

Using Color to Help in the Interactive Concept Formation

Vasco Furtado¹, Alexandre Cavalcante²

^{1,2} University of Fortaleza – UNIFOR, Av. Washington Soares 1321, Fortaleza – CE, Brazil

¹vasco@unifor.br, ²alex@sspds.ce.gov.br

Abstract. This article describes a technique that aims at qualifying a concept hierarchy with colors, in such a way that it can be feasible to promote the interactivity between the user and an incremental probabilistic concept formation algorithm. The main idea behind this technique is to use colors to map the concept properties being generated, to combine them, and to provide a resulting color that will represent a specific concept. The intention is to assign similar colors to similar concepts, thereby making it possible for the user to interact with the algorithm and to intervene in the concept formation process by identifying which approximate concepts are being separately formed. An operator for interactive merge has been used to allow the user to combine concepts he/she considers similar. Preliminary evaluation on concepts generated after interaction has demonstrated improved accuracy.

1 Introduction

Incremental concept formation algorithms accomplish the concept hierarchy construction process from a set of observations—usually an attribute/value paired list—that characterizes an observed entity. By using these algorithms, learning occurs gradually over a period of time.

Different from non-incremental learning (where all observations are presented at the same time), incremental systems are capable of changing the hierarchical structure constructed as new observations become available for processing. These systems, besides closely representing the manner in which humans learn, they present the disadvantage that the quality of the generated concepts depends on the presentation order of the observations.

This work proposes a strategy to help in the identification of bad concept formation, making it possible to initiate an interaction process. The resource that makes this interaction possible is a color-based data visualization technique. The idea is to help users recognize similarities or differences in the conceptual hierarchies being formed. The basic assumption of this strategy is to match up human strengths with those of computers. In particular, by using the human visual perceptive capacity in identifying color patterns, it seeks to aid in the identification of poor concept formation.

The proposed solution consists of mapping colors to concept properties and then mixing them to obtain a concept color. The probability of an entity, represented by a

concept, having a particular property assumes a fundamental role in the mixing process mentioned above, thereby directly influencing the final color of the concept. At the end of the process, each concept of the hierarchy will be qualified by a color. An operator for the interactive merge has been defined to allow the user to combine concepts he/she considers similar. Preliminary evaluations on generated concepts, after such an interaction, have demonstrated that the conceptual hierarchy accuracy has improved considerably.

2 Incremental Probabilistic Concept Formation

Incremental probabilistic concept formation systems accomplish a process of concept hierarchy formation that generalizes observations contained in the node in terms of the conditional probability of their characteristics.

The task that these systems accomplish does not require a “teacher” to pre-classify objects, but such systems use an evaluation function to discover classes with “good” conceptual descriptions. Generally, the most common criterion to qualify how good is a concept is its capacity to make inferences about unknown properties of new entities.

Most of the recent work on this topic is built on the work of Fisher (1987) and the COBWEB algorithm, which forms probabilistic concepts (CP) defined in the following manner. Let: $A = \{a_1, a_2, a_3, \dots, a_n\}$ be the set of all attributes and $v(a_n) = \{v_{n1}, v_{n2}, v_{n3}, \dots, v_{nm}\}$ be the set of all values of an attribute $a_n \in A$, that describes a concept CP where $CP = \{(a_n, \{v_{nm}, P(v(a_n)=v_{nm}|C)\}) / v_{nm} \in v(a_n), 0 < m < Card(v(a_n)), 0 < n < Card(A)\}$. $P(v(a_n)=v_{nm}|C)$ indicates the probability of an entity possessing an attribute a_n with the value v_{nm} given that this entity is a member of class C (extent of CP). Then, consider the pair $p_{nm}=(a_n, v_{nm})$ as being a property of the concept CP.

