Support Vector Machine

Mahdi Roozbahani

Georgia Tech

Some of the slides are inspired based on slides from Andrew Zisserman, Yaser S. Abu-Mostafa.
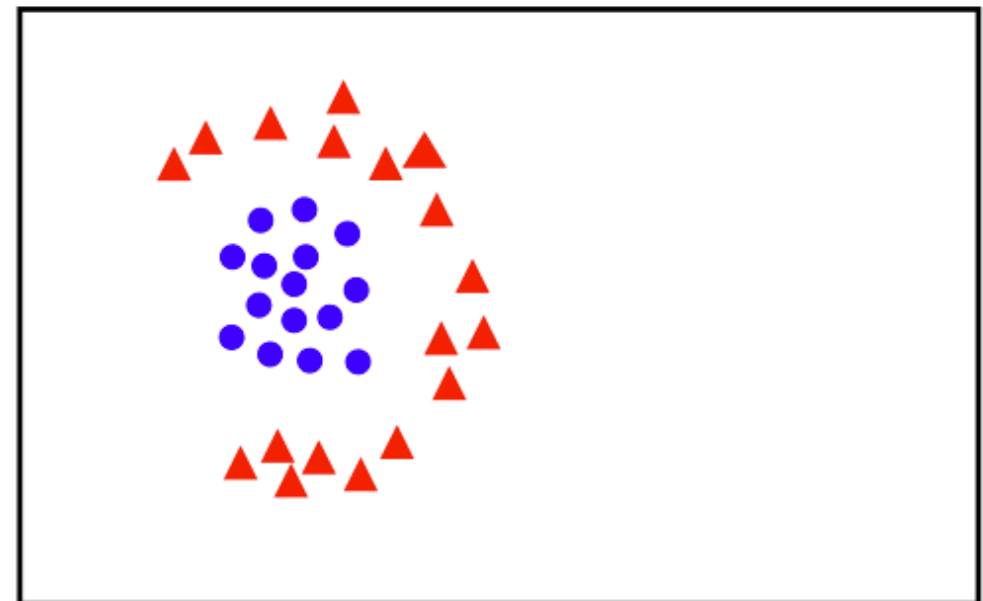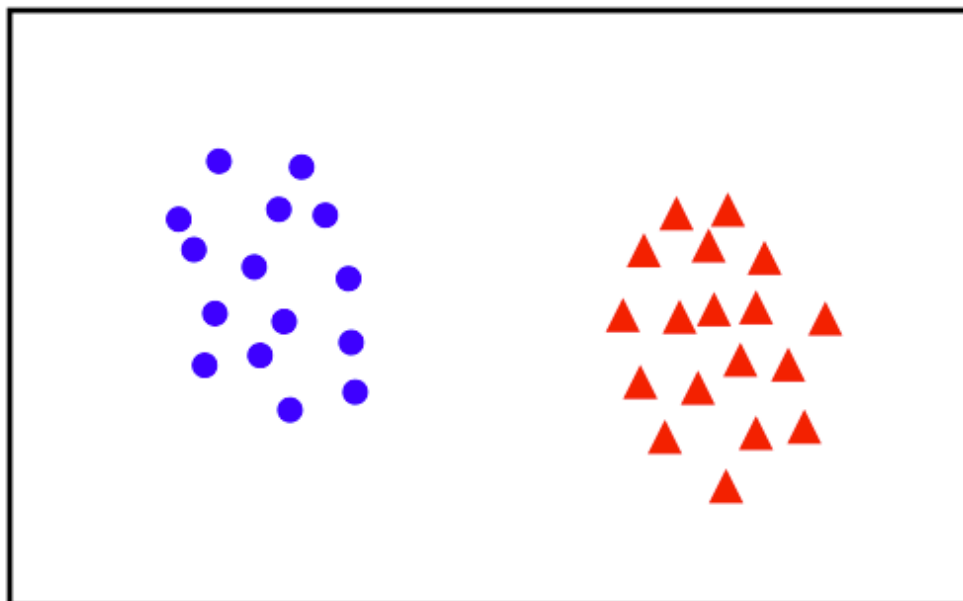
Outline

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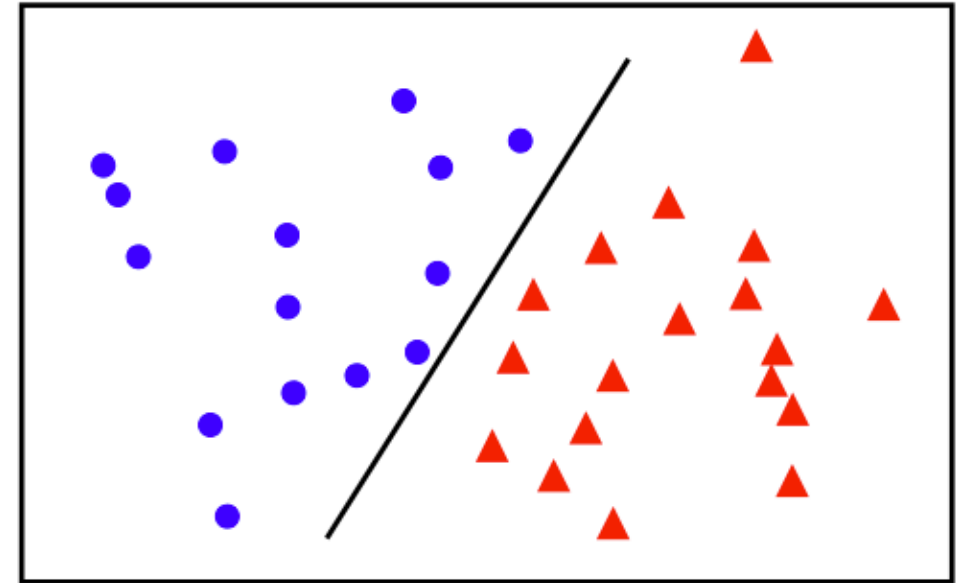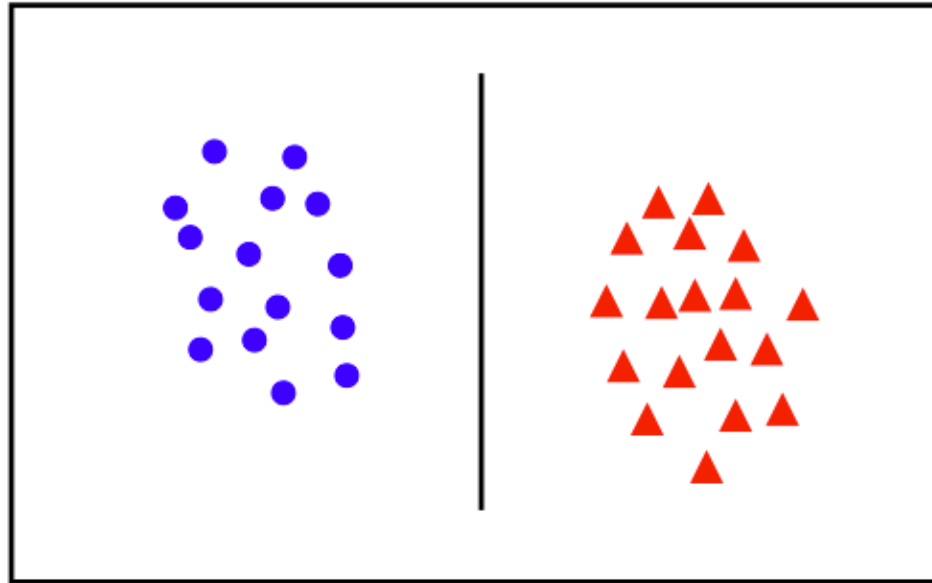
$$f(\mathbf{x}_i) \begin{cases} \geq 0 & +1 \\ < 0 & -1 \end{cases}$$

+ +
- -

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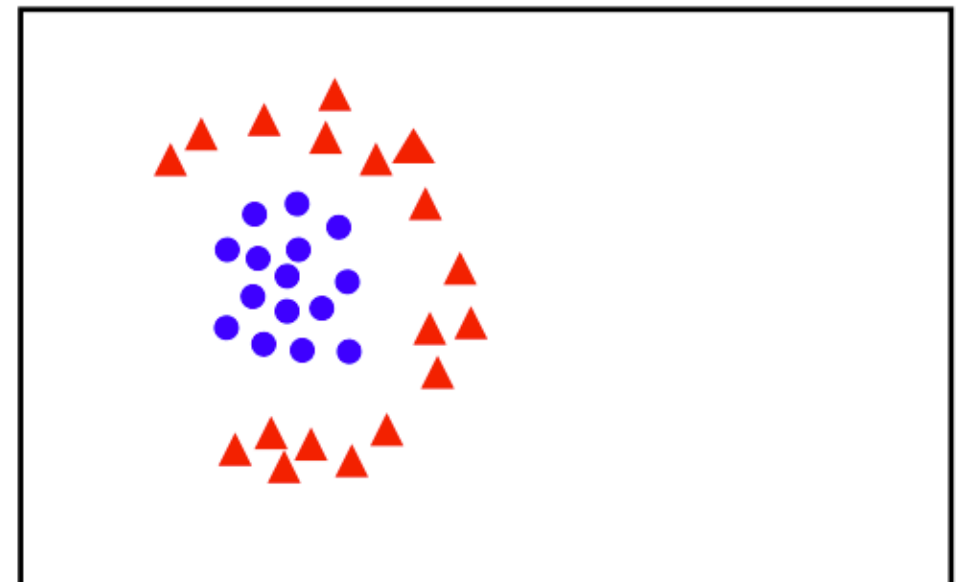
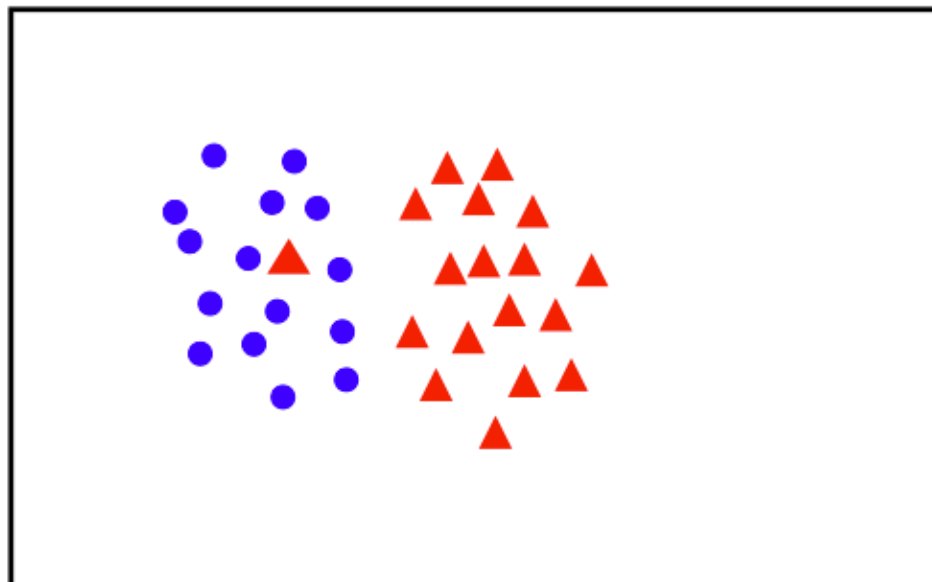# Linear Separability

