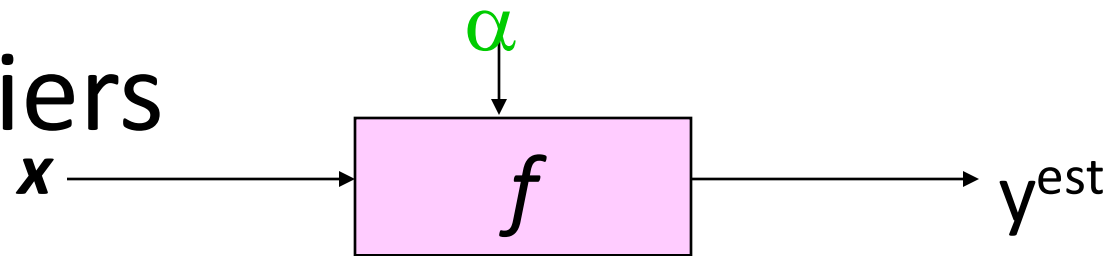


CS60020: Foundations of Algorithm Design and Machine Learning

Sourangshu Bhattacharya

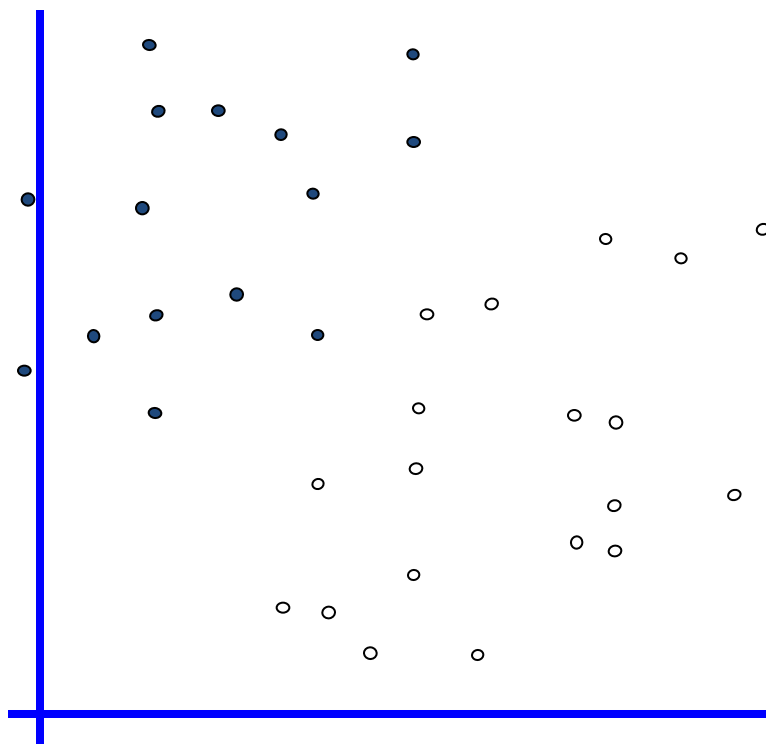
SUPPORT VECTOR MACHINES

Linear Classifiers



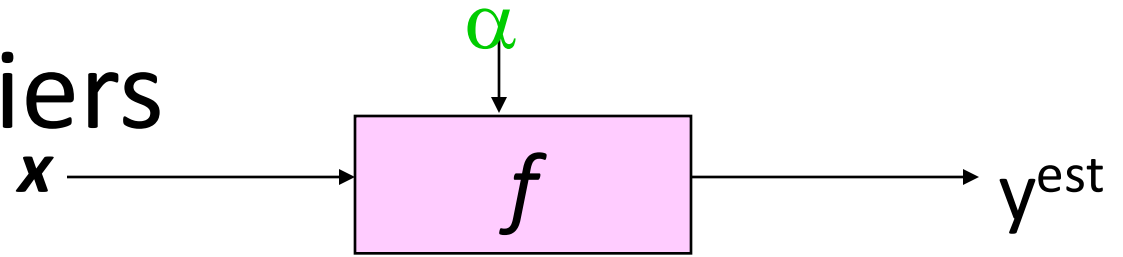
$$f(x, w, b) = \text{sign}(w \cdot x - b)$$

- denotes +1
- denotes -1



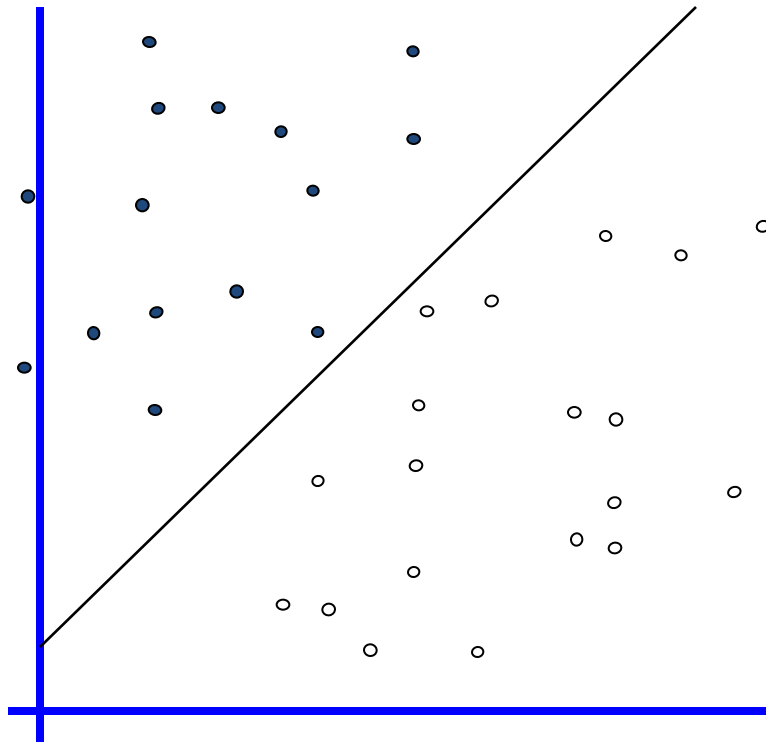
How would you classify this data?

Linear Classifiers



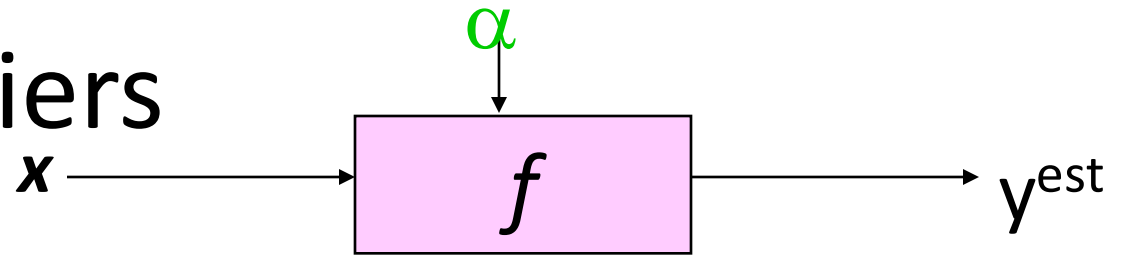
$$f(x, w, b) = \text{sign}(w \cdot x - b)$$

- denotes +1
- denotes -1



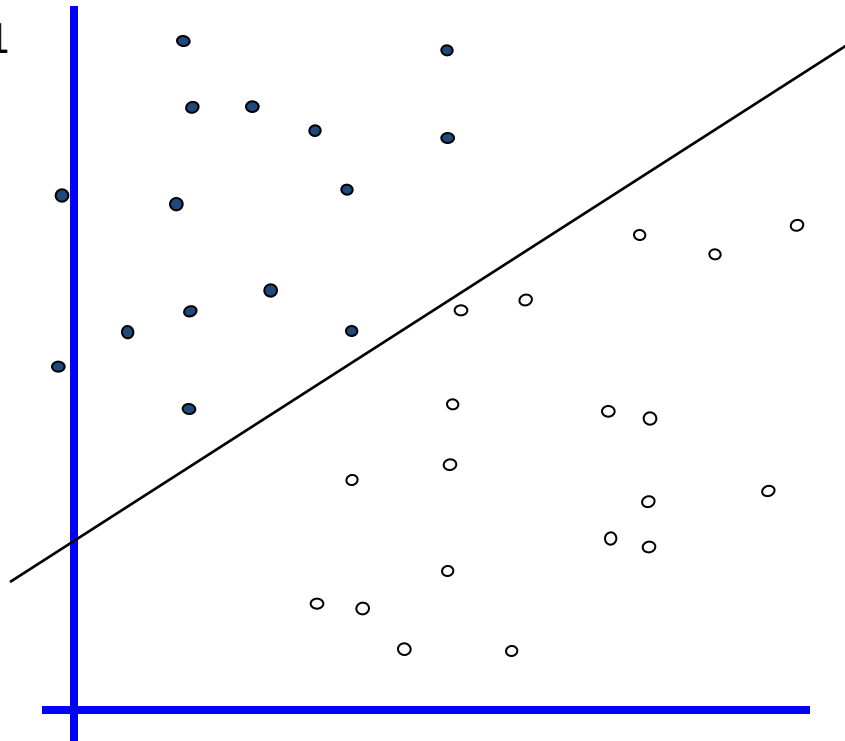
How would you classify this data?

