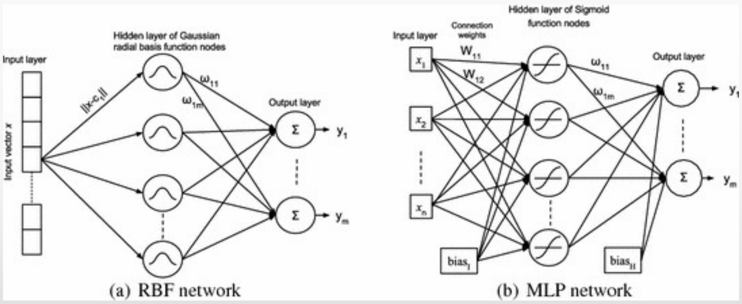
RBF Networks and Support Vector Machine (SVM)

Radial Basis Function (RBF) Networks

INTRODUCTION

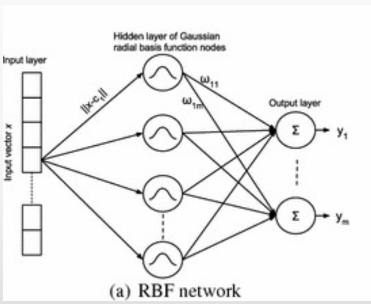
- A radial basis function (RBF) network is an artificial neural network that uses RBFs as activation functions. The output of the network is a linear combination of radial basis functions of the inputs and neuron parameters.



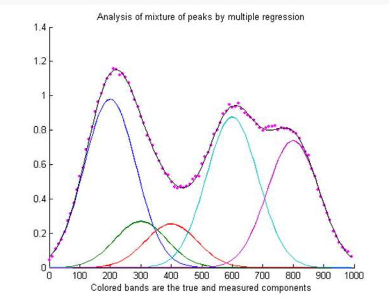
(a) RBF network (b) MLP network

INTRODUCTION

- On the other hand, an RBF neuron measures similarity or distance. RBF network can be seen as a completely different approach by <u>designing a neural network as a curve-fitting (approximation) problem in high-dimensional space.</u>

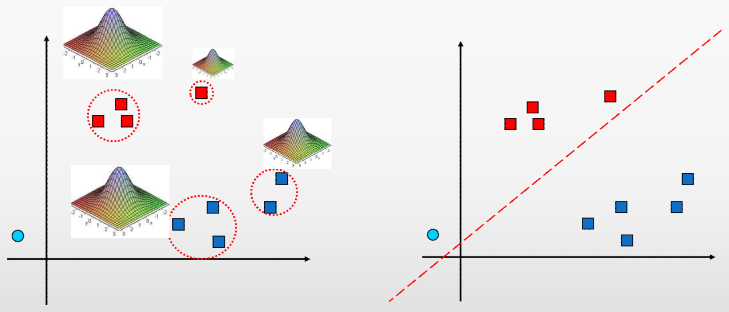


(a) RBF network



Introduction

- Similar to a MLP, an RBF network is a kind of supervised neural network. However, it designs a neural network as curve-fitting problem. That is, approximate function with linear combination of Radial basis functions



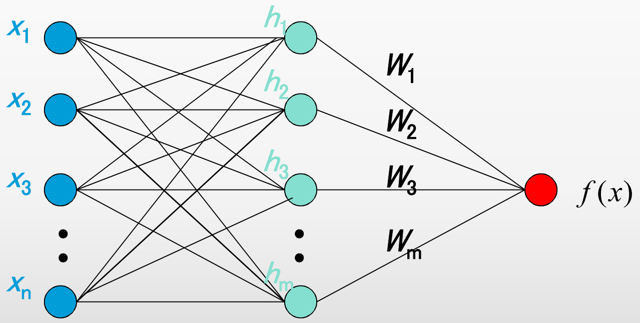
In RBFN In MLP

introduction

Radial Basis Function Network (summary)

- A kind of supervised neural networks
- Design of NN as curve-fitting problem. That is, approximate function with linear combination of Radial basis functions
- Learning
  - find surface in multidimensional space best fit to training data
- Generalization
  - Use of this multidimensional surface <u>to interpolate the test data</u>

