



THE UNIVERSITY *of* TEXAS

HEALTH SCIENCE CENTER AT HOUSTON

SCHOOL *of* HEALTH INFORMATION SCIENCES

Artificial Neural Networks and Pattern Recognition

For students of HI 5323

“Image Processing”

Willy Wriggers, Ph.D.

School of Health Information Sciences

<http://biomachina.org/courses/processing/13.html>

Biology

What are Neural Networks?

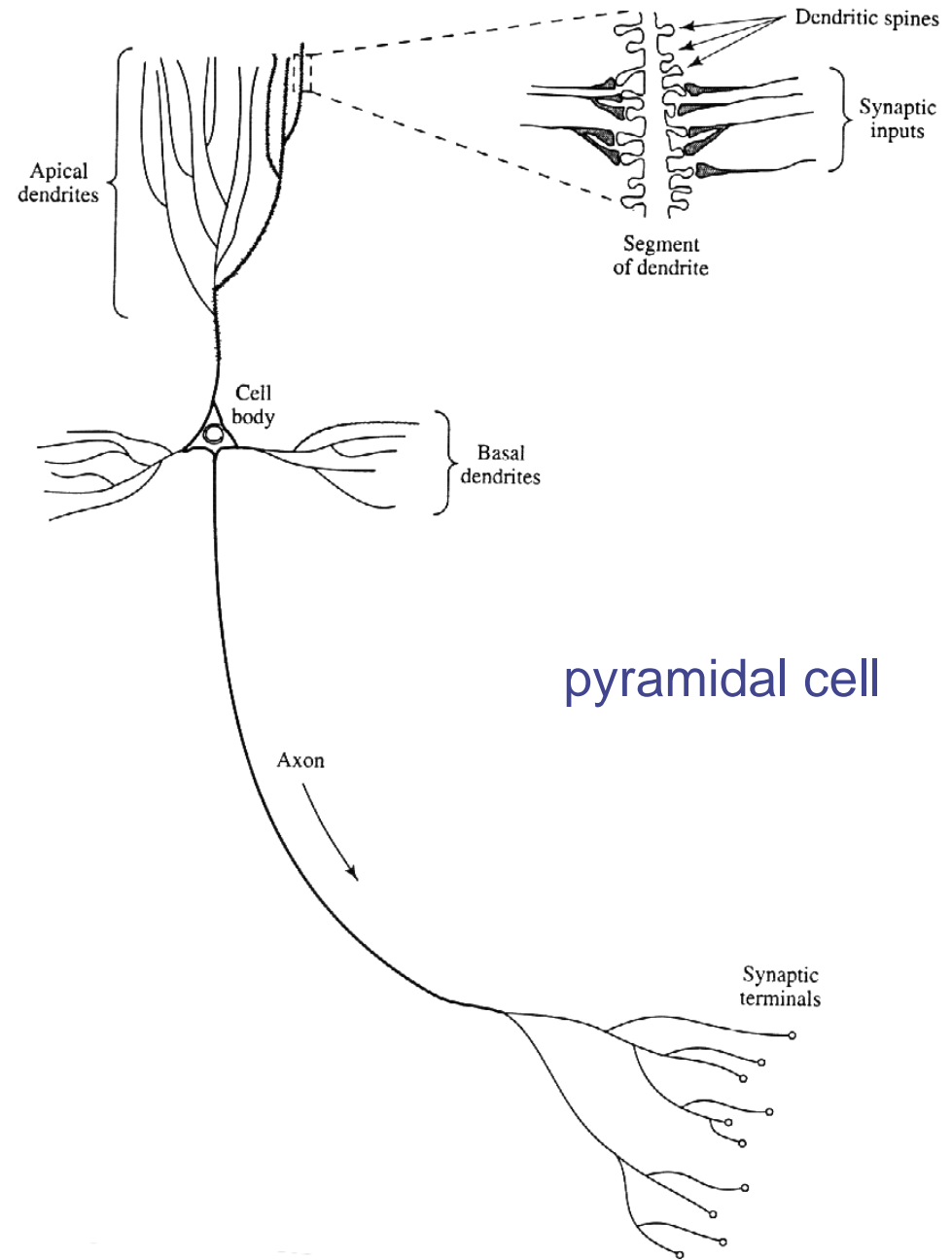
- Models of the brain and nervous system
- Highly parallel
 - Process information much more like the brain than a serial computer
- Learning

- Very simple principles
- Very complex behaviours

- Applications
 - As powerful problem solvers
 - As biological models

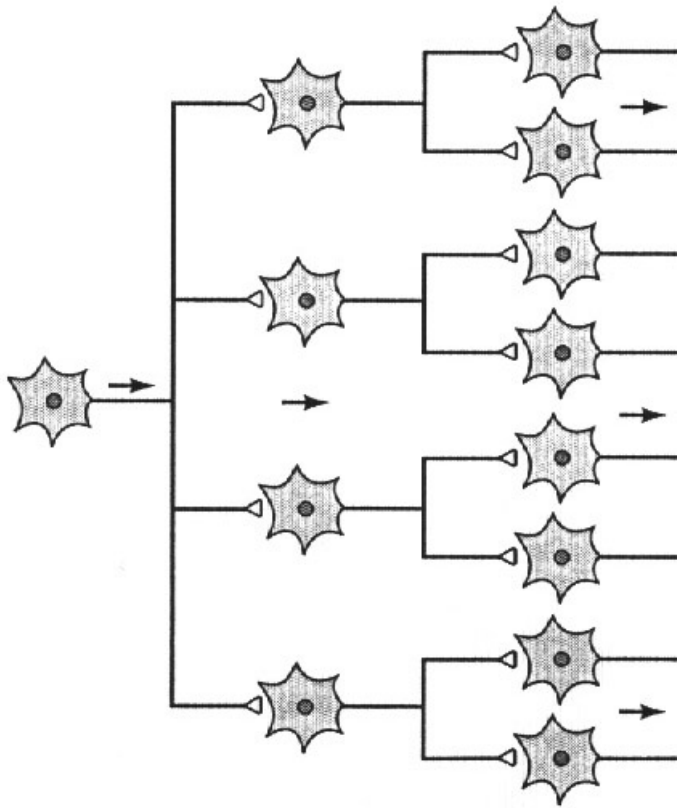
Neuro- Physiological Background

- 10 billion neurons in human cortex
- 60 trillion synapses
- In first two years from birth ~1 million synapses / sec. formed

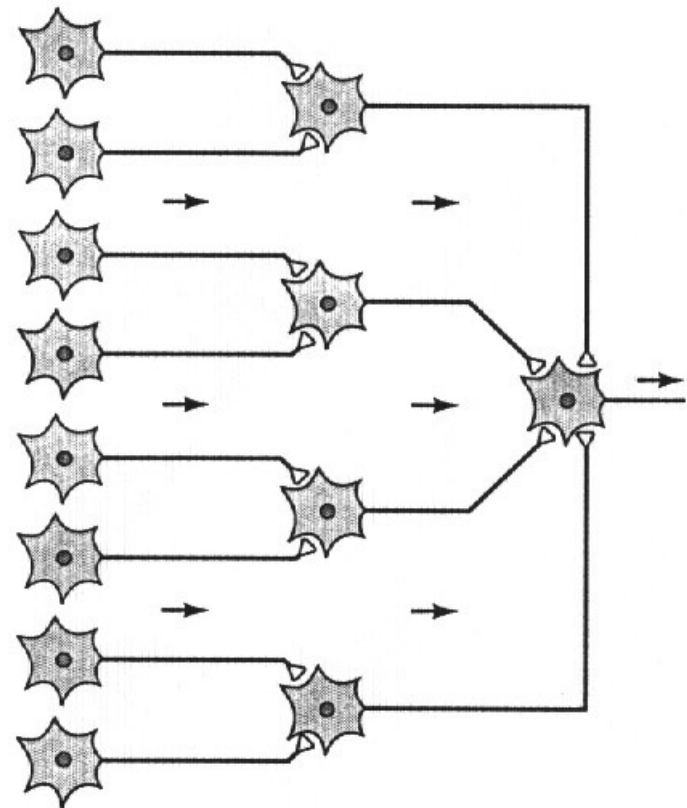


Organizing Principle

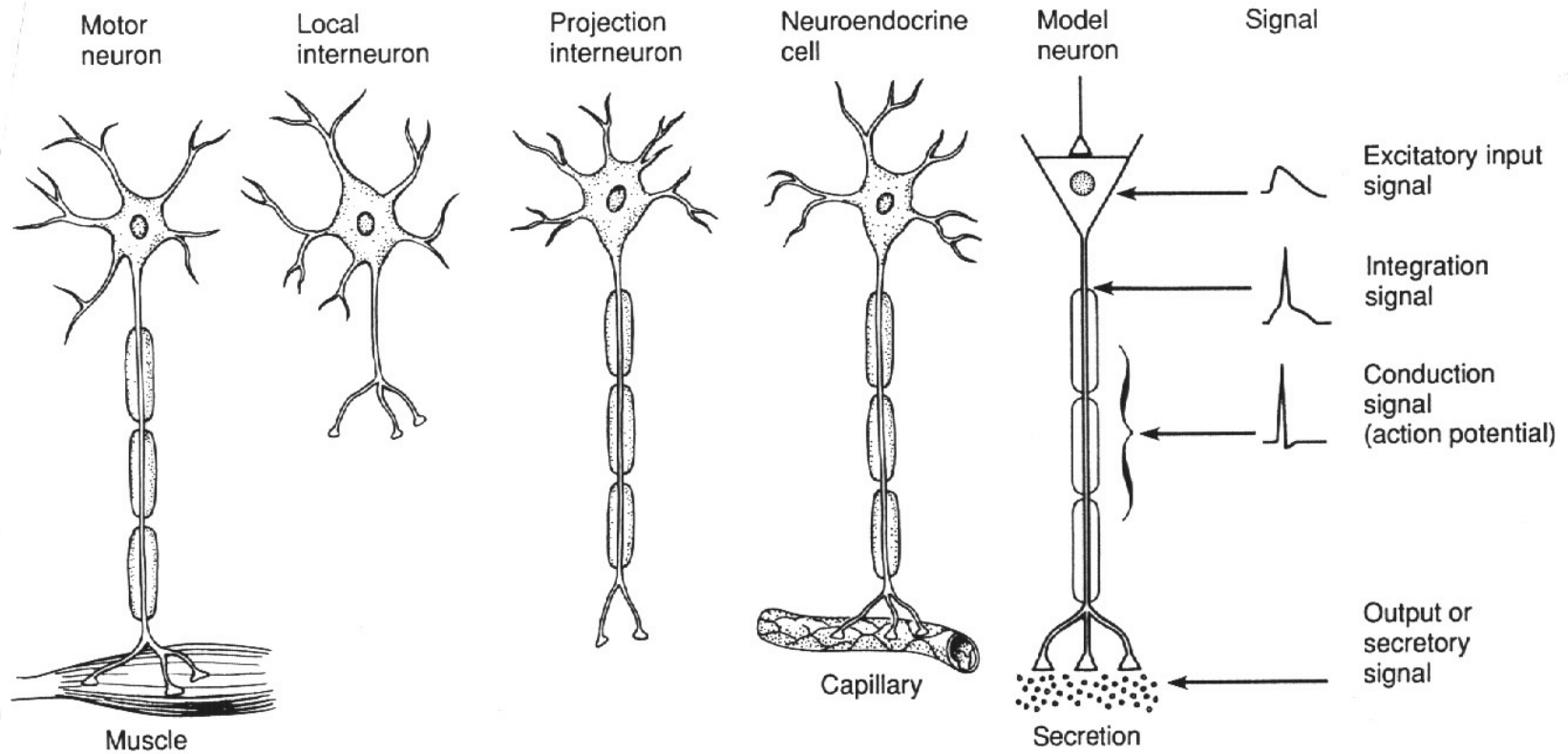
A Divergence



B Convergence

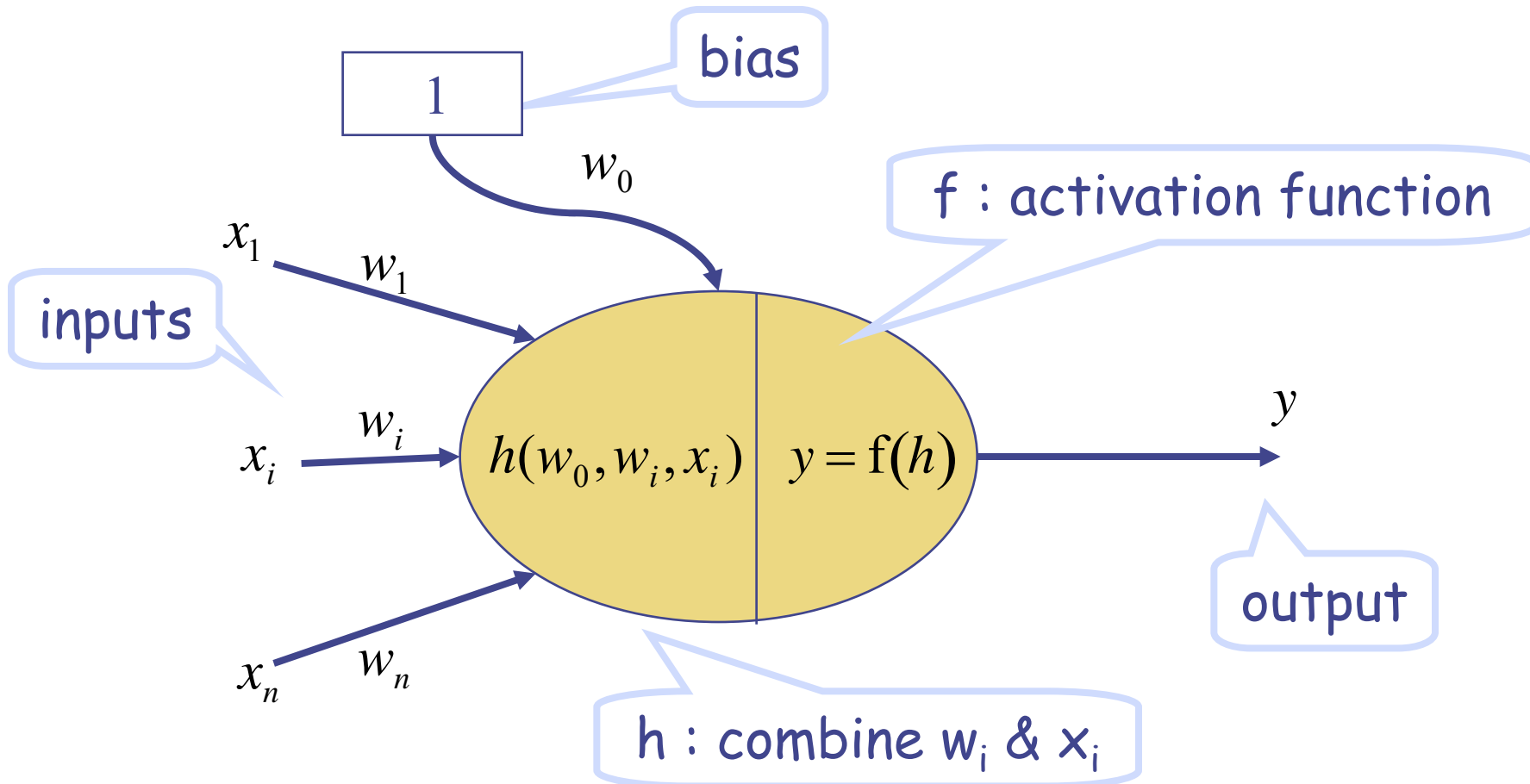


Various Types of Neurons

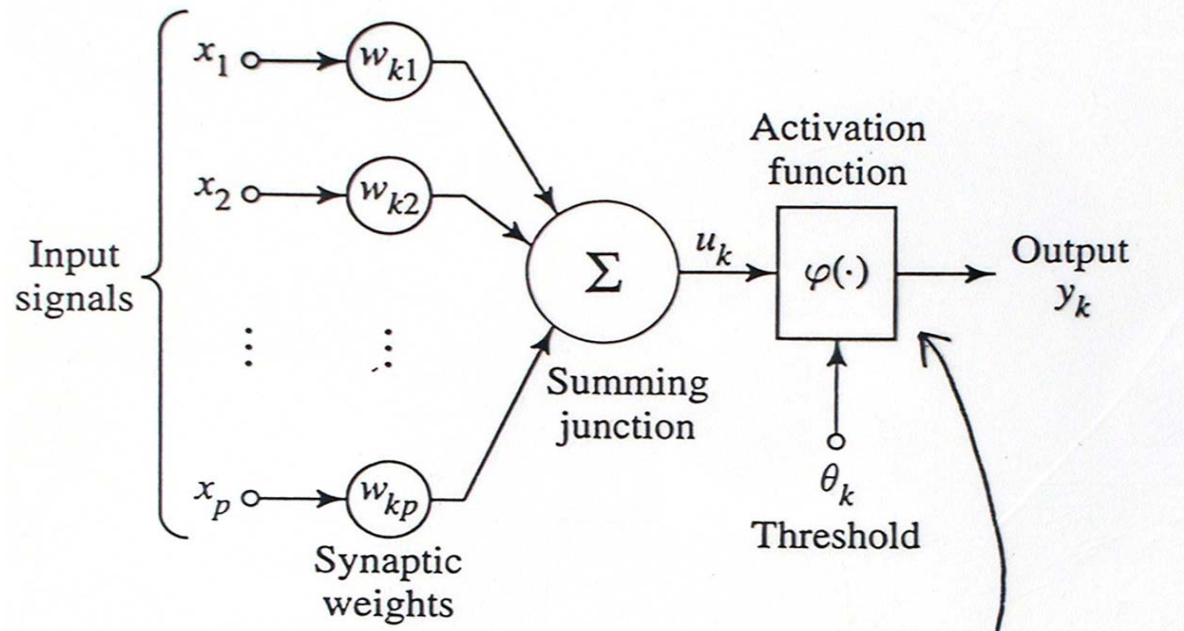


Neuron Models

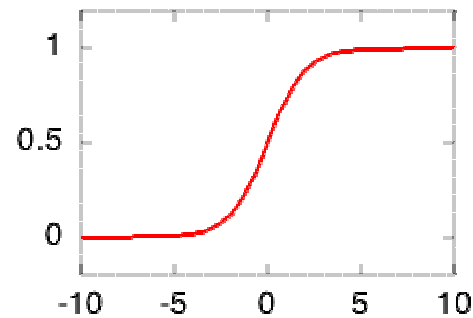
Modeling the Neuron



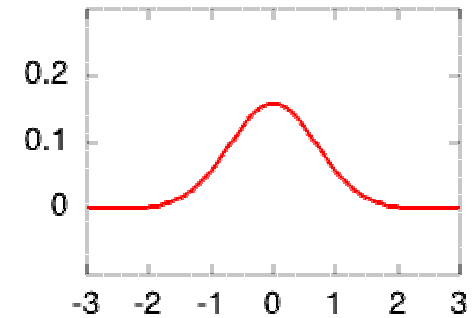
Artificial Neuron Anatomy



sigmoid with rho=1



standard gaussian



Common Activation Functions

- Sigmoidal Function:

$$y = f\left(h = w_0 \cdot 1 + \sum_{i=1}^n w_i \cdot x_i ; \rho\right) = \frac{1}{1 + e^{-h/\rho}}$$

