Support Vector Machines

Support Vector Machines (**SVM)**

- SVM were introduced by Vladimir Vapnik (Vapnik, 1995). Technique based on statistical learning theory.
- The main objective in SVM is to find the hyperplane which separates the d-dimensional data points perfectly into two classes.
- However, since example data is often not linearly separable, SVM's introduce the notion of a “kernel induced feature space" which casts the data points (input space) into a higher dimensional feature space where the data is separable.
- SVM works well with higher dimensional data and thus avoids dimensionality problem.
- SVM's higher-dimensional space doesn't need to be dealt with directly which eliminates overfitting.

Support Vector Machines (**SVM**)

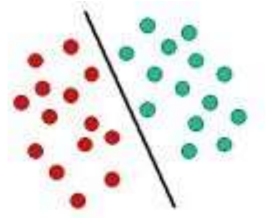
We discuss

- A classification technique when training data are linearly separable known as a Linear SVM
- A classification technique when training data are linearly non-separable known as a Non-linear SVM.
- Maximum margin of hyperplane is also key point

Support Vector Machines - Linear classifier

- Classification tasksare based on drawing separating lines to distinguish between objects of different class labels are known as hyperplane classifiers.

- A decision plane is one that separates between a set of objects having different class labels.

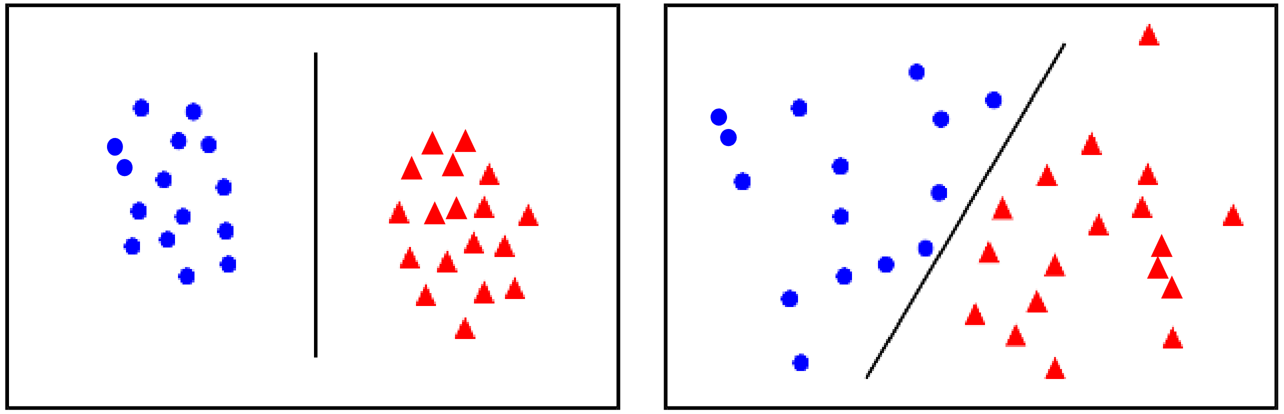


- Any new object falling to the right is labeled, i.e., classified, as GREEN (or classified as RED should it fall to the left of the separating line).

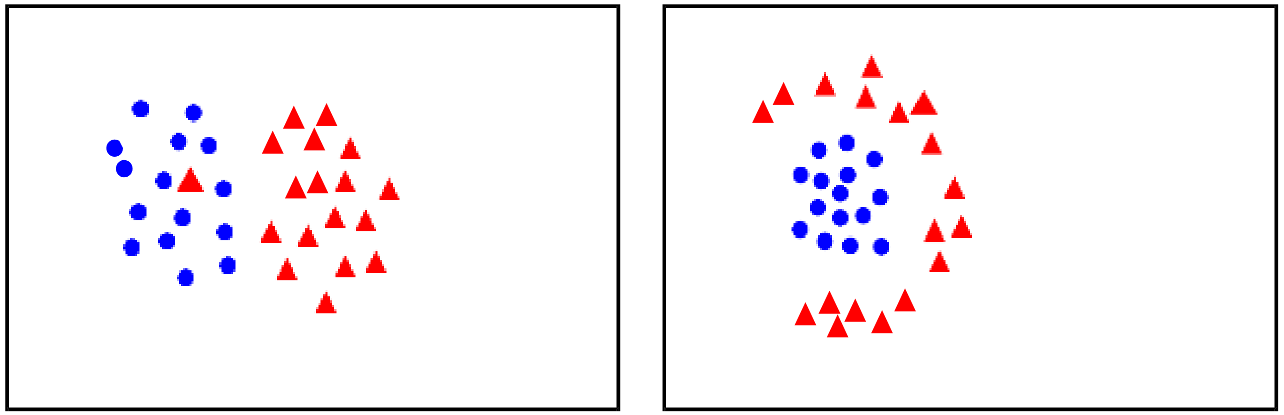
- The objects closest to the hyperplane is called support vectors

Linear separability

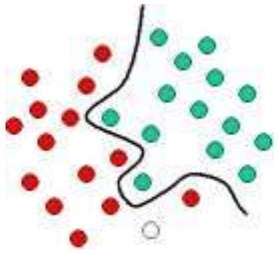
linearly separabl e

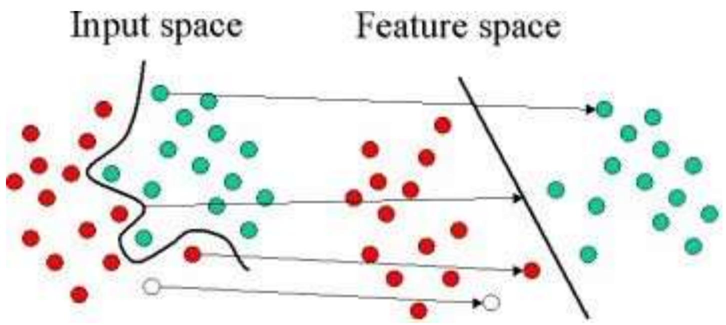


not linearly separabl e



Input Space to Feature Space





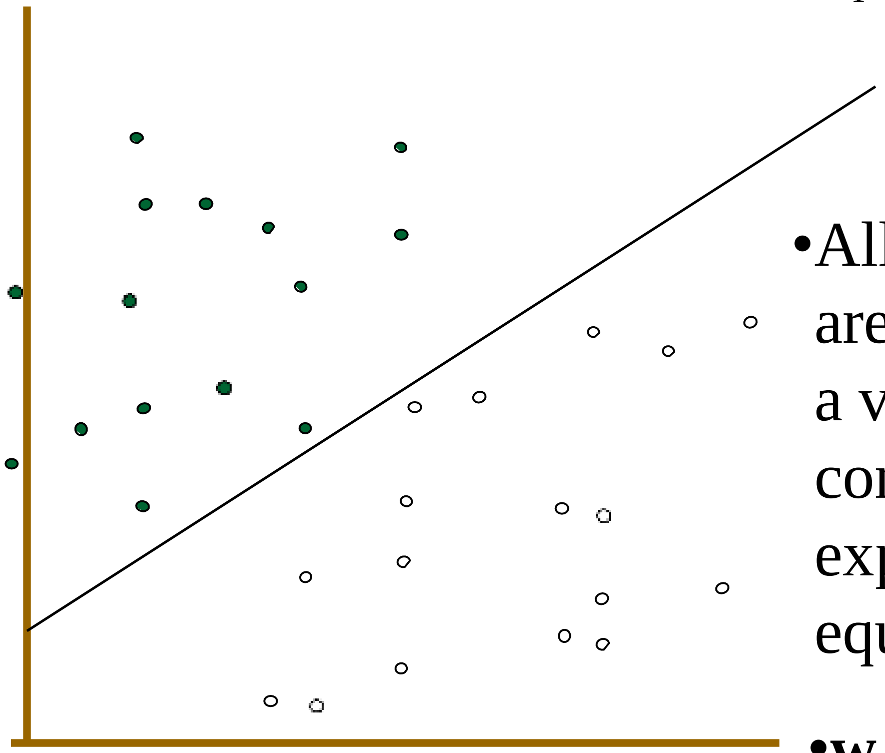
- The original objects are transformed, using a set of mathematical functions, known as kernels.
- Instead of constructing the complex curve, we find an optimal line that can separate the objects.

Linear Classifiers

We are given l training examples $\{x_i, y_i\}$; i = 1.. l , where each example has d inputs ($x_i \in \mathbf{R}^d$), and a class label with one of two values ($y_i \in \{-1, 1\}$.

