



module by the  $F_0^a$  (and  $F_0^b$  respectively) layer in order to avoid proliferation of nodes. Each input vector  $\mathbf{a}$  produces the normalized vector  $\mathbf{A} = (\mathbf{a}, \mathbf{1} - \mathbf{a})$  whose  $L_1$  norm is constant:  $|\mathbf{A}| = n$ .

Let  $M_a$  be the number of nodes in  $F_1^a$  and  $N_a$  be the number of nodes in  $F_2^a$ . Due to the preprocessing step,  $M_a = 2n$ .  $\mathbf{w}^a$  is the weight vector between  $F_1^a$  and  $F_2^a$ . Each  $F_2^a$  node represents a class of inputs grouped together, denoted as a ‘‘category’’. Each  $F_2^a$  category has its own set of adaptive weights stored in the form of a vector  $\mathbf{w}_j^a, j = 1, \dots, N_a$  whose geometrical interpretation is a hyper-rectangle inside the unit box. Similar notations and affirmations are valid for  $ART_b$ , that receives  $m$ -dimensional input vectors. For a classification problem, the class index is the same as the category number in  $F_2^b$ , thus  $ART_b$  can be simply substituted by an  $N_b$ -dimensional vector.

The Mapfield module allows FAM to perform heteroassociative tasks, establishing many-to-one links between various categories from  $ART_a$  and  $ART_b$ , respectively. The number of nodes in Mapfield is equal to the number of nodes in  $F_2^b$ . Each node  $j$  from  $F_2^a$  is linked to every node from  $F_2^b$  via a weight vector  $\mathbf{w}_j^{ab}$ .

The learning algorithm is sketched below. For every training pattern, the vigilance parameter factor is set equal to its baseline value, and all nodes are not inhibited. For each (preprocessed) input  $\mathbf{A}$ , a fuzzy choice function is used to get the response for each  $F_2^a$  category:

$$T_j(\mathbf{A}) = \frac{|\mathbf{A} \wedge \mathbf{w}_j^a|}{\alpha_a + |\mathbf{w}_j^a|}, \quad j = 1, \dots, N_a \quad (1)$$

Let  $J$  be the node with the highest value computed as in (1). If the resonance condition from eq. 2 is not fulfilled, then the  $J$ th node is inhibited such that it will not participate to further competitions for this pattern and a new search for a resonant category is performed. This might lead to creation of a new category in  $ART_a$ .

$$\rho(\mathbf{A}, \mathbf{w}_J^a) = \frac{|\mathbf{A} \wedge \mathbf{w}_J^a|}{|\mathbf{A}|} \geq \rho_a \quad (2)$$

A similar process occurs in  $ART_b$  and let  $K$  be the winning node from  $ART_b$ . The  $F_2^b$  output vector is set to:

$$y_k^b = \begin{cases} 1, & \text{if } k = K \\ 0, & \text{otherwise} \end{cases} \quad k = 1, \dots, N_b \quad (3)$$

An output vector  $\mathbf{x}^{ab}$  is formed in Mapfield:  $\mathbf{x}^{ab} = \mathbf{y}^b \wedge \mathbf{w}_J^{ab}$ . A Mapfield vigilance test controls the match between the predicted vector  $\mathbf{x}^{ab}$  and the target vector  $\mathbf{y}^b$ :

$$\frac{|\mathbf{x}^{ab}|}{|\mathbf{y}^b|} \geq \rho_{ab} \quad (4)$$

where  $\rho_{ab} \in [0, 1]$  is a Mapfield vigilance parameter. If the test from (4) is not passed, then a sequence of steps called *match tracking* is initiated (the vigilance parameter  $\rho_a$  is increased and a new resonant category will be sought

for  $ART_a$ ); otherwise learning occurs in  $ART_a$ ,  $ART_b$  and Mapfield:

$$\mathbf{w}_J^{a(new)} = \beta_a (\mathbf{A} \wedge \mathbf{w}_J^{a(old)}) + (1 - \beta_a) \mathbf{w}_J^{a(old)} \quad (5)$$

(and the analogous in  $ART_b$ ) and  $\mathbf{w}_{Jk}^{ab} = \delta_{kK}$ , where  $\delta_{ij}$  is Kronecker’s delta.

### 3 FAMFW

The FAMFW is a FAM architecture with a generalized distance measure. Since this is the essential difference, we only describe here this modification and illustrate its effect on the categories created.

