Support Vector Machines

Prof. G Panda,
FNAE,FNASc,FIET(UK)
IIT Bhubaneswar

Outline

- Introduction
- Maximum Margin/ Linear Separable classification
- Example
- Soft margin/Linear Non-Separable classification
- Non-Linear soft margin classification.
- Pros and cons.

Introduction

- Support Vector Machine(SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges.
- Mostly used in classification problems.
- An SVM model is a representation of the examples as points in space,so that the examples of the separate categories are divided by a clear gap that is as wide as possible.

SVM: Binary classification

It can be viewed as separating task in feature space.

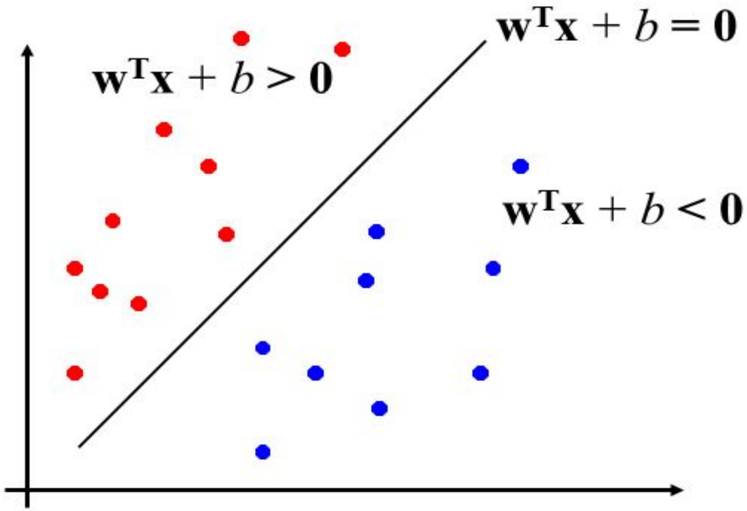
The idea is to find the equation of a line $\mathbf{w}^T \mathbf{x} + \mathbf{b} = \mathbf{0}$ that divides the set of examples in the target classes.

w: decision hyperplane normal vector

x: data points

b: bias

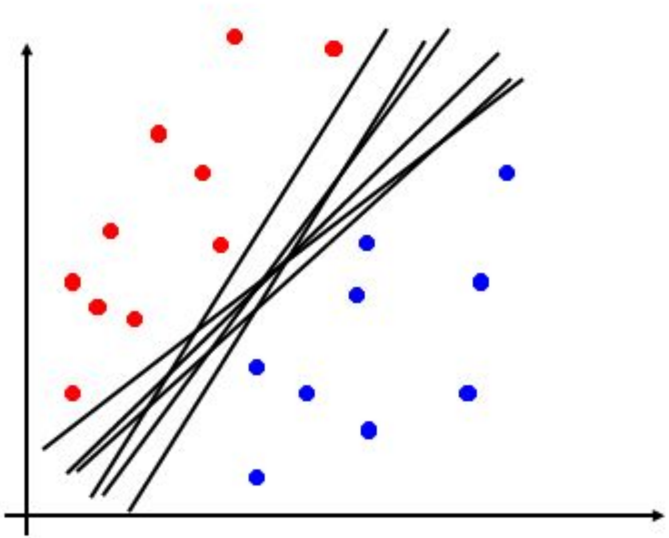
SVM: Binary classification



$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$

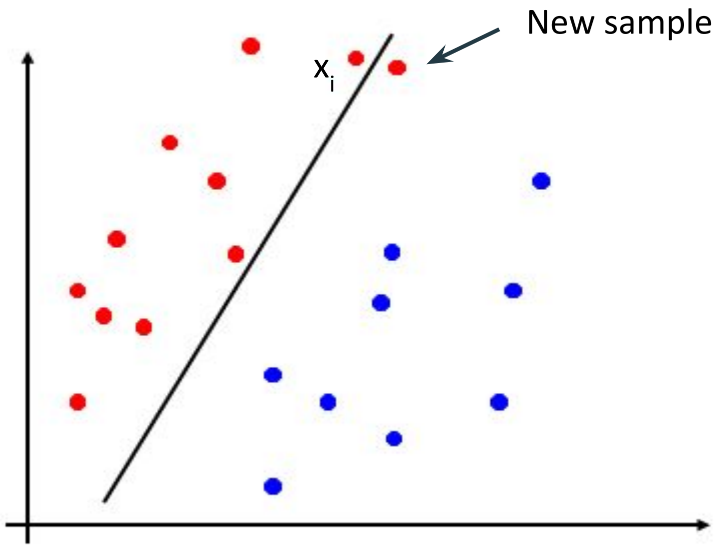
Class = -1 ; if f(x) = -ve
1 ; if f(x) = +ve

