Klassifikation 1+2

KognitiveSysteme,
15+17.05.2017

Klassification

- Einordnen in die Welt
  - Gesellschaftlich definierte Konventionen:
    - Buchstaben, menschliche Artifakte, Kredite
  - Biologisch definierte Kategorien:
    - Katze, Hund,
- Komplex
  - Was definiert einen Stuhl? Eine Katze?
  - Beziehungen zu Komplex, Regeln → Lernen
  - Nie 100% richtig → Wahrscheinlichkeit

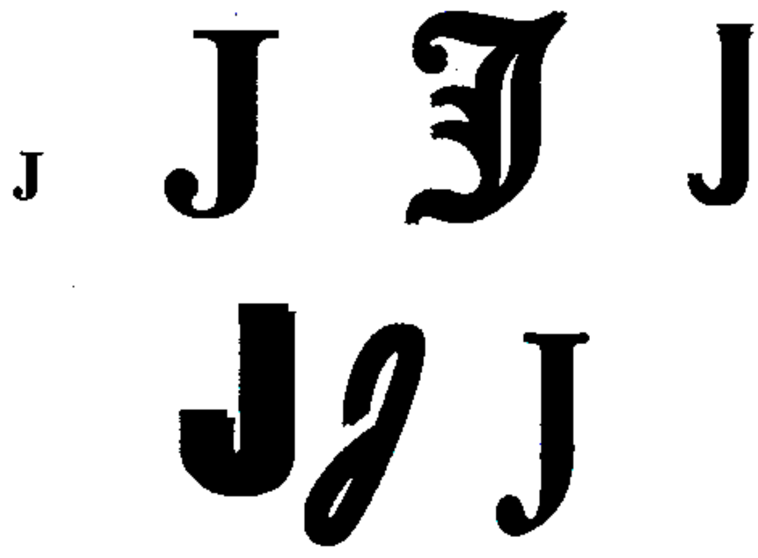
Schablonenanpassung (Template Matching)

- Eine einfache Form der Klassifikation
- Ziel: Ein Muster zu erkennen das einem abgespeicherten Beispiel ähnlich ist.
- Maß der Übereinstimmung zwischen Muster und Schablone (zentriert an (m,n) muss maximiert werden:

$$M_{f,g}(m,n) = \sum_i \sum_j f(i,j)g(i-m, j-n)$$

Beispiel Schablonenanpassung

A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z



Normalisierung der Helligkeit in der Bildverarbeitung

Karlsruher Institut für Technologie

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

J J J J

J J J

$$M_{f,g}(m,n) = \frac{1}{N-1} \sum_i \sum_j \frac{(f(i,j) - \bar{f})(g(i-m, j-n) - \bar{g})}{\sigma_f \sigma_g}$$

σ_f, σ_g : Standardabweichung von f und g

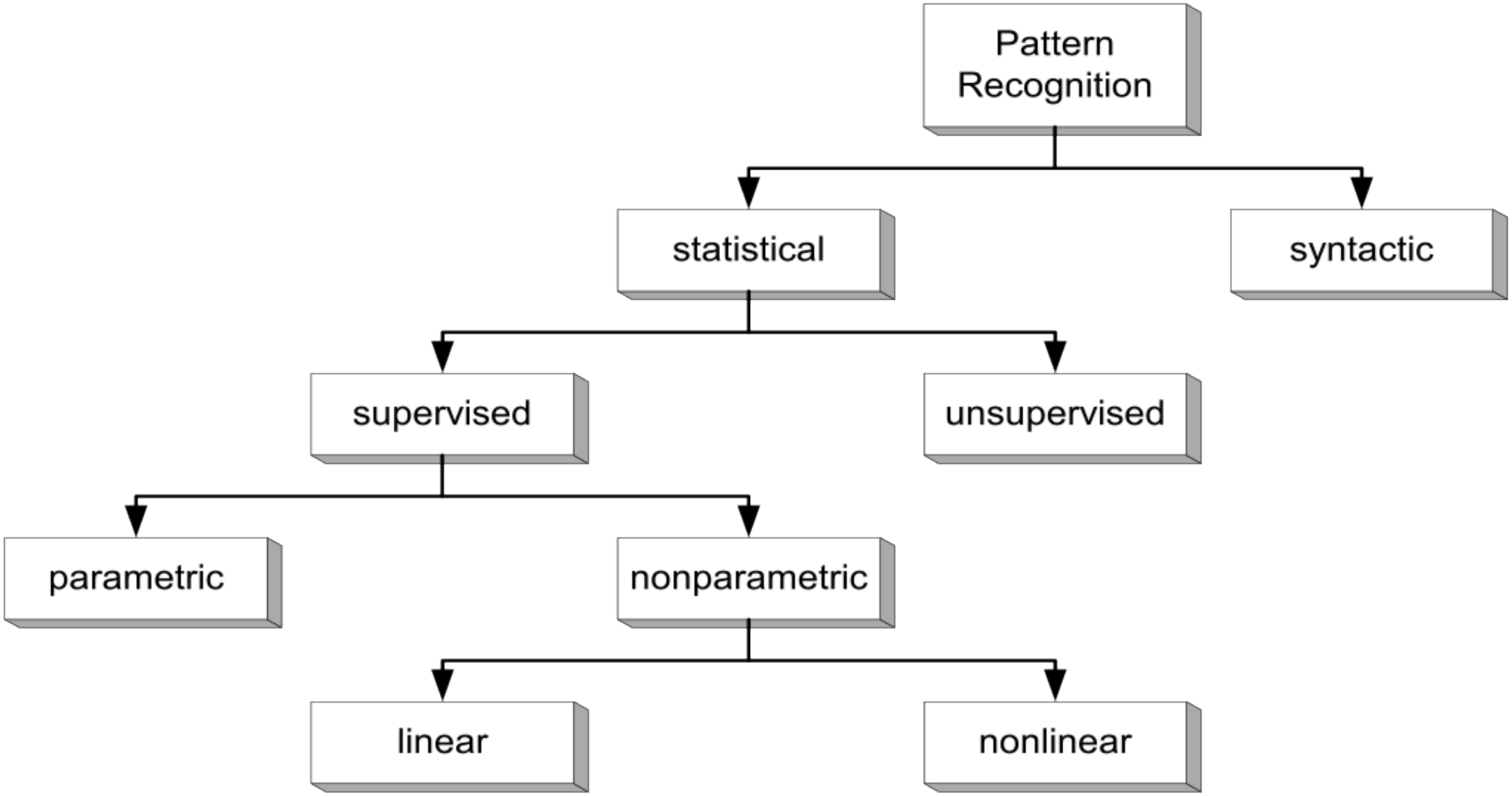
$\bar{f}, \bar{g}$: Mittelwert von f und g

N : Anzahl Pixel im Bild

Pattern Recognition Overview

- Static Patterns, no dependence on Time or Sequential Order
- Important Notions
  - Supervised - Unsupervised Classifiers
  - Parametric - Non-Parametric Classifiers
  - Linear - Non-linear Classifiers
- Classical Methods
  - Bayes Classifier
  - K-Nearest Neighbor
- Connectionist Methods
  - Perceptron
  - Multilayer Perceptrons

Pattern Recognition



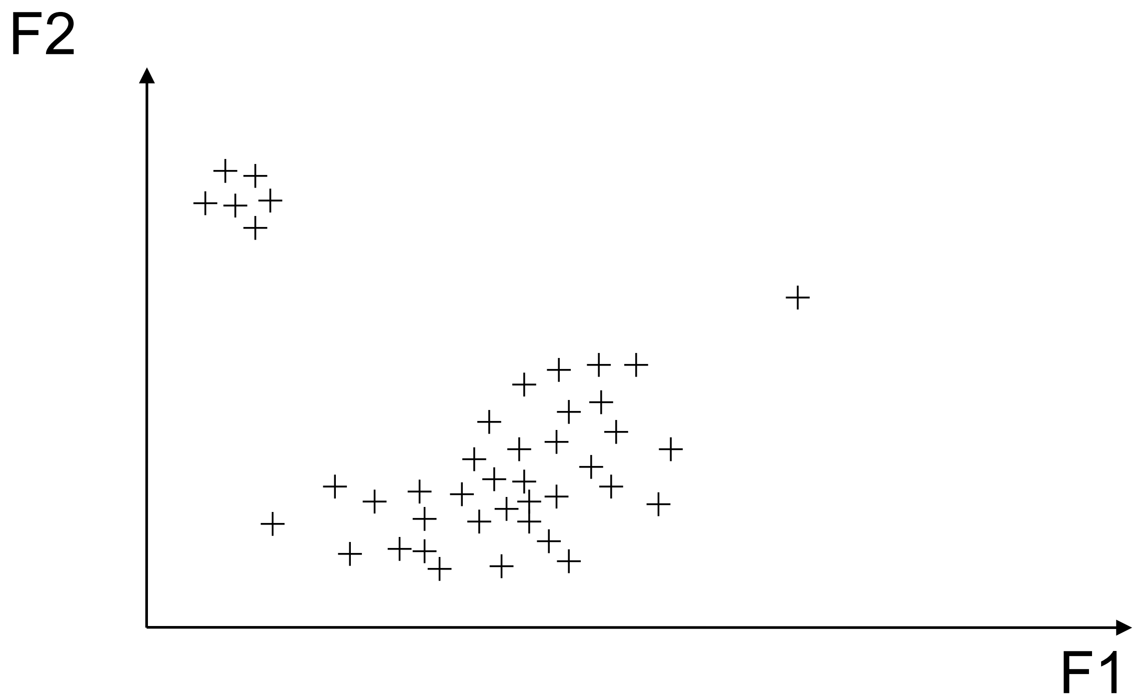
Supervised - Unsupervised

- Supervised training:
  Class to be recognized is known for each sample in training data. Requires a priori knowledge of useful features and knowledge/labeling of each training token (cost!).

- Unsupervised training:
  Class is not known and structure & features are to be discovered automatically.
  Feature-space-reduction,
  Knowledge Induction

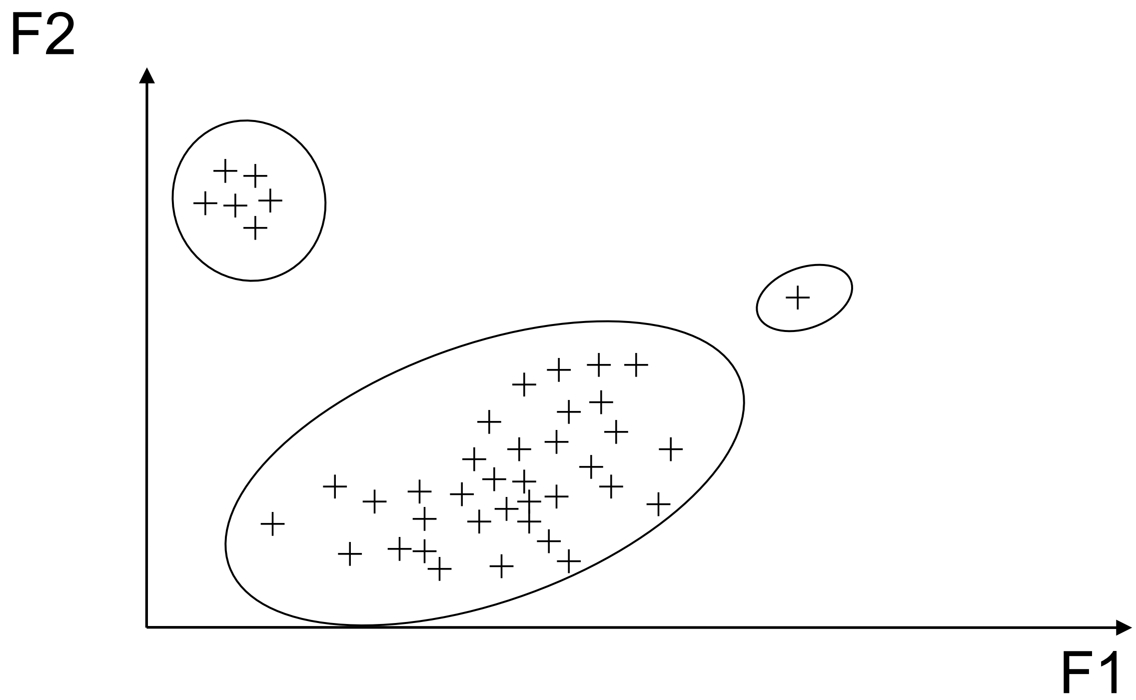
example: clustering, autoassociative nets, deep learning

Unsupervised Classification



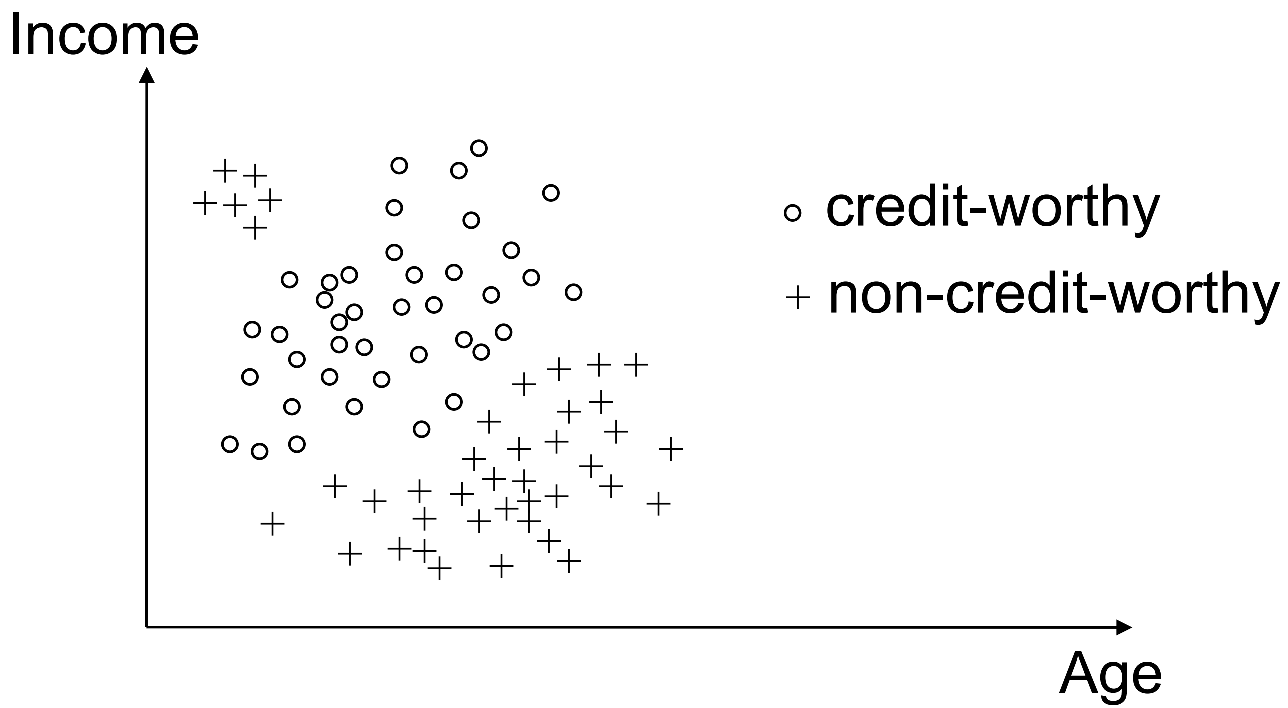
- Classification:
  - Classes Not Known: Find Structure