● denotes +1

○ denotes -1



- All hyperplanes in $\mathbf{R}^d$ are parameterized by a vector (**w**) and a constant (b), expressed using the equation $\mathbf{w} \cdot \mathbf{x} + b = 0$
- **w** is the vector orthogonal to the hyperplane

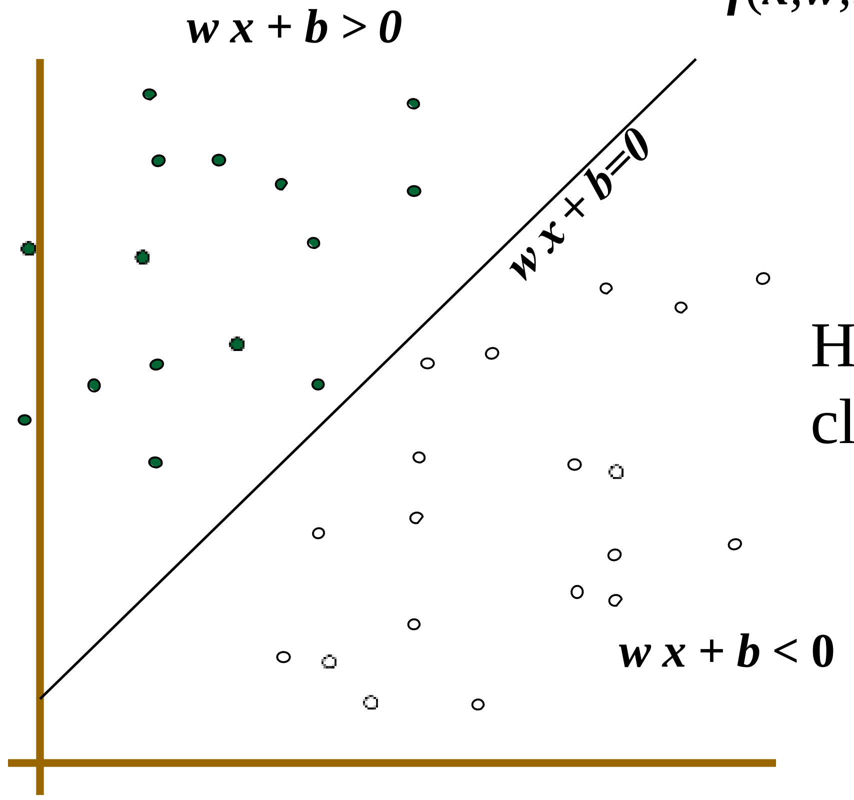
- Given such a hyperplane (**w**,b) that separates the data, using function $f(x) = \text{sign}(\mathbf{w}\cdot \mathbf{x} + b)$

Linear Classifiers

● denotes +1

○ denotes -1

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



How would you classify this data?

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.

Hyperplane and Classification

- $W.X + b = 0$, the equation can be explain as follows.
  - 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.

Max Margin of Linear Classifiers

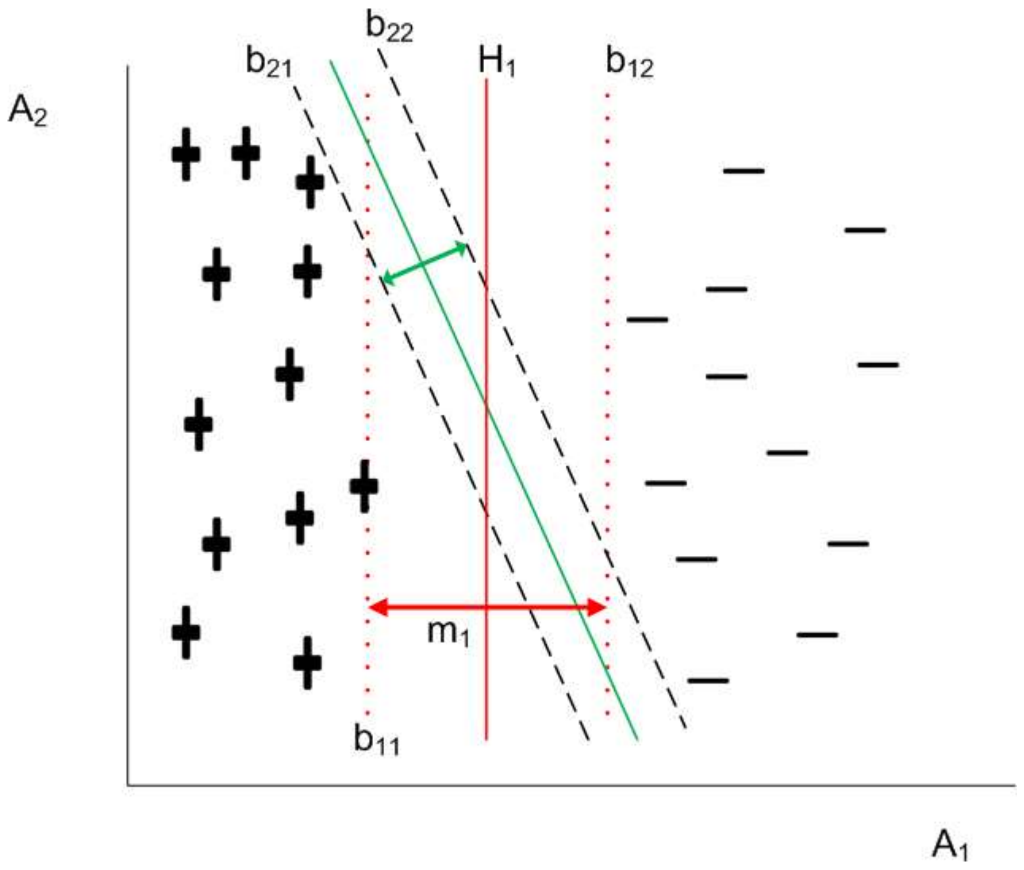
Two problem may arise:

- Whether all hyperplanes are equivalent so far the classification of data is concerned?
- If not, which hyperplane is the best?

- Regarding Classification error (with training data), all of them are with zero error.
- But there is no guarantee that all hyperplanes perform equally well on unseen (i.e., test) data.

- For a good classifier, choose one of the infinite number of hyperplanes, so that it performs better not only on training data but as well as test data.

Hyperplanes with decision boundaries and their margins.



Maximum Margin Hyperplane

- Two hyperplanes H_1 and H_2 have 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. If data is classified using Hyperplane H_1, then it is with larger margin then using Hyperplane H_2.
- The larger the margin, lower is the classification error.

The Perceptron Classifier

Given linearly separable data $\mathbf{x}_i$ labelled into two categories $y_i = \{-1,1\}$, find a weight vector **w** such that the discriminant function

$$f(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x}_i + b$$

separates the categories for i = 1, .., N

- how can we find this separating hyperplane ?

The Perceptron Algorithm

Write classifier as $f(\mathbf{x}_i) = \tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}_i + w_0 = \mathbf{w}^\top \mathbf{x}_i$

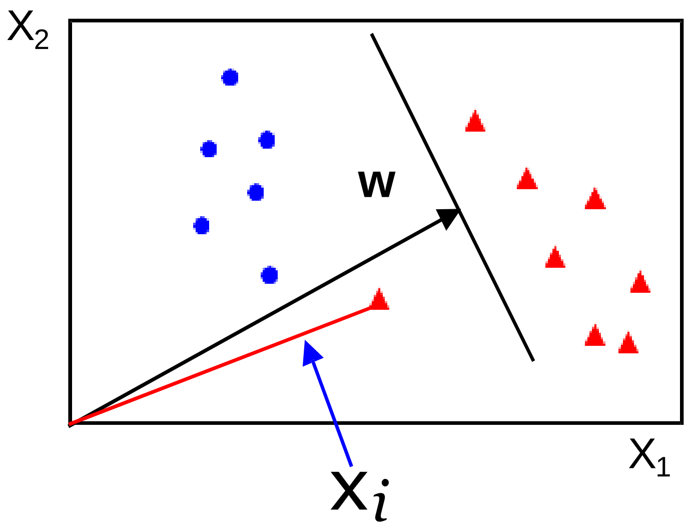
$$\text{where } \mathbf{w} = (\tilde{\mathbf{w}}, w_0), \mathbf{x}_i = (\tilde{\mathbf{x}}_i, 1)$$

- Initialize **w** = 0
- Cycle though the data points { $\mathbf{x}_i$, y_i }
- if $\mathbf{x}_i$ is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha \, \text{sign}(f(\mathbf{x}_i)) \, \mathbf{x}_i$
- Until all the data is correctly classified

