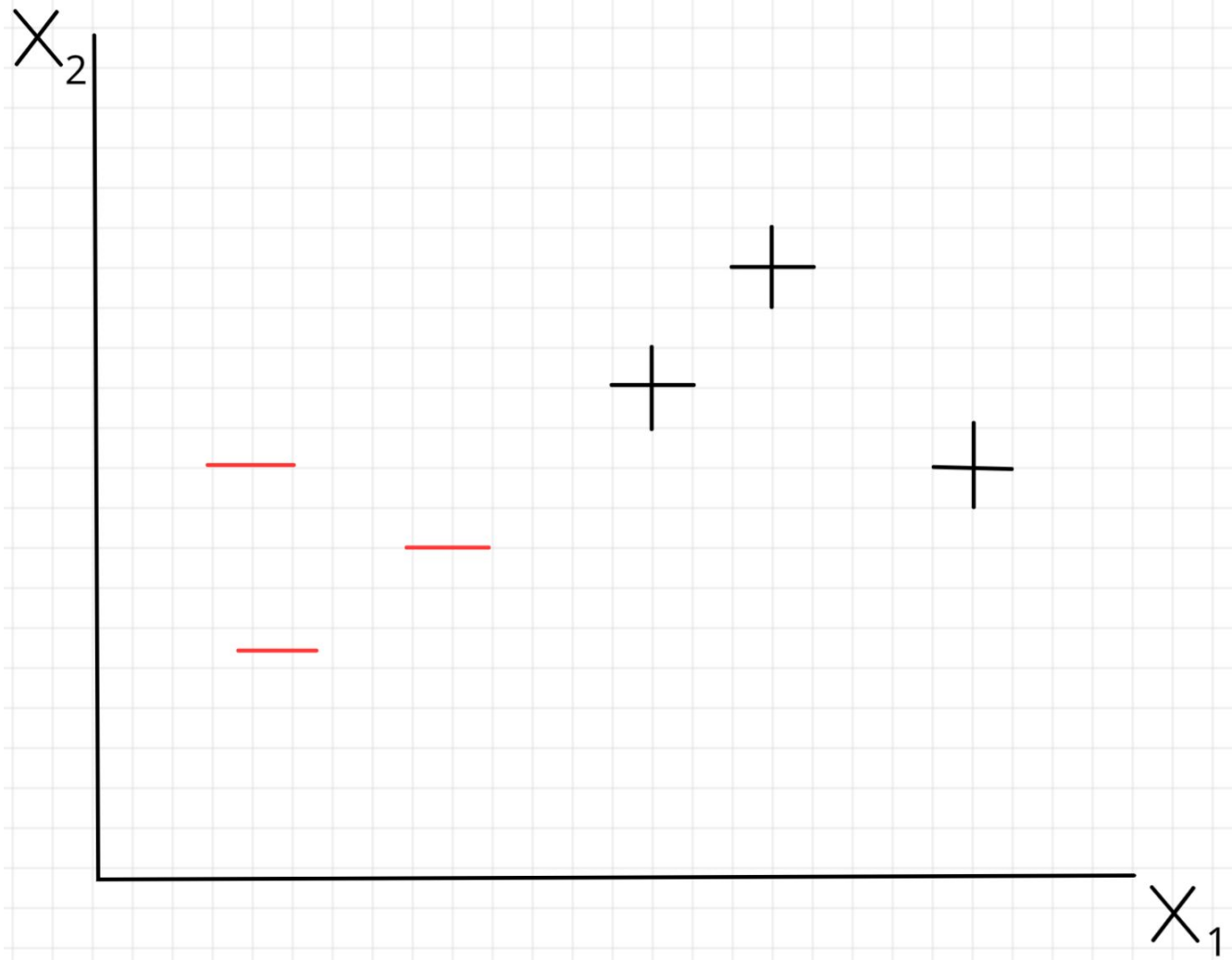
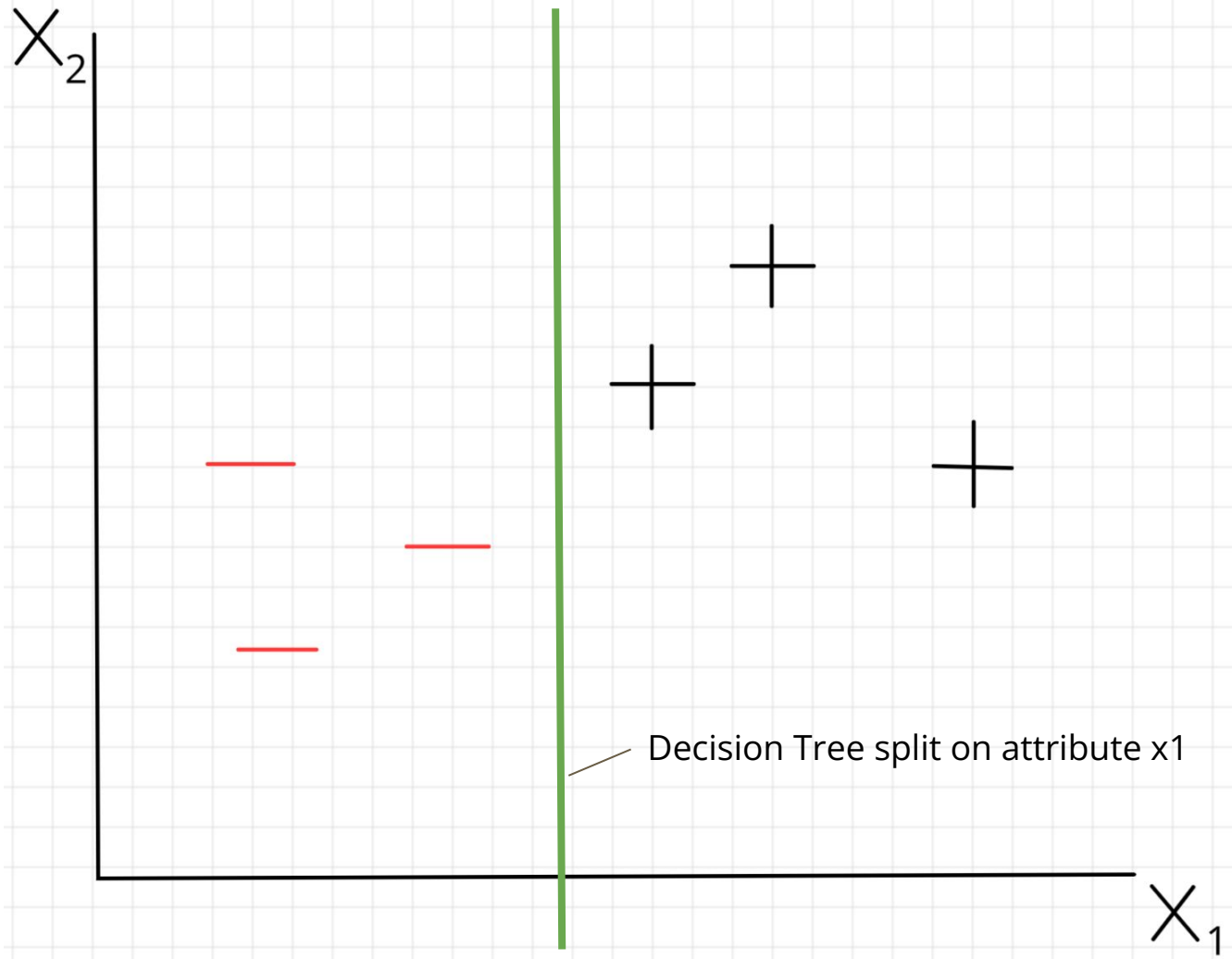
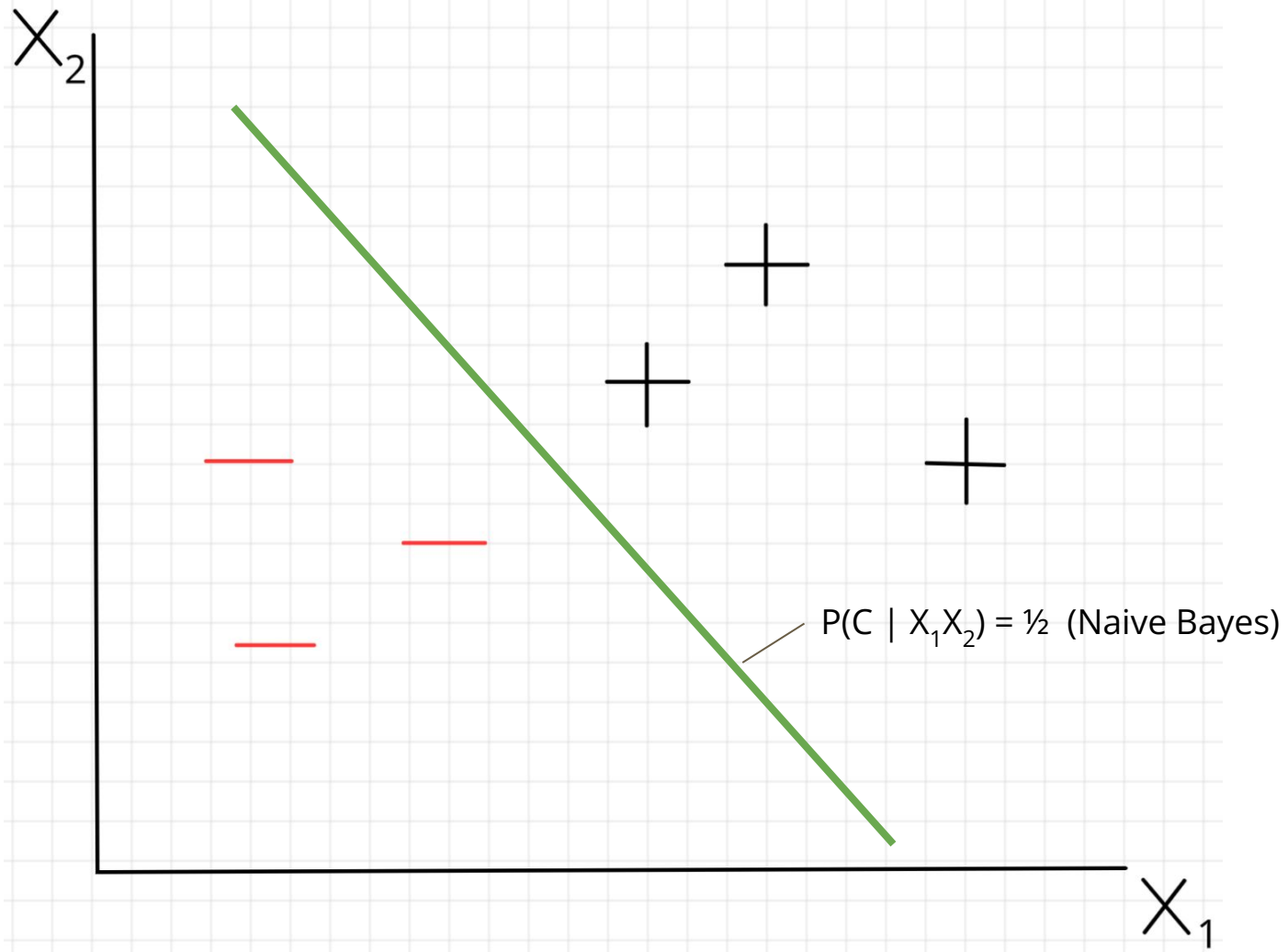

Support Vector Machines

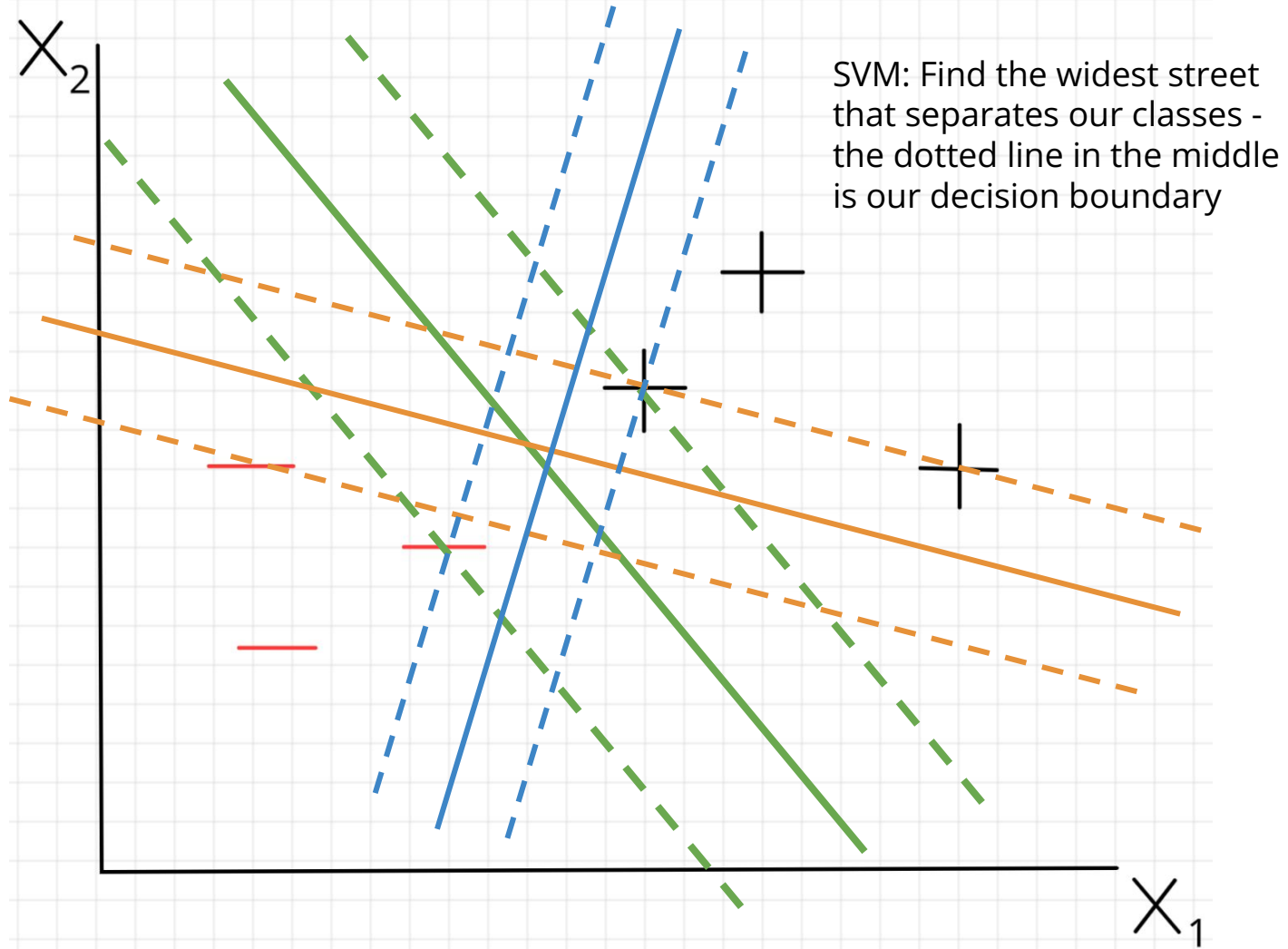
— Boston University CS 506 - Lance Galletti —

