HALMSTAD UNIVERSITY

Learning Systems (DT8008)

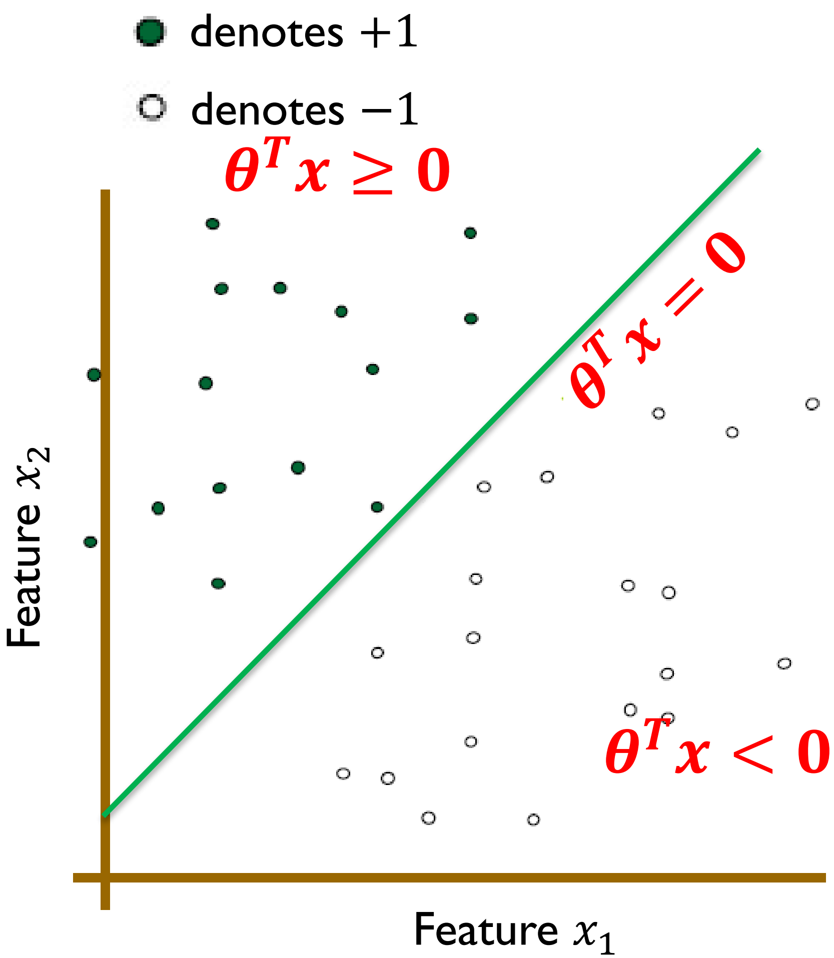
Support Vector Machines (SVM)

Dr. Mohamed-Rafik Bouguelia
mohamed-rafik.bouguelia@hh.se

Halmstad University

Intuition behind the SVM classifier

Intuition behind SVM

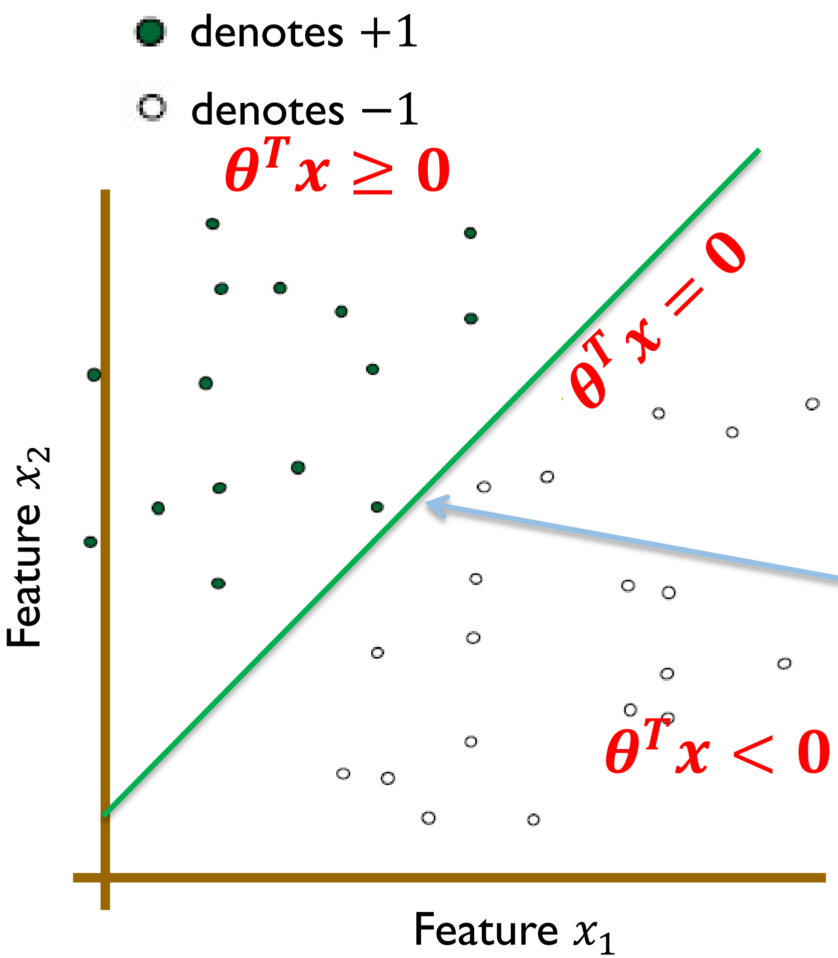


Usually, in linear classification, we try to find a hyperplane $\theta^T x = 0$ (i.e. plan if $d = 3$, or line if $d = 2$), that correctly classifies the training data-points (or most of them).

$$h_\theta(x) = sign(\theta^T x) = \begin{cases} +1 \; if \; \theta^T x \geq 0 \\ -1 \; if \; \theta^T x < 0 \end{cases}$$

NOTE: assuming $x_0 = 1$ for all data-points.

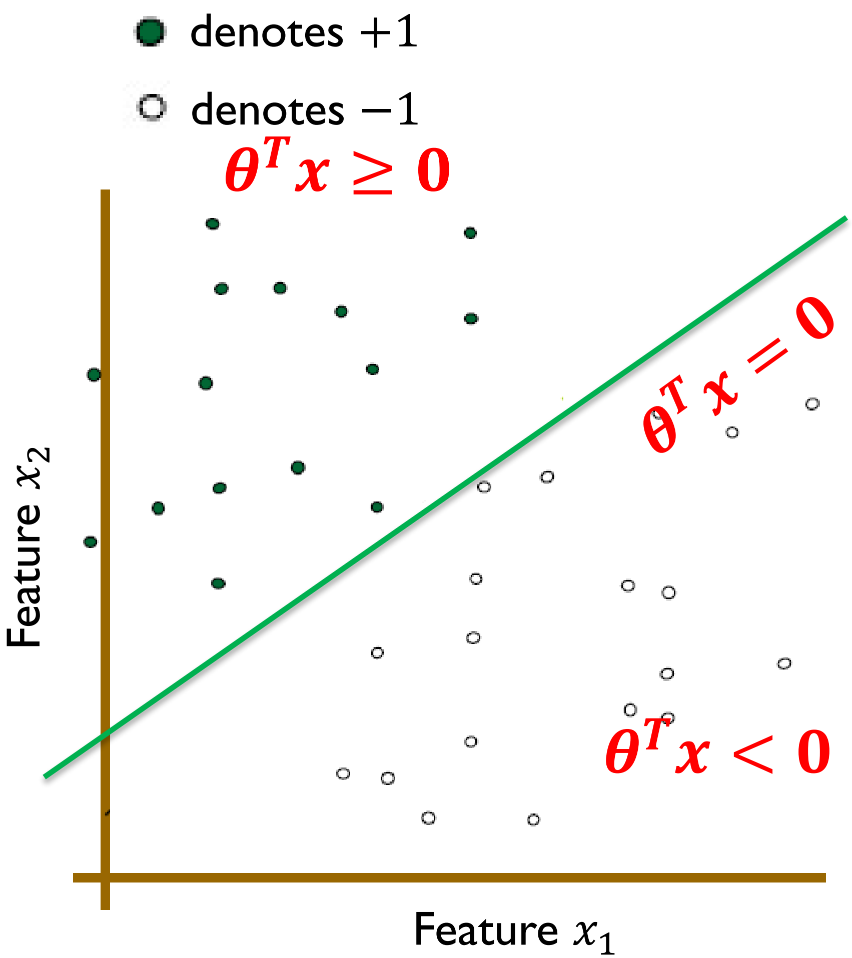
Intuition behind SVM



Usually, in linear classification, we try to find a hyperplane $\theta^T x = 0$ (i.e. plan if $d = 3$, or line if $d = 2$), that correctly classifies the training data-points (or most of them).

In this example, it can be this line.

Intuition behind SVM

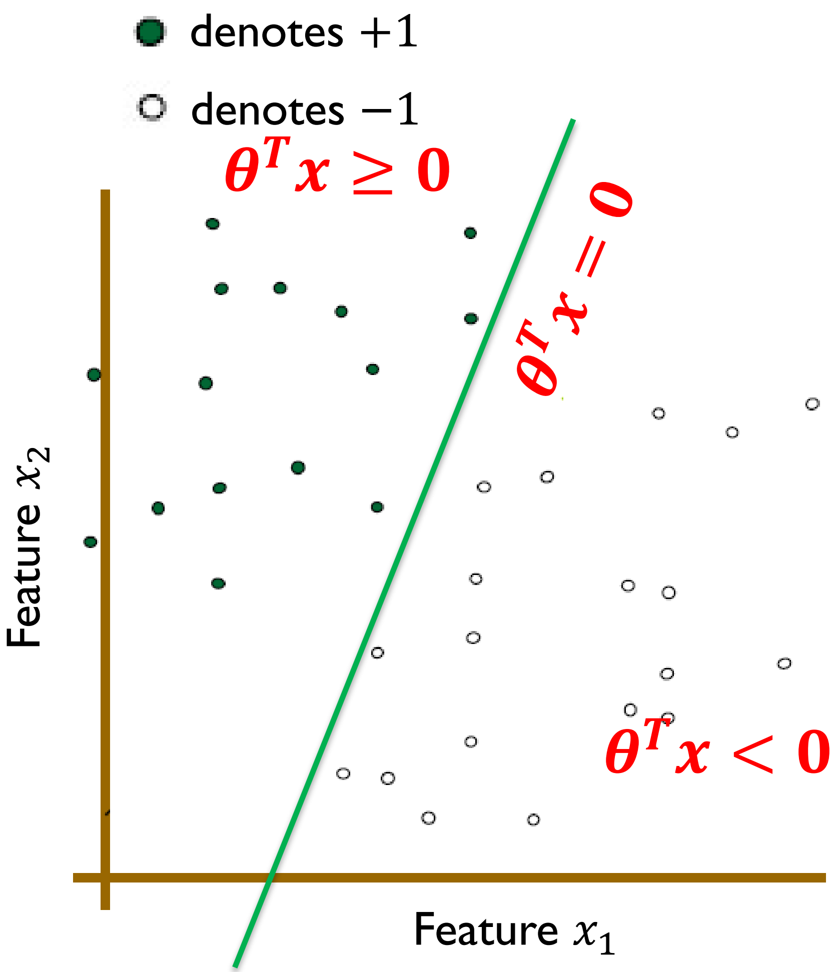


Usually, in linear classification, we try to find a hyperplane $\theta^T x = 0$ (i.e. plan if $d = 3$, or line if $d = 2$), that correctly classifies the training data-points (or most of them).