- Radial Function, e.g.. Gaussian:

$$y = f\left(h = \sum_{i=1}^n (x_i - w_i)^2 ; \sigma = w_0\right) = \frac{1}{2\pi\sigma} e^{-\frac{h^2}{2\sigma^2}}$$

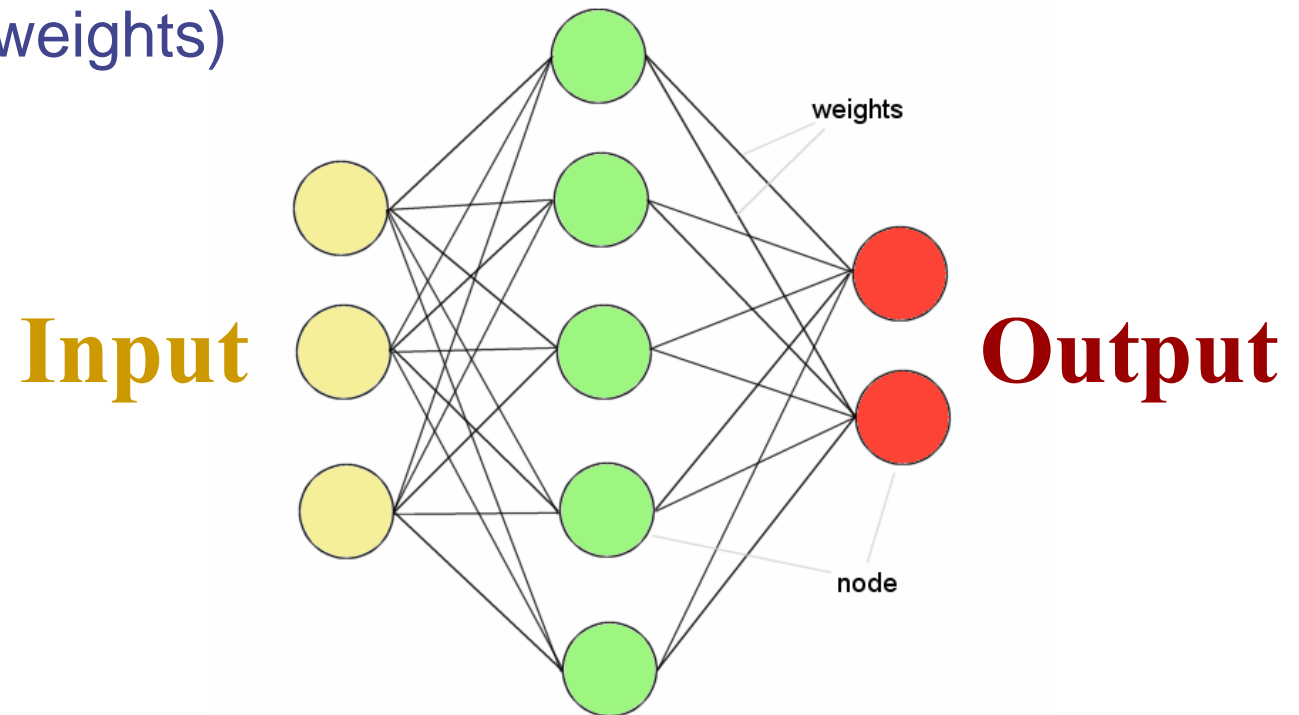
- Linear Function

$$y = w_0 \cdot 1 + \sum_{i=1}^n w_i \cdot x_i$$

Supervised Learning

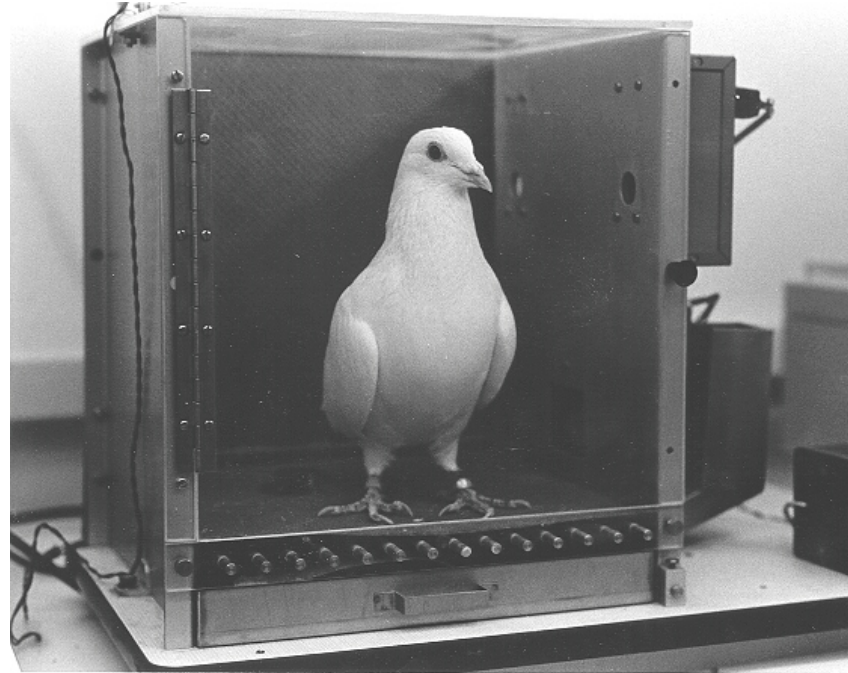
Artificial Neural Networks

- ANNs incorporate the two fundamental components of biological neural nets:
 1. Neurones (nodes)
 2. Synapses (weights)



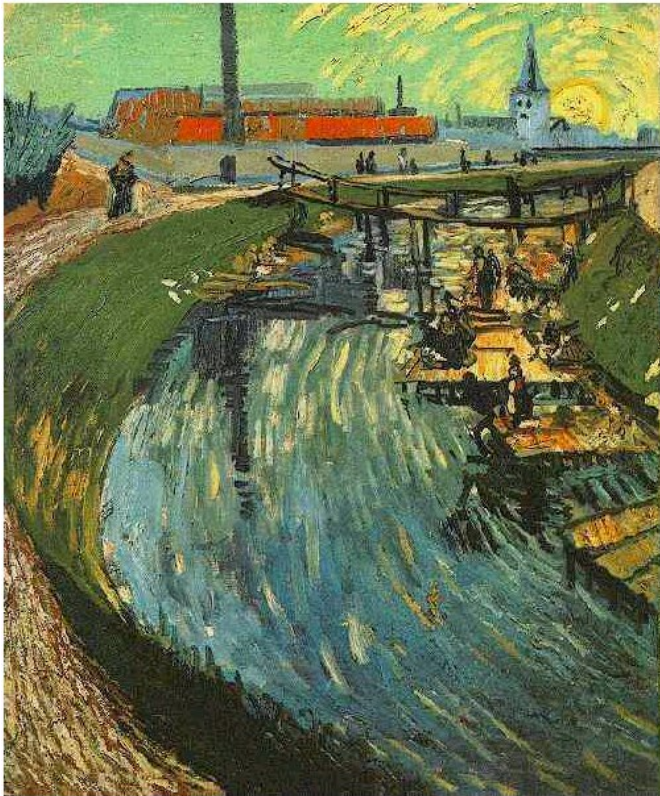
“Pidgeon” ANNs

- Pigeons as art experts (Watanabe *et al.* 1995)



- Experiment:
 - Pigeon in Skinner box
 - Present paintings of two different artists (e.g. Chagall / Van Gogh)
 - Reward for pecking when presented a particular artist (e.g. Van Gogh)

Training Set:



(etc...)

Predictive Power:

- Pigeons were able to discriminate between Van Gogh and Chagall with 95% accuracy (when presented with pictures they had been trained on)
- Discrimination still 85% successful for previously unseen paintings of the artists.
- Pigeons do not simply memorise the pictures
- They can extract and recognise patterns (the ‘style’)
- They generalise from the already seen to make predictions
- This is what neural networks (biological and artificial) are good at (unlike conventional computer)

Real ANN Applications

- Recognition of hand-written letters
- Predicting on-line the quality of welding spots
- Identifying relevant documents in corpus
- Visualizing high-dimensional space
- Tracking on-line the position of robot arms
- ... etc

ANN Design

1. Get a large amount of data: inputs and outputs
2. Analyze data on the PC
 - Relevant inputs ?
 - Linear correlations (ANN necessary) ?
 - Transform and scale variables
 - Other useful preprocessing ?
 - Divide in 3 data sets:
 - Training set
 - Test set
 - Validation set

ANN Design

3. Set the ANN architecture: What type of ANN ?

- Number of inputs, outputs ?
- Number of hidden layers
- Number of neurons
- Learning schema « details »

4. Tune/optimize internal parameters by presenting training data set to ANN

5. Validate on test / validation dataset

Main Types of ANN

Supervised Learning:

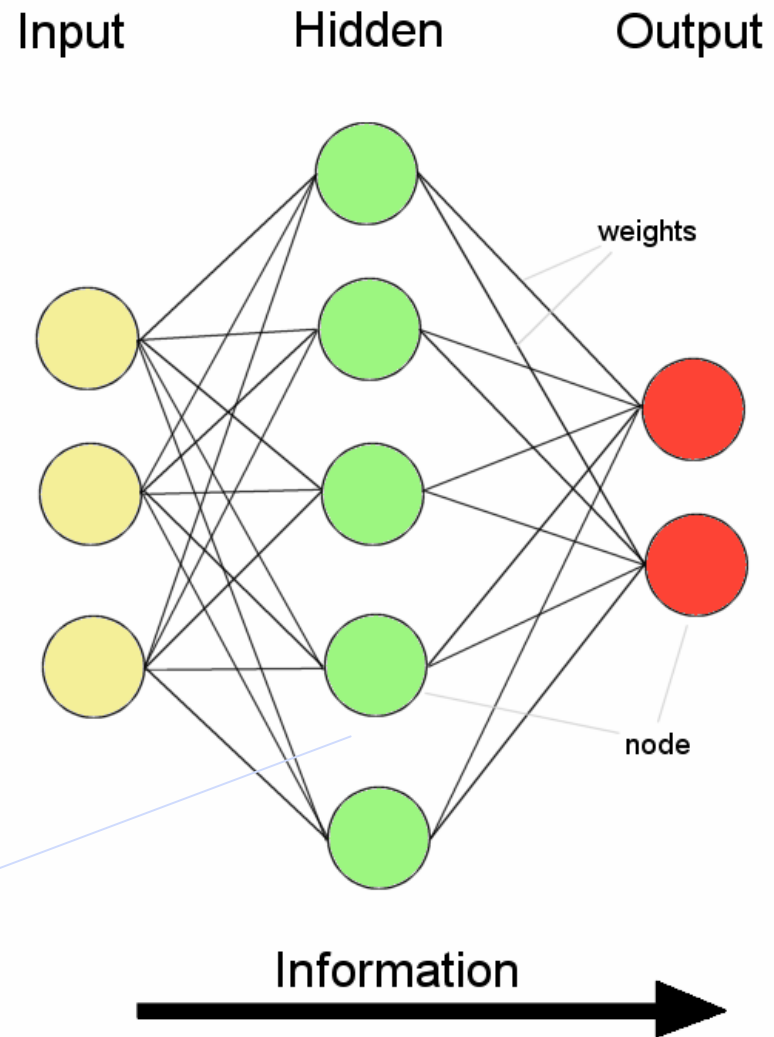
- Feed-forward ANN
 - Multi-Layer Perceptron (with sigmoid hidden neurons)
- Recurrent Networks
 - Neurons are connected to self and others
 - Time delay of signal transfer
 - Multidirectional information flow

Unsupervised Learning:

- Self-organizing ANN
 - Kohonen Maps
 - Vector Quantization
 - Neural Gas

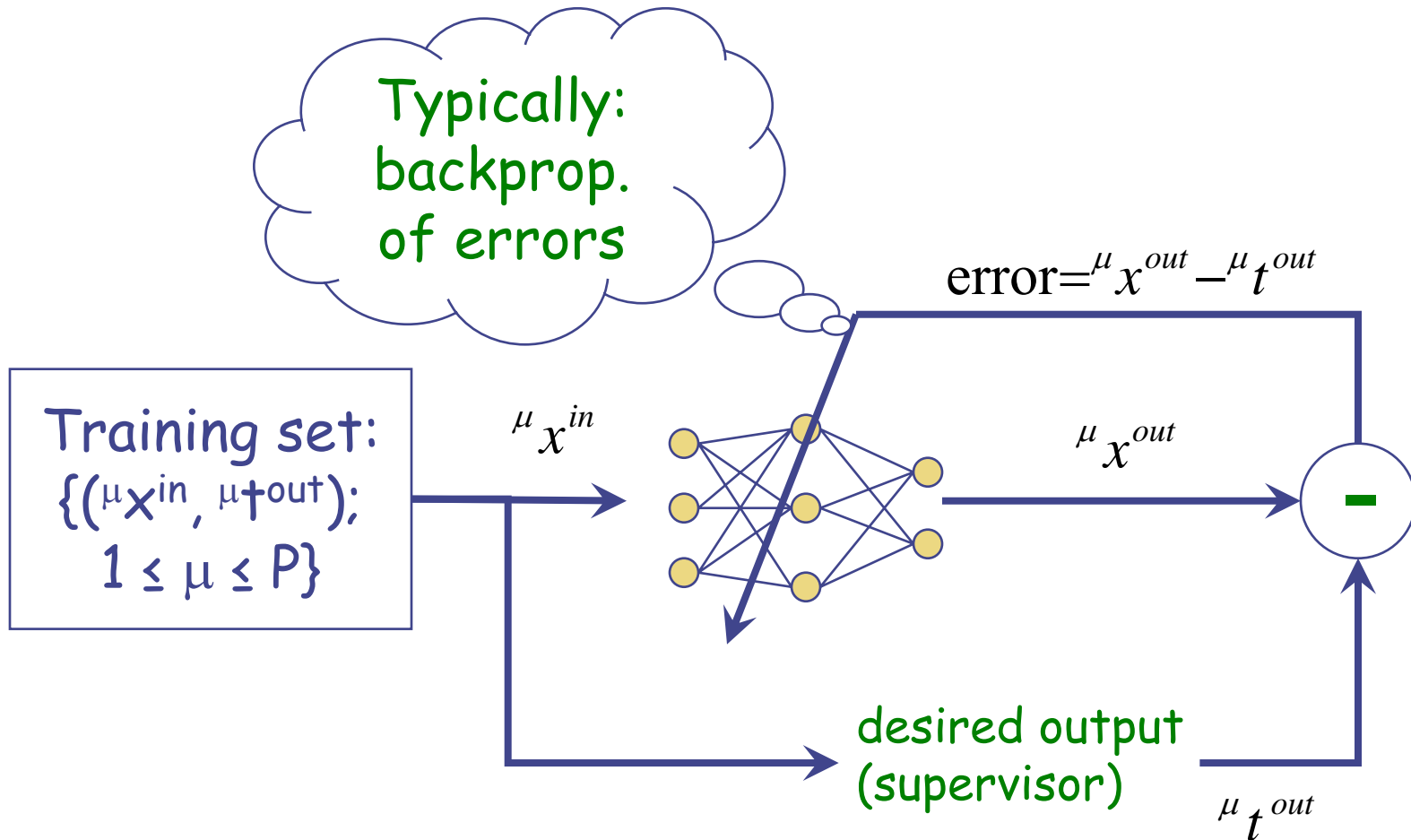
Feed-Forward ANN

- Information flow is unidirectional
 - Data is presented to *Input layer*
 - Passed on to *Hidden Layer*
 - Passed on to *Output layer*
- Information is distributed
- Information processing is parallel



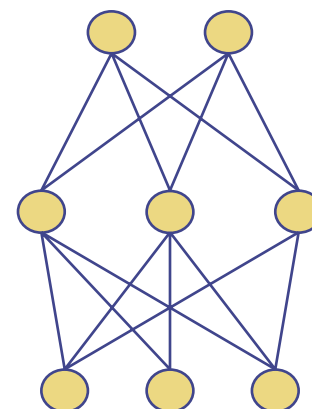
Internal representation (interpretation) of data

Supervised Learning



Important Properties of FFN

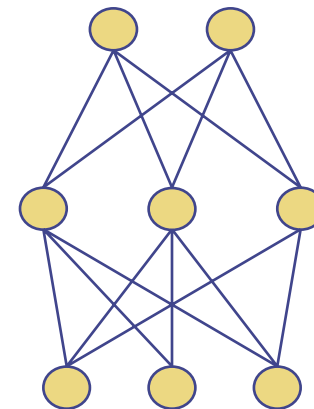
- Assume
 - $g(\mathbf{x})$: bounded and sufficiently regular fct.
 - FFN with 1 hidden layer of finite N neurons
(Transfer function is identical for every neurons)
- \Rightarrow FFN is an **Universal Approximator** of $g(\mathbf{x})$
Theorem by Cybenko et al. in 1989
In the sense of uniform approximation
For arbitrary precision ε



