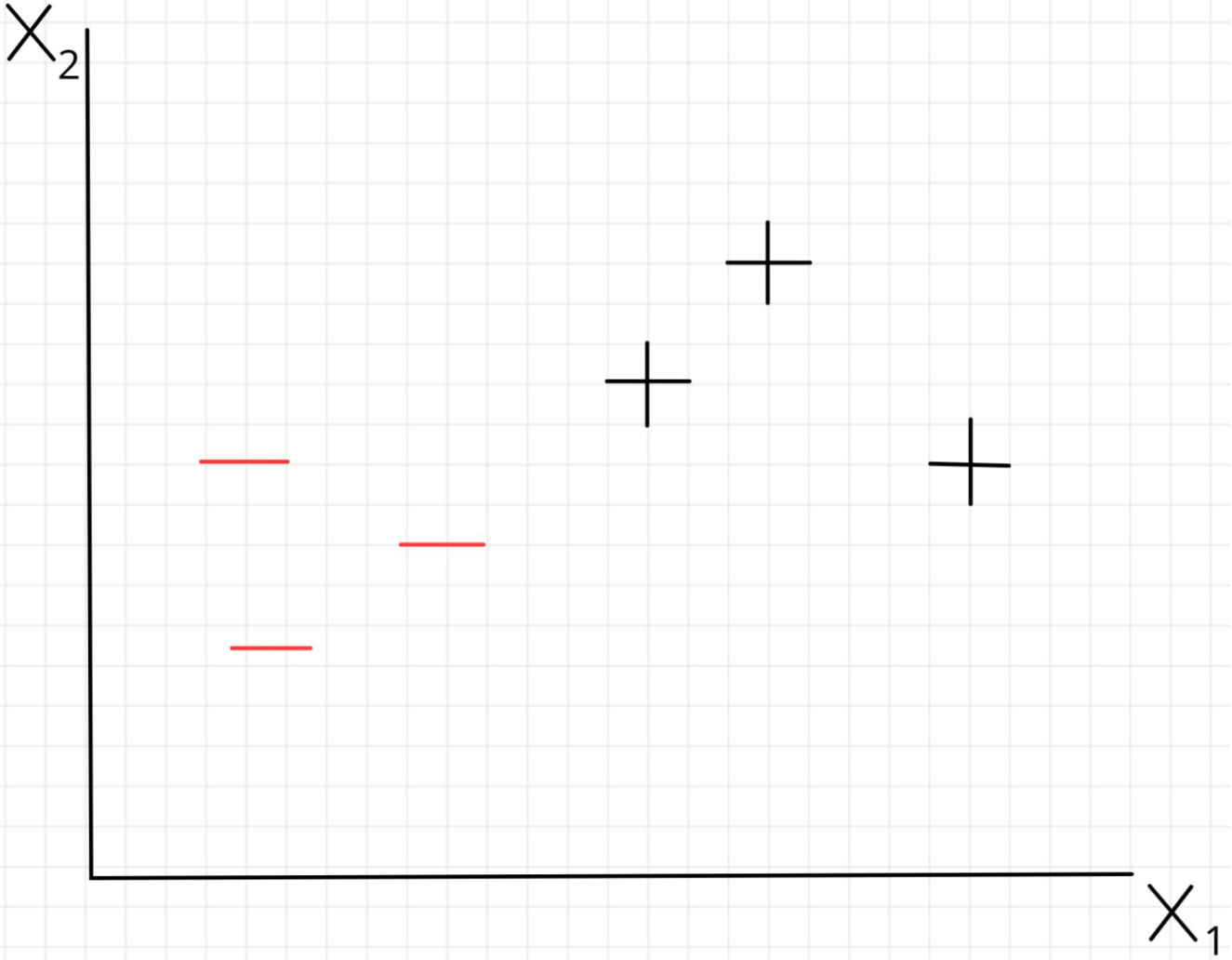
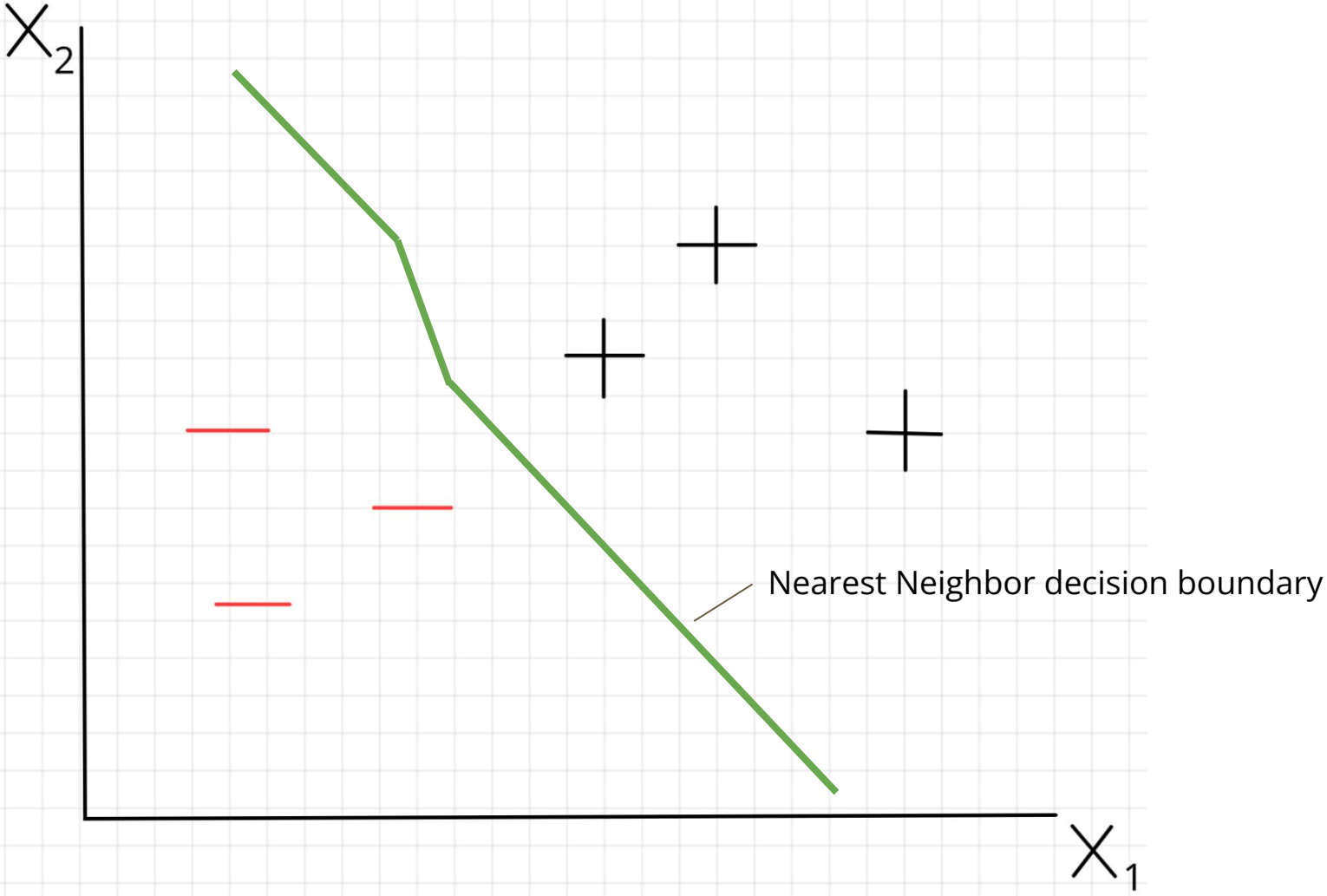
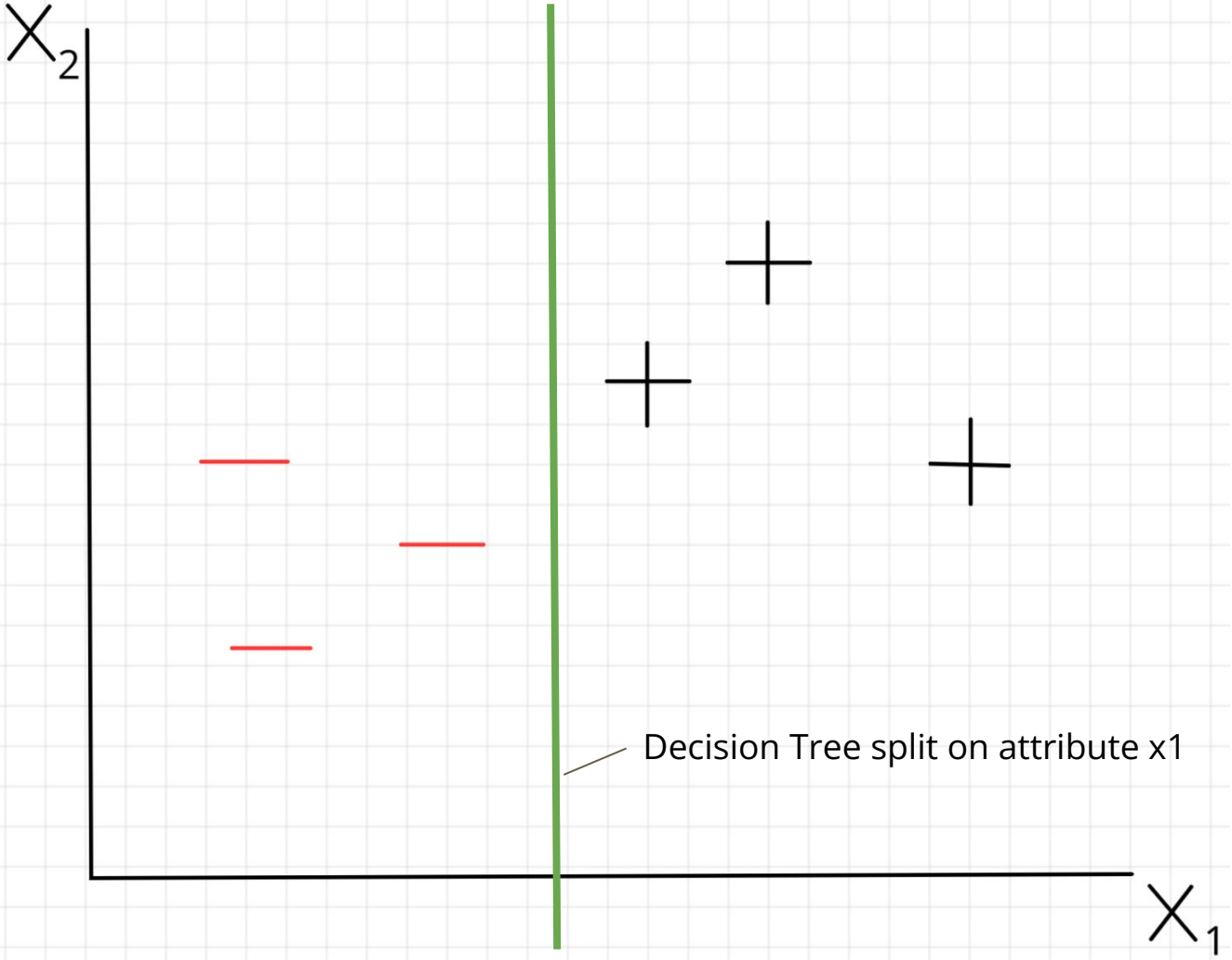
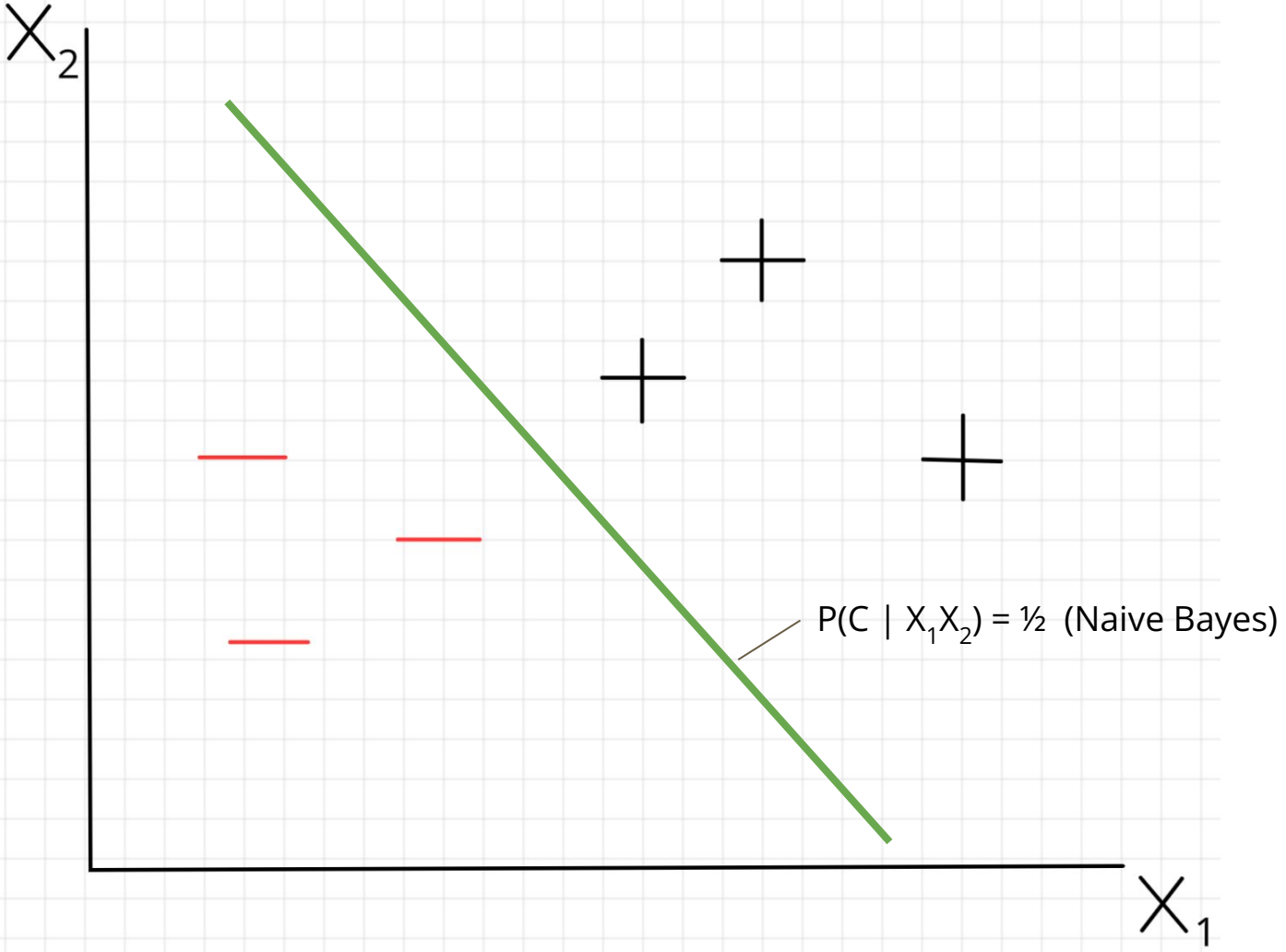
Support Vector Machines

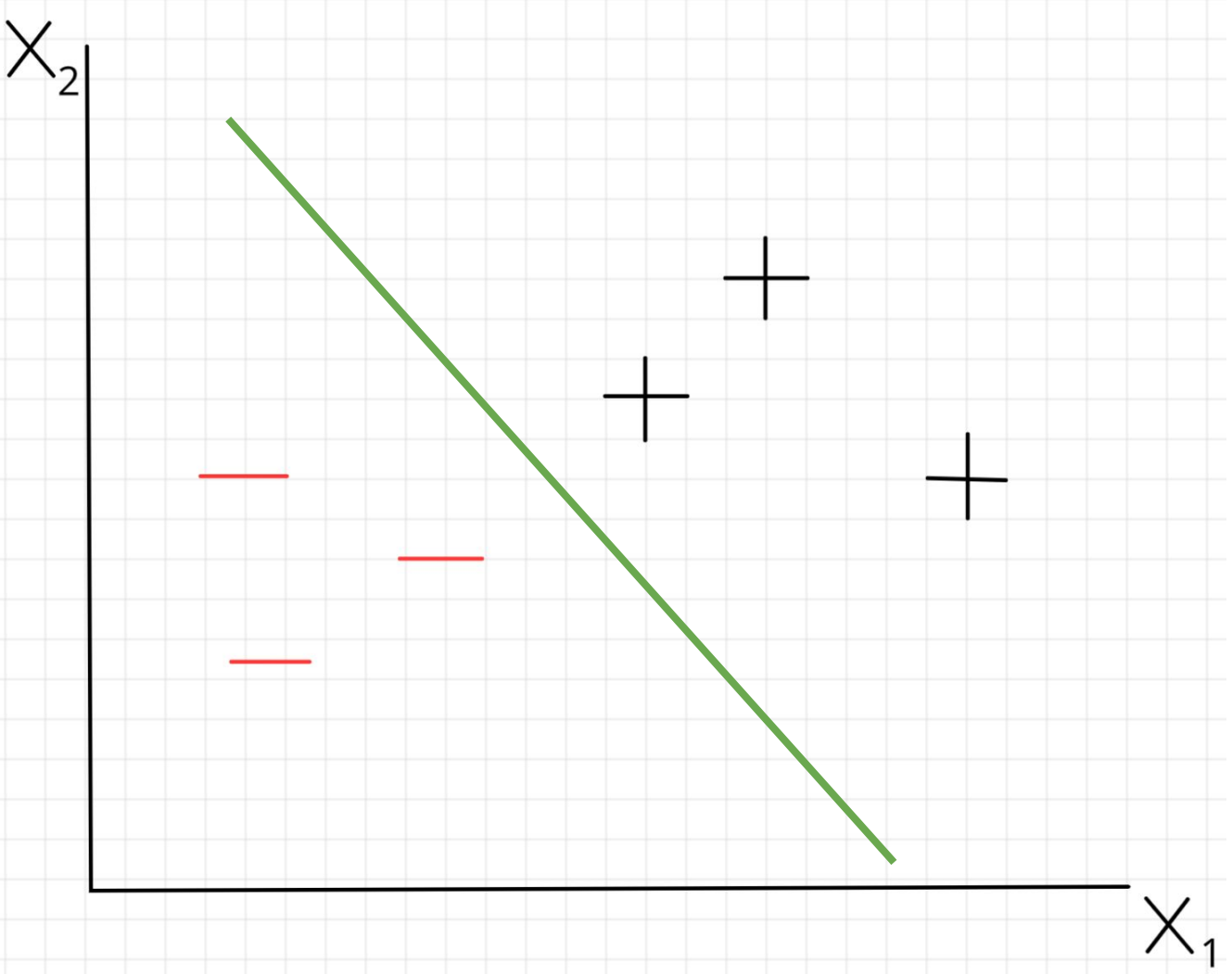
Boston University CS 506 - Lance Galletti

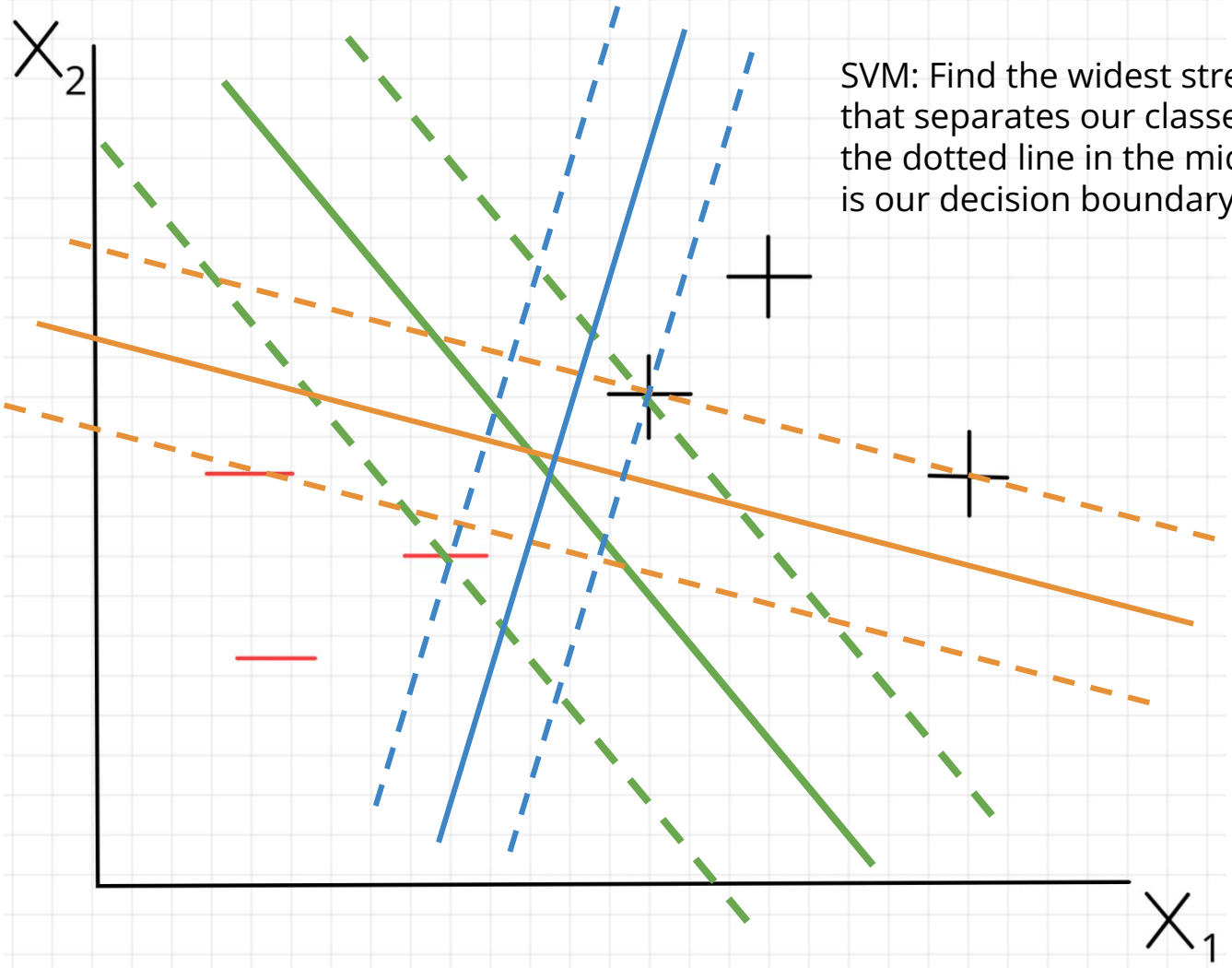

X2
X1


X_2
Nearest Neighbor decision boundary
X_1

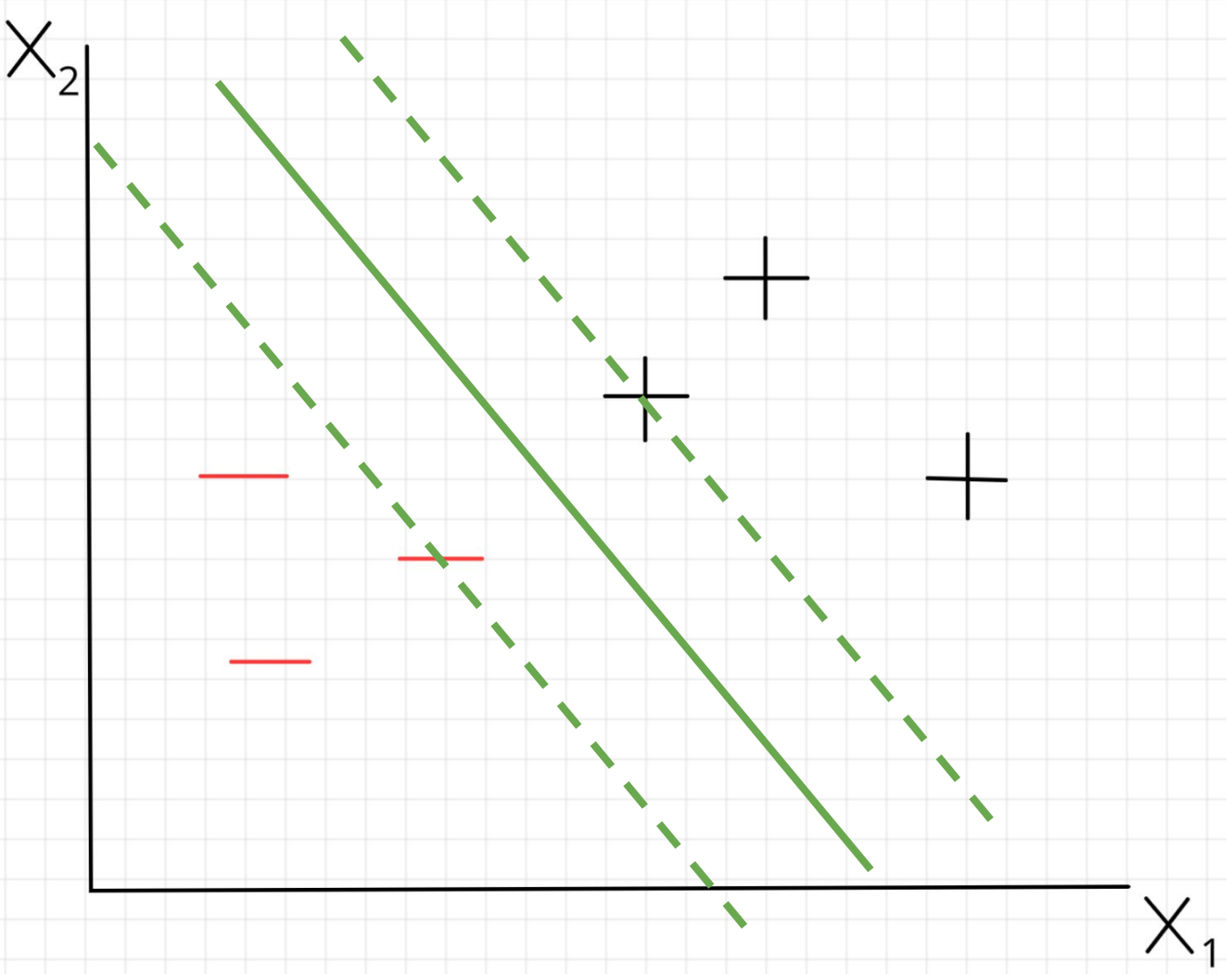

X_2
Decision Tree split on attribute x1
X_1

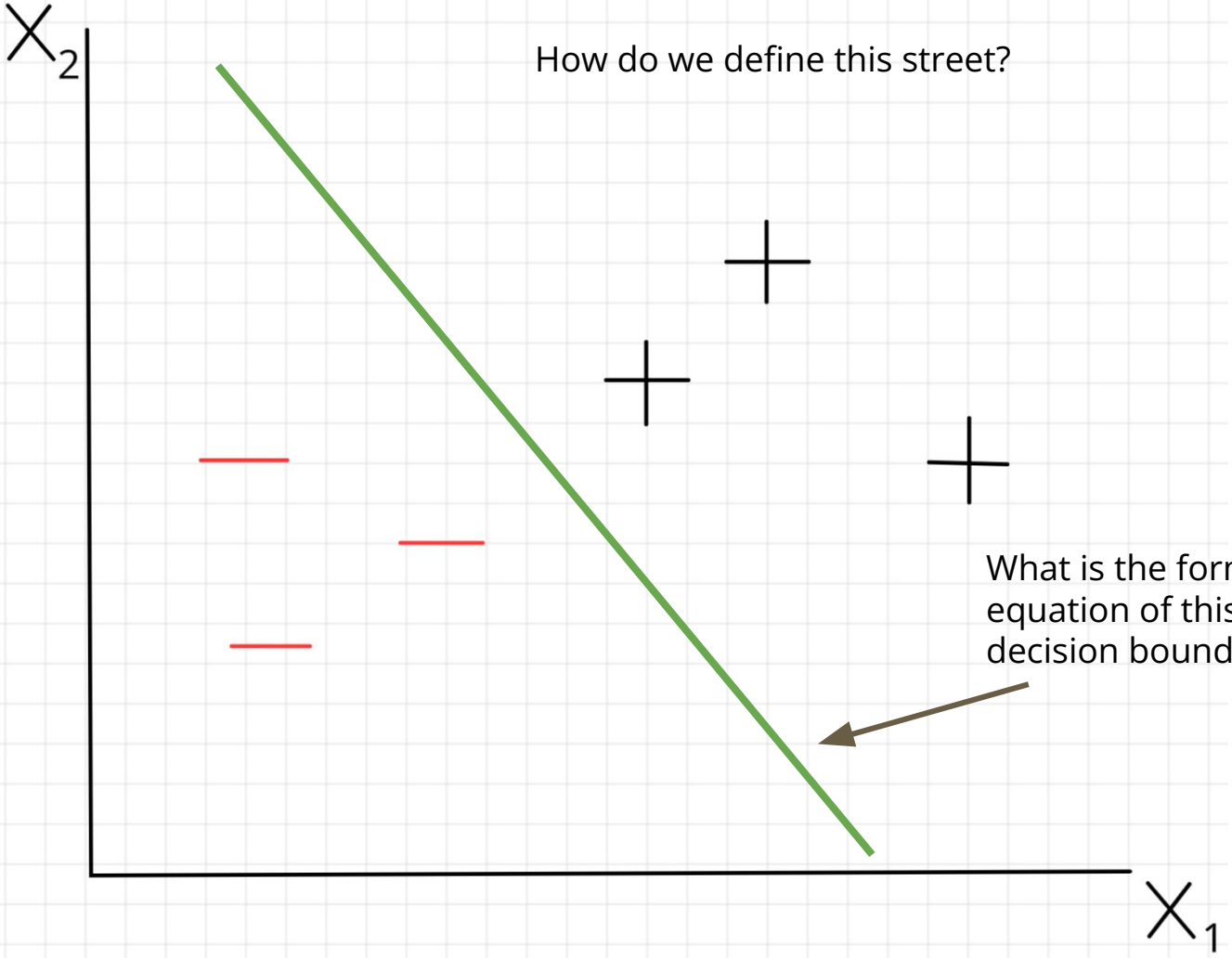

X2
X1
P(C | X1X2) = ½ (Naive Bayes)

