⇒ Hinge loss :

* To make a SVM, we'll modify the first term of the cost function : $[-\log(h_\theta(x))]$ ~~$[-\log(1+\exp$~~

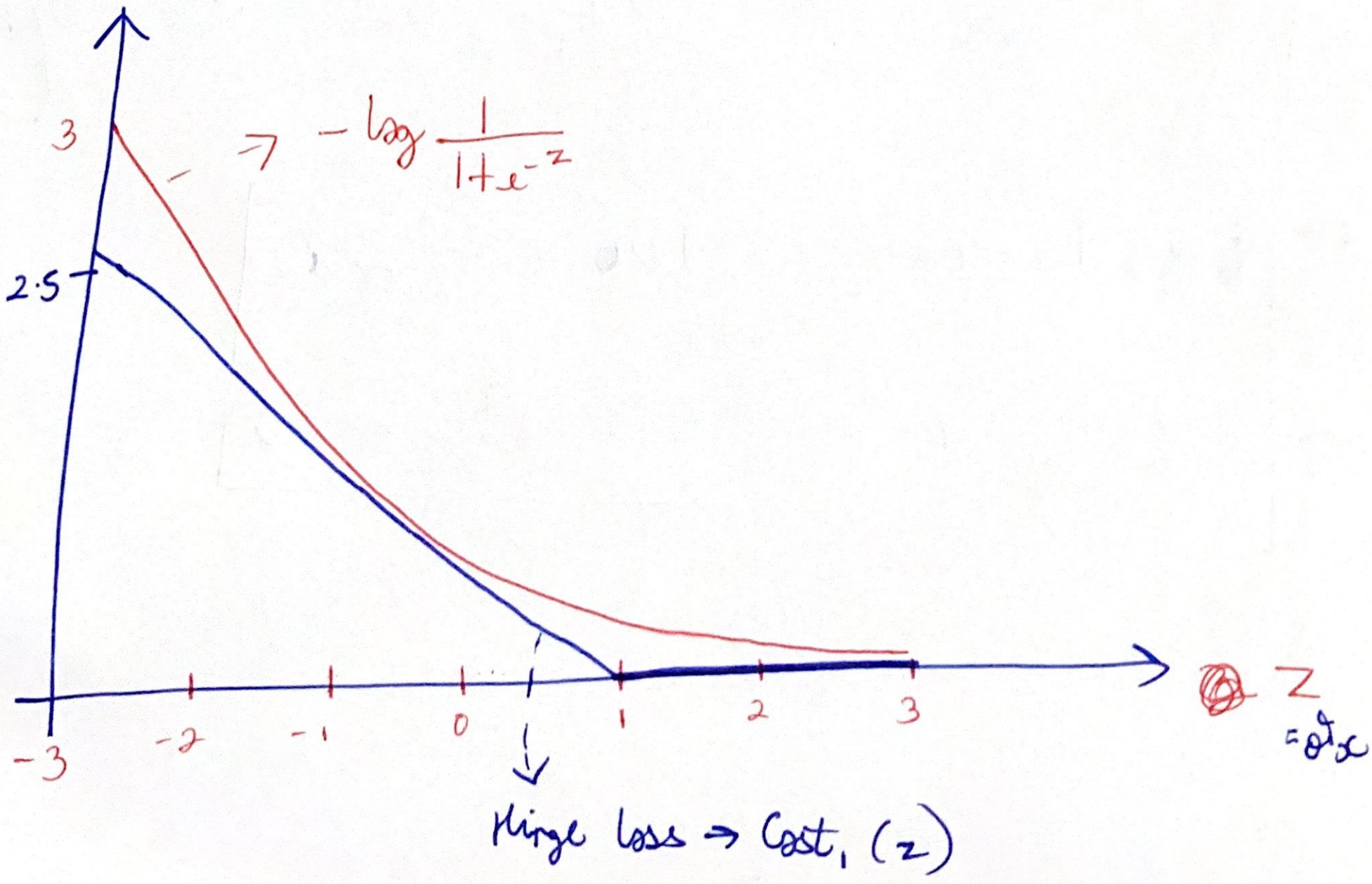
$= \left[-\log\left(\frac{1}{1+\exp(-\theta^T x)}\right)\right]$

{will be referred as z}

so that when $\theta^T x$ is greater than 1, it outputs 0.

* Furthermore, for values of z less than 1, we shall use a straight decreasing line instead of the sigmoid curve.

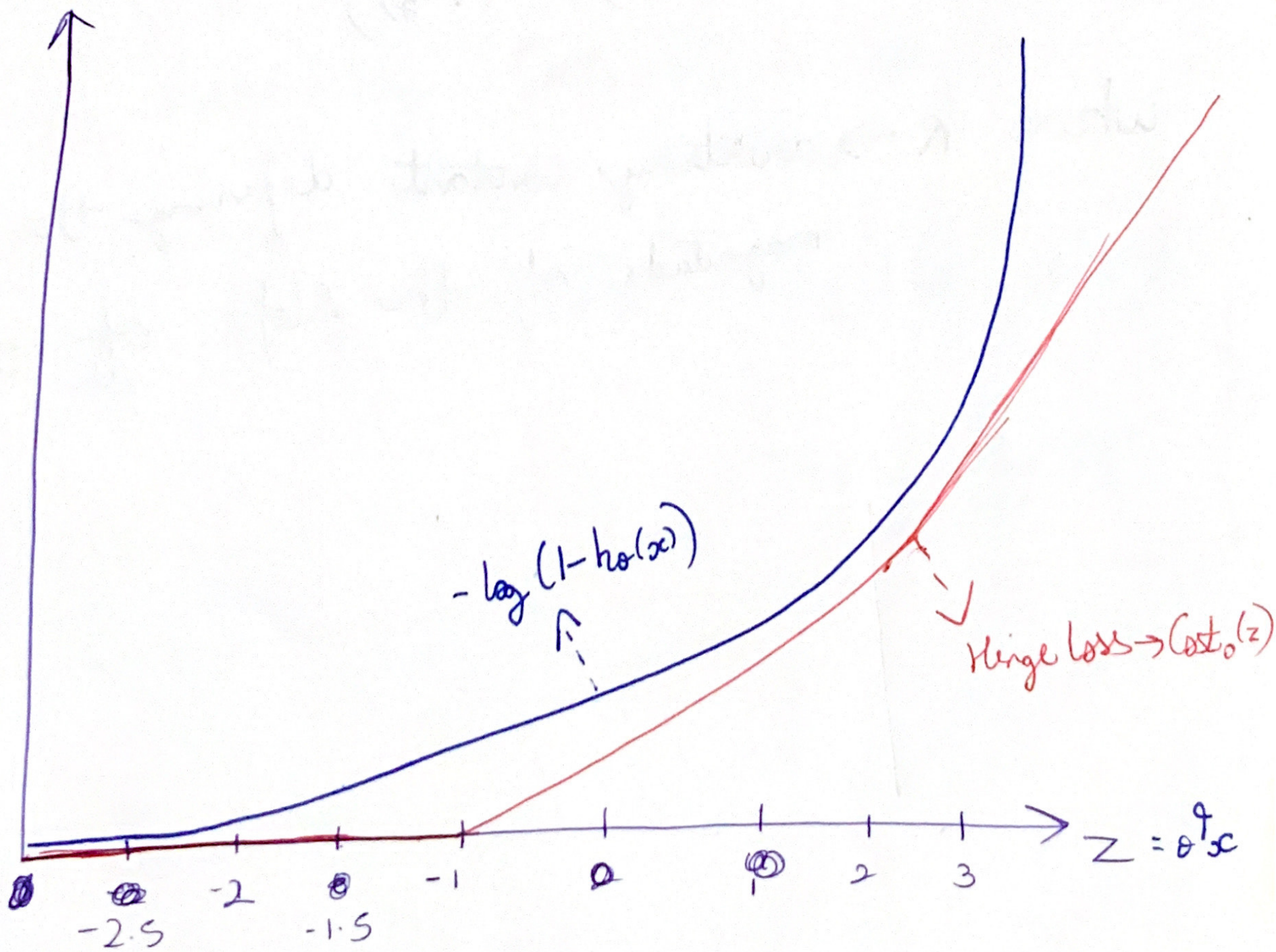
This is called a hinge loss.



