IDIAP RESEARCH REPORT

IDIAP

Martigny - Valais - Suisse

Data binarization by discriminant elimination

Miguel Moreira [1] Alain Hertz [2]
Eddy Mayoraz [1]

IDIAP–RR 99-04

January 1999

revised in May 99

published in
Proceedings of the ICML-99 Workshop: From Machine Learning to Knowledge Discovery in Databases

Dalle Molle Institute for Perceptual Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 − 27 − 721 77 11
fax +41 − 27 − 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

[1] IDIAP—Dalle Molle Institute of Perceptual Artificial Intelligence, P.O. Box 592, CH-1920 Martigny, Switzerland
[2] Département de mathématiques, EPFL, CH-1015 Lausanne, Switzerland

IDIAP Research Report 99-04

Data binarization by discriminant elimination

Miguel Moreira   Alain Hertz   Eddy Mayoraz

January 1999

revised in May 99

published in
Proceedings of the ICML-99 Workshop: From Machine Learning to Knowledge Discovery in Databases

Abstract. This paper is concerned with the problem of constructing a mapping from an arbitrary input space Ω into a binary output space $\{0,1\}^d$, based on a given data set $\mathcal{X} \subset \Omega$ partitioned into classes. The aim is to reduce the total amount of information, while keeping the most relevant of it to the class partitioning. An additional constraint to our problem is that the mapping must have a simple interpretation. Thus, each of the d discriminants is related to one original attribute (e.g. linear combinations of original attributes are not admitted). Besides data compression, the targeted application is the preprocessing for classification techniques that require Boolean input data. While other existing techniques for this problem are constructive (increasing d iteratively, such as decision trees), the method proposed here proceeds by starting with a very large dimension d, and by reducing it iteratively.

Keywords: Data binarization, classification, logical analysis of data, data compression.

Acknowledgements: The support of the Swiss National Science Foundation under grant 2000-053902.98 is gratefully acknowledged.

1 Introduction

The problem addressed in this paper deals with the transformation of data of arbitrary form into binary data. In such terms, this is a very general issue. However, the application targeted here is the *preprocessing of data for a supervised learning task*, and this specifies the general goal in several respects. This preprocessing is a requirement of all learning techniques capable of handling boolean data only.

1.1 Supervised learning

Given a set $\mathcal{X}$ of data belonging to an input space Ω, it is not sufficient to associate a Boolean image to each point $\mathbf{x}$ of $\mathcal{X}$, but an explicit mapping m from Ω into $\{0,1\}^d$ must be constructed in order to get Boolean images for any other point in Ω.

Assuming that a distance measure d_Ω is defined on Ω, a simple solution to this problem could be a vector quantization of the input space, leading to a code-book $C \subset \Omega$, followed by a binary coding of the code-book [Gersho and Gray, 1992]. However, to be satisfactory, the mapping $m : \Omega \rightarrow \{0,1\}^d$ should fulfill the *neighborhood constraint*, in the sense that $d_\Omega(\mathbf{x}, \mathbf{x}')$ should be small if the Hamming distance between $m(\mathbf{x})$ and $m(\mathbf{x}')$ is small. This constraint complicates significantly the binary encoding of the code-book.

Moreover, conventional vector quantization is ruled out by the fact that the underlying learning task is supervised, typically a classification or a regression problem. Let F denote the unknown target function of the learning task, with images either in a discrete unordered set in the case of a classification problem, or in $\mathbf{R}$ in the case of a regression problem. The neighborhood constraint mentioned above is subsumed by the *consistency constraint*, stating that $m(\mathbf{x}) \neq m(\mathbf{x}')$ if $F(\mathbf{x})$ and $F(\mathbf{x}')$ differ (classification) or are "sufficiently" distant (regression).

Regarding consistency as it is defined above, it can be discussed whether the imposition of such a constraint is of interest in supervised learning, as it can lead to over-fitting. However, in the current framework we deal with a preprocessing procedure. Consistency is used as an information-preserving mechanism. It is assumed that the over-fitting problem is handled by the algorithm using data binarization as a preprocessing step.

1.2 Easy interpretation

Other constraints imposed by the applications in focus are that the attributes of the input space can be of any kind (binary, enumerated ordered or unordered, and continuous) and the final model produced by the global learning method must have a clear and simple interpretation. Consequently, each one of the d binary functions of the mapping $m : \Omega \rightarrow \{0,1\}^d$ must involve only one original attribute of the input space Ω. In the sequel, each one of the d binary functions $\Omega \rightarrow \{0,1\}$ composing the binary mapping m is called a *discriminant* and it is restricted to the following types.

When associated to an unordered (or binary) attribute, which is necessarily enumerated, a discriminant is identified to one possible value of this attribute. As an example, to the attribute *color*, taking the values $\{red, yellow, blue\}$, can correspond a discriminant of the type "$color = yellow$". In the case of an *ordered* attribute, a discriminant can be interpreted both analytically as a threshold placed on an attribute value, and geometrically as an hyperplane orthogonal to the corresponding attribute axis, splitting the whole input space into two sub-spaces. So, given an ordered attribute *age* for example, a possible discriminant could be "$age > 45$", taking the value 0 (false) for any point of Ω for which the value of the original attribute *age* is below or equal to 45, and 1 (true) otherwise.

