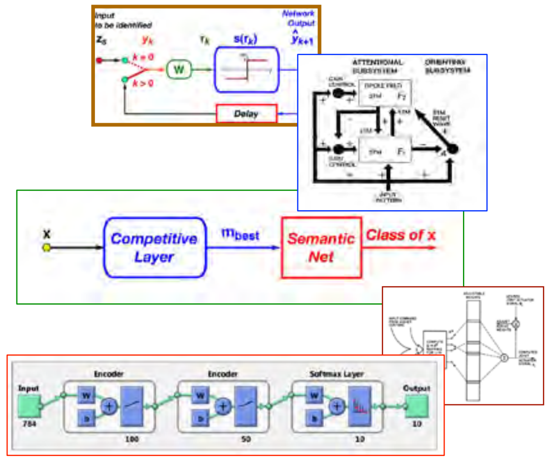
Neural Networks

Robert Stengel
Robotics and Intelligent Systems, MAE 345,
Princeton University, 2017

- Associative/recurrent networks
  - Hopfield network
  - Adaptive resonance theory network
  - Elman/Jordan networks
- Unsupervised training
  - k-means clustering
- Semi-supervised training
  - Self-organizing map
- Cerebellar model articulation controller (CMAC)
- Deep learning
  - Restricted Boltzmann machine
  - Convolutional network
  - Neural Turing Machines

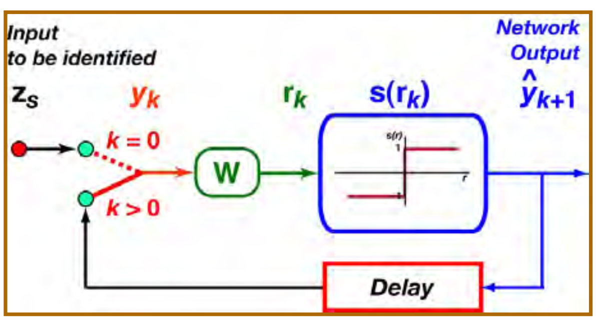


Associative/Recurrent Networks

Associative-Memory Neural Networks

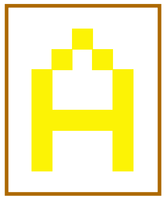
- **Goals**
  - Identify symbols from noisy data, given exemplars of possible features
  - Retrieve full feature from incomplete samples
    - "To be ______"
    - "Snap, crackle, ____"

  - Build a database from related contextual information, e.g., populate features of one categorical set using features in another

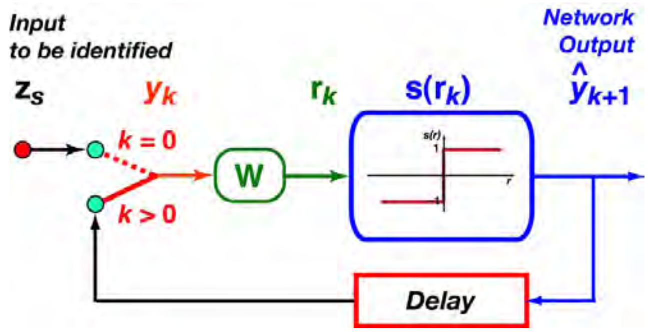


Recurrent Networks

- Recursion to identify an unknown object
  - Network is given a single, fixed input, and it iterates to a solution
- Convergence and stability of the network are critical issues *(discrete-time dynamic system)*
- Single network may have many stable states
  - Classified outputs of the map
  - Pattern recognition with noisy data



Hopfield Network



- Bipolar (–1,1) inputs and outputs
  - dim(y) = *n* x 1
- Supervised training with perfect exemplar outputs
- Noisy measurement of an exemplar as input to be identified
- Network operation

$$\mathbf{z}_s = \mathbf{y}_s + \mathbf{n}_s$$

$$\hat{\mathbf{y}}_0 = \mathbf{z}_s$$

$$\hat{\mathbf{y}}_{k+1} = \mathbf{s}(\mathbf{r}_k) = \mathbf{s}(\mathbf{W}\hat{\mathbf{y}}_k)$$

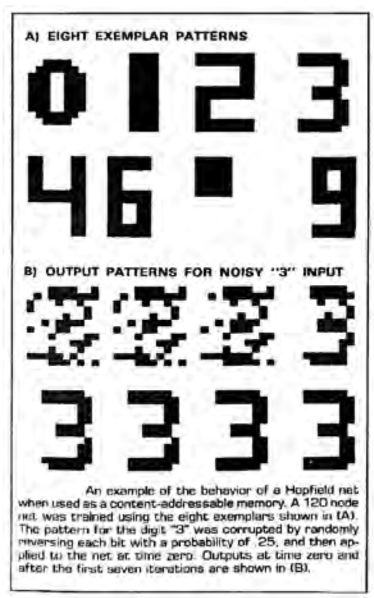
$$\hat{y}_{i_{k+1}} = \begin{cases} 1, & r_{i_k} > 0 \\ Unchanged\ , & r_{i_k} = 0 \quad , i = 1\ to\ n \\ -1, & r_{i_k} < 0 \end{cases}$$

- Iterate to convergence

Training a Hopfield Network

- Network training
  - Given *M* exemplars, $\mathbf{y}_s(n \times 1), \quad s = 1, M$
  - Each exemplar is a character represented by *n* pixels
  - Batch calculation of weighting matrix

$$\mathbf{W} = \sum_{s=1}^{M} \left(\mathbf{y}_s \mathbf{y}_s^T - \mathbf{I}_n \right) = \sum_{s=1}^{M} \begin{bmatrix} y_1^2 - 1 & y_1 y_2 & \ldots \\ y_1 y_2 & y_2^2 - 1 & \ldots \\ \ldots & \ldots & \ldots \end{bmatrix}_s$$



$$n = 120; \quad M = 8$$

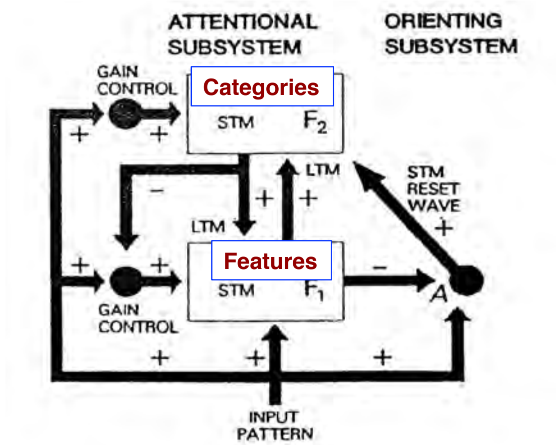
$$\#\,weights = n^2 = 14{,}400$$

- No iterations to define weights
- Large number of weights
- Limited number of exemplars (< 0.15 *n*)
- Similar exemplars pose a problem

