Computational Intelligence Methods and their Applications for Data Analysis

Dr. Jacek M. ZURADA
University of Louisville
Obuda University Symposium
Budapest, September 2, 2015

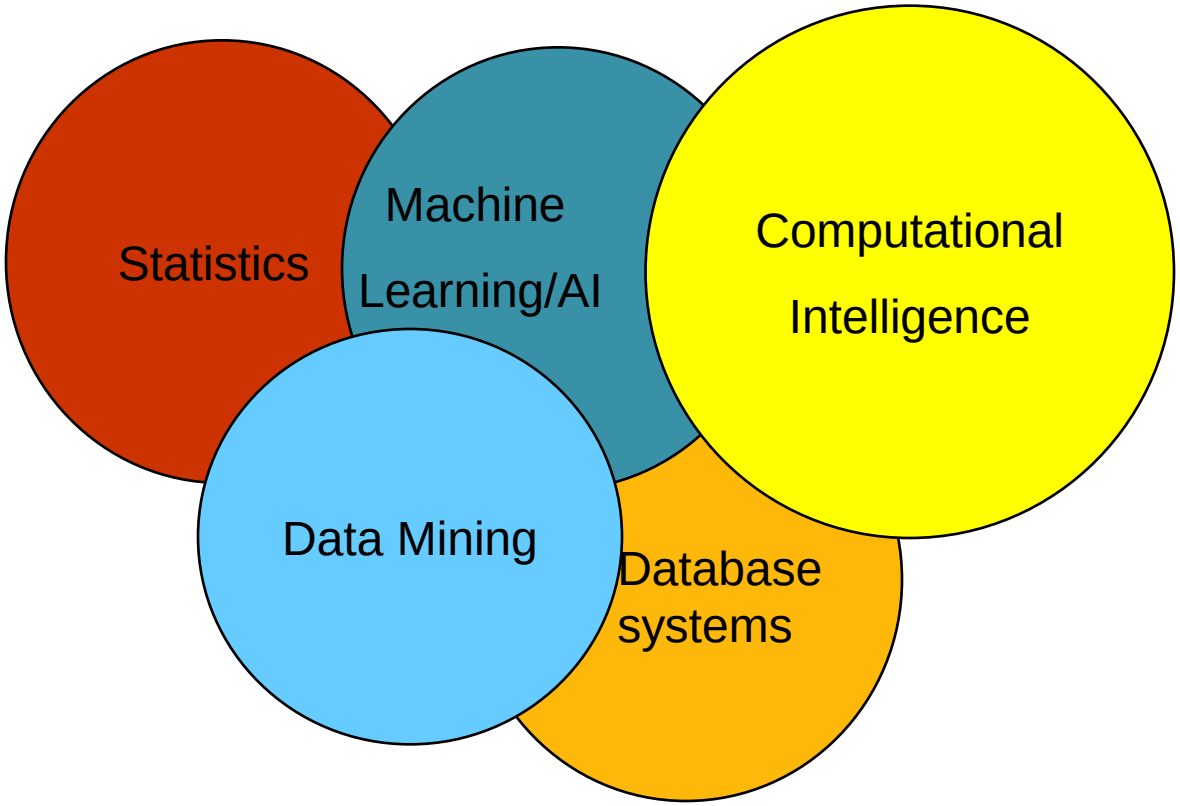
Computational Intelligence Methods and their Applications for Data Analysis

Credits:

- Dr. Tolga Ensari, Technical University of Istanbul
- Jan Chorowski, University of Louisville
- My former PhD students: G. Creech, A. Gaweda, A. Lozowski, A. Malinowski
- Other Researchers listed in references

CI among Data Analysis Techniques

- Data Analysis draws from machine learning/AI, computational intelligence, pattern recognition, statistics, data mining, and database systems



What is Computational Intelligence?

- Computational techniques: reasoning from data
  - A Family of Methods - when supporting other areas of technology
  - Standalone Technology/Gadget/Tool – when CI is embedded in a system that analyzes data previously ‘seen’ (pre-trained system, ex. ZIP code reader)
  - Hybrid Systems that train on data as needed (trainable systems)
- CI supports various areas of technologies and data analytics, incl. Big Data

Overview of the Presentation

Part I: Methods

- Supervised and unsupervised neural networks
- Fuzzy systems
- Genetic algorithms

Part II: Applications

- Control problem solutions with CI methods
- Optimization tasks

Tasks for Data Analysis with CI and ML

- Classification, diagnostics and monitoring
- Clustering
- Rule discovery
- Sequential pattern discovery/prediction
- Regression analysis
- Merger of feature extraction + classification
- Fusion of tasks and inputs: numerical, visual, categorical, nominal, common-sense verbal statement/observation, pure heuristics

ML and NN: Similarities and Differences

Both are part of Artificial Intelligence: aim at imitating human learning and develop intelligent systems through

-mimicking human learning as computational learning

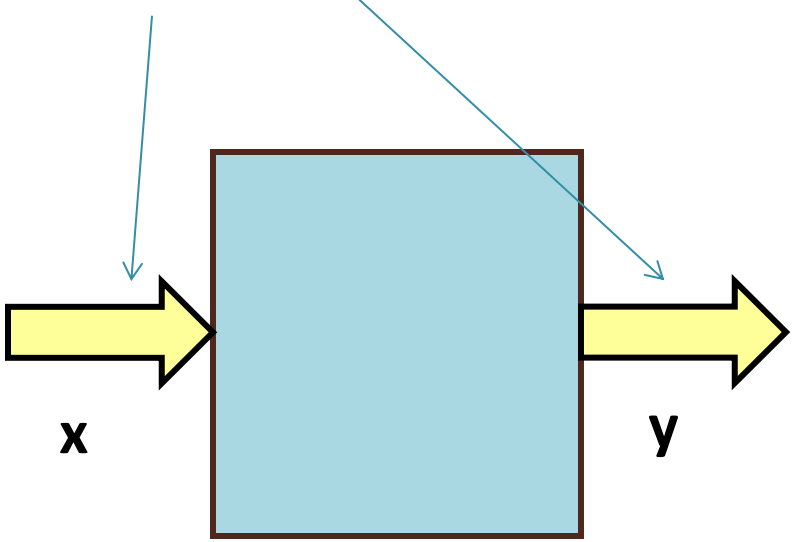
-solving decision-making tasks

-discovering of knowledge (data mining)

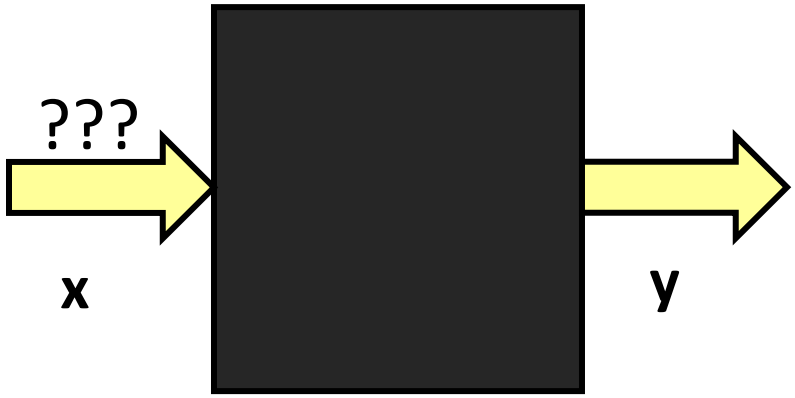
- ML provides good 'mental fit' - generates explicit (symbolic) knowledge, and conceptual partitioning of input space, tends to produce rules
- NN provide good 'data fit'- only maximizes the accuracy of mapping for unseen inputs

ML and NN: Similarities and Differences

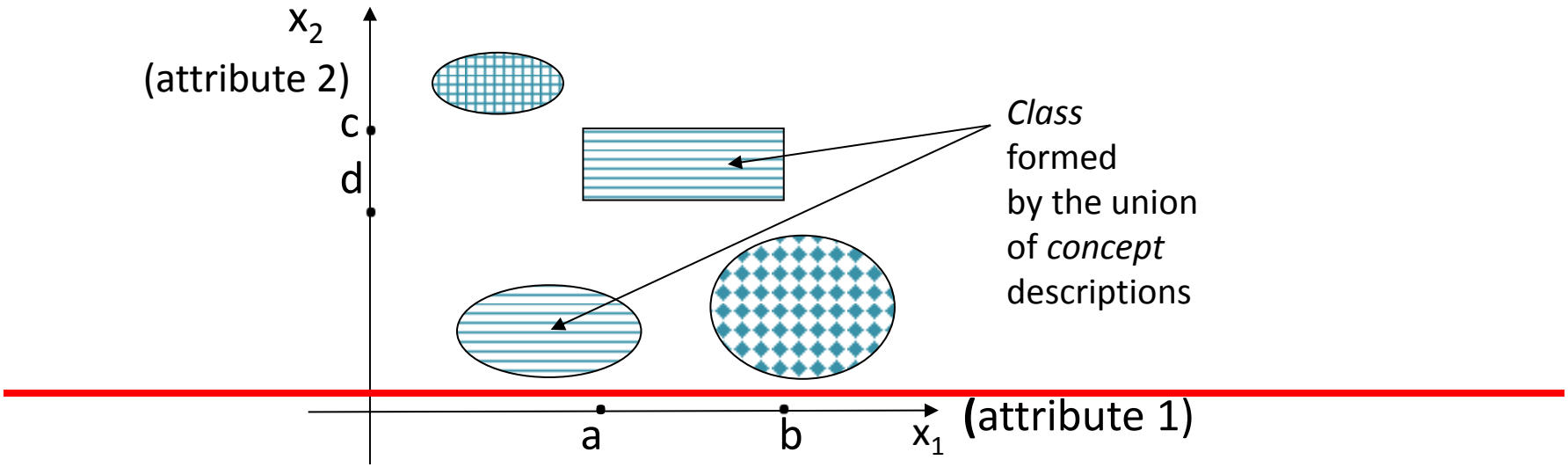
ML-based diagnostics: provides better explanation for a diagnosis by specifying symptoms



NN-based diagnostics: provides typically better accuracy, yet the black-box of little transparency as to how to explain a decision with understandable rules



ML and NN: Similarities and Differences



Ex. Non-overlapping partition of attribute space for three classes

Rule: If ($a<x_1<b$) AND ($c<x_2<d$) Then Class 1

Explains concept descriptions, generates interpretable rules

Rule induction starts at Null Hypothesis (entire plane)

Neural Networks: flexible models developed through learning from data.

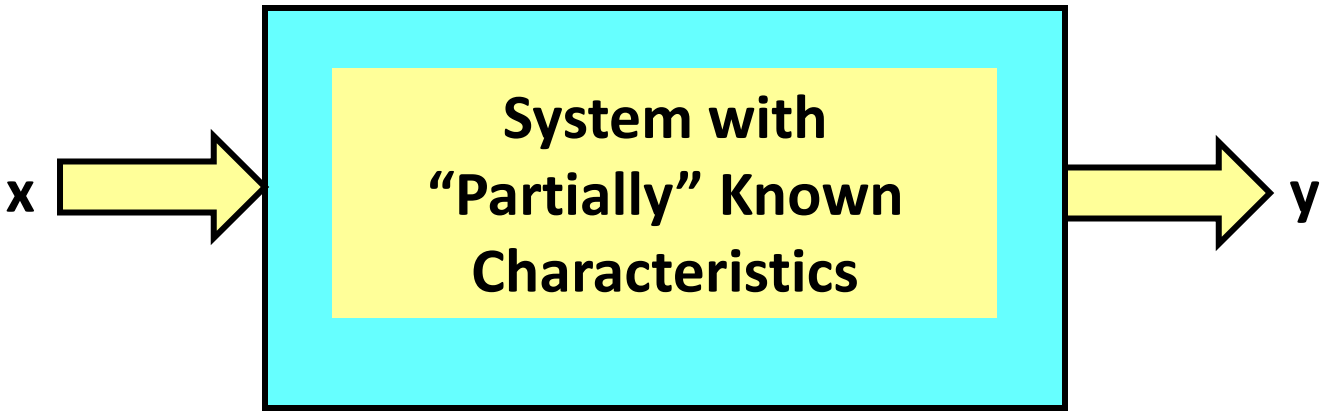
Tasks supported by NNs

-interpolation, approximation
-prediction
-clustering, data mapping

-classification
-principal component extraction
-data compression, data de-noising
-multidimensional data visualization

NN is about learning and re-use of the data models

NN Case I: Incomplete System Info

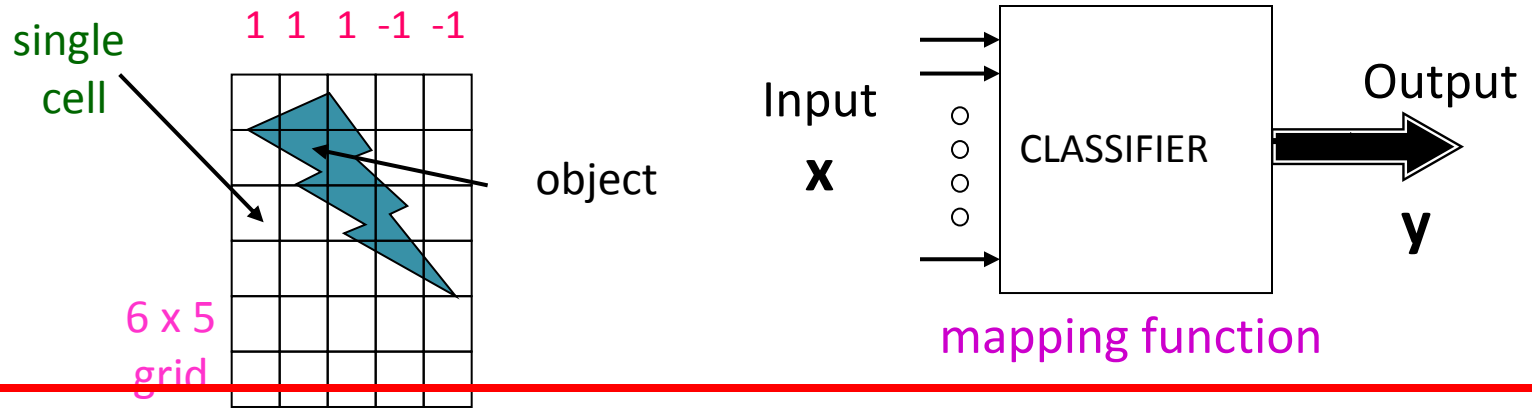


$x=[x_1\ x_2\ x_3\ \ldots\ x_n]^t$ is the input vector R^n

$y=[y_1\ y_2\ y_3\ \ldots\ y_m]^t$ is the output vector R^m

The user has only knowledge as to how this system reacts to some inputs, but no complete I/O relationship, just pairs of input/output data vectors

Example of Case I: x, y are encoded spatial patterns



Spatial Pattern

Components of the input vector are assigned a value of -1 or 1

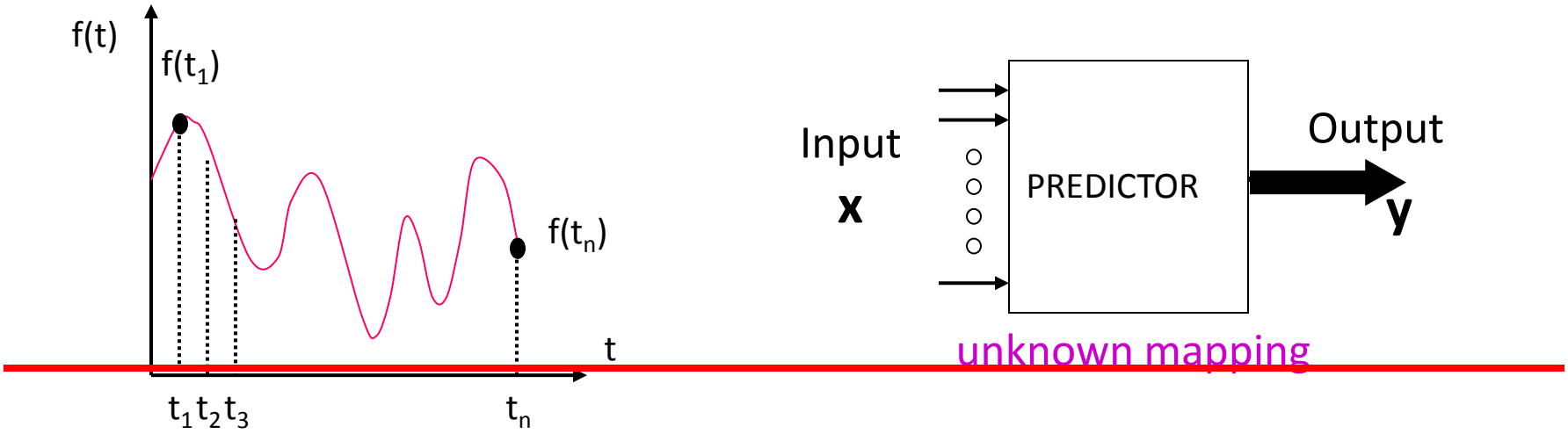
$\mathbf{x} = [1\ 1\ 1\ -1\ -1 \ldots\ -1]^t$

where **x** is (30x1)

Classifier maps the set {**x**} into {**y**}, where x_i are binary-valued (but also real-valued), and y_i are always binary-valued,

i.e., $y_i \in \{-1, 1\}$

Example of Case I: x, y are temporal patterns



Temporal Pattern

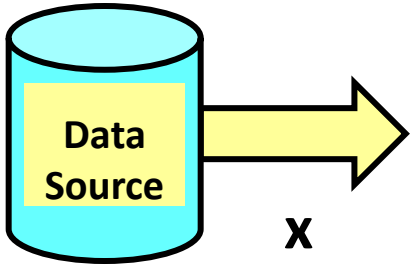
Vectors are formed at discrete time instants at t_1, t_2,…t_n….

