ECE/CS 559 - Lecture Notes Part #9
Support Vector Machines

E. Koyuncu

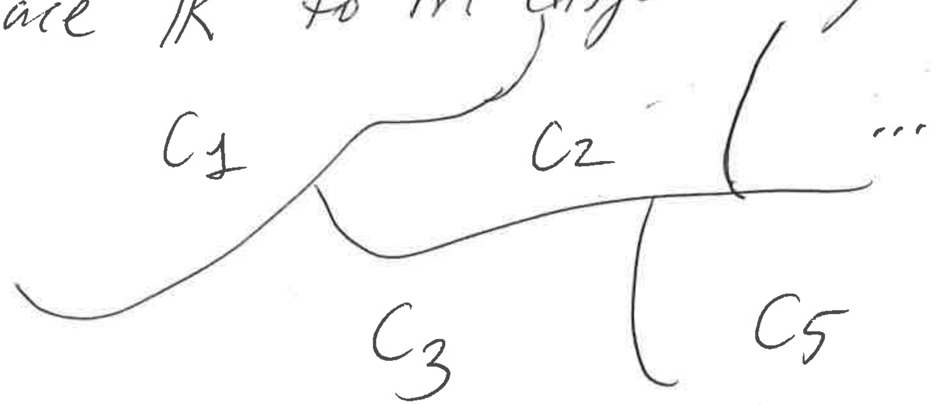
Classification: Suppose we have m classes $C_1, C_2, \dots, C_m \subset \mathbb{R}^{n \times 1}$. The general goal of classification is to find/build a machine/algorithm that provides the correct class information for a given input $x \in \mathbb{R}^{n \times 1}$. As a diagram, we have:

$$x \longrightarrow \boxed{\text{Classifier}} \longrightarrow i(x) \in \{1, \dots, m\}$$

(Class information)

Here, $i(x)$ is called the membership function.

The classes imply a partition of the input space $\mathbb{R}^n$ to m disjoint regions:



Often, the classifier can be constructed by first finding the decision boundaries.

Discriminant Functions

- Suppose that we can find m functions

$$g_1, \dots, g_m : \mathbb{R}^n \longrightarrow \mathbb{R}$$

such that for every $k \in \{1, \dots, m\}$, $x \in \mathbb{R}^n$

$$\left(g_k(x) > g_j(x) \text{ for every } j \in \{1, \dots, m\}, j \neq k \right)$$

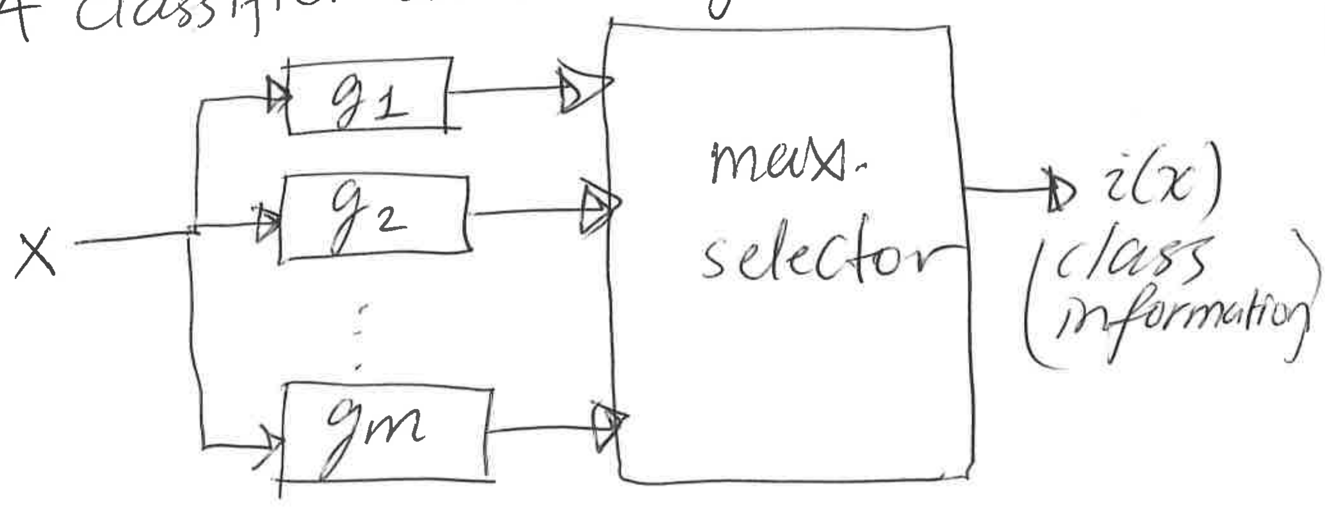
$$\Longleftrightarrow$$

$$x \in C_k$$

In other words, in the region C_k, the relevant function g_k is the maximum among all other functions $g_1, \dots, g_{k-1}, g_{k+1}, \dots, g_m$.

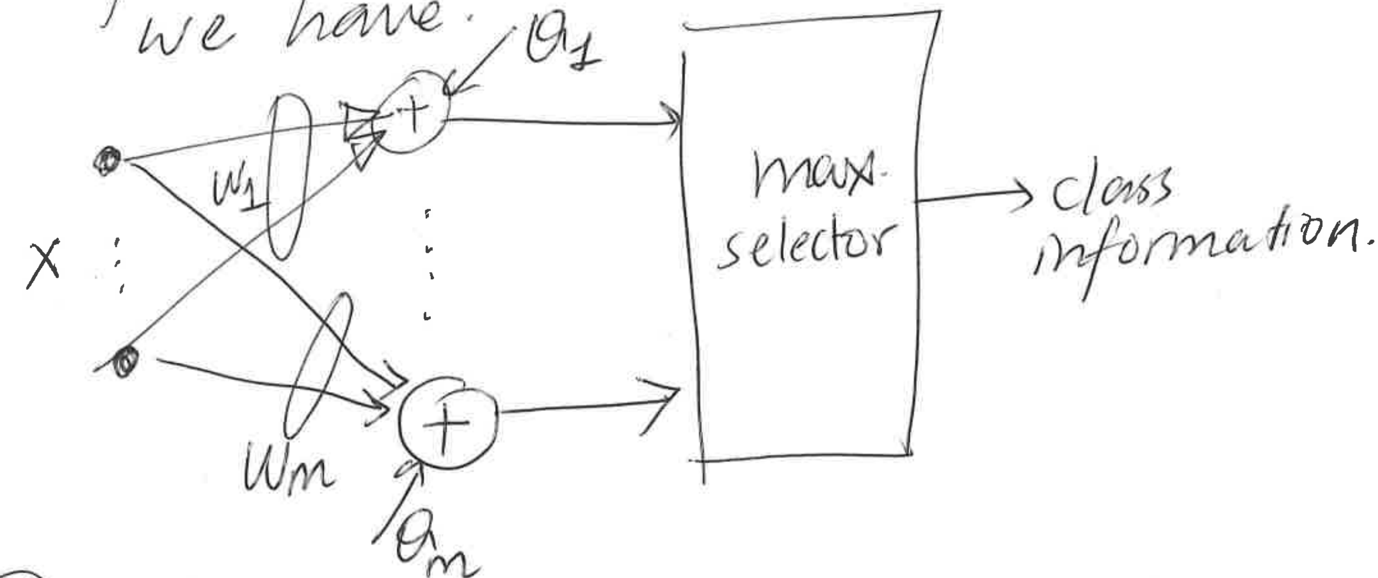
- The functions $g_1, \dots, g_m$ are called discriminant functions.
- If we can find such discriminant functions, there are several benefits:

(1) A classifier can easily be built:

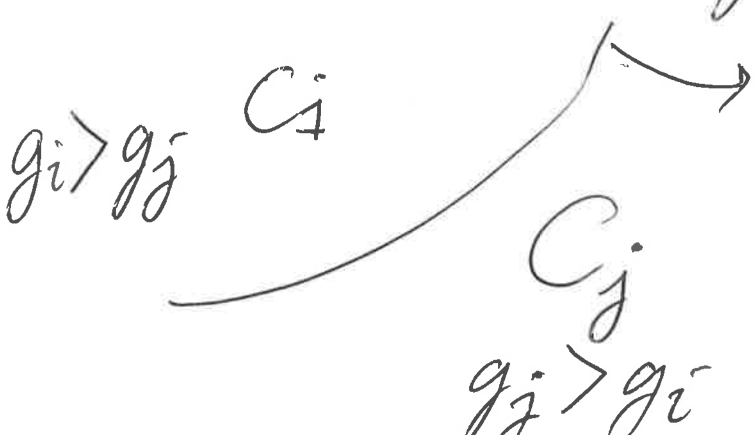


2. If $g_1, \dots, g_m$ are linear, we can even use a standard neural network architecture:

i.e., if $g_i(x) = w_i^T x + \theta_i$ for some weights w_i and bias θ_i, we have:



3. If necessary, the decision boundaries can also be easily computed:



Decision Boundary: $\{x : g_i(x) = g_j(x)\}$.

