Learning to assess from pair-wise comparisons[1]

J. Díez[(*)], J.J. del Coz[(**)], O. Luaces[(**)], F. Goyache[(*)], J. Alonso[(**)], A. M. Peña[(***)], and A. Bahamonde[(**)]

[(*)] SERIDA-CENSYRA-Somió, C/ Camino de los Claveles 604, E-33203 Gijón (Asturias), Spain.
[(**)] Centro de Inteligencia Artificial. Universidad de Oviedo at Gijón, Campus de Viesques, E-33271 Gijón (Asturias), Spain.
[(***)]Facultad de Ingeniería. Universidad Distrital Francisco José de Caldas, Bogotá, Colombia.

jdiez@serida.org, {juanjo, oluaces}@aic.uniovi.es, fgoyache@serida.org, {jalonso, antonio}@aic.uniovi.es

Abstract. In this paper we present an algorithm for learning a function able to assess objects. We assume that our teachers can provide a collection of pair-wise comparisons but encounter certain difficulties in assigning a number to the qualities of the objects considered. This is a typical situation when dealing with food products, where it is very interesting to have repeatable, reliable mechanisms that are as objective as possible to evaluate quality in order to provide markets with products of a uniform quality. The same problem arises when we are trying to learn user preferences in an information retrieval system or in configuring a complex device. The algorithm is implemented using a growing variant of Kohonen's Self-Organizing Maps (growing neural gas), and is tested with a variety of data sets to demonstrate the capabilities of our approach.

1. Introduction

Generally speaking, quality assessment is a complex matter: what we usually need to evaluate are the desirable traits of an object by means of a single number. Frequently though, this number does not strictly reflect an absolute value, but rather the relative quality of the object with respect to others. This is especially true for objects of a biological origin; their quality is dependent on a not always well defined group of multisensorial properties resulting from their chemical composition, the natural structure of the food elements, their interaction and the way in which they are perceived by human senses [13]. This situation becomes even more complex when we consider quality grading of food products from the viewpoint of experts or consumers. Since no detailed grading specifications exist, experts may adopt a quality profile that considerably exceeds that expected by the consumer [2]. The requirements of consumers are usually based on single attributes that characterize primary senses.

[1] The research reported in this paper has been supported in part under MCyT and Feder grant TIC2001-3579

Consequently, the literature reflects disagreement between quality assessments obtained through consumer or expert panels [2, 9].

However, the food industry needs to supply markets with uniform quality products to satisfy consumer demands for normalized quality. Furthermore, if possible, food producers would like to know what the objective (chemical and physical) basis of the assessed quality is from the customer viewpoint so as to improve the acceptability of their products.

The straightforward way to build computable procedures to assess objects is to collect a set of representative assessment events and then to apply a machine learning algorithm that employs regression like CUBIST [6], M5' [16, 20], SAFE (System to Acquire Functions from Examples) [15] or BETS (Best Examples in Training Sets) [7]. However, our experience with biological objects [9, 10] tells us that the complexity of the assessment task means that the repeatability of human evaluations tends to be low. Hence, the reliability of the training material is poor, despite experts having been trained exhaustively and having accumulated a large, valuable body of knowledge used for assessing [12].

Experts or consumers are perfectly able to prefer one object to another, but usually fail when they are asked to label products with a number. There is a kind of *batch effect* that often biases the assessment; human assessors try to number the differences in a relative sense, comparing products with the other partners in the batch. Thus, a product surrounded by worse things will probably obtain a higher assessment than if it were presented with better products. However, although we may find unacceptable individual variability in the absolute number obtained to assess the quality of a given product, the relative position obtained in a batch is quite constant.

In this paper we present a new approach to learning functions capable of assessing objects starting from reliable training material. Our training sets are formed by pairs of object descriptions, given by continuous attributes, where the first one has been considered worse than the second. The goal is then a function able to quantify the quality of objects as coherently as possible with the pair-wise ordering supplied as the training set.