$$\mathbf{x} = [\, f(t_1)\ f(t_2)\ \ldots\ldots f(t_n)\,]^t$$

Predictor maps the set **{x}** into **{y}**, where x_i and y_i are real-valued, and **y** is a vector with m predicted outputs

$$\mathbf{y} = [\, f(t_{n+1})\ f(t_{n+2})\ \ldots\ldots f(t_{n+m})\,]^t$$

NN Case II: Partially Known Data Characteristics



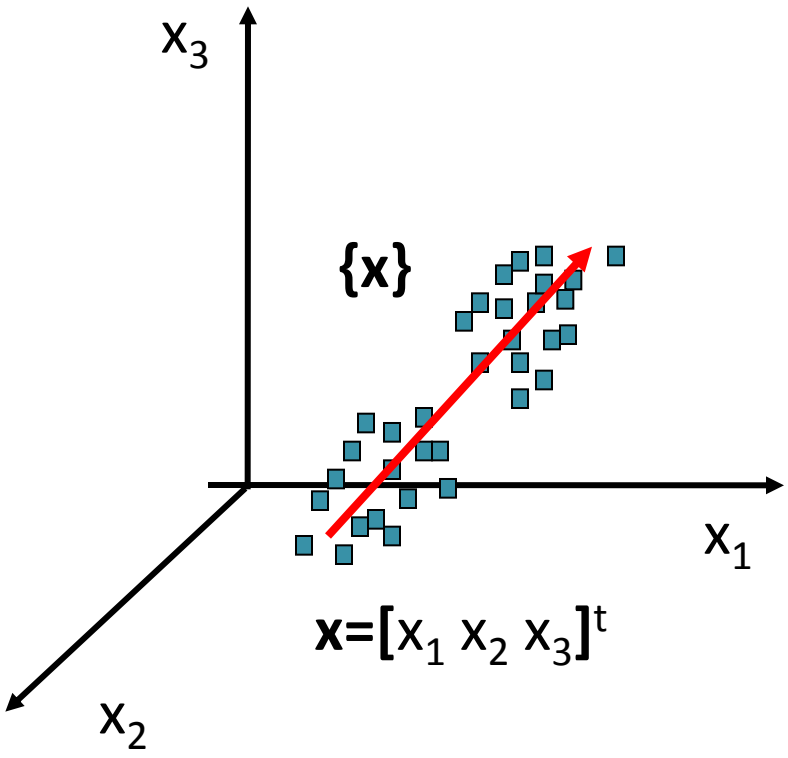
$\mathbf{x}=[x_1\ x_2\ x_3\ \ldots\ x_n]^t$ is the data vector R^n

x can be input or output vector

"Partially" known data characteristics known to the user based on what a Data Source has produced, yet no I/O relationships even exist, neither has the user clues about them/future data.

When experimental data is abundant
in case I: sets {**x**}, {**y**}
in case II: set {**x**}
A user may infer useful data properties and build computational tools
Question: HOW?

Example of Case II: x are multidimensional patterns



Here, the user may be interested in discovering any structure in data, such as

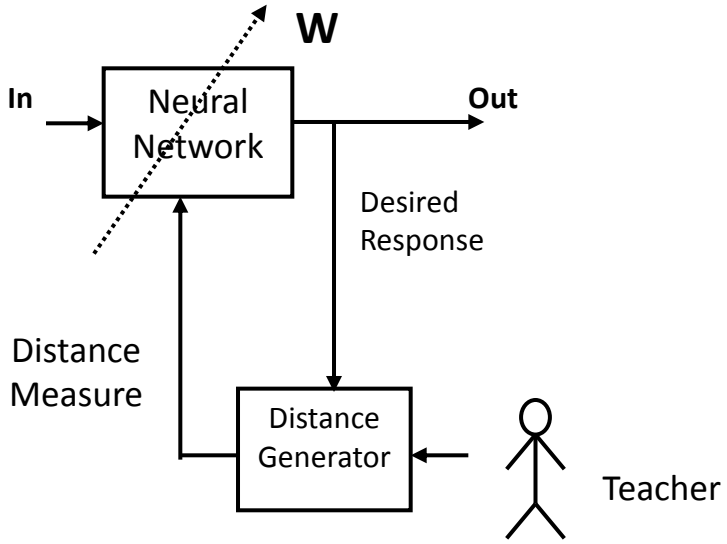
-clusters
-correlation
-possibly compress data (to one or two coordinates)

Note: **x** can be spatial patterns, but can also depict a temporal wave when each point **x** is embedded in time

Time-domain patterns would form trajectories connecting t_1, t_2, t_3, t_4, ….

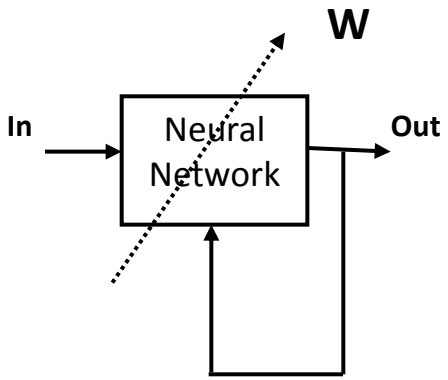
Learning in NNs

Supervised Learning



- Teacher provides the desired responses (also called target patterns)
- Teacher's defined distance between the actual and desired response serves as an error measure that corrects network parameters' matrix **W**
- Network's weights are successively adapted to decrease this error

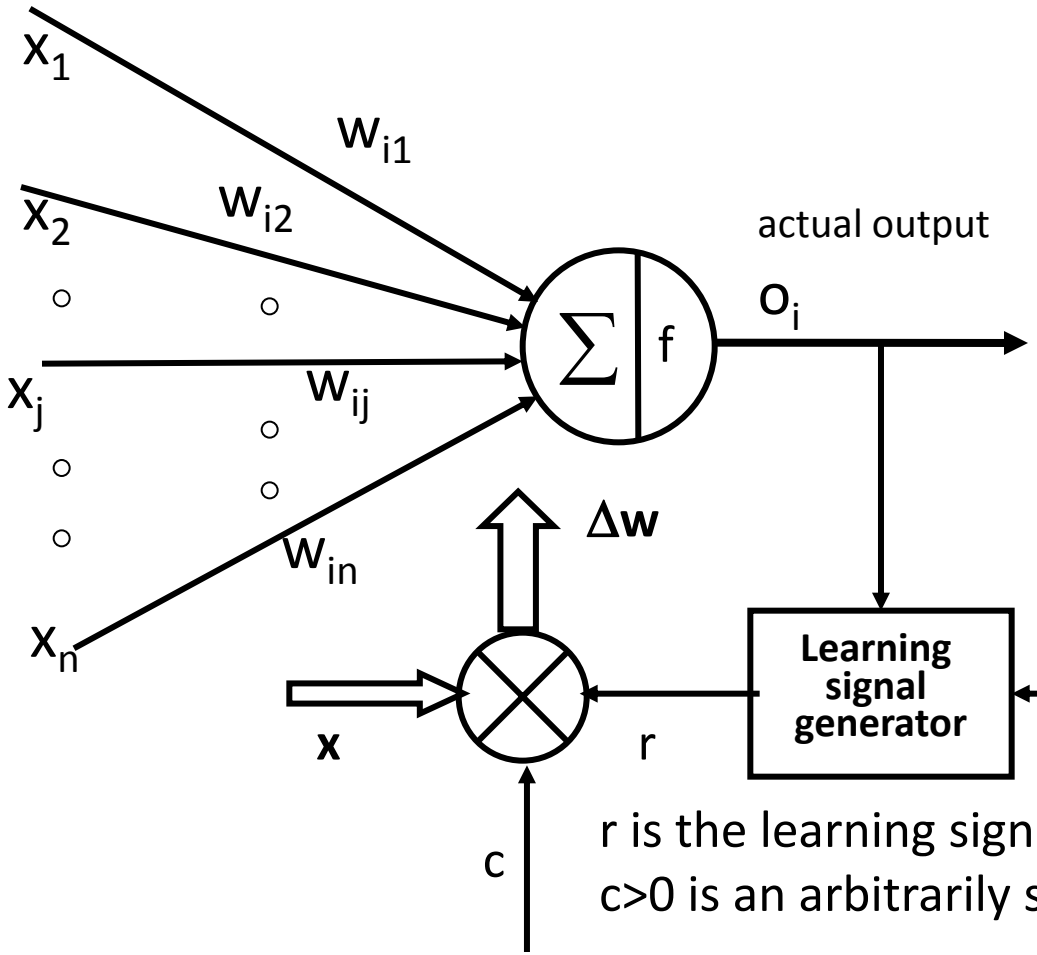
Unsupervised Learning



Primary form of learning (why?)

- Teacher embeds only the rules
- Desired responses not known
- Learning is based on analysis of responses to previous inputs
- Network discovers for itself data regularities or properties and changes in parameters' matrix **W**

General Learning Rule of a single (i-th) neuron



General Weight Adaptation Formula (Learning Rule):

$$\Delta \mathbf{w}(t) = c\, r\, (\mathbf{w}, \mathbf{x}, d)\, \mathbf{x}(t)$$

Note: $\Delta \mathbf{w}(t)$ is proportional to the input vector $\mathbf{x}(t)$!!!

Also, d may be unavailable

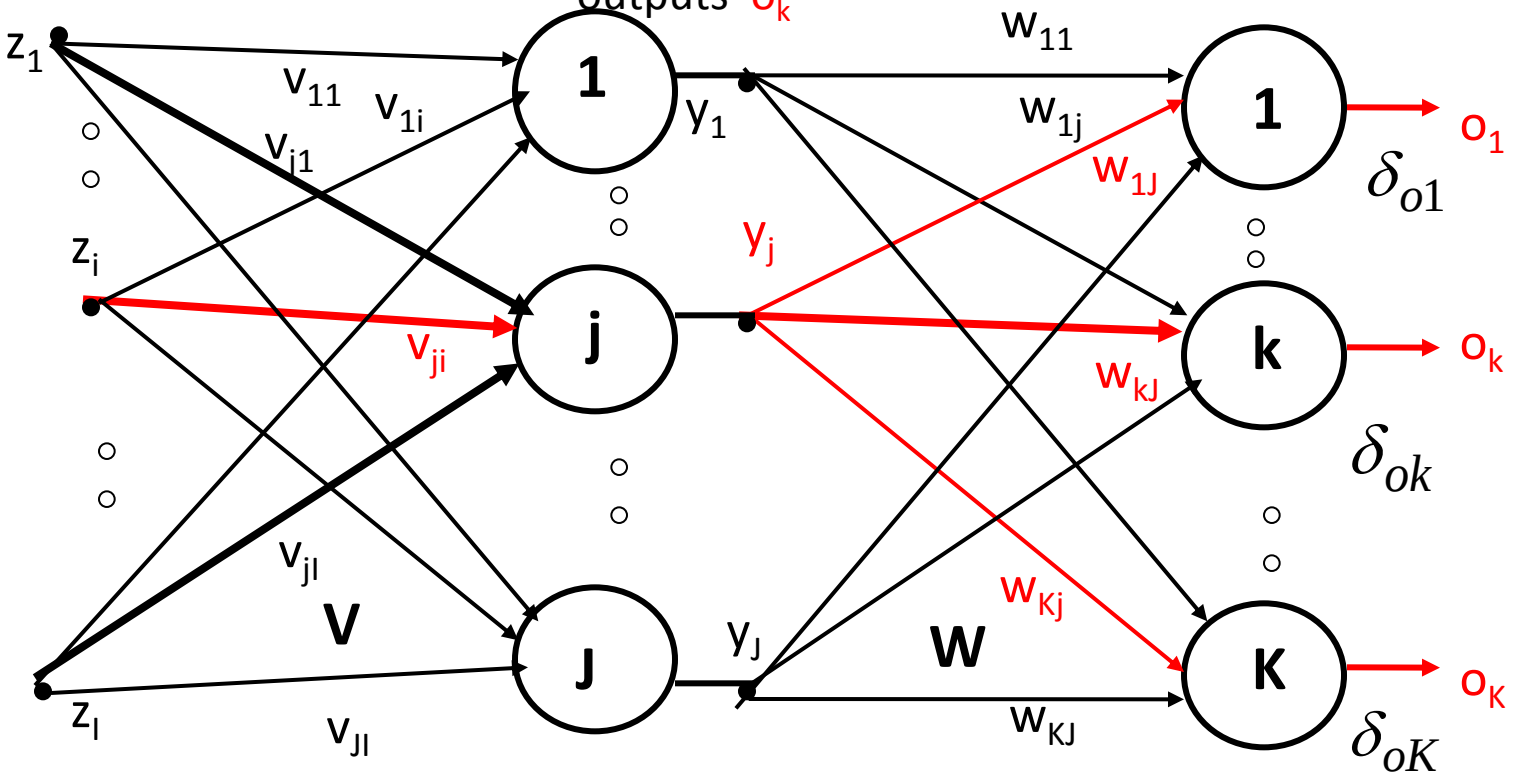
r is the learning signal (error)

c>0 is an arbitrarily selected *learning constant*

For different rules r is defined differently. Time refers to a learning time instant (discrete).

Error Back Propagation Training Algorithm Unveiled

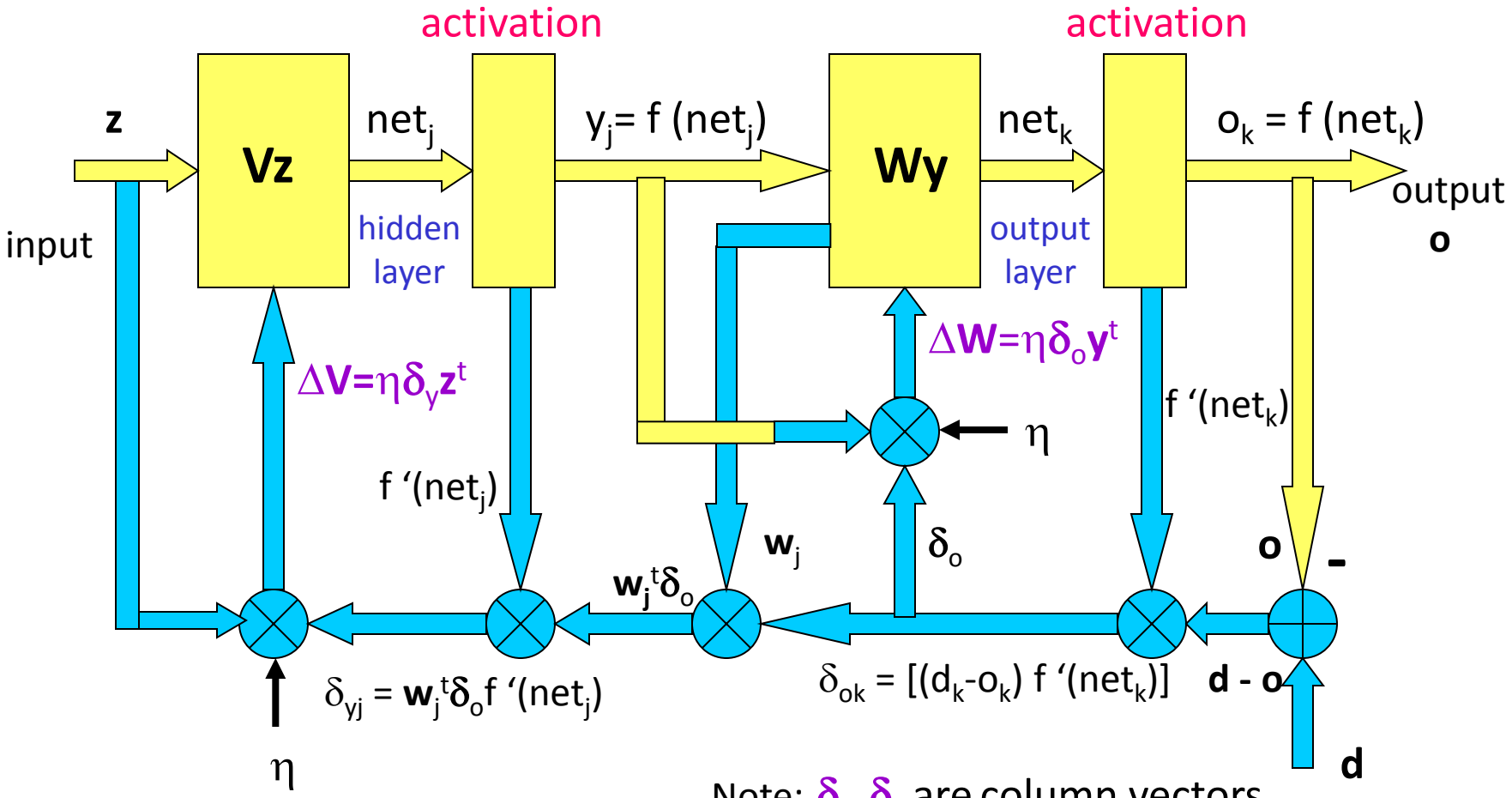
To find adjustment of v_{ji} we need a Generalized Delta Learning Rule as v_{ji} affects all K outputs o_k



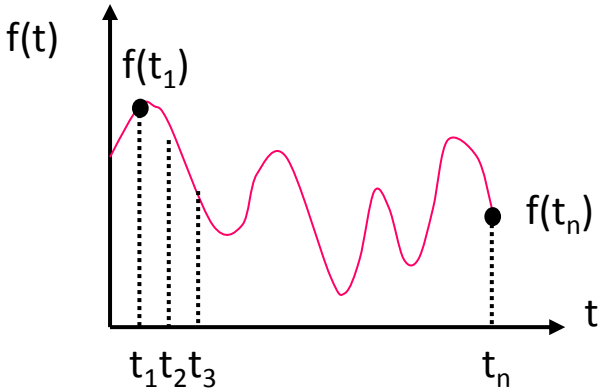
where $\delta_{ok} = (d_k - o_k)o_k'$ are contributing learning signals (errors)

