OENG1116 – Modelling and Simulation of Engineering System

Introduction to Support Vector Machine

(SVM)

Course Lecturer:

Dr Hamid Khayyam
Office: 251-02-34
Phone: 03 9925 4630
Email: hamid.khayyam@rmit.edu.au

RMIT UNIVERSITY

Carbon Nexus Industry (example)

Dr Hamid Khayyam (PhD, SMIEEE)
Research Fellow and Team Leader at Carbon Nexus (2013-2016)
Senior Lecturer at RMIT University (2017-now)

Carbon Nexus Industry (questions)

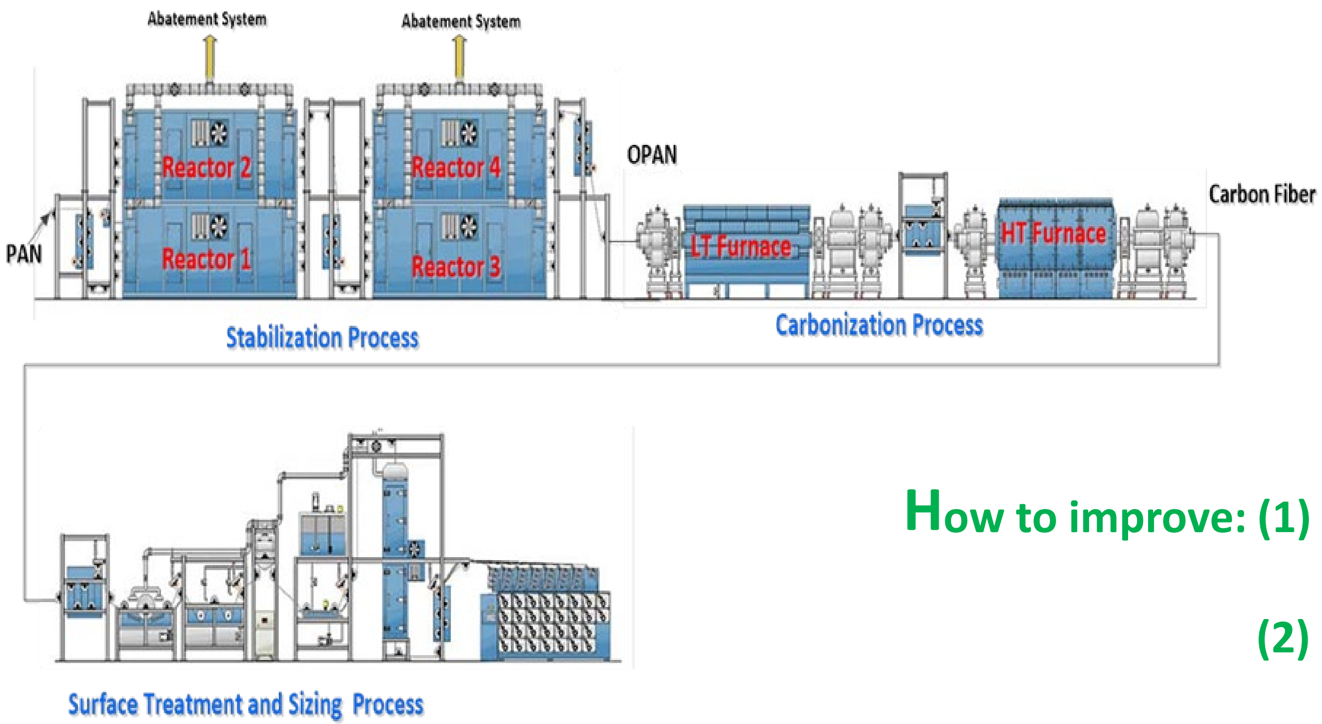


Figure: Industrial carbon fiber manufacturing process

How to improve: (1) Process line productivity

(2) Energy efficiency

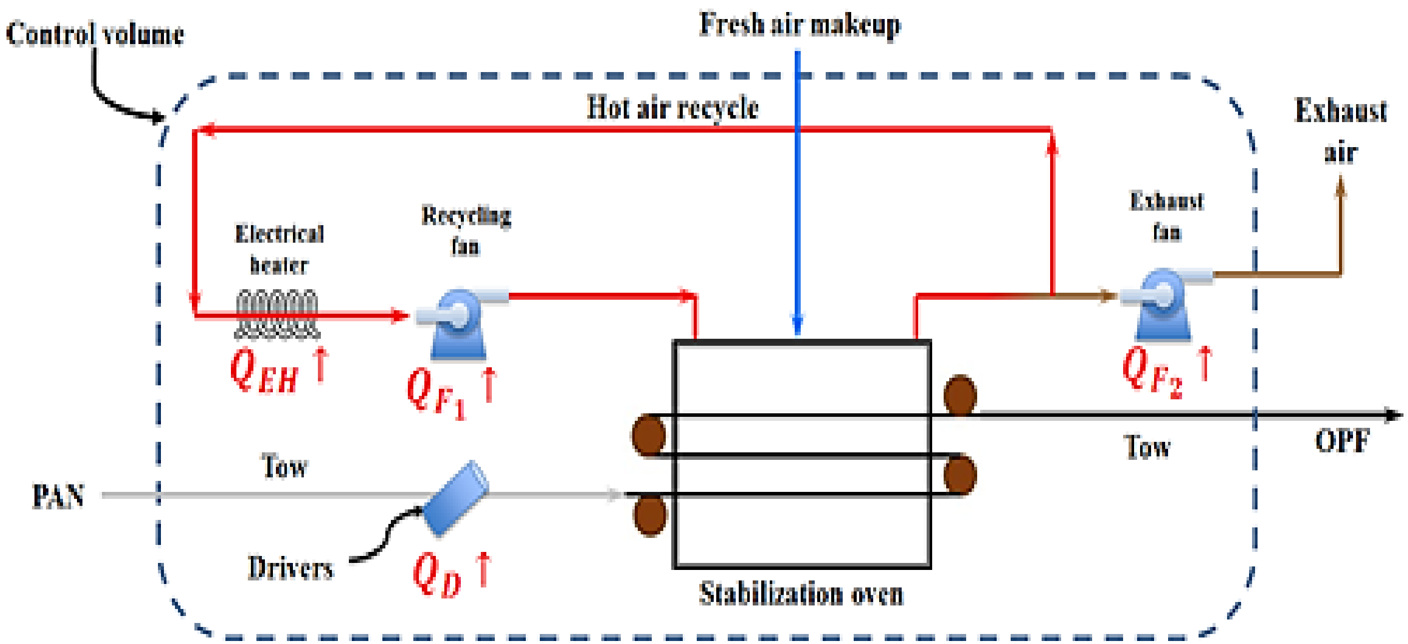
(3) Mechanical properties

(4) Chemical properties

(5) Physical properties

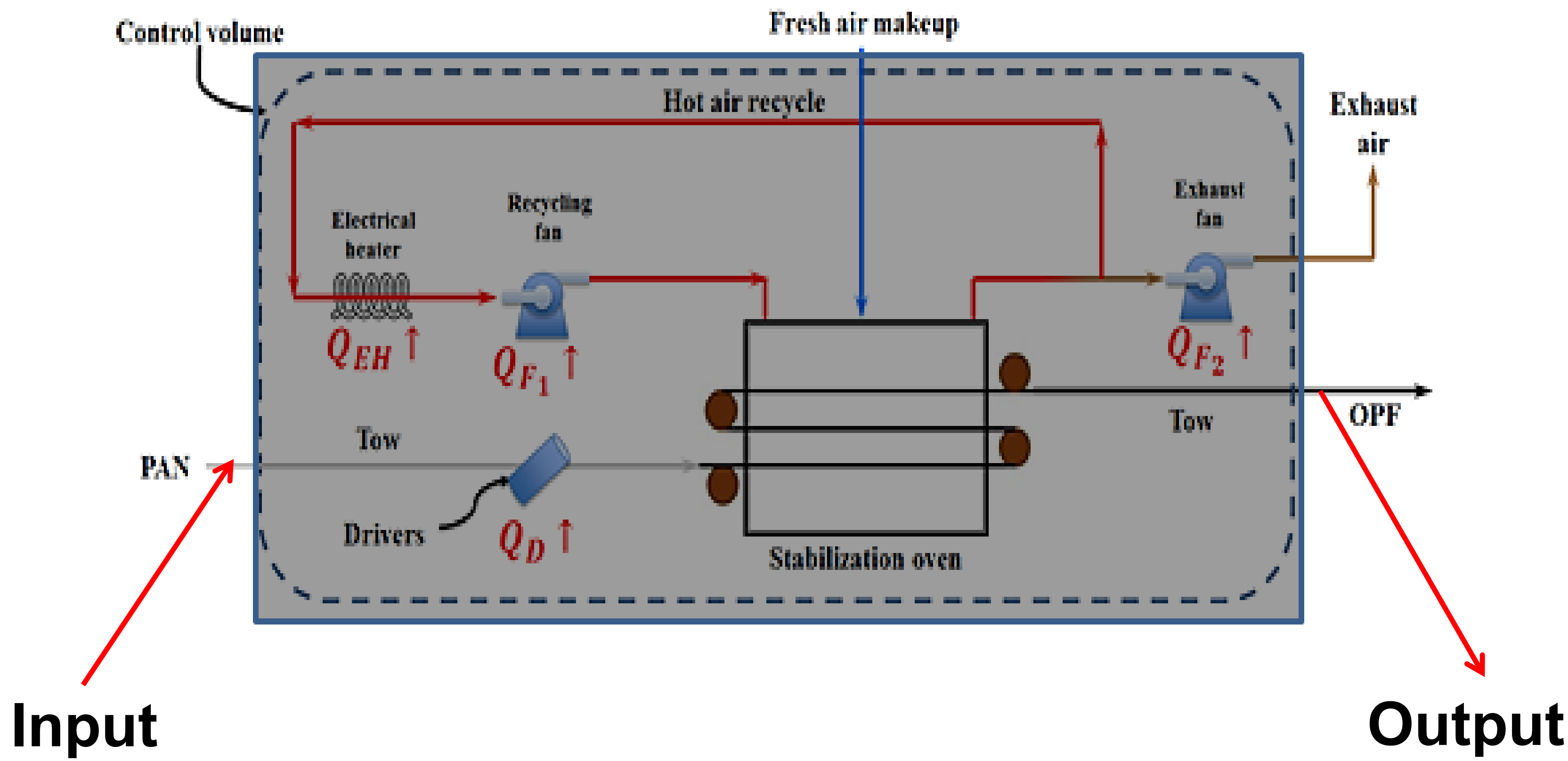
……….

Outline



Fourteen parameters are involved in thermal stabilisation process of carbon fibres production such as: (i) zone set temperature, (ii) circulation air velocity, (iii) catenary, (iv) exhaust off take, (v) fresh air (O2) input, (vi) number of tow, (vii) filament count of each tow, (viii) fibre energy release, (ix) end slot gap height (where fibre exits the oven), (x) composition of PAN precursor, (xi) size consistency, (xii) fibre dwell time required in heated length, (xiii) oven end sealing (to stop fugitive gas emissions), and (ixv) concentration of released gasses inside the process chamber.

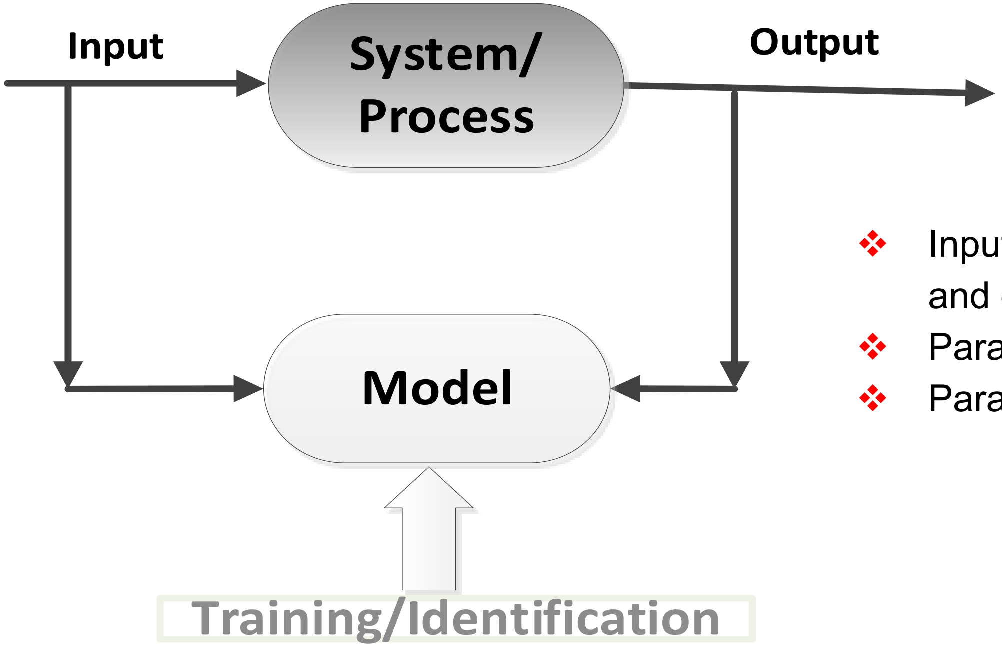
Outline



Control volume
Fresh air makeup
Hot air recycle
Exhaust air
Electrical heater
Recycling fan
Exhaust fan
Q_{EH} ↑
Q_{F_1} ↑
Q_{F_2} ↑
Tow
OPF
PAN
Tow
Drivers
Q_D ↑
Stabilization oven
Input
Output

Data Modelling and Machine Learning

- **Given inputs and outputs and find actual data**



- Input/output data from instrument measurements and expert perceptions
- Parametric model structure
- Parameter estimation

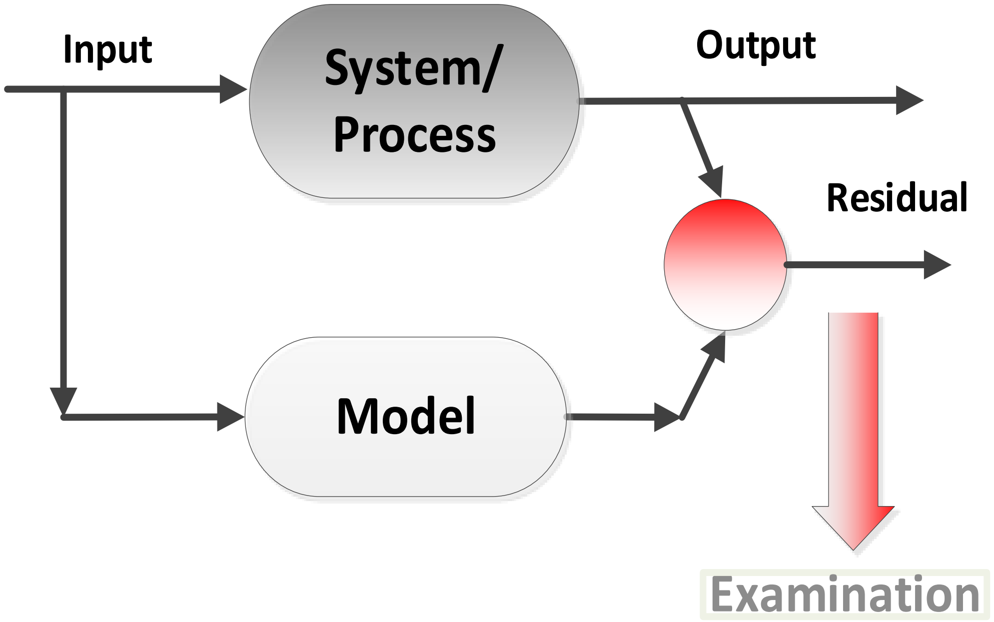
Outline

Excel File

Sample	Temper,, °C	Space-velocity, m/h	Stretching Ratio, %	Energy consump. J
1	227.0347	34.9999	1.6563	4366028.7
2	227.0479	34.9204	1.8843	4376758.0
3	227.0303	34.6017	1.9690	4416004.9
4	227.0207	34.4468	1.9974	4435292.7
5	227.0354	34.9320	1.9050	4374564.9
6	227.0250	34.5050	1.7947	4428064.0
7	227.0347	34.9999	1.6563	4366028.7
8	227.0363	34.7120	1.7976	4402336.3
9	227.0241	34.7599	1.97815	4395544.3
10	227.0328	34.6569	1.8461	4409123.6
11	227.0424	34.5091	1.9327	4428593.5
12	227.0210	34.5647	1.9696	4420182.8
13	227.0353	34.7785	1.9201	4393862.1
14	227.0358	34.94032	1.8036	4373539.9
15	227.0337	34.6926	1.8699	4404642.8
16	227.0267	34.6648	1.9086	4407758.5
17	227.0267	34.6249	1.9833	4412830.4
18	227.0208	34.4847	1.9839	4430421.2
19	227.0436	34.9009	1.9325	4378944.0
20	227.0308	34.8311	1.8919	4386954.5
21	227.0435	34.8821	1.9259	4381295.8
22	227.0348	34.9883	1.8714	4367492.5
23	227.0269	34.8230	1.9942	4387739.9

Data Modelling and Machine Learning

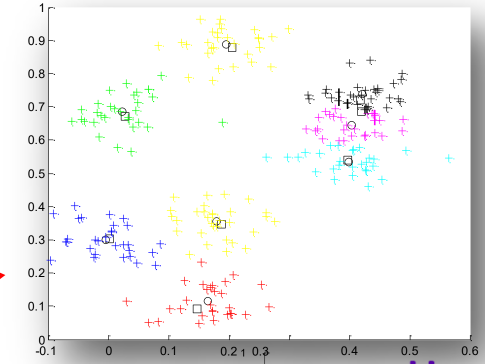
Validation

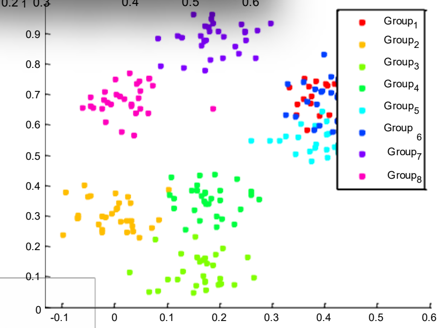


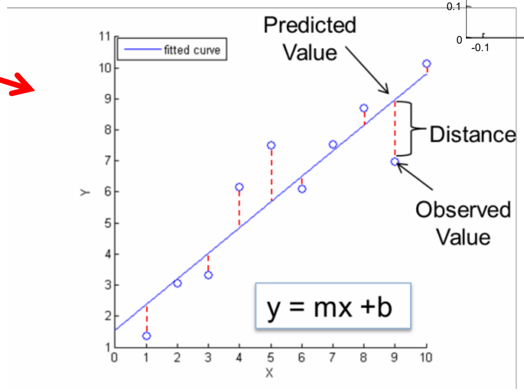
- Examine residuals
- Correlation tests
- A valid model's residuals should be reduced to uncorrelated sequence with zero mean and finite variance

Machine Learning Overview

- Machine Learning with MATLAB
  - Unsupervised Learning
    - Clustering
  - Supervised Learning
    - Classification
    - Regression
- Learn More







[3] Build Machine Learning Models with a MATLAB Trial

Outline

- What is SVM?
- The Optimization Problem
- The Kernel Trick
- Steps in SVM Modelling
- Support Vector Machine - Regression (SVR)
- References.

Outline

- ❑ What is SVM?
- ❑ The Optimization Problem
- ❑ The Kernel Trick
- ❑ Steps in SVM Modelling
- ❑ Support Vector Machine - Regression (SVR)
- ❑ References.

Support Vector Machines (SVM)

→ Support Vector Machine (SVM) is a machine learning method that is widely used for data analyzing and pattern recognizing.

→ The algorithm was invented by Vladimir Vapnik and the current standard appearance was proposed by Corinna Cortes and Vladimir Vapnik.

