Decision Queue Classifier for Supervised Learning Using Rotated Hyperboxes

Jesús Aguilar, José Riquelme and Miguel Toro

Departamento de Lenguajes y Sistemas Informáticos.
Facultad de Informática y Estadística. Universidad de Sevilla.
E-mail: {aguilar, riquelme, mtoro}@lsi.us.es

Keywords: data mining, supervised learning, genetic algorithms.

Abstract. This article describes a new system for learning rules using rotated hyperboxes as individuals of a genetic algorithm (GA). Our method attempts to find out hyperboxes at any orientation by combining deterministic hill-climbing with GA. Standard techniques, such as C4.5, use hyperboxes that are aligned with the coordinate axes. The system uses the decision queue (DQ) as method of representing the rule set. It means that the obtained rules must be applied in specific order, that is, an example will be classify by the i-rule only if it doesn't satisfy the condition part of the i-1 previous rules. With this policy, the number of rules is less because the rules could be one inside of another one. We have tested our system on real data from UCI repository. Moreover, we have designed some two-dimensional artificial databases to show graphically the experiments. The results are summarized in the last section.

1 Introduction

Supervised learning (SL) is used when the data samples have known outcomes that the user wants to predict. This type of learning is the more common form because data are usually collected with some outcome in mind. Human problem solving is normally an exercise in studying input conditions to predict a result based upon previous experience with similar situations. SL algorithms tend to emulate that sort of human behavior.

Decision trees (DT) are a particularly useful tool in the context of machine learning

Helder Coelho (Ed.): IBERAMIA'98, LNAI 1484, pp. 326-336, 1998.

techniques because they perform classification by a sequence of simple, easy-to-understand tests whose semantics is intuitively clear to domain experts. Some techniques, like C4.5, construct decision trees selecting the best attribute by using a statistical test to determine how well it alone classifies the training examples [9]. This class of DTs may be called axis-parallel, because the tests at each node are equivalent to axis-parallel hyperplanes in the space. Others techniques build oblique decision trees (ODT), as OC1[7], that tests a linear combination of the internal attributes at each node, for that, these tests are equivalent to hyperplanes at an oblique orientation to the axes.

At this point, we must remember that, in a domain with N training examples, each described using a k real-valued attributes, there are at most $2^k \binom{N}{k}$ distinct k-dimensional oblique splits; however, for axis-parallel splits, there are only $N \times k$ distinct possibilities, for that reason, it could exhaustively search the best split at each node.

Anyway, to find out the smallest DT (axis-parallel or oblique) is a NP-hard problem [2]. Both methods use hill-climbing, that is, the algorithm never backtracks; therefore, it could be converging to locally optimal solutions that are not globally optimal.

Simpson [12] introduced the idea of using hyperboxes to cluster or classify spatial data. Each hyperbox is viewed as a fuzzy cluster, a fuzzy set in which all of the elements within the hyperbox have membership 1.0 for being in that set, and elements outside the hyperbox can have a positive membership in the set depending on a fuzzy membership rule for that set. Simpson used a deterministic procedure to place and appropriately size hyperboxes to describe data. Hyperboxes were created and sized by considering the data in an ordered sequence. A hyperbox was placed around preliminary data. As subsequent data was added, either the present hyperbox was grown to include the new data, or a new hyperbox was added and the process continued. This procedure was of limited efficacy because it required trial-and-error setting of operator parameters and the final solution depended on the order of presentation of the data, even when the data possessed only spatial an not sequential characteristics. Fogel and Simpson used evolutionary programming to optimize the position of hyperboxes to cluster data in light of a minimum description length criterion (MDL). First, the experiments were restricted to evolving hyperboxes that were aligned with coordinate axes; and afterwards, they included the capability to rotate the hyperboxes. At this point, it is important to note that Fogel's method try to solve the clustering problem, that is, unsupervised learning.