* Similarly, we modify the second term of the cost function: $-\log(1-h_\theta(x)) = -\log\left(1-\frac{1}{1+\exp(-\theta^T x)}\right)$

so that when z is less than -1, it outputs 0.

* For values z greater than -1, we use a straight line increasing line instead of the sigmoid curve.



⇒ Week-7

⇒ Optimization objective

⇒ Support Vector Machine (SVM) : Powerful supervised learning algorithm

* Recall, in logistic regression

→ $h_\theta(x) = \frac{1}{1 + \exp(-\theta^T x)}$

→ $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} -y^i \log(h_\theta(x^i)) - (1-y^i) \log(1 - h_\theta(x^i))$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} -y^i \log\left(\frac{1}{1+e^{-\theta^T x^i}}\right) - (1-y^i) \log\left(1 - \frac{1}{1+e^{-\theta^T x^i}}\right)$$

→ if $y = 1$, $h_\theta(x) \approx 1 \Rightarrow \theta^T x \gg 0$

→ if $y = 0$, $h_\theta(x) \approx 0 \Rightarrow \theta^T x \ll 0$

* $Cost_1(z)$: Cost for classifying when $y = 1$

$Cost_0(z)$: Cost for classifying when $y = 0$

→ Formal definition of cost functions

$$z = \theta^T x$$

$$Cost_0(z) = max(0, k(1+z))$$

$$Cost_1(z) = max(0, k(1-z))$$

where $k \rightarrow$ arbitrary constant defining the magnitude of the slope of the line

⇒ SVM notation is _slightly_ different!

→ Getting rid of the $1/m$ terms

* By We can optimize our cost func^n a bit by multiplying it with m. Note, this does not affect our optimization, since we're simply multiplying our cost function by a +ve constant. eg: Minimizing $(u-5)^2+1$ gives 5
& minimizing $10(u-5)^2+10$ also gives 5

$J(\theta)$

$\therefore$ Our SVM cost func^n becomes:

$$J(\theta) = \underbrace{\sum_{i=1}^{m} y^i \, \text{Cost}_1(\theta^T x^i) + (1-y^i)\,\text{Cost}_0(\theta^T x^i)}_{A} + \frac{\lambda}{2}\underbrace{\sum_{j=1}^{n} \theta_j^2}_{B}$$

→ Using ~~@~~ ~~of~~ $A+B$ instead of $A+\lambda B$

* For SVMs, the convention is to use a different parameter C, instead of λ

$$J(\theta) = C \sum_{i=1}^{m} y^i \text{cost}_1(\theta^T x^i) + (1-y^i)\,\text{cost}_0(\theta^T x^i) + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

* This is equivalent to multiplying the eqn by $C = \frac{1}{\lambda}$, & thus results in the same values when optimized. {Using the arguments discussed earlier for C}

→ Now, when we wish to regularize more (i.e. reduce overfitting), we decrease C (indirectly increasing λ)

& when we wish to regularize less (i.e. reduce underfitting), we increase C (indirectly decreasing λ)

⇒ Hypothesis of SVM

* Unlike logistic, $h_\theta(x)$ doesn't give us a probability (like logistic regression) but instead we get a direct prediction of 1 or 0!!

In technical terms, it is a discriminant function.

$$h_\theta(x) = \begin{cases} 1, & \text{if } \theta^T x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

→ We can now finally state our optimization objective:

Minimize $J(\theta)$ cost func^n for SVM:

$$\min_\theta J(\theta) = \min \left\{ C \sum_{i=1}^{m} y^i \, \text{Cost}_1(\theta^T x^i) + (1-y^i)\,\text{Cost}_0(\theta^T x^i) + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 \right\}$$

→ Recall for the ~~fuck~~ cost regularized cost function for logistic regression is

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} y^i \left(-\log\left(h_\theta(x^i)\right)\right) + \left(1-y^i\right)\left(-\log\left(1-h_\theta(x^i)\right)\right) + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