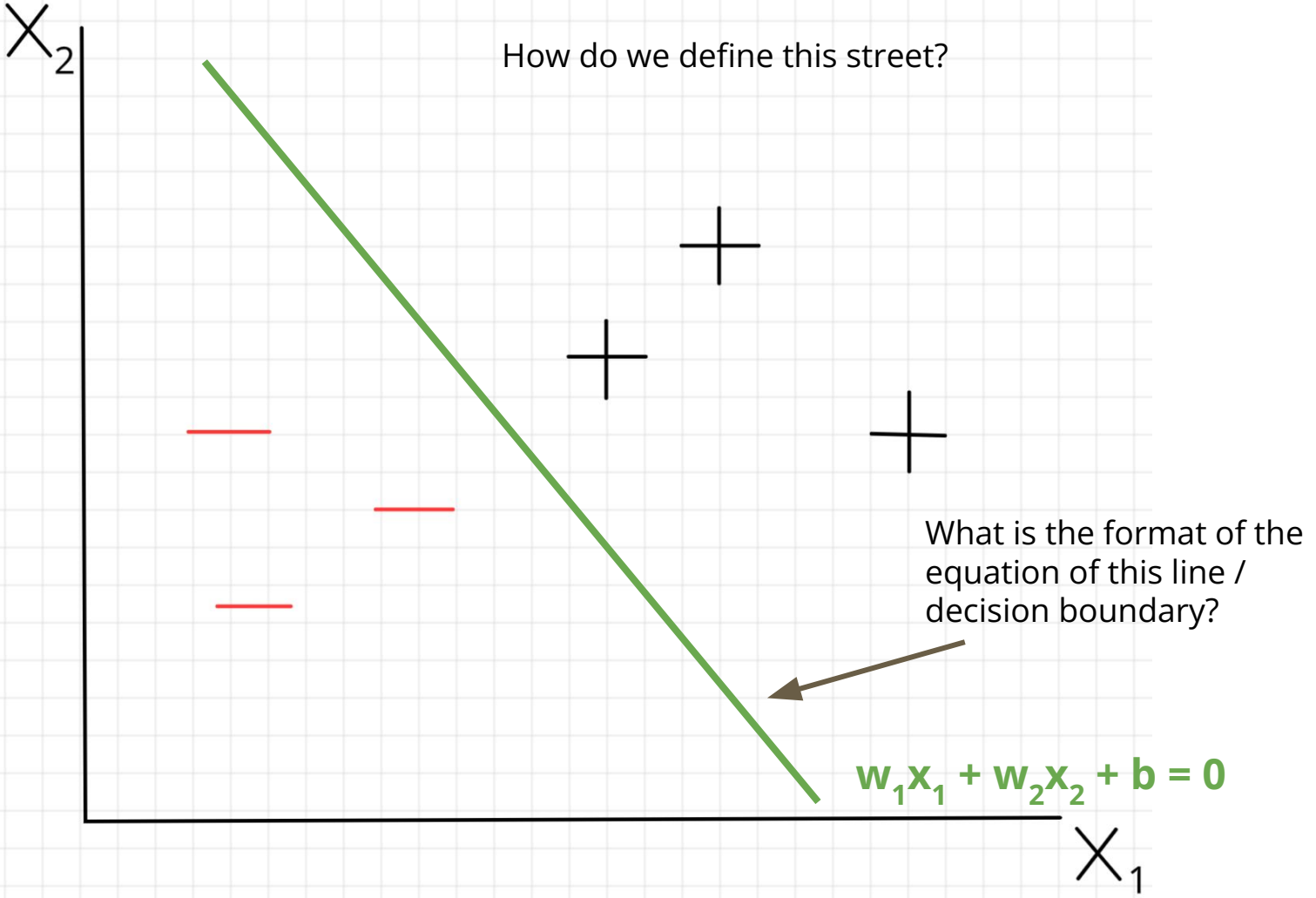

X2
X1

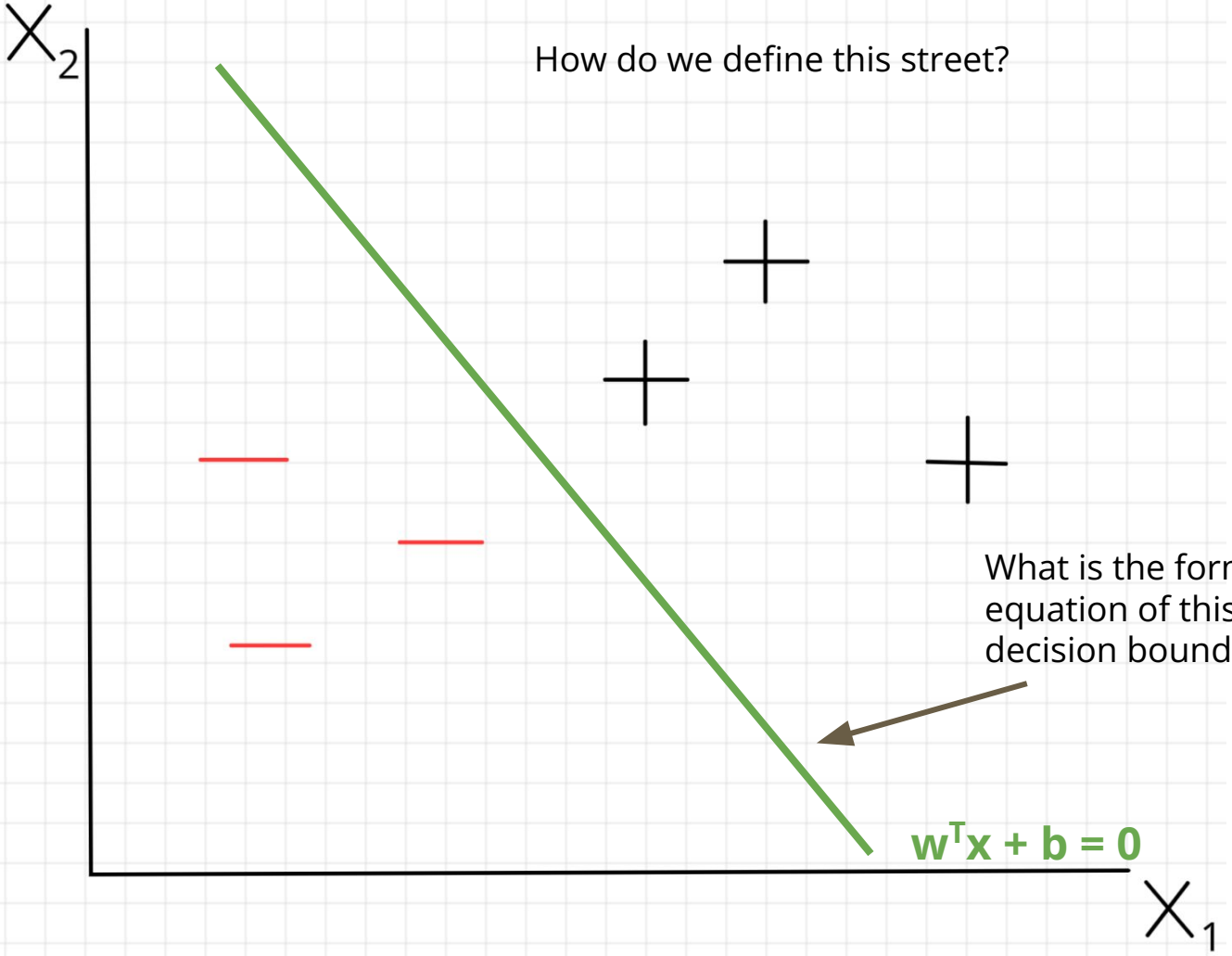


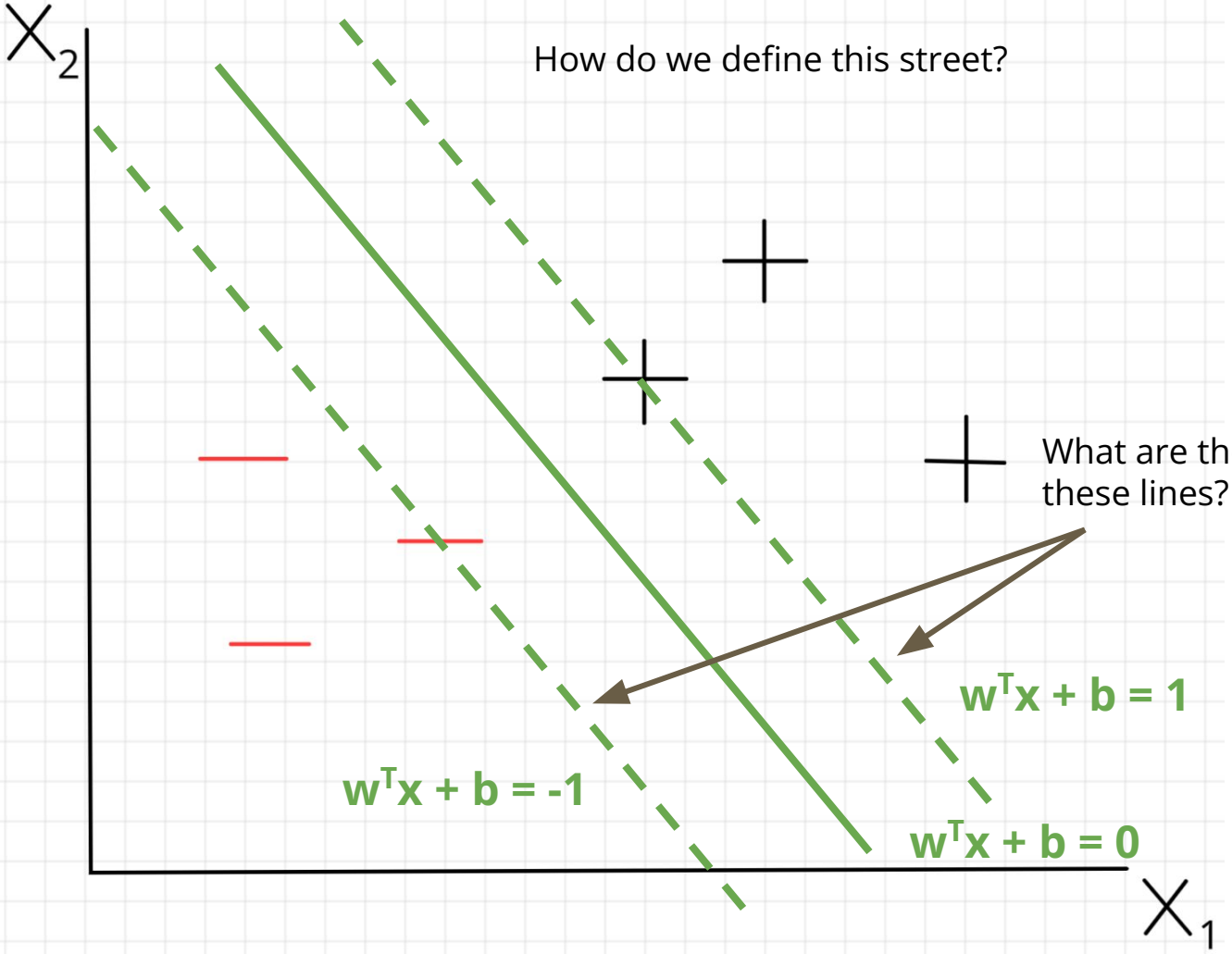
SVM: Find the widest street that separates our classes - the dotted line in the middle is our decision boundary

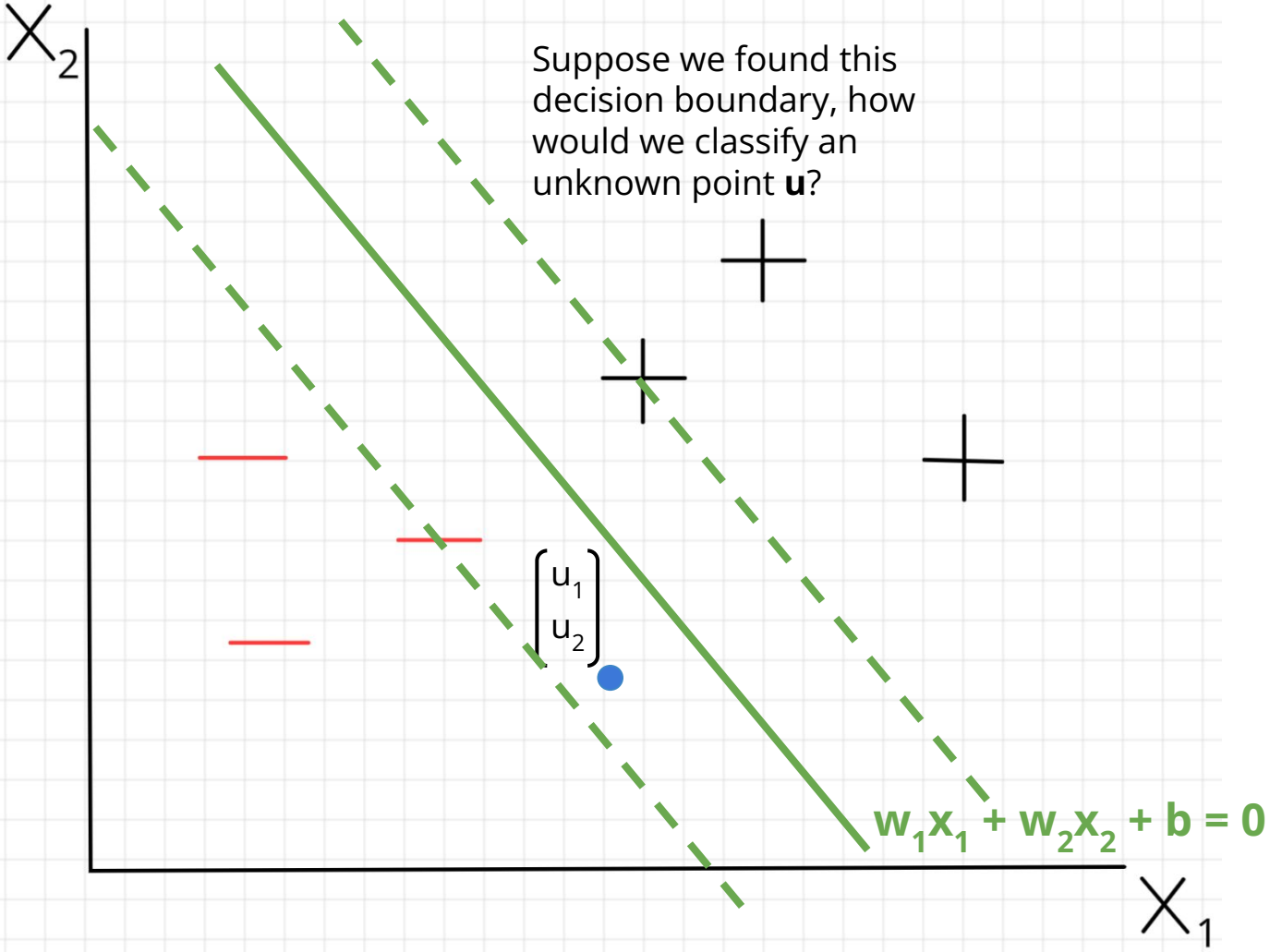

X2
X1

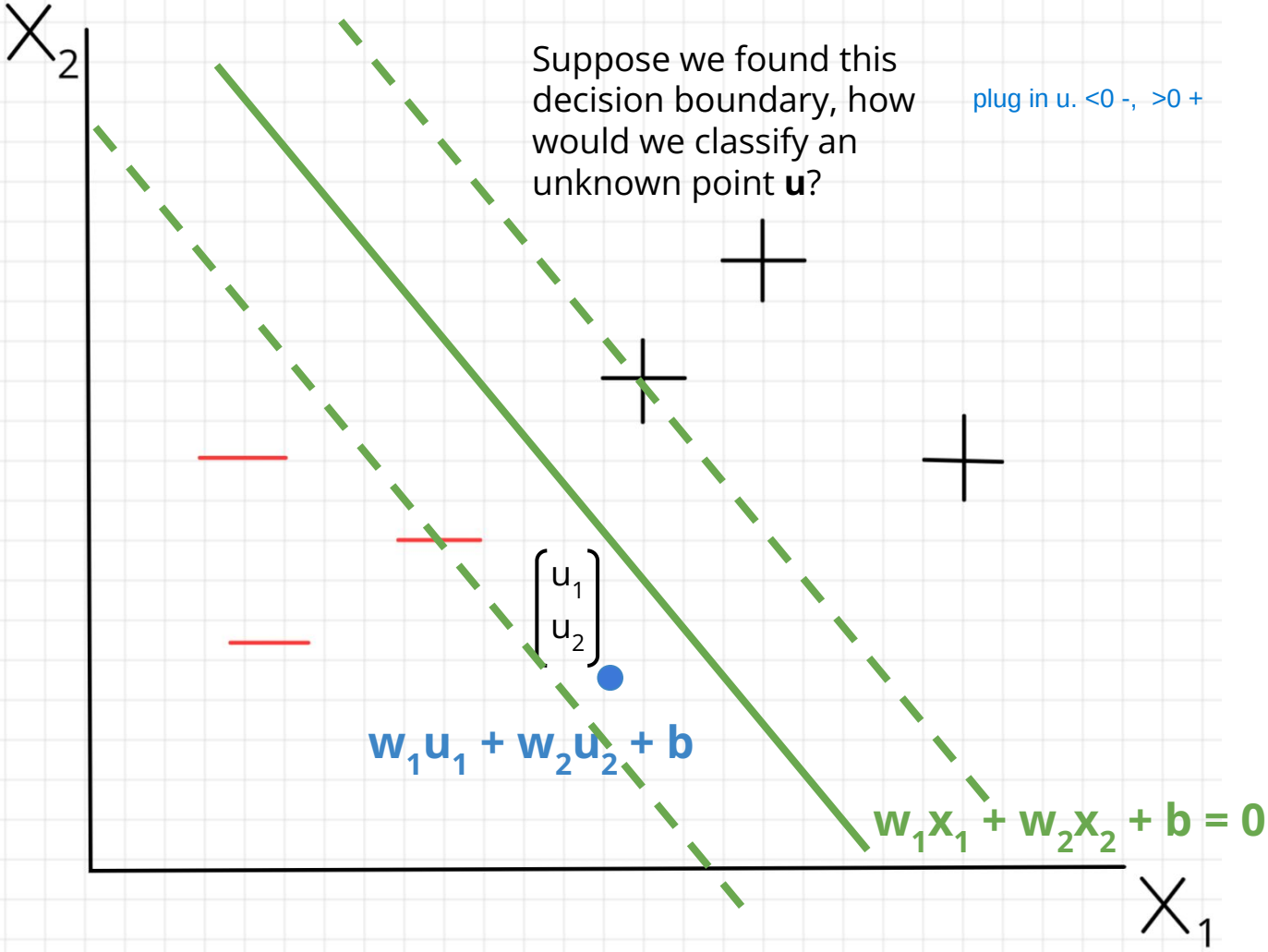

How do we define this street?
What is the format of the equation of this line / decision boundary?
X2
X1

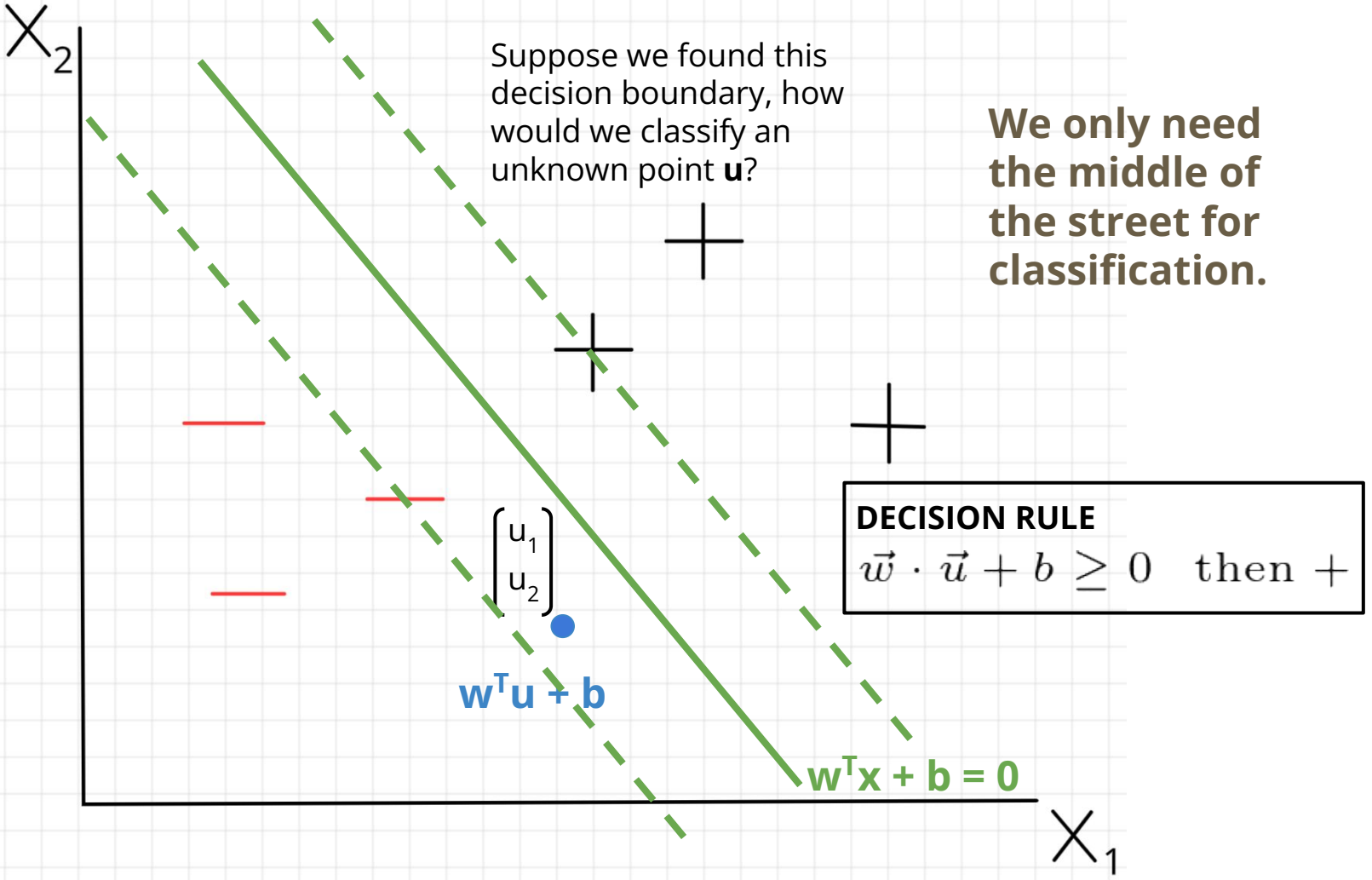

How do we define this street?
What is the format of the equation of this line / decision boundary?
$w_1x_1 + w_2x_2 + b = 0$
X_2
X_1

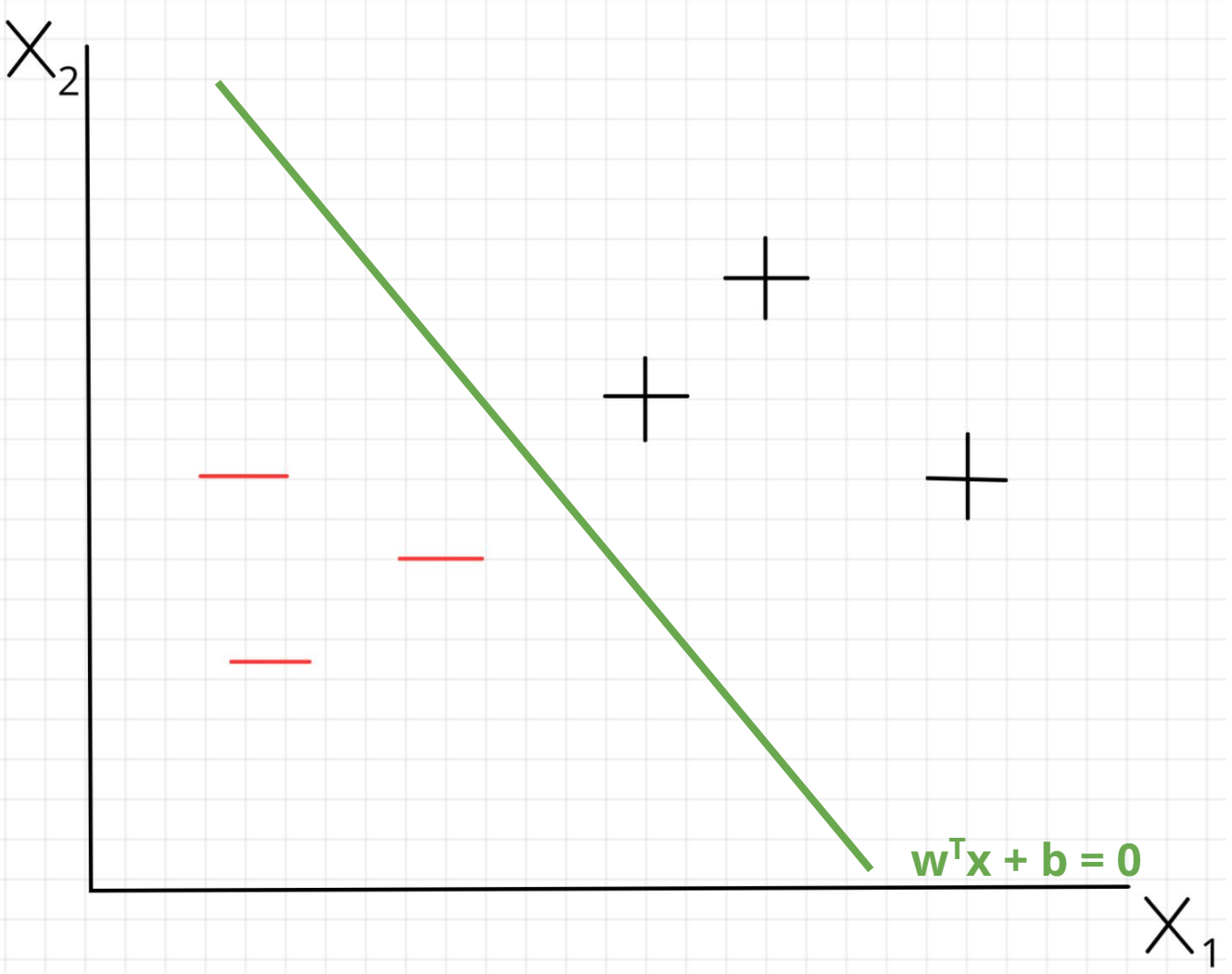

How do we define this street?
What is the format of the equation of this line / decision boundary?
$w^T x + b = 0$
X_2
X_1

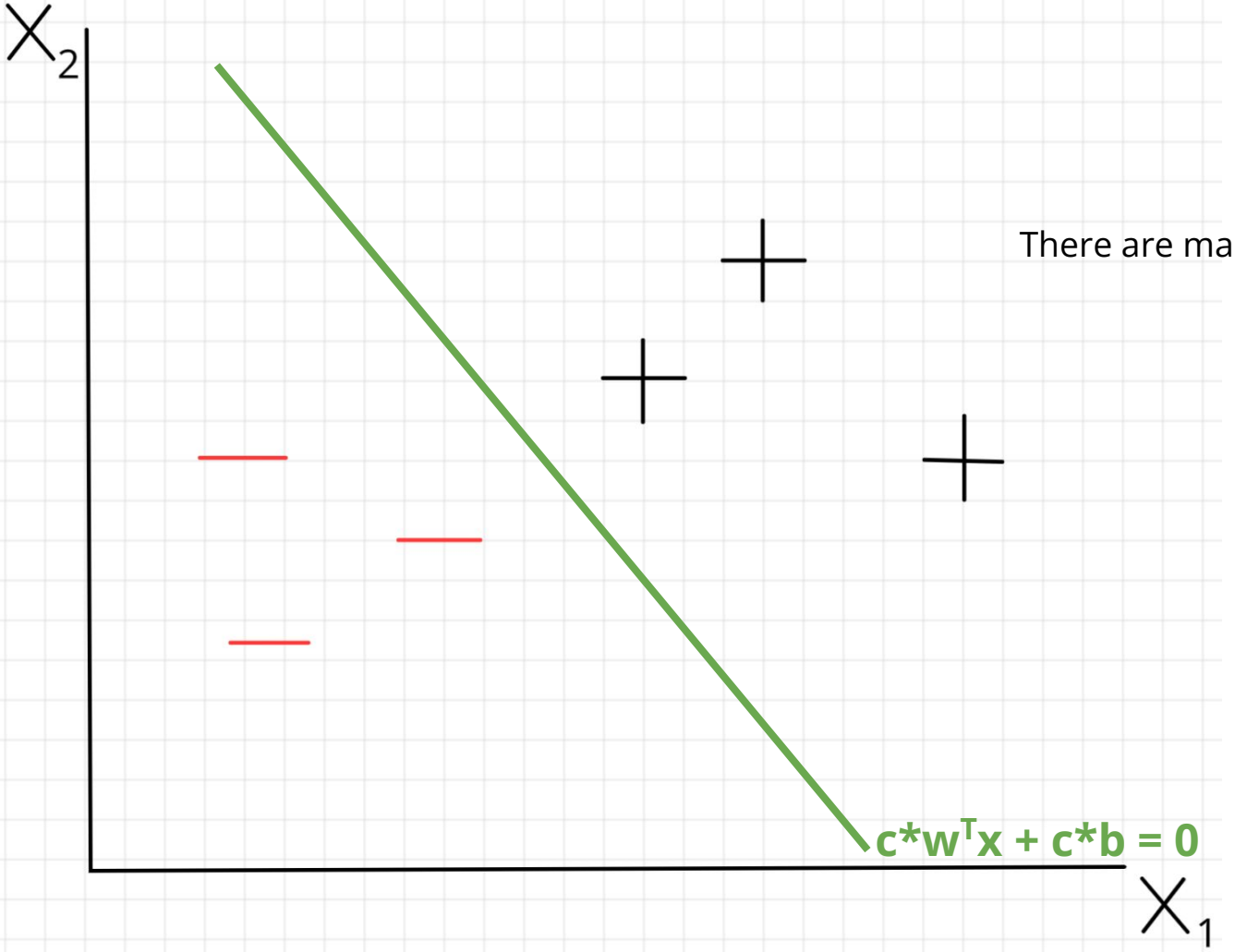

How do we define this street?
What are the equation of these lines?
$w^T x + b = 1$
$w^T x + b = -1$
$w^T x + b = 0$
X_2
X_1