Linear Classifiers



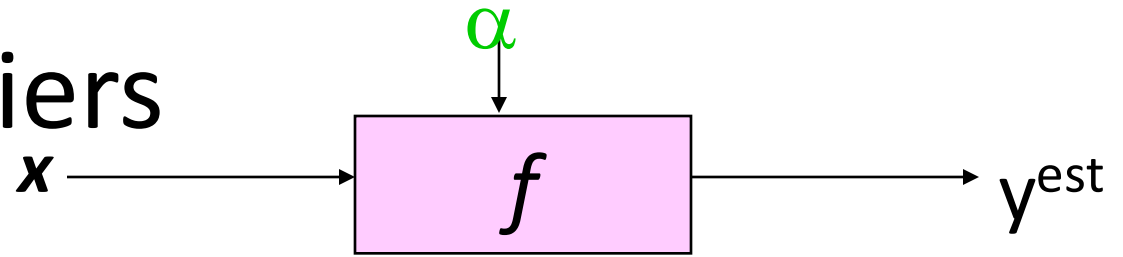
$$f(x, w, b) = \text{sign}(w \cdot x - b)$$

- denotes +1
- denotes -1

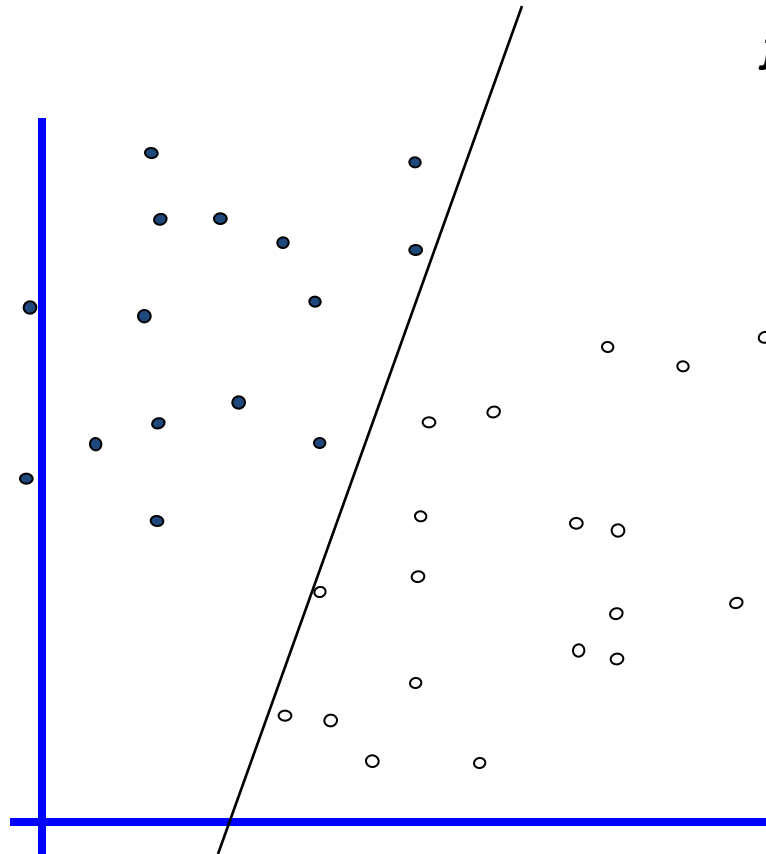


How would you classify this data?

Linear Classifiers



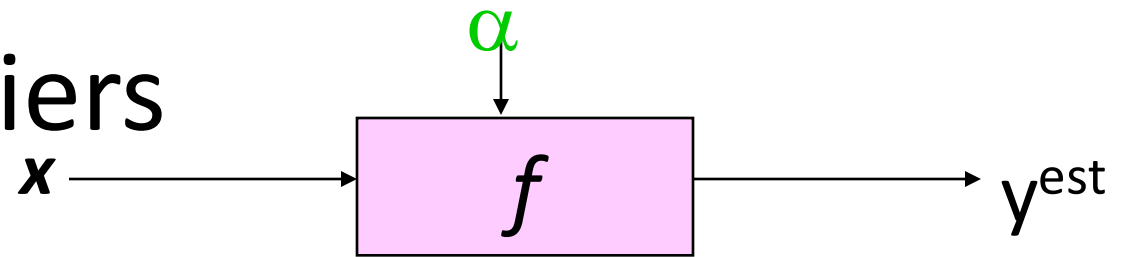
- denotes +1
- denotes -1



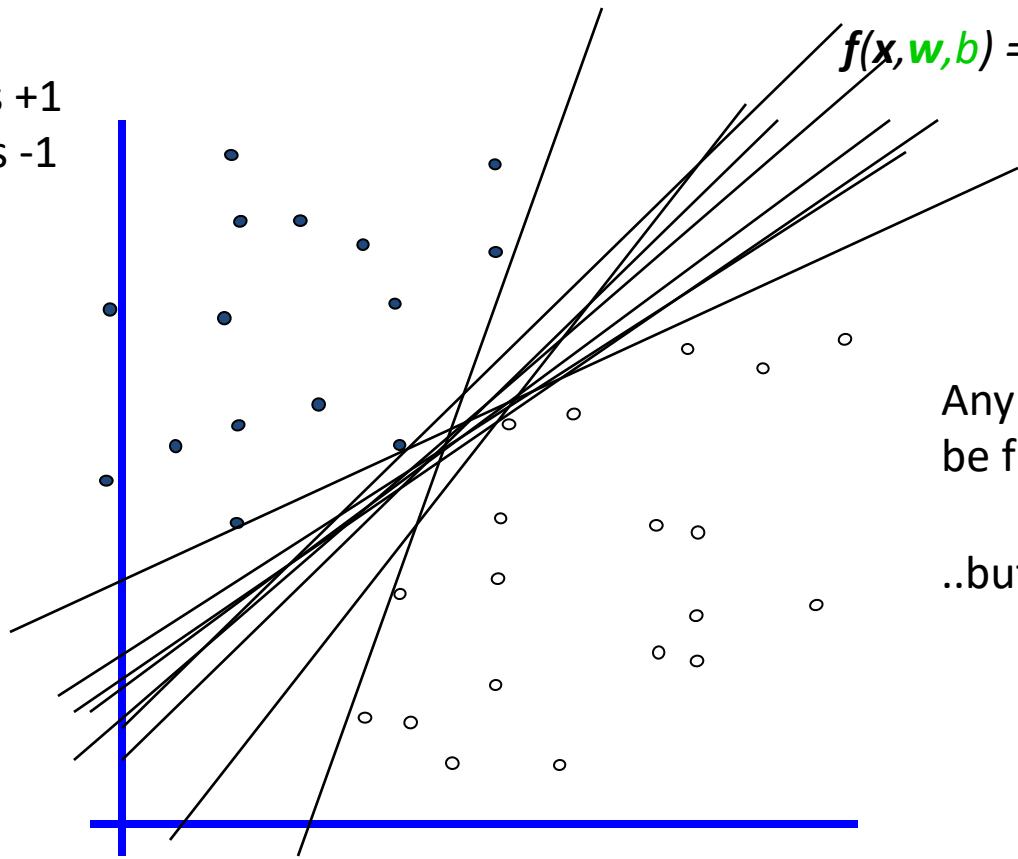
$$f(x, w, b) = \text{sign}(w \cdot x - b)$$

How would you classify this data?

Linear Classifiers



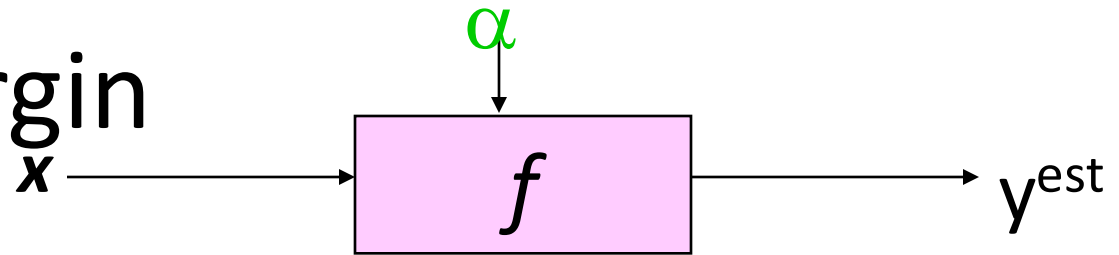
- denotes +1
- denotes -1



Any of these would be fine..

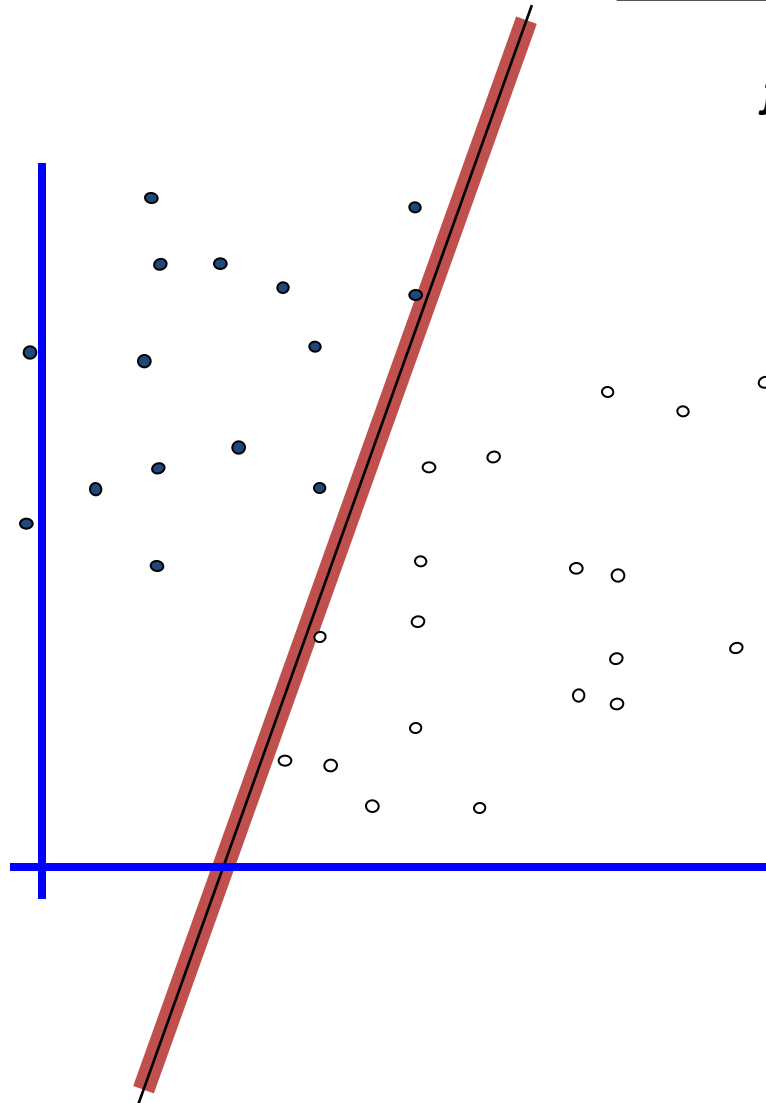
..but which is best?

