Nonlinear Classifiers II

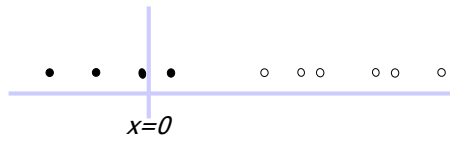
Nonlinear Classifiers: Introduction

- Classifiers
  - Supervised Classifiers
    - **XOR problem**
    - Linear Classifiers
      - Perceptron
      - Least Squares Methods
      - Linear Support Vector Machine
    - **Nonlinear Classifiers**
      - Part I: Multi Layer Neural Networks
      - **Part II: Polynomial Classifier, RBF, Nonlinear SVM**
    - Decision Trees
  - Unsupervised Classifiers

Nonlinear Classifiers: Introduction

- An example: Suppose we're in 1-dimension

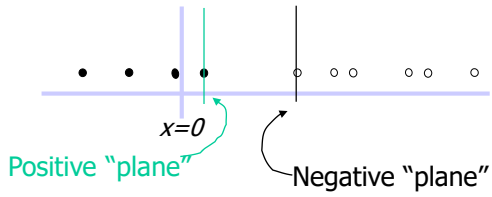
What would a linear SVMs do with this data?



Nonlinear Classifiers: Introduction

- An example: Suppose we're in 1-dimension

Not a big surprise

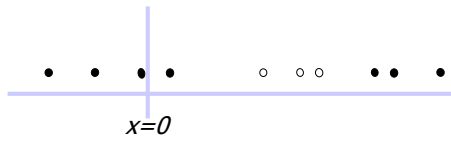


Nonlinear Classifiers: Introduction

5

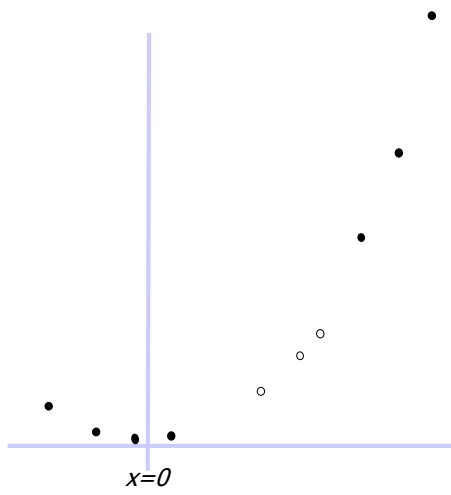
- Harder 1-dimensional dataset

What can be done about this?



