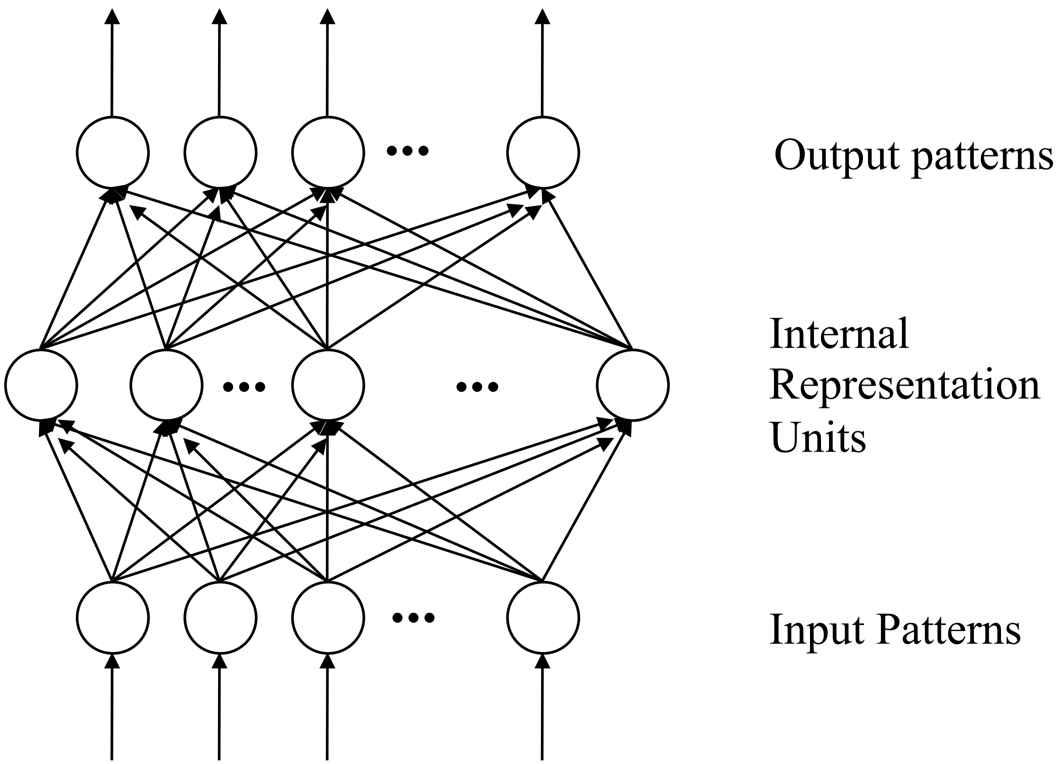
Neuronale Netze

Kohonen Maps

Joshua Winebarger
06.12.2011

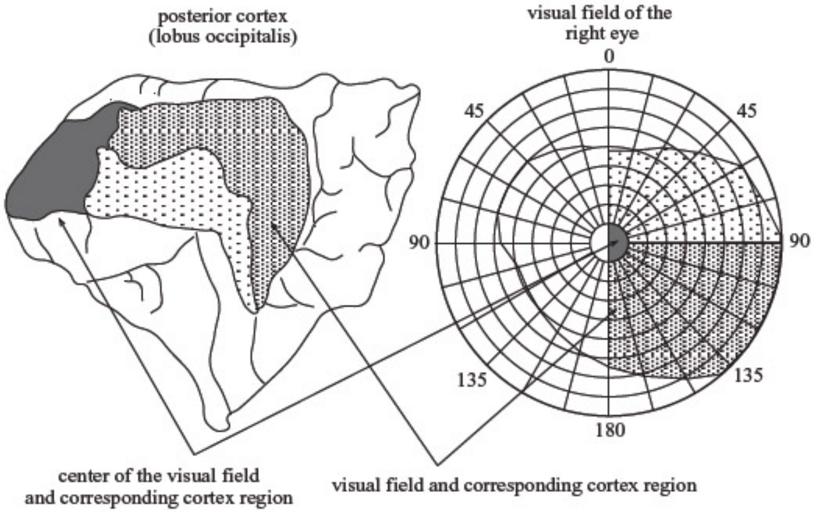
Traditional Feedforward ANNs

- Classification
- Information fed forward
- No fixed number of layers
- No relation amongst output units



Biological Systems

- Spatial organization of information
  - Topological mapping of sensory and motor phenomenon on surface of brain
  - More space dedicated to more frequent patterns
  - e.x. Mapping of visual field onto cortex



Kohonen Maps

- Self-organization
  - Unsupervised competitive learning
- Produces a low-dimensional representation of the input space
- Maps the organization of relationships among patterns in input
- Paradigm introduced by Kohonen
- Precursors appear in work of Grossberg, Rosenblatt, von der Malsburg, others

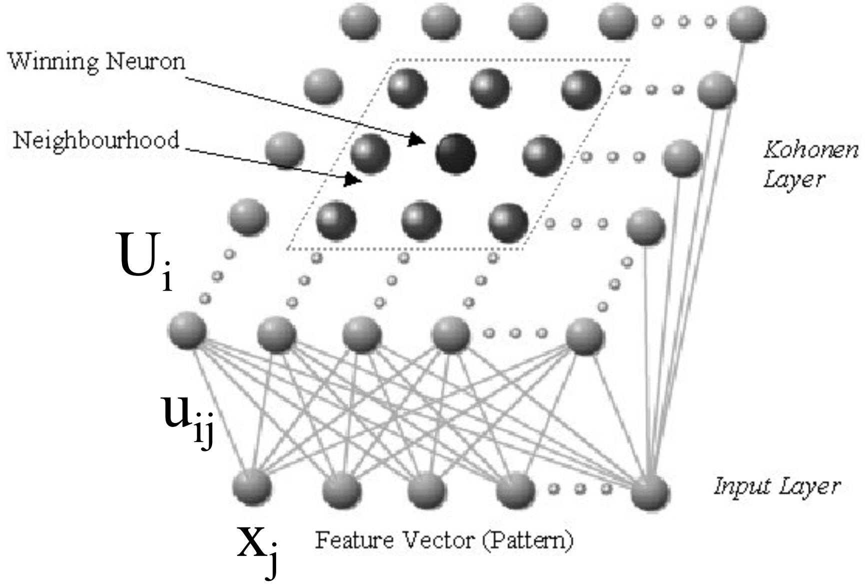
Input and Output Spaces

- Neural networks serve as function which map an input in space *A* to an output space *B*

- In a Kohonen map, those points close in *A* are also close in *B*
  - Preservation of the topological properties of the input space
- A Kohonen map is such that for a given input vector *a*, only one neuron in the network fires

Basic Structure

- Two layers
  - Input Layer
  - Competitive Layer (Kohonen Layer)
- Each input unit is connected to all units in the competitive layer
- Kohonen Maps operate in two modes: training and mapping



Training

- Weights $U_i=[\, u_{i1},\ldots,u_{in}\,]$
- Input $X=[\, x_1,\ldots,x_n\,]$

0. Begin iteration *t*

1. *Initialization* - Assign weights randomly

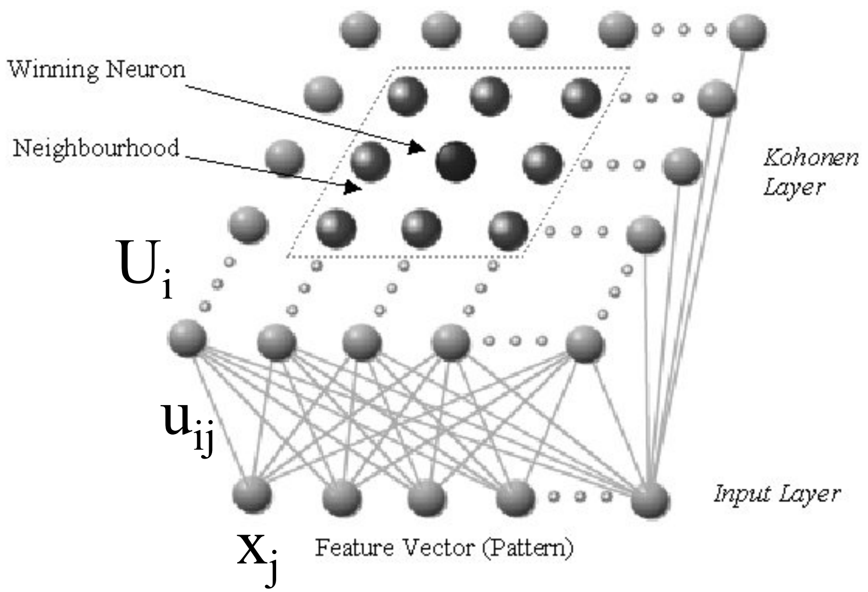
2. *Sampling* – Draw input X^{μ} from input space

3. *Matching* - Find winning neuron
   - Compute matching value for each unit

$$\left\| X^{\mu} - U_i \right\| = \sqrt{\sum_j \left(x_j^{\mu} - u_{ij} \right)^2}$$

   - Unit *c* w/ lowest matching value wins (BMU)

$$c(\mu) = \operatorname*{argmin}_i \left\{ \left\| X^{\mu} - U_i \right\| \right\}$$



Training pt. II

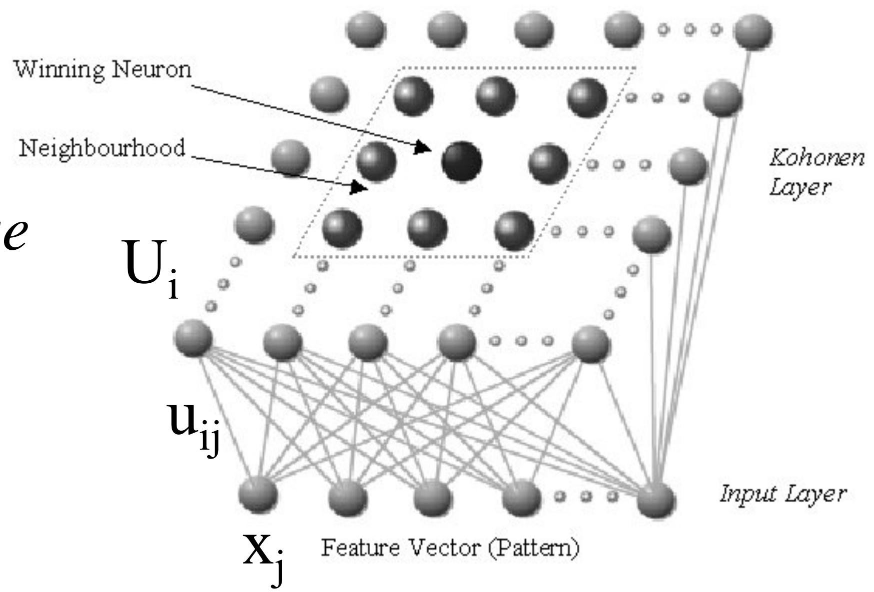
4. *Updating*
    a. Identify neighborhood N_c around unit *c* within distance *d*
    b. Update weights in N_c

$$\Delta u_{ij} = \begin{cases} \alpha(t)\left(x_j^{\mu} - u_{ij}\right) & \forall i \in N_c \\ 0 & \textit{otherwise} \end{cases}$$

$$u_{ij}(t+1) = u_{ij}(t) + \Delta u_{ij}$$

5. *Continuation*
    a. $t = t + 1$
    b. Decrease learning rate and neighborhood with iterations

$$\alpha(t) = \alpha_0\left(1 - \frac{t}{T}\right) \qquad d(t) = \left\lceil d_0\left(1 - \frac{t}{T}\right)\right\rceil$$