Obviously, restricting the mapping to such sharp discriminants could be a weakness in the case of noisy data. In practice, the mapping we construct is from Ω into $\{0, 1, *\}^d$, where $*$ denotes a "don't know", i.e. neither true nor false, activated whenever the observed value is too close to the discriminant. A possible discriminant $age > 45$ is for example one taking value 0 if $age < 43$, value 1 if $age > 47$, and value $*$ for data with attribute age between 43 and 47. In this setting, the consistency constraint requires that if $F(\mathbf{x})$ and $F(\mathbf{x}')$ differ, $m(\mathbf{x})$ and $m(\mathbf{x}')$ must have at least one component in which one is 1 and the other is 0.

1.3 Overview of the paper

It is now possible to state clearly the objective pursued here, which consists of finding a set of discriminants $\mathcal{D}$ consistent on a given data set $\mathcal{X}$ that allows the transformation of $\mathcal{X}$ into another data set $\mathcal{B}$ of the same cardinality but where all the attributes are binary. At the same time, the cardinality of $\mathcal{D}$ should be minimal, while the efficiency of the procedure should be maximal. Although the generalization to the case of regression is possible, the results presented here concern classification tasks exclusively.

Section 2 discusses related work and existing approaches. In Sect. 3 a new algorithm is proposed to find the set D for the binary mapping, which is compared experimentally with Simple-Greedy and a decision tree algorithm in Sect. 4. In Sect. 5 a possible improvement of the algorithm is suggested. Section 6 concludes and discusses further work.

2 Related work

2.1 Minimum set covering

Given a training set $\mathcal{X}$ of samples partitioned into classes, and given a large set $\mathcal{D}$ of D discriminants defining a binary mapping consistent with $\mathcal{X}$, the problem of finding a small subset of $\mathcal{D}$, still consistent with $\mathcal{X}$, can be formalized as a minimum set covering problem. Let $\boldsymbol{A}$ be the matrix with one column per possible discriminant and one row per pair of points that has to be distinguished according to the consistency constraint, $A_{(\mathbf{x},\mathbf{x}'),T} = 1$ if the discriminant T separates the points $\mathbf{x}$ and $\mathbf{x}'$ and is 0 otherwise. A subset of discriminants defines a binary mapping consistent with $\mathcal{X}$ if and only if its characteristic vector $\boldsymbol{z} \in \{0, 1\}^D$ satisfies $\boldsymbol{A}\boldsymbol{z} \geq 1$.

An heuristic is used to solve this minimum set covering problem and find a small set of discriminants satisfying the consistency constraint: Discriminants are selected iteratively, preference is given to those with larger conflict-solving power, measured by the quantity of 1's in the corresponding column of $\boldsymbol{A}$. For each selected discriminant the affected rows are eliminated from $\boldsymbol{A}$, and the process is repeated until the matrix is empty. A straightforward implementation of this greedy heuristic has a complexity in $O(N^2Dd)$, where $N = |\mathcal{X}|$ and d is the size of the final subset [Boros *et al.*, 1996]. In recent work, E. Boros uses column generation techniques to avoid storing this constraint matrix in memory. The quadratic nature hinders its application to problems of moderate-to-large size.

Almuallim and Dietterich [1994] propose an alternative approach to solve this minimum set covering problem in a lower computational complexity. In order to avoid considering pairs of data points, they keep track of the clusters of data that are fragmented by the discriminants selected so far. The discriminants are added in an iterative way until no clusters containing data points from different classes remain, which means that consistency is attained. At each iteration all the not-yet-selected discriminants are tested in order to select the best one, according to a merit function.

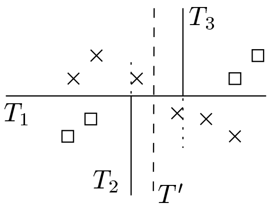
In there paper, they propose essentially three different merit functions associated to a discriminant T, given a subset of already selected discriminants. The first one measures how the entropy would vary with the addition of T. The second measure, called *Simple-Greedy*, counts the number of pairs of points from different classes and not yet seperated that T distinguishes. This measure is identical to the one of the greedy heuristic discussed above [Boros *et al.*, 1996]. A third measure is similar to Simple-Greedy and is called *Weighted-Greedy* because to each pair discriminanted by T and not yet separated, a weight is associated, inversely proporitonal to the number of other discriminants separating this same pair.

The results reported in [1994] show that the entropy measure is slightly less performant than Simple-Greedy, and Weighted-Greedy improves slightly the Simple-Greedy measure. However, only entropy and Simple-Greedy can be implemented very efficiently in $O(NDd)$. This is done by calculating the merit of a discriminant inside each remaining subset of data, and accumulating over all subsets. During subset splits, those containing data points from a single class are eliminated. More details about this algorithm are given in Sect. 4.