Adaptive Resonance Theory Network

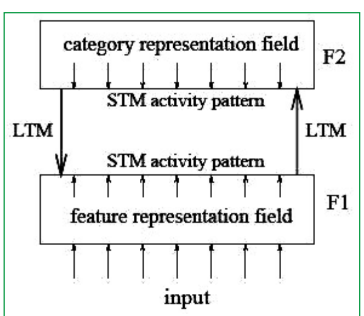
(Grossberg, Carpenter, 1976)

- **Self-organizing/stabilizing network for finding clusters in binary input (ART-1)**
- **Broadly based on cerebellar model**
  - **Long-Term Memory**
  - **Short-Term Memory**
  - **Stability and plasticity**
  - **Unsupervised and supervised learning**
  - **“Bottom-up” input**
  - **“Top-down” priming**
  - **Pre-cursor to “deep learning”**



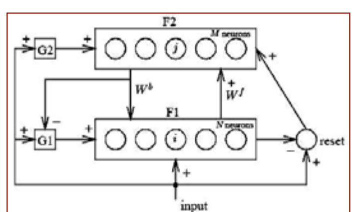
ART-1 Network

Architecture



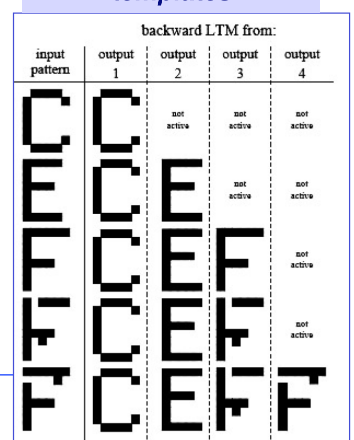
The ART architecture.

Binary Neurons represent Pattern Pixels



The ART1 neural network.

Recursive Training Example: adding new templates



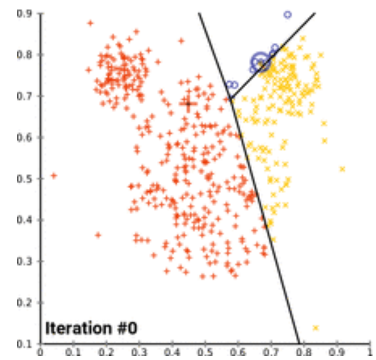
Further Developments:

- Continuous inputs (ART-2)
- Fuzzy Logic (Fuzzy ART)
- Dual-Associative Networks for Pattern Recognition (Lapart, Sandia, 2017)

k-Means Clustering

- **Least-squares clustering of *n* observation sets into *k* regions**

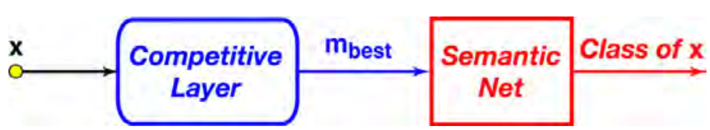
$$\min_{\mu_i} J = \sum_{i=1}^{k} \sum_{j=1}^{n} \left\| \mathbf{x}_j - \boldsymbol{\mu}_i \right\|_2$$



- **i.e., find *centroids* of each region**
- **Once centroids are known, boundaries of regions found from *Voronoi diagram***

Self-Organizing Map

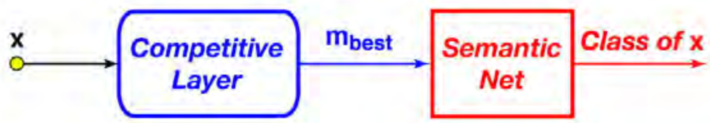
(Kohonen, 1981)



- **Competitive, unsupervised learning in 1st layer**
- **Premise: input signal patterns that are close produce outputs that are close**
- **Ordered inputs produce spatial distribution, i.e., a map**
- **Cells of the map are likened to the cell structure of the cerebral cortex**

> - **x: (n x 1) input vector characterizes features (attributes) of a signal**
> - **m: (n x 1) weight vector of a cell that represents an output class**

Competition in Self-Organizing Map



- Competition is based on minimizing distance from x to m

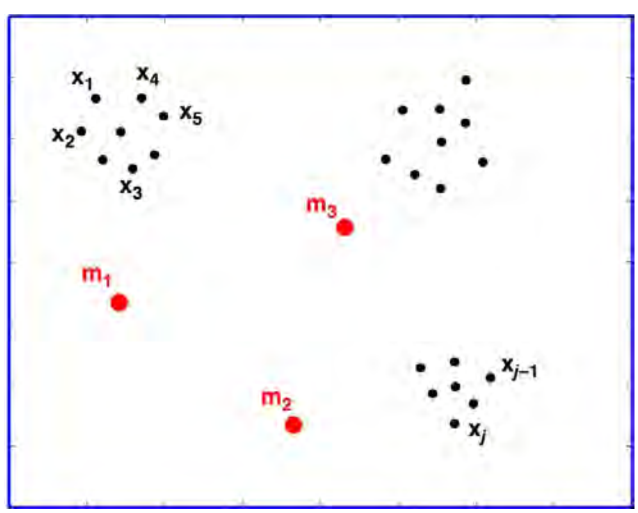
$$\boxed{\begin{array}{l} Cost = distance = \|\mathbf{x}-\mathbf{m}_i\| \\ \min Cost = \min_{\mathbf{m}_i}\|\mathbf{x}-\mathbf{m}\| \end{array}}$$

- m encodes the output classes
- Supervision: Semantic Net decodes the output to identify classes

$$\boxed{\mathbf{m}_1 = \begin{bmatrix} 0 \\ 1 \\ 3 \end{bmatrix} \rightarrow Class\ A; \quad \mathbf{m}_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \rightarrow Class\ B}$$

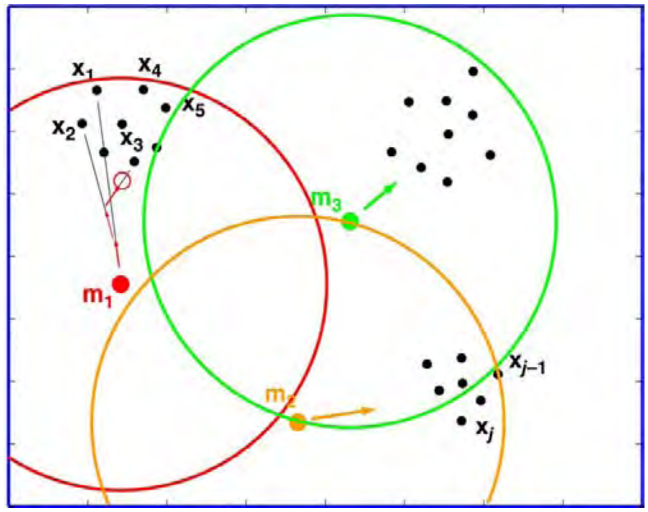
Goal of the Self-Organizing Map

- Given:
  - I output classes
  - Input training set, $\mathbf{x}_j$, $j = 1$ to J
- Find: Cell weights, $\mathbf{m}_i$, $i = 1$ to I that best cluster the data (*i.e., with minimum norm*)
- Initialize the cell weights, $\mathbf{m}_i$, randomly in the space of **x**