Training pt. II (alternative)

4b. Use a smoother neighborhood function

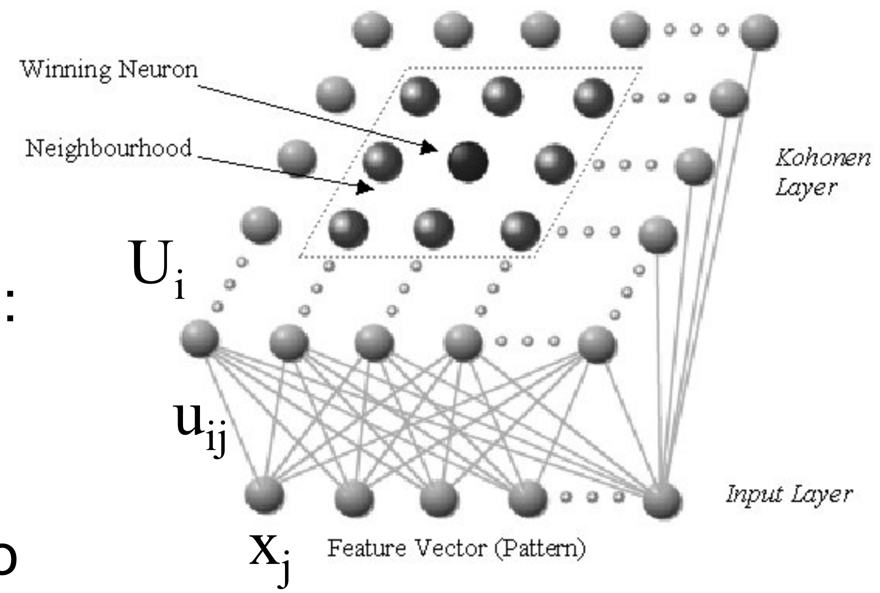
$$\Delta u_{ij} = \Theta(i,c,t)\,\alpha(t)\left(x_j^{\mu} - u_{ij}\right)$$

$$u_{ij}(t+1) = u_{ij}(t) + \Delta u_{ij}$$

Typical choice for Θ is a gaussian:

$$\Theta(i,c,t) = \exp\left(-\left|r_i - r_c\right|^2 / 2\sigma^2\right)$$

Where σ is gradually decreased to increase the size of the neighborhood

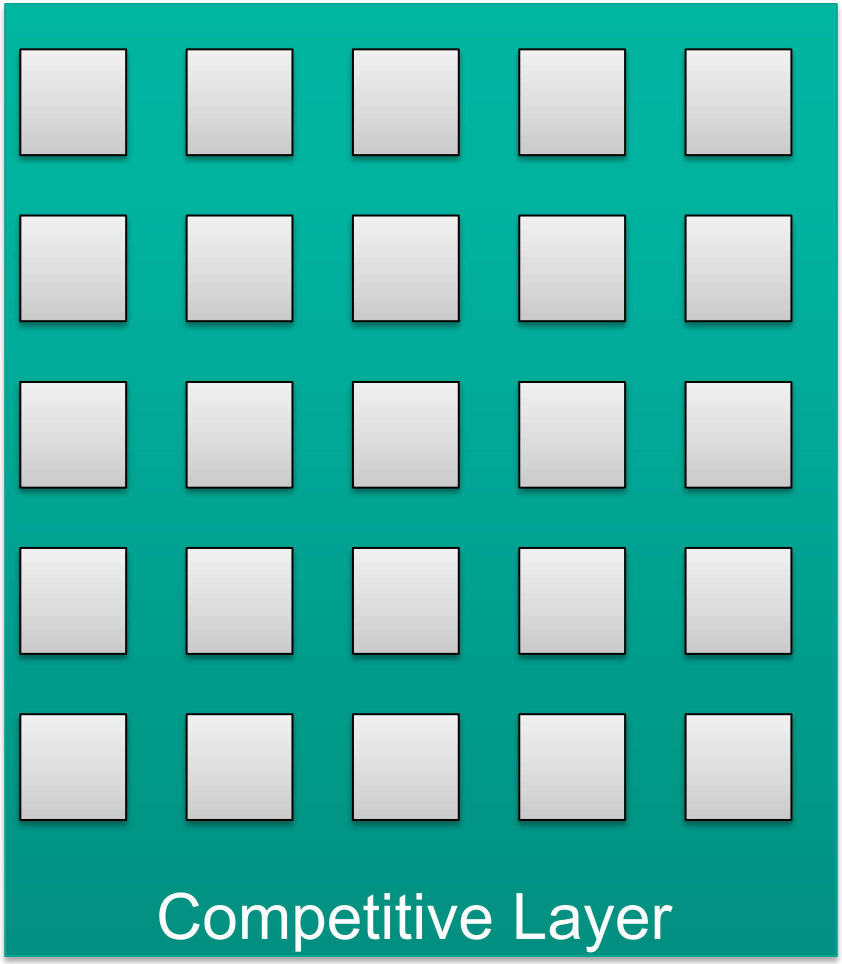


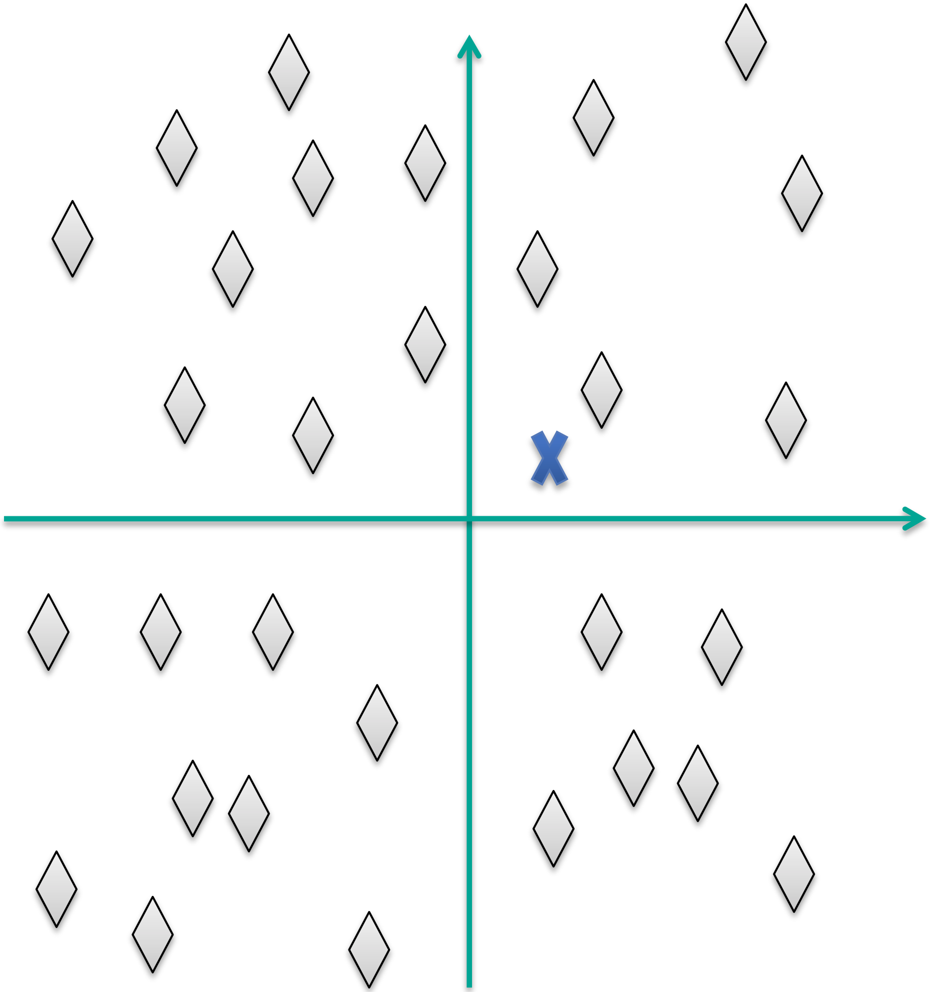
Derivation of the training rule

Karlsruher Institut für Technologie

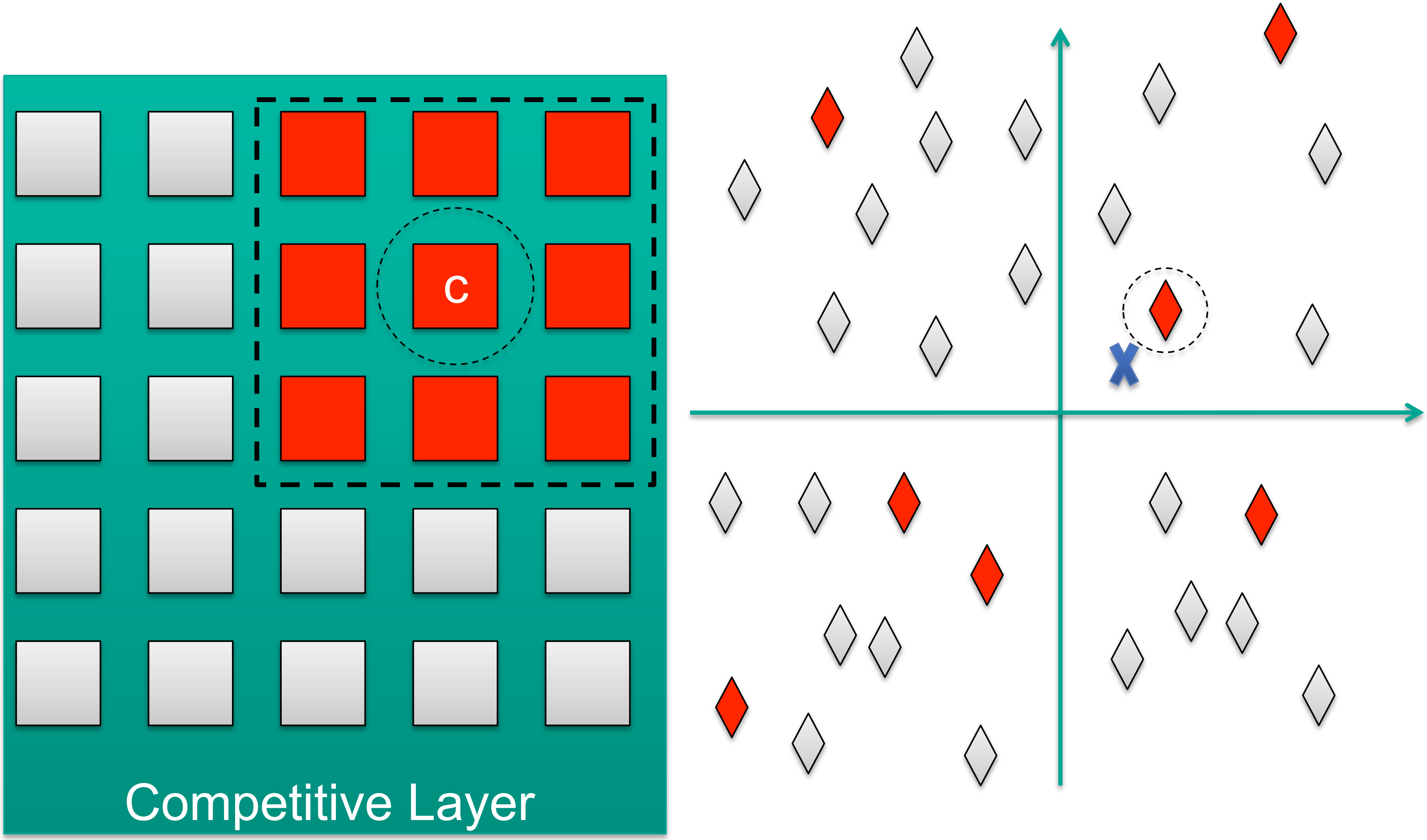
- Cost function: $$E\{u_{i,j}\} = \frac{1}{2}\sum_{ijk\mu} M_k^\mu \Theta(i,k)\left(x_j^\mu - u_{ij}\right)^2$$
$$= \frac{1}{2}\sum_{i\mu} \Theta(i,c)\left|x^\mu - u_i\right|^2$$
- Gradient descent on cost function:
$$\langle u_{ij}\rangle = -\alpha\frac{\partial E}{\partial u_{ij}} = \alpha\sum_{k\mu} M_k^\mu \Theta(i,k)\left(x_j^\mu - u_{ij}\right)$$
$$= \alpha\sum_{\mu} \Theta(i,c)\left(x_j^\mu - u_{ij}\right)$$
- (Sum of Kohonen learning rule over all input indices **μ**)