Linear Separators



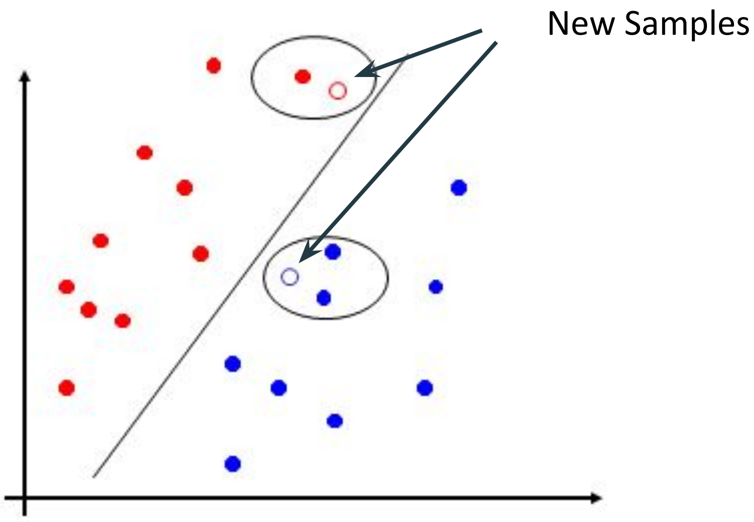
- Any line that separates two classes can be answer to the problem
- Which line to choose?
- Which will be the best separating line?

Linear separators



- Suppose hyperplane is close to sample x_i .
- If we see new sample close to sample 'i', it is likely to be on the wrong side of the hyperplane.
- Poor generalization (performance on unseen data)

Linear separators

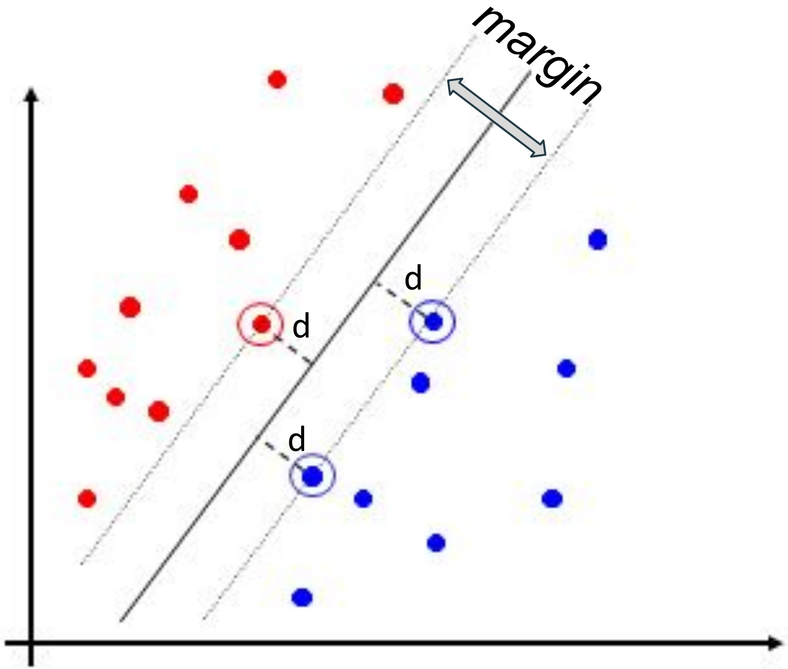


- Hyperplanes as far as possible from any sample.
- New samples close to the old samples will be classified correctly.
- Good generalization.

Maximum Margin Classification

- **Aim** : To find support vector algorithm for the separating hyperplane with the largest margin.
- For the optimal hyperplane, distance to the closest negative sample(class = -1) equal to distance from the closes positive sample(class = +1).
- This means that only the nearest instances to the separator matter.

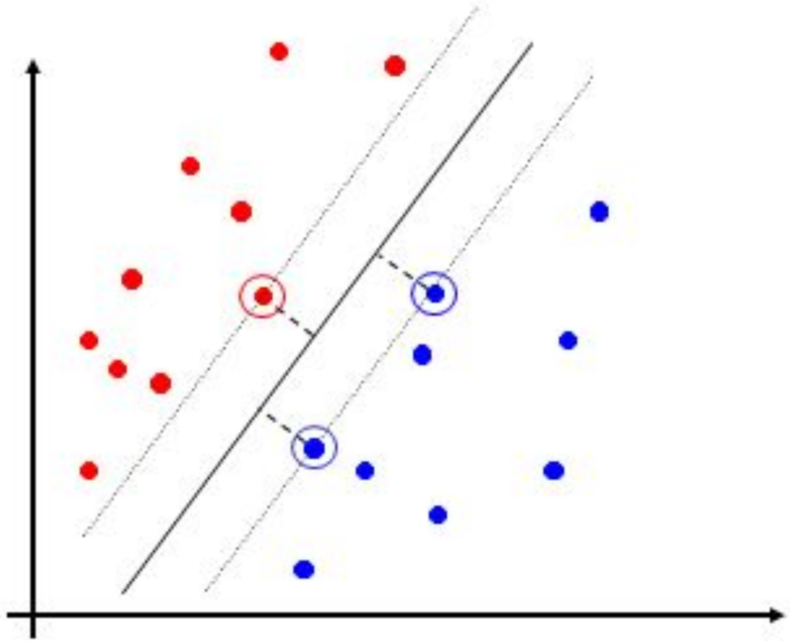
Maximum Margin Classification



Margin is twice the absolute value of distance d of the closest example to the separating hyperplane.

