

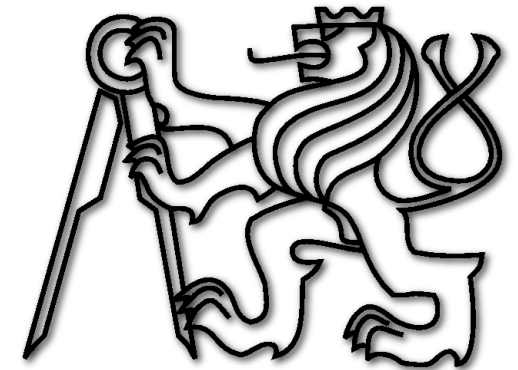
Artificial Neural Networks

Universal Approximation, RBF & GMDH



Jan Drchal
drchajan@fel.cvut.cz

*Computational Intelligence Group
Department of Computer Science and Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague*



Outline

- Universal Approximation.
- RBF networks.
- GMDH, GAME – inductive modelling.

Universal Approximation Property

Artificial Neural Network functions as a transformation T of an input vector X to output vector:

$$\vec{Y} = T(\vec{X}).$$

What transformations are possible with neural networks? Can we represent ANY function, if so → **universal approximation property**

Hilbert's Problems

- List of 23 so-called Hilbert's problems were introduced by David Hilbert in 1900 at Paris conference of the *International Congress of Mathematicians*.
- Greatest unsolved math problems of that time.
- Many of the problems are solved by now.

http://en.wikipedia.org/wiki/Hilbert_problems

Hilbert's Thirteenth Problem

$$x^7 + ax^3 + bx^2 + cx + 1 = 0$$

- Can the solution x (a function of a , b and c) be expressed using finite number of functions of two variables?
- More general: can *any* continuous function of three variables be expressed as a *summation* and *composition* of finitely many continuous functions of two variables?



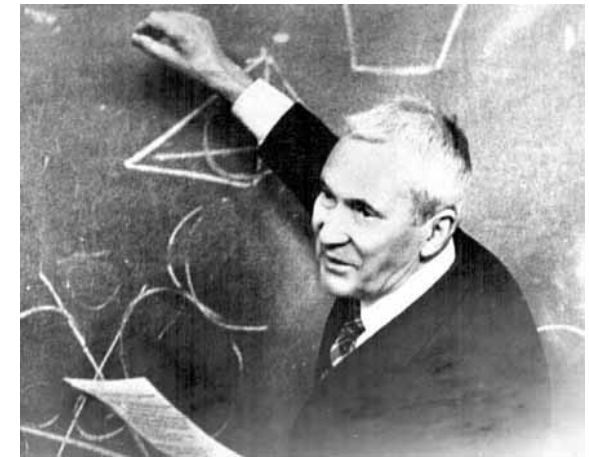
http://en.wikipedia.org/wiki/Hilbert's_thirteenth_problem

Kolmogorov's Theorem

Any continuous function f , having $n \geq 2$ variables, defined in n -dimensional unit hypercube can be expressed as:

$$f(x_1, \dots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^n \Psi_{pq}(x_p) \right)$$

where Ψ_q and Φ_{pq} are continuous functions of a single variable, $p = 1, \dots, n$ and $q = 1, \dots, (2n + 1)$.



Sprecher/Lorentz

More general modification:

$$f(x_1, \dots, x_n) = \sum_{q=1}^{2n+1} \Phi \left(\sum_{p=1}^n \lambda^{pq} \Psi_q(x_p) \right)$$

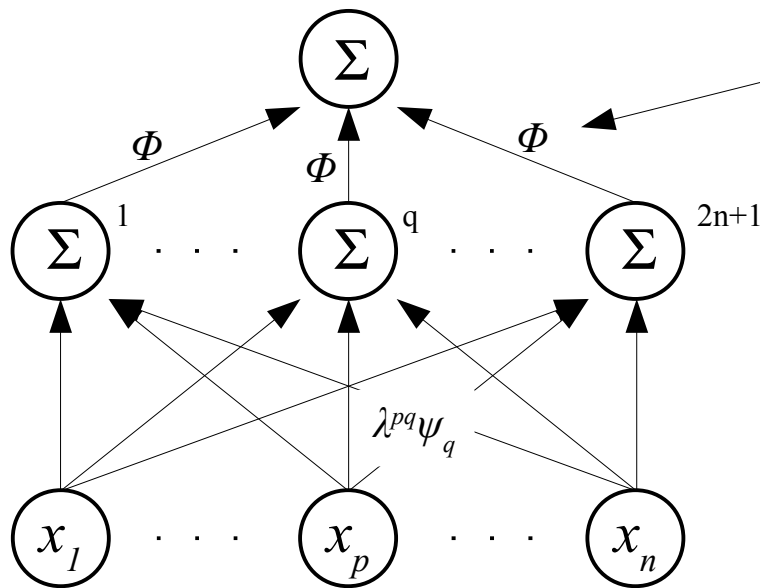
- All Φ_q replaced by single Φ .
- Ψ_{pq} replaced by $\lambda^{pq} \Psi_q$ where λ^{pq} is a real constant and Ψ_q is a monotonically increasing function.
- Ψ_q does not depend on $f \rightarrow$ they are same for given dimension n .

Hecht-Nielsen

- 1987, the expression

$$f(x_1, \dots, x_n) = \sum_{q=1}^{2n+1} \phi \left(\sum_{p=1}^n \lambda^{pq} \Psi_q(x_p) \right)$$

describes a three layer network:



Only Φ depends on f , other parameters are fixed for given dimension.

So what about MLPs?

- 1992, Kůrková:
 - approximate f with arbitrary precision,
 - two hidden layers approximating Ψ and Φ are enough.
 - gives also upper estimates on number of neurons.

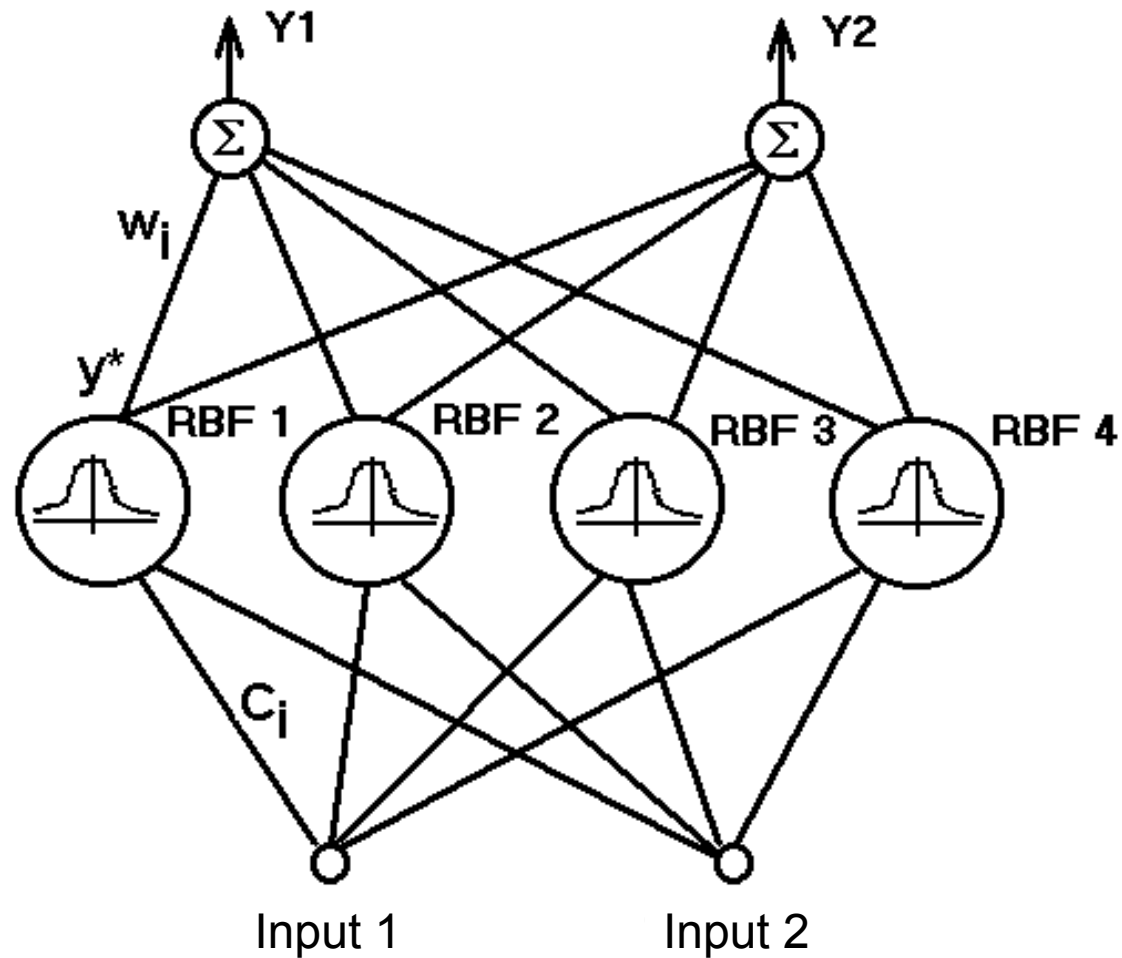
What about Single Hidden Layer?

- Hornik 1991: For *continuous, bounded and non-constant* activation functions a MLP with **single hidden** layer can approximate any continuous function with arbitrarily precision (with enough neurons in the hidden layer).
- Leshno, Lin, Pinkus & Schocken 1993: non-polynomial activation function suffices.

Radial Basis Function (RBF) Networks

- Two layers, different types of neurons in each layer.
- Feed-forward.
- Supervised Learning.
- Broomhead, Lowe (1988).
- Universal approximator.

