Proceedings of the 6th Győr Symposium and 3rd Hungarian-Polish
and 1st Hungarian-Romanian Joint Conference on Computational Intelligence

Possible manners of Fuzzy Cognitive Maps' state reduction

Miklós F. Hatwágner[1], László T. Kóczy[2]

Széchenyi István University, Egyetem sqr. 1., H-9026 Győr, Hungary
[1]Dept. of Information Technology
E-mail: miklos.hatwagner@sze.hu
[2]Dept. of Automation
E-mail: koczy@sze.hu

Abstract: Fuzzy Cognitive Maps (FCM) is a powerful tool to analyze and model the behavior of complex systems. One of the most important elements of FCM is the connection matrix. This square matrix describes the direction and weight of the connections between the different parts (called concepts in FCM theory) of the modeled system. If the number of concepts is only 5-10, the model is clear and easy to use. However in some cases the whole system cannot be described with so many concepts to achieve the appropriate accuracy. The increased number of concepts helps to make the model more accurate but makes the model complicated as well, therefore confusing and less usable. Sometimes it is not obvious at first sight for the modelers, which concepts are really important and which ones are not. The state reduction methods described in this paper can help to reduce the number of concepts in FCM models and keep the accuracy at an acceptable level.

Keywords: Fuzzy Cognitive Maps, state reduction

1. Brief introduction to the FCM

Fuzzy Cognitive Maps (FCM) is a very effective but simple and convenient tool to describe, analyze, model or forecast the behavior of complex systems. FCM is a fuzzy graph structure, and can be considered as a combination of neural networks and fuzzy logic [6]. The representation of FCM by directed, signed fuzzy graph was introduced by Kosko [5].

The nodes/vertices of the graph are called concepts. The concepts represent the components of the examined system. The edges connecting the nodes represent the direction and

importance (weight) of the relations between concepts. During a simulation, the concepts interact with each other, and the dynamics of the modeled system can be observed. If the initial state of the nodes are known, the states in the later time steps can be calculated by a simple loop (see Eq. 1). The loop can be iterated until a limit is reached or an equilibrium state is achieved [3].

$$V_{k+1} = f(N \cdot V_k) \tag{1}$$

In Eq. 1, V_k is the state of the system in the k^{th} step and N is the connection matrix. This matrix contains the weights (w_{ij}) of the edges between concepts (C_i and C_j). The main diagonal of the matrix contain only zeros. f is the state transition function (see Eq. 2).

$$f(x) = \frac{1}{1 + e^{-\lambda x}} \tag{2}$$

λ defines the steepness of the function, and $\lambda > 0$. The concept values have to be in the range $[0,1]$, but the weights are in the $[-1,1]$ double unit interval.

The initial state of the concepts and the connection matrix is generally given by experts/stakeholders, thus it can contain subjective elements. However the model should be objective, thus careful filtering and cumulating of the experts' knowledge is very important [2].

2. State reduction methods

Certain systems, e.g. an Integrated Waste Management System (IWMS) are rather complex and six concepts are not enough to model the system with the required accuracy [1]. The possible solution of this problem is to use more concepts, that is, to increase the complexity of the model. In the above mentioned case, the number of concepts were increased from six to thirty-three. Unfortunately, this process makes the model harder to understand and use in practice. There is a practical demand to make the extended model as simple as possible, but keep it's ability to represent the system's behavior with the required accuracy.

Three state reduction methods will be described in the following. These methods are quite similar, only some details (e.g. the used metrics) are different. In some sense, they can be considered as a generalization of the state reduction of finite state machines and sequential systems. The main idea behind the methods is to create clusters of concepts and use these clusters as concepts to create a simpler model.

At first, there is a one-to-one bind between clusters and concepts: every cluster contain one of the concepts. With other words, $K_i = \{C_i\}$ for every $i = 1 \dots n$ where K_i is the i[th] cluster, C_i is the i[th] concept and n is the number of concepts. In the following steps an agglomerating strategy is used to reduce the number of clusters. In order to apply this strategy, the "distance" between the current cluster (the elements of the cluster) and the concepts not included in the current cluster have to be calculated using an appropriate metrics.

