Probabilistic Neural Networks

Compiled by Milo

mainly based on:

Donald F. Specht. 1990. Probabilistic neural networks. Neural Netw. 3, 1 (January 1990), 109-118.

Main source: Donald F. Specht. 1990. Probabilistic neural networks. Neural Netw. 3, 1 (January 1990), 109-118.

Neural Networks, Vol. 3, pp. 109–118, 1990
Printed in the USA. All rights reserved.

0893-6080/90 $3.00 + .00
Copyright © 1990 Pergamon Press plc

ORIGINAL CONTRIBUTION

Probabilistic Neural Networks

DONALD F. SPECHT

Lockheed Missiles & Space Company, Inc.

(*Received* 5 *August* 1988; *revised and accepted* 14 *June* 1989)

Abstract—*By replacing the sigmoid activation function often used in neural networks with an exponential function, a probabilistic neural network (PNN) that can compute nonlinear decision boundaries which approach the Bayes optimal is formed. Alternate activation functions having similar properties are also discussed. A four-layer neural network of the type proposed can map any input pattern to any number of classifications. The decision boundaries can be modified in real-time using new data as they become available, and can be implemented*

Outline

1. Motivation
2. The Bayes Strategy for Pattern Classification
3. The Probabilistic Neural Network
4. Probability Density Functions
5. Effect of smoothing parameter σ
6. Example
7. Advantages and disadvantages
8. Further information

Motivation

- All neural networks try to determine pattern statistics from a set of training samples and then classify new patterns on the basis of these statistics
- Many methods, such as Backpropagation uses heuristic approaches to discover the underlying class statistics.
- The heuristic approaches usually involve many small modifications to the system parameters that gradually improve system performance:
    1) long computation times for training
    2) the incremental adaptation approach can be susceptible to false minima

Motivation

- To improve upon this approach, a classification method based on established statistical principles was sought:

  “Probabilistic Neural Network”

- The structure is similar to back-propagation, but differs primarily in the activation function (replaced by a statistically derived one)
- Under certain (easily met) conditions, its decision boundary asymptotically approaches the Bayes optimal decision boundary/surface.

The Bayes Strategy for Pattern Classification

- Bayes strategies: strategies on classifying patterns trying to minimizes the expected risk/error
- It can be applied to problems containing any number of categories/classes
- For a simple case, two categories/class-labels $y \in \{A, B\}$
  - A *p*-dimensional vector $\mathbf{x} = [x_1, x_2, \ldots, x_j, \ldots, x_p]$ is to be decided either having class label $y = A$ or $y = B$.
  - The Bayes decision rule:

$$d(\mathbf{x}) = A, \text{ if } P(y = A|\mathbf{x}) > P(y = B|\mathbf{x})$$

$$d(\mathbf{x}) = B, \text{ if } P(y = A|\mathbf{x}) < P(y = B|\mathbf{x})$$

The Bayes Strategy for Pattern Classification

- We know that $P(y|\mathbf{x}) \propto P(\mathbf{x}|y)P(y)$

$$d(\mathbf{x}) = A, \text{ if } P(y = A|\mathbf{x}) > P(y = B|\mathbf{x})$$
$$\Leftrightarrow \quad P(\mathbf{x}|y = A)P(y = A) > P(\mathbf{x}|y = B)P(y = B)$$
$$\Leftrightarrow \quad P(\mathbf{x}|y = A)P(y = A)\ l_A > P(\mathbf{x}|y = B)P(y = B)\ l_B$$

$$d(\mathbf{x}) = B, \text{ if } P(y = A|\mathbf{x}) < P(y = B|\mathbf{x})$$
$$\Leftrightarrow \quad P(\mathbf{x}|y = A)P(y = A) < P(\mathbf{x}|y = B)P(y = B)$$
$$\Leftrightarrow \quad P(\mathbf{x}|y = A)P(y = A)\ l_A < P(\mathbf{x}|y = B)P(y = B)\ l_B$$

l_A and l_B are loss function

The Bayes Strategy for Pattern Classification

- Thus the decision boundary/surface between the region in which the Bayes decision $d(\mathbf{x}) = A$ and the region in which $d(\mathbf{x}) = A$, is given by

$$P(\mathbf{x}|y = A) = K\ P(\mathbf{x}|y = B)$$

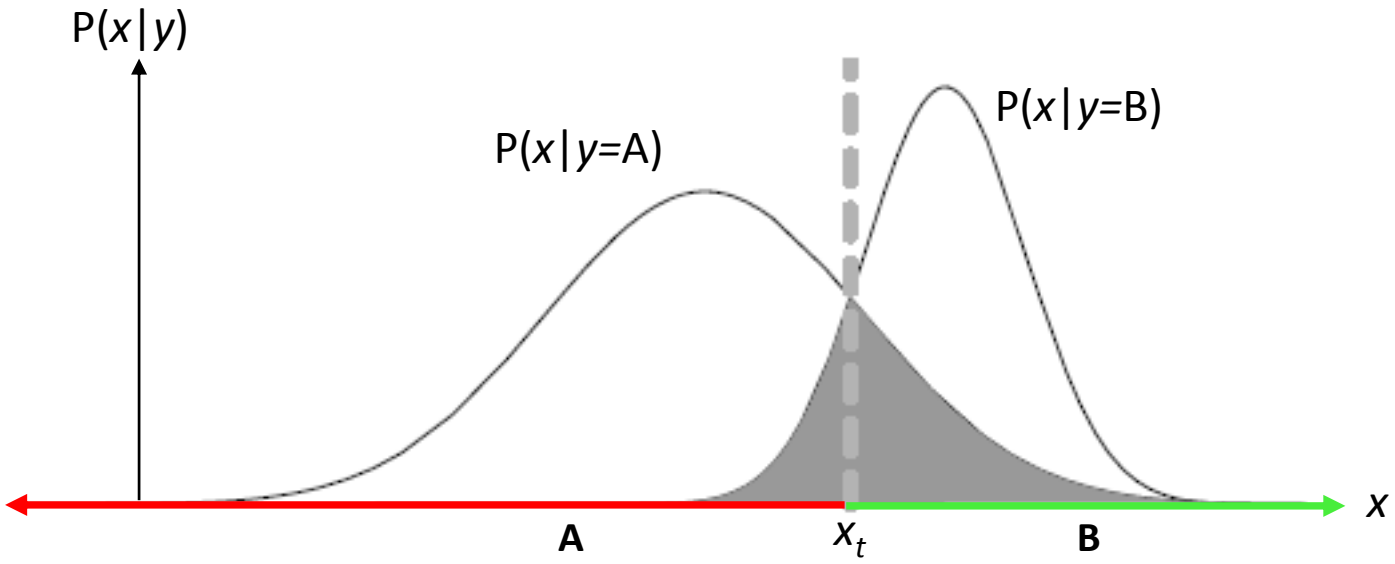
where $K = \frac{P(y=B)\ l_B}{P(y=A)\ l_A}$

Bayesian classifier: Example

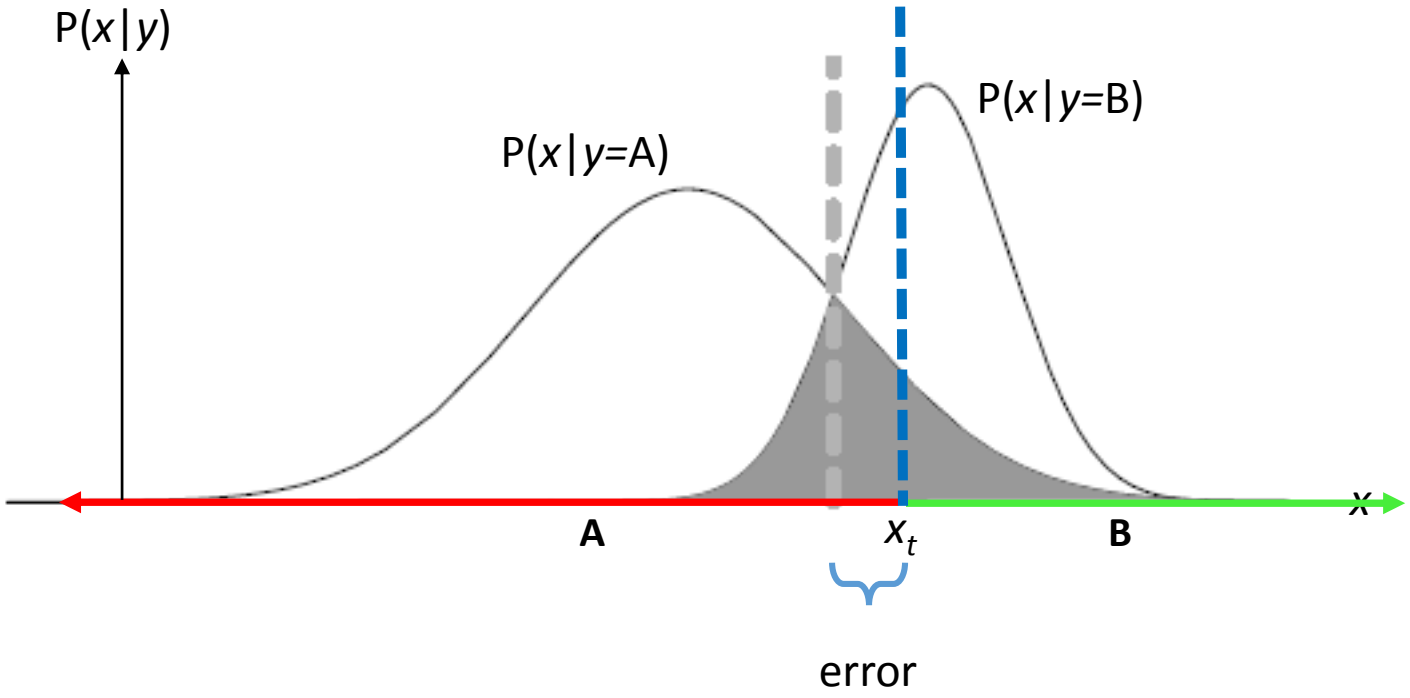
- For simplicity, say **x** is one-dimensional (so write it as x), and $p(x \mid y)$ is Gaussian, for both values of y. Say we know $P(y)$, and we are maximizing standard accuracy. We can explicitly compute the optimal decision boundary: It is given by the point(s!), i.e. value(s) of x, for which

$$P(\, y{=}\mathrm{A} \mid x) = P(\, y{=}\mathrm{B} \mid x)$$

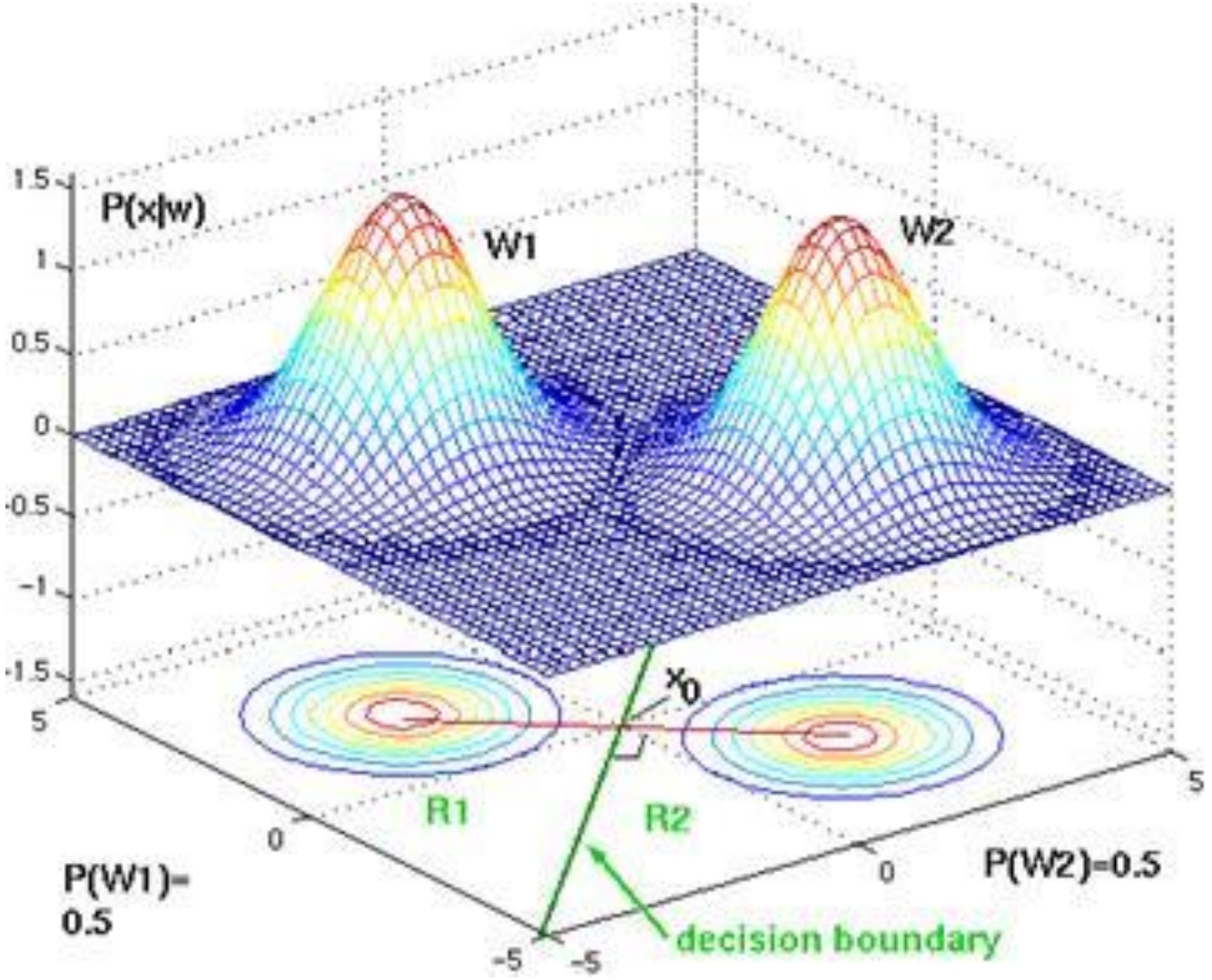
$$p(x \mid y = \mathrm{A})P(y = \mathrm{A}) = p(x \mid y = \mathrm{B})P(y = \mathrm{B})$$



Source: Lecture notes of Introduction to Machine Learning, Patrik Hoyer, University of Helsinki, 2012.