For the special case of 2 classes,

$$x \in C_1 \Leftrightarrow g_1(x) > g_2(x) \Leftrightarrow g_1(x) - g_2(x) > 0$$

$$x \in C_2 \Leftrightarrow g_2(x) > g_1(x) \Leftrightarrow g_1(x) - g_2(x) < 0$$

Hence, if we define

$g(x) = g_1(x) - g_2(x)$, we have

$$x \in C_1 \Leftrightarrow g(x) > 0$$

$$x \in C_2 \Leftrightarrow g(x) < 0,$$

and the decision boundary is given by:

$$g(x) = 0.$$

The entire classifier can be constructed as:

$$x \longrightarrow \boxed{g} \longrightarrow \boxed{\text{sgn}} \longrightarrow \text{output} = \begin{cases} 1, & x \in C_1 \\ 0, & x \in \text{Boundary} \\ -1, & x \in C_2 \end{cases}$$

This classifier is also called a $\boxed{\text{dichotomizer.}}$

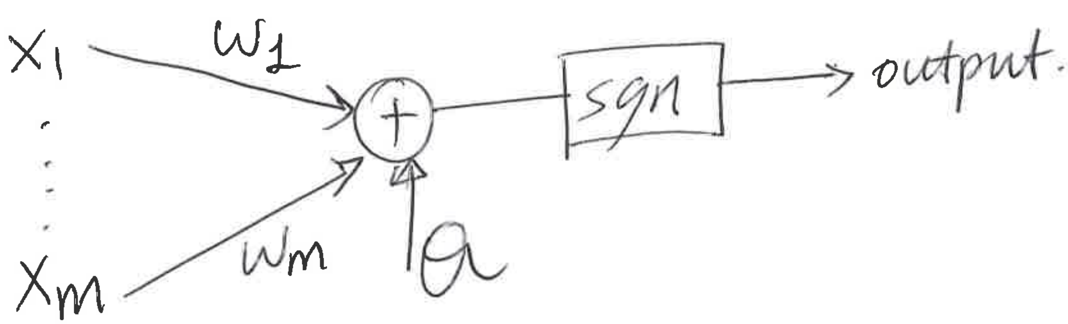
Linear Dicrimmant Functions.

These discrimmant functions are of the form:

$$g(x) = w^T x + \theta$$
$$= w_1 x_1 + \cdots + w_n x_n + \theta,$$

where w_i is the ith component of w, and
x_i " " " " " x.

This is, of course, related to the perceptron:



Properties

① The boundary $H = \{x : g(x) = 0\}$ is a hyperplane.

② The weight vector w is perpendicular to the hyperplane H, i.e

$$w \perp H.$$

To see this, consider two distinct points $x_1, x_2 \in H$.

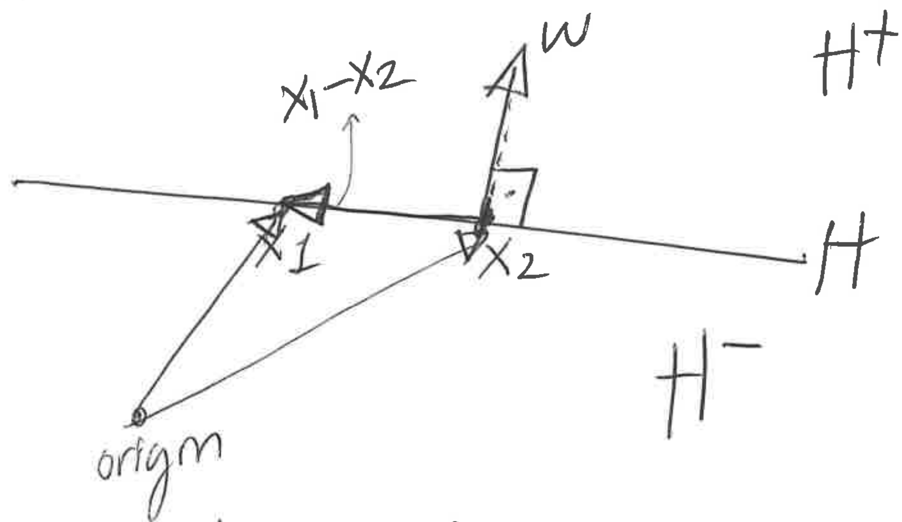
We have $x_1 \in H \Rightarrow g(x_1) = w^T x_1 + \theta = 0$

$x_2 \in H \Rightarrow w^T x_2 + \theta = 0$

Subtracting the two equalities, we obtain

$$w^T (x_1 - x_2) = 0.$$

But, the vector $x_1 - x_2$ is parallel to H, and thus w should be perpendicular to H.



On the diagram,

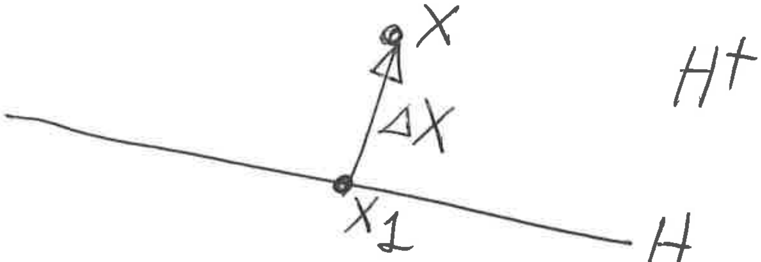
$$H^+ = \{ x : g(x) > 0 \}$$

is the "positive side" of H, and

$$H^- = \{ x : g(x) < 0 \}$$

is the "negative side" of H.

3. As shown in the diagram in the previous page, w points to the positive side H^+ of H. To see this, consider a vector x_1 on H, and a vector x on H^+.



We have

$$g(x_1) = 0 \Rightarrow w^T x_1 + \theta = 0$$
$$g(x) > 0 \Rightarrow w^T x + \theta > 0$$

Subtract the former from the latter to obtain

$$w^T x + \theta - (w^T x_1 + \theta) > 0$$

or equivalenty

$$w^T \Delta x > 0.$$

Since $w^T \Delta x = \|w\| \|\Delta x\| \cos(\alpha)$ where α is the angle between w and Δx, $w^T \Delta x > 0$ implies $\cos(\alpha) > 0$ or $\alpha < 90°$. Since w is perpendicular to H, and Δx points to the + side of H, the only way this could happen is if w also points to the positive side of H.

4. If we know a point, say x_1, on H, we can easily find θ. Indeed, given $g(x_1) = w^T x_1 + \theta = 0$, we simply have $\theta = -w^T x_1$.