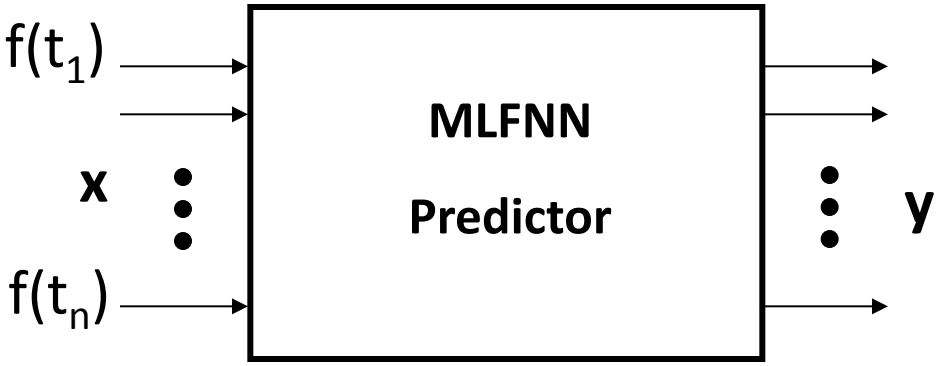
Red weights in the output layer contribute to errors produced by v_{ji}

EBP Algorithm-Block Diagram Compacted



Application of MLFNN for Prediction





$\mathbf{x} = [\, f(t_1)\ f(t_2)\ \ldots\ldots f(t_n)\ ]^t$

Time-Delay NN

$\mathbf{y} = [\, f(t_{n+1})\ f(t_{n+2})\ \ldots\ldots f(t_{n+m})\ ]^t$

vector of m future outputs

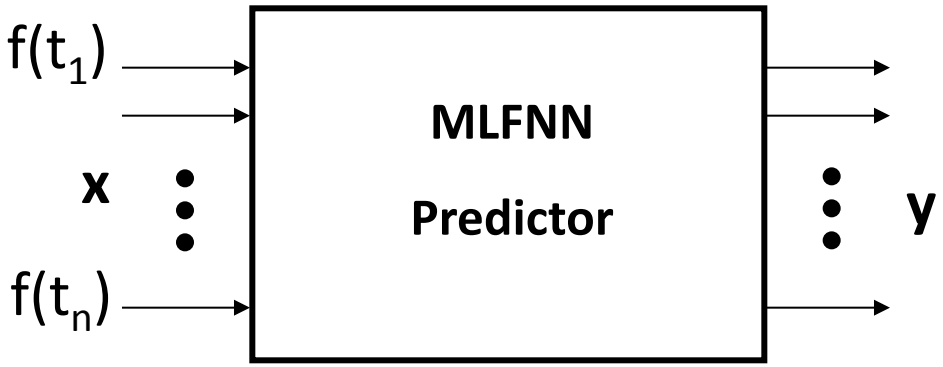
For m=1, we have single-step predictor

$$\mathbf{y} = f(t_{n+1})$$

For a single step predictor computing $f^*(t_{n+1})$, multi-step predictors can be built by shifting the input sequence accordingly by one time delay:

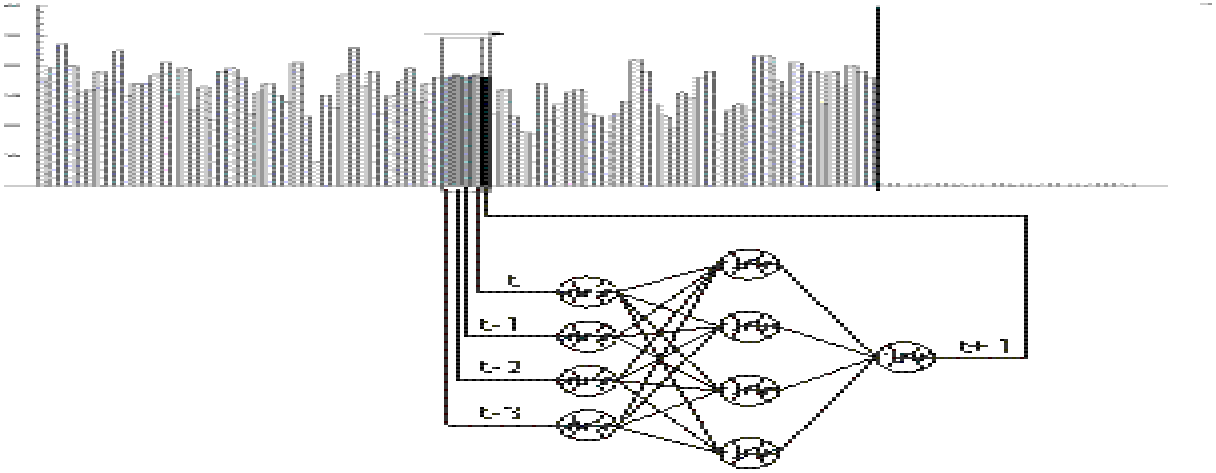
$f^*(t_{n+1})$ replaces $f(t_n)$ at the n-th input,
$f(t_n)$ replaces $f(t_{n-1})$ at the (n-1)th input, etc…
input $f(t_1)$ is now unused .

Application of MLFNN for Prediction



y is vector of m future outputs

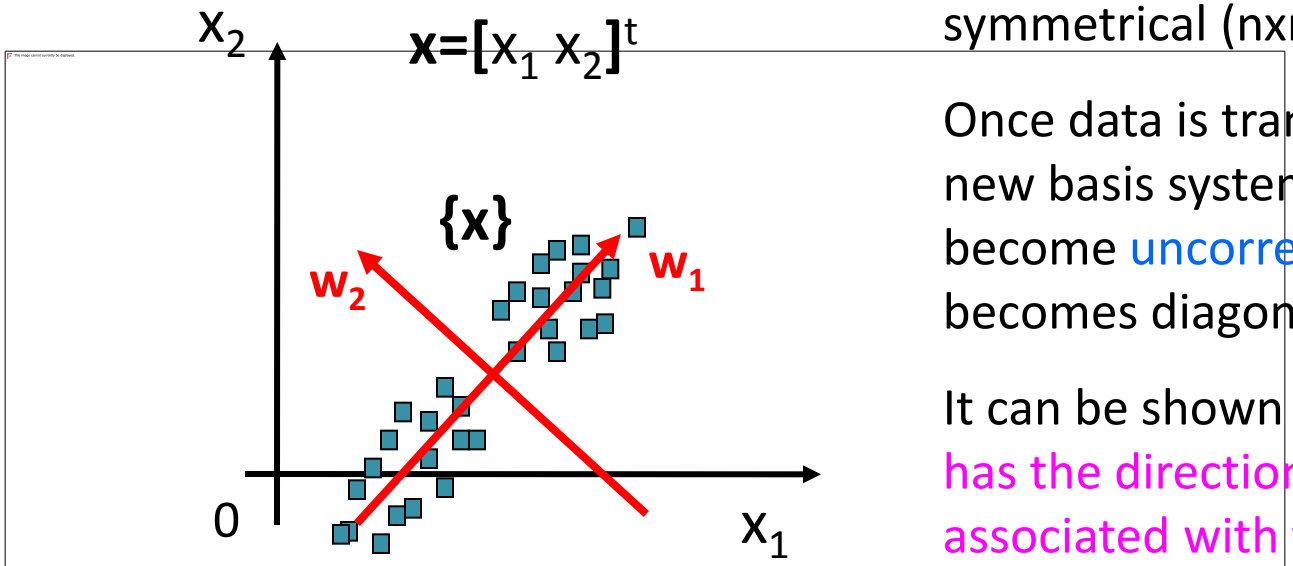
here **y** = $f(t_{n+1})$, n=4



Case II: Principal Component Analysis (PCA)

Data projected on vector $\mathbf{w}_1$ show the highest variance

To reduce vectors **x** (n=2) to a scalar y (m=1), we choose its projection on $\mathbf{w}_1$ (first PC)

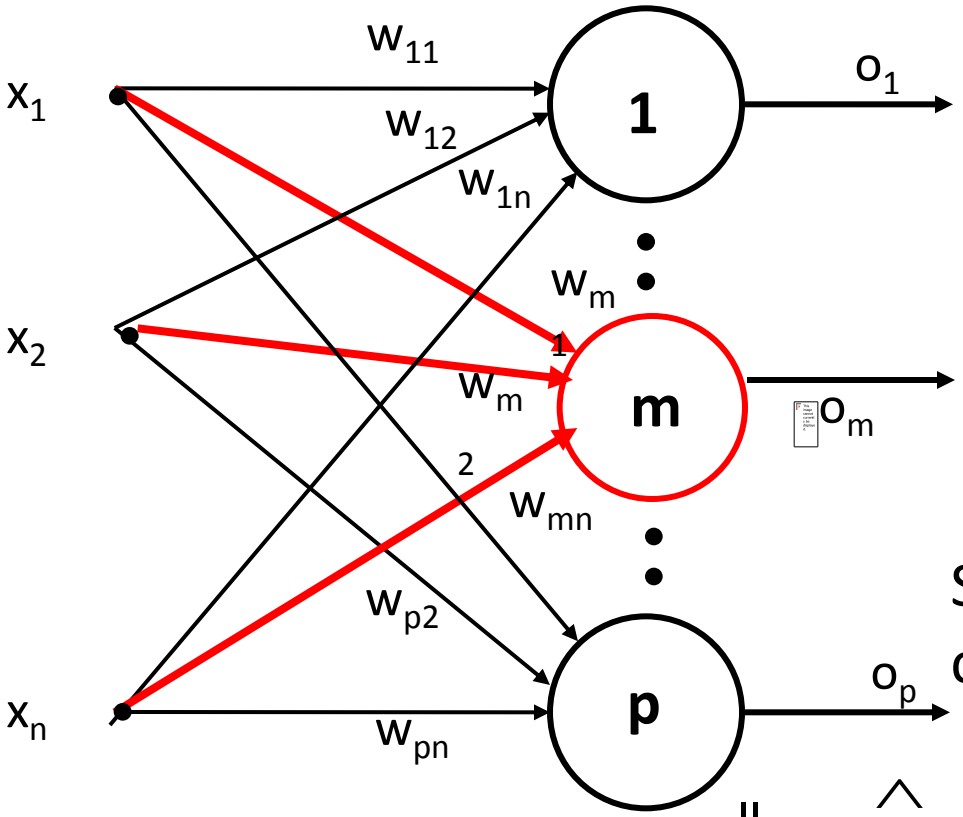


Original data **{x}** is correlated and its autocovariance matrix $\mathbf{R}_{xx}$ is symmetrical (nxn)

Once data is transformed into the new basis system $\mathbf{w}_1$, $\mathbf{w}_2$, **{y}** become uncorrelated, and $\mathbf{R}_{yy}$ becomes diagonal

It can be shown that the first PC has the direction of the eigenvector associated with the largest eigenvalue of the matrix $\mathbf{R}_{xx}$, the second PC has the direction of the eigenvector of the second largest eigenvalue … etc

WTA for Cluster Learning and Learning Vector Quantization (LVQ)



$$\Delta \mathbf{w}_m = \alpha(\mathbf{x} - \hat{\mathbf{w}}_m)$$

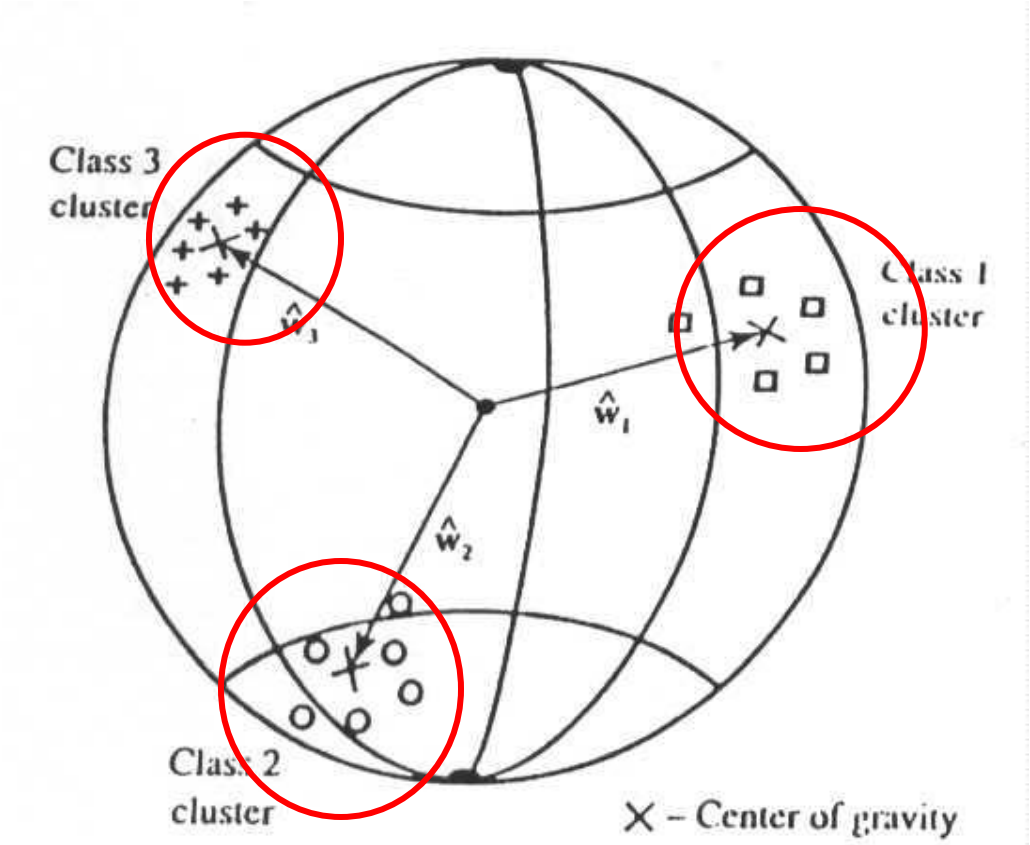
$$\hat{\mathbf{w}}_m^t \mathbf{x} = \max_{i=1,2....p}(\hat{\mathbf{w}}_i^t \mathbf{x})$$

α>0 an arbitrary learning constant

Selecting the winner, or m, corresponds to Step 2 since

$$\| \mathbf{x} - \hat{\mathbf{w}}_m \| = \min_{i=1,2,..\ p} \{ \| \mathbf{x} - \hat{\mathbf{w}}_i \| \}$$

WTA for Cluster Learning and Learning Vector Quantization (LVQ) (ctd)



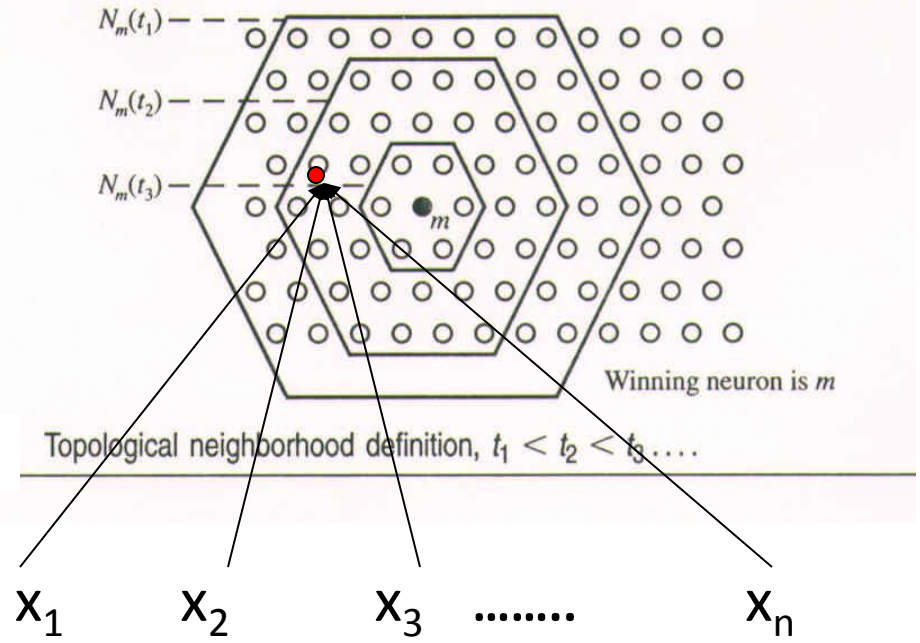
This competitive learning NN clusters new data points NOT used for training

Learning Vector Quantization assigns codebook vectors to each cluster (supervised process) and binary codes to each cluster.

LVQ is used for terse coding of multidimensional data. Here 2 bits are required to encode 3 clusters (instead of encoding a dozen individual data).