Classifier Margin



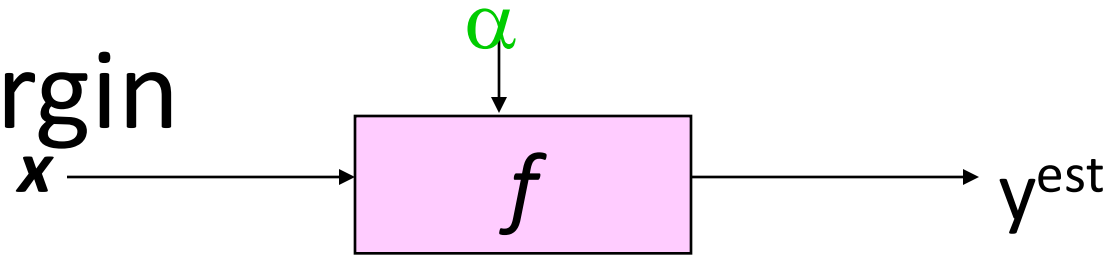
$$f(x, w, b) = \text{sign}(w \cdot x - b)$$

- denotes +1
- denotes -1

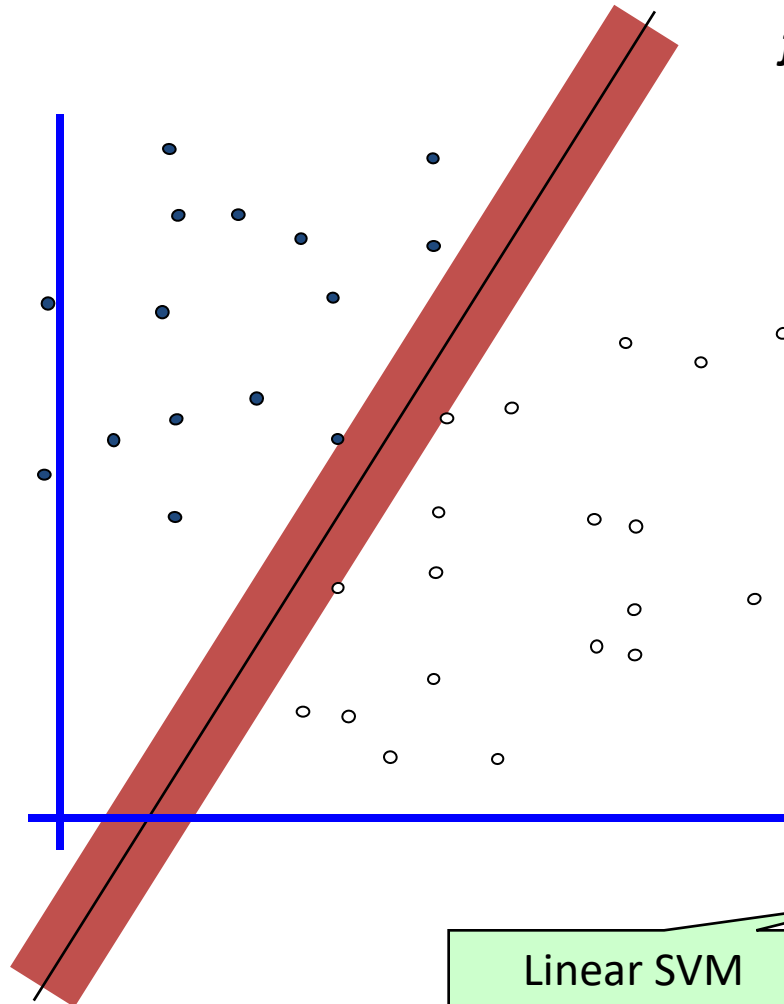


Define the **margin** of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



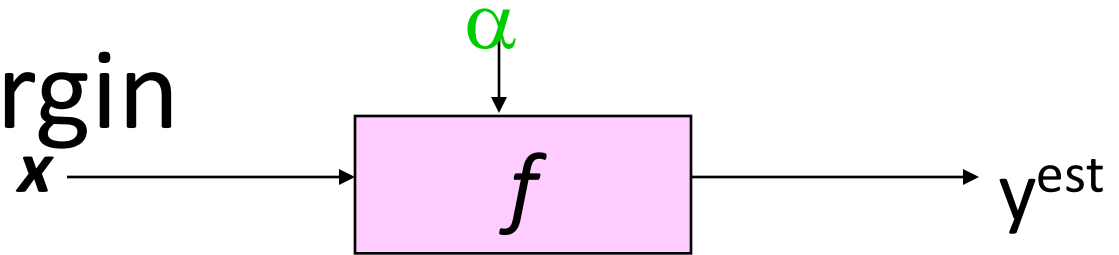
- denotes +1
- denotes -1



$$f(x, w, b) = \text{sign}(w \cdot x - b)$$

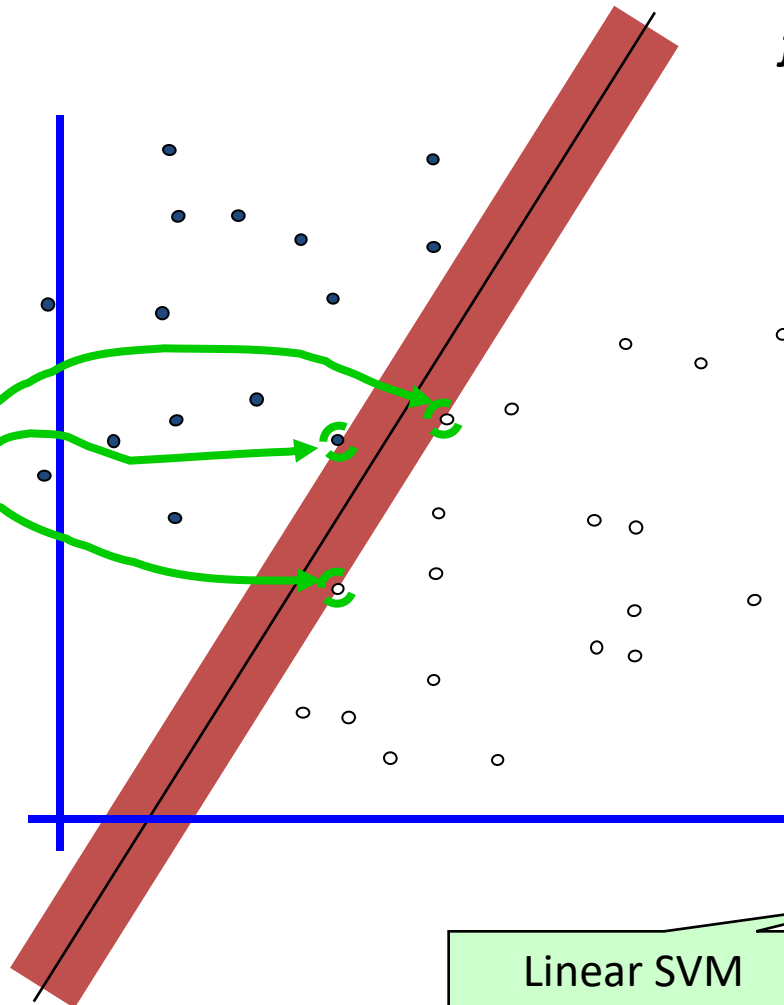
The **maximum margin linear classifier** is the linear classifier with the, um, maximum margin.
This is the simplest kind of SVM (Called an LSVM)

