Support Vector Machine

Debasis Samanta

IIT Kharagpur

dsamanta@iitkgp.ac.in

Autumn 2018

Topics to be covered...

- Introduction to SVM
- Concept of maximum margin hyperplane
- Linear SVM
  1. Calculation of MMH
  2. Learning a linear SVM
  3. Classifying a test sample using linear SVM
  4. Classifying multi-class data
- Non-linear SVM
  - Concept of non-linear data
  - Soft-margin SVM
  - Kernel Trick

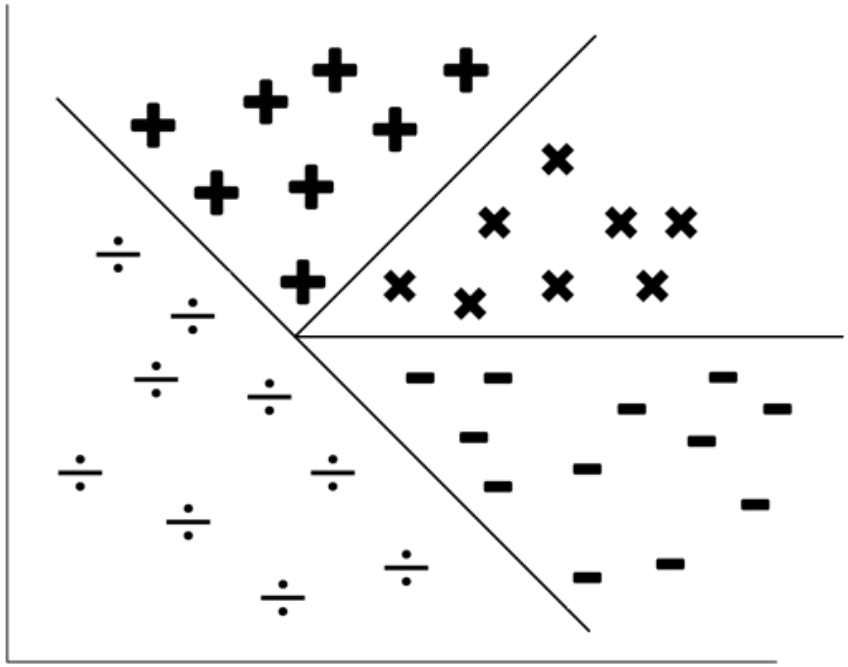
Introduction to SVM

Introduction

- A classification that has received considerable attention is support vector machine and popularly abbreviated as SVM.
- This technique has its roots in statistical learning theory (Vlamidir Vapnik, 1992).
- As a task of classification, it searches for optimal hyperplane(i.e., decision boundary, see Fig. 1 in the next slide) separating the tuples of one class from another.
- SVM works well with higher dimensional data and thus avoids dimensionality problem.
- Although the SVM based classification (i.e., **training** time) is extremely slow, the result, is however highly accurate. Further, testing an unknown data is very fast.
- SVM is less prone to **over fitting** than other methods. It also facilitates compact model for classification.

Introduction

Figure 1: Decision boundary in SVM.



Introduction

In this lecture, we shall discuss the following.

1. Maximum margin hyperplane : a key concept in SVM.
2. Linear SVM : a classification technique when training data are linearly separable.
3. Non-linear SVM : a classification technique when training data are linearly non-separable.

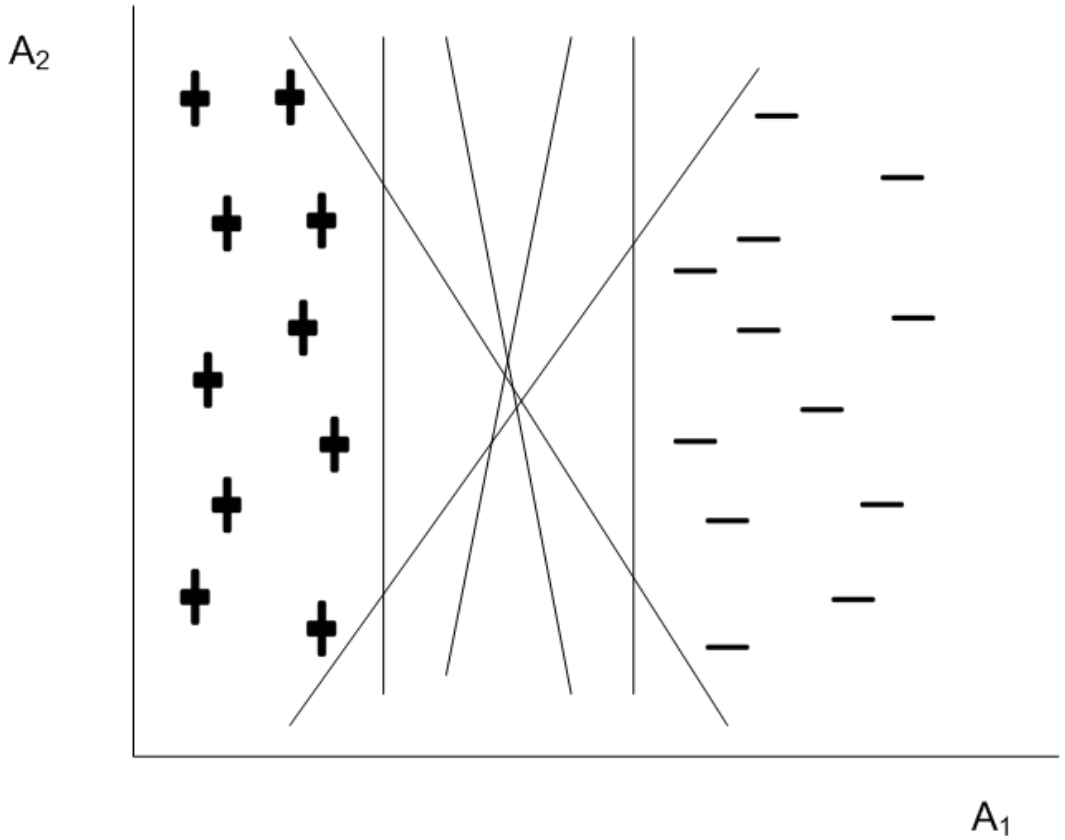
Maximum Margin Hyperplane

Maximum Margin Hyperplane

- In our subsequent discussion, we shall assume a simplistic situation that given a training data D $=\{t_1, t_2, ...t_n\}$ with a set of n tuples, which belong to two classes either + or - and each tuple is described by two attributes say A_1, A_2.

Maximum Margin Hyperplane

Figure 2: A 2D data linearly seperable by hyperplanes



Maximum Margin Hyperplane contd...

- Figure 2 shows a plot of data in 2-D. Another simplistic assumption here is that the data is linearly separable, that is, we can find a hyperplane (in this case, it is a straight line) such that all +'s reside on one side whereas all -'s reside on other side of the hyperplane.
- From Fig. 2, it can be seen that there are an infinite number of separating lines that can be drawn. Therefore, the following two questions arise:
  1. Whether all hyperplanes are equivalent so far the classification of data is concerned?
  2. If not, which hyperplane is the best?

Maximum Margin Hyperplane contd...

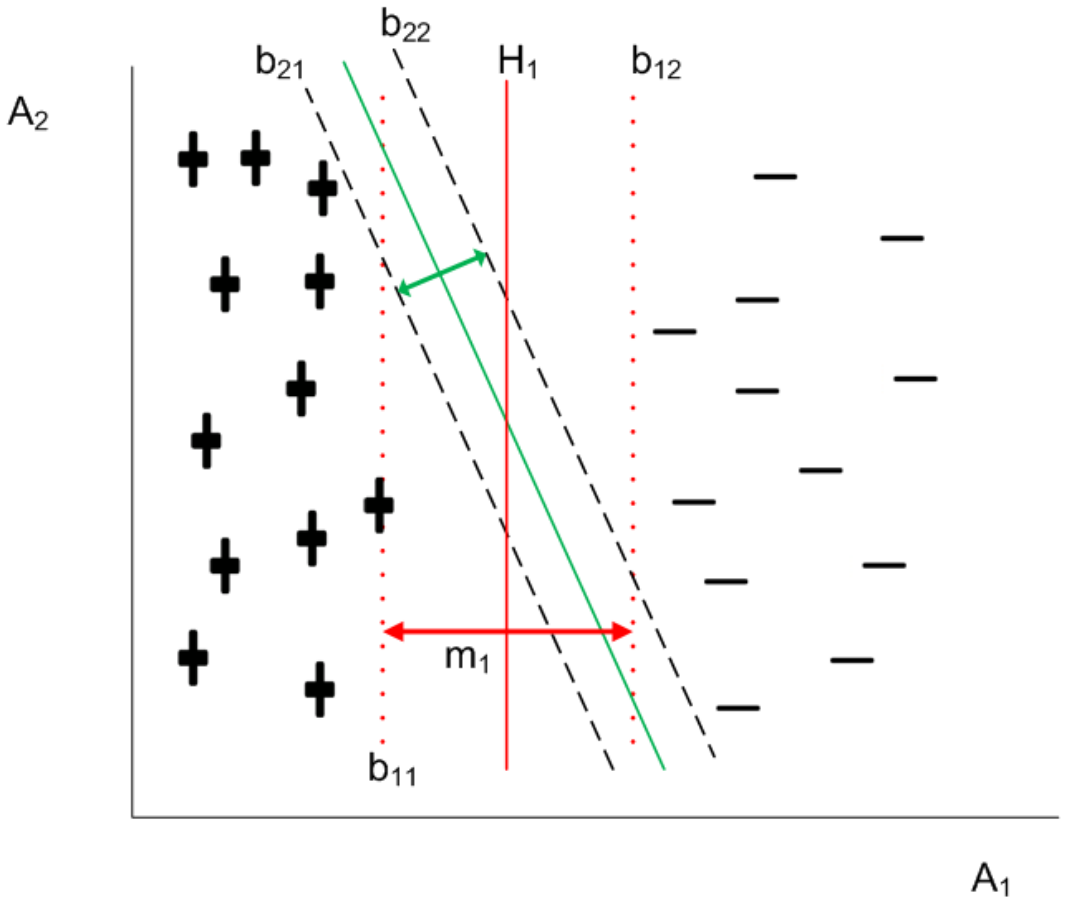
- We may note that so far the classification error is concerned (with training data), all of them are with zero error.
- However, there is no guarantee that all hyperplanes perform equally well on unseen (i.e., test) data.

Maximum Margin Hyperplane contd...

- Thus, for a good classifier it must choose one of the infinite number of hyperplanes, so that it performs better not only on training data but as well as test data.
- To illustrate how the different choices of hyperplane influence the classification error, consider any arbitrary two hyperplanes H_1 and H_2 as shown in Fig. 3.

Maximum Margin Hyperplane

Figure 3: Hyperplanes with decision boundaries and their margins.



Maximum Margin Hyperplane contd...

- In Fig. 3, two hyperplanes H_1 and H_2 have their own boundaries called decision boundaries(denoted as b_{11} and b_{12} for H_1 and b_{21} and b_{22} for H_2).
- A decision boundary is a boundary which is parallel to hyperplane and touches the closest class in one side of the hyperplane.
- The distance between the two decision boundaries of a hyperplane is called the margin. So, if data is classified using Hyperplane H_1, then it is with larger margin then using Hyperplane H_2.
- The margin of hyperplane implies the error in classifier. In other words, the larger the margin, lower is the classification error.

Maximum Margin Hyperplane contd...

- Intuitively, the classifier that contains hyperplane with a small margin are more susceptible to model over fitting and tend to classify with weak confidence on unseen data.
- Thus during the training or learning phase, the approach would be to search for the hyperplane with maximum margin.
- Such a hyperplane is called **maximum margin hyperplane** and abbreviated as MMH.
- We may note the shortest distance from a hyperplane to one of its decision boundary is equal to the shortest distance from the hyperplane to the decision boundary at its other side.
- Alternatively, hyperplane is at the middle of its decision boundaries.

