A Game-Theoretic Approach to Competitive Learning in Self-Organizing Maps

Joseph Herbert and JingTao Yao

Department of Computer Science
University of Regina
Regina, Saskatchewan
Canada, S4S 0A2
{herbertj, jtyao}@cs.uregina.ca

Abstract. Self-Organizing Maps (SOM) is a powerful tool for clustering and discovering patterns in data. Competitive learning in the SOM training process focusses on finding a neuron that is most similar to that of an input vector. Since an update of a neuron only benefits part of the feature map, it can be thought of as a local optimization problem. The ability to move away from a local optimization model into a global optimization model requires the use of game theory techniques to analyze overall *quality* of the SOM. A new algorithm GTSOM is introduced to take into account cluster quality measurements and dynamically modify learning rates to ensure improved quality through successive iterations.

1 Introduction

Self-Organizing Maps (SOM), introduced by Kohonen [1], is an approach to clustering similar patterns found within data [2, 3]. Used primarily to cluster attribute data for pattern recognition, SOMs offer a robust model with many configurable aspects to suit many different applications.

The training of a SOM does not take into consideration certain advantages that could be obtained if multiple measures were used in deciding which neuron to update. Recent research that makes use of dynamic adaptive and structure-adaptive techniques have been proposed [4, 5]. Game theory offers techniques for formulating competition between parties that wish to reach an optimal position. By defining competitive learning in terms of finding a neuron that can perform an action that will improve not only its own position, but also the entire SOM, we may be able to improve the quality of clusters and increase the efficiency of the entire process, moving towards a global optimization process from local optimization found in traditional SOM methods.

This article proposes a new algorithm GTSOM that utilize aspects of game theory. This allows for global optimization of the feature map. This technique could be used to ensure that competitive learning results in the modification of neurons that are truly suitable for improving the training results.

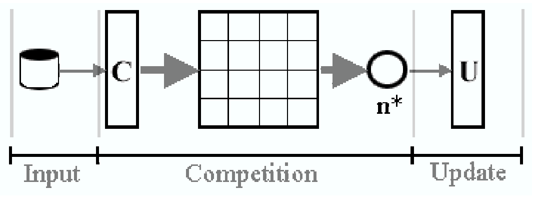

Fig. 1. The layers of a SOM during the training process.

2 A Brief Review of Self-Organizing Maps

At the heart of SOM theory is the concept of creating artificial neurons that are computational duplicates of biological neurons within the human brain [6]. Artificial neural networks follow the model of their biological counterparts. A SOM consists of neurons with weight vectors. Weight vectors are adjusted according to a learning rate α that is decreased over time to allow for fast, vague training in the beginning and specific, accurate training during the remaining runtime.

A SOM model contains three fundamental procedures that are required in order to discover clusters of data. These procedures are similar to that of the knowledge discovery in database process [7, 8]. The first procedure consists of all preprocessing tasks that are required to be completed before training can take place. This includes initializing the weights vectors of each neuron either randomly or by some other method [9, 10]. Another task to be performed is that of input vector creation. Training data for the SOM must be arranged in input vectors, where each vector represents a tuple in an information system or other similarly organized data set.

2.1 SOM Training

In order for a SOM to cluster data, it must be trained with suitable training data. Training a SOM requires the combination of three layers that work in tandem, where an output of one layer is treated as the input to the next layer, as shown in Figure 1.