Maximum Margin



- denotes +1
- denotes -1

Support Vectors are those datapoints that the margin pushes up against

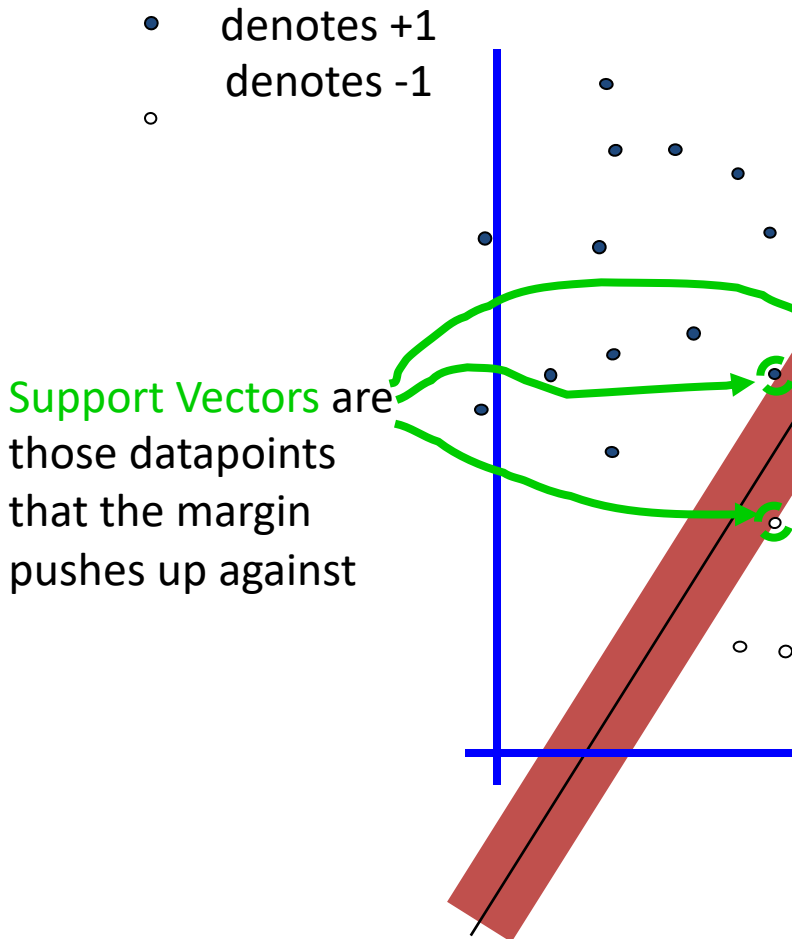


$$f(x, w, b) = \text{sign}(w \cdot x - b)$$

The **maximum margin linear classifier** is the linear classifier with the, um, maximum margin.
This is the simplest kind of SVM (Called an LSVM)

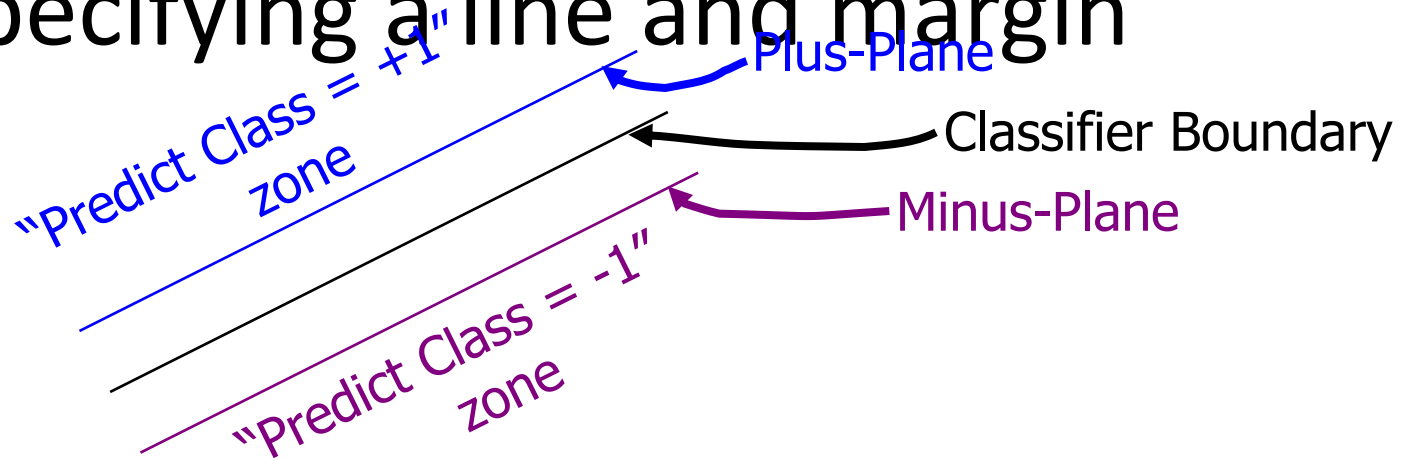
Linear SVM

Why Maximum Margin?



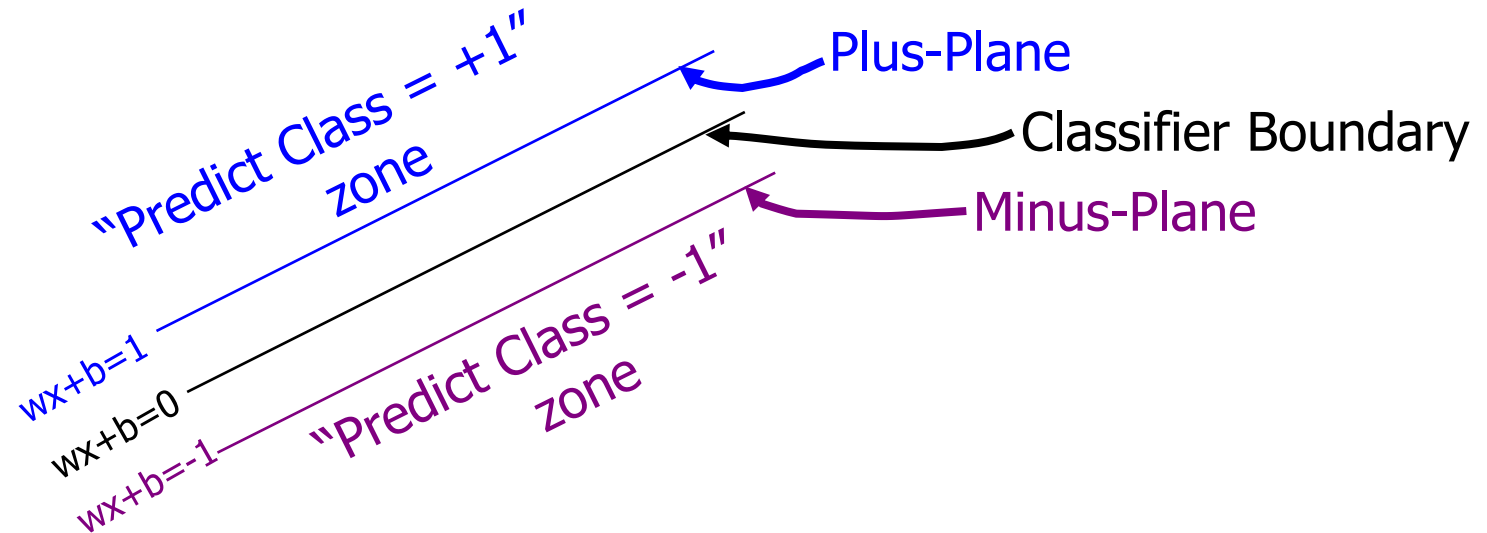
1. Intuitively this feels safest.
2. If we've made a small error in the location of the boundary (it's been jolted in its perpendicular direction) this gives us least chance of causing a misclassification.
3. LOOCV is easy since the model is immune to removal of any non-support-vector datapoints.
4. There's some theory (using VC dimension) that is related to (but not the same as) the proposition that this is a good thing.
5. Empirically it works very very well.

Specifying a line and margin



- How do we represent this mathematically?
- ...in m input dimensions?

Specifying a line and margin



- Plus-plane = $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = +1 \}$
- Minus-plane = $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = -1 \}$

Classify as.. $+1$ if $\mathbf{w} \cdot \mathbf{x} + b \geq 1$

-1 if $\mathbf{w} \cdot \mathbf{x} + b \leq -1$

Universe if $-1 < \mathbf{w} \cdot \mathbf{x} + b < 1$
explodes

Support vector machines

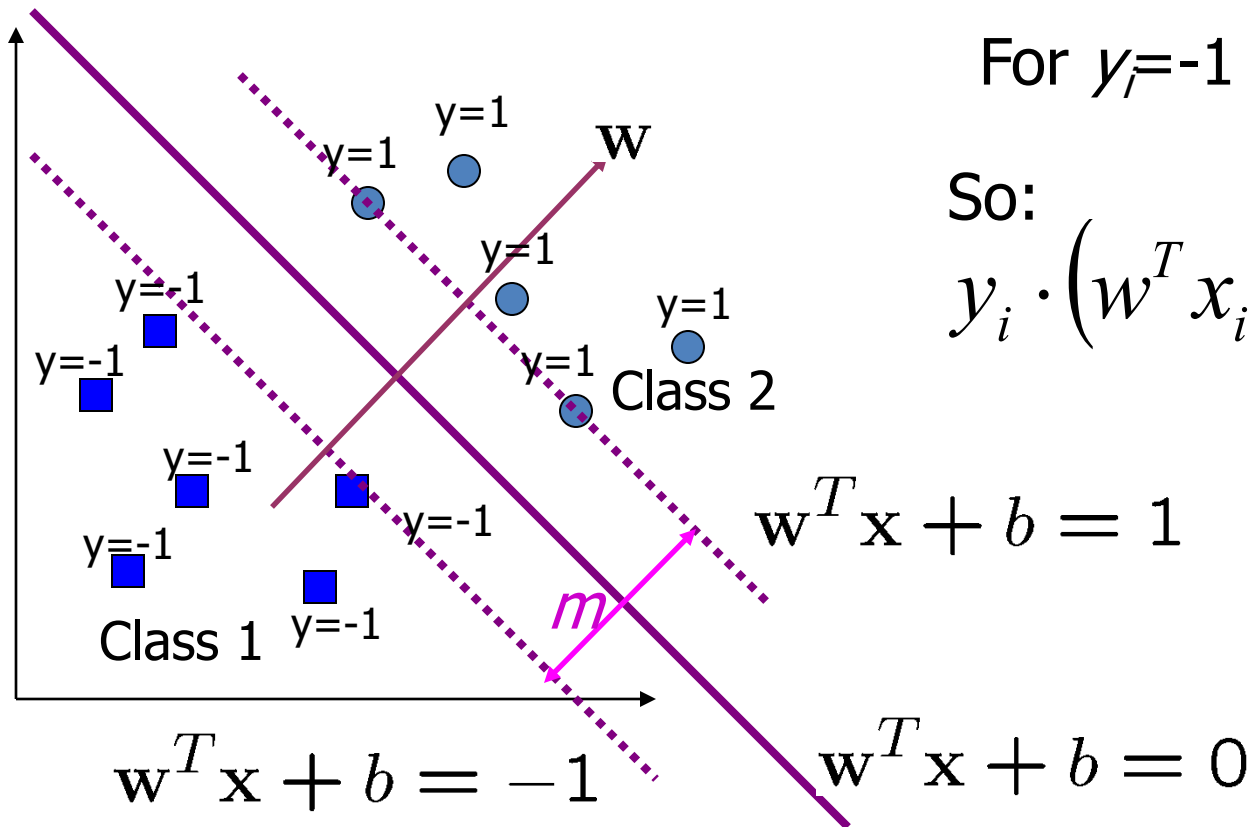
- Let $\{x_1, \dots, x_n\}$ be our data set and let $y_i \in \{1, -1\}$ be the class label of x_i