Genetic algorithms (GA) employ a randomized search method to seed a maximally fit hypothesis [3, 4]. This search is quite different from other learning methods, like mentioned above. The GA search can move much more abruptly, replacing a parent hypothesis by an offspring less likely to fall into the same kind of local minima that can happen with the other methods.

In previous works, we presented a system to classify databases by using hyperboxes (axis-parallel). This system used a GA to search the best solutions and produced a hierarchical set of rules. The hierarchy means that an example will be classify by the i-rule if it does not satisfy the conditions of the i-1 precedent rules. The rules are sequentially obtained until the space is totally covered. The behavior is similar to a

queue, for that reason we have called decision queue (DQ) to the produced rule set. This concept is based on the k-DL, the set of decision lists with conjuntive clauses of size at most k at each decision [11]. A decision list is a list L of pairs where each f_j is a term in C_k^n, each v_i is a value in {0,1}, and the last function f_r is the constant function true.

$$(f_1, v_1),...,(f_r, v_r) \quad \textbf{(1)}$$

A decision list L defines a boolean function as follows: for any assignment $x \in X_n$, L(x) is defined to be equal to v_j where j is the least index such that $f_j(x)=1$ (such an item always exists, since the last function is always true).

DQ is based on DL. Really, DQ is a DL-generalization because it permits codifying functions f_i of continuous attributes and the values v_i can belong to any set.

Futhermore, DQ does not have the last constant function true. However, we could interpret that last function as unknown function, that is, we do not know to which class the example belongs to. Therefore, it may be advisable to say "unknown class" instead of taking an erroneous decision.

In the sense mentioned above, our system has a measure, called unknowledge, to indicate how many test examples have not an associated class. As the number of rules or the allowed error rate (relaxing coefficient) can be given by the domain expert, some unnecessary mistakes could be avoided if the rule set does not assign to the test example a class. Incrementing the relaxing coefficient the unknowledge will be less, but the number of misclassified examples will be higher. The expert, based on experimentation, must determine such parameter.

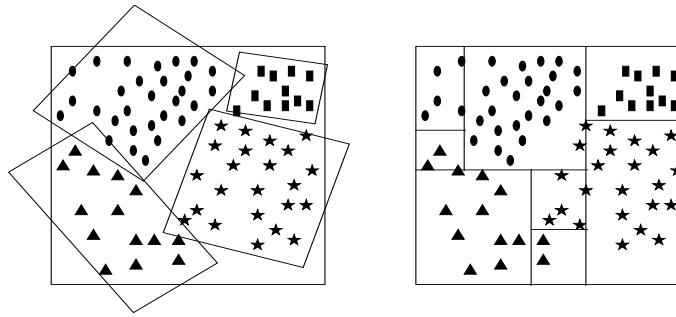

Fig. 1. Rotated versus axis-parallel hyperboxes.

In this paper, we propose to hold the primitive structure of our early works [1,10], but changing the shapes that models the search space. That is to say, current work extends these previous efforts by including the capability to rotate the hyperboxes.

We show in figure 1 an example, in which rotated hyperboxes can find out better solutions than axis-parallel hyperboxes they do. Decision queue policy is applied in order to reduce the number of rules.

With this DQ-method, there is no problem if the regions are overlapped. An extreme case is presented in the next figure.

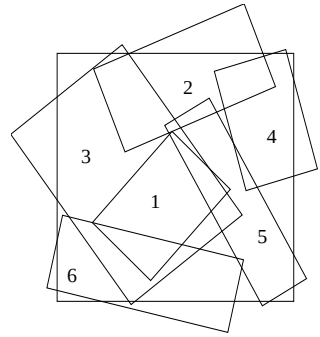

Fig. 2. Decision queues for overlapped regions.

In the other hand, if we use axis-parallel techniques, the number of rules is very high. When does it apply one technique or the other one? In principle, it is not possible to know it, but it could be a good solution to explore the search space with axis-parallel and, increasingly, to try to rotate the best solutions.