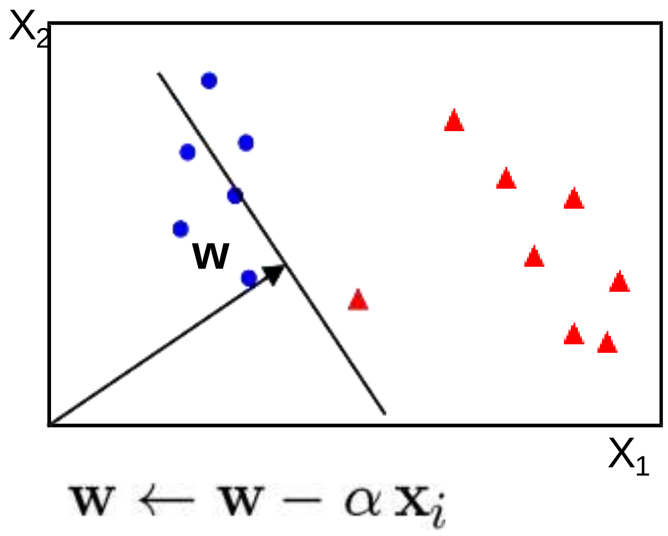
For example in 2D

- Initialize **w** = 0
- Cycle though the data points { $\mathbf{x}_i$, y_i }
  - if $\mathbf{x}_i$ is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha\, \text{sign}(f(\mathbf{x}_i))\, \mathbf{x}_i$
- Until all the data is correctly classified

before update

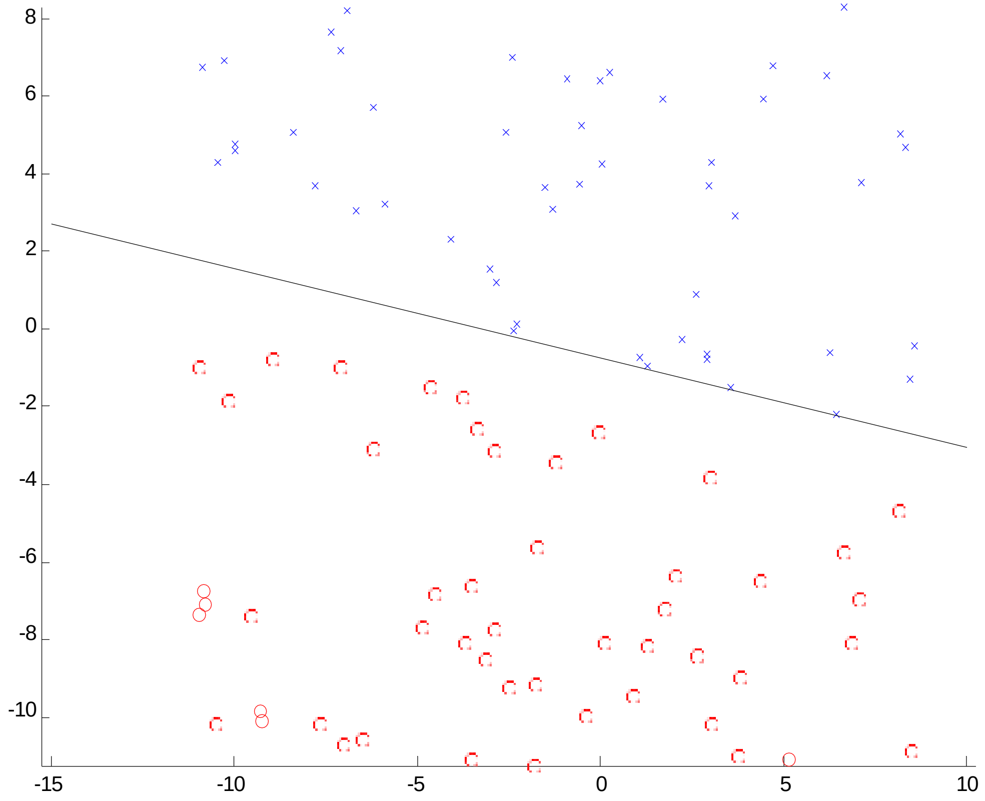


after Update



$$\mathbf{w} \leftarrow \mathbf{w} - \alpha\, \mathbf{x}_i$$

Perceptron example



- if the data is linearly separable, then the algorithm will converge
- convergence can be slow …
- separating line close to training data
- we would prefer a larger margin for generalization

Hyperplane Classifier

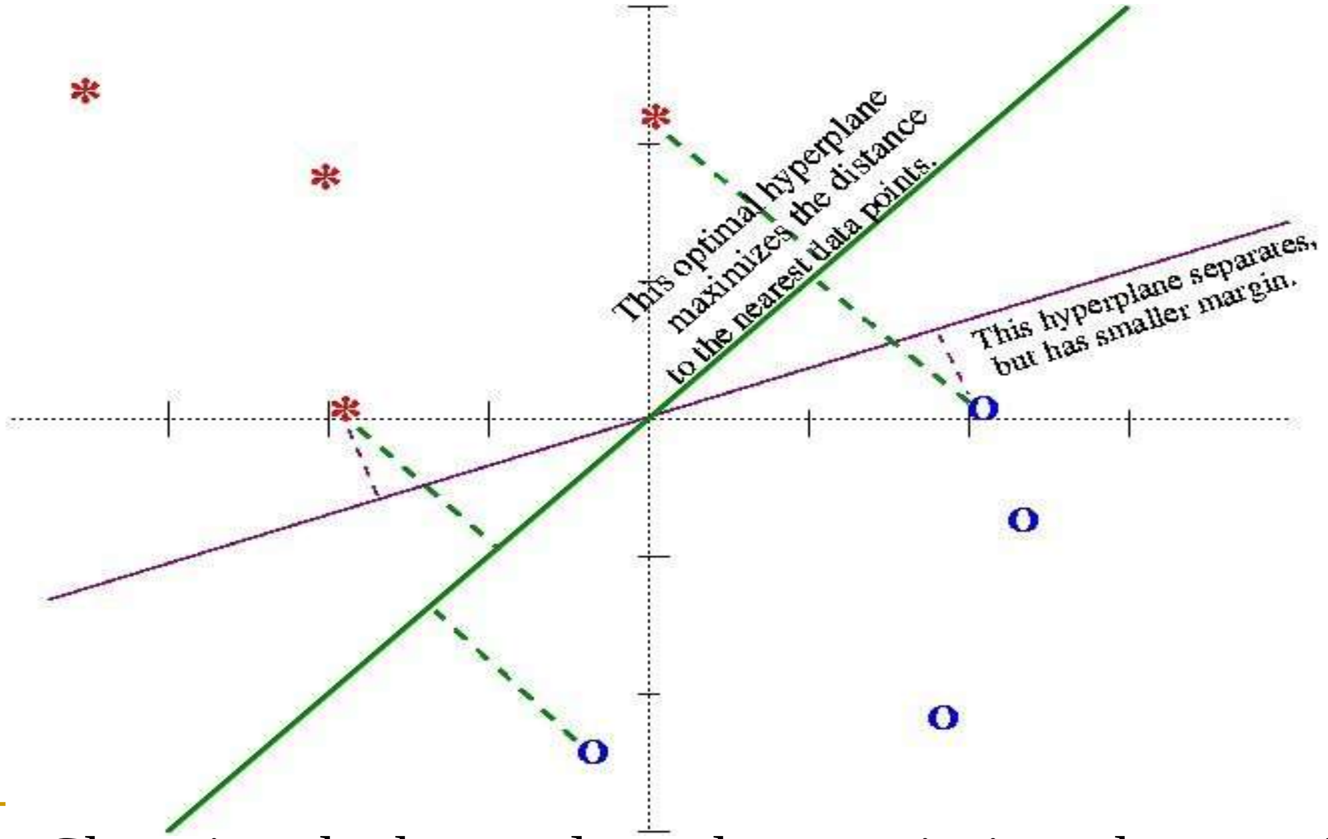
- To avoid such confusions
- A given hyperplane represented by ($\mathbf{w}$,b) is equally expressed by all pairs $\{\lambda\mathbf{w}, \lambda b\}$ for $\lambda \in R^{+}$.
- We define the hyperplane which separates the data from the hyperplane by a “distance” so that at least one example on both sides has a distance of exactly 1.
- That is, we consider those that satisfy:
- $\mathbf{w} \, . \, \mathbf{x_i} + b \geq 1$ when $y_i = +1$
- $\mathbf{w} \, . \, \mathbf{x_i} + b \leq 1$ when $y_i = -1$

$$y_i\,(\mathbf{w} \, . \, \mathbf{x_i} + b) \geq 1 \qquad \forall\, i$$

- To obtain the geometric distance from the hyperplane to a data point, we normalize by the magnitude of **w**.

- We want the hyperplane that maximizes the geometric distance to the closest data points.

$$d((\mathbf{w}, b), \mathbf{x_i}) = [y_i(\mathbf{w}.\mathbf{x_i} + b)] / \|\mathbf{w}\| \geq 1 / \|\mathbf{w}\|$$


This optimal hyperplane maximizes the distance to the nearest data points.
This hyperplane separates, but has smaller margin.