Unsupervised Classification



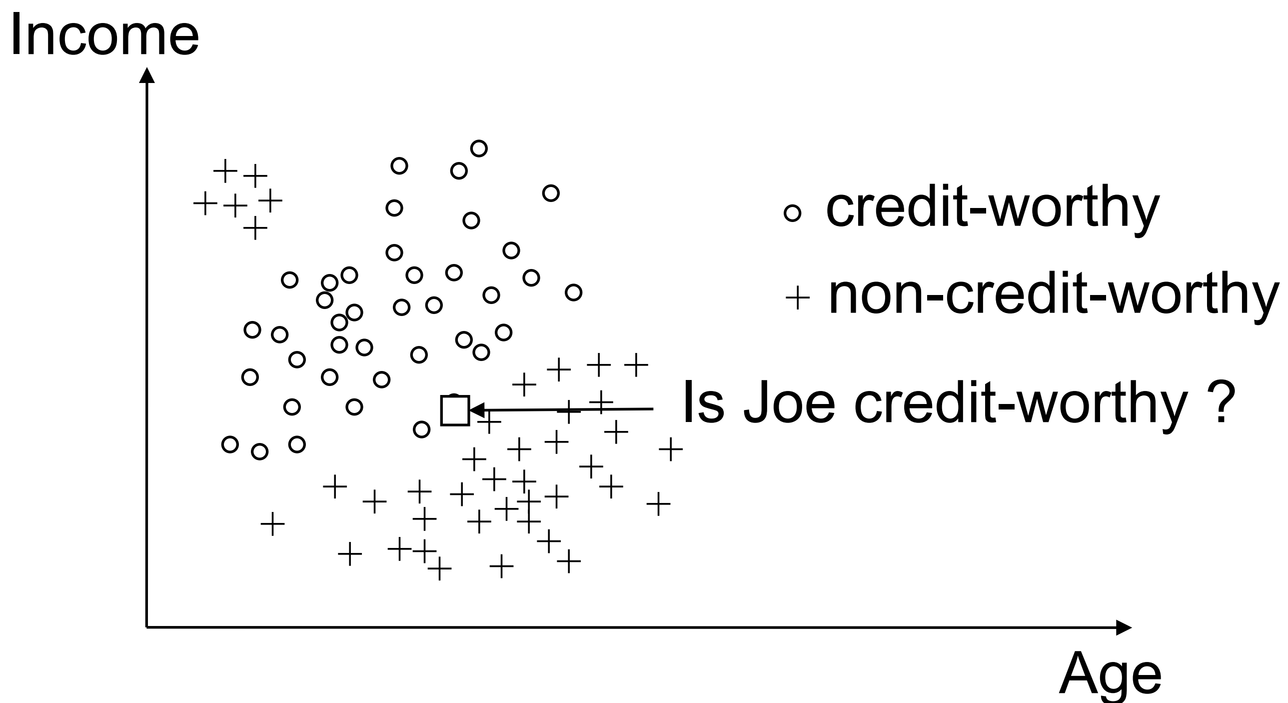
- Classification:
  - Classes Not Known: Find Structure
  - Clustering
  - How? How many?

Supervised Classification



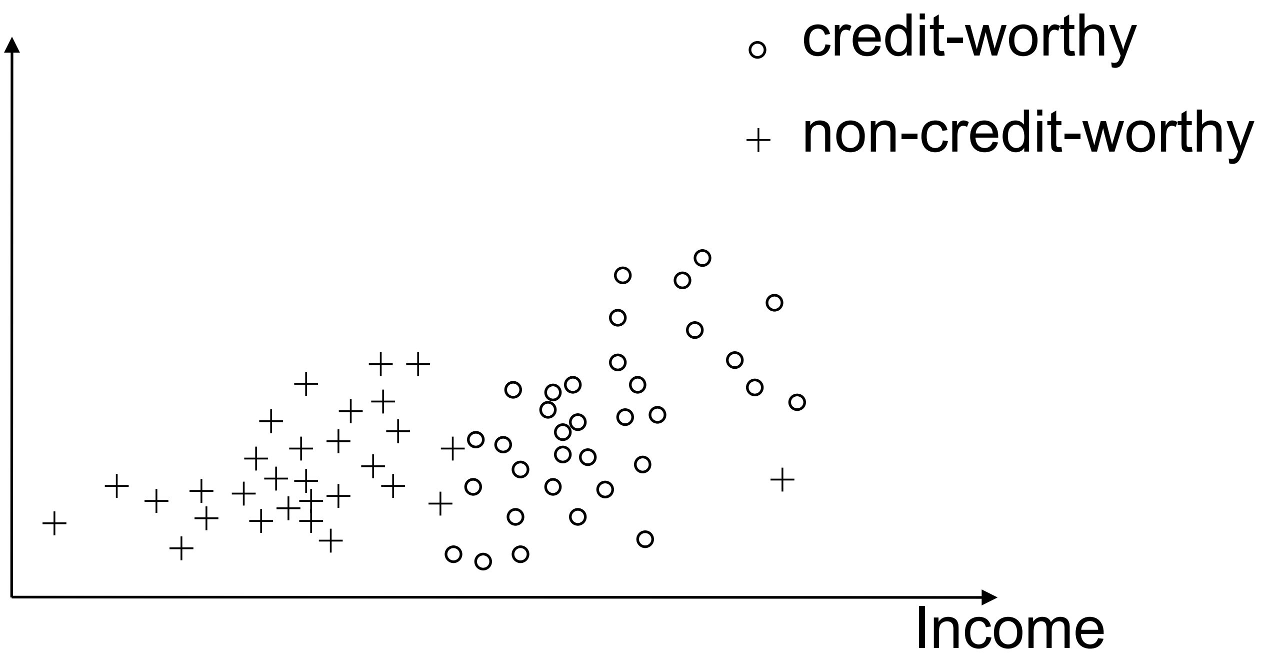
- Classification:
  - Classes Known: Creditworthiness: Yes-No
  - Features: Income, Age
  - Classifiers

Classification Problem

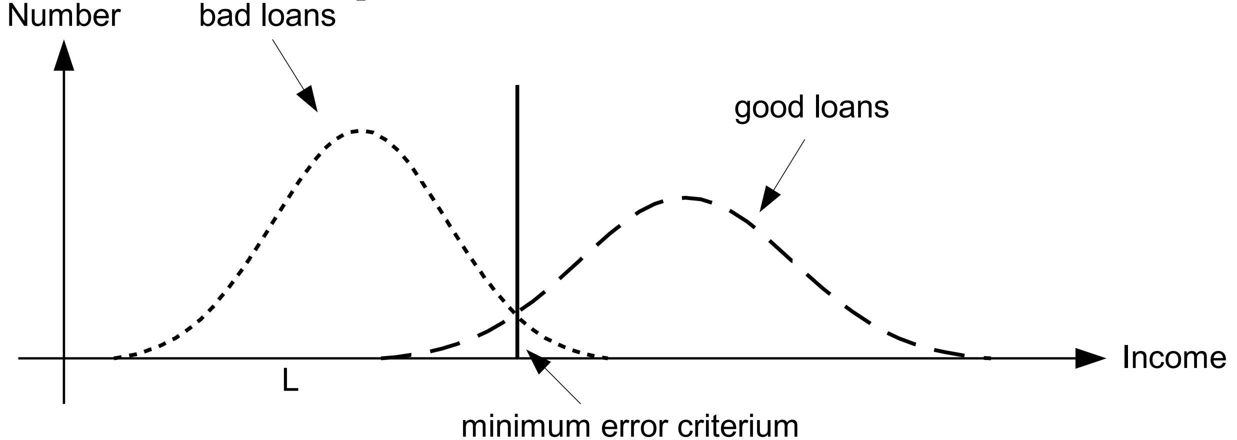


- Features: age, income $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- Classes: creditworthy, non-creditworthy
- Problem: Given Joe's income and age, should a loan be made?
- Other Classification Problems: Fraud Detection, Customer Selection...

Classification Problem



Parametric - Non-parametric



- Parametric:
  - assume underlying probability distribution;
  - estimate the parameters of this distribution.
  - Example: "Gaussian Classifier"
- Non-parametric:
  - Don't assume distribution.
  - Estimate probability of error or error citerion directly from training data.
  - Examples: Parzen Window, k-nearest neighbor, perceptron...

Bayes Decision Theory

Bayes Rule: $P(\omega_j / x) = \dfrac{p(x / \omega_j)P(\omega_j)}{p(x)}$

where $p(x) = \sum_j p(x / \omega_j)P(\omega_j)$

A priori probability $P(\omega_j)$

↓ observation of x

A posteriori probability $P(\omega_j / x)$

Class-conditional Probability Density $p(x / \omega_j)$

Two classes case:

$$P(\text{error} / x) = \begin{cases} P(\omega_1 / x) & \text{if we decide } \omega_2 \\ P(\omega_2 / x) & \text{else} \end{cases}$$

Error is minimized, if we:

Decide ω_1 if $P(\omega_1 / x) > P(\omega_2 / x)$;
ω_2 otherwise

Decide ω_1 if $p(x / \omega_1)P(\omega_1) > p(x / \omega_2)P(\omega_2)$;
ω_2 otherwise

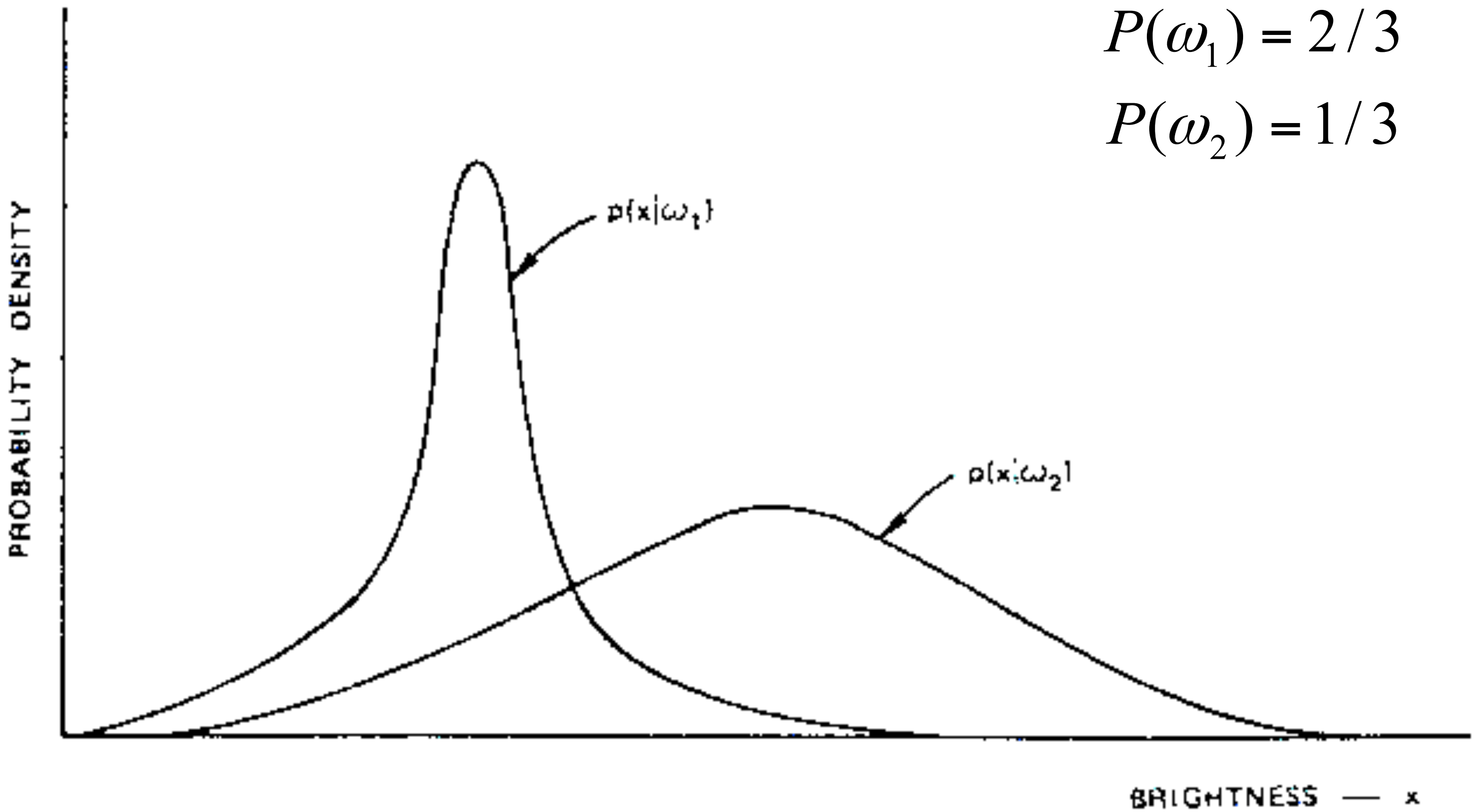
For the multicategory case:

Decide ω_i if $P(\omega_i / x) > P(\omega_j / x)$ for all $j \neq i$

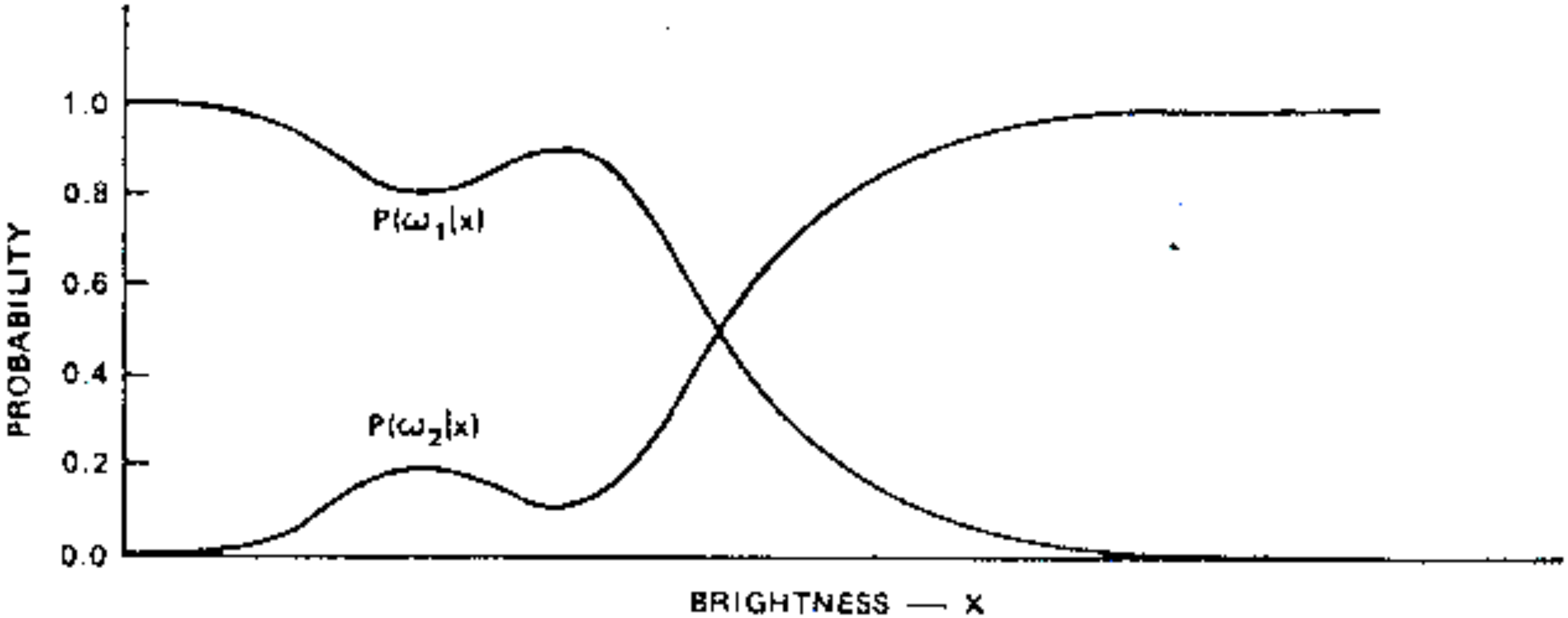
Hypothetical class-conditional probability density function

$$P(\omega_1) = 2/3$$

$$P(\omega_2) = 1/3$$



A posteriori probabilities



Classifier Discriminant Functions

$$g_i(x), i = 1, \ldots, c$$

Assign x to class ω_i , if $g_i(x) > g_j(x)$ for all $j \neq i$

$$g_i(x) = P(\omega_i / x)$$

$$= \frac{p(x / \omega_i) P(\omega_i)}{\sum_{j=1}^{c} p(x / \omega_j)(\omega_j)}$$