→ This application note is to helping understand the concept of support vector machine and how to build a simple support vector machine using Matlab

Support Vector Machines (SVM)

→ Supervised learning methods for classification and regression

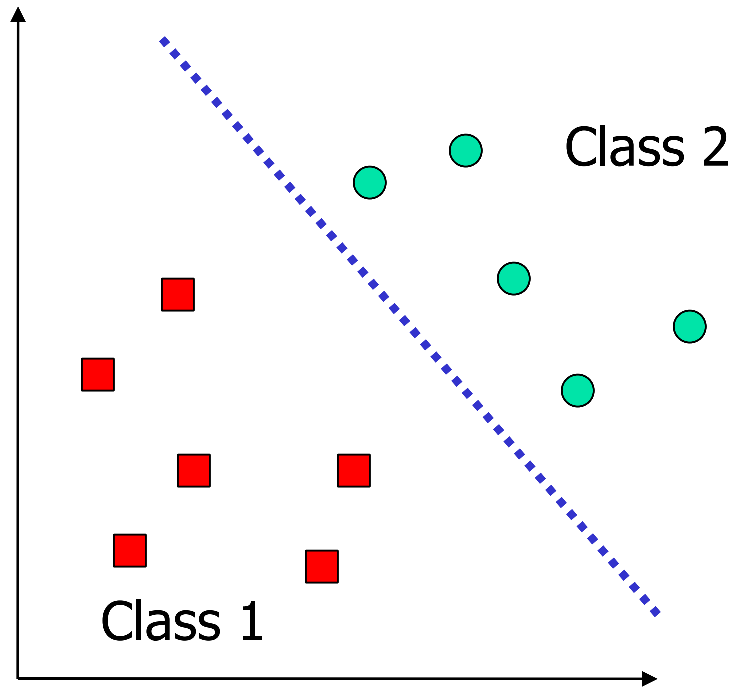
relatively new class of successful learning methods -

→ They can represent non-linear functions and they have an efficient training algorithm

→ SVM got into mainstream because of their exceptional performance in Handwritten Digit Recognition

- 1.1% error rate which was comparable to a very carefully constructed (and complex) ANN

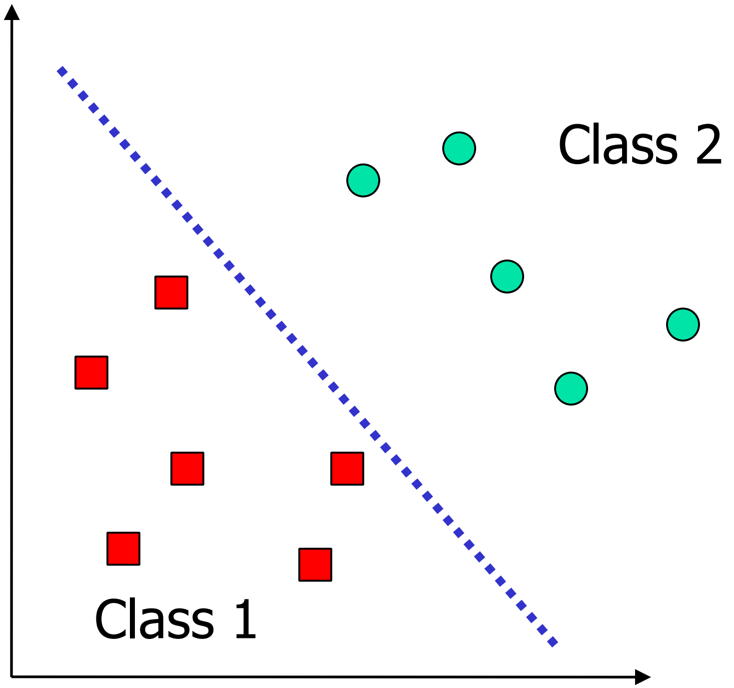
Two Class Problem: Linear Separable Case

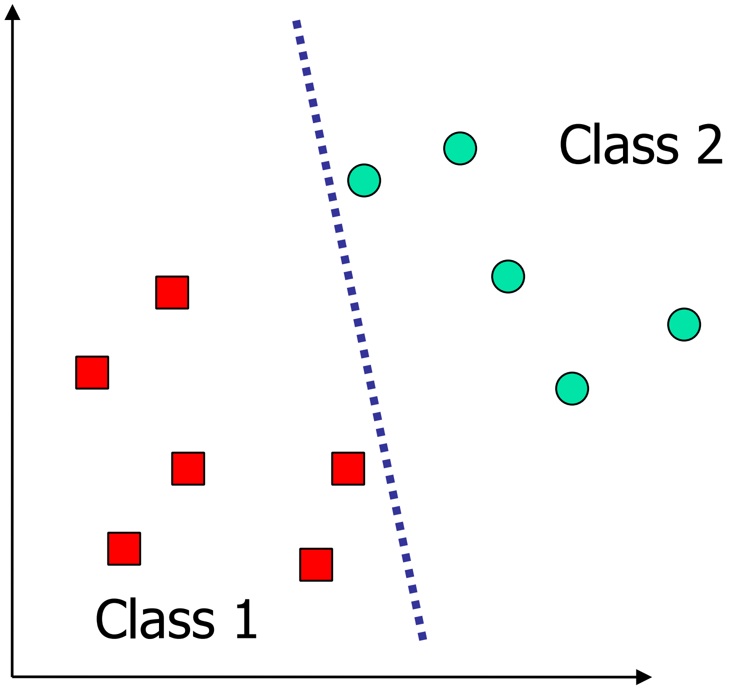


Many decision boundaries can separate these two classes.

Which one should we choose?

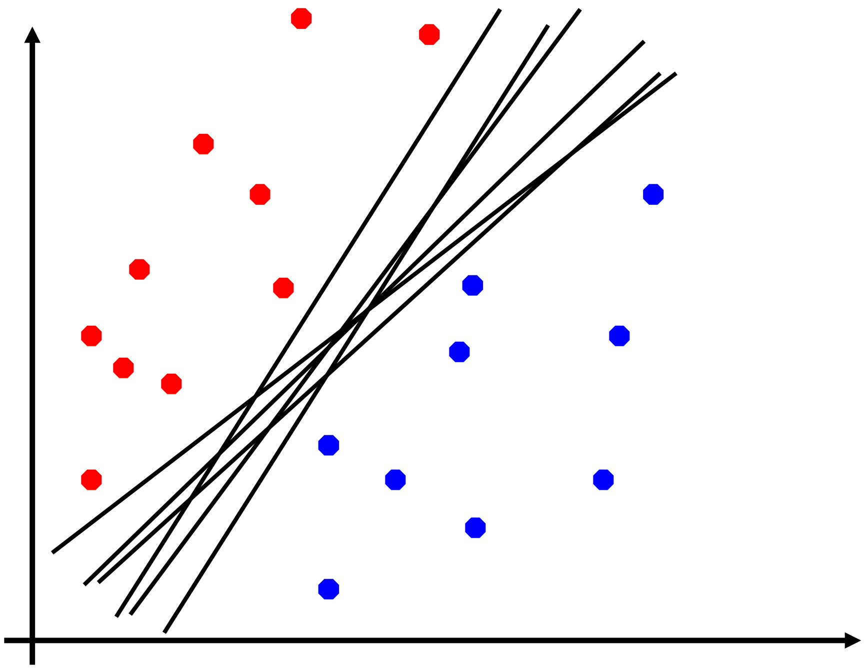
Example of Bad Decision Boundaries





Linear Separators

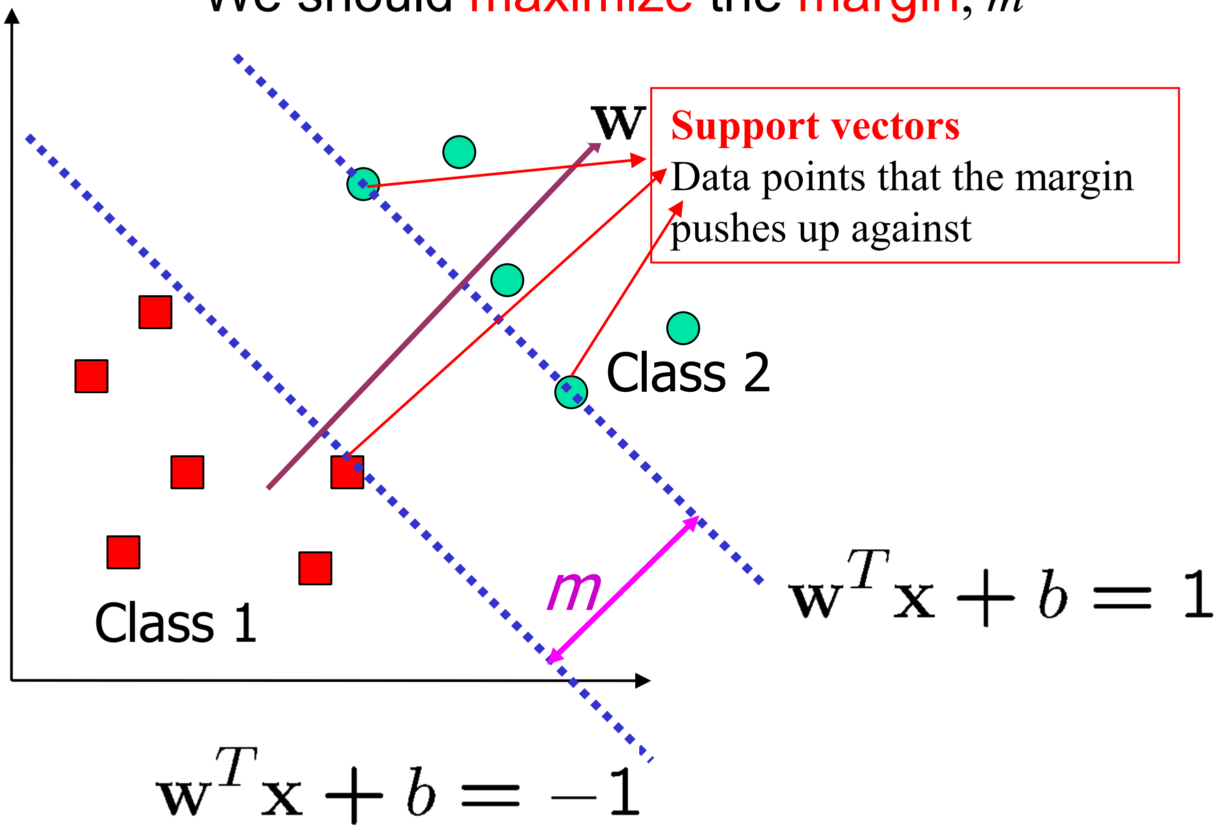
- Which of the linear separators is optimal?



Good Decision Boundary: Margin Should Be Large

The decision boundary should be as far away from the data of both classes as possible

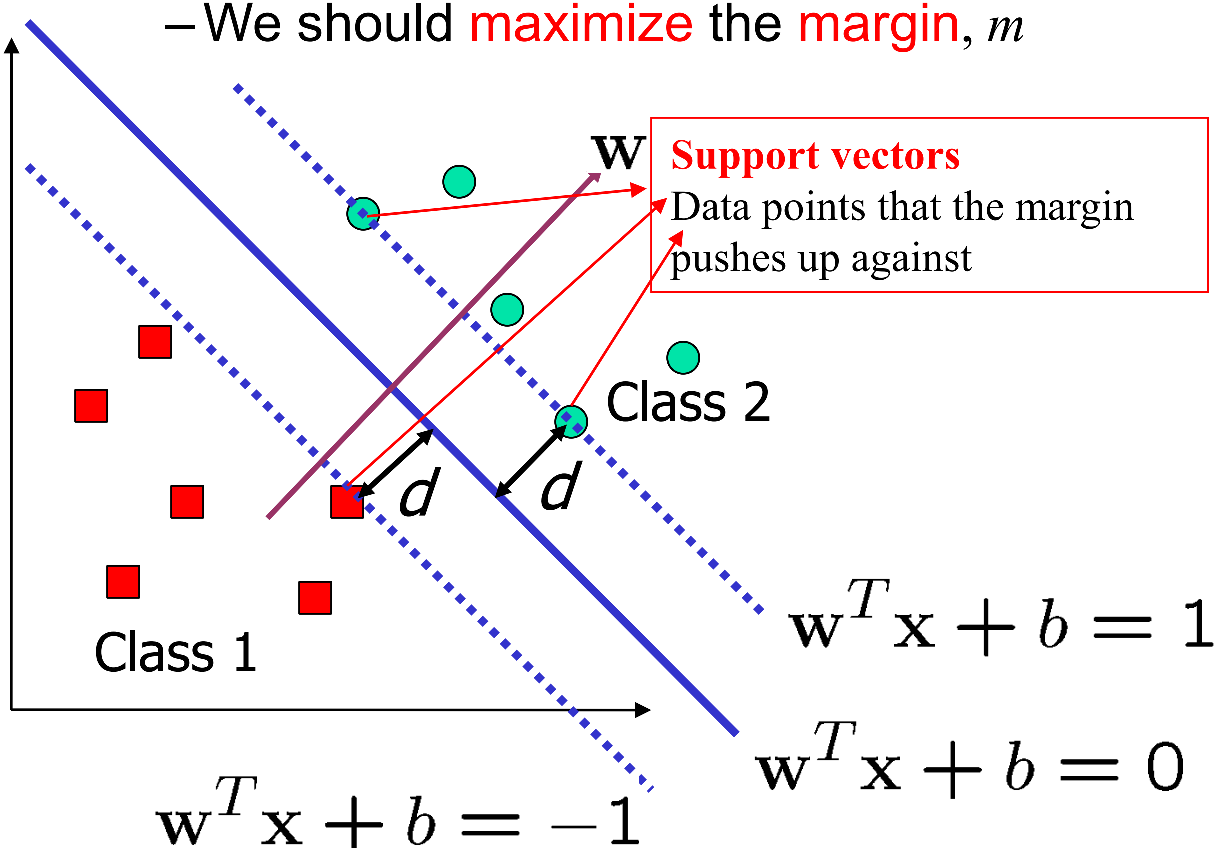
– We should maximize the margin, m



Good Decision Boundary: Margin Should Be Large

The decision boundary should be as far away from the data of both classes as possible

– We should maximize the margin, m



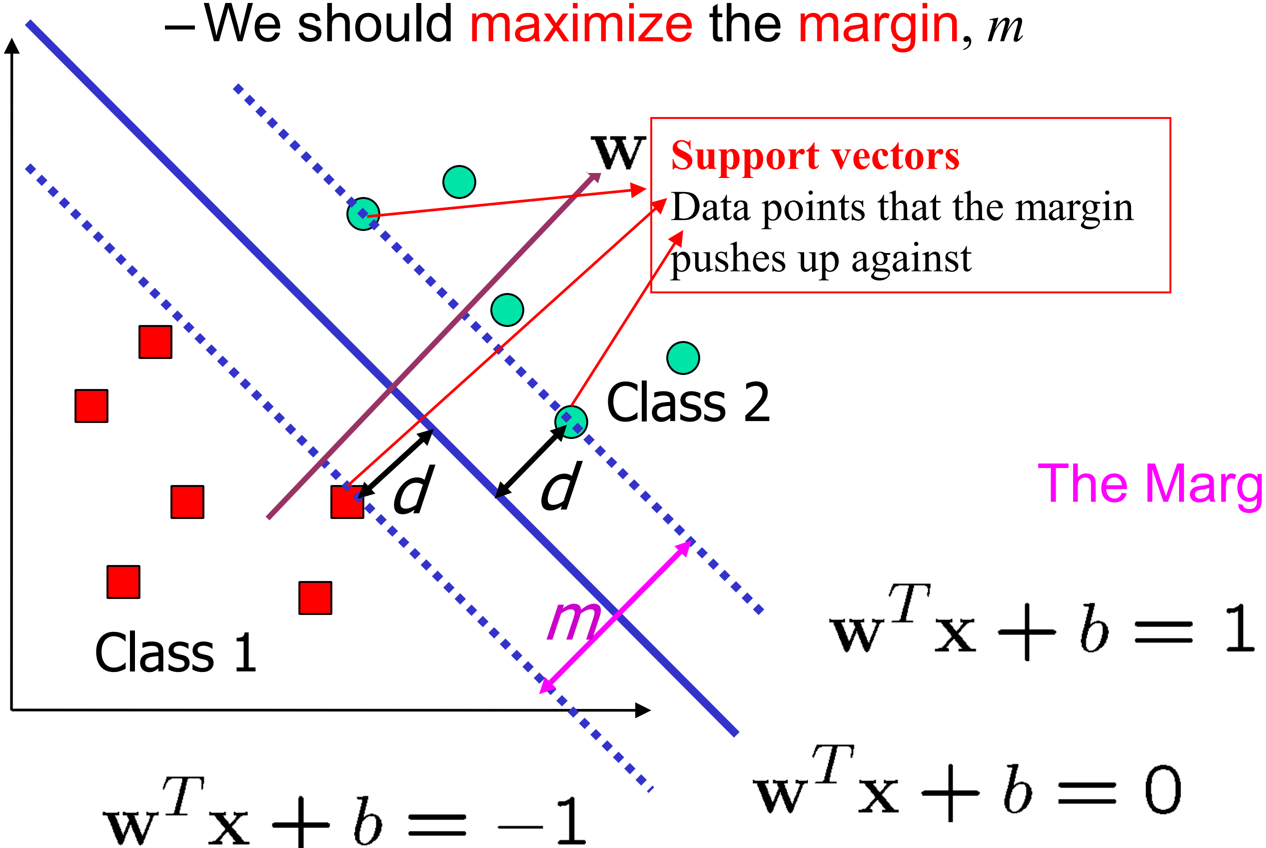
$$d = \frac{\left\| w^t x' + b \right\|}{\|w\|} = \frac{1}{\|w\|}$$

Note: $\|\mathbf{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}.$

Good Decision Boundary: Margin Should Be Large

The decision boundary should be as far away from the data of both classes as possible

– We should maximize the margin, m



$$d = \frac{\left\| w^t x' + b \right\|}{\|w\|} = \frac{1}{\|w\|}$$

The Margin is $\boxed{m = \frac{2}{\|w\|}}$

Note: $\|\mathbf{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}.$

Outline

- What is SVM?
- The Optimization Problem
- The Kernel Trick
- Steps in SVM Modelling
- Support Vector Machine - Regression (SVR)
- References.

The Optimization Problem 1

Let $\{x_1, ..., x_n\}$ be our data set and let $y_i \in \{1,-1\}$ be the class label of x_i

The decision boundary should classify all points correctly $\Rightarrow$

A constrained optimization problem:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \qquad \forall i$$

$$m = \frac{2}{||\mathbf{w}||}$$

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

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

Lagrangian of Original Problem

Problem: Minimize $\frac{1}{2}||\mathbf{w}||^2$

subject to $1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) \leq 0$ for $i = 1, \ldots, n$

Solution:

The Lagrangian:

Lagrangian multipliers

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right)$$

– Note that $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$

Setting the gradient of $\mathcal{L}$ w.r.t. ***w*** and *b* to zero, we have

$$\mathbf{w} + \sum_{i=1}^{n} \alpha_i(-y_i)\mathbf{x}_i = \mathbf{0} \quad \Rightarrow \quad \boxed{\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i}$$

$$\boxed{\sum_{i=1}^{n} \alpha_i y_i = 0} \qquad \alpha_i \geq 0$$

The Dual Optimization Problem

We can transform the problem to its dual

Dot product of X

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

α's → New variables
(Lagrangian multipliers)

This is a convex quadratic programming (QP) problem

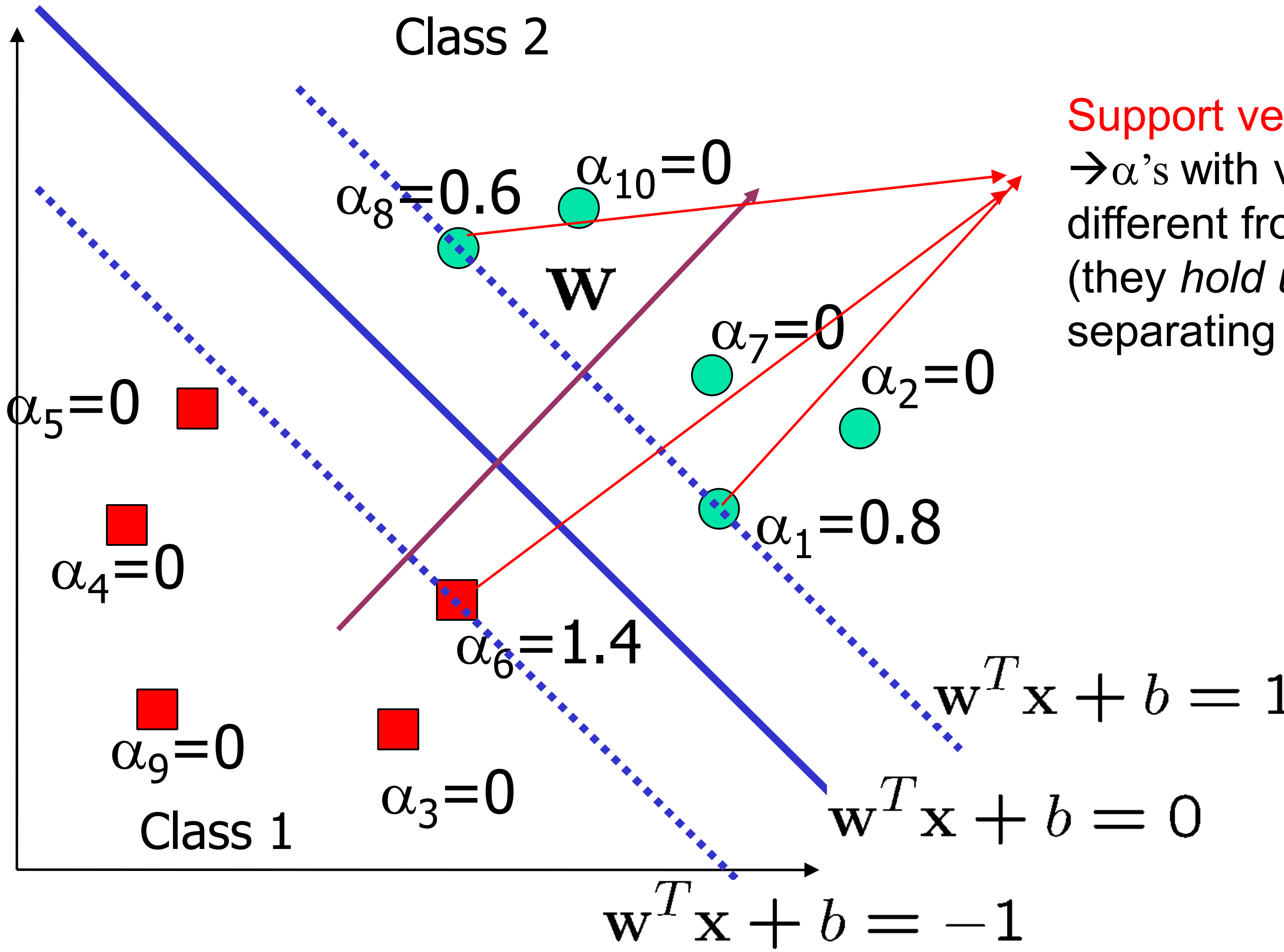
– Global maximum of α_i can always be found

→well established tools for solving this optimization problem (e.g. cplex)

Note:

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

A Geometrical Interpretation (hard margin)

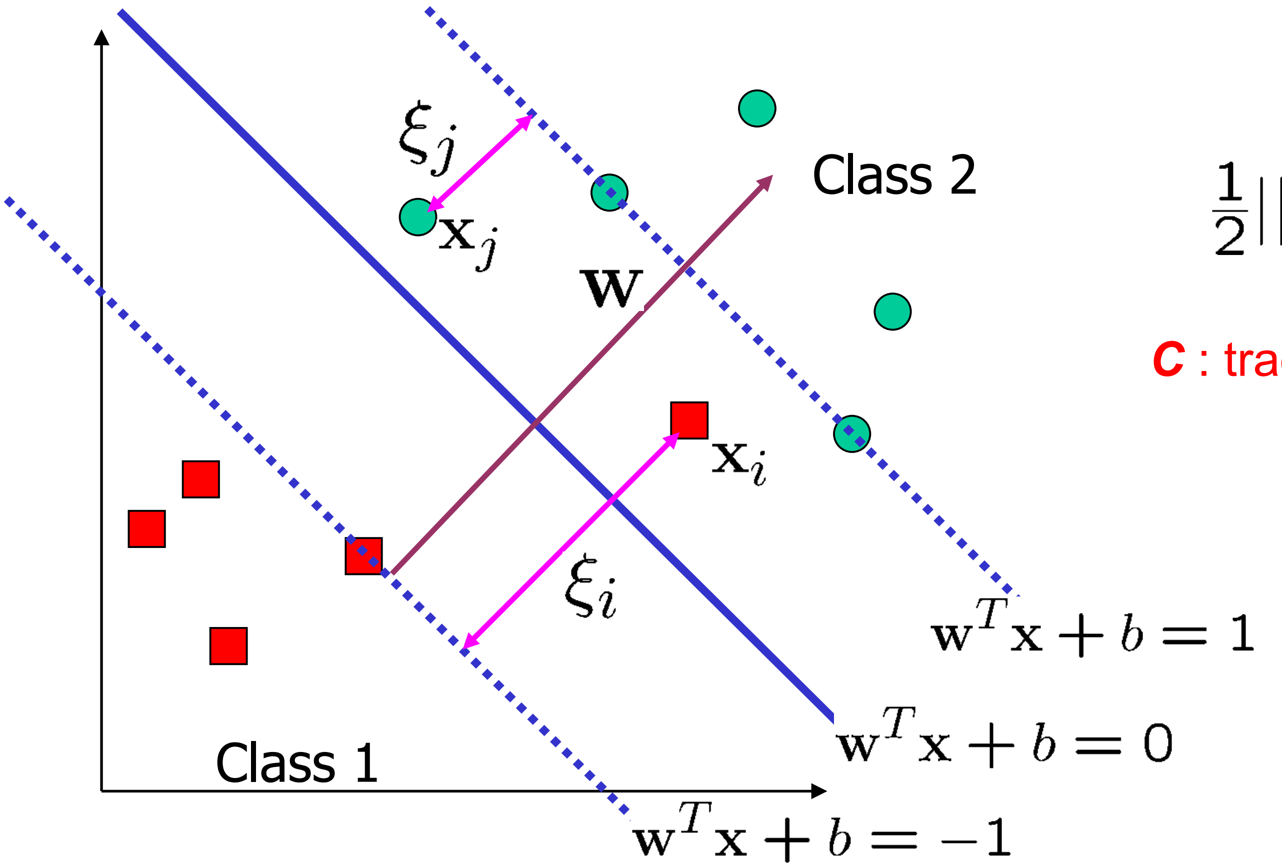


Support vectors
→ α's with values different from zero (they *hold up* the separating plane)!