Example

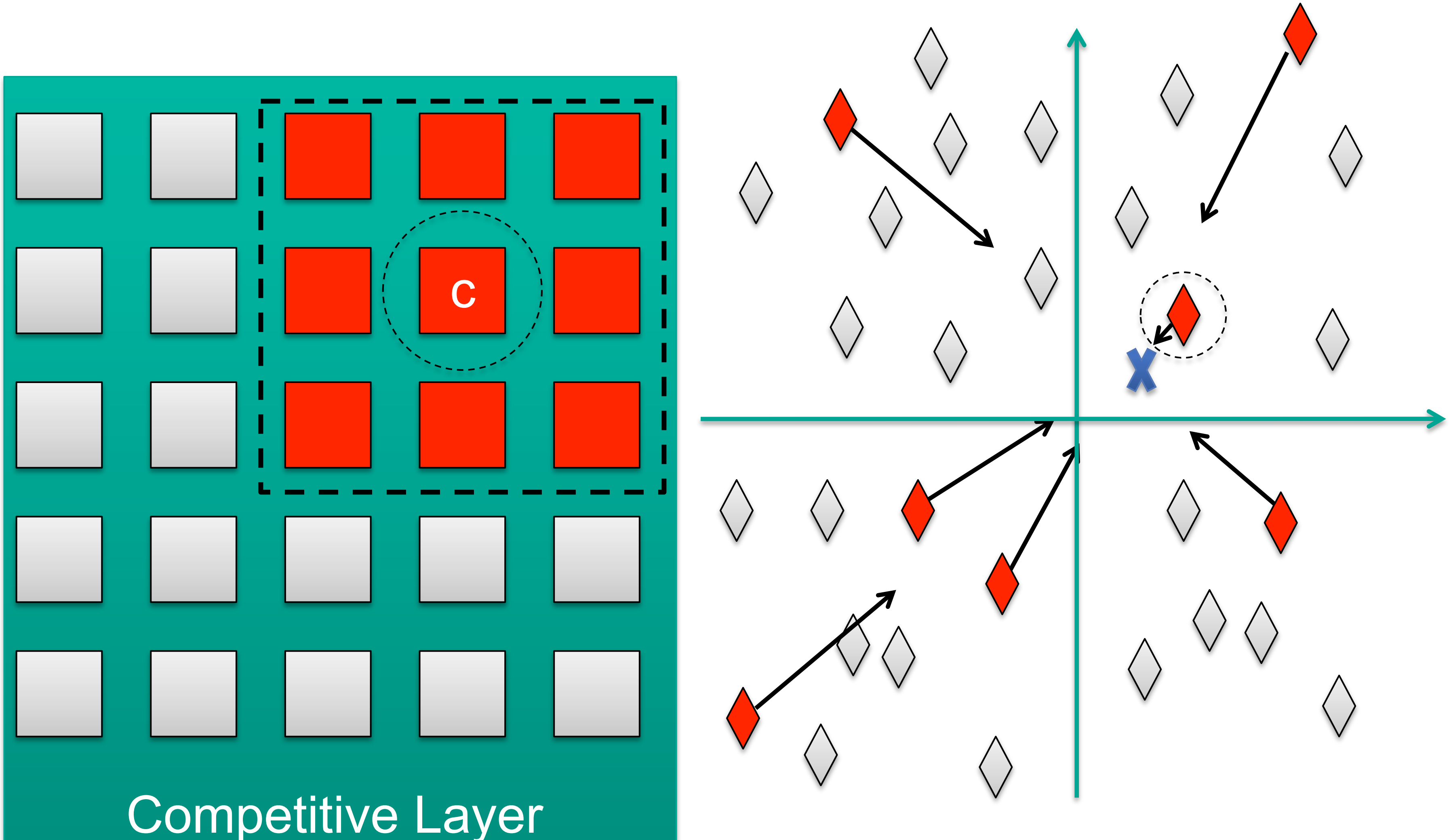




Example

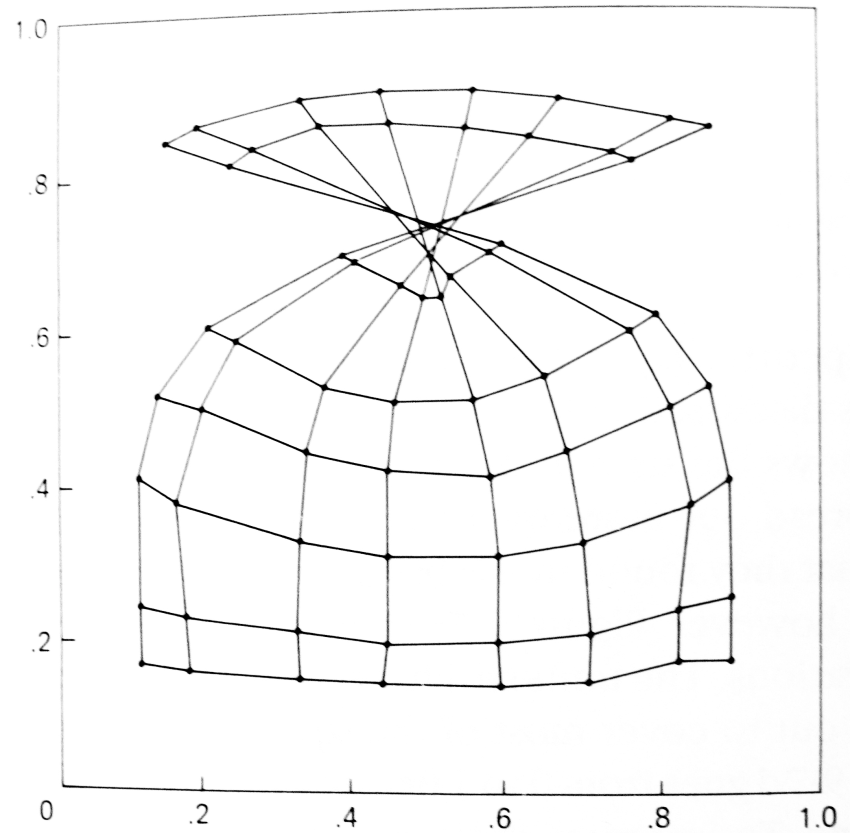


Example



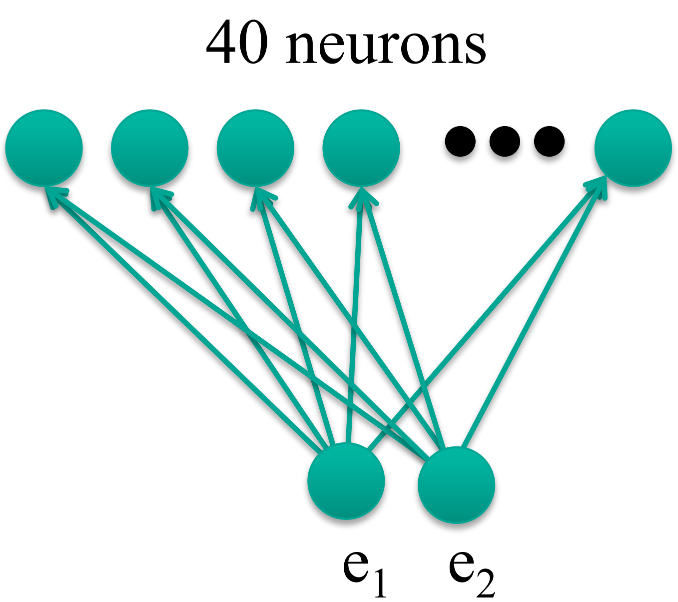
Training Phase

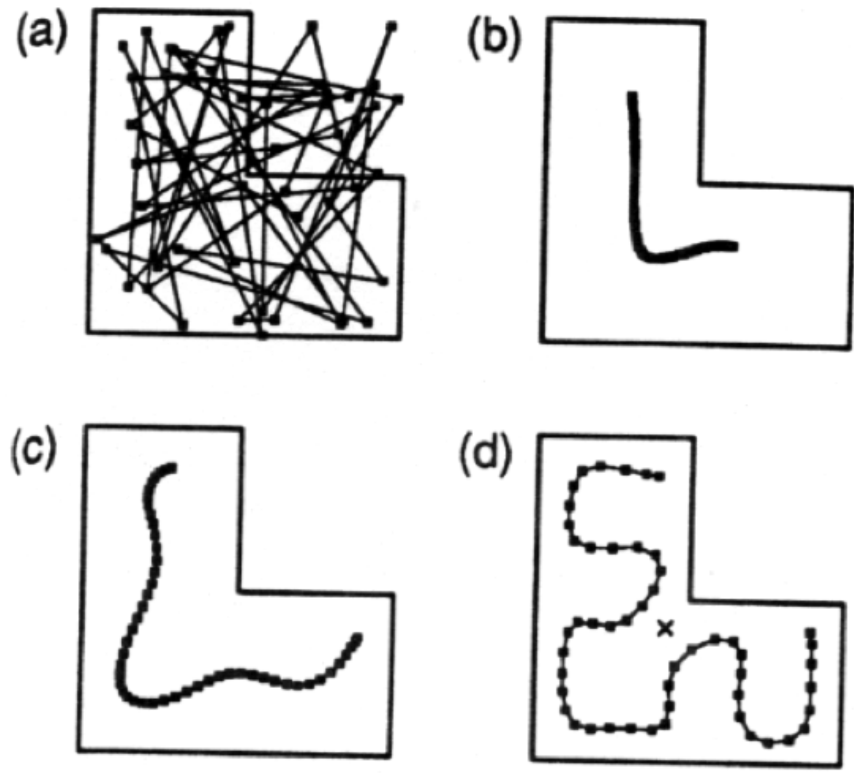
- Two steps:
  1. Unfolding phase
  2. Convergence phase
- Local minima possible
  - More likely with higher complexity of input space
  - Ordering of input patterns can lead to local minima