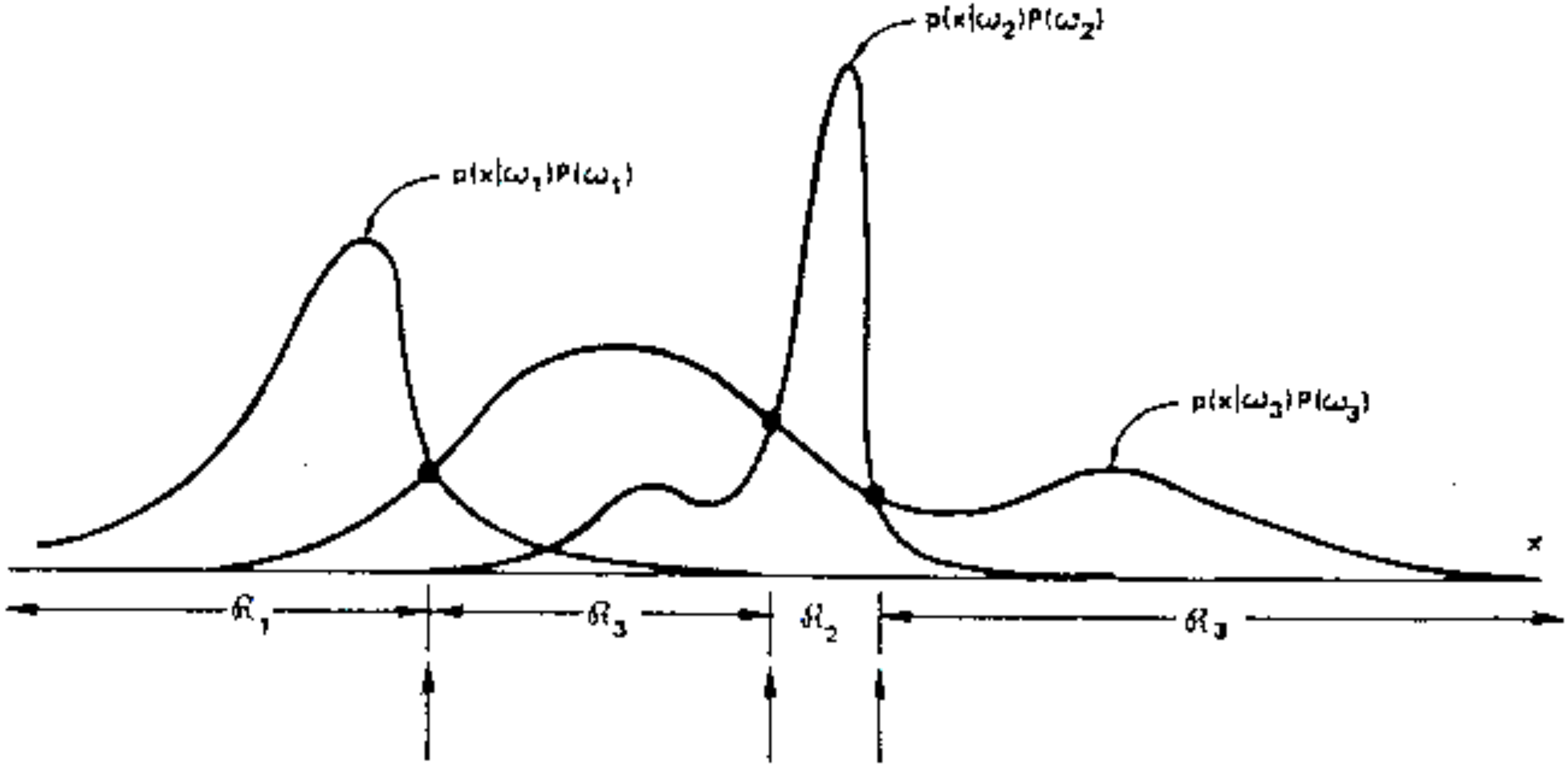
← independent of class i

$$g_i(x) = p(x / \omega_i) P(\omega_i)$$

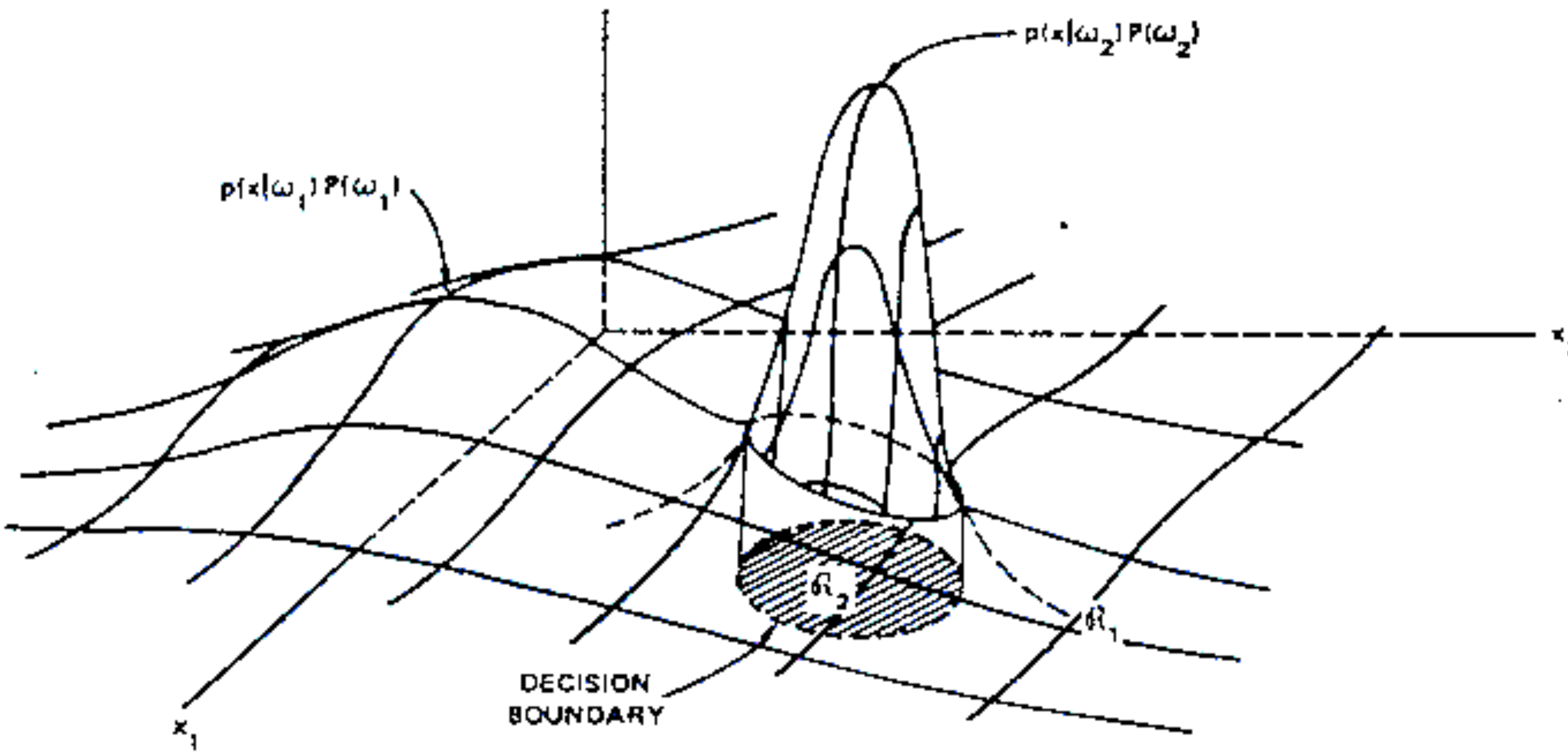
$$g_i(x) = \log(p(x / \omega_i)) + \log(P(\omega_i))$$

class conditional probability density function A priori probability

Examples Decision boundaries I



Examples Decision boundaries II



Classifier Design in Practice

- Need a priori probability $P(\omega_i)$ (not too bad)
- Need class conditional PDF $p(x / \omega_i)$
- Problems:
  - limited training data
  - limited computation
  - class-labelling potentially costly and errorful
  - classes may not be known
  - good features not known
- Parametric Solution:
  - Assume that $p(x / \omega_i)$ has a particular prametric form
  - Most common representative: multivariate normal density

GaussianClassifier

Univariate Normal Density: $p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma} \right)^2 \right]$

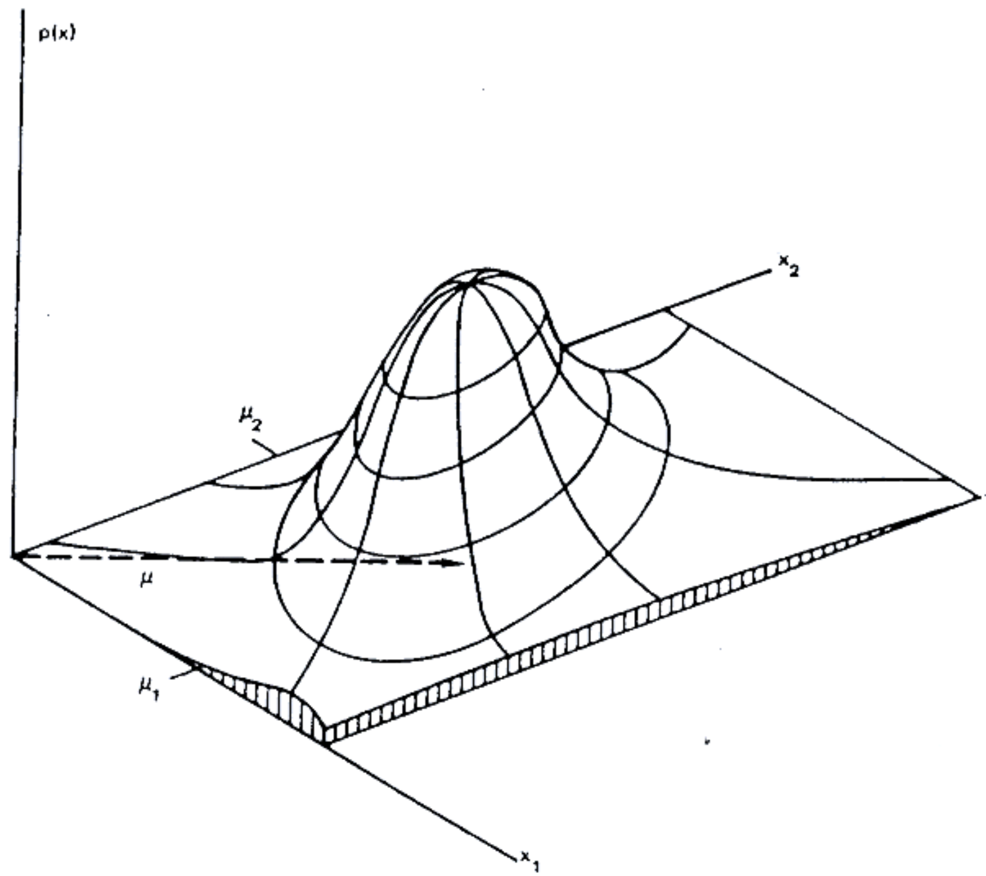
$$\sim N(\mu, \sigma^2)$$

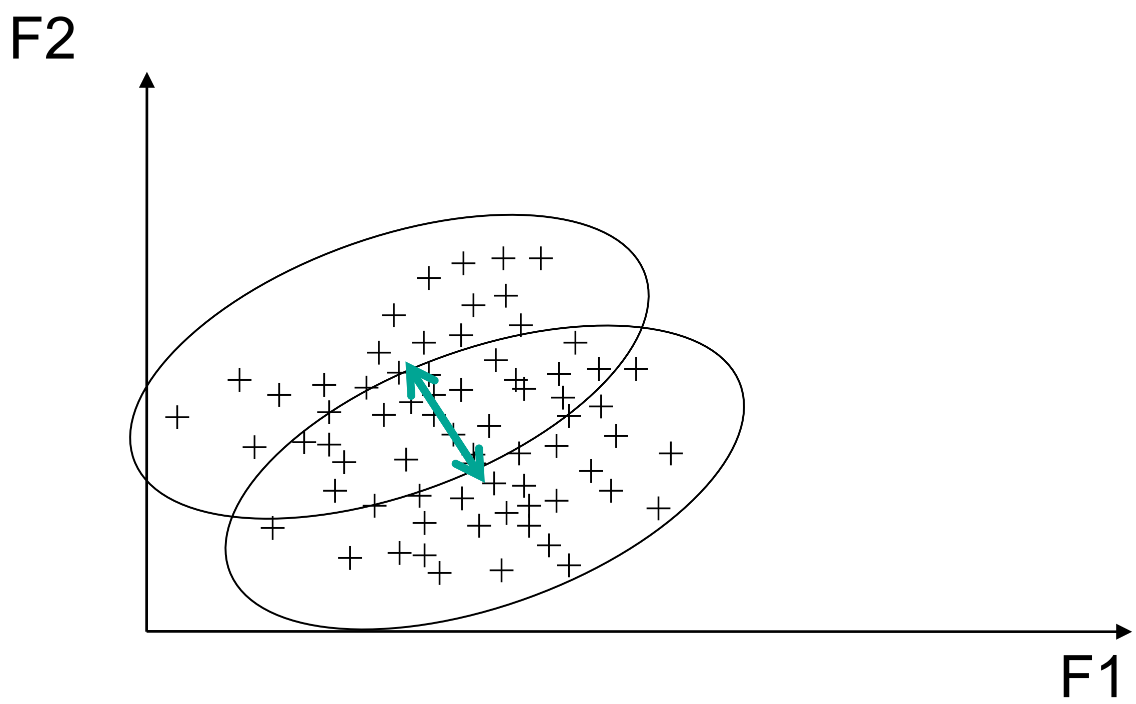
Multivariate Density: $p(\vec{x}) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2} (\vec{x} - \vec{\mu})^t \Sigma^{-1} (\vec{x} - \vec{\mu}) \right]$

