# Support Vector Machine

Nakul Gopalan
Georgia Tech

These slides are based on slides from Andrew Zisserman, Yaser S. Abu-Mostafa, Mahdi Roozbahani
.

# Outline

- Precursor: Linear Classifier and Perceptron ⬅
- Support Vector Machine
- Parameter Learning

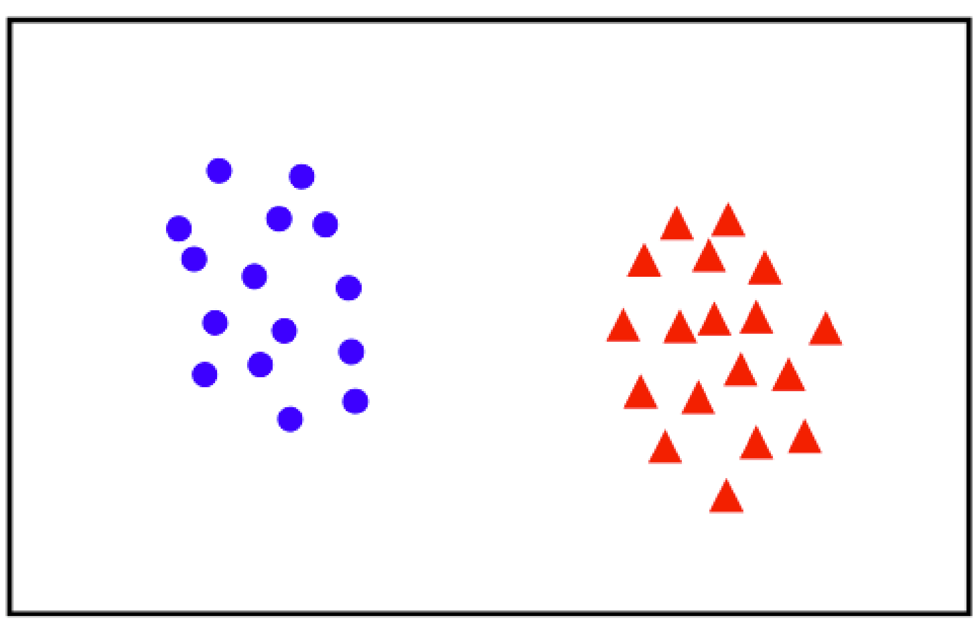
# Binary Classification

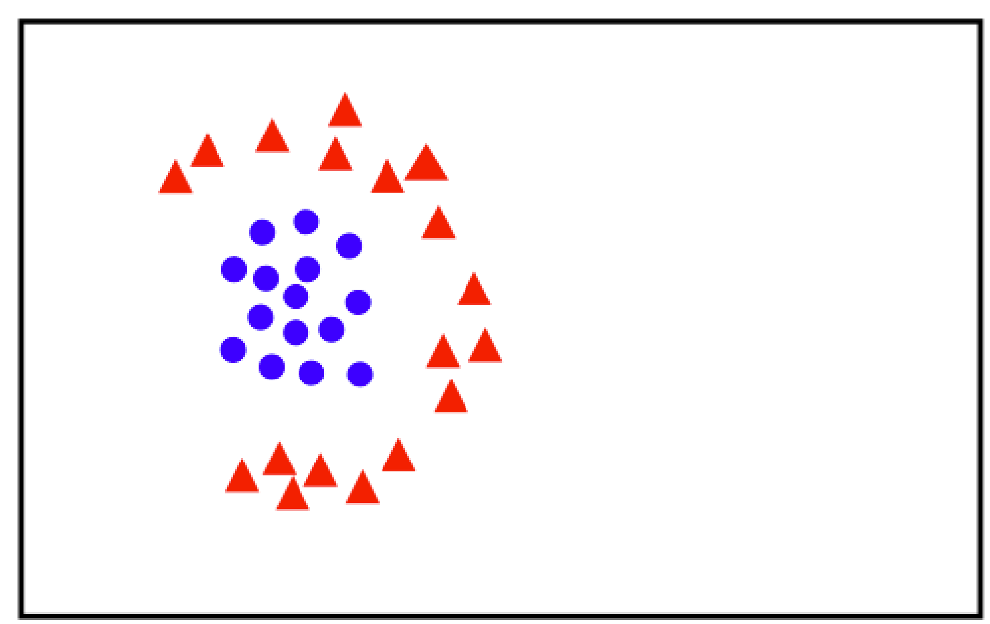
Given training data $(\mathbf{x}_i, y_i)$ for $i = 1 \ldots N$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, learn a classifier $f(\mathbf{x})$ such that

$$f(\mathbf{x}_i) \begin{cases} \geq 0 & +1 \\ < 0 & -1 \end{cases}$$

+ +
- -

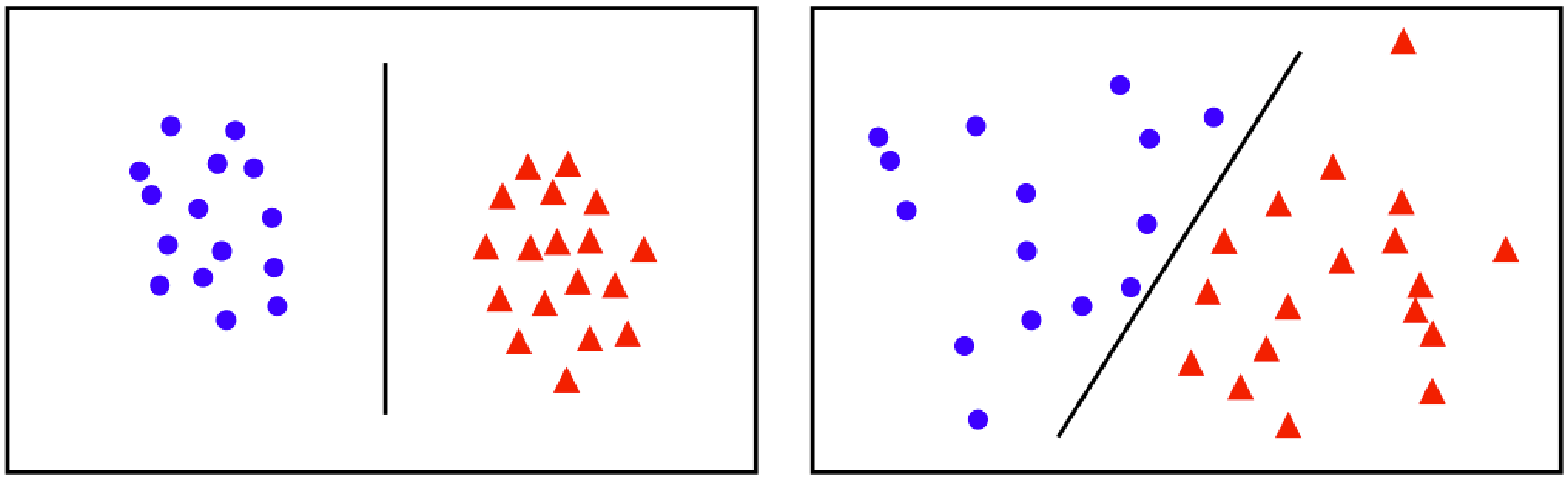
i.e. $y_i f(\mathbf{x}_i) > 0$ for a correct classification.



