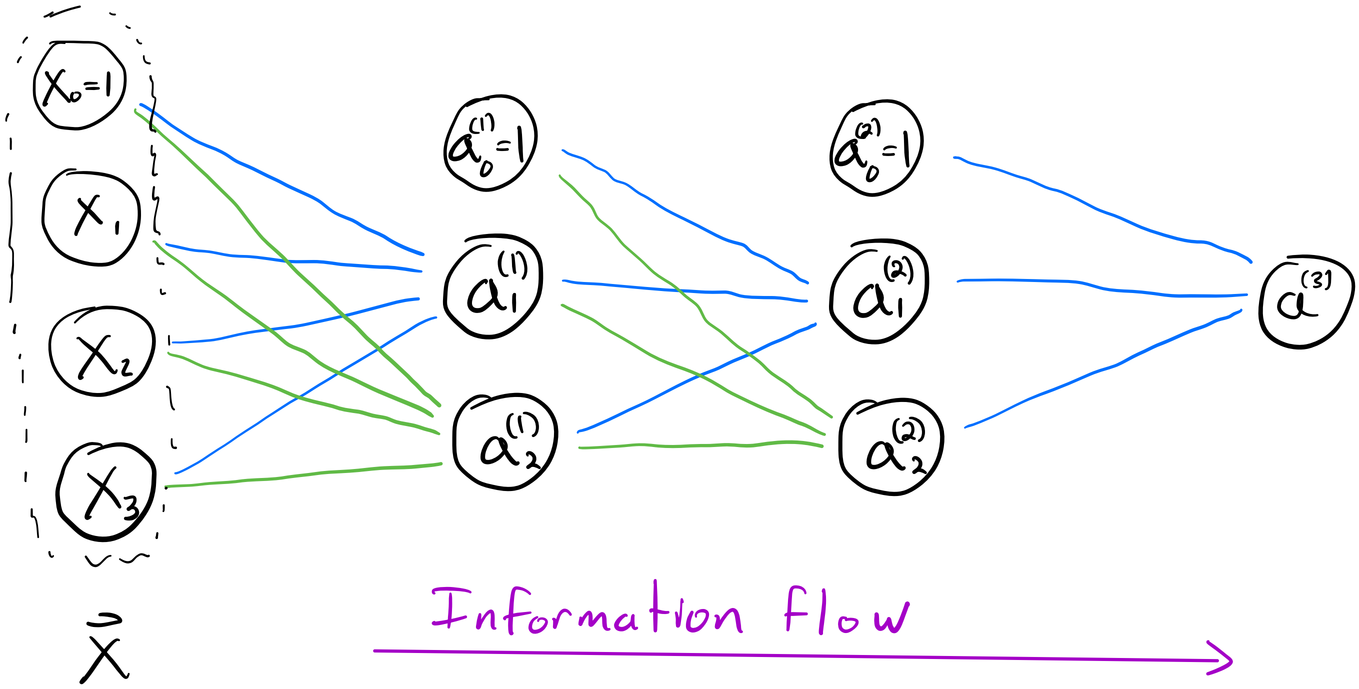
Neural Networks (NN)

A NN is a function $f(\vec{x})$

Ex: $f(\vec{x}) = a^{(3)}$



ERM with NNs (High Level)

$$D = ((\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n))$$

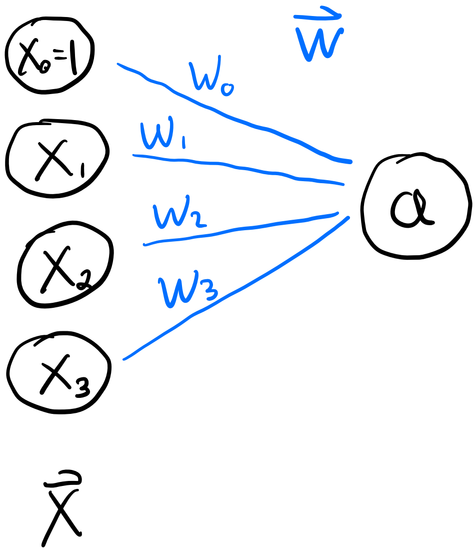
$$\mathcal{A}(D) = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \hat{L}(f)$$

$$\mathcal{F} = \{ f \mid f : \mathcal{X} \to \mathcal{Y} \text{ and } f \text{ is a NN} \}$$

every $f \in \mathcal{F}$ is defined by B weight matrices $W^{(1)}, \dots, W^{(B)}$

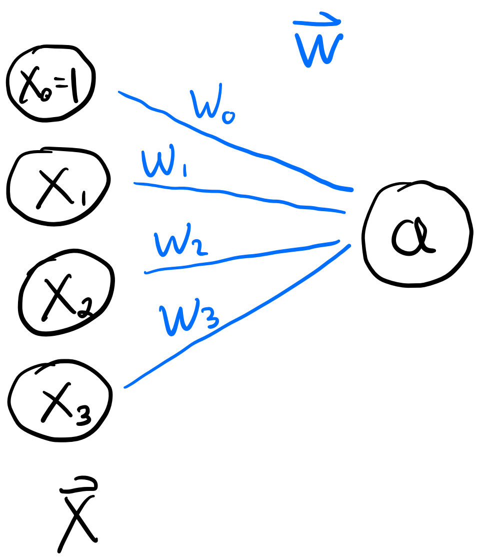
Previously Seen functions as NNs

Linear: $f(\vec{x}) = \vec{x}^T \vec{w}$, $d=3$



$$a = x_0 w_0 + x_1 w_1 + \cdots x_3 w_3$$
$$= \vec{x}^T \vec{w}$$
$$= h(\vec{x}^T \vec{w}) \quad \text{where } h(z) = z$$

