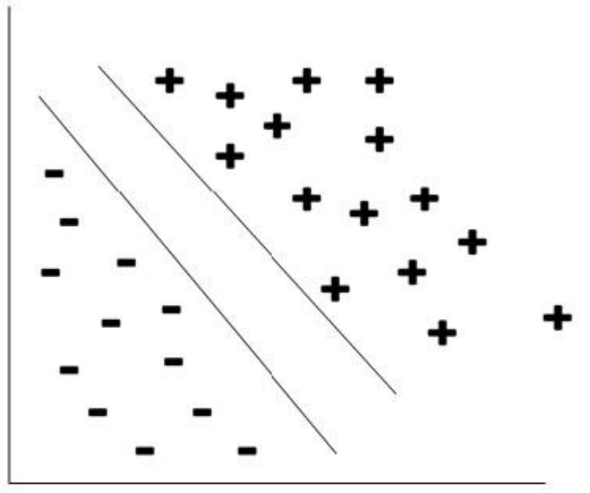
Support Vector Machines (Non-Linear)

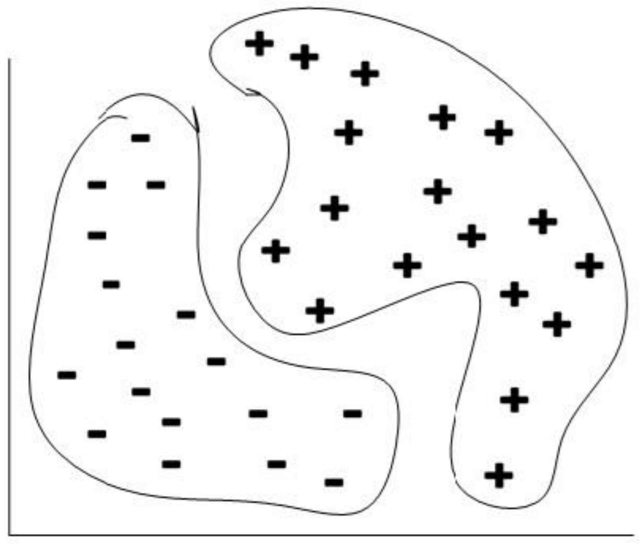
Sachin Tripathi

IIT(ISM), Dhanbad

Introduction

- In general, if data are linearly separable, then there is a Hyperplane otherwise no Hyperplane


(a) Linearly separable

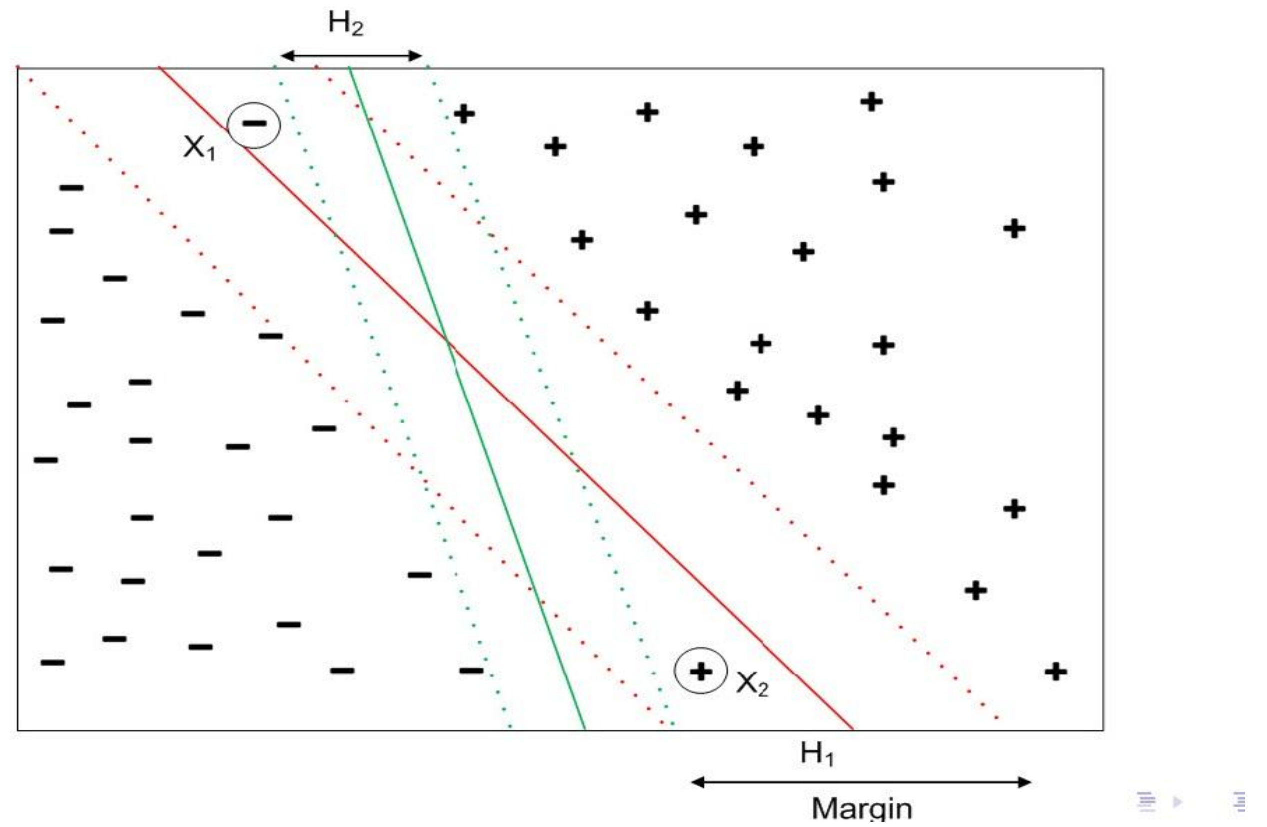

(b) Linearly non-separable

- Such a linearly not separable data can be classified using two approaches.
  1. Linear SVM with soft margin
  2. Non-linear SVM
- In the following, we discuss the extension of linear SVM to classify linearly not separable data.
- Moreover, will discuss non-linear SVM in detail later.