$$\text{For } y_i = 1 \quad w^T x_i + b \geq 1$$

$$\text{For } y_i = -1 \quad w^T x_i + b \leq -1$$

So:

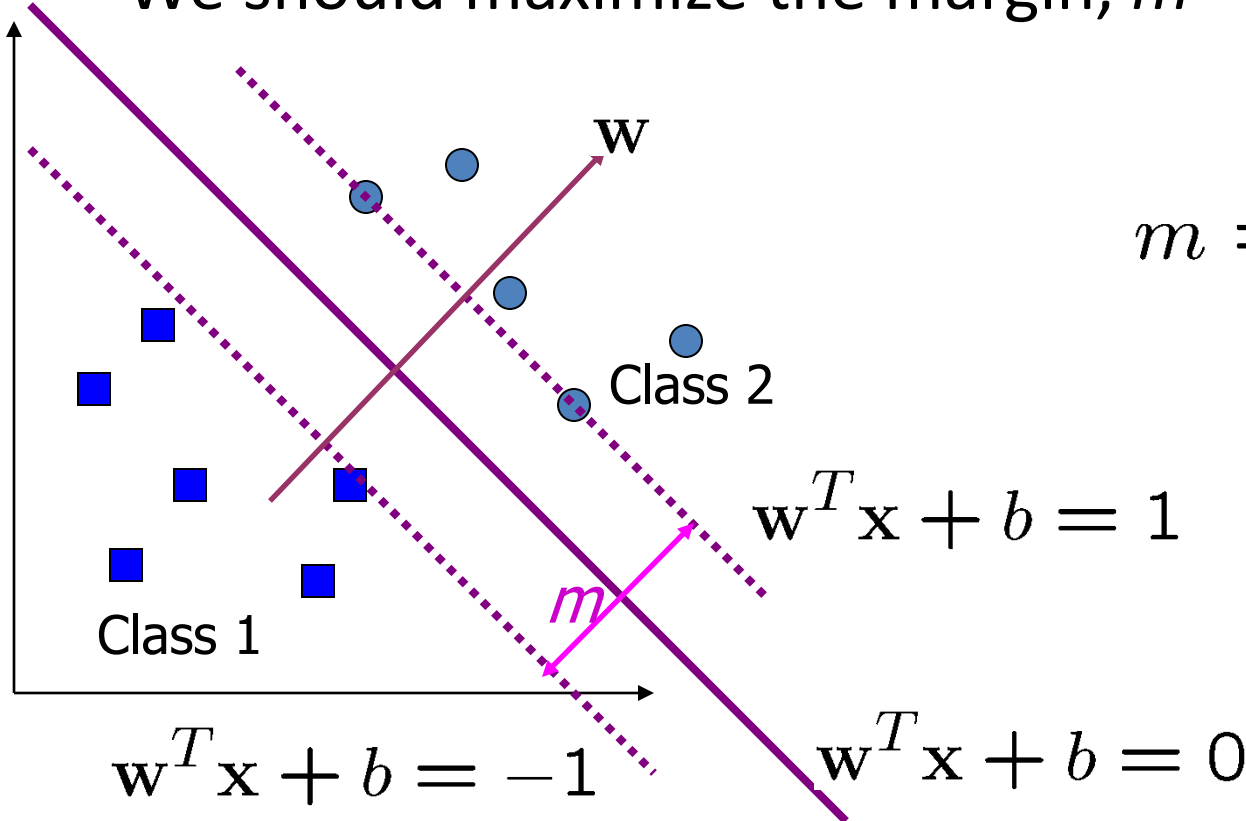
$$y_i \cdot (w^T x_i + b) \geq 1, \forall (x_i, y_i)$$



Large-margin Decision Boundary

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

– We should maximize the margin, m



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

Finding the Decision Boundary

- The decision boundary should classify all points correctly \Rightarrow

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

- The decision boundary can be found by solving the following constrained optimization problem

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

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

- This is a constrained optimization problem. Solving it requires to use Lagrange multipliers

KKT Conditions

- Problem:

$$\min_x f(x) \quad \text{sub. to: } g_i(x) \leq 0 \quad \forall i$$

- Lagrangian: $L(x, \mu) = f(x) + \sum_i \mu_i g_i(x)$

- Conditions:

- Stationarity: $\nabla_x L(x, \mu) = 0.$

- Primal feasibility: $g_i(x) \leq 0 \quad \forall i.$

- Dual feasibility: $\mu_i \geq 0.$

- Complementary slackness: $\mu_i g_i(x) = 0.$

Finding the Decision Boundary

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

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

- The Lagrangian is

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

– $\alpha_i \geq 0$

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

The Dual Problem

- Setting the gradient of L w.r.t. \mathbf{w} and b to zero, we have

$$\begin{aligned} L &= \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{i=1}^n \alpha_i (1 - y_i (\mathbf{w}^T \mathbf{x}_i + b)) = \\ &= \frac{1}{2} \sum_{k=1}^m w^k w^k + \sum_{i=1}^n \alpha_i \left(1 - y_i \left(\sum_{k=1}^m w^k x_i^k + b \right) \right) \end{aligned}$$

n : no of examples, m : dimension of the space

$$\left\{ \begin{array}{l} \frac{\partial L}{\partial w^k} = 0, \forall k \\ \frac{\partial L}{\partial b} = 0 \end{array} \right. \quad \mathbf{w} + \sum_{i=1}^n \alpha_i (-y_i) \mathbf{x}_i = \mathbf{0} \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$$
$$\sum_{i=1}^n \alpha_i y_i = 0$$

The Dual Problem

- If we substitute $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$ to \mathcal{L} , we have

$$\begin{aligned}\mathcal{L} &= \frac{1}{2} \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i^T \sum_{j=1}^n \alpha_j y_j \mathbf{x}_j + \sum_{i=1}^n \alpha_i \left(1 - y_i \left(\sum_{j=1}^n \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i + b \right) \right) \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^n \alpha_i - \sum_{i=1}^n \alpha_i y_i \sum_{j=1}^n \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i - b \sum_{i=1}^n \alpha_i y_i \\ &= -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^n \alpha_i\end{aligned}$$

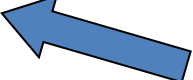

Since $\sum_{i=1}^n \alpha_i y_i = 0$

- This is a function of α_i only

The Dual Problem

- The new objective function is in terms of α_i only
- It is known as the dual problem: if we know \mathbf{w} , we know all α_i ; if we know all α_i , we know \mathbf{w}
- The original problem is known as the primal problem
- The objective function of the dual problem needs to be maximized (comes out from the KKT theory)
- The dual problem is therefore:

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

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


Properties of α_i when we introduce the Lagrange multipliers

The result when we differentiate the original Lagrangian w.r.t. \mathbf{b}

The Dual Problem

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

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

- This is a quadratic programming (QP) problem
 - A global maximum of α_i can always be found

