Self-Organizing Swarm (SOSwarm) for Financial Credit-Risk Assessment

Abstract— **This paper applies a self-organizing Particle Swarm algorithm, *SOSwarm*, for the purposes of credit-risk assessment. SoSwarm can be applied for unsupervised clustering and for classification. In the algorithm, input vectors are projected into a lower dimensional map space producing a visual representation of the input data in a manner similar to a self-organizing map (SOM). However, unlike SOM, the nodes (particles) in this map react to input data during the learning process by modifying their velocities using an adaptation of the Particle Swarm Optimization velocity update step. The utility of SoSwarm is tested by applying it to two important credit-risk assessment problems drawn from the domain of finance, namely the prediction of corporate bond ratings and the prediction of corporate failure. The results obtained on the financial benchmark problems are highly-competitive against those of traditional classification methodologies. The paper makes a further contribution showing that the canonical SOM can be explored within the PSO paradigm. This highlights an important linkage between the heretofore distinct literatures of SOM and PSO.**

I. Introduction

Clustering is a commonly encountered scenario in many data-mining applications, the objective being to uncover a structure in a collection of unlabeled data. Real-world applications of clustering include the mining of customer databases, fraud detection, data compression and image analysis. Over the years, a wide variety of algorithms have been developed for clustering purposes, including K-means, Fuzzy C-means, Hierarchical clustering and Mixture of Gaussians. In addition to these traditional clustering algorithms, several biologically-inspired clustering algorithms have been developed. These are derived from a variety of sources of inspiration including evolutionary processes (genetic algorithm clustering applications include [1], [2], [3], [4], genetic programming clustering applications include [5], [6]) and social systems (examples of ant-based clustering include [7], [8], [9]). Perhaps the best-known family of biologically-inspired clustering algorithms is the self-organising map (SOM). Since the development of SOMs [10], [11] a considerable literature has developed in this field. A particular feature of SOMs is that they typically reduce multi-dimensional data to a low dimensional map (or grid) of nodes, making them a useful tool for data visualization [12].

A recent addition to the family of natural computing algorithms are particle swarm algorithms [13]. Thus far, these algorithms have been primarily applied for either optimization or social modeling purposes [14], [15]. However, it seems natural to extend the Particle Swarm Algorithm (PSA) to perform automated clustering tasks due to the manner in which particles of a swarm cluster to similar regions over time in a PSA. In [16] we introduced the Self-Organizing Swarm (SOSwarm) algorithm which adopts unsupervised learning using a PSA framework. We extend this work in this paper by applying this algorithm to two real-world problems drawn from the finance domain. The paper also illustrates an important linkage between the SOM and PSO literatures.

II. Particle Swarm Optimization

The PSO algorithm was introduced by Kennedy and Eberhart [13] and is described in detail in [14]. In the context of PSO a swarm can be defined as '... a population of interacting elements that is able to optimize some global objective through collaborative search of a space' [14]. The nature of the interacting elements (particles) depends on the problem domain. Typically, the particles are real-valued vectors. These particles move (fly) in an n-dimensional search space, in an attempt to uncover ever-better solutions to the problem of interest.

Each of the particles has two associated properties, a current position and a velocity. Each particle also has a memory of the best location in the search space that it has found so far (p_{best}), and knows the best location found to date by all the particles in the population (g_{best}) or in an alternative version of the algorithm, a local neighborhood around each particle (l_{best}). In the local version of the algorithm, each particle is considered to be linked to a subset of the population of particles, and this linkage structure is fixed at the beginning of the optimization process and remains unchanged during it (see Fig.1). Although a subset of the particles are defined as being 'linked' this does not imply that the particles will be spatially proximate as the algorithm runs. It is quite possible, particularly in the early iterations of the algorithm, that the particles could be a considerable distance from each other.

Whether the local or global communication version is implemented, at each step of the algorithm, particles are displaced from their current position by applying a velocity (or gradient) vector to them. The velocity size / direction is influenced by the velocity in the previous iteration of the algorithm (simulates 'momentum'), and the location of a particle relative to its p_{best} and g_{best} (or l_{best}). Therefore, at each step, the size and direction of each particle's move is a function of its own history (experience), and the social influence of its peer group.

A number of variants of the particle swarm algorithm (PSA) exist. The following paragraphs provide a description of a canonical continuous version of the algorithm.

i. Initialize each particle in the population by randomly selecting values for its location and velocity vectors.
ii. Calculate the fitness value of each particle. If the current fitness value for a particle is greater than the best fitness value found for the particle so far, then revise p_{best}.
iii. Determine the location of the particle with the highest fitness and revise g_{best} if necessary.
iv. For each particle, calculate its velocity according to equation 1.
v. Update the location of each particle according to equation 3.
vi. Repeat steps ii - v until stopping criteria are met.