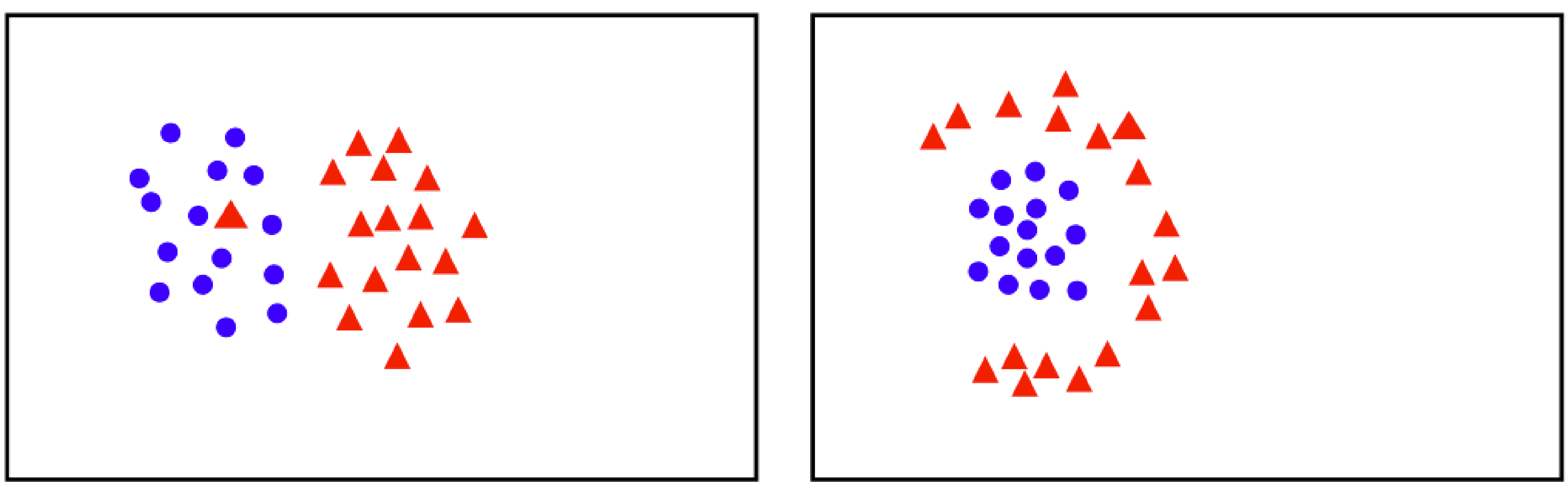


# Linear Separability

linearly
separable



not
linearly
separable

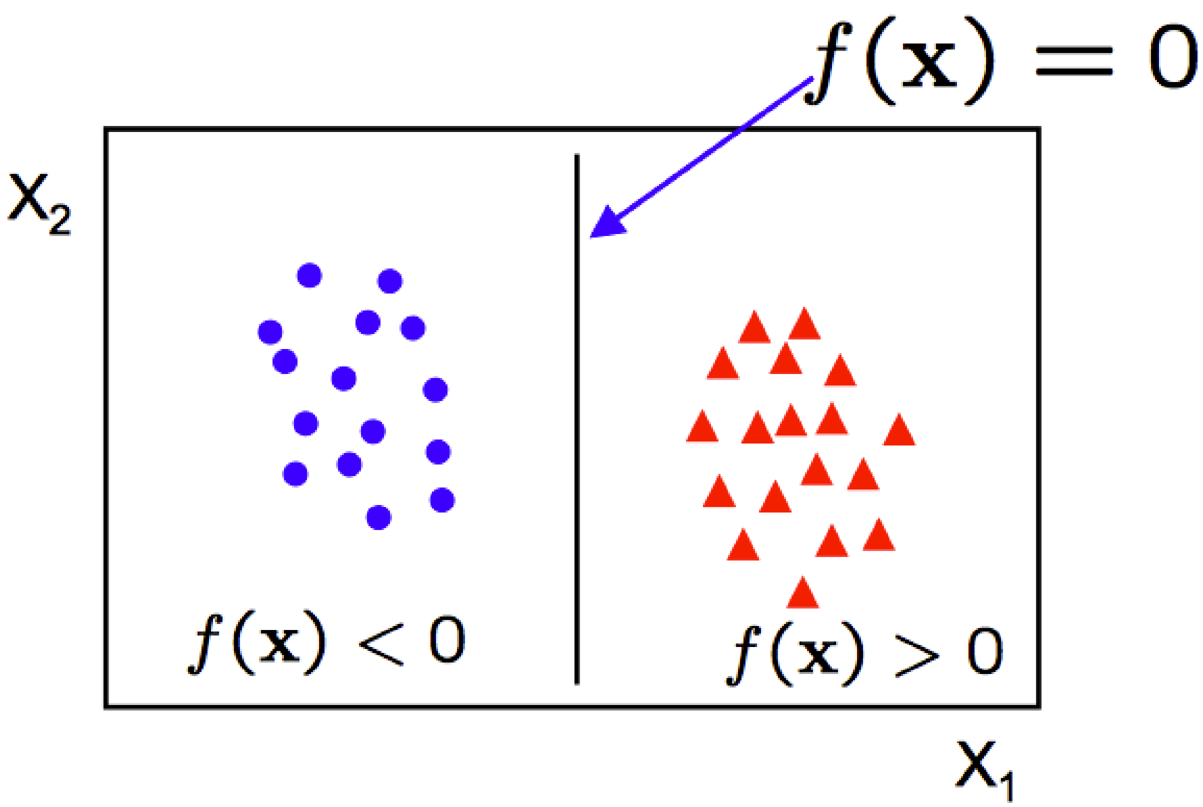


# Linear Classifier

A linear classifier has the form

$$f(x) = x\theta + \theta_0$$

$$\theta_0 \; / \; c \; / \; b$$

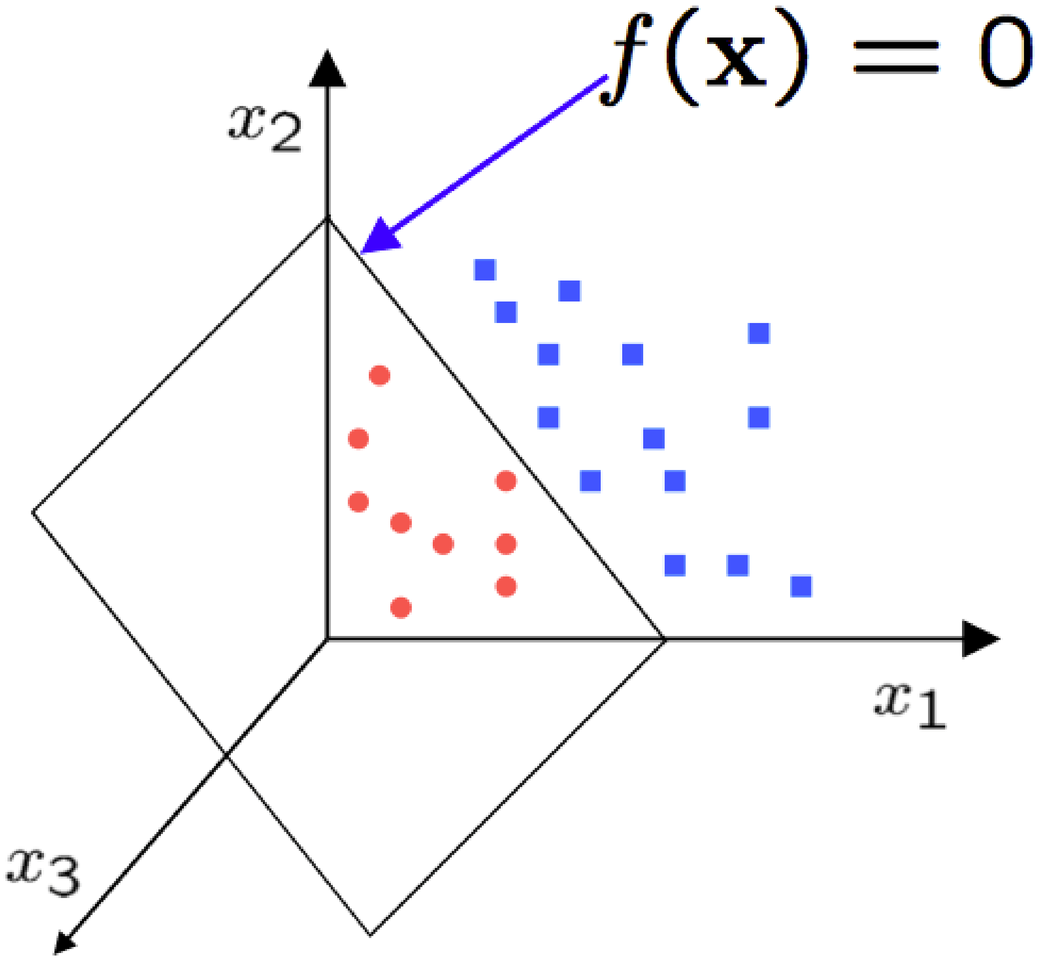



- in 2D the discriminant is a line
- $\theta$ is the normal to the line, and $\theta_0$ the bias
- $\theta$ is known as the weight vector

# Linear Classifier (higher dimension)

A linear classifier has the form

$$f(x) = x\theta + \theta_0$$




- in 3D the discriminant is a plane, and in nD it is a hyperplane

# Perceptron Algorithm

Input: A sequence of training examples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots$
where all $x_i \in \Re^n$, $y_i \in \{-1,1\}$

- Initialize $\mathbf{w}_0 = 0 \in \Re^n$
- For each training example $(\mathbf{x}_i, y_i)$:
  - Predict $y' = \text{sgn}(\mathbf{w}_t^T \mathbf{x}_i)$
  - If $y_i \neq y'$:
    - Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + r\,(y_i\,\mathbf{x}_i)$
- Return final weight vector

Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + r\,\mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - r\,\mathbf{x}_i$

r is the learning rate, a small positive number less than 1

Update only on error. A mistake-driven algorithm

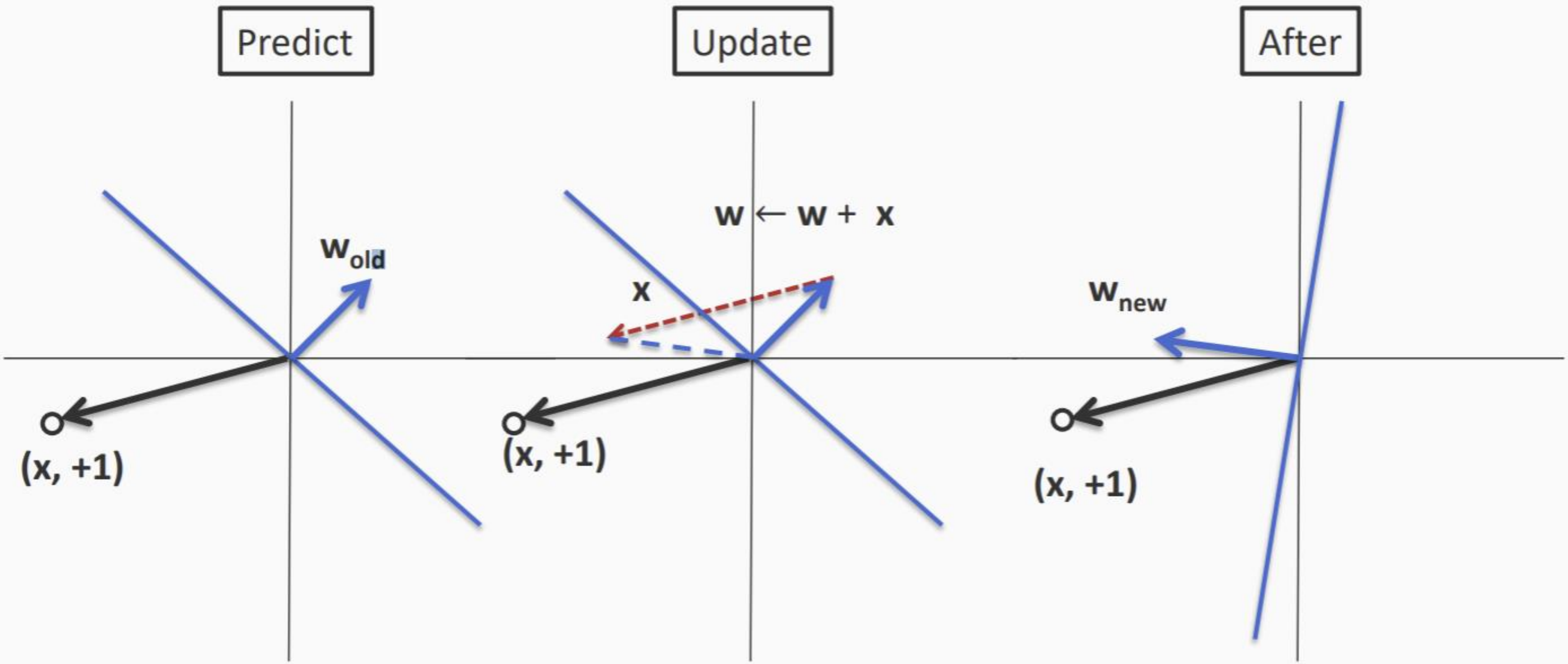
This is the simplest version. We will see more robust versions at the end

Mistake can be written as $y_i \mathbf{w}_t^T \mathbf{x}_i \leq 0$

These slides are from Vivek Srikumar

Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + r\,\mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - r\,\mathbf{x}_i$