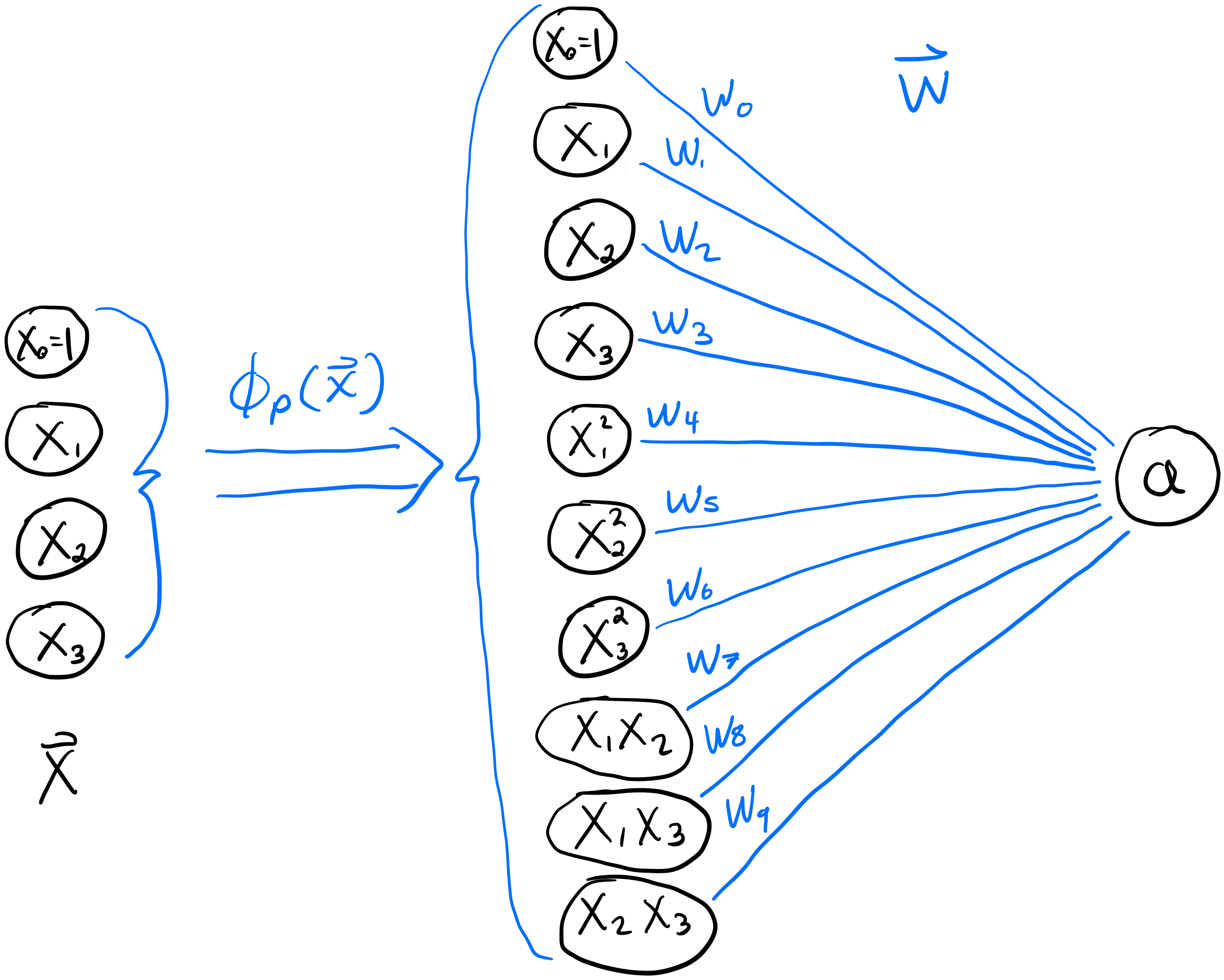
Sigmoid: $f(\vec{x}) = \sigma(\vec{x}^T \vec{w})$



$$a = \sigma\left(x_0 w_0 + x_1 w_1 + \cdots x_3 w_3\right)$$
$$= \sigma\left(\vec{x}^T \vec{w}\right)$$
$$= h(\vec{x}^T \vec{w}) \quad \text{where } h = \sigma$$

Linear (with polynomial features):