Like in [8], for a category  $\mathbf{w}_j$  we define its size  $s(\mathbf{w}_j)$ :

$$s(\mathbf{w}_j) = n - |\mathbf{w}_j| \quad (6)$$

and the distance to a normalized input  $\mathbf{A}$ :

$$dis(\mathbf{A}, \mathbf{w}_j) = |\mathbf{w}_j| - |\mathbf{A} \wedge \mathbf{w}_j| = \sum_{i=1}^n d_{ji} \quad (7)$$

where  $\mathbf{d}_j = \mathbf{w}_j - \mathbf{A} \wedge \mathbf{w}_j = (d_{j1}, \dots, d_{jn})$ . In [8] it is shown that:

$$T_j(\mathbf{A}) = \frac{n - s(\mathbf{w}_j) - dis(\mathbf{A}, \mathbf{w}_j)}{n - s(\mathbf{w}_j) + \alpha_a} \quad (8)$$

$$\rho(\mathbf{A}, \mathbf{w}_J^a) = \frac{n - s(\mathbf{w}_J) - dis(\mathbf{A}, \mathbf{w}_J)}{n} \quad (9)$$

Let us consider a weighted distance  $dis(\mathbf{A}, \mathbf{w}_j; \lambda)$ , a generalized form of the function  $dis(\mathbf{A}, \mathbf{w}_j)$ :

$$dis(\mathbf{A}, \mathbf{w}_j; \lambda) = \sum_{i=1}^n \lambda_i d_{ji} \quad (10)$$

where  $\lambda = (\lambda_1, \dots, \lambda_n)$ , and  $\lambda_i \in [0, n]$  is the weight associated to the  $i$ th feature. We impose the constraint  $|\lambda| = n$ . For  $\lambda_1 = \dots = \lambda_n = 1$ , we obtain the FAM as a particular case.

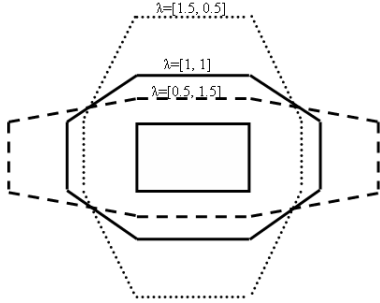
In [8], Charalampidis *et al.* used the following weighted distance:

$$dis(\mathbf{x}, \mathbf{w}_j | \lambda, ref) = \sum_{i=1}^n \frac{(1 - \lambda) l_j^{ref} + \lambda}{(1 - \lambda) l_{ji} + \lambda} d_{ji} \quad (11)$$

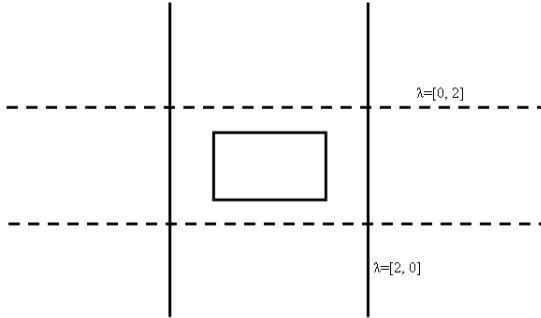
with  $l_j^{ref}$  a function of category  $j$ ’s lengths of the hyper-rectangle and  $\lambda$  a scalar in  $[0, 1]$ . In our case, the function  $dis(\mathbf{A}, \mathbf{w}_j; \lambda)$  does not depend on sides of the category created during learning, but on the weight computed for each feature. This makes our approach very different than the one in [8].

The effect of using the distance  $dis(\mathbf{A}, \mathbf{w}_j; \lambda)$  for a bidimensional category is depicted in Figure 1(a). The hexagonal shapes represent the points situated at constant distance from the category. These shapes are flattened in

the direction of the feature with a larger weight and elongated in the direction of the feature with a smaller weight. This is in accordance with the following intuition: The category dimension in the direction of a relevant feature should be smaller than the category dimension in the direction of a non-relevant feature. Hence, we may expect that more categories will cover the relevant directions than the non-relevant ones.



(a) Bounds for constant weighted distance  $dis(\mathbf{A}, \mathbf{w}_j; \lambda)$  for various values of vector  $\lambda$ . The rectangle from the middle represents a category.



