MEDINFO 2001
V. Patel et al. (Eds)
Amsterdam: IOS Press

Supporting Medical Decisions with Vector Decision Trees

Matej Šprogar[a], Peter Kokol[a], Milan Zorman[a], Vili Podgorelec[a], Ryuichi Yamamoto[b], Gou Masuda[b], Norihiro Sakamoto[c]

[a]*Laboratory for system design, Faculty of electrical engineering and computer science, University of Maribor, Slovenia*
[b]*Osaka Medical College, Takatsuki, Japan*
[c]*Kyushu university, Fukuoka, Japan*

Abstract

The article presents the extension of a common decision tree concept to a multidimensional - vector - decision tree constructed with the help of evolutionary techniques. In contrary to the common decision tree the vector decision tree can make more than just one suggestion per input sample. It has the functionality of many separate decision trees acting on a same set of training data and answering different questions. Vector decision tree is therefore simple in its form, is easy to use and analyse and can express some relationships between decisions not visible before. To explore and test the possibilities of this concept we developed a software tool - DecRain - for building vector decision trees using the ideas of evolutionary computing. Generated vector decision trees showed good results in comparison to classical decision trees. The concept of vector decision trees can be safely and effectively used in any decision making process.

Keywords:

Decision Trees; Decision-making; Data Mining; Genetic Algorithms; Machine Learning

Introduction

In the medical environment decision support systems can enhance medical staff's ability to make key decisions and should aid in and strengthen a choice process. In critical decision-making where mistakes are not allowed the experience and personal feeling of a human expert are irreplaceable. Only intelligent programs, which would learn and think in a human-like manner, could compete with a human expert. Normally we use the data-mining techniques to browse accumulated data and find the desired information, but the idea of intelligent programs gives us some clues of how to upgrade current solutions. They should be combined with new, less limited concepts of computing that are able to exploit all properties of a problem space. This would overcome certain deficiencies of the existing solutions that aren't able to understand the meaning of the results neither show progress in overall process. The classical approach to medical decision making is based on hard coded principles, which are invariable and absolute and are as such a source of many limitations in current decision support systems. To get beyond these boundaries it is necessary not only to change the method in decision support system (DSS), but the principles of programming, too. Evolutionary systems are an alternative programming concept providing needed twist in orthodox programming techniques with the help of natural principles of reproduction, selection and mutation. Combination of evolution and certain decision model has potential to become a powerful and effective decision support framework.

Medical decision support systems should primarily help medical experts and should close the gap between mere facts and real understanding. A chosen DSS model should be efficient, simple to use and should give some explanation of the results; this will help in verification of the proposed solution and can potentially express some new knowledge to the expert. Decision trees have already proved themselves in medical decision systems as a conceptually simple model with a possibility of automatic learning [1]. Decision trees represent knowledge in a simple, hierarchical tree-like fashion what makes them a method of choice for many medical or nursing decision problems. Although successful, the classical statistical approach to decision tree construction has shortages [2] that we can elegantly avoid with the use of evolutionary computing [3]. The classical concept of decision trees defines only one type of decision per decision tree. In certain medical problems this is to limiting- and therefore other models must be considered. We tried to extend the functionality of decision trees by allowing multiple solutions in a single decision tree leaf. This way a single tree would produce results in a form of the decision vector. In that manner a single vector decision tree would express additional functionality but would still possess all the simplicity and advantages of a standard decision tree.

In this paper we present a new concept of vector decision trees together with a way to generate vector decision trees using evolutionary techniques. To test the concept of vector decision trees a decision support for a real world medical problem is presented.

Methods

The decision tree method encompasses a number of specific algorithms, including Classification and Regression Trees (CART), Chi-squared Automatic Interaction Detection (CHAID), C4.5 and C5 (J.R. Quinlan). Vector decision trees are however problematic because they include a vector solution and each component in a vector presents a new dimension in a problem space. This type of solution is impossible with most of the today's top algorithms. Thus it is natural to look for another method of decision tree construction. To combine the best properties of two different worlds we decided to build vector decision trees using the ideas of natural evolution. The evolutionary computing has already been successfully used for decision tree induction [4] and in multiple criteria decision-making [5, 6]. Since the concept of vector decision trees is general in its nature it can be applied to all sorts of medical data or and data in general. Therefore we built a special tool - DecRain - for constructing vector decision trees (VDT).