Training the Self-Organizing Map

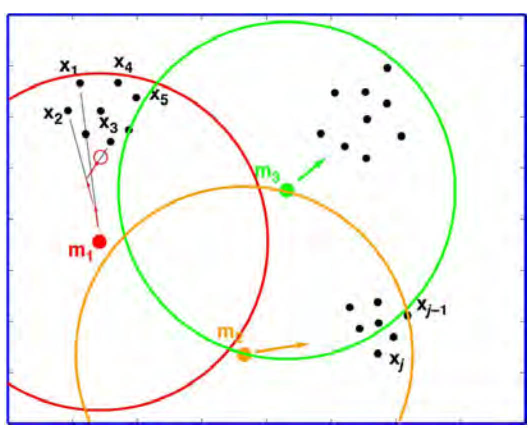
- **Define a neighborhood set within a radius of N_c around each cell, m_i**
  - **Choose N_c to overlap with neighboring cells**
- **Find the best cell-weight match, m_{best}, (i.e., the closest m_i) to the 1st training sample, x_1**



Cell Weight Updates

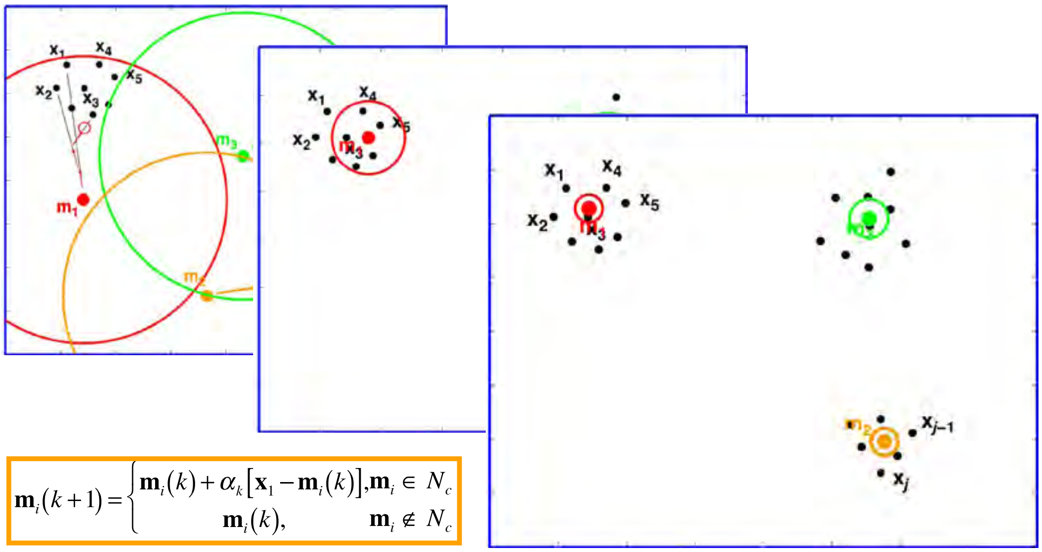
$$\boxed{\mathbf{m}_i(k+1) = \begin{cases} \mathbf{m}_i(k) + \alpha_k\left[\mathbf{x}_1 - \mathbf{m}_i(k)\right], & \mathbf{m}_i \in N_c \\ \mathbf{m}_i(k), & \mathbf{m}_i \notin N_c \end{cases}}$$

- **Update cell weights for all cells in the neighborhood set, N_c, of m_{best}**
  - **α_k = adaptation gain or learning rate**
- **Repeat for**
  - **x_2 to x_J**
  - **m_1 to m_I**
- **Converse of *particle swarm optimization***



Convergence of Cell Weights

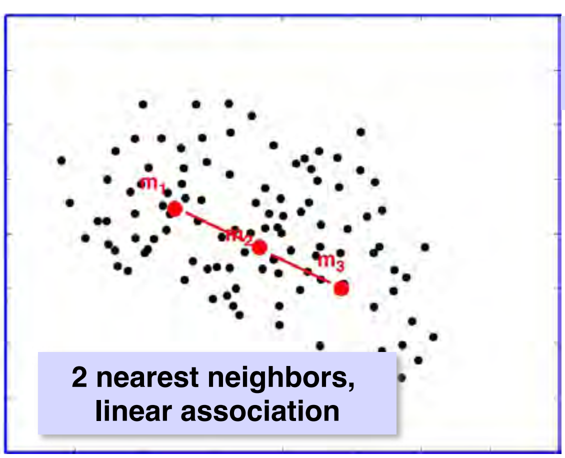
Repeat entire process with decreasing N_c radius until convergence occurs



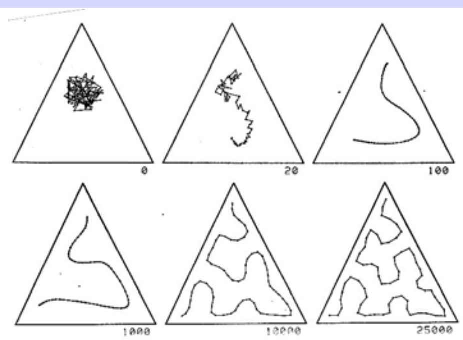
$$\mathbf{m}_i(k+1) = \begin{cases} \mathbf{m}_i(k) + \alpha_k\left[\mathbf{x}_1 - \mathbf{m}_i(k)\right], & \mathbf{m}_i \in N_c \\ \mathbf{m}_i(k), & \mathbf{m}_i \notin N_c \end{cases}$$

Semantic Map

- Association of m_{best} with categorical information
- Contextual information used to generate map of symbols
- Dimensionality and # of nearest neighbors affects final map
  - *Example*: linear association of cell weights
  - Points for cell-weight update chosen randomly

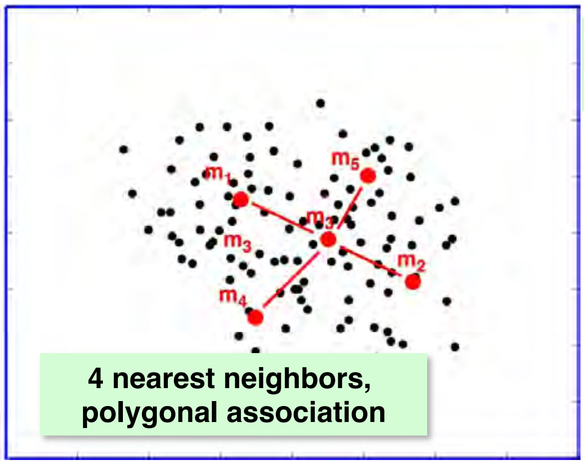


Evolution of points on a line that identifies locations of m_i
(Uniform random field of data points not shown)

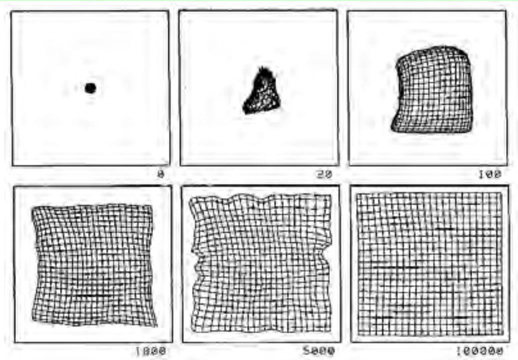


Choice of Neighborhood Architecture

- ***Example***: Map is assumed to represent a grid of associated points
- Number of cell weights specified
- Random starting locations for training