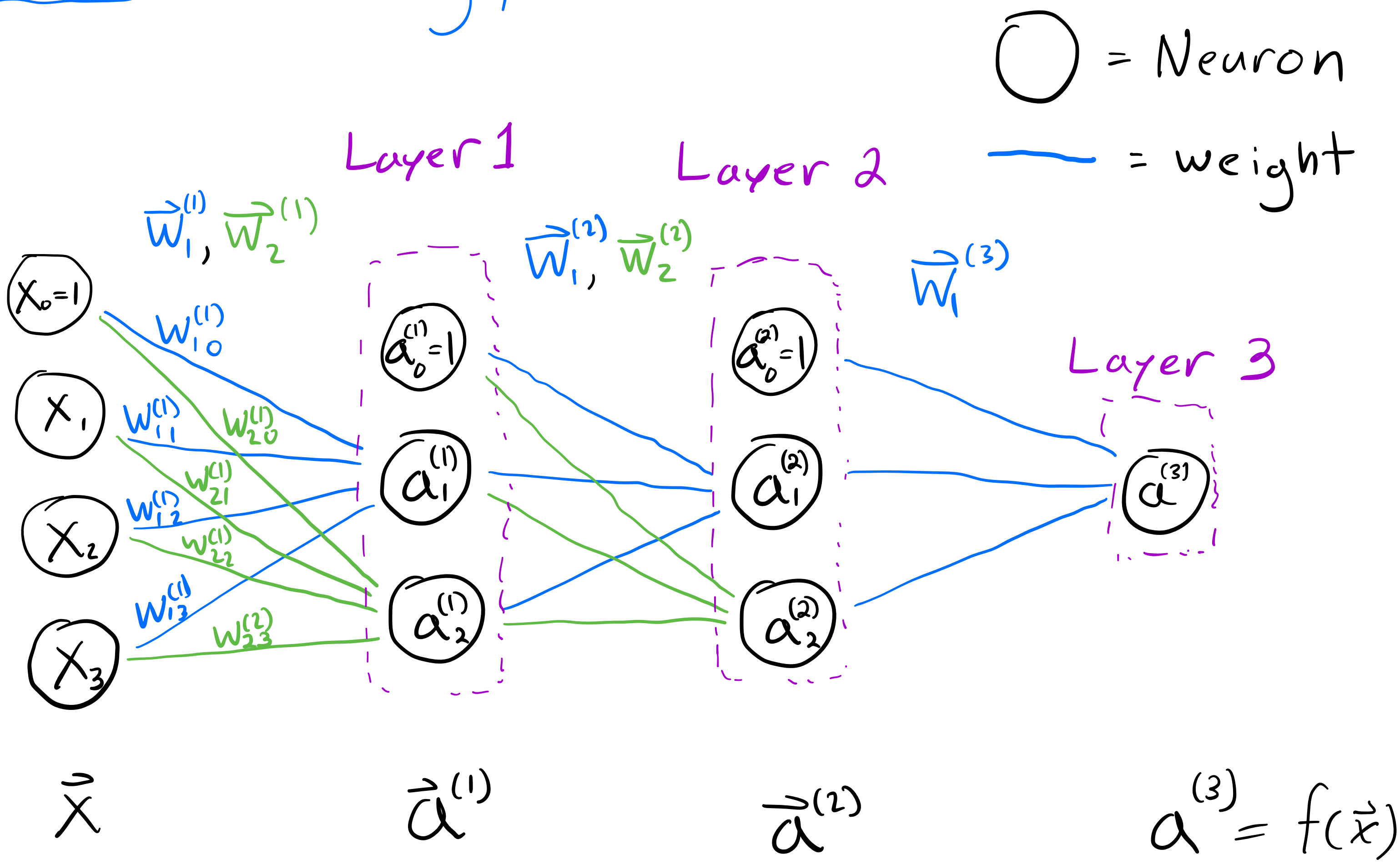
$$f(\vec{x}) = \phi_p(\vec{x})^T \vec{w} = a \quad , d=3, p=2$$