# Geometry of the perceptron update

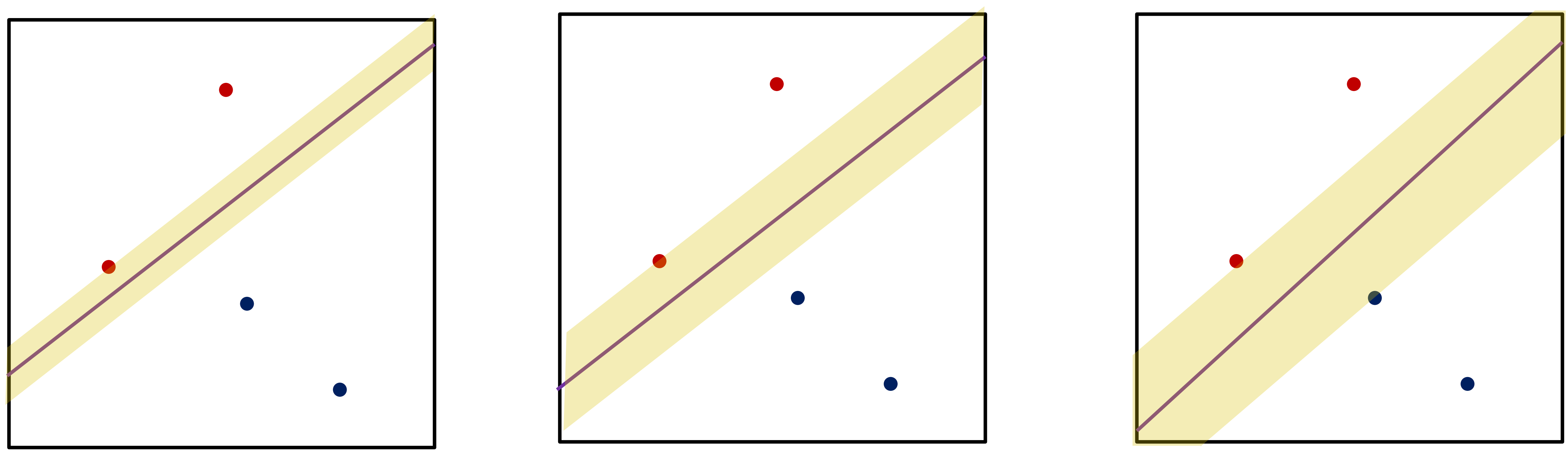



For a mistake on a positive example

These slides are from Vivek Srikumar

# Linear separation

We can have different separating lines



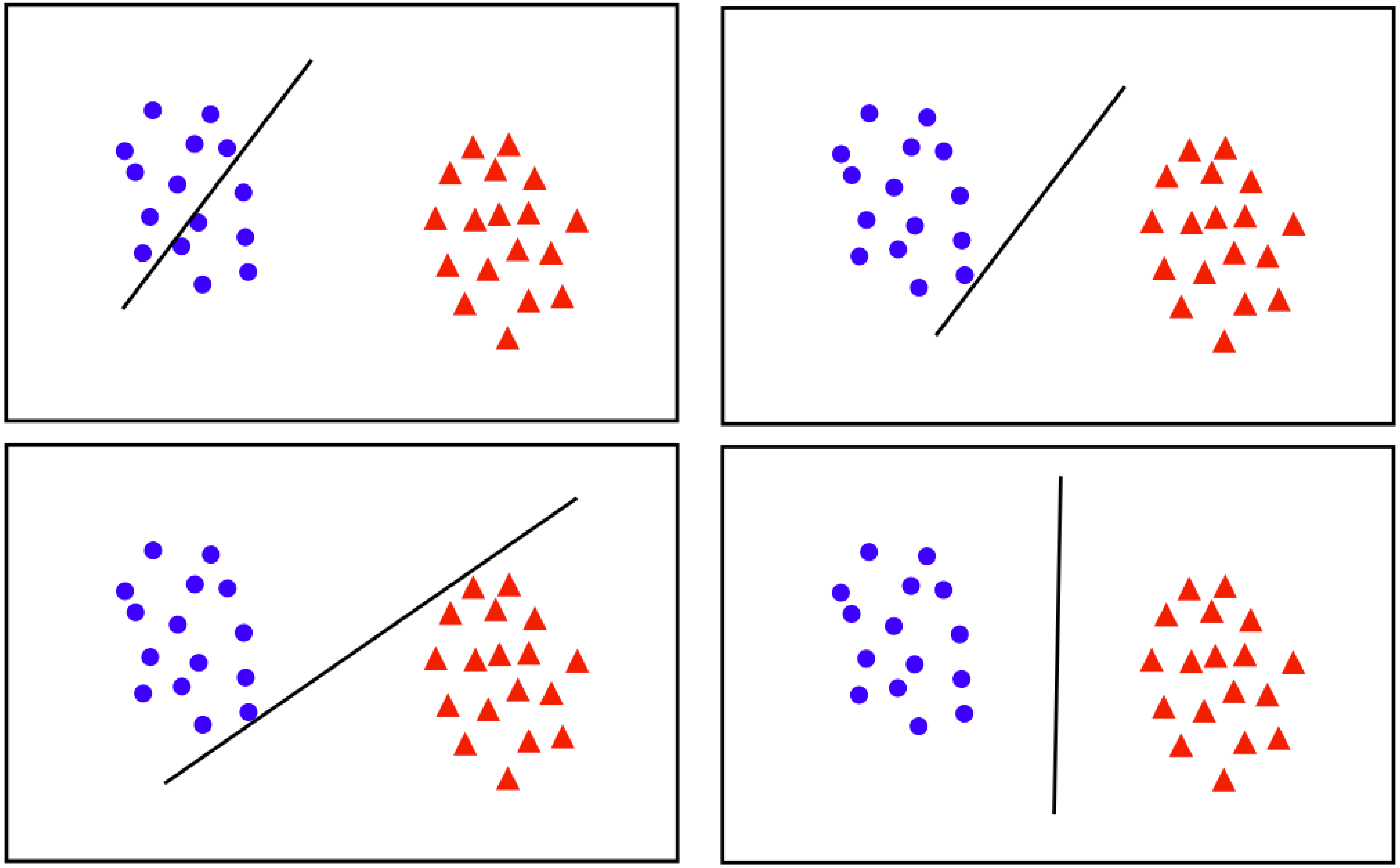
Which line is the best?

Why is the bigger margin better?

What $\boldsymbol{\theta}$ maximizes the margin?

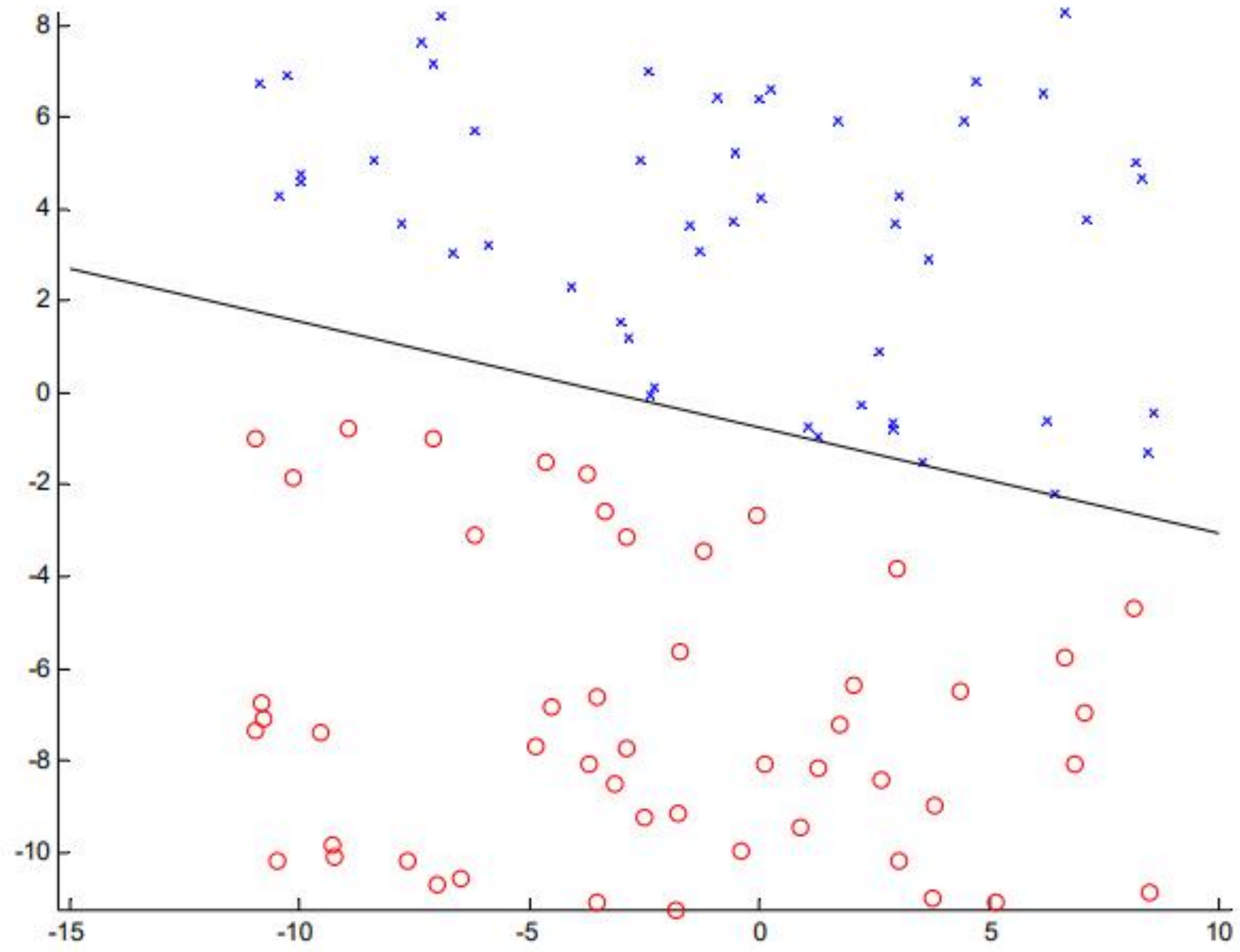
All cases, error is zero and they are linear, so they are all good for generalization.

# What is the Best $\theta$?



- maximum margin solution: most stable under perturbations of the inputs

## Perceptron example



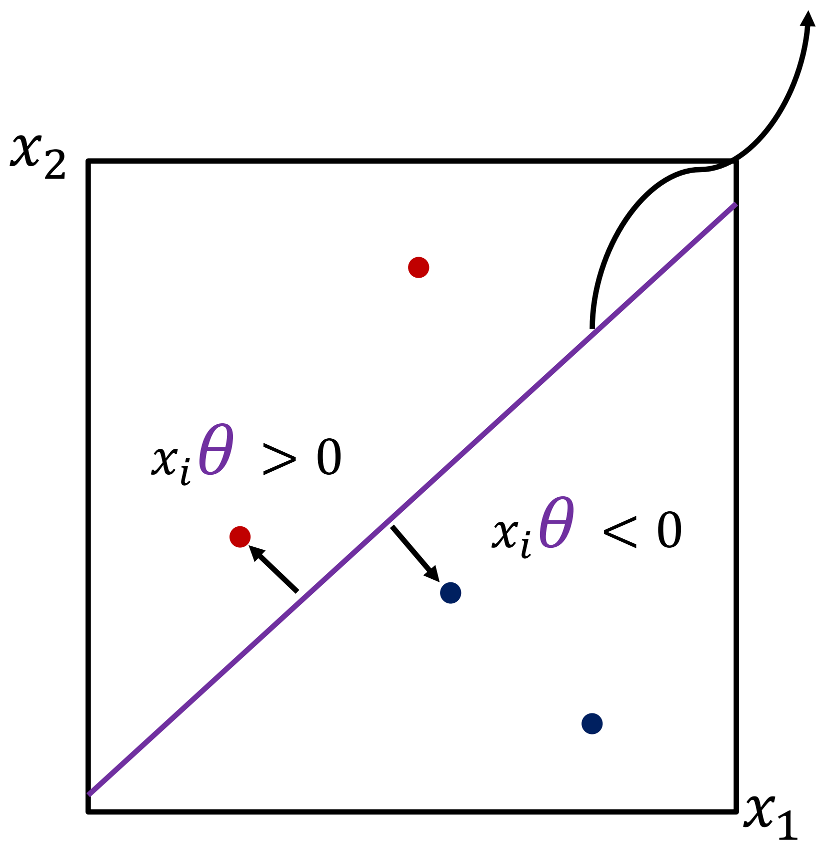

- if the data is linearly separable, then the algorithm will converge
- convergence can be slow …
- separating line close to training data
- we would prefer a larger margin for generalization (better generalization)

# Outline

- Precursor: Linear Classifier and Perceptron
- Support Vector Machine ⬅
- Parameter Learning ⬅

# Finding $\theta$ with a **fat** margin

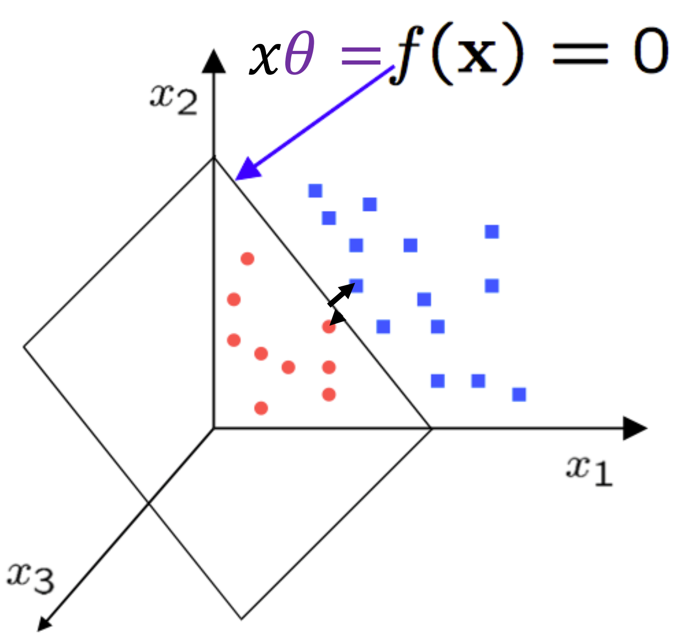
Solution (decision boundary) of the line: $x\theta = 0$




