Machine Learning and Neural Computation: Methods for Data Analysis

David Horn

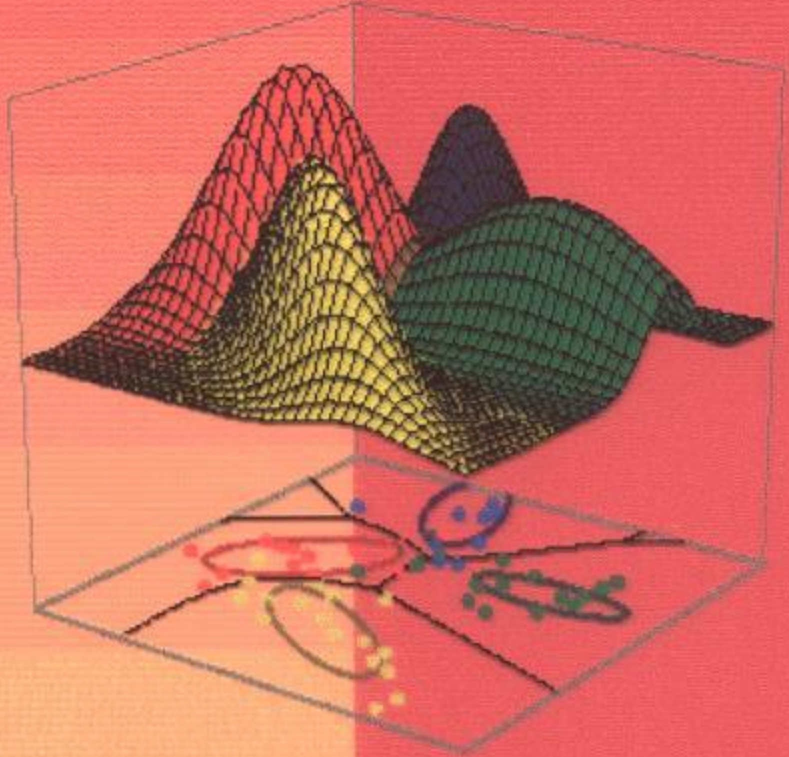
May 27th, 2014

Topics to be discussed:

Pattern classification
Learning from the brain
Neural networks
Back propagation
Support Vector Machine (SVM)
Support Vector Clustering (SVC)
Other clustering methods
Singular Value Decomposition (SVD)
Quantum Clustering (QC)
Dynamic Quantum Clustering (DQC)
Applications

Richard O. Duda
Peter E. Hart
David G. Stork

Pattern Classification



Second Edition

Machine Perception

- Build a machine that can recognize patterns:
  - Speech recognition
  - Fingerprint identification
  - OCR (Optical Character Recognition)
  - DNA sequence identification

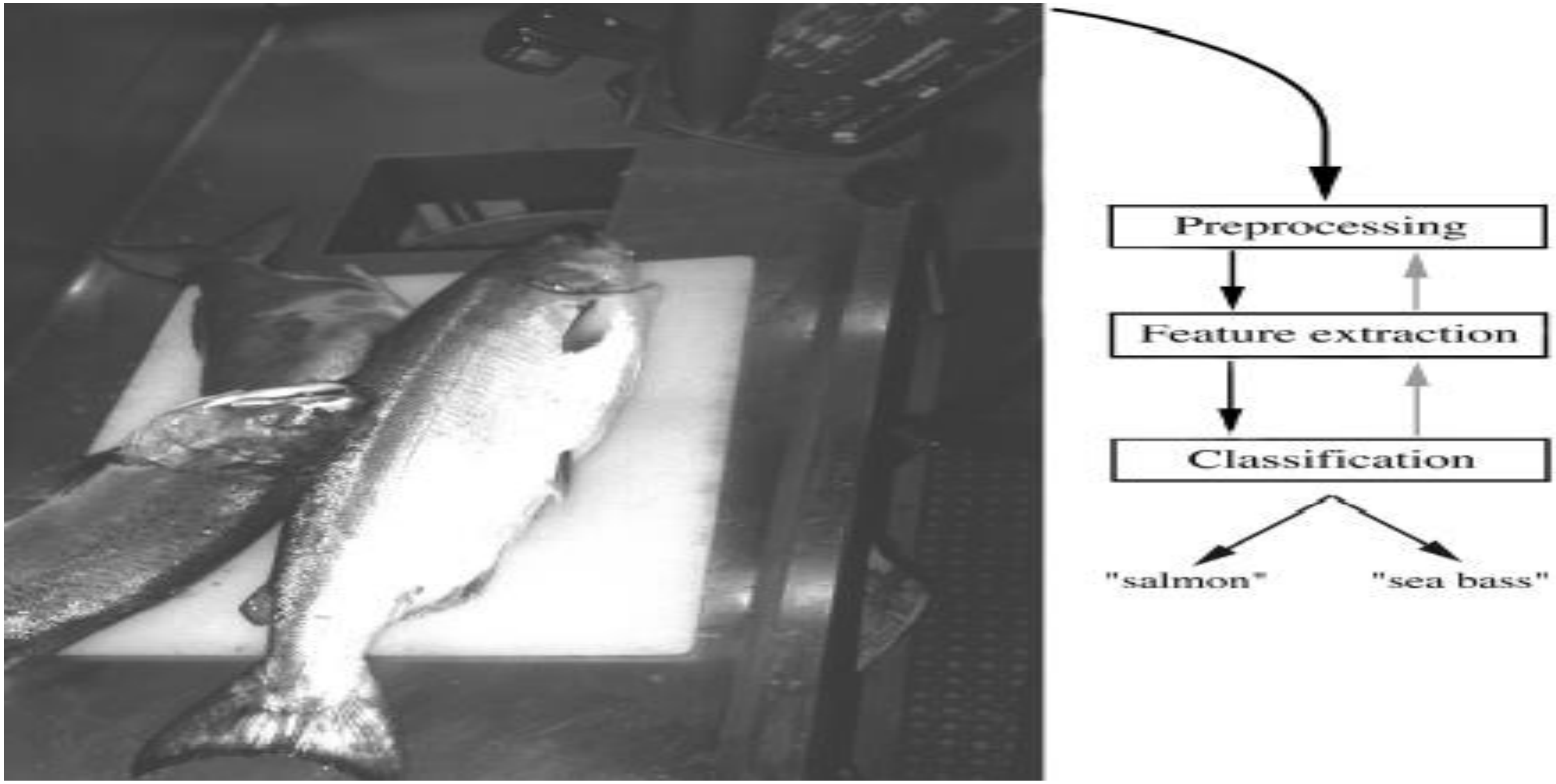
An Example

- “Sorting incoming Fish on a conveyor according to species using optical sensing”

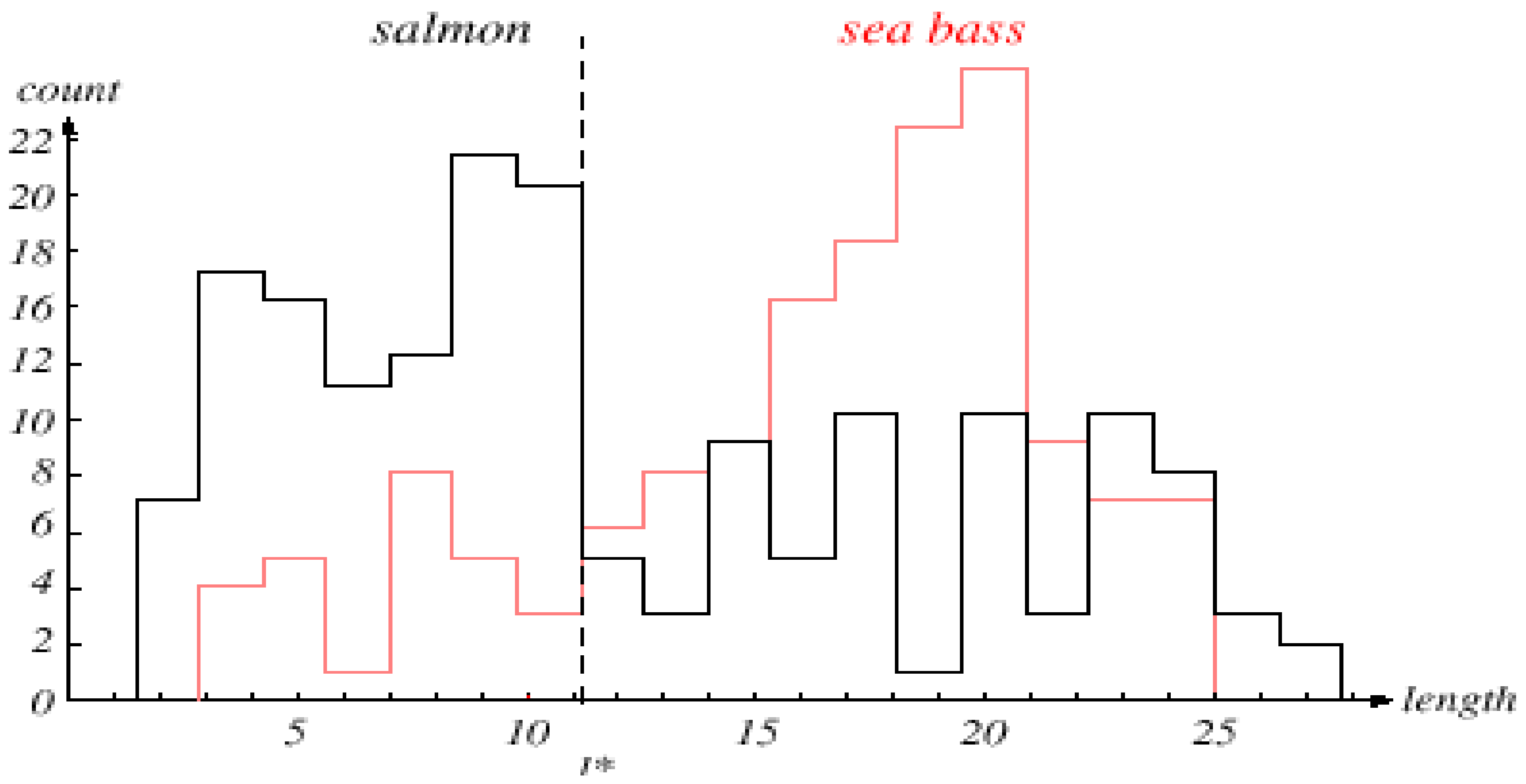
$$\begin{tikzcd} & \text{Sea bass} \\ \text{Species} \arrow[ru, no head] \arrow[rd, no head] & \\ & \text{Salmon} \end{tikzcd}$$

- Problem Analysis
  - Set up a camera and take some sample images to extract features
    - Length
    - Lightness
    - Width
    - Number and shape of fins
    - Position of the mouth, etc…
    - This is the set of all suggested features to explore for use in our classifier!

- Preprocessing
  - Use a segmentation operation to isolate fishes from one another and from the background
- Information from a single fish is sent to a feature extractor whose purpose is to reduce the data by measuring certain features
- The features are passed to a classifier

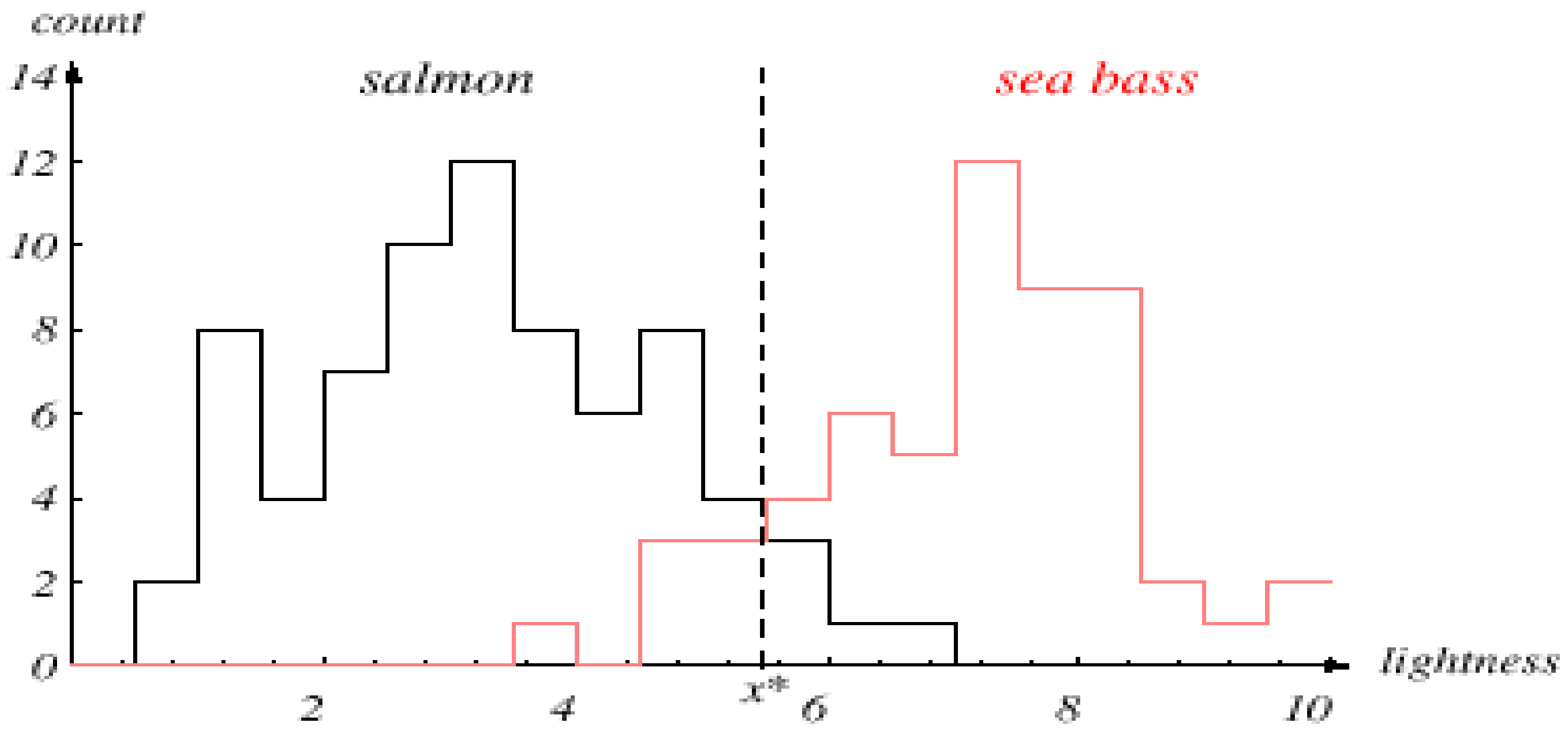

Preprocessing
Feature extraction
Classification
"salmon"
"sea bass"

- Classification
  - Select the length of the fish as a possible feature for discrimination


salmon
sea bass
count
22
20
18
16
12
10
8
6
4
2
0
5
10
15
20
25
l*
length

The length is a poor feature alone!

Select the lightness as a possible feature.


count
14
12
10
8
6
4
2
0
salmon
sea bass
lightness
2
4
x*
6
8
10

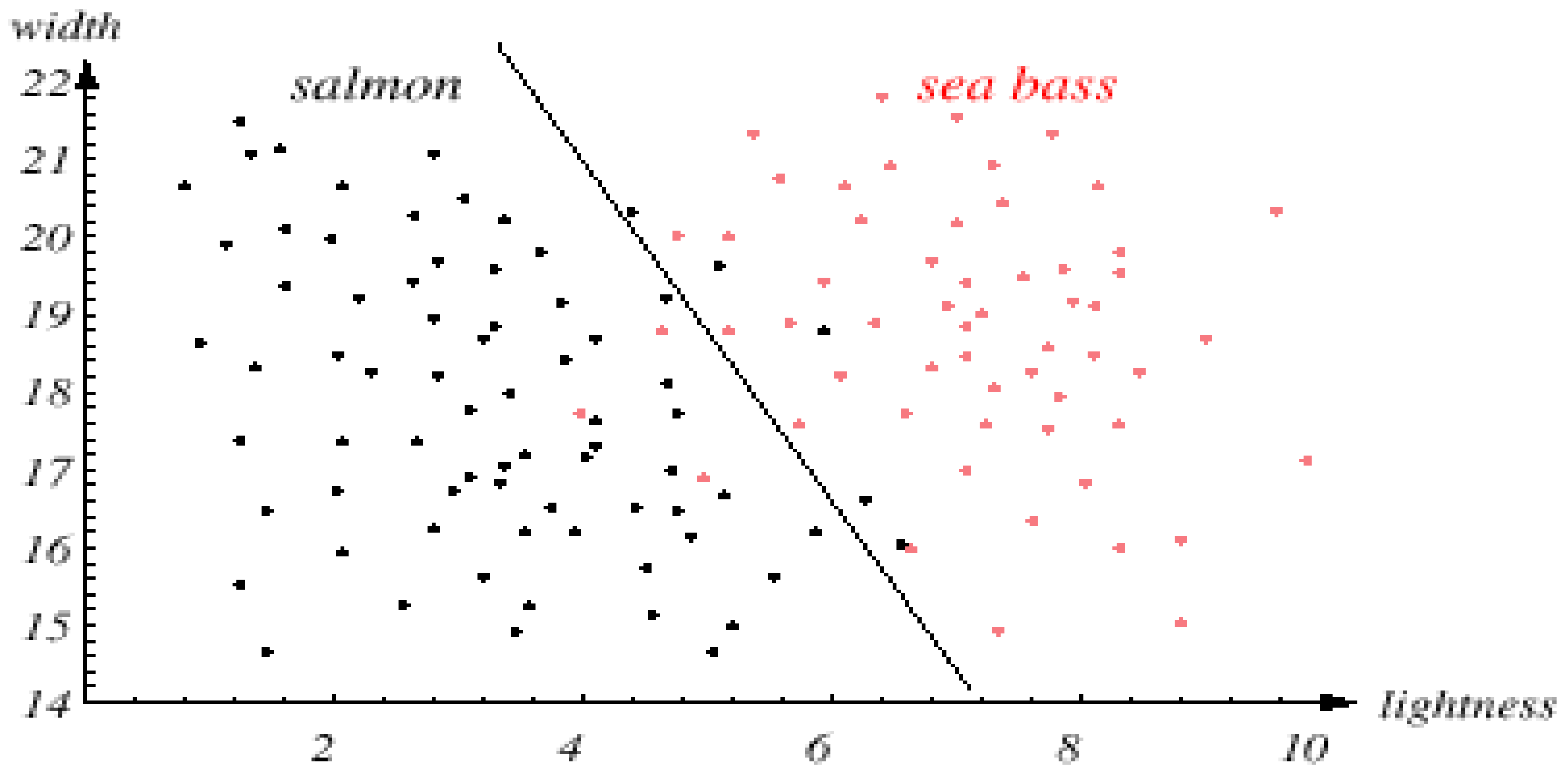
- Threshold decision boundary and cost relationship
  - Move our decision boundary toward smaller values of lightness in order to minimize the cost (reduce the number of sea bass that are classified salmon!)

$$\Downarrow$$

Task of decision theory

- Adopt the lightness and add the width of the fish

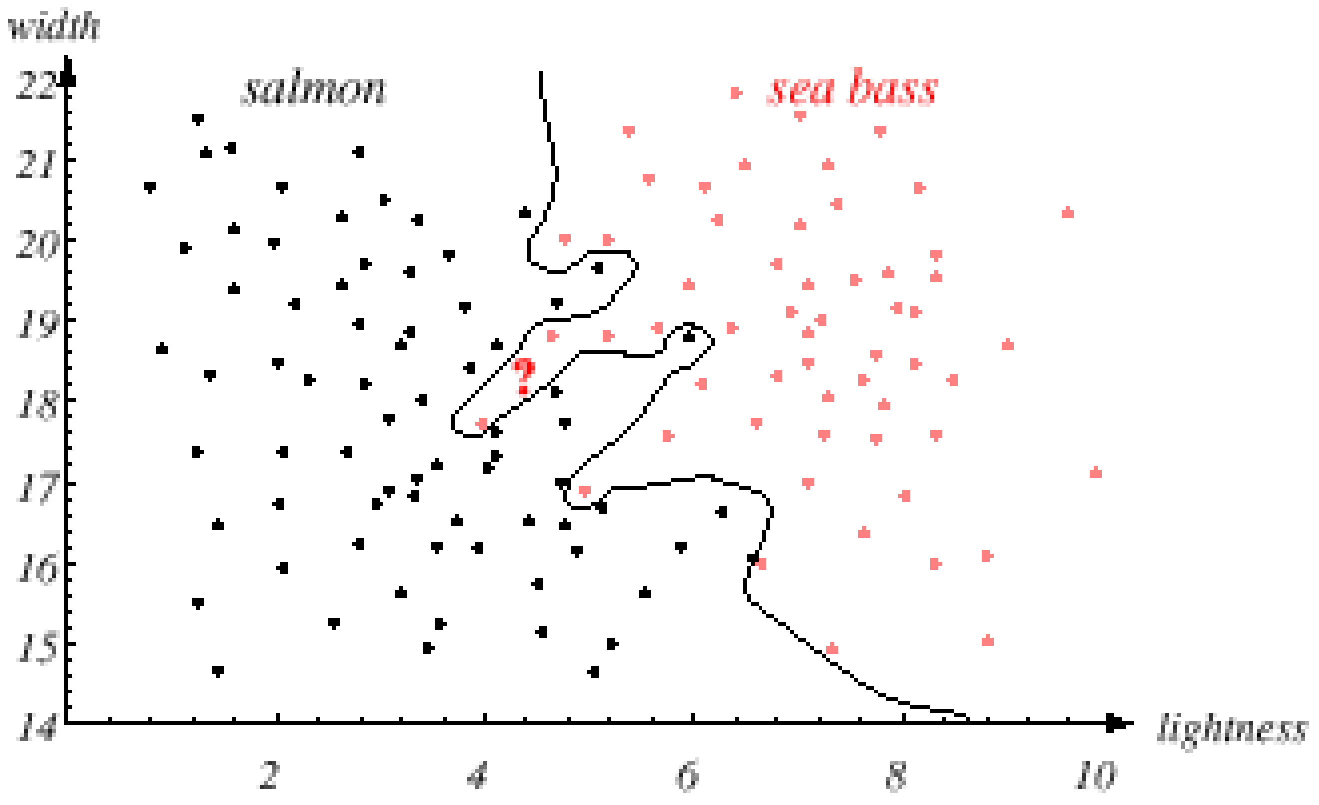
$$\begin{tikzcd}
\text{Fish} \arrow[r] & x^T = [x_1, x_2] \arrow[d] \arrow[dr] & \\
& \textbf{Lightness} & \textbf{Width}
\end{tikzcd}$$



The data is not linearly separable

- We might add other features that are not correlated with the ones we already have. A precaution should be taken not to reduce the performance by adding such “noisy features”

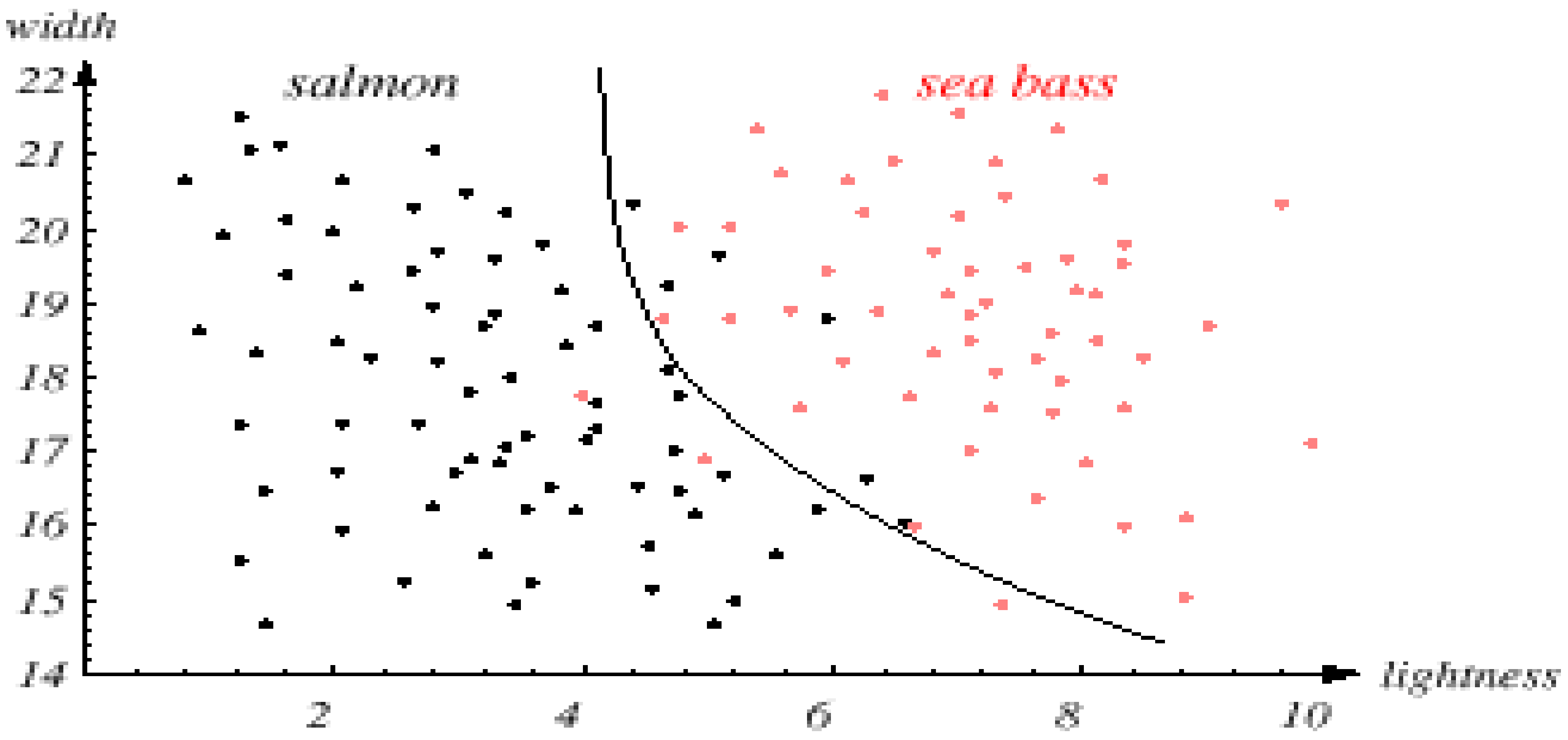
- Ideally, the best decision boundary should be the one which provides an optimal performance such as in the following figure:


width
salmon
sea bass
?
lightness
22
21
20
19
18
17
16
15
14
2
4
6
8
10

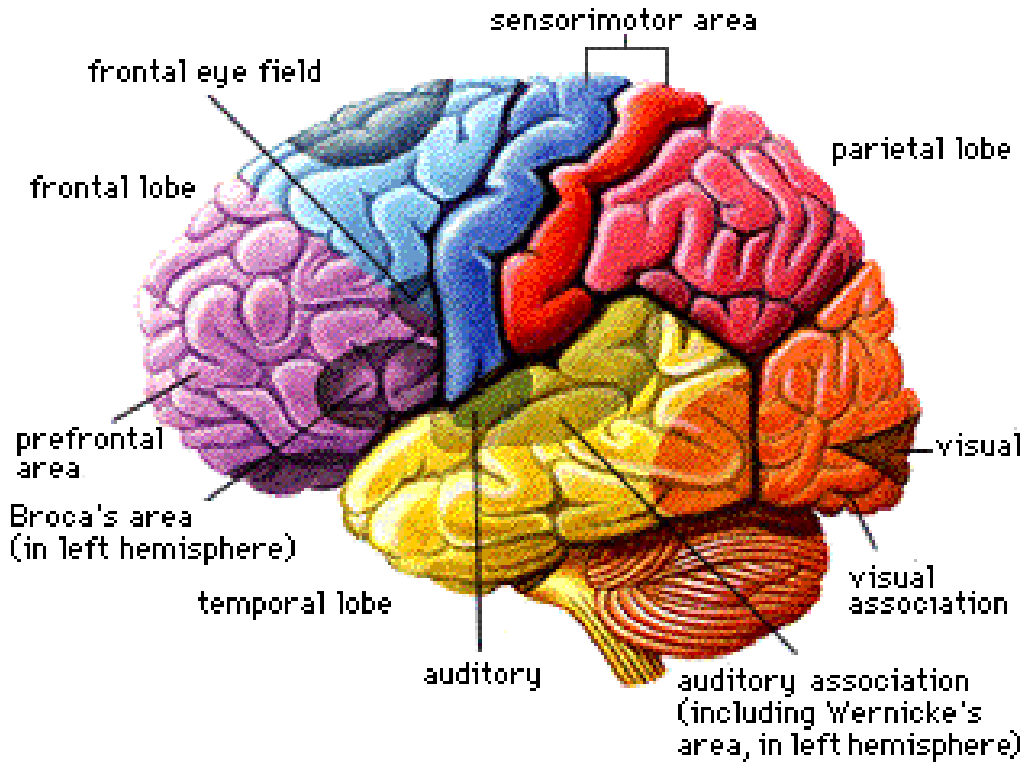
- However, our satisfaction is premature because the central aim of designing a classifier is to correctly classify novel input

$$\Downarrow$$

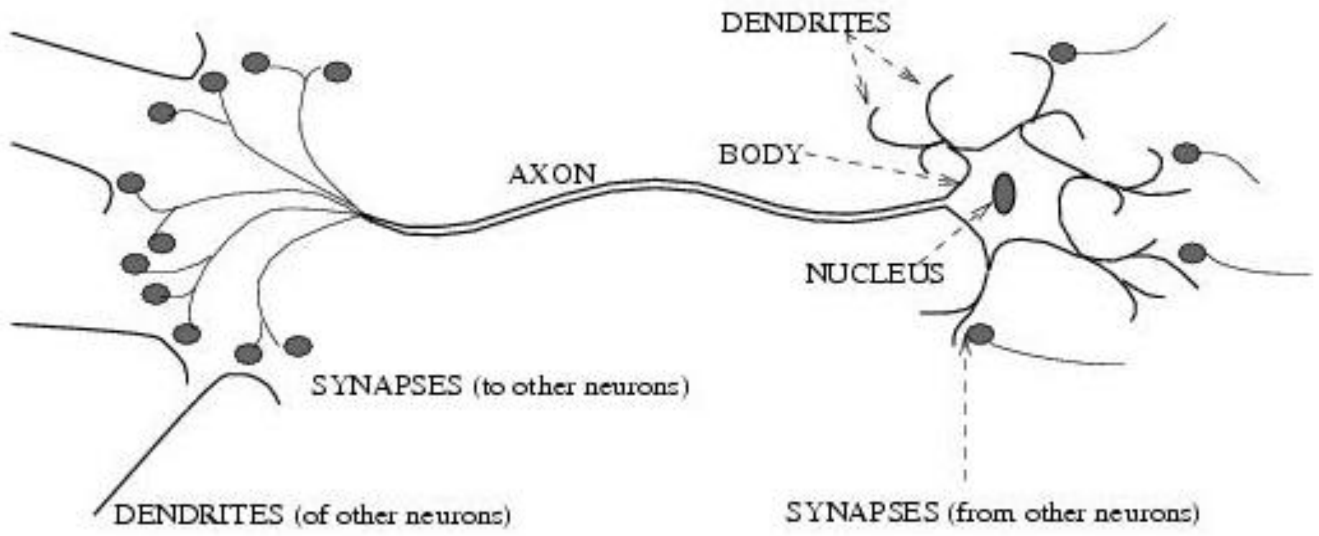
Issue of generalization!