margin = 2d

Maximum Margin Classification



- The decision function is fully specified by a subset of training samples, *the support vectors*.
- Support vectors are the samples closest to the separating hyperplane.
- Optimal hyperplane is completely defined by support vectors

Maximum Margin Classification

Let the training set i : $\{x_i, y_i\}$, i= 1,2,3…..n , $y_i \in \{-1, +1\}$ $x_i \in \mathbf{R}^d$ .The points lie on hyperplane if

$$\mathbf{w}^T\mathbf{x}_i + b = 0$$

Where, W is normal to separating plane

Perpendicular distance from origin to hyperplane = $\frac{b}{||w||}$

Where, $||w||$ Euclidean norm of w.

Maximum Margin Classification

d_+ or d_- : distance from closest positive or negative sample from hyperplane

Margin = $d_+ + d_-$

Distance from sample x_i to the separator is

$$r = \frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|}$$

Maximum Margin Classification

Basic Solution approach :

Assume that all data is at least distance 1 from the hyperplane, then the following two constraints follow for a training set

$$\mathbf{w}^T\mathbf{x}_i + b \geq 1 \qquad \text{if } y_i = 1$$

$$\mathbf{w}^T\mathbf{x}_i + b \leq -1 \qquad \text{if } y_i = -1$$

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

Maximum Margin Classification

For Support vectors, the inequality becomes the equality i.e. support vectors of positive samples lie on the hyperplane H1

i.e., H_1 : $\mathbf{w}^T\mathbf{x}_i + b = 1$, with perpendicular distance from origin = $\frac{|1-b|}{||w||}$

Support vectors of positive samples lie on the hyperplane H_2

I.e., H_2 : $\mathbf{w}^T\mathbf{x}_i + b = -1$, with perpendicular distance from origin = $\frac{|-1-b|}{||w||}$

Maximum Margin Classification

Hence, $d_+ = d_- = \frac{1}{||w||}$

$\therefore$ Margin $= d_+ + d_- = \frac{1}{||w||} \quad \frac{1}{||w||} \quad \frac{2}{||w||}$

Remarks:

- H_1 and H_2 are two parallel hyperplanes (same normal).
- No training data points fall between H_1 and H_2

Maximum Margin Classification

The goal is to find pair of hyperplanes which gives the maximum margin and since margin is inversely proportional to $||w||$ we need to minimize $||w||^2$

The formulation of the optimization problem is:

Minimize : ½ $||w||^2$

subject to $y_i(w^T x_i + b) \geq 1, \forall i$

Lagrangian formulation

Why?

- Constraints in equation $y_i (w^T x_i + b) \geq 1, \forall i$,will be replaced by constraints on Lagrangian multipliers themselves, hence would become much easier to handle the constraints.
- It can be generalized to non-linear case

Lagrangian primal formulation

$$L_p = \underbrace{\frac{1}{2}\|w\|^2}_{\text{Objective function}} + \sum_{i=1}^{l} \alpha_i \underbrace{\left(1 - y_i\left(w^T x_i + b\right)\right)}_{\text{constraint}}$$

$\alpha_i \geq 0$, $\forall i$ are Lagrange multipliers

l is number of input values

This is the Lagrange **primal formulation** of optimization problem..

Lagrangian primal formulation

- Now we need to minimize L_p with respect to 'w' and 'b' and simultaneously required that derivatives of L_p w.r.t. all α_i vanish, all subject to constraints $\alpha_i \geq 0$.

$$\min_{\forall w,b} L_P$$

- The objective function ½ $||w||^2$ s convex, hence this is convex quadratic programming problem(since objective function is quadratic equation subject to constraints)

Lagrangian dual formulation

Since it is a convex quadratic programming problem, we can equivalently solve the following **“dual ” problem**.

Maximize L_p such that gradients of L_p w.r.t. W and b vanish such that $\alpha_i \geq 0$.

Property: Maximum of L_p w.r.t constraints occur at same values of W, b and α

as the minimum of L_p w.r.t constraints.

Lagrangian dual formulation

Requiring the gradient of L_p w.r.t. W and b vanish i.e., If we compute the derivative of L_p with respect to w and b and we set them to zero gives the following condition.

