# CHAPTER 3

## ADAPTIVE RESONANCE THEORY NETWORKS

Back propagation network is very powerful in the sense that it can simulate any continuous function given a certain number of hidden neurons and a certain forms of activation functions as demonstrated in 2.2. But training a back propagation network is quite time consuming. It takes thousands of epochs for the network to reach the equilibrium and it is not guaranteed that it can always land at the global minimum. Once a back propagation is trained, the number of hidden neurons and the weights are fixed. The network cannot learn from new patterns unless the network is re-trained from scratch. Thus we consider the back propagation networks don't have *plasticity*. Assuming that the number of hidden neurons can be kept constant, the plasticity problem can be solved by retraining the network on the new patterns using on-line learning rule. However it will cause the network to forget about old knowledge rapidly. We say that such algorithm is not *stable.* The contradiction between plasticity and stability and phenomenon is called *plasticity/stability dilemma* [17].

Adaptive Resonance Theory (ART) is a new type of neural network. It is designed by Grossberg in 1976 [25] to solve *plasticity /stability dilemma*. The first version of ART, ART-1, proposed by Carpenter and Grossberg in 1987, is used to cluster binary data. Since then several variations of ART have been developed. The most important ones are: ART-2 [8], an extension of ART-1, used to cluster analog data, ARTMAP [10], a supervised learning mechanism for binary data, and Fuzzy ARTMAP [13], a supervised learning algorithm for analog data. Many researchers around the world have proposed other types of

ART networks. Among them are Adaptive Hamming Net (AHN) [28], Gaussian ART(GA) [49], simplified Fuzzy ARTMAP (SFAM) [29], simplified ART (SART) [1][3], and relatively new $\mu$ ARTMAP [23]. The contribution of AHN is that it realized the inefficiency in ART algorithm and improved it by converting the searching problem to an optimization problem. Gaussian ART (GA) [49] introduces Gaussian mixture model into ART. SFAM is essentially sequential counterpart of parallel Fuzzy ARTMAP. SART represents a group of ART which can be implemented optimally. $\mu$ ARTMAP [23] was designed to resolve the category proliferation problem encountered in Fuzzy ARTMAP. In the following sections we will briefly describe some popular ART networks. A list of notations used through out the explanations follows:

$\mathbf{x}$ : $\mathbf{x} = (x_1, x_2, ..., x_D)^T$ , *a pattern*

$\mathbf{X}$ : *set of patterns or sample,* $\mathbf{x} \in \mathbf{X}$

*D:* *number of features (dimension of the feature space)*

*C:* *number of output nodes*

*J:* *index of clusters (output nodes)*

*k:* *index of classes (categories) index*

*K:* *number of categories (classes)*

$\beta$ : *learning rate (* $\beta < 1$*)*

$\rho$ : *vigilance*

$R_0$ : *radius of initial sphere of a cluster*

*m:* *mahalanobis distance (* $m = (\mathbf{x} - \mathbf{w})^T \mathbf{S}^{-1} (\mathbf{x} - \mathbf{w})$ *)*

*W:* *collection of templates of output node,* $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_C\}$

*N:* *collection of output node sizes* $N = \{N_1, N_2, ..., N_C\}$

*L:* *collection of output nodes labels* $L = \{L_1, L_2, ..., L_C\}$

$\omega$: *collection of categories* $\omega = \{\omega_1, \omega_2, ..., \omega_K\}$

$\omega_k$ : *the k-th class, k = 1,2,…,K*

$\mathbf{w}(:, j)$ : *sample mean (template) of j-th output node*

$\mathbf{S}(:,:, j)$ : *covariance matrix of j-th output node*

$\mathbf{Q}(:,:, j)$ : *inverse covariance matrix of* $\mathbf{S}(:,:, j)$

### 3.1 ART-1

ART-1 is the first version of ART-based networks proposed by Carpenter and Grossberg[7]. The network was intended for unsupervised clustering of binary data. It has two major subsystems: *attentional subsystem* and *orienting subsystem*. The attentional subsystem is a one layer neural network. It has *D* input neurons to learn *D* -dimensional data and *C* output neurons to map *C* maximum clusters. Initially all output neurons are uncommitted[1]. Once an output neuron learned from a pattern, it becomes committed. The activation function is computed at all committed output neurons. The input and output is connected by both top-down and bottom-up weights. Baraldi & Parmiggiani have proved mathematically that the bottom-up and top-down attentional module is equivalent to an attentional system with only forward connections [1]. Baraldi and Alpaydin generalize this result to all ART-1 based networks [2] by stating : “the attentional module of all ART 1

[1] Uncommitted output nodes are “space holders” for the future output nodes, which is necessary in hardware implementation (hardware doesn’t use dynamic structures as software)

based systems is functionally equivalent to a feed-forward network featuring no top-down connections. The architecture of simplified ART is shown in Figure 3-1.