(b) Bounds for constant distance  $dis(\mathbf{A}, \mathbf{w}_j; \lambda)$  for null feature weight. The rectangle in the middle represents the category.

Figure 1. Geometric interpretation of constant distance when using  $dis(\mathbf{A}, \mathbf{w}_j; \lambda)$  for bidimensional patterns

As shown in Figure 1(b) for the bidimensional case, when one uses null weights for a specific feature, the bounds are reduced to parallel lines on both sides of the rectangle representing the category. In this extreme case, the discriminative distance is the one along the remaining feature dimension. This is another major difference between our approach and the one in [8], where, while using function  $dis(\mathbf{x}, \mathbf{w}_j | \lambda, ref)$ , the contours of a constant weighted distance are inside some limiting hexagons. In our approach, the contour is insensitive to the actual value of the null weighted feature.

## 4 Features weights obtained with the ESRNG algorithm

In our experiments, we used the ESRNG procedure to determine the weights of the generalized distance measure in FAMFW. We will outline the principal steps of the ESRNG algorithm, introduced with details in [11]. ESRNG is an iterative algorithm that simultaneously adapts a set of reference vectors for as least as possible quantization error on all feature vectors and updates the input features relevances maximizing an informational energy measure. ESRNG has the following steps:

1. Update the reference vectors using the SRNG scheme.
2. Update the relevance factors.
3. Repeat steps 1 and 2 for all samples from the training set.

The SRNG algorithm uses a generalized Euclidean distance. Two reference vectors are updated at each step:  $\mathbf{w}_j$  and  $\mathbf{w}_k$ , which are the closest to the current input vector  $\mathbf{x}_i$  from the same class and from another class. The updating formulas for the two reference vectors can be found in [11].

The ESRNG algorithm generates numeric values assigned to each input feature, quantifying their importance in the classification task: the most relevant feature receives the highest numeric value. The formula that iteratively adapts the relevances is:

$$\lambda^{(t+1)} = \lambda^{(t)} - \alpha \frac{1}{4\sigma^2} G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2 \mathbf{I}) \cdot (\mathbf{y}_2 - \mathbf{y}_1) \mathbf{I}(\mathbf{x}_1 - \mathbf{w}_{j(1)} - \mathbf{x}_2 + \mathbf{w}_{j(2)}),$$

where  $\mathbf{w}_{j(1)}$  and  $\mathbf{w}_{j(2)}$  are the closest prototypes to  $\mathbf{x}_1$  and  $\mathbf{x}_2$ , respectively, two consecutive training vectors from different classes.  $G$  is the multidimensional Gaussian kernel used in the Parzen windows approximation of the continuous probabilities densities. We have used these relevance factors as feature weights in the FAMFW algorithm.

## 5 Experiments

We illustrate the FAMFW behavior on artificial and real datasets. First, we used an artificial dataset: points were distributed inside of two circles. Second, we used Iris dataset, which has 150 patterns in three classes, two of them being overlapped.

### 5.1 Artificial dataset: two circles

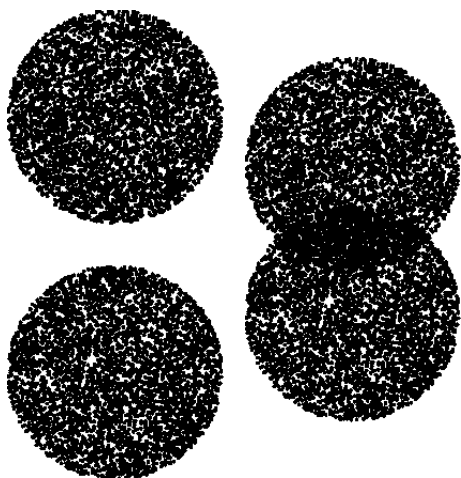
Two datasets were generated: the first represents a pair of non-overlapping circles. The second corresponds to a pair of partially overlapping circles. In both cases, the circles are vertically stacked (Figures 2(a) and 2(b)) and each of them is considered an output class.

Table 1. Number of  $ART_a$  categories and PCC for non-overlapping circles and various values of  $\lambda$ .