Let $\boldsymbol{x_i}$ to be the nearest data point to the line (plane):

$$|x_i\theta| > 0$$

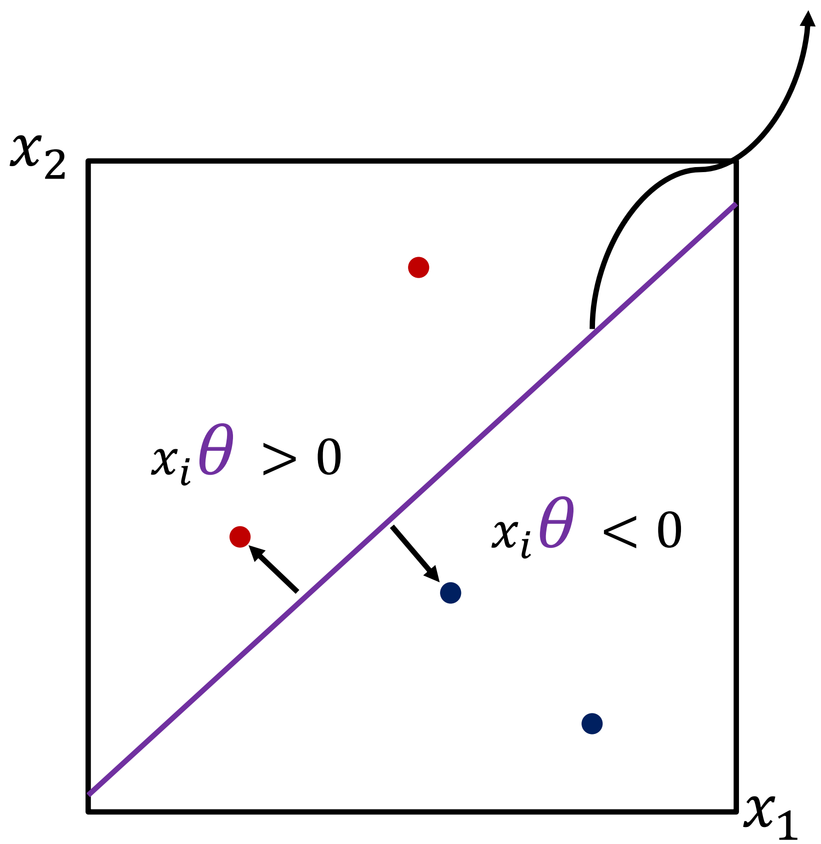
Our line solution is $x\theta = 0$

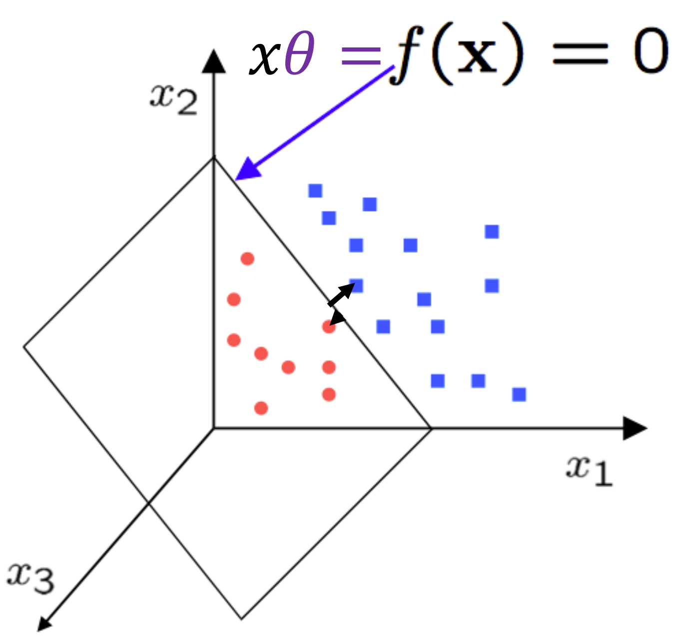



Does it matter if $\theta$ is scaled up or down for the decision boundary?

# Finding $\theta$ with a **fat** margin

Solution (decision boundary) of the line: $x\theta = 0$




Let $x_i$ to be the nearest data point to the line (plane):

$$|x_i\theta| > 0$$

Our line solution is $x\theta = 0$

Does it matter if $\theta$ is scaled up or down for the decision boundary?

$|x_i\theta| = 1$ → normalization

Let's pull out $\theta_0$ from $\theta = (\theta_1, \ldots, \theta_d)$ and call it be $b$

Decision boundary would be: $x\theta + b = 0$

# Computing the distance

The distance between $\boldsymbol{x_i}$ and the line $x\theta + b = 0$ where $|x_i\theta + b| = 1$

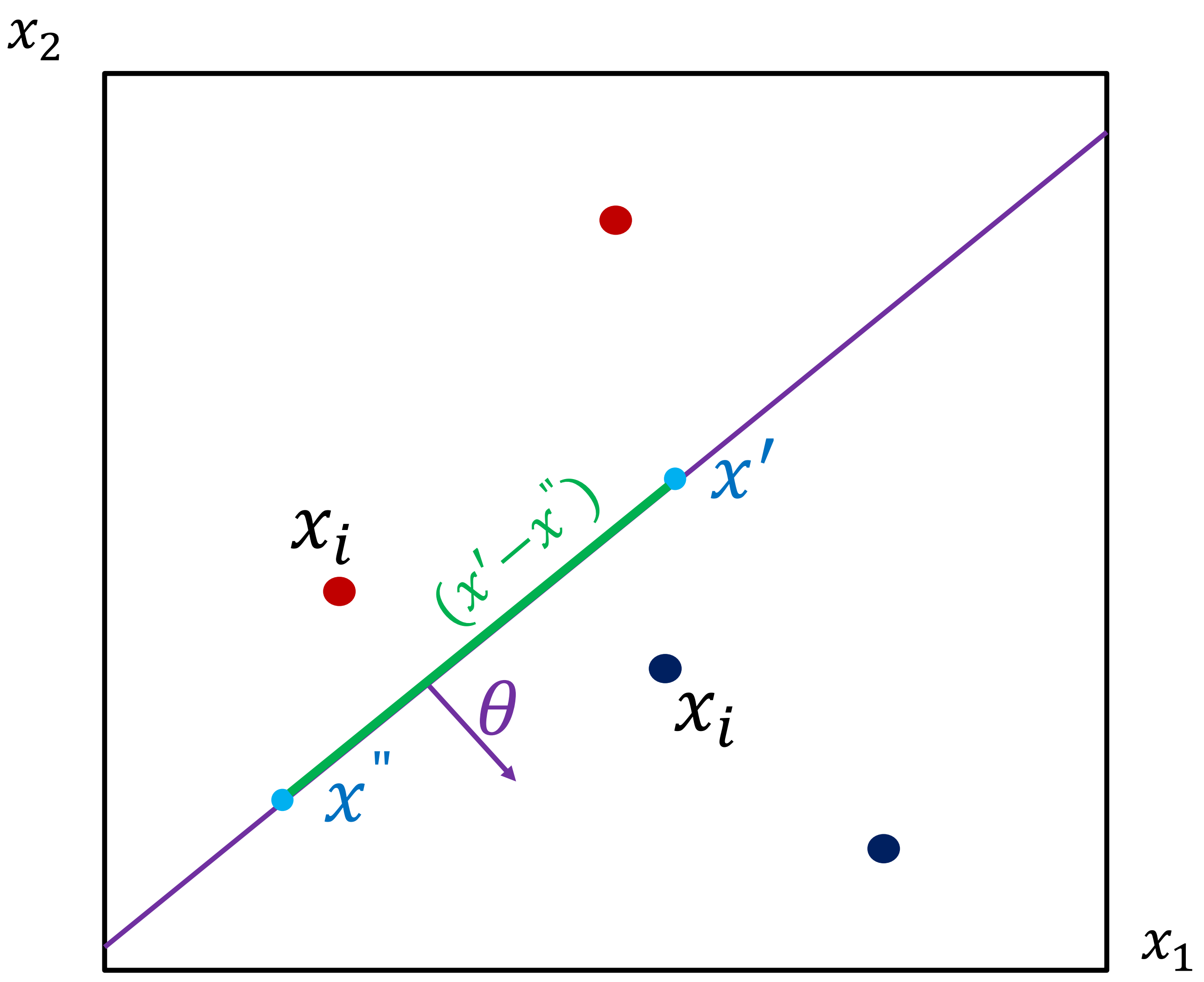
The vector $\theta$ is perpendicular to the decision line.

Consider $x'$ and $x''$ on the plane

$x'\theta + b = 0$ and $x''\theta + b = 0$

$$x'\theta + b = x''\theta + b$$

$$\Rightarrow (x' - x'')\theta = 0$$



# What is the distance of my fat margin?

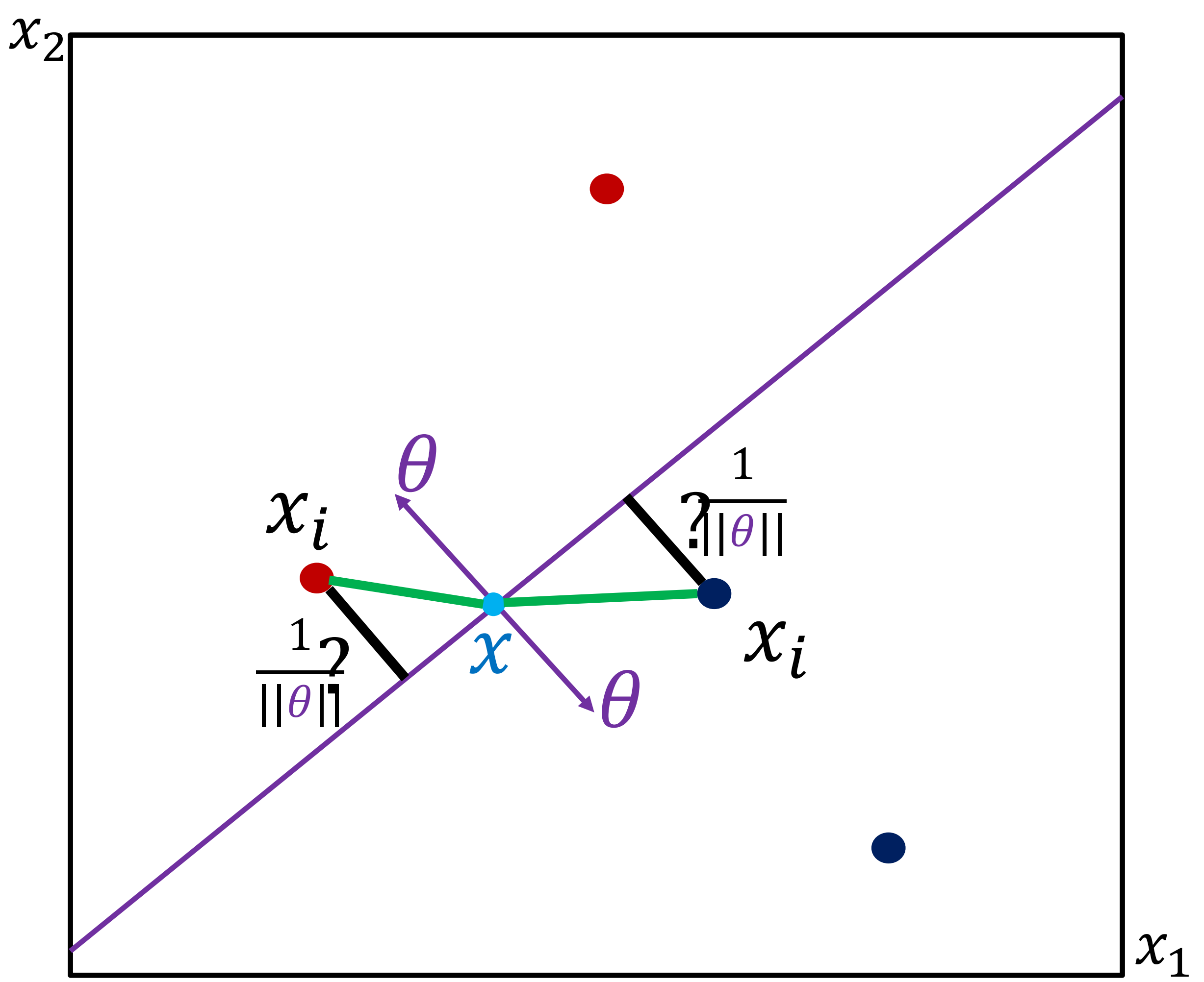
What is the distance between $x_i$ and the plane?

