NEURAL NETWORKS

COMPOSING BASIS FUNCTIONS

GENERAL PROCESS

$$f(x, w, \beta^1, \beta^2) = w^\top \phi^2 \left(\phi^1(x, \beta^1), \beta^2\right)$$

$$h^1 = \phi^1(x, \beta^1)$$

$$h^2 = \phi^2(h^1, \beta^2)$$

$$\hat{y} = w^\top h^2$$

BENEFITS OF COMPOSITION

GENERAL PROCESS

- Don’t have to consider all feature interactions in one basis function
- Can focus on local feature extraction (e.g., patches in an image)
- Can increase function complexity with width and depth
  - More compositions —> more expressive function approximation
  - More features —> more expressive function approximation
- Some functions are compositional in nature
  - Images: lines->groups of lines->(nose, eyes)->face
  - Text math: “give an answer for $3 \times (4 - 3) - 2/10$”
    - identify numbers and operations, perform operations in order->combine results.

QUIZ

THE STRUCTURE OF NEURAL NETWORKS

ABSTRACT PROCESS

A multi-layered neural network is a composition of functions $f^1, f^2, \ldots, f^k$

Each layer has some weight matrix W^i (this could be more than just a matrix)

$f^i \colon \mathscr{H}^{i-1} \times \mathbb{R}^{n_{i-1} \times n_i} \to \mathscr{H}^i$, where $\mathscr{H}^i$ is the output space of the i^{th} layer, $H^0 = \mathscr{X}$ is the input space, and n_i is the dimensionality of $\mathscr{H}^i$.

The output of the network is

$$f^k\left(f^{k-1}\left(\ldots\left(f^2\left(f^1\left(x, W^1\right), W^2\right), \ldots\right), W^{k-1}\right), W^k\right)$$

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 $h^k = f\left(x, \{W^i\}_{i=1}^k\right)$

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 $h^k = f(x, \theta)$, $\theta = \{W^i\}_{i=1}^k$

THE STRUCTURE OF NEURAL NETWORKS

ABSTRACT PROCESS

For a regression problem, we can take the output h^k and put it into the mean squared error loss function.

$$l(X, Y, \theta) = \left(f(X, \theta) - Y\right)^2 = \left(H^k - Y\right)^2$$

H^i is a random variable that depends on X

We can then think about $\frac{\partial}{\partial \theta} l(X, Y, \theta)$ to optimize the weights (more on this later)

LINEAR NEURAL NETWORKS

THE SIMPLEST NETWORK

For linear neural networks, f^i is just a linear function of the inputs, e.g.,

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

Where $W^i \in \mathbb{R}^{n_i \times n_{i-1}}$

We treat $x = h^0, h^1, h^2, \ldots, h^k$ as row vectors instead of column vectors

- $h^i \in \mathbb{R}^{1 \times n_i}$
- this is for code optimization reasons (implementation varies)
- if column vectors, then $f^i(h^{i-1}, W^i) = W^i h^{i-1}$

LINEAR NEURAL NETWORKS

THE SIMPLEST NETWORK

$$
\begin{aligned}
f^1\left(h^0, W^1\right) &= h^0 {W^1}^\top \\
&= x{W^1}^\top \\
&= \begin{bmatrix} x_1 & x_2 & \cdots & x_{n_0} \end{bmatrix} \begin{bmatrix} W^1_{1,1} & W^1_{1,2} & \cdots & W^1_{1,n_0} \\ W^1_{2,1} & W^1_{2,2} & \cdots & W^1_{2,n_0} \\ \vdots & \vdots & \ddots & \vdots \\ W^1_{n_1,1} & W^1_{n_1,2} & \cdots & W^1_{n_1,n_0} \end{bmatrix}^\top
\end{aligned}
$$

LINEAR NEURAL NETWORKS

THE SIMPLEST NETWORK

$$
\begin{aligned}
f^1\left(h^0, W^1\right) &= h^0 {W^1}^\top \\
&= x {W^1}^\top \\
&= \begin{bmatrix} x_1 & x_2 & \cdots & x_{n_0} \end{bmatrix} \begin{bmatrix} W^1_{1,1} & W^1_{1,2} & \cdots & W^1_{n_1,1} \\ W^1_{1,2} & W^1_{2,2} & \cdots & W^1_{n_1,2} \\ \vdots & \vdots & \ddots & \vdots \\ W^1_{1,n_0} & W^1_{2,n_0} & \cdots & W^1_{n_1,n_0} \end{bmatrix}
\end{aligned}
$$

LINEAR NEURAL NETWORKS

THE SIMPLEST NETWORK

$$
\begin{aligned}
f^1\left(h^0, W^1\right) &= h^0 {W^1}^\top \\
&= x {W^1}^\top \\
&= \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n_0} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n_0} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n_0} \end{bmatrix} \begin{bmatrix} W^1_{1,1} & W^1_{1,2} & \cdots & W^1_{n_1,1} \\ W^1_{1,2} & W^1_{2,2} & \cdots & W^1_{n_1,2} \\ \vdots & \vdots & \ddots & \vdots \\ W^1_{1,n_0} & W^1_{2,n_0} & \cdots & W^1_{n_1,n_0} \end{bmatrix}
\end{aligned}
$$

LINEAR NEURAL NETWORKS

THE SIMPLEST NETWORK

We can represent the whole network with just a single layer for this special network.

$$k = 2$$

$$h^1 = x{W^1}^\top$$

$$h^2 = h^1{W^2}^\top = x{W^1}^\top{W^2}^\top$$

LINEAR NEURAL NETWORKS

THE SIMPLEST NETWORK

We can represent the whole network with just a single layer for this special network.

$k = 2$

$h^1 = x{W^1}^\top$

$h^2 = h^1{W^2}^\top = x{W^1}^\top{W^2}^\top$

$A = W^2W^1$

$h^2 = xA^\top$ — this is just a single linear layer with weights A

For k layers, we have $A = W^kW^{k-1}\ldots W^2W^1$

LINEAR NEURAL NETWORKS

THE SIMPLEST NETWORK

This transformation tells us that no matter how many layers we add, the network will not become any more expressive (able to represent more functions).

This shows that we need some form of nonlinearity between layers if we want a useful network.

Note: Linear networks are used in theory-based research because they are easier to analyze and can provide some insights into what neural networks do or how they are trained.

NEURAL NETWORK LAYERS

MULTILAYER PERCEPTION

The most standard layer in a neural network is called a *Dense* layer

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

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

There is also often a bias term added before the activation function

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

Where $b^i \in \mathbb{R}^{n^i}$. This term is optional, and its efficacy has been debated.

A network of just these layers is called a *multilayer perceptron* (MLP) or a *Dense Network* (more modern)

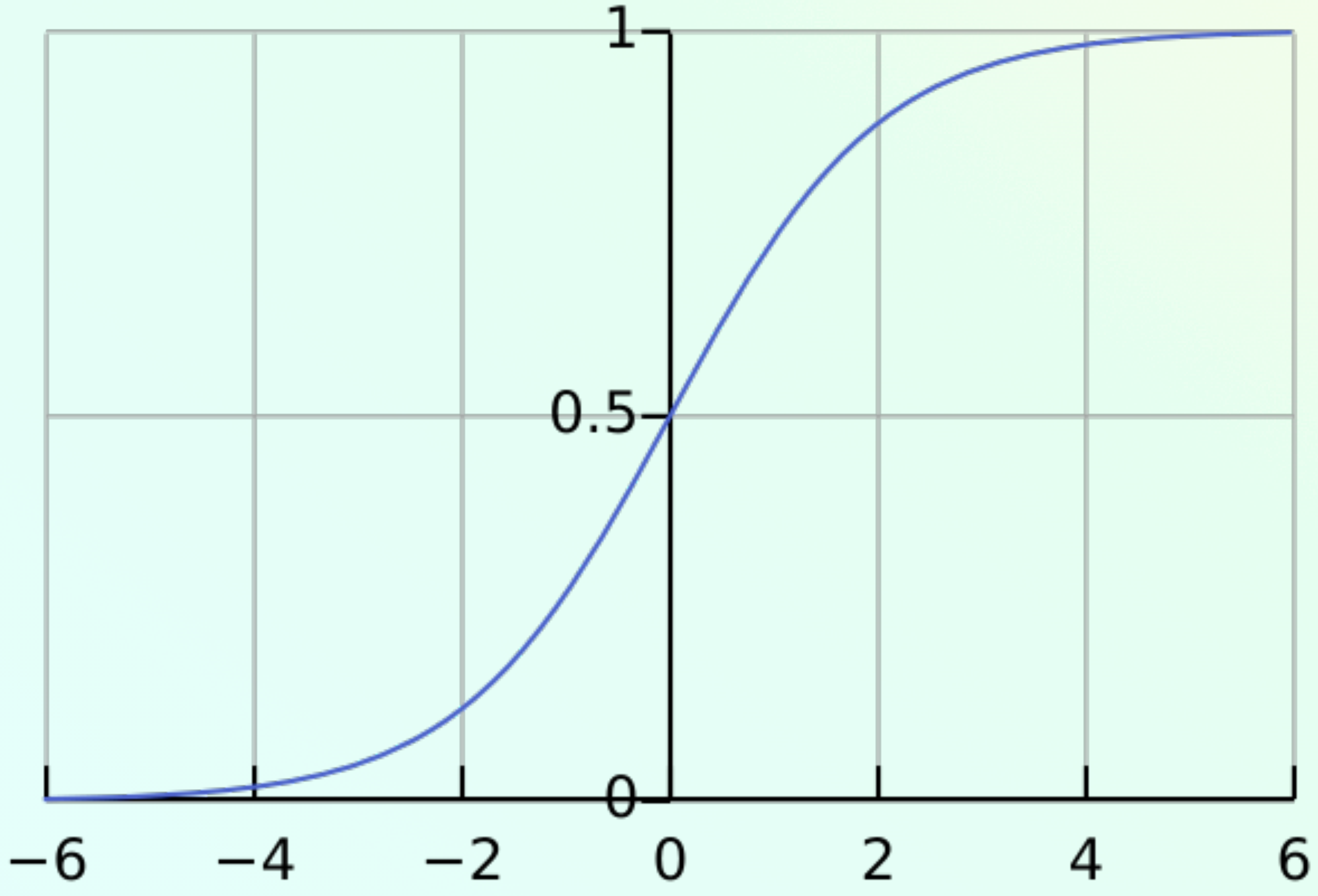
ACTIVATION FUNCTION

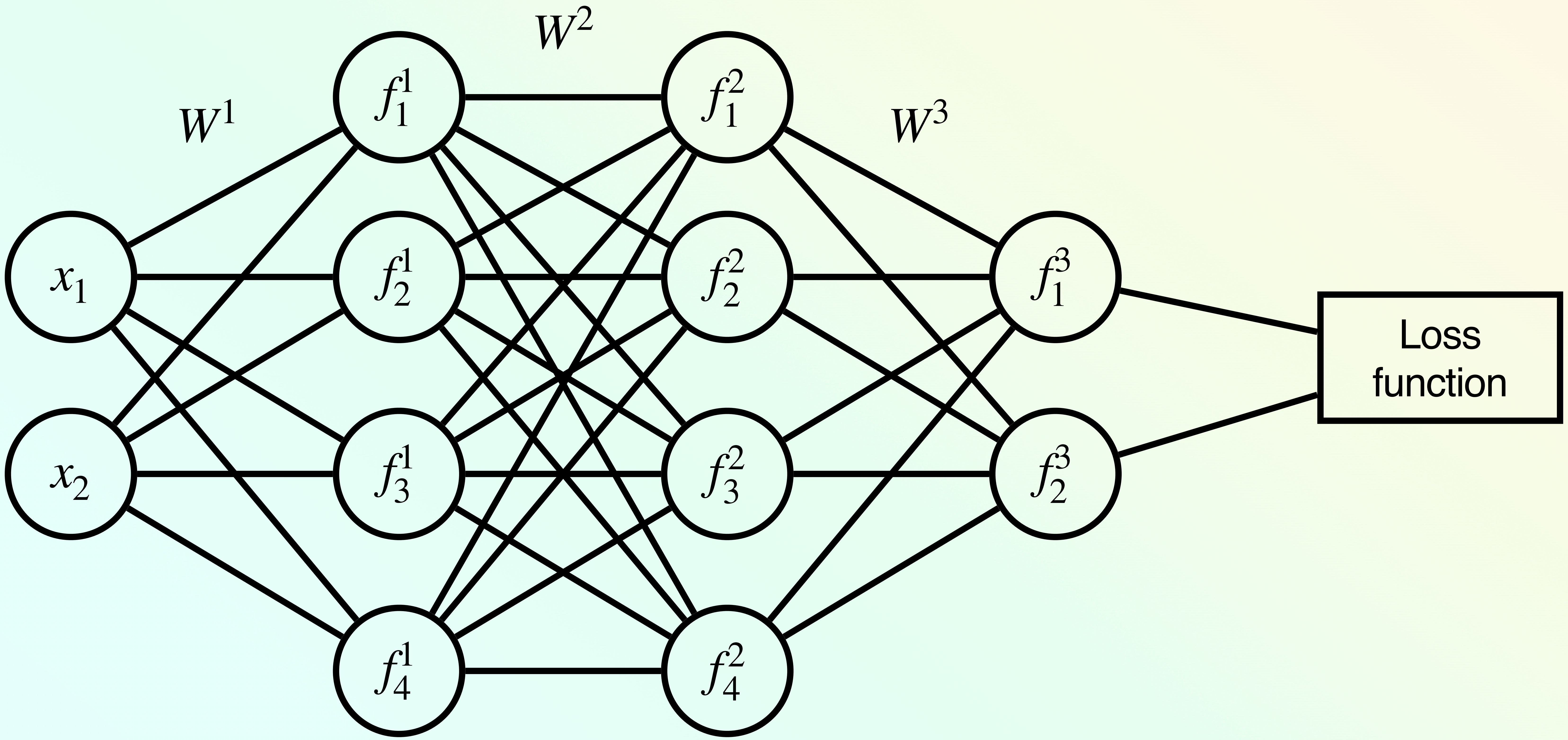
SIGMOID

There are many different activation functions.

