A Balanced Differential Learning algorithm in Fuzzy Cognitive Maps

Alberto Vázquez Huerga

Departament de Llenguatges i Sistemes Informátics
Universitat Politécnica de Catalunya (UPC)
C\ Jordi Girona 1-3.
E0834, Barcelona, Spain.
avazquez@lsi.upc.es

Abstract

Fuzzy Conceptual Maps have become an important means for describing a particular domain showing the concepts (variables) and the relationship between them. They have been used for several tasks like simulation processes, forecasting or decision support. In general, the task of creating Fuzzy Conceptual Maps is made by experts in a certain domain but it is very promising the automatic creation of Fuzzy Conceptual Maps from raw data. In this paper we present a new algorithm (the Balanced Differential Algorithm) to learn Fuzzy Conceptual Maps from data. We compare the results obtained from the proposed algorithm versus the results obtained from the Differential Hebbian algorithm. Based on the results we conclude that the algorithm proposed seems to be better to learn patterns and model a given domain than in the classical approach.

Fuzzy Cognitive Maps

Cognitive Maps (CM) also called causal maps are (apart from Bayesian Networks [Pearl 1988]) a useful model to represent concepts or variables in a given domain and their causal-effect relations.
Cognitive Maps were first introduced by Axelrod [Axelrod 1976] in 1976 to model causal relations inferred between the concepts of a given domain. A CM is a directed graph, where nodes are the concepts of the given environment and arrows among nodes represent causal relations between concepts

A special kind of CM are the *Fuzzy Cognitive Maps* (FCM) that were introduced by Kosko [Kosko 1986a] to represent fuzzy cause-effect relations instead of the crisp cause-effect relations represented in the original CM. FCMs are fuzzy-signed digraphs with feedback [Kosko 1986a][Kosko 1988]. Nodes in the graph that are Fuzzy Sets representing concepts. Directed edges (arrows) represent causal-effect relations between the concepts as in the case of generic CMs. Arrows can have positive or negative values, a positive value shows a positive causal connection. As is depicted in fig1 the value of concept B increases or decreases as concept A increases or decreases. Whereas in fig2 a negative causal connection causes the value of the concept B to decrease when the value of concept A increases, and also a negative causal connection causes the value of concept B to increase when the value of concept A decreases.

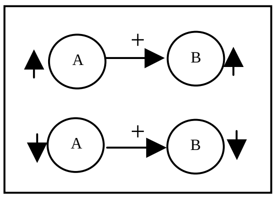

fig1.Positive causal weight

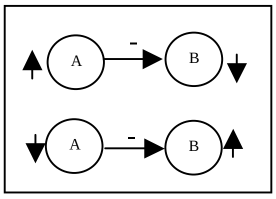

fig2. Negative causal weight

FCMs have been used as an alternative to expert systems in several areas like economics, sociology or simulation. In the literature there are a lot of implementations of FCMs to model specific environments like decision making and policy-making [Carlsson and Fullér][Stylos and Groumpos]. The process to build a FCM is similar to the process used to create a knowledge base in an expert system. First, one or more domain experts identify the concepts and their causal relationships. They talk about if two concepts have strong, weak, null, etc… causal relation. This use of linguistic labels to explain the grade of causal relation is very used in Fuzzy Logic and has a direct translation in a FCM. The "degree-of-causality" values in the connecting edges indicate how much one concept causes another. Values can range from -1, indicating a strong negative impact, through 0, or no impact, to +1, a strong positive impact.

An example is shown in fig3. We can see in the table the linguistic labels used for the domain experts and a possible automatic translation to weights in the causal web of the FCM.

SYMBOLIC VALUES	NUMERIC VALUES
Affects a lot	1.0
Affects	0.5
Do not Affects	0.0
Affects negatively	-0.5
Affects negatively a lot	-1.0

fig3. Mapping between labels and values

A FCM using this kind of tables can be easily designed by the human domain experts and can be used to have a graphical model of the domain. This model of causal relations is easy to understand by a human agent because the causal relations are expressed in a graphical way. We can see in fig4 a simple example of a FCM in the literature [Levi and Tetlock 1980] to make a straightforward model explaining how the Japanese made the decision to attack Pearl Harbor. Here in the example, there are only values of 1.0 or –1.0 represented as + or – respectively in order to make the example easier to understand the null relation between two concepts is not usually drawn. The FCM can be implemented as a non-symmetrical weighted square matrix.

