Comments on "A Fuzzy Neural Network and its Application to Pattern Recognition"

Nikhil R. Pal, Gautam K. Mandal, and Eluri Vijaya Kumar

Abstract—**This note analyzes the unsupervised fuzzy neural network (FNNU) of Kwan and Cai and finds the following: the FNNU is a clustering net, not a classifier net, and the number of clusters the network settles to may be less or more than the actual number of pattern classes—sometimes it could even be equal to the number of training data points! The huge number of connections in the FNNU can be drastically reduced without degrading its performance. The algorithm does not have any learning capability for its parameters. Computational experience shows that usually the performance of an multilayer perceptron (MLP) is comparable to that of even a supervised version of FNN (trained by gradient descent algorithm) in terms of recognition scores, but an MLP has a much faster convergence than the supervised version of FNN.**

Index Terms— **Classifier, clustering, fuzzy neural network, pattern recognition.**

I. FNN of Kwan and Cai

Recently, Kwan and Cai[1] proposed an interesting unsupervised fuzzy neural network that can extract structural information from two-tone images. The effectiveness of unsupervised fuzzy neural network (FNNU) has been demonstrated in the recognition of shifted and distorted version of English characters and numbers.

The network consists of four layers. The first layer is a two-dimensional array of nodes and its size is exactly the same as that of the input pattern (say $m \times n$). The size of the second layer is the same as that of the first layer and each node in the second layer is connected to all nodes in the first layer. The number of nodes in the third layer is set *dynamically* and *should be* the same as that of the number of classes (C), but the algorithm may find it different. Every node of layer 3 is connected to every node of layer 2. The fourth layer is a competitive layer and has the same number of nodes as the third layer.

The output of nodes in the first layer is given by $y^1_{ij} = (x_{ij}/x_{\max})$ for $i = 1$ to m; $j = 1$ to n where $x_{ij} (\geq 0)$ is the (i, j)th pixel value of the input pattern and $x_{\max}$ is the maximum pixel value among all input patterns. Every node (p, q) in the second layer first computes $U_{pq} = \max^m_{i=1}(\max^n_{j=1}\{w[p-i, q-j]y^1_{ij}\})$ for $p = 1$ to m; $q = 1$ to n where $w[p-i, q-j] = \exp(-\beta^2((p-i)^2 + (q-j)^2))$ is the connection weight between (p, q)th node in the second layer and (i, j)th node of first layer and β is a constant. As every node in layer 2 is connected with every node in layer 3, each node in layer 2 produces C outputs, (C = number of classes). The rth output of the (p, q)th node in the second layer gives the extent the (p, q)th pixel supports the rth class. The (p, q)th node in the second layer then uses U_{pq} to compute membership to the rth class by

$$y^2_{pq,r} = \begin{cases} 1 - \frac{2}{\alpha}|U_{pq} - \theta_{pq,r}|, & \text{if } 0 \leq |U_{pq} - \theta_{pq,r}| \leq \frac{\alpha}{2} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

for $\alpha > 0$, $p = 1$ to m, $q = 1$ to n, $r = 1$ to C, where α is the base and $\theta_{pq,r}$ is the peak of the symmetric triangular membership function $y^2_{pq,r}$ and is computed using $\theta_{pq,r} = \max^m_{i=1}(\max^n_{j=1}\{w[p-i, q-j]x_{ij,k}\})$. Here k is index of the training pattern that defines a new class. The activation function of a node in the third layer computes the minimum of all inputs using $y^3_r = \min^m_{p=1}(\min^n_{q=1}(y^2_{pq,r}))$ for $r = 1$ to C. The nodes in the fourth layer compute the maximum value of outputs of layer 3. *Actually the process starts with $C = 1$ and C is increased depending on a threshold T_f and the outputs of layer 3.* To be more specific, if $(1 - \max^C_{r=1} y^3_r) > T_f$, then a new class (cluster) is created, i.e., C is incremented by one.