- If the number of training data instances violating linear separability is less, then linear SVM classifier can be used to classify them.

- Suppose, X1 and X2 are two instances.
- We see that the Hyperplane H1 classifies wrongly both X1 and X2.
- Also, we may note that with X1 and X2, we could draw another Hyperplane namely H2, which could classify all training data correctly.
- However, H1 is more preferable than H2 as H1 has higher margin compared to H2 and thus H1 is less susceptible to over fitting.

Problem with linear SVM for Non linearly separable data.



Suppose, X_1 and X_2 are two instances.

We see that the hyperplane H_1 classifies wrongly both X_1 and X_2.

Also, we may note that with X_1 and X_2, we could draw another hyperplane namely H_2, which could classify all training data correctly.

However, H_1 is more preferable than H_2 as H_1 has higher margin compared to H_2 and thus H_1 is less susceptible to over fitting.

- In other words, a linear SVM can be refitted to learn a Hyperplane that is tolerable to a small number of non-separable training data.
- The approach of refitting is called soft margin approach (hence, the SVM is called Soft Margin SVM), where it introduces slack variables to the inseparable cases.
- More specifically, the soft margin SVM considers a linear SVM Hyperplane (i.e., linear decision boundaries) even in situations where the classes are not linearly separable.

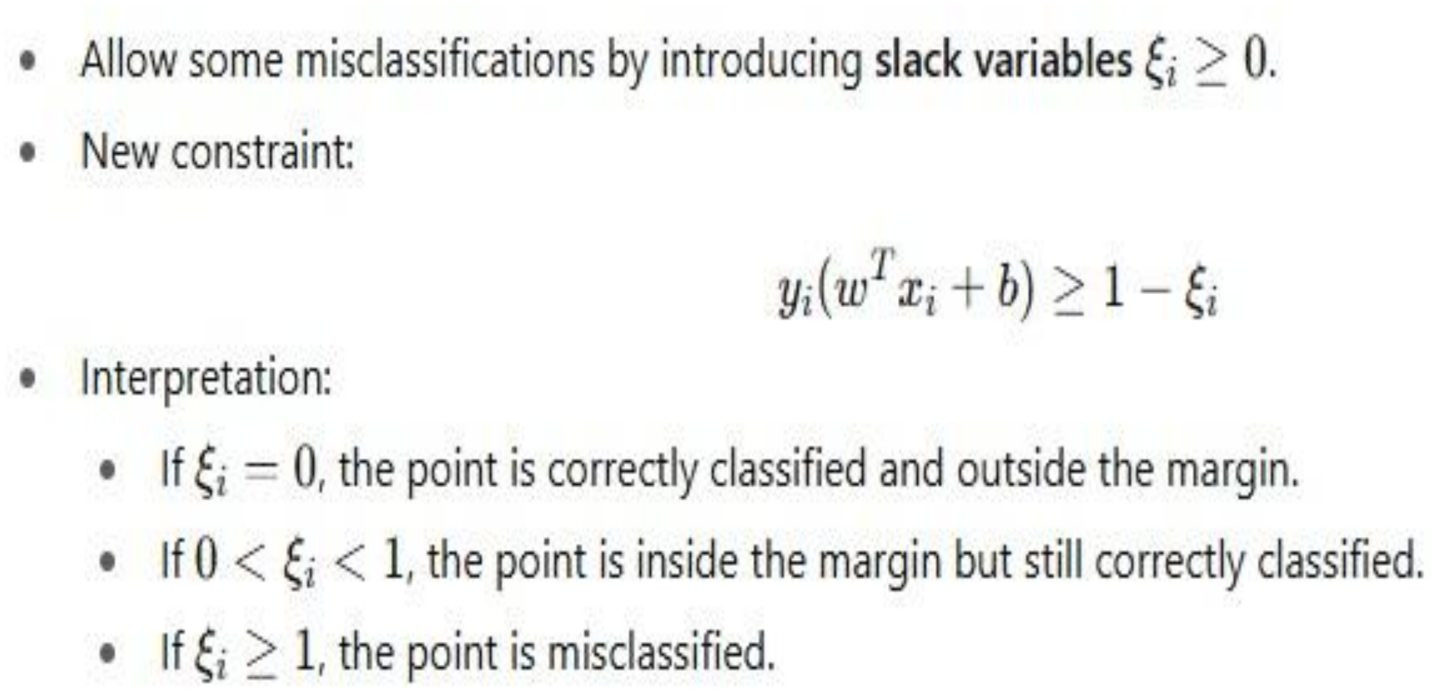
Recall that for linear SVM, we are to determine a maximum margin hyperplane $W.X + b = 0$ with the following optimization:

$$\textit{minimize } \frac{||W||^2}{2}$$

$$\textit{subject to } y_i.(W.x_i + b) \geq 1, i = 1, 2, ..., n$$

In soft margin SVM, we consider the similar optimization technique except that a relaxation of inequalities, so that it also satisfies the case of linearly not separable data.