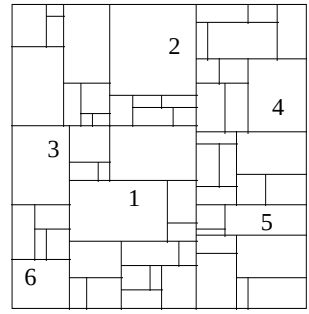

Fig. 3. Axis-parallel solution to the figure 2.

The number of rules, in figure 3, is very high. The numbers represents parts of the regions found out by using rotated hyperboxes, as shows figure 2.

2 Description

2.1 Environment

In order to apply GAs to a learning problem, we need to select an internal representation of the space to be searched and define an external function that assigns fitness to candidate solutions. Both components are critical to the successful

application of the GAs to the problem of interest. Information of the environment comes from a data file, where each example has a class and a number of attributes. The GA uses real codification; that is, an individual is formed by an n-tuple of real. If k is the dimension, an individual has exactly $3 \times k$ values: $2 \times k$ for the boundaries of each dimension; $k - 1$ for the angles of rotations, in radians, anti-clockwise around the hyperbox centre; and 1 for the class. The next figure shows a n-tuple:

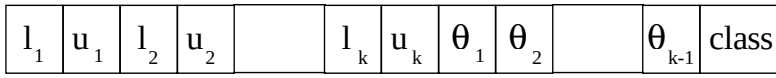

Fig. 4. Representation of an individual.

where l_i and u_i represent the lower and upper bounds of the individual, respectively, for every dimension; θ_i is the rotation angle; and class. In 2-dimension is possible to put an hyperbox at any orientation by using only one rotation; in k-dimension it is necessary k-1 rotations.

We consider that an example belongs to the area determined for an individual if it satisfies its condition part. Thus, let an example be given by $P_j=(p_1, p_2, ..., p_k, c)$ then it will be into the defined region by the individual (or equivalently, a rule will be covered by the rule) $ind_h = (l_1, u_1, l_2, u_2, ..., l_k, u_k, \theta_1, \theta_2, ..., \theta_{k-1}, class)$ if rotating the example $P=(p_1, p_2, ..., p_k)$ with the angles $-\theta_1, -\theta_2, ..., -\theta_{k-1}$ with relation to the centre of the hyperbox defined by $(l_1, u_1, l_2, u_2, ..., l_k, u_k)$, then the result P' belongs to this hyperbox.

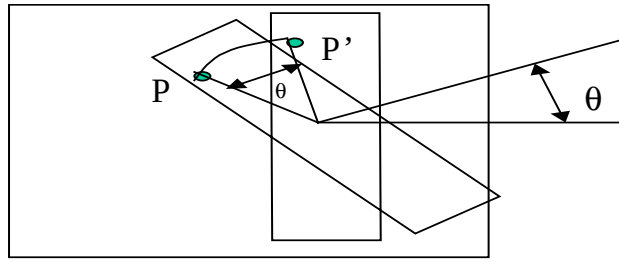

Fig. 5. Rotation of R the angle θ is equivalent to rotate P the angle -θ around the centre of R.

For example, in three dimensions the example (p_1, p_2, p_3, c) will be covered by the rule $(l_1, u_1, l_2, u_2, l_3, u_3, \theta_1, \theta_2, class)$ if P' satisfies

$$l_1 \le p'_1 \le u_1 \wedge l_2 \le p'_2 \le u_2 \wedge l_3 \le p'_3 \le u_3 \quad (2)$$

where P' is obtained as follows:

Let $(m_1, m_2, m_3) = ((l_1 + u_1)/2, (l_2 + u_2)/2, (l_3 + u_3)/2)$ be the centre of the hyperbox defined by the rule, then the coordinates of P' are:

$$(p'_1, p'_2, p'_3) = (p_1 - m_1, p_2 - m_2, p_3 - m_3) \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_1 & -\operatorname{sen}\theta_1 \\ 0 & \operatorname{sen}\theta_1 & \cos\theta_1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\operatorname{sen}\theta_2 & 0 \\ \operatorname{sen}\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} + (m_1, m_2, m_3 \quad \textbf{(3)}$$

and it will be correctly classify if its class is equal to c.