Nonlinear Classifiers: Introduction

6

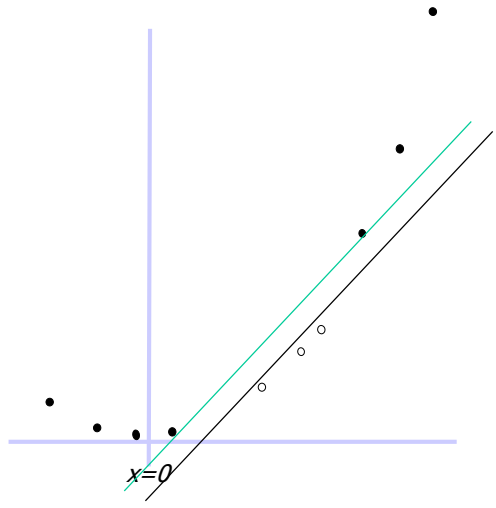


non-linear basis function

$$\mathbf{z}_k = (x_k, x_k^2)$$

Nonlinear Classifiers: Introduction

7

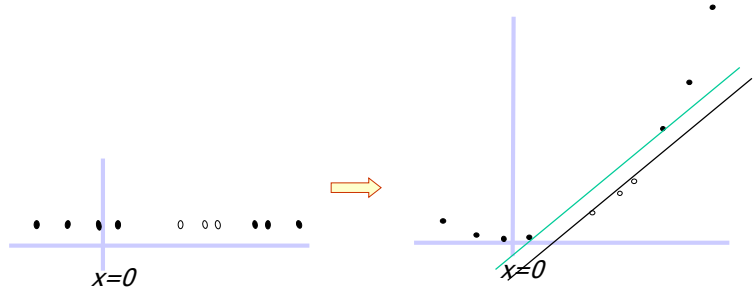


non-linear basis function

$$\mathbf{z}_k = (x_k, x_k^2)$$

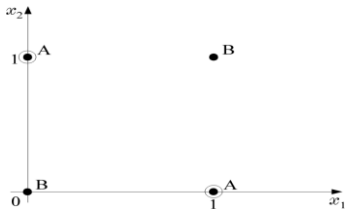
Nonlinear Classifiers: Introduction

8



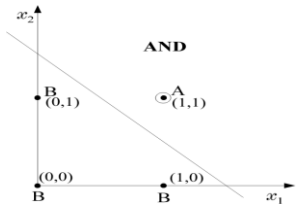
- **Linear classifiers** are simple and computationally efficient.
- However for nonlinearly separable features, they might lead to very inaccurate decisions.
- Then we may trade simplicity and efficiency for accuracy using a **nonlinear classifier**.

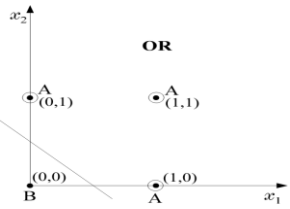
The XOR problem



x_1	x_2	XOR	Class
0	0	0	B
0	1	1	A
1	0	1	A
1	1	0	B

- There is no single line (hyperplane) that separates class A from class B. On the contrary, AND and OR operations are linearly separable problems.





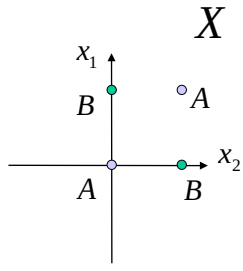
Nonlinear Classifiers: Agenda

Part II: Nonlinear Classifiers

- **Polynomial Classifier**
  - Special case of a Two-Layer Perceptron
  - Activation function with non linear input
- Radial Basis Function Network
  - Special case of a two-layer network
  - Radial Basis activation Function
  - Training is simpler and faster
- Nonlinear Support Vector Machine

Polynomial Classifier: XOR problem

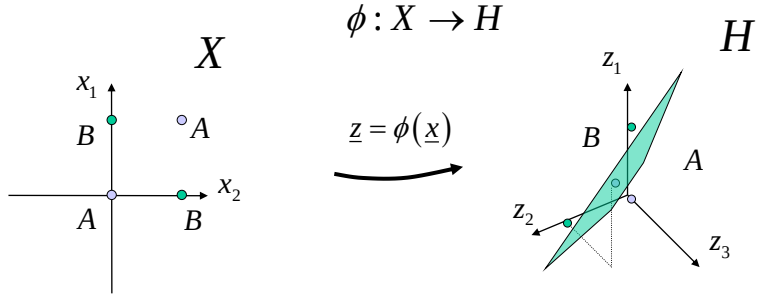
- XOR problem with polynomial function.
  - With nonlinear polynomial function classes can be classified.
  - Example XOR-Problem:



linear not separable!

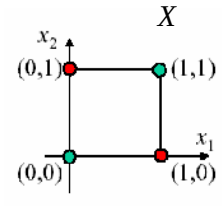
Polynomial Classifier: XOR problem

- XOR problem with polynomial function.
  - With nonlinear polynomial functions, classes can be classified.
  - Example XOR-Problem:

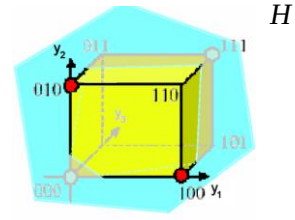


…but with a polynomial function!