It is worth noting that in many applications it turned out that getting a subset of discriminant close to minimal and satistying the consistency constraint reduced too much the information of the original data. An interesting generalization of the consistency is the c-consistency, requiring that for each pair of data from different classes, there are at least c discriminants separating them. This generalization provides a parameter to control the final number of discriminants in the solution, while ensuring that as much information as possible for the classification problem is preserved. Unfortunately, there are no simple ways to generalize Simple-Greedy in order to handle c-consistency problems efficiently. On the contrary, the extention to c-consistency constraints of the approach implemented in [Boros *et al.*, 1996] is straightforward.

2.2 Local procedures

The resolution of the binarization problem is obtained as byproduct by several classical algorithms addressing directly the supervised classification problem. When a standard decision tree technique is run without early stopping criterion and without pruning, all the data of $\mathcal{X}$ getting to one particular leaf are of the same class. Thus, the set of discriminants associated to each node satisfies the consistency constraint. The drawback of this method, when used only for discriminant generation, is that it can produce many redundant discriminants due to its local character, since at lower levels of the tree only small fractions of the data set are used to determine each new discriminant. A simple situation highlighting this problem is depicted in Fig. 1, where a typical decision tree would have a root node with discriminant T_1 and left and right nodes with discriminants T_2 and T_3, while a global analysis could consider simply T_1 and T' as a solution. Of course, the fact that at lower levels of the tree the



Discriminant sets generated:

- local procedure: $\{T_1, T_2, T_3\}$
- global procedure: $\{T_1, T'\}$

Figure 1: Example of discriminant sets generated for a small, two-dimensional, 2-class data set.

analysis deals only with small sets of data is extremely beneficial to the computational time.

Among the other algorithms that produce as byproduct a consistent set of discriminants, but suffering from the locality problem, let us mention the nearest hyper-rectangle approaches [Salzberg, 1991, Wettschereck and Dietterich, 1995]. In this case, the discriminants derive from the edges of the hyper-rectangles.

2.3 Feature discretization

The approaches of Sects. 2.1 are deeply related to the problem discussed here. With lesser affinity, additional related work can be found in the framework of *feature discretization*, which is a preprocessing technique involving the transformation of continuous-valued features into discrete-valued ones. This is a requirement of some learning algorithms, mostly based on decision rules and decision trees. For algorithms handling continuous-valued attributes, preliminary discretization may accelerate the induction process, simplify the obtained models, and even improve accuracy ([Pfahringer, 1995], [Dougherty *et al.*, 1995]). Binarization differs from feature discretization in the sense that the former produces possibly several binary attributes from the same original attribute.

2.4 Feature selection

Yet another domain where similar approaches can be found is *feature selection*. The general goal consists in selecting subsets of attributes (features) by rejecting those that are irrelevant or misleading for the learning task and keeping the most relevant ones. The main motivations include accuracy improvements, reduced training times, and simpler concept models. In [John *et al.*, 1994], existing procedures within this domain are classified into *filter* and *wrapper* methods. The latter use the accuracy of the learned models as objective function during the subset search, while the filter methods consist strictly of preprocessing steps using only the training data. Reducing the initially generated discriminant set, as described in the beginning of the current section, fits in the filter group.

3 The IDEAL algorithm

The approach proposed here is called *Iterative Discriminant Elimination Algorithm (IDEAL)*. As opposed to alternative approaches mentioned in Sect. 2, it is an eliminative algorithm, as it starts with an exhaustive discriminant set that is reduced in the course of the execution. It is mainly motivated by the limitation of the overall computation effort by means of data structures that hold auxiliary information.

3.1 Binarizing a data set

For the binarization of a data set, a three-stage procedure is here considered: first, an exhaustive set of discriminants is generated. This set is then simplified by a discriminant elimination procedure that reduces its redundancy while keeping its consistency with regard to the data set at hand. Finally, the data set is binarized using the obtained solution. The IDEAL algorithm is thus wrapped by the **Binarize** procedure:

$\mathcal{X}$: set of data points
$\mathcal{D}$: set of discriminants
$\mathcal{B}$: set of data points in binary format

```
procedure Binarize ( 𝒳 )
   𝒟 := InsertDiscriminants( 𝒳 )
   𝒟 := IDEAL( 𝒳, 𝒟 )
   ℬ := ApplyDiscriminants( 𝒟, 𝒳 )
   return( ℬ )
endProc
```

Note that during classification of unseen data, the latter are binarized by means of the mapping obtained with the training data.

3.2 The segment

Figure 2 illustrates the manner in which discriminants are inserted initially in the case of ordered attributes, one between every two consecutive projected data points from different classes. For enumerated unordered attributes, a discriminant is created for each verified attribute value, which discriminates points with the corresponding attribute value from all the others.

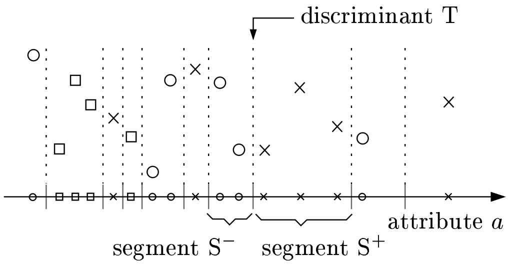


Figure 2: Example of the insertion of discriminants along an ordered attribute of a two-dimensional, 3-class data set.

