns, we often find that $\text{nlq}_i > 1$, meaning that more than one label shares the same q_i. This is the overlapped area of patterns. In general this is the case for initial values of i; As the value of i approaches n, factor nlq_i tend to be associated with one label and nlabels (p) = 1. From an intuitively viewpoint each pattern p represent a hypercube in a space of n dimensions. Function freq (p) measures the number of records from R_T in that hypercube. Function labels (p) represents the set of labels present in that hypercube.

4 Other Definitions

4.1 Distance Between Patterns

Several supervised learning methods use a metric distance to compare attribute values. [7]. We calculate the minimum distance between two patterns applying the minimum square method:

$$d(p, p') = \left(\sqrt{\sum_{i=1}^{n} | p_i - p_i' |^2}\right) / l_1 \quad (13)$$

The overall frequency of pattern p represented by $l_{1,}$ is used to add weight to the measure of distance.

4.2 The strength of a Pattern

We differentiate between strong and weak patterns according to the number of single class labels associated to each one of its n sub-patterns. If $(nlq)_i = 1$, sub-pattern q_i is strong. The contrary means a weak sub-pattern. We define function strongp(p) as a measure of pattern strength as follows:

$$\text{strongp}(p) = | \{ i : 1..n. \mid \text{nlabels}(q_i) = 1 \} | \quad (14)$$

Function strongp() varies from 0 to n and represents the total number of sub-patterns q_i of p associated with just one label. When strongp (p) = n, the pattern strength is maximum. If (nlabels (q_i) > 1 then $(nlq)_i = 0$, a weak sub-pattern. An important property of sub-patterns is that once the i^{th} sub-pattern becomes associated with one label, then all subsequent $p_{i+1,}$ $p_{i+2,}$ p_{i+n} sub-patterns also relate to the same label. When $i = n$, class overlapping should disappear; we expect that the label attached to the n^{th} element is the label attached to pattern p as a whole. This is our basic assumption: *no two identical full patterns are related to different classes.* If such case exists probably due to noise data, we consider the pattern as completely weak. We could not make any prediction about its label. This same criterion has been used before. See [11] and [31].

4.3 Exclusive Values

We define exclusive values as the unique maximum and minimum interval values shown by a given $p_i \in p$ always associated with the same label in R_T. From the point of view of a label, this represents its association with a given p_i and some value k for all patterns p in R_T. This can be represented with the following functions:

$\text{sr}(i, k) = \{ r \in R_T \mid \text{pat}(r) = <p1,..,pi,..,p_n> \text{ and } p_i = k \}$; $\text{sl}(i, k) = \text{label}((\text{sr}(i, k))$
and function

$$\text{nsl}(i,k) = \left|\text{sl}(i,k)\right| \quad (15)$$

Then, we define exclusive values as:

$$a = \text{ex}(p) = Ni : 1..n \mid (\text{nsl}(i, p_i) = 1)) \quad (16)$$

The number of p_i values in pattern p that appear always related to a given class.

4.4 The Majority Class

Training set R_T includes one or more subsets of records associated with the same class l; that is: $R_T = Rl_1 \; \dot{E} \; Rl_2 \; \dot{E} \; Rli \; \dot{E} \; Rl_j$ where Rl_j is the set of records with label l_j. We define *majority class* as the class with maximum number of records in R_T, calculated as follows:

$$\text{lmaj}(R_T) = \max(R1_1, Rl_2, .., Rl_j) \quad (17)$$

In populations with say, two classes l_1 and l_2, where class l_1 is strongly predominant, we will predict l_2, only when strong evidence is found that this is the case [14].

5 Algorithm

The algorithm is executed in two phases. The first corresponds to a pre-processing. It is the *learning* phase. It consists in reading sequentially all records in R_T and applying to each one of them the function in (7) converting them into p patterns and storing them into a trie structure. In this process the label indicator and a frequency l_i for each sub-pattern q_i, are stored as well. Hence, for every pattern in R_T, the structure holds:

$$p = <(q_1, nlq_1, l_1), (q_2, nlq_2, l_2), .., (q_n, nlq_n, l_n)> \quad (18)$$

We define P as the set of patterns in R_T with freq () > zero. This is to say, all existing patterns in R_T. Each one of them is associated with a known class l.

$$P = \{p \mid \text{freq}(p) > 0 \} \quad (19)$$

The second phase is the actual classification process. This is the *predictive* phase. A new test file R_X is read in sequentially; for each record r in R_X, the algorithm performs the conversion $p^x = \text{pat}(r)$ generating the target pattern p^x of unknown class label. For each $p^x{}_i$ value in p^x, the algorithm looks in the stored pattern structure for the closest p_i values using its distance metric obtaining pattern p^+. This is the closest existing pattern in R_T with respect to p^x. We repeat this operation in a slightly different way to obtain a "second" best pattern p^-. These two patterns represent our solution hypotheses. One of them will be chosen to predict the class for the unseen pattern p^x.

