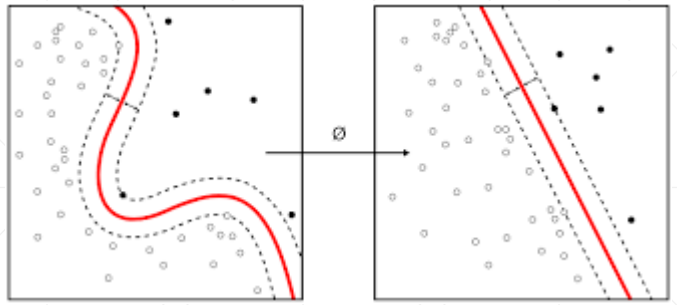


Support Vector Machine

~Abhishek Kumar

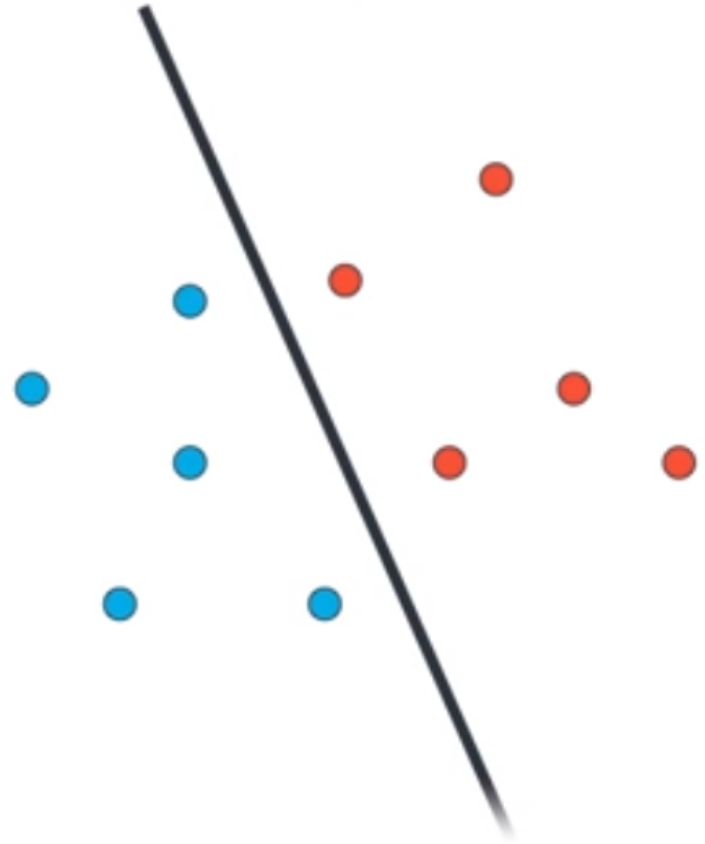
SVM

- Supervised
- Classification algorithm

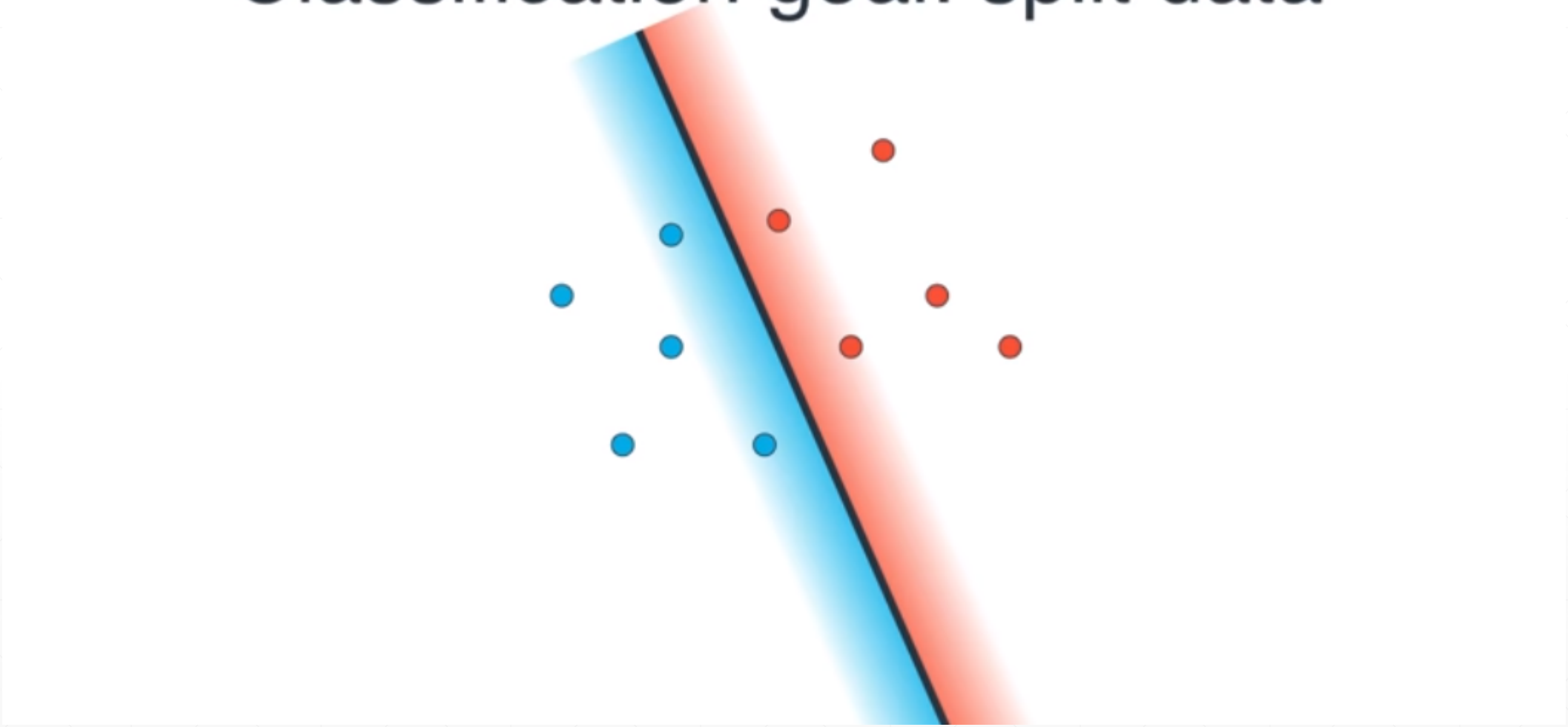
Algo 1:
Maximal Margin Classifier

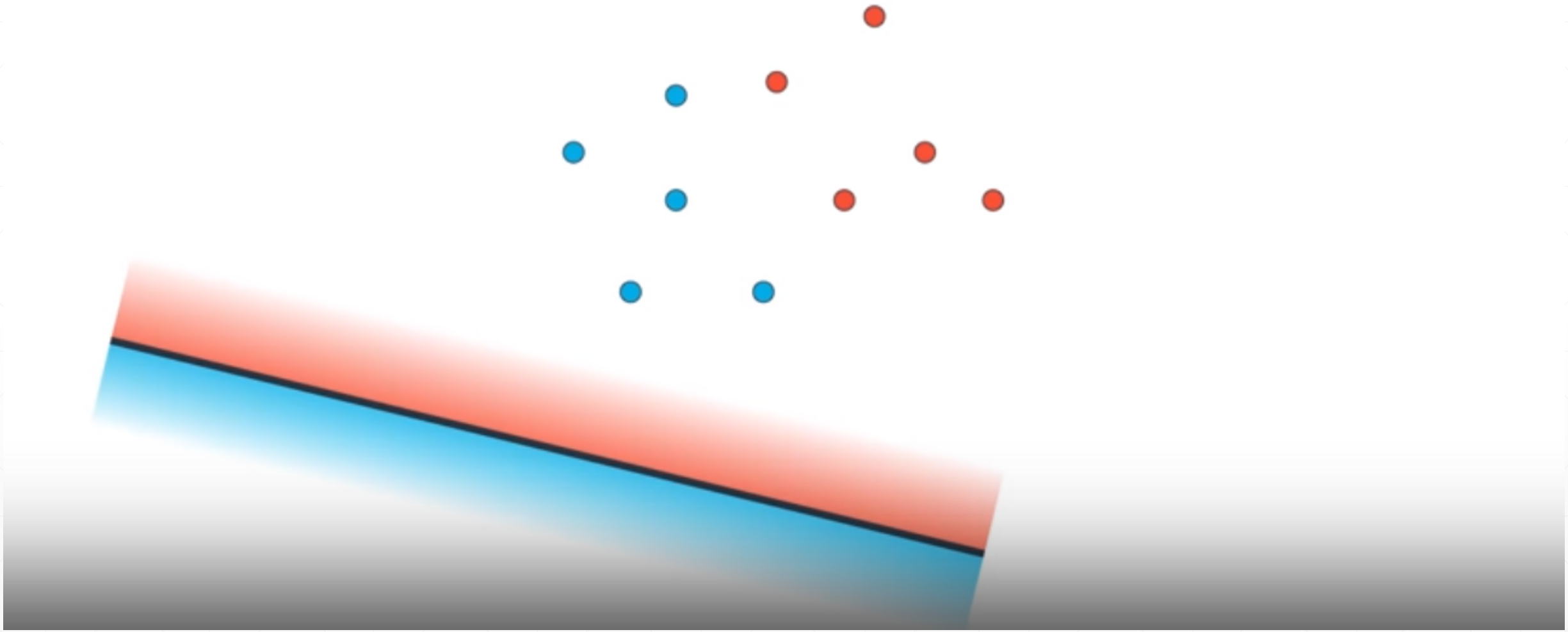
Classification goal: split data

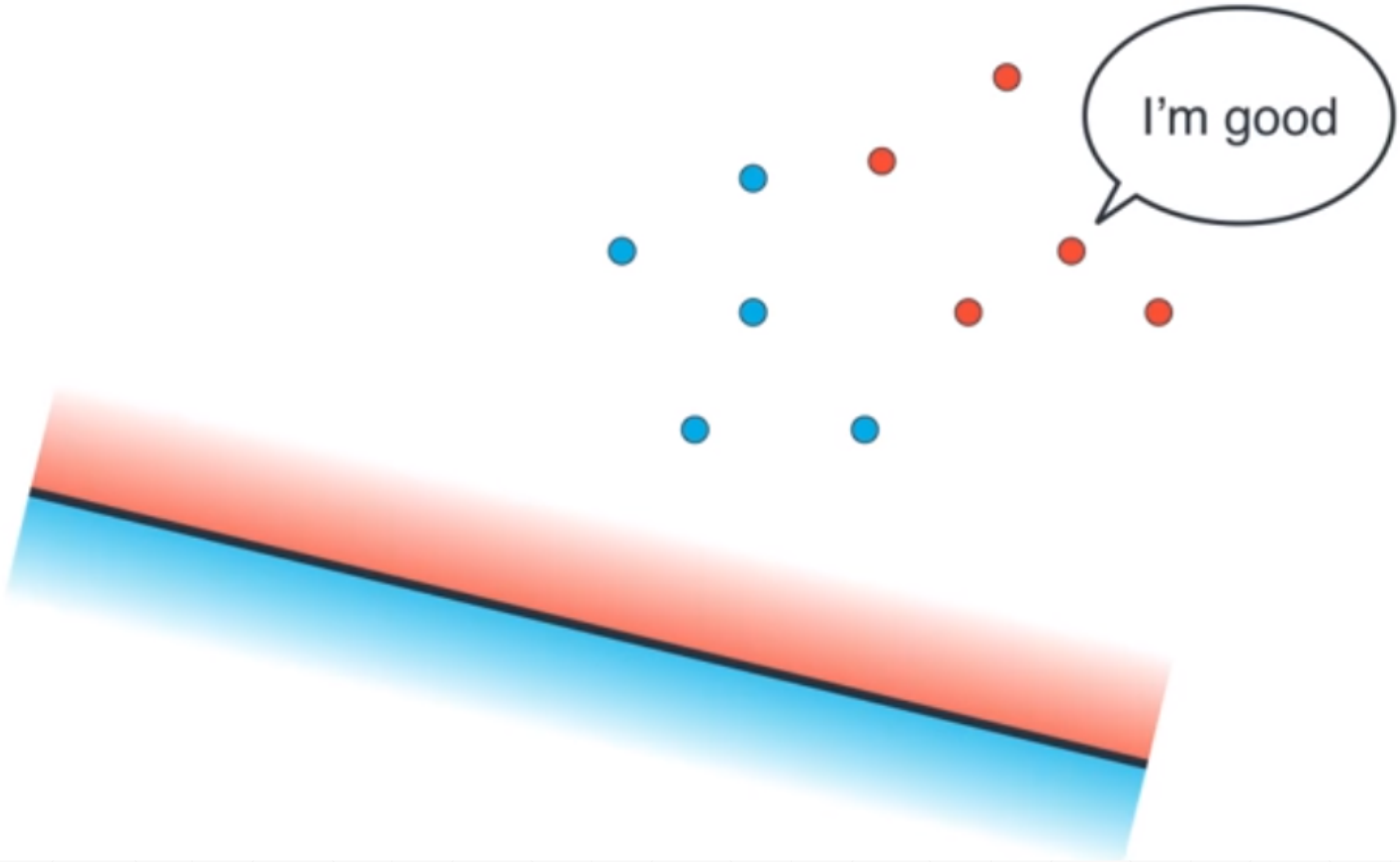
Classification goal: split data

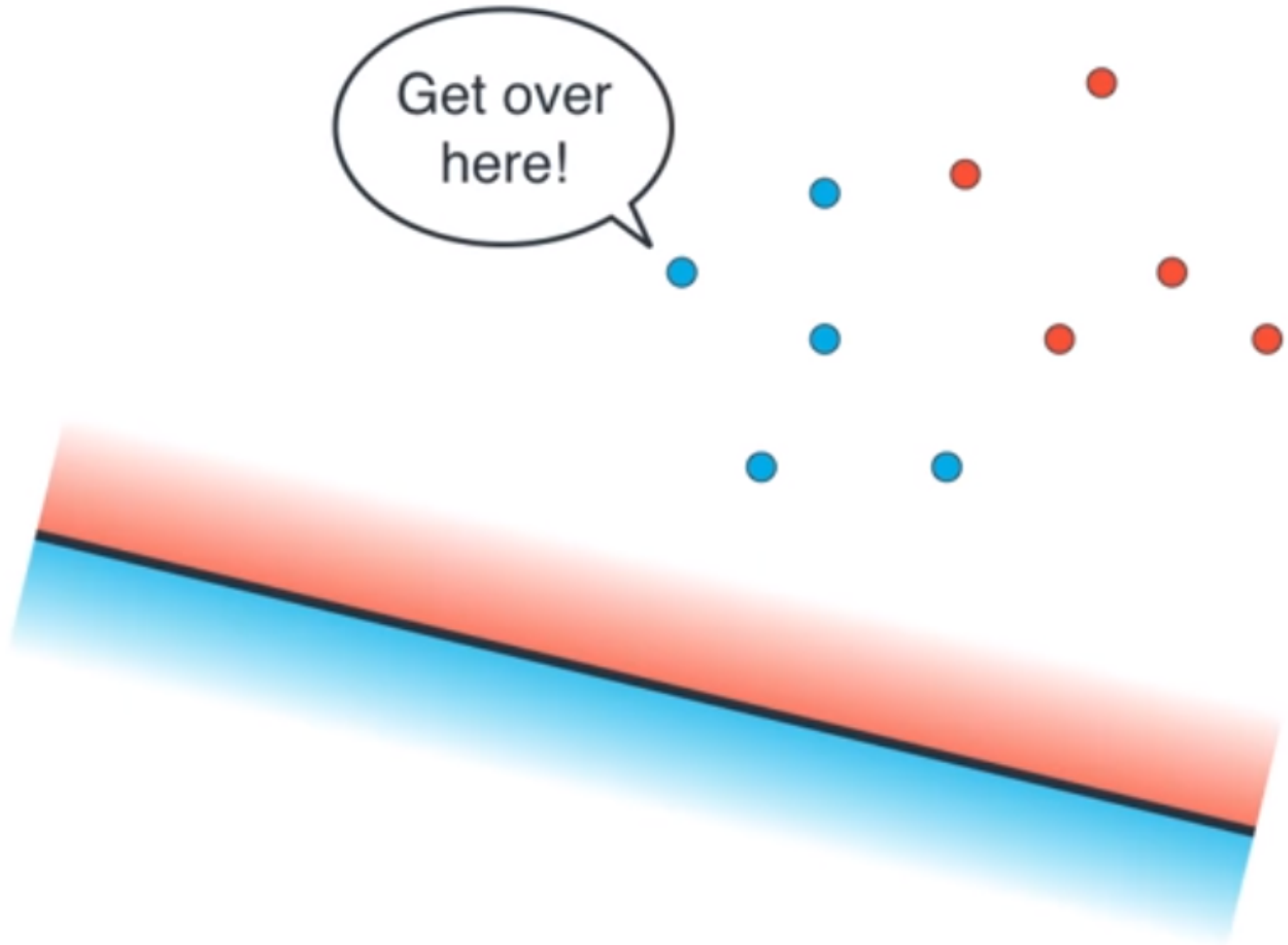


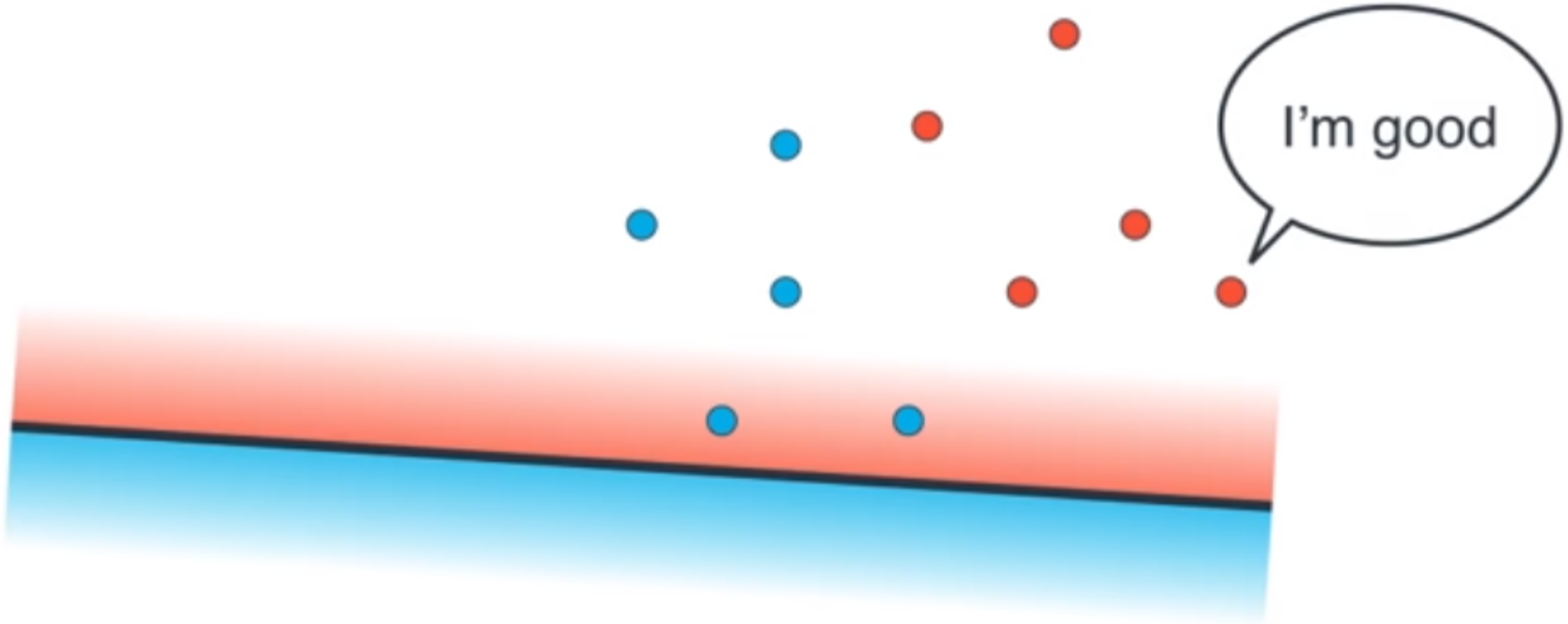
Classification goal: split data

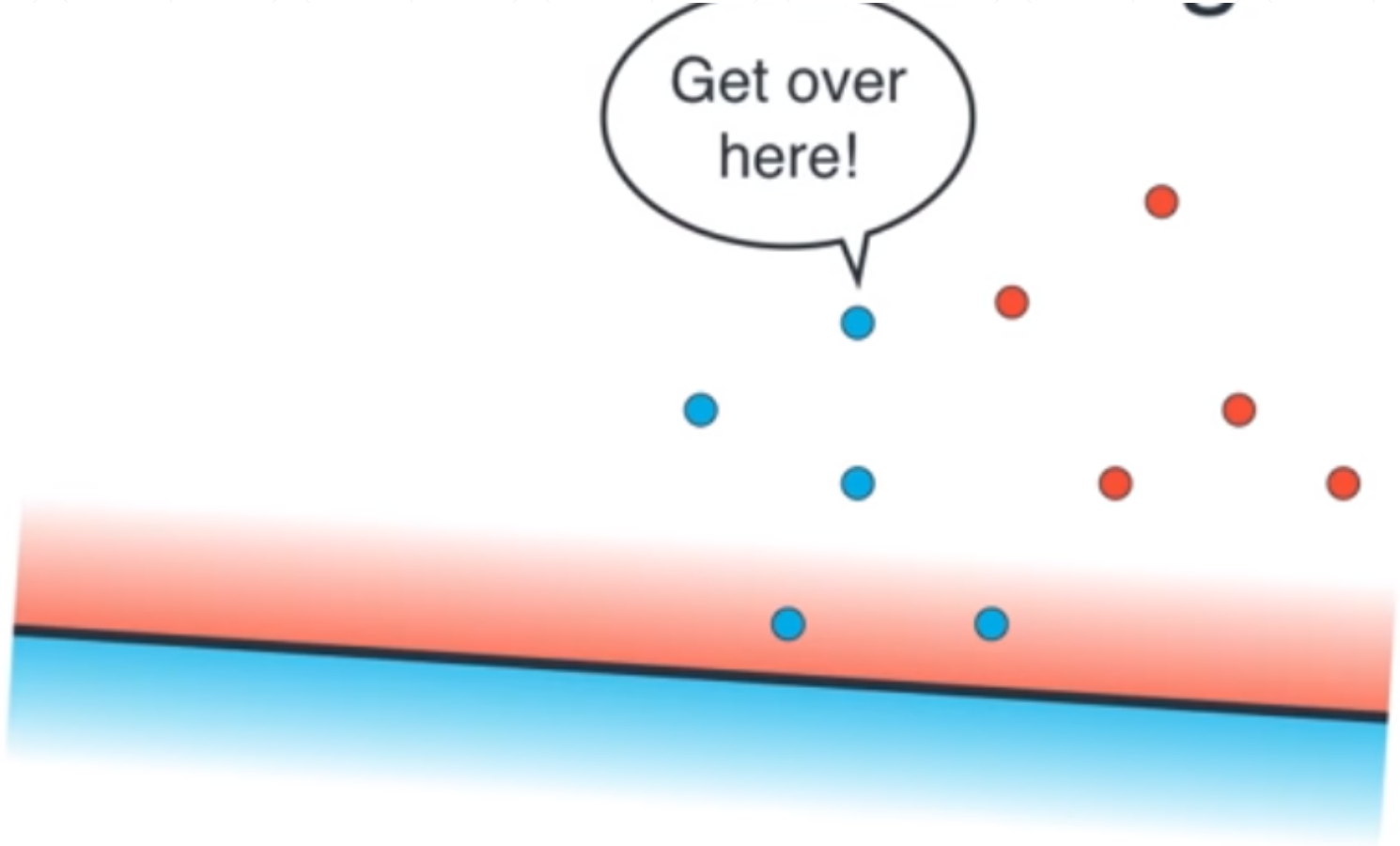


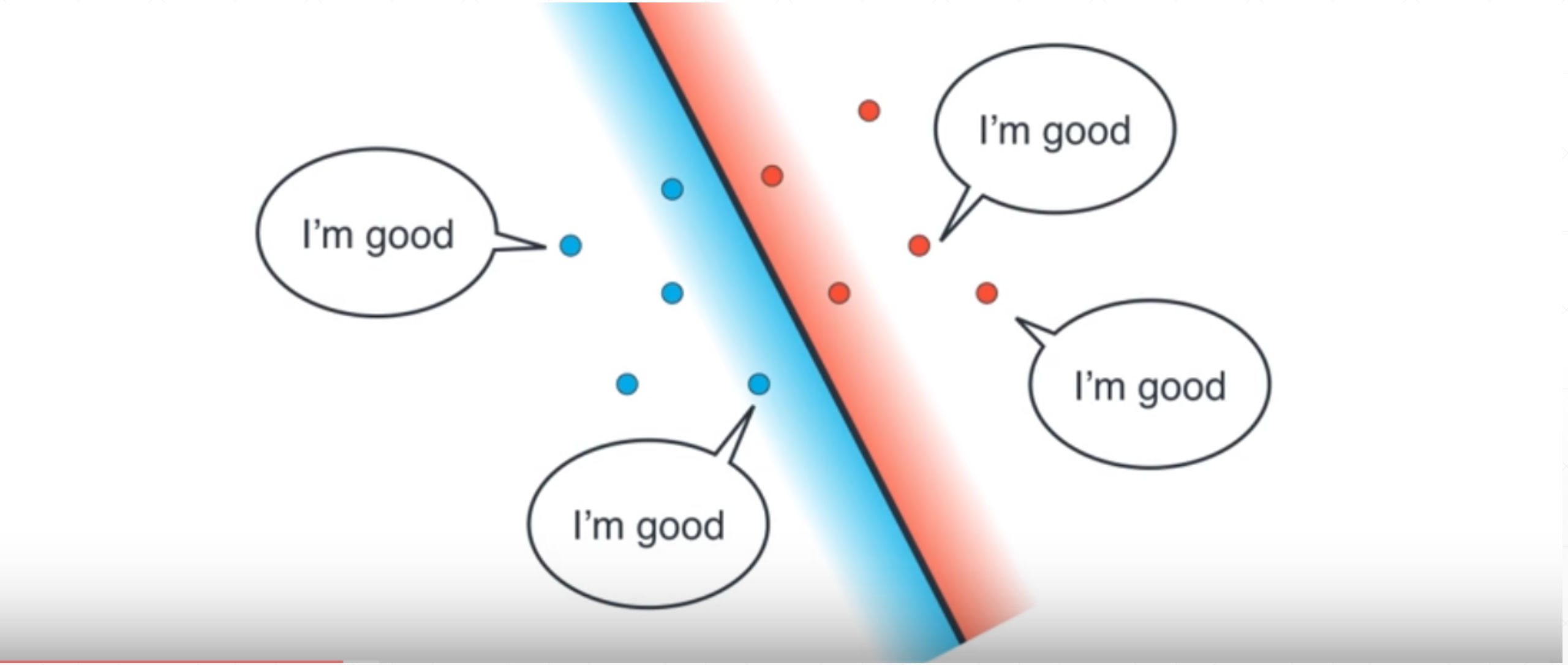



I’m good