Suppose we found this decision boundary, how would we classify an unknown point **u**?
X_2
X_1
$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$
$w_1x_1 + w_2x_2 + b = 0$

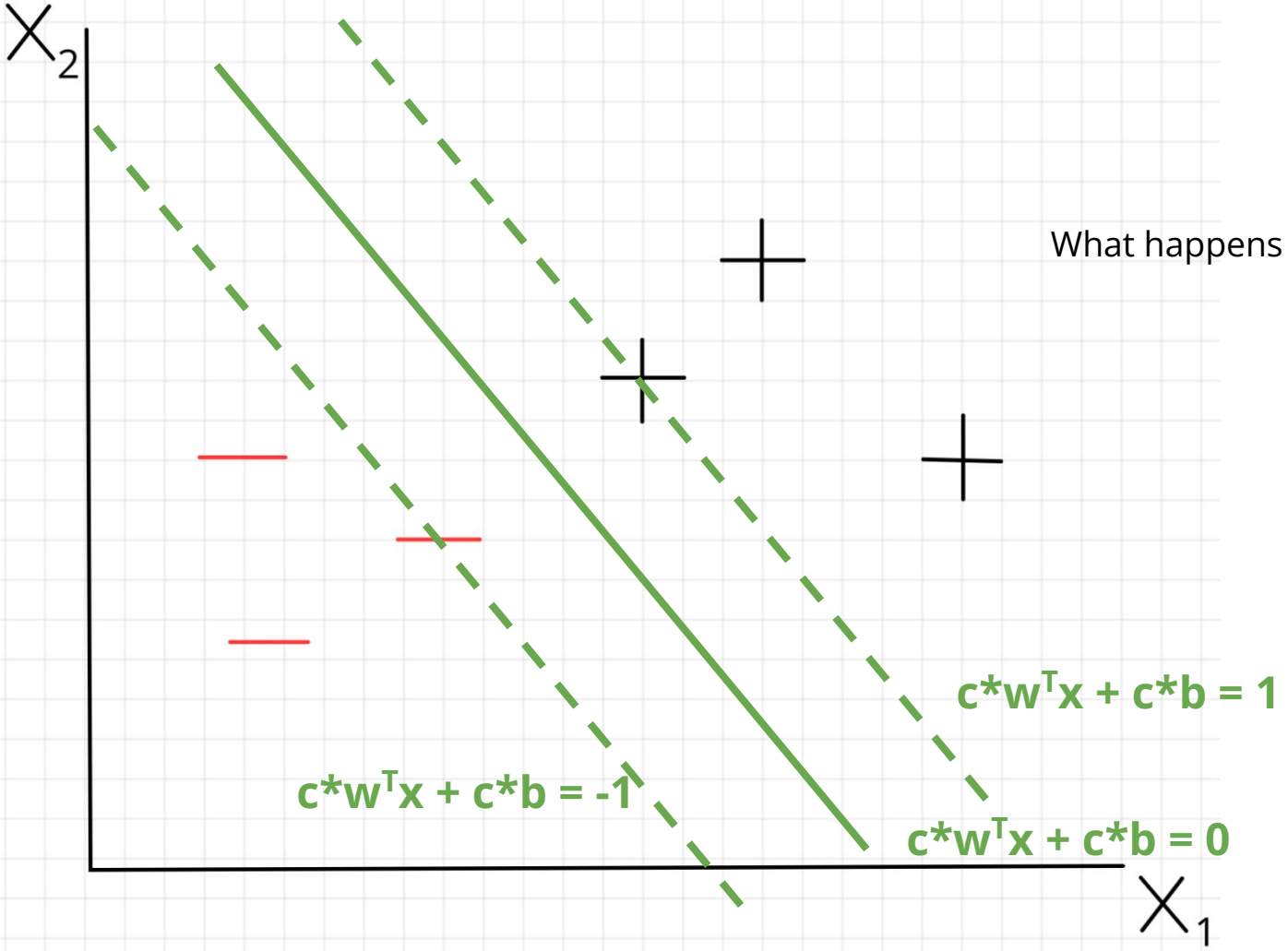

Suppose we found this decision boundary, how would we classify an unknown point **u**?
plug in u. <0 -, >0 +
X_2
X_1
$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$
$w_1u_1 + w_2u_2 + b$
$w_1x_1 + w_2x_2 + b = 0$


X2
X1
Suppose we found this decision boundary, how would we classify an unknown point u?
We only need the middle of the street for classification.
DECISION RULE
$\vec{w} \cdot \vec{u} + b \geq 0$ then +
u1
u2
wTu + b
wTx + b = 0

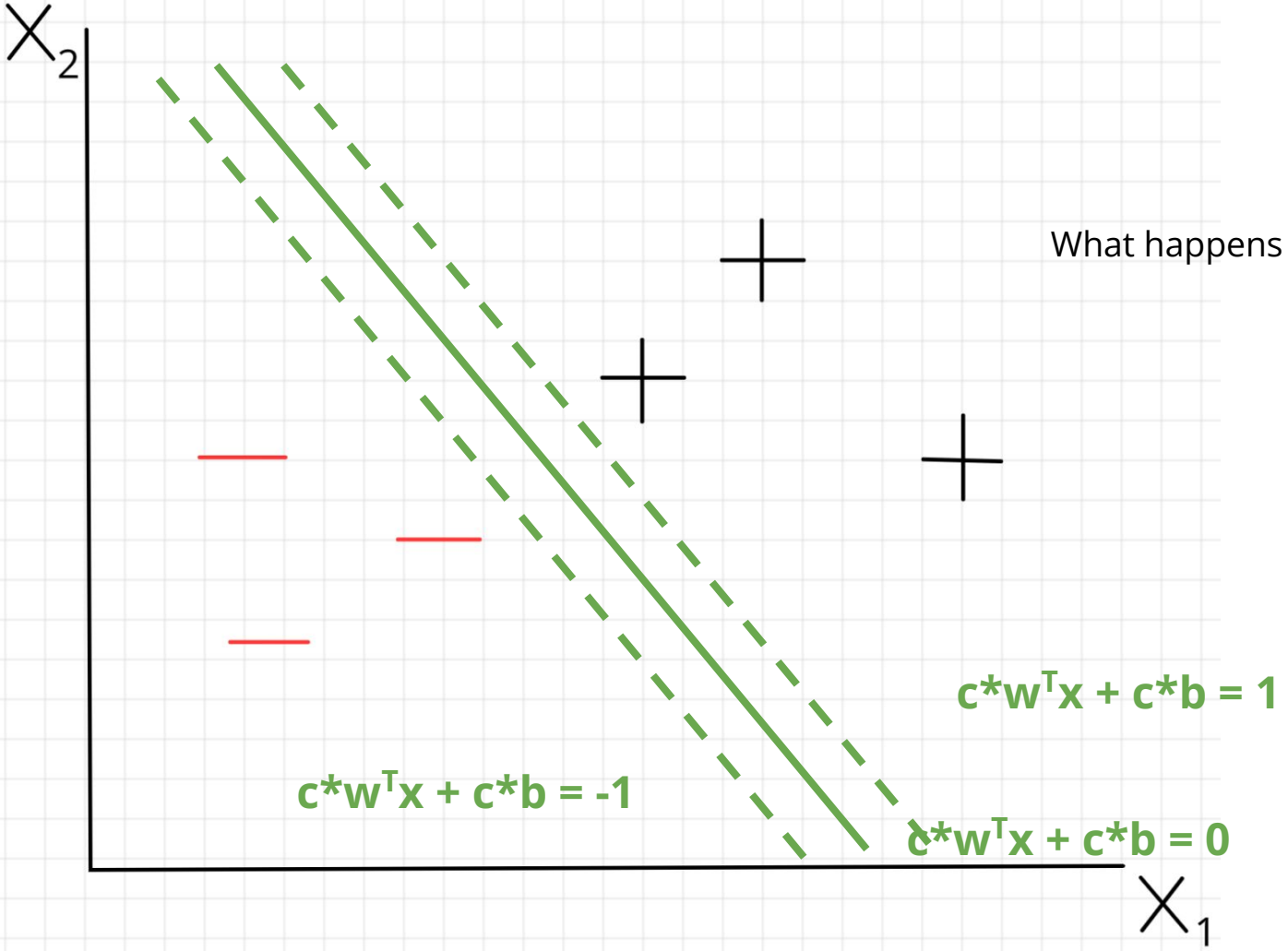

X_2
$w^T x + b = 0$
X_1



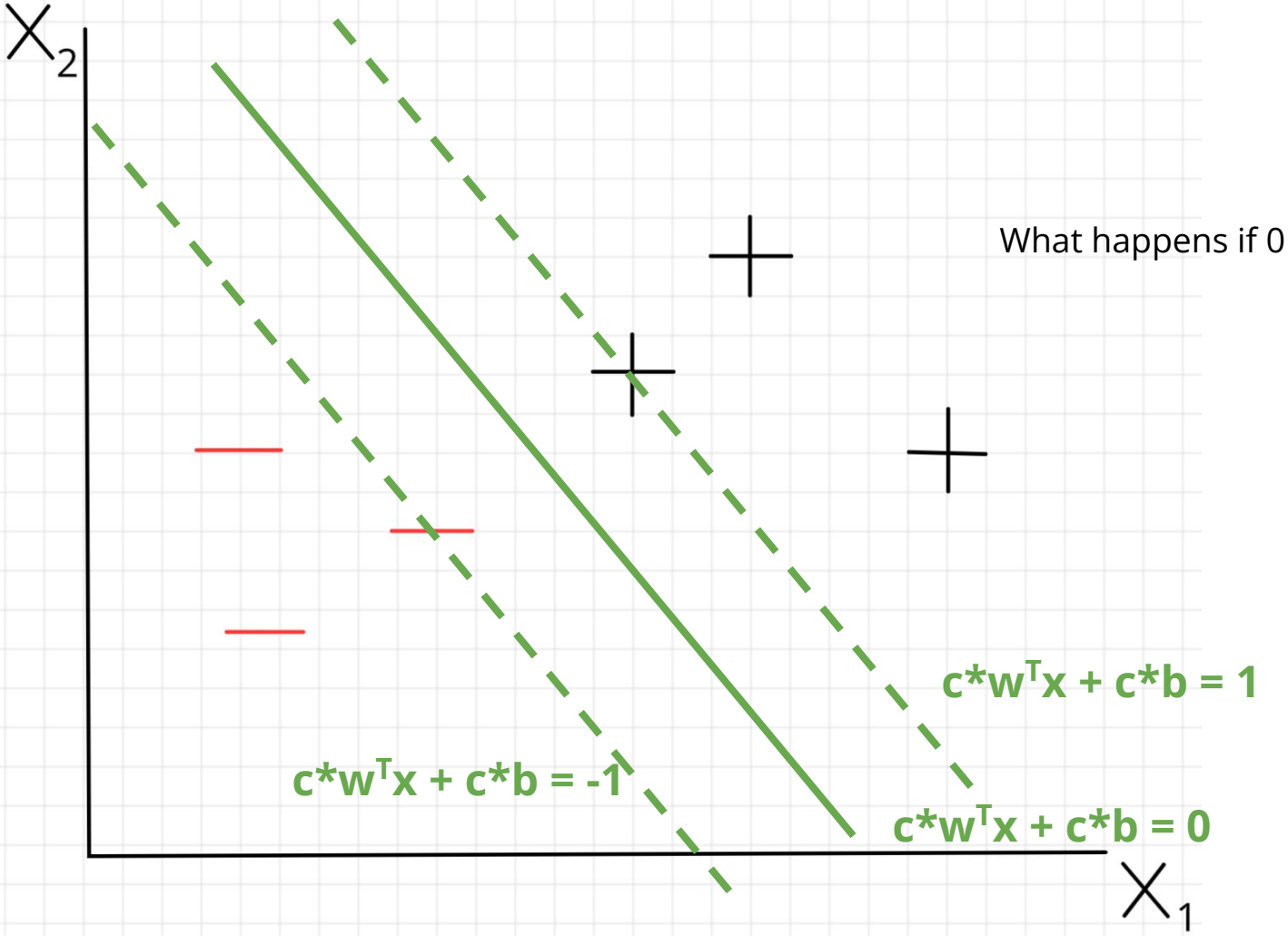
There are many **w**'s and **b**'s



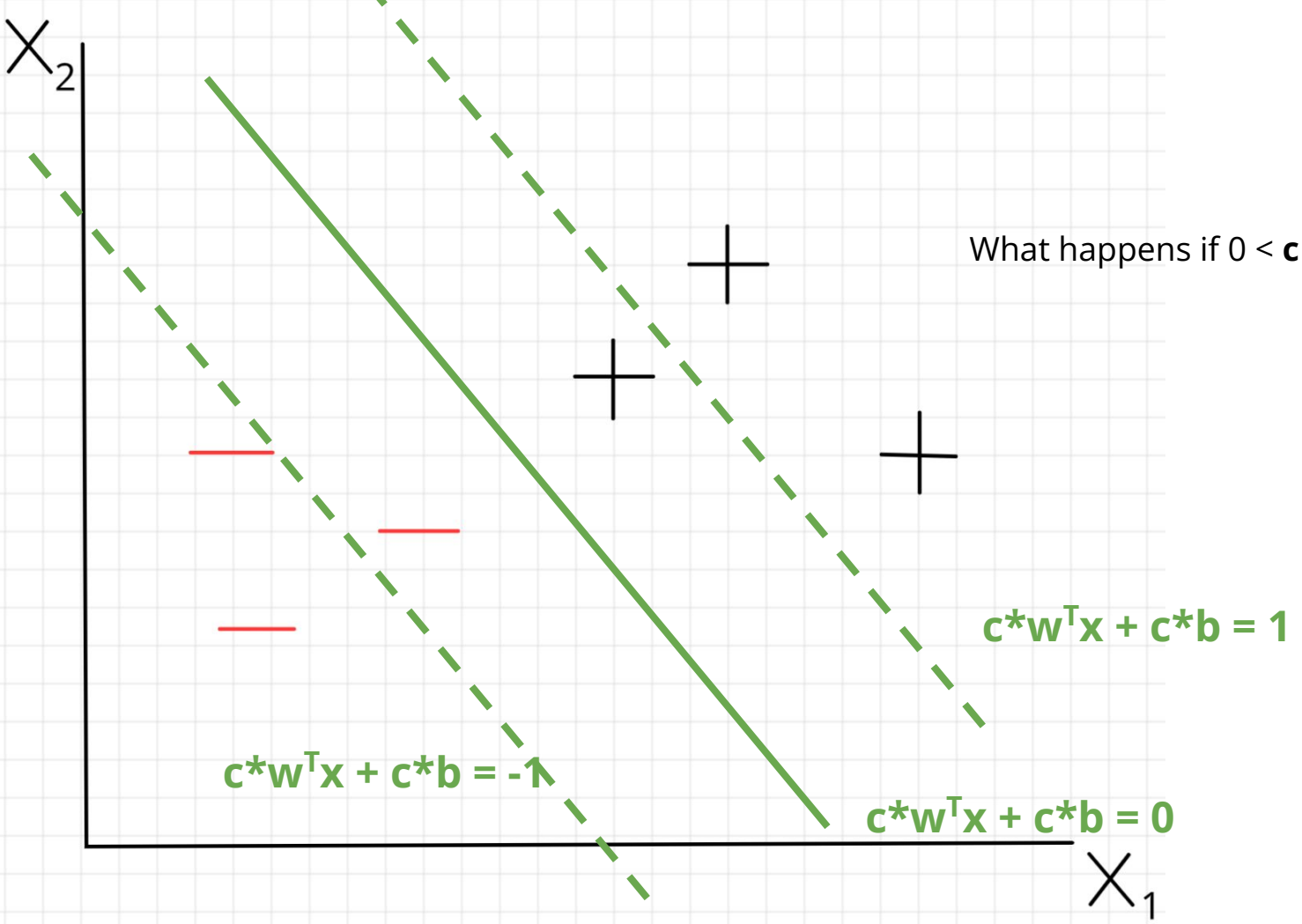
What happens if **c** > 1?



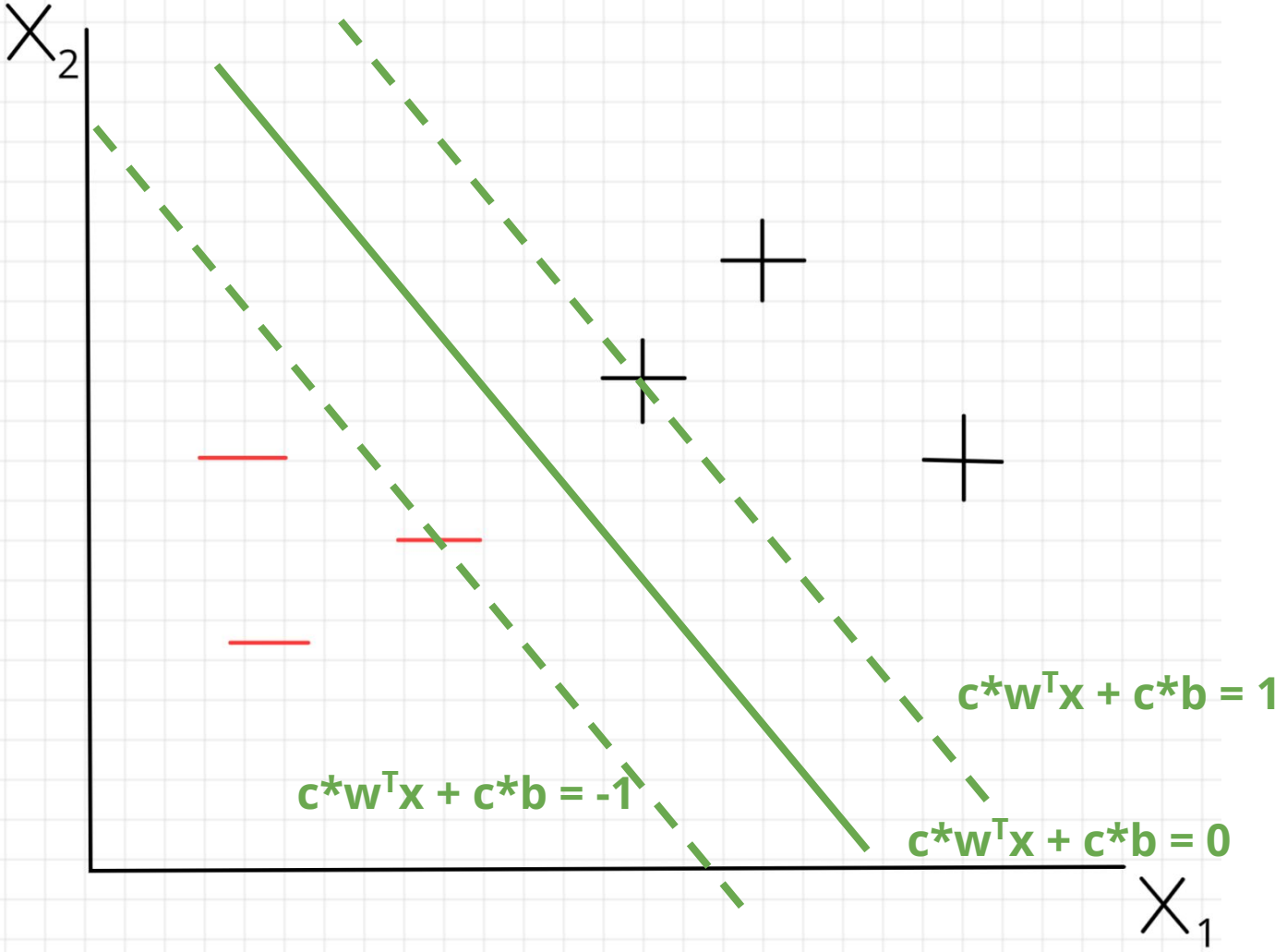
What happens if **c** > 1?

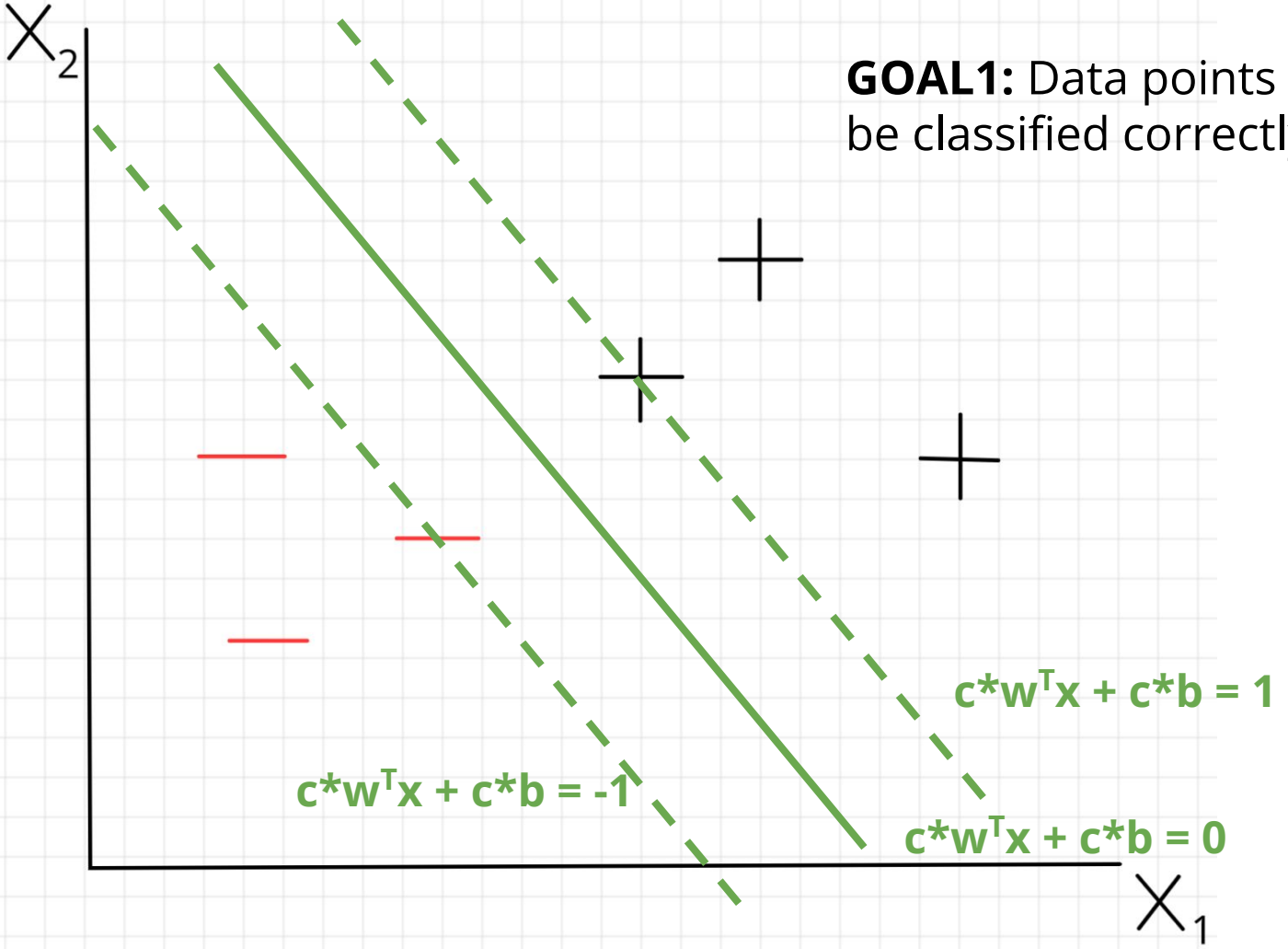


What happens if 0 < **c** < 1?

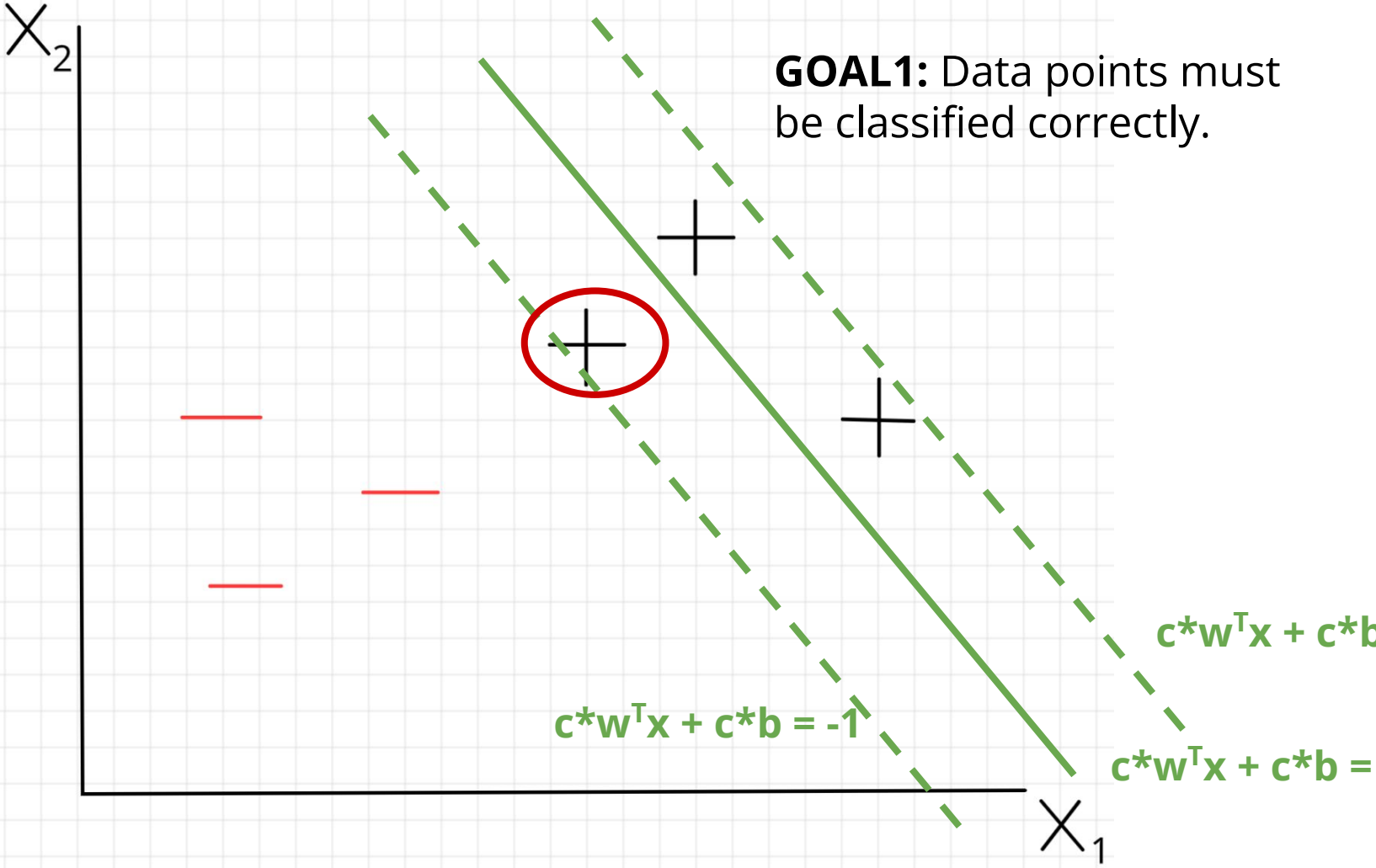


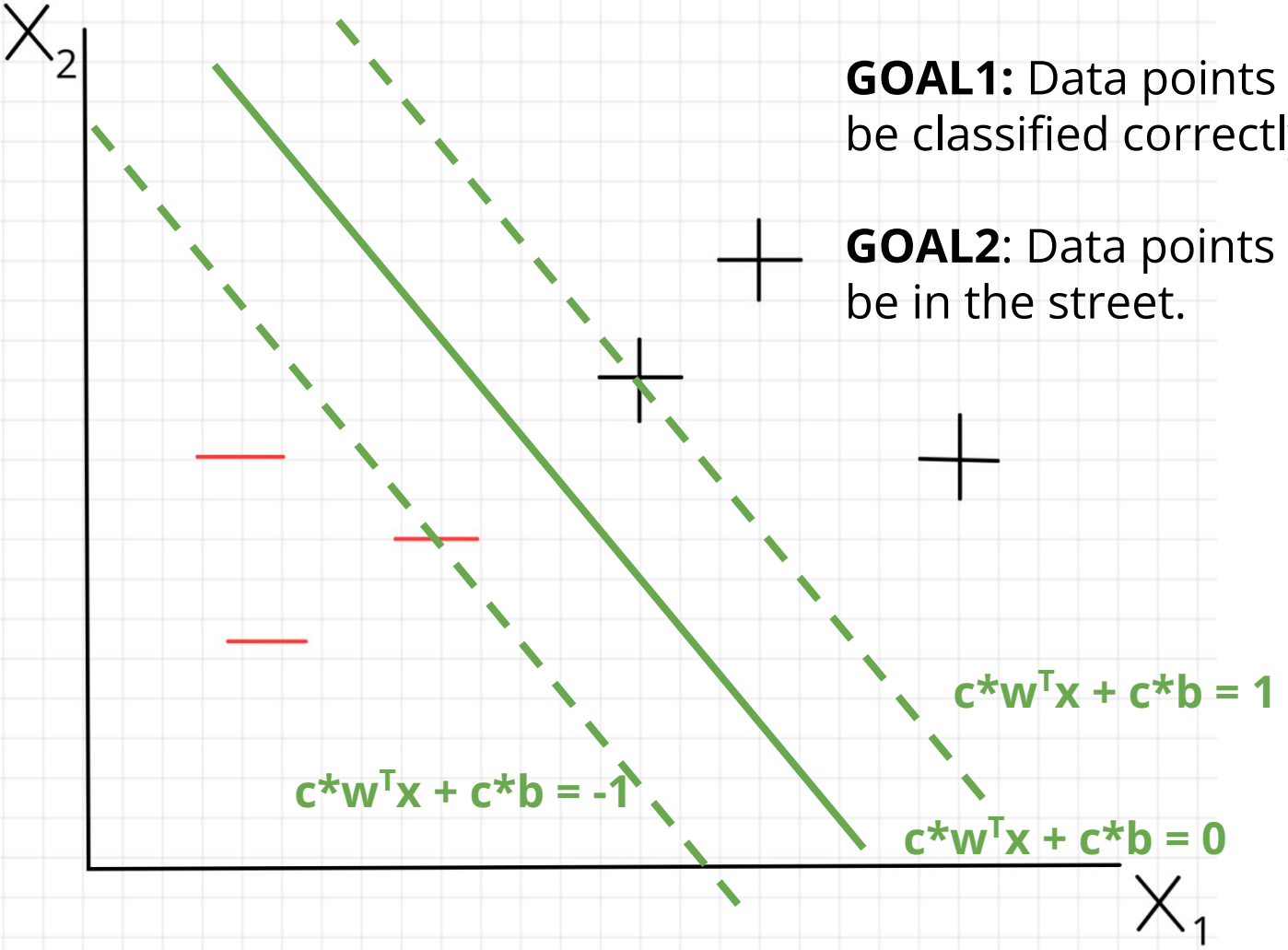
What happens if 0 < **c** < 1?


