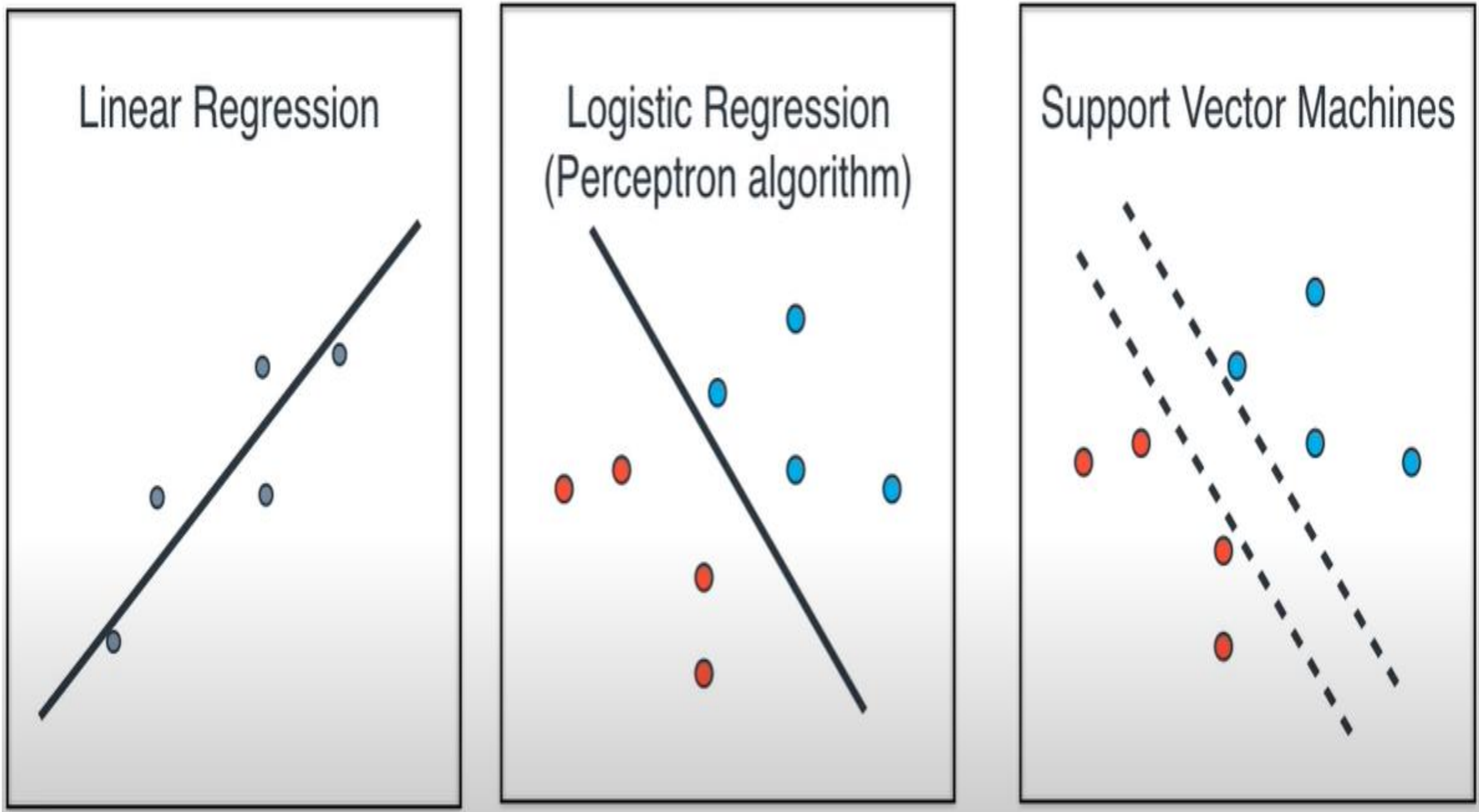
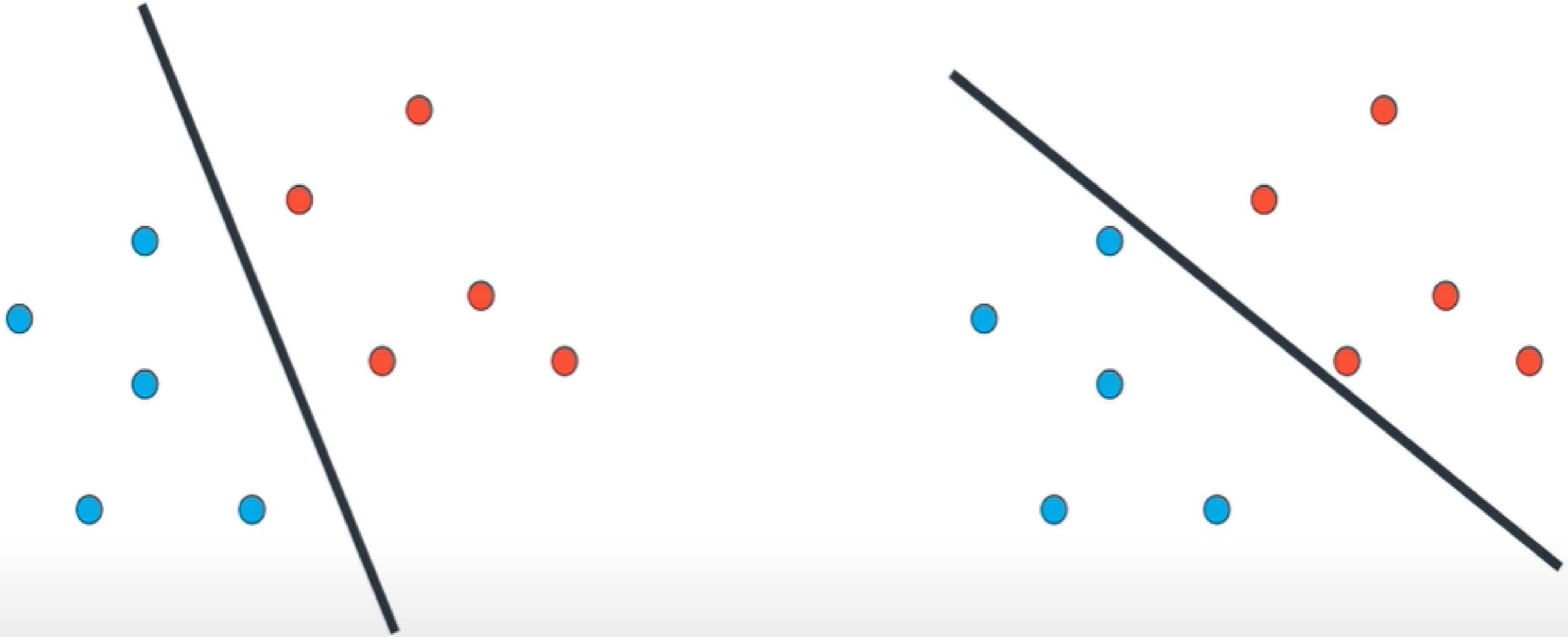
SVM

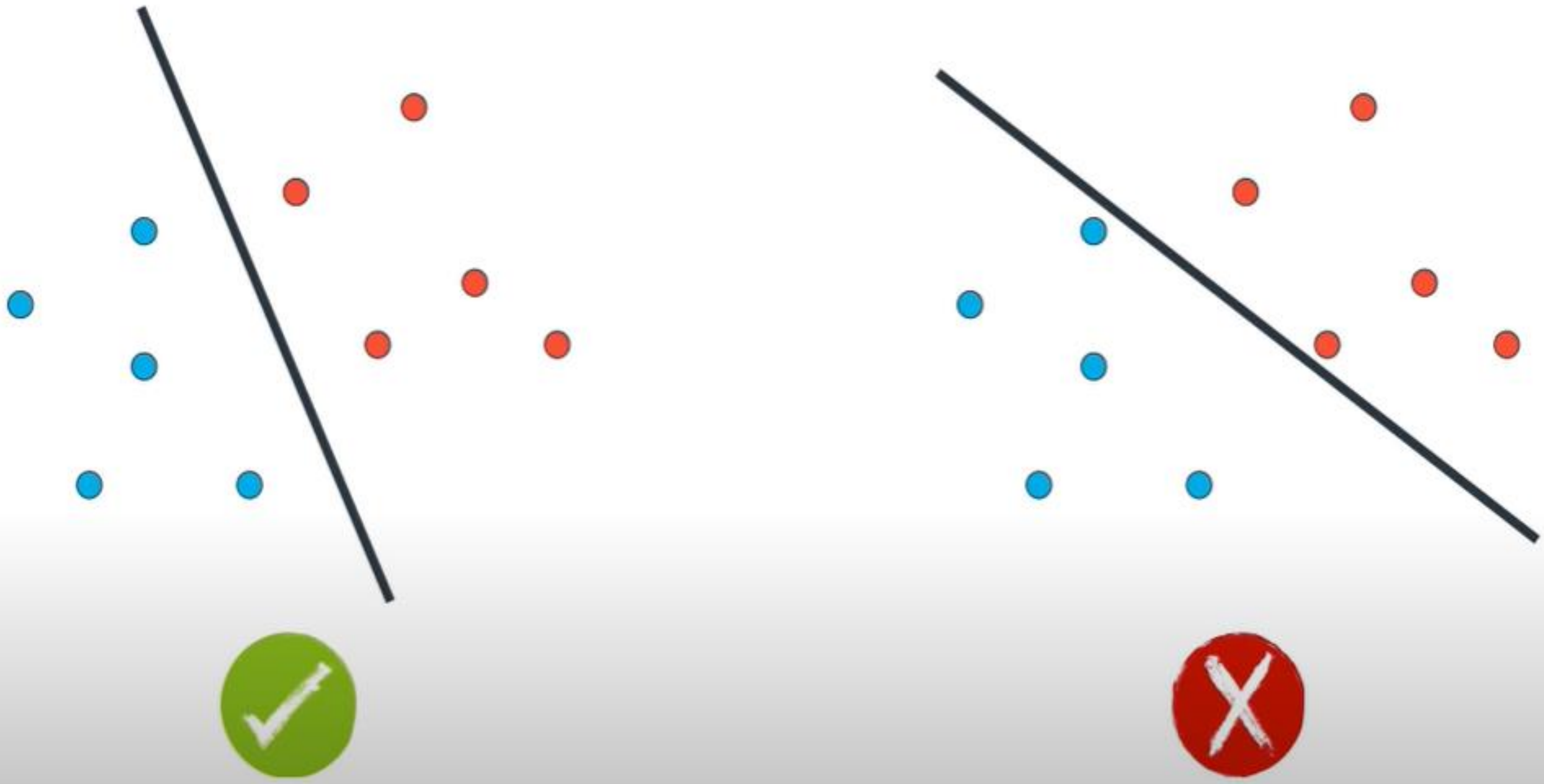
SUPPORT VECTOR MACHINE


Linear Regression
Logistic Regression
(Perceptron algorithm)
Support Vector Machines