Linear SVM

Linear SVM

- A SVM which is used to classify data which are linearly separable is called linear SVM.
- In other words, a linear SVM searches for a hyperplane with the maximum margin.
- This is why a linear SVM is often termed as a **maximal margin classifier** (MMC).

Finding MMH for a Linear SVM

- In the following, we shall discuss the mathematics to find the MMH given a training set.
- In our discussion, we shall consider a binary classification problem consisting of n training data.
- Each tuple is denoted by (X_i, Y_i) where $X_i = (x_{i1}, x_{i2}.....x_{im})$ corresponds to the attribute set for the i^{th} tuple (data in m-dimensional space) and $Y_i \in$ [+,-] denotes its class label.
- Note that choice of which class should be labeled as + or - is arbitrary.

Finding MMH for a Linear SVM

- Thus, given $\{(X_i, Y_i)\}_{i=1}^{n}$, we are to obtain a hyperplane which separates all $X_i|_{i=1}^{n}$ into two sides of it (of course with maximum gap).
- Before, going to a general equation of a plane in n-dimension, let us consider first, a hyperplane in 2-D plane.

Equation of a hyperplane in 2-D

- Let us consider a 2-D training tuple with attributes A_1 and A_2 as $X = (x_1, x_2)$, where x_1 and x_2 are values of attributes A_1 and A_2, respectively for X.
- Equation of a plane in 2-D space can be written as

  $w_0 + w_1 x_1 + w_2 x_2 = 0$ [e.g., $ax + by + c = 0$]

  where w_0, w_1, and w_2 are some constants defining the slope and intercept of the line.
- Any point lying above such a hyperplane satisfies

$$w_0 + w_1 x_1 + w_2 x_2 > 0 \quad (1)$$

Equation of a hyperplane in 2-D

- Similarly, any point lying below the hyperplane satisfies

$$w_0 + w_1 x_1 + w_2 x_2 < 0 \tag{2}$$

- An SVM hyperplane is an n-dimensional generalization of a straight line in 2-D.

- It can be visualized as a plane surface in 3-D, but it is not easy to visualize when dimensionality is greater than 3!

Equation of a hyperplane

- In fact, Euclidean equation of a hyperplane in R^m is

$$w_1 x_1 + w_2 x_2 + ... + w_m x_m = b \tag{3}$$

where w_i's are the real numbers and b is a real constant (called the intercept, which can be positive or negative).

Finding a hyperplane

- In matrix form, a hyperplane thus can be represented as

$$W.X + b = 0 \tag{4}$$

where W =[w_1,w_2.....w_m] and X = [x_1,x_2.....x_m] and b is a real constant.

- Here, W and b are parameters of the classifier model to be evaluated given a training set D.

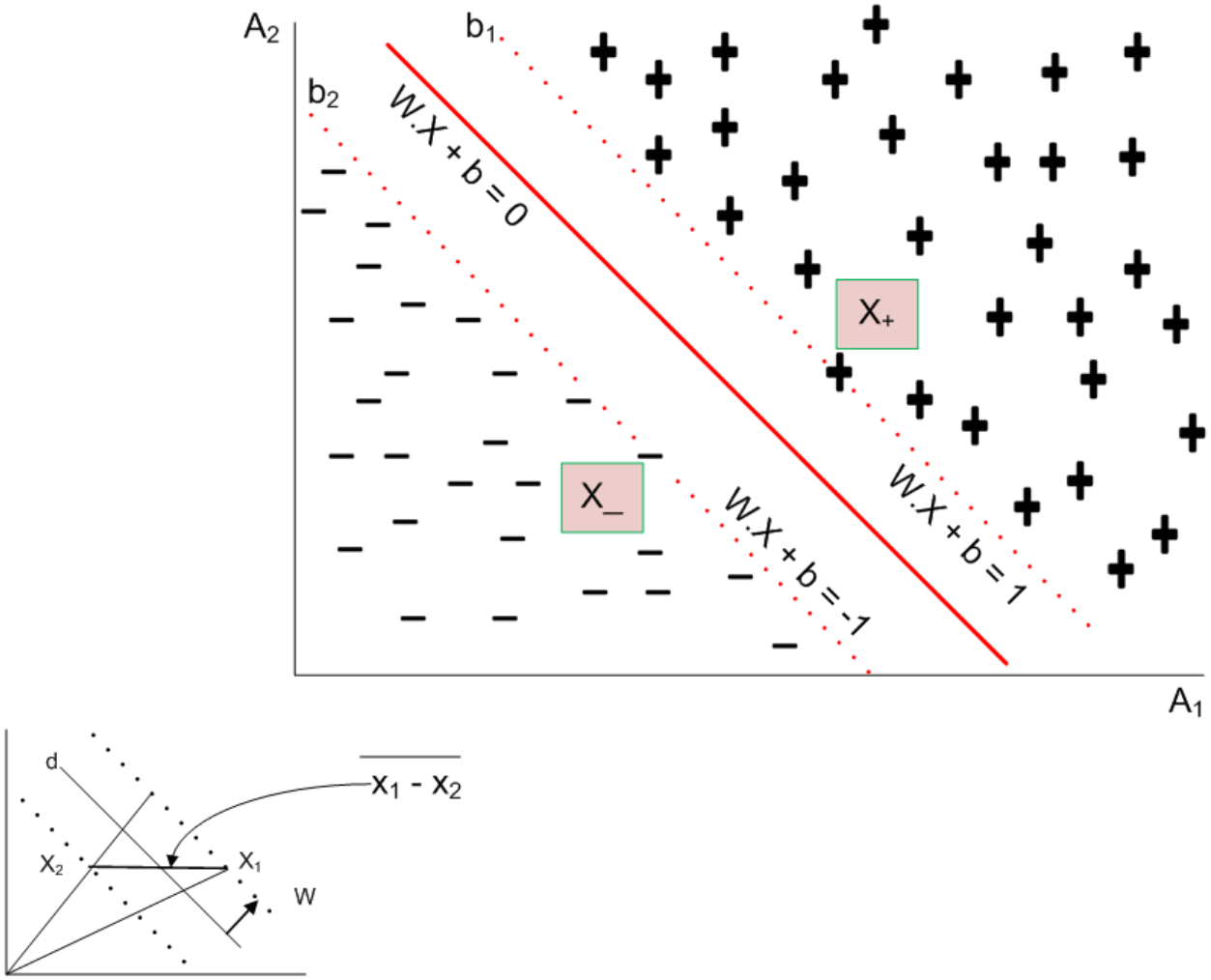
Finding a hyperplane

- Let us consider a two-dimensional training set consisting two classes + and - as shown in Fig. 4.
- Suppose, b_1 and b_2 are two decision boundaries above and below a hyperplane, respectively.
- Consider any two points X_+ and X_- as shown in Fig. 4.
- For X_+ located above the decision boundary, the equation can be written as

$$W.X_+ + b = K \qquad \text{where } K > 0 \tag{5}$$

Finding a hyperplane

Figure 4: Computation of the MMH



Finding a hyperplane

- Similarly, for any point X_- located below the decision boundary, the equation is

$$W.X_- + b = K' \qquad \text{where } K' < 0 \tag{6}$$

- Thus, if we label all +'s are as class label + and all -'s are class label -, then we can predict the class label Y for any test data X as

$$(Y) = \begin{cases} + & \text{if } W.X + b > 0 \\ - & \text{if } W.X + b < 0 \end{cases}$$

Hyperplane and Classification

- Note that $W.X + b = 0$, the equation representing hyperplane can be interpreted as follows.
  - Here, W represents the orientation and b is the intercept of the hyperplane from the origin.
  - If both W and b are scaled (up or down) by dividing a non zero constant, we get the same hyperplane.
  - This means there can be infinite number of solutions using various scaling factors, all of them geometrical representing the same hyperplane.

Hyperplane and Classification

- To avoid such a confusion, we can make W and b unique by adding a constraint that $W'.X + b' = \pm 1$ for data points on boundary of each class.
- It may be noted that $W'.X + b' = \pm 1$ represents two hyperplane parallel to each other.
- For clarity in notation, we write this as $W.X + b = \pm 1$.
- Having this understating, now we are in the position to calculate the margin of a hyperplane.

Calculating Margin of a Hyperplane

- Suppose, x_1 and x_2 (refer Figure 3) are two points on the decision boundaries b_1 and b_2, respectively. Thus,

$$W.x_1 + b = 1 \tag{7}$$

$$W.x_2 + b = -1 \tag{8}$$

or

$$W.(x_1 - x_2) = 2 \tag{9}$$

- This represents a dot (.) product of two vectors W and x_1-x_2. Thus taking magnitude of these vectors, the equation obtained is

$$d = \frac{2}{||W||} \tag{10}$$

where $||W|| = \sqrt{w_1^2 + w_2^2 + ...w_m^2}$ in an m-dimension space.

Calculating Margin of a Hyperplane

We calculate the margin more mathematically, as in the following.

- Consider two parallel hyperplanes H_1 and H_2 as shown in Fig. 5. Let the equations of hyperplanes be

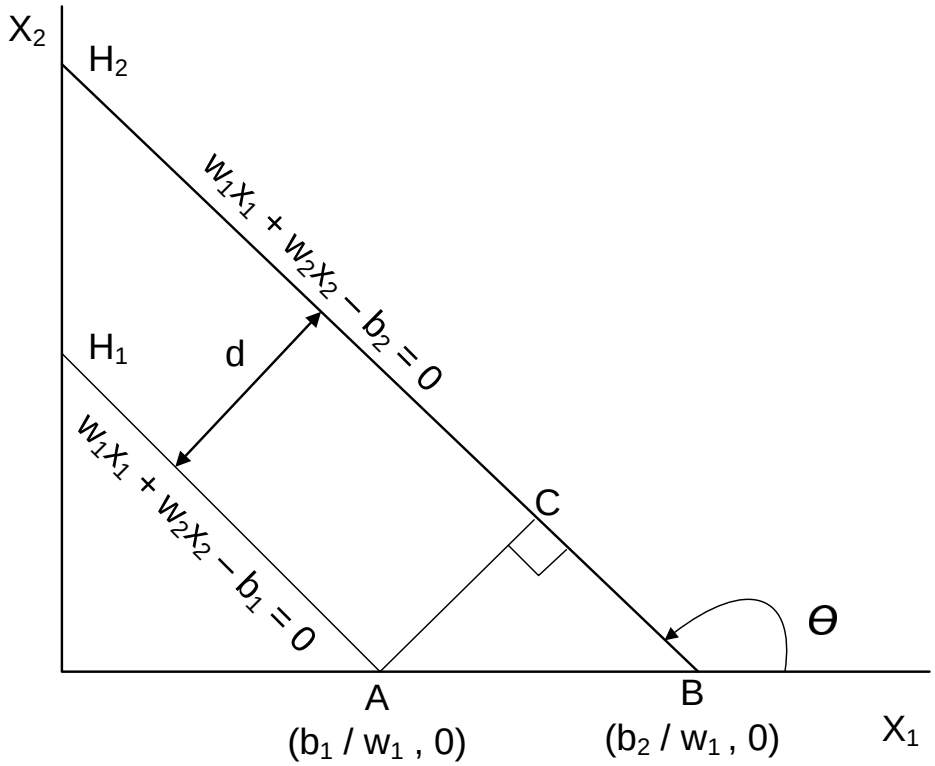
$$H_1 : w_1 x_1 + w_2 x_2 - b_1 = 0 \tag{11}$$

$$H_2 : w_1 x_1 + w_2 x_2 - b_2 = 0 \tag{12}$$

- To draw a perpendicular distance d between H_1 and H_2, we draw a right-angled triangle ABC as shown in Fig.5.

Calculating Margin of a Hyperplane

Figure 5: Detail of margin calculation.