Many of the new features in $\phi_p(\vec{x})$ may not useful

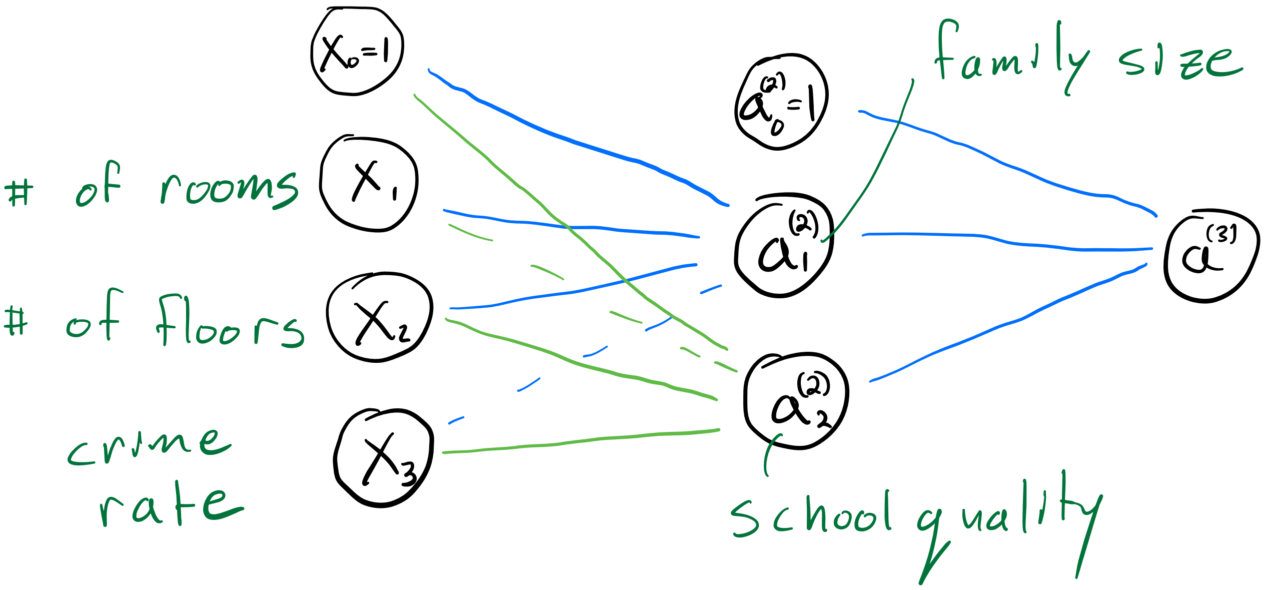
When p is large this is very wasteful

NN Terminology



Ex: Regression (House price prediction)

$\mathcal{Y} = \mathbb{R}, \ \mathcal{X} = \mathbb{R}^{d+1}, \ d = 3$



Ex: Binary Classification (Cat or No Cat)

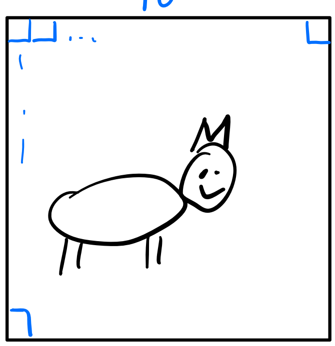
$y = \{0, 1\}$ $\quad$ $\mathcal{X} = \mathbb{R}^{256+1}$

No cat (0) $\quad$ cat (1)

