CN550: Neural and Computational Models of Recognition, Memory, and Attention

Unsupervised Learning

Lecture 3

Instructors: Anatoli Gorchetchnikov <anatoli@bu.edu>
Heather Ames <starfly@cns.bu.edu>
Teaching Fellow: Karthik Srinivasan <skarthik@bu.edu>

Unsupervised vs. Supervised

Supervised algorithms give a formula for an input to predict the output based on the learned examples

The basic skill of unsupervised learning is the ability to compare two instances and judge the similarity between them in order to determine whether they belong to the same class of instances

The idea is identical to the one we used for the supervised learning, namely measure the distance between some attribute values

The difficulty comes from the fact that we know neither which attributes are important, nor which distances are large enough to separate instances into different classes

This knowledge has to come along with the learning process

Five Reasons to Use Unsupervised Learning

Collecting and labeling a large set of data points can be costly

One can build basic classification structure with unlabeled data and then assign labels from smaller labeled set

Properties of individual instances might change with time, tracking these in an unsupervised mode might improve performance

We can use unsupervised methods to find features that will then be useful for categorization (“smart preprocessing”)

In the early stages of an investigation it may be valuable to gain some insight into the nature or structure of the data

Unsupervised Approach

Establish a set of dimensions that might be relevant and then try to group our instances into classes on the basis of their similarities in these dimensions

We can use our general knowledge to determine relevant dimensions, but how do we judge the relevance?

A complete system trained on all dimensions might become slow and cumbersome

There still might be some relevant dimensions that we did not include just because we did not perceive them

The whole issue of selection of appropriate dimensions is a complex area of ongoing research

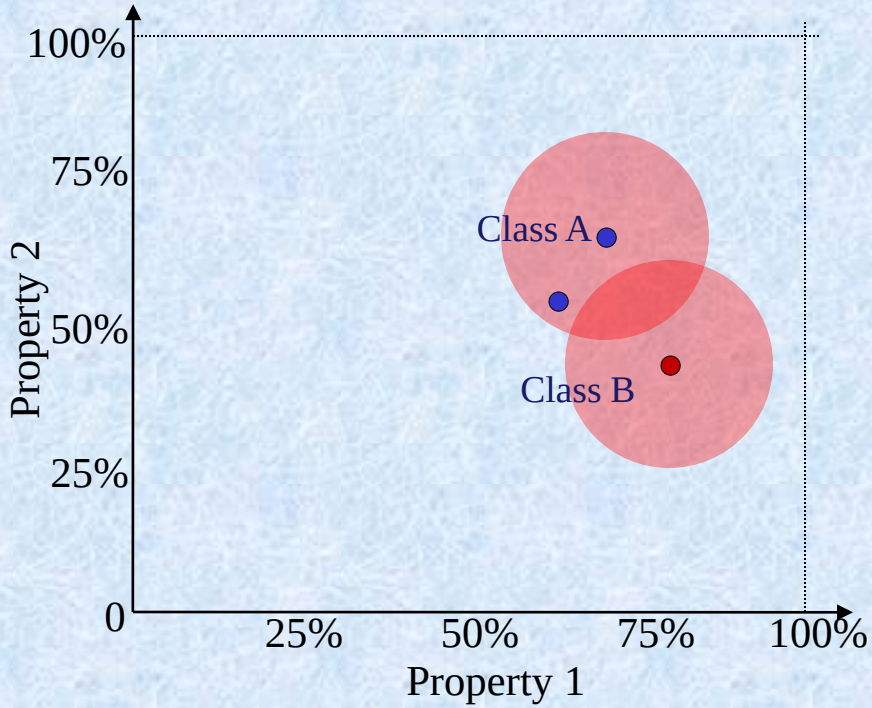
Leader Algorithm

Partition inputs into clusters of radius no greater than T

As with many other fast learning algorithms the partitioning depends on the order of presentation

First input – a leader, classified as class A

Put subsequent instances in class A as long as distance from the leader is no greater than T



Otherwise use instance as a center of a new class B

Next instance goes in class A if possible, otherwise in class B if possible, otherwise…

Leader Algorithm

Advantages:

- Fast and simple
- Only needs to keep leaders in memory
- Incremental

Disadvantages:

- Order-dependent
- Biased towards earlier established classes
- Uses a fixed distance, so assumes identical distributions for all classes

K-means Algorithm

Given a set of inputs partition them into K classes

The mean (average) of each cluster is stored

Local optimization method: try to find a partition to minimize a given error function, e.g. sum of squares of distances from the mean

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x_j \in S_i} \left\| x_j - \mu_i \right\|^2$$

Brute force approach for incremental adjustment

- Partition inputs
- For each input see if moving it to a different cluster reduces error
- Continue until no further reduction of error occurs

K-means Algorithm

Can become stuck in a local minimum

To overcome start with different initial partitions, run several times

The complexity is not so permissive to do it:

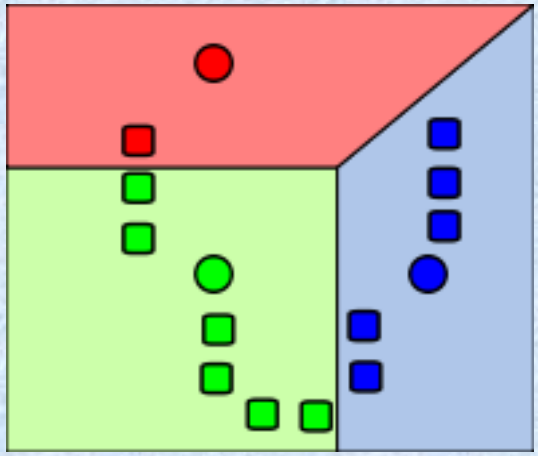
General case in NP-hard

For a fixed space dimensionality d and number of clusters k the complexity is $O(n^{dk+1}\log n)$ where n is a size of a training set

Lloyd's K-means Algorithm

Fast heuristic approach (Lloyd's algorithm):

1. Pick initial cluster means
2. Split the data points into classes according to these means

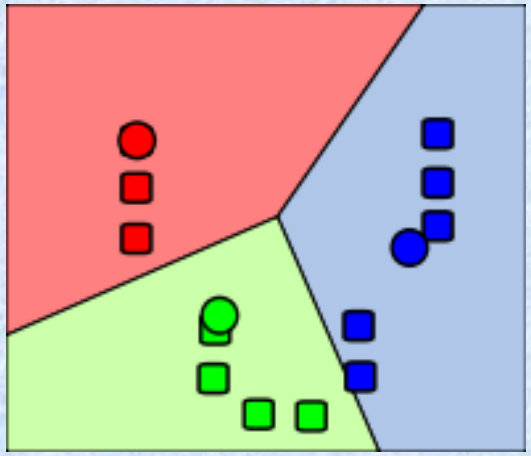


Lloyd's K-means Algorithm

Fast heuristic approach (Lloyd's algorithm):

1. Pick initial cluster means
2. Split the data points into classes according to these means
3. Recalculate the means based on the current membership

Continue 2 and 3 until all means stopped moving



K-means Algorithm

Lloyd's algorithm is also prone to falling in local minimum, but since it is so fast, multiple runs are not an issue

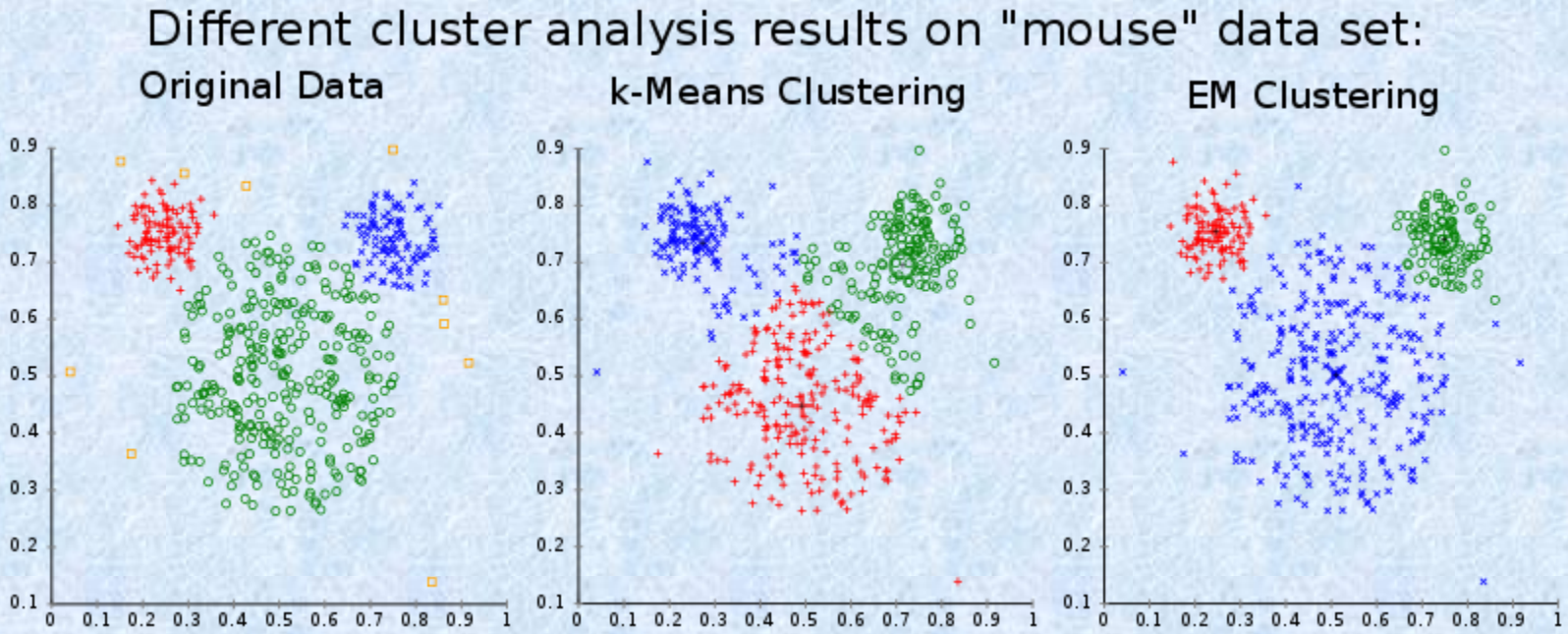
There can be artificially constructed data sets where the convergence takes exponential time, but real data rarely leads to it