RBF Architecture



RBF Neurons

- **Hidden layer:**

- inner potential

$$\phi = \sqrt{\sum_{i=1}^n (x_i - c_i)^2}$$

- non-linear activation function

$$y^* = f(\Phi)$$

- **Output layer:**

- linear perceptron

$$y = \sum_{i=1}^n w_i y_i^*$$

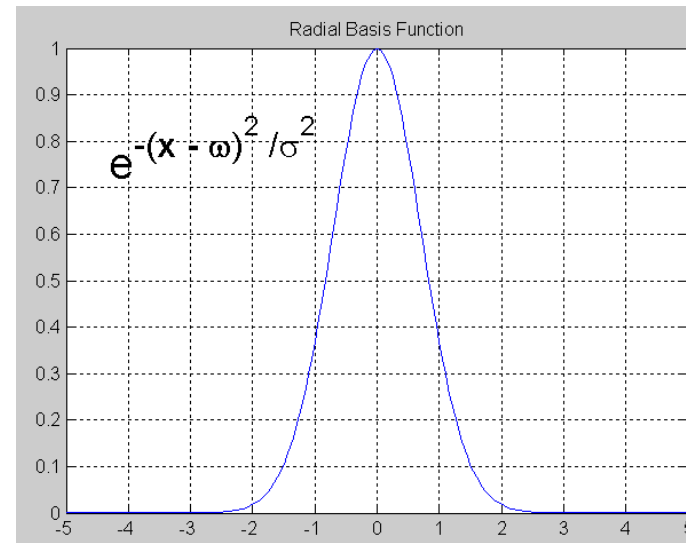
Typical RBF Activation Functions

$$y^* = \varphi$$

$$y^* = \varphi^2 \log \varphi$$

$$y^* = (\varphi^2 + \beta)^\alpha, \beta \geq 0, 0 < \alpha < 1$$

$$y^* = e^{-\frac{\varphi^2}{\sigma^2}}, \sigma > 0$$



Sphere of Influence

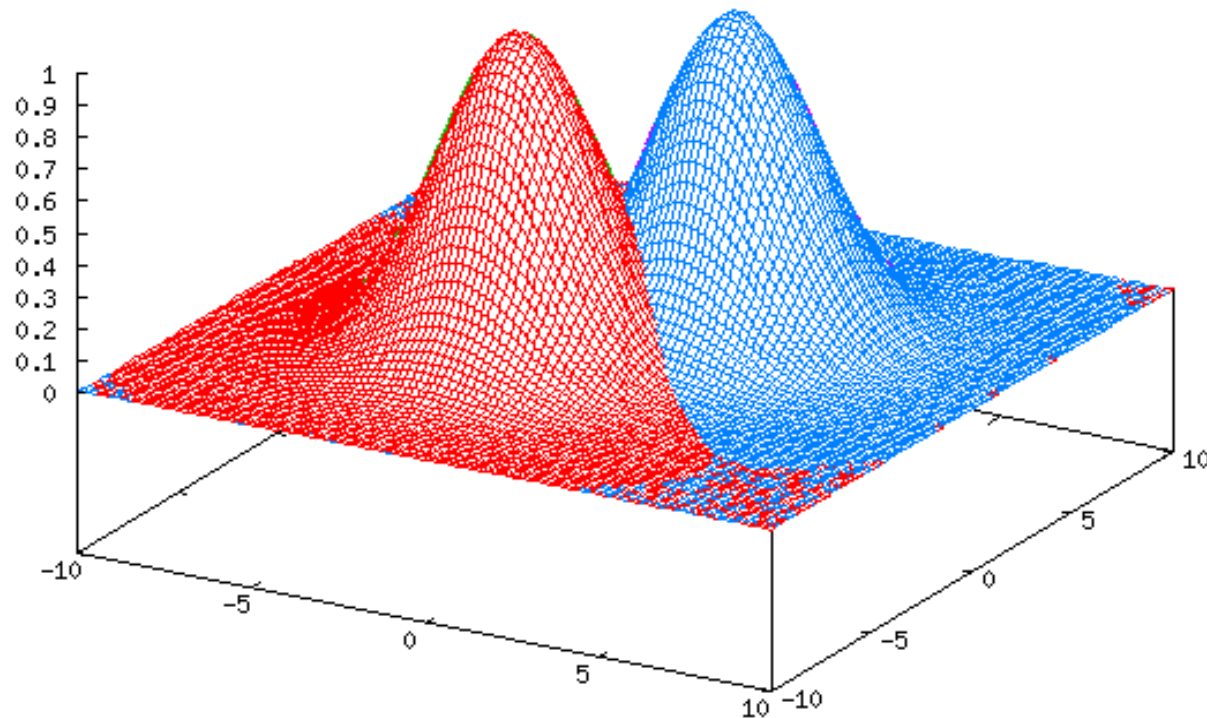
- Hypersphere with center C and radius R .
- Determined by euclidean metric.
- The center is called **the prototype**.

- The prototype represents a cluster of input data.

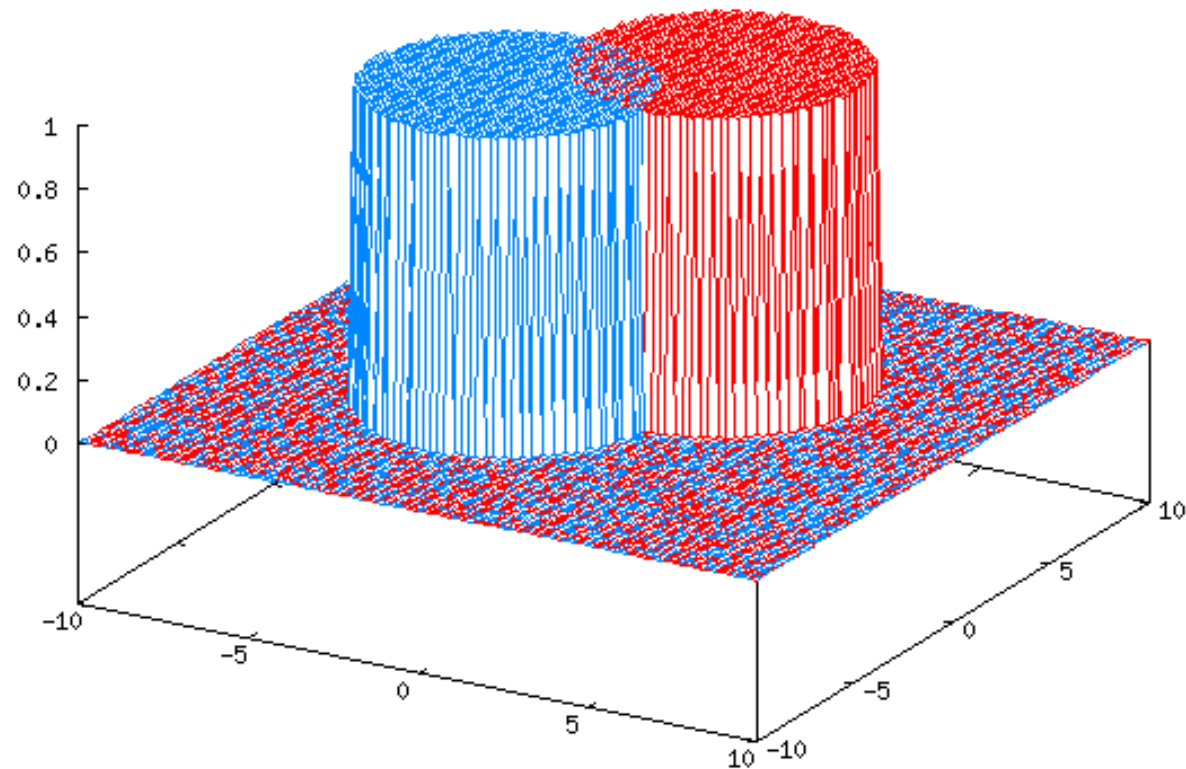
How to Determine the Sphere of Influence?

- Mostly we use Gauss function:
 - If an input vector has same weights as the prototype ($\varphi = 0$), the function gains maximum ($= 1$). This is also the maximum activation value of the neuron.
- With increasing distance of input vector from the prototype the activity of neuron decreases.
- σ (which corresponds to a variance of normal distribution) determines a gain (width) of the activation function.

Sphere of Influence: Example



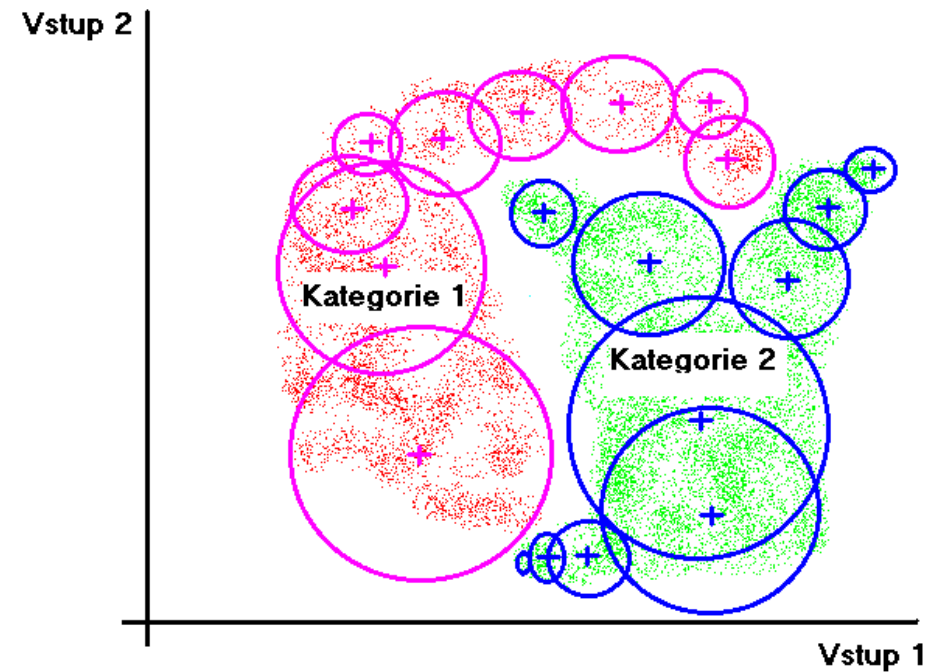
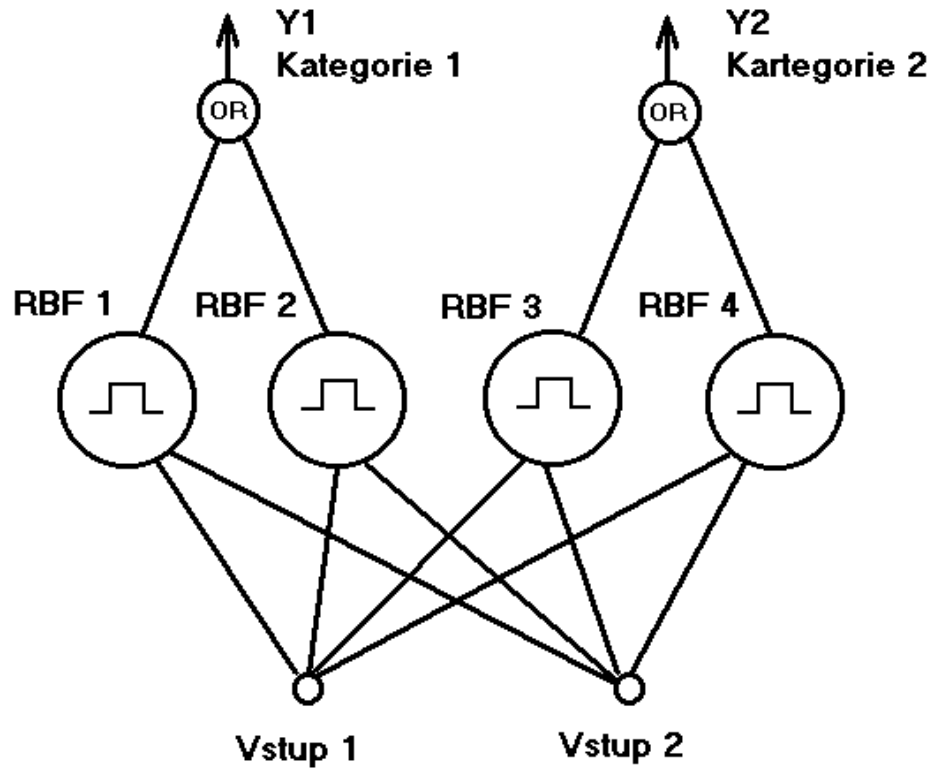
Sphere of Influence for Hardware Implementations



Neuron Types Discussion

- RBF neurons:
 - inner potential is a measure of distance between input vector and the prototype (represented by the weights),
 - activation function delimits the sphere of influence.
- Output neurons:
 - accumulate RBF neurons' outputs to achieve accurate approximation, OR
 - perform union of sets for classification.