The update algorithm for particle i's velocity vector v_i is:

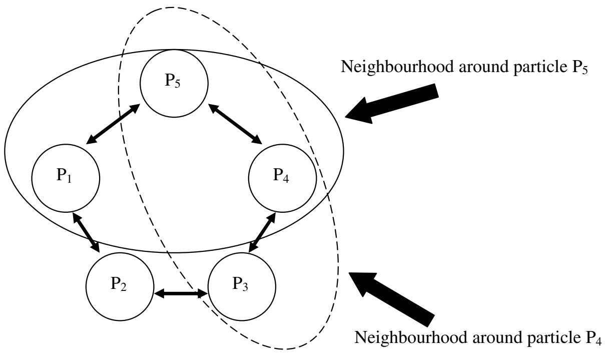
Neighbourhood around particle P₅
P₅
P₁
P₄
P₂
P₃
Neighbourhood around particle P₄

Fig. 1. Ring topology where l_{best} is defined using a 3 node neighborhood. This corresponds to a gbest

$$\begin{aligned} v_i(t+1) = w * v_i(t) + (c_1 * R_1 * (p_{best} - x_i)) \\ + (c_2 * R_2 * (g_{best} - x_i)) \end{aligned} \quad (1)$$

where,

$$w = wmax - ((wmax - wmin)/itermax) * iter \quad (2)$$

In equation 1, p_{best} is the location of the best solution found to-date by particle i, g_{best} is the location of the global-best solution found by all particles to date, c_1 and c_2 are the weights associated with the p_{best} and the g_{best} terms in the velocity update equation, x_i is particle i's current location, and R_1 and R_2 are randomly drawn from U(0,1). The parameter w represents a momentum coefficient which is reduced according to equation 2 as the algorithm iterates. In equation 2, $itermax$ and $iter$ are the total number of iterations the algorithm will run for, and the current iteration value respectively. The parameters $wmax$ and $wmin$ set the upper and lower boundaries on the value of the momentum coefficient. The velocity update on any dimension is constrained to a maximum value of $vmax$. Once the velocity update for particle i is determined, its position is updated (equation 3), and p_{best} is updated if necessary (equations 4 & 5).

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (3)$$

$$p_{best\ i}(t+1) = p_{best\ i}(t) \text{ if, } f(x_i(t)) \leq f(y_i(t)) \quad (4)$$

$$p_{best\ i}(t+1) = x_i(t) \text{ if, } f(x_i(t)) > f(y_i(t)) \quad (5)$$

After the location of all particles have been updated, a check is made to determine whether g_{best} needs to be updated (equation 6).

$$\hat{y} \in (y_0, \ldots, y_n) | f(\hat{y}) = \max(f(y_0), \ldots, f(y_n)) \quad (6)$$

$$g_{best}(t+1) = g_{best}(t) \text{ if, } f(g_{best}(t)) > f(\hat{y}(t)) \quad (7)$$

$$g_{best}(t+1) = \hat{y}(t) \text{ if, } f(g_{best}(t)) \leq f(\hat{y}(t)) \quad (8)$$

A. Elements of PSA

In each iteration of the algorithm, a particle is stochastically accelerated towards its previous best position and towards a global (or neighborhood) best position, thereby forcing particles to continually search in the most-promising regions found so far in the solution space. The weight coefficients c_1 and c_2 control the relative impact of the p_{best} and g_{best} locations on the velocity of a particle. Low values for c_1 and c_2 allow each particle to explore far away from already uncovered good points (there is less emphasis on past learning), high values of the parameters encourage more intensive search of regions close to these points. The random coefficients r_1 and r_2 ensure that the algorithm is stochastic. A practical effect of r_1 and r_2, is that neither the individual nor the social learning terms are always dominant.

The neighborhood structure plays a critical role in determining the nature of the communication between particles during the search process. If the neighborhood is set at 1 (each particle only communicates with itself), then each particle searches independently of all other particles. If the neighborhood= N (the number of particles in the swarm), all particles can communicate with each other, and we have the g_{best} version of the PSO algorithm.

B. Sample Applications of PSA

Particle Swarm algorithms have been successfully applied to a diverse range of problems including Financial Modeling [28], the automatic generation of programs ([17], [18]) using a grammatical representation borrowed from Grammatical Evolution [19], [20], [21], [22], [23], and the construction of Artificial Neural Networks [24].