$16 \times 16 = 256 = d$

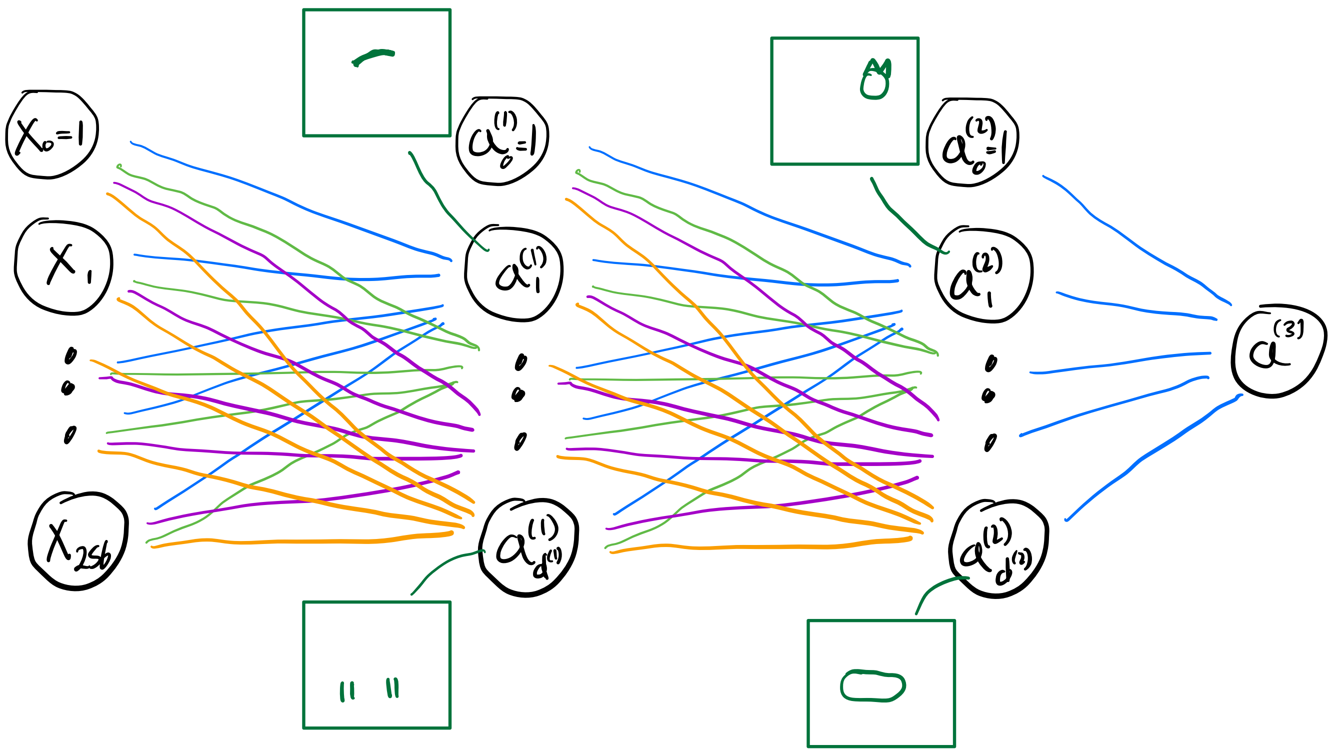
16

16



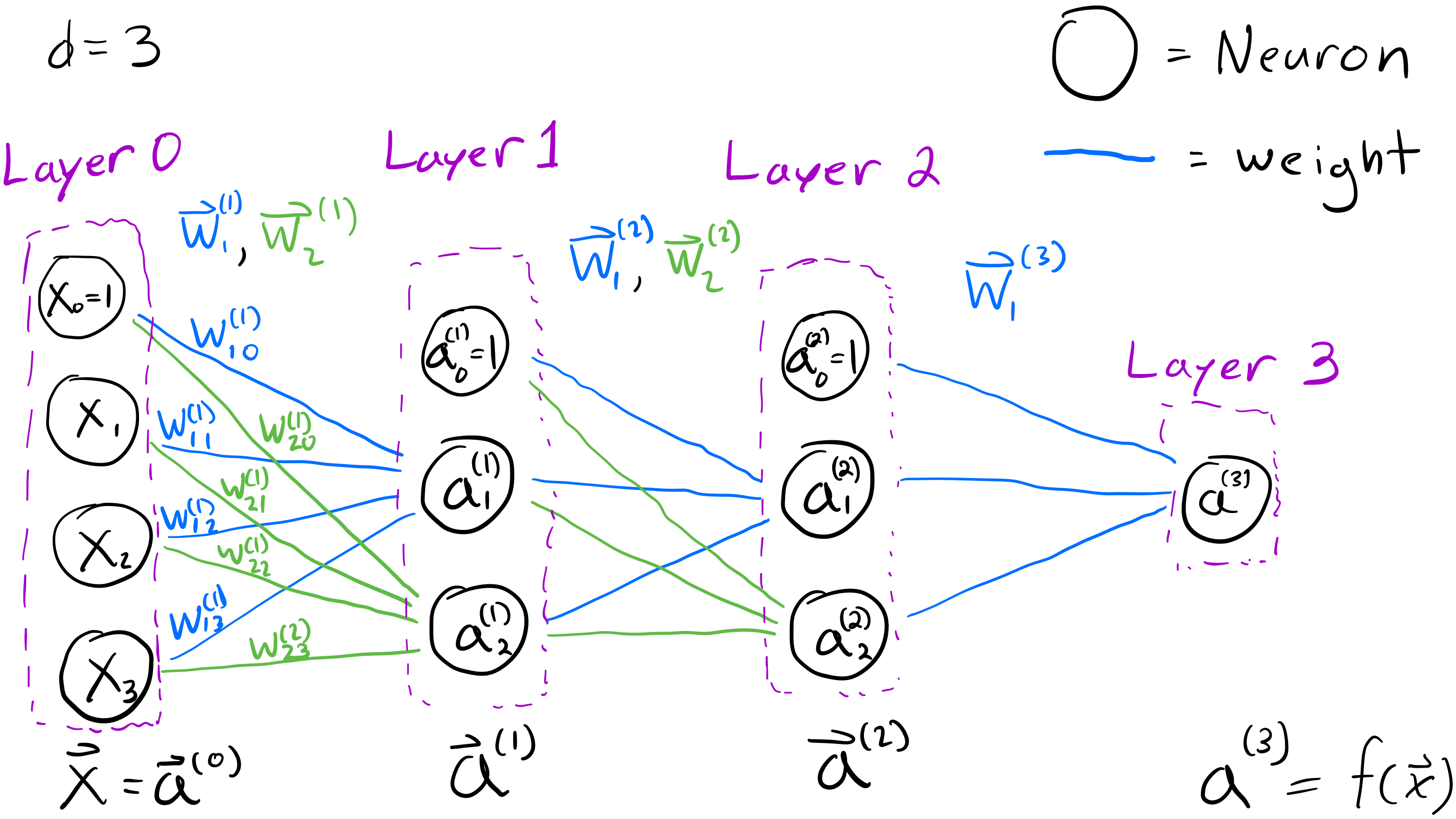
$$\longrightarrow \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{256} \end{pmatrix} = \vec{x}$$

$f(\vec{x}) = a^{(3)}$ probability of a cat



NN Definition (formal)

$d = 3$



activation

$$\vec{a}^{(1)} = (a_0^{(1)} = 1,\ a_1^{(1)},\ a_2^{(1)})$$

where $a_1^{(1)} = h^{(1)}(z_1^{(1)})$, $a_2^{(1)} = h^{(1)}(z_2^{(1)})$

activation function $h^{(1)}: \mathbb{R} \to \mathbb{R}$ ex: sigmoid σ

and $z_1^{(1)} = (\vec{a}^{(0)})^T \vec{w}_1^{(1)}$, $z_2^{(1)} = (\vec{a}^{(0)})^T \vec{w}_2^{(1)}$

pre-activation

$\vec{a}^{(0)} \in \mathbb{R}^{d+1}$, $\vec{w}_1^{(1)}, \vec{w}_2^{(1)} \in \mathbb{R}^{d+1}$, $\vec{a}^{(1)} \in \mathbb{R}^{2+1}$

$$\vec{a}^{(2)} = (a_0^{(2)} = 1,\ a_1^{(2)},\ a_2^{(2)})$$

where $a_1^{(2)} = h^{(2)}(z_1^{(2)})$, $a_2^{(2)} = h^{(2)}(z_2^{(2)})$

and $z_1^{(2)} = (\vec{a}^{(1)})^T \vec{w}_1^{(2)}$, $z_2^{(2)} = (\vec{a}^{(1)})^T \vec{w}_2^{(2)}$

