Lecture 13. Support Vector Machine

Xin Chen

These slides are based on slides from Mahdi Roozbahani

Outline

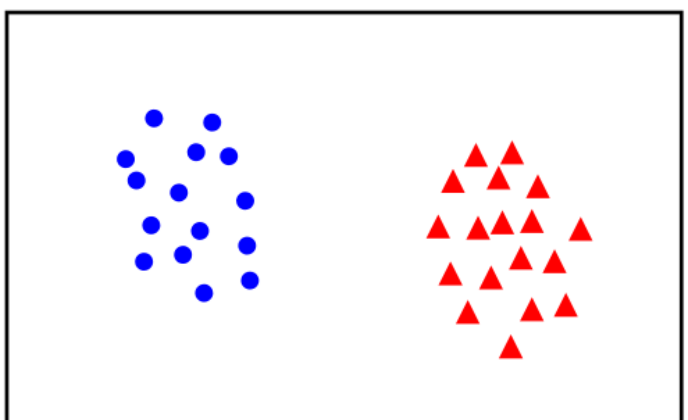
- Precursor: Linear Classifier and Perceptron
- Support Vector Machine
- Parameter Learning

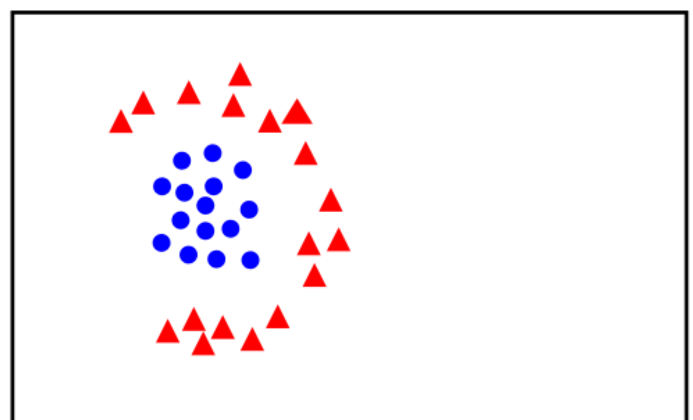
Binary Classification

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

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

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

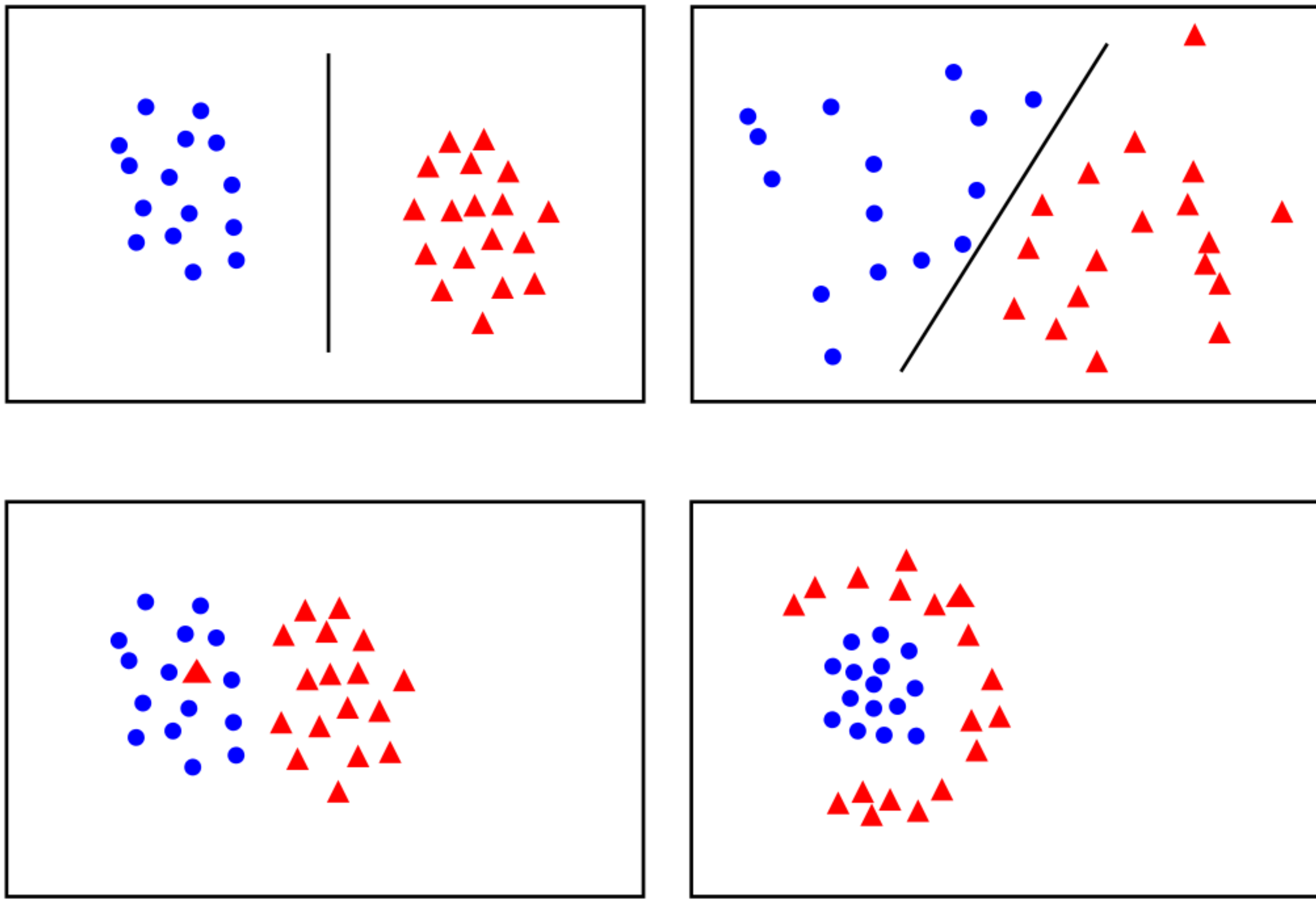




Linear Separability

linearly separable

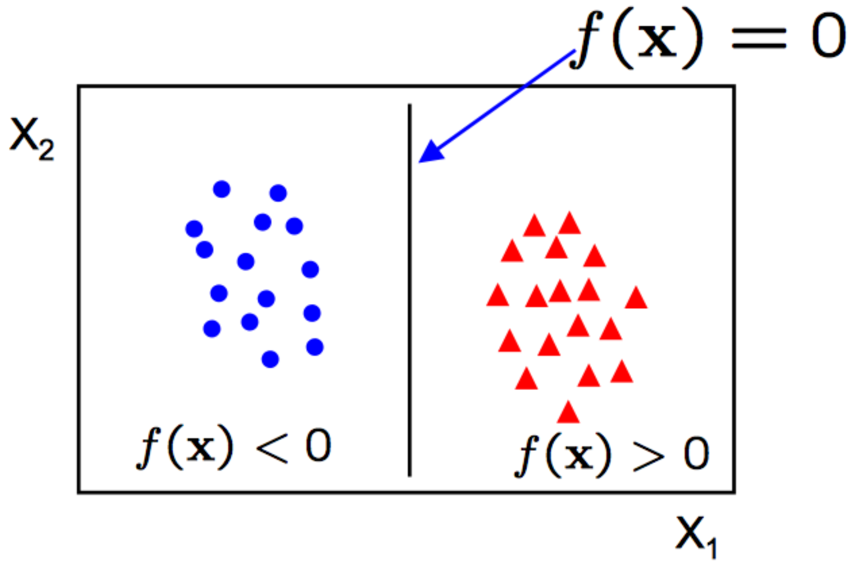
not linearly separable



Linear Classifier

A linear classifier has the form

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

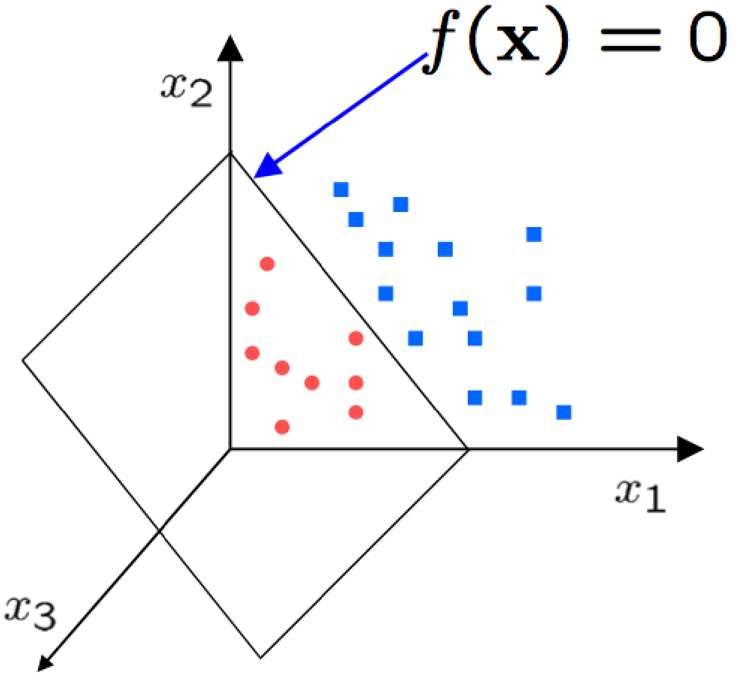


- in 2D the discriminant is a line
- θ is the normal to the line, and θ_0 e bias
- θ is known as the weight vector

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

The Perceptron Classifier

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

$$f(x_i) = x_i\theta$$ Bias is inside θ now

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

Perceptron Algorithm:

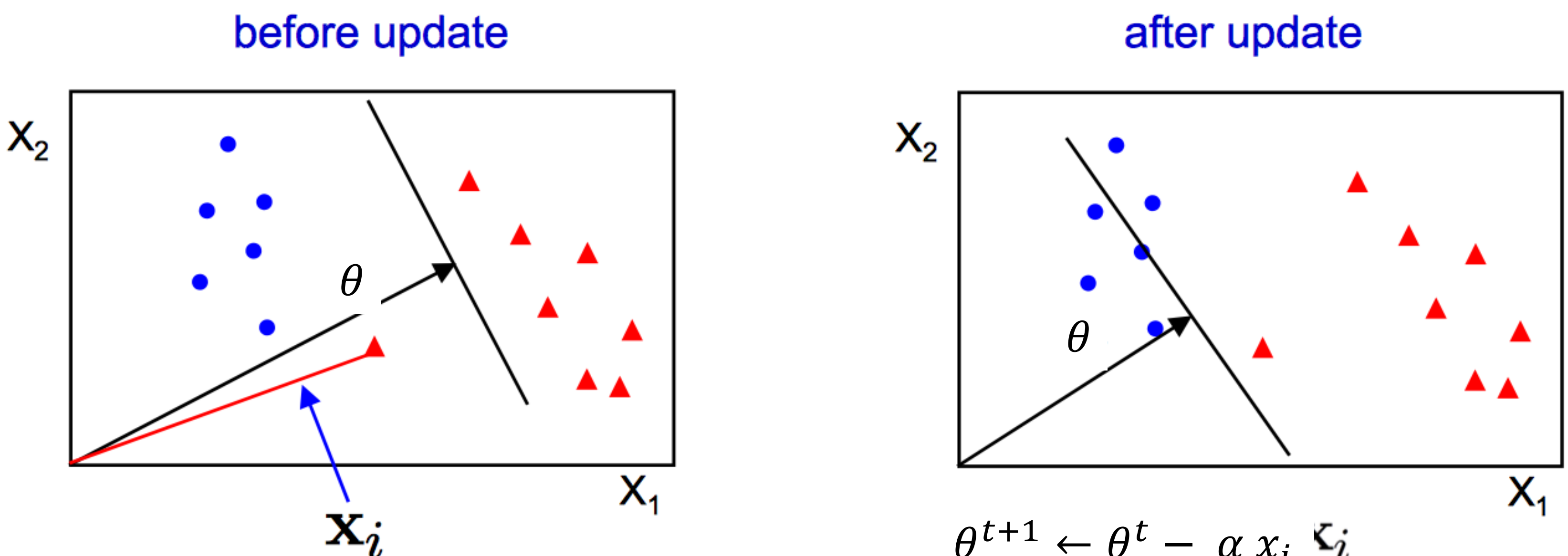
Misclassified

Ex. $y_i f(x_i) < 0$

actual predicted

- Initialize $\theta = 0$
- Go through each data point $\{x_i, y_i\}$
- If x_i is misclassified then $\theta^{t+1} \leftarrow \theta^t + \alpha\, sign[f(x_i)]x_i$
- Until all datapoints are correctly classified

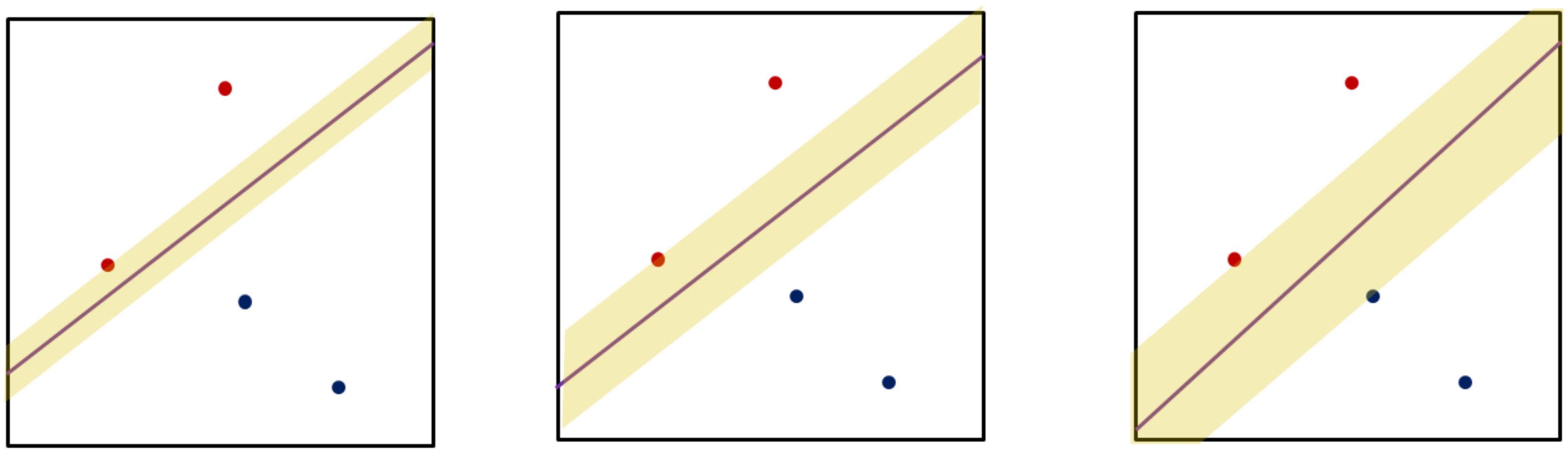
- Initialize θ =0