ARCHITECTURE



Input layer Hidden layer Output layer

architecture

Radial basis function

$$f(x) = \sum_{j=1}^{m} w_j h_j(x)$$

$$h_j(x) = \exp\left(-\beta_j \| x - c_j \|^2\right)$$

Where c_j is center of a region,

β_j is a parameter describing the width of the receptive field

DESIGNING

- Require
  - Determine the number of radial basis neurons
  - Determine the radial basis function centers
  - Select the radial basis function width parameter
    - ✓ smaller width
      alerting in untrained test data
    - ✓ larger width
      network of smaller size & faster execution

learning strategies

CAN ALSO BE TRAINED LIKE AN MLP (1/2)

$$E=\frac{1}{2}\sum_{i=1}^{N}(d_i(x)-f_i(x))^2=\frac{1}{2}\sum_{i=1}^{N}e_i(x)^2 \qquad f(x)=\sum_{j=1}^{m}w_j h_j(x)$$

$$h_j(x)=\exp\left(-\beta_j \|x-c_j\|^2\right)$$

- Linear weights (output layer)

$$w_j \leftarrow w_j-\eta_1\frac{\partial E}{\partial w_j}, \qquad j=1,2,\ldots,m$$

- Positions of centers (hidden layer)

$$c_j \leftarrow c_j-\eta_2\frac{\partial E}{\partial c_j}, \qquad j=1,2,\ldots,m$$

- Spreads of centers (hidden layer)

$$\beta_j \leftarrow \beta_j-\eta_3\frac{\partial E}{\partial \beta_j}, \qquad j=1,2,\ldots,m$$

learning strategies

CAN ALSO BE TRAINED LIKE AN MLP (2/2)

$$E = \frac{1}{2}\sum_{i=1}^{N}(d_i(x) - f_i(x))^2 = \frac{1}{2}\sum_{i=1}^{N} e_i(x)^2 \qquad f(x) = \sum_{j=1}^{m} w_j h_j(x)$$

$$h_j(x) = \exp\left(-\beta_j \| x - c_j \|^2\right)$$

- Gradient of linear weights (output layer)

$$\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial e_i(x)} \frac{\partial e_i(x)}{\partial w_j} = -\sum_{i=1}^{N} e_i(x) h_j(x), \qquad j = 1,2,\ldots,m$$

- Gradient of positions of centers (hidden layer)

$$\frac{\partial E}{\partial c_j} = \frac{\partial E}{\partial e_i(x)} \frac{\partial e_i(x)}{\partial h_j(x)} \frac{\partial h_j(x)}{\partial c_j} = -\sum_{i=1}^{N} e_i(x) w_j h_j(x) \beta_j \| x - c_j \|, \qquad j = 1,2,\ldots,m$$

- Gradient of spreads of centers (hidden layer)

$$\frac{\partial E}{\partial \beta_j} = \frac{\partial E}{\partial e_i(x)} \frac{\partial e_i(x)}{\partial h_j(x)} \frac{\partial h_j(x)}{\partial \beta_j} = \sum_{i=1}^{N} e_i(x) w_j h_j(x) \| x - c_j \|^2, \qquad j = 1,2,\ldots,m$$

MLP VS RBFN

Global hyperplane	Local receptive field
EBP (Error BP)	LMS (Least mean square)
Local minima	Serious local minima
Smaller number of hidden neurons	Larger number of hidden neurons
Shorter computation time	Longer computation time
Longer learning time	Shorter learning time

MLP vs RBFN

Approximation

- MLP : Global network
  - All inputs cause an output
- RBF : Local network
  - Only inputs near a receptive field produce an activation
  - Can give "don't know" output

Support Vector Machine (SVM)

SVM : a margin optimized classifier

SVMs developed by Vapnik in 1963, have gained wide acceptance because of their high generalization ability for a wide range of applications in 1990's.

Introduction

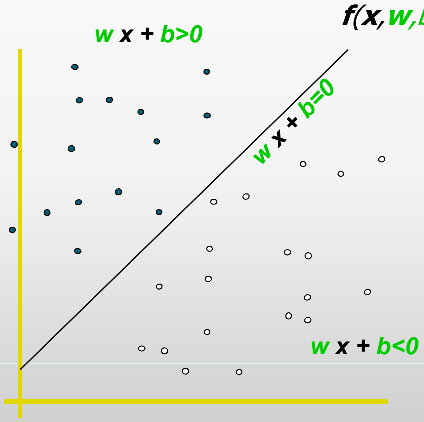
-- SVM : a margin optimized classifier

- Linear classifier

• denotes +1

∘ denotes -1

$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}\ \mathbf{x} + b)$



How would you classify this data?