Let's take any point $x$ on the line:

Distance would be projection of $(x_i - x)$ vector on $\theta$.

To project the vector, we need to normalize $\theta$ to get the unit vector.

$\hat{\theta} = \frac{\theta}{||\theta||} \Rightarrow \text{distance} = \left| (x_i - x)\hat{\theta} \right|$ which is the dot product



# What is the distance of my fat margin?

What is the distance between $x_i$ and the plane?

Let's take any point $x$ on the line:

Distance would be projection of $(x_i - x)$ vector on $\theta$.

To project the vector, we need to normalize $\theta$ to get the unit vector.

$$\hat{\theta} = \frac{\theta}{||\theta||} \Rightarrow \text{distance} = \left| (x_i - x)\hat{\theta} \right|$$ which is the dot product

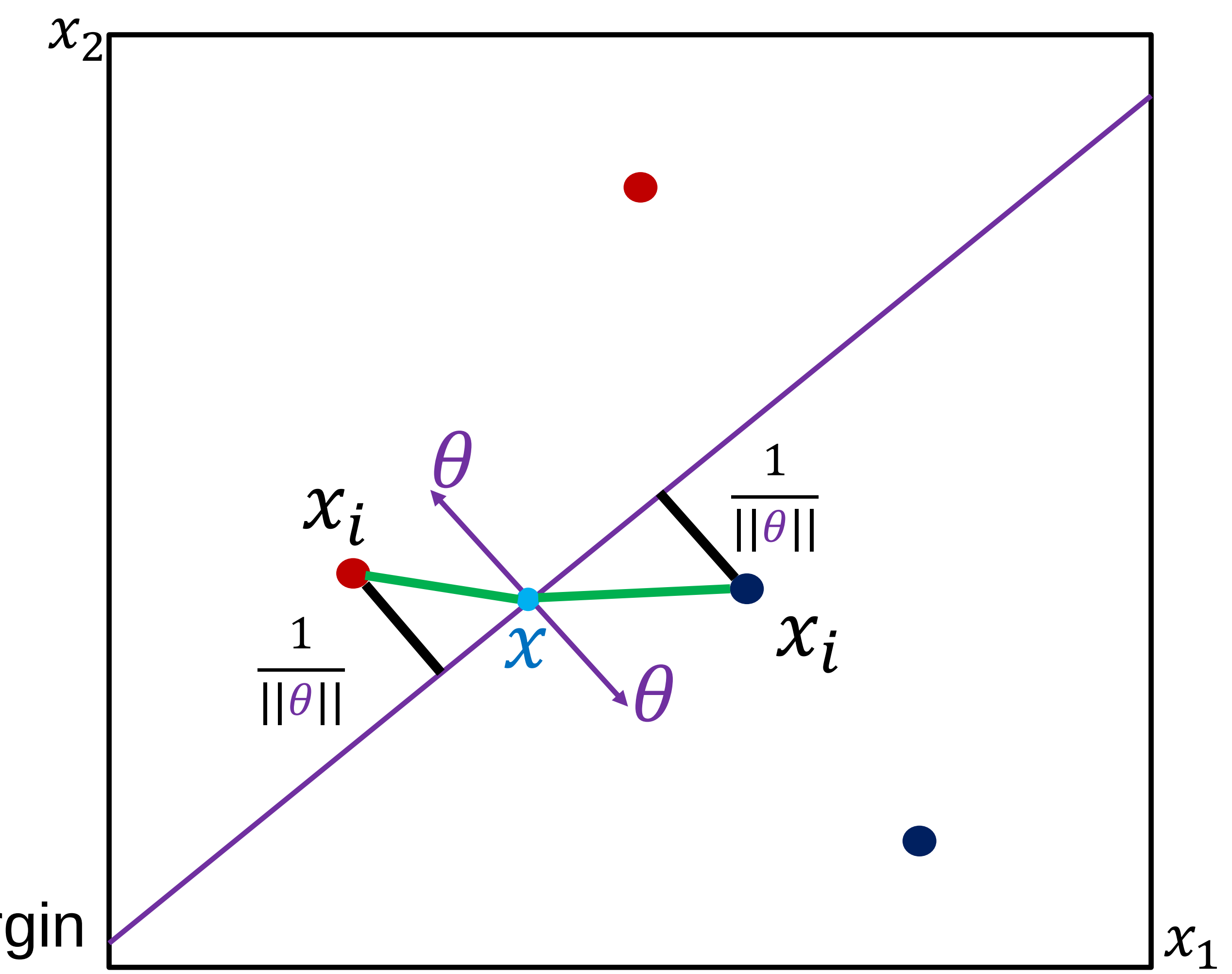
$$\text{distance} = \frac{1}{||\theta||} |(x_i\theta - x\theta)|$$

$$= \frac{1}{||\theta||} |(\underbrace{x_i\theta + b}_{\text{My constraint}} \underbrace{- x\theta - b}_{\text{A point on the decision line}})| = \frac{1}{||\theta||}$$

My constraint $|x_i\theta + b| = 1$

A point on the decision line $x\theta + b = 0$

The margin



# Now we need to maximize the margin

Maximize $\frac{1}{||\theta||}$

Subject to Min value of $|x_i\theta + b| = 1 \Rightarrow nearest\ neighbour$

$i = 1,2, \dots, N$

There is a "min" in our constraining; it can be hard to optimize this problem(non-convex form)

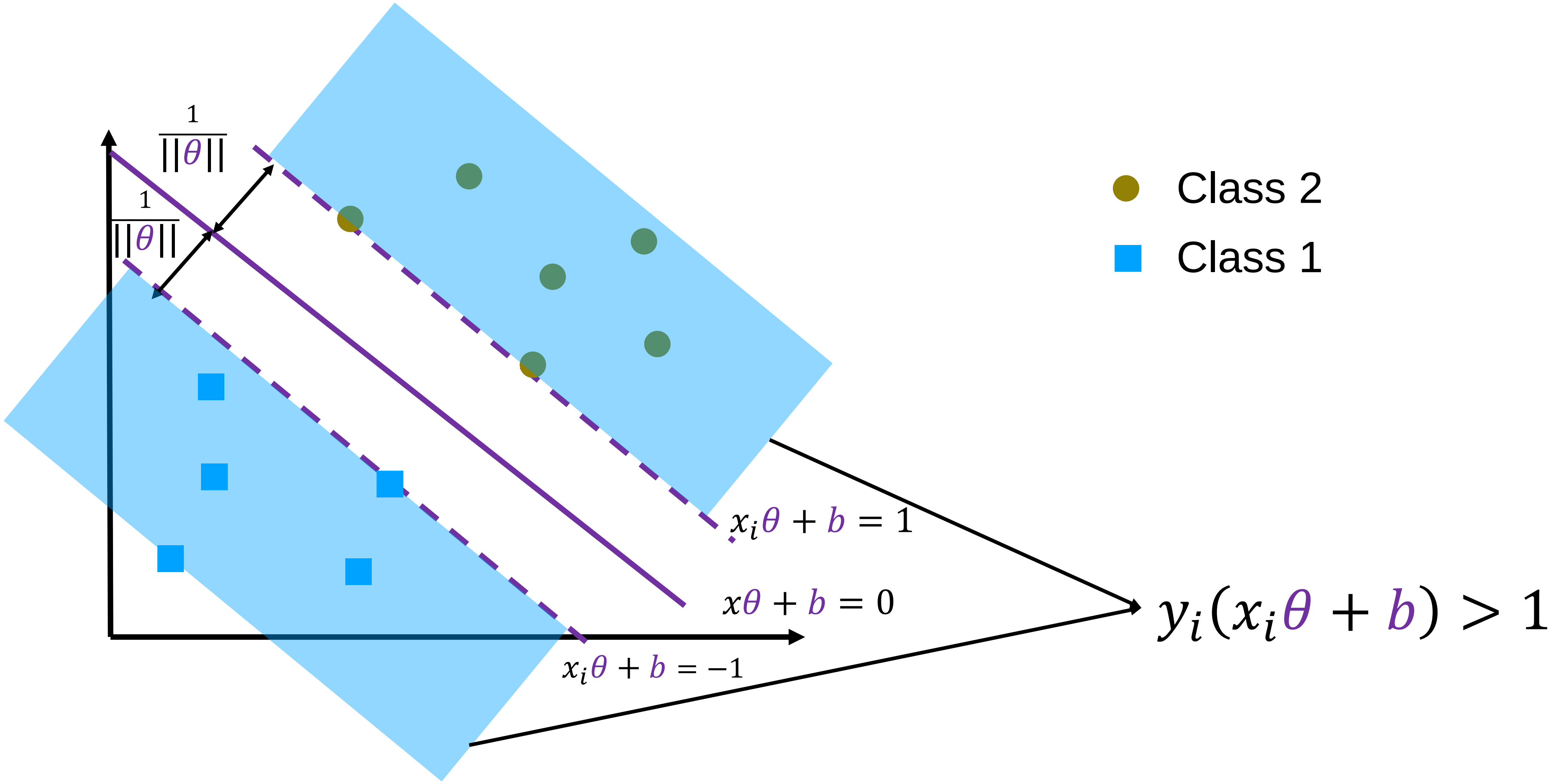
Can I write the following term to get rid of absolute value?

$$|x_i\theta + b| = y_i(x_i\theta + b) \Rightarrow \text{for a correct classification}$$