Kohonen's Self-organizing Feature Maps: Learning Algorithm

Hexagonal neighborhoods, N; t is training time



1) *Winner Selection*:

$$\mathbf{w}_m^t \mathbf{x} = \max_i (\mathbf{w}_i^t \mathbf{x})$$

where $i \in$ *all neurons*

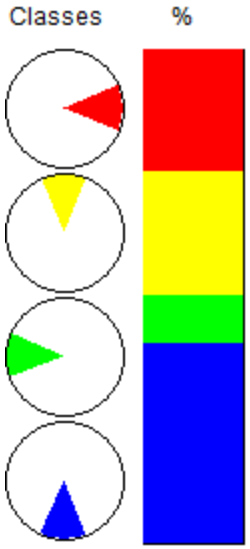
2) *Weights Update*

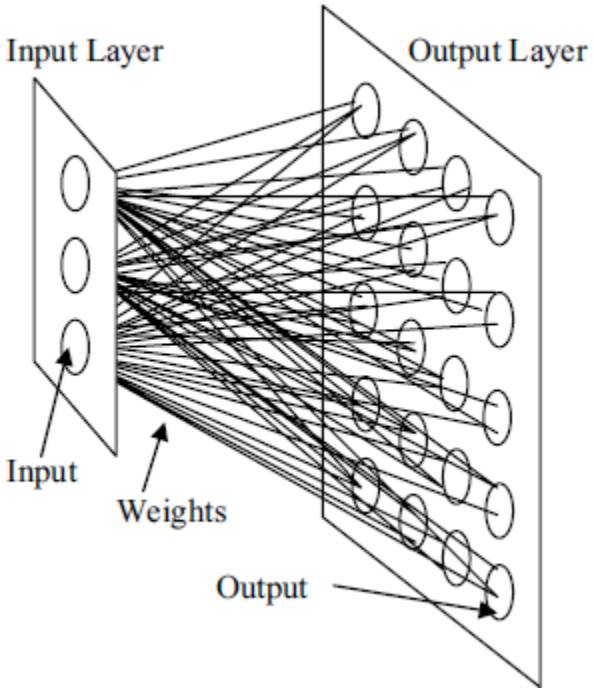
$$\Delta \mathbf{w}_i = \alpha(\mathbf{x} - \mathbf{w}_i),\ i \in N(t)$$

α>0 arbitrary learning constant

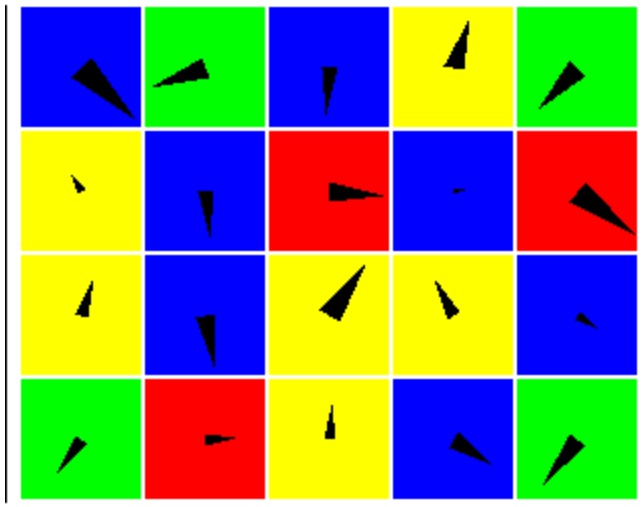
- Neurons learn only within the neighborhood N of the winner, m-th neuron
- The radius of the neighborhood N shrinks gradually (initially N includes the entire map, in final stages radius is zero)
- Weights no longer need to be normalized after each step

Kohonen's Self-organizing Feature Maps (ctd)



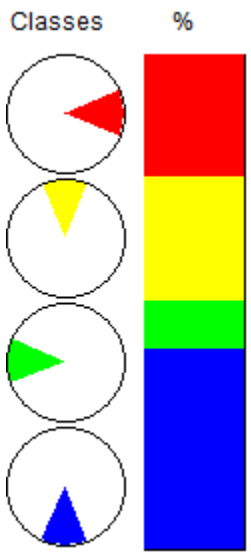


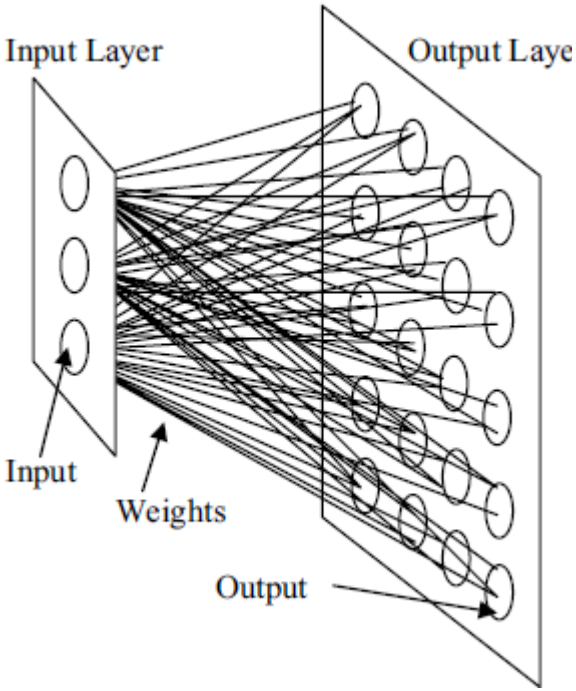
Initial (lack of) organization

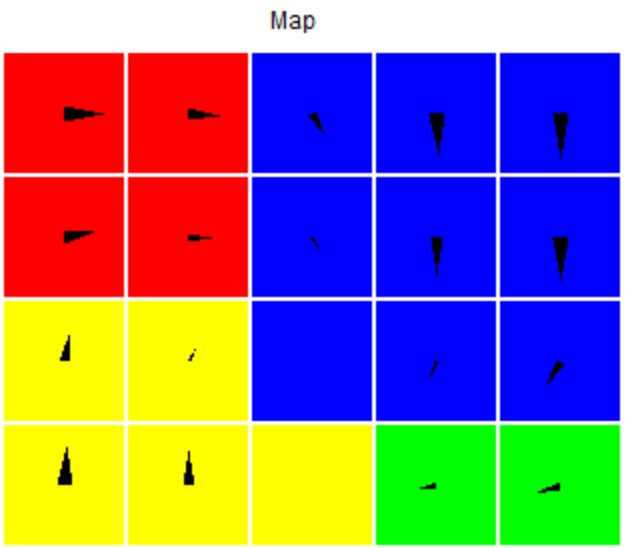


- Each n-dimensional input vector connected to each neuron (WTA, competitive learning)
- Neurons at fixed geometrical locations

Kohonen's Self-organizing Feature Maps (ctd)







Emerged organization

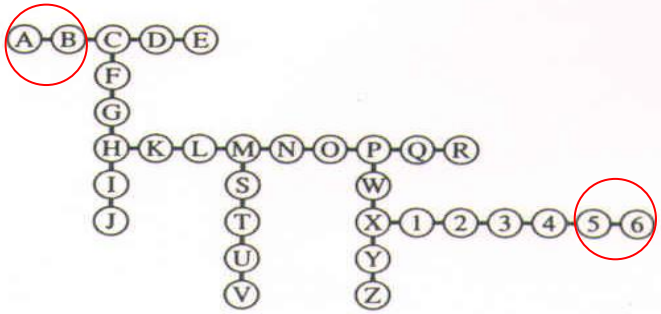
Kohonen's Self-organizing Feature Maps (ctd)

	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
Attribute																																
x_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
x_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
x_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
x_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
x_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

(a)

B	C	D	E	*	Q	R	*	Y	Z
A	*	*	*	*	P	*	*	X	*
*	F	*	N	O	*	W	*	*	1
*	G	*	M	*	*	*	*	2	*
H	K	L	*	T	U	*	3	*	*
*	I	*	*	*	*	*	*	4	*
*	J	*	S	*	*	V	*	5	6

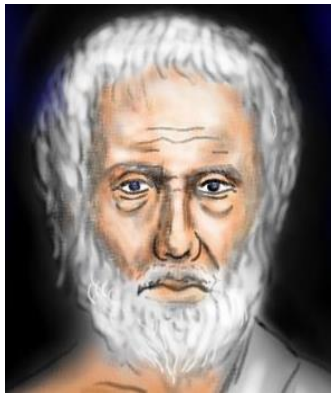
(b)



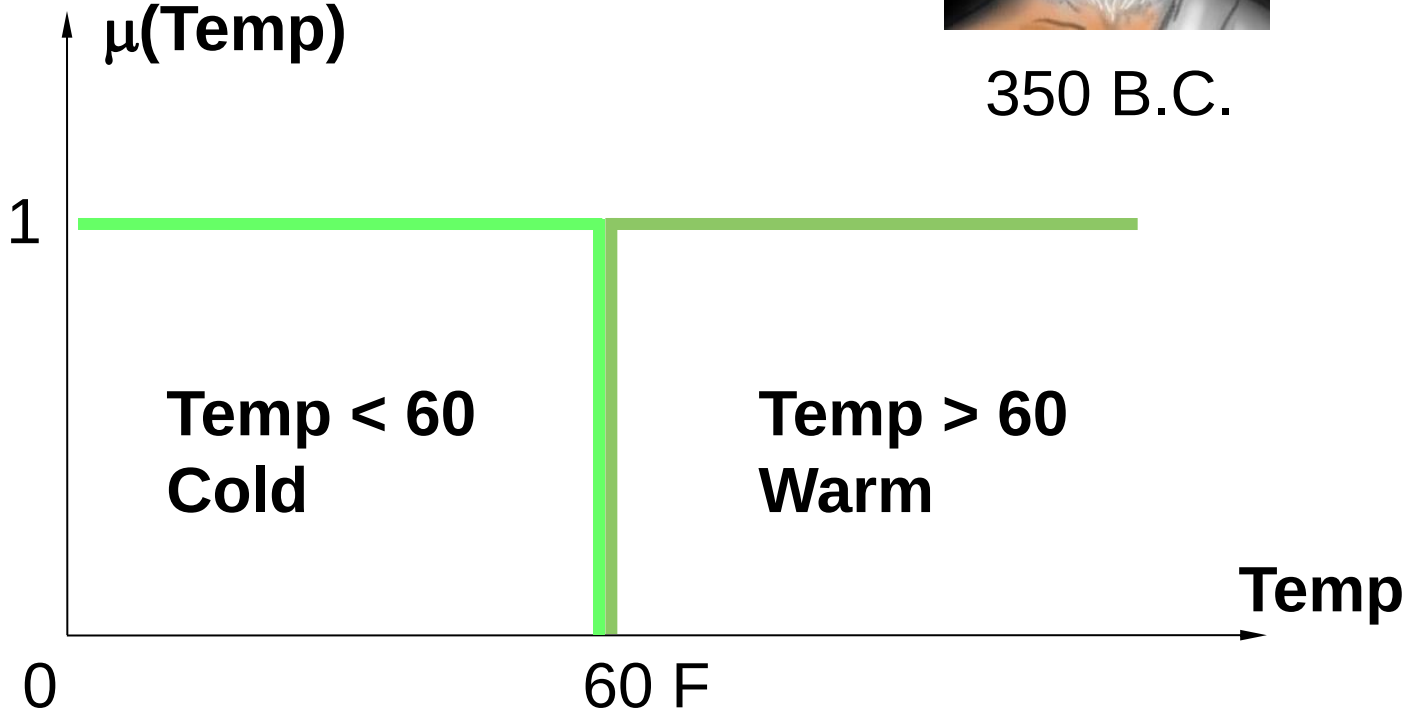
(c)

Self-organizing feature mapping example: (a) list of patterns, (b) feature map produced after training, and (c) minimum spanning tree. [from Kohonen (1984). © Springer Verlag; reprinted with permission.]

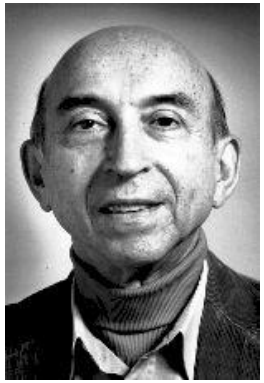
Fuzzy Sets



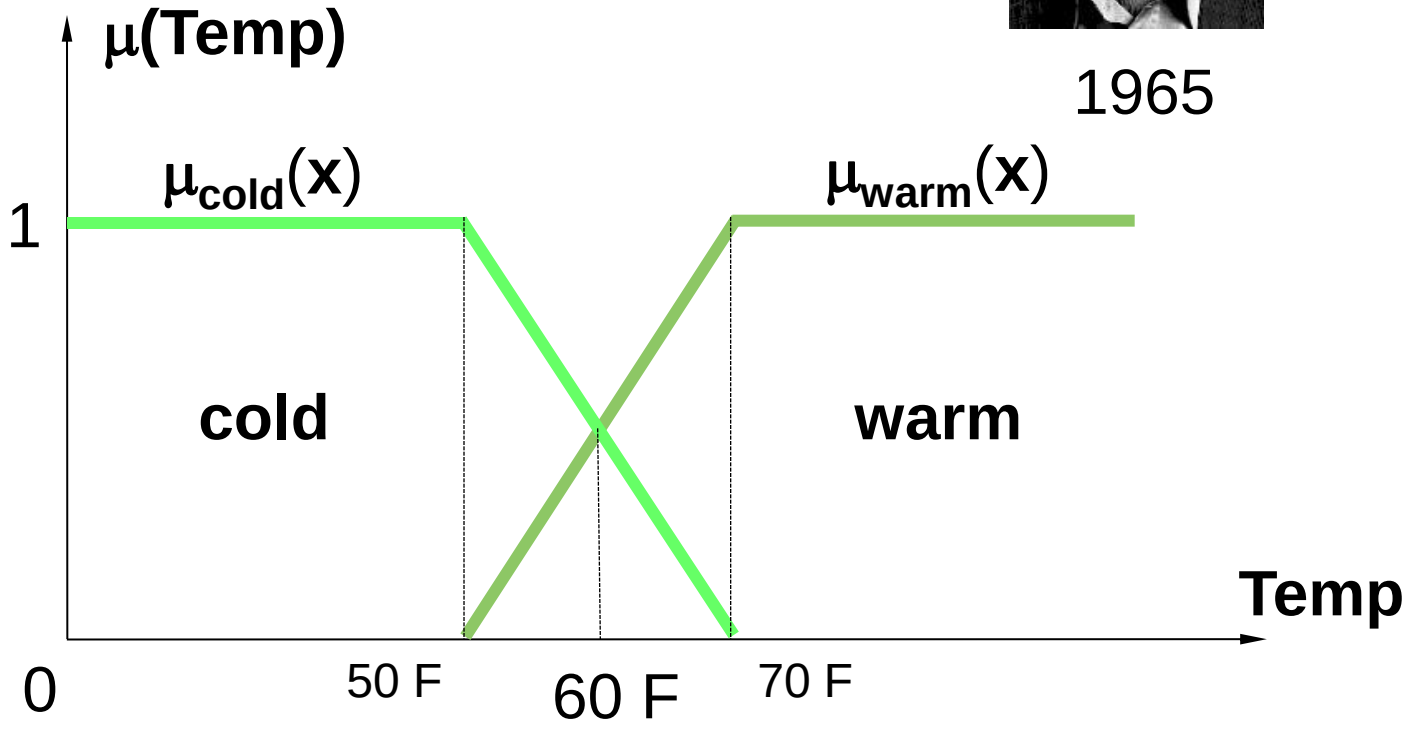
350 B.C.



Fuzzy Sets



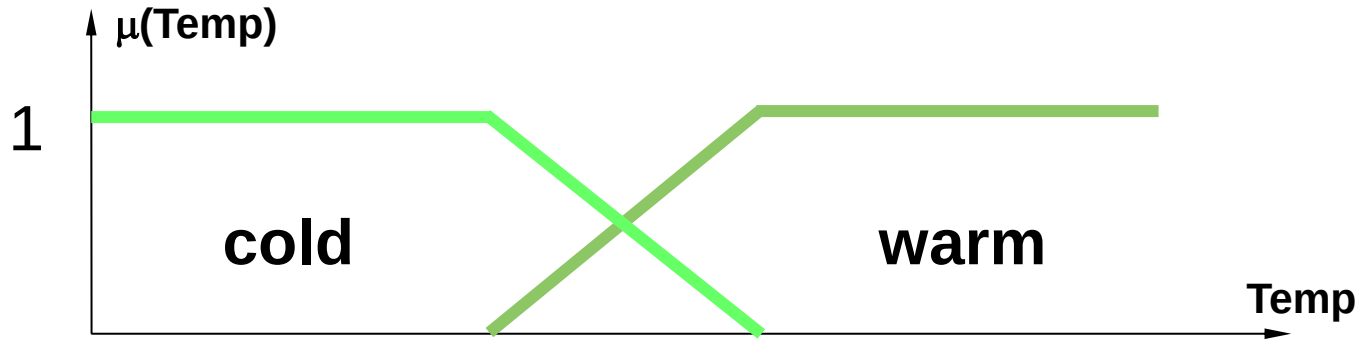
1965



Fuzzy Set Operations

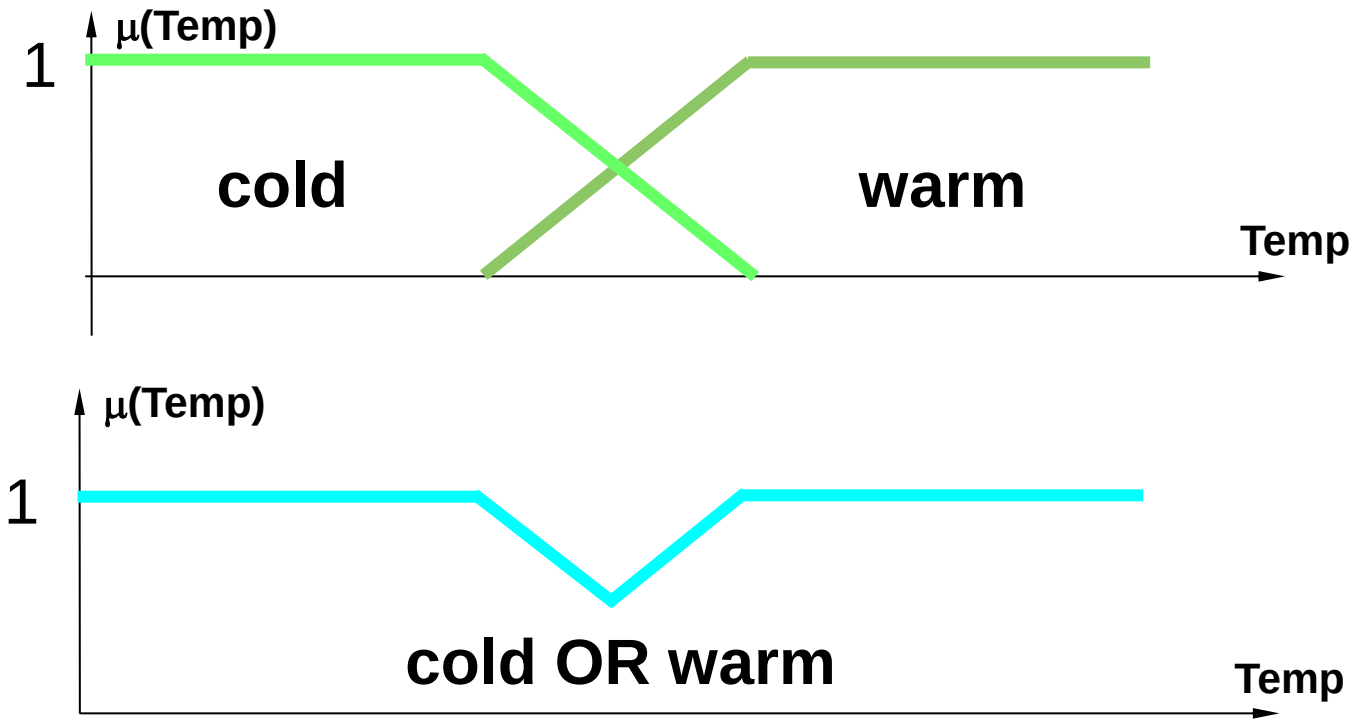
Intersection (AND): $\mu_{cold \cap warm} = \min [\mu_{cold}(x), \mu_{warm}(x)]$