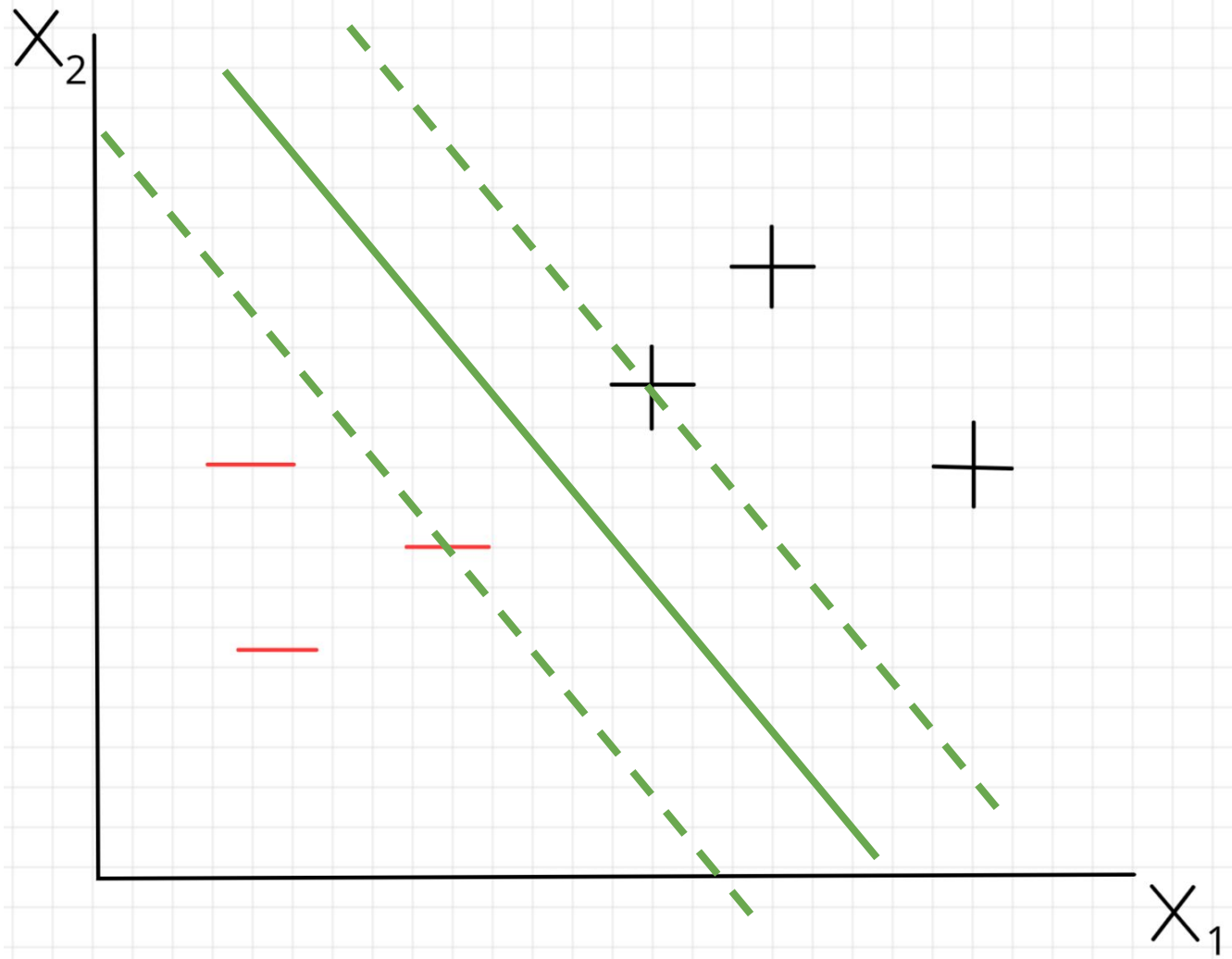


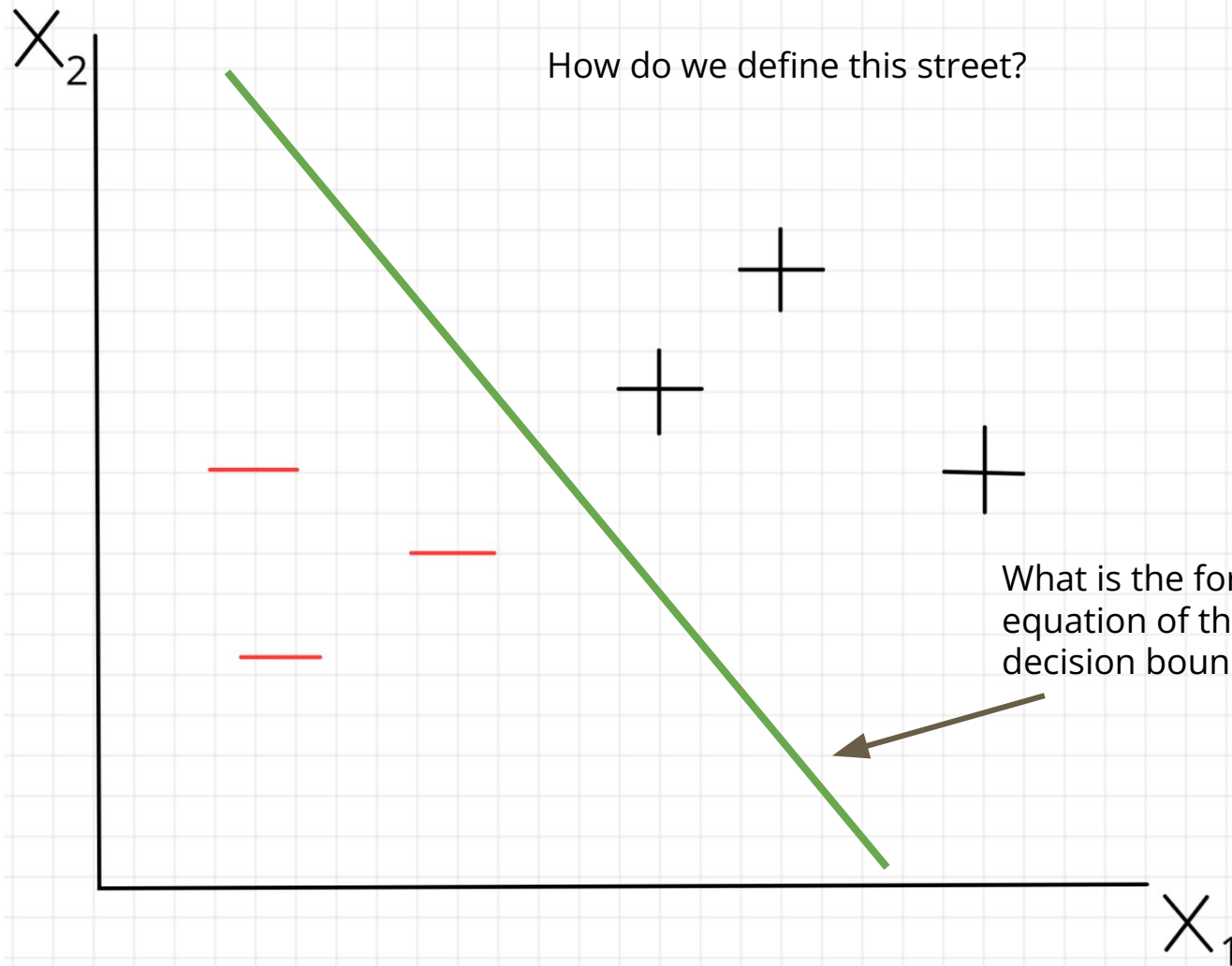


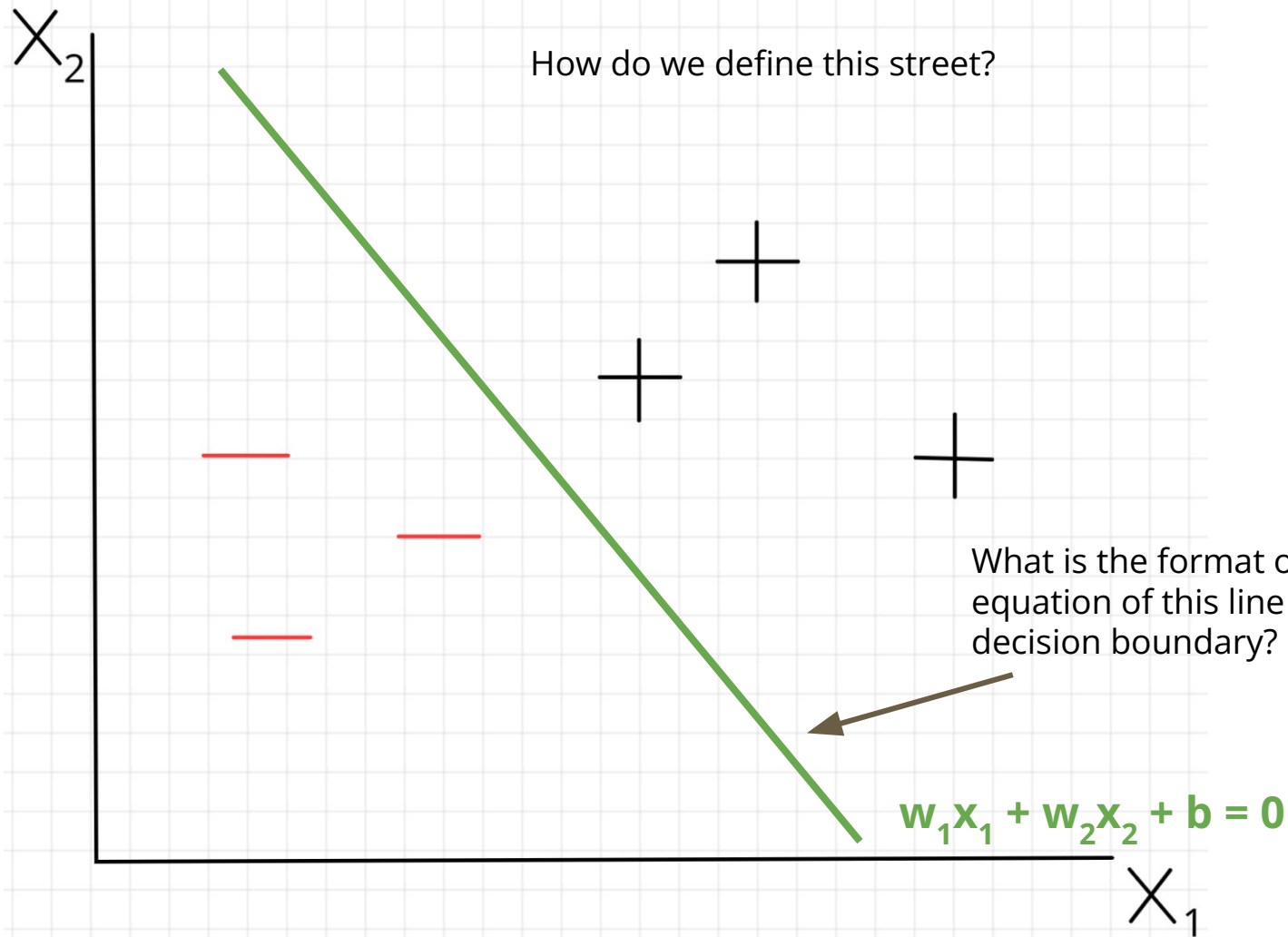


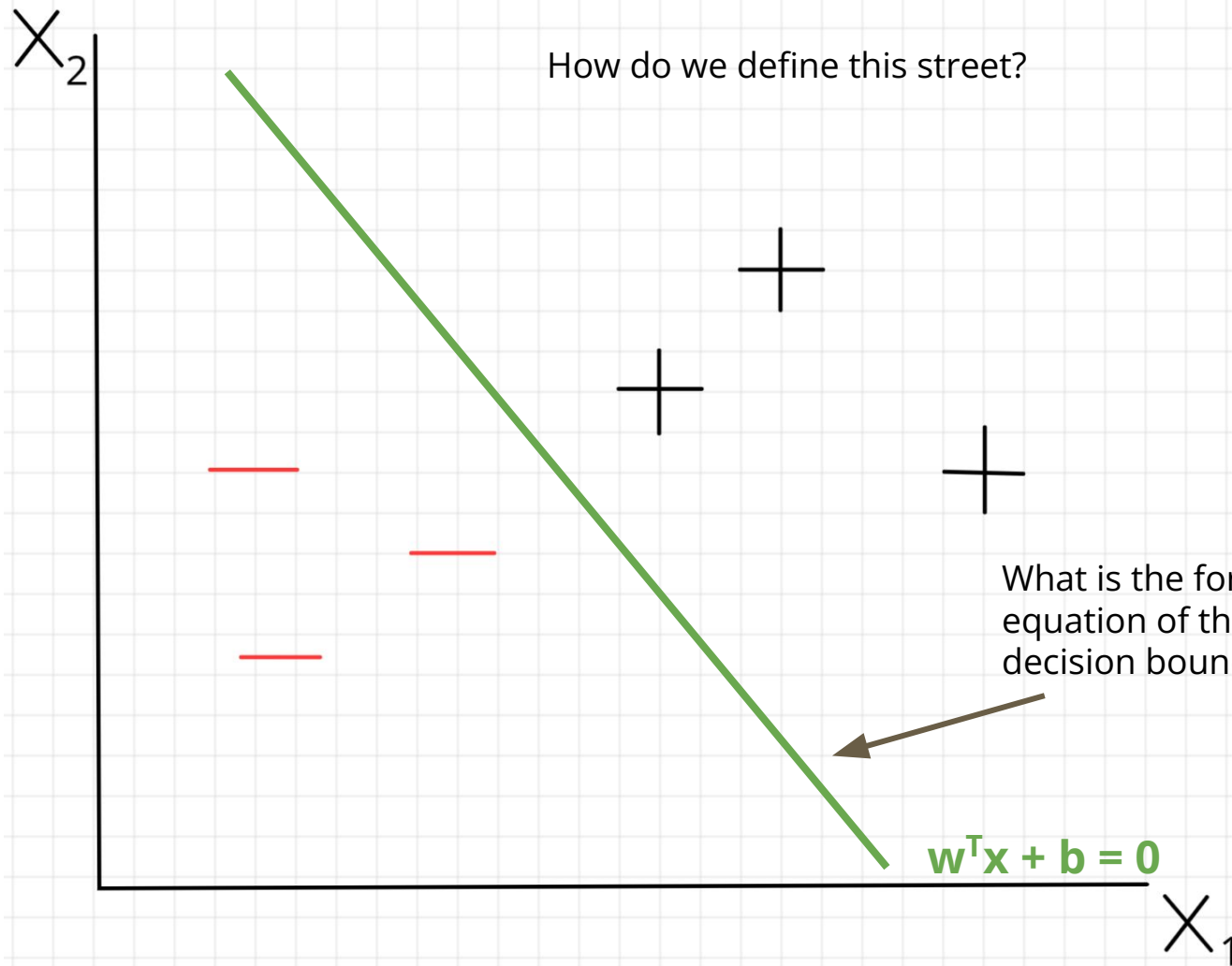


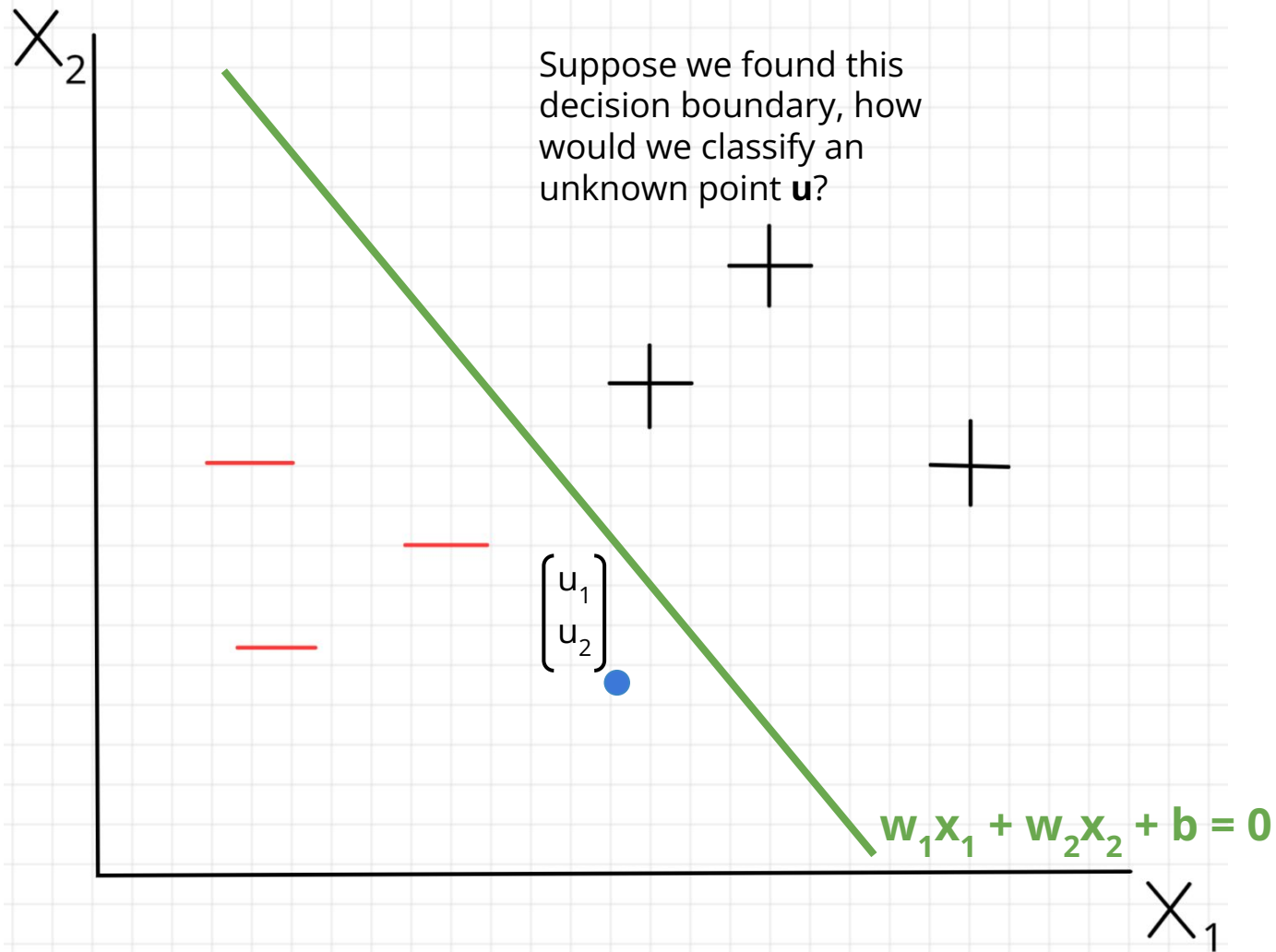


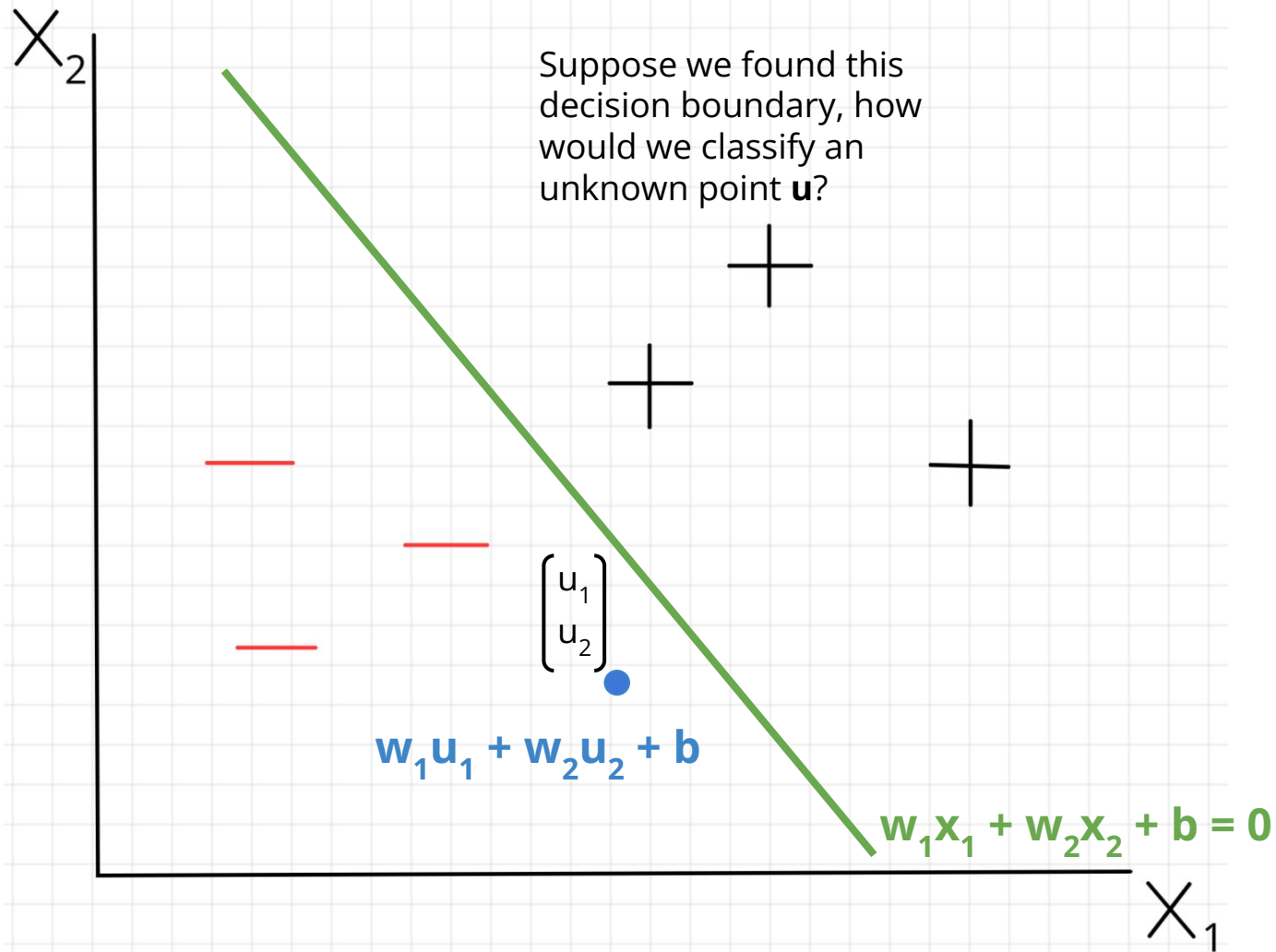


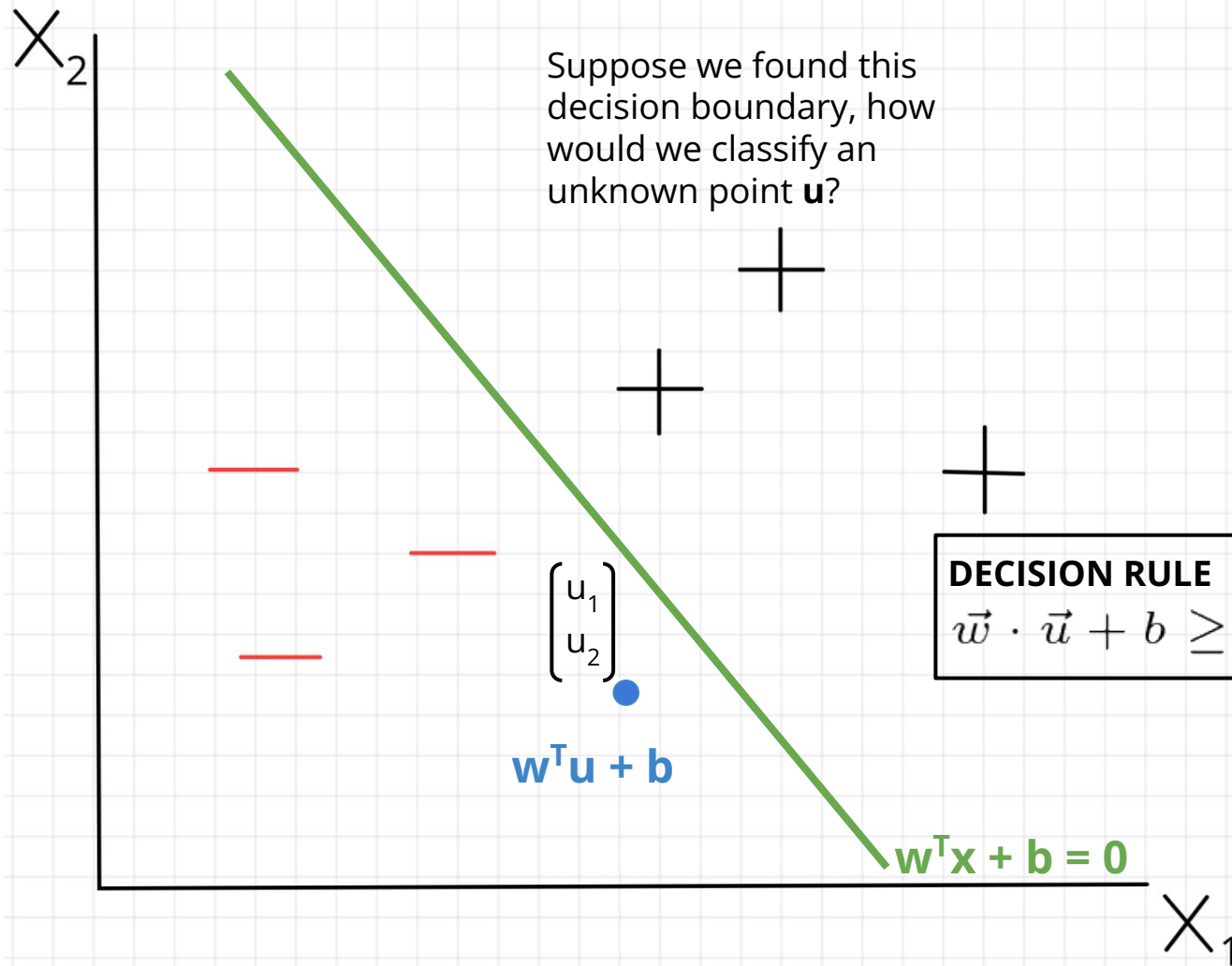


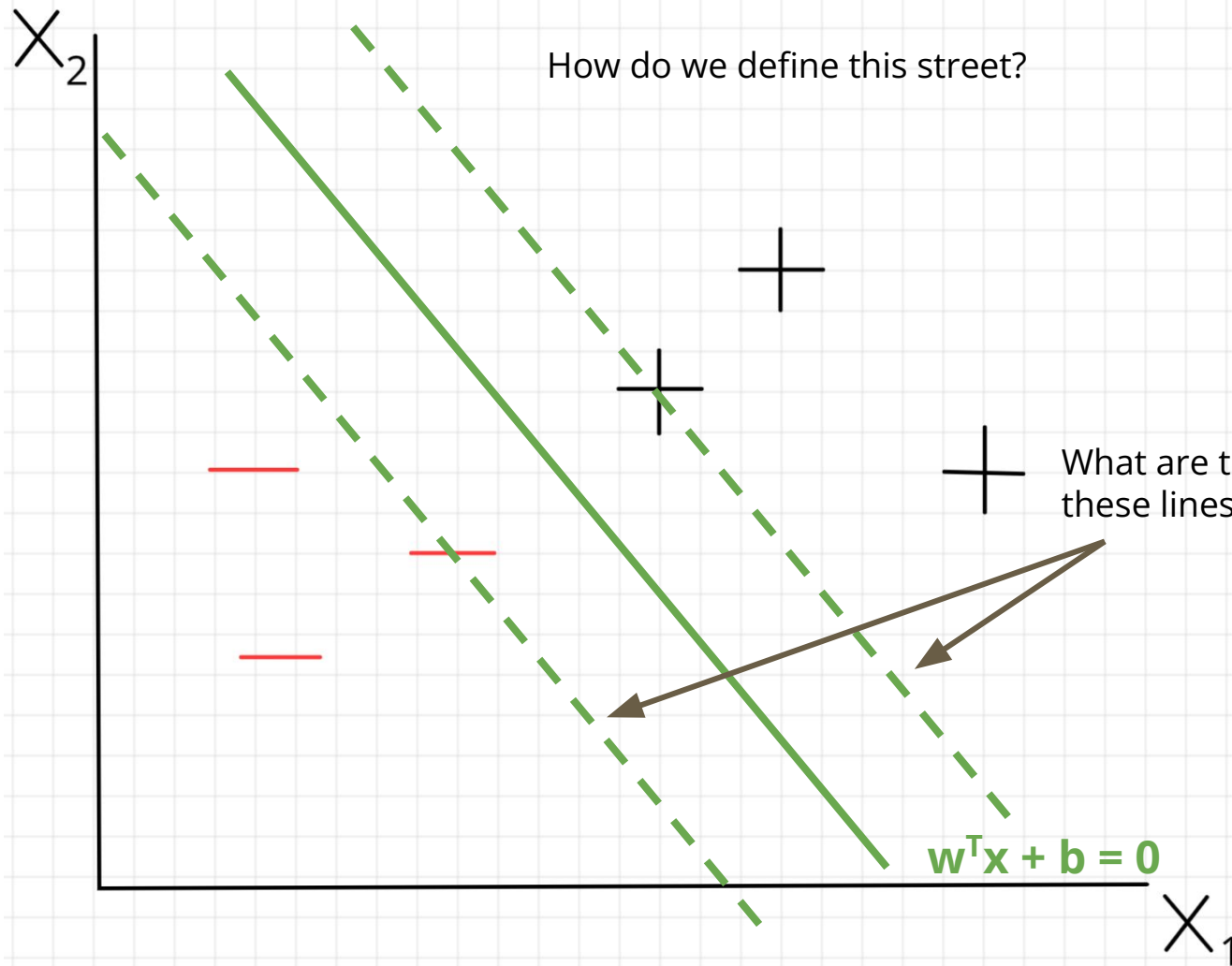


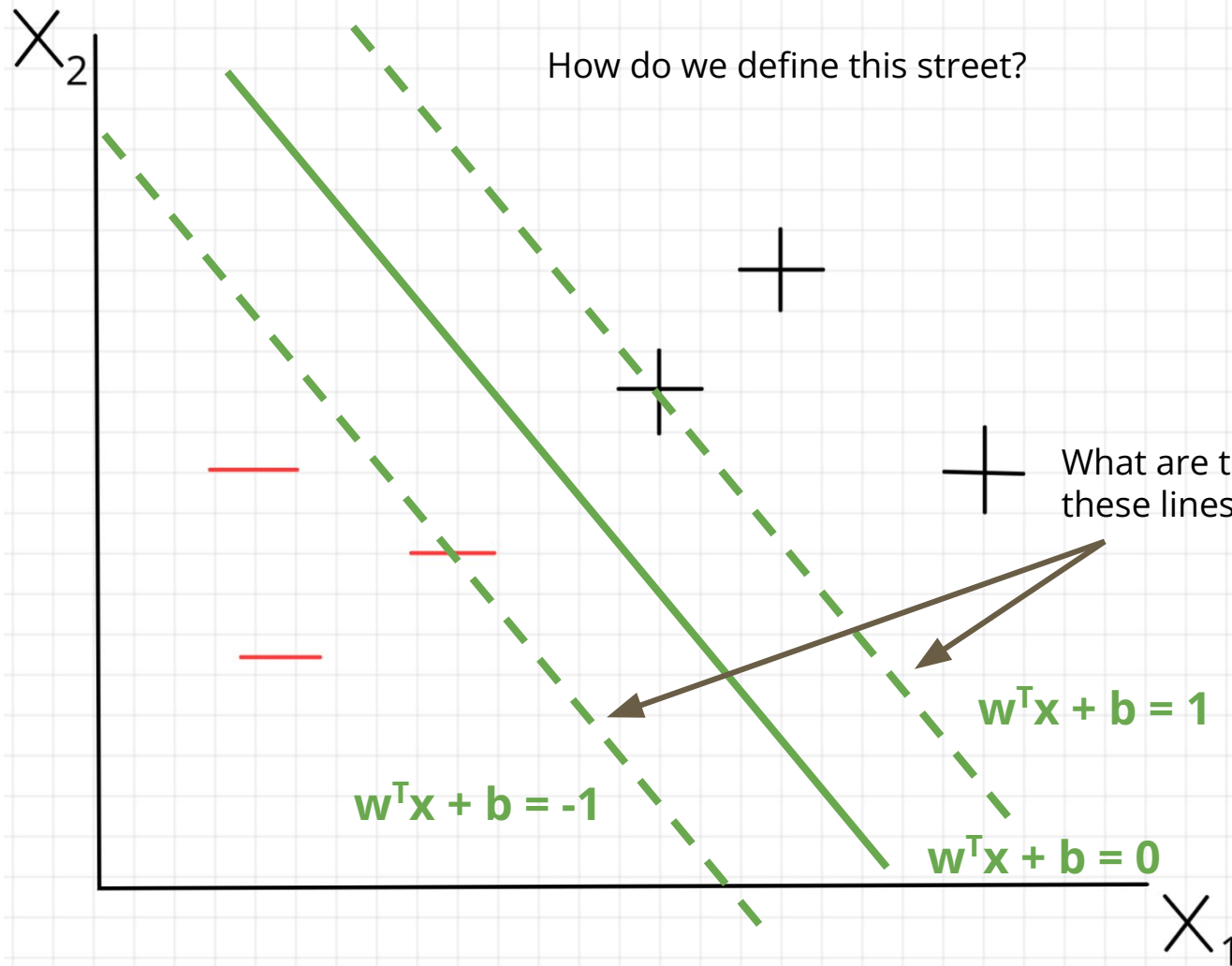


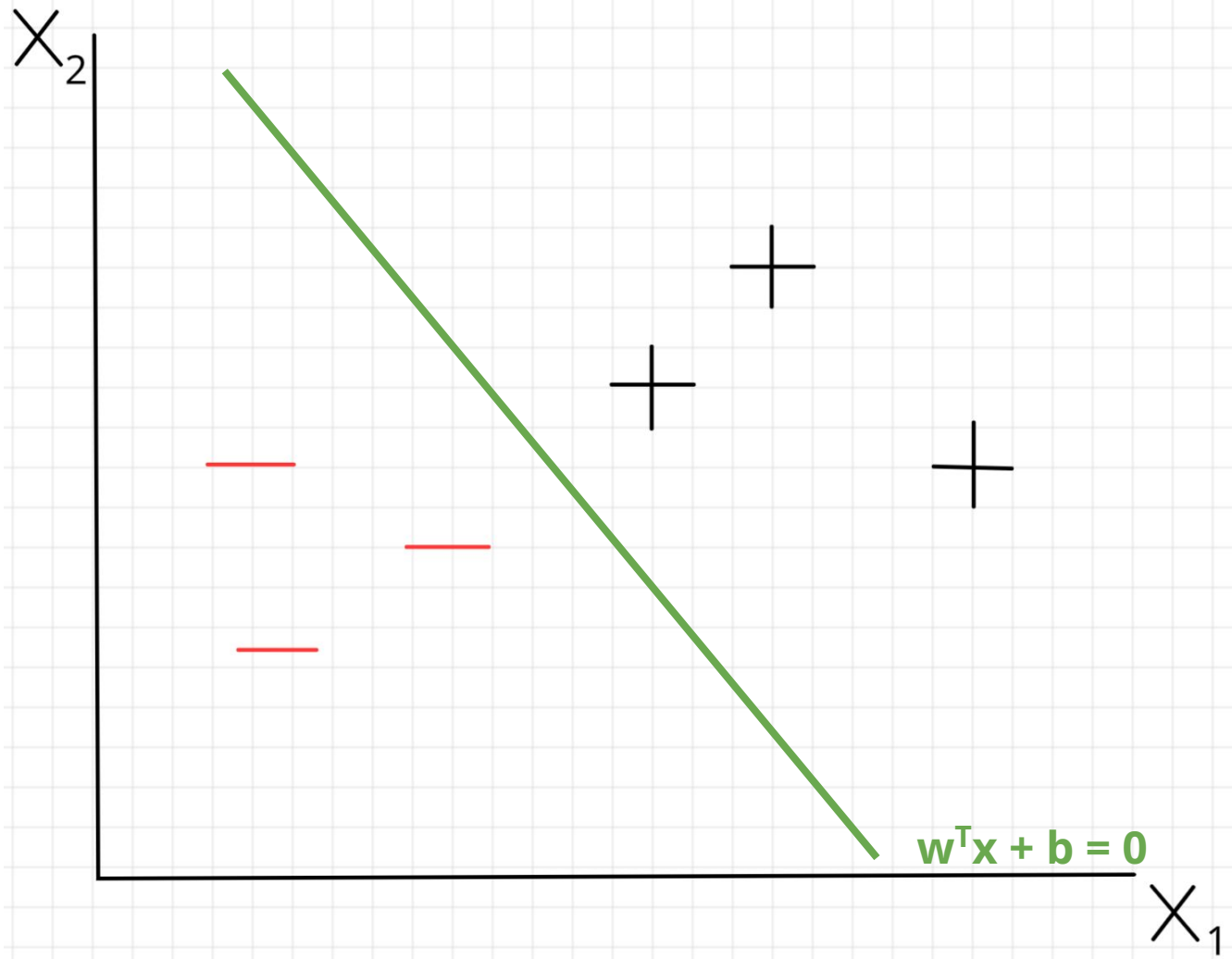


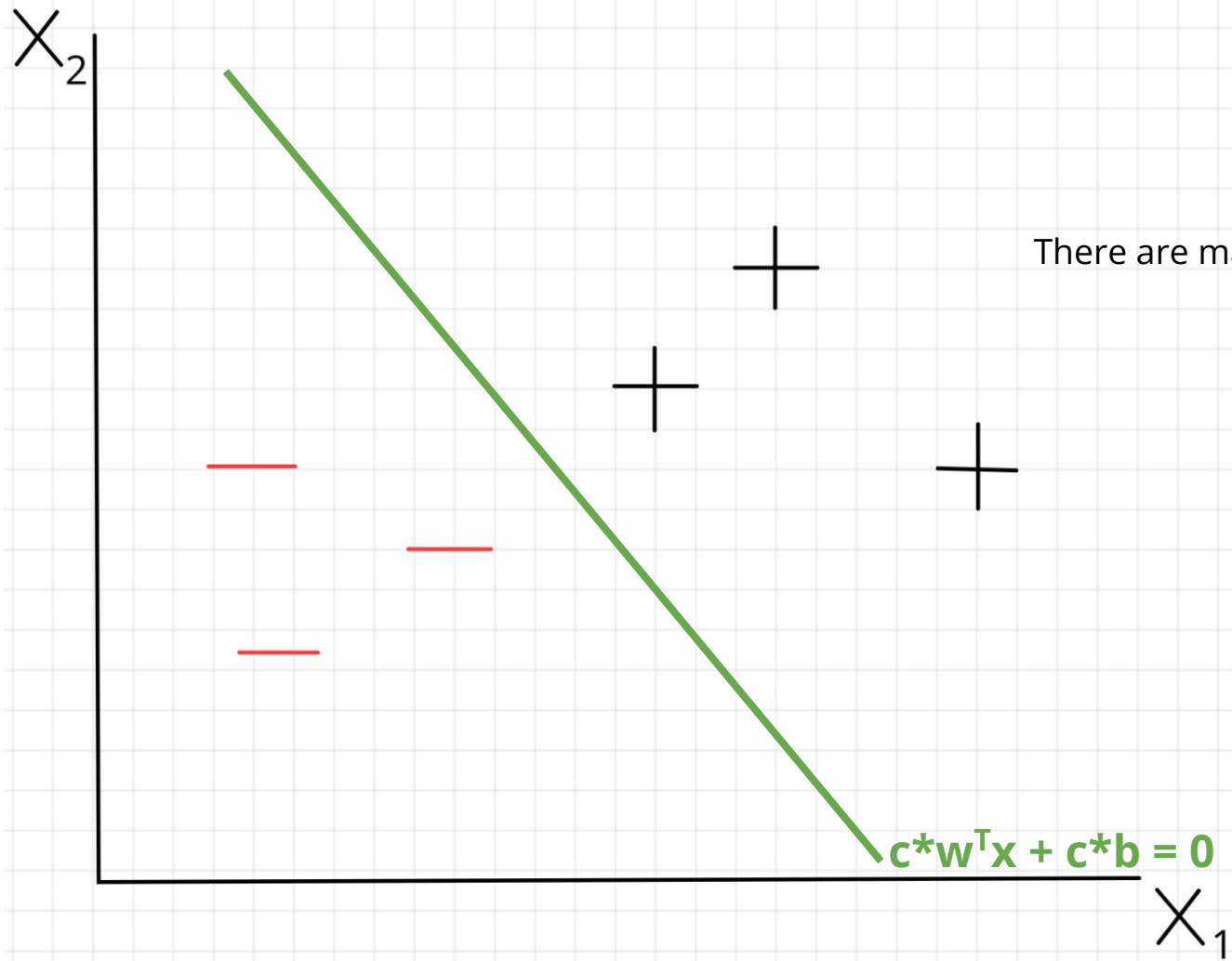




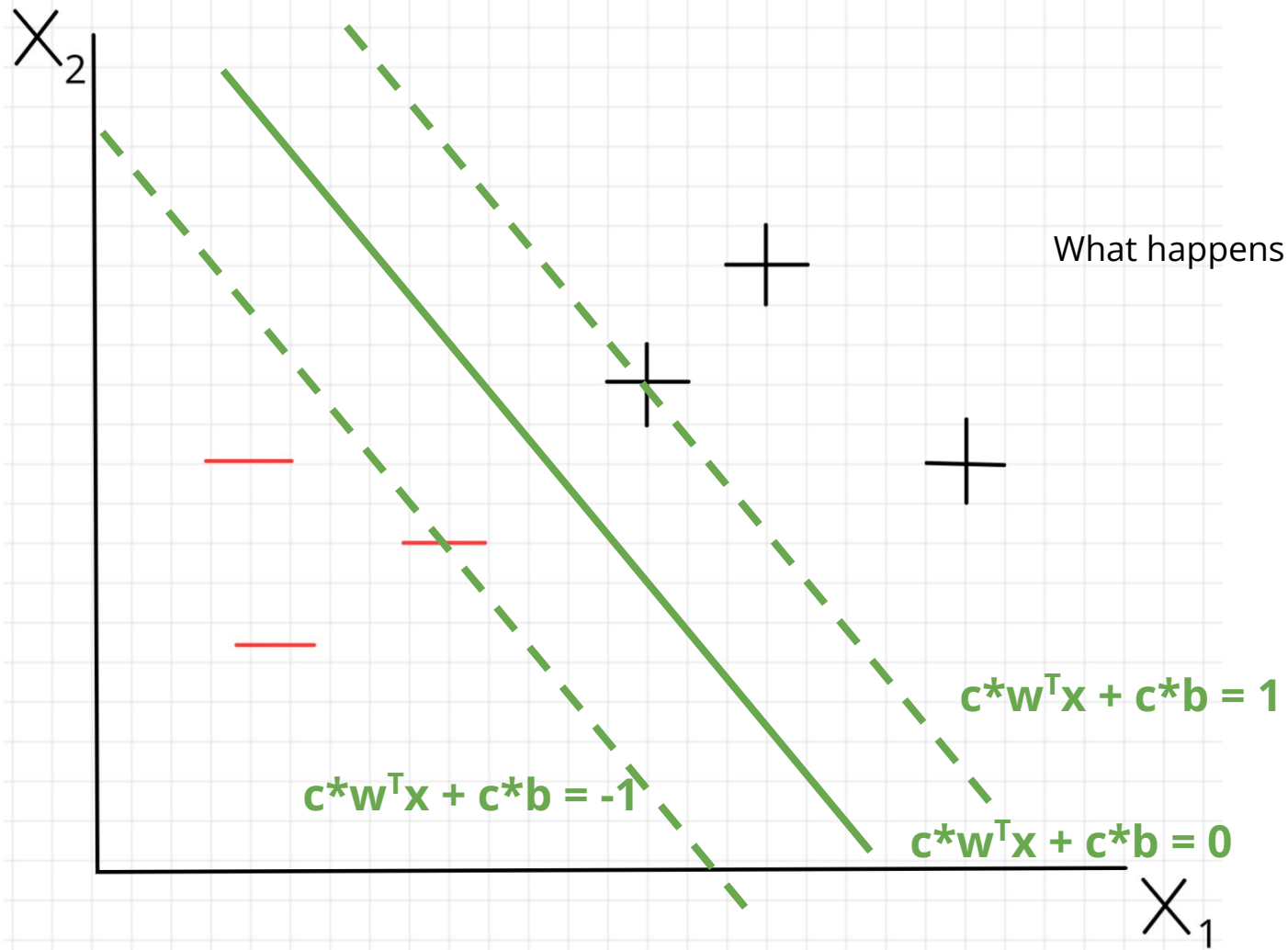


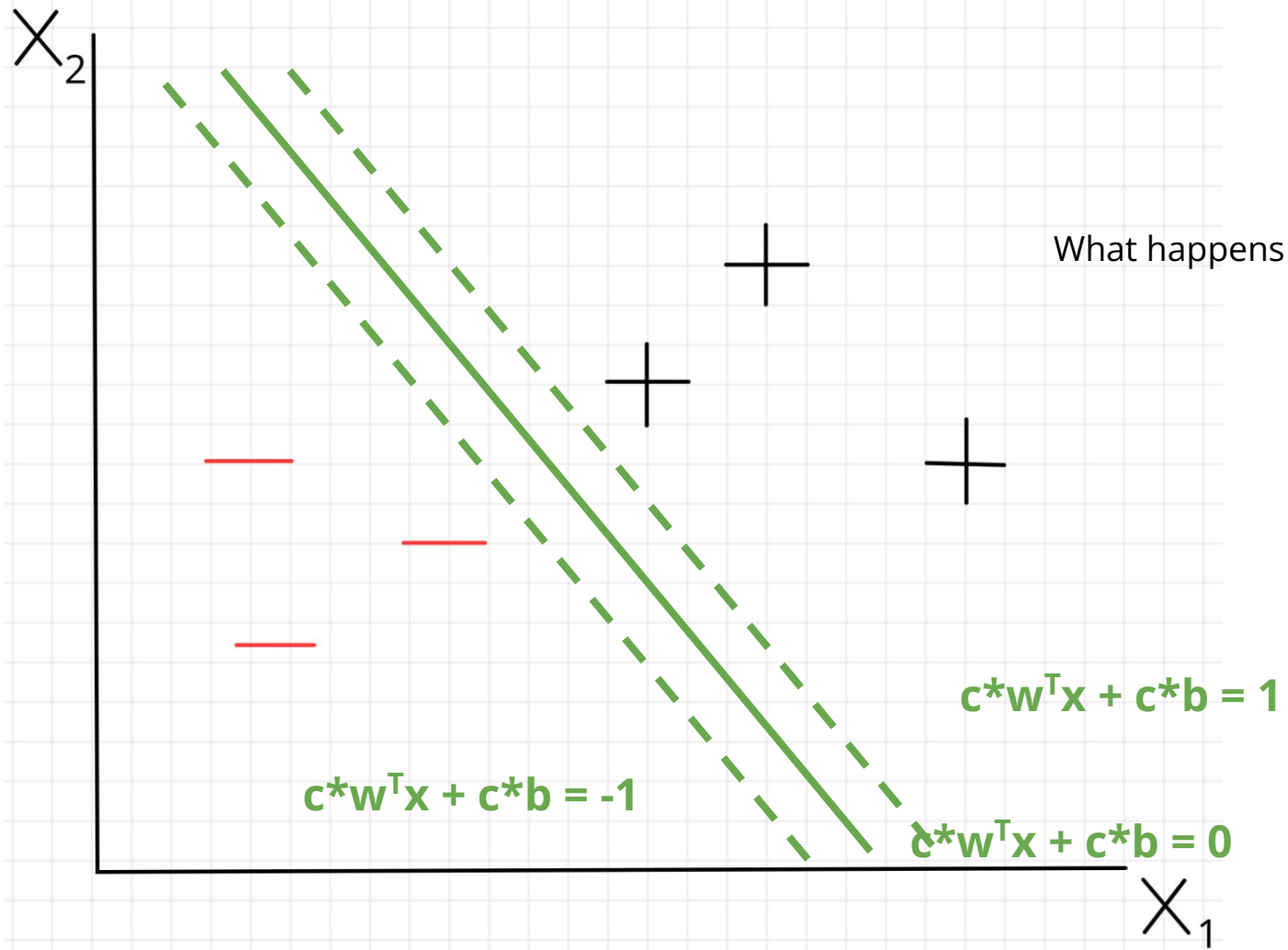