… or this line …

HALMSTAD UNIVERSITY

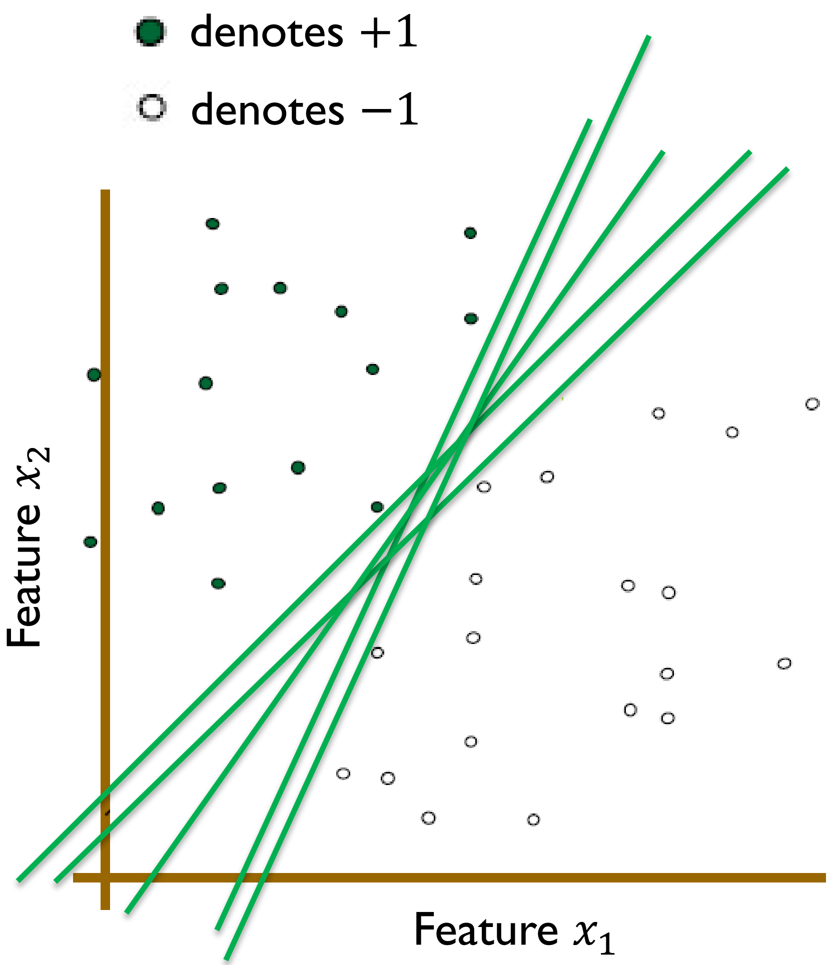
Intuition behind SVM



Usually, in linear classification, we try to find a hyperplane $\theta^T x = 0$ (or plan in 3d, line in 2d), that correctly classifies the training data-points (or most of them).

… or maybe this line …

Intuition behind SVM

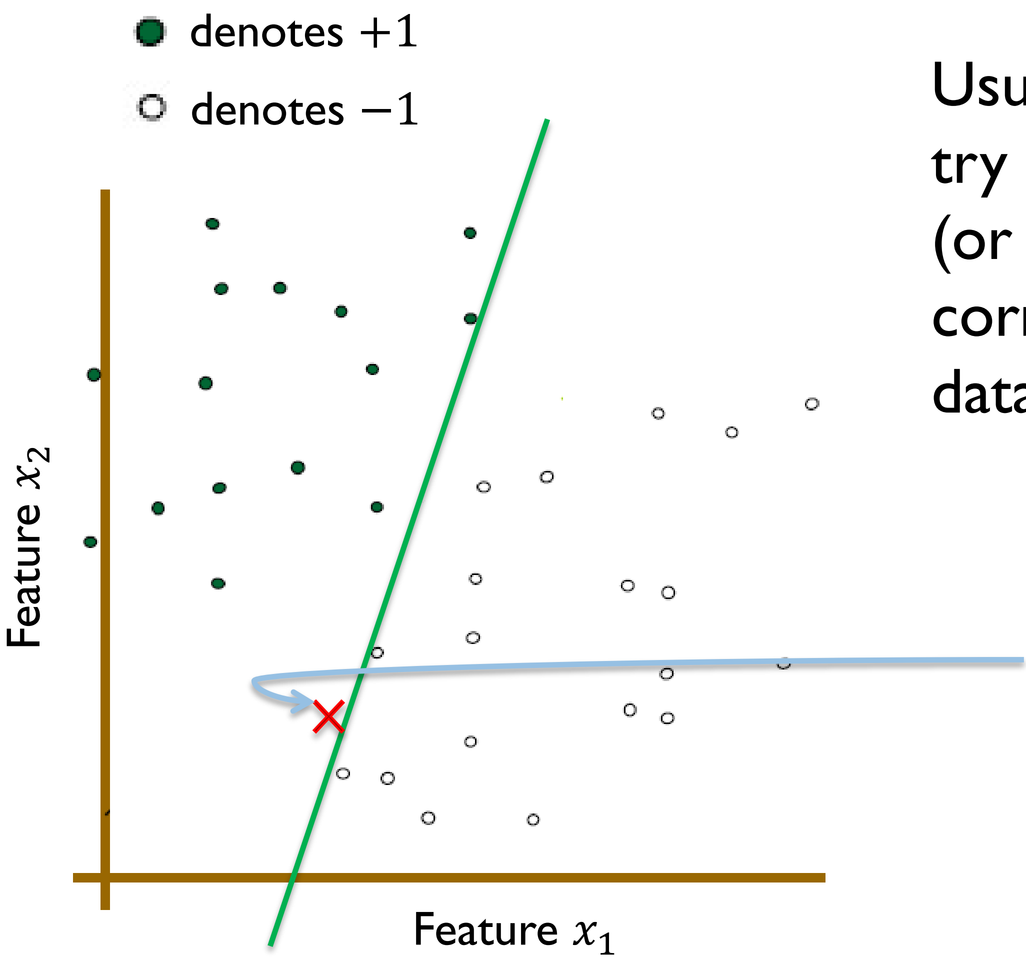


Usually, in linear classification, we try to find a hyperplane $\theta^T x = 0$ (or plan in 3d, line in 2d), that correctly classifies the training data-points (or most of them).

… Any of these lines would also be fine. There is an infinite number of such lines.

But which one is best?

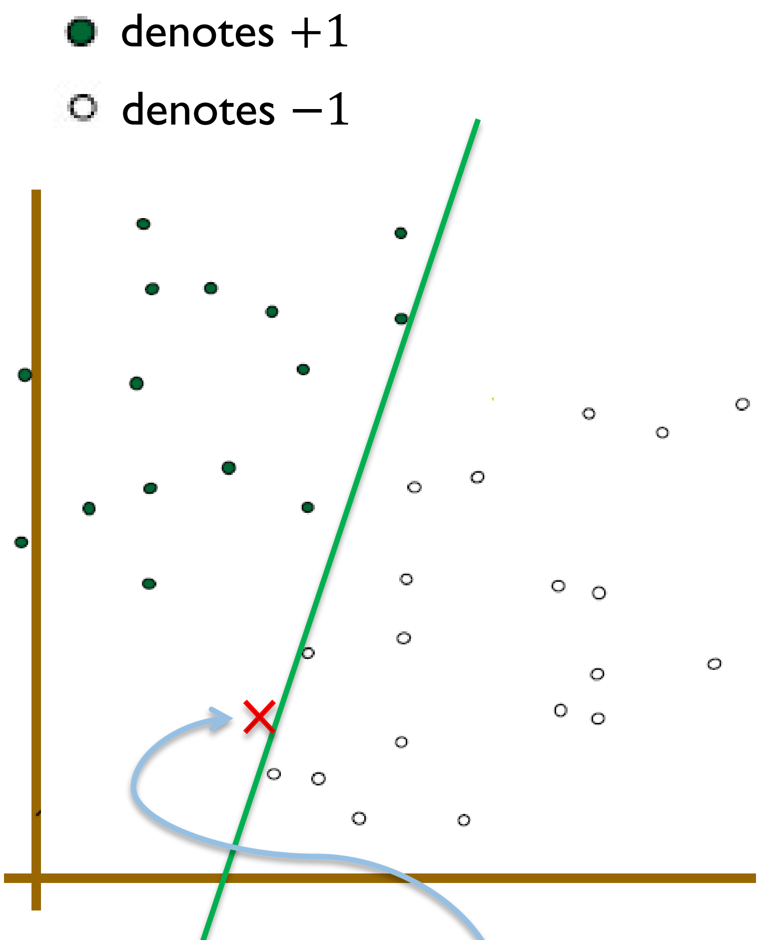
Intuition behind SVM



Usually, in linear classification, we try to find a hyperplane $\theta^T x = 0$ (or plan in 3d, line in 2d), that correctly classifies the training data-points (or most of them).

How would you classify this new data-point $\times$?

Intuition behind SVM



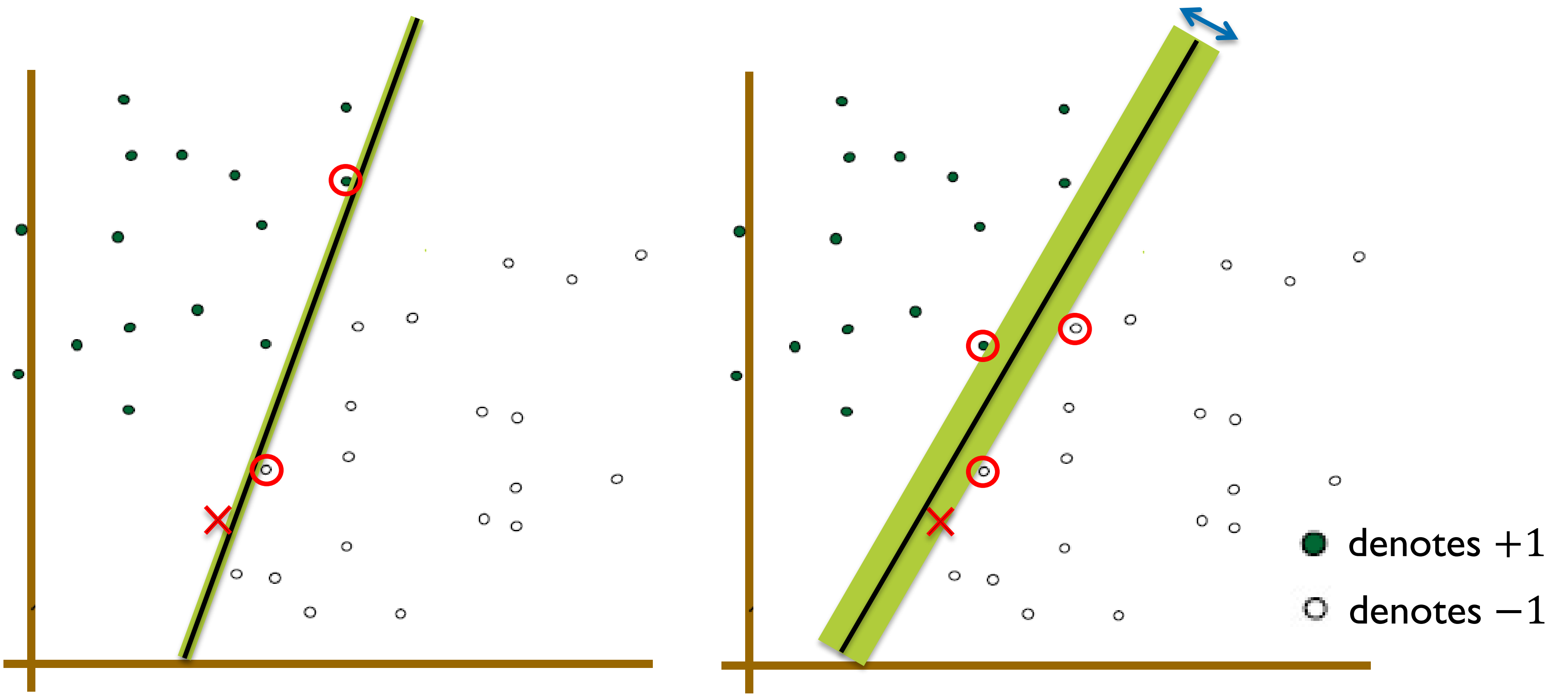
Usually, in linear classification, we try to find a hyperplane $\theta^T x = 0$ (or plan in 3d, line in 2d), that correctly classifies the training data-points (or most of them).

How would you classify this new data-point × ?

Looks very similar to the points in the -1 class, but <u>misclassified</u> into the $+1$ class

Intuition behind SVM

- Define the margin of a linear classifier as the width that the boundary could be increased by, before hitting a data point.