Calculating Margin of a Hyperplane

- Being parallel, the slope of H_1 (and H_2) is $tan\theta = -\frac{w_1}{w_2}$.
- In triangle ABC, AB is the hypotenuse and AC is the perpendicular distance between H_1 and H_2.
- Thus , $sin(180 - \theta) = \frac{AC}{AB}$ or $AC = AB \cdot sin\theta$.

  $AB = \frac{b_2}{w_1} - \frac{b_1}{w_1} = \frac{|b_2 - b_1|}{w_1}$, $sin\theta = \frac{w_1}{\sqrt{w_1^2 + w_2^2}}$

  (Since, $tan\theta = -\frac{w_1}{w_2}$).
- Hence, $AC = \frac{|b_2 - b_1|}{\sqrt{w_1^2 + w_2^2}}$.

Calculating Margin of a Hyperplane

- This can be generalized to find the distance between two parallel margins of any hyperplane in n-dimensional space as

$$d = \frac{|b_2 - b_1|}{\sqrt{w_1^2 + w_2^2 + \cdots + w_n^2}} \simeq \frac{|b_2 - b_1|}{\parallel W \parallel}$$

where, $\parallel W \parallel = \sqrt{w_1^2 + w_2^2 + \cdots + w_n^2}$.

- In SVM literature, this margin is famously written as $\mu(W, b)$.

Calculating Margin of a Hyperplane

- The training phase of SVM involves estimating the parameters W and b for a hyperplane from a given training data.
- The parameters must be chosen in such a way that the following two inequalities are satisfied.

$$W.x_i + b \geq 1 \qquad if \quad y_i = 1 \tag{13}$$

$$W.x_i + b \leq -1 \qquad if \quad y_i = -1 \tag{14}$$

- These conditions impose the requirements that all training tuples from class $Y = +$ must be located on or above the hyperplane $W.x + b = 1$, while those instances from class $Y = -$ must be located on or below the hyperplane $W.x + b = -1$ (also see Fig. 4).

Learning for a Linear SVM

- Both the inequalities can be summarized as

$$y_i(W.x_1 + b) \geq 1 \qquad \forall_i i = 1, 2, ..., n \qquad (15)$$

- Note that any tuples that lie on the hyperplanes H_1 and H_2 are called support vectors.
- Essentially, the support vectors are the most difficult tuples to classify and give the most information regarding classification.
- In the following, we discuss the approach of finding MMH and the support vectors.
- The above problem is turned out to be an optimization problem, that is, to maximize $\mu(W, b) = \frac{2}{\|w\|}$.

Searching for MMH

- Maximizing the margin is, however, equivalent to minimizing the following objective function

$$\mu^{'}(W, b) = \frac{||W||}{2} \tag{16}$$

- In nutshell, the learning task in SVM, can be formulated as the following constrained optimization problem.

$$\begin{aligned} & \textit{minimize} \quad \mu^{'}(W, b) \\ \textit{subject to} \quad & y_i(W.x_i + b) \geq 1, \quad i = 1, 2, 3.....n \end{aligned} \tag{17}$$

Searching for MMH

- The above stated constrained optimization problem is popularly known as convex optimization problem, where objective function is quadratic and constraints are linear in the parameters W and b.
- The well known technique to solve a convex optimization problem is the standard Lagrange Multiplier method.
- First, we shall learn the Lagrange Multiplier method, then come back to the solving of our own SVM problem.

Lagrange Multiplier Method

- The Lagrange multiplier method follows two different steps depending on type of constraints.

1. **Equality constraint optimization problem:** In this case, the problem is of the form:

$$minimize\ f(x_1, x_2, ..., x_d)$$

$$subject\ to\ g_i(x) = 0, i = 1, 2, ..., p$$

2. **Inequality constraint optimization problem:** In this case, the problem is of the form:

$$minimize\ f(x_1, x_2, ..., x_d)$$

$$subject\ to\ h_i(x) \leq 0, i = 1, 2, ..., p$$

Lagrange Multiplier Method

Equality constraint optimization problem solving

- The following steps are involved in this case:
  1. Define the Lagrangian as follows:

$$(X, \lambda) = f(X) + \sum_{i=1}^{p} \lambda_i . g_i(x) \tag{18}$$

where $\lambda_i's$ are dummy variables called Lagrangian multipliers.

  2. Set the first order derivatives of the Lagrangian with respect to x and the Lagrangian multipliers $\lambda_i's$ to zero's. That is

$$\frac{\delta L}{\delta x_i} = 0, i = 1, 2, ..., d$$
$$\frac{\delta L}{\delta \lambda_i} = 0, i = 1, 2, ..., p$$

  3. Solve the $(d + p)$ equations to find the optimal value of $X = [x_1, x_2 ....., x_d]$ and $\lambda_i's$.

Example: Equality constraint optimization problem

Suppose, minimize $f(x, y) = x + 2y$
subject to $x^2 + y^2 - 4 = 0$

1. Lagrangian $L(x, y, \lambda) = x + 2y + \lambda(x^2 + y^2 - 4)$
2. $\frac{\delta L}{\delta x} = 1 + 2\lambda x = 0$
   $\frac{\delta L}{\delta y} = 1 + 2\lambda y = 0$
   $\frac{\delta L}{\delta \lambda} = x^2 + y^2 - 4 = 0$
3. Solving the above three equations for x, y and λ, we get $x = \mp\frac{2}{\sqrt{5}}$, $y = \mp\frac{4}{\sqrt{5}}$ and $\lambda = \pm\frac{\sqrt{5}}{4}$

Lagrange Multiplier Method

Example : Equality constraint optimization problem

- When $\lambda = \frac{\sqrt{5}}{4}$,
  $x = -\frac{2}{\sqrt{5}}$,
  $y = -\frac{4}{\sqrt{5}}$,
  we get f$(x, y, \lambda) = -\frac{10}{\sqrt{5}}$
- Similarly, when $\lambda = -\frac{\sqrt{5}}{4}$,
  $x = \frac{2}{\sqrt{5}}$,
  $y = \frac{4}{\sqrt{5}}$,
  we get f$(x, y, \lambda) = \frac{10}{\sqrt{5}}$
- Thus, the function f(x, y) has its minimum value at
  $x = -\frac{2}{\sqrt{5}}, y = -\frac{4}{\sqrt{5}}$

Lagrange Multiplier Method

Inequality constraint optimization problem solving

- The method for solving this problem is quite similar to the Lagrange multiplier method described above.
- It starts with the Lagrangian

$$L = f(x) + \sum_{i=1}^{p} \lambda_i . h_i(x) \tag{19}$$

- In addition to this, it introduces additional constraints, called **Karush-Kuhn-Tucker (KKT) constraints**, which are stated in the next slide.

Lagrange Multiplier Method

Inequality constraint optimization problem solving

$$\frac{\delta L}{\delta x_i} = 0, i = 1, 2, ..., d$$

$$\lambda_i \geq 0, i = 1, 2....., p$$

$$h_i(x) \leq 0, i = 1, 2....., p$$

$$\lambda_i . h_i(x) = 0, i = 1, 2....., p$$

Solving the above equations, we can find the optimal value of $f(x)$.

Example: Inequality constraint optimization problem

Consider the following problem.

Minimize $f(x, y) = (x-1)^2 + (y-3)^2$
subject to $x + y \leq 2$,
$y \geq x$

- The Lagrangian for this problem is
  $L = (x-1)^2 + (y-3)^2 + \lambda_1(x + y - 2) + \lambda_2(x - y)$.

  subject to the KKT constraints, which are as follows:

Example: Inequality constraint optimization problem

$$\frac{\delta L}{\delta x} = 2(x-1) + \lambda_1 + \lambda_2 = 0$$

$$\frac{\delta L}{\delta y} = 2(y-3) + \lambda_1 - \lambda_2 = 0$$

$$\lambda_1(x+y-2) = 0$$

$$\lambda_2(x-y) = 0$$

$$\lambda_1 \geq 0, \lambda_2 \geq 0$$

$$(x+y) \leq 2, y \geq x$$

Example: Inequality constraint optimization problem

To solve KKT constraints, we have to check the following tests:

- Case 1: $\lambda_1 = 0, \lambda_2 = 0$

  $$2(x-1) = 0 \mid 2(y-3) = 0 \Rightarrow x = 1, y = 3$$

  since, $x + y = 4$, it violates $x + y \leq 2$; is not a feasible solution.

- Case 2: $\lambda_1 = 0, \lambda_2 \neq 0$ $2(x-y) = 0 \mid$
  $2(x-1) + \lambda_2 = 0 \mid$
  $2(y-3) - \lambda_2 = 0$
  $\Rightarrow x = 2, y = 2$ and $\lambda_2 = -2$
  since, $x + y \leq 4$, it violates $\lambda_2 \geq 0$; is not a feasible solution.

Example: Inequality constraint optimization problem

- Case 3: $\lambda_1 \neq 0, \lambda_2 = 0$ $2(x+y) = 2$
  $2(x-1) + \lambda_1 = 0$
  $2(y-3) + \lambda_1 = 0$

  $\Rightarrow x = 0, y = 2$ and $\lambda_1 = 2$; this is a feasible solution.

- Case 4: $\lambda_1 \neq 0, \lambda_2 \neq 0$ $2(x+y) = 2$
  $2(x-y) = 0$
  $2(x-1) + \lambda_1 + \lambda_2 = 0$
  $2(y-3) + \lambda_1 - \lambda_2 = 0$

  $\Rightarrow x = 1, y = 1$ and $\lambda_1 = 2$ $\lambda_2 = -2$
  This is not a feasible solution.

LMM to Solve Linear SVM

- The optimization problem for the linear SVM is inequality constraint optimization problem.
- The Lagrangian multiplier for this optimization problem can be written as

$$L = \frac{||W||^2}{2} - \sum_{i=1}^{n} \lambda_i (y_i (W.x_i + b) - 1) \tag{20}$$

where the parameters $\lambda_i's$ are the Lagrangian multipliers, and $W = [w_1, w_2 ..... w_m]$ and b are the model parameters.

LMM to Solve Linear SVM

- The KKT constraints are:

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

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

$$\lambda \geq 0, i = 1, 2, ...n$$

$$\lambda_i [y_i (W.x_i + b) - 1] = 0, i = 1, 2, ...n$$

$$y_i (W.x_i + b) \geq 1, i = 1, 2, ...n$$

- Solving KKT constraints are computationally expensive and can be solved using a typical linear/ quadratic programming technique (or any other numerical technique).

LMM to Solve Linear SVM

- We first solve the above set of equations to find all the feasible solutions.
- Then, we can determine optimum value of $\mu(W, b)$.

Note:

1. Lagrangian multiplier λ_i must be zero unless the training instance x_i satisfies the equation $y_i(W.x_i + b) = 1$. Thus, the training tuples with $\lambda_i > 0$ lie on the hyperplane margins and hence are support vectors.
2. The training instances that do not lie on the hyperplane margin have $\lambda_i = 0$.

Classifying a test sample using Linear SVM

- For a given training data, using SVM principle, we obtain MMH in the form of W, b and λ_i's. This is the machine (i.e., the SVM).
- Now, let us see how this MMH can be used to classify a test tuple say X. This can be done as follows.

$$\delta(X) = W.X + b = \sum_{i=1}^{n} \lambda_i.y_i.x_i.X + b \tag{21}$$

Note that

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

Classifying a test sample using Linear SVM

- This is famously called as “Representer Theorem” which states that the solution W always be represented as a linear combination of training data.

$$Thus, \delta(X) = W.X + b = \sum_{i=1}^{n} \lambda_i.y_i.x_i.X + b$$

Classifying a test sample using Linear SVM

- The above involves a dot product of $x_i.X$,where x_i is a support vector (this is so because $(\lambda_i) = 0$ for all training tuples except the support vectors), we can check the sign of $\delta(X)$.
- If it is positive, then X falls on or above the MMH and so the SVM predicts that X belongs to class label +. On the other hand, if the sign is negative, then X falls on or below MMH and the class prediction is -.