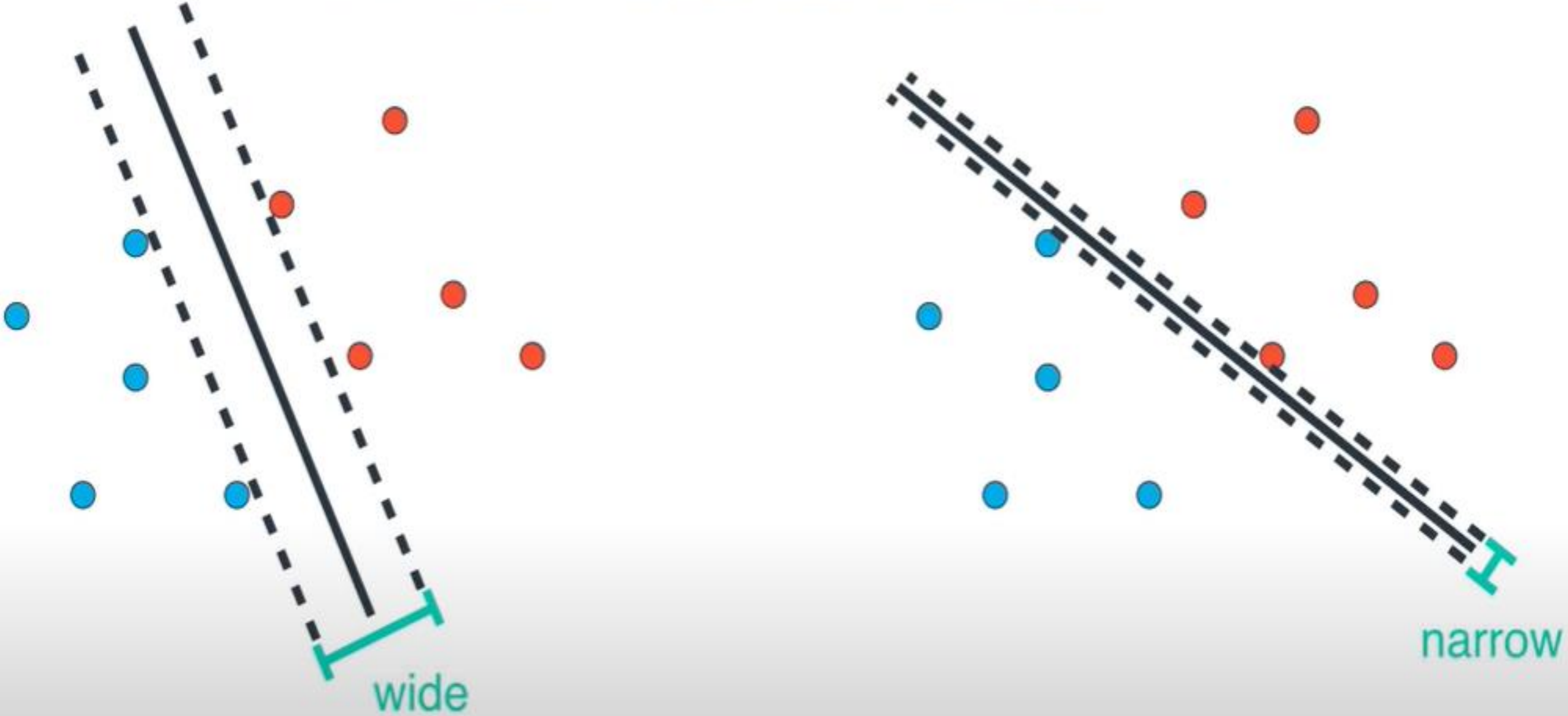
Which line is better?



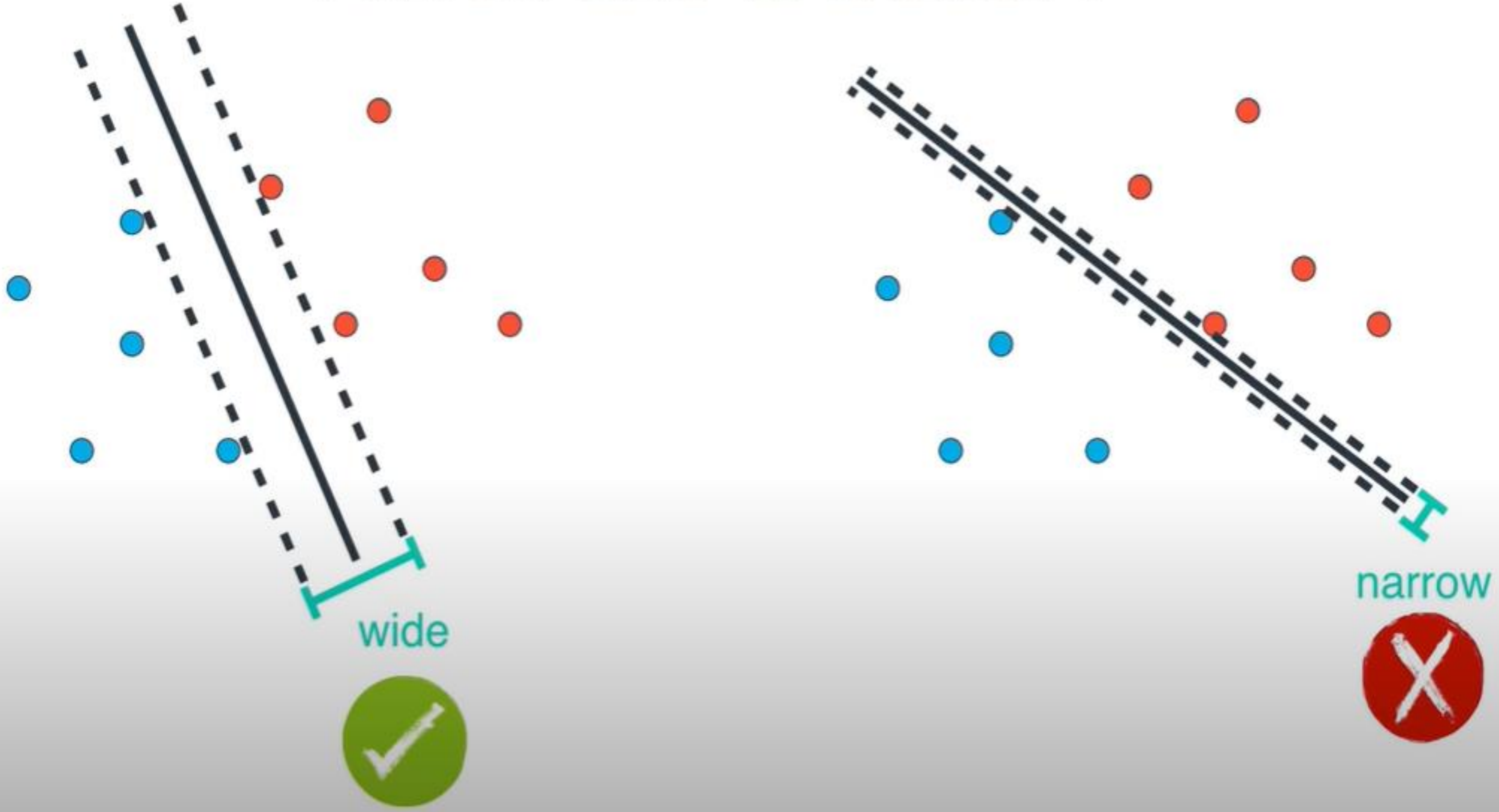
Which line is better?



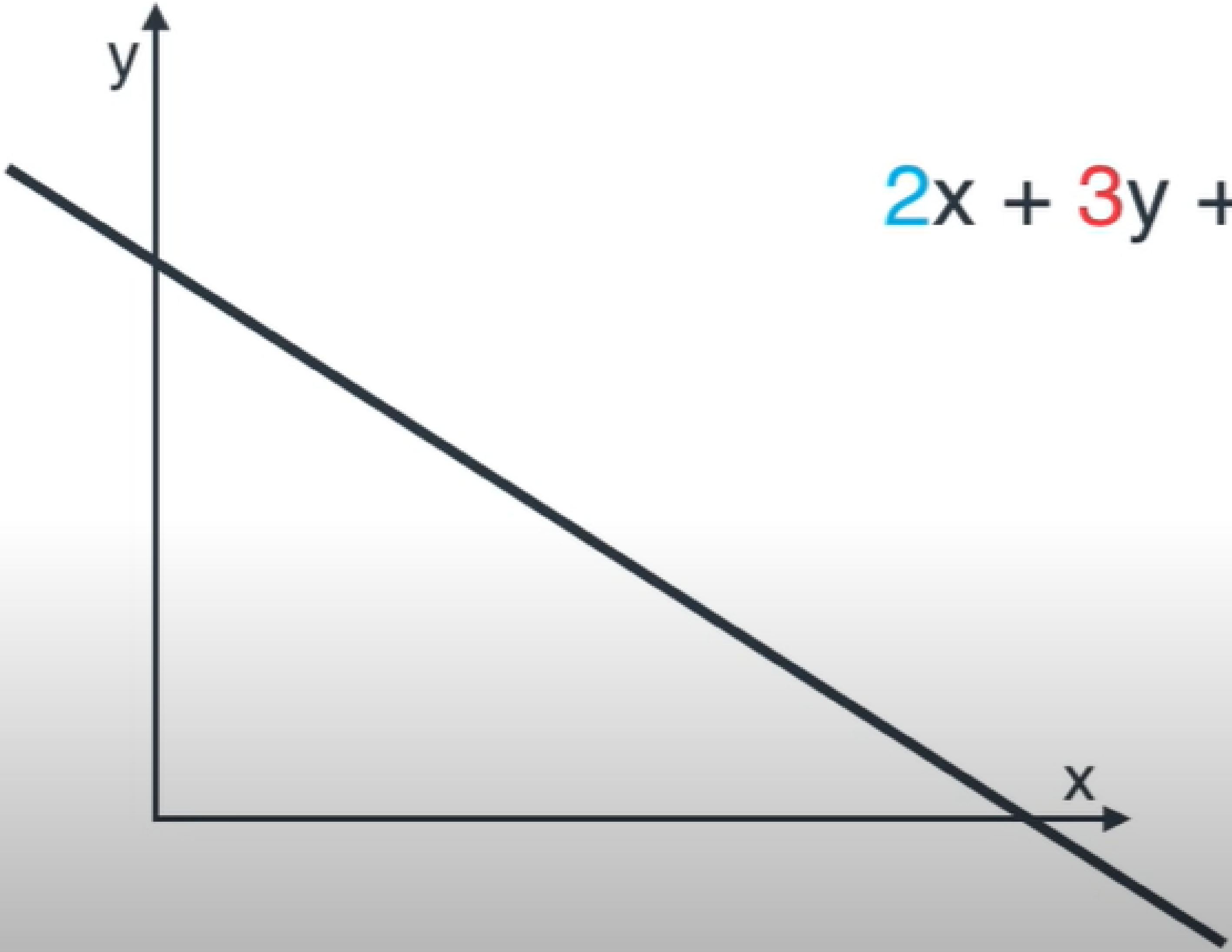
Which line is better?



Which line is better?