The discriminants relating to the same attribute are positioned in an ordered manner, so that it is possible to refer to the preceding and succeeding discriminants of a discriminant T. The same holds for segments. We will refer to the *lower* and *higher adjoining* segments of T as the two segments lying between T and respectively its preceding and succeeding discriminants (S^- and S^+ in the figure). Likewise, the notion of *adjacent* segments of a segment and *adjacent* discriminants of a discriminant will be applied. These relations hold exclusively in the realm of ordered attributes. For the unordered ones, besides the lack of order, there is a bijective correspondence between the set of segments and the set of discriminants of the same attribute, with each discriminant *overlapping* its respective segment. The set of segments $\mathcal{S}$ is thus derived from the existing discriminant and data sets.

3.3 Discriminant elimination

The elimination is based on an iterative procedure where, at each iteration, a candidate discriminant is subject to a redundancy test, which determines whether it is eliminated or kept. The order in which the discriminants are selected is discussed in the next section. For the time being, let us consider that the discriminants are ranked in the beginning according to some weighting, and that at each iteration the one with the least weight is the candidate for elimination, each discriminant being allowed to be candidate only once.

In the framework of ordered attributes, let us define the *local conflicts* solved by a discriminant T as the pairs of points from different classes lying in its two adjoining segments, i.e., those pairs that are separated only by T along the corresponding dimension. The redundancy check for T consists basically of the following procedure: for each one of its conflicts, test if the pair of points is already separated by at least one discriminant in another dimension. If the outcome of the test is positive for all the pairs of points in conflict, T is considered redundant and can thus be eliminated. This is exposed in the pseudo-code below.

The procedure **SeparatedElsewhere** verifies the existence, in another dimension, of a discriminant separating the two points $\mathbf{x}^-$ and $\mathbf{x}^+$, which corresponds to testing if there is a dimension for which the two points lie in different segments. Note that, for a given dimension, if $\mathbf{x}^-$ or $\mathbf{x}^+$ have unknown value, they are considered non-separated therein.

The efficient execution of the algorithm relies to a great extent in the existence of information representing the sequential positioning of segments and discriminants of the same attribute, as well as the membership of each data point to its corresponding segment for each attribute dimension. The latter avoids searching all discriminants of an attribute to find those separating a particular pair of points, inside **SeparatedElsewhere**. The usefulness of these informations extends to the updating operations effectuated after each discriminant elimination.

$\mathcal{S}$	: set of segments
S	: segment $\equiv$ set of data points
T	: discriminant (weighted)
$\mathbf{x}$	: data point
$w(T)$	: weight of discriminant T

```
procedure IDEAL ( 𝒳, 𝒟 )
  𝒮 := InsertSegments( 𝒳, 𝒟 )
  while ∃T with w(T) < ∞
    T := arg min( w(T) )
    S⁻, S⁺ := adjoining segments of T
    redundantDiscr := TRUE
    forEach pair ⟨ x⁻,x⁺⟩,  x⁻ ∈ S⁻,  x⁺ ∈ S⁺,  F(x⁻) ≠ F(x⁺)
      if not SeparatedElsewhere( x⁻,x⁺ )
        redundantDiscr := FALSE
        abort loop
      endIf
    endFor
    if redundantDiscr
      S^new := S⁻ ∪ S⁺
      𝒮 := 𝒮 \ {S⁻, S⁺} ∪ {S^new}
      𝒟 := 𝒟 \ {T}
      T⁻, T⁺ := adjacent discriminants of T
      w(T⁻) := WeightDiscriminant( T⁻ )
      w(T⁺) := WeightDiscriminant( T⁺ )
    else
      w(T) := ∞
    endIf
  endWhile
  return( 𝒟 )
endProc
```

For the sake of simplicity, the pseudo code presented does not take into account the occurrence of unordered, enumerated attributes. For these, a discriminant and a corresponding segment are created for each attribute value. This situation carries with it a certain redundancy, since any two points of different classes are separated by the discriminants of both. It can be said that none of those discriminants is non-redundant. When such a discriminant is eliminated, its segment is also eliminated, and the corresponding points are positioned in a so-called "neutral segment", which receives all the points coming from the eliminated segments of the attribute to which it belongs. In this framework, the number of conflicts of a discriminant from an unordered attribute is always calculated based on its segment, on one side, and the neutral segment for that dimension on the other.

3.4 Weighting the discriminants

The greedy nature of the elimination procedure causes the order in which the candidate discriminants are chosen to be of major importance. With the assumption that a discriminant is charged of making separations between points from different classes and that the most separations it does, the most important it is, it seems logical to try to eliminate first the discriminants charged of fewer separations, because of three reasons: first, the task of verifying their redundancy will be easier, since there will be less conflicts to test; then, given that these discriminants will be most probably redundant, that task will be better payed back; finally, and most importantly, given that the minimal amount of required separations is fixed, fewer discriminants will remain in the end, i.e. those that "work the most". Two main formulations have been adopted to define the importance of a discriminant:

Number of local conflicts As defined in Sect. 3.3.