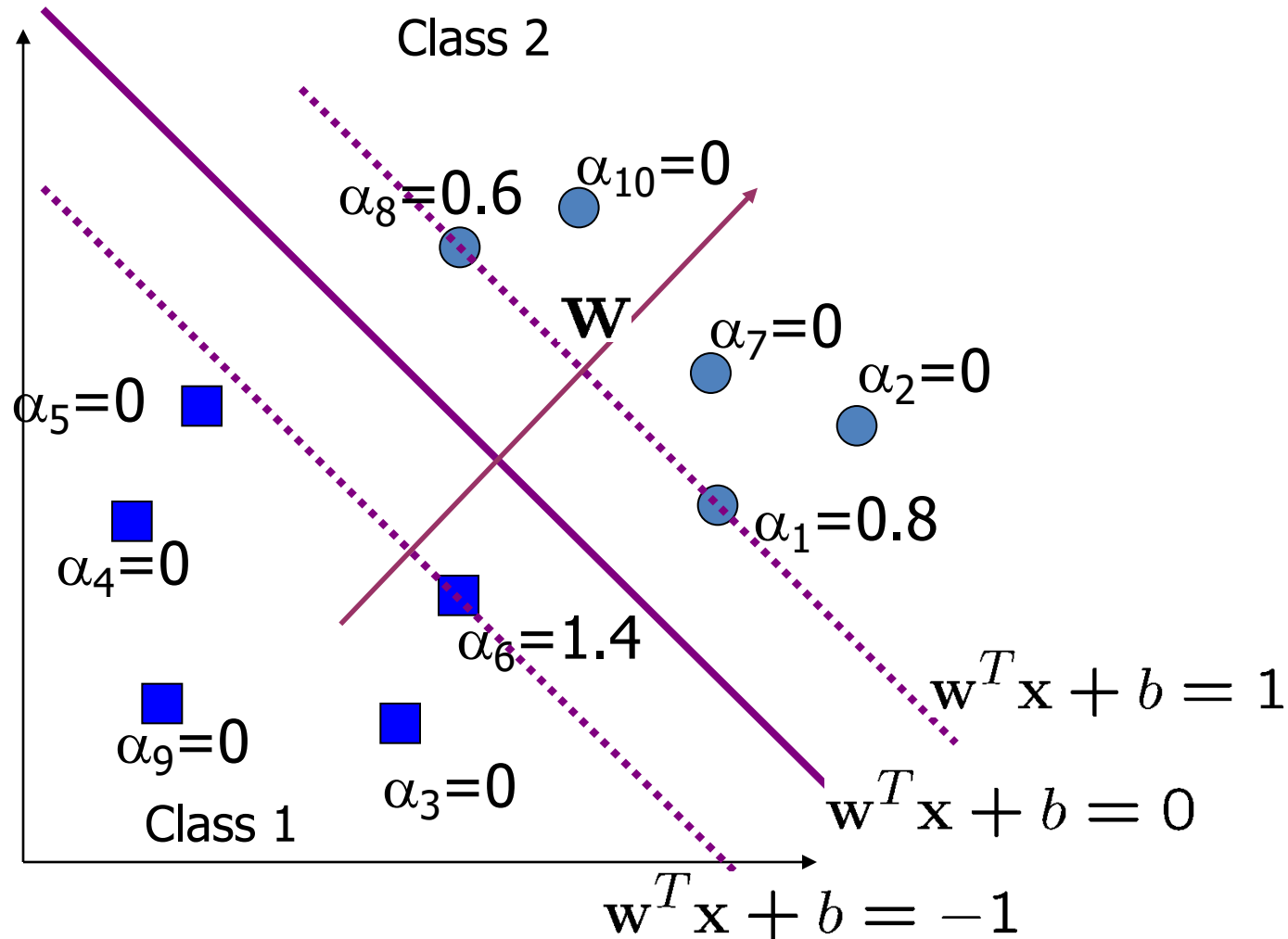
- \mathbf{w} can be recovered by
$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$$

Characteristics of the Solution

- Many of the α_i are zero
 - Complementary slackness: $\alpha_i(1 - y_i(w^T x_i + b)) = 0$
 - Sparse representation: \mathbf{w} is a linear combination of a small number of data points
- \mathbf{x}_i with non-zero α_i are called support vectors (SV)
 - The decision boundary is determined only by the SV
 - Let t_j ($j=1, \dots, s$) be the indices of the s support vectors.
We can write

$$\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

A Geometrical Interpretation

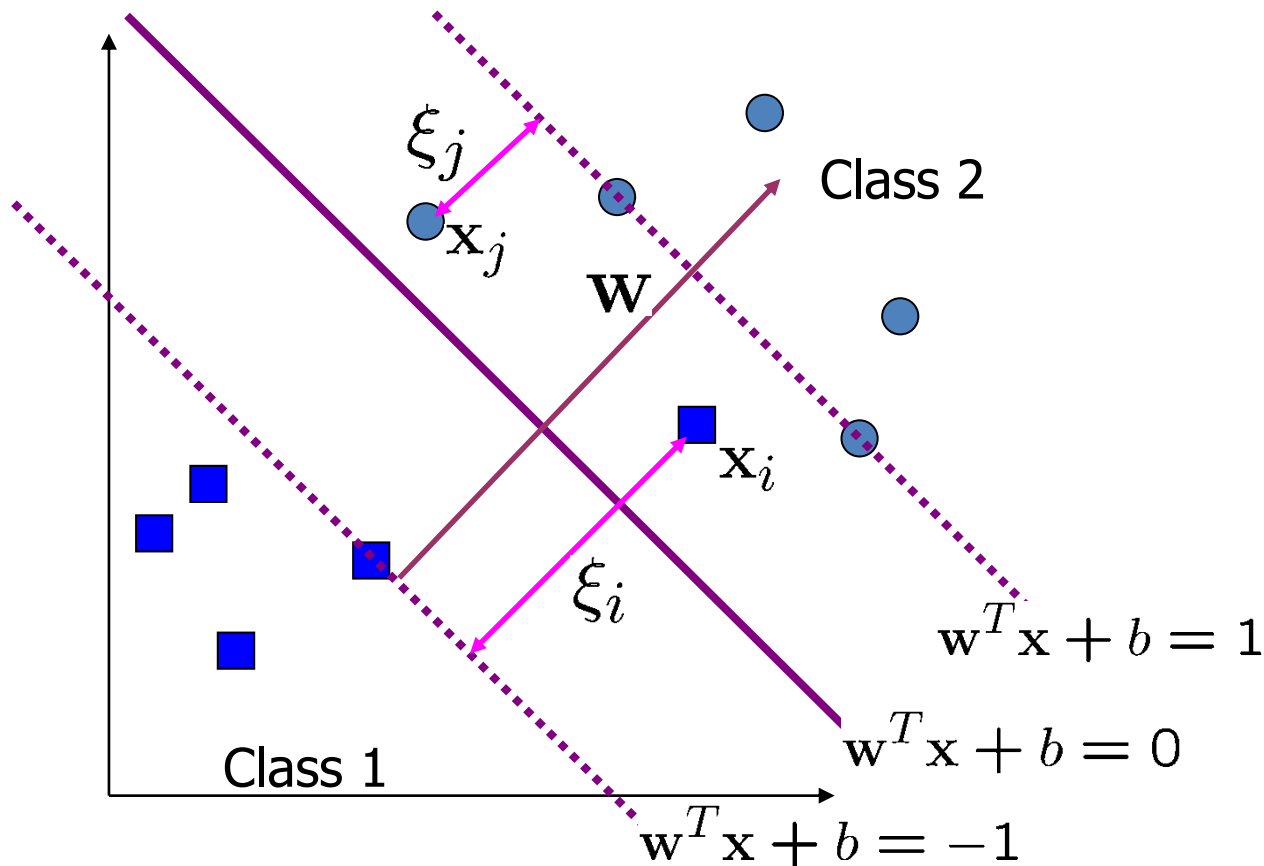


Characteristics of the Solution

- For testing with a new data \mathbf{z}
 - Compute $\mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^s \alpha_{t_j} y_{t_j} (\mathbf{x}_{t_j}^T \mathbf{z}) + b$ and classify \mathbf{z} as class 1 if the sum is positive, and class 2 otherwise
 - Note: \mathbf{w} need not be formed explicitly

Non-linearly Separable Problems

- We allow “error” ξ_i in classification; it is based on the output of the discriminant function $\mathbf{w}^T \mathbf{x} + b$
- ξ_i approximates the number of misclassified samples



Soft Margin Hyperplane

- The new conditions become

$$\begin{cases} \mathbf{w}^T \mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T \mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

- ξ_i are “slack variables” in optimization
 - Note that $\xi_i=0$ if there is no error for \mathbf{x}_i
 - ξ_i is an upper bound of the number of errors
- We want to minimize

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

subject to $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$

- C : tradeoff parameter between error and margin

The Optimization Problem

$$L = \frac{1}{2} w^T w + C \sum_{i=1}^n \xi_i + \sum_{i=1}^n \alpha_i (1 - \xi_i - y_i (w^T x_i + b)) - \sum_{i=1}^n \mu_i \xi_i$$

With α and μ Lagrange multipliers, POSITIVE

$$\frac{\partial L}{\partial w_j} = w_j - \sum_{i=1}^n \alpha_i y_i x_{ij} = 0$$

$$\vec{w} = \sum_{i=1}^n \alpha_i y_i \vec{x}_i = 0$$

$$\frac{\partial L}{\partial \xi_j} = C - \alpha_j - \mu_j = 0$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^n y_i \alpha_i = 0$$

The Dual Problem

$$L = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \vec{x}_i^T \vec{x}_j + C \sum_{i=1}^n \xi_i +$$
$$+ \sum_{i=1}^n \alpha_i \left(1 - \xi_i - y_i \left(\sum_{j=1}^n \alpha_j y_j x_j^T x_i + b \right) \right) - \sum_{i=1}^n \mu_i \xi_i$$

With $\sum_{i=1}^n y_i \alpha_i = 0$ and $C = \alpha_j + \mu_j$

$$L = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \vec{x}_i^T \vec{x}_j + \sum_{i=1}^n \alpha_i$$