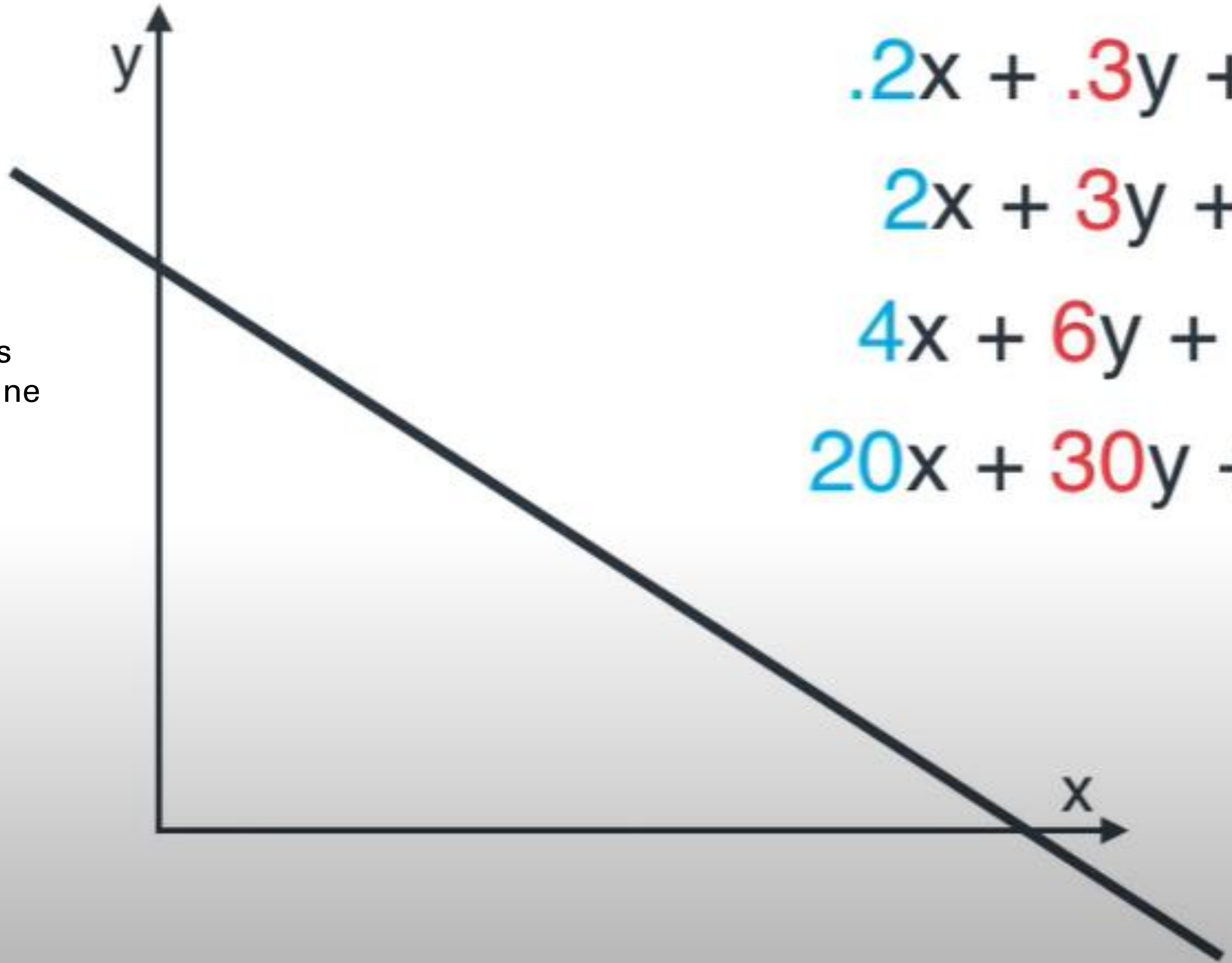


How to separate lines?



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

How to separate lines?



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

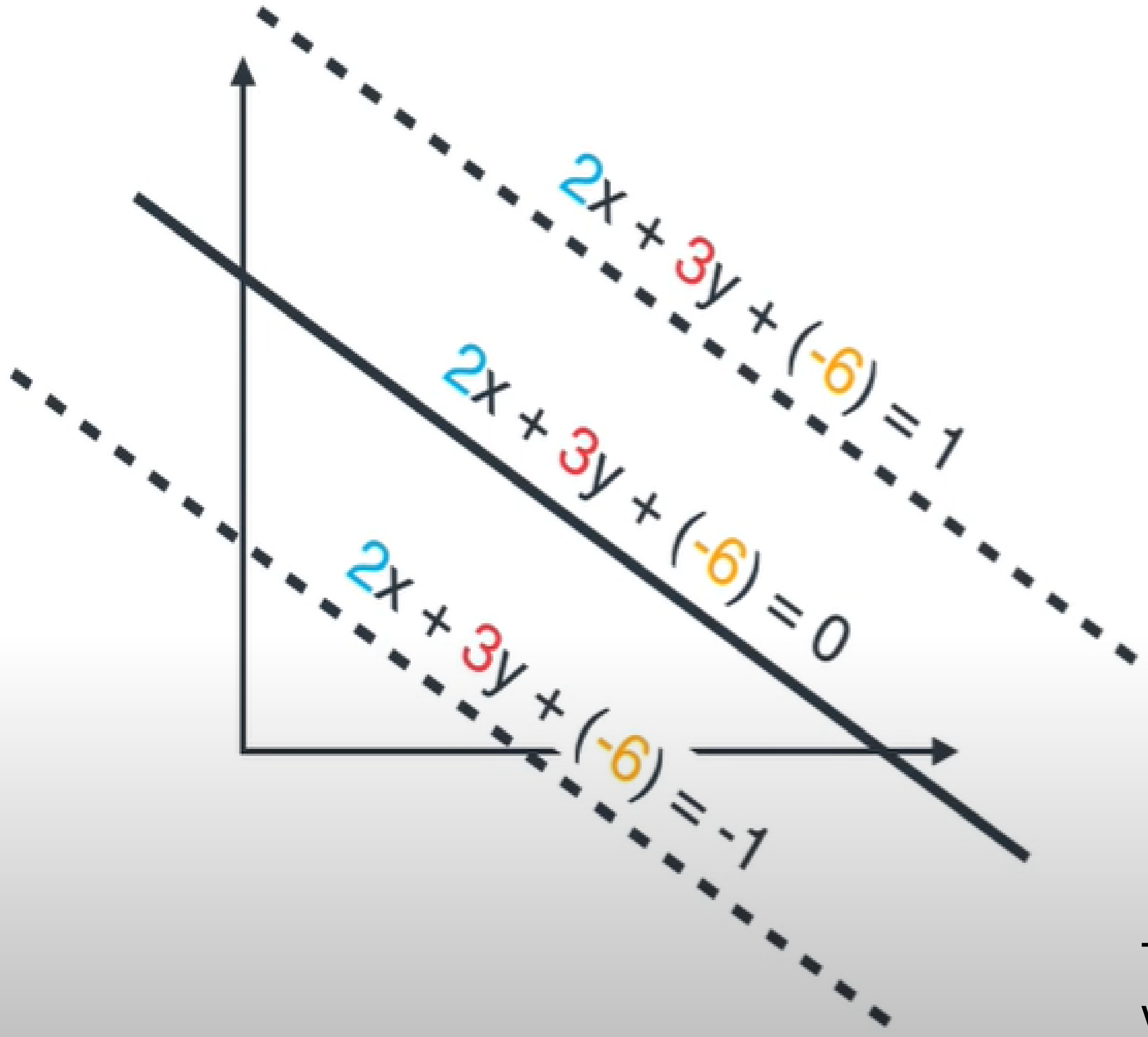
$$2x + 3y + (-6) = 0$$

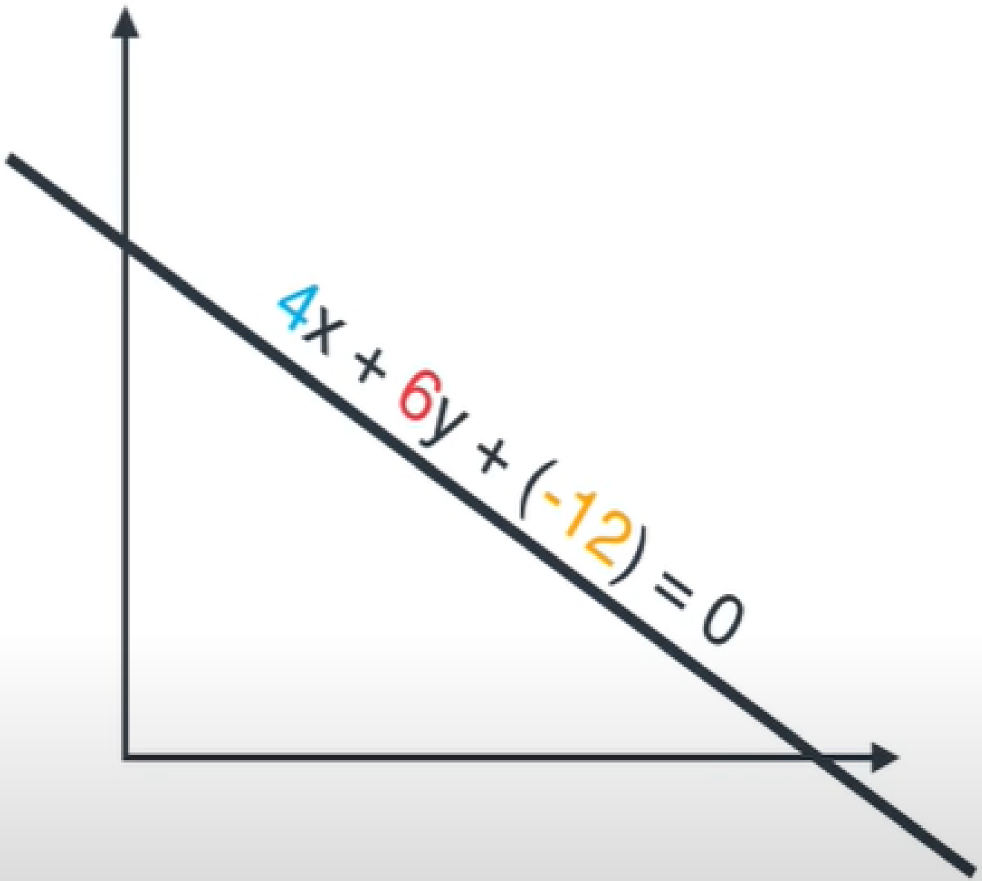
$$4x + 6y + (-12) = 0$$

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

Note: All these equations alien / satisfy the same line

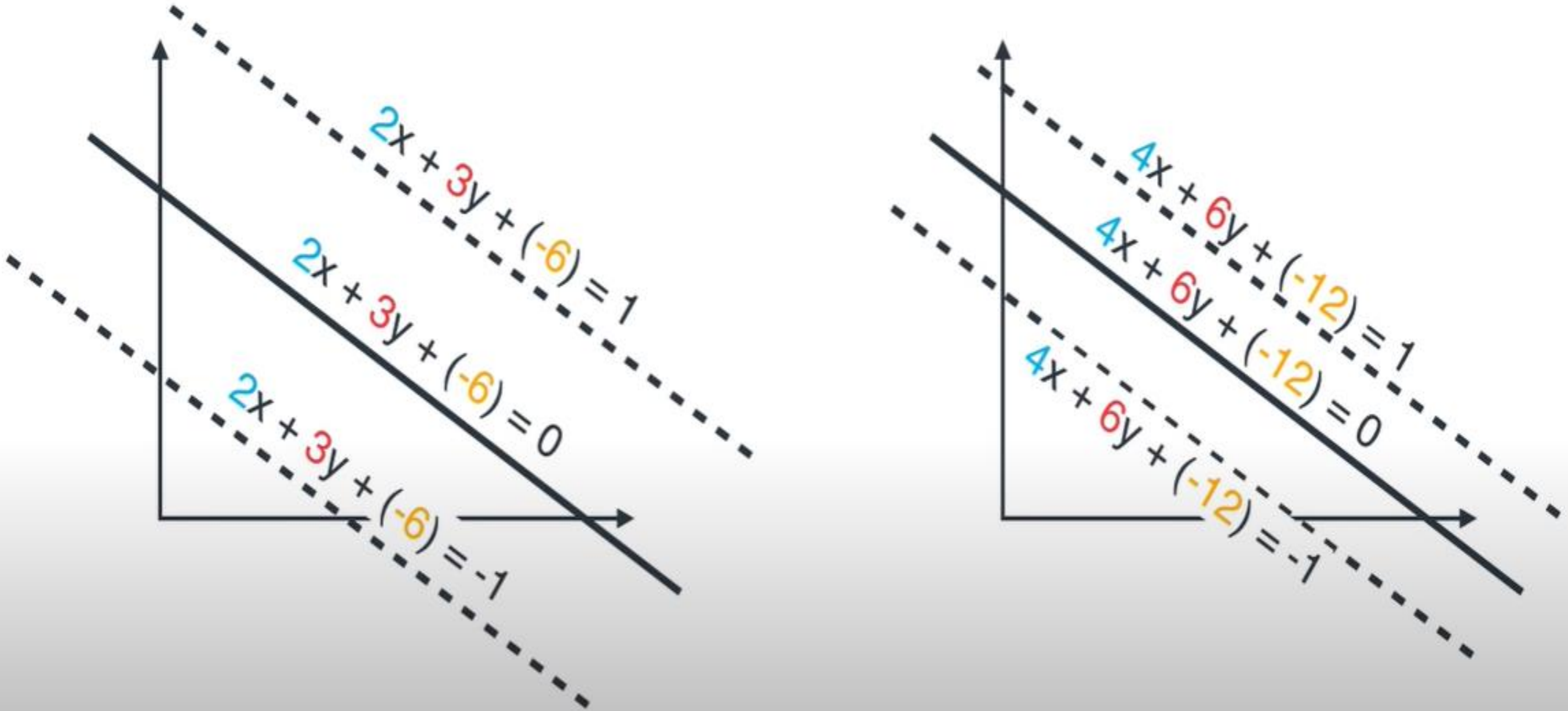
How to separate lines?