The incremental character of processing observations causes the presentational order of these observations to influence the concept formation process. Consider the set of observations in the domain of animals in Table 1:

Table 1. Set of observations

	Body Cover	Heart	Body Temp.	Fertiliz	Olfact	
1	Hair	4	Reg.	Int.	Good	Mammal1
2	Scales	2	Unreg.	Ext.	Hybrid	Fish2
3	Hair	4	Reg.	Int.	Medium	Mammal2
4	Feather	4	Reg.	Ext.	Sensitive	Bird1
5	Scales	2	Unreg.	Ext.	None	Fish1

When processing the observations with COBWEB in the order of 1,3,4,5, and 2, it may be noticed that the concept hierarchy formed does not reflect an ideal hierarchy for the domain since the two mammals are not in the same class, as it can be seen in Figure 1.

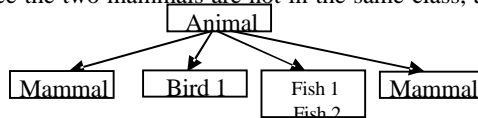


Fig. 1. Hierarchical structure generated in a bad order by Cobweb

3 Modeling Colors

In 1931, the CIE (*Comission Internationale de L'Eclairage*) defined its first model of colors. An evolution of this first CIE color model led the CIE to define two other models: the CIELuv and the CIELab, which represented that the Euclidean distance between two coordinates represents two colors, and the same distance between two coordinates represents other two colors, agreeing on the same difference in visual perception (Wyszecki, 1982).

The CIELab standard began to influence new ways for the measurement of visual differences between colors, such as the CIELab94 and the CMC (Sharma & Trussell, 1997). For this work, similarity between colors is the first key point for the solution, and it will be used extensively in the color mixture algorithm.

On the other hand, the color models may also be used to define colors according to their properties. These properties are luminosity (L), hue (H), which is the color that an object is perceived (green, blue, etc.) and saturation (S) or chrome (C), indicating the depth in which the color is either vivid or diluted (Fortner & Meyer, 1997). These color spaces are denominated HLS or HLC, and they will be further applied in this work to map colors to properties.

4 Mixing Color in Different Proportions

We defined a color-mixing algorithm following the assumption that the resulting mixture of two colors must be perceptually similar to the color being mixed that carries the higher weight in the mixing process.

The proposed algorithm considers the CIELab94 metric as a measure of similarity/dissimilarity between colors. We define the function $CIELab94(R_1, R_2)$, which measures the extent to which the color R_1 and the color R_2 resemble each other in accordance with the CIELab94 model. The range of results points out that the smaller the calculated result of the CIELab94 metric is, the greater the similarity between the colors involved will be.

4.1 Mixing Two Colors

When mixing two colors, consider that the set $CLAB = \{(L,a,b) \mid L, a \text{ and } b\}$ are coordinates of the CIELab model. The colors R_1 and $R_2 \in CLAB$, and the weights p_1 and p_2 , associated with the colors R_1 and R_2 , are such that, $p_1+p_2=1$. The color R_r , the mixture result of R_1 and R_2 , is then calculated in the following manner:

```
function Mix2Colors(R1,R2,p2)
1. Let RG be the color corresponding to the medium
   point in the line between  $R_1$  and  $R_2$ 
2. Let SimRG be the similarity between RG and  $R_1$ 
3. Let SimRR be  $p_2 \times$  similarity between  $R_1$  and  $R_2$ 
```

¹ L stands for luminosity, a stands for the red/green aspect of the color, and b stands for the yellow/blue aspect.

4. If SimRG equals SimRR then Return RG
5. If SimRG is greater than SIMRR then
 - Let R_2 be RG
 - else Let R_1 be RG
6. Go to 1

It should be underscored that a color is a point in a three dimensional space. That is why it is necessary to compute the medium point between two colors. The function *Mix2colors* searches for the color represented by this medium point, so that its similarity with the first color that participated in the mixture is equal to the weight p_2 multiplied by the similarity between the colors that are being mixed.

