Machine Learning

CS 539

Worcester Polytechnic Institute
Department of Computer Science
Instructor: Prof. Kyumin Lee

HW3

- https://canvas.wpi.edu/courses/58900/assignments/356655
- Due date is July 2.

$w = \theta_{1 \dots d}$

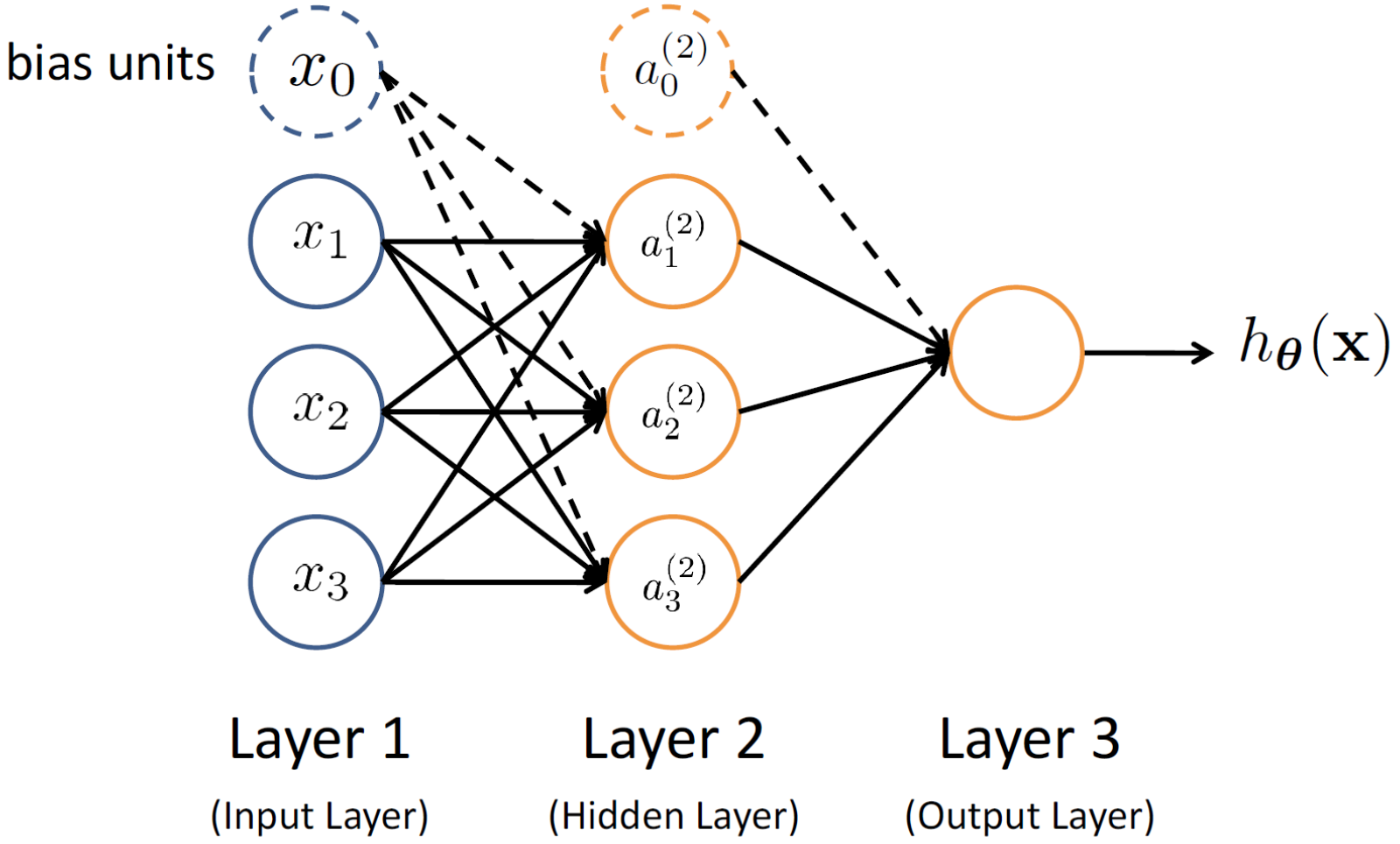
$b = \theta_0$

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial w}$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial b}$$

Neural Networks

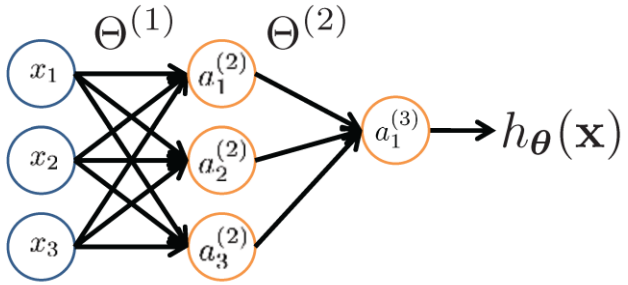
Neural Network



Feed-Forward Process

- Input layer units are set by some exterior function (think of these as **sensors**), which causes their output links to be **activated** at the specified level
- Working forward through the network, the **input function** of each unit is applied to compute the input value
  - Usually this is just the weighted sum of the activation on the links feeding into this node
- The **activation function** transforms this input function into a final value
  - Typically this is a **nonlinear** function, often a **sigmoid** function corresponding to the "threshold" of that node