Polynomial Classifier: XOR problem



$\underline{z} = \phi(\underline{x})$



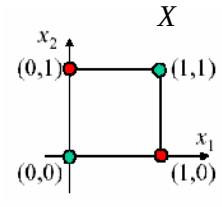
With $\underline{z} = \begin{bmatrix} x_1 \\ x_2 \\ x_1 x_2 \end{bmatrix}$ we obtain:

$\phi(0,0) \rightarrow (0,0,0)$
$\phi(0,1) \rightarrow (0,1,0)$
$\phi(1,0) \rightarrow (1,0,0)$
$\phi(1,1) \rightarrow (1,1,1)$

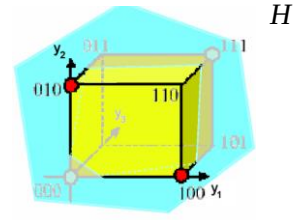
… that's separable in H by the Hyperplane:

$$g(\underline{z}) = \frac{1}{4} - 1z_1 - 1z_2 + 2z_3 = 0$$

Polynomial Classifier: XOR problem



$\underline{z} = \phi(\underline{x})$



Hyperplane: $g(\underline{z}) = \underline{w}\underline{z} + w_0 = 0$

$$g(\underline{z}) = \frac{1}{4} - z_1 - z_2 + 2z_3 = 0$$

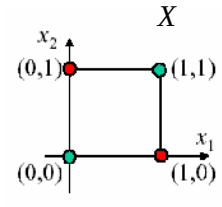
is Hyperplane in H

$$g(\underline{x}) = \frac{1}{4} - x_1 - x_2 + 2x_1x_2$$

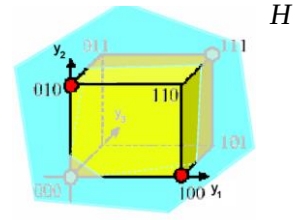
is Polynom in X

X		H			
		z_1	z_2	z_3	
x_1	x_2	x_1	x_2	x_1x_2	$\underline{\omega}$
0	0	0	0	0	A(true)
0	1	0	1	0	B(false)
1	0	1	0	0	B(false)
1	1	1	1	1	A(true)

Polynomial Classifier: XOR problem



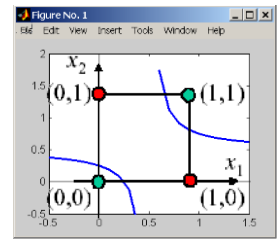
$$\underline{z} = \phi(\underline{x})$$



Decision Surface in X

$$g(\underline{x}) = \frac{1}{4} - 1x_1 - 1x_2 + 2x_1x_2 \quad \begin{matrix} \geq 0 & \underline{x} \in A \\ < 0 & \underline{x} \in B \end{matrix}$$

$$x_2 = (x_1\text{-}0.25)/(2x_1\text{-}1)$$

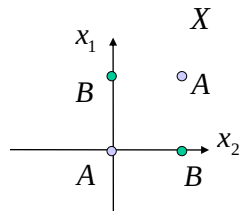


MatLab:

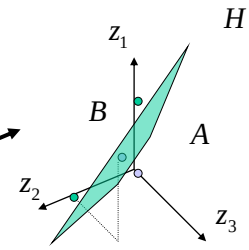
```
>> x1=[-0.5:0.1:1.5];
>> x2=(x1-0.25)./(2*x1-1);
>> plot(x1,x2);
```

Polynomial Classifier: XOR problem

- With nonlinear polynomial functions, classes can be classified in original space X
  - Example: XOR-Problem



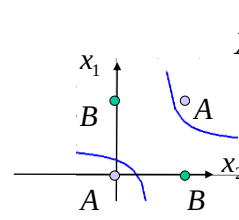
$$\underline{z} = \phi(\underline{x})$$



was not linear separable!

… but linear separable in H !

… and separable in X with a polynomial function!



Polynomial Classifier

more general

- Decision function is approximated by a polynomial function $g(x)$, of order p e.g. $p = 2$:

$$g(\underline{x}) = w_0 + \sum_{i=1}^{l} w_i x_i + \sum_{i=1}^{l-1} \sum_{m=i+1}^{l} w_{im} x_i x_m + \sum_{i=1}^{l} w_{ii} x_i^2$$

$$g(\underline{x}) = \underline{w}^T \underline{z} + w_0,$$

with

$$\underline{w}^T = \left[w_1, w_2, w_{12}, w_{11}, w_{22} \right],$$

$$\underline{z} = \left[x_1, x_2, x_1 x_2, x_1^2, x_2^2 \right]^T \text{ and } \underline{x} = \left[x_1, x_2 \right]^T$$