III. Self-Organizing Map

Self-organizing maps (SOM) [10], [11], [12] are loosely inspired by the self-organizing capability of neurons in the cortex. Experimental evidence has shown that certain parts of the brain perform specific tasks such as processing touch, sound and visual stimuli. In these regions, neurons spatially self-organize, or cluster, depending on their function. Inspired by these processes of self-organization, SOMs are artificial neural nets which use unsupervised learning to adapt (organize) themselves in response to signal inputs.

A SOM acts to project (compress) input data vectors onto a low-dimensional space, typically a two-dimensional grid structure, thereby producing a visual representation of the input data. The unsupervised learning process is based on measures of similarity amongst the input data vectors. During the training process, the network undergoes self-organization as like input data patterns are grouped or clustered together on the grid structure. SOMs have been utilized for a variety of clustering and classification problems including speech recognition and medical diagnosis [29]. The SOM bears similarities with the traditional statistical technique of Principal Component Analysis (PCA). However, unlike PCA the projection of the input data is not necessarily restricted to be linear.

The SOM consists of two layers, the input layer (a holding point for the input data), and the *mapping* layer. The input layer has as many nodes as there are input variables. The two layers are fully connected to each other and each of the nodes in the hidden layer has an associated weight vector, with one weight for each connection with the input layer.

The aim of the SOM is to group like input data-vectors together on the mapping layer, therefore the method is topology preserving as items which are close in the input space are also close in the mapping space. During training the data vectors are presented to the SOM through the input layer one at a time. The nodes in the mapping layer *compete* for the input data vector. The winner is the mapping node whose vector of incoming connection weights most closely resembles the components of the input data vector. The winner has the values of its weight vector adjusted to move them towards the values of the input data vector, and the mapping layer nodes in the neighborhood of the winning node also have their weight vectors altered to become more like the input data vector (a form of co-operation between the neighboring nodes). As more input data vectors are passed through the network, the weight vectors of the mapping layer nodes self-organize. The self-organization process also encourages the mapping layer weight vectors to congregate to regions of the input space where the training data is concentrated, with relatively few (if any) weight vectors being located in sparsely populated regions of the input space. The self-organizing map therefore tends to approximate the probability density function of the input data. Fig. 3 provides a stylized illustration of a trained SOM for the three data clusters in Fig. 2. The twelve (in this example) mapping layer nodes have self-organized so that the weight vectors for four nodes have moved towards each cluster of data in the feature space.

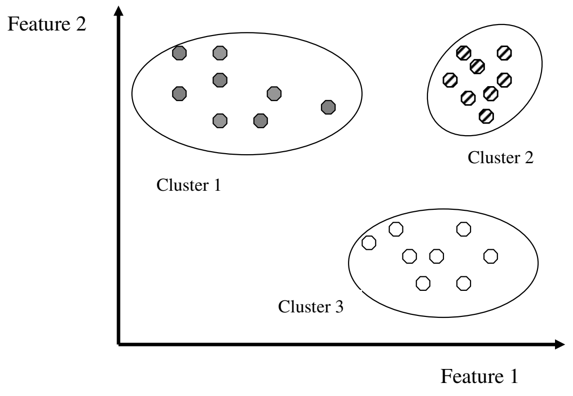

Fig. 2. Example where data splits neatly into three clusters, based on the two input features of each item

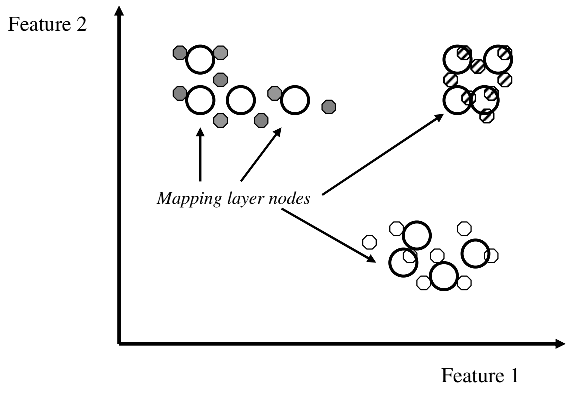

Fig. 3. A stylized illustration of a trained net. Four mapping nodes have migrated to each data cluster during the training process

By the end of the training process, different parts of the mapping layer respond strongly to specific regions of input space. Once training of the network is complete, the clusters obtained can be examined in order to gain better insight into the underlying dataset. Questions which can be addresses include: what input items have been grouped together, and what are the typical values for each input in a specific cluster?