Neural Network



$a_i^{(j)} =$ "activation" of unit i in layer j

$\Theta^{(j)} =$ weight matrix controlling function mapping from layer j to layer j + 1

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

If network has s_j units in layer j *and* s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ has dimension $s_{j+1} \times (s_j+1)$

$$\Theta^{(1)} \in \mathbb{R}^{3\times 4} \qquad \Theta^{(2)} \in \mathbb{R}^{1\times 4}$$

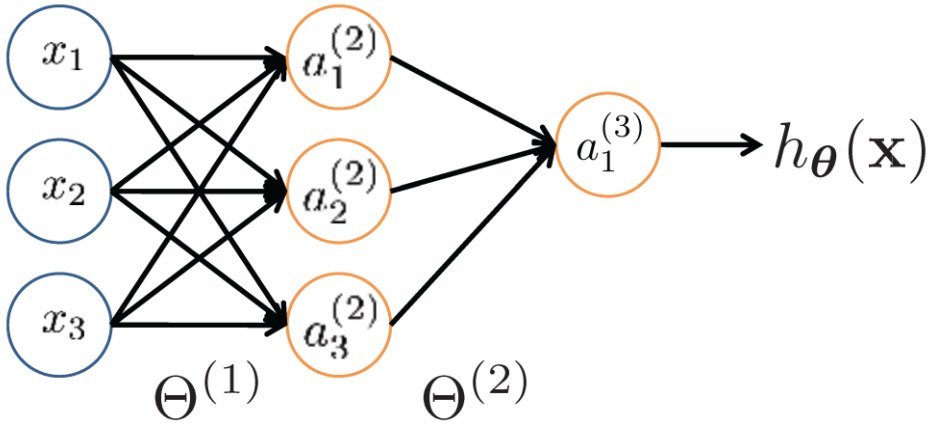
Vectorization

$$a_1^{(2)} = g\left(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3\right) = g\left(z_1^{(2)}\right)$$

$$a_2^{(2)} = g\left(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3\right) = g\left(z_2^{(2)}\right)$$

$$a_3^{(2)} = g\left(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3\right) = g\left(z_3^{(2)}\right)$$

$$h_\Theta(\mathbf{x}) = g\left(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)}\right) = g\left(z_1^{(3)}\right)$$



Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)}\mathbf{x}$$

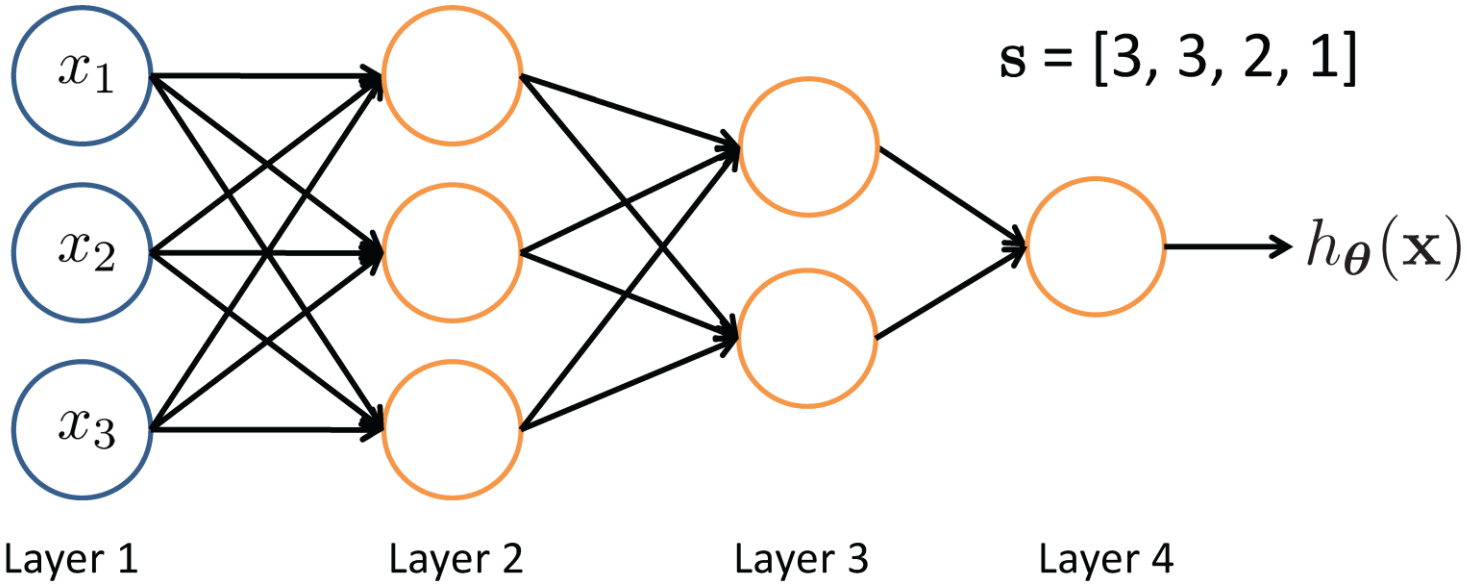
$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