Important Properties of FFN

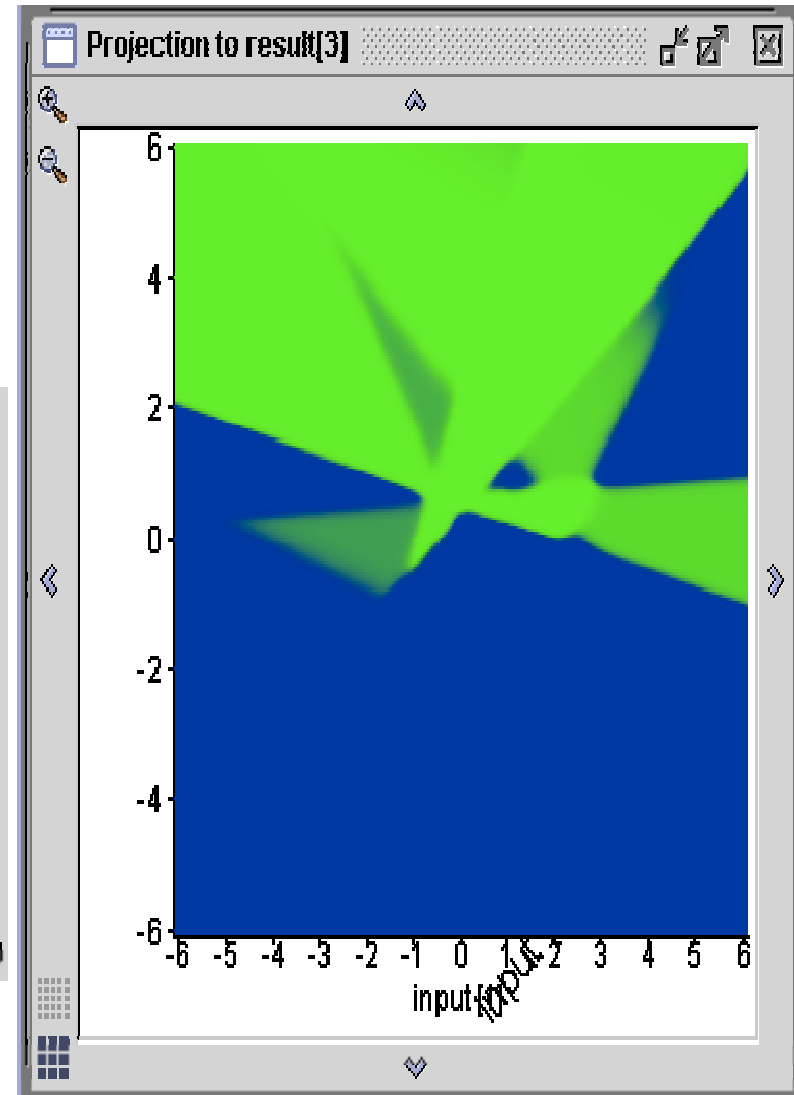
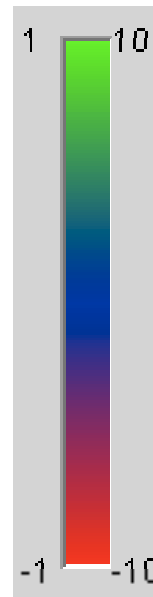
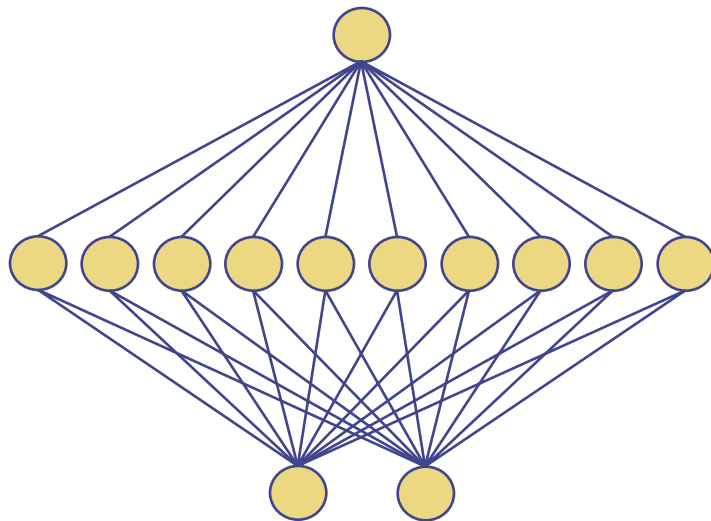
- Assume
 - FFN as before
(1 hidden layer of finite N neurons, non linear transfer function)
 - Approximation precision ε
- $\Rightarrow \#\{w_i\} \sim \# \text{ inputs}$
Theorem by Barron in 1993

ANN is **more parsimonious** in $\#\{w_i\}$ than a linear approximator
[linear approximator: $\#\{w_i\} \sim \exp(\# \text{ inputs})$]

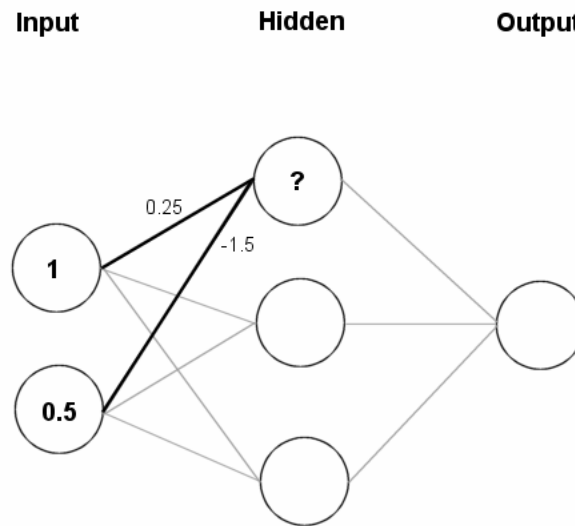


Roughness of Output

- Outputs depends of the whole set of weighted links $\{w_{ij}\}$
- Example: output unit versus input 1 and input 2 for a $2 \times 10 \times 1$ ANN with random weights



Feeding Data Through the FNN



$$(1 \times 0.25) + (0.5 \times (-1.5)) = 0.25 + (-0.75) = -\mathbf{0.5}$$

Squashing: $\frac{1}{1 + e^{0.5}} = 0.3775$

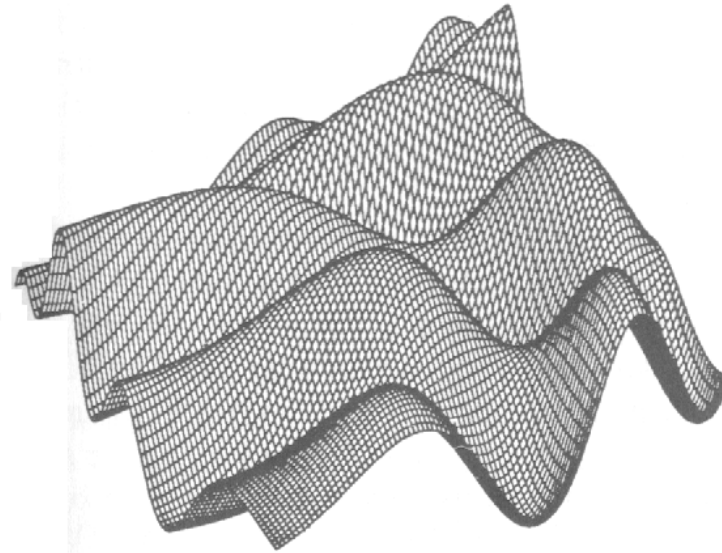
Feeding Data Through the FNN

- Data is presented to the network in the form of activations in the input layer
- Examples
 - Pixel intensity (for pictures)
 - Molecule concentrations (for artificial nose)
 - Share prices (for stock market prediction)
- Data usually requires preprocessing
 - Analogous to senses in biology
- How to represent more abstract data, e.g. a name?
 - Choose a pattern, e.g.
 - 0-0-1 for “Chris”
 - 0-1-0 for “Becky”

Training the Network

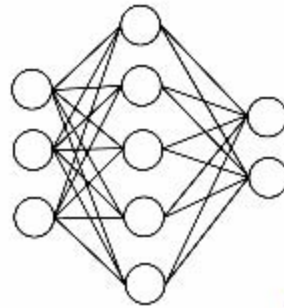
How do we adjust the weights?

- Backpropagation
 - Requires training set (input / output pairs)
 - Starts with small random weights
 - Error is used to adjust weights (supervised learning)
- Gradient descent on error landscape



Backpropagation

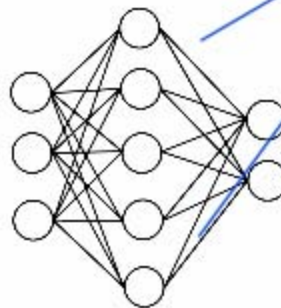
1.



Wallace

Wallace - Darwin (calculate error)

2.

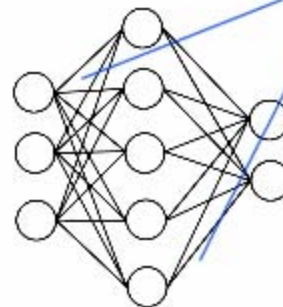


adjust weights

Wallace'

Wallace - Darwin (calculate error)

3.




adjust weights

Darwin

Backpropagation

- **Advantages**
 - It works!
 - Relatively fast
- **Downsides**
 - Requires a training set
 - Can be slow to converge
 - Probably not biologically realistic
- **Alternatives to Backpropagation**
 - Hebbian learning
 - Not successful in feed-forward nets
 - Reinforcement learning
 - Only limited success in FFN
 - Artificial evolution
 - More general, but can be even slower than backprop

Applications of FFN

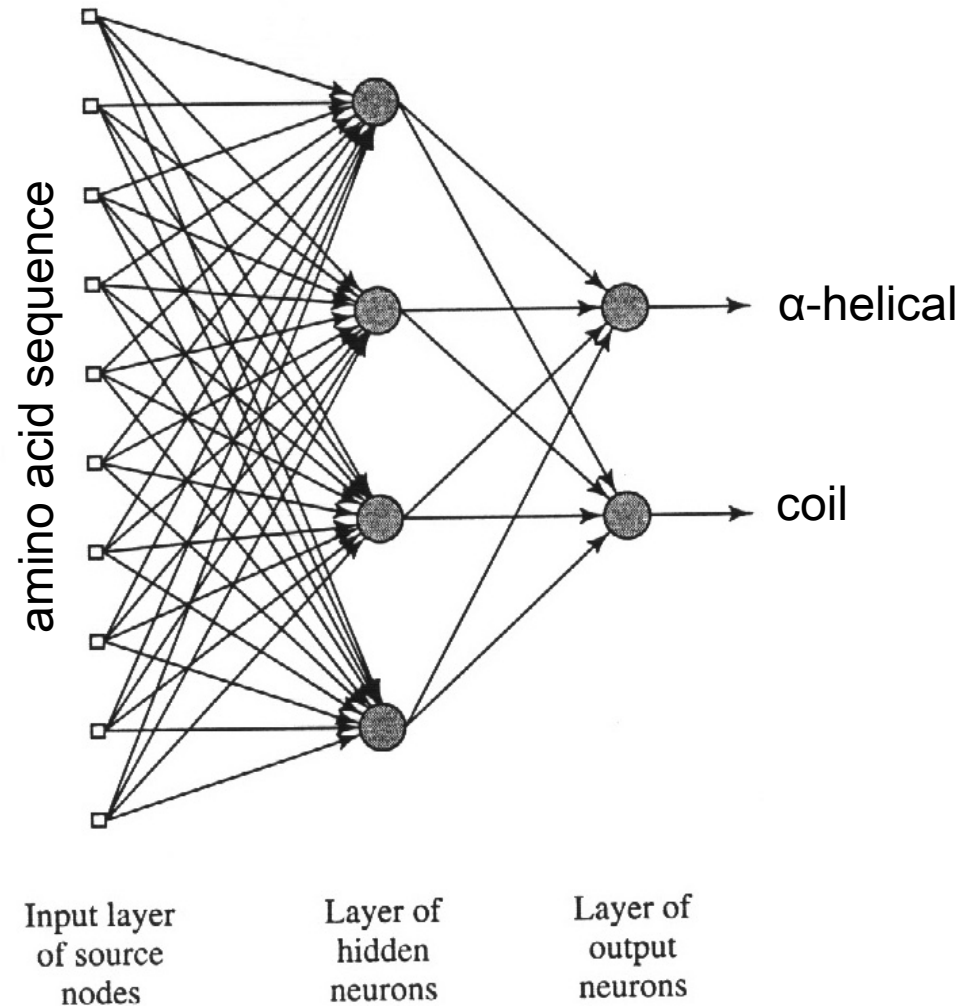
- Pattern recognition
 - Character recognition
 - Face Recognition
- Sonar mine/rock recognition (Gorman & Sejnowski, 1988)
- Navigation of a car (Pomerleau, 1989)
- Stock-market prediction
- Pronunciation (NETtalk) 
(Sejnowski & Rosenberg, 1987)

Protein Secondary Structure Prediction

(Holley-Karplus, Ph.D., etc):

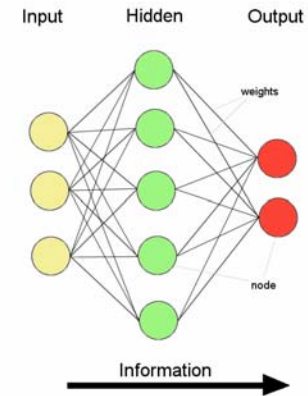
Supervised learning:

- Adjust weight vectors so output of network matches desired result



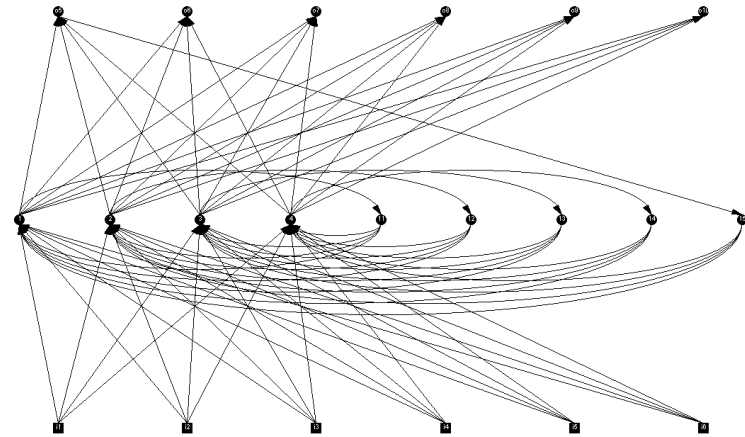
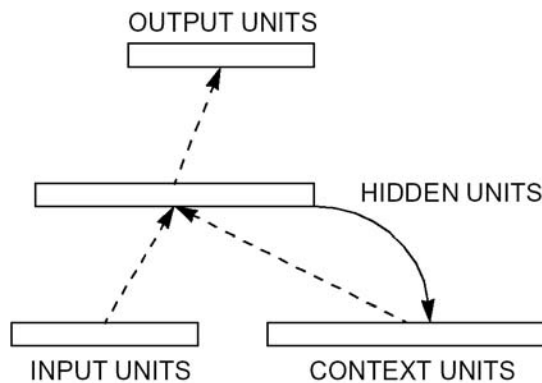
Recurrent Networks

- Feed forward networks:
 - Information only flows one way
 - One input pattern produces one output
 - No sense of time (or memory of previous state)
- Recurrency
 - Nodes connect back to other nodes or themselves
 - Information flow is multidirectional
 - Sense of time and memory of previous state(s)
- Biological nervous systems show high levels of recurrency (but feed-forward structures exists too)



Elman Nets

- *Elman nets* are feed forward networks with partial recurrency



- Unlike feed forward nets, Elman nets have a *memory* or *sense of time*

Elman Nets

Classic experiment on language acquisition and processing (Elman, 1990)

- **Task**

- Elman net to predict successive words in sentences.

- **Data**

- Suite of sentences, e.g.
 - “The boy catches the ball.”
 - “The girl eats an apple.”
- Words are input one at a time

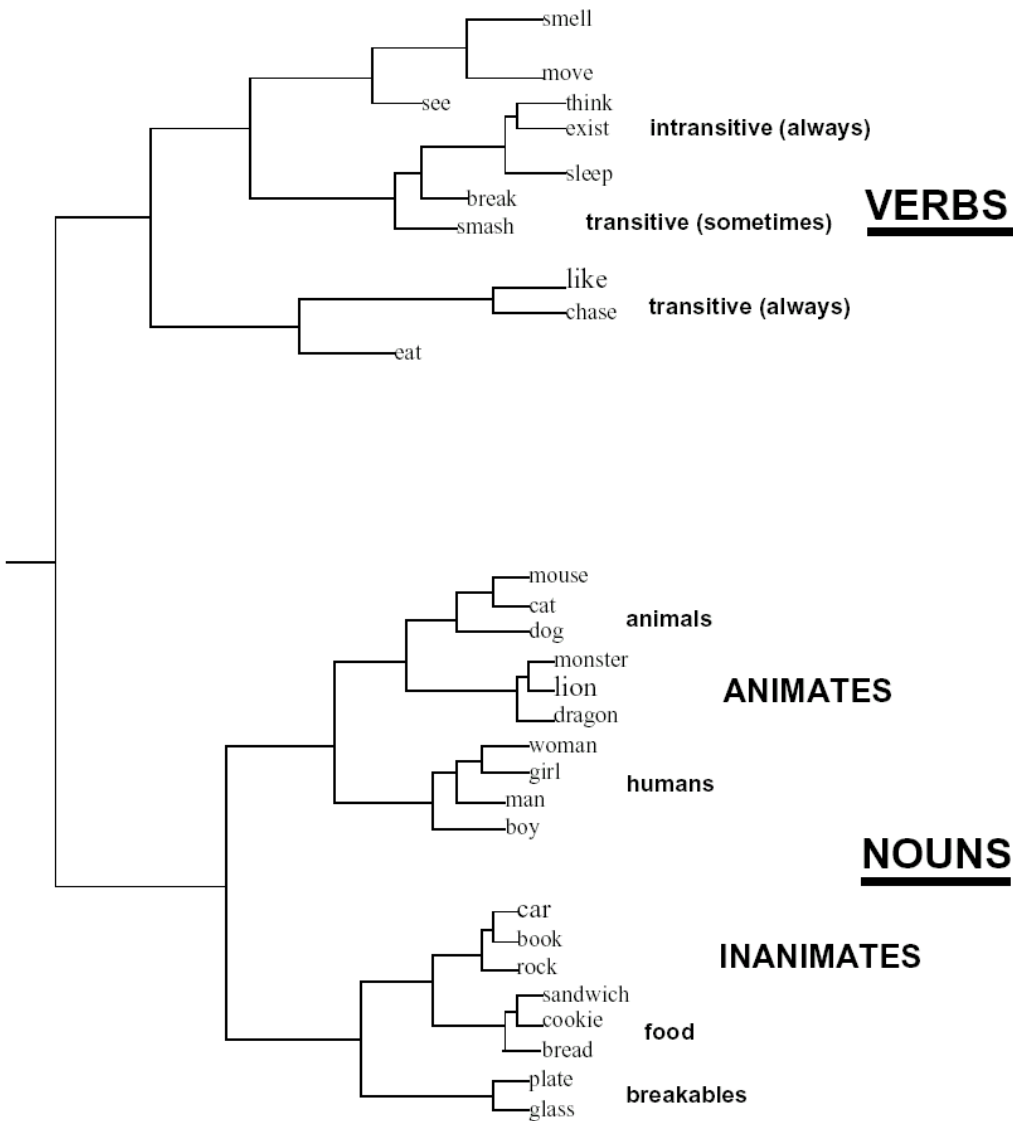
- **Representation**

- Binary representation for each word, e.g.
 - 0-1-0-0-0 for “girl”