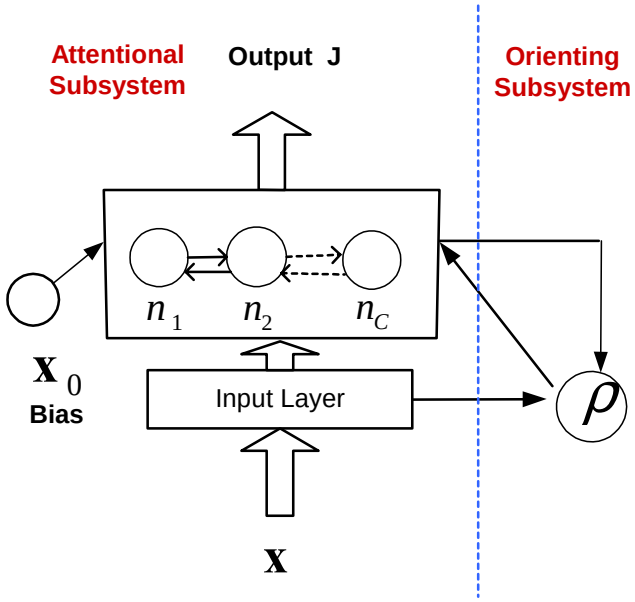


Figure 3-1: Architecture of a simplified ART.

The weight vectors are initialized to 1, for instance, $w_{j1} = 1, w_{j2} = 1, ..., w_{jD} = 1$, $1 \le j \le C$. The orienting subsystem is a qualifier, where the match function of the candidate elected by the attentional system is compared against the vigilance. $\rho$. It uses the winner-take-all learning strategy. If the condition satisfied, the pattern will be learned by the winning node, otherwise the activation function for the candidate will be set to 0, and a new candidate is elected and tested. The searching procedure keeps on going until either a candidate meets the vigilance constraint or no more candidates are left. If none of the output nodes can encode the pattern, a new node is committed to the pattern.

Our interpretation of simplified sequential implementation of parallel counterpart of ART-1 is shown in Figure 3-2.

The goal of the network training is to find a set of templates, which best represent the underlying structure of the samples. Suppose the set of templates $W = \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_C\}$ and the number of patterns from *X* associated with each template $N = \{N_1, N_2, ..., N_C\}$. It is important to note that the number of output nodes, sets *W* and *N* are growing dynamically. The algorithm of ART-1 follows: The three functions, *T ()*, *M ()* and *U ()*, used in the algorithm shown in Figure 3-2 are defined as follows:

```
C= 0;                                  Initialize the number of current templates
W = {ones(D), ones(D),…ones(D)}        Initialize set of templates
while (X not empty)
Learning loop
{
    get x;                             Get a pattern from X
    new = true;                        Set flag "new node needed"
    loop j = 1,C                       Compute activation value for all templates
        t_j = T(x, w_j);
    loop  i = 1,C                      Search for resonance
    {
          J = arg max_{j≤C} T_j        Find template with highest activation value
            if M(x, w_J) > ρ           When resonance occurs
            {
                  w_J := U(x, w_J);    Update the template
                new = false;           No new node needed
                break;                 Stop the search for the resonant node
              }
             else                      If resonance doesn't occur,
          T_J = 0                      Reset w_J
     }                                 Continue search for resonant node
     NEWNODE(new);                     Create new node if needed
}                                      End of learning loop
```

Figure 3-2: Clustering algorithm of sequential implementation of ART-1.

*NEWNODE(new)* in Figure 3-2 is a macro routine that allocate a new node (template) to the network. It is shown in Figure 3-3.

```
if new == true
{
    C := C + 1                  Increment current number of templates
    w_C := x                    The new template is initially equal to x
    W := W ⊕ w_C                Add new template to the set W
}
```

Figure 3-3: Macro routine of *NEWNODE()* in ART-1.

$T(\mathbf{x}, \mathbf{w})$ is called the *choice function* [7] or activation function, which is used to measure the degree of the resemblance of $\mathbf{x}$ with $\mathbf{w}_j$,

$$T(\mathbf{x}, \mathbf{w}_j) = \frac{\left\| \mathbf{x} \cap \mathbf{w}_j \right\|}{\alpha + \left\| \mathbf{w}_j \right\|}, \tag{3.1}$$

where $\alpha$ is a choice parameter, $\alpha > 0$.