RBF as a Classifier



Application Areas

- Same as MLPs.
- Universal approximator (1991: Park, Sandberg).

Training RBF Networks

- Two phases:
 - training prototypes,
 - training output neurons.
- Training prototypes:
 - Unsupervised: Cluster Analysis.
- Training output neurons:
 - Supervised.

$$F(\mathbf{x}) = \sum_{i=1}^K w_i \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_i\|^2}{2\sigma_i^2}\right)$$

3
supervised

2
unsupervised

1

3-steps method:

Training Prototypes I

- Estimate the number of clusters in input data :(
- Define a *membership function* $m(\mathbf{x})$:
 - determines whether an input pattern belongs to a certain cluster,
 - choose cluster by the closest prototype.
- Estimate coordinates of all vectors \mathbf{C}_p which correspond to cluster centers:
 - we use a well-known K-means algorithm.

Training Prototypes II: K-means

- **Steps of K-means algorithm:**
 - randomly initialize centers \mathbf{C}_i of RBF neurons,
 - *) evaluate $m()$ for all input patterns,
 - determine a new center \mathbf{C}_k as an average of all patterns which belong to cluster k according to $m()$.
 - stop when $m()$ does not change anymore, otherwise go to *)

Training Prototypes III

- There exists also an *adaptive version* of the previous algorithm:
 - reminds perceptron learning algorithms (i.e. delta rule),
 - moves a center closer to input vectors.

Training Prototypes IV: Adaptive K-means

- **Steps of adaptive K-means:**
 - randomly initialize centers \mathbf{C}_i ,
 - *) take input pattern \mathbf{X} ,
 - determine closest center \mathbf{C}_k and change its coordinates according to:
$$\bar{C}_k^{(t+1)} = \bar{C}_k^t + \eta (\bar{X}^{(t)} - \bar{C}_k^t)$$
where η denotes adaptation rate, which decreases with successive iterations.
 - stop when $\eta = 0$ or after certain # of iterations. Otherwise continue with *).

Training Prototypes V: Estimate # of Clusters

- Start with zero number of clusters,
- *) take input pattern \mathbf{X} ,
- Find the closest cluster \mathbf{k} , If the distance between \mathbf{C}_k and $\mathbf{X} <$ some constant r , modify:

$$\bar{C}_k^{(t+1)} = \bar{C}_k^t + \eta (\bar{X}^{(t)} - \bar{C}_k^t)$$

- If the distance $> r$, create a new center with coordinates of \mathbf{X} :

$$\bar{C}_k^{t+1} = \bar{X}^{(t)}$$

stop when $\eta = 0$ or after certain # of iterations. Otherwise continue with *).

- How to choose r ?

Training Prototypes V: Determine σ

- Parameter σ is determined by a root mean squared deviation (RMSD) of distance between patterns and cluster center:

$$\sigma_k = \sqrt{\frac{1}{Q} \sum_{q=1}^Q \|\bar{\mathbf{C}}_k - \bar{\mathbf{X}}_q\|^2}$$

- where \mathbf{X}_q is q -th pattern which belongs to cluster with center \mathbf{C}_k .

Other approach used fixed σ .

Training Output Neurons

- Find weights by minimization of this energy function:

$$\Delta \bar{w}^{(t)} = -\eta \nabla E^{(t)} = \eta (D^{(t)} - Y^{(t)}) Y^{*(t)}$$

- Is it familiar?

$$E = \frac{1}{2} \sum_{t=1}^m \sum_{i=1}^n \left(d_i^{(t)} - y_i^{(t)} \right)^2$$

MLP vs. RBF

- **Learning**

- RBF learns much faster.

- **Applications**

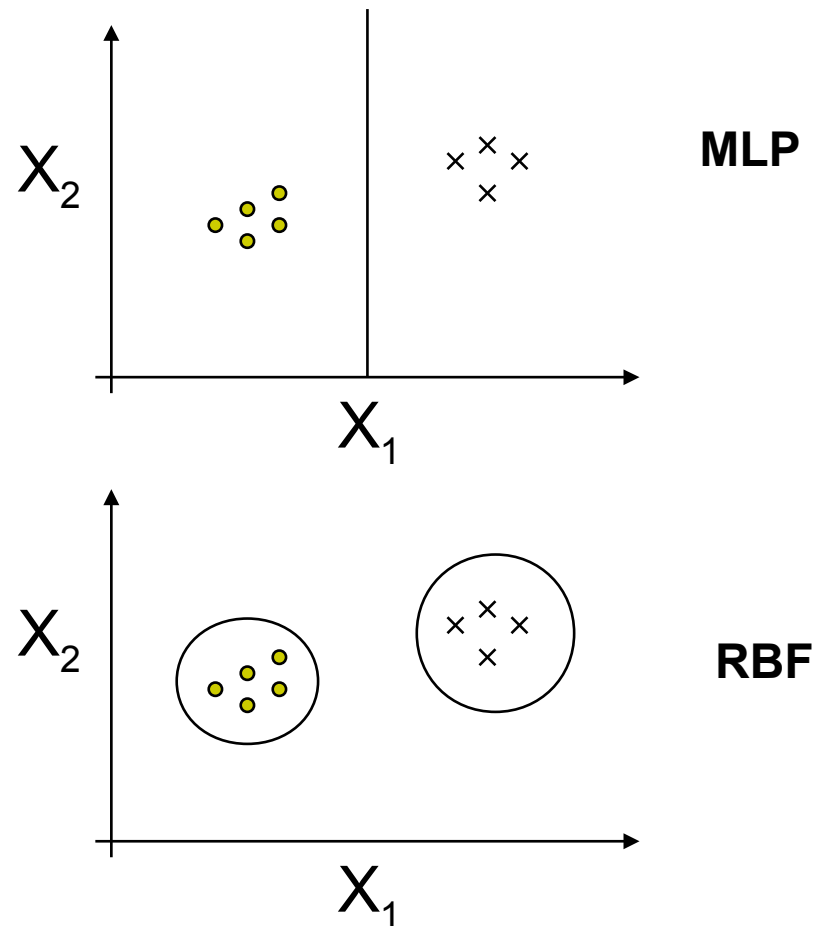
- MLP/RBF usable for regression & classification

- **Properties**

- both are universal approximators.

- **As classifiers:**

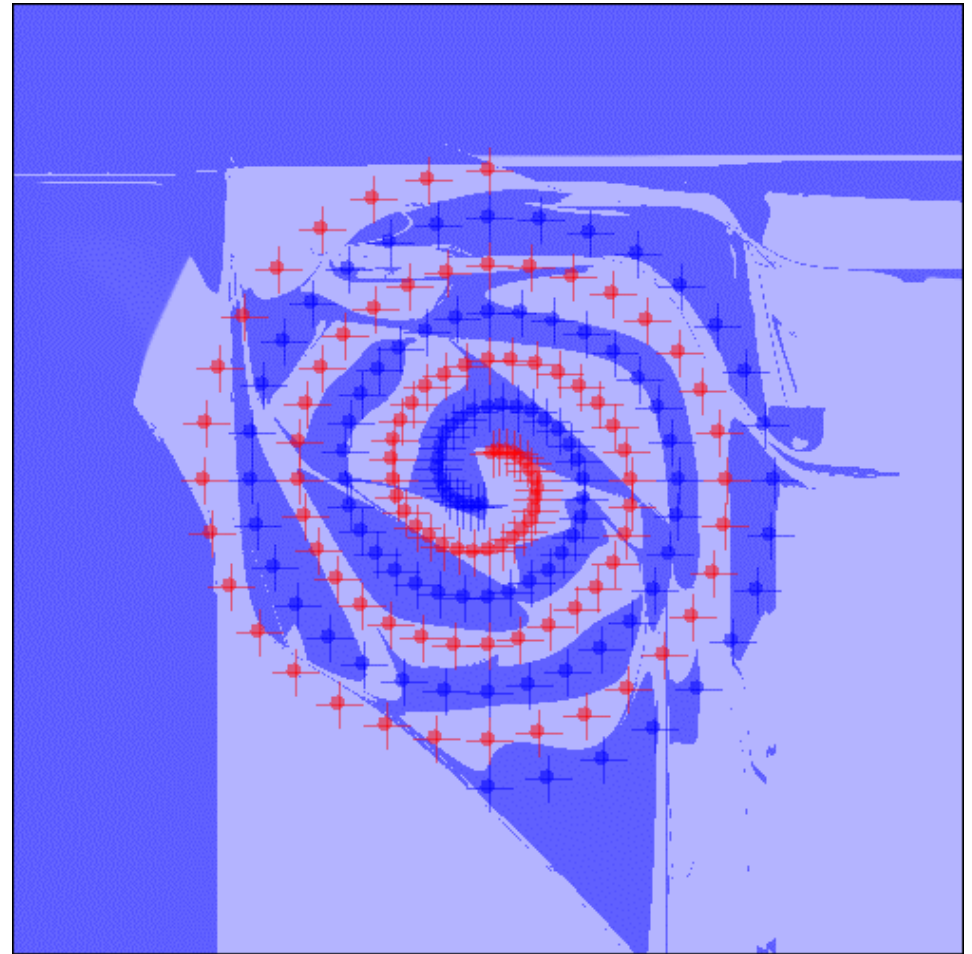
- MLPs - hyperplanes,
- RBFs - hyperspheres.