Derivation of L_p w.r.t w : $w + \sum_{i=1}^{n} \alpha_i(-y_i)x_i = 0 \Rightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i$

Derivation of L_p w.r.t b = $\sum_{i=1}^{n} \alpha_i y_i = 0$

Lagrangian dual formulation

Substitute those values into Lagrangian equation of primal L_p.

$$
\begin{aligned}
\mathcal{L} &= \frac{1}{2}\sum_{i=1}^{n}\alpha_i y_i x_i^T \sum_{j=1}^{n}\alpha_j y_j x_j + \sum_{i=1}^{n}\alpha_i\left(1 - y_i\left(\sum_{j=1}^{n}\alpha_j y_j x_j^T x_i + b\right)\right) \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\alpha_i y_i \sum_{j=1}^{n}\alpha_j y_j x_j^T x_i \\
&\quad -b\sum_{i=1}^{n}\alpha_i y_i \quad (and \quad given \quad that \quad \sum_{i=1}^{n}\alpha_i y_i = 0) \\
&= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n}\alpha_i \quad (rearranging \quad terms)
\end{aligned}
$$

Lagrangian dual formulation

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

Remarks:

- L_p and L_D are different formulation, they arise with same objective function but with different constraints.
- This formulation only depends on the α_i
- Solution: minimize L_p or maximize L_D.

Lagrangian dual formulation

Formulation of Dual problem

$$\text{Maximize } \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

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

This a Quadratic Programming problem (QP)

This means that the parameters form a parabolloidal surface, and an optimal can be found(since it is convex function).

Lagrangian dual formulation

W can be obtained from

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

b can be obtained from a positive support vector (x_+) knowing that

$w^T x_i + b = 1$ or

Can be obtained from a negative support vector (x_-) knowing that

$w^T x_i + b = -1$

Lagrangian dual formulation

- Many of the α_i are zero.
  - w is a linear combination of a small number of examples
  - We obtain a sparse representation of the data.
- The examples x_i with non zero α_i are the support vectors (SV).
- The vector of parameters can be expressed as:

$$w = \sum_{\forall i \in SV} \alpha_i y_i x_i$$

SVM training phase

Maximize L_D w.r.t. α_i such that $\alpha_i \geq 0$ with the solution $w = \sum_{i=1}^{n} \alpha_i y_i x_i$

In the solution those points for which $\alpha_i > 0$ are support vectors and they lie on Hyperplanes H_1 or H_2.

For other points $\alpha_i = 0$ and lie on the either sides of H_1 or H_2 such that

$$y_i(w^T x_i + b) > 1, \forall i$$

SVM test phase

In order to classify a new instance z we just have to compute.

$$sign(w^T z + b) = sign(\sum_{\forall i \in SV} \alpha_i y_i x_i^T z + b)$$

This means that w does not need to be explicitly computed

SVM:Example

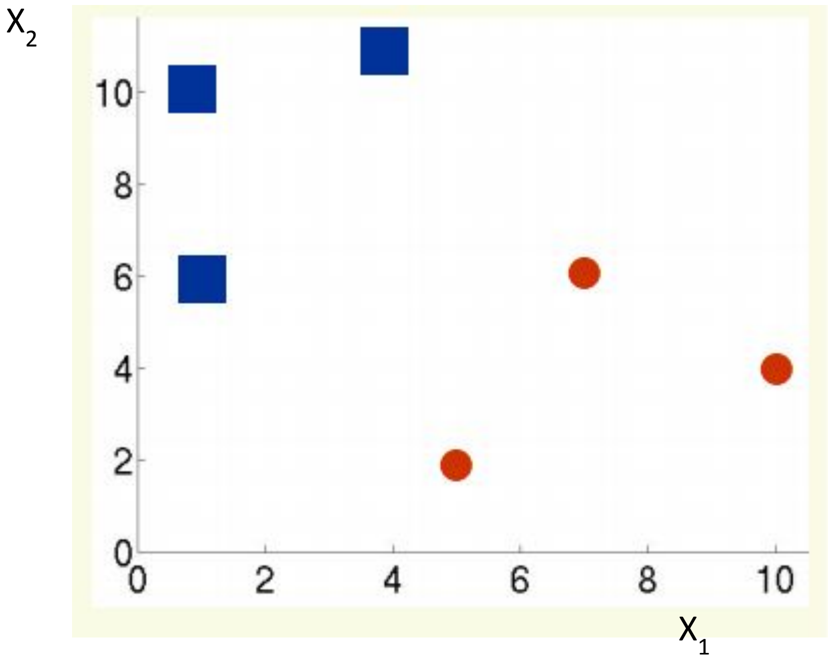
Given data :

Class-1 : [1,6] ,[1,10],[4,11]

Label y_i = 1

Class-2 : [5.2],[7,6],[10,4]

Label y_i = -1



SVM:Example

To represent all inputs consider array X

$$X = \begin{pmatrix} 1 & 6 \\ 1 & 10 \\ 4 & 11 \\ 5 & 2 \\ 7 & 6 \\ 10 & 4 \end{pmatrix} \quad \text{and labels } Y = \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix}$$