Input Drawn from Uniform Distributions

2D to 1D





Mapping Mode + Dimensionality Reduction

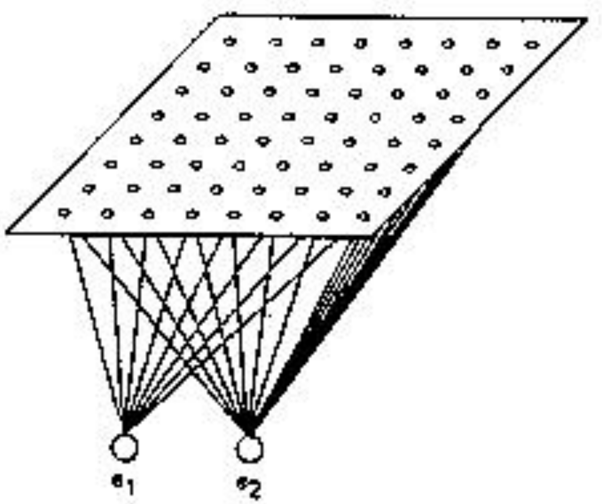
Mapping Mode

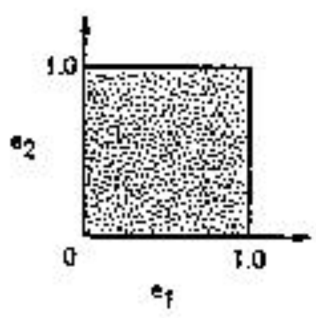
- (After training)
  - In order to map an new input to the network
    - Simply find the best-matching unit

Dimensionality Reduction

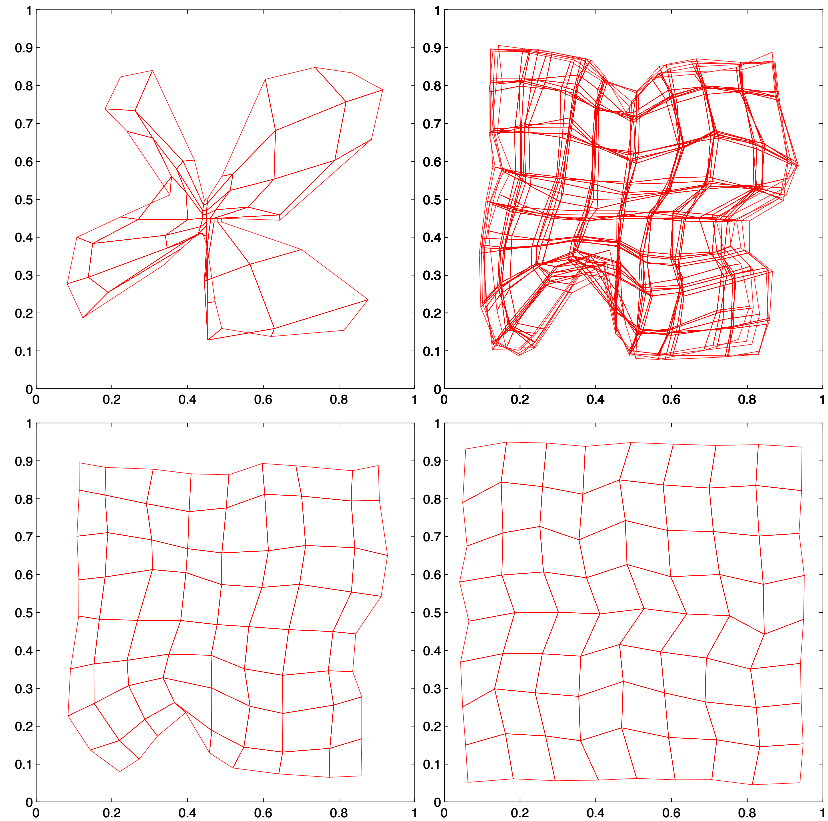
- Dimension of input is number of coefficients in E
- Dimension of map is number of axes
- Interest is often in reducing a high-dimensional problem to a low-dimensional one

Input Drawn from Uniform Distributions

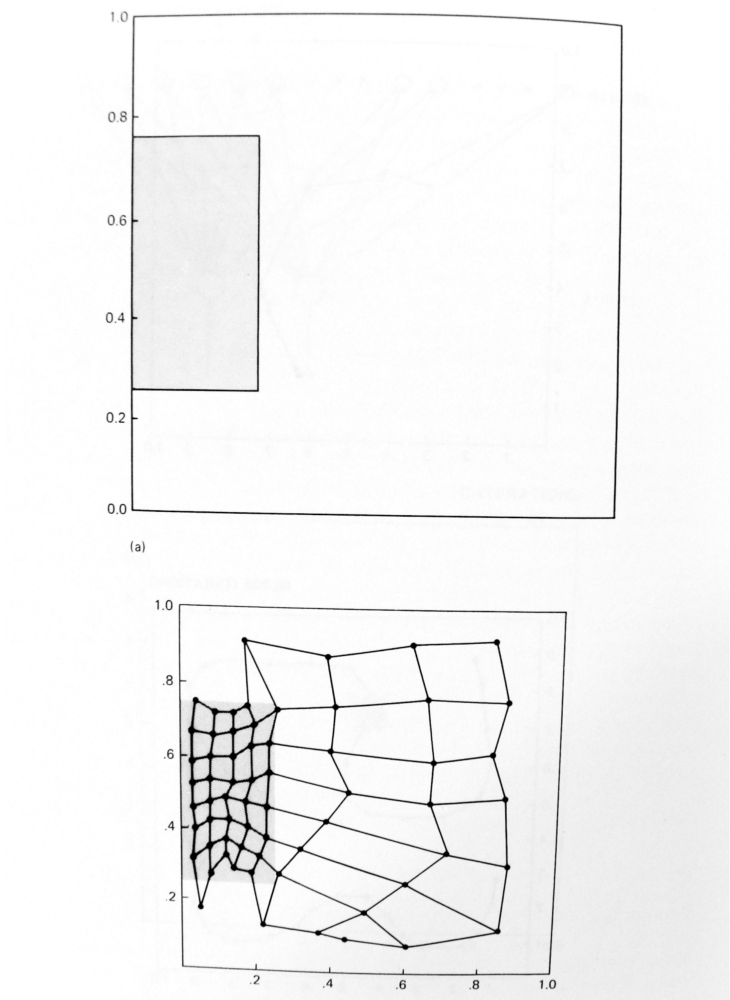




TRAINING PATTERNS [e_1, e_2]



Nonuniform Probability Densities



- Density of inputs higher in grey region
- Greater number of neurons will be drawn to this region

3D Feature Map

DEMO 1

Colors

DEMO 2

Karlsruher Institut für Technologie

Visualizing a dataset

DEMO 3

Organization of an SOM

DEMO 4

How does an SOM produce organization?

- Propositions:
  - **Ordering** and **Convergence**
- From Kohonen (1D case):
  - ***Assumptions:*** *Let* x *be a stochastic variable. Starting with randomly chosen initial values for the weights* u_i, *the set of numbers* $\{U\}=(u_1,u_2,\ldots u_l)$
  - ***Proposition 1:*** $\{U\}$ *will become ordered with* $t\rightarrow\infty$ *through the process defined by (A), (B), and (C)*
  - ***Proposition 2:*** *Once ordered, the set remains so.*
  - ***Proposition 3:*** *The density of the* u_i *will finally approximate some monotonic function of the pdf* $p(x)$

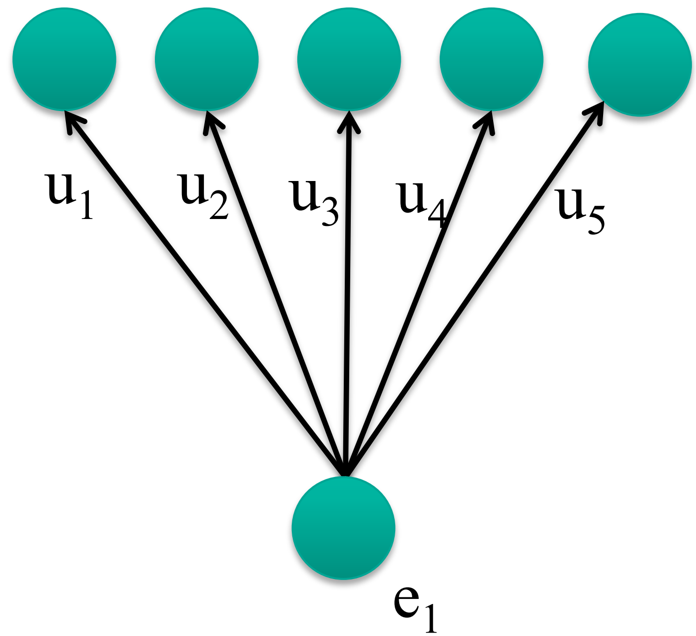
$$(A)\quad \|x-u_c\| = \min_i\{\|x-u_i\|\}$$

$$(B)\quad N_c=\{\max(1,c-1),c,\min(l,c+1)\}$$

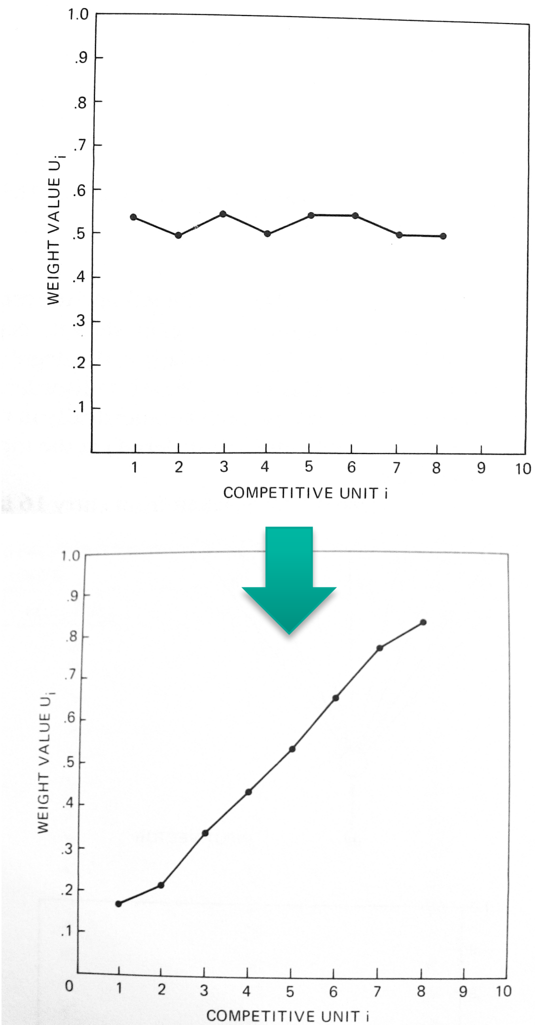
$$(C)\quad \begin{array}{ll} du_i/dt=\alpha(x-u_i) & \text{for } i\in N_c \\ du_i/dt=0 & \text{otherwise} \end{array}$$