A. PSO-SOM Hybrids

There are several ways that a PSO-SOM hybrid could be constructed. In [24] and [25] a PSO algorithm was used to refine the weight vectors for a SOM obtained after an initial application of a standard SOM training methodology. In this approach each particle consisted of a complete set of weights for the SOM and the object was to improve the initial clustering result by applying PSO to the population of weight vectors. The approach in our study differs fundamentally from the above and is outlined in the next section.

IV. Self-Organizing Swarm

The Self-Organizing Swarm (SOSwarm) operates in a similar fashion to a SOM with the adoption of a 2-d mapping layer. The components of this layer are considered as 'particles'. Instead of simply adjusting node weights in the map space with respect to the training input vectors, the particles in the mapping layer adjust their values using an adapted form of a PSA velocity update function.

The canonical form of the PSA update embeds two key elements:

- a history, and
- a social influence.

History is embedded in the PSA via the momentum term and the p_{best} components of the velocity update equation. The social influence is embedded via the influence of either g_{best} or l_{best} in the velocity update (g_{best} also embeds a swarm 'history'). In this study, we simplify the update equation by only using the momentum term to embed a particle's history. We initially also omit the social influence term but include a form of social communication using a neighborhood as described below. Section VI provides a further discussion on this point. In undertaking our experiments we apply the output from an unsupervised SOSwarm learning process for classification purposes. An outline of the SOSwarm algorithm used for this purpose is presented below.

```
0   initialize location and velocity of particles in mapping layer
randomly

1   for( max number of iterations )

2      for( each input training vector in turn )

3           set gbest to be the input vector

4           set pbest of each particle at its
            current position

5           find particle with closest match to gbest

6           denote this particle as the firing particle

7           update firing particle's velocity
            and position vectors

8      endfor

9   endfor

10  assign class to each particle using training data

11  calculate classification accuracy using test data
```

In order to determine the firing particle (the particle that is the closest match to an input vector) a simple distance calculation is adopted

$$Firing\, particle = \underset{i}{argmin} \; \|V - P_i\| \qquad (9)$$

where V corresponds to the input vector, P_i is the i^{th} particle's position vector, i is the number of particles in the swarm. A number of alternative distance functions could be adopted and we adopt Euclidean distance as outlined in equation 10 where d is the dimensionality of the vector or particle.

$$Firing\, particle = \underset{i}{argmin} \; \left\| \sqrt{\sum_{1}^{d} (V_d - P_{id})^2} \right\| \qquad (10)$$

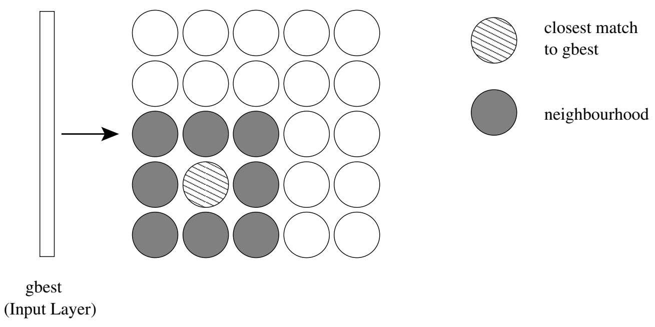

Fig. 4. A Self-Organizing Swarm (SOSwarm) with a 2-d mapping layer.

A visual representation of SOSwarm is presented in Fig. 4 with the adoption of a 2-d mapping layer. The particles are arranged *a priori* into a fixed neighborhood topology, a simple grid in this example. The firing particle, that is, the particle whose position vector is closest to the current input vector (designated as g_{best}) updates its position vector in a particle swarm style according to the PSA velocity and position update equations. In addition, particles lying within a fixed neighborhood of the firing particle also adjust their position vectors using the same equations, implicitly embedding a form of social communication between neighboring particles.

Once the map has been trained, each node in the mapping layer is assigned a class label using the training data, based on a simple majority voting scheme. In calculating the in sample and out sample classification accuracy, the distance between each input data vector and each mapping node is calculated, with the input data vector being assigned the class label of the mapping node it is closest to.

V. Experimental Setup & Results

In order to assess the utility of the SOSwarm algorithm, two credit risk assessment classification problems are examined.

A. Problems Examined

Classification is a commonly encountered decision scenario in business. Examples include decisions as to whether or not to invest in a firm, to extend credit to a new customer, or to extend a bank loan. In each of these scenarios, the possibility of financial loss exists if a firm is incorrectly classified as being financially healthy. We select two credit risk assessment scenarios, the assignment of a bond rating to a firm based on publicly-available financial data and the prediction of corporate failure, also using publicly-available financial data.