Vector decision tree

Decision trees are simple hierarchical structures that can be generated using automatic learning. Their form expresses learned knowledge and can be used for supporting medical decision processes [7, 8]. Ordinary decision trees (DTs) hold one solution in each of their leaf nodes. Input pattern can thus be classified as belonging to one of the possible classes, but all of the classes are actually answers to one question only. If we would like to classify the same input case in more than one way, we would have to use different decision trees for each type of classification. This complicates the decision making process and disturbs the simplicity of a single decision tree. With the introduction of a vector of classifications into the decision tree leaf this problem is solved in the simplest way possible. Now each leaf classifies a specific data case in more than one manner and therefore a single VDT answers more questions simultaneously. The single vector decision tree has the functionality of many separate common decision trees. It is simpler in its form and is easier to use and analyse.

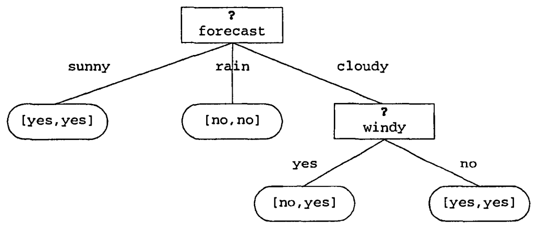

Figure 1 - Vector decision tree

To present the idea we extended a classical example showing a simple decision tree for a trivial problem of deciding whether to go for a walk or not. VDT differs from the classical decision tree only in leaf nodes, where more classifications are made in a form of a vector. If the simple tree answered only yes / no for a planned trip, now a VDT suggests whether it's a good time to wash the car, too.

As shown in figure 1 the VDT leaf nodes return a vector of decisions, where the first component in a vector answers the first question and second the second question respectively. This is a very simple tree with only four leaf nodes. Real decision trees normally classify the samples of a particular class using several different leaf nodes. In this example the [yes, yes] classification is proposed in two of the leaf nodes. We should note here that the VDT with a vector of size 1 represents a classical decision tree. In general some dependencies exist between different questions asked and the vector decision tree is supposed to capture and use these dependencies.

The important thing in a vector solution is the correlation of questions (decisions). The decree of correlation is visible when comparing a VDT with the corresponding common decision trees. Vector decision tree can only suggest combinations (vectors) actually present in the data set. It is also obvious that a vector decision tree giving correlated decisions will be smaller than the VDT with fully independent decisions. With the VDT concept in mind we now need a methodology for building, the vector decision trees from the scratch. Existing statistical concepts for decision tree construction are somewhat limited - they prefer smaller classifications, are sensible to missing attribute values and sometimes just don't produce results of adequate quality. One might argue to choose the different decision model instead of trying out different programming techniques. Because we wanted to keep the simplicity and expressiveness of decision trees we turned to evolutionary systems and genetic algorithms.

Evolutionary computation and genetic algorithms

Evolutionary computation is based on the principles of natural evolution. It differs from the classical programming in one major point - deterministic algorithms create one good solution object using complex theories, e.g. information theory or statistics, in evolutionary computation we have a population of individuals (solutions) that evolve and adapt to their environment. The strength of the evolution comes out of the simple principles of selection, reproduction with crossover and mutation. These fixed operators navigate through the given search space. To produce a good solution we need an appropriate environment that will give advantage to the individuals with preferred properties.

Genetic algorithms (GA) are randomised parallel search algorithms that model natural selection. They are a robust search method requiring little information to search effectively in large, poorly understood spaces. Genetic algorithms operate on a set of individuals within the search space. Each individual is represented with a genetic code - genotype - that effectively defines the individual's phenotype - its behavior and properties. Genotype structure is determined at the design time of the GA. The simulation system in GA then starts a loop in which the individuals will evolve. This loop uses three genetic operators: selection, crossover (recombination) and mutation. To evolve towards the optimal solution we need to know the quality of an individual and that is the goal of the fitness function. The GA designer should select a fitness function that gives a better score to individuals with the desired phenotype behaviour. Selection operator selects a set of individuals that will survive into the next generation using their fitness scores. They will have a chance to breed and produce the offspring using the crossover operator. Crossover should produce healthy new individuals with possibly good properties from both parents. The mutation operator is used to occasionally alter the genotype of an individual in a random manner. After replacing the old with the new individuals the GA loop is repeated until some termination criterion is met. The final solution of a GA run is the individual with the best fitness score. It should be noted that GAs do not always conform to the form described above.