- **Training method**

- Backpropagation

Elman Nets

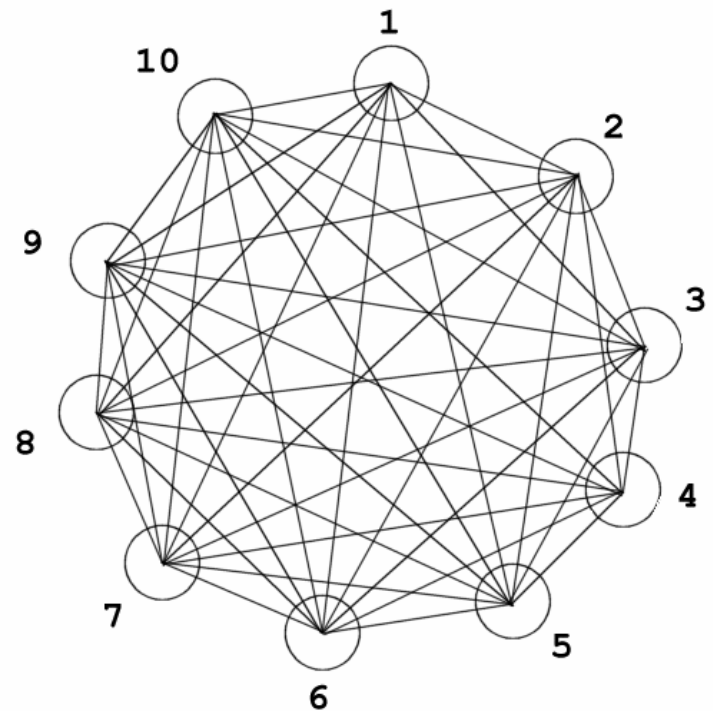


Internal
representation
of words

Hopfield Networks

- Sub-type of recurrent neural nets
 - Fully recurrent
 - Weights are symmetric
 - Nodes can only be *on* or *off*
 - Random updating
- Learning: **Hebb rule** (cells that fire together wire together)
- Can recall a memory, if presented with a corrupt or incomplete version

→ **auto-associative** or
content-addressable memory



Hopfield Networks

Task: store images with resolution of 20x20 pixels

→ Hopfield net with 400 nodes

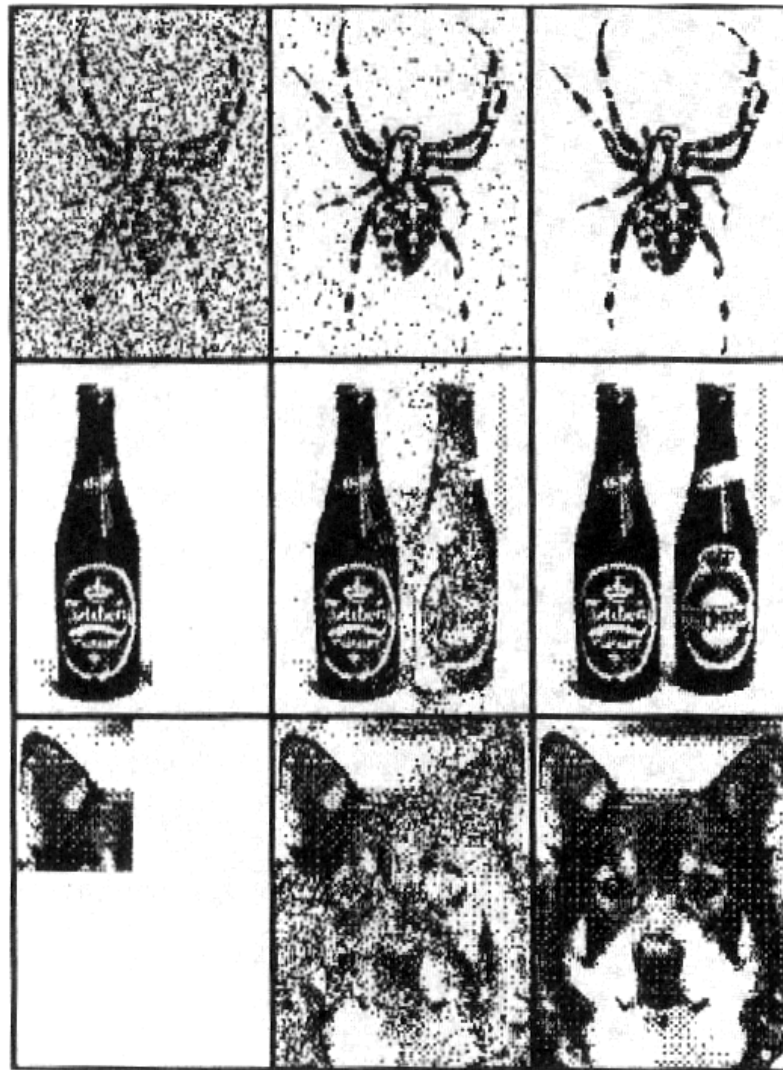
Memorise:

1. Present image
2. Apply Hebb rule (*cells that fire together, wire together*)
 - Increase weight between two nodes if both have same activity, otherwise decrease
3. Go to 1

Recall:

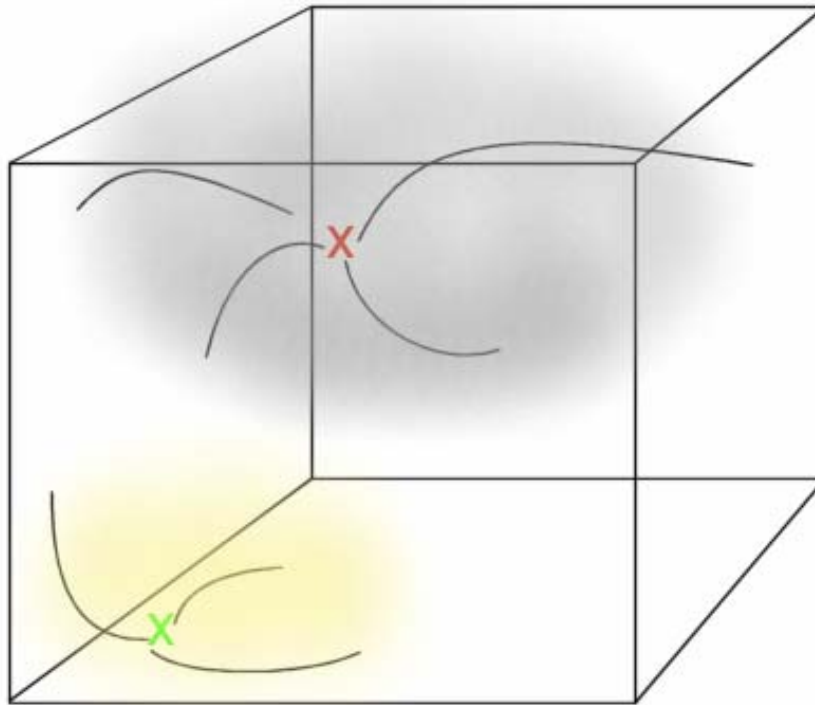
1. Present incomplete pattern
2. Pick random node, update
3. Go to 2 until settled

Hopfield Networks



Hopfield Networks

- Memories are attractors in state space



Catastrophic Forgetting

- *Problem:* memorising new patterns corrupts the memory of older ones
→ Old memories cannot be recalled, or spurious memories arise
- *Solution:* allow Hopfield net to **sleep**

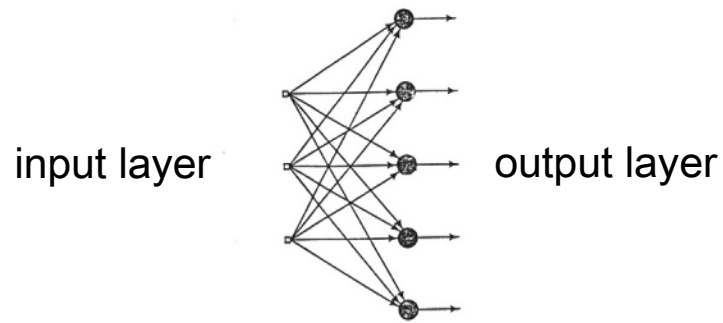
Solutions

- **Unlearning** (Hopfield, 1986)
 - Recall old memories by random stimulation, but use an *inverse* Hebb rule
 - ‘Makes room’ for new memories (basins of attraction shrink)

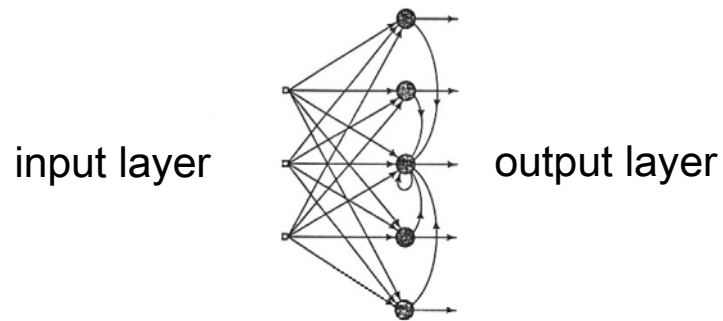
- **Pseudorehearsal** (Robins, 1995)
 - While learning new memories, recall old memories by random stimulation
 - Use *standard* Hebb rule on new and old memories
 - Restructure memory
 - Needs short-term + long term memory
 - Mammals: hippocampus plays back new memories to neo-cortex, which is randomly stimulated at the same time

Unsupervised Learning

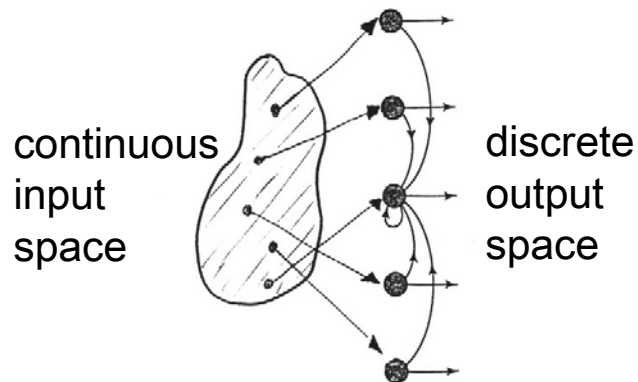
Unsupervised (Self-Organized) Learning



feed-forward (supervised)



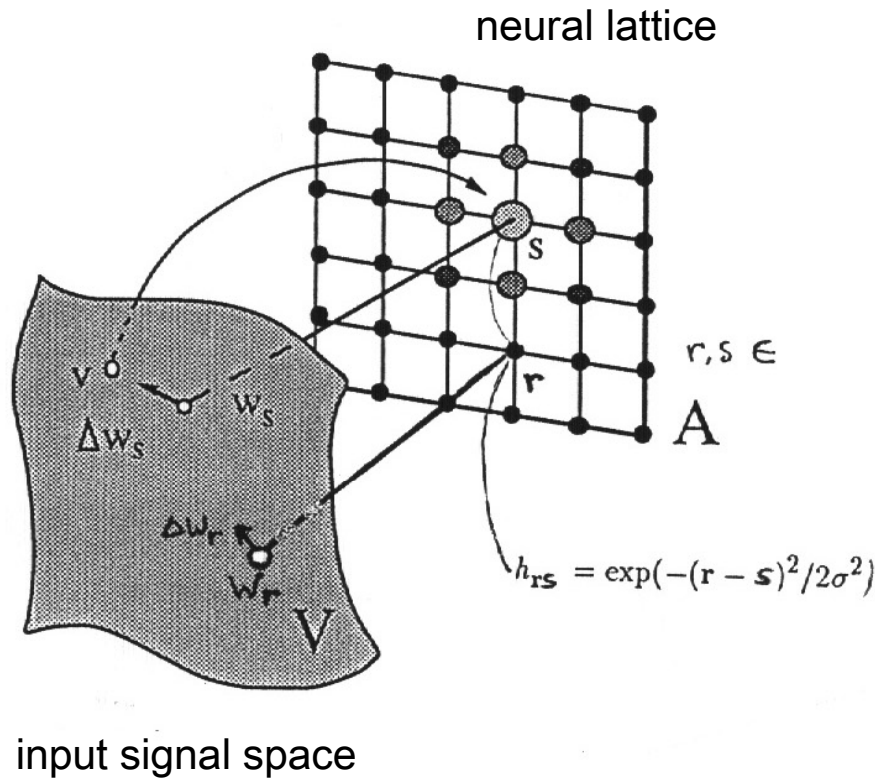
feed-forward + lateral feedback
(recurrent network, still supervised)



self-organizing network (unsupervised)

Self Organizing Map (SOM)

Kohonen, 1984



0. *Initialization*: Start with appropriate initial values for the synaptic strengths w_r . In the absence of any a priori information, the w_r can be chosen at random.
1. *Choice of Stimulus*: Choose, according to the probability density $P(v)$, a random vector v representing a "sensory signal."
2. *Response*: Determine the corresponding "excitation center" S from the condition

$$\|v - w_s\| \leq \|v - w_r\| \quad \text{for all } r \in A. \quad)$$

3. *Adaptation Step*: Carry out a "learning step" by changing the synaptic strengths according to

$$w_r^{new} = w_r^{old} + \epsilon h_{rs} (v - w_r^{old})$$

and continue with step 1.

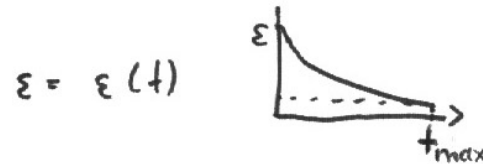
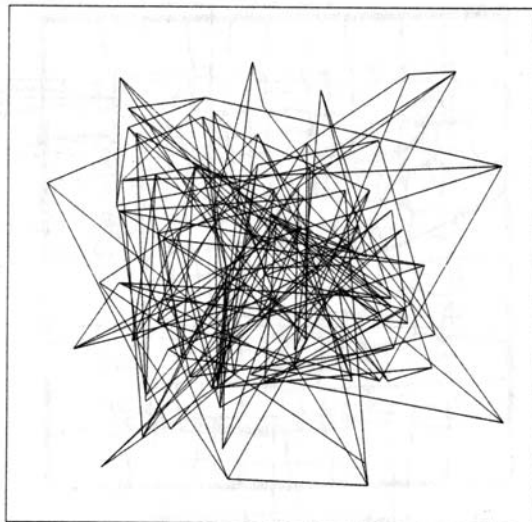


Illustration of Kohonen Learning

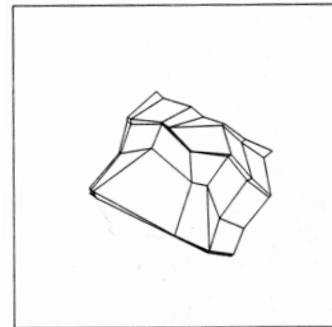
Inputs: coordinates (x,y) of points
drawn from a square

Display neuron j at position x_j, y_j where
its s_j is maximum

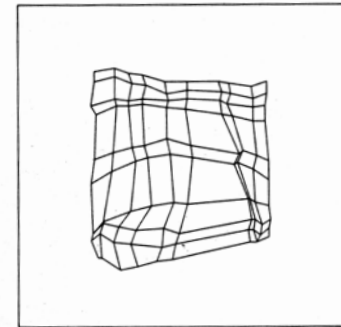


random initial positions

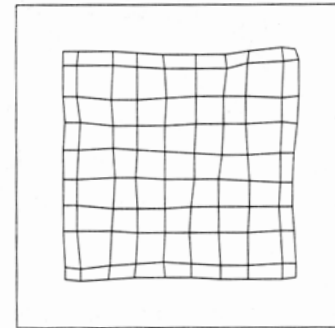
100 inputs



200 inputs



1000 inputs

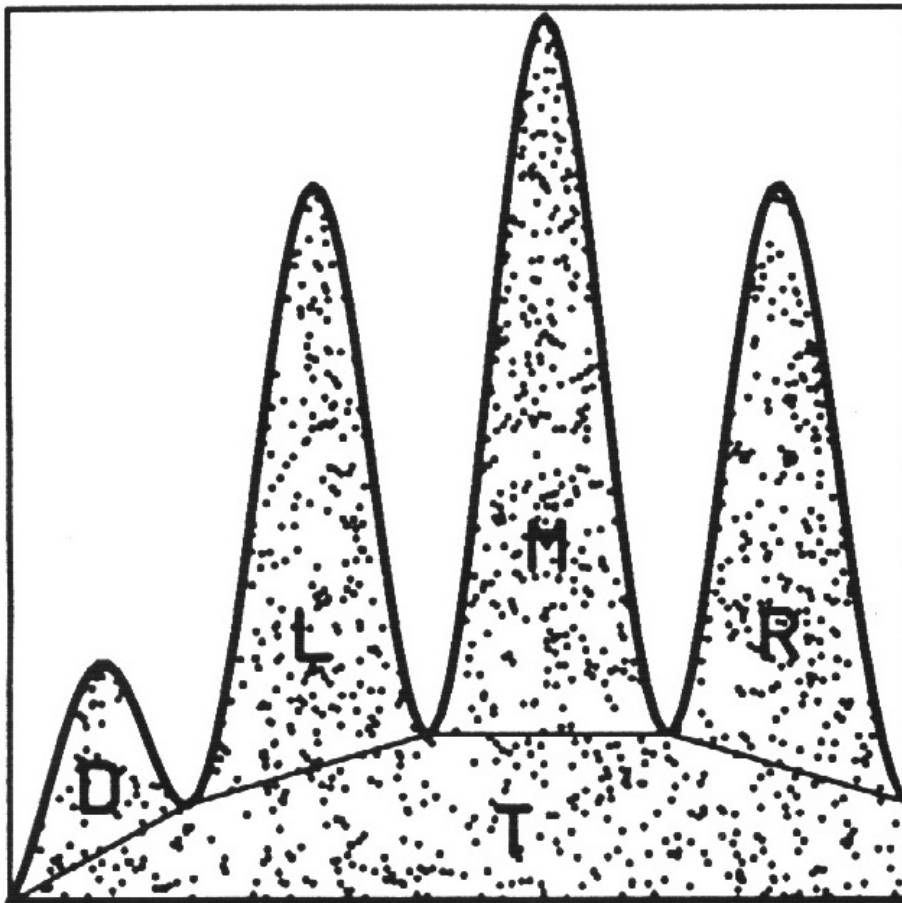


Why use Kohonen Maps?