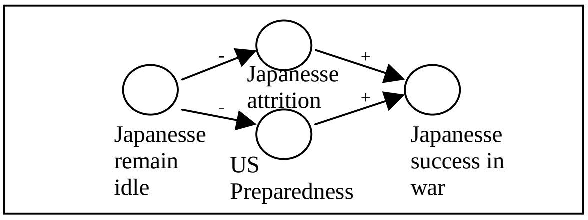

fig4. Example of a Causal Cognitive

Once the FCM has been designed we can provide an input of the actual value of every concept or variable in the environment that the FCM wants to model. Then the FCM non-linear dynamical system acts as a neural network. Every concept C_i of the input is updated through the weighted matrix of the FCM using the commonly used formula depicted in fig5 until the input converges to a fixed point (if the system reaches the equilibrium), limit cycle or a chaotic attractor [Dickerson and Kosko 1994].

$$C_i(t+1) = \theta\left[\sum_{k=1}^{N} w_{kj}(t) * C_k(t)\right]$$

$$\theta(x) = \frac{1}{1+e^{-c(x-T)}}$$

fig5. Computing the states using the sigmoid function

Adaptive Fuzzy Cognitive Maps

Sometimes it is not possible to have a human expert to model the domain because the number and complexity of the variables involved are so huge that the task to built by hand the FCM is very difficult. In this case the solution is to use adaptive fuzzy cognitive maps. An adaptive FCM learns its causal web from data. There are well-known techniques to learn causal patterns from data: Correlation or Hebbian Learning in FCMs or Temporal Associative Memories (TAM) [Amari 1972]. In [Dickerson and Kosko 1994] the Correlation encoding using the TAM method is widely explained and discussed as a poor model to infer causality because it treats in the same way zero and negative causal edges generating “spurious” causal implications.

In this sense *Differential Hebbian Learning* [Kosko 1986b] encodes causal changes to avoid spurious causality. The basic idea here is to update those weights in the causal web that are directly related to the changes in the concepts’ values. If the value concepts change in the same direction, the algorithm increases its positive causal weight in the FCM (the weight between the two concepts becomes “more” positive) otherwise if the value concepts change in opposite direction the algorithm increases its negative causal weight (the weight between the two concepts becomes “more” negative). The algorithm computes the discrete changes along time: $\Delta C_i(t) = C_i(t) - C_i(t-1)$. Being $C_i(t)$ the value of the *i* Concept at time *t*. The rule to calculate the weights can be summarised with the following formula:

$$w_{ij}(t+1) = \begin{cases} w_{ij}(t) + c_t\left[\Delta C_i(x_i)\Delta C_j(x_j) - w_{ij}(t)\right] & if \quad \Delta C_i(x_i) \neq 0 \\ w_{ij}(t) & if \quad \Delta C_i(x_i) = 0 \end{cases}$$

fig6. Computing the weights. The Differential Hebbian Learning Algorithm

The c_t is a factor used to slowly forget the old weights for the new ones. The factor is calculated with the following formula:

$$c_t(t_i) = 0.1\left[1 - \frac{t_i}{1.1N}\right]$$

fig7. Learning Factor

The main problem that the classical differential hebbian learning has is the fact that weights measure the causal-effect strength between two concepts C_i, C_j only have into account these two concepts C_i, C_j. For example we can see the following concepts:

	C_0	C_1	C_2	C_3	C_4
t_0	0.0	0.0	0.0	0.0	0.0
t_1	0.0	1.0	1.0	1.0	0.0
t_2	1.0	0.0	0.0	0.0	0.0

Fig8. Example of value evolution

The classical algorithm detects that C_1, C_2 and C_3 have changed in time t_1 while in time t_2 C_0 has changed. So, it is inferred that C_1, C_2 and C_3 are positive cause for C_0. The problem here is: the learnt weights are too general. If we use as training set the data depicted in fig8 and we apply the differential hebbian learning algorithm the FCM obtained infer that only if one of the three concepts have the value of 1.0 the concept C_0 changes to 1.0. It is clear that in general this is not true!. The correct pattern that should be learnt is the following: when the three variables have changed to 1.0 then C_0 has to change to 1.0.