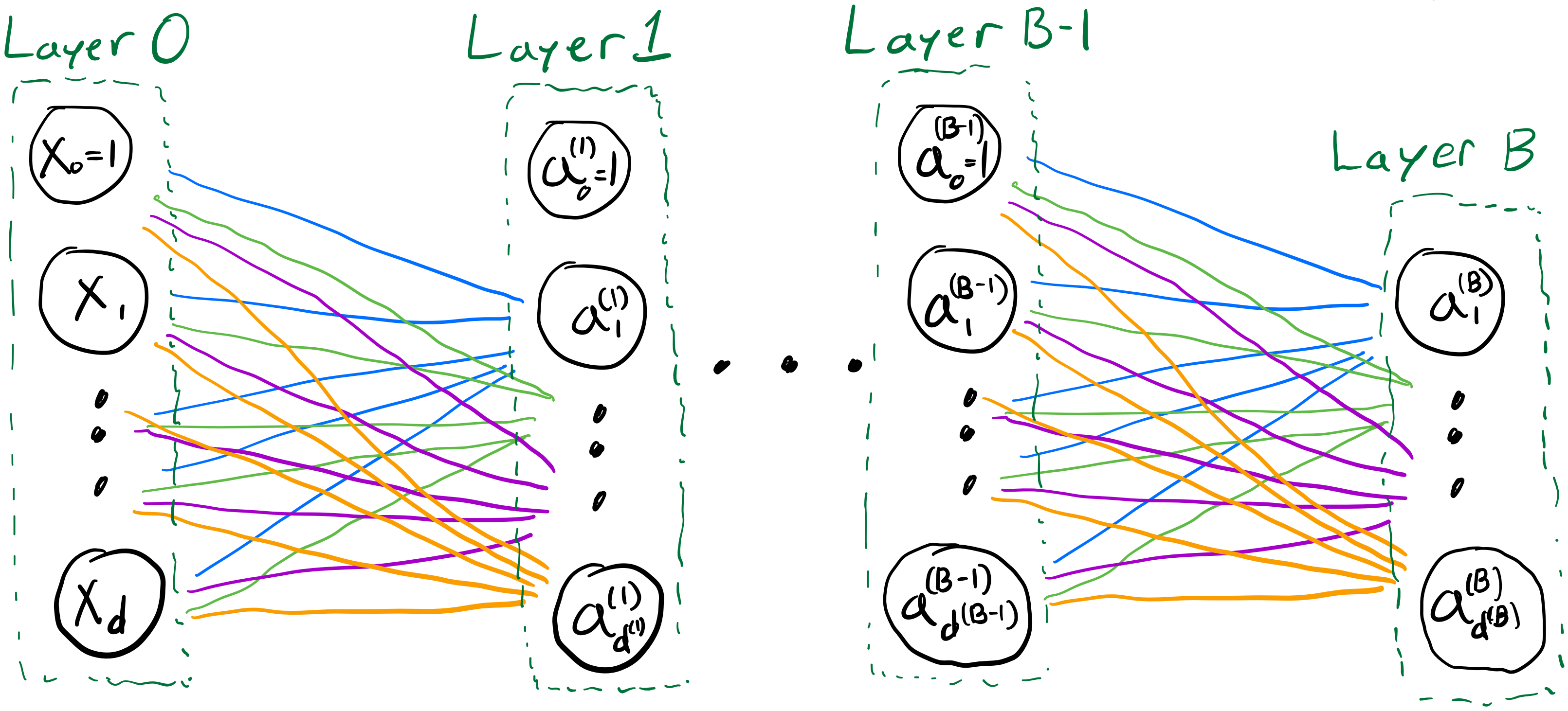
$\vec{w}_1^{(2)}, \vec{w}_2^{(2)} \in \mathbb{R}^{2+1}$, $\vec{a}^{(2)} \in \mathbb{R}^{2+1}$

$a^{(3)} = h^{(3)}(z^{(3)})$ where $z^{(3)} = (\vec{a}^{(2)})^T \vec{w}^{(3)}$

$\vec{w}_j^{(3)} \in \mathbb{R}^{2+1}$, $a^{(3)} \in \mathbb{R}$

In general:

No Bias in Layer B



For layer $b \in \{1, \ldots, B\}$

Weights

$$\vec{w}_1^{(b)}, \ldots, \vec{w}_{d^{(b)}}^{(b)} \in \mathbb{R}^{d^{(b-1)}+1}$$

Pre-activations

$$\vec{z}^{(b)} = (z_1^{(b)}, \ldots, z_{d^{(b)}}^{(b)}) \in \mathbb{R}^{(b)}$$

where $z_j^{(b)} = (\vec{a}^{(b-1)})^T \vec{w}_j$ for $j \in \{1, \ldots, d^{(b)}\}$