The first layer, denoted as the input layer, consists of a data store to be formatted into a set of input vectors P. An input vector represents a *tuple* within the data set. Each input vector $\boldsymbol{p} \in P$ is used as input for the next layer of a SOM. The second layer, denoted as the competition layer, manages the competitive learning methods within the SOM [11]. This layer determines which neuron n_i has a weight vector $\boldsymbol{w}_i$ with minimum distance (maximum similarity) to $\boldsymbol{p}$. From this layer, a winning neuron n_i^* is marked to be updated in the third layer. The third layer, denoted as the update layer, updates the weight vector associated with the winning neuron that was used as input. After the updating of the neuron, the weight vector is more attuned to that of the input vector.

```
Data: A set of $m$ input vectors $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m\}$
Input: A threshold $q_m$ for maximum iterations to be executed.
Output: A feature map $A'$
1  for each neuron $n_i \in W$ do
2      initialize $\boldsymbol{w}_i$ randomly ;
3  end
4  while $(q \leq q_m)$ or $(\forall\ \boldsymbol{p}_k \in P, n_i^*(q) = n_i^*(q-1))$ do
5      $\alpha_q$ = adjusted $\alpha_{q-1}$ for iteration $q$ ;
6      $d_q$ = adjusted $d_{q-1}$ for iteration $q$ ;
7      for each $\boldsymbol{p}_k \in P$ do
8          $n_i^*(q) = \mathrm{Compet}(\boldsymbol{p}_k, W)$ ;
9          Update_w$(n_i^*(q), \boldsymbol{p}_k, \alpha_q)$ ;
10         Update_N$(N_{n_i^*(q)}(d_q), \boldsymbol{p}_k, \alpha_q)$ ;
11     end
12 end
```

Algorithm 1: The SOM Training Method

A data set P contains individual *tuples* of an information system translated into input vectors, $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m\}$. A set of artificial neurons, $W = \{n_1, \ldots, n_n\}$, is arranged in a grid-like topology of fixed dimensionality. Each neuron in W has a weight vector $\boldsymbol{w}_i$ of the same dimensionality as the input vectors $\boldsymbol{p}_j$.

Each neuron $n_i \in W$ has a set of neurons whose proximity is within that defined by d, a scalar whose value is changed according to an iteration q. Therefore, for each neuron n_i, the neighborhood $N_i(d) = \{n_r, \ldots, n_s\}$ consists of all neurons that have connectivity to n_i within distance d. The learning rate is used as a scalar that determines how much a weight vector $\boldsymbol{w}_i$ is changed to become more similar to that of the current input vector.

2.2 Competitive Learning in SOM

To find a neuron $n_i \in W$ that has a weight vector closest to $\boldsymbol{p}_k$, similarity measures [12] are observed between each neuron and the input vector.

Once a winning neuron n_i^* has been identified, the weight vector must be updated according to the learning rate α_q corresponding to iteration q. In addition, the neighborhood of that neuron must be updated so that neurons connected to the winner reflect continued similarity to the new information presented to the network. In Algorithm 1, this process is done with functions Update_w and Update_ N, functions that update the winning neuron and its neighborhood respectively. The update of a winning neuron and the update of the winning neuron's neighborhood is shown in Equation 1 and Equation 2 respectively. Equation 1 is known as the Kohonen rule [6].

$$\boldsymbol{w}_i^*(q) = \boldsymbol{w}_i^*(q-1) + \alpha(\boldsymbol{p}_k(q) - \boldsymbol{w}_i^*(q-1)) \ . \tag{1}$$

$$\boldsymbol{w}_{N_{i^*}(d)}(q) = \boldsymbol{w}_{N_{i^*}(d)}(q-1) + \alpha\prime(\boldsymbol{p}_k(q) - \boldsymbol{w}_{N_{i^*}(d)}(q-1)) \ . \tag{2}$$

The modified learning rate $\alpha\prime$ denotes a smaller learning rate that is used on the neurons in $N_{i^*}(d)$. We wish to use a smaller learning rate to signify that although these neurons did not win the competition for the input vector, they do have some connectivity to the neuron that did. The learning rate α in Equation 1 is derived from a decreasing polynomial formula [13].

Algorithm 1 shows the steps taken to train the SOM. The process of updating a neuron and its neighbors can be thought of as a local optimization procedure. For any given input vector, the update layer in Figure 1 only adjusts neurons based on a very small instance of the overall patterns in the full data set.

3 Incorporating Game Theory into SOM Training

Although individual neurons have the ability to improve their situation during each competition, a collective goal for the entire SOM is not considered. The transition between local optimization techniques to those of global optimization must occur in order to solve problems of density mismatch and physical adjacency errors. The concept of overall SOM quality must be defined in order to progress to a state in which properties between overall neuron relationships and input vectors can be measured.

