Learned from Neural Networks

R.P.W. Duin

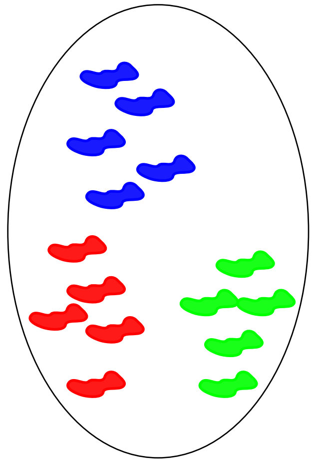
Pattern Recognition Group
Delft University of Technology
The Netherlands

duin@tn.tudelft.nl

Lommel, June 2000

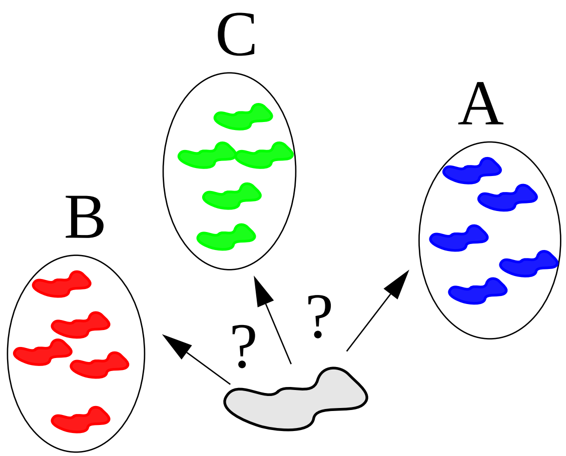
The Pattern Recognition Problem

Unsupervised Problem
(Clustering)



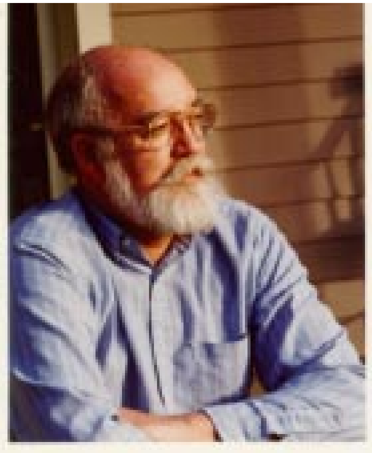
Define the classes
for a set of real world objects

Supervised Problem
(Classification)



Find the class of a
new, real world object
given a number of examples

Hard AI: The Brain is a Computer

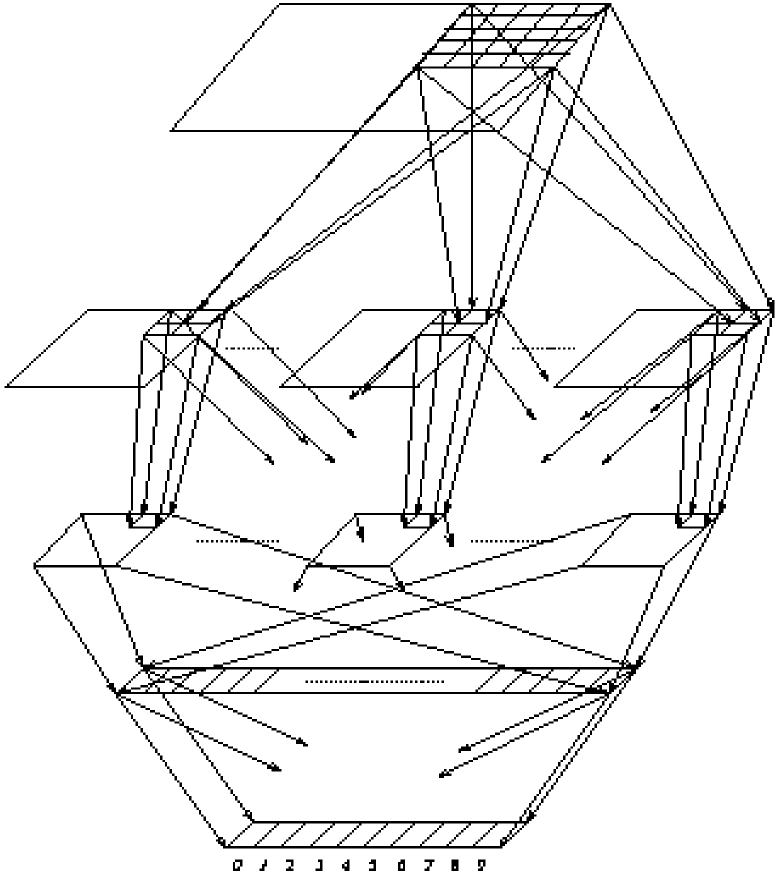


Dennett: There is no scientific need to take into account the concept of consciousness. Its existance can be denyied.

Where to Attack the Pattern Recognition Problem?

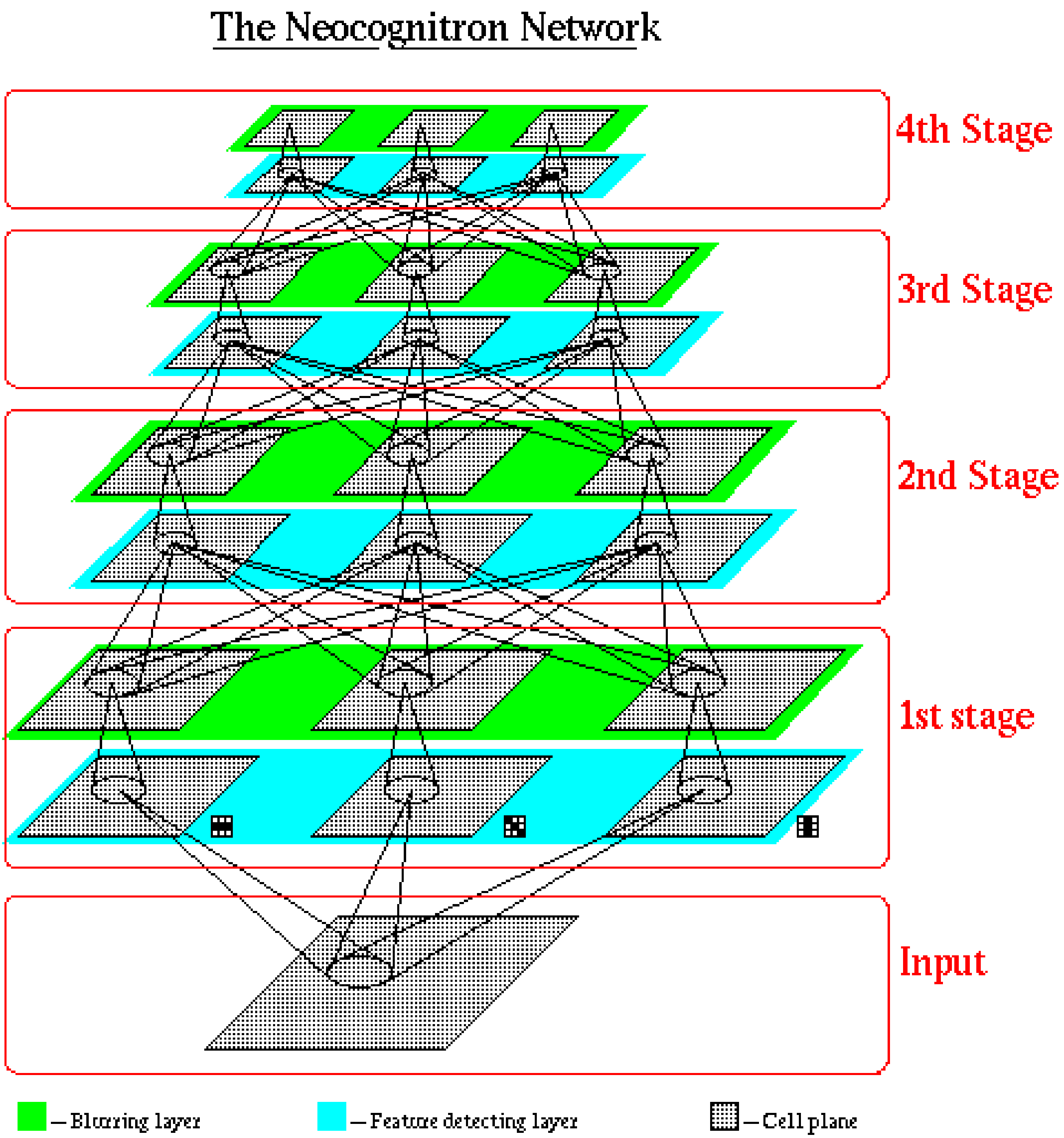
	<u>Matter</u>	<u>Mind</u>	
			Plato
			Descartes
Searle	Model the Brain	Generalize the Rules	Grenander
Penrose			Pavlidis
Dennett	*Neuro-Biology*	*Artificial Intelligence*	Goldfarb
LeCun	*Perception*	*Statistical Pattern Recognition*	Kanal
Grossberg			Breiman
Koenderink	Model the Senses	Generalize from Examples (Given the Sensors)	Bacon
			Aristotle