Assignment step sometimes is called expectation, and update step is called maximization, which makes this a version of generalized expectation maximization algorithm

Variations can include fuzzy cluster assignment, methods to choose initial assignment, and artificial class reassignments to avoid local minima

K-means Algorithm

Problems: fixed number of clusters, spherical shape of clusters, same size of clusters



Gaussian based EM is more powerful algorithm as it uses not only means but also variances and covariances

Competitive Learning

Algorithm similar to k-means implemented with neural networks is called competitive learning

Main difference:

- k-means uses bath learning and all means are adjusted simultaneously,
- competitive neural net uses incremental learning and only the cluster center for a currently presented instance is adjusted

Advantage: does not alter the far away clusters

Disadvantage: no global optimization

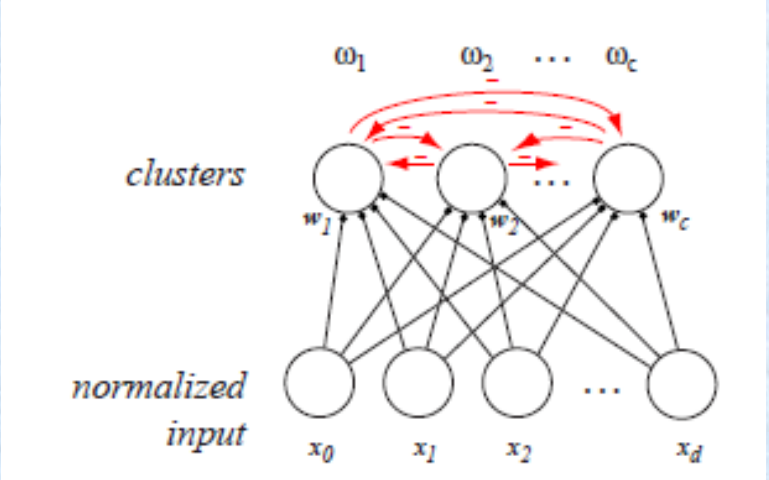
Competitive Learning

First layer has normalized representation of input pattern

Second layer has cluster nodes

Competition between these nodes ensures only one cluster as a winner (WTA) or more distributed representation

Only winning clusters update the weights



Weights follow the presented input pattern

$$\dot{w}_{ij} = \eta x_i y_j - \alpha y_j w_{ij}$$

or

$$\dot{w}_{ij} = \left(\eta x_i - \alpha w_{ij} \right) y_j$$

von der Malsburg (1973) Model

Model of the orientation selectivity in the visual cortex

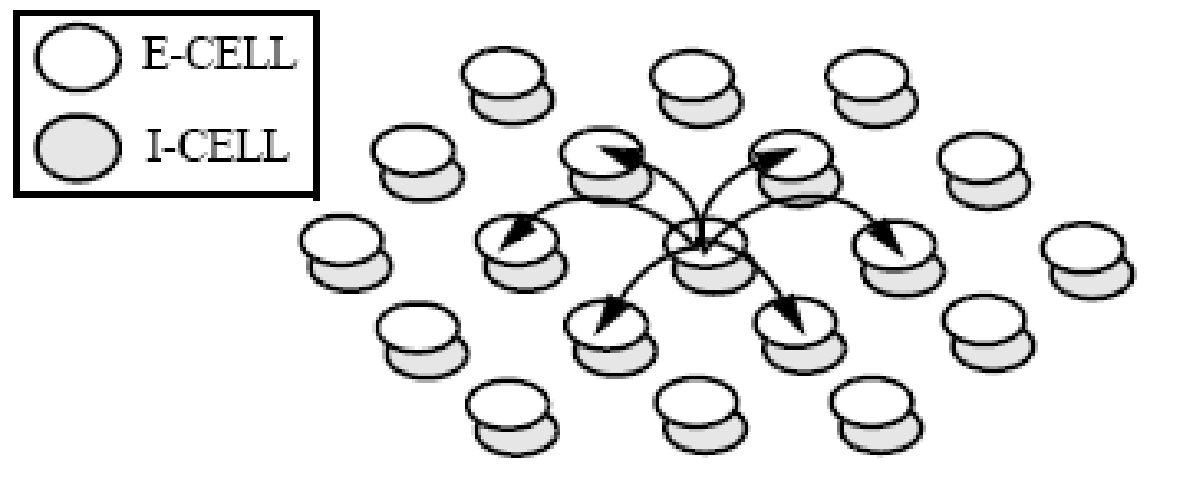
The model contained three types of horizontal interactions between cortical cells: E-cell to E-cell, E-cell to I-cell, and I-cell to E-cell

I-cells do not interact with each other

All connections within the cortical level are non-adaptive (fixed weight)

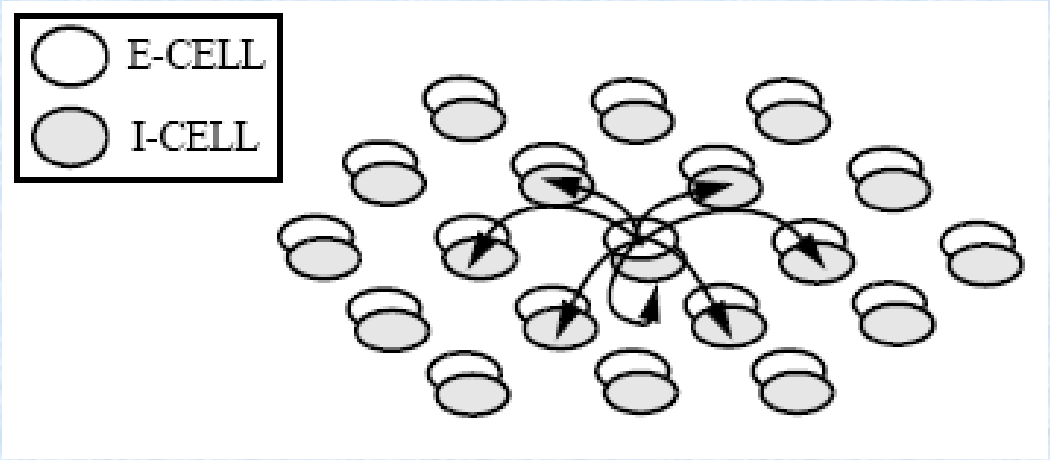
(1) E-cell to E-cell connections (p_{ij}) –

E-cells excite their immediate neighbors in the array



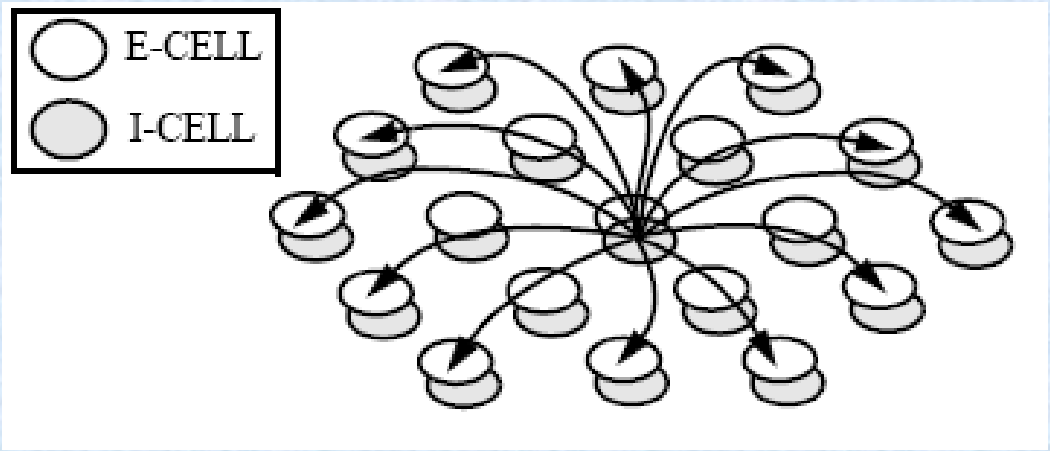
(2) E-cell to I-cell connections (r_{ij}) –