1) Bond Rating: When a company wants to issue traded debt (bonds), it must obtain a credit rating for the issue from at least one recognized rating agency (Standard & Poor's (S&P), Moody's, Fitches' or Dominion Bond Rating Service). The credit rating represents the rating agency's opinion at a specific date of the creditworthiness of a borrower in general (an *issuer credit rating*), or in respect of a specific debt issue (a *bond credit rating*). These ratings impact on the borrowing cost, and the marketability of issued bonds. Most rated debt is publicly tradable on stock markets, and bond ratings are typically changed infrequently. An accurate bond-rating prediction model could indicate whether the current rating of a bond is still justified. To the extent that an individual investor could predict a bond re-rating before other investors foresee it, this may provide a trading edge.

The dataset consists of financial data drawn from the financial statements of 791 non-financial US companies, along with their associated S&P bond-issuer credit-rating. In this case, we restrict attention to discriminating between investment grade vs junk grade ratings. In the dataset 57% of companies have an investment-grade rating (AAA, AA, A or BBB), and 43% have a junk rating. To allow time for the preparation of year-end financial statements, the filing of these statements with the Securities and Exchange Commission (SEC), and the development of a bond rating opinion by Standard & Poor's rating agency, the bond rating of the company as at 30 April 2000, is matched with financial information drawn from their financial statements as at 31 December 1999.

A subset of 600 firms was randomly sampled from the total of 791 firms to produce two groups of 300 investment grade and 300 junk rated firms. The 600 firms were randomly allocated to the training set (420) or the hold-out sample (180), ensuring that each set was equally balanced between investment and non-investment grade ratings.

A total of eight financial variables were selected for inclusion in this study. The selection of these variables was guided both by prior literature in bankruptcy prediction, literature on bond rating prediction, and by preliminary statistical analysis. The financial ratios chosen during the selection process were:

i. Current ratio
ii. Retained earnings to total assets
iii. Interest coverage
iv. Debt ratio
v. Net margin
vi. Market to book value
vii. Total assets
viii. Return on total assets

2) Corporate Failure: A sample of 178 (89 failed and 89 non-failed) publicly quoted US firms was drawn from the period 1991 to 2000 in order to train and test the classifier. Only firms with sales exceeding $1M, which had existed for at least three years prior to entry into Chapter 7 or Chapter 11 and which were outside the financial sector were considered for inclusion in the sample. Failed and non-failed firms were matched both by industry sector and size (sales revenue three years prior to failure). The set of 178 matched firms was randomly divided into model building (128 firms) and out-of-sample (50 firms) datasets. The dependent variable is binary (0,1), representing either a non-failed or a failed firm. Prior to the selection of the potential explanatory variables for inclusion in this study, a total of ten previous studies were examined [30], [31], [32], [33], [34], [35], [36], [37], [38], [39]. These studies employed a total of 58 distinct ratios. A subset of 22 of the most commonly used financial ratios was selected for this study. The selected ratios were:

i. EBIT/Sales
ii. EBITDA/Sales
iii. EBIT/Total Assets
iv. Gross Profit/Sales
v. Net Income/Total Assets
vi. Net Income/Sales
vii. Return on Assets
viii. Return on Equity
ix. Return on Investment
x. Cash/Sales
xi. Sales/Total Assets
xii. Inventory/Cost of Goods Sold
xiii. Inventory/Working Capital
xiv. Fixed Assets/Total Assets
xv. Retained Earnings/Total Assets
xvi. Cash from Operators/Sales
xvii. Cash from Operations/Total Liabilities
xviii. Working Capital/Total Assets
xix. Quick Assets/Total Assets
xx. Total Liabilities/Total Assets
xxi. Leverage
xxii. EBIT/Interest

In the corporate failure case, we construct three distinct classifiers which predict impending failure (or not) one to three years in advance of actual failure. Each of these models is designated T-1, T-2 and T-3 respectively. More details on each of these problem domains can be found in [26], [27], [28].

B. Methodology

Initially, the SOSwarm clustering algorithm is applied and then the nodes on the resulting map are labeled. The labeled nodes are then used to classify the data. We then report the classification accuracies on each dataset.