4.2 Mixing n Colors

To mix n colors, first, it is necessary to mix two colors, and the result obtained is mixed with the third color. This procedure is extended for n colors. The weight by which each color influences the result of the mixture is proportional to the order in which the color participates in the mixing process. For instance, the first two colors, when mixed, participate with 0.5 each. The third color, participates with 0.33, since the first two will already have influenced 0.66 of the final color. Generalizing, let i be the order that a color is considered in the process of mixing n colors, the influence of each color in the process is given by $1/i^2$. Different order of color mixture can produce

The function to mix n colors has the following steps: let $RT=\{R_1,R_2,R_3,\dots,R_n\}$ be the set of colors to be mixed, where each color $R_n \in CLAB$, and the mixture of the colors of the set RT will be accomplished by the following function:

```
function Mixncolor (RT)
  1. Let M be  $R_1$ 
  2. For i=2 to n
    Let M be Mix2colors (M, $R_i$ , (1/i))
  return M
```

The Mixncolor function accepts, as a parameter, a set of colors to be mixed (set RT), and returns a single color belonging to the set CIELab. It calls the Mix2colors function with three parameters: (i) the color R_i of the set RT, (ii) the result of the mixture (M) obtained from the two previous colors, and (iii) a weight $1/i$ for a color R_i .

5 Aiding the Identification of Poor Concept Hierarchy Formations Using Colors

The strategy developed to aid in the identification of poor concept hierarchy formation is done in two phases. The first one maps the initial colors to concept properties and the second phase mixes these colors, concluding with the resulting color of the concept.

² Different order of color mixture can produce different results, but this won't be a problem, since the same process will be applied to every concept

5.1 Initial Color Mapping of Probabilistic Concept Properties






The initial color mapping attributes, to each property $p_{nm}=(a_n, v_{nm})$ of a probabilistic concept, a color $R_{nm} \in CLAB$, so that, at the end of this procedure, a set denominated RM will be obtained, made up of all these mapped colors. To carry out this task, we have as parameters: (i) the set of properties $CR = \{(a_n, v_{nm}) / v_{nm} \in v(a_n)\}$, formed by all of the properties of CP, (ii) the value for minimum luminosity, L_{ini} , and, (iii) the value for maximum luminosity, L_{fin} . In this work, we used values between 50 and 98 for these latter parameters in order not to generate excessively dark color patterns.

The procedure initiates going through all the attributes a_n of set A , which will receive a coordinate H of the color that is being mapped. Knowing that, coordinate H of the HLC model varies from 0° to 360° , we have for the set of observations in table 1, the following values for H : 72, 144, 216, and 288.

The second step seeks to attribute the coordinates L and C , for each value of attribute a_n . First, for each value $v(a_n)$ of a_n a value of L is calculated. The third column in table 2 shows the coordinates L calculated for the set of observations in table 1. Finally, coordinate C is calculated so that its value is the biggest possible, whose transformation of all the values of H given a same L , returns only valid RGB values³ ($R \geq 0$ and $\leq 255, G \geq 0$ and $\leq 255, B \geq 0$ and ≤ 255).

Table 2 describes the mapping of H, L and C for the two first attributes of the example in Table 1.

Table 2. Example of HLC color mapping from Table 1

Atr/Valor	H	L	C	Color
$n,m=1,1$	0°	50	34	
$n,m=1,2$	0°	74	39	
$n,m=1,3$	0°	98	2	
$n,m=2,1$	72°	50	34	
$n,m=2,2$	72°	98	2	

5.2 Processing Mapped Colors

In order to complete the color qualification process of the hierarchical structure, we will consider the following parameters: (i) the RM set of the initial mapping, (ii) the conditional probabilities of the properties $P(a=v|C)$. The conditional probability of each property will function as the weight that the function *Mix2Colors* needs. As a final product, we will have set RT (input of the *Mixncolor* function). The algorithm to generate RT and its explanation follows:

```
function GenerateColor forAttr
  1. for each attribute a of a probabilistic concept
    Let  $R_r$  be the color of the first value  $v_1$  of a
    Let  $P_{acum}$  be  $P(a=v_1|C)$ 
    for  $I=2$  to number of values of a
      Let  $P_2$  be  $P(a=v_i|C)$ 
      Let  $R_r$  be Mix2Colors ( $R_r, R_i, P_2 / (P_{acum} + P_2)$ )
```

³ That heuristic aims at having RGB valid for all lines for the H, L and C being chosen.

```

Let  $P_{acum} \leftarrow P_2 + P_{acum}$ 
Insert  $R_r$  into the set  $RT$ 
Return  $RT$ 