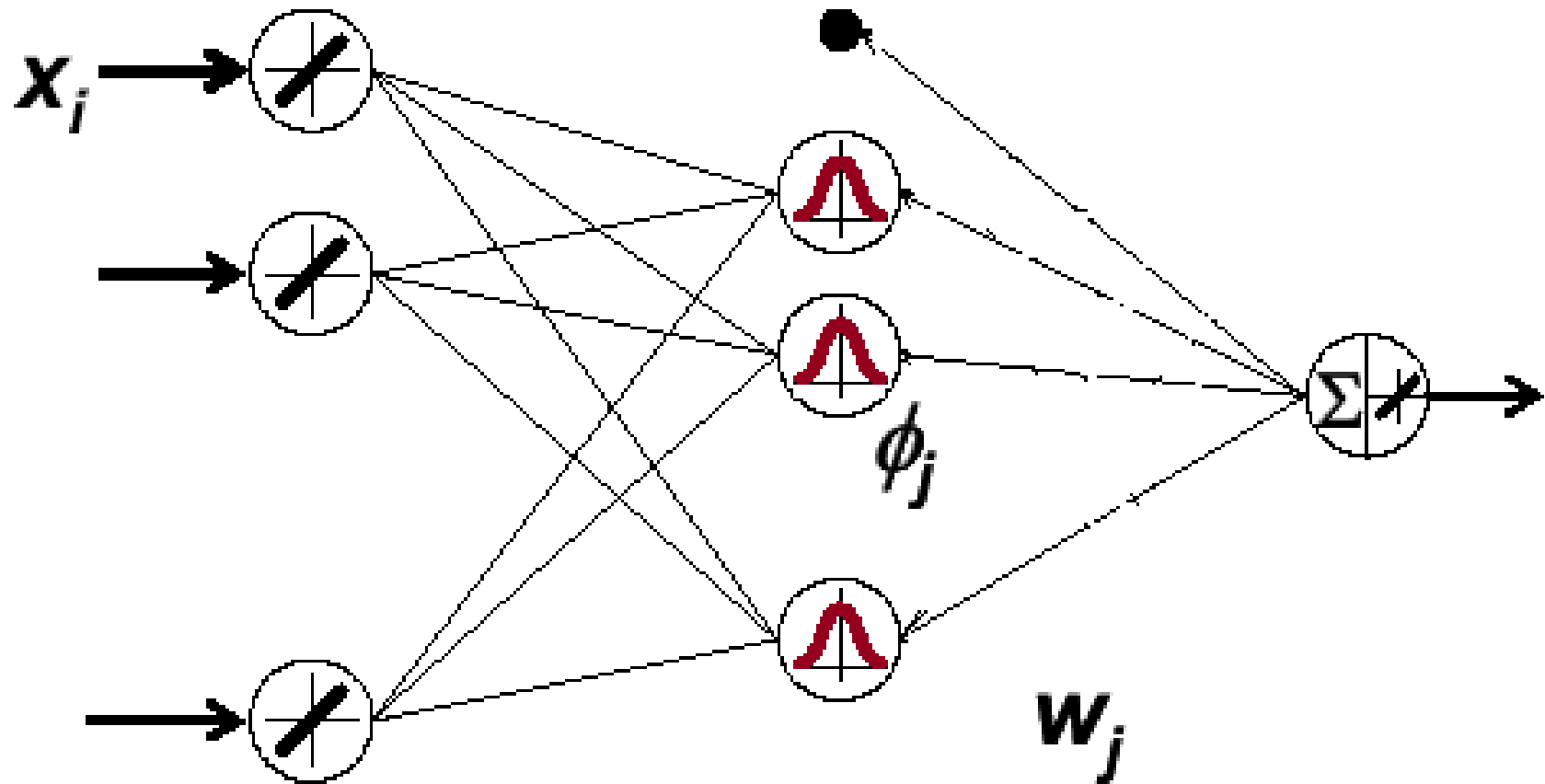
The Spiral problem

One of the most challenging artificial benchmarks: separate two inter-winded spirals.

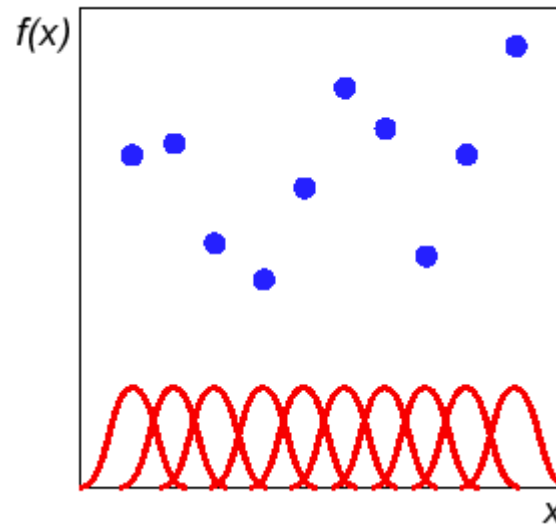
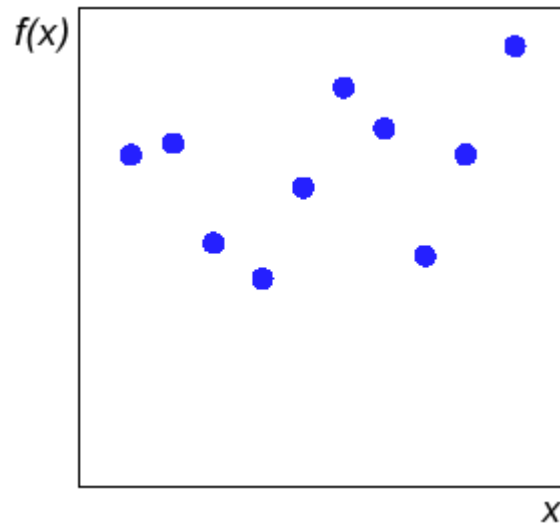
How can MLP solve it?
How about RBF?



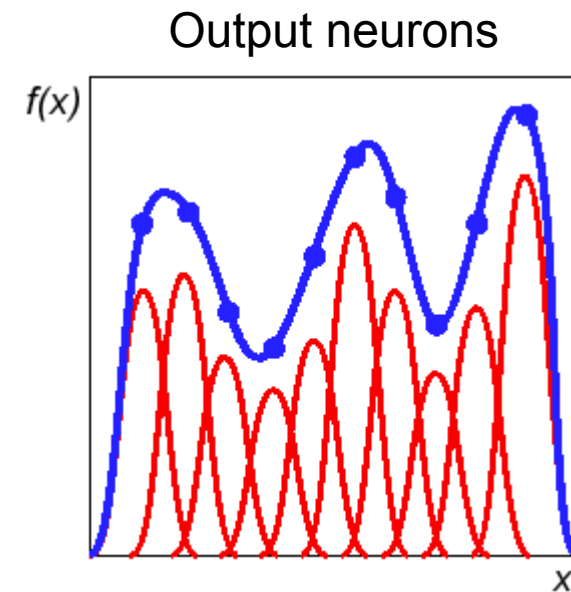
RBF for Approximation



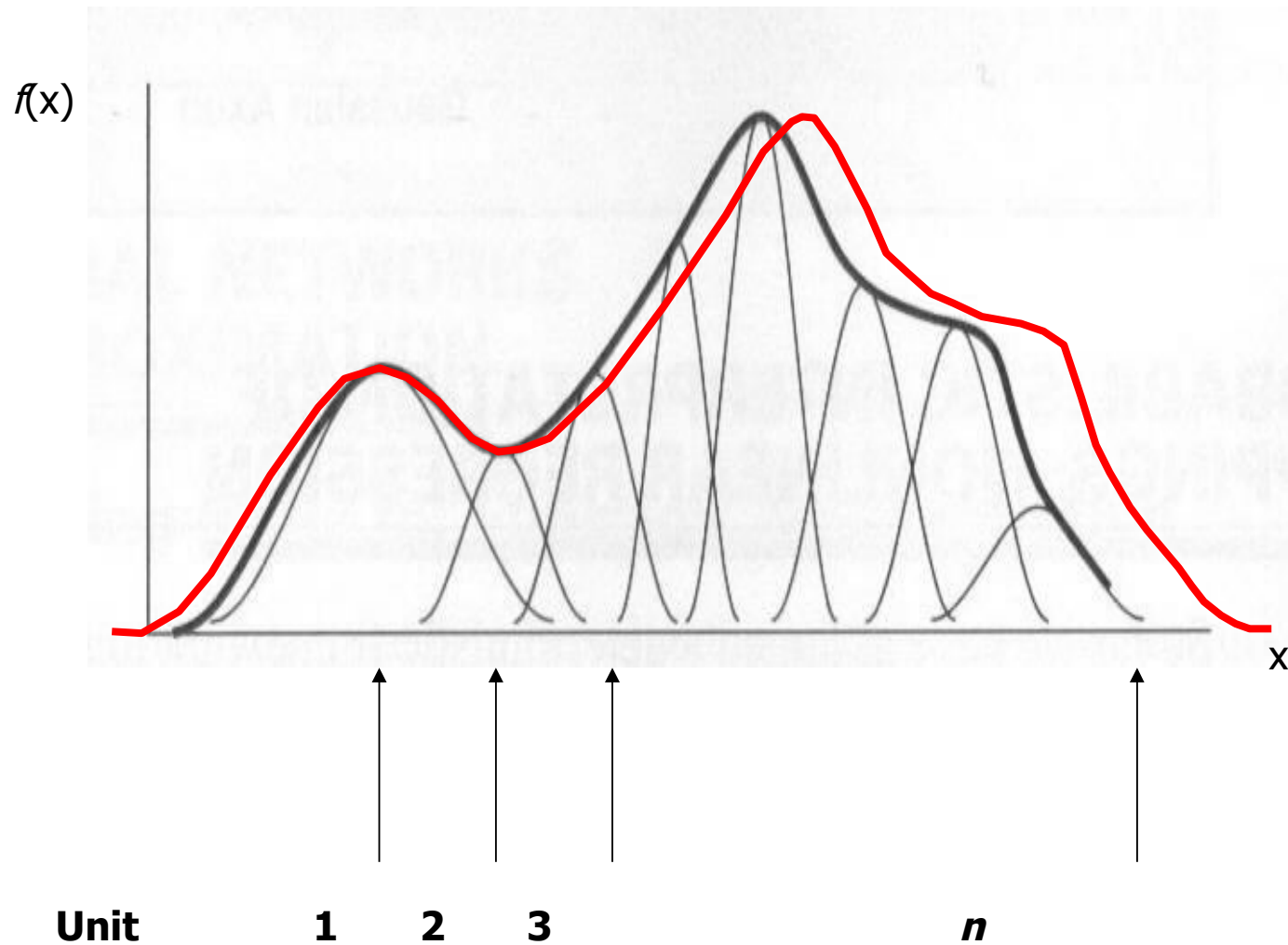
Approximation by RBF: Example



RBF neurons



Approximation by RBF: Example contd.



GMDH

- **Group Method of Data Handling.**
- By A.G. Ivakhnenko, 1968.
- Network is constructed by **induction**.
- Multiple layers.
- Feed-forward.
- Supervised.
- See <http://www.gmdh.net> ...