$M(\mathbf{x}, \mathbf{w}_j)$ is called the *match function*, which is used to qualify how good is the likeness of $\mathbf{w}_j$ to $\mathbf{x}$.

$$M(\mathbf{x}, \mathbf{w}_j) = \frac{\left\| \mathbf{x} \cap \mathbf{w}_j \right\|}{\left\| \mathbf{x} \right\|}, \tag{3.2}$$

The function is used in conjunction with the *vigilance parameter* $\rho \in (0,1]$, where $M(\mathbf{x}, \mathbf{w}_j) > \rho$ means a good match (resonance). The vigilance is the most important network parameter that determines its resolution: larger vigilance value normally yields larger number of output nodes and good precision.

$U(\mathbf{x}, \mathbf{w}_j)$ is called the *update function,* which is used to update a template after it resonances with a pattern:

$$U(\mathbf{x}, \mathbf{w}_j) = (1-\beta)\mathbf{w}_j + \beta(\mathbf{x} \cap \mathbf{w}_j) \tag{3.3}$$

where $\beta$ is learning rate, $0 < \beta \leq 1$. Higher values of $\beta$ result in faster learning. It is called the fast learning in ART when $\beta = 1$.

The operator $\cap$ in equations (3.1) through (3.3) is bitwise AND operator, such that $\mathbf{a} \cap \mathbf{b} = (a_1 \text{ AND } b_1, a_2 \text{ AND } b_2, ..., a_C \text{ AND } b_C)$, and $\|a\|$ is the number of ones in vector $\mathbf{a}$, $\|a\| = \sum_{i=1}^{D} a_i$.

### 3.2 Fuzzy ART

Later, Carpenter, Grossberg et al [11] extended the capability of ART-1 to clustering of analog patterns. Fuzzy ART dynamics are described in terms of fuzzy set-theory operations [13][31][51], for instance, the bitwise AND operator $\cap$ in ART-1 is replaced by the fuzzy AND operator $\wedge$: $\mathbf{a} \wedge \mathbf{b} = (\min(a_1, b_1), \min(a_2, b_2), ..., \min(a_D, b_D))$, and norm operator $\|.\|$ is defined by $\|\mathbf{a}\| = \sum_{i=1}^{D} |a_i|$. Therefore, Fuzzy ART works with both binary and analog patterns. Fuzzy ART requires input pattern to be normalized to prevent category proliferation. The normalization is done by using complement coding, which is $\mathbf{x} = (\mathbf{x}, \mathbf{x}^c) = (x_1, x_2, ..., x_D, 1-x_1, 1-x_2, ..., 1-x_D)$. It can be easily proven that $\| x \| \equiv D$. The general structure of the fuzzy ART-1 is essentially the same as ART-1.

## 3.3 ARTMAP Networks

ART and Fuzzy ART are unsupervised clustering methods. ARTMAP [10] on the other hand performs incremental supervised leaning of labeled patterns. ARTMAP contains a pair of ART modules, $ART_a$ and $ART_b$. Patterns (without labels) are sent to $ART_a$, and their labels are sent to $ART_b$. $ART_a$ and $ART_b$ are linked by an associative learning network and an internal controller that ensures system to operate in real time [13]. If a prediction made by $ART_a$ disconfirmed by $ART_b$, a mechanism called match tracking will be triggered. It increases the vigilance at $ART_a$, which leads to the selection of new candidate. The algorithm of sequential version of ARTMAP is given in Figure 3-4 - Figure 3-5. Comparing with the algorithms in Figure 3-2, algorithm in Figure 3-4 has one extra loop which checks if the label of the pattern matches with the label of the template. If labels match, the algorithm proceeds as that in Figure 3-2. If label doesn't match, the vigilance is boosted to activation value of current candidate plus a small positive number, and the current winning node is suppressed. The macro routine of *NEWNODE (new)* used to allocate a new node (template) to the network in Figure 3-5 is basically the same as in Figure 3-2 except that the label information is stored in Figure 3-5.