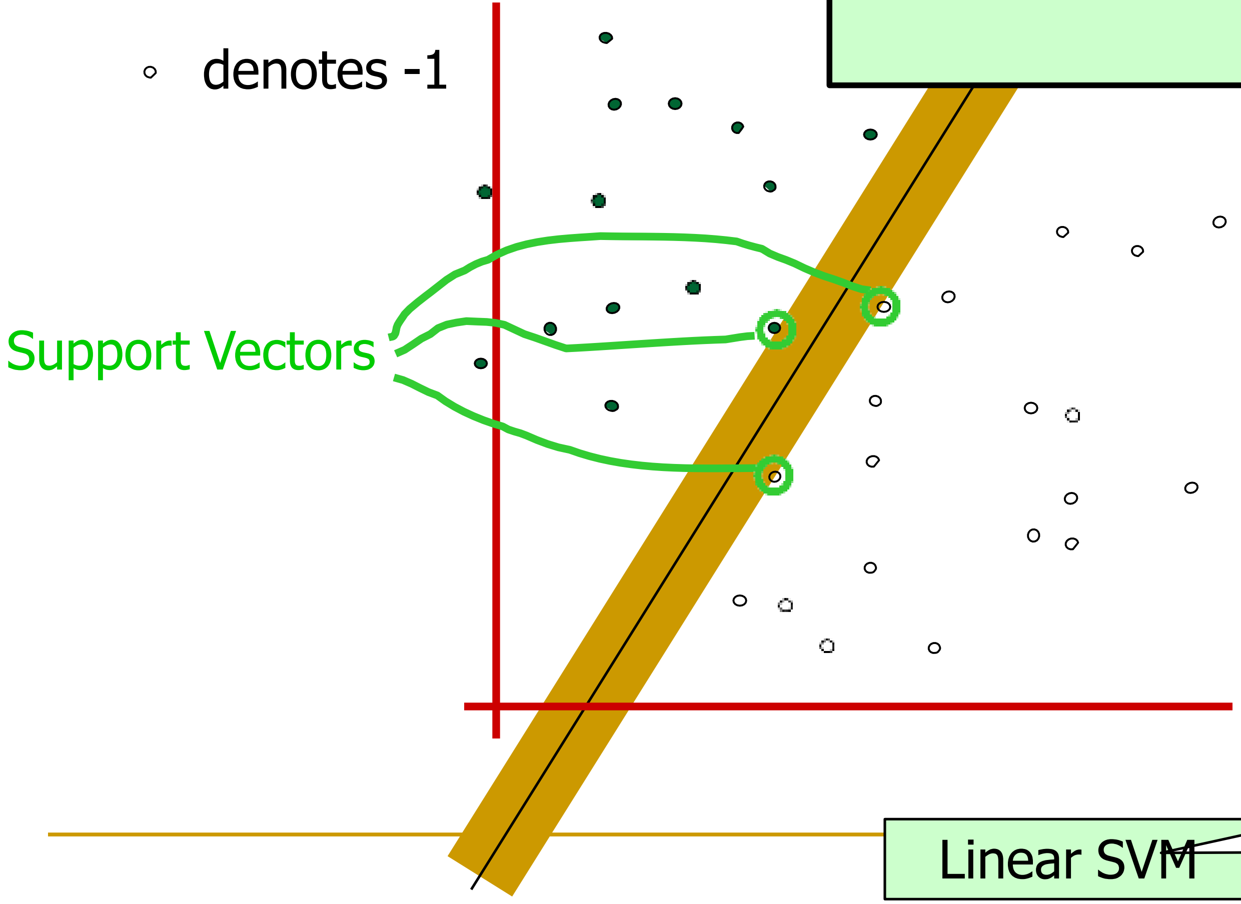
Choosing the hyperplane that maximizes the margin

Maximum Margin

1. Maximizing the margin
2. support vectors are important

● denotes +1

○ denotes -1



The maximum margin linear classifier

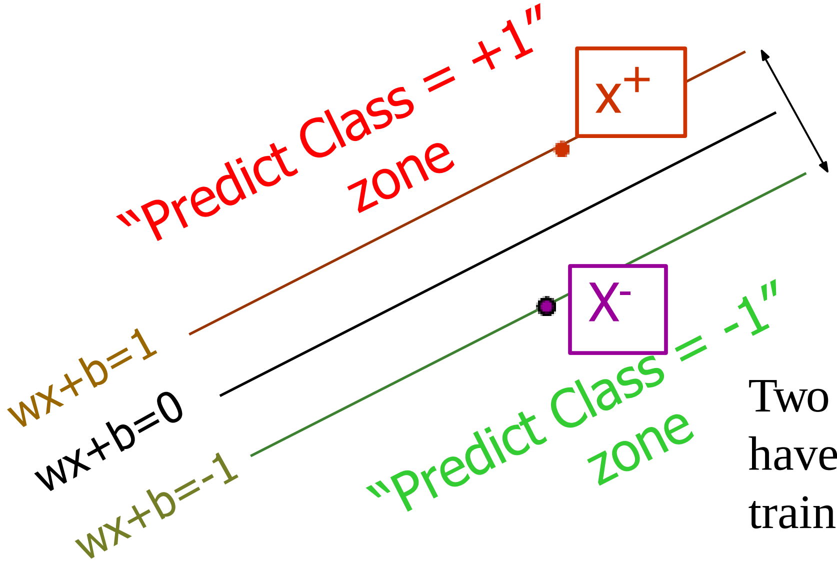
This is the simplest kind of SVM (Called an LSVM)

Linear SVM

Linear SVM for Linearly Not Separable Data

- 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.
- .

Linear SVM Mathematically



M=Margin Width

Two hyperplanes are parallel (they have the same normal) and that no training points fall between them.

What we know:

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

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

Linear SVM Mathematically

- Goal: 1) Correctly classify all training data

$$\left.\begin{aligned} wx_i + b &\geq 1 \quad \text{if } y_i = +1 \\ wx_i + b &\leq 1 \quad \text{if } y_i = -1 \end{aligned}\right\}$$

$$y_i(wx_i + b) \geq 1 \quad \text{for all } i$$

2) Maximize the Margin $M = \frac{2}{\|\mathbf{w}\|}$ or same as minimize $\frac{1}{2} w^t w$

- We can formulate a constrained optimization Problem and solve for **w** and b

- Minimize $\Phi(w) = \frac{1}{2} w^t w$

  subject to $\forall i \quad y_i(wx_i + b) \geq 1$

Searching for MMH

- 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 Multipliers

- Consider a problem: $min_x\, f(x)$ subject to $h(x) = 0$

- We define the Lagrangian $L\,(x, \alpha) = f(x) - \alpha\, h\,(x)$

- α is called "Lagrange multiplier"

- Solve: $\min_x \max_\alpha L\,(x, \alpha)$ subject to $\alpha \geq 0$

Original Problem:

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

Construct the Lagrangian Function for optimization

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i [y^{(i)}(w^T x^{(i)} + b) - 1]$$ S. T. $\alpha_i \geq 0; \ \forall_i$

Our goal is to: $max_{\alpha \geq 0}\, min_{w,b}\, \mathcal{L}(w,b,\alpha)$ OR $min_{w,b}\, max_{\alpha \geq 0}\, \mathcal{L}(w,b,\alpha)$

$$\frac{\partial}{\partial w}\mathcal{L}(w,b,\alpha) = w - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} = 0$$

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$$

The derivative with respect to b

$$\frac{\partial}{\partial b} C(w,b,\alpha) = \sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

Substituting we get:

$$C(w,b,\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)} - b \sum_{i=1}^{m} \alpha_i y^{(i)}$$

$$= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)}$$

$\max_{\boldsymbol{\alpha}}$: $\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)} . x^{(j)})$

Subject to $0 \leq \alpha_i \leq C \;\; for \; \forall i = 1 \ldots m$ and $\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$

$\boldsymbol{\alpha}$ is the vector of *m* non-negative Lagrange multipliers to be determined, and C is a constant

Optimal hyperplane : $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i x_i$

- The vector **w** is just a linear combination of the training examples.

$$C(w, b, \alpha) = \frac{1}{2}\mathbf{w}.\mathbf{w} - \sum_i \alpha_i \left[\left(\mathbf{w}.\mathbf{x}_i + b \right) y_i - 1 \right]$$

$$\alpha_i \geq 0, \ \forall_i$$

$$y_i \left(\vec{w} \cdot \vec{x}_i + b \right) = 1 \quad \text{(1)}$$

$$y_i y_i \left(\vec{w} \cdot \vec{x}_i + b \right) = y_i \quad \text{(2)}$$

$$\left(\vec{w} \cdot \vec{x}_i + b \right) = y_i \quad \text{(3)}$$

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$$

$b = y_k - w^T \mathbf{x}_k$ for any $\mathbf{x}_k$ such that $\alpha_k \neq 0$

$$\begin{aligned} w^T x + b &= \left(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}\right)^T x + b \\ &= \sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b. \end{aligned}$$

- If we've found the α_i's, in order to make a prediction, we have to calculate a quantity that depends only on the inner product between x and the points in the training set.
- Each non-zero α_i indicates that corresponding $\mathbf{x_i}$ is a **support vector.**
- Then the classifying function will have the form:

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

- Notice that it relies on an *inner product* between the test point $\mathbf{x}$ and the support vectors $\mathbf{x_i}$.