Evolution of vector decision trees

First we must specify the vector decision tree's genotype (the VDT is an individual in the population of trees). The most common representation in GA is a bit string encoding because bit strings are easy to handle. We can however implement the genetic operators for tree-like structures easily and can have therefore a tree-like structure for the genotype, too. This means the VDT's genotype can be the same as its phenotype. This allows faster fitness calculation since we can calculate the fitness directly using the genotype. The tree-like genotype is a dynamic structure and has no constant size nor shape.

When the genotype coding of an individual is determined we need to create a fitness function that will calculate a fitness score for the individual - see Equation (1). The main factor in the fitness is the accuracy. Because there are many dimensions in the vector solution we can calculate average accuracy over all dimensions (*hits*) and also accuracy for the fully correctly classified vectors (*jackpots*). The size of the learning set is denoted by u (number of cases), v is the dimension of the solution vector (number of questions asked), v_{ij}' is count of correct classifications of the i-th question's j-th class and v_{ij} is the correct value for v_{ij}' that can be easily obtained from the learning data. The fitness score is based on the observed properties of a competing tree [9]. Since we are developing a data mining method the VDT must be judged according to its behaviour on a given data set.

$$fitness = f(w_1 \cdot \frac{hits}{v \cdot u},\ w_2 \cdot \frac{jackpots}{u},\ w_3 \cdot \underset{i,j}{E}(\frac{v_{ij}'}{v_{ij}}),\ w_4 \cdot Min(\frac{v_{ij}'}{v_{ij}}),\ w_5 \cdot penalty(\frac{introns}{size})) \tag{1}$$

Once we have a fitness function we can randomly create any number of individuals that will form the initial GA population. The trees (their genotype) must be logically correct and the random choice of internal node attributes should provide the diversity in genetic code. Because the trees have random topologies they usually have a wide range of fitness scores.

With the initial population ready we can enter the main GA loop where the genetic operators will alter the genotype of individuals. First we need to select the individuals that will produce offspring. This is a task for the selection operator that uses the calculated fitness scores. After some of the individuals are chosen as parents for the new generation we can delete all other individuals. Using the crossover operator the parents then reproduce and create a new population of individuals. Each new individual must be evaluated by the fitness function before the loop can close. The mutation operator is used sparingly because the mutations are usually destructive but they are also the source of fresh genetic code.

Selection, crossover and mutation

The three genetic operators are the core of GA [9]. Each operator has a specific task it must finish in order for the GA to evolve the individuals. Because there are many ways to complete these tasks many different solutions exist.

The selection operator, for example, can sort the individuals based on their fitness and then select the first few ones. Or it can select each individual with a different probability or may even create a kind of a tournament. The crossover and mutation operators are bound to the genetic code and must properly change the genetic material. This means that they must produce a healthy individual in respect to the structure but the quality of the new code is not known. Because we are dealing with tree-like genetic code we must create the appropriate crossover and mutation operator while for selection we can choose any of the well known selection schemes [9, 10].

The crossover operator creates two descendants from a pair of parent individuals. On tree-like structures it simply exchanges two sub-trees. While the exchange itself is fairly simple we first need to select the two sub-trees. This is more precisely explained in [9]

The mutation operator introduces random changes in the genetic code of the object. Mutations are mostly destructive but are necessary to bring new freshness in the older genetic material. Mutations change the fragile balance established through the evolution with a change in the contents, structure or the size of an individual. For the tree-like genotype it is easy to implement point mutation (new test in node), permutation (reorder the sub-trees) and expand & collapse sub-tree mutation [11].