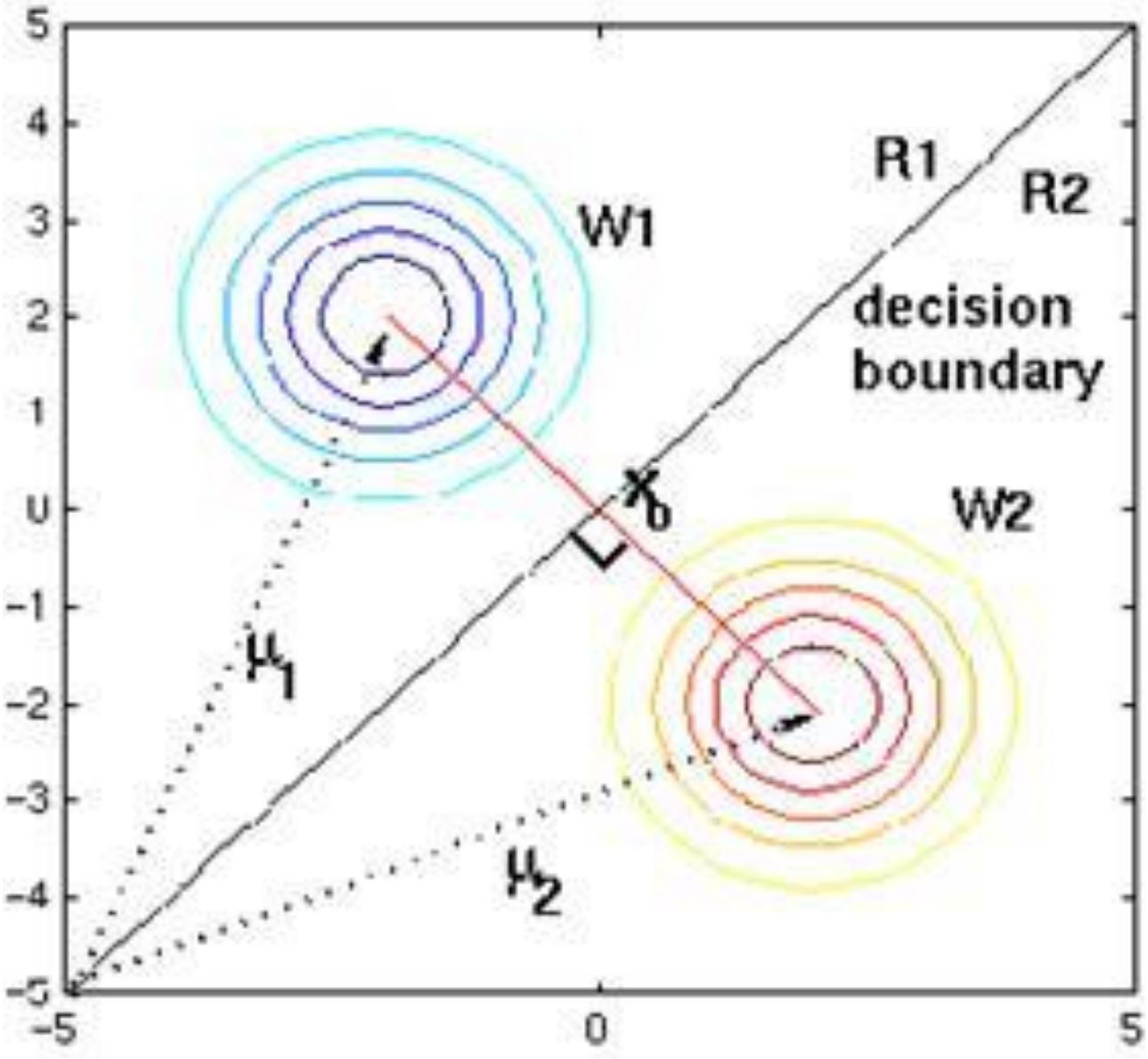


Source: Lecture notes of Introduction to Machine Learning, Patrik Hoyer, University of Helsinki, 2012.



Two bivariate normal distributions, whose priors are exactly the same. Therefore, the decision boundary is exactly at the midpoint between the two means. The decision boundary is a line orthogonal to the line joining the two means.

Source: https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm



Source: https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm

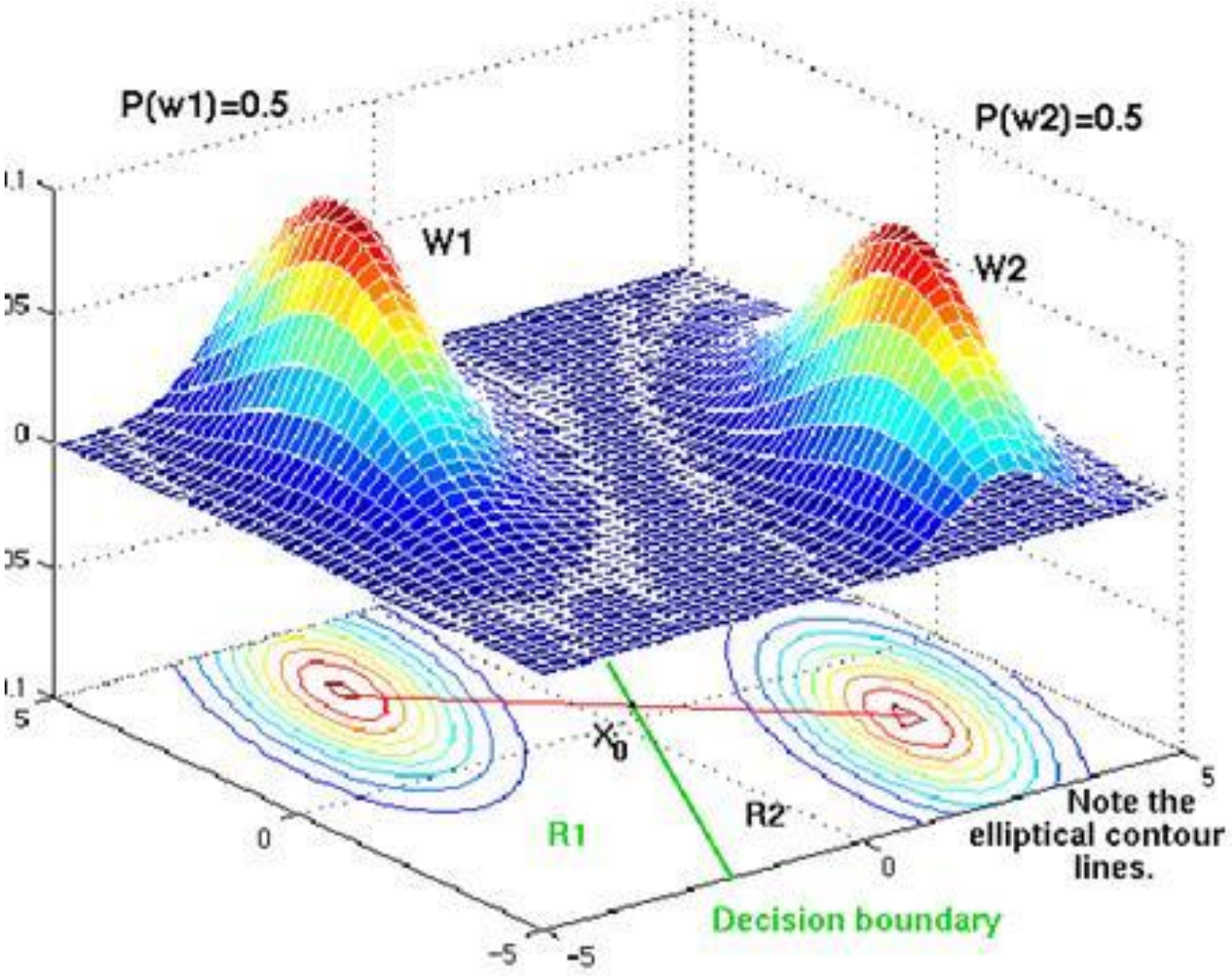
Recall: Bayes strategy

- Bayes strategies: strategies on classifying patterns trying to minimizes the expected risk/error

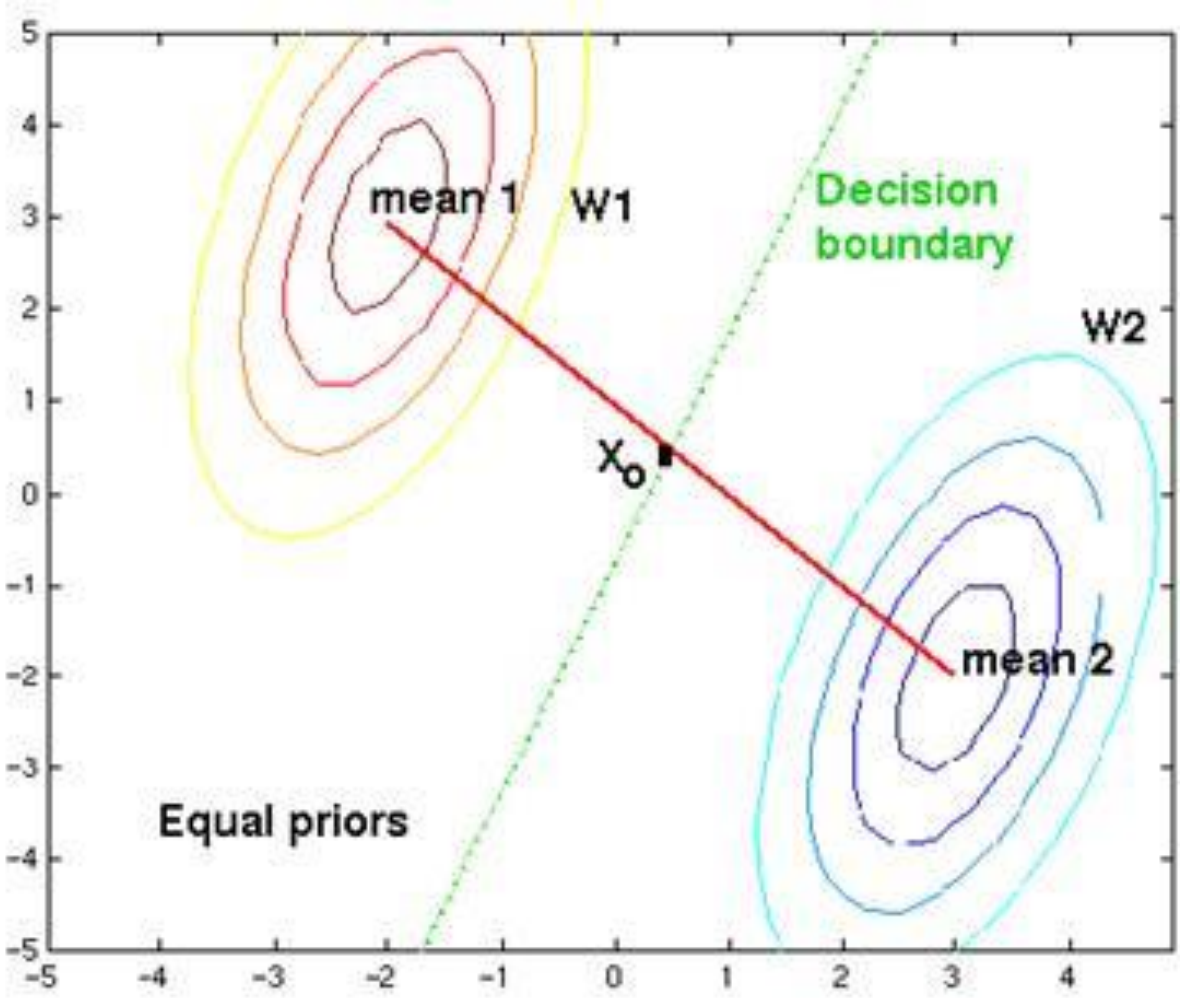
- Minimizing expected error = maximizing correct classification → finding optimal decision boundary