Note:

1. Once the SVM is trained with training data, the complexity of the classifier is characterized by the number of support vectors.
2. Dimensionality of data is not an issue in SVM unlike in other classifier.

Illustration : Linear SVM

- Consider the case of a binary classification starting with a training data of 8 tuples as shown in Table 1.
- Using quadratic programming, we can solve the KKT constraints to obtain the Lagrange multipliers λ_i for each training tuple, which is shown in Table 1.
- Note that only the first two tuples are support vectors in this case.
- Let $W = (w_1, w_2)$ and b denote the parameter to be determined now. We can solve for w_1 and w_2 as follows:

$$w_1 = \sum_i \lambda_i . y_i . x_{i1} = 65.52 \times 1 \times 0.38 + 65.52 \times -1 \times 0.49 = -6.64 \quad (22)$$

$$w_2 = \sum_i \lambda_i . y_i . x_{i2} = 65.52 \times 1 \times 0.47 + 65.52 \times -1 \times 0.61 = -9.32 \quad (23)$$

Illustration : Linear SVM

Table 1: Training Data

A_1	A_2	y	λ_i
0.38	0.47	+	65.52
0.49	0.61	-	65.52
0.92	0.41	-	0
0.74	0.89	-	0
0.18	0.58	+	0
0.41	0.35	+	0
0.93	0.81	-	0
0.21	0.10	+	0

Illustration : Linear SVM

Figure 6: Linear SVM example.

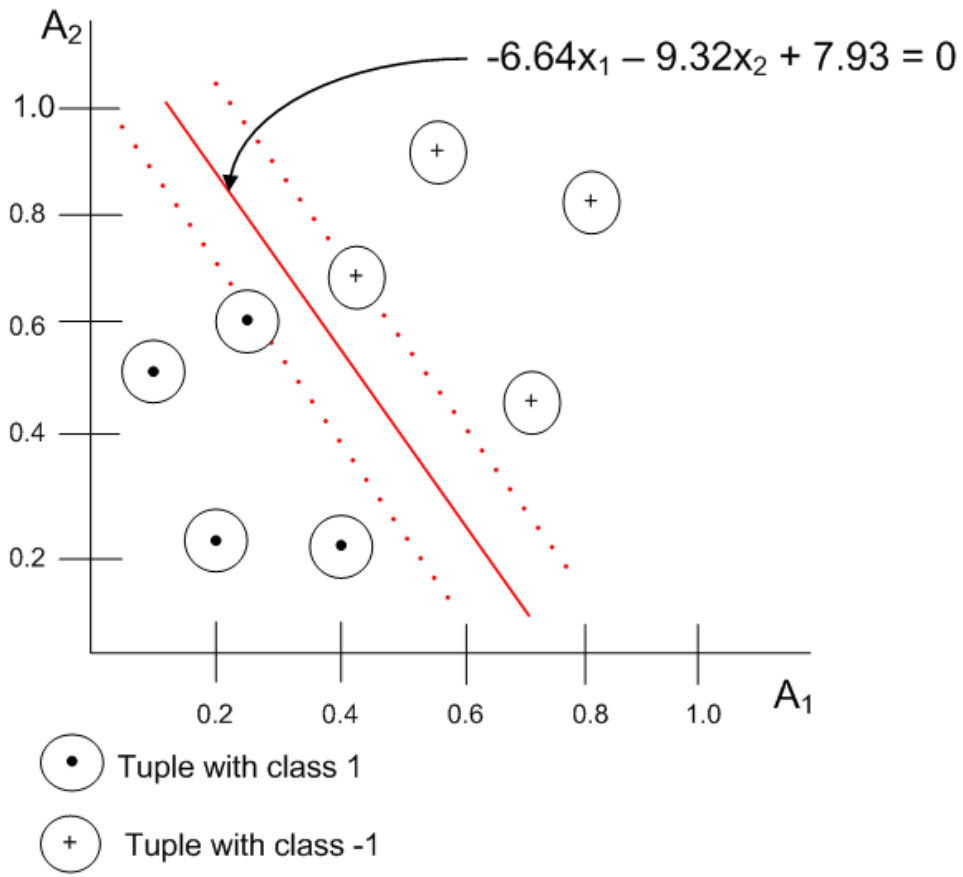


Illustration : Linear SVM

- The parameter b can be calculated for each support vector as follows
  $b_1 = 1 - W.x_1$ // for support vector x_1
  $= 1 - (-6.64) \times 0.38 - (-9.32) \times 0.47$ //using dot product
  $= 7.93$
  $b_2 = 1 - W.x_2$ // for support vector x_2
  $= 1 - (-6.64) \times 0.48 - (-9.32) \times 0.611$ //using dot product
  $= 7.93$

- Averaging these values of b_1 and b_2, we get $b = 7.93$.

Illustration : Linear SVM

- Thus, the MMH is $-6.64x_1 - 9.32x_2 + 7.93 = 0$ (also see Fig. 6).
- Suppose, test data is $X = (0.5, 0.5)$. Therefore,
  $\delta(X) = W.X + b$
  $= -6.64 \times 0.5 - 9.32 \times 0.5 + 7.93$
  $= -0.05$
  $= -ve$

- This implies that the test data falls on or below the MMH and SVM classifies that X belongs to class label -.

Classification of Multiple-class Data

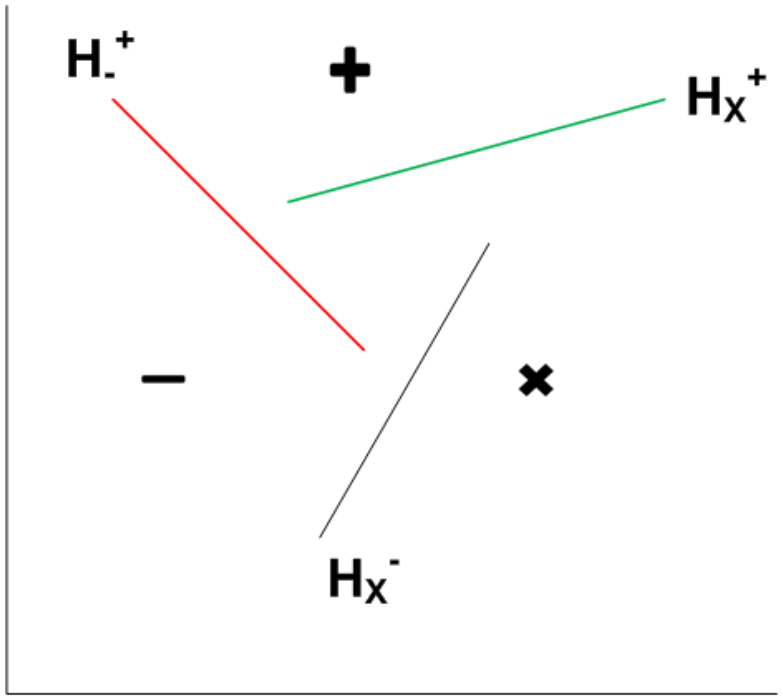
- In the discussion of linear SVM, we have limited to binary classification (i.e., classification with two classes only).
- Note that the discussed linear SVM can handle any n-dimension, $n \geq 2$. Now, we are to discuss a more generalized linear SVM to classify n-dimensional data belong to two or more classes.
- There are two possibilities: all classes are pairwise linearly separable, or classes are overlapping, that is, not linearly separable.
- If the classes are pair wise linearly separable, then we can extend the principle of linear SVM to each pair. Theare two strategies:
  1. One versus one (OVO) strategy
  2. One versus all (OVA) strategy

Multi-Class Classification: OVO Strategy

- In OVO strategy, we are to find MMHs for each pair of classes.
- Thus, if there are n classes, then ${}^{n}c_2$ pairs and hence so many classifiers possible (of course, some of which may be redundant).
- Also, see Fig. 7 for 3 classes (namely +, - and $\times$). Here, H_y^x denotes MMH between class labels x and y.
- Similarly, you can think for 4 classes (namely +, -, $\times$, $\div$) and more.

Multi-Class Classification: OVO Strategy

Figure 7: 3-pairwise linearly separable classes.



Multi-Class Classification: OVO Strategy

- With OVO strategy, we test each of the classifier in turn and obtain $\delta_i^j(X)$, that is, the count for MMH between i^{th} and i^{th} classes for test data X.
- If there is a class i, for which $\delta_i^j(X)$ for all j(and $j \neq i$), gives same sign, then unambiguously, we can say that X is in class i.

Multi-Class Classification: OVA Strategy

- OVO strategy is not useful for data with a large number of classes, as the computational complexity increases exponentially with the number of classes.
- As an alternative to OVO strategy, OVA(one versus all) strategy has been proposed.
- In this approach, we choose any class say C_i and consider that all tuples of other classes belong to a single class.

Multi-Class Classification: OVA Strategy

- This is, therefore, transformed into a binary classification problem and using the linear SVM discussed above, we can find the hyperplane. Let the hyperplane between C_i and remaining classes be MMH_i.
- The process is repeated for each $C_i \epsilon [C_1, C_2, ...C_k]$ and getting MMH_i.
- Thus, with OVA strategies we get k classifiers.

Multi-Class Classification: OVA Strategy

- The unseen data X is then tested with each classifier so obtained.
- Let $\delta_j(X)$ be the test result with MMH_j, which has the maximum magnitude of test values.
- That is

$$\delta_j(X) = max_{\forall i}\{\delta_i(X)\} \tag{24}$$

- Thus, X is classified into class C_j.

Multi-Class Classification: OVA Strategy

Note:

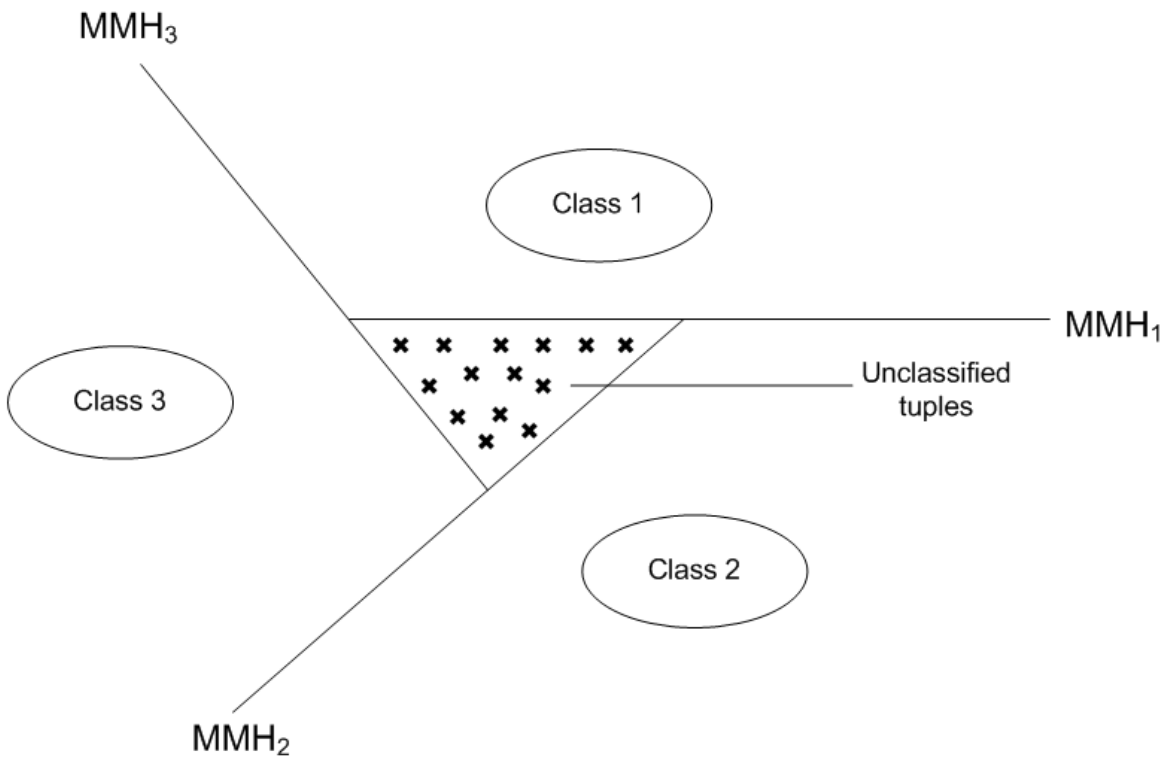
- The linear SVM that is used to classify multi-class data fails, if all classes are not linearly separable.
- If one class is linearly separable to remaining other classes and test data belongs to that particular class, then only it classifies accurately.