The core idea is to consider each training instance as an indication of a direction where we can find an increase in quality. Thus, the vectorial difference of compared products is interpreted as a kind of coding of the local behavior of the assessment function. In this way, the learning algorithm is a clustering that uses a growing version [8] of Kohonen's Self-Organizing Maps (SOM) [11], where each cell encapsulates a regression rule.

After presenting the geometrical motivations of the algorithm followed by the implementation details, we close the paper with a section devoted to presenting the experimental results obtained with our assessment learner.

2. Geometrical motivation of the algorithm

Let **u** and **v** be vectors describing the features of two objects that our experts compare, resulting in **u** being *worse* than **v**; in symbols, $\mathbf{u} < \mathbf{v}$. Then, we seek a function f such that $f(\mathbf{u}) < f(\mathbf{v})$. If we assume that f behaves linearly, at least in the surroundings of our vectors, we have to find a vector **w** such that

$$f_w(\mathbf{u}) = \mathbf{u}\cdot\ \mathbf{w} < \mathbf{v}\cdot\ \mathbf{w} = f_w(\mathbf{v}) \qquad (1)$$

where, for vectors **z** and **t,** we represent their inner product by $\mathbf{z}\cdot\mathbf{t}$.

From a geometrical point of view, function f_w represents the distance to the hyperplane $\mathbf{u}\cdot\mathbf{w} = 0$; i.e. the hyperplane of vectors perpendicular to **w**. If we search for **w** considering only normalized vectors (i.e. $\|\mathbf{w}\| = 1$), the largest difference between $f_w(\mathbf{u})$ and $f_w(\mathbf{v})$ values is reached when **w** is the normalized vector in the address of ($\mathbf{v}$-$\mathbf{u}$). In fact,

$$f_w(\mathbf{v}–\mathbf{u}) = (\mathbf{v}–\mathbf{u})\cdot\mathbf{w} \le \|\mathbf{v}\text{-}\mathbf{u}\|\cdot\ \|\mathbf{w}\| = \|\mathbf{v}\text{-}\mathbf{u}\| = (\mathbf{v}–\mathbf{u})\cdot(\mathbf{v}–\mathbf{u})/\|\mathbf{v}\text{-}\mathbf{u}\| = f_{(\mathbf{v}\text{-}\mathbf{u})/\|\mathbf{v}\text{-}\mathbf{u}\|}(\mathbf{v}–\mathbf{u}) \qquad (2)$$

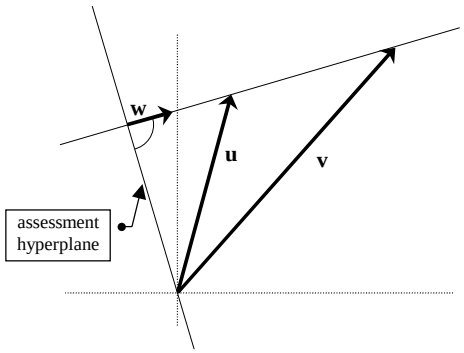

Fig. 1. Given two objects represented by vectors **u** and **v**, if **u** is worse than **v**, the normal vector in the direction of the difference, $\mathbf{w} = (\mathbf{v}\text{-}\mathbf{u})/\|\mathbf{v}\text{-}\mathbf{u}\|$, defines a hyperplane, the distance from which is a suitable local assessment function.

In the general case we start from a family of *comparisons*

$$\{\mathbf{u}_i < \mathbf{v}_i : i = 1, \ldots, n\} \qquad (3)$$

and wish to induce a function f, with local linear behavior, and which hopefully is capable of distinguishing as often as possible that $\mathbf{u}_i$ is worse than $\mathbf{v}_i$, because $f(\mathbf{u}_i) < f(\mathbf{v}_i)$. The algorithm proposed in this paper uses the geometrical intuition introduced in this section as the basic building block of such a function f. Hence, the main task of the algorithm will be to combine the local guidelines suggested by each comparison supplied in the training set.