$$\text{Add } a_0^{(2)} = 1$$

$$\mathbf{z}^{(3)} = \Theta^{(2)}\mathbf{a}^{(2)}$$

$$h_\Theta(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

Other Network Architectures



L denotes the number of layers

$\mathbf{s} \in \mathbb{N}^{+L}$ contains the numbers of nodes at each layer

- Not counting bias units
- Typically, $s_0 = d$ (# input features) and $s_{L-1} = K$ (# classes)

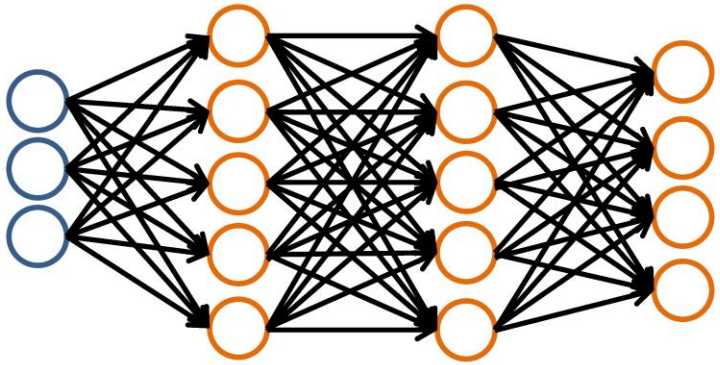
Multiple Output Units: One-vs-Rest

Pedestrian

Car

Motorcycle

Truck



$$h_\Theta(\mathbf{x}) \in \mathbb{R}^K$$

We want:

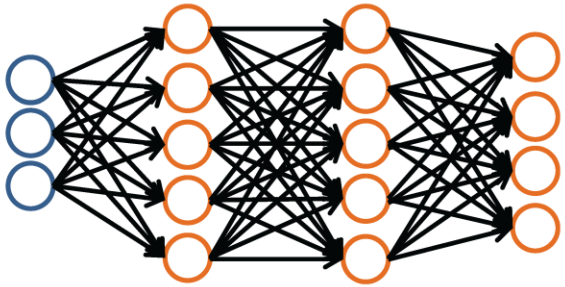
$$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$ when pedestrian

$$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$ when car

$$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$ when motorcycle

$$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$ when truck

Multiple Output Units: One-vs-Rest



$h_\Theta(\mathbf{x}) \in \mathbb{R}^K$

We want:

$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ when pedestrian

$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ when car

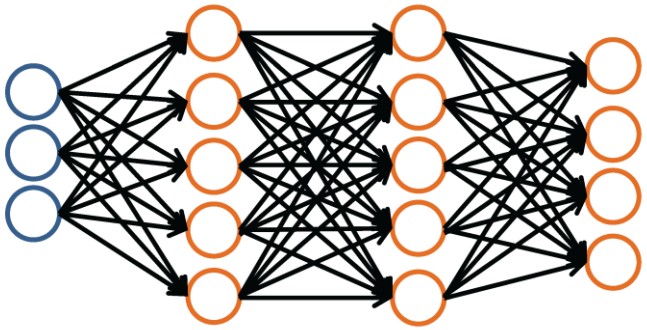
$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ when motorcycle

$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ when truck

---

- Given $\{(\mathbf{x}_1, y_1),\ (\mathbf{x}_2, y_2),\ \ldots,\ (\mathbf{x}_n, y_n)\}$
- Must convert labels to 1-of-K representation
  - e.g., $\mathbf{y}_i = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ when motorcycle, $\mathbf{y}_i = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ when car, etc.

