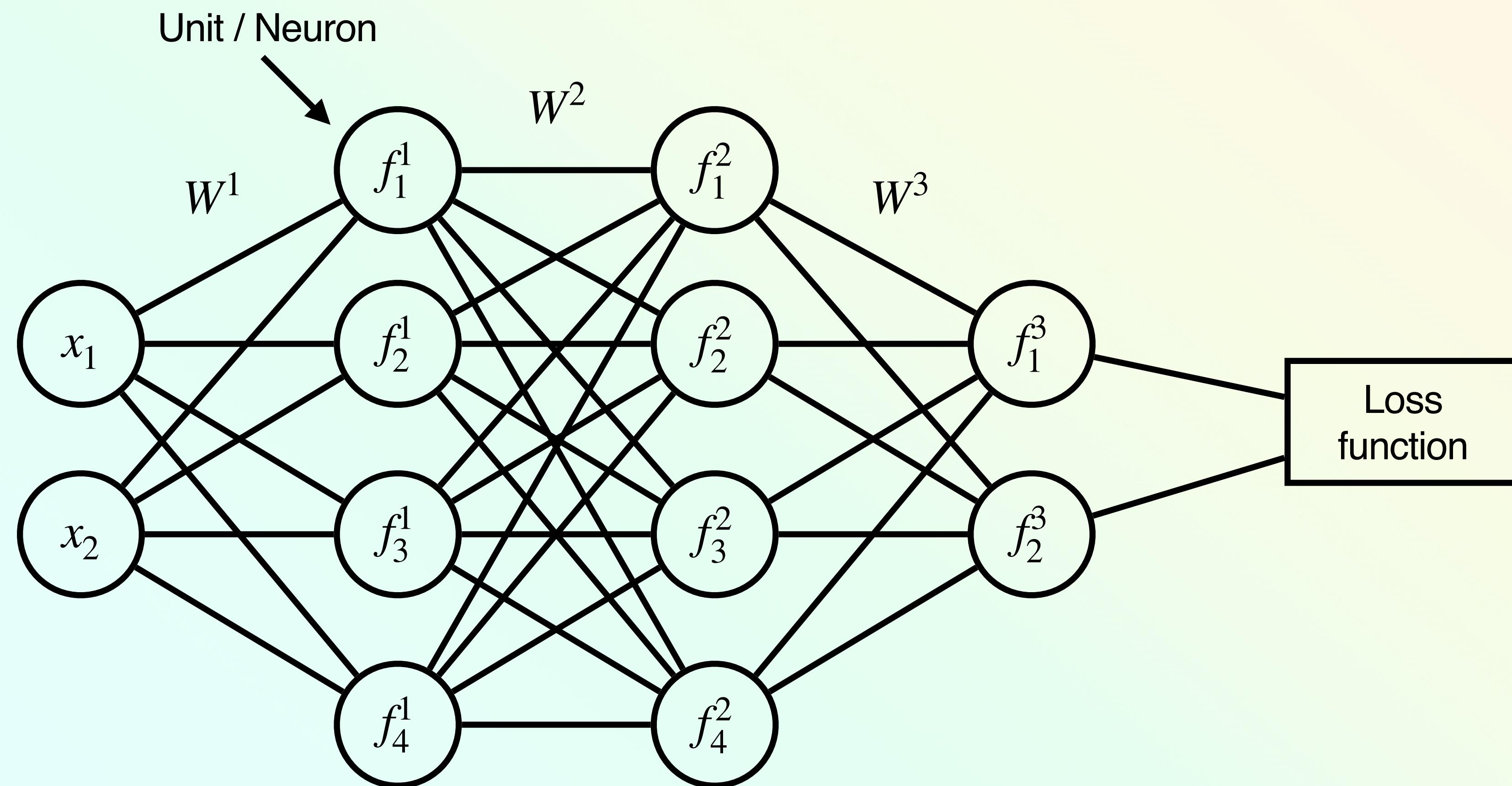


NEURAL NETWORKS

Input Layer

Hidden Layers

Output Layer



$$h^0$$

$$h^1 = f^1(h^0, W^1)$$

$$h^2 = f^2(h^1, W^2)$$

$$h^3 = f^3(h^2, W^3)$$

THE STRUCTURE OF NEURAL NETWORKS

ABSTRACT PROCESS

We can write the neural network outputs as a sequential process.

$$\begin{aligned} h^0 &= x \\ h^1 &= f^1(h^0, W^1) \\ h^2 &= f^2(h^1, W^2) \\ &\vdots \\ h^i &= f^i(h^{i-1}, W^i) \\ &\vdots \\ h^k &= f^k(h^{k-1}, W^k) \end{aligned}$$

To be concise, we can write the network output as $h^k = f(x, \theta)$, $\theta = \{W^i\}_{i=1}^k$

NEURAL NETWORK LAYERS

MULTILAYER PERCEPTION

$$f^i(h^{i-1}, W^i) = \sigma\left(h^{i-1}W^{i\top}\right)$$

$h_{i-1} \in \mathbb{R}^{1 \times n_{i-1}}$, $h_i \in \mathbb{R}^{1 \times n_i}$ — row vectors

$W^i \in \mathbb{R}^{n_i \times n_{i-1}}$ each row corresponds to the weights for one output unit

$\sigma: \mathbb{R} \rightarrow \mathbb{R}$ is a nonlinear function called an *activation function* that is applied elementwise

NOTATION FOR NEURAL NETWORKS

$x \in \mathbb{R}^{m \times n_0}$ is a matrix of m data points containing n_0 features for each data point

Each row is a data point

$$x = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n_0} \\ x_{1,1} & x_{1,2} & \cdots & x_{1,n_0} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n_0} \end{bmatrix}$$

$h^0 = x$ input to the neural network

NOTATION FOR NEURAL NETWORKS

$$h^1 = f^1(h^0, W^1) = \sigma(h^0 W^{1\top}) = \sigma(z^1)$$

$h^1 \in \mathbb{R}^{m \times n_1}$ outputs of the first layer of the neural network. Each row is a different data point

$$h^0 W^{1\top} = \begin{bmatrix} h_{1,1}^0 & h_{1,2}^0 & \cdots & h_{1,n_0}^0 \\ h_{1,1}^0 & h_{1,2}^0 & \cdots & h_{1,n_0}^0 \\ \vdots & \vdots & \ddots & \vdots \\ h_{m,1}^0 & h_{m,2}^0 & \cdots & h_{m,n_0}^0 \end{bmatrix} \begin{bmatrix} W_{1,1}^1 & W_{2,1}^1 & \cdots & W_{n_1,1}^1 \\ W_{1,2}^1 & W_{2,2}^1 & \cdots & W_{n_1,2}^1 \\ \vdots & \vdots & \ddots & \vdots \\ W_{1,n_0}^1 & hW_{2,n_0}^1 & \cdots & W_{n_1,n_0}^1 \end{bmatrix} = \begin{bmatrix} z_{1,1}^1 & z_{1,2}^1 & \cdots & z_{1,n_1}^1 \\ z_{1,1}^1 & z_{1,2}^1 & \cdots & z_{1,n_1}^1 \\ \vdots & \vdots & \ddots & \vdots \\ z_{m,1}^1 & z_{m,2}^1 & \cdots & z_{m,n_1}^1 \end{bmatrix} = z^1$$

Note the matrix on the right in z^1 is $W^{1\top}$, $W^1 \in \mathbb{R}^{n_1 \times n_0}$, $W^{1\top} \in \mathbb{R}^{n_0 \times n_1}$

NOTATION FOR NEURAL NETWORKS

For any layer i \circlearrowleft

$$h^i = f^i(h^{i-1}, W^i) = \sigma\left(h^{i-1}W^{i\top}\right) = \sigma(z^i) \leftarrow \text{Generalization}$$

$$h^i \in \mathbb{R}^{m \times n_i}, z^i \in \mathbb{R}^{m \times n_i}, W^i \in \mathbb{R}^{n_i \times n_{i-1}}$$

NOTATION FOR NEURAL NETWORKS

For the output layer (layer k)

The activation function may be the identify function, e.g., $\sigma(x) = x$

$$h^k = f^i(h^{k-1}, W^k) = h^{k-1}W^{k\top} = z^k$$

We do this when

Targets $y \in \mathbb{R}$ and loss is mean squared error *Regression*

sometimes for classification (numerical precision reasons, future lecture)

OUTPUT UNITS

multiple outputs

$y \in \mathbb{R}^{n_y}$ — Mean square error loss and linear output units (one data point (x, y))

$$l(x, y, \theta) = \|h^k - y\|_2^2 = \sum_{i=1}^{n_y} (h_i^k - y_i)^2$$

The only change is how we calculate this based on different weights that relate to the outcome we are targeting.

$y \in \{0, 1\}$ — negative log-likelihood and sigmoid output units

$$l(x, y, \theta) = -y \ln h^k - (1 - y) \ln(1 - h^k)$$

MULTICLASS CLASSIFICATION

Want to classify cats, dogs, planes, cars, horses, etc

$y \in \{1,2,3,4,\dots, n\}$ – each number represents a specific class

MULTICLASS CLASSIFICATION

Want to classify cats, dogs, planes, cars, horses, etc

$y \in \{1, 2, 3, 4, \dots, n\}$ – each number represents a specific class

Objective function:

Can still minimize the NLL, i.e., $-\ln \Pr(\hat{Y} = y | X = x)$

How to represent a probability distribution over multiple classes?