3. The algorithm: clustering partial functions

In line with the discussions presented in the previous section, the comparison examples of (3) give rise to a set of 2n *pairs* of vectors as follows:

$$\{((\mathbf{v}_i - \mathbf{u}_i)/\|\mathbf{v}_i - \mathbf{u}_i\|, \mathbf{u}_i) : i = 1, \ldots, n\} \cup \{((\mathbf{v}_i - \mathbf{u}_i)/\|\mathbf{v}_i - \mathbf{u}_i\|, \mathbf{v}_i,) : i = 1, \ldots, n\}. \qquad (4)$$

If (**w, u**) is such a pair, we understand it to be a suggestion of a regression rule indicating that the assessment of a vector **z** is

$$f_w(\mathbf{z}) \text{ if } \mathbf{z} \text{ is in the vicinity of } \mathbf{u}. \quad (5)$$

Given that $f_w(\mathbf{z}) = \mathbf{z} \cdot \mathbf{w}$, we will usually identify **w** with the linear ***function*** f_w. Likewise, we will refer to **u** as the ***conditions*** of the rule. For short, we write $\mathbf{w} \leftarrow \mathbf{u}$

In general, we are pursuing an assessment function f defined by parts of the whole attribute space. In other words, our assessment function will be given by a list of regression rules

$$(\mathbf{w}_1 \leftarrow \mathbf{u}_1); (\mathbf{w}_2 \leftarrow \mathbf{u}_2); \ldots; (\mathbf{w}_m \leftarrow \mathbf{u}_m) \quad (6)$$

that must be evaluated by means of a minimal distance criterion. In symbols, the function f that is finally induced will work as follows for an arbitrary vector **z**.

$$f(\mathbf{z}) = \mathbf{w}_k \cdot \mathbf{z} \qquad \text{if } \|\mathbf{z} - \mathbf{u}_k\| \leq \|\mathbf{z} - \mathbf{u}_j\|, \forall\, j = 1, \ldots, m \quad (7)$$

A first attempt to define the list of regression rules (**6**) is to consider the whole set of pairs (**w**, **u**) defined in (**4**), but these rule set must be improved: it is too big and may contain a lot of noise. Therefore, the idea of our learning algorithm (see Algorithm **1**) is to cluster similar conditions **u**, and then to attach a function **w** according to the functions of the pairs of the same cluster (see Figure **2**). To this end, we use a growing version of Kohonen's Self-Organizing Maps (SOM) [11]: growing neural gas (*GNG*) of Fritzke [8]. This approach has the advantage that we do not need to define a priori configuration parameters like SOM layout dimensions or the radius used throughout the adaptation.

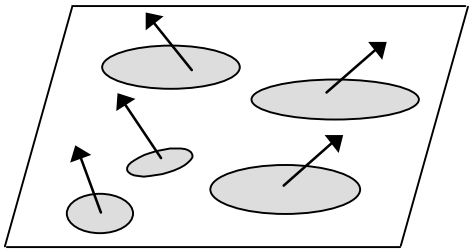

Fig. 2. The clusters of partial functions represents, in each node, an environment in the attribute space of the objects to be assessed, drawn in gray in the picture, and a vector pointing in the direction to measure the assessments. In other words, the map represents a set of regression rules to be applied by means of a nearest-distance criterion.

The GNG graph starts with two nodes $\mathbf{u}_1$, $\mathbf{u}_2$ representing two points in the domain of the assessment function, in each iteration step a new node is added trying to fit better this space. The number of steps (N) followed by *GNG* conditions the granularity of the regression rules. By default, N is the number of comparisons divided by 10.

Once we have a number of clusters represented by $\mathbf{u}_1$, $\mathbf{u}_2$,…,$\mathbf{u}_m$, we consider the set of comparisons ($\mathbf{t}_2 < \mathbf{t}_1$) where each $\mathbf{t}_k$ is closer to the same $\mathbf{u}_i$ than to any other $\mathbf{u}_j$.

These comparisons will be used to compute a local linear approximation of the assessment function in the surroundings of $\mathbf{u}_i$.