- Image Analysis
 - Image Classification
- Data Visualization
 - By projection from high D \rightarrow 2D
 - Preserving neighborhood relationships
- Partitioning Input Space
 - Vector Quantization (Coding)

Example:

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992).



Example:

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992).

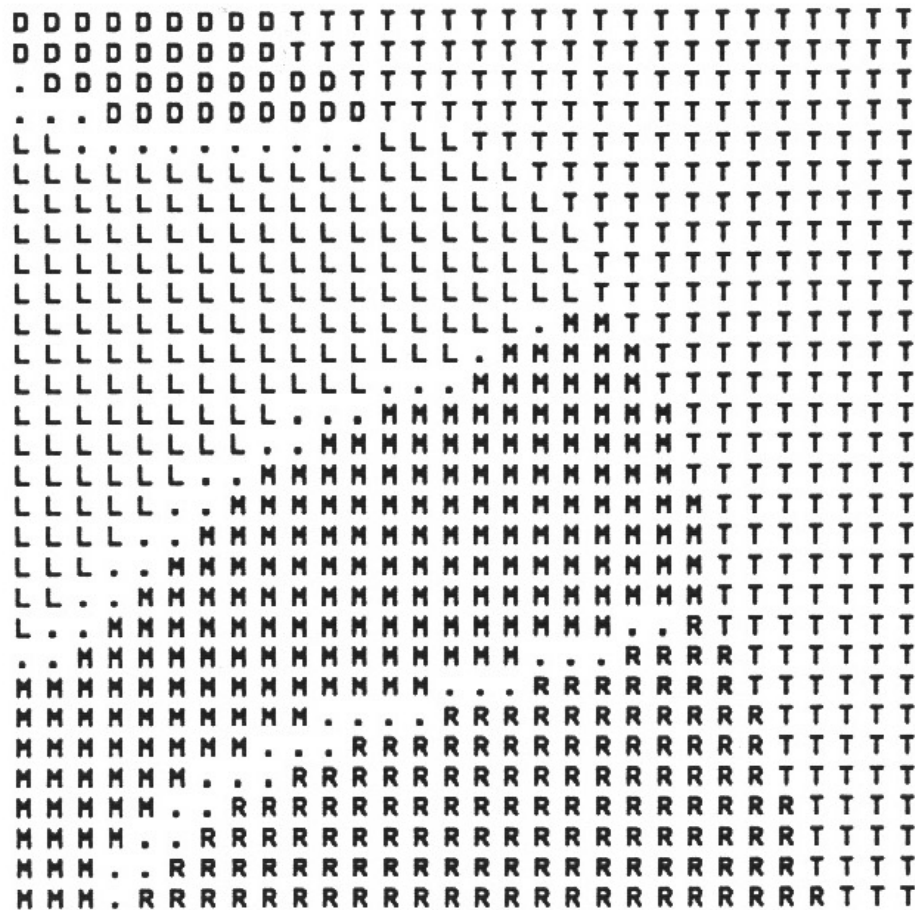


Figure 7.4 After a total of 20,000 touch stimuli, the connections between the neurons and the hand regions have become completely ordered. This assignment is now neighborhood preserving, and it reproduces the correct topological arrangement of the regions D,L,M,R and T. The map created here is also in good qualitative agreement with maps experimentally found in the cortex.

Example:

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992).

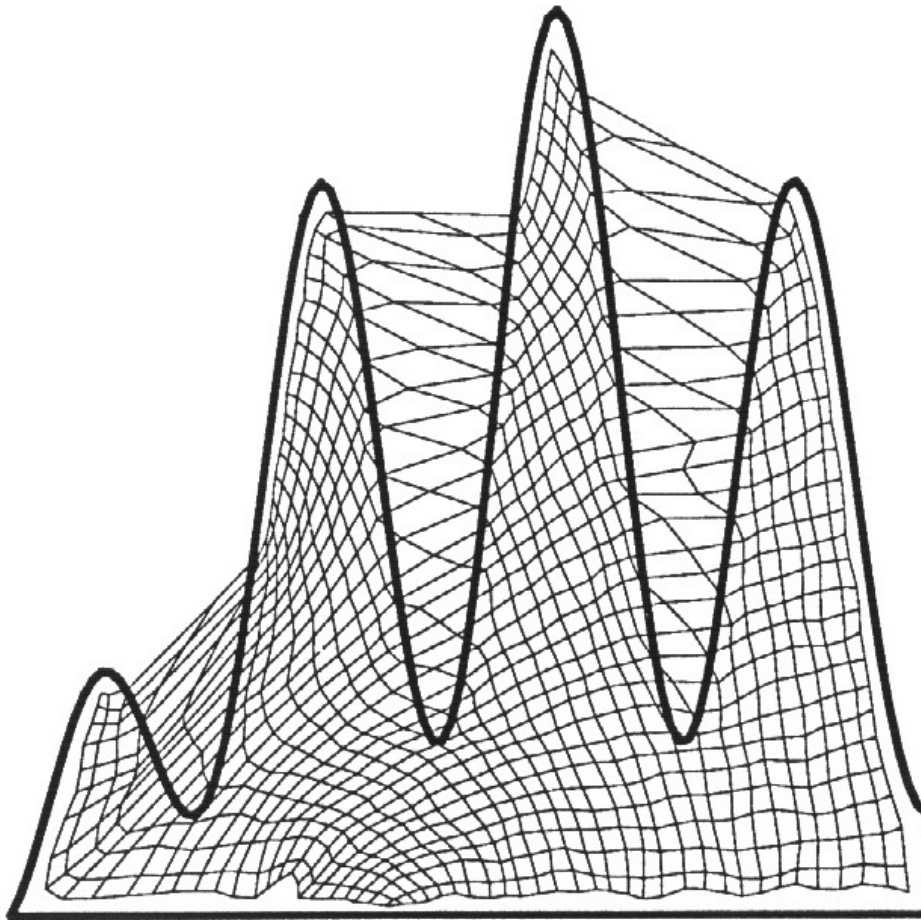


Figure 7.5 Here, the assignment of Fig. 7.4 is represented as the familiar “imbedding” of the neuron lattice in the space V , i.e., on the hand surface. To this end, each neuron is marked at the position of the center of gravity \tilde{w}_r of the touch receptors from which it receives input, and the resulting locations are connected by lines if the neurons are adjacent in the lattice.

Example:

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992).

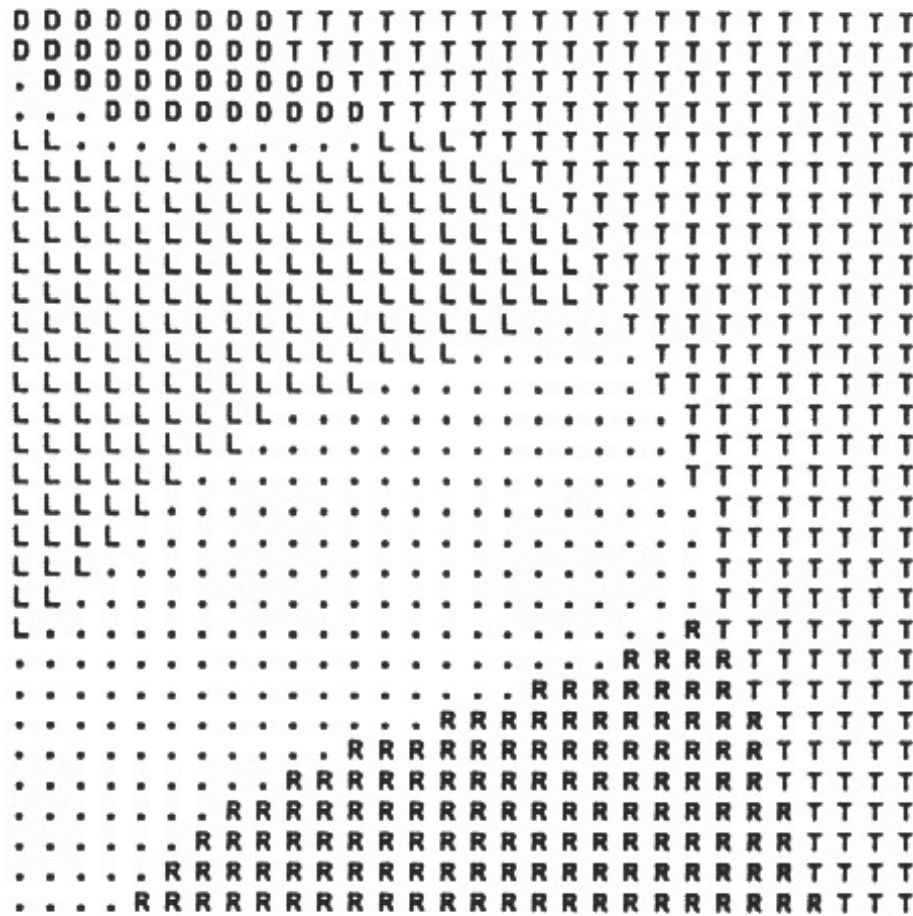


Figure 7.6 After the map of Fig. 7.4 has been obtained, the middle finger M is “amputated”, i.e., for the rest of the simulation touch stimuli are no longer selected from this region; consequently the neurons designated by dots, which were previously connected to M, are now deprived of their former sensory inputs.

Example:

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992).

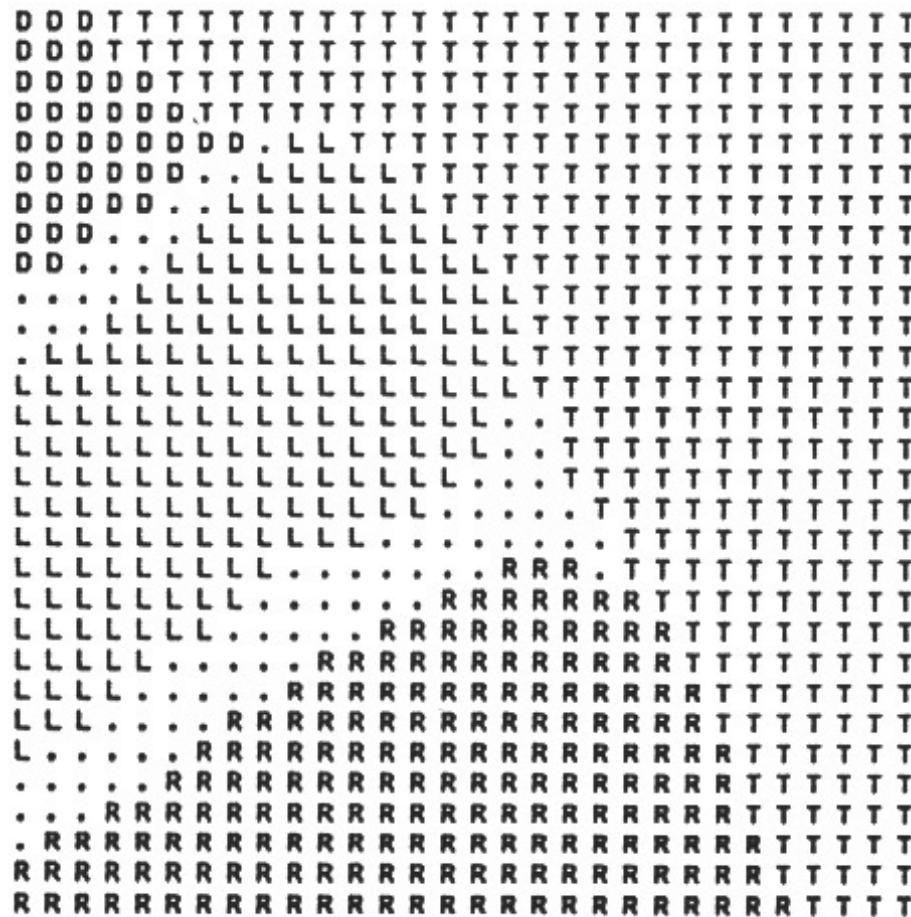


Figure 7.7 After another 50,000 touch stimuli, the map has reorganized itself in such a way that only a few neurons are still silent. The representation of the remaining hand regions D, L, R and T is now distributed over a larger number of neurons and, therefore, possesses a higher spatial resolution than before the "amputation."

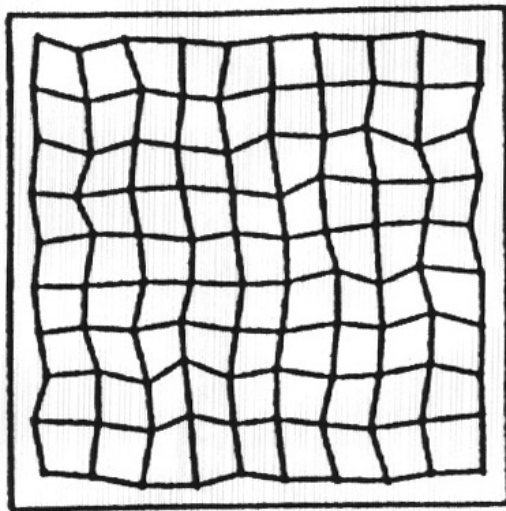
Example:

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992).

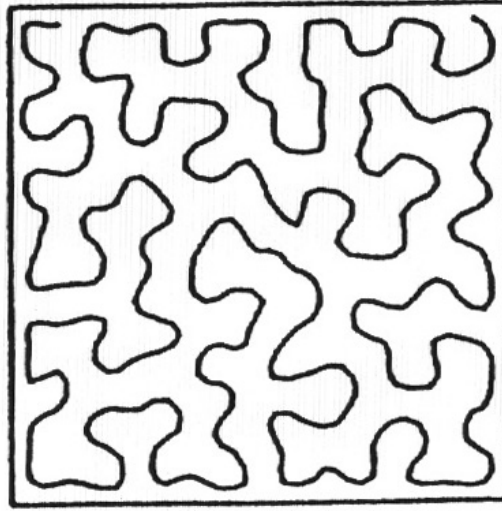


Figure 7.8 Readaptation of the somatosensory map of the hand region of an adult nocturnal ape due to the amputation of one finger. (a) (left) Before the operation, each finger in the map is represented as one of the regions 1-5. (b) (right) Several weeks after amputation of the middle finger, the assigned region 3 has disappeared, and the adjacent regions have correspondingly spread out (after Fox, 1984).

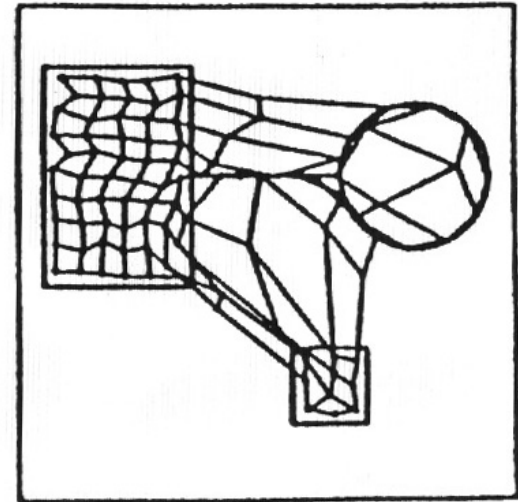
Representing Topology with the Kohonen SOM



(a)



(b)

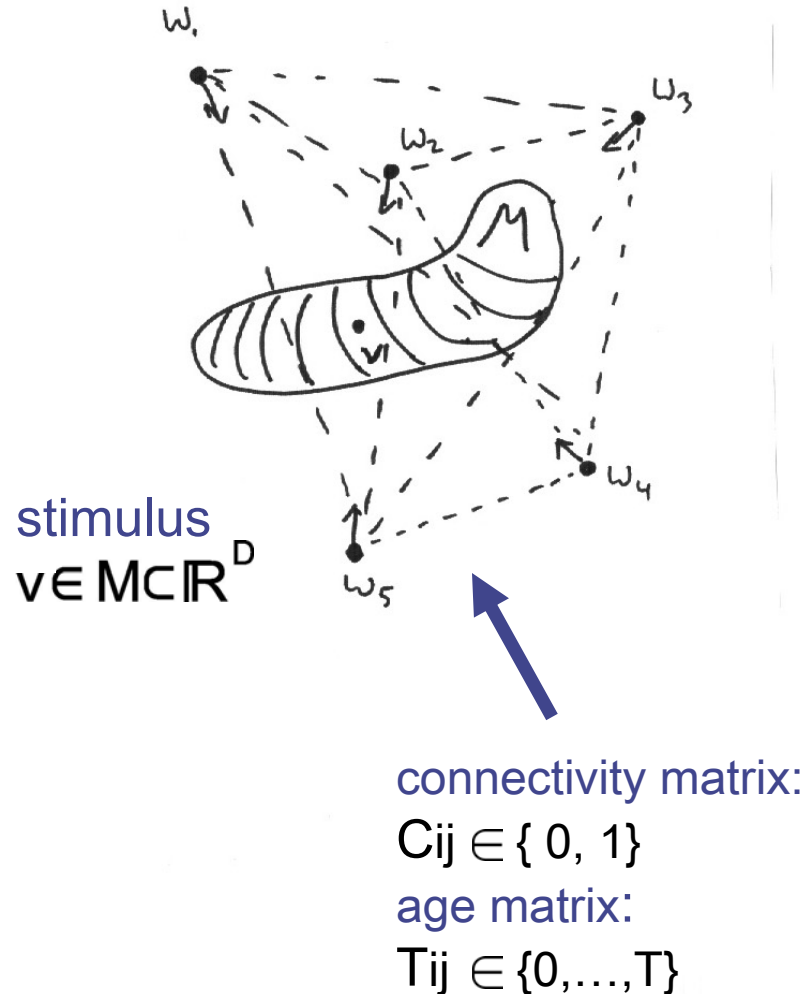


(c)