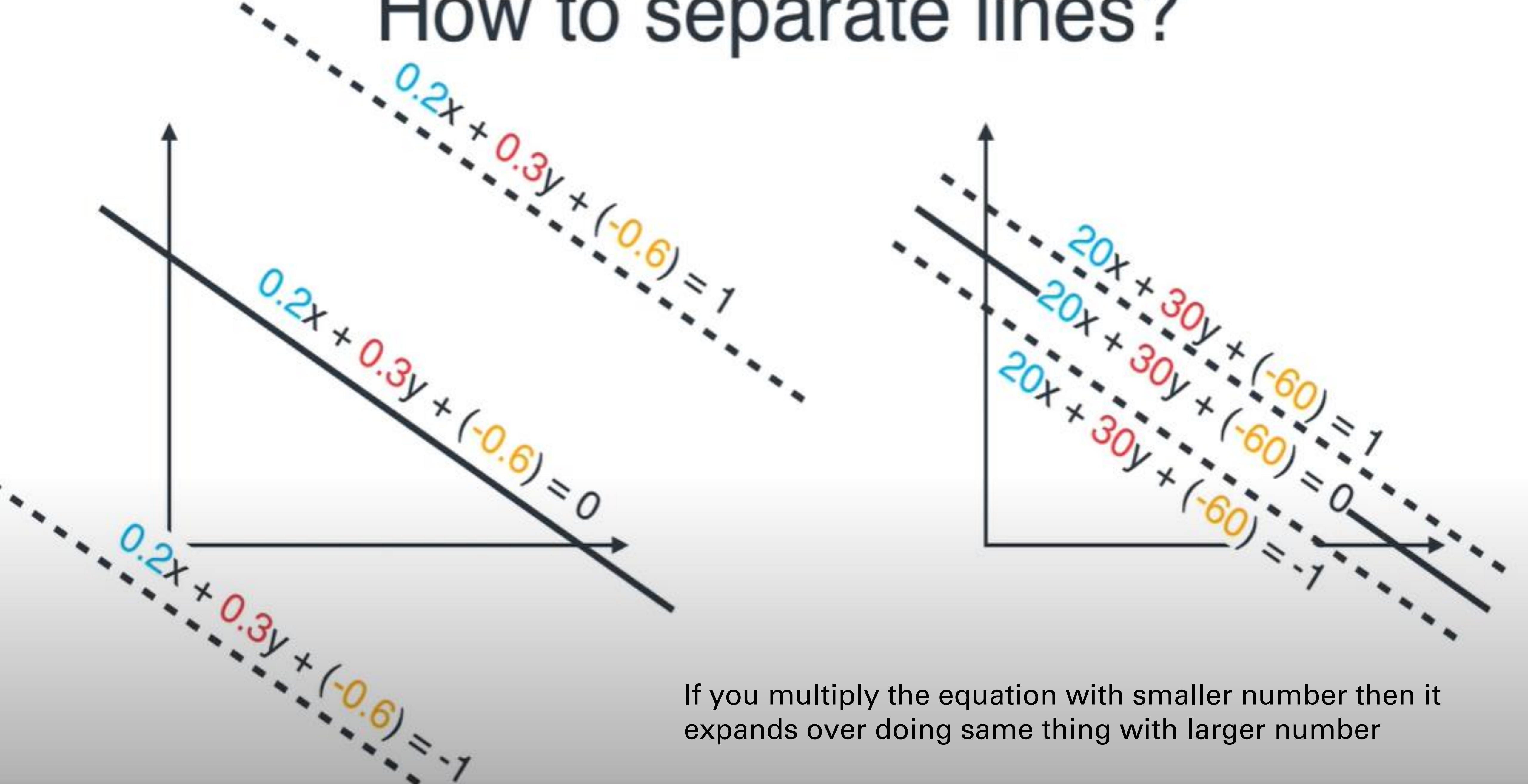


This is what actually changes even though we have different coefficients of same line one with left hand and right hand side

How to separate lines?



How to separate lines?

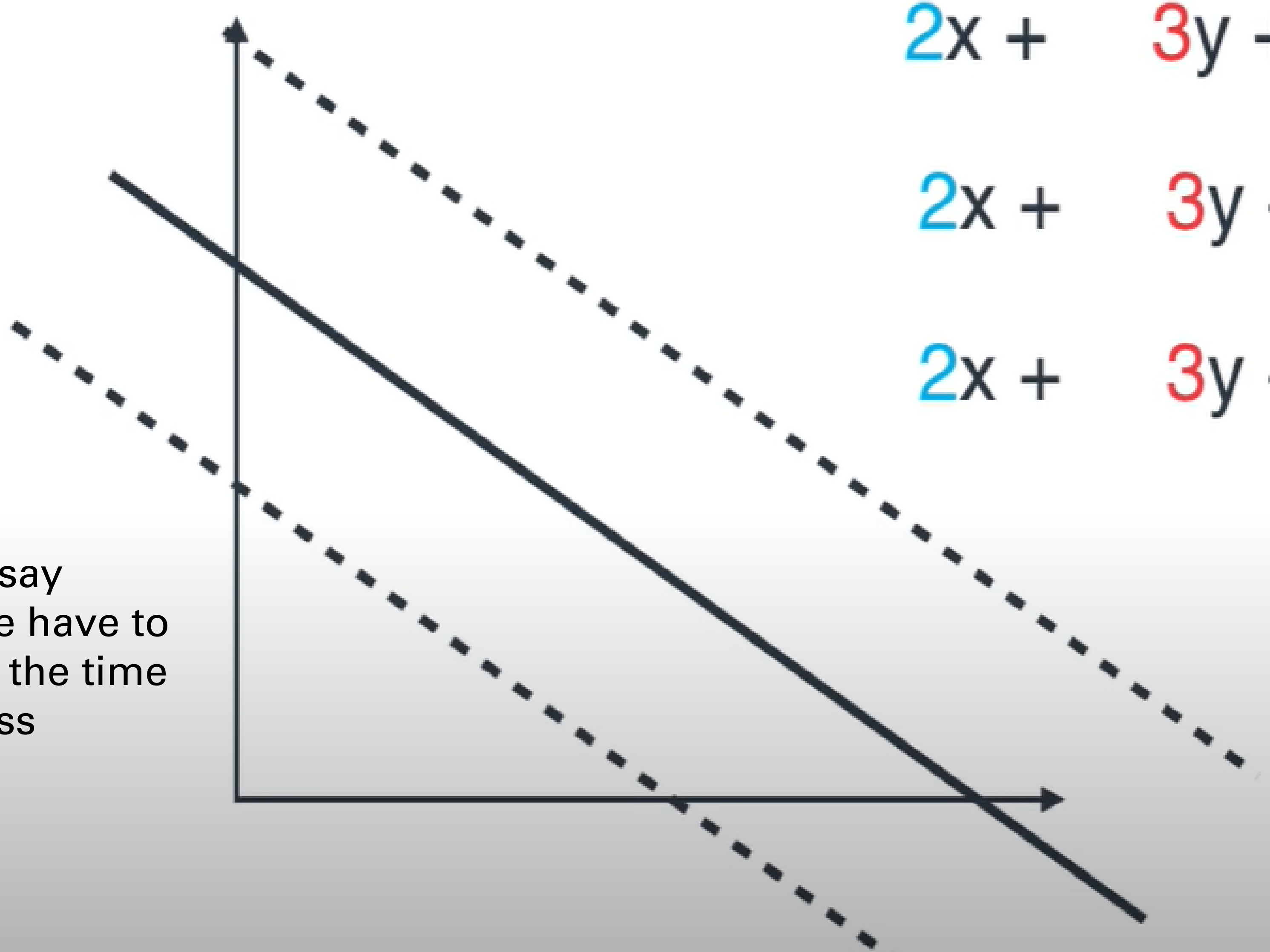


If you multiply the equation with smaller number then it expands over doing same thing with larger number

Expanding rate

Expanding rate

0.99



$$2x + 3y + (-6) = -1$$

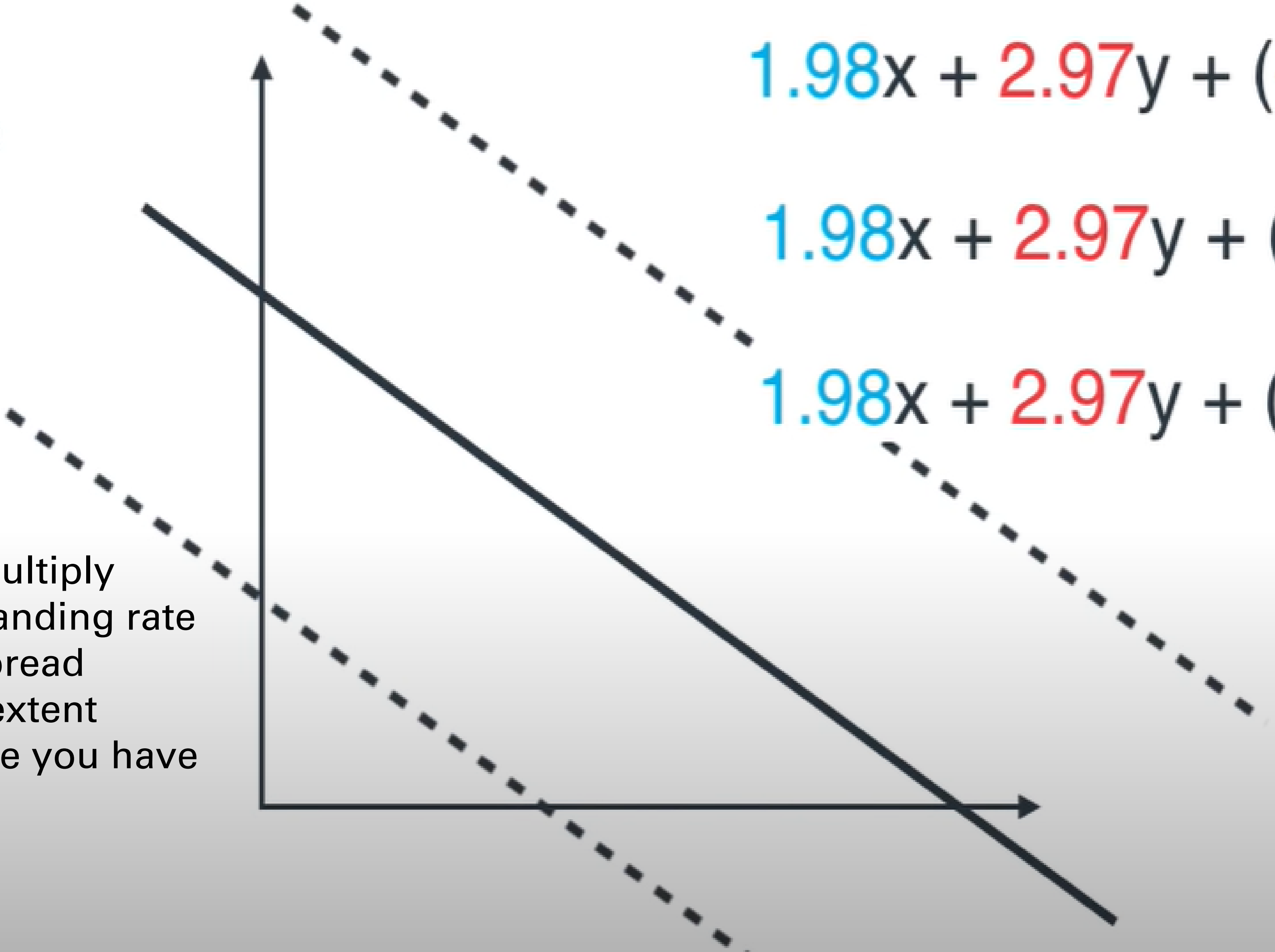
$$2x + 3y + (-6) = 0$$

$$2x + 3y + (-6) = 1$$

Expanding rate is lets say learning rate in ML. we have to take smaller values all the time to start with the process