Multi-Class Classification: OVA Strategy

- Further, it is possible to have some tuples which cannot be classified none of the linear SVMs (also see Fig.8).
- There are some tuples which cannot be classified unambiguously by neither of the hyperplanes.
- All these tuples may be due to noise, errors or data are not linearly separable.

Multi-Class Classification: OVA Strategy

Figure 8: Unclassifiable region in OVA strategy.



Non-Linear SVM

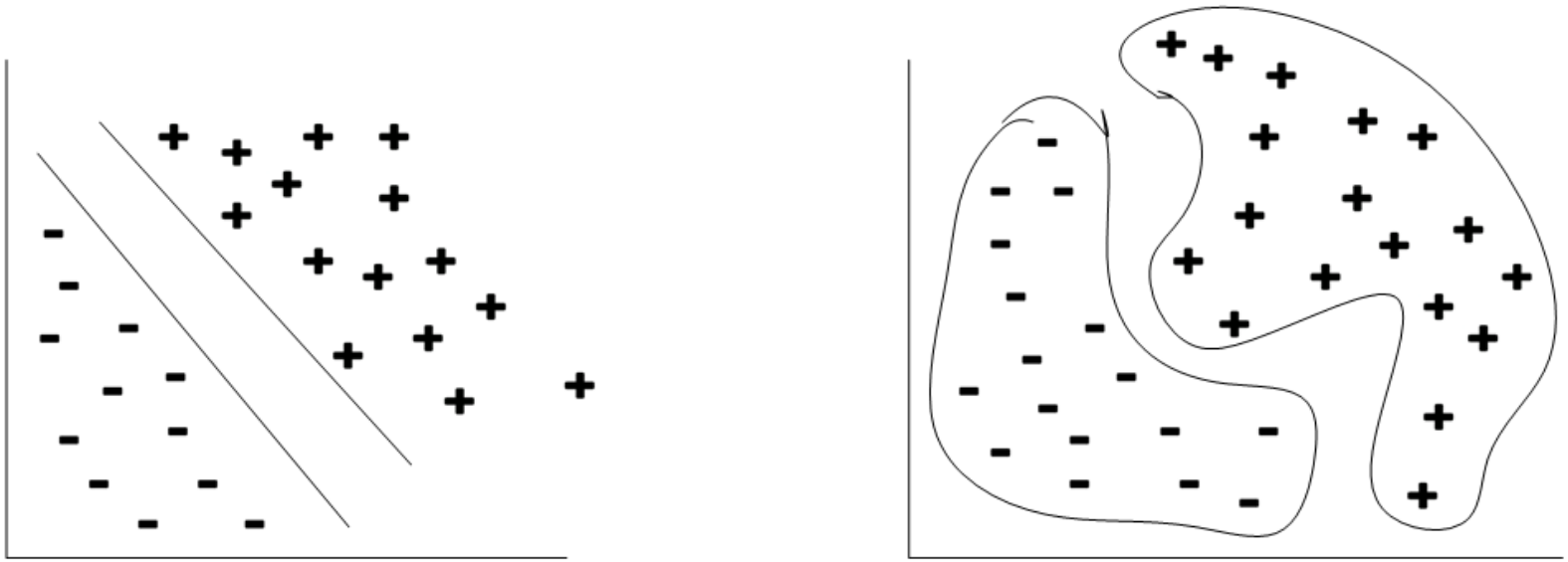
Non-Linear SVM

SVM classification for non separable data

- Figure 9 shows a 2-D views of data when they are linearly separable and not separable.
- In general, if data are linearly separable, then there is a hyperplane otherwise no hyperplane.

Linear and Non-Linear Separable Data

Figure 9: Two types of training data.



(a) Linearly separable

(b) Linearly non-separable

SVM classification for non separable data

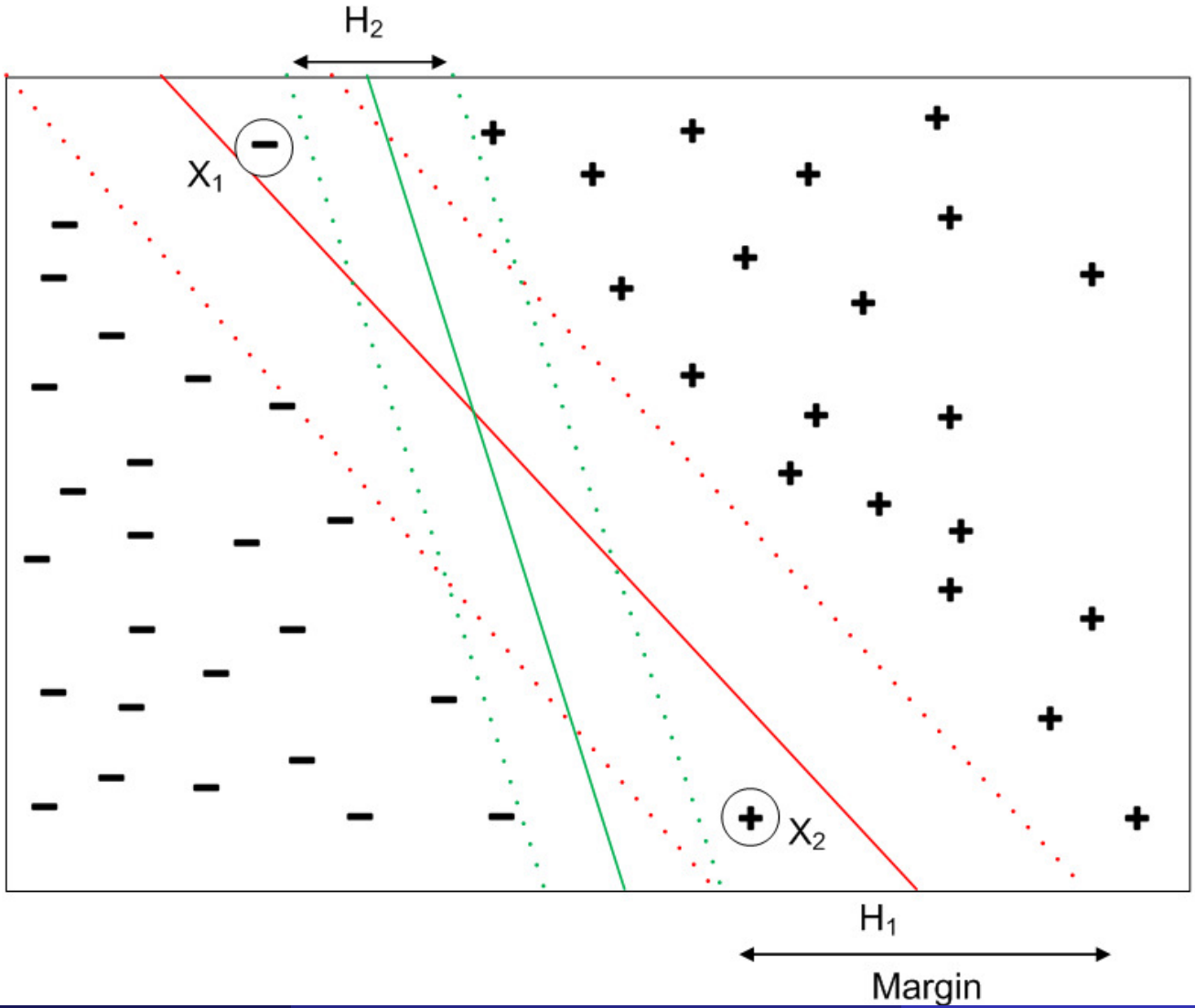
- Such a linearly not separable data can be classified using two approaches.
  1. Linear SVM with soft margin
  2. Non-linear SVM
- In the following, we discuss the extension of linear SVM to classify linearly not separable data.
- We discuss non-linear SVM in detail later.

Linear SVM for Linearly Not Separable Data

- If the number of training data instances violating linear separability is less, then we can use linear SVM classifier to classify them.
- The rational behind this approach can be better understood from Fig. 10.

Linear SVM for Linearly Not Separable Data

Figure 10: Problem with linear SVM for linearly not separable data.



Linear SVM for Linearly Not 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.

Linear SVM for Linearly Not Separable Data

- 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.
- The concept of Soft Margin SVM is presented in the following slides.

Soft Margin SVM

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

$$minimize \frac{||W||^2}{2} \tag{25}$$

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

- 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

- To do this, soft margin SVM introduces slack variable (ξ), a positive-value into the constraint of optimization problem.
- Thus, for soft margin we rewrite the optimization problem as follows.

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

$$subject\ to\ (W.x_i + b) \geq 1 - \xi_i, if\ y_i = +1$$

$$(W.x_i + b) \leq -1 + \xi_i,\ if\ y_i = -1 \tag{26}$$

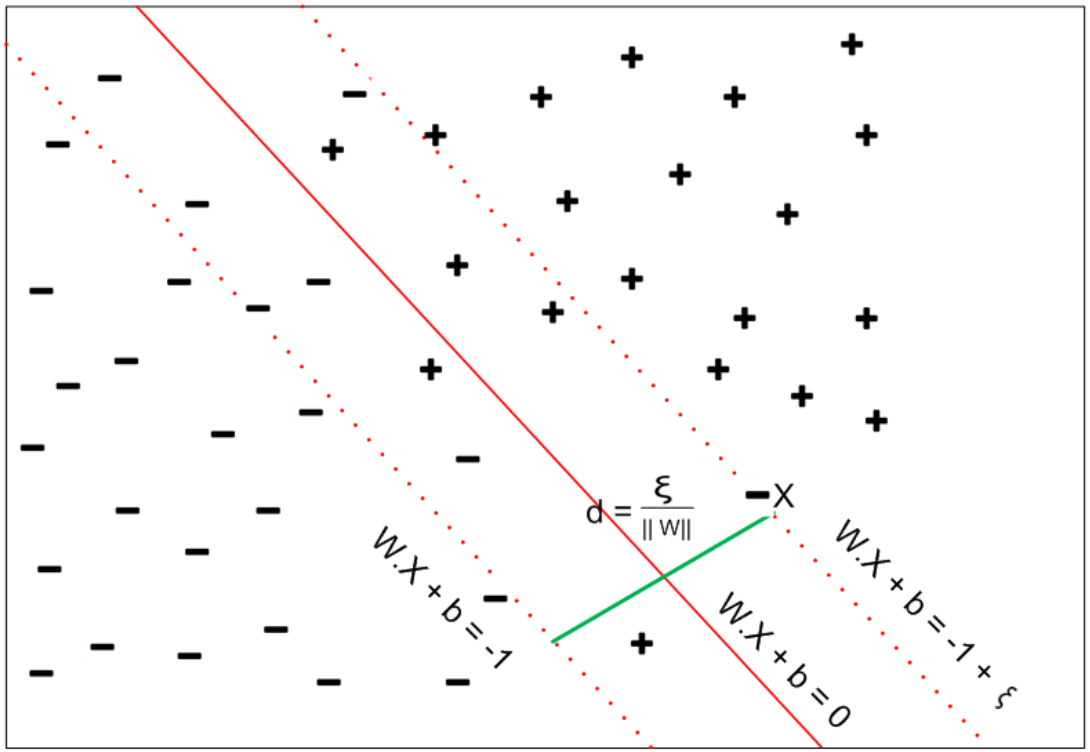
where $\forall_i, \xi_i \geq 0$.