A Balanced Differential Learning Algorithm

A new algorithm to update the weights is proposed based on the differential hebbian learning explained in the last section. The aim of this new approach is to avoid the problem explained in the previous section (sometimes the cause-effect relation is distributed among concepts).

Given two concepts C_i and C_j being C_j cause of C_i, the computed weight now is normalised with the number of other concepts C_k that are acting at the same time as a cause of change for the concept C_i. Another important issue is that FCMs have in the proposed algorithm diagonal links (a concept that has a causal-effect link to itself) to ensure that one concept C_i is active (value of 1.0 or close to 1.0) without the presence of another concept C_J that causes this effect in C_i. For diagonal links the weight computed in the FCM is the mean of all the values of the concept along the training set. Summarising, the proposed rule to update weights is the presented in fig 9.

$$w_{ij}(t+1)=\begin{cases} i=j & w_{ij}(t)+C_i/N \\ i\neq j \quad \Delta C_i(x_i)\Delta C_j(x_j)>0 & w_{ij}(t)+c_t\left[\dfrac{\Delta C_i(x_i)/\Delta C_j(x_j)}{\sum_{\substack{k=1..n\\ \Delta C_i(x_i)\Delta C_{ki}(x_{ki})>0}}\Delta C_i(x_i)/\Delta C_k(x_k)}-w_{ij}(t)\right] \\ i\neq j \quad \Delta C_i(x_i)\Delta C_j(x_j)<0 & w_{ij}(t)+c_t\left[\dfrac{-\Delta C_i(x_i)/\Delta C_j(x_j)}{\sum_{\substack{k=1..n\\ \Delta C_i(x_i)\Delta C_{ki}(x_{ki})<0}}\Delta C_i(x_i)/\Delta C_k(x_k)}-w_{ij}(t)\right] \end{cases}$$

fig9. Computing the weights with the Balanced Differential Algorithm

In the formula we can see the computation of the weights in terms of Probability, the diagonal links (i=j) are the probability that the concept has the value of 1.0. This probability is obtained computing the mean of all the values the concept C_j (x_j) takes along the training set.

The other causal-effect links can be seen like a conditional probability. When there is a change in a concept $C_j(x_j)$ $(\Delta C_j(x_j)\neq 0)$ to the same direction that $C_i(x_i)$ $(\Delta C_i(x_i)\Delta C_j(x_j)>0)$ the value computed updates the weight between the concepts and converts the link in a positive causal link. Otherwise if there is a change in a concept $C_J(x_J)$ $(\Delta C_j(x_j)\neq 0)$ to a different direction $(\Delta C_i(x_i)\Delta C_j(x_j)<0)$ the value computed by the algorithm updates the weight between the concepts and converts the link in a negative causal link. These weights are obtained computing the quotient $\Delta C_i(x_i)/\Delta C_j(x_j)$ that measures the proportional change factor that the concept $C_i(x_i)$ shows when the concept $C_j(x_j)$ changes along time. This value is computed normalising it with the change factor of the other concepts $C_k(x_k)$ that can be acting as a cause of the change of the concept $C_i(x_i)$. These other collateral causal relations are detected because they are changing at the same time and in the same direction than $C_j(x_j)$. The normalisation allows estimating the importance of causal relation between the concept $C_i(x_i)$ and the concept $C_j(x_j)$. If there are a lot of concepts $C_k(x_k)$ involved in changes of the concept $C_i(x_i)$ then the concept $C_j(x_j)$ has lower importance than if it was the only one concept that causes changes in the value of $C_i(x_i)$.

Experimental results

The aim of our experiments has been to compare the classical differential hebbian algorithm versus our proposed balanced differential algorithm. We have worked with four different training sets. Every training set represents a pattern of changes in the values of the concepts and is given to the algorithm repeated times until a FCM has been learnt. Then for each FCM we introduce an input that belongs to the pattern to learn and then we can see if the FCM can reproduce the limit cycle that represents the pattern. We have depicted in the next figures the value 1.0 as a white cell and the 0.0 as a white cell.