width
salmon
sea bass
lightness
22
21
20
19
18
17
16
15
14
2
4
6
8
10

Neural computation – learning from the brain



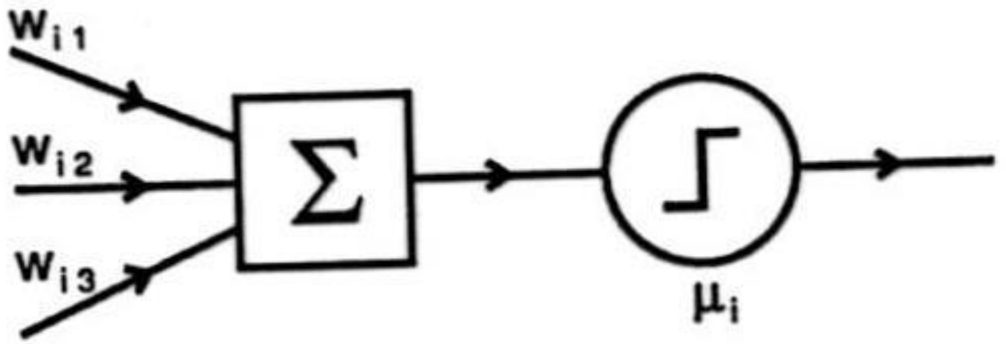
• Biological neural activity


DENDRITES
BODY
AXON
NUCLEUS
SYNAPSES (to other neurons)
DENDRITES (of other neurons)
SYNAPSES (from other neurons)

- Each neuron has a *body*, an *axon*, and many *dendrites*
  - Can be in one of the two states: *firing* and *rest.*
  - Neuron fires if the total incoming stimulus exceeds the threshold
- *Synapse*: thin gap between axon of one neuron and dendrite of another.
  - Signal exchange
  - Synaptic strength/efficiency

Mc-Cullock and Pitts neurons

$$n_i(t+1) = \Theta\left(\sum_j w_{ij} n_j(t) - \mu_i\right)$$



A single node like that (a Perceptron) can be used to represent linearly separable data

Introduction

ANN

- **Nodes**
  - input
  - output
  - node function
- **Connections**
  - connection strength

Bio NN

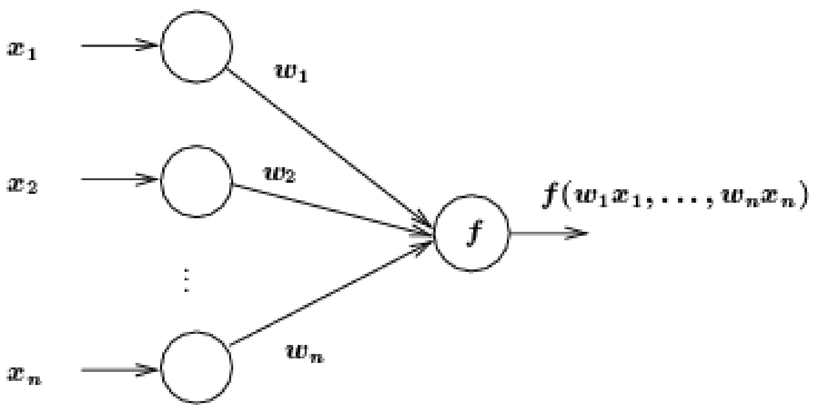
- **Cell body**
  - signal from other neurons
  - firing frequency
  - firing mechanism
- **Synapses**
  - synaptic strength

- Highly parallel, simple local computation (at neuron level) achieves global results as emerging property of the interaction (at network level)
- Pattern directed (meaning of individual nodes only in the context of a pattern)
- Fault-tolerant/graceful degrading
- Learning/adaptation plays important role.

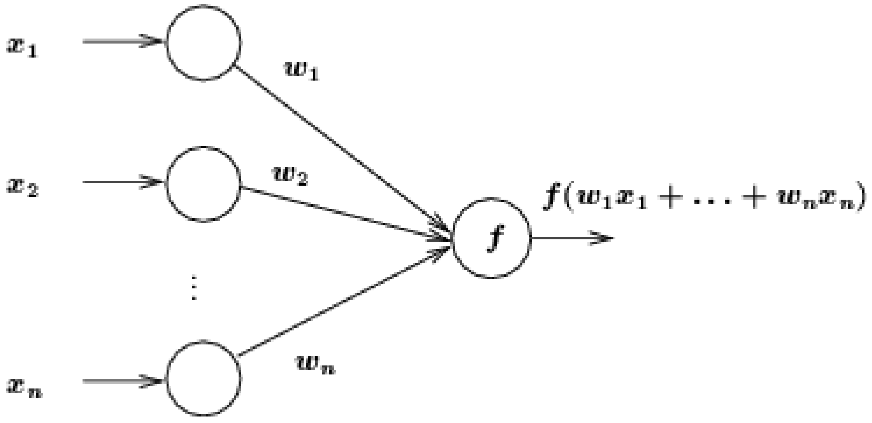
ANN Neuron Models : the Perceptron

- Each node has one or more inputs from other nodes, and one output to other nodes
- Input/output values can be
  - Binary {0, 1}
  - Bipolar {-1, 1}
  - Continuous
- All inputs to one node come in at the same time and remain activated until the output is produced
- Weights associated with links
- $f(net)$ is the node function

  $net = \sum_{i=1}^{n} w_i x_i$ is most popular



General neuron model



Weighted input summation

Node Function

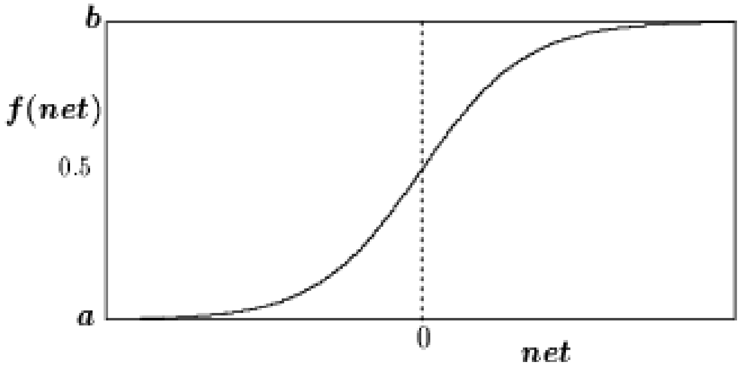
- **Sigmoid function**
  - S-shaped
  - Continuous and everywhere differentiable
  - Rotationally symmetric about some point (*net = c*)
  - Asymptotically approach saturation points

$$\lim_{\text{net}\to-\infty} f(\text{net}) = a \quad \lim_{\text{net}\to\infty} f(\text{net}) = b$$

  - Examples:

$$f(\text{net}) = z + \frac{1}{1 + \exp(-x \cdot \text{net} + y)}$$

$$f(\text{net}) = \tanh(x \cdot \text{net} - y) + z,$$



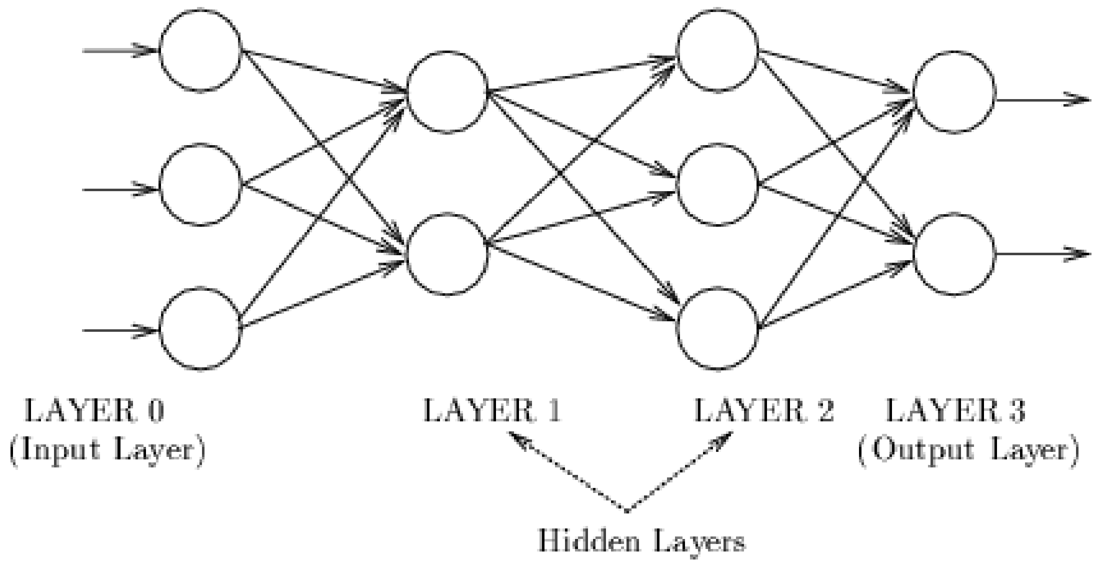
Sigmoid function

When $y = 0$ and $z = 0$:
$a = 0$, $b = 1$, $c = 0$.
When $y = 0$ and $z = -0.5$
$a = -0.5$, $b = 0.5$, $c = 0$.

Larger x gives steeper curve

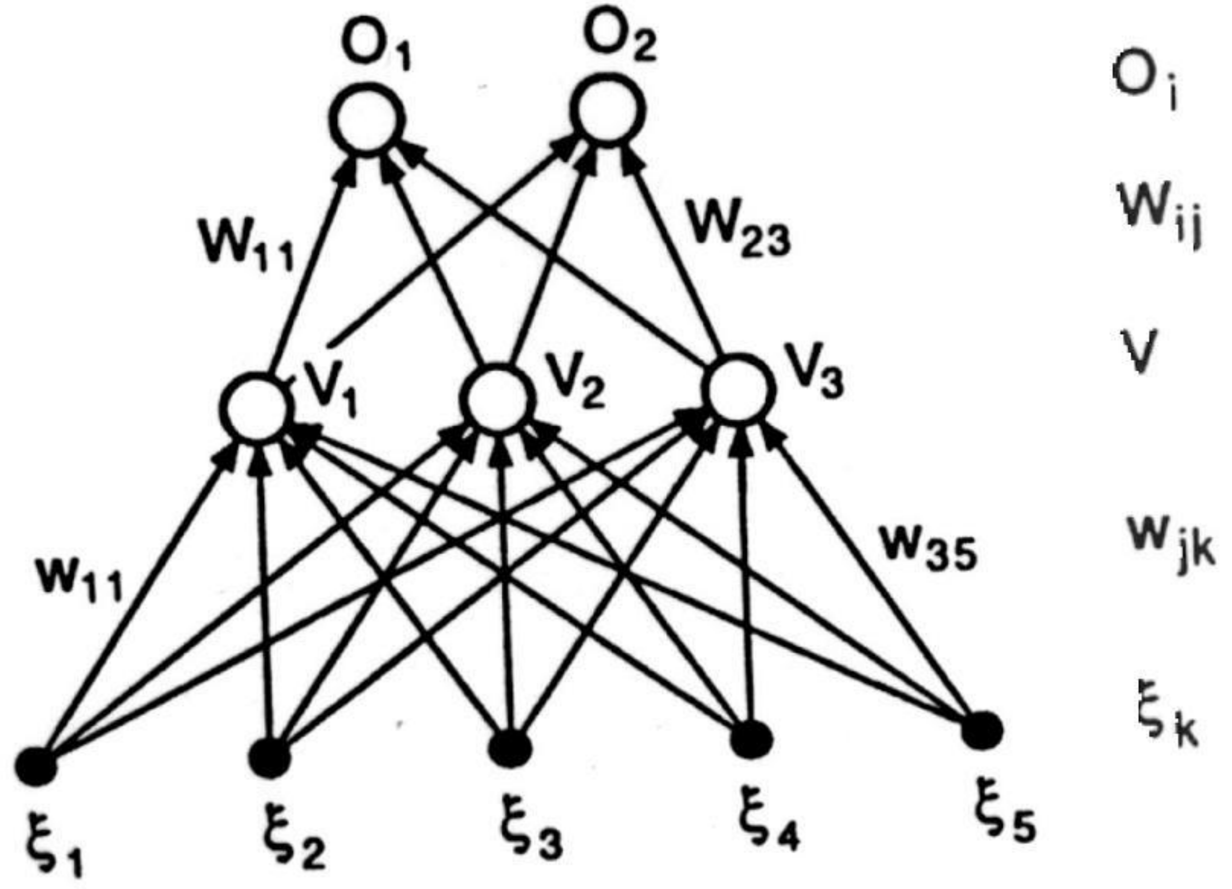
Network Architecture

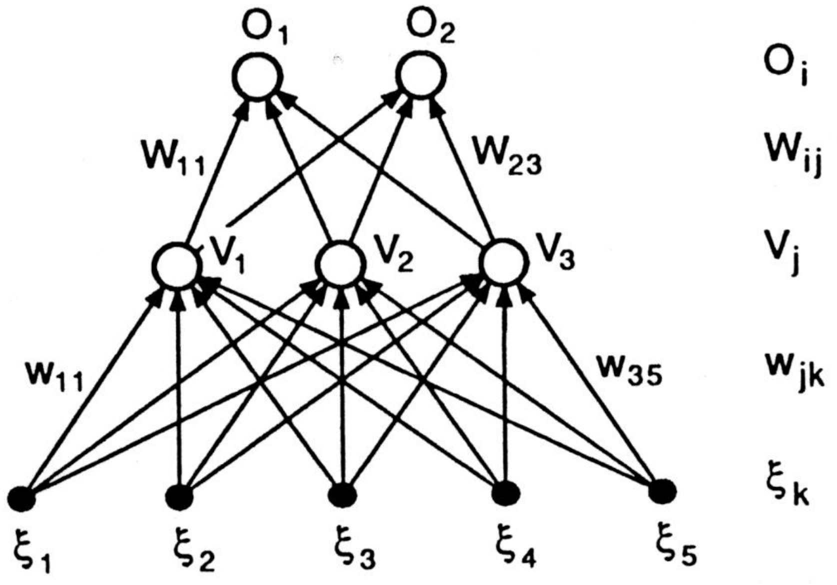
- **Feedforward Networks**
  - A connection is allowed from a node in layer *i* only to nodes in layer *i* + 1.
  - Most widely used architecture.



Conceptually, nodes at higher levels successively abstract features from preceding layers

MLP with sigmoid transfer-functions MLP=multi-layer perceptron=feed-forward network





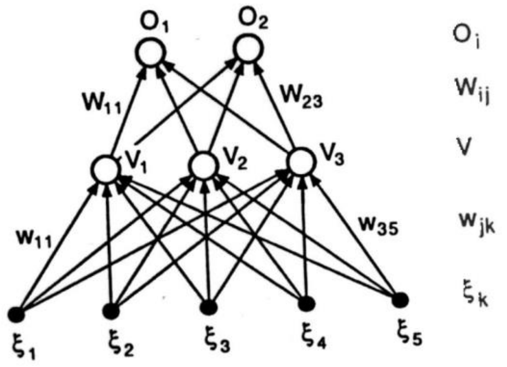
Given pattern μ, hidden unit j receives a net input

$$h_j^\mu = \sum_k w_{jk} \xi_k^\mu$$

and produces output

$$V_j^\mu = g(h_j^\mu) = g\Big(\sum_k w_{jk} \xi_k^\mu\Big) .$$

ξ_1 ξ_2 ξ_3 ξ_4 ξ_5



Given pattern μ, hidden unit j receives a net input

$$h_j^\mu = \sum_k w_{jk}\xi_k^\mu$$

and produces output

$$V_j^\mu = g(h_j^\mu) = g\Big(\sum_k w_{jk}\xi_k^\mu\Big).$$

Output unit i thus receives

$$h_i^\mu = \sum_j W_{ij}V_j^\mu = \sum_j W_{ij}g\Big(\sum_k w_{jk}\xi_k^\mu\Big)$$

and produces for the final output

$$O_i^\mu = g(h_i^\mu) = g\Big(\sum_j W_{ij}V_j^\mu\Big) = g\Big(\sum_j W_{ij}g\Big(\sum_k w_{jk}\xi_k^\mu\Big)\Big).$$

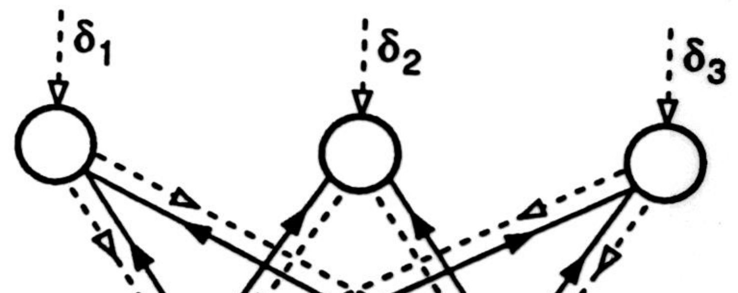
$$E[\mathbf{w}] = \frac{1}{2}\sum_{\mu i}[\zeta_i^\mu - O_i^\mu]^2$$

LMS

$$E[\mathbf{w}] = \frac{1}{2}\sum_{\mu i}\Big[\zeta_i^\mu - g\Big(\sum_j W_{ij}\, g\Big(\sum_k w_{jk}\xi_k^\mu\Big)\Big)\Big]^2.$$

$$\begin{aligned}
\Delta w_{jk} &= -\eta\frac{\partial E}{\partial w_{jk}} = -\eta\sum_\mu \frac{\partial E}{\partial V_j^\mu}\frac{\partial V_j^\mu}{\partial w_{jk}} \\
&= \eta\sum_{\mu i}[\zeta_i^\mu - O_i^\mu]g'(h_i^\mu)W_{ij}\,g'(h_j^\mu)\xi_k^\mu \\
&= \eta\sum_{\mu i}\delta_i^\mu W_{ij}\,g'(h_j^\mu)\xi_k^\mu
\end{aligned}$$

Back Propagation



Our usual error measure or cost function

$$E[\mathbf{w}] = \frac{1}{2}\sum_{\mu i}[\zeta_i^\mu - O_i^\mu]^2 \tag{6.5}$$

now becomes

$$E[\mathbf{w}] = \frac{1}{2}\sum_{\mu i}\Big[\zeta_i^\mu - g\Big(\sum_j W_{ij}\, g\Big(\sum_k w_{jk}\xi_k^\mu\Big)\Big)\Big]^2 . \tag{6.6}$$

This is clearly a continuous differentiable function of every weight, so we can use a gradient descent algorithm to learn appropriate weights. In one sense this is all there is to back-propagation, but there is great practical importance in the form of the resulting update rules.

For the hidden-to-output connections the gradient descent rule gives

$$\begin{aligned}\Delta W_{ij} &= -\eta\frac{\partial E}{\partial W_{ij}} = \eta\sum_\mu[\zeta_i^\mu - O_i^\mu]g'(h_i^\mu)V_j^\mu \\ &= \eta\sum_\mu \delta_i^\mu V_j^\mu\end{aligned} \tag{6.7}$$

where we have defined

$$\delta_i^\mu = g'(h_i^\mu)[\zeta_i^\mu - O_i^\mu] . \tag{6.8}$$

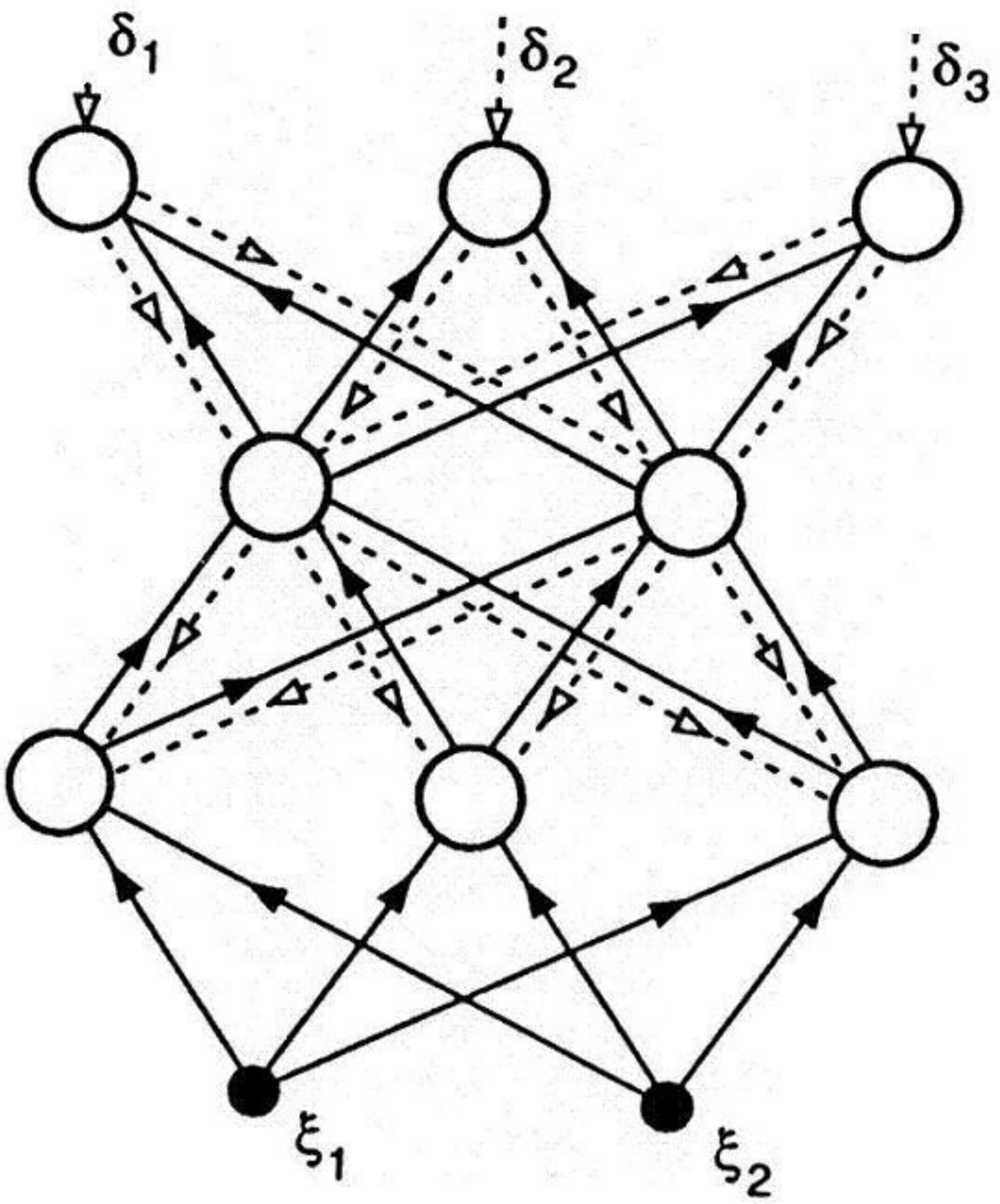
The result is of course identical to that obtained earlier (equations (5.50) and (5.51)) for a single layer perceptron, with the output V_j^μ of the hidden units now playing the role of the perceptron input.