linearly separable

not linearly separable

Linear Classifier

A linear classifier has the form

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

$f(\mathbf{x}) = 0$

X_2

$f(\mathbf{x}) < 0$ $f(\mathbf{x}) > 0$

X_1
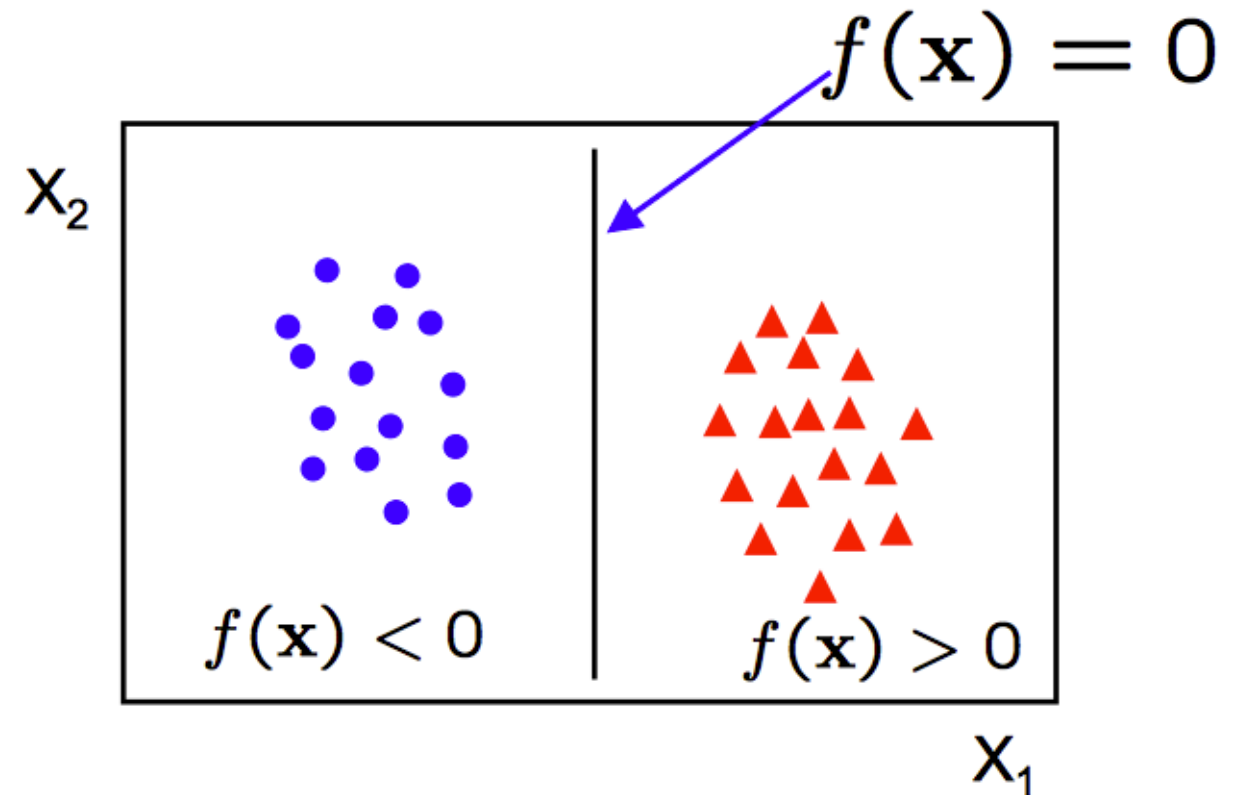
- in 2D the discriminant is a line

- θ is the normal to the line, and θ_0 the bias

- θ is known as the weight vector

6

Linear Classifier (higher dimension)

A linear classifier has the form

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

$f(\mathbf{x}) = 0$
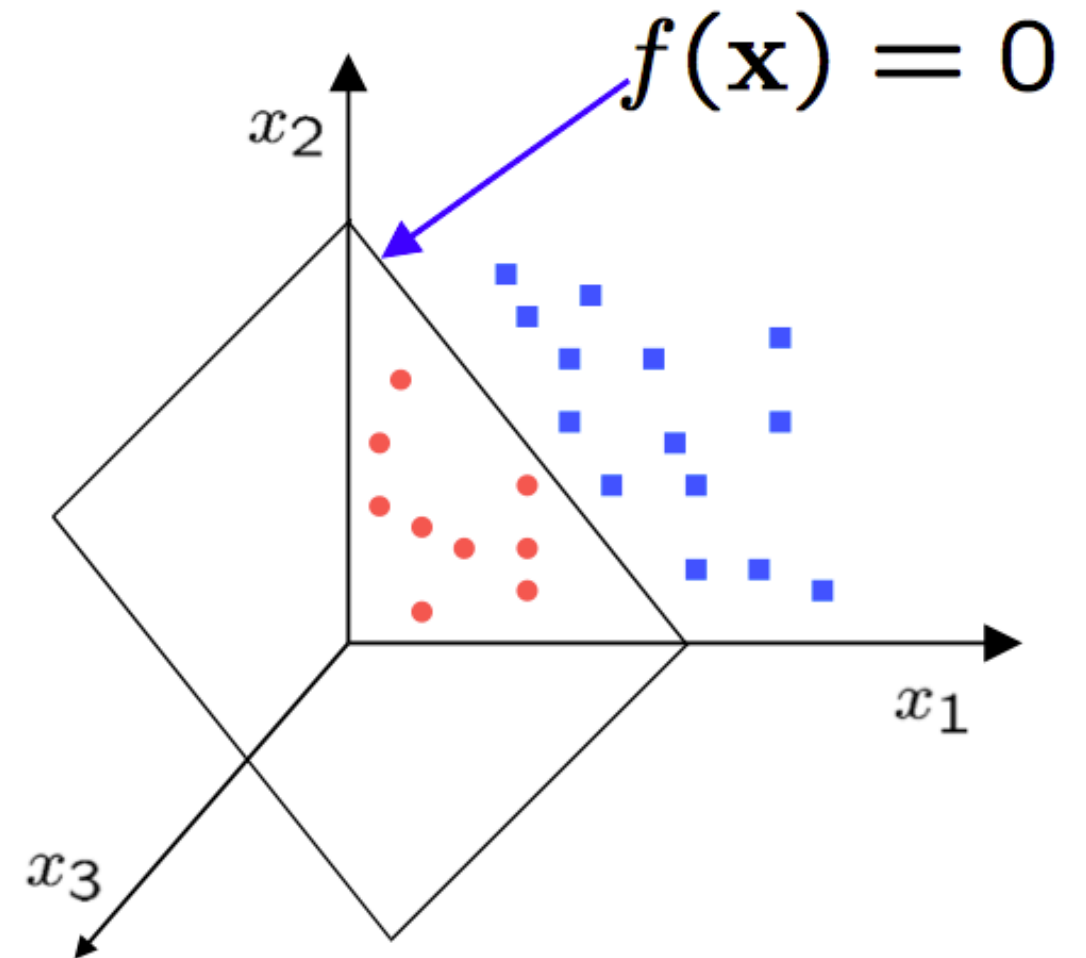
- in 3D the discriminant is a plane, and in nD it is a hyperplane

The Perceptron Classifier

Considering $\boldsymbol{x}$ is linearly separable and $\boldsymbol{y}$ has two labels of $\{-1,1\}$

$$f(x_i) = x_i \theta \qquad \text{Bias is inside } \theta \text{ now}$$

How can we separate datapoints with label 1 from datapoints with label -1 using a line?

Perceptron Algorithm:

Misclassified

Ex. $y_i f(x_i) < 0$

actual   predicted

- Initialize $\theta = 0$
- Go through each datapoint $\{x_i, y_i\}$
    - If x_i is misclassified then $\theta^{t+1} \leftarrow \theta^t + \alpha \, y_i x_i$
- Until all datapoints are correctly classified

8

- Initialize $\theta = 0$
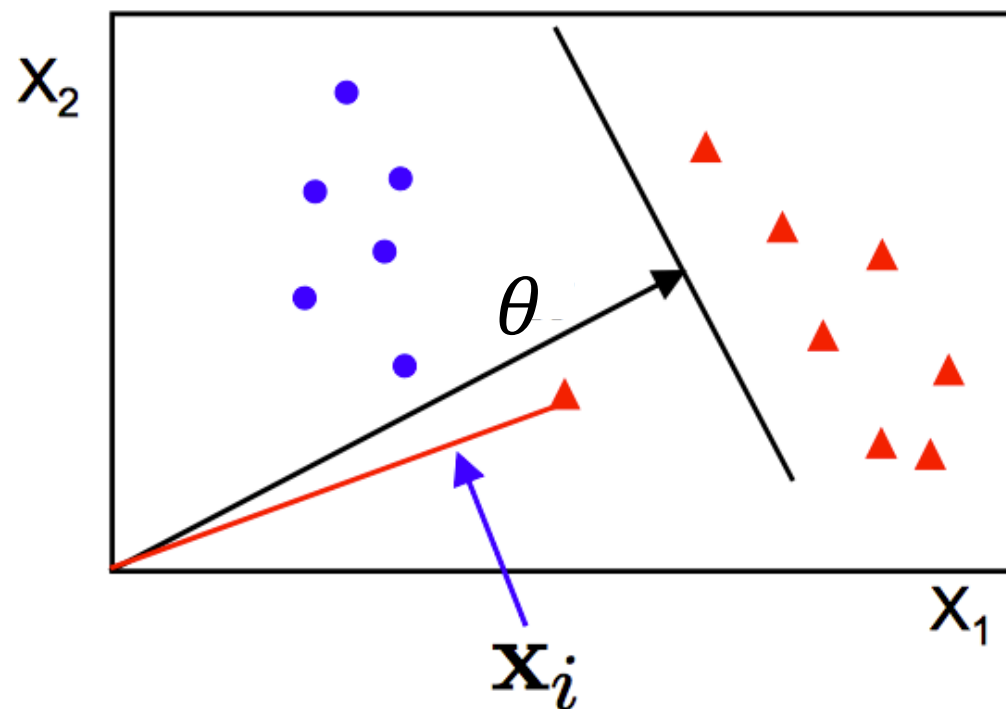- Go through each datapoint $\{x_i, y_i\}$
  - If x_i is misclassified then $\theta^{t+1} \leftarrow \theta^t + \alpha y_i x_i$