Patterns p^+ and p^- can be defined as follows: For any sub-pattern q_i, we define the set of k integers next(q_i). An integer k is in next(q_i) if $<q_i, k>$ if is a sub-pattern of P. Given p^x, then p^+ and p^- are defined recursively:

Let $q^+{}_i$, $q^-{}_i$, and $q^x{}_i$, be sub-patterns of p^+, p^- and p^x respectively. Further, let be

$q^{+}_{i+1} = \langle q^{+}_{i,} k^{+} \rangle$, $q^{-}_{i+1} = \langle q^{-}_{i,} k^{-} \rangle$ and $q^{x}_{i+1} = \langle q^{x}_{i,} k^{x} \rangle$. Thus, k^{+} is the closest integer in next (q^{+}_{i}) to k^{x} and k^{-} is the closest integer in next ($q^{-}_{i,}$) to k^{x}.

Pattern p^{+} is the closest to p^{x} in R_T and is calculated first. If two identical patterns exist in R_T and R_X, then $p^{x} = p^{+}$, and labels(p^{x}) = labels$(p^{+}.)$ Pattern p^{-} is searched next. The selected pattern must be different from p^{+}, but as close as possible to p^{x}. If a k^{-} value different from k^{+} is not available to add in next sub-pattern $\langle q^{-}_{i,} k^{-} \rangle$, existing value k^{+} is used instead. When this is achieved, q^{-}_{i} becomes distinct from $q^{+}_{i,}$ Subsequent p^{-}_{i+1} values are calculated as p^{+} was.

Unseen p^{x} instances are predicted based on the closest distance from p^{+} and p^{-} (13), plus three other parameters. The first is *the strength of a pattern*. From strength function in (14), we define the following two functions:

$$st^{+} = strongp(p^{+}) \text{ and } \quad st^{-} = strongp(p^{-}) \tag{20}$$

We consider a pattern strong, if the area were it belongs is clearly more disjunctive than others, with respect to a given class.

Secondly, the algorithm sees whether patterns p^{+} and p^{-} show the presence of *exclusive values*. Applying the exclusive function ex() from (16) we can obtain parameters a^{+} and a^{-} as follows:

$$a^{+} = ex(p^{+}) \text{ and } a^{-} = ex(p^{-}) \tag{21}$$

Factor a represents a degree of confidence in the disjunctive quality of a pattern's set of attribute values. Its value increases from zero by one as more p_i values present that feature. The algorithm will favor the pattern showing a larger ex (p) value.

If after applying these criteria the algorithm still cannot predict a label for p^{x}, then function lmaj() is used.

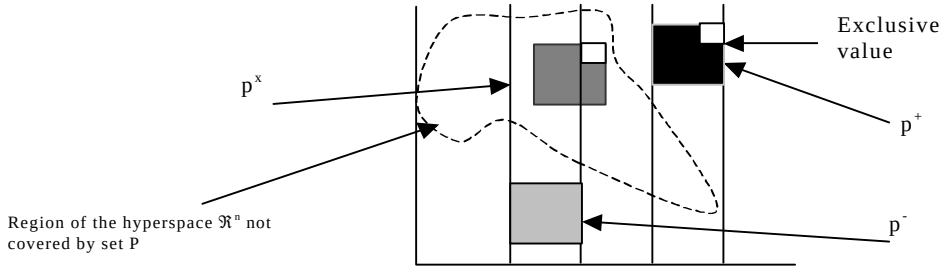

Figure 1. A new pattern p^{x} from R_X. Patterns p^{+} and p^{-} are obtained from R_T. Pattern p^{x} is closer to p^{+} in the hyperspace. It shows an exclusive value corresponding to the same class as p^{+}.

6 Results

We have performed experiments with our algorithm that we call Trie-Class on several datasets, and compared with published results in the literature. Datasets used for testing

comes from public domain in the web, mainly from the UCI[19]. Records with unknown attribute values were removed or converted to its mean value. Using a random function each time on the complete file, the standard ten-fold cross validation procedure has been done for all results. Accuracy values are calculated for the test file. We have used around 60% of records for the training file R_T; the remaining 40% for test file R_X. Results obtained with other tools were taken from several sources, namely [22], [20], [21], [24], [25] and [3].