Neural Network Classification



Given:

$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$

$\mathbf{s} \in \mathbb{N}^{+L}$ contains # nodes at each layer

- $s_0 = d$ (# features)

Binary classification

y = 0 or 1

1 output unit $(s_{L-1} = 1)$

Multi-class classification (K classes)

$\mathbf{y} \in \mathbb{R}^K$ e.g. $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

pedestrian car motorcycle truck

K output units $(s_{L-1} = K)$

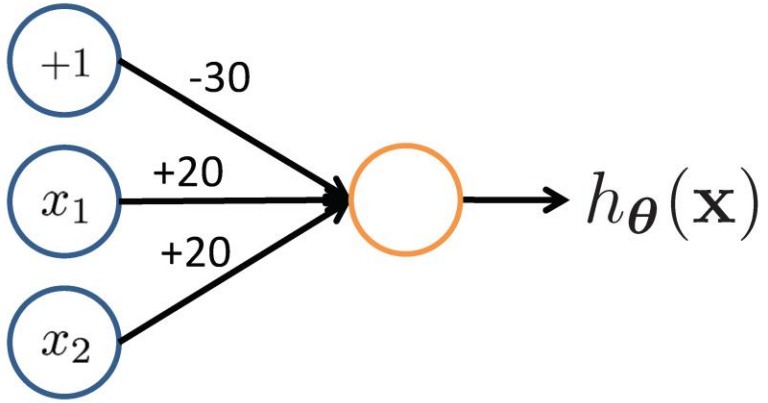
Understanding Representations

Representing Boolean Functions

Simple example: AND

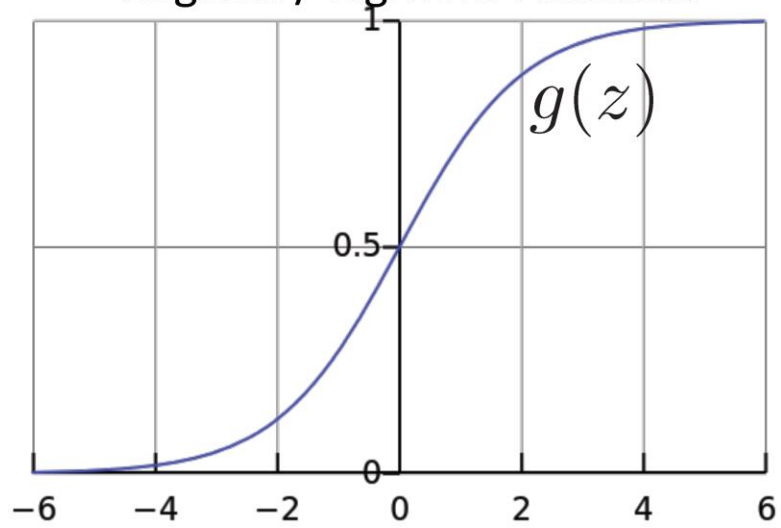
$x_1, x_2 \in \{0, 1\}$

$y = x_1 \text{ AND } x_2$



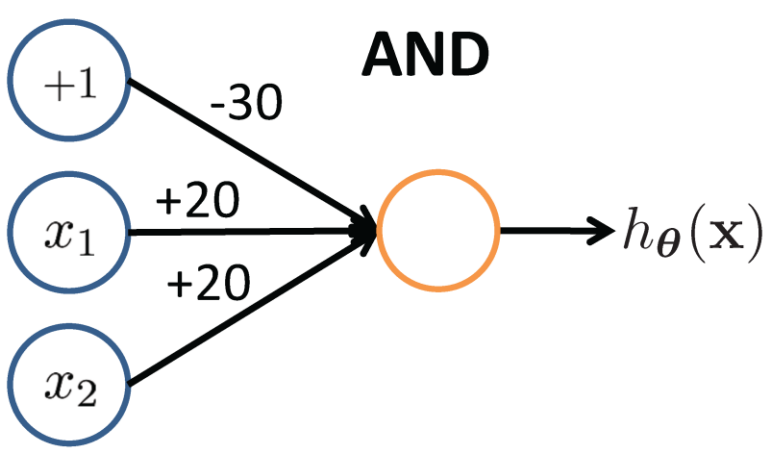
$$\mathrm{h}_\Theta(\mathbf{x}) = g(\text{-}30 + 20x_1 + 20x_2)$$