Introduction

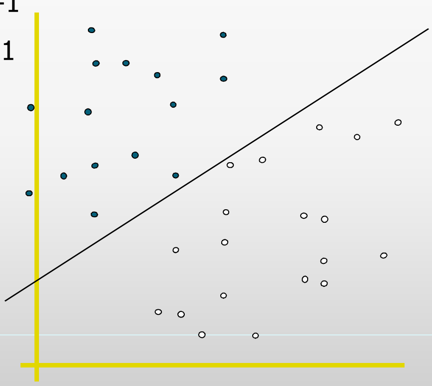
-- SVM : a margin optimized classifier

- Linear classifier

• denotes +1

∘ denotes -1

$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}\ \mathbf{x} + b)$



How would you classify this data?

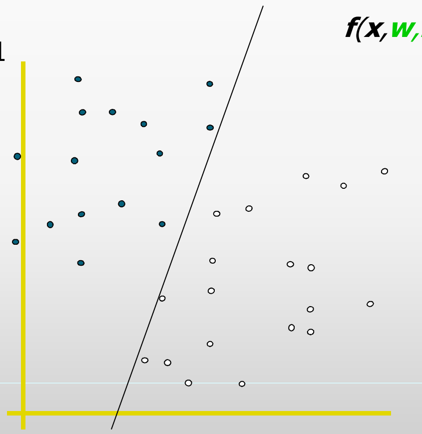
Introduction

-- SVM : a margin optimized classifier

- Linear classifier

$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\ \mathbf{x} + b)$

- denotes +1
- denotes -1



How would you classify this data?

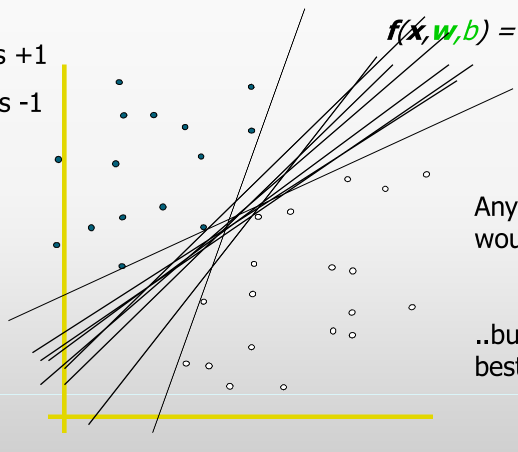
Introduction

-- SVM : a margin optimized classifier

- Linear classifier

$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\ \mathbf{x} + b)$

- denotes +1
- denotes -1



Any of these would be fine..

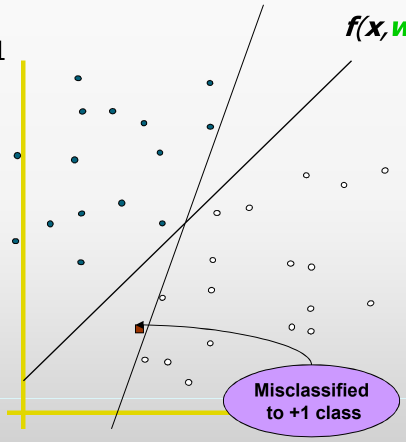
..but which is best?

Introduction
-- SVM : a margin optimized classifier

- Linear classifier

• denotes +1

◦ denotes -1



f(**x**,**w**,*b*) = *sign*(**w x** + *b*)

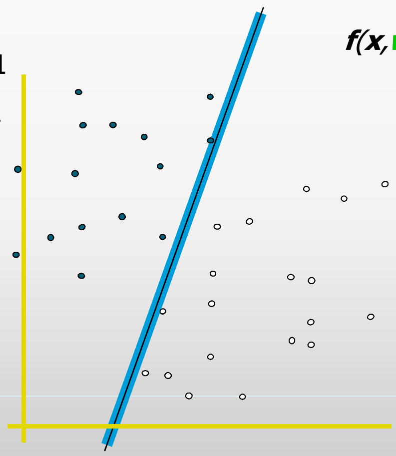
How would you classify this data?

