AI Theory

Review gradient descent

1- initialize w

2- for epochs

loss

gradient

update

generic weights

more formally, the algorithm is stated as:

GD(w, Data):

for i := 1 to epochs:

$$F(w) = \frac{1}{M} \sum_{i=1}^{|Data|} (w \cdot x - y_i)^2$$

$$\frac{\partial F}{\partial w} = 2x\,(wx - y_i)$$

$$w = w - \alpha \frac{\partial F}{\partial w}$$

Stochastic Gradient Descent:

Update step 2 of GD as:

for epochs:

<u>update</u>

for x,y in Data:

Loss

Gradient

update w

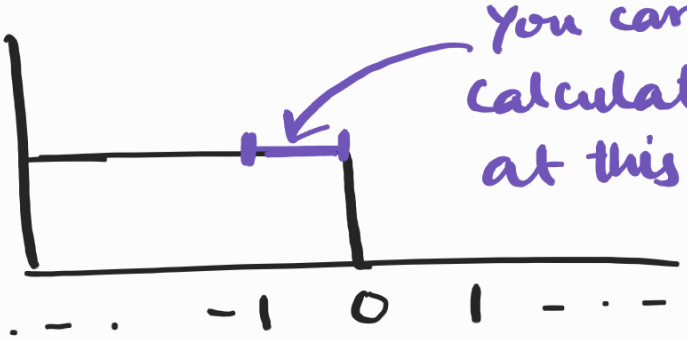
What should be the step size α ?

- Constant $\alpha = 0.1$
- Decreasing $\alpha = \dfrac{1}{\sqrt{\#\text{ updates made so far}}}$

Zero-one loss :

$$\text{Loss}_{0-1}(x, y, w) = \mathbb{1}\left[w(x)y\right] \leq 0$$

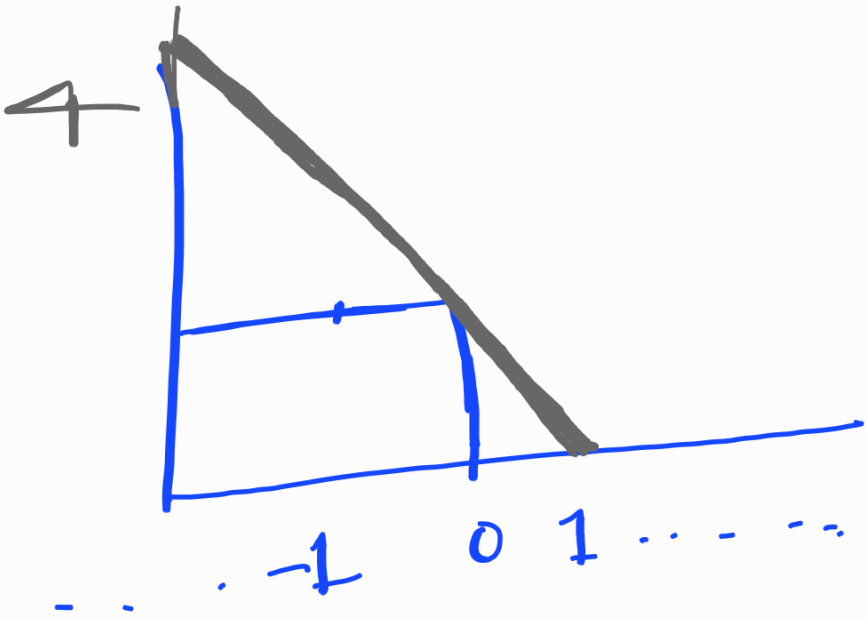
$$\frac{\partial \text{loss}}{\partial w} = \phi(x)y$$



You can only calculate gradient at this point

So in binary classification, zero-one loss does not work for gradient descent.