Non-Linearly Separable Problems (soft margin)

We allow "error" ξ_i in classification; it is based on the output of the discriminant function $\mathbf{w}^T\mathbf{x}+b$

ξ_i approximates the number of misclassified samples



New objective function:

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

C : tradeoff parameter between error and margin; chosen by the user; large C means a higher penalty to errors

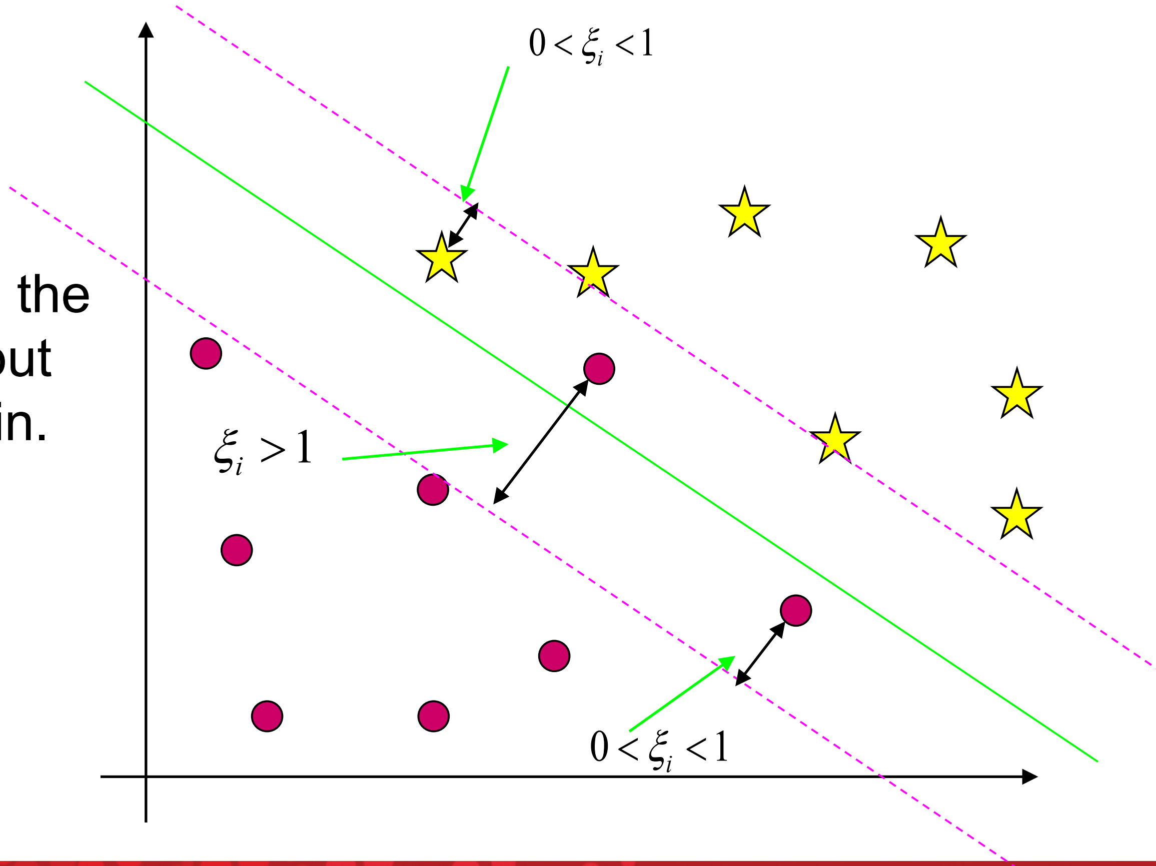
Non-Linearly Separable Problems (soft margin)

-We allow "error" ξ_i in classification. We use "slack" Variables $\boxed{\xi_1, \xi_2, \ldots\ldots \xi_n}$ (one for each sample).

ξ_i Is the deviation error from ideal place for sample *i*:

-If $0 < \xi_i < 1$ then sample *i* is on the right side of the hyperplane but within the region of the margin.

-If $\xi_i > 1$ then sample *i* is on the wrong side of the hyperplane.



The primal optimization problems (soft margin)

-We change the constrains to $y_i(w^t x_i + b) \geq 1 - \xi_i \quad \forall i \quad \xi_i \geq 0$

instead of $y_i(w^t x_i + b) \geq 1 \quad \forall i$.

Our optimization problem now is:

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i$$

Such that: $y_i(w^t x_i + b) \geq 1 - \xi_i \quad \forall i \quad \xi_i \geq 0$

$C > 0$ is a constant. It is a kind of penalty on the term $\sum_{i=1}^{n} \xi_i$

It is a tradeoff between the margin and the training error. It is a way to control overfitting along with the maximum margin approach[11].

The Dual optimization problems (soft margin)

Our dual optimization problem now is:

$$\text{maximize} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

Such that:

$$0 \le \alpha_i \le C \quad \forall i \quad and \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

-We can find "w" using : $w = \sum_{i=1}^{n} \alpha_i y_i x_i \qquad 0 < \alpha_i < C$

$$\alpha_i [y_i (w^t x_i + b) - 1] = 0$$

Which value for "C"

should we choose.

$$\alpha_i = 0 \Rightarrow y_i (w^T x_i + b) > 1$$

$$0 < \alpha_i < C \Rightarrow y_i (w^T x_i + b) = 1$$

$$\alpha_i = C \Rightarrow y_i (w^T x_i + b) < 1 \quad (\text{points } with\ \xi_i > 0)$$

The primal optimization problems (soft margin)

-Finding the “Right” value for “C” is one of the major problems of SVM:

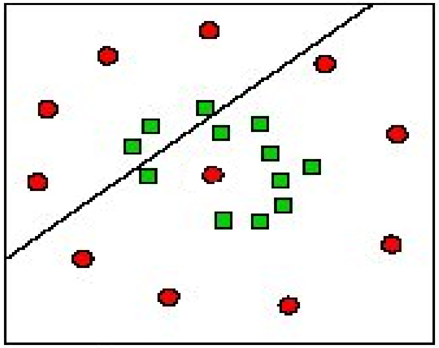
-“C” plays a major role in controlling overfitting.

-Larger C → less training samples that are not in ideal position (which means less training error that affects positively the Classification Performance (CP)) But smaller margin (affects negatively the (CP)).C large enough may lead us to overffiting (too much complicated classifier that fits only the training set)

-Smaller C → more training samples that are not in ideal position (which means more training error that affects negatively the Classification Performance (CP)) But larger Margin (good for (CP)). C small enough may lead to underffiting (naïve classifier)

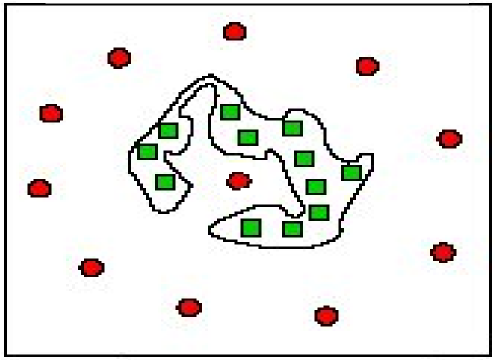
“C” Problem: Overfitting and Underfitting(soft margin)

Under-Fitting

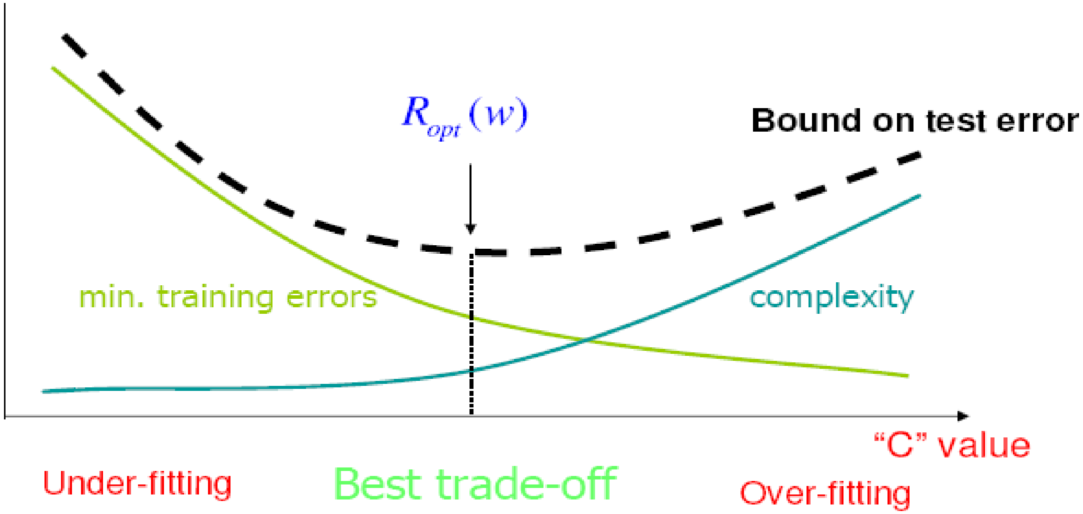


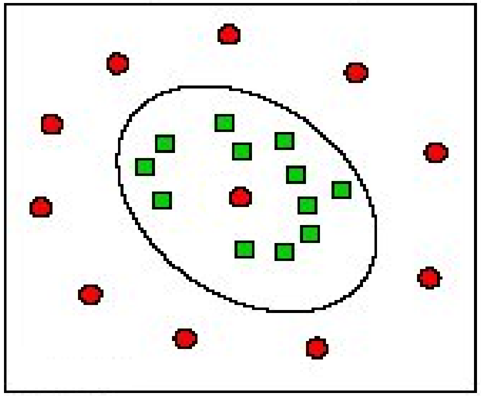
Too much simple!

Over-Fitting



Too much complicated!



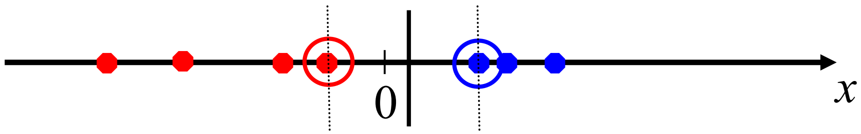


Trade-Off

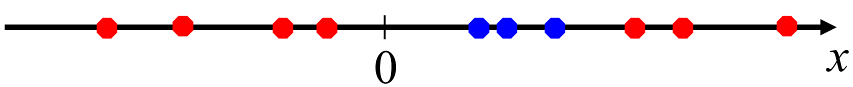
Based on [12] and [3]

Non-Linear SVMs

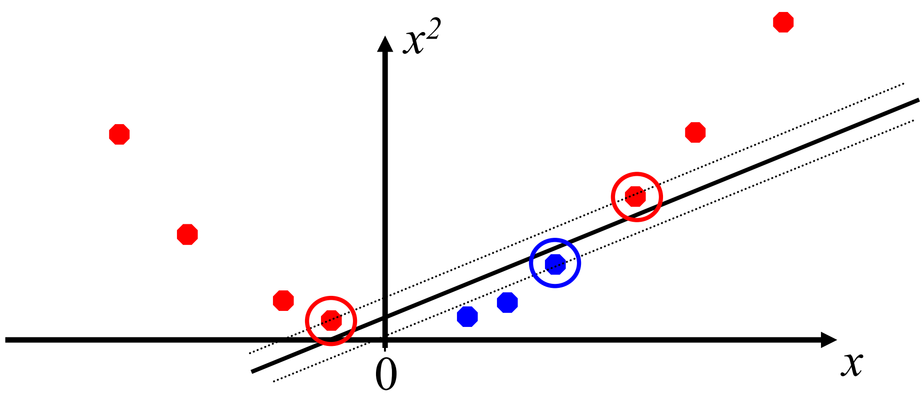
- Datasets that are linearly separable with some noise work out great:



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



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



The Optimization Problem 2

The dual of the problem is:

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \boxed{C \geq \alpha_i \geq 0,} \sum_{i=1}^{n} \alpha_i y_i = 0$$

$\mathbf{w}$ is also recovered as: $\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$

The only difference with the linear separable case is that there is an upper bound C on α_I

Once again, a QP solver can be used to find α_i efficiently!!!

Non-Linear SVM

How could we generalize this procedure to non-linear data?

Vapnik in 1992 showed that transforming input data $\mathbf{x}_i$ into a higher dimensional makes the problem easier.

Similar to Hidden Layers in ANN

- We know that data appears only as dot products $(\mathbf{x}_i.\mathbf{x}_j)$

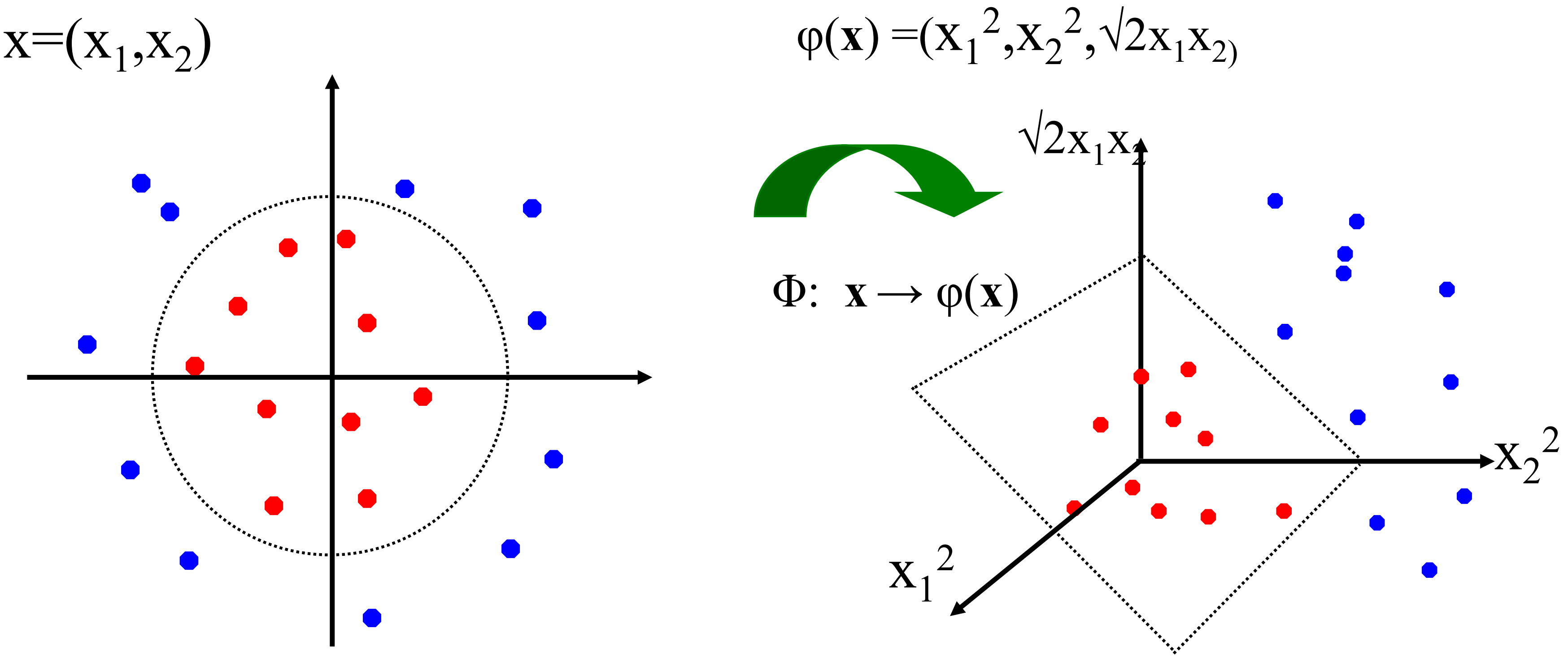
- Suppose we transform the data to some (possibly infinite dimensional) space $\mathbf{H}$ via a mapping function Φ such that the data appears of the form $\Phi(\mathbf{x}_i)\Phi(\mathbf{x}_j)$

Why?

- Linear operation in **H** is equivalent to non-linear operation in input space.

Non-linear SVMs: Feature Space

General idea: the original input space (x) can be mapped to some higher-dimensional feature space ($\varphi(\mathbf{x})$)where the training set is separable:



If data are mapped into higher a space of sufficiently high dimension, then they will in general be linearly separable; N data points are in general separable in a space of N-1 dimensions or more!!!

This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Transformation to Feature Space

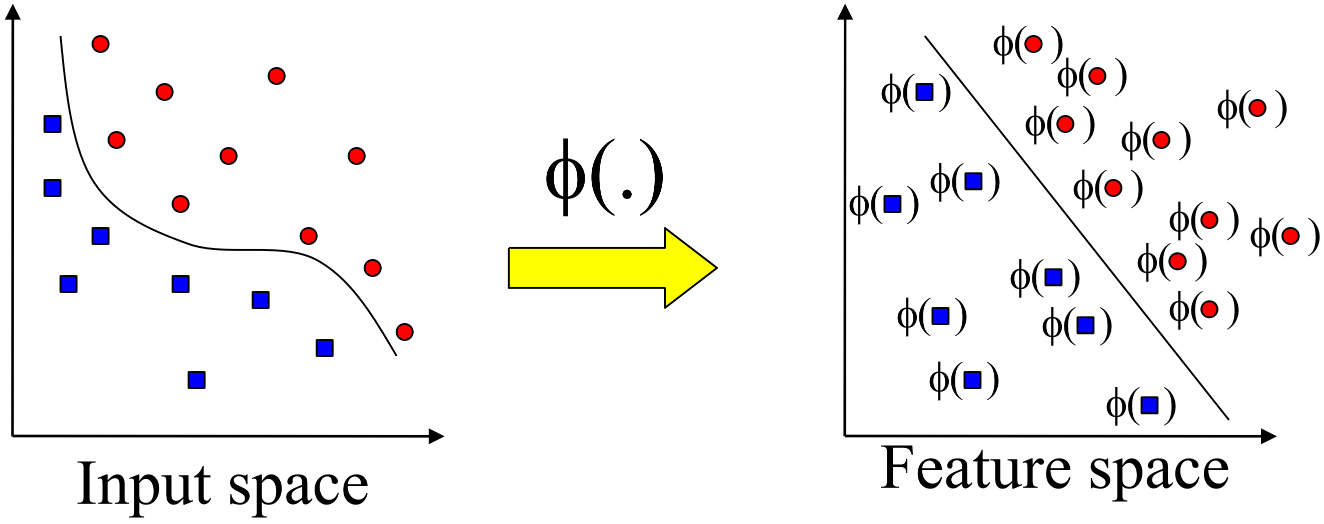
Possible problem of the transformation

– High computation load due to high-dimensionality and hard to get a good estimate

SVM solves these two issues simultaneously

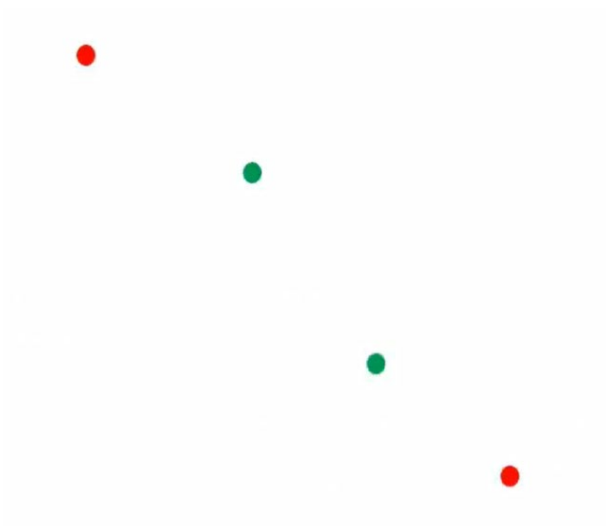
– "Kernel tricks" for efficient computation