Evolution of grid points that identify locations of m_i
(Uniform random field of data points not shown)



Minimum Spanning Tree

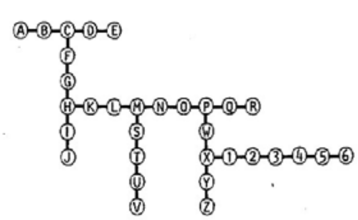
Example: Hexagonal map association identification
32 points with 5 attributes that may take six values (0, 1, 2, 3, 4, 5)

Input Data Matrix

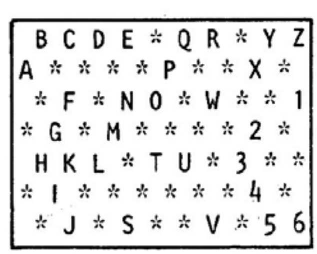
Attribute \ Item	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	1	2	3	4	5	6
a_1	1	2	3	4	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
a_2	0	0	0	0	0	1	2	3	4	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
a_3	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	3	3	3	3	6	6	6	6	6	6	6	6	6	6
a_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	1	2	3	4	2	2	2	2	2	2
a_5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6

Minimum spanning tree: smallest total edge length

Hexagonal lattice of grid points that identify locations of m_i



Minimal spanning tree corresponding to Table



Self-organized map of the data matrix of Table

Semantic Identification

Example of semantic identification
Each item for training has symbolic expression and context
Categories: noun, verb, adverb

(a)

Bob/Jim/Mary	1
horse/dog/cat	2
beer/water	3
meat/bread	4
runs/walks	5
works/speaks	6
visits/phones	7
buys/sells	8
likes/hates	9
drinks/eats	10/11
much/little	12
fast/slowly	13
often/seldom	14
well/poorly	15

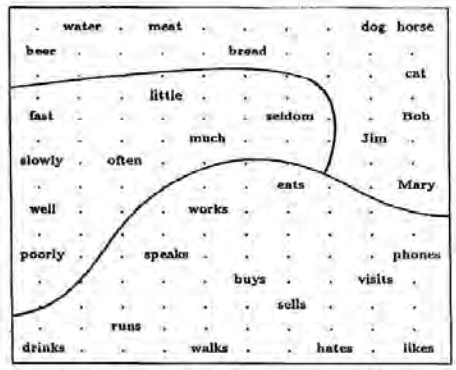
(b)

Sentence Patterns:		
1-5-12	1-9-2	2-5-14
1-5-13	1-9-3	2-9-1
1-5-14	1-9-4	2-9-2
1-6-12	1-10-3	2-9-3
1-6-13	1-11-4	2-9-4
1-6-14	1-10-12	2-10-3
1-6-15	1-10-13	2-10-12
1-7-14	1-10-14	2-10-13
1-8-12	1-11-12	2-10-14
1-8-2	1-11-13	2-11-4
1-8-3	1-11-14	2-11-12
1-8-4	2-5-12	2-11-13
1-9-1	2-5-13	2-11-14

(c)

Mary likes meat
Jim speaks well
Mary likes Jim
Jim eats often
Mary buys meat
dog drinks fast
horse hates meat
Jim eats seldom
Bob buys meat
cat walks slowly
Jim eats bread
cat hates Jim
Bob sells beer
(etc.)

Outline of vocabulary used in this experiment. (a) List of used words (nouns, verbs, and adverbs), (b) sentence patterns, and (c) some examples of generated three-word-sentences.



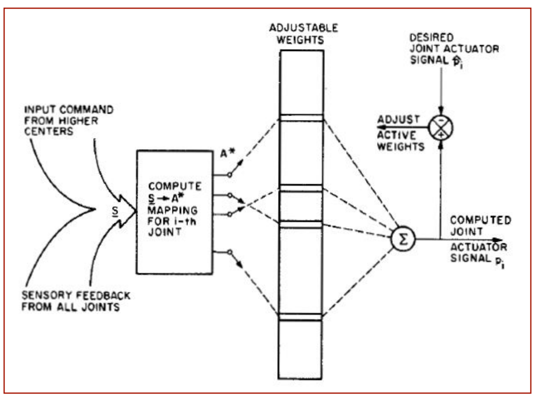
"Semantic map" obtained on a network of 10 × 15 cells after 2000 presentations of word-context-pairs derived from 10 000 random sentences of the kind shown in Fig. 10(c). Nouns, verbs, and adverbs are segregated into different domains. Within each domain a further grouping according to aspects of meaning is discernible.

Ritter, Kohonen, 1989

Cerebellar Model Articulation Controller (CMAC)

- **Another precursor to deep learning**
- **Inspired by models of human cerebellum**
- **CMAC: Two-stage mapping of a vector input to a scalar output**
- **First mapping: Input space to association space**
  - ***s* is fixed**
  - **a is binary**
- **Second mapping: Association space to output space**
  - ***g* contains learned weights**

Albus, 1975



$$s : x \rightarrow \mathbf{a}$$
$$Input \rightarrow Selector\ vector$$

$$g : \mathbf{a} \rightarrow y$$
$$Selector\ vector \rightarrow Output$$

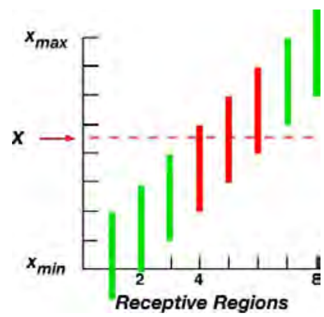
Example of Single-Input CMAC Association Space

$$\boxed{\begin{array}{c} s : x \rightarrow \mathbf{a} \\ Input \rightarrow Selector\ vector \end{array}}$$

- x is in (x_{min}, x_{max})
- Selector vector, **a**, is binary and has N elements
- Input quantization = $(x_{max} - x_{min}) / N$
- Receptive regions of association space map x to **a**
  - Analogous to neurons that "fire" in response to stimulus
- N_A = Number of receptive regions

$$N_A = N + C - 1 = \dim(\mathbf{a})$$

- C = Generalization parameter = # of overlapping regions

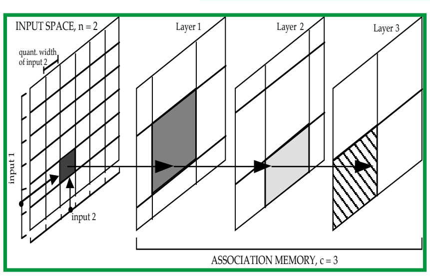


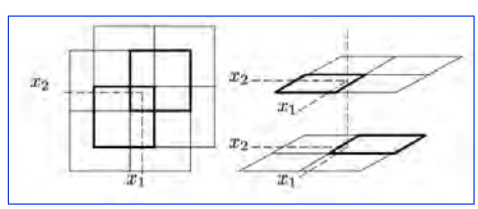
$$\mathbf{a} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}^T$$
$$C = 3$$