Experiment 1

The first pattern we have shown to the learning algorithm (fig 10) is quite difficult to learn because is not only determined by the value of the concepts but for the specific point of time the concepts have determined values. For example concepts c2 and c4 with value 1.0 in time t1 causes that concepts c1 and c5 change their value to 1.0 in time t2. Otherwise in time t3 concepts c2 and c4 with value 1.0 causes concept c3 to change its value from 0.0 to 1.0 and do not modify the value of c1 and c5 in this case.

	c1	c2	c3	c4	c5
t0	■	■	□	■	■
t1	■	□	■	□	■
t2	□	■	■	■	□
t3	■	□	■	□	■
t4	■	■	□	■	■

fig 10. The first pattern to learn.

Both classical and new algorithms learn and abstract model of the causal effect relation between concepts or variables.

After 15 epochs with input [0,0,1,0,0] (values of concepts in time t0 in the pattern to learn) the balanced algorithm finds a two steps limit cycle (fig11) that is the fusion of the input at time t0 and the input at time t3 of the pattern to learn. The classical algorithm shows after 30 epochs the same outcome than the proposed algorithm. In this experiment the algorithms converge to the same solution but the classical algorithm is slower than the proposed one.

	c1	c2	c3	c4	c5
t0	■	□	■	□	■
t1	□	■	□	■	□
t2	■	□	■	□	■
t3	□	■	□	■	□
t4	■	□	■	□	■

fig11. The learnt pattern (new algorithm, 15 epochs, classical algorithm, 30 epocs)

Experiment 2

In this experiment we present a pattern (depicted in fig12) with more concepts and a longer cycle to learn but with less indetermination:

	c1	c2	c3	c4	c5	c6	c7
t0	□	■	■	■	■	■	■
t1	■	□	□	□	■	■	■
t2	■	■	■	■	□	■	■
t3	■	■	■	■	■	□	■
t4	■	□	■	■	■	■	■
t5	■	■	■	□	■	■	□
t6	□	■	■	■	■	■	■

fig12. The second pattern to learn.

In fig13 we can see that the pattern learnt by our proposed algorithm in 50 epochs is quite similar to the original one. There is only one mistake, in time t5 the concept c4 takes the value 0 instead of 1. This can be explained in a similar way that in the previous example, concept c2 when changes from 0 to 1 in time t4 causes that c4 changes his value from 0 to 1 in t5. In time t1 when concept c2 changes his value from 0 to 1 in the next time (t2) c4 does not change his value. We can see that when a concept not only depends on his value but also depends on the exact point of time the concept has a value, it is very difficult to learn a pattern. This problem is not solved by any of the two discussed algorithms.

Now, in the fig14 we can see that the limit cycle learnt from the classical algorithm is not so close to the original pattern. For example we can see that with the classical algorithm it is impossible to learn that the concept c5 has to change from 0 to 1 only when concepts c2, c3 and c4 have the value of one because every concept is treated independently with respect the others.

	c1	c2	c3	c4	c5	c6	c7
t0	□	■	■	■	■	■	■
t1	■	□	□	□	■	■	■
t2	■	■	■	■	□	■	■
t3	■	■	■	■	■	□	■
t4	■	□	■	■	■	■	■
t5	■	■	■	■	■	■	□
t6	□	■	■	■	■	■	■

fig13. The learnt pattern (new algorithm) after 50 epochs.

	c1	c2	c3	c4	c5	c6	c7
t0	■	□	□	■	■	■	■
t1	■	■	■	■	□	■	□
t2	□	■	■	■	■	□	■
t3	■	□	□	■	■	■	■
t4	■	■	■	■	□	■	□
t5	□	■	■	■	■	□	■
t6	■	□	□	■	■	■	■

fig14. The learnt pattern (classical algorithm) after 100 epochs.

Experiment 3

The third experiment is similar than the previous with more concepts involved. The pattern to learn is depicted in fig15.