The procedure followed to find a linear function with coefficients $\mathbf{w} = (a_1, \ldots, a_d)$ is taken from OC1 [14] only slightly modified for this purpose. In fact, what we are looking for is a vector $\mathbf{w}$ such that $\mathbf{w} \cdot (\mathbf{t}_1 - \mathbf{t}_2) > 0$ as many times as possible. We can start with $\mathbf{w}$ being the average of the normalized differences

$$\mathbf{w} = (a_1, \ldots, a_d) = \text{Average}\{(\mathbf{t}_1 - \mathbf{t}_2)/\| \mathbf{t}_1 - \mathbf{t}_2 \| : \mathbf{t}_1 > \mathbf{t}_2 \ \& \ \mathbf{t}_1, \mathbf{t}_2 \in \text{cluster}(\mathbf{u}_i)\}. \quad (\mathbf{8})$$

Now we try to improve the coefficients a_m, one at a time. The key observation is that for each normalized difference $(\mathbf{t}_1 - \mathbf{t}_2)/\|\mathbf{t}_1 - \mathbf{t}_2\| = (x_1, \ldots, x_d)$ we have that

$$\mathbf{w} \cdot (\mathbf{t}_1 - \mathbf{t}_2) = \Sigma(a_i * x_i : i = 1..d) > 0, \quad (\mathbf{9})$$

when $x_m > 0$, is equivalent to

$$a_m > -(a_1 x_1 + a_2 x_2 + \ldots + a_{m-1} x_{m-1} + a_{m+1} x_{m+1} + \ldots + a_d x_d) / x_m = U \quad (\mathbf{10})$$

or the opposite when $x_m < 0$. When $x_m = 0$, the value of the coefficient a_m does not matter. So, for fixed values of all other coefficients, each equation (10) represents a constraint on the values of a_m. Therefore, we sort all U values and consider as possible setting for a_m the midpoints between each pair of consecutive U's. We select the a_m that satisfies the greater number of constraints. Following the procedure of OC1, we iterate this step in order to adjust all the coefficients until no further optimisation can be achieved.

If the number of clusters is high, for instance whenever we use the default value for N, the number of training examples divided by 10, then the previous approach inspired in OC1 can be skipped (the results are quite similar). We can simply update the function $\mathbf{w}$ attached to a cluster $\mathbf{u}$ as the average of the functions $\mathbf{w}$' of pairs ($\mathbf{w}$',$\mathbf{u}$') whose winner node is $\mathbf{u}$.

In any case, the regression rules so found need a final improvement process. The idea is that $\mathbf{w} \leftarrow \mathbf{u}$ may correctly resolve assessments of objects near $\mathbf{u}$. That is, when $\mathbf{t}_1 > \mathbf{t}_2$, and both $\mathbf{t}_1$ and $\mathbf{t}_2$ are near $\mathbf{u}$, $\mathbf{w}$ was devised for obtaining $\mathbf{w} \cdot (\mathbf{t}_1 - \mathbf{t}_2) > 0$. But $\mathbf{w}$ may fail when one of the objects is going to be assessed by another rule $\mathbf{w'} \leftarrow \mathbf{u'}$. To solve these situations we are going to look for adequate *slope modifiers* a and independent terms b such that the function of the regression rule will now be

$$a(\mathbf{w} \cdot \square) + b \leftarrow \mathbf{u}. \quad (\mathbf{11})$$

The procedure followed to find a and b for each regression rule is almost the same that we have just described for adjusting the coefficients a_m of each w. The only difference is that now we consider comparisons where only one of the objects is near the condition of the rule to be improved.

```
Function LEARN TO ASSESS COMPARISONS FROM EXAMPLES (LACE)
        (comparisons set {u_i<v_i: i=1,…,n}, number of steps N) {
   E = {( (v_i − u_i)/‖v_i − u_i‖ , u_i): i=1,…,n} ∪ {( (v_i − u_i)/‖v_i − u_i‖ , v_i,): i=1,…,n};
   // To have comparable values in [0,1]
   Normalize each component of conditions u_i and v_i in E pairs;
   // Now, we cluster the conditions of E examples
   GNG(conditions(E), steps = N); //by default N = |E|/(2*10)
   Let (w_1,u_1), (w_2,u_2),…,(w_m,u_m) be the nodes of the graph
        where w_i are the average values of the training
        instances having node i as the nearest one
   //the next loop can be safety skipped when N is high
   for each node (w_i, u_i) in graph do {
     //notice that the function w_i is an arbitrary value
     w_i = Oc1{t_1-t_2: (t_2 < t_1) && ‖u_i,t_1‖≤‖u_j,t_1‖ && ‖u_i,t_2‖≤‖u_j,t_2‖ j ≠ i}
   }
   improve relative slopes and independent terms of regression rules;
   return regression rules;
}
```