SVM:Example

$$Y^T.Y = \begin{pmatrix} 1 & 1 & 1 & -1 & -1 & -1 \\ 1 & 1 & 1 & -1 & -1 & -1 \\ 1 & 1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 & 1 \end{pmatrix}$$

$$X.X^T = \begin{pmatrix} 37 & 61 & 70 & 17 & 43 & 34 \\ 61 & 101 & 114 & 25 & 67 & 50 \\ 70 & 114 & 137 & 42 & 94 & 84 \\ 17 & 25 & 42 & 29 & 47 & 58 \\ 43 & 67 & 94 & 47 & 85 & 94 \\ 34 & 50 & 84 & 58 & 94 & 116 \end{pmatrix}$$

SVM:Example

Therefore Matrix H with $H_{ij} = y_i y_j x_i^T x_j$ can be written as

$$H = Y^TY.X^TX = \begin{pmatrix} 37 & 61 & 70 & -17 & -43 & -34 \\ 61 & 101 & 114 & -25 & -67 & -50 \\ 70 & 114 & 137 & -42 & -94 & -84 \\ -17 & -25 & -42 & 29 & 47 & 58 \\ -43 & -67 & -94 & 47 & 85 & 94 \\ -34 & -50 & -84 & 58 & 94 & 116 \end{pmatrix}$$

SVM:Example in Matlab

Matlab expects quadratic programming to be stated in the canonical (standard) form which is

minimize $\mathbf{L_D(\alpha) = f^T \alpha + \frac{1}{2} \alpha^T H \alpha}$ constrained to $\mathbf{A\alpha \leq a}$ and $\mathbf{B\alpha = b}$.

where A,B,H are n by n matrices and f, a, b are vectors

Need to convert our optimization problem to canonical form

Maximize $$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}^t H \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}$$ constrained to

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad \alpha_i \geq 0 \ \forall i$$

SVM:Example in Matlab

Multiply by –1 to convert to minimization: $L_D(\alpha) = -\sum_{i=1}^{n} \alpha_i + \frac{1}{2}\alpha^t H\alpha$

Let $f = \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix}$ = -ones(6,1), then we can write

Minimize $L_D(\alpha) = f^T \alpha + ½\ \alpha^T H\ \alpha$

First constraint is $\alpha i \geq 0\ \forall i$

$A = \begin{bmatrix} -1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -1 \end{bmatrix}$ = -eye(6), $a = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$ = zeros(6,1)

Rewrite the first constraint in canonical form : $A\alpha \leq a$

SVM:Example in Matlab

Our Second constraint is $\sum_{i=1}^{n} \alpha_i y_i = 0, \quad \alpha_i \geq 0 \;\; \forall i$

Let B = [[Y] ; [zeros(5,6)]

and b = $\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$ = zeros(6,1)

Second constraint in canonical form is **Bα = b.**

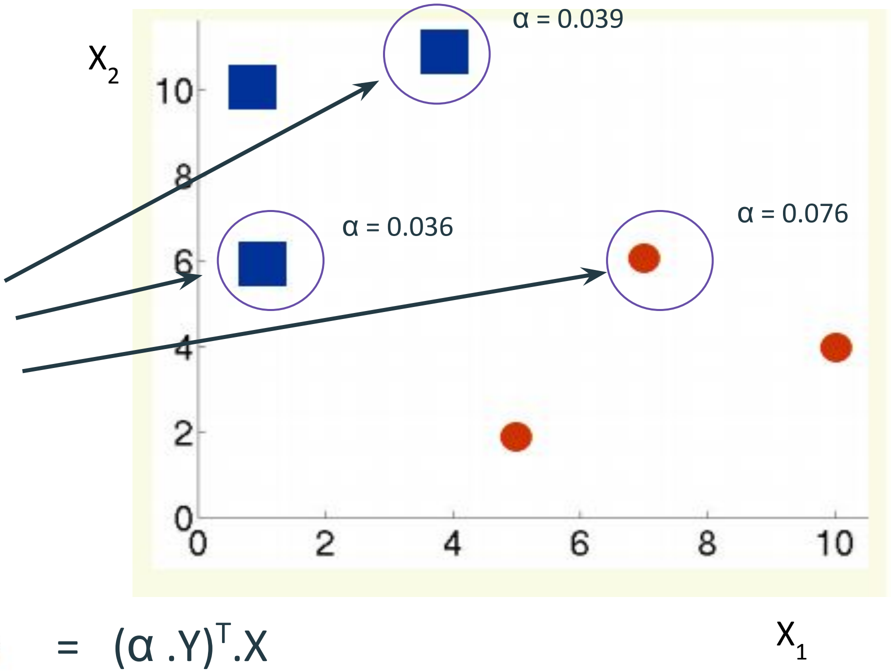
α = quadprog(H, f, A, a, B, b) %%in matlab

SVM:Example

Solution

$$\alpha = \begin{pmatrix} 0.036 \\ 0 \\ 0.039 \\ 0 \\ 0.076 \\ 0 \end{pmatrix}$$

Support Vectors



Find W using $w = \sum_{\forall i \in SV} \alpha_i y_i x_i \quad = \quad (\alpha\ .Y)^T.X$