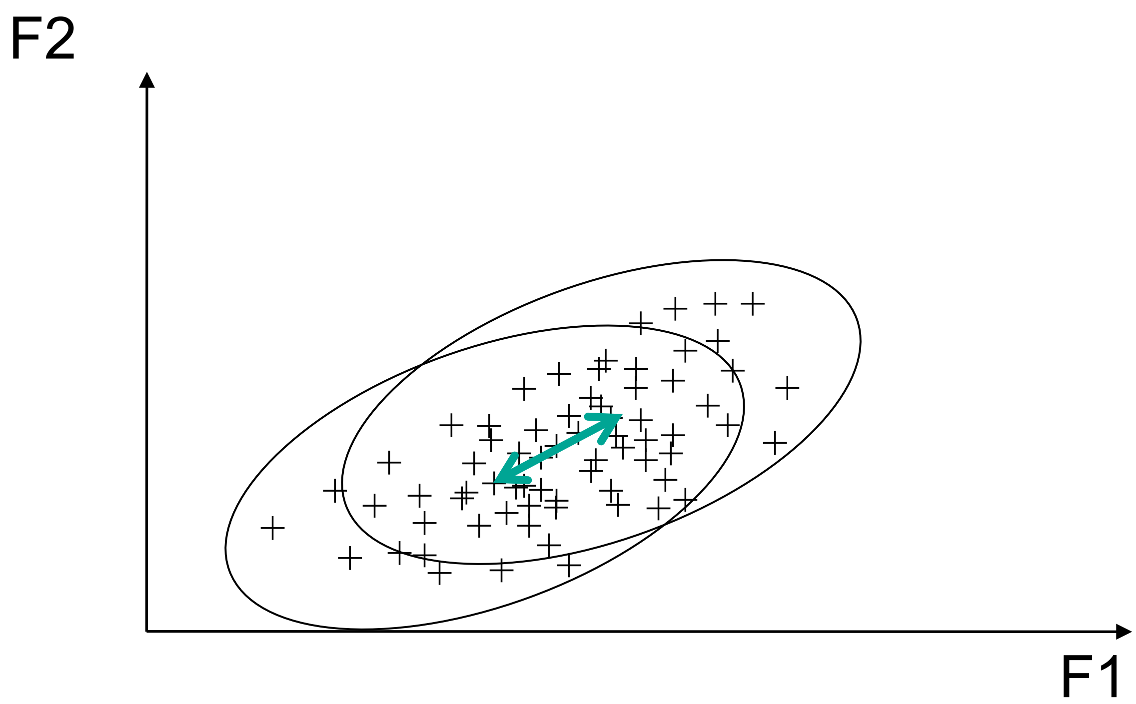
$$\sim N(\vec{\mu}, \Sigma)$$

$$g_i(\vec{x}) = -\frac{1}{2} (\vec{x} - \vec{\mu}_i)^t \Sigma_i^{-1} (\vec{x} - \vec{\mu}_i) - \frac{d}{2} \log(2\pi) - \frac{1}{2} \log|\Sigma_i| + \log P(\omega_i)$$

Bivariate Normal Density




F2
F1


F2
F1

Gaussian Classifier

- For each class i, need to estimate from trainingdata:
- covariance matrix Σ_{i}
- mean vector $\vec{\mu}_i$

Estimation of Parameters

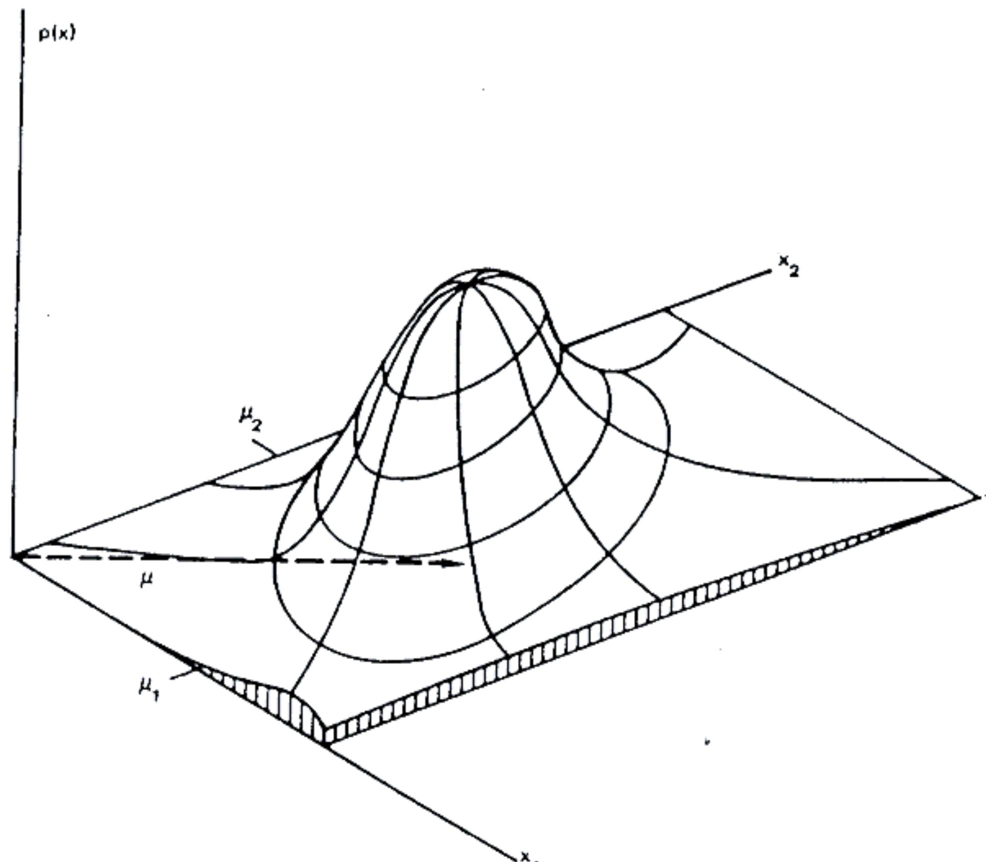
- MLE, Maximum Likelihood Estimation

- For Multivariate Case:

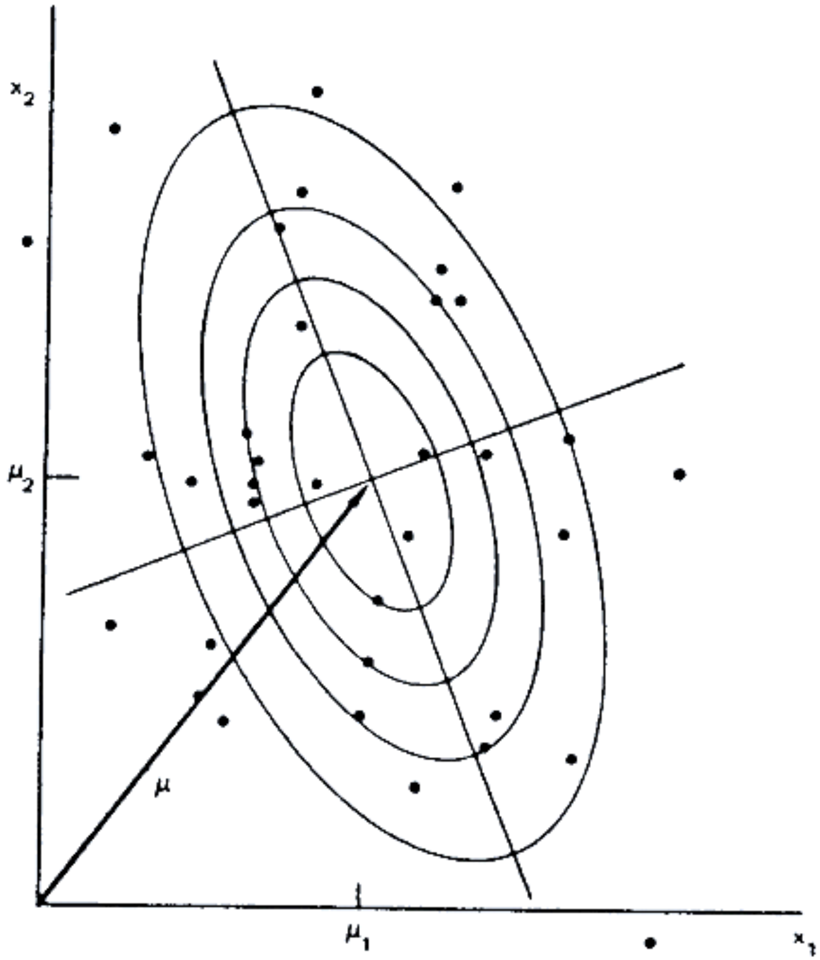
$$\vec{\mu}_i = \frac{1}{N}\sum_{k=1}^{N}\vec{x}_k$$

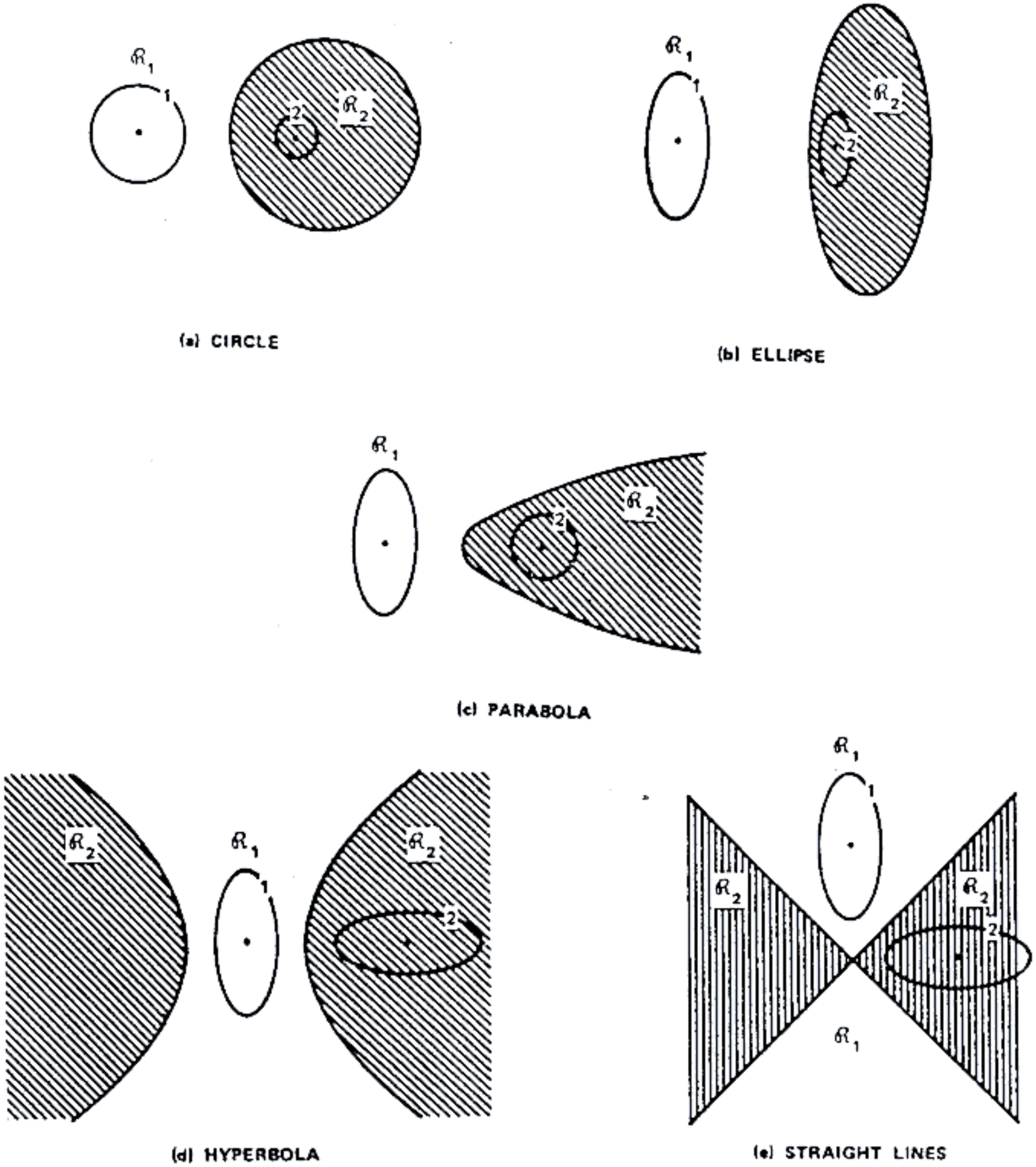
$$\Sigma = \frac{1}{N}\sum_{k=1}^{N}(\vec{x}_k - \vec{\mu})(\vec{x}_k - \vec{\mu})^T$$