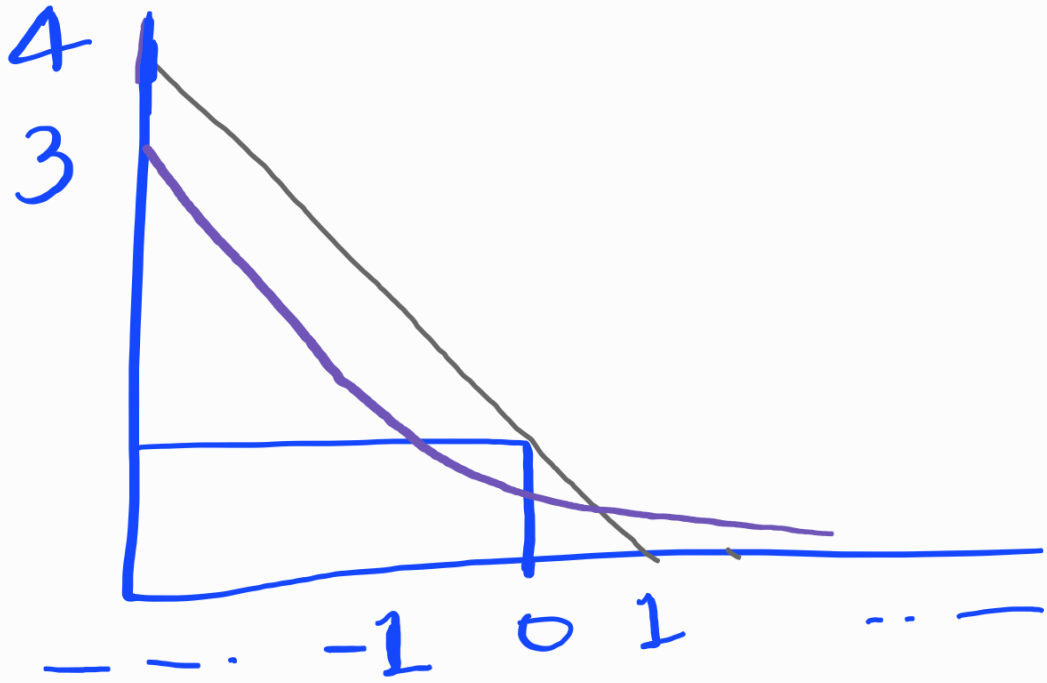
Solution: Introduce a new type of loss.



Hinge loss = 1 − margin

$$\boxed{\text{Hinge loss} = 1 - w(\phi)x \cdot y}$$

Logistic Regression :

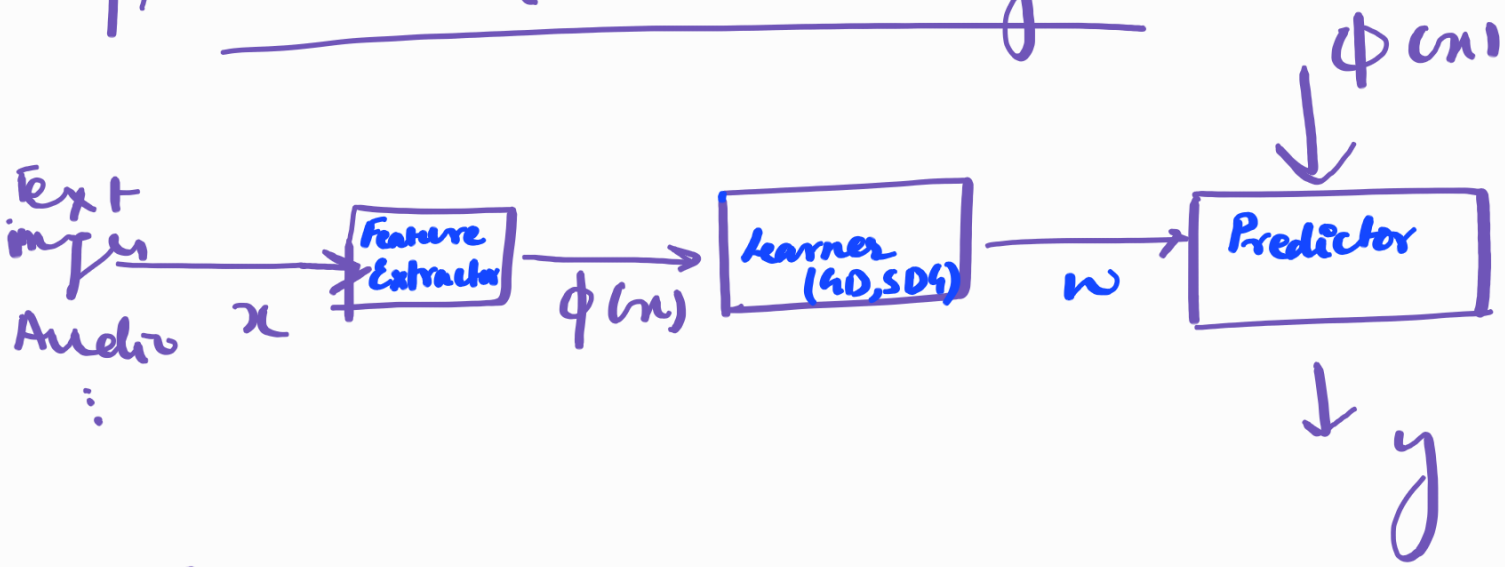


$$\text{loss}_{\text{logistic}} = \log\left(1 + \frac{1}{e^{wx+y}}\right)$$

↳ is almost same as hinge loss.