Historically, the most common is the sigmoid, which we used in logistic regression.

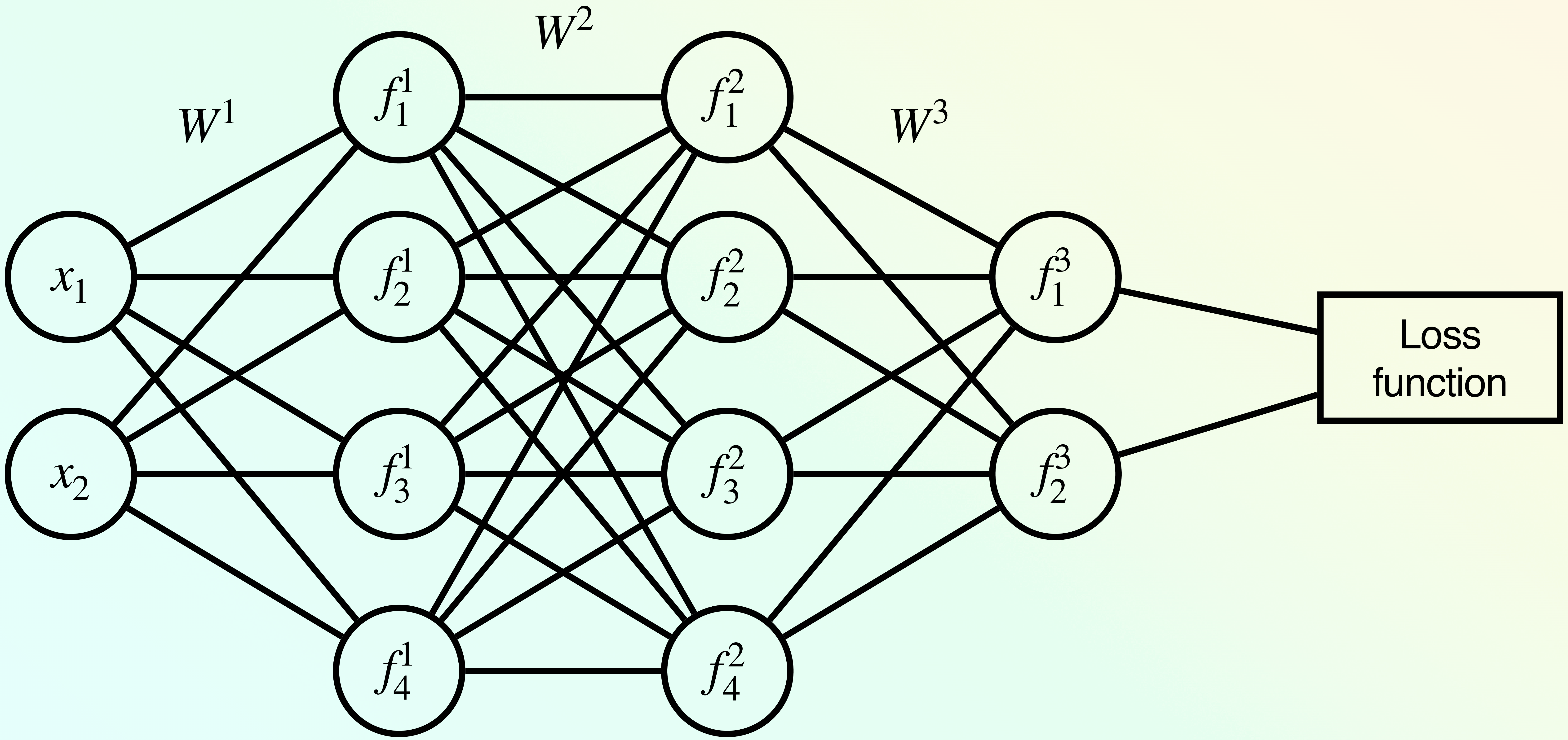
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



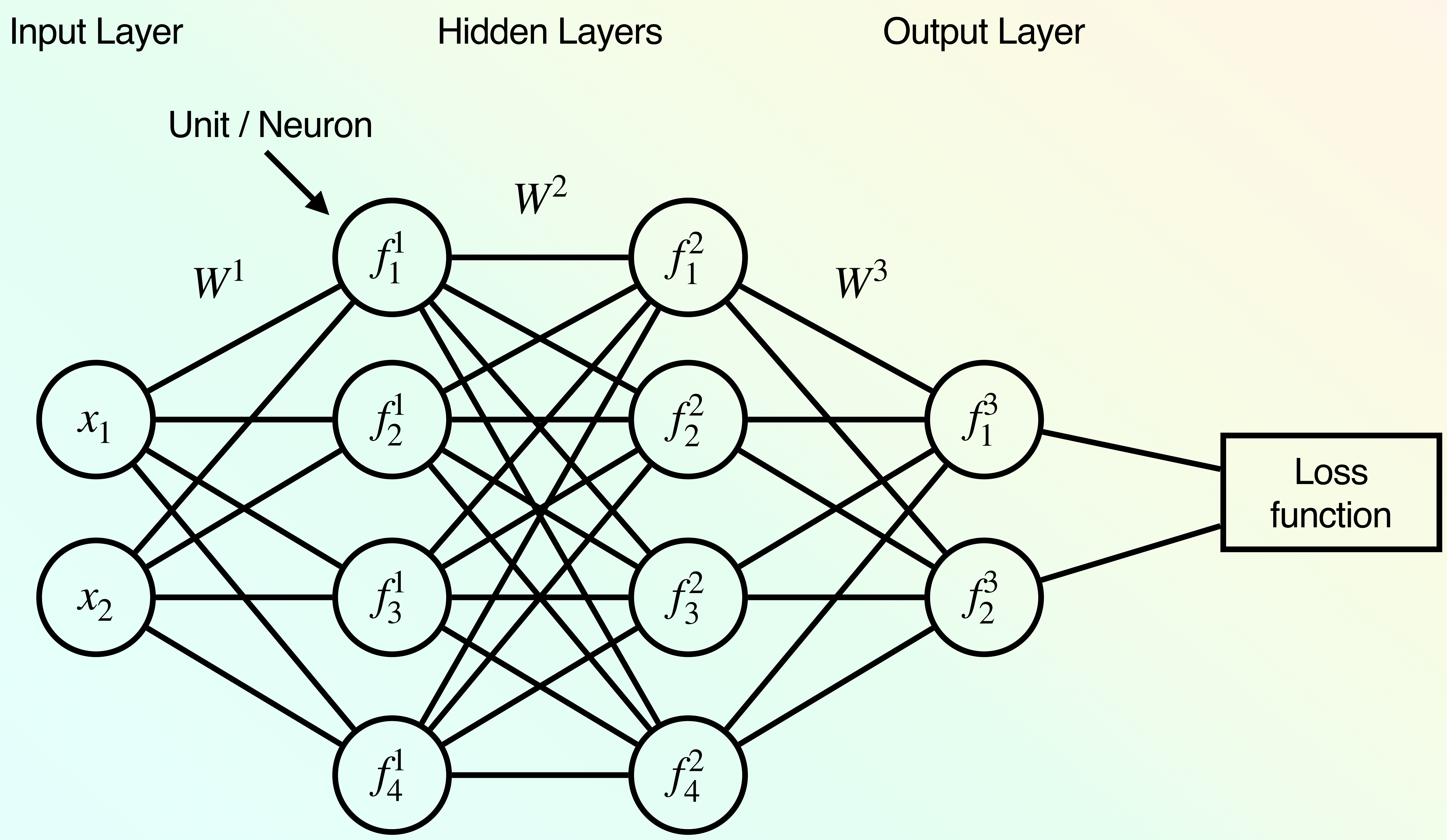


h^0 $\quad h^1 = f^1(h^0, W^1) \quad h^2 = f^2(h^1, W^2) \quad h^3 = f^3(h^2, W^3)$

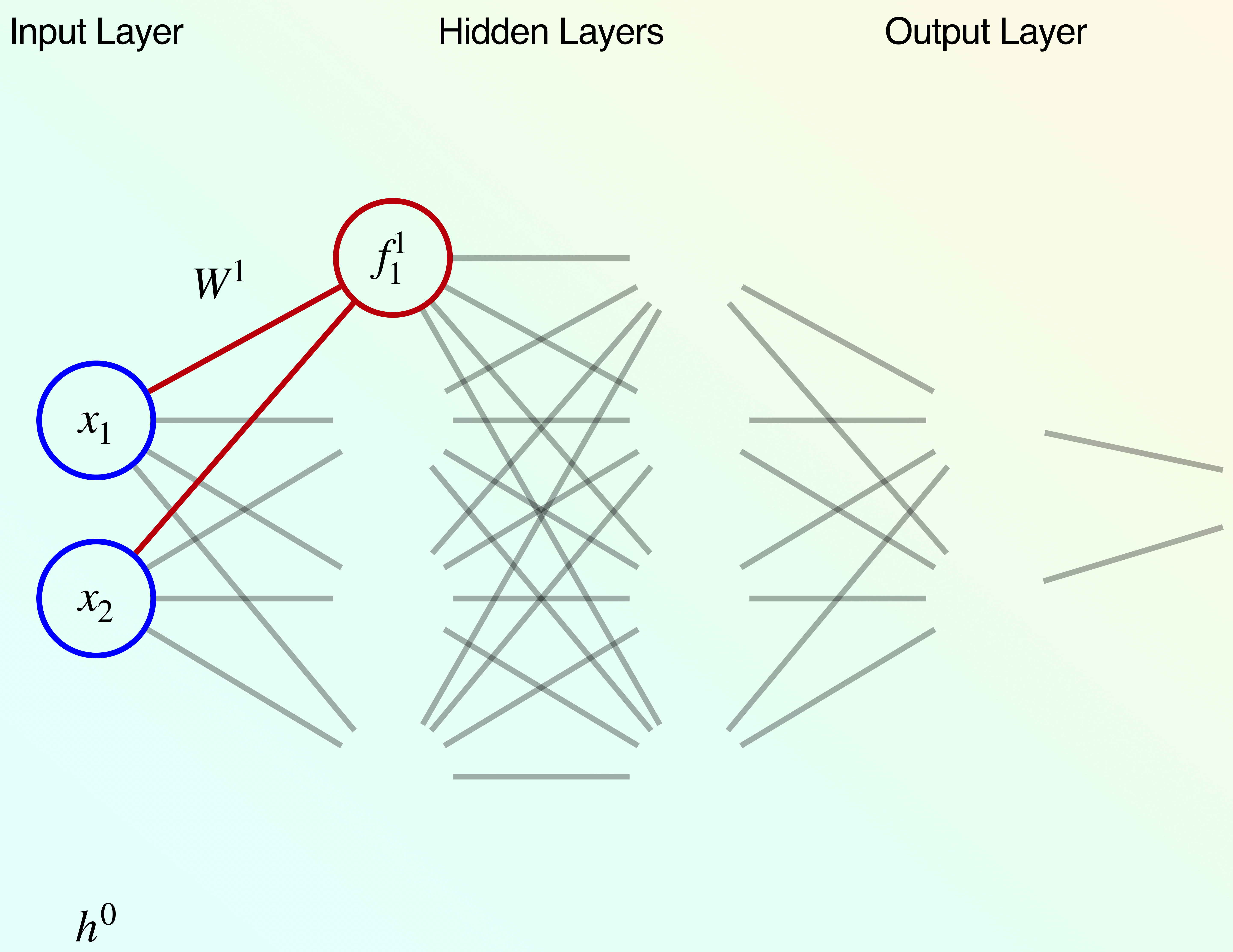
Input Layer　　Hidden Layers　　Output Layer