- Thus, in soft margin SVM, we are to calculate W, b and ξ_is as a solution to learn SVM.

Soft Margin SVM : Interpretation of ξ

- Let us find an interpretation of ξ, the slack variable in soft margin SVM. For this consider the data distribution shown in the Fig. 11.

Figure 11: Interpretation of slack variable ξ.



Soft Margin SVM : Interpretation of ξ

- 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.

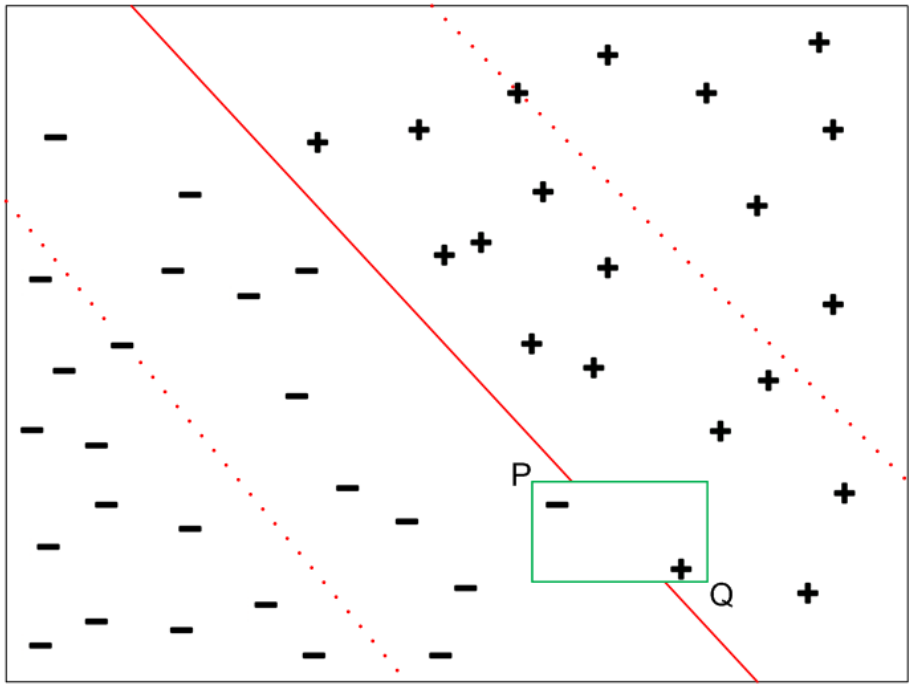
Soft Margin SVM : Interpretation of ξ

- In principle, Eqn. 25 and 26 can be chosen as the optimization problem to train a soft margin SVM.
- However, the soft margin SVM should impose a constraint on the number of such non linearly separable data it takes into account.
- This is so because a SVM may be trained with decision boundaries with very high margin thus, chances of misclassifying many of the training data.

Soft Margin SVM : Interpretation of ξ

- This is explained in Fig. 12. Here, if we increase margin further, then P and Q will be misclassified.
- Thus, there is a trade-off between the length of margin and training error.

Figure 12: MMH with wide margin and large training error.



Soft Margin SVM : Interpretation of ξ

- 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

$$f(W) = \frac{||W||^2}{2} + c.\sum_{i=1}^{n}(\xi_i)^{\phi} \tag{27}$$

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.

Solving for Soft Margin SVM

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

$$L = \frac{||W||^2}{2} + c.\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \lambda_i (y_i(W.x_i + b) - 1 + \xi_i) - \sum_{i=1}^{n} \mu_i.\xi_i \quad (28)$$

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

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

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

$$\mu_i.\xi_i = 0$$

Solving for Soft Margin SVM

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

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

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

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

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

Solving for Soft Margin SVM

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

$$\lambda_i + \mu_i = c \text{ and } \mu_i.\xi_i = 0 \forall_i = 1, 2, ...n. \tag{29}$$

- 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:**
  1. $\lambda_i \neq 0$ for support vectors or has $\xi_i > 0$ and
  2. $\mu_i = 0$ for those training data which are misclassified, that is, $\xi_i > 0$.

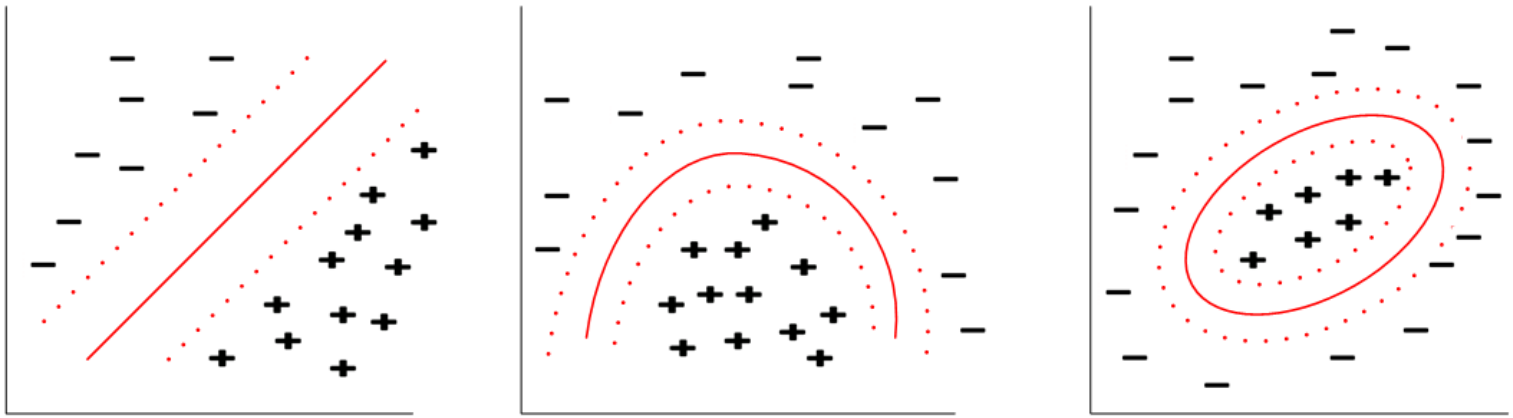
Non-Linear SVM

- 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 hypersurface).

Non-Linear SVM

- For understanding this, see Figure 13.
- Note that a linear hyperplane is expressed as a linear equation in terms of *n*-dimensional component, whereas a non-linear hypersurface is a non-linear expression.

Figure 13: 2D view of few class separabilities.



(a) Linear hyperplane (b) Nonlinear hyperplane (c) Nonlinear hypersurface

Non-Linear SVM

- A hyperplane is expressed as

$$linear: \ w_1 x_1 + w_2 x_2 + w_3 x_3 + c = 0 \tag{30}$$

- Whereas a non-linear hypersurface is expressed as.

$$Nonlinear: \ w_1 x_1^2 + w_2 x_2^2 + w_3 x_1 x_2 + w_4 x_3^2 + w_5 x_1 x_3 + c = 0 \tag{31}$$

- 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.

Non-Linear SVM

- This can be achieved in two major steps.
  1. 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 (i.e., $\approx$ training data) not by the dimension of data.
  2. 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 SVM.
- 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.

Concept of Non-Linear Mapping

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.

$$X(x_1, x_2, x_3) = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_1^2 + w_5 x_1 x_2 + w_6 x_1 x_3 + c$$

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:

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

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

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

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

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

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

Concept of Non-Linear Mapping

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

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

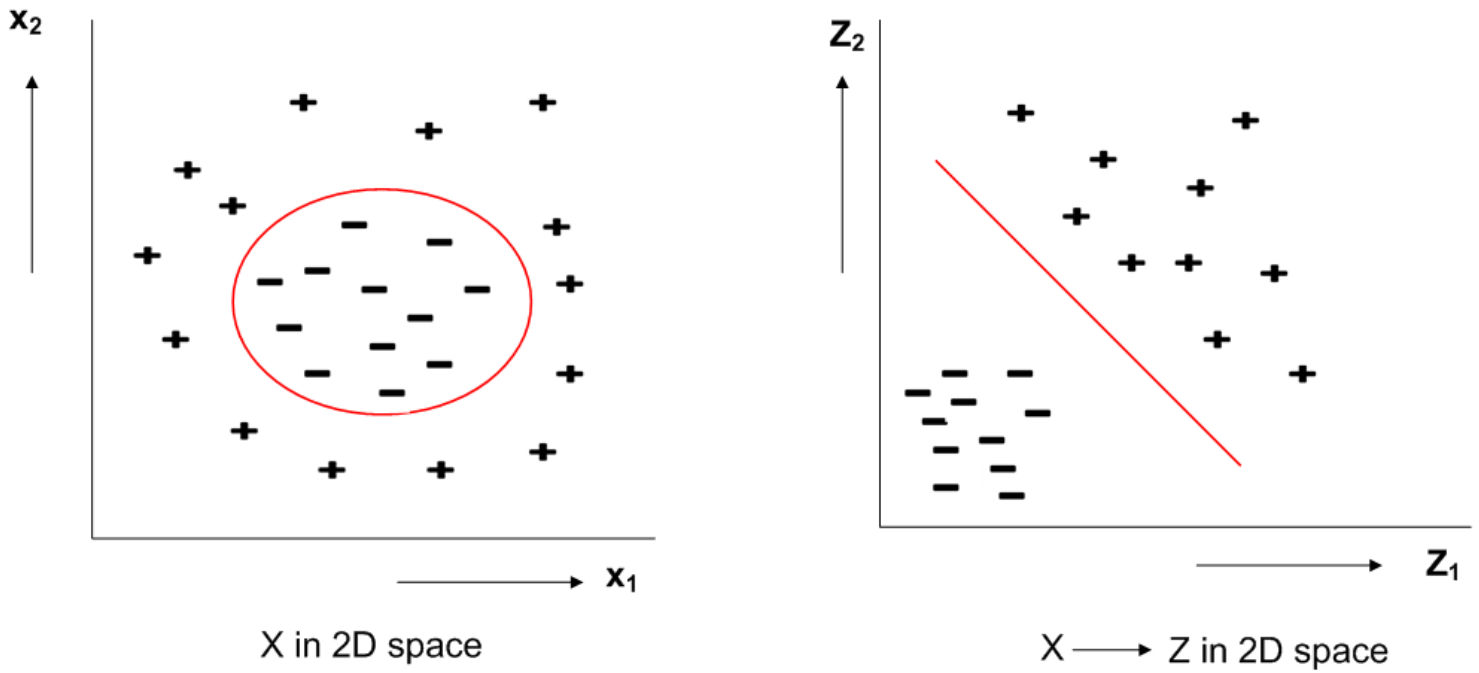
- Thus, if Z space has input data for its attributes x_1, x_2, x_3 (and hence Z's values), then we can classify them using linear decision boundaries.

Concept of Non-Linear Mapping

Example: 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 -).

Figure 14: Non-linear mapping to Linear SVM.