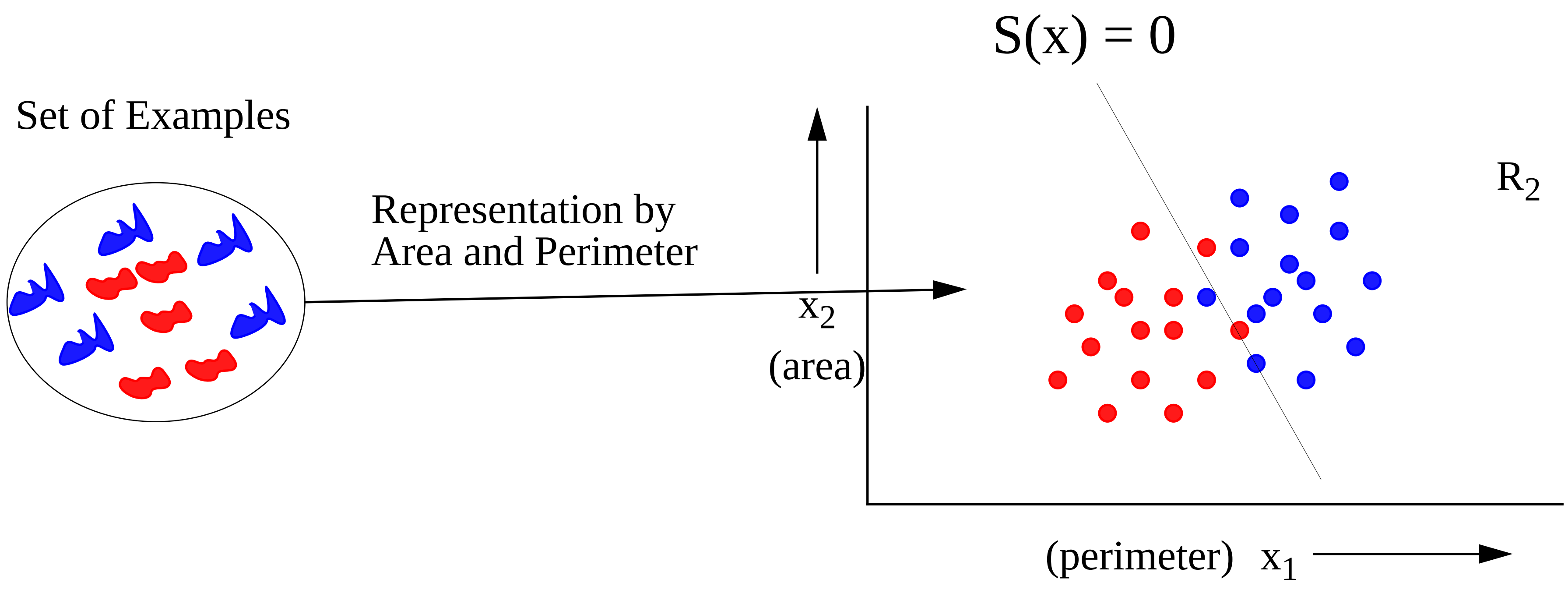
Shared Weight Network



		# Cells	# Connections	# Weights
L1	Input layer			
	16 x 16	256		
	12 x (8 x 8) x (5 x 5 + 1)		19968	
	12 x (8 x 8) x (1) + 12 x (5 x 5)			1068
L2	Feature maps			
	12 x (8 x 8)	768		
	12 x (4 x 4) x (8 x (5 x 5) + 1)		38592	
	12 x (4 x 4) x (1) + 12 x (8 x (5 x 5))			2592
L3	Subsampling maps			
	12 x (4 x 4)	192		
	30 x (12 x (4 x 4) + 1)		5790	5790
L4	Hidden layer			
	30	30		
	10 x (30 + 1)		310	310
L5	Output layer			
	10	10		
	Total	1256	64660	9760

Figure 5.2: An example 2D shared weights ANN.


The Neocognitron Network
4th Stage
3rd Stage
2nd Stage
1st stage
Input
– Blurring layer
– Feature detecting layer
– Cell plane



Fisher: $S(x) = (\hat{\mu}_A - \hat{\mu}_B)^T \hat{G}^{-1} x + \text{constant}$

Problems with the Traditional PR Approach

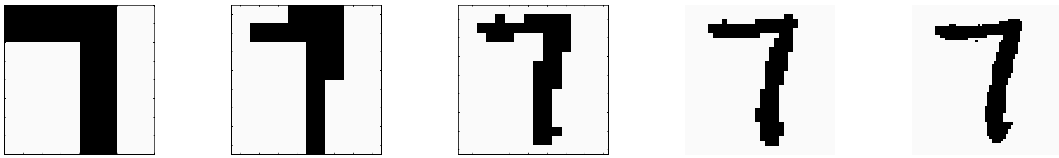
$\mathbf{x} = (x^1, x^2, \ldots, x^k)$ - k dimensional feature space

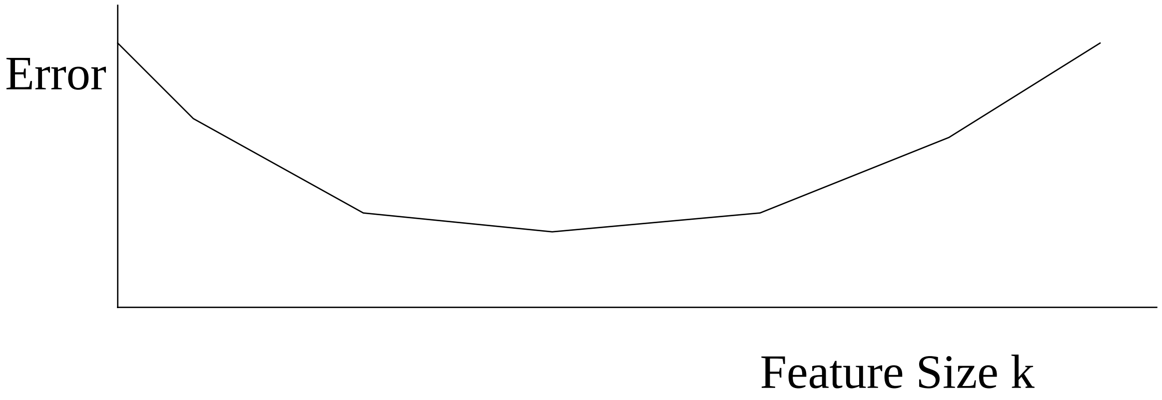
$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ - training set