  - Special case of a Two-Layer Perceptron
  - Activation function with polynomial input

Nonlinear Classifiers: Agenda

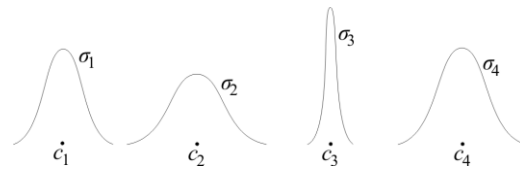
Part II: Nonlinear Classifiers

- Polynomial Classifier
- **Radial Basis Function Network**
  - Special case of a two-layer network
  - Radial Basis activation Function
  - Training is simpler and faster
- Nonlinear Support Vector Machine
- Application: ZIP Code, OCR, FD (W-RVM)
- Demo: libSVM, DHS or Hlavac

Radial Basis Function

- Radial Basis Function Networks (RBF)
  - Choose $g(\underline{x}) = w_0 + \sum_{i=1}^{k} w_i g_i(\underline{x})$

with $g_i(\underline{x}) = \exp\left(-\frac{\|\underline{x}-\underline{c}_i\|^2}{2\sigma_i^2}\right)$

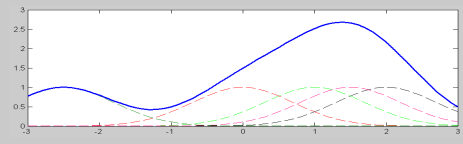


Radial Basis Function

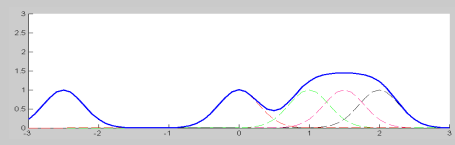
$g(\underline{x}) = w_0 + \sum_{i=1}^{k} w_i g_i(\underline{x})$ with $g_i(\underline{x}) = \exp\left(-\frac{\|\underline{x}-\underline{c}_i\|^2}{2\sigma_i^2}\right)$

Examples:

$c_i = -2.5,\ 0.0,\ 1.0,\ 1.5,\ 2.0,$
$i = 1,\ldots,k,$
$k = 5,$
$\sigma = 1/\sqrt{2}$



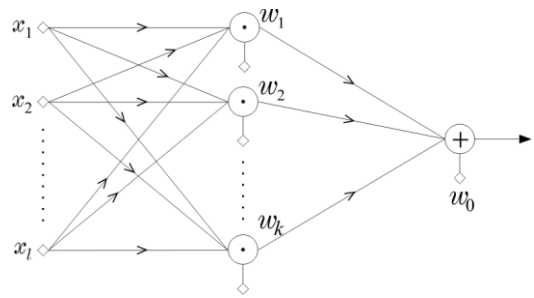
$c_i = -2.5,\ 0.0,\ 1.0,\ 1.5,\ 2.0$
$i = 1,\ldots,k,$
$k = 5,$
$\sigma = 1/\sqrt{12}$



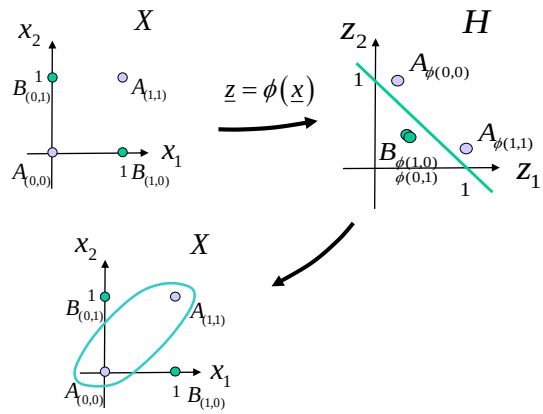
How to choose c_i, σ_i, k?

Radial Basis Function

- Radial Basis Function Networks (RBF)
  - Equivalent to a single layer network, with RBF activations and linear output node.