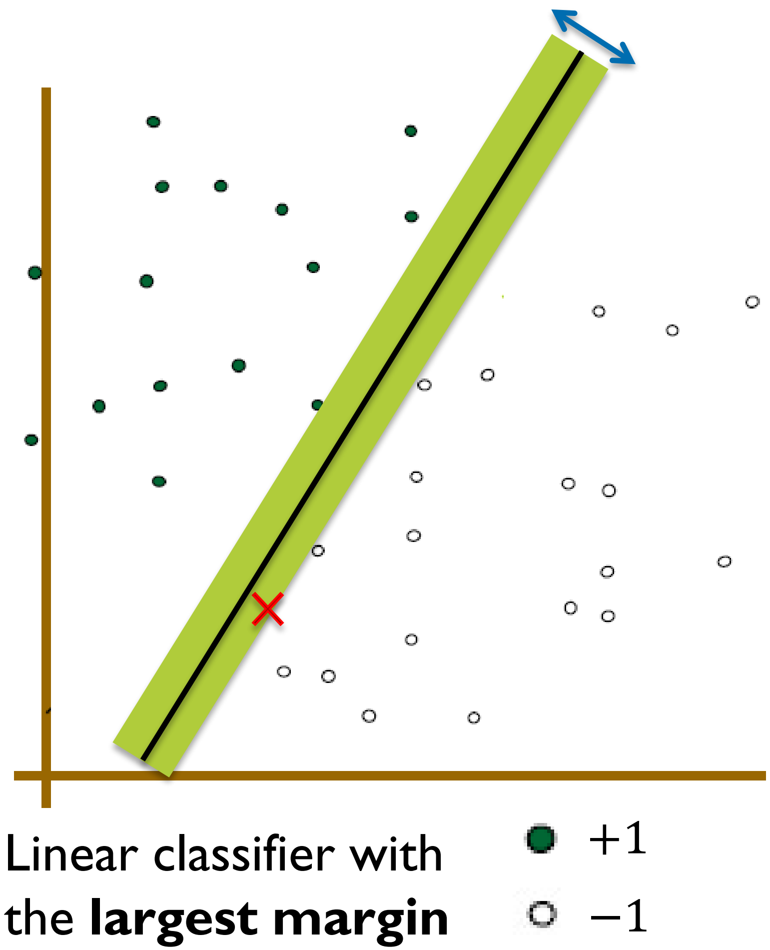


Linear classifier with a **small margin**

Linear classifier with a **large margin**

Intuition behind SVM

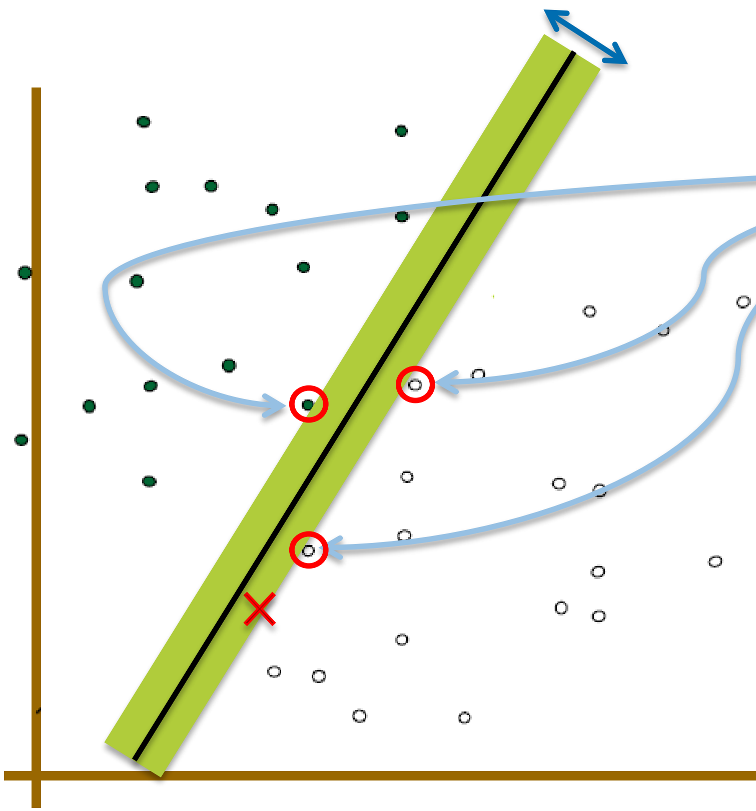
- The maximum margin linear classifier is the linear classifier with the maximum margin. It is unique.



Linear classifier with the **largest margin**

Intuition behind SVM

- The maximum margin linear classifier is the linear classifier with the maximum margin. It is unique.



Support vectors are those data-points that the margin pushes up against

Linear classifier with the **largest margin**

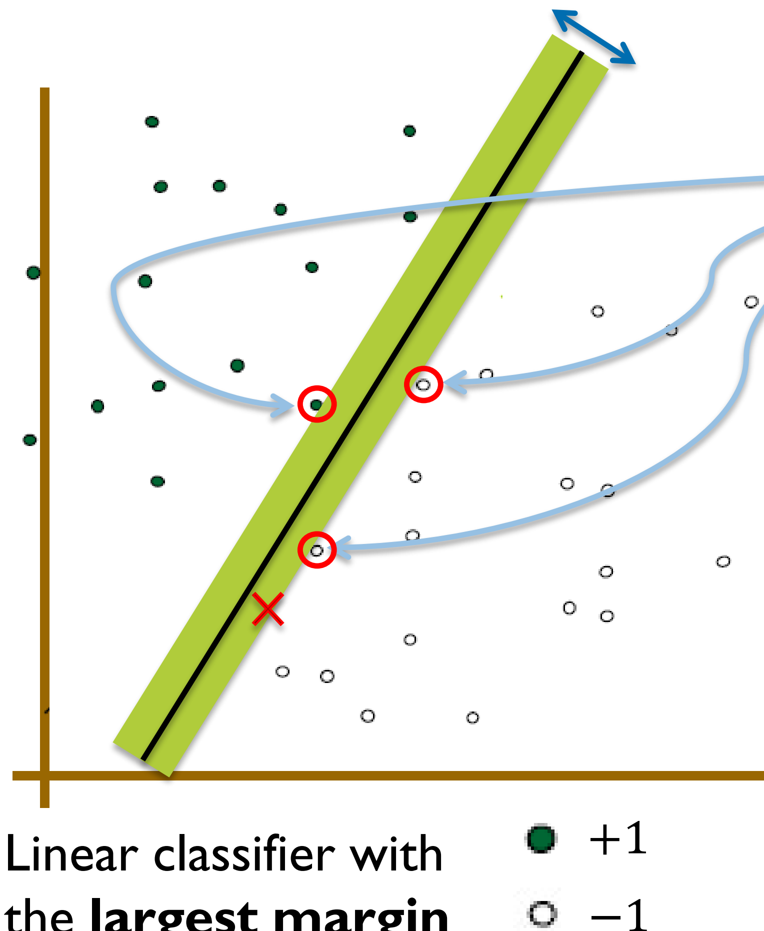
$+1$

-1

HALMSTAD UNIVERSITY

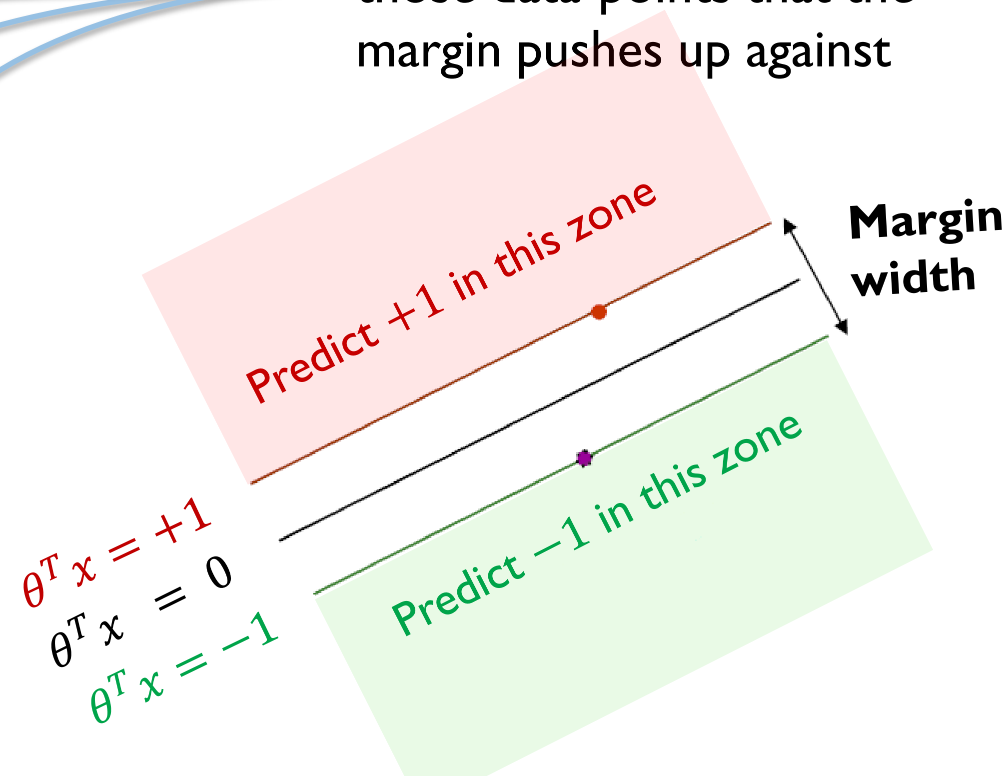
Intuition behind SVM

- The maximum margin linear classifier is the linear classifier with the maximum margin. It is unique.



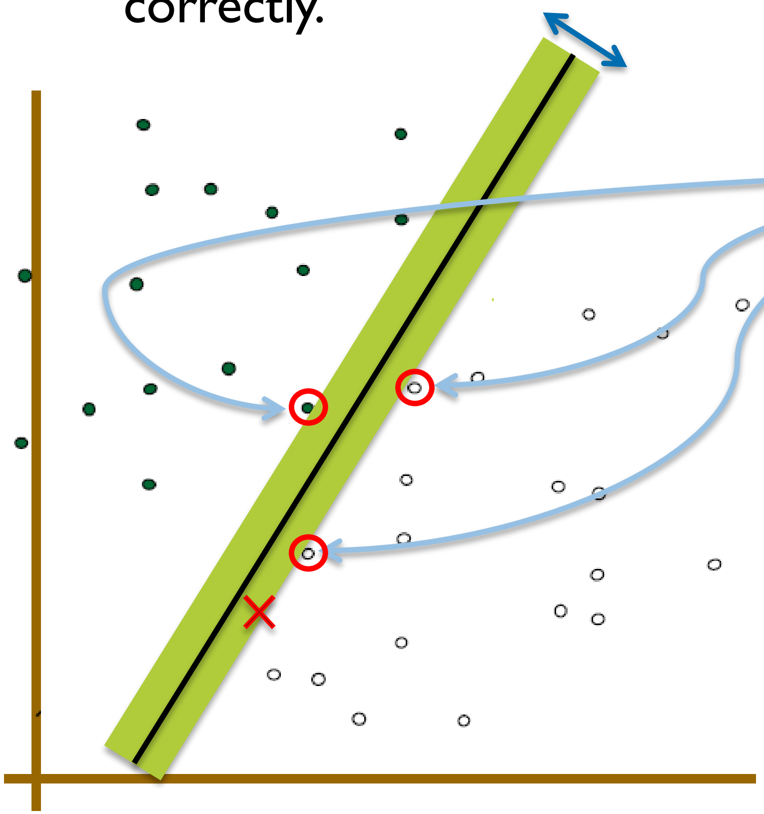
Linear classifier with the **largest margin**

Support vectors are those data-points that the margin pushes up against

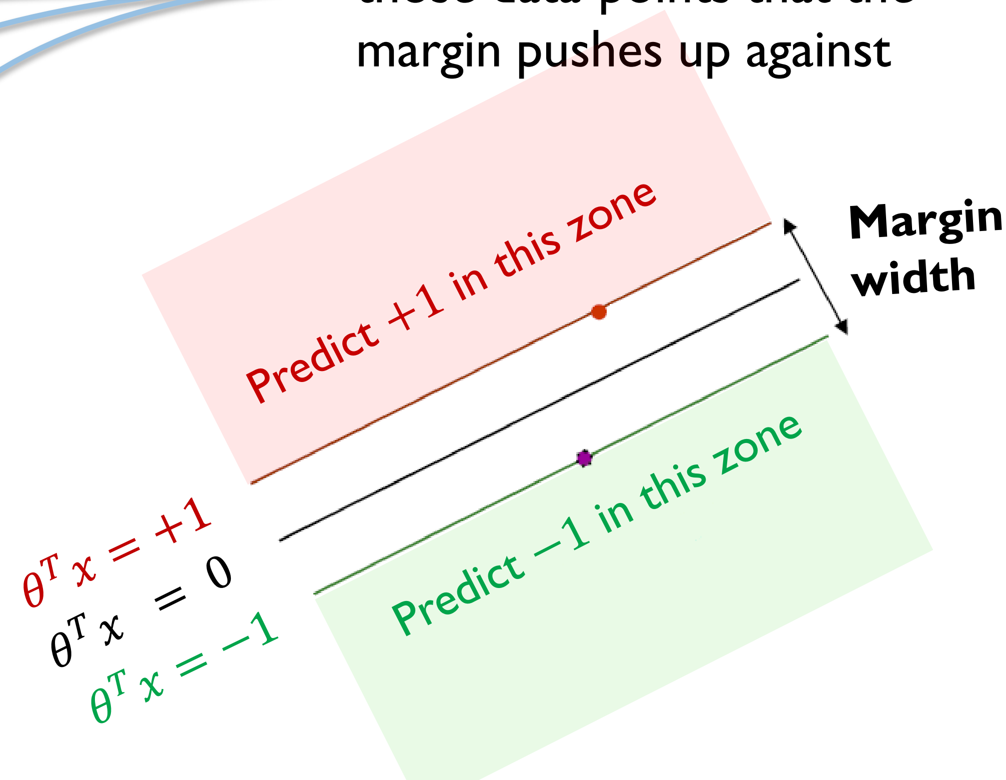


Intuition behind SVM

- This is the main principal behind the **simplest version of SVM**. It finds the hyperplane with the maximum margin, that separates the two classes correctly.