2.2 Algorithm

The algorithm is a typical sequential covering GA [6]. It chooses the best individual of the evolutionary process, transforming it into a rule, which is used to eliminate data from the training file [13]. In this way, the training file is reduced for the following iteration. A termination criterion could be reached when more examples to cover do not exist.

The method of generating the initial population consists of randomly selecting for every individual of the population an example from the training file. After, it is obtained an interval to which the example belongs adding and subtracting a random quantity from the values of the example. Moreover, the angles are randomly generated between zero and $\pi/2$. Sometimes, the examples very near to the boundaries are hard to cover during the evolutionary process. For solving it, the search space is increased (actually, lower bound is decreased 5%, and upper bound is increased 5%). For example in 1-dimension, let a and b be the lower and upper bounds of the attribute; then, the range of the attribute is b-a; now, we randomly choose an example (x_1, class) from the training file; last, a possible individual of the population could be:

$$(x_1 - range * k_1, x_1 + range * k_2, class) \quad \textbf{(4)}$$

where k_1 and k_2 are random values belonging to [0,1], and class is the same of that of the example.

The evolution module includes elitism: the best individual of every generation is replicated to the next one. A set of children is obtained from copies of the parents, randomly selecting it, but depending on their fitness values. The remainder is formed through crossovers. Afterwards, mutation is applied depending on a probability. Crossovers are specifically designed, choosing a value among one of the three segments formed inside the interval of the attribute by putting the two values of the individual as cross points. That is, for every attribute, an individual has two values, and then those values are partitioning the interval in three segments. We select randomly a value inside of a segment also randomly chooses. The next figure shows the procedure:

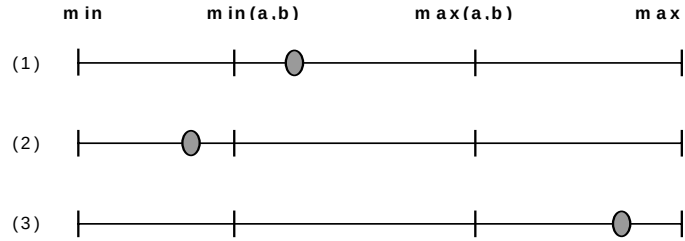

Fig. 6. Crossover operator.

One of the three types of crossover operator could be applied, depending on a probability. The first one is more conservative and second and third ones are more explorative. When the crossover is applied to any angle location, the first one is always used.

Mutation is applied in two different ways: if the location corresponds to a value of the interval, then a quantity is subtracted or added, depending on whether it is the lower or the upper bound, respectively (the distance actually is the lower euclidean distance between any two examples); if the location corresponds to an angle, it is randomly generated another one.

Furthermore, it is advisable to explore the space with the best individual, because we cannot know what angles are the best, and we cannot either know if an angle is going to be better than other is. In this way, we are using hill-climbing technique, what could find out best solutions; despite of inconveniences said in the introduction. The method consists of exploring close rotated regions with the same centre. The angles of the exploration belongs to the interval $[-\pi/6,\pi/6]$, with an increment of $\pi/30$. Then, every best rule explores the search space with other ten regions around the same centre for each attribute. However, in order to reach better fitness value, the interval is also modified the same quantity as mutation used.

To improve the best individual is a difficult task. If the fitness value is better, then the new angle replaces to the old, and one value of one attribute is modified as mentioned above. This method allows rotating an hyperbox using only one dimension. To explore 10 new orientations at the beginning (the first generations) can produce the typical problems of the hill-climbing methods. For that, we recommend to use few explorations at the start and increase it toward the final. Thus, the last generations explore more than the first ones.