Expanding rate

Expanding rate



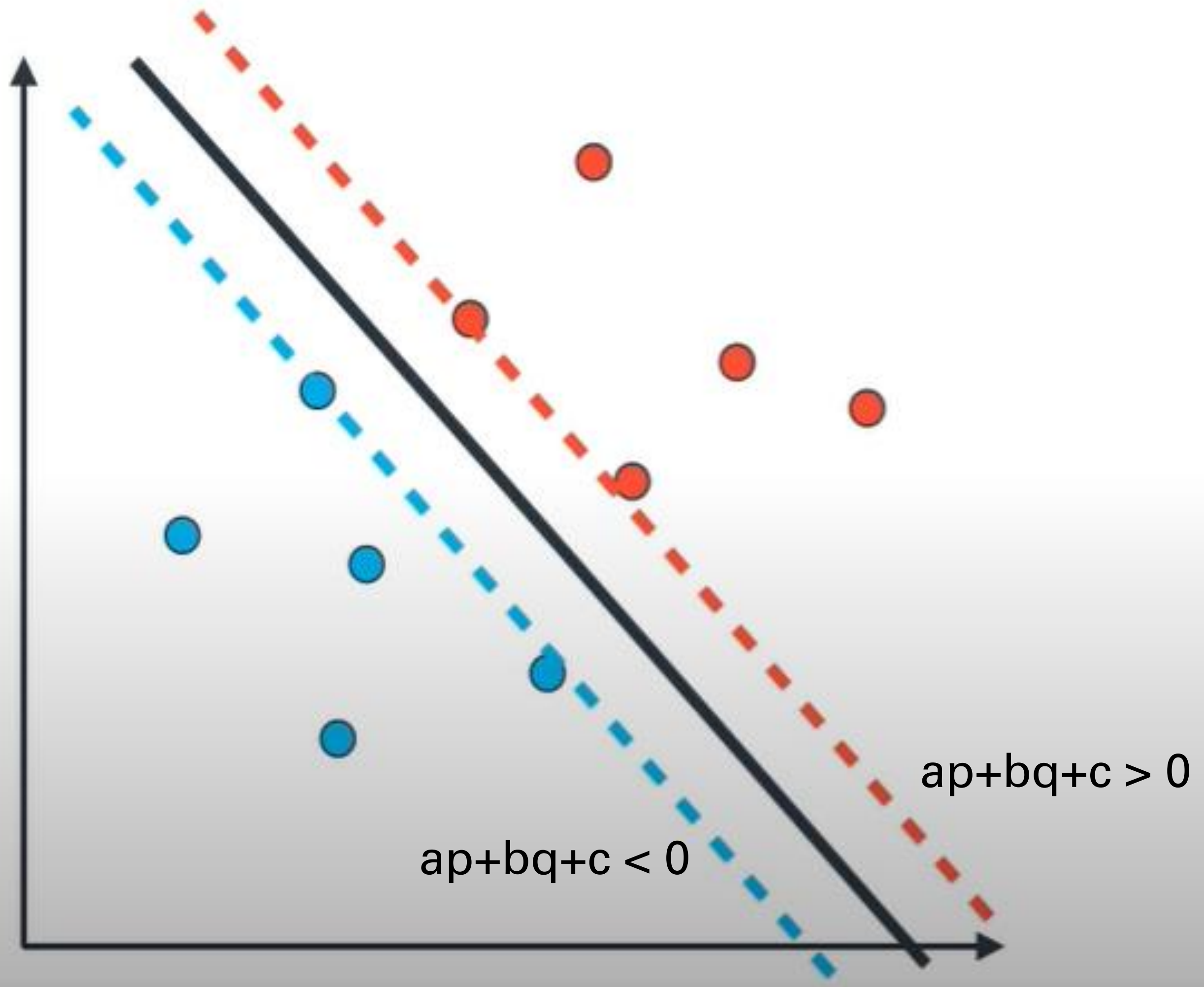
$$1.98x + 2.97y + (-5.94) = -1$$

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

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

The moment you multiply equations with expanding rate of 0.99 then your spread increases up to an extent compared to the one you have

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)

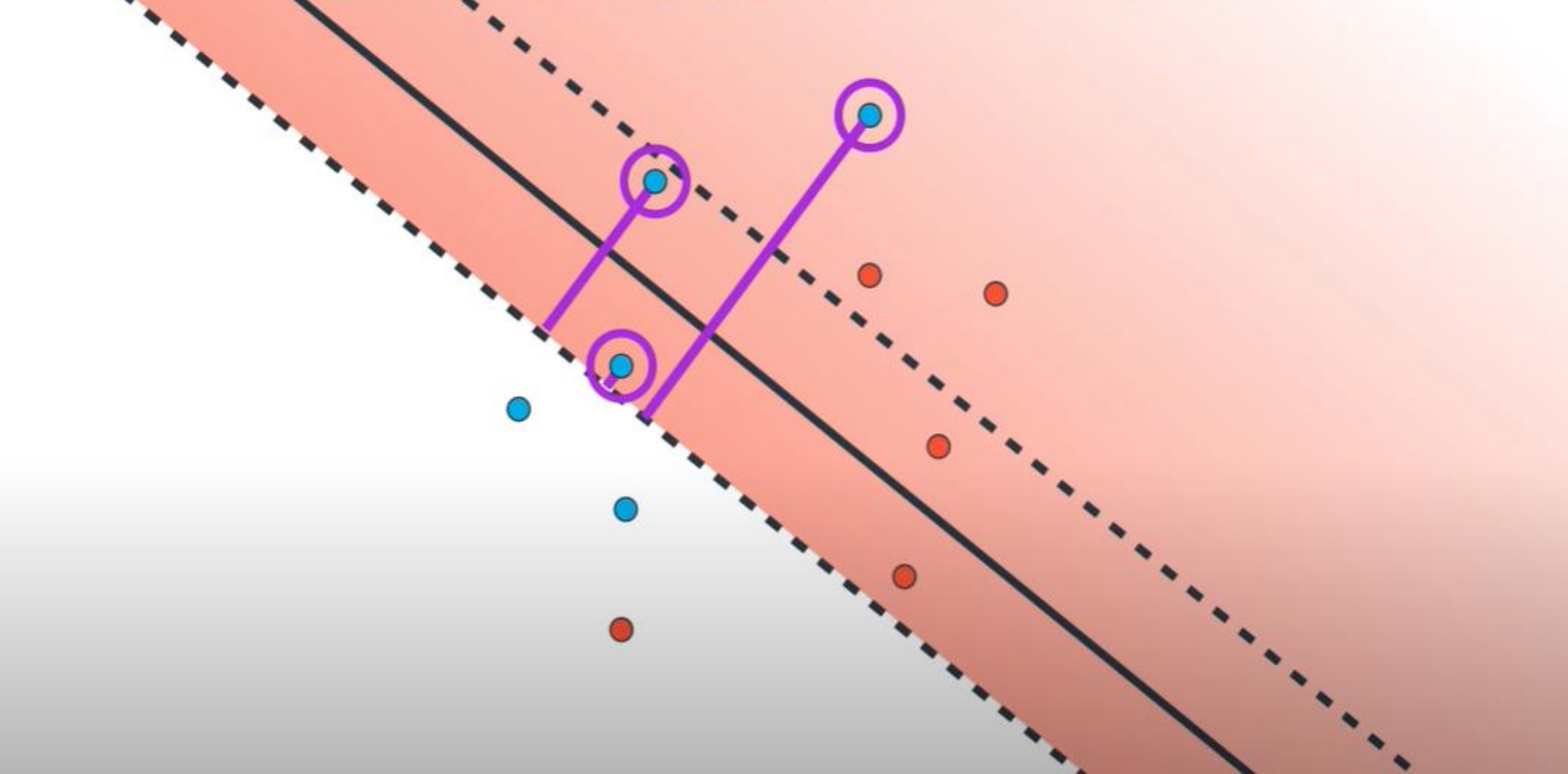
Step 3: Pick a learning rate. **0.01**

Step 4: Pick an expanding rate. **0.99**

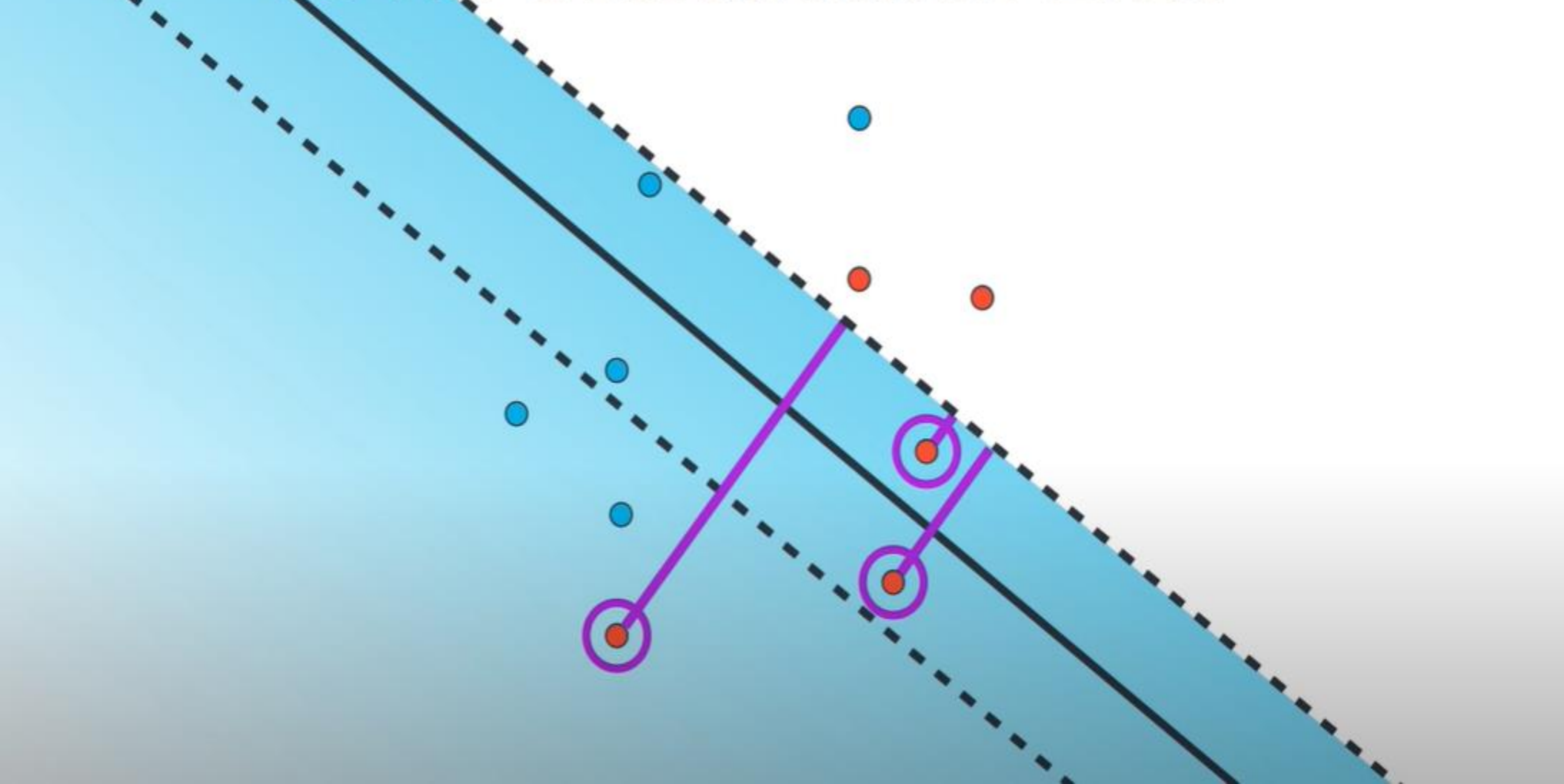
Step 5: (repeat **1000** times)