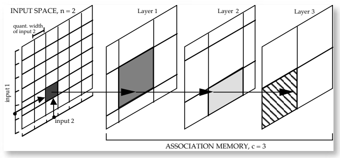
CMAC Output and Training

- In higher dimensions, association space is dim(x), a plane, cube, or hypercube
- Potentially large memory requirements
- Granularity (quantization) of output
- Variable generalization and granularity

2-dimensional association space
Rectangular receptive regions







CMAC Output and Training

- **CMAC output, *y*, (*i.e.*, control command) from activated cells of *c* Associative Memory layers**

$$y_{CMAC} = \mathbf{w}^T \mathbf{a} = \sum_{i=j}^{j+C-1} w_{i,activated}$$

j= index of first activated region

- **Least-squares training of CMAC weights, w**
  - Analogous to synapses between neurons

$$w_{j_{new}} = w_{j_{old}} + \frac{\beta}{c}\left(y_{desired} - \sum_{i=1}^{c} w_{i_{old}} \right)$$

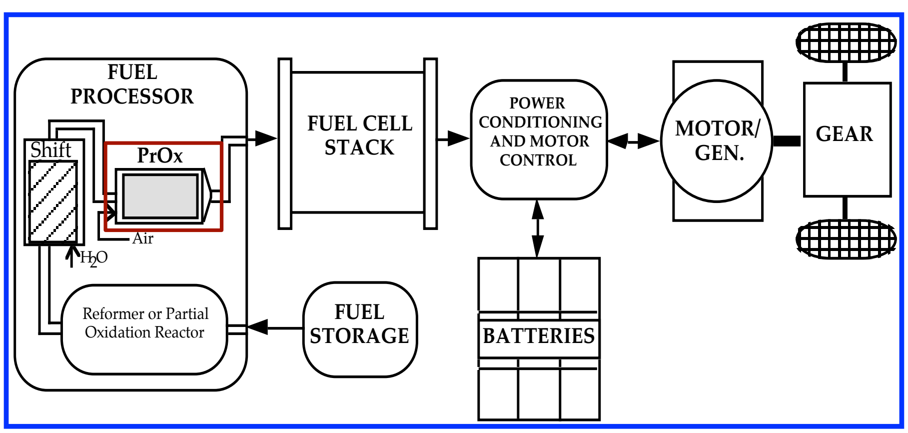
β is the learning rate and w_j is an activated cell weight

- **Localized generalization and training**

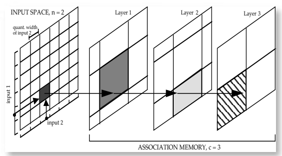
CMAC Control of a Fuel-Cell Pre-Processor

(Iwan and Stengel)

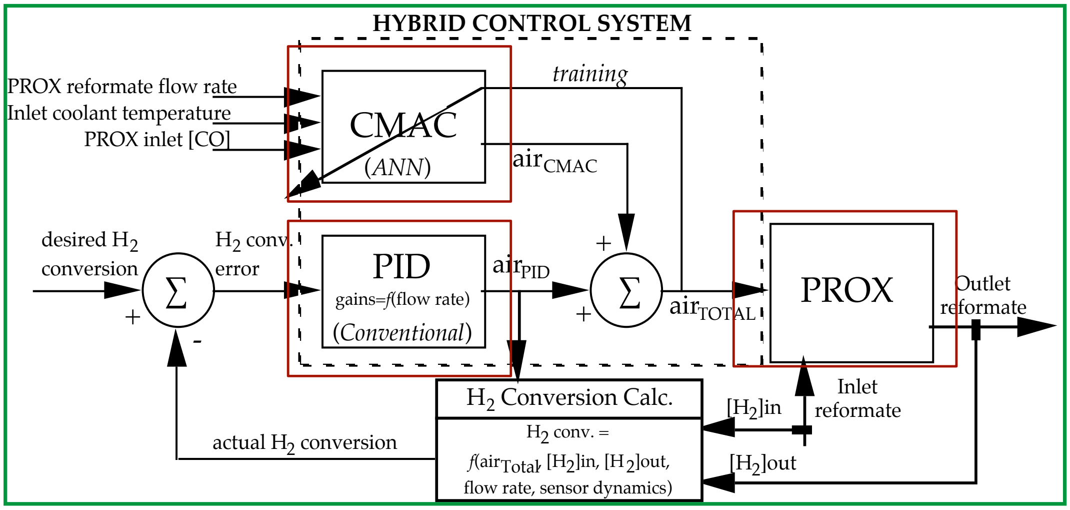
Fuel cell produces electricity for electric motor

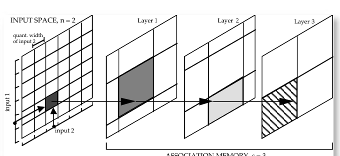


Pre-processor produces hydrogen for the fuel cell <u>and</u> carbon monoxide, which "poisons" the fuel cell catalyst



CMAC/PID Control System for Preferential Oxidizer



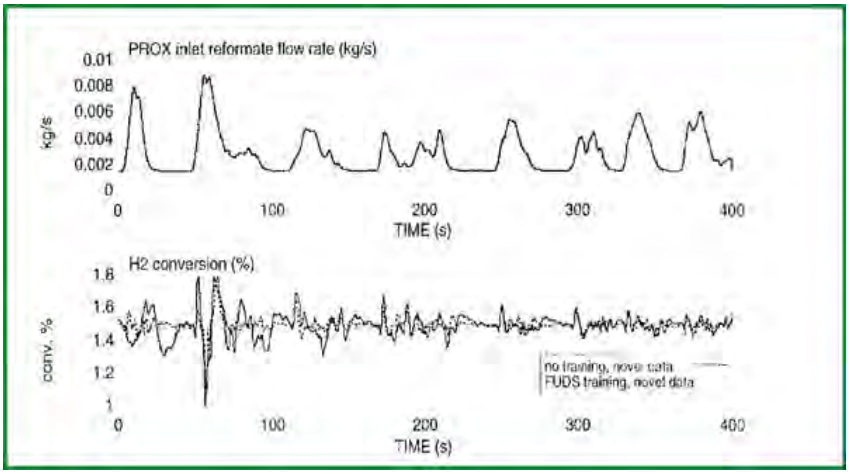


Summary of 3-D CMAC Characteristics

- **Inputs and Number of Divisions for receptor cubes:**
  - **PrOx inlet reformate flow rate (95)**
  - **PrOx inlet cooling temperature (80)**
  - **PrOx inlet CO concentration (100)**
- **Output: PrOx air injection rate**
- **Associative Layers, *C*: 24**
- **Number of Associative Memory Cells/Weights and Layer Offsets: 1,276 and [1,5,7]**
- **Learning Rate, : ~0.01**
- **Sampling Interval: 100 ms**