2.1. Metric "A"

The first method uses metric "A". It measures the absolute differences between 1) C_i and C_k, and 2) C_j and C_k, where $i,j,k = 1 \dots n$, $i \neq j \neq k$. If the differences are less than a specified ϵ value ($|w_{ik} - w_{jk}|/2 < \epsilon$ and $|w_{ki} - w_{kj}|/2 < \epsilon$), the concept is added to the current cluster. Fig 1 shows the applied metrics, while Fig 2 gives the exact description of the cluster building process.

```
function isNearA(i, j, eps) // i, j = concept indexes
  near = true;
  for(k=0; k<n and near==true; k++) // n = no. of concepts
    if(k!=i and k!=j)
      if((abs(w(i, k) - w(j, k))/2.) >= eps or
         (abs(w(k, i) - w(k, j))/2.) >= eps)
        near = false
  return near
```

Figure 1. Calculation of metric "A"

The state reduction process starts with the execution of `buildAllClusters` function. This function requests the creation of clusters using different initial concepts and guarantees the uniqueness of the clusters. The cluster building itself is the job of the `buildCluster` function. This function adds the concept under investigation to the cluster if the return value of `isNearA` makes it possible.

The number of concept merges depends on the value of ϵ. If the value of it is too low, only a few merge can be made, and the model remains too complex. On the other side, high ϵ value result in very simple and inaccurate model. The value must be in the $[0,1]$ interval, and have to be determined in every different case.

In the case of the above mentioned IWMS, quite high ϵ were needed to reduce the

```
function buildCluster(initialConcept, eps)
  c = {initialConcept}
  for(i=0; i<n; i++)
    if(i != initialConcept)
      member = true
      while(member and hasNextElement(c))
        j = nextElement(c)
        member = isNearA(j, i, eps)
        if(member)
          c = c + {i}
  return c

function buildAllClusters(eps)
  clusters = {}
  for(i=0; i<n; i++)
    k = buildCluster(i, eps)
    if(!isElementOf(k, clusters))
      clusters = clusters + {k}
  return clusters
```

Figure 2. C-style pseudo-code of the state reduction algorithm, part 1

number of concepts in the desired extent (see Table 1), because a small proportion of the concepts were often "too different" from the members of the clusters.

2.2. Metric "B"

A second method was developed using metric "B" in order to lower the value of ϵ (see `isNearB` in Fig 3). In this case the current concept is added to the cluster if the difference between concepts is less than a specified value in a predefined proportion (p) of the cases. The other parts of the method remained the same.

This method has two important parameters: ϵ and p. The goal of ϵ is the same as before. Small p values help to merge concepts into the same cluster if the "distance" between the concepts of the cluster and the concept under investigation is greater than ϵ in an insignificant number of cases. However high values of p would make possible to merge everything into the same cluster, therefore high p values should be avoided. Because p means "probability", it must be in the $[0,1]$ interval.

Table 1. The number of concepts in the reduced connection matrix, using different metrics

Metric "A"		Metric "B"			Metric "C"	
ϵ	No. of concepts	ϵ	p	No. of concepts	ϵ	No. of concepts
0.3	28	0.1	0.2	30	0.011	30
0.4	25	0.2	0.05	30	0.016	28
0.5	18	0.2	0.1	26	0.022	24
0.6	15	0.2	0.2	23	0.027	22
0.7	12	0.3	0.05	23	0.04	20
0.8	4	0.3	0.1	21	0.048	18
		0.3	0.2	15	0.054	15
		0.4	0.05	19	0.06	12
		0.4	0.1	10		

```
function isNearB(i, j, eps, p) // i, j = concept indexes
  near = 0
  far = 0
  for(k=0; k<n; k++) // n = number of concepts
    if(k!=i and k!=j)
      if((abs(w(i, k) - w(j, k))/2.) < eps)
        near = near + 1
      else
        far = far + 1
      if((abs(w(k, i) - w(k, j))/2.) < eps)
        near = near + 1
      else
        far = far + 1
  if(near==0 or far/near >= p)
    return false
  else
    return true
```

Figure 3. Calculation of metric "B"

The results given by metric "B" can be observed in Table 1 as well. It can be seen, that metric "B" made possible to decrease to value of ϵ, but the authors have investigated a third metric, too.