x



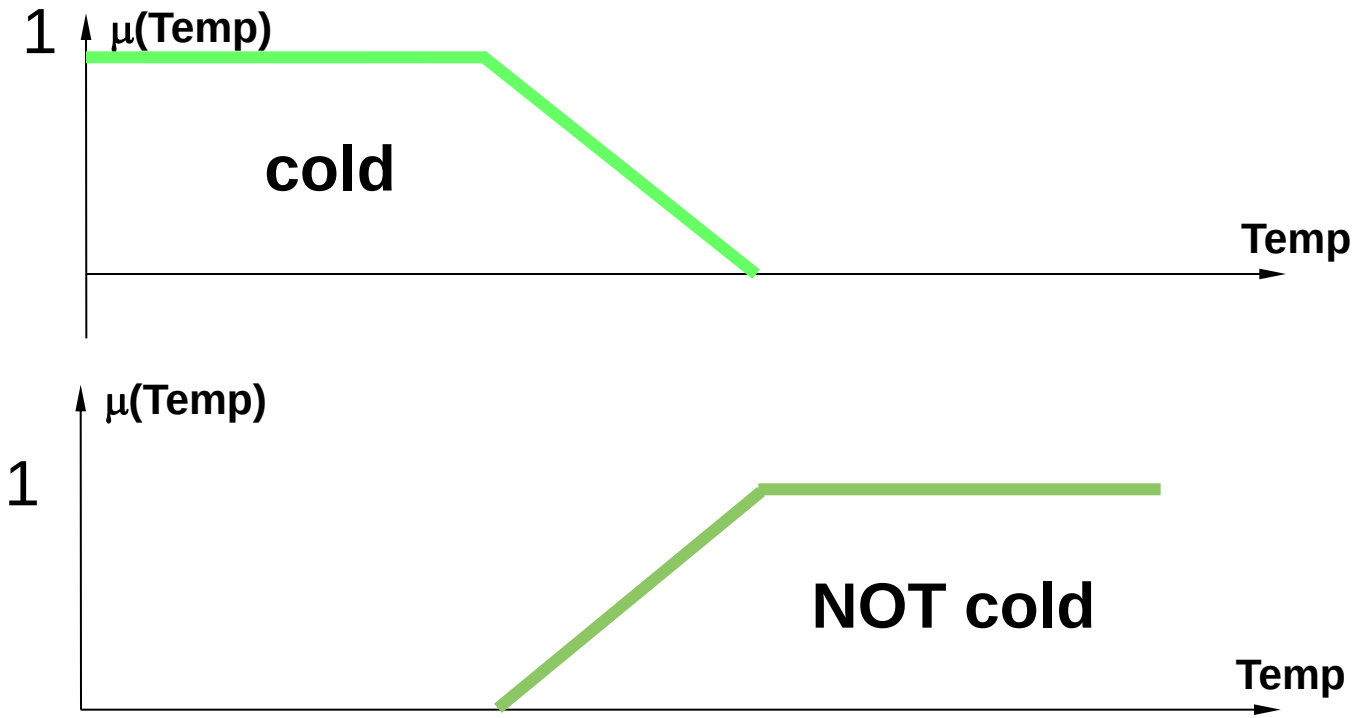
Fuzzy Set Operations

Union (OR): $\mu_{cold \cup warm} = \max_{x} [\, \mu_{cold}(x), \mu_{warm}(x) \,]$

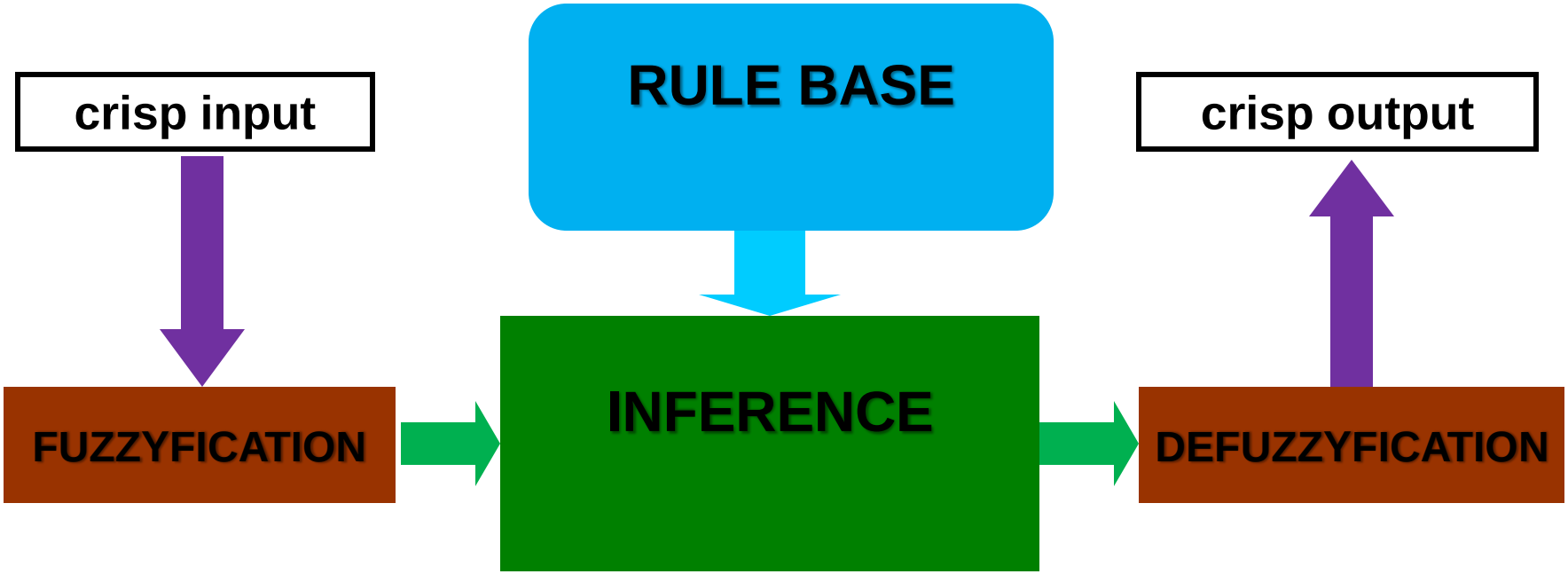


Fuzzy Set Operations

Negation (NOT): $\mu_{\text{not cold}} = 1 - \mu_{\text{cold}}(x)$



Fuzzy Expert System



Practical Example
"Poor man's air conditioner in detail"

- Input variables, known when measurable:
  - Temperature (ºF)
  - Humidity (%)

- Output, can be computed:
  - Cooling fan speed (rpm)

- Fuzzify>Execute rules>Defuzzify

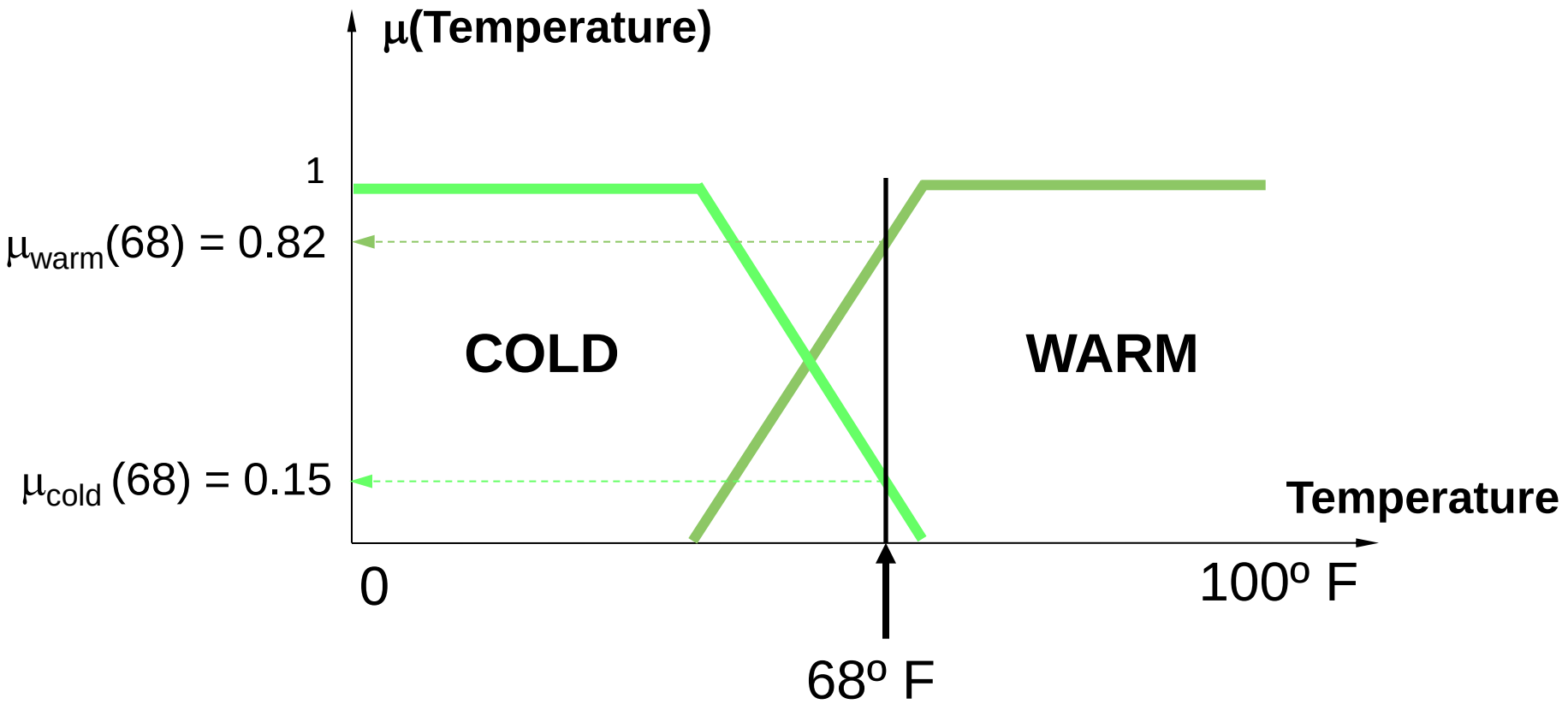
Rule Base: common-sense rules known to the human

1. IF Temperature is WARM AND Humidity is HIGH
   THEN Fan Speed is VERY FAST

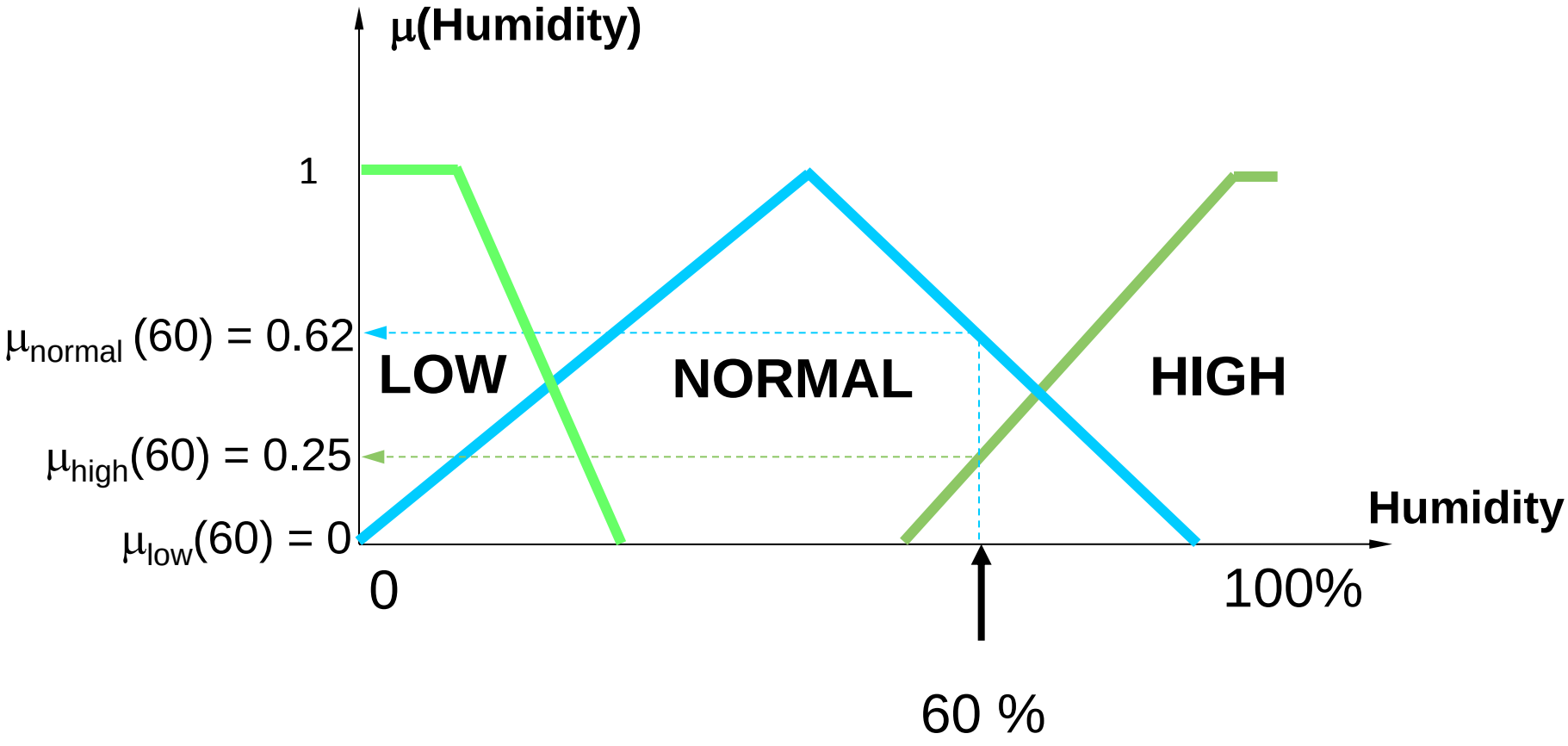
2. IF Temperature is WARM AND Humidity is NORMAL
   THEN Fan Speed is FAST