Linear classifier with the **largest margin**

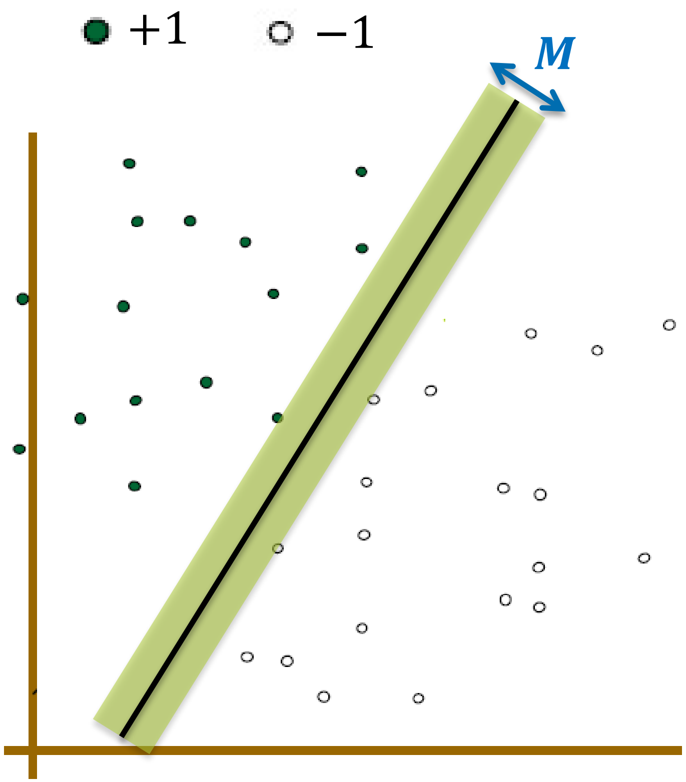


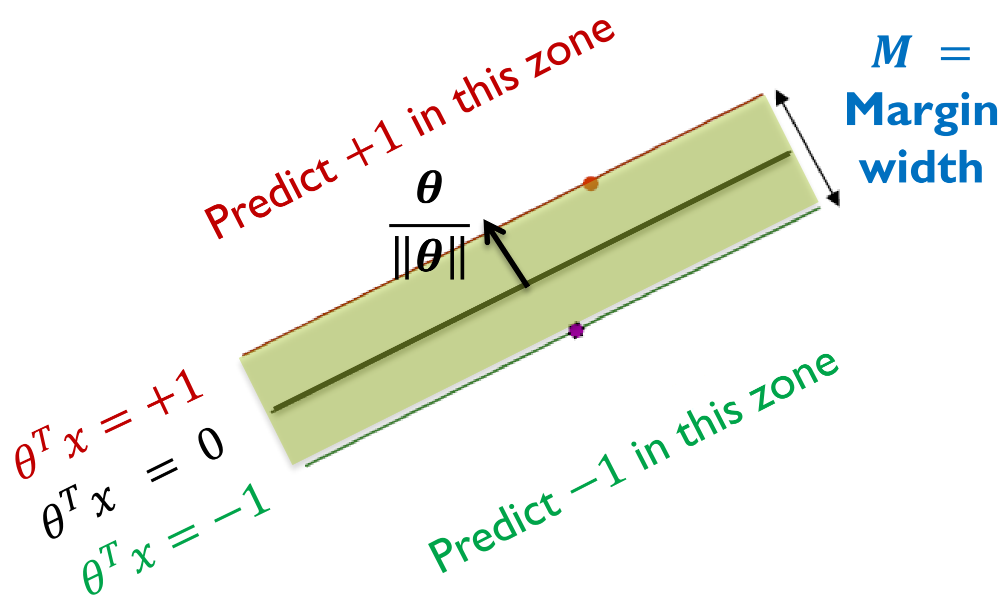
Defining the Optimization Problem for a Simple Linear SVM

Simple Linear SVM Optimization Problem

Two objectives:

1. We want to find the hyperplane with the largest margin $\boldsymbol{M}$.
2. We want the hyperplane to correctly classify all training data-points.
   - We will see how to relax this 2nd objective later.





HALMSTAD UNIVERSITY

Simple Linear SVM Optimization Problem

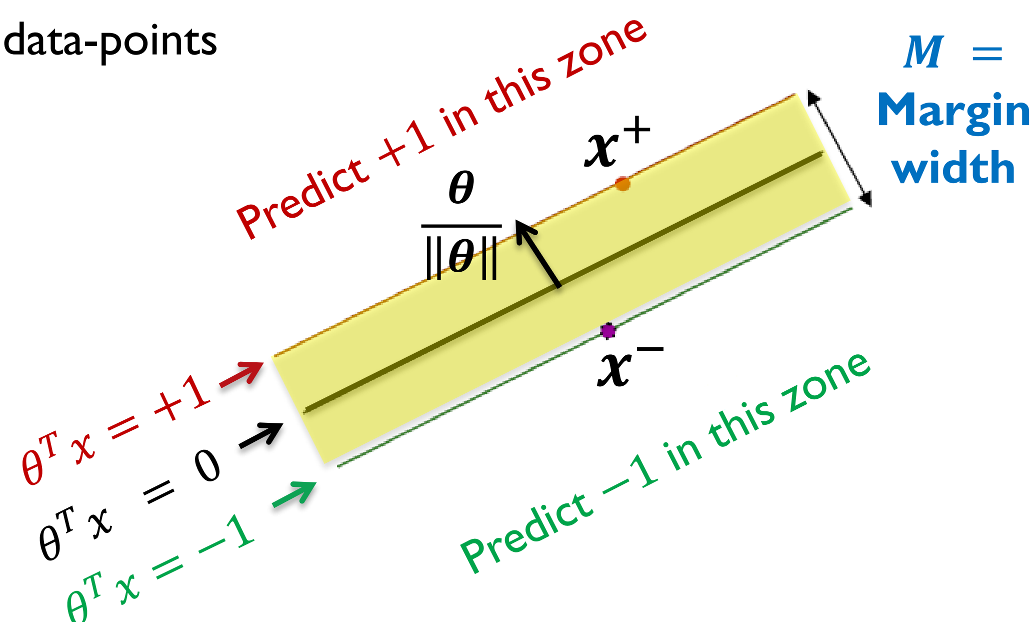
Two objectives:

1. We want to find the hyperplane with the largest margin $\boldsymbol{M}$.

- The margin area where $-1 < \theta^T x^+ < +1$ (in yellow) does not contain any training data-points.
- We want to predict $+1$ for all data-points where $\theta^T x \geq +1$
- We want to predict -1 for all data-points where $\theta^T x \leq -1$

Let x^+ be a point on the 1st extremity of the margin and x^- be a point on the 2nd extremity of the margin. So:

$$\theta^T x^+ = +1$$
$$\theta^T x^- = -1$$



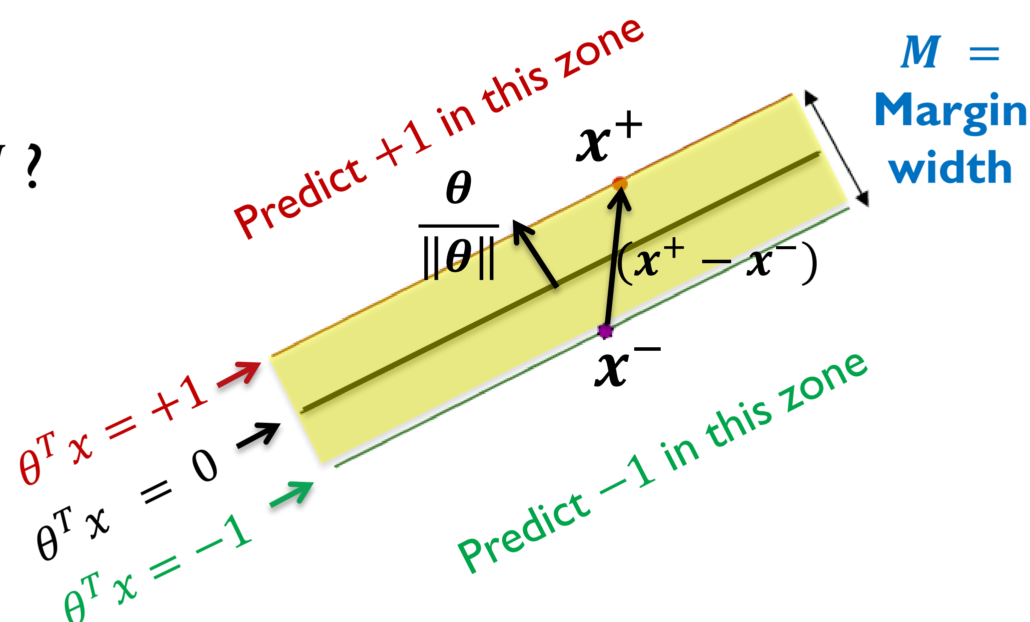
Simple Linear SVM Optimization Problem

Two objectives:

1. We want to find the hyperplane with the largest margin $\boldsymbol{M}$.

$$\left.\begin{aligned}\theta^T x^+ &= +1 \\ \theta^T x^- &= -1\end{aligned}\right\} \quad \theta^T(x^+ - x^-) = 2$$

What is the margin width M ?