## 3.4 Geometric Representation of Fuzzy ART with Complement Coding

Fuzzy ARTMAP use hyper rectangles to represent category weights in a supervised learning paradigm [13]. The same geometrical representation can be found in the nested generalized exemplar (NGE) system [35][37] and the fuzzy min-max classifier (FMMC) system [41].

```
ω = {};                                  Initialize set of classes (categories)
L = {};                                  Initialize the map (set of labels)
W = {};                                  Initialize set of templates
N = {};                                  Initialize the cluster sizes
C = 0;                                   Initialize the current number of templates
while (X not empty)
Learning loop
{
    get x;                               Get a labeled pattern from X
    new = true;                          Set flag "new node needed"
    if (label(x) ∉ ω)                    If the class hasn't been seen so far
        ω := ω ⊕ label(x);               Add new class to ω
    else
    {
        loop j = 1,C                     Compute activation value for all templates
            t_j = T(x, w_j);
        loop j = 1,C                     Search for resonance
        {
            J = arg max_{j≤C} T_j;       Find template with highest activation value
            if (L(J) = label(x))         If the label of the winner match
            {
                if (M(x, w_J) > ρ        And the resonance occurs
                {
                    w_J := U(x, w_J);    Update the template
                    new = false;         No new node needed
                    break;               Stop the search for the resonant node
                }
                else                     If no match
                    T_J = 0;             Reset template J
            }                            If label or distance don't match
            else
            {
                T_J = 0;                 Reset its activation value, so J will not be elected
                ρ = T_J + ε;             Match tracking triggered
            }                            Continue search for resonant node
    }
    NEWNODE(new);                        Create new node if needed
}
```

Figure 3-4: Clustering algorithm of sequential implementation of ARTMAP.

```
if new = true
{
    C := C + 1                      Increment current number of templates
    N_C := N_C ⊕ 1;                 The new cluster has one sample
    w_C := x                        The new template is initially equal to x
    W := W ⊕ w_C                    Add new template to the set W
    L := L ⊕ label(x) ……            The new template has the same label as the pattern
}
```

Figure 3-5: Marco routine of *NEWNODE ()* in ARTMAP

To simplify things, let us suppose that we are dealing with two-dimensional sample $\mathbf{X}$. $\mathbf{x}$ is any pattern belongs to $\mathbf{X}$, $\mathbf{x} = (x_1, x_2)$. After complement coding, the pattern becomes:

$$\mathbf{I} = (\mathbf{x}, \mathbf{x}^c) = (x_1, x_2, 1 - x_1, 1 - x_2). \tag{3.4}$$

In the same way, a template *j* is coded as:

$$\mathbf{w}_j = (\mathbf{u}_j, \mathbf{v}_j^c). \tag{3.5}$$

The geometric meaning of the template can be understood as a rectangle $R_j$, where $\mathbf{u}_j$ and $\mathbf{v}_j$ are two diagonal corners.

When an output neuron *J* is first committed with a pattern $\mathbf{x}^{(i)}$, the template *J* is $\mathbf{w}_J = (\mathbf{x}_i, \mathbf{x}_i^c)$, which is a point in feature space as seen in Figure 3-6.

When the template *J* is chosen again by pattern $\mathbf{x}^{(k)}$, suppose fast learning is used, the template *J* is updated by:

$$\begin{aligned}\mathbf{w}_J &= \left((\mathbf{x}^{(k)} \wedge \mathbf{x}^{(i)}), (\mathbf{x}^{(k)c} \wedge \mathbf{x}^{(i)c})\right) \\ &= \left((\mathbf{x}^{(k)} \wedge \mathbf{x}^{(i)}), (\mathbf{x}^{(k)} \vee \mathbf{x}^{(i)})^c\right) . \\ &= \left(\mathbf{u}_j, \mathbf{v}_j^c\right)\end{aligned} \tag{3.6}$$

And, it grows into a rectangle as shown in Figure 3-7.

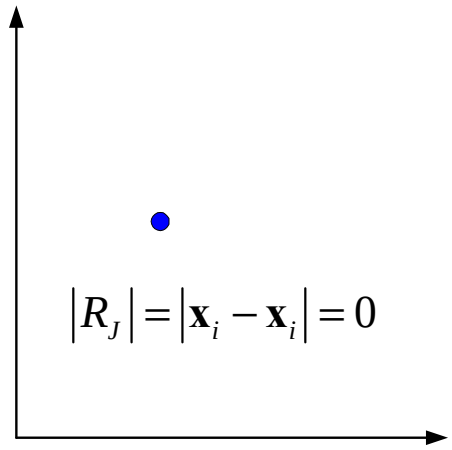


Figure 3-6: Template *J* is encoded with one pattern.