```

To calculate the color R_r of an attribute, we mix, two by two, the colors of each value of this attribute. For such, the variable R_r is initialized with the color of the first value v_1 of attribute a . P_{acum} is set up with the conditional probability of the attribute a , which is equal to v_1 given class C . After this, the procedure enters in a loop that treats each color of the values of the attributes a , using the *Mix2colors* function. It uses the partial result R_r and the color R_i that is being processed. The parameter $P_2/(P_{acum}+P_2)$ normalizes the accumulated weight of the values so far considered and the weight of the current property P_2 , so that $P_{acum} + P_2 = 1$, as the *Mix2colors* function requires.

Finally, the generated set RT feeds the *Mixncolor* function resulting in the final color of the concept. This process is repeated for each concept of the hierarchy.

Figure 2 shows the colored concept hierarchy for the example described in section 2. Note that the colors aid the user in perceiving the need to restructure the hierarchy since the two mammals, which did not form a single class, have a similar resulting color in the eyes of the user:

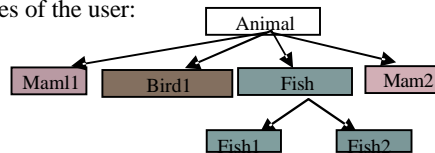


Fig. 2. Hierarchical structure qualified with colors

6 Evaluation of the Color Heuristic

We have defined an evaluation method, which seeks to prove that two things will take place:

1. Highly similar concepts will result in highly similar colors;
2. Concepts of low similarity will result in colors also of low similarity.

It is important to state that the proposed method will qualify each pair of equal concepts with equal colors. However, it cannot guarantee that similar concepts will receive similar colors, but in most cases, this will be true⁴. The evaluation process will use two basic functions. One function aims to measure the similarity between two probabilistic concepts and the other one aims to measure the similarity between two colors that represent these same concepts. The first was defined in (Talavera & Béjar, 1998), which considers two probabilistic concepts as similar if their probabilistic distributions⁵ are highly intersecting. The second is the *CIELab94*(C_1, C_2) function already seen in this article.

⁴ The proposed method doesn't guarantee this because the color space is not linear and sometimes little variation produces big color perception variation

⁵ A probabilistic distribution of a concept is the set of its properties associated to its conditional probabilities

The basic idea is to generate a concept hierarchy with associated colors and to evaluate the similarity between the concepts, two by two, in terms of similarity of content and of color. The ranges of similarities of concepts were defined as ten by ten, and for each band the average of the similarity values among the colors of the compared concepts was calculated. Three databases were considered in the tests. The first two databases are composed of animal observations, with 105 and 305 observations, and the third is the Mushrooms base (UCI, 2003), composed of 1000 observations.

Figure 3 shows the analysis defined in the previous paragraphs for the three bases considered. Note that for all bases there is a decrease of metric CIE94 as the measure of probabilistic similarity among them increases. This evidence reveals that the heuristic strategy we have defined for the concept qualification with colors reaches its main goal that is to generate similar colors for similar concepts in the greatest number of cases possible.