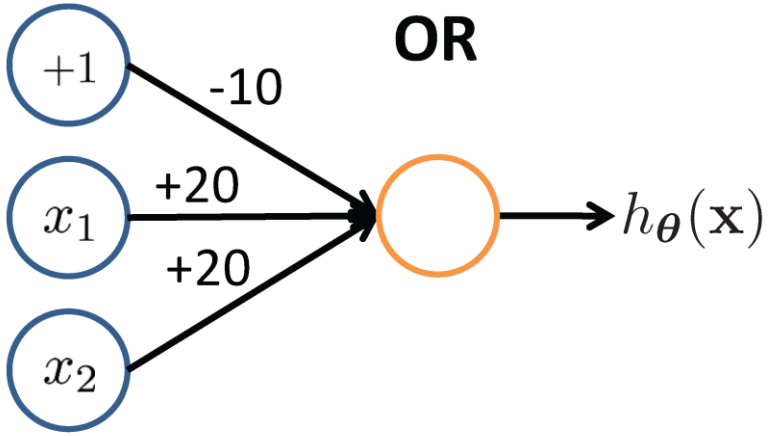
Logistic / Sigmoid Function

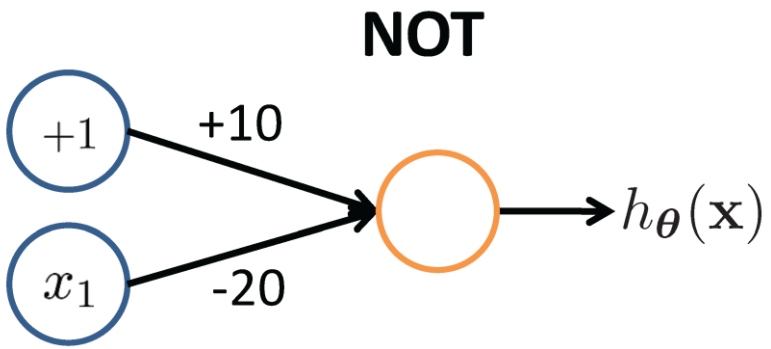


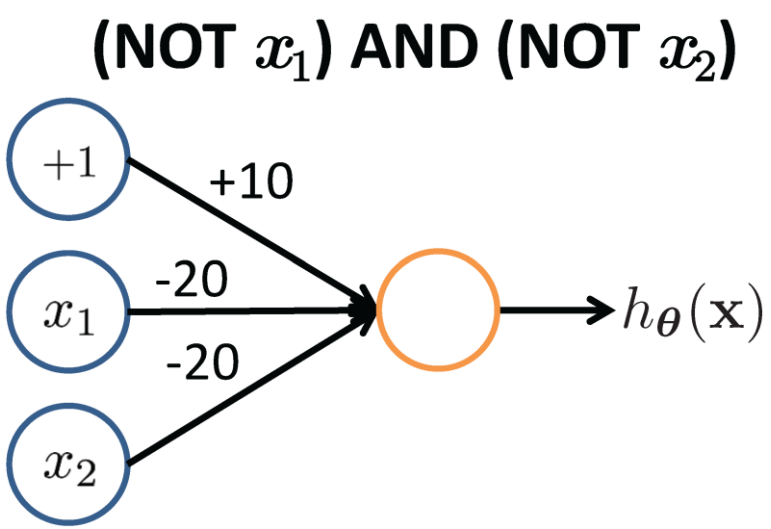
x_1	x_2	$\mathrm{h}_\Theta(\mathbf{x})$
0	0	$g(\text{-}30) \approx 0$
0	1	$g(\text{-}10) \approx 0$
1	0	$g(\text{-}10) \approx 0$
1	1	$g(10) \approx 1$