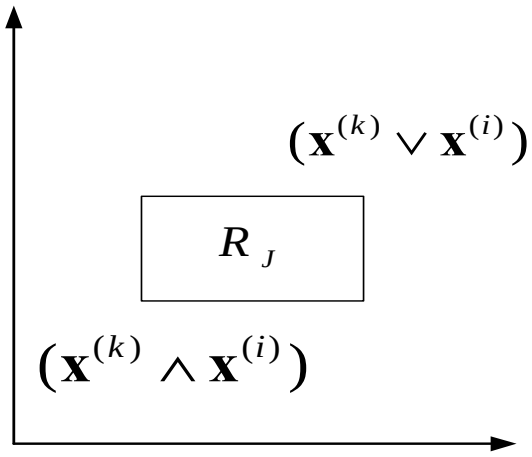


Figure 3-7: Template *J* is encoded with two patterns.

From Figure 3-7, we observe that $R_J$ grows with the learning process. It is the smallest rectangle that covers all the patterns it has learned. This property demonstrates that ART inherently is stable, in other words the old knowledge will always be retained.

In general, if $\mathbf{x}$ has dimension *D*, the hyper rectangle $R_j$ includes the two vertices $\wedge_j \mathbf{x}$ and $\vee_j \mathbf{x}$ , where the i-th component of each vector is defined by the equations [11]:

$$\left(\wedge_j \mathbf{x}\right)_i = \min\left\{x_i : \mathbf{x} \textit{ has been coded by category } j\right\} \tag{3.7}$$

$$\left(\vee_j \mathbf{x}\right)_i = \max\left\{x_i : \mathbf{x} \textit{ has been coded by category } j\right\} \tag{3.8}$$

and the weight $\mathbf{w}_j$ is given by:

$$\mathbf{w}_j = \left(\wedge_j \mathbf{x}, \left(\vee_j \mathbf{x}\right)^c\right) \tag{3.9}$$

## 3.5 Gaussian ARTMAP

Williamson has recognized two weaknesses of the Fuzzy ARTMAP [49]: sensitivity to noise, and inefficiency representation of fuzzy categories. To solve the problem, Williamson incorporated Gaussian distribution into choice, match and classification function. The geometrical representation of category changes from hyper rectangles to hyper diagonal ellipsoids. The benefit of this innovation is that choice and match functions increase monotonically toward the center of a cluster, which therefore effectively prevents the proliferation of clusters caused by noise [49]. In addition, the Gaussian distributions are the most common distributions in nature. They have especially useful generalization properties in high dimensional spaces [17][33][34][49]. This new network is called Gaussian ARTMAP (GA). One limitation of GA is that it uses independent Gaussian distributions.

Thus, GA needs more diagonal ellipsoids to cover the same amount of data points than using general ellipsoids, which is illustrated in Figure 3-8.

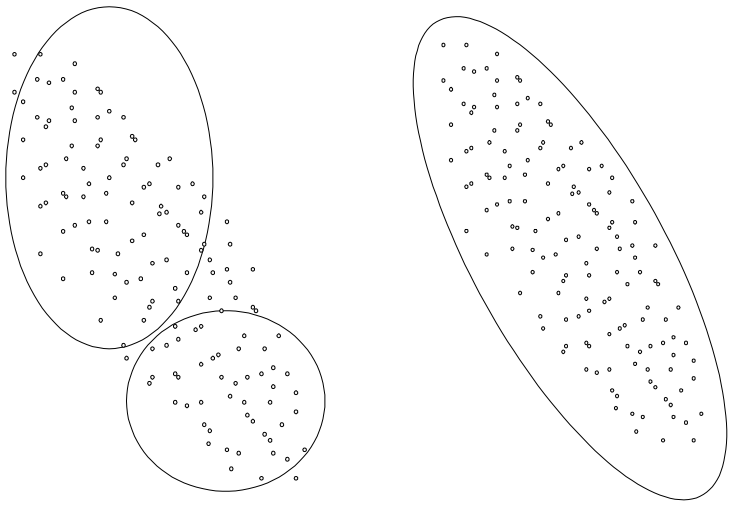

Figure 3-8: To cover the same set of points, fewer ellipsoids are required by using general ellipsoids as oppose to the diagonal ellipsoids.