Simple Linear SVM Optimization Problem

Two objectives:

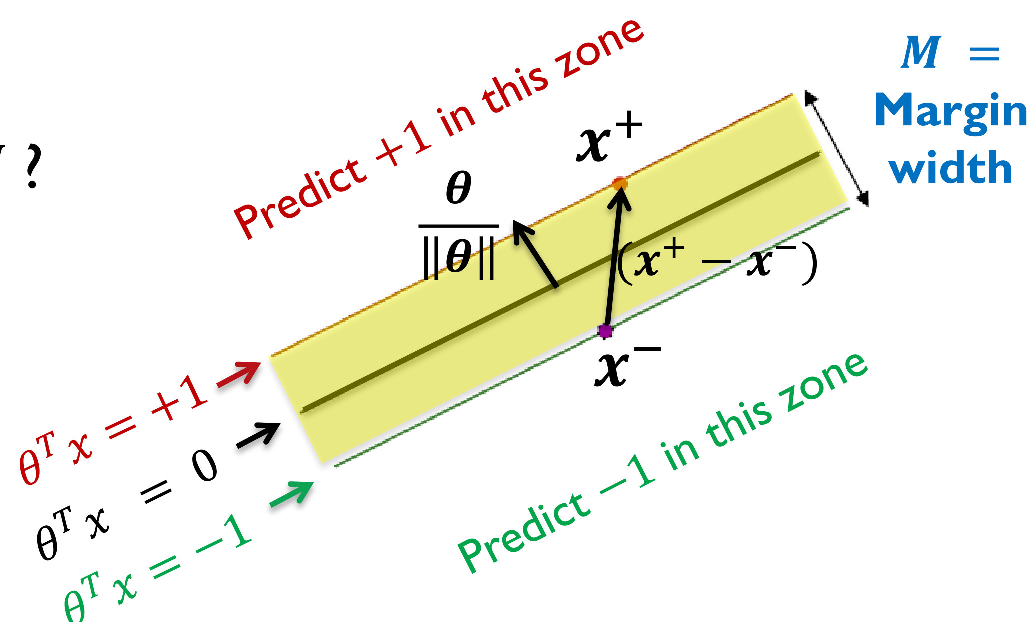
1. We want to find the hyperplane with the largest margin $\boldsymbol{M}$.

$$\left.\begin{aligned}\theta^T x^+ &= +1\\ \theta^T x^- &= -1\end{aligned}\right\} \quad \theta^T(x^+ - x^-) = 2$$

What is the margin width M ?

$$M = \frac{\theta^T(x^+ - x^-)}{||\theta||} = \frac{2}{||\theta||}$$

So, as a 1st objective, we want to maximize $\boldsymbol{M} = \frac{\mathbf{2}}{||\boldsymbol{\theta}||}$



Simple Linear SVM Optimization Problem

Two objectives:

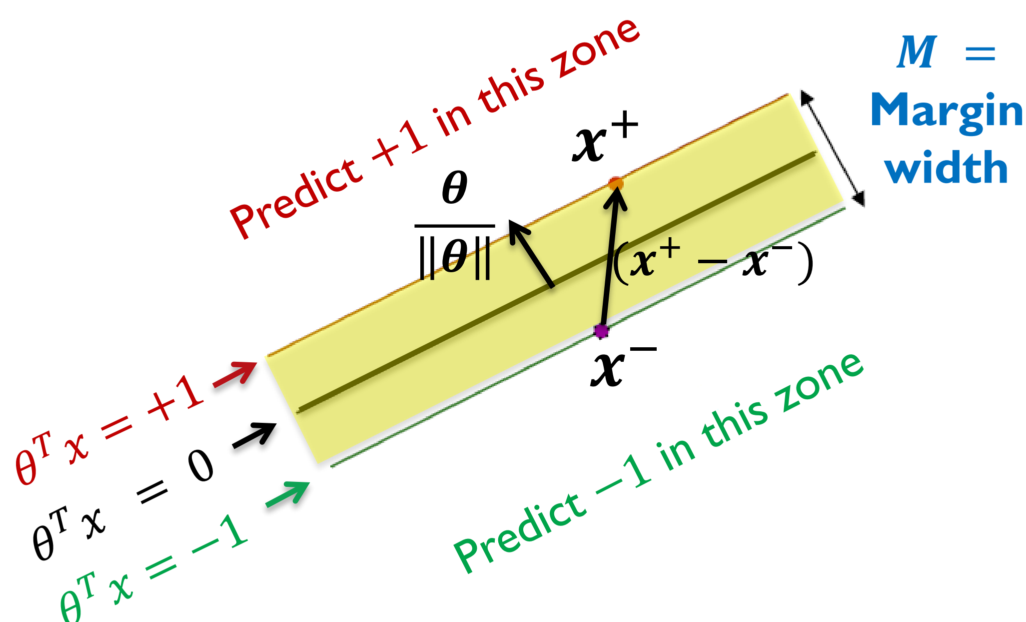
1. We want to find the hyperplane with the largest margin $\boldsymbol{M}$.

As a 1st objective, we want to maximize $M = \frac{2}{\|\theta\|}$

Is this objective alone sufficient ? No.

By choosing a parameter vector θ with $\|\theta\| \approx 0$, you will maximize this objective. But such θ wouldn't be useful.

So we need some constraints … (2nd objective)



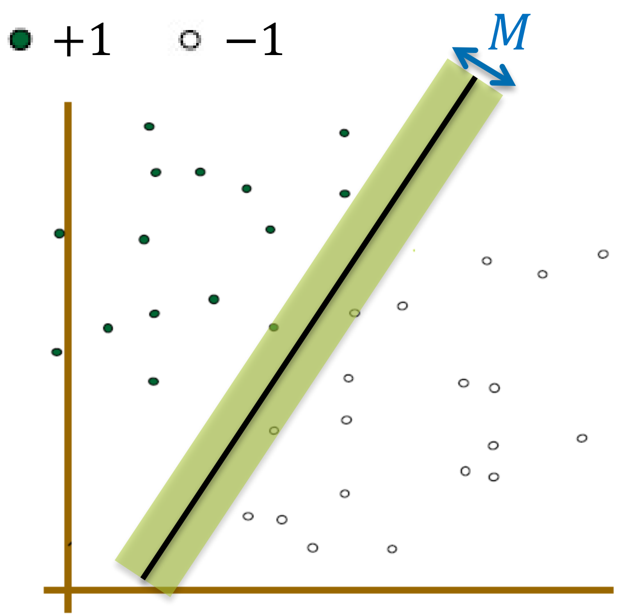
HALMSTAD UNIVERSITY

Simple Linear SVM Optimization Problem

Two objectives:

1. We want to find the linear hyperplane with the largest margin $\boldsymbol{M}$.
   ➔ $\max_{\theta} M = \frac{2}{\|\theta\|}$

2. We want the hyperplane to correctly classify all training data-points.
   - We will see how to relax this 2nd objective later.



- How do we formalize this 2nd objective ?
  Each training data-point is correctly classified …

HALMSTAD UNIVERSITY

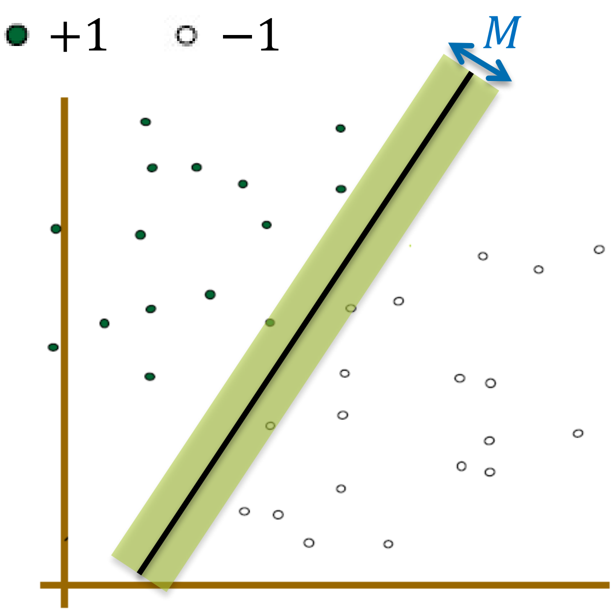
Simple Linear SVM Optimization Problem

Two objectives:

1. We want to find the linear hyperplane with the largest margin $\boldsymbol{M}$.

   ➔ $$\max_{\theta} M = \frac{2}{\|\theta\|}$$

2. We want the hyperplane to correctly classify all training data-points.
   - We will see how to relax this 2nd objective later.



$$\left.\begin{array}{l} \theta^T x^{(i)} \geq +1 \quad \text{if} \quad y^{(i)} = +1 \\ \theta^T x^{(i)} \leq -1 \quad \text{if} \quad y^{(i)} = -1 \end{array}\right] \text{for } i = 1, \dots, n$$

$$\left. \boldsymbol{y^{(i)}\, \theta^T x^{(i)} \geq +1} \right] \text{ for } i = 1, \dots, n$$

Simple Linear SVM Optimization Problem

Two objectives:

1. We want to find the linear hyperplane with the largest margin $\boldsymbol{M}$.
   ➔ $\max_{\theta} M = \frac{2}{\|\theta\|}$

2. We want the hyperplane to correctly classify all training data-points.
   - We will see how to relax this 2nd objective later.

   ➔ $y^{(i)}\, \theta^T x^{(i)} \geq 1$ for $i = 1, \ldots, n$

So, our constrained optimization problem is:

$$\max_{\theta} \frac{2}{\|\theta\|}$$

subject to $\quad y^{(i)}\, \theta^T x^{(i)} - 1 \geq 0 \qquad \forall i$

Solving the Optimization Problem of the Simple SVM

Solving the optimization problem

- Constrained optimization problem:

$$\left.\max_{\theta} \frac{2}{\|\theta\|}\right\} \text{ similar to: } \min_{\theta} \frac{1}{2}\|\theta\|^2$$

subject to $y^{(i)}\,\theta^T x^{(i)} - 1 \geq 0 \qquad \forall i \in \{1, \dots, n\}$

- Using the Lagrange multipliers (α_i) it becomes:

$$\min_{\theta} \underbrace{\frac{1}{2}\|\theta\|^2 - \sum_{i=1}^{n} \alpha_i \left(y^{(i)}\,\theta^T x^{(i)} - 1\right)}_{\boldsymbol{L(\theta)}}$$

$L(\theta)$ is a convex function.

We can compute $\frac{\partial L}{\partial \theta} = 0$ and solve for θ

Solving the optimization problem

$$\min_{\theta} \ \frac{1}{2}\|\theta\|^2 - \sum_{i=1}^{n} \alpha_i \ (y^{(i)}\ \theta^T x^{(i)} - 1)$$

For simplification, let's just consider the first term as: $\frac{1}{2}\left\|\hat{\theta}\right\|^2 = \frac{1}{2}\sum_{j=1}^{d} \theta_j^2$

$$L(\theta) = \frac{1}{2}\sum_{j=1}^{d} \theta_j^2 - \sum_{i=1}^{n} \alpha_i \ (y^{(i)}\ \theta^T x^{(i)} - 1)$$

$$\frac{\partial L}{\partial \theta_j} = \theta_j - \sum_{i=1}^{n} \alpha_i\, y^{(i)}\, x_j^{(i)} = 0 \quad \Longrightarrow \quad \theta_j = \sum_{i=1}^{n} \alpha_i\, y^{(i)}\, x_j^{(i)} \qquad \dots\dots\dots (1)$$

$$\frac{\partial L}{\partial \theta_0} = 0 - \sum_{i=1}^{n} \alpha_i\, y^{(i)} = 0 \quad \Longrightarrow \quad \sum_{i=1}^{n} \alpha_i\, y^{(i)} = 0 \qquad \dots\dots\dots (2)$$

Solving the optimization problem

Briefly:

- By replacing (1) into $L(\theta)$ and considering (2) as a constraint, and solving the new optimization problem with respect to α_i, we can find the values of α_i for $i = 0, \dots, \mathrm{n}$

- The new optimization problem becomes:

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

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i \, y^{(i)} = 0$$

$$\alpha_i \geq 0, \text{ for all } \alpha_i$$

- Most of the α_i will be equal to 0.
- Each non-zero α_i indicates that the corresponding $x^{(i)}$ is a **support vector**.
- **Notice that** solving the optimization problem involves computing the dot products $x^{(i)^{T}} x^{(j)}$ between all pairs of training data-points.

Solving the optimization problem

- The solution has the form:
  - $\boxed{\hat{\theta} = \sum_{i=1}^{n} \alpha_i \, y^{(i)} \, x^{(i)}}$, where $\hat{\theta} = \begin{bmatrix} \theta_1 \\ \dots \\ \theta_d \end{bmatrix}$
  - $\boxed{\theta_0 = y^{(k)} - \hat{\theta}^T x^{(k)}}$, where $(x^{(k)}, y^{(k)})$ is any support vector (with $\alpha_i \neq 0$)

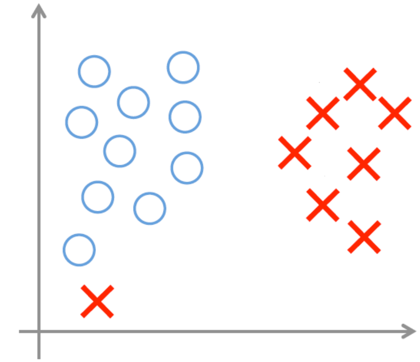
- The hypothesis function is:

$$h_\theta(x) \quad = \quad \theta^T x \quad = \quad \theta_0 + \hat{\theta}^T x \quad = \quad \boxed{\theta_0 + \sum_{i=1}^{n} \alpha_i \, y^{(i)} \boldsymbol{x^{(i)}}^{\boldsymbol{T}} \boldsymbol{x}}$$

  – **Notice that** it relies on a dot product between the test data-point x and the support vectors ${x^{(i)}}^T$

SVM with soft margin
(SVM in the natural form)

SVM with soft margin



- Hard Margin
  – What we saw previously was a simplified SVM, where we required all training data-points to be classified correctly.