h^0　　$h^1 = f^1(h^0, W^1)$　　$h^2 = f^2(h^1, W^2)$　　$h^3 = f^3(h^2, W^3)$

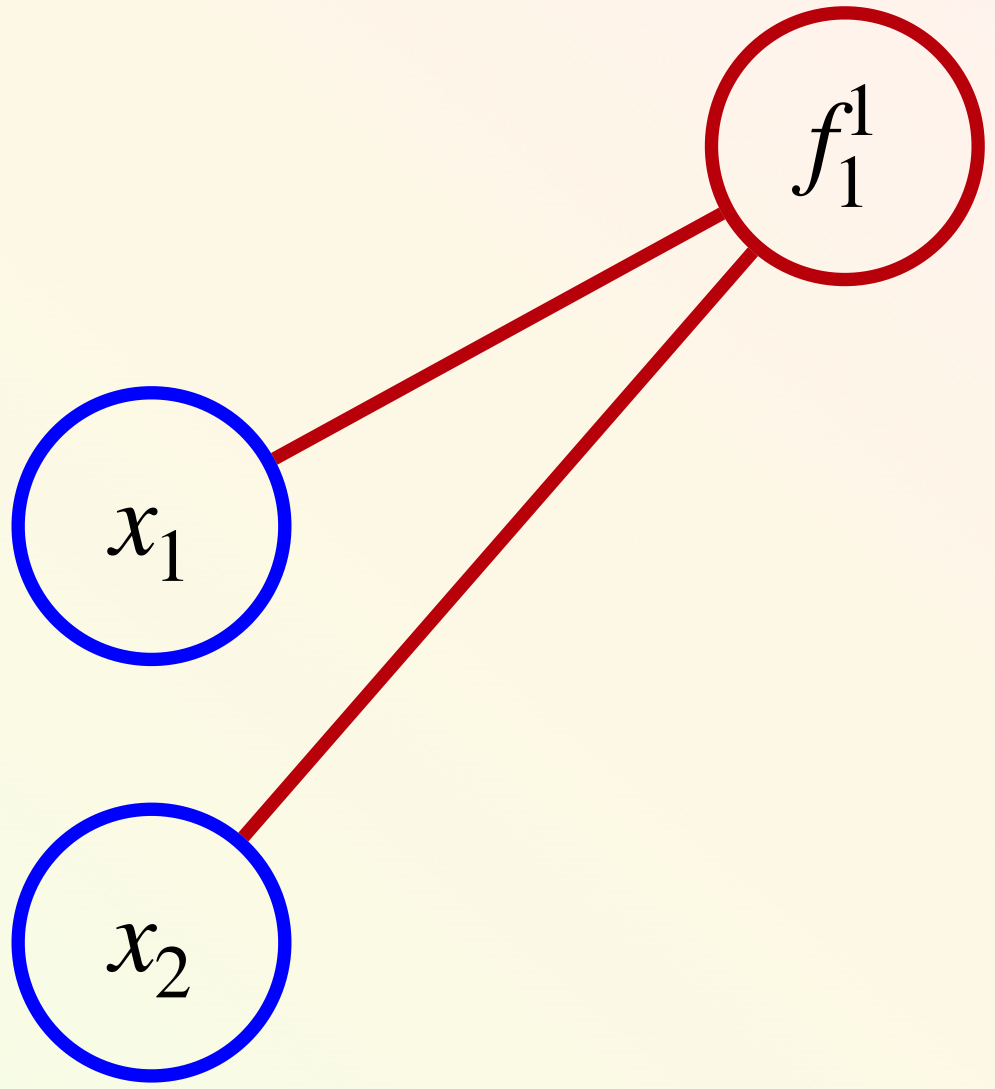


h^0 $\quad h^1 = f^1(h^0, W^1) \quad h^2 = f^2(h^1, W^2) \quad h^3 = f^3(h^2, W^3)$


Input Layer
Hidden Layers
Output Layer
W^1
f_1^1
x_1
x_2
h^0

SINGLE NEURAL UNIT

BASIC OPERATION

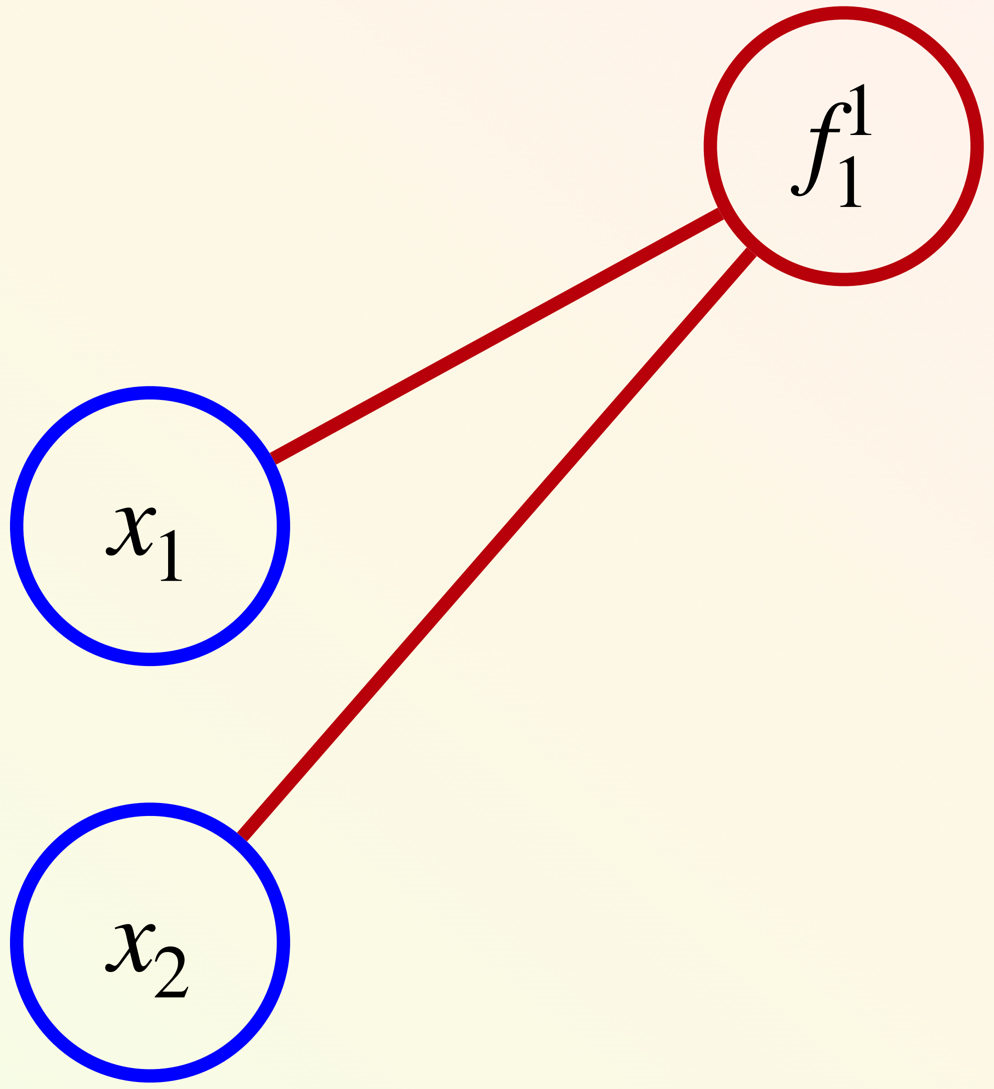


$f^1(x, W^1)$ – want the first output of the first layer

$f_1^1 : \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \to \mathbb{R}$ – takes in two vectors as input

$$f_1^1(x, w^1) = \sigma\left(\sum_{i=1}^{n_0} x_i w_i^1\right)$$

SINGLE NEURAL UNIT



BASIC OPERATION

$f^1(x, W^1)$ — want the first output of the first layer

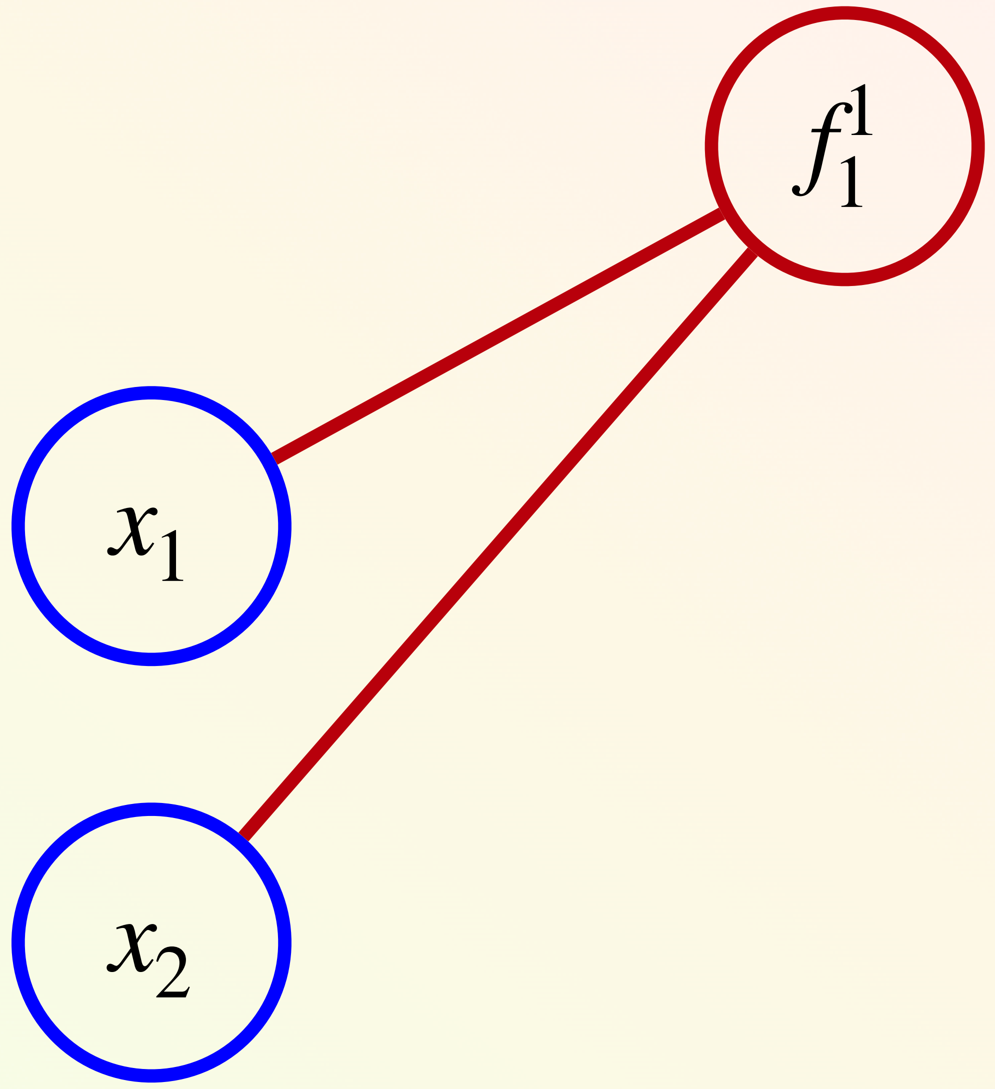
$f_1^1 : \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \to \mathbb{R}$ — takes in two vectors as input

$$f_1^1(x, w^1) = \sigma\left(\sum_{i=1}^{n_0} x_i w_i^1\right)$$

Each weight can be thought of as a synapse connecting the input neurons x to the output neuron f_1^1 — extreme simplification of neurons