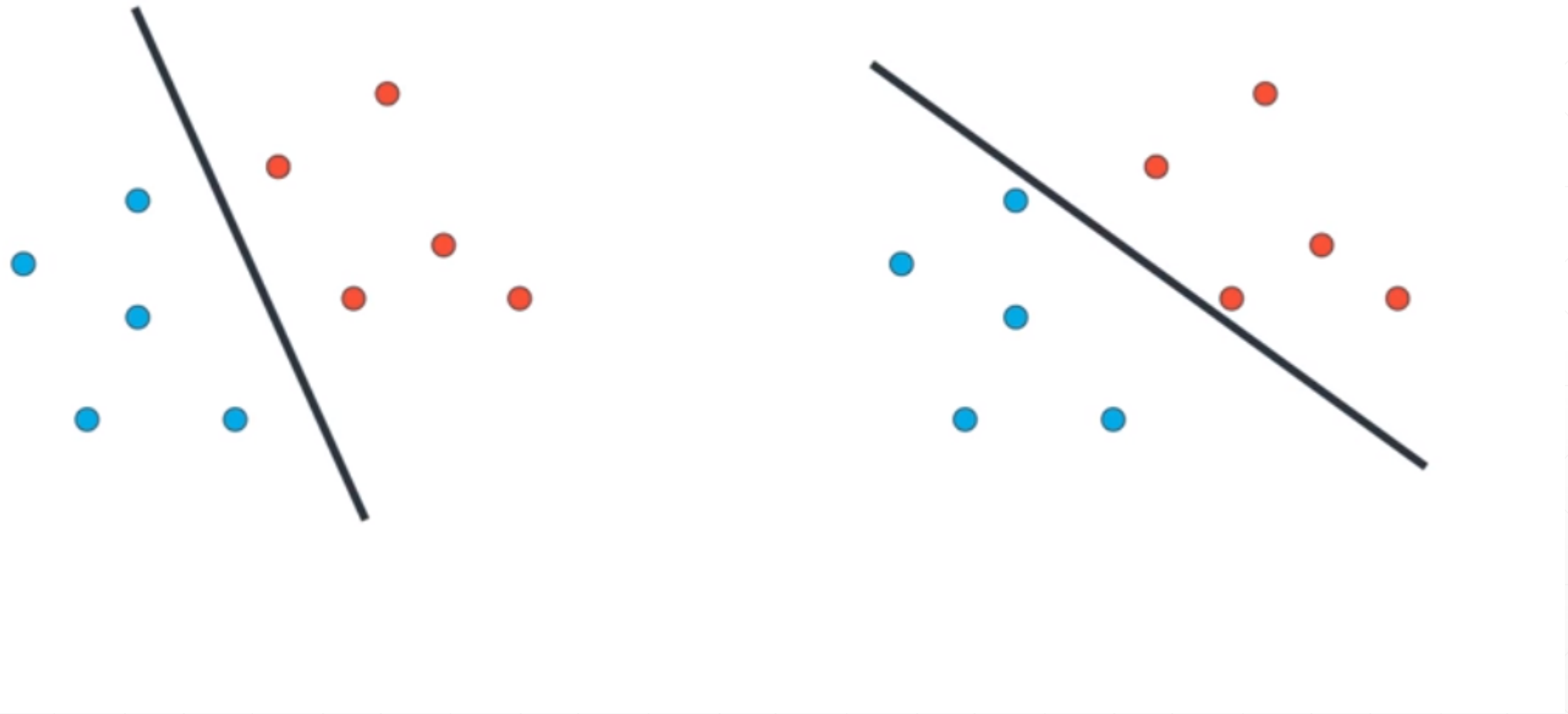

Get over here!


I’m good

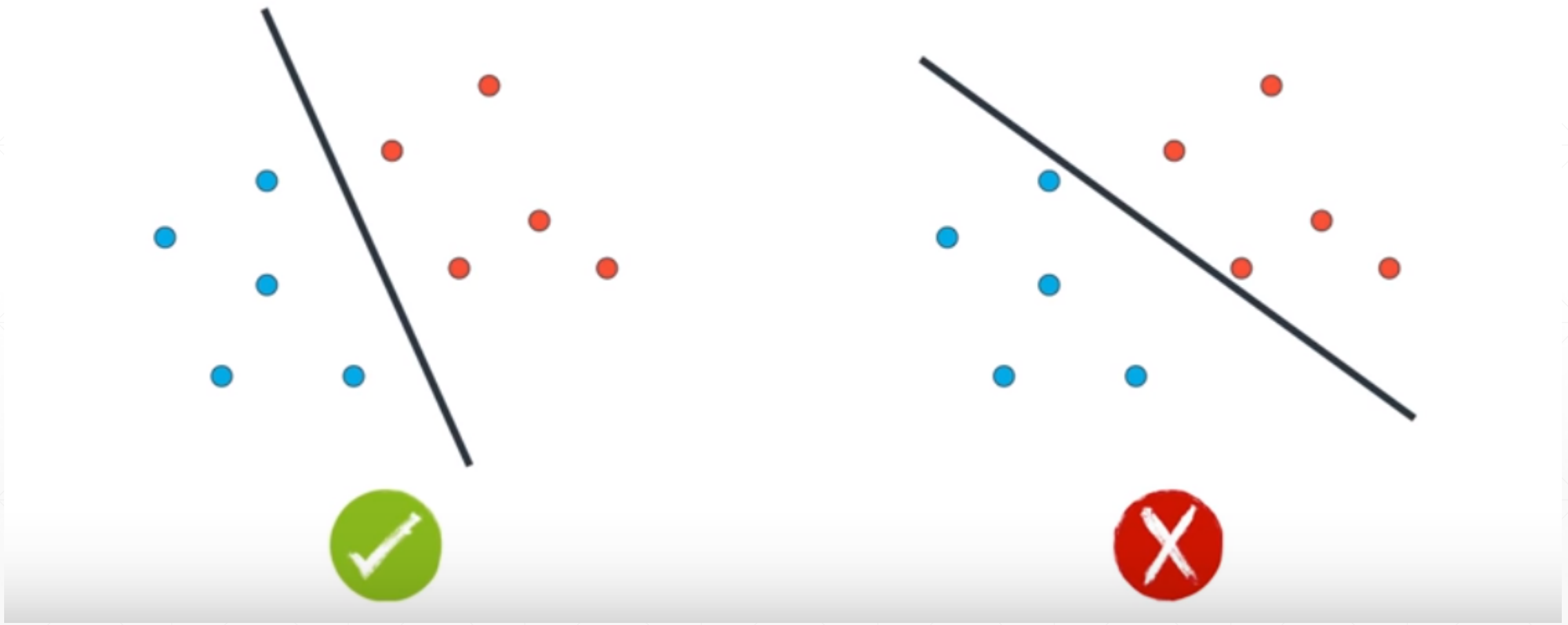

Get over
here!


I'm good
I'm good
I'm good
I'm good

Which line is better?



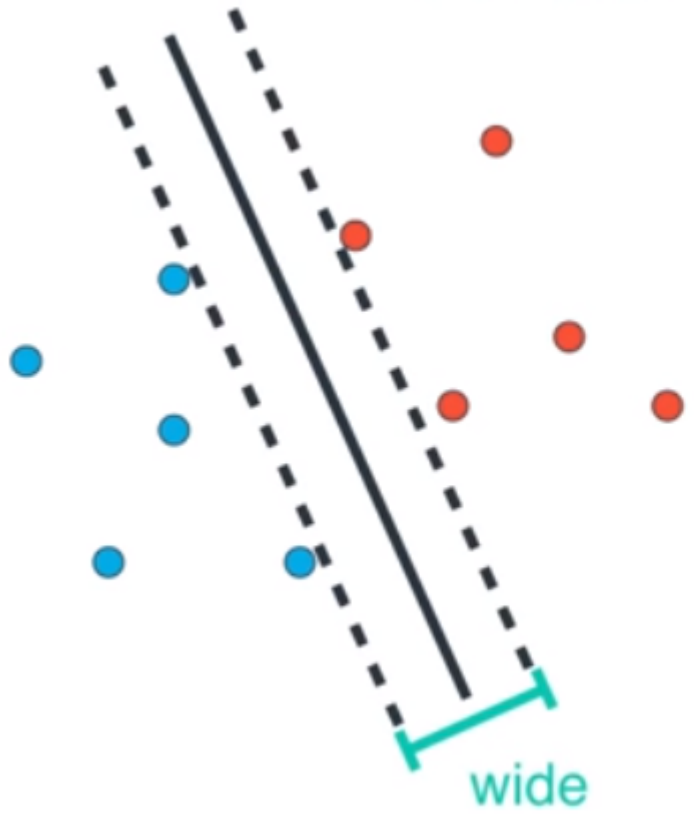
Which line is better?

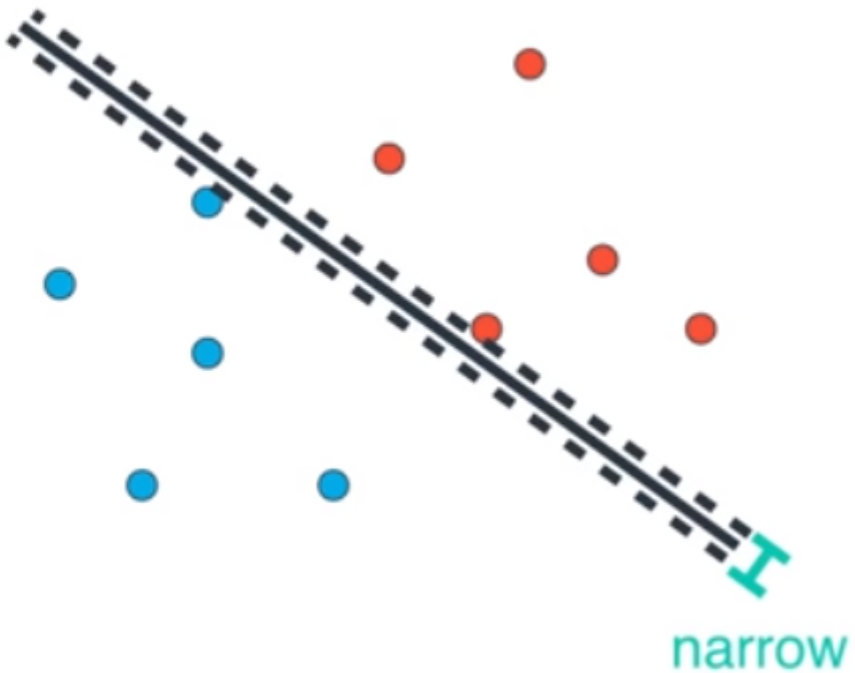


How do we tell the computer to pick the best line?

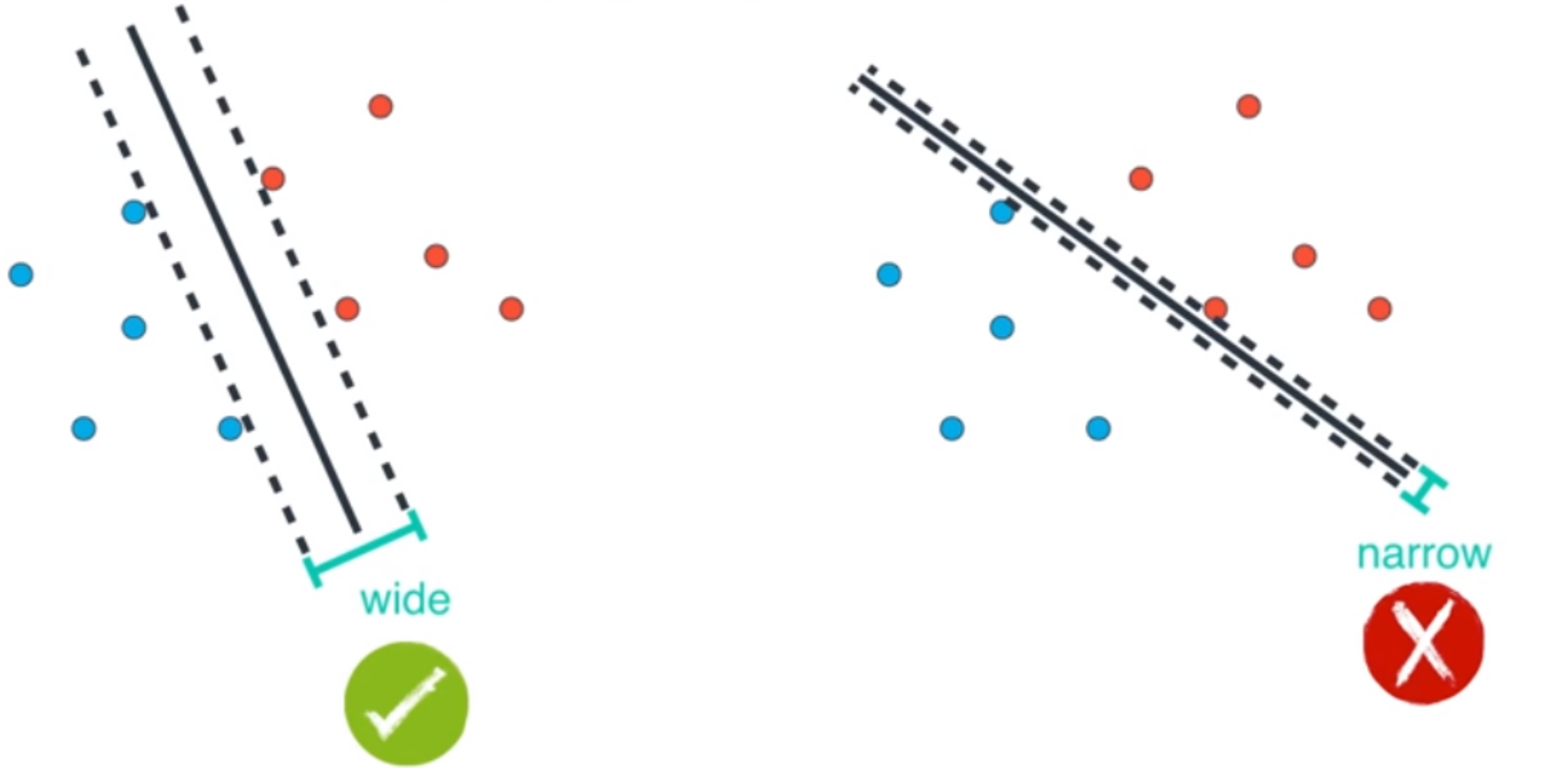
- Can we say that? We should find two lines that space apart as far as possible from each other

Which line is better?

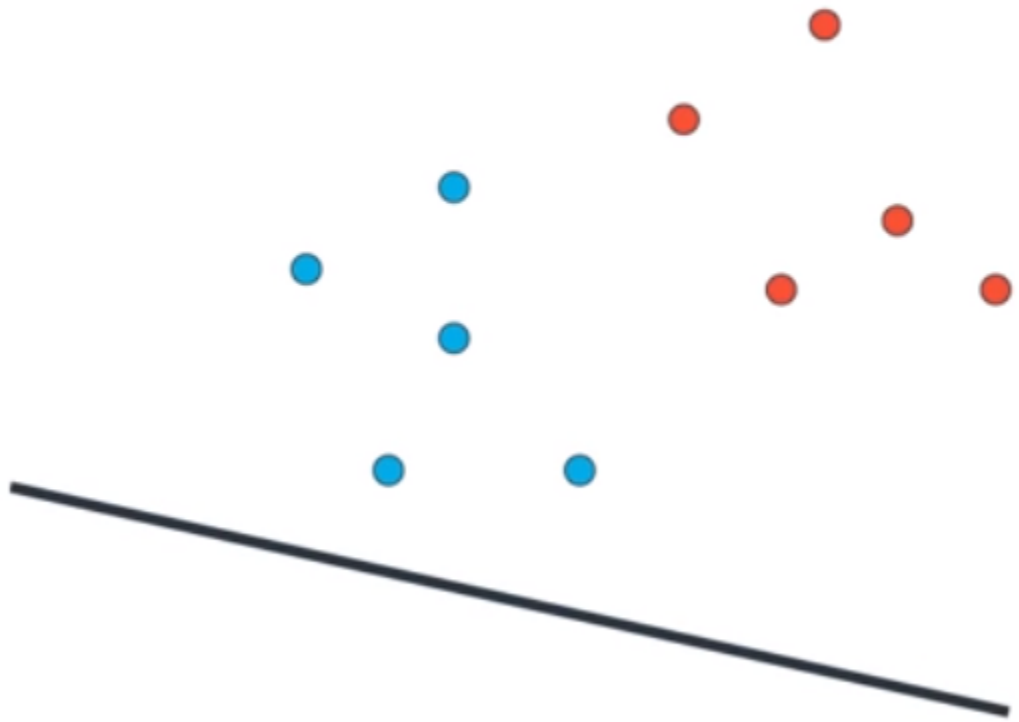




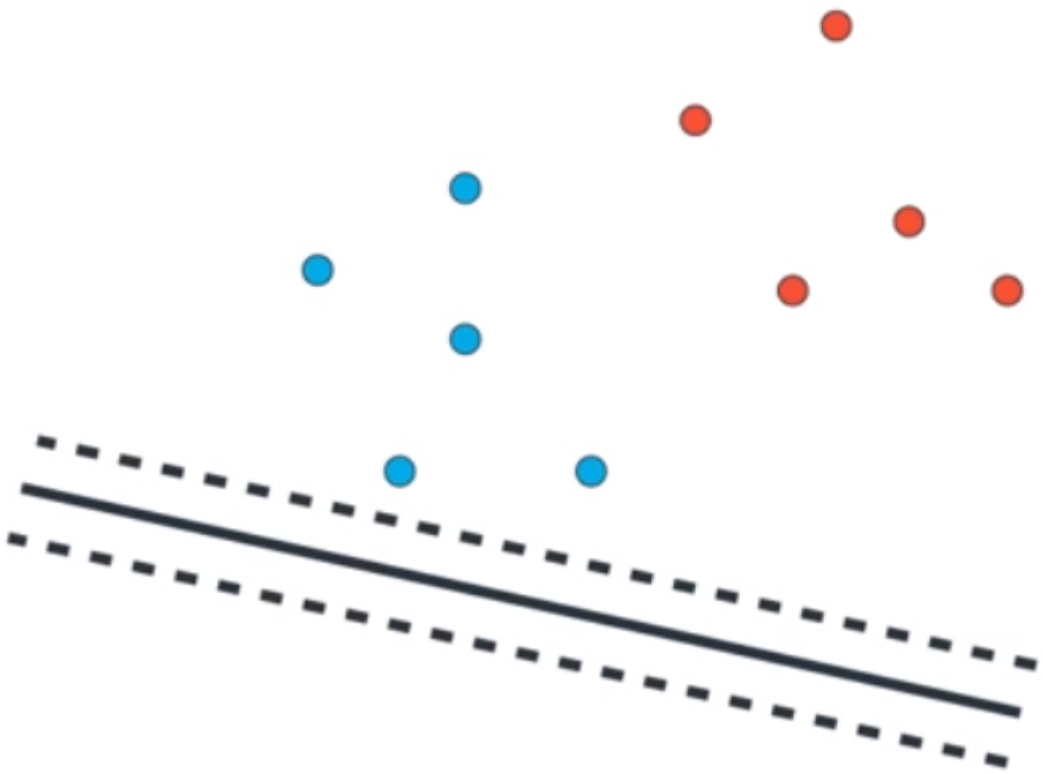
Which line is better?