Minimum Distance Classifiers

Suppose that the two classes we would like to seperate consist of one member each, ie

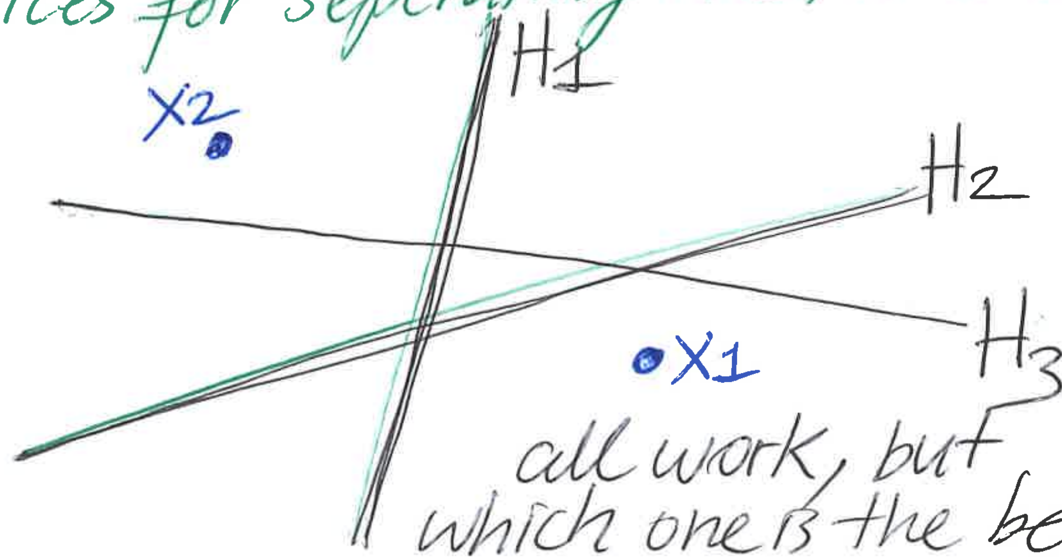
$$C_1 = \{x_1\}, \text{ and } C_2 = \{x_2\}.$$

Question: What is the best hyperplane that seperates the two classes?

Equivalently,

What is the best linear classifier?

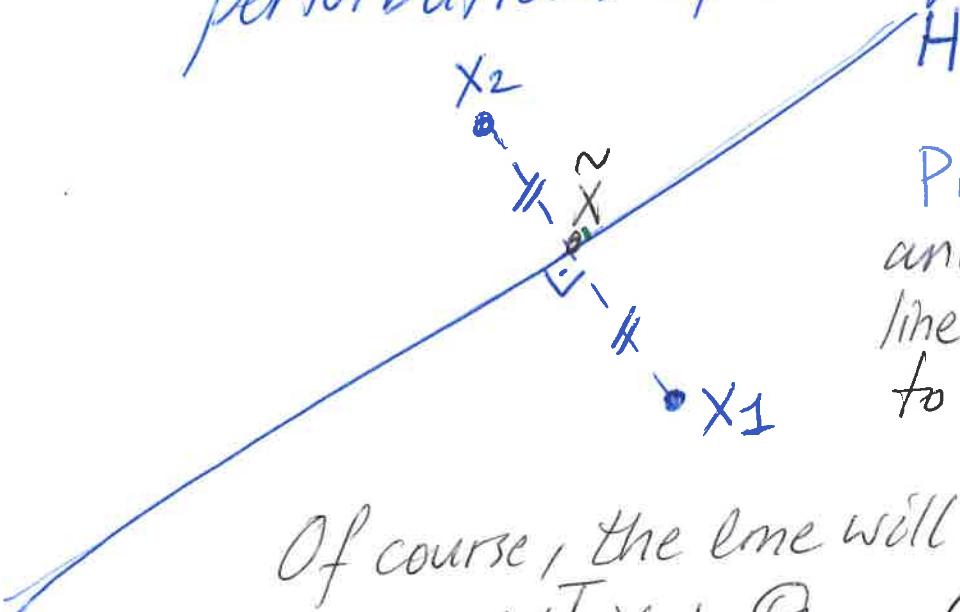
(Note that we have infinitely many choices for seperating the two classes:)



all work, but which one is the best?

Although different lines could be the "best" according to different criteria, intuitively, one should choose the line that is equidistant to x_1 and x_2

$\Rightarrow$ This way, one can tolerate larger perturbations of the input.



Problem: Given x_1 and x_2, find the line that is equidistant to x_1 and x_2.

Of course, the line will be of the form

$$w^T x + \theta = 0.$$

We already saw that w is perpendicular to H, so that we could as well choose $w = x_1 - x_2$, which is necessarily a vector that is perpendicular to H! To find θ, we note that the midpoint $\tilde{x} = \dfrac{x_1 + x_2}{2}$ should be on the line, so that

$$(x_1 - x_2)^T \frac{x_1 + x_2}{2} + \theta = 0 \Rightarrow \theta = \frac{-1}{2}(x_1 - x_2)^T (x_1 + x_2)$$

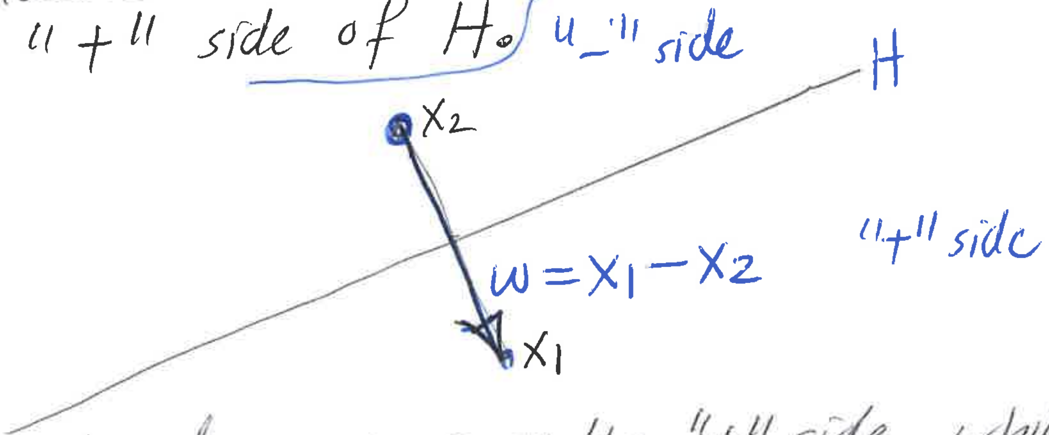
$$= \frac{-1}{2}\left(x_1^T x_1 + x_1^T x_2 - x_2^T x_1 - \|x_2\|^2\right)$$

$$= \frac{-1}{2}\left(\|x_1\|^2 - \|x_2\|^2\right)$$

So, the discriminant function becomes

$$g(x) = (x_1 - x_2)^T x - \frac{1}{2}\left(\|x_1\|^2 - \|x_2\|^2 \right).$$

Recall that $w = x_1 - x_2$ necessarily points to the "+" side of H.



Therefore, x_1 is on the "+" side, while x_2 is on the "−" side.

In fact, $g(x_1) = \frac{1}{2}\|x_1 - x_2\|^2 \geqslant 0$ and $g(x_2) = -\frac{1}{2}\|x_1 - x_2\|^2 \leqslant 0$. (after some straightforward calculations)

You may also want to manipulate the condition $g(x) \geqslant 0$ a little bit to get something more familiar:

$$
\begin{aligned}
& g(x) \geqslant 0 \iff \\
& (x_1 - x_2)^T x - \tfrac{1}{2}\left(\|x_1\|^2 - \|x_2\|^2 \right) \geqslant 0 \iff \\
& x_1^T x - \tfrac{1}{2}\|x_1\|^2 \geqslant x_2^T x - \tfrac{1}{2}\|x_2\|^2 \iff \\
& \|x_1\|^2 - 2x_1^T x \leqslant \|x_2\|^2 - 2x_2^T x \iff \\
& \|x_1\|^2 - 2x_1^T x + \|x\|^2 \leqslant \|x_2\|^2 - 2x_2^T x + \|x\|^2 \iff \\
& \|x_1 - x\|^2 \leqslant \|x_2 - x\|^2
\end{aligned}
$$

multiply both sides by −2

$$\|x - x_1\| \leq \|x - x_2\| \quad \square.$$

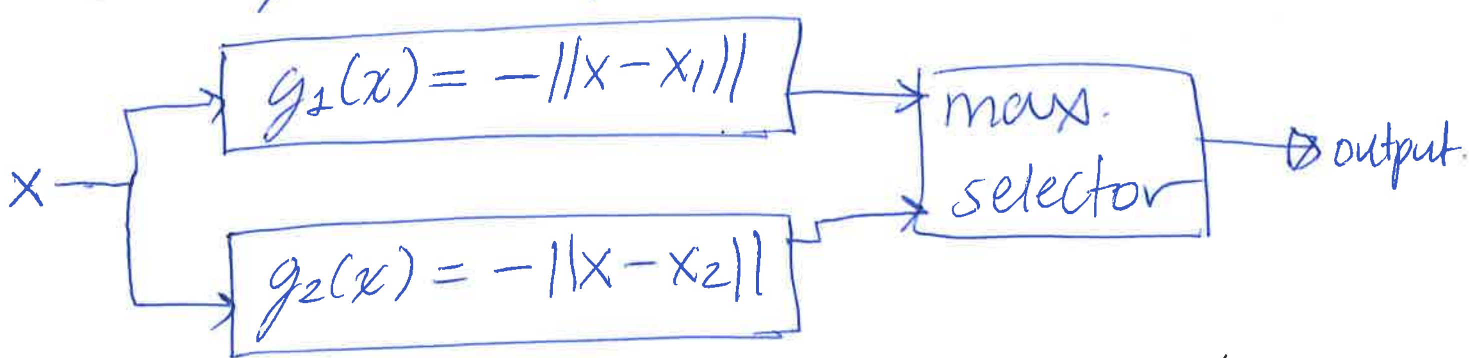
Intuitively, this should be clear. If we are closer to x_1 than x_2, we are on the "+" side. Otherwise, we are on the "−" side.