GENERALIZING BINARY CLASSIFICATION

$$\text{Sigmoid } \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\Pr(\hat{Y} = 1 | X = x) = \sigma(z), \quad z = h^{k-1}W^k \quad z \in \mathbb{R}$$

z represents the log odds of $\hat{Y} = 1$, i.e., how likely is the label 1.

GENERALIZING BINARY CLASSIFICATION

$$\text{Sigmoid } \sigma(z) = \frac{1}{1 + e^{-z}} = \frac{e^z}{1 + e^z}$$

$$\Pr(\hat{Y} = 1 | X = x) = \sigma(z), \quad z = h^{k-1}W^k \quad z \in \mathbb{R}$$

z represents the log odds of $\hat{Y} = 1$, i.e., how likely is the label 1.

If we increaseed the 1 would $\sigma(z)$ increase?

GENERALIZING BINARY CLASSIFICATION

$$\text{Sigmoid } \sigma(z) = \frac{1}{1 + e^{-z}} = \frac{e^z}{1 + e^z}$$

$$\Pr(\hat{Y} = 1 | X = x) = \sigma(z), \quad z = h^{k-1}W^k \quad z \in \mathbb{R}$$

z represents the log odds of $\hat{Y} = 1$, i.e., how likely is the label 1.

If we increaseed the 1 would $\sigma(z)$ increase?

- No, it decreases, and decreasing the 1 makes it increase
- Increasing 1 makes the probability of $Y = 0$ increase

GENERALIZING BINARY CLASSIFICATION

likelihood of class 0
vs
for class 1
 $z = [z_0, z_1]$

$$\sigma(z) = \frac{e^{z_1}}{e^{z_0} + e^{z_1}}$$

$$\Pr(\hat{Y} = 1 | X = x) = \sigma(z)$$

$$\Pr(\hat{Y} = 0 | X = x) = 1 - \sigma(z)$$

GENERALIZING BINARY CLASSIFICATION

$$\begin{aligned}\Pr(\hat{Y} = 0 \mid X = x) &= 1 - \sigma(z) \\&= 1 - \frac{e^{z_1}}{e^{z_0} + e^{z_1}} \\&= \frac{e^{z_0} + e^{z_1}}{e^{z_0} + e^{z_1}} - \frac{e^{z_1}}{e^{z_0} + e^{z_1}} \\&= \frac{e^{z_0}}{e^{z_0} + e^{z_1}}\end{aligned}$$

GENERALIZING BINARY CLASSIFICATION

$$\Pr(\hat{Y} = 0 | X = x) = \frac{e^{z_0}}{e^{z_0} + e^{z_1}}$$

Likelihood

$$\Pr(\hat{Y} = 1 | X = x) = \frac{e^{z_1}}{e^{z_0} + e^{z_1}}$$

Normalization

CREATING A PROBABILITY DISTRIBUTION

Probability distribution

$\forall x, \Pr(X = x) \geq 0$ – all events have a nonnegative probability

$\sum_x \Pr(X = x) = 1$ – probability of all events sum to one.

CREATING A PROBABILITY DISTRIBUTION

Probability distribution

$\forall x, \Pr(X = x) \geq 0$ — all events have a nonnegative probability

$\sum_x \Pr(X = x) = 1$ — probability of all events sum to one.

$z \in \mathbb{R}^n$ — scores representing the likelihood of each class

1. Make those scores positive
2. Normalize so that the total probability sums to 1

SOFTMAX

$z \in \mathbb{R}^n$

$$\Pr(\hat{Y} = i | X = x) = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}}$$

Make positive

Normalize

SOFTMAX

$$z \in \mathbb{R}^n$$

$$\Pr(\hat{Y} = i | X = x) = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}}$$

Make positive
Normalize

Positivity: $\forall x \in (-\infty, \infty), e^x > 0$

Total probability: $\sum_{i=1}^n \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}} = \frac{\sum_{i=1}^n e^{z_i}}{\sum_{j=1}^n e^{z_j}} = 1$

SOFTMAX OUTPUT UNIT

$$z = h^{k-1} W^k \in \mathbb{R}^n$$

$$\sigma(z) = \left[\frac{e^{z_1}}{\sum_{j=1}^n e^{z_j}}, \frac{e^{z_2}}{\sum_{j=1}^n e^{z_j}}, \dots, \frac{e^{z_n}}{\sum_{j=1}^n e^{z_j}} \right]$$

Probability of
class 1 class 2
↓
Normalize

class \cong probability

This will provide
a likelihood of
belonging to class

SOFTMAX OUTPUT UNIT

$$z = h^{k-1}W^k \in \mathbb{R}^n$$

$$\sigma(z) = \left[\frac{e^{z_1}}{\sum_{j=1}^n e^{z_j}}, \frac{e^{z_1}}{\sum_{j=1}^n e^{z_j}}, \dots, \frac{e^{z_n}}{\sum_{j=1}^n e^{z_j}} \right]$$

$$h^k = \sigma(z) = \left[\Pr(\hat{Y} = 1 | X = x), \Pr(\hat{Y} = 2 | X = x), \dots, \Pr(\hat{Y} = n | X = x) \right]$$

↖ ↘ ↗

Class 1 Class 2 Class n

LOSS FUNCTION FOR SOFTMAX

$$-\ln \Pr(\hat{Y} = y | X = x) = - \sum_{i=1}^n \mathbf{1}_{y=i} \ln h_{1,i}^k$$

↗ Just choose the ones that
corresponds to class i

LOSS FUNCTION FOR SOFTMAX

$$-\ln \Pr(\hat{Y} = y | X = x) = -\sum_{i=1}^n \mathbf{1}_{y=i} \ln h_{1,i}^k$$

Instead of for loops

$y = [0, \dots, 1, \dots, 0]$ – One-hot vector, the i^{th} element is 1 if $y = i$, all others are 0

$$l(x, y, \theta) = -y(\ln h^k)^\top = -\sum_{i=1}^n y_i \ln h_{1,i}^k$$

QUIZ

SOFTMAX PROPERTIES

Softmax has n parameters for n probabilities

Only need $n - 1$ parameters to define the probability distribution.

$$\Pr(X = 1) \doteq p_1, \Pr(X = 2) \doteq p_2, \dots, \Pr(X = \underbrace{n-1}_{\text{↑}}) \doteq p_{n-1}, \Pr(X = n) \doteq 1 - \sum_{i=1}^{n-1} p_i$$

Pr of the last term

we only need

$(n-1)$ parameters

SOFTMAX PROPERTIES

Softmax has n parameters for n probabilities

Only need $n - 1$ parameters to define the probability distribution.

$$\Pr(X = 1) \doteq p_1, \Pr(X = 2) \doteq p_2, \dots, \Pr(X = n - 1) \doteq p_{n-1}, \Pr(X = n) \doteq 1 - \sum_{i=1}^{n-1} p_i$$

A distribution with $\geq n$ parameters and n possible outcomes is called *over parameterized*

Multiple ways to represent the same probability distribution

SOFTMAX PROPERTIES

Softmax is over-parameterized:

Can set $z_n = 0$ and still represent probabilities for each class

$$i < n, \Pr(\hat{Y} = i | X = x) = \frac{e^{z_i}}{1 + \sum_{j=1}^{n-1} e^{z_j}}$$

\leftarrow sigmoid

$$\Pr(\hat{Y} = n | X = x) = \frac{1}{1 + \sum_{j=1}^{n-1} e^{z_j}}$$

In practice, both forms work equally well. Easier to do an over-parameterized version in calculations

SOFTMAX PROPERTIES

Invariance to translation

For $c \in \mathbb{R}$, $\sigma(z) = \sigma(z + c)$

In practice, we often subtract off the largest value to prevent numerical instability, e.g.,

$$\sigma(z - \max_i z_i) \quad \text{avoids numerical stability}$$

SOFTMAX PROPERTIES

Not invariant to rescaling:

$$\alpha > 0, \sigma(z) \neq \sigma(\alpha z)$$

$\alpha \rightarrow 0$ – probabilities become uniform, e.g., $1/n$

$\alpha \rightarrow \infty$ – probabilities become 0 except for the largest i , which has a probability 1