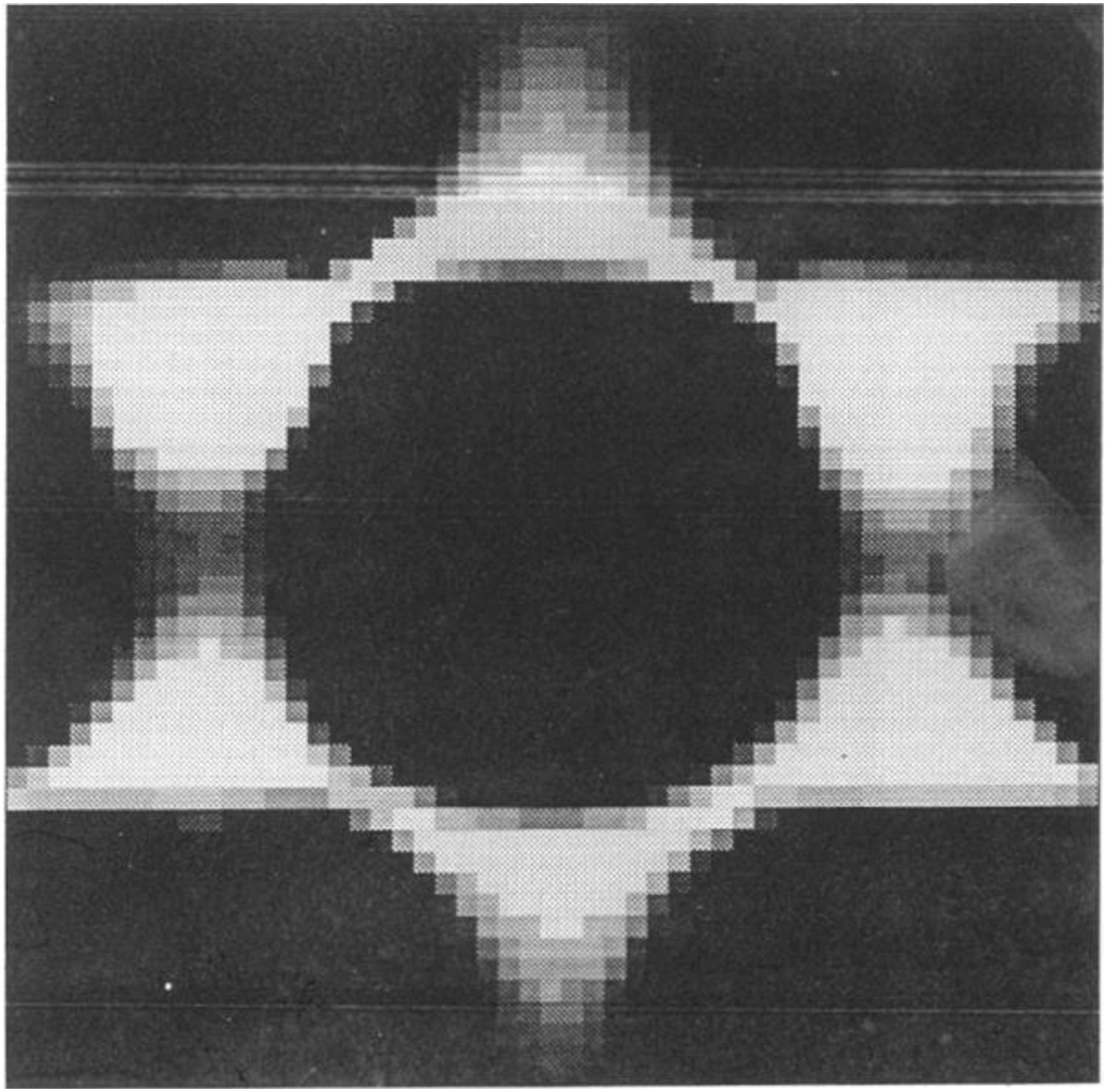
For the input-to-hidden connections Δw_{jk} we must differentiate with respect to the w_{jk}'s, which are more deeply embedded in (6.6). Using the chain rule, we obtain

$$\begin{aligned}\Delta w_{jk} &= -\eta\frac{\partial E}{\partial w_{jk}} = -\eta\sum_\mu \frac{\partial E}{\partial V_j^\mu}\frac{\partial V_j^\mu}{\partial w_{jk}} \\ &= \eta\sum_{\mu i}[\zeta_i^\mu - O_i^\mu]g'(h_i^\mu)W_{ij}g'(h_j^\mu)\xi_k^\mu \\ &= \eta\sum_{\mu i}\delta_i^\mu W_{ij}g'(h_j^\mu)\xi_k^\mu \\ &= \eta\sum_\mu \delta_j^\mu \xi_k^\mu\end{aligned} \tag{6.9}$$

with

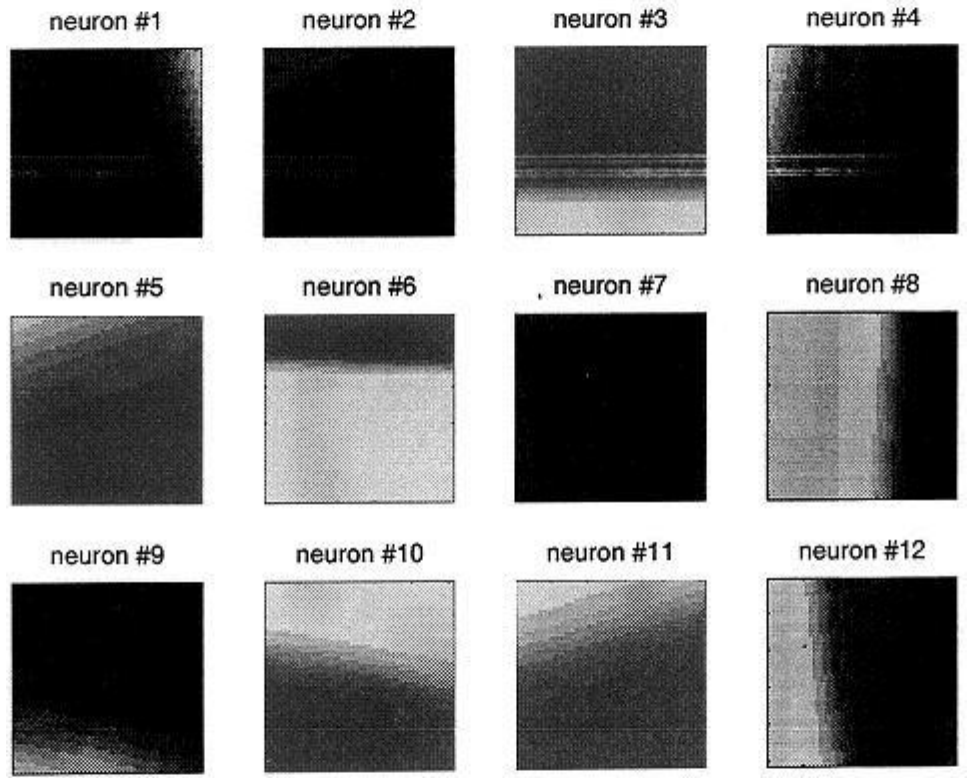
$$\delta_j^\mu = g'(h_j^\mu)\sum_i W_{ij}\delta_i^\mu . \tag{6.10}$$


δ_1
δ_2
δ_3
ξ_1
ξ_2

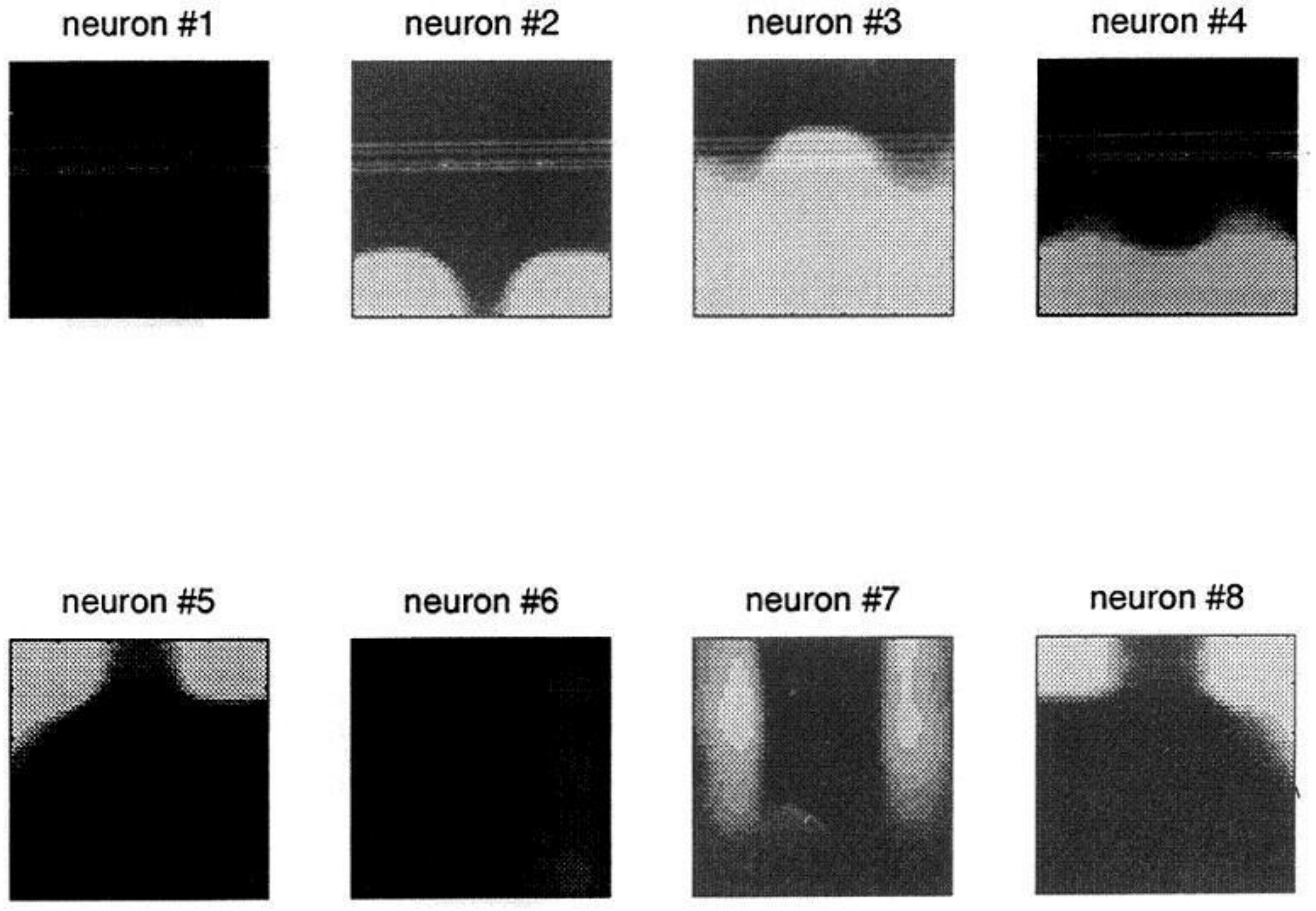


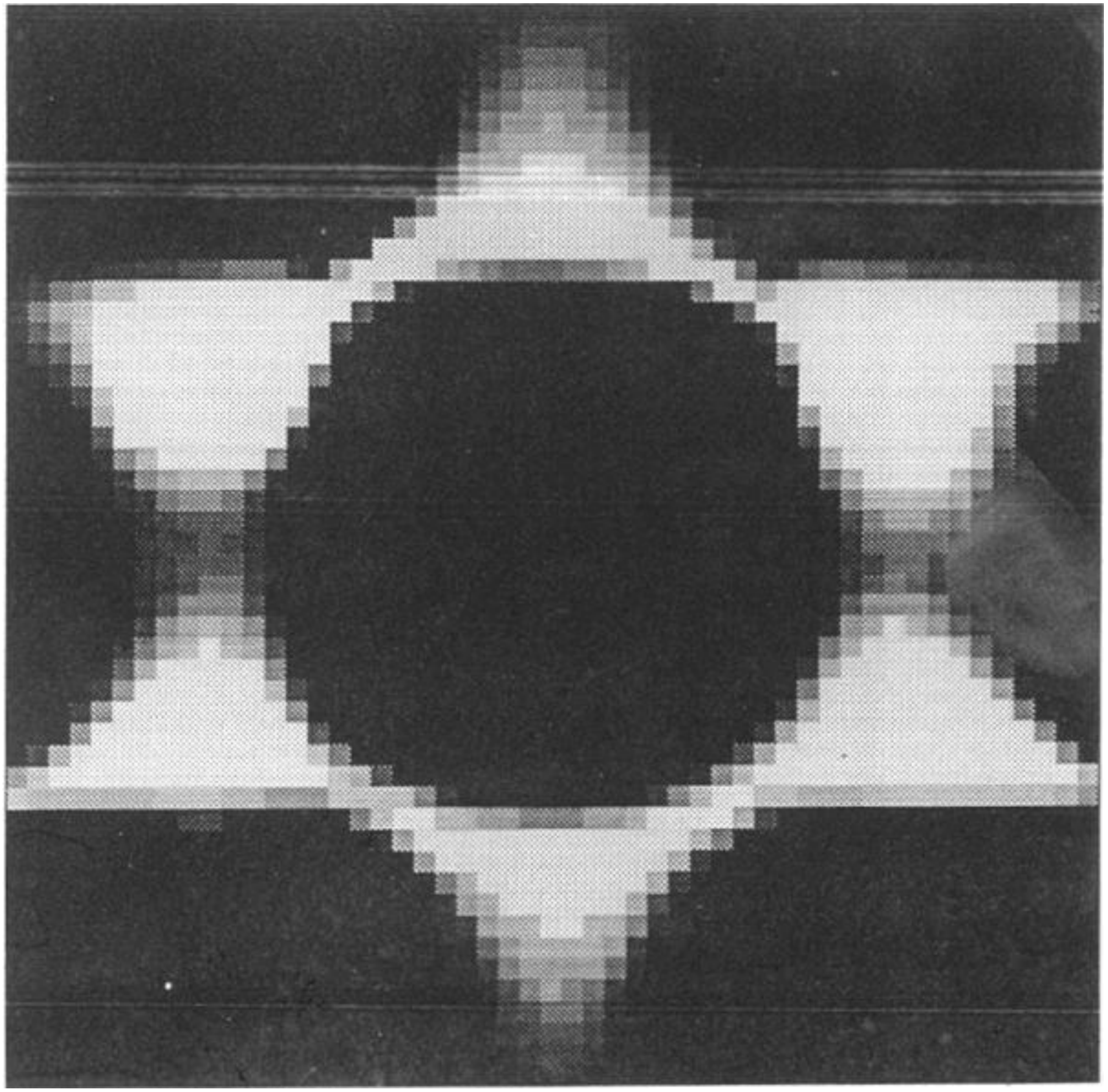
Neural network: 2(x,y)->12->8->1(grey-scale)

First hidden layer – the resulting ‘receptive fields’

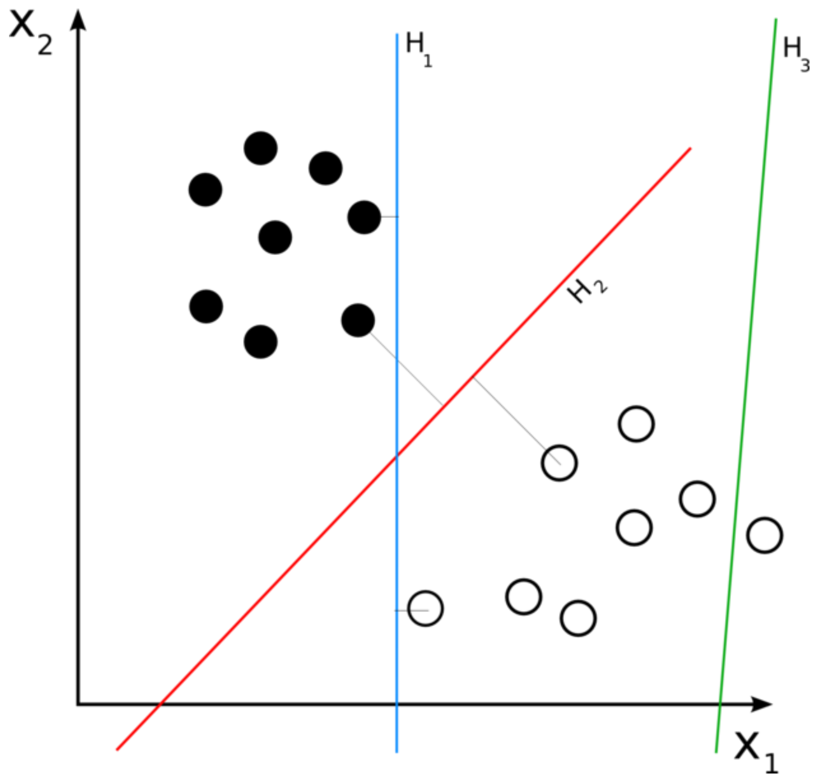


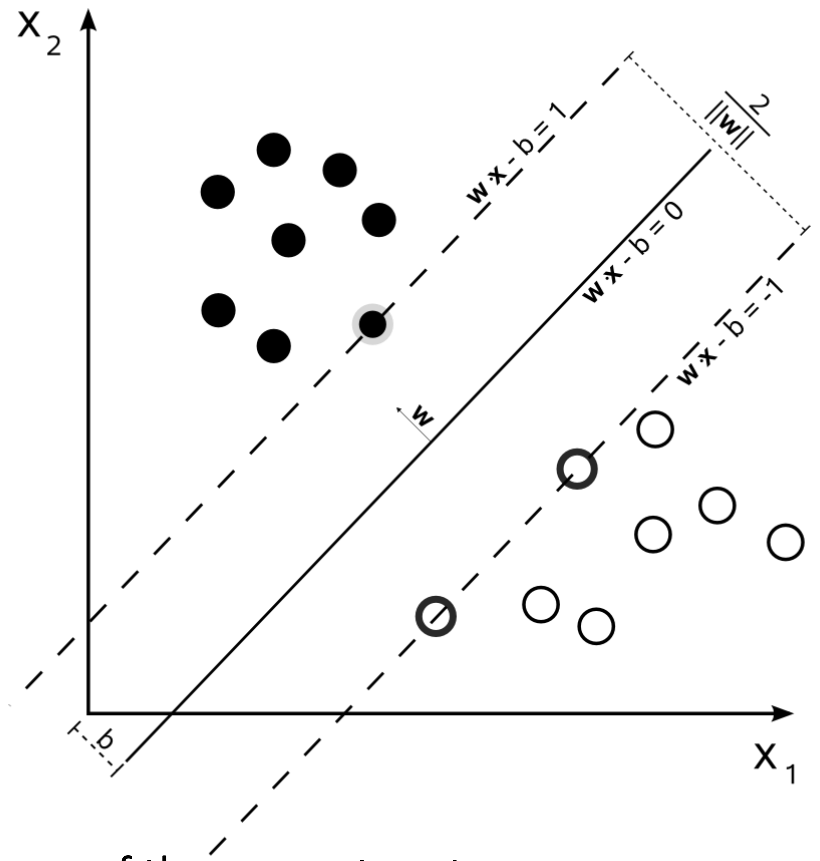
The second hidden layer





Support Vector Machine (SVM)




X_2
$\mathbf{w}\cdot\mathbf{x}-b=1$
$\frac{2}{\|\mathbf{w}\|}$
$\mathbf{w}\cdot\mathbf{x}-b=0$
$\mathbf{w}\cdot\mathbf{x}-b=-1$
$\mathbf{w}$
b
X_1

Find maximal-margin plane. Express it in terms of the support vectors.

Data points are vectors x with labels y=1 or -1

Formulation as an Optimization Problem

Hyperplane with maximum margin: minimize

$$||\mathbf{w}||^2$$

(recall: margin $\sim 1/||\mathbf{w}||$) subject to

$$y_i \cdot [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] \geq 1 \quad \text{for } i = 1 \dots m$$

(i.e. the training data are separated correctly).

Lagrange Function (e.g., [6])

Introduce Lagrange multipliers $\alpha_i \geq 0$ and a Lagrangian

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i \left(y_i \cdot [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] - 1 \right).$$

L has to minimized w.r.t. the *primal variables* $\mathbf{w}$ and b and maximized with respect to the *dual variables* α_i

- if a constraint is violated, then $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 < 0 \longrightarrow$
  - α_i will grow to increase L — how far?
  - $\mathbf{w}$, b want to decrease L; i.e. they have to change such that the constraint is satisfied. If the problem is separable, this ensures that $\alpha_i < \infty$.
- similarly: if $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 > 0$, then $\alpha_i = 0$: otherwise, L could be increased by decreasing α_i *(KKT conditions)*

Derivation of the Dual Problem

At the extremum, we have

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \quad \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0,$$

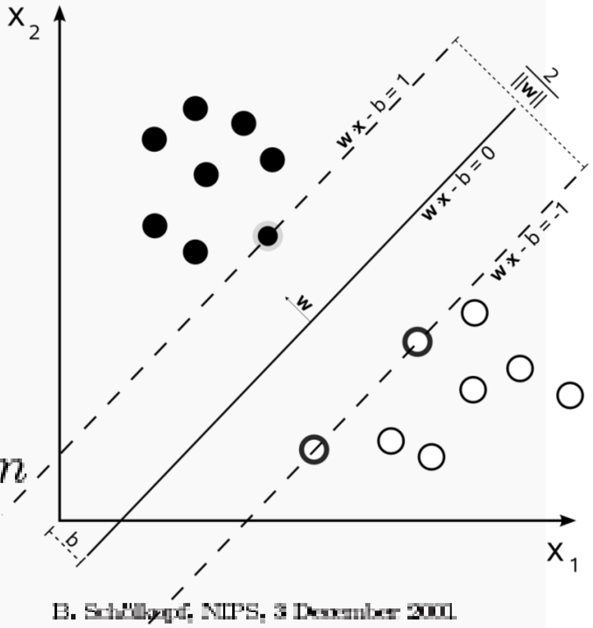
i.e.

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

and

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i.$$

Substitute both into L to get the *dual problem*



Dual Problem

Dual: maximize

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

subject to

$$\alpha_i \geq 0, \quad i = 1, \dots, m, \quad \text{and} \quad \sum_{i=1}^{m} \alpha_i y_i = 0.$$

Both the final decision function and the function to be maximized are expressed in dot products $\longrightarrow$ can use a kernel to compute

$$\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \langle \Phi(x_i), \Phi(x_j) \rangle = k(x_i, x_j).$$

Nonseparable Problems

[4, 15]

If $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$ cannot be satisfied, then $\alpha_i \to \infty$.

Modify the constraint to

$$y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$$

with

$$\xi_i \geq 0$$

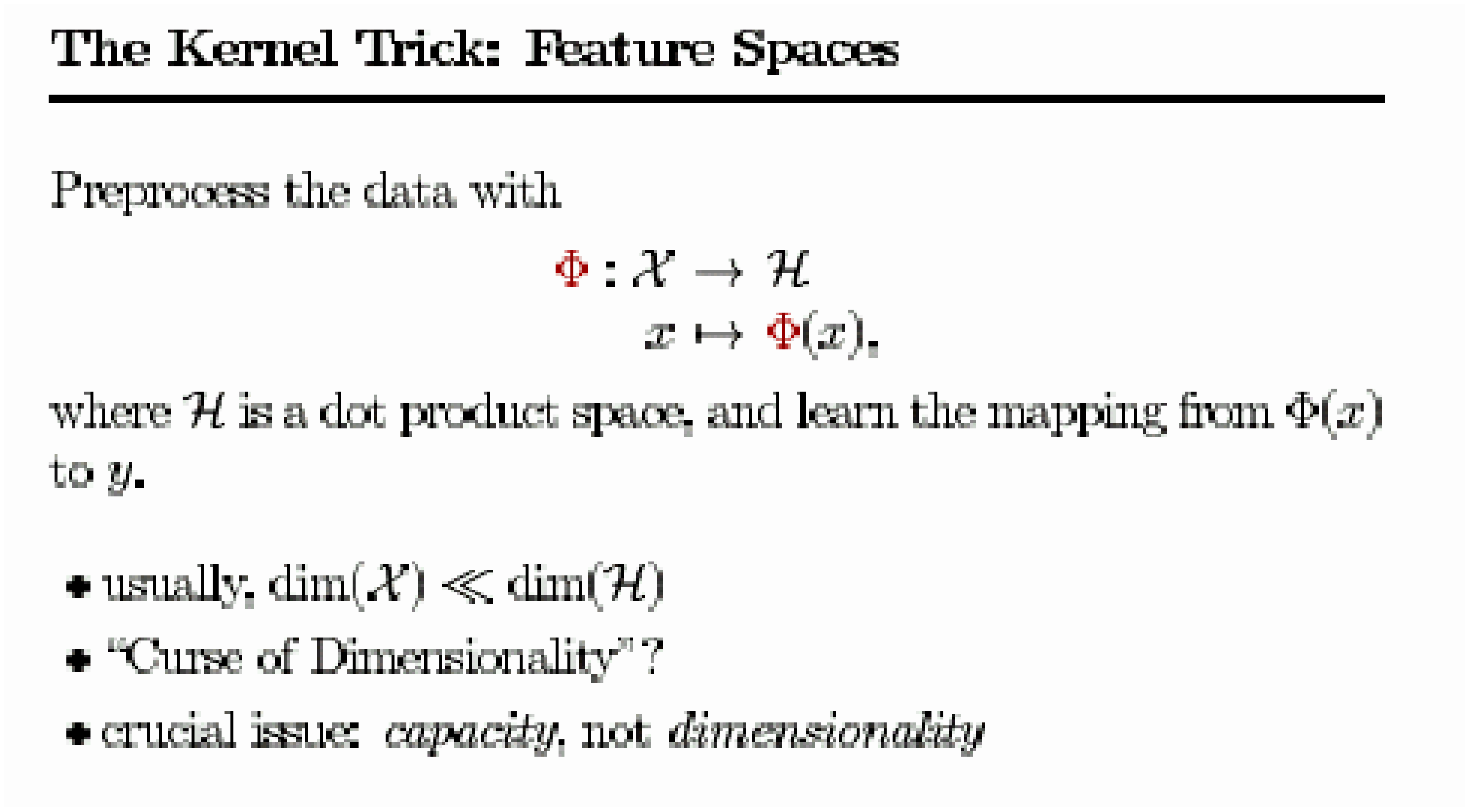
("*soft margin*") and add

$$C \cdot \sum_{i=1}^{m} \xi_i$$

in the objective function.

Same dual, with additional constraints $\alpha_i \leq C$.

Expanding SVM onto Hilbert space



The Kernel Trick: Feature Spaces

Preprocess the data with

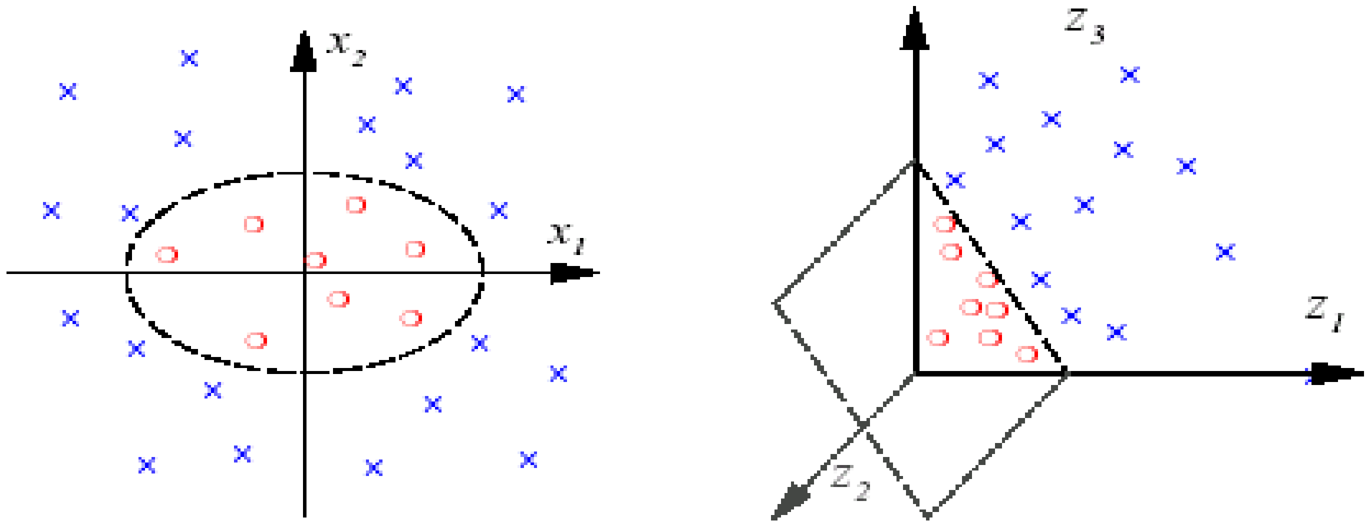
$$\begin{aligned} \Phi : \mathcal{X} &\to \mathcal{H} \\ x &\mapsto \Phi(x), \end{aligned}$$

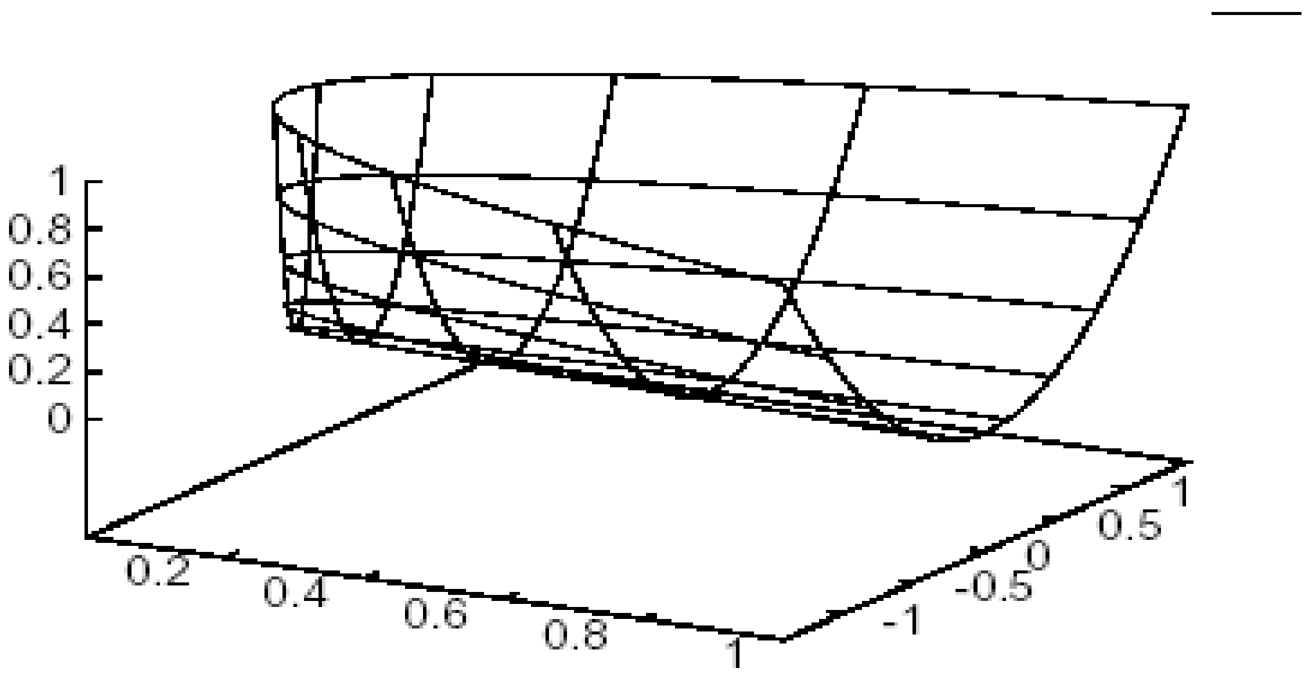
where $\mathcal{H}$ is a dot product space, and learn the mapping from $\Phi(x)$ to y.