X2
X1
c*wTx + c*b = 1
c*wTx + c*b = -1
c*wTx + c*b = 0



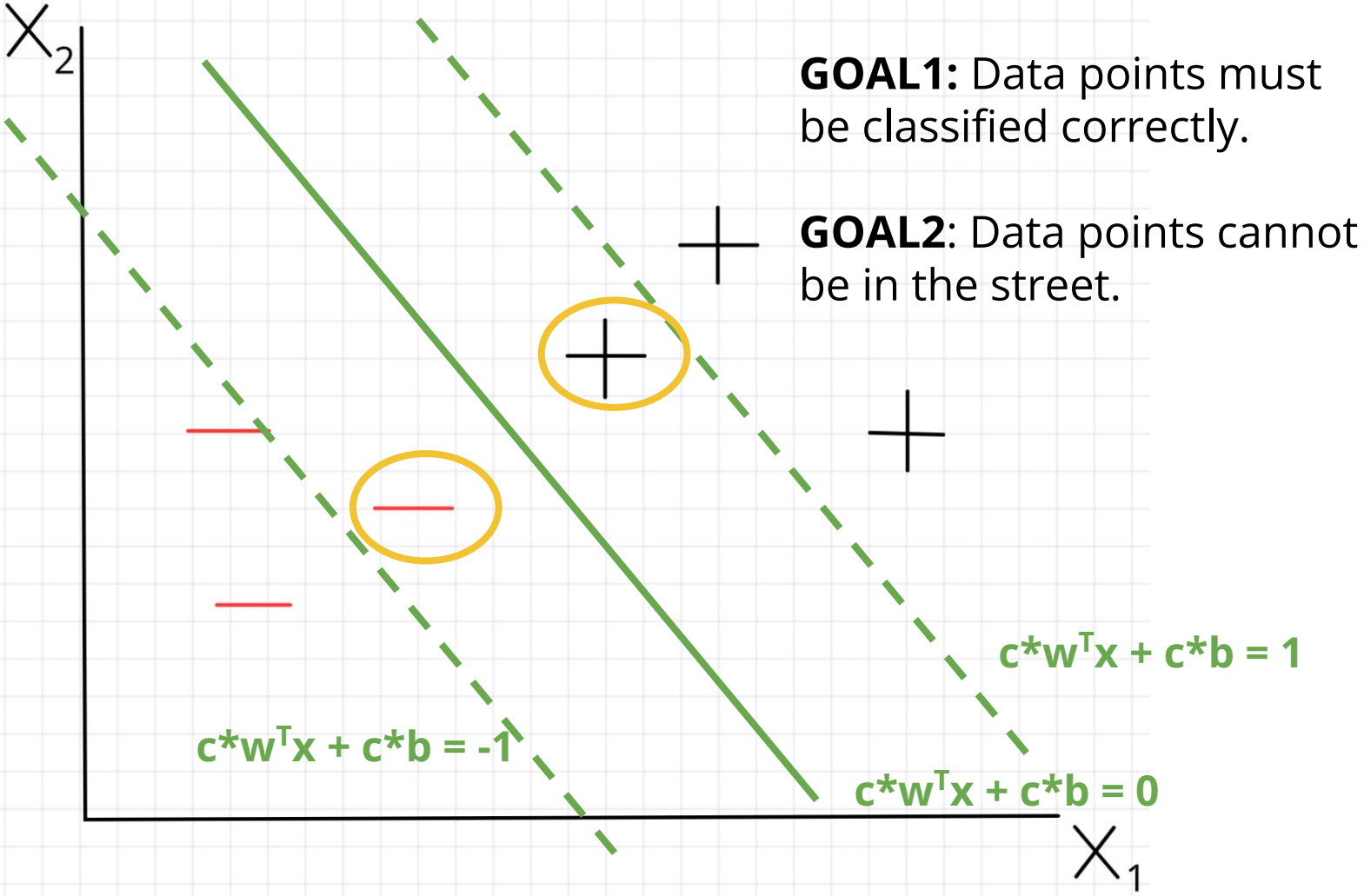
GOAL1: Data points must be classified correctly.


X2
X1
GOAL1: Data points must be classified correctly.
c*wTx + c*b
c*wTx + c*b = -1
c*wTx + c*b =



GOAL1: Data points must be classified correctly.

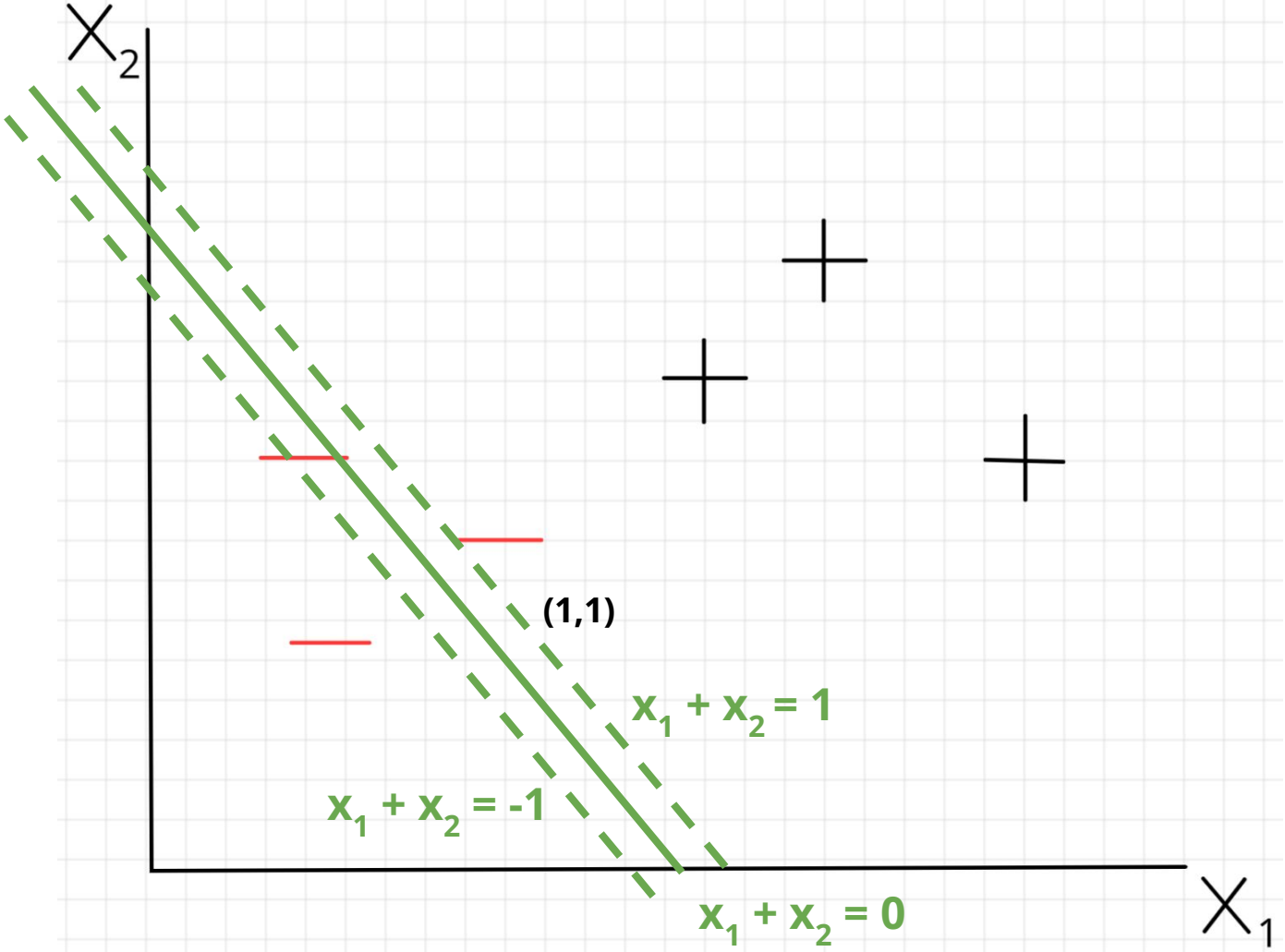
GOAL2: Data points cannot be in the street.

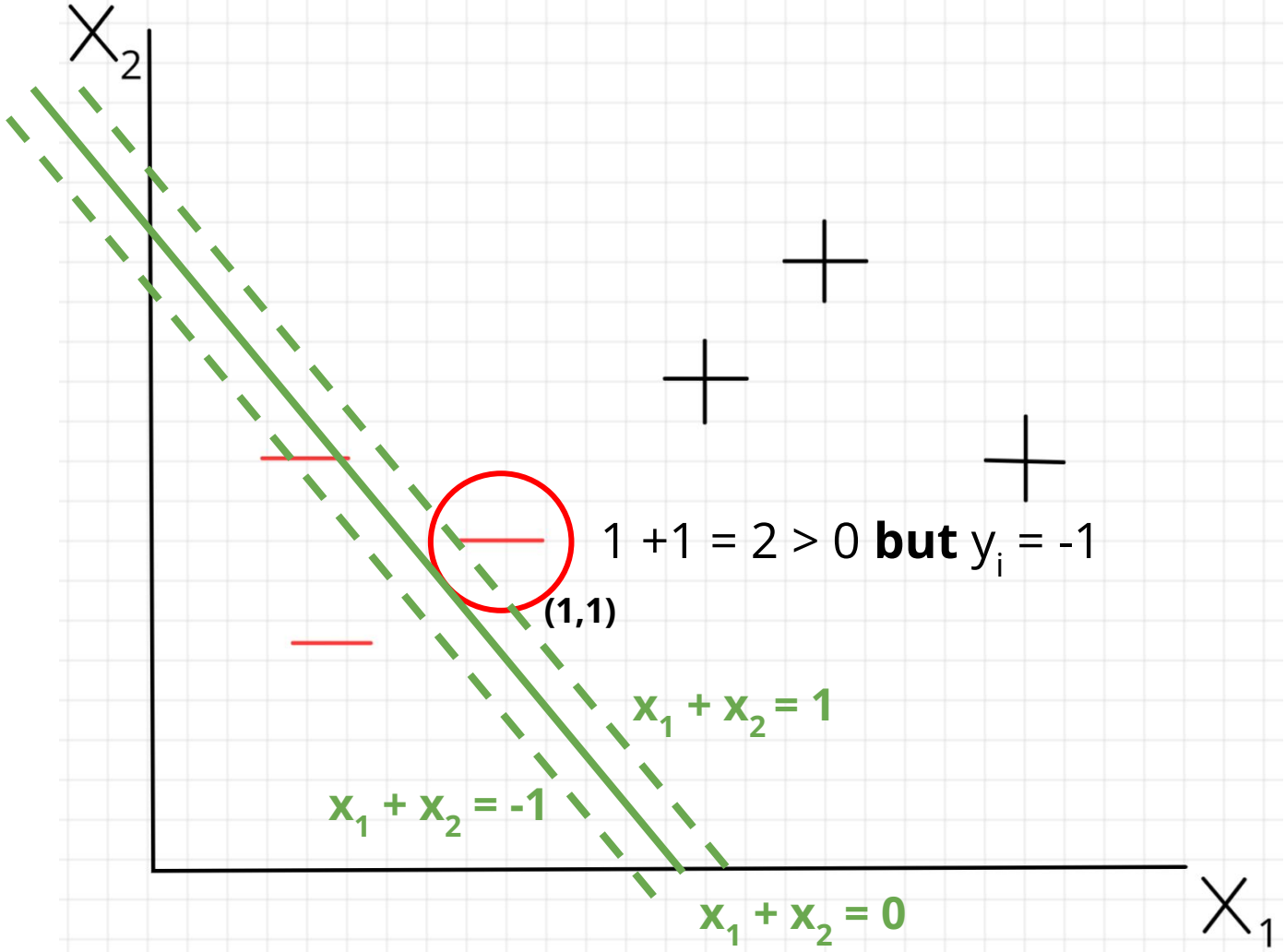


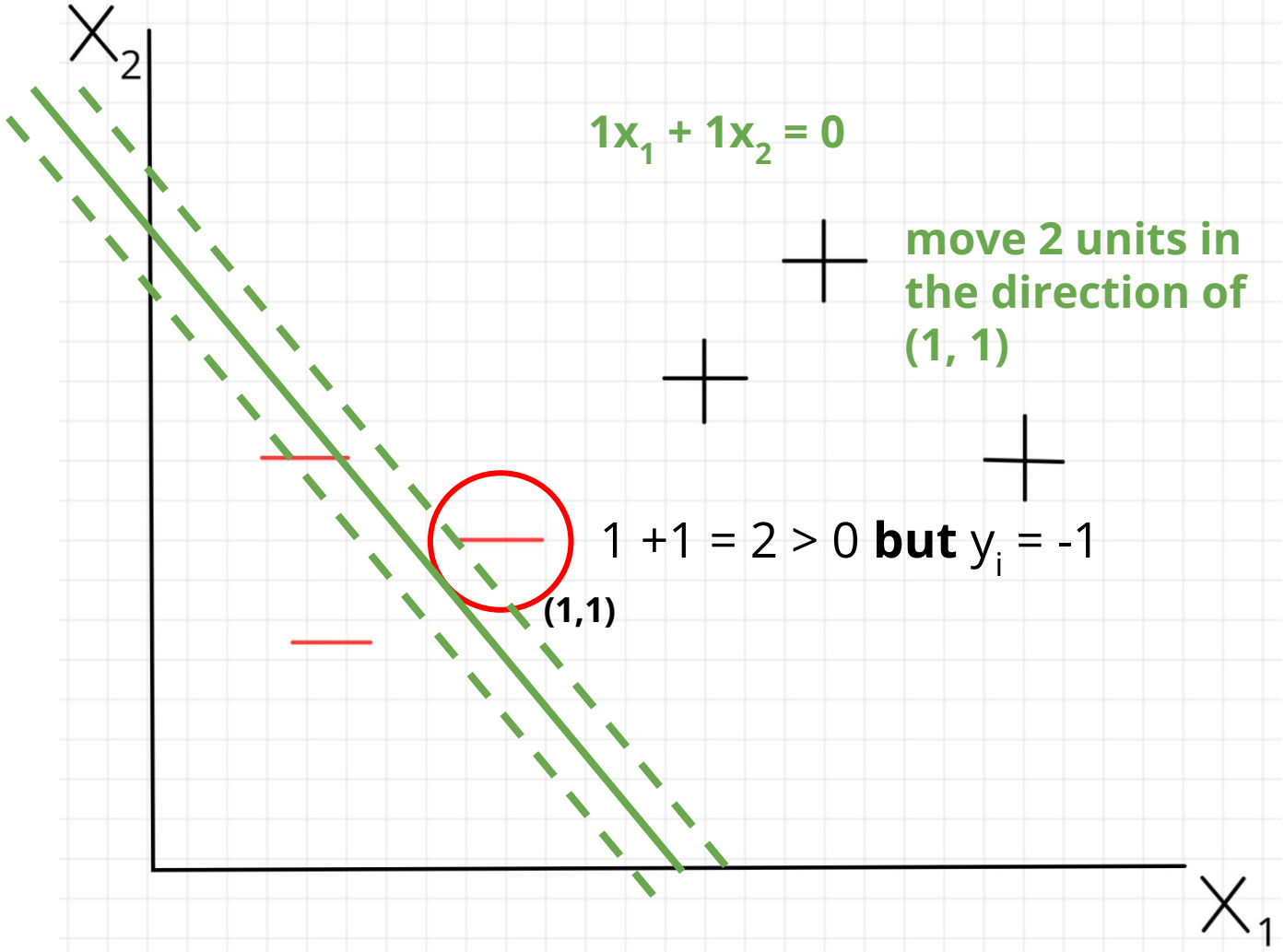
GOAL1: Data points must be classified correctly.

GOAL2: Data points cannot be in the street.

Learning w and b


X2
(1,1)
x1 + x2 = 1
x1 + x2 = -1
x1 + x2 = 0
X1


X_2
1 +1 = 2 > 0 **but** y_i = -1
(1,1)
$x_1 + x_2 = 1$
$x_1 + x_2 = -1$
$x_1 + x_2 = 0$
X_1

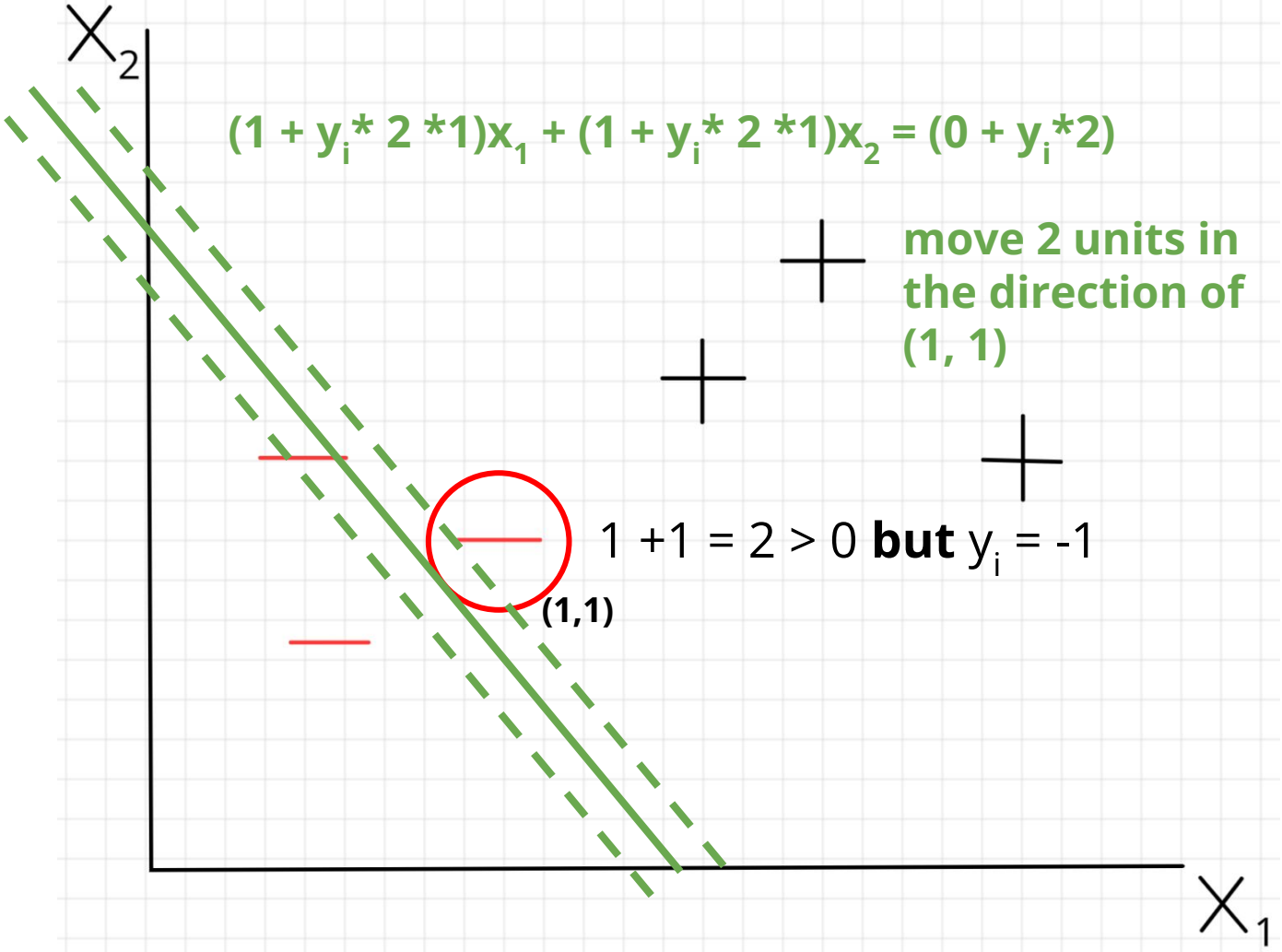

X_2
$1x_1 + 1x_2 = 0$
move 2 units in the direction of (1, 1)
1 +1 = 2 > 0 **but** $y_i = -1$
(1,1)
X_1

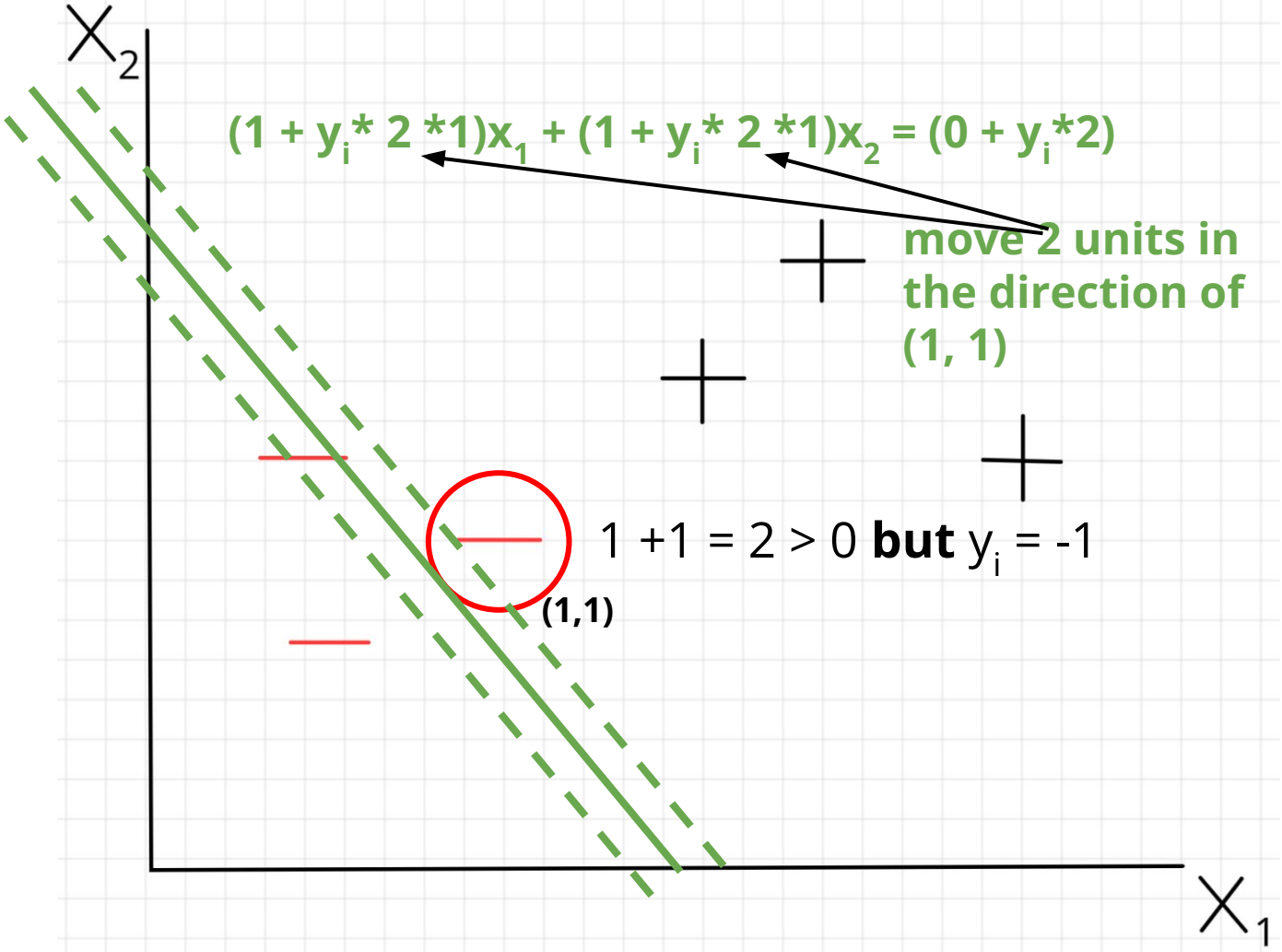
To move the street in the direction of a point

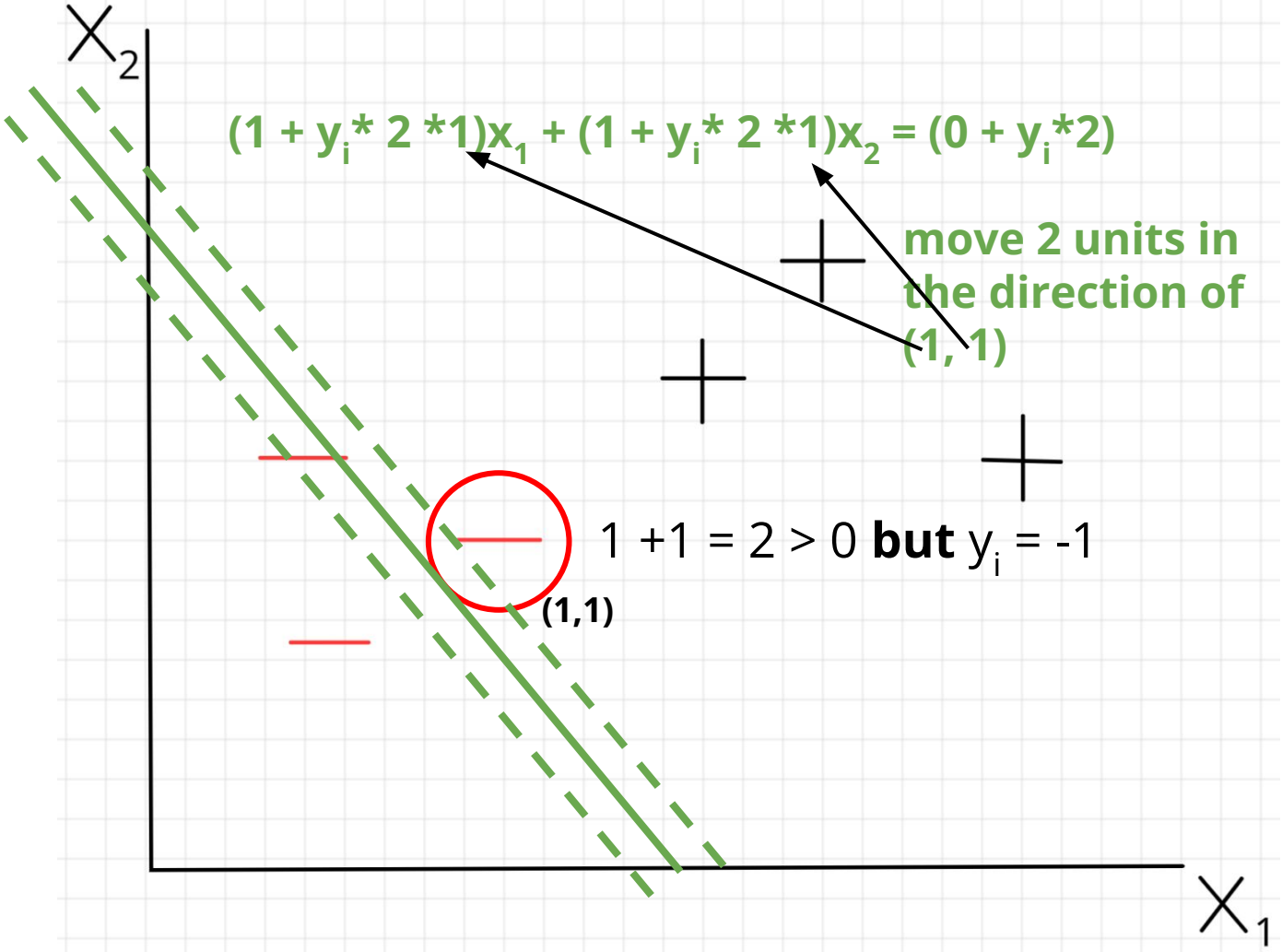
Pick a step size **a**, in order to move **a** steps in the direction of **x**