Soft Margin SVM Idea



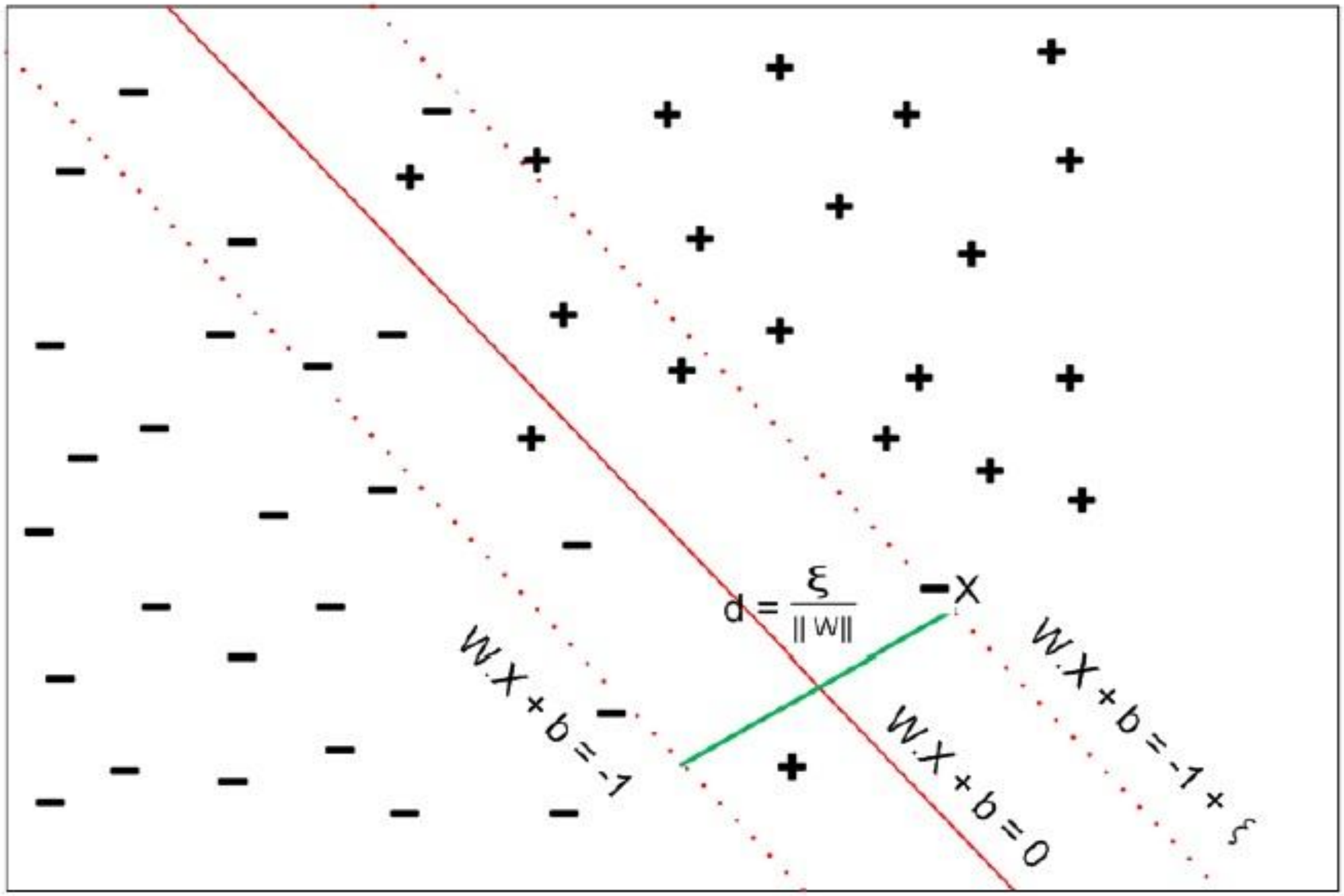
- Allow some misclassifications by introducing **slack variables** $\xi_i \geq 0$.
- New constraint:

$$y_i(w^T x_i + b) \geq 1 - \xi_i$$

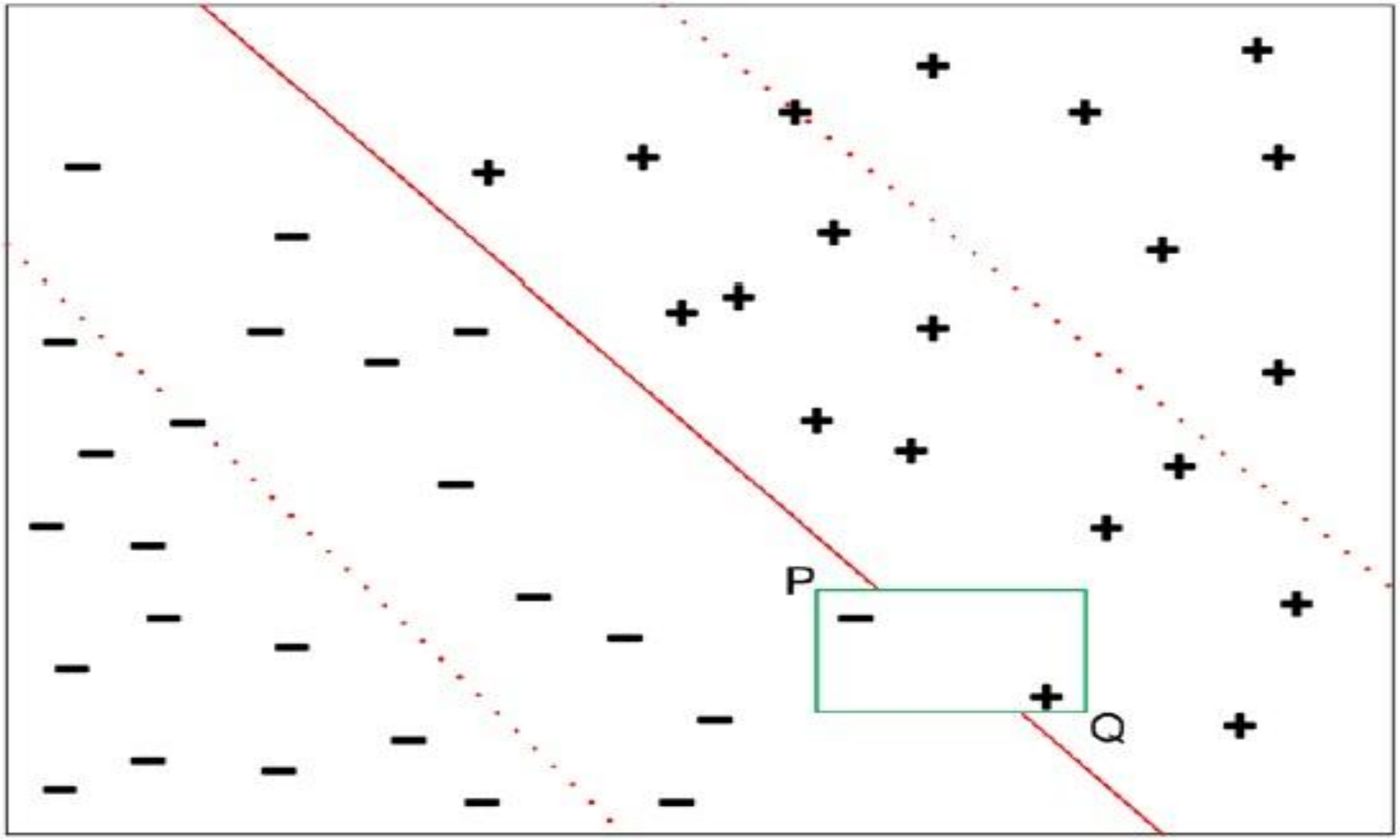
- Interpretation:
  - If $\xi_i = 0$, the point is correctly classified and outside the margin.
  - If $0 < \xi_i < 1$, the point is inside the margin but still correctly classified.
  - If $\xi_i \geq 1$, the point is misclassified.