Going one more step, the inequality on top of this page is equivalent to the condition:

$$\|x - x_2\| - \|x - x_1\| \geq 0.$$

So, we could as well choose the discriminant function $g(x) = \|x - x_2\| - \|x - x_1\|$. It results in the same decision regions.

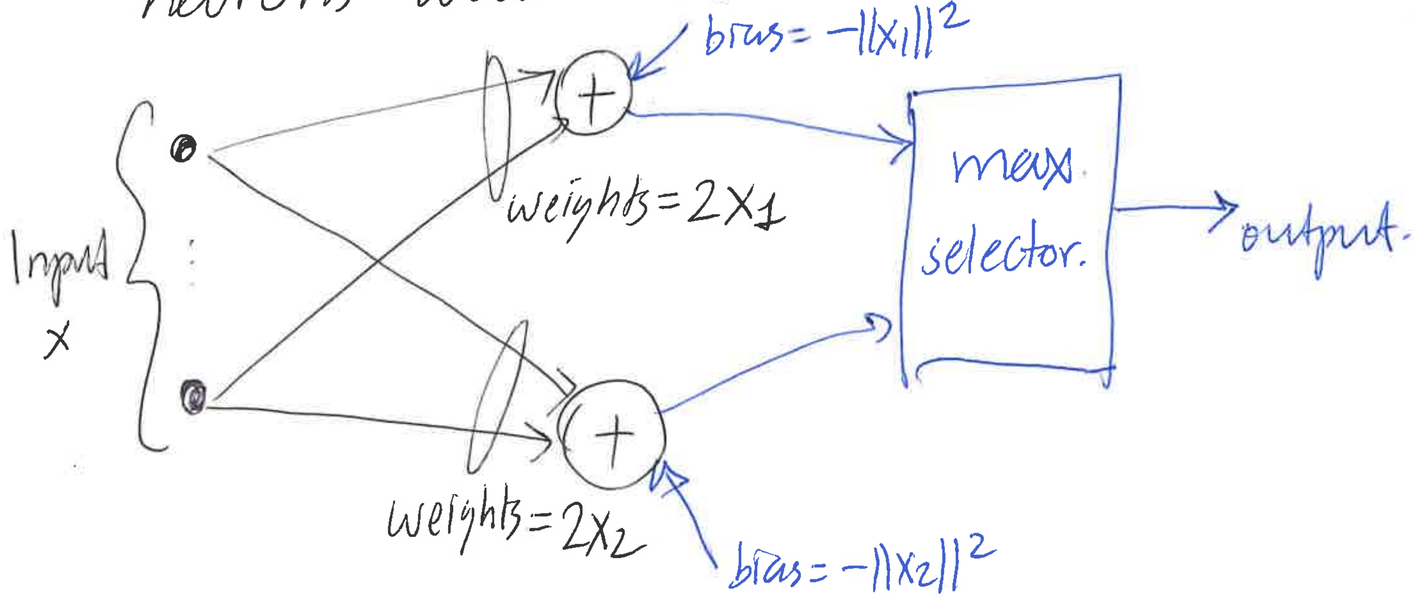
An equivalent implementation is:



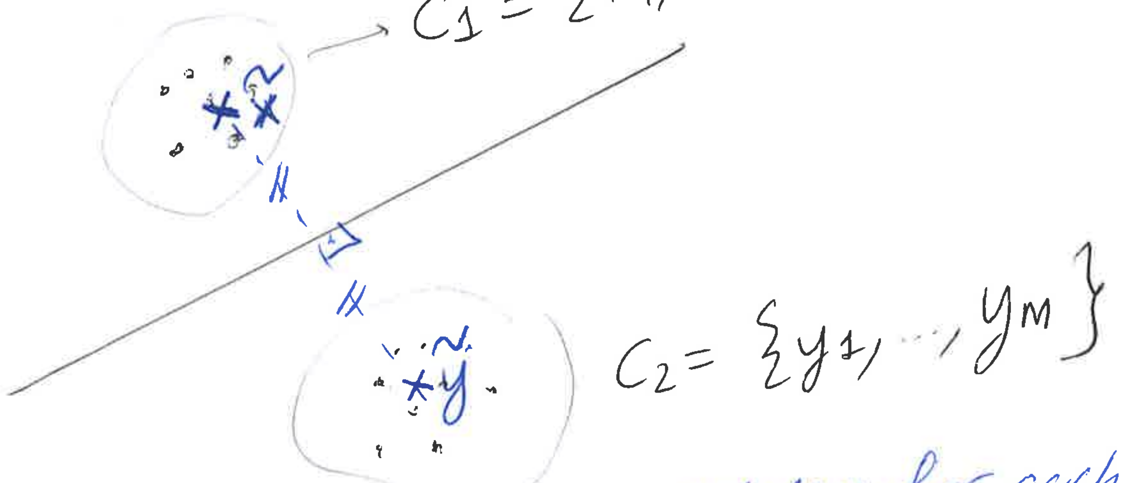
we have output $= \begin{cases} 1, & \text{if } x \text{ is closer to } x_1. \\ 2, & \text{" " " " " } x_2. \end{cases}$

A tie occurs when x is on the line that is equidistant to x_1 & x_2.

Another implementation using "actual" neurons would be:



What if we have more than one member in each class?



We can choose a class representative for each class;

$$\tilde{x} = \frac{1}{N}(x_1 + \dots + x_N), \text{ and}$$

$$\tilde{y} = \frac{1}{M}(y_1 + \dots + y_M), \text{ and design}$$

a minimum distance classifier based on $\tilde{x}$ and $\tilde{y}$.