The Bayes Strategy for Pattern Classification

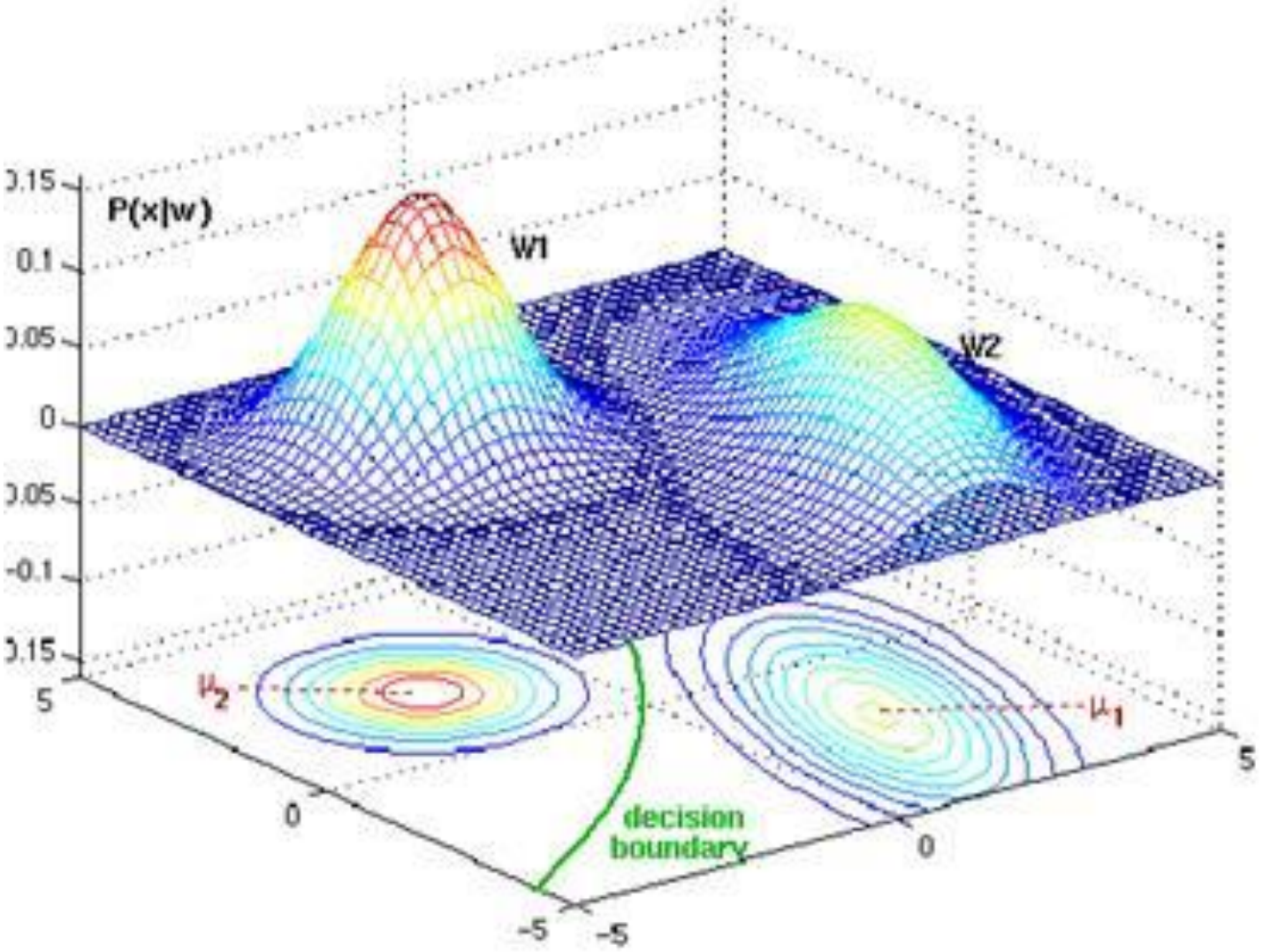
- Decision boundary/surface can be arbitrarily complex, since there is no restriction on the probability densities
- However, the probability density functions (PDF) must satisfy:
    - they are everywhere non-negative,
    - they are integrable,
    - their integrals over all space equal unity



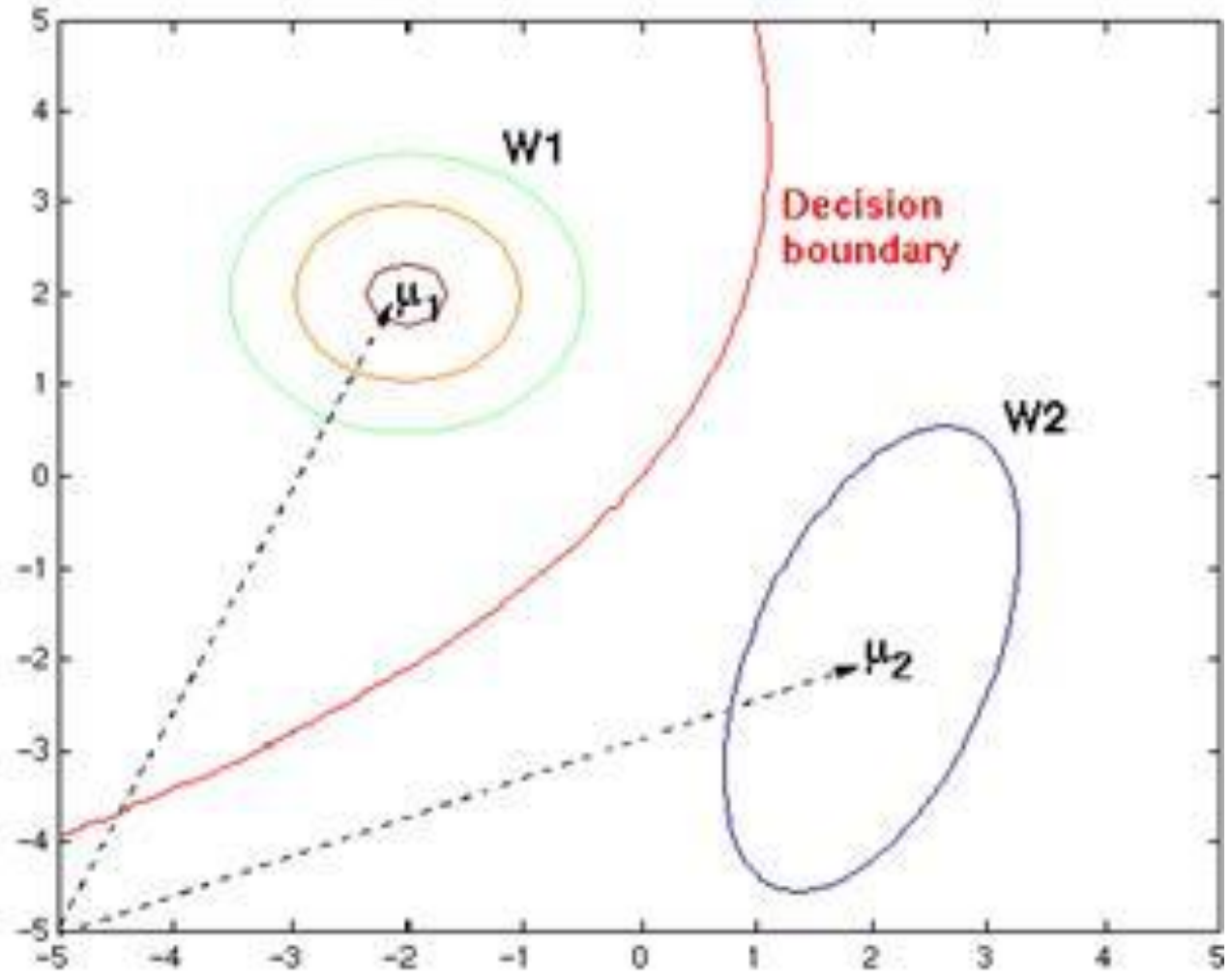
Source: https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm



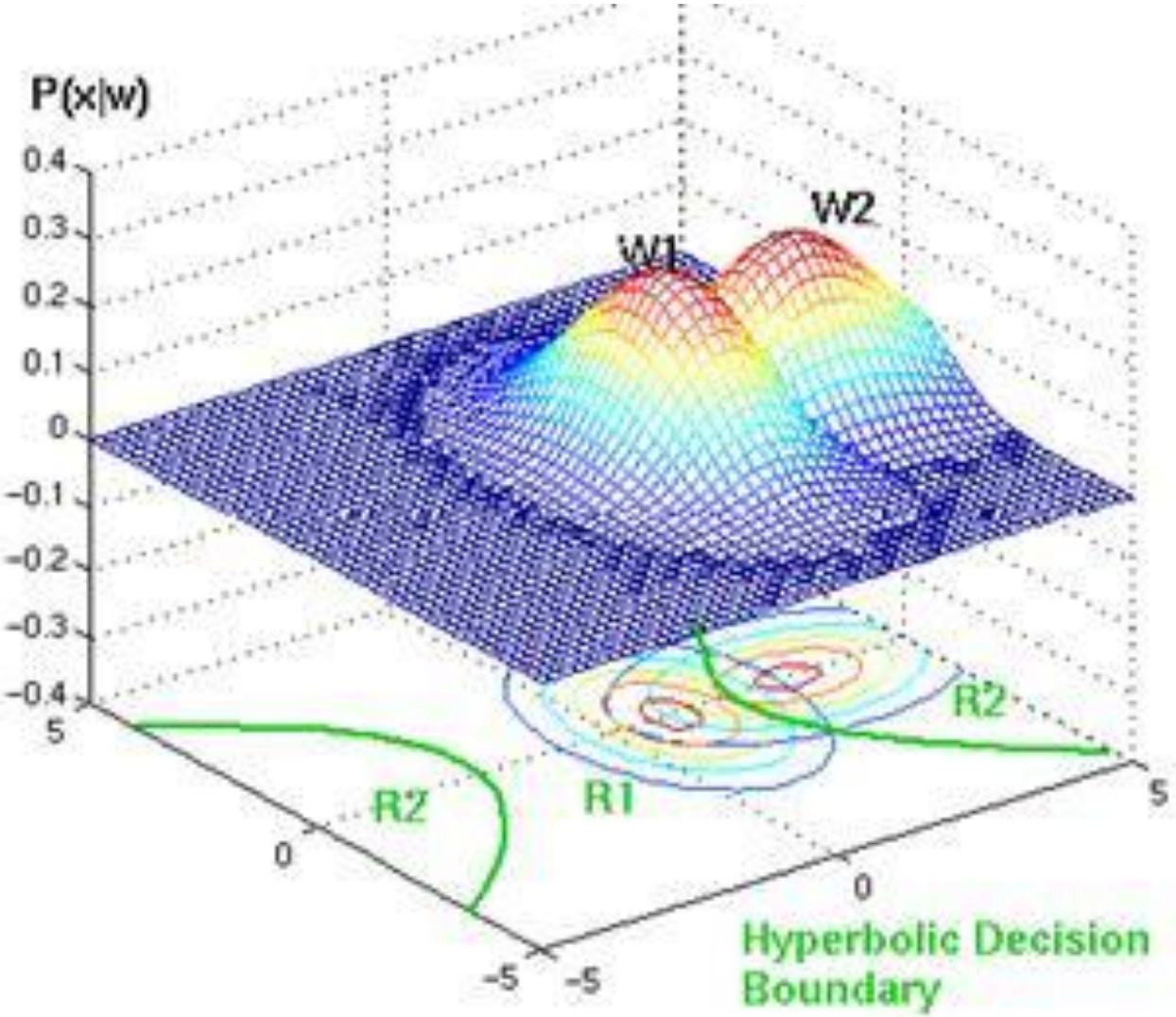
Source: https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm



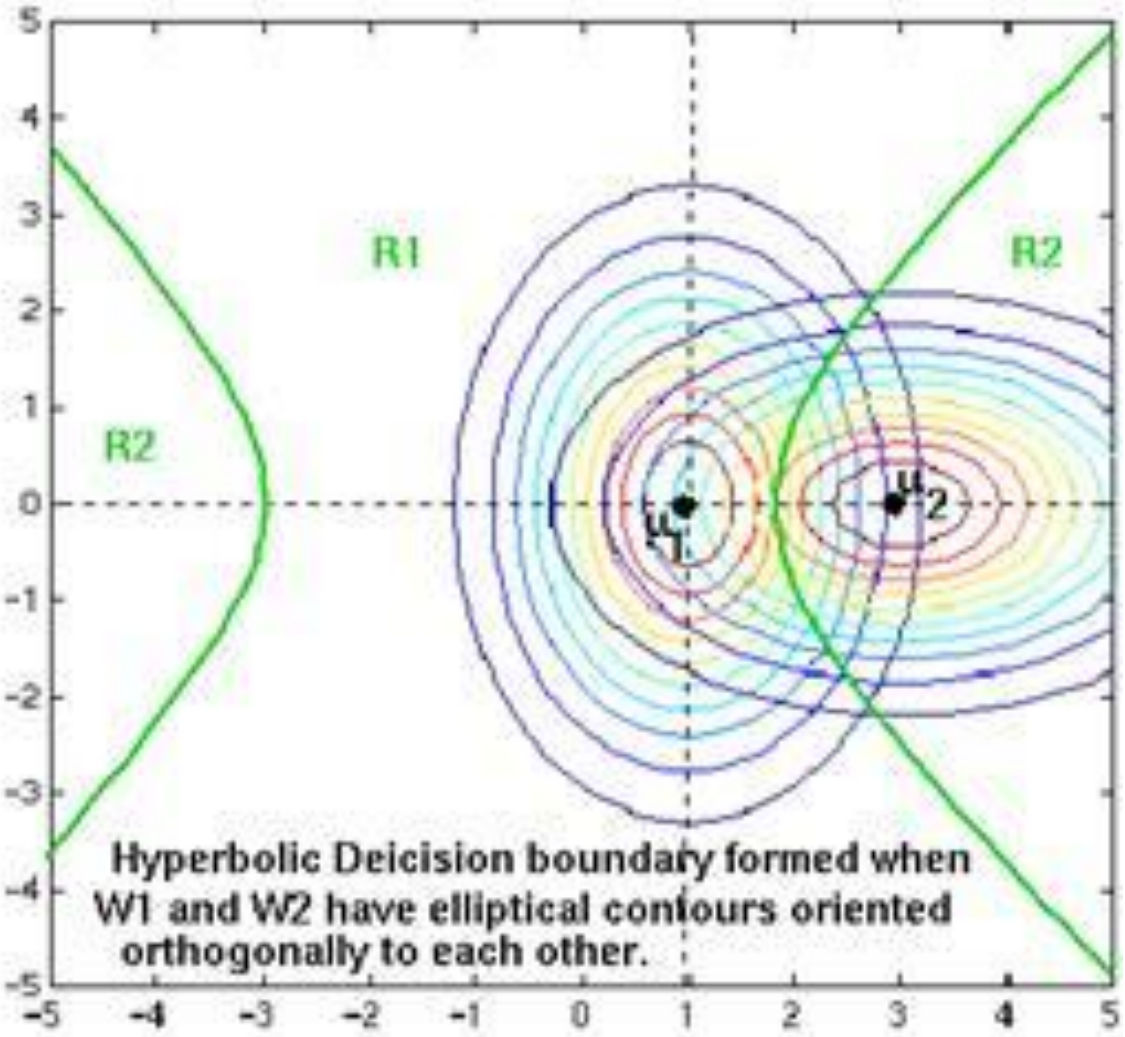
Source: https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm



Source: https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm



Source: https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm



Source: https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm

The Bayes Strategy for Pattern Classification

- The key is the ability to estimate PDFs based on training patterns.
- A priori probabilities are usually known or can be estimated accurately, and the loss functions require subjective evaluation.
- However, if the probability densities of the patterns in the categories to be separated are unknown, and all we have is a set of training patterns (training samples), then these samples provide the only clue to the unknown underlying probability densities.

The Bayes Strategy for Pattern Classification

- Parzen (1962) showed that a class of PDF estimators asymptotically approaches the underlying parent density provided only that it is continuous. Therefor, the PDF must be continuous.

- The accuracy of the decision boundaries depends on the accuracy of estimating the underlying PDFs

Recall: Bayes strategy

- Bayes strategies: strategies on classifying patterns trying to minimizes the expected risk/error

- Minimizing expected error = maximizing correct classification → finding optimal decision boundary → finding/estimating the underlying PDFs

- Assume (often) that data are drawn from Gaussian distribution, therefor the underlying PDF could be estimated by finding estimates of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ from the data, and then substituting these estimates into the formula of Gaussian density.

Estimate Probability Density Function using Kernel Density Estimation (KDE)

- For univariate i.i.d sample drawn from some distribution with an unknown density f

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

- For multivariate

$$\hat{f}(\mathbf{x}) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right)$$

K = kernel (a non-negative function that integrates to one). The normal kernel is often used.
x_i = i-th sample
x = an observation/input
h = smoothing parameter
n = sample size
$\mathbf{x}_i$ = i-th sample vector
$\mathbf{x}$ = an observation/input as a vector

Estimate Probability Density Function using Gaussian Kernel

- For univariate

$$\hat{f}(x) = \frac{1}{n\sqrt{2\pi\sigma^2}} \sum_{i=1}^{n} e^{-\frac{(x-x_i)^2}{2\sigma^2}}$$

x = unknown (input)
x_i = i-th sample
σ = smoothing parameter
n = sample size

- For multivariate

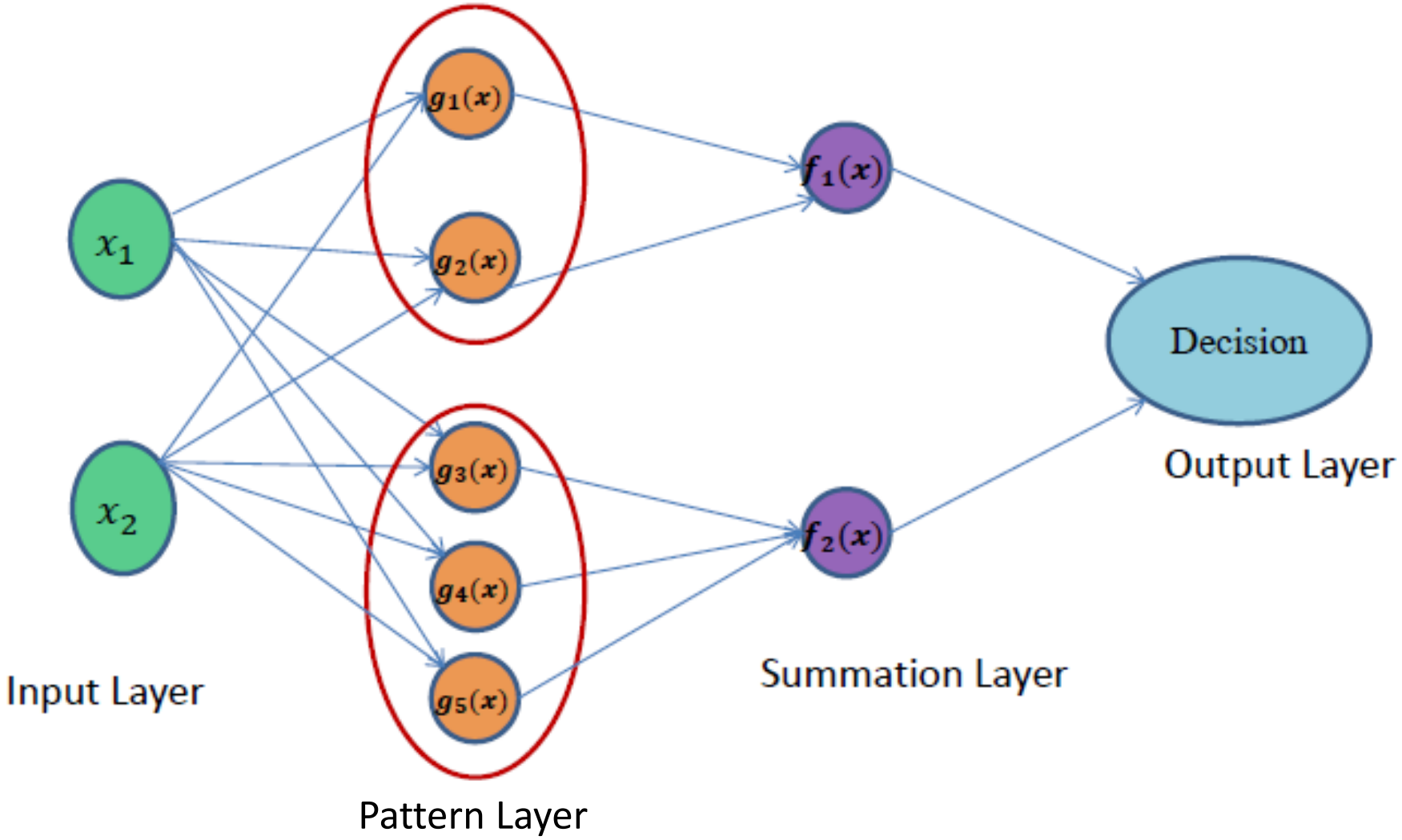
$$\hat{f}(\mathbf{x}) = \frac{1}{n\sqrt{(2\pi\sigma^2)^p}} \sum_{i=1}^{n} e^{-\frac{\left||\mathbf{x}-\mathbf{x}_i|\right|^2}{2\sigma^2}}$$

$\mathbf{x}$ = unknown (input)
$\mathbf{x}_i$ = i-th sample
σ = smoothing parameter
n = sample size
p = dimension

Probabilistic Neural Network

- Introduced by Donald F. Specht in the early 1990s.
- PNN is a feed forward Neural Network
- 4 layers architecture consists of
  - Input Layer
  - Pattern Layer
  - Summation Layer
  - Output Layer

Probabilistic Neural Network



PNN - Training

- In a sense, PNN is a Nearest Neighbor Algorithm with different kind of distance measuring
- Training involves only finding best parameter σ for the Gaussian distribution
- The Architecture can be summarized as
  - # of neuron in Input Layer = dimension of input data
  - # of neuron in Pattern Layer = number of training vectors
  - # of neuron in Summation Layer = number of classes

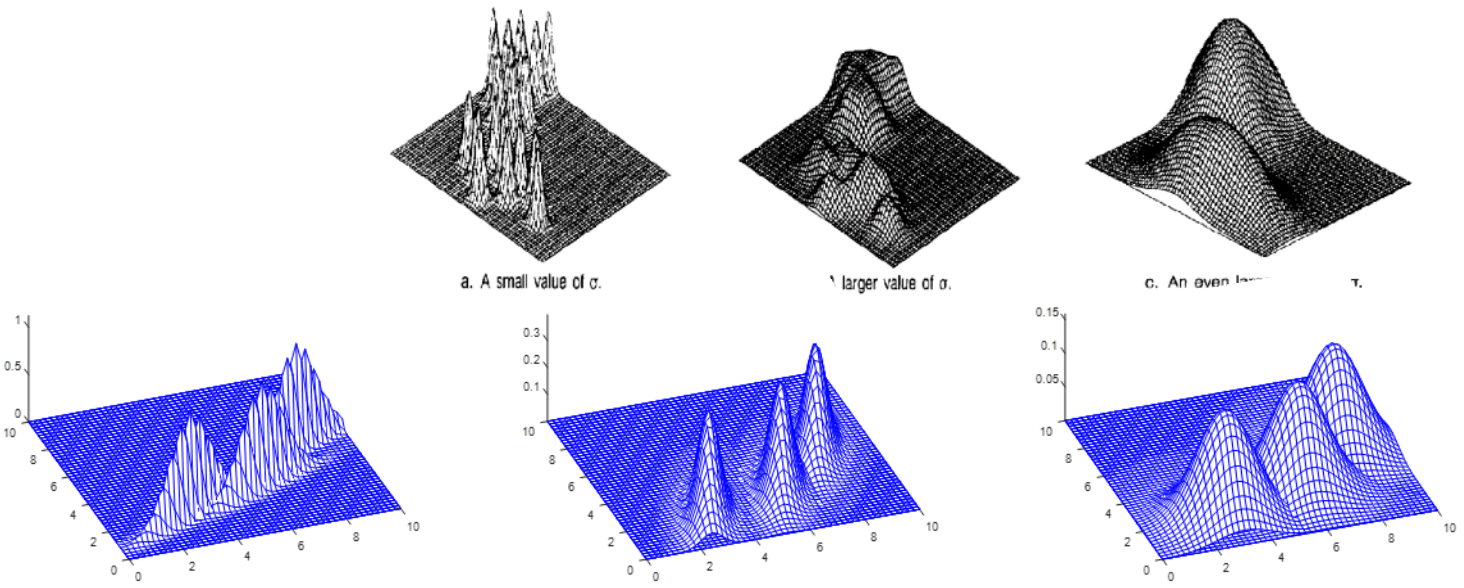
PNN – Testing

- When testing, PNN will calculate PDF $f_{\boldsymbol{C}}(x)$ for each class $\boldsymbol{C} \in \{c_1, c_2, \ldots, c_k\}$ by measuring new data x with all available data train of each class
- In output layer, PNN will output class which has the highest probability