- Until all datapoints are correctly classified

$$\theta^{t+1} \leftarrow \theta^t - \alpha x_i$$
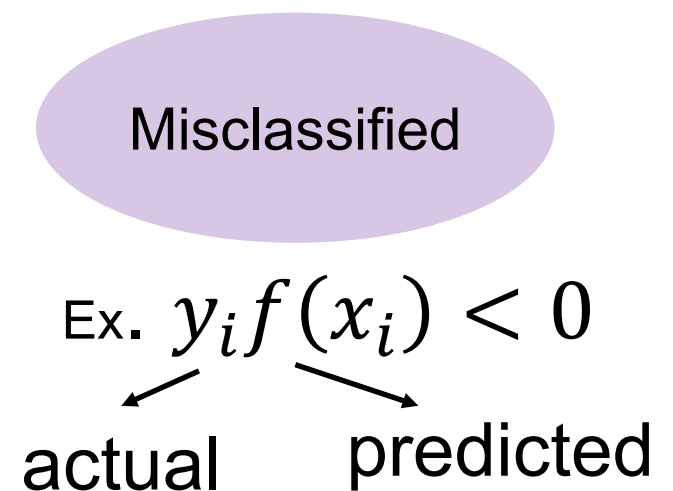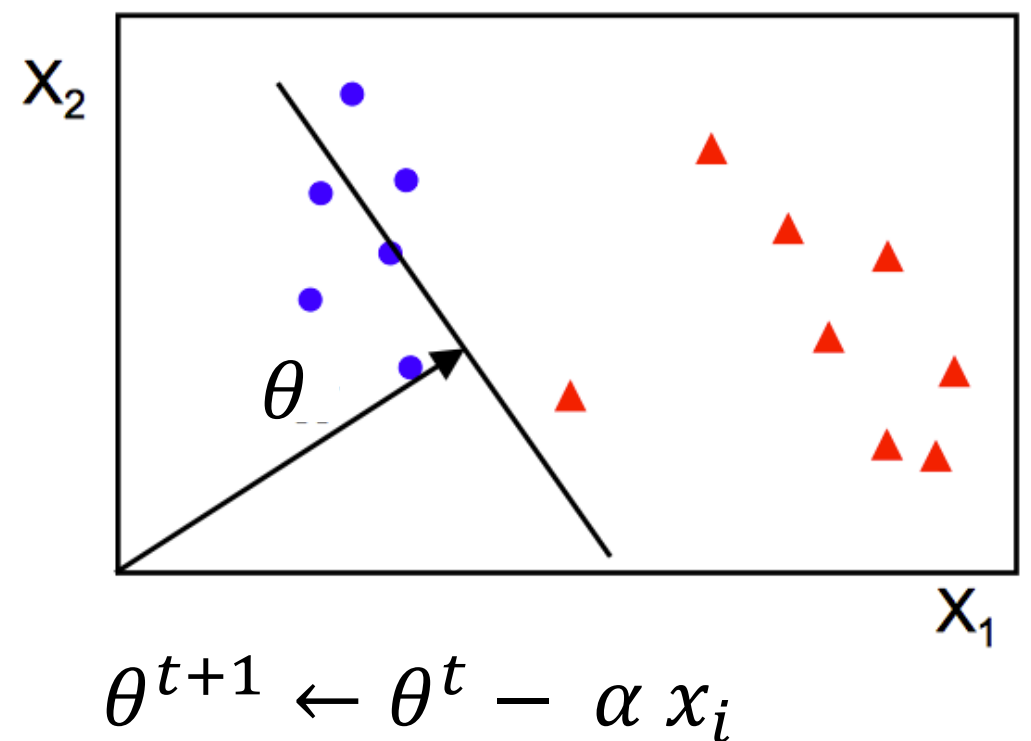
Linear separation

We can have different separating lines
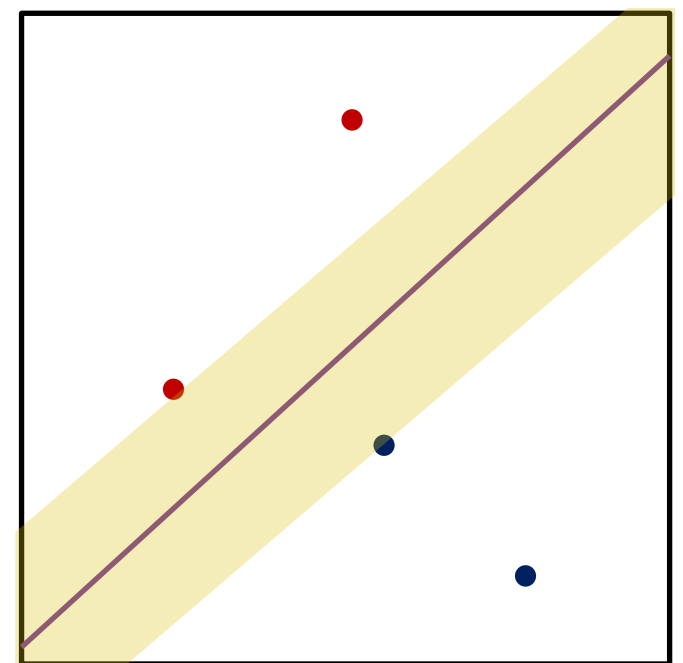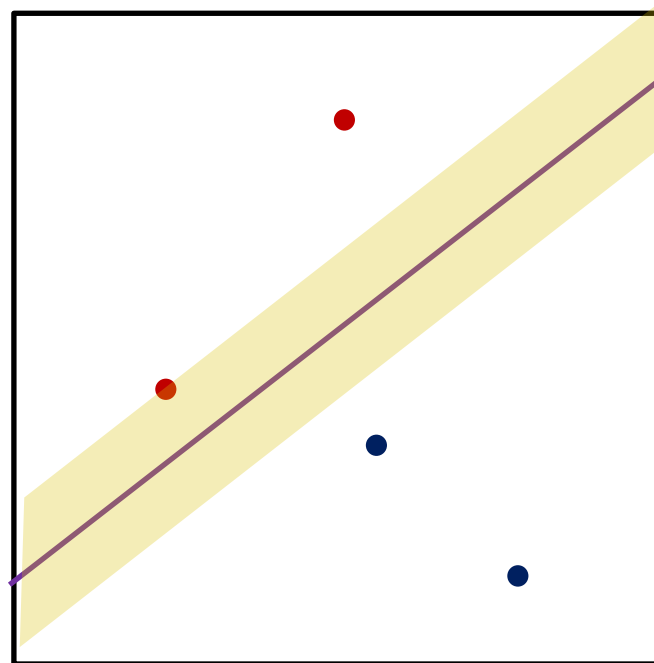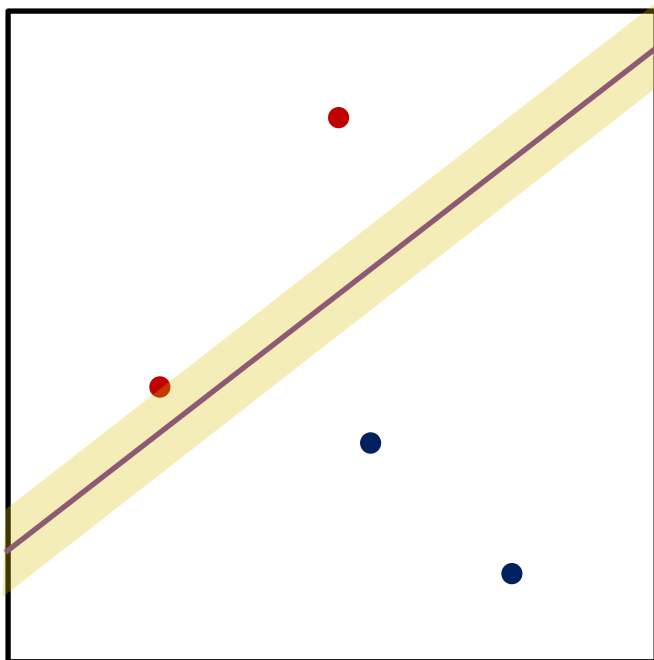
Which line is the best?

Why is the bigger margin better?

What θ maximizes the margin?