Activations

$$\vec{a}^{(0)} = \vec{x} \in \mathbb{R}^{d+1} = \mathbb{R}^{d^{(0)}+1}, \quad d = d^{(0)}$$

$$\vec{a}^{(b)} = (a_0^{(b)} = 1, a_1^{(b)}, \ldots, a_{d^{(b)}}^{(b)}) \in \mathbb{R}^{d^{(b)}+1}$$ except $b = B$

$$\vec{a}^{(B)} = \left(a_1^{(B)}, \ldots, a_{d^{(B)}}^{(B)}\right) \in \mathbb{R}^{d^{(B)}}$$

where $a_j^{(b)} = h\left(z_j^{(b)}\right)$ for $j \in \{1, \ldots, d^{(b)}\}$

Activation function

$h^{(b)}: \mathbb{R} \to \mathbb{R}$

$$\boxed{\text{NN}: \ f(\vec{x}) = \vec{a}^{(B)}}$$

Defining the number of layers B and the number of neurons in each layer: $d^{(1)}, \ldots, d^{(B)}$, and the activation functions $h^{(1)}, \ldots, h^{(B)}$ defines a "NN architecture"

Matrix Notation

Represent $w_1^{(b)}, \dots, w_{d^{(b)}}^{(b)}$ as a matrix:

$$W^{(b)} = \begin{bmatrix} | & | & & | \\ w_1^{(b)} & w_2^{(b)} & \cdots & w_{d^{(b)}}^{(b)} \\ | & | & & | \end{bmatrix}$$

dimension: $\left(d^{(b-1)}+1\right)$ by $d^{(b)}$

applied element-wise

$$\vec{a}^{(b)} = h^{(b)}\left((\vec{a}^{(b-1)})^T W^{(b)}\right)$$

$$h^{(b)}(\vec{v}) = h^{(b)}(v_1, \dots, v_d)$$
$$= \left(h^{(b)}(v_1), \dots, h^{(b)}(v_d)\right)$$

$$f(\vec{x}) = \vec{a}^{(B)} = h^{(B)}\left(h^{(B-1)}\left(\dots h^{(2)}\left(h^{(1)}\left((\vec{x})^T W^{(1)}\right) W^{(2)}\right) \dots W^{(B-1)}\right) W^{(B)}\right)^T$$

ERM with Neural Networks

$$D = ((\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n))$$

$$\mathcal{A}(D) = \hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \hat{L}(f) \text{ where } \hat{L}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(\vec{x}_i), y_i)$$

$\mathcal{F} = \{ f \mid f : \mathcal{X} \to \mathcal{Y}$ and f is a NN with a specific architecture$\}$

every $f_{W^{(1)}, \dots, W^{(B)}} \in \mathcal{F}$ is defined by B weight matrices $W^{(1)}, \dots, W^{(B)}$

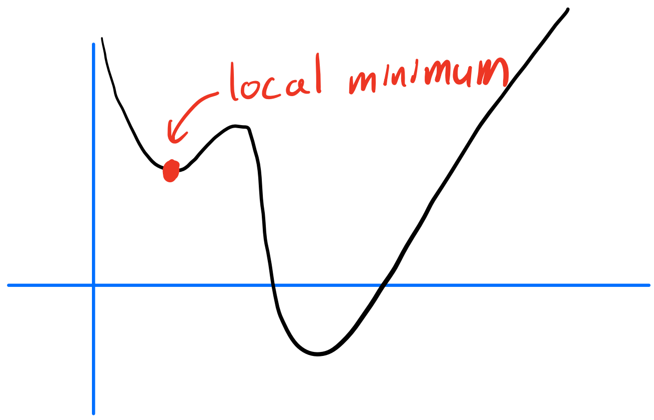
$\mathcal{A}(D) = \hat{f}$ where $\hat{f}(\vec{x}) = \vec{a}^{(B)}$ is defined by

$$\hat{W}^{(1)}, \dots, \hat{W}^{(B)} = \underset{W^{(1)}, \dots, W^{(B)}}{\operatorname{argmin}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} \ell\left(f_{W^{(1)}, \dots, W^{(B)}}(\vec{x}_i), y_i\right)}_{= \hat{L}(W^{(1)}, \dots, W^{(B)})}$$

Usually no closed form solution

$$\left(\vec{w}_j^{(b)}\right)^{(t+1)} = \left(\vec{w}_j^{(b)}\right)^{(t)} - \eta^{(t)} \nabla_{\vec{w}_j^{(b)}} \hat{L}\left(\left(W^{(1)}\right)^{(t)}, \dots, \left(W^{(B)}\right)^{(t)}\right)$$