α is called the temperature of the softmax



$$\Pr(\hat{Y} = i | X = x) = \frac{e^{\alpha z_i}}{\sum_j e^{\alpha z_j}}$$

ACTIVATION FUNCTIONS

FOR HIDDEN LAYERS

$$\text{Sigmoid: } \sigma(x) = \frac{1}{1 + e^{-x}}$$

ACTIVATION FUNCTIONS

FOR HIDDEN LAYERS

$$\text{Sigmoid: } \sigma(x) = \frac{1}{1 + e^{-x}}$$

close

$$\text{Tanh: } \sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2 \frac{1}{1 + e^{-2x}} - 1$$

$$\text{ReLU (Rectified Linear Unit): } \sigma(x) = \max(0, x) \leftarrow \text{Always } +ve$$

$$\text{Softplus: } \sigma(x) = \ln(1 + e^x)$$

ACTIVATION FUNCTIONS

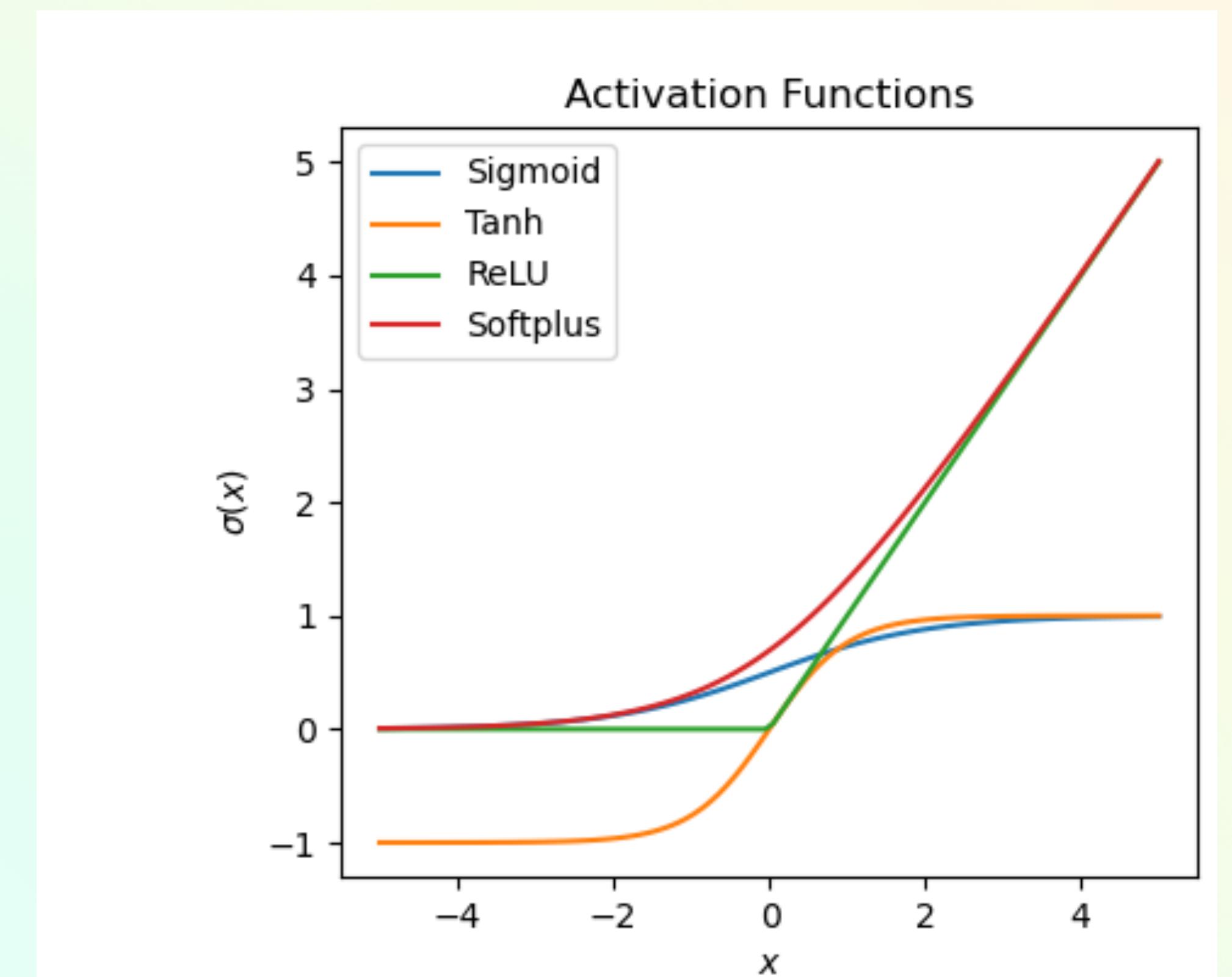
FOR HIDDEN LAYERS

$$\text{Sigmoid: } \sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\text{Tanh: } \sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2 \frac{1}{1 + e^{-2x}} - 1$$

ReLU (Rectified Linear Unit): $\sigma(x) = \max(0, x)$

Softplus: $\sigma(x) = \ln(1 + e^x)$



ACTIVATION FUNCTIONS

FOR HIDDEN LAYERS

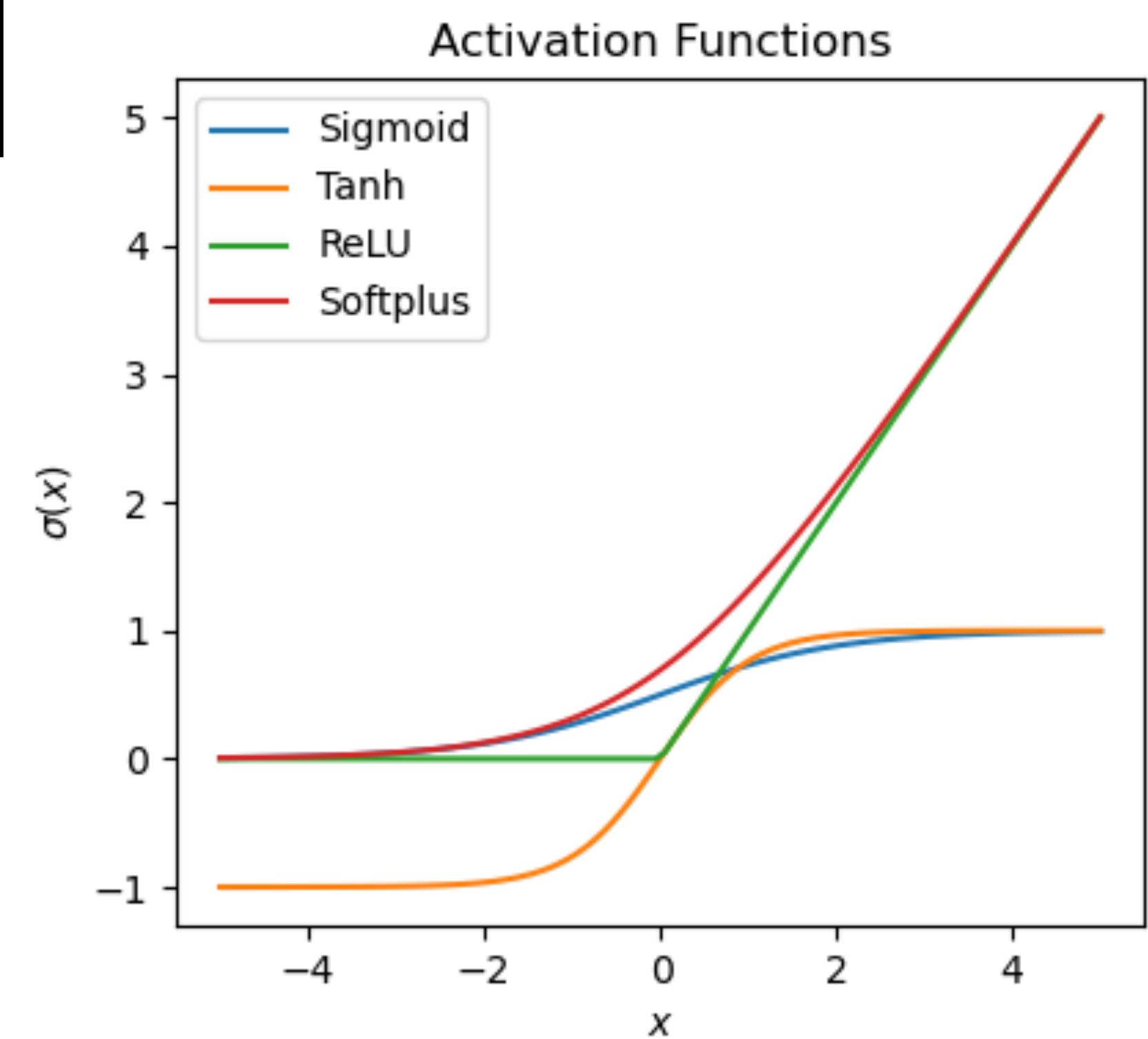
$$\text{Sigmoid: } \sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\text{Tanh: } \sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2 \frac{1}{1 + e^{-2x}} - 1$$

$$\text{ReLU (Rectified Linear Unit): } \sigma(x) = \max(0, x)$$

$$\text{Softplus: } \sigma(x) = \ln(1 + e^x)$$

Which should we use?