- free neurons from lattice...
- stimulus-dependent connectivities

The “Neural Gas” Algorithm

(Martinetz & Schulten, 1992)



- (i) assign initial values to the pointers $w_i \in \mathbb{R}^D$, $i = 1, \dots, N$ and set all connection strengths C_{ij} to zero;
- (ii) select an input pattern $v \in M$ with equal probability for each v ;
- (iii) for each unit i determine the number k_i of units j with

$$\|v - w_j\| < \|v - w_i\|$$

by, for example, determining the sequence $(i_0, i_1, \dots, i_{N-1})$ with

$$\|v - w_{i_0}\| < \|v - w_{i_1}\| < \dots < \|v - w_{i_{N-1}}\|;$$

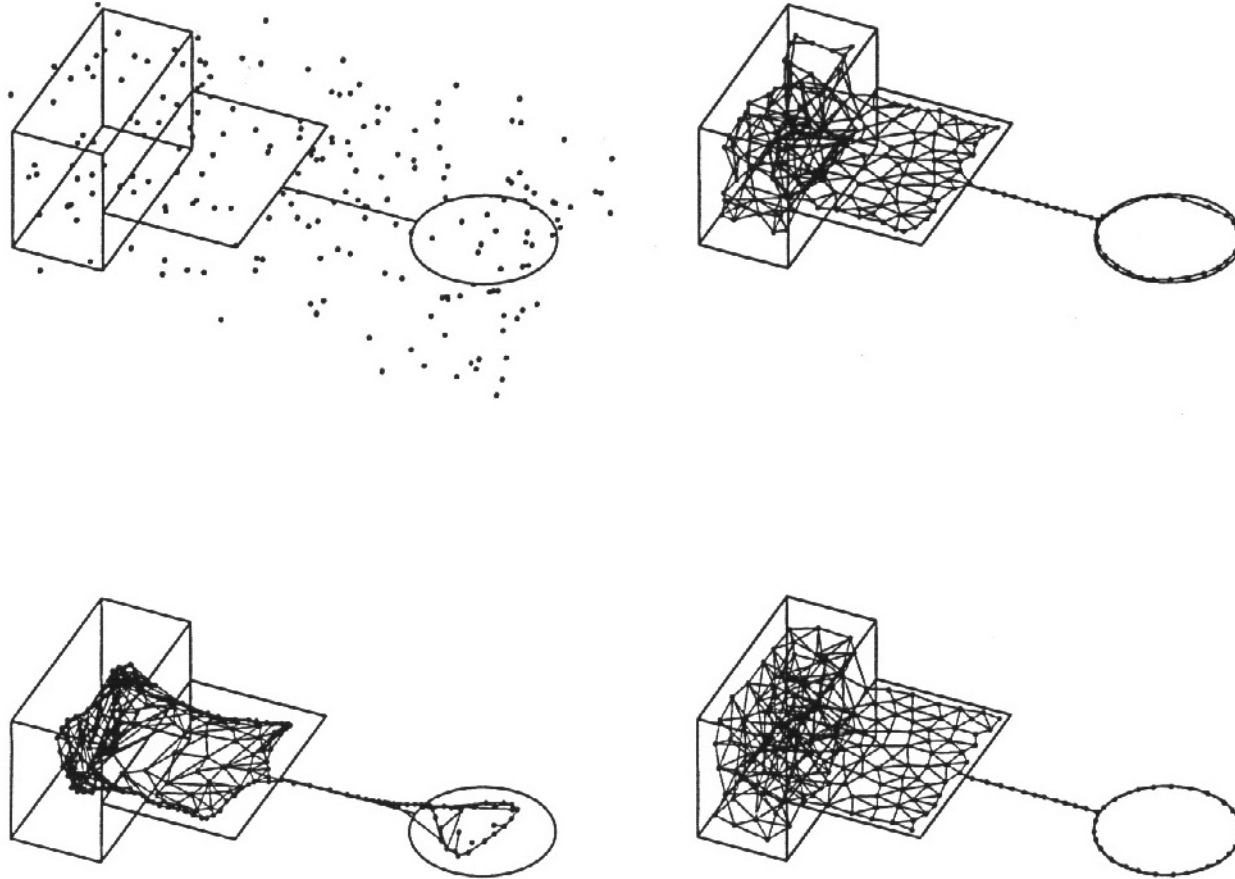
- (iv) perform an adaptation step of the pointers w_i according to the neural gas algorithm by setting

$$w_i^{\text{new}} = w_i^{\text{old}} + \varepsilon \cdot e^{-k_i/\lambda} (v - w_i^{\text{old}}), \quad i = 1, \dots, N;$$

- (v) if $C_{i_0 i_1} = 0$, set $C_{i_0 i_1} > 0$ and $t_{i_0 i_1} = 0$, that is, connect i_0 and i_1 . If $C_{i_0 i_1} > 0$, set $t_{i_0 i_1} = 0$, that is, refresh connection $i_0 - i_1$;
- (vi) increase the age of all connections of i_0 by setting $t_{i_0 j} = t_{i_0 j} + 1$ for all j with $C_{i_0 j} > 0$;
- (vii) remove those connections of unit i_0 the age of which exceeds T by setting $C_{i_0 j} = 0$ for all j with $C_{i_0 j} > 0$ and $t_{i_0 j} > T$; continue with (ii).

$$\varepsilon = \varepsilon(t), \quad \lambda = \lambda(t), \quad T = T(t)$$

Example



Example (cont.)

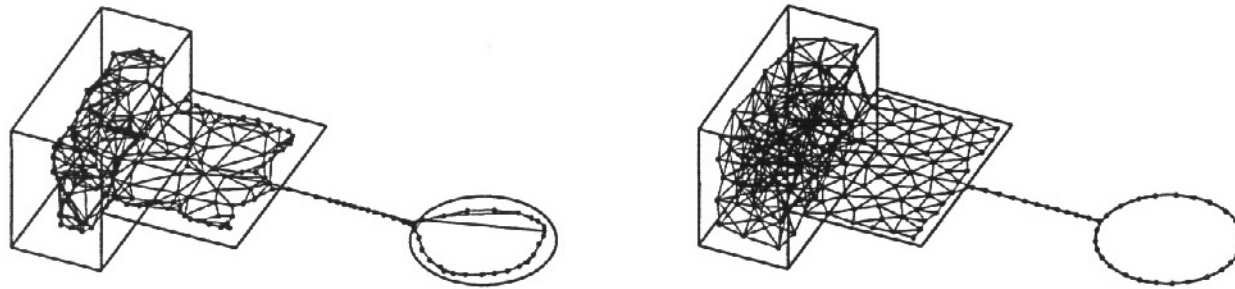
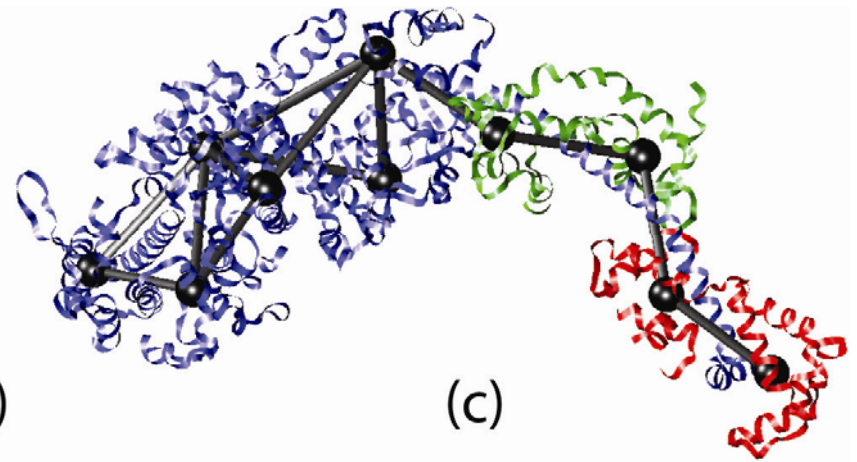
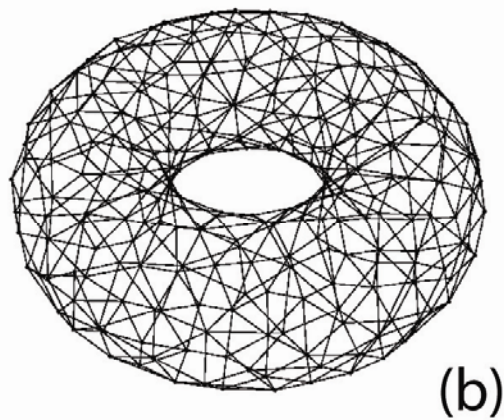
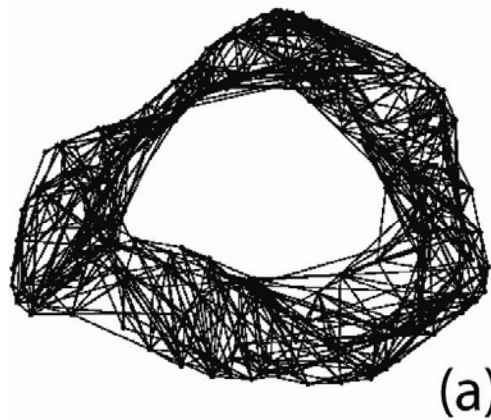


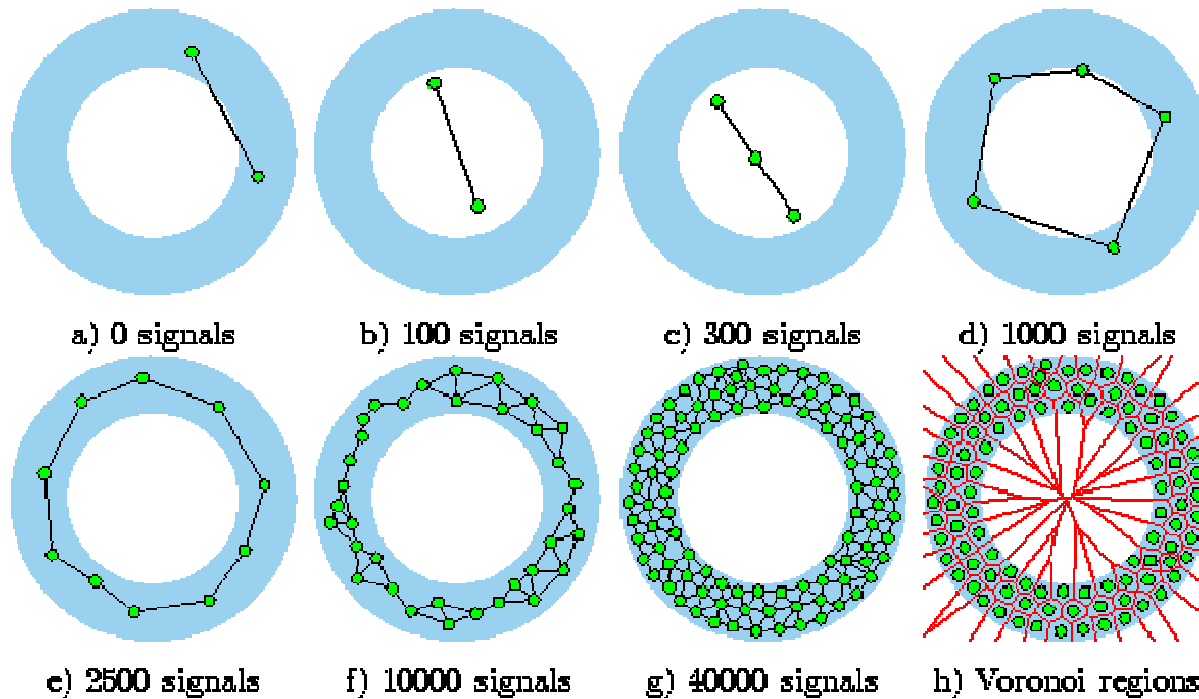
Fig.2: The “neural gas” network quantizing a topologically heterogeneously structured input data manifold. The data manifold consists of a three-dimensional (right parallelepiped), a two-dimensional (rectangle), and a one-dimensional (circle and connecting line) subset. The dots mark the centers of the receptive fields M_i determined by the formal synaptic weights w_i . Connections between neural units i, j , i.e., $C_{ij} = 1$, are indicated by connecting lines between the locations w_i and w_j . Depicted are the initial state, the network after 5 000, 10 000, 15 000, 25 000, and at the final state after 40 000 adaptation steps (from top left to bottom right) At the end of the adaptation procedure the connections between the neural units reflect the topological structure and the corresponding dimensionality of the data manifold.

More Examples: Torus and Myosin S1



Growing Neural Gas

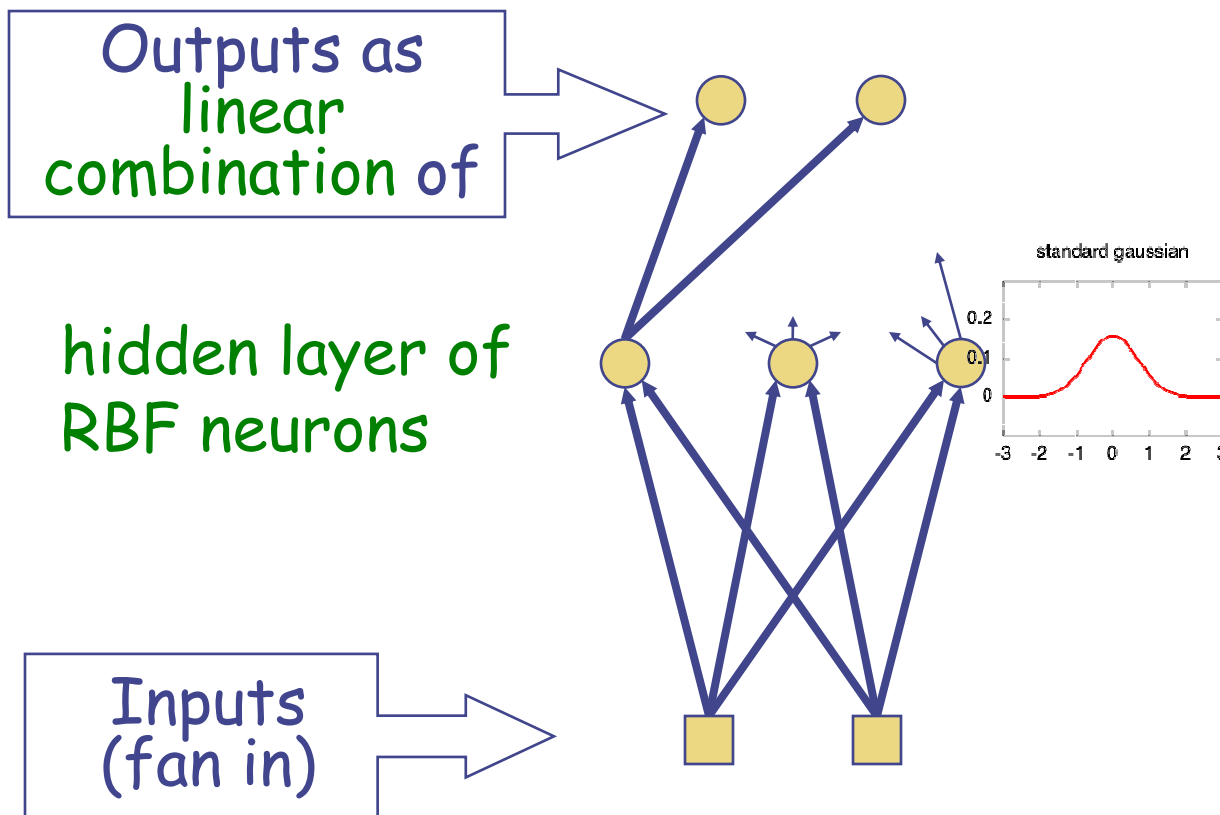
GNG = Neural gas &
dynamical creation/removal of links



Why use GNG ?

- Adaptability to Data Topology
 - Both dynamically and spatially
- Data Analysis
- Data Visualization

Radial Basis Function Networks



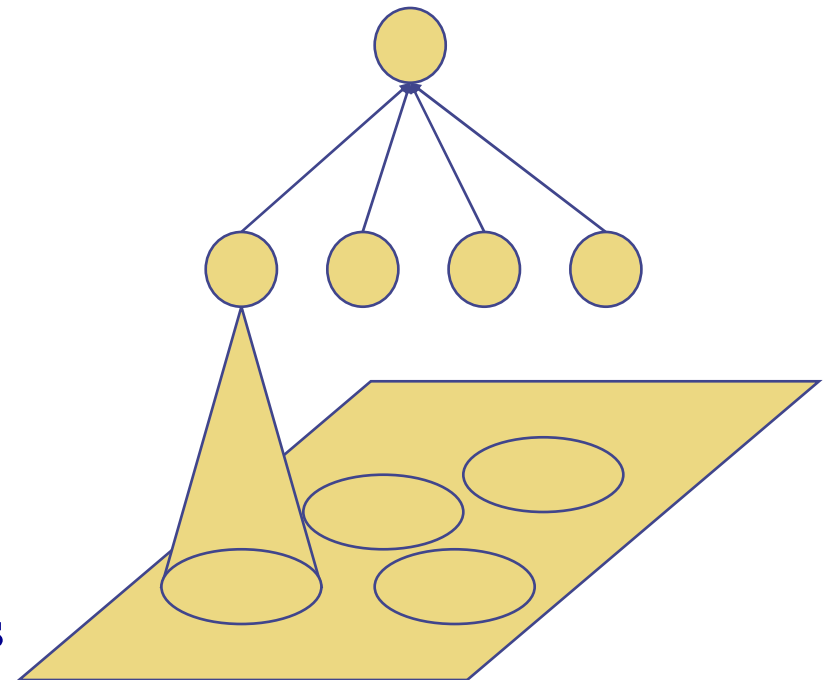
Usually apply a
unsupervised learning
procedure

•Set number of neurons
and then adjust :

- 1.Gaussian centers
- 2.Gaussian widths
- 3.weights

Why use RBF ?

- Density estimation
- Discrimination
- Regression
- Good to know:
 - Can be described as Bayesian Networks
 - Close to some Fuzzy Systems



Demo

Internet Java demo <http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/DemoGNG/GNG.html>

- Hebb Rule
- LBG / k -means
- Neural Gas
- GNG
- Kohonen SOM

Revisiting Quantization

Vector Quantization

Lloyd (1957) } Digital Signal Processing,
Linde, Buzo, & Gray (1980) } Speech and Image Compression.
Martinetz & Schulten (1993) } Neural Gas.