- usually, $\dim(\mathcal{X}) \ll \dim(\mathcal{H})$
- "Curse of Dimensionality"?
- crucial issue: *capacity*, not *dimensionality*

Example: All Degree 2 Monomials

$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$

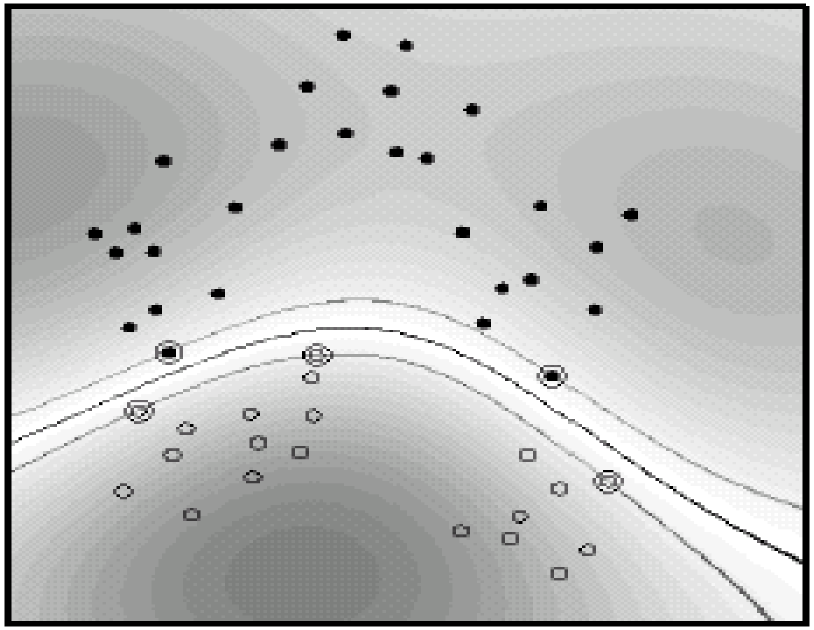




igure 8. Image, in $\mathcal{H}$, of the square $[-1,1] \times [-1,1] \in \mathbf{R}^2$ under the mapping Φ.

Toy Example with Gaussian Kernel

$$k(x, x') = \exp\left(-\|x - x'\|^2\right)$$



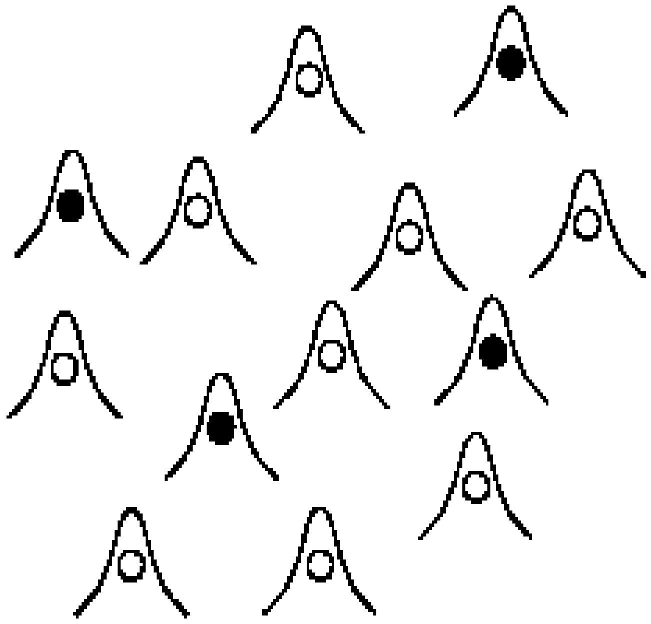
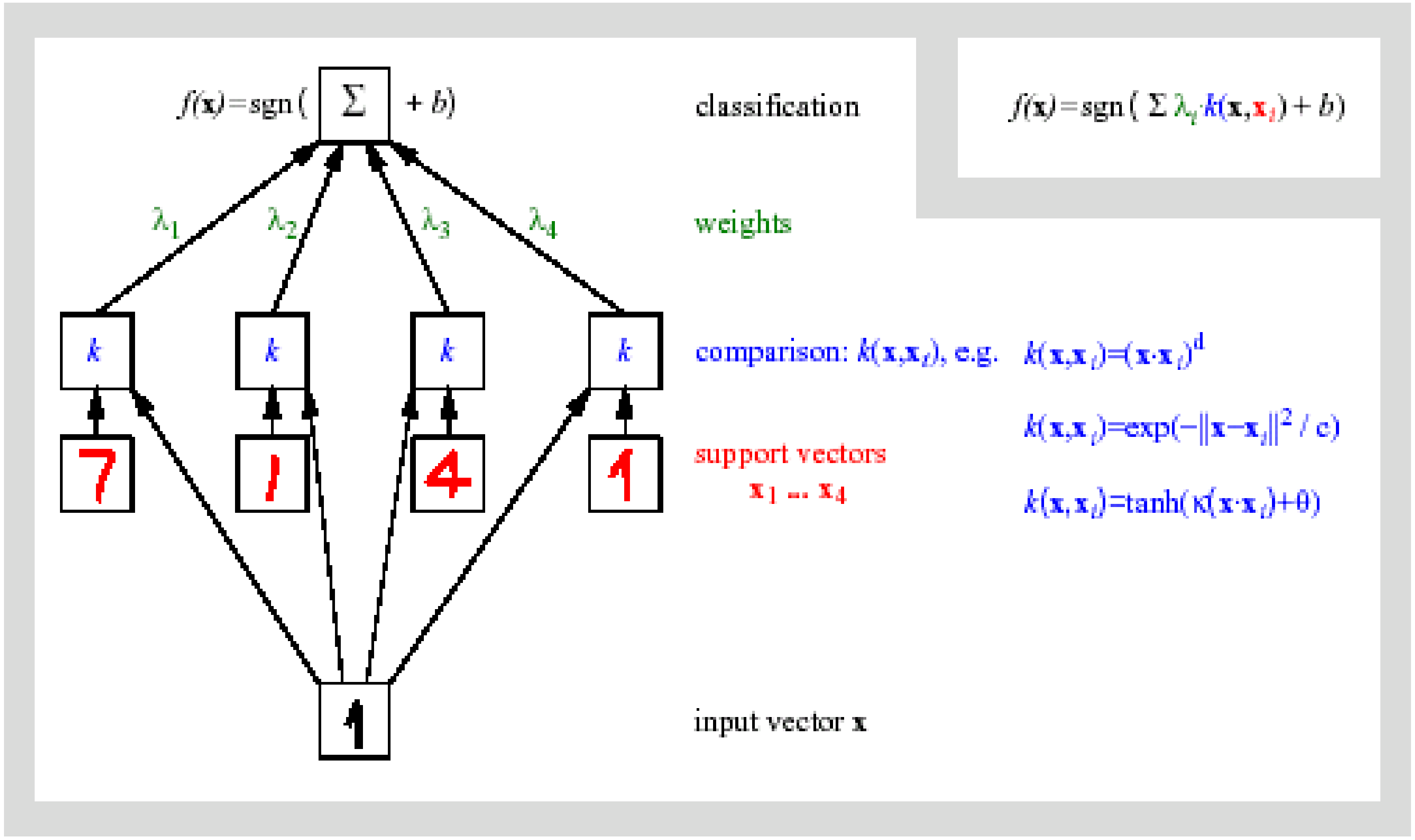


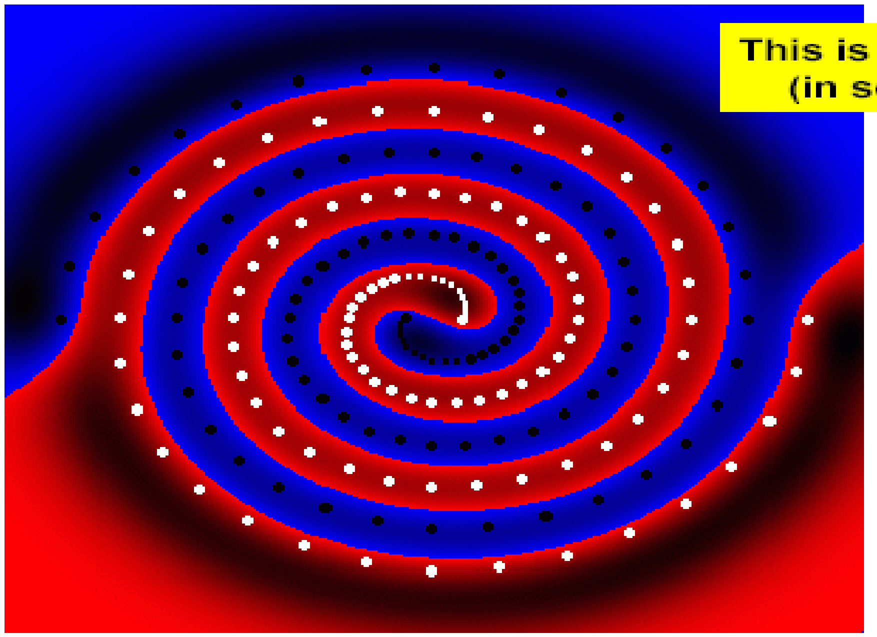
Figure 11. Gaussian RBF SVMs of sufficiently small width can classify an arbitrarily large number of training points correctly, and thus have infinite VC dimension

The SVM Architecture



$f(\mathbf{x}) = \mathrm{sgn}(\ \Sigma\ + b)$
classification
$f(\mathbf{x}) = \mathrm{sgn}(\ \Sigma\ \lambda_i \cdot k(\mathbf{x}, \mathbf{x}_i) + b)$
λ_1 λ_2 λ_3 λ_4
weights
k k k k
comparison: $k(\mathbf{x}, \mathbf{x}_i)$, e.g.
$k(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x} \cdot \mathbf{x}_i)^d$
$k(\mathbf{x}, \mathbf{x}_i) = \exp(-\|\mathbf{x} - \mathbf{x}_i\|^2 / c)$
$k(\mathbf{x}, \mathbf{x}_i) = \tanh(\kappa(\mathbf{x} \cdot \mathbf{x}_i) + \theta)$
support vectors
$\mathbf{x}_1 \ldots \mathbf{x}_4$
input vector $\mathbf{x}$

Flexibility of SVMs…



This is a hyperplane!
(in some space)

www.support-vector.net/nello.html

Support Vector Clustering

Given points x in data space, define images in Hilbert space.

Require all images to be enclosed by a minimal sphere in Hilbert space.

Reflection of this sphere in data space defines cluster boundaries.

Two parameters: width of Gaussian kernel and fraction of outliers

Ben-Hur, Horn, Siegelmann & Vapnik. JMLR 2 (2001) 125-127

An enclosing sphere is defined by:

$$||\Phi(\mathbf{x}_j) - \mathbf{a}||^2 \leq R^2$$

Φ - map into feature space

$\mathbf{a}$: center of the sphere.

- Goal: minimize R^2 over all choices of $\mathbf{a}$ using the Largragian:

$$L = R^2 - \sum_j (R^2 - ||\Phi(\mathbf{x}_j) - \mathbf{a}||^2)\beta_j$$

β_j Lagrange multiplier

Derivatives with respect to R and $\mathbf{a}$:

$$\sum_j \beta_j = 1$$

$$\mathbf{a} = \sum_j \beta_j \Phi(\mathbf{x}_j)$$

The KKT complementarity conditions:

$$(R^2 - ||\Phi(\mathbf{x}_j) - \mathbf{a}||^2)\beta_j = 0$$

- $\beta_j \neq 0 \;\; \Rightarrow \;\; R^2 - ||\Phi(\mathbf{x}_j) - \mathbf{a}||^2 = 0$

Points with $\beta_j \neq 0$ are on the surface of the sphere (support vectors).

- Wolfe dual form:

$$W = \sum_j \Phi(\mathbf{x}_j)^2 \beta_j - \sum_{i,j} \beta_i \beta_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

with the constraints $\sum \beta_j = 1$

- The SV trick: represent the dot products by a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$

Lagrangian now becomes:

$$W = \sum_j K(\mathbf{x}_j, \mathbf{x}_j)\beta_j - \sum_{i,j} \beta_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j).$$

- No need to know the specific form of Φ.

$$R = \{R(\mathbf{x}_i) \mid \mathbf{x}_i \text{ is a support vector }\}$$

The enclosing contour: $\{\mathbf{x} \mid R(\mathbf{x}) = R\}$

The shape of contour governed by the kernel parameter:

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-q||\mathbf{x}_i - \mathbf{x}_j||^2}$$

- As q increases the contour becomes a tighter fit; for certain values of q observe splitting.
- Need to identify the different components.
- Complexity: $O(N^2 D)$

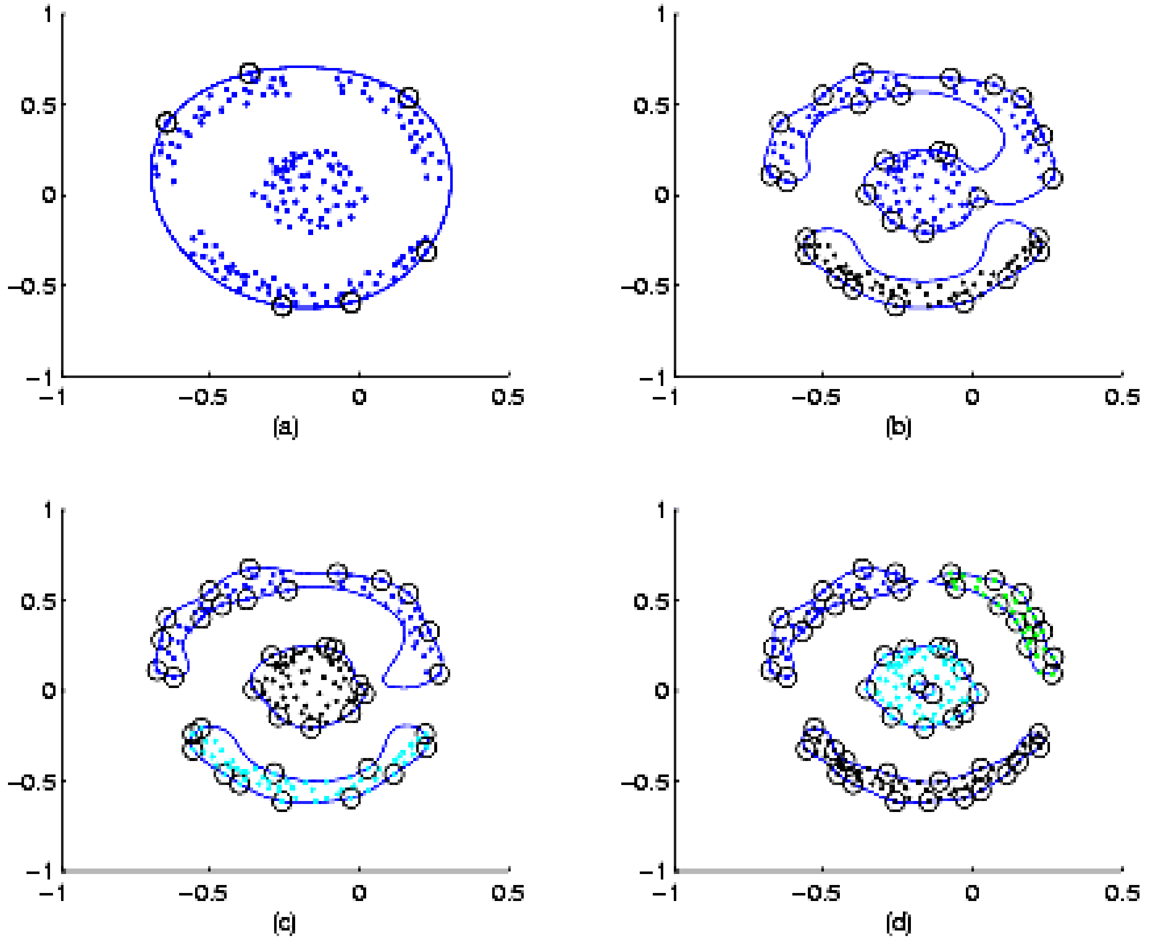
Variation of q allows for clustering solutions on various scales

q=1,

20,

24,

48



Clustering

Introduction to clustering

Dimensional reduction by SVD

Quantum Clustering

Dynamical Quantum Clustering

A few concepts from machine learning

Classification – supervised learning

Data are **labeled** vectors in feature space

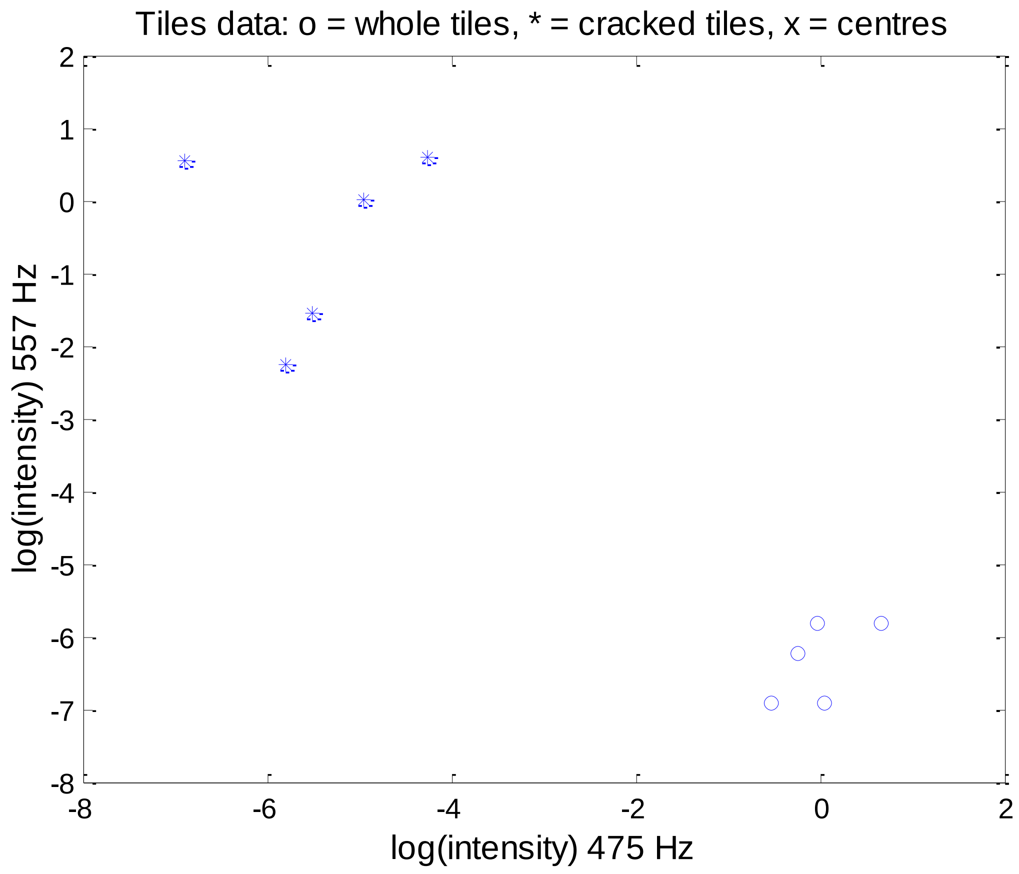
Clustering – unsupervised learning

Conventionally based on **distances** between data

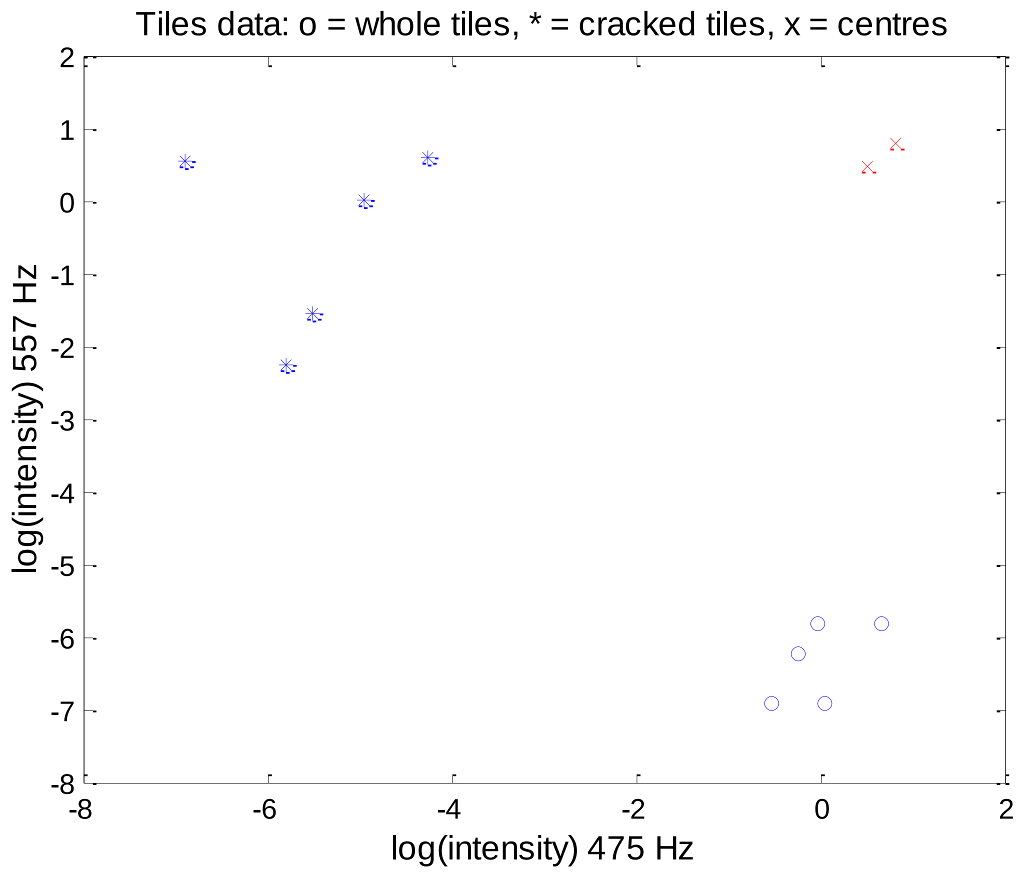
- Exclusive vs. Overlapping Clustering
- Hierarchical vs. Global Clustering
- Formal vs. Heuristic Clustering

  Example:

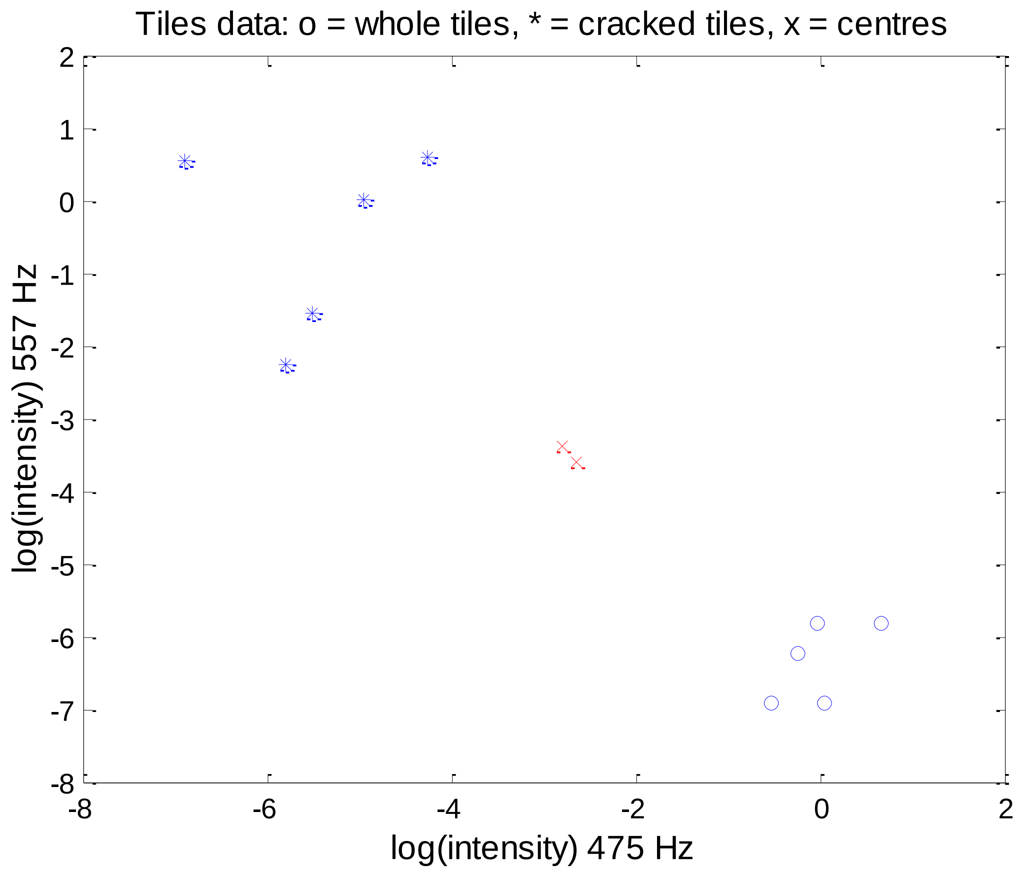
  K-Means: exclusive, global, heuristic



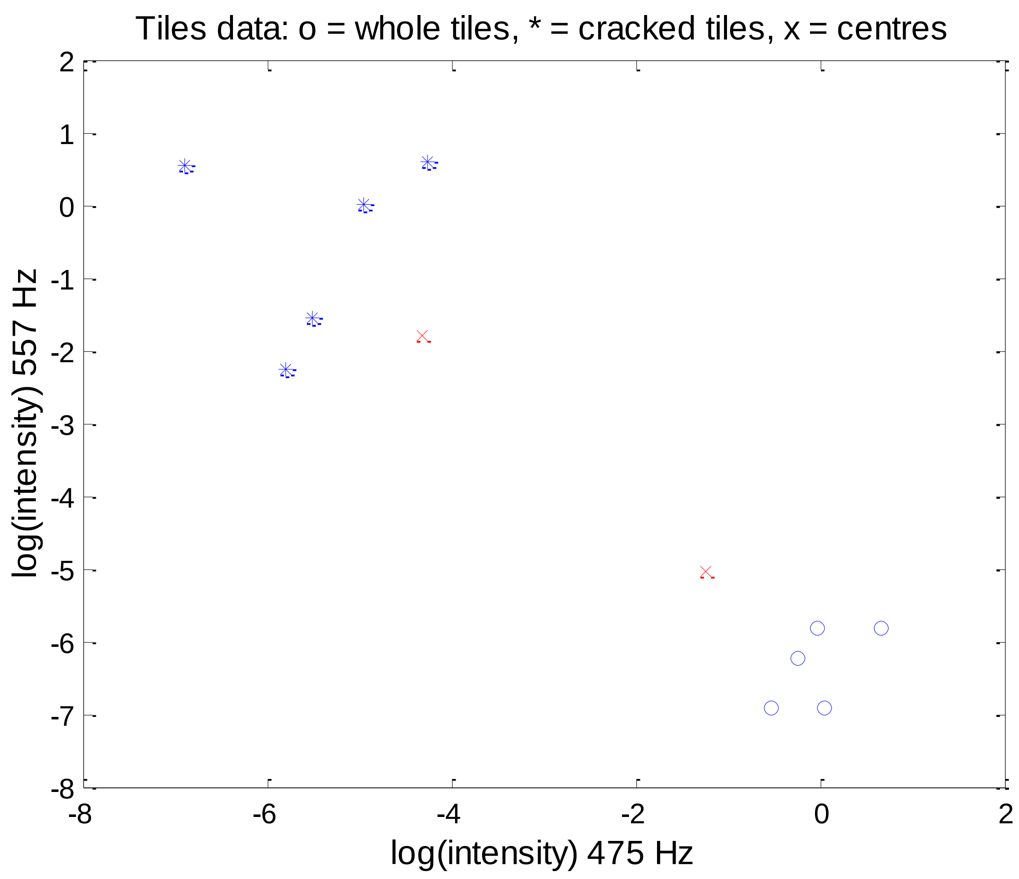
Two classes of data described by (o) and (*). The **objective** is to reproduce the two classes by K=2 clustering.