Informal Proof of Ordering

- **1D example:**
- Assume no unit is near an edge
- Compare values of weights [u_1 … u_5]
- 4 pairs of adjacent units
  - Two possible orderings of weight values
  - 2^4=16 orderings for 5 units



Informal Proof of Ordering

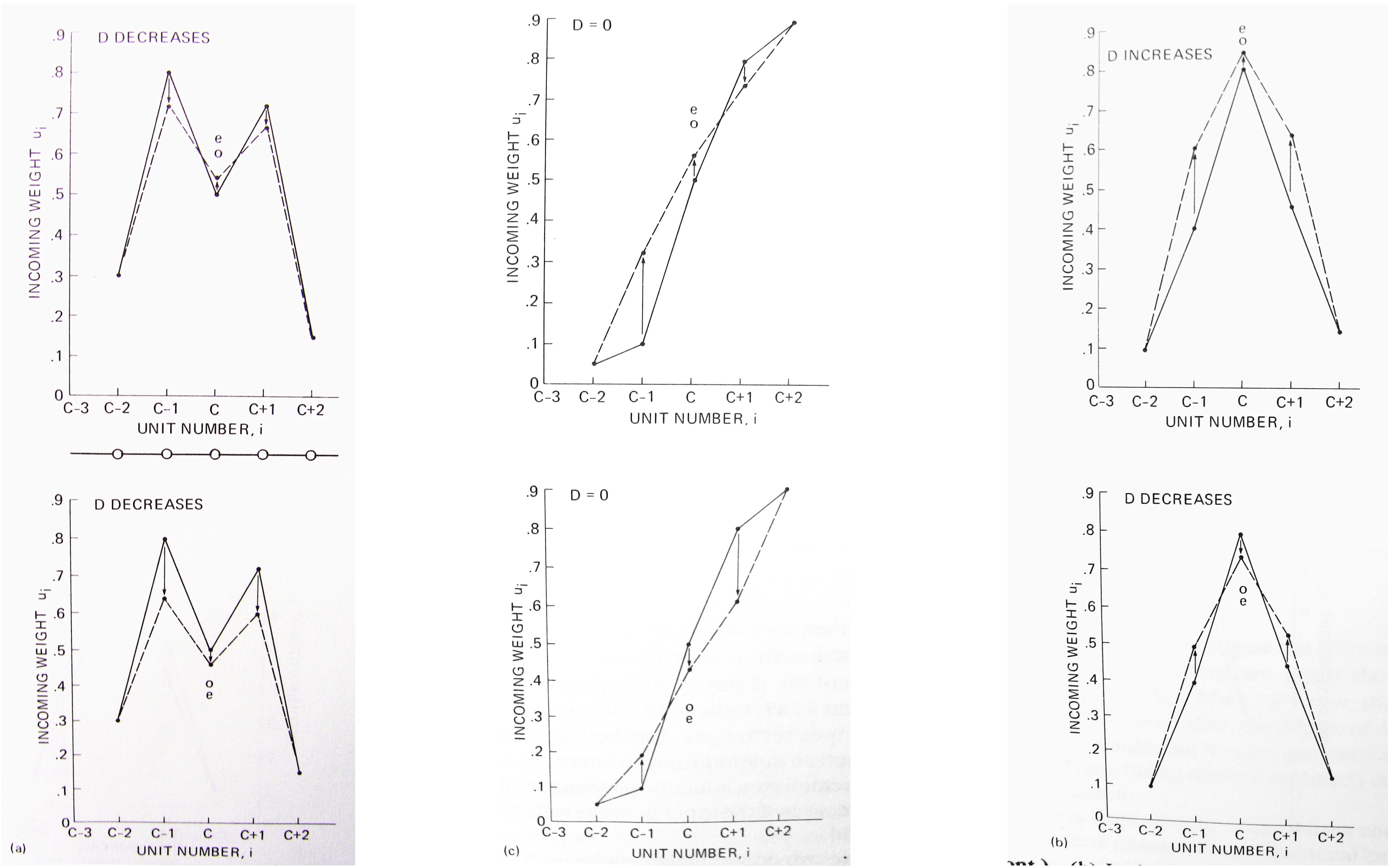


- We will examine for three of the 16 cases what happens when an input is presented to the middle unit (to avoid edge effects)
- Define a measure of disorder:

$$D = \sum_{i=2}^{t} \left| u_i - u_{i-1} \right| - \left| u_t - u_1 \right|$$

- Assume e is a $RV \in [0,1]$
- Assume *c* is center unit, *d*=1

Informal Proof of ordering



Proof of Ordering (More Rigorous)

- **Assume** neurons *1,2,…,l*
- **Let** $u_i = u_i(t)$ in ***R***
- **Let** $x = x(t)$ in ***R***
  - With prob. dens. p(x) on $[a,b]$
- **Assume** equations (A), (B), (C)
- **Assume** winning unit *c* is unique in all cases

$$(A) \quad \|x-u_c\| = \min_i \{\|x-u_i\|\}$$

$$(B) \quad N_c = \{\max(1,c-1), c, \min(l,c+1)\}$$

$$(C) \quad \begin{array}{ll} du_i/dt = \alpha(x-u_i) & \text{for } i \in N_c \\ du_i/dt = 0 & \text{otherwise} \end{array}$$

$$D = \sum_{i=2}^{t} |u_i - u_{i-1}| - |u_t - u_1|$$

Proof of Ordering (More Rigorous)

- **Assume:**

i. $x(t)$ is almost surely integrable on finite intervals

ii. $p(x)$ is independt of t and >0 on [a,b] only

iii. $x(t)$ attains all values on [a,b] almost surely during all time intervals [t,∞)

iv. The initial values for u_i are choes randomly from a distribution on [a,b]

- **Theorem**: In a process defined by (A), (B), and (C), {U} will become almost surely ordred asymptotically

$$(A) \quad \|x - u_c\| = \min_i \{\|x - u_i\|\}$$

$$(B) \quad N_c = \{\max(1, c-1), c, \min(l, c+1)\}$$

$$(C) \quad \begin{array}{ll} du_i/dt = \alpha(x - u_i) & \text{for } i \in N_c \\ du_i/dt = 0 & \text{otherwise} \end{array}$$

$$D = \sum_{i=2}^{t} |u_i - u_{i-1}| - |u_t - u_1|$$

Proof of Ordering (More Rigorous)

- **Partial Proof:**
- **Step 1:**
  - Consider only the case for $3 \le c \le l\text{-}2$
  - Define the partial sum $S(c)$: $$S(c) = \sum_{i=c-1}^{c+2} |u_i - u_{i-1}| \text{ for } 3 \le c \le l-2$$
  - For any t, the sign of $|u_i\text{-}u_{i-1}|$ attains one of at most 16 combinations
  - Consider dS/dt in two cases

$$\dot{u}_{c-2} = 0$$
$$\dot{u}_{c-1} = \alpha\left(x - u_{c-1}\right)$$
$$\dot{u}_c = 0$$
$$\dot{u}_{c+1} = \alpha\left(x - u_{c+1}\right)$$
$$\dot{u}_{c+2} = 0$$

Case	$u_c - u_{c-1}$	$u_{c+1} - u_c$	$u_{c+2} - u_{c+1}$
$a0$	>0	>0	≥ 0
$a1$	>0	>0	≤ 0

$$S_{a0} = -u_{c-2} + u_{c+2} \qquad \dot{S}_{a0} = 0$$
$$S_{a1} = -u_{c-2} + 2u_{c+1} - u_{c+2} \qquad \dot{S}_{a1} = 2\alpha\left(x - u_{c+1}\right) < 0$$

Proof of Ordering (More Rigorous)

- **Step 1 (Cont.'d)**
  - For all 14 other cases, S≤0
  - S≤0 also for other values of c not considered here
  - Partial sum *dS/dt≤0* → D is monotonically decreasing

- **Step 2:**
  - In view of step 1, D(t) must tend to some limit

$$D^* = \lim_{t\to\infty} D(t)$$

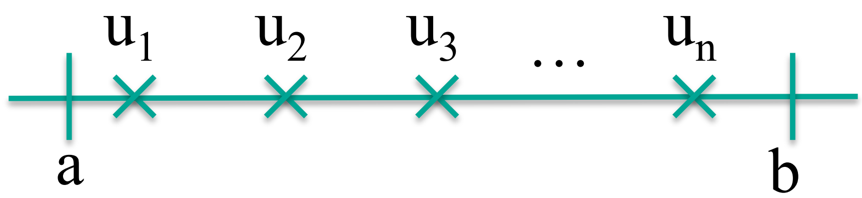
  - D^*=0: **Proof by contradiction:**
    - Since D^* is constant, dD^*/dt=0 for all t
    - Assume D^*>0 and a certain ordering of {U} without loss of generality
    - Examining a series of cases, we are inevitably led to contradictions
    - Therfore D^* must equal 0
    - Thus the asymptotic state is ordered

Convergence Analysis in 1-D Case

$$u^{new} = u + \alpha(e - u)$$

- 1 neuron, 1 input
- *e* bounded in [a, b]
- If 0 < α < 1, then *u* is bounded in [a, b]
- Therefore E(u) is bounded
- Therefore $$E\left\{\frac{du}{dt}\right\} = 0$$

Convergence Analysis in 1-D Case



- Assume $a<u_1<u_2<\ldots<u_n<b$
- The average attraction on each weight is zero in its domain
- Therefore the weights stay bounded
- Therefore $$E\left\{\frac{du_i}{dt}\right\}=0$$
- This is true only if $E(u_i)$ are distributed homogenously (the attraction from the left balances that of the right)