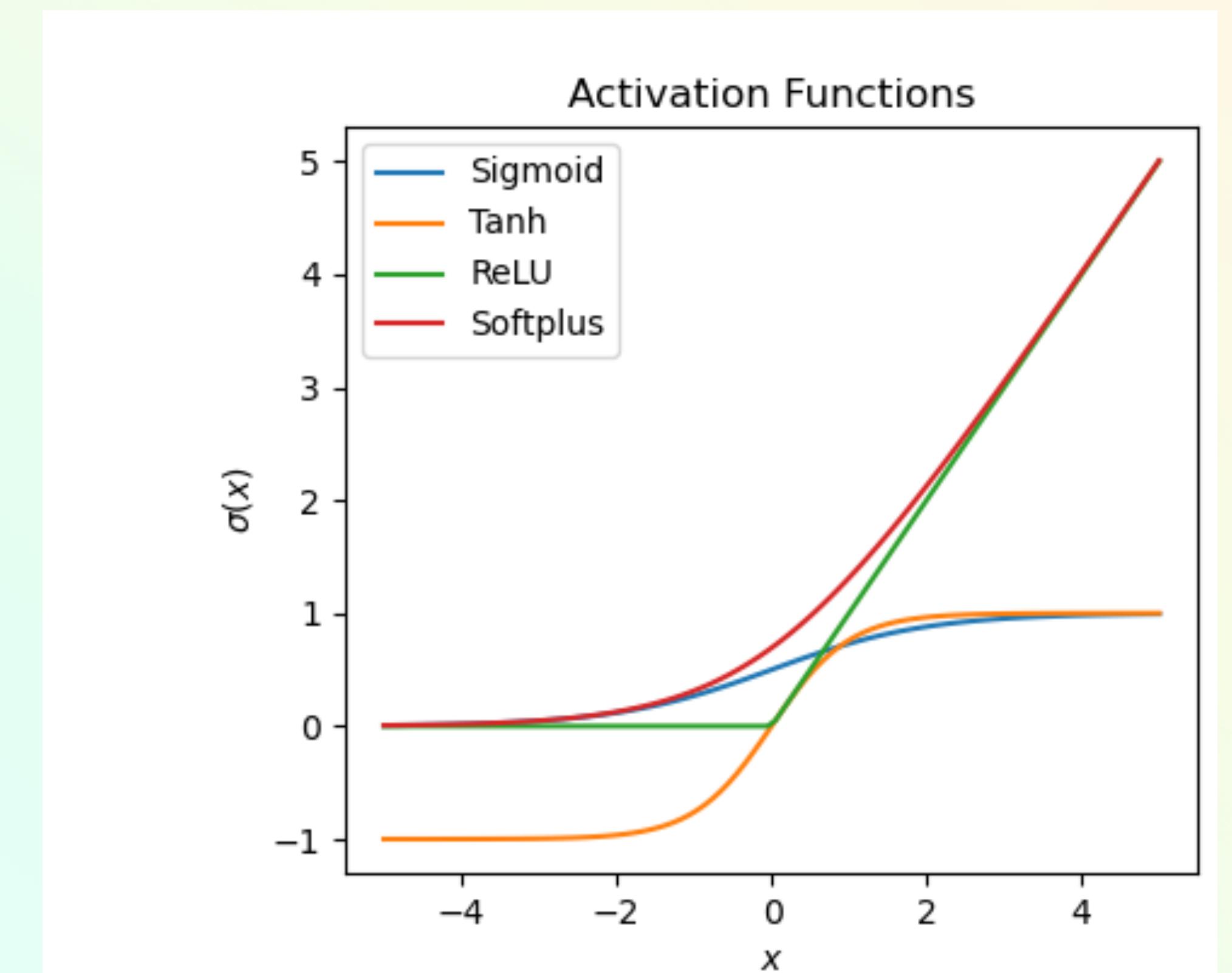
ACTIVATION FUNCTIONS

FOR HIDDEN LAYERS

With enough hidden units, they all can represent similar functions

We want to choose one that makes learning easy

- Derivative for weights at each hidden layer should not be flat



PARTIAL DERIVATIVES

FOR EACH HIDDEN LAYER

will do this in the
3rd homework

For a single data point x, y

For a single data point x, y

chain rule

$$\frac{\partial l(x, y, \theta)}{\partial w_i} = ? = \underbrace{\frac{\partial L(x, y, \theta)}{\partial h^k}}_{\substack{\approx \\ \text{backward pass}}} \underbrace{\frac{\partial h^k}{\partial w_i}}_{\substack{\approx \\ \text{backward pass}}} \Rightarrow \underbrace{\frac{\partial L(x, y, \theta)}{\partial h^k}}_{\substack{\approx \\ \text{backward pass}}} \underbrace{\frac{\partial f^k(h^{k-1}, w^k)}{\partial w_i}}_{\substack{\approx \\ \text{backward pass}}} \underbrace{\frac{\partial h^{k-1}}{\partial w_i}}_{\substack{\approx \\ \text{backward pass}}} \dots \underbrace{\frac{\partial f^{k-1}(h^{k-2}, w^{k-1})}{\partial w_i}}_{\substack{\approx \\ \text{backward pass}}} \underbrace{\frac{\partial h^{k-2}}{\partial w_i}}_{\substack{\approx \\ \text{backward pass}}} \dots \underbrace{\frac{\partial f^1(h^0, w^1)}{\partial w_i}}_{\substack{\approx \\ \text{backward pass}}}$$

PARTIAL DERIVATIVES

FOR EACH HIDDEN LAYER

$$\begin{aligned}\frac{\partial l(x, y, \theta)}{\partial W^i} &= \frac{\partial l(x, y, \theta)}{\partial h^k} \frac{\partial h^k}{\partial W^i} \\&= \frac{\partial l(x, y, \theta)}{\partial h^k} \frac{\partial f(h^{k-1}, W^k)}{\partial W^i} \\&= \frac{\partial l(x, y, \theta)}{\partial h^k} \frac{\partial f^k(h^{k-1}, W^k)}{\partial h^{k-1}} \frac{\partial h^{k-1}}{\partial W^i} \\&= \frac{\partial l(x, y, \theta)}{\partial h^k} \frac{\partial f^k(h^{k-1}, W^k)}{\partial h^{k-1}} \frac{\partial f^{k-1}(h^{k-2}, W^{k-1})}{\partial W^i} \\&= \frac{\partial l(x, y, \theta)}{\partial h^k} \frac{\partial f^k(h^{k-1}, W^k)}{\partial h^{k-1}} \frac{\partial f^{k-1}(h^{k-2}, W^{k-1})}{\partial h^{k-2}} \frac{\partial h^{k-2}}{\partial W^i} \\&\vdots \\&= \frac{\partial l(x, y, \theta)}{\partial h^k} \frac{\partial f^k(h^{k-1}, W^k)}{\partial h^{k-1}} \frac{\partial f^{k-1}(h^{k-2}, W^{k-1})}{\partial h^{k-2}} \dots \frac{\partial f^i(h^{i-1}, W^i)}{\partial W^i}\end{aligned}$$

PARTIAL DERIVATIVES

FOR EACH HIDDEN LAYER

$$\begin{aligned}\frac{\partial f^i(h^{i-1}, W^i)}{\partial W^i} &= \frac{\partial \sigma(h^{i-1} W^{i\top})}{\partial W^i} \\ &= \frac{\partial \sigma(h^{i-1} W^{i\top})^\top}{\partial h^{i-1} W^{i\top}} \frac{\partial h^{i-1} W^{i\top}}{\partial W^i} \end{aligned}$$

↑
input ↑
 output
 weight

PARTIAL DERIVATIVES

FOR EACH HIDDEN LAYER

$$\begin{aligned}\frac{\partial f^i(h^{i-1}, W^i)}{\partial W^i} &= \frac{\partial \sigma(z^i)}{\partial W^i} \\ &= \frac{\partial \sigma(z^i)^\top}{\partial z^i} \frac{\partial z^i}{\partial W^i}\end{aligned}$$

PARTIAL DERIVATIVES

FOR EACH HIDDEN LAYER

$$\begin{aligned}\frac{\partial f^i(h^{i-1}, W^i)}{\partial W^i} &= \frac{\partial \sigma(z^i)}{\partial W^i} \\ &= \frac{\partial \sigma(z^i)}{\partial z^i} \left(\frac{\partial z^i}{\partial W^i} \right)^\top \\ &= \begin{bmatrix} \frac{\partial \sigma(z_{1,1}^i)}{\partial z_1^i} \\ \frac{\partial \sigma(z_{1,2}^i)}{\partial z_2^i} \\ \vdots \\ \frac{\partial \sigma(z_{1,n_i}^i)}{\partial z_{1,n_i}^i} \end{bmatrix}^\top h^{i-1}\end{aligned}$$

PARTIAL DERIVATIVES

FOR EACH HIDDEN LAYER

$$\frac{\partial f^i(h^{i-1}, W^i)}{\partial W^i} = \begin{bmatrix} \frac{\partial \sigma(z_{1,1}^i)}{\partial z_1^i} \\ \frac{\partial \sigma(z_{1,2}^i)}{\partial z_2^i} \\ \vdots \\ \frac{\partial \sigma(z_{1,n_i}^i)}{\partial z_{1,n_i}^i} \end{bmatrix}^\top h^{i-1}$$

If $\frac{\partial \sigma(z_{1,j}^i)}{\partial z_{1,j}^i}$ is near zero, then the derivative is also near zero.