Effect of smoothing parameter $\boldsymbol{\sigma}$

Smoothing Effect

- Nature of the PDF varies as we change σ.
  - Small σ creates distinct modes
  - Larger σ allows interpolation between points
  - Very Large σ approximate PDF to Gaussian



Determining smoothing parameter σ

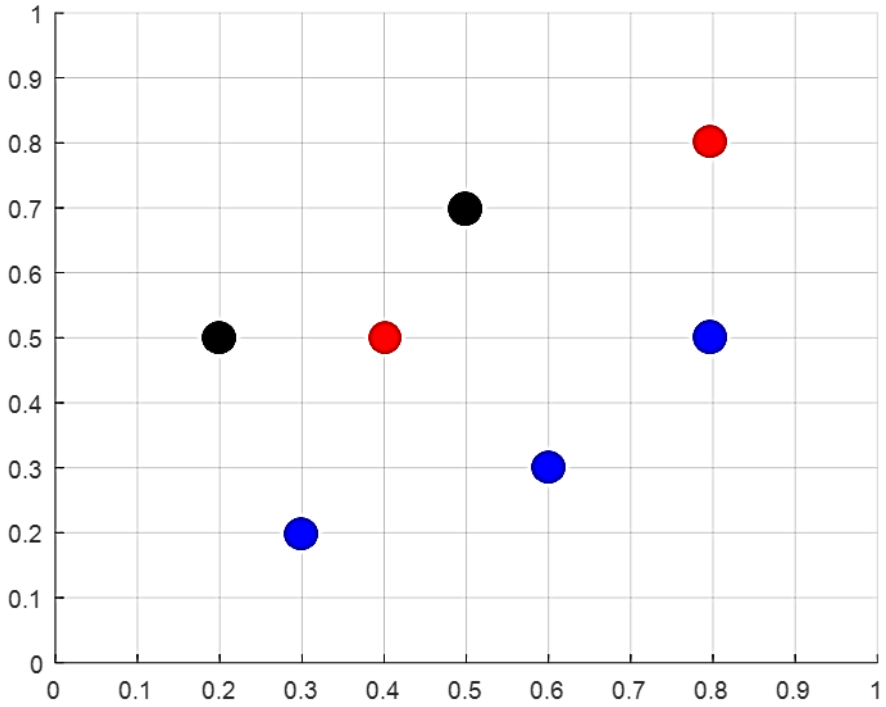
- Educated guess based on knowledge of the data
- Using a heuristic technique
- Brute force
- Using Jackknifing
  - Systematic testing of values for sigma over some range
  - Bounding the optimal value to some interval, then
  - Shrinking the interval

Two ways for smoothing parameter σ

1) All classes use the same σ value
2) Each classes has its own σ_k, use parameter g to determine σ_k

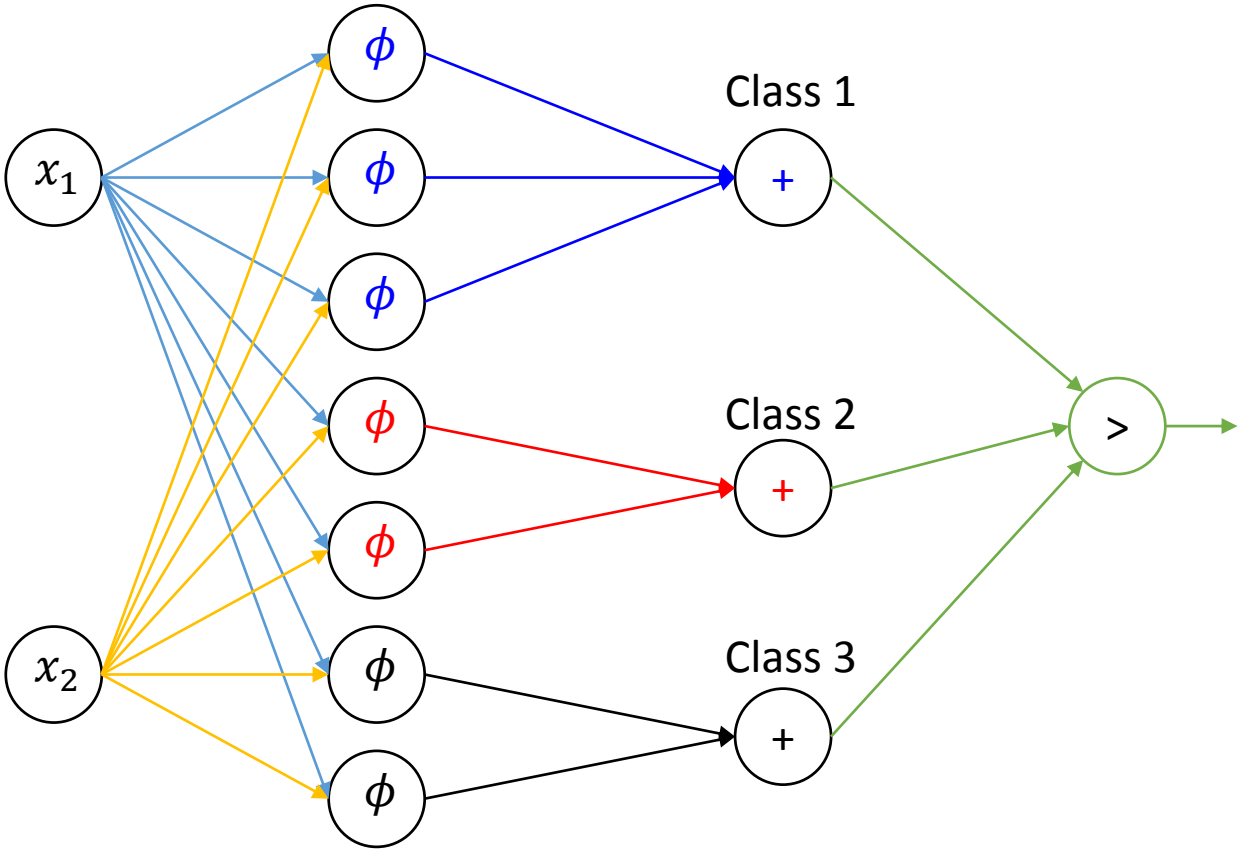
Example

Training set

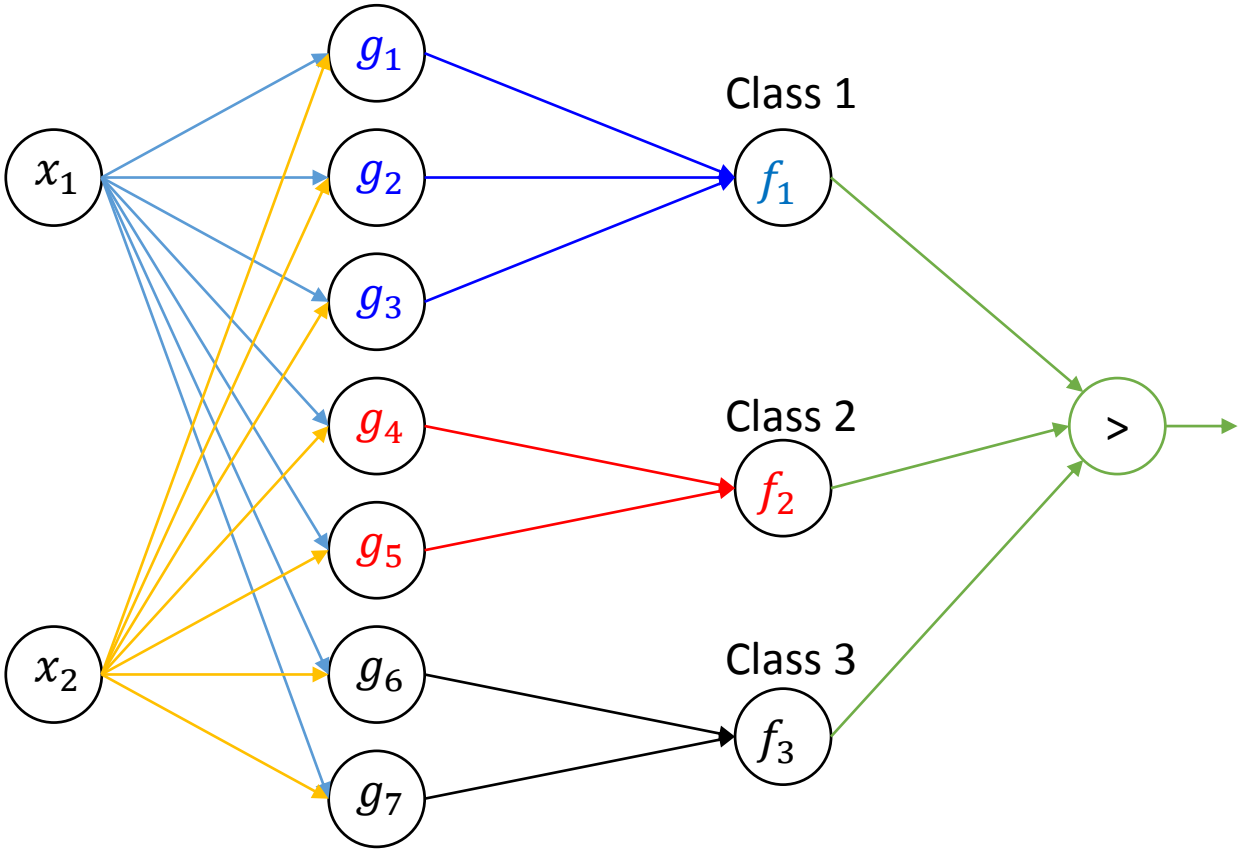


x_1	x_2	y
.3	.2	1
.6	.3	1
.8	.5	1
.4	.5	2
.8	.8	2
.2	.5	3
.5	.7	3

The Network



The Network



1st way

Probability Density Function

- Each data unit corresponds to a pattern unit which is a Gaussian density function with $p = 2$

$$g_i(\mathbf{x}) = \frac{1}{\sqrt{(2\pi\sigma^2)^2}} e^{-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{2\sigma^2}}$$

- Therefor, the PDF for this training set

$$f_c(\mathbf{x}) = \frac{1}{n_c}\sum_{i=1}^{n_c} g_i(\mathbf{x}) = \frac{1}{2\pi\sigma^2 n_c}\sum_{i=1}^{n_c} e^{-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{2\sigma^2}}$$

- Let's say the smoothing σ is equal for all classes, then the equation will be just

x_1	x_2	y
.3	.2	1
.6	.3	1
.8	.5	1
.4	.5	2
.8	.8	2
.2	.5	3
.5	.7	3

$$f_c(\mathbf{x}) = \frac{1}{n_c}\sum_{i=1}^{n_c} e^{-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{2\sigma^2}}$$

and

$$g_i(\mathbf{x}) = e^{-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{2\sigma^2}}$$

Probability Density Function

Assume σ =0.1 for all classes

x_1	x_2	y	$g_i(\mathbf{x}) = \exp\left(-\frac{\|\|\mathbf{x}-\mathbf{x}_i\|\|^2}{2\sigma^2}\right)$
.3	.2	1	$g_1(x) = \exp\{-\frac{(x_1-.3)^2+(x_2-.2)^2}{2(.1)^2}\}$
.6	.3	1	$g_2(x) = \exp\{-\frac{(x_1-.6)^2+(x_2-.3)^2}{2(.1)^2}\}$
.8	.5	1	$g_3(x) = \exp\{-\frac{(x_1-.8)^2+(x_2-.5)^2}{2(.1)^2}\}$
.4	.5	2	$g_4(x) = \exp\{-\frac{(x_1-.4)^2+(x_2-.5)^2}{2(.1)^2}\}$
.8	.8	2	$g_5(x) = \exp\{-\frac{(x_1-.8)^2+(x_2-.8)^2}{2(.1)^2}\}$
.2	.5	3	$g_6(x) = \exp\{-\frac{(x_1-.2)^2+(x_2-.5)^2}{2(.1)^2}\}$
.5	.7	3	$g_7(x) = \exp\{-\frac{(x_1-.5)^2+(x_2-.7)^2}{2(.1)^2}\}$