– Minimize $||\mathbf{w}||^2$ can lead to a "good" classifier



Example 2: Non-linear SVMs, Feature Space

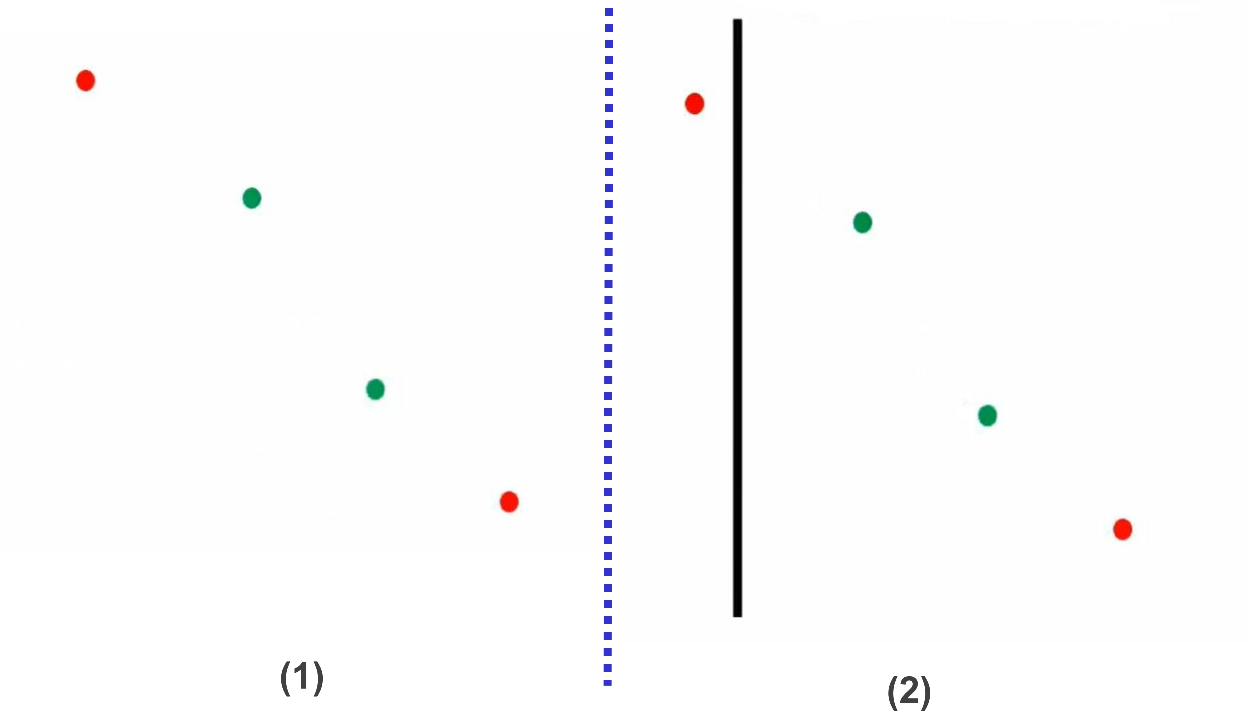
Separate the classes



(1)

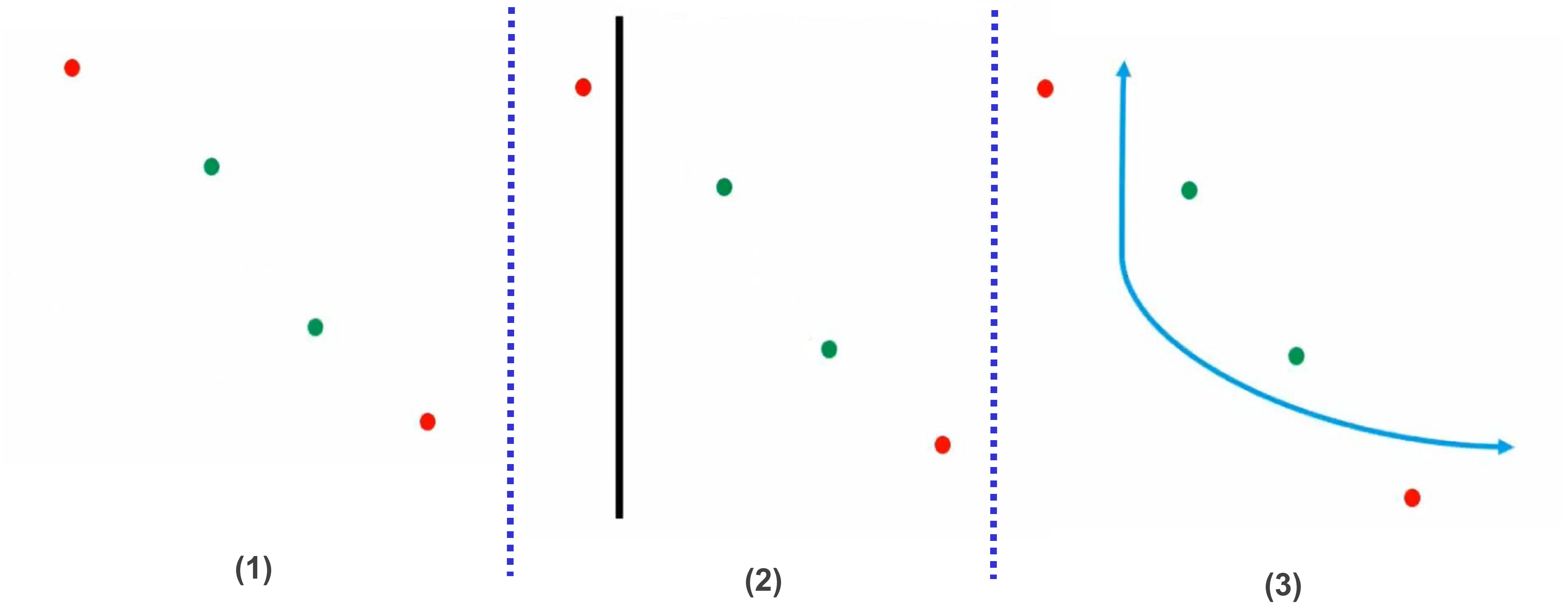
Example 2: Non-linear SVMs, Feature Space

When the line is not enough…

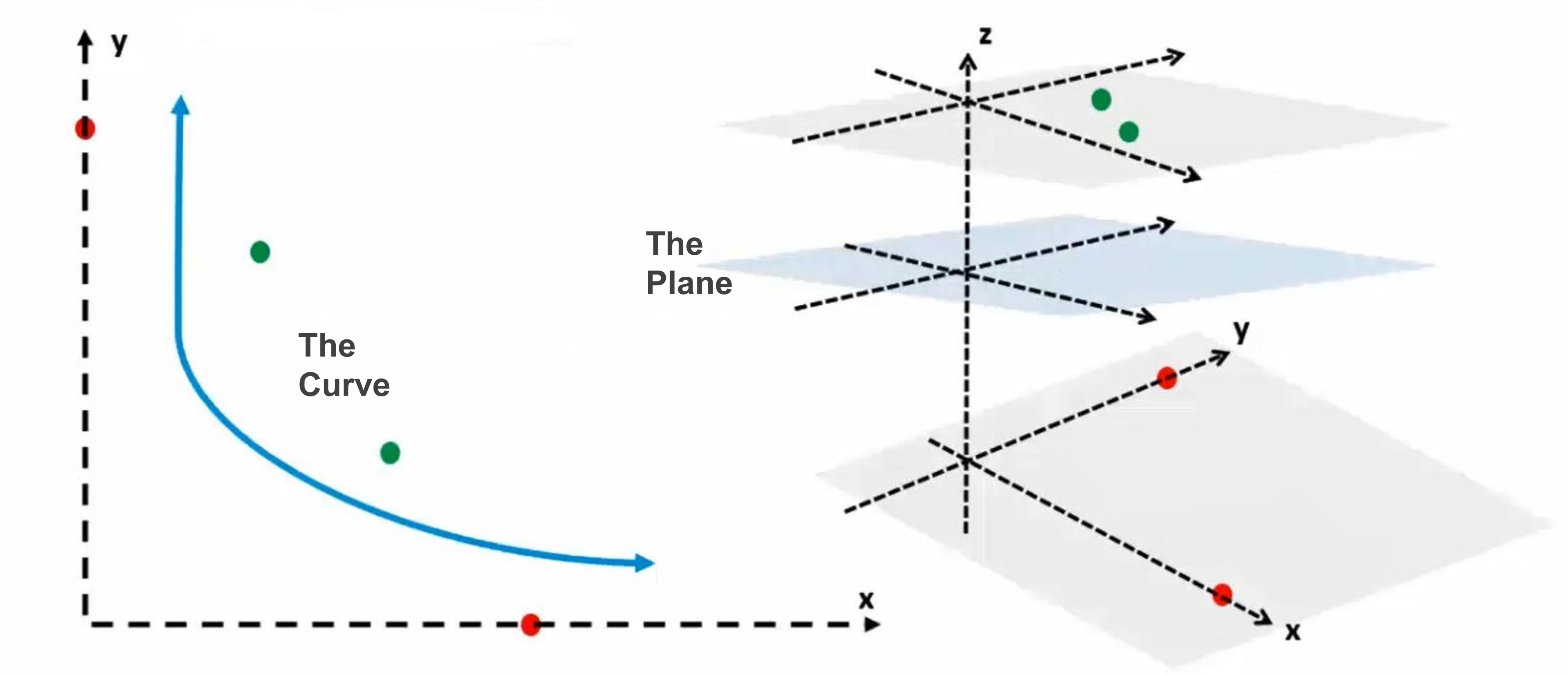


Example 2: Non-linear SVMs, Feature Space

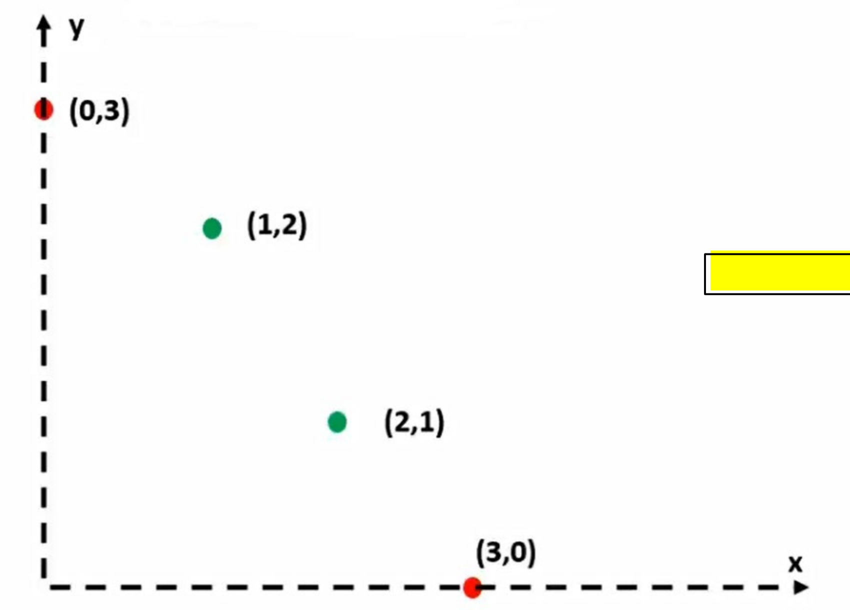
When the line is not enough…



Example 2: Non-linear SVMs, Feature Space



Example 2: Non-linear SVMs, Feature Space

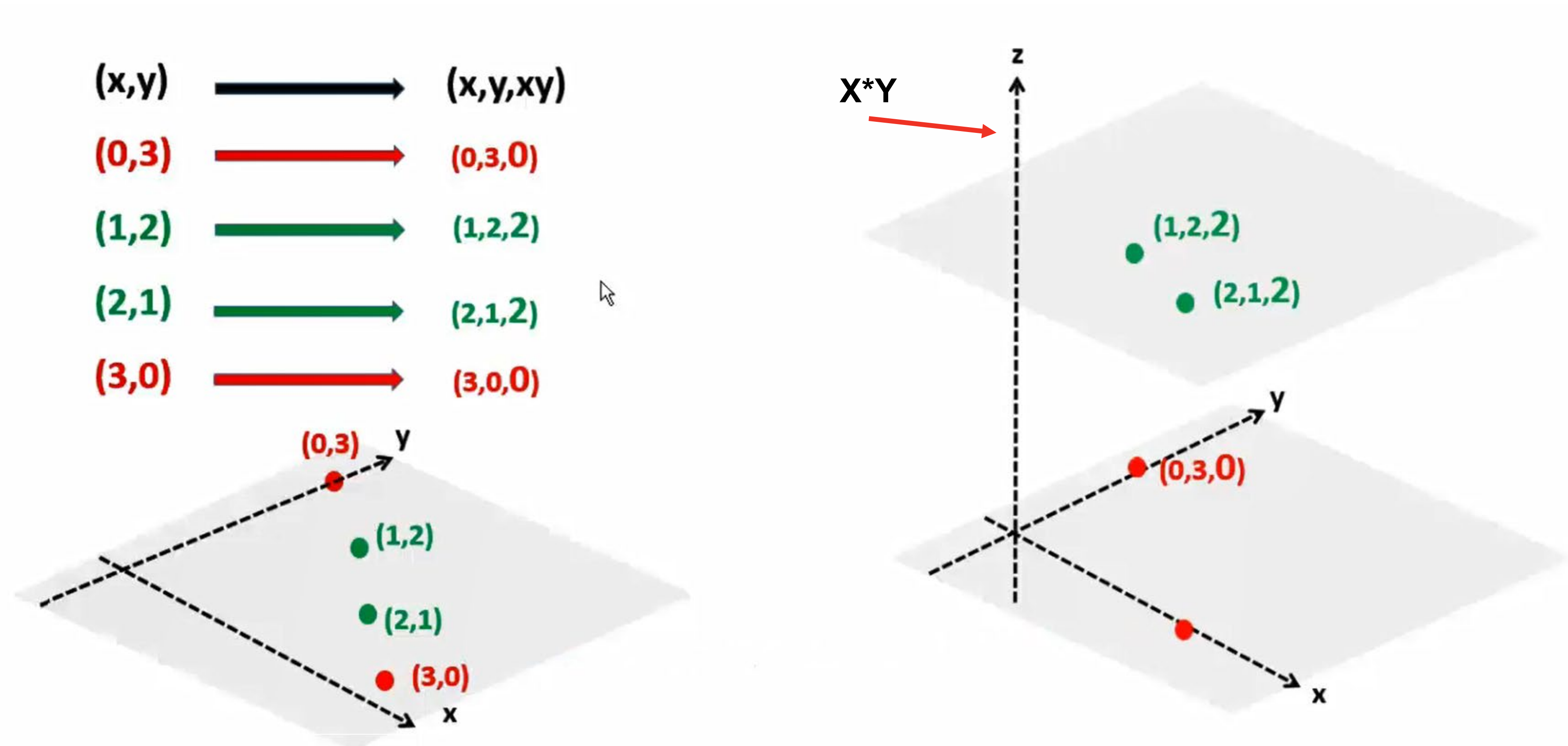


The equation that best represents the graph are:

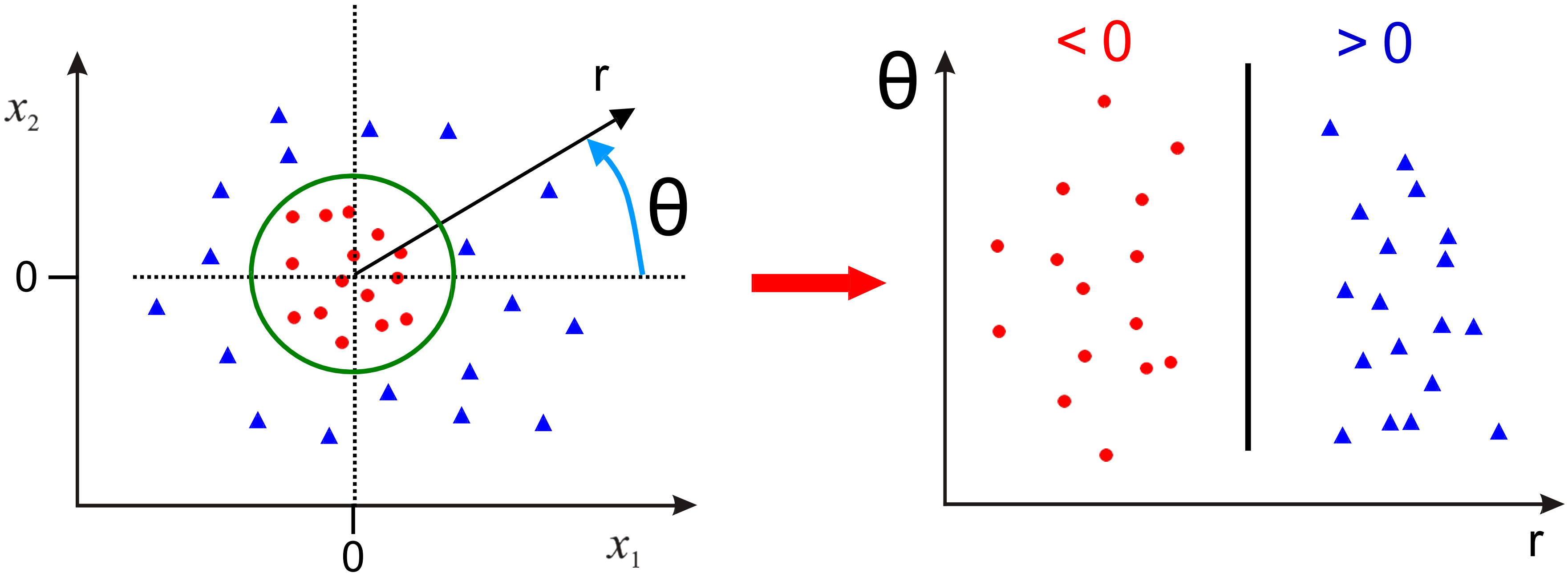
- ❑ X+Y
- ❑ X*Y
- ❑ X^2

	(0,3)	(1,2)	(2,1)	(3,0)
X+Y	3	3	3	3
X*Y	0	2	2	0
X^2	0	1	4	9

Example 2 : Non-linear SVMs, Feature Space



Example 3 : Non-linear SVMs, Feature Space



- Data is linearly separable in polar coordinates
- Acts non-linearly in original space

SVM – Matlab Solver Algorithm

Both dual soft-margin problems are quadratic programming problems. Internally, fitcsvm has several different algorithms for solving the problems.

- **SMO:** For one-class or binary classification, if you do not set a fraction of expected outliers in the data, then the default solver is Sequential Minimal Optimization (SMO). SMO minimizes the one-norm problem by a series of two-point minimizations. SMO is relatively fast. For more details on SMO, see [ref].

- **ISDA:** For binary classification, if you set a fraction of expected outliers in the data, then the default solver is the Iterative Single Data Algorithm. Like SMO, ISDA solves the one-norm problem. For more details on ISDA, see [ref].

- **L1QP:** For one-class or binary classification, quadprog to solve the one-norm problem. quadprog uses a good deal of memory, but solves quadratic programs to a high degree of precision. For more details, see Quadratic Programming Definition [ref].

Outline

- What is SVM?
- The Optimization Problem
- The Kernel Trick
- Steps in SVM Modelling
- Support Vector Machine - Regression (SVR)
- References.

The Kernel Trick Definition

- A function that takes as its inputs vectors in the original space and returns the dot product of the vectors in the feature space is called a *kernel function*

- More formally, if we have data $\mathbf{x}, \mathbf{z} \in X$ and a

  map $\phi : X \rightarrow \Re^N$ then

$$k(\mathbf{x}, \mathbf{z}) = <\phi(\mathbf{x}), \phi(\mathbf{z})>$$

is a kernel function

Now we only need to compute *k(**x**, **z**)* and we don't need to perform computations in high dimensional space explicitly. This is what is called the Kernel Trick.

An Important Point

- Using kernels, we do not need to embed the data into the space $\Re^N$ explicitly, because a number of algorithms only require the inner products between image vectors!