Concept of Non-Linear Mapping

Example: Non-linear mapping to linear SVM

- We see that all instances of class -1 can be separated from instances of class +1 by a circle, The following equation of the decision boundary can be thought of:

$$X(x_1, x_2) = +1 \text{ if } \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} > 2$$

$$X(x_1, x_2) = -1 \text{ otherwise} \tag{32}$$

Concept of Non-Linear Mapping

Example: Non-linear mapping to linear SVM

- The decision boundary can be written as:

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

$$or\ x_1^2 - x_1 + x_2^2 - x_2 = 0.46$$

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

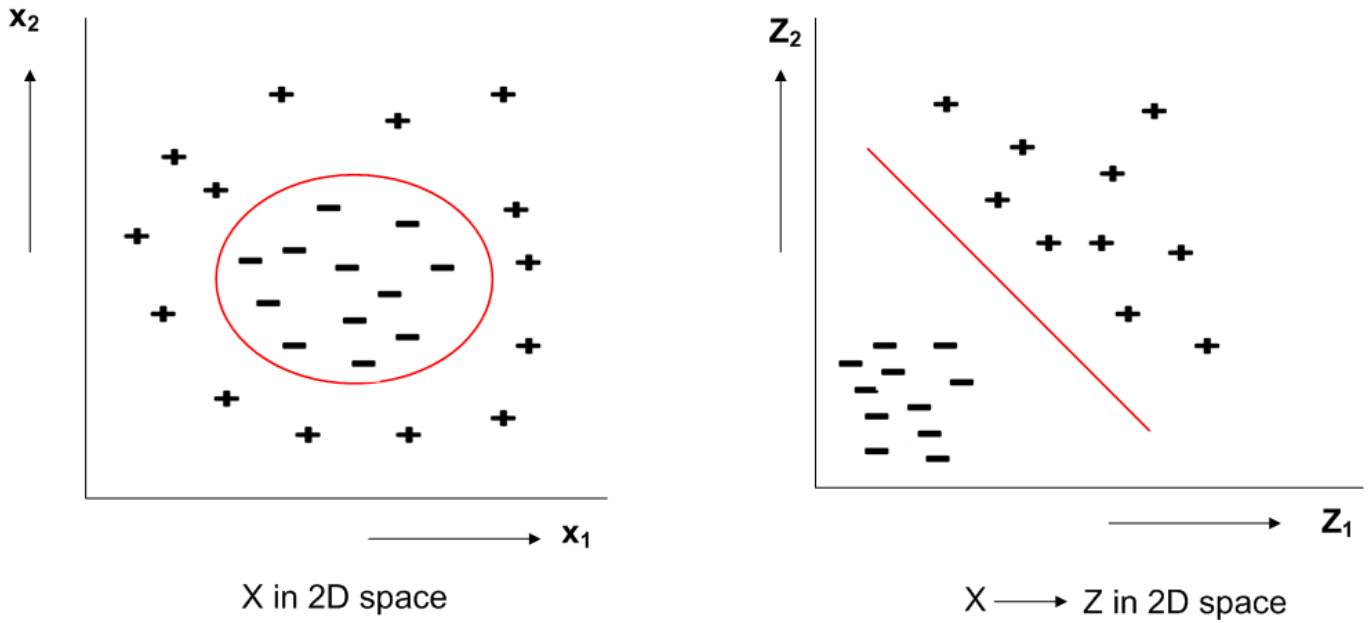
$$Z(z_1, z_2) : \phi_1(x) = x_1^2 - x_1, \phi_2(x) = x_2^2 - x_2 \tag{33}$$

Concept of Non-Linear Mapping

Example: Non-linear mapping to linear SVM

- The Z space when plotted will take view as shown in Fig. 15, where data are separable with linear boundary, namely $Z : z_1 + z_2 = -0.46$

Figure 15: Non-linear mapping to Linear SVM.



Non-Linear to Linear Transformation: Issues

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

  1. 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.

  2. 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.

Non-Linear to Linear Transformation: Issues

- ...

3. 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 $X_i.X$ (in $\delta(X)$ see Eqn. 18), 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.
4. Computational cost: Solving the quadratic constrained optimization problem in the high dimensional feature space is too a computationally expensive task.

Non-Linear to Linear Transformation: Issues

- Fortunately, mathematicians have cleverly proposes an elegant solution to the above problems.
- Their solution consist of the following:
  1. Dual formulation of optimization problem
  2. Kernel trick
- In the next few slides, we shall learn about the above-mentioned two topics.

Dual Formulation of Optimization Problem

- We have already learned the Lagrangian formulation to find the maximum margin hyperplane as a linear SVM classifier.
- Such a formulation is called primal form of the constraint optimization problem.
- Primal form of Lagrangian optimization problem is reproduced further

$$Minimize \frac{||W||^2}{2}$$

$$Subject\ to\ y_i(W.x_i + b) \geq 1,\ i = 1, 2, 3.....n. \tag{34}$$

Dual Formulation of Optimization Problem

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

$$L_p = \frac{||W||^2}{2} - \sum_{i=1}^{n} \lambda_i (y_i (W.x_i + b) - 1) \tag{35}$$

where λ_i's are called Lagrange multipliers.

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

Dual Formulation of 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.

$$\frac{\delta L_p}{\delta W} = 0 \Rightarrow W = \sum_{i=1}^{n} \lambda_i . y_i . x_i \tag{36}$$

$$\frac{\delta L_p}{\delta b} = 0 \Rightarrow \sum_{i=1}^{n} \lambda_i . y_i = 0 \tag{37}$$

Dual Formulation of Optimization Problem

- 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:

$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i . \lambda_j . y_i . y_j . x_i . x_j \tag{38}$$

Dual Formulation of Optimization Problem

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

$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i . \lambda_j . y_i . y_j . x_i . x_j \tag{39}$$

Dual Formulation of Optimization Problem

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

1. 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.
2. 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.
3. 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.

Dual Formulation of Optimization Problem

- ...

4. 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.

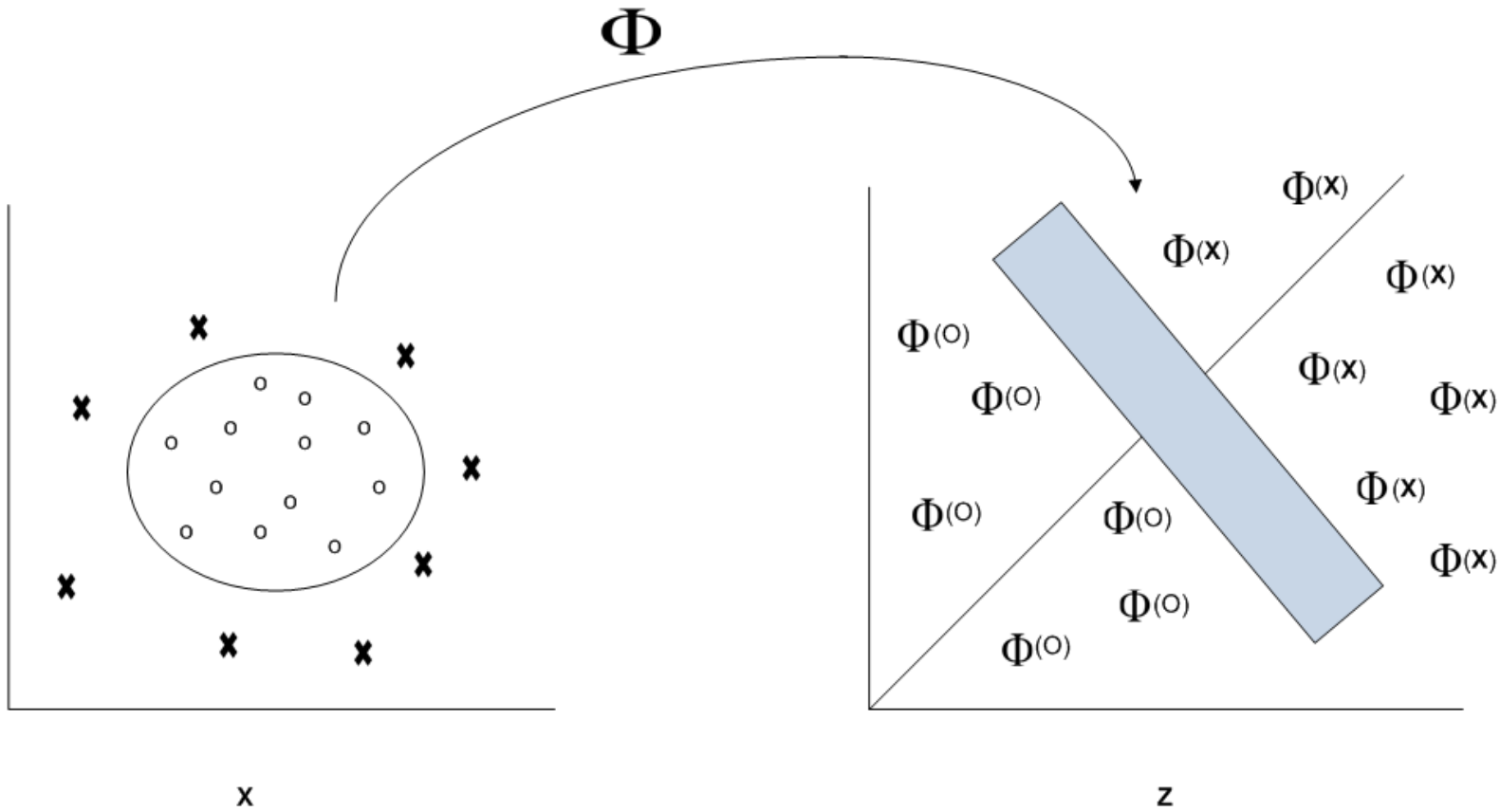
Kernel Trick

- We have already covered an idea that training data which are not linearly separable, can be transformed into a higher dimensional feature space such that in higher dimensional transformed space a hyperplane can be decided to separate the transformed data and hence original data(also see Fig.16).
- Clearly the data on the left in the figure is not linearly separable.

  Yet if we map it to a 3D space using ϕ and the mapped data, then it is possible to have a decision boundaries and hence hyperplane in 3D space.

Kernel Trick

Figure 16: SVM classifier in transformed feature space



Example: SVM classifier for non-linear data

- Suppose, there are a set of data in R^2(i.e., in 2-D space), ϕ is the mapping for $X \epsilon R^2$ to $Z(z_1, z_2, z_3) \epsilon R^3$, in the 3-D space.

$$R^2 \Rightarrow X(x_1, x_2)$$

$$R^3 \Rightarrow Z(z_1, z_2, z_3)$$

Example: SVM classifier for non-linear data

- $$\phi(X) \Rightarrow Z$$

$$z_1 = x_1^2, z_2 = \sqrt{2}x_1x_2, z_3 = x_2^2$$

- The hyperplane in R^2 is of the form

$$w_1x_1^2 + w_2\sqrt{2}x_1x_2 + w_3x_2^2 = 0$$

- Which is the equation of an ellipse in 2D.

Example: SVM classifier for non-linear data

- After the transformation, ϕ as mentioned above, we have the decision boundary of the form

$$w_1 z_1 + w_2 z_2 + w_3 z_2 = 0$$

- This is clearly a linear form in 3-D space. In other words, $W.x + b = 0$ in R^2 has a mapped equivalent $W.z + b' = 0$ in R^3

- This means that data which are not linearly separable in 2-D are separable in 3-D, that is, non linear data can be classified by a linear SVM classifier.

Kernel Trick

- The above can be generalized in the following.
  - **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:**

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

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

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

Kernel Trick

- Now, question here is how to choose ϕ, the mapping function $X \Rightarrow Z$, so that linear SVM can be directly applied to.
- A breakthrough solution to this problem comes in the form of a method as the kernel trick.
- We discuss the kernel trick in the following.
- We know that (.) dot product is often regarded as a measure of similarity between two input vectors.
- For example, if X and Y are two vectors, then

$$X.Y = |X||Y|cos\theta$$

- Here, similarity between X and Y is measured as cosine similarity.
- If θ=0 (i.e., cosθ=1), then they are most similar, otherwise orthogonal means dissimilar.

Kernel Trick

- Analogously, if X_i and X_j are two tuples, then $X_i.X_j$ is regarded as a measure of similarity between X_i and X_j.
- Again, $\phi(X_i)$ and $\phi(X_j)$ are the transformed features of X_i and X_j, respectively in the transformed space; thus, $\phi(X_i).\phi(X_j)$ is also should be regarded as the similarity measure between $\phi(X_i)$ and $\phi(X_j)$ in the transformed space.
- This is indeed an important revelation and is the basic idea behind the kernel trick.
- Now, naturally question arises, if both measures the similarity, then what is the correlation between them (i.e., $X_i.X_j$ and $\phi(X_i).\phi(X_j)$).
- Let us try to find the answer to this question through an example.