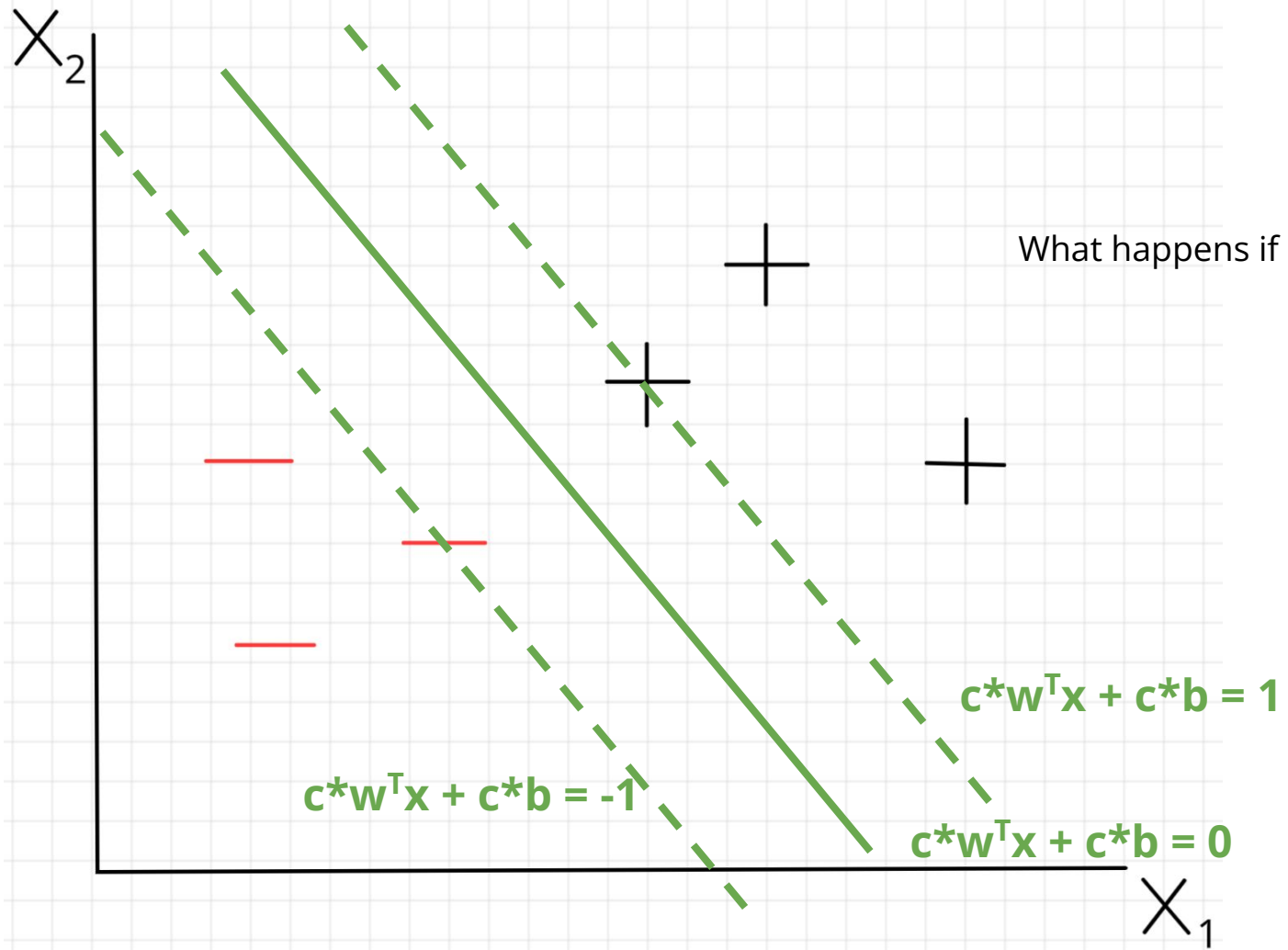


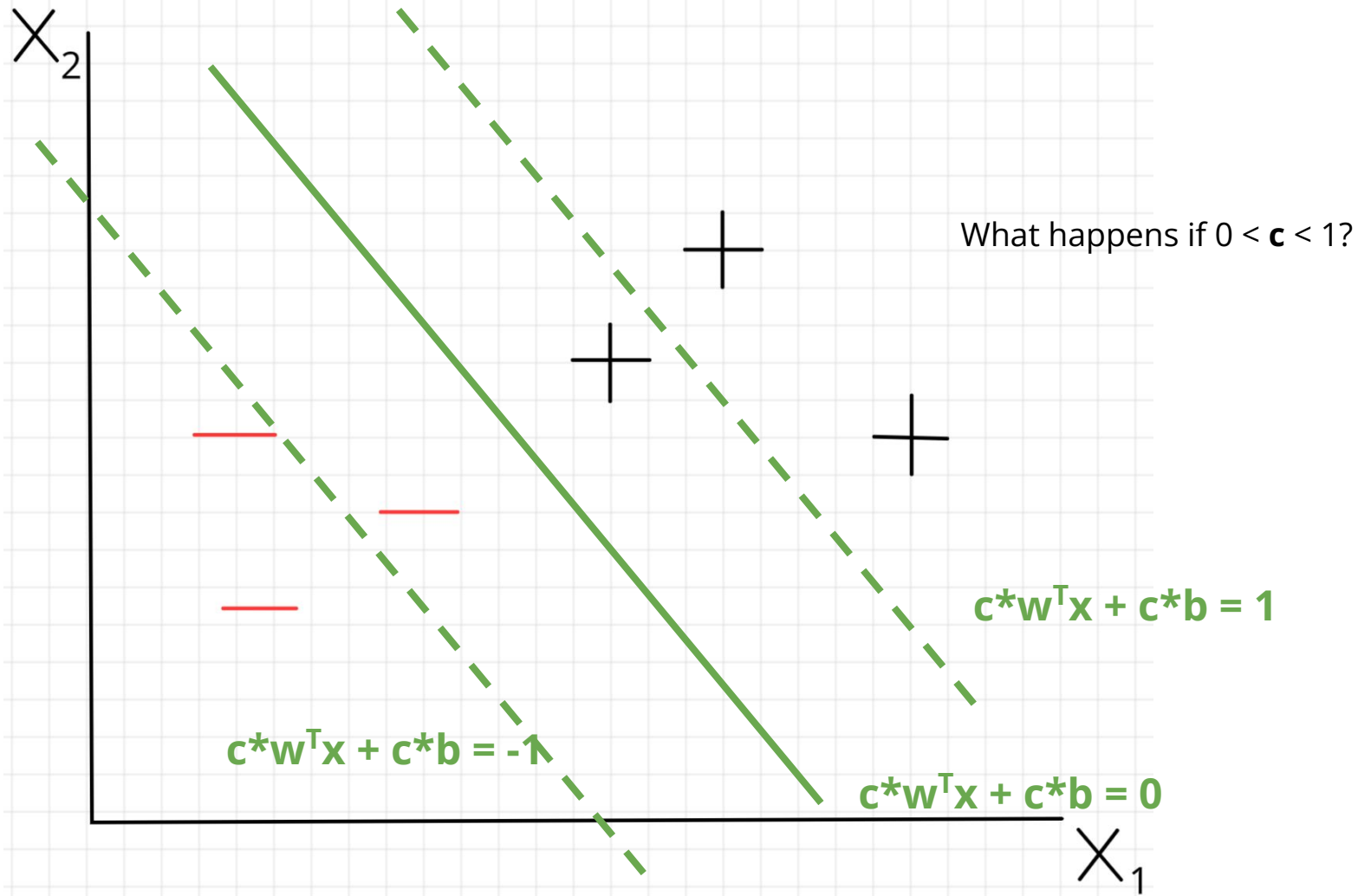


There are many **w**'s and **b**'s









Find the widest street

Means finding the smallest **w** and **b** that separates the classes

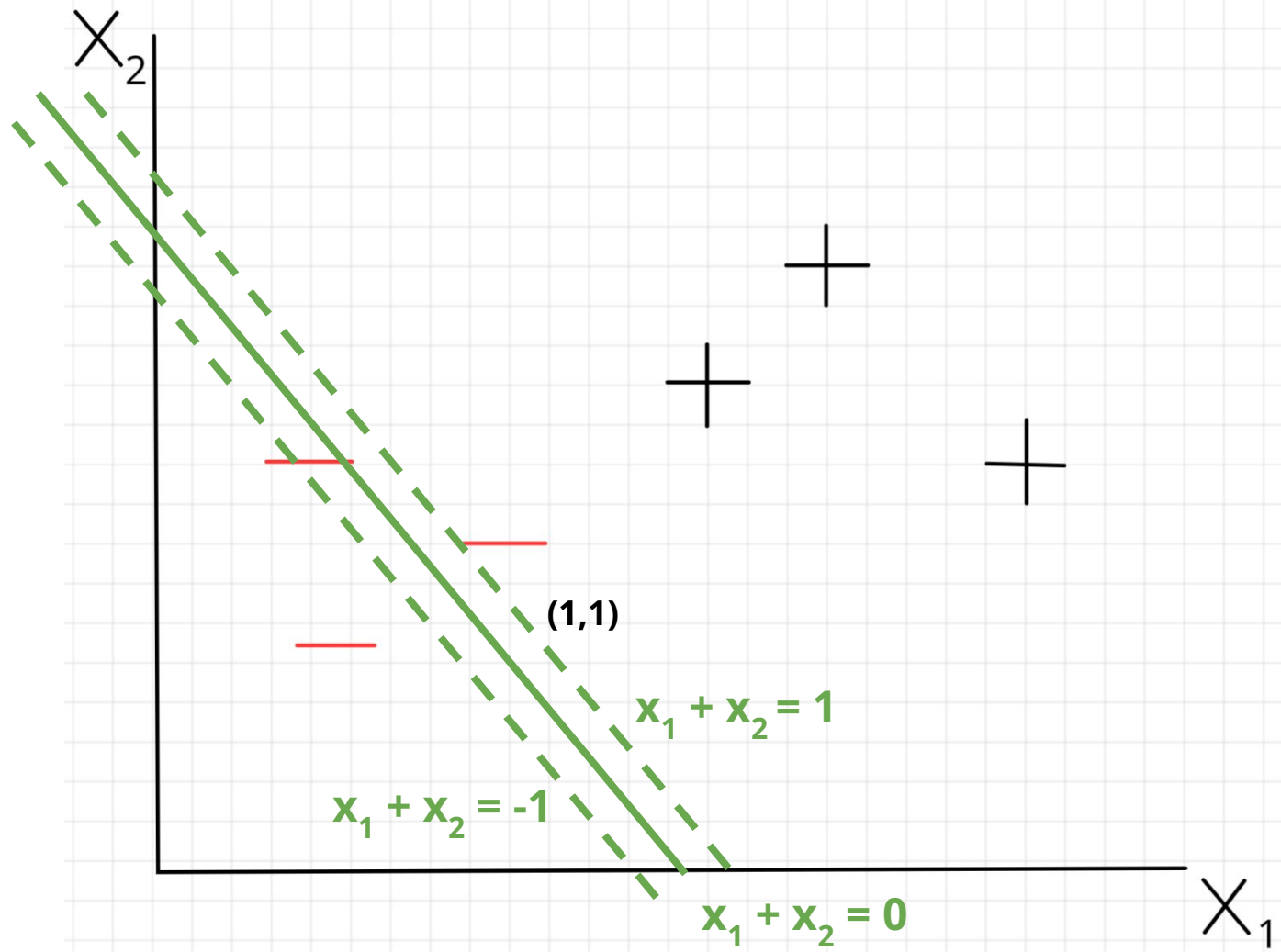
$$y_i = \begin{cases} +1 & \text{if } x_i \text{ is a } + \text{ sample} \\ -1 & \text{if } x_i \text{ is a } - \text{ sample} \end{cases}$$

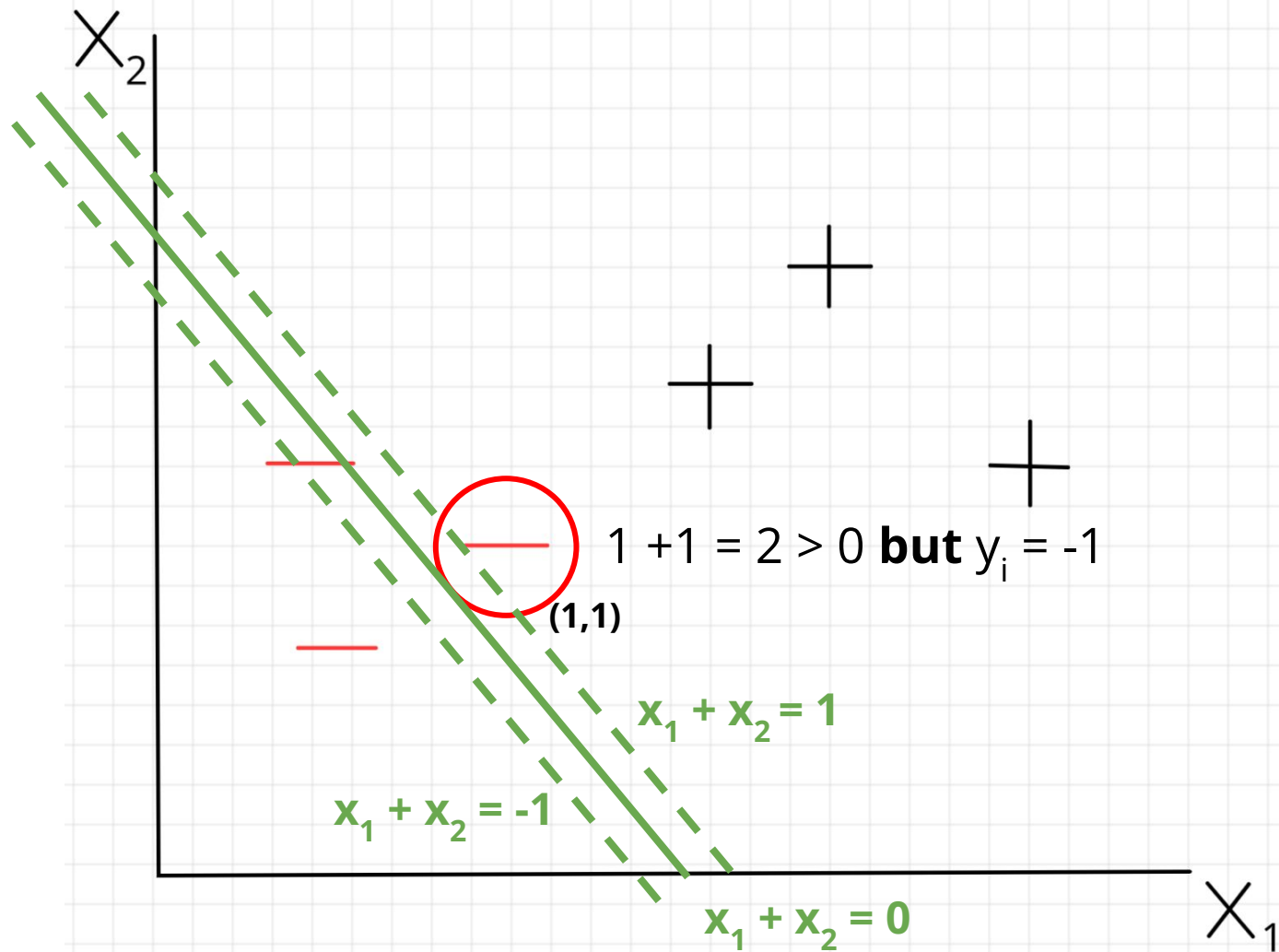
Assuming our data is linearly separable, we want to impose the constraint that **none of our samples can be in the street**. That is:

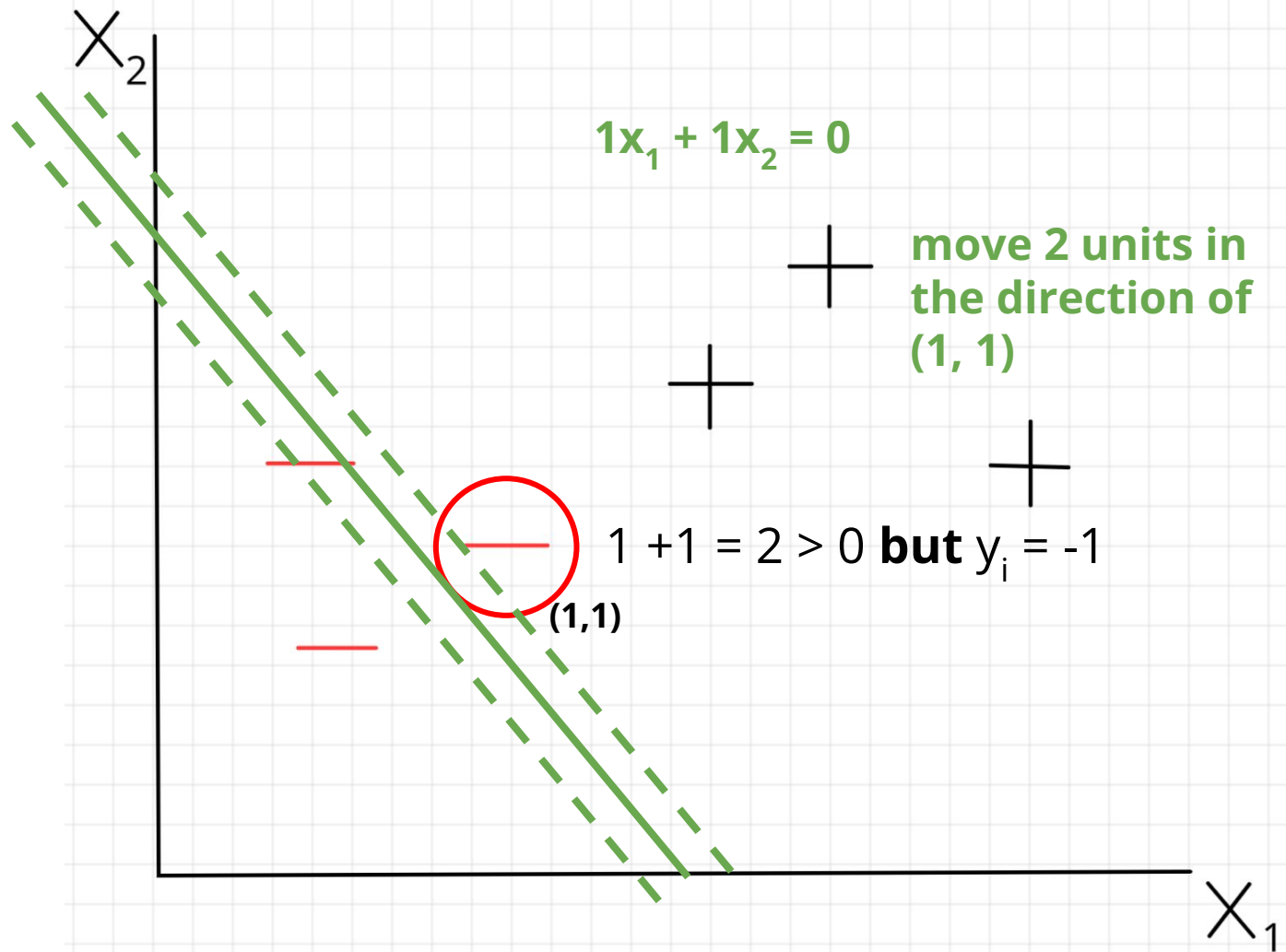
$$\vec{w} \cdot \vec{x}_+ + b \geq 1$$

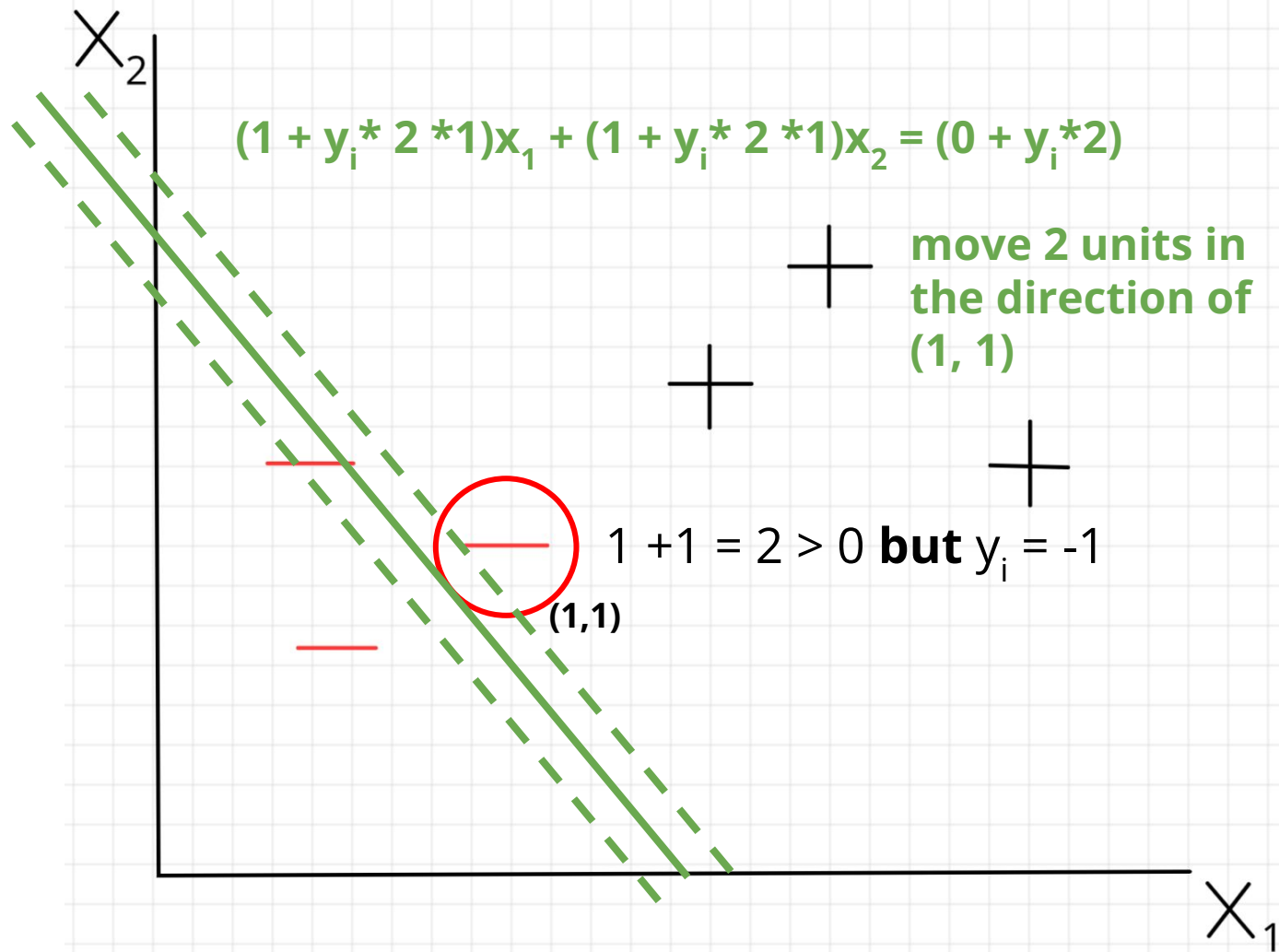
$$\vec{w} \cdot \vec{x}_- + b \leq -1$$

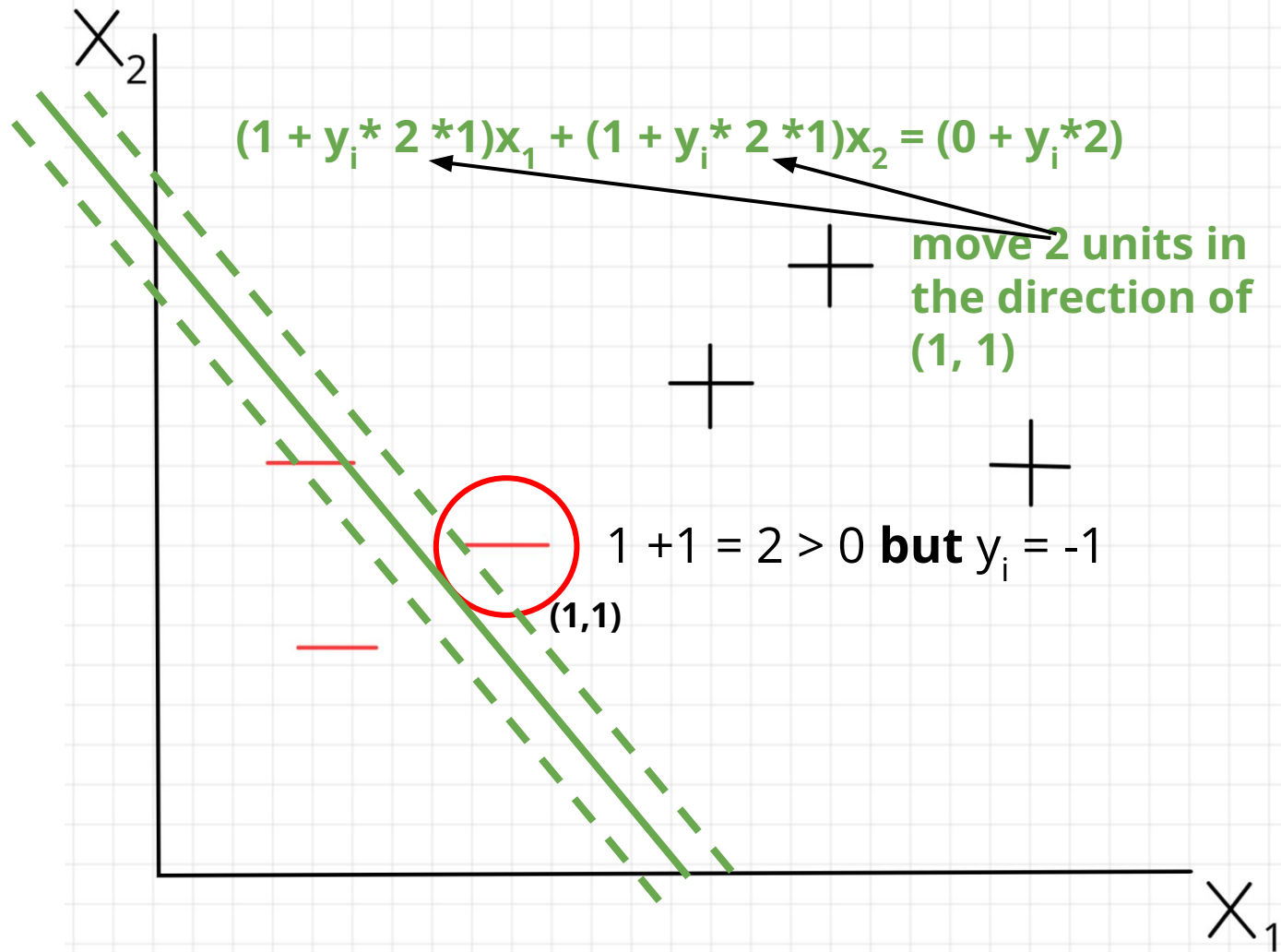
Learning w and b

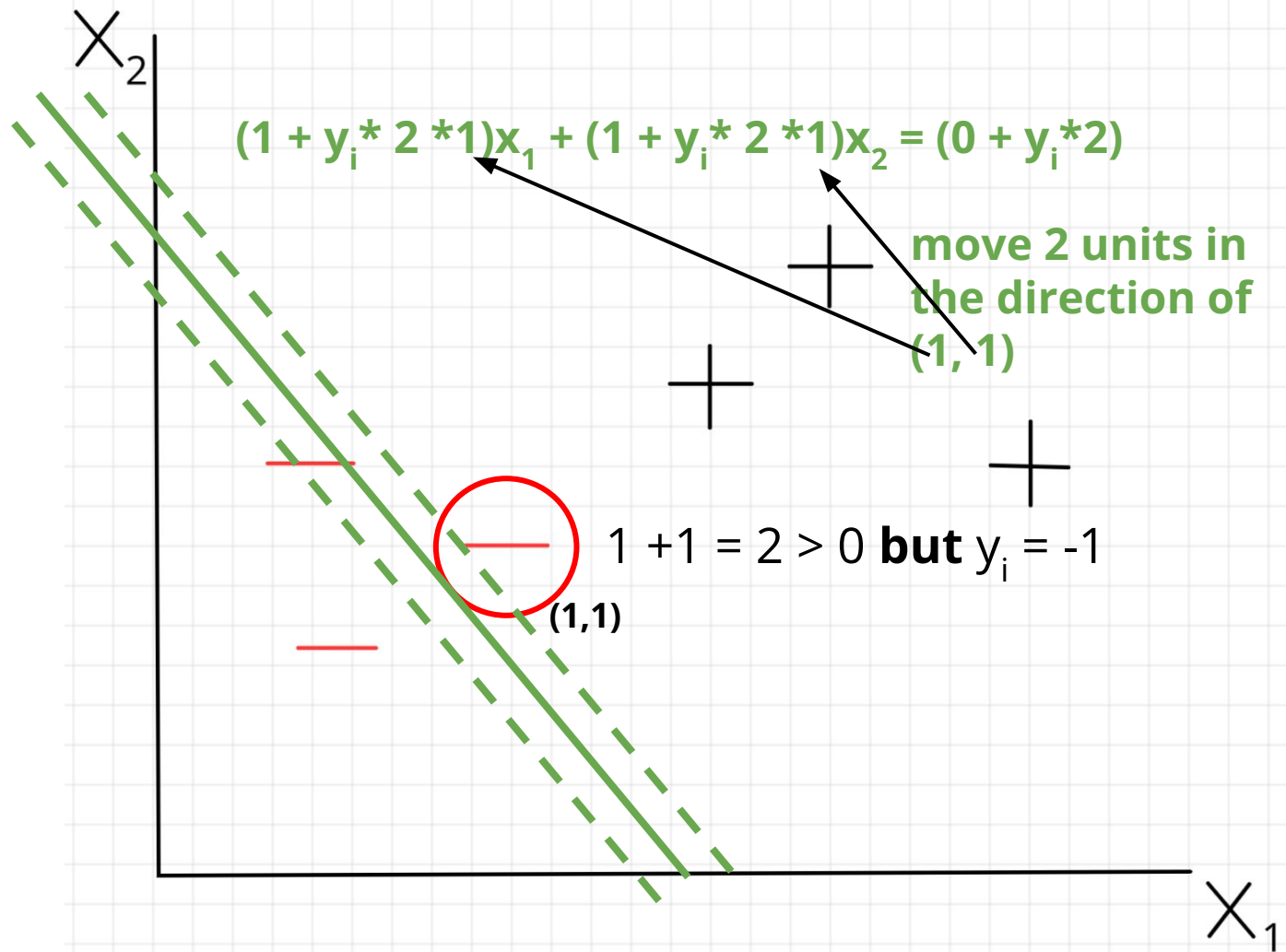


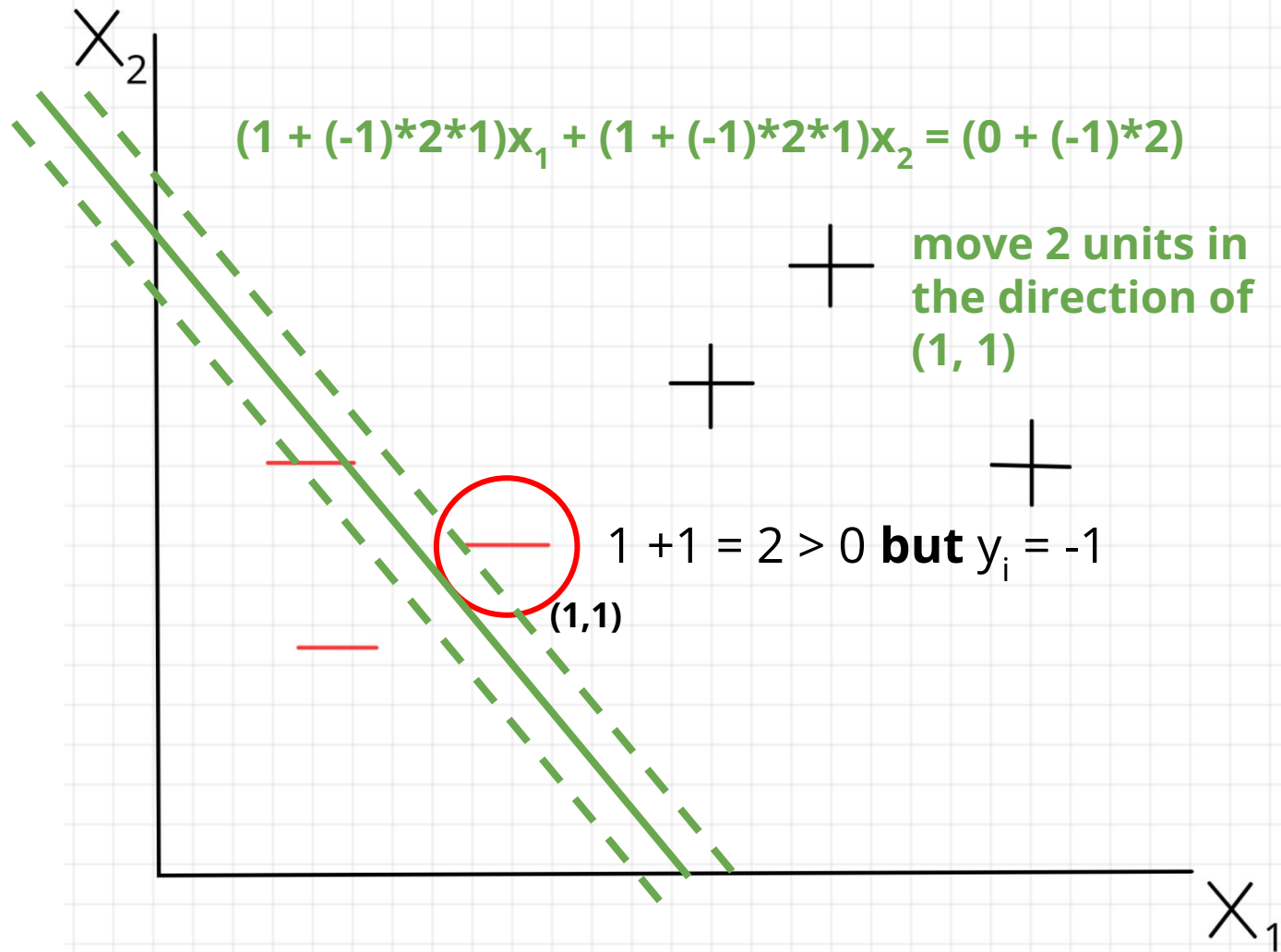


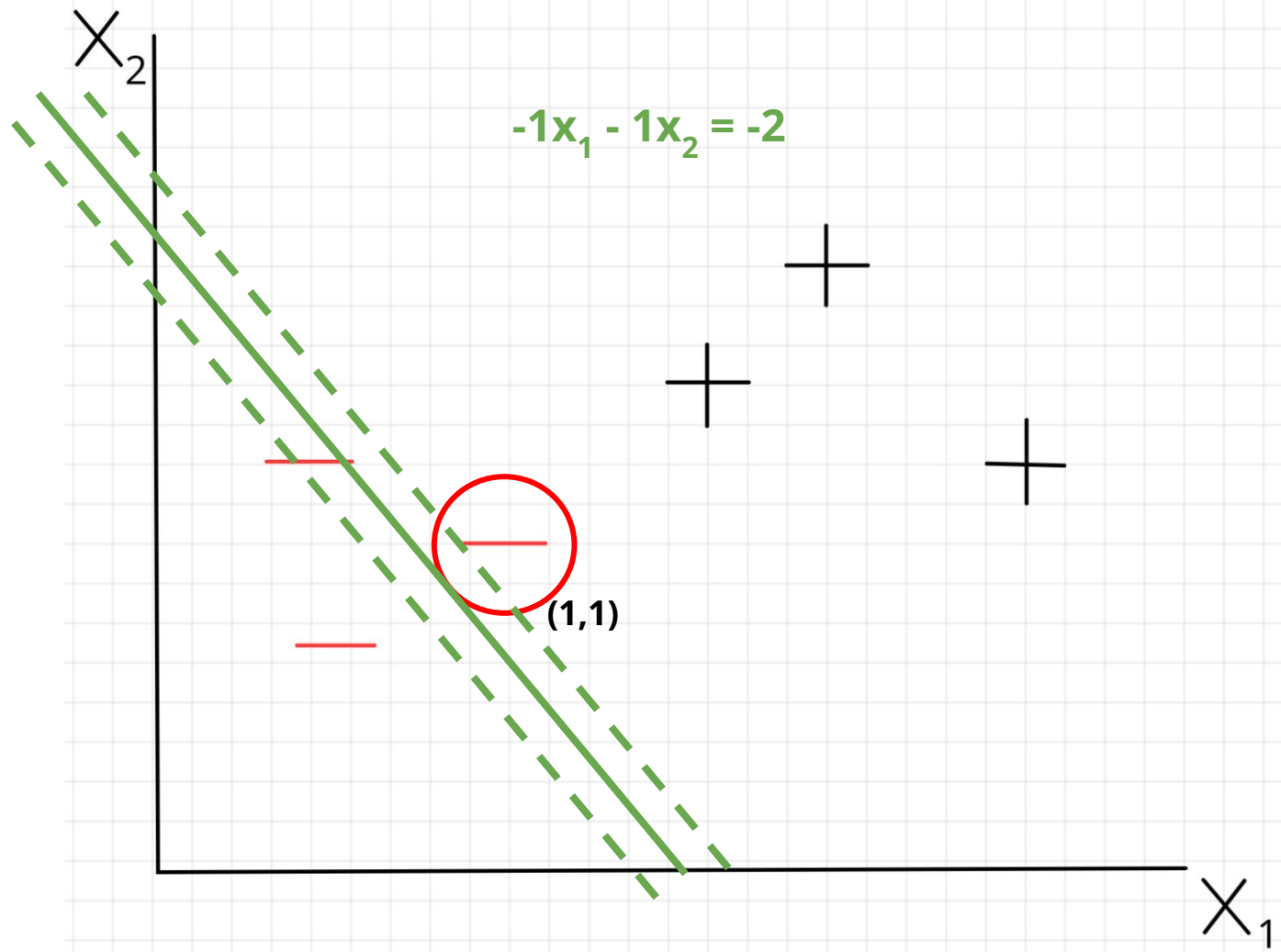


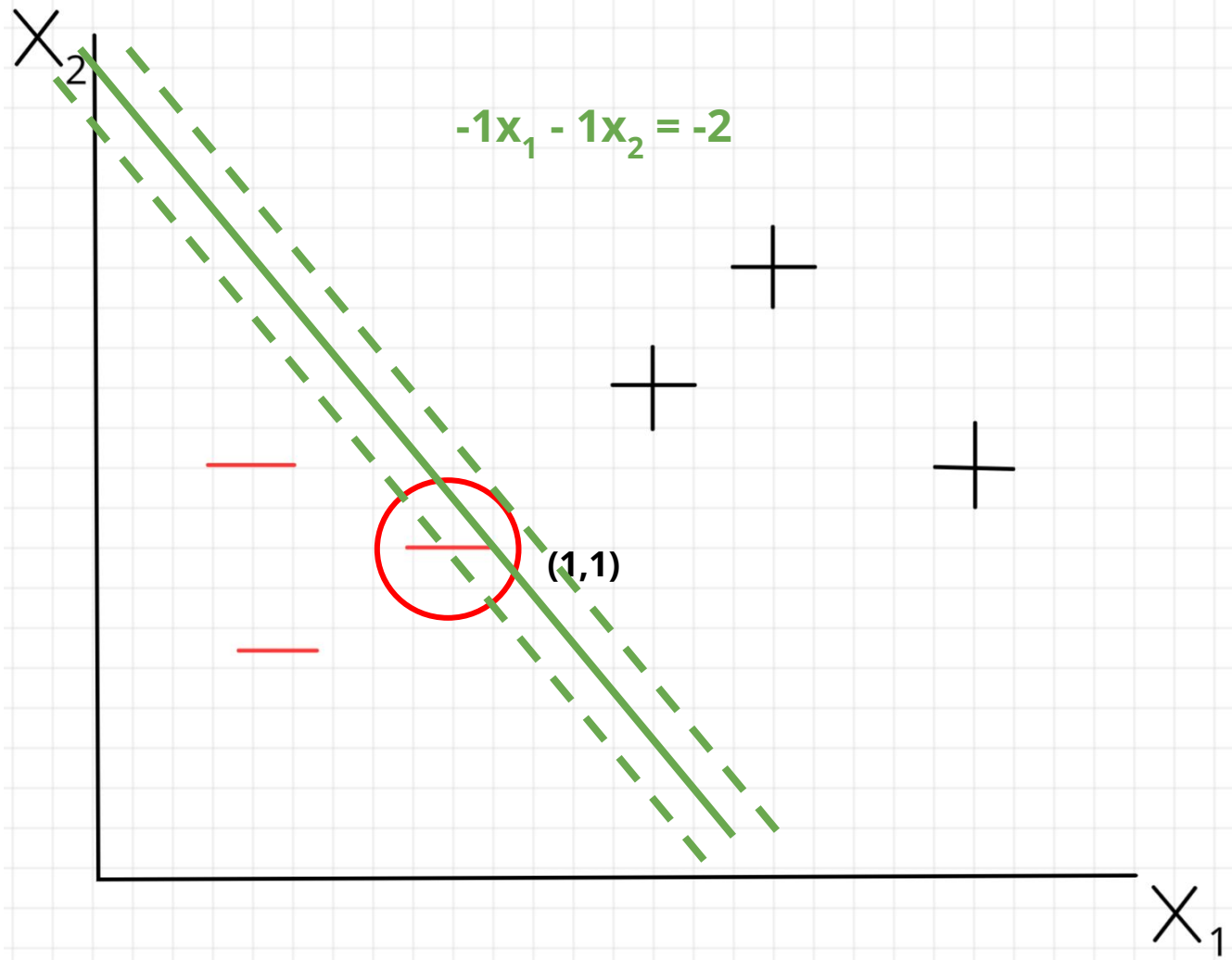












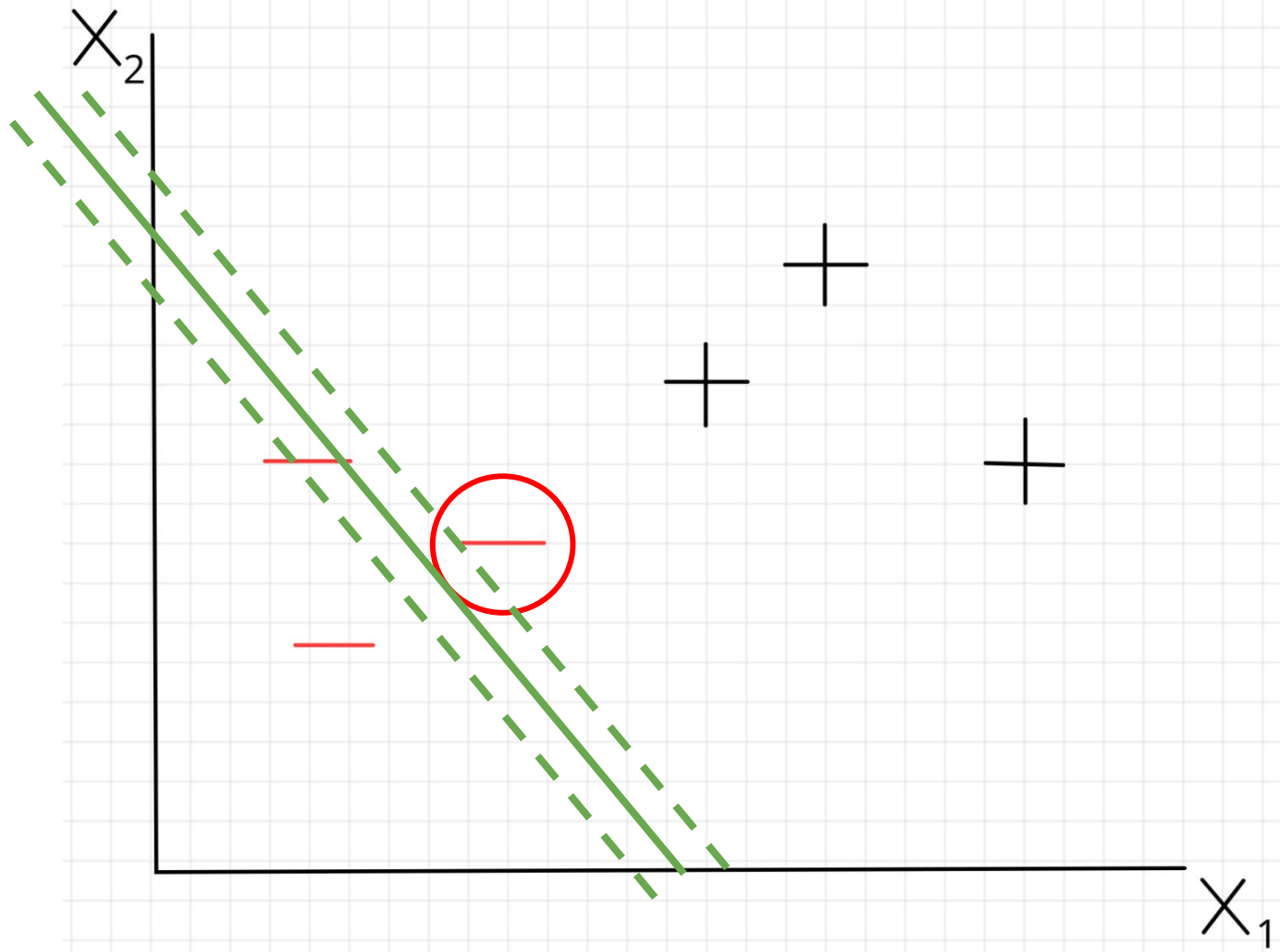
To move the street in the direction of a point