Entropy Given $p_k^{\mathcal{X}}$ as the relative frequency of class k in a data set $\mathcal{X}$ with K classes, the *entropy* $e(\mathcal{X})$, of the data can be computed as

$$e(\mathcal{X}) \;=\; -\sum_{k=1}^{K} p_k^{\mathcal{X}} \;\ln p_k^{\mathcal{X}} \;.$$

If a discriminant T separates the data set $\mathcal{X}$ into two subsets $\mathcal{X}^-$ and $\mathcal{X}^+$, then the split entropy $e_{split}(\mathcal{X},T)$ is

$$e_{split}(\mathcal{X},T) \;=\; \frac{|\mathcal{X}^-|}{|\mathcal{X}|}\, e(\mathcal{X}^-) \;+\; \frac{|\mathcal{X}^+|}{|\mathcal{X}|}\, e(\mathcal{X}^+) \;,$$

which is a weighted sum of the two sub-entropies. The *gain* $g(T)$ conveyed by a discriminant can simply be computed as:

$$g(T) \;=\; e(\mathcal{X}) \;-\; e_{split}(\mathcal{X},T) \;,$$

and applied as a weighting measure. This criterion is successfully used by Quinlan in the decision-tree building algorithm C4.5 [Quinlan, 1993] for the choice of the splits.

The locality of the measure based on the number of conflicts (segment level) contrasts with the global nature of the entropy, defined over the whole data set. No clear indication exists, however, about the advantages of a local measure over a global one or vice-versa, fact that lead to the development of the alternative versions of each of the two measures: A local version of the entropy, calculated based on the two adjoining segments of T, as well as the calculation of the number of conflicts between the two complete subsets of $\mathcal{X}$ generated by the split in T.

Table 1 contains results on the size of the discriminant set obtained by each one of the four discussed weighting procedures. In addition, the criterion of random choice has also been included in the tests for comparison. This is justified by the fact that none of the other proposed criteria is guaranteed to be meaningful in the particular framework of this algorithm. In the latter case, 50 experiments were made with different random seeds and the results averaged. The data sets used are available from the UCI repository of machine learning databases [Blake *et al.*, 1998]. A summary is provided in Table 3. Since no training/testing split was used in the reported experiments, the sets were applied as a whole as input for the binarization procedure.

Here it should be emphasized that the weights of the discriminants are permanently up-to-date, since any discriminants affected by an elimination are instantly reweighted, as is shown in the pseudo-code of Sect. 3.3. However, for the global measures the weights are static and need only to be calculated once in the beggining of the procedure.

Overall, the number of conflicts seems to provide a better measure than the entropy, and the effect of a local or a global measure depends of the specific weighting method used. Number of local conflicts and global entropy appear as the best combinations.

According to the results, the choice of the best weighting method depends on the particular features of the data set at hand. However, the observed differences can only be considered significant after multiple run experiments with random data subsets and a statistical test. The local conflict measure is adopted in the remaining experiments.

Table 1: Comparison between the final discriminant set size obtained with different discriminant weighting measures. The values in bold indicate the winning measure in the corresponding row, random choice excluded. The penultimate row contains the percentage of the initial set size achieved with each procedure, averaged over all the data sets. The bottom row shows the number of wins, including ties.

	Initial	Final set size					
	set	n.conflicts		entropy		random choice	
data set	size	local	global	local	global	avg.	min.
abalone	5779	192	**177**	724	220	282.6 ±21.6	240
allhyper	440	**21**	33	36	34	30.4 ±4.3	22
allhypo	548	**20**	32	43	48	35.2 ±6.2	17
anneal	134	31	31	**30**	**30**	30.9 ±2.1	28
audiology	92	23	26	24	**21**	27.6 ±3.2	21
car	15	**14**	**14**	**14**	**14**	14.0 ±0.0	14
dermatology	141	**13**	14	14	15	17.9 ±1.6	13
ecoli	301	**24**	**24**	30	26	27.3 ±2.1	23
glass	692	**17**	19	49	25	22.4 ±1.8	18
heart-disease	309	**14**	17	21	18	20.0 ±2.8	14
krkopt	39	**38**	**38**	**38**	**38**	37.6 ±0.7	36
letter	234	59	**58**	62	59	65.9 ±2.6	61
mushroom	112	**7**	8	12	12	11.7 ±3.0	8
nursery	19	**17**	**17**	**17**	**17**	17.0 ±0.0	17
page-blocks	3378	**45**	67	83	53	53.2 ±2.7	47
pi-diabetes	856	**24**	26	57	43	31.9 ±2.3	27
segmentation	9817	**28**	50	151	47	40.4 ±6.8	31
soybean	97	36	36	**34**	35	39.5 ±3.0	34
vehicle	1215	**34**	39	111	54	48.3 ±5.1	37
vowel	7077	**26**	46	196	83	34.1 ±3.1	30
yeast	374	**39**	42	68	48	49.4 ±3.5	43
avg. % of initial size		21.5%	22.1%	24.0%	22.5%	23.1%	
number of wins		16	6	5	5		

4 A comparison between approaches

Experimental comparisons were made between IDEAL, Simple-Greedy (SG) and a decision tree algorithm (DT) which in the present case consisted of Quinlan's C4.5 [Quinlan, 1993]. Details of the implementation of each algorithm are given below.