Case study

The decRain tool has been used in various real world medical applications like mitral valve prolapse diagnosis, breastfeeding analyses, cancer prediction and others [9].

One of the recent and more interesting applications is the treatment determination for diabetes patients. The attributes consisted of various parameters like sex, age, height, weight, BMI, retinopathy, etc, altogether 21 continuous attributes and 30 discrete attributes. The decision vector consisted of two treatments, namely the *type of treatment (DietOnly, POdrugs, Insulin)* and the *photocoagulation therapy (Yes, No)*. The training set contained 754 training objects, while the test set contained 377 test objects. The majority of objects in training and test set had missing values.

We divided the results of decision-making on the test set in two tables. In table 1 are the results for the *type of treatment*, where the accuracy reached 60.4%. The results for *photocoagulation therapy* (table 2) were much better, and reached 90.7%. The overall accuracy for both treatments was 75.6%.

Table 1. Testing results for Treatment

Treatment	Prediction				
	DietOnly	Podrugs	Insulin	?	*Accuracy*
DietOnly	66	43	6	6	*54.5%*
Podrugs	44	117	15	3	*65.3%*
Insulin	5	22	45	5	*58.4%*
total accuracy: 60.4%					

Table 2. Testing results for Photocoagulation Therapy

Photocoagulation Therapy	Prediction			
	Yes	No	?	*accuracy*
Yes	39	12	2	*73.5%*
No	9	303	12	*93.5%*
total accuracy: 90.7%				

Conclusion

When selecting appropriate medical decision support system the most important thing to consider is the selection of an appropriate decision model. Decision trees are simple and robust and have proved themselves in many medical applications. Presented extension of the decision tree concept with a vector solution showed some interesting features. The vector decision tree has the functionality of many separate decision trees and can make more than just prediction per input sample. Combined with the power of genetic evolution the vector decision trees can produce good results in any problem space. If we look at a classical

```
{754} ? DiseaseDuration
  :..[<= 9.253]{296} ? DiseaseDuration
  :   :..[<= 7.370]{220} ? VisualDisturbance
  :   :   :..[Yes]{13: 3+7+3 5+8} POdrugs No {3/5}
  :   :     ..[No]{207: 126+67+14 1+206} DietOnly No {51/86}
  :     ..[ 7.370]{76} ? Retinopathy
  :   :..[NDR]{56} ? BMI-2
  :   :   :..[<= 22.200]{20: 11+7+2 0+20} DietOnly No {8/14}
  :   :     ..[ 22.200]{36} ? FamilyHistory
  :   :   :..[Yes]{16: 4+11+1 0+16} POdrugs No {5/10}
  :   :     ..[No]{20: 12+8+0 0+20} DietOnly No {6/12}
  :   :..[PDR]{4: 1+2+1 4+0} POdrugs Yes {3/5}
  :   :..[prePDR]{4: 2+2+0 4+0} DietOnly Yes {1/1}
  :     ..[SDR]{10: 2+6+2 1+9} POdrugs No {6/9}
    ..[ 9.253]{458} ? SMBG
      :..[Yes]{109} ? Retinopathy
      :   :..[NDR]{37: 2+5+30 1+36} Insulin No {11/18}
      :   :..[PDR]{24: 0+6+18 22+2} Insulin Yes {8/13}
      :   :..[prePDR]{9: 0+2+7 6+3} Insulin Yes {3/6}
      :     ..[SDR]{39: 0+7+32 7+32} Insulin No {17/23}
        ..[No]{344} ? Retinopathy
          :..[NDR]{194} ? BMI-2
          :   :..[<= 16.788]{6: 4+1+1 0+6} DietOnly No {0/2}
          :     ..[ 16.788]{188: 56+120+12 0+188} POdrugs No {67/104}
          :..[PDR]{46} ? BMI-2
          :   :..[<= 21.239]{16: 0+5+11 11+5} Insulin Yes {2/6}
          :     ..[ 21.239]{30: 4+24+2 23+7} POdrugs Yes {6/14}
          :..[prePDR]{22} ? VisualDisturbance
          :   :..[Yes]{7: 0+5+2 6+1} POdrugs Yes {2/3}
          :     ..[No]{15: 0+12+3 5+10} POdrugs No {3/4}
            ..[SDR]{77: 11+54+12 6+71} POdrugs No {15/28}
```