Want an activation function that does not have small derivatives ←

ACTIVATION FUNCTIONS

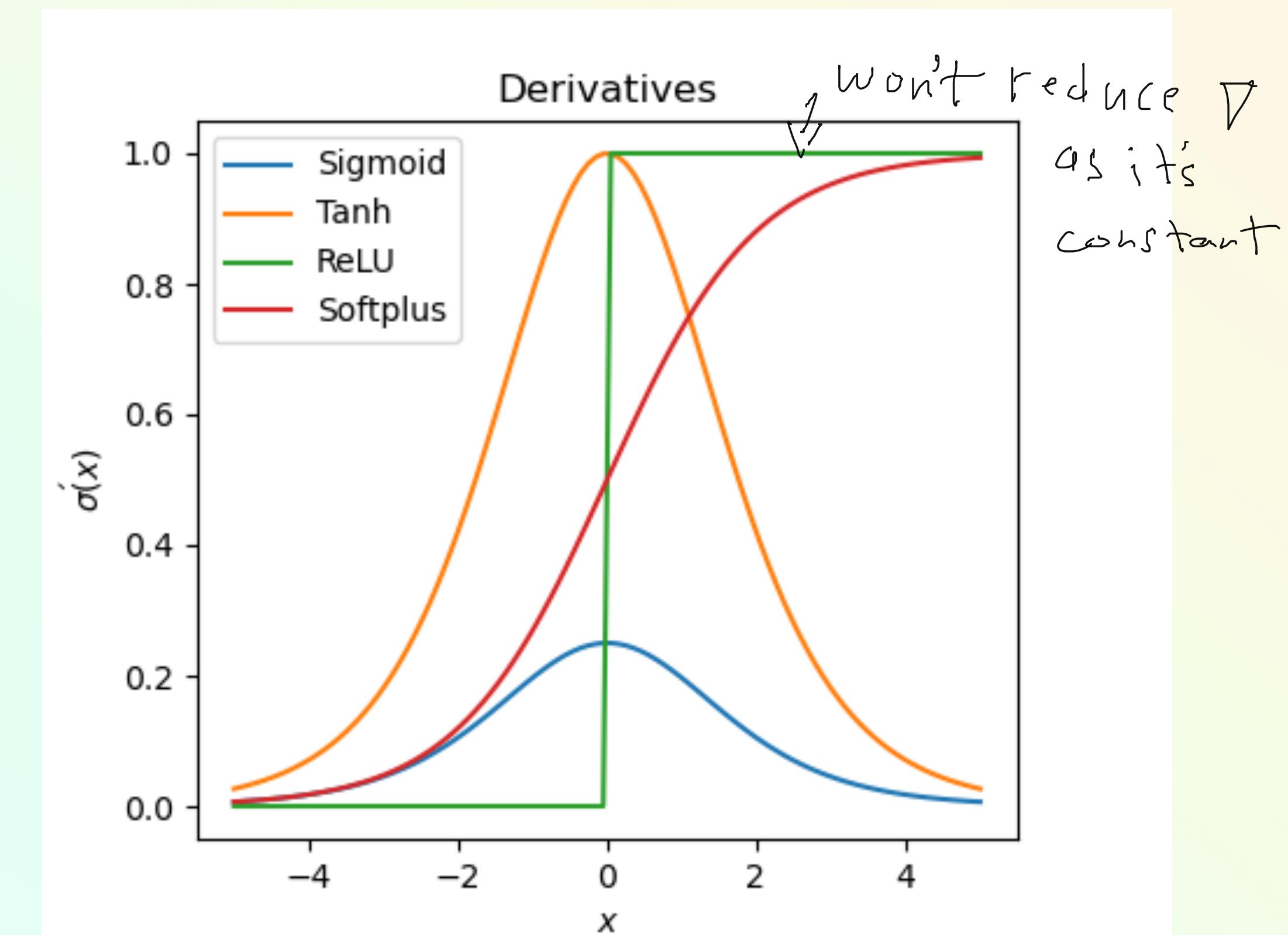
FOR HIDDEN LAYERS

$$\text{Sigmoid: } \sigma = \frac{1}{1 + e^{-x}}$$

$$\text{Tanh: } \sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2 \frac{1}{1 + e^{-2x}} - 1$$

ReLU (Rectified Linear Unit): $\sigma(x) = \max(0, x)$

Softplus: $\sigma(x) = \ln(1 + e^x)$



ACTIVATION FUNCTIONS

FOR HIDDEN LAYERS

important

Sigmoid: makes gradient smaller

Tanh: better than sigmoid near 0

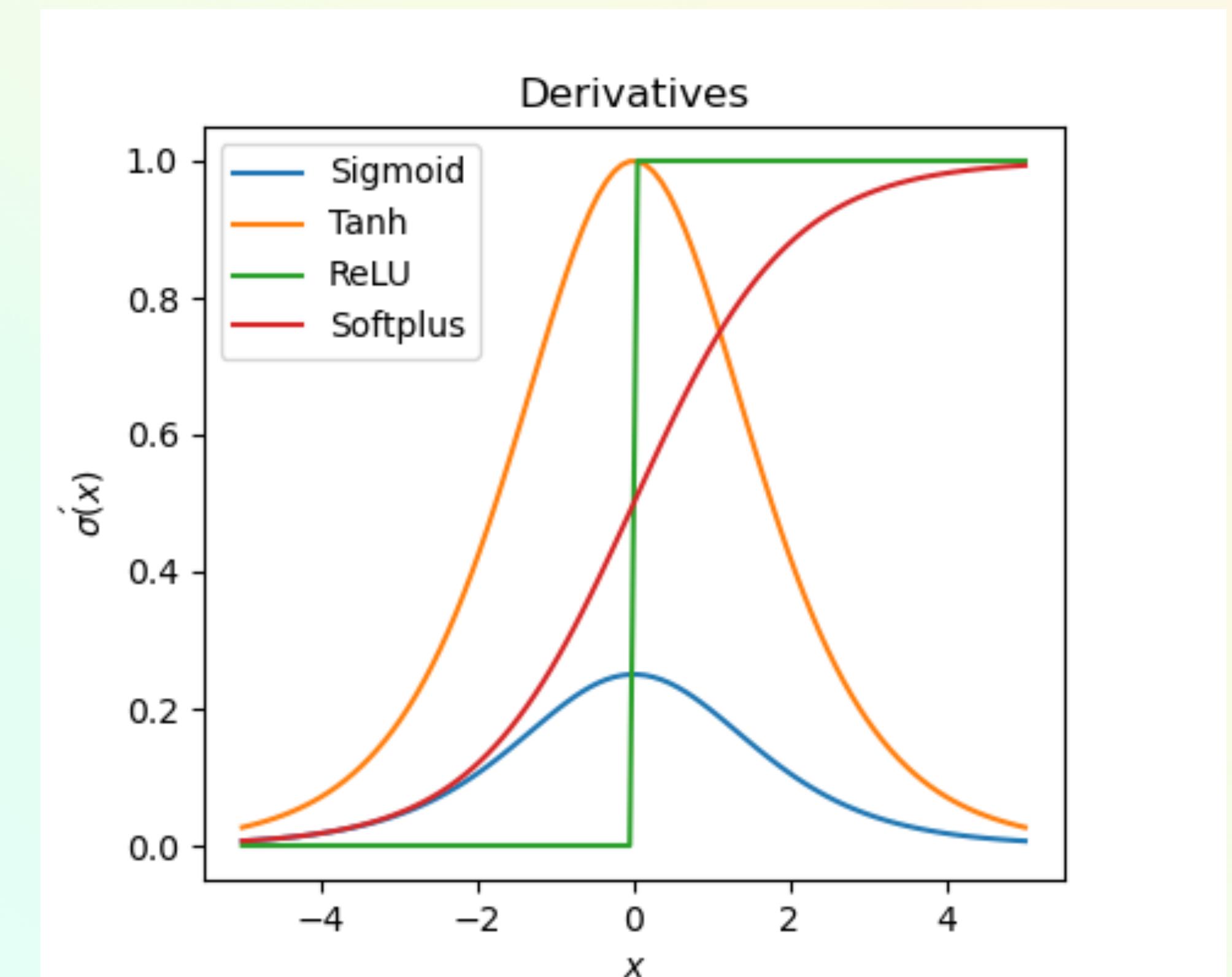
- can initialize weights so z is close to zero

ReLU: Great when $x > 0$!