Example: Correlation between $X_i.X_j$ and $\phi(X_i).\phi(X_j)$

- Without any loss of generality, let us consider a situation stated below.

$$\phi : R^2 \Rightarrow R^3 = x_1^2 \Rightarrow z, x_2^2 \Rightarrow z_2, \sqrt{2}x_1x_2 \Rightarrow z_3 \tag{40}$$

- Suppose, $X_i = [x_{i1}, x_{i2}]$ and $X_j = [x_{j1}, x_{j2}]$ are any two vectors in R^2.

- Similarly, $\phi(X_i) = [x_{i1}^2, \sqrt{2}.x_{i1}.x_{i2}, x_{i2}^2]$ and $\phi(X_j) = [x_{j1}^2, \sqrt{2}.x_{j1}.x_{j2}, x_{j2}^2]$ are two transformed version of X_i and X_j but in R^3.

Example: Correlation between $X_i.X_j$ and $\phi(X_i).\phi(X_j)$

- Now,

$$\phi(X_i).\phi(X_j) = [x_{i1}^2, \sqrt{2}.x_{i1}.x_{i2}, x_{i2}^2] \begin{bmatrix} x_{j1}^2 \\ \sqrt{2}x_{j1}x_{j2} \\ x_{j2}^2 \end{bmatrix}$$

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

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

$$= \{[x_{i1}, x_{i2}] \begin{bmatrix} x_{j1} \\ x_{j2} \end{bmatrix}\}^2$$

$$= (X_i.X_j)^2$$

Kernel Trick : Correlation between $X_i.X_j$ and $\phi(X_i).\phi(X_j)$

- With reference to the above example, we can conclude that $\phi(X_i).\phi(X_j)$ are correlated to $X_i.X_j$.
- In fact, the same can be proved in general, for any feature vectors and their transformed feature vectors.
- A formal proof of this is beyond the scope of this course.
- More specifically, there is a correlation between dot products of original data and transformed data.
- Based on the above discussion, we can write the following implications.

$$X_i.X_j \Rightarrow \phi(X_i).\phi(X_j) \Rightarrow K(X_i, X_j) \tag{41}$$

- Here, $K(X_i, X_j$ denotes a function more popularly called as kernel function

Kernel Trick : Significance

- This kernel function $K(X_i, X_j)$ physically implies the similarity in the transformed space (i.e., nonlinear similarity measure) using the original attribute X_i, X_j. In other words, K, the similarity function compute a similarity of both whether data in original attribute space or transformed attribute space.
- **Implicit transformation**
  - The first and foremost significance is that we do not require any ϕ transformation to the original input data at all. This is evident from the following re-writing of our SVM classification problem.

$$\textit{Classifier}: \ \delta(X) = \sum_{i=1}^{n} \lambda_i . Y_i . K(X_i, X) + b \tag{42}$$

$$\textit{Learning}: \ \textit{maximize} \sum_{i=1} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i . \lambda_j Y_i Y_j . K(X_i . X_j) \tag{43}$$

$$\textit{Subject to } \lambda_i \geq 0 \textit{ and } \sum^{n} \lambda_i . Y_i = 0 \tag{44}$$

Kernel Trick : Significance

- **Computational efficiency:**
  - Another important significance is easy and efficient computability.
  - We know that in the discussed SVM classifier, we need several and repeated round of computation of dot products both in learning phase as well as in classification phase.
  - On other hand, using kernel trick, we can do it once and with fewer dot products.
  - This is explained in the following.

Kernel Trick : Significance

- We define a matrix called design matrix(X), which contains all data and gram matrix (K), which contains all dot products are as follows:

$$\textit{Design matrix}: \; X = \begin{bmatrix} X_1 \\ X_2 \\ . \\ . \\ . \\ X_n \end{bmatrix}_{n \times N} \tag{45}$$

  where n denotes the number of training data in N-dimensional data space.
- Note that X contains all input data in the original attribute space.

Kernel Trick : Significance

$$Gram\ matrix: \ K = \begin{bmatrix} X_1^T.X_1 & X_1^T.X_2 & \dots & X_1^T.X_n \\ X_2^T.X_1 & X_2^T.X_2 & \dots & X_2^T.X_n \\ \vdots & \vdots & \vdots & \vdots \\ X_n^T.X_1 & X_n^T.X_2 & \dots & X_n^T.X_n \end{bmatrix}_{n \times n} \quad (46)$$

- Note that K contains all dot products among all training data and as $X_i^T.X_j = X_j^T.X_i$. We, in fact, need to compute only half of the matrix.
- More elegantly all dot products are mere in a matrix operation, and that is too one operation only.

Kernel Trick : Significance

In nutshell, we have the following.

- Instead of mapping our data via ϕ and computing the dot products, we can accomplish everything in one operation.
- Classifier can be learnt and applied without explicitly computing $\phi(X)$.
- Complexity of learning depends on n (typically it is $O(n^3)$), not on N, the dimensionality of data space.
- All that is required is the kernel $K(X_i, X_j)$
- Next, we discuss the aspect of deciding kernel functions.

Kernel Functions

Before going to learn the popular kernel functions adopted in SVM classification, we give a precise and formal definition of kernel $k(X_i, X_j)$.

Definition 10.1:

A kernel function $K(X_i, X_j)$ is a real valued function defined on R such that there is another function $\phi : X \rightarrow Z$ such that $K(X_i, X_j) = \phi(X_i).\phi(X_j)$. Symbolically we write $\phi : R^m \rightarrow R^n : K(X_i, X_j) = \phi(X_i).\phi(X_j)$ where $n > m$.

Kernel Functions : Example

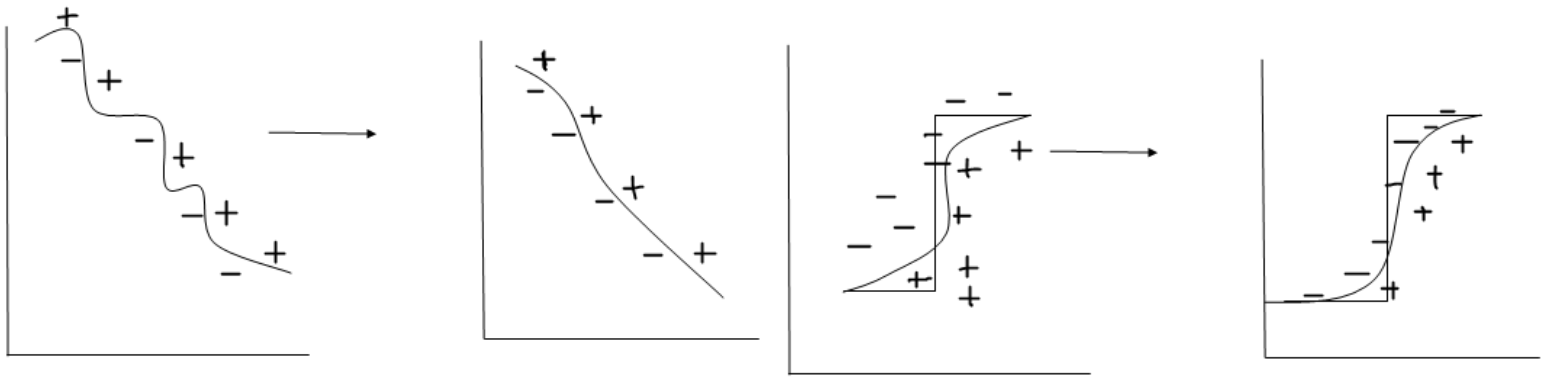
Table 2: Some standard kernel functions

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

Kernel Functions : Example

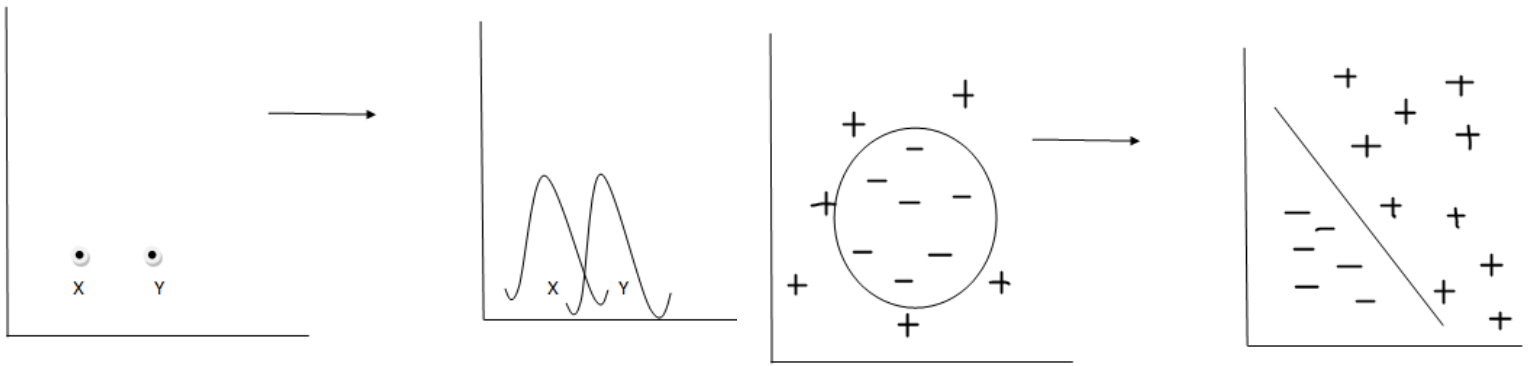
- Different kernel functions follow different parameters. Those parameters are called magic parameters and to be decided a priori.
- Further, which kernels to be followed also depends on the pattern of data as well as prudent of user.
- In general, polynomial kernels result in a large dot products, Gaussian RBF produces more support vectors than other kernels.
- To have an intuitive idea of kernels to be used in non-linear data handling is shown in Fig. 17.

Kernel Functions : Example



(a) Polynomial kernel

(b) Sigmoid kernel



(c) Laplacian kernel

(d) Gaussian RBF kernel

Figure 17: Visual interpretation of few kernel functions.

Mercers Theorem: Properties of Kernel Functions

- Other than the standard kernel functions, we can define of our own kernels as well as combining two or more kernels to another kernel.
- It is interesting to note the following properties. If K_1 and K_2 are two kernels then,

  1. K_1+c
  2. aK_1
  3. aK_1+bK_2
  4. $K_1.K_2$

  are also kernels. Here, a, b and $c \epsilon R^+$.

Mercers Theorem: Properties of Kernel Functions

- Another requirement for kernel function used in non-linear SVM is that there must exist a corresponding transformation such that the kernel function computed for a pair of vectors is equivalent to the dot product between the vectors in the transformed space.
- This requirement can be formally stated in the form of Mercer's theorem.

Mercers Theorem: Properties of Kernel Functions

Theore 10.1: Mercer's Theorem

A kernel function K can be expressed as $K(X, Y) = \phi(X).\phi(Y)$, if and only if, for any function $g(x)$ such that $\int g(x)^2.dx$ is finite then

$$\int K(X, Y)g(x)g(y)dxdy \geq 0$$

- The kernels which satisfy the Mercer's theorem are called Mercer kernels.
- Additional properties of Mercer's kernels are:

Mercers Theorem: Properties of Kernel Functions

- **Symmetric:** $K(X, Y) = K(Y, X)$
- **Positive definite:** $a^T.K.\alpha \geq 0$ for all $\alpha \epsilon R^N$ where K is the $n \times n$ gram matrix.
- It can be prove that all the kernels listed in Table 2 are satisfying the (i) kernel properties, (ii) Mercer's theorem and hence (iii) Mercer kernels.
- **Practice**Prove that $k(x, z) = X - X^T z$ is not a valid kernel.

Characteristics of SVM

1. 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.
2. SVM is the best suitable to classify both linear as well as non-linear training data efficiently.
3. SVM can be applied to categorical data by introducing a suitable similarity measures.
4. 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.