- 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

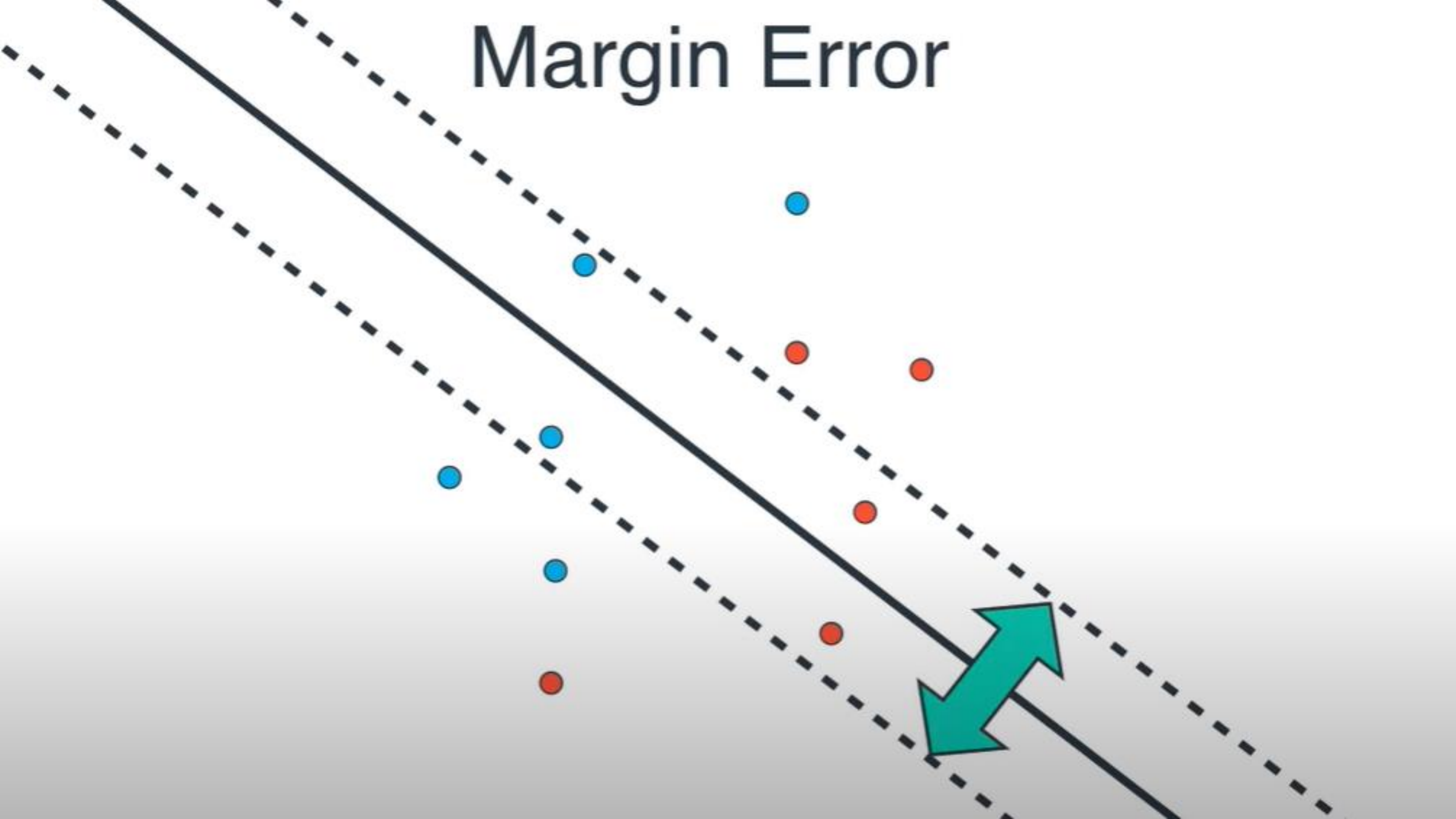
SVM Classification Error



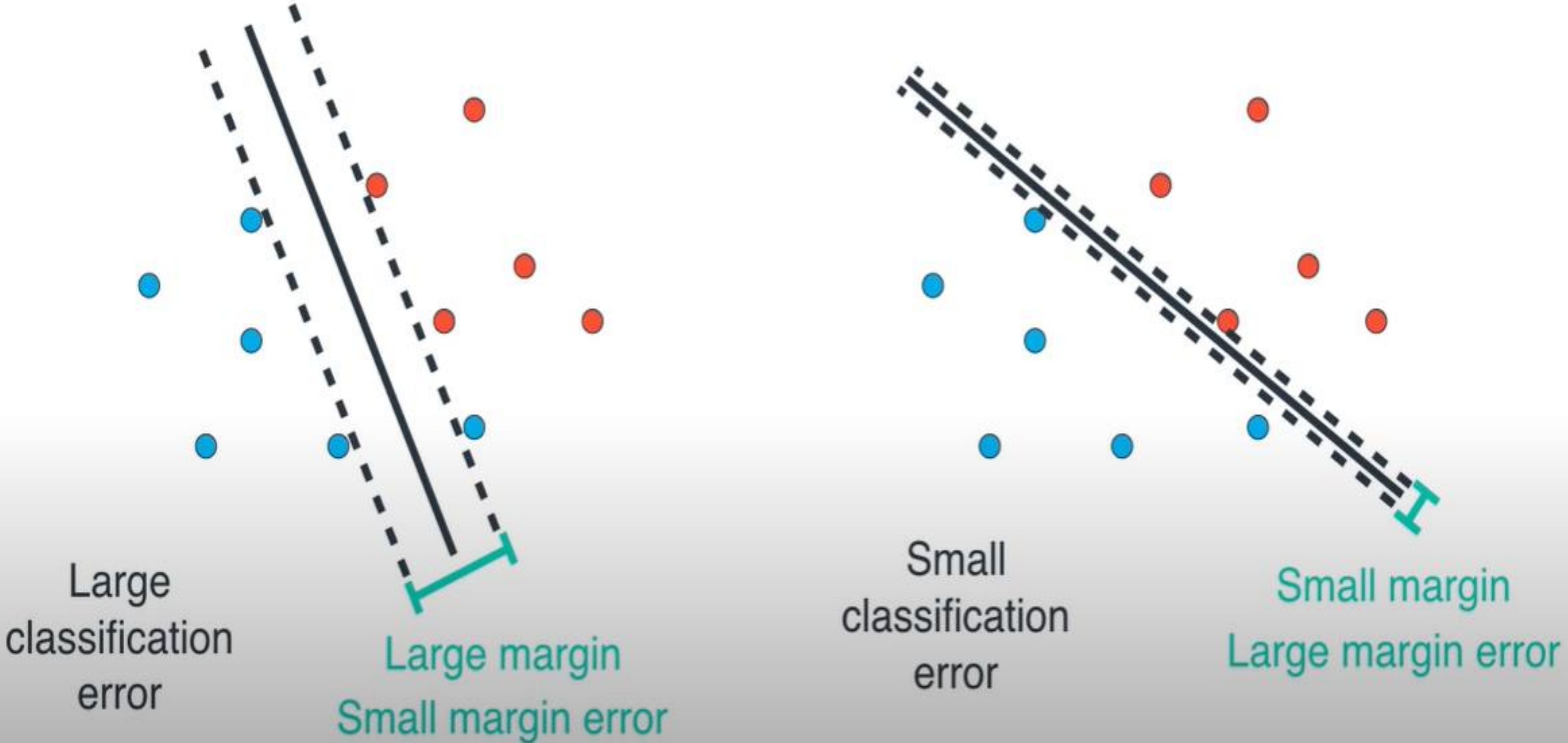
SVM Classification Error



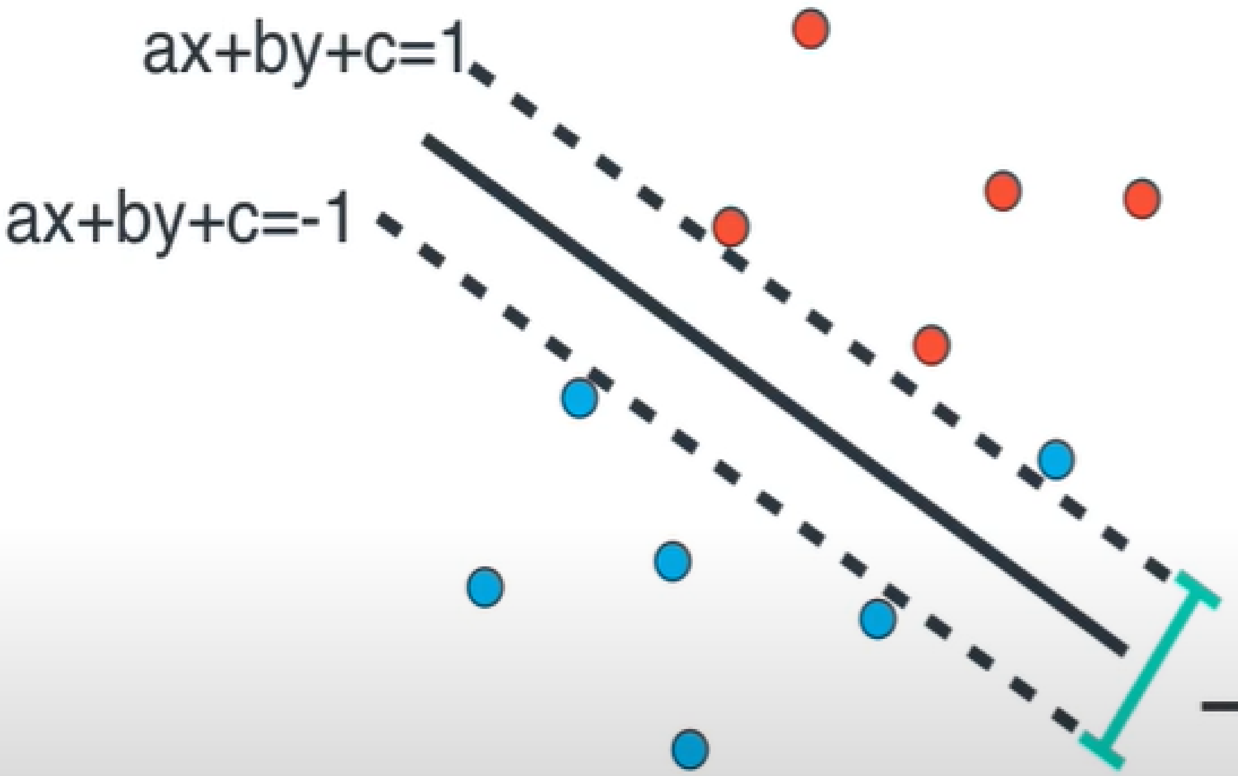
Margin Error



Margin Error

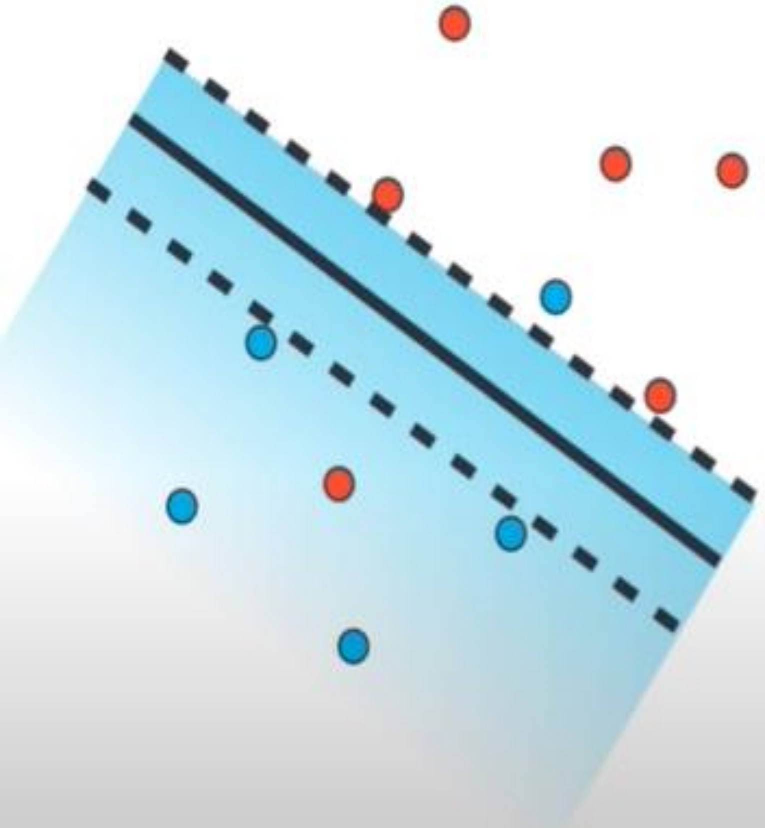


Margin Error