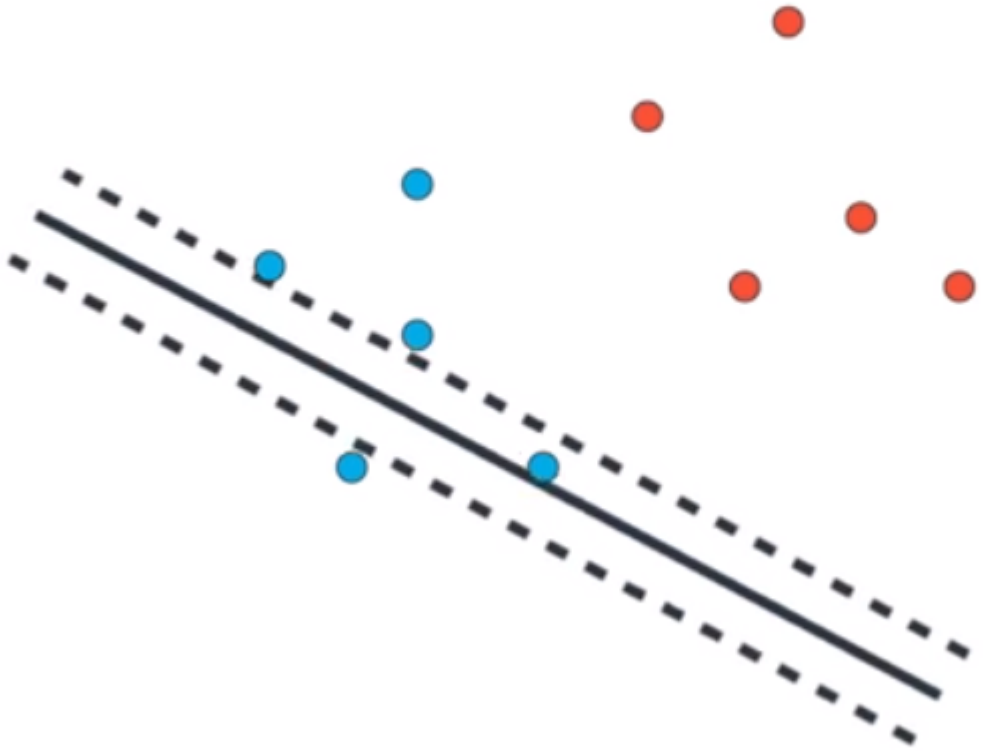
Split data - separate lines



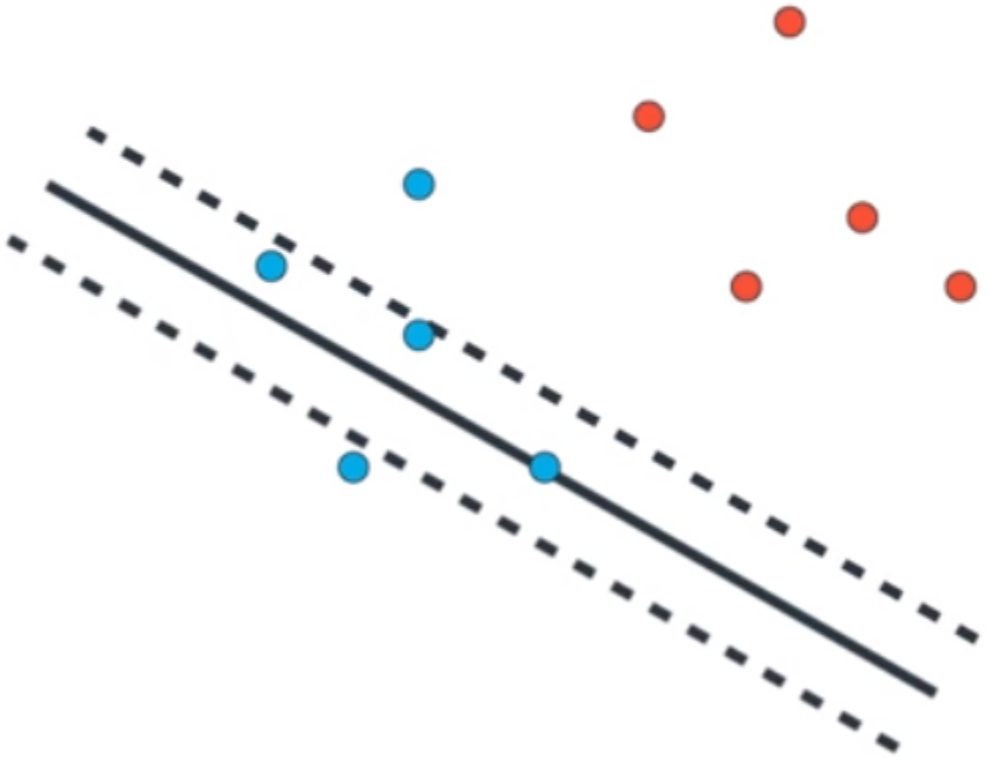
Split data - separate lines



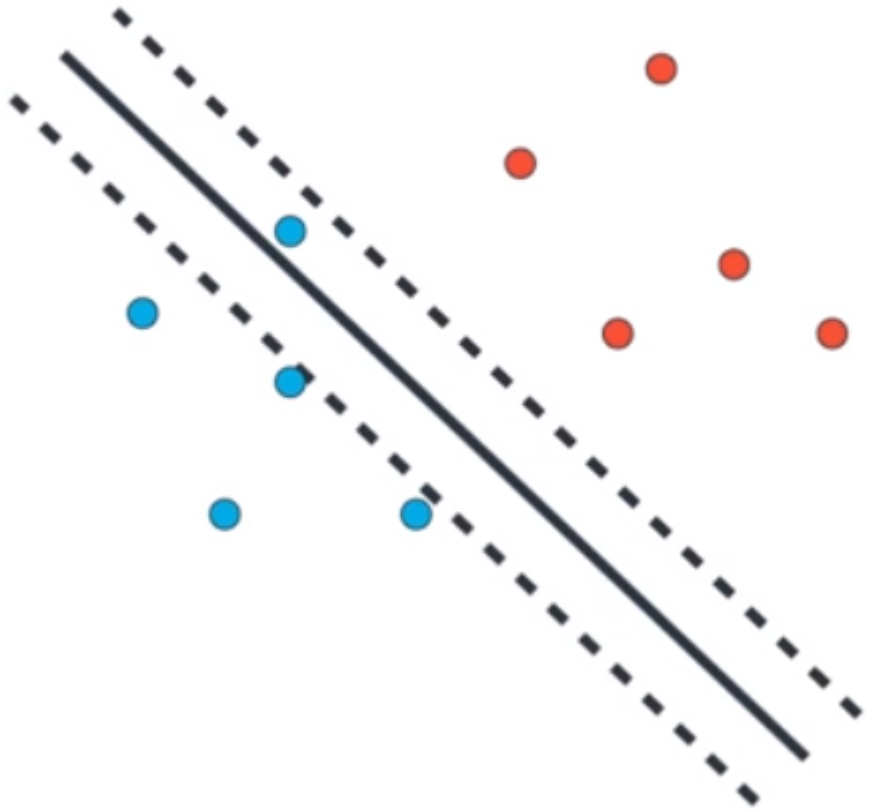
Split data - separate lines



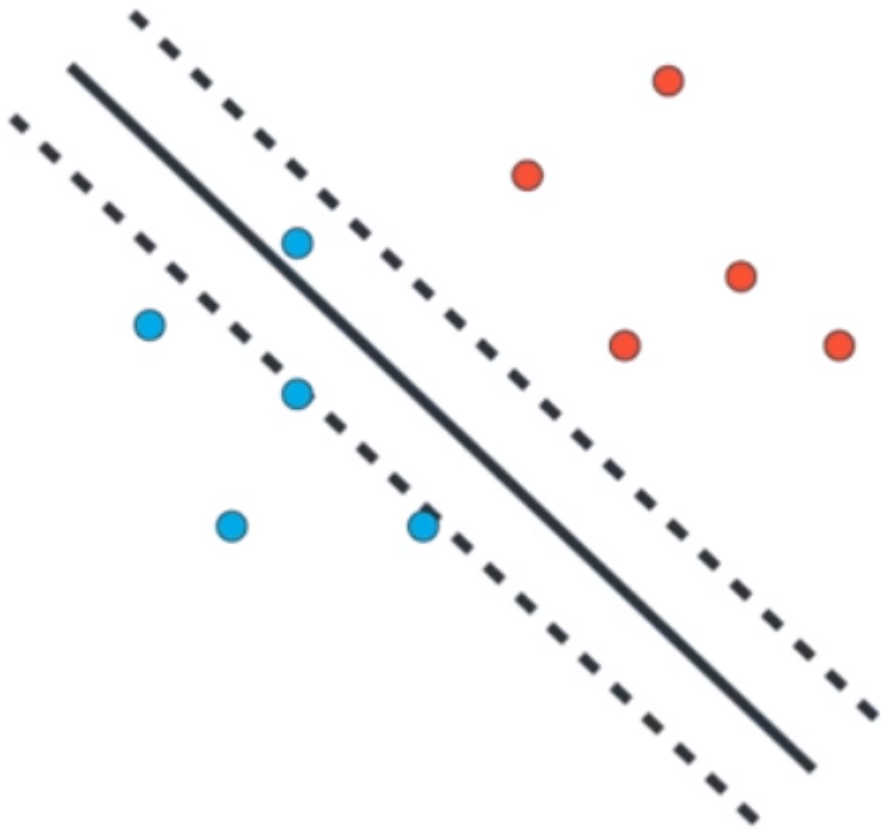
Split data - separate lines



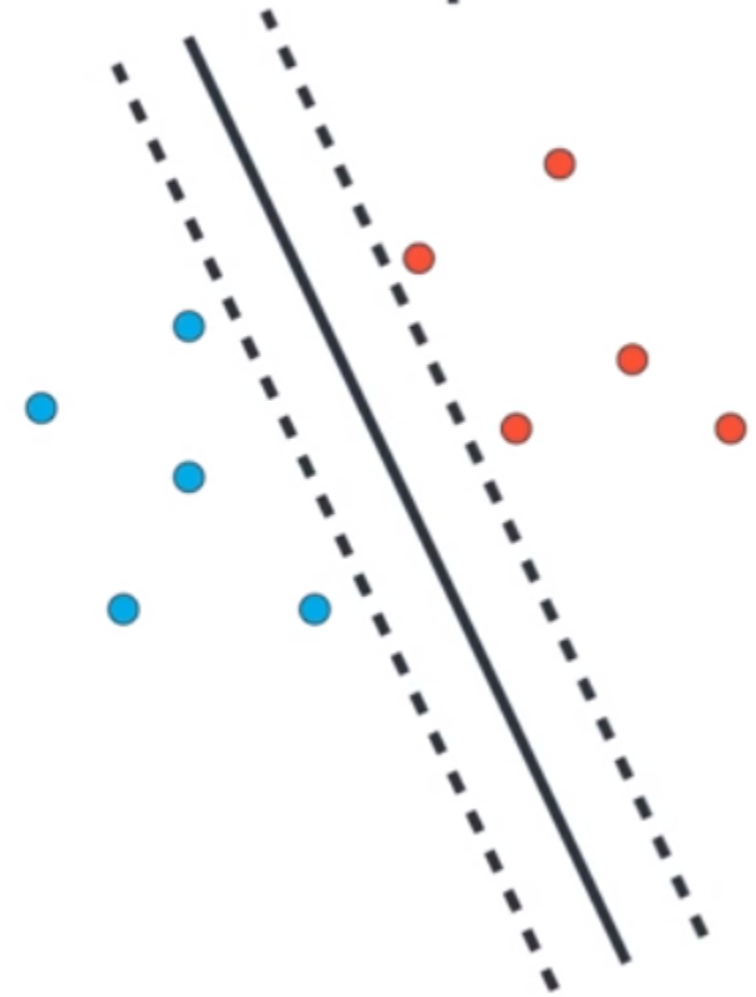
Split data - separate lines



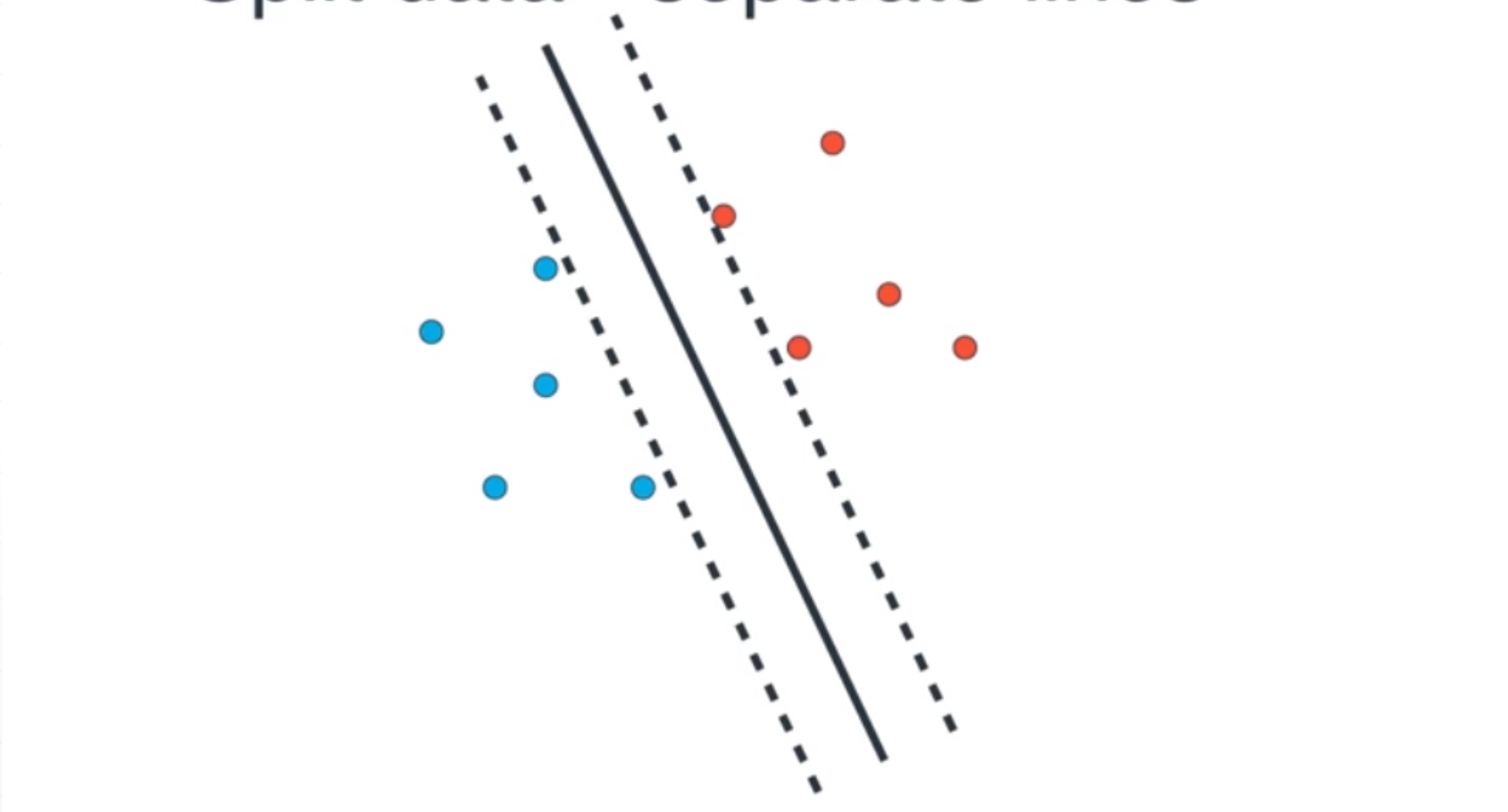
Split data - separate lines



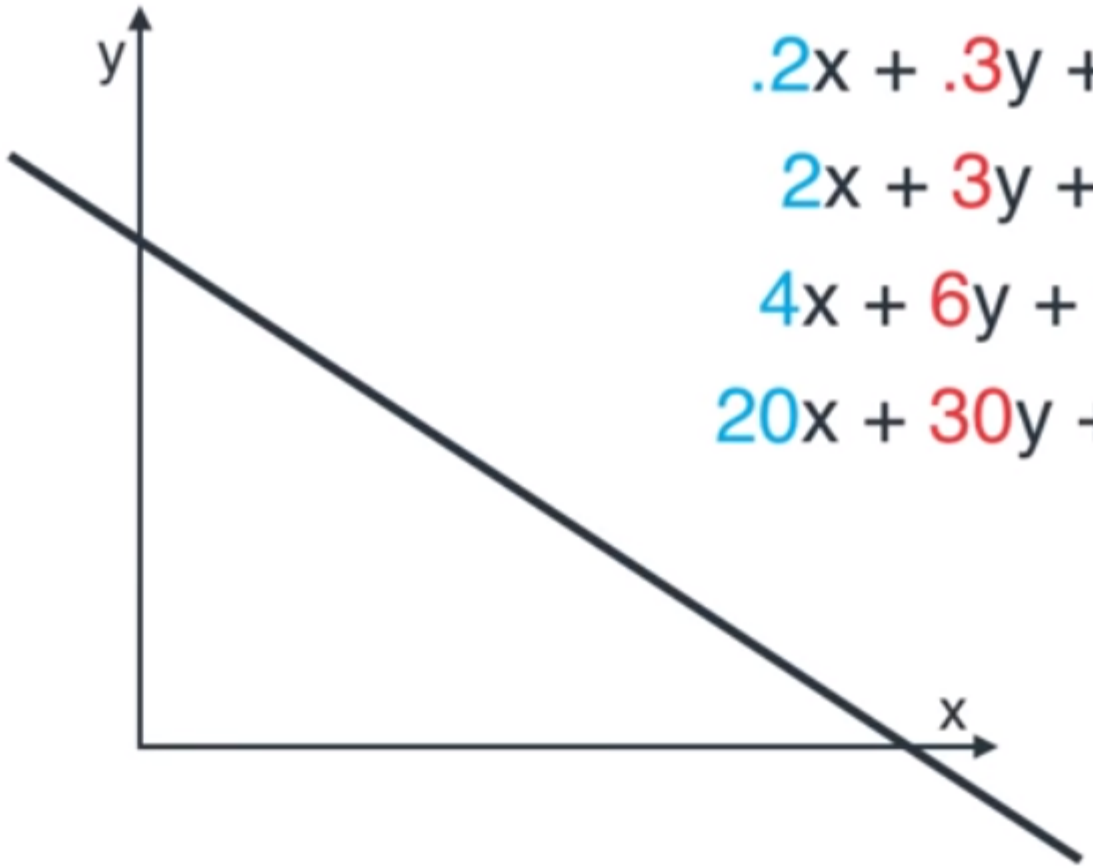
Split data - separate lines



Split data - separate lines



How to separate lines?



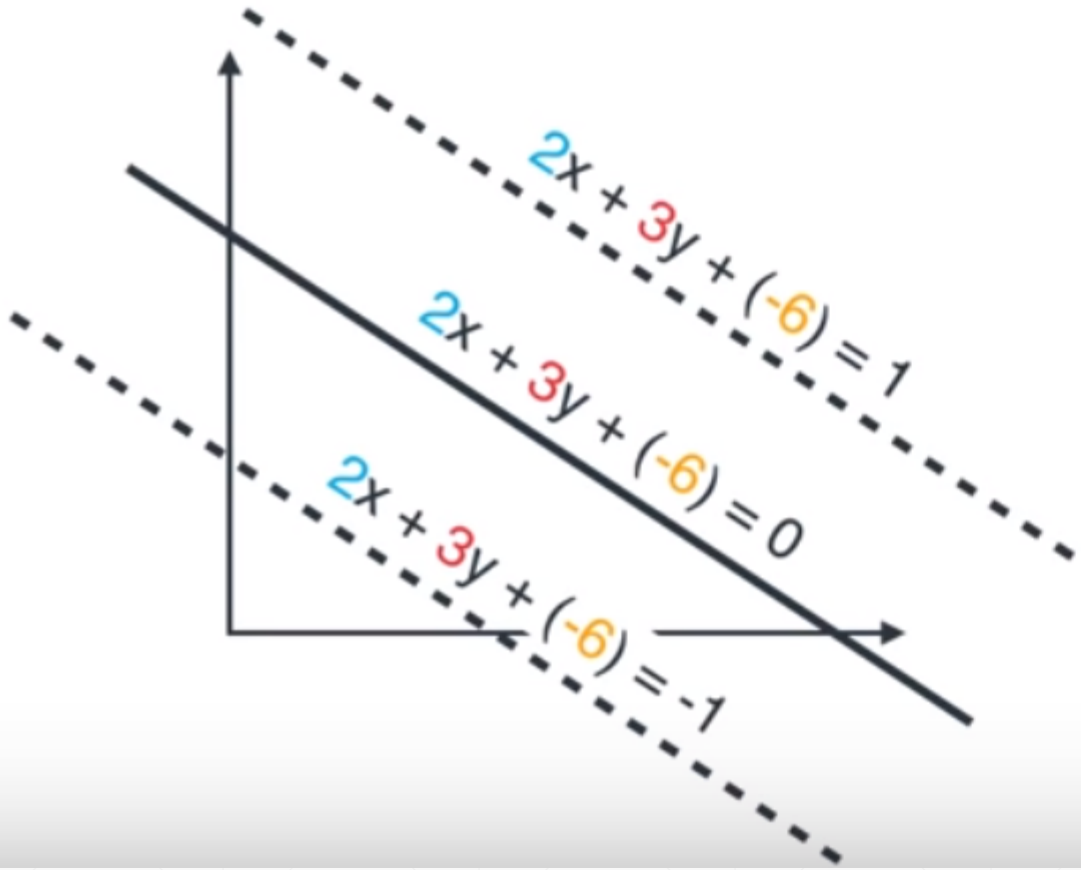
$.2x + .3y + (-.6) = 0$

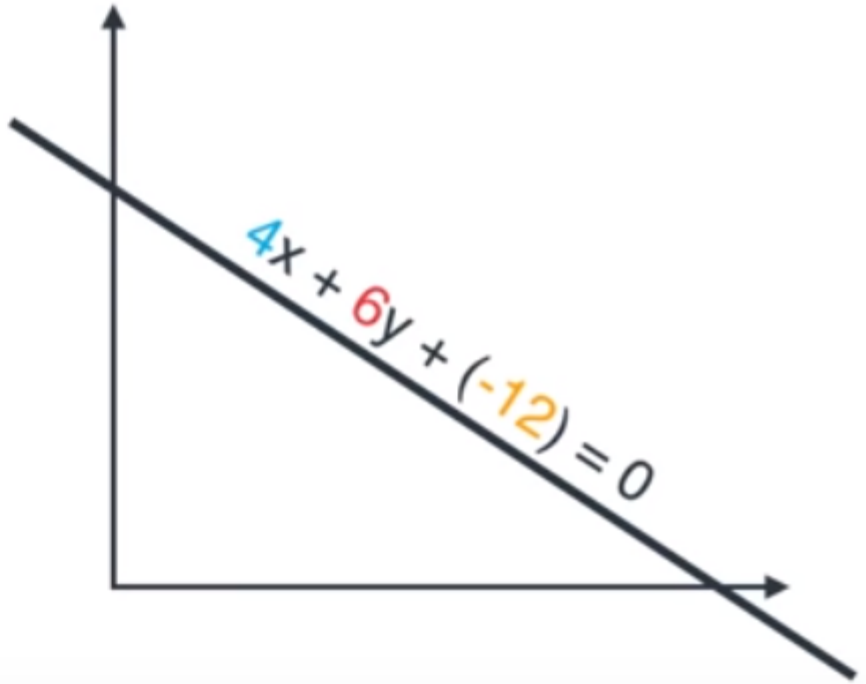
$2x + 3y + (-6) = 0$

$4x + 6y + (-12) = 0$

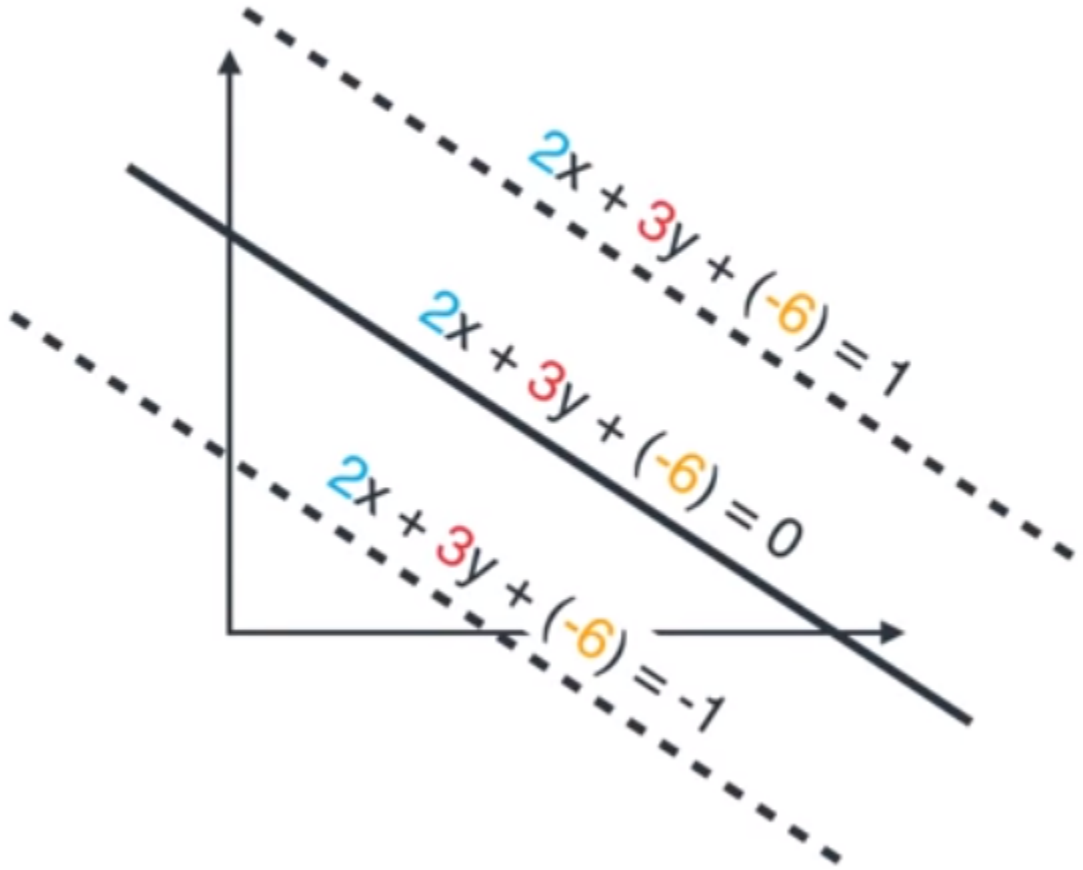
$20x + 30y + (-60) = 0$

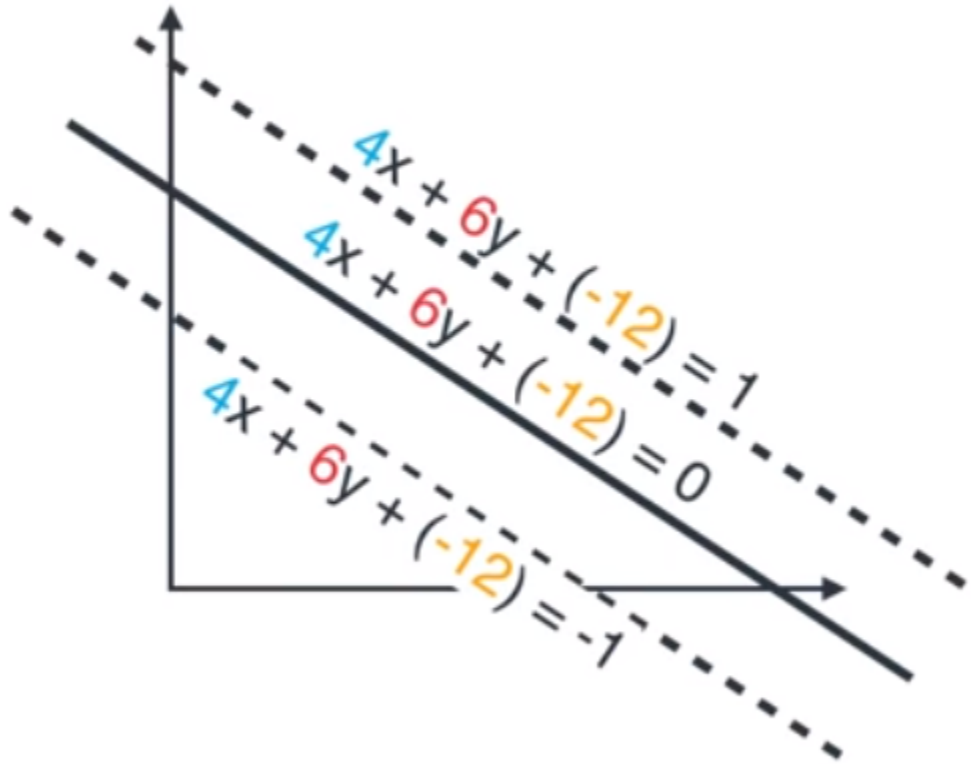
How to separate lines?



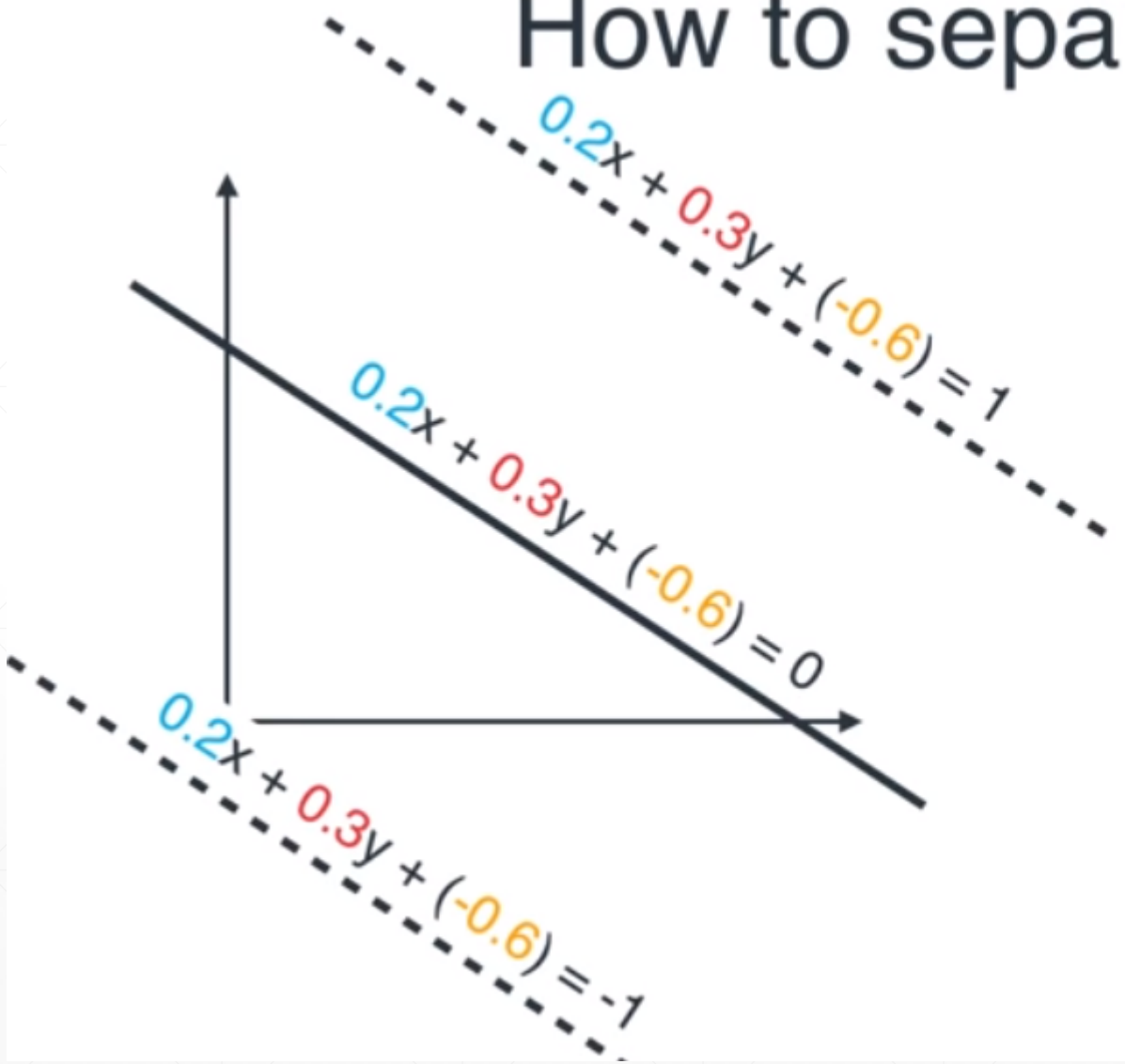


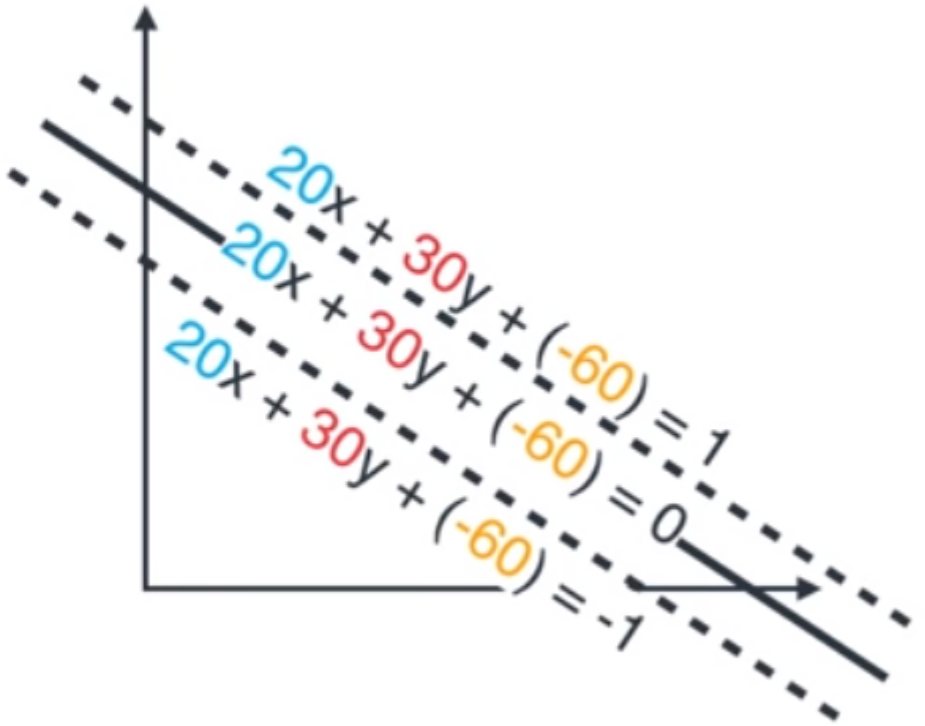
How to separate lines?