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

What is the Best θ?
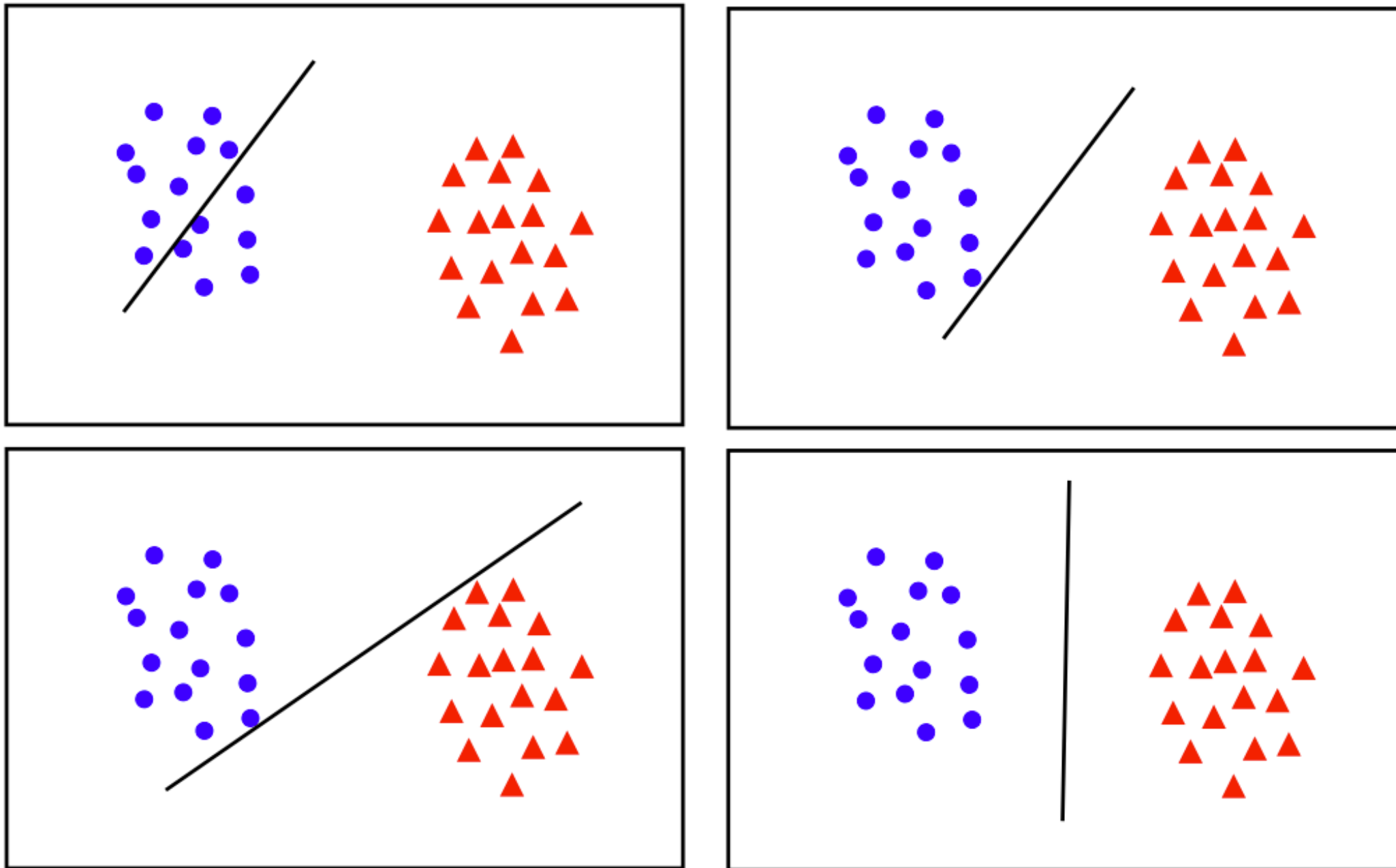
- **maximum margin** solution: most stable under perturbations of the inputs

Perceptron example
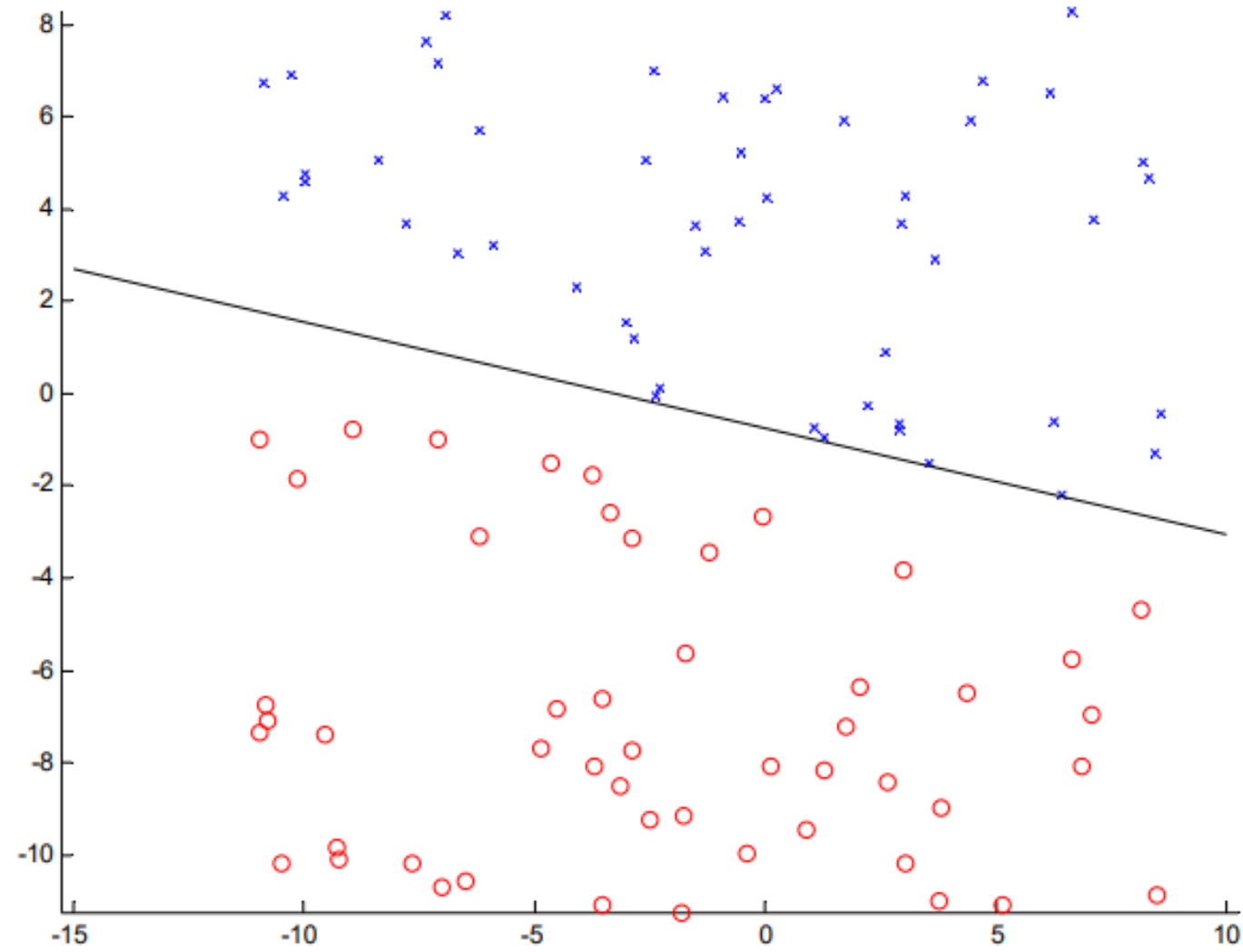
- if the data is linearly separable, then the algorithm will converge

- 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 θ with a **fat** margin

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

x_2

$x_i\theta > 0$

$x_i\theta < 0$

x_1

$x\theta = f(\mathbf{x}) = 0$

x_2

x_1

x_3
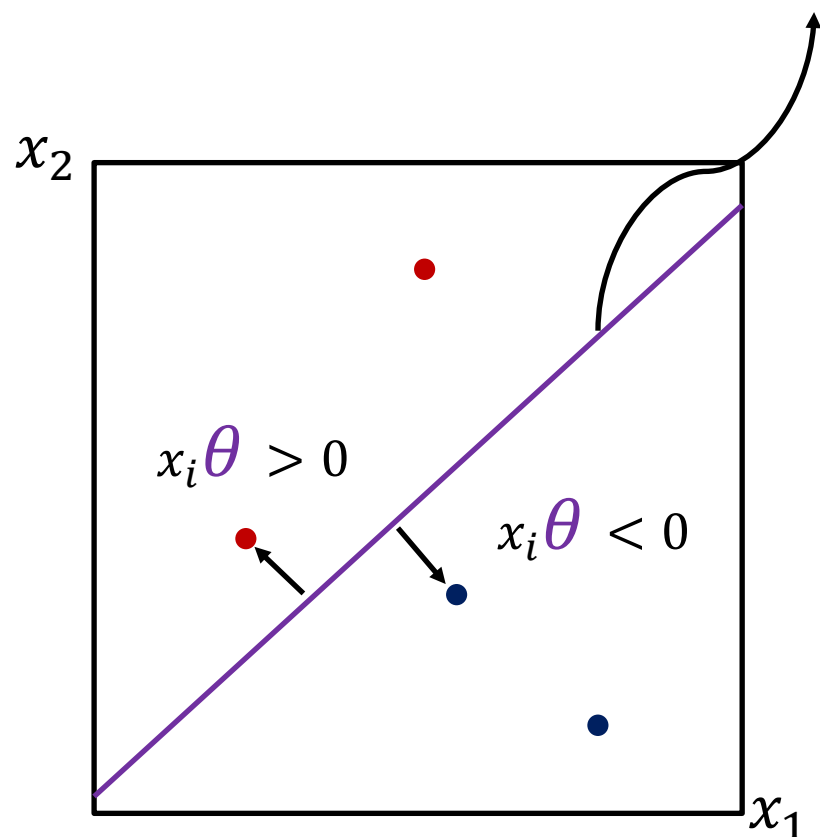
Let x_i to be the nearest data point to the line (plane):

$$|x_i\theta| > 0$$

Our line solution is $x\theta = 0$

Does it matter if I scale up or down θ for the decision boundary?

$$|x_i\theta| = 1 \rightarrow \text{normalization}$$

Let's pull out θ_0 from $\theta = (\theta_1, \ldots, \theta_d)$ and call it be b

Decision boundary would be: $\boxed{x\theta + b = 0}$
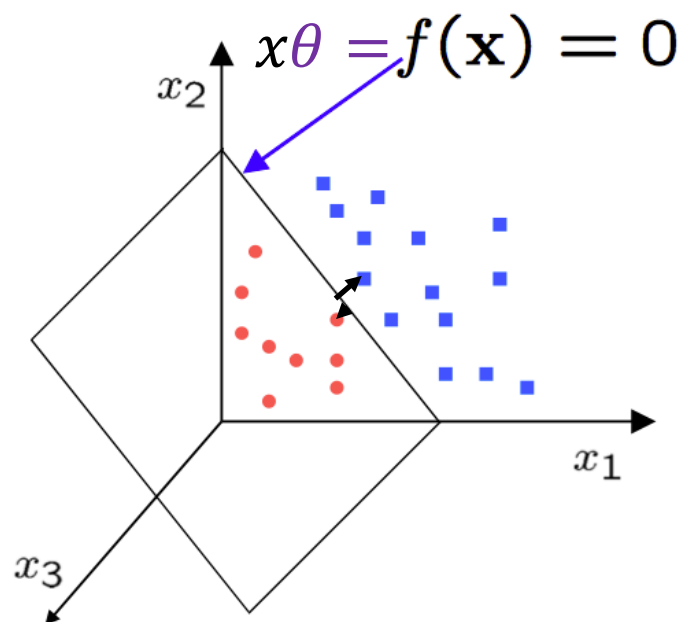
Computing the distance

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

The vector θ is perpendicular to the decision line.

You should ask me why?