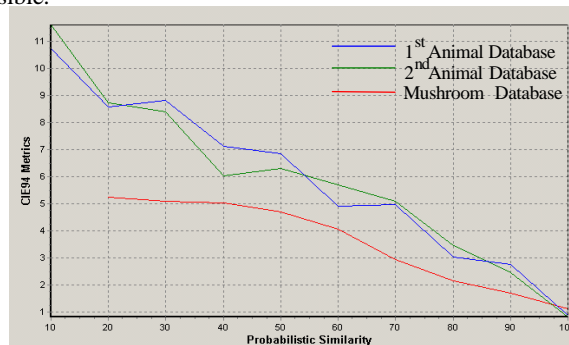


Fig. 3. Evolution of probabilistic similarity versus similarity among colors

7 Interacting with the concept formation process

The main goal of the strategy developed to qualify a hierarchical structure with colors is to make interaction between the structure and the user feasible. Thus, it is possible to improve the quality of the concept hierarchy easily because instead of accessing the probabilistic values of each concept in order to compare them one by one, the user can use his/her visual ability to have a global view of the conceptual structure and to identify similarities. The interaction is simple and intuitive because the user only has to identify two similar colors, comparing the probabilistic distribution of the concepts, and proceeds with the merge of the two colored concepts, if he/she considers interesting. To do that, we define an operator called *I-merge* similar to COBWEB's original merge operator. Unlike COBWEB's merge that only combines concepts in the same hierarchical level, with *I-merge* it is possible to merge concepts, which are in different levels of the hierarchy. The algorithm below explains the steps of *I-merge*:

```
function I-Merge (NodeOrigin, NodeTarget)
    Subtract probabilities of nodes in the path from the
    NodeOrigin to root
```

```

Merge NodeOrigin and NodeTarget (as COBWEB)
Add probabilities of nodes in the path from the
merged node to root

```

The original node counters will be used to subtract the counters from the hierarchical nodes, starting at the parent node of the original node until the root node is reached. Once that is accomplished, a merge node, resultant of the juxtaposing of the original and destination nodes, is created, and it will be hierarchically superior to the destination node, accumulating the counters of the two clustered nodes. Finally, the node counters will be updated beginning with the parent node of this node cluster, until the root node is reached.

8 Accuracy Evaluation

To evaluate whether the method proposed here has improved the accuracy of the probabilistic concept formation, an animal database with 105 observations was used. This set of observations was divided into 80 training observations and 25 test observations. The accuracy test consists in modifying a test observation by ignoring an attribute and classifying this observation in a previously built concept hierarchy. From the concept found, the algorithm must suggest a value for the attribute based on the attribute value with higher predictability. This process is performed for each attribute of each test observation. The higher the number of correct suggestions, the better the concept hierarchy is, in terms of prediction.

The procedure begins with the application of COBWEB and the visualization of the hierarchical structure formed by means of a tool that we developed to visualize colored concept hierarchies called *SmartTree*. The initial shape of the hierarchical structure is shown in figure 4 where each colored square represents a concept.

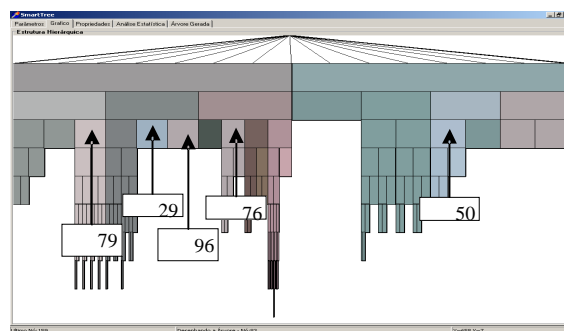


Fig. 4. Conceptual structure for ANIMALS database

For that initial structure, the inference test is carried out using the test set of 25 observations, with 47% of errors observed. The performance of the user begins at this moment. He/She observes that node 29 has a color similar to node 50. The user then asks *SmartTree* about the probabilistic similarity between them to finally decide to merge them. In this example, 3 mergers were carried out, linking the following

nodes: 29 to 50, 76 to 96, and the result of the latter to node 79. The application of the inference tests on the structure, after each merge, indicates the following evolution in the accuracy of the hierarchical structure:

- After the 1st merge: 43% of errors;
- After the 2nd merge: 38% of errors;
- After the 3rd merge: 32% of errors;

Figure 5 shows the format of the resulting tree. Besides a substantial increase in terms of accuracy, it may be verified that the resultant tree presents more uniformity with more clustered concepts.

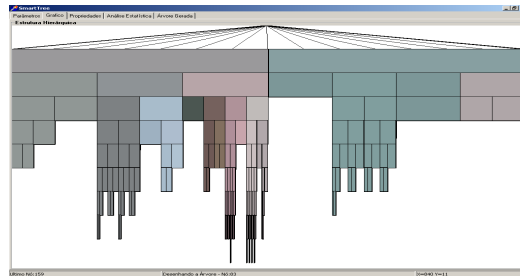


Fig. 5. Resultant tree after the interactive merge process

A second test was done with the Mushrooms database composed of 1000 observations. We use 900 training observations and 100 for testing. The initial accuracy indicated 28.2% of errors. After the first *I-merge*, that rate decreased to 25.9% and after a second *I-merge*, that rate still reduced to 25.2%.