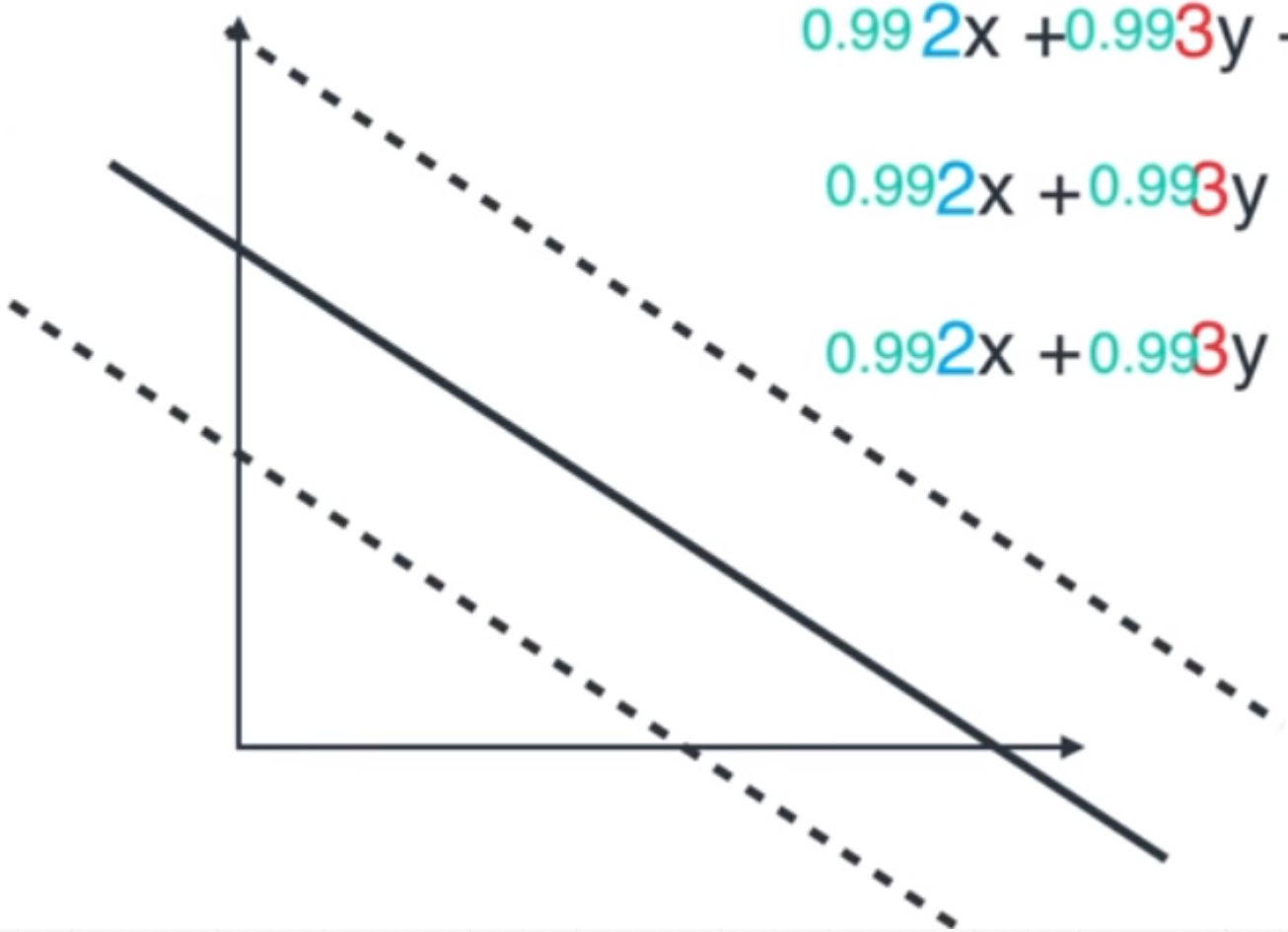
How to separate lines?





Expanding rate

Expanding rate



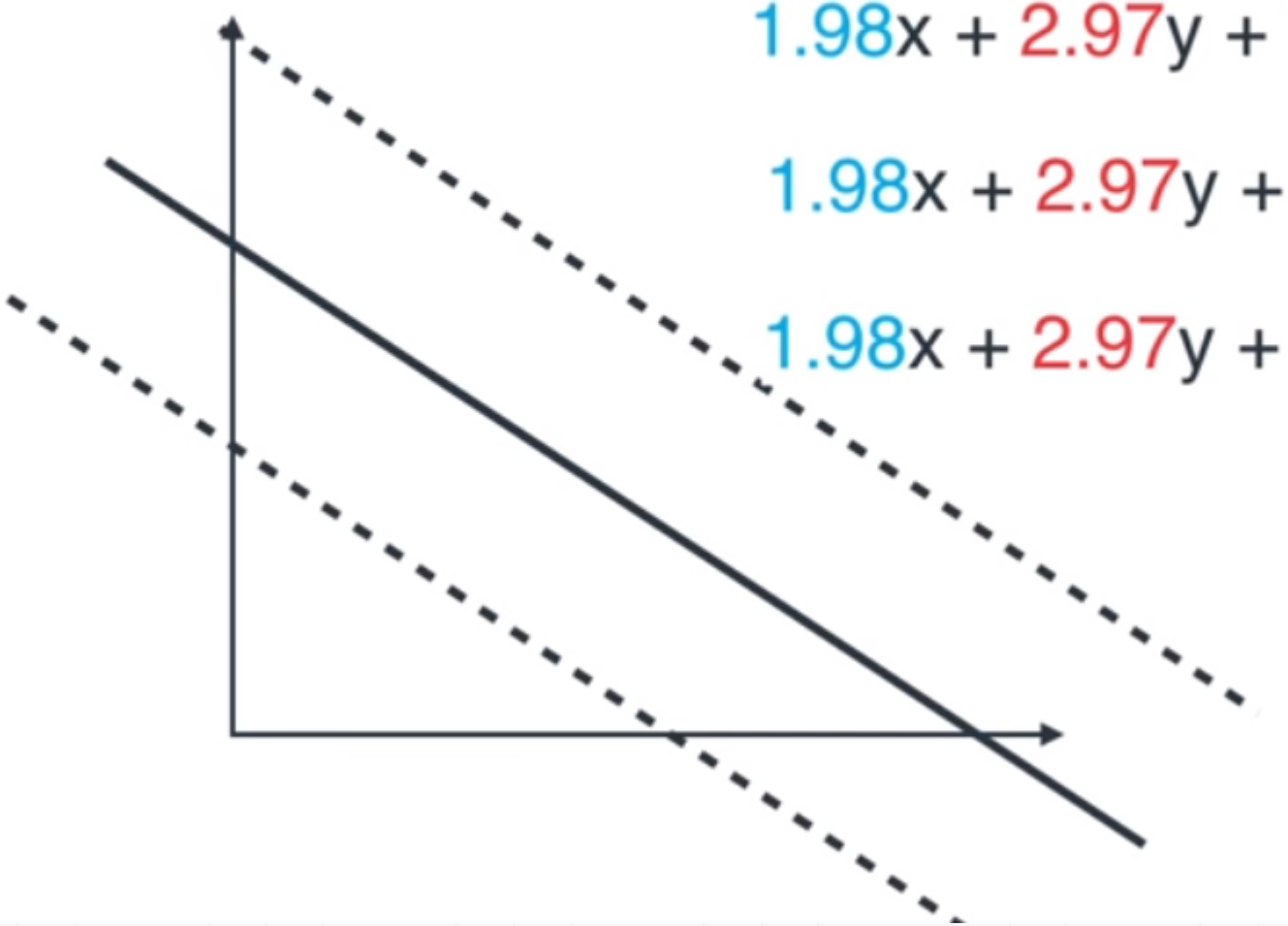
$$0.992x + 0.993y + 0.99(-6) = -1$$

$$0.992x + 0.993y + 0.99(-6) = 0$$

$$0.992x + 0.993y + 0.99(-6) = 1$$

Expanding rate

Expanding rate



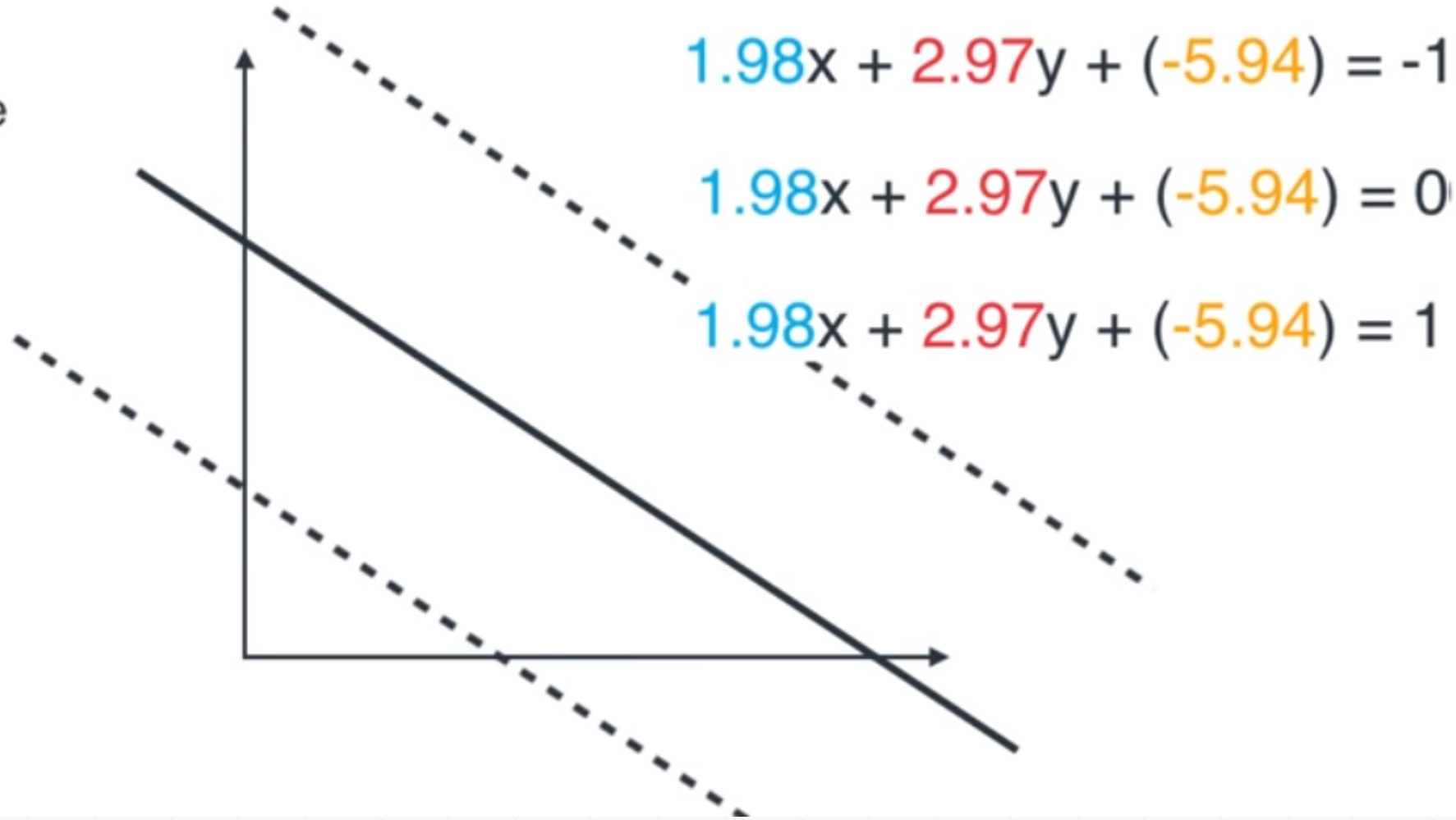
$1.98x + 2.97y + (-5.94) = -1$

$1.98x + 2.97y + (-5.94) = 0$

$1.98x + 2.97y + (-5.94) = 1$

Expanding rate

Expanding rate

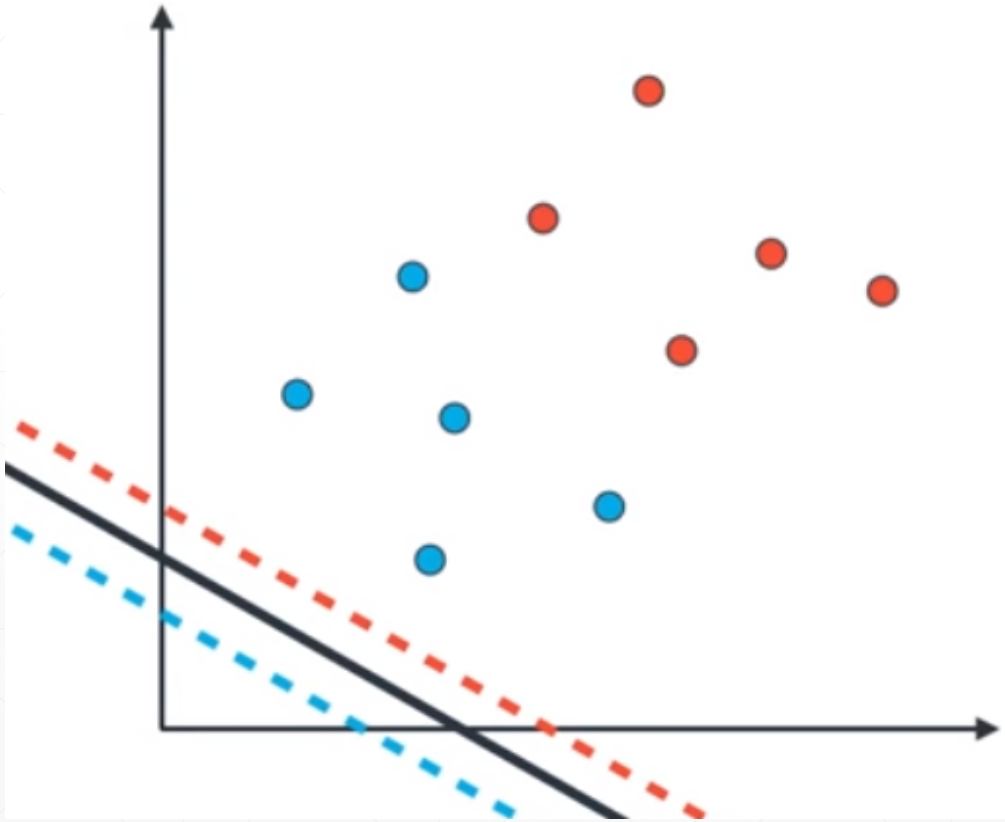


$1.98x + 2.97y + (-5.94) = -1$

$1.98x + 2.97y + (-5.94) = 0$

$1.98x + 2.97y + (-5.94) = 1$

SVM algorithm

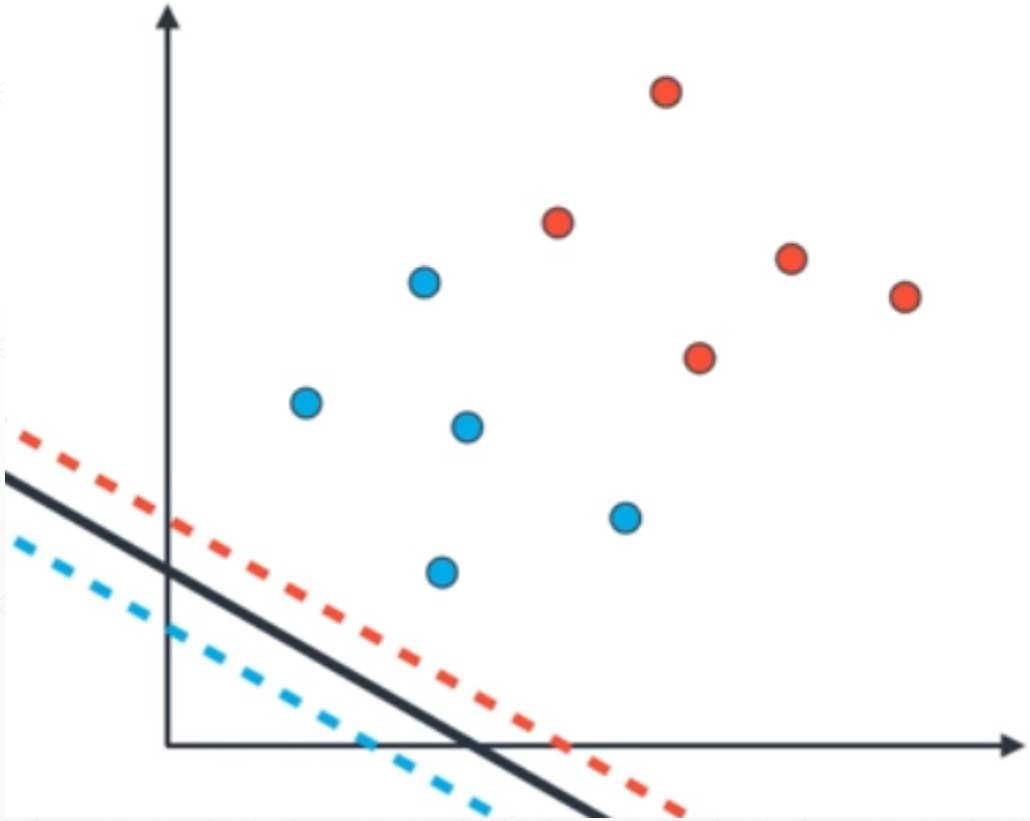


Step 1: Start with a line, and two equidistant parallel lines to it.

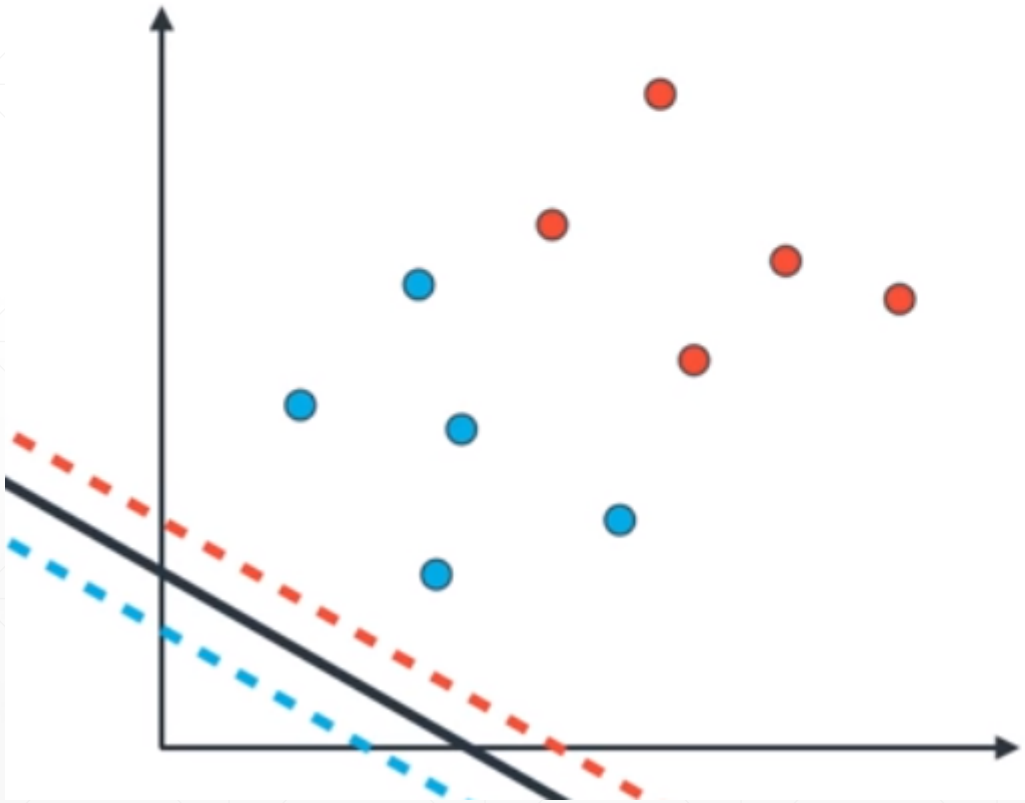
SVM algorithm

Step 1: Start with a line, and two equidistant parallel lines to it.

Step 2: Pick a large number. 1000 (number of repetitions, or epochs)



SVM algorithm

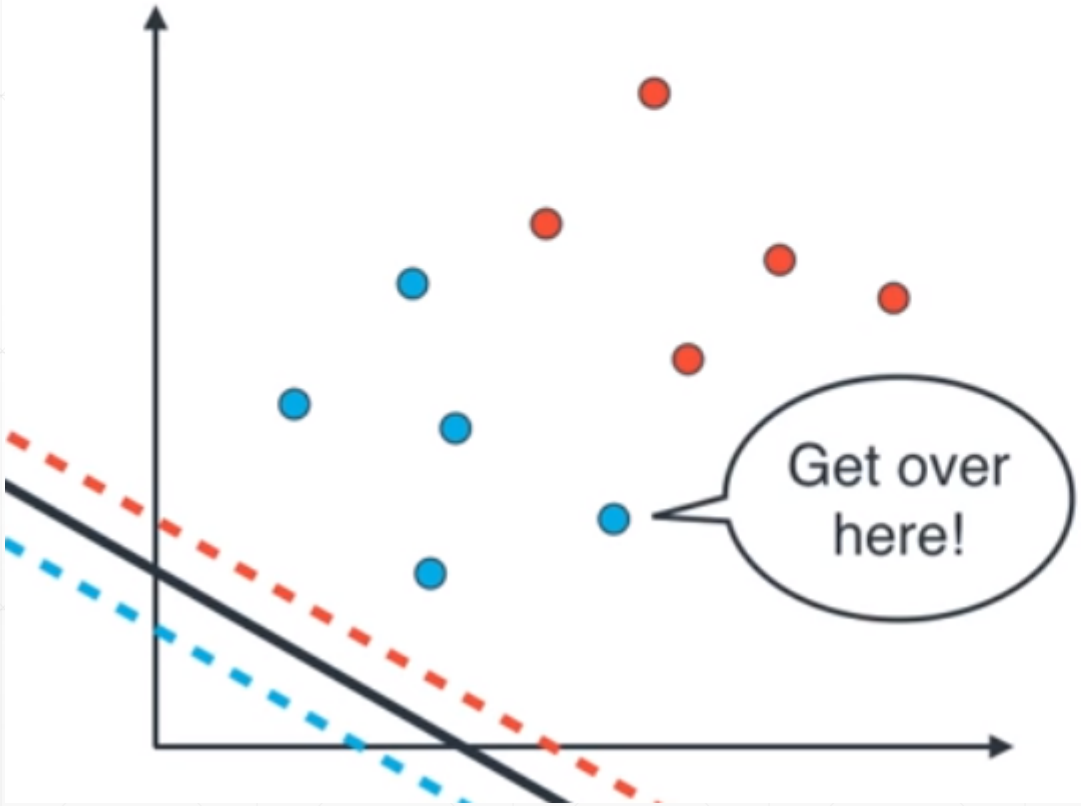


Step 1: Start with a line, and two equidistant parallel lines to it.

Step 2: Pick a large number. 1000 (number of repetitions, or epochs)

Step 3: Pick a number close to 1. (the expanding factor) 0.99

SVM algorithm



Step 1: Start with a line, and two equidistant parallel lines to it.

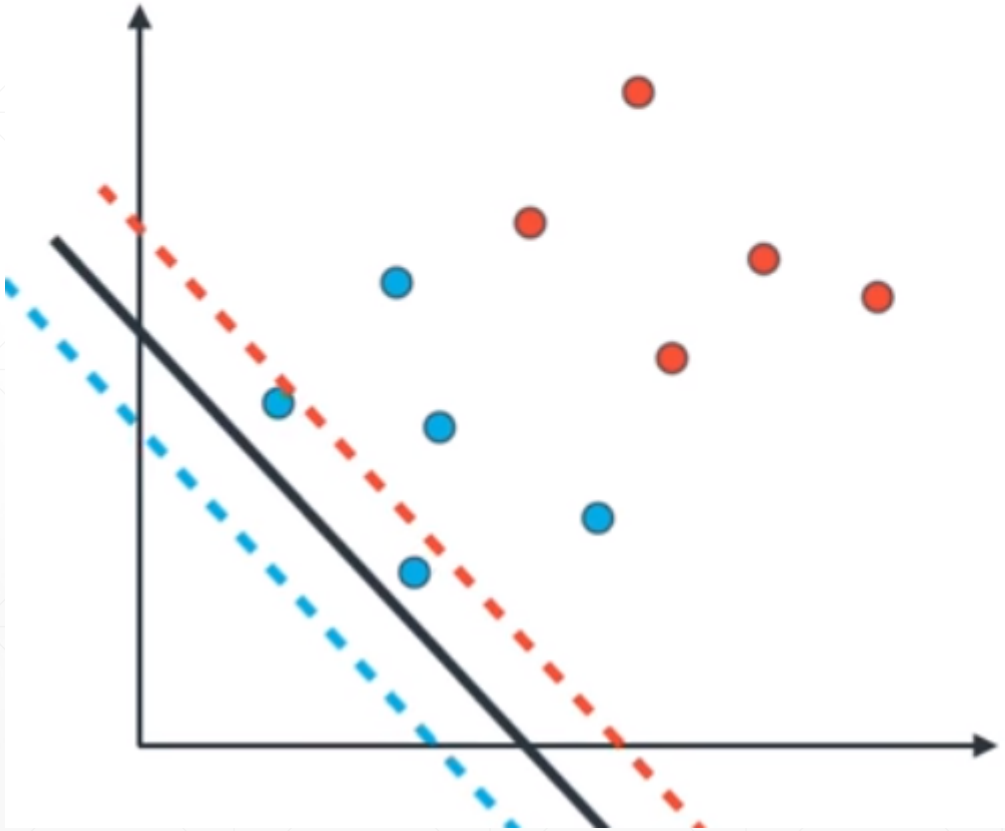
Step 2: Pick a large number. 1000 (number of repetitions, or epochs)

Step 3: Pick a number close to 1. (the expanding factor) 0.99

Step 4: (repeat 1000 times)

- Pick random point
- If point is correctly classified:
  - Do nothing
- If point is incorrectly classified
  - Move line towards point

SVM algorithm



Step 1: Start with a line, and two equidistant parallel lines to it.