Test no.	$\lambda = [1, 1]$		$\lambda = [0.8, 1.2]$		$\lambda = [0.5, 1.5]$		$\lambda = [0, 2]$	
	No. of $ART_a$ categories	PCC	No. of $ART_a$ categories	PCC	No. of $ART_a$ categories	PCC	No. of $ART_a$ categories	PCC
1	6	98.14%	6	99.17%	5	99.87%	7	99.95%
2	6	99.25%	6	99.3%	7	99.53%	7	99.7%
3	6	98.92%	5	100%	5	100%	5	100%
4	6	100%	6	100%	6	100%	8	100%
5	6	99.21%	6	99.35%	6	99.65%	5	100%
Avg.	6	99.104%	5.8	99.564%	5.8	99.81%	6.4	99.93%

Table 2. Number of  $ART_a$  categories and PCC for overlapping circles and various values of  $\lambda$ .

Test no.	$\lambda = [1, 1]$		$\lambda = [0.8, 1.2]$		$\lambda = [0.5, 1.5]$		$\lambda = [0, 2]$	
	No. of $ART_a$ categories	PCC	No. of $ART_a$ categories	PCC	No. of $ART_a$ categories	PCC	No. of $ART_a$ categories	PCC
1	8	84.08%	7	88.77%	9	87.59%	7	90.14%
2	6	79.75%	8	80.98%	8	81.58%	8	88.12%
3	7	87.45%	7	88.36%	7	88.91%	6	89.52%
4	8	88.59%	8	89.41%	8	90.06%	8	90.06%
5	8	88.11%	8	88.6%	8	88.7%	8	88.26%
Avg.	7.4	85.596%	7.6	87.224%	8	87.368%	7.4	89.22%



(a) Test dataset for two non-intersecting circles.

(b) Test dataset for two intersecting circles.

Figure 2. Artificial datasets: pairs of circles. In each case, one circle is an output class.

The training set consisted of 20 patterns and the test set contained 10000 patterns, with approximately the same number of patterns for each class. Different simulations were used to generate five pairs of train/test sets. The points are uniformly, independently and identically distributed inside the circles. For non-overlapping circles, the centres were (50, 50) and (50, 170), respectively, and for the overlapping circles (50, 50) and (50, 120). The radius was always 50.

The tests were performed with the following feature weights:  $\lambda = [1, 1]$  (the particular case of FAM),  $\lambda = [0.8, 1.2]$ ,  $\lambda = [0.5, 1.5]$ , and  $\lambda = [0, 2]$ . The individual and averaged values for the number of  $ART_a$  categories and the percent of correct classification (PCC) are reported in Tables 1 and 2 for non-overlapping and overlapping circles, respectively. For the last three values of  $\lambda$ , the number of  $ART_a$  categories is relatively the same as for the FAM. However, the PCC increases with the weight associated to the  $y$  feature, reaching the highest value when the  $x$  feature is completely omitted, i.e. when  $\lambda = [0, 2]$ . The PCC, for increasing value of the second weight, is monotonically increasing and at least as large as the PCC obtained for  $\lambda = [1, 1]$ .

## 5.2 The Iris dataset

This standard dataset contains 3 classes of 50 instances each, where each class refers to a type of the iris plant [14].

Table 3. Number of  $ART_a$  categories and PCC for Iris Dataset, for 10 orders of the dataset.

Test no.	$\lambda = [1, 1, 1, 1]$		$\lambda = [0.65249, 0.74508, 1.45877, 1.14366]$	
	Number of $ART_a$ categories	PCC	Number of $ART_a$ categories	PCC
1	16	93.33%	14	93.33%
2	14	96.66%	14	96.66%
3	7	96.66%	7	100.0%
4	6	90.0%	7	93.33%
5	11	96.66%	9	96.66%
Avg.	10.8	94.66%	10.2	96.0%

One class is linearly separable from the other 2; the latter are not linearly separable from each other.

In order to compare the behavior of FAM and FAMFW we performed the following experiments. We generated five random permutations of the whole set of 150 patterns. For each permutation, we selected the first 90 patterns as training set, the next 30 patterns as validation set and the last 30 patterns as test set.

In the first set of experiments, we considered  $\lambda = [1, 1, 1, 1]$ , which corresponds to FAM. We used 10 values for  $\rho_a$ : 0.0, 0.1, ..., 0.9 and 10 values for  $\beta_a$ : 0.1, 0.2, ..., 1.0.

In the second set of experiments, we obtained the feature weights by applying the ESRNG algorithm:  $\lambda = [0.65249, 0.74508, 1.45877, 1.14366]$ . We decided to limit the upper value of  $\beta_a$  to 0.5, because the large values increased the number of  $ART_a$  categories. We considered 10 values for  $\rho_a$ : 0.0, 0.1, ..., 0.9 and 5 values for  $\beta_a$ : 0.1, 0.2, 0.3, 0.4, 0.5.