Consider x' and x'' on the plane

$$x'\theta + b = 0 \quad \text{and} \quad x''\theta + b = 0$$

$$\downarrow$$

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

$$\implies (x' - x'')\theta = 0$$
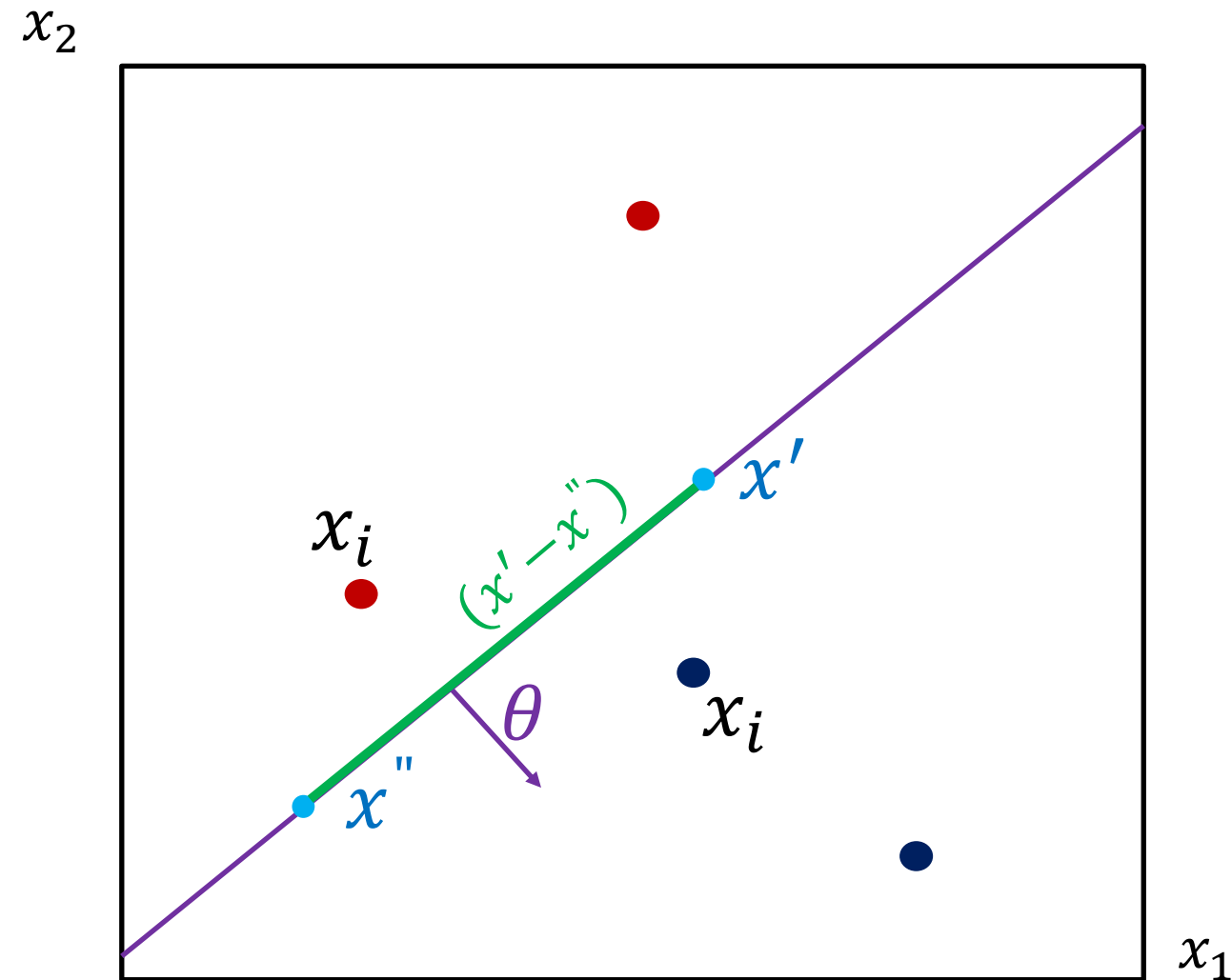
What is the distance of my fat margin?

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 θ.

To project the vector, we need to normalize θ to get the unit vector.

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

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

$$= \frac{1}{||\theta||} |(\underbrace{x_i\theta + b}_{\substack{\text{My constraint} \\ |x_i\theta + b| = 1}} - \underbrace{x\theta - b}_{\substack{\text{A point on the} \\ \text{decision line} \\ x\theta + b = 0}})| = \frac{1}{||\theta||}$$
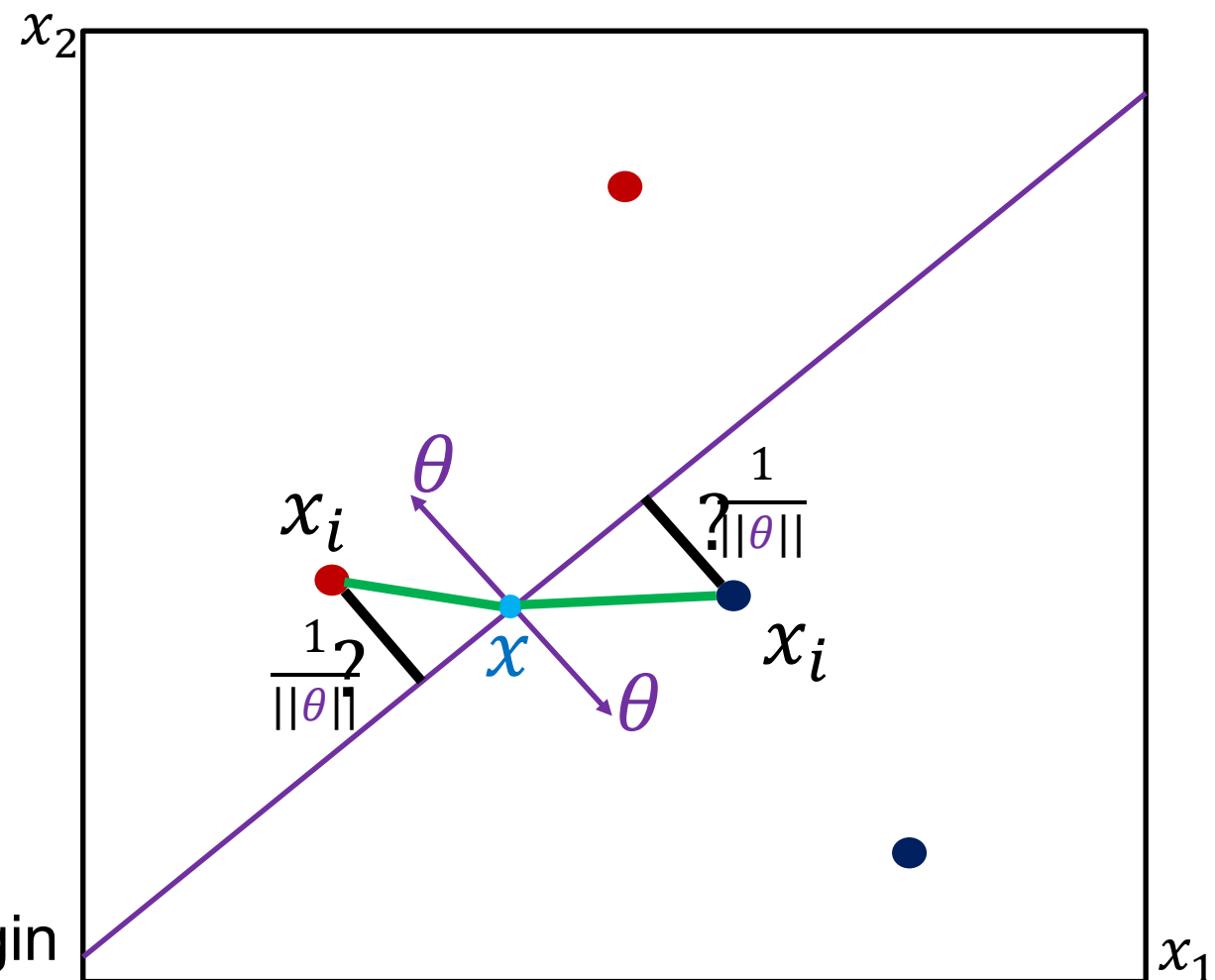
The margin