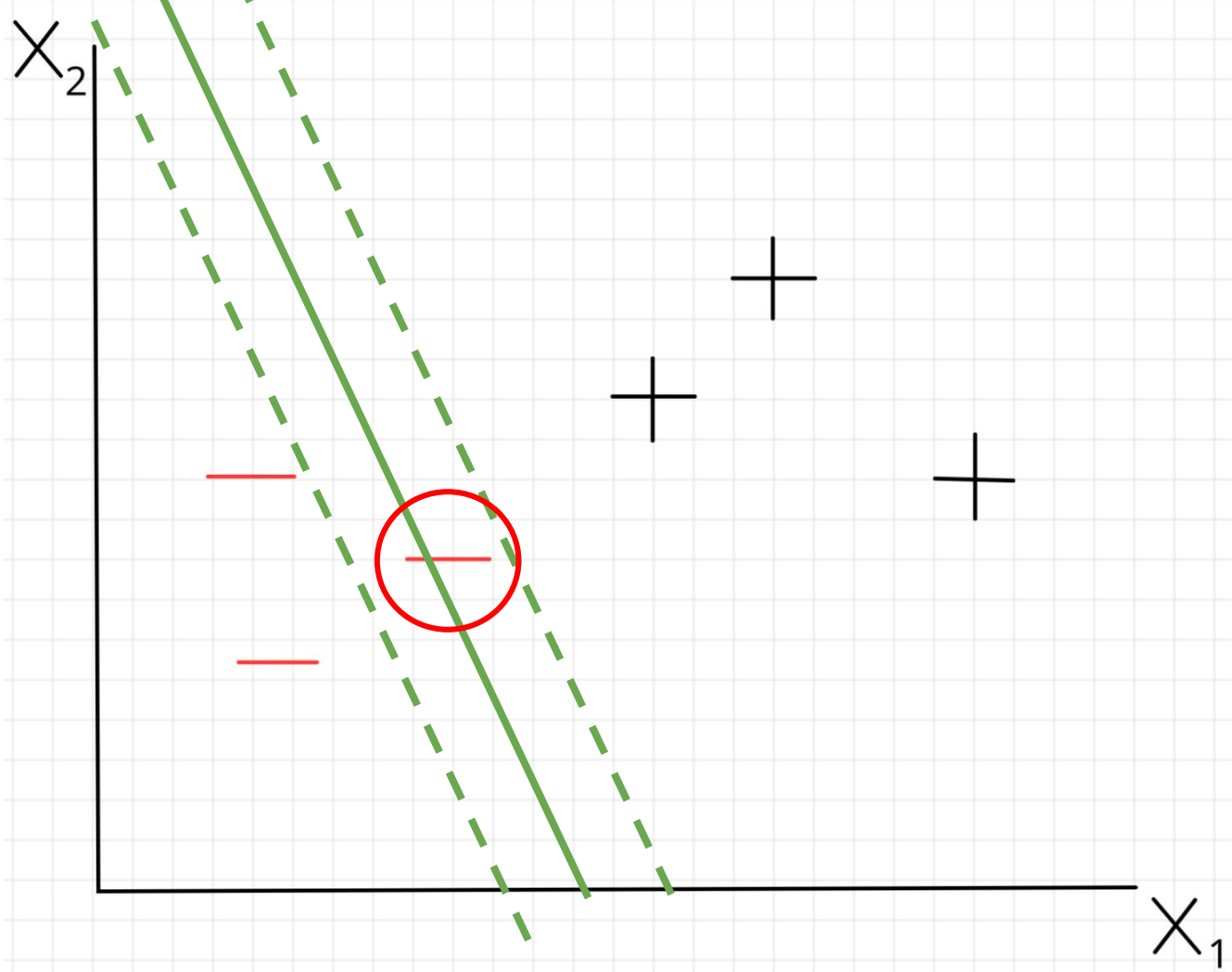
Pick a step size **a**, in order to move **a** steps in the direction of **x**

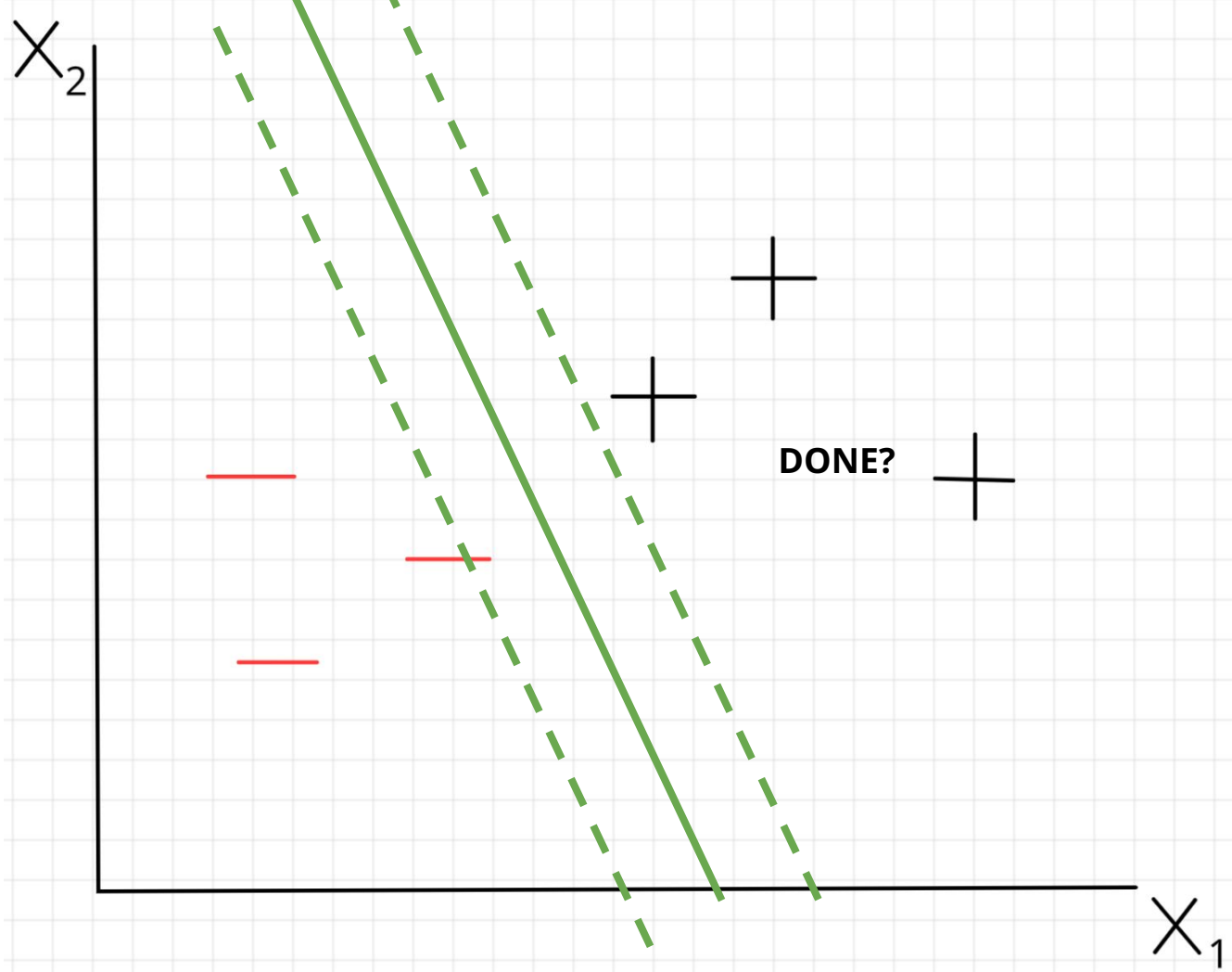
$$\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} + y_i * \mathbf{x} * a$$

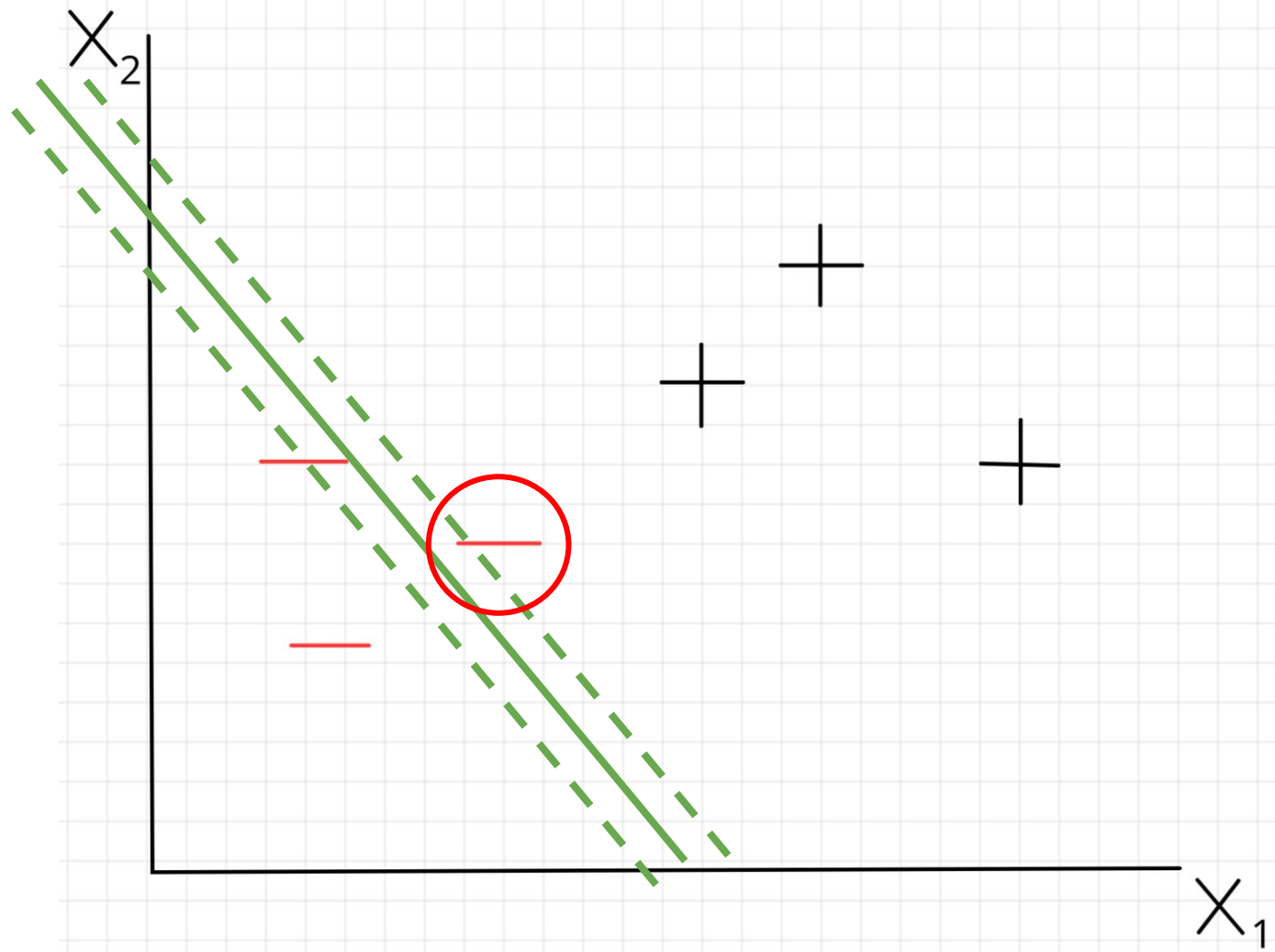
$$\mathbf{b}_{\text{new}} = \mathbf{b}_{\text{old}} + y_i * a$$

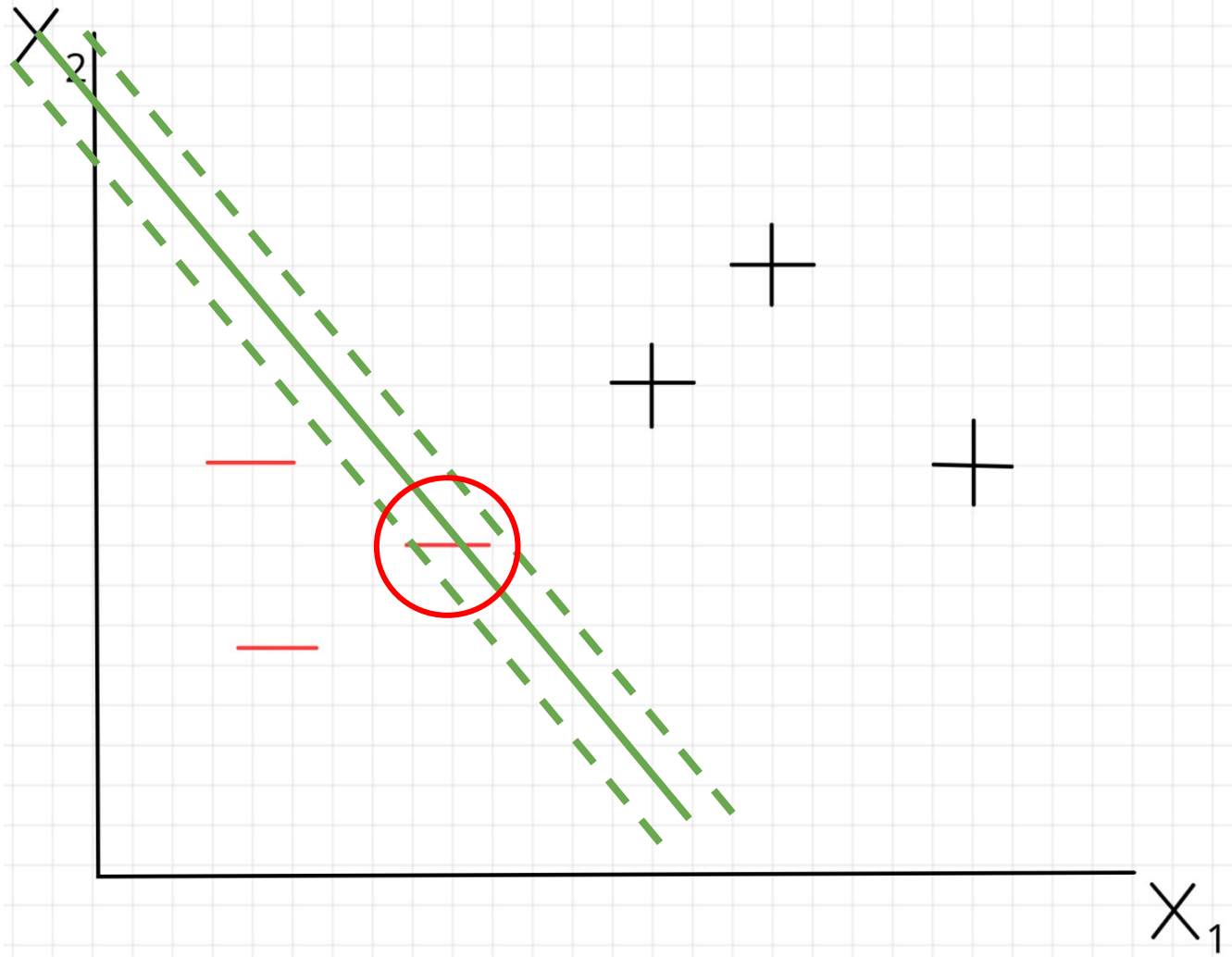
Now we know how to move the street in the right direction - but...