Radial Basis Function: XOR problem



$$\underline{z} = \phi(\underline{x}) = \begin{bmatrix} \exp(-\|\underline{x} - \underline{c}_1\|^2) \\ \exp(-\|\underline{x} - \underline{c}_2\|^2) \end{bmatrix}$$

$$\underline{c}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \underline{c}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \sigma_1 = \sigma_2 = \frac{1}{\sqrt{2}}$$

$$\Rightarrow \phi: \quad \begin{bmatrix} 0 \\ 0 \end{bmatrix} \to \begin{bmatrix} 0.135 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} \to \begin{bmatrix} 1 \\ 0.135 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \to \begin{bmatrix} 0.368 \\ 0.368 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} \to \begin{bmatrix} 0.368 \\ 0.368 \end{bmatrix}$$

$$g(\underline{z}) = z_1 + z_2 - 1 = 0$$

$$g(\underline{x}) = \exp(-\|\underline{x} - \underline{c}_1\|^2) + \exp(-\|\underline{x} - \underline{c}_2\|^2) - 1 = 0$$

… not linear separable pattern set in X.

… separable using a nonlinear function (RBF) in X that separates the set in H with a linear decision hyperplane!

Radial Basis Function

- Decision function as summation of k RBF's

$$g(\underline{x}) = w_0 + \sum_{i=1}^{k} w_i \exp\left(-\frac{(\underline{x}-\underline{c}_i)^T (\underline{x}-\underline{c}_i)}{2\sigma_i^2} \right)$$

- Training of the RBF networks
  1. Fixed centers: Choose centers randomly among the data points. Also fix σ_i's. Then $g(\underline{x}) = w_0 + \underline{w}^T \underline{z}$ is a typical linear classifier design.
  2. Training of the centers c_i: This is a nonlinear optimization task.
  3. Combine supervised and unsupervised learning procedures.
  4. The unsupervised part reveals clustering tendencies of the data and assigns the centers at the cluster representatives.

Nonlinear Classifiers: Agenda

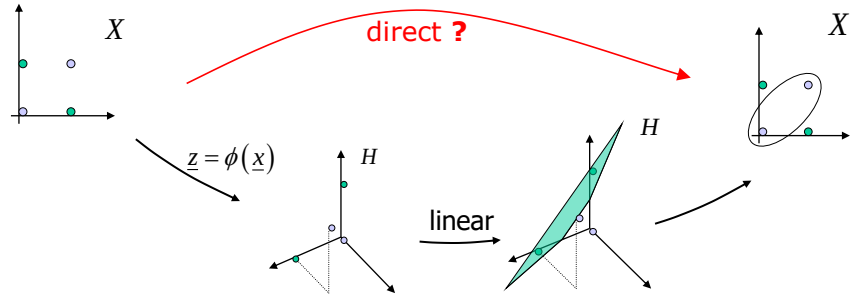
Part II: Nonlinear Classifier

- Polynomial Classifier
- Radial Basis Function Network
- **Nonlinear Support Vector Machine**
- Application: ZIP Code, OCR, FD (W-RVM)
- Demo: libSVM, DHS or Hlavac

Nonlinear Classifiers: SVM

XOR problem:

- linear separation in high dimensional space H via nonlinear functions (polynomial and RBF's) in the original space X.
- for this we found nonlinear mappings $\phi(x)$: $X \rightarrow H$



Is that possible without knowing the mapping function ϕ ?!?

Non-linear Support Vector Machines

– Recall that, the probability of having linearly separable classes increases as the **dimensionality** of feature vectors **increases**.

Assume the mapping:

$$\underline{x} \in R^l \rightarrow \underline{z} \in R^k, \quad k > l$$

-> Then use linear SVM in R^k

Non-linear SVM

- Support Vector Machines: with $\underline{x} \rightarrow \underline{z} \in R^k$
  - Recall that in this case the dual problem formulation will be

$$\arg\max_{\underline{\lambda}} \left(\sum_{i}^{N} \lambda_i - \frac{1}{2} \sum_{i,j}^{N} \lambda_i \lambda_j y_i y_j z_i^T z_j \right) \text{ subject to } \sum_{i=1}^{N} \lambda_i y_i = 0, \quad \lambda_i \geq 0$$

where $\underline{z}_i \in R^k$, $y \in \{-1,1\}$ (class labels)

  - the classifier will be

$$g(\underline{z}) = \underline{w}^T \underline{z} + w_0 = \sum_{i=1}^{N_s} \lambda_i y_i \underline{z}_i^T \underline{z} + w_0$$

Non-linear SVM

- Thus, only inner products in a high dimensional space are needed!