Multicategory case

Suppose we have classes

$$C_1 = \{ x_{11}, \dots, x_{1,N_1} \}$$

$$C_2 = \{ x_{21}, \dots, x_{2,N_2} \}$$

$$\vdots$$

$$C_K = \{ x_{K1}, \dots, x_{K,N_K} \}$$

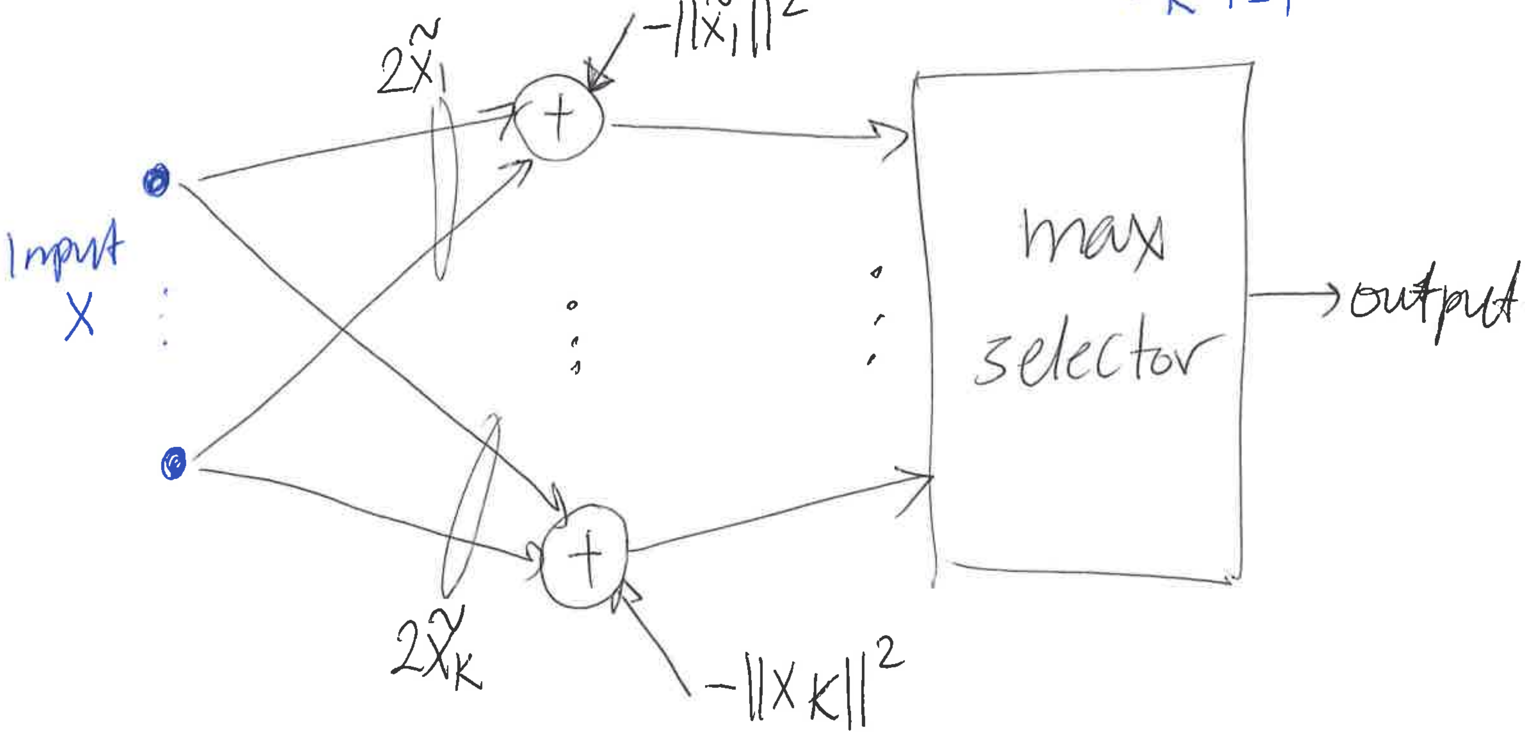
Class representatives

$$\tilde{x}_1 = \frac{1}{N_1} \sum_{i=1}^{N_1} x_{1,i}$$

$$\tilde{x}_2 = \frac{1}{N_2} \sum_{i=1}^{N_2} x_{2,i}$$

$$\vdots$$

$$\tilde{x}_K = \frac{1}{N_K} \sum_{i=1}^{N_K} x_{K,i}$$



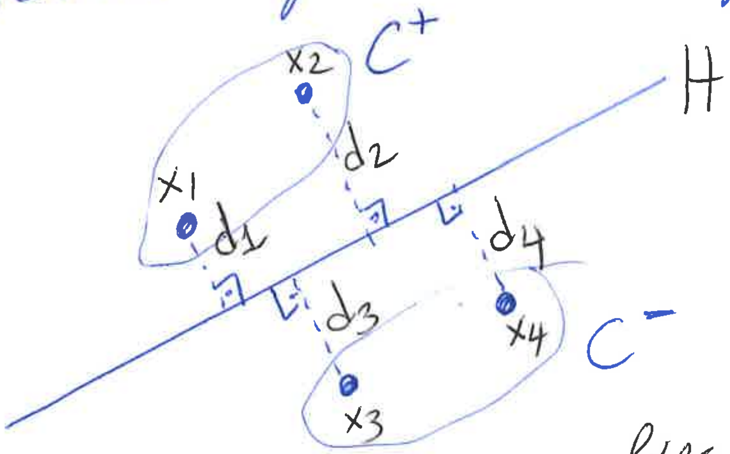
These class-representative-based approaches work quite decently when the classes are geometrically well-separated, e.g.

But, if class members are close, intertwined, we may have problems.

Support Vector Machines (finally!)

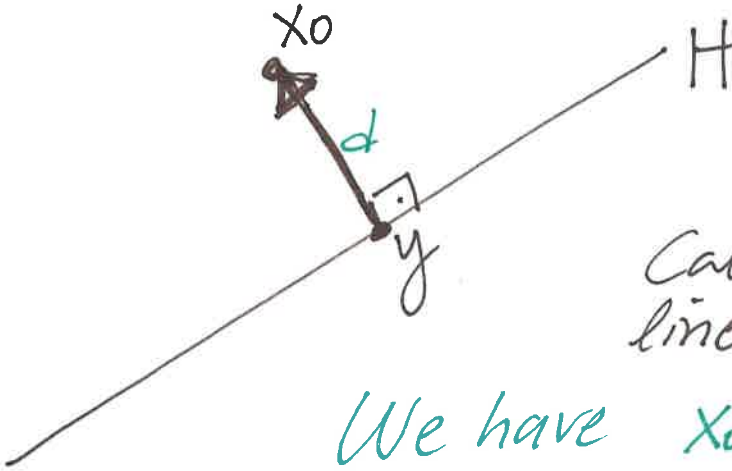
- Could be considered as an extension of the minimum distance classifier to the case of more than one class member in the classes.
- We will first study what is called the "linear SVM."
- Suppose we have classes $C^+ \subset \mathbb{R}^d$ and $C^- \subset \mathbb{R}^d$ that are linearly separable.
- The desired outputs for the patterns $x_1, \ldots, x_n$ are given by $d_1, \ldots, d_n \in \{-1, +1\}$.
- In particular $x_i \in C^+ \Rightarrow d_i = 1$, and $x_i \in C^- \Rightarrow d_i = -1$.
- Since the classes are linearly separable, PTA can be used to find the weights & bias of a neuron that can seperate the two classes.
- But, as we mentioned, there are infinitely many solutions. Which one is the best?

In the case of linear SVM, the best classifier should maximize the minimum distance of the input patterns to the seperating hyperplane, while satisfying the desired outputs. For example, for the diagram on the right, the best H should maximize $\min\{d_1, d_2, d_3, d_4\}$ while making sure that x_1 & x_2 lies on one side of H and x_3 & x_4 lies on the other side.



To design such a classifier, we recall the formula for calculating the distance of a point x_0 to the hyperplane $H = \{x : w^T x + \theta = 0\}$.

Obviously if $x_0 \in H$, then the distance $= 0$. So, suppose x_0 is on the "+" side of H, i.e. $w^T x_0 + \theta > 0$. We have the diagram below on the next page:



We draw a line that passes through x_0 and perpendicular to H.

Call the intersection of this line with H as point y.

We have $x_0 = y + d \frac{w}{\|w\|}$

(Here, we have used the fact that w is perpendicular to H and points to the + direction)

Multiplying both sides by w^T, we obtain

$$w^T x_0 = w^T y + d \frac{w^T w}{\|w\|}$$

Adding θ to both sides, we have

$$w^T x_0 + \theta = \underbrace{w^T y + \theta}_{= 0 \text{ (as } y \text{ is on } H)} + d \frac{w^T w}{\|w\|}.$$

Using $w^T w = \|w\|^2$, we get

$$d = \frac{w^T x_0 + \theta}{\|w\|}$$

Recall that we assumed x_0 to be on the + side so that $w^T x_0 + \theta > 0$. This implies

$$\boxed{d = \frac{|w^T x_0 + \theta|}{\|w\|}} \Longrightarrow$$

A similar calculation shows that this formula holds when x_0 is on the − side as well.

Now recall the linear SVM problem.
We have points $x_1, \dots, x_n$ with desired classes $d_1, \dots, d_n \in \{-1, +1\}$. We can formulate the linear SVM as the following optimization problem:

$$\max_{w,\theta} \min_i \frac{|w^T x_i + \theta|}{\|w\|} \quad \text{s.t} \quad \begin{array}{l} w^T x_i + \theta \geqslant 0 \text{ if } d_i = 1 \\ w^T x_i + \theta \leqslant 0 \text{ if } d_i = -1 \\ i = 1, \dots, n \end{array}$$

The constraints represent the correct classification requirements. It is easy to see that they are equivalent to the conditions $d_i(w^T x_i + \theta) \geqslant 0$, $i = 1, \dots, n$.