Bivariate Normal Density



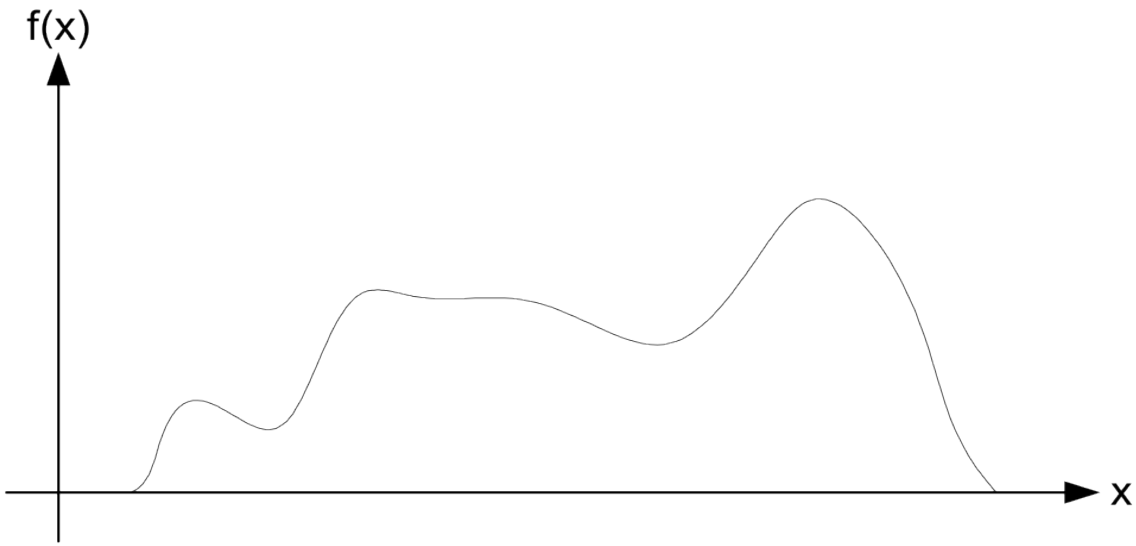
Scatter Diagram





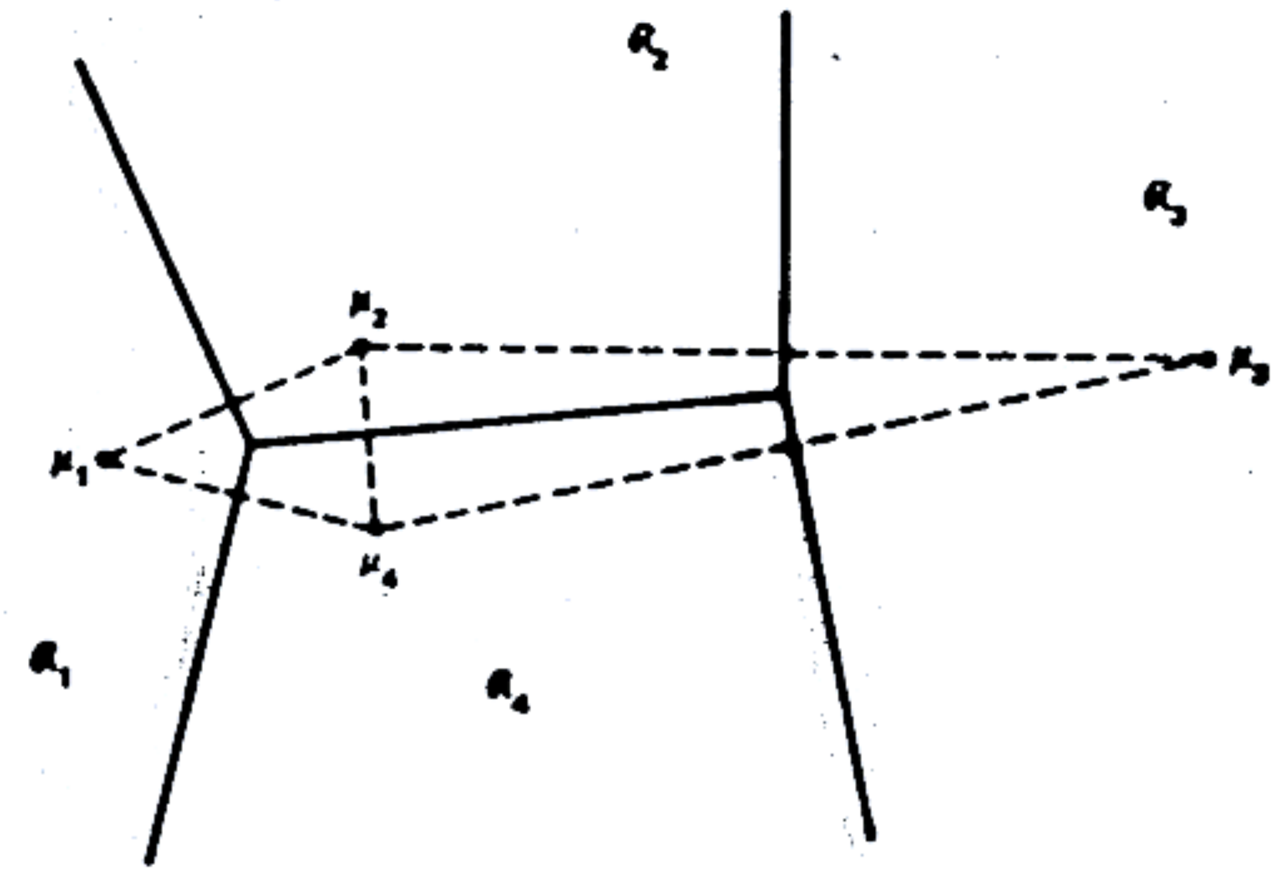
Forms for decision boundaries for the general bivariate normal case

Problems



- Normal distribution does not model this situation well.
- other densities may be mathematically intractable.
  –>non-parametric techniques

Decision Boundaries for a Minimum-Distance Classifier



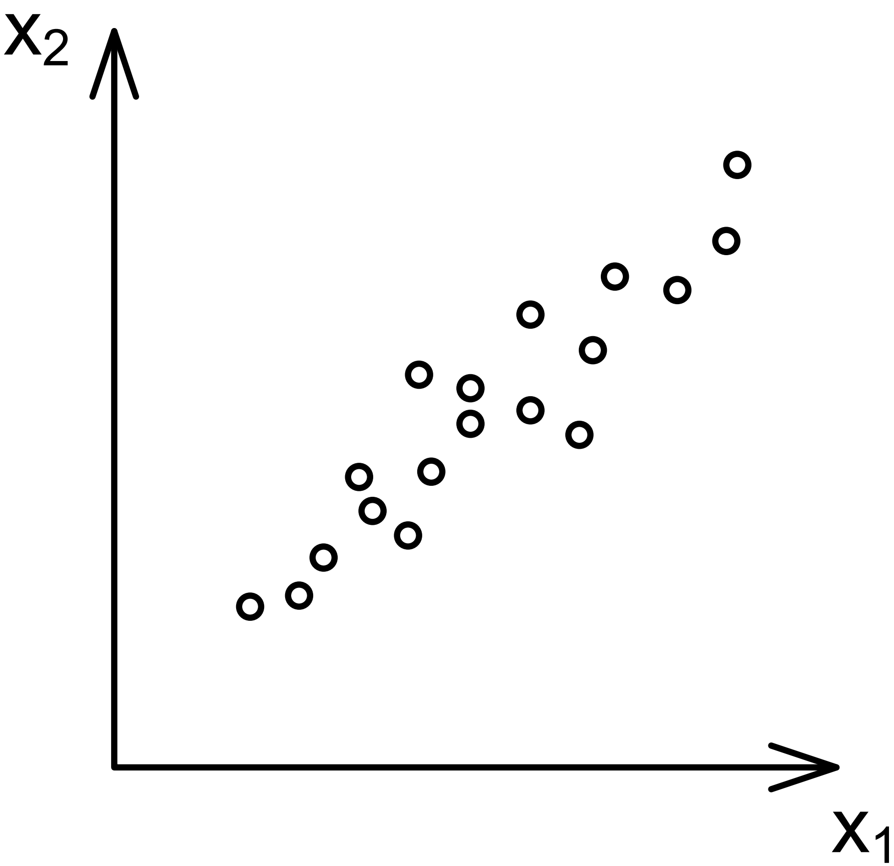
Problems of Classifier Design

- Features:
  - What and how many features should be selected?
  - Any features?
  - The more the better?
  - If additional features not useful (same mean and covariance), classifier will automatically ignore them?

Curse of Dimensionality

- Generally, adding more features indiscriminantly leads to worse performance!
- Reason:
  - Training Data vs. Number of Parameters (Dimensions!)
  - Limited training data.
- Solution:
  - select features carefully
  - reduce dimensionality
  - Principle Component Analysis
  - Learn Intermediate/Hidden Representations

Principal Component Analysis (PCA)



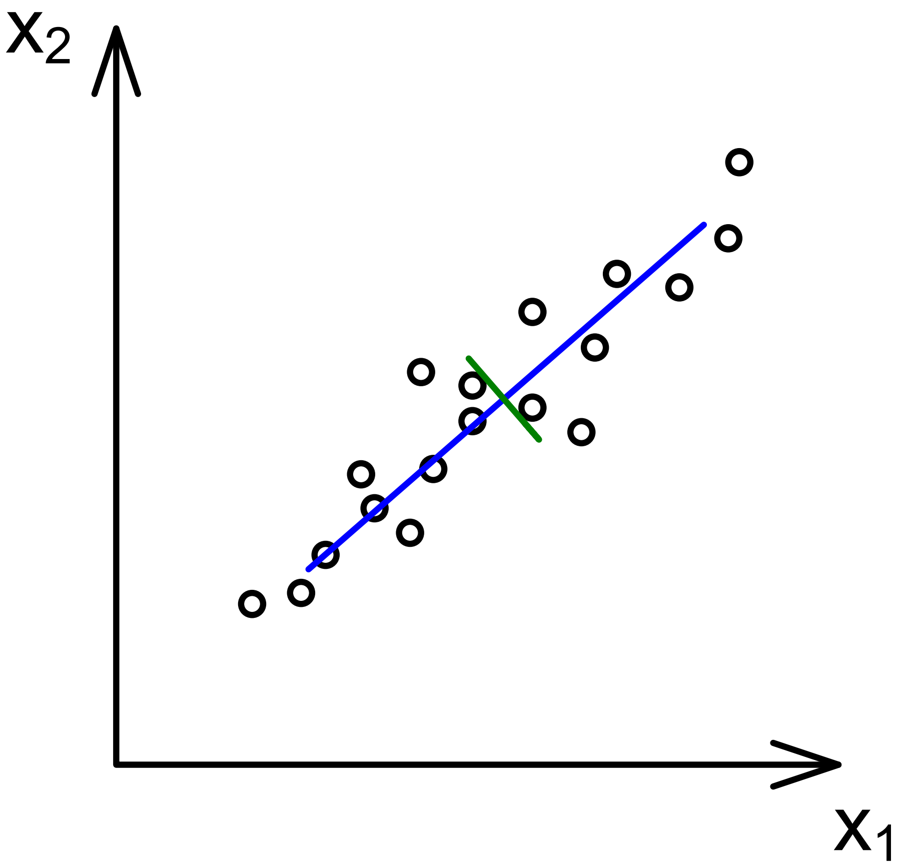
Assumption:

Single dimensions are correlated

Aim:

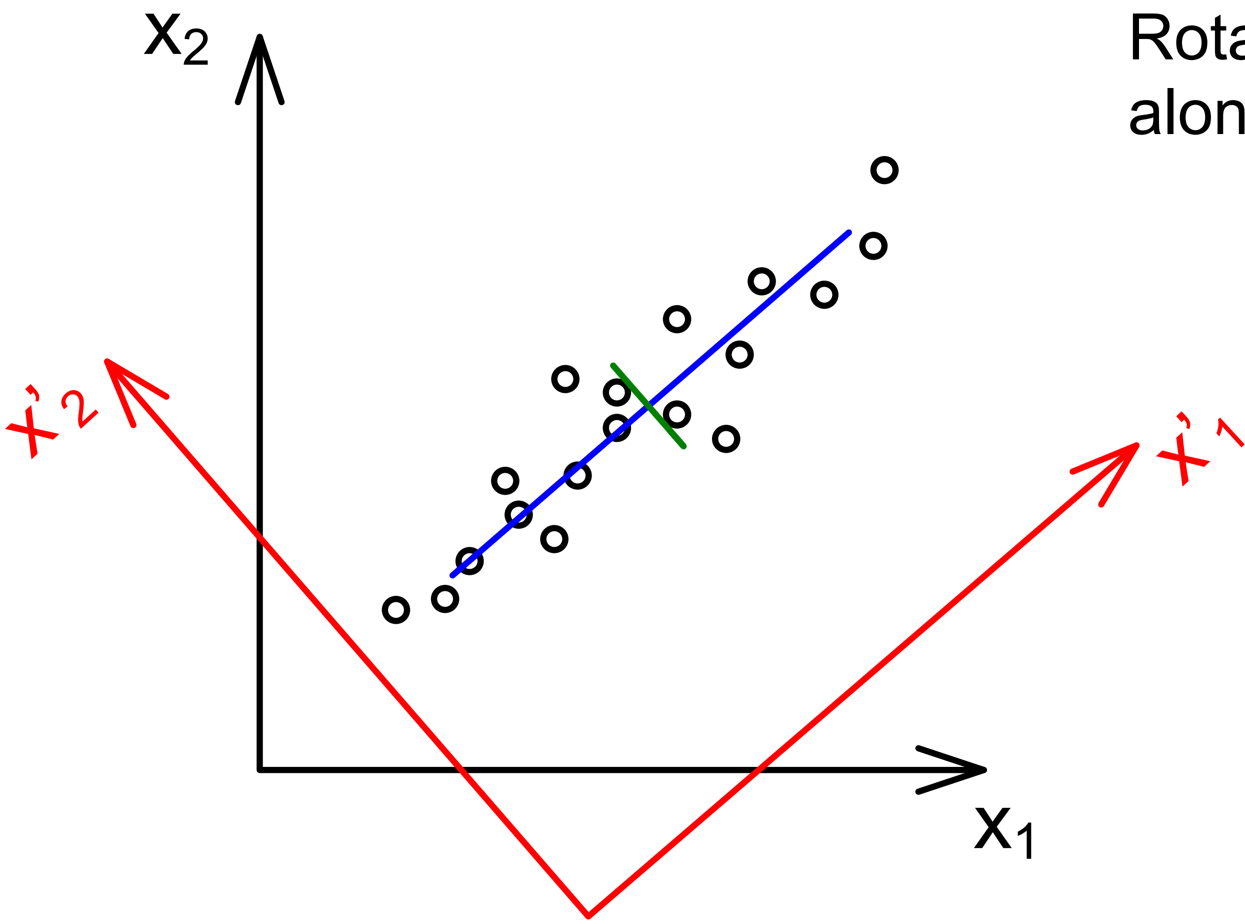
Reduce number of dimensions with minimum loss of information

Principal Component Analysis (PCA)



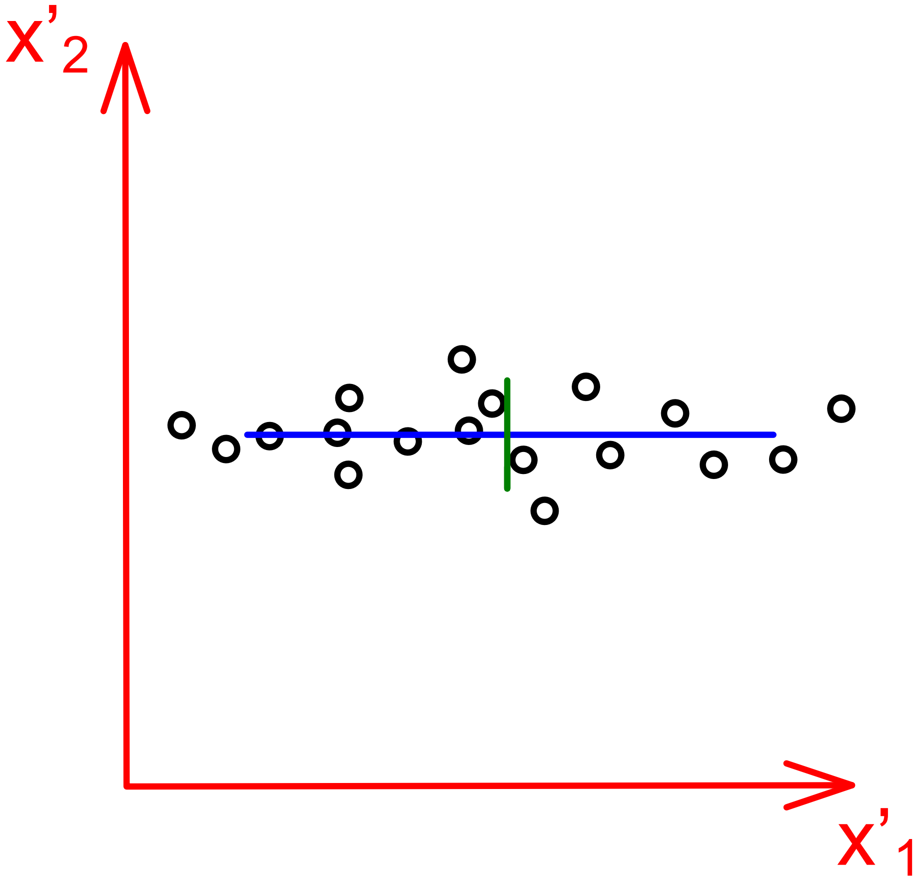
Find the axis along the highest variance

Principal Component Analysis (PCA)