$\hat{L}(W^{(1)}, \ldots, W^{(B)})$ is usually not convex



Softmax

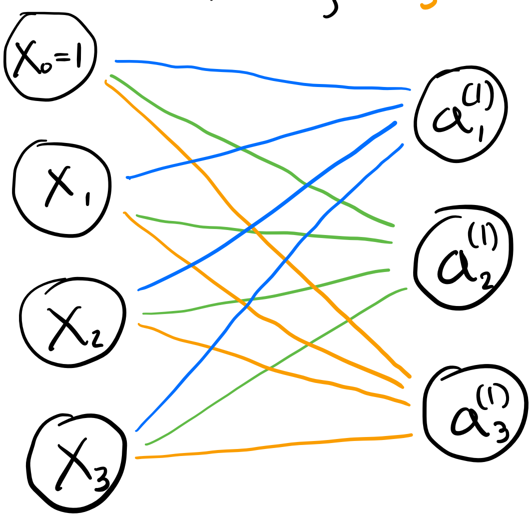
$d=3, k=3,$

$$\sigma(\vec{x}^T\vec{w}_1^{(1)}, \vec{x}^T\vec{w}_2^{(1)}, \vec{x}\vec{w}_3^{(1)})$$

$$= \left(\sigma_1(\vec{x}^T\vec{w}_1^{(1)}, \vec{x}^T\vec{w}_2^{(1)}, \vec{x}\vec{w}_3^{(1)}), \sigma_2(\vec{x}^T\vec{w}_1^{(1)}, \vec{x}^T\vec{w}_2^{(1)}, \vec{x}\vec{w}_3^{(1)}), \sigma_3(\vec{x}^T\vec{w}_1^{(1)}, \vec{x}^T\vec{w}_2^{(1)}, \vec{x}\vec{w}_3^{(1)})\right)^T$$

Softmax cannot be implemented by a single layer NN

$\vec{w}_1^{(1)}, \vec{w}_2^{(1)}, w_3^{(1)}$



$$a_1^{(1)} = h^{(1)}(\vec{x}^T\vec{w}_1^{(1)})$$

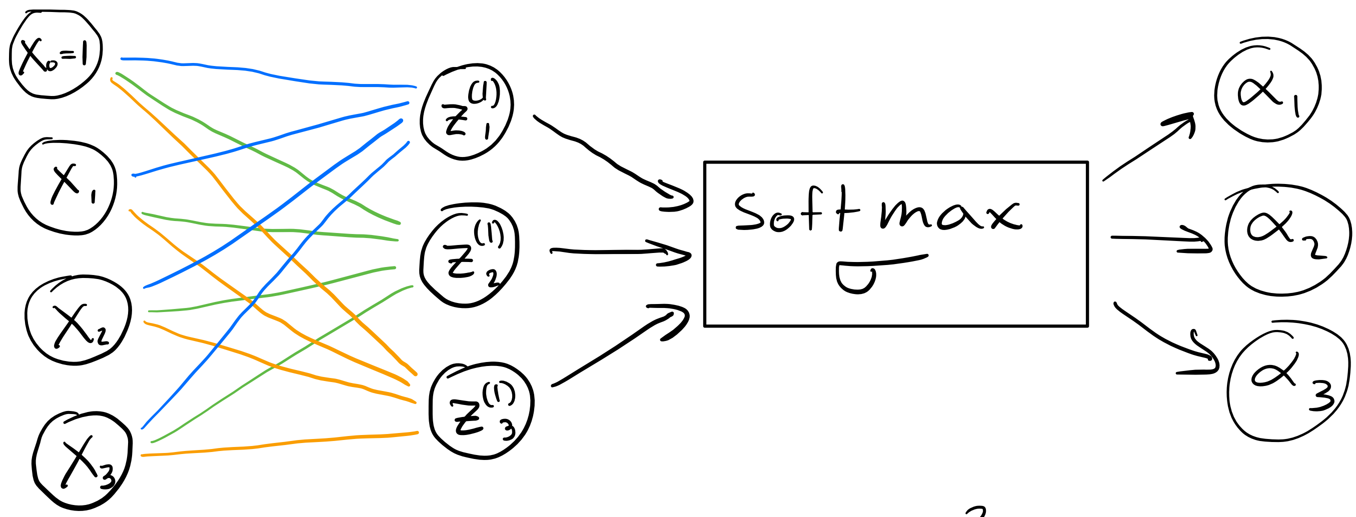
$$a_2^{(1)} = h^{(1)}(\vec{x}^T\vec{w}_2^{(1)})$$

$$a_3^{(1)} = h^{(1)}(\vec{x}^T\vec{w}_3^{(1)})$$

if $\quad h^{(1)}(z) = z$

$$f(\vec{x}) = \vec{a}^{(1)} = (\vec{x}^T\vec{w}_1^{(1)}, \vec{x}^T\vec{w}_2^{(1)}, \vec{x}\vec{w}_3^{(1)})^T$$

$\sigma(f(\vec{x}))$ post-processing step



$$\sum_{q=1}^{3} \alpha_q = 1 \,, \quad \alpha_q \in [0,1]$$

$$\sigma\left(z_1^{(1)}, z_2^{(1)}, z_3^{(1)}\right)$$

$$= \left(\frac{\exp\left(z_1^{(1)}\right)}{\sum_{q=1}^{3} \exp\left(z_q^{(1)}\right)} \,, \frac{\exp\left(z_2^{(1)}\right)}{\sum_{q=1}^{3} \exp\left(z_q^{(1)}\right)} \,, \frac{\exp\left(z_3^{(1)}\right)}{\sum_{q=1}^{3} \exp\left(z_q^{(1)}\right)} \right)^T$$