$\{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ - class labels

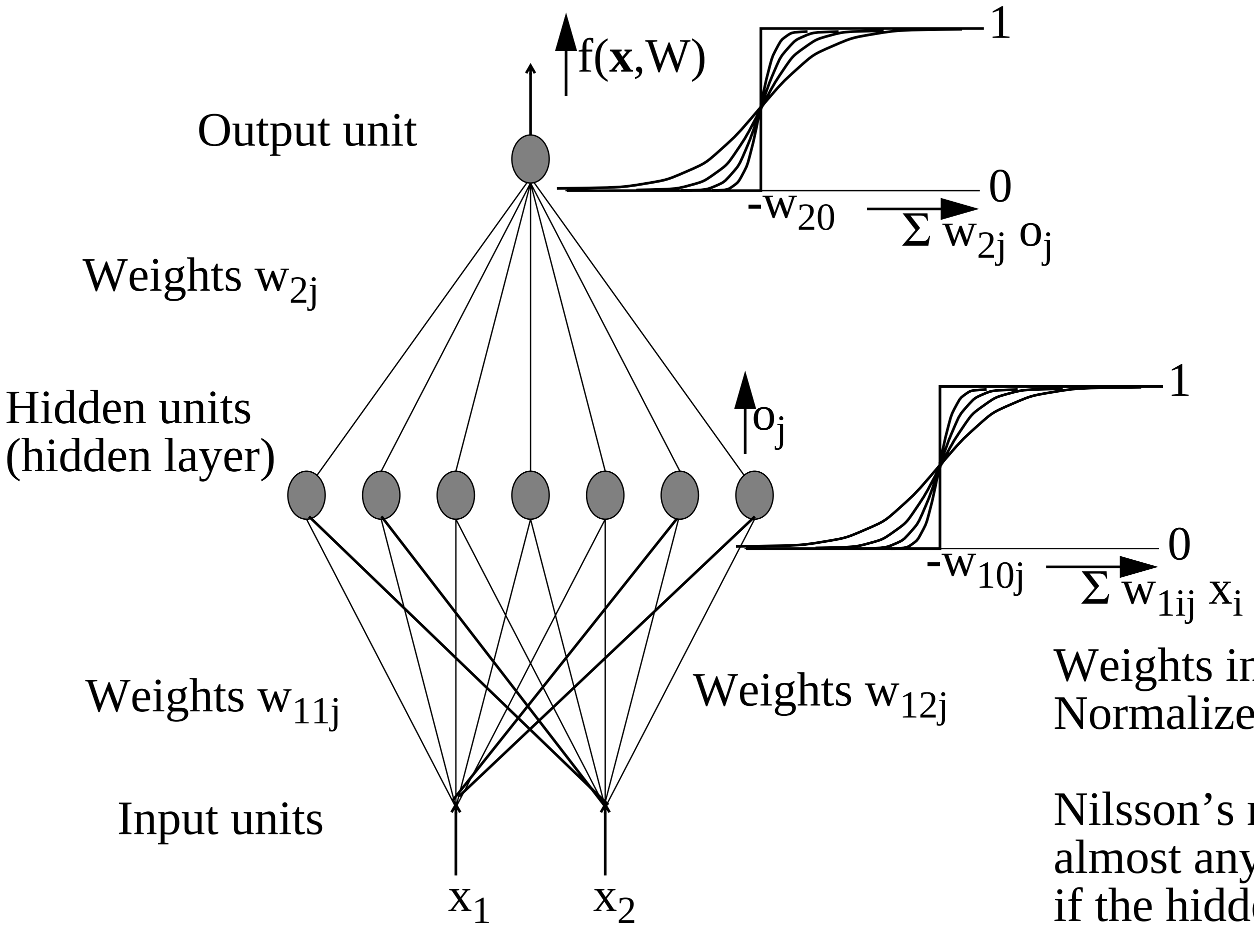
$\Big\}$ $D(\mathbf{x})$ - classifier , $\varepsilon = \text{Prob}\ (\ D(\mathbf{x}) \neq \lambda(\mathbf{x})\ )$

$\varepsilon(m)$: monotonically decreasing , $\varepsilon(k)$: peaks !





Neural Networks



Weights in all layers
Normalized outputs (0,1) in all layers

Nilsson's machine is included, so
almost any set of objects can be separated
if the hidden layer is large enough.

More output units are possible
More hidden layers are possible.

Backpropagation Training Rule

Network: $f(\mathbf{x}_p , W)$

Training set: $\{(\mathbf{x}_1 , t_1), (\mathbf{x}_2 , t_2), ... (\mathbf{x}_n , t_n)\}$

Target values (labels): t_p

Network error: $E = \sum_p \{ t_p - f(\mathbf{x}_p , W) \}^2$

Gradient descent: $W \leftarrow W + \Delta W = W + \sum_p \Delta_p W$

Generalized delta rule: $\Delta_p W$: $\Delta_p w_{ji} = \eta \delta_{pj} o_{pi}, \forall i,j$

Fixed stepsize: η

output units (layer k): $\delta_{pk} = (t_{pk} - o_{pk}) o_{pk} (1 - o_{pk})$

hidden units (layer j): $\delta_{pj} = o_{pj} (1 - o_{pj}) \sum_k \delta_{pk} w_{kj}$

The ’errors’ in the lower layers (j) are computed using the corrections of the layers above (k): backpropagation.

The Art of Training a Network

Using a neural network classifier is not straightforward all:

- Architecture (numbers of hidden layers and hidden units)
- input representation
- output representation
- target values
- reject values
- initialization procedure
- batches, partitioned learning set (size?) or individual training
- adding noise (amount?) to input or weights?
- step size η
- momentum term α

"The backpropagation training procedure can be a user's nightmare"

(Weiss & Kulikowski)

Large Network Examples

application	#weights	#samples	error	ref.
text -> speech	25000	5000	0.20	Sejnowski
sonar target rec	1105	192	0.15	Gorman
car control	>36000	1200	car drives on winding road	Pomerleau
back-gammon	>11000	3000	computer champion	Tesauro
sex rec from faces	>36000	90	0.09	Golomb
char rec	9900	5000	0.055	Sato
remote sensing	1800	50	0.05-0.10	Kamata
signature verif.	480	280	0.05	Sabourin

T.J. Sejnowski and C.R. Rosenberg, *NETtalk: a parallel network that learns to read aloud*, The John Hopkins University Electrical Eng. and Comp. Science, 1986.

P. Gorman and T.J. Sejnowski, *Learned Classification of Sonar Targets Using Massively Parallel Network*, IEEE Transactions on ASSP, vol. 36, no. 7, July 1988.

D. Pomerleau, ALVINN: *Ann Autonomous Land Vehicle in a Neural Network*, in: David S. Touretzky, Advances in Neural Information Processing Systems I, 1989

G. Tesauro, *Neurogammon wins computer olympiad*, Neural Computation, vol. 1, pp 312-323, 1990

B.A. Golomb, D.T. Lawrence, T.J. Sejnowski, Sexnet: *A neural network identifies sex from human faces*, Adv. in Neural Inf. Proc. Sys. I, 1989

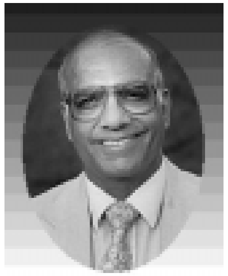
A. Sato, K. Yamada, J. Tsukumo, and T. Temma, *Neural network models for incremental learning*, ICNN, Helsinki, 1991.

S.-I. Kamata, R.O. Eason, A. Perez, and E. Kawaguchi, *A Neural Network Classifier for LANDSAT Image Data*, Proc. 11th ICPR, The Hague, Vol 2, 573-576, 1992

R. Sabourin and J-P. Drouhard, *Off-Line Signature Verification Using Directional PDF and Neural Networks*, Proc. 11th ICPR, The Hague, Vol 2, 321-325, 1992

Neural Network Appreciation in PR

"Artificial Intelligence and Neural Networks have deceived and spoiled two generations of computer scientists just by these names" (Rosenfeld, Oulu 1989)



"Neural Networks has brought new enthusiasm and spirit to the next generation of young researchers."
(Kanal, Jerusalem 1994)