Also, letting $\gamma = \min_i |w^T x_i + \theta|$, the optimization problem above is equivalent to:

$$\max_{\substack{w,\theta \\ \gamma \geqslant 0}} \frac{\gamma}{\|w\|} \quad \text{s.t} \quad \begin{array}{l} d_i(w^T x_i + \theta) \geqslant 0, \; i = 1, \dots, n. \\ \gamma = \min_i |w^T x_i + \theta| \end{array}$$

But $\gamma = \min_i |w^T x_i + \theta|$ if and only if

$$|w^T x_i + \theta| \geqslant \gamma \quad \forall i \in \{1, \dots, n\}.$$

Therefore, we equivalently have to solve:

$$\max_{\substack{w,\theta \\ \gamma \geqslant 0}} \frac{\gamma}{\|w\|} \quad \text{s.t} \quad \begin{array}{l} d_i(w^T x_i + \theta) \geqslant 0 \\ |w^T x_i + \theta| \geqslant \gamma \end{array} \quad i = 1, \dots, n.$$

Again, the two sets of constraints can easily be shown to be equivalent to $d_i(w^T x_i + \theta) \geqslant \gamma$, $i = 1, \dots, n$.
As a result we obtain,

$$\max_{\substack{w,\theta \\ \gamma \geqslant 0}} \frac{\gamma}{\|w\|} \quad \text{s.t} \quad d_i(w^T x_i + \theta) \geqslant \gamma, \; i = 1, \dots, n.$$

Considering the substitutions $w \leftarrow \frac{w}{\gamma}$, $\theta \leftarrow \frac{\theta}{\gamma}$ we obtain:

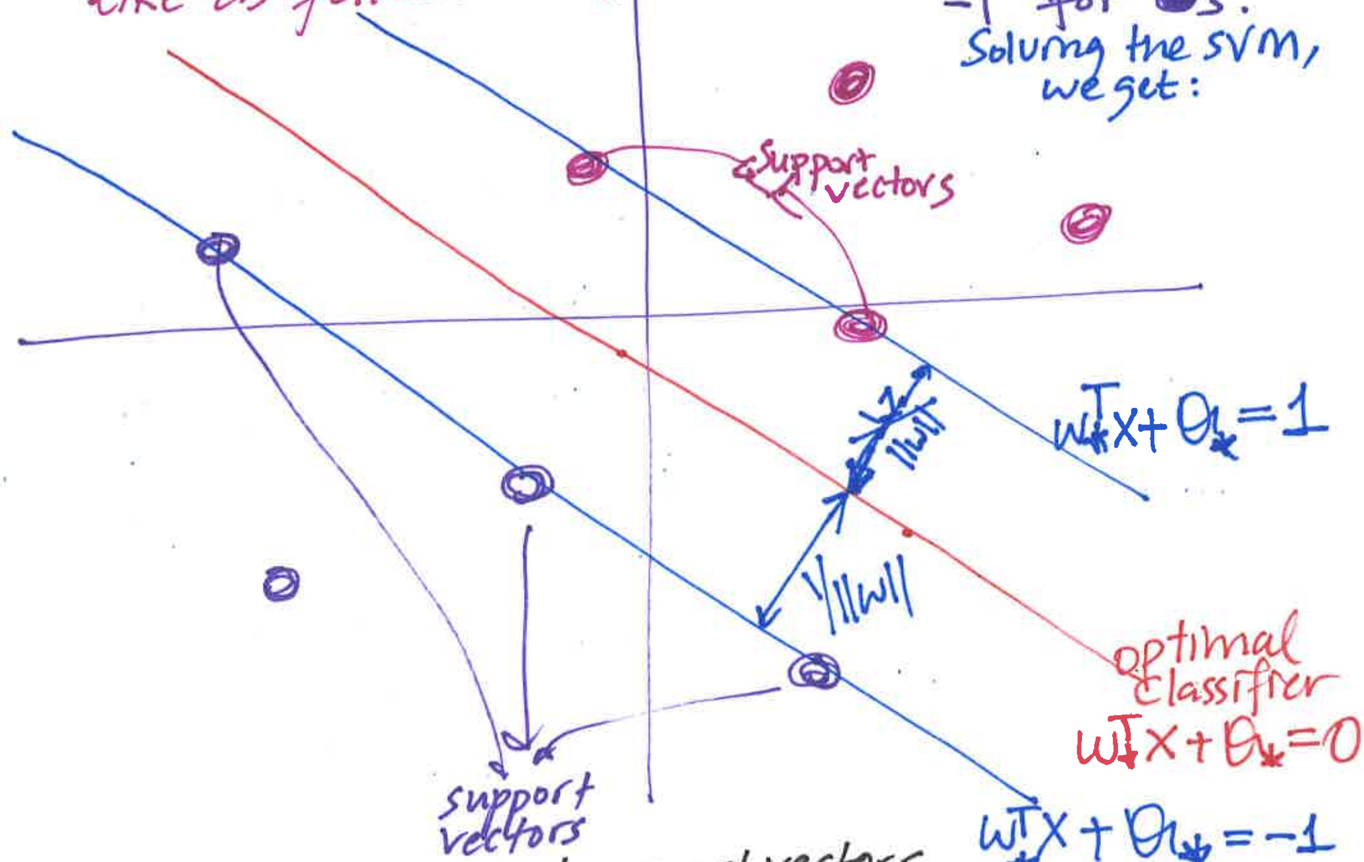
$$\max_{w,\theta} \frac{1}{\|w\|} \quad \text{s.t} \quad d_i(w^T x_i + \theta) \geq 1, \quad i = 1, \dots, n \;\dots\; \text{X}$$

Now, suppose we solve this optimization problem somehow to get solutions w_* and θ_*. We will have

$$d_i(w_*^T x_i + \theta_*) = 1 \quad \text{for } i \in I \text{, and}$$
$$d_i(w_*^T x_i + \theta_*) > 1 \quad \text{for } i \in \{1, \dots, n\} - I$$

where $I \subset \{1, \dots, n\}$ is an index set.

These are the active constraints of the optimization problem. The corresponding x_i, $i \in I$ are called the support vectors. The geometry looks like as follows: Suppose we want $+1$ for ●s, and -1 for ●s.



One reason these are called support vectors is that if you remove any non-support vector from the set of inputs and resolve the SVM, you get the same solution.

Equation (X) on top of the previous page is equivalent to:

$$\min_{w, \theta} \frac{1}{2}\|w\|^2 \text{ s.t. } d_i(w^T x_i + \theta) \geq 1, \; i=1,\dots,n$$

This is a minimization of a quadratic function subject to linear constraints and there are many software packages to solve this kind of problems. Yet, another form of the optimization problem is much more suitable for "Kernel ~~SVM~~" that we will introduce later.

To obtain this alternate form, we will use Lagrange multipliers $\alpha_1, \dots, \alpha_n \geq 0$. The optimization problem on top of this page is equivalent to:

$$\min_{w, \theta} \max_{\substack{\alpha_1, \dots, \alpha_n \\ \geq 0}} \underbrace{\left[\frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i \left(d_i(w^T x_i + \theta) - 1 \right) \right]}_{= \mathcal{L}(\alpha, w, \theta)} \quad \cdots (Y)$$

To see this suppose all constraints are satisfied. Then, the solution to the inner maximization is $\alpha_1 = \dots = \alpha_n = 0$, and we effectively minimize the "original" objective function $\frac{1}{2}\|w\|^2$. On the other hand, suppose one of the constraints, say, with index $k \in \{1, \dots, n\}$ is violated. Then, the solution to the inner maximization satisfies $\alpha_k = \infty$. This makes the entire expression inside the square brackets in (Y) equal to ∞. The corresponding w, θ the cannot be a solution to the optimization as they result in ∞ cost.