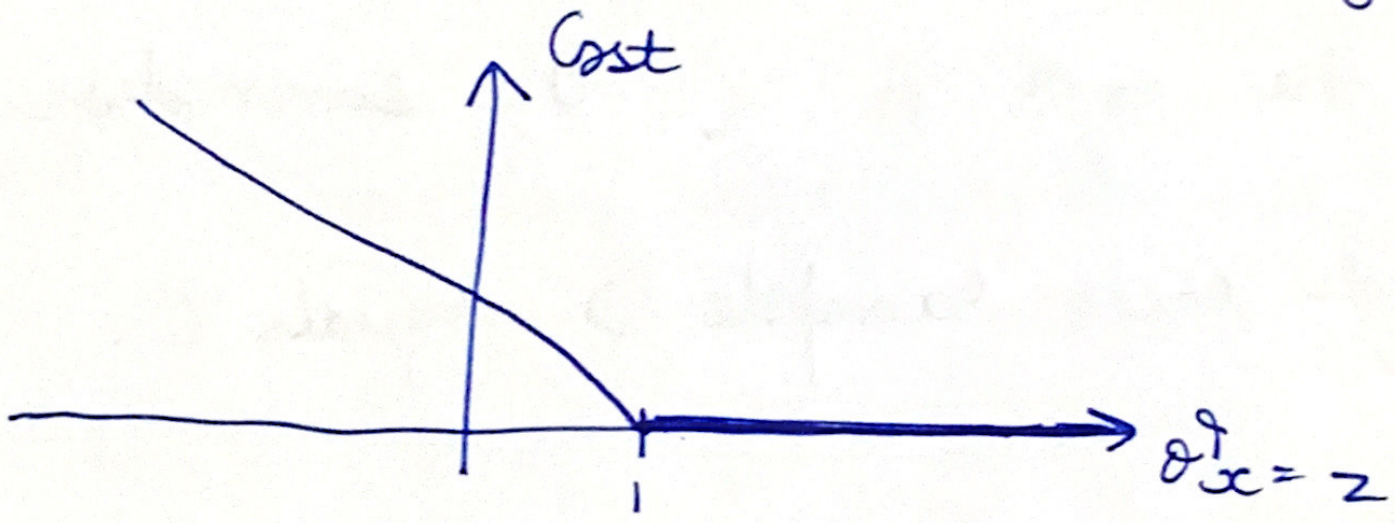
We can transform this into the cost function for SVMs by substituting $\text{Cost}_0(z)$ & $\text{Cost}_1(z)$:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} y^i \text{Cost}_1\left(\theta^T x^i\right) + \left(1-y^i\right) \text{Cost}_0\left(\theta^T x^i\right) + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

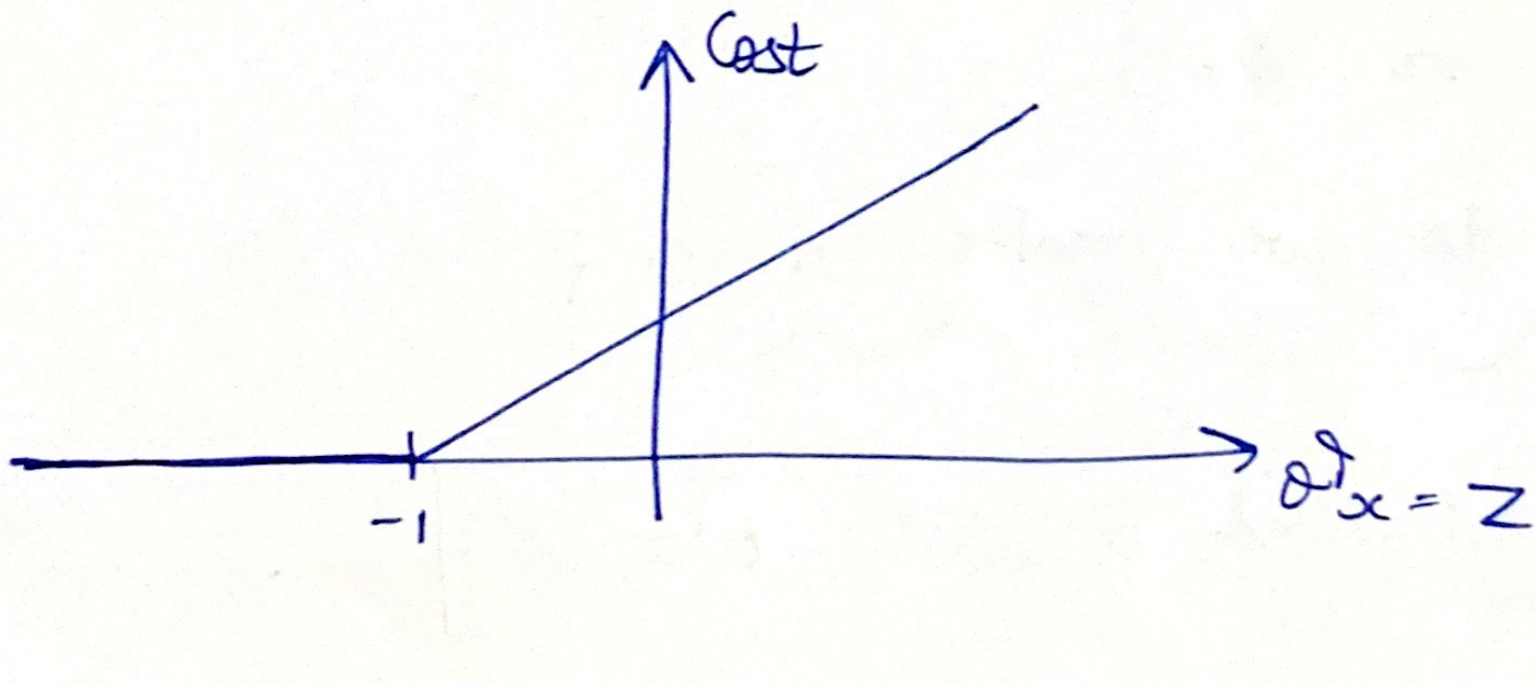
⇒ Large margin intuition

* A useful way to think about SVMs is to think of them as Large Margin Classifiers.

* If $y = 1$, we want $\theta^T x \geq 1$ (not just ≥ 0)



* If $y = 0$, we want $\theta^T x \leq -1$ (not just ≤ 0)



* So, in addition to what logistic regression does, a SVM also throws an extra safety margin factor. Hence, the name large margin classifier.

~~What happens when we have value of C~~

⇒ What happens when we give C a really huge value?