Simple-Greedy This algorithm was originally designed to handle two classes, one negative and one positive. The merit function calculated for a discriminant T is therefore:

$$w(T) \;=\; \sum^{P\,\in\,\mathcal{P}} p_P^{T=1}\cdot n_P^{T=1} \;+\; p_P^{T=0}\cdot n_P^{T=0},$$

where P is a data subsample from the set $\mathcal{P}$ of subsamples, which at the first iteration contains the single element $\{\mathcal{X}\}$. The quantities p_P and n_P are the number of points in subsample P of the positive and negative class respectively. $w(T)$ represents the number of conflicts left to be solved by T and is thus to be minimized. We now generalize to the case of K classes:

Simple-Greedy does not consider the existence of points with unknown values for some attributes. Originally, when a subsample P is split by a discriminant T, the points for which T=1 are sent to one side of the split and those for which T=0 to the other. A modification was introduced which consists in sending all points for which T=? to both sides of the split. This has some effect in the calculation of the merit function.

We can now generalize the merit function to the case of K classes and the eventual presence of unknown values. To simplify the notation, we define q and q' as the number of points in P from class

k and k' respectively.

$$w(T) = \sum^{P\in\mathcal{P}} \sum_{k=1}^{K-1} \sum_{k'=k+1}^{K} \begin{array}{l} q^{T=1}(q'^{T=1} + q'^{T=?}) + \\ q^{T=0}(q'^{T=0} + q'^{T=?}) + \\ q^{T=?}(q'^{T=0} + q'^{T=1}) + \\ 2\ q^{T=?}\ q'^{T=?}\ . \end{array}$$

This formula calculates the number of unsolved conflicts taking into account the fact that each data point for which T is unknown is added to both split sides of a subsample.

Decision tree For C4.5 to be used as a discriminant set generator, the trees must be fully-grown, i.e. no errors are allowed in the training set, even if certain leafs contain a single data point, and no pruning is allowed either. These conditions comply with the consistency constraint and the program was adjusted accordingly. From the resulting tree each node is considered a discriminant. However, two or more equal nodes at different levels of the tree are merged into one single discriminant.

Table 2 shows the results of the experiments both in terms of final discriminant set size and execution time. It is interesting to note that Ideal and SG achieve quite comparable results

Table 2: Resulting discriminant set size and execution time of the compared approaches: Ideal, Simple-Greedy (SG) and decision tree (DT).

data set	initial set size	final set size: Ideal	SG	DT	execution time: Ideal	SG	DT
`abalone`	5779	192	171	1282	20.5	195×10^3	24.2
`allhyper`	440	21	18	60	56.6	16.3	2.1
`allhypo`	548	20	19	40	58.1	14.6	1.0
`anneal`	134	31	33	107	2.8	6.2	1.4
`audiology`	92	22	22	71	0.8	10.4	0.5
`car`	15	14	14	22	2.1	0.3	0.1
`dermatology`	141	13	13	17	4.9	1.2	0.3
`ecoli`	301	24	20	49	0.3	5.9	0.2
`glass`	692	17	15	41	0.4	5.6	0.3
`heart-disease`	309	14	12	62	0.8	1.4	0.3
`krkopt`	39	34	34	48	34.1	217.6	5.7
`letter`	234	59	58	252	3497.3	2586.9	84.1
`mushroom`	112	7	6	30	1440.9	4.6	0.5
`nursery`	19	17	17	27	106.6	2.5	0.6
`page-blocks`	3378	45	39	144	108.8	396.1	12.7
`pi-diabetes`	856	24	22	123	2.1	17.0	0.8
`segmentation`	9817	28	24	72	121.0	698.8	5.6
`soybean`	97	25	22	105	6.5	43.9	0.7
`vehicle`	1215	34	26	137	14.5	44.9	1.9
`vowel`	7077	26	22	123	9.2	773.6	3.1
`yeast`	374	39	41	307	4.0	139.4	2.5
avg. % of initial size		20.0%	19.7%	*48.1%			

* Note that in this case the initial set is not the same as with the two other algorithms. It can even be observed that for some data sets the final size obtained with decision trees is larger than the initial value for the two other algorithms.

even though their search path is different, one being eliminative and the other constructive. This has probably some relation with a certain similarity between their merit functions.

Previous to the examination of the execution times, we proceed to the time complexity analysis of Ideal and SG. In the case of Ideal, $O(AN \log N)$ is needed to sort the attribute projections of data points and insert the segments, A being the number of attributes. The evaluation of the merit of each discriminant and sorting the list cost $O(K^2D + D\log D)$. The main part of the algorithm needs $O(DN^2A)$ to test the redundancy of all the discriminants in the initial set, test their discriminated pairs of data points over other attributes, and sort the discriminant according to the merit function. Considering that N is usually much large than K or $\log D$, the total complexity is

$$O(AN^2D)\ .$$

For SG, d discriminants are selected at most and for each one $O(D(NK^2)+N)$ is the bound for testing every available discriminant, calculating the merit function and splitting the data subsets. The total is then

$$O(dDNK^2)\ .$$

We note the quadratic bound in N for IDEAL, although it is a very loose bound. This aspect is further discussed in Sect. 5. We note also the quadratic bounds in D and K for SG, although it becomes quadratic in D only in case the resulting size d is close to the initial size D.