We next show an overview of the DQ-Classifier.

```
While exists examples in training file

Step 1. Initialize population

Step 2. Repeat num_generations times
```

```
            Step 2.1. Evaluation
            Step 2.2. Select the best
            Step 2.3. Replication
            Step 2.4. Crossover and Mutation
            Step 2.6. Improve the best
    Step 3. Put the best one in Decision Queue
    Step 4. Eliminate the covered ones by the best one
```

Fig. 7. Overview of DQ-Classifier.

A possible criterion to implement the mutation operator consists of distinguishing between mutation of values and mutation of angles, as two independent operators. Mutation of values could be a higher probability of application than mutation of angles, and also, to incorporate to the evaluation function the distance from the individual (rule) to the closer example of the same class. Thus, we can penalty the rotated rules with wrong angles; that is, the new individual is not going near to the closer example of the same class.

2.3 Fitness Function

The evolutionary algorithm minimizes the fitness function *f* for each individual. It is given by

$$\textit{if } G(i)*RC<=CE(i) \textit{ then } CE(i)=0 \tag{5}$$

$$f(i)=\frac{V}{T}+\frac{G(i)}{1+CE(i)}$$

where *T* is the cardinality of the training file, *V* is a new factor called coverage (the rule coverage is the side of a k-dimensional hypercube which volume is equivalent to the volume of the covered k-dimensional region by the rule); *CE(i)* is the class errors, which are produced when the *i* example belongs to the region defined by the rule, but it does not the same class; *G(i)* is the number of goals of the rule; *RC* is the relaxing coefficient. Every rule can quickly expands for finding more examples due to V in the fitness function.

2.4 Relaxing Coefficient

Databases uses as training files have not areas clearly differentiated, for that, to obtain a rule system totally coherent involves a high number of rules. We show in

previous paper [1] a system capable of producing a rule set exempt from error rate; however sometimes, it is interesting to reduce the number of rules for having a rule set which may be used like a comprehensible linguistic model. When databases present a distribution of examples very hard to classify, then it is advisable to use a relaxing coefficient [10]. Many times, we are more interested in understanding the structure of the databases than in the error rate. In this way, it could be better a system with less cardinality (despite some errors) than too many rules (with 0% of error rate). Then, it may be interesting to introduce the *relaxing coefficient* for understanding the behavior of databases by decreasing the number of rules. RC indicates what percentage of examples inside of a rule can have different class to the rule. RC behaves like the upper bound of the error with respect to the training file, that is, as an allowed error rate.

To deal efficiently with noise and find a good value for RC, the expert should have an estimate of the noise percentage in its data.

3 Application

3.1 Ex Profeso Databases

We have designed some databases of varying complexity to show graphically the experiments. These databases are shown in the fig. 8. Results are in table 1.

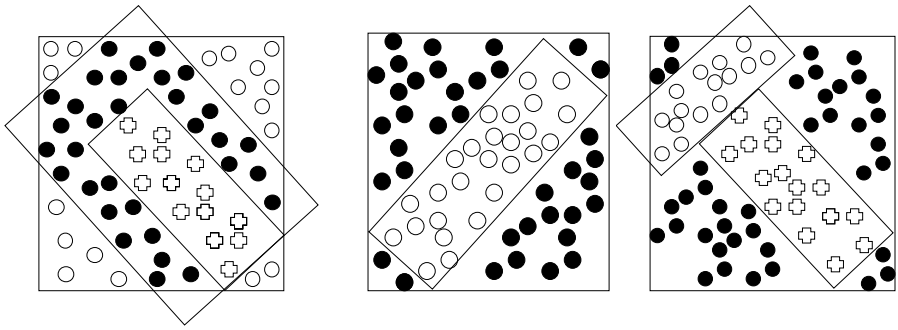

Fig. 8. Ex profeso databases named DB1, DB2 and DB3.