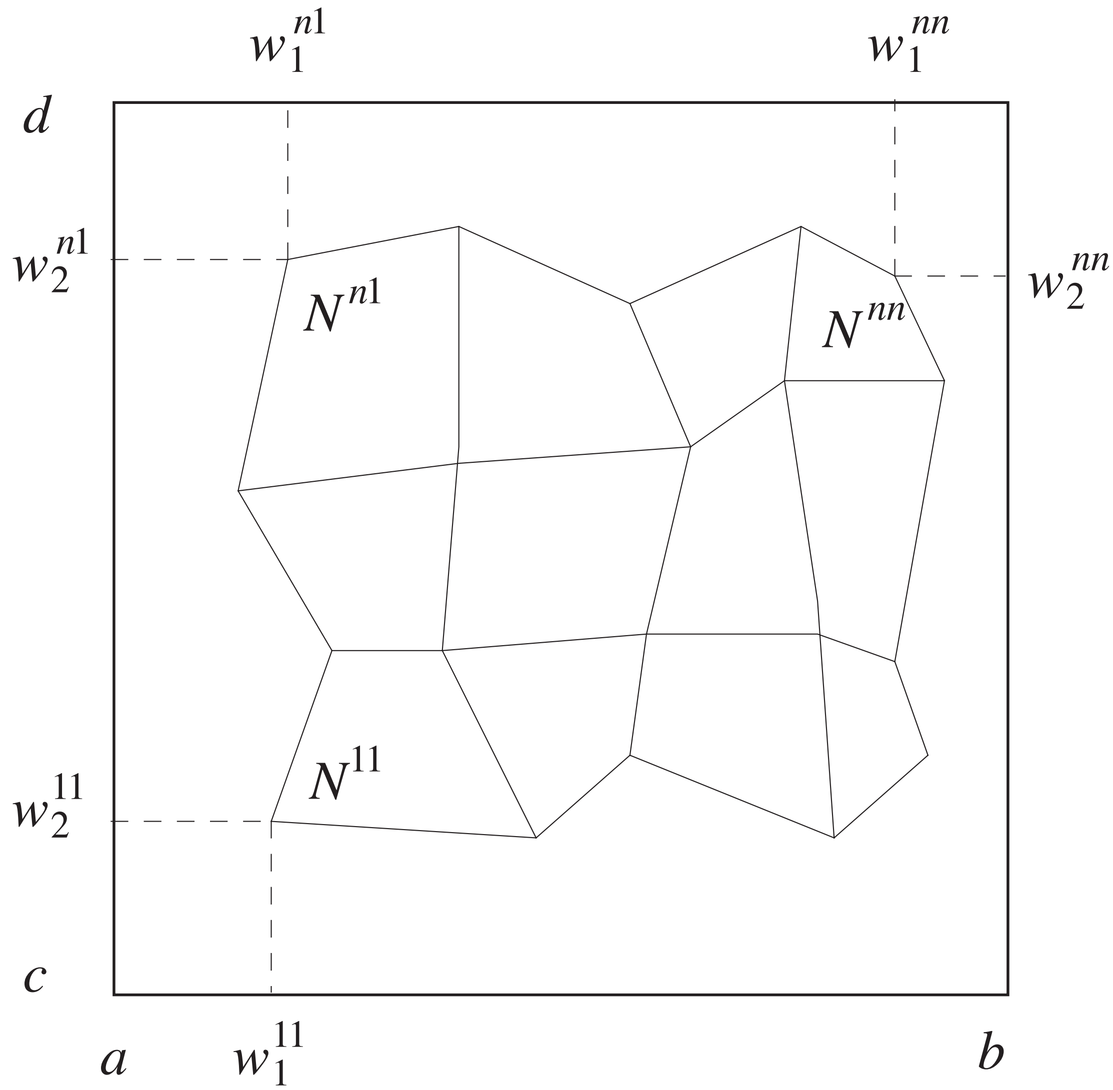
Convergence Analysis in 2D Case

- Assume monotonic ordering of network weights in two dimensions:

  $$w_1^{ij} < w_1^{ik} \quad \text{if } j < k$$

  $$w_2^{ij} < w_2^{kj} \quad \text{if } i < k$$

- 2D problem breaks down into two 1D problems



Convergence Analysis in 2D Case

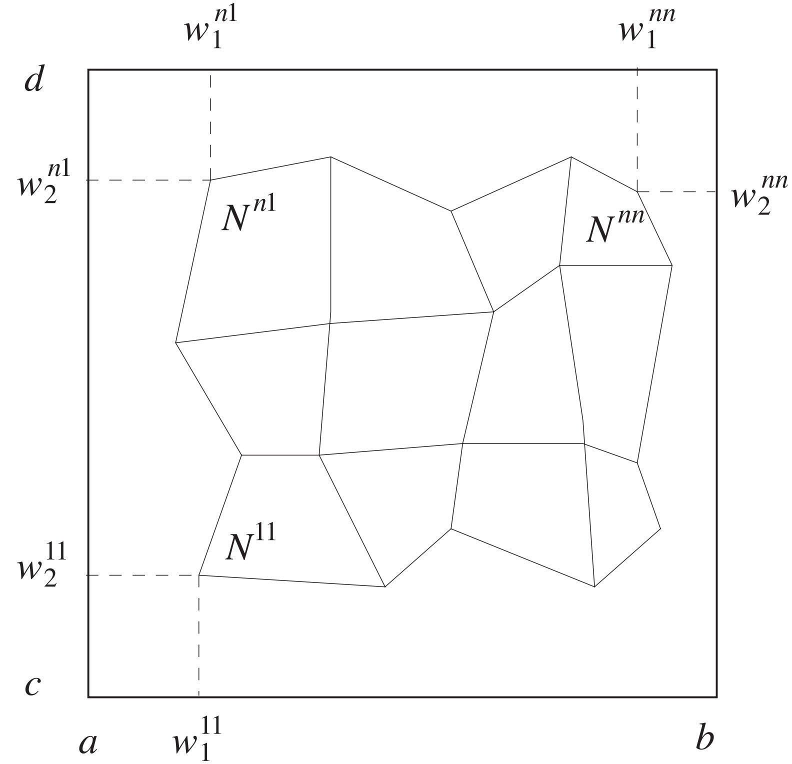
- Consider one column:
- Define the average value of weights in the *j.* column:

$$w_1^j = \frac{1}{n}\sum_{i=1}^{n} w_1^{ij}$$

- These averages are monotonically arranged:

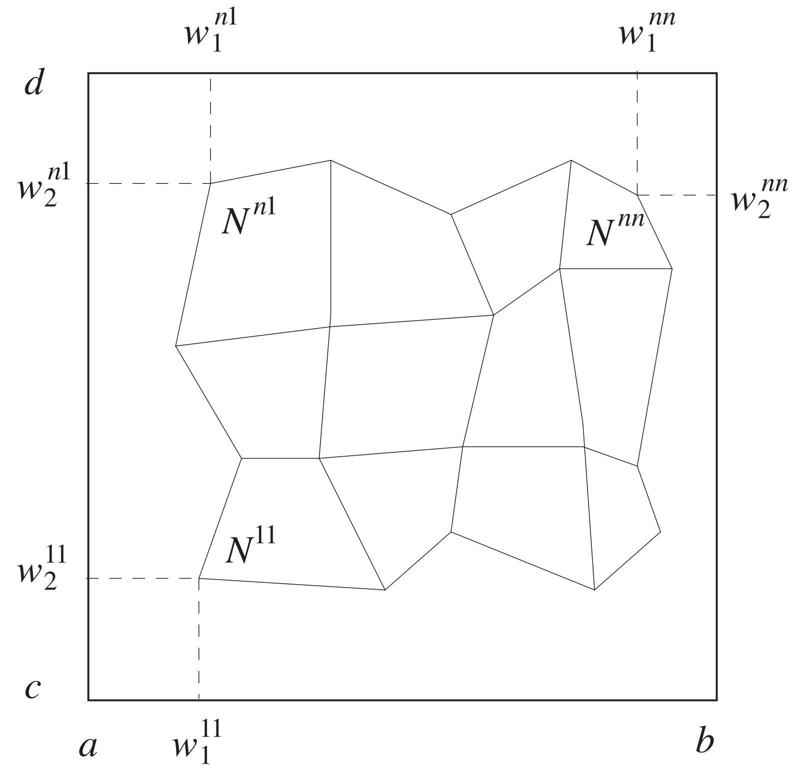
$$a < w_1^1 < w_1^2 < \ldots < w_1^n < b$$

- → $E(w_1^i)$ will reach a homogenous distribution on *[a,b]*
- $w_1^{11}, w_1^{21}, \ldots w_1^{n1}$ will oscillate around $E\{w_1^1\}$
- Analogous reasoning for each column and each of the rows
- → Convergence to a stable state (with small enough α)



Convergence Analysis in 2D Case

- Condition for arrival at stable state:
  - Unfolding of the randomly intialized map
  - Can fail in early stages
  - No general solution for conditions which can guarantee this unfolding



Measurement of Quality

Quantization Error

- There will always be some difference between a given input pattern and its closest neuron

$$\sum_{k \in K} \left\| e_k - c_j \right\|$$

$$j = \underset{l}{\operatorname{argmin}} \left\| e_k - c_l \right\|$$

- Gives notion of quality of mapping

Topographic Error

- Proportion of all vectors for which the first and second BMUs are not adjacent

$$TE = \frac{1}{N} \sum_{i=1}^{N} u(E_i)$$

$$u(E_i) = \begin{cases} 1 & \text{if 1. and 2. BMUare adjacent} \\ 0 & \text{else} \end{cases}$$