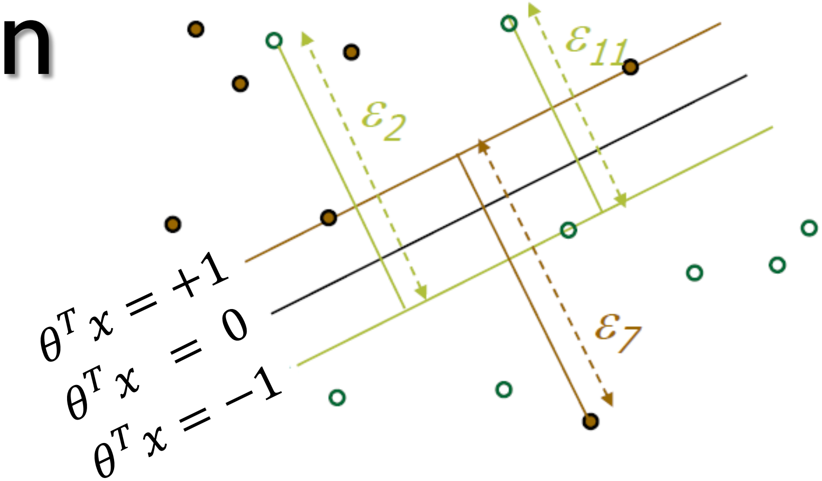
- What if the training dataset is noisy, has outliers, or a hyperplane cannot correctly classify all the training data-points?
  – Soft margin: slack variables ε_i can be added to allow misclassification of difficult data-points (i.e. noise/outliers).

$$\min_{\theta} \underbrace{\frac{1}{2}\left\|\hat{\theta}\right\|^2}_{\text{Margin}} + C \sum_{i=1}^{n} \underbrace{\max\{0, 1 - y^{(i)}\,\theta^T x^{(i)}\}}_{\text{(hinge loss). Cost/Loss of classifying one data-point}}$$

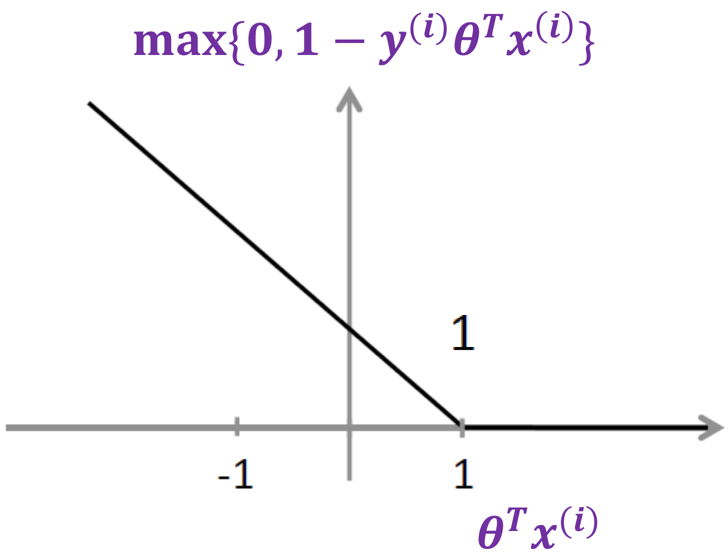
Regularization parameter

SVM with soft margin

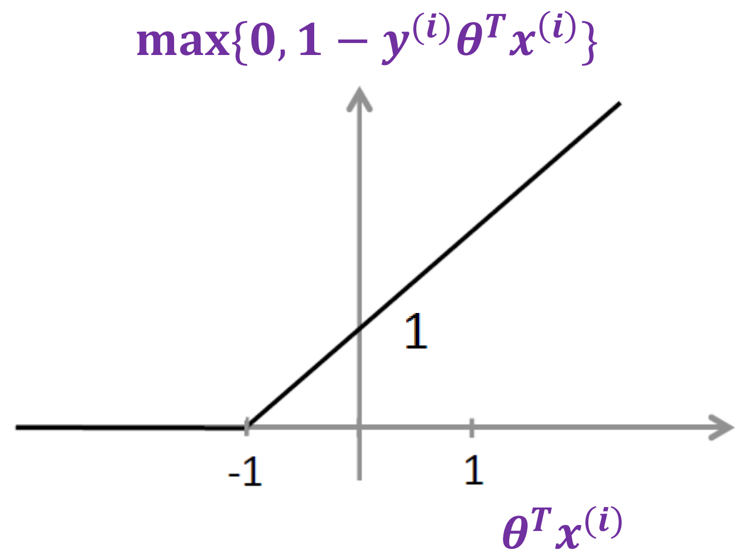
$$\min_{\theta} \; \frac{1}{2}\|\hat{\theta}\|^2 + C\sum_{i=1}^{n} \overbrace{\max\{0, 1 - y^{(i)}\, \theta^T x^{(i)}\}}^{\text{SVM uses the hinge loss}}$$



Case where $y^{(i)} = +1$



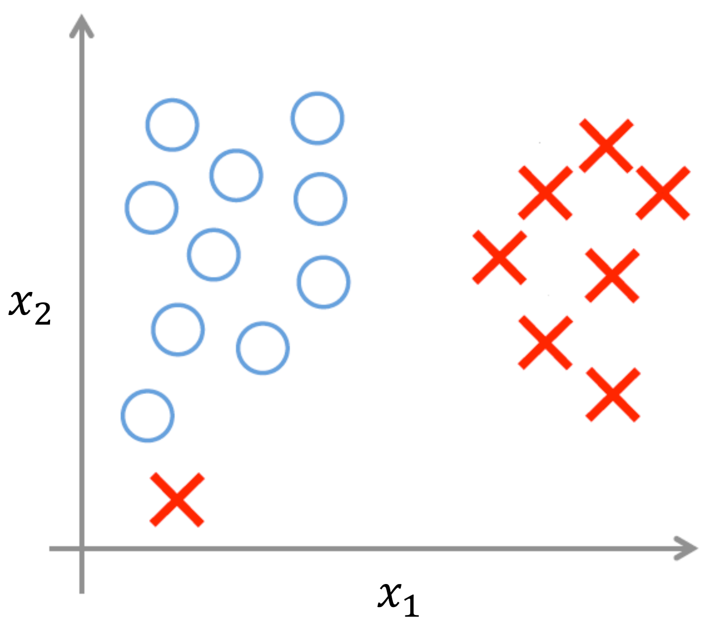
Case where $y^{(i)} = -1$



SVM with soft margin

$$\min_{\theta} \quad \frac{1}{2}\|\hat{\theta}\|^2 + C\sum_{i=1}^{n} \max\{0, 1 - y^{(i)}\,\theta^T x^{(i)}\}$$

- Parameter C can be viewed as a way to control overfitting.
  - Trade-off between
    - Having a large margin
    - Classifying correctly (with small cost/loss) the training data-points.

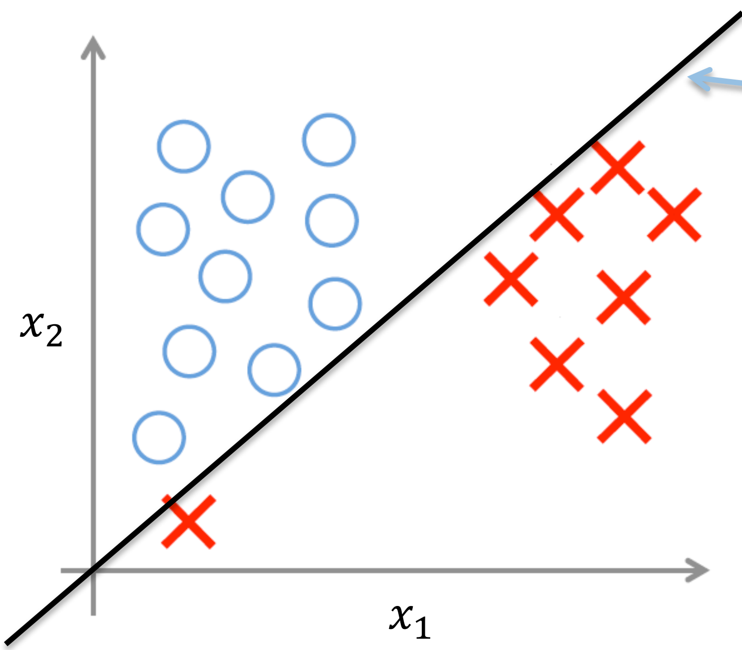


In this example, what would be the linear decision boundary **if C is very large** ?

SVM with soft margin

$$\min_{\theta} \; \frac{1}{2}\|\hat{\theta}\|^2 + C\sum_{i=1}^{n} \max\{0, 1 - y^{(i)}\,\theta^T x^{(i)}\}$$

- Parameter C can be viewed as a way to control overfitting.
  - Trade-off between
    - Having a large margin
    - Classifying correctly (with small cost/loss) the training data-points.



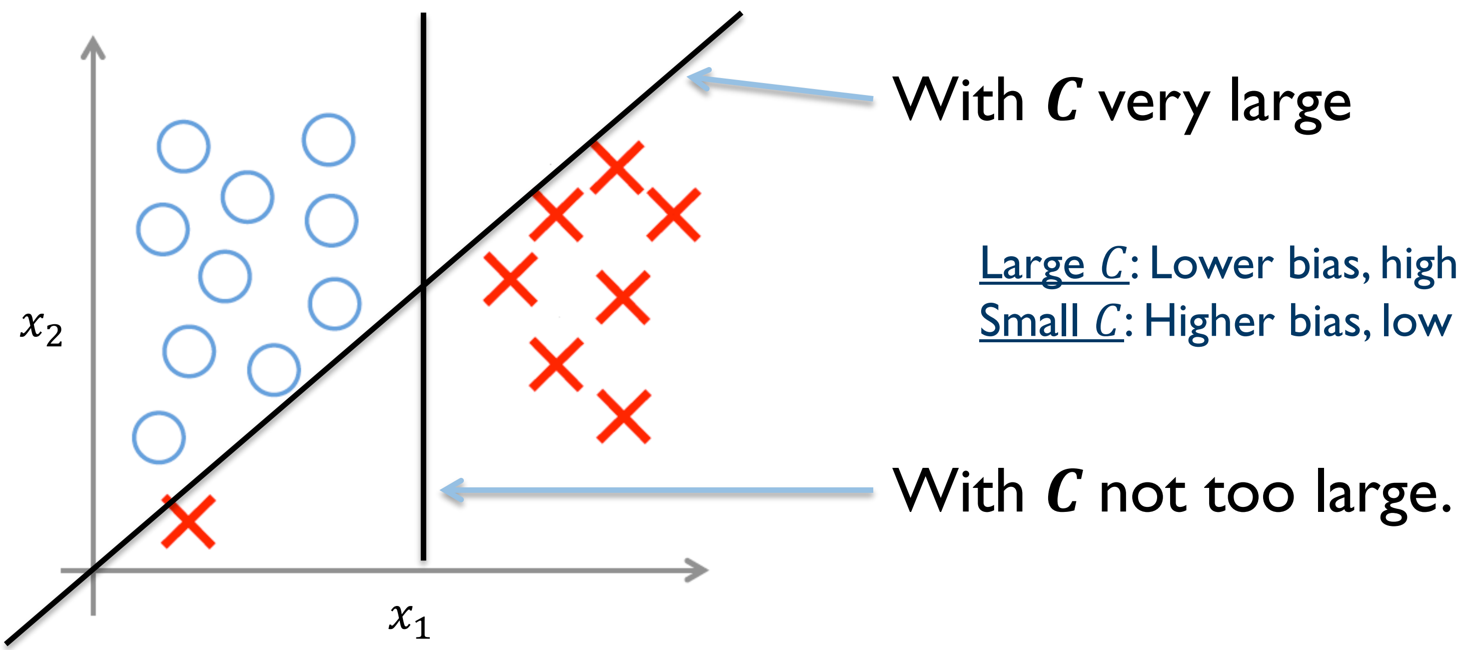
With C very large

It becomes similar to the hard margin. It will try to classify all the training data correctly; but will not generalize well.

SVM with soft margin

$$\min_{\theta} \; \frac{1}{2}\|\hat{\theta}\|^2 + C\sum_{i=1}^{n} \max\{0, 1 - y^{(i)}\,\theta^T x^{(i)}\}$$

- Parameter C can be viewed as a way to control overfitting.
  - Trade-off between
    - Having a large margin
    - Classifying correctly (with small cost/loss) the training data-points.