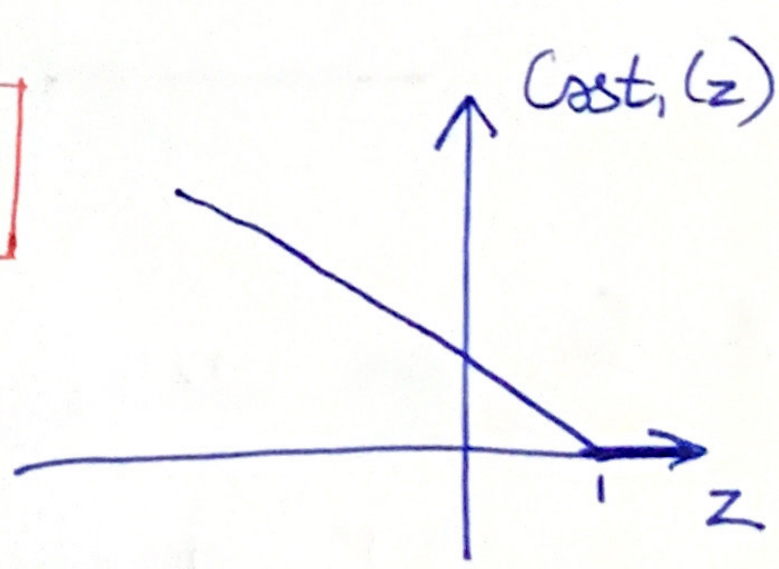
* When we give C a very large value (eg. 100 000), our optimizing function will constrain θ such that the eqn A (the summation of the cost for each example) equals 0.

$$\Rightarrow \sum_{i=1}^{m} \left[y^i \, \text{Cost}_1(\theta^T x^i) + (1 - y^i) \, \text{Cost}_0(\theta^T x^i) \right] = 0$$

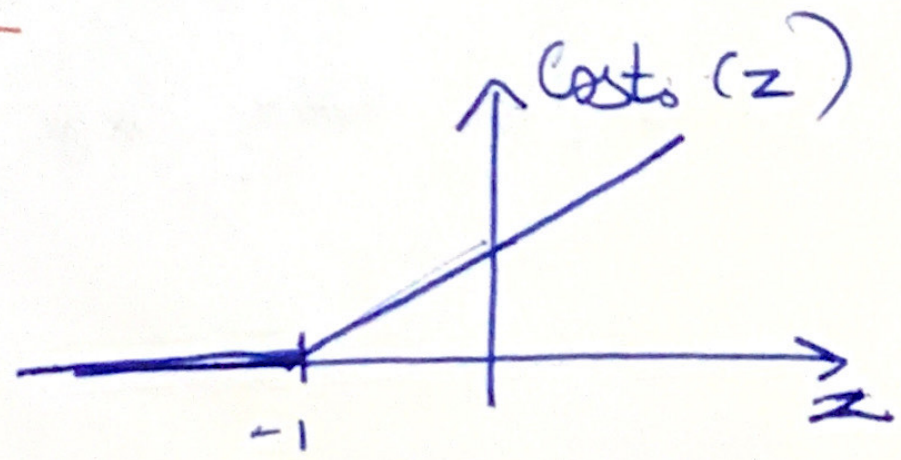
or $A = 0$

* How do we make $A = 0$?

→ If $y = 1$: Find θ s.t. $\theta^T x \geq 1$



→ If $y = 0$: Find θ s.t. $\theta^T x \leq -1$



* This reduces our cost function to :

$$J(\theta) = C.0 + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

$$\therefore \boxed{J(\theta) = \frac{1}{2} \sum_{j=1}^{n} \theta_j^2}$$

* Then, our optimization objective becomes,

$$\min_\theta J(\theta) \Rightarrow \min_\theta \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

$$\text{s.t.} \quad \theta^T x^i \geq 1 \quad \forall \; y^i = 1$$

$$\& \; \theta^T x^i \leq -1 \quad \forall \; y^i = 0$$

** In logistic regression, we take the output of the linear function & squash the value within the range of $[0, 1]$ using sigmoid function.

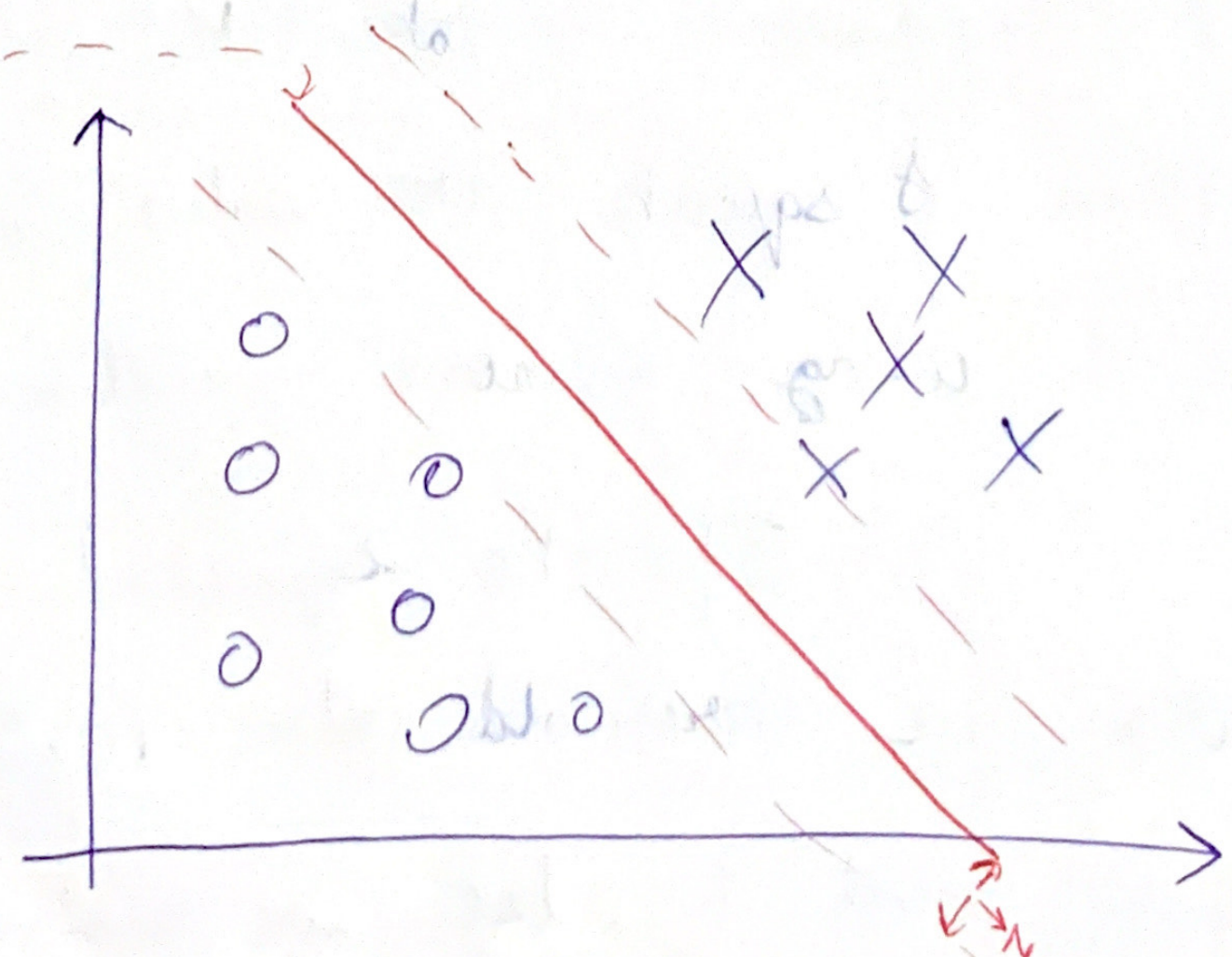
If the squashed value is greater than the threshold value (0.5) we assign it a label 1 else we assign it a label 0.