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
t0	□	■	■	■	■	■	■	■	■	■
t1	■	□	■	■	■	■	■	■	■	■
t2	■	■	□	■	■	■	■	■	■	■
t3	■	■	■	□	□	□	□	■	■	■
t4	■	■	■	■	■	■	■	□	■	■
t5	■	■	■	□	■	■	■	■	□	□
t6	□	■	■	■	■	■	■	■	■	■

fig15. The third pattern to learn.

The balanced differential algorithm after 40 epochs learns the given pattern without any mistakes. The reason is that this pattern has ambiguities that can be resolved because of the distribution of the causality between concepts. For example, when concept c3 changes from 0 to 1 the concept c7 only changes at next step if concept c4, c5 and c6 have changed as well. This does not occur with the classical algorithm as we can see in fig17. After more than 40 epochs the classical algorithm is not able to learn the pattern.

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
t0	□	■	■	■	■	■	■	■	■	■
t1	■	□	■	■	■	■	■	■	■	■
t2	■	■	□	■	■	■	■	■	■	■
t3	■	■	■	□	□	□	□	■	■	■
t4	■	■	■	■	■	■	■	□	■	■
t5	■	■	■	□	■	■	■	■	□	□
t6	□	■	■	■	■	■	■	■	■	■

fig16. The learnt pattern (new algorithm) after 40 epochs.

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
t0	□	■	□	■	■	■	■	□	■	■
t1	■	□	■	□	□	□	□	■	□	□
t2	□	■	□	■	■	■	■	□	■	■
t3	■	□	■	□	□	□	□	■	□	□
t4	□	■	□	■	■	■	■	□	■	■
t5	■	□	■	□	□	□	□	■	□	□
t6	□	■	□	■	■	■	■	□	■	■

fig17. The learnt pattern (classical algorithm) after 40 epochs.

Experiment 4

The last experiment has been extracted from [Dickerson and Kosko 1994]. In the paper the authors include a comparison between a learnt FCM using the differential hebbian algorithm and a hand-designed FCM. Here, we have compared the differential hebbian algorithm with our balanced differential hebbian algorithm. The inputs that we presented to the learning algorithms are the depicted in fig18. This pattern is the limit cycle that we obtain if we apply the input [1,0,0,0,0,0,0,0,0,0] to the hand made FCM that is included in the section 1.4 of [Dickerson and Kosko 1994].

Once we have applied 800 epochs to the learning algorithms we obtain two different FCM's. The FCM built with the balanced learning algorithm with input [1,0,0,0,0,0,0,0,0,0] reproduce exactly the same pattern that we presented to the algorithm (fig19). The FCM built with the classical algorithm cannot reproduce the original pattern (fig20).

	c0	c1	c2	c3	c4	c5	c6	c7	c8	C9
t0	□	■	■	■	■	■	■	■	■	■
t1	■	■	■	■	■	■	□	■	■	■
t2	■	■	■	■	■	■	■	□	■	■
t3	■	■	■	■	■	■	■	■	□	■
t4	■	□	■	■	□	■	■	■	■	■
t5	■	■	□	■	□	■	■	■	■	■
t7	■	■	□	□	□	■	■	■	■	■
t8	□	■	■	□	□	■	■	■	■	■
t9	□	■	■	■	■	■	■	■	■	■

fig18. The fourth pattern to learn

We can now have a look to the fig19 and fig20 to the FCMs generated with the two algorithms and the later training data. The links between concepts (nodes of the FCM) are similar, but there are two main differences. One of them is recurrent links in the balanced algorithm. These recurrent links (diagonal links) always have a positive value because they are the mean of the concept values that can not be smaller than 0. Diagonal links (as is explained in the previous section) are necessary to ensure that the value of a concept was the same (in mean) when there are not other concepts that could cause to change his value. We are only using a heuristic but it works very well in all the experiments. A possible explanation is that mean reflects if a concept in the training set has low or high values. When we present the test set a high value in a diagonal link leads to maintain the input value to the next iteration (in the training set this concept had high values). A low value in a diagonal link leads to obtain in the next iteration a value close to 0 like it was expected given the training set.

	c0	c1	c2	c3	c4	c5	c6	c7	c8	C9
t0	□	■	■	■	■	■	■	■	■	■
t1	■	■	■	■	■	■	□	■	■	■
t2	■	■	■	■	■	■	■	□	■	■
t3	■	■	■	■	■	■	■	■	□	■
t4	■	□	■	■	□	■	■	■	■	■
t5	■	■	□	■	□	■	■	■	■	■
t7	■	■	□	□	□	■	■	■	■	■
t8	□	■	■	□	□	■	■	■	■	■
t9	□	■	■	■	■	■	■	■	■	■

fig19. The learnt pattern (new algorithm) after 800 epochs.