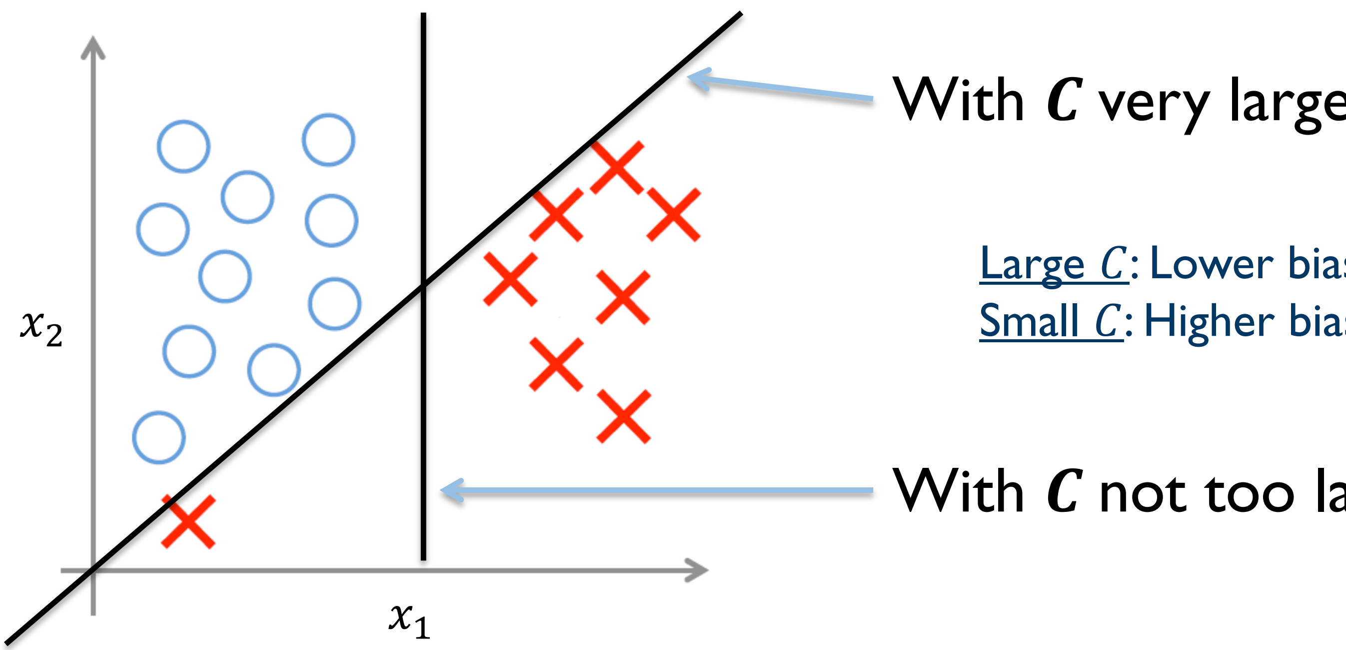
Large C: Lower bias, high variance.
Small C: Higher bias, low variance.

SVM with soft margin

So the regularization param C plays the inverse role of the λ regularization param that you have seen in the previous lectures.

$$\min_{\theta} \quad \frac{1}{2}\|\hat{\theta}\|^2 + C\sum_{i=1}^{n} \max\{0, 1 - y^{(i)}\,\theta^T x^{(i)}\}$$

- Parameter C can be viewed as a way to control overfitting.
  - Trade-off between
    - Having a large margin
    - Classifying correctly (with small cost/loss) the training data-points.

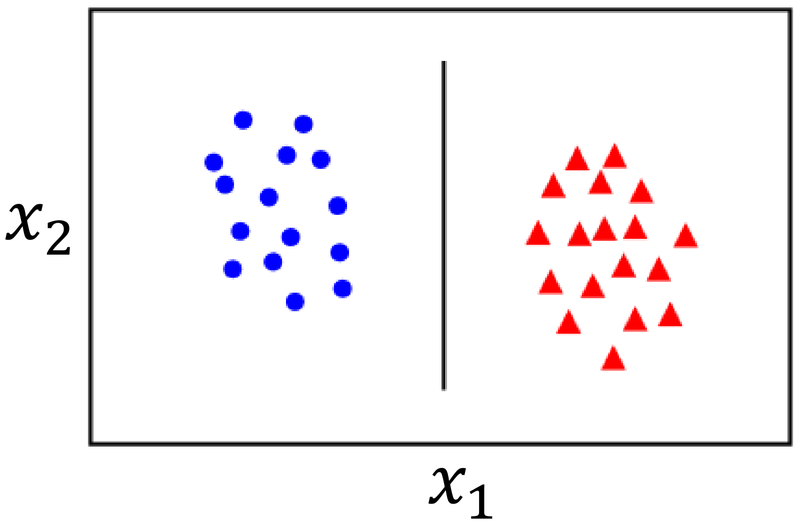


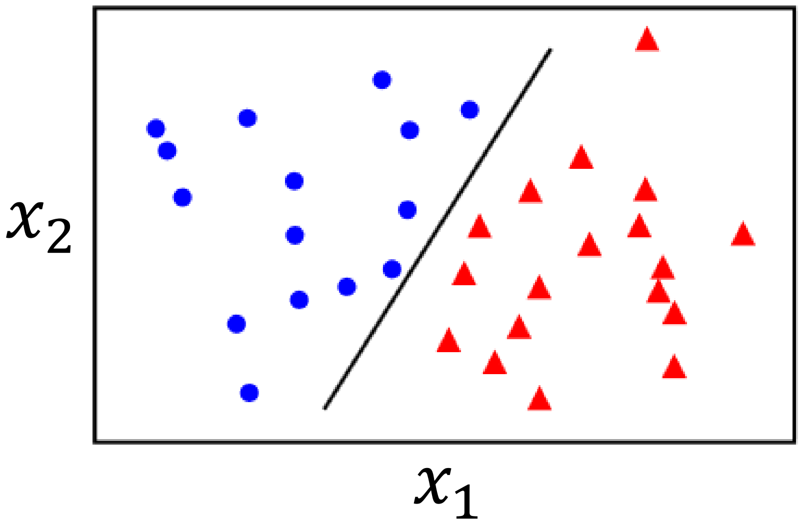
Large C: Lower bias, high variance.
Small C: Higher bias, low variance.

Nonlinear SVM using Kernels

Linear Separability

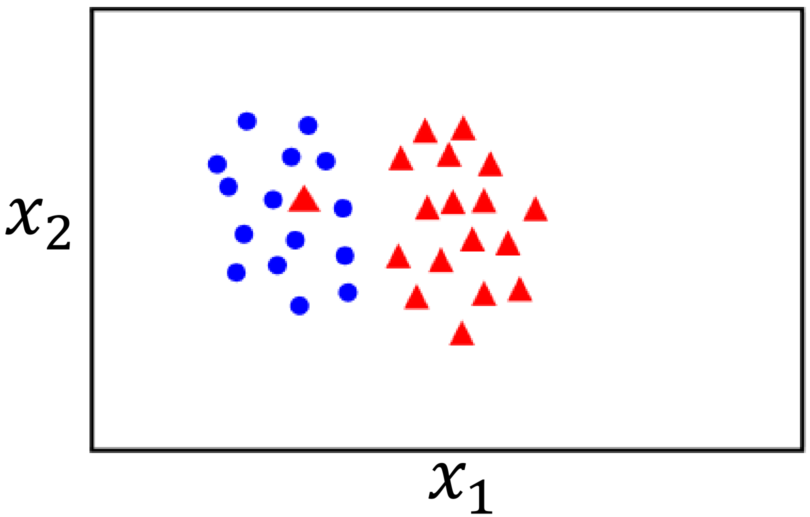
- In these two datasets, the two classes are linearly separable.



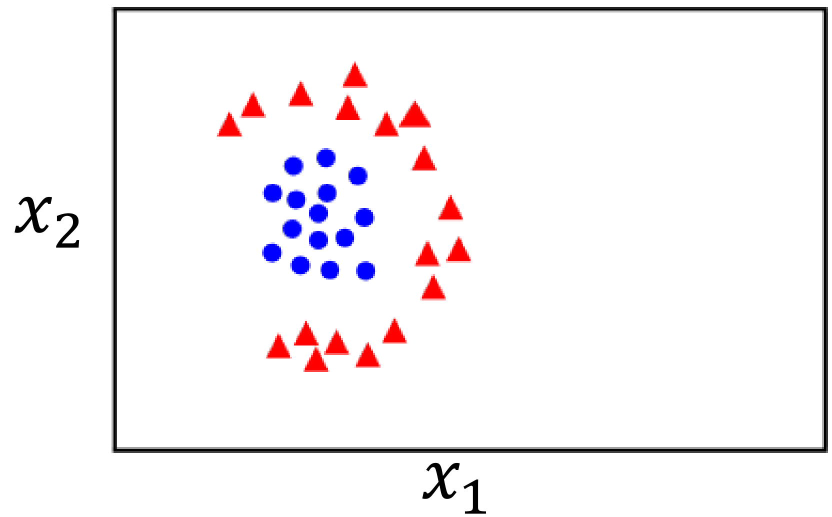


Linear Inseparability

- In both these datasets, the two classes are not linearly separable.



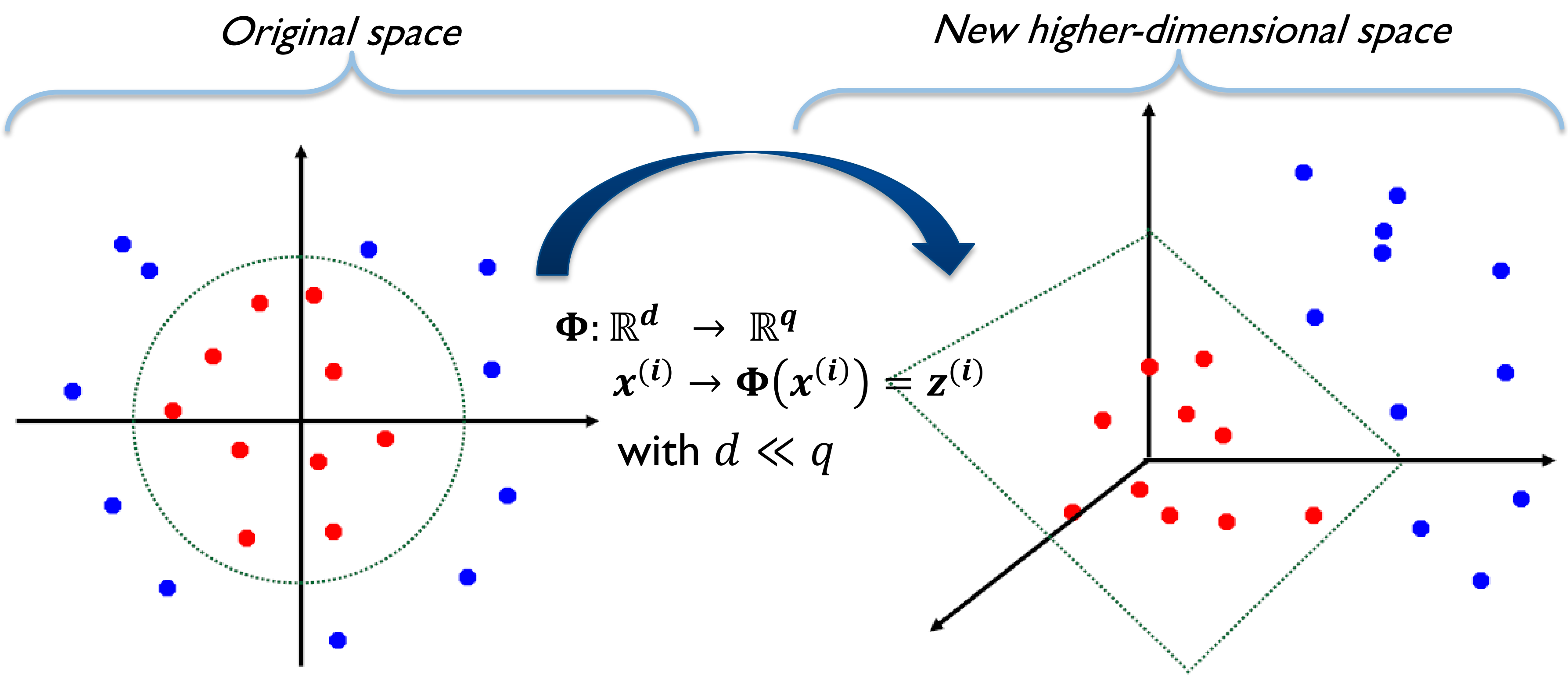
A soft-margin linear SVM trained on this dataset, will probably have a good (low) generalization error.