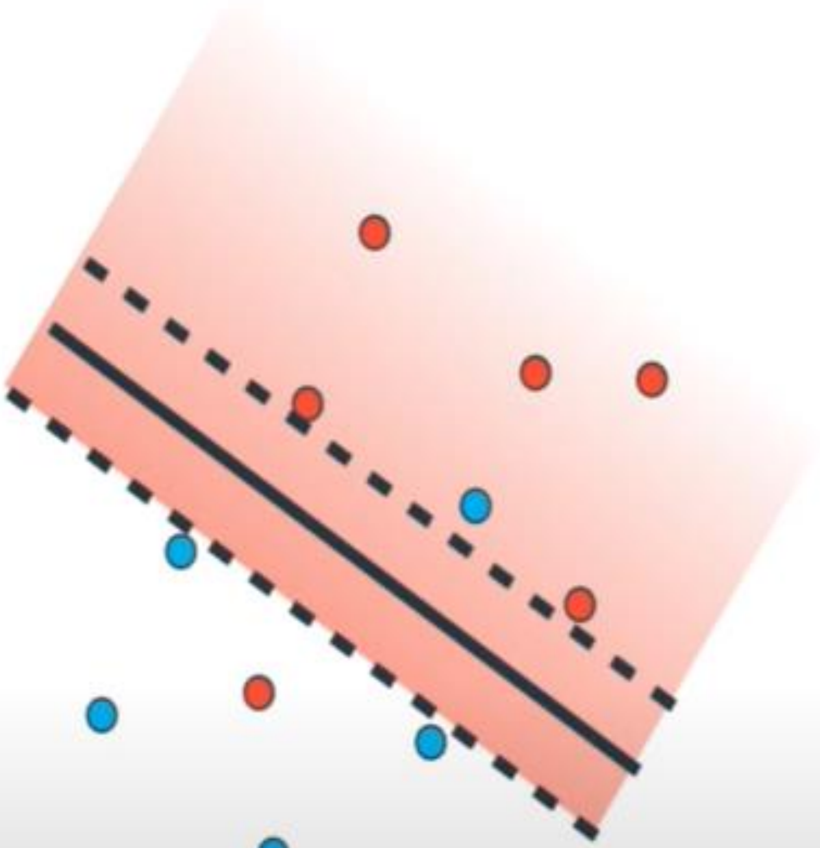


$$\frac{2}{\sqrt{a^2 + b^2}}$$

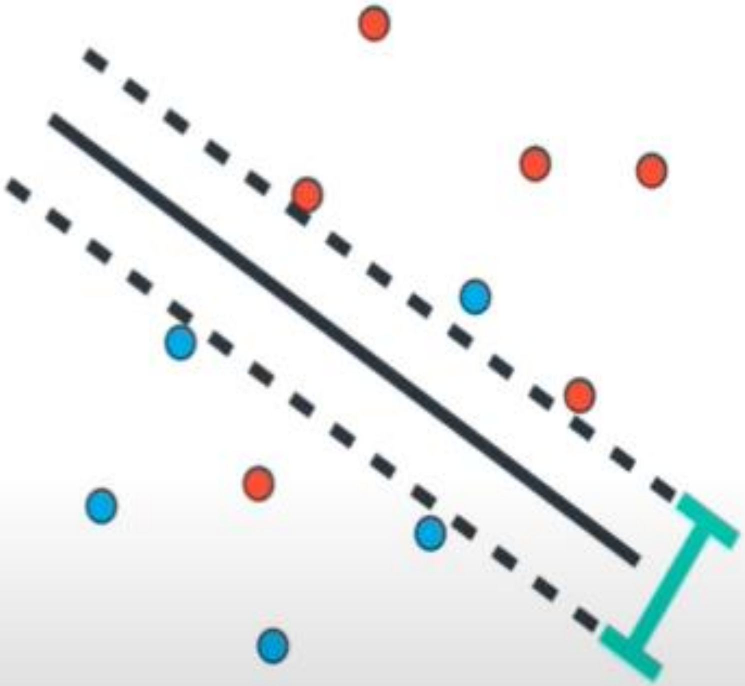
SVM Error



Blue Classification Error

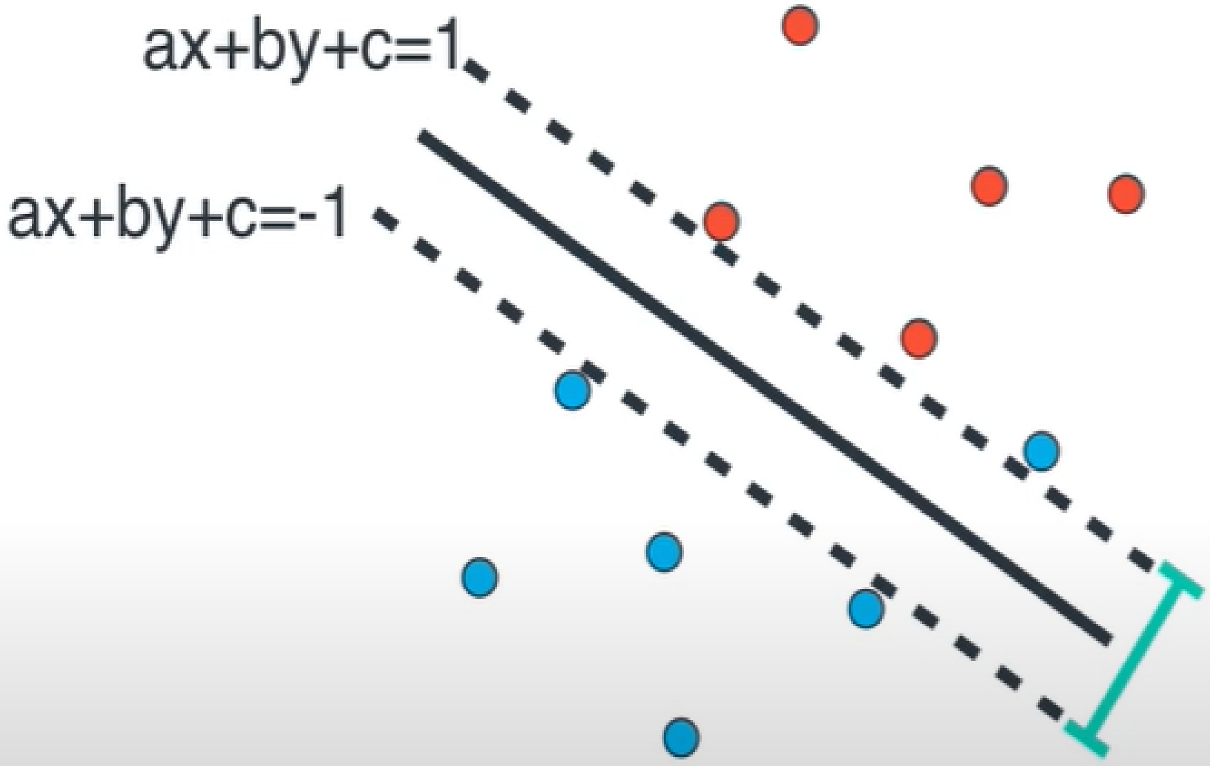


Red Classification Error



Margin Error

Challenge - Gradient Descent



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

expanding factor!