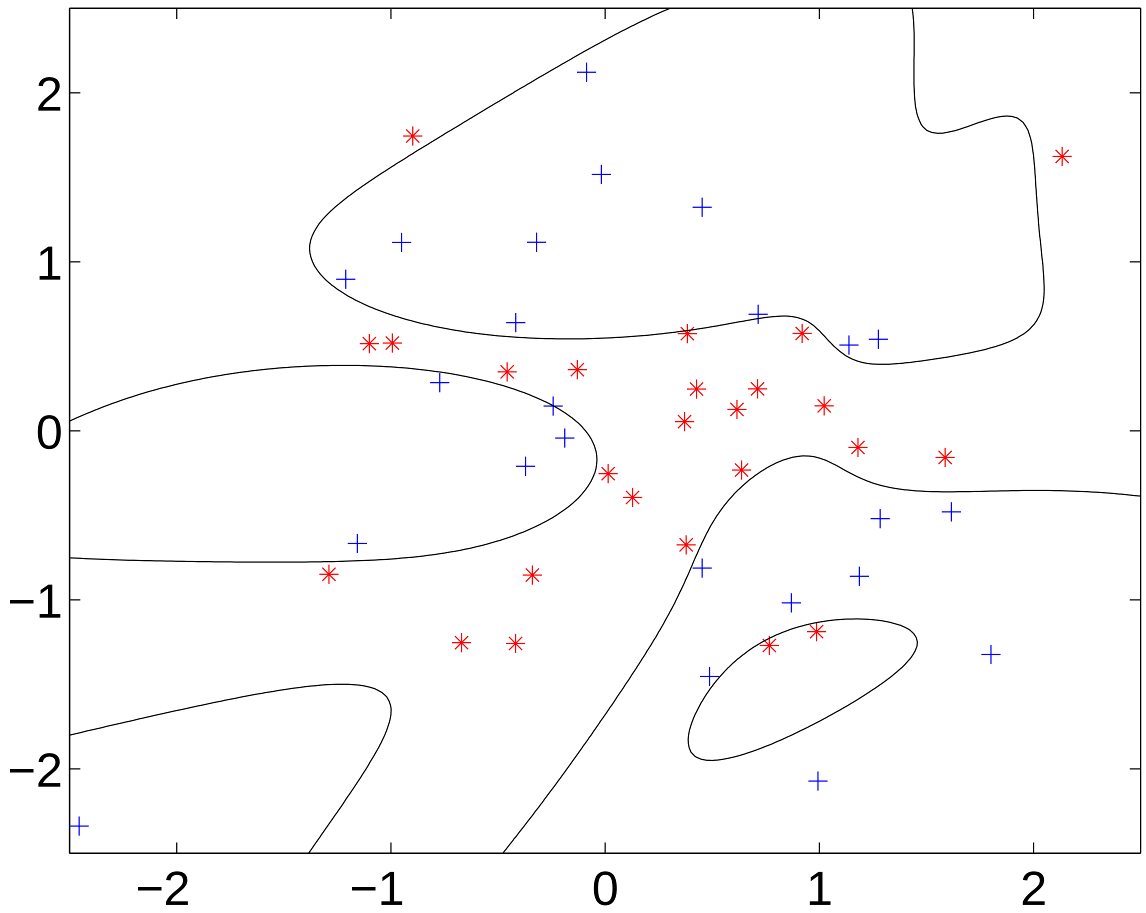
"Just a short look at the architecture of a Neural Network is sufficient to see that the thing simply doesn't have the moral right to show any reasonable performance"
(Breiman, Edinburgh, 1995)

My Neural Network Problem

"Your problem, Dr. Duin, is that you want to understand the neural network. You will have to accept that the interesting aspect of neural networks is that their behavior cannot be understood. " (NN, Delft, 1991)

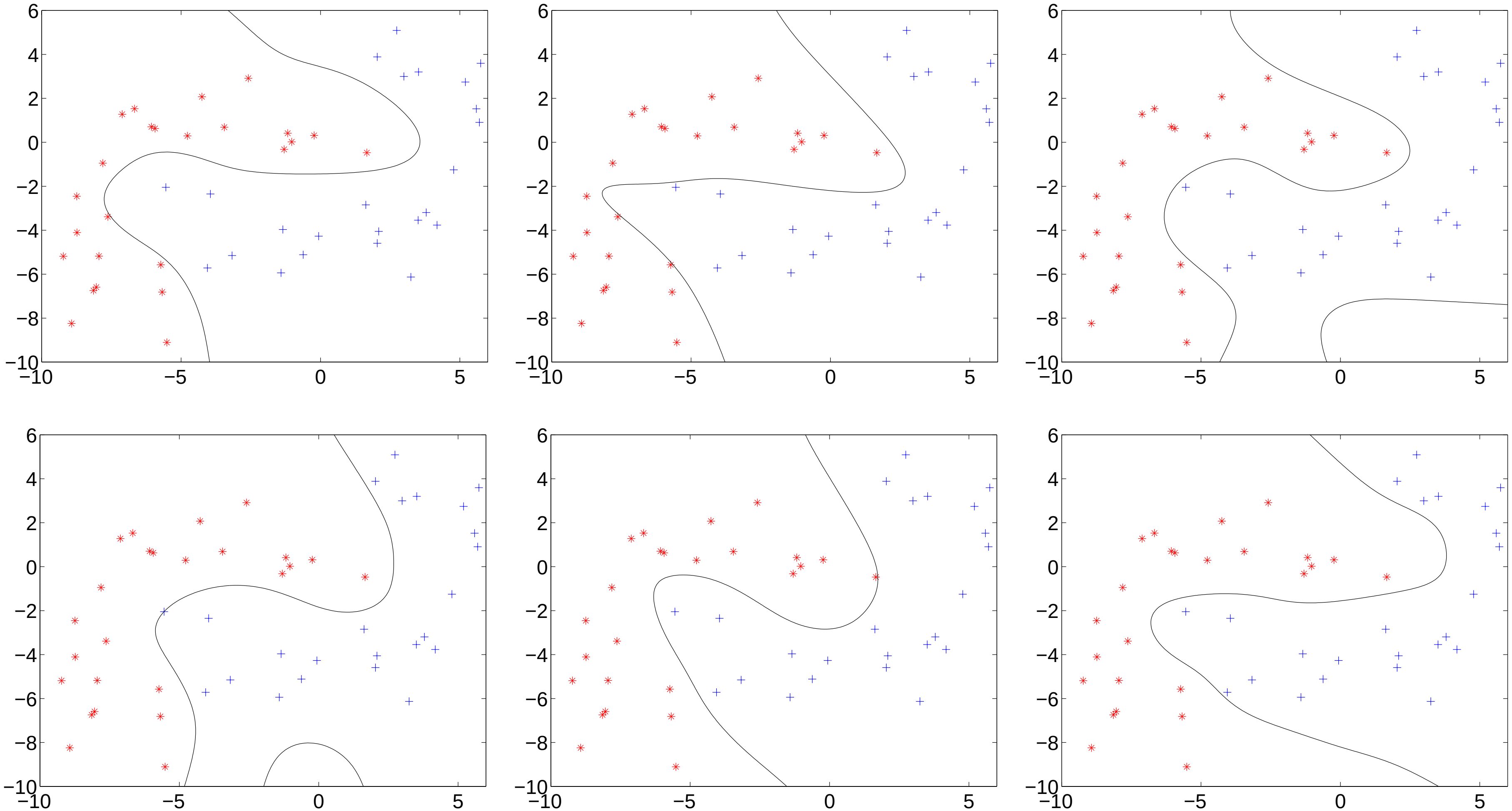
So, it is just magic!

Neural Network as Universal Approximator



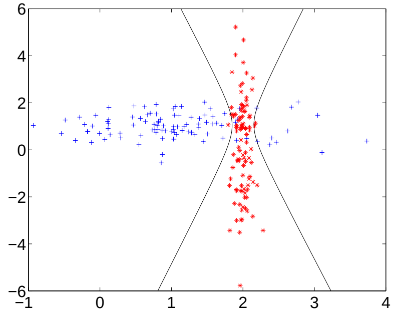
A sufficiently large neural network can solve almost any problem.

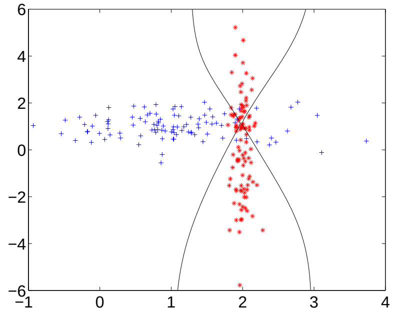
Neural Network as Universal Problem Solver

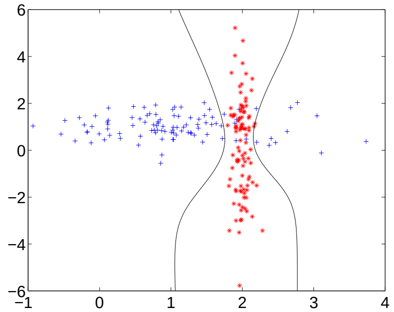


A neural network can solve a problem in many different ways

Training versus Implementing







Any function can be implemented on a neural network

Consequently, it can be trained by any rule.