Representations

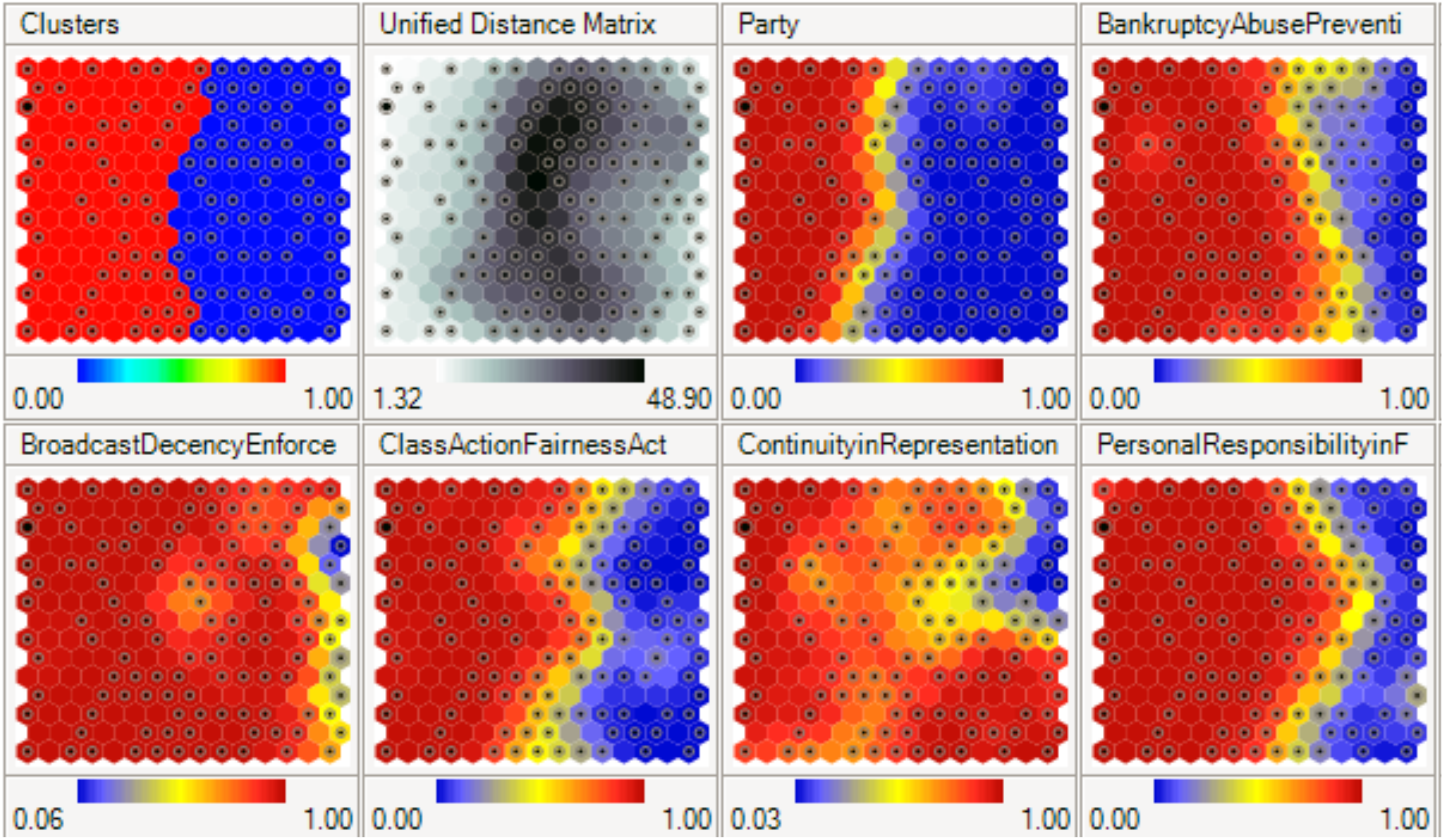
U-matrix

- Distances between the weights of each node are represented as shades of grey on the map
- Clusters of similiar data appear white

Component Planes

- Each plane represents the value assigned by each neuron for each component in the vector

Representations



Best Practices for Generating Good SOMs

- Scaling of input coefficients
  - Orientation of ref. vectors in input space depends on scaling of vector components
  - No simple rule for rescaling
  - With high input Dim., normalize variance of each component over trainin data
  - Try heuristic rescaling and check quality with quantization error
- Forcing representations to desired map location
  - Sometimes we may want to map « normal » data to a specific place on the map
  - Use copies of this data for the initial values of the weights at these locations
  - Keep learning rate low for these locations

Best Practices for Generating Good SOMs

- Learning with a small number of training samples
  - Number of iterations may be much greater than number of samples
  - Alternatives:
    - Cyclical presentation
    - Random presentation
    - Bootstrap learning
  - Results:
    - Ordered cyclic application « not noticeably worse than other methods »
- SOM for sequential signals

Optimal Dimension?

Considerations

- Consider case of points on a sphere
  - Input is 3D
  - However, best Kohonen map is 2D
- Some suggest to compute effective dimension of data before selecting dim. of SOM

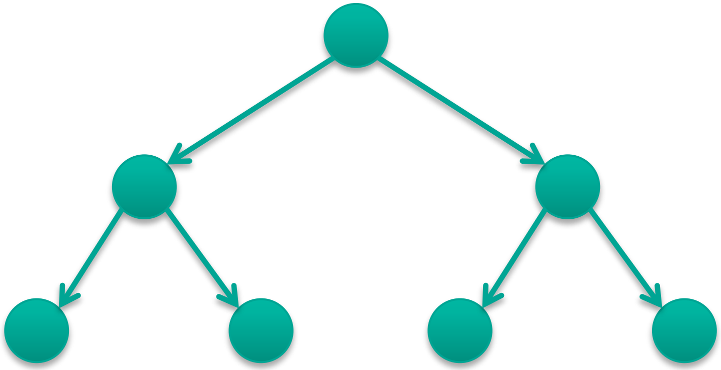
Computing dimension of data experimentally

- Measure variation in $N(\varepsilon)$ with varying ε
- $N(\varepsilon)$ is the number of data points closer to another data point than ε
- Ex. Points on a 2D plane in 3D
  - $N(\varepsilon) \approx \varepsilon^2 \rightarrow$ Two dimensions
- Plot $log(N(\varepsilon))$ vs $log(\varepsilon)$
  - Slope of regression line is fractal dimension of data

Some Ideas for Modifications of the SOM

- Different matching criteria
  - Different metrics
  - Other criteria for matching
- Traditional optimization methods to accelerate searching
- Hierarchical searching
  - Tree-search SOM
    - Hierarchy of SOMs
  - Hypermap
    - Use subset of input to find candidate nodes
    - Select winner with other input

- Tree-search
  - Worst-case linear search: N
  - Worst-case binary tree search: $\log_2 N + 1$



Use of SOMs for Sequential Data

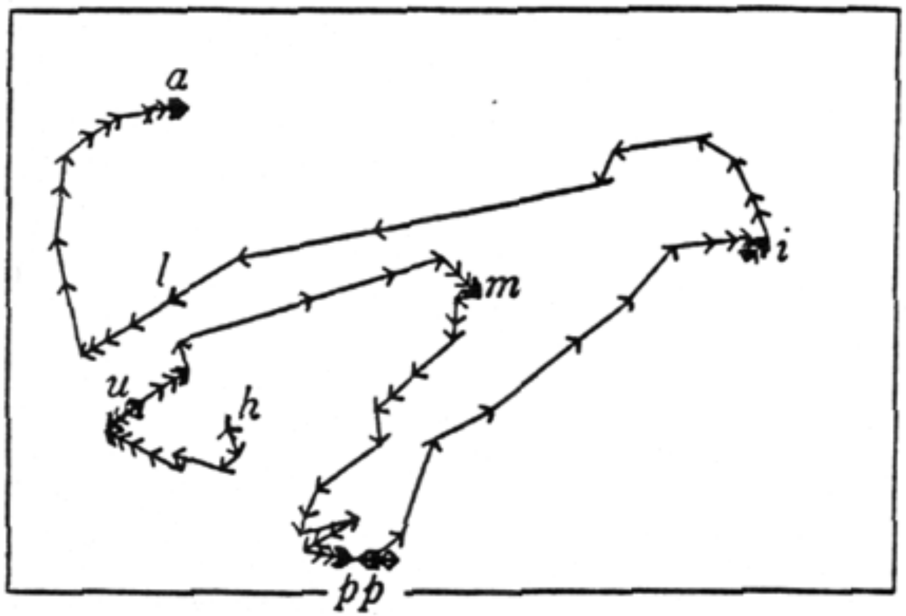
- Dynamic resizing of the map depending on results (i.e. quantization error during learning)
- Enhancement of rare cases
  - SOM represents $p(x)$
  - Many important cases may occupy no space on the map
  - Enhancement through increased value of α for these input samples
- Original SOM idea based on matching of *static* signal patterns
  - Asymptotic state not steady unless topological relationships between patterns are steady
- For sequential patterns
  - Use of time window
    - Concatenation of successive samples into pattern vector of higher dimension
  - Time dependence reflected in order of elements in input vector

Some Ideas for Modifications of the SOM

- Linear Initialization
  - Begin weights in ordered state
  - 1. Determine the two eigenvectors of the autocorrelation matrix of *x* that have the largest eignvalues
  - 2. Let these eigenvectors span a two-dimensional linear subspace
  - 3. Define a rectangular array along this subspace
    - Centroid coincides with mean of x(t)
  - 4. Intial values of weights are then identified with the array points
  - 5. Number of cells in horizontal / vertical should be proportional to the two largest eigenvalues
- One may directly start the learning with the convergence phase thereafter

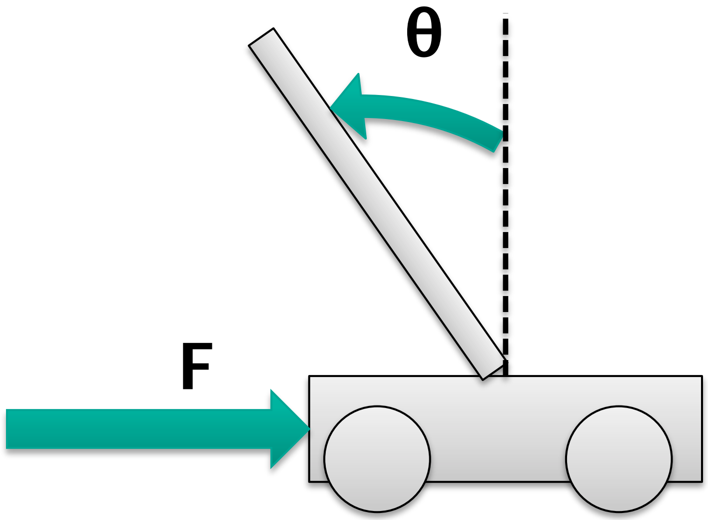
Applications – Phoneme Recogntion

- Kohonen built a system for recognizing Finnish phonemes
- 21 distinct phonemes
- Speech sampled at 8ms
  - FFT→ 15-component vector
- Two-layers
  - 15 input units
  - 96 competitive units
- Result: Similarity map of phonemes