Figure 2. Sample vector decision tree for diabetes treatment determination

decision tree as on a way to transform an input pattern into one classification, then decision trees are also very simple computer programs consisting of only if - then directives, what puts us closer to the field of genetic programming. When compared to generic genetic programs many further extensions to genetic representations of decision trees are possible, yet they might change the simplicity and transparency found in decision trees. Proposed extension with vector solution preserves all of the decision tree's properties and has overall positive effect on its capabilities. In the paper we presented the vector decision tree concept, proposed and described a way to generate such trees and implemented a vector decision tree building tool named DecRain. Using this tool we were able to carry out a comparison study on many medical problems. In most cases the vector decision trees were superior in presenting more knowledge, discovering new knowledge, decision power and also accuracy. Indeed, since the vector decision tree is by its nature still a decision tree we believe this new concept can be safely and effectively used in any medical decision-making problem with more than just one classification in question. In the DecRain tool we successfully combined the concept of vector decision trees with genetic algorithms. Produced vector decision trees was usually simpler than the combination of its single decision relatives because it was able to grasp certain knowledge out of relationships between questions asked. These relationships were hidden to common decision trees until now. Since the DecRain tool implements only the basic functions needed for evolutionary development of vector decision trees we believe the good results in the field of multiple criteria are due to the proposed vector concept. In contrary to deterministic algorithms the evolutionary approach doesn't always produce the same results even when given the same settings and parameters, but it has great potential and is more robust. And as always in evolutionary computing we can never be sure if the found solution is the best possible - it's again up to the medical expert to decide when to stop searching.

References

[1] Kokol P et al. Decision trees and automatic learning and their use in cardiology, *Journal of Medical Systems* 19(4) (1994).

[2] Hleb-Babič Š, Šprogar M, Zorman M, Kokol P and Turk DM. Evaluating breastfeeding advantages using decision trees, in: Proceedings, Twelfth IEEE Symposium on *Computer Based Medical Systems* CBMS, Stamford, Connecticut, 1999,pp.144-149.

[3] Podgorelec V, Kokol P. Self-adaptation of evolutionary constructed decision trees by information spreading, in: Proceedings of the *International Conference on Artificial Neural Nets and Genetic Algorithms*, Springer Verlag, 1999, pp.294-301.

[4] Podgorelec V, Kokol P. Self-adapting evolutionary decision support model, in: Proceedings of the 1999 *IEEE International Symposium on Industrial Electronics ISIE'99*, IEEE Press, 1999, pp. 1484-1489.

[5] Kursawe F. Evolution strategies for vector optimization, in: Preliminary Proceedings of the 10th *International Conference on Multiple Criteria Decision Making*, G.H. Tzeng and P.L. Yu, eds., National Chiao Tung University, Taipei, 1992, pp.187-193.

[6] Kursawe F. *A Variant of Evolution Strategies for Vector Optimization*, University of Dortmund, http://ls-11-www.informatik.uni-dortmund.de/~kursawe/.

[7] Quinlan JR. Induction of decision trees, *Machine Learning 1* (1986), 81-106.

[8] Quinlan JR. Decision trees and instance based classifiers, *Artificial Intelligence and Robotics* (I 997), 521-535.

[9] Šprogar M, Kokol P, Hleb-Babič Š, Podgorelec V, Zorman M. Vector decision trees. *Intelligent Data Analysis* 4 (2000), pp.305.321.

[10] Blickle T, Thiele L. A comparison of selection schemes used in evolutionary algorithms, in: *Intl Journal on Evolutionary Computation* 4(4) (1996).

[11] Banzhaf W, Nordin P, Keller RE, Francone ED. *Genetic Programming* - An *Introduction,* Morgan Kaufmann Publishers Inc., 1998.

Address for correspondence

Matej Šprogar, Laboratory for system design, Faculty of electrical engineering and computer science, University of Maribor, Smetanova 17, SI-2000 Maribor, Slovenia

E-mail: *matej.sprogar@uni-mb.si*