** In SVM, we take the output of the linear function & if that output is greater than 1, we identify it with one class & if the output is -1, we identify it with another class.

Since we have a larger (or wider) range now from [-1, 1] which acts as a margin, SVM are called large margin classifiers.

* The SVM will separate the negative & positive examples by a large margin.

SVM decision boundary

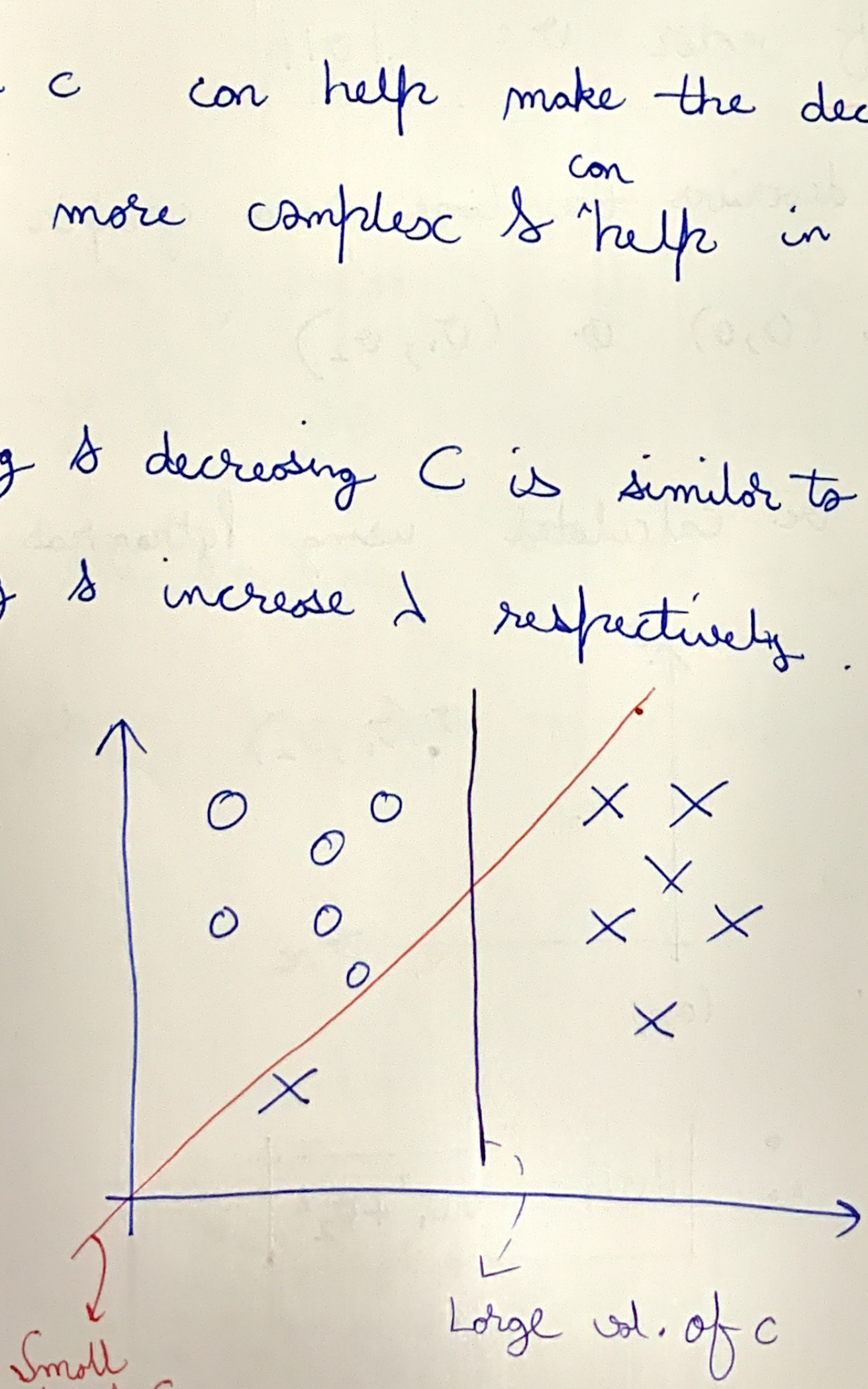


* The large margin is only achieved when C is very large & & when your data is 'linearly seperable' with no outliers.

* Data is linearly seperable when a straight line can seperate the +ve & -ve sample.

* Reducing C can help make the decision boundary more complex & can help in covering outliers.

* Increasing & decreasing C is similar to decreasing & increase λ respectively.



⇒ Vector inner product

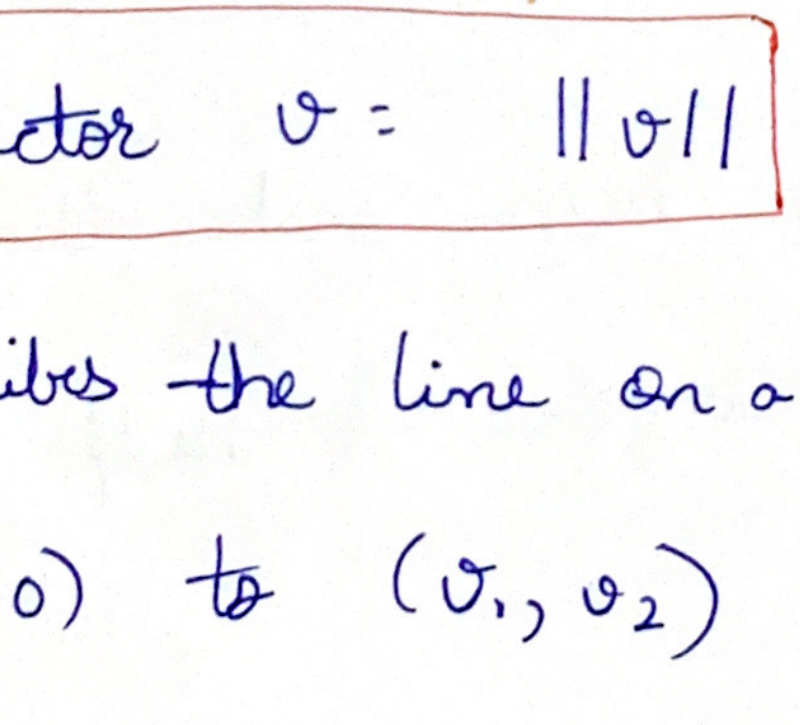
→ Say we have 2 vectors, u & v:

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

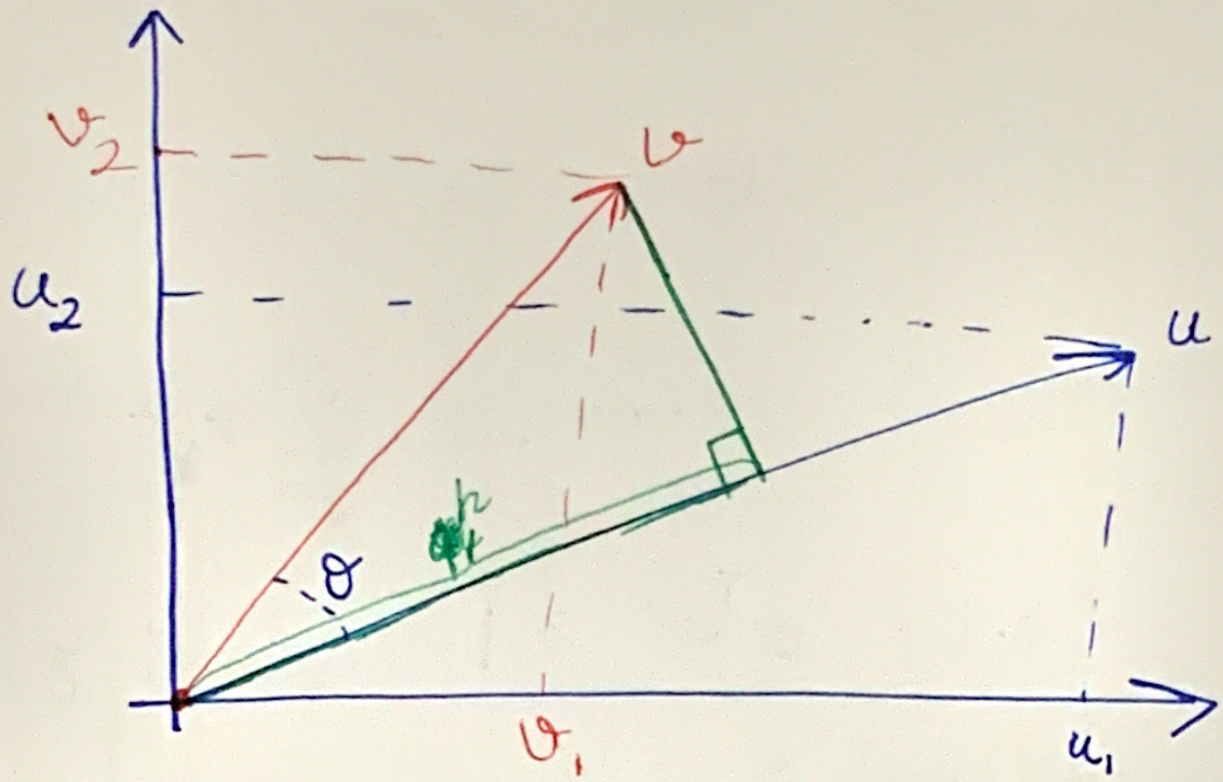
→ **Length of vector v = $\|v\|$**

* Also describes the line on a graph from origin $(0,0)$ to (v_1, v_2)

* Can be calculated using Pythagoras theorem:



$$\therefore \|v\| = \sqrt{v_1^2 + v_2^2}$$



$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

* The projection of vector v onto vector u is found by taking a right angle from u to the end of v, creating a right triangle.

* p = Length of projection of v onto the vector u

* $\vec{u} \cdot \vec{v} = u_1 v_1 + u_2 v_2 = u^T v$; where $u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ $v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$

Also $\vec{u} \cdot \vec{v} = \|u\| \|v\| \cos\theta$

$= \|u\| (p)$

$\therefore$ $\boxed{\vec{u} \cdot \vec{v} = p \|u\|}$

greater than 90°; the p will be −ve.

⇒ SVM decision boundary : $\min \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$

$$\text{s.t.} \quad \theta^T x^i \geq 1 \quad \text{if } y^i = 1$$

$$\& \quad \theta^T x^i \leq -1 \quad \text{if } y^i = 0$$

* For convience of notation, we'll settle 2 simplifications:

1) Set $\theta_0 = 0$ ⇒ Hyperplane will only pass through origin, i.e. no intercept term

2) Set $n = 2$ → $(x^1, y^1), (x^2, y^2)$

2) Set $n=2$, i.e. each example only has 2 features

→ Given we only have 2 parameters we can simplify our function to

$$\frac{1}{2}(\theta_1^2 + \theta_2^2) = \frac{1}{2}\left(\sqrt{\theta_1^2 + \theta_2^2}\right)^2 = \frac{1}{2}(\|\theta\|)^2$$