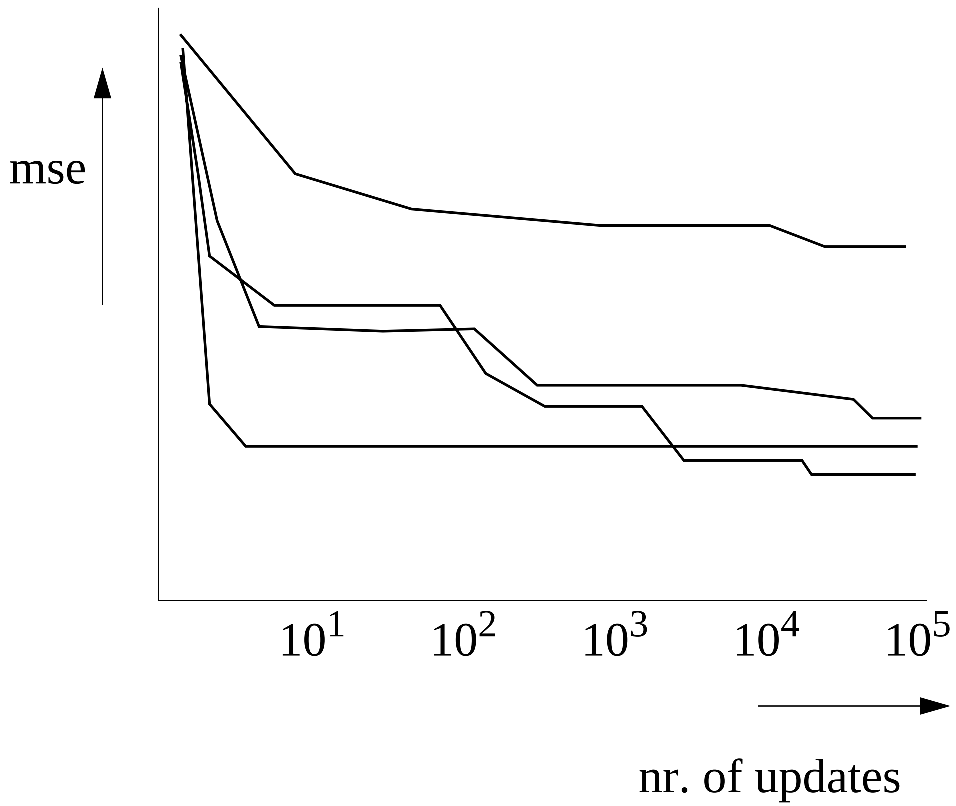
The architecture is general, and thereby not special.

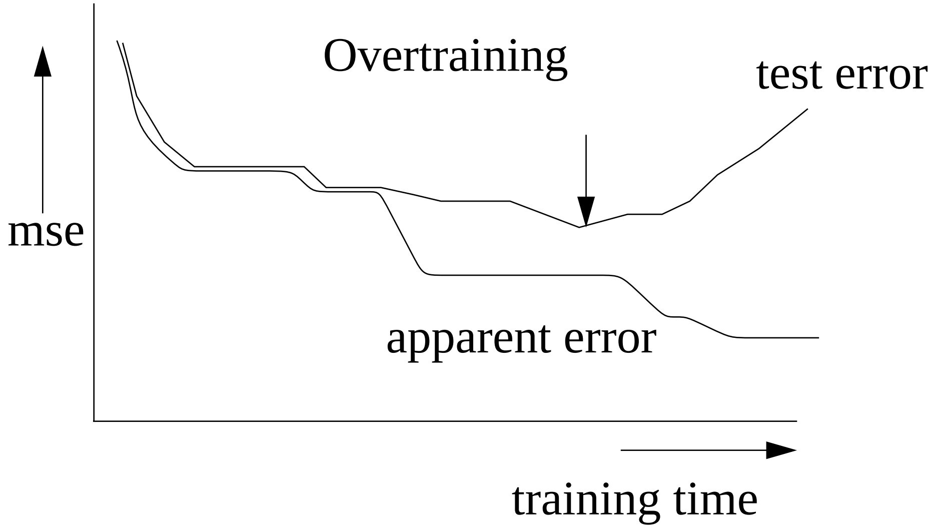
What is special, is the original training rule.