=> Something clever (**kernel trick**):
Compute the inner products in the **high** dimensional space as functions of inner products performed in the **low** dimensional space!!!

Non-linear SVM

– Is this POSSIBLE?? Yes. Here is an example

Let $\underline{x} = [x_1, x_2]^T \in R^2$

Let $\underline{x} \to \underline{z} = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix} \in R^3$

It is easy to show that $\underline{z}_i^T \underline{z}_j = (\underline{x}_i^T \underline{x}_j)^2$

$$(\underline{x}_i^T \underline{x}_j)^2 = \left(x_{i1}x_{j1} + x_{i2}x_{j2}\right)^2$$

$$= x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{j1}x_{i2}x_{j2} + x_{i2}^2 x_{j2}^2$$

$$= \left(x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2\right) \begin{pmatrix} x_{j1}^2 \\ \sqrt{2}x_{j1}x_{j2} \\ x_{j2}^2 \end{pmatrix} = \underline{z}_i^T \underline{z}_j$$

Non-linear SVM

- Mercer's Theorem

Let $\underline{x} \to \underline{\phi}(\underline{x}) \in H$

To guarantee that the symmetric function $K(\underline{x}_i, \underline{x}_j)$ (kernel) can be represented as

$$\sum_r \phi_r(\underline{x}_i)\phi_r(\underline{x}_j) = K(\underline{x}_i, \underline{x}_j)$$

that is an inner product in H,

it is necessary and sufficient that

$$\int K(\underline{x}_i, \underline{x}_j) g(\underline{x}_i) g(\underline{x}_j) d\underline{x}_i d\underline{x}_j \geq 0 \qquad (1)$$

for any $g(\underline{x})$: $\int g^2(\underline{x})\, d\underline{x} < +\infty$ (2)

Non-linear SVM

- Kernel Function
  - So, any kernel $K(\underline{x},\underline{y})$ satisfying (1) & (2), corresponds to an inner product in SOME space!!!
  - Kernel trick: We do not have to know the mapping function $\Phi(x)$, but for some kernel functions we try to linearly separate pattern sets in a high dimensional space only using a function of the inner product in the original space.

Non-linear SVM

- Kernel Functions: Examples
  - Polynomial: $K(\underline{x}_i,\underline{x}_j)=(\underline{x}_i^T\underline{x}_j+1)^q,\ \ q>0$
  - Radial Basis Functions:

$$K(\underline{x}_i,\underline{x}_j)=\exp\left(-\frac{\|\underline{x}_i-\underline{x}_j\|^2}{\sigma^2}\right)$$

  - Hyperbolic Tangent:

$$K(\underline{x}_i,\underline{x}_j)=\tanh(\beta\underline{x}_i^T\underline{x}_j+\gamma)$$

for appropriate values of β, γ (e.g. β =2 and γ =1).

Non-linear SVM

Support Vector Machines Formulation

– Step 1: Choose appropriate kernel. This implicitly assumes a mapping to a higher dimensional (yet, not known) space.