3. IF Temperature is COLD AND Humidity is LOW
   THEN Fan Speed is SLOW

Fuzzification 1 of 2: Temperature

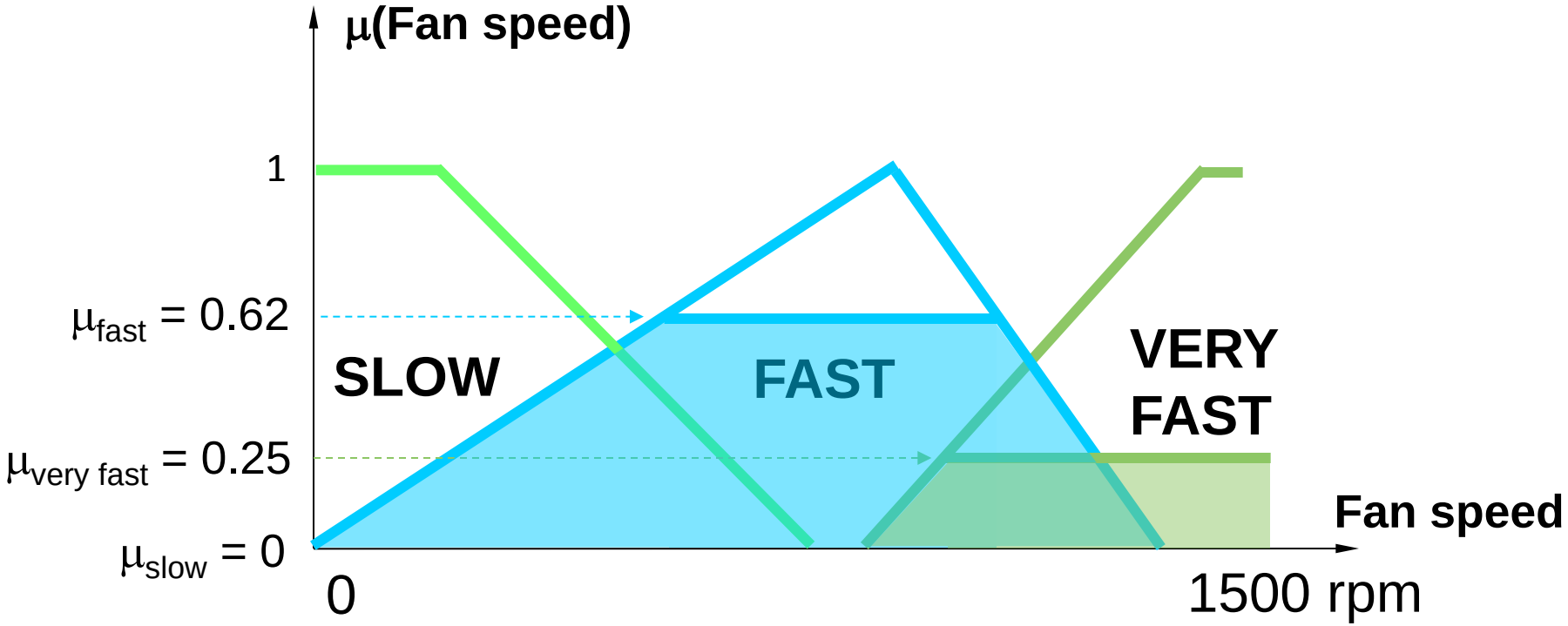


Fuzzification 2 of 2: Humidity

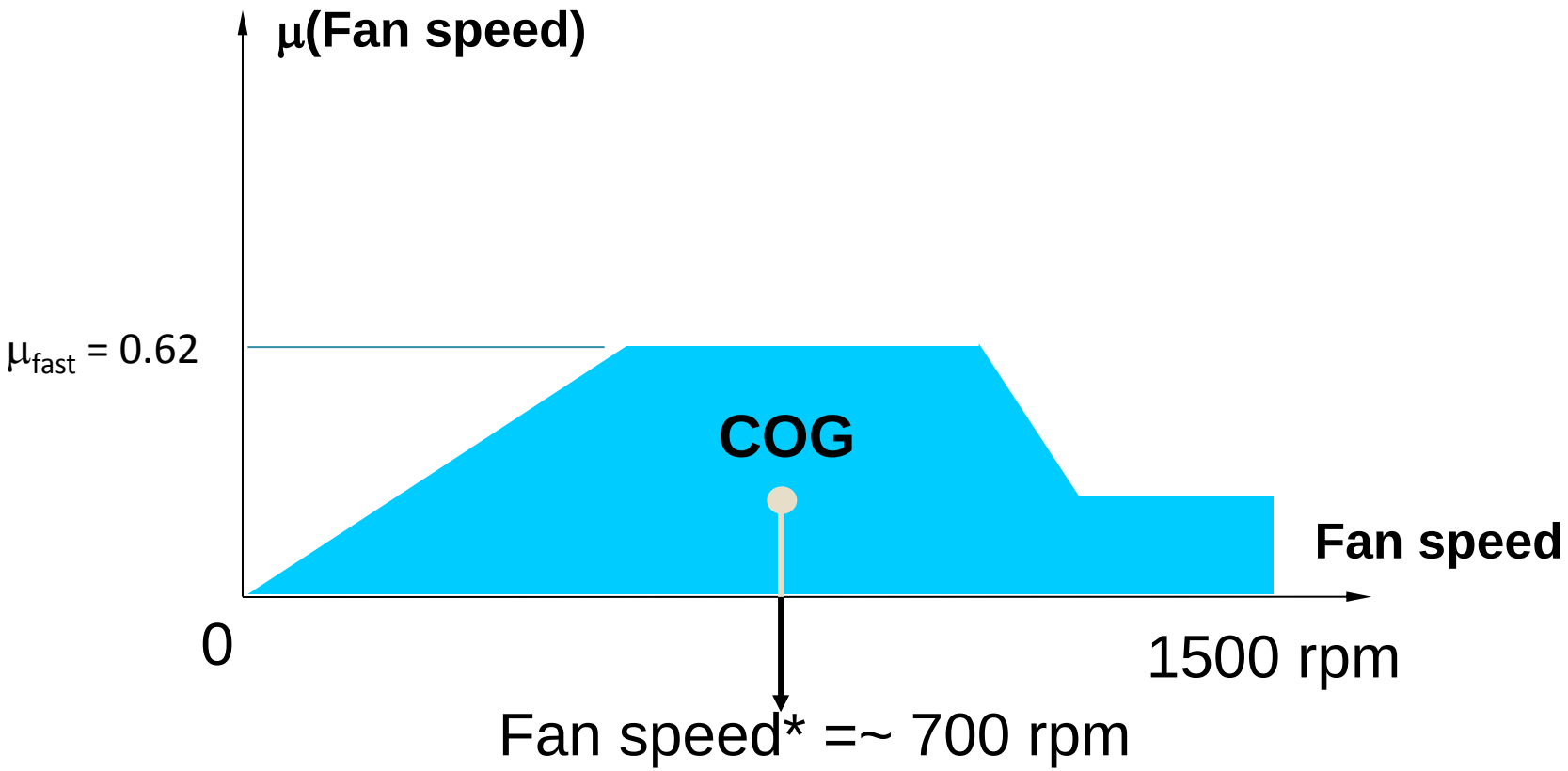


Defuzzyfication: computing output fuzzy set

3 rules to implement: R1-Very Fast, R2-Fast, R3-Slow



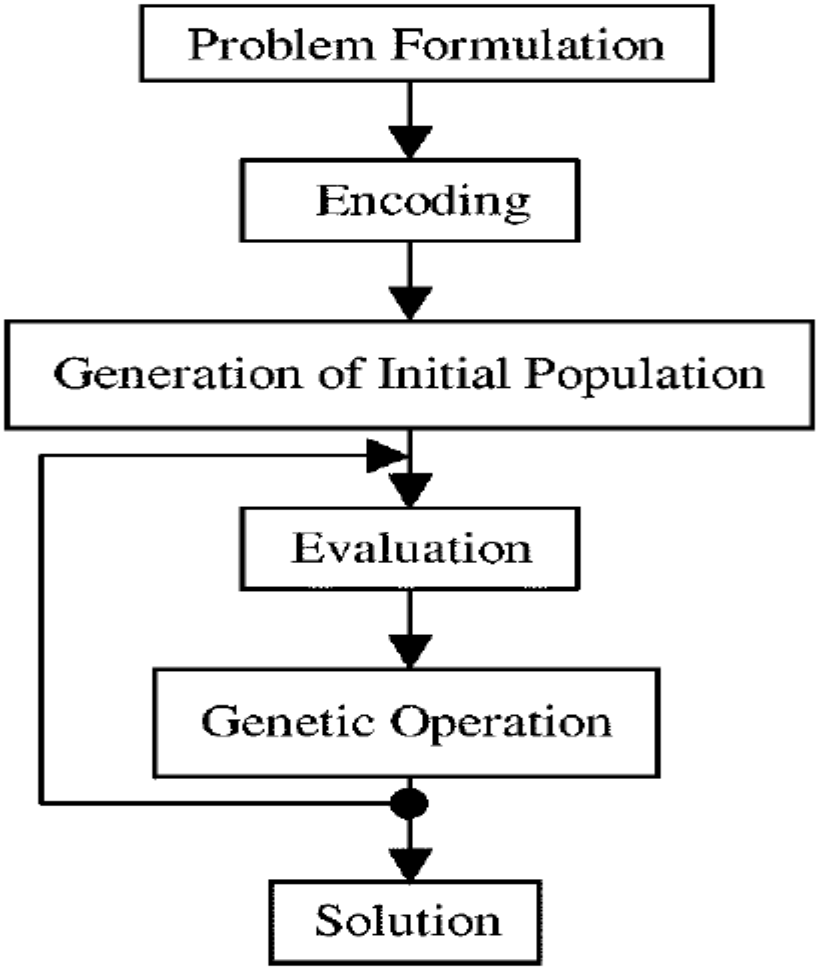
Defuzzyfication: computing output fuzzy set



Genetic Algorithms: Quick Review

- A class of probabilistic optimization algorithms
- Inspired by the biological evolution process
- Uses concepts of “Natural Selection” and “Genetic Inheritance” (Darwin 1859)
- Originally developed by John Holland (1975)
- Well suited for hard problems where little is known about the underlying search space
- Widely-used in business, science and engineering, typically for complex optimization problems

Evolutionary Design Cycle

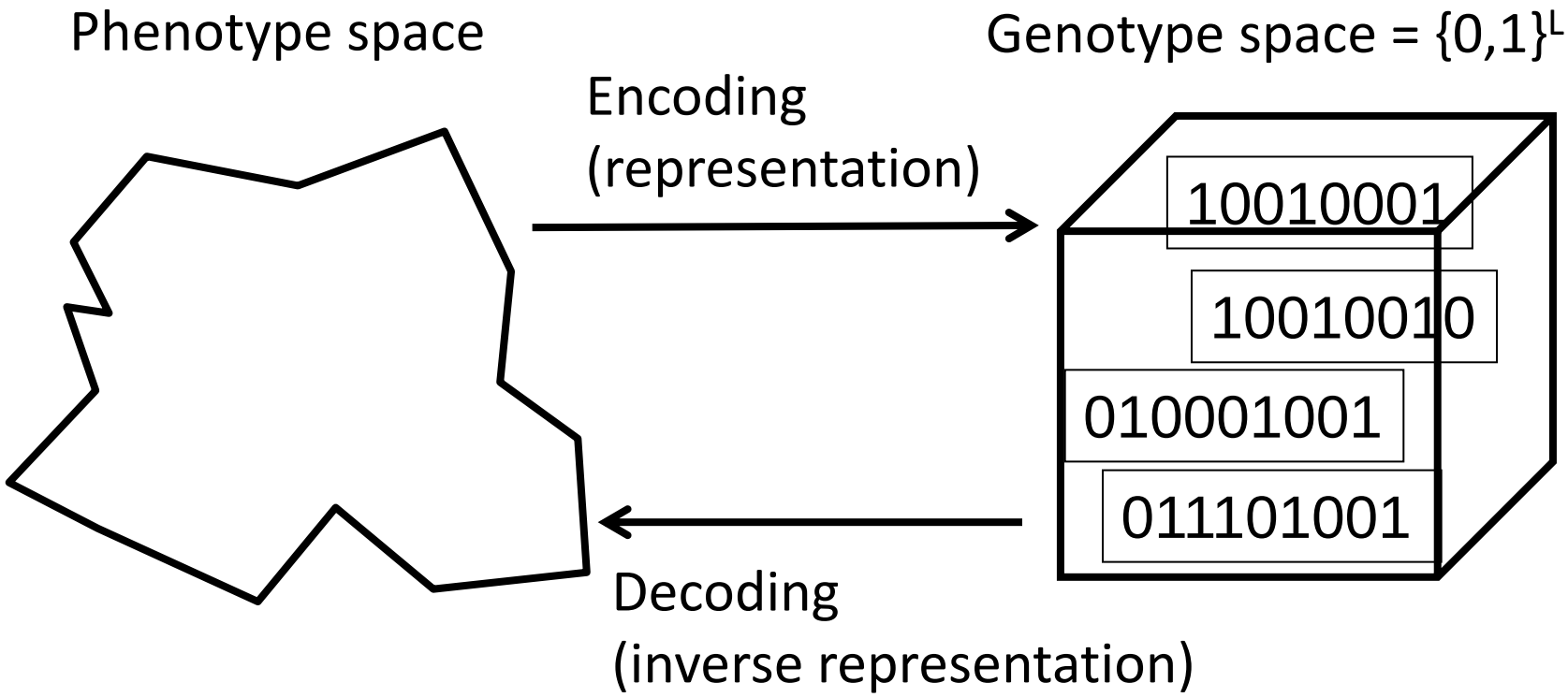


Definition of Genetic Algorithm

- A probabilistic search algorithm that iteratively transforms a set (called a population) of mathematical objects (typically fixed-length binary character strings), each with an associated fitness value, into a new population of offspring objects using
  - the Darwinian principle of natural selection, and
  - using operations that are patterned after naturally occurring genetic operations, such as crossover and mutation.

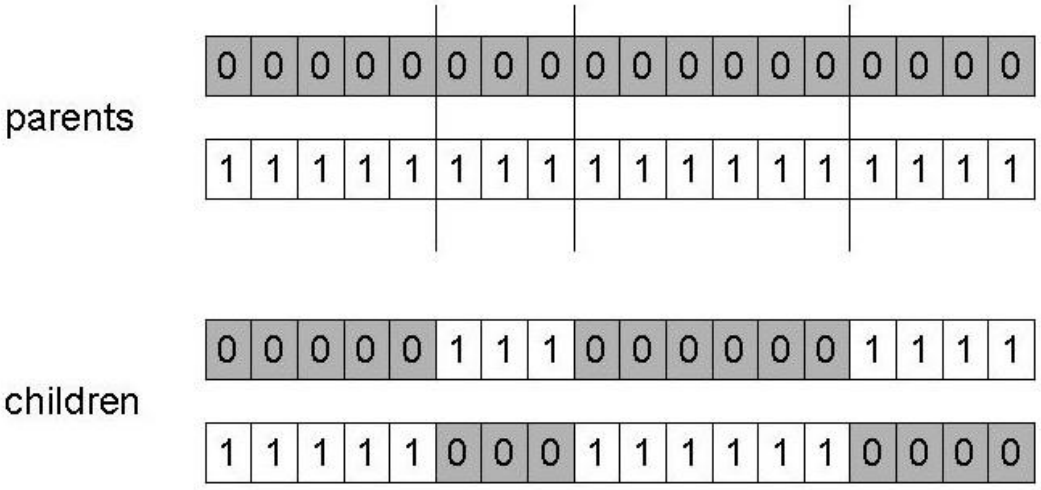
Representation

A String is called a Chromosome
Its element is called a Gen



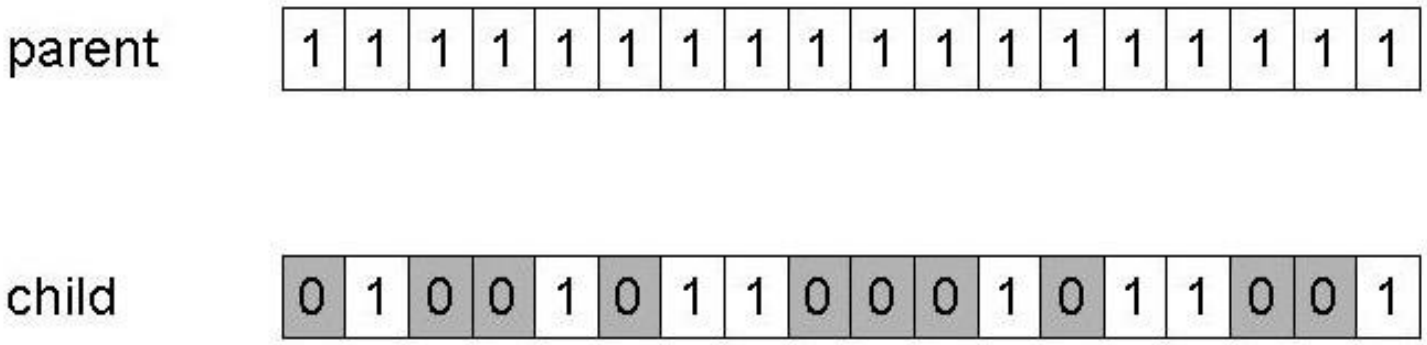
Chromosome Operators: n-point crossover

- Choose n random crossover points
- Split along those points
- Glue parts, alternating between parents
- Generalisation of one-point (still some positional bias)



Chromosome Operators: mutation

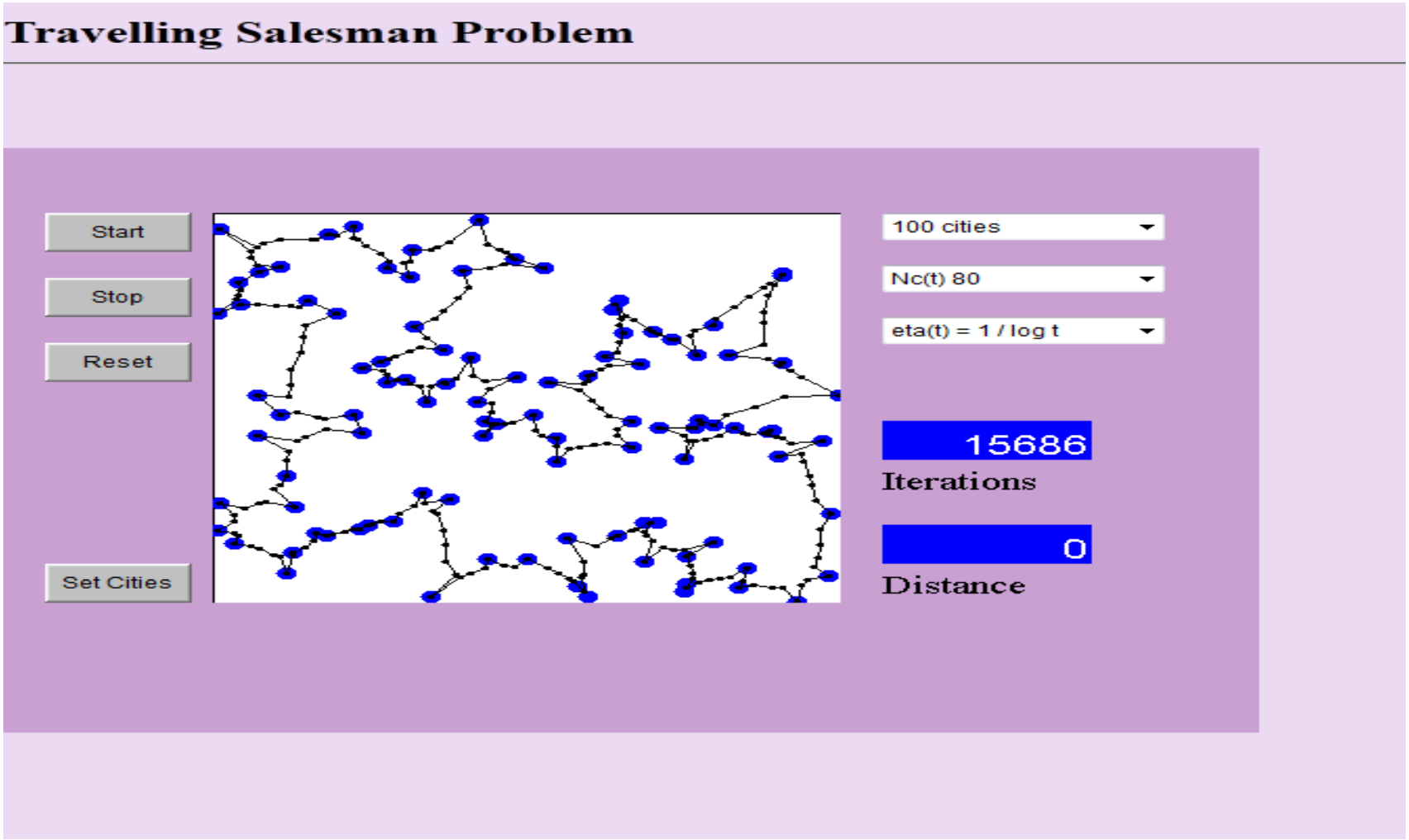
- Alter each gene independently with a probability p_m
- p_m is called the mutation rate
  - Typically between 1/pop-size and 1/chromosome-length