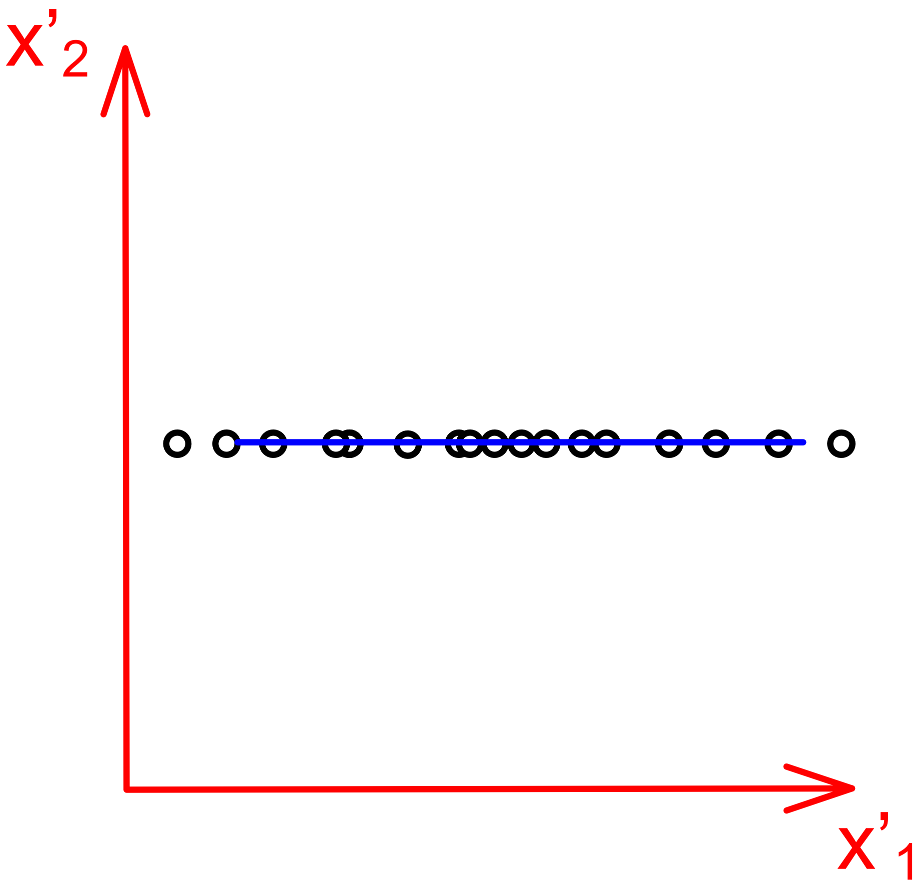
Rotate the space along the axis

Principal Component Analysis (PCA)



Dimensions are uncorrelated now

Principal Component Analysis (PCA)



Remove dimensions with low variance

=> Reduction of dimensionality with minimum loss of information

Risk

- May refuse to make decision in ambiguous cases (–> margin)
- Estimate cost of each decision (some more costly than others)

$$\Omega = \{\omega_1, \ldots, \omega_s\}$$ s states of nature

$$A = \{\alpha_1, \ldots, \alpha_a\}$$ a possible actions

Risk

Loss function

$\lambda(\alpha_i / \omega_j)$ loss of action α_i , given state ω_j

$$P(\omega_j / \vec{x}) = \frac{P(\vec{x} / \omega_j) P(\omega_j)}{P(\vec{x})}$$

Expected loss of taking action α_i

$$R(\alpha_i / \vec{x}) = \sum_{j=1}^{s} \lambda(\alpha_i / \omega_j) P(\omega_j / \vec{x})$$

↑ conditional risk

Risk

$$R(\alpha_i / \vec{x}) = \sum_{j=1}^{s} \lambda(\alpha_i / \omega_j) P(\omega_j / \vec{x})$$

Minimize expected loss by selecting action α_i that minimizes conditional risk.

<u>Two category case:</u>

$$\lambda(\alpha_i / \omega_j) \hat{=} \lambda_{ij}$$

$$R(\alpha_1 / \vec{x}) = \lambda_{11} P(\omega_1 / \vec{x}) + \lambda_{12} P(\omega_2 / \vec{x})$$

$$R(\alpha_2 / \vec{x}) = \lambda_{21} P(\omega_1 / \vec{x}) + \lambda_{22} P(\omega_2 / \vec{x})$$

Risk

decide ω_1 if $R(\alpha_1 / \vec{x}) < R(\alpha_2 / \vec{x})$

decide ω_1 if $(\lambda_{21} - \lambda_{11}) P(\omega_1 / \vec{x}) > (\lambda_{12} - \lambda_{22}) P(\omega_2 / \vec{x})$

decide ω_1 if $(\lambda_{21} - \lambda_{11}) P(\vec{x} / \omega_1) P(\omega_1) > (\lambda_{12} - \lambda_{22}) P(\vec{x} / \omega_2) P(\omega_2)$

decide ω_1 if $$\frac{p(\vec{x} / \omega_1)}{p(\vec{x} / \omega_2)} > \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \cdot \frac{P(\omega_2)}{P(\omega_1)}$$

↑ Likelihood Ratio

↑ independent of x

Minimum Error Rate Classification

- Decision rule to minimize error rate
- Define zero-one loss function

$$\lambda(\alpha_i / \omega_j) = \begin{cases} 0 & : i = j \\ 1 & : i \neq j \end{cases} \quad i, j = 1..c$$

$$R(\alpha_i / \vec{x}) = \sum_{j=1}^{c} \lambda(\alpha_i / \omega_j) p(\omega_j / \vec{x})$$

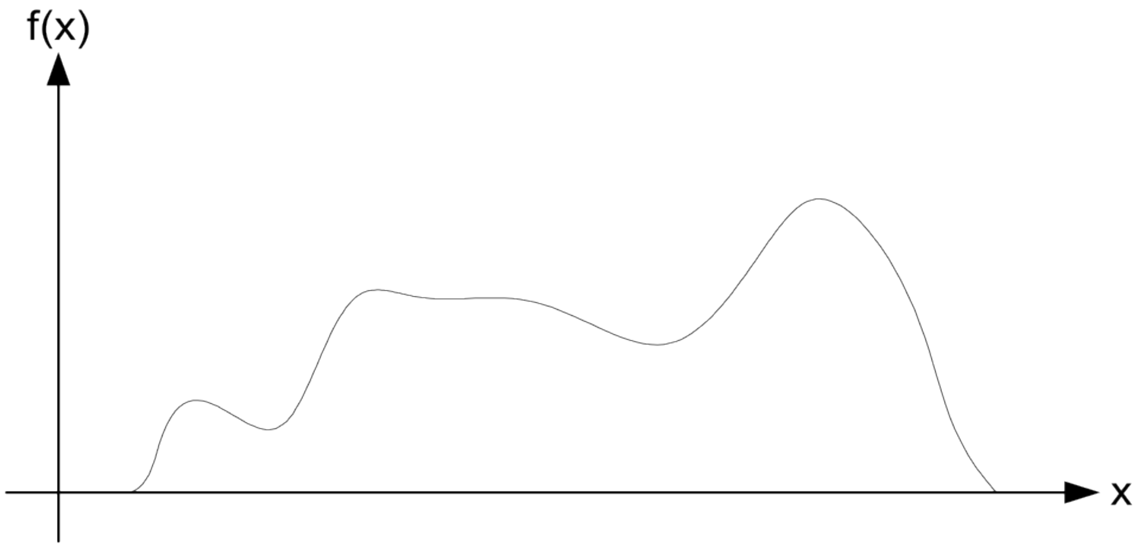
$$= \sum_{j \neq i} p(\omega_j / \vec{x})$$

$$= 1 - p(\omega_i / \vec{x})$$

Minimum Error Rate Classification

- To minimize risk and the average probabililty of error, select i that maximizes posterior $p(\omega_i / \vec{x})$
- Decide ω_i if $p(\omega_i / \vec{x}) > p(\omega_j / \vec{x})$ for all $j \neq i$

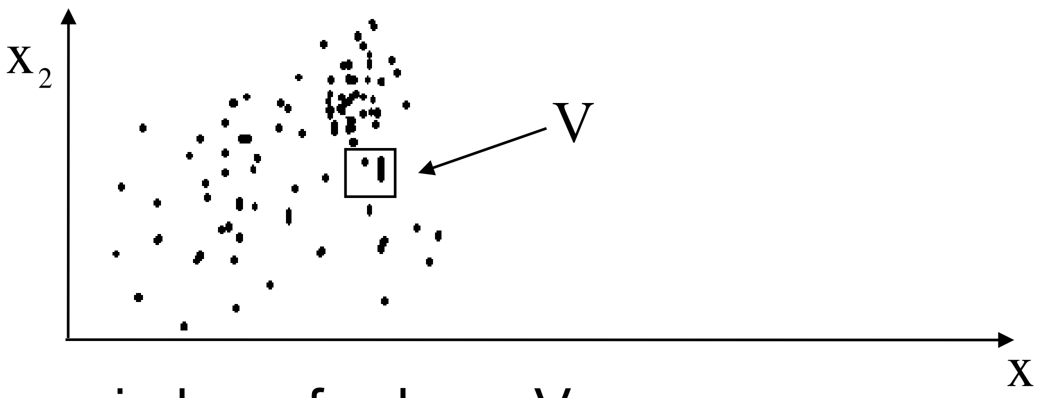
Problems



- Normal distribution does not model this situation well.
- other densities may be mathematically intractable.
  –>non-parametric techniques

Non-Parametric Techniques: Parzen Windows

- No Assumptions about the distribution are made estimate p(x) directly from data



- Choose a window of volume V
- Count the number of samples that fall inside the window.

$$p(x) \approx \frac{k/n}{V}$$

- :k = count
  :n = number of samples

Parzen Windows

- Problem:
  Volume too large -> loose resolution
  Volume too small -> erratic, poor estimate

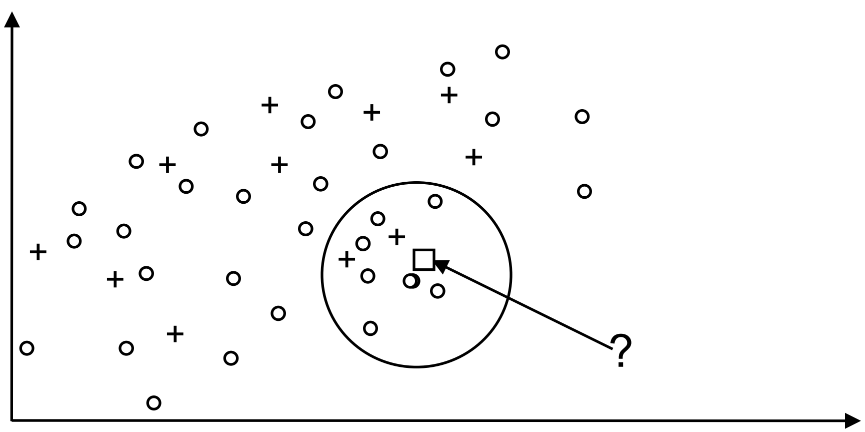
- set $V_n = \frac{1}{\sqrt{n}}$

$$\left.\begin{array}{l} \lim\limits_{n\to\infty} V_n = 0 \\ \lim\limits_{n\to\infty} k_n = \infty \\ \lim\limits_{n\to\infty} k_n / n = 0 \end{array}\right\} p_{guess}(x) \to p(x)$$

K-Nearest Neighbors (KNN)

- Idea: Let the volume be a function of the data. Include k-nearest neighbors in estimate.
  set $k = \sqrt{n}$ k-nearest neighbor rule for classification
  To classify sample x:
  - Find k-nearest neighbors of x.
  - Determine the class most frequently represented among those k samples (take a vote)
  - Assign x to that class.

k = 9
7 o
2 +
$\Rightarrow$ classify o



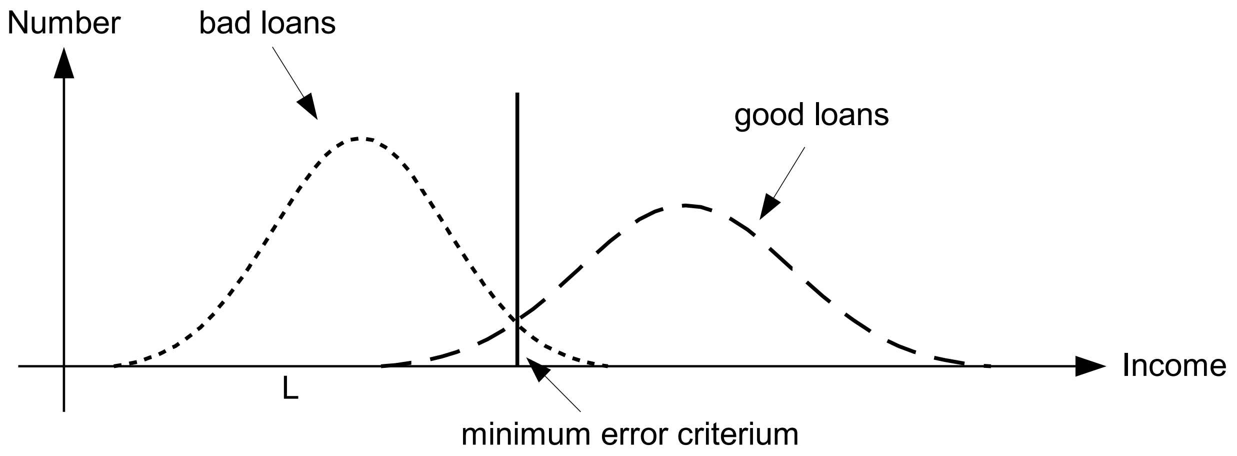
KNN-Classifier: Problem

- For finite number of samples n,
  we want k to be:
  large for reliable estimate
  small to guarantee that all k neighbors
  are reasonably close.

- Need training database to be larger.

  "There is no data like more data."

Parametric - Non-parametric



- Parametric:
  - assume underlying probability distribution;
  - estimate the parameters of this distribution.
  - Example: "Gaussian Classifier"
- Non-parametric:
  - Don't assume distribution.
  - Estimate probability of error or error citerion directly from training data.
  - Examples: Parzen Window, k-nearest neighbor, perceptron...