II. Some Remarks on FNNU

The FNNU is not a classifier net. It is essentially a clustering network that dynamically determines the number of clusters. The algorithm does not require class labels. It requires a few user supplied parameters, α, β, and T_f whose choice has strong influence on the performance of the system. Although, they have not used, they suggested to choose β such that at a distance of $\sqrt{8}$, the response of the Gaussian function is 0.5. The network architecture (the number of nodes in the output layer) is heavily dependent on T_f. Depending on the choice of T_f, the number of classes the network settles to may be more, sometimes even less, than the actual number of classes. The self-organizing process does not ensure that the number of nodes in the output layer would be equal to the actual number of classes. There is no guarantee that there exists a set of α, β, and T_f, which will terminate the algorithm at the desired number of clusters and, even if one is lucky to get such a set of parameters, the identified clusters may not necessarily correspond to actual classes present in the training set. Sometimes, the number of classes suggested by the net may become equal to the number of patterns, which usually is much more than the actual number of classes. In fact, authors indeed got 108 classes with a training set containing 108 patterns over 36 alphanumeric characters. In such cases, the net is not expected to have a good generalization.

Use of the FNNU for gray images may create problem as two pixels with different gray values, but being located at different positions may produce the same output response.

The number of connections in the network could be prohibitively large even for an image of size 256 × 256. For example, in this case the number of connections between layers 1 and 2 is 2^{32}.

Authors have mentioned about a learning algorithm for the net. However, the proposed algorithm sets the weights and parameters heuristically.

For a binary image, the output of the (p, q)th node is given by either the response of the Gaussian function at (0, 0) (when (p, q)th pixel contains one) or that of the nonzero pixel closest to (p, q). Therefore, connecting the (p, q)th node of layer 2 to only nodes within a neighborhood N_d of node (p, q) in layer 1 should suffice. In general, such a reduced network with a proper choice of d will have nearly the same localization property of the Gaussian function but will avoid the undesirable influence of the closet nonzero pixel

which is quite away from a zero pixel under consideration. If for a binary image $N_d(p,q)$ contains at least one pixel with nonzero value then the performance of the FNNU and FNNU with reduced connections are identical.

To get around the other problem of generating more classes than the actual number pattern classes, one can make a supervised version of the FNNU with the same architecture except the number of nodes in the output layer which is predetermined and is set exactly equal to actual number of classes C. In order to facilitate derivation of a gradient descent learning algorithm, the activation function of the nodes in the third layer is changed from min to a soft-min operation. Nodes in the second layer may still compute max or use a soft-max operation. A possible soft-max (min) operator SM could be

$$SM(x_1, x_2, \cdots, x_n) = \frac{\sum_i x_i e^{-sx_i}}{\sum_i e^{-sx_i}}$$

where s is a parameter of SM. SM can effectively realize *min* with a large positive value of s and max with a large negative value of s.

We can now really learn α and θ as follows. Let $X_T = \{X_1, X_2, \cdots, X_l\}$ be the set of training patterns and $\{T_1, T_2, \cdots, T_l\}$ be the set of class label vectors where X_i is an $m \times n$ image pattern and T_i is its C-dimensional label vector such that

$$t_{ir} = \begin{cases} 1 & \text{if } X_i \in \text{class } r \\ 0, & \text{otherwise.} \end{cases}$$

Let the output of the C nodes in the third layer with X_i as input be denoted by the C-dimensional vector Y_i, where $y_{ir} = y_r^3$. Let $e_i = \|T_i - Y_i\|^2$. Now using gradient descent we can find a set of α and θ such that

$$E = \sum_{i=1}^{l} e_i = \sum_{i=1}^{l} \|T_i - Y_i\|^2 = \sum_{i=1}^{l} \sum_{r=1}^{C} (t_{ir} - y_{ir})^2$$

is minimum.

We made some computational experiments similar to those in Kwan and Cai [1] using the supervised version of the net both with reduced and full connections as well as with the conventional multilayer perceptron (MLP).

III. Conclusions

1) This network as proposed in [1] is not a classifier, but a clustering net that dynamically determines the number of classes and, in the worst case, the number of extracted clusters can be equal to number of patterns in the training set.
2) The massive connections between layer 1 and layer 2 are neither necessary nor desirable. Computational experiments showed that the local connectivity over a neighborhood is enough.
3) The algorithm does not have any learning capability for its parameters. Equipping the FNNU with a gradient descent learning eliminated some of its problems and it could achieve generalization comparable to that of an ordinary MLP, but the MLP usually required less learning time when both algorithms are terminated using the same criterion.

References

[1] H. K. Kwan and Y. Cai, "A fuzzy neural network and its application to pattern recognition," *IEEE Trans. Fuzzy Syst.*, vol. 2, pp. 185–193, Aug. 1994.