But trained on this dataset, it will have a high generalization error.

Higher dimensional feature space

- General idea: the original feature space can always be mapped to some new higher-dimensional feature space where the classes are separable.
- Using a so called "*kernel trick*", we can still use SVM without explicitly doing this mapping, i.e. without explicitly computing the new data-points $z^{(i)}$.



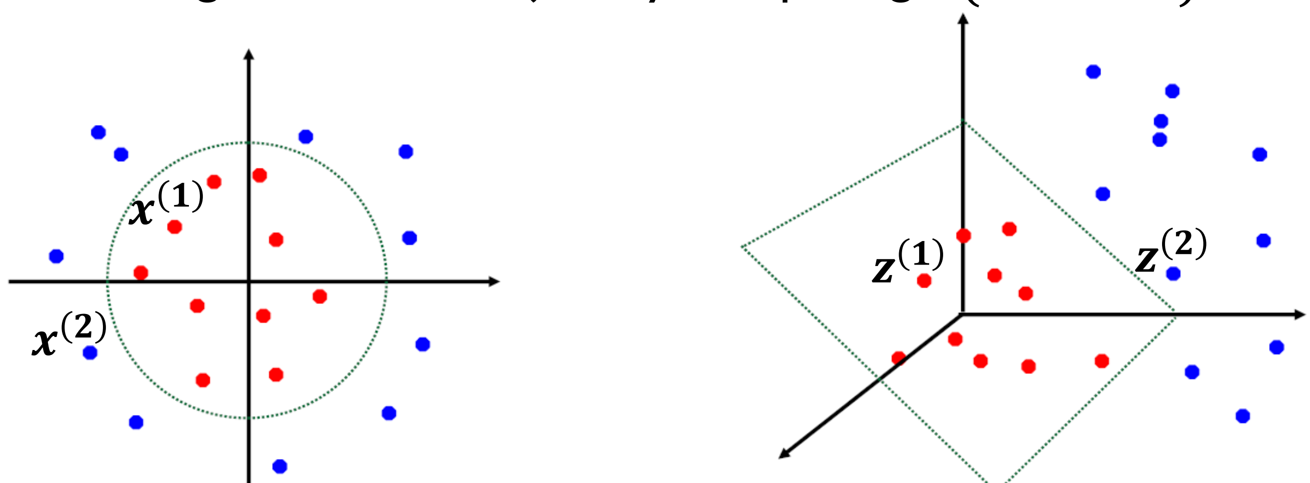
Kernel Function

- A ***kernel function*** $k(x^{(i)}, x^{(j)})$ is some function that corresponds to a ***dot product*** between two vectors $z^{(i)} = \Phi(x^{(i)})$ and $z^{(j)} = \Phi(x^{(j)})$ in some higher-dimensional feature space.

- In other words, a function k of two vectors $x^{(i)}$ and $x^{(j)}$ is a *kernel function*, if it can be written as the dot product between two new vectors (which are transformations of the original vectors) :

$$k(x^{(i)}, x^{(j)}) = \Phi(x^{(i)})^T \Phi(x^{(j)})$$

- This means that we can compute the dot product between e.g. $z^{(1)}$ and $z^{(2)}$, without even knowing $z^{(1)}$ and $z^{(2)}$; just by computing $k(x^{(1)}, x^{(2)})$



Kernel Function

Example:

Let $x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \end{bmatrix}$, $x^{(j)} = \begin{bmatrix} x_1^{(j)} \\ x_2^{(j)} \end{bmatrix}$, $k\left(x^{(i)}, x^{(j)}\right) = \left(1 + {x^{(i)}}^T x^{(j)}\right)^2$

- Is this a kernel function ?
- We need to show that $k\left(x^{(i)}, x^{(j)}\right) = \Phi\left(x^{(i)}\right)^T \Phi\left(x^{(j)}\right)$

Kernel Function

Example:

Let $x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \end{bmatrix}$, $\quad x^{(j)} = \begin{bmatrix} x_1^{(j)} \\ x_2^{(j)} \end{bmatrix}$, $\quad k\left(x^{(i)}, x^{(j)}\right) = \left(1 + {x^{(i)}}^T x^{(j)}\right)^2$

$$k\left(x^{(i)}, x^{(j)}\right) = 1 + {x_1^{(i)}}^2 {x_1^{(j)}}^2 + 2x_1^{(i)} x_1^{(j)} x_2^{(i)} x_2^{(j)} + {x_2^{(i)}}^2 {x_2^{(j)}}^2 + 2x_1^{(i)} x_1^{(j)} + 2x_2^{(i)} x_2^{(j)}$$

$$= \underbrace{\begin{bmatrix} 1 & {x_1^{(i)}}^2 & \sqrt{2}x_1^{(i)} x_2^{(i)} & {x_2^{(i)}}^2 & \sqrt{2}x_1^{(i)} & \sqrt{2}x_2^{(i)} \end{bmatrix}}_{\boldsymbol{\Phi}\left(\boldsymbol{x}^{(i)}\right)^{\boldsymbol{T}}} \underbrace{\begin{bmatrix} 1 \\ {x_1^{(j)}}^2 \\ \sqrt{2}x_1^{(j)} x_2^{(j)} \\ {x_2^{(j)}}^2 \\ \sqrt{2}x_1^{(j)} \\ \sqrt{2}x_2^{(j)} \end{bmatrix}}_{\boldsymbol{\Phi}\left(\boldsymbol{x}^{(j)}\right)}$$

So, yes, this is a kernel function.

Kernel Function

- Examples of kernel functions
  - Linear : $k\left(x^{(i)}, x^{(j)}\right) = {x^{(i)}}^T x^{(j)}$
  - Polynomial of power p : $k\left(x^{(i)}, x^{(j)}\right) = \left(1 + {x^{(i)}}^T x^{(j)}\right)^p$
  - Gaussian (radial-basis function network) :

$$k\left(x^{(i)}, x^{(j)}\right) = e^{-\frac{\left\|x^{(i)} - x^{(j)}\right\|^2}{2\sigma^2}}$$

The Kernel Trick (SVM)

- E.g. remember the hypothesis function of the original simplified SVM:

$$h_\theta(x) = \theta^T x = \theta_0 + \sum_{i=1}^{n} \alpha_i \, y^{(i)} \boldsymbol{x^T x^{(i)}}$$

  - It involves a dot product between the test data-point x and the support vectors $x^{(i)^T}$

- Instead of explicitly mapping the data to a higher dimensional space, we can just use a kernel function, and the hypothesis function would have the same form:

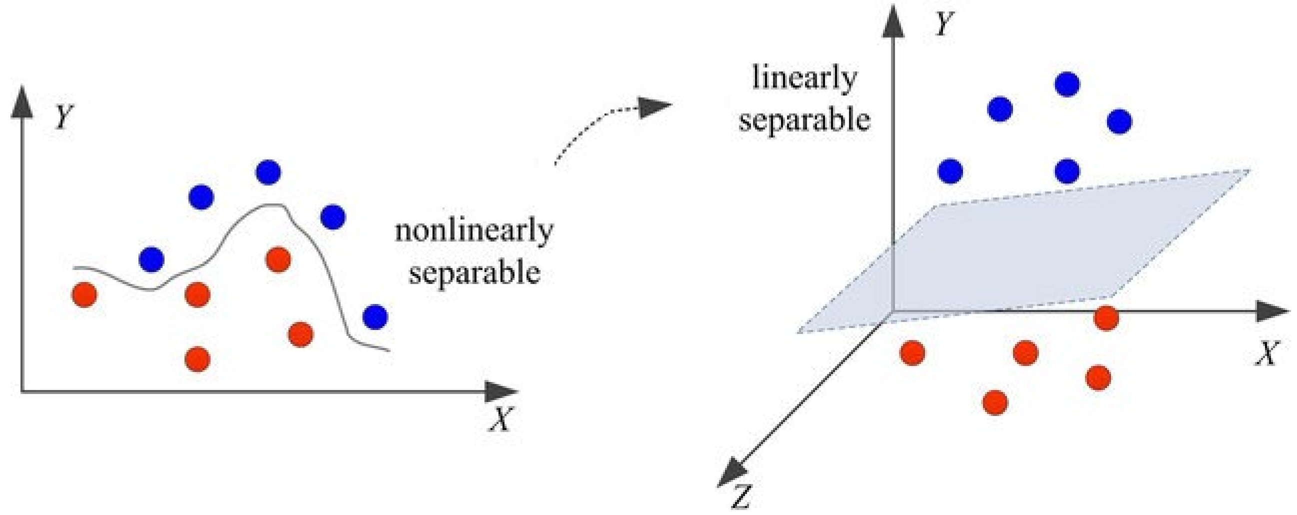
$$h_\theta(x) = \theta^T x = \theta_0 + \sum_{i=1}^{n} \alpha_i \, y^{(i)} \underbrace{\boldsymbol{k(x^T x^{(i)})}}_{z^T z^{(i)}}$$

Because since k is a kernel function, we know that $k(x, x^{(i)}) = \underbrace{\Phi(x)^T}_{z^T} \underbrace{\Phi(x^{(i)})}_{z^{(i)}}$

So we can use the dot product between the higher dimensional vectors, without explicitly knowing them (i.e. a trick).

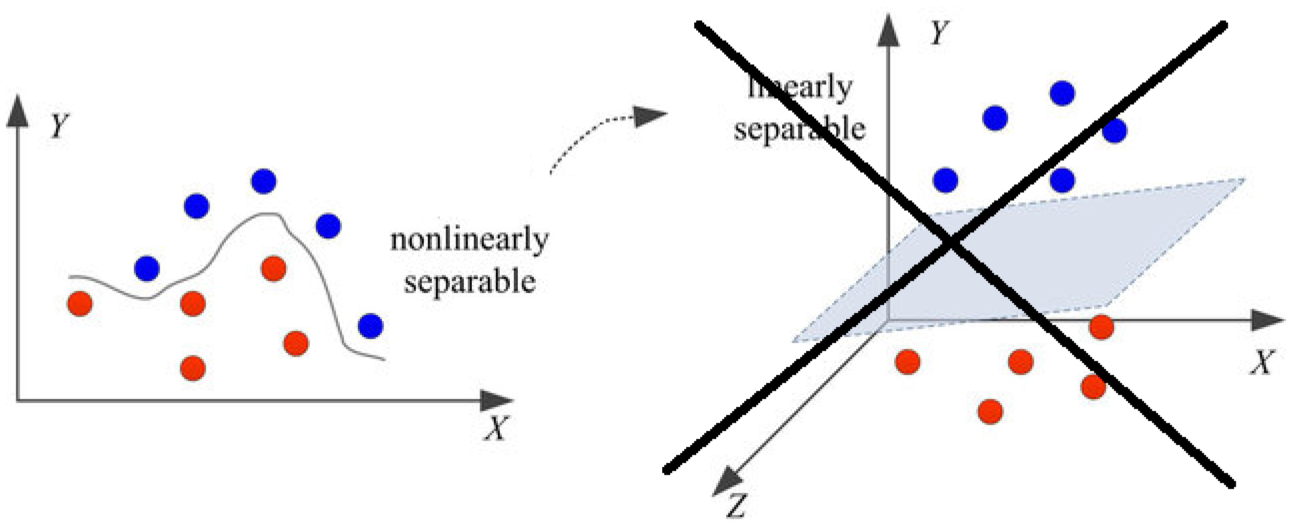
Nonlinear SVM (with kernel trick)

- SVM locates a separating hyperplane in the feature space and classifies 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.



Nonlinear SVM (with kernel trick)

- SVM locates a separating hyperplane in the feature space and classifies 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.



Without explicitly mapping the data to this higher dimensional space. Just using the kernel trick.

Some properties for SVM

- Sparseness of solution when dealing with large data sets
  - only support vectors are used to specify the separating hyper-plane
- Ability to handle large feature spaces
  - Complexity does not depend a lot on the dimensionality of the feature space.
- Overfitting can be controlled by soft margin approach (using the C regularization parameter)
- Nice math property:
  - a simple convex optimization problem which is guaranteed to converge to a single global solution.