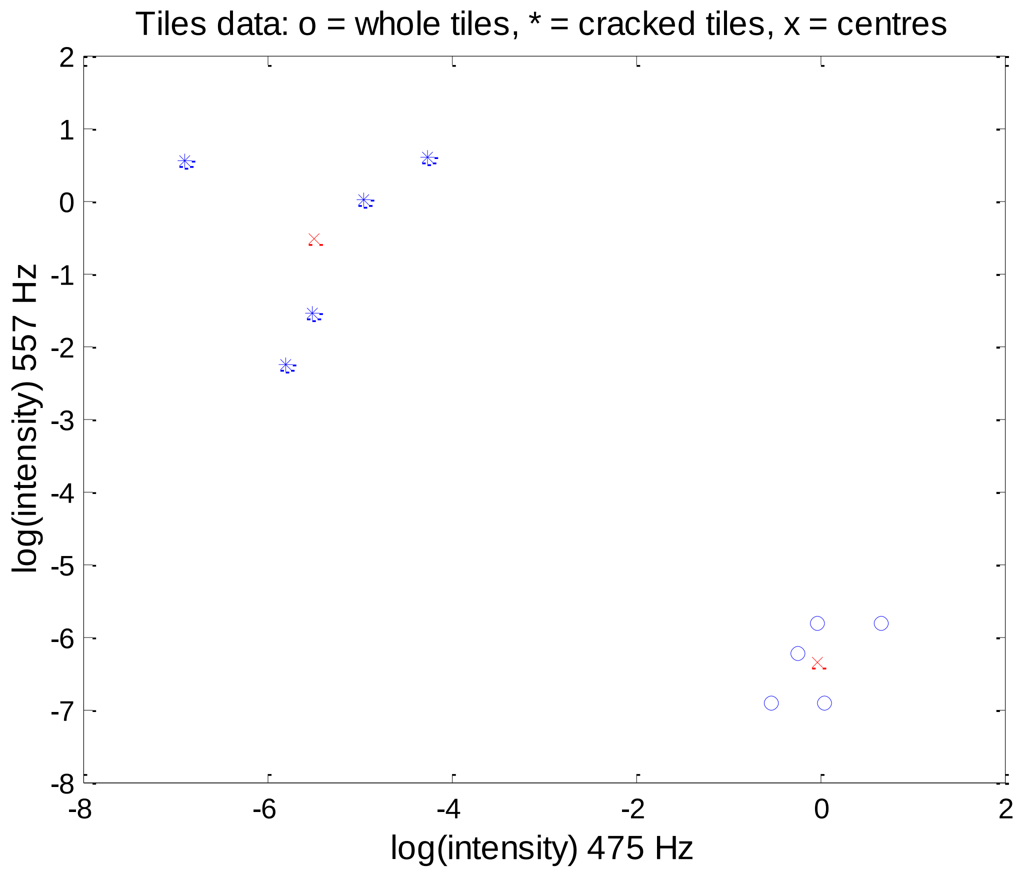
1. Place two cluster centres (x) at random.
2. Assign each data point (* and o) to the nearest cluster centre (x)



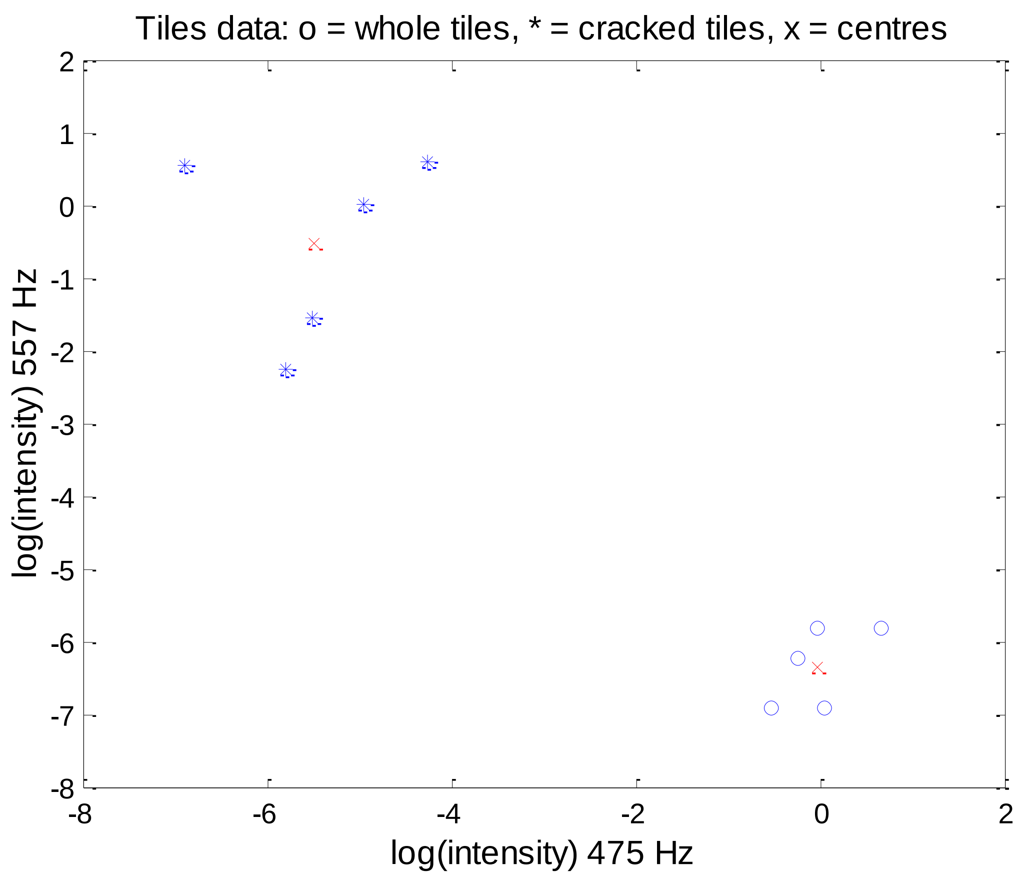
1. Compute the new centre of each class
2. Move the crosses (x)


Tiles data: o = whole tiles, * = cracked tiles, x = centres
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-8
-6
-4
-2
0
2
log(intensity) 557 Hz
log(intensity) 475 Hz

Iteration 2


Tiles data: o = whole tiles, * = cracked tiles, x = centres
log(intensity) 557 Hz
log(intensity) 475 Hz
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-8
-6
-4
-2
0
2

Iteration 3



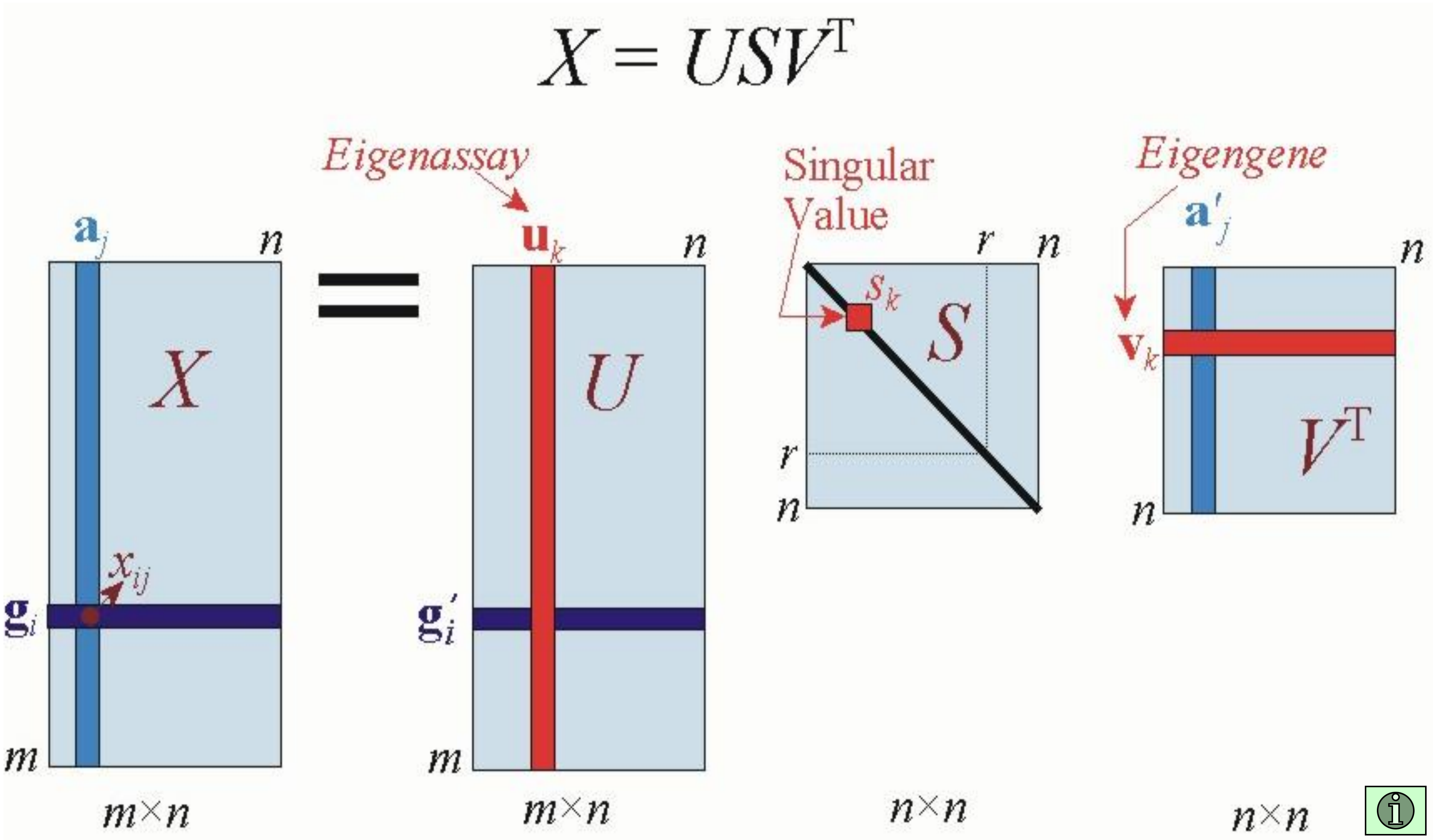
Iteration 4 (then stop, because no visible change)
Each data point belongs to the cluster defined by the nearest centre

Big data

Existing big data are millions of data points and thousands of features

The need for very large dimensional reduction and/or feature selection

SVD – Singular Value Decomposition



Preprocessing: Singular Value Decomposition

SVD involves expanding an *mxn* matrix *X* of rank *k*=min(*m,n*) into a sum of *k* unitary matrices of rank 1, in the following way:

$$X = \sum_{i}^{k} \sigma_i u_i v_i^T$$

This can be rewritten in the matrix representation

$$X = U \Sigma V^T$$

where ∑is a (non-square) diagonal matrix, and U,V are orthogonal matrices. Ordering the non-zero elements of ∑ in descending order, we can get an approximation of lower rank r of the matrix *X* by choosing Σ^r_{jj}=0 for j>r leading to

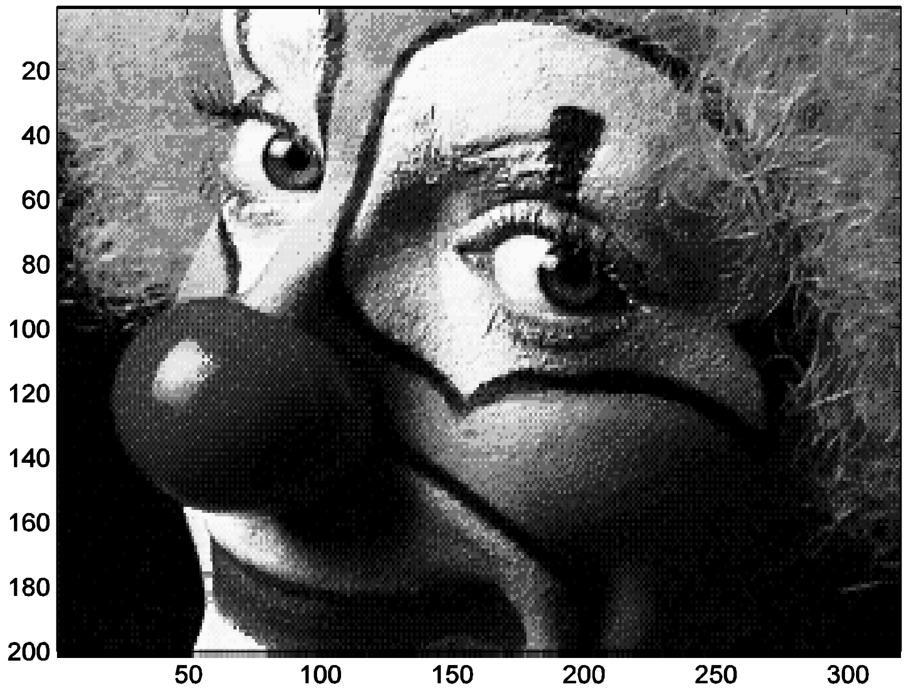
$$Y = U \Sigma^r V^T$$

Singular Value Decomposition continued

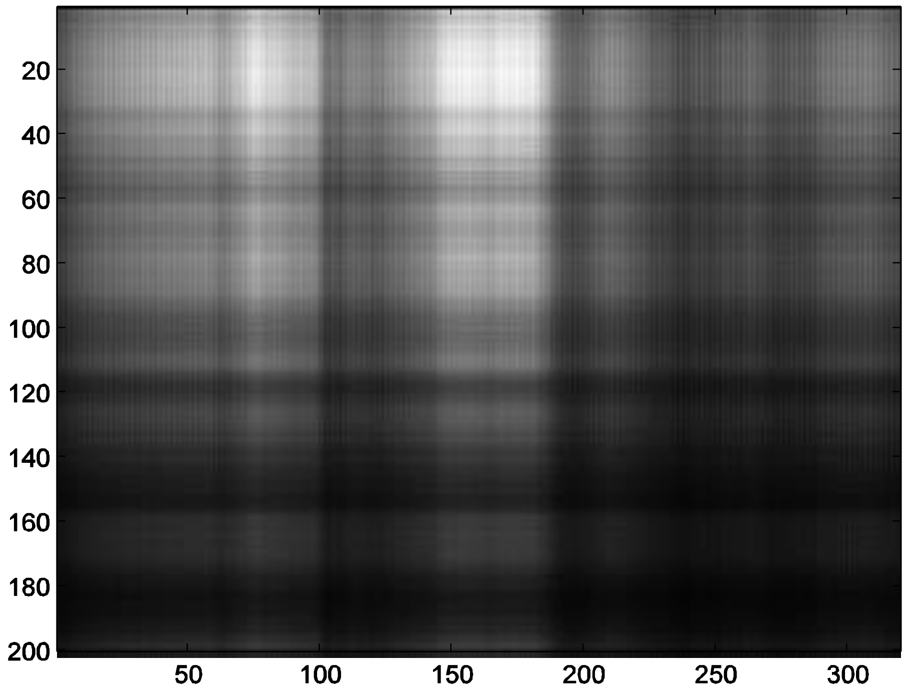
This is the best approximation of rank r to *X*, i.e. it leads to the minimal sum of square deviations

$$S = \sum_{i}^{m} \sum_{j}^{n} (X_{ij} - Y_{ij})^2$$

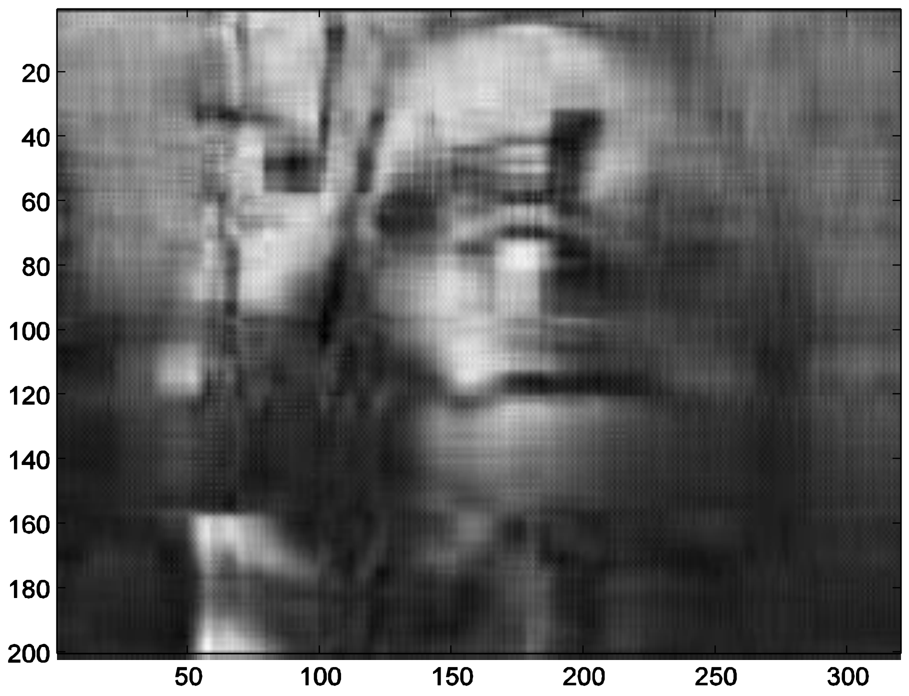
Processing: SVD dimensional reduction



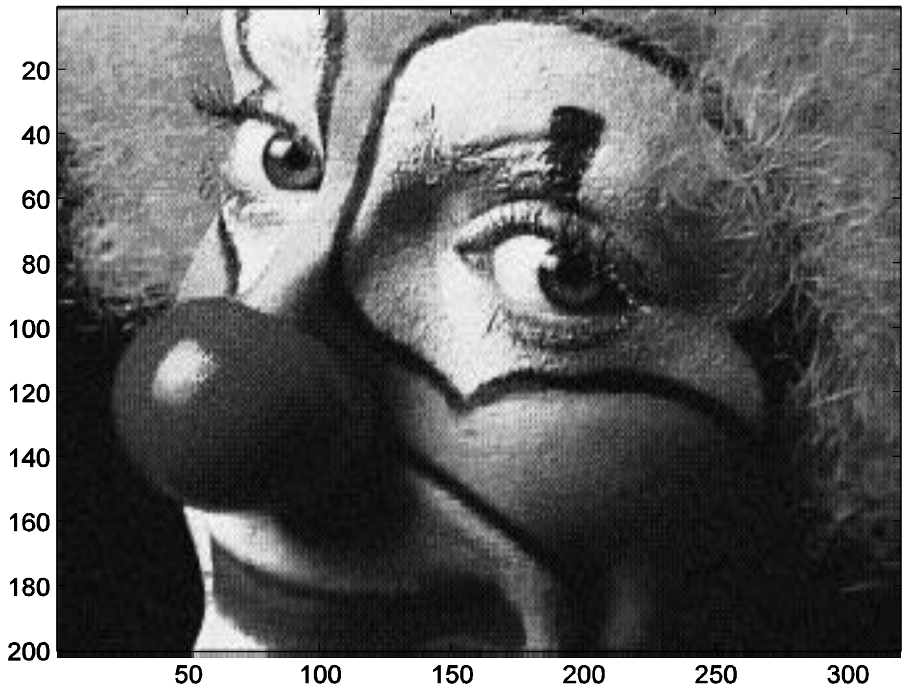
SVD – reduction to 1 dimension



SVD – reduction to 10 dimensions (10 features)



SVD – reduction to 100 dimensions



Applications of SVD

- Dimensionality reduction, compression
- Noise reduction
- Pattern search, clustering

Example: microarray of expression data, “DNA chips”

Relation between SVD and PCA

$$X = USV^T$$

leads to

$$XX^T = US^2U^T$$

and

$$X^TX = VS^2V^T$$

SVD uses the same unitary transformations as PCA performed on the rows or columns of X (using the columns or rows as feature spaces).

The singular values of SVD are the square roots of the eigenvalues of PCA.

The Quantum Clustering trick: The potential transform

Represent data points by Gaussians.
Scale-space approach: turn data-points into Gaussians and study the sum of Gaussians, representing distribution of data

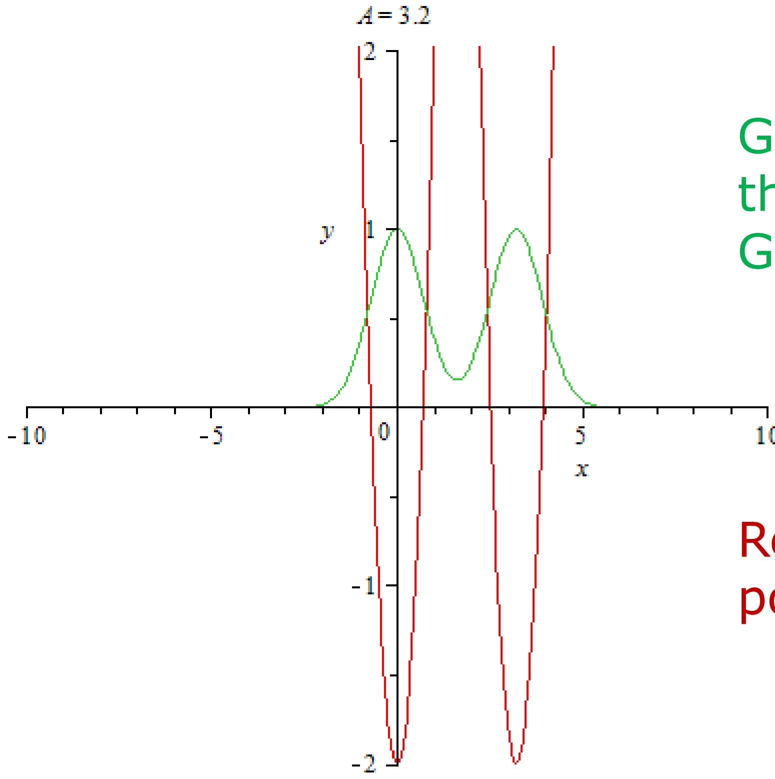
$$\varphi(\vec{x}) = \sum_{i=1}^{n} e^{-\frac{1}{2\sigma^2}(\vec{x}-\vec{x_i})\cdot(\vec{x}-\vec{x_i})}$$

For this probability amplitude we define the potential transform V

$$-\frac{\sigma^2}{2}\nabla^2\varphi + V(\vec{x})\varphi = 0 \qquad V(\vec{x}) = \frac{\sigma^2}{2\varphi}\nabla^2\varphi$$

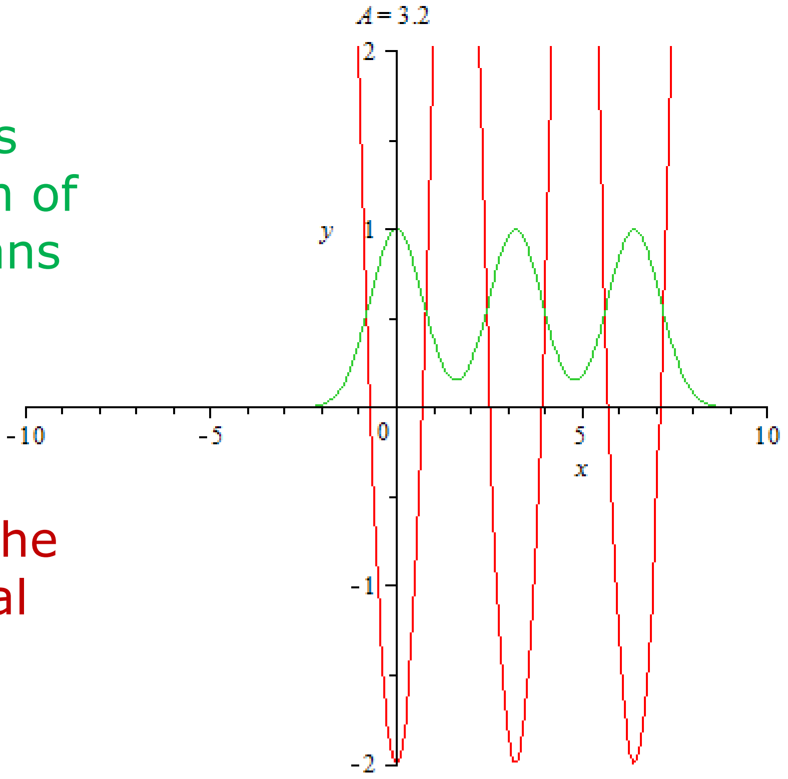
A single Gaussian transforms into a harmonic potential

Comparing sums of Gaussians (centered at 0, A, 2A) and their potentials

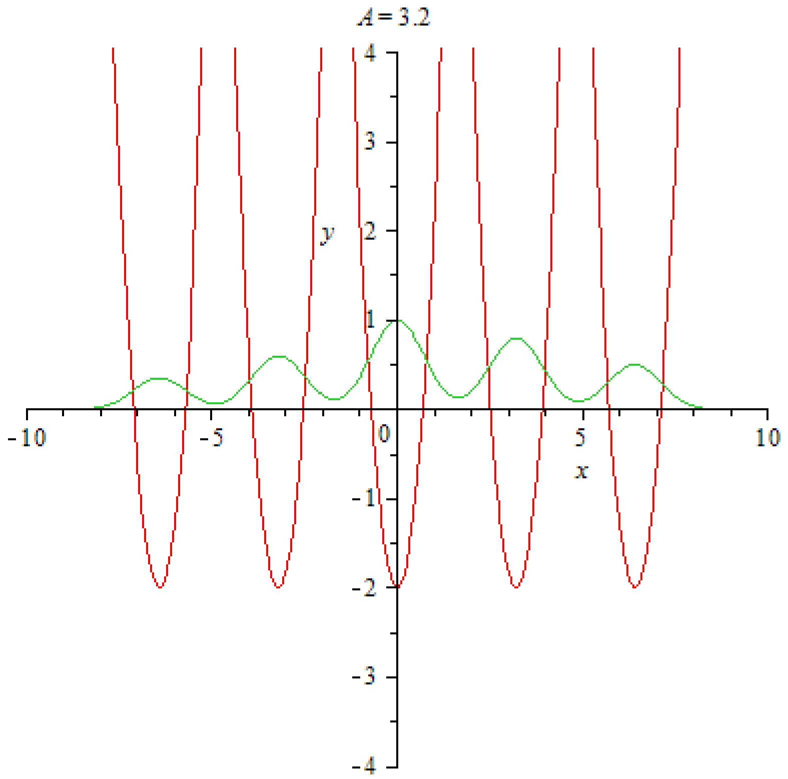


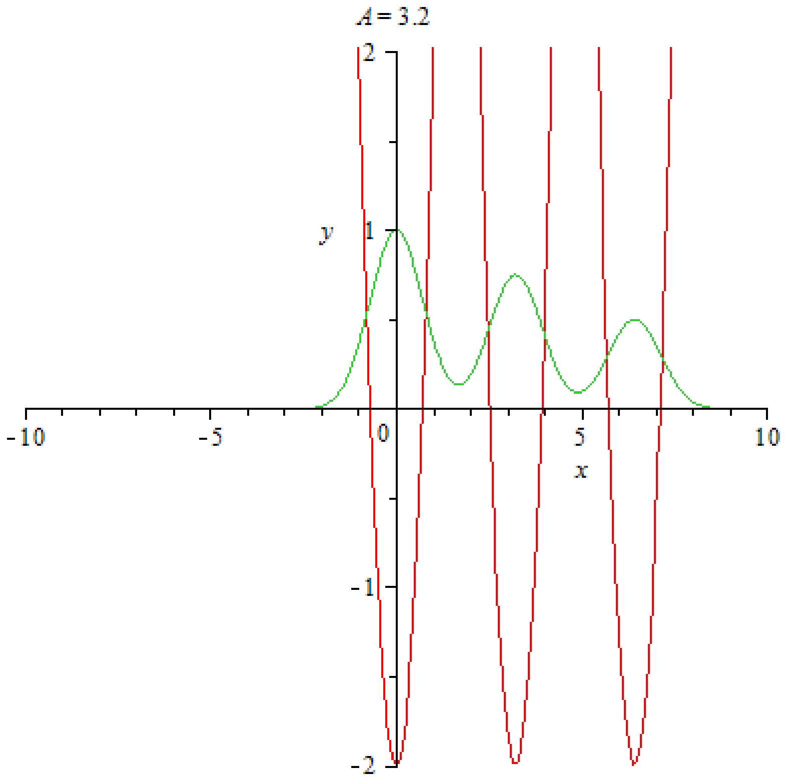
Green is the sum of Gaussians

Red is the potential

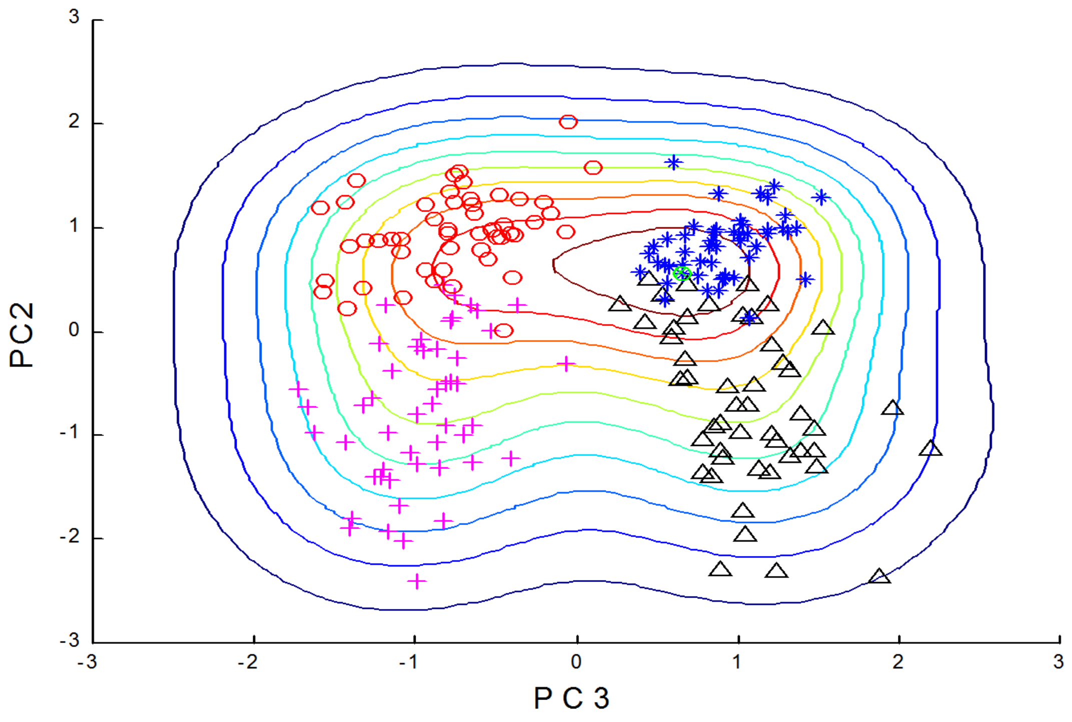


The potential can be thought of as an unbiased way of **contrast enhancing** the Parzen function to better **reveal structure** in the data