If min $|x_i\theta + b| = 1 \Rightarrow so\ it\ can\ be\ at\ least\ 1$

Maximize $\frac{1}{||\theta||}$

Subject to $y_i(x_i\theta + b) \geq 1$ for $i = 1,2, \dots, N$




Maximize $\frac{1}{||\theta||}$

Subject to $y_i(x_i\theta + b) \geq 1$ for $i = 1,2,\ldots,N$

Minimize $\frac{1}{2}\theta\theta^T$

Subject to $y_i(x_i\theta + b) \geq 1$ for $i = 1,2,\ldots,N$

# Constrained optimization

Minimize $\frac{1}{2}\theta\theta^T$

Subject to $y_i(x_i\theta + b) \geq 1$ for $i = 1,2, \ldots, N$

$\theta \in \mathbb{R}^d, b \in \mathbb{R}$

Using Lagrange method: But wait, there is an **inequality** in our constraints

We use **Karush-Kuhn-Tucker (KKT)** condition to deal with this problem

# Constrained optimization

Minimize $\frac{1}{2}\theta\theta^T$

Subject to $y_i(x_i\theta + b) \geq 1$ for $i = 1,2,\ldots,N$

$\theta \in \mathbb{R}^d, b \in \mathbb{R}$

Using Lagrange method: But wait, there is an **inequality** in our constraints

We use **Karush-Kuhn-Tucker (KKT)** condition to deal with this problem

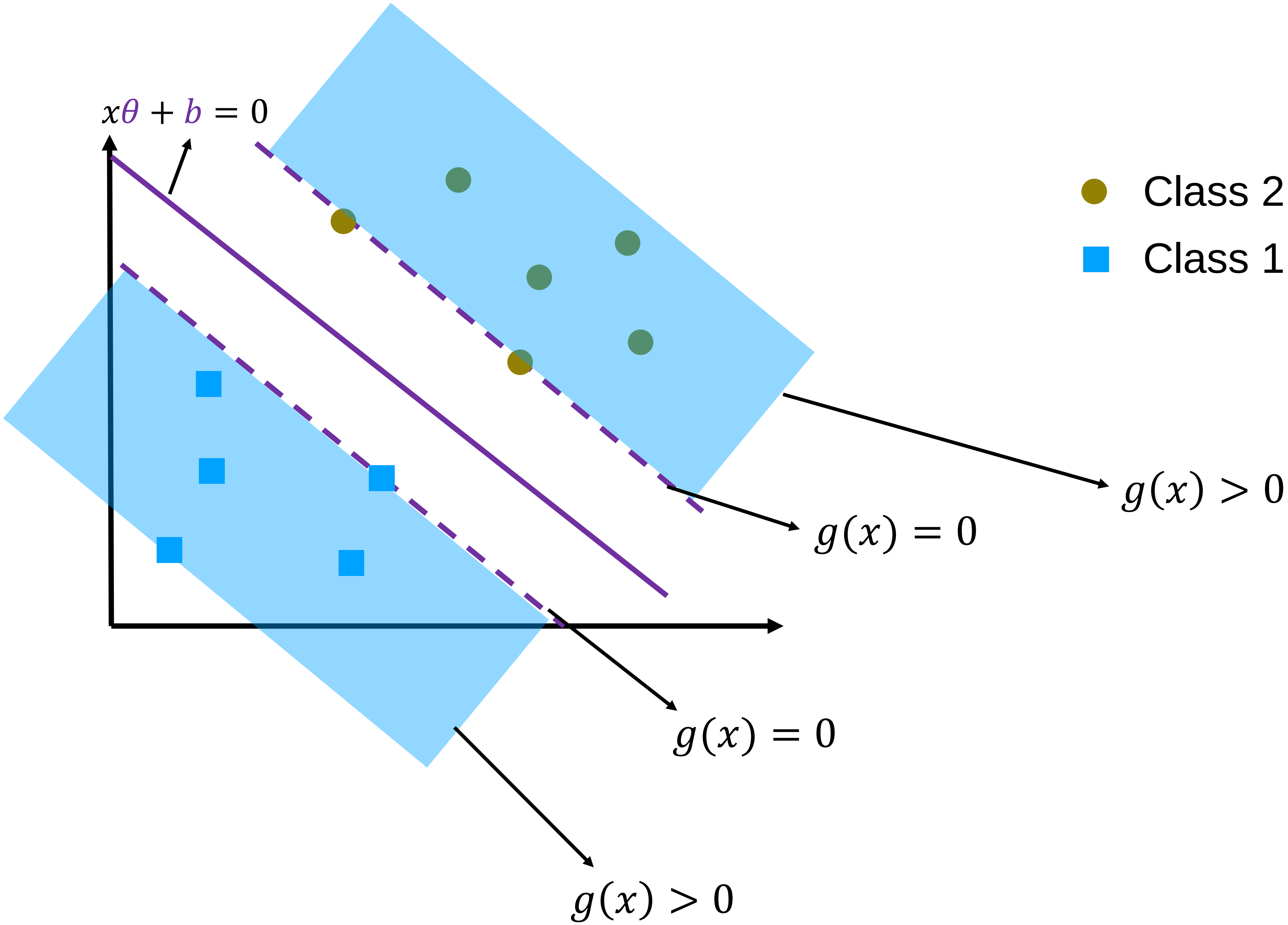
$g(x) = y_i(x_i\theta + b) - 1$ $\quad\quad$ $\alpha = lagrange\ multiplier$

We need to optimize $\theta, b, and\ \alpha$

1) $g(x) \geq 0$ Primal feasibility

2) $\alpha \geq 0$ Dual feasibility

3) $g(x)\alpha = 0$ Complementary slackness $\Rightarrow \begin{cases} g(x) > 0, & \alpha = 0 \\ \alpha > 0, & g(x) = 0 \end{cases}$




$$g(x) = y_i(x_i\theta + b) - 1$$

3) $g(x)\alpha = 0$ Complementary slackness $\Rightarrow \begin{cases} g(x) > 0, & \alpha = 0 \\ \alpha > 0, & g(x) = 0 \end{cases}$

# Lagrange formulation

Minimize $\frac{1}{2}\theta\theta^T$ s.t. $y_i(x_i\theta + b) - 1 \geq 0$

$$\mathcal{L}(\theta, b, \alpha) = \frac{1}{2}\theta\theta^T - \sum_{i=1}^{N} \alpha_i (y_i(x_i\theta + b) - 1)$$

*Minimize w.r.t* $\theta$ and $b$ and maximize *w.r.t* each $\alpha_i \geq 0$

$$\nabla_\theta \mathcal{L}(\theta, b, \alpha) = \theta - \sum_{i=1}^{N} \alpha_i y_i x_i = 0$$

$$\nabla_b \mathcal{L}(\theta, b, \alpha) = -\sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\theta = \sum_{i=1}^{N} \alpha_i y_i x_i$$

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

Let’s substitute these in the Lagrangian:

$$\mathcal{L}(\theta, b, \alpha) = \frac{1}{2}\theta\theta^T - \sum_{i=1}^{N} \alpha_i (y_i(x_i\theta + b) - 1)$$

$$\mathcal{L}(\theta, b, \alpha) = \sum_{i=1}^{N} \alpha_i + \frac{1}{2}\theta\theta^T - \sum_{i=1}^{N} \alpha_i (y_i(x_i\theta + b))$$

$$\mathcal{L}(\theta, b, \alpha) = \sum_{i=1}^{N} \alpha_i + \frac{1}{2}\theta\theta^T - \sum_{i=1}^{N} \alpha_i \left(y_i(x_i\theta)\right) = \sum_{i=1}^{N} \alpha_i + \frac{1}{2}\theta\theta^T - \theta\theta^T =$$

$$= \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\theta\theta^T$$

$$\theta = \sum_{i=1}^{N} \alpha_i y_i x_i \qquad \sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\mathcal{L}(\theta, b, \alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \theta \theta^T$$

$$\mathcal{L}(\theta, b, \alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j x_i x_j^T$$

maximize *w.r.t* each $\alpha_i \geq 0$ for $i = 1, \ldots, N$

and

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

# The solution – quadratic programming

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j x_i x_j^T$$

Quadratic programming packages usually use “min”

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j x_i x_j^T - \sum_{i=1}^{N} \alpha_i$$

$$\min_{\alpha} \frac{1}{2} \alpha^T \begin{bmatrix} y_1 y_1 x_1 x_1^T & y_1 y_2 x_1 x_2^T & \dots & y_1 y_N x_1 x_N^T \\ y_2 y_1 x_2 x_1^T & y_2 y_2 x_2 x_2^T & \dots & y_2 y_N x_2 x_N^T \\ \dots & \dots & \dots & \dots \\ y_N y_1 x_N x_1^T & y_N y_2 x_N x_2^T & \dots & y_N y_N x_N x_N^T \end{bmatrix} \alpha + (-I^T) \alpha$$

$$\min_{\alpha} \frac{1}{2} \alpha^T \begin{bmatrix} y_1 y_1 x_1 x_1^T & y_1 y_1 x_1 x_2^T & \cdots & y_1 y_N x_1 x_N^T \\ y_2 y_1 x_2 x_1^T & y_2 y_2 x_2 x_2^T & \cdots & y_2 y_N x_2 x_N^T \\ \cdots & \cdots & \cdots & \cdots \\ y_N y_1 x_N x_1^T & y_N y_2 x_n x_2^T & \cdots & y_N y_N x_N x_N^T \end{bmatrix} \alpha + (-I^T)\alpha$$

Quadratic coefficients

Linear term

Subject to $\sum_{i=1}^{N} \alpha_i y_i = y^T \alpha = 0$

Linear equality constraint

$$lower\ bound(0) \leq \alpha \leq \text{upper bound}(\infty)$$

Pass these to a quadratic programming package

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - 1^T \alpha \quad \text{subject to} \quad y^T \alpha = 0; \alpha \geq 0$$

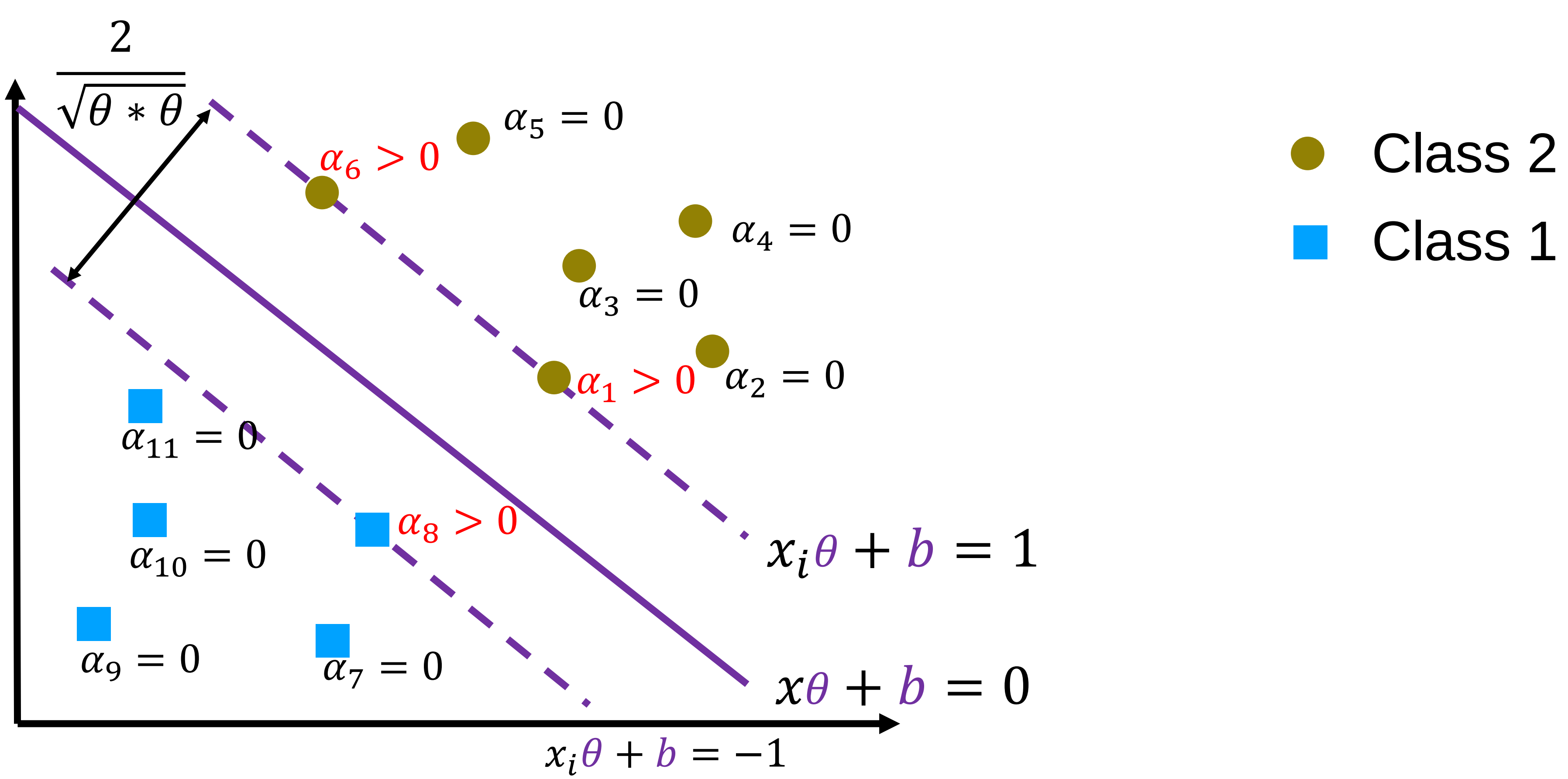
Quadratic programming will give us $\alpha$