If we see carefully fig19 and fig20 we will see another difference in the FCMs: most of the links have different values. The explanation is that our algorithm distributes the causal weight among nodes as we have explained in the previous section. For example, the link between concept c1 and c2 has a value of 0.49 (close to 0.5) because is not the single concept that cause c2 to change his value. It shares the causal-effect relation with the concept c5 that also have a value of 0.49 (close to 0.5). Then the model obtained is more accurate: when concept c1 has a value close to 1 the value of the concept c5 will not change unless concept c2 has a value close to 1 as well. This control of concept co-ocurrence is not possible when we apply the differential hebbian learning algorithm.

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
t0	□	■	■	■	■	■	■	■	■	■
t1	■	■	■	■	■	□	■	■	■	■
t2	■	■	■	■	■	■	□	■	■	■
t3	■	■	■	■	■	■	■	□	■	■
t4	■	□	■	■	□	■	■	■	■	■
t5	□	■	□	■	■	■	■	■	■	■
t7	■	■	■	■	■	□	■	■	■	■
t8	■	■	■	■	■	■	□	■	■	■
t9	■	■	■	■	■	■	■	□	■	■

fig20. The learnt pattern (classical algorithm) after 1600 epochs.

We can compare the FCMs (fig21 and fig22), obtained after computing the two learning algorithms. As we expected, in fig21 we can see some weights that are not so close to 1.0 or 0.0 than in fig22 because the proposed learning algorithm distributes better the causal-effect relation between variables.

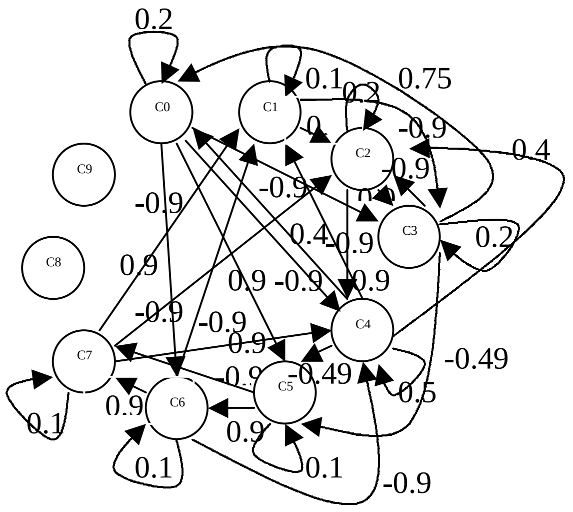

fig21. The FCM learnt after 800 epochs with the new algorithm

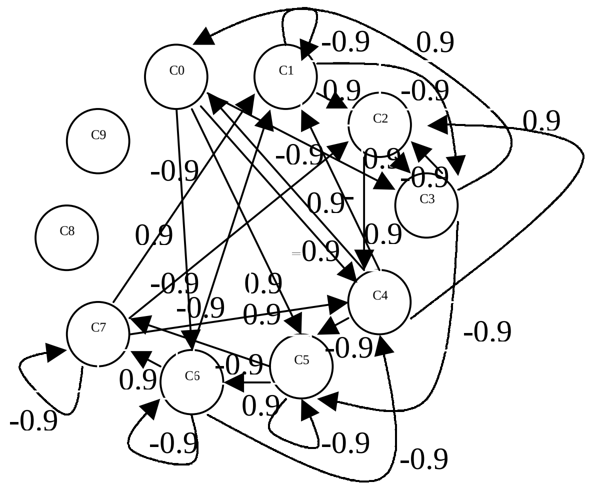

fig22. The FCM learnt after 800 epochs with the classical algorithm

Conclusions

In this paper we have studied the differential hebbian learning algorithm applied to build automatic FCMs and we have proposed an evolution of the traditional algorithm to another one that takes into account more than one concept in order to calculate the weights for causal-effect links between variables. The algorithm, that we have called Balanced Differential learning algorithm, has been implemented and tested with four different training sets and we have observed that is better than the Differential Hebbian learning algorithm. In general improves the quality of the patterns learned besides is not necessary to present the training set so many times.