The Optimization Problem

- The dual of this new constrained optimization problem is

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

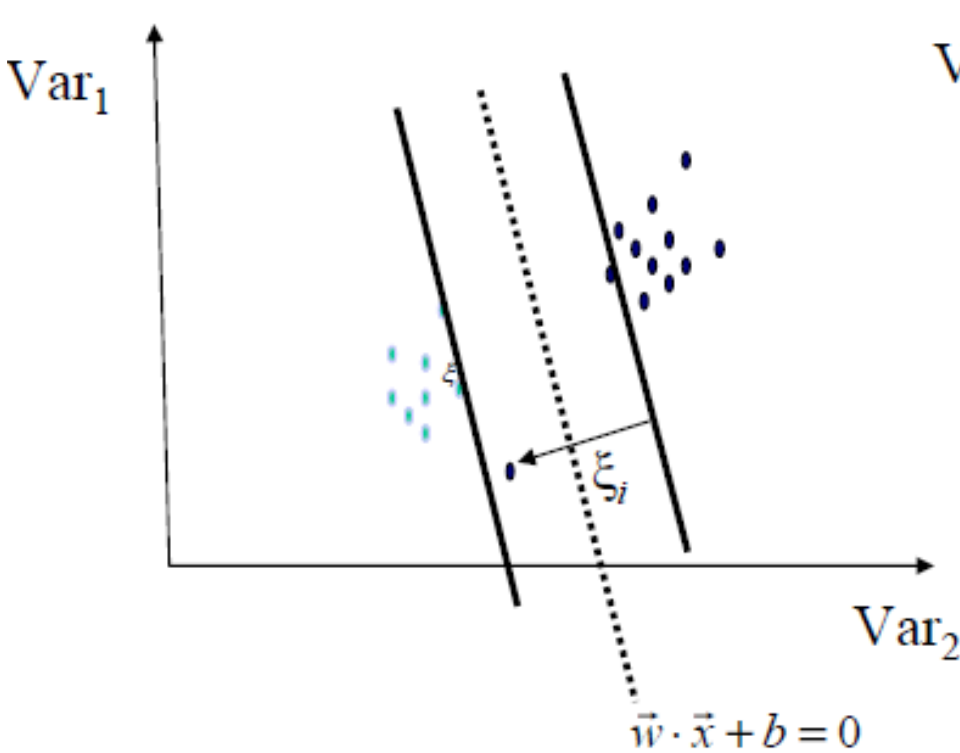
$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$$

- New constraints derived from $C = \alpha_j + \mu_j$ since μ and α are positive.
- \mathbf{w} is recovered as $\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$
- This is very similar to the optimization problem in the linear separable case, except that there is an upper bound C on α_i now
- Once again, a QP solver can be used to find α_i

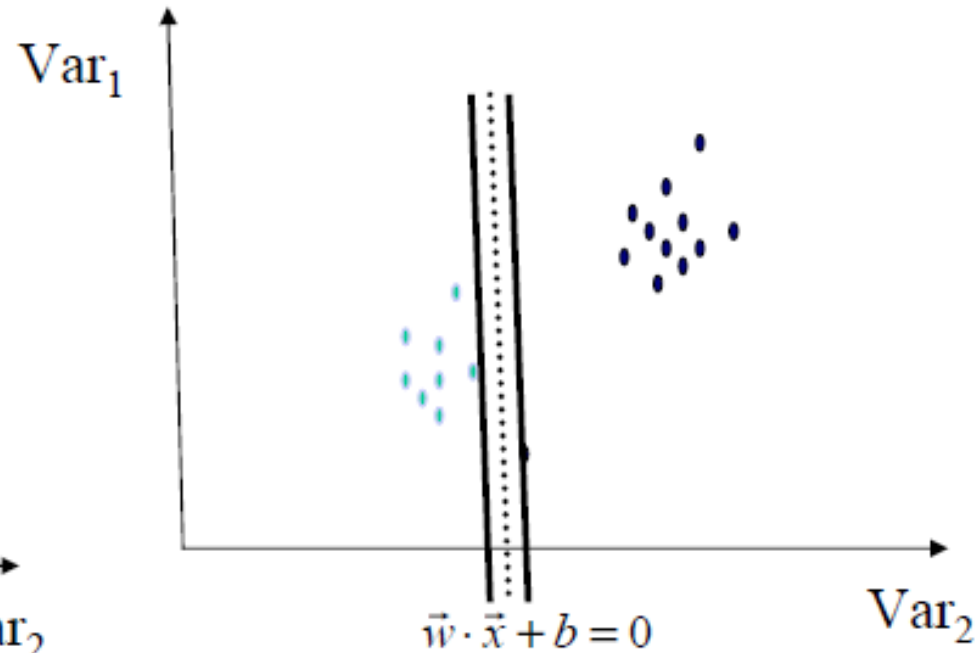
$$\frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i$$

- The algorithm try to keep ξ low, maximizing the margin
- The algorithm does not minimize the number of error. Instead, it minimizes the sum of distances from the hyperplane.
- When C increases the number of errors tend to lower. At the limit of C tending to infinite, the solution tend to that given by the hard margin formulation, with 0 errors

Soft margin is more robust to outliers



Soft Margin SVM



Hard Margin SVM

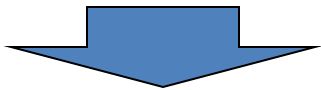
Extension to Non-linear Decision Boundary

- So far, we have only considered large-margin classifier with a linear decision boundary
- How to generalize it to become nonlinear?
- Key idea: transform \mathbf{x}_i to a higher dimensional space to “make life easier”
 - Input space: the space the point \mathbf{x}_i are located
 - Feature space: the space of $\phi(\mathbf{x}_i)$ after transformation
- Why transform?
 - Linear operation in the feature space is equivalent to non-linear operation in input space
 - Classification can become easier with a proper transformation. In the XOR problem, for example, adding a new feature of x_1x_2 make the problem linearly separable

Extension to Non-linear Decision Boundary

- So far, we have only considered large-margin classifier with a linear decision boundary
- How to generalize it to become nonlinear?
- Key idea: transform \mathbf{x}_i to a higher dimensional space to “make life easier”
 - Input space: the space the point \mathbf{x}_i are located
 - Feature space: the space of $\phi(\mathbf{x}_i)$ after transformation
- Why transform?
 - Linear operation in the feature space is equivalent to non-linear operation in input space
 - Classification can become easier with a proper transformation. In the XOR problem, for example, adding a new feature of x_1x_2 make the problem linearly separable

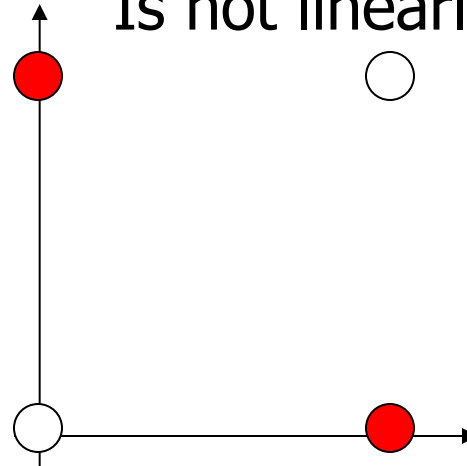
X	Y	
0	0	0
0	1	1
1	0	1
1	1	0



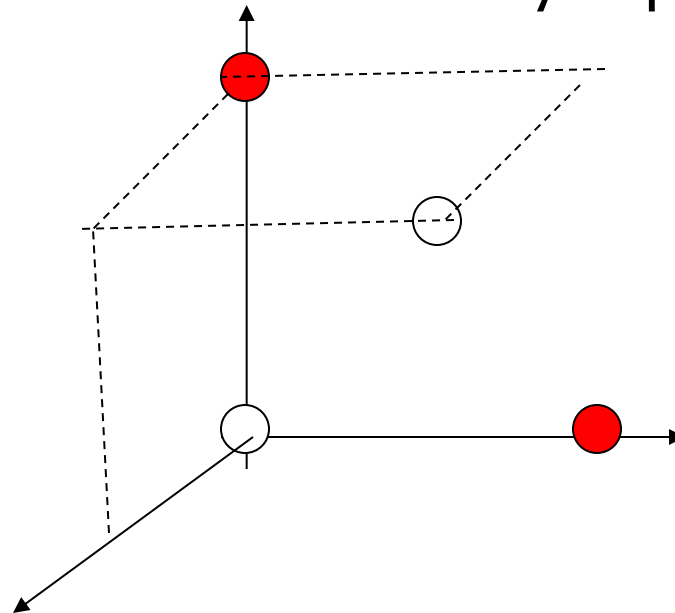
X	Y	XY	
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