The optimization problem (Y) in the previous page can be solved using what is called Karush-Kuhn-Tucker conditions. These conditions say (we omit the proof here) that

$$\min_{w,\theta} \max_{\alpha_i \geq 0} \mathcal{L}(\alpha, w, \theta) = \max_{\alpha_i \geq 0} \min_{w,\theta} \mathcal{L}(\alpha, w, \theta)$$

and, moreover, the optimal solutions θ^*, w^* make the gradients of $\mathcal{L}(\alpha, w, \theta)$ vanish, i.e.

$$\nabla_w \mathcal{L}(\alpha, w, \theta)\Big|_{w=w^*} = 0 \quad \text{and}$$

$$\frac{\partial \mathcal{L}(\alpha, w, \theta)}{\partial \theta}\Bigg|_{\theta=\theta^*} = 0$$

The first zero gradient condition yields

$$w^* = \sum_i \alpha_i d_i x_i$$

and the second condition yields

$$\sum_i \alpha_i d_i = 0.$$

Therefore, we have to solve:

$$\max_{\substack{\alpha_i \geq 0 \\ \sum_i \alpha_i d_i = 0}} \mathcal{L}\Big(\alpha, \sum_i \alpha_i d_i x_i, \theta^*\Big)$$

Substituting, we obtain:

$$\max_{\substack{\alpha_i \geq 0 \\ \sum_i \alpha_i d_i = 0}} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j d_i d_j x_i^T x_j \right)$$

This is a quadratic optimization problem with a linear constraint. Can be solved via standard libraries for optimization. Suppose the solutions are (with some abuse of notation) $\alpha_1, \dots, \alpha_n$.

Vectors x_i with $\alpha_i > 0$ are support vectors.

We can find the weights as $w^* = \sum_i \alpha_i d_i x_i$.

How to find the bias θ?

<u>Solution</u>: Get an index k such that $\alpha_k > 0$ (i.e, x_k is a support vector). We have

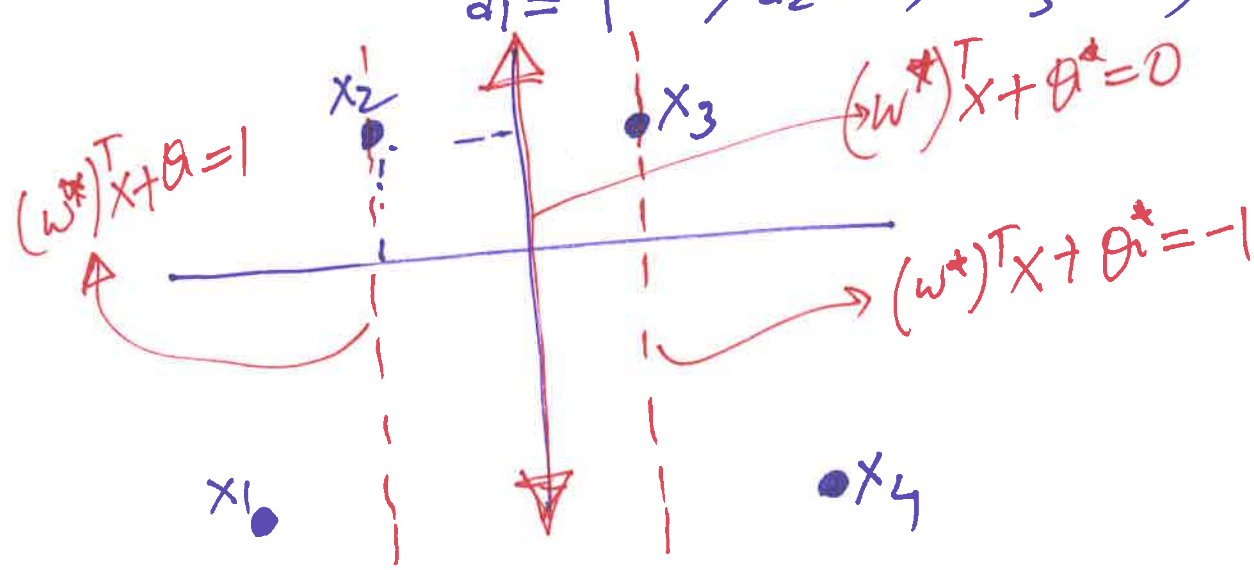
$$d_k (w^{*T} x_k + \theta^*) = 1, \text{ or equivalently}$$

$$w^{*T} x_k + \theta^* = d_k$$

$$\Rightarrow \theta^* = d_k - w^{*T} x_k.$$

<u>Example</u>. Suppose $x_1 = \begin{pmatrix} -2 \\ -2 \end{pmatrix}$, $x_2 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$, $x_3 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $x_4 = \begin{pmatrix} 2 \\ -2 \end{pmatrix}$

$d_1 = 1$, $d_2 = 1$, $d_3 = -1$, $d_4 = -1$.



Solving the optimization problem, we obtain

$\alpha_1 = 0$, $\alpha_2 = \alpha_3 = \frac{1}{2}$, $\alpha_4 = 0$, resulting in

$w^* = \sum_i \alpha_i d_i x_i = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$. The support vectors are x_2 and x_3.

Since $\cancel{x_2}$ is a support vector $(\alpha_2 > 0)$,

we obtain, $\theta^* = d_2 - (w^*)^T x_2 = 0$

We would have gotten the same result if we used instedd x_3.

An easy way to find support vectors / solve linear SVM.

Consider an index i with $d_i = 1$, and index j with $d_j = -1$.

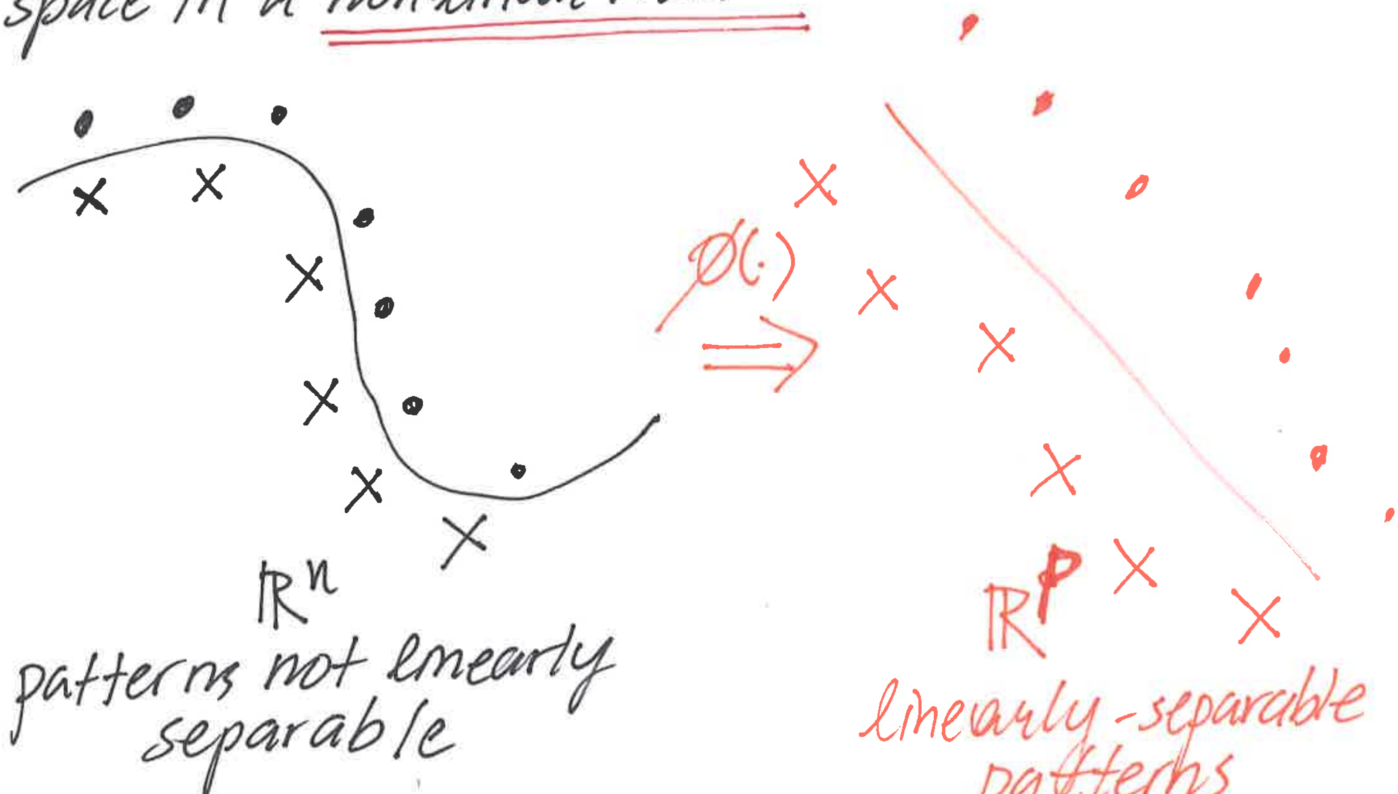
Consider the minimum distance classifier seperating x_i and x_j. Obviously, both x_i and x_j are within distance $\frac{1}{2}\|x_i - x_j\|$ to the classifying hyperplane.