↳ but is more smooth than hinge loss.

Traditional ML cycle:



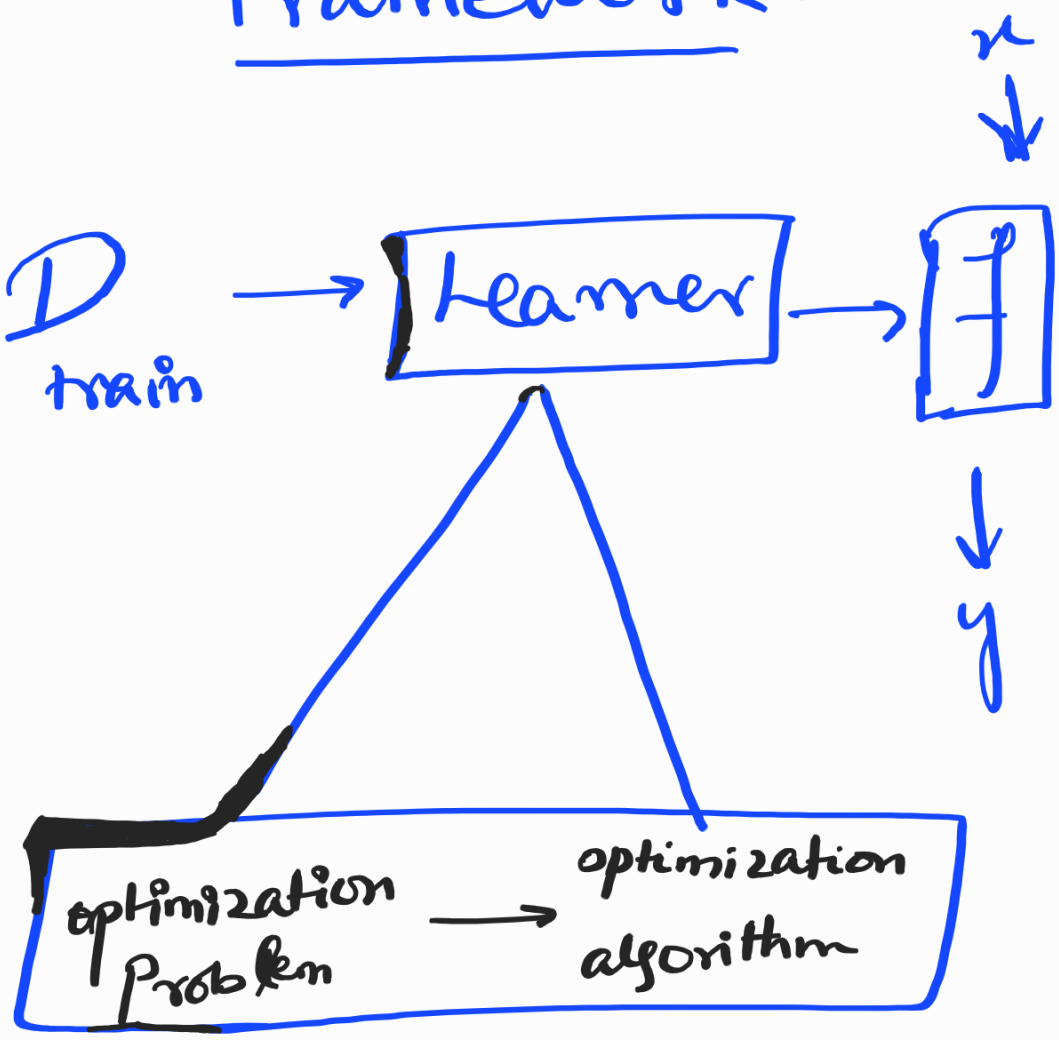
for x, y :

gradient descent works in this manner.

for x_1, x_2, y:

what should we do?