3.1 Measuring SOM Quality

The competitive layer in the traditional SOM model does not have the ability to find a neuron which best represents the current input vector as well as having the ability to improve the quality of neuron placement and density. Improving quality in a SOM could include an increased ability to create and define better clusters. In order to determine the quality of a SOM, definitions on what is considered a high-quality cluster must be discovered. Clusters can be defined in two ways: by the actual input data that was used to adjust the weight vectors or by the neurons associated with that data.

With the two abilities to define clusters, two methods of representing clusters arise. A centroid vector can be used as a representation of the cluster. This vector could be calculated by taking the average of all weight vectors that the cluster includes. Second, a neuron whose weight vector is most similar to that of the average weight vector of all neurons could be given representation status. In addition to the two methods of representing clusters in a SOM, two methods can be used in order to find a neuron required in the latter method:

1. If a centroid input vector for a cluster is known, we can simply discover which neuron that centroid input vector is most similar to.
2. If we wish for the calculations of centroid to be strictly neuron based, we can find groups of neurons and determine which of those neurons have won more competitions.

The above methods allow us to measure the overall quality of a SOM. Using the ability to calculate physical distance between clusters on the feature map as well as the ability to calculate the density of a particular cluster can enable a new algorithm to determine which neuron is best suited to be updated. These quality measures can be used together to see how much a particular neuron, if updated, can improve the overall quality of the feature map.

3.2 Game Theory

In order to facilitate global optimization techniques in competitive learning, a method must be employed that can take into consideration possible improvements of overall SOM quality. Game theory provides a suitable infrastructure to determine which neurons provide the best increase in feature map quality. By manipulating the learning rate applied to both the winning neuron and its neighbors, as well as the size of a neighborhood that should be taken into consideration, a set of strategies with expected payoffs can be calculated.

Game theory, introduced by von Neumann and Morgenstern [14], has been used successfully in many areas, including economics [15, 16], networking [17], and cryptography [18, 19]. Game theory offers a powerful framework for organizing neurons and to determine which neuron may provide the greatest increase in overall SOM quality.

In a simple game put into formulation, a set of players $O = \{o_1, \dots, o_n\}$, a set of actions $S = \{a_1, \dots, a_m\}$ for each player, and the respective payoff functions for each action $F = \{\mu_1, \dots, \mu_m\}$ are observed from the governing rules of the game. Each player chooses actions from S to be performed according to expected payoff from F, usually some a_i maximizing payoff $\mu_i(a_i)$ while minimizing other player's payoff. A payoff table is created in order to formulate certain payoffs for player strategies, which is shown in Table 1.

3.3 Game-Theoretic Competitive Learning in SOM

With the ability to precisely define neuron clusters within a SOM, measures can be used in order to define overall quality of the network. These measures, such as the size of clusters, the distance between clusters, and the appropriate cluster size to represent input can be combined to give a certain payoff value to a particular neuron, if chosen as a winner. When the competitive phase begins, a ranking can be associated with each neuron according to its distance from

		$n_j^*(q)$		
		$a_{j,1}$	$\dots$	$a_{j,r}$
$n_i^*(q)$	$a_{i,1}$	$< \mu_{i,1}, \mu_{j,1} >$	$\dots$	$< \mu_{i,1}, \mu_{j,r} >$
	$\vdots$	$\vdots$	$\dots$	$\vdots$
	$a_{i,r}$	$< \mu_{i,r}, \mu_{j,1} >$	$\dots$	$< \mu_{i,r}, \mu_{j,r} >$

Table 1. Payoff table created by second Competition layer.

the input vector. Using the ranked list of neurons, a new competition layer is constructed in order to determine which neuron and which strategy or action should be taken. This new model architecture is shown in Figure 2.

The first Competition layer is modified so that instead of determining which neuron is most similar to the current input vector, the layer now ranks neurons according to each similarity measure obtained. There is an opportunity here to include a dynamic, user-defined threshold value t_1 that can deter any neurons that are beyond a certain similarity measure to be included in the ranked set as shown in Equation 3 and Equation 4:

$$W' = \{n_1^*(q), \ldots, n_n^*(q)\} \ , \tag{3}$$

where $\forall n_i^*(q) \in W$,

$$|\boldsymbol{w}_i^*(q) - \boldsymbol{p}_i| \leq t_1 \ , \tag{4}$$

and $1 \leq i \leq n$. This allows the user to specify a degree of minimum similarity desired when having the first competition layer computing which neurons should enter the second competition layer.