- Go through each datapoint $\{x_i, y_i\}$
- If x_i is misclassified then $\theta^{t+1} \leftarrow \theta^t + \alpha\, sign(\, f(x_i)\,) x_i$
- Until all datapoints are correctly classified



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

Linear separation

We can have different separating lines



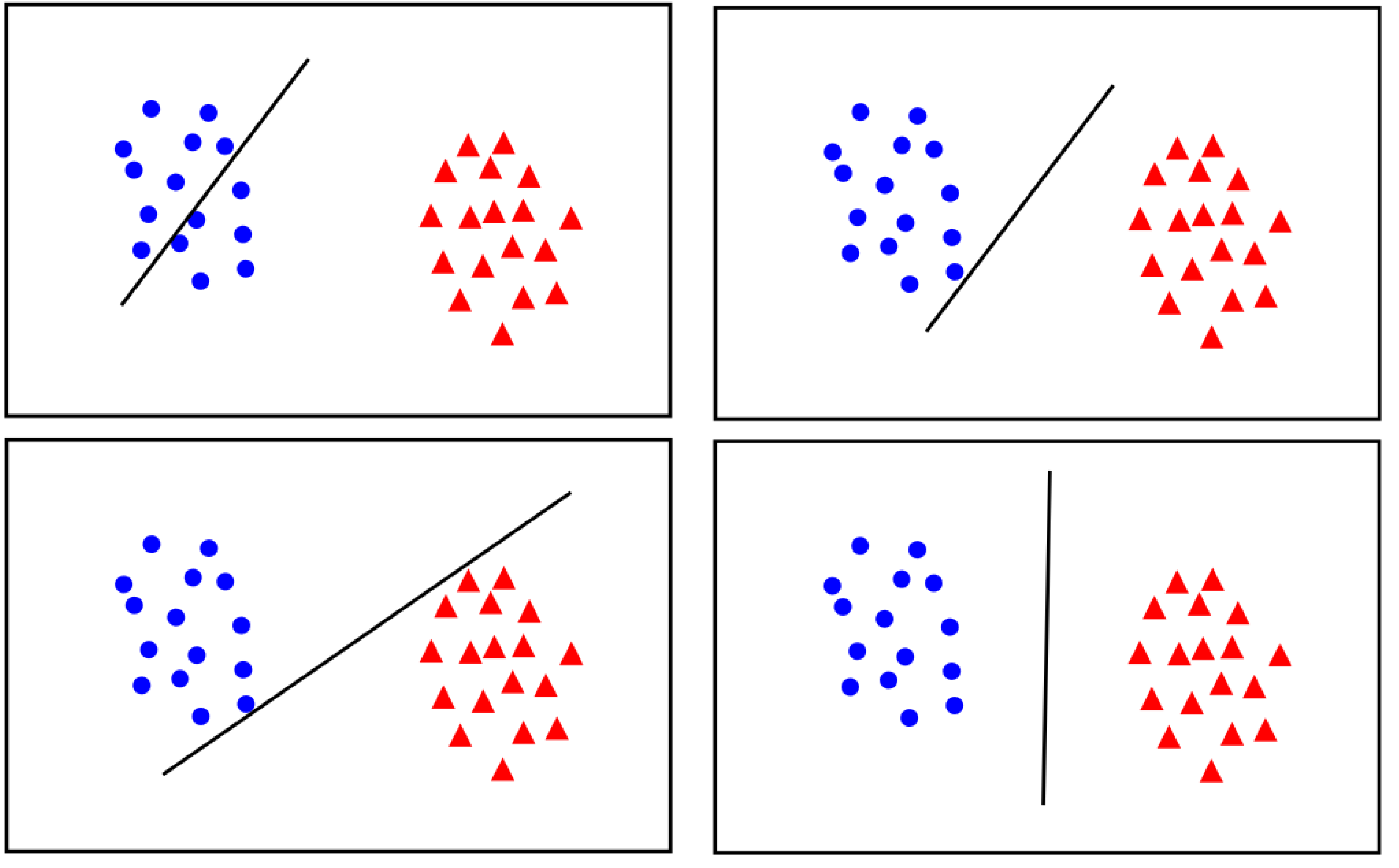
Why is the bigger margin better?