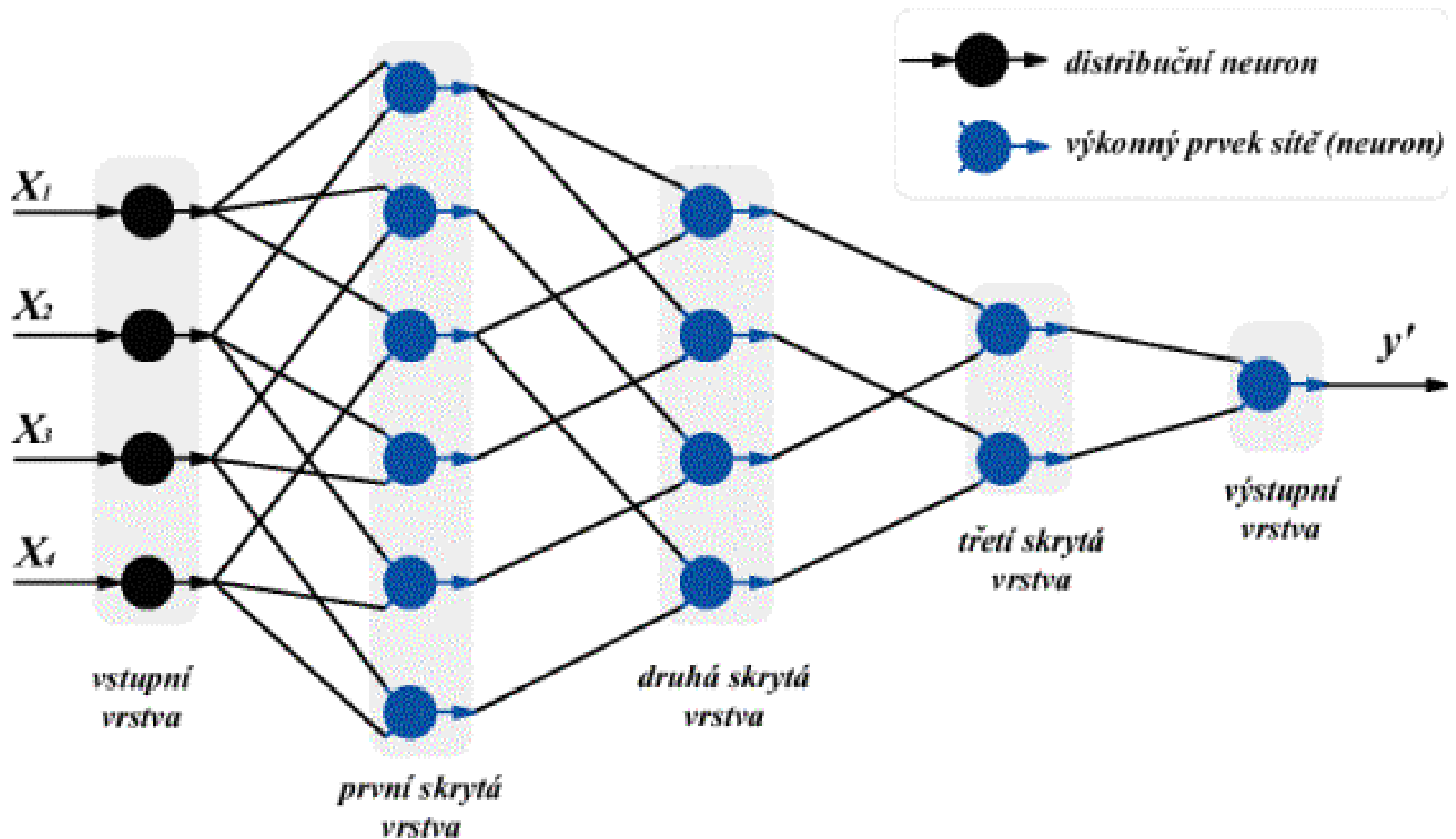
Induction vs. Deduction

- There are two approaches to create a model of a system:
 - **data-driven**: based on observations of a real system. The task is to infer general rule from specific cases → this the way GMDH model is built → **induction**,
 - **theory-driven**: based on theoretical knowledge of system functionality → the way to create mathematical models → **deduction**.

GMDH Types

- Parametric
 - parameters are set during training,
 - MIA (Multilayer Iterative Algorithm) → partial induction
 - COMBI (Combinatorial Algorithm) → full induction
- Other variants are non-parametric.

GMDH MIA Architecture

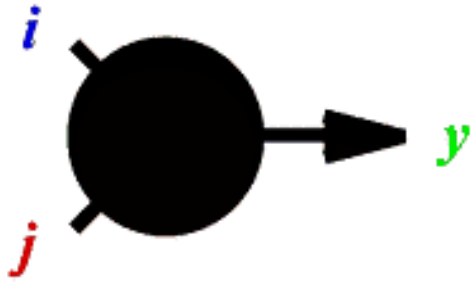


Remarks

- The structure (topology) of this network forms itself during learning process.
- # of neurons and even # of hidden layers changes during learning,
→ network self-organizes (but in a different way than SOM)

GMDH MIA Neuron

$$y = ai^2 + bij + cj^2 + di + ej + f$$



**Ivakhnenko's
polynomial.**

Example of other Ivakhnenko's polynomials:

$$y = a + bi + cj ,$$

$$y = ai + bjk .$$

MIA Characteristics

- MIA network uses supervised learning.
- MIA uses single type of neurons (unlike other GMDH variants).
- Training:
 - network is built layer by layer,
 - layers are added until some error criterion is met.
- Network evaluation is simple:
 - feed-forward, only single output.

MIA Training

- **Configure k-th layer** (k denotes actual layer)
 - create new neurons in the layer,
 - compute all six coefficients of polynomial.
- **Selection of neurons in k-th layer**
 - elimination of unsuccessful neurons.
- Repeat or **finish training**.

MIA Training II

input
variables

output
variable

v_1 ○

v_2 ○

v_3 ○

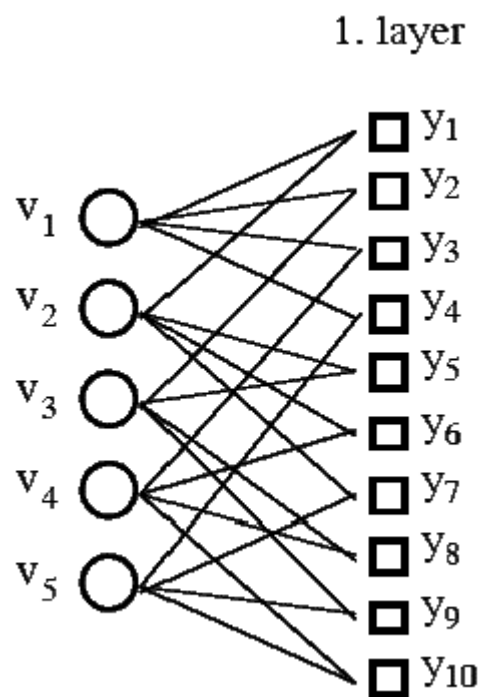
v_4 ○

v_5 ○

○ y

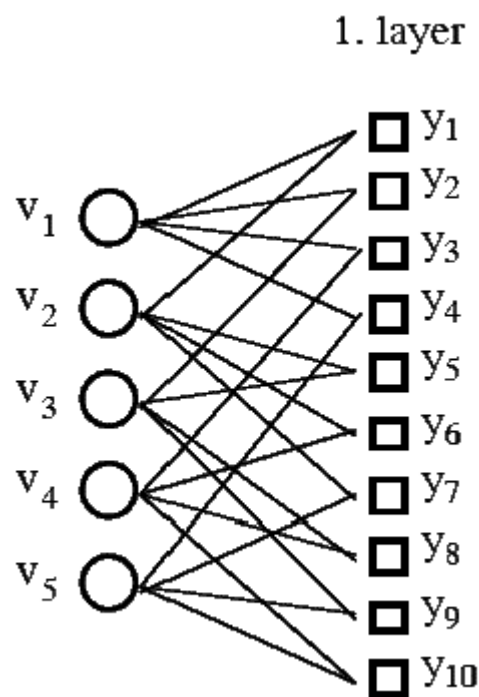
Lemke,Müller:SELF-ORGANIZING DATA MINING BASED ON GMDH PRINCIPLE

MIA Training III



- 1st layer creation:
- Active neurons with **optimized** transfer function $y_k = f_k(v_i, v_j)$.
- Two inputs only!
- They form “initial population”.
- How many neurons?

MIA Training III



- 1st layer creation:
- Active neurons with **optimized** transfer function $y_k = f_k(v_i, v_j)$.
- Two inputs only!
- They form “initial population”.
- How many neurons?

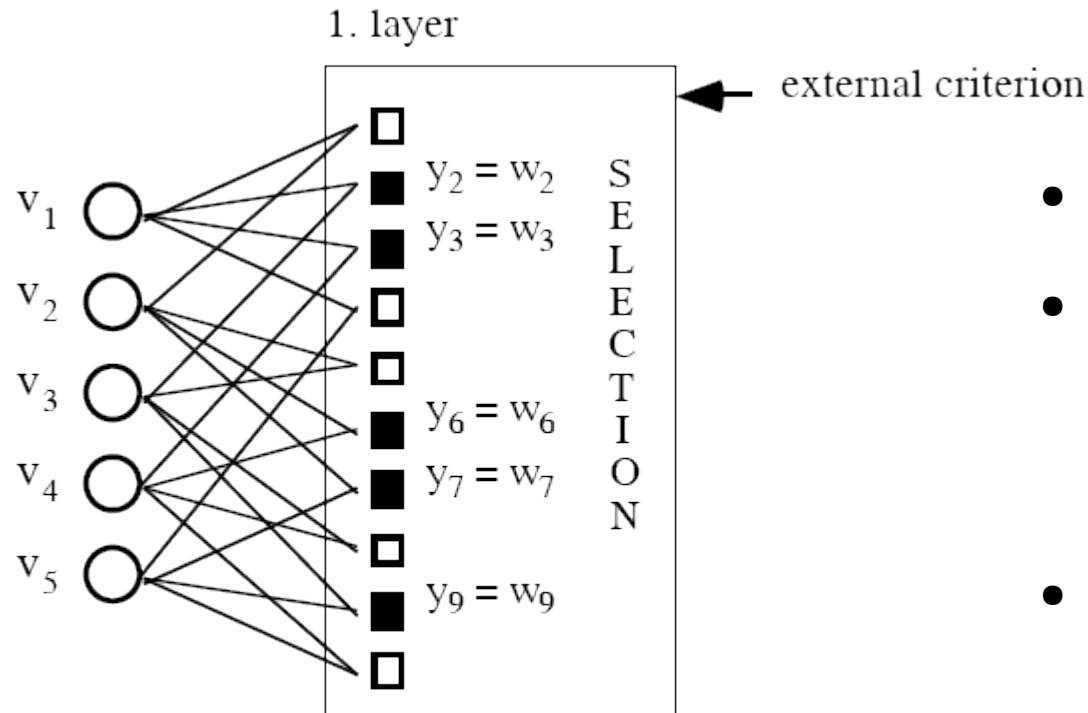
$$\binom{N}{2} = \frac{N(N-1)}{2}$$

MIA Training IV

- How to optimize neuron's coefficients?
 - Least Mean Squares (LMS),
 - we have to optimize six coefficients
 - choose 6 random input vectors
 - solve system of 6 linear equations