XOR

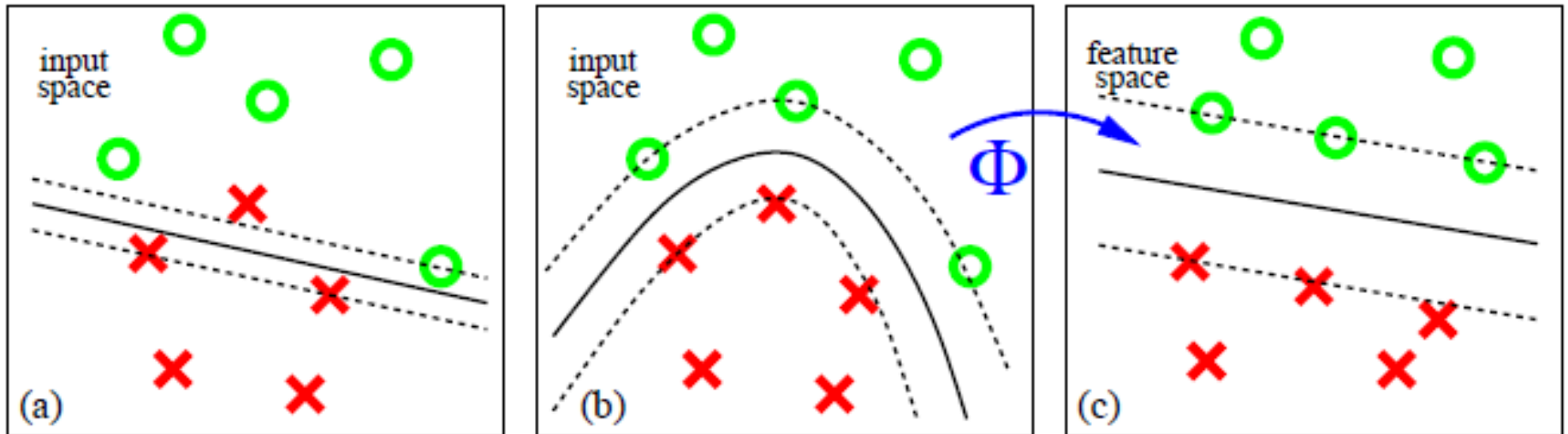
Is not linearly separable



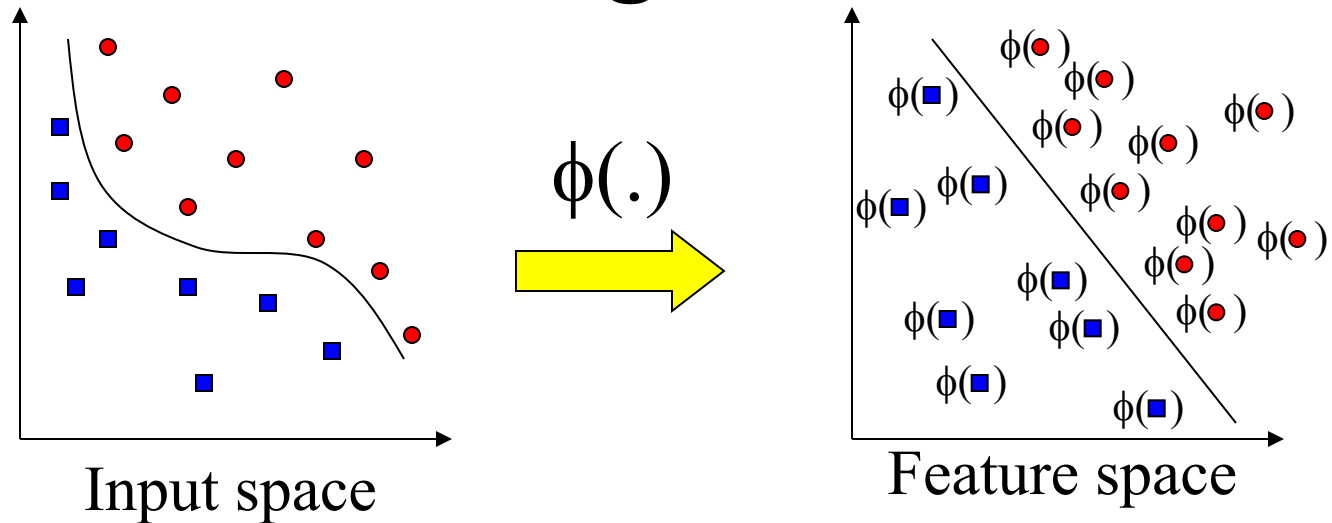
Is linearly separable



Find a feature space



Transforming the Data



Note: feature space is of higher dimension than the input space in practice

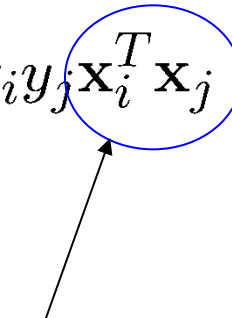
- Computation in the feature space can be costly because it is high dimensional
 - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

The Kernel Trick

- Recall the SVM optimization problem

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

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



- The data points only appear as **inner product**
- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations (angles, distances) can be expressed by inner products
- Define the kernel function K by

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

An Example for $\phi(\cdot)$ and $K(\cdot, \cdot)$

- Suppose $\phi(\cdot)$ is given as follows

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

- An inner product in the feature space is

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

- So, if we define the kernel function as follows, there is no need to carry out $\phi(\cdot)$ explicitly

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

- This use of kernel function to avoid carrying out $\phi(\cdot)$ explicitly is known as the **kernel trick**

Kernels

- Given a mapping: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$
a kernel is represented as the inner product

$$K(\mathbf{x}, \mathbf{y}) \rightarrow \sum_i \varphi_i(\mathbf{x})\varphi_i(\mathbf{y})$$

A kernel must satisfy the Mercer's condition:

$$\forall g(\mathbf{x}) \int K(\mathbf{x}, \mathbf{y})g(\mathbf{x})g(\mathbf{y})d\mathbf{x}d\mathbf{y} \geq 0$$

Modification Due to Kernel Function

- Change all inner products to kernel functions
- For training,

Original

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

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

With kernel function

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

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

Modification Due to Kernel Function

- For testing, the new data \mathbf{z} is classified as class 1 if $f \geq 0$, and as class 2 if $f < 0$

Original

$$\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$
$$f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^T \mathbf{z} + b$$

With kernel
function

$$\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$
$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^s \alpha_{t_j} y_{t_j} K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$

More on Kernel Functions

- Since the training of SVM only requires the value of $K(\mathbf{x}_i, \mathbf{x}_j)$, there is no restriction of the form of \mathbf{x}_i and \mathbf{x}_j
 - \mathbf{x}_i can be a sequence or a tree, instead of a feature vector
- $K(\mathbf{x}_i, \mathbf{x}_j)$ is just a similarity measure comparing \mathbf{x}_i and \mathbf{x}_j
- For a test object \mathbf{z} , the discriminant function essentially is a weighted sum of the similarity between \mathbf{z} and a pre-selected set of objects (the support vectors)

$$f(\mathbf{z}) = \sum_{\mathbf{x}_i \in \mathcal{S}} \alpha_i y_i K(\mathbf{z}, \mathbf{x}_i) + b$$

\mathcal{S} : the set of support vectors

Kernel Functions

- In practical use of SVM, the user specifies the kernel function; the transformation $\phi(\cdot)$ is not explicitly stated
- Given a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$, the transformation $\phi(\cdot)$ is given by its eigenfunctions (a concept in functional analysis)
 - Eigenfunctions can be difficult to construct explicitly
 - This is why people only specify the kernel function without worrying about the exact transformation
- Another view: kernel function, being an inner product, is really a similarity measure between the objects

A kernel is associated to a transformation

- Given a kernel, in principle it should be recovered the transformation in the feature space that originates it.
- $K(x,y) = (xy+1)^2 = x^2y^2+2xy+1$

It corresponds the transformation $x \rightarrow \begin{pmatrix} x^2 \\ \sqrt{2}x \\ 1 \end{pmatrix}$

Examples of Kernel Functions

- Polynomial kernel of degree d

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d$$

- Polynomial kernel up to degree d

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

- Radial basis function kernel with width σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2 / (2\sigma^2))$$

- The feature space is infinite-dimensional

- Sigmoid with parameter κ and θ

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \mathbf{x}^T \mathbf{y} + \theta)$$

- It does not satisfy the Mercer condition on all κ and θ

Building new kernels

- If $k_1(x,y)$ and $k_2(x,y)$ are two valid kernels then the following kernels are valid

- *Linear Combination*

$$k(x, y) = c_1 k_1(x, y) + c_2 k_2(x, y)$$

- *Exponential*

$$k(x, y) = \exp[k_1(x, y)]$$

- *Product*

$$k(x, y) = k_1(x, y) \cdot k_2(x, y)$$

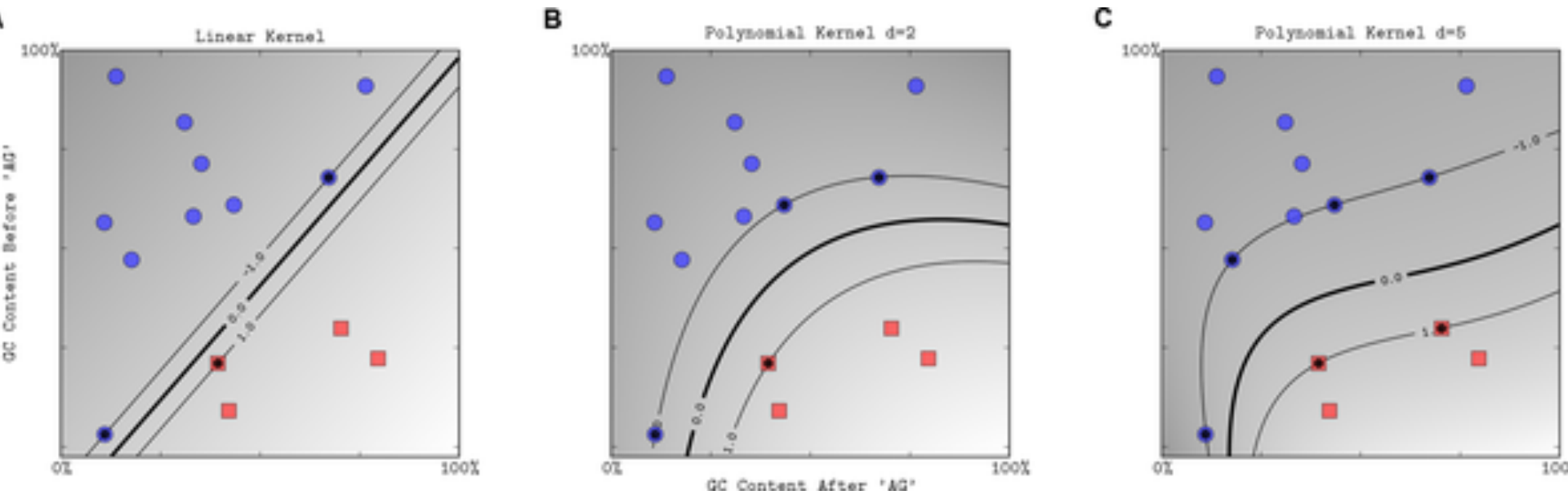
- *Polynomial transformation (Q: polynomial with non negative coefficients)*

$$k(x, y) = Q[k_1(x, y)]$$

- *Function product (f: any function)*

$$k(x, y) = f(x)k_1(x, y)f(y)$$

Polynomial kernel



Gaussian RBF kernel