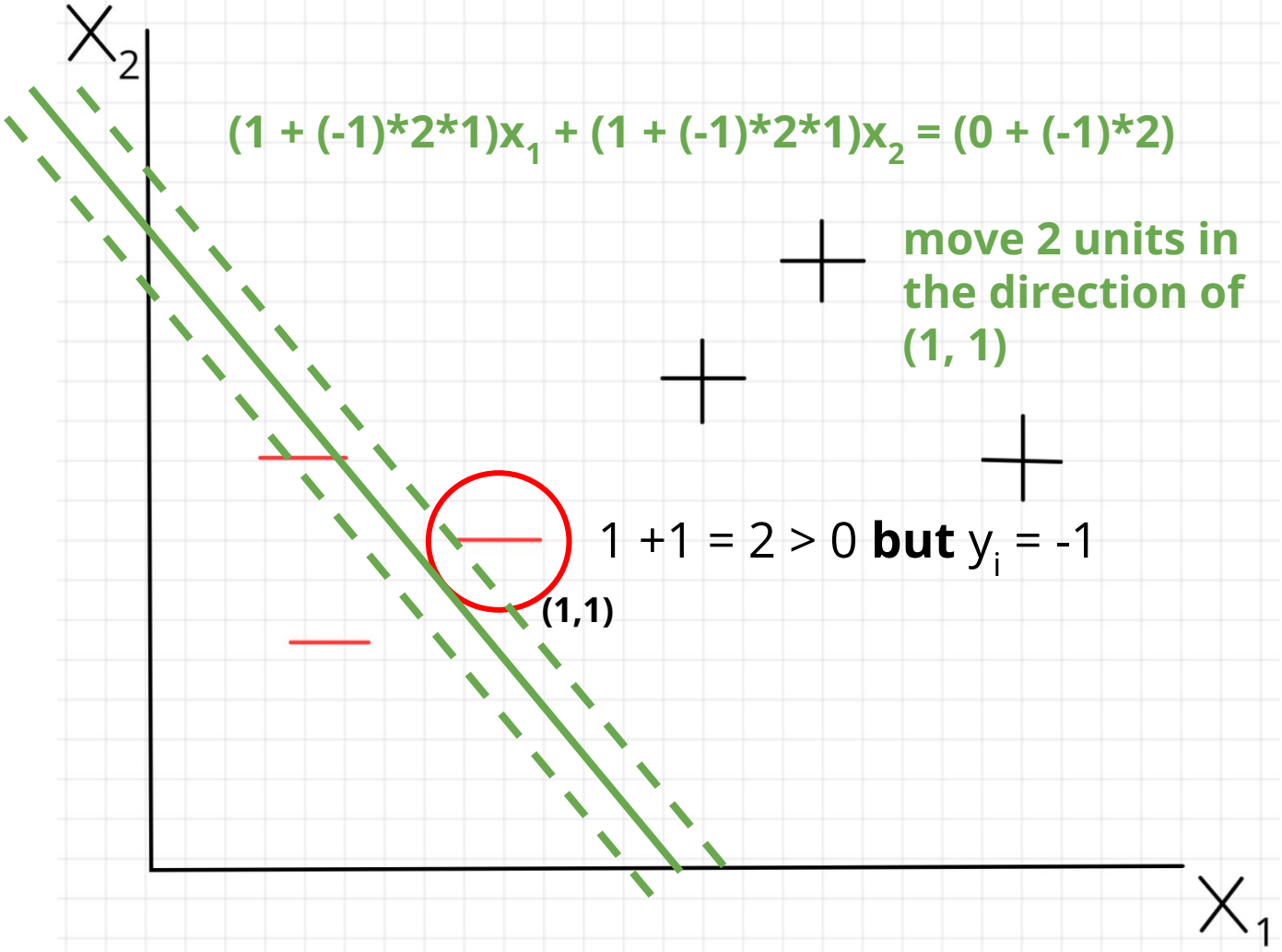
$$\mathbf{w}_{new} = \mathbf{w}_{old} + y_i * x * a$$

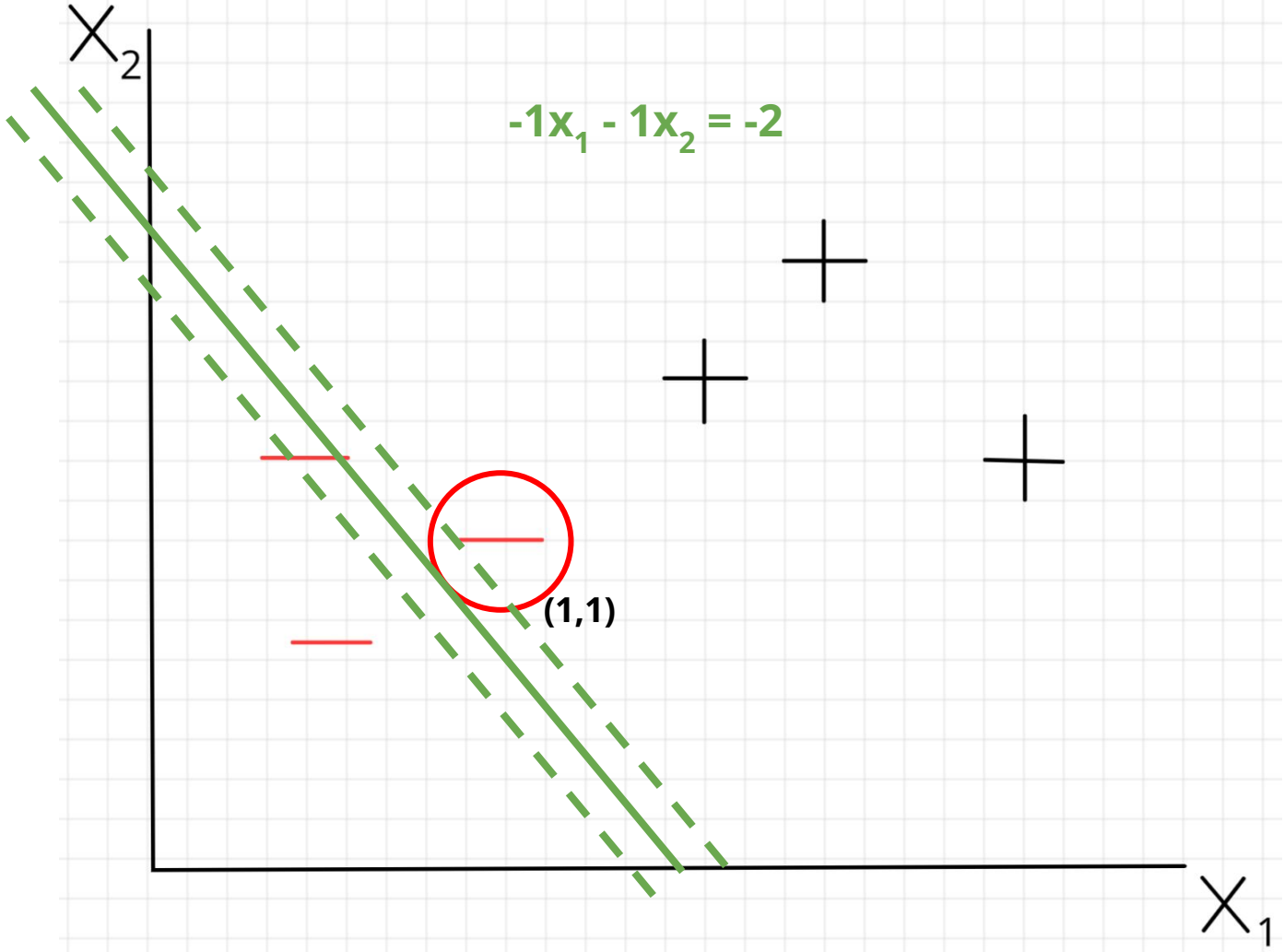
$$\mathbf{b}_{new} = \mathbf{b}_{old} + y_i * a$$

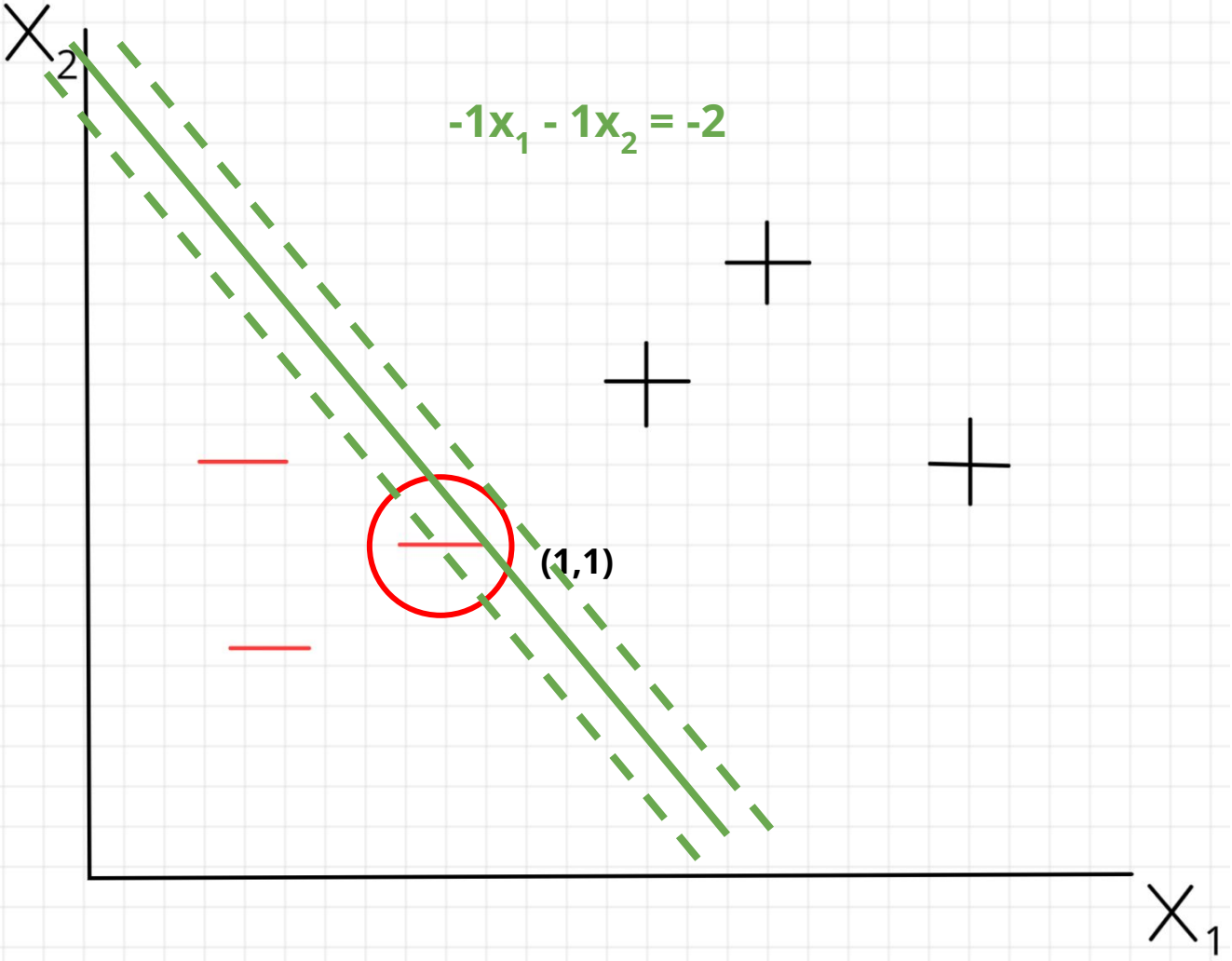

X_2
$(1 + y_i * 2 * 1)x_1 + (1 + y_i * 2 * 1)x_2 = (0 + y_i * 2)$
move 2 units in the direction of (1, 1)
1 +1 = 2 > 0 **but** $y_i = -1$
(1,1)
X_1


X_2
$(1 + y_i * 2 * 1)x_1 + (1 + y_i * 2 * 1)x_2 = (0 + y_i * 2)$
move 2 units in the direction of (1, 1)
1 +1 = 2 > 0 **but** $y_i = -1$
(1,1)
X_1

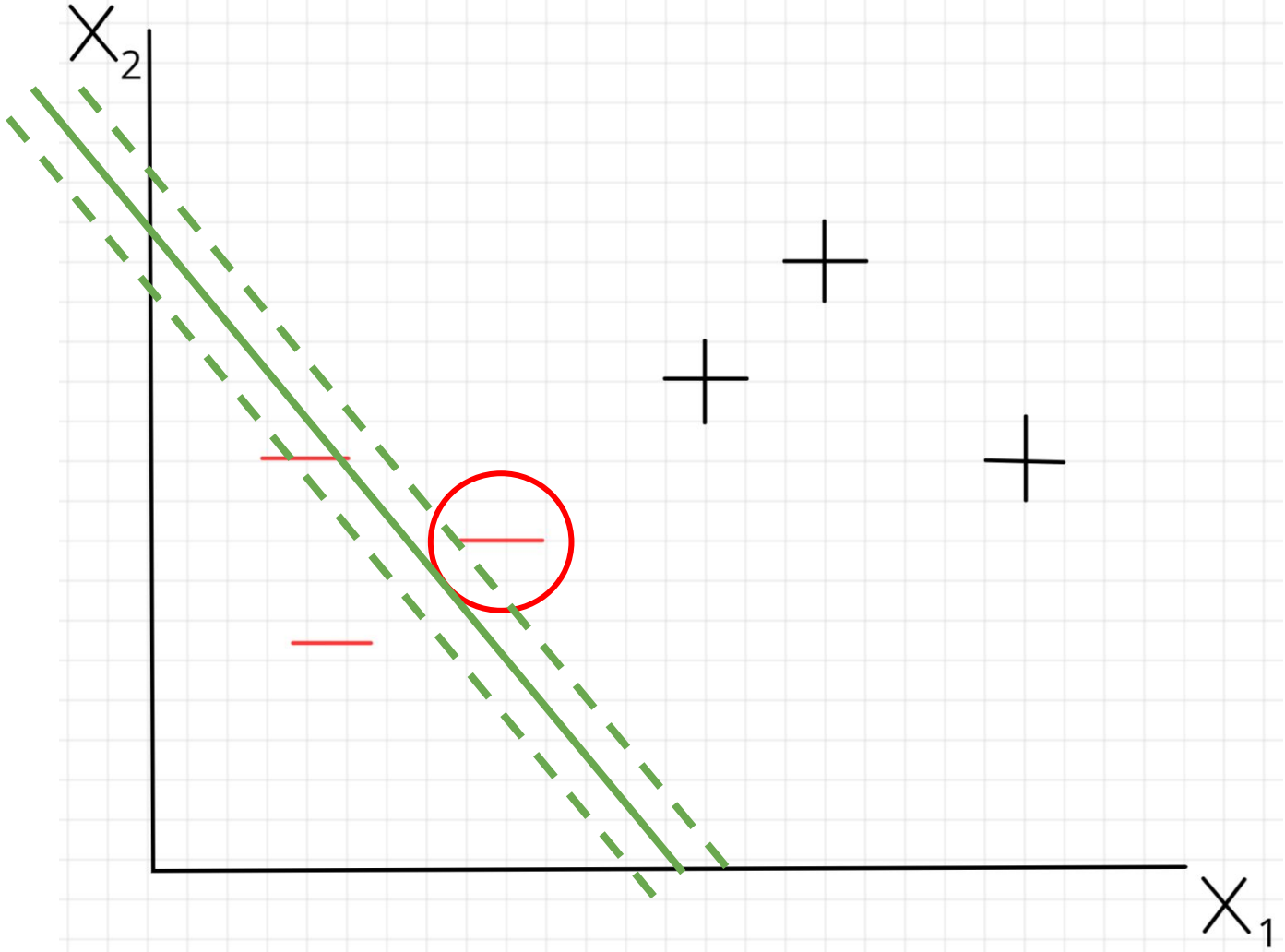

X_2
$(1 + y_i * 2 * 1)x_1 + (1 + y_i * 2 * 1)x_2 = (0 + y_i * 2)$
move 2 units in the direction of (1, 1)
1 +1 = 2 > 0 **but** $y_i = -1$
(1,1)
X_1

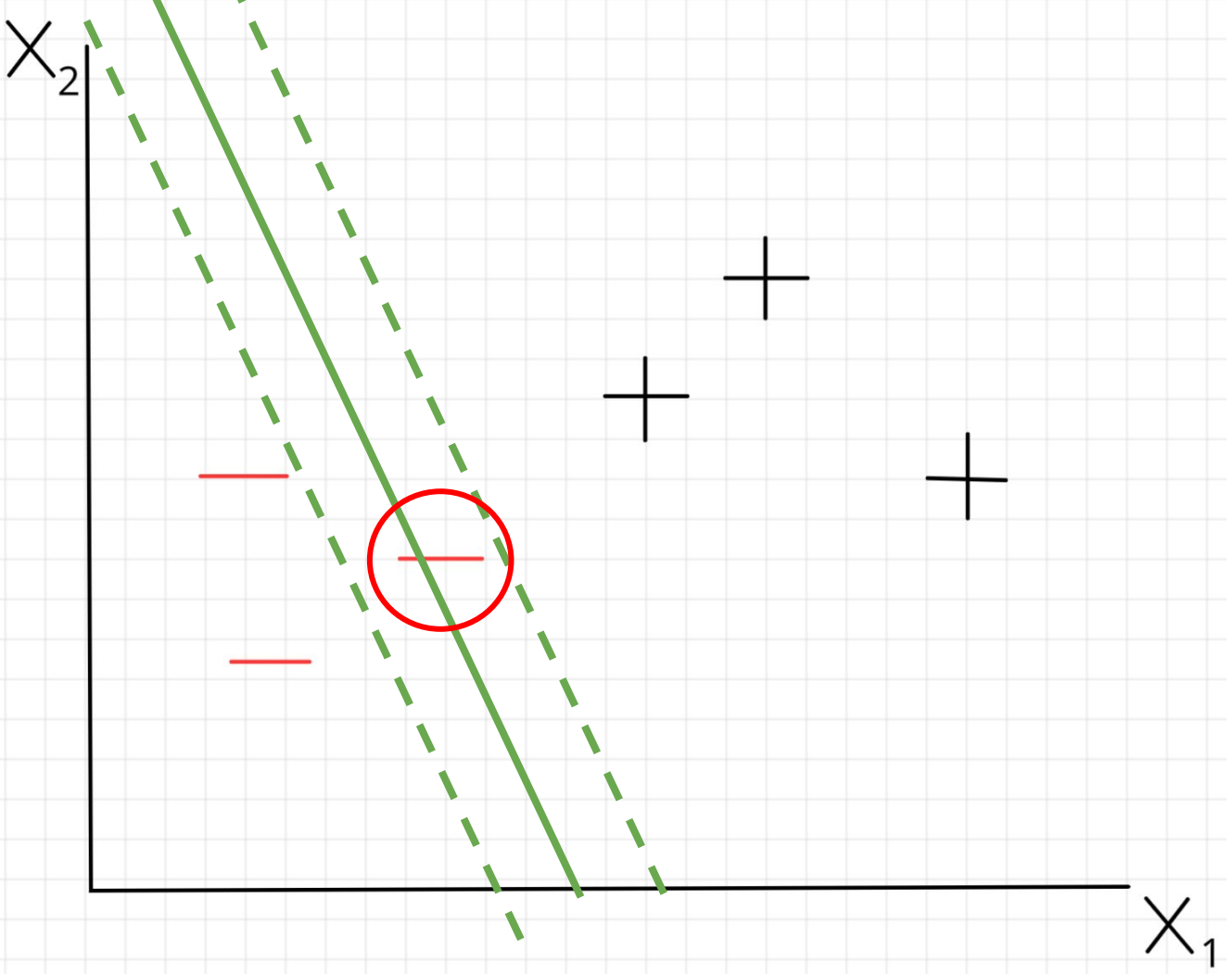

X_2
$(1 + (-1)*2*1)x_1 + (1 + (-1)*2*1)x_2 = (0 + (-1)*2)$
move 2 units in the direction of (1, 1)
1 +1 = 2 > 0 **but** $y_i = -1$
(1,1)
X_1

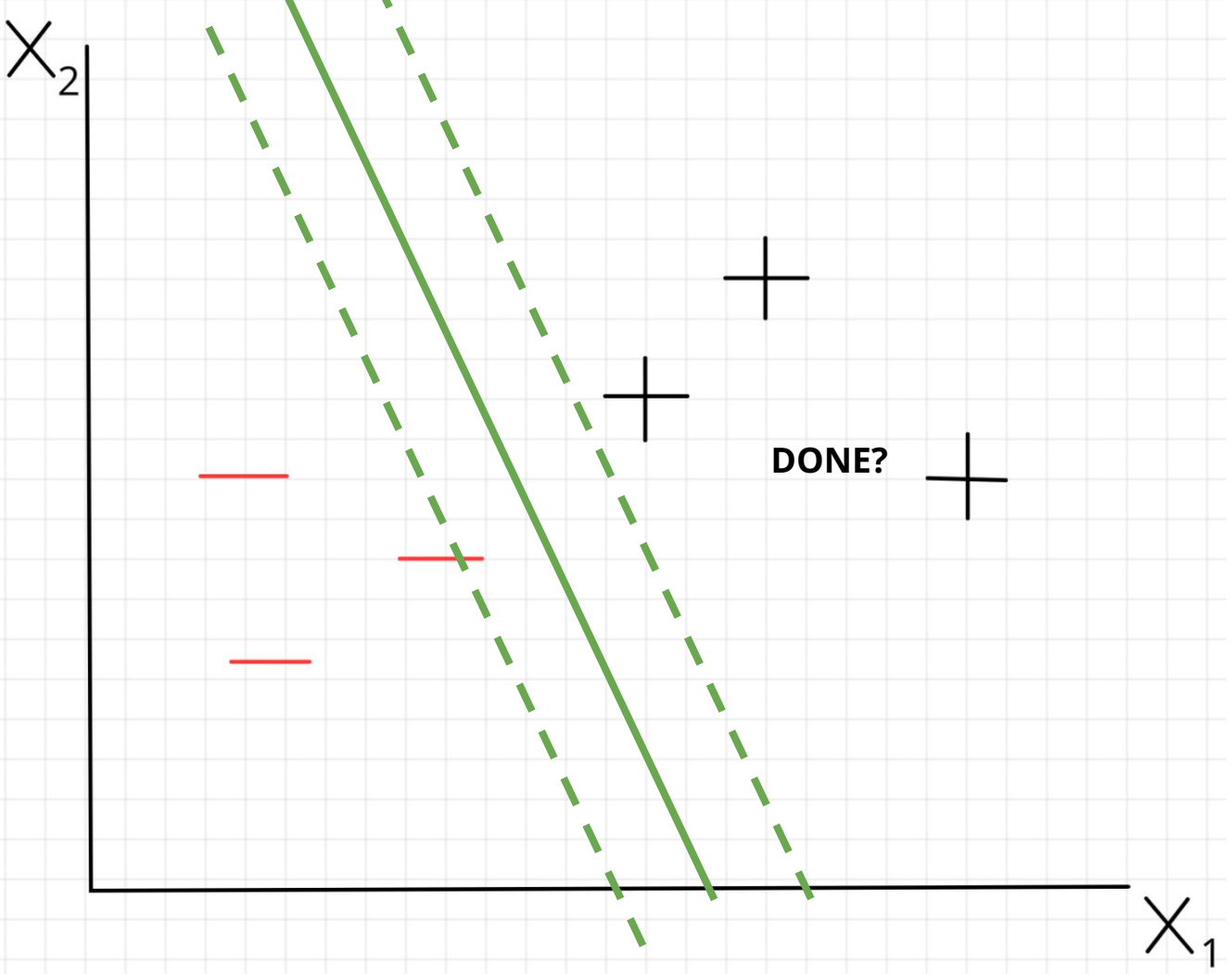

X_2
$-1x_1 - 1x_2 = -2$
(1,1)
X_1

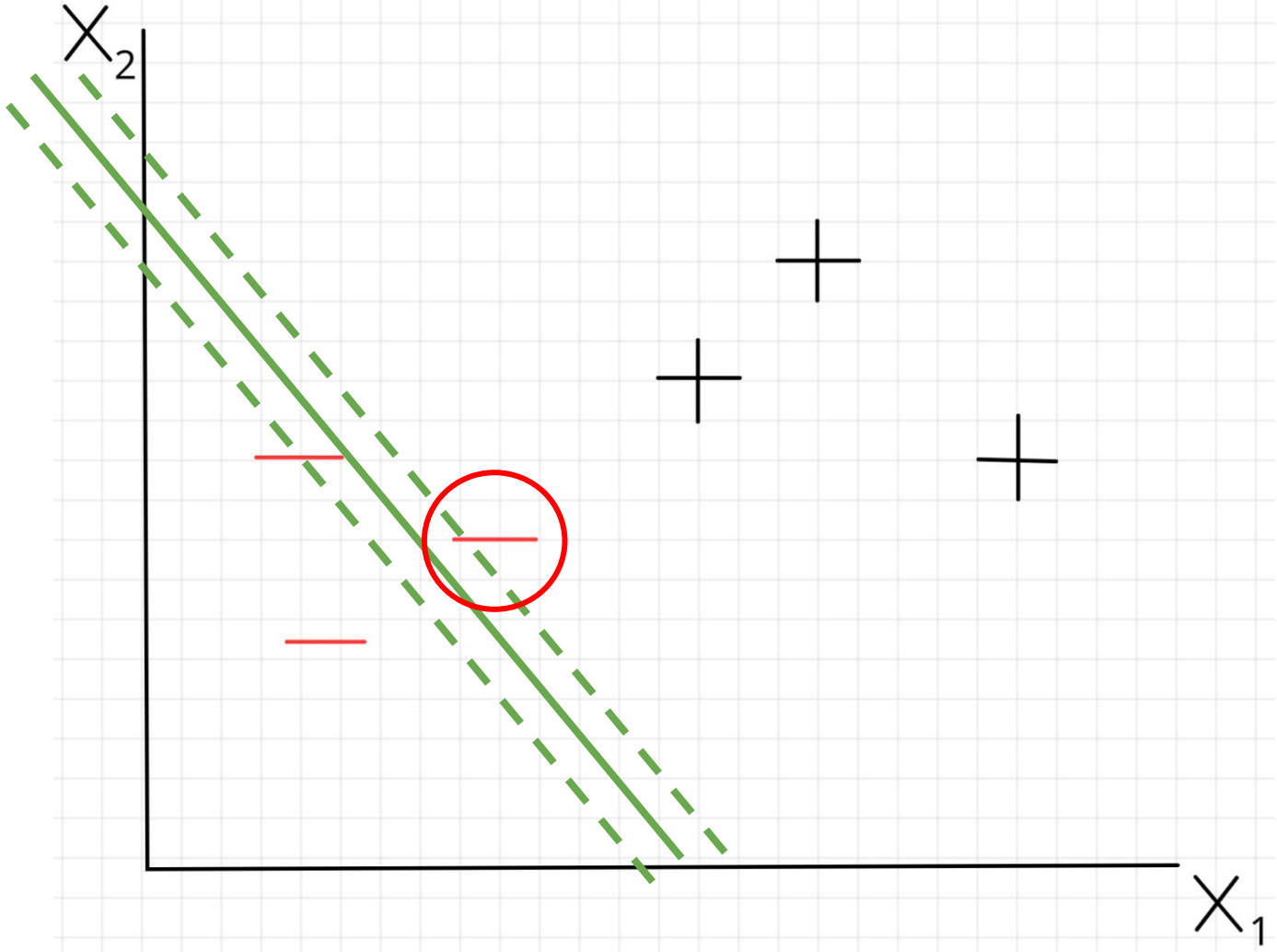

X_2
$-1x_1 - 1x_2 = -2$
(1,1)
X_1

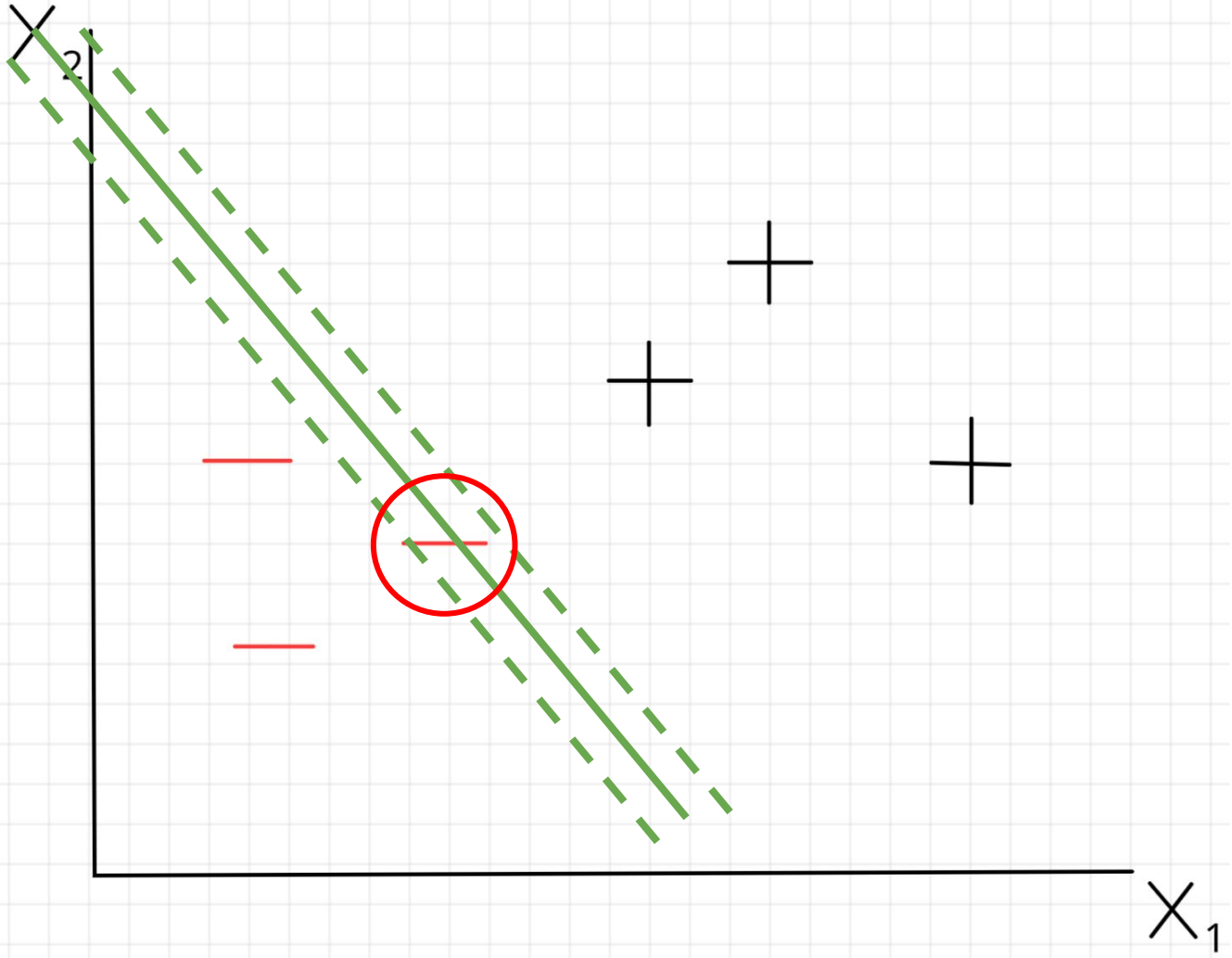
Now we know how to move the street in the right direction - but...