Framework:



A Regression Example

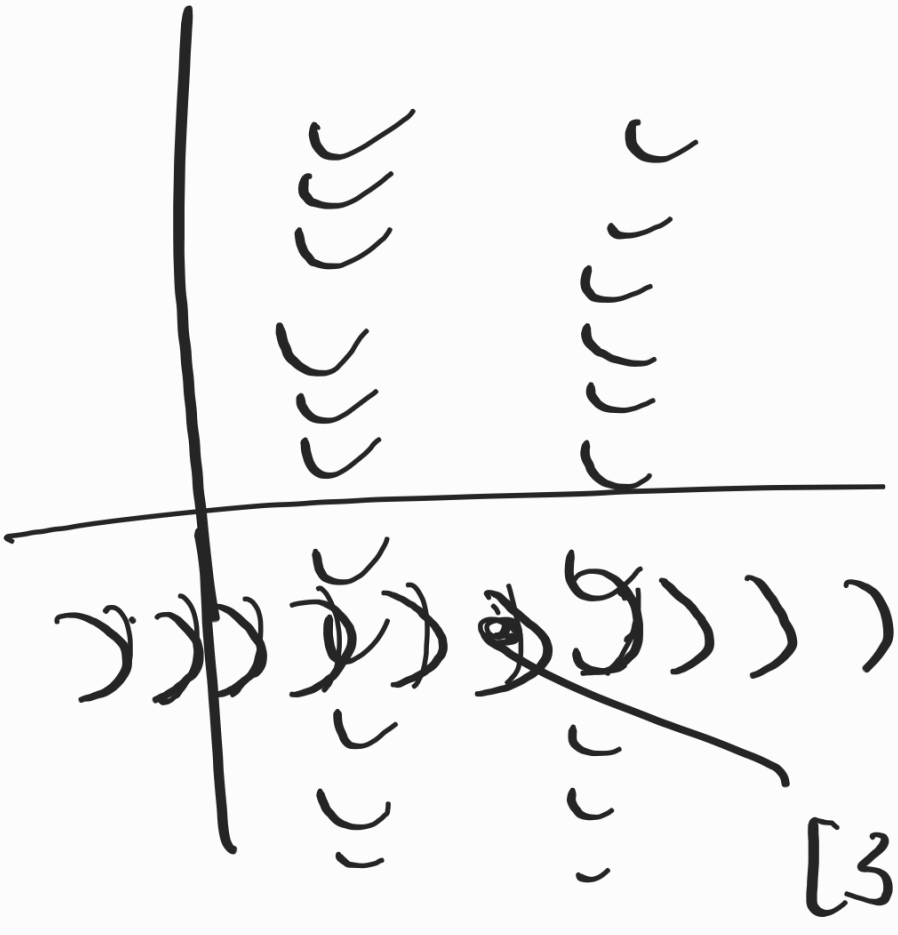
x	y	$\text{Loss} = (wx - y)^2$
$(1,0)$	2	$(w_1 - 2)^2$
$(1,0)$	4	$(w_1 - 4)^2$
$(0,1)$	-1	$(w_2 + 1)^2$

$$\text{Train loss}(w) = \frac{1}{3}\left\{(w_1-2)^2+(w_1-4)^2+(w_2+1)^2\right\}$$