Then, if the distance of all other points to the hyperplane are $\geq \frac{1}{2}\|x_i - x_j\|$, then the minimum distance classifier we already found is the solution to the linear SVM.

- Example: In the example on the previous page, consider the min. distance classifier for x_2 & x_3. Both x_2 & x_3 are 1-far to the classifier. The distance of x_1 & x_4 are $= 2 \geq 1$. So, the min. dist. classifier for x_2 & x_3 is also the solution to the linear SVM.

Kernel SVM.

* What if the training set is not linearly separable?
* Intuition: Patterns that are not linearly separable are more "likely" to be linearly separable when they are mapped to a higher dimensional space in a non-linear manner.



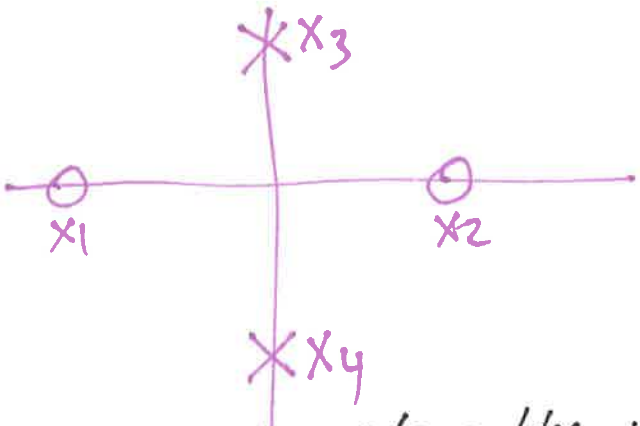
usually the dimension of the target space is chosen to be much larger than the dimension of the original space. $p \gg n$.

In fact, there are cases where one works with $p = \infty$!

- How to choose $\phi(\cdot)$? Typically this is induced by a "Kernel function" which can be a polynomial, Gaussian, etc. kernel (to be discussed in detail later).

Example: $x_1 = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$, $x_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $x_3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, $x_4 = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$

$d_1 = 1$, $d_2 = 1$, $d_3 = -1$, $d_4 = -1$.



: patterns are not linearly separable in the original space.

Now, consider the mapping $\phi(x) = \begin{pmatrix} x_1 \\ x_2 \\ x_1^2 \end{pmatrix} \in \mathbb{R}^3$,

where $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ is a vector in the original two-dimensional space $\mathbb{R}^2$.

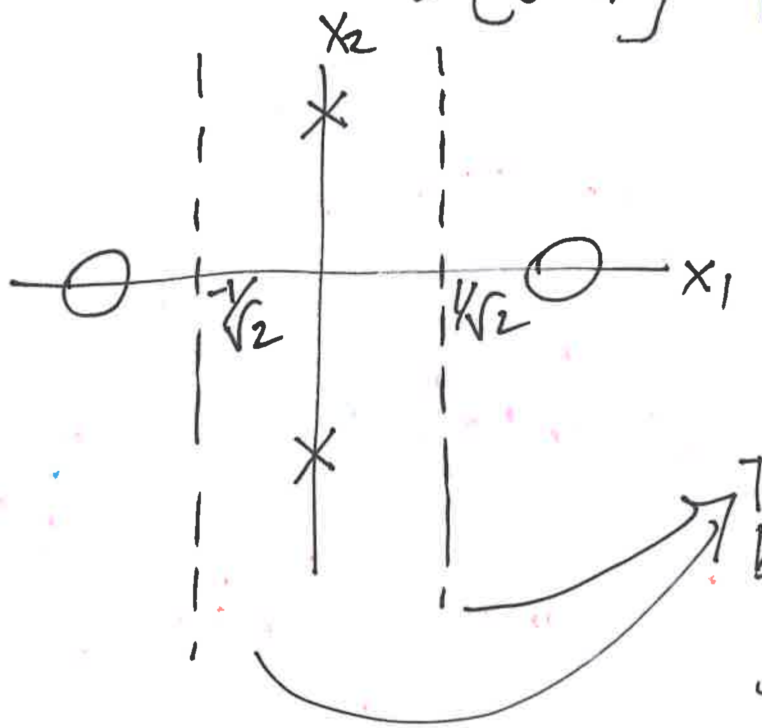
The vectors $\phi(x_1) = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$, $\phi(x_2) = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$, $\phi(x_3) = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$, $\phi(x_4) = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}$

are linearly seperable in the 3D space.

In fact, solving the linear SVM in the 3D space, we obtain the hyperplane $2(x_3 - \frac{1}{2}) = 0$.

In other words, the discriminant function in the 3D space is $g\left[\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}\right] = 2(x_3 - \frac{1}{2})$.

The discriminant function in the original space is then $g\left[\begin{pmatrix}x_1\\x_2\end{pmatrix}\right] = 2\left(x_1^2 - \frac{1}{2}\right)$.



These are the decision boundaries where $g\left[\begin{pmatrix}x_1\\x_2\end{pmatrix}\right] = 0 \Leftrightarrow x_1 = \mp\frac{1}{\sqrt{2}}$

- The target space is typically called the feature space, and the mapping ϕ is called the feature mapping.

- Of course, we would like to find the optimal seperator in the feature space, i.e. solve the linear SVM in the feature space as we did in the example on the previous page.
- For this purpose recall the dual problem

$$L(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j d_i d_j x_i^T x_j$$

$$\max_\alpha L(\alpha) \text{ s.t } \alpha_i \geq 0, \quad \sum_i \alpha_i d_i = 0.$$

Solution: $g(x) = \underbrace{\sum_i \alpha_i d_i x_i^T}_{w^T} x + \theta$

$\theta = d_k - w^T x_k$, where x_k is a support vector.

- We could solve linear SVM in the feature space by just replacing x_i's by $\phi(x_i)$s.

So, define.

$$L(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j d_i d_j (\phi(x_i))^T \phi(x_j)$$

Solve $\max_\alpha L(\alpha)$ s.t $\alpha_i \geq 0$, $\sum_i \alpha_i d_i = 0$.

Use the solution to calculate

$$w = \sum_i \alpha_i d_i \phi(x_i).$$

Find a support vector $\alpha_k > 0$. Then, calculate

$$\theta = d_k - w^T \phi(x_k)$$

* The problem is that the feature space may have a very high dimension and it may not even be feasible to calculate w and θ.

- Kernel trick: We only need the classifier

$$g(x) = w^T \cancel{\phi(x)} + \theta$$
$$= \sum_i \alpha_i d_i (\phi(x_i))^T \phi(x) + \theta,$$

where $\theta = d_k - \sum_{i=1}^{n} \alpha_i d_i (\phi(x_i))^T \phi(x_k)$

- We observe that everything can be written in terms of a Kernel function

$$K(x, y) = (\phi(x))^T \phi(y).$$

For example, we have,

$$L(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j d_i d_j K(x_i, x_j)$$

$$g(x) = \sum_i \alpha_i d_i K(x_i, x) + \theta_1, \text{ where}$$

$$\theta_1 = d_K - \sum_{i=1}^{n} \alpha_i d_i K(x_i, x_k).$$

So, we only need to know the Kernel function, we do not need to actually work with the feature vectors!

Example Kernels :

Linear : $K(x_i, x_j) = x_i^T x_j$ (linear svm)

Polynomial : $K(x_i, x_j) = (1 + x_i^T x_j)^d$, d is typically an integer.

Gaussian : $K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}}$, $\sigma > 0$