Gradient Descent Training Characteristics

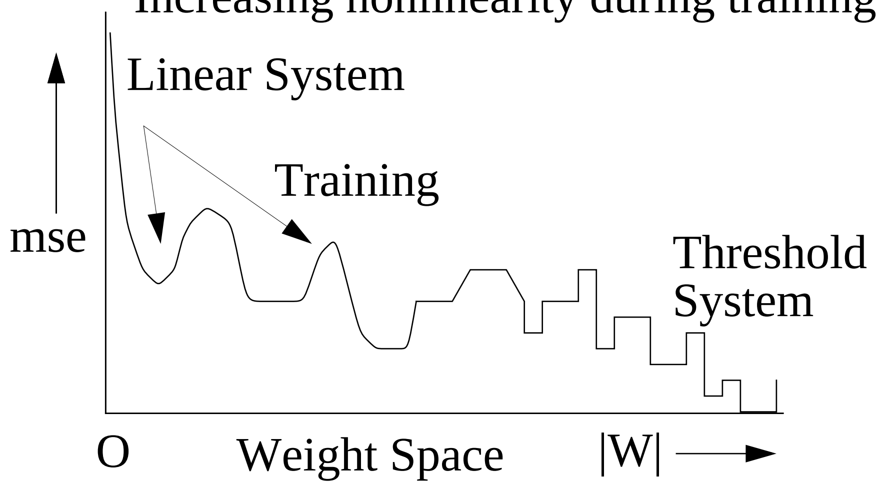
Not reproducible

Random initialization

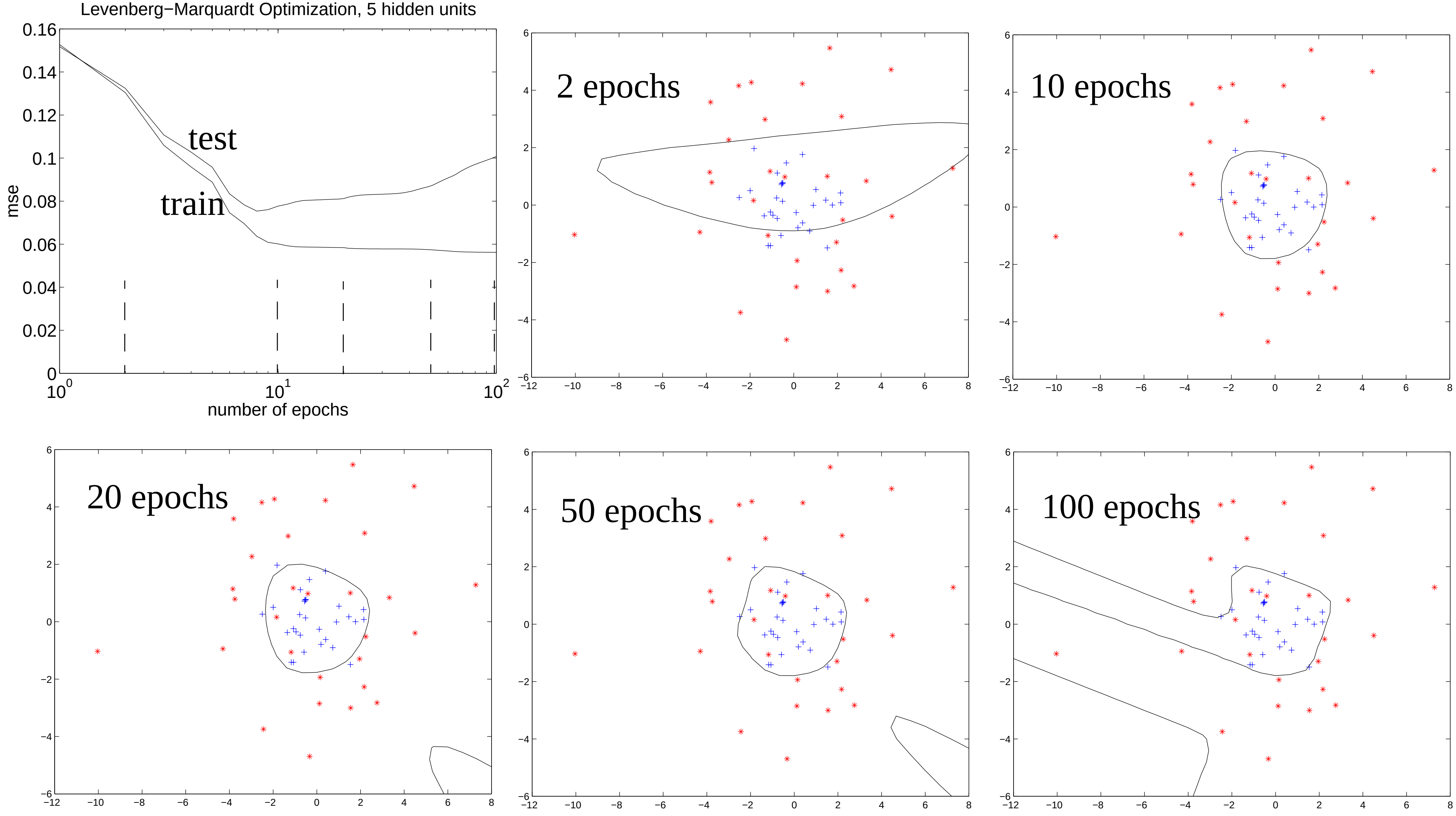




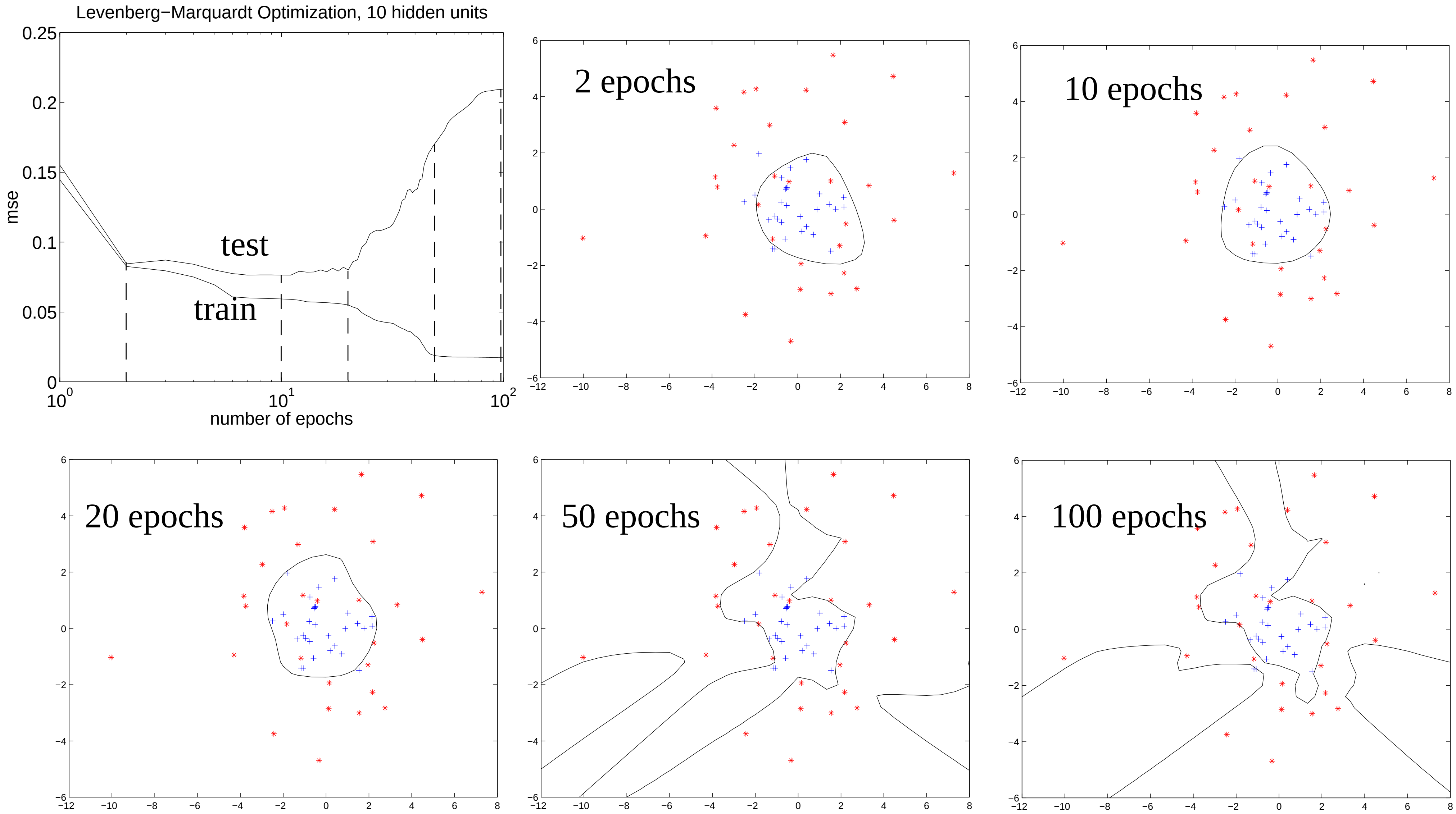
Increasing nonlinearity during training



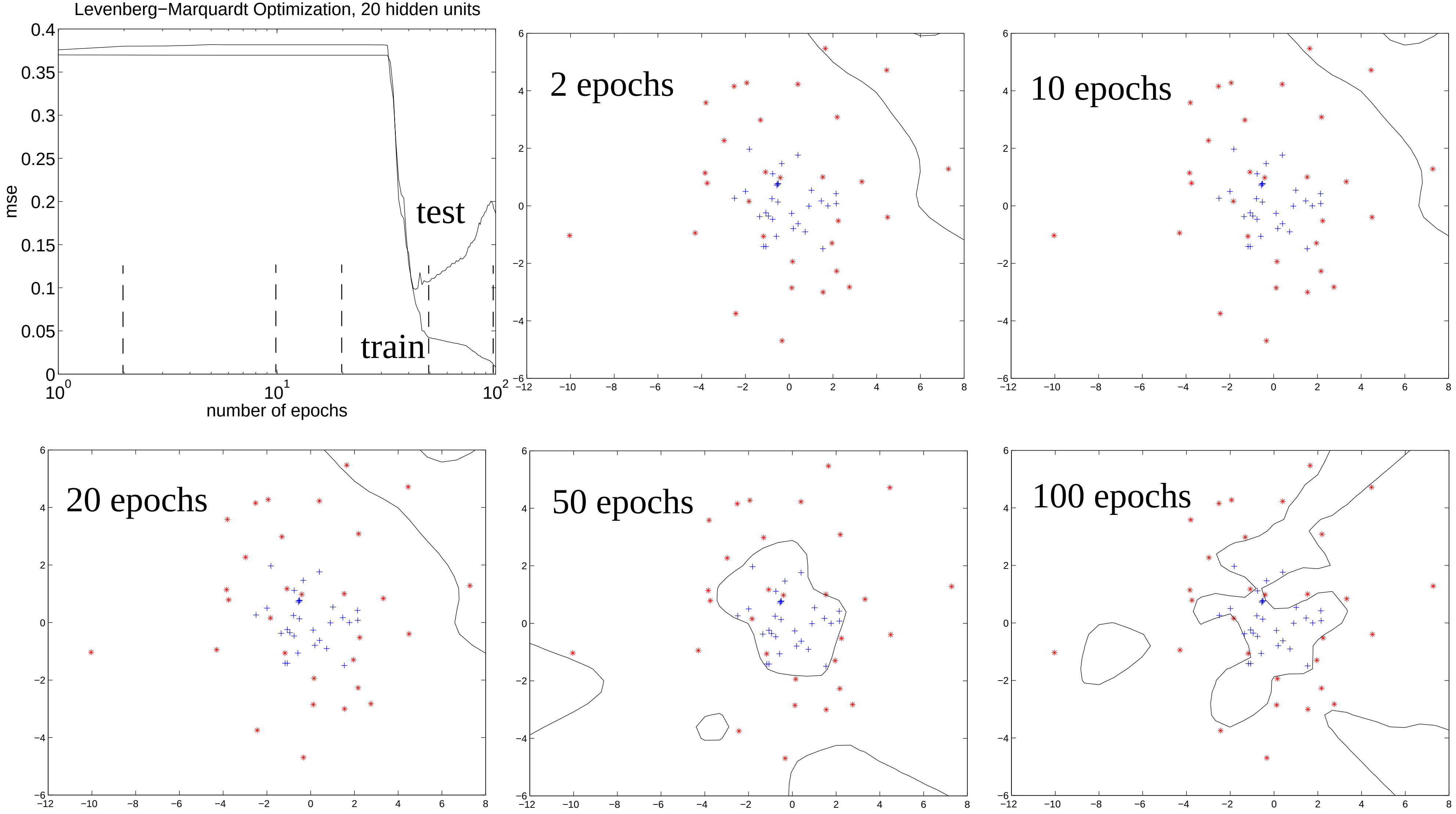
Overtraining Example - 5 Hidden Units



Overtraining Example - 10 Hidden Units



Overtraining Example - 20 Hidden Units



Redundancy

Neural Networks usually have more layers and neurons than necessary.

Training a more simple network, that is able to implement the same function, however, appear to be difficult.

Many neurons may be given random weights (and not trained) without causing problems.

--> Redundancy helps, but demands more time.

Speed and Memory

Training and testing may be very slow and memory demanding.

Special hardware helps, as it does for other procedures.

Are Neural Networks Better?