SVM:Example

$\alpha .Y =$ [0.036 , 0 , 0.04 , -0 , -0.076 , -0]

$W = (\alpha .Y)^T.X = [-0.33 , 0.20] \Rightarrow w_1 = -0.33 \quad w_2 = 0.20$

We can find b using $w^T x_i + b = 1$ for positive support vector i =1

$b = 1 - w^T x_1 = 1 -(-0.33x1 + 6x0.20) = 1 -(-0.33 + 1.20)$

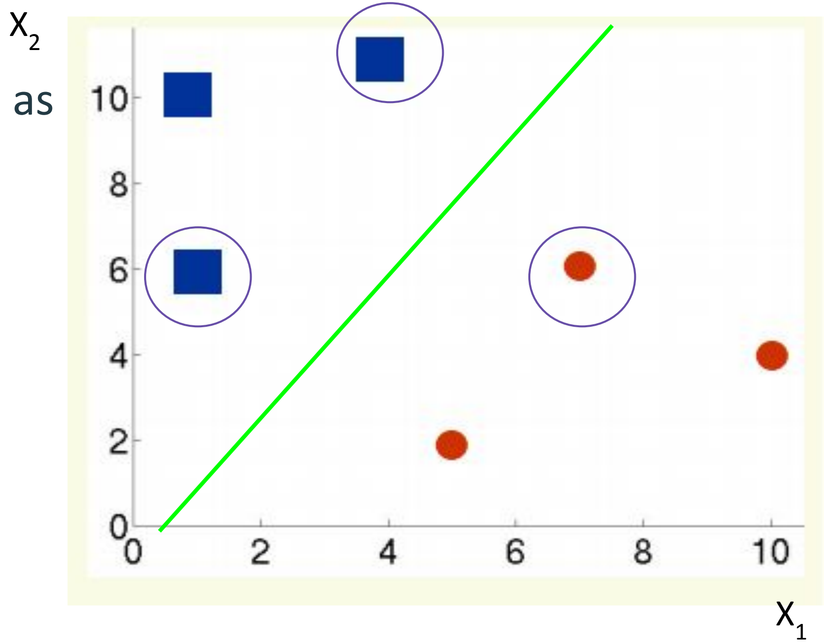
$= 0.13$

SVM:Example

Equation of the line can be written as

$-0.33\,X_1 + 0.20\,X_2 + 0.13 = 0$

(since $w^T x_i + b = 0$)



SVM:Example

Testing Phase:

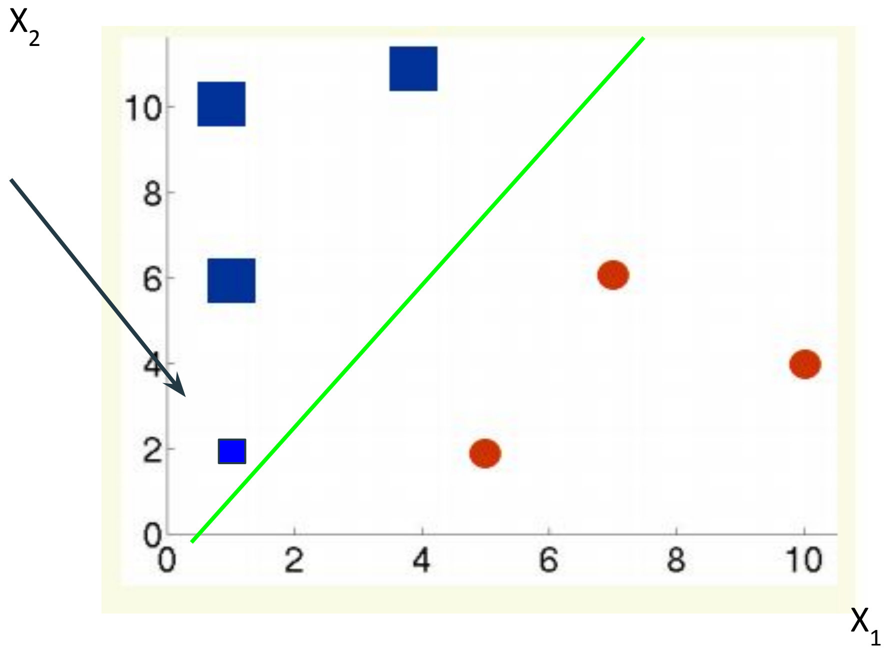
Suppose give a new point z = [1,2]

Find $sign(w^T z + b)$

sign(-0.33x1 0.20x2 + 0.13)