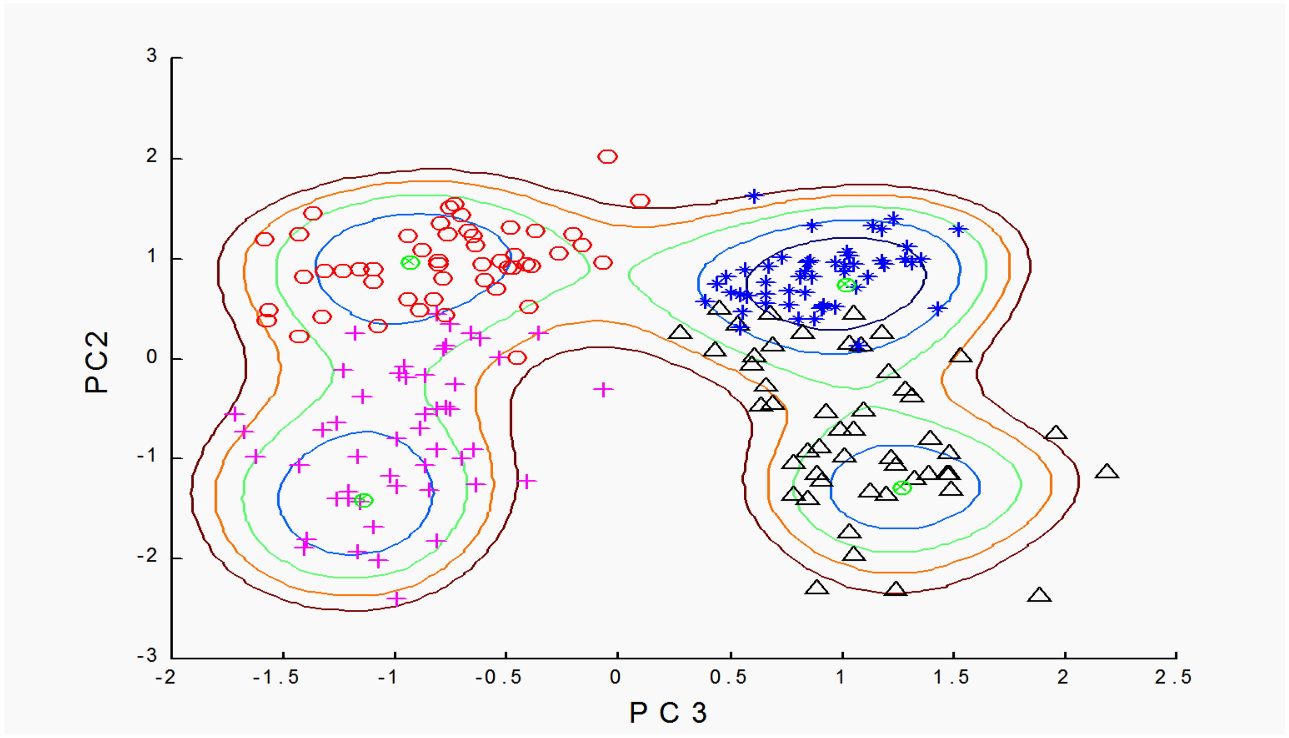


The Crabs Example (from Ripley's textbook)
4 classes, 50 samples each, d=5



A topographic map of the probability distribution for the crab data set with $\sigma=1/\sqrt{2}$. There exists only one maximum although there are 4 classes.

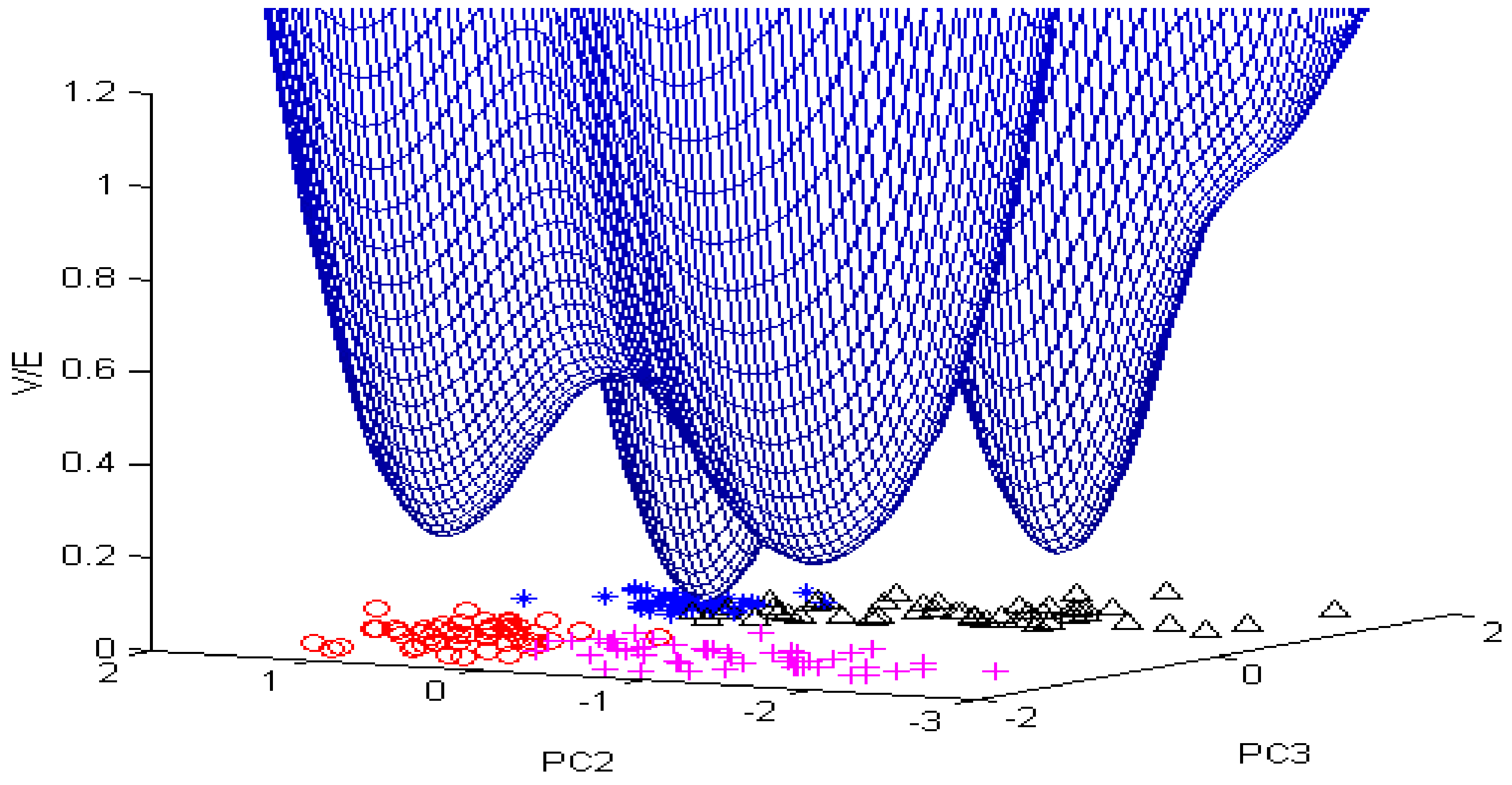
The potential transform
exhibits four minima identified with cluster centers



A topographic map of the potential for the crab data set with $\sigma=1/\sqrt{2}$.

Quantum Clustering: Horn and Gottlieb, Phys. Rev. Lett. 88 (2002) 018702

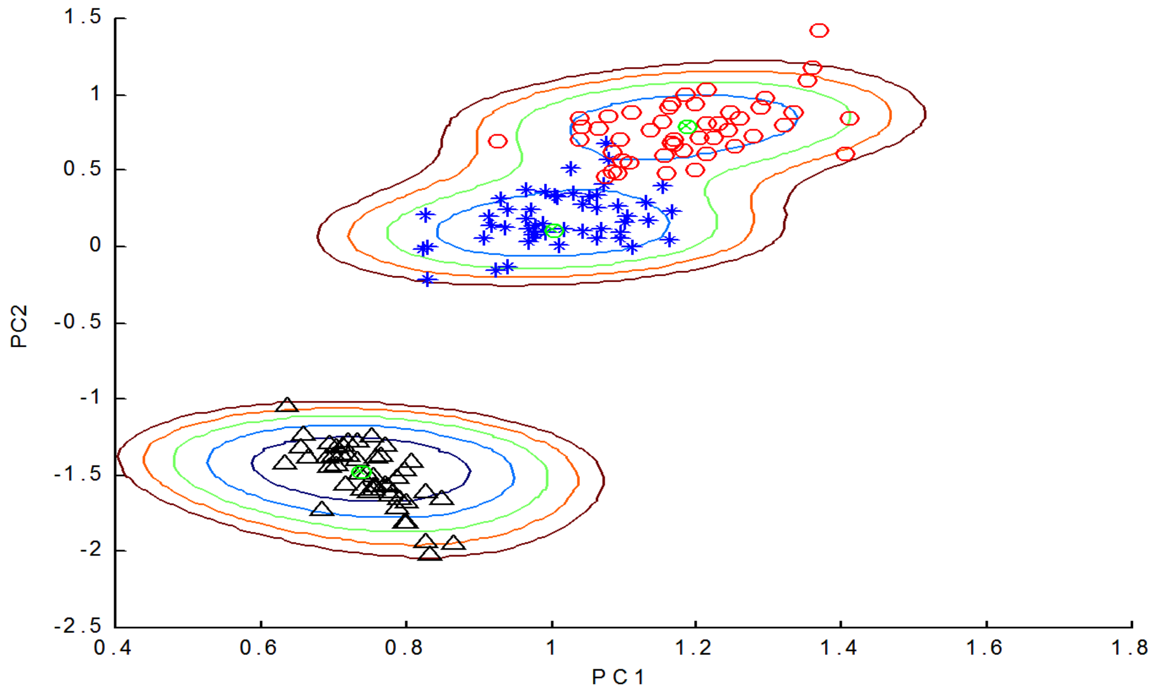
The Crabs Example - Contd.



A three dimensional plot of the potential for the crab data set with $\sigma=1/\sqrt{3}$

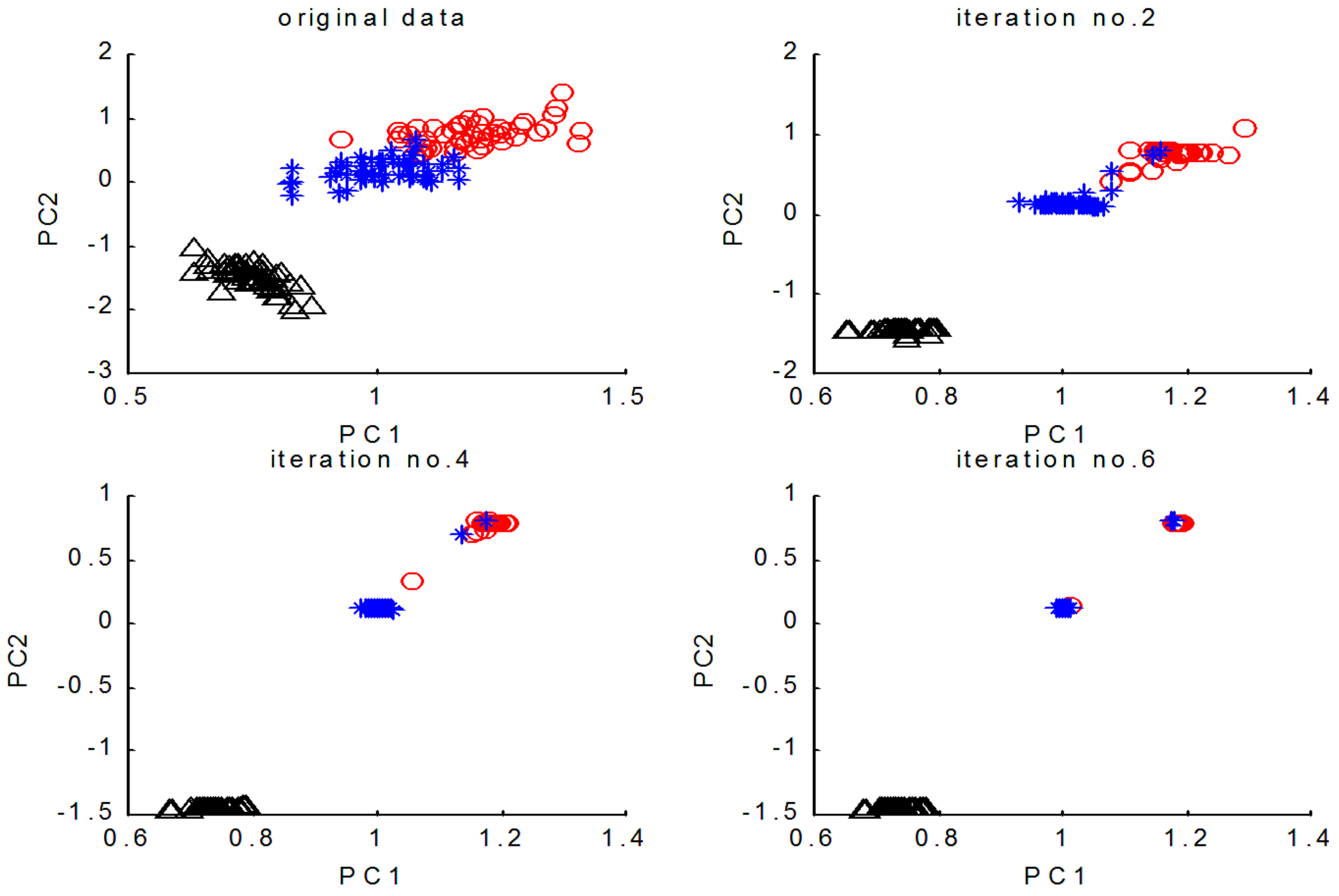
Use gradient descent to let points “fall” into potential minima.

The Iris Example
3 classes, each containing 50 samples, d=4



A topographic map of the potential for the iris data set with σ=0.25 using principal components 1 and 2. The three minima are denoted by crossed circles. The contours are set at values V=cE for c=0.2,…,1.

The Iris Example - Gradient Descent Dynamics



Dynamic Quantum Clustering

Replace the gradient-descent algorithm by a solution of the time-dependent Schrödinger equation, starting with each of the original Gaussians

$$-i\frac{\partial\Psi_i(\vec{x},t)}{\partial t} = \left(-\frac{\nabla^2}{2m} + V(\vec{x})\right)\Psi_i(\vec{x},t)$$

and tracing the convergence of its center-of-mass

$$\langle\, x(\vec{t})\,\rangle = \int d\vec{x}\,\Psi^*(\vec{x},t)\,\vec{x}\,\Psi(\vec{x},t)$$

M. Weinstein and D. Horn, 2009. Dynamic quantum clustering: a method for visual exploration of structures in data. Phys. Rev. E 80, 066117.

Dynamic Quantum Clustering

The differential equation can be solved algebraically by expanding the Hamiltonian within the n Gaussian states defined at the n data-points. Thus, for any dimension, the problem can be reduced to an nXn set of matrix elements.

$$H = \frac{p^2}{2m} + V(x)$$

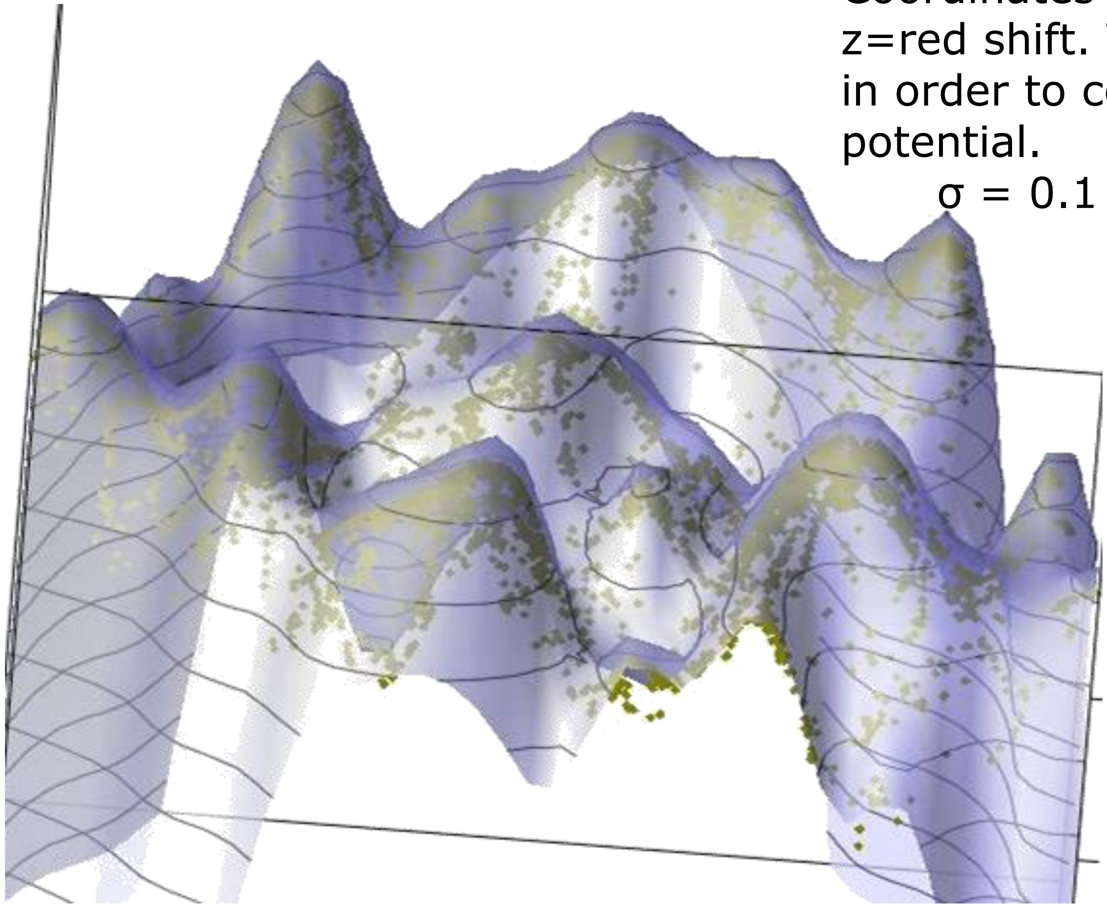
$$H_{ij} = \langle \psi_i | H | \psi_j \rangle$$

$$N_{ij} = \langle \psi_i | \psi_j \rangle$$

$$\vec{X}_{ij} = \langle \psi_i | \vec{x} | \psi_j \rangle$$

Then exponentiate the finite matrix and compute the time evolution of the expectation values

Sloan Digital Sky Survey: 335K galaxies

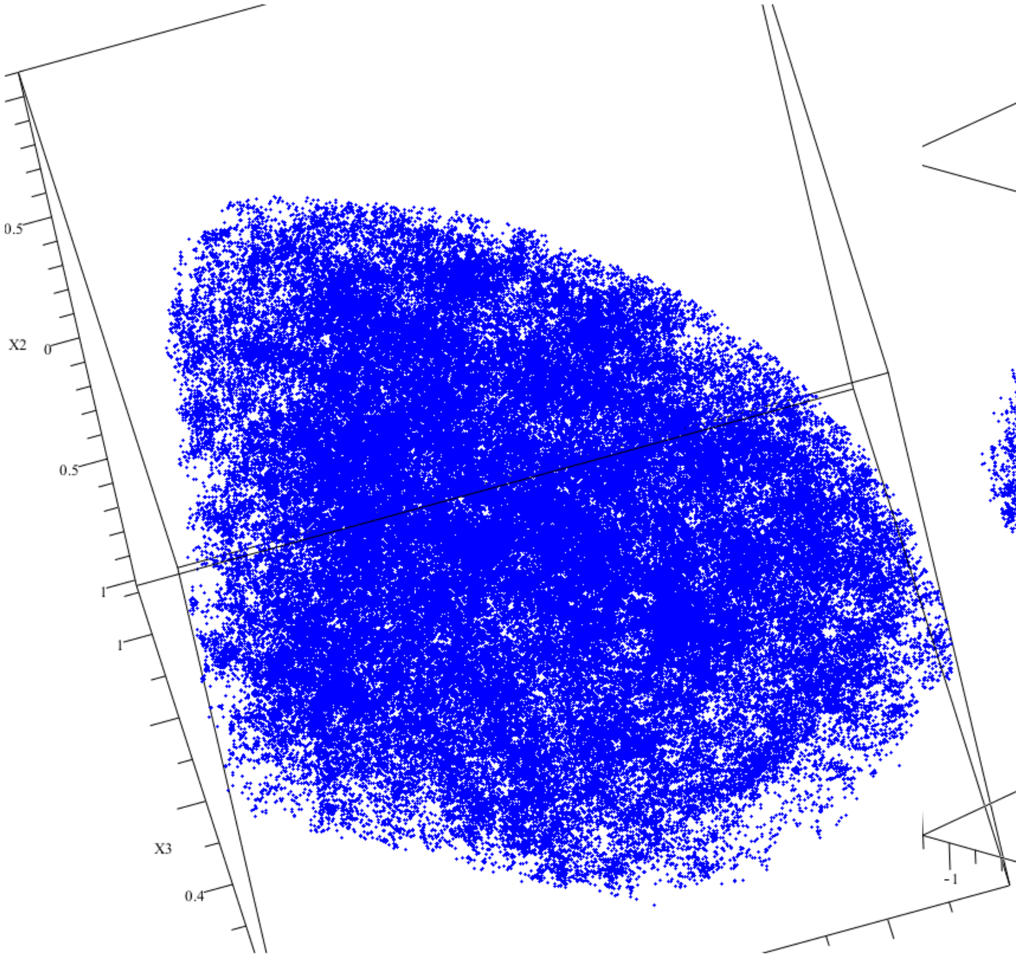


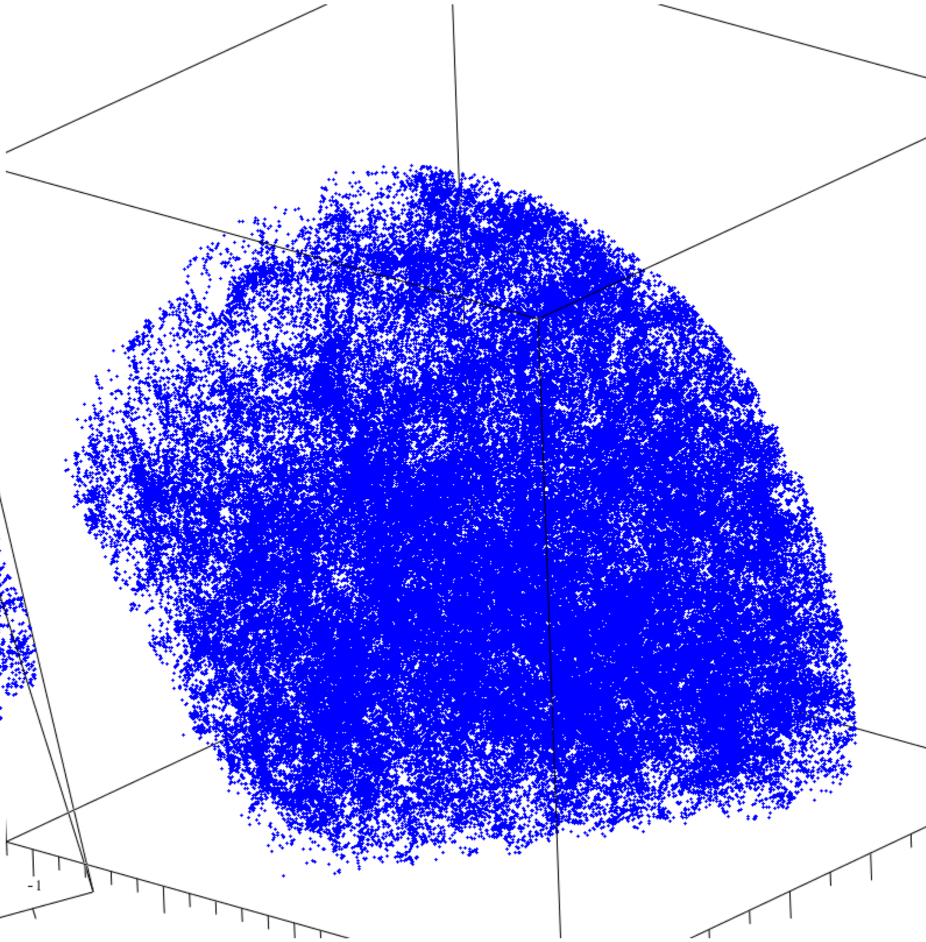
Coordinates are angles θ φ and z=red shift. Take a thin slice in z in order to compare data with potential.