SINGLE NEURAL UNIT

BASIC OPERATION



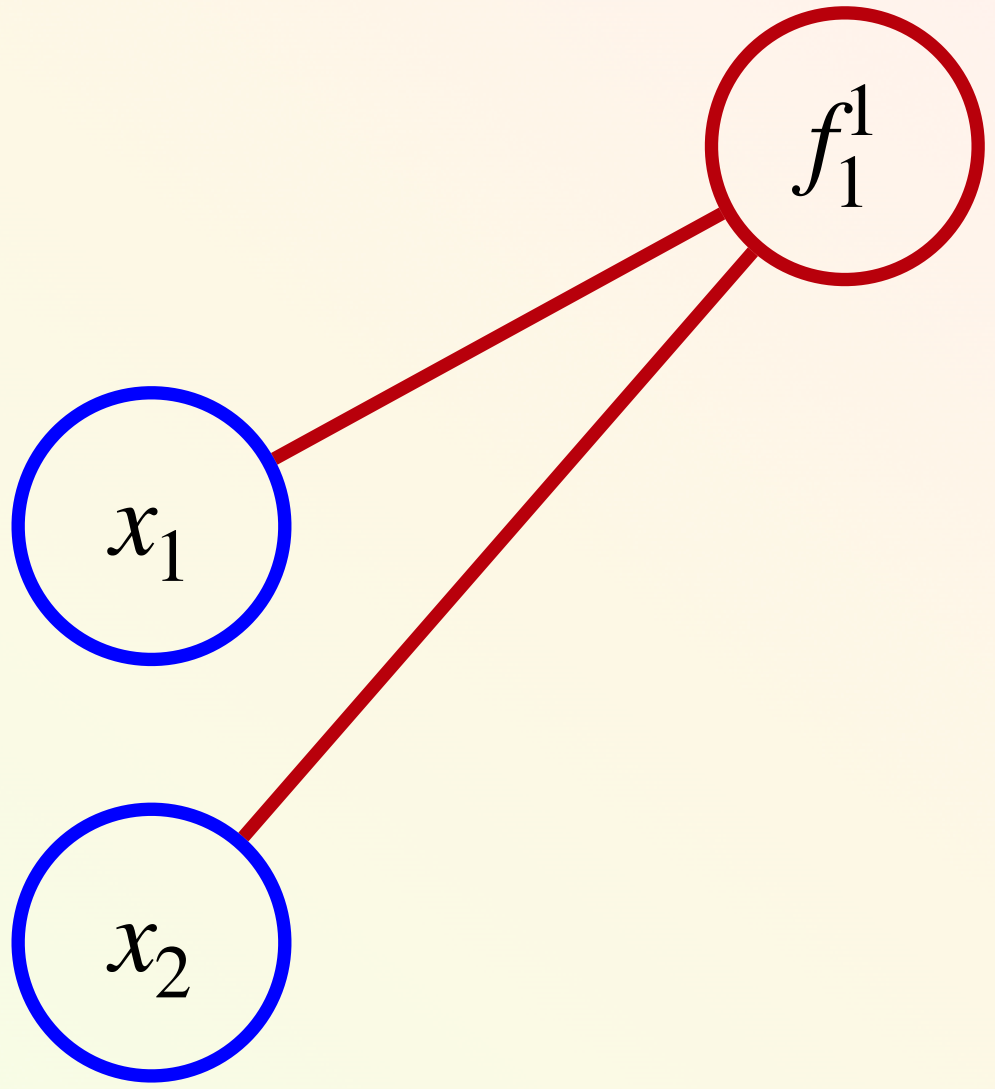
$f^1(x, W^1)$ – want the first output of the first layer

$f_1^1 : \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \to \mathbb{R}$ – takes in two vectors as input

$$f_1^1(x, w^1) = \sigma\left(\sum_{i=1}^{n_0} x_i w_i^1\right) = \sigma\left(x^\top w^1\right)$$

SINGLE NEURAL UNIT

BASIC OPERATION



$f^1(x, W^1)$ – want the first output of the first layer

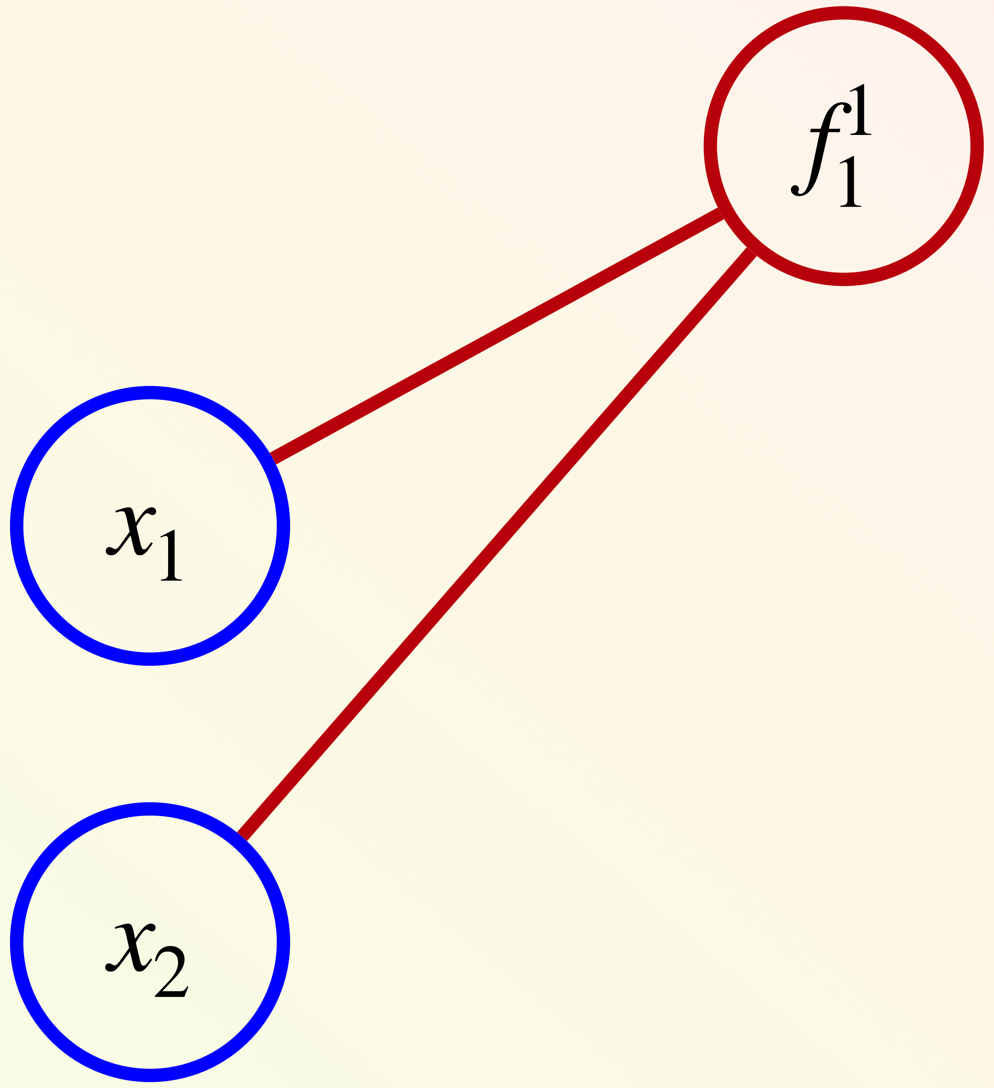
$f_1^1 : \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \to \mathbb{R}$ – takes in two vectors as input

$$f_1^1(x, w^1) = \sigma\left(\sum_{i=1}^{n_0} x_i w_i^1\right) = \sigma\left(x^\top w^1\right) = \frac{1}{1 + e^{-x^\top w^1}}$$

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

SINGLE NEURAL UNIT

BASIC OPERATION



To compute output:

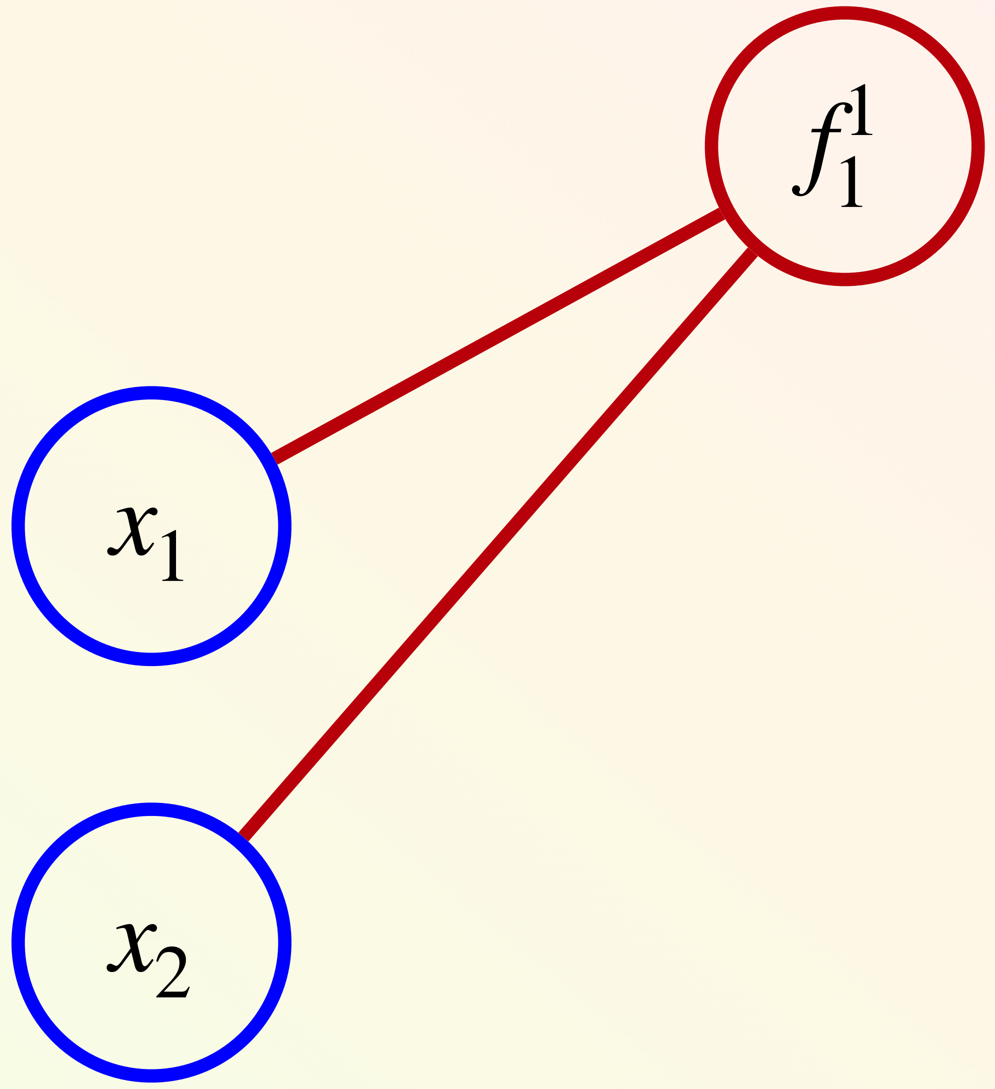
$$z_1 = x^\top w^1$$

$$h_1^1 = \sigma(z)$$

return h_1^1

Repeat for each output unit

SINGLE NEURAL UNIT



BASIC OPERATION

To compute output:

$$z_1 = x^\top w^1$$

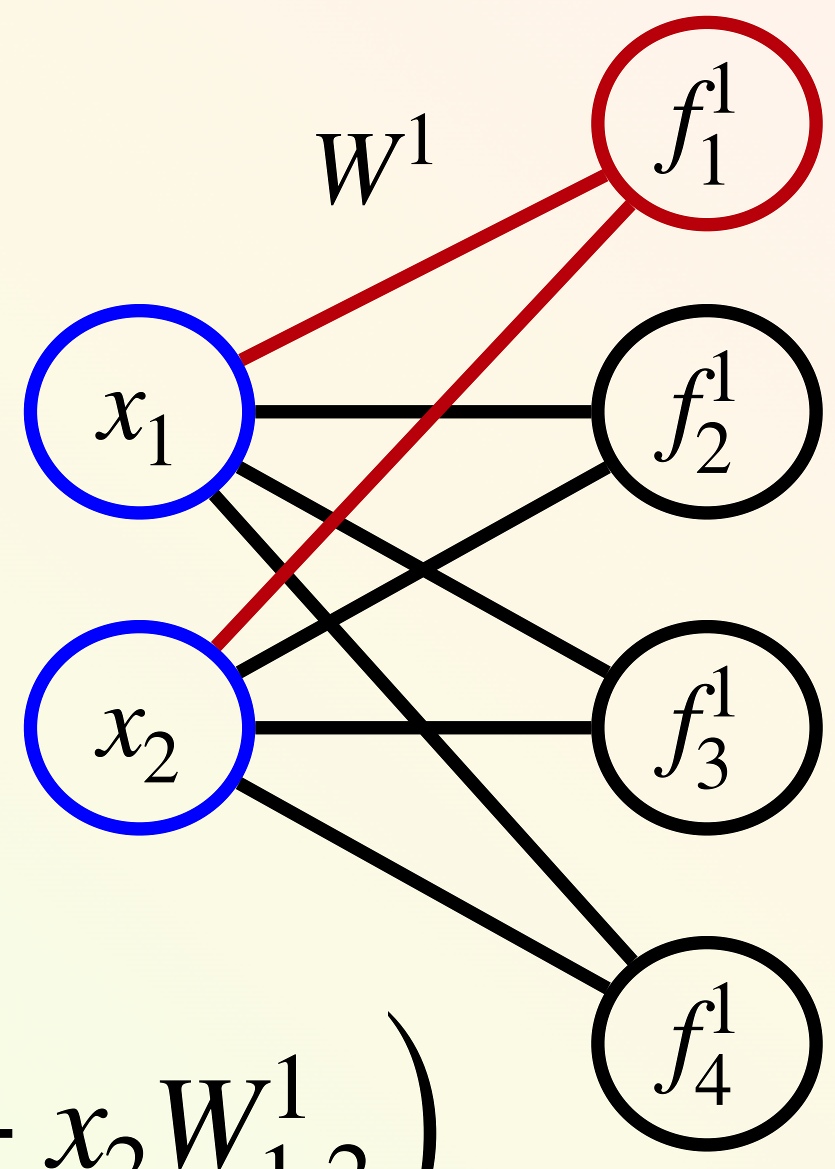
$$h_1^1 = \sigma(z)$$

return h_1^1

Repeat for each output unit

What weighs to use for w^i?