The following parameters were used for the SoSwarm algorithm, $c_1 = 1.0$, $c_2 = 2.0$, $wmax = 0.9$, $wmin = 0.4$, $cmin = 0$, $cmax = 1$, and $vmax = cmax$. The population of particles was set at 100 (a $10 * 10$ grid structure). The algorithm was run for a total of 10,000 iterations. The parameter values were set after a number of initial trial and error experiments. As the mapping process utilizes a distance metric, the input variables in each dataset were normalized independently in each dimension into the range $[0 \rightarrow 1]$.

The distance metric in equation 7 is used to determine the particle that is the closest match, and a fixed grid neighborhood topology is adopted, with the range of the neighborhood as illustrated in Fig. 4. That is, for particles not on the edges of the grid a particle will have at most 8 neighbors, which will be subjected to updates if that particle fires.

The bond dataset was recut 5 times between train and test data and the corporate failure dataset was recut 10 times. Thirty independent runs of the SOSwarm algorithm were conducted on each recut. The classification results obtained for the unseen test data are presented in Table I. The results reported consist of the best, the mean best and the mean average accuracy, obtained across the thirty runs, averaged over all data recuts. The results are encouraging with SOSwarm producing a competitive performance against the best previous classification accuracies reported on these problems.

In previous studies, applying a variety of classification methods to the bond dataset over the same five recuts, best results of 82.74% (85.22) in-sample (out-sample) were obtained using linear discriminant analysis averaged across all five recuts. Applying grammatical evolution to the same dataset produced best results of 86.78% (86.26) in-sample (out-sample) [28].

In the case of the corporate failure data, best results of 81.3% (78) in-sample (out-sample) were obtained using linear discriminant analysis averaged across all ten recuts for T-1 data. The corresponding results for T-2 and T-3 were 76.6% (58) and 75% (58) respectively. Applying grammatical evolution to the same dataset produced best results of 85.9% (80) in-sample (out-sample) [28]. The corresponding results for T-2 and T-3 were 82.8% (80) and 75.8% (70) respectively. Hence, it can be seen that SoSwarm has produced competitive results when compared with those of other classification techniques.

TABLE I

CLASSIFICATION ACCURACY OBTAINED ON THE OUT-OF-SAMPLE DATA FOR THE SELF-ORGANIZING SWARM ALGORITHM ACROSS THE BENCHMARK PROBLEMS AVERAGED ACROSS ALL THE RECUTS OF THE DATASET IN EACH CASE.

Problem	**best (%)**	**mean best (%)**	**av.mean (std.dev.)**
Bond rating	86.66	86.22	81.62 (2.30)
Corporate failure T-1	77.14	71.42	60.44 (5.69)
Corporate failure T-2	74.28	69.14	57.09 (5.72)
Corporate failure T-3	70.57	70.57	57.85 (6.28)

VI. SOSWARM AND SOM

Although, as already noted in section III-A, a number of studies have combined PSO and SOM methodologies, significantly however, no previous study has examined the deeper linkages between the two methodologies. The teasing out of such linkages is important as both paradigms are well-developed and are widely used. The drawing of parallels between both paradigms opens up a door for useful cross-fertilization between each.

In the canonical SOM, the update of a firing node's weight vector is governed by:

$$x_i(t+1) = \eta(t)h(t)(x_i(t) - \beta) \tag{11}$$

where x_i is the firing node's weight vector, η is the time-varying learning rate and β is the input vector. Hence, after firing, the weight vector of the relevant node in the mapping layer, and those of its neighbors which are defined by the neighborhood function $h(t)$, are adjusted in order to more closely resemble the input vector. Ignoring the update of neighboring nodes, and thinking of eq. 11 in particle swarm terms, it is apparent that it can be written as a velocity update:

$$v_i(t+1) = \eta(t)(x_i(t) - \beta) \tag{12}$$

Of course, the component $\eta(t)$ in eq. 12, in effect a weighting term, is similar in concept to the weight parameter c_1 in eq. 1. Hence, the canonical SOM update equation can be closely approximated by a reduced (non-momentum) version of the canonical PSA update equation.

This parallel between the SOM and a reduced form PSA suggests multiple possibilities for the creation of new hybrid algorithms for self-organization. For example, the PSA embeds momentum, a form of personal particle history which is not included in the canonical SOM. Like the learning rate in the SOM, the momentum term in the PSA velocity update is time-varying, and it reduces over time in order to encourage particle convergence. The SOSwarm algorithm described in sect. IV includes momentum.