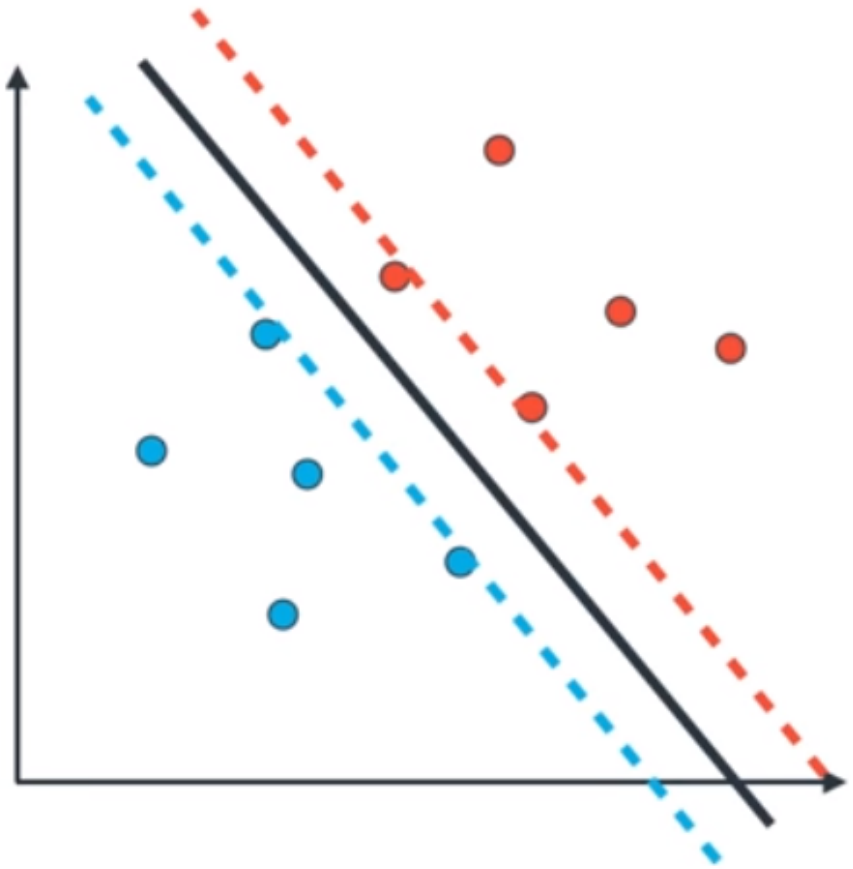
Step 2: Pick a large number. 1000 (number of repetitions, or epochs)

Step 3: Pick a number close to 1. (the expanding factor) 0.99

Step 4: (repeat 1000 times)

- Pick random point
- If point is correctly classified:
  - Do nothing
- If point is incorrectly classified
  - Move line towards point
- Separate the lines using the expanding factor

SVM algorithm



Step 1: Start with a line, and two equidistant parallel lines to it.

Step 2: Pick a large number. 1000 (number of repetitions, or epochs)

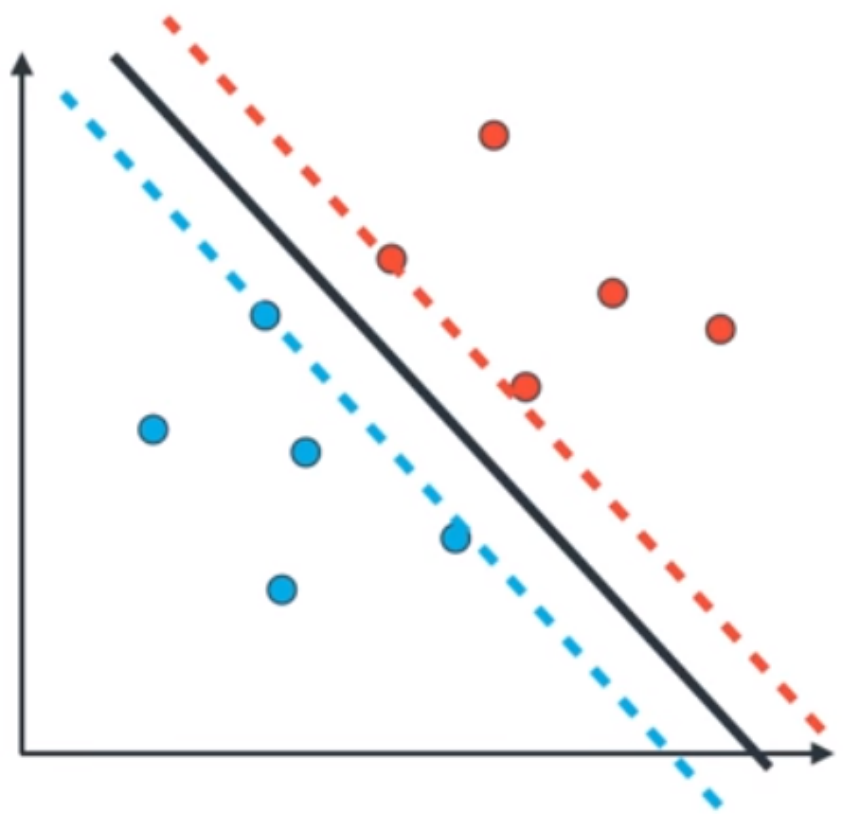
Step 3: Pick a number close to 1. (the expanding factor) 0.99

Step 4: (repeat 1000 times)

- Pick random point
- If point is correctly classified:
  - Do nothing
- If point is incorrectly classified
  - Move line towards point
- Separate the lines using the expanding factor

Step 5: Enjoy your lines that separate the data!

SVM algorithm



Step 1: Start with a random line of equation $ax + by + c = 0$.
Draw parallel lines with equations:
- $ax + by + c = 1$, and
- $ax + by + c = -1$

Step 2: Pick a large number. **1000** (number of repetitions, or epochs)

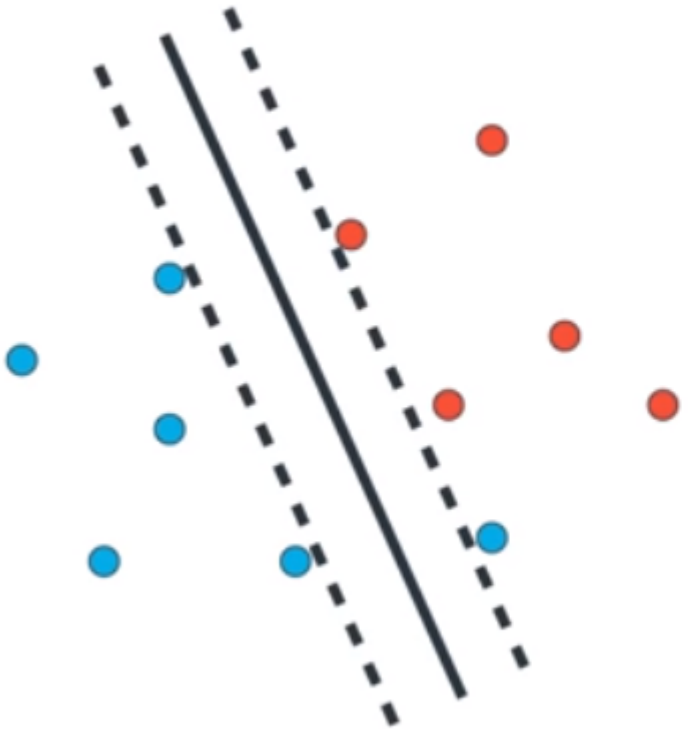
Step 3: Pick a learning rate. **0.01**

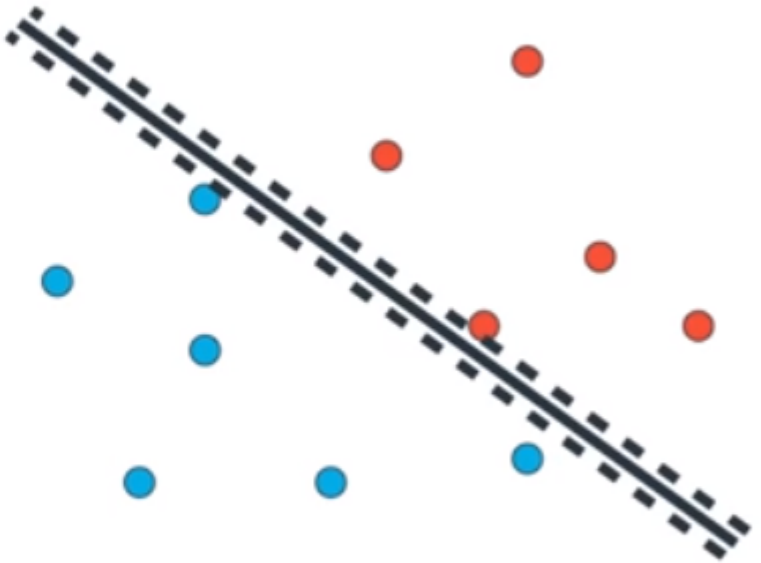
Step 4: Pick an expanding rate. 0.99

Step 5: (repeat **1000** times)
- Pick random point **(p,q)**
- If point is correctly classified
  - Do nothing
- If point is blue, and $ap+bq+c > 0$
  - Subtract 0.01**p** to a
  - Subtract 0.01**q** to b
  - Subtract 0.01 to c
- If point is, red and $ap+bq+c < 0$
  - Add 0.01**p** to a
  - Add 0.01**q** to b
  - Add 0.01 to c
- **Multiply a, b, c, by 0.99**

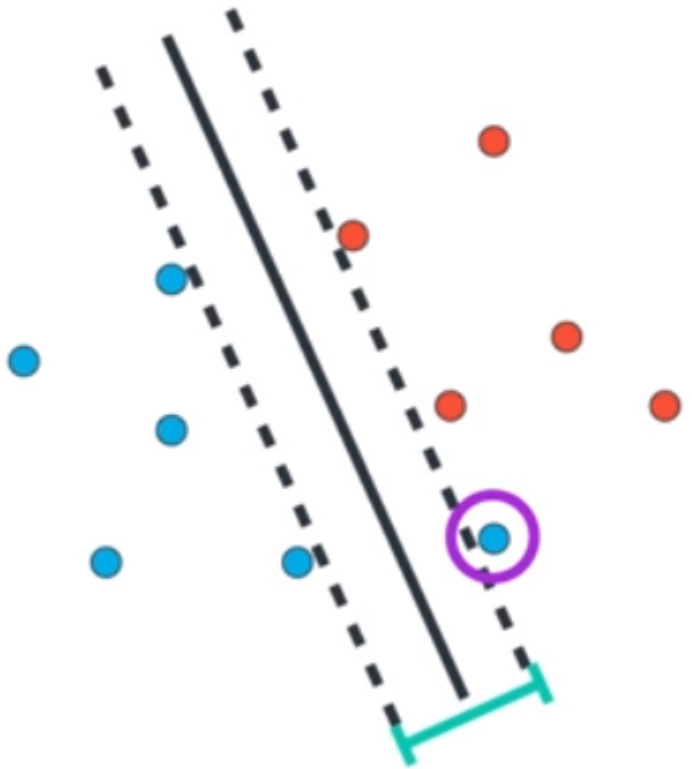
Algo 2:
Support Vector Classifier
(aka soft margin classifier)

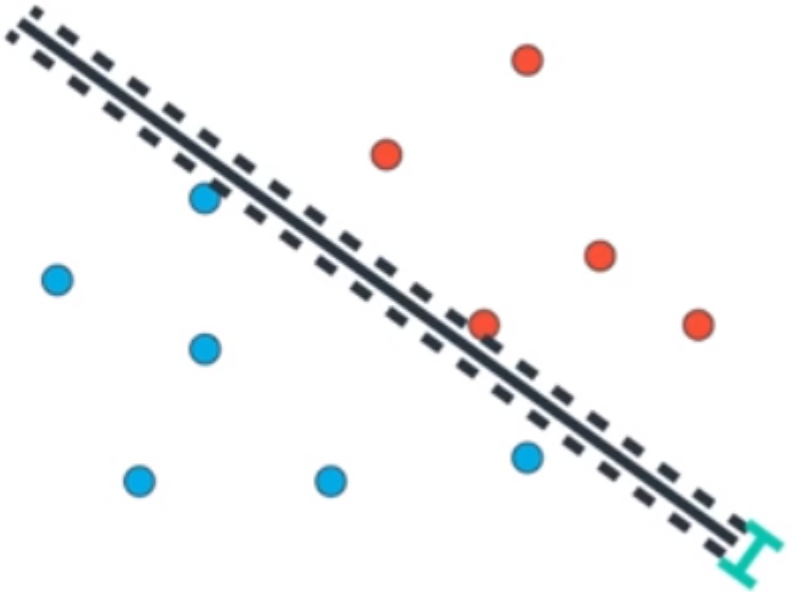
Which line is better?

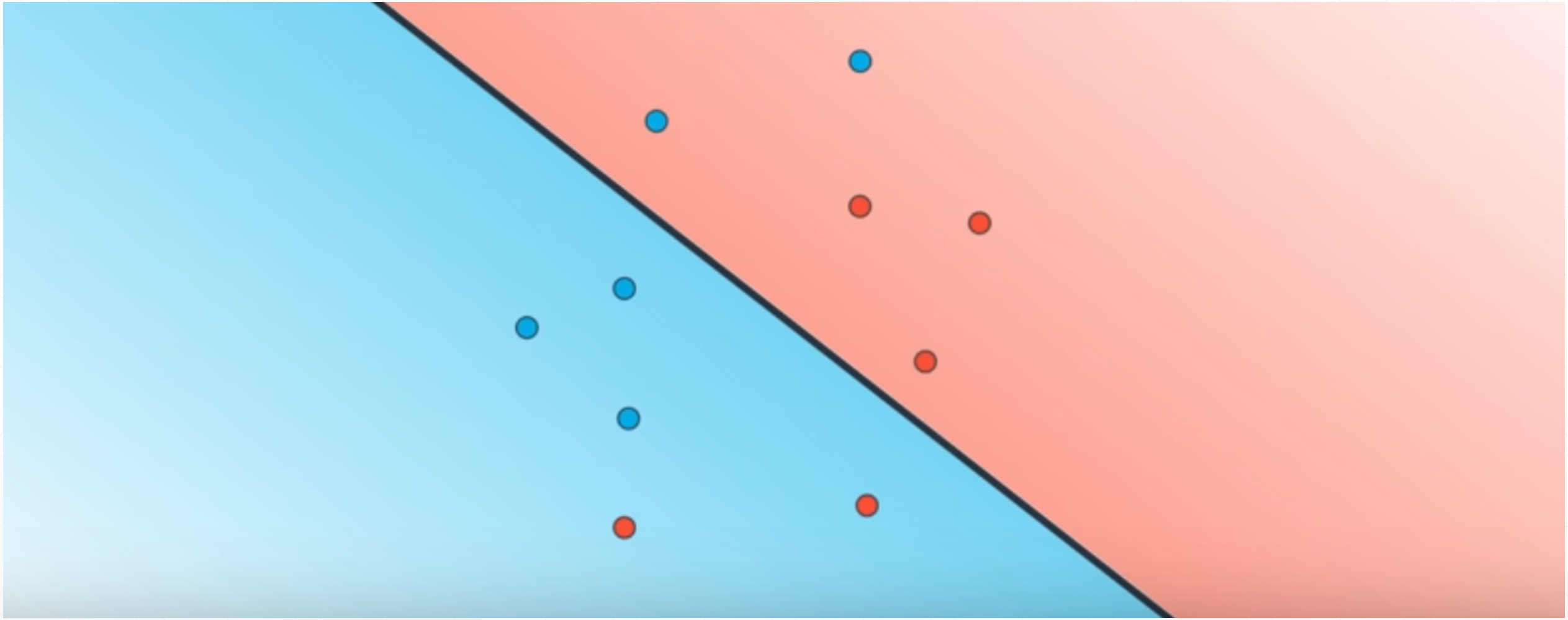


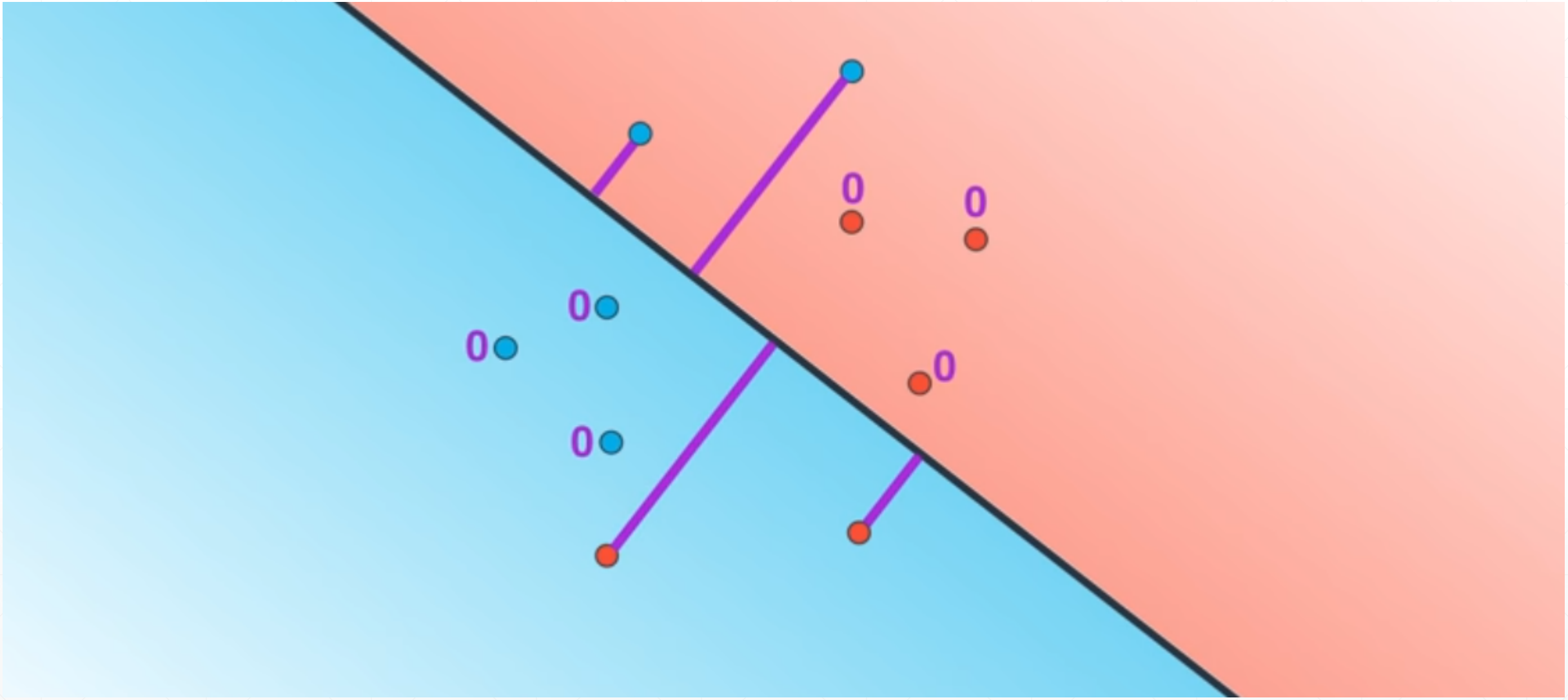


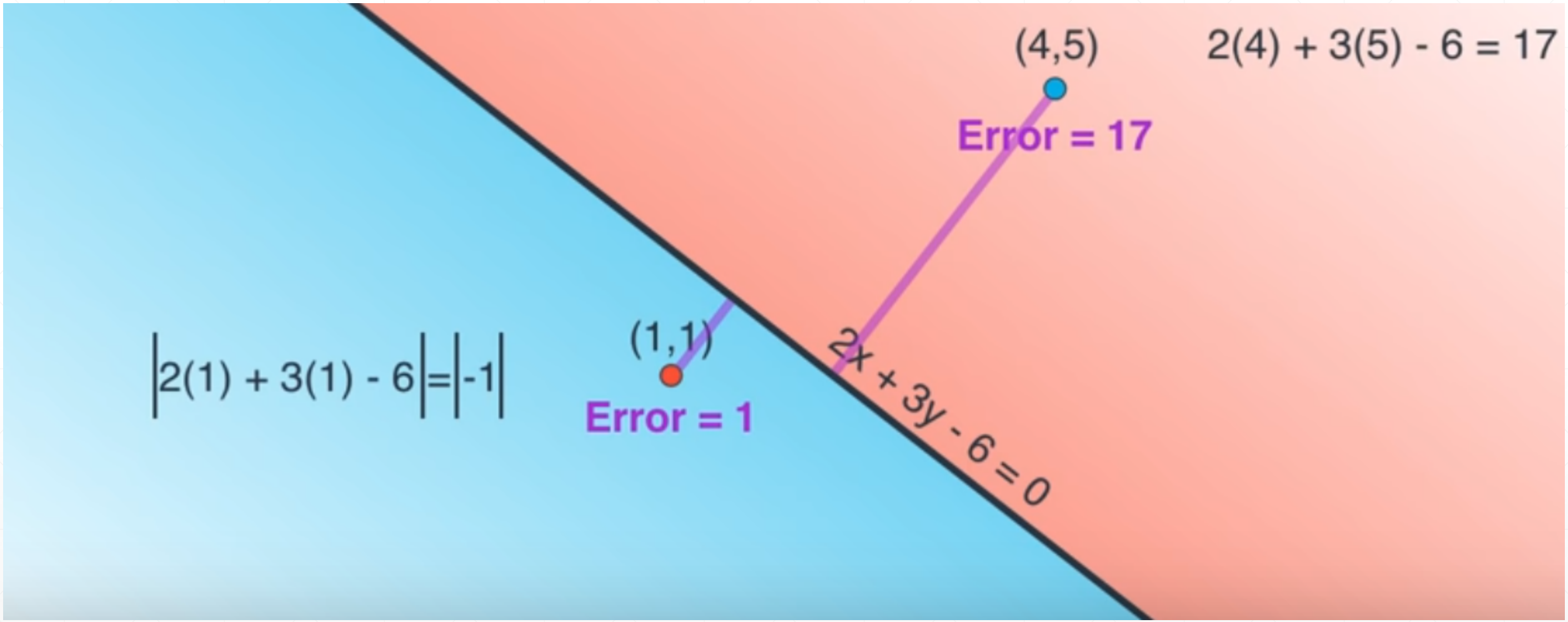
Which line is better?