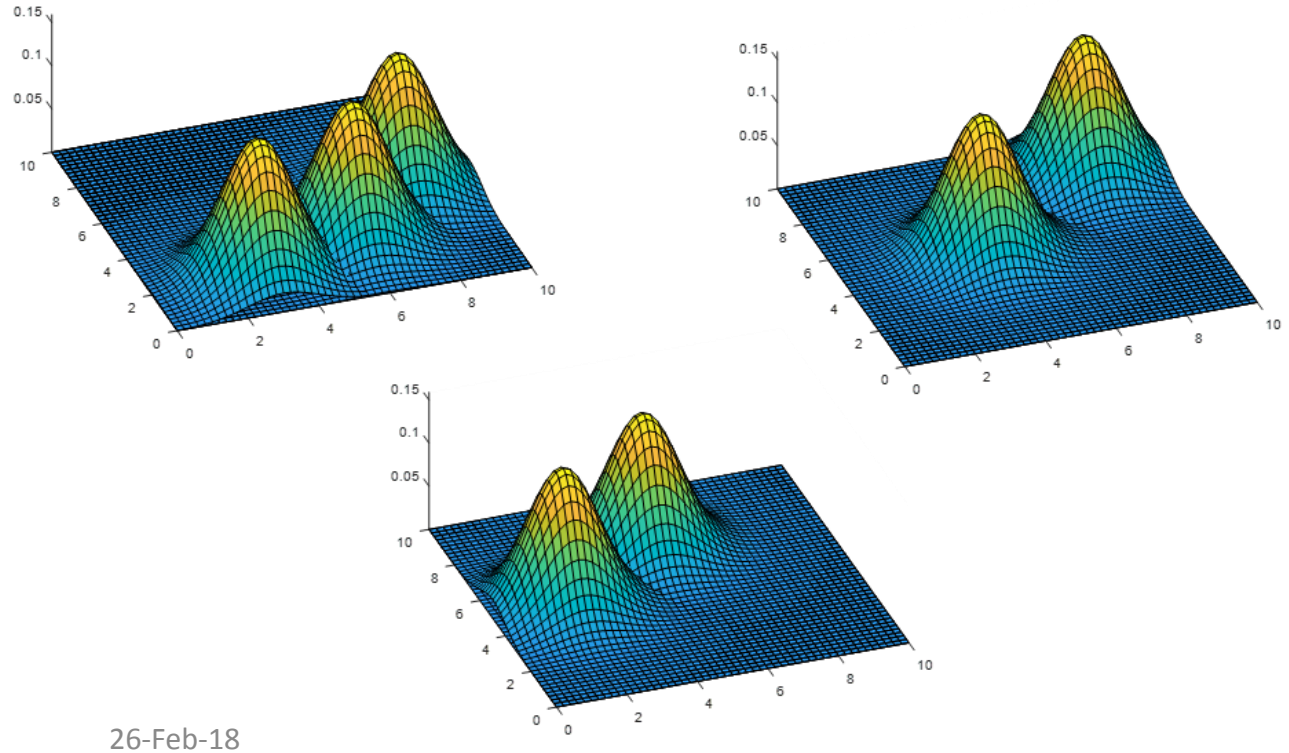
$f_1(\mathbf{x}) = [g_1(\mathbf{x}) + g_2(\mathbf{x}) + g_3(\mathbf{x})\ ] / 3$

$f_2(\mathbf{x}) = [g_4(\mathbf{x}) + g_5(\mathbf{x})\ ] / 2$

$f_3(\mathbf{x}) = [\ g_6(\mathbf{x}) + g_7(\mathbf{x})\ ] / 2$

Probability Density Function

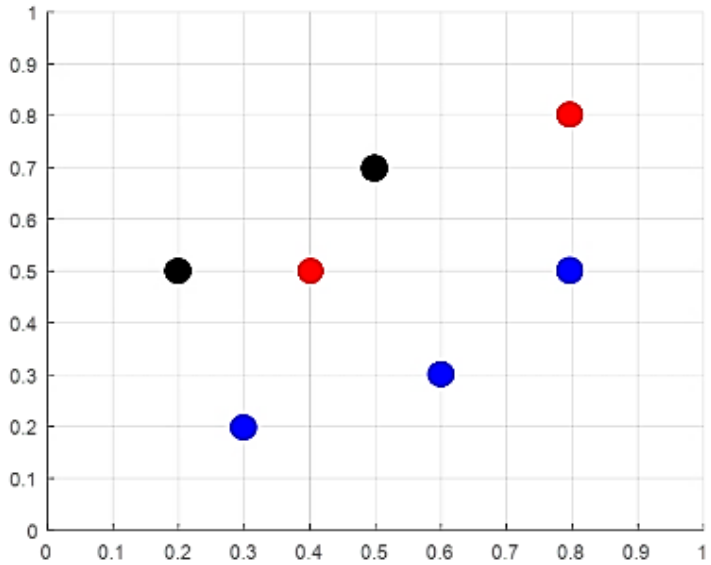
- Each data unit corresponds to a pattern unit which is a Gaussian Function



x_1	x_2	y
.3	.2	1
.6	.3	1
.8	.5	1
.4	.5	2
.8	.8	2
.2	.5	3
.5	.7	3

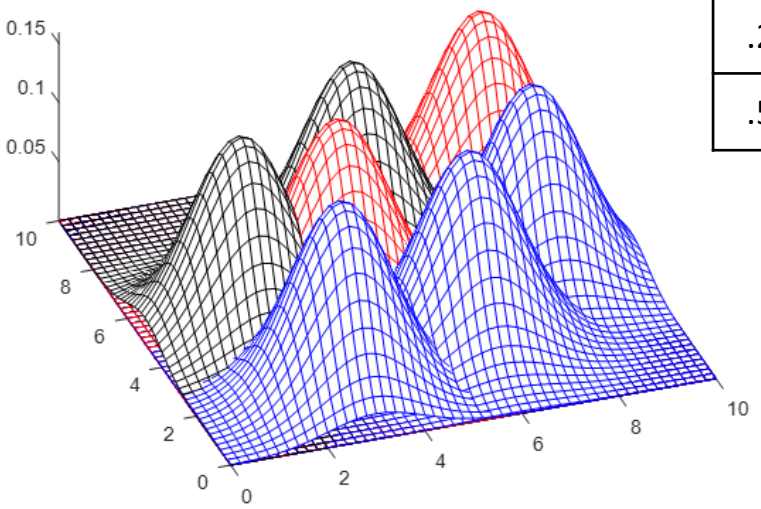
Source: Slide Pembelajaran Mesin, ADF, Universitas Telkom, 2018

Population PDF

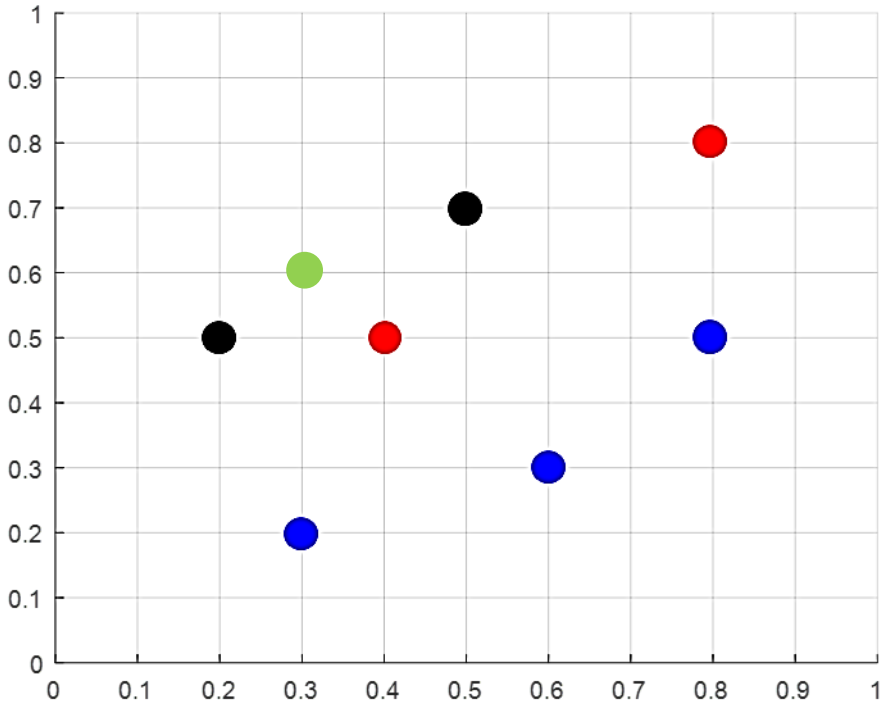


x_1	x_2	y
.3	.2	1
.6	.3	1
.8	.5	1
.4	.5	2
.8	.8	2
.2	.5	3
.5	.7	3

Find/observe σ value that gives best classification rate



Testing on new data



x_1	x_2	y
.3	.2	1
.6	.3	1
.8	.5	1
.4	.5	2
.8	.8	2
.2	.5	3
.5	.7	3
.2	.6	??

Testing on new data

x_1	x_2	y	$g_i(x) = \exp\left(-\frac{\|\|x-x_k\|\|^2}{2\sigma^2}\right)$	$g_i(x)$
.3	.2	1	$g_1(x) = \exp\{-\frac{(x_1-.3)^2+(x_2-.2)^2}{2(.1)^2}\}$	0
.6	.3	1	$g_2(x) = \exp\{-\frac{(x_1-.6)^2+(x_2-.3)^2}{2(.1)^2}\}$	0
.8	.5	1	$g_3(x) = \exp\{-\frac{(x_1-.8)^2+(x_2-.5)^2}{2(.1)^2}\}$	0
.4	.5	2	$g_4(x) = \exp\{-\frac{(x_1-.4)^2+(x_2-.5)^2}{2(.1)^2}\}$	0.082
.8	.8	2	$g_5(x) = \exp\{-\frac{(x_1-.8)^2+(x_2-.8)^2}{2(.1)^2}\}$	0
.2	.5	3	$g_6(x) = \exp\{-\frac{(x_1-.2)^2+(x_2-.5)^2}{2(.1)^2}\}$	0.606
.5	.7	3	$g_7(x) = \exp\{-\frac{(x_1-.5)^2+(x_2-.7)^2}{2(.1)^2}\}$	0.006

$$\mathbf{x} = (0.2, 0.6)$$

$$f_1(\mathbf{x}) = [g_1(\mathbf{x}) + g_2(\mathbf{x}) + g_3(\mathbf{x})\,] / 3$$
$$f_2(\mathbf{x}) = [g_4(\mathbf{x}) + g_5(\mathbf{x})\,] / 2$$
$$f_3(\mathbf{x}) = [\,g_6(\mathbf{x}) + g_7(\mathbf{x})\,] / 2$$

$$f_1(\mathbf{x}) = 0$$
$$f_2(\mathbf{x}) = 0.041$$
$$f_3(\mathbf{x}) = 0.306$$

2nd way

Determining smoothing parameter σ

// reset weights:
FORALL prototypes p_i^k **DO**
  $A_i^k = 0.0$
ENDFOR
// train one complete epoch
FORALL training pattern $(\vec{x}, k)$ **DO**:
  IF $\exists p_i^k : p_i^k(\vec{x}) \geq \theta^+$ **THEN**
    $A_i^k += 1.0$
  ELSE
    // "commit": introduce new prototype
    $m_k += 1$
    $\vec{\mu}_{m_k}^k = \vec{x}$
    $A_{m_k}^k = 1.0$
    $$\sigma_{m_k}^k = \min_{\substack{l \neq k \\ 1 \leq j \leq m_l}} \left\{ \sqrt{-\frac{||\vec{\mu}_j^l - \vec{\mu}_{m_k}^k||^2}{\ln \theta^-}} \right\}$$
  ENDIF
  // "shrink": adjust conflicting prototypes
  FORALL $l \neq k, 1 \leq j \leq m_l$ **DO**
    $$\sigma_j^l = \min \left\{ \sigma_j^l , \sqrt{-\frac{||\vec{x} - \vec{\mu}_j^l||^2}{\ln \theta^-}} \right\}$$
ENDFOR

{ Tahap pertama }
For setiap pola ρ_i
  $w_{i=} \rho_i$
  Bentuk unit pola dengan masukan vektor bobot w_i
  Hubungkan unit pola pada unit penjumlah untuk masing-masing kelas
End
Tentukan konstanta $|C_k|$ untuk setiap unit penjumlah

{ Tahap ke dua }
For setiap pola ρ_i
  k = kelas ρ_i
  Cari jarak, d_i, dengan pola terdekat pada kelas k
  d_{tot} [k] = d_{tot} [k] + d_i
End

For setiap kelas k
  σ_k = (g . d_{tot} [k]) / $|C_k|$
End

Probability Density Function

- The smoothing σ is different for each class σ_k

$$\sigma_k = \frac{(g \,.\, d_{tot}[k])}{|C_k|}$$

where $d_{tot}[k]$ is total minimum distance of class k, and $|C_k|$ is sample size of class k.