Activation path of phonemes in Finnish word *humppila*

Applications – Approximation of Functions – Pole-balancing system



- Cart with pivoting pole
- Goal: Keep pole vertical

$$f(\theta) = \alpha \sin\theta + \beta \frac{d\theta}{dt}$$

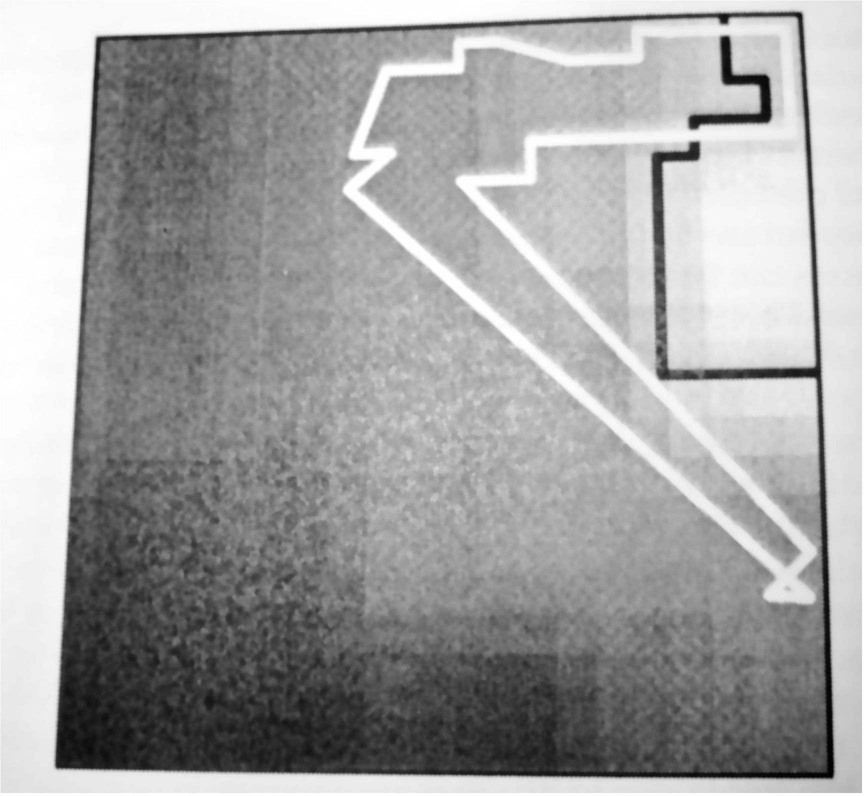
- Let:
  - $x = \theta$
  - $y = d\theta/dt$
  - $z = f$
- We have a three-dimensional surface in (x,y,z)
- Adapt two-dimensional SOM to surface
- Control aspect:
  - For a given x and y, find neuron U_k for which weights u_{k1} *and* u_{k2} are closest to x and y
  - Then $f(z)$ can be taken as u_{k3}
- Map can be updated incrementally as new data arrives

Applications – Analysis of Large Systems

- Understanding and modeling of complex, interrelated variables in large systems is problematic (factory, manufacturing, distribution, industry, etc.)
- Automated measurement produces large amounts of data
- Convert measurements to some simple & comprehensible display
  - Reduce dimensionality
  - Preserve relationships between system states
  - Allow operators to visually follow system state
  - Help understand future behavior
  - Enable fault identification

Applications – Analysis of Large Systems

- Consider system with several real measurements
  - Normalize dynamic ranges
  - Include some control variables
- Once trained on the system, the SOM can be used in two modes:
  1. Trajectory tracking on original SOM
  2. Trajectory tracking on component planes

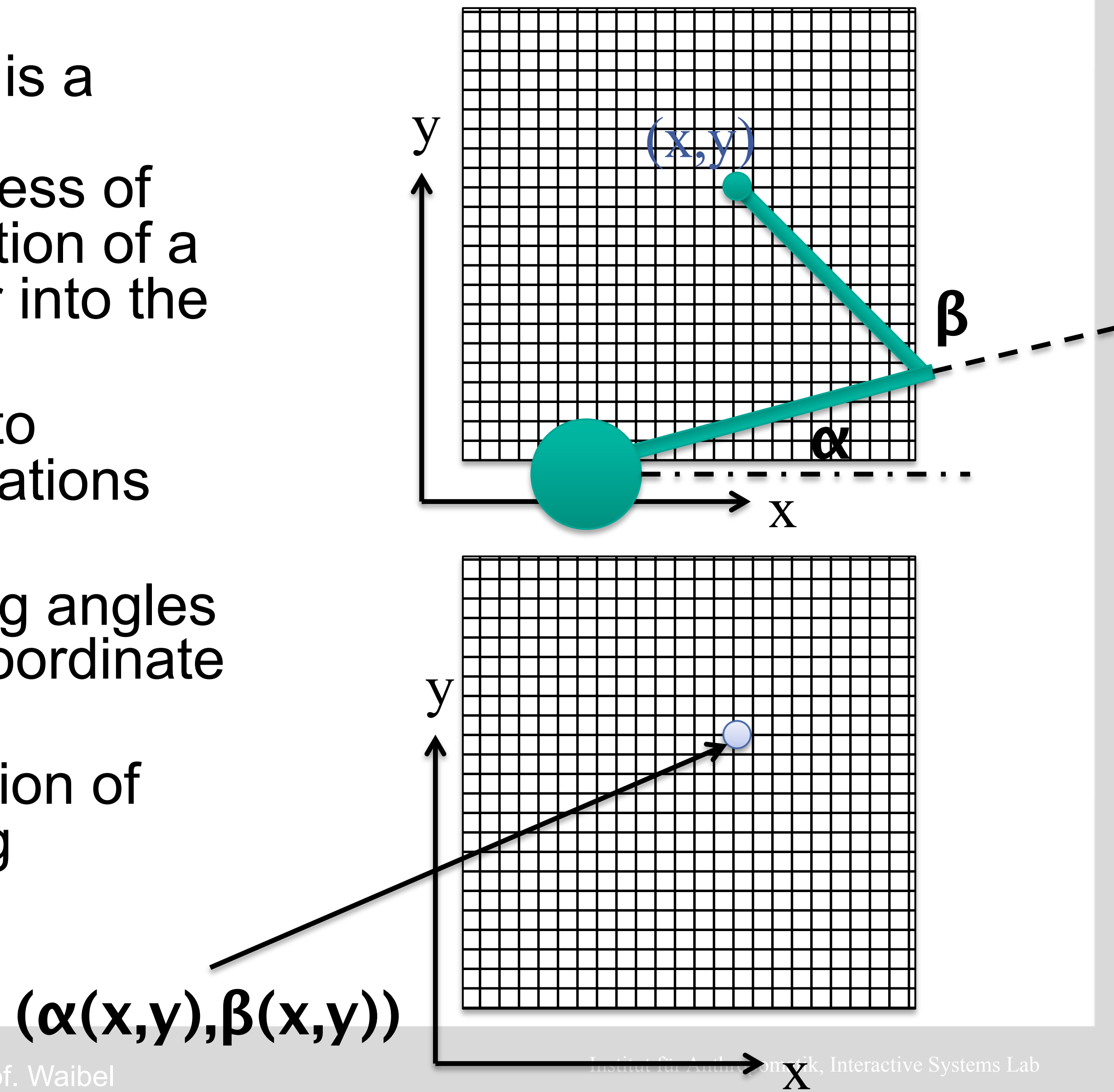


Applications – Analysis of Large Systems -- Fault Identification

- Fault detection can be based on quantized error:
  - Compare input vector to all weights
  - When distance exceeds a given threshold, we probably have a fault situation
  - (Operation point in a space not covered by the training data)
- Fault visualization
  - Faults may be rare & true measurements for training may not be possible
    - → Simulate faults
- If measures of most typical faults available / simulated
  - SOM can be used as monitor
- If operating conditions vary greatly or one cannot define « normal » conditions
  - SOM must be used on two levels
    - 1st level map – fault detection map with quantization error
    - 2nd level map – more detailed – identify reason of fault
      - Store sequence of input before & during occurence

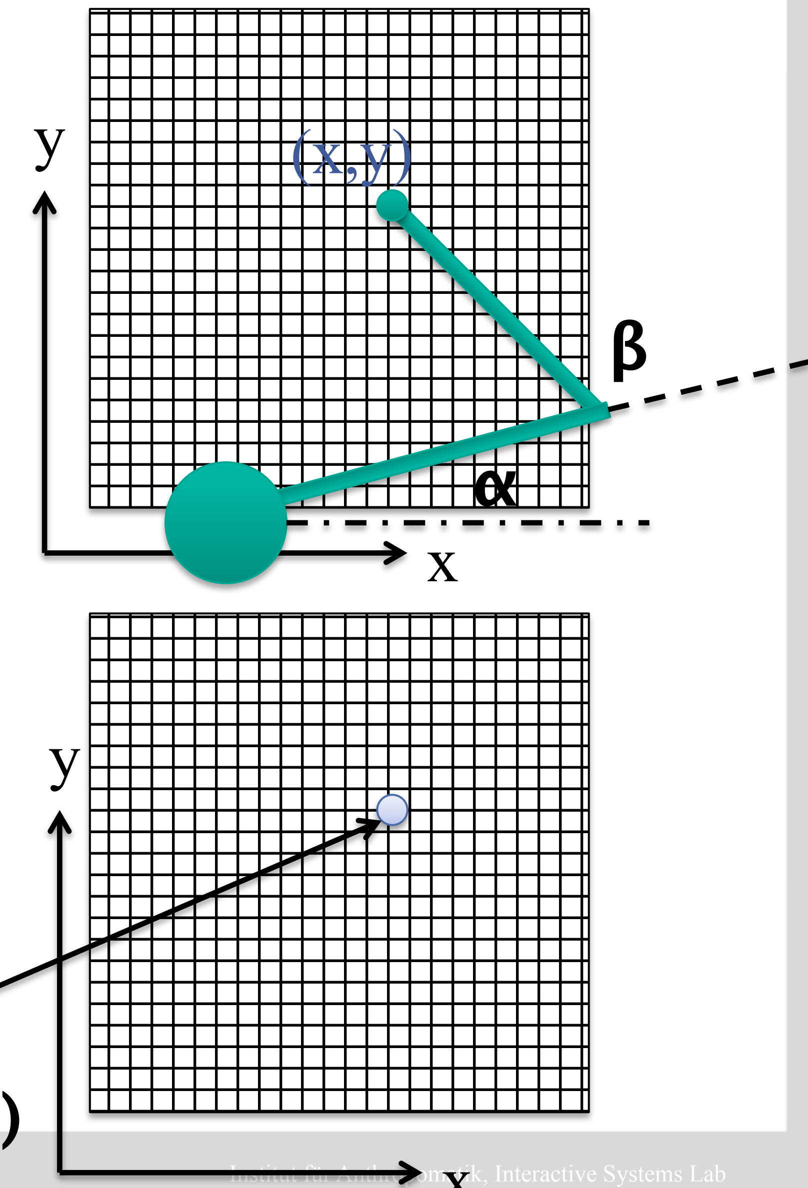
Applications – Inverse Kinematics

- Inverse kinematics is a concept in robotics describing the process of translating the position of a robot's end effector into the angles of its joints
- Move end effector to various random locations on grid
- Store corresponding angles at corresponding coordinate in map
- Result: Approximation of the function relating position to angles



Applications – Inverse Kinematics

- Motion from point A to point B:
  - Use table as lookup and interpolate values
- Inclusion of obstacles in workspace is possible



Applications – Traveling Salesman Problem

- Salesman must visit *N* cities
- Wishes to minimize trajectory length
- NP-hard problem in combinatorial optimization

Applications – Traveling Salesman Problem

- Kohonen Maps can solve this problem <u>approximately</u>
- Assume map is ring of neurons
- 1 neuron per city
- 2 dimensions – coordinates of cities

Applications – Traveling Salesman Problem

- Drawbacks:
  - No cost function associated with the process

Applications – Traveling Salesman Problem

DEMO 5

Preprocessing of Line Figures

- ANNs act as classifiers which work well if input data elements represent *static* properties
- Natural signals are often dynamic
  - Sub-elements are mutually dependent
- Most ANNs do not tolerate even insignficant transformations of patterns (rotation, translation, or scale)
- One should never use ANN methods for classification of images without preprocessing
- Preprocessing should select a set of features invariant with respect to transformations of input