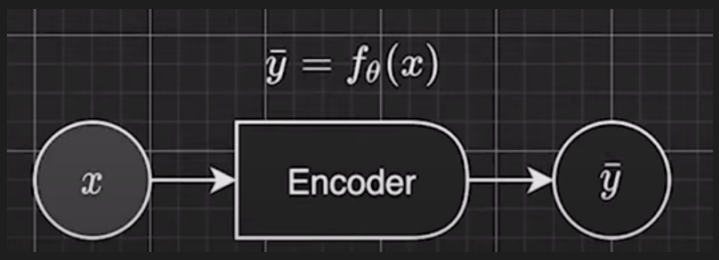
Forward Propagation



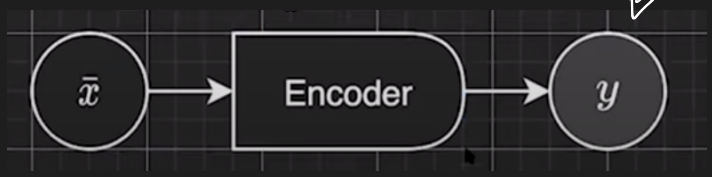

given (observation)

Inference: Compute something given something else

optimization problem

$$\bar{x} = \arg\min_x((y - f_\theta(x))^2)$$

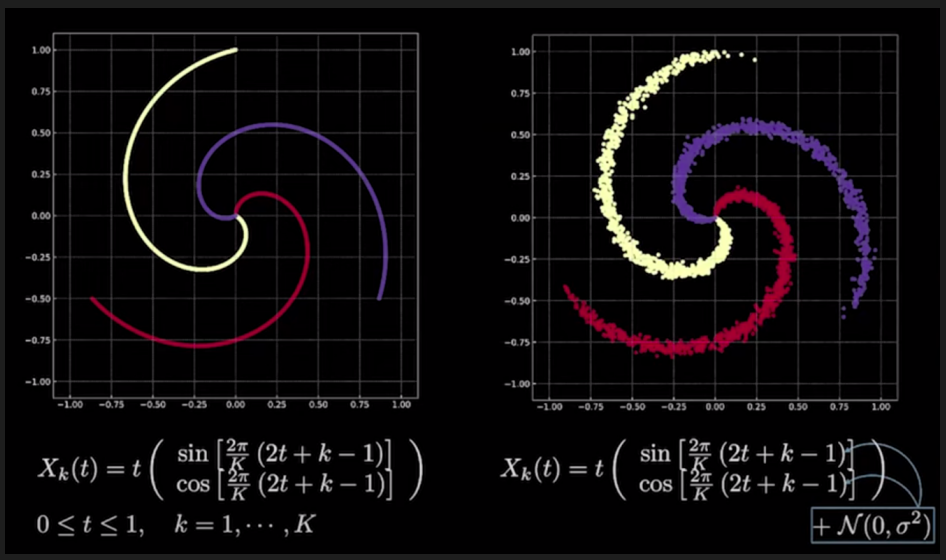
given (observation)




Q: "Is backprop only used for training?"

A: No! in the previous case backprop is used for inference.

# Supervised Learning.




- 3 classes

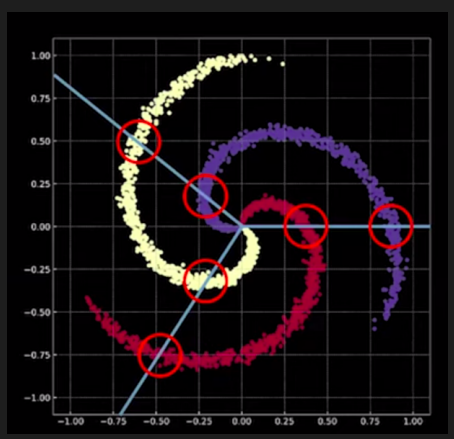
Input:

- 2d data points

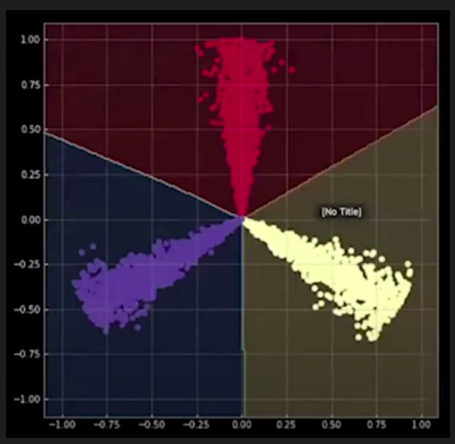
Output:

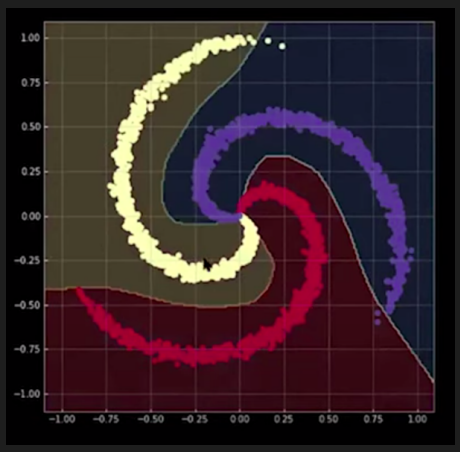
- Which class

A linear model is not enough



# Two ways of seeing the boundaries

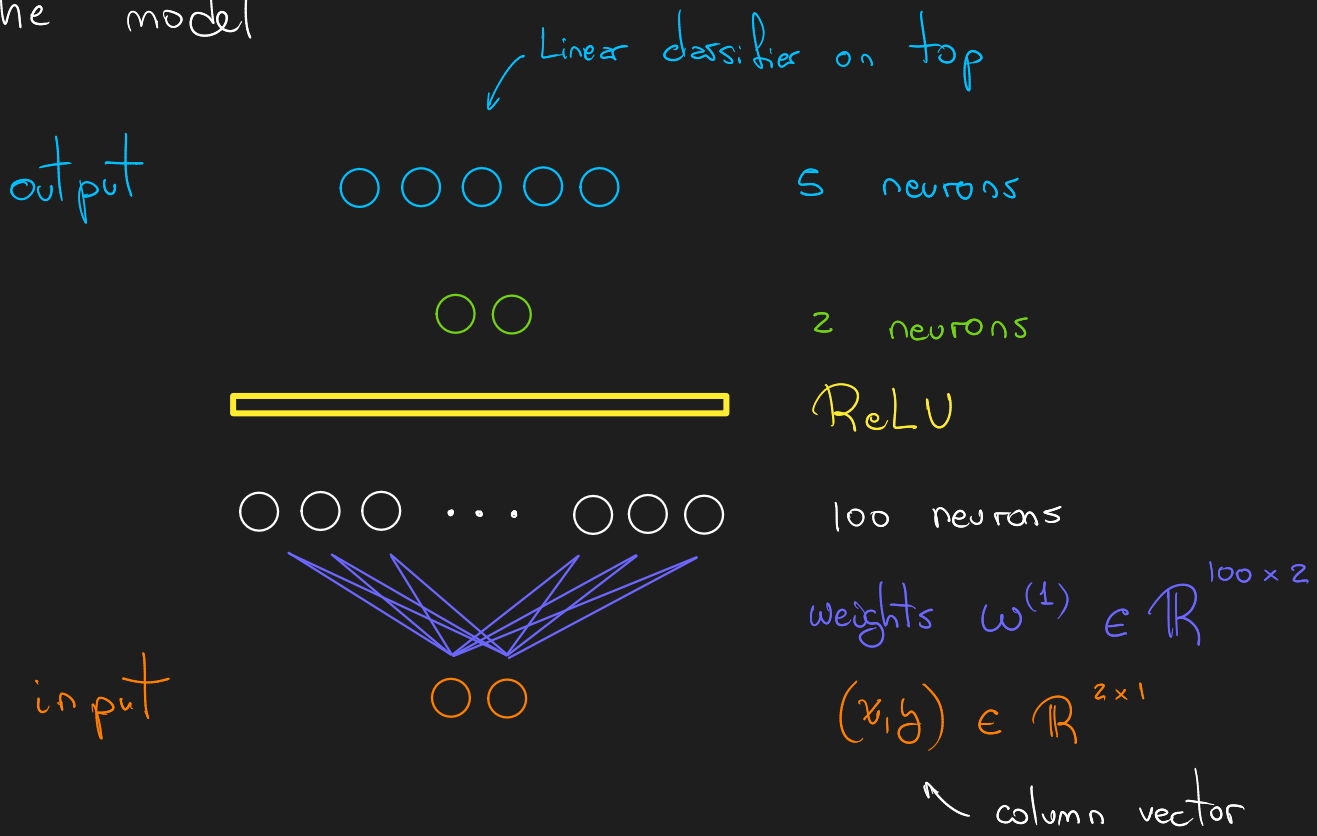


↑

After transforming the space, the classes are linearly separable.