E-cells excite the corresponding I-cell and its immediate neighbors in the array

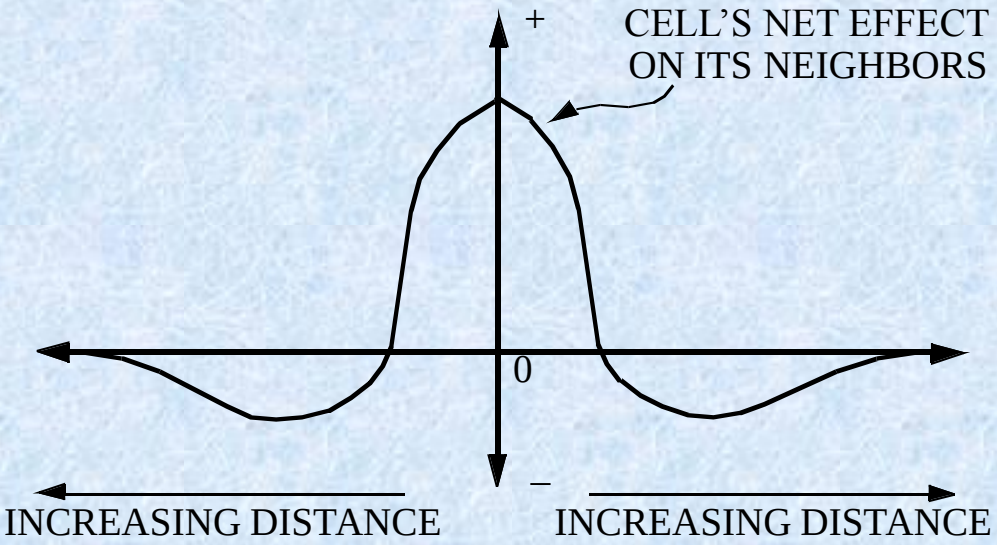


(3) I-cell to E-cell connections (q_{ij}) –

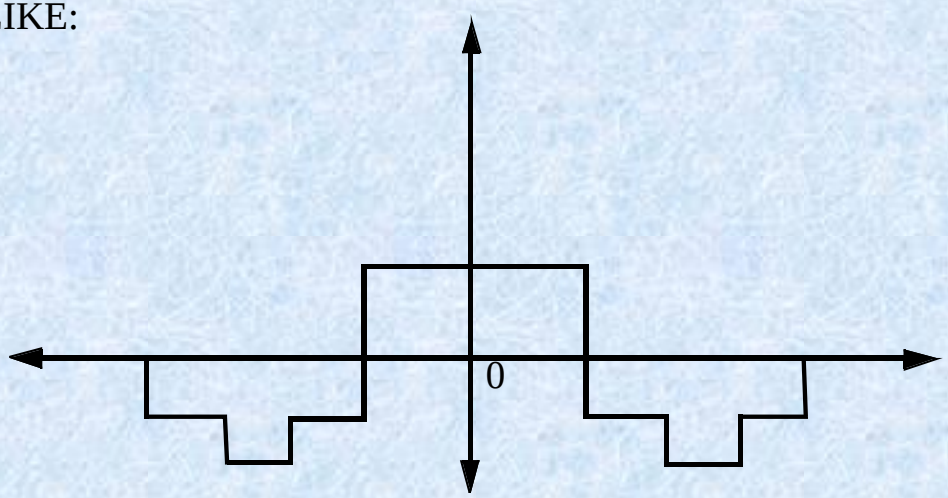
I-cells inhibit the next-to-immediate neighbor E-cells



These connections are meant to approximate a smoothly varying on-center, off-surround competitive network amongst the E-cells



MALSBURG IMPLEMENTATION IS ACTUALLY MORE LIKE:



Equations

The model has separate activation equations for the E-cells and I-cells due to the different connectivity of the two cell types (recall that I-cells receive no retinal input nor input from other I-cells)

E_i = ACTIVITY OF CORTICAL E-CELL

I_i = ACTIVITY OF CORTICAL I-CELL

R_i = ACTIVITY OF RETINAL CELL

s_{ij} = ADAPTIVE WEIGHT FROM RETINAL CELL i TO E-CELL j

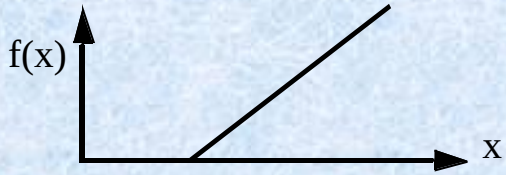
p_{ij} = FIXED WEIGHT FROM E-CELL i TO E-CELL j

q_{ij} = FIXED WEIGHT FROM I-CELL i TO E-CELL j

r_{ij} = FIXED WEIGHT FROM E-CELL i TO I-CELL j

a = DECAY RATE PARAMETER

f() = THRESHOLD LINEAR SIGNAL FUNCTION



$$\frac{dE_i}{dt} = -aE_i + \sum_j p_{ji} f(E_j) + \sum_k s_{ki} R_i - \sum_l q_{li} f(I_l)$$

The STM equations for the cortical cells are:

$$\frac{dI_i}{dt} = -aI_i + \sum_j r_{ji} f(E_j)$$

Weight Update Equations

Learning on each trial is done in two steps:

(1) Hebbian weight changes: $s'_{ij}(t+1) = s'_{ij}(t) + hR_iE_j$

Where *h* is a learning rate parameter.

(2) Weight normalization: $s_{ij} = \dfrac{s'_{ij}}{\sum_i s'_{ij}}$

This step can be thought of as maintaining a conservation of synaptic strength projecting to a cortical e-cell

von der Malsburg (1973) Model

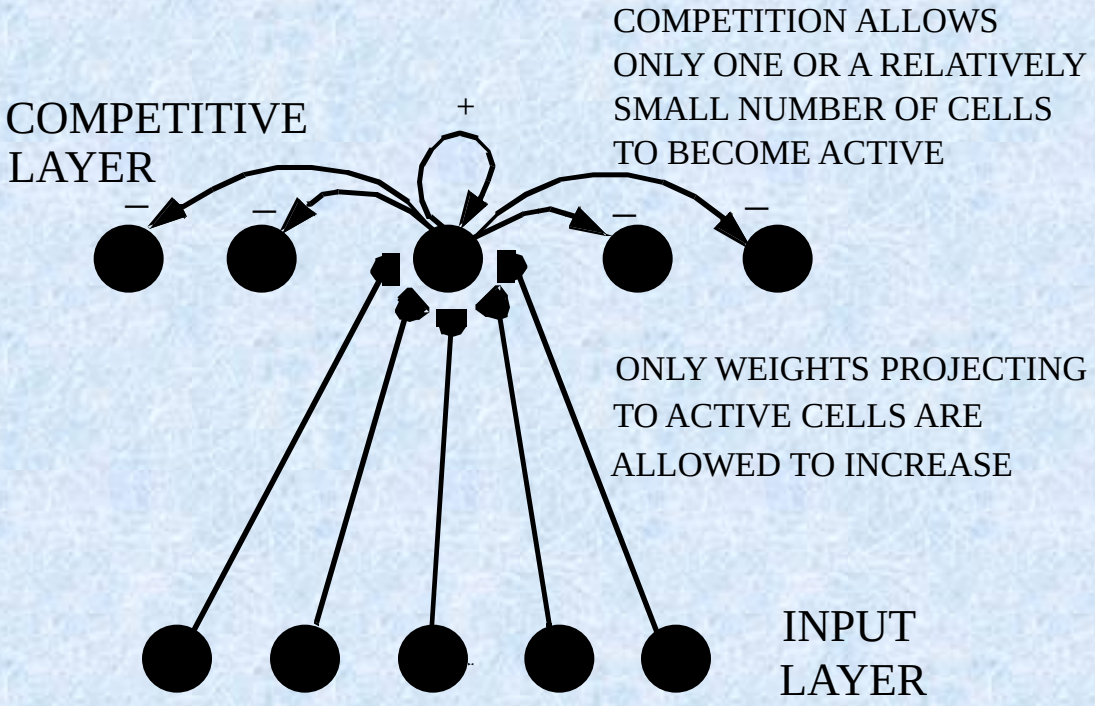
In von der Malsburg model the competition is distance dependent, as a result cluster nodes for similar patterns are closer together than cluster nodes for less similar patterns

This gives a smoothly varying fuzzy classification where class nodes are also clustered

In non-biological context this can allow building hierarchical clustering networks

The Network of *Grossberg (1976)*

Network is a standard competitive learning network in which the F2 layer is a shunting recurrent competitive field (RCF) with faster-than-linear signal function:



The Network of *Grossberg (1976)*

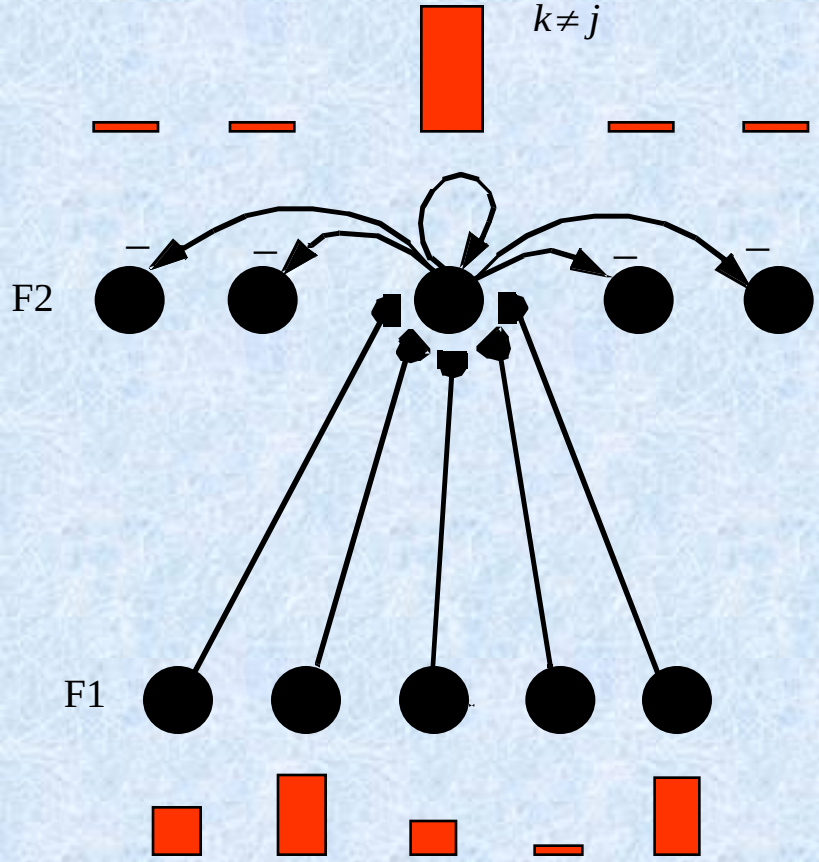
$$\frac{dy_j}{dt} = -Ay_j + \left(B - y_j\right)\left(f(y_j) + I_j\right) - \left(C + y_j\right)\sum_{k \neq j} f(y_k)$$

where $I_j = \sum_i x_i w_{ij}$

or $\vec{x} \cdot \vec{w}_j = \left\|\vec{x}\right\| \left\|\vec{w}_j\right\| \cos\theta$

So the winner will be the cell that has incoming weights closest to the current input pattern

(same is true for von der Malsburg network)



The Instar Learning Law

Grossberg (1976) studied the effects of using an "instar" learning law with Hebbian growth and post-synaptically gated decay in the F1 to F2 weights

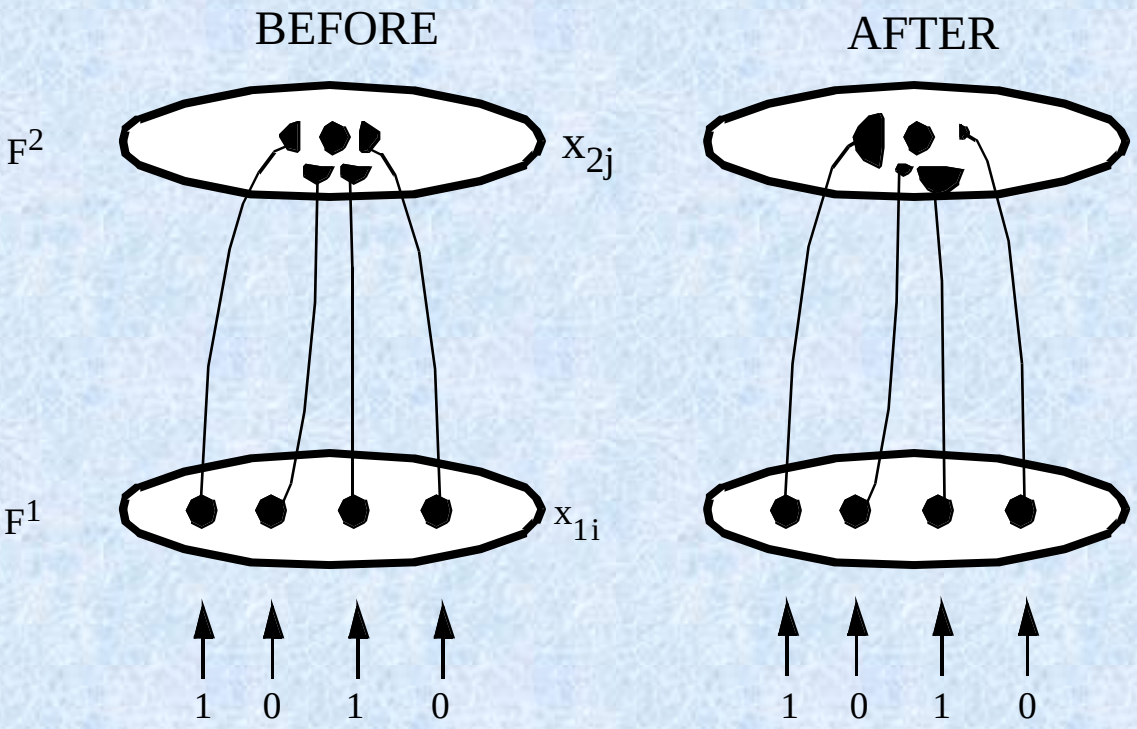
$$\dot{w}_{ij} = \eta x_i y_j - \alpha y_j w_{ij}$$

As F2 is winner-take-all, a substantial learning will only occur in the weight vector projecting to the F2 cell that won the competition (given that the rate of learning is slower then the rate of equilibration in F2)

What happens to the afferent weight vector for the F2 cell that won the competition?

$$\dot{w}_{ij} = \left(x_i - \frac{\alpha}{\eta} w_{ij} \right) \eta y_j$$

The weight vector becomes more like the current input vector



So the same cell is even more likely to win next time

Weight "Normalization" in *Grossberg (1976)*

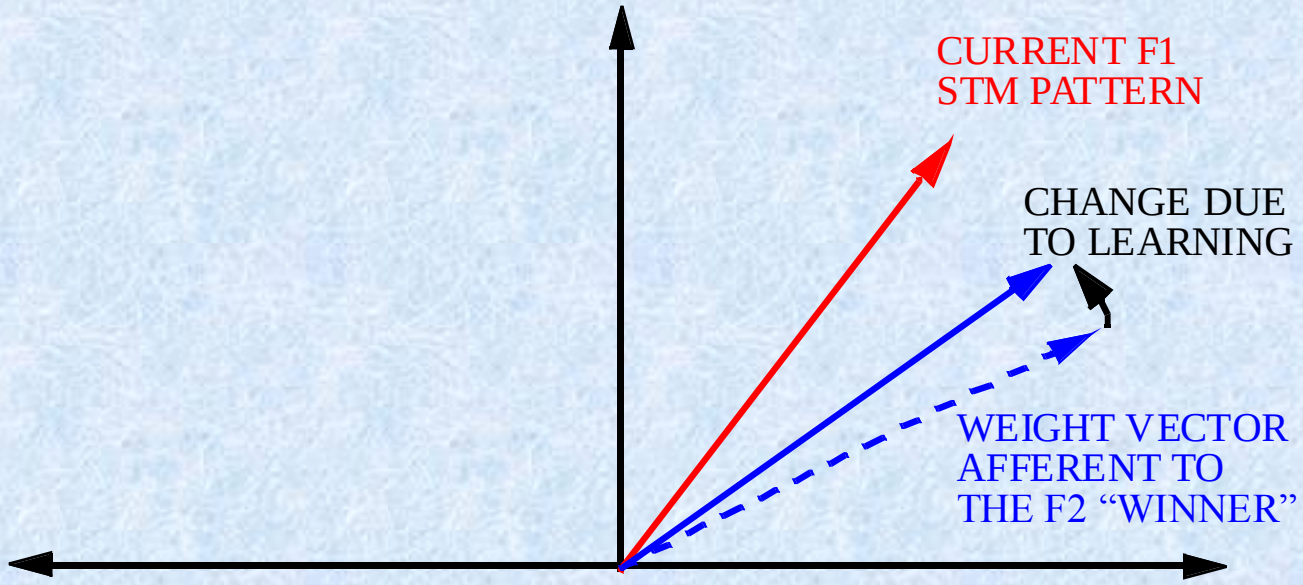
If the F1 layer is also a shunting competitive network with normalized activity, then the weight vector projecting to each F2 cell will be approximately normalized as well after learning has taken place

$$\dot{w}_{ij} = \left(x_i - \frac{\alpha}{\eta} w_{ij} \right) \eta y_j$$

Thus, each component of the F2 cell's weight vector tracks the corresponding component of the input vector at F1, and since the input vector is normalized, the weight vector becomes approximately normalized as well

Issue with Stability in SOM

That is, the afferent weight vector becomes more parallel to the current F1 STM pattern



Recall that we can set up a sequence of inputs so that the weight vector keeps rotating around and never stabilizes

Issues with Competitive Learning

Even with non-pathological data sequence it is possible that the weights will never stabilize, so the algorithm will never terminate

One approach is to reduce the learning rate, but then

- after some time we will not learn new patterns
- we will not be able to track changes through time

This leads to stability-plasticity dilemma

Another issue: we need to know the number of clusters (or its upper bound) in advance

- For batch learning we can set an optimization function based on the number of clusters
- For incremental learning we can dynamically recruit class nodes (but this adds order dependency)

Adaptive Resonance Theory (ART)

Grossberg, S. (1975). A neural model of attention, reinforcement, and discrimination learning. *Studies of Mind and Brain, Chapter 6.*

Employs many of the elements that are presented in the next (1976) paper in a more distilled way as ART