The optimal values of  $\rho_a, \beta_a$  for FAMFW, with respect to the validation set, were used to train the FAMFW on the joined train and validation set, and then used for the test set. The results (number of  $ART_a$  categories and PCC) are reported in Table 3. For FAMFW with features weights determined by ESRNG, we obtained better or equal PCC values as with the FAM, with a slightly smaller number of  $ART_a$  categories.

## 6 Conclusion

The experiments with artificial datasets served as a proof-of-concept for our system. The dataset had one irrelevant feature (the  $x$  coordinate). Decreasing the associated weight we could improve the PCC, while maintaining a relatively constant number of input categories. As expected, the highest PCC was obtained for a null weight associated to the irrelevant feature. The test with overlapping classes was encouraging.

For the Iris dataset, the results were also good. Experimentally, we discovered that small values for  $\beta_a$  are more appropriate for the FAMFW, since they generate moderate extensions of the  $ART_a$  regions.

According to our preliminary experiments, using the

feature relevances can improve the FAM algorithm. Other feature algorithms may be also used. It is an open problem how to determine, from the nature of the training data, if the preliminary stage for computing the feature relevances is profitable. For datasets with very unequal feature relevances, our approach appears to be more performant than the basic FAM algorithm. Experiments with more complex and difficult datasets should be performed in the future.

## References

- [1] S. Grossberg, How does a brain build a cognitive code? *Psychological Review*, 1:151, 1980.
- [2] G. A. Carpenter, S. Grossberg, A massively parallel architecture for a self-organizing neural pattern recognition machine, *Computer Vision, Graphics and Image Processing*, vol. 37, 1987, 54–115
- [3] G. A. Carpenter, S. Grossberg, ART 2: Self organization of stable category recognition codes for analog input patterns, *Applied Optics*, vol 26, 1987, 4919–4930
- [4] G. A. Carpenter, S. Grossberg, ART 3: Hierarchical search using chemical transmitters in self-organizing patterns recognition architectures, *Neural Networks*, vol 3, 1989, 129–152
- [5] G. A. Carpenter, S. Grossberg, D. B. Rosen, Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system, *Neural Networks*, 1991, vol. 4, 759–771
- [6] G. A. Carpenter, S. Grossberg, J. H. Reynolds, ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network, *Neural Networks*, 1991, vol. 4, 565–588
- [7] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, D. B. Rosen, Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps, *IEEE Transactions on Neural Networks*, 1992, vol. 3, no. 5, 698–713

- [8] D. Charalampidis, G. Anagnostopoulos, M. Georgiopoulos, T. Kasparis, Fuzzy ART and Fuzzy ARTMAP with Adaptively Weighted Distances, *Proceedings of the SPIE, Applications and Science of Computational Intelligence, Aerosense 2002*, Vol. 4739, pp. 86-97
- [9] T. M. Martinetz, S. G. Berkovich, K. J. Schulten, Neural-gas network for vector quantization and its application to time-series prediction, *IEEE Trans. Neural Networks*, 4, 1993, 558–569.
- [10] B. Hammer, M. Strickert, T. Villmann, Supervised neural gas with general similarity measure, *Neural Processing Letters*, vol. 21, no. 1, 2005, 21–44.
- [11] R. Andonie, A. Çağaron, Feature Ranking using Supervised Neural Gas and Informational Energy, *Proc. of IEEE International Joint Conference on Neural Networks (IJCNN2005)*, Montreal, Canada, July 31 - August 4, 2005.
- [12] Chee Peng Lim, Robert Harrison, ART-Based Autonomous Learning Systems: Part I - Architectures and Algorithms, *Innovations in ART Neural Networks*, L. C. Jain and B. Lazzarini and U. Halici, eds., Springer, 2000
- [13] Mohammad Taghi, Vakil Baghmisheh, Pavešić Nikola, A Fast Simplified Fuzzy ARTMAP Network, *Neural Process. Lett.*, vol. 17, no. 3, 2003, 273–316
- [14] A. Asuncion, D.J. Newman, *UCI Machine Learning Repository*, University of California, Irvine, School of Information and Computer Sciences, <http://www.ics.uci.edu/~mllearn/MLRepository.html>, 2007.