2.3. Metric "C"

The third method uses metric "C". The function `isNearC` uses normalized, squared Euclidean distance. This method gave the best results (see Table 1).

```
function isNearC(i, j, eps) // i, j = concept indexes
  sum = 0;
  for(k=0; k<n; k++) // n = number of concepts
    if(k!=i and k!=j)
      dout = w(i, k)-w(j, k)
      sum = sum + dout * dout
      din = w(k, i)-w(k, j)
      sum = sum + din * din
  if(sum / ((n-2)*8) < eps)
    return true
  else
    return false
```

Figure 4. Calculation of metric "C"

2.4. Determining the weight of the merged concepts

Having the concepts reduced, the weights of the new connection matrix has to be calculated. This is the goal of the `getWeight` function (see Fig 5). The calculated weight is the average weight of the connections between two clusters a and b. Self-loops were not taken into account.

3. Conclusion

The modeling of complex systems using FCM sometimes needs a lot of concepts. These models are very complex and are hard to use and understand. In these cases the three state reduction methods investigated in this paper can help to make the model simpler while retain the required accuracy. The third method that uses metric "C" produced the best

```
function getWeight(a, b)
  count = 0
  sum = 0
  while(hasNextElement(a))
    i = nextElement(a)
      while(hasNextElement(b))
        j = nextElement(b)
        if(i != j)
          w = w(i, j)
          sum = sum + w
          count = count + 1
  if(count == 0)
    return 0
  else
    return sum/count
```

Figure 5. C-style pseudo-code of the state reduction algorithm, part 2

results in the course of modeling an Integrated Waste Management System. To define the parameter values of the methods (p, ϵ) the help of experts and some experimenting is necessary.

The suggested fuzzy state reduction methods can be considered as a generalization of the reduction of finite state machines and it is based on fuzzy tolerance relations [4].

References

[1] Buruzs, A., Hatwágner, M. F., Pozna, R. C. and Kóczy, L. T.: *Advanced Learning of Fuzzy Cognitive Maps of Waste Management by Bacterial Algorithm*, in Proceedings of IFSA World Congress and NAFIPS Annual Meeting, IEEE, pp. 890–895, 2013

[2] Isak, K. G. Q., Wildenberg, M., Adamescu, M., Skov, F., De Blust, G. and Varjopuro, R.: *A Long-Term Biodiversity, Ecosystem and Awareness Research Network Manual for Applying Fuzzy Cognitive Mapping – Experiences from ALTER-Net*, Project no. GOCE-CT-2003-505298, ALTER-Net Deliverable type: Report, WPR6-2009-02 - Deliv-erable 4.R6.D2., 2009

[3] Ketipi, M. K., Koulouriotis, D. E., Karakasis, E. G., Papakostas, G. A. and Tourassis, V. D.: *A Flexible Nonlinear Approach to Represent Cause–effect Relationships in FCMs*, J. of Applied Soft Computing, vol. 12, issue 12, pp. 3757–3770, 2012

[4] Klir G. J., Folger T. A.: *Fuzzy Sets, Uncertainty and Information*, Prentice Hall, 1987

[5] Kosko, B.: *Fuzzy Cognitive Maps*, Int. J. of Man–Machine Studies, vol. 24, no. 1, pp. 65–75, 1986

[6] Stylos, C. D., Georgopoulos, V. C. and Groumpos, P. P.: *The Use of Fuzzy Cognitive Maps in Modeling Systems*, in Proceedings of 5th IEEE Mediterranean Conf. on Control and Systems, Paphos, Cyprus, 1997