Once a ranked set of neurons has been created, the second competition layer starts to create competition tables of the form shown in Table 1. A neuron n_i^* with possible actions $S = \{a_{i,1}, \ldots, a_{i,r}\}$ and payoffs calculated from corresponding utility functions $U = \{\mu_{i,1}, \ldots, \mu_{i,r}\}$ competes against neuron n_j^* with the same action and utility sets. The neuron whose specific action $a_{i,k}$ results in the greatest overall SOM quality is chosen to be the winner.

With the addition of quality measures, neurons are now ranked in partial order. For example, a particular neuron n_i^* may have a higher ranking than n_j^* in terms of a particular similarity measure between itself and the input vector, but the neuron may not have that same ranking when additional quality measures are taken into account. The second competition layer must take into consideration not only similarity to input, but also how much each neuron can increase or decrease feature map quality. Many different ranking of neurons in W' may occur when more than one measure is used.

There are two possible ways of creating tables to govern the second phase of competition. First, neurons can be initially paired randomly with each other.

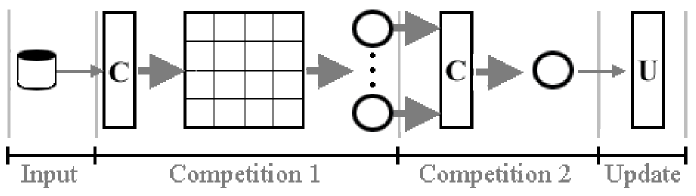

Fig. 2. The layers of GTSOM including the addition of another competition layer used during the training process.

Victors of each "round" move on to the next round, where new tables are created for the neurons that have been awarded victories. This process proceeds until a total victory is declared for one neuron. Second, for a set $W = \{n_1^*(q), \dots, n_n^*(q)\}$ of ranked neurons, an *n-dimensional* payoff table can be created. With n neurons ranked and entering competition, each with r possible actions, a total of r^n cells must be observed to determine which neuron gives the best quality or utility value for this iteration.

3.4 A Strategy to Adjust the Learning Rate

Actions performed by a particular neuron could possibly include parameters such as adjustable learning rates or adjust neighborhood size. Such actions can be called *strategies* to describe an action that can be modified in order to create new actions. A strategy of adjust the learning rate α can be modified so that there is an action for an increased adjustment, decreased adjustment, and a no-change scenario. This strategy can improve clusters by forcing subsequent input vectors that are similar to the current input to have a greater possibility to be more similar to a different neuron than it did on a previous iteration in the case of an increased learning rate. That is, the input vector will have an increased likelihood to be closer to a different neuron next iteration. A decreased learning rate will result in a diminished similarity adjustment between the victor and the current input vector, resulting in negligible change from subsequent iterations.

A set of actions detailing neighborhood size for a particular neuron is useful when cluster sizes are desired to either grow or diminish. An increased neighborhood size will modify a larger number of neurons to become more similar to the current input vector. This may result in less dense clusters if desired. In contrast, a decreased neighborhood size could have an exact opposite effect, decreasing the size and increasing the density of clusters. If clusters are too far apart, the density of a particular cluster could be dismissed so that cluster boundaries become closer. Also, if clusters are too compact, the density of some clusters could be increased in order to increase distance between centroids.

4 GTSOM Implementation

The process of ranking neurons according to similarity, creating payoff tables, and determining winning neurons is introduced in Algorithm 2. Training will stop when either of the following three conditions are met on line **4**.

1. If a maximum number of specified iterations have been performed.
2. If no neurons have won competitions for new input vectors that were not won before during previous iterations.
3. If the overall quality of the SOM has reached or moved beyond that of a user-defined threshold.