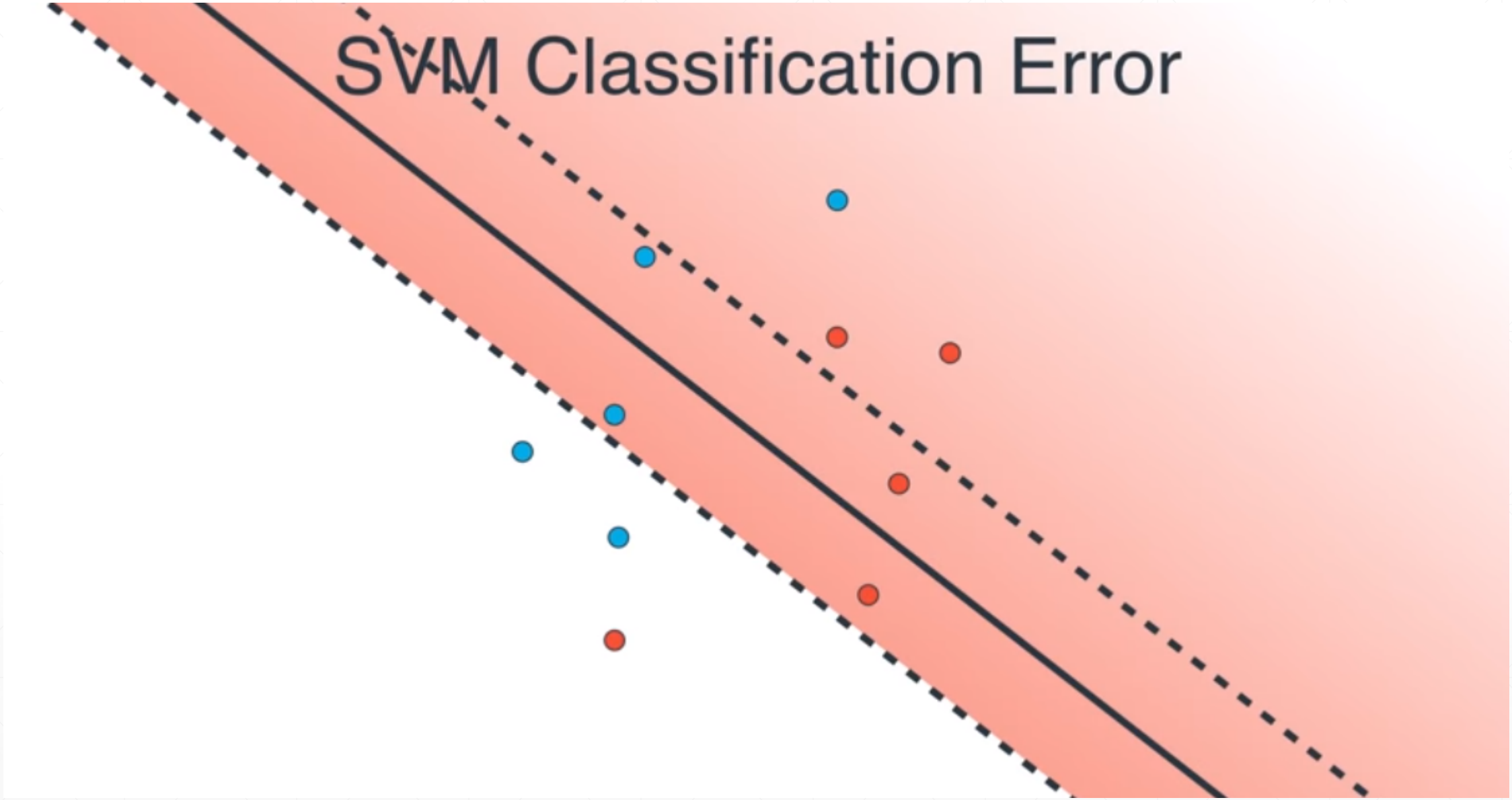


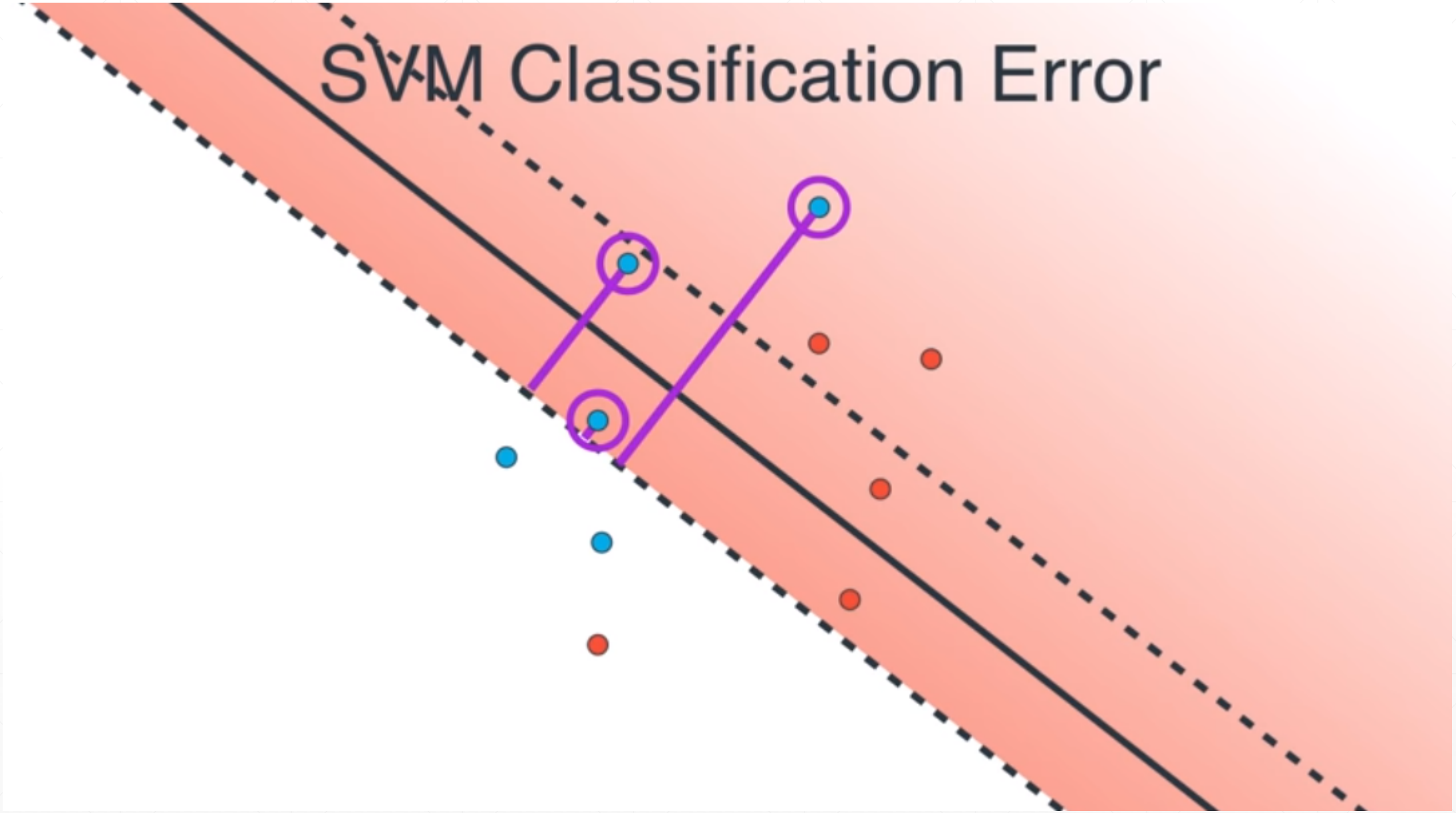


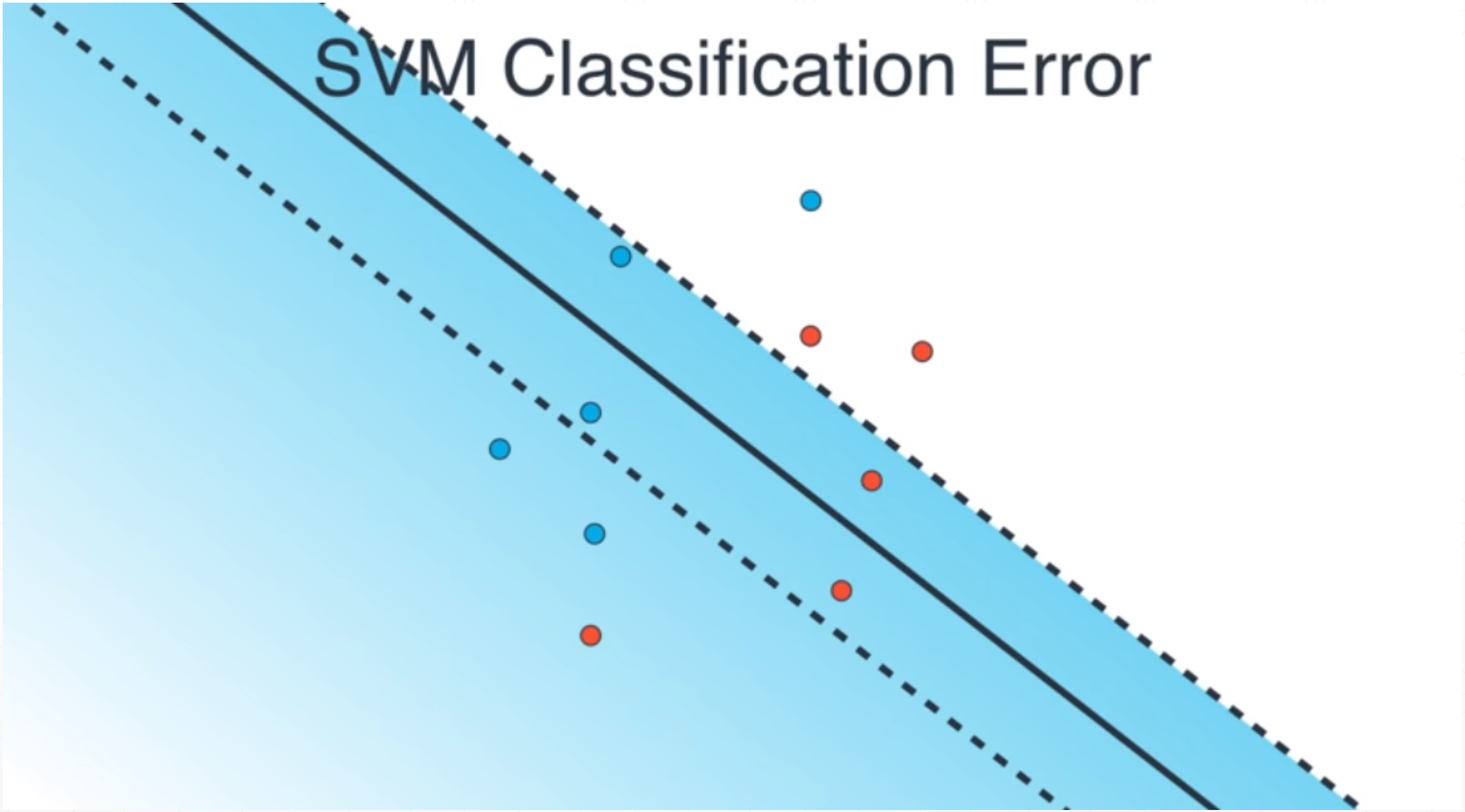


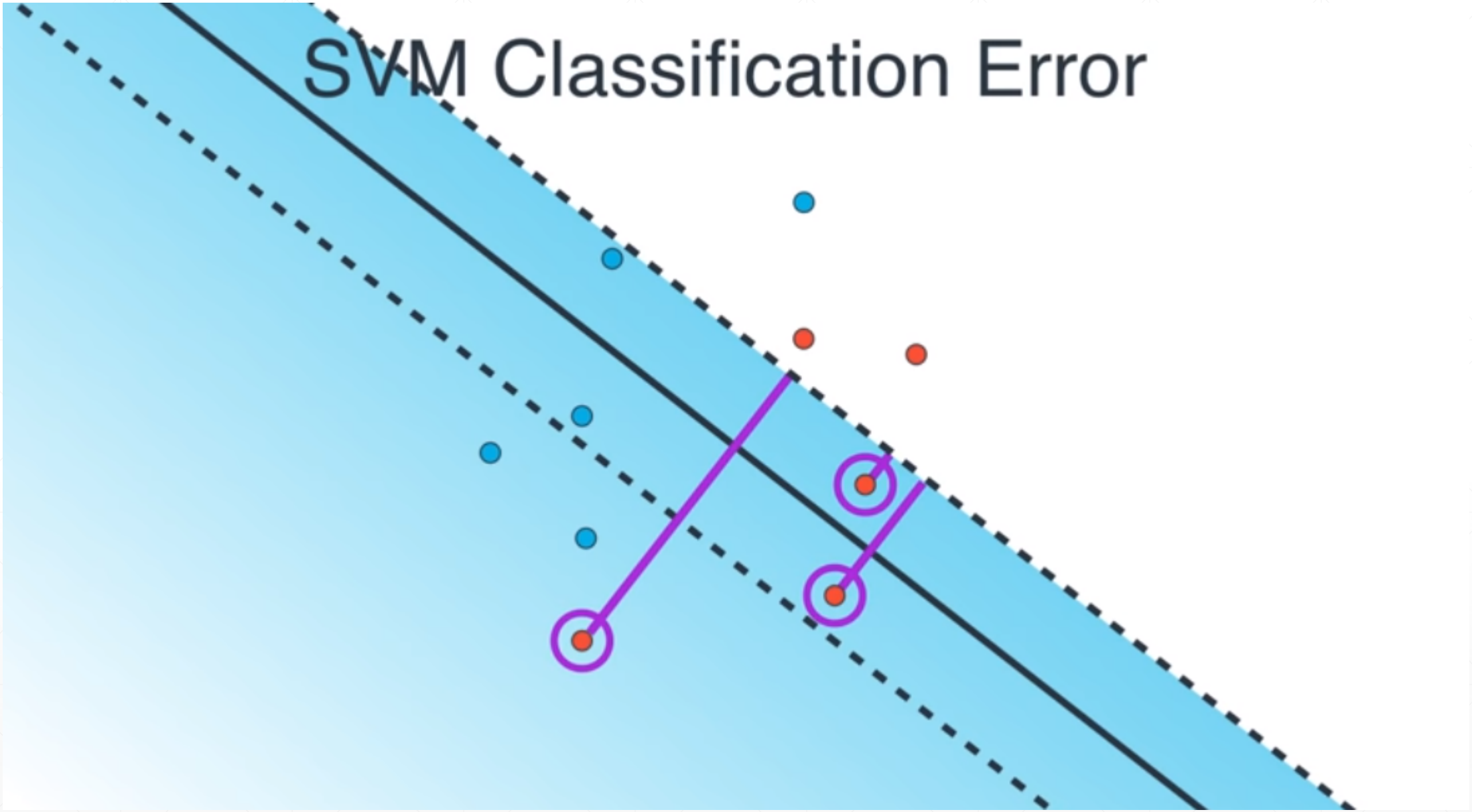

0
0
0
0
0
0

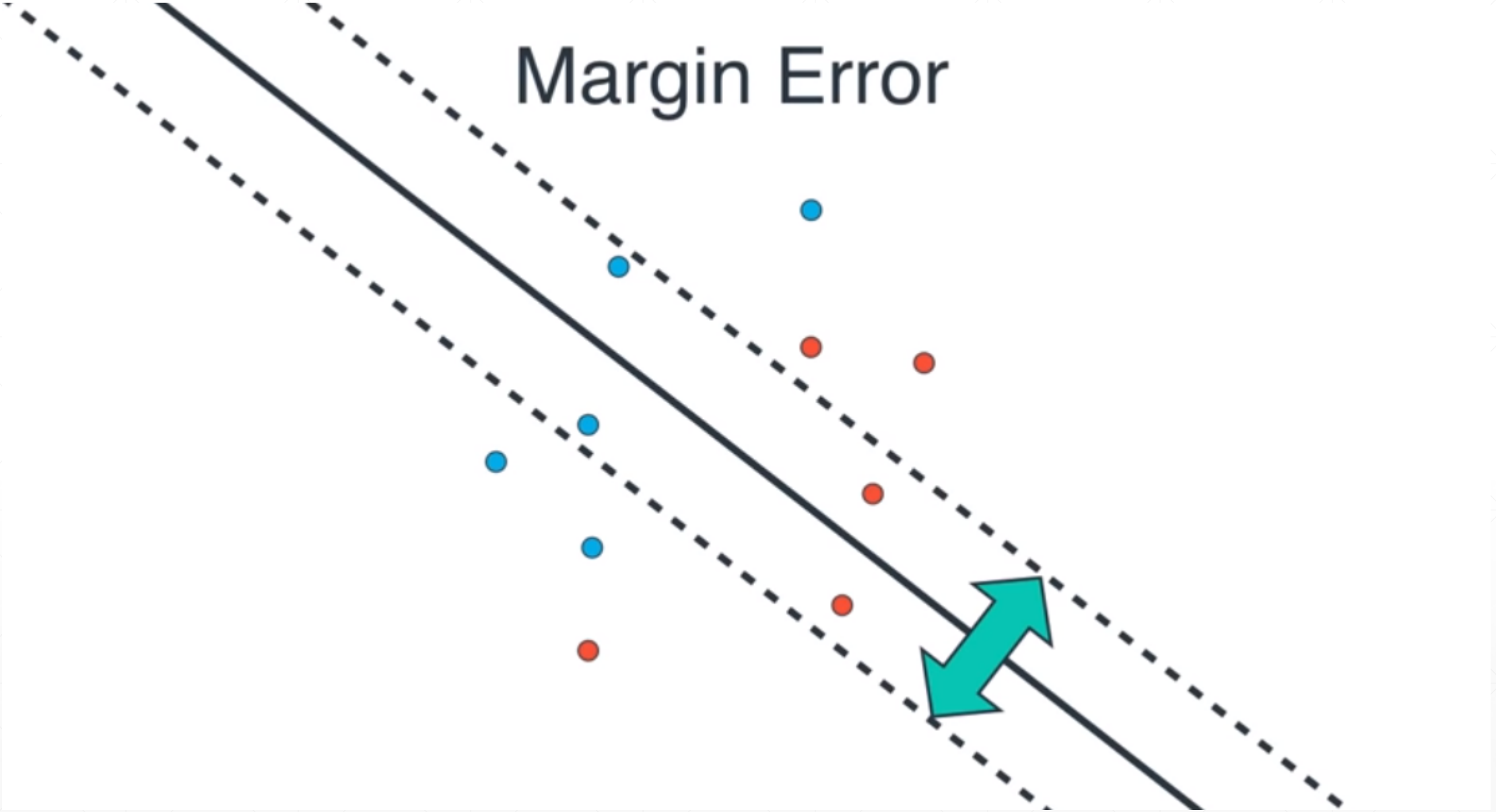

(4,5)
2(4) + 3(5) - 6 = 17
Error = 17
(1,1)
|2(1) + 3(1) - 6| = |-1|
Error = 1
2x + 3y - 6 = 0