Encode data (in \mathfrak{R}^D) using a finite set $\{w_j\}$ ($j=1,\dots,k$) of *codebook vectors*.
Delaunay triangulation divides \mathfrak{R}^D into k *Voronoi polyhedra* (“receptive fields”):

$$V_i = \left\{ v \in \mathfrak{R}^D \mid \|v - w_i\| \leq \|v - w_j\| \forall j \right\}$$

Vector Quantization

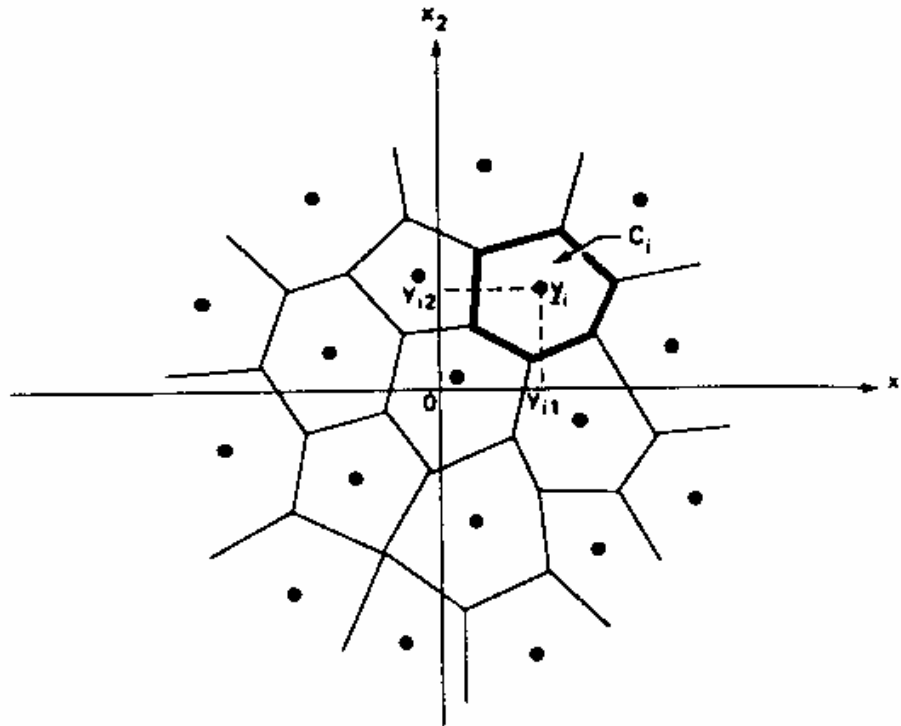
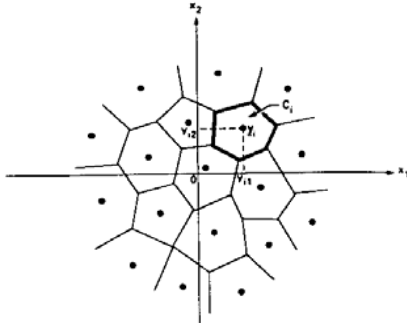


Fig. 3. Partitioning of two-dimensional space ($N = 2$) into $L = 18$ cells. All input vectors in cell C_i will be quantized as the code vector y_i . The shapes of the various cells can be very different.

k-Means a.k.a. Linde, Buzo & Gray (LBG)



Encoding Distortion Error:

$$E = \sum_{i \text{ (data points)}} \left\| v_i - w_{j(i)} \right\|^2 d_i$$

Lower $E(\{w_j(t)\})$ iteratively: Gradient descent $\forall r$:

$$\Delta w_r(t) \equiv w_r(t) - w_r(t-1) = -\frac{\varepsilon}{2} \cdot \frac{\partial E}{\partial w_r} = \varepsilon \cdot \sum_i \delta_{rj(i)} (v_i - w_r) d_i .$$

Inline (Monte Carlo) approach for a sequence $v_i(t)$ selected at random according to probability density function d_i :

$$\Delta w_r(t) = \tilde{\varepsilon} \cdot \delta_{rj(i)} \cdot (v_i(t) - w_r).$$

Advantage: fast, reasonable clustering.

Limitations: depends on initial random positions,
difficult to avoid getting trapped in the many local minima of E

Neural Gas Revisited

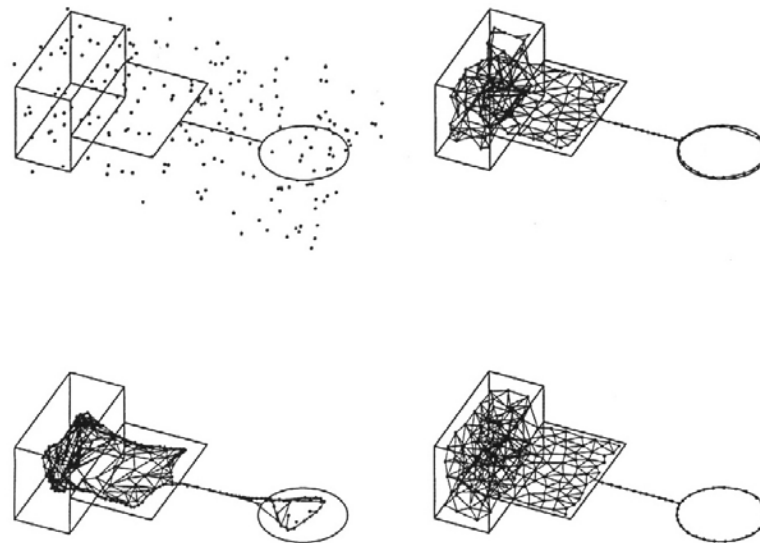
Avoid local minima traps of k -means by smoothing of energy function:

$$\forall r: \Delta w_r(t) = \tilde{\varepsilon} \cdot e^{\frac{-s_r}{\lambda}} \cdot (v_i(t) - w_r),$$

Where $s_r(v_i(t), \{w_j\})$ is the closeness rank:

$$\|v_i - w_{j_0}\| \leq \|v_i - w_{j_1}\| \leq \dots \leq \|v_i - w_{j_{(k-1)}}\|$$

$$s_r = 0 \quad s_r = 1 \quad s_r = k - 1$$



Neural Gas Revisited

Note: $\lambda \rightarrow 0$: k -means.

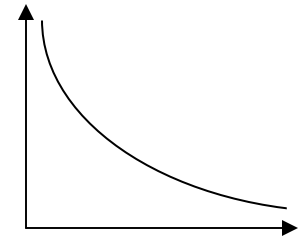
$\lambda \neq 0$: not only “winner” $w_{j(i)}$ also second, third, ... closest are updated.

Can show that this corresponds to stochastic gradient descent on

$$\tilde{E}(\{w_j\}, \lambda) = \sum_{r=1}^k e^{\frac{-s_r}{\lambda}} \sum_i \left\| v_i - w_{j(i)} \right\|^2 d_i .$$

Note: $\lambda \rightarrow 0$: $\tilde{E} \rightarrow E$. k -means.

$\lambda \rightarrow \infty$: \tilde{E} parabolic (single minimum). } $\Rightarrow \lambda(t)$



Neural Gas Revisited

Q: How do we know that we have found the global minimum of E ?

A: We don't (in general).

But we can compute the statistical variability of the $\{w_j\}$ by repeating the calculation with different seeds for random number generator.

Codebook vector variability arises due to:

- statistical uncertainty,
- spread of local minima.

A small variability indicates good convergence behavior.

Optimum choice of # of vectors k : variability is minimal.

Pattern Recognition

Pattern Recognition

Definition: “The assignment of a physical object or event to one of several prespecified categories” -- Duda & Hart

- A **pattern** is an object, process or event that can be given a name.
- A **pattern class** (or category) is a set of patterns sharing common attributes and usually originating from the same source.
- During **recognition** (or **classification**) given objects are assigned to prescribed classes.
- A **classifier** is a machine which performs classification.

PR Applications

- **Optical Character Recognition (OCR)**

- Handwritten: sorting letters by postal code, input device for PDA's.
- Printed texts: reading machines for blind people, digitalization of text documents.

- **Biometrics**

- Face recognition, verification, retrieval.
- Finger prints recognition.
- Speech recognition.

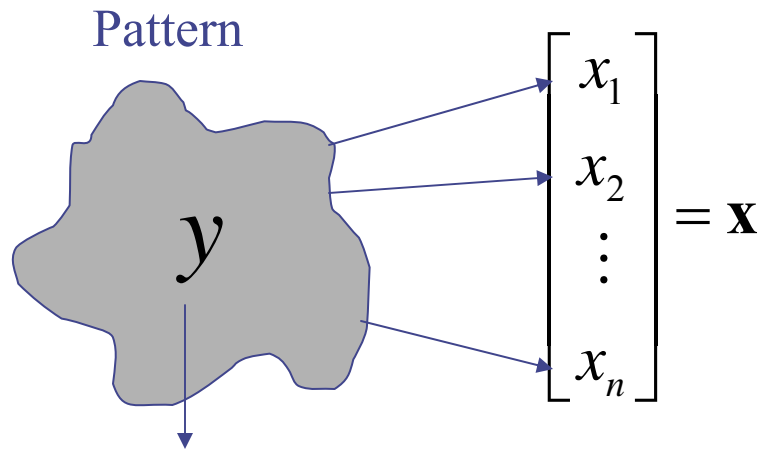
- **Diagnostic systems**

- Medical diagnosis: X-Ray, EKG analysis.
- Machine diagnostics, waster detection.

Approaches

- **Statistical PR:** based on underlying statistical model of patterns and pattern classes.
- **Structural (or syntactic) PR:** pattern classes represented by means of formal structures as grammars, automata, strings, etc.
- **Neural networks:** classifier is represented as a network of cells modeling neurons of the human brain (connectionist approach).

Basic Concepts



Feature vector $\mathbf{x} \in X$

- A vector of observations (measurements).
- \mathbf{x} is a point in feature space X .

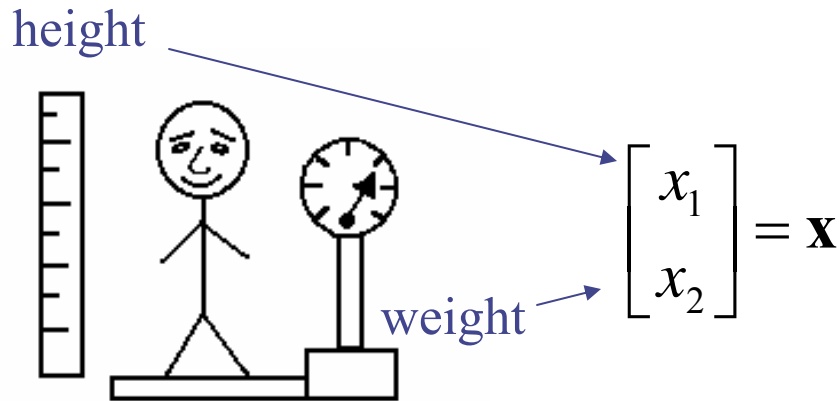
Hidden state $y \in Y$

- Cannot be directly measured.
- Patterns with equal hidden state belong to the same class.

Task

- To design a classifier (decision rule) $q: X \rightarrow Y$
which decides about a hidden state based on an onbservation.

Example



Task: jockey-hoopster recognition.

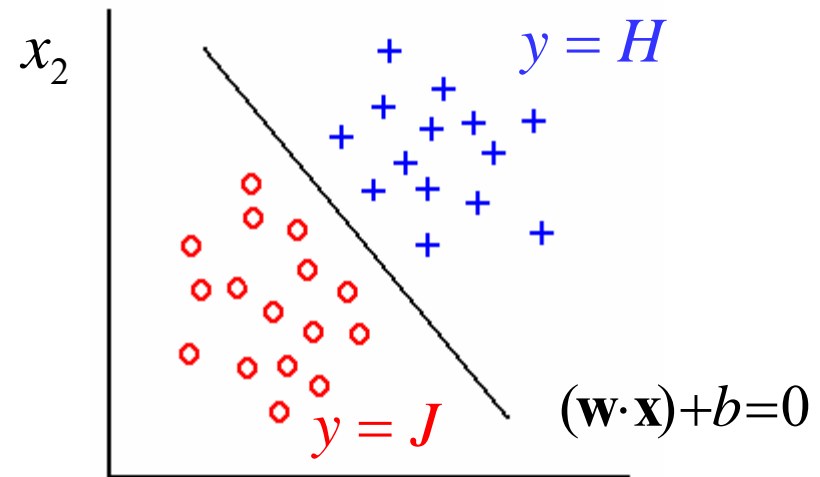
The set of hidden state is $Y = \{H, J\}$

The feature space is $X = \mathbb{R}^2$

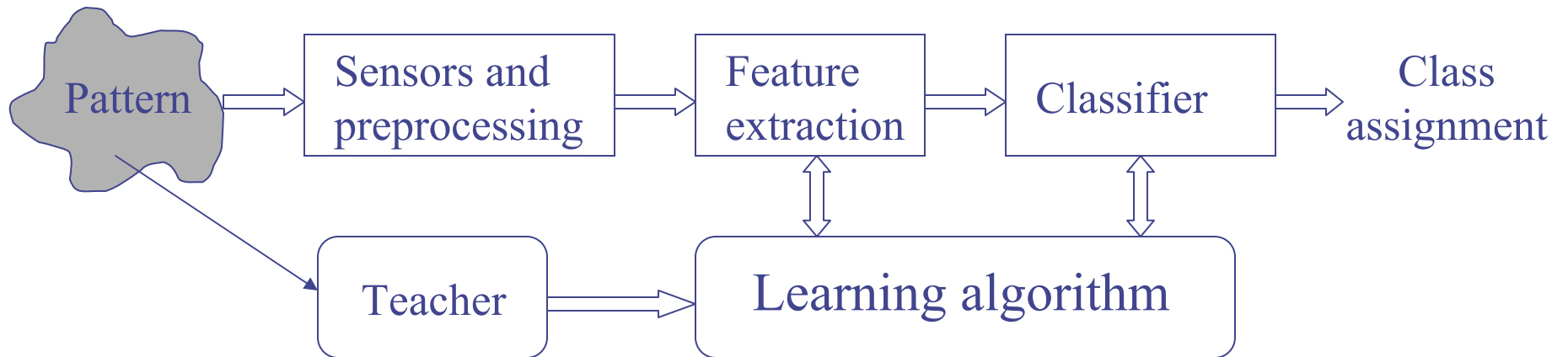
Training examples $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_l, y_l)\}$

Linear classifier:

$$q(\mathbf{x}) = \begin{cases} H & \text{if } (\mathbf{w} \cdot \mathbf{x}) + b \geq 0 \\ J & \text{if } (\mathbf{w} \cdot \mathbf{x}) + b < 0 \end{cases}$$



Components of a PR System

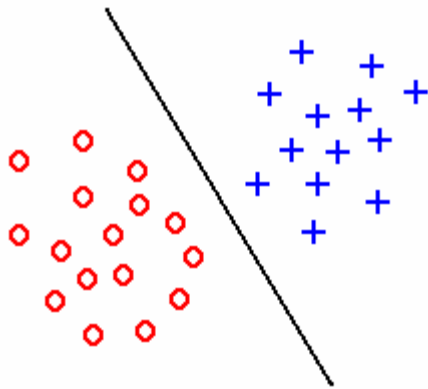


- **Sensors and preprocessing.**
- **A feature extraction** aims to create discriminative features good for classification.
- **A classifier.**
- **A teacher** provides information about hidden state -- supervised learning.
- **A learning algorithm** sets PR from training examples.

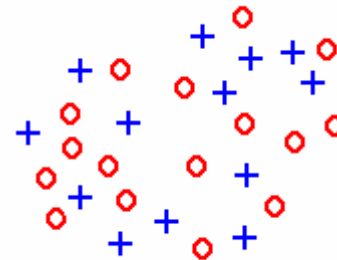
Feature Extraction

Task: to extract features which are good for classification.

- Good features:
- Objects from the same class have similar feature values.
 - Objects from different classes have different values.



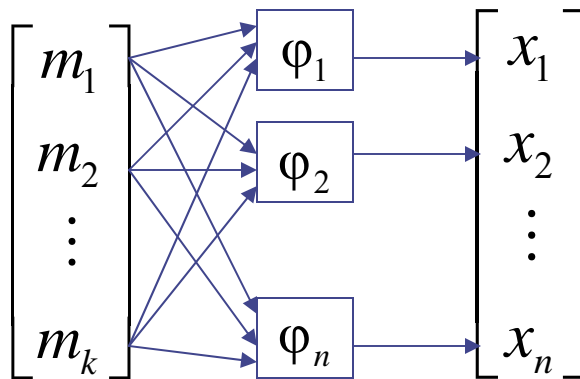
“Good” features



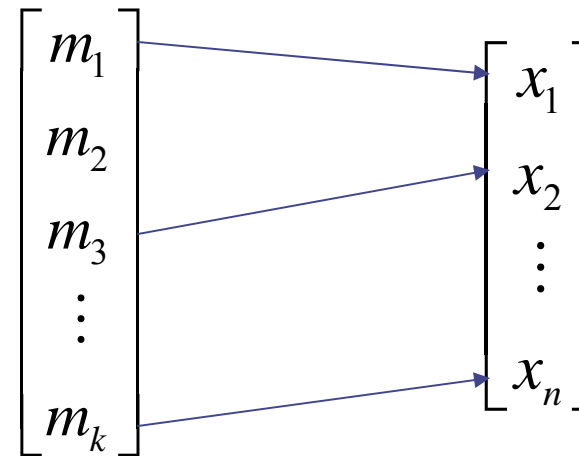
“Bad” features

Feature Extraction Methods

Feature extraction



Feature selection



Problem can be expressed as optimization of parameters of feature extractor $\varphi(\theta)$.