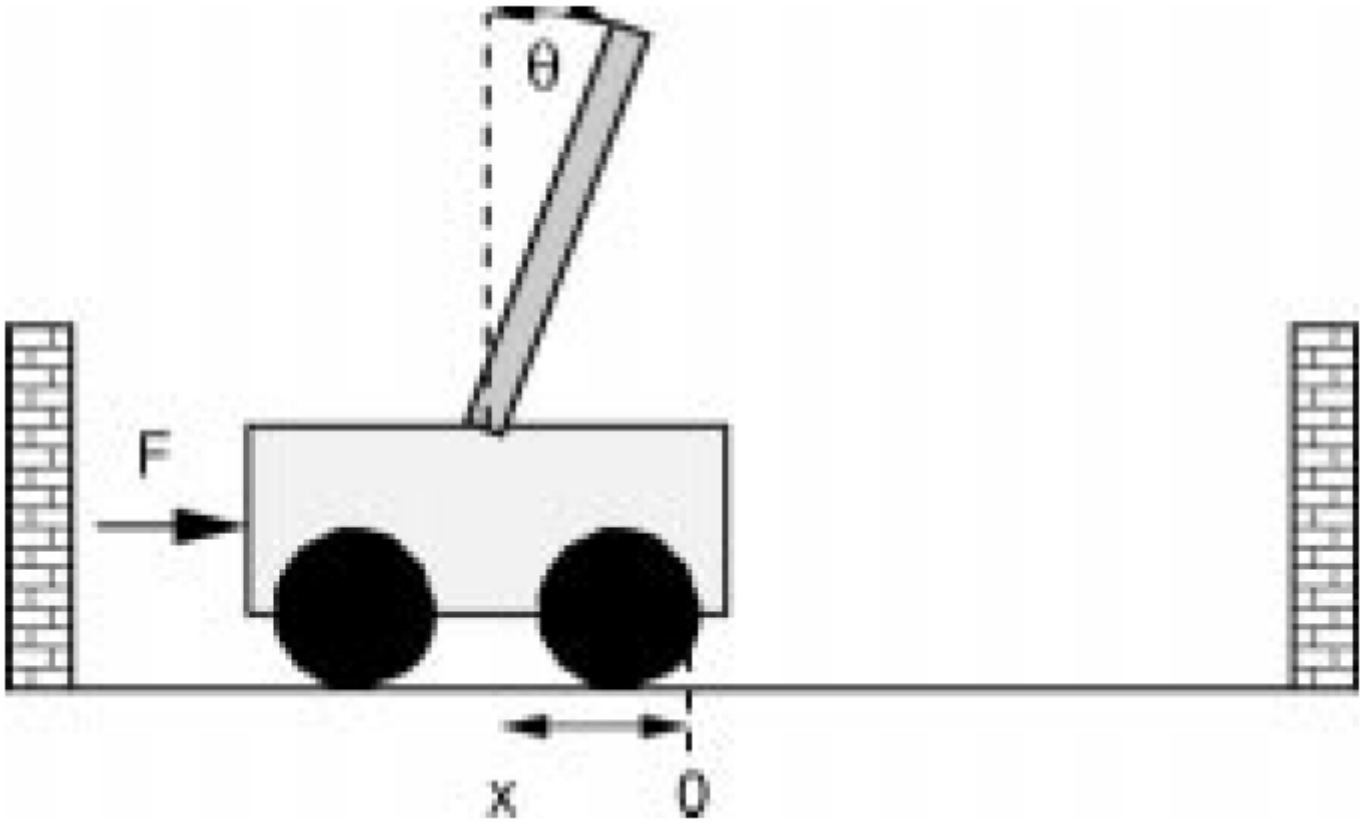
Part II: Applications of CI Techniques

- TSP Solution
- Inverted pendulum control
- Antenna design
- Newsgroups membership

Self-Organizing Feature Maps



Inverted Pendulum Control Problem



Inverted Pendulum Control Problem

The dynamics of the cart pole system are given by

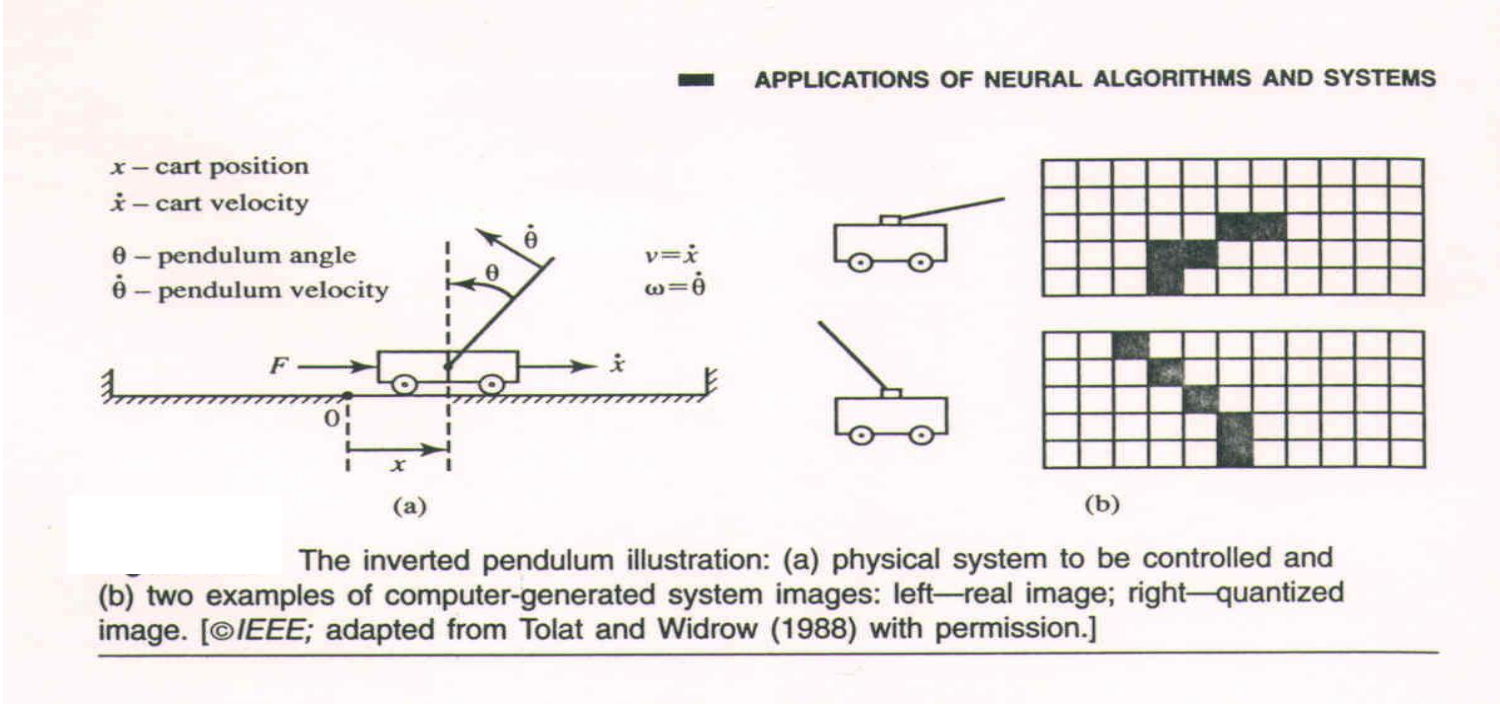
$$\dot{\theta} = \omega$$

$$\dot{\omega} = \frac{g \sin\theta - \dfrac{\cos\theta(F + m_p l \omega^2 \sin\theta)}{m_c + m_p}}{l\left(4/3 - \dfrac{m_p \cos^2\theta}{m_c + m_p}\right)}$$

$$\dot{x} = v$$

$$\dot{v} = \frac{F + m_p l(\omega^2 \sin\theta - \dot{\omega}\cos\theta)}{m_c + m_p}$$

Inverted Pendulum Control Problem:NN [18]



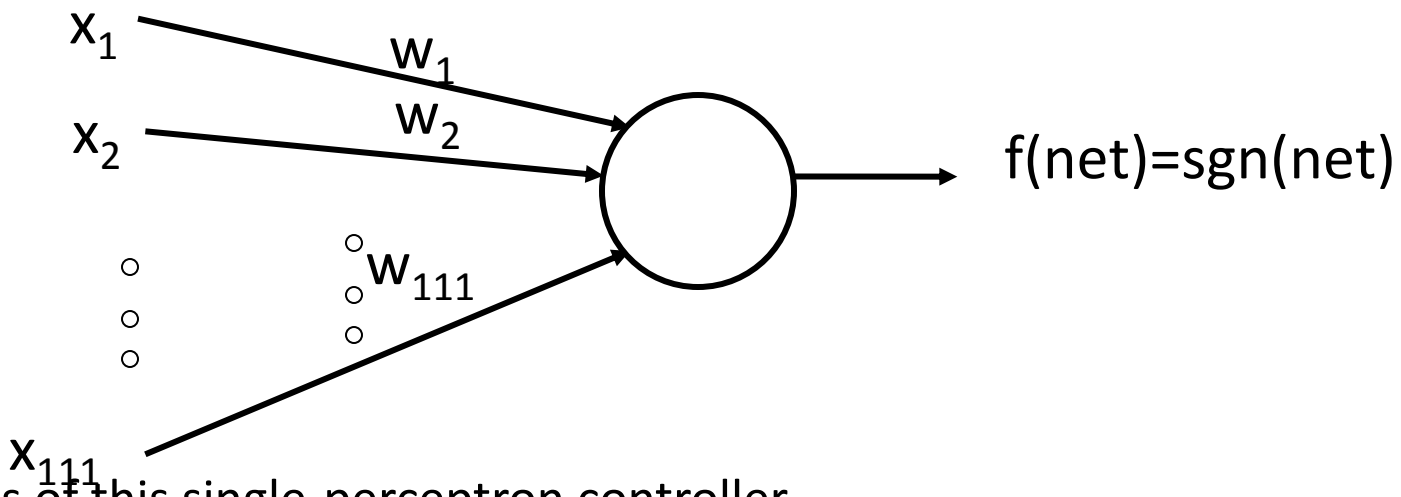
The inverted pendulum illustration: (a) physical system to be controlled and (b) two examples of computer-generated system images: left—real image; right—quantized image. [©*IEEE;* adapted from Tolat and Widrow (1988) with permission.]

$$F(t) = ax + b\dot{x} + c\theta + d\dot{\theta}$$

$$F(t) \cong \alpha\ sgn[ax + b\dot{x} + c\theta + d\dot{\theta}]$$

$$F(t) \cong \alpha\ sgn[(a+b)x - bx_{-1} + (c+d)\theta - d\theta_{-1}]$$

Inverted Pendulum Control Problem:NN

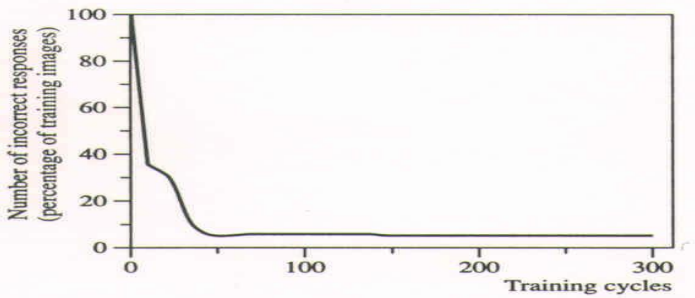


Features of this single-perceptron controller

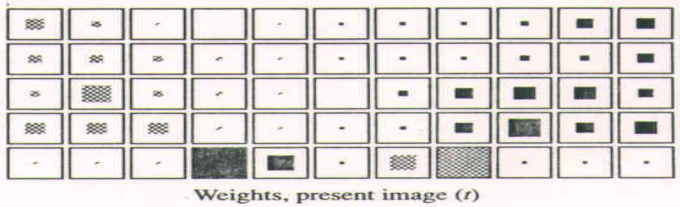
- inputs 1-55 - 'crude' present image
- inputs 56-110 -'crude' most recent image, input 111 is bias
- no need to write differential/difference equation
- no need to know mass, moments, etc
- raw and crude visual info is inserted into learning
- learning F commands issued by an intelligent observer (teacher)
- teacher knows no physics, control, neural networks

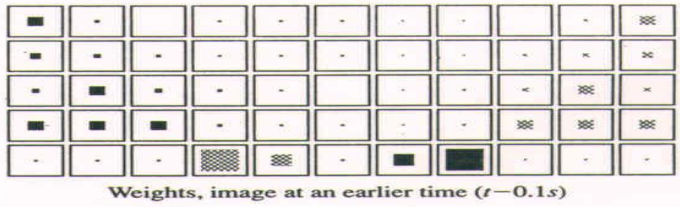
Merging of visual info with common sense learning [18]

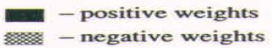
Inverted Pendulum Control Problem:NN



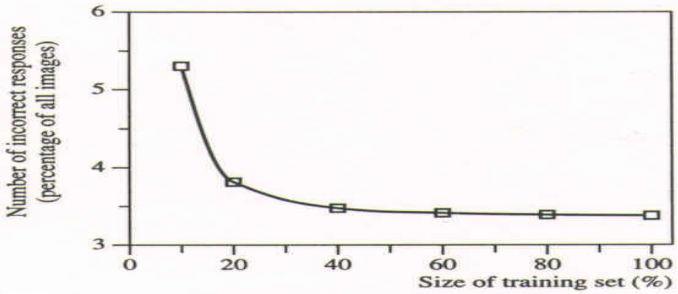
(a)







(b)



(c)

Training of the cart-pendulum controller: (a) learning profile example, (b) image of al weights, 4225 images used, and (c) analysis of generalization capability of the trained controller. [©*IEEE;* adapted from Tolat and Widrow (1988) with permission.]

What these positive weights are telling ???