- <u>We never need the coordinates of the data in the feature space!</u>

Kernel Example

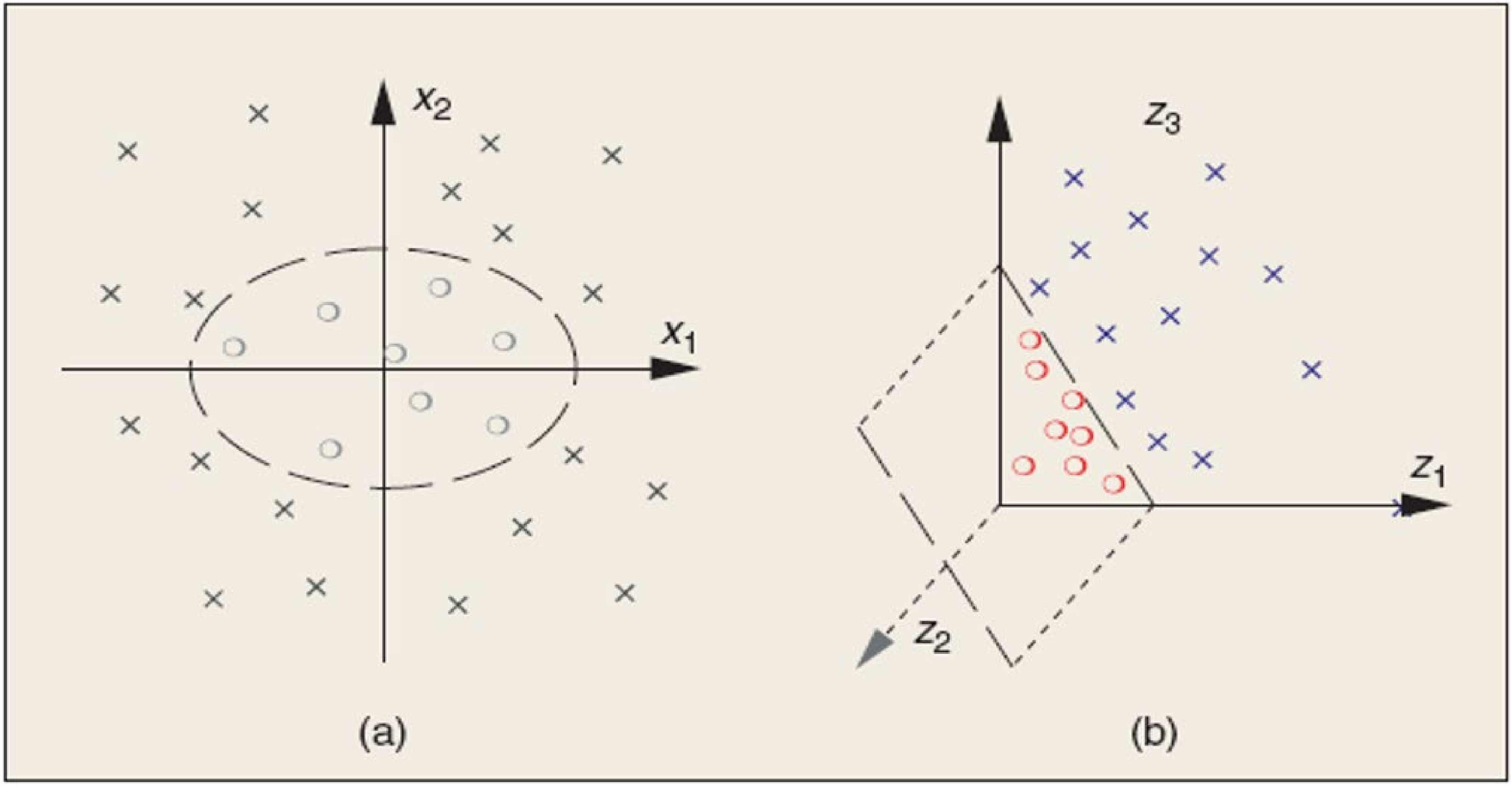
- Consider a two-dimensional input space $X \subseteq \Re^2$ with the feature map:

$$\phi : \mathbf{x} = (x_1, x_2) \mapsto \phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1 x_2) \in F = \Re^3$$

Now consider the inner product in the feature space:

$$\begin{aligned}\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle &= \left\langle (x_1^2, x_2^2, \sqrt{2}x_1 x_2), (z_1^2, z_2^2, \sqrt{2}z_1 z_2) \right\rangle \\ &= x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 x_2 z_1 z_2 \; = (x_1 z_1 + x_2 z_2)^2 \\ &= \langle \mathbf{x}, \mathbf{z} \rangle^2\end{aligned}$$

Kernel Example



▲ 1. *Effect of the map* $\phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$. *(a) Input space* $\mathcal{X}$ *and (b) feature space* $\mathcal{H}$.

Kernel Example

- Then $k(\mathbf{x},\mathbf{z}) = \langle \mathbf{x},\mathbf{z} \rangle^2$
- But $k(\mathbf{x},\mathbf{z})$ is also the kernel that computes the inner product of the map

$$\psi(\mathbf{x}) = (x_1^2, x_2^2, x_1 x_2, x_2 x_1) \in F = \Re^4$$

- This shows that a given feature space is not unique to a given kernel function

Example Transformation

Consider the following transformation

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

Define the kernel function K (**x**,**y**) as

$$\begin{aligned} \left\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) \right\rangle &= (1 + x_1y_1 + x_2y_2)^2 \\ &= K(\mathbf{x}, \mathbf{y}) \end{aligned}$$

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

The inner product $\phi(.)\phi(.)$ can be computed by K without going through the map $\phi(.)$ explicitly!!!

Modification Due to Kernel Function

Change all inner products to kernel functions

For training,

Original

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

With kernel function

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Example

Suppose we have 5 1D data points

– $x_1=1$, $x_2=2$, $x_3=4$, $x_4=5$, $x_5=6$, with 1, 2, 6 as class 1 and 4, 5 as class 2 $\Rightarrow y_1=1$, $y_2=1$, $y_3=-1$, $y_4=-1$, $y_5=1$

We use the polynomial kernel of degree 2

– $K(x,y) = (xy+1)^2$

– C is set to 100

Max. $$\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=j=1}^{N}\alpha_i\alpha_j y_i y_j x_i x_j$$

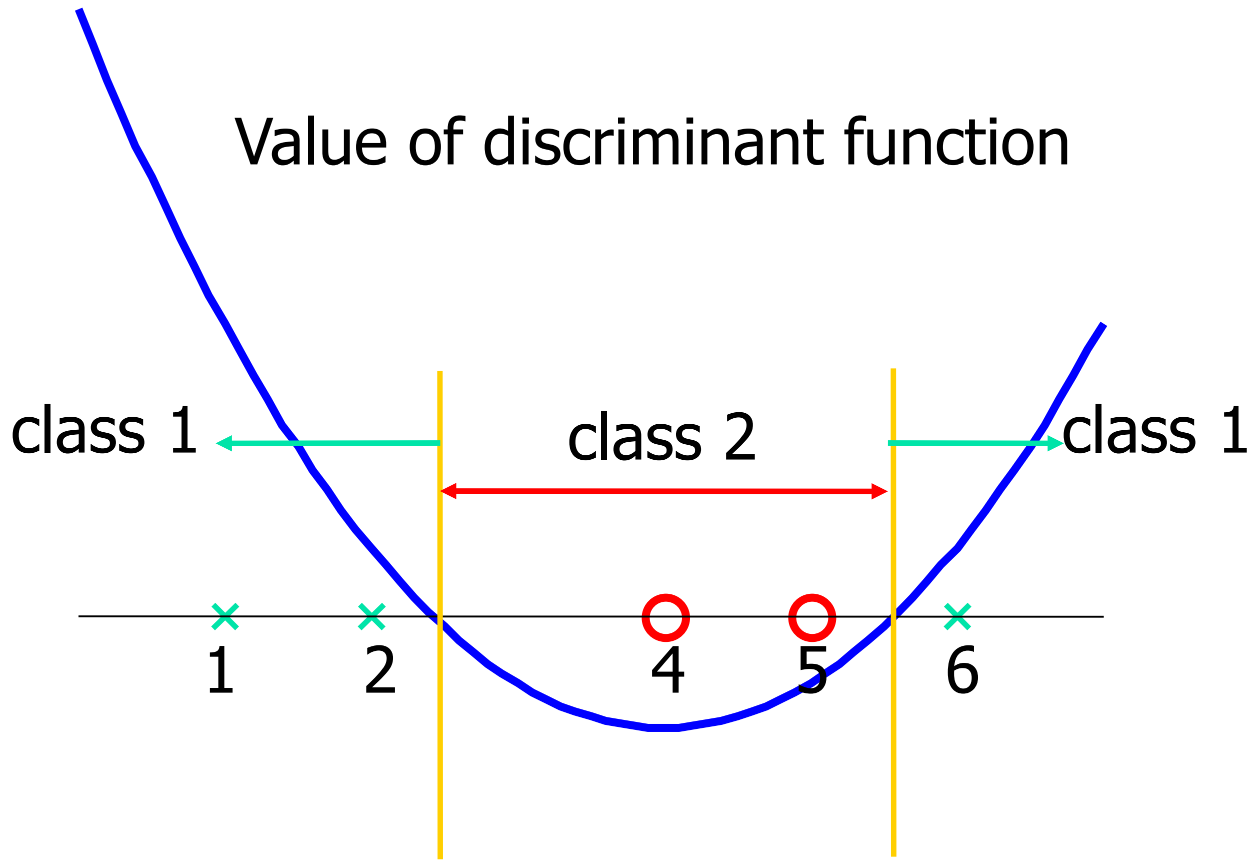
We first find α_i (i=1, …, 5) by

Subject to $$C \ge \alpha_i \ge 0, \sum_{i=1}^{N}\alpha_i y_i = 0$$

$$\text{max.} \quad \sum_{i=1}^{5}\alpha_i - \frac{1}{2}\sum_{i=1}^{5}\sum_{j=1}^{5}\alpha_i\alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \ge \alpha_i \ge 0, \sum_{i=1}^{5}\alpha_i y_i = 0$$

Example



Value of discriminant function
class 1
class 2
class 1
1
2
4
5
6

Choosing the Kernel Function

The kernel function is important because it creates the kernel matrix, which summarizes all the data

Many principles have been proposed (diffusion kernel, Fisher kernel, string kernel, …)

There is even research to estimate the kernel matrix from available information

In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try.

Note: that SVM with RBF kernel is closely related to RBF neural networks, with the centers of the radial basis functions automatically chosen for SVM

The Kernel Trick

- We want to map the patterns into a high-dimensional feature space *F* and compare them using a dot product

- To avoid working in the space *F*, choose a feature space in which the dot product can be evaluated directly using a nonlinear function in the input space

- This is called the *kernel trick*

$$k(\mathbf{x}, \mathbf{z}) = <\phi(\mathbf{x}), \phi(\mathbf{z})>$$

The Kernel Trick

- The relationship between the kernel function *K* and the mapping f(.) is

$$K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$$

  - This is known as the kernel trick

- In practice, we specify *K*, thereby specifying f(.) indirectly, instead of choosing f(.)

- Intuitively, *K* (**x**,**y**) represents our desired notion of similarity between data **x** and **y** and this is from our prior knowledge

- *K* (**x**,**y**) needs to satisfy a technical condition (Mercer condition) in order for f(.) to exist

The Kernel Trick

Recall:

Note that data only appears as dot products

maximize $$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=j=1}^{N} \alpha_i \alpha_j y_i y_j x_i x_j$$

subject to $$C \geq \alpha_i \geq 0, \sum_{i=1}^{N} \alpha_i y_i = 0$$

Since data is only represented as dot products, we need not do the mapping explicitly.

Introduce a Kernel Function (*) *K* such that: $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$

(*)Kernel function – a function that can be applied to pairs of input data to evaluate dot products in some corresponding feature space

The Kernel Trick

Linear SVM

$$x_i \cdot x_j$$

Non-linear SVM

$$\phi(x_i) \cdot \phi(x_j)$$

map data into new space, then take the inner product of the new vectors.

Kernel function

$$k(x_i \cdot x_j)$$

the image of the inner product of the data is the inner product of the images of the data.

SVM – Kernel Functions

- **Polynomial kernel** (No prior knowledge about the data)
- **Gaussian kernel** (No prior knowledge about the data)
- **Gaussian radial basis function (RBF)** (No prior knowledge about the data)
- **Laplace RBF kernel** (No prior knowledge about the data)
- **Hyperbolic tangent kernel** (Use it in neural networks)
- **Sigmoid kernel** (Use it as proxy for neural networks)
- **Bessel function of the first kind Kernel** (Use it to remove the cross term in mathematical functions)
- **ANOVA radial basis kernel** (Use it in regression problems)
- **Linear splines kernel in one-dimension** (for large sparse data vectors)

Choosing the Kernel Function

Probably the most tricky part of using SVM:

- The kernel function is important because it creates the kernel matrix, which summarizes all the data
- Many principles have been proposed (diffusion kernel, Fisher kernel, string kernel, …)
- There is even research to estimate the kernel matrix from available information

✓ In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try

Note: SVM with RBF kernel is closely related to RBF neural networks, with the centers of the radial basis functions automatically chosen for SVM

The Most Used Type of Kernel Functions

Linear

$$k(\mathrm{x}_i, \mathrm{x}_j) = \mathrm{x}_i \, \mathrm{x}_j$$

Polynomial

$$k(\mathrm{x}_i, \mathrm{x}_j) = (\mathrm{x}_i + \mathrm{x}_j)^{\mathrm{d}}$$

Gaussian Radial Basis function (RBF)

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

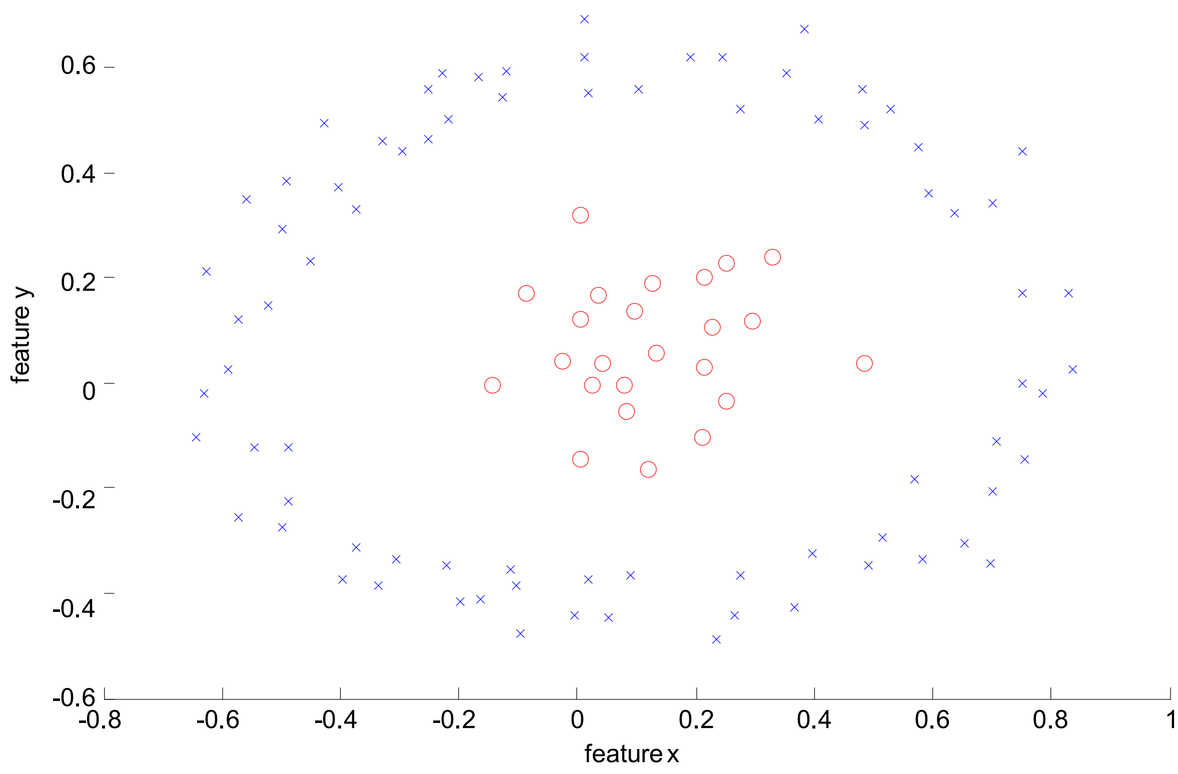
Sigmoid

$$k(\mathrm{x_i},\mathrm{x_j}) = \tanh(\beta_0 \mathrm{x_i}^{\mathrm{T}} \mathrm{x_j} + \beta_1)$$

SVM Softwares

- A list of SVM implementation can be found at:
  http://www.kernel-machines.org/software.html
- Some implementation (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementation of SVM
- Several Matlab toolboxes for SVM are also available