$dError/da = 2a$

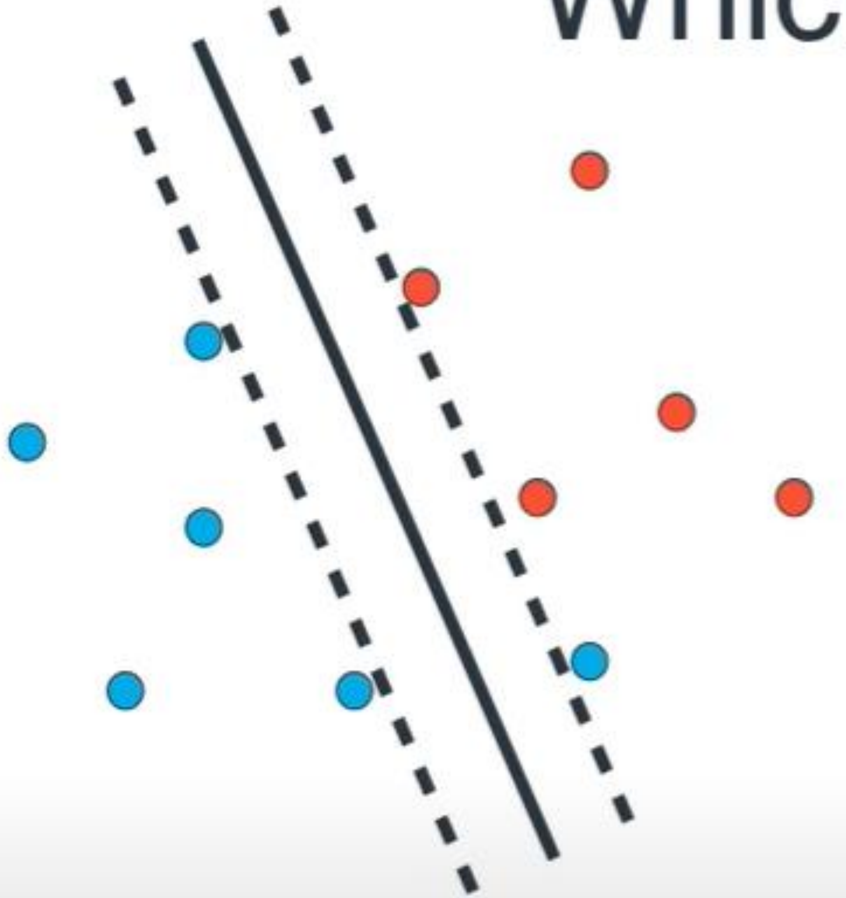
$dError/db = 2b$

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

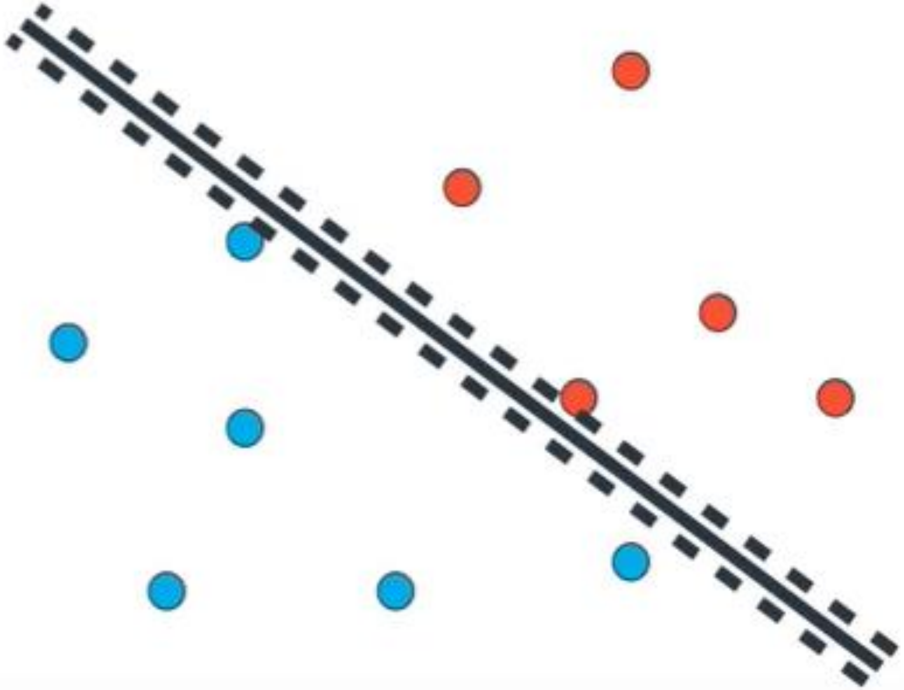
$$b \longrightarrow b - \eta\, 2b = b\boxed{(1-2\eta)}$$

Reminder: Theta new := theta old – learning rate (d/d theta(J))

Which line is better?



Small C
Focus on margin

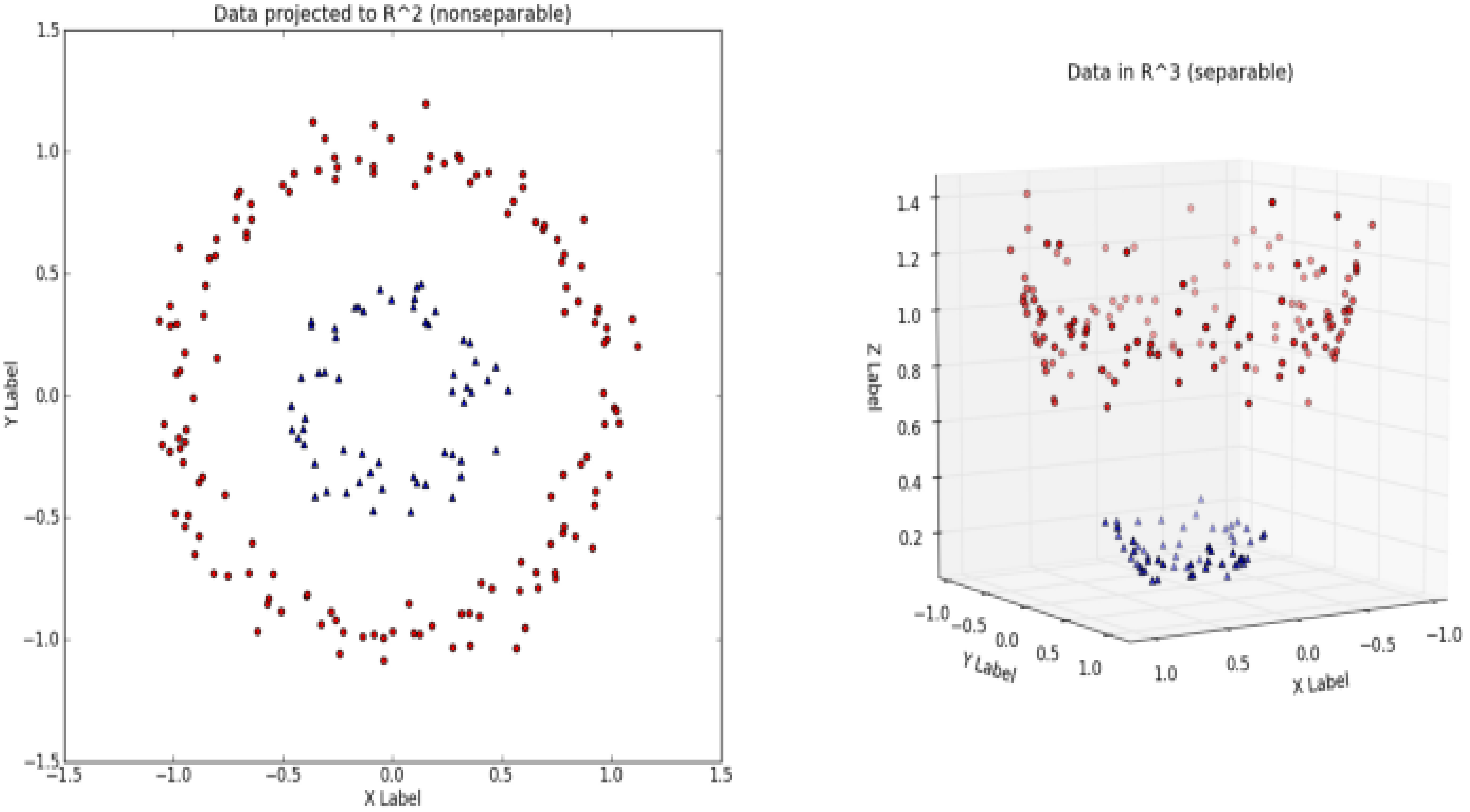


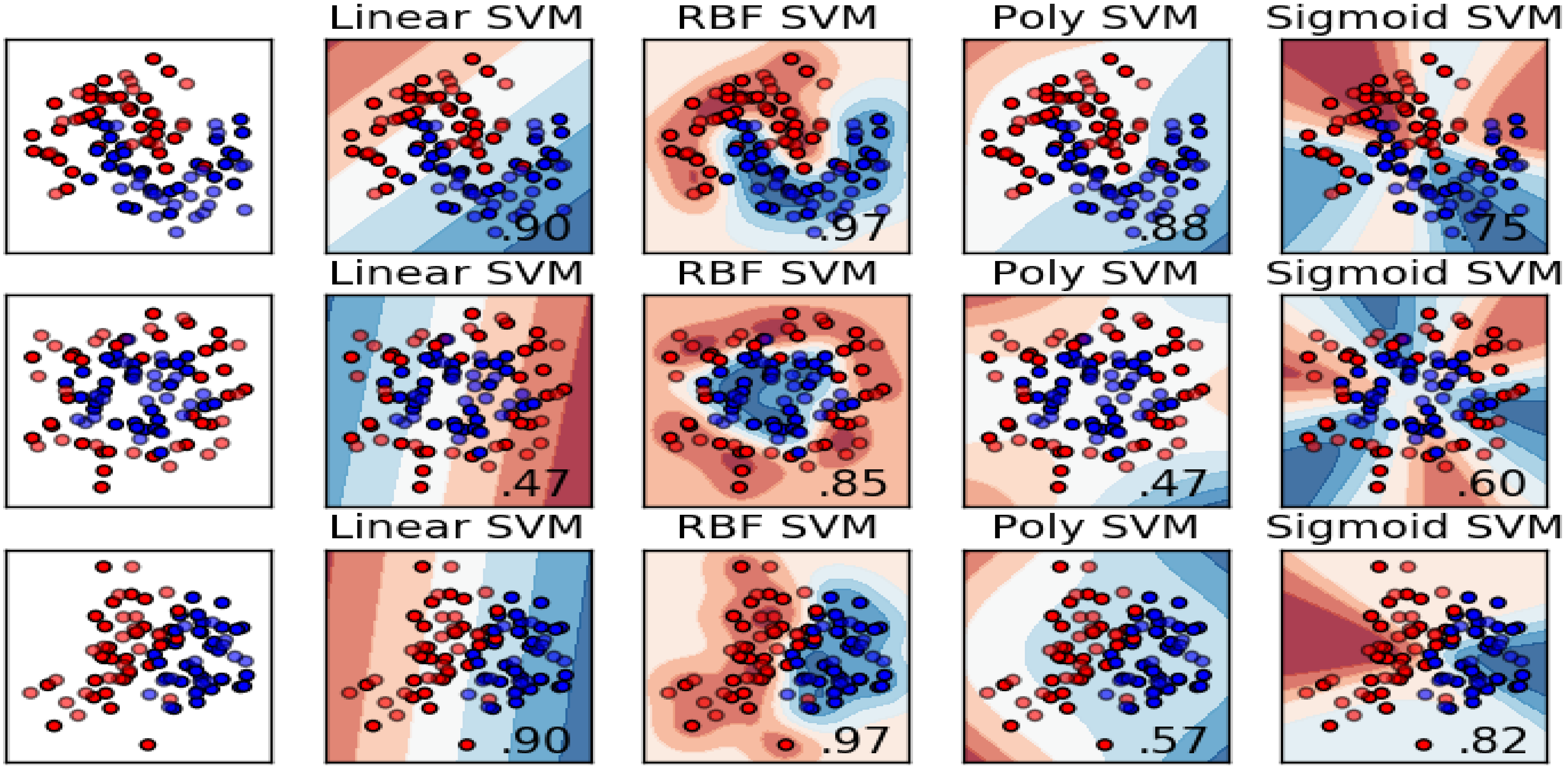
Large C
Focus on classification

C $\boxed{\text{Classification Error}}$ + ~~Margin Error~~

Support vector machines

- Common kernel functions for SVM
  - linear $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 \cdot \mathbf{x}_2$
  - polynomial $k(\mathbf{x}_1, \mathbf{x}_2) = (\gamma\, \mathbf{x}_1 \cdot \mathbf{x}_2 + c)^d$
  - Gaussian or radial basis $k(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\gamma \left\|\mathbf{x}_1 - \mathbf{x}_2\right\|^2\right)$
  - sigmoid $k(\mathbf{x}_1, \mathbf{x}_2) = \tanh(\gamma\, \mathbf{x}_1 \cdot \mathbf{x}_2 + c)$


Data projected to R^2 (nonseparable)
Y Label
X Label
Data in R^3 (separable)
Z Label
Y Label
X Label


Linear SVM
RBF SVM
Poly SVM
Sigmoid SVM
.90
.97
.88
.75
Linear SVM
RBF SVM
Poly SVM
Sigmoid SVM
.47
.85
.47
.60
Linear SVM
RBF SVM
Poly SVM
Sigmoid SVM
.90
.97
.57
.82

HOW TO CHOOSE A KERNEL ?

Let : n= number of features, m = number of training samples

1. n > m : use logistic regression or SVM with no kernel (or linear kernel)

2. n < m : use SVM with gaussian kernel

note: most of the time 'linear' & 'rbf' kernels do well.