Representing Boolean Functions

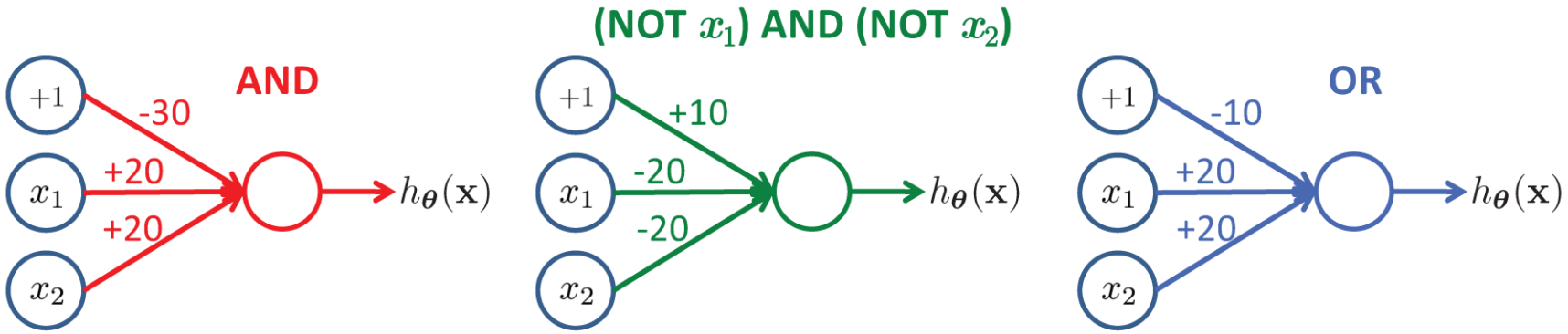


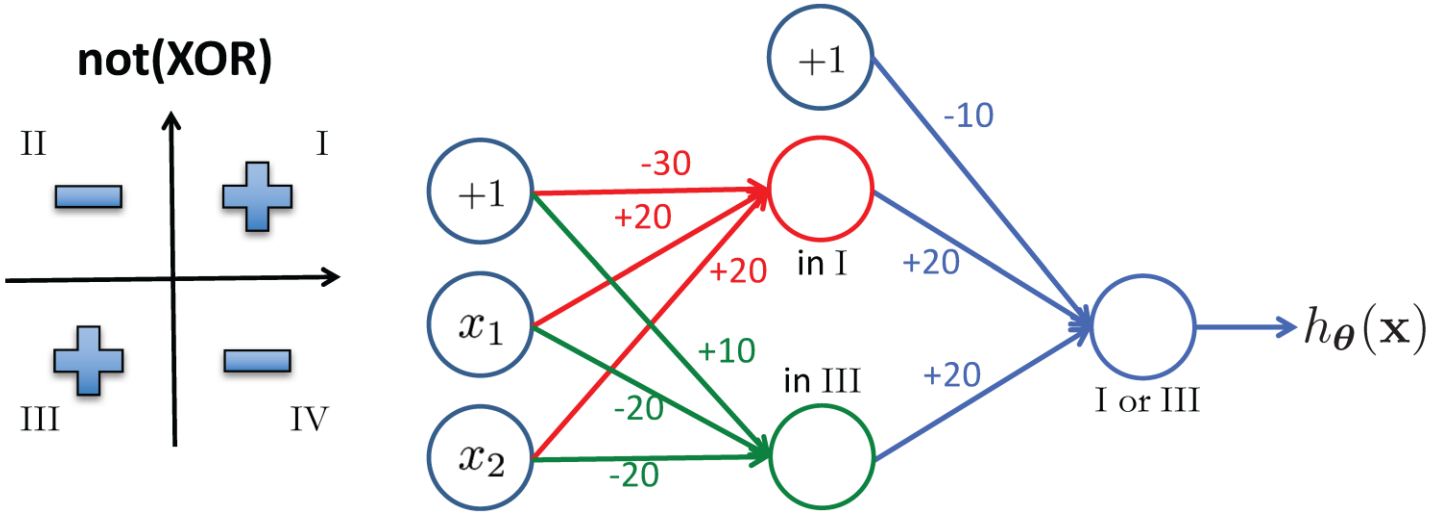




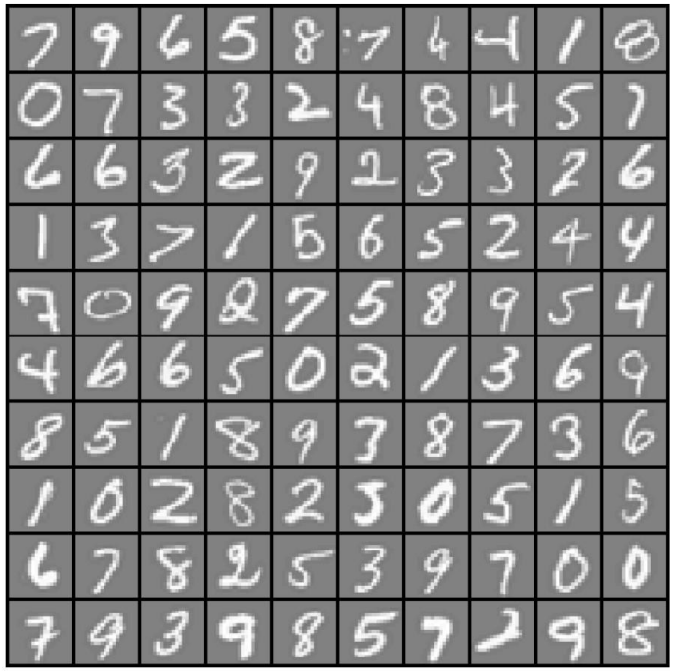


Combining Representations to Create Non-Linear Functions