RBF Kernel SVM Example



- Data is not linearly separable in original feature space

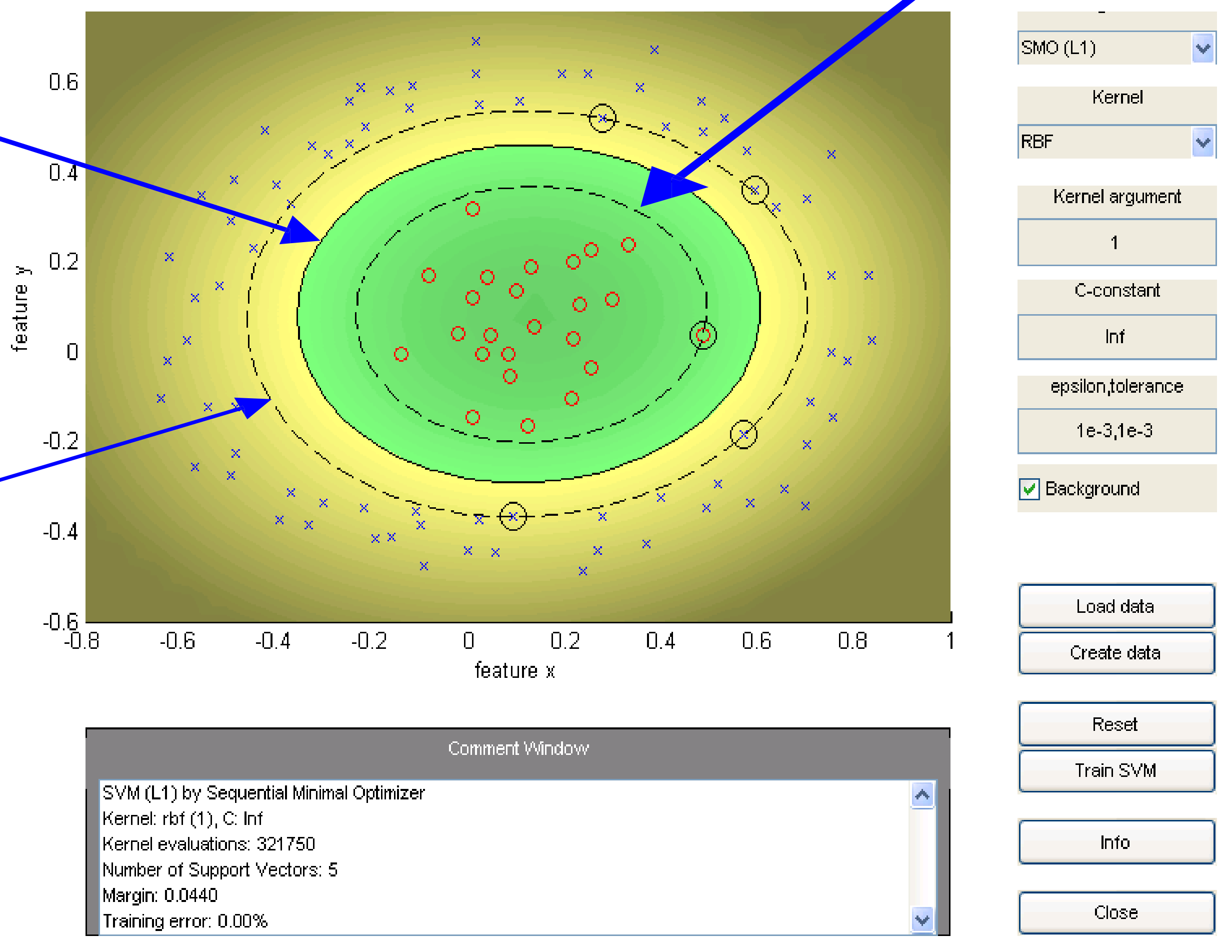
RBF Kernel SVM Example

$\sigma = 1.0, C = \infty$

$f(x) = 1$

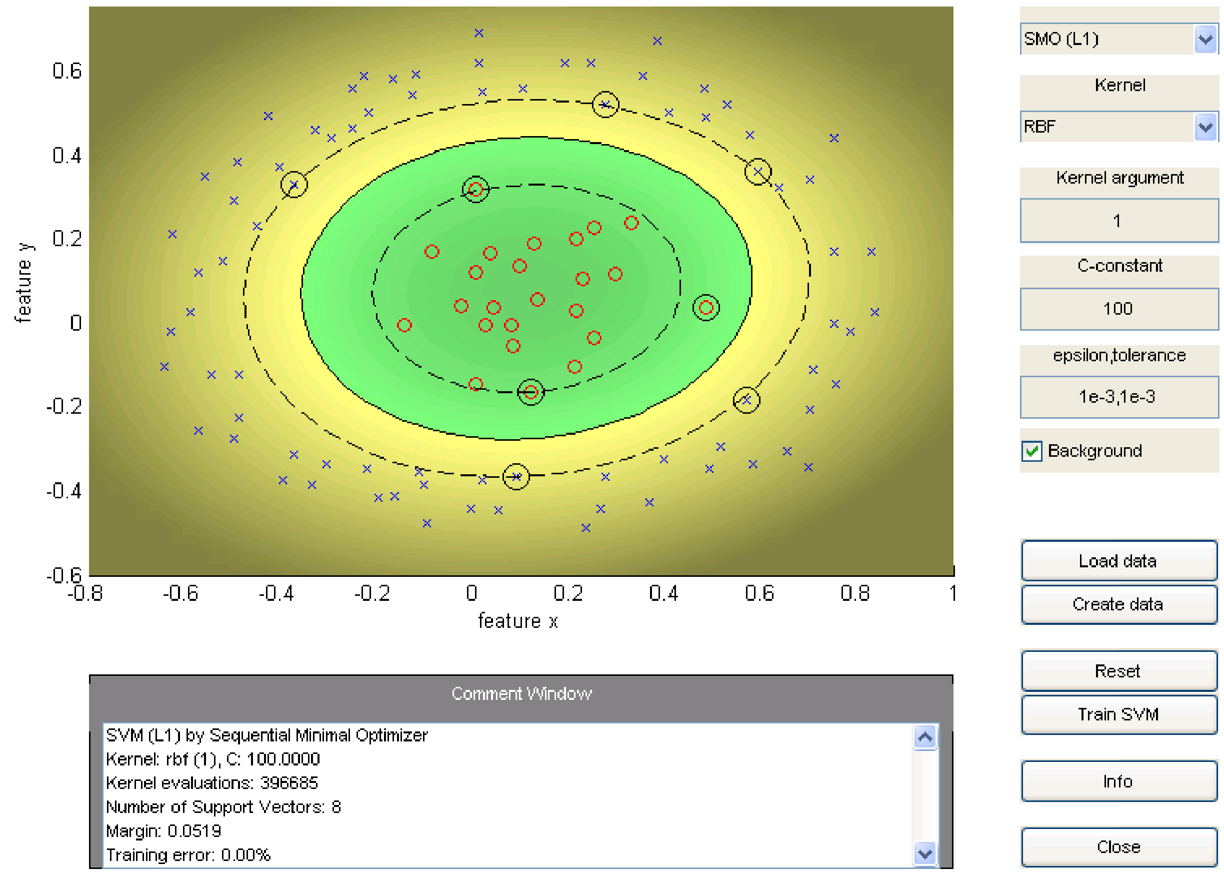
$f(x) = 0$

$f(x) = -1$



RBF Kernel SVM Example

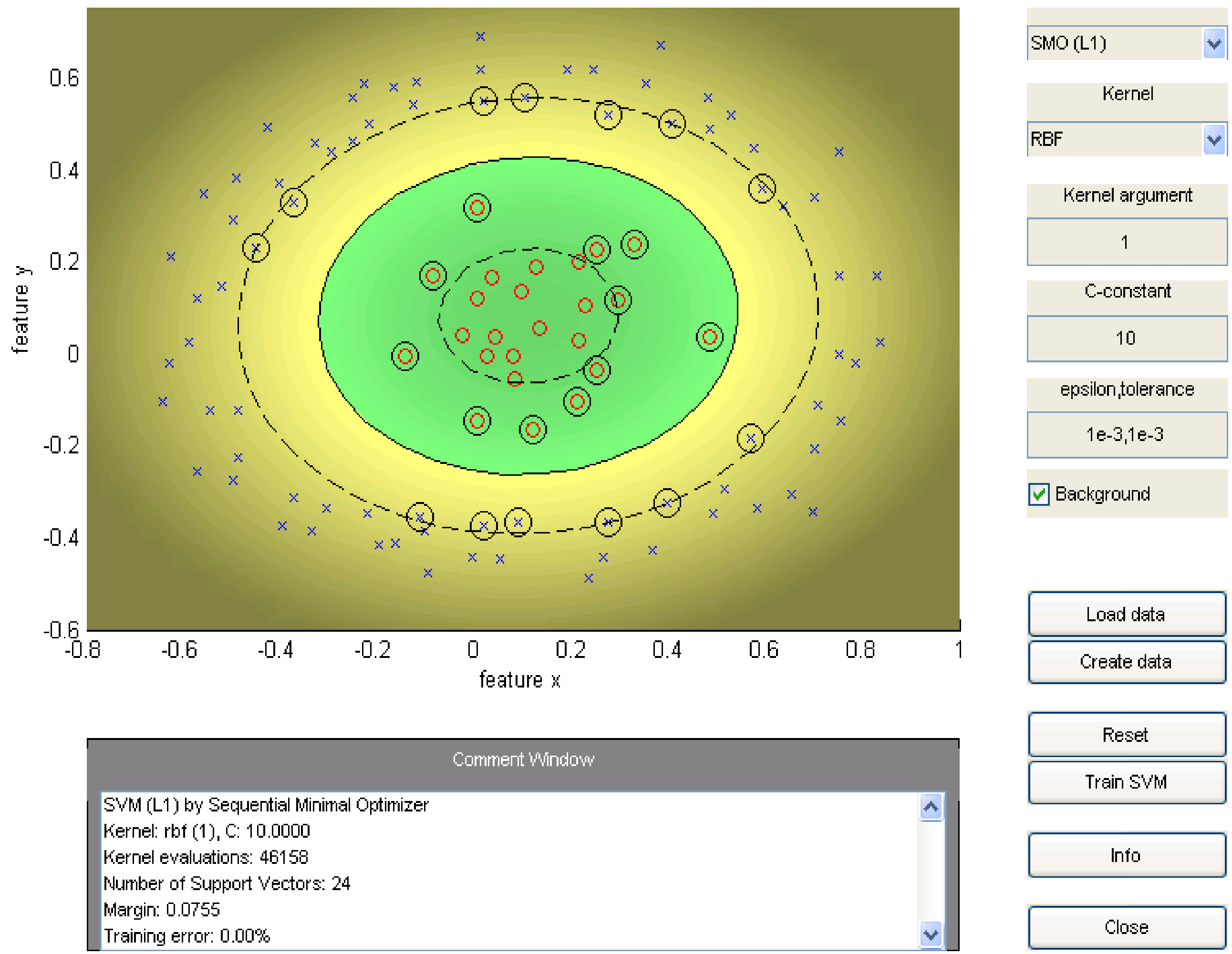
$\sigma = 1.0, C=100$



Decrease C, gives wider (soft) margin

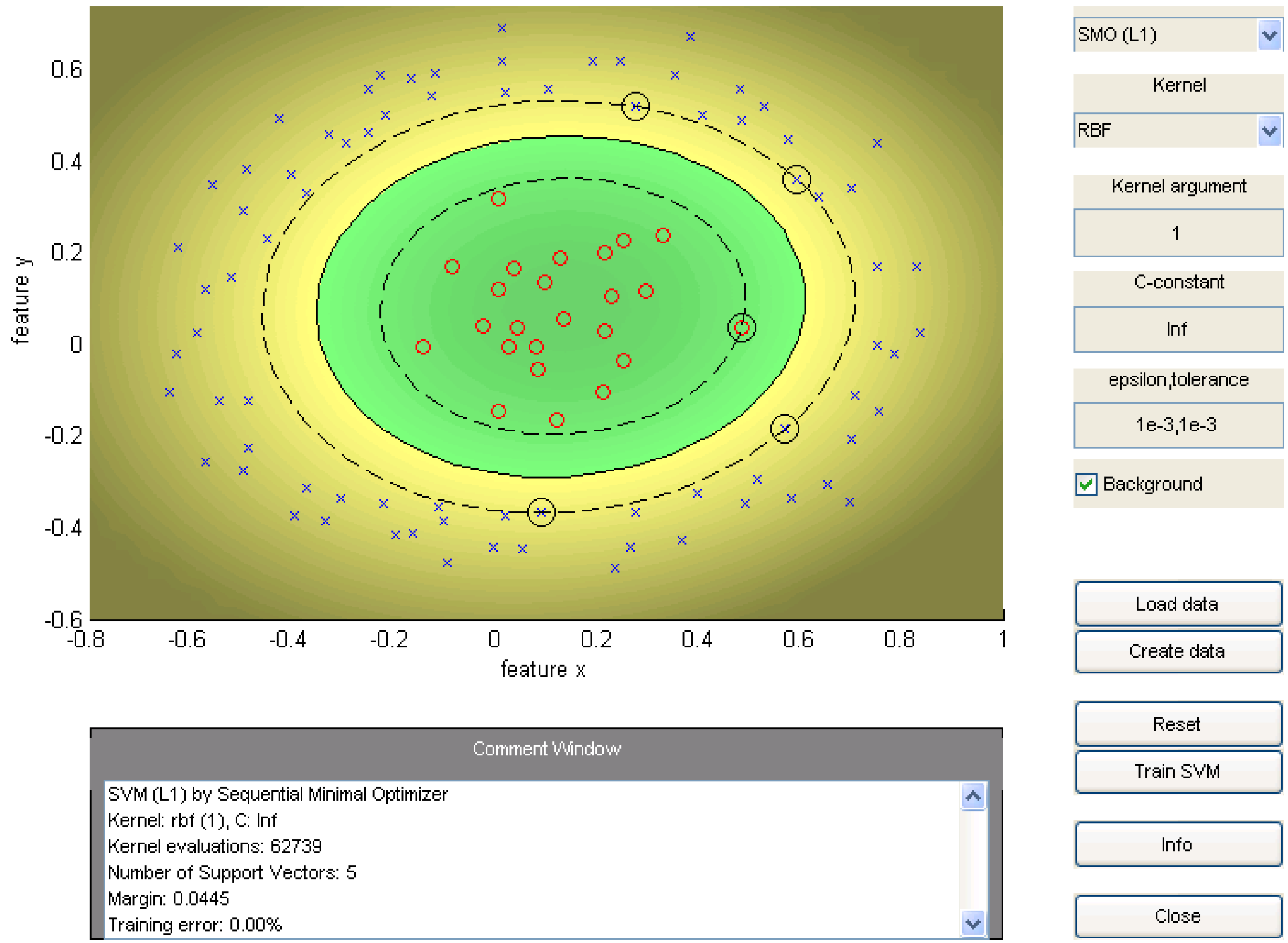
RBF Kernel SVM Example

$\sigma = \ 1.0, \ C=10$



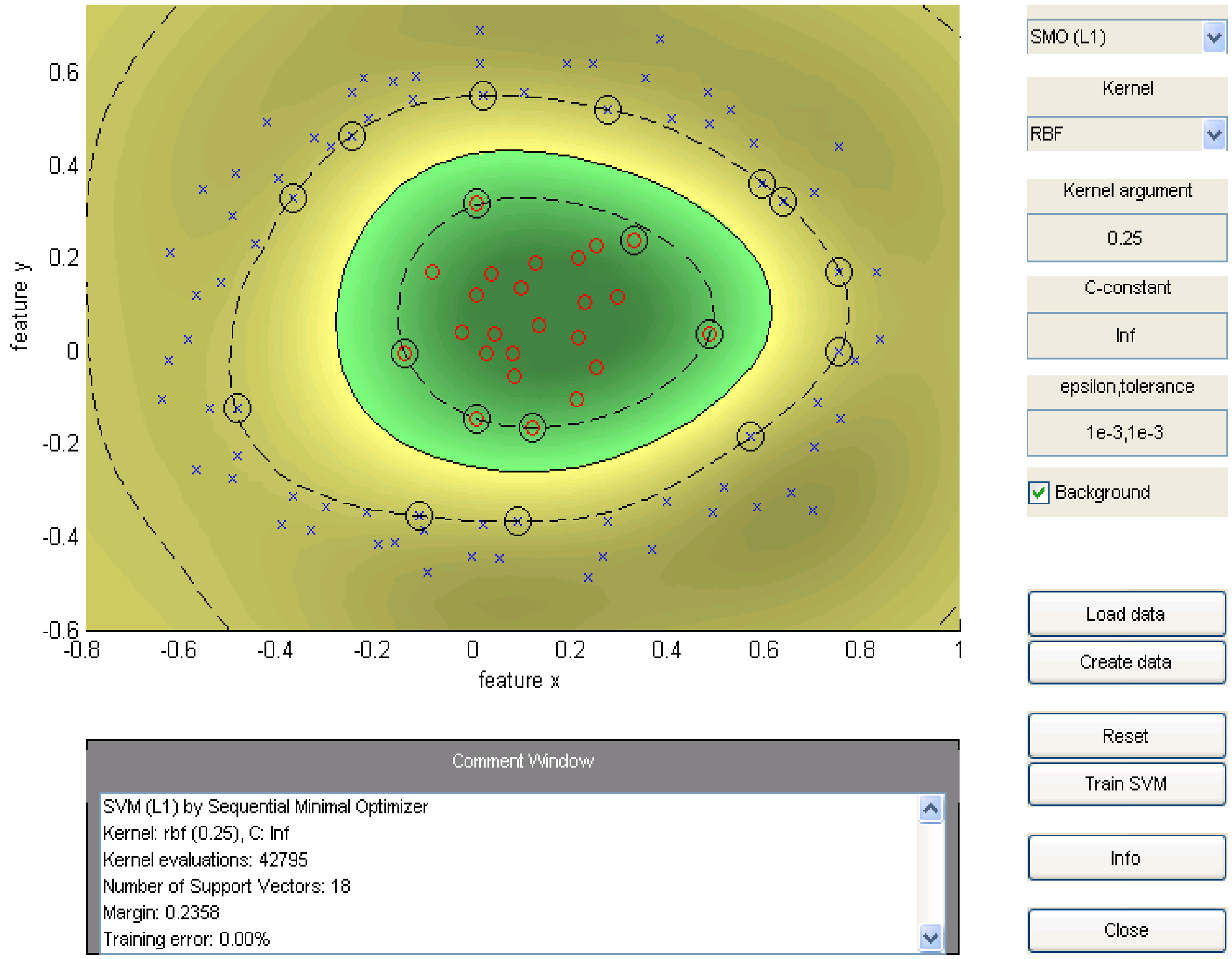
RBF Kernel SVM Example

$\sigma = 1.0 \quad C = \infty$



RBF Kernel SVM Example

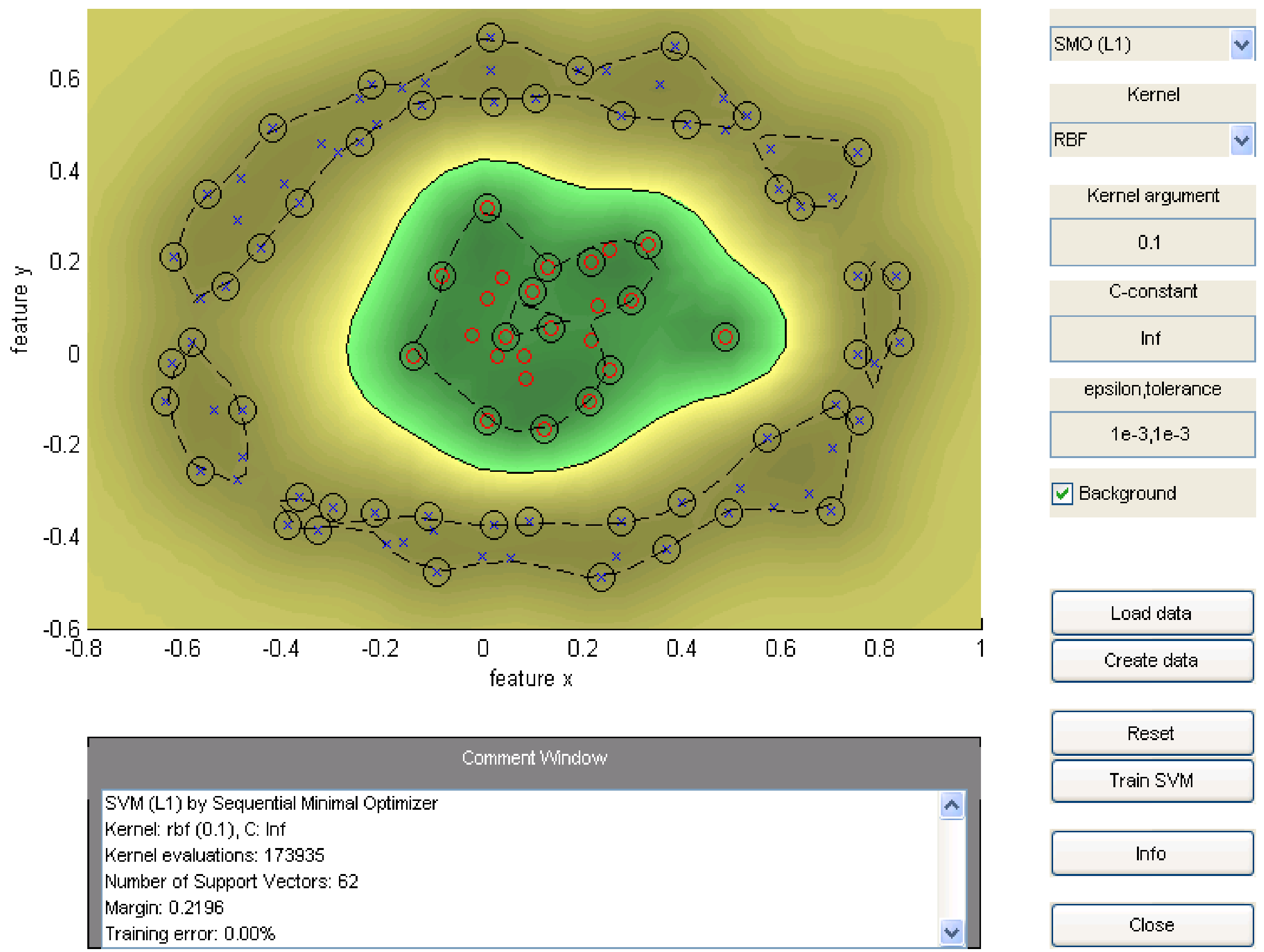
$\sigma = 0.25 \quad C = \infty$



Decrease sigma, moves towards nearest neighbour classifier

RBF Kernel SVM Example

$\sigma = 0.1 \quad C = \infty$



Outline

- What is SVM?
- The Optimization Problem
- Soft vs Hard Margin SVM
- The Kernel Trick
- Steps in SVM Modelling
- Support Vector Machine - Regression (SVR)
- References.

Steps in SVM Modelling

1. Prepare data matrix $\{(x_i,y_i)\}$
2. Select a Kernel function
3. Select the error parameter *C*
4. "Train" the system (to find all α_i)
5. New data can be classified using α_i and Support Vectors

Software

A list of SVM implementation can be found at http://www.kernel-machines.org/software.html

Some implementation (such as LIBSVM) can handle multi-class classification

SVMLight is among one of the earliest implementation of SVM

Several Matlab toolboxes for SVM are also available

Summary

Weaknesses:

- Training (and Testing) is quite slow compared to ANN
  - Because of Constrained Quadratic Programming
- Essentially a binary classifier
  - However, there are some tricks to evade this.
- Very sensitive to noise
  - A few off data points can completely throw off the algorithm
- Biggest Drawback: The choice of Kernel function.
  - There is no “set-in-stone” theory for choosing a kernel function for any given problem (still in research...)
  - Once a kernel function is chosen, there is only ONE modifiable parameter, the error penalty *C*.

Summary

Strengths:

- Training is relatively easy
  - We don’t have to deal with local minimum like in ANN
  - SVM solution is always global and unique (check “Burges” paper for proof and justification).
- Unlike ANN, doesn’t suffer from “curse of dimensionality”.
  - How? Why? We have infinite dimensions?!
  - Maximum Margin Constraint: DOT-PRODUCTS!
- Less prone to overfitting
- Simple, easy to understand geometric interpretation.
  - No large networks to mess around with.

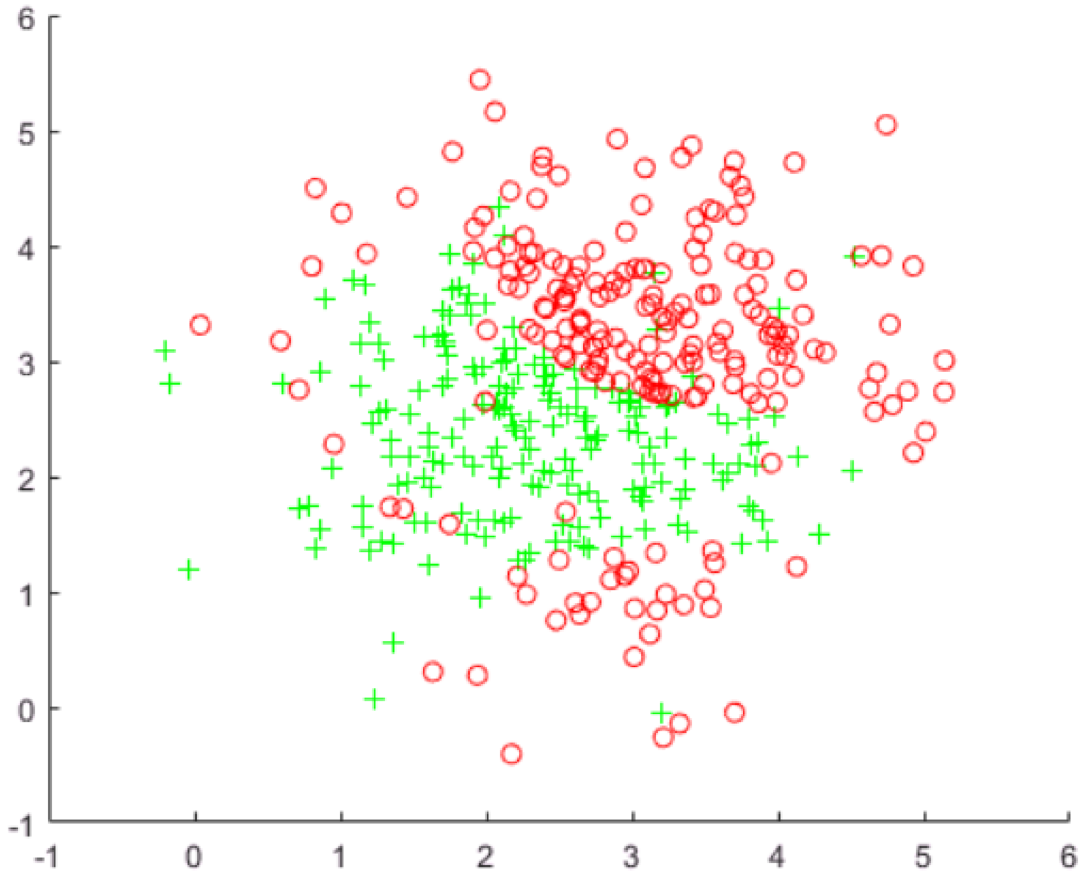
Applications of SVMs

- Bioinformatics
- Machine Vision
- Text Categorization
- Ranking (e.g., Google searches)
- Handwritten Character Recognition
- Time series analysis

→Lots of very successful applications!!!