# The model




Animation: Linear interpolation between ○○ and ○○

100 %
99 %
⋮

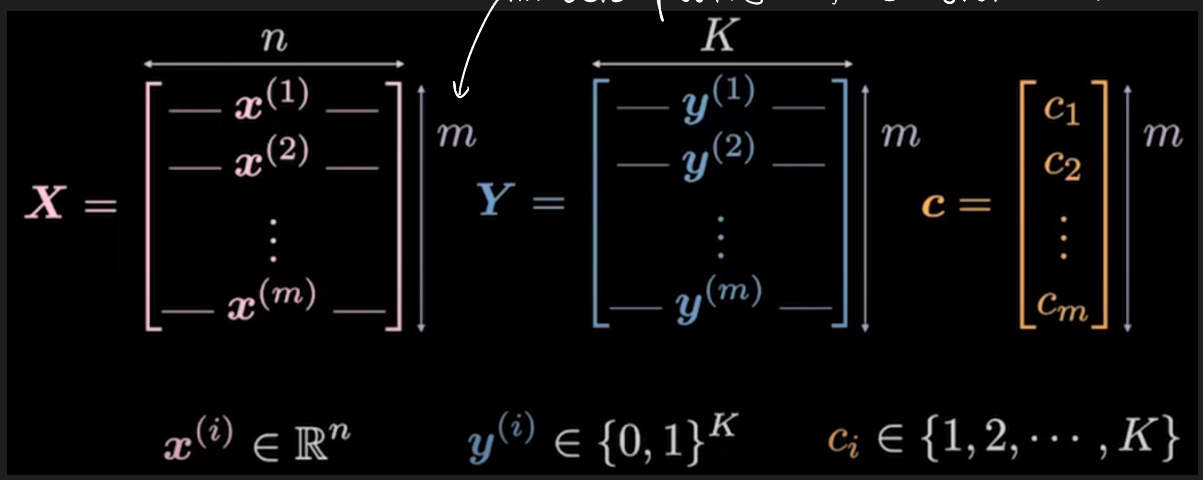
0 %
1 %
⋮

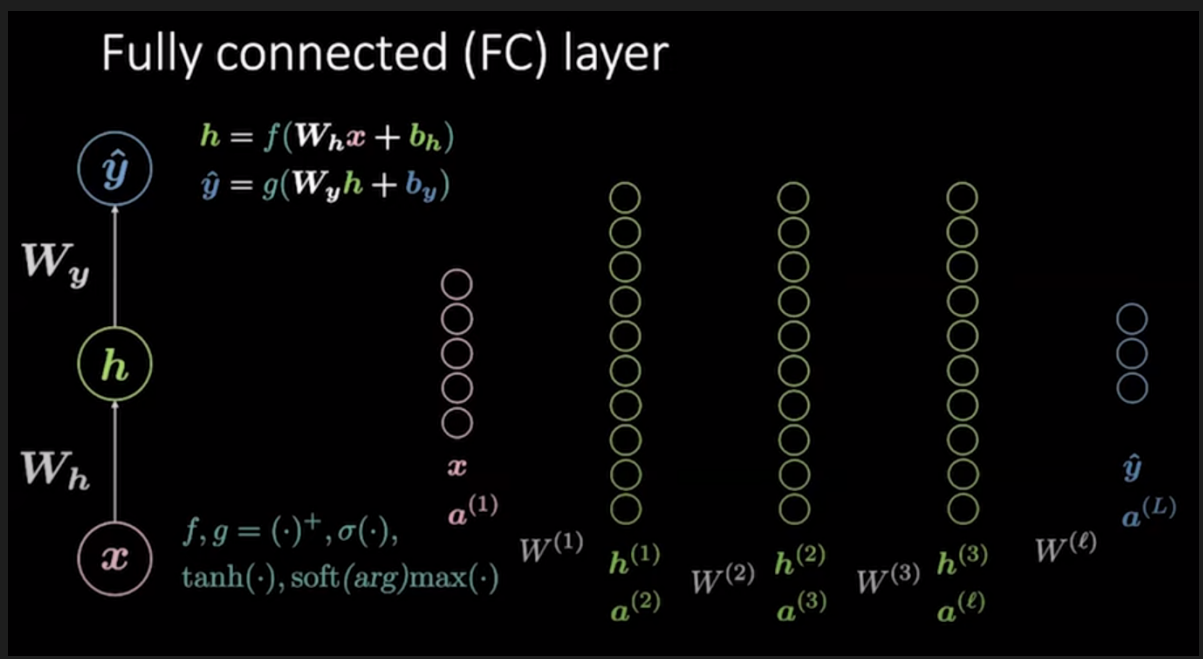
Having 100 neurons in the second layer let us move data into 100-dimensional space.