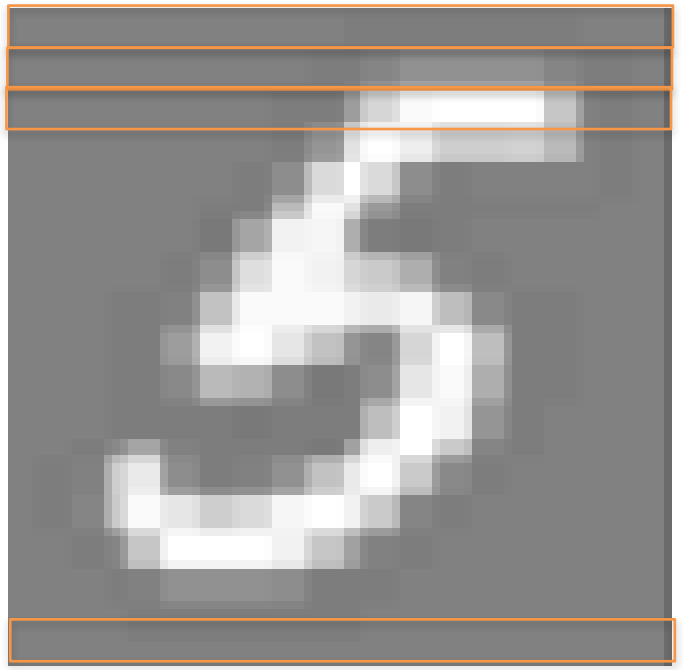




Layering Representations

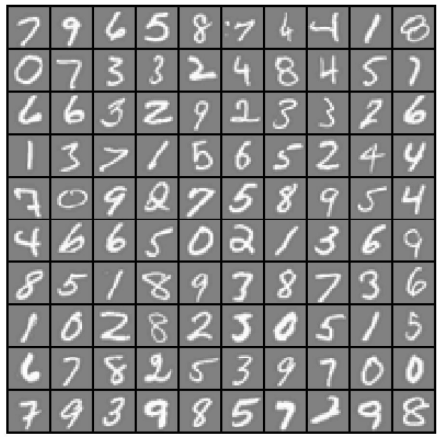


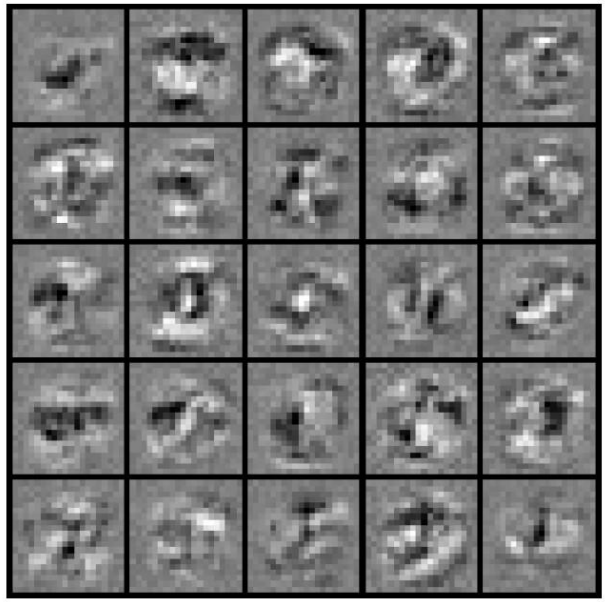
20 × 20 pixel images
$d = 400$ 10 classes