Decision Function g(x)

$g(\vec{x}) > 0 \quad \Rightarrow$ Class A

$g(\vec{x}) < 0 \quad \Rightarrow$ Not class A

$g(\vec{x}) = 0 \quad \Rightarrow$ No decision

$$g(\vec{x}) = \sum_{i=1}^{n} w_i x_i + w_0 = \vec{w}^T \vec{x} + w_0$$

$\vec{x} = (x_1, \ldots, x_n)^T$ Feature vector

$\vec{w} = (w_1, \ldots, w_n)^T$ Weight vector

w_0 Threshold weight

Linear Discriminant Functions

- No assumption about distributions (non-parametric)
- Linear Decision Surfaces
- Begin by supervised training (given classes of training data)
- Discriminant function:

$$g(x) = w_0 + \sum_{i=1}^{n} w_i x_i = \sum_{i=0}^{n} w_i x_i; \qquad x_0 = 1$$

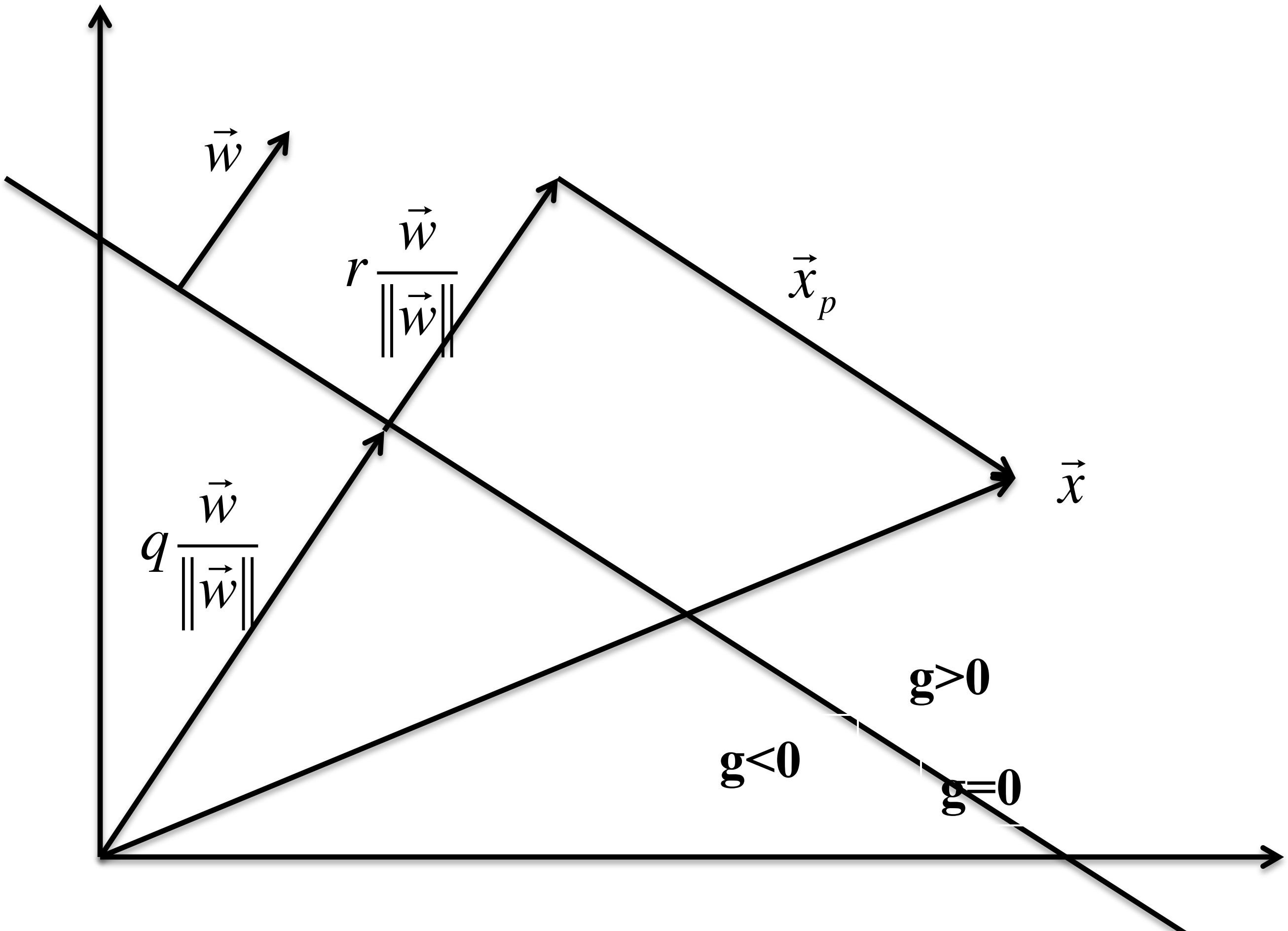
- Two category case:

$$g_1(x) > 0 \quad \Rightarrow \quad \text{class } 1$$

$$g_1(x) < 0 \quad \Rightarrow \quad \text{class } 2$$

- g(x) gives distance from decision surface…

Linear Discriminant Function I



$$g(\vec{x}) = \vec{w}^T q\frac{\vec{w}}{\|\vec{w}\|} + \vec{w}^T r\frac{\vec{w}}{\|\vec{w}\|} + \vec{w}\vec{x}_p + w_0 = -w_0 + r\|\vec{w}\| + w_0$$

Linear Discriminant Functions II

On Hyperplane H: $g(\vec{x}) = \sum_{i=1}^{n} w_1 x_1 + w_0 = \vec{w}^T \vec{x} + w_0 = 0$

$$\Rightarrow \quad \vec{x} = q \frac{\vec{w}}{\|\vec{w}\|} + r \frac{\vec{w}}{\|\vec{w}\|} + \vec{x}_p$$

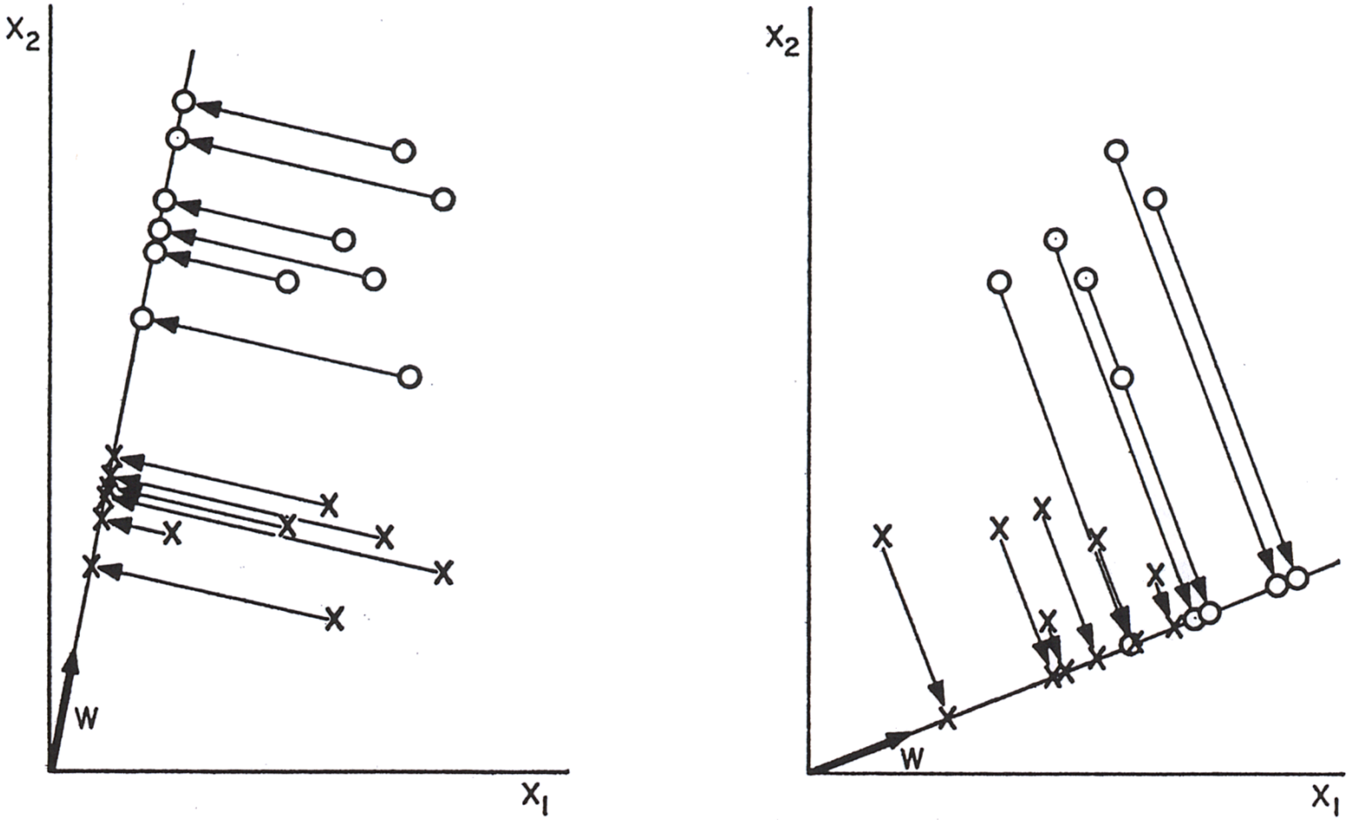
Since vector $q \frac{\vec{w}}{\|\vec{w}\|}$ is on hyperplance:

$$g\left(q \frac{\vec{w}}{\|\vec{w}\|} \right) = 0 = q\|\vec{w}\| + w_0 \Rightarrow \boxed{q = -\frac{w_0}{\|\vec{w}\|}}$$

And with $g(\vec{x}) = \vec{w}^T q \frac{\vec{w}}{\|\vec{w}\|} + \vec{w}^T r \frac{\vec{w}}{\|\vec{w}\|} + \vec{w}\vec{x}_p + w_0 = -w_0 + r\|\vec{w}\| + w_0$

We get $\boxed{g(\vec{x}) = r\|\vec{w}\|}$ => g is distance from hyperplane

Discriminant Analysis



Goal: find an orientation of the line, were the projected samples are well separated.

Fisher-Linear Discriminant

- Dimensionality Reduction;
- Project a set of multidimensional points onto a line $y = \vec{w}\vec{x}$
- Fisher Discriminant is function that maximizes criterion

$$g(x) = \frac{\left|\tilde{m}_1 - \tilde{m}_2\right|}{\tilde{s}_1 + \tilde{s}_2}$$

$$\tilde{m}_i = \frac{1}{n_i} \sum_{y \in Y_i} y$$ where sample mean for projected

$$\tilde{s}_i^{\,2} = \sum_{y \in Y_i} (y - \tilde{m}_i)^2$$ samples scatter for projected samples

Fisher Linear Discriminant

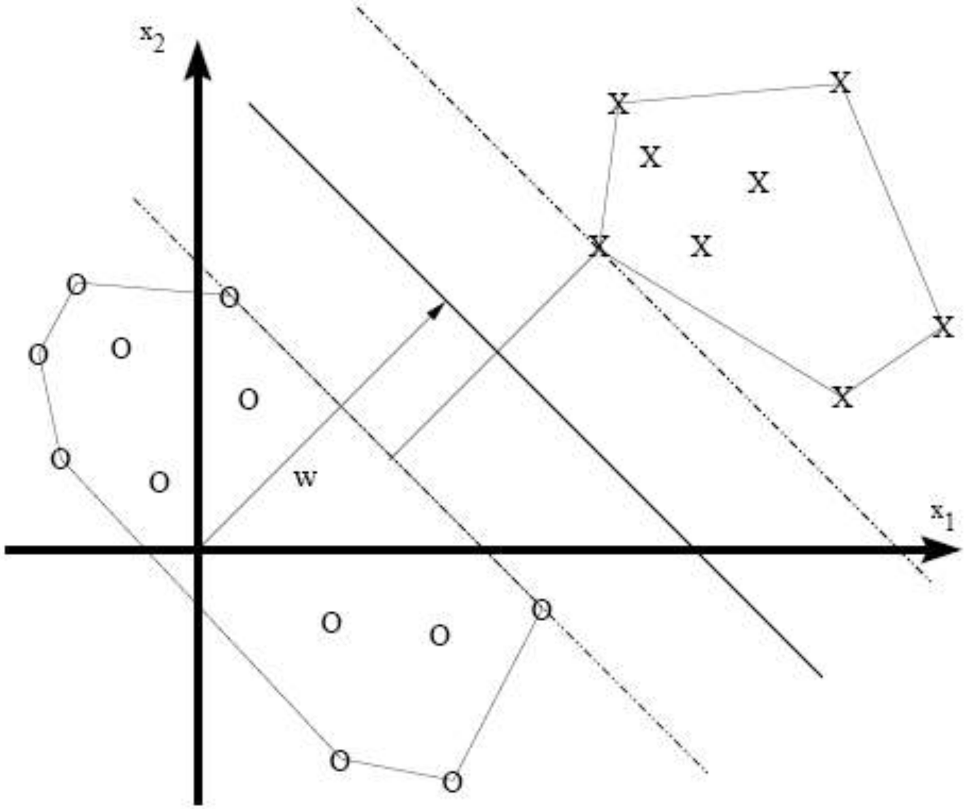
Fisher's linear discriminant:

$$\vec{w} = s_w^{-1}(\vec{m}_1 - \vec{m}_2)$$

$$s_w = s_1 + s_2$$ (within-class scatter matrix)

$$s_i = \sum_{x \in X_i} (\vec{x} - \vec{m}_i)(\vec{x} - \vec{m}_i)^T$$

Linear Separable

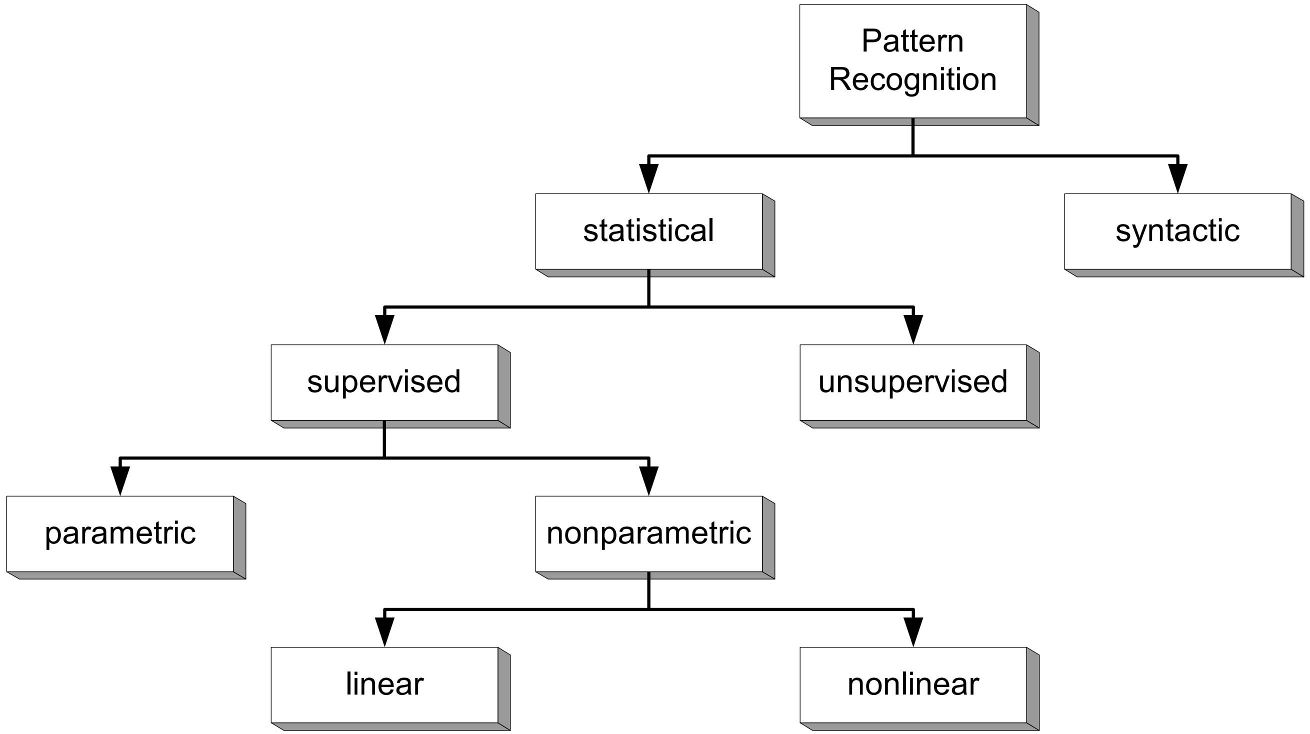


- Each class can be surounded by a convex polygon
- Maximum „safety“ area is half of the distance of the polygons