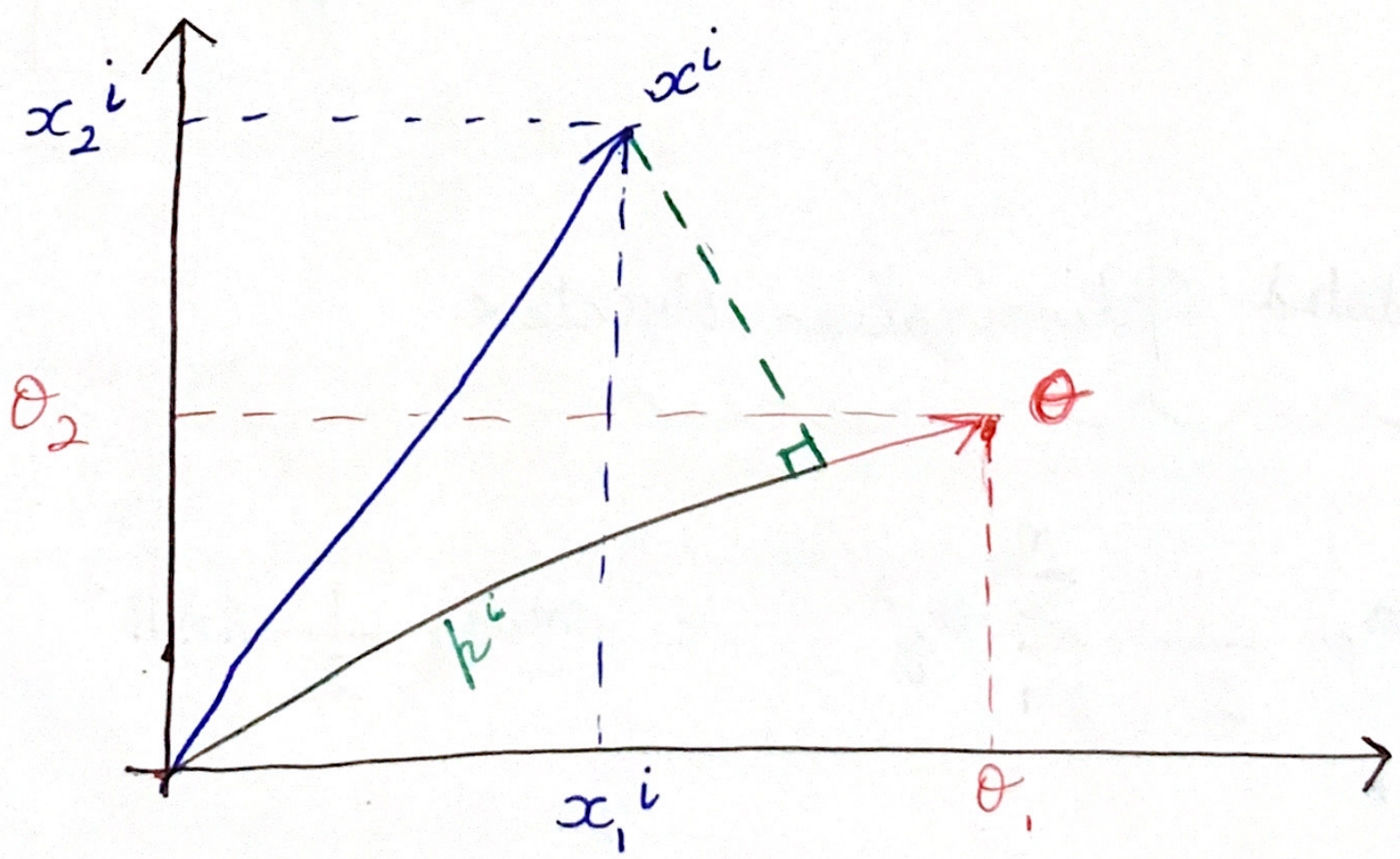
* So, the SVM is trying to minimize the squared length of the vector θ.

→ Given this, what are the $\theta^T x$ parameters doing?

* For understanding this, we can take an example $x^i \equiv (x_1^i, x_2^i)$. And suppose

$$x^i = \begin{bmatrix} x_1^i \\ x_2^i \end{bmatrix} \quad \& \quad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$$\therefore \theta^T x^i = p^i \|\theta\| = \theta_1 x_1^i + \theta_2 x_2^i$$



... we can modify our eqⁿs:

$$\theta^T x^i \geq 1 \quad \text{if } y^i = 1$$

$$\theta^T x^i \leq -1 \quad \text{if } y^i = 0$$

to

$$p^i \, \|\theta\| \geq 1 \quad \text{if } y^i = 1$$

$$p^i \, \|\theta\| \leq -1 \quad \text{if } y^i = 0$$

→ Updated Optimization objective

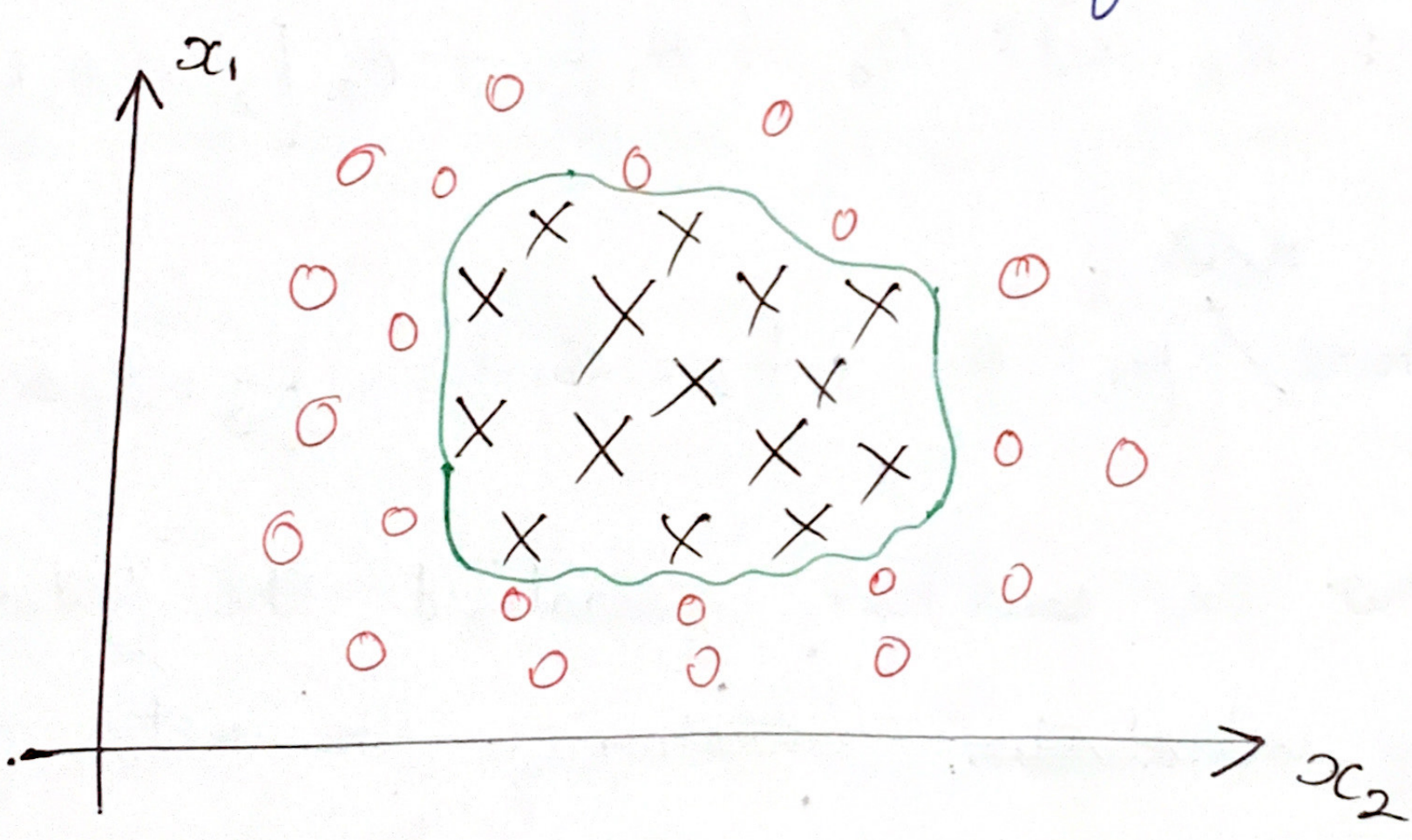
$$\min_\theta \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 = \min_\theta \frac{1}{2} \|\theta\|^2$$