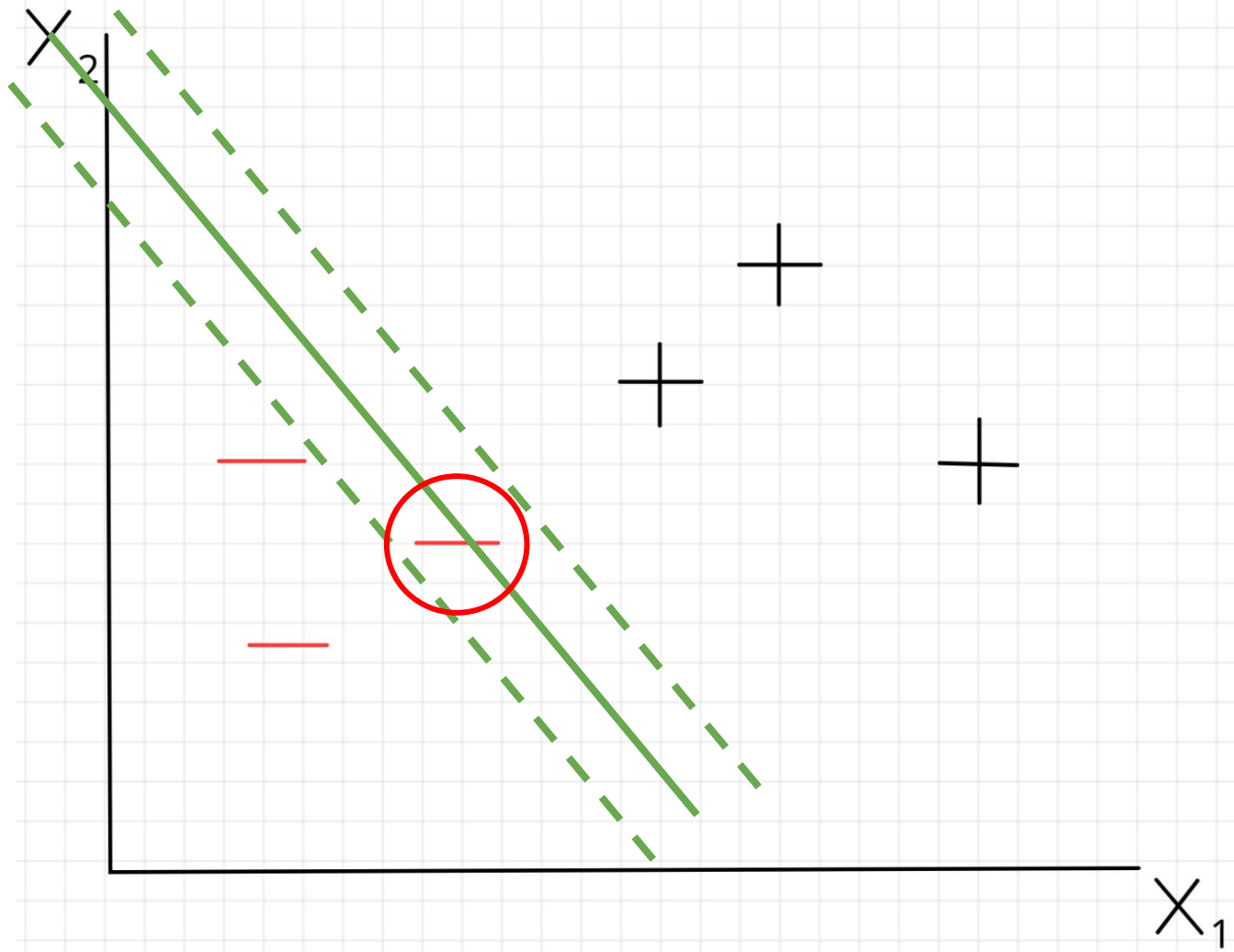


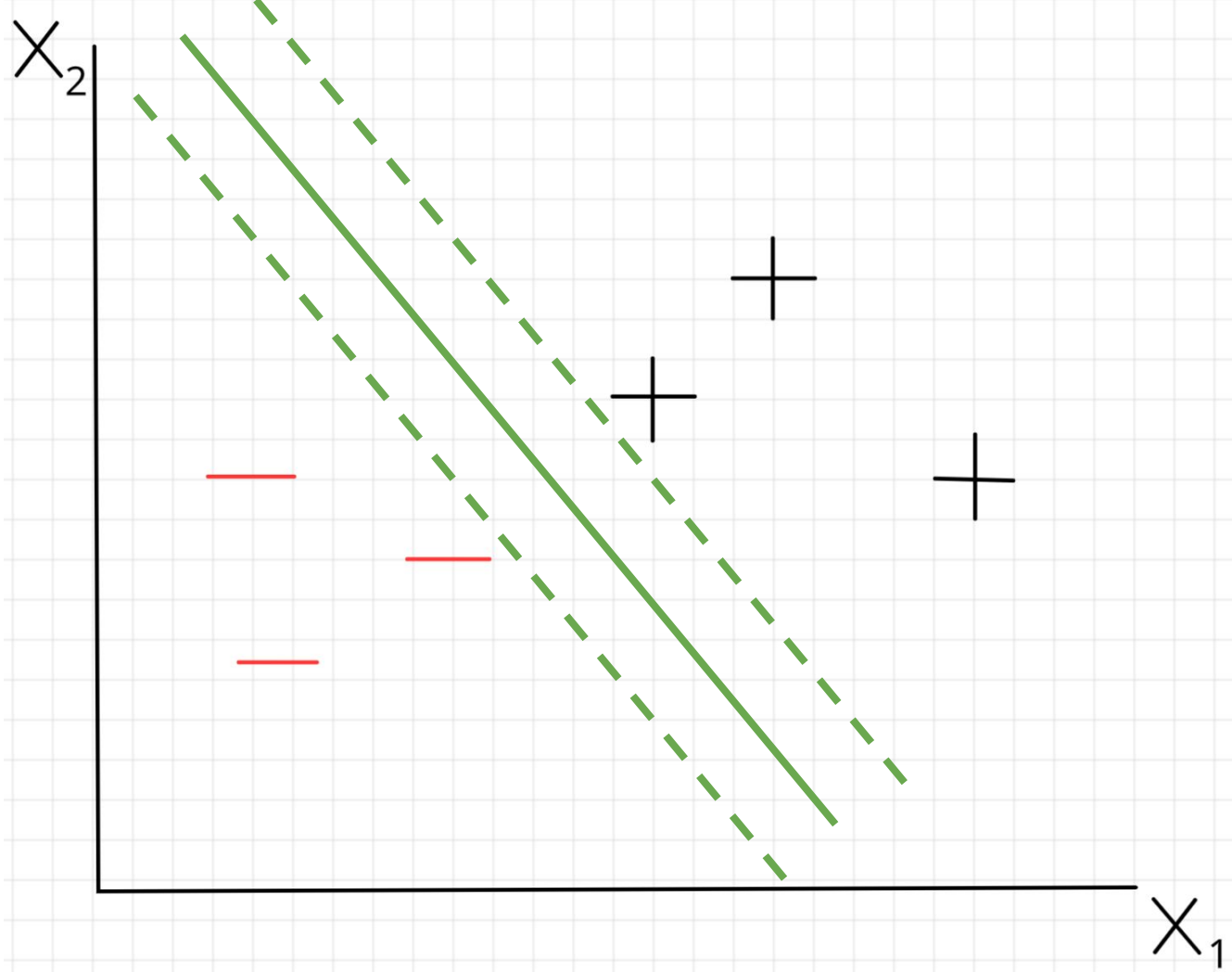


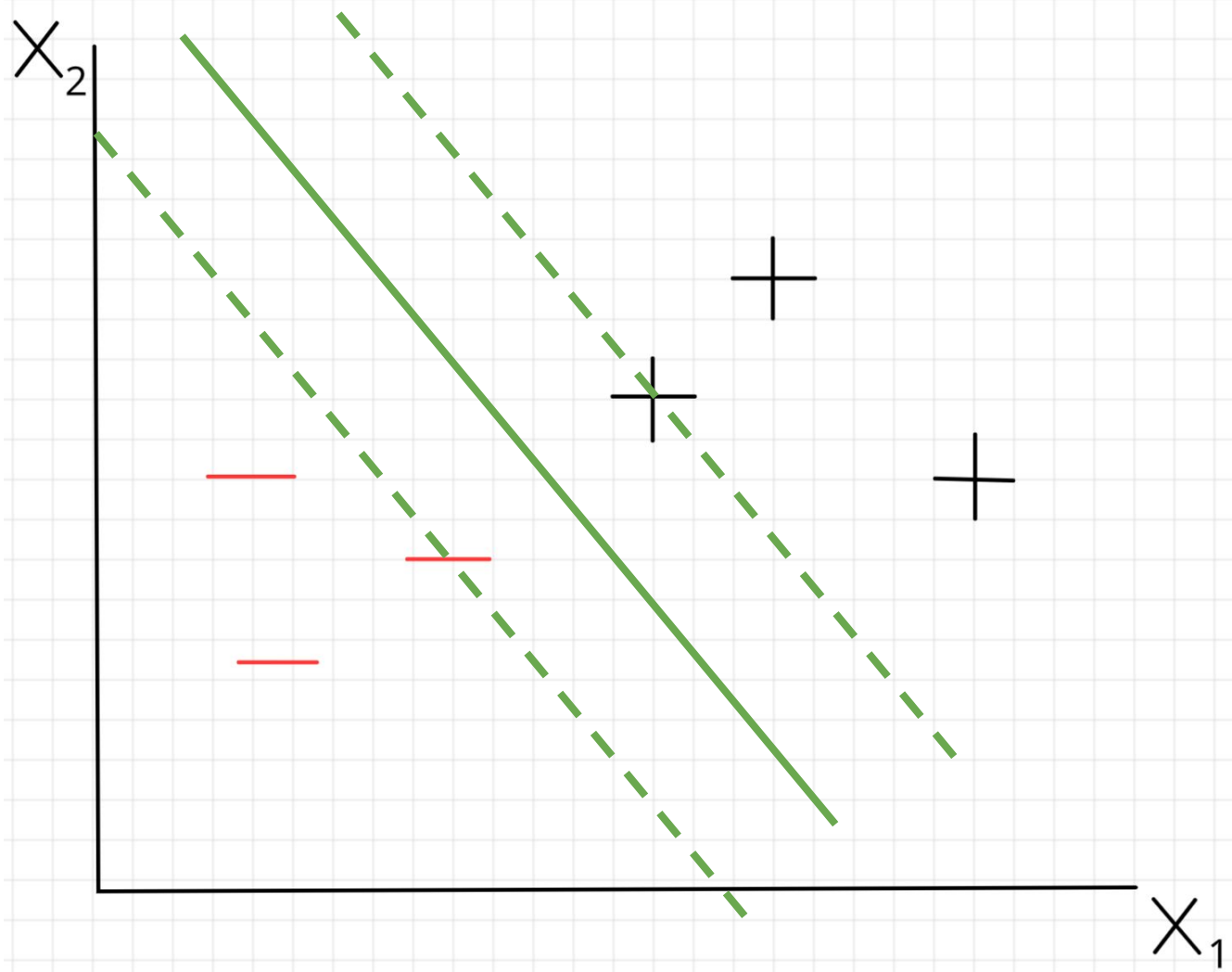






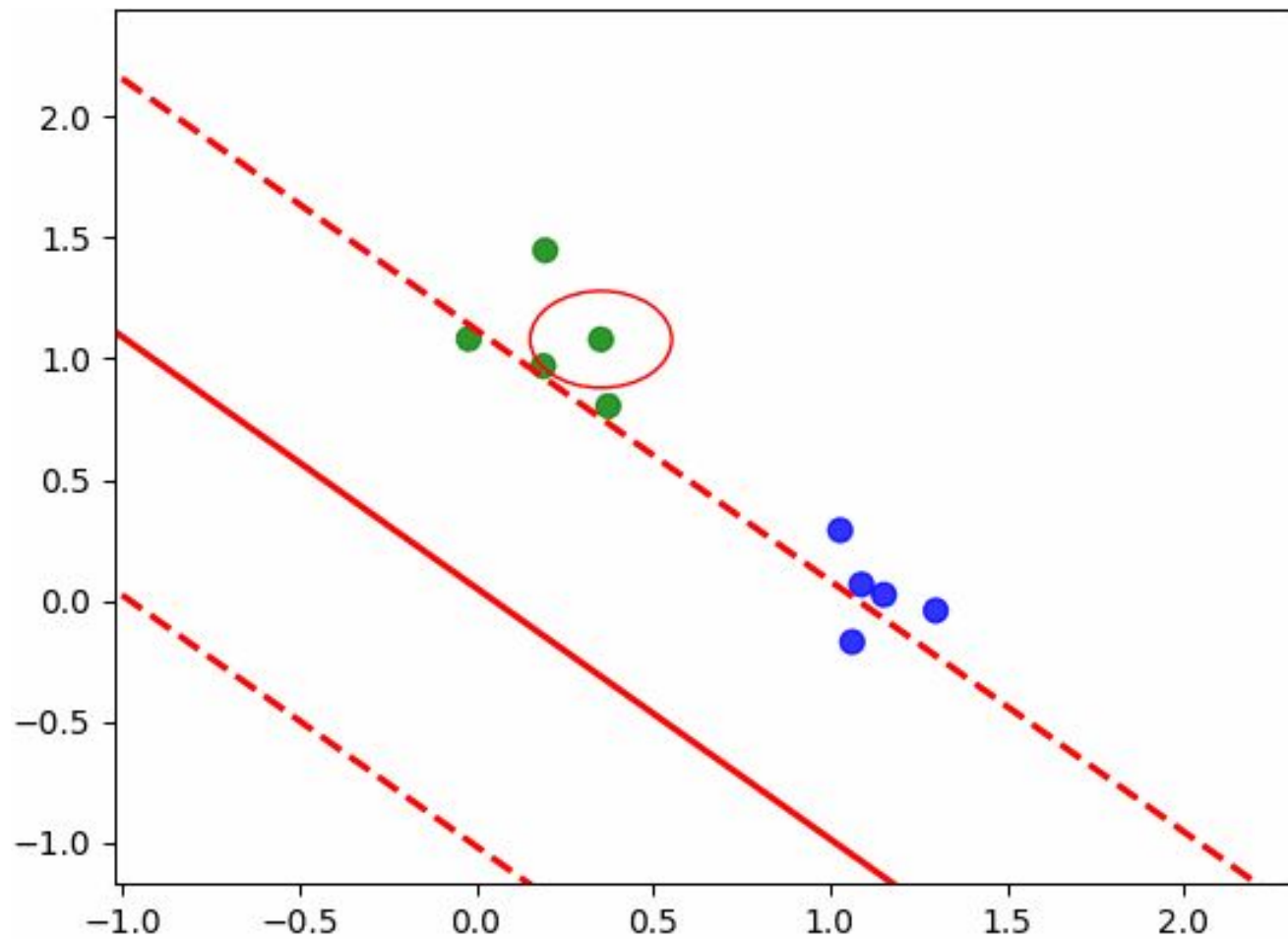




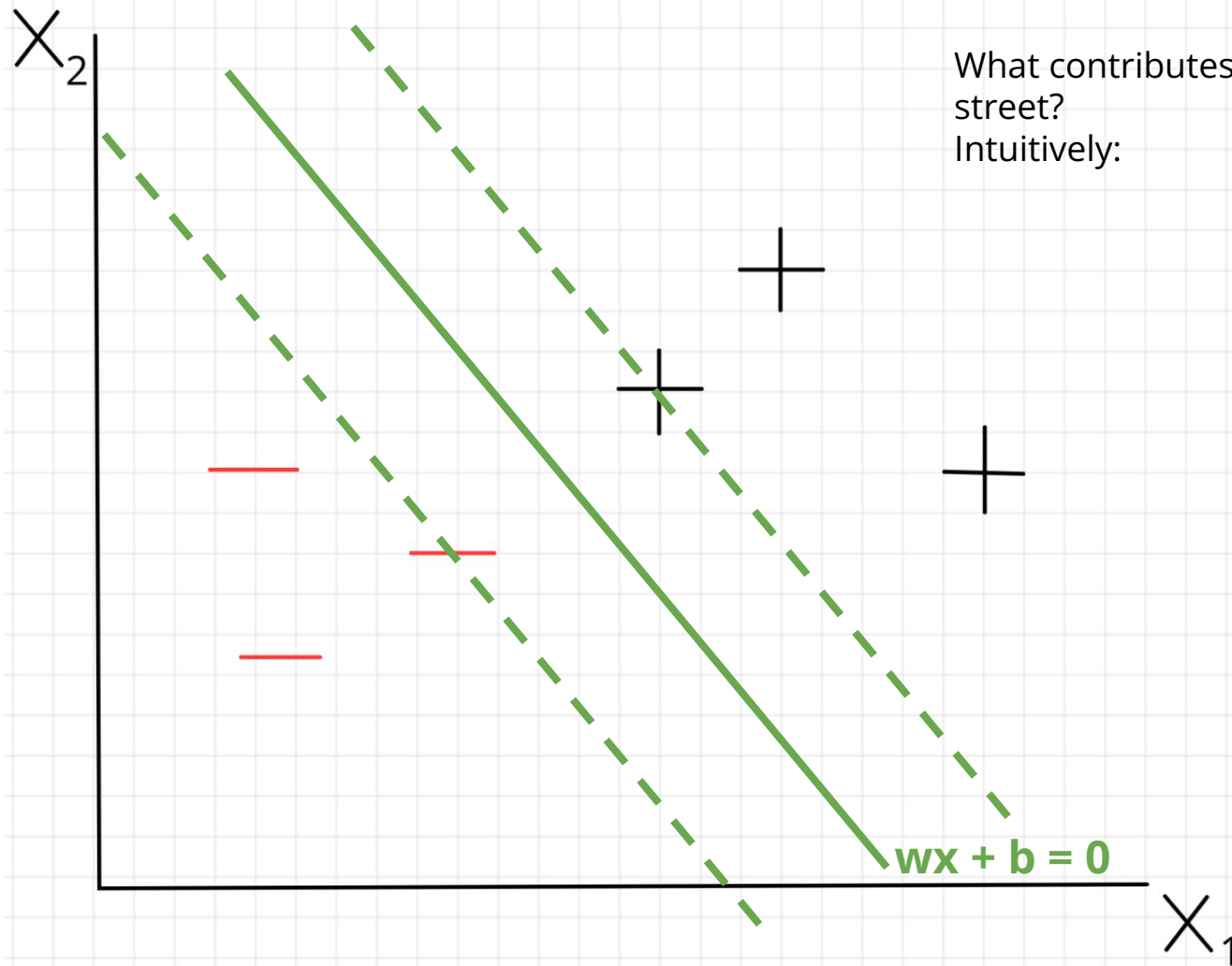


Full Algorithm (Perceptron Algorithm)

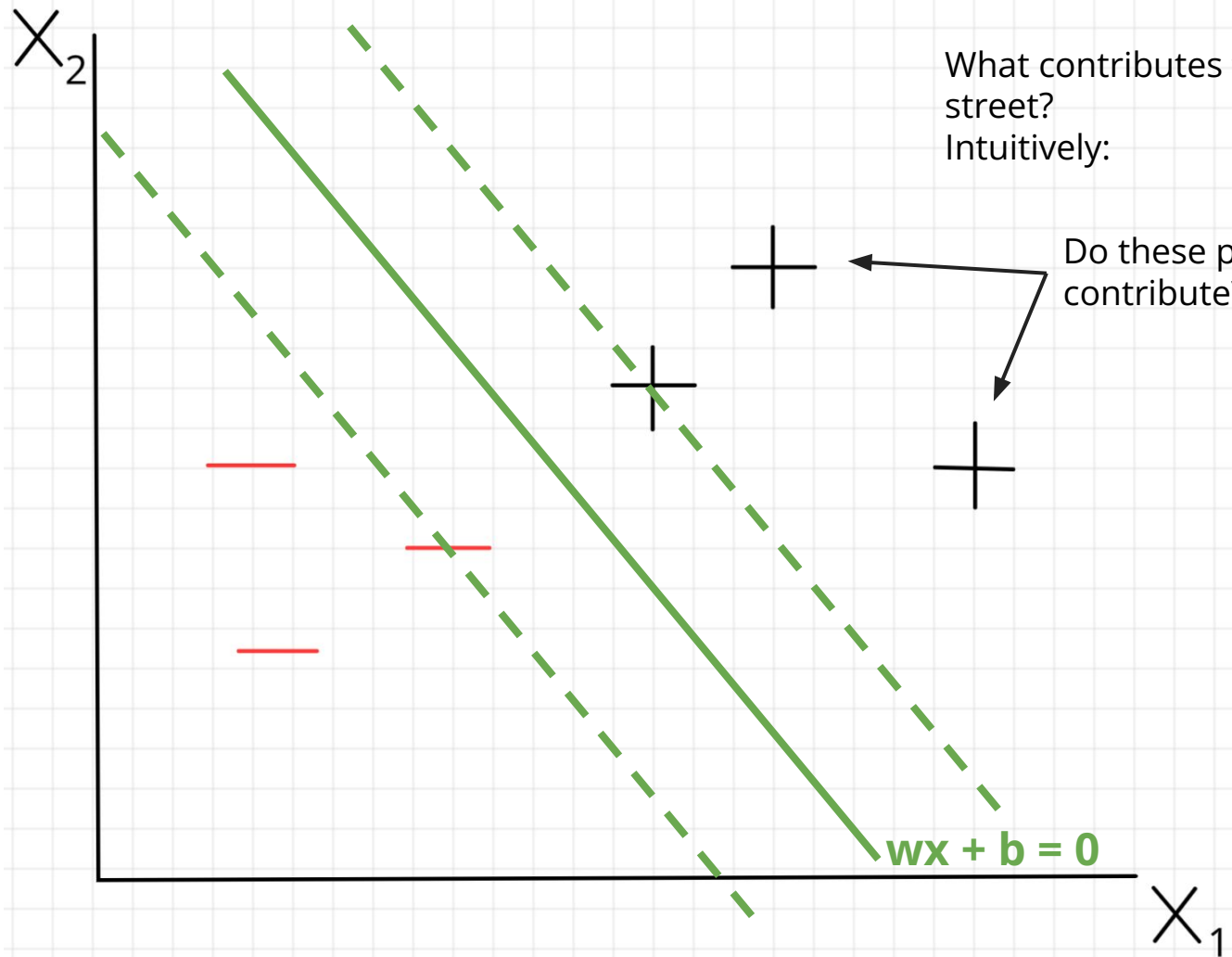
- Start with random line $w_1x_1 + w_2x_2 + b = 0$
- Define:
 - A total number of iterations (ex: 100)
 - A learning rate **a** (not too big not too small)
 - An expanding rate **c** (< 1 but not too close to 1)
- Repeat **number of iterations** times:
 - Pick a point (x_i, y_i) from the dataset
 - If 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**)



Worksheet a) -> c)

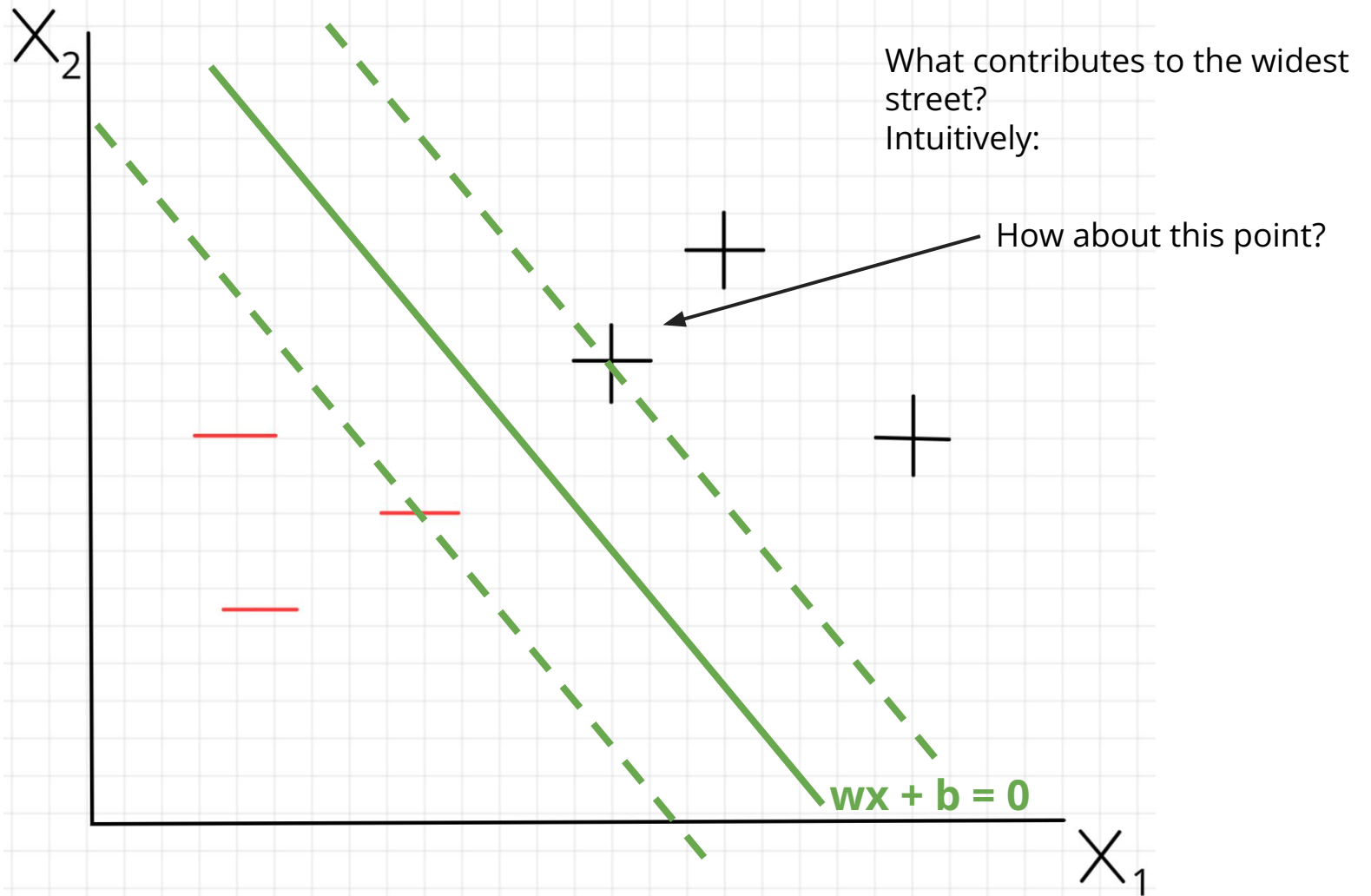


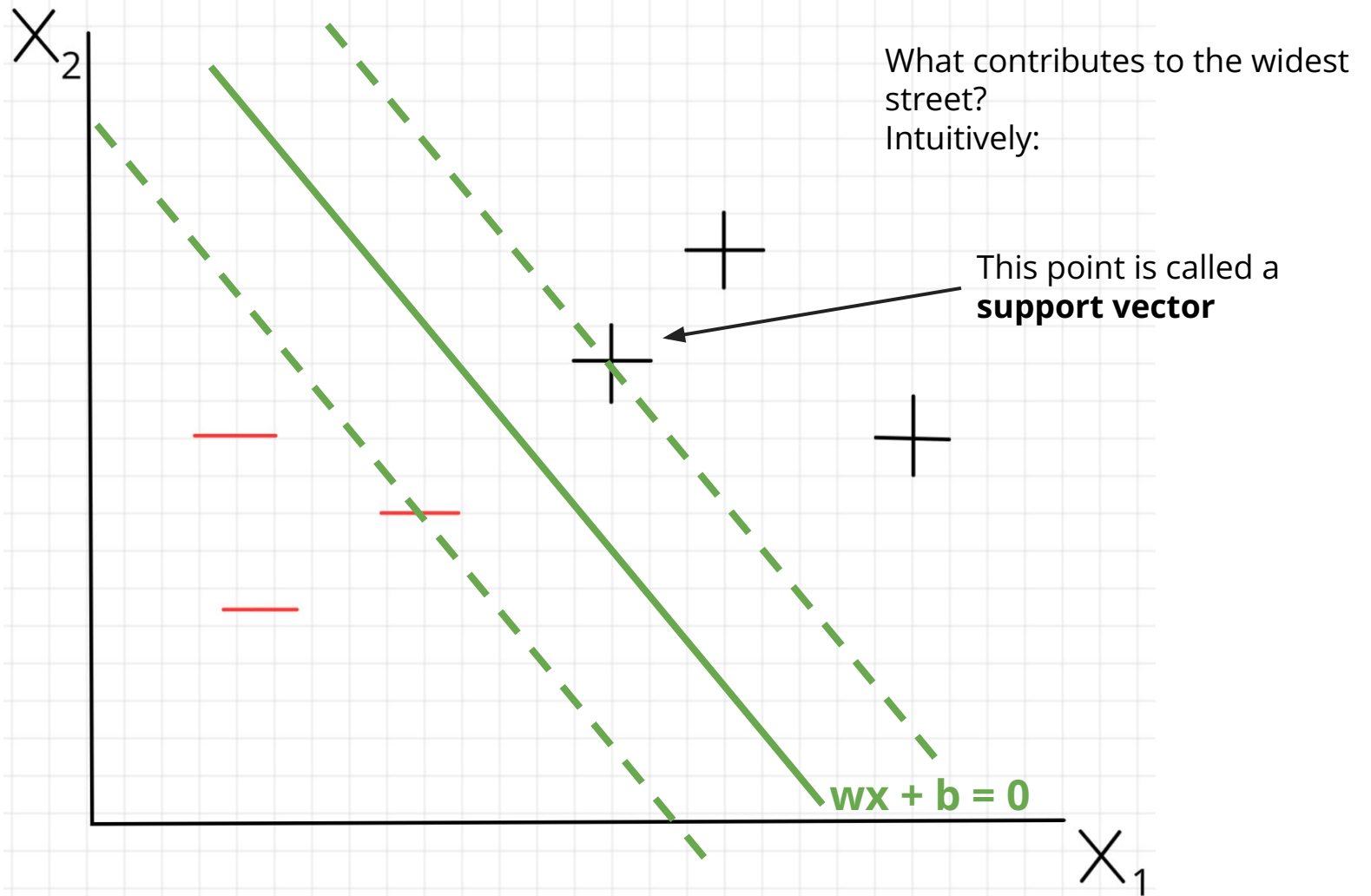
What contributes to the widest street?
Intuitively:



What contributes to the widest street?
Intuitively:

Do these points
contribute?





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 \mathbf{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

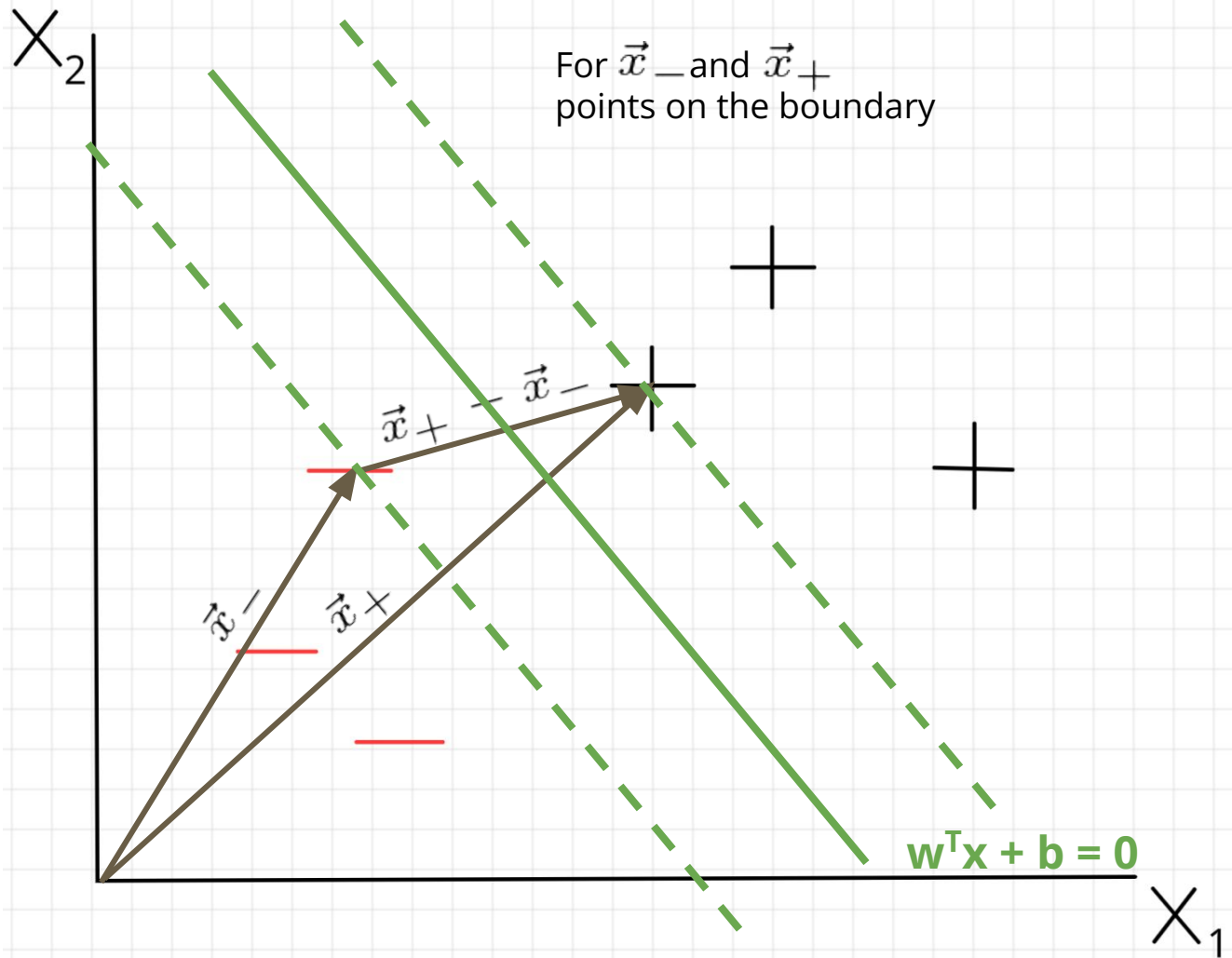
Find the widest street subject to...