➢ Thus, in soft margin SVM, we have to calculate W, b and slack variable as a solution to learn SVM.

Interpretation of Slack Variable



- The data X is one of the instances that violates the constraints to be satisfied for linear SVM.
- Thus, $W.X + b = -1 + \xi$ represents a hyperplane that is parallel to the decision boundaries for class - and passes through X.
- It can be shown that the distance between the hyperplanes is $d = \frac{\xi}{||W||}$.
- In other words, ξ provides an estimate of the error of decision boundary on the training example X.


P
Q

To avoid this problem, it is therefore necessary to modify the objective function, so that penalizing for margins with a large gap, that is, large values of slack variables.

The modified objective function can be written as

$$f(W) = \frac{||W||^2}{2} + c.\sum_{i=1}^{n}(\xi_i)^{\phi}$$

where c and ϕ are user specified parameters representing the penalty of misclassifying the training data.

Usually, $\phi = 1$. The larger value of *c* implies more penalty.

We can follow the Lagrange multiplier method to solve the inequality constraint optimization problem, which can be reworked as follows:

$$L = \frac{||W||^2}{2} + c.\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \lambda_i(y_i(W.x_i + b) - 1 + \xi_i) - \sum_{i=1}^{n} \mu_i.\xi_i$$

Here, λ_i's and μ_i's are Lagrange multipliers.
The inequality constraints are:

$$\xi_i \geq 0, \lambda_i \geq 0, \mu_i \geq 0.$$

$$\lambda_i\{y_i(W.x_i + b) - 1 + \xi_i\} = 0$$

$$\mu_i.\xi_i = 0$$

The KKT constraints (in terms of first order derivative of *L* with respect to different parameters) are:

$$\frac{\delta L}{\delta w_j} = w_j - \sum_{i=1}^{n} \lambda_i . y_i . x_{ij} = 0$$

$$w_j = \sum_{i=1}^{n} \lambda_i . y_i . x_{ij} = 0 \ \forall_i = 1, 2....n$$

$$\frac{\delta L}{\delta b} = -\sum_{i=1}^{n} \lambda_i . y_i = 0$$

$$\sum_{i=1}^{n} \lambda_i . y_i = 0$$

$$\frac{\delta L}{\delta \xi_i} = c - \lambda_i - \mu_i = 0$$

$$\lambda_i + \mu_i = c \text{ and } \mu_i.\xi_i = 0 \forall_i = 1, 2, ...n.$$

The above set of equations can be solved for values of $W = [w_1, w_2.......w_m], b, \lambda_i's, \;\mu_i's$ and $\xi_i's$.

Note:

$\lambda_i \neq 0$ for support vectors or has $\xi_i > 0$ and

$\mu_i = 0$ for those training data which are misclassified, that is, $\xi_i > 0$.

Non-Linear SVM

- Linear SVM undoubtedly better to classify data if it is trained by linearly separable data.
- Linear SVM also can be used for non-linearly separable data,provided that number of such instances is less.
- However, in real life applications, number of data overlapping is so high that soft margin SVM cannot cope to yield accurate classifier.
- As an alternative to this there is a need to compute a decision boundary, which is not linear (i.e., not a Hyperplane rather hyper surface).

A hyperplane is expressed as

$$linear: \ w_1x_1 + w_2x_2 + w_3x_3 + c = 0$$

Whereas a non-linear hypersurface is expressed as.

$$Nonlinear: \ w_1x_1^2 + w_2x_2^2 + w_3x_1x_2 + w_4x_3^2 + w_5x_1x_3 + c = 0$$

The task therefore takes a turn to find a nonlinear decision boundaries, that is, nonlinear hypersurface in input space comprising with linearly not separable data.

This task indeed neither hard nor so complex, and fortunately can be accomplished extending the formulation of linear SVM, we have already learned.