Neural networks usually are not better in simple problems

.

They don't offer fixed procedures.

They offer a complicated toolbox and not a single off-the-shelf tool.

Application demands a skilled analyst.

Averaged error rates and standard deviations over 10 runs

dataset	NMean	norm	k-NNR	1-NNR	DTree	ANN
IRIS	*0.077* 0.019	**0.025** 0.010	*0.048* 0.019	*0.053* 0.017	*0.071* 0.031	*0.052* 0.026
IMOX	*0.115* 0.027	*0.102* 0.026	0.086 0.018	0.071 0.023	0.092 0.045	0.088 0.031
80X	0.114 0.054	0.123 0.074	0.077 0.083	0.082 0.088	*0.255* 0.099	0.118 0.078
BLOOD	*0.163* 0.034	0.125 0.035	0.131 0.035	*0.153* 0.041	*0.158* 0.048	0.123 0.033
GLASS	*0.569* 0.049	*0.431* 0.098	0.303 0.040	0.286 0.045	*0.334* 0.052	*0.380* 0.075
SONAR	*0.352* 0.072	*0.315* *0.061*	0.194 0.050	0.188 0.044	*0.307* 0.043	*0.236* 0.034
DNORM	*0.334* 0.045	*0.151* 0.041	*0.185* 0.045	*0.212* 0.038	*0.344* 0.045	**0.121** 0.017

Conclusions on Neural Networks

Neural Networks are oversized, general function approximators that work because of the training rule:

- start from linearity.
- stop in one of the first moderately nonlinear local minima.
- have many (built-in) regularization possibilities, including a slow optimization rule (back-propagation).
- do not reproduce.
- are computational intensive.

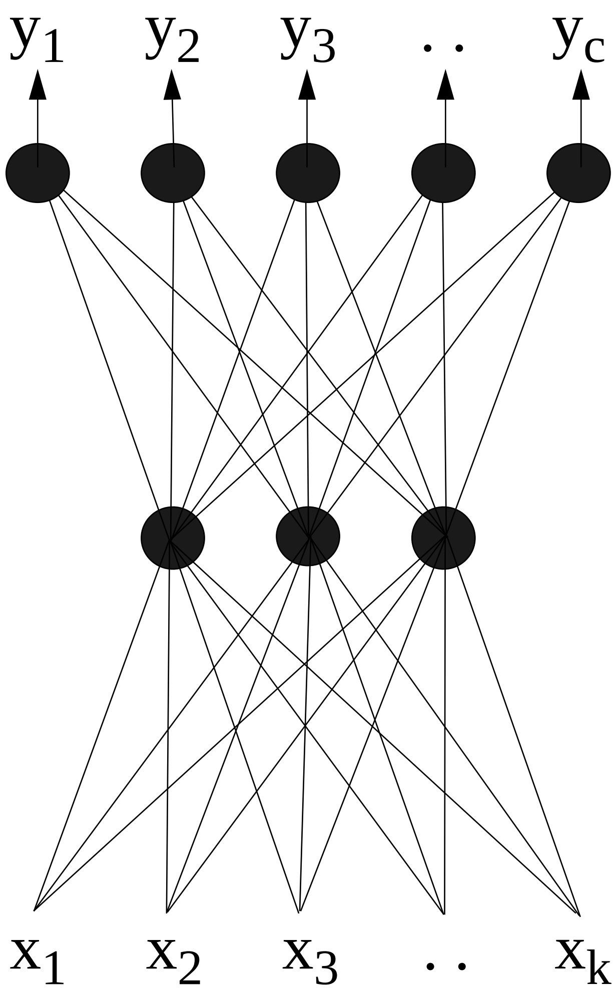
Final Conclusion on Neural Networks for Pattern Recognition

<u>Matter</u>	<u>Mind</u>
Model the Brain	Generalize the Rules
Neuro-Biology	*Artificial Intelligence*
Perception	*Statistical Pattern Recognition*
Model the Senses	Generalize from Examples (Given the Sensors)

Pattern Recognition is looking for general procedures for learning from examples.

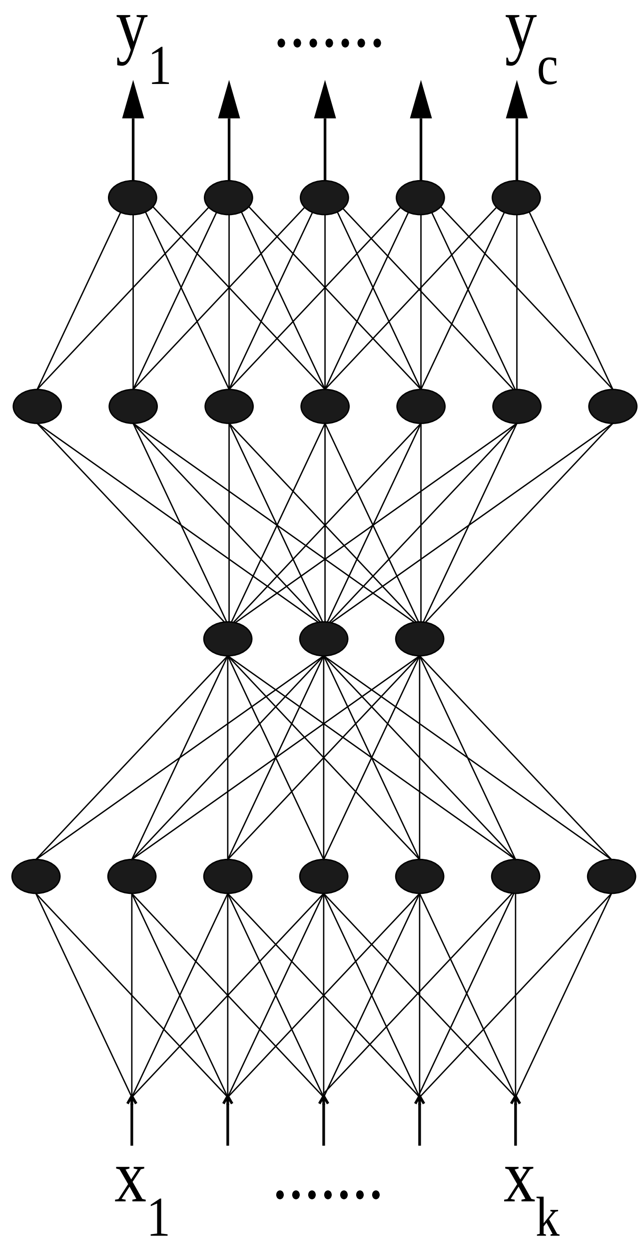
Neural Networks offer a toolbox to find specific solutions for specific problems.