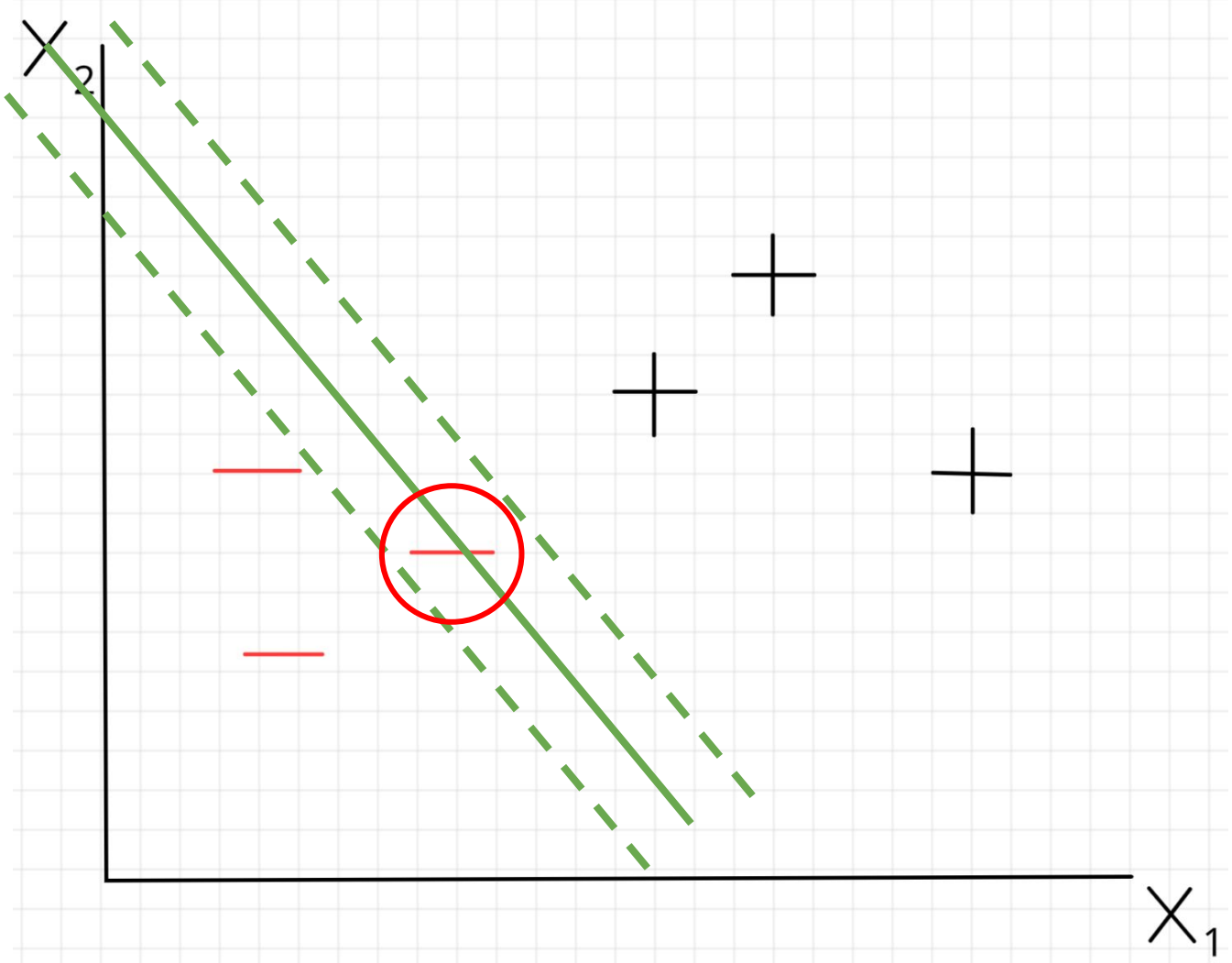

X2
X1

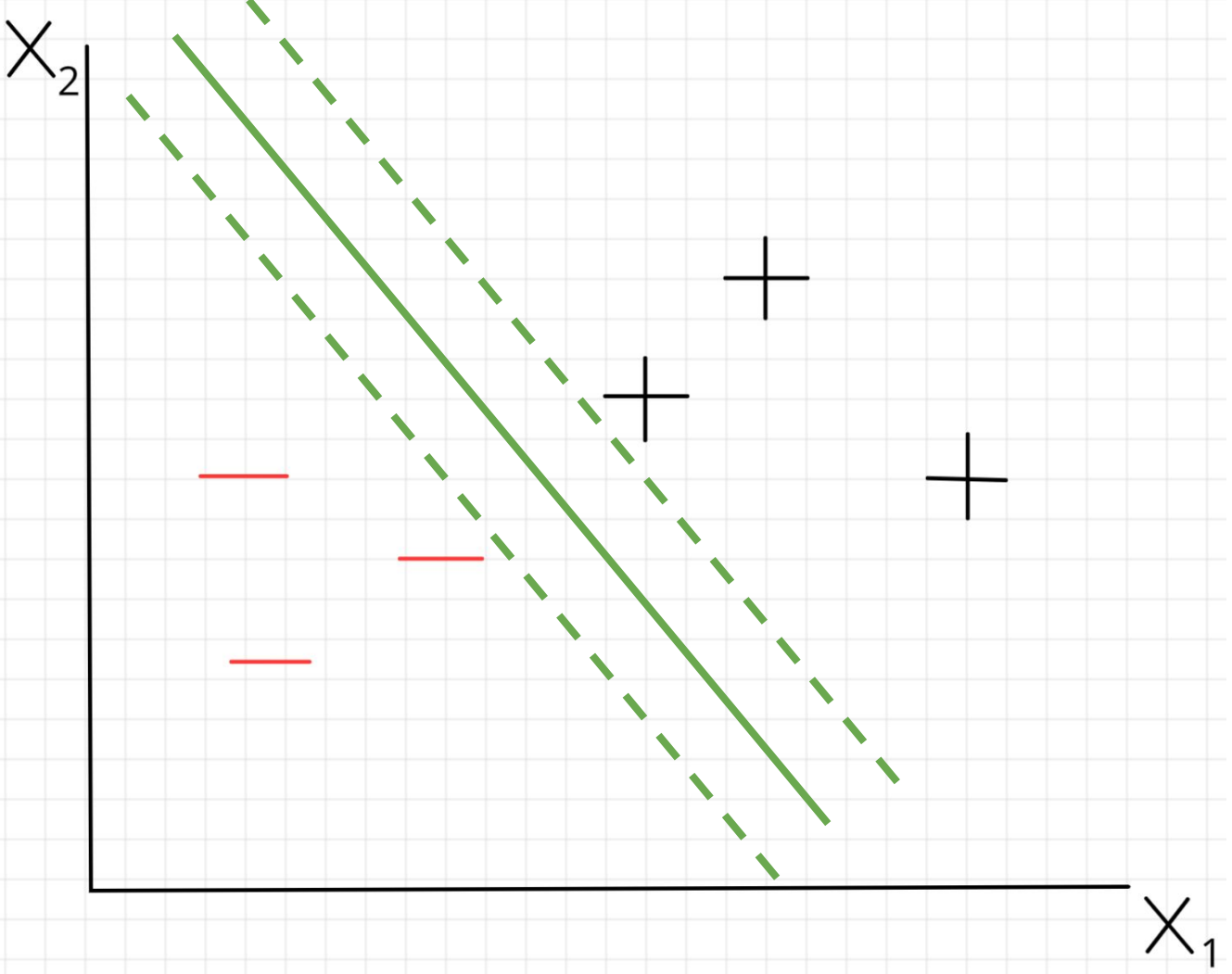

X2
X1

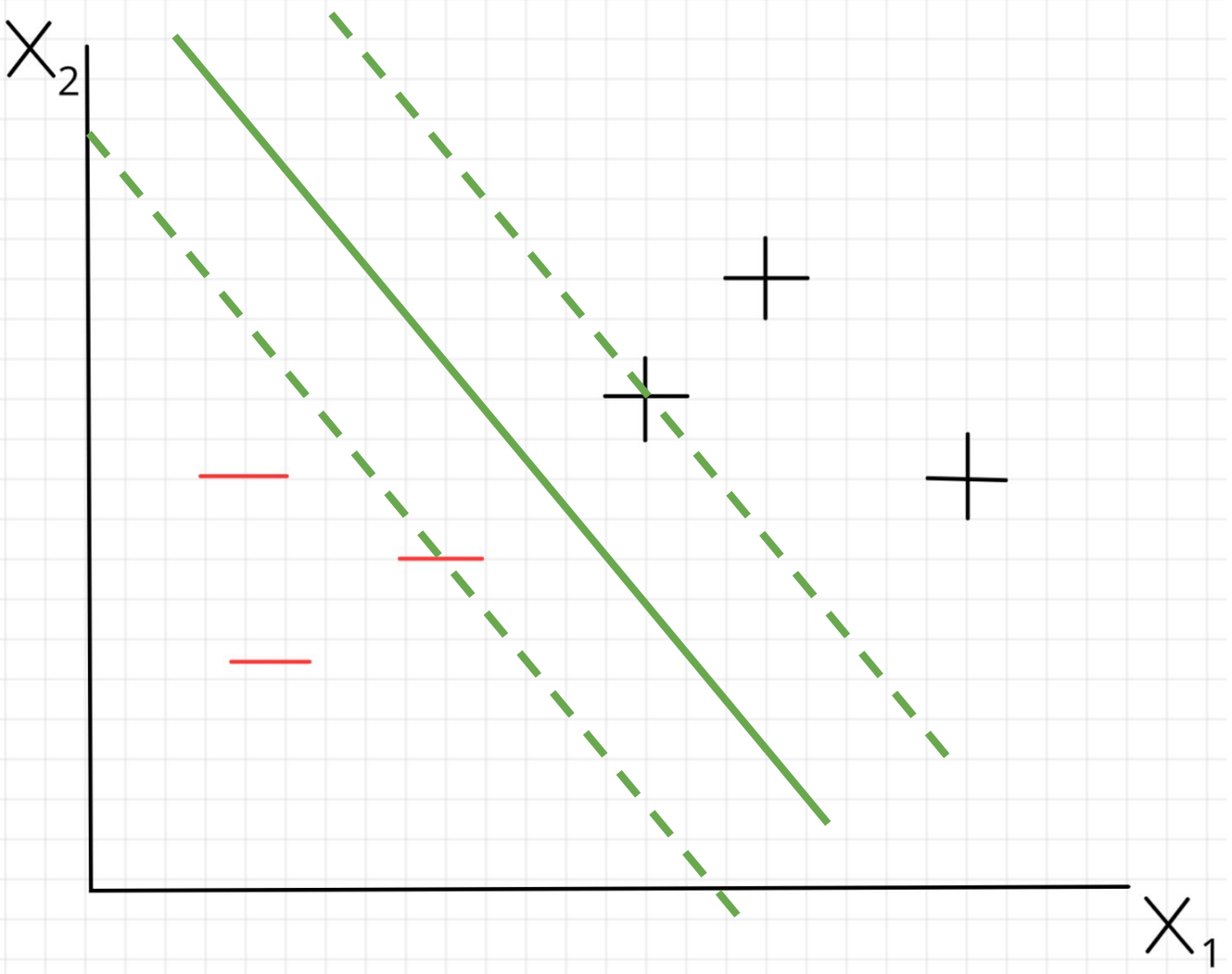

X_2
DONE?
X_1


X2
X1


X2
X1

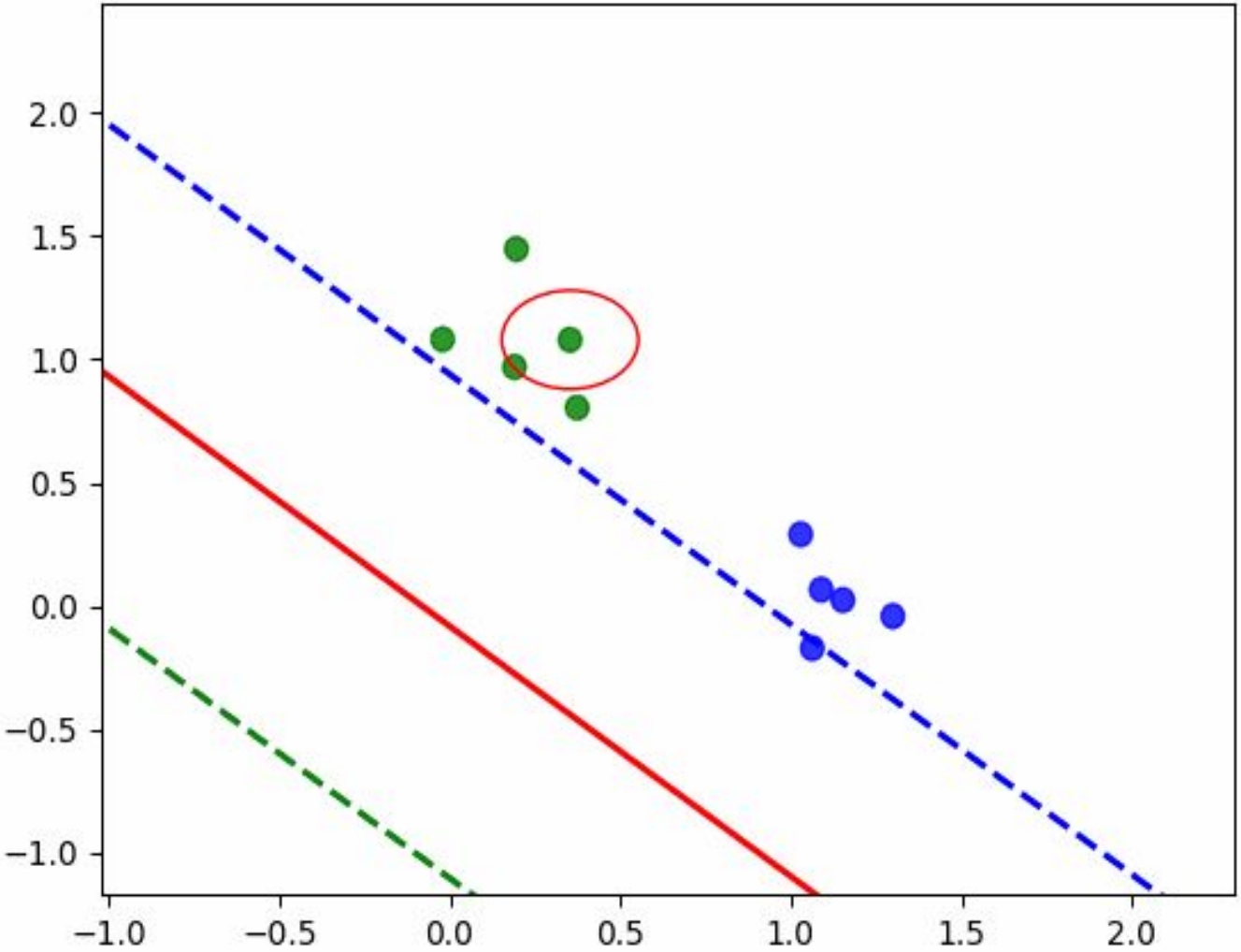

X2
X1


X2
X1

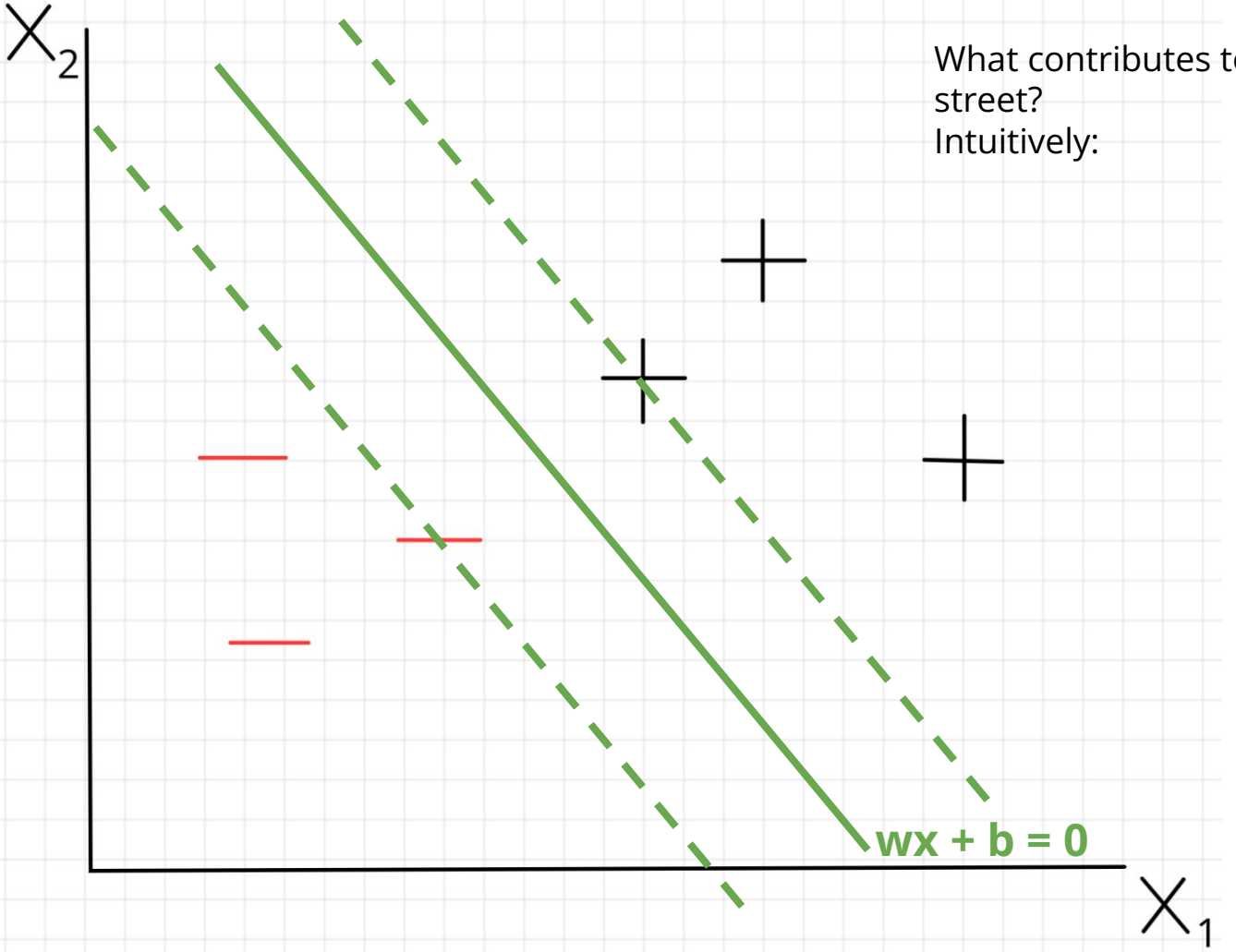

X2
X1

Full Algorithm (Perceptron Algorithm)

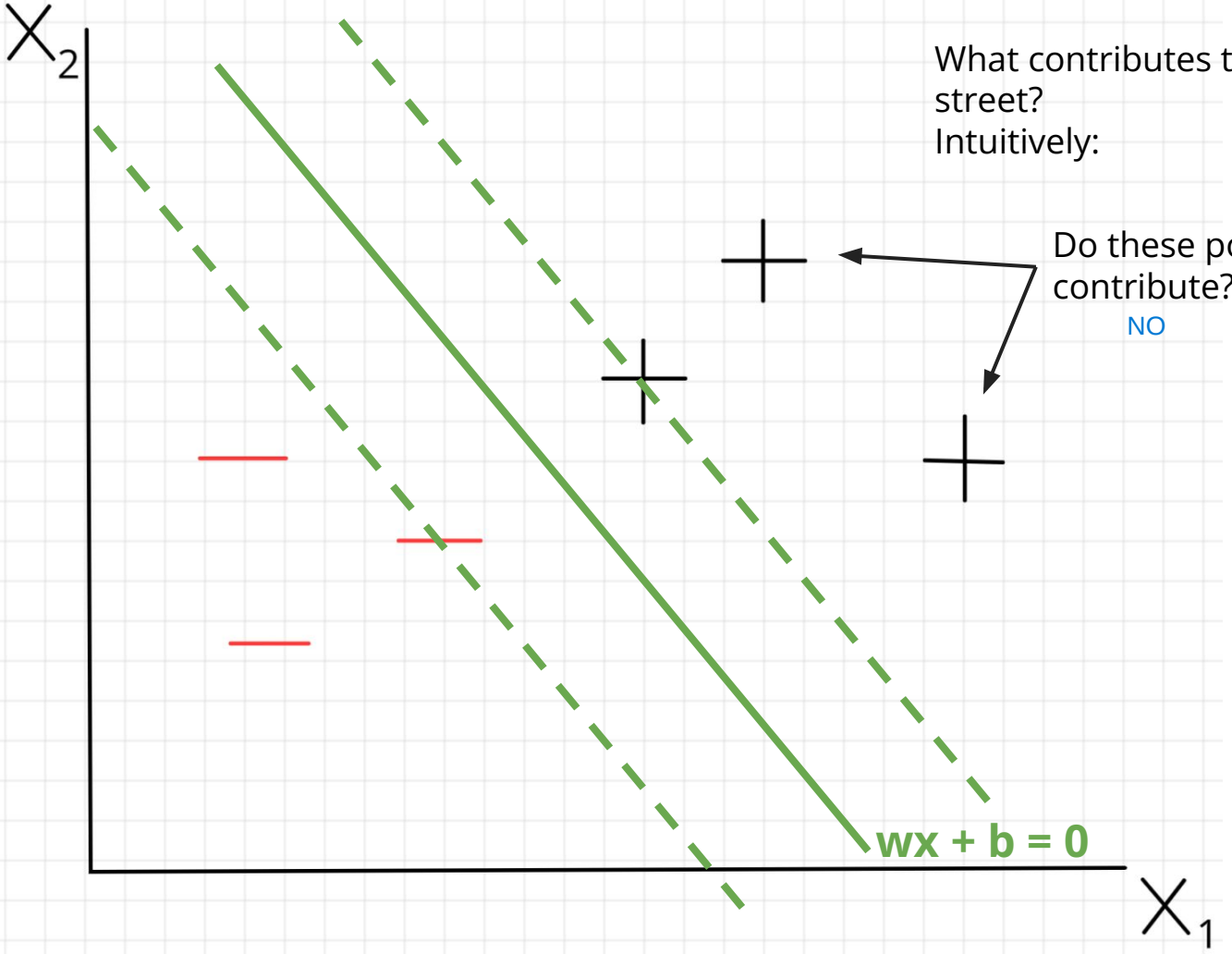
- Start with random line $w_1x_1 + w_2x_2 + b = 0$
- Define:
  - A total number of iterations (ex: 100)
  - A learning rate **a** (not too big not too small)
  - An expanding rate **c** (< 1 but not too close to 1)
- Repeat **number of iterations** times:
  - Pick a point (x_i, y_i) from the dataset
  - If correctly classified: do nothing
  - If incorrectly classified:
    - Adjust w_1 by adding $(y_i * a * x_1)$, w_2 by adding $(y_i * a * x_2)$, and b by adding $(y_i * a)$
  - Expand or retract the width by **c** (multiply the new line by **c**)

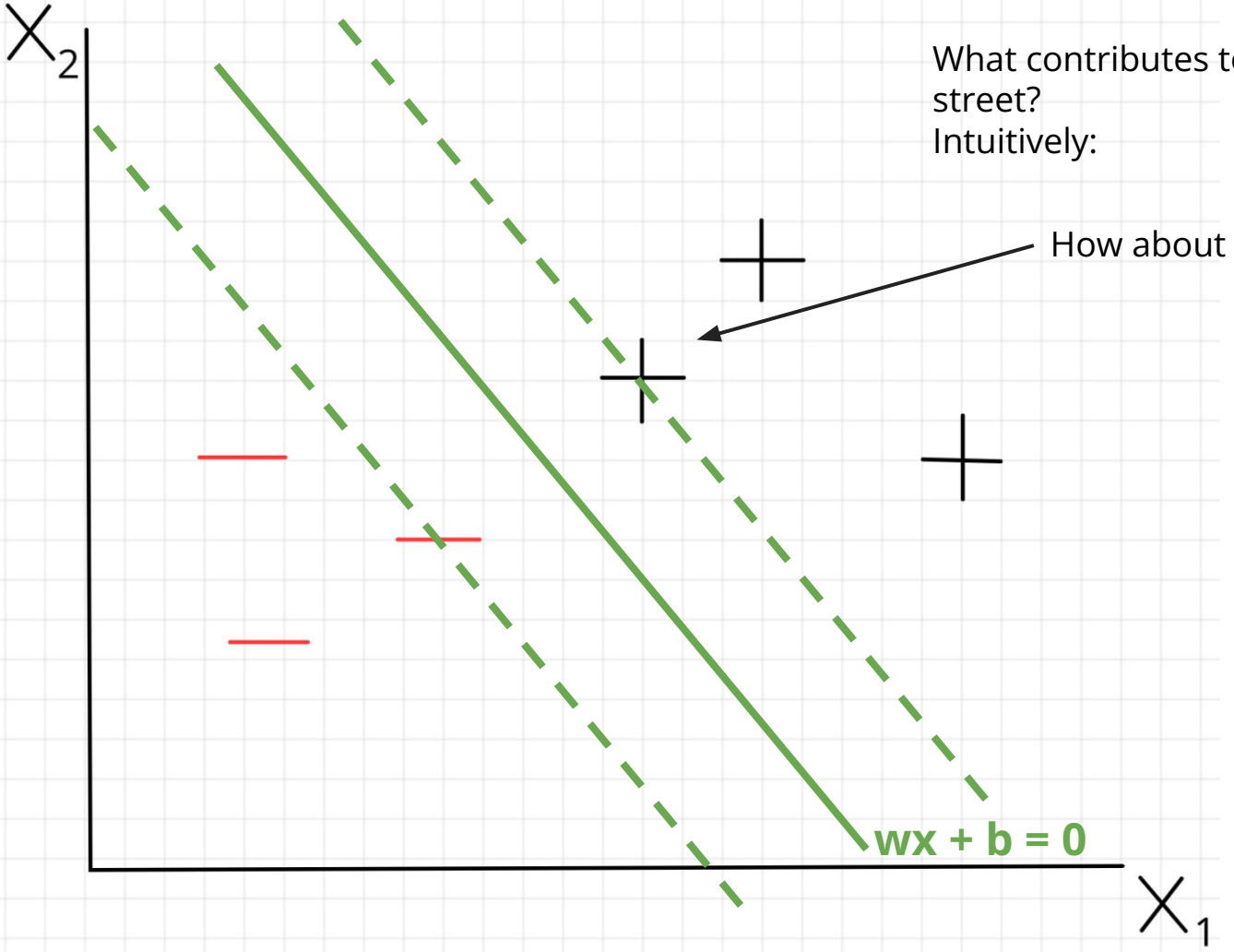

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

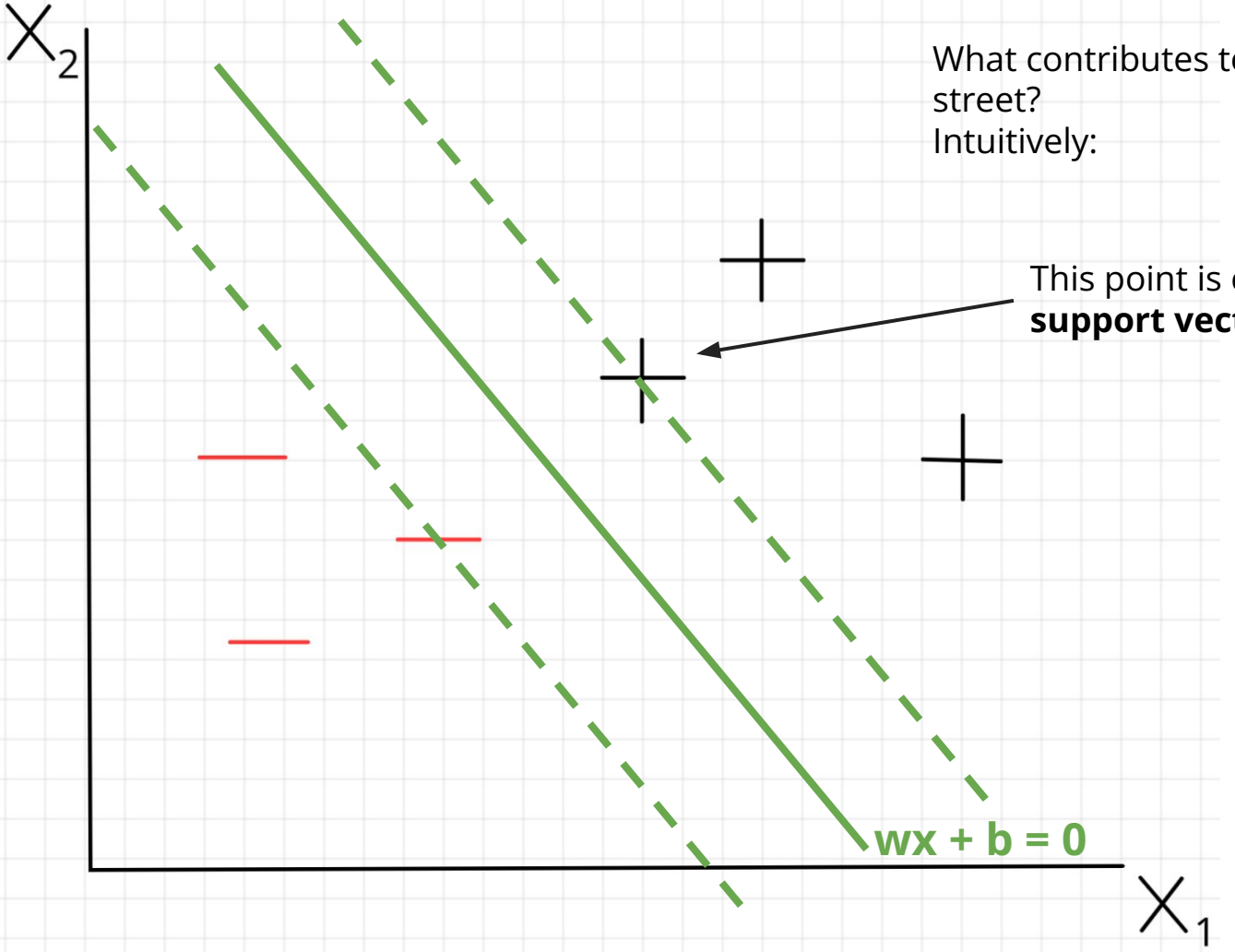
Diving into the math



What contributes to the widest street?
Intuitively:


What contributes to the widest street?
Intuitively:
Do these points contribute?
NO
wx + b = 0
X2
X1


What contributes to the widest street?
Intuitively:
How about this point?
wx + b = 0
X2
X1


What contributes to the widest street?
Intuitively:
This point is called a **support vector**
X_2
X_1
$wx + b = 0$

Find the widest street subject to...