Another interesting possibility, drawing on the use of peer learning in the PSA, is the incorporation of an additional 'peer-learning' term into the SoSwarm velocity update. For example, a topology consisting of a series of small overlapping neighborhoods could be defined between the particles before the algorithm started, with eq. 1 being extended by the addition of the term $c_3 * r_3 * (y_{local} - x_i(t))$, where y_{local} is the location of a randomly selected neighboring particle of $x_i(t)$. This social learning would tend to lessen the disruptive impact of an anomalous input vector during the learning process. This could prove especially useful in environments where training data is noisy or errorful.

VII. CONCLUSIONS & FUTURE WORK

This paper describes the Self-Organizing Swarm algorithm and illustrates its utility by applying it to two credit-risk assessment problems. Classification accuracies reveal that SOSwarm produces competitive results on the problems analyzed when compared with previous benchmark results on the same datasets. The paper also highlights an interesting linkage between PSO and SOM.

There are several interesting avenues of future research. A variety of distance metrics could be used in calculating the distance between input vectors and each member of the swarm. In the implementation of SoSwarm in this study, we utilized a simple distance metric, but several other metrics could be applied. Another interesting avenue is to investigate the effect of implementing differing neighborhood topologies between the particles in the swarm. It would also be interesting to examine in what circumstances a reducing size of neighborhood over the course of the algorithm would be beneficial. Other possible extensions of the study include the investigation of different swarm sizes and different velocity update formulations. Although we have applied SOSwarm for classification purposes in this study, it could clearly also be applied for clustering purposes, opening up such potential applications as gene clustering, and customer database segmentation. It would also be interesting to explore the utility of the SOSwarm algorithm for such applications.

References

[1] Franti, P., Kivijarvi, J., Kaukoranta, T., Nevalainen, O. Genetic Algorithms for Large Scale Clustering Problems, *The Computer Journal*, 40:547-554, 1997.

[2] Maulik, U. and Bandyopadhyay, S. Genetic algorithm-based clustering technique, *Pattern Recognition*, 33:1455-1465, 2000.

[3] Tseng, L. and Yang, S. A genetic approach to the automatic clustering problem, *Pattern Recognition*, 34:415-424, 2001.

[4] Garai, G. Chaudhuri, B. A novel genetic algorithm for automatic clustering, *Pattern Recognition Letters*, 25(2):173-187, 2004.

[5] De Falco, I., Tarantino, E., Delia Cioppa A. and Gagliardi, F. A novel grammar-based genetic programming approach to clustering, in *Proceedings of the 2005 ACM symposium on Applied computing*, Santa Fe, New Mexico, pp 928-932, 2005.

[6] De Falco, I., Tarantino, E., Delia Cioppa, A. and Fontanella, F. An Innovative Approach to Genetic Programming based Clustering, *Advances in Soft Computing*, 55-64, 2006.

[7] Deneubourg. J., Gross, S., Franks, N., Sendova-Franks, A., Detrain, C. and Chretien, L. The dynamics of collective sorting robot-like ants and ant-like robots, *Proceedings of 1st Conference on Simulation of Adaptive Behavior: From Animals to Animats (SAB 90)*, in Meyer, J. and Wilson, S. (eds), MIT Press, Cambridge, MA, USA, pp 356-365, 1991.

[8] Lumer, E. and Faieta, B. Diversity and adaptation in populations of clustering ants, *Proceedings of Third International Conference on Simulation of Adaptive Behaviour*, pp 501-508, 1994.

[9] Bonabeau, E., Dorigo, M. and Theraulaz, G. *Swarm Intelligence: From natural to artificial systems*, Oxford: Oxford University Press, 1999.

[10] Kohonen, T. Self-organized formation of topologically correct feature maps, *Biological Cybernetics*, 43:59-69, 1982.

[11] Kohonen, T. The Self-Organizing Map, *Proceedings of the IEEE*, 78(9):1464-1480, 1990.

[12] Kohonen, T. The SOM Methodology, *in Visual Explorations in Finance with self-organizing maps*, edited by Deboeck, G. and Kohonen, T., p. 159-167, Berlin: Springer-Verlag, 1998.

[13] Kennedy, J. and Eberhart, R. Particle swarm optimization, *Proceedings of the IEEE International Conference on Neural Networks*, December 1995, pp. 1942-1948, IEEE Press, 1995.

[14] Kennedy, J., Eberhart, R. and Shi, Y. *Swarm Intelligence*, San Mateo, California: Morgan Kauffman, 2001.

[15] Brabazon, A., Silva, A., Ferra de Sousa, T., O'Neill, M., Matthews, R. and Costa, E. Investigating strategic inertia using OrgSwarm, *Informatica*, 29(2):125-141, 2005.