Flow Rate and Hydrogen Conversion of CMAC/PID Controller

- H_2 conversion command (across PrOx only): 1.5%
- Novel data, with (---) and without pre-training (—)
- Federal Urban Driving Cycle (= FUDS)

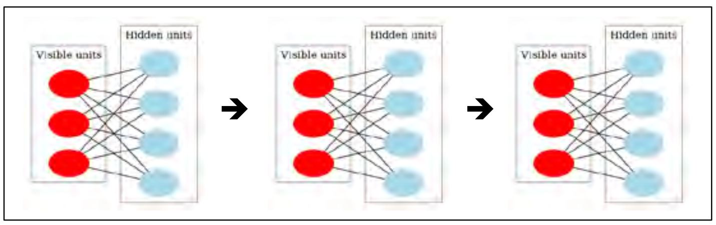


Comparison of PrOx Controllers on Federal Urban Driving Cycle

	mean H_2 error	maximum H_2 error	mean CO out	max. CO out	net H_2 output
	%	%	ppm	ppm	%
• Fixed-Air	0.68	0.87	6.3	28	57.2
• Table Look-up	0.13	1.43	6.5	26	57.8
• PID	0.05	0.51	7.7	30	58.1
• CMAC/PID	0.02	0.16	7.3	26	58.1

Deep Learning with Restricted Boltzmann Machine

- Multiiple layers of RBMs
- Semi-supervised learning
  - Clustering (visible) units
  - Sigmoid (hidden) units
- Pre-train each layer separately and contextually (*unsupervised*)
- Fine-tune with backpropagation (*supervised*)
- Restrict connections between layers
- Goal is to overcome *"vanishing or exploding gradient problem"* in multi-layer back-propagation

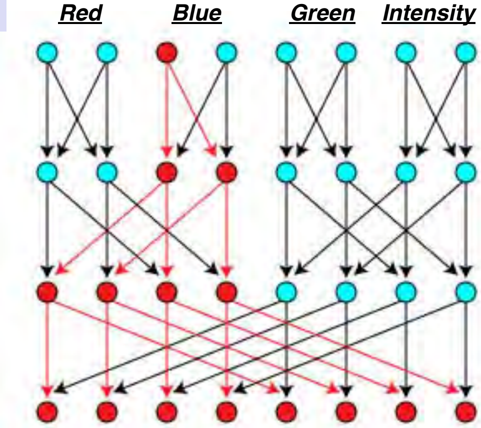


Hinton et al, 2006

Sparse Deep Network

- Partitioned input space
- Expanding network connections

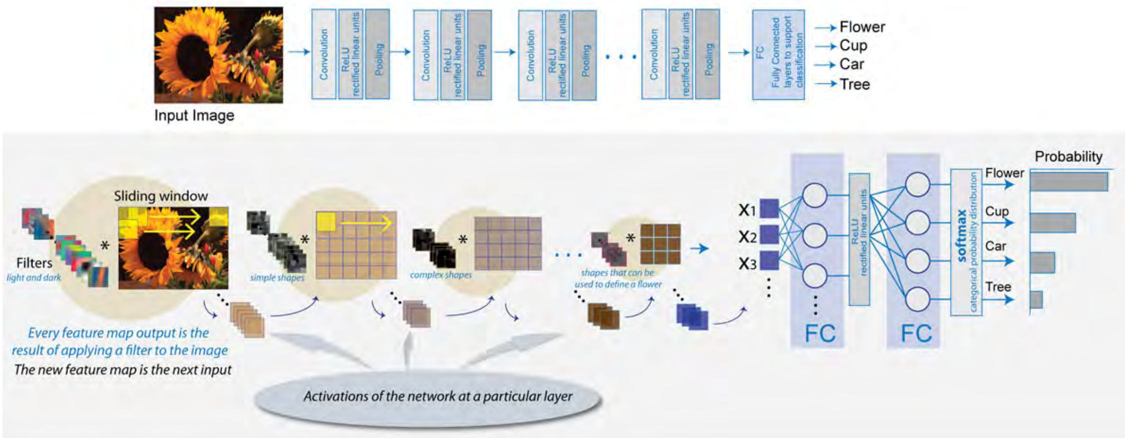
Filtered Input <u>Red</u> <u>Blue</u> <u>Green</u> <u>Intensity</u>



Analyzed Image

- Fully connected final layer

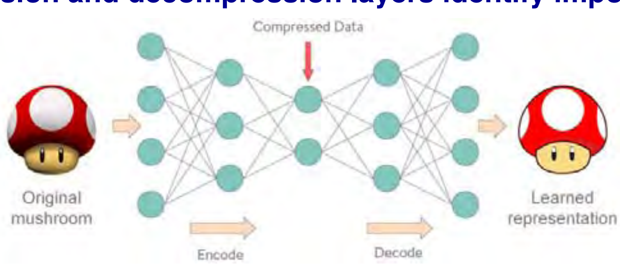
Convolutional Neural Network



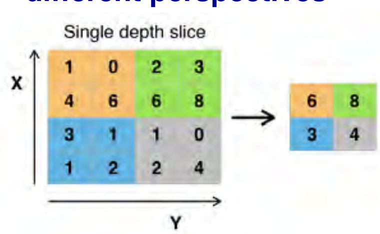
- **Decomposition of image**
  - Sliding window of receptive fields
  - Pooling (dimension reduction)
  - Simply connected networks
- **Repeated sequence of operations**
  - Convolution (cross-correlation)
  - Rectification neurons (ReLu)
  - Fully connected networks

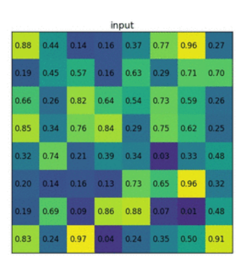
Autoencoding and Pooling

- Autoencoding: Same number of inputs and outputs
- Compression and decompression layers identify important attributes



- Max pooling: selection of important attributes
- Dimension reduction of features
- Enhanced invariance in characterization of a feature in different perspectives





Convolution

- Cross-correlation of outputs from previous layers
- Apply to partitioned receptive fields

"Heat Map"

$$\mathbf{C}_S = \mathbf{D}^T\mathbf{D}$$

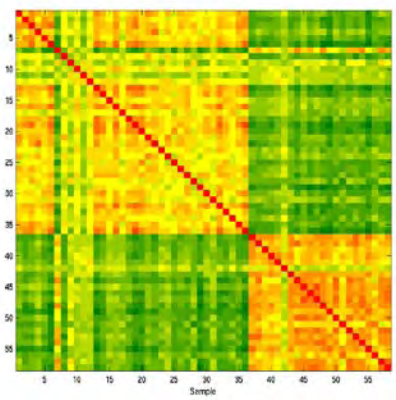
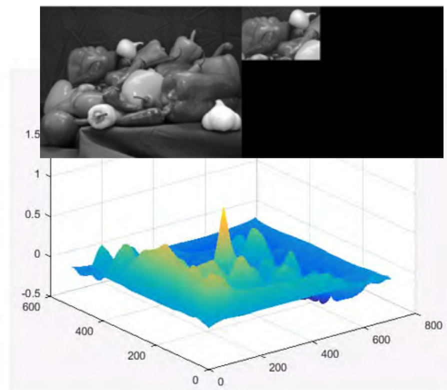