Non-linear SVM

SVM Formulation

- Step 2:

$$\underline{\lambda} = \arg\max_{\underline{\lambda}}\left(\sum_i \lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j K(\underline{x}_i, \underline{x}_j)\right)$$

$$\text{subject to:} \quad 0 \le \lambda_i \le C, \quad i = 1, 2, ..., N$$

$$\sum_i \lambda_i y_i = 0$$

This results to an implicit combination

$$\underline{w} = \sum_{i=1}^{N_s} \lambda_i y_i \underline{\Phi}(\underline{x}_i)$$

Non-linear SVM

– SVM Formulation

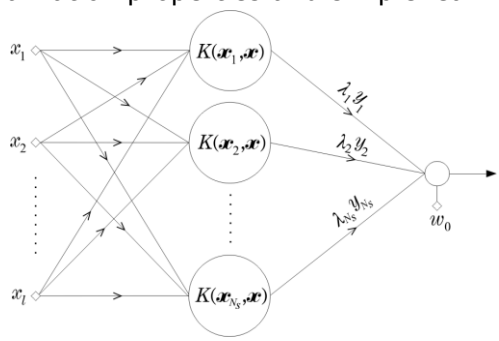
- Step 3: Assign $\underline{x}$ to

$$\omega_1 \text{ if } g(\underline{x}) = \sum_{i=1}^{N_s} \lambda_i y_i \, K(\underline{x}_i, \underline{x}) + w_0 \geq 0$$

$$\omega_2 \text{ if } g(\underline{x}) = \sum_{i=1}^{N_s} \lambda_i y_i \, K(\underline{x}_i, \underline{x}) + w_0 < 0$$

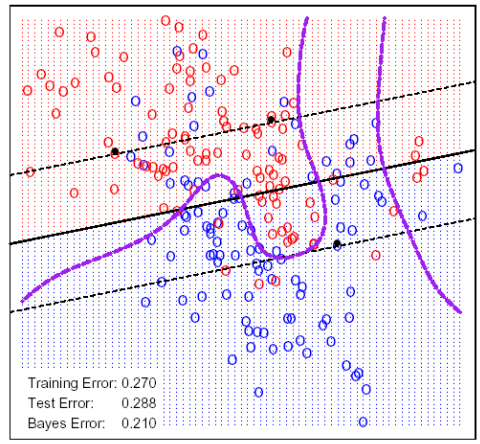
Non-linear SVM

- SVM: The non-linear case
  - The SVM Architecture
  - SVM special case of a two-layer neural network with special activation function and a different learning method.
  - Their attractiveness comes from their good generalization properties and simple learning.



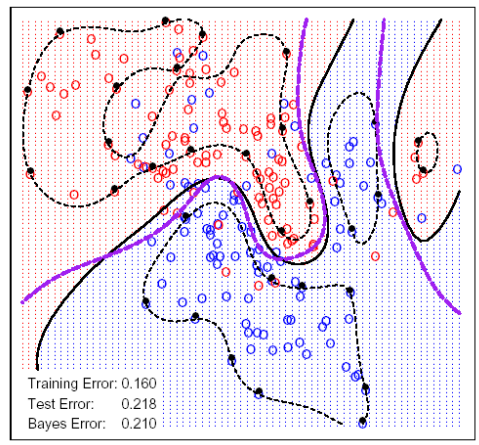
Non-linear SVM

- Linear SVM – Pol. SVM in the input space X



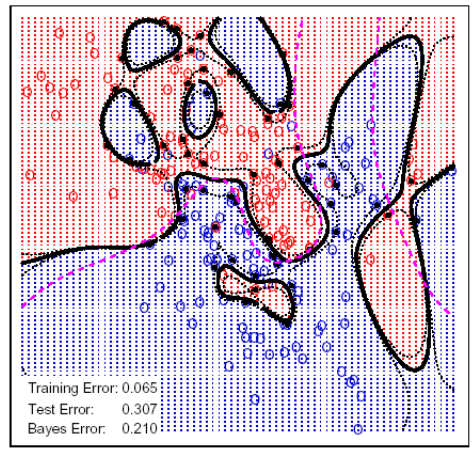
Non-linear SVM

- Pol. SVM – RBF SVM in the input space X



Nonlinear Classifiers: SVM

- Pol. SVM – RBF SVM in the input space X



Nonlinear Classifiers: SVM

- Software

  - SVM^{light}: Thorsten Joachims - free software in C, known for quality and speed.
  - *LIB SVM*: free software based on Platt's SMO algorithm and Joachims code, written by Chih-Chung Chang and Chih-Jen Lin.
  - *Equbits*: Commercial software package which automates the tuning and model selection with SVMs