Nonlinear Mapping

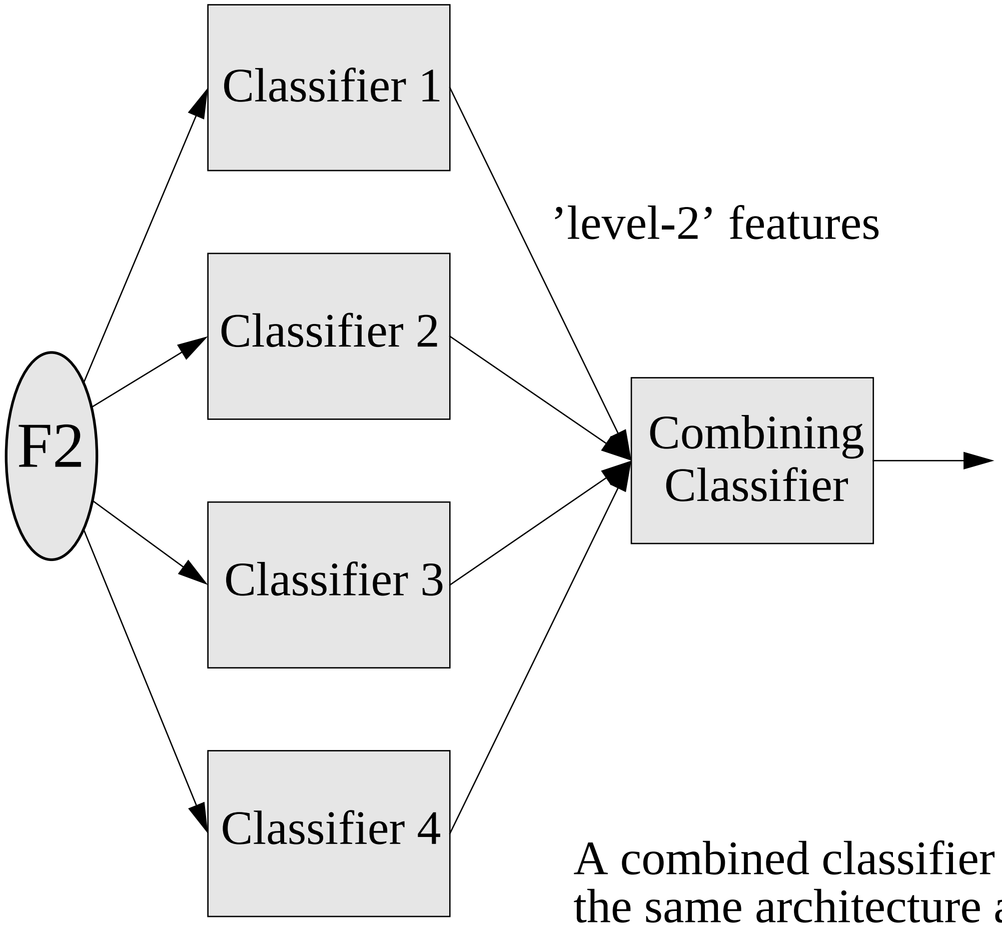


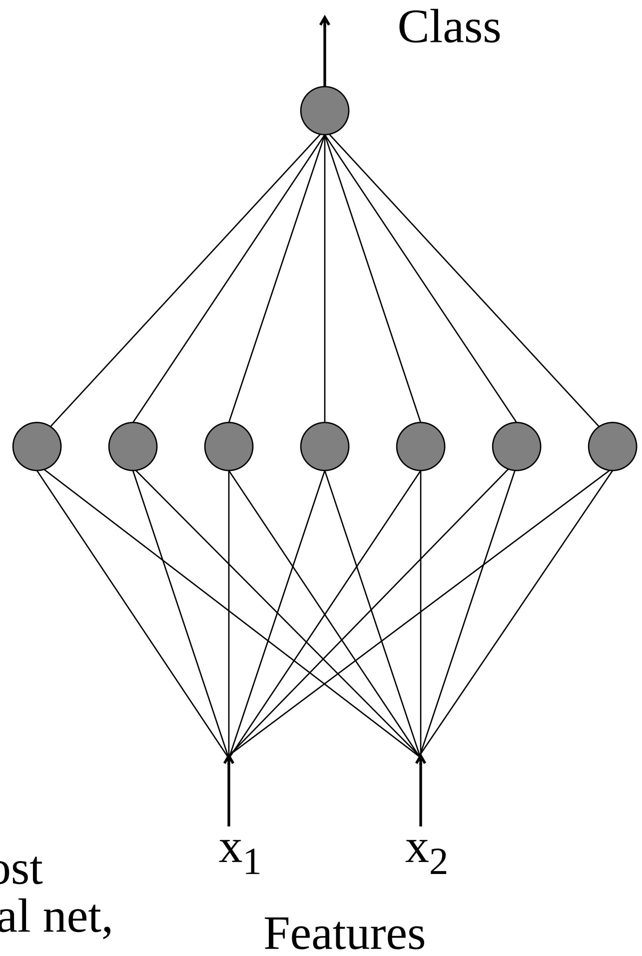
Interpretation:

1. Find a (non)linear subspace.
2. Classify.



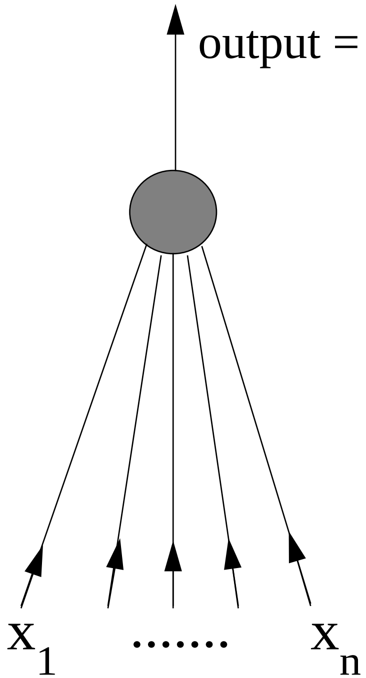
Combining Classifiers





A combined classifier has almost
the same architecture as a neural net,
but is trained differently.

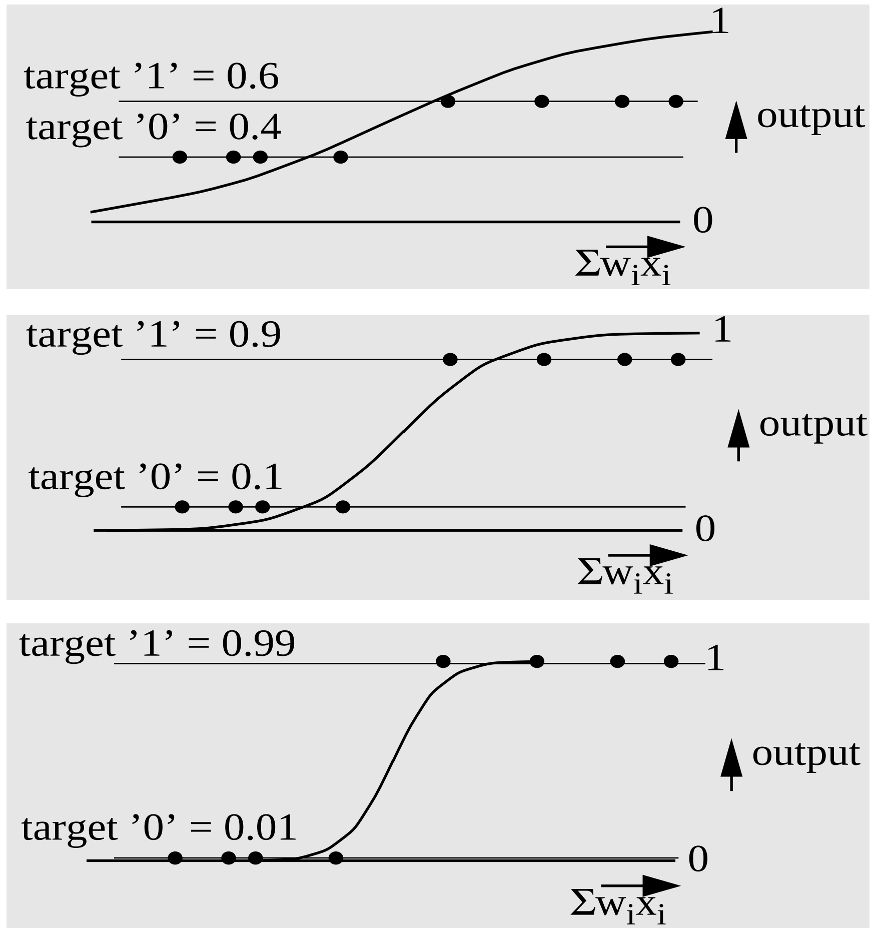
Neural Network --> Support Vector Machines

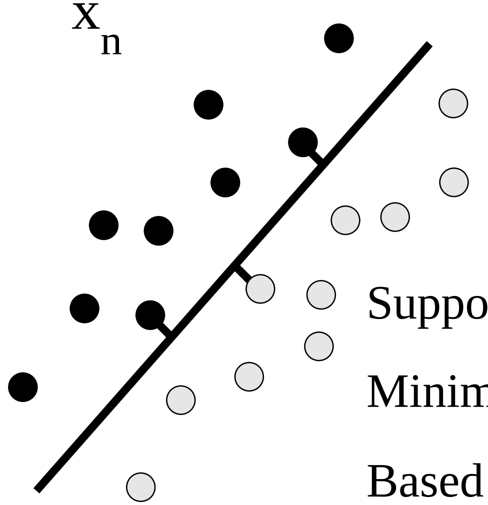


$$\text{output} = f\left(\Sigma\, w_i x_i + w_0\right) = \frac{1}{1 + e^{-\mathbf{wx} - w_0}}$$

$$\text{mse} = \sum (f_j - t_j)^2$$

t_j : target $\mathbf{x}_j$





Support Vector Classifier

Minimize training set to a support set

Based on inner products $X_i^T X_j$

What Has Been Learned

Understanding the use of redundancy, oversized systems and regularization.

The use of controlled moderate nonlinearities: Nonlinear mapping techniques.

Better classifiers: support vector machines

Soft outputs: confidences and fuzzy memberships

The construction and use of complicated systems: combined classifiers.