SVM Classification Error

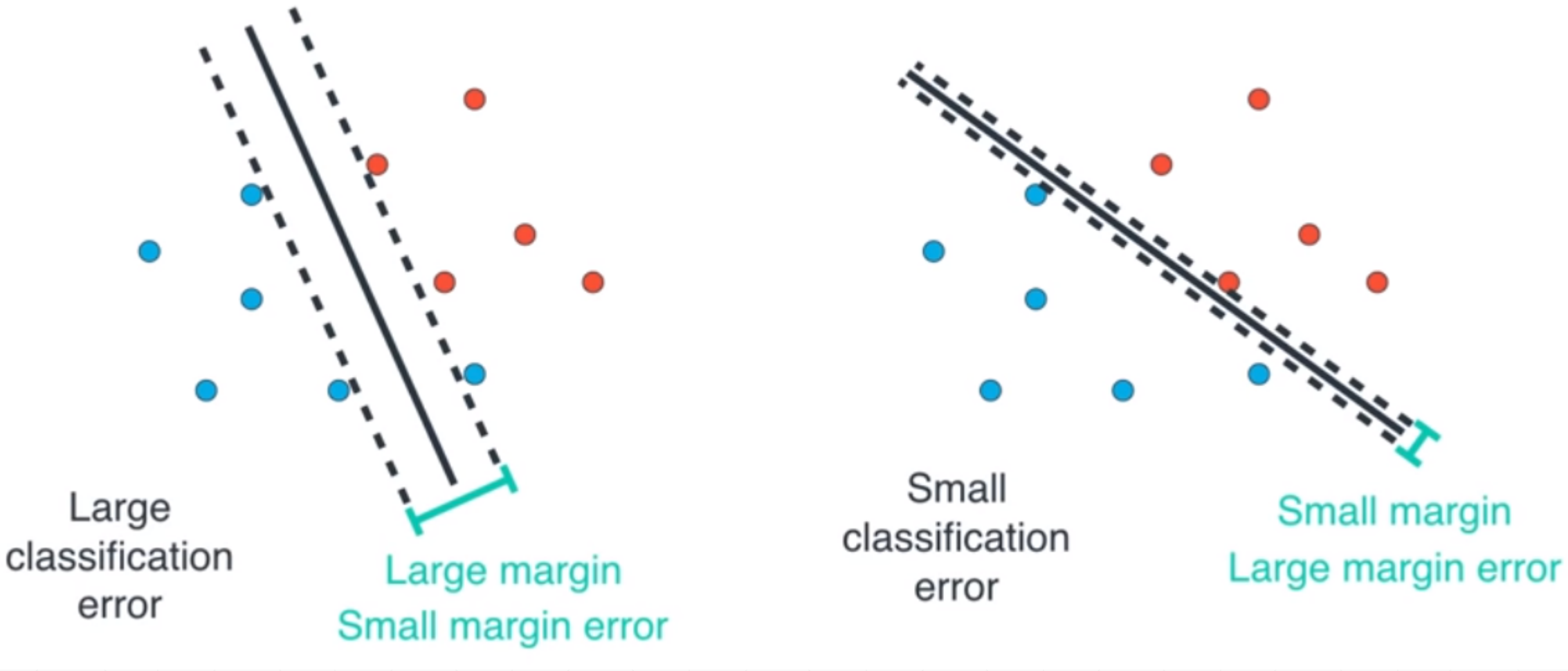

SVM Classification Error


SVM Classification Error

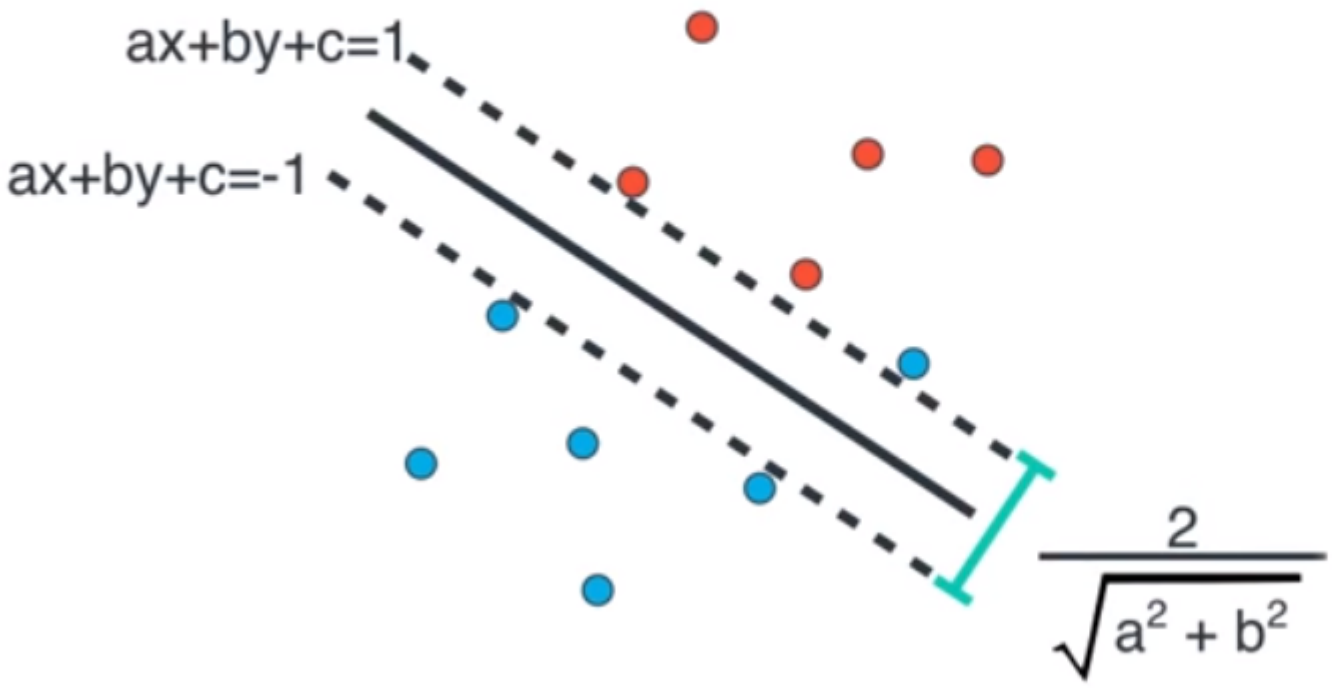

SVM Classification Error


Margin Error

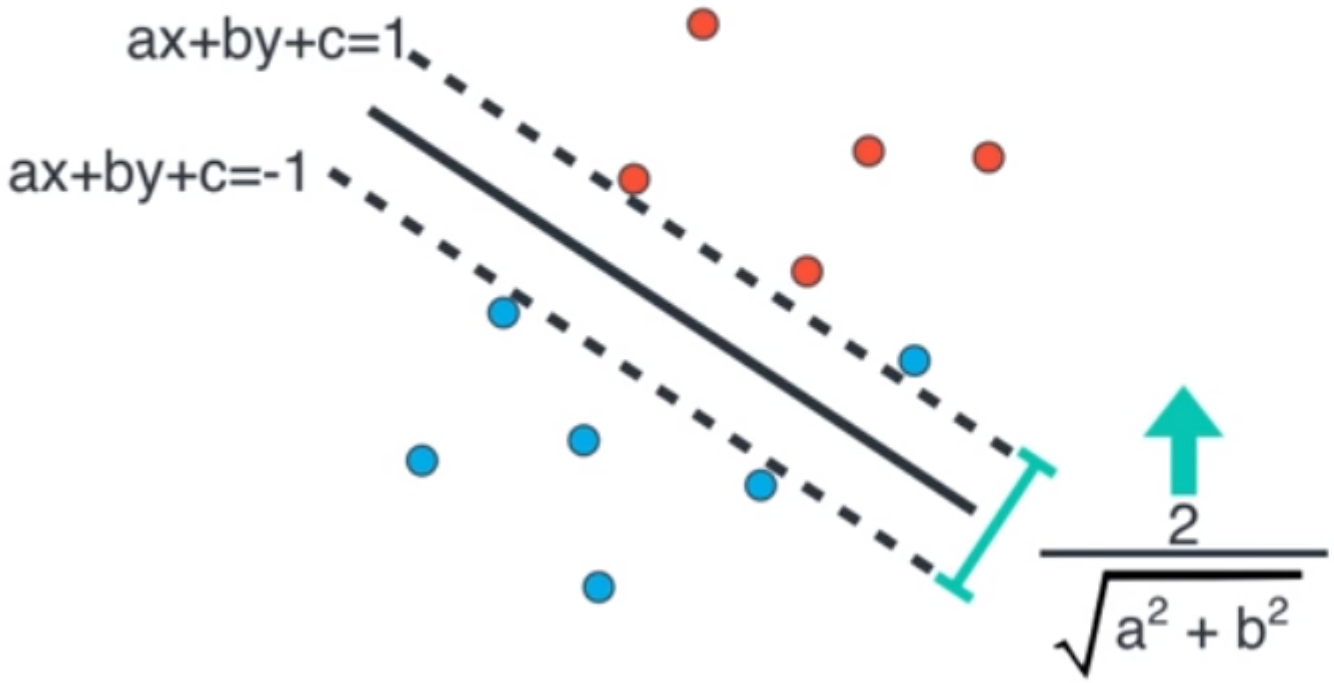
Margin Error



Margin Error



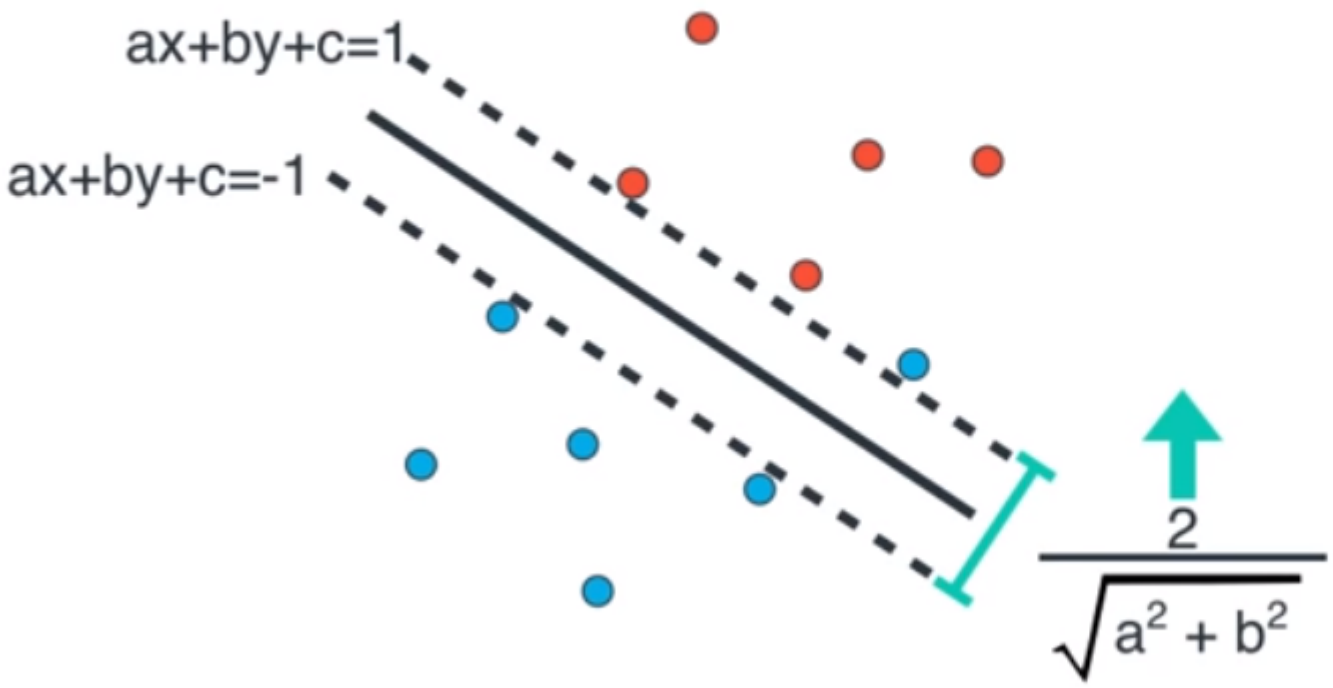
Margin Error



$$\frac{2}{\sqrt{a^2 + b^2}}$$

Margin error = $a^2 + b^2$

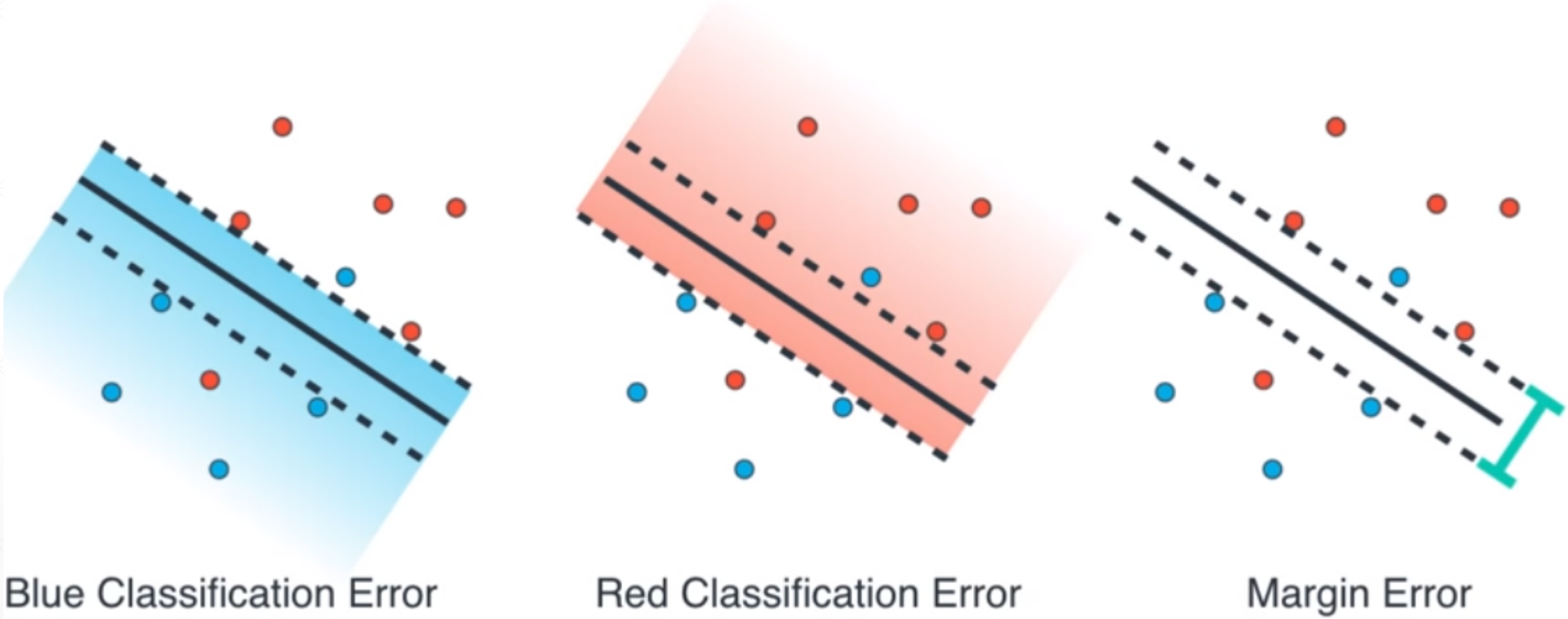
Margin Error



Regularization term!

Margin error = $a^2 + b^2$

SVM Error

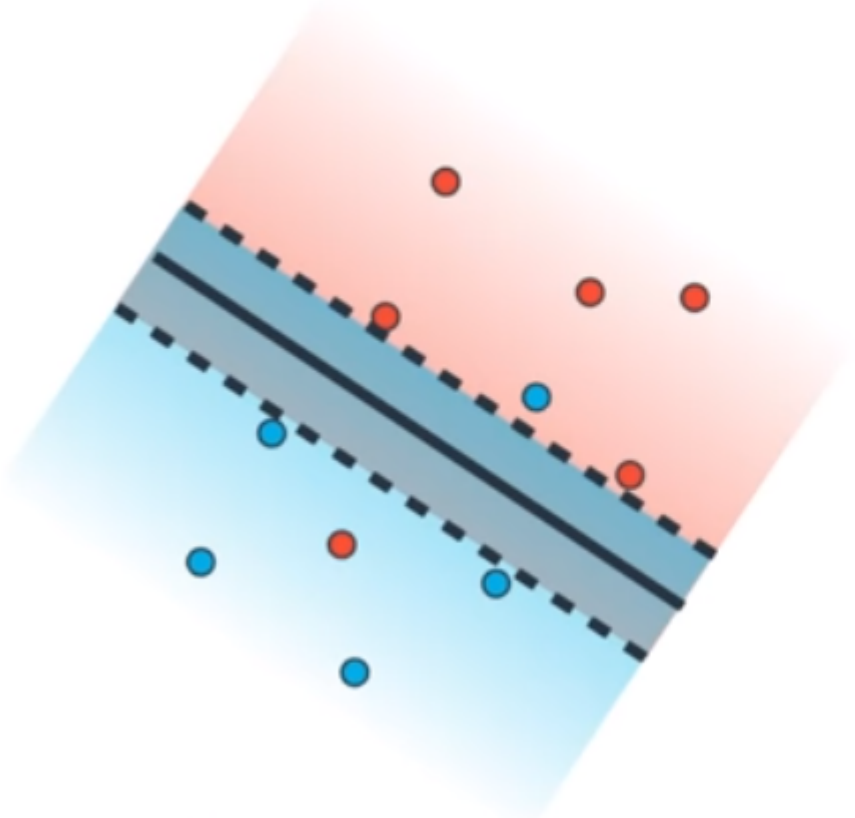


Blue Classification Error

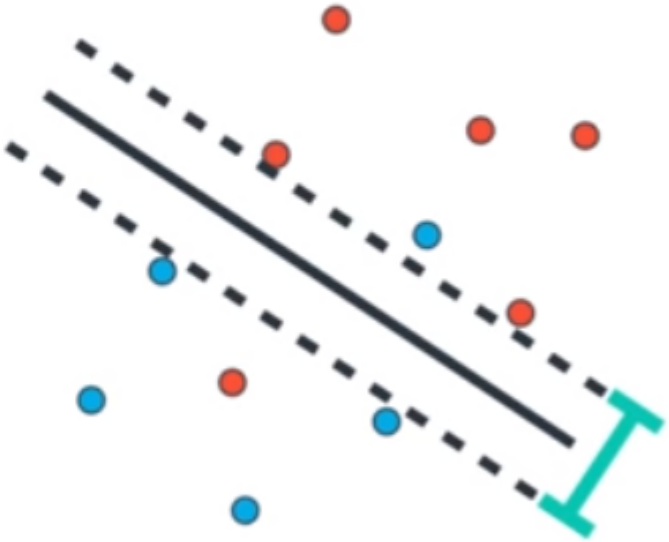
Red Classification Error

Margin Error

SVM Error

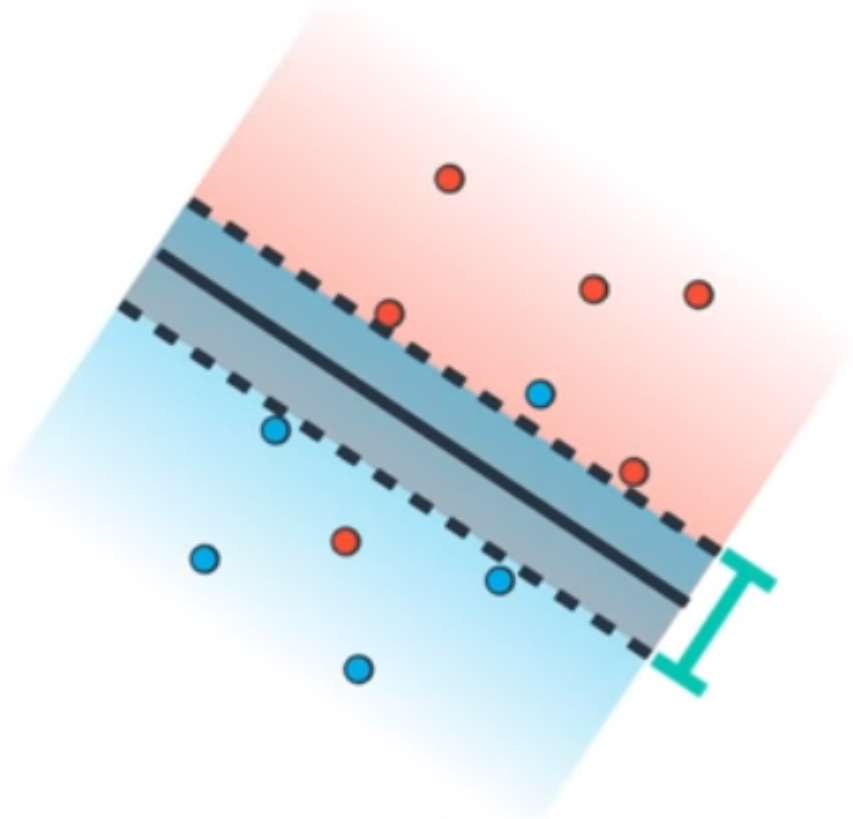


Classification Error



Margin Error

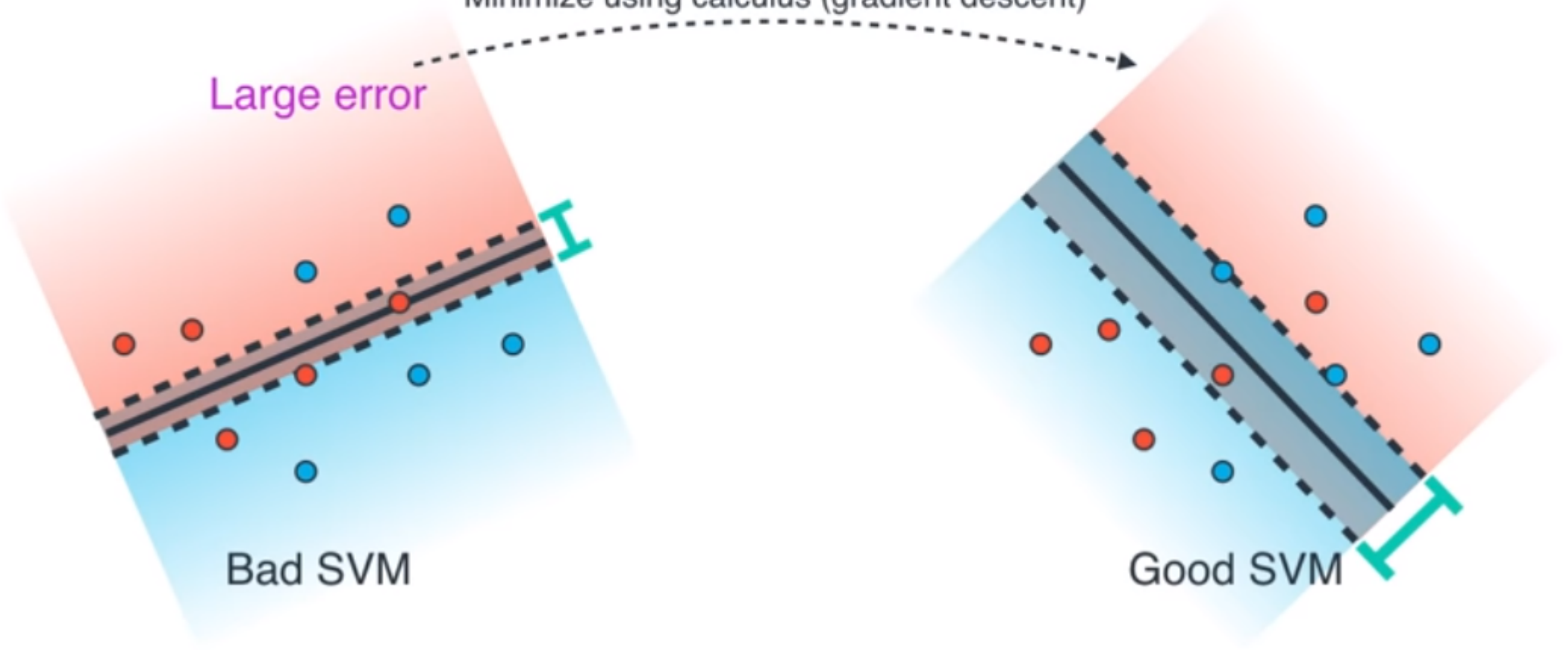
SVM Error



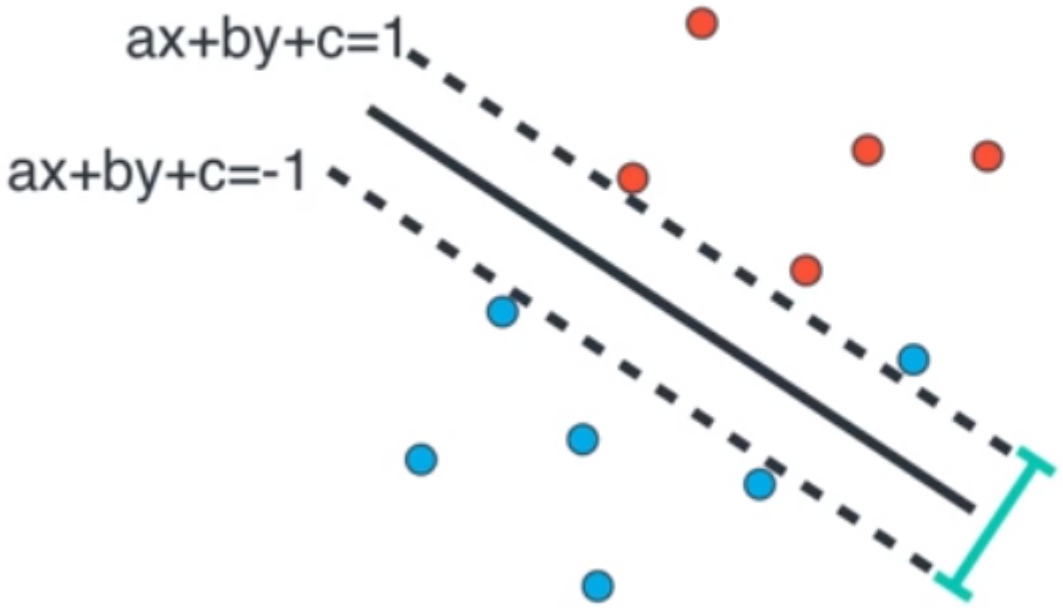
Gradient Descent

Same as the SVM trick!

Minimize using calculus (gradient descent)



Challenge - Gradient Descent



Margin error = $a^2 + b^2$

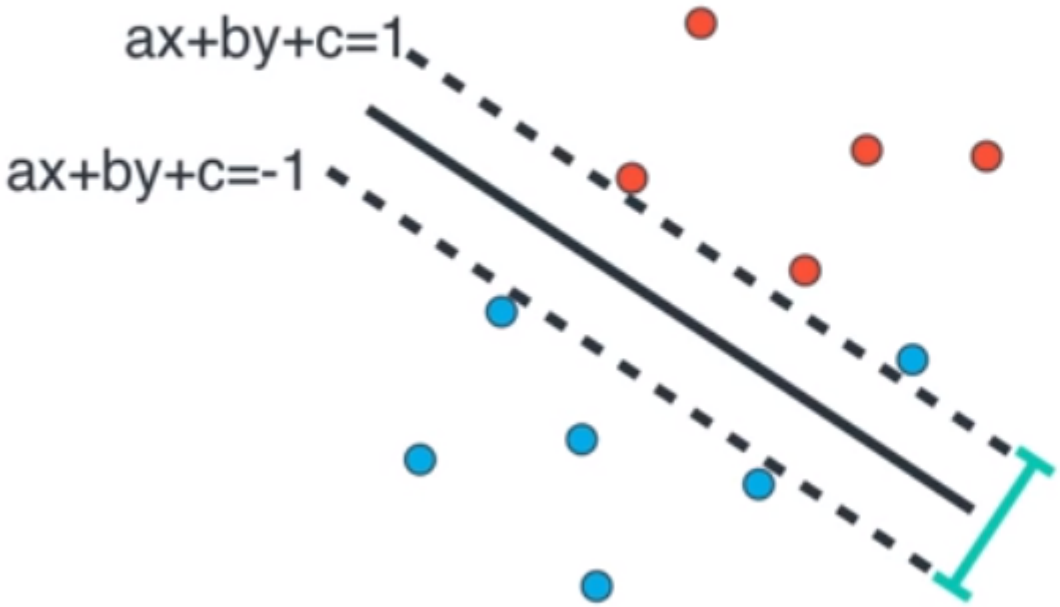
dError/da = 2a

dError/db = 2b

$a \longrightarrow a - \eta\, 2a = a(1 - 2\eta)$

$b \longrightarrow b - \eta\, 2b = b(1 - 2\eta)$

Challenge - Gradient Descent



Margin error = $a^2 + b^2$

expanding factor!

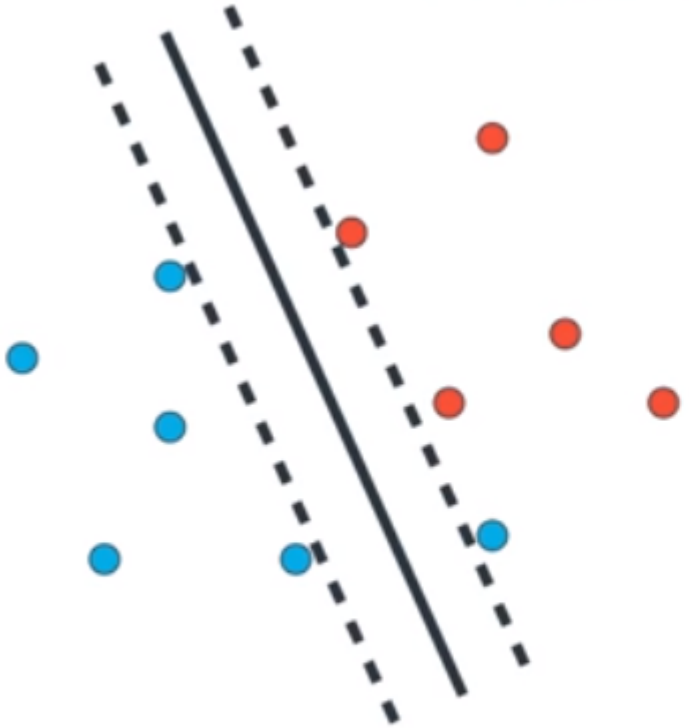
dError/da = 2a

dError/db = 2b

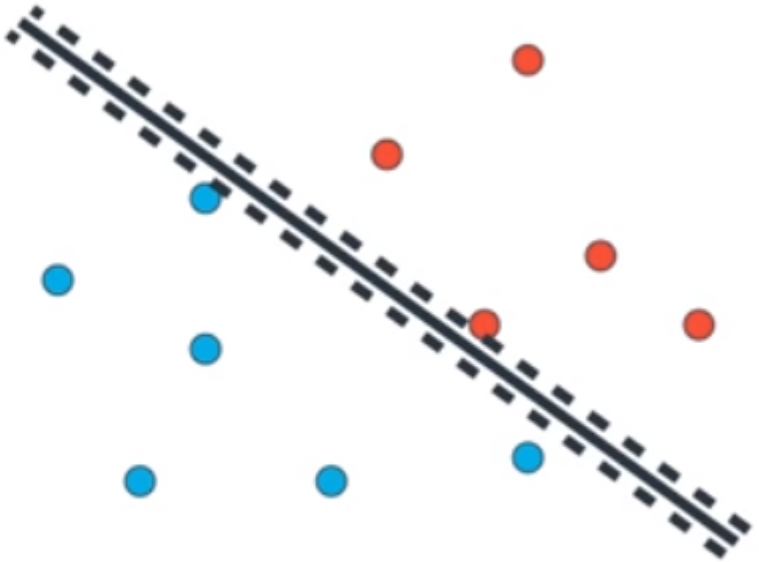
$a \longrightarrow a - \eta\, 2a = a(1 - 2\eta)$

$b \longrightarrow b - \eta\, 2b = b(1 - 2\eta)$

Which line is better?



Classification Error + Margin Error



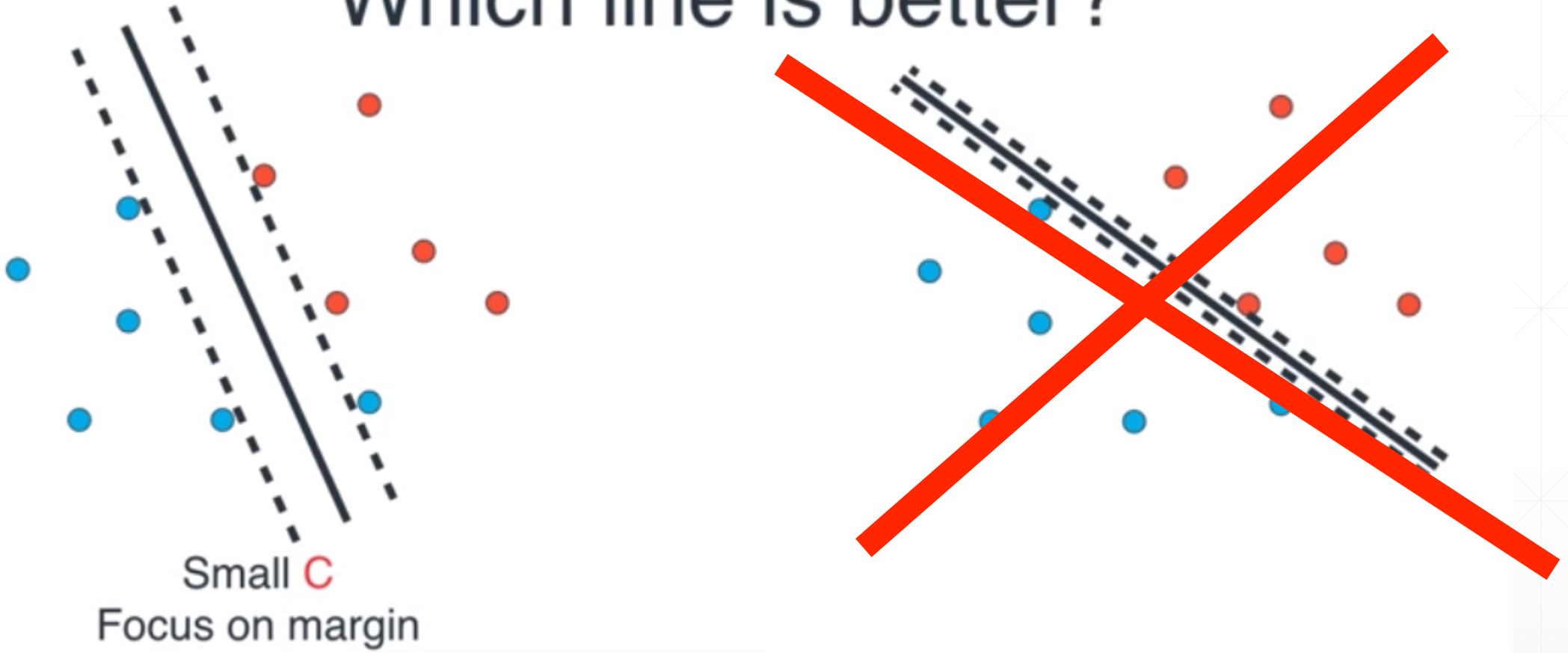
Classification Error + Margin Error

The C parameter

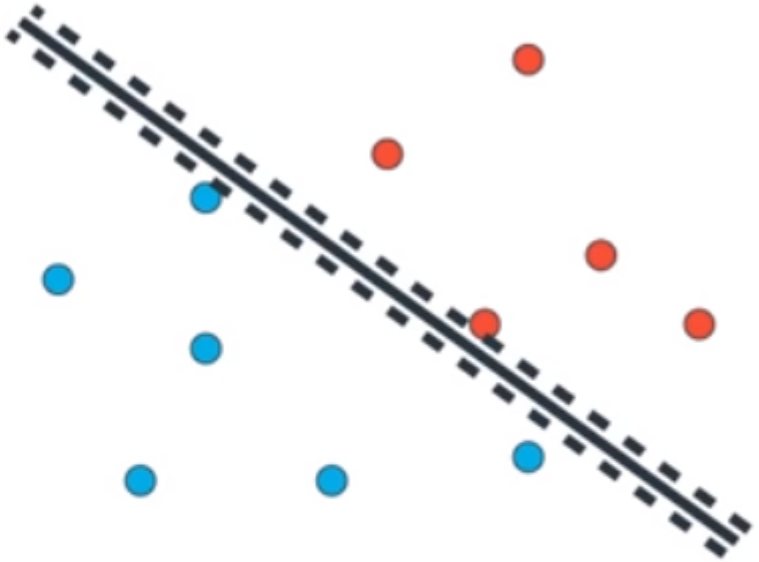
Which line is better?

C Classification Error + Margin Error

Which line is better?