Algorithm 1. The algorithm that learns to assess from pair-wise comparison examples (LACE).

4. Experimental results

In order to test the validity of our approach we conducted a number of experiments. The idea is to deal with assessment problems where we know *a priori* the kind of results that we would like to obtain.

To illustrate the way that our algorithm works, we start with a simple problem. Let us consider objects describable by only one continuous attribute **x** with values in [0, 1], and having as *true* assessment function the parabola ta(x)= -x(x-1), see Figure (**1**, a). To build a training set of comparisons E, we generated 3000 pairs (x1,x2) with values in [0,1], and we added to E the pair (x1,x2) if ta(x1) > ta(x2), and we added (x2,x1) otherwise. Our algorithm learned from E the function f drawn in Figure (**2**, b). Notice that while the actual values of f(x) and ta(x) are quite different, the relative values are almost the same. In fact, building a test set of comparisons using the same procedure followed for E, we only found that the 0.03% of the pairs were erroneously ordered by f.

A second package of experiments (see Table **1**) was carried out with objects describable by two continuous attributes: **x** and **y**. Once an assessment function had been fixed, the objects were randomly generated as 2-dimensional points in the stated rectangles; once we have generated two such objects, they are written in the comparison set, the worse one (according to the corresponding goal assessment function) first. We additionally generated another test set of comparisons, changing the random seed. Both sets had 3000 pairs. The errors reported are the percentage of test pairs that were misplaced by the assessment function learned by our algorithm. These data sets should be easy problems for our learner, and in fact were so, as can be seen in the scores reported in Table (**2**): However, we can appreciate significantly better scores when the regions with different assessment behavior are separated.

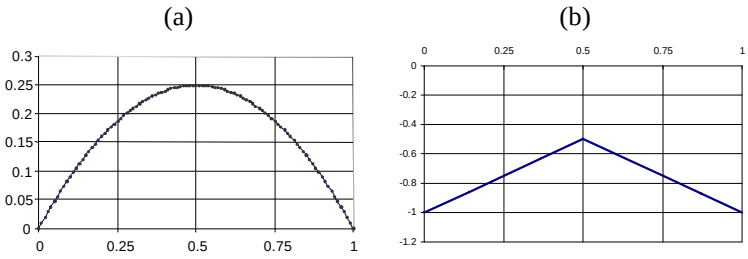

Fig. 3. The objects to be assessed are described by $x \in [0, 1]$. (a) The true assessment is ta(x) = -x(x-1). (b) The function learned by our algorithm f. Only 1 of the 3000 test pairs is erroneously ordered by f.

Table 3. Experiments carried out with goal functions defined by two linear subfunctions with separate domains. The original objects to be assessed were vectors in the rectangles $[0, 999] \times [0,999]$ in the first two rows, and for the other two $[200,299] \times [0,999]$, and $[700,799] \times [0,999]$. Both training and test sets have 3000 elements. We used only 3 steps to adapt the underlying GNG graph.