Table I. Table I. Statlog Heart disease file.

Method	Accuracy %	Reference
Trie-CLASS	96.2	Ours
Naï ve Bayes	83.6	WEKA, RA
K*	76.7	WEKA, RA
IB1	74.0	WEKA, RA
1R	71.4	WEKA, RA
T2	68.1	WEKA, RA
MLP+BP	65.6	ToolDiag, RA
FOIL	64.0	WEKA, RA
RBF	60.0	ToolDiag, RA
InductH	58.5	WEKA, RA

Table II. Pima Indian Diabetes file.

Method	Accuracy %	Reference
Trie_CLASS	89.5	Ours
LogDisc	77.7	Statlog
Incnet	77.6	N.jankowski
DIPOL92	77.6	Statlog
Linear Discr. Analysis	77.2 -77.5	Statlog, Ster & Dobnikar
SMART	76.8	Statlog
GTO DT(5xCV)	76.8	Bennet and Blue
ASI	76.6	Ster & Dobnikar
kNN, k=22, Manhattan	75.5	Karol Grudzinski
OC1(10 5-fold CV)	73.4 -75.4	Murthy et al.
C4.5	73.0	Statlog .

Table III. Heart disease, Cleveland file.

Method	Accuracy %	Reference
Trie_CLASS	98.2	Ours

Incnet	90.0	N. Jankowski
28-NN,stand Euclidean, 7 features	85.1±	WD/KG
LDA	84.5	Ster & Dobnikar
Fisher discriminant analysis	84.2	Ster & Dobnikar
16-NN, stand, Euclidean	84.0±0.6	NCU
25-NN, stand, Euclidean	83.6 ±0.5	NCU
FSM, 82.4-84% on test only	84.0	R. Adamczak
Naïve Bayes	82.5-83.4	Rafal, Ster, Dobnikar
C4.5(5xCV)	77.8	Bennet and Blue

Table IV. Annealing file. (Sterling & Buntine).
Attribute intervals used: width= 20, strength = 700, len = 50 and thick = 50.

Method	Accuracy %	Reference
Trie-CLASS	98.6	Ours
LB	96.4	Bing Liu
CBA	96.4	Bing Liu
RIPPER	95.4	Bing Liu
C4.5 (AdaBoost Ensemble)	95.1	Quinlan [24]

Table V. Satellite image file (STATLOG version)
We used only four attributes as suggested: 17, 18,19 and 20.

Method	Accuracy %	Time
Trie_CLASS	91.1	8
k-NN	90.6	944
LVQ	89.5	44
Dipol92	88.9	111
Radial	87.9	74
Alloc80	86.8	28757
IndCart	86.2	9
CART	86.2	14
MLP+BP	86.1	53
Bayesian Tree	85.3	10
C4.5	85.0	1
New ID	85.0	53

Table VI. Iris file.

Method	Accuracy %	Reference
Trie-CLASS	97.9	Ours
C4.5	95.2	Quinlan
OC1	94.7 ± 3.1	Murthy
CART-AP	93.8 ± 3.7	Murthy
CART-LC	93.5 ± 2.9	Murthy
C1-AP	92.7 ± 2.4	Murthy

Iris original data using intervals equals to 0.1 for all attributes produced 81.1% accuracy. If attributes are ordered as: < *petal length, petal width, sepal length, sepal width*, >, accuracy increased to 92.6%. If file was ordered and intervals were increased from 0.1 to 0.2 accuracy increased to the figure shown in this table.

7 Implementation

We have implemented *Trie-CLASS*, an instance-based learning algorithm [18] using a trie-like structure used to hold patterns from R_T. Tries are essentially M-ary trees whose nodes are M-index vectors [14]. The key structure guides the branching. In our case, we index (or subscript) on patterns. Thus, pattern values of $p = \acute{a}p_1, p_2,..p_i,..p_n\rangle$ are inserted one by one into the trie with branching on p_i. To hold attribute domain intervals, each trie node is divided into an array without using any compression. See [1] and [2]. Array size is obtained applying the *equal width* discretization heuristic[4]. The number of intervals is calculated as j = (upper - lower) / interval. The size of the interval determines the array size and is taken from a dictionary. Each trie node's cell holds the triple $p = \acute{a}q_i, nlq_i, l_i\tilde{n}$. Value p_i is indirectly represented by its array position. Value l_1 is a counter holding the frequency of sub-pattern q_i. Class indicator nlq_i shows the evolution between class overlapping situations. For two identical sub-patterns associated with more than one class, the indicator is undefined. Otherwise it represents sub-pattern(s) always related to the same class label.