Solution: $\alpha = \alpha_1, \dots, \alpha_N$

KKT condition ($\alpha_i g_i(\theta) = 0$): $\alpha_i (y_i (x_i \theta + b) - 1) = 0$

$(y_i (x_i \theta + b) - 1) > 0 \quad \Rightarrow \quad \alpha_i = 0$

$(y_i (x_i \theta + b) - 1) = 0 \quad \Rightarrow \quad \alpha_i > 0 \Rightarrow x_i \; is \; a \; support \; vector$



# Training

$$\theta = \sum_{i=1}^{N} \alpha_i y_i x_i$$

No need to go over all datapoints

$$\rightarrow \theta = \sum_{x_i \, in \, SV} \alpha_i y_i x_i$$

and for $b$ pick any support vector and calculate:

$$y_i(x_i\theta + b) = 1$$

# Testing

For a new test point s

Compute:

$$s\theta + \mathrm{b} = \sum_{x_i \, in \, SV} \alpha_i y_i x_i s^T + b$$

Classify s as class 1 if the result is positive, and class 2 otherwise

# Training

$$\theta = \sum_{i=1}^{N} \alpha_i y_i x_i$$

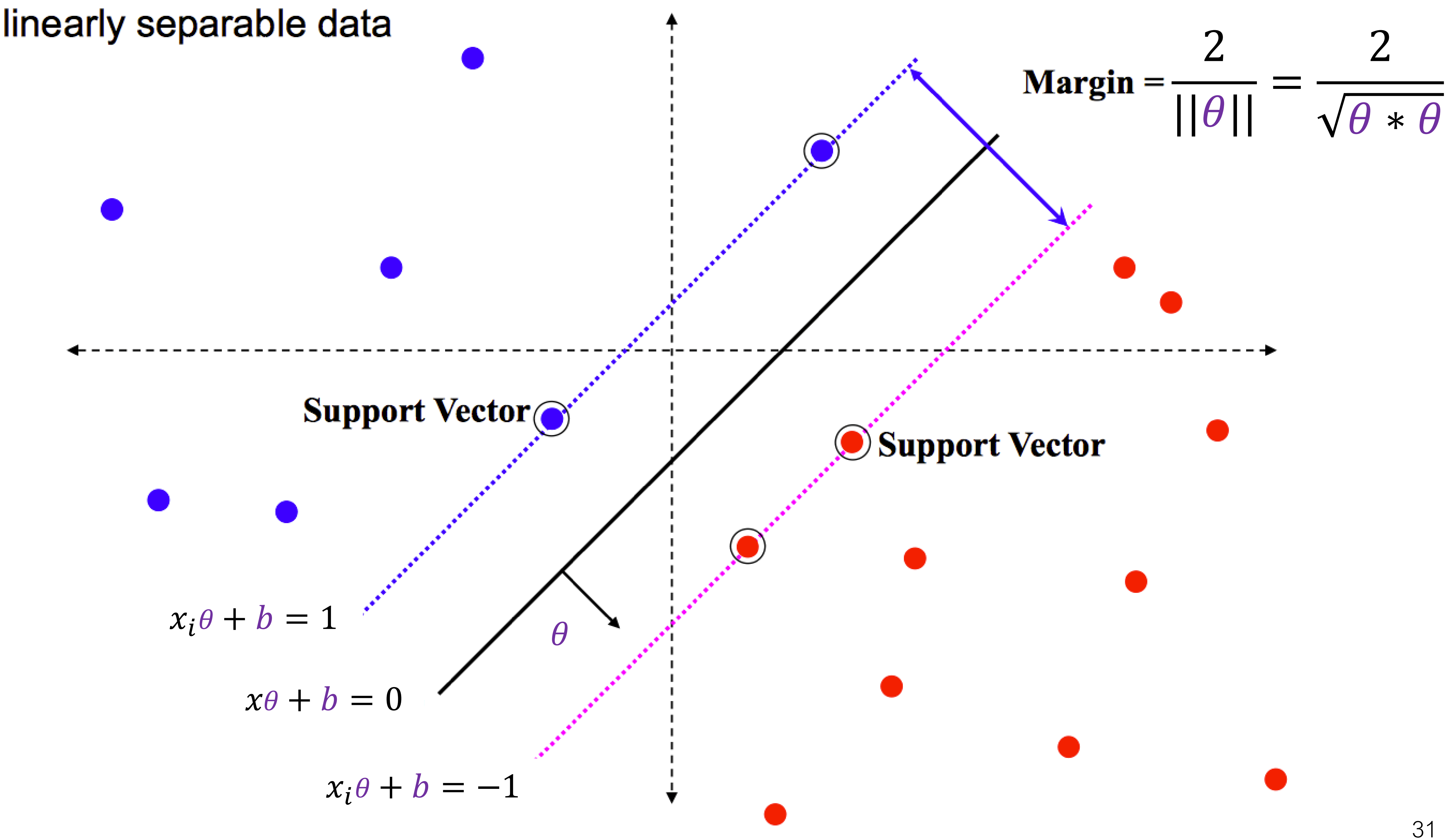
No need to go over all datapoints

$$\rightarrow \theta = \sum_{x_i\, in\, SV} \alpha_i y_i x_i$$

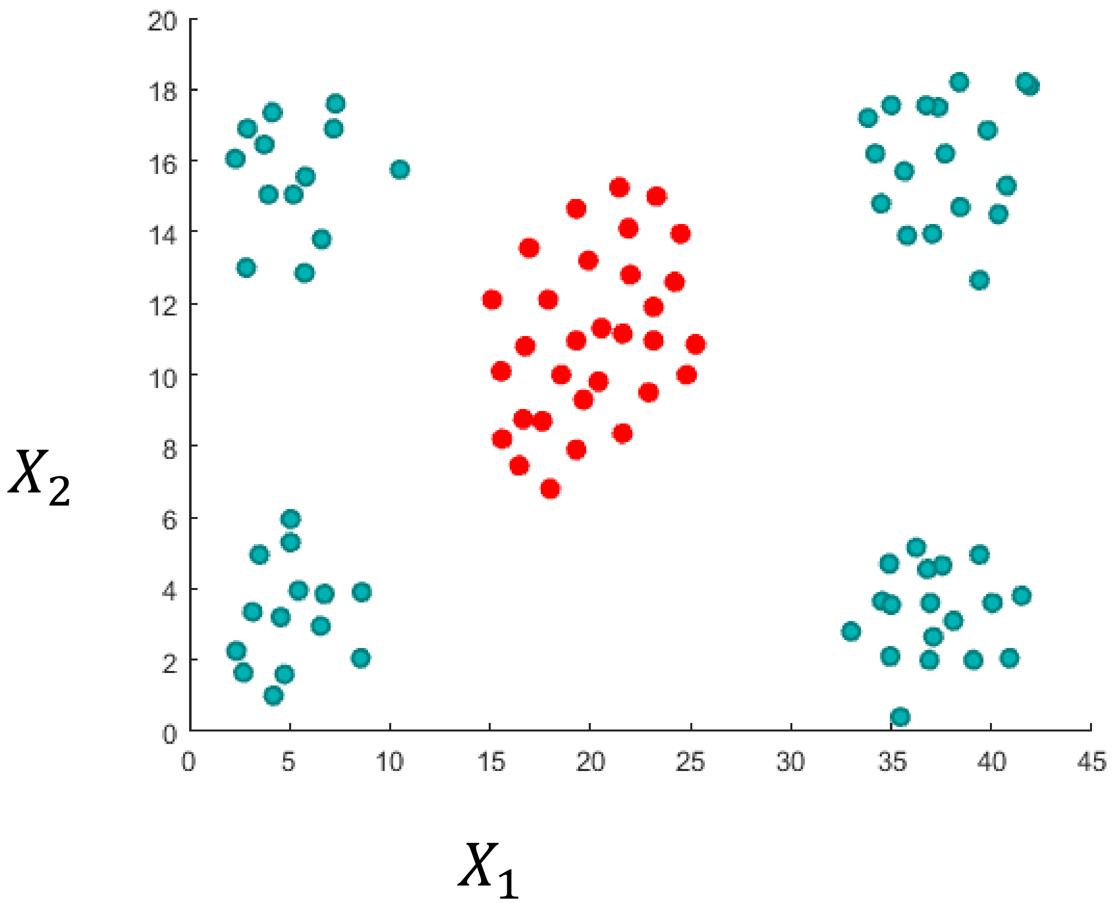
and for $b$ pick any support vector and calculate:

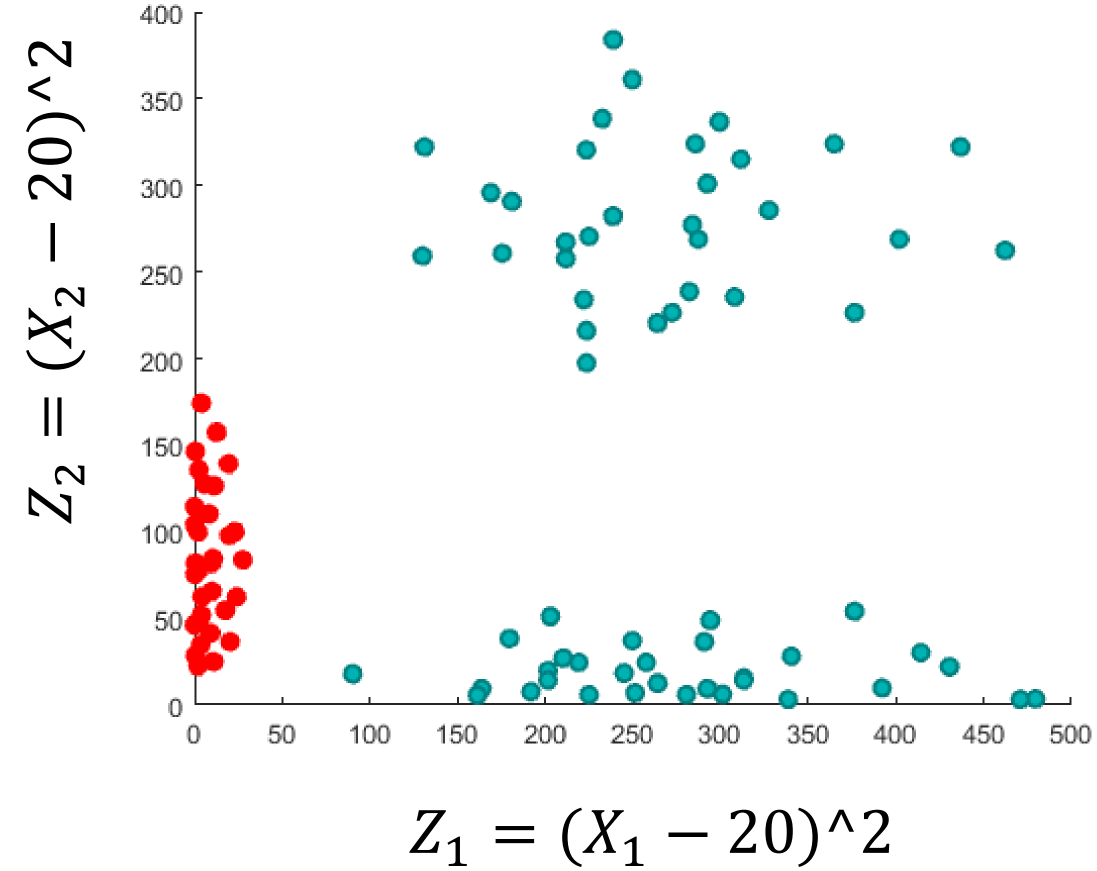
$$y_i(x_i\theta + b) = 1$$

# Geometric Interpretation



# From $x$ to $z$ space

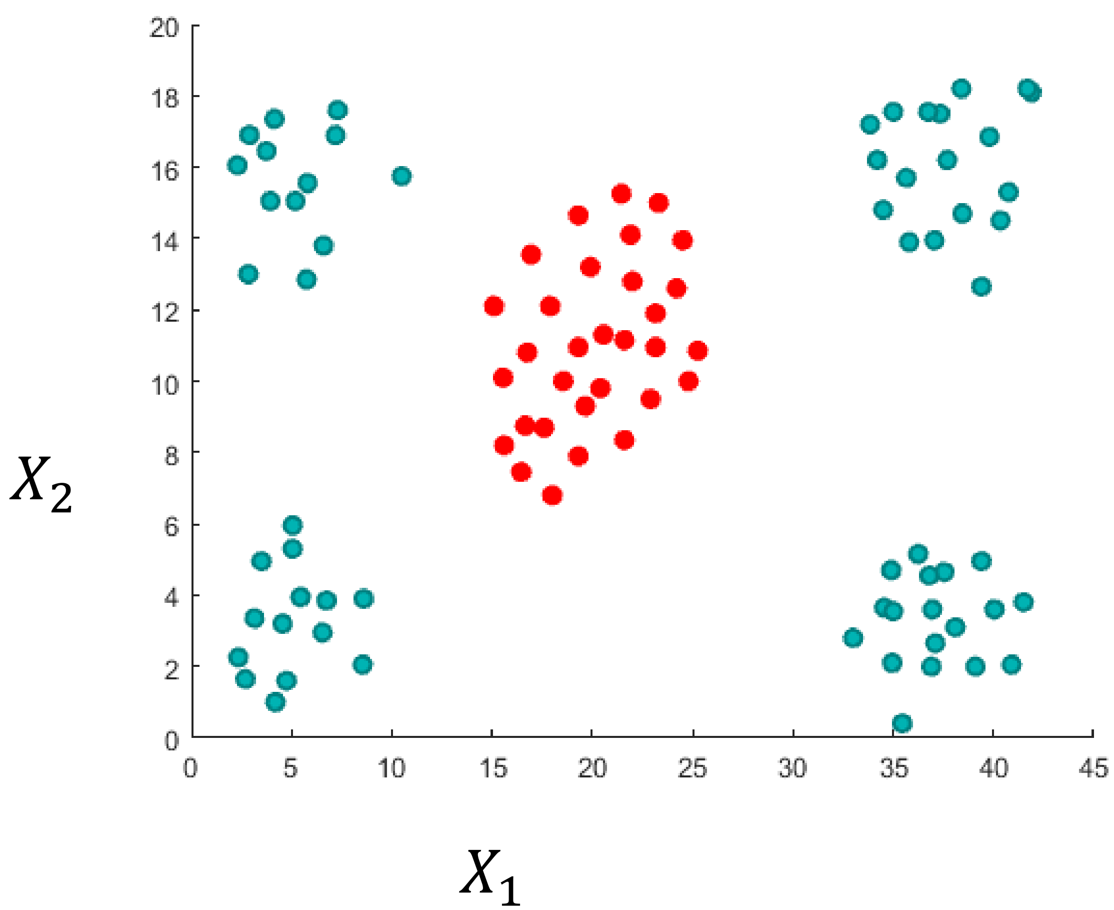



$$X \xrightarrow{\quad (X_1^2, X_2^2) \quad} Z$$

# In $x$ space

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j x_i x_j^T$$



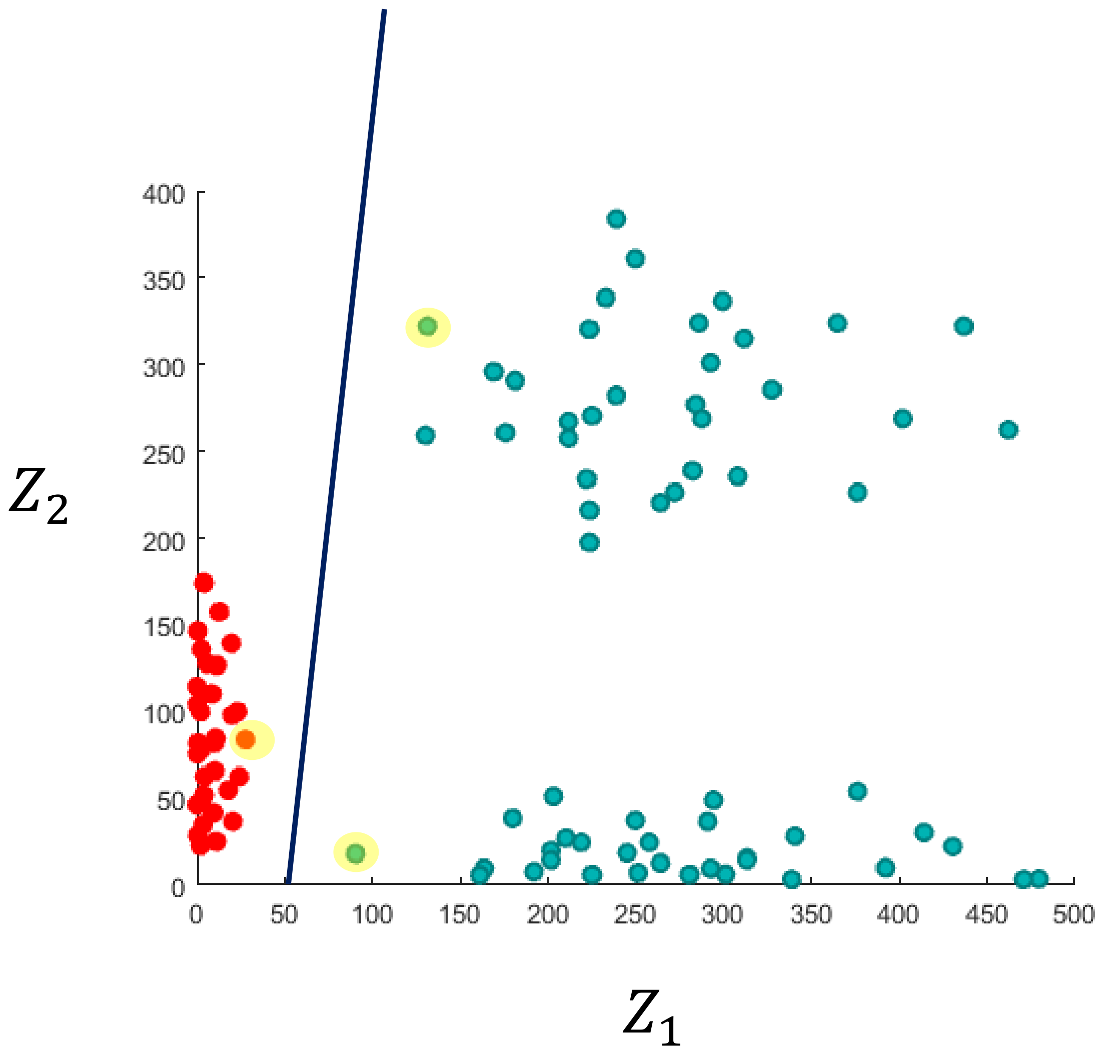

$let's\ say\ x\ is\ n \times d$

$x\mathrm{x}^{\mathrm{T}}$ will be $\mathrm{n} \times n$

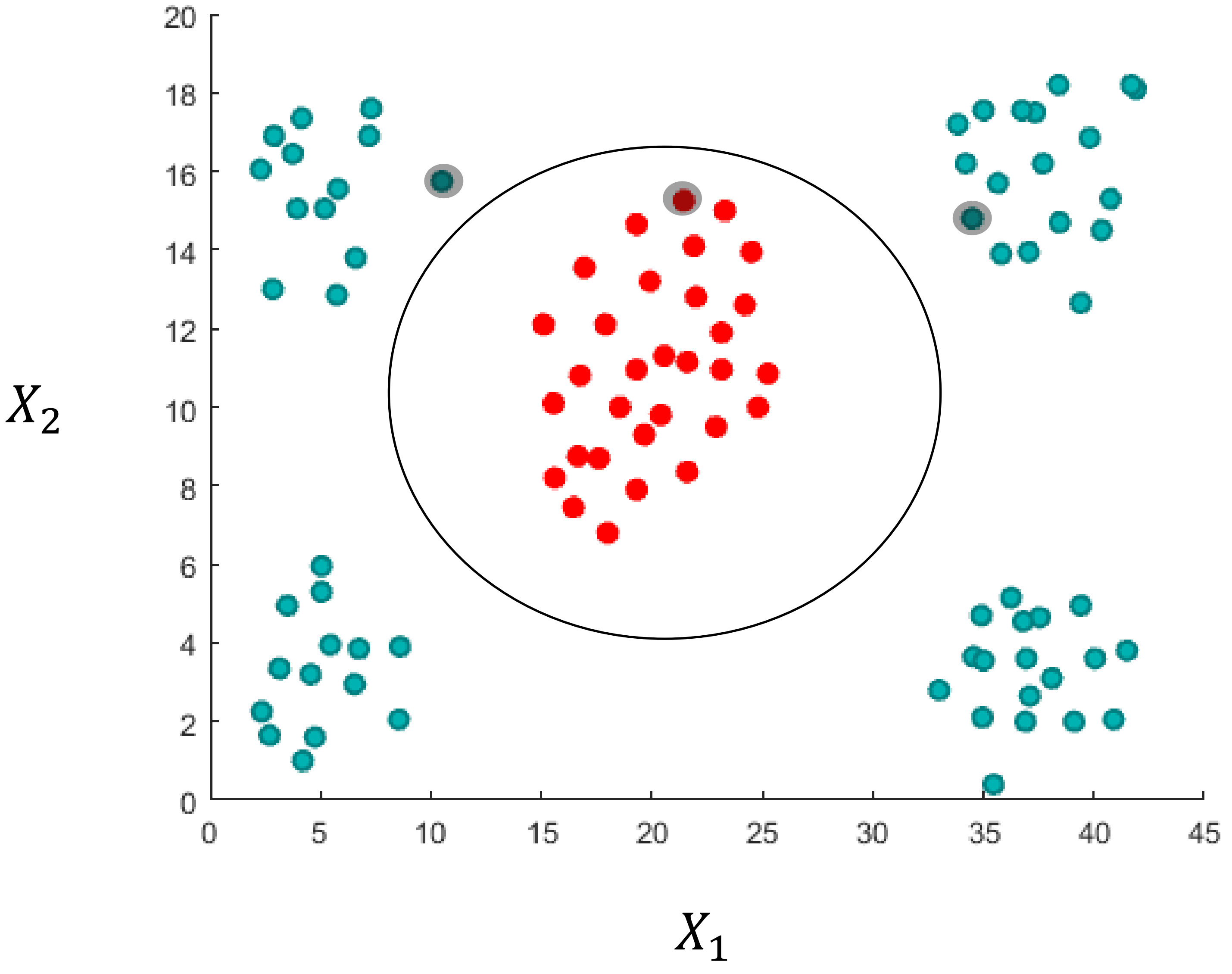
If I add millions of dimensions to $x$, would it affect the final size of $x\mathrm{x}^{\mathrm{T}}$?

# In $z$ space

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j z_i z_j^T$$



# In $x$ space, they are called pre-images of support vectors



# Take-Home Messages

- Linear Separability
- Perceptron
- SVM: Geometric Intuition and Formulation