- The Gaussian kernel for this training set

$$g_i(\mathbf{x}) = e^{-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{2(\sigma_k)^2}}$$

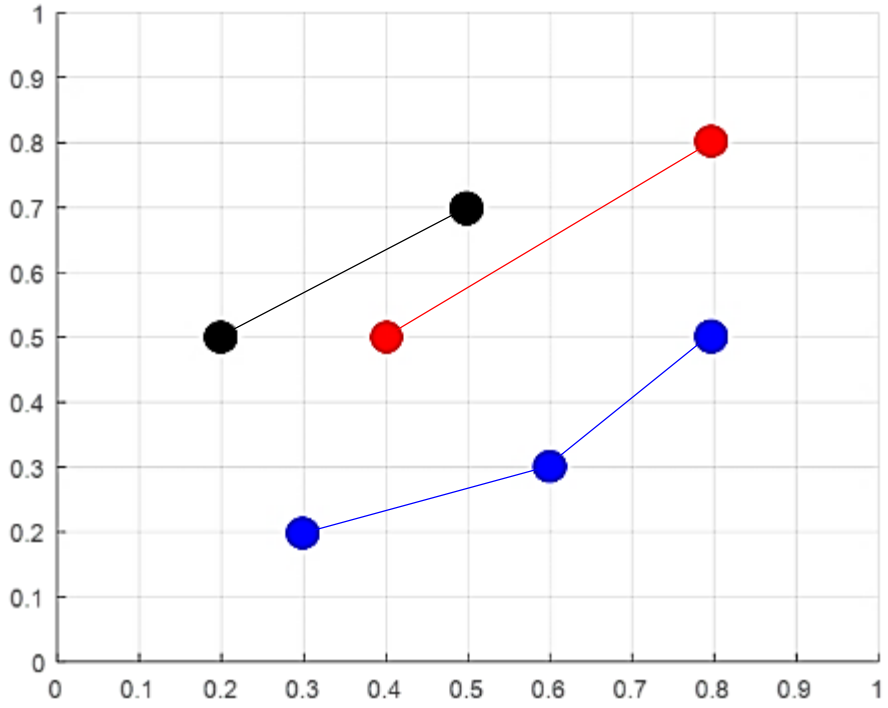
- The PDF

$$f_k(\mathbf{x}) = \sum_{i=1}^{n_k} g_i(\mathbf{x}) = \sum_{i=1}^{n_k} e^{-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{2(\sigma_k)^2}}$$

x_1	x_2	y
.3	.2	1
.6	.3	1
.8	.5	1
.4	.5	2
.8	.8	2
.2	.5	3
.5	.7	3

Calculate Total of Minimum Distance

x_1	x_2	y	dist	Closest to data
.3	.2	1	0.316	2
.6	.3	1	0.282	3
.8	.5	1	0.282	2
.4	.5	2	0.500	5
.8	.8	2	0.500	4
.2	.5	3	0.360	7
.5	.7	3	0.360	6



Calculate Smoothing factor

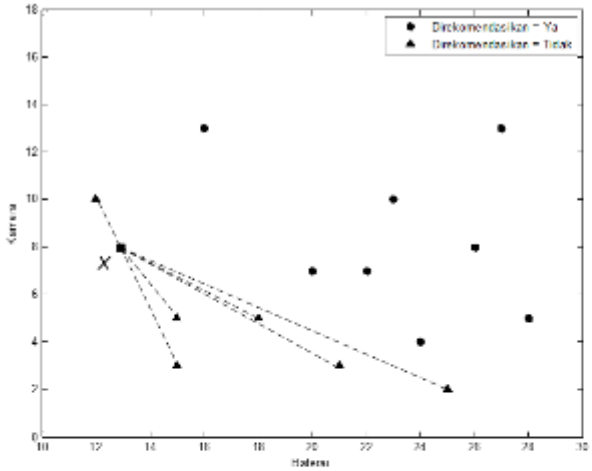
x_1	x_2	y	dist
.3	.2	1	0.316
.6	.3	1	0.282
.8	.5	1	0.282
.4	.5	2	0.500
.8	.8	2	0.500
.2	.5	3	0.360
.5	.7	3	0.360

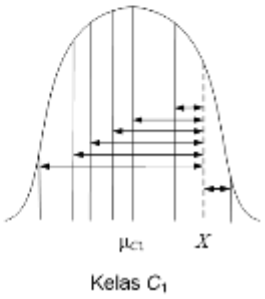
$$dtot_1 = 0.881$$
$$dtot_2 = 1$$
$$dtot_3 = 0.721$$

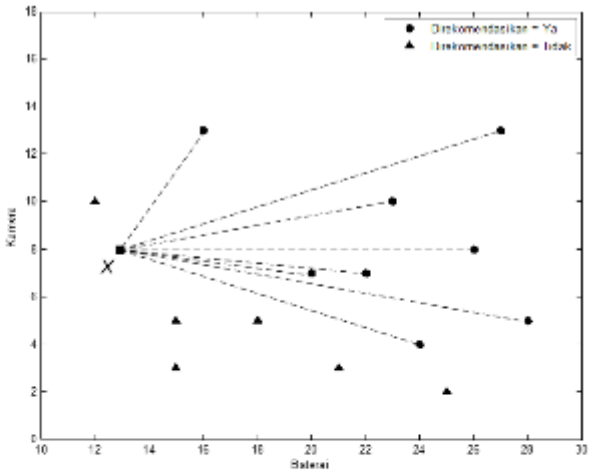
$$\sigma_c = (g.dtot_c)/n_c$$
$$\sigma_1 = (g.dtot_1)/n_1 = (g * 0.811)/3$$
$$\sigma_2 = (g.dtot_2)/n_2 = (g * 1)/2$$
$$\sigma_3 = (g.dtot_3)/n_3 = (g * 0.721)/2$$

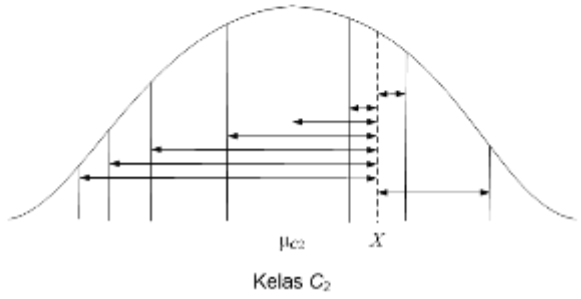
Brute force to find g value that gives best classification rate

Data Distance





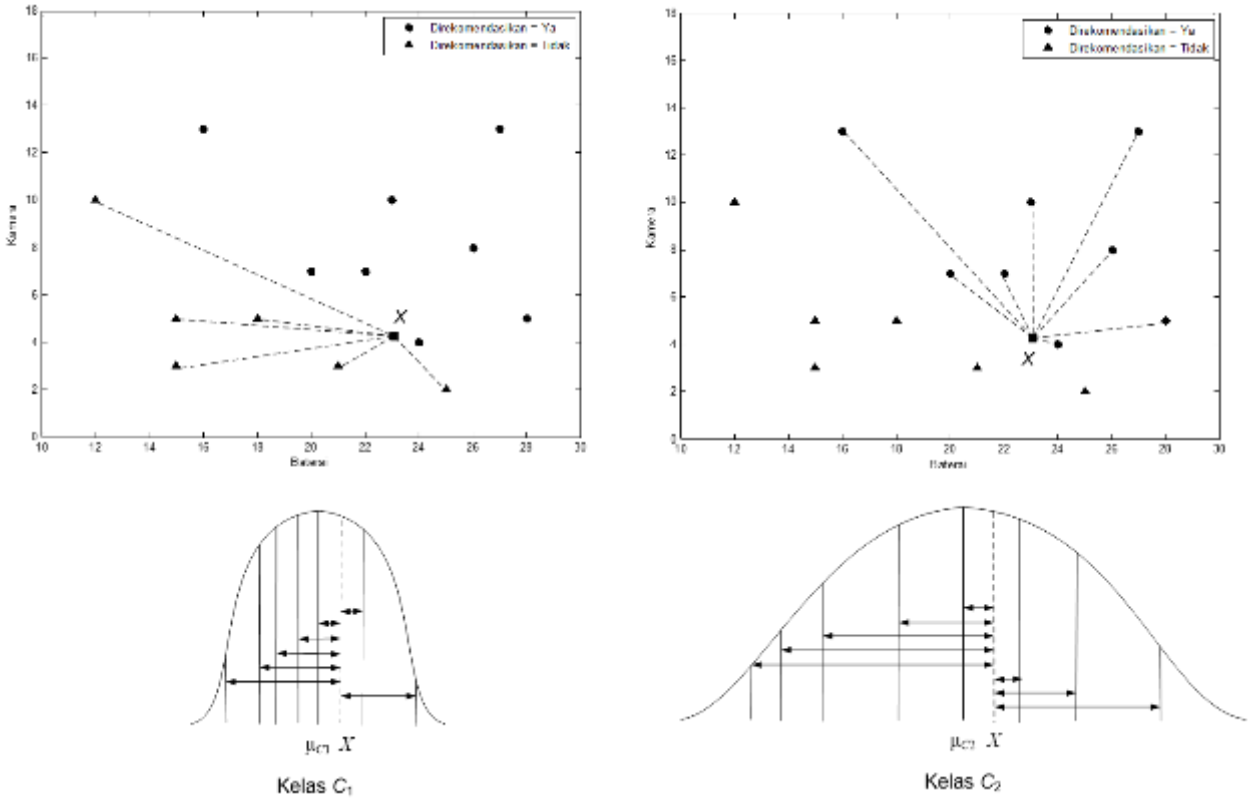




Smoothing parameter in C1 and C2 is far different.

Thus without parameter g, the gap between two smoothing functions is big enough to support a decision

Data Distance



Smoothing parameter in C1 and C2 is similar

Using parameter g, the gap between two smoothing functions can be enlarged to give a better support in decision