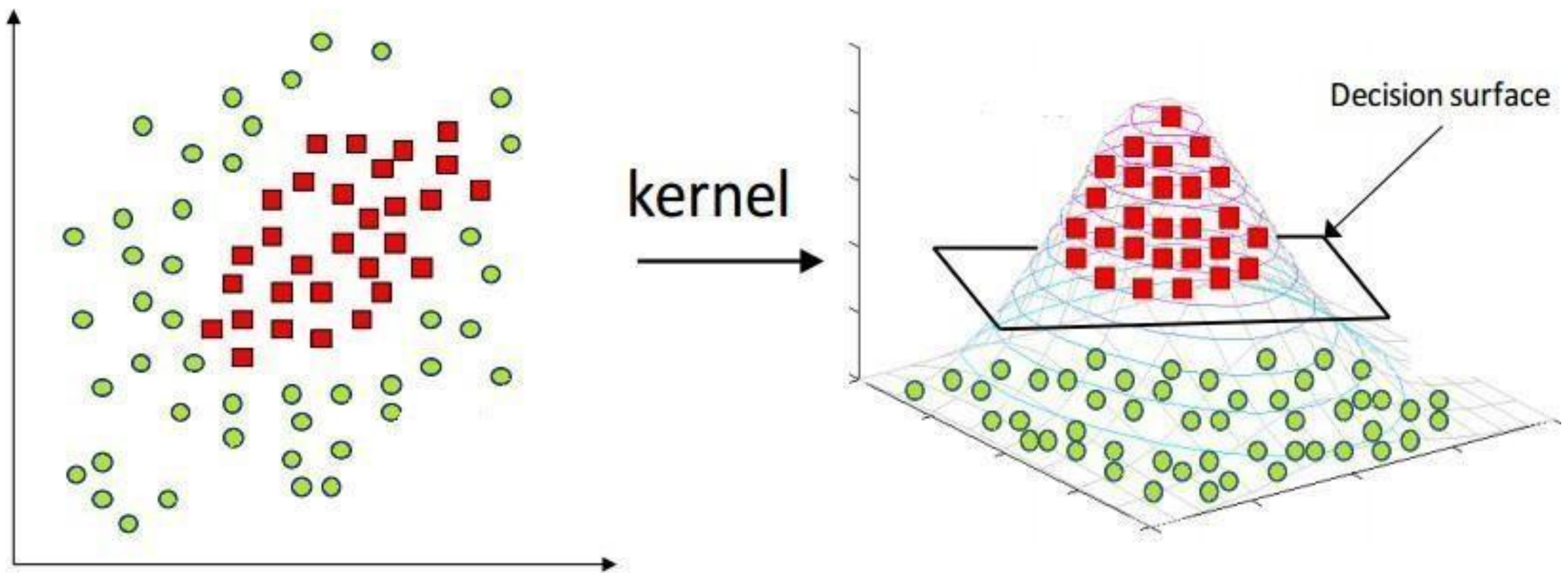
- Note that a linear Hyperplane is expressed as a linear equation in terms of n-dimensional component, whereas a non-linear hyper surface is a non-linear expression.
- This can be achieved in two major steps.

  - Transform the original (non-linear) input data into a higher dimensional space (as a linear representation of data).Note that this is feasible because SVM's performance is decided by number of support vectors not by the dimension of data.

  - Search for the linear decision boundaries to separate the transformed higher dimensional data.

- The above can be done in the same line as we have done for linear

Kernel in SVM (contd)



- In nutshell, to have a nonlinear SVM, the trick is to transform non-linear data into higher dimensional linear data.
- This transformation is popularly called non linear mapping or attribute transformation. The rest is same as the linear SVM.

- In order to understand the concept of non-linear transformation of original input data into a higher dimensional space, let us consider a non-linear second order polynomial in a 3-D input space.

In order to understand the concept of non-linear transformation of original input data into a higher dimensional space, let us consider a non-linear second order polynomial in a 3-D input space.

$$X(x_1, x_2, x_3) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_1^2 + w_5x_1x_2 + w_6x_1x_3 + c$$

The 3-D input vector $X(x_1, x_2, x_3)$ can be mapped into a 6-D space $Z(z_1, z_2, z_3, z_4, z_5, z_6)$ using the following mappings:

$$z_1 = \phi_1(X) = x_1$$

$$z_2 = \phi_2(X) = x_2$$

$$z_3 = \phi_3(X) = x_3$$

$$z_4 = \phi_4(X) = x_1^2$$

$$z_5 = \phi_5(X) = x_1.x_2$$

$$z_6 = \phi_6(X) = x_1.x_3$$

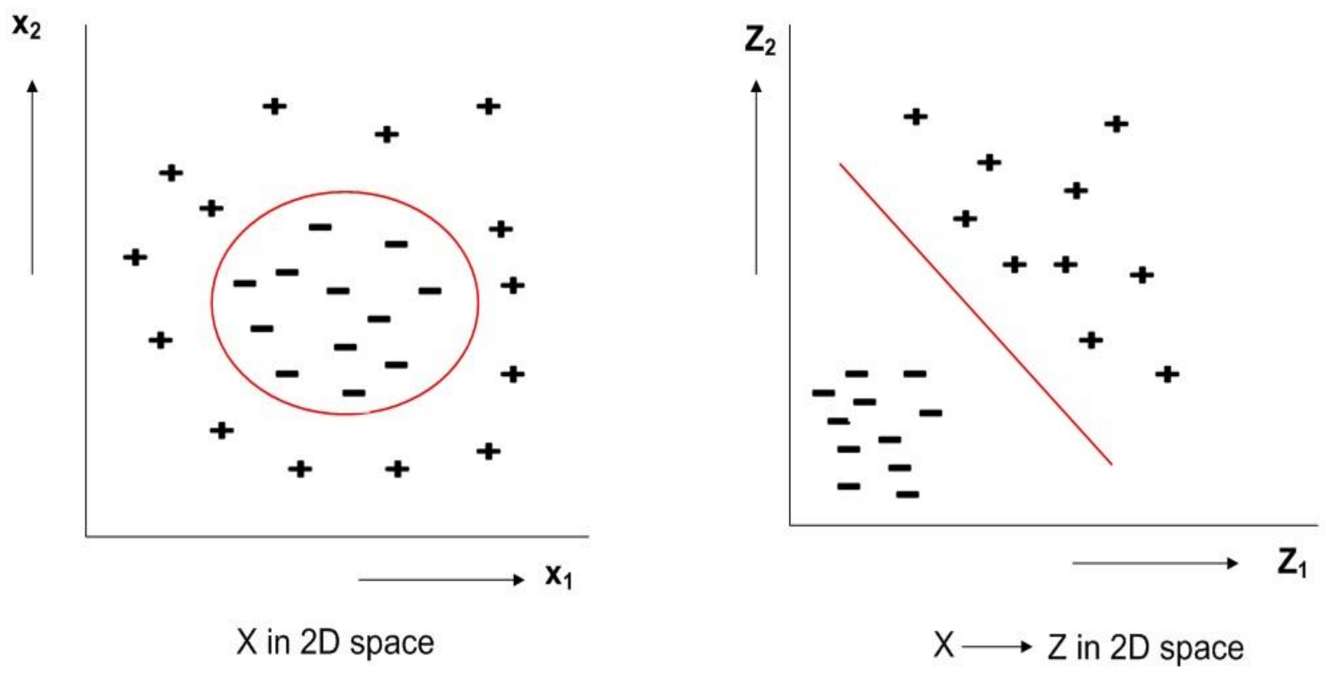
The transformed form of linear data in 6-D space will look like.

$$Z : w_1 z_1 + w_2 z_2 + w_3 z_3 + w_4 z_4 + w_5 z_5 + w_6 x_1 z_6 + c$$

Thus, if Z space has input data for its attributes x_1, x_2, x_3 (and hence Z's values), then we can classify them using linear decision boundaries.

Non-linear mapping to linear SVM

- The below figure shows an example of 2-D data set consisting of class label +1 (as +) and class label -1 (as -).



- We see that all instances of class -1 can be separated from instances of class +1 by a circle. The decision boundary can be written as:

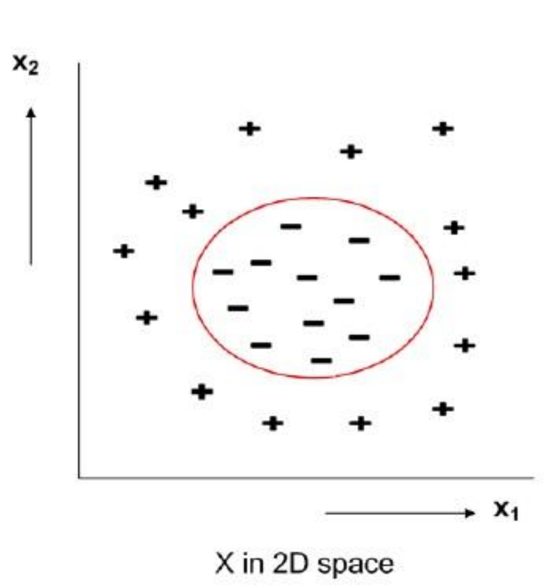
$$X = \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} = 0.2$$

$$\text{or } x_1^2 - x_1 + x_2^2 - x_2 = 0.46$$

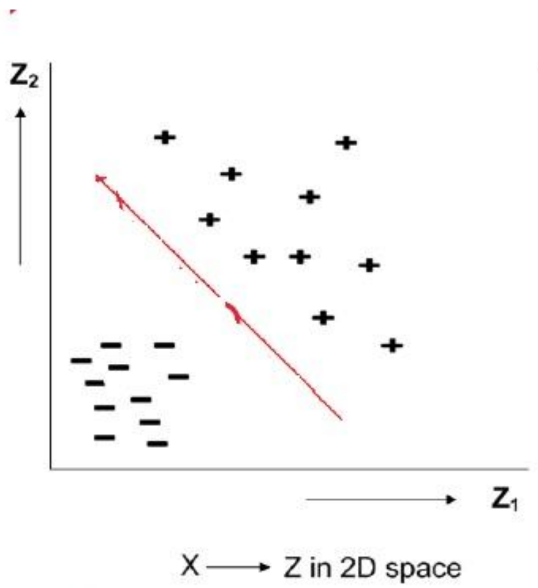
- A non-linear transformation in 2-D space is proposed as follows:

$$Z(z_1, z_2) : \phi\ (x) = x_1^2 - x_1, \phi_2(x) = x_2^2 - x_2$$

- The Z space when plotted will take view as shown below, where data are separable with linear boundary, namely $Z : z_1 + z_2 = -0.46$



X in 2D space



$X \longrightarrow Z$ in 2D space

- The non linear mapping and hence a linear decision boundary concept looks pretty simple but there are many potential problems to do so.

  - Mapping: How to choose the non linear mapping to a higher dimensional space? In fact, the ϕ-transformation works fine for small example but, it fails for realistically sized problems.

  - Cost of mapping: For n-dimensional input instances there exist

$$N_H = \frac{(N+d-1)!}{d!(N-1)!}$$

different monomials comprising a feature space of dimensionality N_H. Here, d is the maximum degree of monomial.

- Dimensionality problem: It may suffer from the curse of dimensionality problem often associated with a high dimensional data.

- More specifically, in the calculation of W.X or Xi.X during testing we need n multiplications and n additions (in their dot products) for each of the n-dimensional input instances and support vectors, respectively.As the number of input instances as well as support vectors are enormously large, it is therefore, computationally expensive.

- Computational cost: Solving the quadratic constrained optimization problem in the high dimensional feature space is too a computationally expensive task.