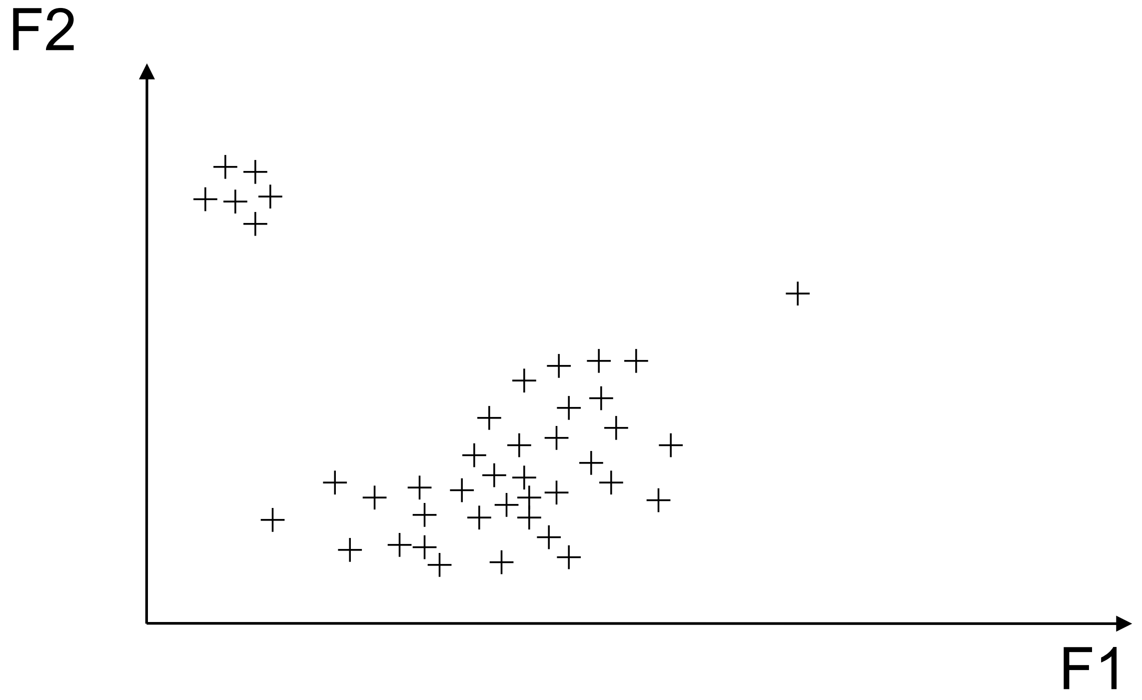
Unsupervised Learning

Alex Waibel

Pattern Recognition

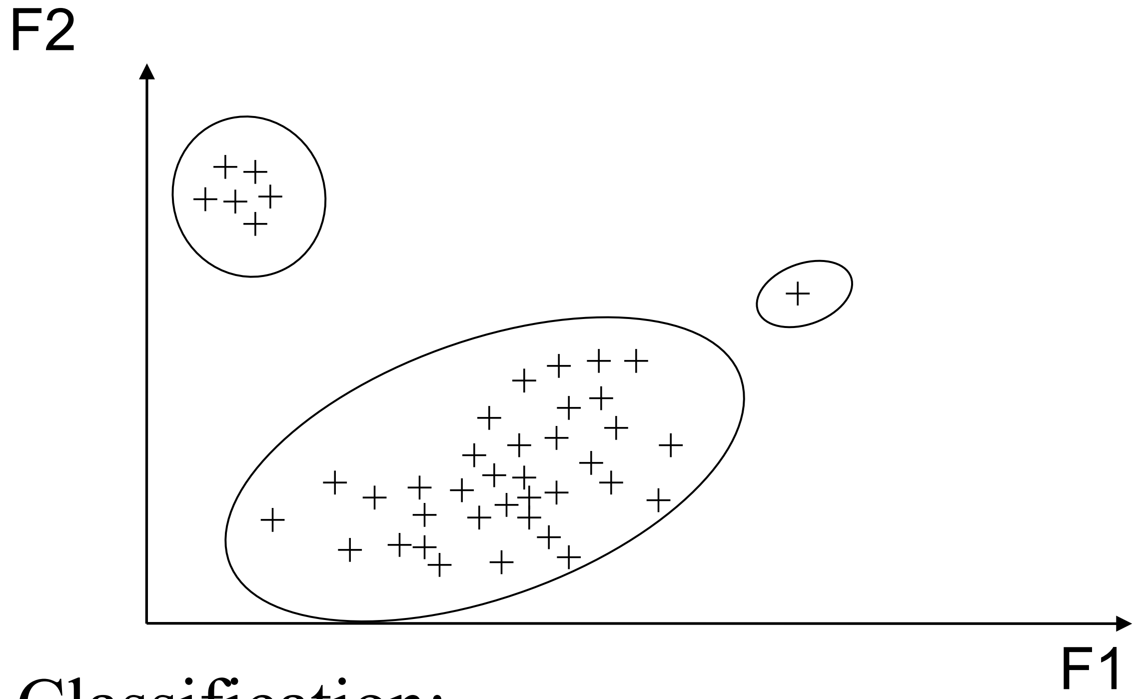


Unsupervised Classification



- Classification:
  - Classes Not Known: Find Structure

Unsupervised Classification



- Classification:
  - Classes Not Known: Find Structure
  - Clustering
  - How? How many?

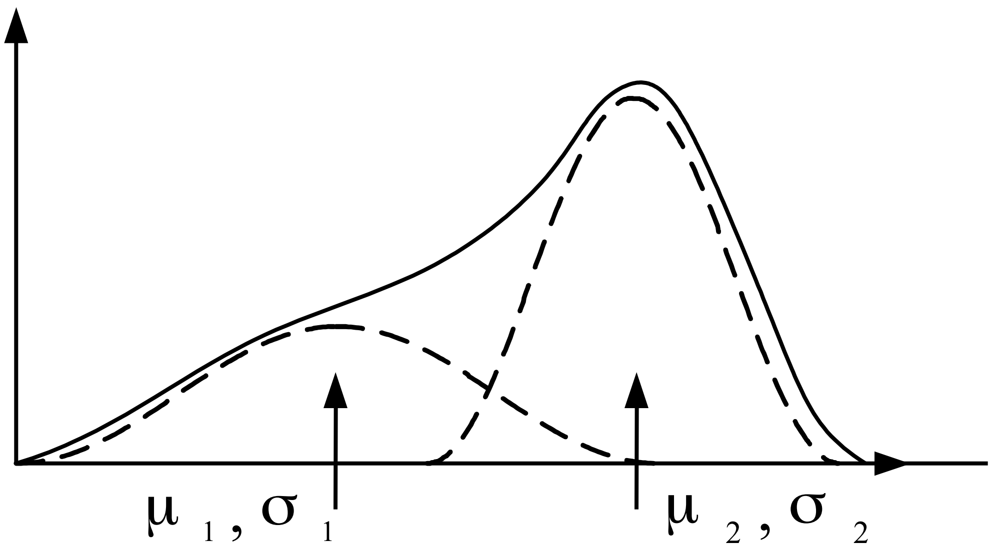
Unsupervised Learning

- Data collection and labeling costly and time consuming
- Characteristics of patterns can change over time
- May not have insight into nature or structure of data

→ **Classes NOT known**

Mixture Densities

- Samples come from c classes
- A priori probability $P(\omega_j)$
- Assume: The forms for the class-conditional PDF's $P(X / \omega_j, \Theta_j)$ are known (usually normal)
- Unknown parameter vector $\Theta_1 \dots \Theta_c$



Mixture Densities

Problem: Hairy Math

Simplification/ Approximation:

Look only for means -> Isodata

1. Choose initial $\mu_1 \ldots \mu_c$
2. Classify n samples to closest mean
3. Recompute means from samples in class
4. Means changed? Goto step 2, else stop

Mixture Densities

Isodata, problems:

- Choosing initial means μ
- Knowing number of classes
- Assuming distribution
- What is "closest"?

Clustering

- Similarity
- Criterion function
- Samples in same class should extremize criterion function, that measures cluster quality
- Example: sum of error criterion

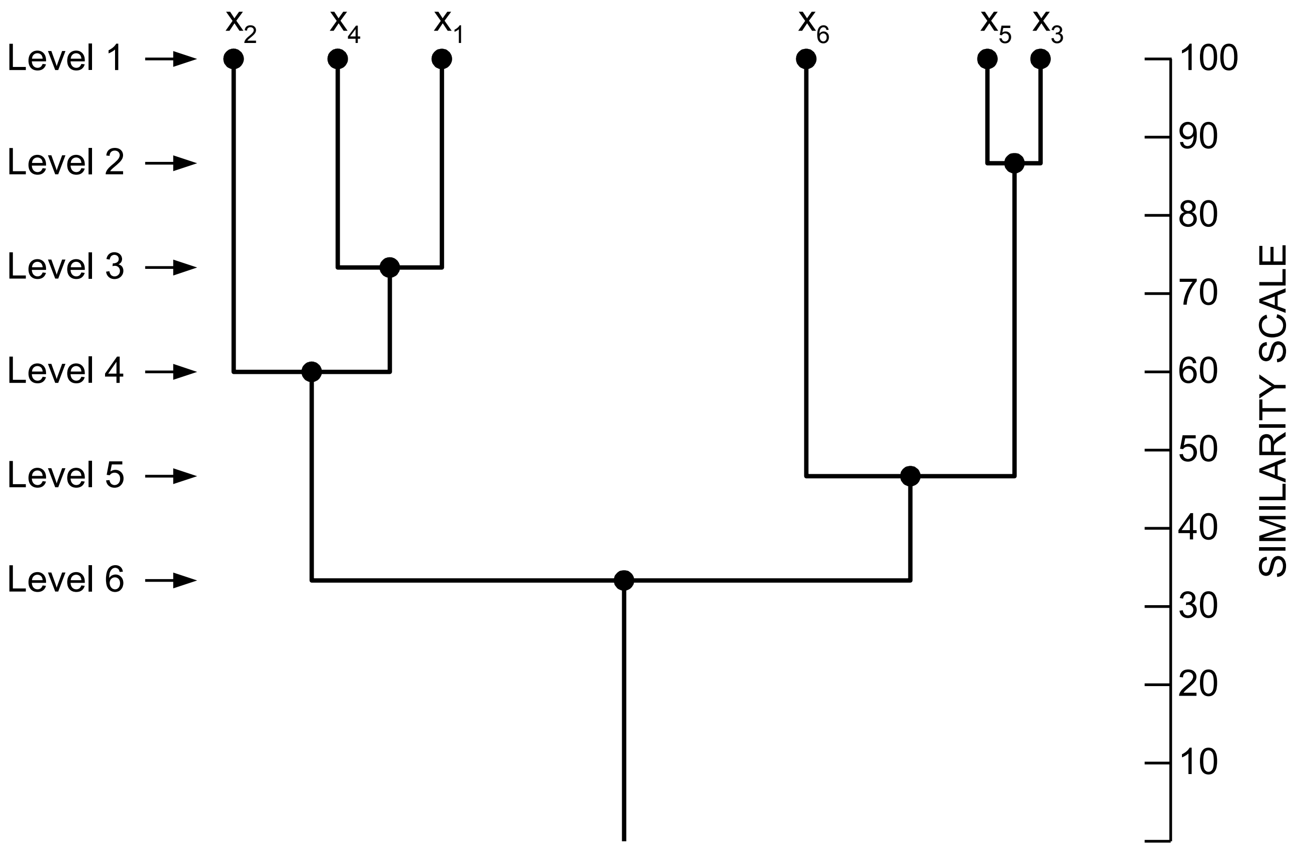
$$J = \sum_{i=1}^{c} \sum_{max} ||x - m_i||^2$$

Hierarchical Clustering

- Need not determine c
- Need not guess initial means

1. Initialize c := n
2. Find nearest pair of distinct clusters X_i and X_j
3. Merge them and decrement c
4. If $c \leq C_{stop}$ stop, otherwise goto step 2

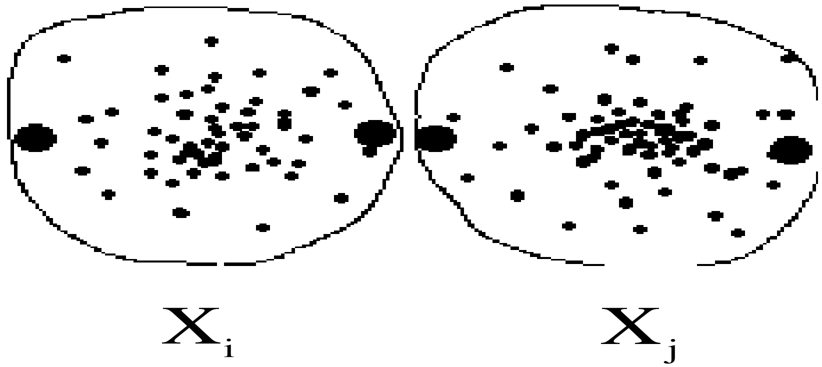
Dendrogram for hierarchical clustering



Similarity

What constitutes "nearest cluster"?

- D_{min}
- D_{ave}
- D_{max}

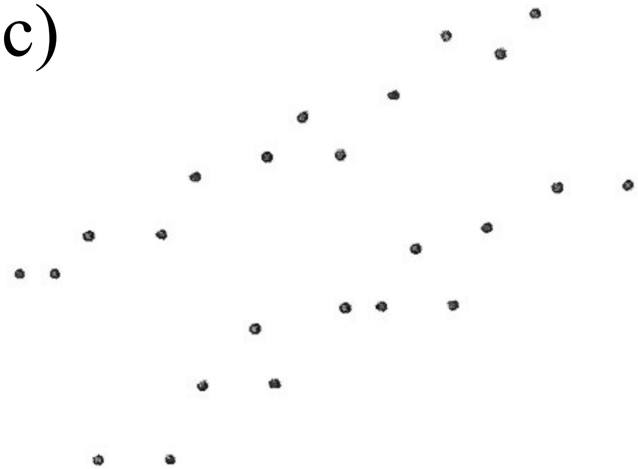


Three illustrative examples

a)

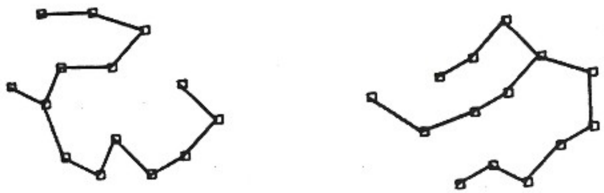
b)

c)

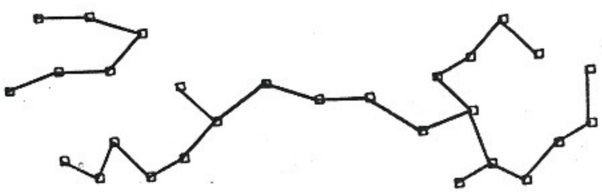


Results of the nearest-neighbor algorithm

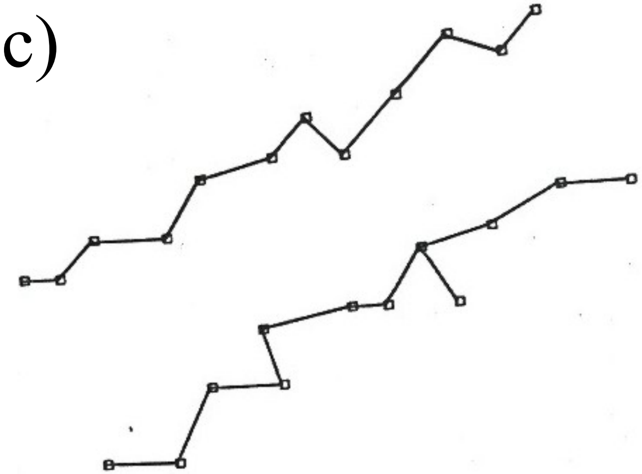
a)



b)



c)



Results of the furthest-neighbor algorithm