- if x is always < 0 , then it will be a useless unit

Softplus: Like ReLU but smooth

- still shrinks gradients at each layer



ACTIVATION FUNCTIONS

FOR HIDDEN LAYERS

Sigmoid: makes gradient smaller

Tanh: better than sigmoid near 0

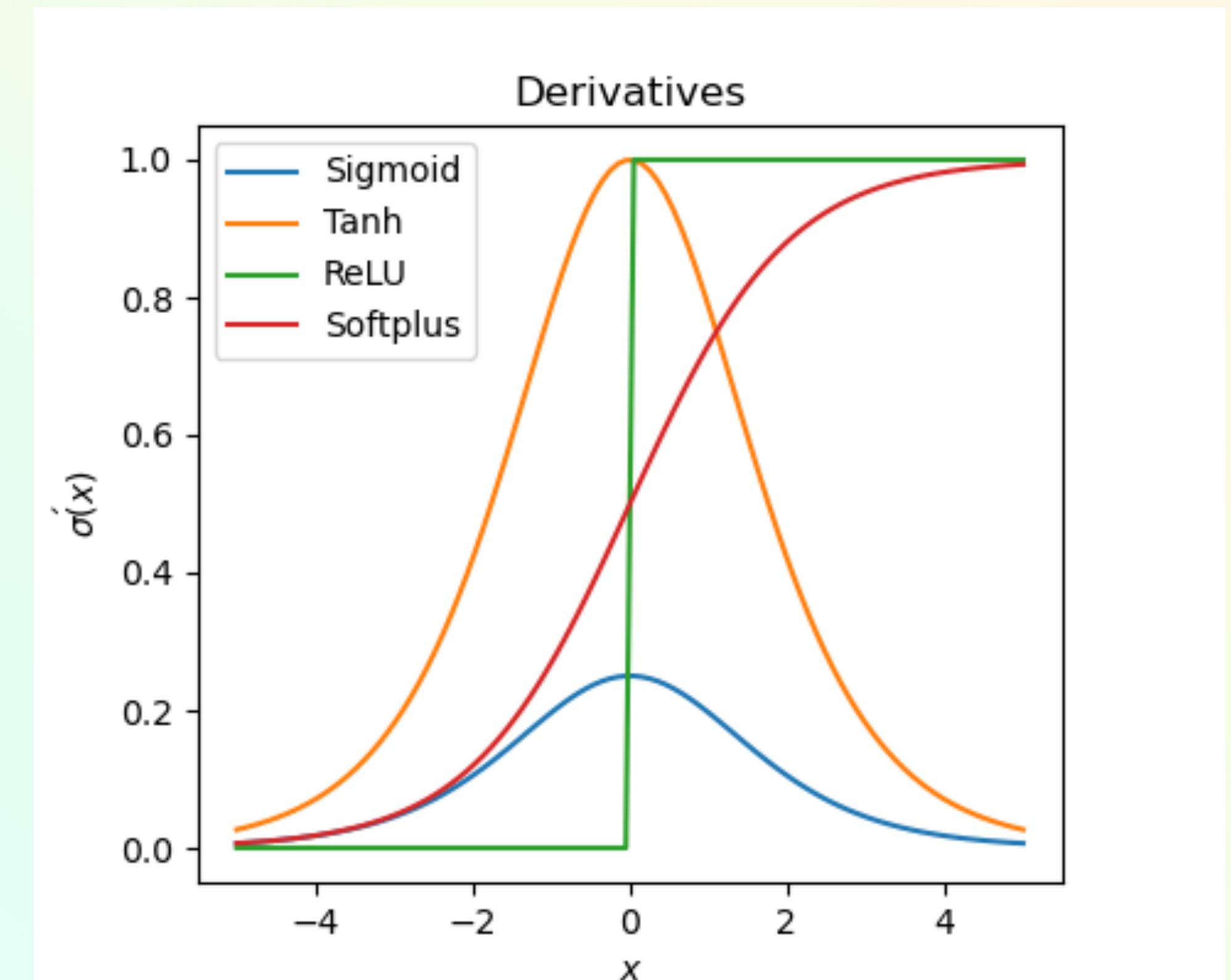
- can initialize weights so z is close to zero

ReLU: Great when $x > 0$!

- if x is always < 0 , then it will be a useless unit

Softplus: Like ReLU but smooth

- still shrinks gradients at each layer



PARTIAL DERIVATIVES

FOR EACH HIDDEN LAYER

$$\frac{\partial l(x, y, \theta)}{\partial W^i} = \frac{\partial l(x, y, \theta)}{\partial h^k} \frac{\partial f^k(h^{k-1}, W^k)}{\partial h^{k-1}} \frac{\partial f^{k-1}(h^{k-2}, W^{k-1})}{\partial h^{k-2}} \cdots \frac{\partial f^i(h^{i-1}, W^i)}{\partial W^i}$$

Each layer has the potential to increase or decrease the magnitude of the derivatives

PARTIAL DERIVATIVES

FOR EACH HIDDEN LAYER

$$\frac{\partial l(x, y, \theta)}{\partial W^i} = \frac{\partial l(x, y, \theta)}{\partial h^k} \frac{\partial f^k(h^{k-1}, W^k)}{\partial h^{k-1}} \frac{\partial f^{k-1}(h^{k-2}, W^{k-1})}{\partial h^{k-2}} \cdots \frac{\partial f^i(h^{i-1}, W^i)}{\partial W^i}$$

Each layer has the potential to increase or decrease the magnitude of the derivatives

Let $A^i = \frac{\partial f^i(h^{i-1}, W^i)}{\partial h^{i-1}}$ ← derivative with respect to the input-

$$\frac{\partial l(x, y, \theta)}{\partial W^i} = \frac{\partial l(x, y, \theta)}{\partial h^k} \left(\prod_{j=i+1}^k A^j \right) \frac{\partial f^i(h^{i-1}, W^i)}{\partial W^i}$$

PARTIAL DERIVATIVES

FOR EACH HIDDEN LAYER

$$\frac{\partial l(x, y, \theta)}{\partial W^i} = \frac{\partial l(x, y, \theta)}{\partial h^k} \frac{\partial f^k(h^{k-1}, W^k)}{\partial h^{k-1}} \frac{\partial f^{k-1}(h^{k-2}, W^{k-1})}{\partial h^{k-2}} \dots \frac{\partial f^i(h^{i-1}, W^i)}{\partial W^i}$$

Each layer has the potential to increase or decrease the magnitude of the derivatives

$$\text{Let } A^i = \frac{\partial f^i(h^{i-1}, W^i)}{\partial h^{i-1}}$$

$$\frac{\partial l(x, y, \theta)}{\partial W^i} = \frac{\partial l(x, y, \theta)}{\partial h^k} \left(\prod_{j=i+1}^k A^j \right) \frac{\partial f^i(h^{i-1}, W^i)}{\partial W^i}$$

The product of A^j 's can quickly go to 0 or grow much greater than 1 — exploding or vanishing gradients

Deep networks can be harder to train because of this property. (Used to)

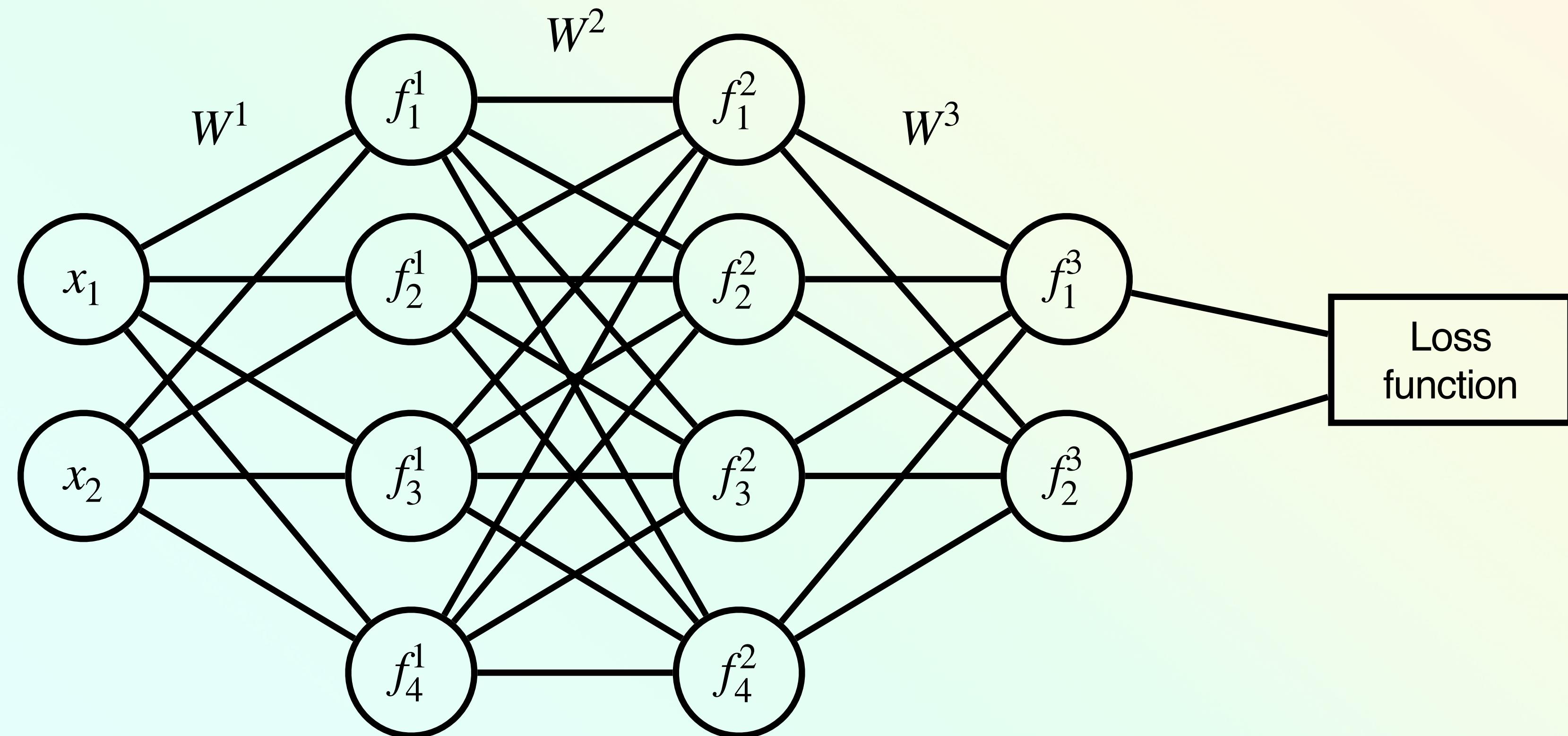
QUIZ

How wide?
How deep?