The execution time comparison between IDEAL and SG gives empirical support to the complexity analysis laid above. Although the time results do not follow blindly the provided formulas, some typical patterns can be identified and, with the aid of Table 3, the influence of the respective sensitive parameters can be observed. Namely, the combined influence of D and K over SG is apparent in `segmentation`, `soybean`, `vowel`, and `yeast`, being particularly striking in `abalone`. Cases where the value of N is considerable comparing to D and at the same time considerable loss is verified for IDEAL are `car`, `mushroom`, and `nursery`.

Beyond the effective time complexity, two conceptual issues establish the difference between these two procedures. One is the fact, already mentioned, that IDEAL is eliminative, while SG is constructive: SG is executed until consistency is attained, while in IDEAL consistency is never lost, if verified initially. (This may not be the case for datasets with missing values, e.g. `soybean`.) The other, very important aspect is that IDEAL is heavily grounded on the original attributes.

Concerning the comparison between IDEAL, SG, and decision trees, the numbers in the table emphasize the differences between a local approach, fast but with large final sets, and the two global ones, more time-consuming but with better final results.

5 A suitable stopping criterion

Despite the encouraging results obtained with the IDEAL algorithm in terms of discriminant set size, for some of the data sets included in the tests the minimization of the binary mapping still takes a considerable amount of time. However, in all the reported tests the algorithm has been given the freedom to run till the end, that is, to test the redundancy of all the discriminants in the initial set. Figure 3 shows the evolution of the elimination process with the conflict-based weighting method for four of the data sets. In the case of Letter Recognition, for example, the algorithm starts with 234 discriminants and takes almost 3500 seconds to find a final set with 59. However, after 1000 sec. only 69 discriminants remained, and after 3000 sec. 60. This phenomenon was observed for the majority of the data sets tested and it finds explanation in two main factors: the probability of finding redundant discriminants decreases with time, while the effort spent on each redundancy test increases, due to the fusion of segments that takes place at each discriminant elimination, and the consequent increase in the number of conflicts to be tested.

This observation demonstrates the need for a suitable stopping criterion as the execution time can, for some problems, be largely reduced at the cost of a minimal increase in the number of discriminants.

Table 3: Summary of the data sets used in the experiments. The rightmost column indicates the number of discriminants generated by the procedure **InsertDiscriminants** (see Sec. 3.1).

data Set	classes	attrib.	data points	initial discrim.
abalone	28	9	4177	5779
allhyper	5	29	3772	440
allhypo	4	29	3772	548
anneal	5	38	898	134
audiology	24	69	226	92
car	4	6	1728	15
dermatology	6	34	366	141
ecoli	8	7	336	301
glass	6	19	214	692
heart-disease	2	14	303	309
krkopt	18	6	28056	39
letter	26	16	20000	234
mushroom	2	22	8124	112
nursery	5	8	12960	19
page-blocks	5	10	5473	3378
pi-diabetes	2	8	768	856
segmentation	17	19	2310	9817
soybean	19	35	683	97
vehicle	14	18	846	1215
vowel	11	10	990	7077
yeast	10	8	1484	374

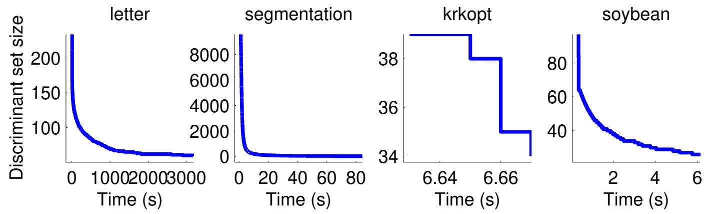


Figure 3: Evolution of the discriminant elimination procedure for four data sets with the local conflict-based weighting method.

In addition, it suggests a convenient method to implement the criterion, based on the gradient of the plotted function. Nevertheless, the asymptotic elimination behavior is non-uniform among problems and cannot, in reality, be granted. The two curves at the right of Fig. 3 serve as counter-examples. Another aspect to take into account is associated with the actual procedure used to calculate the variation of the discriminant set size. The plots of Fig. 3 are based on time, but this is not necessarily the best indication of the state of the process, an alternative being the number of past elimination iterations combined with the observed frequency of rejections (candidates that are not redundant).

Overall, the observation of the different behavior patterns provided by the figure indicates that a robust criterion for appropriately stopping the elimination procedure is not obvious and is yet to be determined. This subject should deserve considerable attention in the continuing work.

6 Conclusions and further research

The approach that is proposed here (IDEAL) is capable of finding, in the context of a supervised learning task, a consistent mapping from a set of arbitrary data into a set with a binary format. The quest for consistent mappings into Boolean spaces of small dimensions requires a global analysis of the data set, which implies high computational times comparing to local methods such as decision trees or nearest hyper-rectangles. However, the inclusion of heuristics that make extensive use of temporary auxiliary information allows this procedure to be accomplished in a time less than quadratic in the number of data. This represents an important result, as the algorithm constitutes a useful tool to be used in conjunction with learning methods that deal with data in binary format, as is the case of logical analysis of data [Boros *et al.*, 1996].