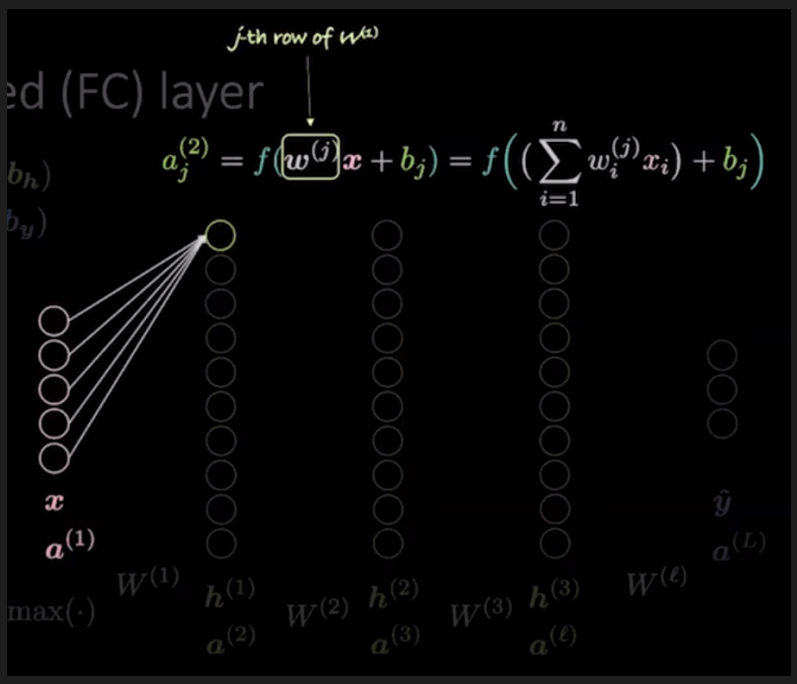
## Train data

X : Design matrix

c : Collection of labels / classes.