Which line is the best?

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

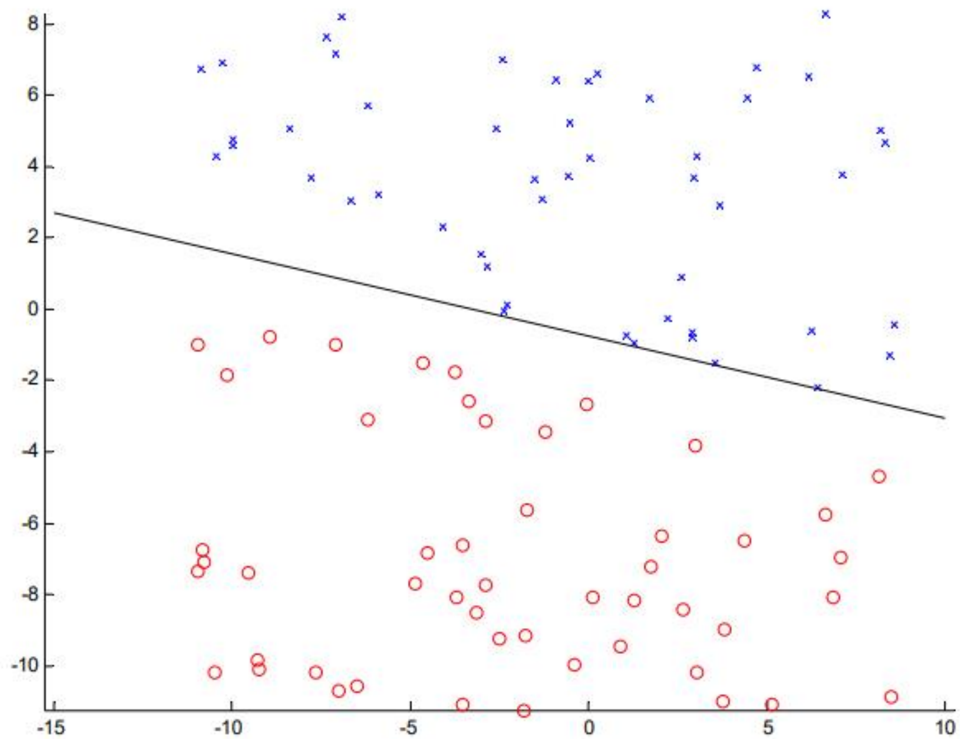
All cases, error is zero and they are linear, so they are all good for generalization.

What is the Best θ?



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

Perceptron example



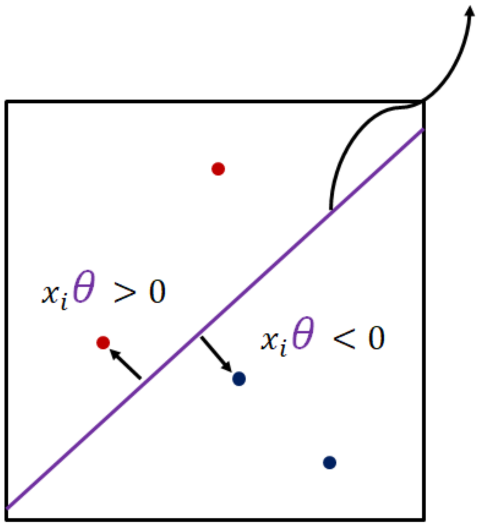
- if the data is linearly separable, then the algorithm will converge
- convergence can be slow …
- separating line close to training data
- we would prefer a larger margin for generalization (better generalization)

Outline

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

Finding θ with a **fat** margin

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



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?