Empirical and analytical comparisons have been made between IDEAL and an alternative approach, Simple-Greedy[Almuallim and Dietterich, 1994]. Their differences have been emphasized, although in further work some of their concepts can possibly be exchanged or merged in order to attain better performances. While the Simple-Greedy algorithm proceeds by constructing a solution iterative, IDEAL prunes a large initial solution. In the former case, the process goes on until consistency is reached, while in the latter case, the rule is to prune without loosing consistency until no discriminant can be dropped. Simple-Greedy has in general a lower complexity than IDEAL. On the other hand, IDEAL allows to look for discriminant sets such that every pair of data from different classes are separated by at least c discriminants, where c could be any positive integer.

Another major difference between IDEAL and the other algorithms is that, due to its local merit function, IDEAL takes into account the original attributes information. In particular, the solutions proposed by IDEAL are biased towards discriminant sets well spread over the set of attributes, i.e. solutions were many discriminants are related to the same attribute are avoided, if possible.

Concerning further work, an important step to be made is to test the actual quality of the obtained discriminant sets, with the application of the binarized set for classification. The comparison between the approaches studied here has to be re-analyzed in light of the results produced by the classification procedure in which they are to be used.

As discussed in Sect. 5, a suitable stopping criterion can possibly improve the interest of the algorithm due to the considerable reduction of its execution time with the compromise of a slightly larger final discriminant set, also because the size of the discriminant set is possibly not the ultimate goal to attain, but rather the quality of the results produced on the learning task where it is to be applied.

Several particular subjects are already handled by the proposed algorithm but are not reported here, for the sake of simplicity. For example, the treatment of missing information can be quite straightforward but leads to three-valued logical data $(0, 1, *)$. The current version of the algorithm is also able to deal with discriminants along continuous attributes that assign the unknown value $*$ to continuous values that are too close to the cut point of the discriminant (see Sect. 1.2). In some applications with an ordered output, monotonicity constraints might be imposed between some ordered attributes and the output. For example, in a medical database designed to characterize the risk of heart attack, the expert might be interested in imposing that any dependency between the age and the risk of heart attack is monotonic positive. In these situations, it is essential to introduce these constraints already in the binarization procedure, as the definition of *consistency* is slightly modified as follows. For any two data points $\mathbf{x}$ and $\mathbf{x}'$ with outputs $F(\mathbf{x}) > F(\mathbf{x}')$, there must exist either a discriminant T without monotonicity constraint such that $m_T(\mathbf{x}) \neq m_T(\mathbf{x}')$, or a discriminant T, say with a positive monotonicity constraint, such that $m_T(\mathbf{x}) > m_T(\mathbf{x}')$.

Finally, one may argue that this algorithm is not robust towards outliers or points with incorrect outputs, given that if a point wrongly classified as class 1 lies in the middle of a cluster of points of class 2, several discriminants might be introduced only to isolate this point. When this binarization is used just as preprocessing for a classification method using binary data this is not critical, as long as the final classification method deals with such points adequately. The recognition of outliers into the binarization algorithm would however be a plus and it is a topic of further investigation.

References

[Almuallim and Dietterich, 1994] Hussein Almuallim and Thomas G. Dietterich. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69(1–2):279–306, 1994.

[Blake *et al.*, 1998] C. Blake, E. Keogh, and C.J. Merz. UCI repository of machine learning databases, 1998. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[Boros *et al.*, 1996] E. Boros, P. L. Hammer, Toshihide Ibaraki, A. Kogan, E. Mayoraz, and I. Muchnik. An implementation of logical analysis of data. RRR 22-96, RUTCOR–Rutgers University's Center For Operations Research, July 1996. `http://rutcor.rutgers.edu:80/~rrr/` Submitted.

[Dougherty *et al.*, 1995] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In Armand Prieditis and Stuart Russel, editors, *Machine Learning: Proceedings of the Twelfth International Conference*, San Francisco, CA, 1995. Morgan Kaufman Publishers.

[Gersho and Gray, 1992] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, 1992.

[John *et al.*, 1994] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In William W. Cohen and Haym Hirsh, editors, *Machine Learning: Proceedings of the Eleventh International Conference*, pages 121–129, San Francisco, CA, 1994. Morgan Kaufman.

[Pfahringer, 1995] Bernhard Pfahringer. Compression-based discretization of continuous attributes. In A. Prieditis and S. Russell, editors, *Proceedings of the 12th International Conference on Machine Learning (ICML'95)*, Los Altos/Palo Alto/San Francisco, 1995. Morgan Kaufmann.

[Quinlan, 1993] J. R. Quinlan. *C4.5 Programs for Machine Learning.* Morgan Kaufmann, 1993.

[Salzberg, 1991] Steven Salzberg. A nearest hyperrectangle learning method. *Machine Learning*, 6:251–276, 1991.

[Wettschereck and Dietterich, 1995] D. Wettschereck and T. G. Dietterich. An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning*, 19(1):5–28, 1995.