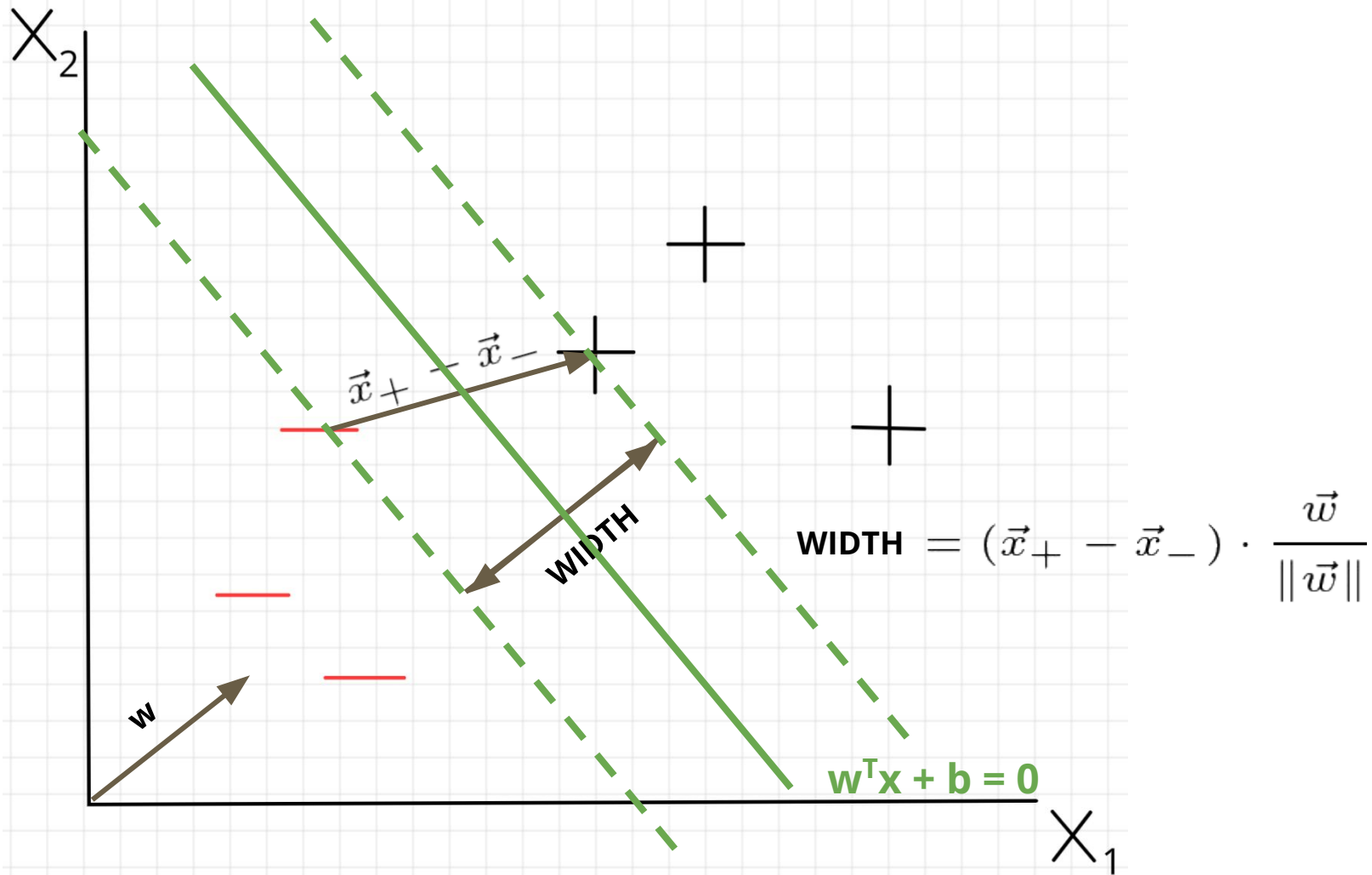
If we multiply our sample decision rules by this new variable:

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

Meaning, for \vec{x}_i on the decision boundary, we want:

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





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

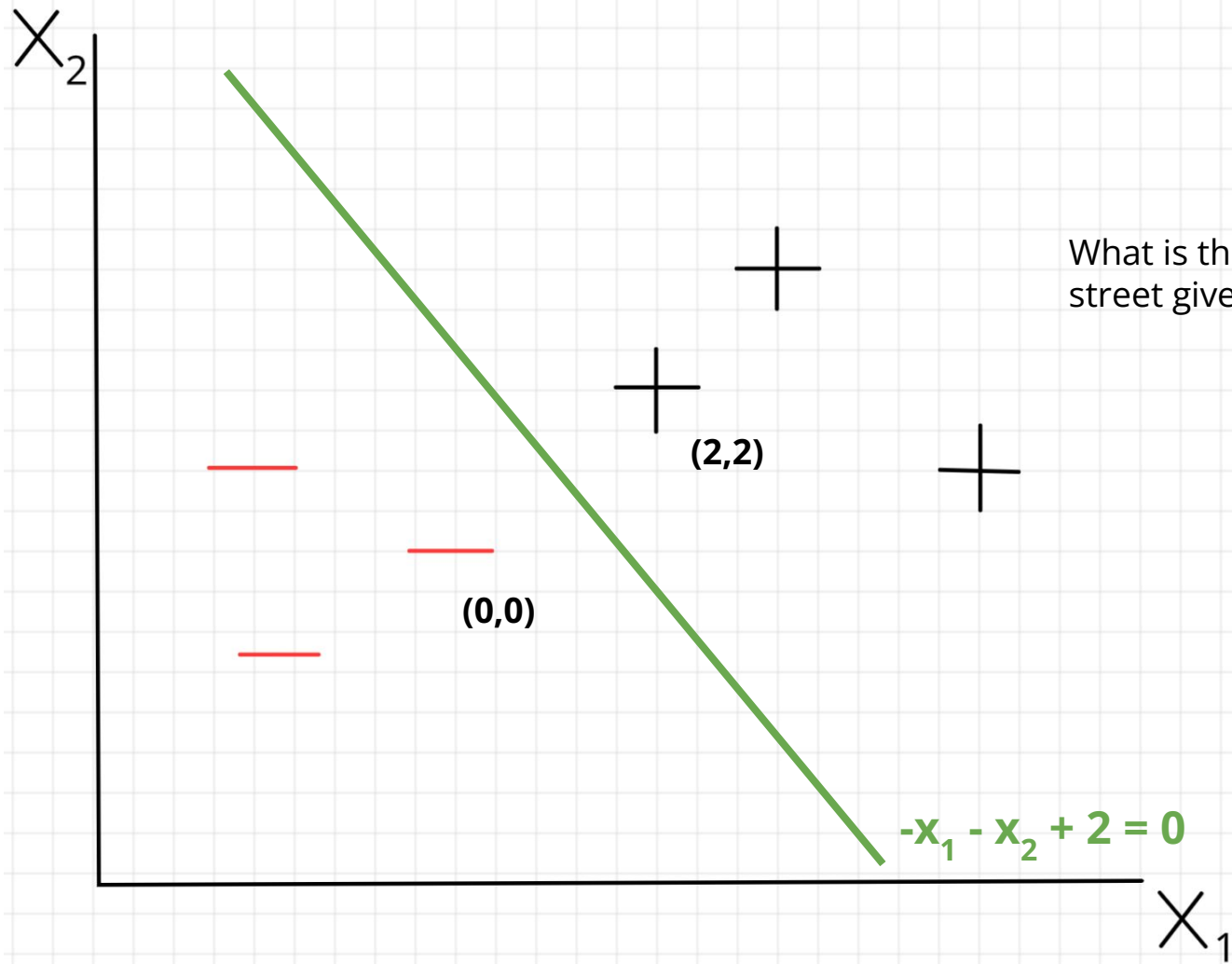
$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 = 0$$

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

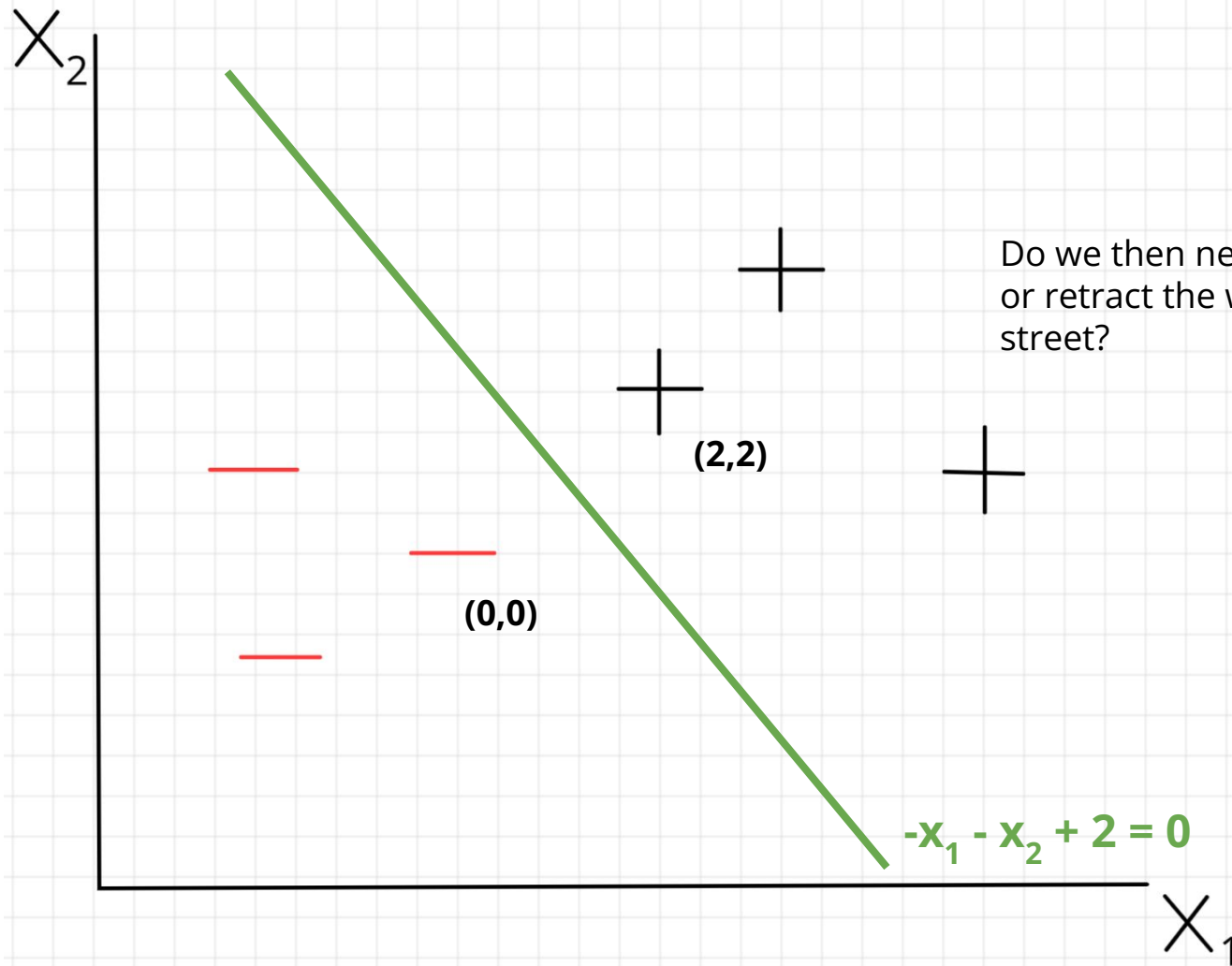
What does that mean?

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

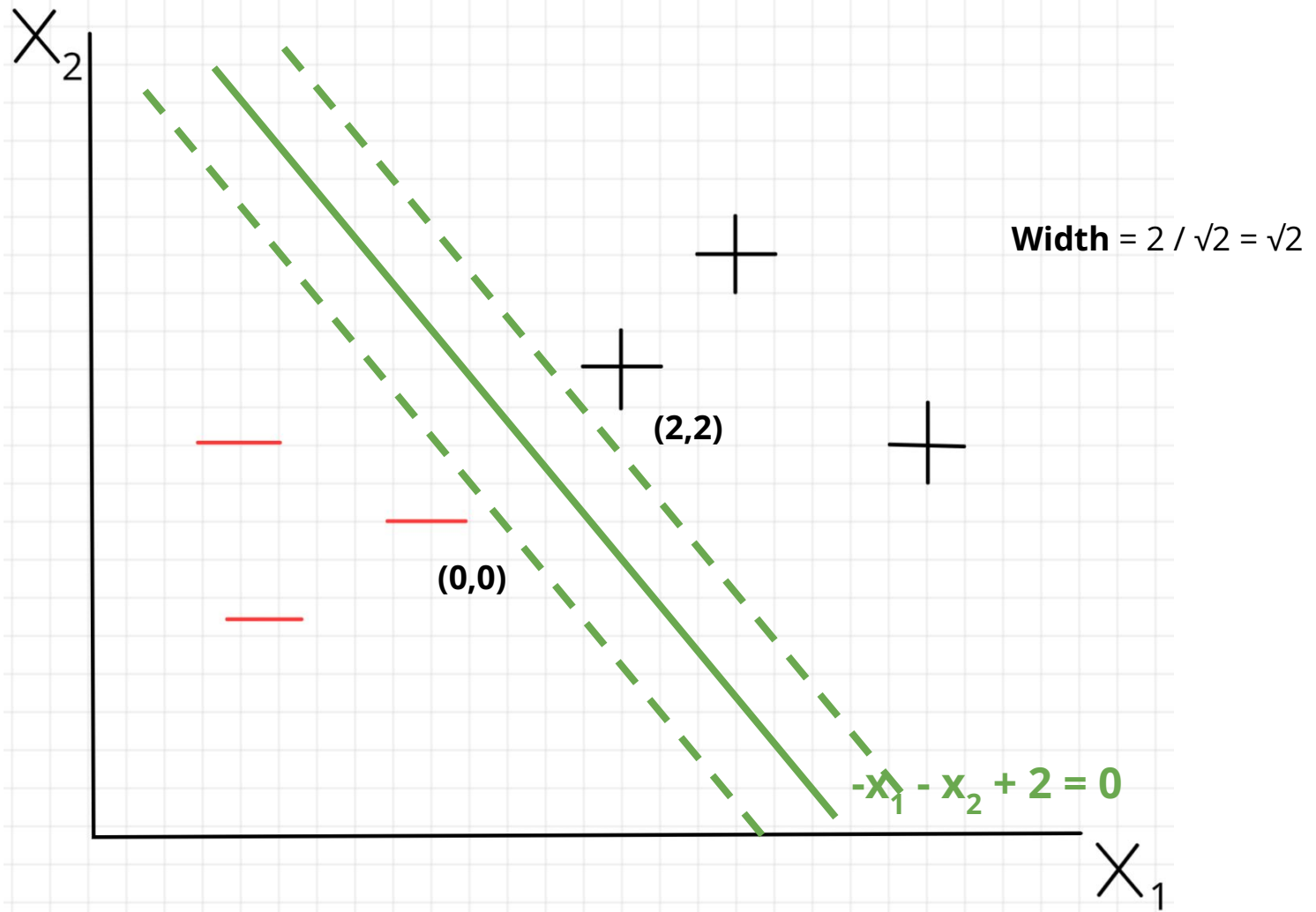
Aligns with what we found previously.



What is the width of the street given by w ?



Do we then need to expand or retract the width of the street?



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

$$\begin{aligned}\max\left(\frac{2}{\|\vec{w}\|}\right) &= \min(\|\vec{w}\|) \\ &= \min\left(\frac{1}{2} \|\vec{w}\|^2\right)\end{aligned}$$

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

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

where α_i is 0 for \vec{x}_i not on the boundary.

Let's take the partial derivative of L wrt to \vec{w} to see what \vec{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!

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

To move the street in the direction of a point

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

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

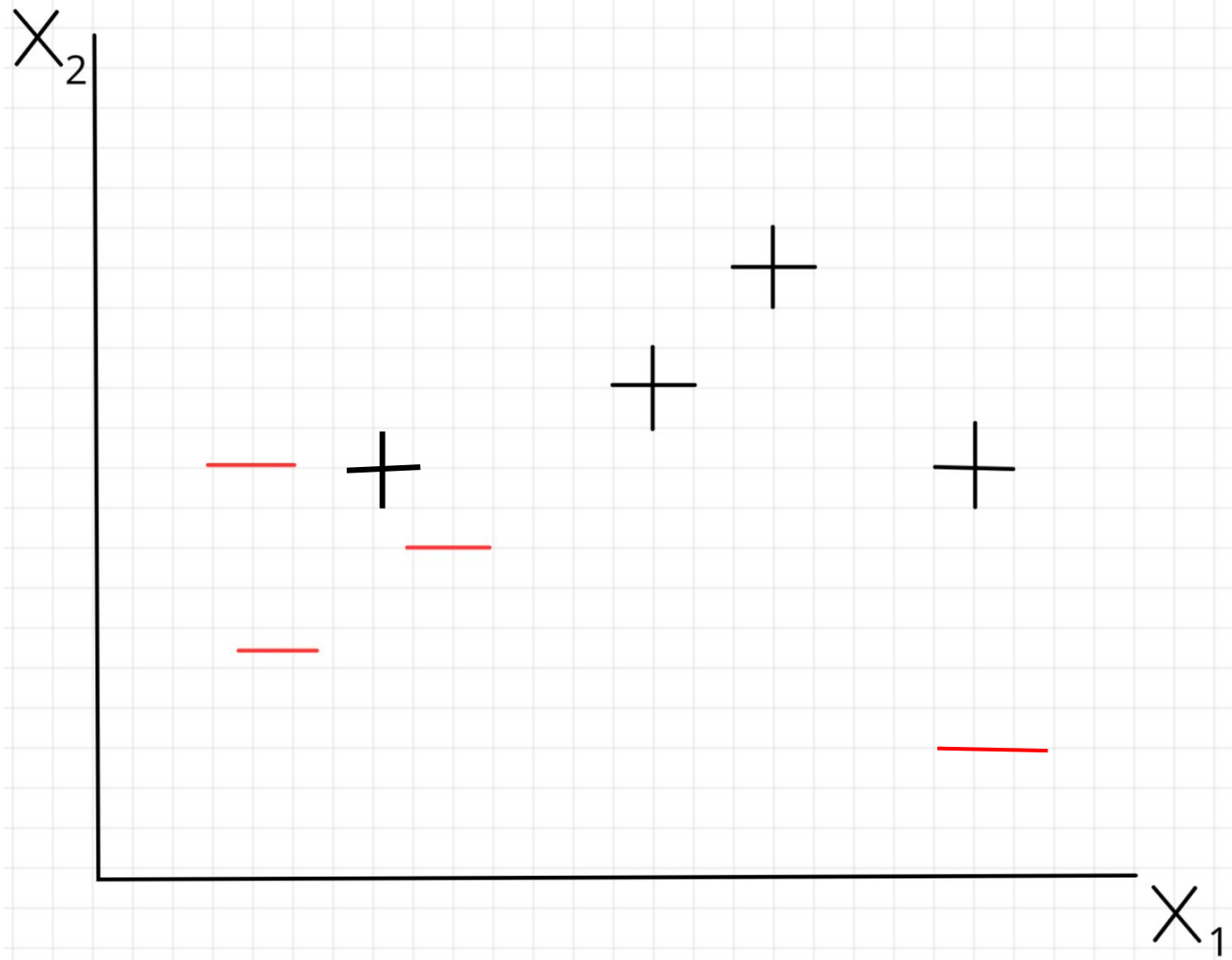
Quadratic Programming

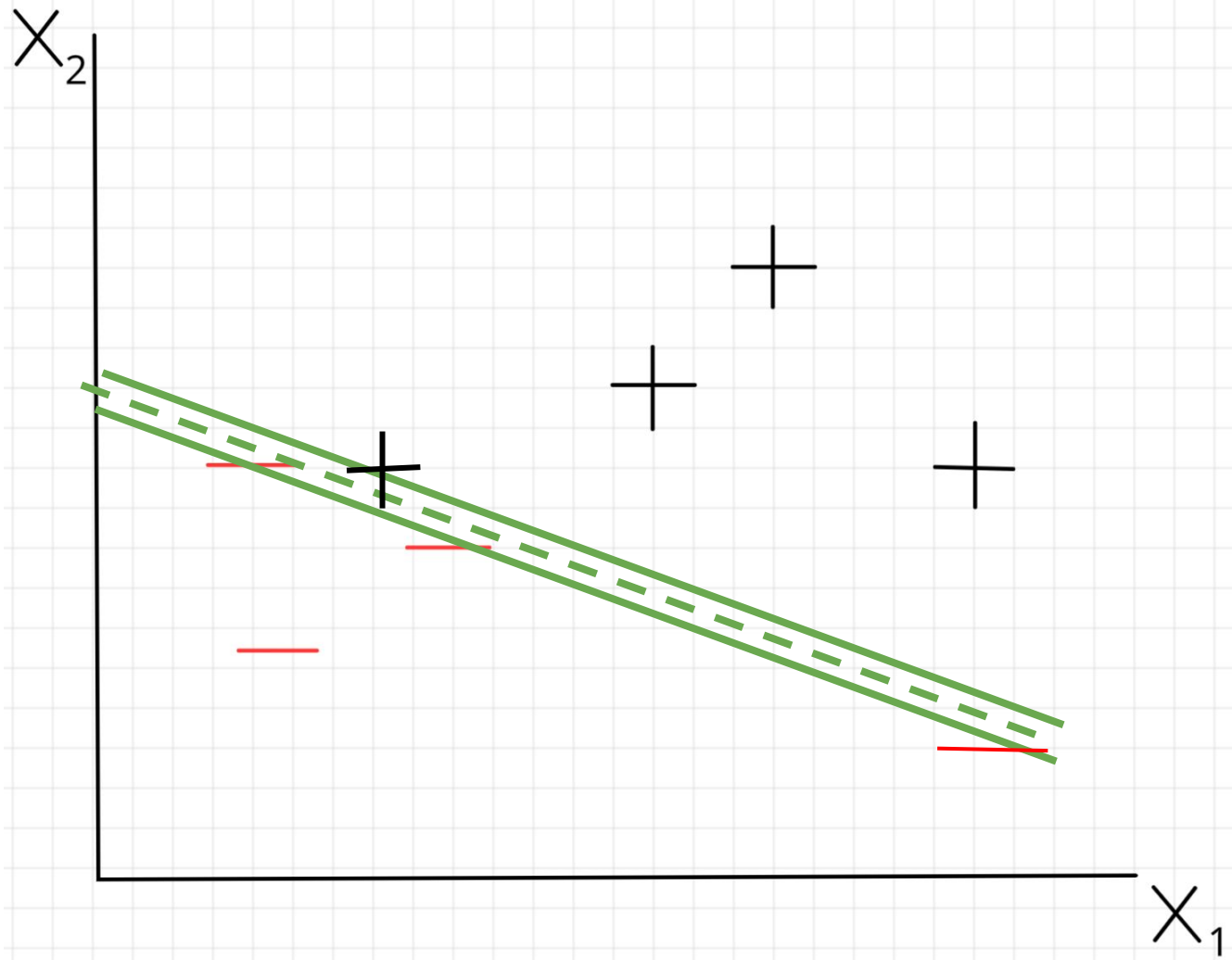
Solving lagrange multipliers requires numerical optimization methods called quadratic programming solvers.