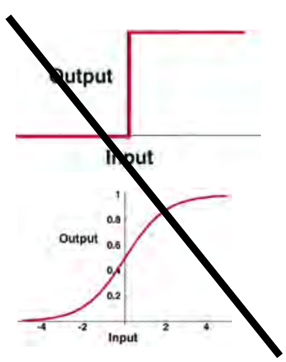
Image Convolution

$$z_k\left(x_k, y_k\right) = \sum_{i=0}^{K} c_k\left[x_k y_{k+1}\right]$$

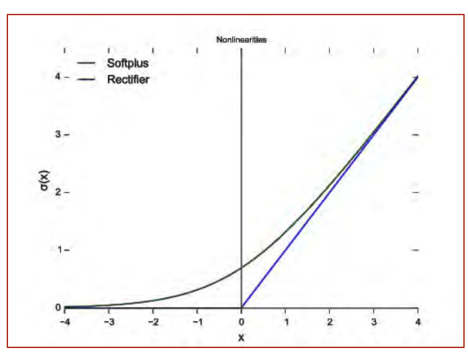


Rectified Linear Unit (ReLu)

- Simple alternative to hardlim, sigmoid nodes $\boxed{y = \max(0, y)}$
- Faster, more accurate classification in some applications



➔

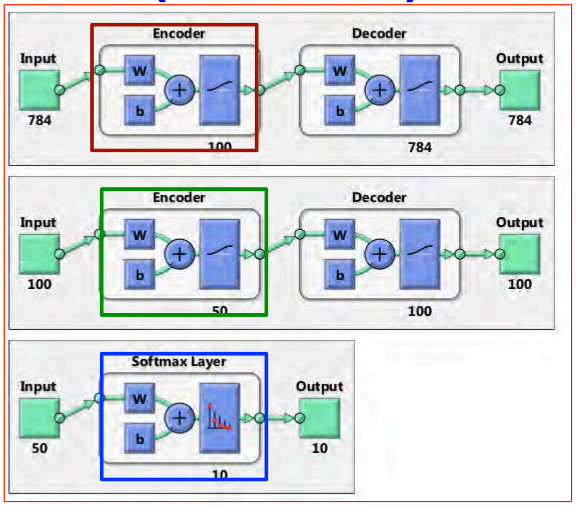


Convolution Neural Network (ConvNet)

1^{st} Encoder Unsupervised

2^{nd} Encoder Unsupervised

Preliminary NN training Supervised

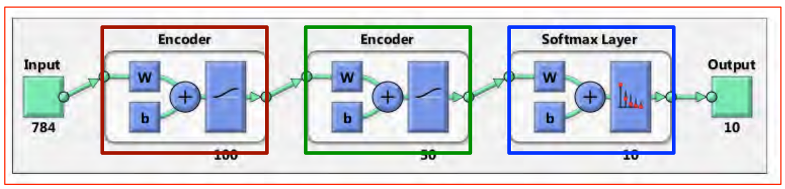


Output matched to input (1)

Output matched to input (2)

Output (2) trained for classification

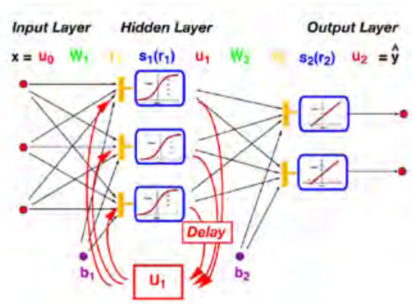
DeepNet trained to classes

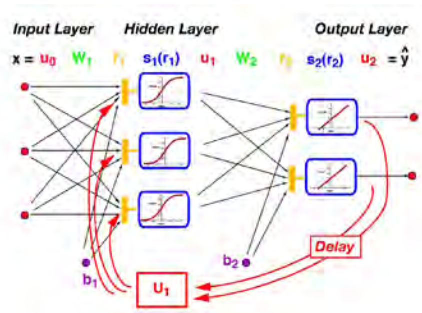


More on Recurrent Neural Networks

- **Feedback added to a feed-forward neural network** ***(discrete-time dynamic system)***
- ***One-step memory*** **introduced to network**

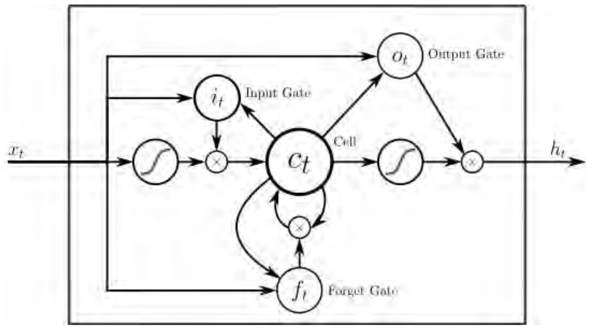
Elman Network	***Jordan Network***
$\mathbf{u}_1(k)=\mathbf{s}_1\left[\mathbf{W}_1\mathbf{x}(k)+\mathbf{U}_1\mathbf{u}_1(k-1)+\mathbf{b}_1\right]$	$\mathbf{u}_1(k)=\mathbf{s}_1\left[\mathbf{W}_1\mathbf{x}(k)+\mathbf{U}_1\mathbf{u}_2(k-1)+\mathbf{b}_1\right]$
$\mathbf{u}_2(k)=\mathbf{s}_2\left[\mathbf{W}_2\mathbf{u}_1(k)+\mathbf{b}_2\right]$	$\mathbf{u}_2(k)=\mathbf{s}_2\left[\mathbf{W}_2\mathbf{u}_1(k)+\mathbf{b}_2\right]$





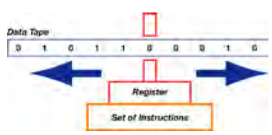
Long Short-Term Memory

- **Memory held until new value overwrites**
- **Cell, input gate, output gate, forget gate**



$$
\begin{aligned}
\mathbf{u}_f(k) &= \mathbf{s}_g\left[\mathbf{W}_f\mathbf{x}(k)+\mathbf{U}_f\mathbf{u}_h(k-1)+\mathbf{b}_f\right]\\
\mathbf{u}_i(k) &= \mathbf{s}_g\left[\mathbf{W}_i\mathbf{x}(k)+\mathbf{U}_i\mathbf{u}_h(k-1)+\mathbf{b}_i\right]\\
\mathbf{u}_o(k) &= \mathbf{s}_g\left[\mathbf{W}_o\mathbf{x}(k)+\mathbf{U}_o\mathbf{u}_h(k-1)+\mathbf{b}_o\right]\\
\mathbf{u}_c(k) &= \mathbf{u}_f(k).*\mathbf{u}_c(k-1)+\mathbf{u}_i(k).*\mathbf{s}_c\left[\mathbf{W}_c\mathbf{x}(k)+\mathbf{U}_c\mathbf{u}_h(k-1)+\mathbf{b}_c\right]\\
\mathbf{u}_h(k) &= \mathbf{u}_o(k).*\mathbf{s}_h\left[\mathbf{u}_c(k)\right]
\end{aligned}
$$