[16] O'Neill, M. and Brabazon, A. Self-Organizing Swarm (SoSwarm): A Particle Swarm Algorithm for Unsupervised Learning, in *Proceedings of the Congress on Evolutionary Computation (CEC 2006)*, pp. 2649-2654, IEEE Press: New Jersey, 2006.

[17] O'Neill, M., Brabazon, A. Grammatical Swarm, in LNCS 3102 *Proc. of the Genetic and Evolutionary Computation Conference GECCO 2004*, Seattle, WA, USA, pp. 163-174, Springer, 2004.

[18] O'Neill, M., Brabazon, A., Adley, C. The automatic generation of programs for Classification using Grammatical Swarm, in *Proc. of the Congress on Evolutionary Computation CEC 2004*, Portland, OR, USA, pp. 104-110, IEEE Press, 2004.

[19] O'Neill, M., Ryan, C. *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*, Kluwer Academic Publishers, 2003.

[20] O'Neill, M. *Automatic Programming in an Arbitrary Language: Evolving Programs in Grammatical Evolution*, PhD thesis, University of Limerick, 2001.

[21] O'Neill, M., Ryan, C. Grammatical Evolution, *IEEE Trans. Evolutionary Computation*, 5(4):349-358, 2001.

[22] O'Neill, M., Ryan, C., Keijzer M., Cattolico M. Crossover in Grammatical Evolution. *Genetic Programming and Evolvable Machines*, 4(1):67-93, 2003.

[23] Ryan, C., Collins, J.J., O'Neill, M. Grammatical Evolution: Evolving Programs for an Arbitrary Language, in *Proc. of the First European Workshop on GP*, pp. 83-95, Berlin: Springer-Verlag, 1998.

[24] Xiao, X., Dow, E.R., Eberhart, R., Miled, Z.B., Oppelt, R.J. A hybrid self-organizing maps and particle swarm optimization approach. *Concurrency and Computation: Practice and Experience*, 16(9):895-915, 2004.

[25] Xiao, X., Dow, E.R., Eberhart, R., Miled, Z.B., Oppelt, R.J. Gene-Clustering Using Self-Organizing Maps and Particle Swarm Optimization, in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 22-26 April 2003, Nice, France, IEEE Press, 2003.

[26] Brabazon, A. and Keenan, P. A hybrid genetic model for the prediction of corporate failure, *Computational Management Science*, 1(3-4):293-310, 2004.

[27] Brabazon, A. and O'Neill, M. Credit Classification Using Grammatical Evolution, *Informatica*, 30(3):325-335, 2006.

[28] Brabazon, A. and O'Neill, M. *Biologically Inspired Algorithms for Financial Modelling*, Berlin: Springer, 2006.

[29] Gurney, K. *An introduction to neural networks*, University College London Press, London, 1997.

[30] Altman, E. Financial ratios, discriminant analysis and the prediction of corporate bankruptcy, *Journal of Finance*, 23:589-609, 1968.

[31] Altman, E., Haldeman, R. and Narayanan, P. ZETA analysis: A new model to identify bankruptcy risk of corporations, *Journal of Banking and Finance*,1:29-54, 1997.

[32] Back, B., Laitinen, T., Sere, K. and van Wezel, M. Chosing Bankruptcy Predictors Using Discriminant Analysis, Logit Analysis and Genetic Algorithms, *Technical Report No. 40, Turku Centre for Computer Science*, Turku School of Economics and Business Administration, 1996.

[33] Beaver, W. Financial ratios as predictors of failure, *Journal of Accounting Research - Supplement: Empirical Research in Accounting*, 71-102, 1996.

[34] Dambolena, I. and Khoury, S. Ratio stability and corporate failure, *Journal of Finance*, 35(4):1017-1026, 1980.

[35] Kahya, E. and Theodossiou, P. Predicting corporate financial distress: A time-Series CUSUM methodology, *Review of Quantitative Finance and Accounting*, 13:71-93, 1996.

[36] Moody's RiskCalc For Private Companies: Moody's Default Model, `http://www.riskcalc.moodysrms.com`, 2000.

[37] Ohlson, J. Financial ratios and the probabilistic prediction of bankruptcy, *Journal of Accounting Research*, 18:109-131, 1980.

[38] Serrano-Cina, C. Self organizing neural networks for financial diagnosis, *Decision Support Systems*, 17(3):227-238, 1996.

[39] Sung, T., Chang, N. and Lee, G. Dynamics of modelling in data nining: interpretative approach to bankruptcy prediction, *Journal of Management Information Systems*, 16(1):63-85, 1999.