Now we need to maximize the margin

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

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 $\dfrac{1}{||\theta||}$

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

Class 2

Class 1

$x_i\theta + b = 1$

$x\theta + b = 0$
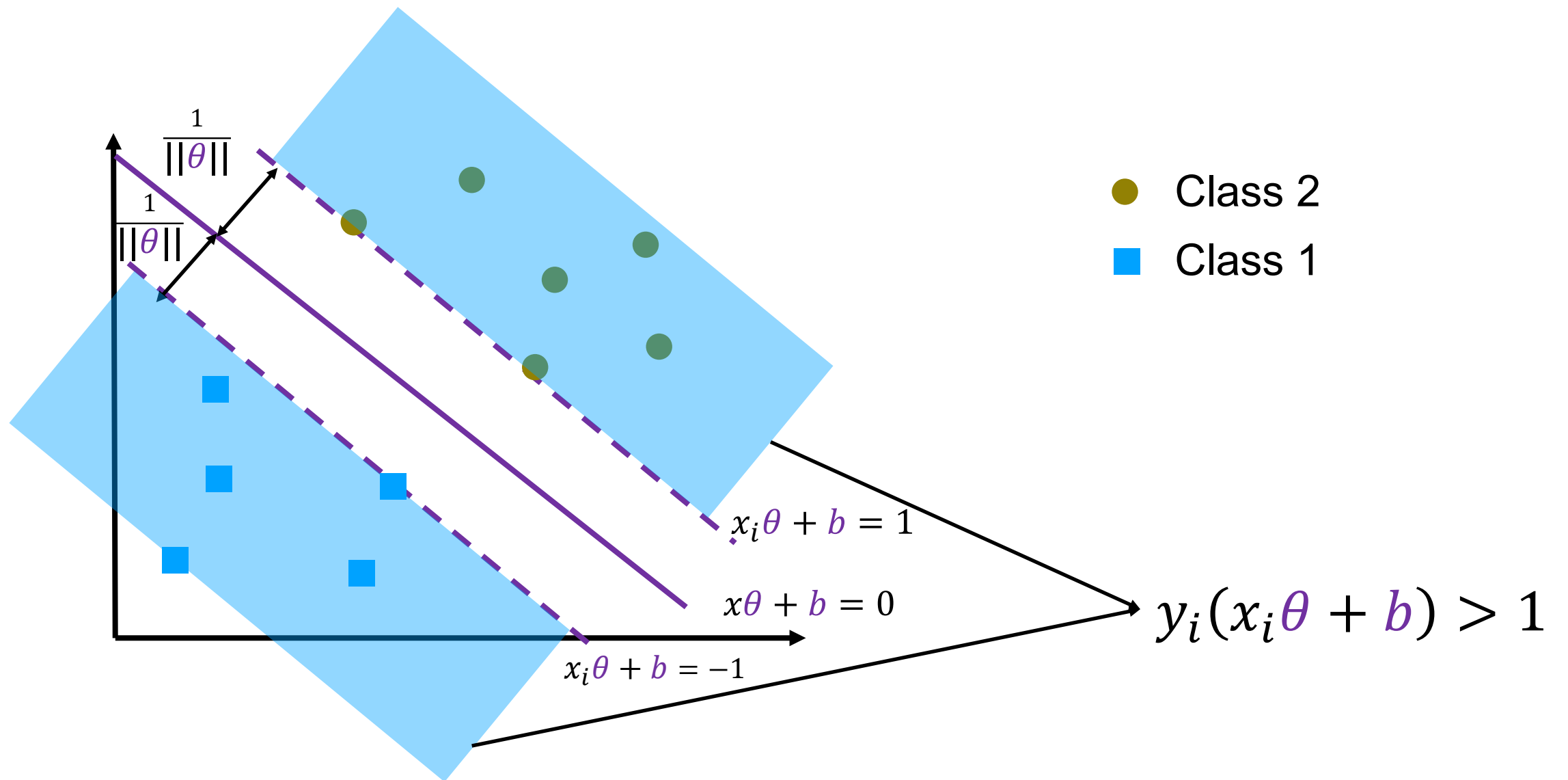
$x_i\theta + b = -1$

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

Maximize $\dfrac{2}{||\theta||}$

If $\theta \neq 0,$ there exists a max value

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

Minimize $\quad \dfrac{1}{2}\theta\theta^T$

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

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

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

We need to optimize

1) Stationary $\qquad \theta, b, and\ \alpha$

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

3) $\alpha \geq 0$ Dual feasibility

4) $g(x)\alpha = 0$ Complementary slackness $\Rightarrow \begin{cases} g(x) > 0, & \alpha = 0 \\ \alpha > 0, & g(x) = 0 \end{cases}$
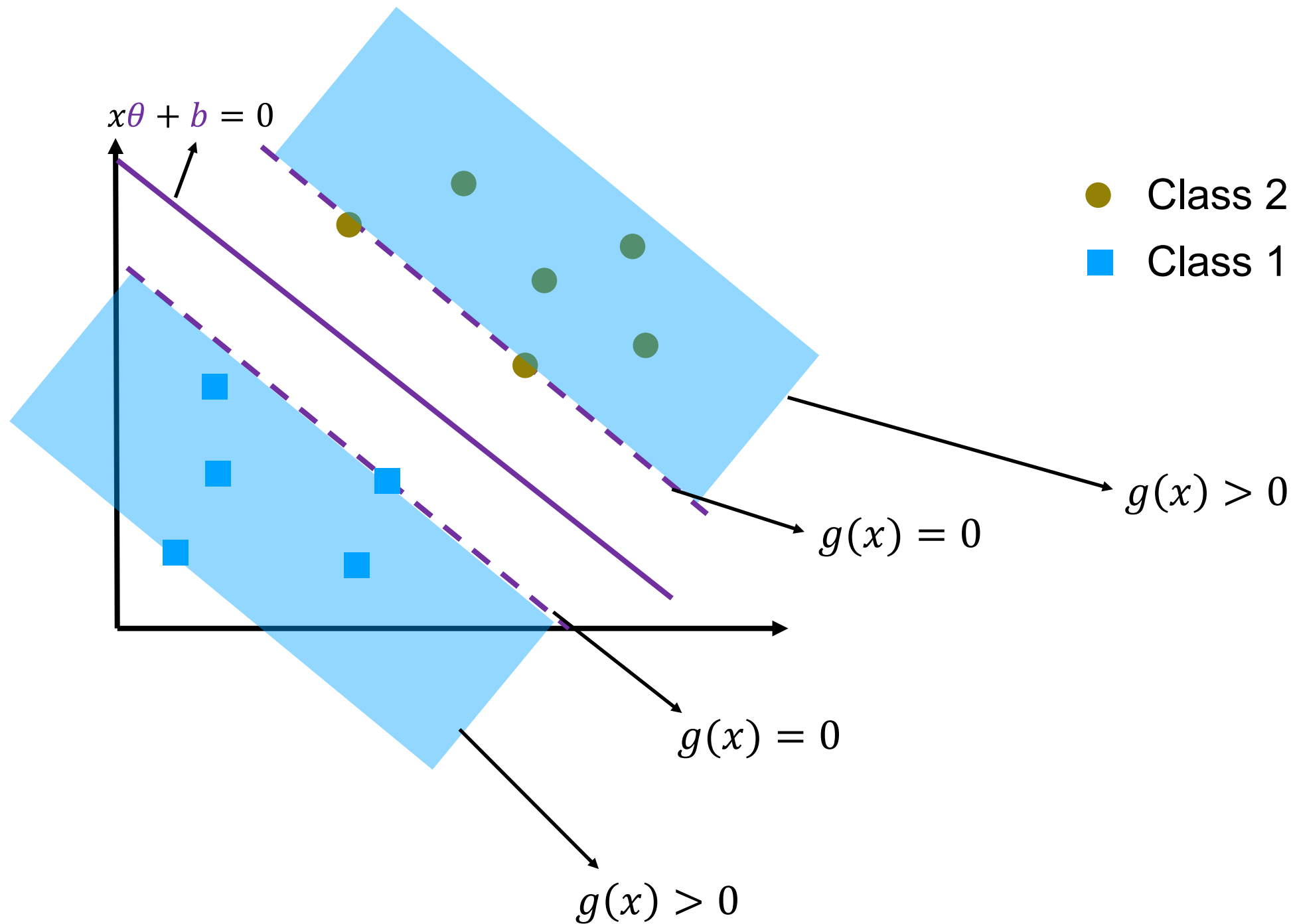
$$g(x) = y_i(x_i\theta + b) - 1$$

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

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

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

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

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

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

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

Let's substitute these in the Lagrangian:

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

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

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

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

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

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

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

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

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"

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

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

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

Linear term

Quadratic coefficients

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

Linear equality constraint

Pass these to a quadratic programming package

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

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - 1^T \alpha \qquad \text{subject to} \qquad y^T \alpha = 0; \alpha \geq 0$$
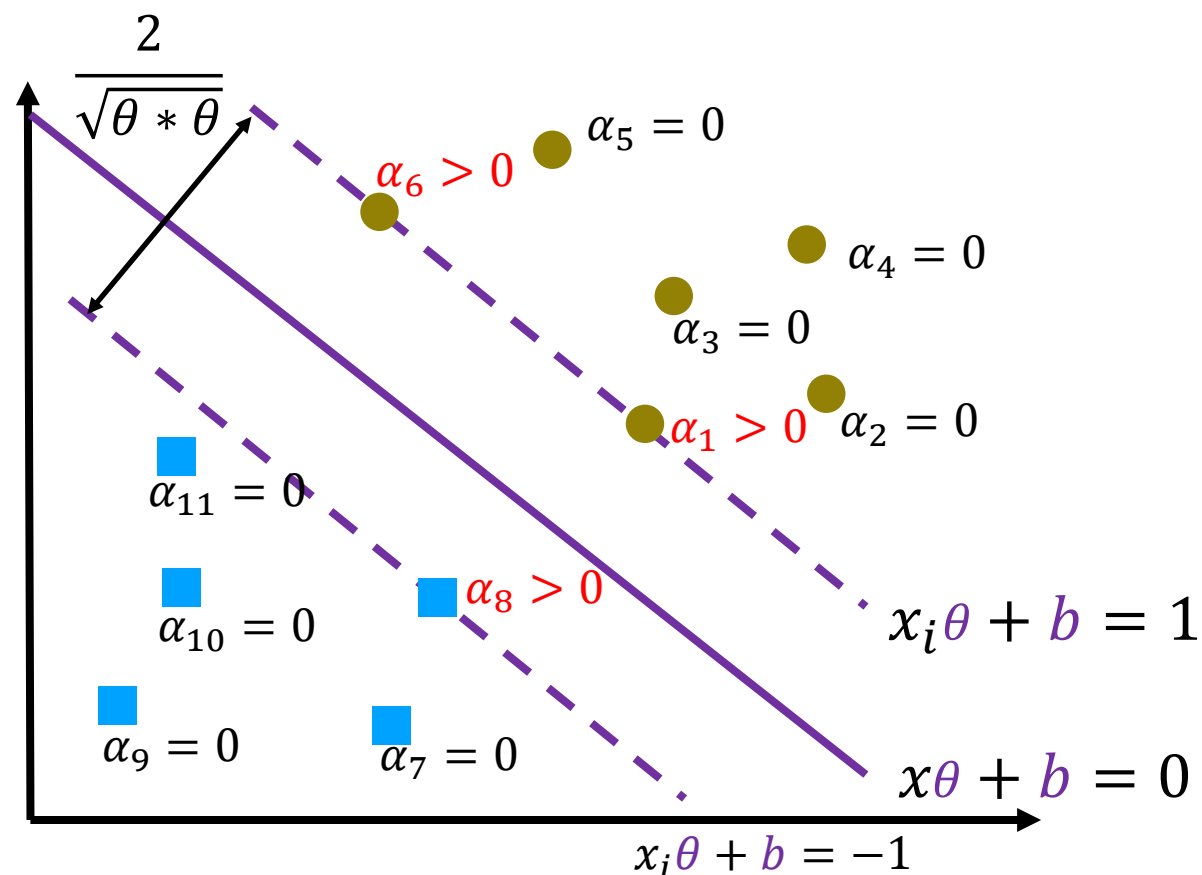
Quadratic programming will give us α

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

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

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

$(y_i(x_i\theta + b) - 1) = 0 \qquad \Rightarrow \qquad \alpha_i > 0 \Rightarrow x_i \text{ is a support vector}$