$$y = ai^2 + bij + cj^2 + di + ej + f$$

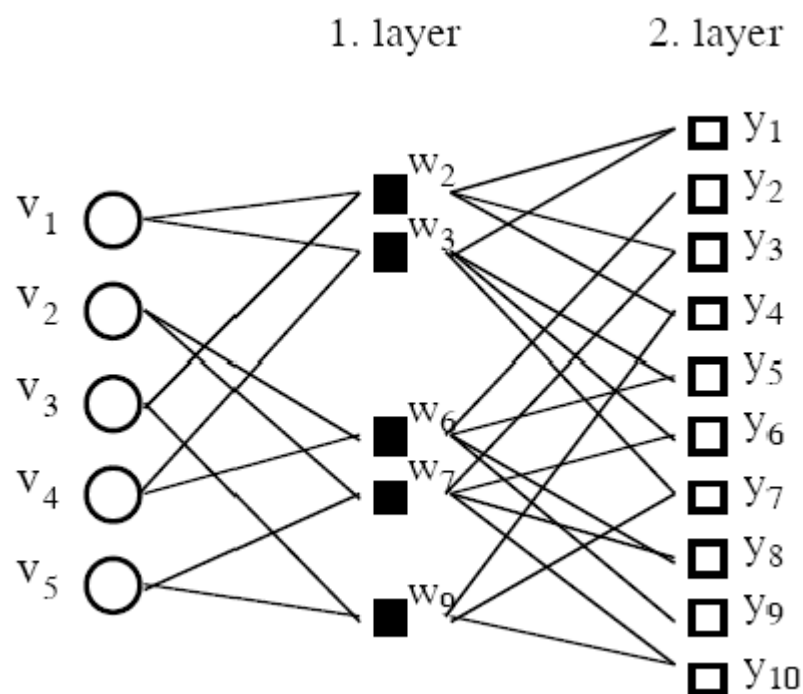
MIA Training V



- selected neuron (survives)
- not selected neuron (dies)

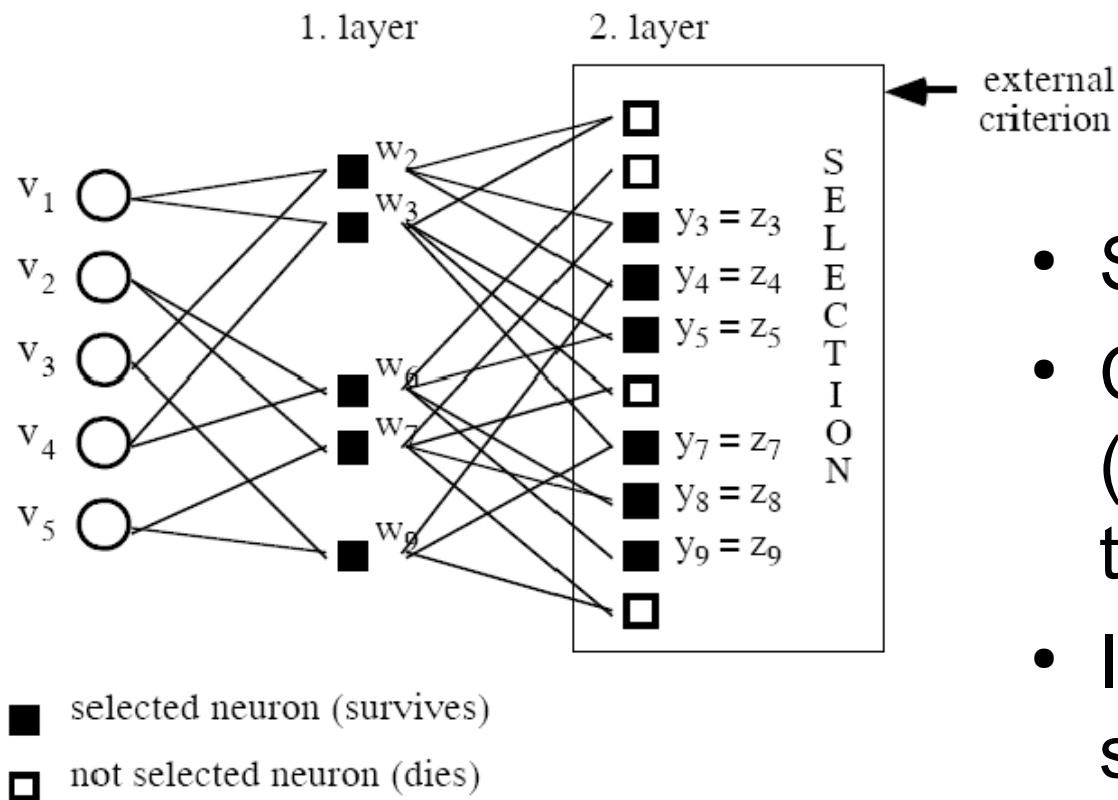
- Next step: Selection.
- According to a criterion (see later) some neurons survive, others not...
- If we are satisfied with the output (low error) → stop.
- Similar to evolutionary algorithms...

MIA Training VI



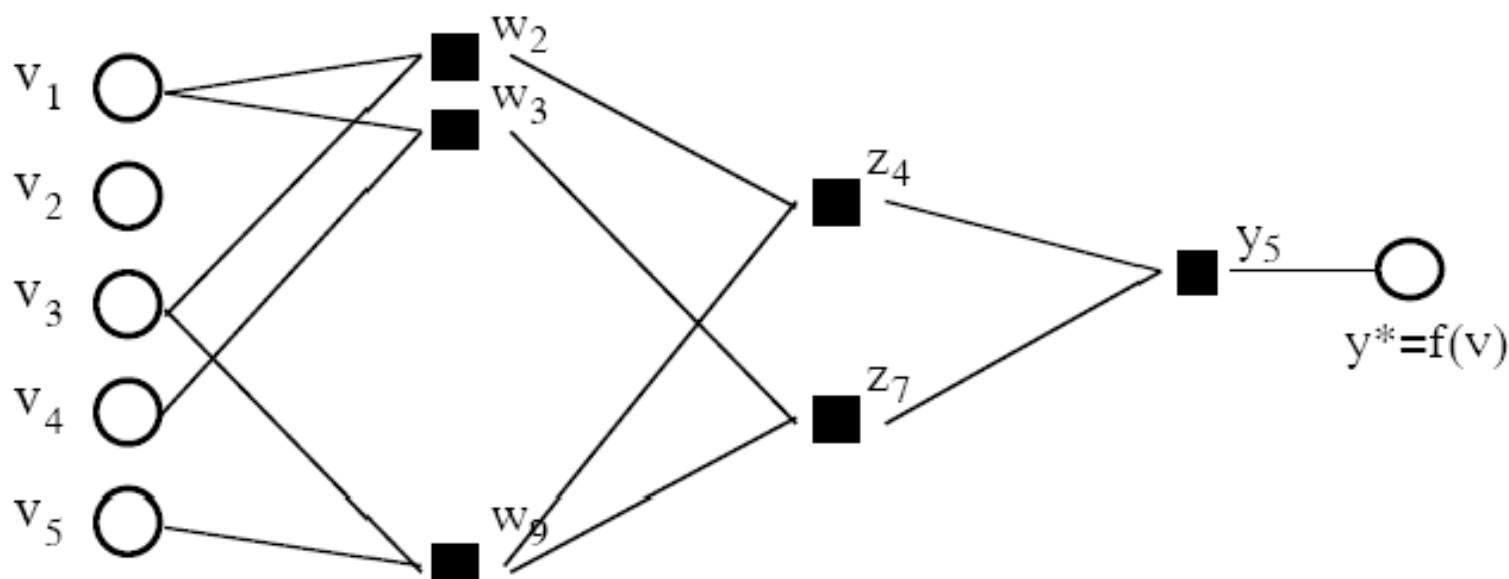
- 2nd layer:
- Again, initial population has to be created.
- Active neurons form the possible network output.
- Next step?

MIA Training VII



- Selection again.
- Competing neurons (survive and die) are of the same complexity.
- If we are still not satisfied with them → next layer is created.

MIA Training VII



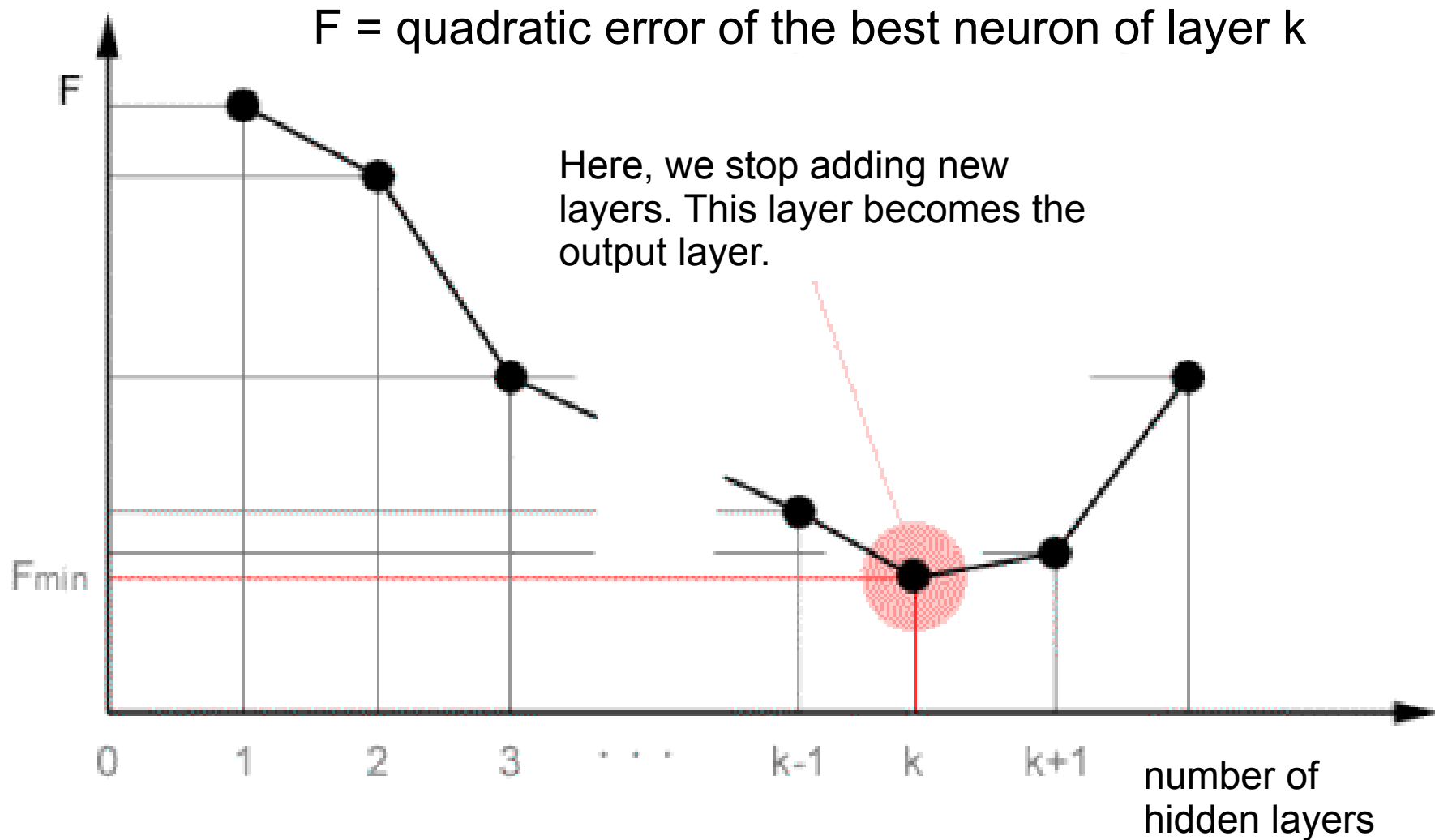
explicit analytically available optimal complex model.

network status:

order of regression model y^* : ≤ 8

number of variables v in model y^* : 4 (from initially 5 variables)

When to End Training?