Introduction
-- SVM : a margin optimized classifier

- Linear classifier

• denotes +1

◦ denotes -1



f(**x**,**w**,*b*) = *sign*(**w x** + *b*)

Define the **margin** of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

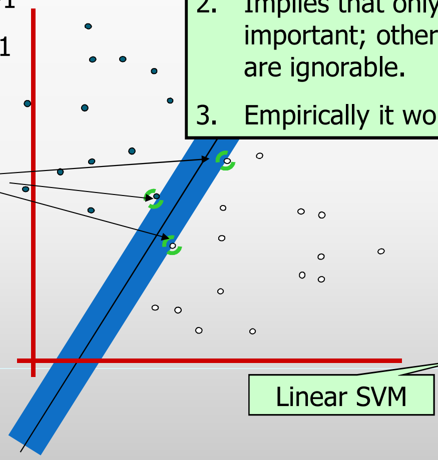
Introduction

-- SVM : a margin optimized classifier

- Linear classifier

• denotes +1

◦ denotes -1



1. Maximizing the margin is good according to intuition and PAC theory
2. Implies that only support vectors are important; other training examples are ignorable.
3. Empirically it works very very well.

Support Vectors are those datapoints that the margin pushes up against

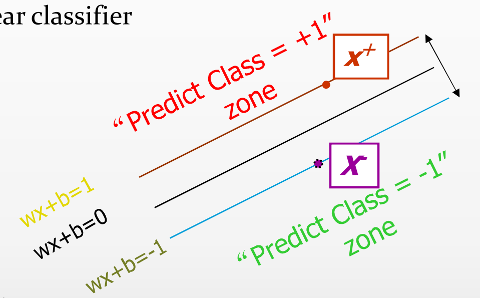
classifier is the linear classifier with the maximum margin.

This is the simplest kind of SVM (Called an LSVM)

SVM Formulation

-- Definition of margin

- Linear classifier



M=Margin Width

What we know:

- $\mathbf{w} \cdot \mathbf{x}^+ + b = +1$
- $\mathbf{w} \cdot \mathbf{x}^- + b = -1$
- $\mathbf{w} \cdot (\mathbf{x}^+ - \mathbf{x}^-) = 2$

$$M = \frac{(x^+ - x^-) \cdot w}{|w|} = \frac{2}{|w|}$$

SVM Formulation
-- Maximizing the margin

- Goal: 1) **Correctly classify all training data**

$$wx_i + b \geq 1 \qquad \textit{if } y_i = +1$$
$$wx_i + b \leq -1 \qquad \textit{if } y_i = -1$$
$$y_i(wx_i + b) \geq 1 \qquad \text{for all i}$$

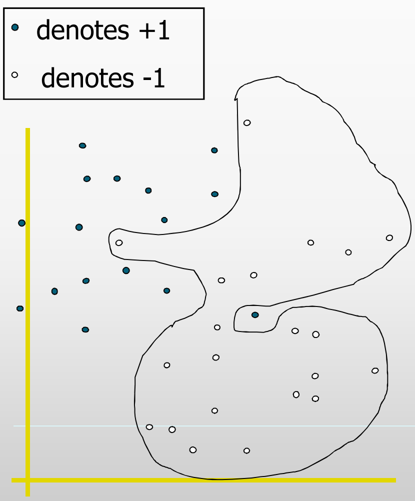
2) Maximize the Margin $M = \frac{2}{|w|}$

same as minimize $\frac{1}{2} w^t w$

- **We can formulate a Quadratic Optimization Problem and solve for w and b**
-

Minimize	$\Phi(w) = \frac{1}{2} w^t w$
subject to	$y_i(wx_i + b) \geq 1 \quad \forall i$

SVM Formulation
-- Problem: overfitting in noisy case



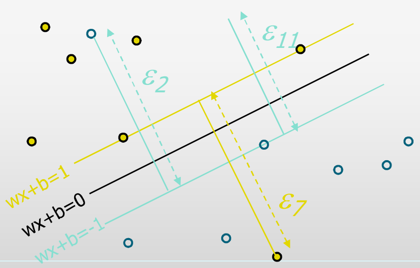
- **Hard Margin:** **So far we require all data points be classified correctly**
  - **No training error**
- **What if the training set is noisy?**
  - **Solution 1: use very powerful kernels**

OVERFITTING!

SVM Formulation
-- Introducing a soft margin

- Non-separable classifier

Slack variables **ξ_i can be added to allow misclassification of difficult or noisy examples.**



What should our quadratic optimization criterion be?