$\sigma = 0.1$

Upside Down Potential Plot: the points were slightly raised above the potential for better visibility

Sloan Digital Sky Data: Demonstrating DQC flow





EXAMPLE :NANO-CHEMISTRY

UNBIASED ANALYSIS OF X-RAY ABSORPTION DATA

Data collected at the Stanford Synchrotron Radiation Light source (SSRL), using the TXM-XANES microscope, a new device that enables an efficient study of hierarchically complex materials

Analyzing Big Data with Dynamic Quantum Clustering
M. Weinstein, F. Meirer, A. Hume, Ph. Sciau, G. Shaked, R. Hofstetter, E. Persi, A. Mehta, D. Horn http://arxiv.org/abs/1310.2700

Very complex problem: Interface of materials

- Sample data: Roman pottery
  - Red and Black colors are due to different iron oxides
- Similar problems:
  - Lithium-ion batteries
  - Catalyst breakdown

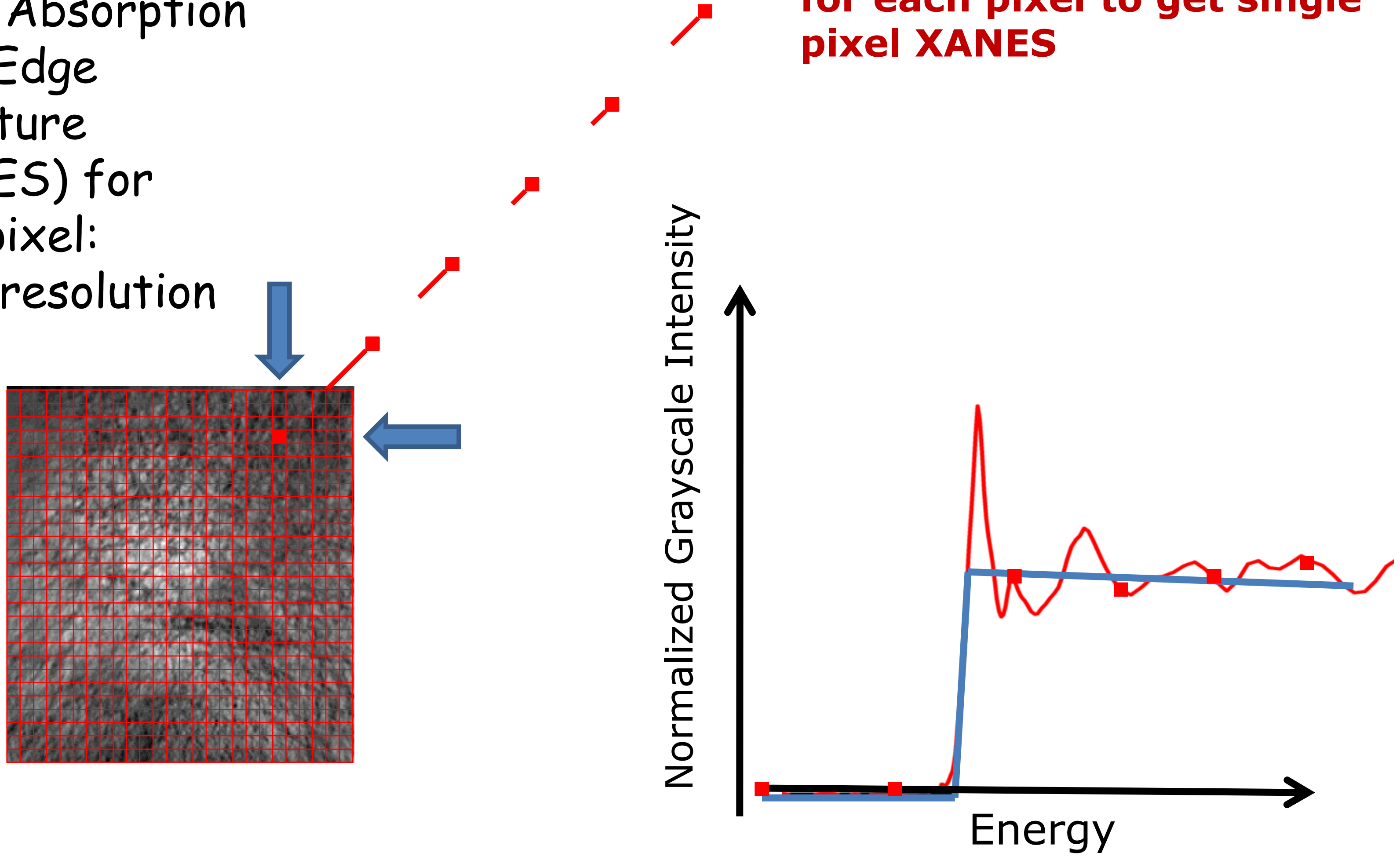
What Will We Learn?

- This is a big, noisy dataset
  - 669,000 x-ray absorption spectra at 146 energies (the energies = features)
  - Full of experimental artifacts
- Goal
  - To group spectra into similar shapes, because the shape correlates with the iron oxide present in the sample
  - There is a needle in this haystack!
- Requirement
  - To do this without assumptions (i.e. in an unsupervised manner)

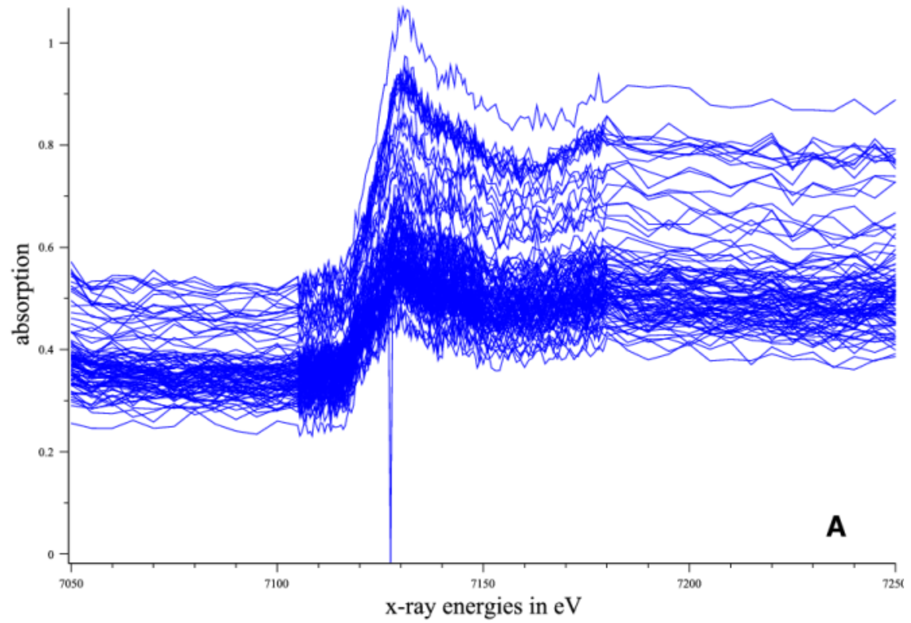
TXM-Xanes

- **Collect one high resolution absorption image at each energy**
- **Trace the absorption value for each pixel to get single pixel XANES**

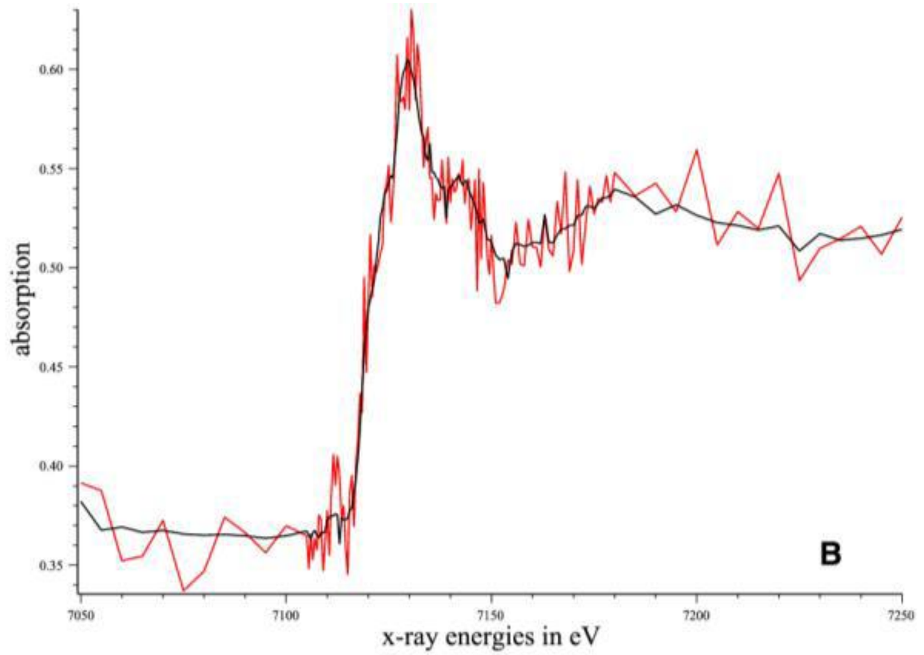
X-ray Absorption Near Edge Structure (XANES) for each pixel: 30nm resolution



Applying SVD reduction from 146 to 5 dimensions reduces much of the noise in the X-ray absorption spectrum



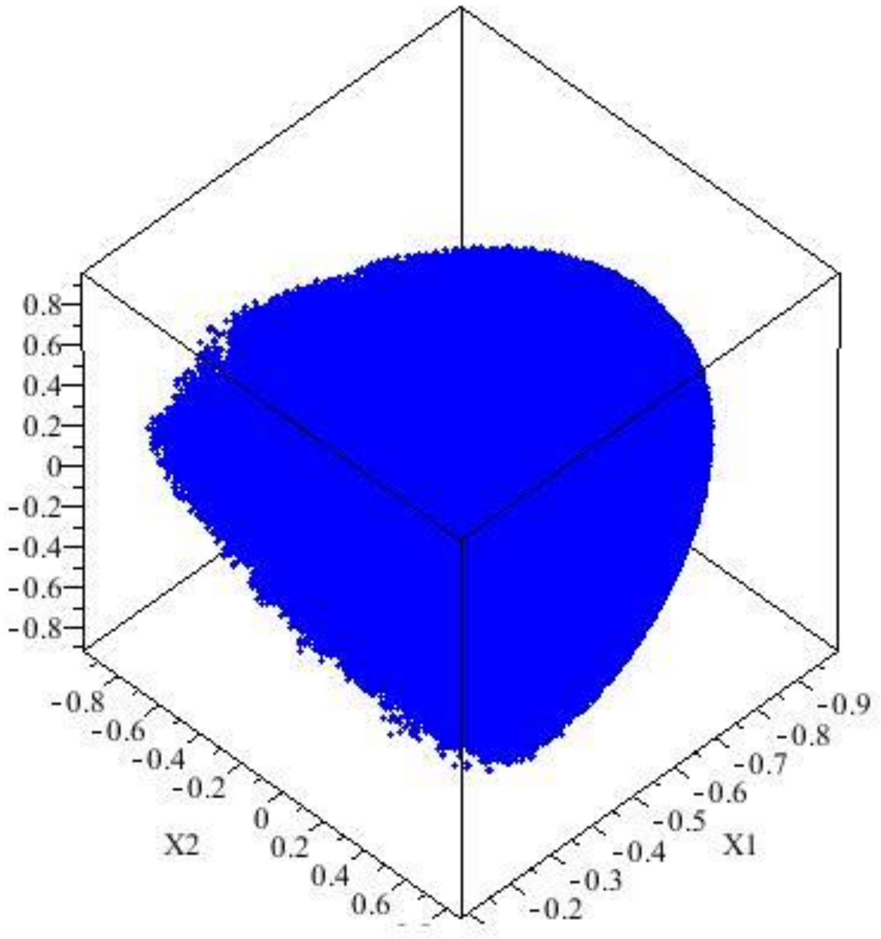
collection of raw data



Red – a typical curve
Black-after noise reduction

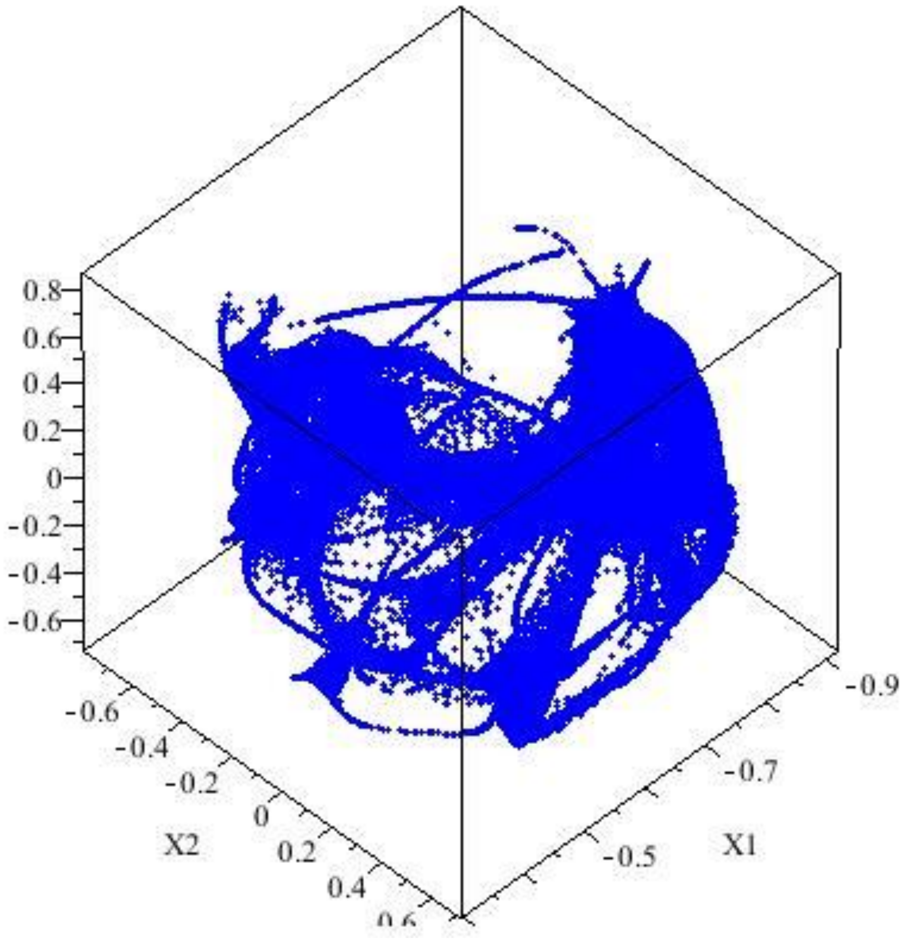
669,000 points in 5 dim feature space, projected onto a unit sphere

Clustering Process:



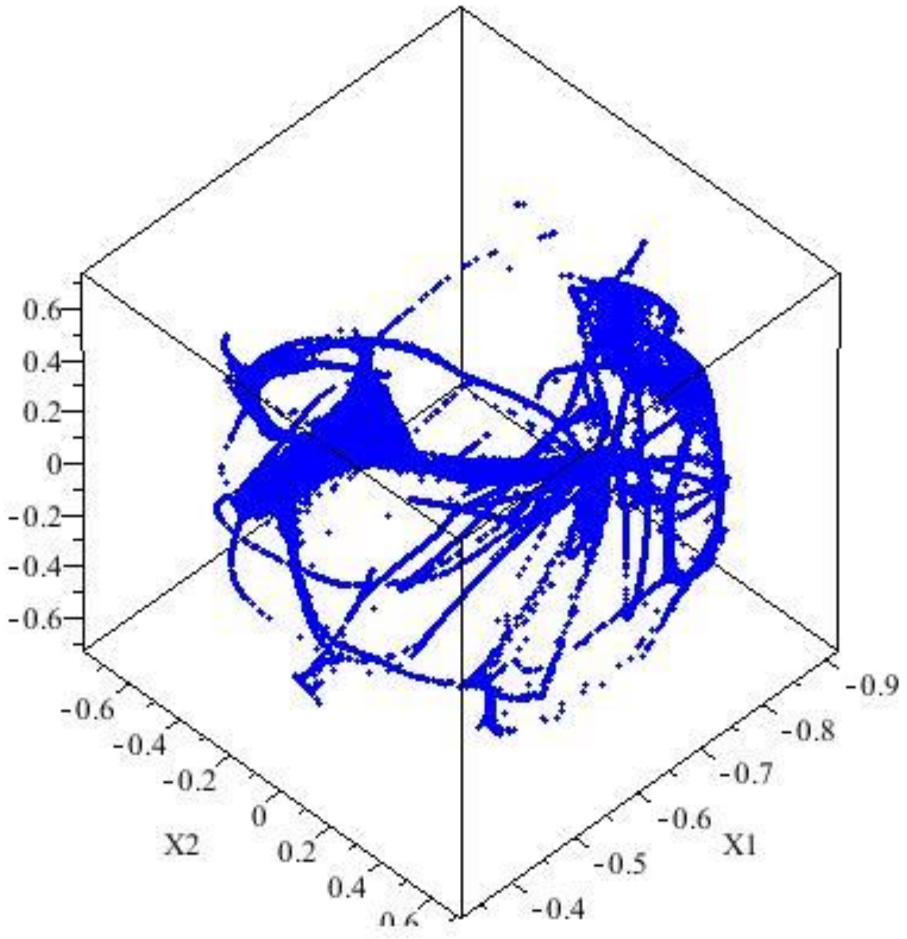
Clustering Process:

Data collapses into clumps and strands



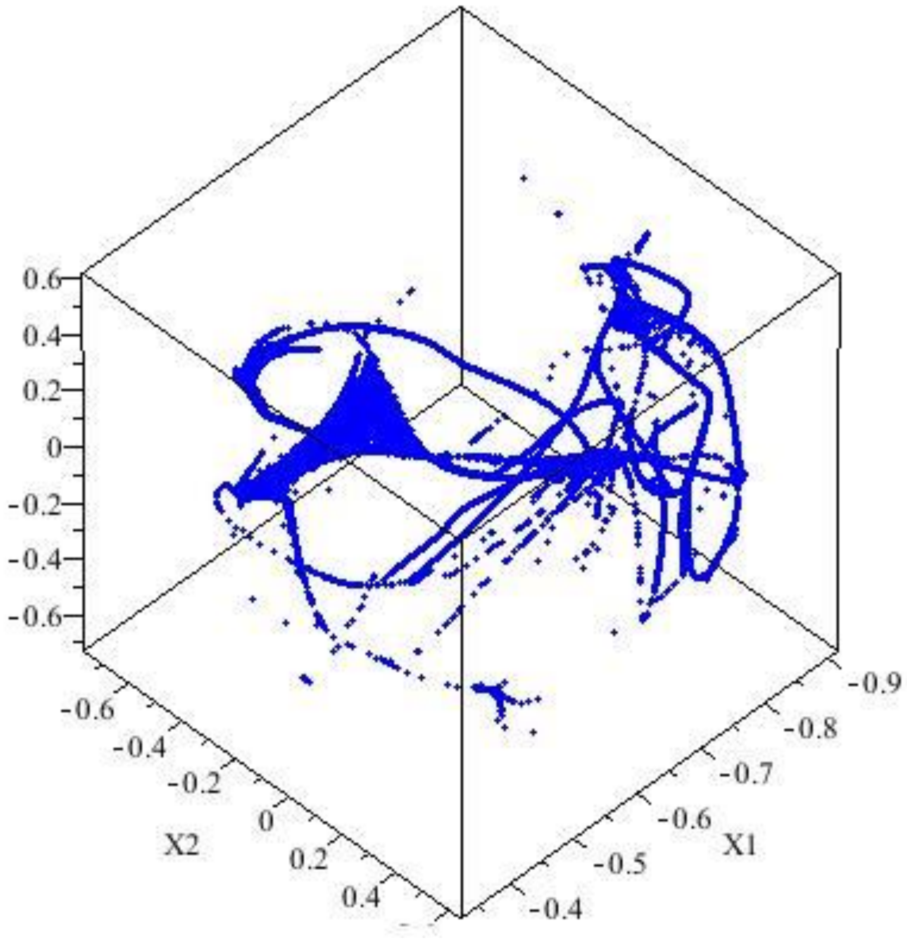
Clustering Process:

Data collapses into clumps and strands



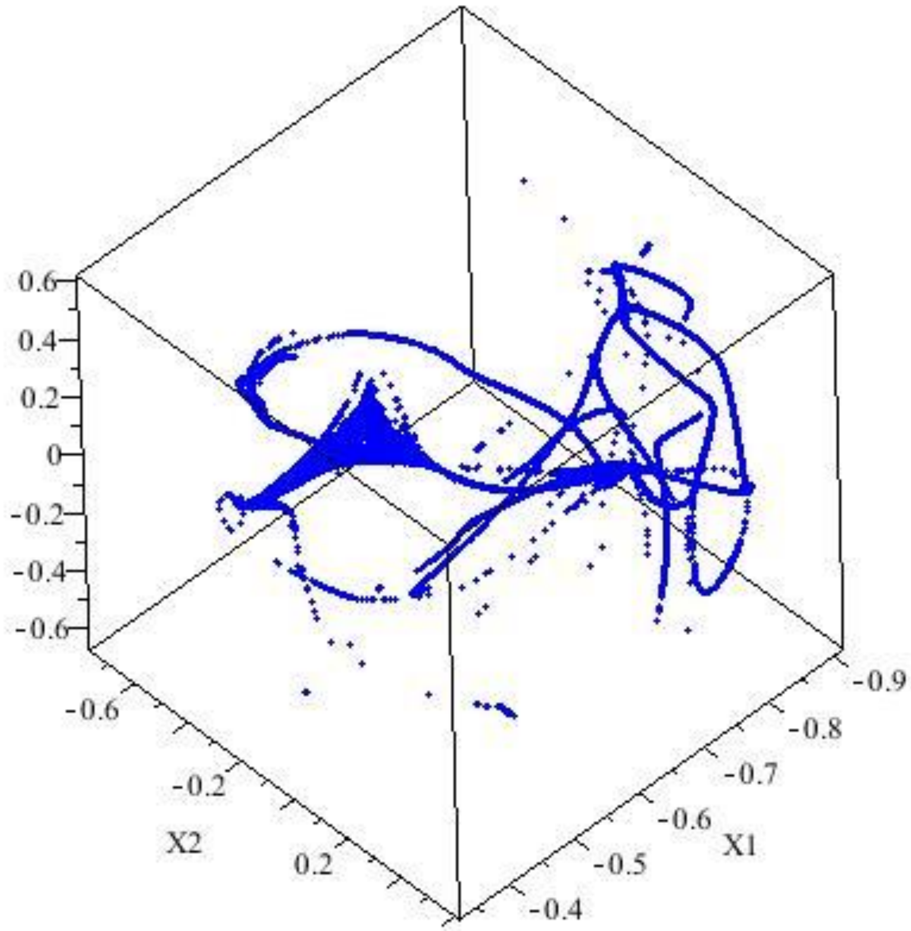
Clustering Process:

Some strands collapse to points, others remain



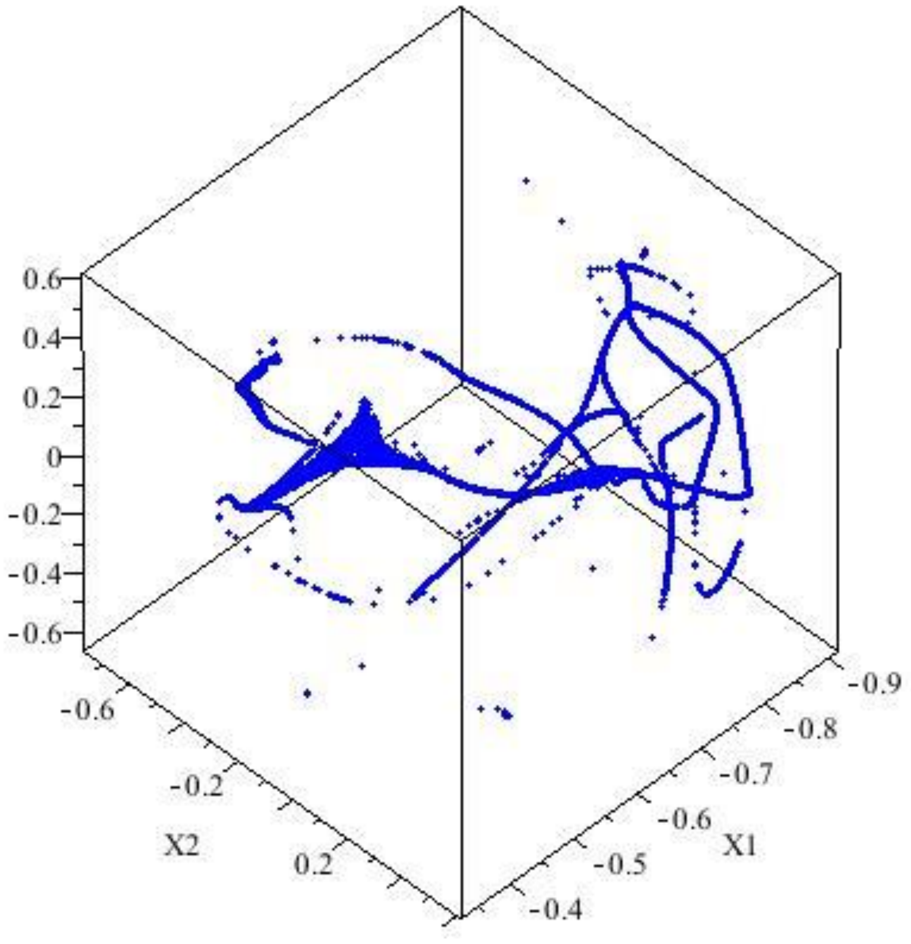
Clustering Process:

Some strands collapse to points, others remain



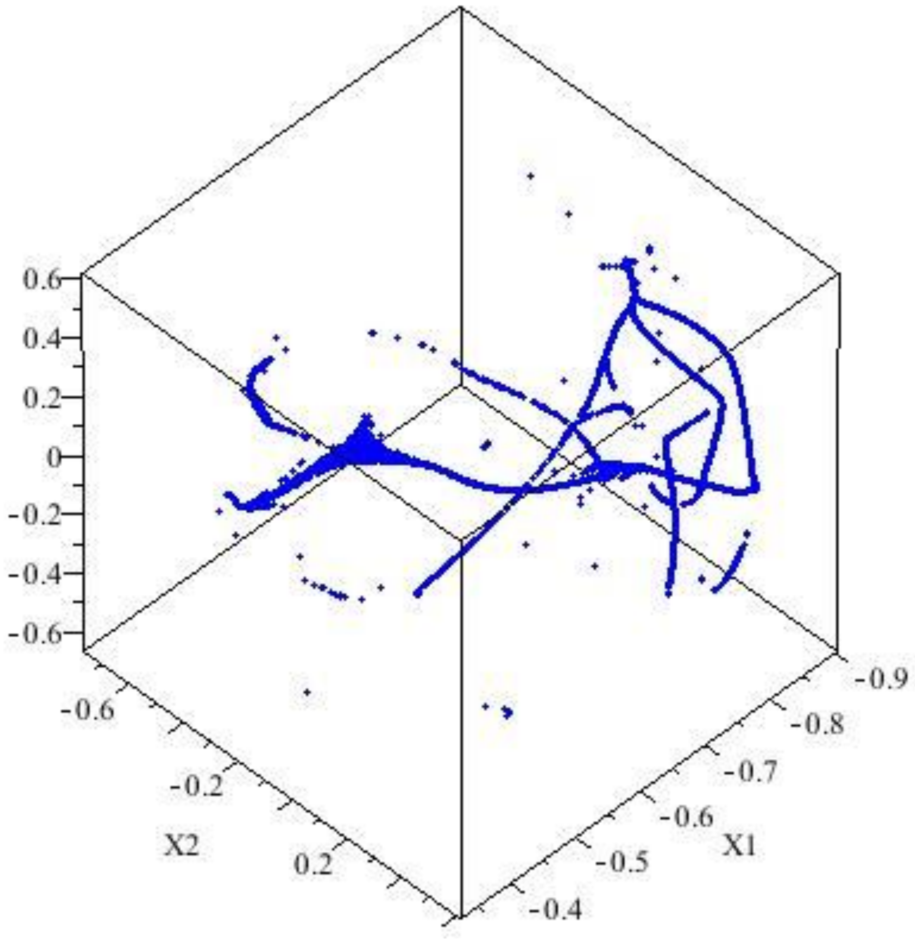
Clustering Process:

Separation continues



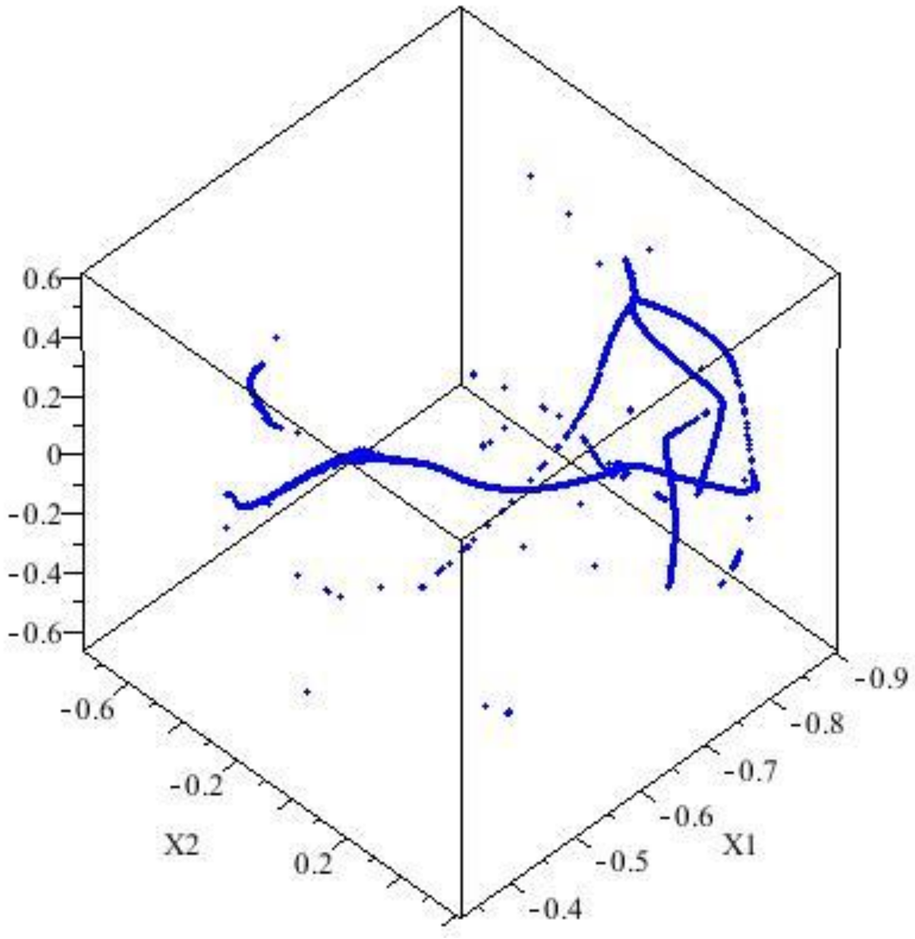
Clustering Process:

Separation continues

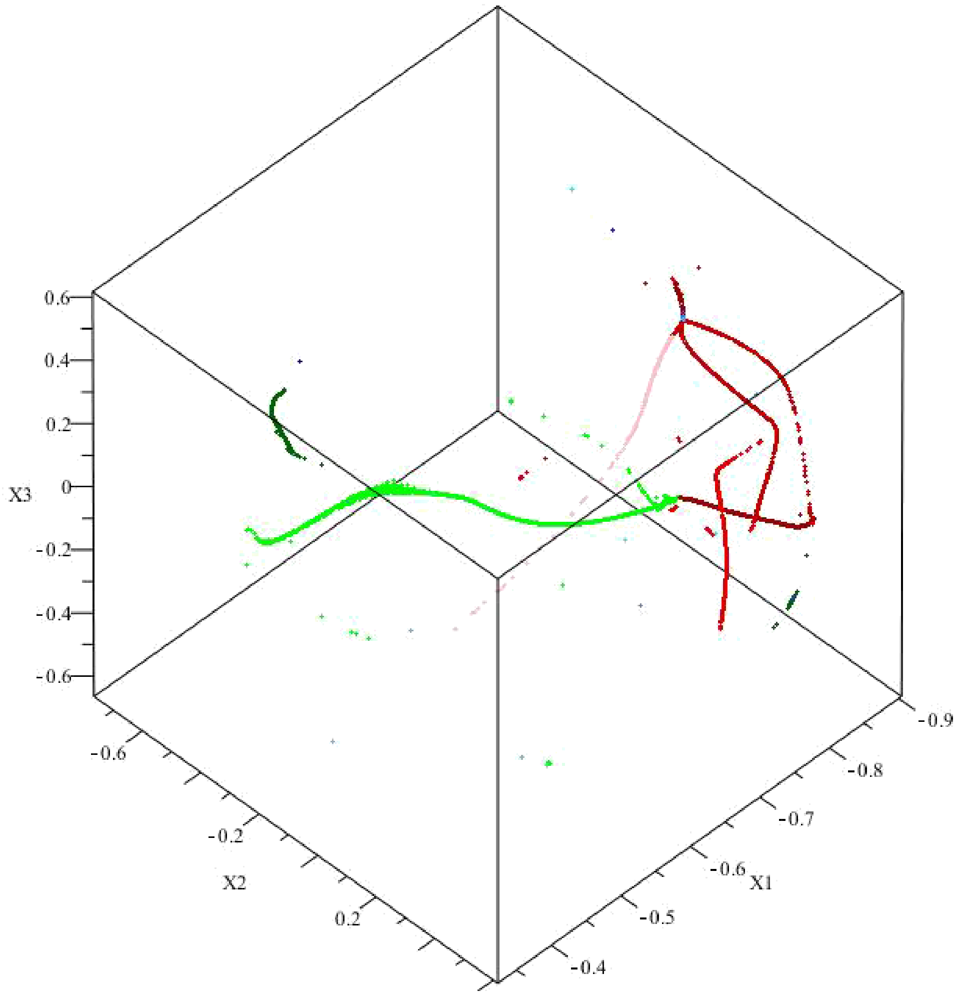


Clustering Process results in Structures and Point Clusters

Identify each connected string as a different structure. Color each structure differently.