Probability Density Function

Assume, after using parameter g, each σ_k = 0.1

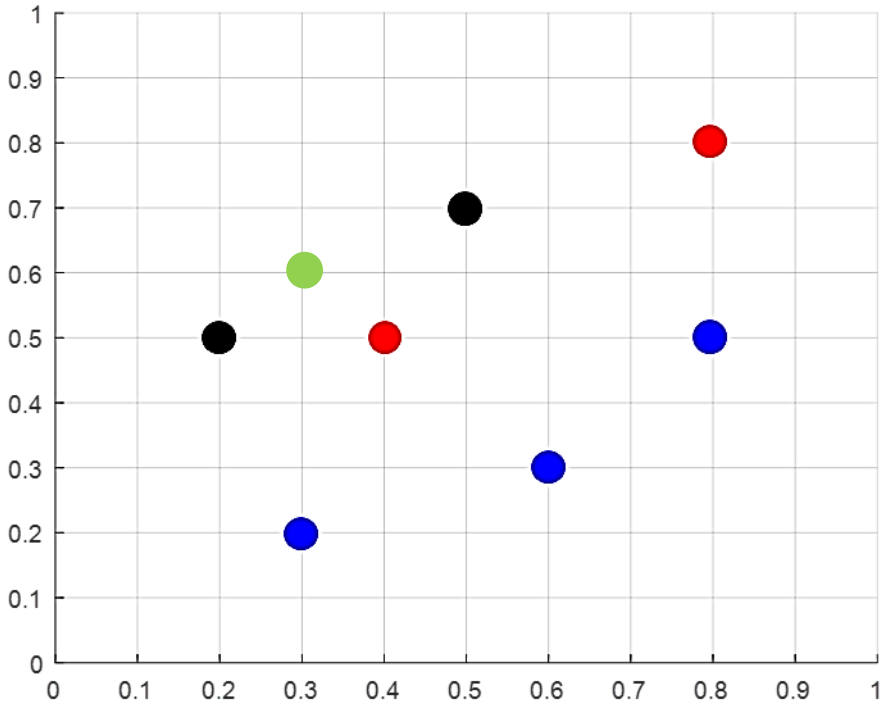
x_1	x_2	y	$g_i(\mathbf{x}) = \exp\left(-\frac{\|\|\mathbf{x}-\mathbf{x}_i\|\|^2}{2\sigma^2}\right)$
.3	.2	1	$g_1(x) = \exp\{-\frac{(x_1-.3)^2+(x_2-.2)^2}{2(.1)^2}\}$
.6	.3	1	$g_2(x) = \exp\{-\frac{(x_1-.6)^2+(x_2-.3)^2}{2(.1)^2}\}$
.8	.5	1	$g_3(x) = \exp\{-\frac{(x_1-.8)^2+(x_2-.5)^2}{2(.1)^2}\}$
.4	.5	2	$g_4(x) = \exp\{-\frac{(x_1-.4)^2+(x_2-.5)^2}{2(.1)^2}\}$
.8	.8	2	$g_5(x) = \exp\{-\frac{(x_1-.8)^2+(x_2-.8)^2}{2(.1)^2}\}$
.2	.5	3	$g_6(x) = \exp\{-\frac{(x_1-.2)^2+(x_2-.5)^2}{2(.1)^2}\}$
.5	.7	3	$g_7(x) = \exp\{-\frac{(x_1-.5)^2+(x_2-.7)^2}{2(.1)^2}\}$

$$f_1(\mathbf{x}) = g_1(\mathbf{x}) + g_2(\mathbf{x}) + g_3(\mathbf{x})$$

$$f_2(\mathbf{x}) = g_4(\mathbf{x}) + g_5(\mathbf{x})$$

$$f_3(\mathbf{x}) = g_6(\mathbf{x}) + g_7(\mathbf{x})$$

Testing on new data



x_1	x_2	y
.3	.2	1
.6	.3	1
.8	.5	1
.4	.5	2
.8	.8	2
.2	.5	3
.5	.7	3
.2	.6	??

Testing on new data

Assume, after using parameter g, each σ_k = 0.1

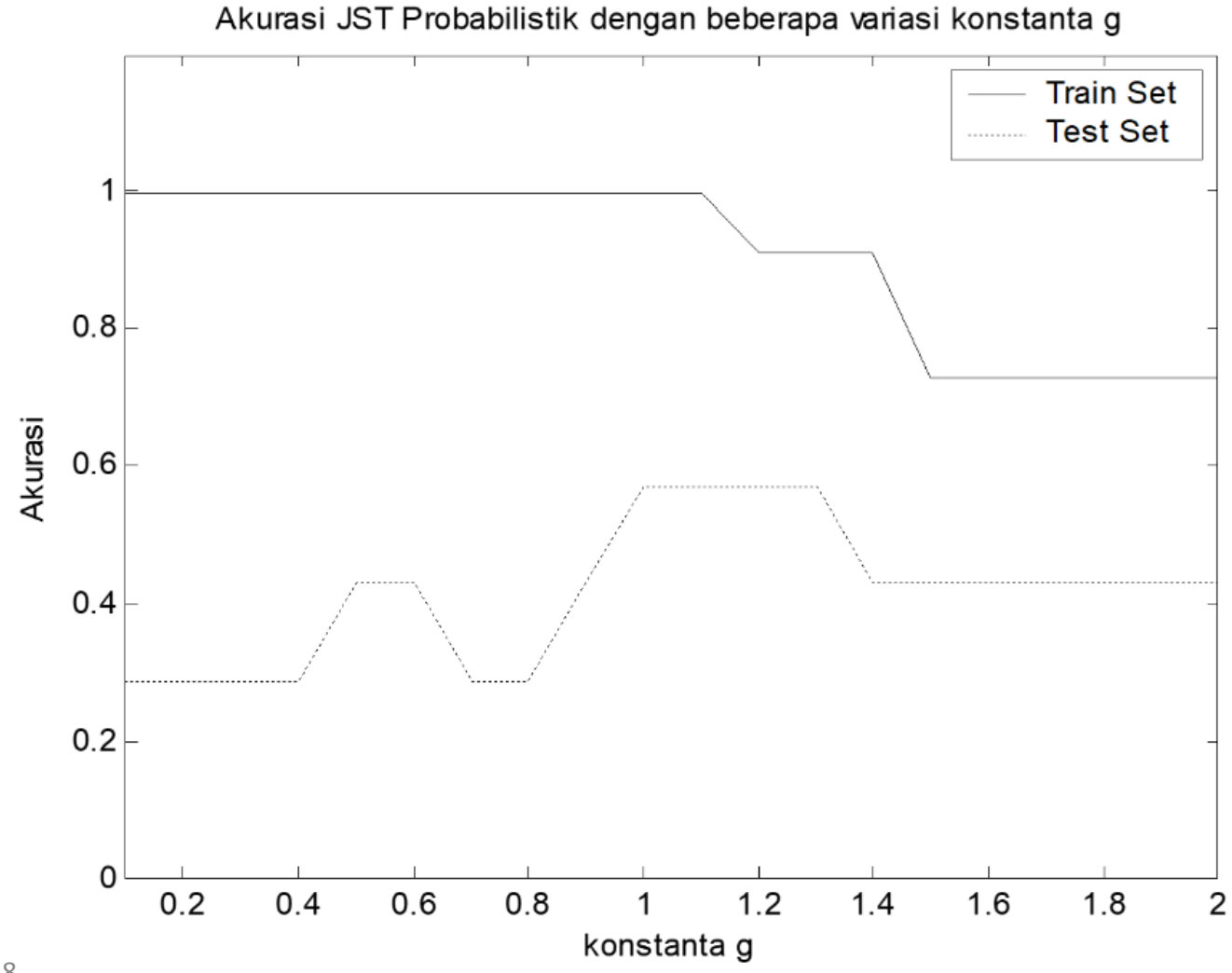
x_1	x_2	y	$g_i(x) = \exp\left(-\frac{\|\|x-x_k\|\|^2}{2\sigma^2}\right)$	$g_i(x)$
.3	.2	1	$g_1(x) = \exp\{-\frac{(x_1-.3)^2+(x_1-.2)^2}{2(.1)^2}\}$	0
.6	.3	1	$g_2(x) = \exp\{-\frac{(x_1-.6)^2+(x_1-.3)^2}{2(.1)^2}\}$	0
.8	.5	1	$g_3(x) = \exp\{-\frac{(x_1-.8)^2+(x_1-.5)^2}{2(.1)^2}\}$	0
.4	.5	2	$g_4(x) = \exp\{-\frac{(x_1-.4)^2+(x_1-.5)^2}{2(.1)^2}\}$	0.082
.8	.8	2	$g_5(x) = \exp\{-\frac{(x_1-.8)^2+(x_1-.8)^2}{2(.1)^2}\}$	0
.2	.5	3	$g_6(x) = \exp\{-\frac{(x_1-.2)^2+(x_1-.5)^2}{2(.1)^2}\}$	0.606
.5	.7	3	$g_7(x) = \exp\{-\frac{(x_1-.5)^2+(x_1-.7)^2}{2(.1)^2}\}$	0.006

$$\mathbf{x} = (0.2, 0.6)$$

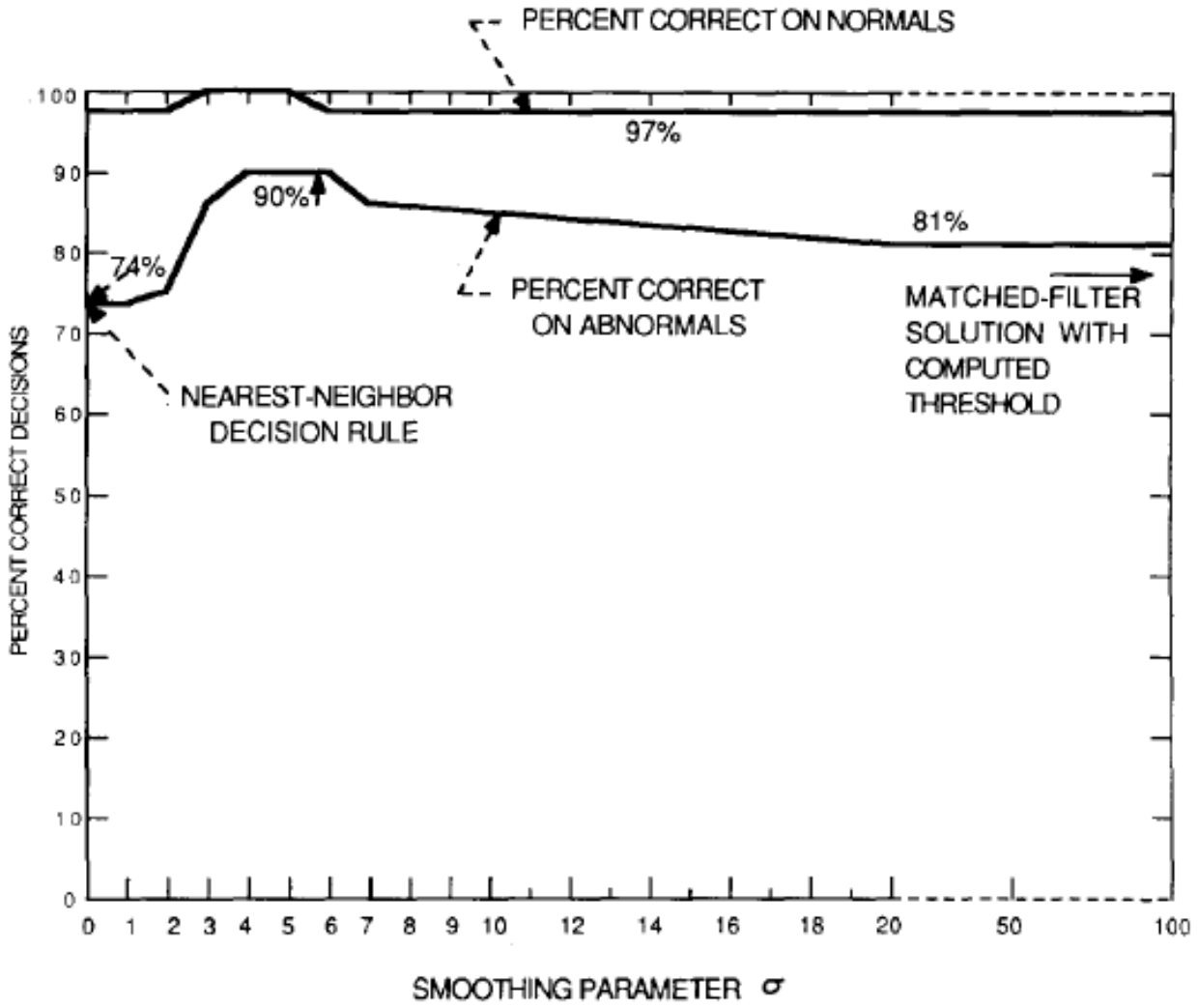
$$f_1(\mathbf{x}) = g_1(\mathbf{x}) + g_2(\mathbf{x}) + g_3(\mathbf{x})$$
$$f_2(\mathbf{x}) = g_4(\mathbf{x}) + g_5(\mathbf{x})$$
$$f_3(\mathbf{x}) = g_6(\mathbf{x}) + g_7(\mathbf{x})$$

$$f_1(\mathbf{x}) = 0$$
$$f_2(\mathbf{x}) = 0.082$$
$$f_3(\mathbf{x}) = 0.612$$

Smoothing parameter σ controlled by parameter g



Equal smoothing parameter σ for all classes



Advantages and Disadvantages

- Advantages
  - Fast training process
    - Orders of magnitude faster than backpropagation
  - An inherently parallel structure
  - Guaranteed to converge to an optimal classifier as the size of the representative training set increases
    - No local minima issues
  - Training samples can be added or removed without extensive retraining
- Disadvantages
  - Not as general as backpropagation
  - Large memory requirements
  - Slow execution of the network
  - Requires a representative training set
    - Even more so than other types of NN's
  - It is vital to find an accurate smoothing parameter (σ)

Further information

- Sometimes, vectors (in training set and test set) had better to be normalized to unit length
- For example, a p-dimensional vector $\mathbf{x} = [x_1, x_2, \ldots, x_p]$, it becomes $\mathbf{z}$

$$\mathbf{z} = \frac{\mathbf{x}}{norm(\mathbf{x})} = \frac{\mathbf{x}}{\sqrt{x_1^2 + x_2^2 + \cdots + x_p^2}}$$