Primary motivation for ART: a need to stabilize bottom-up incremental learning

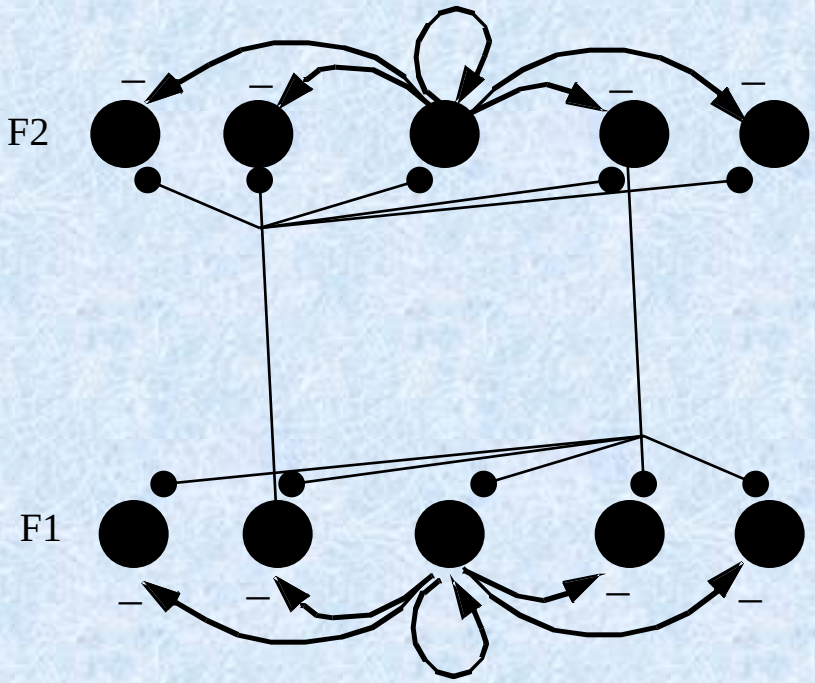
Suggested solution is an addition of top-down adaptive filter

Adaptive Resonance Theory (ART)

Now both F1 and F2 are RCFs, the network becomes almost symmetrical

But

- F1 has linear signal function (preserves the pattern but normalizes it) while F2 has faster than linear signal function (WTA)
- Bottom up weights use instar, while top down weights use outstar



$$\dot{w}_{ij} = \eta x_i y_j - \alpha y_j w_{ij}$$

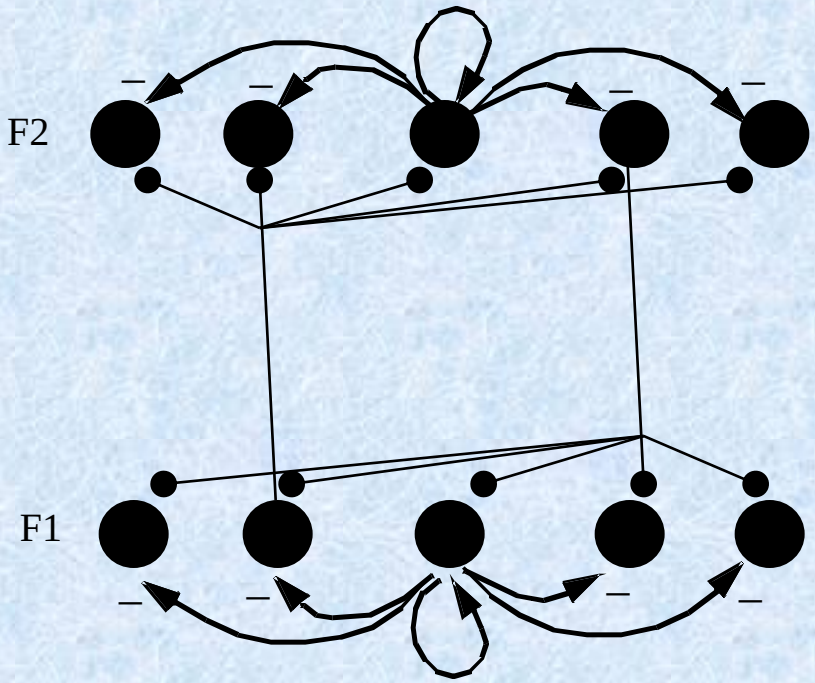
$$\dot{w}_{ji} = \eta x_i y_j - \alpha y_j w_{ji}$$

Adaptive Resonance Theory (ART)

The core idea is that a resonance between F1 and F2 differentiates between insignificant transients and significant states

Note that the paper talks about signal amplification during resonance, but this is not quite true due to normalizing properties of RCFs

Direct input to F2 can serve as top-down priming, decision, expectancy, etc



The core advantage of ART is active self-regulation of the learning process

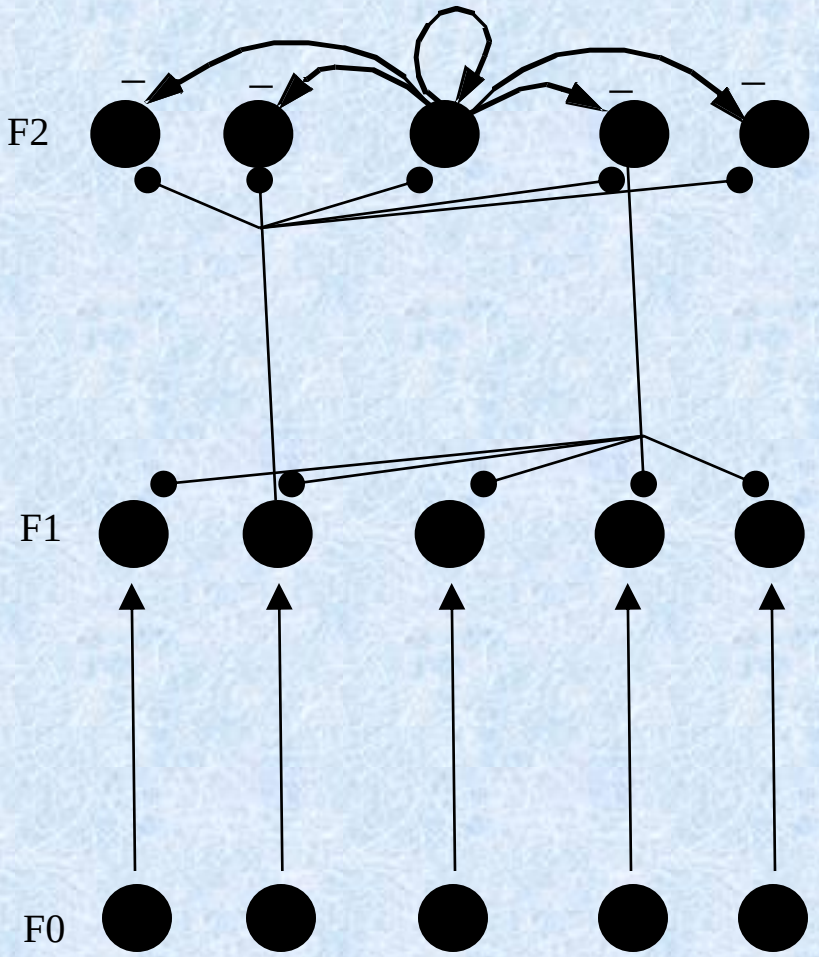
ART 1

Simple test architecture, binary inputs

How to preserve the input intact?

Split the matching layer F1 from the input layer F0

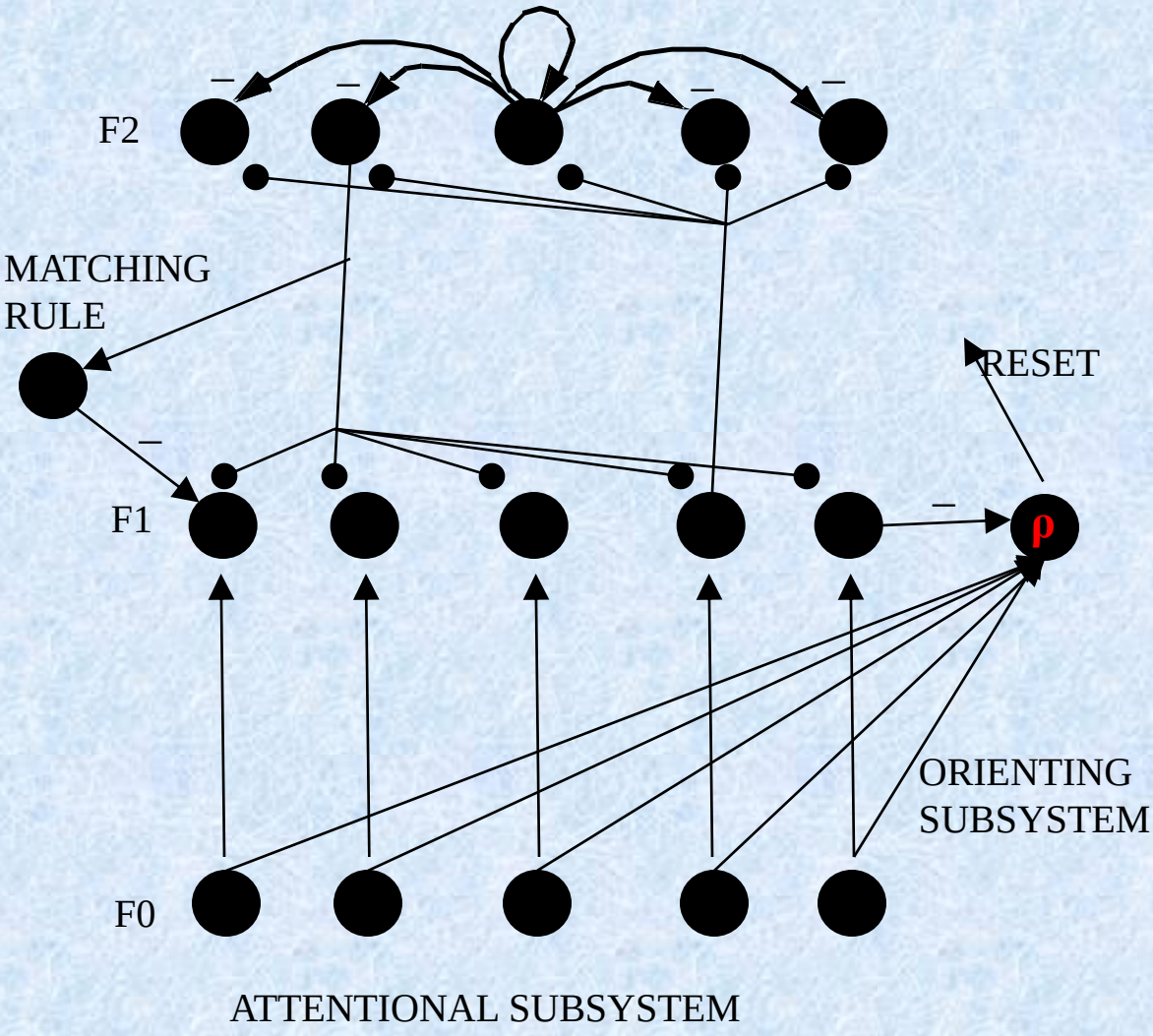
Now F0 and F2 have binary signals, while F1 can have continuous activation