### 3.5.1 Choice Function

Each GA output node is represented by its sample mean, standard deviation along each dimension, and the number of patterns it encloses. We have known from Bayes' theorem that the posterior probability of cluster *j* given patter **x** is:

$$P(j \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid j)P(j)}{p(\mathbf{x})}. \tag{3.10}$$

Since the clusters are defined by separable Gaussian distribution, the conditional probability density of **x** given cluster *j* is:

$$p(\mathbf{x} \mid j) = \frac{1}{(2\pi)^{D/2}\prod_{i=1}^{D}\sigma_{ji}} e^{\left(-\frac{1}{2}\sum_{i=1}^{D}\left(\frac{w_{ji}-x_i}{\sigma_{ji}}\right)^2\right)}. \tag{3.11}$$

Assume there are *C* clusters, the prior probability of cluster *j* is the ratio of patterns it encompasses to the total number of patterns, which is

$$P(j) = \frac{N_j}{\sum_{j=1}^{m} N_j}. \tag{3.12}$$

The choice function of a template is proportional to the logarithm of its posterior probability, which is

$$\begin{aligned} T_j(\mathbf{x}) &= \log\left( (2\pi)^{D/2}\, p(\mathbf{x} \mid j) P(j) \right) \\ &= -\frac{1}{2}\sum_{i=1}^{D}\left( \frac{w_{ji} - x_i}{\sigma_{ji}} \right)^2 - \log\left( \prod_{i=1}^{D} \sigma_{ji} \right) + \log\left( P(j) \right) \end{aligned}. \tag{3.13}$$

Since the denominator of (3.10), $p(\mathbf{x})$, is the same across all the clusters, it is omitted, so is the dimensional scaling factor, $(2\pi)^{D/2}$. The cluster with the maximum choice function is selected as the candidate.

$$J = \arg\max_{1 \le j \le m} (T_j(\mathbf{x})). \tag{3.14}$$

### 3.5.2 Match Function

The match function of the candidate is

$$\begin{aligned} M(\mathbf{x}, J) &= -\frac{1}{2}\sum_{i=1}^{D}\left( \frac{x_i - w_{Ji}}{\sigma_{Ji}} \right)^2 \\ &= T(\mathbf{x}) + \log\left( \prod_{i=1}^{D} \sigma_{Ji} \right) - \log\left( P(J) \right) \end{aligned}. \tag{3.15}$$

As seen, it is equivalent to the Mahalanobis distance from **x** to the candidate template *J*. Resonance happens when $M(\mathbf{x}, J) > \rho$.

### 3.5.3 Update Function

When template *J* qualifies to learn from a pattern, it updates the following three parameters:

$$\begin{aligned} &N_J := N_J + 1; \\ &w_{Ji} := (1-\beta_J) w_{Ji} + \beta_J x_i; \qquad i = 1, 2, ..., D \\ &\sigma_{Ji} := \sqrt{(1-\beta_J)\sigma_{Ji}^2 + \beta_J (w_{Ji} - x_i)^2}; \end{aligned} \tag{3.16}$$

The learning rate $\beta_J$ changes with the cluster size $N_J$, i.e. $\beta_J = 1/N_J$.

### 3.5.4 Classification

After training GA network, the feature space is divided into *C* hyper ellipsoids, whose principal axes are parallel with coordinate axes. Each class maps to one or more such ellipsoids. Essentially each of these classes is equivalent to an independent Gaussian mixture model. Therefore, the posterior probability of class $c_k$ given sample **x** can be expressed as:

$$P(\omega_k \mid \mathbf{x}) = \sum_{j,\, L(j)=\omega_k} P\left(L(j) \mid \mathbf{x}\right) \sim \sum_{j,\, L(j)=\omega_k} \exp\left(T(\mathbf{x}, \mathbf{w}_j, \boldsymbol{\sigma}_j)\right). \tag{3.17}$$

Consequently, the maximum likelihood estimate of the class of an unknown (unlabeled) sample **y** will be given as:

$$class = \arg\max_{\omega_k} \left(P(\omega_k \mid \mathbf{y})\right) \tag{3.18}$$

Williamson has shown [49] in several benchmarks that the Gaussian ARTMAP has better performance than fuzzy ARTMAP in terms of the hit rate and the number of required output nodes.

## 3.6 Adaptive Hamming Net

As we recall in ART models, a candidate elected according to the maximum choice function, needs further to go through matching test, to decide if it is a winner. In the worst case, all the committed output neurons have to be searched. As we can see the searching procedure is sequential operations embedded in parallel architecture, which is bottleneck of the algorithm and makes the advantage of parallel architecture can't be fully realized. Adaptive Hamming net, proposed by Cheng-An Hung and Sheng-Fuu Lin [28], was designed to solve the problem.