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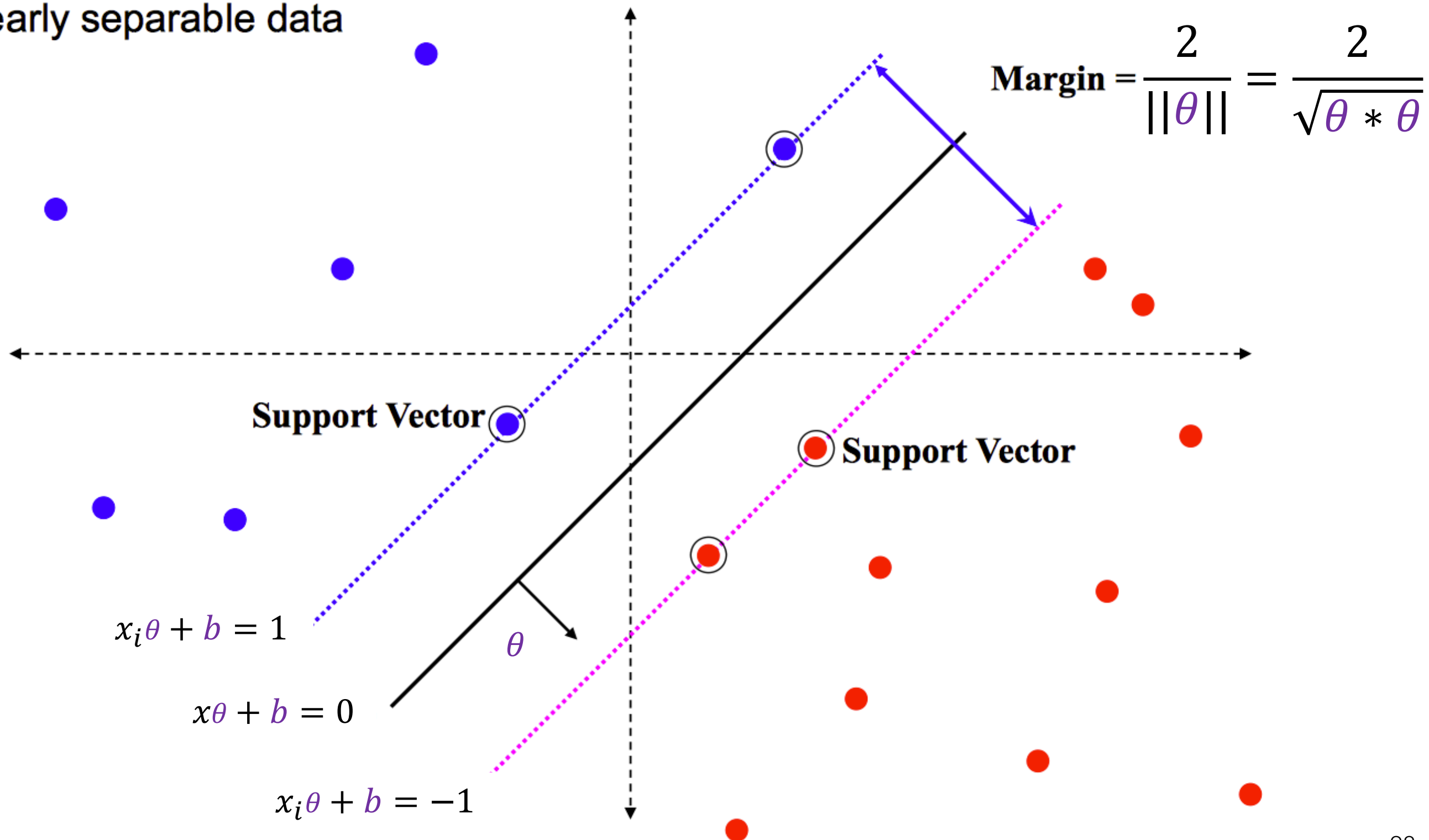
Testing

For a new test point s

Compute:

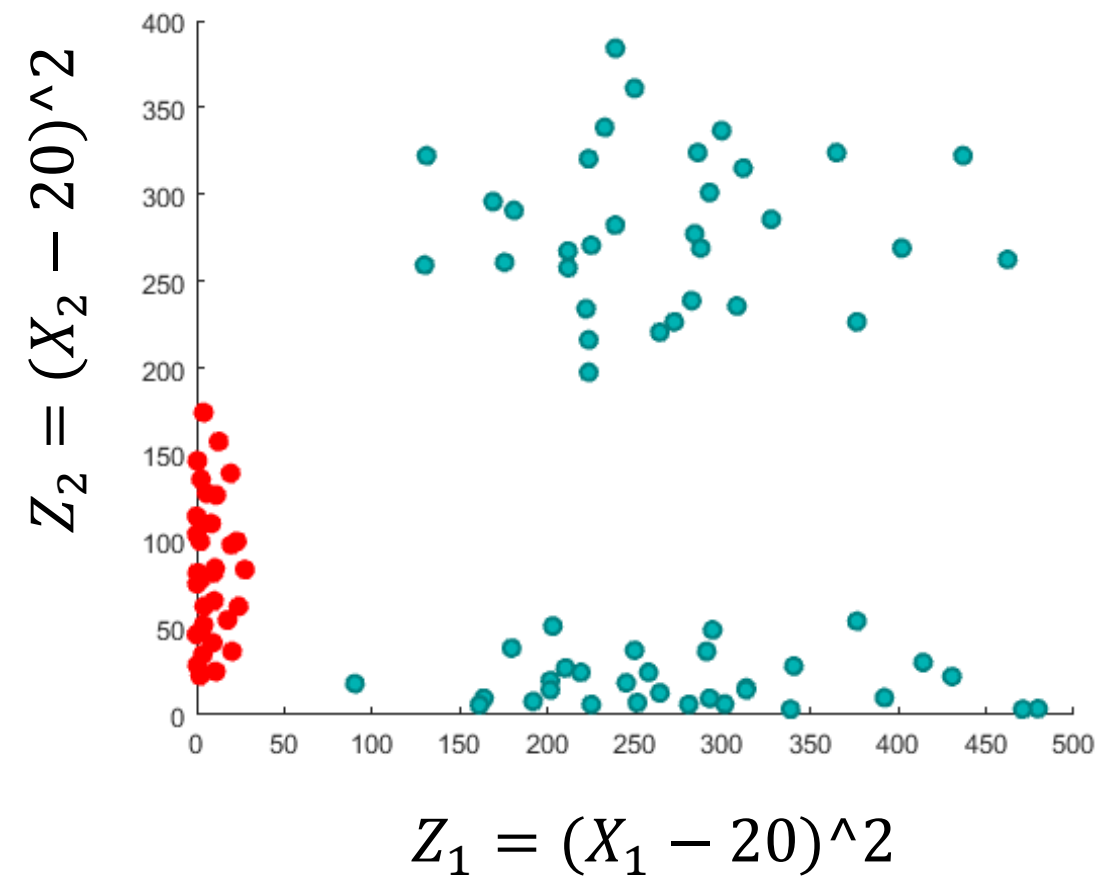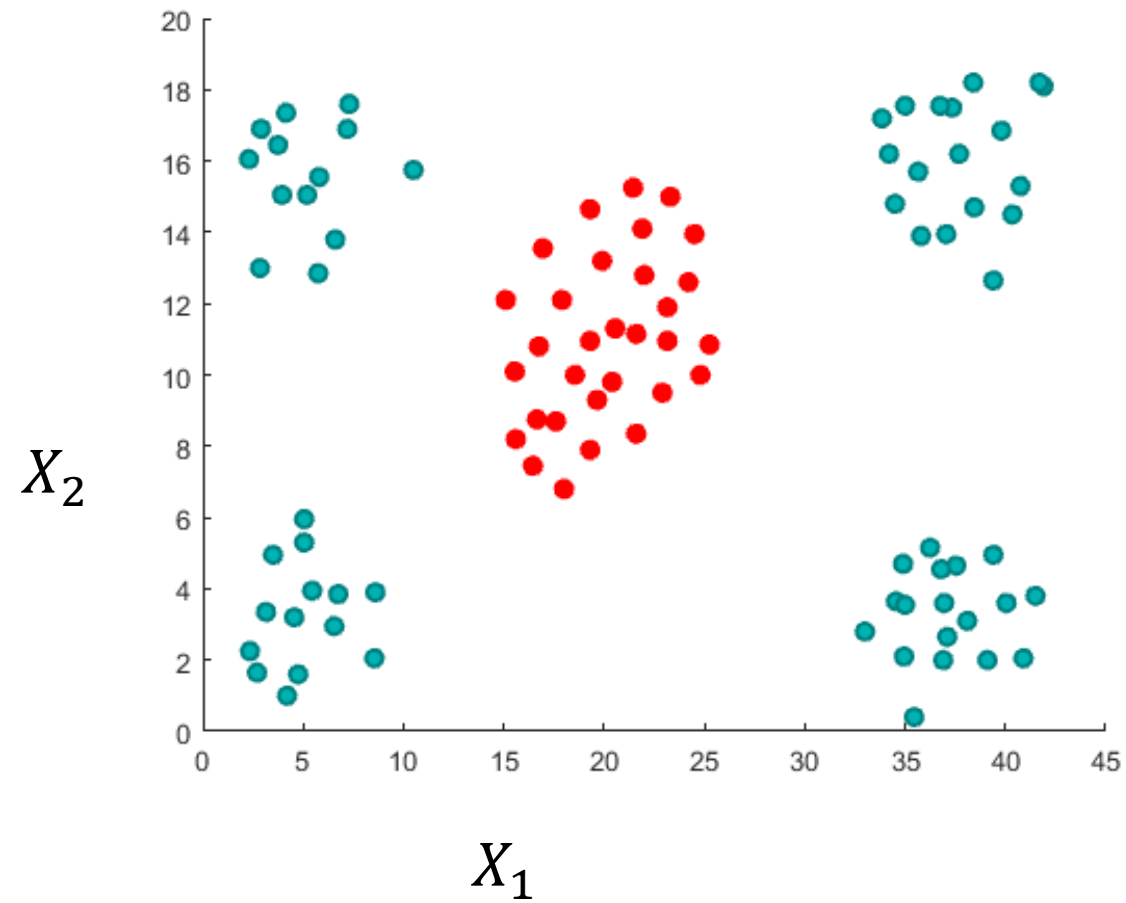$$s\theta + b = \sum_{x_i \, in \, SV} \alpha_i y_i x_i s^T + b$$