s.t. $p^i . \|\theta\| \geq 1$ if $y^i = 1$

$p^i . \|\theta\| \leq -1$ if $y^i = 0$

⇒ Kernels : This technique is used to adapat support vector machines in order to develop complex non-linear classifiers.

⇒ Supp. we have a training set with a non linear decision boundary like the fll. :



is

* One way of doing this is by introducing higher order (polynomial terms) in while creating our hypothesis function, i.e.

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \ldots$$

Predict $y = 1$ if $h_\theta(x) \geq 0$ & $y = 0$ otherwise

→ Another way of writing this (Vector notation) is by taking the dot product of the θ vector by a new feature vector f, which simply contains the various high order x terms.

eg: $$h_\theta(x) = \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \dots$$

where $$f_1 = x_1 \,,\; f_2 = x_1 x_2 \,,\; f_3 = x_1 x_2 \,, \dots$$

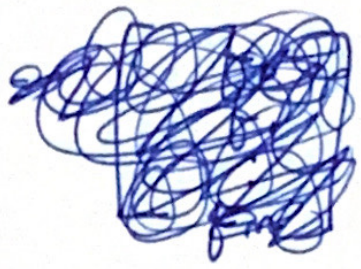
→ Now we have generalized the feature vector by introducing the vector f!! This will come in handy later.

→ Ques: Is there a different/better choice of the features $f_1, f_2, f_3, \dots$ other than higher polynomial terms?

* For images, higher order polynomials can become computationally expensive because of presence of large amount of high

⇒ We'll use kernels!

→ Idea : Given an input vector $x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_m \end{bmatrix}$,

we'll choose new feature vector f 

depending upon the "similarity" b/w x & some manually chosen points known as landmarks (denoted by $l^1, l^2, \ldots$)

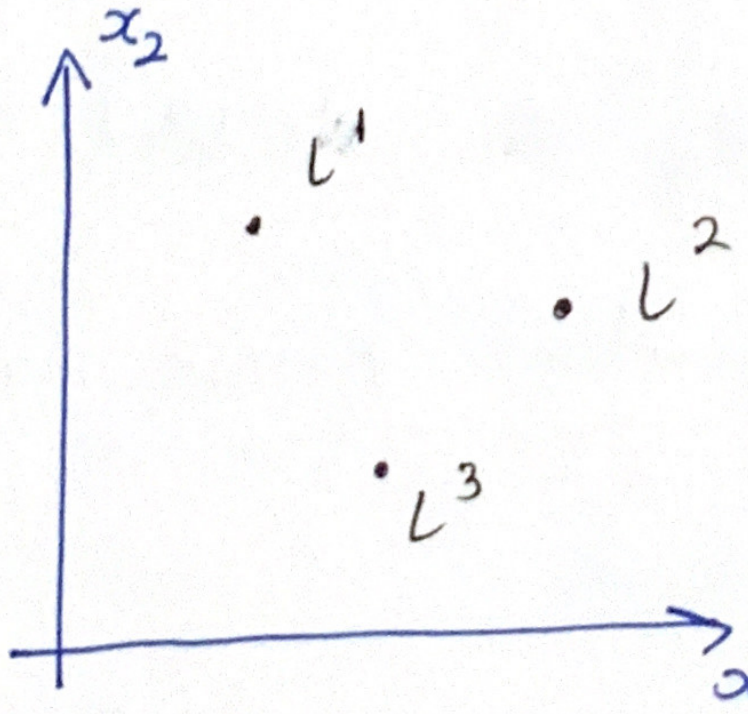
* This "similarity" is calculated using a similarity function known as kernel or kernel function

⇒ eg: Suppose we have $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

* We want to define 3 features in this example.

* Have a graph of x_1 vs x_2 (just define the space, don't graph anything)

* Since we want 3 features, manually pick 3 points in that space. These points l^1, l^2, l^3 were chosen manually & are called landmarks.



* Given x, define f_i as the similarity b/w x & l^i.

$$\Rightarrow \boxed{f_i = \text{similarity}(x, l^i) = \exp\left(-\frac{\|x - l^i\|^2}{2\sigma^2}\right)}$$

* $\|x - l^i\|$ denotes the euclidean distance b/w the vector (or point in 2-D) x & the vector l^i.

* $\sigma \rightarrow$ Standard deviation

  $\sigma^2 \rightarrow$ Variance

* Also, $$\|x - l^i\|^2 = \sum_{j=1}^{n} (x_j - l_j^i)^2$$

* Instead of writing "similarity" b/w x & l^i, we might write $f_i = k(x, l^i)$

$\therefore$ $$f_1 = k(x, l^1) = \exp\left(-\frac{\|x - l^1\|^2}{2\sigma^2}\right)$$

Similar for f_2 & f_3

Kernels & similarity :

→ Let see what these "kernels" & actualy do

→ $f_i = k(x, l^i) = \exp\left(-\frac{\|x - l^i\|^2}{2\sigma^2}\right)$

$= \exp\left(-\frac{\sum_{j=1}^{n} (x_j - l_j^i)^2}{2\sigma^2}\right)$

→ If $x \approx l^i$: $f_i \approx \exp\left(\frac{-0^2}{2\sigma^2}\right) \approx 1$

→ If x is for from l^i : $f_i \approx \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right)$

≈ 0

→ In other words, if x & the landmark are close, then the similarity will be close to 1, & if x & the landmark are for away from each other, the similarity will be close to 0.