The disadvantage of the presented algorithm is that is a little bit slower, to compute every weight it is necessary to consider the other concepts involved in the causal-effect relation for a target concept.

The observed results have shown that the proposed algorithm improves the FCMs obtained with training sets but still more experiments have to be done.

Finally, the area of FCM learning is very promising because the cognitive maps (FCMs) obtained are directly interpretable by a human and are a useful tool to extract information from data about the relations between the concepts or variables inside a domain.

Future Work

The algorithm has to be tested now with real data training sets and use the obtained FCMs to model real domains. In addition has to be tested with variables that could have float values not only binary values as in our examples.

The work made has allowed us to start working with inductive rule extraction and ontologies.

Inductive rule extraction

The algorithm presented is very suitable to automatic rule extraction because to build the FCM takes into account not only the value of individual concepts but the relationship between them. As we have seen the differential hebbian algorithm has connections very close to 1.0 or –1.0 in all concepts and the rules that we can obtain are in general less expressive.

Ontologies

We are very interested in the relationships that we see between formal Ontologies (or less formal like thesauri) with Fuzzy Cognitive Maps. Ontologies are very useful tools to model a particular domain and are increasingly used (every time more and more) to sort out a wide range of problems, they have an important fault though: they have to be designed by hand.

We are working with FCMs as a first step to obtain an automatic or semi-automatic tool to build Ontologies in FCMs. They can be obtained in an automatic or semiautomatic way (eg. By using a learning algorithm like the ones shown in this paper) and they model the cause-effect relationships that we do not need model by hand in the Ontology. As a future work we also will have to sort out an important issue: how to model and integrate (in an automatic way if possible) the generic-specific relationships among terms (classes and subclasses) that are commonly used in hand-made Ontologies.

References

[Axelrod 1976] Axelrod, R. 1976. Structure of Decision: the Cognitive Maps of Political Elites, *Princeton University Press*, Princeton, New Jersey.

[Kosko 1986a] Kosko, B. 1986. Fuzzy Cognitive Maps, *International Journal of Man-Machine Studies*, 24:65-75.

[Kosko 1988] Kosko, B. 1988. Hidden Patterns in Combined and Adaptive Knowledge Networks, *International Journal of Approximate Reasoning*, 2:337-393.

[Dickerson and Kosko 1994] Dickerson, J. A. and Kosko, B. 1994. Virtual Worlds as Fuzzy Cognitive Maps, *Presence,* 3(2), 173-189.

[Kosko 1986b] Kosko, B. 1986. Differential hebbian learning. *AIP Conference Proceedings* 151:265—270.

[Amari 1972] Amari, S. 1972. Learning patterns and pattern sequences by self organizating nets of threshold elements. *IEEE Transactions on Computers*, 21:1197-1206.

[Pearl 1988] Pearl, J. 1988. Probabilistic Reasoning in Intelligent Systems. *San Mateo, CA: Morgan Kauffman.*

[Carlsson and Fullér 1996] Carlsson, C and Fullér, R. 1996. Adaptive Fuzzy Cognitive Maps for Hyperknowledge Representation in Strategy Formation Process, *Proceedings of International Panel Conference on Soft and Intelligent Computing,*Technical University of Budapest 43-50.

[Stylios and Groumpos] Stylios, C.D. and Groumpos, P.P. 2000. Fuzzy Cognitive Maps in Modelling Supervisory Control Systems. *Journal of Intelligent & Fuzzy Systems* 8, 2:83-98.

[Levi and Tetlock] Levi, A and Tetlock, P. E. 1980. A Cognitive Analysis of Japan's 1941 Decision for War. *Journal of Conflict Resolution*, 24:195-211.