- Their solution consist of the following:
  - Dual formulation of optimization problem
  - Kernel trick

Dual Formulation of Optimization

- The primal form of the above mentioned inequality constraint optimization problem(according to Lagrange multiplier method) is given by

$$L_p = \frac{||W||^2}{2} - \sum_{i=1}^{n} \lambda_i(y_i(W.x_i + b) - 1)$$

where λ_i's are called Lagrange multipliers.

This L_p is called the primal form of the Lagrangian optimization problem.

The dual form of the same problem can be derived as follows.

To minimize the Lagrangian, we must take the derivative of L_p with respect to W, b and set them to zero.

$$\frac{\delta L_p}{\delta W} = 0 \Rightarrow W = \sum_{i=1}^{n} \lambda_i . y_i . x_i$$

$$\frac{\delta L_p}{\delta b} = 0 \Rightarrow \sum_{i=1}^{n} \lambda_i . y_i = 0$$

$$\|w\|^2 = w^\top w.$$

So the relation is:

$$w = \sum_i \alpha_i y_i x_i,$$

and then

$$\|w\|^2 = w^\top w = \left(\sum_i \alpha_i y_i x_i\right)^\top \left(\sum_j \alpha_j y_j x_j\right).$$

Expanding gives

$$\|w\|^2 = \sum_i \sum_j \alpha_i \alpha_j y_i y_j \, x_i^\top x_j.$$

- From above two equation we get Lagrangian L as

$$L = \sum_{i=1}^{n} \lambda_i + \frac{1}{2} \sum_{i,j} \lambda_i . y_i . \lambda_j . y_j . x_i . x_j - \sum_{i=1}^{n} \lambda_i . y_i \sum_{j=1}^{n} (\lambda_j . y_j . x_j) x_i$$

$$= \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i . \lambda_j . y_i . y_j . x_i . x_j$$

- This form is called the dual form of Lagrangian and distinguishably written as:

$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i . \lambda_j . y_i . y_j . x_i . x_j$$

There are key differences between primal (L_p) and dual (L_D) forms of Lagrangian optimization problem as follows.

- L_p involves a large number of parameters namely W, b and λ_i's. On the other hand, L_D involves only λ_i's, that is, Lagrange multipliers.

- L_p is the minimization problem as the quadratic term is positive. However, the quadratic term in L_D is negative sign, Hence it is turned out to be a maximization problem.

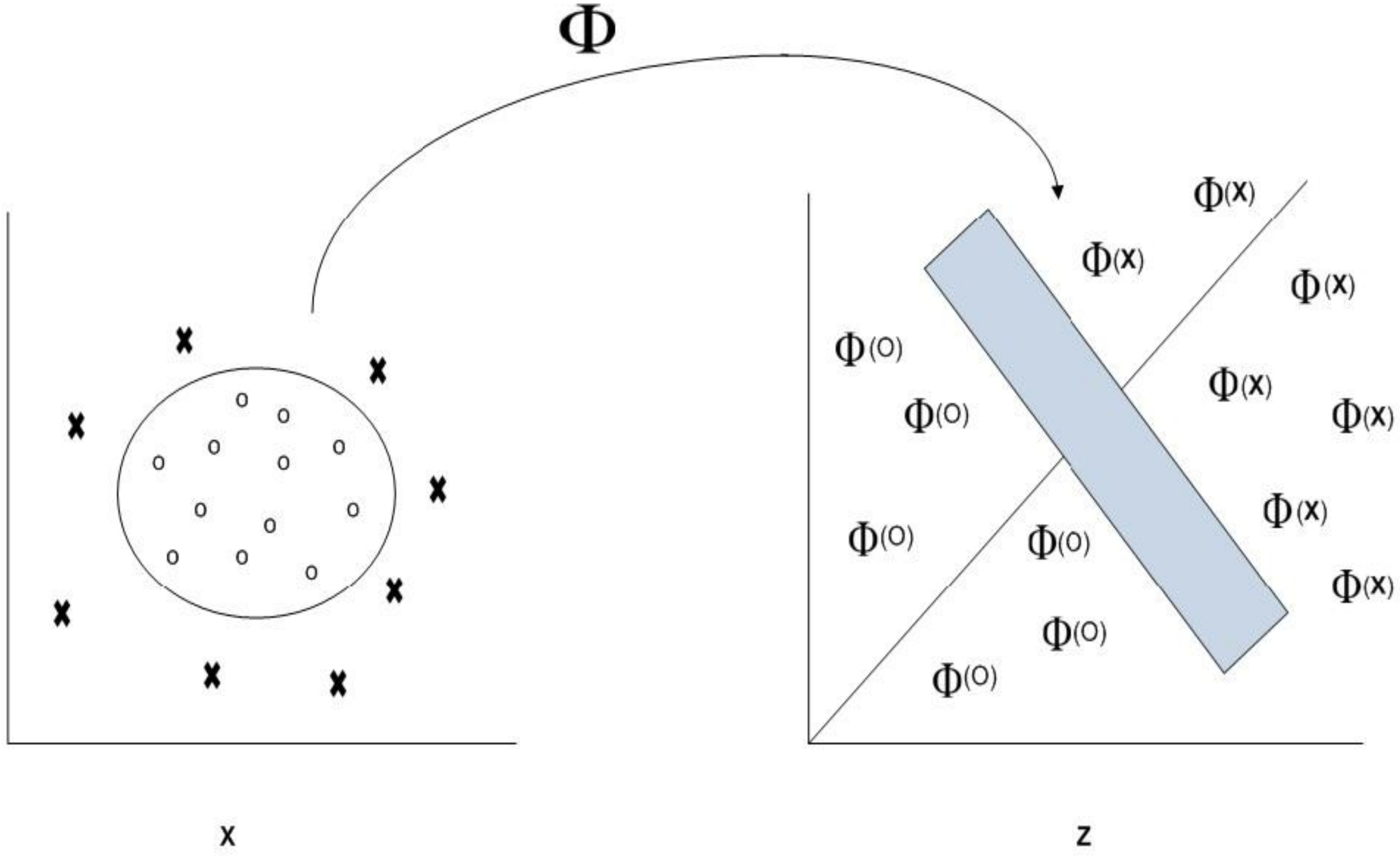
- L_p involves the calculation of $W.x$, whereas L_D involves the calculation of $x_i.x_j$. This, in fact, advantageous, and we will realize it when we learn Kernel-based calculation.