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:

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

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

Illustration : Linear SVM

Table 1: Training Data

x_1	x_2	y	λ
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

Linear SVM example.

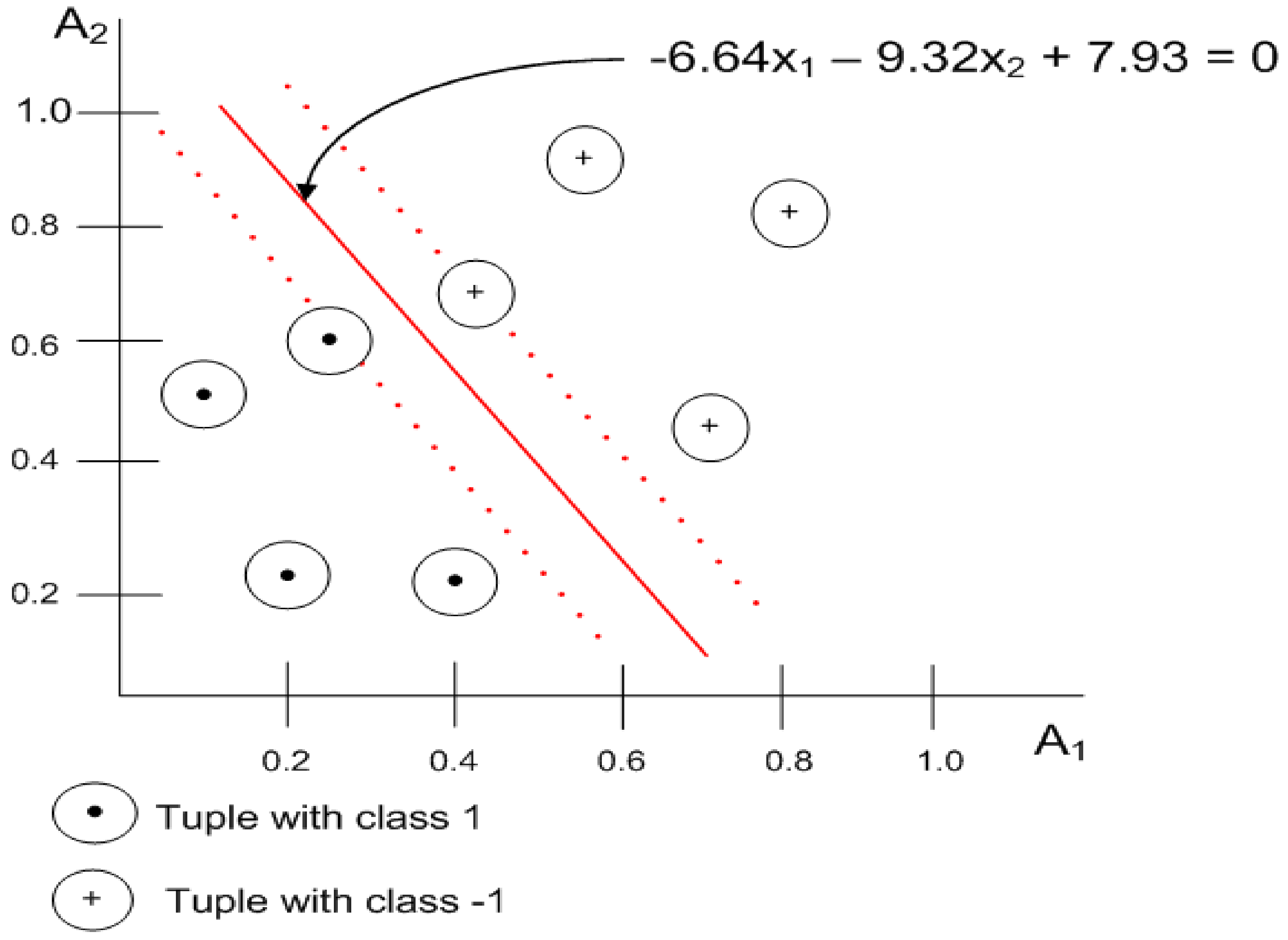


Illustration : Linear SVM

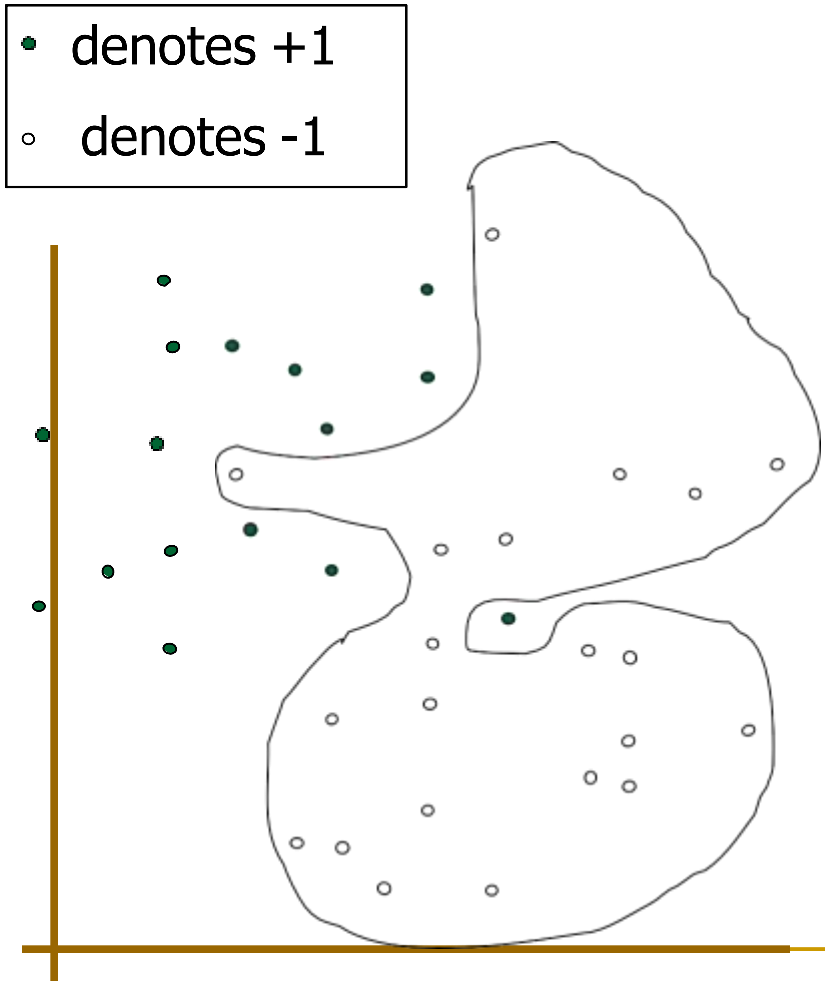
- The parameter b can be calculated for each support vector as follows
  $b_1 = 1 - W.x_1$ // for support vector x_1
  $= 1 - (-6.64) \times 0.38 - (-9.32) \times 0.47$ //using dot product
  $= 7.93$
  $b_2 = 1 - W.x_2$ // for support vector x_2
  $= 1 - (-6.64) \times 0.48 - (-9.32) \times 0.611$ //using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

Dataset with noise



- **Hard Margin:** So far we require all data points be classified correctly.

  - No training error

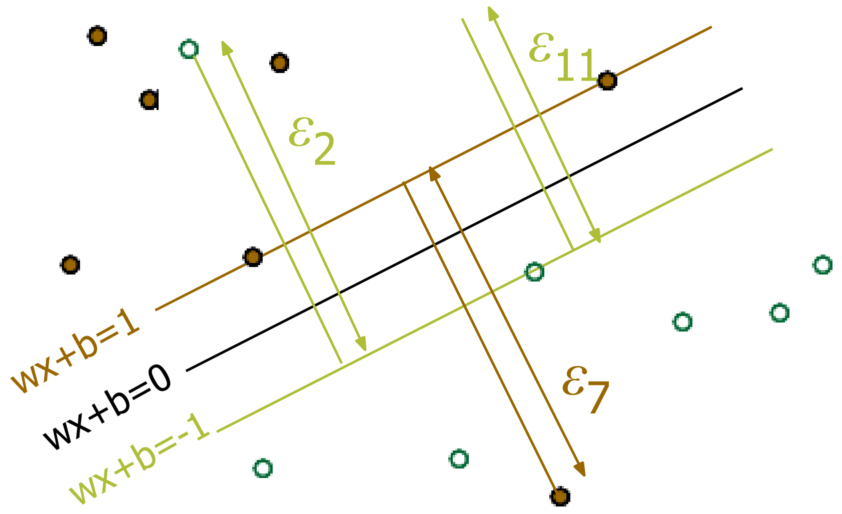
- What if the training set is noisy?

  - Solution 1: use very powerful kernels

OVERFITTING!

Soft Margin Classification

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



What should our quadratic optimization criterion be?