LAYER OUTPUTS



BASIC OPERATION

$$W^1 \in \mathbb{R}^{n_1 \times n_0}$$

$$W^1 = \begin{bmatrix} W^1_{1,1} & W^1_{1,2} \\ W^1_{2,1} & W^1_{2,2} \\ W^1_{3,1} & W^1_{3,2} \\ W^1_{4,1} & W^1_{4,2} \end{bmatrix} = \begin{bmatrix} {w^1}^\top \\ {w^2}^\top \\ {w^3}^\top \\ {w^4}^\top \end{bmatrix}$$

$$f_1^1(x, w^1) = \sigma(x^\top w^1) = \sigma\left(x_1 W^1_{1,1} + x_2 W^1_{1,2}\right)$$

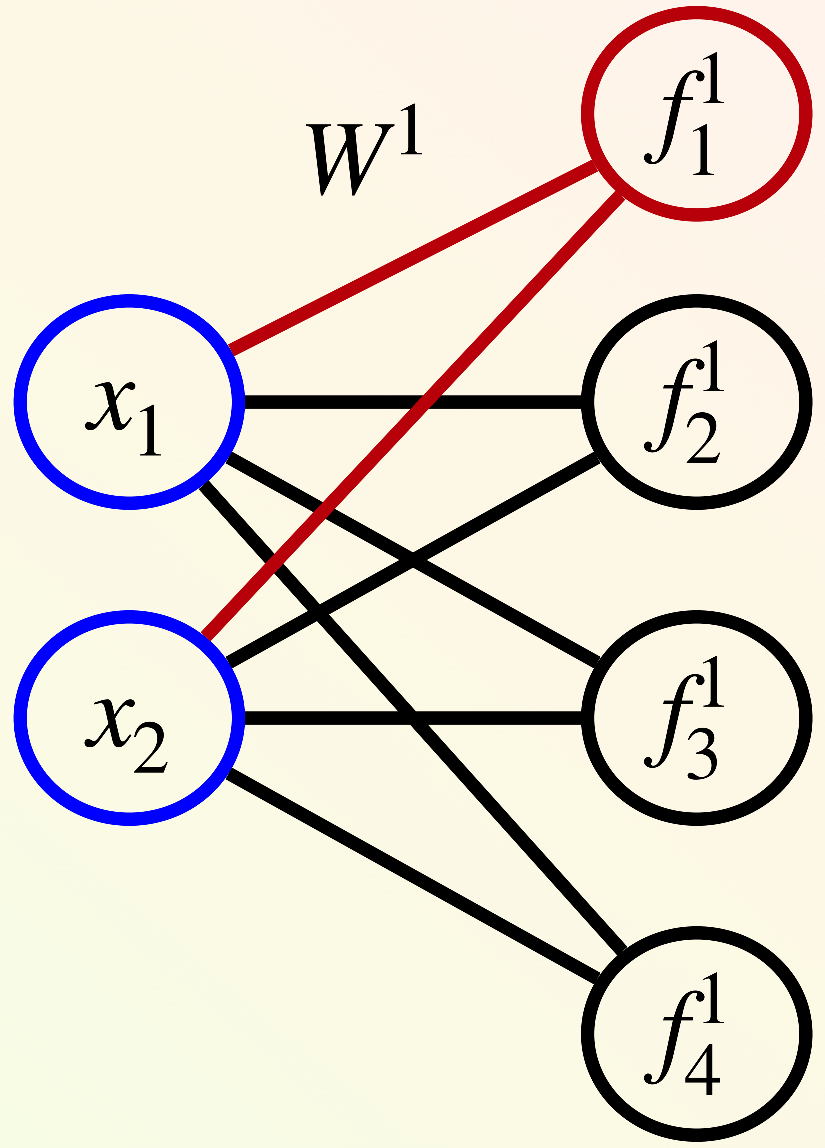
$$f_2^1(x, w^2) = \sigma(x^\top w^2) = \sigma\left(x_1 W^1_{2,1} + x_2 W^1_{2,2}\right)$$

$$f_3^1(x, w^3) = \sigma(x^\top w^3) = \sigma\left(x_1 W^1_{3,1} + x_2 W^1_{3,2}\right)$$

$$f_3^1(x, w^4) = \sigma(x^\top w^4) = \sigma\left(x_1 W^1_{4,1} + x_2 W^1_{4,2}\right)$$

LAYER OUTPUTS

BASIC OPERATION



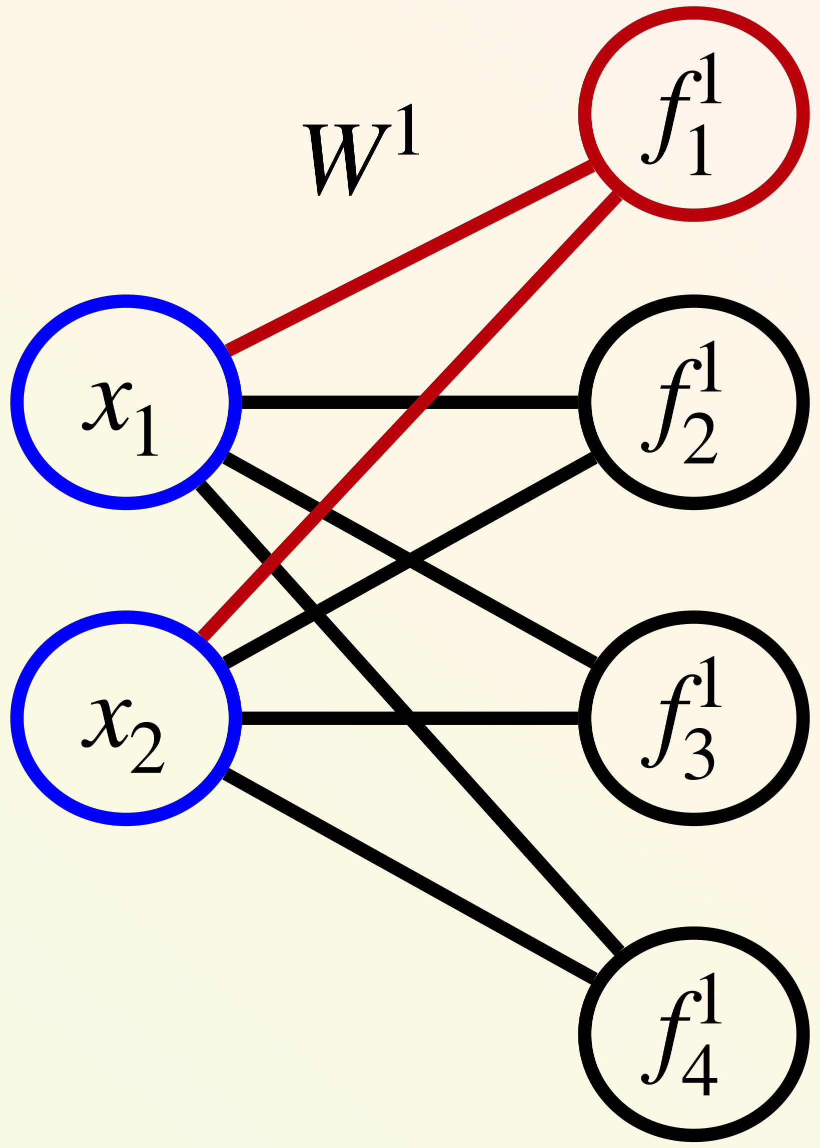
$$W^1 \in \mathbb{R}^{n_1 \times n_0}$$

$$W^1 = \begin{bmatrix} W^1_{1,1} & W^1_{1,2} \\ W^1_{2,1} & W^1_{2,2} \\ W^1_{3,1} & W^1_{3,2} \\ W^1_{4,1} & W^1_{4,2} \end{bmatrix} = \begin{bmatrix} {w^1}^\top \\ {w^2}^\top \\ {w^3}^\top \\ {w^4}^\top \end{bmatrix}$$

$$f_1^1(x, w^1) = \sigma(x^\top w^1) = \sigma\left(z_1\right)$$
$$f_2^1(x, w^2) = \sigma(x^\top w^2) = \sigma\left(z_2\right)$$
$$f_3^1(x, w^3) = \sigma(x^\top w^3) = \sigma\left(z_3\right)$$
$$f_3^1(x, w^4) = \sigma(x^\top w^4) = \sigma\left(z_4\right)$$

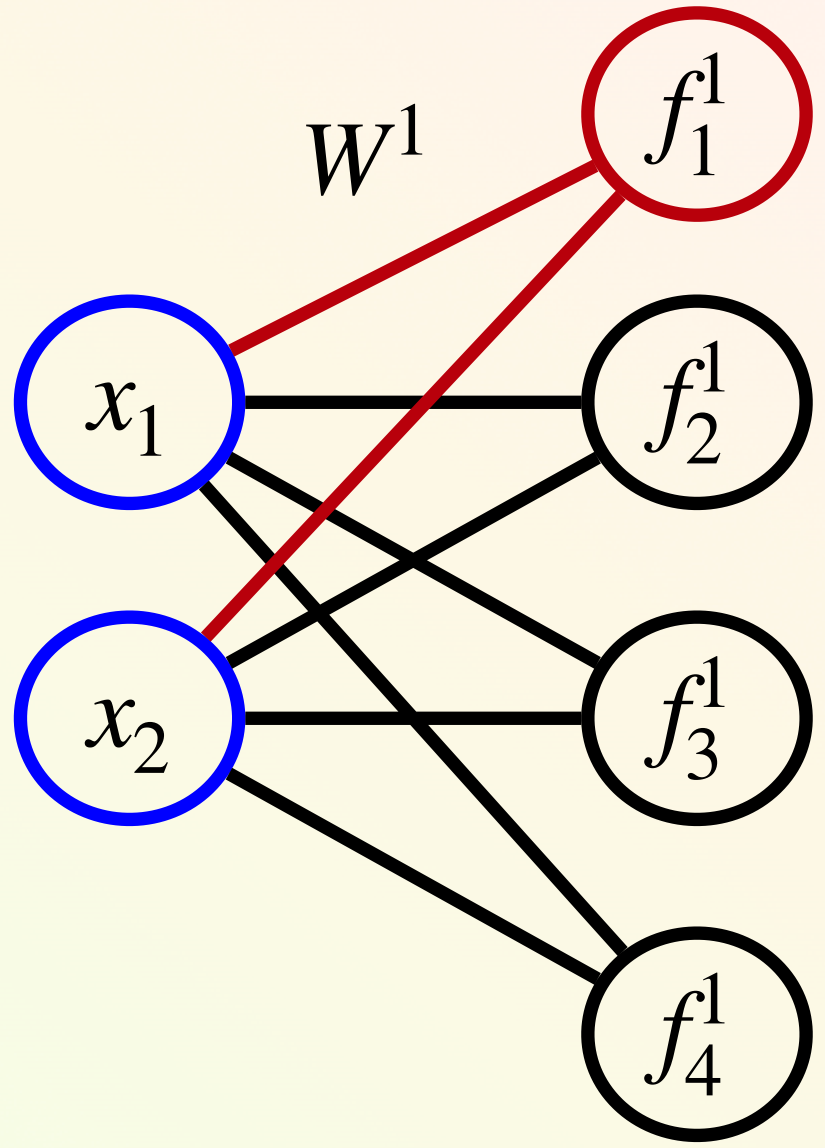
LAYER OUTPUTS

BASIC OPERATION



$$z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} x_1 W^1_{1,1} + x_2 W^1_{1,2} \\ x_1 W^1_{2,1} + x_2 W^1_{2,2} \\ x_1 W^1_{3,1} + x_2 W^1_{3,2} \\ x_1 W^1_{4,1} + x_2 W^1_{4,2} \end{bmatrix} = [x_1, x_2] \begin{bmatrix} W^1_{1,1} & W^1_{2,1} & W^1_{3,1} & W^1_{4,1} \\ W^1_{1,2} & W^1_{2,2} & W^1_{3,2} & W^1_{4,2} \end{bmatrix} = x {W^1}^\top$$