The SVM classifier with primal form is $\delta_p(x) = W.x + b$ with $W = \sum_{i=1}^{n} \lambda_i.y_i.x_i$ whereas the dual version of the classifier is

$$\delta_D(X) = \sum_{i=1}^{m} \lambda_i.y_i.(x_i.x) + b$$

where x_i being the i^{th} support vector, and there are m support vectors. Thus, both L_p and L_D are equivalent.

Kernel Trick



Classifier:

$$\delta(x) = \sum_{i=1}^{n} \lambda_i y_i y_i . x + b$$

$$\delta(z) = \sum_{i=1}^{n} \lambda_i y_i \phi(x_i) . \phi(x) + b$$

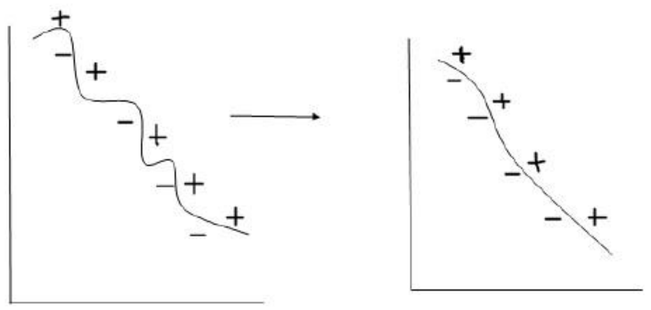
Learning:

$$Maximize \ \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j . y_i . y_j . x_i . x_j$$

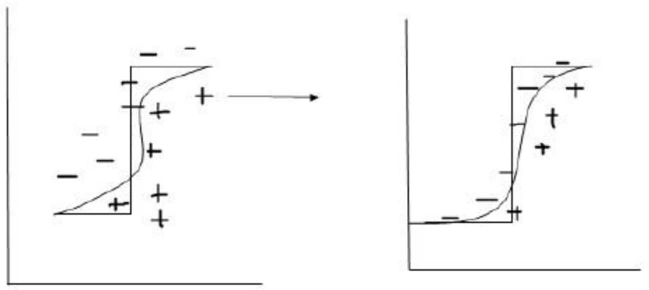
$$Maximize \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j . y_i . y_j \phi(x_i) . \phi(x_j)$$

Subject to: $\lambda_i \geq 0, \ \sum_i \lambda_i . y_i = 0$

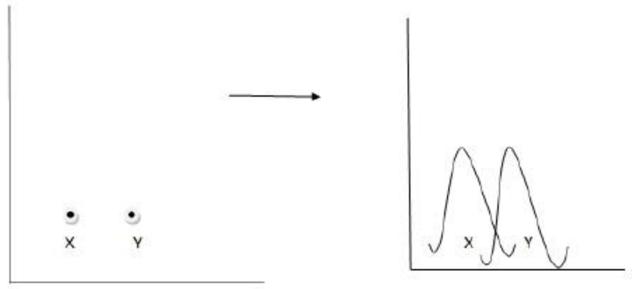
Visual interpretation of few kernel



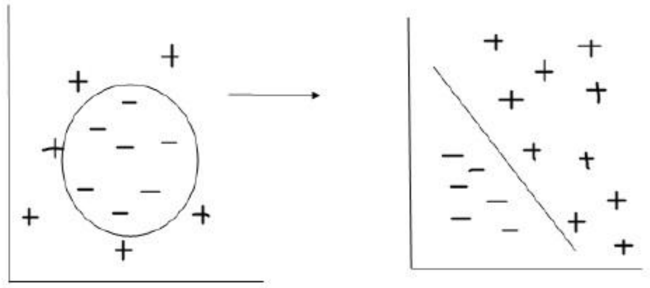
(a) Polynomial kernel



(b) Sigmoid kernel



(c) Laplacian kernel



(d) Gaussian RBF kernel

Some standard kernel functions

Kernel name	**Functional form**	**Remark**
Linear kernel	$K(X,Y) = X^T Y$	The simplest kernel used in linear SVM
Polynomial kernel of degree p	$K(X,Y) = (X^T Y + 1)^\rho, \rho > 0$	It produces a large dot products. Power ρ is specified apriori by the user.
Gaussian (RBF) kernel	$K(X,Y) = e^{\frac{c\|\|x-y\|\|^2}{2\sigma^2}}$	It is a nonlinear kernel called Gaussian Radia Bias function kernel
Laplacian kernel	$K(X,Y) = e^{-\lambda\|\|x-y\|\|}$	Follows laplacian mapping
Sigmoid kernel	$K(X,Y) = tanh(\beta_0 X^T y + \beta_1)$	Followed when statistical test data is known
Mahalanobis kernel	$K(X,Y) = e^{-(X-y)^T A(x-y)}$	Followed when statistical test data is known

Characteristics of SVM

- The SVM learning problem can be formulated as a convex optimization problem, in which efficient algorithms are available to find the global minimum of the objective function. Other methods namely rule based classifier, ANN classifier etc. find only local optimum solutions.
- SVM is the best suitable to classify both linear as well as non-linear training data efficiently.
- SVM can be applied to categorical data by introducing a suitable similarity measures.
- Computational complexity is influenced by number of training data not the dimension of data. In fact, learning is a bit computationally heavy and hence slow, but classification of test is extremely fast and accurate.

Thank You