Which line is better?

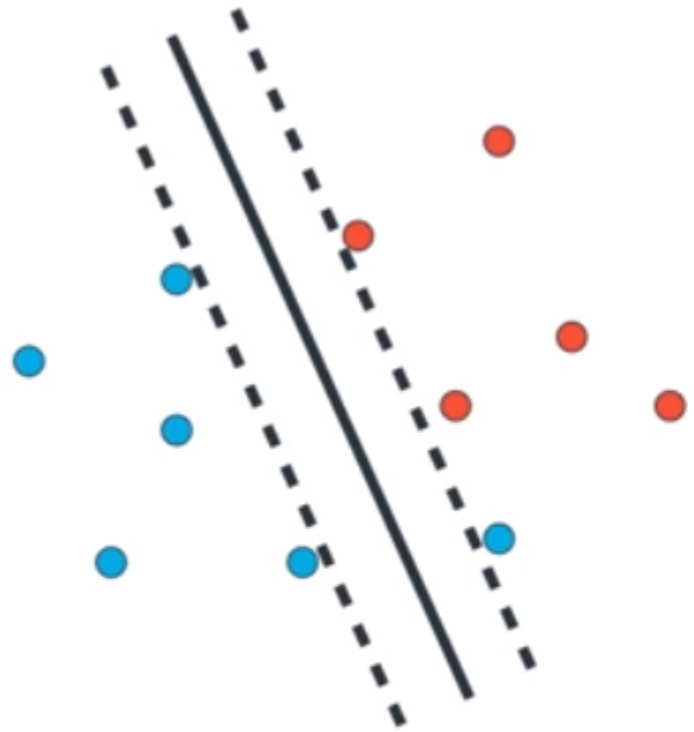


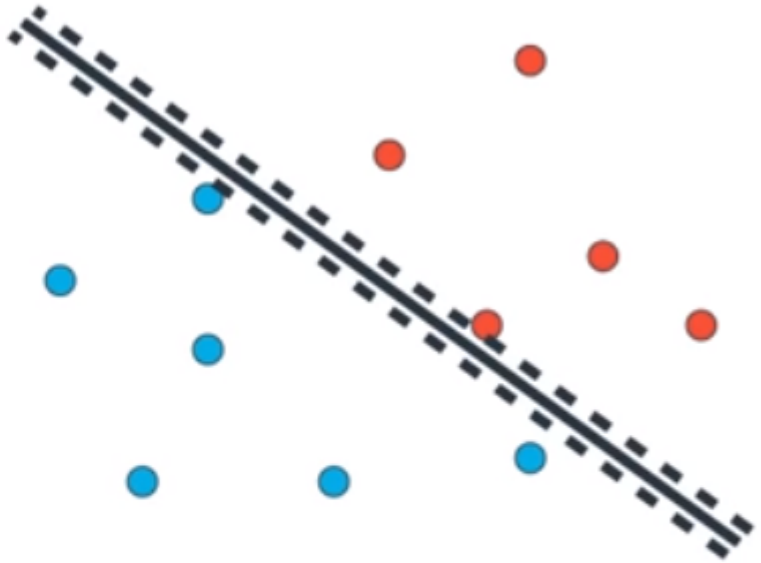
Large C
Focus on classification

C [Classification Error] + ~~[Margin Error]~~

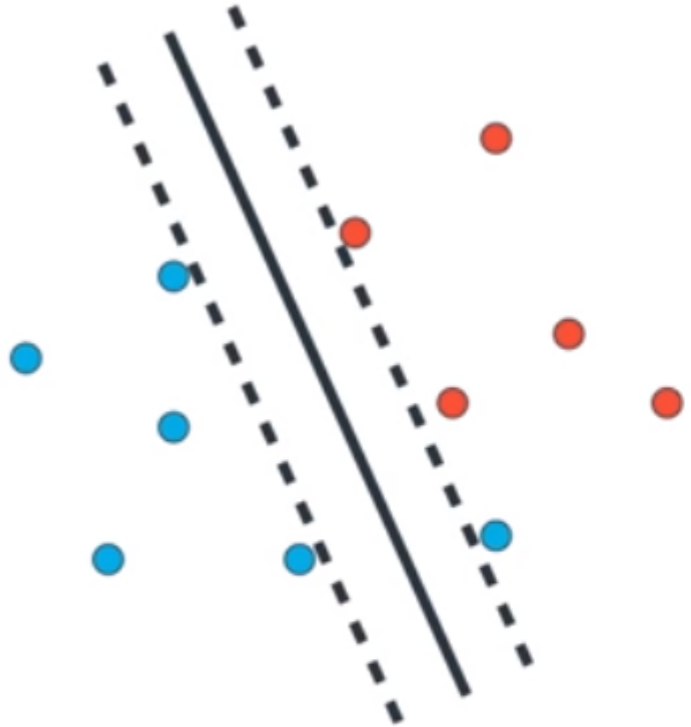
How does this interact with Bias - Variance Tradeoff?

How does this interact with Bias - Variance Tradeoff?

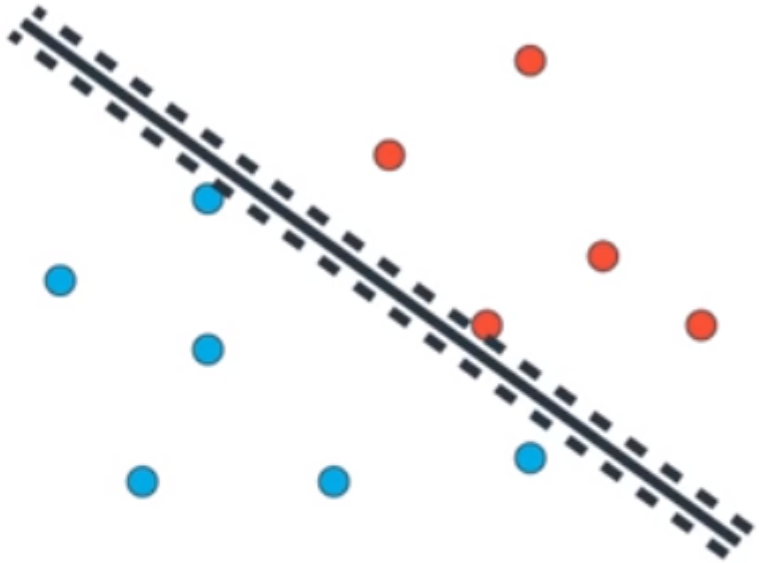




How does this interact with Bias - Variance Tradeoff?

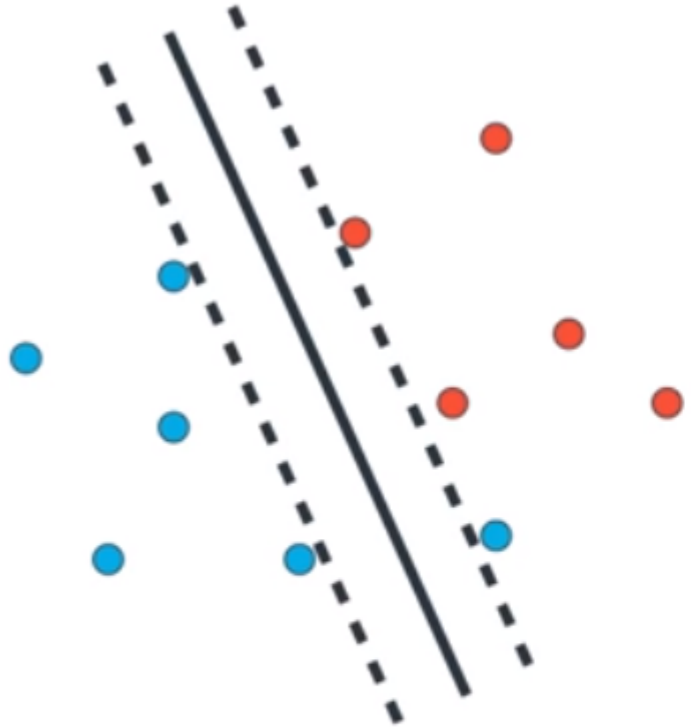


Small C



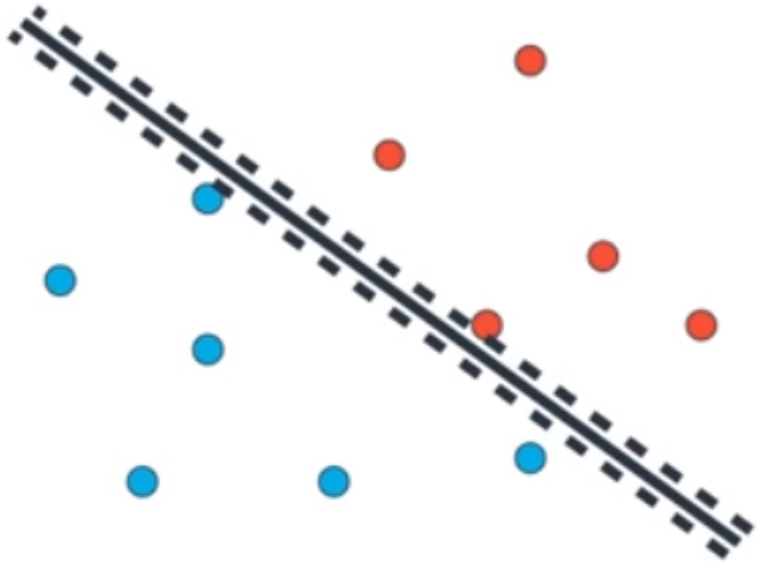
Large C

How does this interact with Bias - Variance Tradeoff?



Small C

High bias, low var



Large C

High var, low bias

Another way to think about the math of C and regularization:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \left\{ \sum_{i=1}^{n} \max\left[0, 1 - y_i f(x_i)\right] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

betas of softmax margins
(lines that equal 1 and -1)

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \left\{ \sum_{i=1}^{n} \max\left[0, 1 - y_i f(x_i)\right] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

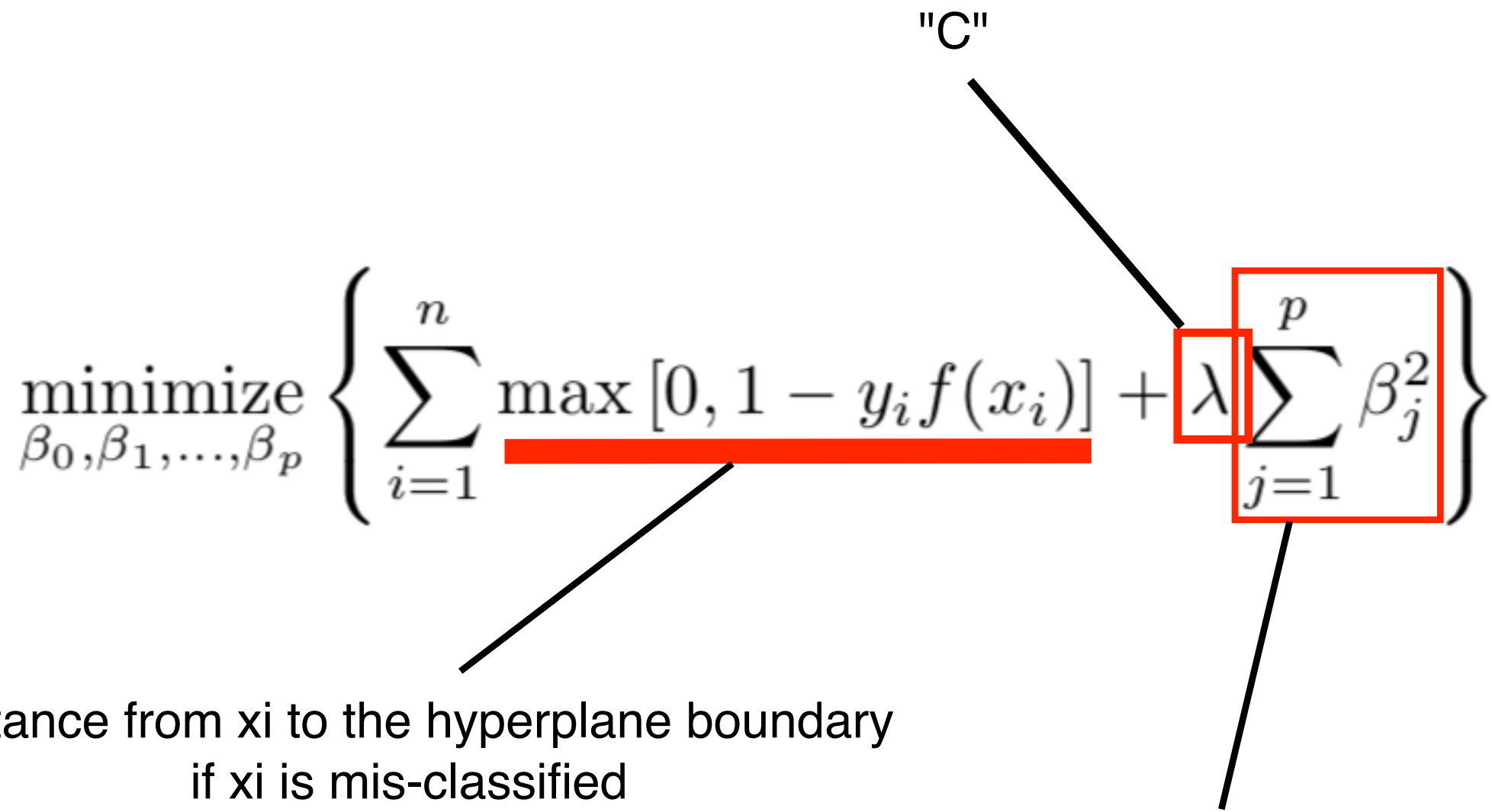
$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \left\{ \sum_{i=1}^{n} \max\left[0, 1 - y_i f(x_i)\right] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

distance from xi to the hyperplane boundary
if xi is mis-classified

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \left\{ \sum_{i=1}^{n} \max\left[0, 1 - y_i f(x_i)\right] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

distance from xi to the hyperplane boundary
if xi is mis-classified

regularization term (ie, width of the margin)


"C"
$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \left\{ \sum_{i=1}^{n} \max\left[0, 1 - y_i f(x_i)\right] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$
distance from xi to the hyperplane boundary
if xi is mis-classified
regularization term (ie, width of the margin)

As "C" increases, "penalty" to margin grows larger

"C"

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \left\{ \sum_{i=1}^{n} \max\left[0, 1 - y_i f(x_i)\right] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

As "C" increases, "penalty" to margin grows larger

"C"

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \left\{ \sum_{i=1}^{n} \max\left[0, 1 - y_i f(x_i)\right] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

Consequently:

- large "C" -> smaller margin
- small "C" -> larger margin

Look familiar?

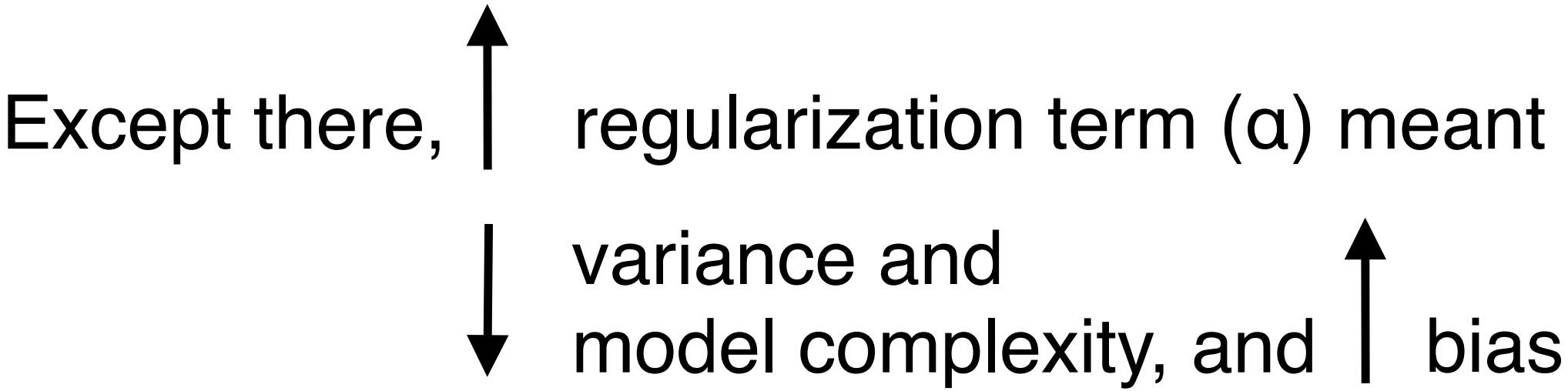
$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \left\{ \sum_{i=1}^{n} \max\left[0, 1 - y_i f(x_i)\right] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \left\{ \sum_{i=1}^{n} \max\left[0, 1 - y_i f(x_i)\right] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

Min Loss + Penalty

Like Ridge / Lasso!

Like Ridge / Lasso!



Like Ridge / Lasso!

Except there, ↑ regularization term (α) meant

↓ variance and
model complexity, and ↑ bias

Here, ↑ regularization term (C) means

↑ variance and
model complexity, and ↓ bias

Like Ridge / Lasso!

Except there, ↑ regularization term (α) meant

↓ variance and
model complexity, and ↑ bias

Here, ↑ regularization term (C) means

↑ variance and
model complexity, and ↓ bias

"C is the inverse of alpha"

2 Class -> Multiclass

2 Class -> Multiclass

Two strategies:

2 Class -> Multiclass

Two strategies:

One-against-one approach

2 Class -> Multiclass

Two strategies:

One-against-one approach

- n classes * (n classes - 1) / 2 models constructed
- data trained on 1 class v 1 class in these models
- different strategies to assign data classes based on model results

2 Class -> Multiclass

Two strategies:

One-against-one approach

- n classes * (n classes - 1) / 2 models constructed
- data trained on 1 class v 1 class in these models
- different strategies to assign data classes based on model results

One-against-all approach

2 Class -> Multiclass

Two strategies:

One-against-one approach

- n classes * (n classes - 1) / 2 models constructed
- data trained on 1 class v 1 class in these models
- different strategies to assign data classes based on model results

One-against-all approach

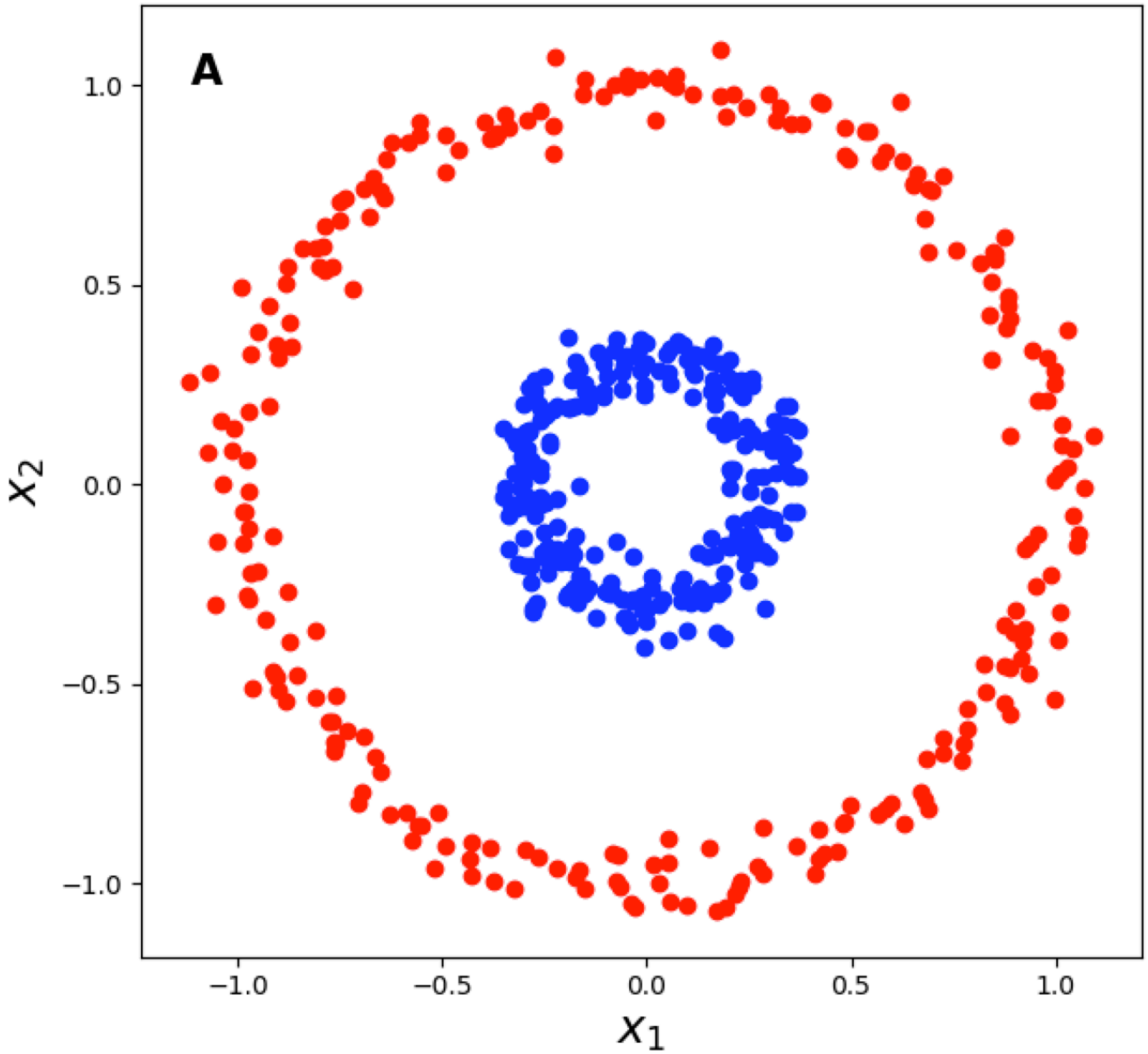
- n models constructed
- data trained on 1 class v all other classes in these models
- if data is not assigned to any class, goes to closest class

Algo 3:
Support Vector Machines

KERNALS!!!!!

KERNALS!!!!!

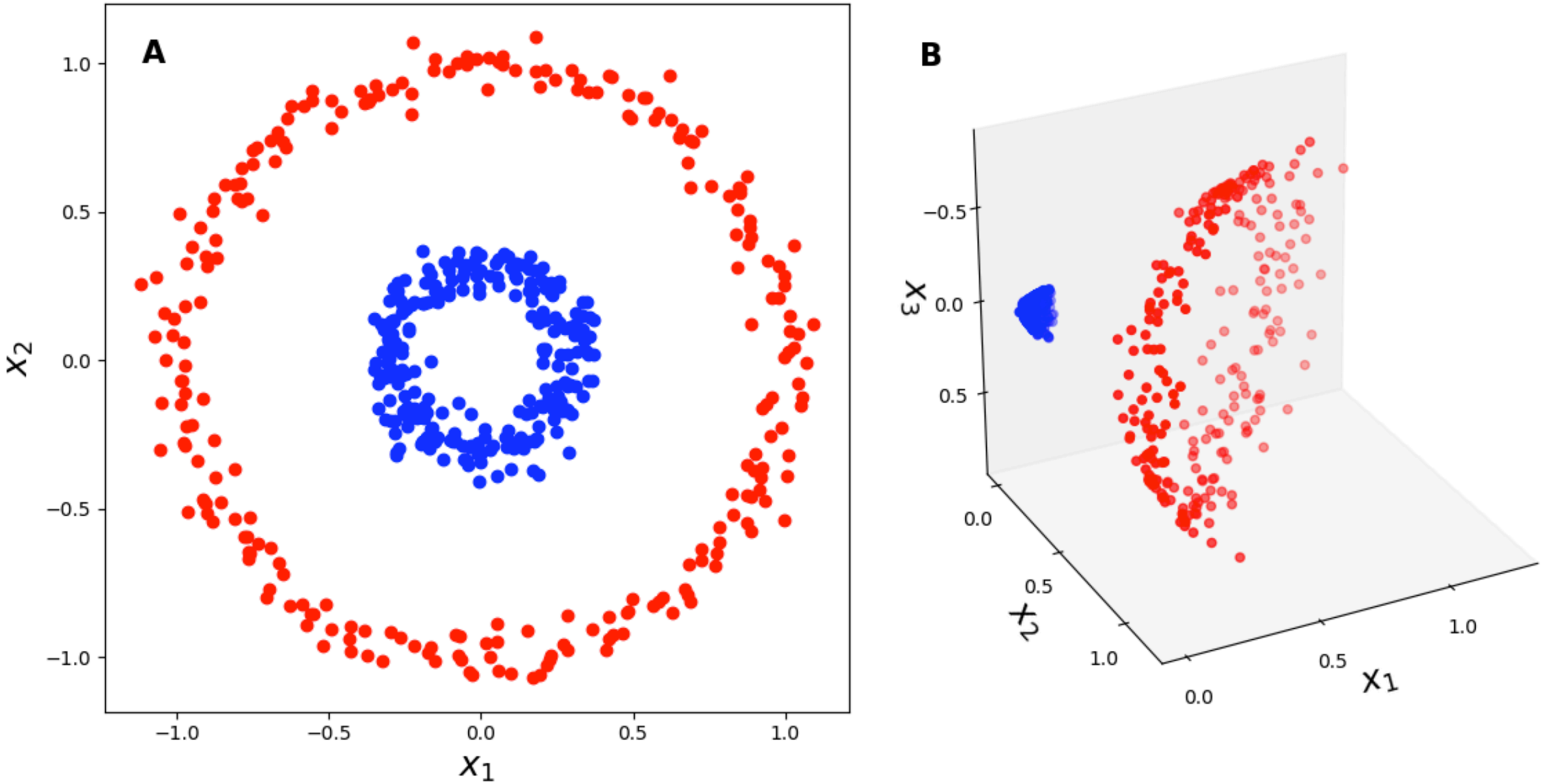
Problem: data shaped in way that makes decision boundary impossible


A
1.0
0.5
0.0
−0.5
−1.0
x_2
−1.0
−0.5
0.0
0.5
1.0
x_1

KERNALS!!!!!

Problem: data shaped in way that makes decision boundary impossible

Solution: add features that enable extension of data into another space


A
B
1.0
0.5
0.0
−0.5
−1.0
x_2
x_1
x_3
x_2
x_1

KERNALS!!!!!

Main kernals used:

KERNALS!!!!!

Main kernals used:

- poly - uses polynomial functions

KERNALS!!!!!

Main kernals used:

- poly - uses polynomial functions
- rbf - projects regions of space based on class density

KERNALS!!!!!

Kernals used frequently in other techniques to separate boundaries or enable complexity beyond linearity

- PCA

CAREFUL: prone to overfitting

Discussion

Thank you!