ART 1

Orienting subsystem on the right compares the mismatch between F0 and F1

If it is greater than vigilance ρ then the F2 is reset



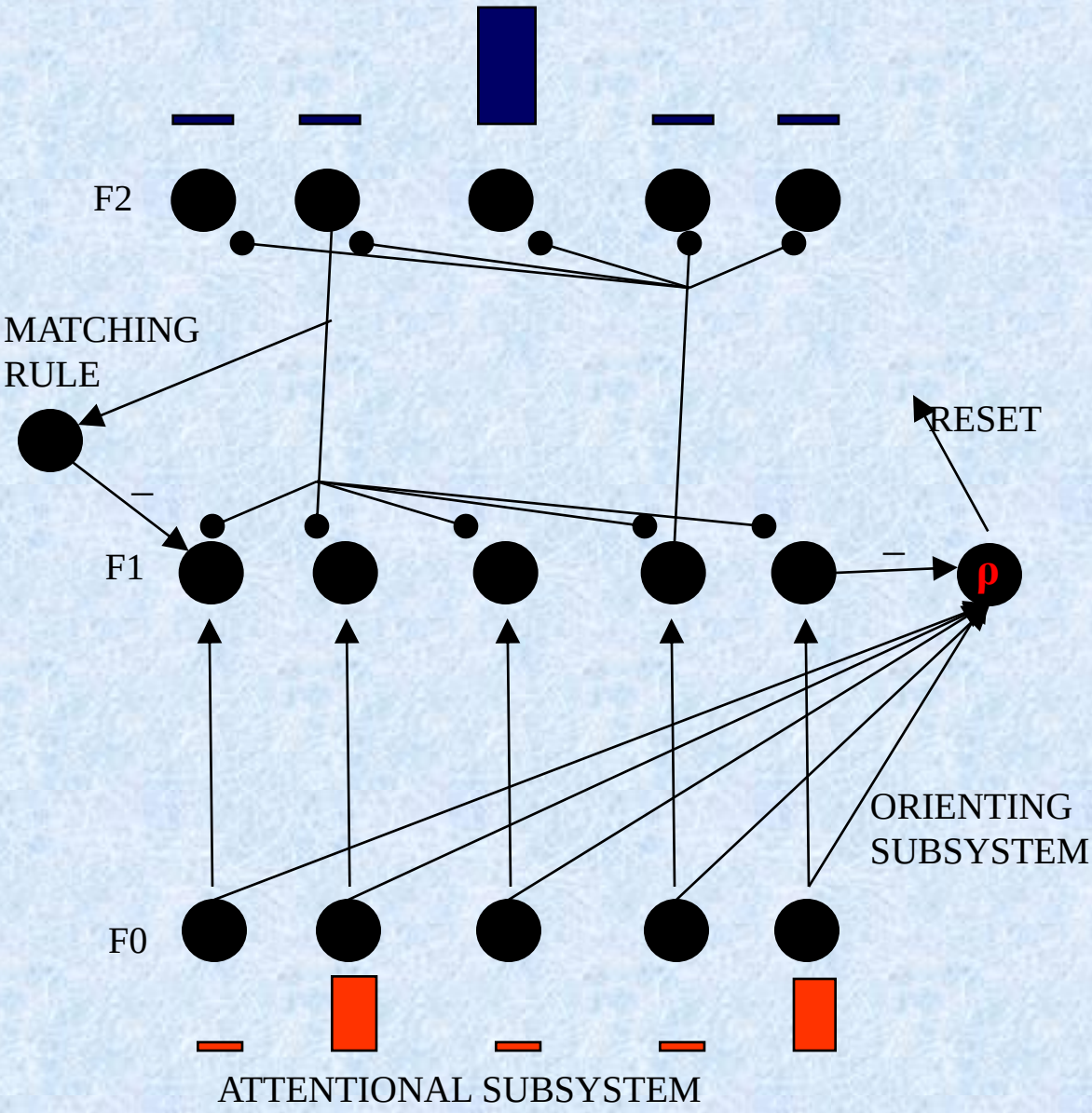
ART 1

The winner y_J favors overlapping w_{ij} and I

$$y_j = \frac{\left|\vec{x} \cap \vec{w}_j\right|}{\alpha + \left|\vec{w}_j\right|} \leq \frac{\left|\vec{w}_j\right|}{\alpha + \left|\vec{w}_j\right|}$$

To avoid a slow dynamics of RCF it is replaced by the *max* operator

$$y_J = \max y_j$$



ART 1

Due to the learning laws

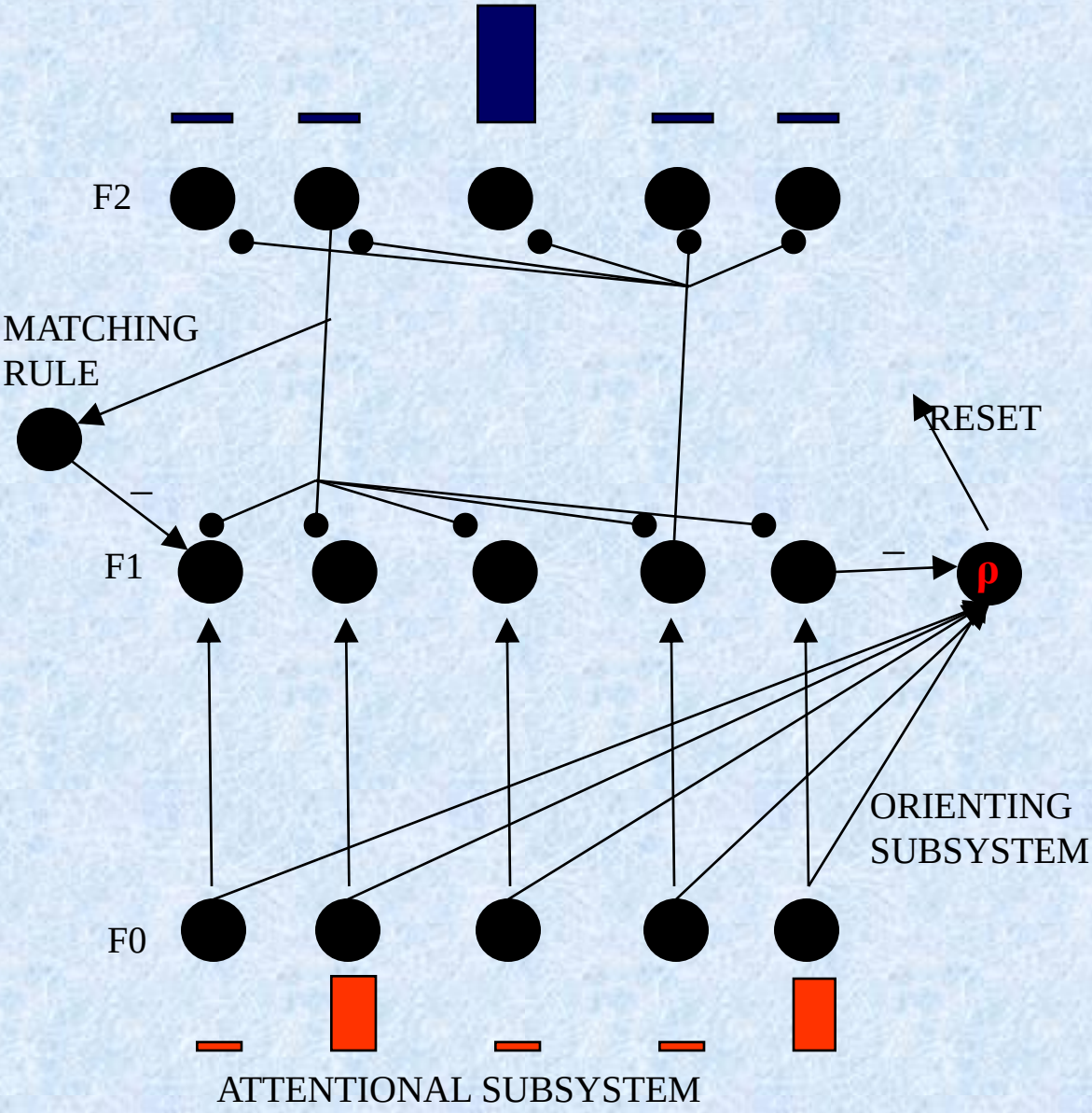
$$\dot{w}_{ij} = \eta x_i y_j - \alpha y_j w_{ij}$$