Input Layer

Hidden Layers

Output Layer



$$h^0$$

$$h^1 = f^1(h^0, W^1)$$

$$h^2 = f^2(h^1, W^2)$$

$$h^3 = f^3(h^2, W^3)$$

UNIVERSAL FUNCTION APPROXIMATION

THEOREM-ISH

For some function f on a compact set \mathcal{X} there exists a neural network with a single hidden layer that can approximate f to an arbitrarily small precision

- hidden layer may need to be infinitely wide
- other results for infinite depth (see [Wikipedia](#))

Does not say how to find that network, only that it exists.

CHOOSING WIDTH

GENERAL INSIGHTS

Width controls how many linear functions that layer can represent

- Decreasing the width over the layers or keeping them the same is the general strategy
- A wide layer after a narrow layer provides no value

Adding depth allows for identifying and repeating patterns

- more depth is usually more helpful
- the number of functions that can be represented grows faster with depth than width

ACTIVATION FUNCTIONS

WHAT THEY REPRESENT

$$z^i = h^{i-1} W^{i-1 \top} + b^i$$

$$\sigma(x) = \frac{1}{1 + e^{-z^i}}$$

ACTIVATION FUNCTIONS

WHAT THEY REPRESENT

$$z^i = h^{i-1} W^{i-1}^\top + b^i$$

$$\sigma(x) = \frac{1}{1 + e^{-z^i}}$$

W – controls direction of a hyperplane

b – controls position of that plane relative to the origin

ACTIVATION

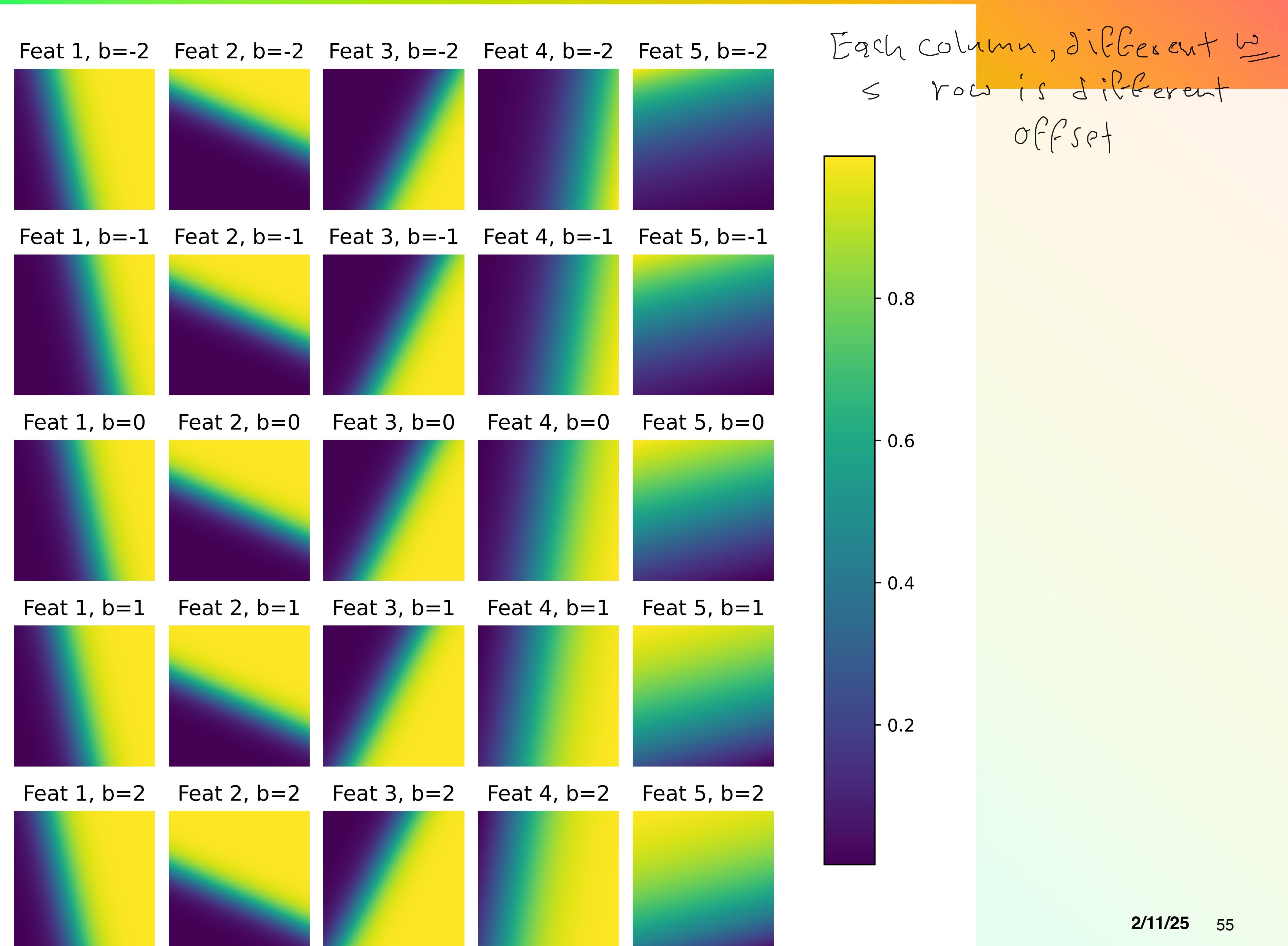
WHAT THEY REPRESENT

$$z^i = h^{i-1} W^{i-1}^\top$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