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

Geometric Interpretation
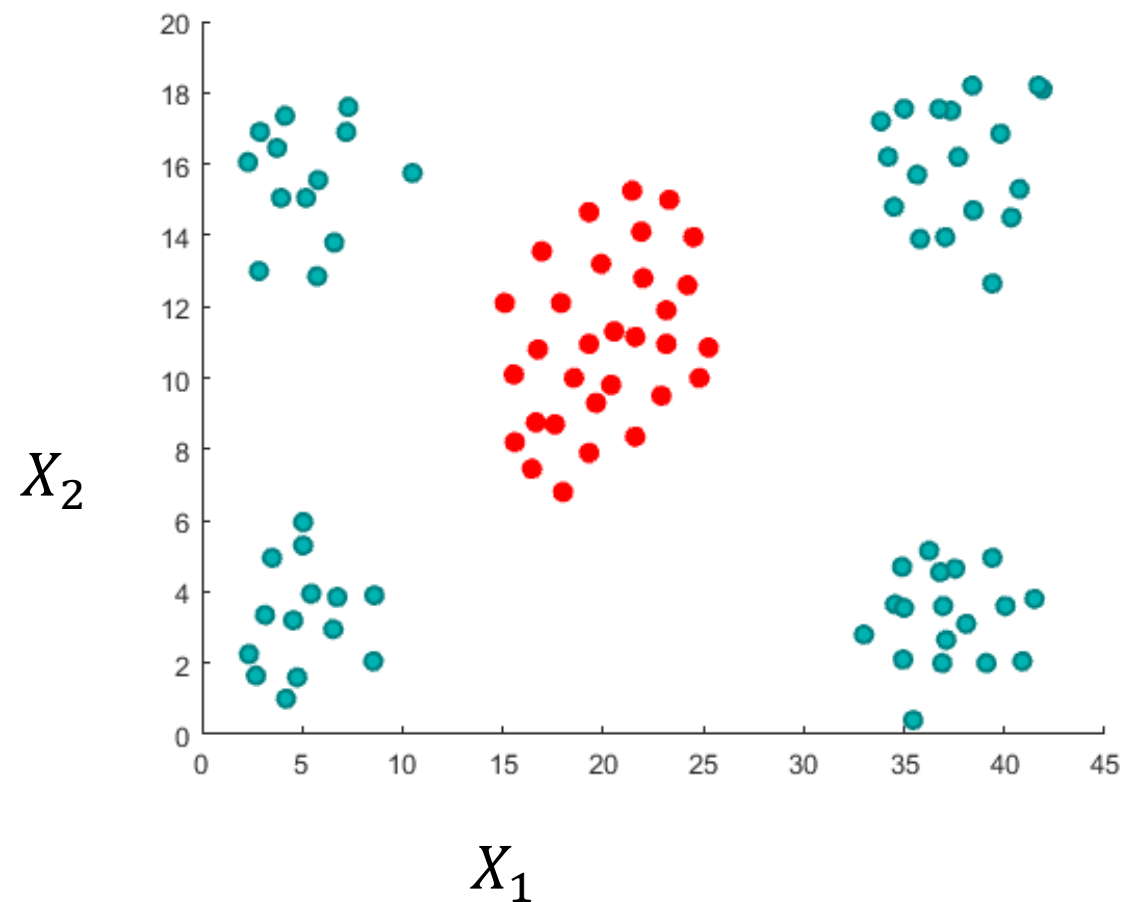

linearly separable data

$$\text{Margin} = \frac{2}{||\theta||} = \frac{2}{\sqrt{\theta * \theta}}$$

Support Vector

Support Vector

$x_i\theta + b = 1$

θ

$x\theta + b = 0$

$x_i\theta + b = -1$

From x to z space

X_2

X_1

$Z_2 = (X_2 - 20)\text{^}2$

$Z_1 = (X_1 - 20)\text{^}2$

$X \longrightarrow Z$

(X_1^2, X_2^2)

In x space

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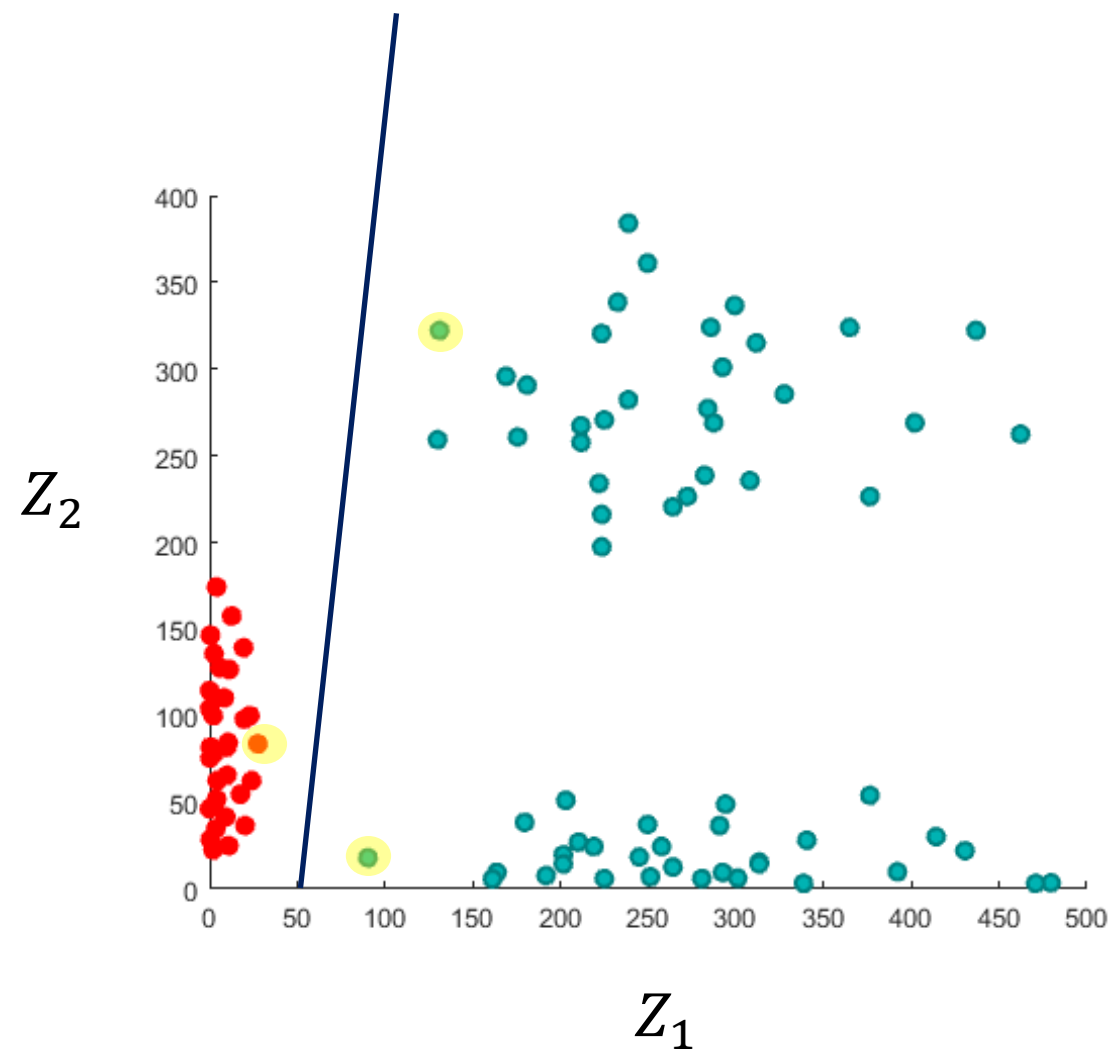

$let's\ say\ x\ is\ n \times d$
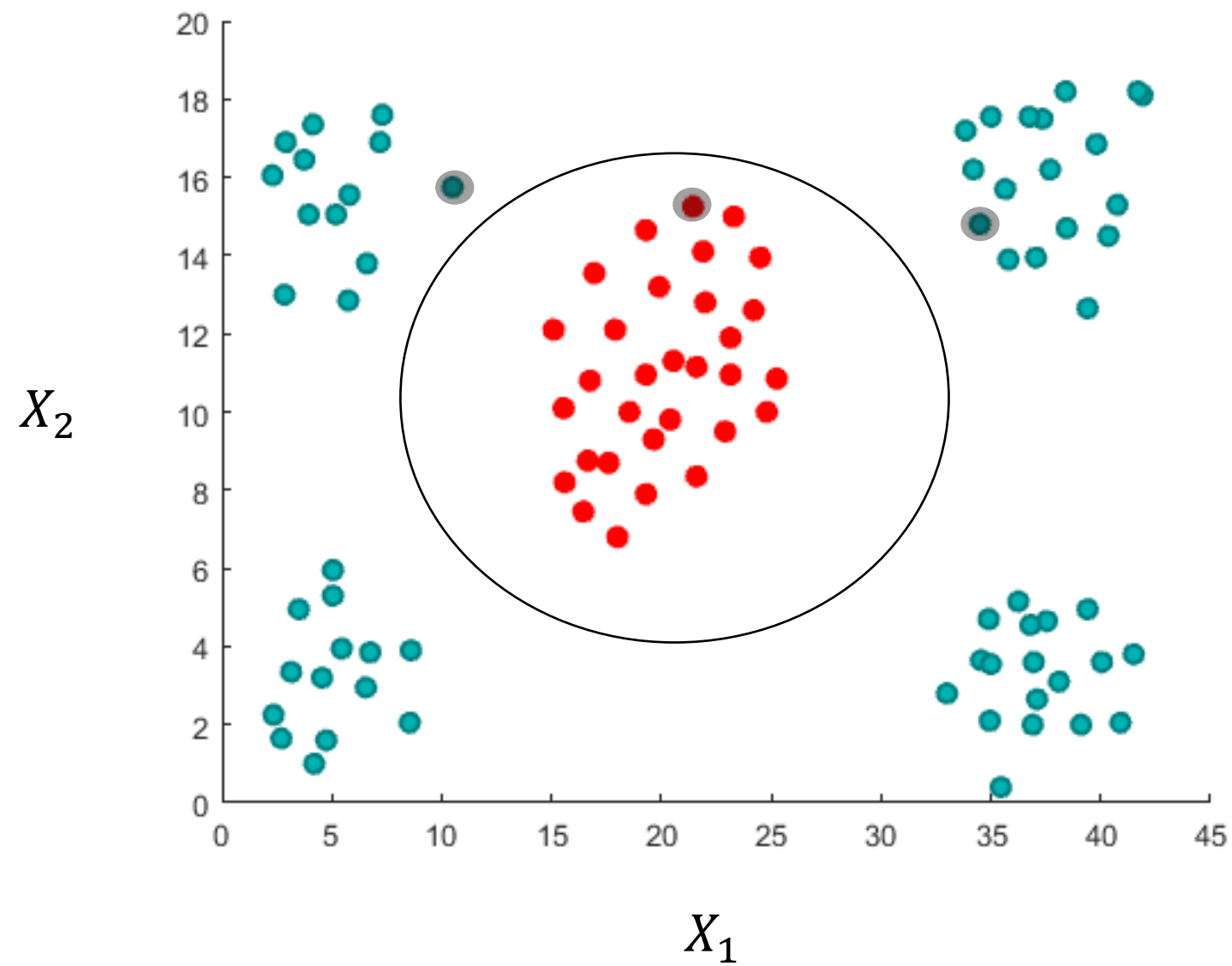$x\mathrm{x}^{\mathrm{T}}\ will\ be\ n \times n$

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