We want our samples to lie beyond the street. That is:

$$\vec{w} \cdot \vec{x}_+ + b \geq 1$$

$$\vec{w} \cdot \vec{x}_- + b \leq -1$$

Note: for an unknown **u**, we can have

$$-1 < \vec{w} \cdot \vec{u} + b < 1$$

Find the widest street subject to...

Let's introduce a variable

$$y_i = \begin{cases} +1 & \text{if } x_i \text{ is a } + \text{sample} \\ \\ -1 & \text{if } x_i \text{ is a } - \text{sample} \end{cases}$$

Note: this is effectively the class label of x_i

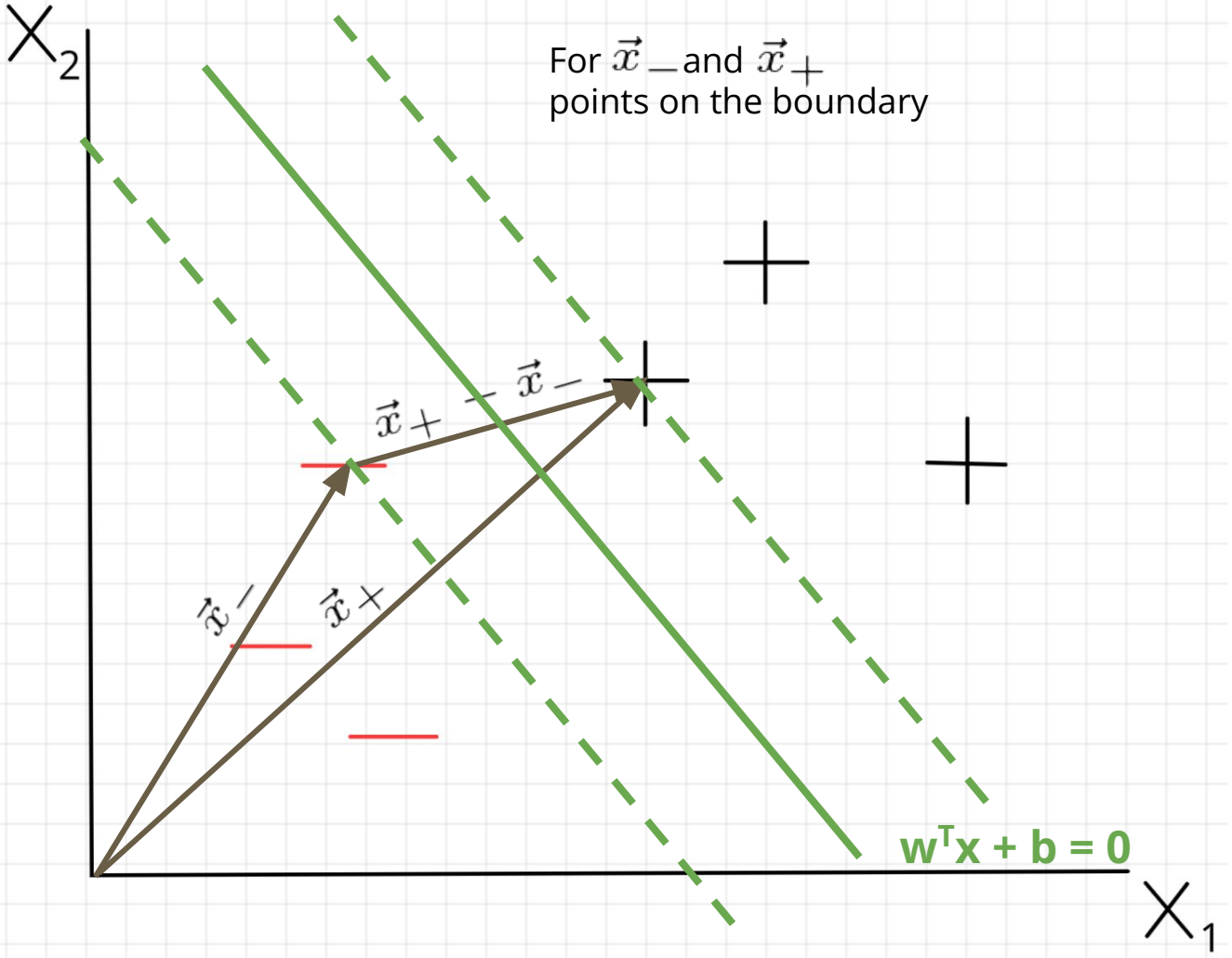
Find the widest street subject to...

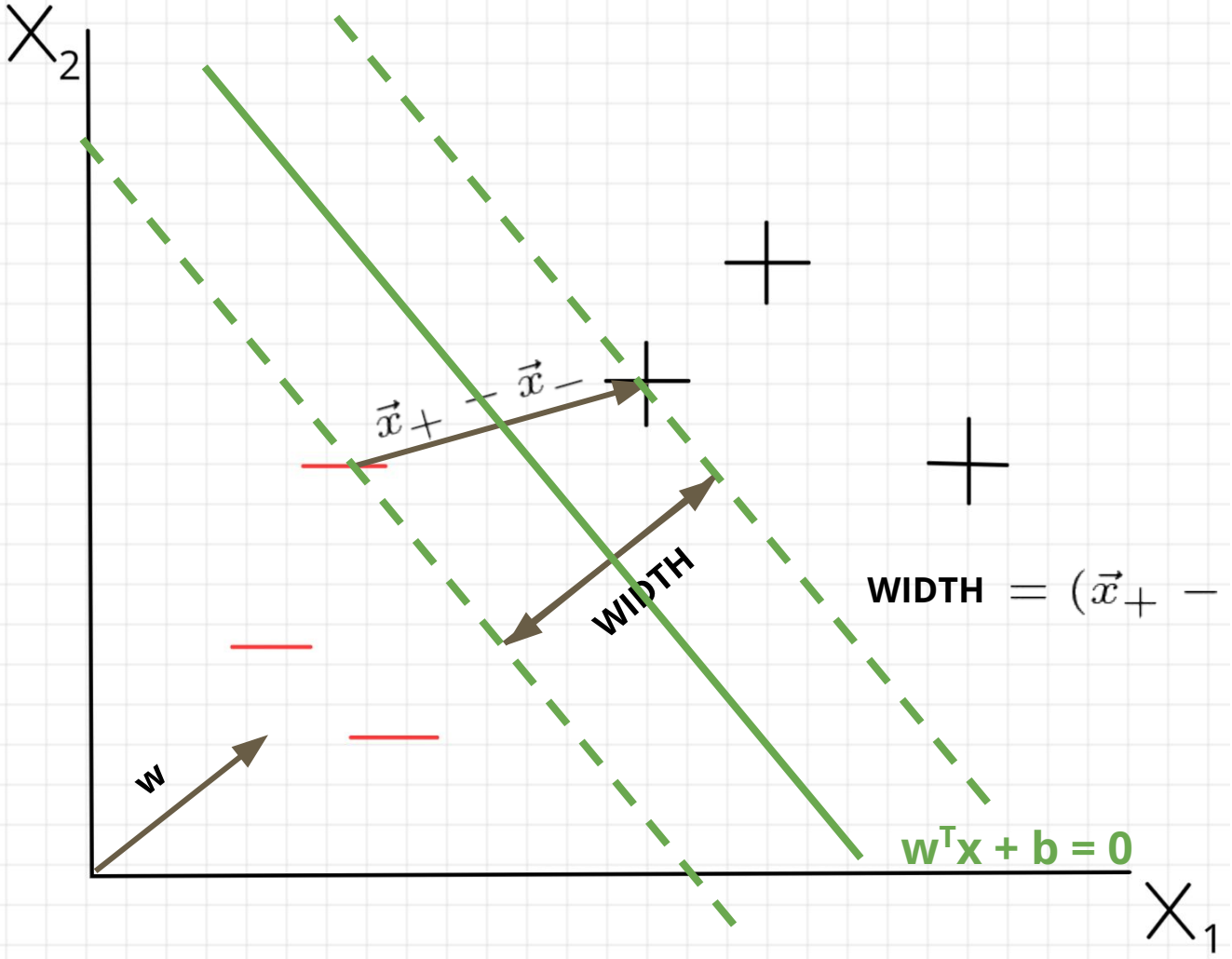
If we multiply our sample decision rules by this new variable:

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1$$

Meaning, for x_i on the decision boundary, we want:

$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 = 0$$


For $\vec{x}_-$ and $\vec{x}_+$
points on the boundary
X_2
X_1
$\vec{x}_+ - \vec{x}_-$
$\vec{x}_-$
$\vec{x}_+$
$w^T x + b = 0$



$$\text{WIDTH} = (\vec{x}_+ - \vec{x}_-) \cdot \frac{\vec{w}}{\|\vec{w}\|}$$

How to find the width of the street

We know that **WIDTH** $= (\vec{x}_+ - \vec{x}_-) \cdot \frac{\vec{w}}{\|\vec{w}\|}$ for $\vec{x}_-$ and $\vec{x}_+$ points on the boundary

And, since they are on the boundary, we know that

$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 = 0$$

Hence, **WIDTH = ?**

How to find the width of the street

We know that **WIDTH** $= (\vec{x}_+ - \vec{x}_-) \cdot \frac{\vec{w}}{\|\vec{w}\|}$ for $\vec{x}_-$ and $\vec{x}_+$ points on the boundary

And, since they are on the boundary, we know that

$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 = 0$$

Hence, **WIDTH** $= \frac{2}{\|\vec{w}\|}$

What does that mean?

Size of **w** is inversely proportional to the width of the street.

Aligns with what we found previously.

How to find the widest street

Goal is to maximize the width

$$\max\left(\frac{2}{\|\vec{w}\|}\right)$$

Subject to:

$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 = 0$$

How to find the widest street

Goal is to maximize the width

$$\max\left(\frac{2}{\|\vec{w}\|}\right) = \min(\|\vec{w}\|)$$

$$= \min\left(\frac{1}{2}\|\vec{w}\|^2\right)$$

Subject to:

$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 = 0$$

How to find the widest street

Can use Lagrange multipliers to form a single expression to find the extremum of

$$L = \frac{1}{2}\|\vec{w}\|^2 - \sum_i \alpha_i \left[y_i(\vec{x}_i \cdot \vec{w} + b) - 1\right]$$

where α_i is 0 for x_i not on the boundary.

Let's take the partial derivative of **L** wrt to **w** to see what **w** looks like at the extremum of **L**.

How to find the widest street

$$\frac{\partial L}{\partial \vec{w}} = \vec{w} - \sum_i \alpha_i y_i \vec{x}_i = 0$$

$$\implies \vec{w} = \sum_i \alpha_i y_i \vec{x}_i$$

Means **w** is a linear sum of vectors in our sample/training set!

$$\sum_i \alpha_i < x_i, x > + b \geq 0 \quad \text{then} +$$

To move the street in the direction of a point

$$\mathbf{a}_{i,new} = \mathbf{a}_{i,old} + y_i * a$$

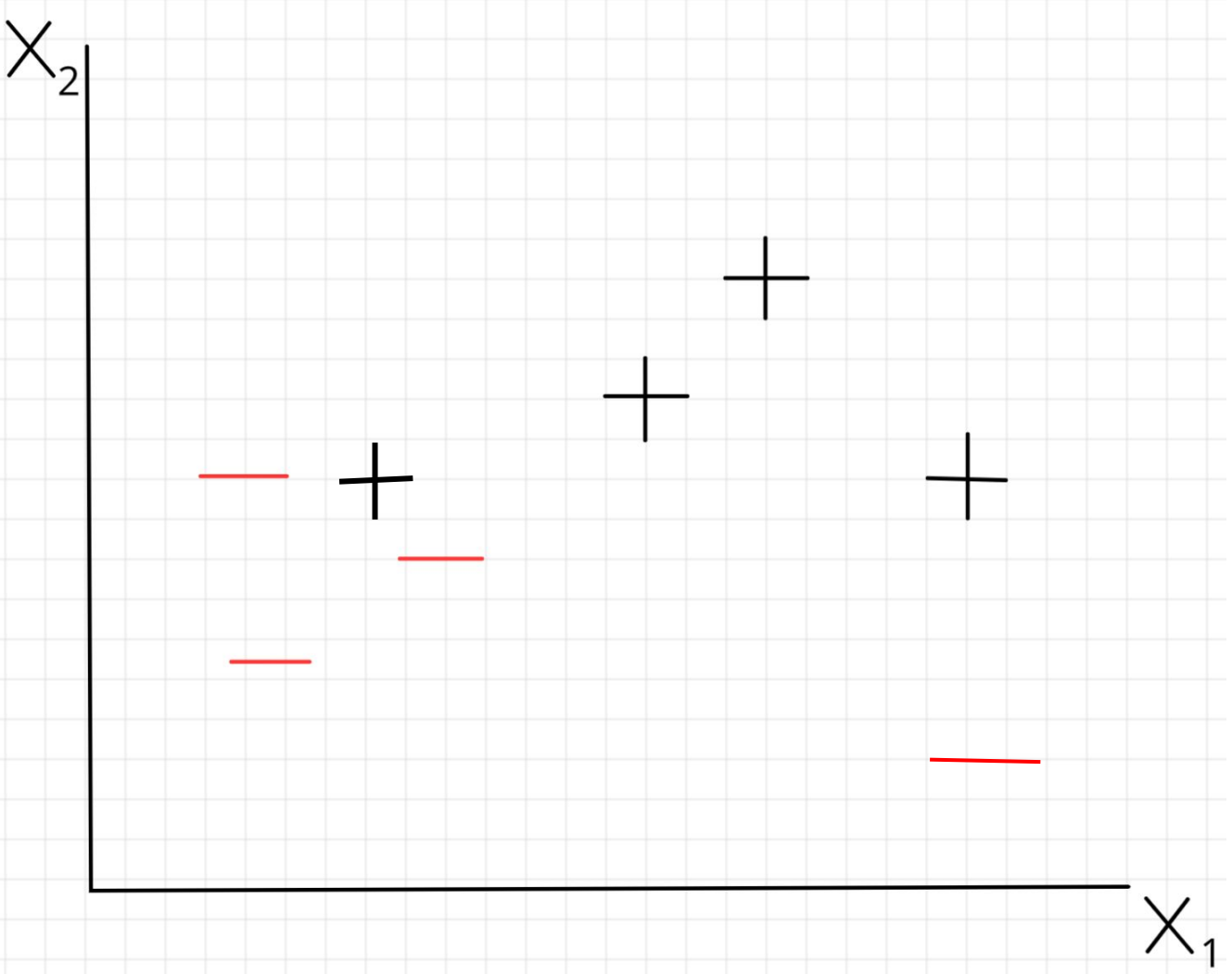
$$\mathbf{b}_{new} = \mathbf{b}_{old} + y_i * a$$

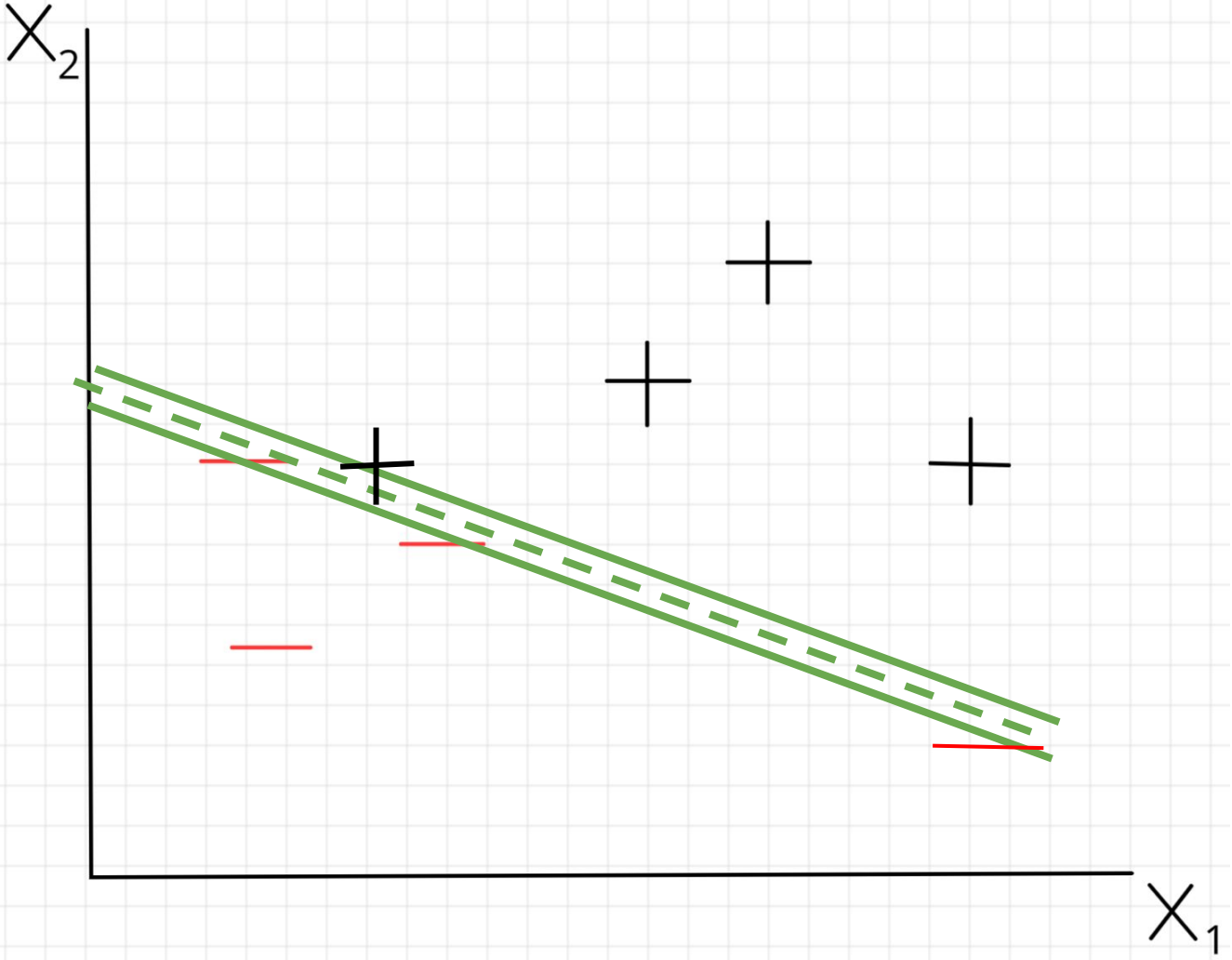
Quadratic Programming

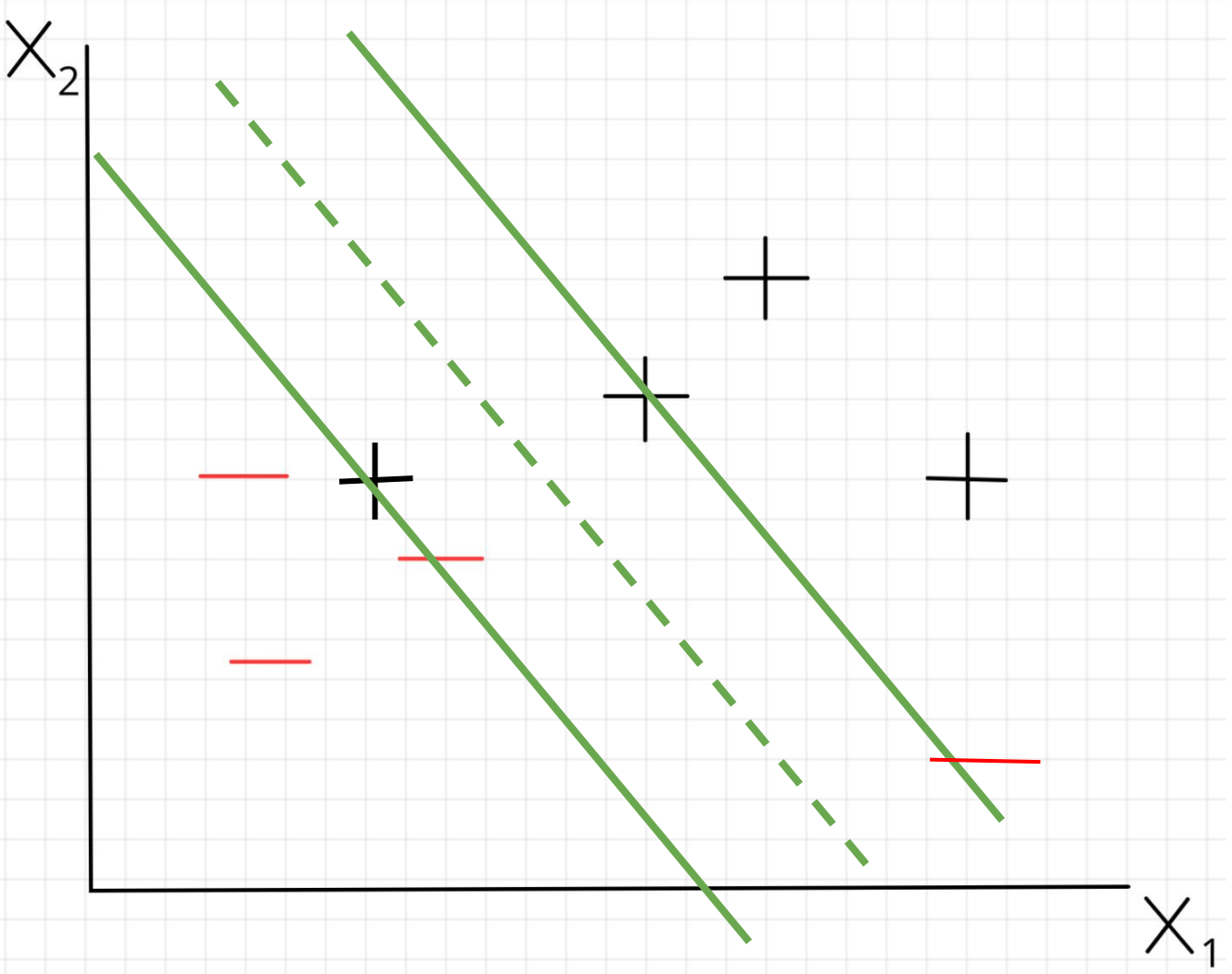
Solving lagrange multipliers requires numerical optimization methods called quadratic programming solvers.