$$= \frac{1}{3}\left\{w_1^2+4-4w_1 + w_1^2+16+8w_1 + w_2^2+1+2w_2\right\}$$

$$= \frac{1}{3}\left\{2w_1^2+20+4w_1+w_2^2+1+2w_2\right\}$$

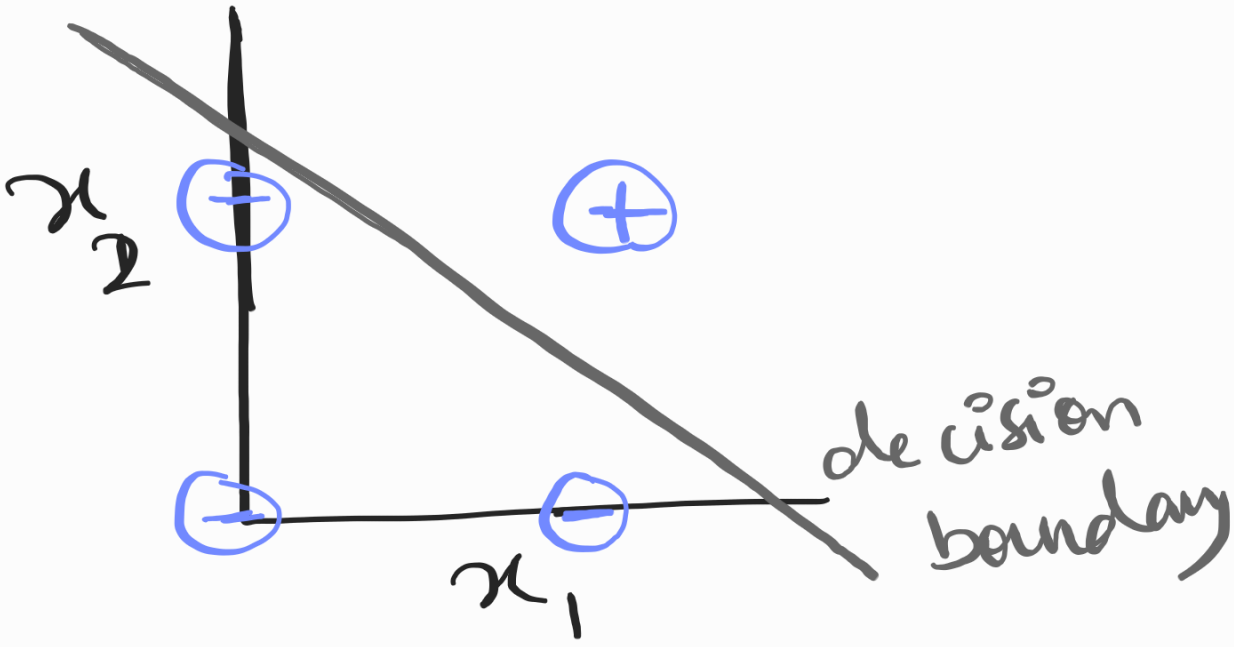
$$\text{Train loss} = \frac{1}{3}\left\{2w_1^2+w_2^2+4w_1+2w_2+21\right\}$$



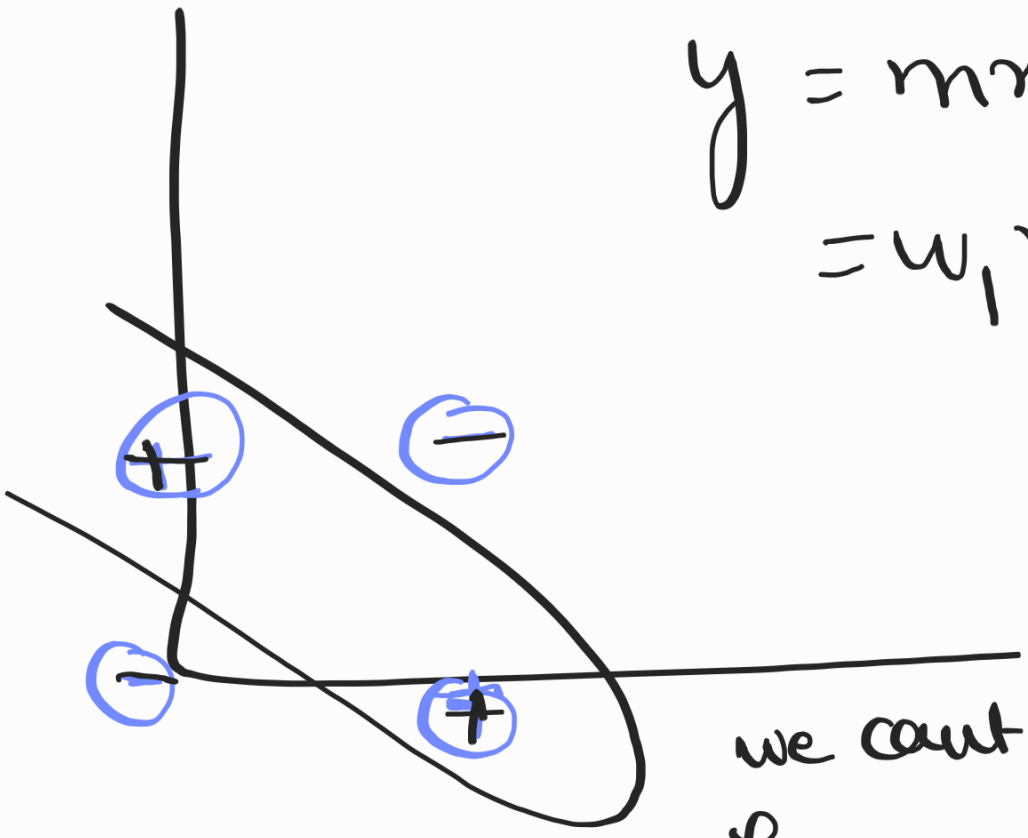
$[3, -1]$

Class Task: for given points, draw a decision boundary:

x_1	x_2	y
1	0	-1
0	1	-1
1	1	1
0	0	-1



x_1	x_2	y
0	0	-1
1	0	+1
0	1	+1
1	1	-1

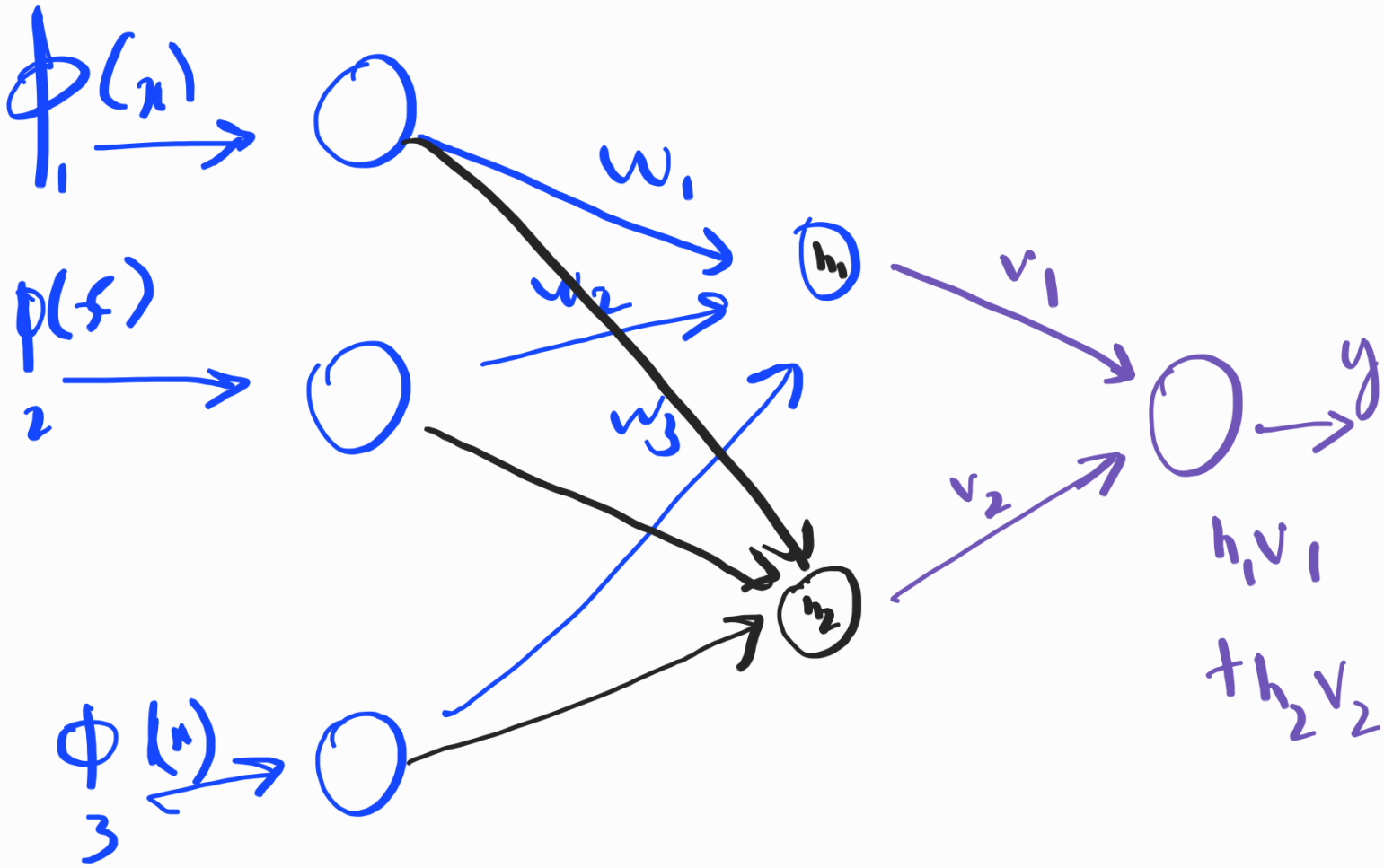


$$y = mx + c$$

$$= w_1 x_1 + w_2 x_2 + c$$

we cant use linear for non-linear decisions

Neural Networks



explanation of this diagram