$$\dot{w}_{ji} = \eta x_i y_j - \alpha y_j w_{ji}$$

and identical initial weights

$$w_{ij} = w_{ji}$$

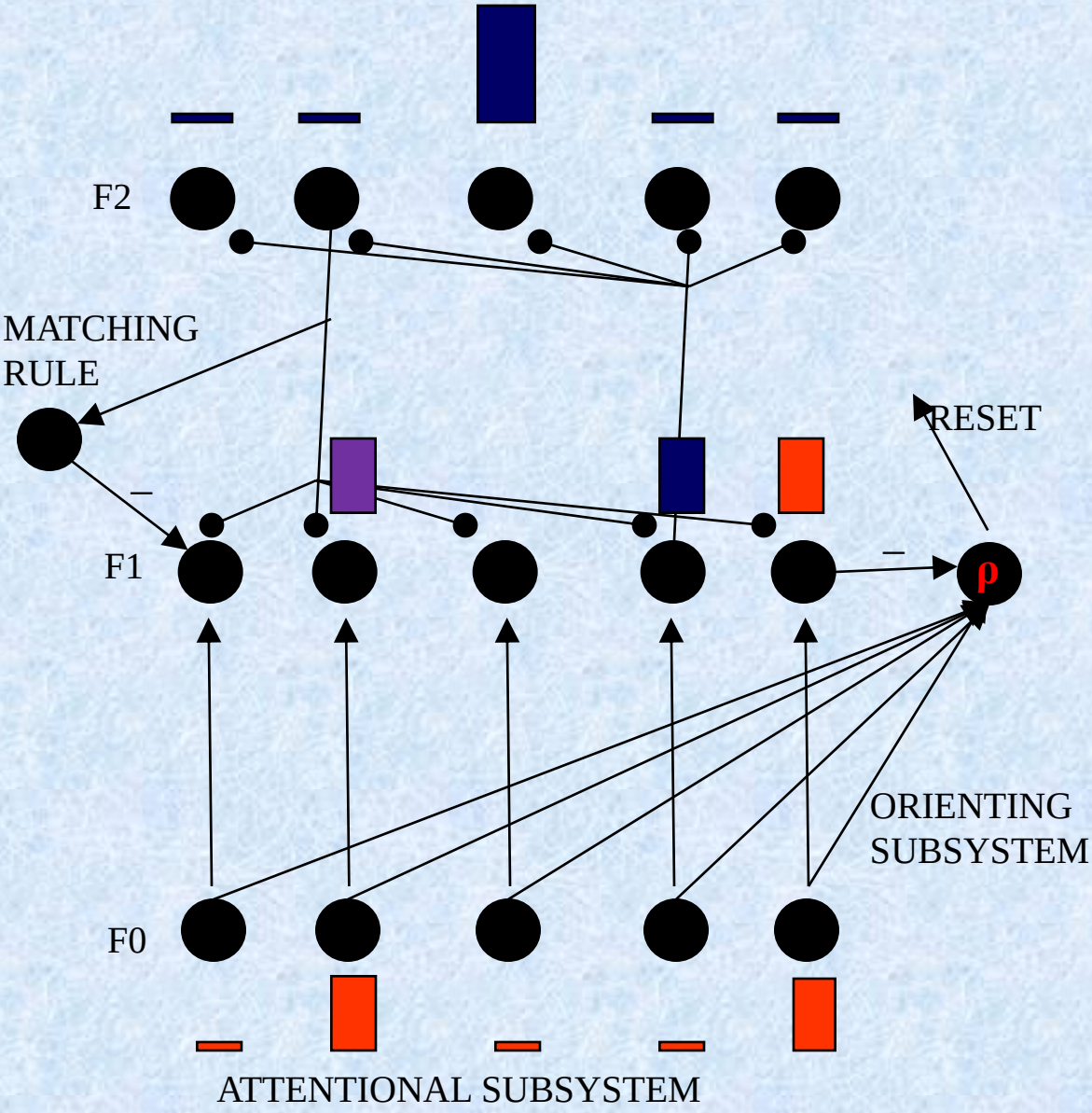
top-down and bottom-up weights stay equal: this saves memory and cost of computation



ART 1

Lets say we got a partial match (purple)

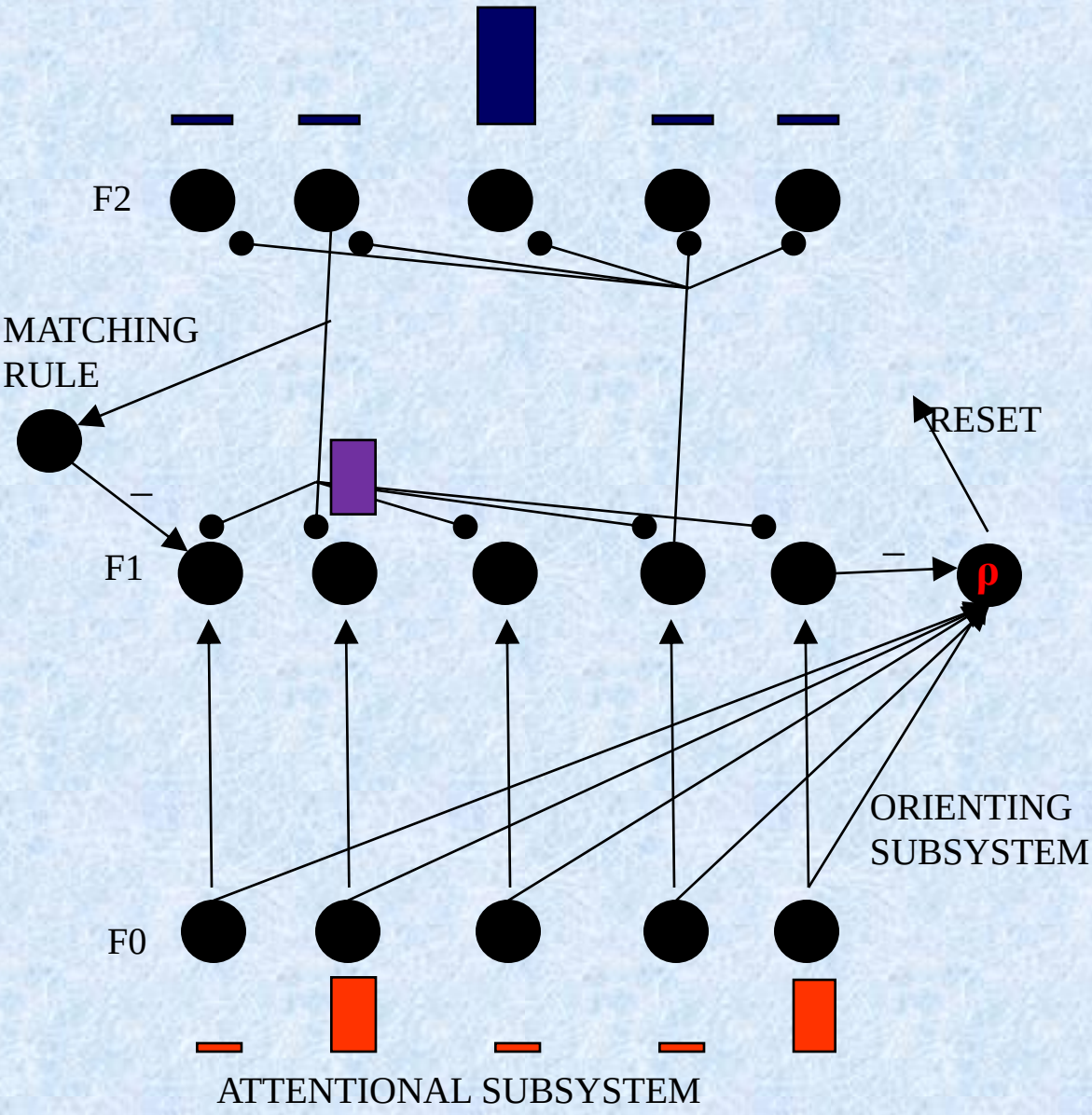
Without the matching rule weights will increase for purple and red component and decrease for blue component