Minimize

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

Hard Margin v.s. Soft Margin

- **The old formulation:**

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\, \mathbf{w}^{\mathrm{T}}\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\, (\mathbf{w^T x_i} + \mathrm{b}) \geq 1$

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

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\, \mathbf{w}^{\mathrm{T}}\mathbf{w} + \lambda \Sigma \xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\, (\mathbf{w^T x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

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

Linear SVMs: Overview

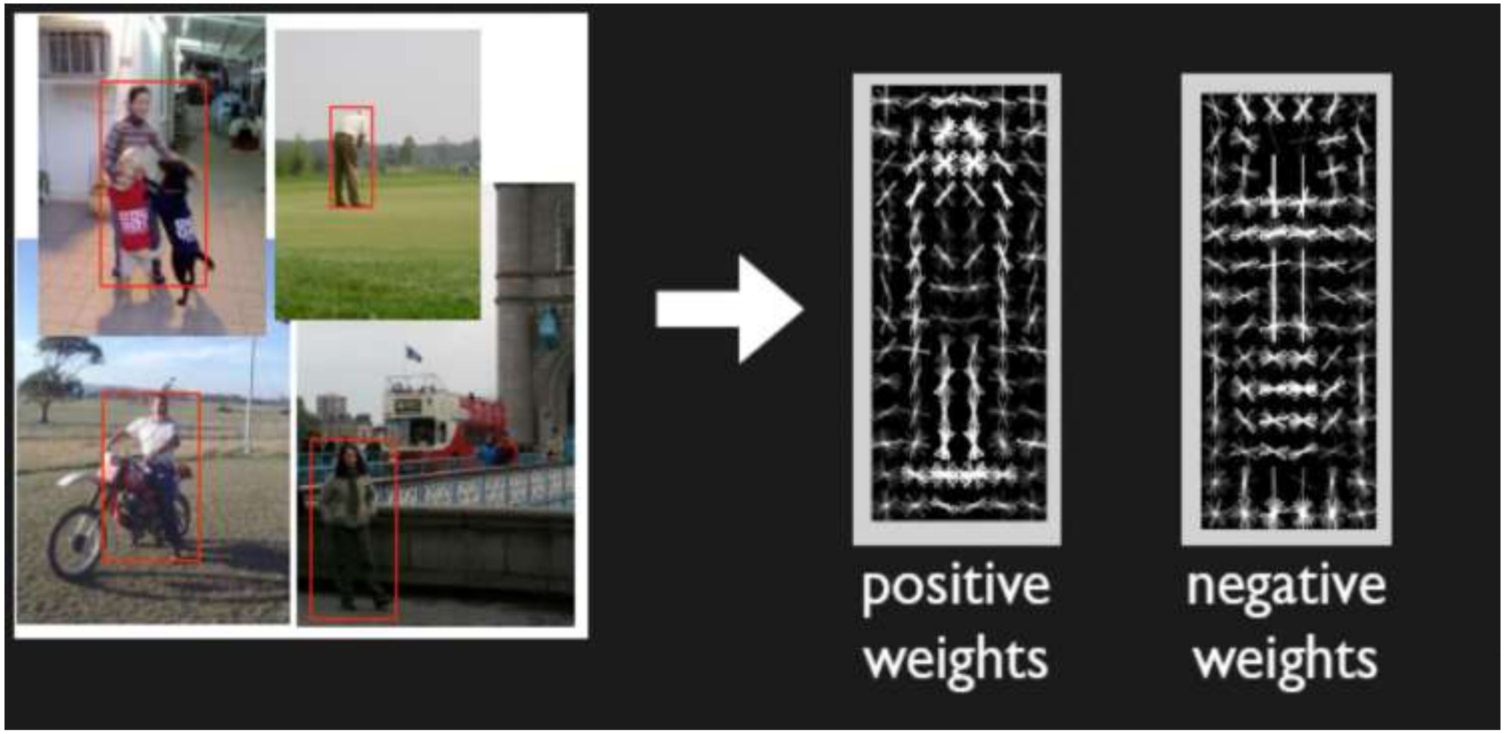
- **The classifier is a *separating hyperplane.***
- **Most "important" training points are support vectors; they define the hyperplane.**
- **Quadratic optimization algorithms can identify which training points x_i are support vectors with non-zero Lagrangian multipliers α_i.**
- **Both in the dual formulation of the problem and in the solution training points appear only inside dot products:**

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

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

Learned model

$$f(\mathrm{x}) \quad = \quad \mathrm{w}^{>}\mathrm{x} + b$$



Slide from Deva Ramanan

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,

  $\delta(X) = W.X + b$

  $= -6.64 \times 0.5 - 9.32 \times 0.5 + 7.93$

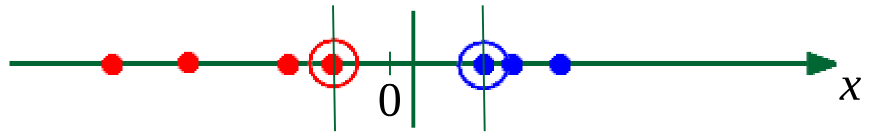
  $= -0.05$

  $= -ve$

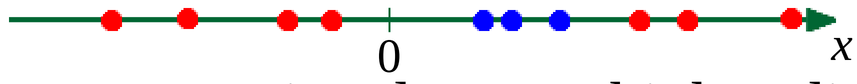
- This implies that the test data falls on or below the MMH and SVM classifies that X belongs to class label -.

Non-linear SVMs

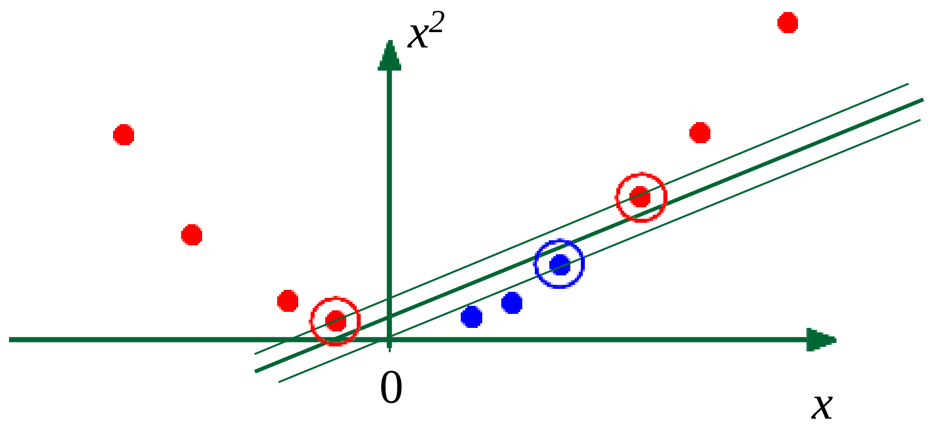
- Datasets that are linearly separable with some noise work out great:



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



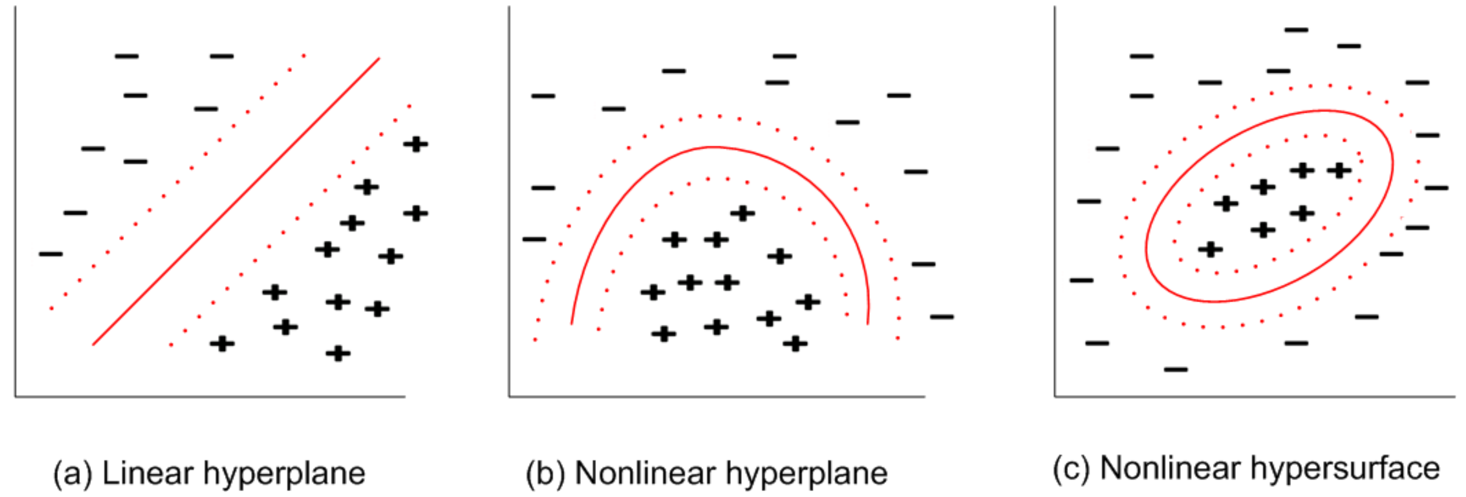
- How about… mapping data to a higher-dimensional space:



Non-Linear SVM

- For understanding this, .
- 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.