3.2 Databases from UCI Repository

The experiments described in this section are from UCI Repository [7]. We use five cross validation in all our experiments to estimate classification accuracy. This cross validation experiment consists of the following steps: randomly divide the data into

two disjoint partitions (70% an 30%); build a rule set using 70% of data and test the rule set with 30% of data; for each partitions the number of correct classification of the rule set are all counted and divided it by the number of instances of the test file to compute the classification accuracy; the five values are summed and divided by five; report one hundred minus this accuracy multiplies by 100 and the average of the number of rules. We have chosen C4.5 to compare the results, also with five experiments and cross validation which are shown in Table 2.

DATABASE	**C4.5**		**DQ-CLASSIFIER-RH**	
	ERROR RATE	NUMBER OF RULES	ERROR RATE	NUMBER OF RULES
DB1(200,2,2)	12.9	**17**	3.3	3
DB2(200,2,2)	16.25	12.5	**2.7**	**2**
DB3(200,2,2)	12.13	11	**6.6**	**3**

Table 1. The results of artificial databases.

DATABASE	**C4.5**		**DQ-CLASSIFIER-RH**	
	ERROR RATE	NUMBER OF RULES	ERROR RATE	NUMBER OF RULES
IRIS (150, 4, 3)	6.3	**4.4**	**4.83**	3.6
BREAST CANCER (683, 9,	13.8	5.2	**5.38**	**2.2**
PIMA (768, 8, 2)	28.4	77.6	**26.35**	**17.4**
WINE (178, 13, 3)	**7.2**	**5.0**	10.1	6.4

Table 2. Databases (number of examples, dimension, number of classes).

It is very important to note that every execution has been realized with a population of 50 individuals and 50 generations. Very low numbers considering the number of examples and number of dimensions of the databases.

Decision queue is very relevant in relation to the number of rules.

4 Conclusions

A supervised learning tool to classify databases with rotated hyperboxes is presented in this paper. It produces a decision queue where conditions of each rule indicate if an example belongs to a rotated hyperbox. The number of rules is reduced with regard to other systems, like C4.5; and improves the flexibility to construct a classifier varying the relaxing coefficient.

References

[1] Aguilar, J. and Riquelme, J. COGITO: Un sistema de Autoaprendizaje basado en Algoritmos Genéticos. III Jornadas de Informática, pp. 79-88, Cádiz, 1997.
[2] Blum, A. and Rivest, R. L. Training a 3-node neural network is np-complete. In Proceedengs of the First ADM Workshop on the Computational Learning Theory, pp. 9-18, Cambrige, MA, 1988.
[3] Ghozeil, A. and Fogel, D.B. Discovering Patterns in Spatial Data Using Evolutionary Programming. Genetic Program Conference 96.
[4] Goldberg, D. Genetic Algorithms in search, optimization and machine learning. Addison-Wesley Publishing Company, Inc. 1989.
[5] Michalewicz, Z. Genetic Algorithms + Data Structures = Evolution Programs. Second Edition, Springer-Verlag, 1994.
[6] Mitchell, T. Machine Learning. MacGraw-Hill, 1997.
[7] Murphy, P. and Aha, D.W. UCI Repository of Machine Learning Databases. Dept. of Information and Computer Science. University of California, Irvine, 1994.
[8] Murthy, S. K., Kasif, S. and Salzberg, S. A system for induction of oblique decision trees. Journal of Artificial Intelligence Research, 1994. Morgan Kaufmann Publishers.
[9] Quinlan, J. R. C4.5: programs for Machine Learning. Morgan Kaufmann Pub.,1993.
[10] Riquelme J. and Aguilar, J. COGITO 2.0: Una herramienta para obtener un Clasificador Jerárquico en Apredizaje Supervisado. Conferencia de la Asociación Española para la Inteligencia Artificial. CAEPIA'97. pp. 489-498, Málaga, 1997.
[11] Rivest, R.L. Learning Decision Lists. Machine Learning, 87. pp. 229-246.
[12] Simpson, P.K. Fuzzy Min-Max Neural Networks. II. Clustering. IEEE Trans. Fuzzy Systems, Vol. 1:1,32-45.
[13] Venturini, G. SIA: a Supervised Inductive Algorithm with Genetic Search for Learning Attributes based Concepts.