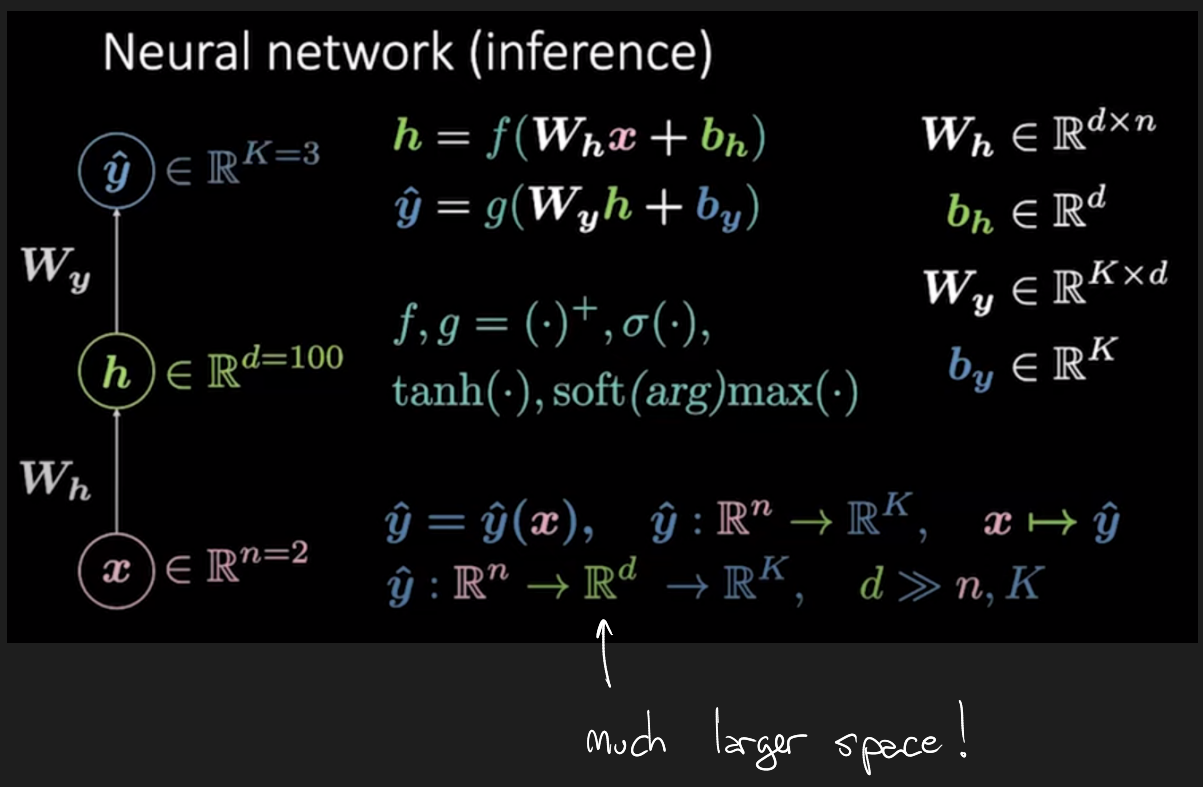






Linear function of linear function of linear function = Linear function

BUT

If using floatting point approximation on a computer, it can be seen as a non linear function in between linear functions.



Note: Counting from 1 (not zero)

## Neural network (training I)

$$h = f(W_h x + b_h)$$
$$\hat{y} = g(W_y h + b_y)$$

logits: output of final layer

$$\text{soft}(\arg)\max(\boldsymbol{l})[c] \doteq \frac{\exp(\boldsymbol{l}[c])}{\sum_{k=1}^{K} \exp(\boldsymbol{l}[k])} \in (0,1)$$

$$\mathcal{L}(\hat{\boldsymbol{Y}}, \boldsymbol{c}) \doteq \frac{1}{m} \sum_{i=1}^{m} \ell(\hat{\boldsymbol{y}}^{(i)}, c_i), \quad \ell(\hat{\boldsymbol{y}}, c) \doteq -\log(\hat{\boldsymbol{y}}[c])$$

cross entropy / negative log-likelihood

$$\boldsymbol{x}, \quad c = 1 \quad \Rightarrow \quad \boldsymbol{y} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$\hat{\boldsymbol{y}}(\boldsymbol{x}) = \begin{pmatrix} \sim 1 \\ \sim 0 \\ \sim 0 \end{pmatrix} \quad \Rightarrow \quad \ell\left(\begin{pmatrix} \sim 1 \\ \sim 0 \\ \sim 0 \end{pmatrix}, 1\right) \to 0^+$$

$$\hat{\boldsymbol{y}}(\boldsymbol{x}) = \begin{pmatrix} \sim 0 \\ \sim 1 \\ \sim 0 \end{pmatrix} \quad \Rightarrow \quad \ell\left(\begin{pmatrix} \sim 0 \\ \sim 1 \\ \sim 0 \end{pmatrix}, 1\right) \to +\infty$$

## Neural network (training II)

$$h = f(W_h x + b_h)$$
$$\hat{y} = g(W_y h + b_y)$$

$$\Theta \doteq \{\boldsymbol{W_h}, \boldsymbol{b_h}, \boldsymbol{W_y}, \boldsymbol{b_y}\}$$

$$J(\Theta) \doteq \mathcal{L}(\hat{Y}(\Theta), \boldsymbol{c}) \in \mathbb{R}^+$$

$$\frac{\partial J(\Theta)}{\partial \boldsymbol{W_y}} = \frac{\partial J(\Theta)}{\partial \hat{\boldsymbol{y}}} \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{W_y}}$$

$$\frac{\partial J(\Theta)}{\partial \boldsymbol{W_h}} = \frac{\partial J(\Theta)}{\partial \hat{\boldsymbol{y}}} \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{W_h}}$$

back-propagation