9 Related Work

Proposals to solve the order problem are based on algorithmic alternatives implemented in the original concept formation model. The original proposal of Fisher (1987) has already considered two operators (*merge* and *split*) in an attempt to minimize the problem. Along the same lines, ARACHNE (McKusick & Langley, 1991) included two others in an attempt to adjust the hierarchy generated. The problem with these alternatives is that restructuring the tree is only done at the local partition level. Further, to reduce problems in the order of time complexity, operators only act upon the two best nodes of the partition.

Fisher et al. (1992) showed that a database that contains consecutive dissimilar observations, based on Euclidean distance, tend to form a good hierarchy. Biswas *et al.* (1994) adapted that study in the ITERATE algorithm. Later, Fisher (1996) suggested minimizing the effects of the order through an interactive optimization process running in the background.

Another line of study is based on the mingling of non-incremental techniques with those of the incremental approach. This work is exemplified in (Atlintas 1995), where instances already added to the hierarchy are reprocessed together with the new observation until a measure of structural stability is attained. Upon concluding this

phase, the process continues incrementally following the models already commented on.

10 Conclusion

We have defined a heuristic method to give colors to probabilistic concepts to allow a user to interact with the conceptual structure. Moreover, we have defined a way to user interaction via the I-merge operator, and we showed that improvements on the accuracy of concepts could be easily obtained as a result of this interaction.

This work is innovative and multidisciplinary, since it combines resources of Graphics Computation - in this case, color technology - with concept formation from Artificial Intelligence. It has demonstrated that topics related to the concept formation process, as a problem of dependence on observational presentation order, can be dealt with this focus.

Other alternatives of the use of this approach are being investigated for combining concepts, which are produced via distributed data mining in grid computing architectures. Improvements on SmartTree for elaborating different strategies for concept visualization are also in development.

References

1. Altintas, I., N.: Incremental Conceptual Clustering without Order Dependency. Master's Degree Thesis, Middle East Technical University (1995)
2. Biswas, G., Weiberg, J., & Li, C.: ITERATE: A conceptual clustering method for knowledge discovery in databases. In Innovative Applications of Artificial Intelligence in the Oil and Gas Industry, Editions Technip (1994)
3. Fisher, D. H.: Knowledge Acquisition via Incremental Conceptual Clustering. Machine Learning, 2 (1987) 139-172
4. Fisher, D., Xu, L., & Zard, N.: Order effects in clustering. Proceedings of the Ninth International Conference on Machine Learning. Aberdeen, UK: Morgan Kaufmann (1992) 163-168
5. Fisher, D.: Iterative optimization and simplification of hierarchical clusterings. Journal of Artificial Intelligence and Research, 4 (1996) 147-179
6. Fortner, B., Meyer, T. E.: Number by Colors: A Guide to Using Color to Understand Technical Data. Springer, ISBN 0-387-94685-3 (1997)
7. McKusick, K., & Langley, P.: Constraints on Tree Structure in Concept Formation. Proceedings of the 12th International Joint Conference on Artificial Intelligence, (pp. 810-816), Sydney, Australia (1991)
8. Sharma, G., & Trussell, H. J.: Digital Color Imaging, IEEE Transactions on Image Processing, Vol.6, No.7 (1997)
9. Talavera, L., Béjar, J.: Efficient and Comprehensible Hierarchical Clusterings in Proceedings of the First Catalan Conference on Artificial Intelligence, CCIA98. Tarragona, Spain, ACIA Bulletin, no 14-15, (1998) 273-281
10. UCI. In <http://www.ics.uci.edu/~mlearn/MLSummary.html/01/03/> (2003)
11. Wyszecki G., and Stiles, W. S.: Color Science: Concepts and Methods Quantitative Data and Formulae, 2nd Ed. New York, Wiley (1982)