The architecture of an adaptive Hamming net us illustrated in Figure 3-9. An adaptive Hamming net is a two-layer neural network. The number of hidden neurons is equal to the number of output neurons. The input layer is fully connected with the hidden layer, while hidden layer and the output layer is connected through single bottom-up weight from hidden neuron *j* to output neuron *j*. The hidden layer is implemented as a matching score net and the output layer is implemented as MAXNET. In the layer of matching score net, all the prototypes which are similar enough to the input pattern are activated. In MAXNET, the outputs of activated neurons are calculated and the prototype of the largest choice function is selected as the winner. We observe from here that Adaptive Hamming net essentially inverses the order of operations of choice and match function as opposed to that of ART. This change effectively converts the problem of searching for the winner to an optimization problem.

### 3.6.1 Initialization of Network

The weights of input-to-hidden layer are initialized to 1,

$$w_{j1} = w_{j2} = ...w_{jD} = 1, \quad j = 1,...,C, \tag{3.19}$$

and the weights of hidden-to-output layers are initialized by:

$$w_C < ... < w_j < ... < w_1 < \frac{1}{\alpha + D}, \tag{3.20}$$

which makes sure that a new category is chosen in the order of $j = 1, 2, ..., C$.

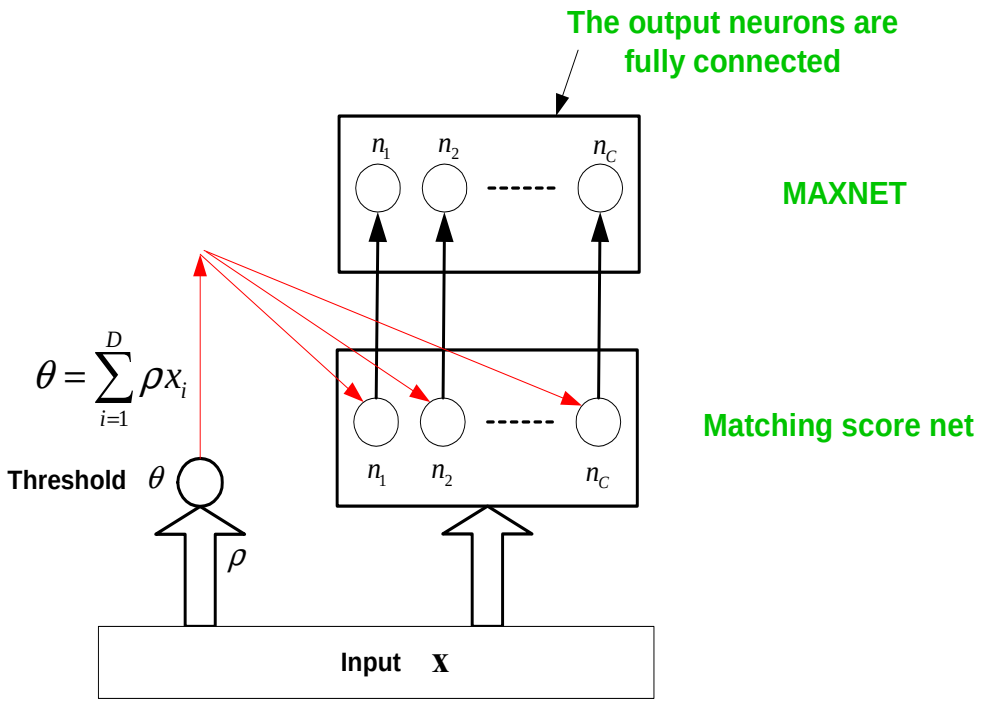


Figure 3-9: Architecture of an adaptive Hamming net.

### 3.6.2 Match Score Net

Activation function at the hidden neuron $j$ is defined by:

$$a_j^{(1)} = \sum_{i=1}^{D} w_{ji} x_i \ . \tag{3.21}$$

A step function (piecewise linear function),

$$y_j^{(1)} = f(a_j^{(1)}) = \begin{cases} 0 & if\ a_j^{(1)} < \theta \\ a_j^{(1)} & if\ a_j^{(1)} > \theta \end{cases} \tag{3.22}$$

was used to select a set of prototypes. The prototypes, whose activation functions are larger than threshold $\theta$, will be activated, otherwise will be suppressed. One important feature of adaptive Hamming net is that the threshold $\theta$ is a function of input pattern,

$$\theta = \sum_{i=1}^{D} \rho x_i \,, \tag{3.23}$$

where $\rho$ is the vigilance, $0 \le \rho \le 1$. After substituting (3.23) and (3.21) into (3.22), the condition for an output neuron to be activated can be explicitly expressed as:

$$\frac{\sum_{i=1}^{D} w_{ji} x_i}{\sum_{i=1}^{D} x_i} \ge \rho \,. \tag{3.24}$$

Equation (3.24) is similar to the match function (3.2) in ART-1.