Possibilities to Determine Fmin :)

$\frac{N}{N-p} s_e^2$	$s_{e,p}^2$	$s_e^2 + s_y^2 \ln N$	$\frac{P}{N}$	Schwarz	<i>Regularity criterion</i>	$AB = \frac{1}{N_B} \sum_{i=1}^{N_B} [y_i - \hat{y}_i(A)]^2 \rightarrow \min.$
$\frac{N+p}{N-p} s_e^2$	FPE	$N \ln s_e^2 + p \ln N$		Rissanen	<i>Discrimination criterion</i>	$\sum_{i=1}^N (y_i - \hat{y}_i)^2$
$s_e^2 + 2\sigma_p^2 \frac{P}{N}$	PSE	$\frac{2ps_e^2}{N-p} - Ns_{y^m}^2$		Kocerka		$\delta^2 = \frac{\sum_{i=1}^N (y_i - \hat{y}_i)^2}{\sum_{i=1}^N (y_i - \bar{y})^2} < 1.0$
$\frac{Ns_e^2}{\sigma_c^2} + 2p - N$	Mallow C_p	$\frac{\sum_{t=1}^N (\bar{y}_t^m - y_t)^2}{N}$		$i^2(N)$	<i>Validation criterion</i>	$R_{ij} = 1 - \sqrt{\frac{\frac{1}{N-P_i} \sum_{t=1}^N (y_t - y_t^i)^2}{\frac{1}{N-P_j} \sum_{t=1}^N (y_t - y_t^j)^2}}$
$N \ln s_e^2 + 2p + c$	AIC	$\sum_{t=1}^N y_t^2$				
$N \ln s_{e,p}^2 + p \ln \left(\frac{s_{y^m}^2}{s_{e,p}^2} \frac{N}{p} \right)$	BIC	s_e^2		$\left[I - \lambda \sqrt{\frac{k(\ln(\frac{N}{k}) + I) - \ln \alpha}{N}} \left(I - \frac{I}{4} \frac{\ln(\frac{N}{k}) + I - \ln \alpha}{N} \right) \right]$		Vapnik
$s_e^2 = \frac{1}{N} \sum_{t=1}^N (y_t - y_t^m)^2 = \frac{1}{N} \sum_{t=1}^N e_t^2; \quad s_y^2 = \frac{1}{N} \sum_{t=1}^N (y_t - \bar{y})^2; \quad s_{y^m}^2 = \frac{1}{N} \sum_{t=1}^N (y_t^m)^2$						

Most Common

Regularity Criterion:

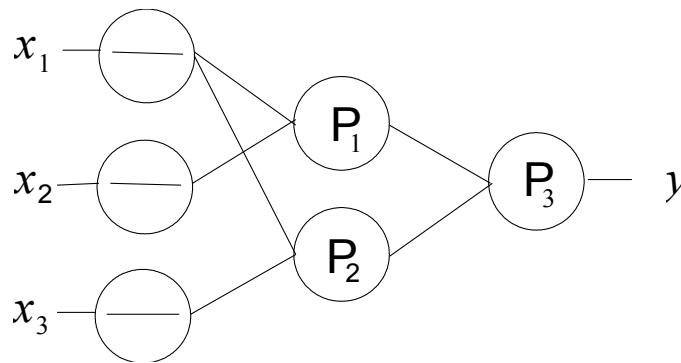
$$AB = \frac{1}{N_B} \sum_{i=1}^{N_B} [y_i - \hat{y}_i(A)]^2 \rightarrow \min.$$

$\hat{y}_i(A)$... neuron's output trained on the A-subset.

Neurons are then ordered according to their error on validation data B-subset.

GMDH → Mathematical Model

- Example:



P_i are linear

$$y = P_3(P_2(x_1, x_3), P_1(x_1, x_2)) = a_{31}(a_{11}x_1 + a_{12}x_3 + a_{13}) + a_{32}(a_{21}x_1 + a_{22}x_2 + a_{23}) + a_{33} =$$
$$(a_{31}a_{11} + a_{32}a_{21})x_1 + (a_{32}a_{22})x_2 + (a_{31}a_{12})x_3 + a_{31}a_{13} + a_{32}a_{23} + a_{33}$$

GMDH → Mathematical Model II

- Problems:
 - real systems are too complex and...
 - equations become too confusing.
- Solution?
 - Visualize model behaviour!

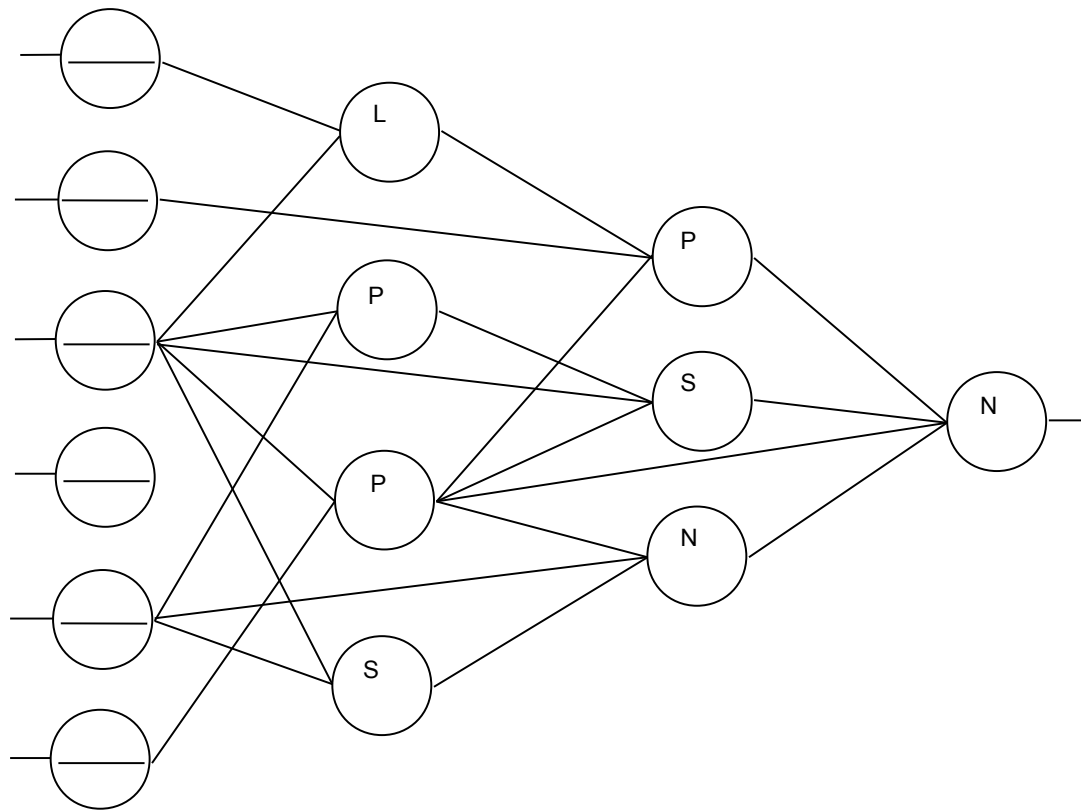
COMBI vs. MIA

- What's the difference between partial and full induction?
- COMBI uses supervised learning, too.
- COMBI employs different types of neurons, but:
 - same in a single layer,
 - “more complex” in following layers,
 - not connected to a previous layer, but to network inputs
 - always combine “the most fit”.

Notes

- The full induction network (COMBI) works with a full input space all the time.
- It tries to apply successively complex transformations.
- The partial induction network (MIA) applies only a single transformation to most promising input subspaces.
 - It is a heuristic which speeds up the algorithm.

ModGMDH



Different neuron types:

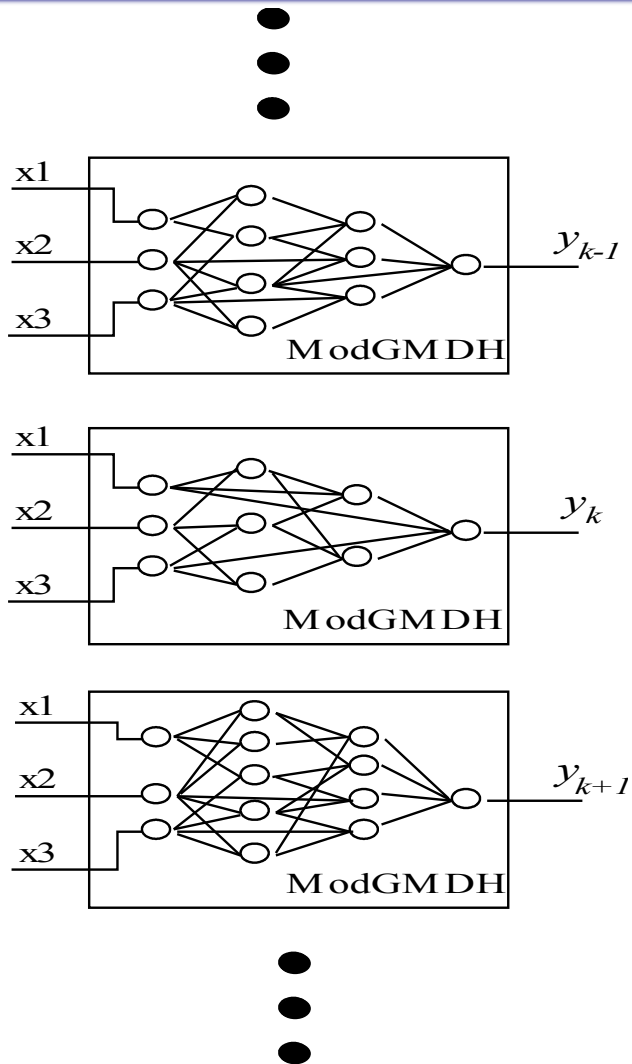
L - linear,

P - polynomial,

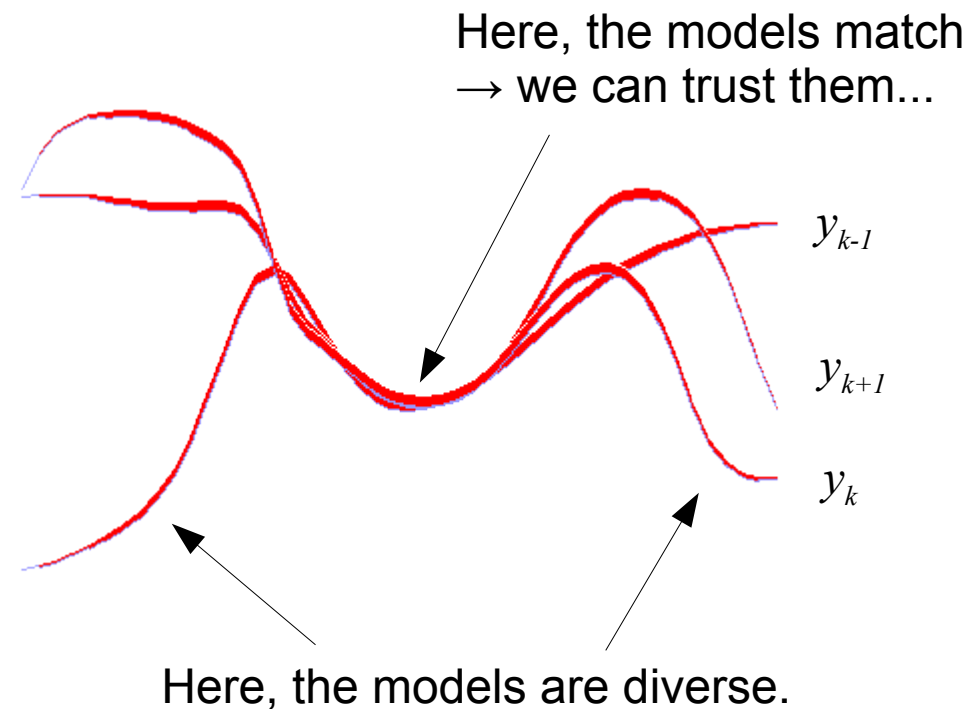
S - sigmoid,

N - perceptron

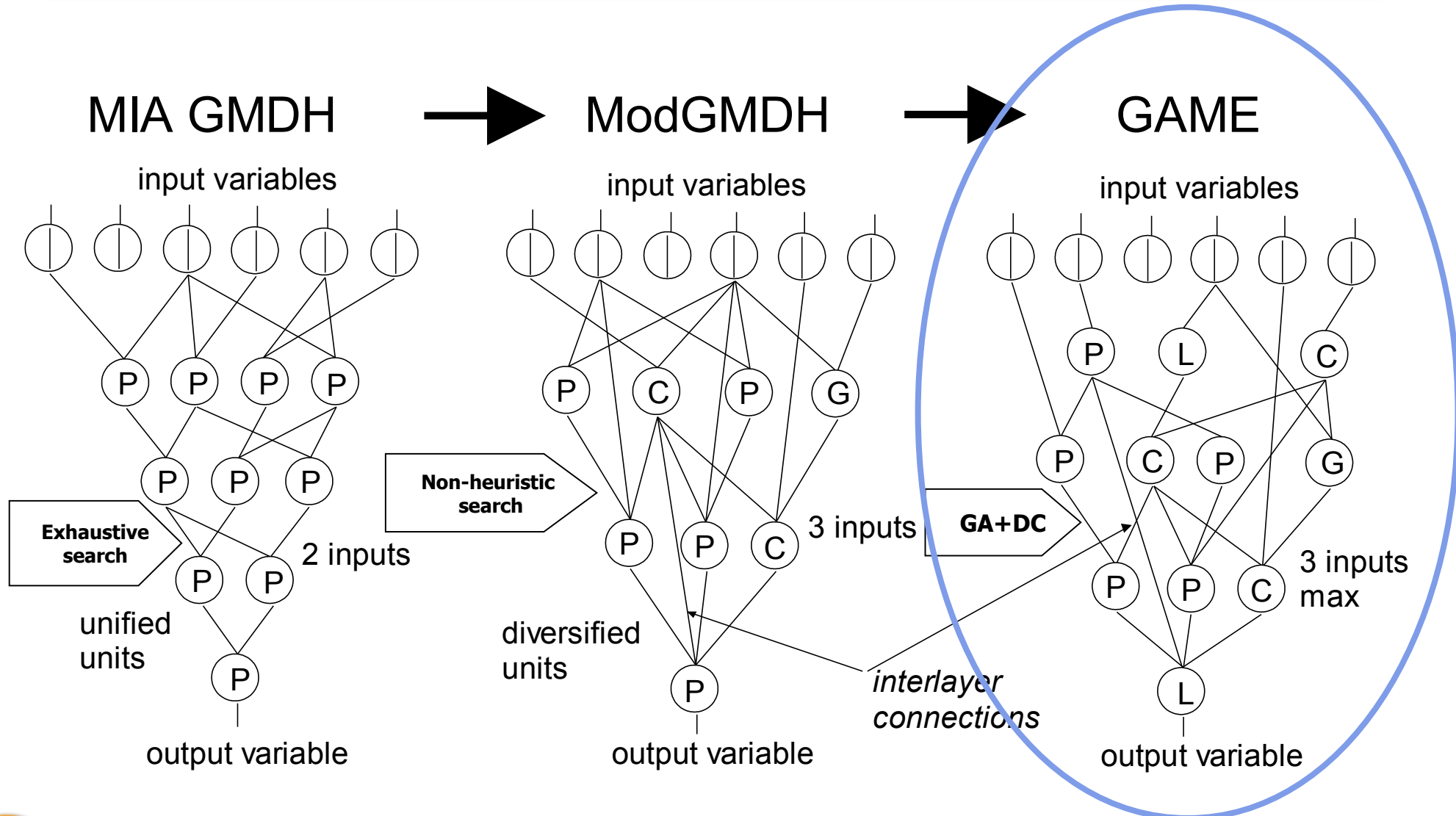
Creating Diverse Models



We can produce multiple models instead of a one.

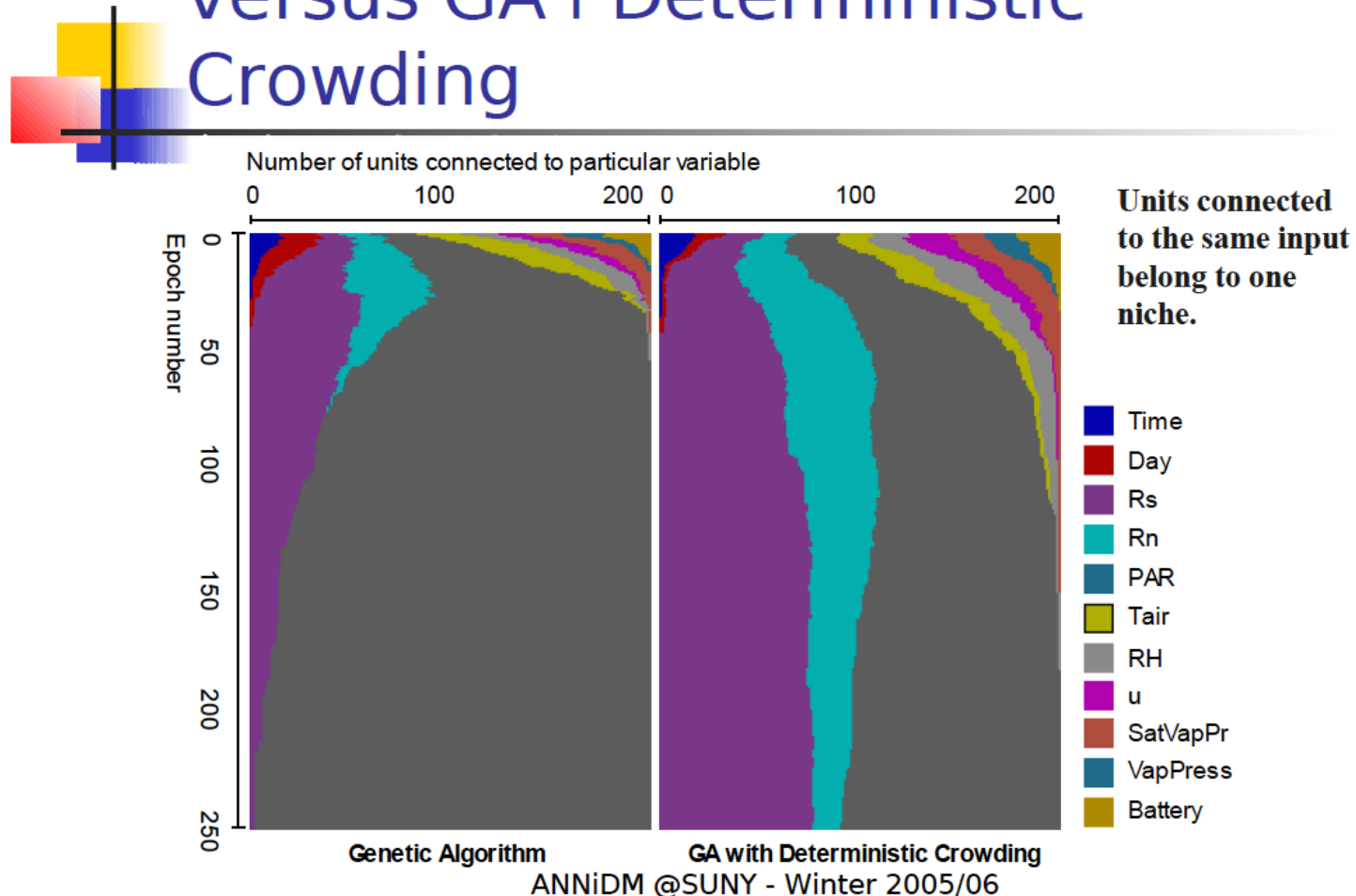


GAME (P. Kordik - CTU)



Why to Use Niching GA?

Functionality test: Standard GA versus GA+Deterministic Crowding



Other Inductive Approaches

- Cascade Correlation:
diploma thesis by Minh Duc Do:
<https://dip.felk.cvut.cz/browse/details.php?f=F3&d=K13136&y=2009&a=dom1&t=dip>
- NEAT and other TWEANNs (Topology and Weight Evolving Neural Networks)
→ we will see later...

Next Lecture

- Time series processing.
- Recurrent neural networks.