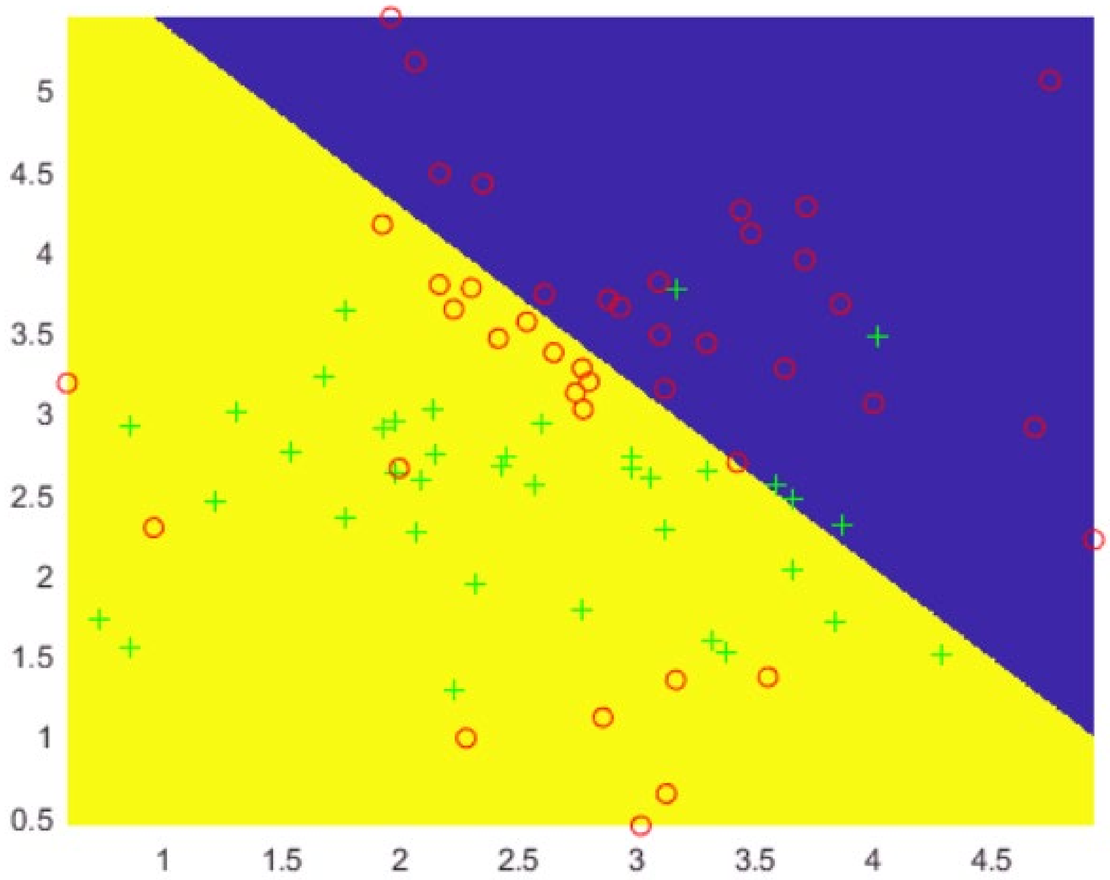
Example of SVM Using Various Kernels

Data Pre-processing



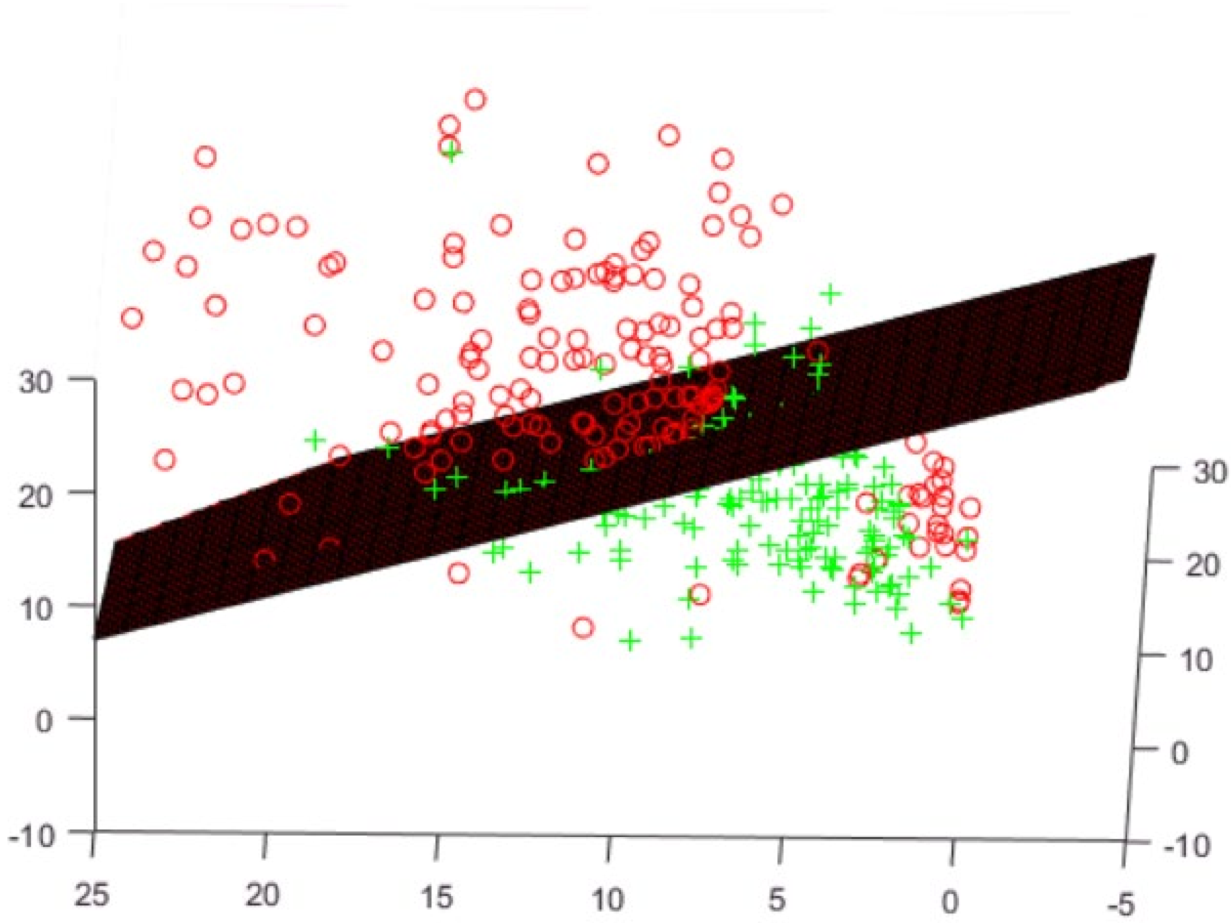
Example of SVM Using Various Kernels

Decision Boundary in input space (Linear Kernel)



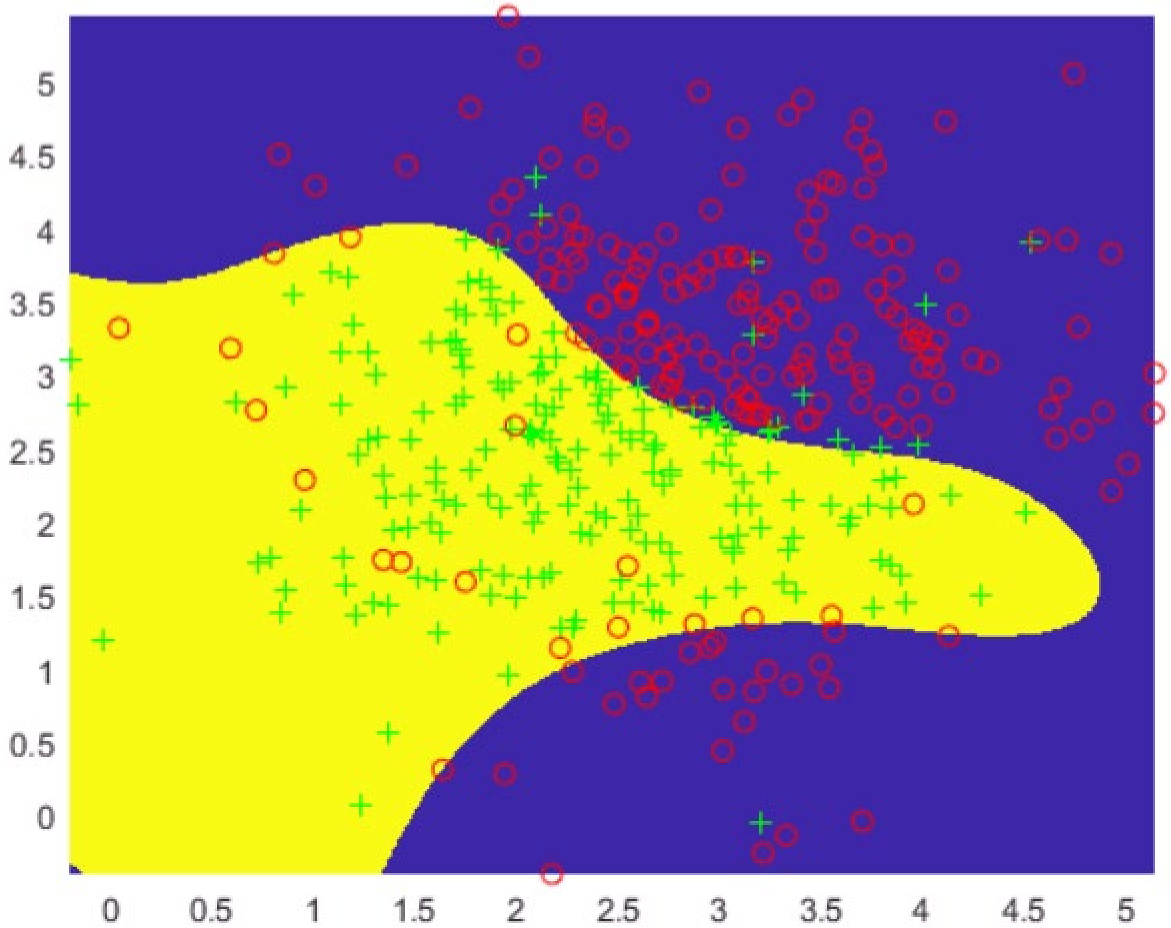
Example of SVM Using Various Kernels

Applying Polynomial Kernel (Poly2)



Example of SVM Using Various Kernels

Decision Boundary in input space (RBF)



Other Types of SVM

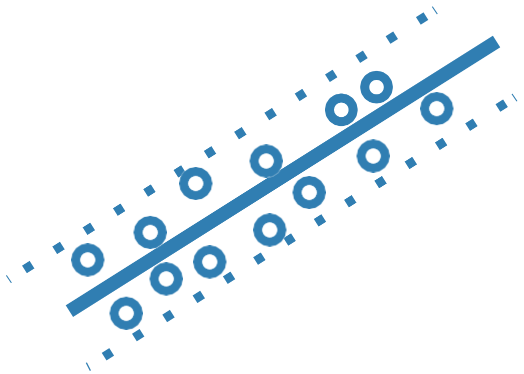
- SVMs that perform regression (SVR) .
- SVMs that perform clustering.
- SVM formulations that take into consideration difference in cost of misclassification for the different classes.
- Kernels suitable for sequences of strings, or other specialized kernels.

Outline

- ❑ What is SVM?
- ❑ The Optimization Problem
- ❑ The Kernel Trick
- ❑ Steps in SVM Modelling
- ❑ Support Vector Machine - Regression (SVR)
- ❑ References.

Support Vector Machine - Regression (SVR)

SVM regression algorithms work like SVM classification algorithms, but are modified to be able to predict a continuous response.
Instead of finding a hyperplane that separates data, SVM regression algorithms find a model that deviates from the measured data by a value no greater than a small amount, with parameter values that are as small as possible (to minimize sensitivity to error).



Best Used...

- For high-dimensional data

(where there will be a large number of predictor variables)

Support Vector Regression (SVR)

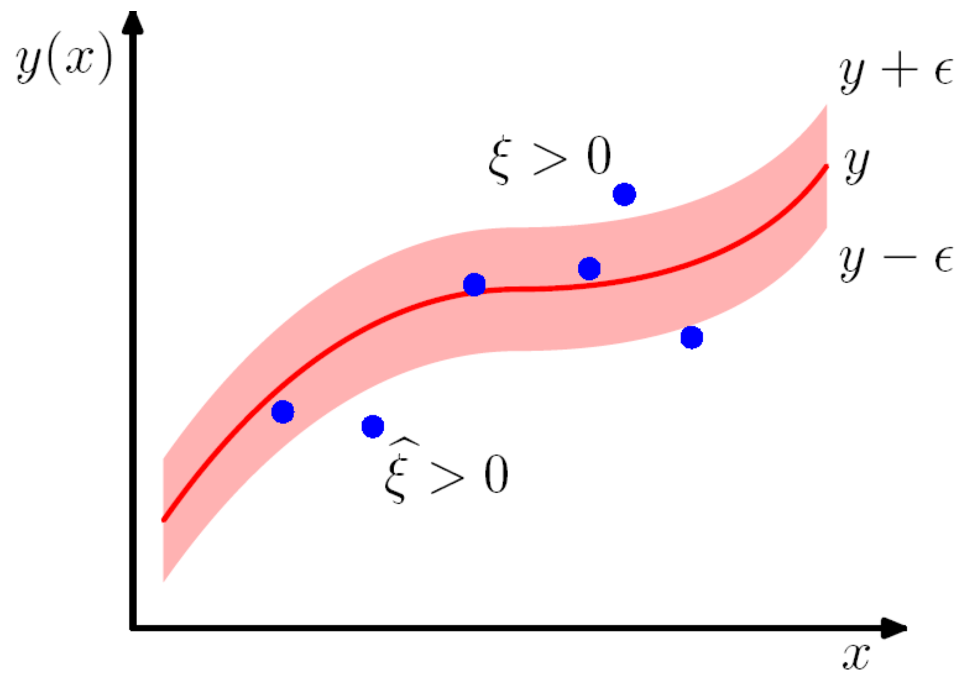
For a target point to lie inside the tube:

$$y_n - \in \leq t_n \leq y_n + \in$$

Introduce slack variables to allow points to lie outside the tube:

$$t_n \leq y(x_n) + \in + \xi_n$$

$$t_n \geq y(x_n) - \in - \xi_n^{-}$$



Support Vector Regression (SVR):Error Function

Minimize:

$$C\sum_{n=1}^{N}(\xi_n+\xi_n^-)+\frac{1}{2}\|w\|^2$$

Subject to:

$$\xi_n \geq 0 \qquad \text{and} \qquad t_n \leq y(x_n)+\in+\xi_n$$

$$\xi_n^- \geq 0 \qquad\qquad\qquad t_n \geq y(x_n)-\in-\xi_n^-$$

Support Vector Regression (SVR): Lagrangian

Minimize:

$$L = C\sum_{n=1}^{N}(\xi_n + \xi_n^-) + \frac{1}{2}\|w\|^2 - \sum_{n=1}^{N}(\mu_n\xi_n + \mu_n^-\xi_n^-) - \sum_{n=1}^{N}a_n(\in + \xi_n + y_n - t_n) - \sum_{n=1}^{N}a_n^-(\in + \xi_n^- - y_n + t_n)$$

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{n=1}^{N}(a_n - a_n^-)\phi(x_n)$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N}(a_n - a_n^-) = 0$$

$$\frac{\partial L}{\partial \xi_n} = 0 \Rightarrow a_n + \mu_n = C$$

$$\frac{\partial L}{\partial \xi_n^-} = 0 \Rightarrow a_n^- + \mu_n^- = C$$

Support Vector Regression (SVR): How to determine b?

Karush-Kuhn-Tucker (KKT) conditions:

$$a_n(\in + \xi_n + y_n - t_n) = 0$$

$$a_n^-(\in + \xi_n^- - y_n + t_n) = 0$$

$$(C - a_n)\xi_n = 0$$

$$(C - a_n^-)\xi_n^- = 0$$

Support vectors are points that lie on the boundary or outside the tube

$$b = t_n - \in - w^T \phi(x_n) = t_n - \in - \sum_{m=1}^{N} (a_m - a_m^-) k(x_n, x_m)$$

Support Vector Regression (SVR)

http://www.saedsayad.com/support_vector_machine_reg.htm

The Kernel Motivation

- Problem: Representing data in a high-dimensional space is computationally difficult

- Alternative solution to the original problem:

  Calculate a similarity measure in the feature space instead of the coordinates of the vectors there, then apply algorithms that only need the value of this measure

  - Use dot product as similarity measure

Support Vector Regression (SVR)

- Find a function, f(x), with at most □-deviation from the target y

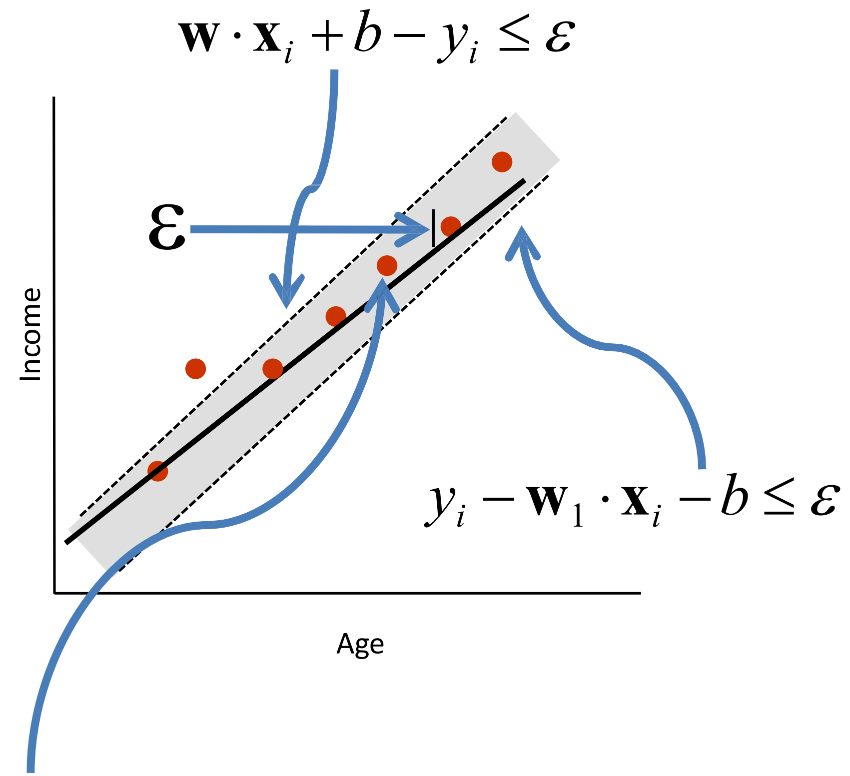
The problem can be written as a convex optimization problem

$$\min \frac{1}{2}\|\mathbf{w}\|^2$$
$$s.t.\ y_i - \mathbf{w}_1 \cdot \mathbf{x}_i - b \le \varepsilon;$$
$$\mathbf{w}_1 \cdot \mathbf{x}_i + b - y_i \le \varepsilon;$$

C: trade off the complexity

What if the problem is not feasible?

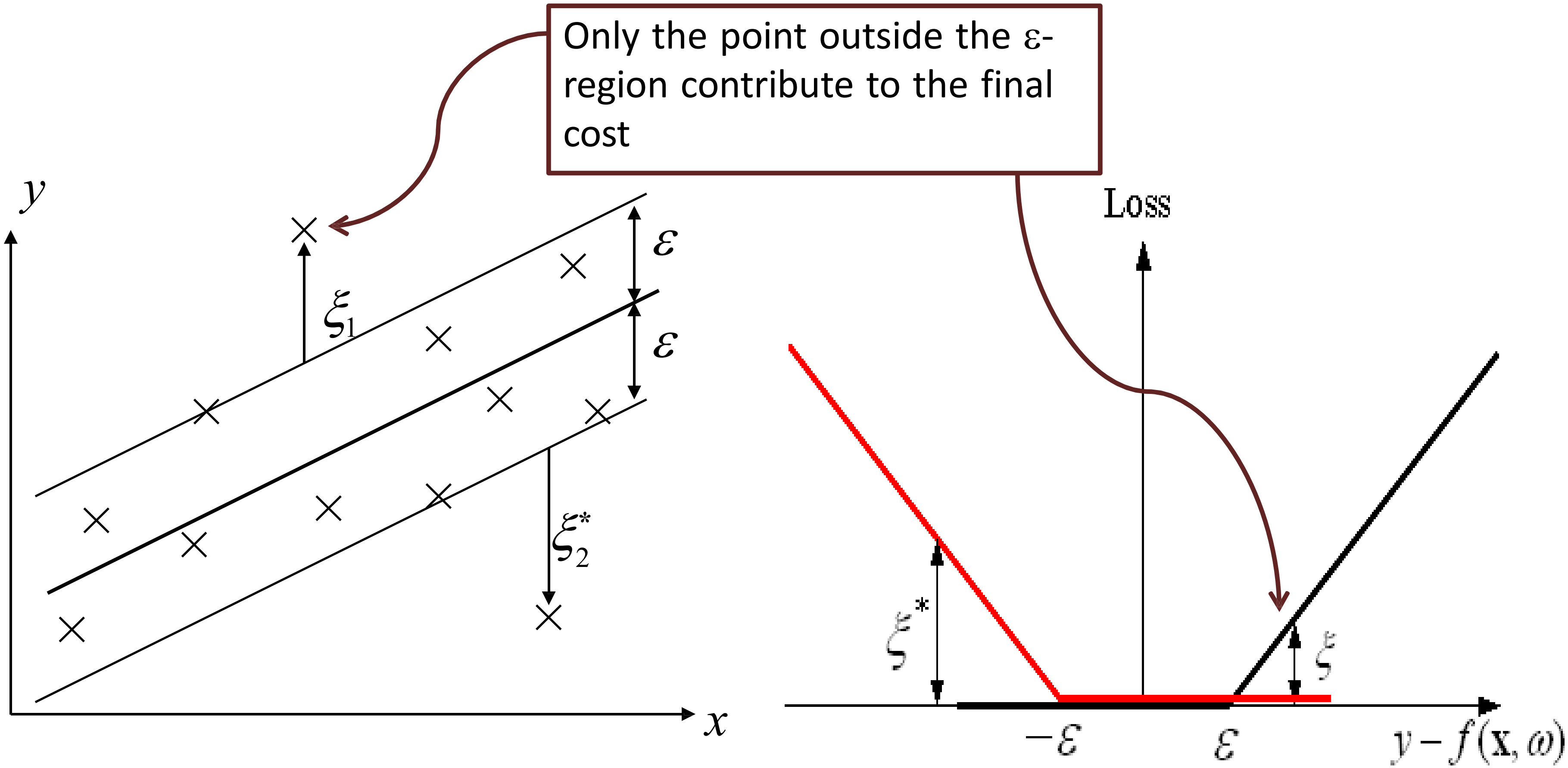
We can introduce slack variables (similar to soft margin loss function).