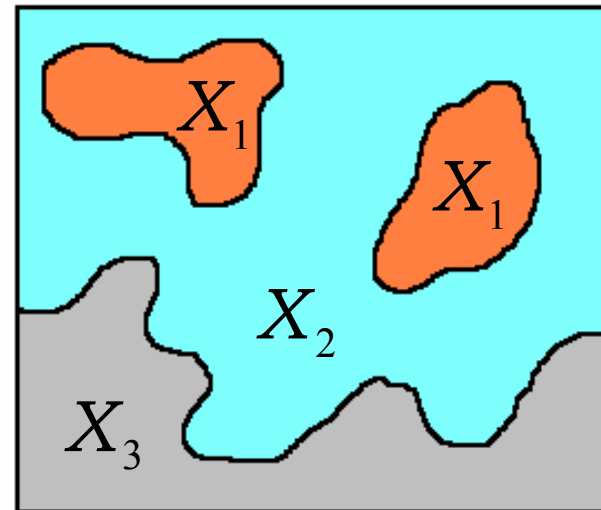
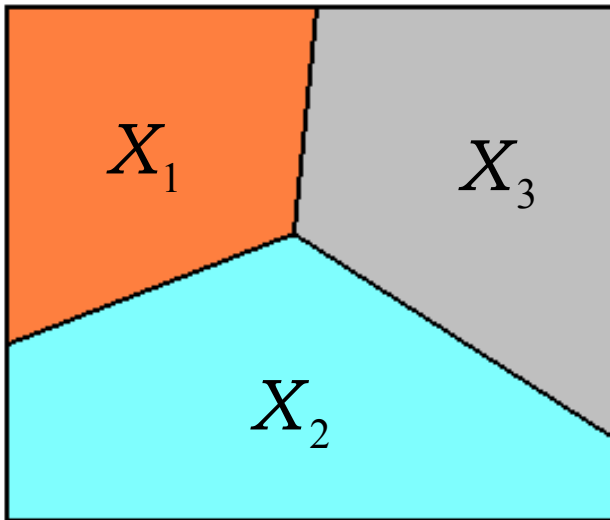
Supervised methods: objective function is a criterion of separability (discriminability) of labeled examples, e.g., linear discriminant analysis (LDA).

Unsupervised methods: lower dimensional representation which preserves important characteristics of input data is sought for, e.g., principal component analysis (PCA).

Classifier

A classifier partitions feature space X into **class-labeled regions** such that

$$X = X_1 \cup X_2 \cup \dots \cup X_{|Y|} \quad \text{and} \quad X_1 \cap X_2 \cap \dots \cap X_{|Y|} = \{0\}$$



The classification consists of determining to which region a feature vector \mathbf{x} belongs to.

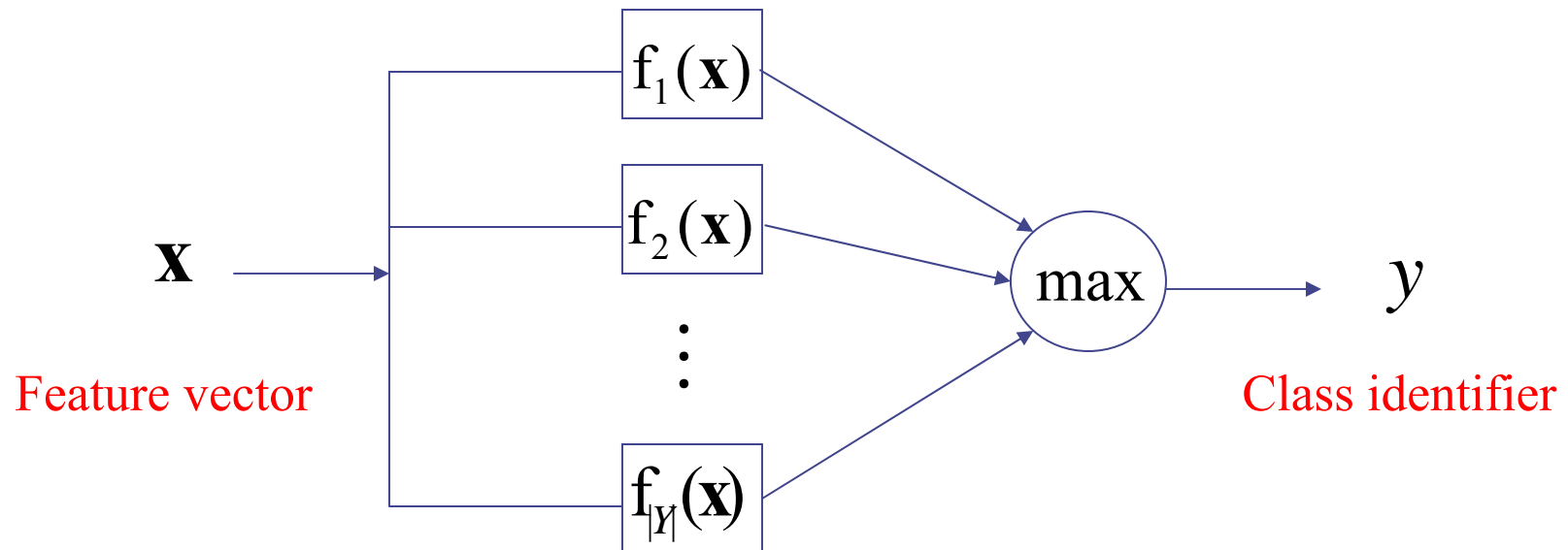
Borders between **decision boundaries** are called decision regions.

Representation of a Classifier

A classifier is typically represented as a set of discriminant functions

$$f_i(\mathbf{x}) : X \rightarrow \mathcal{R}, i = 1, \dots, |Y|$$

The classifier assigns a feature vector \mathbf{x} to the i -th class if $f_i(\mathbf{x}) > f_j(\mathbf{x}) \quad \forall j \neq i$



Discriminant function

Bayesian Decision Making

- The Bayesian decision making is a fundamental statistical approach which allows to design the optimal classifier if complete **statistical model is known**.

<u>Definition:</u>	Observations	X	A loss function	$W : Y \times D \rightarrow R$
	Hidden states	Y	A decision rule	$q : X \rightarrow D$
	Decisions	D	A joint probability	$p(\mathbf{x}, y)$

Task: to design decision rule q which minimizes Bayesian risk

$$R(q) = \sum_{y \in Y} \sum_{x \in X} p(\mathbf{x}, y) W(q(\mathbf{x}), y)$$

Example of a Bayesian Task

Task: minimization of classification error.

A set of decisions D is the same as set of hidden states Y .

0/1 - loss function used
$$W(q(\mathbf{x}), y) = \begin{cases} 0 & \text{if } q(\mathbf{x}) = y \\ 1 & \text{if } q(\mathbf{x}) \neq y \end{cases}$$

The Bayesian risk $R(q)$ corresponds to probability of misclassification.

The solution of Bayesian task is

$$q^* = \arg \min_q R(q) \Rightarrow y^* = \arg \max_y p(y | \mathbf{x}) = \arg \max_y \frac{p(\mathbf{x} | y) p(y)}{p(\mathbf{x})}$$

Limitations of the Bayesian Approach

- The statistical model $p(\mathbf{x}, y)$ is mostly not known therefore learning must be employed to estimate $\bar{p}(\mathbf{x}, y)$ from training examples $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell)\}$ -- **plug-in Bayes**.
- **Non-Bayesian methods** offers further task formulations:
 - A partial statistical model is available only:
 - $p(y)$ is not known or does not exist.
 - $p(\mathbf{x}|y, \theta)$ is influenced by a non-random intervention θ .
 - The loss function is not defined.
 - Examples: Neyman-Pearson's task, Minimax task, etc.

Discriminative Approaches

Given a class of classification rules $q(\mathbf{x};\boldsymbol{\theta})$ parametrized by $\boldsymbol{\theta} \in \Xi$ the task is to find the “best” parameter $\boldsymbol{\theta}^*$ based on a set of training examples $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell)\}$ -- **supervised learning**.

The **task of learning**: recognition which classification rule is to be used.

The way how to perform the learning is determined by a selected **inductive principle**.

Empirical Risk Minimization Principle

The true expected risk $R(q)$ is approximated by **empirical risk**

$$R_{\text{emp}}(q(x; \boldsymbol{\theta})) = \frac{1}{\ell} \sum_{i=1}^{\ell} W(q(\mathbf{x}_i; \boldsymbol{\theta}), y_i)$$

with respect to a given labeled training set $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell)\}$.

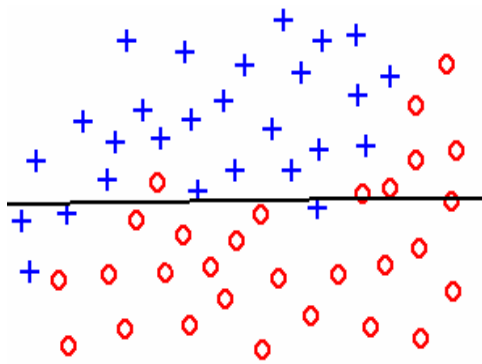
The learning based on the **empirical minimization principle** is defined as

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} R_{\text{emp}}(q(\mathbf{x}; \boldsymbol{\theta}))$$

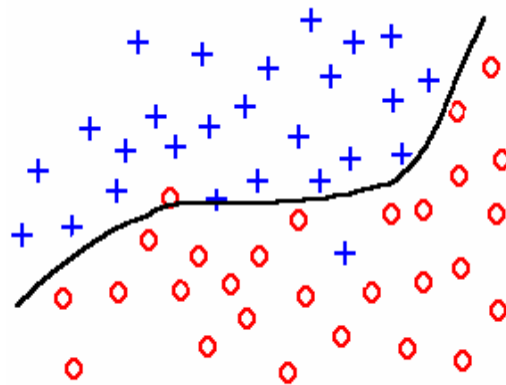
Examples of algorithms: **Perceptron, Back-propagation, etc.**

Overfitting and Underfitting

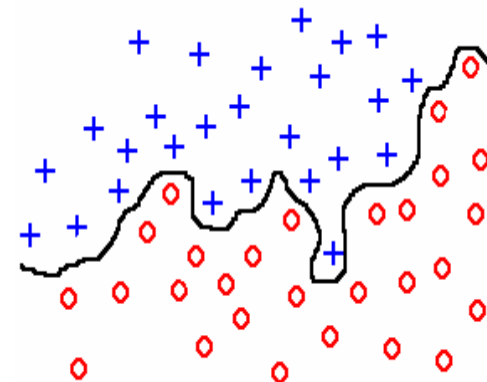
Problem: how rich class of classifications $q(\mathbf{x};\theta)$ to use.



underfitting



good fit



overfitting

Problem of generalization: a small empirical risk R_{emp} does not imply small true expected risk R .

Structural Risk Minimization Principle

Statistical learning theory -- Vapnik & Chervonenkis.

An upper bound on the expected risk of a classification rule $q \in Q$

$$R(q) \leq R_{emp}(q) + R_{str}\left(\frac{1}{\ell}, h, \log \frac{1}{\sigma}\right)$$

where ℓ is number of training examples, h is VC-dimension of class of functions Q and $1-\sigma$ is confidence of the upper bound.

SRM principle: from a given nested function classes Q_1, Q_2, \dots, Q_m , such that

$$h_1 \leq h_2 \leq \dots \leq h_m$$

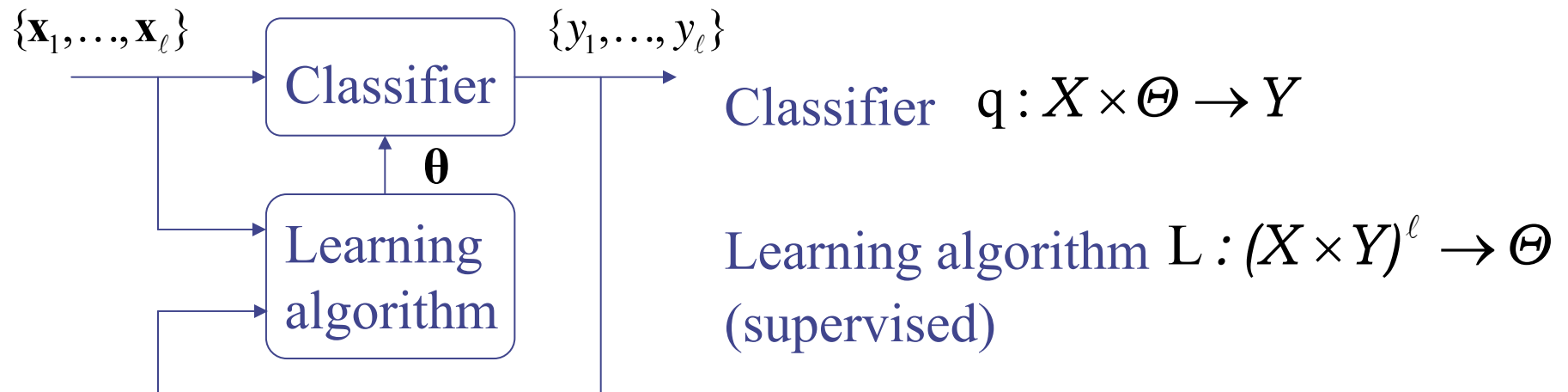
select a rule q^* which minimizes the upper bound on the expected risk.

Unsupervised Learning

Input: training examples $\{\mathbf{x}_1, \dots, \mathbf{x}_\ell\}$ without information about the hidden state.

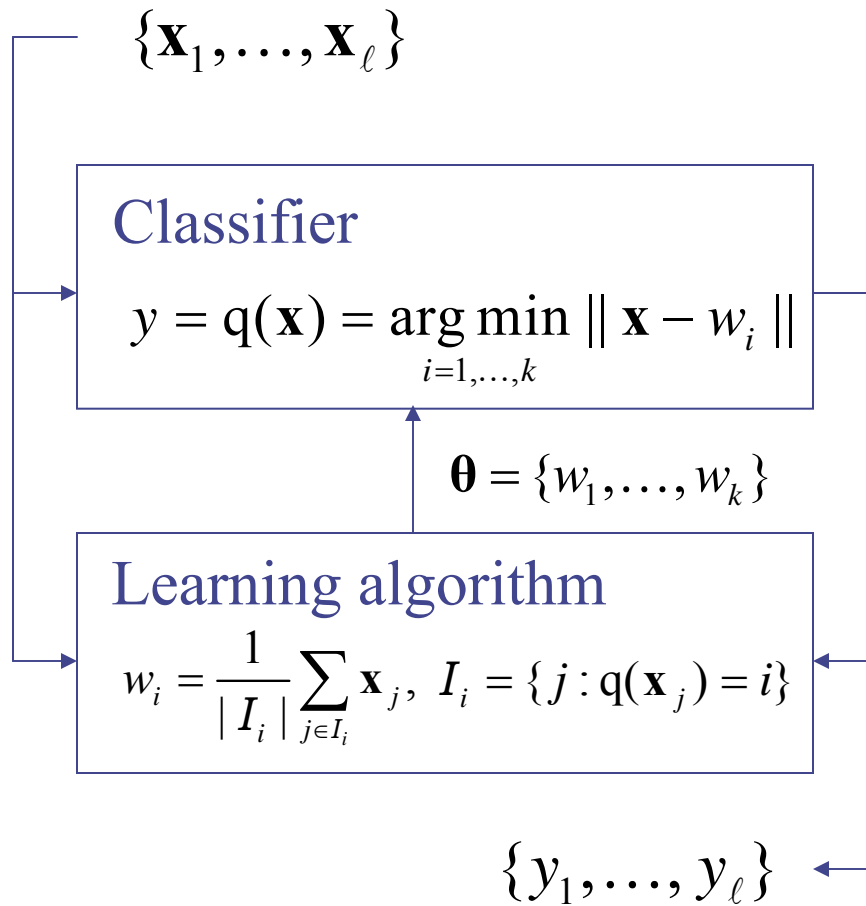
Clustering: goal is to find clusters of data sharing similar properties.

A broad class of **unsupervised learning algorithms**:



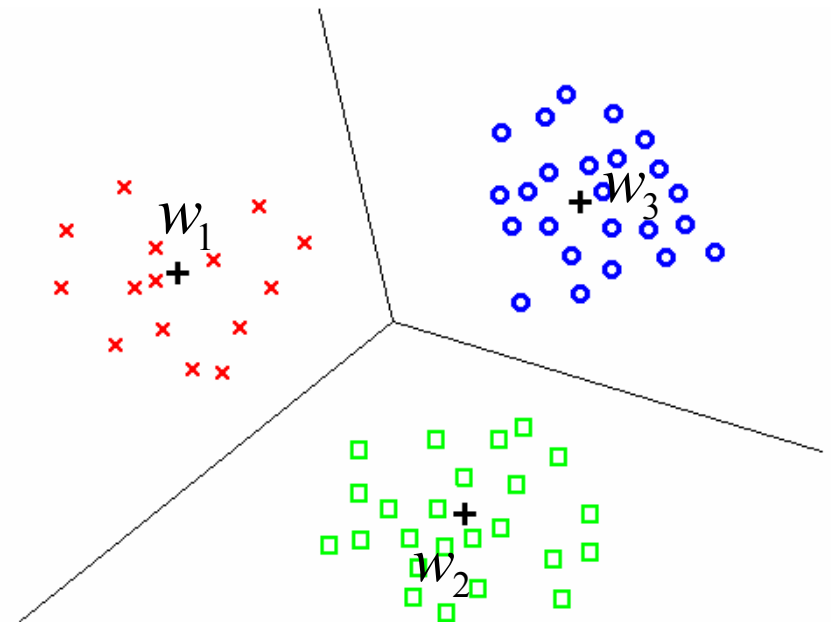
Example

k-Means Clustering:



Goal is to minimize

$$\sum_{i=1}^{\ell} \|\mathbf{x}_i - w_{q(\mathbf{x}_i)}\|^2$$



Neural Network References

- *Neural Networks, a Comprehensive Foundation*, S. Haykin, ed. Prentice Hall (1999)
- *Neural Networks for Pattern Recognition*, C. M. Bishop, ed Clarendon Press, Oxford (1997)
- *Self Organizing Maps*, T. Kohonen, Springer (2001)

Some ANN Toolboxes

- Free software
 - SNNS: Stuttgarter Neural Network Systems & Java NNS
 - GNG at Uni Bochum
- Matlab toolboxes
 - Fuzzy Logic
 - Artificial Neural Networks
 - Signal Processing

Pattern Recognition / Vector Quantization References

Textbooks

Duda, Hart: Pattern Classification and Scene Analysis. J. Wiley & Sons, New York, 1982. (2nd edition 2000).

Fukunaga: Introduction to Statistical Pattern Recognition. Academic Press, 1990.

Bishop: Neural Networks for Pattern Recognition. Clarendon Press, Oxford, 1997.

Schlesinger, Hlaváč: Ten lectures on statistical and structural pattern recognition. Kluwer Academic Publisher, 2002.