$|x_i\theta| = 1$ → normalization

Let's pull out θ_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 plane $x\theta + b = 0$ where $|x_i\theta + b| = 1$

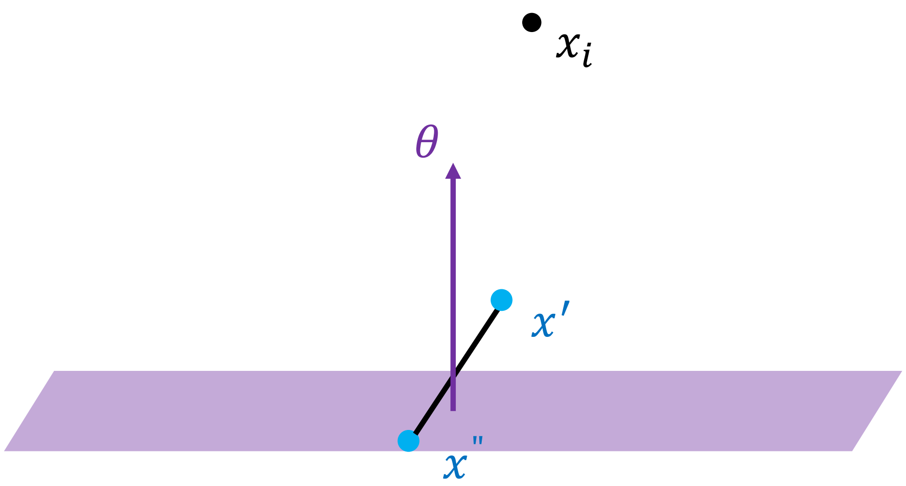
The vector θ is perpendicular to the decision boundary plane.

You should ask me why?

Consider x' and x'' on the plane

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

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



What is the distance?

What is the distance between x_i and the plane?

Let's take any point x on the plane:

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

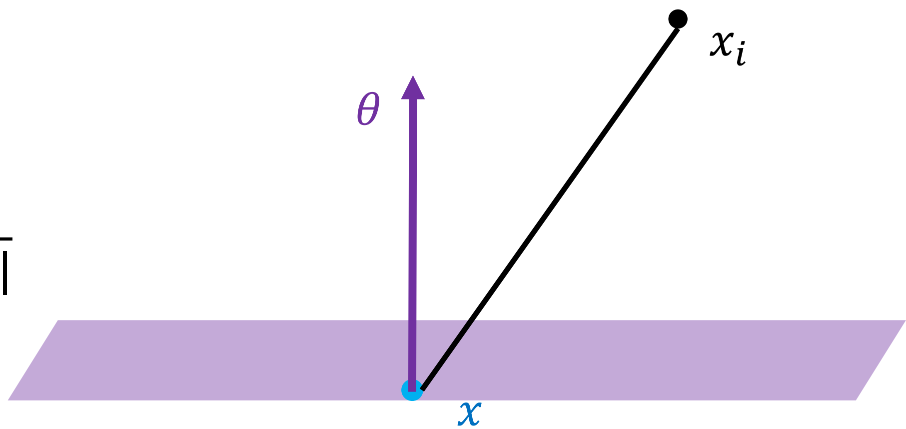
To project the vector, we need to normalize θ to get the unit vector.

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

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

$$= \frac{1}{||\theta||} |(x_i\theta + b - x\theta - b)| \qquad = \frac{1}{||\theta||}$$

The margin



Now we need to maximize the margin

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

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)

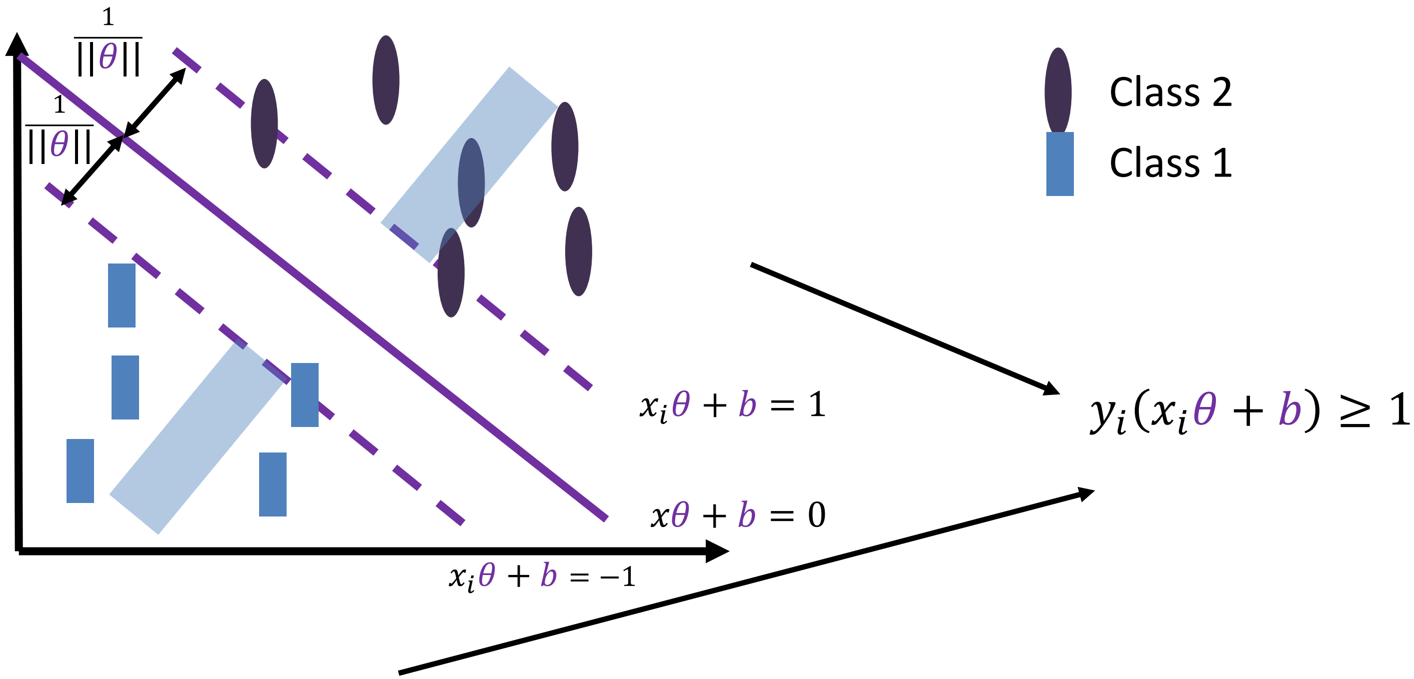
Can I write the following term to get rid of absolute value?