### 3.6.3 MAXNET

The output function at neuron *j* is defined as:

$$y_j^{(2)} = w_j y_j^{(1)} \,. \tag{3.25}$$

The winning neuron is the one which has the largest output function:

$$J = \arg\max_{1 \le j \le C} y_j^{(2)} = \arg\max_{1 \le j \le C} w_j y_j^{(1)} \,. \tag{3.26}$$

After we plug (3.22) and (3.25) into (3.26), the winner node is found by:

$$J = \arg\max_{1 \le j \le C} \; w_j \sum_{i=1}^{D} w_{ji} x_i$$
$$= \arg\max_{1 \le j \le C} \; \frac{\sum_{i=1}^{D} w_{ji} x_i}{\alpha + D + \varepsilon} \quad , \tag{3.27}$$

where $\varepsilon$ is a small positive number which makes the condition in (3.20) satisfied.

Eqn. (3.27) is similar to the choice function in ART-1 (3.1).

### 3.6.4 Fast Learning

For fast-learning, the learning rule for the input-to-hidden weights is:

$$w_{Ji} \coloneqq w_{Ji} x_i \,, \tag{3.28}$$

while for the hidden-to-output weights, the learning rule follows:

$$w_J = \frac{1}{\alpha + y_J^{(1)}} \tag{3.29}$$

## 3.7 SFAM

SFAM stands for simplified Fuzzy ARTMAP proposed by Kasuba [29] in 1993, It is equivalent to the sequential implementation of ARTMAP.

## 3.8 EART, S-Fuzzy ART and SART

As we recall that the AHN converts the maximization problem of ART to the optimization implementation [28]. It characterizes no “mismatch reset condition and repeat search process.” [3]. Andrew Baraldi and Ethem Alpaydin [3] name this type of implementation scheme computationally efficient ART or short for EART. There are two versions of EART, EART-1 and EART-2, based on whether match function increases monotonically with activation function.

The EART-1 is equivalent to the sequential version of the parallel AHN processing scheme , where its activation function doesn't increase with its match function .

In case of activation function increasing with the match function, if the match function corresponding to the largest activation doesn't satisfy resonance condition, then the other activations will not pass resonance test either. Thus only the output node of the largest activation needs to be tested against the resonance condition and no search process is needed. If the resonance condition is satisfied, learning will happen; otherwise a new template will be created. This implementation scheme is called EART-2 [3].

In ART-1 based clustering networks, activation function measures the closeness of a pattern to a template while the match function measures the closeness of a template to a pattern. $T(\mathbf{x}, \mathbf{w}_j) \neq T(\mathbf{w}_j, \mathbf{x})$ and $M(\mathbf{x}, \mathbf{w}_j) \neq M(\mathbf{w}_j, \mathbf{x})$. Andrew Baraldi and Ethem Alpaydin argue that [3] "ART 1-based clustering networks employ an inherently nonsymmetrical architecture to compute an intrinsically symmetrical degree of match". They proposed *symmetric Fuzzy ART* (*S-Fuzzy ART*) which adopts symmetric activation and match function [3], such that $T(\mathbf{x}, \mathbf{w}_j) = T(\mathbf{w}_j, \mathbf{x})$ and $T(\mathbf{x}, \mathbf{w}_j) = M(\mathbf{x}, \mathbf{w}_j)$ . The two instances of the symmetric activation and match function are[3]:

$$T(\mathbf{x}, \mathbf{w}_j) = M(\mathbf{x}, \mathbf{w}_j) = \frac{1}{1 + \left\| \mathbf{x} - \mathbf{w}_j \right\|^2} \tag{3.30}$$

and

$$T(\mathbf{x}, \mathbf{w}_j) = M(\mathbf{x}, \mathbf{w}_j) = \frac{\sum_{d=1}^{D} \min\{x_d, w_d\}}{\sqrt{\sum_{d=1}^{D} x_d \sum_{d=1}^{D} w_d}} \tag{3.31}$$

S-Fuzzy ART can be implemented using scheme of EART-2. It has shown that S-Fuzzy ART is superior to Fuzzy ART in terms of clustering accuracy and robustness with respect to changes in the order of presentation of the input data [3].

After S-Fuzzy ART met its original goal, Andrew Baraldi and Ethem Alpaydin proposed a new group of ART networks called simplified ART (SART) [3], which is a generalization of S-Fuzzy ART and can be implemented by using EART schemes. GART and S-Fuzzy ART are two examples of SART.