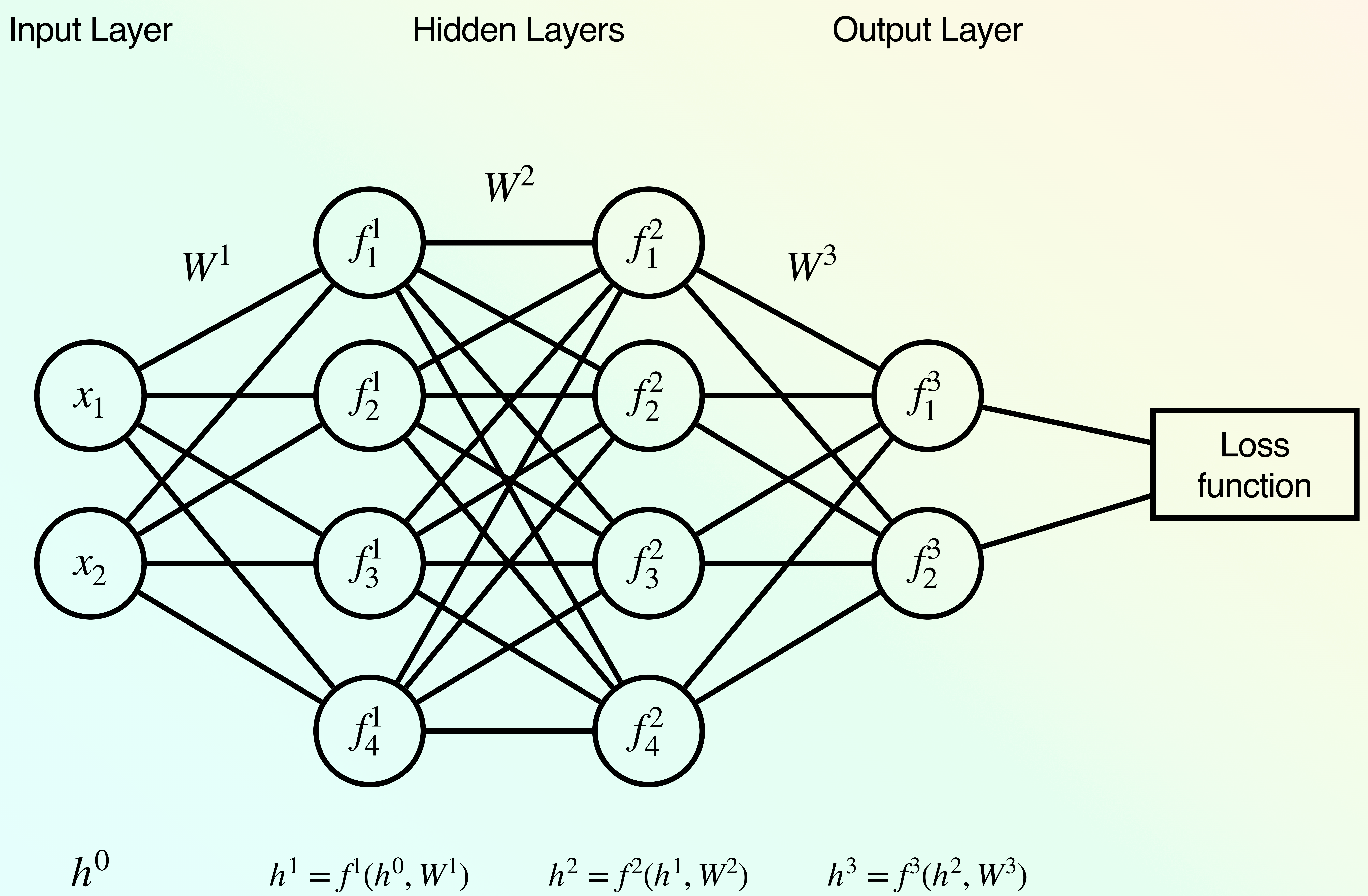
LAYER OUTPUTS



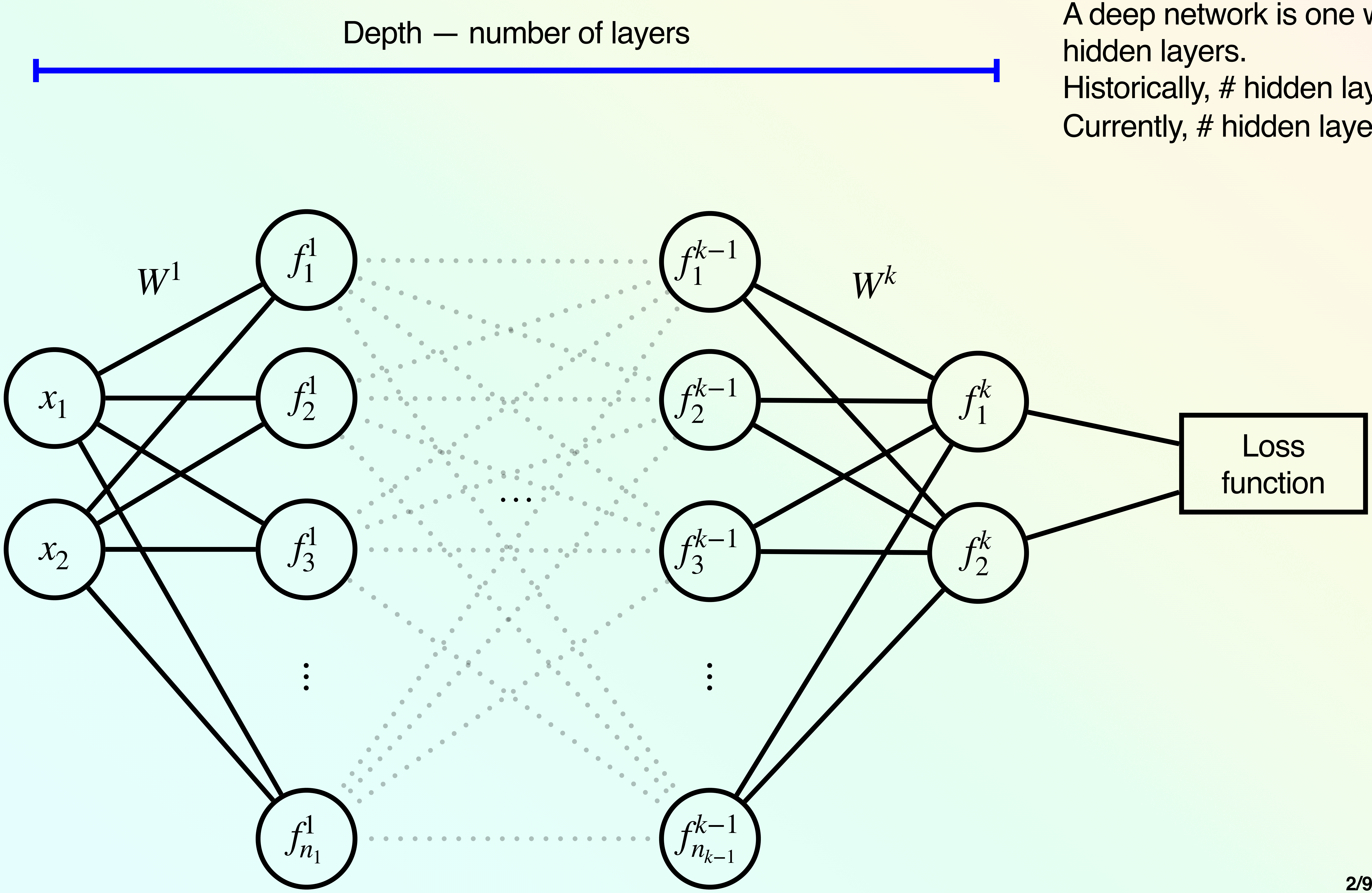
BASIC OPERATION

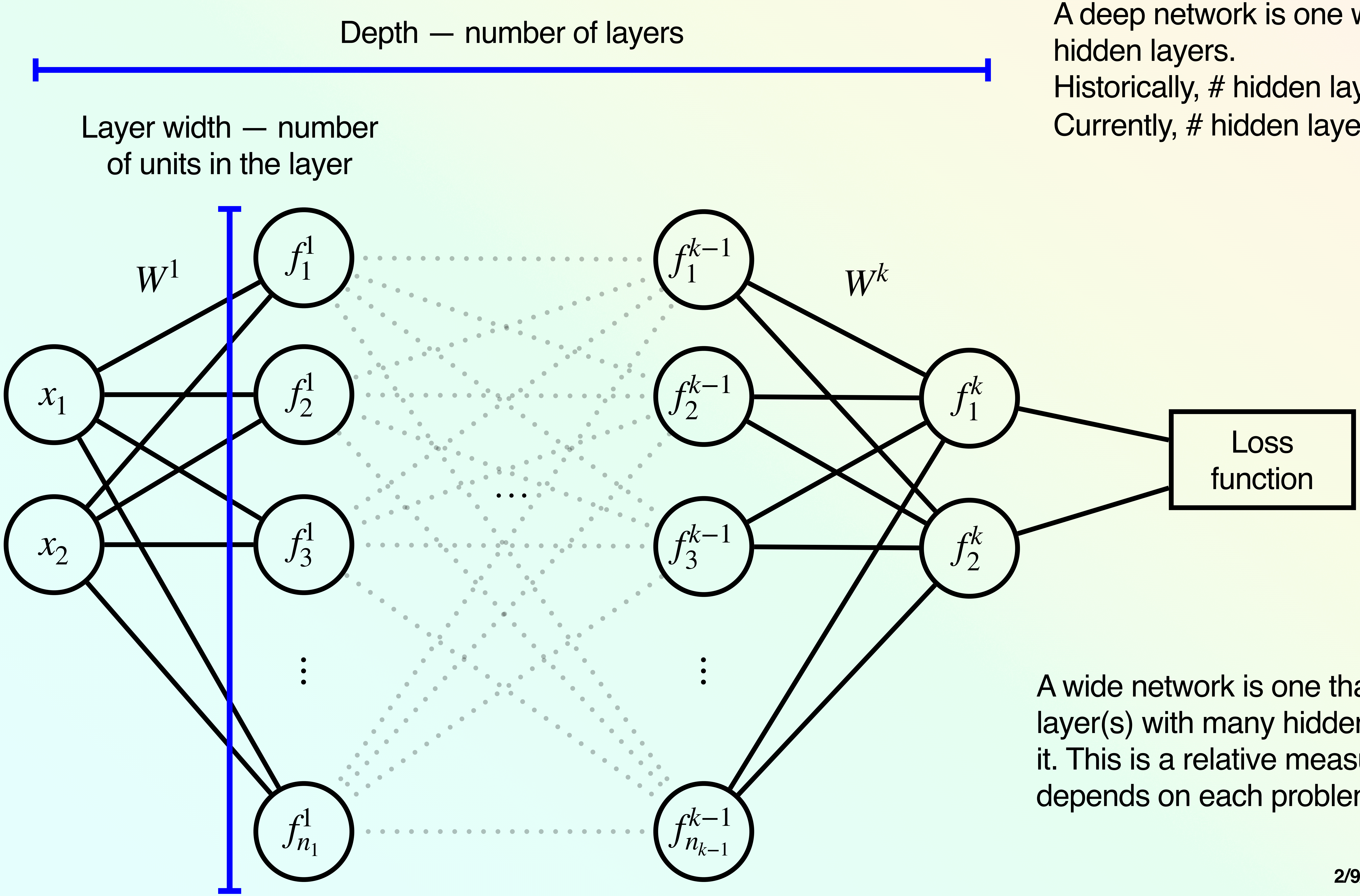
$$z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} x_1 W^1_{1,1} + x_2 W^1_{1,2} \\ x_1 W^1_{2,1} + x_2 W^1_{2,2} \\ x_1 W^1_{3,1} + x_2 W^1_{3,2} \\ x_1 W^1_{4,1} + x_2 W^1_{4,2} \end{bmatrix} = [x_1, x_2] \begin{bmatrix} W^1_{1,1} & W^1_{2,1} & W^1_{3,1} & W^1_{4,1} \\ W^1_{1,2} & W^1_{2,2} & W^1_{3,2} & W^1_{4,2} \end{bmatrix} = x {W^1}^\top$$

$f^1(x, W^1) = \sigma\left(x{W^1}^\top\right)$ – compute them in all one linear algebra operation


Input Layer
Hidden Layers
Output Layer
W^1
W^2
W^3
x_1
x_2
f_1^1
f_2^1
f_3^1
f_4^1
f_1^2
f_2^2
f_3^2
f_4^2
f_1^3
f_2^3
Loss function
h^0
$h^1 = f^1(h^0, W^1)$
$h^2 = f^2(h^1, W^2)$
$h^3 = f^3(h^2, W^3)$

A deep network is one with many hidden layers.
Historically, # hidden layers ≥ 2
Currently, # hidden layers ≥ 10?