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

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

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

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



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

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

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

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

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

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

$\gamma = lagrange\ multiplier$

KKT → $g(x)\gamma = 0$ $\Rightarrow \begin{cases} g(x) > 0, & \gamma = 0 \\ g(x) = 0, & \gamma > 0 \end{cases}$

$w.r.t\ \ Maximize\ \gamma \geq 0$

Lagrange formulation

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

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

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

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

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

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

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

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

Let's substitute these in the Lagrangian:

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

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

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

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

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

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

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

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

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”

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

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

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

Linear term

Quadratic coefficients

Pass these to a quadratic programming package

Subject to

$$y^T \alpha = 0$$

Linear equality constraint

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

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

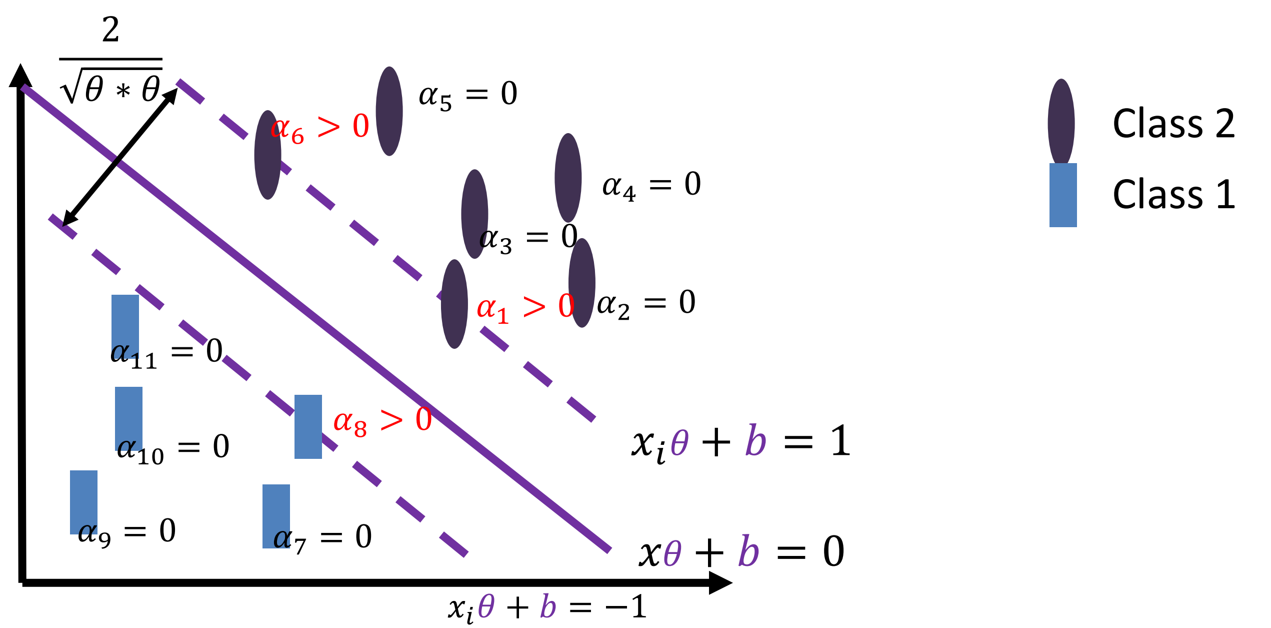
Quadratic programming will give us α

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

KT 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

Testing

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

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

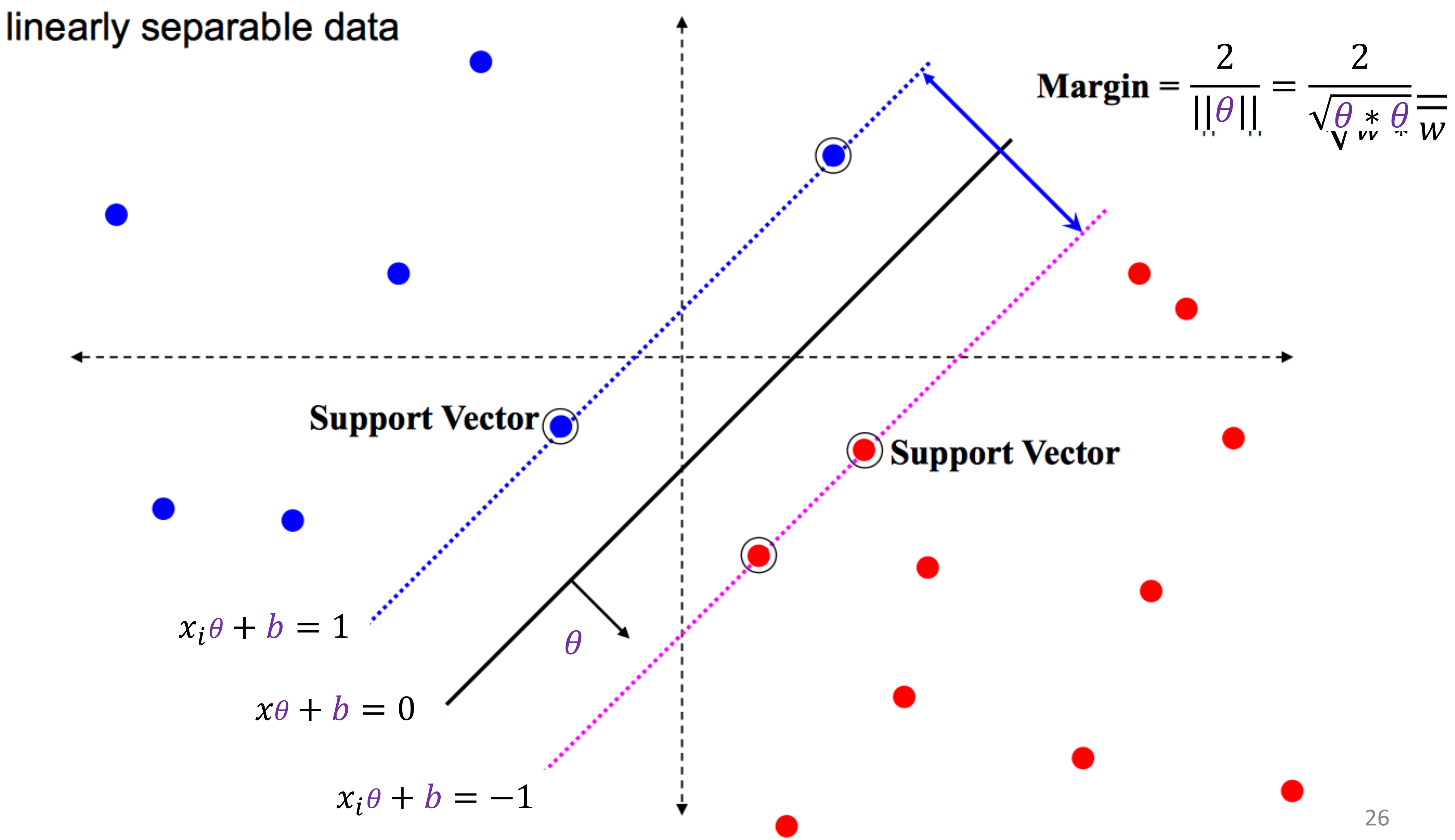
For a new test point s

Compute:

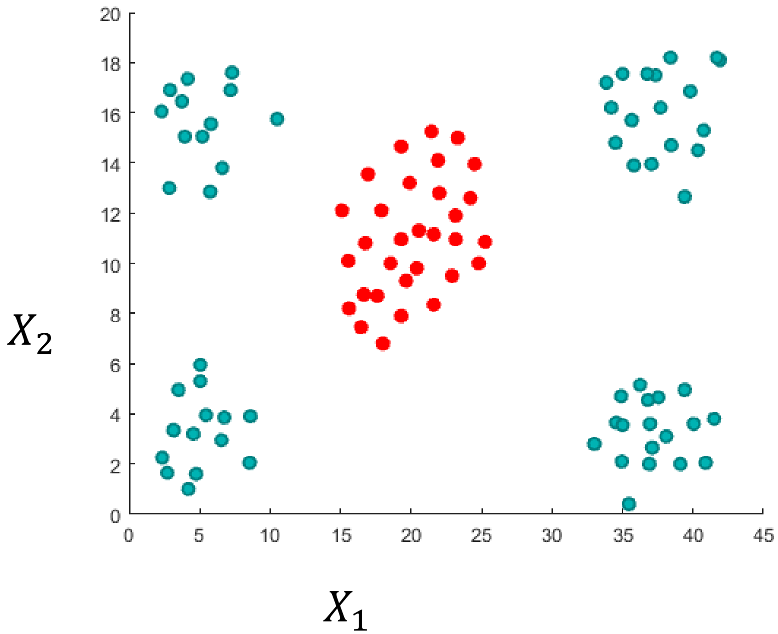
$$\mathrm{s}\theta + \mathrm{b} = \sum_{x_i \, in \, SV} \alpha_i y_i x_i s^T + b$$

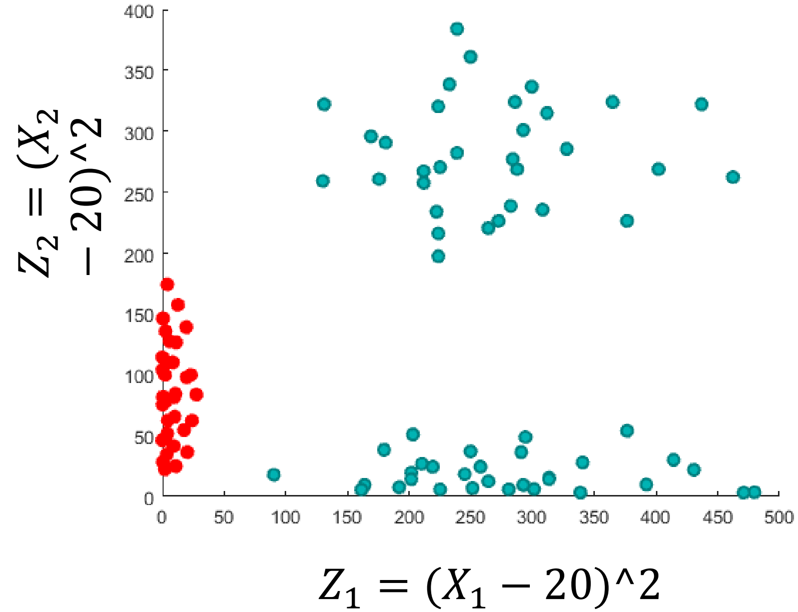
Classify s as class 1 if the result is positive, and class 2 otherwise