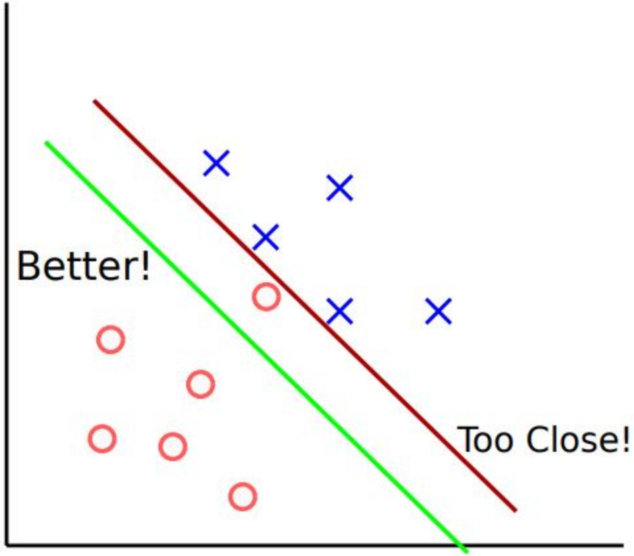
sign(-0.07 + 0.13) = positive

Therefore z is in class 1



SVM:Linear Non-Separable

- Sometimes data has errors and we want to ignore them to obtain a better solution
- Sometimes data is just non linearly separable
- We can obtain better linear separators being less strict.

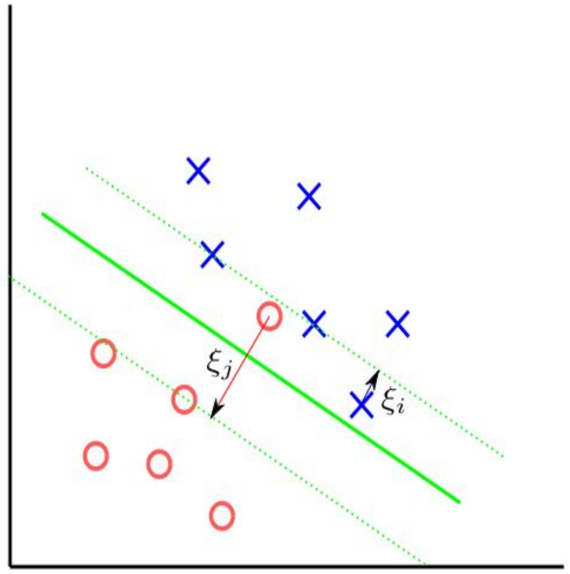


SVM : Soft Margin classification

- We want to be permissive for certain examples, allowing that their classification by the separator diverge from the real class.
- This can be obtained by adding to the problem what is called slack variables (ξ_i)
- This variables represent the deviation of the examples from the margin.
- Doing this we are relaxing the margin, we are using a soft margin

SVM : Soft Margin classification

- We are allowing an error in classification based on the separator $w^T x_i + b$.
- The values of ξ_i approximate the number of misclassifications.
- ξ_i is a measure of deviation from the ideal for sample i.
  - ξ_i>1 sample i is on the wrong side of the separating hyperplane.
  - 0< ξ_i <1 sample i is on the right side of separating hyperplane but within the region of maximum margin.
  - ξ_i < 0 is the ideal case for sample i



SVM : Soft Margin classification

In order to minimize the error, we can minimize $\sum_i \xi_i$ introducing the slack variables to the constraints of the problem:

$$
\begin{array}{ll}
w^T x_i + b \geq 1 - \xi_i & y_i = 1 \\
w^T x_i + b \geq -1 + \xi_i & y_i = -1 \\
\xi_i \geq 0 &
\end{array}
$$

ξ_i = 0 if there are no errors (linearly separable problem)

SVM : Primal Optimization Problem

We need to introduce this slack variables on the original problem, we have now:

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

Subject to $y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \forall i, \xi_i \geq 0$

Now we have an additional parameter C that is the tradeoff between the error and the margin. If C is small, we allow a lot of samples not in ideal position and if C is large, we want to have very few samples not in ideal position. Smaller c larger margin and vice versa.

SVM : Dual Optimization problem

Performing the transformation to the dual problem we obtain the following:

We can recover the solution as

Maximize: $\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$

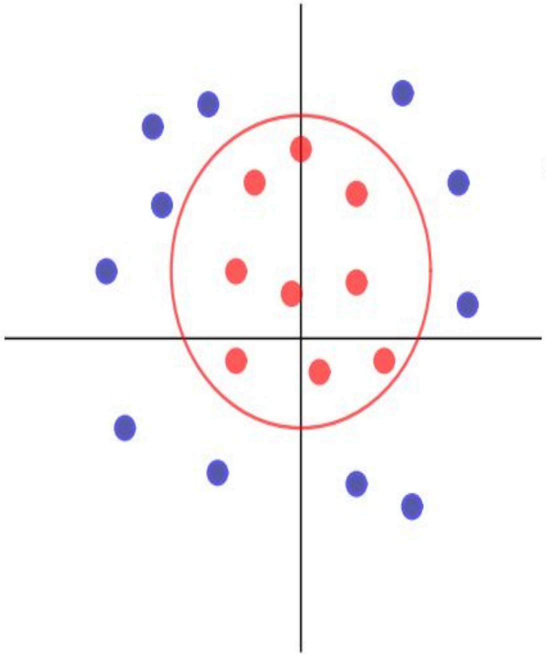
Subject to $\sum_{i=1}^{n} \alpha_i y_i = 0, \quad C \geq \alpha_i \geq 0 \ \ \forall i$

This problem is very similar to the linearly separable case, except that there is a upper bound C on the values of α_i.