A deep network is one with many hidden layers.
Historically, # hidden layers ≥ 2
Currently, # hidden layers $\geq 10?$

A wide network is one that has layer(s) with many hidden units in it. This is a relative measure at is depends on each problem.

Quiz

OUTPUT UNITS

$$h^k = f^k(h^{k-1}, W^k) = f(x, \theta)$$

Need to have compatible loss functions and network outputs h^k

mean squared error: $l(x, y, \theta) = \left(f(x, \theta) - y\right)^2$

Negative log-likelihood: $l(x, y, \theta) = -\ln \Pr(Y = y \,|\, X = x)$

OUTPUT UNITS

Mean squared error:

$y \in \mathbb{R}$

$f^k(h^{k-1}, W^k) = h^{k-1}W^{k^\top} = h^k$ — linear layer with $W^k \in \mathbb{R}^{1 \times n_{k-1}}$ — one output unit

OUTPUT UNITS

Mean squared error:

$y \in \mathbb{R}^{n_y}$ — multiple scalars to predict

$W^k \in \mathbb{R}^{n_y \times n_{k-1}}$ — n_y output units for f^k

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

OUTPUT UNITS

Negative log-likelihood

$y \in \{0,1\}$ – binary classification

$W^k \in \mathbb{R}^{1 \times n_{k-1}}$ —one output unit

$f^k(h^{k-1}, W^k) = \sigma(h^{k-1} W^{k^\top})$ – σ is sigmoid

OUTPUT UNITS

Negative log-likelihood

$y \in \{0,1\}$ – binary classification

$W^k \in \mathbb{R}^{1 \times n_{k-1}}$ —one output unit

$f^k(h^{k-1}, W^k) = \sigma(h^{k-1} W^{k^\top})$ – σ is sigmoid

$h^k = \Pr(Y = 1 \mid X = x)$

$l(x, y, \theta) = y \ln h^k + (1 - y) \ln(1 - h^k)$ —same as the linear classifier

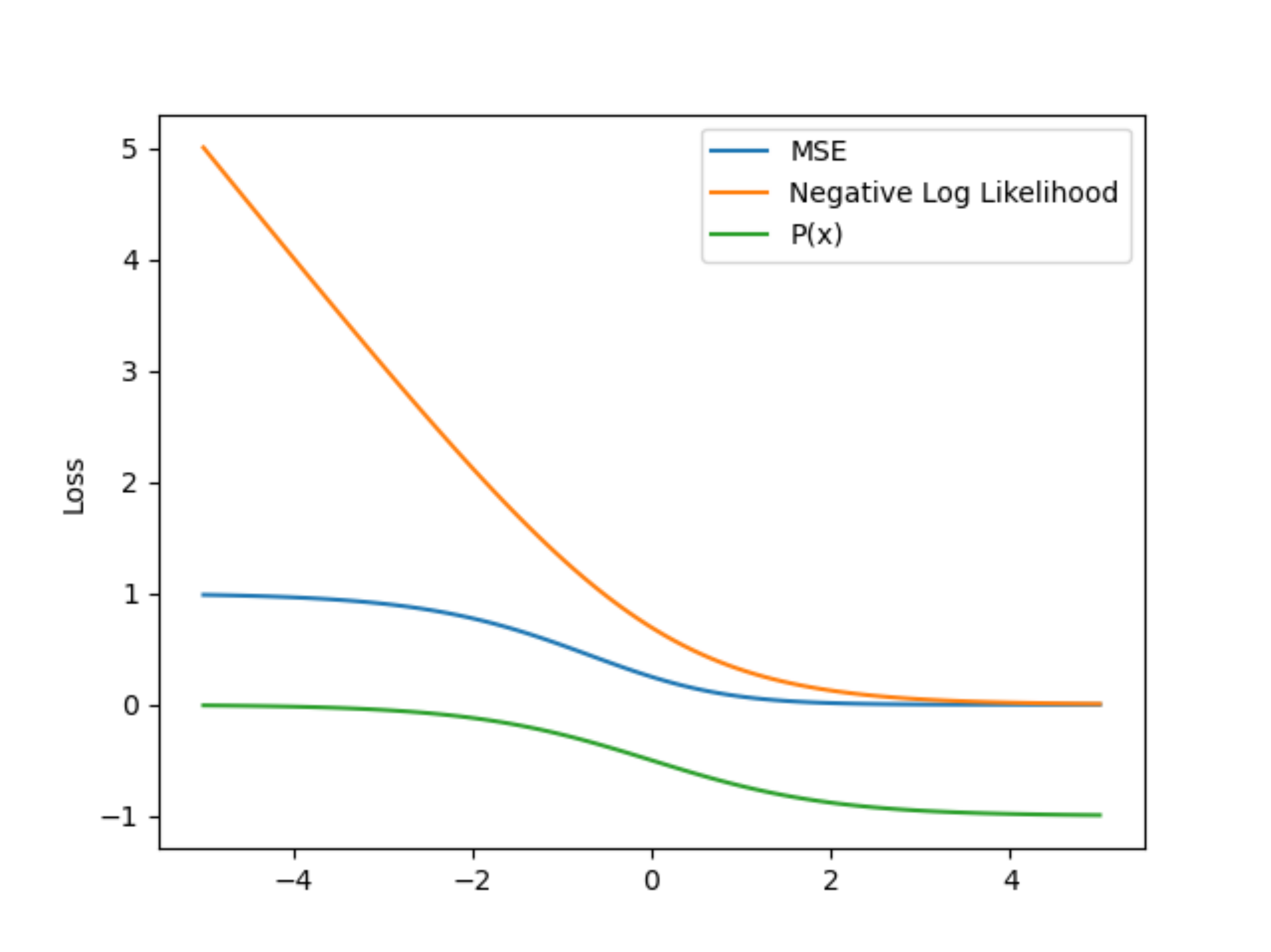
OUTPUT UNITS

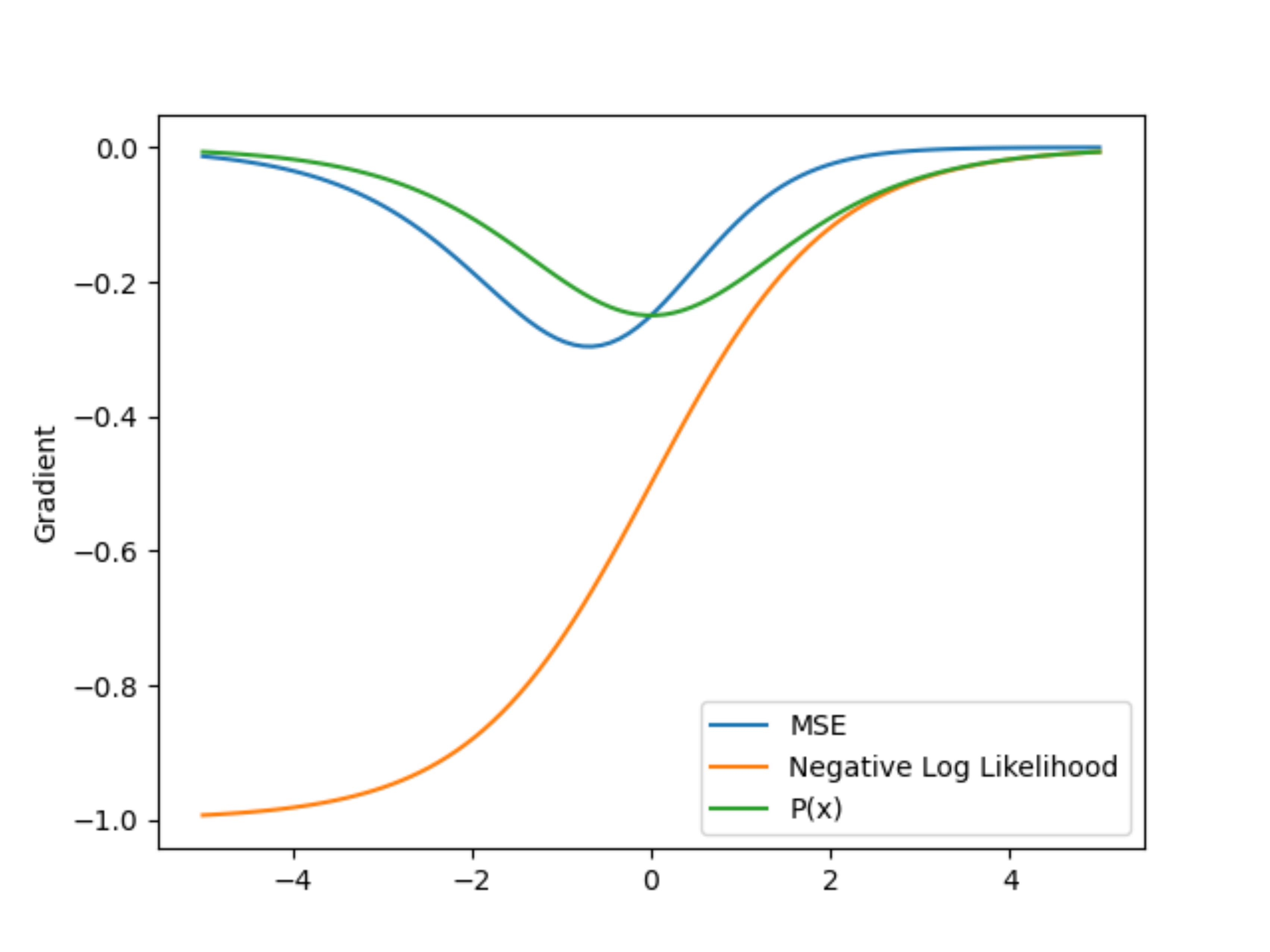
What about MSE with sigmoid for classification?

Or using $\Pr(Y = y \,|\, X = x)$ instead of $\ln \Pr(Y = y \,|\, X = x)$?