A traditional SOM stops when the first two stopping conditions are met. With the addition of the third condition, training time may be reduced if a certain

```
**Data**: A set of $m$ input vectors $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m\}$
**Input**: A threshold $q_m$ for maximum iterations to be executed.
**Output**: A feature map $A'$
1  for each neuron $n_i \in W$ do
2      Initialize $\boldsymbol{w}_i$ randomly ;
3  end
4  while $(q \leq q_m)$ or $(\forall\, \boldsymbol{p}_i \in P, n_i^*(q) = n_i^*(q-1))$ or $(\mu(A) \geq t_2)$ do
5      $\alpha_q$ = adjusted $\alpha_{q-1}$ for iteration $q$ ;
6      $d_q$ = adjusted $d_{q-1}$ for iteration $q$ // neighborhood distance ;
7      for each $\boldsymbol{p}_k \in P$ do
8          Find set $W' = \{n_1^*(q), \ldots, n_n^*(q)\}$ ;
9          for each $< n_i^*(q), n_j^*(q) >$ pair in $W'$ do
10             $T_{i,j} = (N, S_{i,j}, F_{i,j})$, where
11             $N = \{n_i^*(q), n_j^*(q)\}$,
12             $S_{i,j}$ =set of actions for $n_i^*(q)$ and $n_j^*(q)$,
13             $F_{i,j}$ =set of utility functions returning quality of $A$.
14             $\alpha_q = \pm a_i^*$, where $a_i^*$ =the action that best improves A. ;
15         end
16         Choose $n_q^*(\boldsymbol{p}_i)$ whose utility function $\mu_i$ has maximum payoff action $a_i$ ;
17         Update_w($n_i^*(q), \boldsymbol{p}_k, \alpha_q$) // update winning neuron ;
18         Update_N($N_{n_i^*(q)}(d_q), \boldsymbol{p}_k, \alpha_q$) // update neighborhood of $n^*$ ;
19     end
20 end
```

Algorithm 2: The Training Method GTSOM

quality has been reached. For example, if the desired quality of the feature map has been reached before q_m iterations have been performed, training may stop ahead of schedule. This threshold may correlate with the number of iterations that are to be performed or it may represent the desired precision of weight vectors belonging to individual neurons. A lower threshold will most likely result in a lower number of iterations performed. As precision increases with respect to the number of iterations performed (smaller learning rate), a lower number of iterations will result in the algorithm completing with a learning rate above that of the final desired learning rate.

Lines **7-19** iterate the first and second competition layers for every input vector in P. Line **8**, executing the first competition layer, creates a set of ranked neurons according to their similarity to the input vector. The third embedded repetitive structure ranks neurons according to their similarity to the current input vector. An interesting opportunity arises here when clusters are starting to be defined. There may be an option to include centroid neurons in this set once they have been discovered. This leads to the eventuality that no new clusters will be formed. Another user-defined threshold could be specified if this method is used, comparable to the maximum number of clusters desired. This also decreases the number of distance measures to be calculated between the neuron weight vectors and the current input vector.

The second competition layer is shown in lines **9-15**. Using the set of ranked neurons, tables are created for each neuron pair within W'. This table $T_{i,j} = (N, S_{i,j}, F_{i,j})$, the *payoff table for neurons* n_i *and* n_j, includes the neurons themselves, a set containing actions S_i and S_j for the neurons, and a set containing utility functions F_i and F_j that returns the quality of the feature map given action $a_i \in S_i$ and $a_j \in S_j$. Once these tables have been created, the neuron with the action that provides the greatest increase in feature map quality through the utility function is chosen as the final winner in the competition process. The action is executed (learning rate modification or neighborhood size) and update procedures are performed.

A large value for t_1 may result in increased computation time as it will result in a larger W'. Since tables are created and observed for each distinct pair of neurons within W', the similarity threshold must be considered carefully. A value too small for t_1 may result in incomplete competition, where neurons that may offer valuable actions could be ignored based on their dissimilarity to the current input vector.

The threshold t_2 found on line **4** gives the option of stopping the training process when a certain overall SOM quality has been reached. Too high of a threshold, although perhaps representing a high quality preference, may result in no computational efficiency improvement. This threshold may never be reached before maximum iterations have occurred. Too low of a threshold could result in too few iterations being performed. Since the learning rate α is adjusted during each iteration, it may not get an opportunity to become sufficiently small for precise weight vector updating.