8 Conclusion

A new classification algorithm using normalized instance-based patterns has been used. Tests on several datasets have shown very accurate results. In fact, results are better than all those compared in presented tables including the DNA dataset. See table in [3].

[4] Ching [6] calculates for each attribute i Ai = m / (3 * C); m = number of training records.

We think this confirms that local hypotheses generated for each new record to be classified, produce more accurate results than one single hypothesis function. Incorporating into the decision algorithm exclusive values, the strength of patterns and their weight has increase accuracy predictability up to 3-4% in some cases we have tested.

References

1. Andersson, A, Nilsson, S., Faster Searching in Tries and Quad trees – An Analysis of Level Compression, in Proceedings of the Second European Symposium on Algorithms, 1994, pp 82-93
2. Aoe, J, et al. A trie compaction algorithm for a large set of keys, IEEE Transactions on Knowledge and Data Engineering, vol.8, nº 3, June 1996
3. Bin Liu, Y. Ma, and C.K. Wong, Improving an Association Rule Base Classifier, School of Computing, National University of Singapore (Web)
5. Cheeseman, P., Stutz, J., Bayesian Classification (AutoClass): Theory and Results, in Proc. Fifth Int. Conf. Machine Learning, California, 1988
6. Ching, J. Y.,Wong, A.K.C. & Chan, K.C.C, Class-dependent discretization for inductive learning from continuous and mixed-mode data. IEEE Trans. on PAMI, 17:641-651, 1995
7. Cios, K., W. Pedrycz, R. Swiniarski, Data Mining Methods for Knowledge Discovery, Kluwer Academic, 1998
9. Duda, R., Hart, P., Pattern classification and scene analysis, New York: John Wiley & Sons, 1973
10. Elder J. F., Abbott D. W. A Comparison of Leading Data Mining Tools, 4th Int. Conf. KDD, 1998, N. York, US
11. Everitt, B. S. & Hand, D.J. Finite Mixture Distributions, London: Chapman and Hall, 1981
12. Fredkin, E. Trie Memory Comm. ACM, vol.3 nº9, pp 490-500, 1960
13. Goldberg, D., Genetic Algorithms in search, Optimization and Machine Learning, Addison-Wesley, Reading, MA.,1989
14. Johnson, R.A., Wichern, D. W. Applied Multivariate Statistical Analysis, Prentice-Hall,1998
15. Knuth, D., The Art of Computer Programming, Vol. 3, Sorting and Searching, Addison-Wesley, 2nd. Ed,1997
16. Michalski, R., Mozetic, I., Hong, J., and Lavrac, N. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. Proceedings of the 5th National Conference on Artificial Intelligence. Philadelphia, Morgan Kaufmann 1041-1047, 1986
17. Mitchell, M., An Introduction to Genetic Algorithms, MIT Press, 1996
18. Mitchell, T. Machine Learning, McGraw Hill, 1997
19. Murphy,P.M.,&AhaDW.(1994).UCI Repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science. Patrick M. Murphy (Repository Librarian)
20. Murthy, K. V. S., On growing Better Decision Trees from Data, Ph.D. Thesis, University Johns Hopkins, BA., USA, 1996, 288 pp
21. Murthy S., Kasif, S., Salzberg, S., A System for Induction of Oblique Decision Trees, Journal of Artificial Intelligence, Research 2 (1994) 1-32
22. NCU Nicholas Copernicus University, Department of Computer Methods, Computational Intelligence Laboratory, Poland. (http://www.phys.uni.torun.pl/projects/datases.html)
23. Quinlan, J. R., Induction of Decision Trees. Machine Learning Journal 1(1):81-106,1986
24. Quinlan, J.R. MiniBoosting Decision Trees, enviado para publicación en Journal of Artificial Inteligence, 1998
25. Quinlan, J. R., Bagging, Boosting, and C4.5, University of Sydney
26. Ripley B.D, Pattern Recognition and Neural Networks, Cambridge University Press, 1996
31. Titterington, D.M., Smith, A.F.M.; and Makov, U.E. Statistical Analysis of Finite Mixture Distributions, 1985, New York: John Wiley & Sons
32. Usama M. Fayyad, Automating the Analysis and Cataloging of Sky Surveys, in Advances in Knowledge Discovery And Data Mining, Usama Fayyad et.al. AAAI Press/ The MIT Press, 1996