https://pypi.org/project/qpsolvers/
https://cvxopt.org/examples/tutorial/qp.html

Trade-off between width and error


X_2
X_1


X2
X1


X2
X1

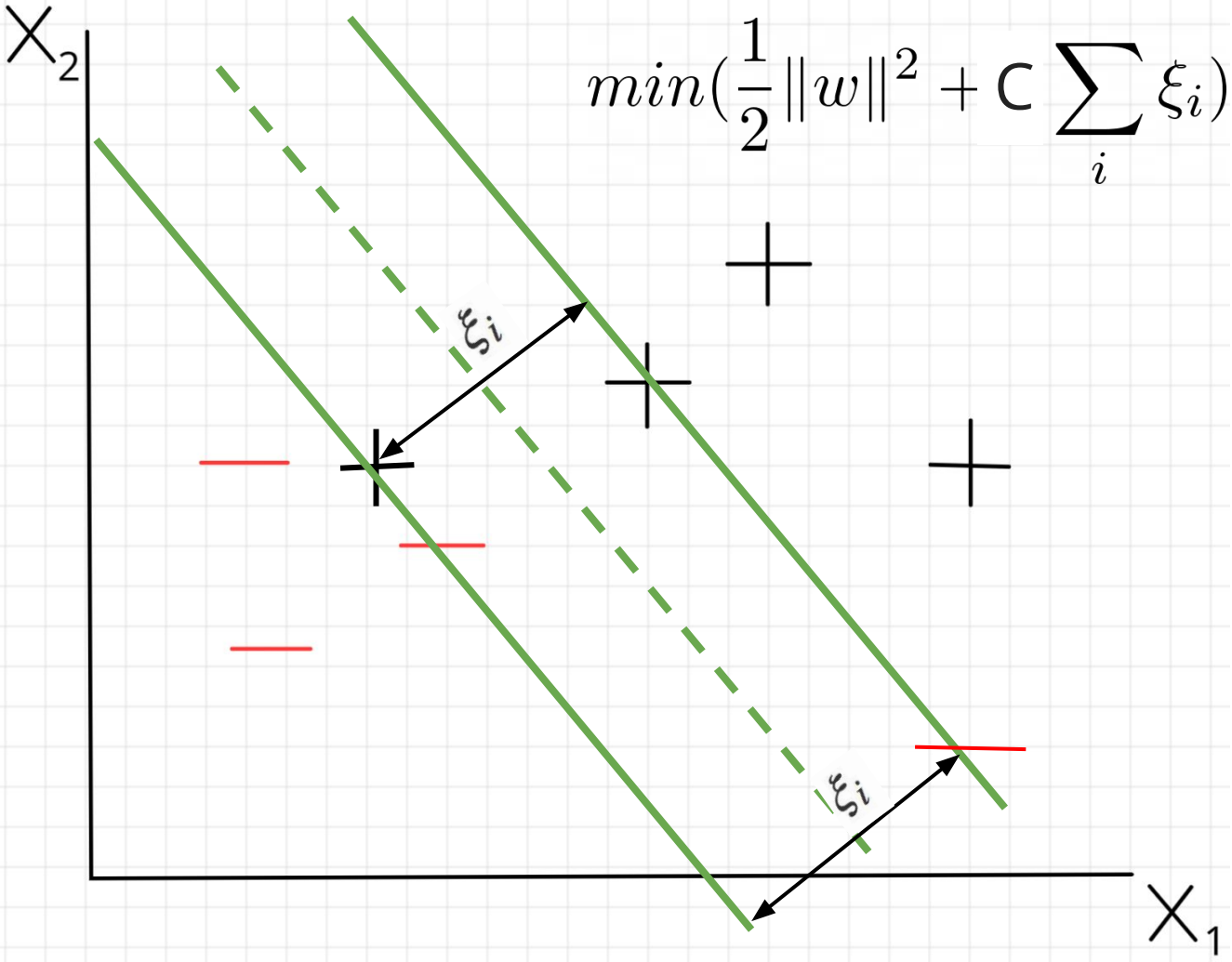
How to find the widest street

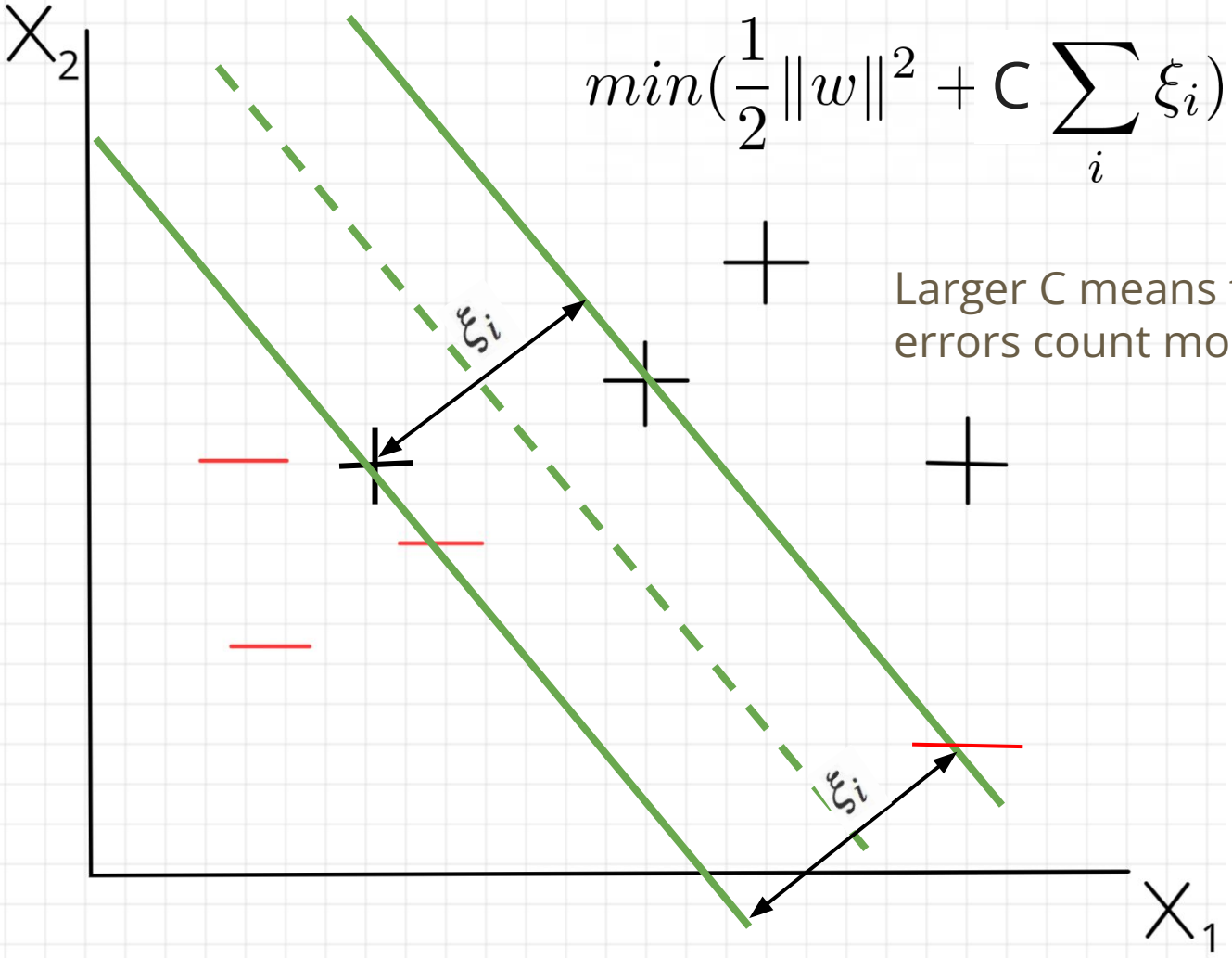
Goal is to maximize the width

$$min(\frac{1}{2}\|w\|^2 + C\sum_i \xi_i)$$

Subject to:

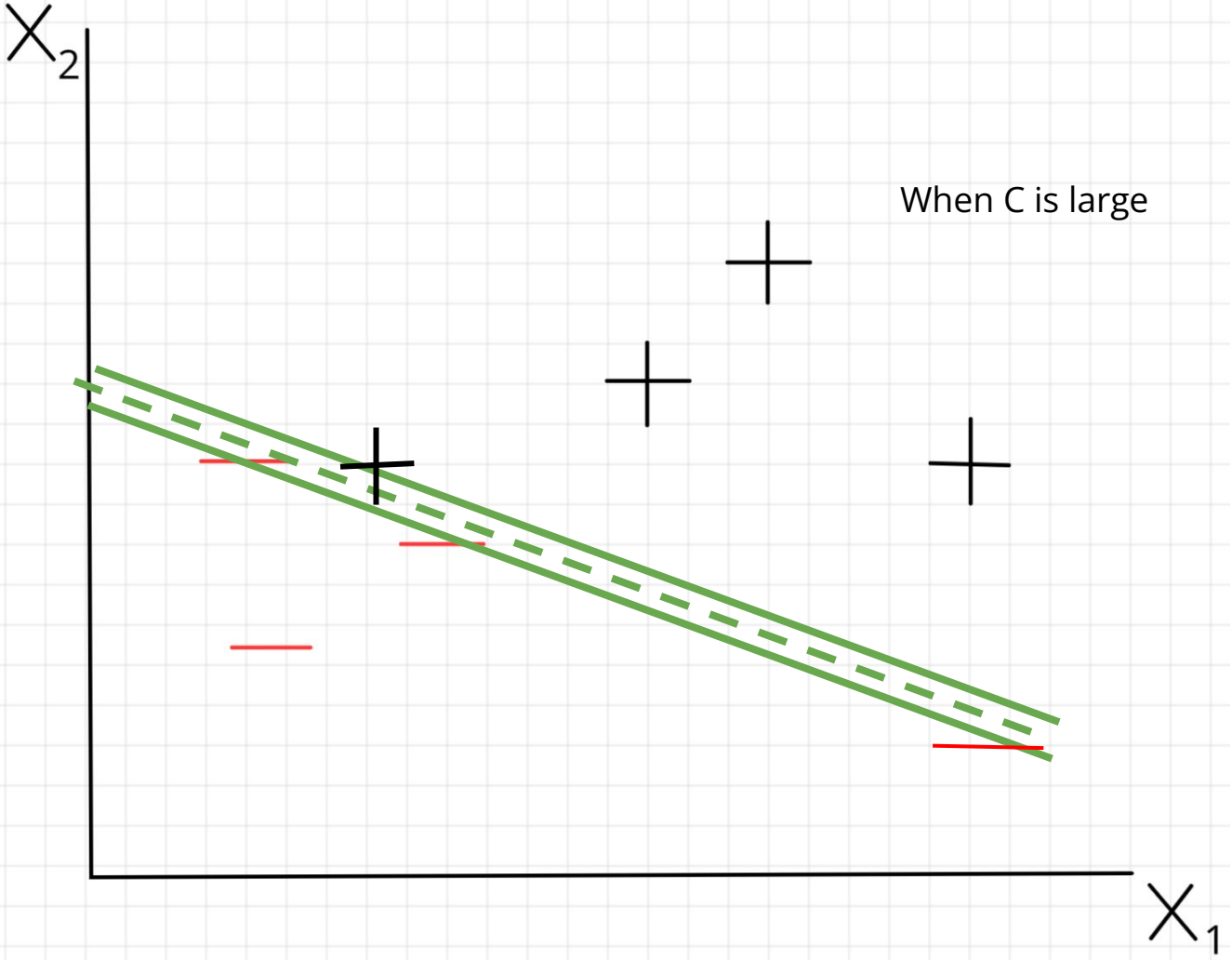
$$y_i(\vec{w}.\vec{x}_i + b) \geq 1 - \xi_i$$

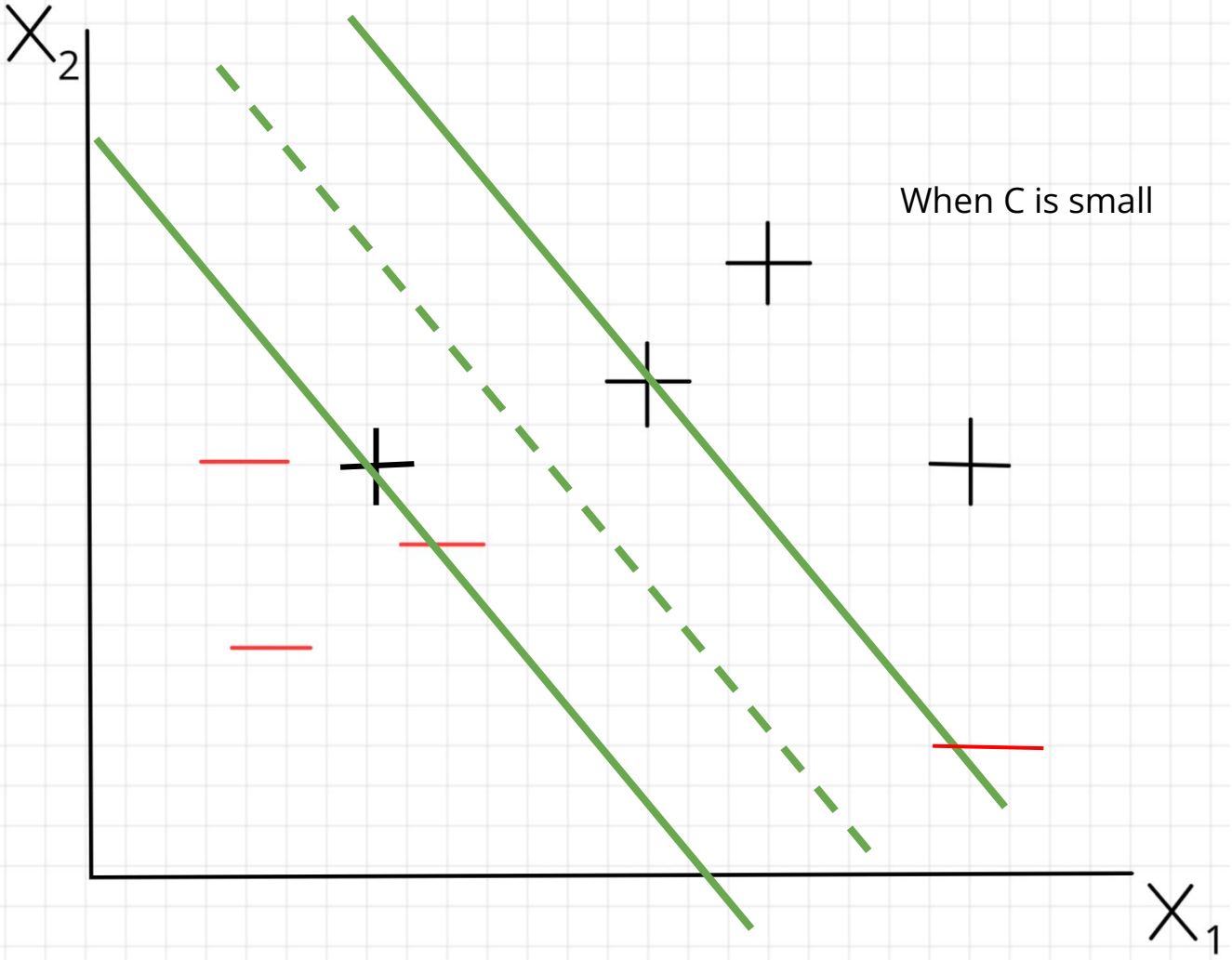

$min(\frac{1}{2}\|w\|^2 + C\sum_i \xi_i)$
X_2
X_1
ξ_i
ξ_i



$$min(\frac{1}{2}\|w\|^2 + C\sum_i \xi_i)$$

Larger C means these errors count more


X2
When C is large
X1

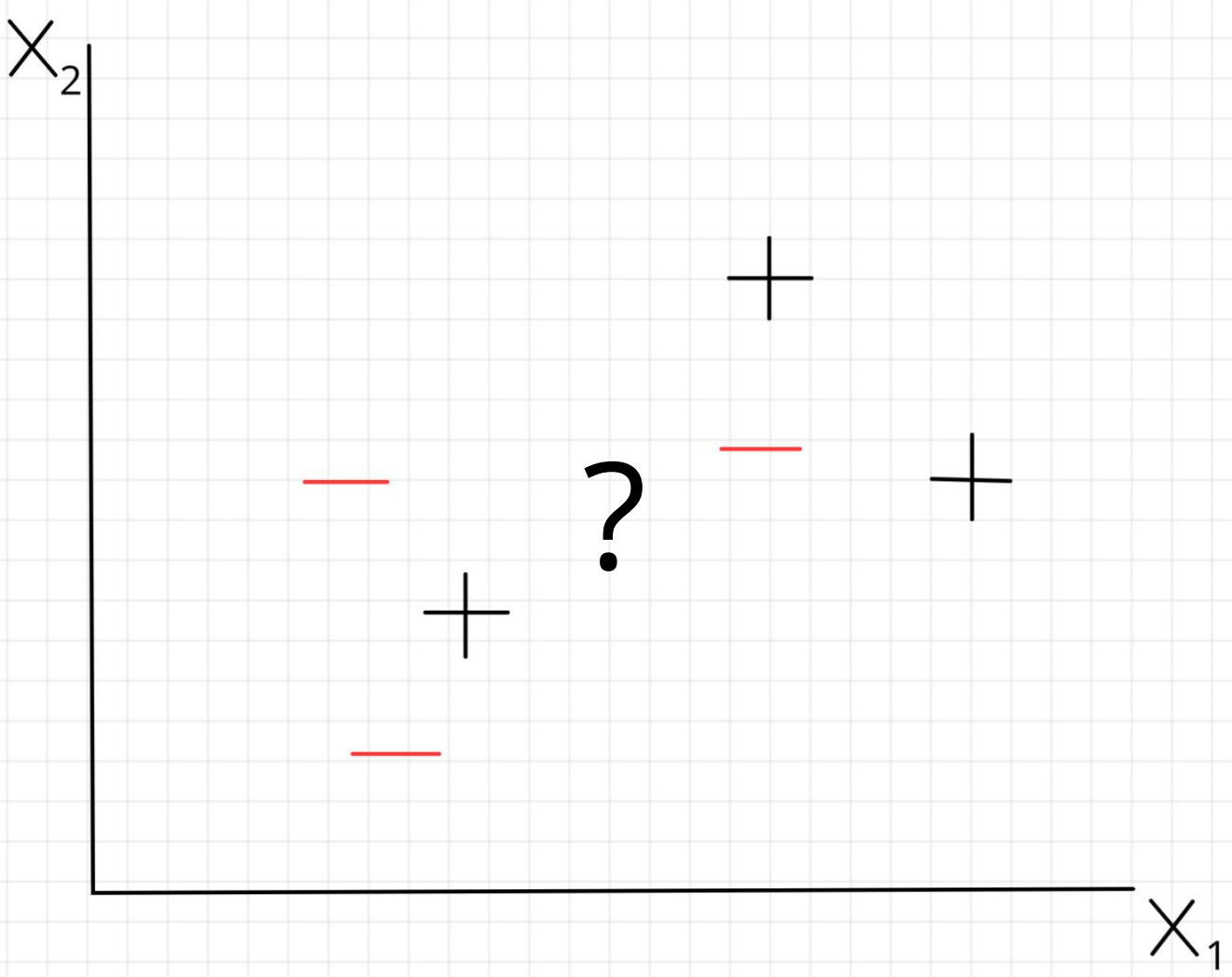

X2
When C is small
X1

Best practices

If C is too large, model might get too complex to try to avoid misclassification -> Overfitting

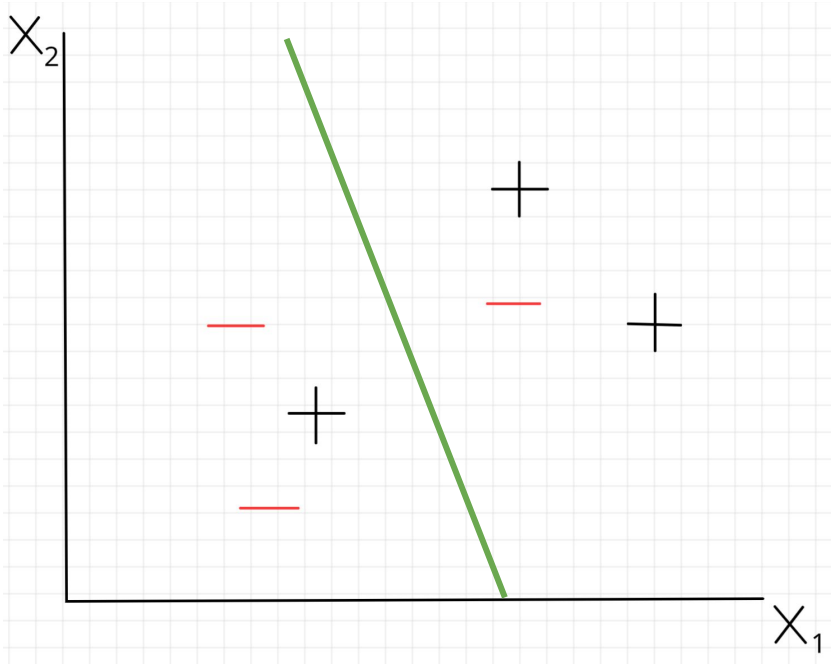
If C is too small, model might be too simple -> underfitting

What if there is no line?

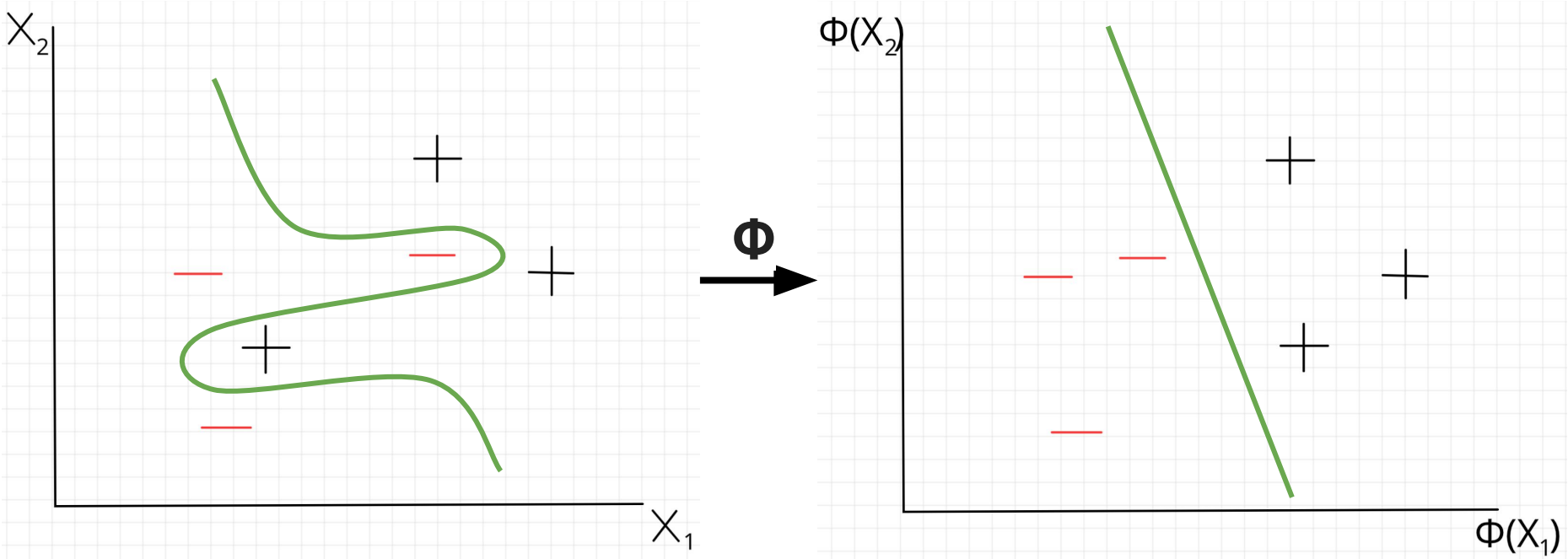

X2
?
X1

Option 1: Soft Margins

Can allow for some points in the dataset to be misclassified. (i.e. tune C)



Option 2: Change perspective



But how to find Φ?

Turns out we don't need to find or define a transformation Φ!

Recall:

$$\sum_i \alpha_i < x_i, x > + b \geq 0 \quad \text{then} +$$

But how to find Φ?

Turns out we don't need to find or define a transformation Φ!

Recall:

$$\sum_i \alpha_i < x_i, x > + b \geq 0 \quad \text{then} +$$

But how to find Φ?

Turns out we don't need to find or define a transformation Φ!

we only need to define

$$K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$$

Called a Kernel function. This is often referred to as the "kernel trick".

$$\sum_i \alpha_i K(x_i, x) + b \geq 0 \quad \text{then} +$$

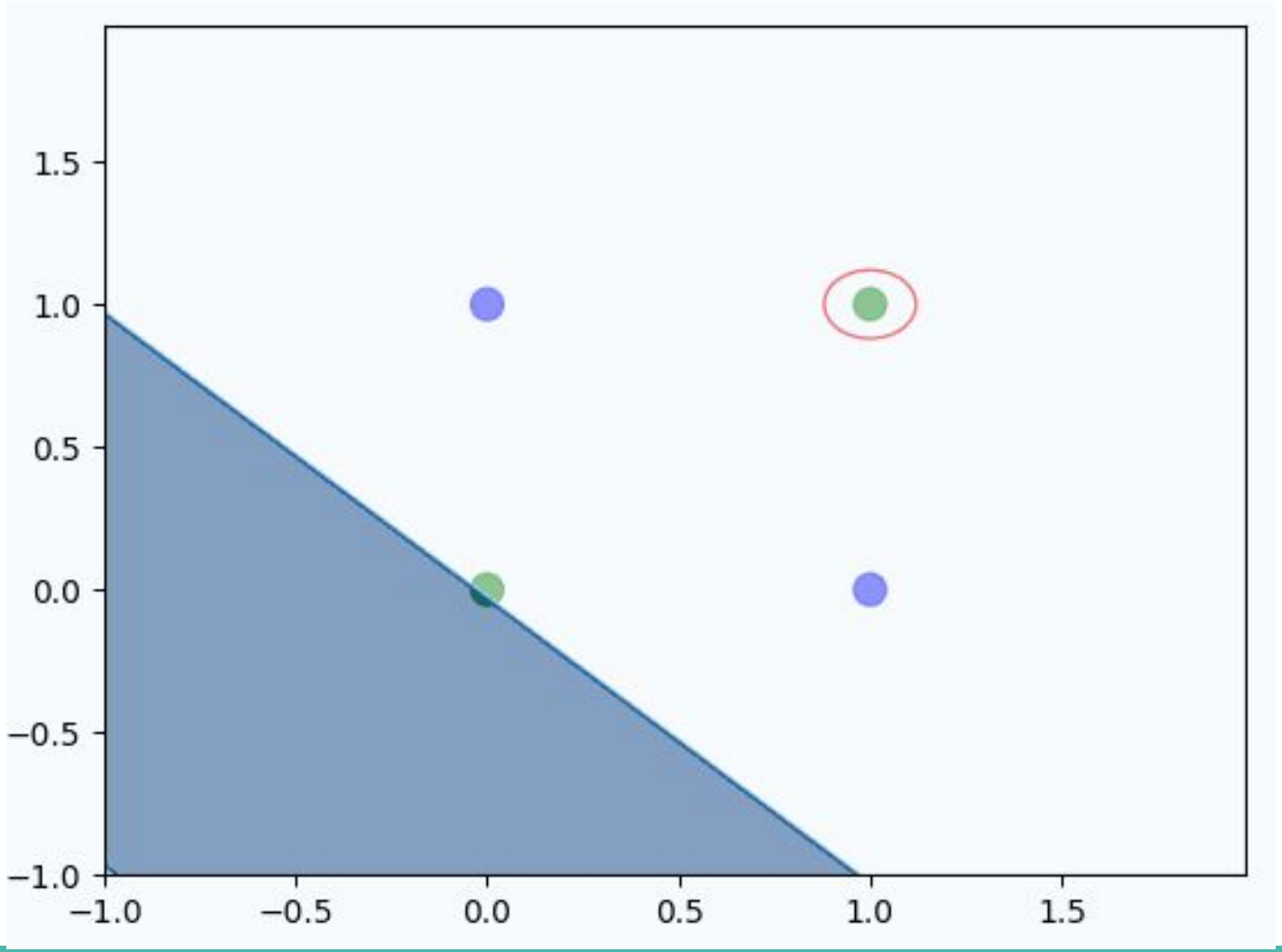
Example Kernel Functions

Polynomial Kernel

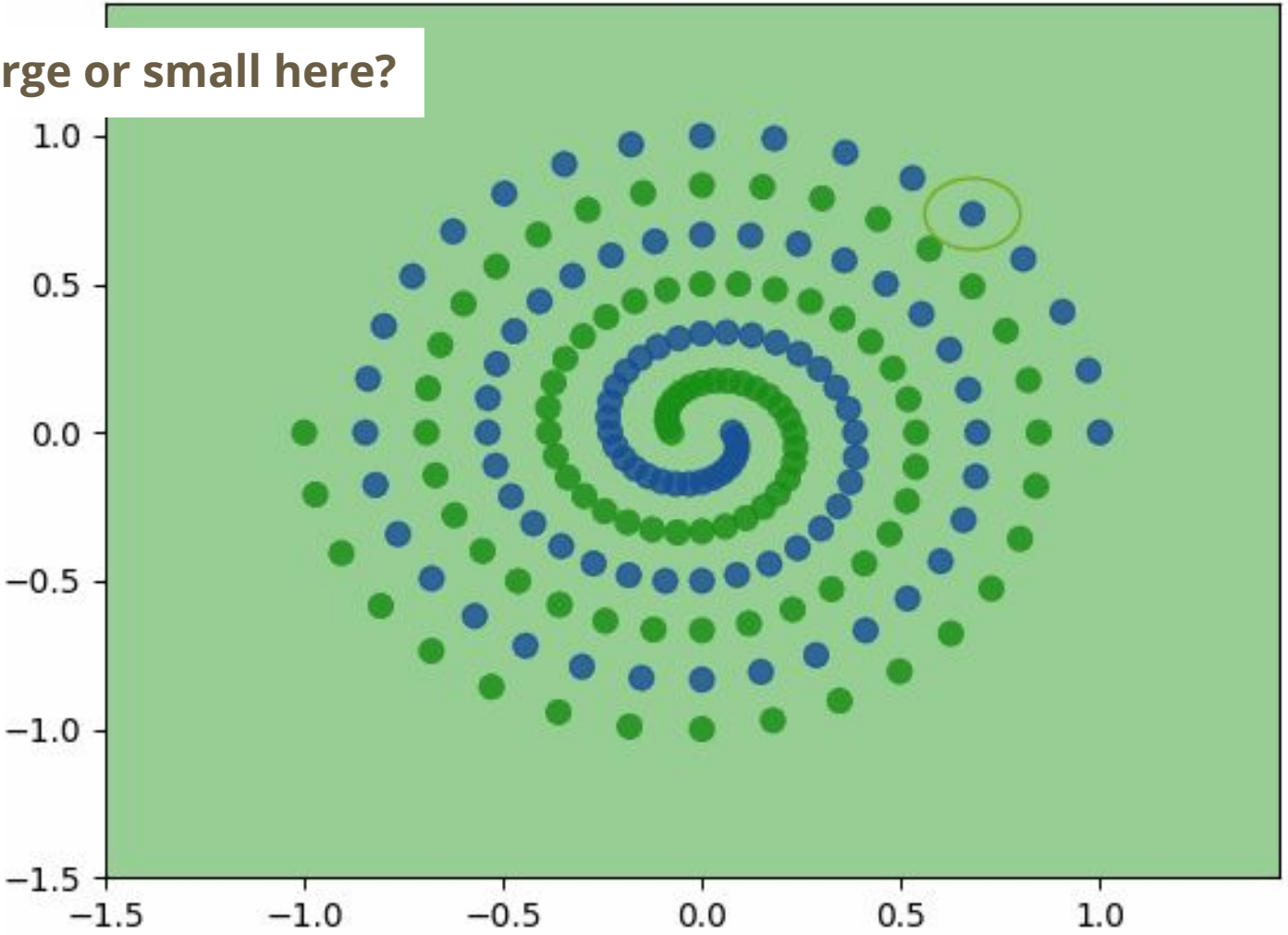
$$K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + 1)^n$$

Radial Basis Function Kernel

$$K(\vec{x}_i, \vec{x}_j) = e^{\frac{\|\vec{x}_i - \vec{x}_j\|}{\sigma}}$$


1.5
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
1.5

Q: is C large or small here?



Kernel Function (intuition)

- The inner product of a space describes how close / similar points are
- Kernel Functions allow for specifying the closeness / similarity of points in a hypothetical transformed space
- The hope is that with that new notion of closeness, points in the dataset are linearly separable.

More info

https://medium.com/mlearning-ai/support-vector-machines-16241417ee6d