Goal functions	Error
$f(x,y) = \begin{cases} x + 10y & x \le 500 \\ 10x + y & x > 500 \end{cases}$	4.83%
$f(x,y) = \begin{cases} x + y & x \le 500 \\ x - y & x > 500 \end{cases}$	5.46%
$f(x,y) = \begin{cases} x + 10y & x \in [200,299] \\ 10x + y & x \in [700,799] \end{cases}$	0.23%
$f(x,y) = \begin{cases} x + y & x \in [200,299] \\ x - y & x \in [700,799] \end{cases}$	0.20%

Finally, we used some publicly available regression datasets in an attempt to deal with almost *real-world* data. We built training and test sets providing comparisons between the class values of pairs of examples instead of training on their class labels as in the conventional setting; for each example we randomly selected other 10 examples, 8 of them were placed in the training set and the other 2 went to the test set. In order to compare the achievements of LACE, we used two well-known regression learners: M5' [16, 20], and Cubist [6]. We trained M5' and Cubist with the whole dataset, that is considering not only the description of the objects, but the numeric class too. To test what they learned, we compared the values provided for each component of the comparison. The scores so found are reported in Table (**4**).

Let us remark that the comparative reported in Table (**4**) is not fair for our LACE. The reason is that regression learners have access to the true numerical classes for *all test examples*, while LACE can only see pairs where there are differences, but without knowing the amount of those differences. As was pointed out in the introduction, in *real-world* cases we will not have the numeric classes and so we will not able to use M5' or Cubist.

Table 4. Error scores of our learner in publicly available regression datasets in addition to the parabola dataset described above. The *CPU*, *Body fat* were dowloaded from Cubist URL [6], while *Boston housing*, and *Liver disorders* can be found at UCI Repositoty [1]. The number of steps followed by GNG was the default value, i.e., the number of training comparisons divided by 10. Notice that LACE reached only 0.03% errors when N was 3 in the parabola dataset.

dataset	Cubist	M5'	LACE
CPU	13.16%	11.00%	11.48%
Boston Housing	8.99%	9.19%	7.01%
Body fat	17.26%	15.48%	11.10%
Liver disorders	31.59%	31.30%	14.63%
Parabola	0.86%	9.13%	3.93%
Average	14.37%	15.22%	9.63%

5. Related work

Tesauro tackled a similar problem in [17] for finding a function able to select the most preferable alternative in his famous backgammon player. His proposal was to enforce a symmetric neural network architecture consisting of two separate subnetworks, one for each object in the comparison. In addition, he enforced that both subnetworks have the same weights (only multiplied by -1 in the output layer). However, this restriction in the training mechanism only worked properly with perceptron networks, at least in his application field. Other perceptron approaches are described in [18,19].

In information retrieval, user preferences were modelled by means of *preference predicates* learned from a set of comparisons [3, 4, 5]. This is a quite different approach since our aim is to obtain a function able to assess grader preferences with a number; for our purposes it is not enough to know which object is preferable. Additionally, once you have a preference predicate, to order a set of objects is a NP-hard problem [5] since the transitivity of the learned predicate is not guaranteed at all.

6. Conclusions

In this paper, we have presented a new approach to obtaining sound assessment functions of objects. Our approach allows us to make use of a kind of knowledge capable of satisfactorily ranking a set of objects from the best to the worst, but that fails in assessing the ‘goodness’ of a single object with an absolute number. Assessments carried out in an absolute way are strongly affected by a batch effect in the sense that they tend to number the quality of an object with respect to the other objects in a batch, but not in an absolute sense, as we hope for when we assign a number to quality. This situation is characteristic of biological objects, and especially in the food industry, in which the rules for deciding the degree of quality of a product

are not usually well defined, but the ranking of products is quite constant and well accepted on the part of consumers and market operators.

From a computational point of view, we have to obtain a float function from training sets without categorical or continuous classes. The problem has been tackled with a growing modification of Kohonen's SOM based on a geometrical intuition of the transformations that should be applied to the training data. The algorithm thus built was tested with both artificial and real-world data in order to show the abilities of the method proposed. The results reflect a very high degree of accuracy.

The limitations of our approach, which should be overcome in a future work, have to do with the granularity of the underlying GNG graph that clusters training data. Additionally, we hope that an improvement in the placement of conditions (**u**) in regression rules (**w** ← **u**) would provide a better performance of solutions with a lower number of steps, see Table 3.