ART 1

- The 2/3 matching rule adds TD and BU and subtracts 1 when TD is active
- So you have to have both TD and BU to get non-zero activity
- This way only purple component will learn
- That's given that we did pass the vigilance test

$$\frac{|\vec{x}|}{|\vec{I}|} \geq \rho$$



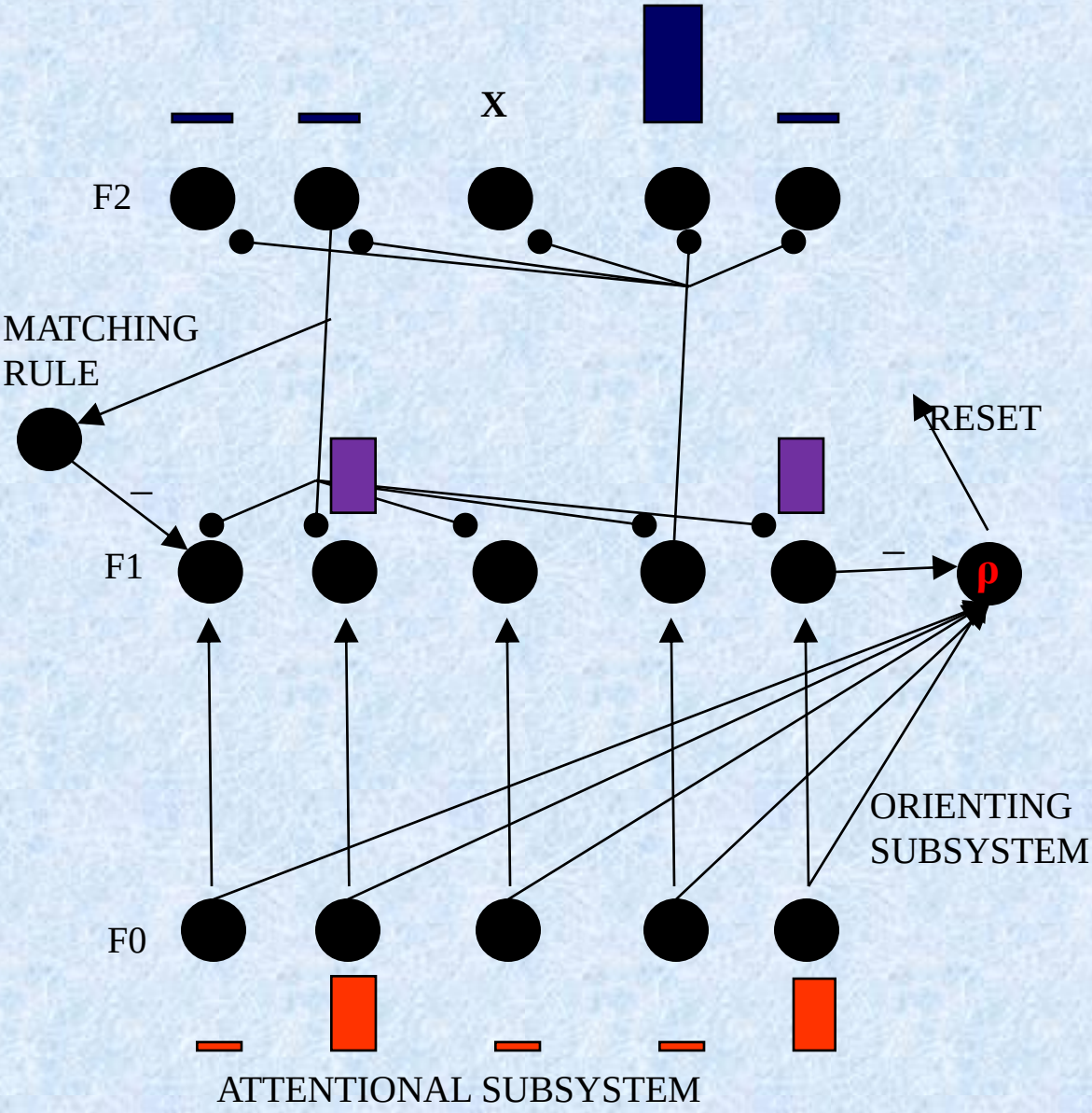
ART 1

If we fail the vigilance test $\frac{|\vec{x}|}{|\vec{I}|} < \rho$

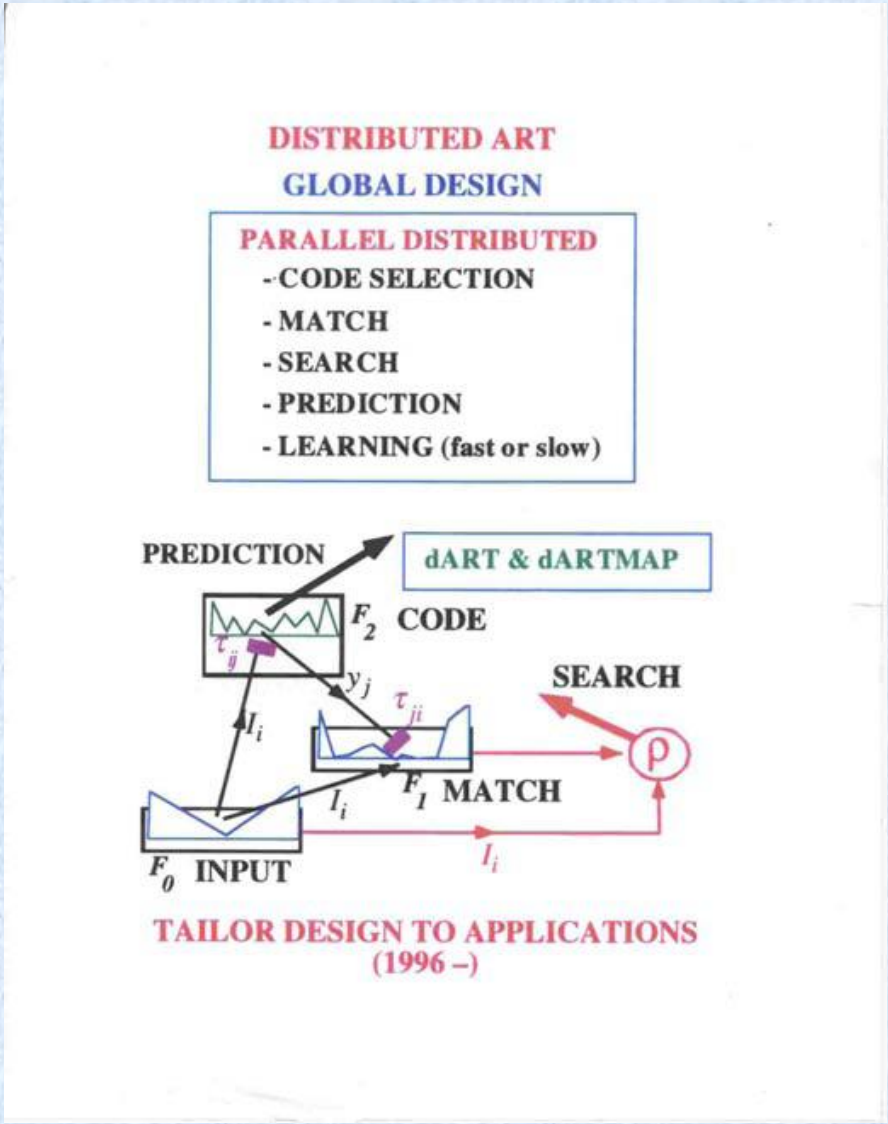
then reset happens

System keeps the list of unsuccessful winners for given input and removes them from competition

If none of F2 nodes is good enough then a new node is created



Distributed ART



Different architecture

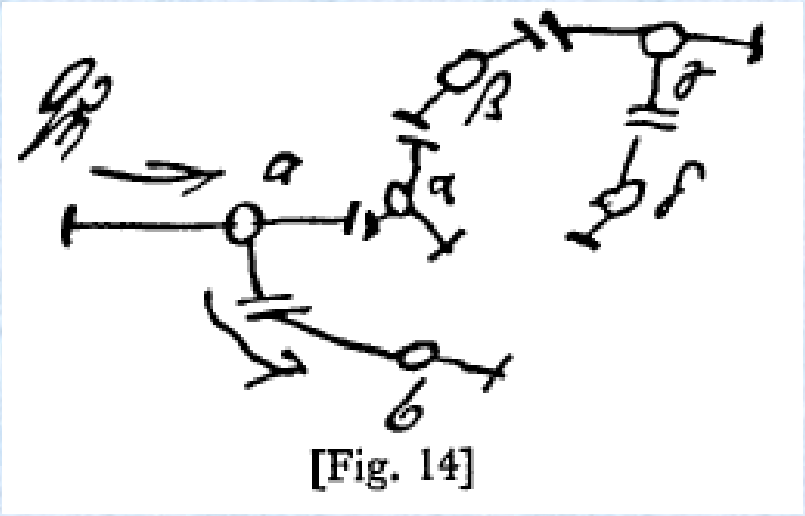
No actual resonance

Weights are subtractive rather then multiplicative

Careful! "Tailor design to applications" actually means "does not work in general cases"

Aside: there is also SMART by Versace & Grossberg, it restores the original biological concepts

How Sigmund Predicted Gail's Dreams



[Fig. 14]

Keep in mind that this is 50 years before McCullough & Pitts and only 25 years after the Golgi stain discovery, 11 years before Golgi and Cajal got Nobel prise

Talking about neurons and synapses

As well as attention, self-organization, normalization, excitation and inhibition

"keeping constant something in its functional relations that we might describe as sum of excitation"

"inhibition by the ego that makes … distinguishing between perception and memory"

How Sigmund Predicted Gail's Dreams

"Experience will teach that discharge not to be initiated till the indication of reality has arrived"