5 Conclusion

We have proposed a new approach to competitive learning in SOMs. The opportunity to create a model to facilitate global optimization of the feature map requires methods to acquire the overall quality of the feature map. These methods take the form of measuring distance between clusters, cluster density and cluster size.

An additional competitive layer has been added to the traditional SOM model as well as modifying the original competition that results in the proposed GT-SOM algorithm. A similarity ranking within a user-defined threshold between neuron weight vectors and input vectors is used as a basis for the creation of payoff tables between neurons. Payoffs are calculated according to strategy set containing possible actions for each neuron. Each action results in a numeric utility or payoff which may improve or diminish SOM quality. Finding the neuron whose action maximizes the quality of the SOM for that iteration is now possible, enabling neurons to be picked not only on similarity but on strength. Clusters can be increased or decreased in size or density in order to attempt to reach a user-defined threshold for overall desired quality of the SOM. Future research will focus on training result analysis between the traditional SOFM training method and the proposed GTSOFM training algorithm.

References

1. Kohonen, T.: Automatic formation of topological maps of patterns in a self-organizing system. In: Proceedings of the Scandinavian Conference on Image Analysis. (1981) 214–220
2. Huntsberger, T., Ajjimarangsee, P.: Parallel self-organizing feature maps for unsupervised pattern recognition. International Journal of General Systems **16**(4) (1990) 357–372
3. Tsao, E., Lin, W., Chen, C.: Constraint satisfaction neural networks for image recognition. Pattern Recognition **26**(4) (1993) 553–567
4. Cho, S.B.: Ensemble of structure-adaptive self-organizing maps for high performance classification. Inf. Sci. **123**(1-2) (2000) 103–114
5. Hung, C., Wermter, S.: A dynamic adaptive self-organising hybrid model for text clustering. In: Proceedings of the Third IEEE International Conference on Data Mining. (2003) 75–82
6. Hagan, M.T., Demuth, H.B., Beale, M.H. In: Neural Network Design. PWS Publishing Company, Boston (1996)
7. Brachman, R.J., Anand, T.: The process of knowledge discovery in databases: A human-centered approach. In: Advances in knowledge discovery and data mining, American Association for Artificial Intelligence (1996) 37–58
8. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery: an overview. In: Advances in knowledge discovery and data mining, American Association for Artificial Intelligence (1996) 1–34
9. Chandrasekaran, V., Liu, Z.Q.: Topology constraint free fuzzy gated neural networks for pattern recognition. IEEE Transactions on Neural Networks **9**(3) (1998) 483–502
10. Pal, S., Dasgupta, B., Mitra, P.: Rough self organizing map. Applied Intelligence **21**(3) (2004) 289–299
11. Fritzke, B.: Some competitive learning methods. Technical report, Institute for Neural Computation. Ruhr-Universit at Bochum (1997)
12. Santini, S., Jain, R.: Similarity measures. IEEE Transactions: Pattern Analysis and Machine Intelligence **21**(9) (1999) 871–883
13. Kolen, J.F., Pollack, J.B.: Back propagation is sensitive to initial conditions. In: Advances in Neural Information Processing Systems. Volume 3. (1991) 860–867
14. von Neumann, J., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press, Princeton (1944)
15. Nash, J.: The bargaining problem. Econometrica **18**(2) (1950) 155–162
16. Roth, A.: The evolution of the labor market for medical interns and residents: a case study in game theory. Political Economy **92** (1984) 991–1016
17. Bell, M.: The use of game theory to measure the vulnerability of stochastic networks. IEEE Transactions on Reliability **52**(1) (2003) 63– 68
18. Fischer, J., Wright, R.N.: An application of game-theoretic techniques to cryptography. Discrete Mathematics and Theoretical Computer Science **13** (1993) 99–118
19. Gossner, O.: Repeated games played by cryptographically sophisticated players. Technical report, Catholique de Louvain - Center for Operations Research and Economics (1998)