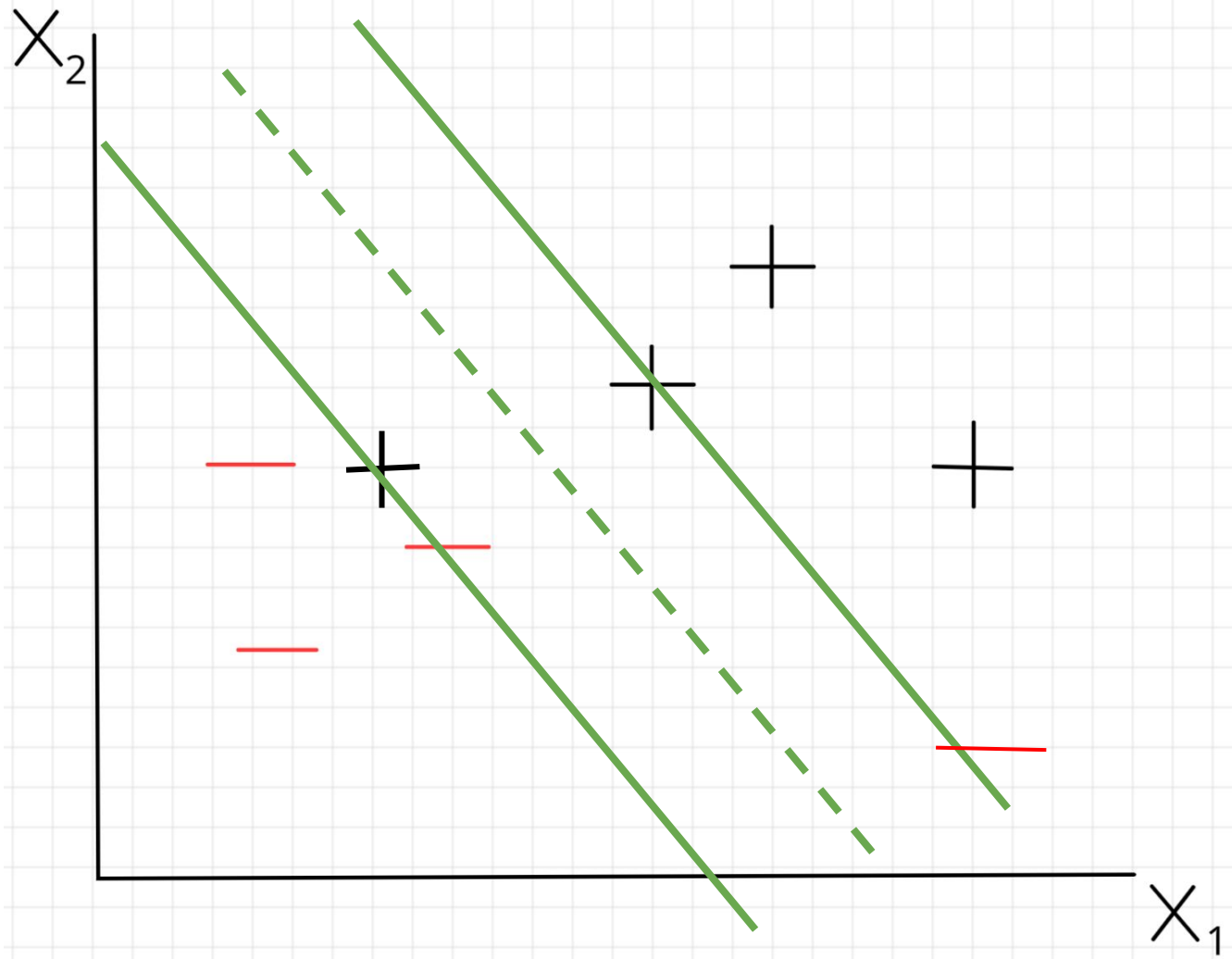
<https://pypi.org/project/qpsolvers/>

<https://cvxopt.org/examples/tutorial/qp.html>

Trade-off between width and error







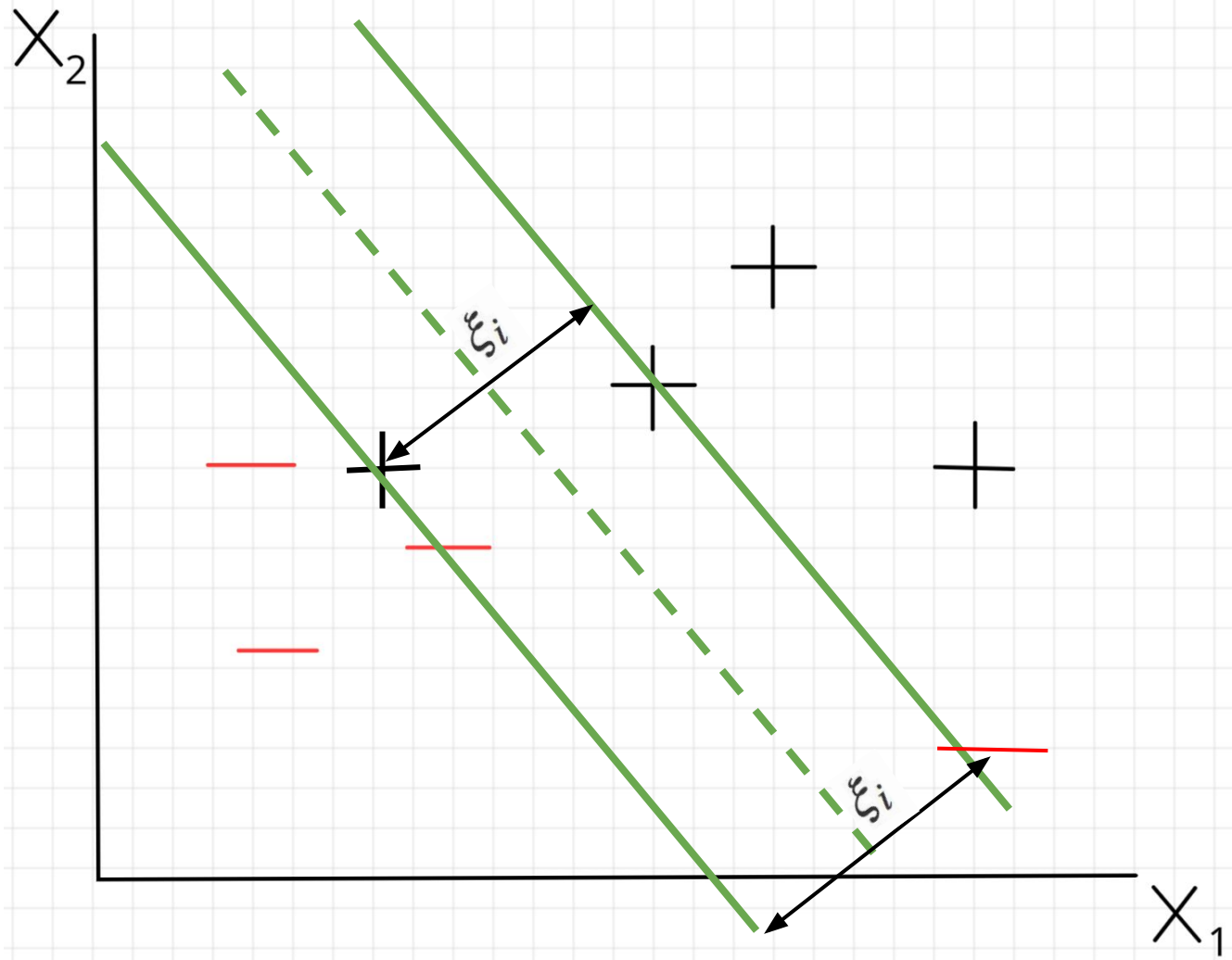
How to find the widest street

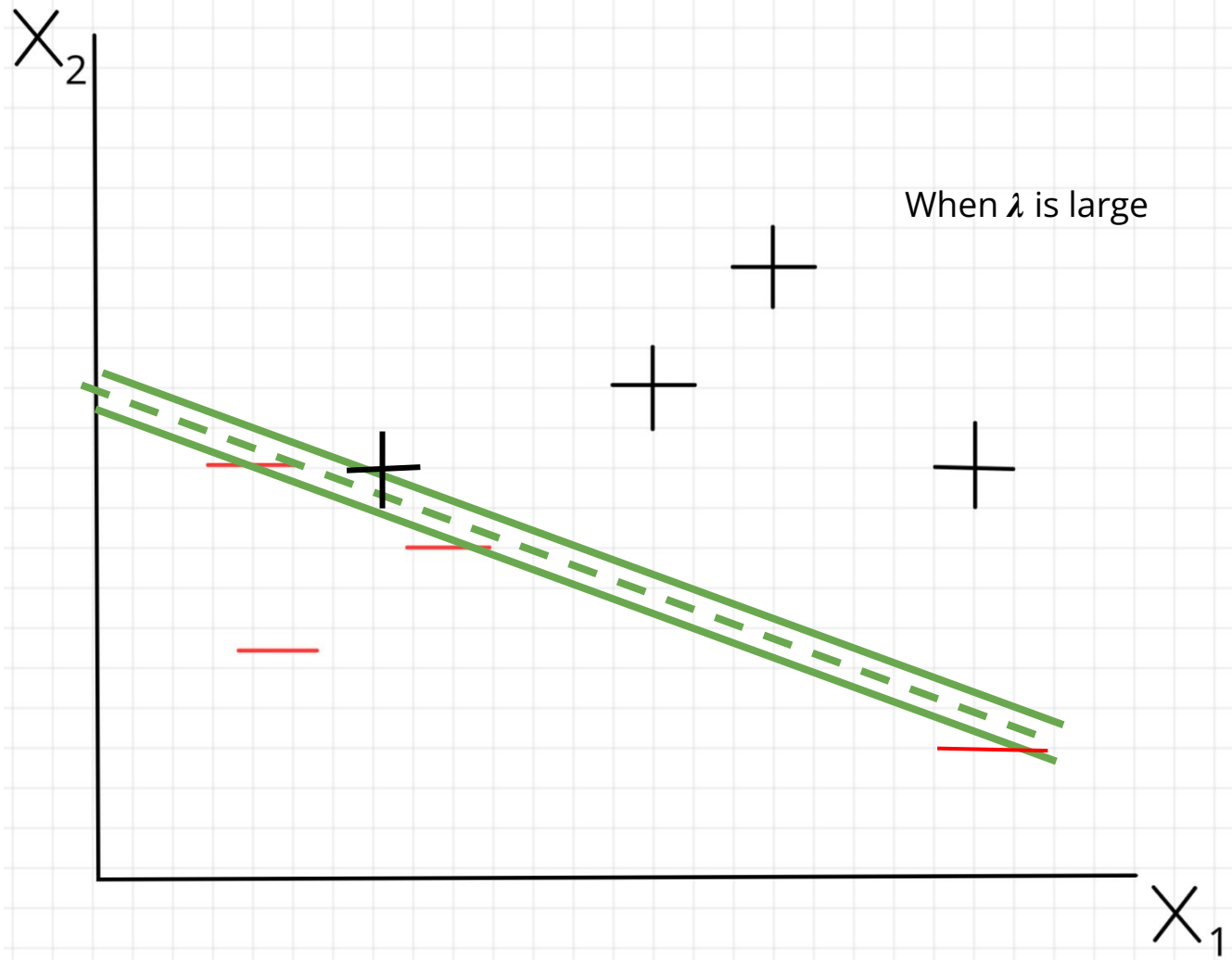
Goal is to maximize the width

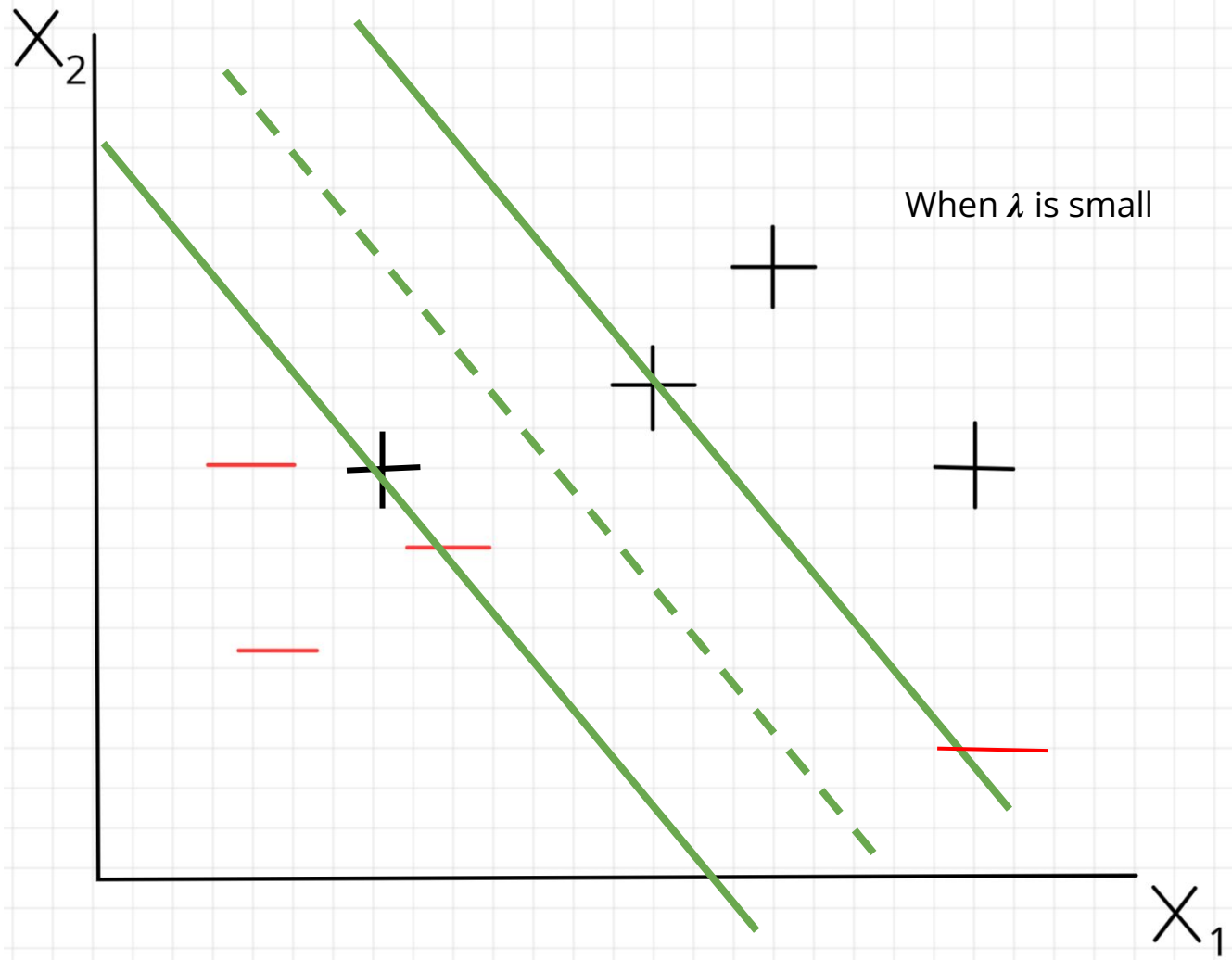
$$\min(\frac{1}{2} \|w\|^2 + \lambda \sum_i \xi_i)$$

Subject to:

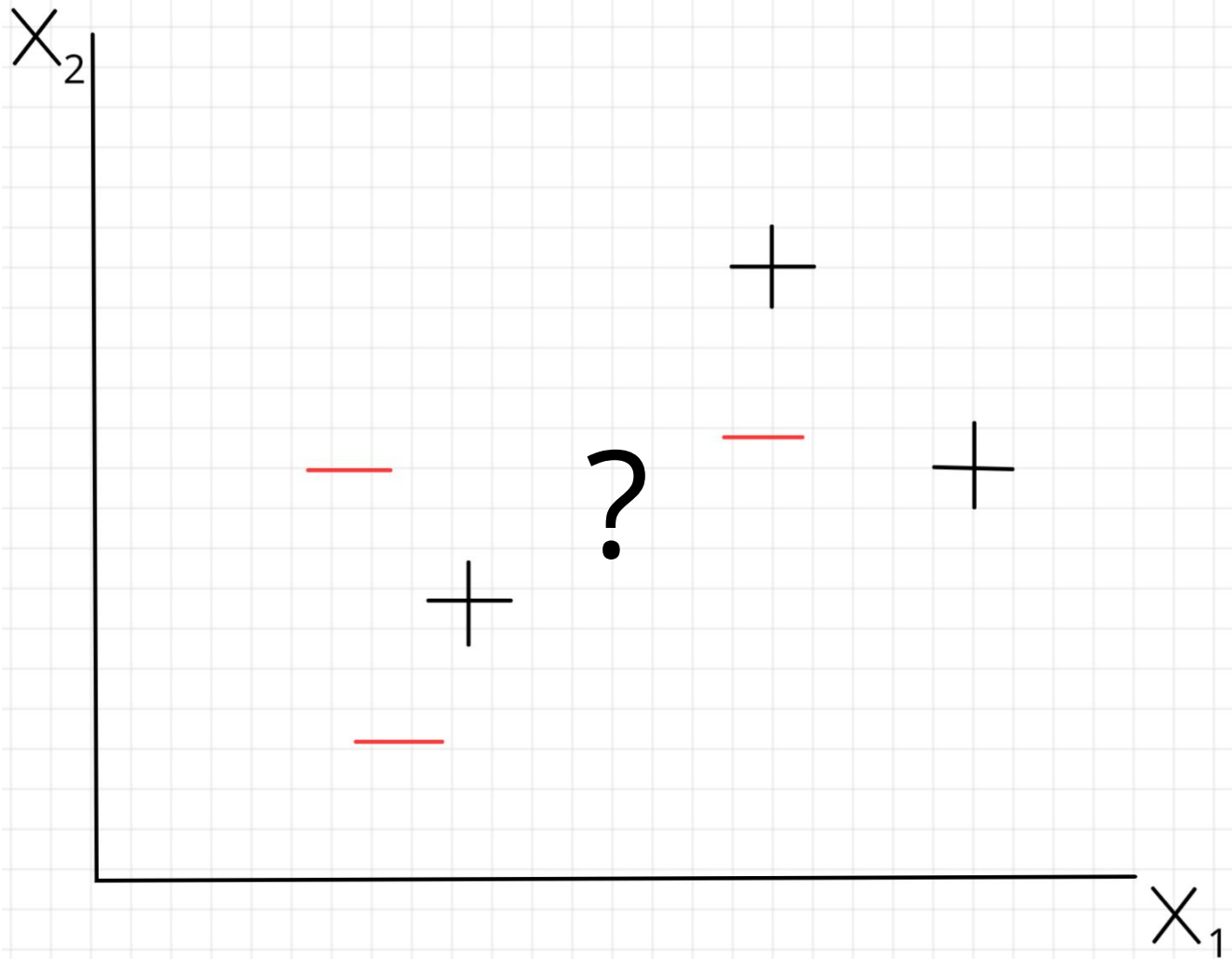
$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i$$





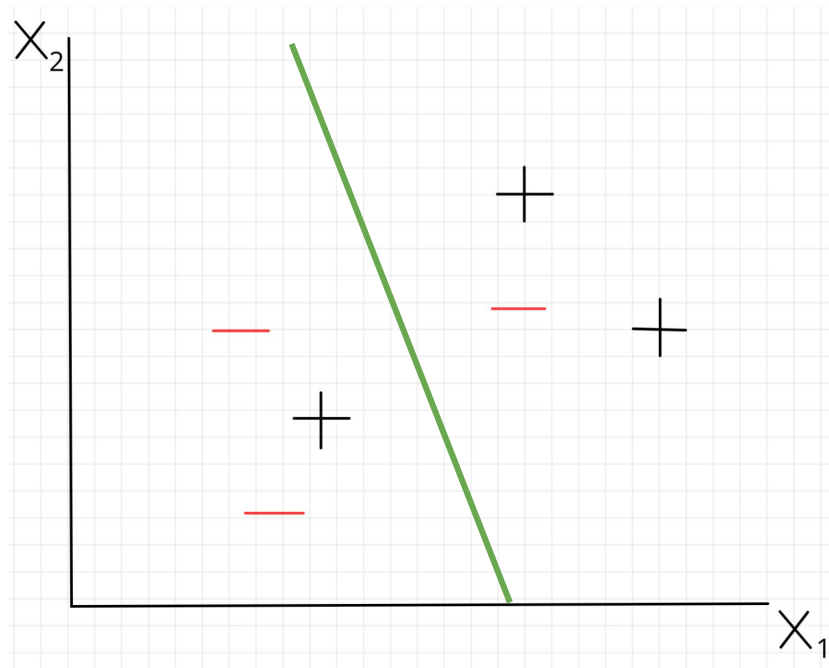


What if there is no line?

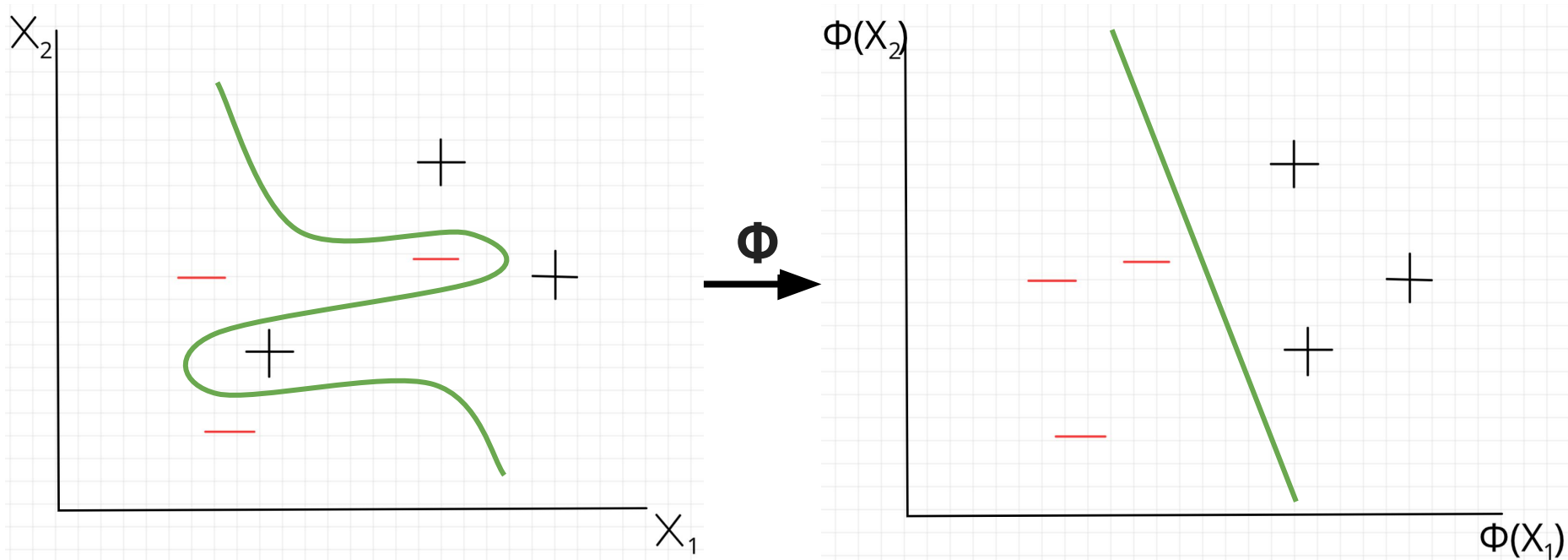


Option 1: Soft Margins

Can allow for some points in the dataset to be misclassified.



Option 2: Change perspective



But how to find Φ ?

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

Recall:

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

But how to find Φ ?

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

Recall:

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

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”.

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

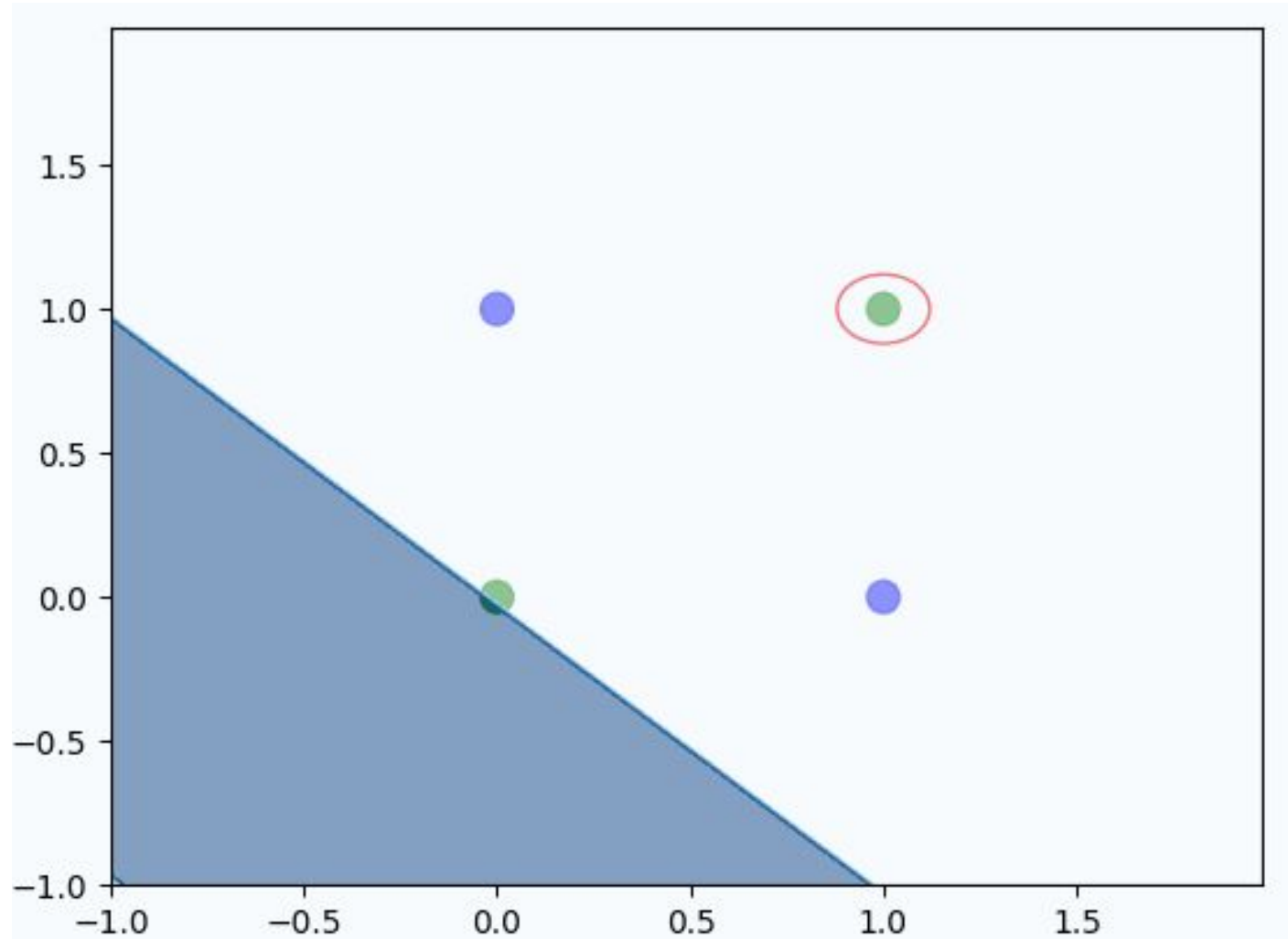
Example Kernel Functions

Polynomial Kernel

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

Radial Basis Function Kernel

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



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/mllearning-ai/support-vector-machines-16241417ee6d>