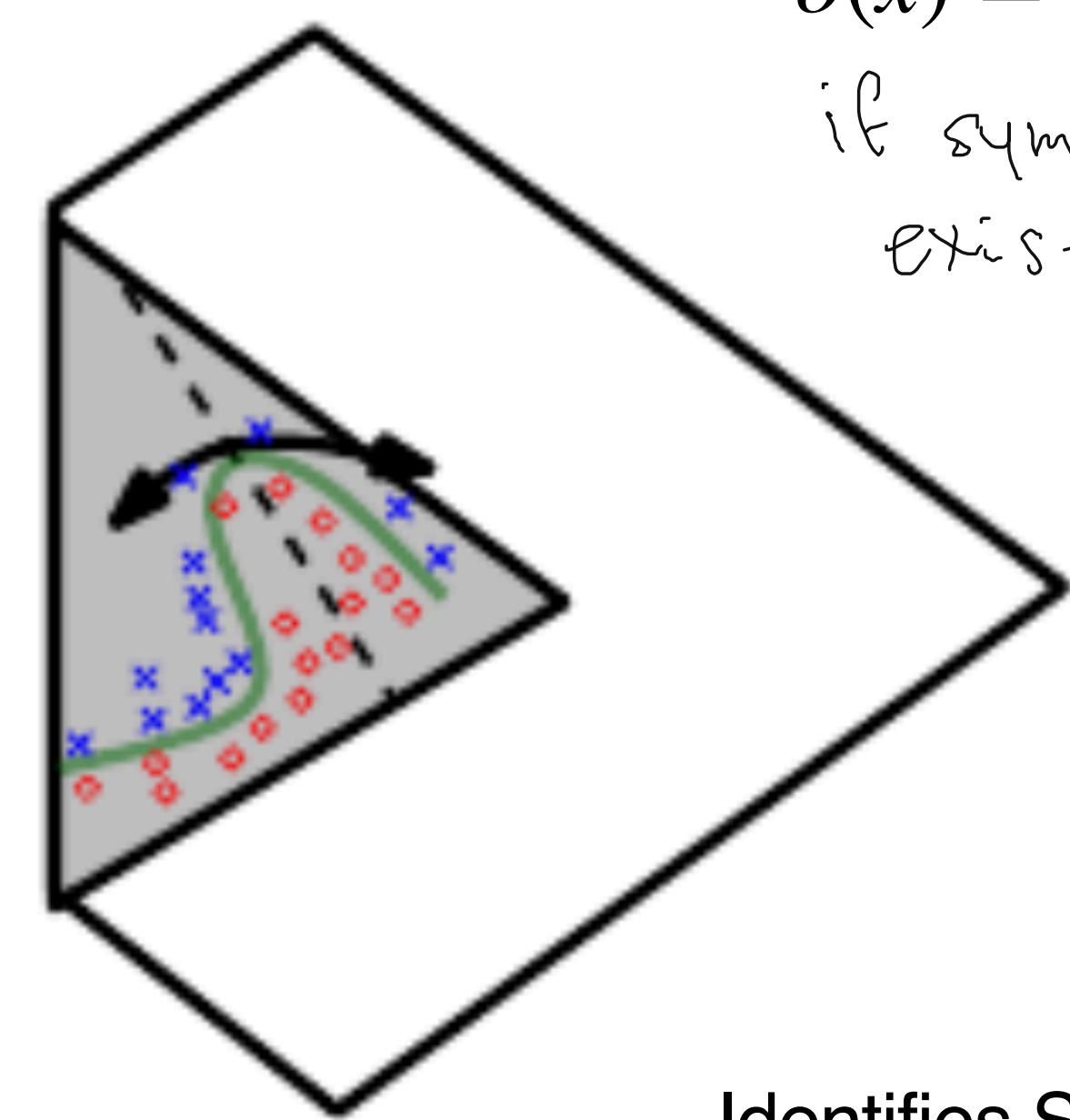
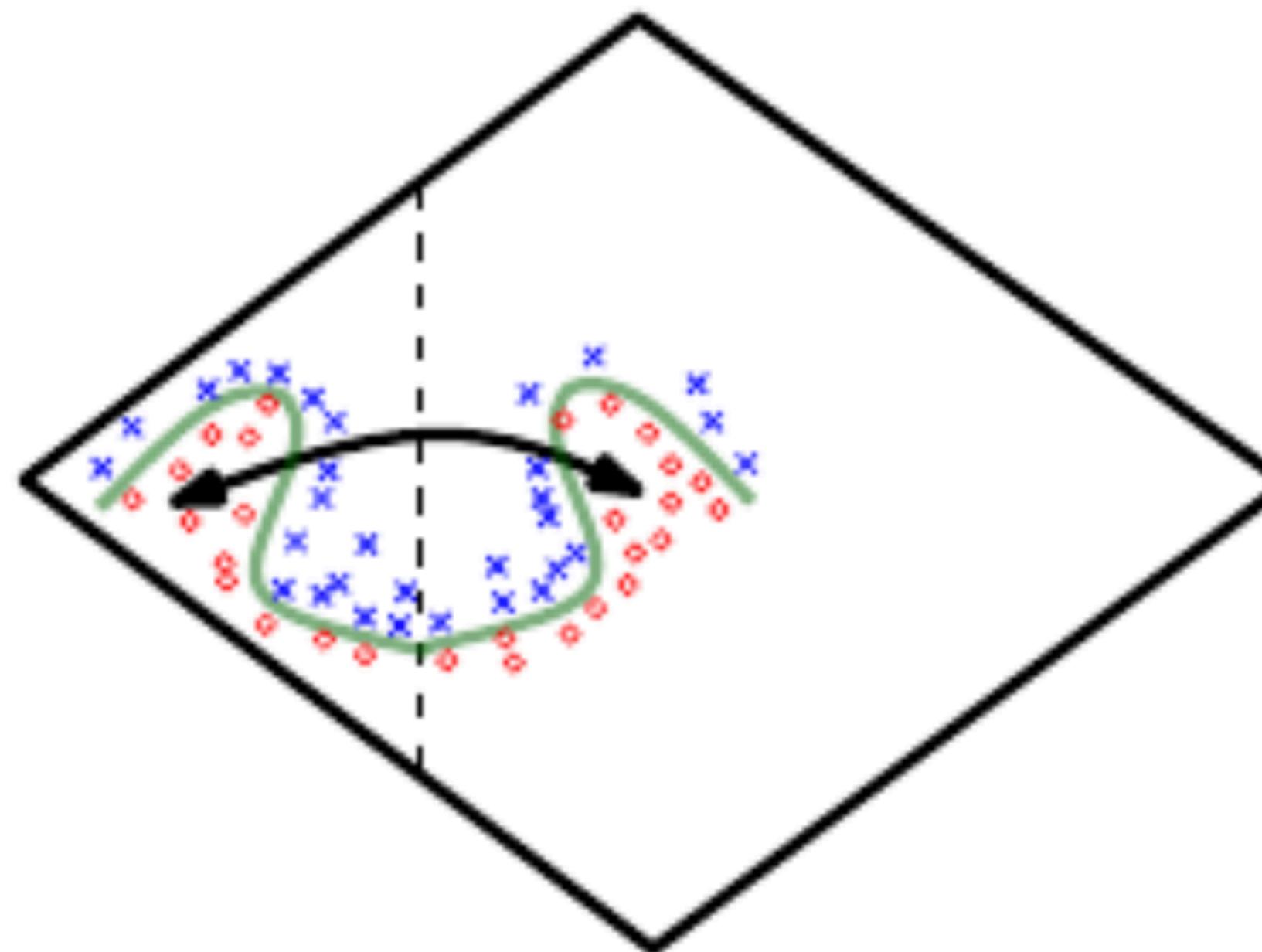
W — controls dimensions

b — controls position



ACTIVATION FUNCTIONS

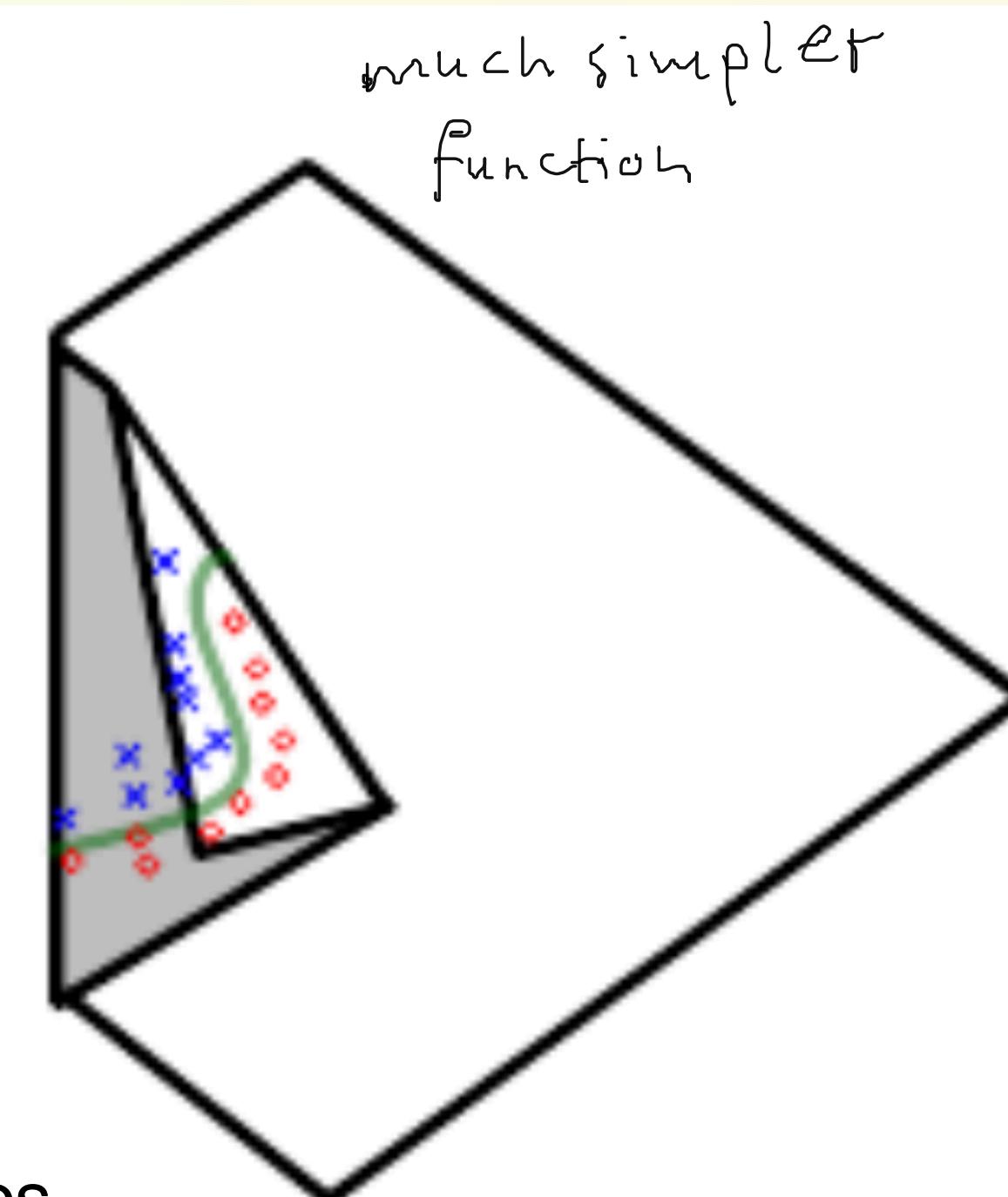
WHAT THEY REPRESENT , what they do



$$\sigma(x) = \underline{|x|}$$

if symmetries
exist

Identifies Symmetries



NEXT CLASS

Next Class — Training Neural Networks