Klaus, 2015

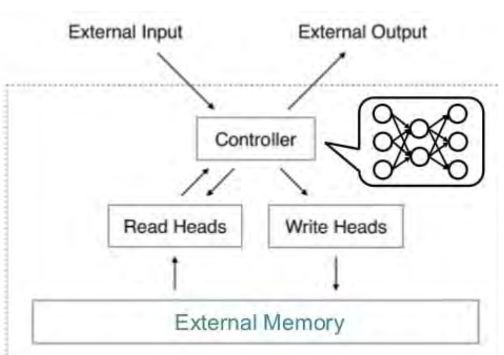


Neural Turing Machines

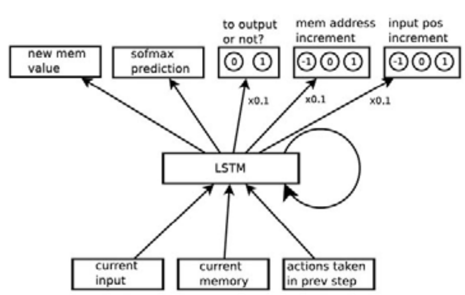
Sigmoid networks can be "Turing complete", Siegelmann & Sontag, 1991, 1995

- **Trainable read/write access to memory**
- **Controller/program implemented by neural networks**

- **Reinforcement-Learning NTM**
- **uses either feed-forward or LSTM neurons**
- **Improves on LSTM neurons**



Graves, 2014



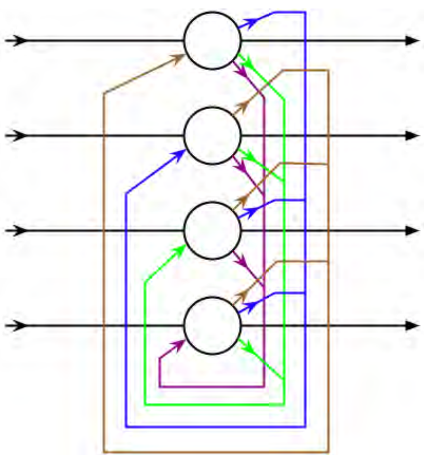
Zaremba, 2015

Next Time:
Communication, Information, and Machine Learning

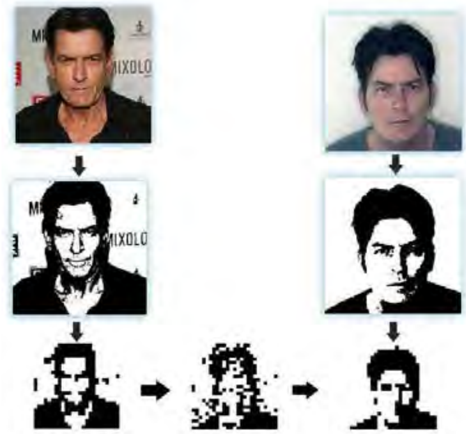
Supplemental Material

Hopfield Network

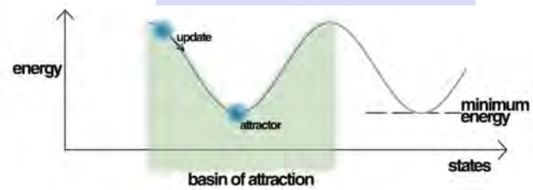
Alternative plot of 4-node network



Novel Image **Exemplar**



"Energy Landscape"

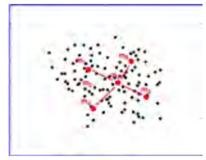


Linear Vector Quantization

- Incorporation of supervised learning in Semantic Net
- Classification of groups of outputs
- Type 1
  - Addition of codebook vectors, m_c, with known meaning

$$\mathbf{m}_c(k+1)=\begin{cases}\mathbf{m}_c(k)+\alpha_k\left[\mathbf{x}_k-\mathbf{m}_c(k)\right], & \textit{if classified correctly}\\ \mathbf{m}_c(k)-\alpha_k\left[\mathbf{x}_k-\mathbf{m}_c(k)\right], & \textit{if classified incorrectly}\end{cases}$$

Linear Vector Quantization



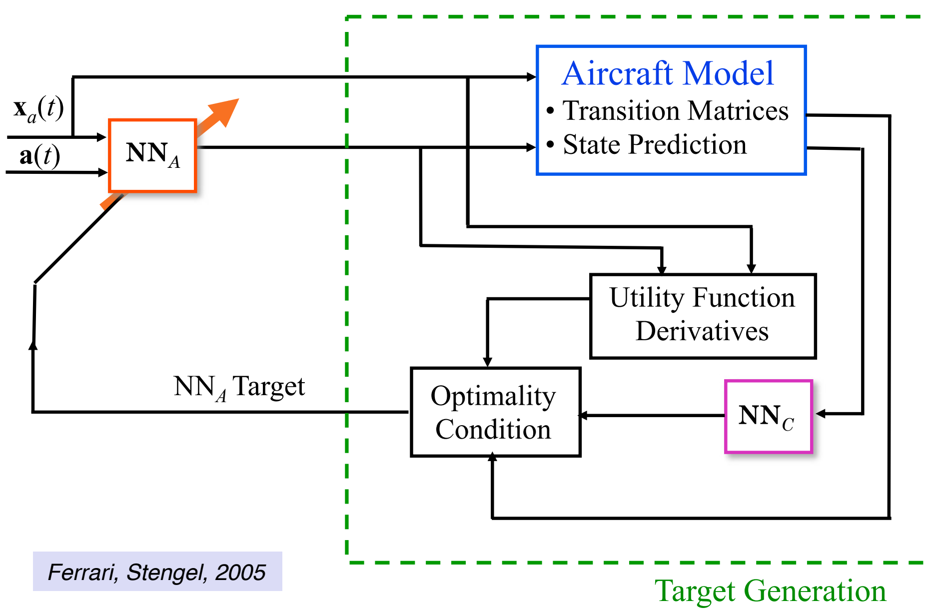
- **Type 2**
  - **Inhibition of nearest neighbor whose class is known to be different, e.g.,**
    - **x belongs to class of m_j but is closer to m_i**

$$\mathbf{m}_i(k+1) = \mathbf{m}_i(k) - \alpha_k\left[\mathbf{x}_k - \mathbf{m}_i(k)\right]$$

$$\mathbf{m}_j(k+1) = \mathbf{m}_j(k) + \alpha_k\left[\mathbf{x}_k - \mathbf{m}_j(k)\right]$$

Adaptive Critic Proportional-Integral Neural Network Controller

Adaptation of Control Network



Ferrari, Stengel, 2005

Adaptive Critic Proportional-Integral Neural Network Controller

Adaptation of Critic Network