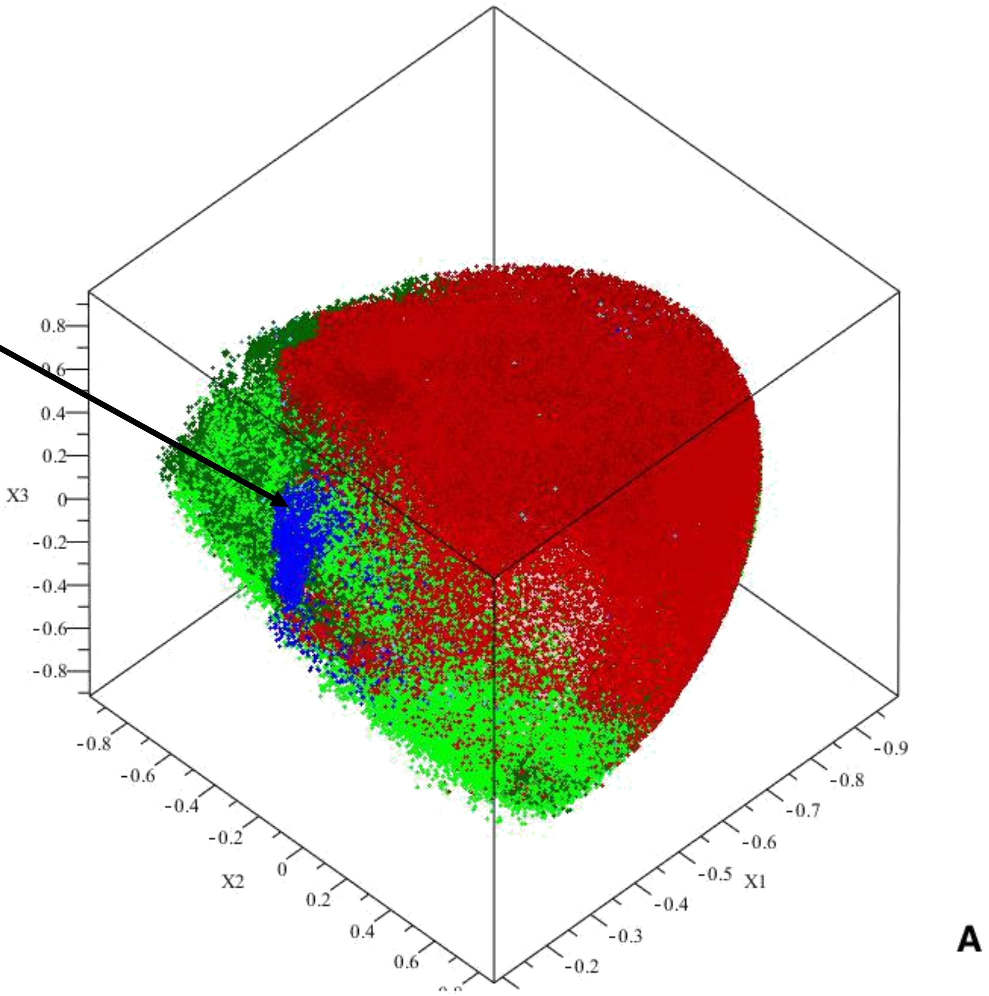
Coloring of the structures and clusters



F

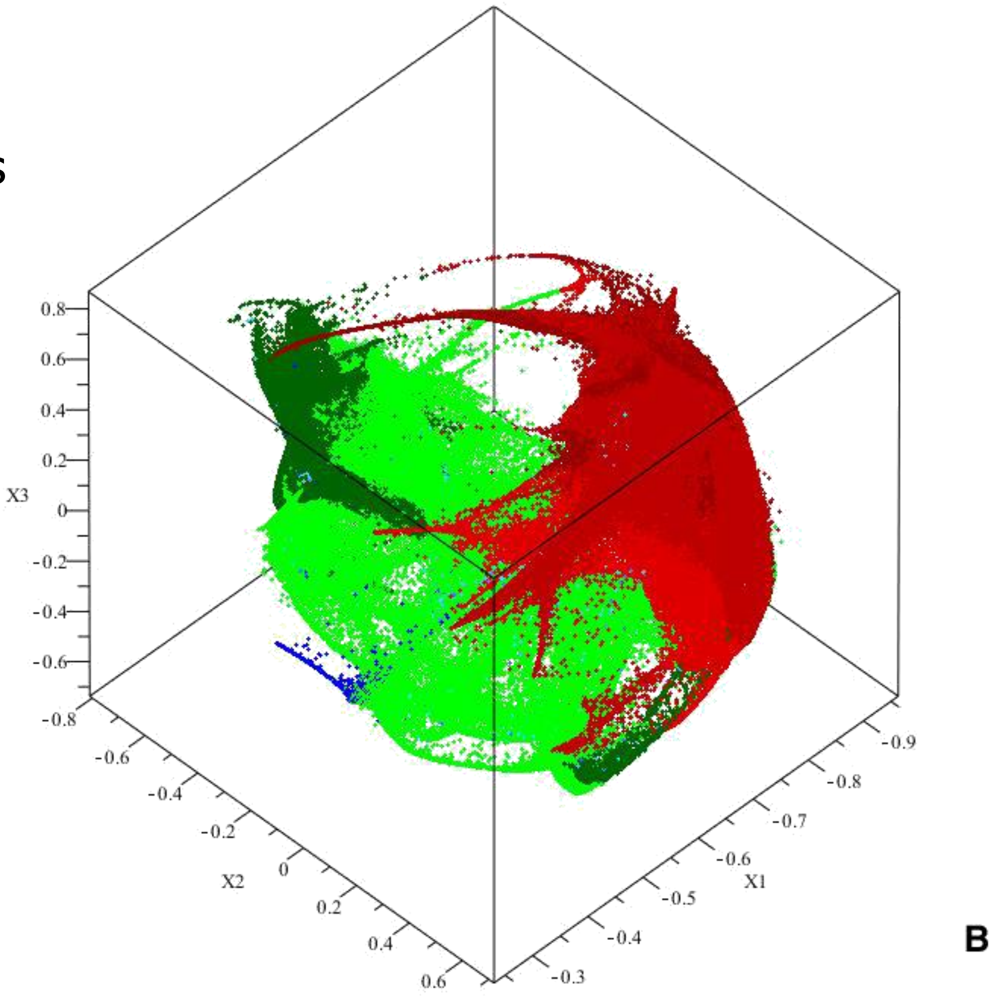
Clustering process redrawn with colors

The needle in the haystack



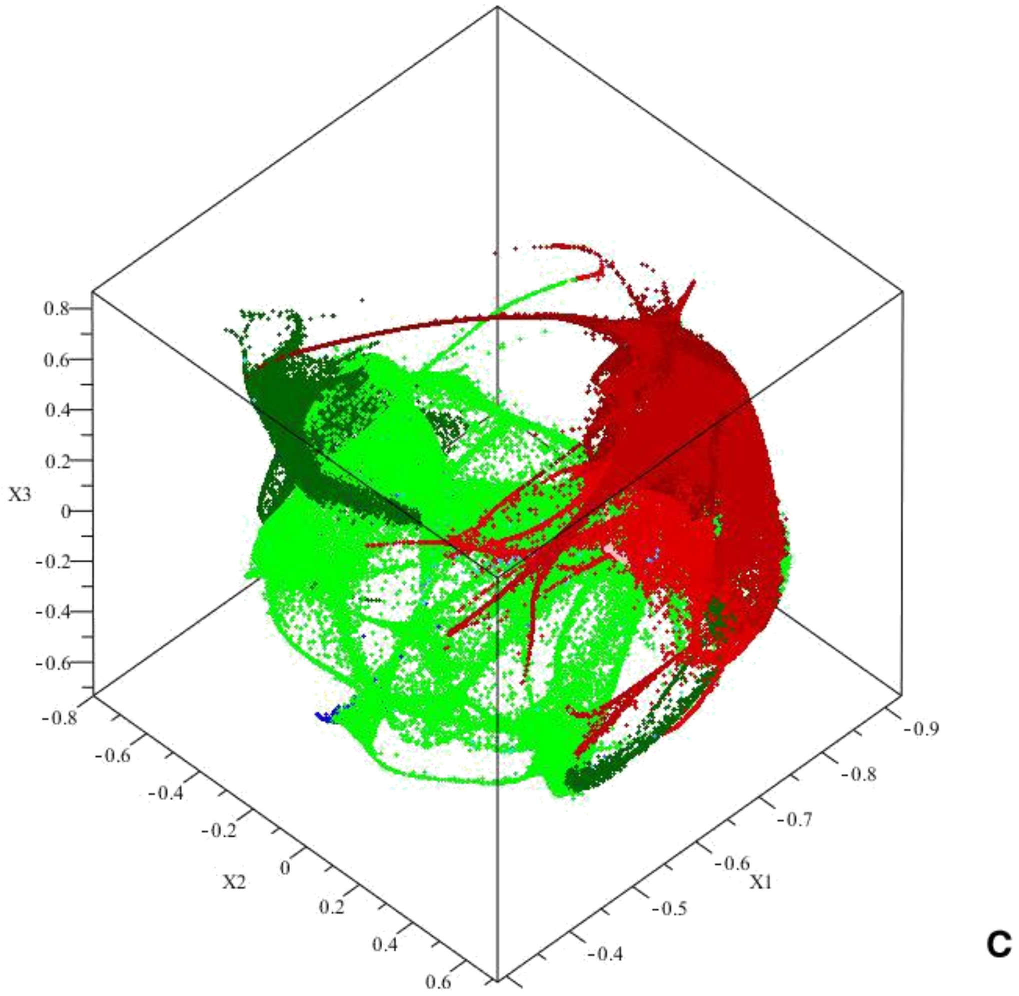
Clustering Process

Data collapses into clumps and strands. The blue data swiftly separates



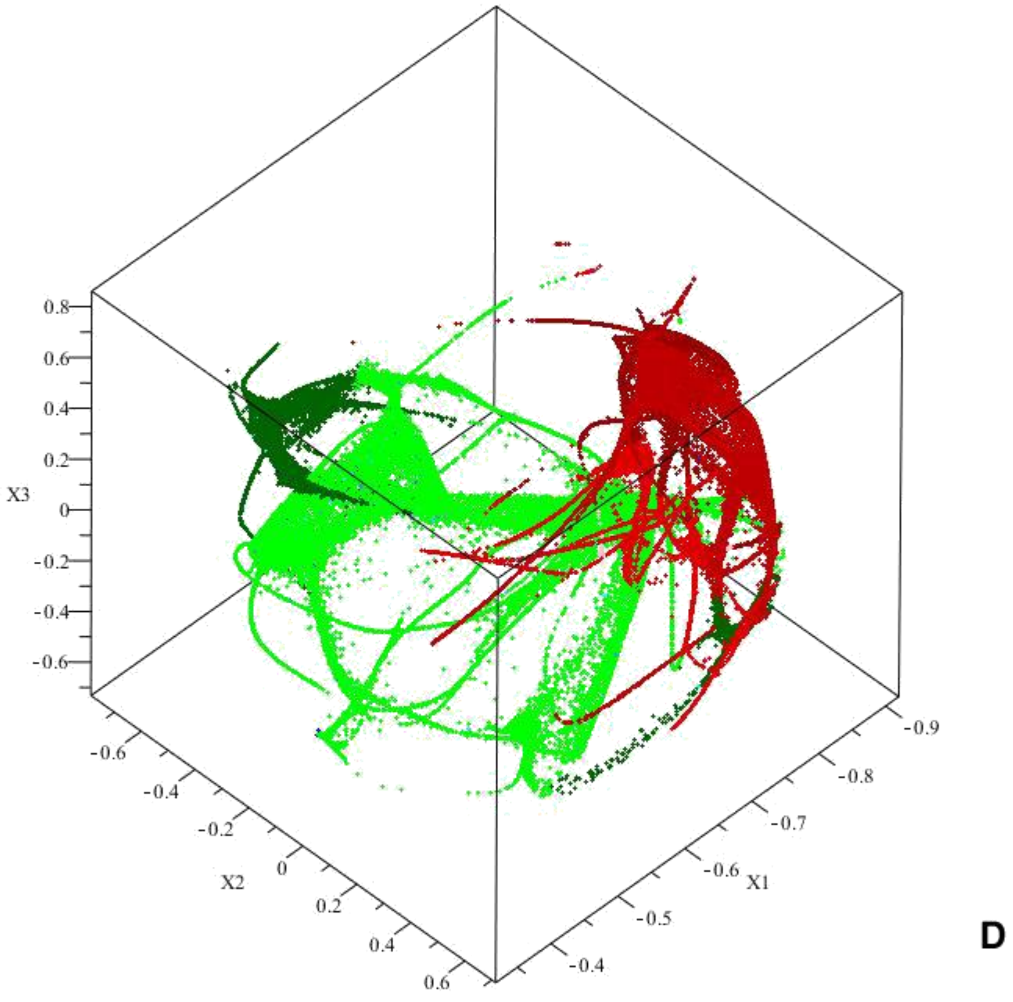
Clustering Process

Data collapses into clumps and strands



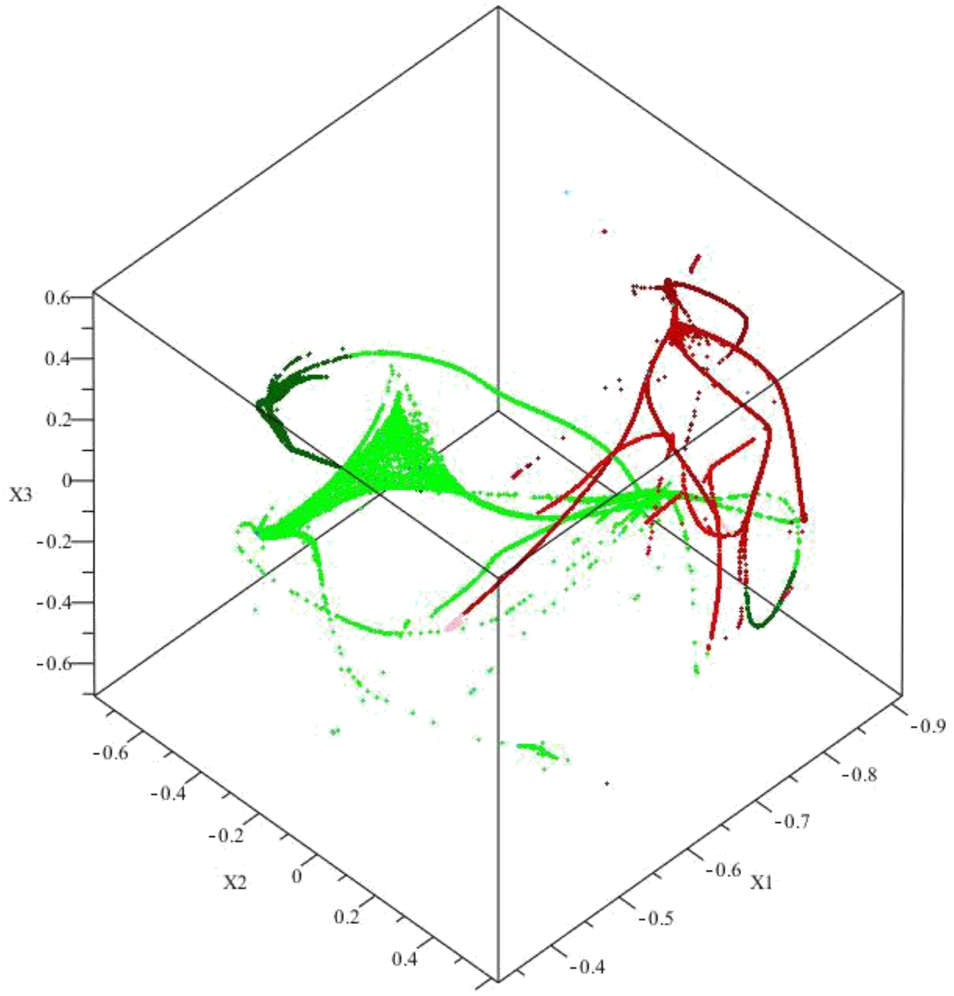
Clustering Process

Some strands collapse to points, others remain



Clustering Process

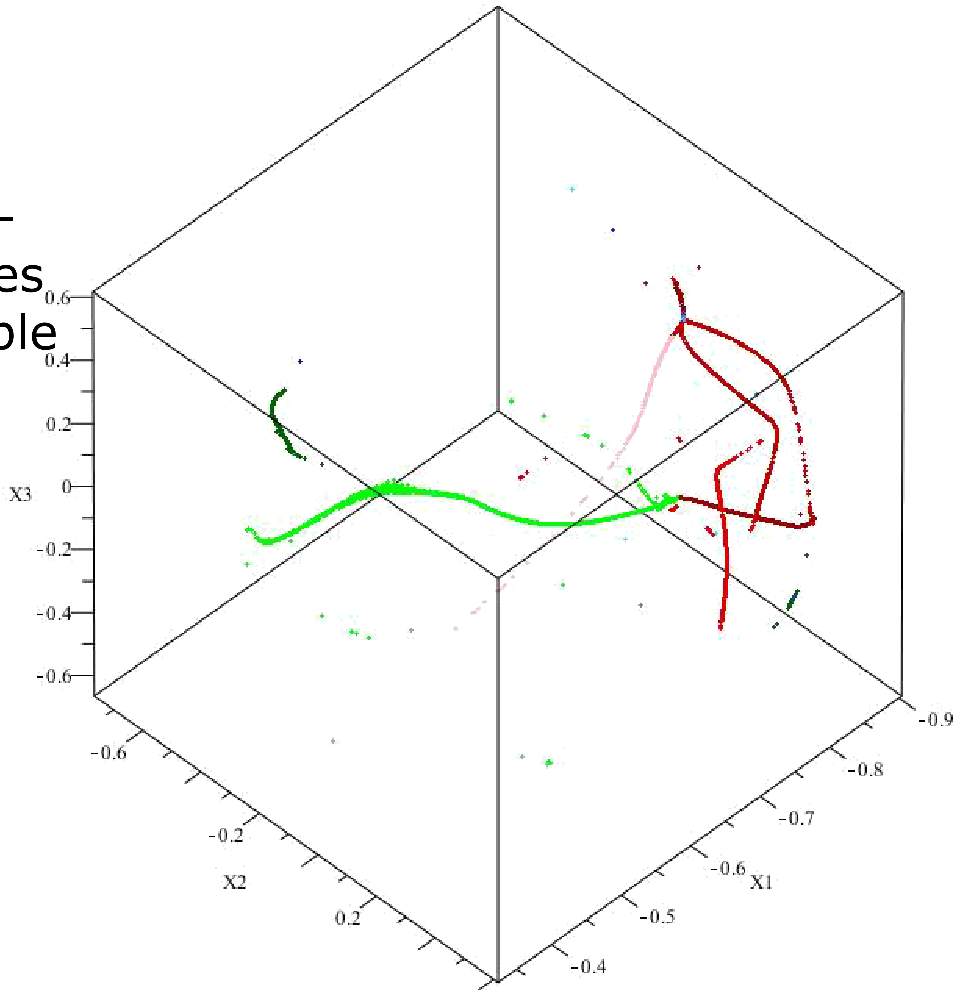
Some strands collapse to points, others remain



E

Clustering Process

Separation continues leading to non-trivial structures as well as simple clusters



F

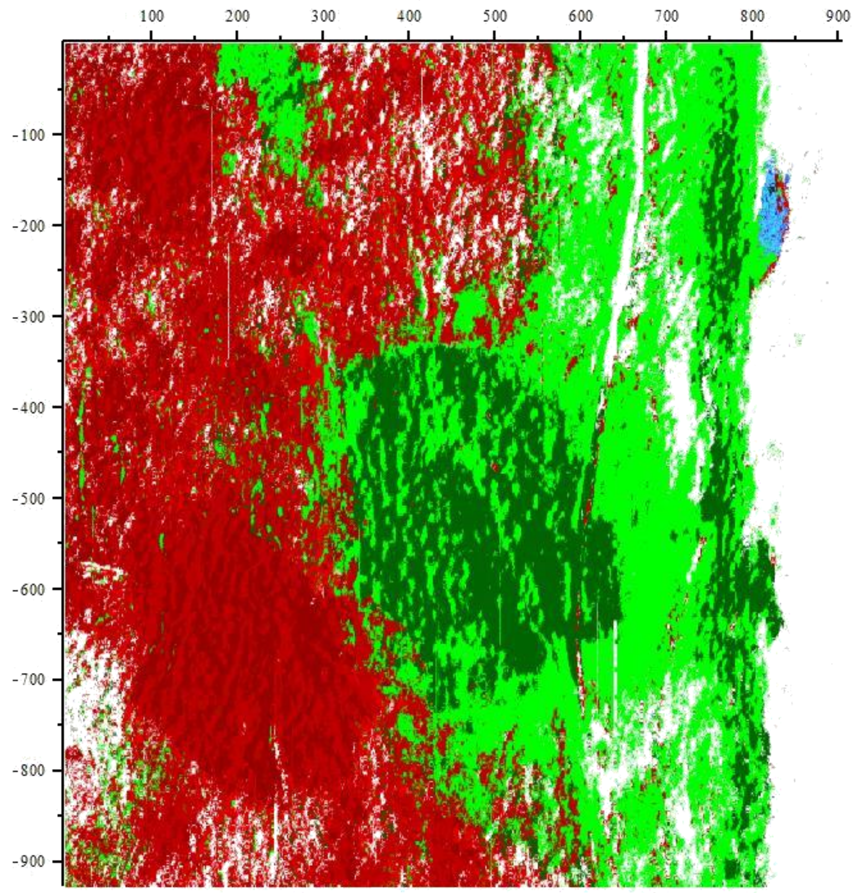
Further analysis of this data involved

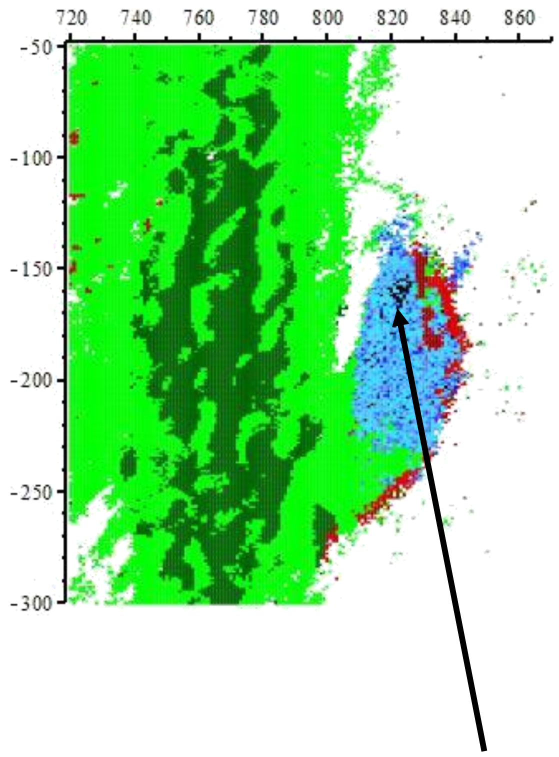
- Returning to original data and averaging all instances which belonged to distinct points and branches of the DQC convergence plot
- Comparing with known spectra of iron compounds
- Reanalysis of the blue component of the data, involving renewed DQC flow

This has led to

- Understanding of the two major components of the data
- Discovery of minute fractions of pure iron and magnetite in the blue component of the data

Locations of the different types of compounds on the relic fragment





The real needle in the haystack: 60 points out of 669,000

Conclusions from the XANES analysis

- The data-set is very large, noisy, and low-contrast . Nonetheless DQC can find small regions of interest because it is sensitive to small variations in the density of data-points.
- We have demonstrated the unexpected result that large, complex sets of data often contain non-trivial topological structures in their SVD space.
- Chemical results: The presence of a small metallic Fe cluster surrounded by a more oxidized (magnetite-like) phase was a surprise, and would have been impossible to extract without an unbiased and unsupervised search method like DQC.

Summary

Classification:
Linear separability – Perceptron learning.
Otherwise – MLP (ANN) using back-propagation algorithm.
The importance of low generalization errors.
SVM – alternative to ANN. Kernel methods.
Clustering:
Simplest method – k-means.
Analysis of large data sets requires preprocessing, e.g. by SVD (PCA).
Clustering using QM concepts: QC and DQC.

Resources of papers and programs.
http://horn.tau.ac.il for papers and some software tools, such as
http://horn.tau.ac.il/compact.html
the compact software provides matlab tools for clustering, including SVD, SVC and QC