Minimize

$$\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k$$

SVM Formulation
-- Maximizing the soft margin

- Hard Margin vs. Soft Margin
  - **The old formulation:**

    Find $\mathbf{w}$ and b such that
    $\mathbf{\Phi}(\mathbf{w}) = ½\, \mathbf{w}^{\mathrm{T}}\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
    $y_i\,(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + \mathrm{b}) \geq 1$

  - **The new formulation incorporating slack variables:**

    Find $\mathbf{w}$ and b such that
    $\mathbf{\Phi}(\mathbf{w}) = ½\, \mathbf{w}^{\mathrm{T}}\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
    $y_i\,(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

  - **Parameter C can be viewed as a way to control overfitting.**

SVM Formulation
-- Maximizing the soft margin (cont'd)

The following Lagrangian should be considered

$$L(w,b,\xi;\ \alpha,\nu) = \frac{1}{2}w^T w + C\sum_{K=1}^{N}\xi_k - \sum_{k=1}^{N}\alpha_k(y_k[w^T x_k + b] - 1 + \xi_k) + \sum_{k=1}^{N}\nu_k\xi_k$$

where $\alpha_k \geq 0,\ \nu_k \geq 0$ for $k = 1,\ldots,N$

- The solution is given by the saddle point of the Lagrangian

$$\max_{\alpha,\nu}\min_{w,b,\xi}(L(w,b,\xi;\alpha,\nu))$$

$$\begin{cases} \frac{\partial L}{\partial w} = 0 \rightarrow w = \sum_{K=1}^{N}\alpha_k y_k x_k \\ \frac{\partial L}{\partial b} = 0 \rightarrow \sum_{k=1}^{N}\alpha_k y_k = 0 \\ \frac{\partial L}{\partial \xi} = 0 \rightarrow 0 \leq \alpha_k \leq C, k = 1,\ldots,N \end{cases}$$

SVM Formulation
-- Maximizing the soft margin (dual form)

- Linear SVMs: Overview
  - **The classifier is a *separating hyperplane.***
  - **Quadratic optimization algorithms can identify which training points x_i are support vectors with non-zero Lagrangian multipliers α_i.**

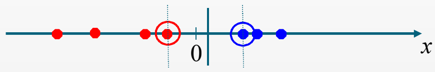
Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - ½\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

SVM
An margin optimized
linear classifier

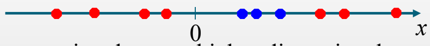
$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$

SVM Extension
-- classification on different feature spaces

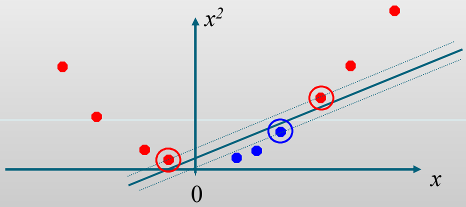
- Nonlinear classifier
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

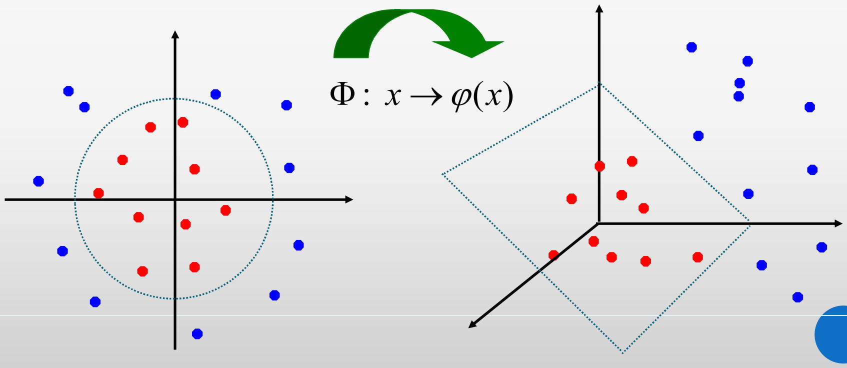


- How about… mapping data to a higher-dimensional space:



SVM Extension
-- Mapping to higher-dimensional feature space

- Nonlinear classifier
- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



SVM Extension: nonlinear classifier -- Kernel tricky

- **Find $\alpha_1 \ldots \alpha_N$ such that**

$$\max_{\alpha_1 \ldots \alpha_N} Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$\text{subject to} \quad (1) \sum_{i=1}^{N} \alpha_i y_i = 0 \,, \qquad (2)\; 0 \le \alpha_i \le C \;\; \text{for} \;\; \forall \alpha_i$$

- The kernel function

$$K\left(x_i, x_j\right) = \varphi\left(x_i\right)^T \varphi\left(x_j\right)$$

- ***Decision value*** is

$$v = \sum_{i=1}^{N} \alpha_i y_i K\left(x, x_i\right) + b$$

- ***Classifier*** is $y(x) = \text{sign}(v)$

SVM classifier -- RBF network with optimized margin