References

1. Blake, C., Merz, C. J.: *UCI Repository of machine learning databases* [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science (1998)
2. Bollen A.F., Kusabs N.J., Holmes G. and Hall M.A.: Comparison of consumer and producer perceptions of mushroom quality. Proc. Integrated View of Fruit and Vegetable Quality International Multidisciplinary Conference, W.J. Florkowski, S.E. Prussia and R.L. Shewfelt (eds.), Georgia (2002) 303-311
3. Branting, K.L., Broos, P.S.: Automated Acquisition of User Preferences. International Journal of Human-Computer Studies, (1997) 46:55-77.
4. Branting, K.L.: Active Exploration in Instance-Based Preference Modeling, Proceedings of the Third International Conference on Case-Based Reasoning (ICCBR-99), Germany (1999)
5. Cohen, W.W., Shapire, R.E., Singer, Y.: Learning to order things. Journal of Artificial Intelligence Research (1999) 10, 243-270
6. Cubist (2000). Release 1.09, http://www.rulequest.com/cubist-info.html
7. Del Coz, J. J., Luaces, O., Quevedo, J. R., Alonso, J., Ranilla, J., Bahamonde, A.: Self-Organizing Cases to Find Paradigms. Lecture Notes in Computer Sciences, Springer-Verlag, Berlin (1999) Vol. 1606, 527-536
8. Fritzke, B.: A growing neural gas network learns topologies, Advances in Neural Information Processing Systems 7, G. Tesauro, D. S. Touretzky and T. K. Leen (eds.), MIT Press, Cambridge MA (1995) 625-632
9. Goyache, F., Bahamonde, A. Alonso, J., López, S., Alonso, J., del Coz J.J., Quevedo, J.R., Ranilla, J., Luaces, O., Alvarez, I., Royo, L. and Díez J.: The usefulness of Artificial Intelligence techniques to assess subjective quality of products in the food industry. Trends in Food Science and Technology, in press (2002)
10. Goyache, F., del Coz, J.J., Quevedo, J.R., López, S., Alonso, J., Ranilla, J., Luaces, O., Alvarez, I. and Bahamonde, A.: Using artificial intelligence to design and implement a morphological assessment system in beef cattle. Animal Science (2001), 73: 49-60
11. Kohonen, T.: Self-Organizing Maps. Springer Series of Information Science. Springer-Verlag, Berlin (1995)
12. Kusabs N., Bollen F., Trigg L., Holmes G., Inglis S.: Objective measurement of mushroom quality. Proc. New Zealand Institute of Agricultural Science and the New

Zealand Society for Horticultural Science Annual Convention, Hawke's Bay, New Zealand (1998)
13. Meilgaard, M., Civille, G.V., Carr, B.T.: Sensory evaluation techniques. CRC Press, Inc., Boca Raton, Florida (1987)
14. Murthy, S. K., Kasif, S., Salzberg, S.: A system for induction of oblique decision trees. Journal of Artificial Intelligence Research, (1994) 2, 1-32
15. Quevedo, J.R., Bahamonde, A.: Aprendizaje de Funciones Usando Inducción sobre Clasificaciones Discretas. Proceedings CAEPIA'99 VIII Conferencia de la Asociación Española para la Inteligencia Artificial, Murcia, Spain (1999) Vol. I, 64-71
16. Quinlan, J. R.: Learning with continuous classes. Proceedings 5th Australian Joint Conference on Artificial Intelligence. World Scientific, Singapore, (1992), 343-348.
17. Tesauro, G.: Connectionist learning of expert preferences by comparison training. In Advances in Neural Information Processing Systems, Proceedings NIPS'88, MIT Press (1989) 99-106
18. Utgoff, J. P., Clouse, J.: Two kinds of training information for evaluation function learning. In Proceedings AAAI'91, MIT Press (1991) 596-600
19. Utgoff, J.P., Saxema, S.: Learning preference predicate. In Proceedings of the Fourth International Workshop on Machine Learning, Morgan Kaufmann, San Francisco (1987) 115-121
20. Wang Y., Witten I.H.: Inducing of Model Trees for Predicting Continuous Classes. Proceedings of European Conference on Machine Learning. Prague, Czech Republic (1997) 128-137.