OUTPUT UNITS

Loss and gradient for $y = 1$



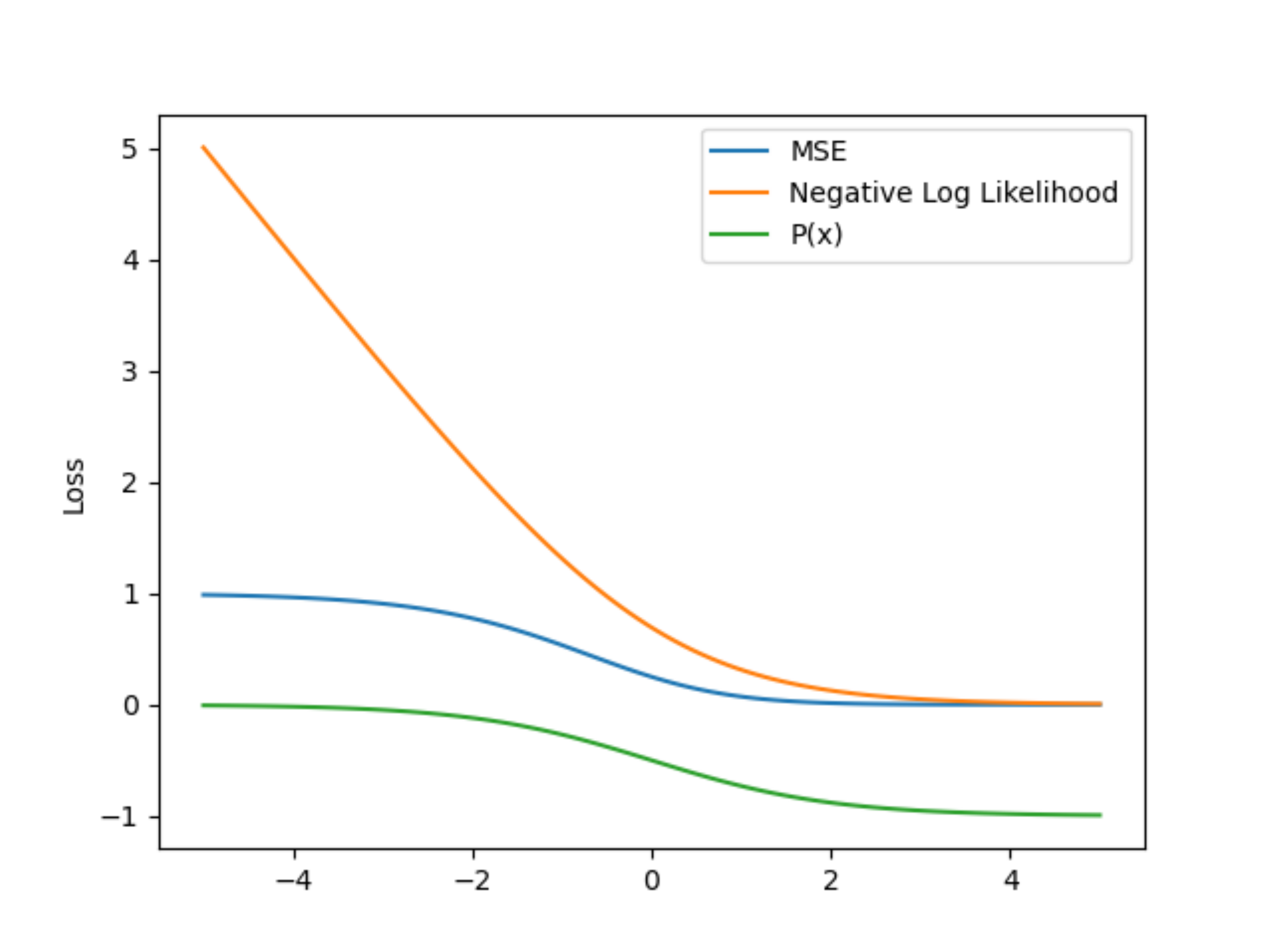


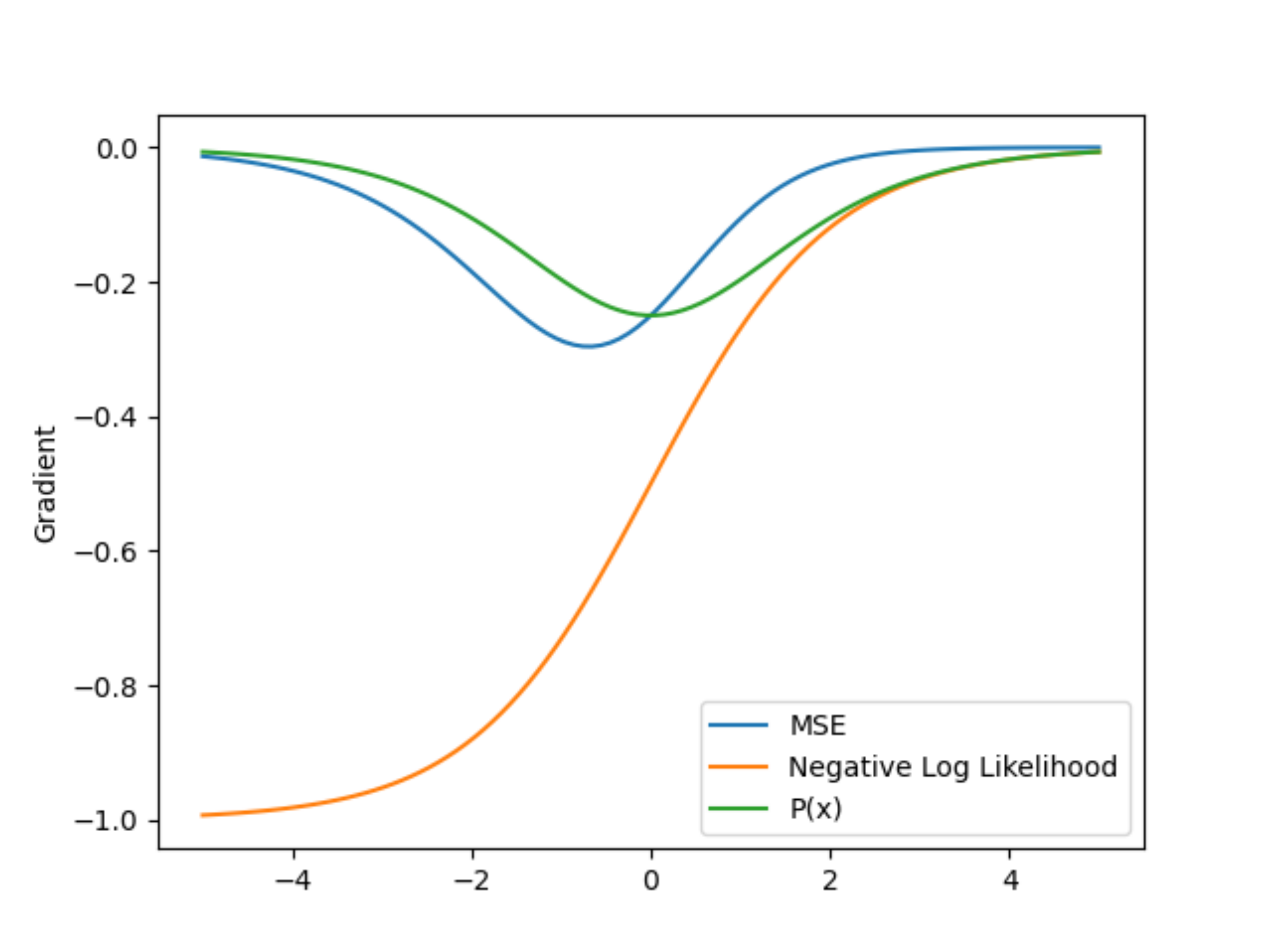
OUTPUT UNITS

Loss and gradient for $y = 1$

We want the gradient to be large when the error is large

Flat gradients make learning slow





ACTIVATION FUNCTIONS

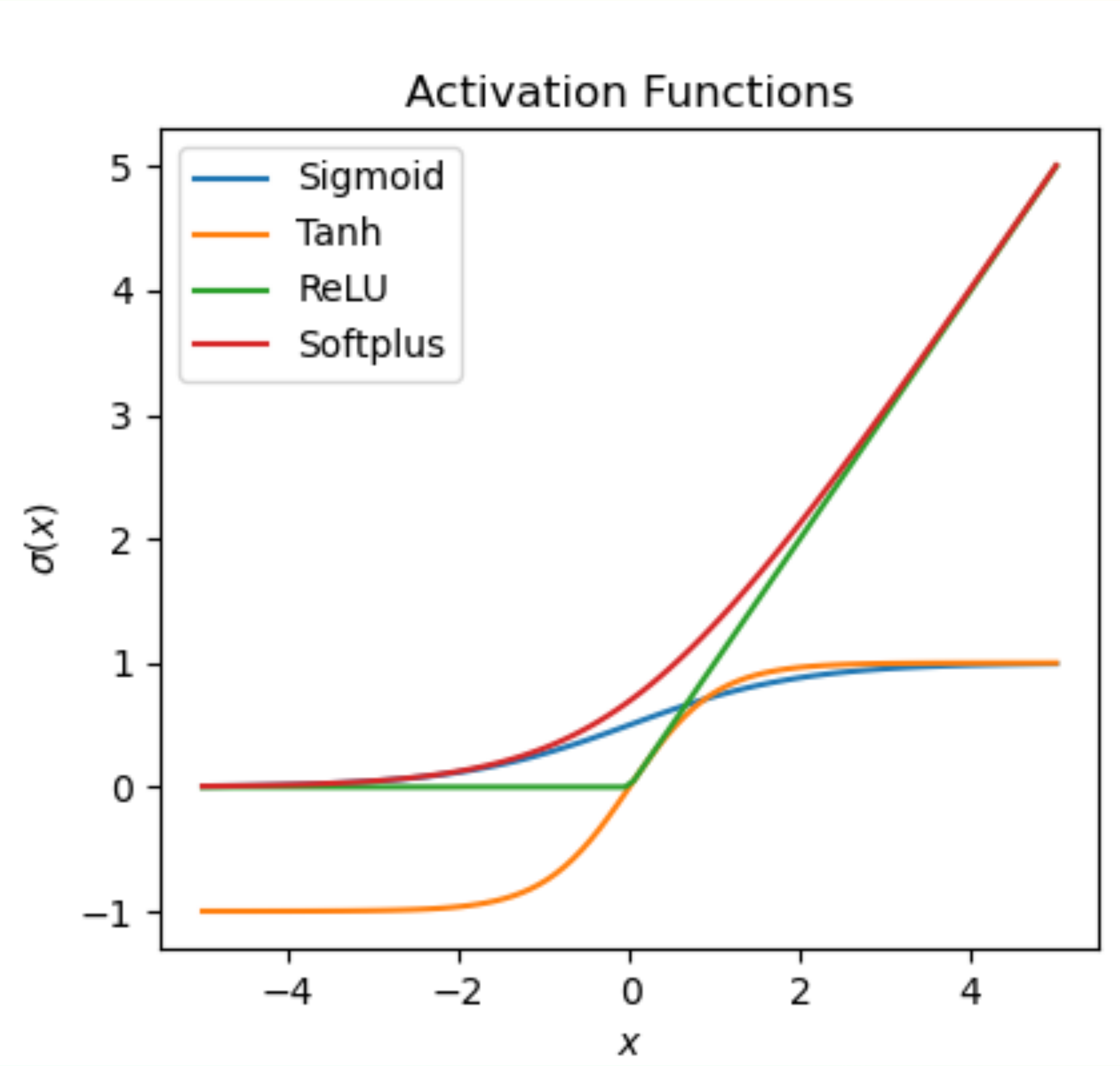
BASIC OPERATION

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

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

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

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



ACTIVATION FUNCTIONS

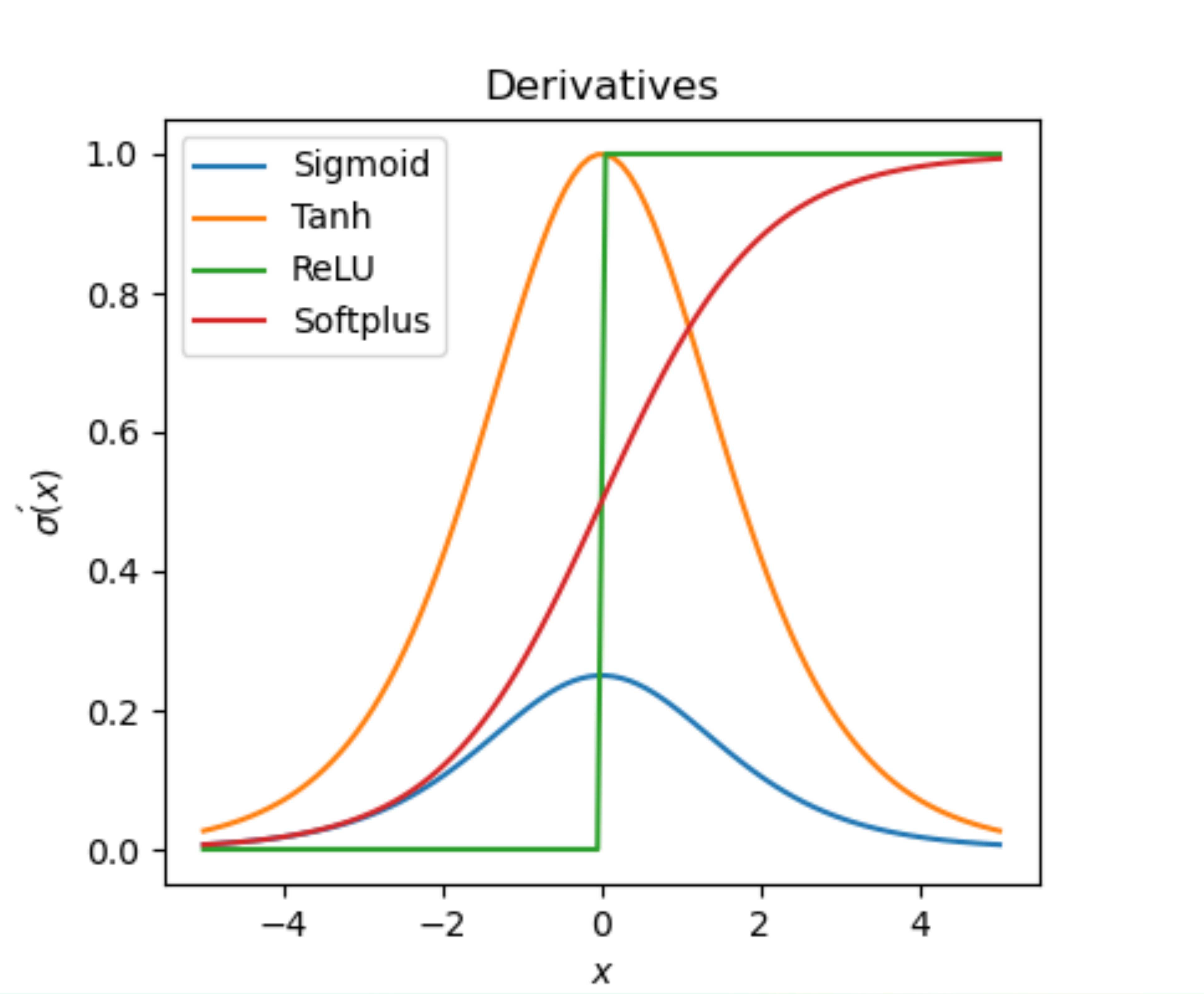
BASIC OPERATION

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

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

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

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



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

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

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

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

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

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

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

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

sometimes for classification (numerical precision reasons, future lecture)

NEXT CLASS

Next Class — More Neural Networks!