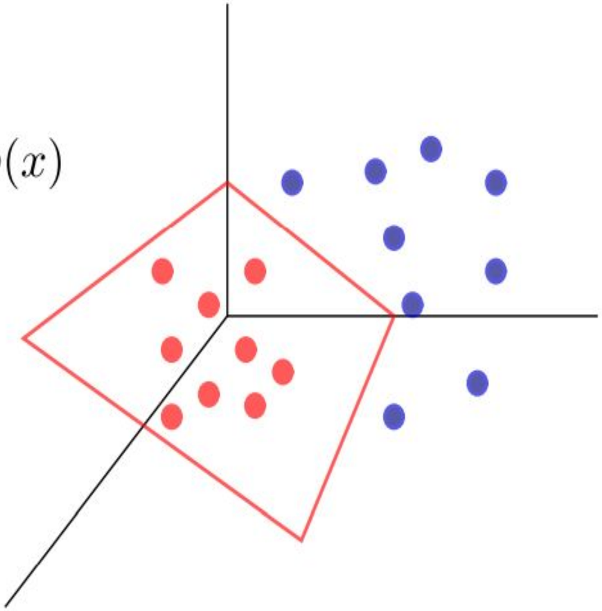
SVM: Non-linear classification

- So far we have only considered large margin classifiers that use a linear boundary.
- In order to have better performance we have to be able to obtain non-linear boundaries.
- The idea is to transform the data from the input space (the original attributes of the examples) to a higher dimensional space using a function $\varphi(x)$.
- This new space is called the feature space .
- The advantage of the transformation is that linear operations in the feature space are equivalent to non-linear operations in the input space.

Feature Space transformation



$$\phi : x \Rightarrow \varphi(x)$$



XOR - Problem

The XOR problem is not linearly separable in its original definition, but we can make it linearly separable if we add a new feature $x_1 \cdot x_2$

x_1	x_2	$x_1 \cdot x_2$	x_1 XOR x_2
0	0	0	1
0	1	0	0
1	0	0	0
1	1	1	1

The linear function $h(x) = 2x_1x_2 - x_1 - x_2 + 1$ classifies correctly all the examples.

Transforming the data

- Working directly in the feature space can be costly.
- We have to explicitly create the feature space and operate in it.
- We may need infinite features to make a problem linearly separable.
- We can use what is called the **Kernel trick**

The Kernel trick

- In the problem that define a SVM only the inner product of the examples is needed(Data in training algorithm is in the form of dot products $X_i.X_j$).
- This means that if we can define how to compute this product in the feature space, then it is not necessary to explicitly build it.
- There are many geometric operations that can be expressed as inner products, like angles or distances

The Kernel trick

Mapping of inner product to higher dimension

$$X_i^T.X_j \;\rightarrow\; \phi(X_i)^T.\phi(X_j)$$

We can define the kernel function as:

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

We would only need to use "k" in the training algorithm and never even required to know what is $\phi(.)$

The Kernel Example

We can show how this kernel trick works in an example. Lets assume a feature space defined as:

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

A inner product in this feature space is:

$$\left\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) \right\rangle = (1 + x_1 y_1 + x_2 y_2)^2$$

So, we can define a kernel function to compute inner products in this space as:

$K(x, y) = (1 + x_1 y_1 + x_2 y_2)^2$

and we are using only the features from the input space.

The dual problem

In all the previous equations we replace $X_i^T.X_j$ with $K(X_i, X_j)$

Due to the introduction of the kernel function, the optimization problem has to be modified:

Maximize: $$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

Subject to $$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad C \geq \alpha_i \geq 0 \ \ \forall i$$

Non-Linear Classification

To classify new samples

$$h(z) = sign\left(\sum_{\forall i \in SV} \alpha_i, y_i k(x_i, z) + b\right)$$

Example:

Polynomial kernel of degree d : $K(x, y) = (x^T y + 1)^d$

Gaussian kernel with width σ : $K(x, y) = \exp(-||x - y||^2 / 2\sigma^2)$

Pros and Cons

Pros:

- objective function has no local minima .
- Complexity of the classifier is characterized by the number of support vectors rather than the dimensionality of the transformed space.

Cons:

- tends to be slower than other methods .
- quadratic programming is computationally expensive.

References

- An Introduction to Support Vector Machines and Other Kernel-based Learning ,2000,Book by John Shawe-Taylor and Nello Cristianini.
- Support Vector Machines,12 August 2008,Book by Ingo Steinwart
- Knowledge Discovery with Support Vector Machines, 2009,Book by Lutz H. Hamel

Thank you