Non-Linear SVM

- A hyperplane is expressed as

$$linear: \ w_1x_1 + w_2x_2 + w_3x_3 + c = 0 \tag{30}$$

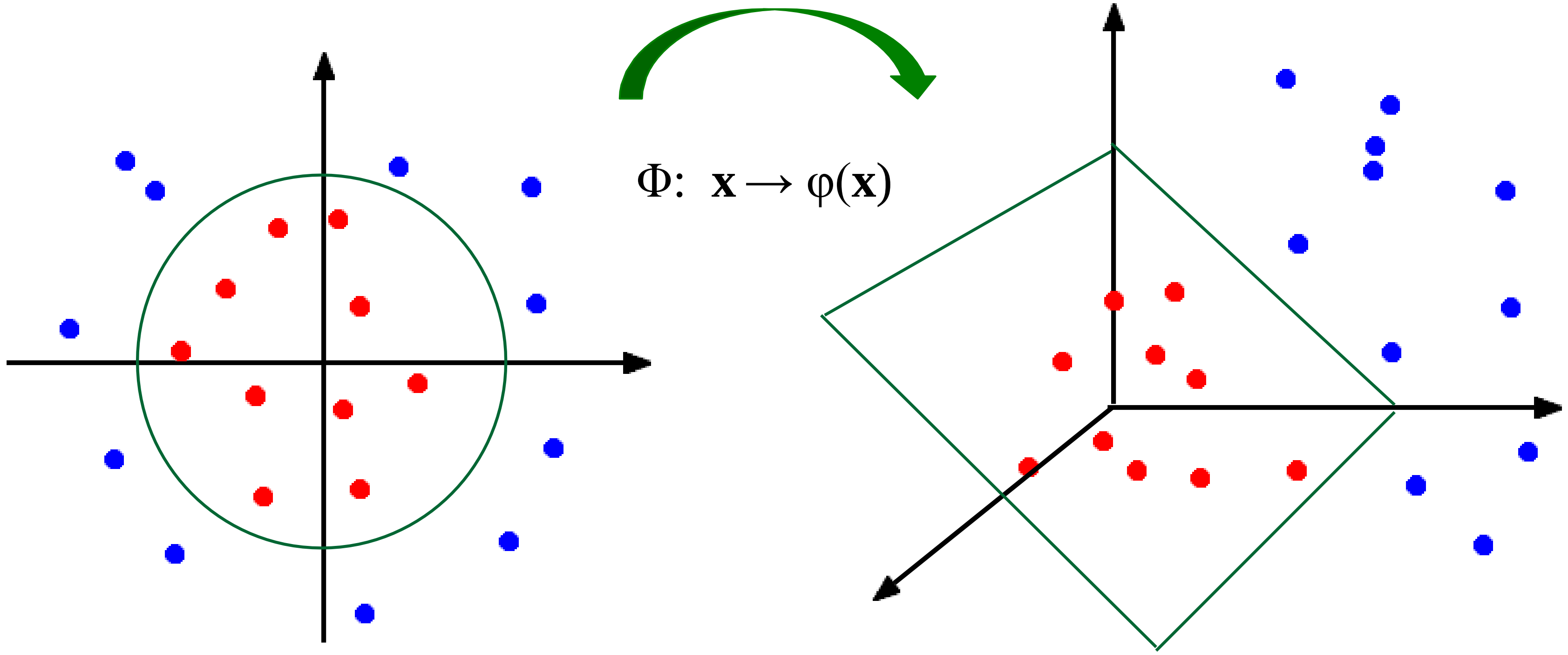
- Whereas a non-linear hypersurface is expressed as.

$$Nonlinear: \ w_1x_1^2 + w_2x_2^2 + w_3x_1x_2 + w_4x_3^2 + w_5x_1x_3 + c = 0 \tag{31}$$

- 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 SVMs: Feature spaces

- General idea: the original input space (nonlinear separable data) can always be mapped to some higher-dimensional feature space where the training set is linearly separable:



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 -).

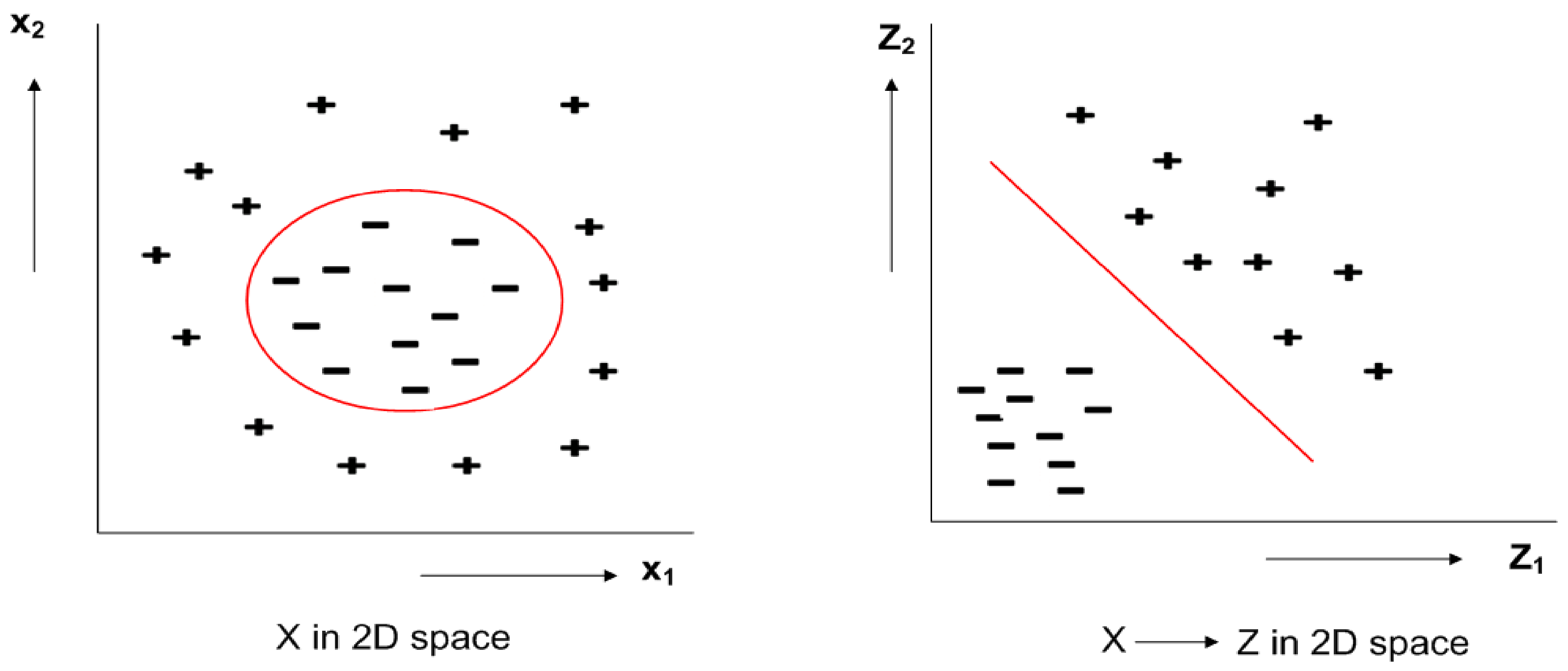


Figure: Non-linear mapping to Linear SVM.