We do not care about errors as long as they are less than ε

Support Vector Regression

Assume linear parameterization $f(\mathbf{x},\omega) = \mathbf{w} \cdot \mathbf{x} + b$

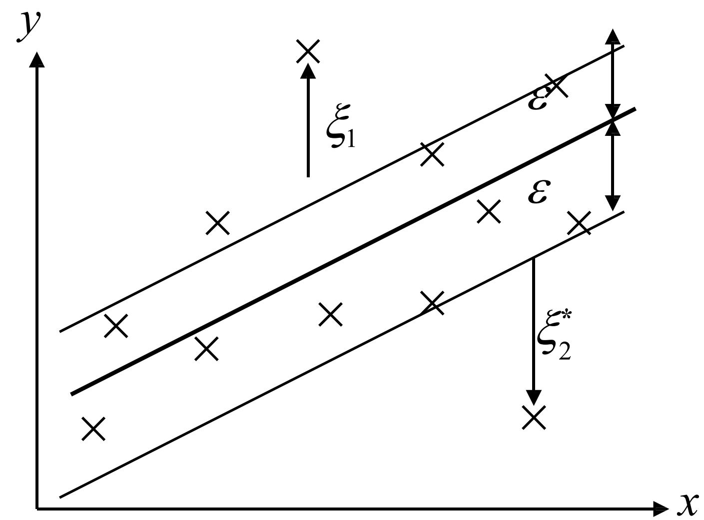


$$L_\varepsilon(y, f(\mathbf{x},\omega)) = \max\left(|\, y - f(\mathbf{x},\omega)\,| - \varepsilon, 0\right)$$

Soft Margin

Given training data

$$(\mathbf{x}_i, y_i) \quad i = 1, \ldots, m$$



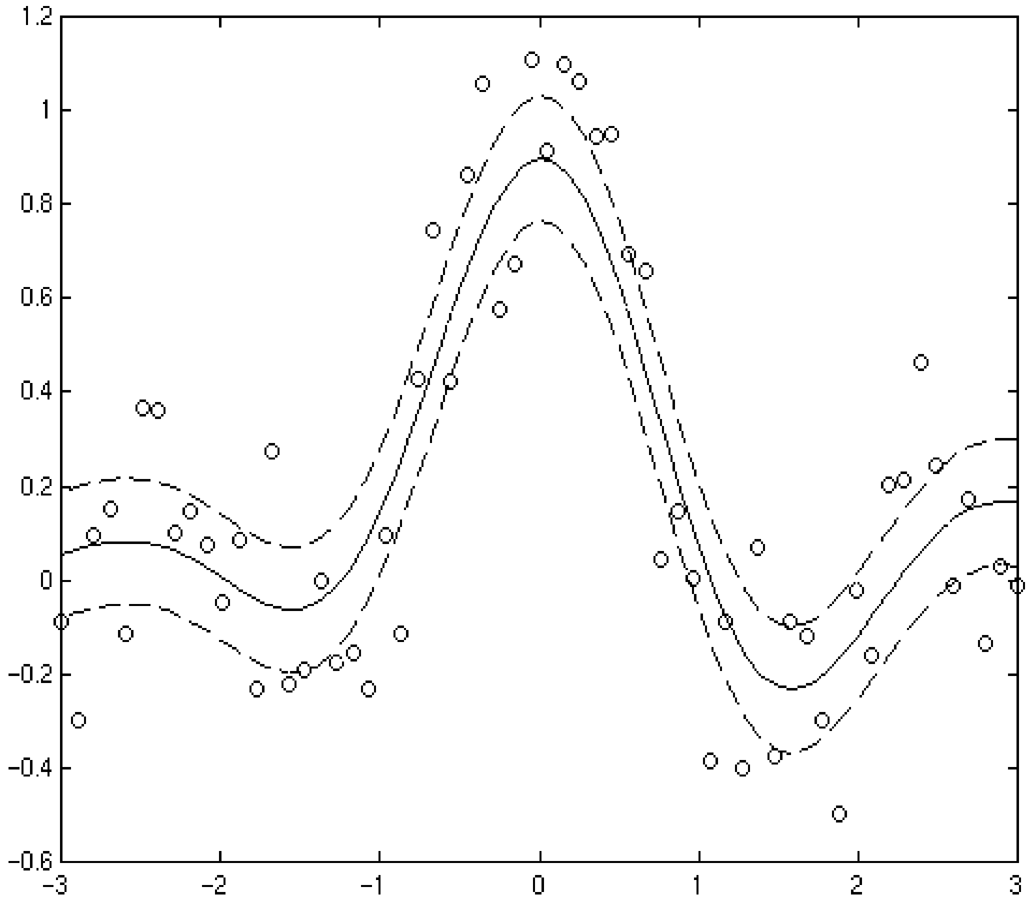
Minimize

$$\frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{m} (\xi_i + \xi_i^*)$$

Under constraints

$$\begin{cases} y_i - (\mathbf{w} \cdot \mathbf{x}_i) - b \le \varepsilon + \xi_i \\ (\mathbf{w} \cdot \mathbf{x}_i) + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0, i = 1, \ldots, m \end{cases}$$

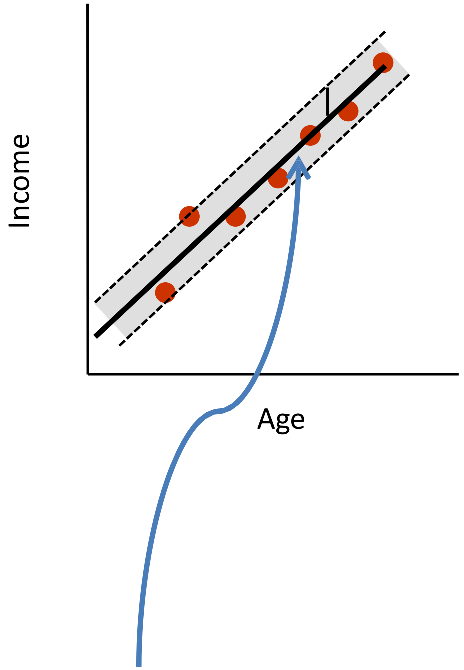
How About a Non-linear Case?



Linear Versus Non-Linear SVR

- **Linear case**

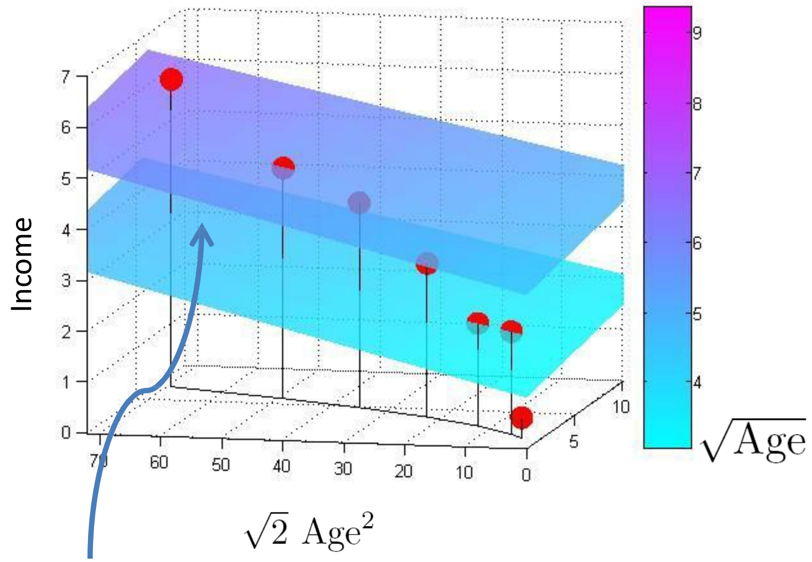
$$f : age \rightarrow income$$



$$y_i = \mathbf{w}_1 \cdot \mathbf{x}_i + b$$

- **Non-linear case**
  - Map data into a higher dimensional space, e.g.,

$$f : (\sqrt{age}, \sqrt{2}age^2) \rightarrow income$$



$$y_i = \mathbf{w}_1 \sqrt{\mathbf{x}_i} + \mathbf{w}_2 \sqrt{2}\mathbf{x}_i^2 + b$$

Support Vector Regression (Gaussian Kernel)

Initialization

```
clear all;clc;
% Zscore Normalization
data=zscore(csvread('GaussaianData.csv'))
;
% Input -> x, Output -> y
x=data(:,1:end-1);
y=data(:,end);
% Number of data points
N=length(data);
alpha=zeros(N,1);
% Tolerence value
norm1=Inf; tol=10e-1;
% Maximum number of iterations
itr=0; maxItr=10e2;
eps=0.1;
```

Support Vector Regression (Gaussian Kernel)

Algorithm

```
while (norm1>tol && itr<maxItr)
alpha_old=alpha;
alpha_=alpha;
for i=1:N
alpha(i)=alpha(i) + y(i) -
eps*sign(alpha(i))...
-alpha'*kernel(x,x(i,:),'g')';
if alpha_(i)*alpha(i)<0
alpha(i)=0;
end
end
norm1=norm(alpha_old-alpha);
itr=itr+1;
end
fprintf('Total number of iteration %d',itr)
```

Support Vector Regression (Gaussian Kernel)

Initialization

```
clear all;clc;
% Zscore Normalization
data=zscore(csvread('GaussaianData.csv'))
;
% Input -> x, Output -> y
x=data(:,1:end-1);
y=data(:,end);
% Number of data points
N=length(data);
alpha=zeros(N,1);
% Tolerence value
norm1=Inf; tol=10e-1;
% Maximum number of iterations
itr=0; maxItr=10e2;
eps=0.1;
```

Support Vector Regression (Gaussian Kernel)

Weights

```
w=sum(alpha.*x)
```

Bias

```
b=mean(y-(w*x')' -eps*ones(N,1))
```

Predicted values

```
for j=1:N
fx1(j,:)=alpha(j)*kernel(x,x(j,:),'g')';
end
fx=sum(fx1)';
disp('[Actual Values Predicted Values]')
```

Support Vector Regression (Gaussian Kernel)

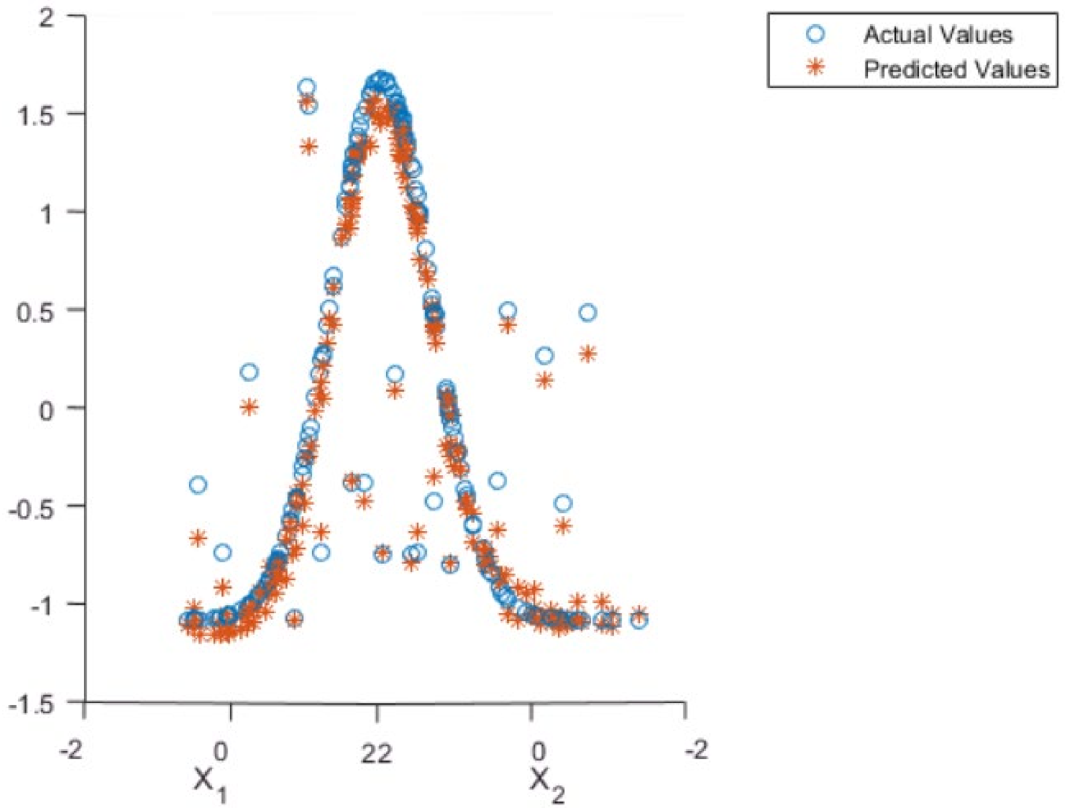
Mean Square error (Gaussian Kernel)

```
mse=norm(y-fx)^2/N
```

mse = 0.0132

Plotting

```
figure
hold on
scatter3(x(:,1),x(:,2),y)
scatter3(x(:,1),x(:,2),fx,'*')
hold off
xlabel({'X_1'});
ylabel({'X_2'});
view([-46.4 -0.40]);
legend1 = legend('Actual Values','Predicted Values');
```



Conclusion

- SVM and SVR is a useful alternative to neural networks

- Two key concepts of SVM and SVR:
  - **(i) maximize the margin**
  - **(ii) the kernel trick**

- Many active research is taking place on areas related to SVM and SVR

- Many SVM and SVR implementations are available on the web for you to try on your data set!

Model Evaluation/Performance

- Mean Square Error (MSE) is the average squared difference between outputs and targets.
  Lower values are better.
  Zero means no error.

$$MSE = \frac{\sum_{i=1}^{n}(X_{obs,i} - X_{model,i})^2}{n} \quad (1)$$

where Xobs is observed values and Xmodel is modelled values.

- Root Mean Square Error (RMSE) is the average root squared difference between outputs and targets.
  Lower values are better.
  Zero means no error.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(X_{obs,i} - X_{model,i})^2}{n}} \quad (2)$$

where Xobs is observed values and Xmodel is modelled values.

- Regression (R) values measure the correlation between outputs and targets.
  An R value of 1 means a close relationship, 0 a random relationship

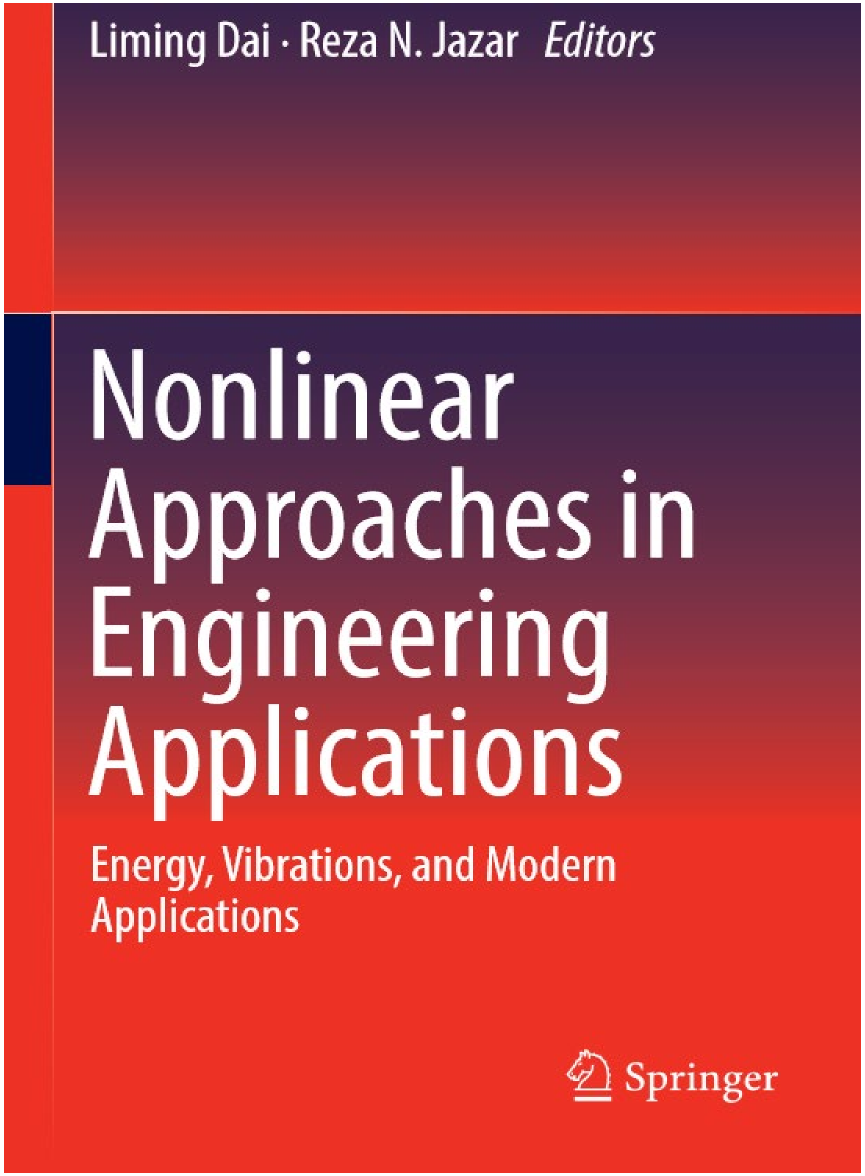
$$R = \frac{n(\sum x_o x_m) - (\sum x_o)(\sum x_m)}{\sqrt{[n\sum x_o^2 - (\sum x_o)^2)][n\sum x_m^2 - (\sum x_m)^2]}} \quad (3)$$

where Xo is observed values and Xm is modelled values.

Outline

- What is SVM?
- The Optimization Problem
- The Kernel Trick
- Steps in SVM Modelling
- Support Vector Machine - Regression (SVR)
- References

References



Chapter 12
Limited Data Modelling Approaches for Engineering Applications

Hamid Khayyam, Gelayol Golkarnarenji, and Reza N. Jazar

12.1 Introduction

Over the past several years, the study of various complex systems has been of great interest to researchers and scientists. Complex systems and problems are very pervasive and appear in different application areas including education, healthcare, medicine, finance, marketing, homeland security, defense, and environmental management, among others. In these systems, many components are involved with nonlinear interactions. Forecasting the future state of a complex system and designing such a system are very costly, time consuming, and compute intensive due to project times and technical constraints in industry. To overcome these complexities and save considerable amount of cost, time, and energy, modelling can be utilized. Modelling is generally defined as mathematical realization and computerized analysis of abstract representation of real systems. It helps achieve comprehensive insight into the functionality of the modelled systems, investigate the performance and behavior of processes, and finally optimize the process control. Mathematical modelling is an inexpensive and a powerful paradigm to deal with real-world complex problems. It comprises a wide range of computational methods. This technique can lower the costs by reducing the number of experiments and increasing the safety by forecasting the events, the results of laboratory tests, or the industrial data (Dobre and Sanchez Marcano 2007; Pham 1998; Rodrigues and Minceva 2005).

H. Khayyam (✉) • R.N. Jazar
School of Engieering, RMIT University, Melbourne, VIC, Australia
e-mail: hamid.khayyam@rmit.edu.au

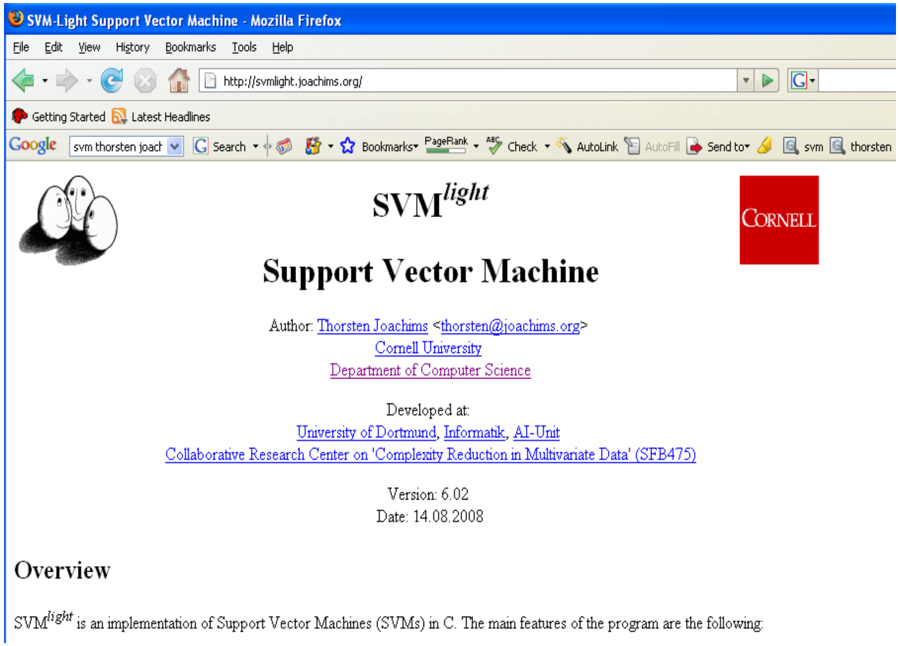
G. Golkarnarenji
Institute for Frontier Materials, Carbon Nexus, Deakin University, Waurn Ponds, VIC, Australia

 345
L. Dai, R.N. Jazar (eds.), *Nonlinear Approaches in Engineering Applications*,
https://doi.org/10.1007/978-3-319-69480-1_12

References

- Burges, C. “A Tutorial on Support Vector Machines for Pattern Recognition.” Bell Labs. 1998
- Law, Martin. “A Simple Introduction to Support Vector Machines.” Michigan State University. 2006
- Prabhakar, K. “An Introduction to Support Vector Machines”
- Bierens, Herman J. Introduction to Hilbert Spaces. Pennsylvania State University. 24 June 2007.
- Burges, Christopher. A Tutorial on Support Vector Machines for Pattern Recognition.
- Cristianini, Shawe-Taylor, Suanders. Kernel Methods: A Paradigm for Pattern Analysis. Kernel Methods in Bioengineering, Signal and Image Processing. 2007.
- Schölkopf, Bernhard. Statistical Learning and Kernel Methods.
- Schölkopf, Bernhard. The Kernel Trick for Distances.
- Martin Law : SVM lecture for CSE 802 CS department MSU.
- Andrew Moore: “Support vector machines” CS school CMU.
- Vikramaditya Jakkula : “Tutorial on Support vector machines” school of EECS Washington State University .

Resources



http://www.kernel-machines.org

http://www.support-vector.net/

http://www.support-vector.net/icml-tutorial.pdf

http://www.kernel-machines.org/papers/tutorial-nips.ps.gz

http://www.clopinet.com/isabelle/Projects/SVM/applist.html

http://www.cs.cornell.edu/People/tj/

http://svmlight.joachims.org/

Resources in Matlab

- Mathworks “Train support vector machine classifier”. <http://www.mathworks.com/help/toolbox/bioinfo/ref/svmtrain.html> (4/6/2011)

- “Support vector machine”. <http://en.wikipedia.org/wiki/Support_vector_machine> (4/6/2011)

- Jason Weston, “Support Vector Machine (and Statistical Learning Theory) Tutorial”, NEC Labs America.