At these locations at t and t-1, F is positive:

The cart requires push to the right!!! [18]

Inverted Pendulum Control Problem:FS [20]

Objective function: $J = \int_{t=0}^{\inf} x' P x + u' Q u \, dt$

State vector: $\boldsymbol{x} = (\theta, \omega, x, v)^{\mathrm{t}}$ and $\mathrm{Q} = 0.1$ (scalar)

$$P = \begin{pmatrix} 20.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 4.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.5 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.5 \end{pmatrix}$$

Inverted Pendulum Control Problem

- Common sense verbal rules

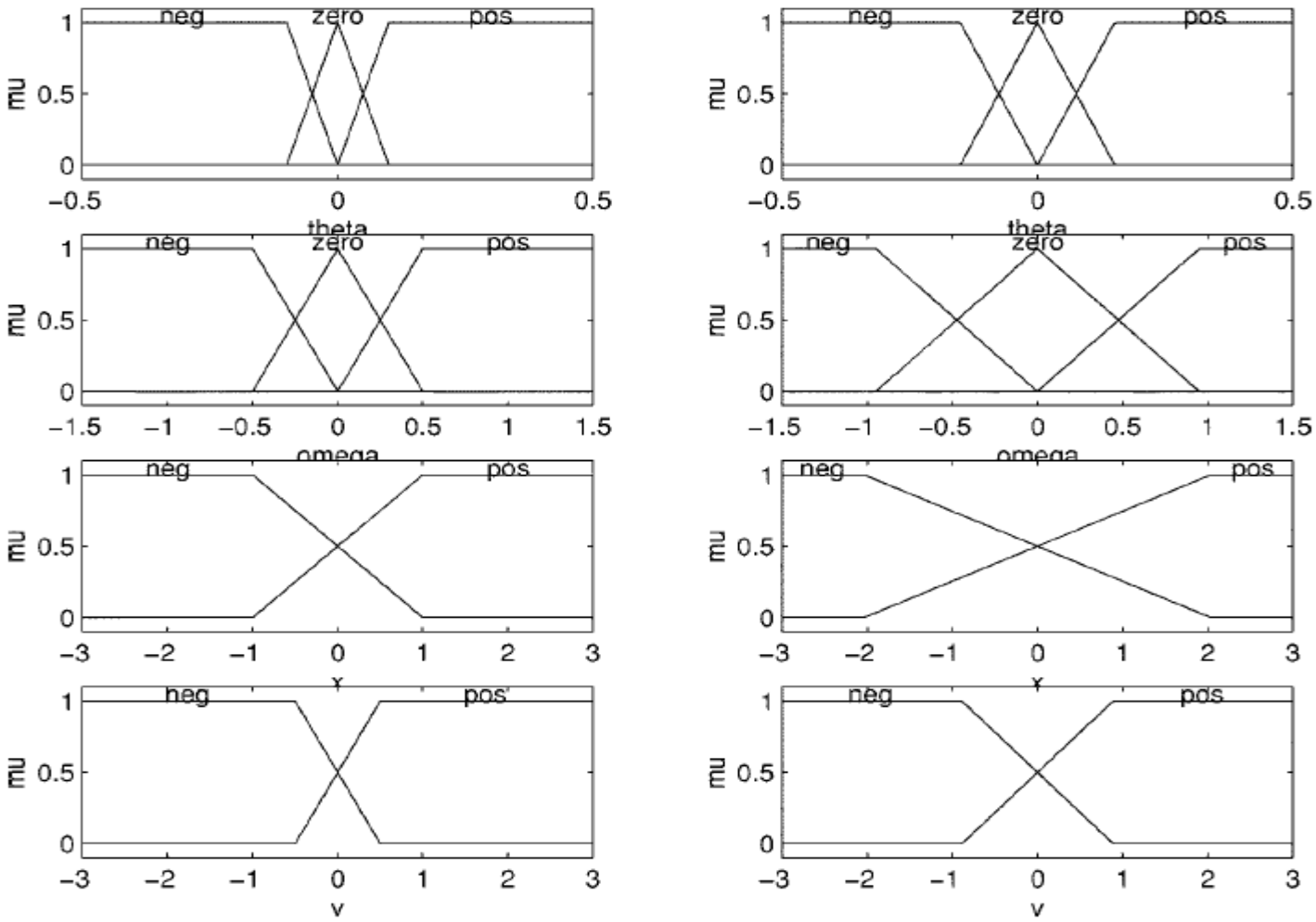
F	x is neg	θ		
	v is neg	neg	zero	pos
	neg	nb	nm	ns
ω	zero	nm	ns	ze
	pos	ns	ze	ps

F	x is neg	θ		
	v is pos	neg	zero	pos
	neg	nm	ns	ze
ω	zero	ns	ze	ps
	pos	ze	ps	pm

F	x is pos	θ		
	v is neg	neg	zero	pos
	neg	nm	ns	ze
ω	zero	ns	ze	ps
	pos	ze	ps	pm

F	x is pos	θ		
	v is pos	neg	zero	pos
	neg	ns	ze	ps
ω	zero	ze	ps	pm
	pos	ps	pm	pb

Inverted Pendulum Control Problem



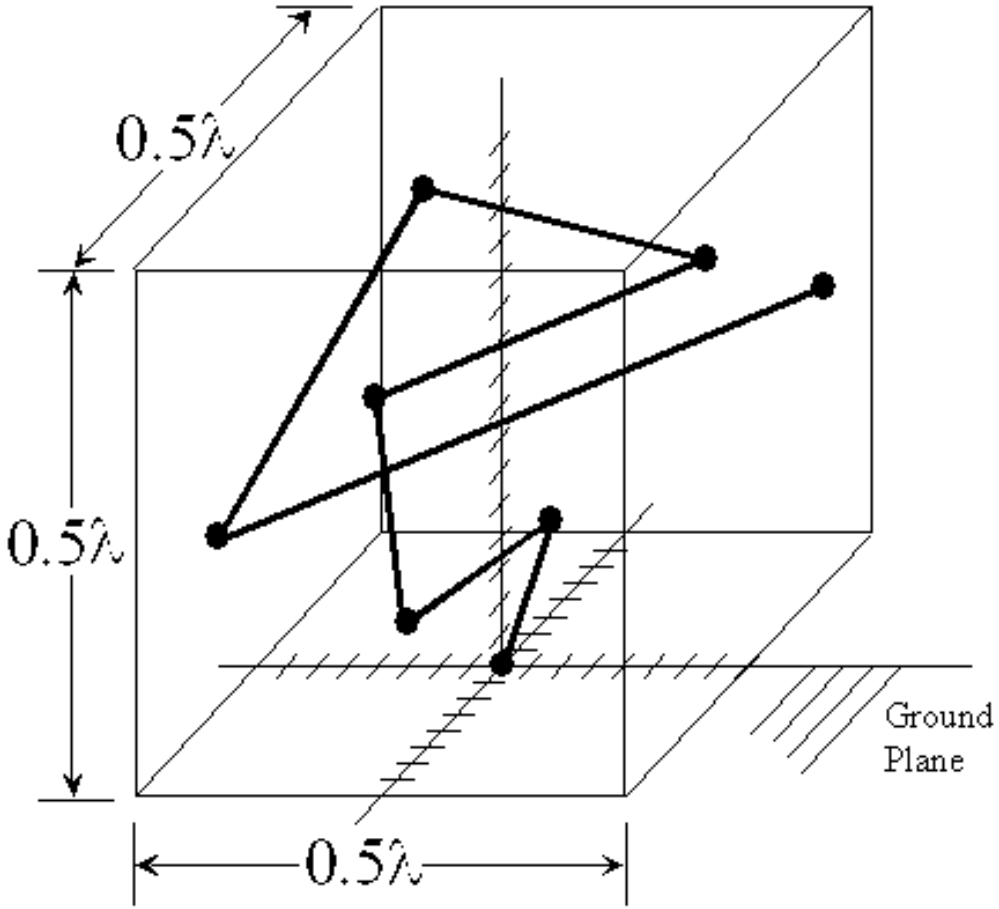
Antenna Design Specificatons [19]

- Ground-to-satellite communications antenna for cars and handsets
- We desire near-uniform gain pattern 10° above the horizon
- Fitness is measured on the antenna's radiation patterns.

Antenna Design Starting Point

- The problem (Altshuler and Linden 1998) is to determine the x-y-z coordinates of the 3-dimensional ends (X1, Y1, Z1, X2, Y2, Z2,… , X7, Y7, Z7) of 7 straight wires so that the resulting 7-wire antenna satisfies certain performance requirements
- The first wire starts at feed point (0, 0, 0) in the middle of the ground plane
- The antenna must fit inside the 0.5λ cube

Antenna: Final Design



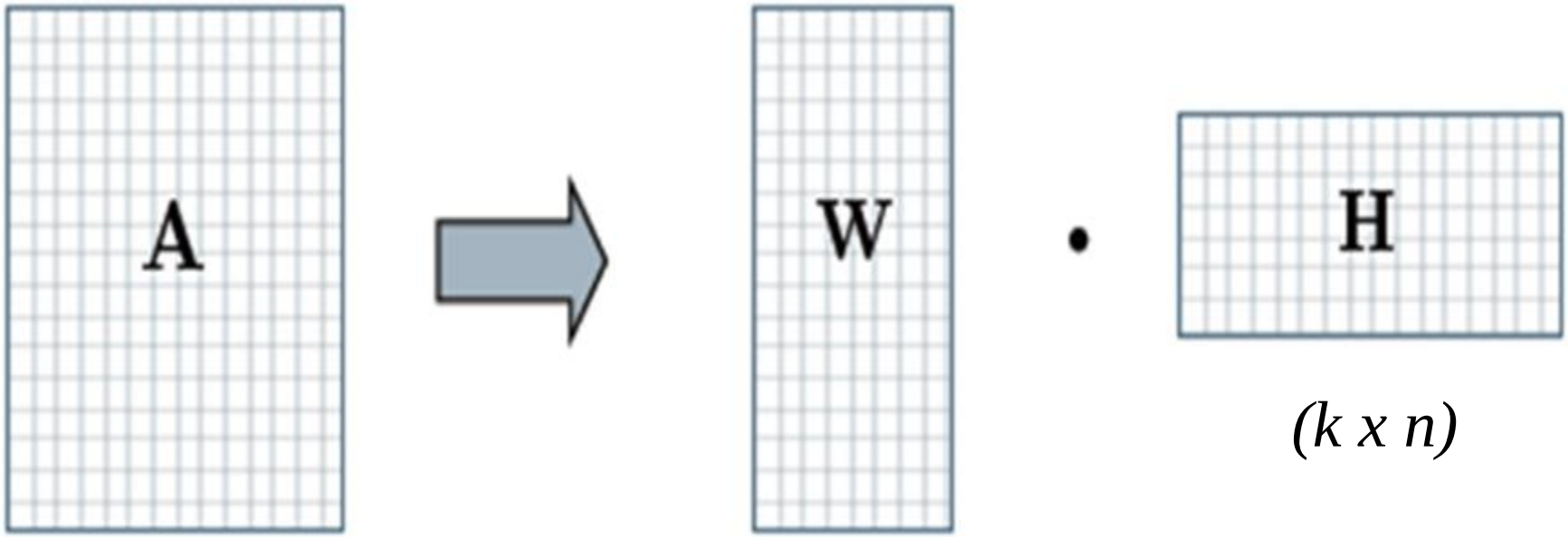
Antenna Genome

X_1	Y_1	Z_1	….	X_7	Y_7	Z_7
+0010	**-1110**	**+0001**	….	**-1011**	**+0011**	**+0011**

- 105-bit chromosome (genome)
- Each x-y-z coordinate is represented by 5 bits (4-bit granularity for data plus a sign bit)
- Total chromosome is $3 \times 7 \times 5 = 105$ bits

Matrix Factorization for Clustering

- Decompose A into component matrices W and H
- Simple problem, rich solutions/implications

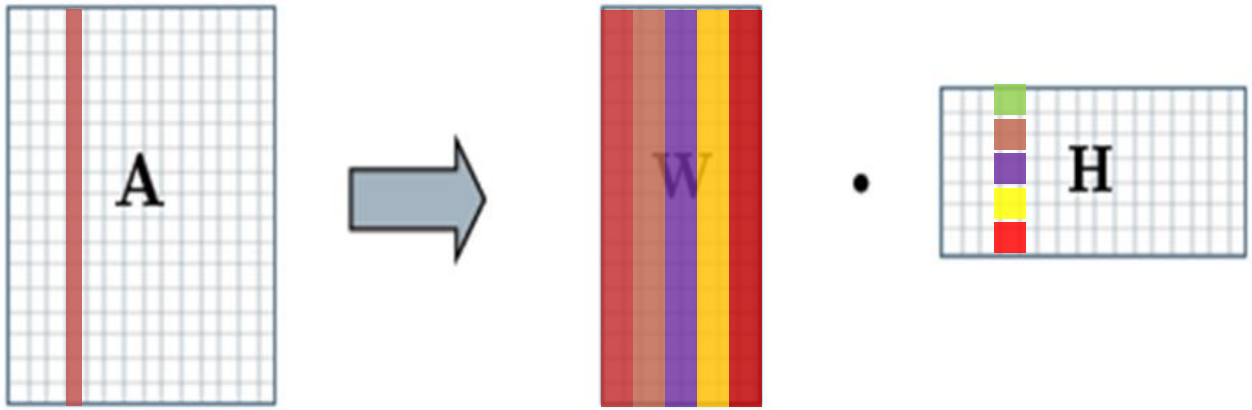


Matrix Factorization for Docs Clustering

- Every **column** A^k of A represents a **sample document**
- Decompose:

$$A \approx WH$$

- The **columns** W^j of W are **basis vectors**
- There are k **basis vectors** W^j of W (j=1,2..k)
- The **columns** of H are the **encodings** of the data:
- $A^i = \sum_j W^j H_{ji}$



Definition of NMF

- '***A***' is a given data matrix, ***A****: (m x n)*
  We are looking for ***W***: *(m x k)* ≥ 0 and ***H***: (*k x n*) ≥ 0
  where ***k*** << *min(m,n)* for the best approximation on:

$$\boldsymbol{A \approx W H}$$

min $\|\boldsymbol{A} - \boldsymbol{W}\boldsymbol{H}\|^2_F$ → Objective (Cost) Function

W → Basis vectors matrix, tend to be sparse
H → Encoding matrix, usually also sparse
(nonnegative lower dimensional representation)
k → Low-rank value

Document Clustering Using NMF [22]

1 ➢ **Example:** MATLAB Output after using NMF Clustering (**k=10** clusters) for 20-Newsgroups Data Set:
religion christian peopl god line detail valu moral server scienc talk object jesus saw mac built arab frank dwyer configur → **RELIGION**

2 name uk mathew shall folk tree righteous pin speed ram ps mb meg isa centri slot ns simm

3 mail drive help pleas info anybodi video manufactur monitor vga → **COMPUTER**

4 name com server file help sandvik newton appl kent ignor spread window

5 stein brad kill guess imagin final water org reveal river sourc israel isra arab civilian ncsu mb alan norton lebanon hasan nysernet hernlem lebanes

6 net know object option thank summar advanc compil righteous anybodi driver latest site ftp ati window bio

7 avail price street charg card uk mathew cost sorri plus display fix driver super vga ati ultra mb ship diamond beast armenian

8 atheism version atheist exist god stein edu answer cs keith ve ac charley wingat mango umd contradictori imag ultb isc rit mozumd il

9 version word god rather man brad keep shall said hear turkish org tree heart righteous receiv luke bless davidian ps isa turkey armenian sdpa armenia urartu → **POLITICS**

10 card file po cwru hear format mous summar compil islam convert job email muslim luke bless bus diamond slot

Concluding Remarks

Learning-based CI/AI techniques for data analysis increasingly

- Powerful
- Less costly
- Easier to use, with better UI
- Standardized, often as software packages (ex. MATLAB, and many other tools available)
- Flexible

Concluding Remarks

- Thank you for your attention and interest

  www.jacekzurada.org

  jacek.zurada@louisville.edu

  or

  j.zurada@ieee.org

References

[1] Wang J. and Kusiak A., *Computational Intelligence in Manufacturing Handbook*, CRC Press, 2001.

[2] Dudek G. And Jenkin M., *Computational Principles of Mobile Robotics,* Cambridge University Press, 2010.

[3] Zurada J. M., *Introduction to Artificial Neural Systems*, St. Paul, West, 1992.

[4] L. C. Jain, *Applied Genetic Programming and Machine Learning*, CRC Press 2010.

[5] Kordon K. A., *Applying Computational Intelligence*, Springer, 2010.

[6] Huang T. W., Yeh S. Y., and Ho T. Y., *A Network-Flow Based Pin-Count Aware Routing Algorithm for Broadcast Electrode-Addressing EWOD Chips*, IEEE Int. Conf. on Computer-Aided Design, USA, 2010.

[7] Bullinaria J. A. and Li X., A*n Introduction to Computational Intelligence Techniques for Robot Control*, Industrial Robot: An International Journal, Vol. 34 Iss: 4, pp.295 – 302 (2007)

[8] Harley R. G. and Liang J, *Computational Intelligence in Smart Grids*, IEEE Symposium Series on Computational Intelligence (SSCI), 11-15 (2011)

[9] Paraskevi Z., *Robot Handling Fabrics Towards Sewing Using Computational Intelligence Methods,* Robotic Systems - Applications, Control and Programming, InTech Publication, February 2012.

[10] Alippi C., Pelosi G., and Roveri M., *Computational Intelligence Techniques to Detect Toxic Gas Presence*, IEEE Int. Conf. on Computational Intelligence for Measurement Systems and Applications, Spain, 12-14 July, 2006.

References

[11] Ruz G. A., Estevez P. A., and Ramirez P. E., *Automated Visual Inspection System for Wood Defect Classification Using Computational Intelligence Techniques*, International Journal of Systems Science, Vol. 40, No. 2, pp. 163-172, February 2009.

[12] Jabbari A., Jedermann R., and Lang W., *Application of Computational Intelligence for Sensor Fault Detection and Isolation*, World Academy of Science, Engineering and Technology 33, 2007.

[13] Venayagamoorthy G. K., *Potentials and Promises of Computational Intelligence for Smart Grids*, IEEE Symposium on Computational Intelligence Applications in Smart Grids, France, July 26-30, 2009.

[15] Russel S., Norvig P., Artificial Intelligence – A Modern Approach, Prentice Hall, 1995

[16] Frank Hoffmann, "Evolutionary algorithms for fuzzy control system design", Proceedings of the IEEE, vol. 89, no. 9, Sep. 2001

[17] Petr Kadlec, Bogdan Gabrys, and Sibylle Strandt, "Data-driven soft sensors in the process industry", Computers & Chemical Engineering, 33(4):795–814, 2009

[18] V. Tolat, B. Widrow, "An adaptive 'broom balancer' with visual inputs", IEEE International Conference on Neural Networks, Vol. 2, 641-647, San Diego, 1988

[19] Altshuler; Edward E. and Linden; Derek S., "*Process for the Design of Antennas using Genetic Algorithm*", United States Patent 5,719,794, February 17, 1998.

[20] Frank Hoffmann, "Evolutionary algorithms for fuzzy control system design", Proceedings ofthe IEEE, invited paper, vol. 89, no. 9, Sep. 2001,

[21]Creech, G.L., Zurada, J.M., Aronhime, P.B., Feedforward Neural Networks for Estimating IC Parametric Yield and Device Characterization, Proc. of the 1995 IEEE International Symposium on Circuits and Systems, Seattle, Washington, April 29 - May 3, 1995, vol. 2, pp. 1520-1523

[22] Tolga Ensari, Jan Chorowski, Jacek M. Zurada, "Correntropy-based Document Clustering via Nonnegative Matrix Factorization", ICANN September 2012, Switzerland