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:

$$X(x_1, x_2) = +1$$

if $$\sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} > 2$$

$$X(x_1, x_2) = -1 \textit{ otherwise}$$

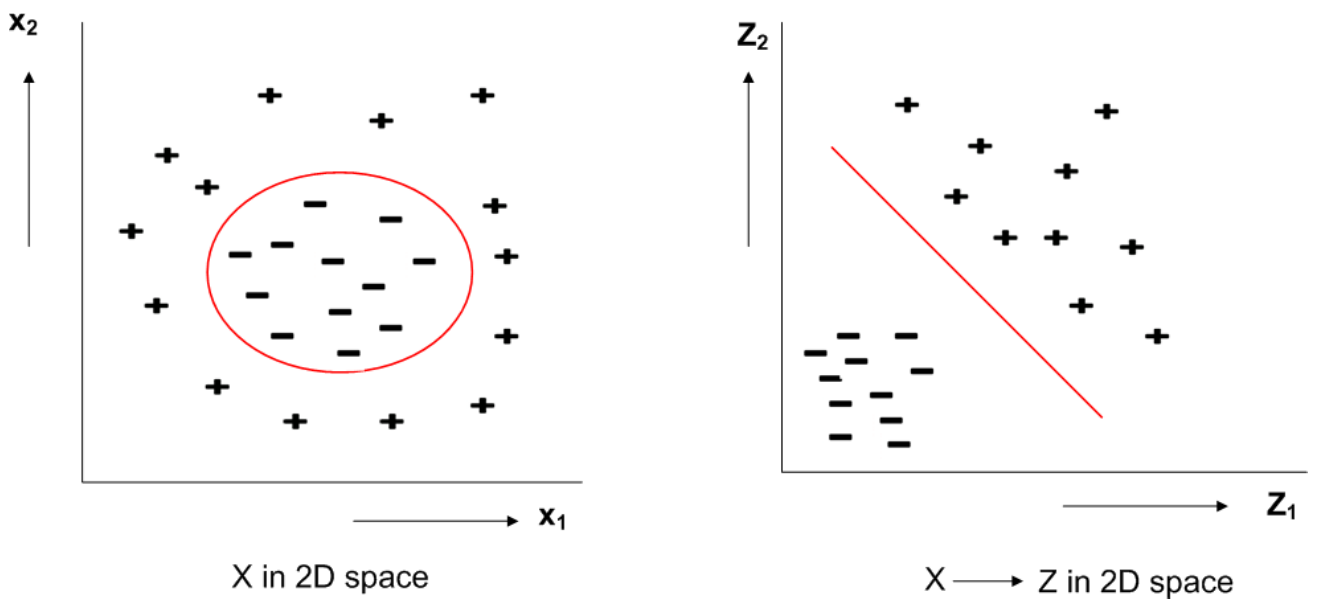
Concept of Non-Linear Mapping

Example: Non-linear mapping to linear SVM

- The Z space when plotted will take view where data are separable with linear boundary,

  $Z : z_1 + z_2 = -0.46$

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

- 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
-

Mapping the Inputs to other dimensions - the use of Kernels

- Finding the optimal curve to fit the data is difficult.
- To “pre-process" the data in such a way that the problem is transformed into one of finding a simple hyperplane.
- We define a mapping $z = \varphi(x)$ that transforms the d-dimensional input vector x into a (usually higher) d*-dimensional vector z.
- To choose a $\varphi()$ so that the new training data $\{\varphi(x_i),y_i\}$ is separable by a hyperplane.
- How do we go about choosing $\varphi()$?

Kernel Trick

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.

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

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

Kernel Trick

Example: SVM classifier for non-linear data

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

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

- 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.

Kernel Trick

Example: SVM classifier for non-linear data

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

$$w_1z_1 + w_2z_2 + w_3z_2 = 0$$

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

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

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

  - **Learning:**

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

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

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, $\varphi(X_i)$ and $\varphi(X_j)$ are the transformed features of X_i and X_j, respectively in the transformed space; thus, $\varphi(X_i).\varphi(X_j)$ is also should be regarded as the similarity measure between $\varphi(X_i)$ and $\varphi(X_j)$ in the transformed space.
- This 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 $\varphi(X_i).\varphi(X_j)$).
- Let us try to find the answer to this question through an example.

Kernel Trick

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

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

$$\varphi : R^2 \Rightarrow R^3 = x_1^2 \Rightarrow z, x_2^2 \Rightarrow z_2, \sqrt{2}x_1x_2 \Rightarrow z_3 \quad (40)$$

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

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

Kernel Trick

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

- Now,

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

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

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

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

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

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

- With reference to the above example, we can conclude that $\varphi(X_i).\varphi(X_j)$ 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.
- 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.

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

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

Kernel Trick : Significance

- **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.
  -

Efficient dot-product of polynomials

Polynomials of degree exactly *d*

d=1

$$\phi(u).\phi(v) = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} . \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = u_1 v_1 + u_2 v_2 = u.v$$

d=2

$$\phi(u).\phi(v) = \begin{pmatrix} u_1^2 \\ u_1 u_2 \\ u_2 u_1 \\ u_2^2 \end{pmatrix} . \begin{pmatrix} v_1^2 \\ v_1 v_2 \\ v_2 v_1 \\ v_2^2 \end{pmatrix} \begin{aligned} &= u_1^2 v_1^2 + 2u_1 v_1 u_2 v_2 + u_2^2 v_2^2 \\ &= (u_1 v_1 + u_2 v_2)^2 \\ &= (u.v)^2 \end{aligned}$$

For any *d* **:**

$$\phi(u).\phi(v) = (u.v)^d$$

- Taking a dot product and exponentiating gives same results as mapping into high dimensional space and then taking dot produce

The "Kernel Trick"

- **The linear classifier relies on dot product between vectors $K(x_i,x_j)=x_i^Tx_j$**
- **If every data point is mapped into high-dimensional space via some transformation Φ: $x \rightarrow \varphi(x)$, the dot product becomes:**

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

- **A *kernel function* is some function that corresponds to an inner product in some expanded feature space.**
- **Example:**

2-dimensional vectors $x=[x_1\ \ x_2]$; let $K(x_i,x_j)=(1+x_i^Tx_j)^2$,

Need to show that $K(x_i,x_j)=\varphi(x_i)^T\varphi(x_j)$:

$$K(x_i,x_j)=(1+x_i^Tx_j)^2,$$
$$= 1+x_{i1}^2x_{j1}^2 + 2\,x_{i1}x_{j1}\,x_{i2}x_{j2} + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$
$$= [1\ \ x_{i1}^2\ \ \sqrt{2}\,x_{i1}x_{i2}\ \ x_{i2}^2\ \ \sqrt{2}x_{i1}\ \ \sqrt{2}x_{i2}]^T[1\ \ x_{j1}^2\ \ \sqrt{2}\,x_{j1}x_{j2}\ \ x_{j2}^2\ \ \sqrt{2}x_{j1}\ \ \sqrt{2}x_{j2}]$$
$$= \varphi(x_i)^T\varphi(x_j),\quad \text{where } \varphi(x) = [1\ \ x_1^2\ \ \sqrt{2}\,x_1x_2\ \ x_2^2\ \ \sqrt{2}x_1\ \ \sqrt{2}x_2]$$

Non-linear SVMs Mathematically

- **Dual problem formulation:**

$$\boxed{\begin{array}{l}\textbf{Find } \alpha_1 \ldots \alpha_N \textbf{ such that} \\ Q(\alpha) = \Sigma\alpha_i - \tfrac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(x_i, x_j) \textbf{ is maximized and} \\ (1)\ \ \Sigma\alpha_i y_i = 0 \\ (2)\ \alpha_i \geq 0 \textbf{ for all } \alpha_i\end{array}}$$

- **The solution is:**

$$\boxed{f(x) = \Sigma\alpha_i y_i K(x_i, x_j) + b}$$

- **Optimization techniques for finding α_i's remain the same!**

Examples of Kernel Functions

- Linear: $K(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$

- Polynomial of power *p*: $K(\mathbf{x}_i,\mathbf{x}_j) = (1 + \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^p$

Produces large dot products. Power *ρ* is specified apriori by the user.

- Gaussian (radial-basis function network):

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

Examples of Kernel Functions

- Laplacian:

$$K(X, Y) = e^{-\lambda||x-y||}$$

- Mahalanobis:

$$K(X, Y) = e^{-(X-y)^T A(x-y)}$$

Followed when statistical test data is known

- Sigmoid: $K(X, Y) = tanh(\beta_0 X^T y + \beta_1)$

Followed when statistical test data is known

Nonlinear SVM - Overview

- SVM locates a separating hyperplane in the feature space and classify points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function
- The kernel function plays the role of the dot product in the feature space.

Properties of SVM

- **Flexibility in choosing a similarity function.**
- **Sparseness of solution when dealing with large data sets**

  -only support vectors are used to specify the separating hyperplane
- **Ability to handle large feature spaces**

  -complexity does not depend on the dimensionality of the feature space
- **Overfitting can be controlled by soft margin approach**
- **Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution**
- **Feature Selection**

Weakness of SVM

- **It is sensitive to noise**

  -A relatively small number of mislabeled examples can dramatically decrease the performance

- **It only considers two classes**

  - how to do multi-class classification with SVM?

  - Answer:

  1) with output arity m, learn m SVM's

  - **SVM 1 learns "Output==1" vs "Output != 1"**
  - **SVM 2 learns "Output==2" vs "Output != 2"**
  - **:**
  - **SVM m learns "Output==m" vs "Output != m"**

  2)To predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

Some Issues

- **Choice of kernel**
  - **- Gaussian or polynomial kernel is default**
  - **- if ineffective, more elaborate kernels are needed**
  - **-domain experts can give assistance in formulating appropriate similarity measures**

- **Choice of kernel parameters**
  - **- e.g. σ in Gaussian kernel**
  - **- σ is the distance between closest points with different classifications**
  - **- In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.**

- **Optimization criterion** – Hard margin v.s. Soft margin
  - **- a lengthy series of experiments in which various parameters are tested**