Geometric Interpretation



From x to z space



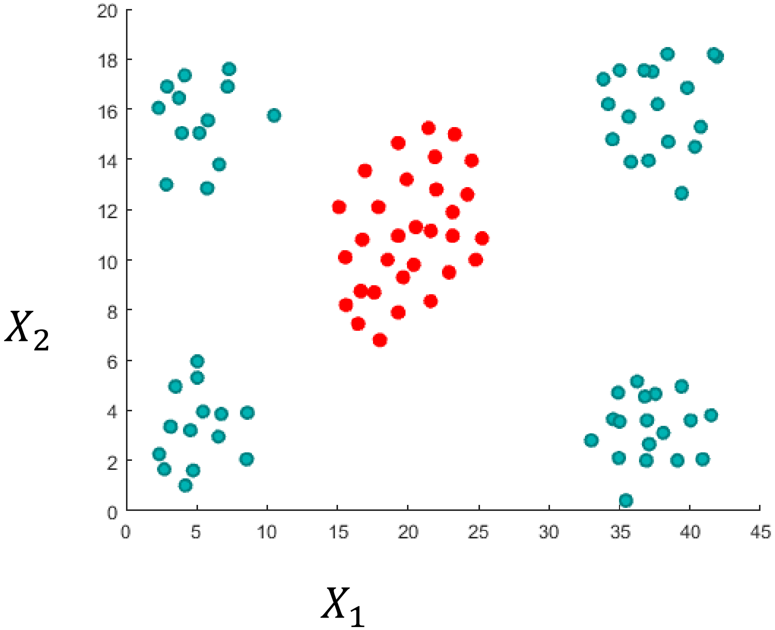


$$X \longrightarrow Z$$

$$(X_1^2, X_2^2)$$

In x space

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



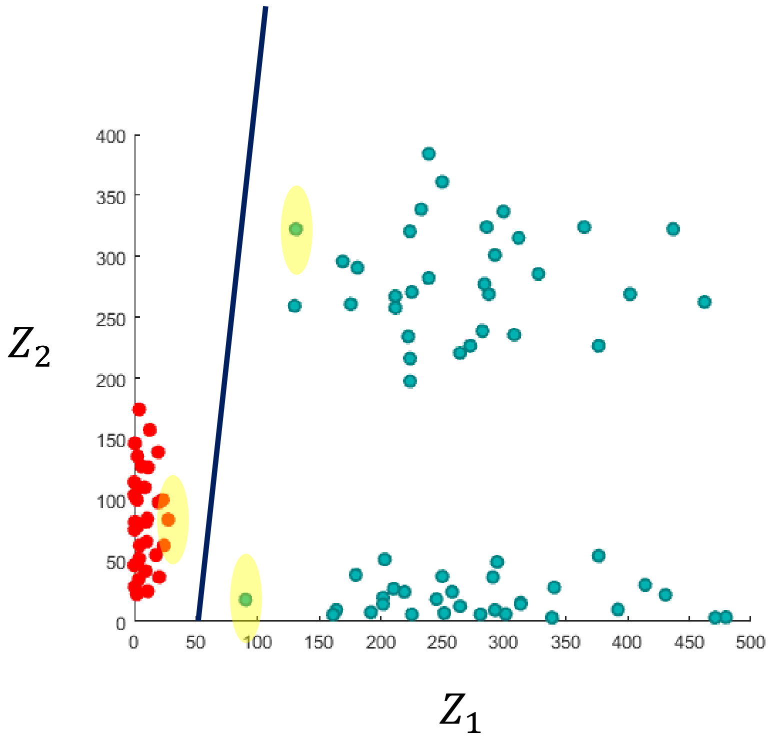
$let's\ say\ x\ is\ n \times d$

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

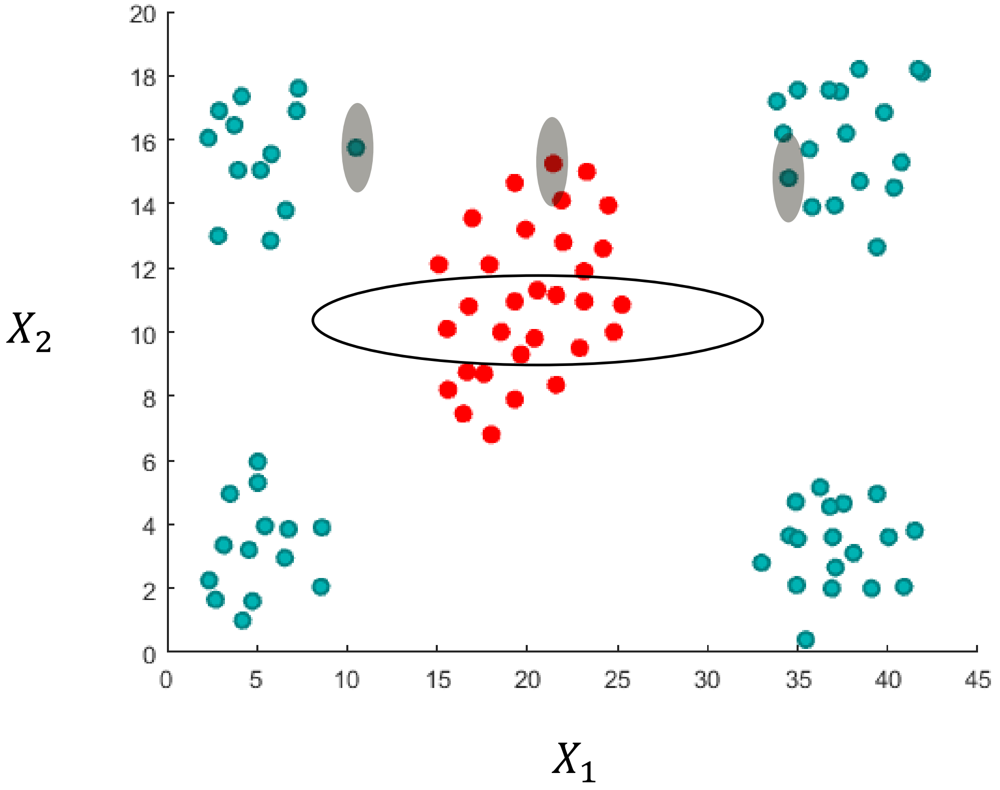
If I add millions of dimensions to x, would it affect the final size of $x\mathrm{x}^{\mathrm{T}}$?

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