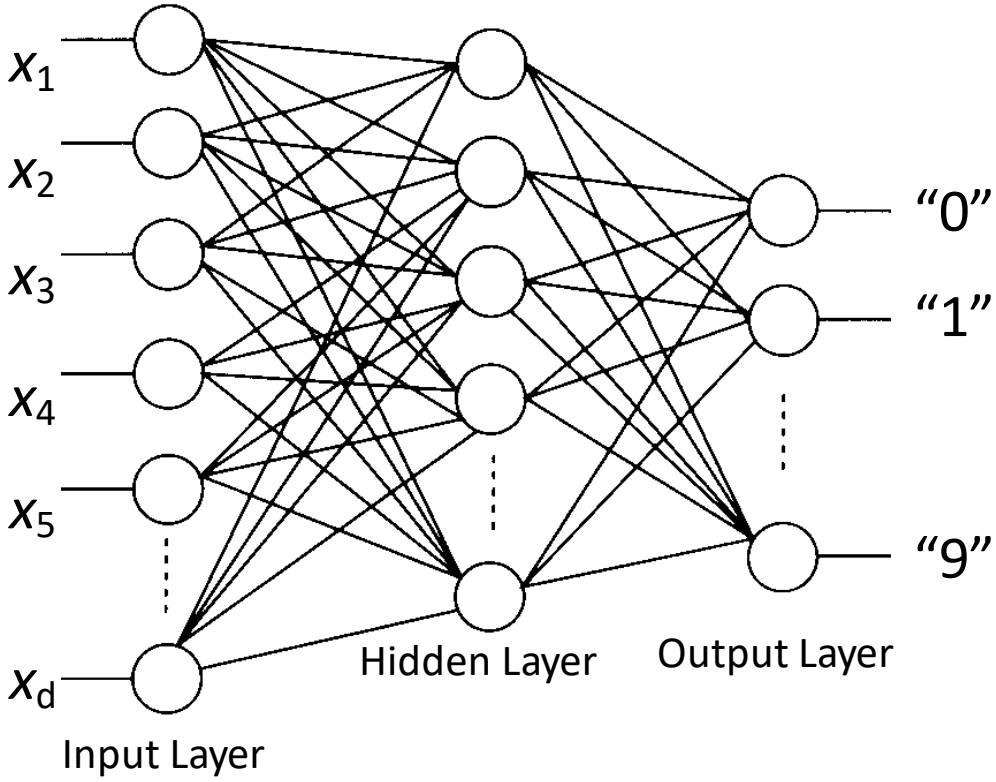


Each image is “unrolled” into a vector x of pixel intensities

Layering Representations



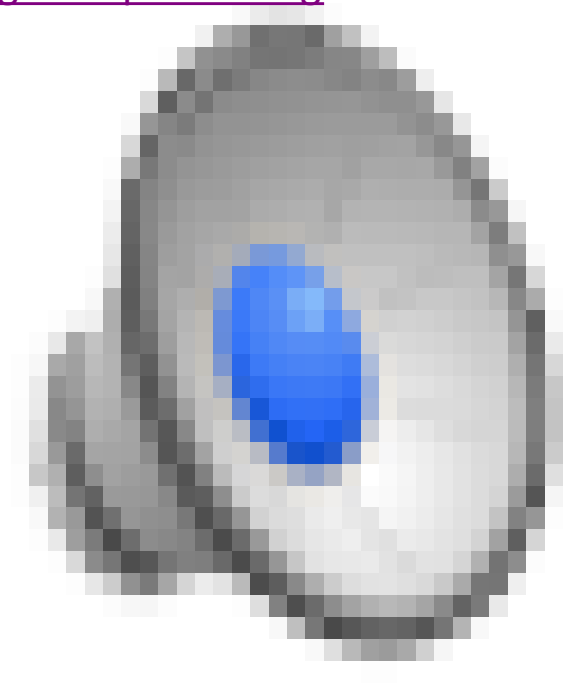




Visualization of
Hidden Layer

LeNet 5 Demonstration:
https://www.youtube.com/watch?v=FwFduRA_L6Q&ab_channel=YannLeCun

LeNet layers:
https://en.wikipedia.org/wiki/LeNet#:~:text=In%20general%2C%20LeNet%20refers%20to,in%20large%2Dscale%20image%20processing.



Neural Network Learning

Batch Perceptron

Given training data $\left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$
Let $\boldsymbol{\theta} \leftarrow [0, 0, \ldots, 0]$
Repeat:
    Let $\boldsymbol{\Delta} \leftarrow [0, 0, \ldots, 0]$
    for $i = 1 \ldots n$, do
        if $y^{(i)}\boldsymbol{x}^{(i)}\boldsymbol{\theta} \leq 0$ // prediction for ith instance is incorrect
            $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta} + y^{(i)}\boldsymbol{x}^{(i)}$
    $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta}/n$ // compute average update
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\boldsymbol{\Delta}$
Until $\|\boldsymbol{\Delta}\|_2 < \epsilon$

- Simplest case: α = 1 and don't normalize, yields the fixed increment perceptron
- Each increment of outer loop is called an **epoch**

Learning in NN: Backpropagation

- Similar to the perceptron learning algorithm, we cycle through our examples
  - If the output of the network is correct, no changes are made
  - If there is an error, weights are adjusted to reduce the error

- The trick is to assess the blame for the error and divide it among the contributing weights

Cost Function

Logistic Regression:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log h_{\boldsymbol{\theta}}(\mathbf{x}_i) + (1-y_i)\log(1-h_{\boldsymbol{\theta}}(\mathbf{x}_i))] + \frac{\lambda}{2n}\sum_{j=1}^{d}\theta_j^2$$

Neural Network: Binary classification (1 or 0)

$$J(\Theta) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log h_{\Theta}(\mathbf{x}_i) + (1-y_i)\log\left(1-h_{\Theta}(\mathbf{x}_i)\right)]$$
$$+\frac{\lambda}{2n}\sum_{l=1}^{L-1}\sum_{i=1}^{s_{l-1}}\sum_{j=1}^{s_l}\left(\Theta_{ji}^{(l)}\right)^2$$

General NN Cost Function

$$J(\Theta) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\big(h_\Theta(\mathbf{x}_i), y_i\big) \quad + \frac{\lambda}{2n}\sum_{l=1}^{L-1}\sum_{i=1}^{s_{l-1}}\sum_{j=1}^{s_l}\left(\Theta_{ji}^{(l)}\right)^2$$

Given $h_\Theta(\mathbf{x}_i) = \hat{y}$

Loss of Binary classification (1 or 0) ← log loss

$$\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y}))$$

Loss of Regression ← squared error loss

$$\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

Loss of Multi-class classification with softmax regression ← cross-entropy loss

$$\mathcal{L}(\hat{y}, y) = -\sum_{j=1}^{k} \mathbf{1}\{y = j\} \log \hat{y}_j = -\sum_{j=1}^{k} y_j \log \hat{y}_j$$

Optimizing the Neural Network

$$J(\Theta) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\big(h_\Theta(\mathbf{x}_i), y_i\big) \quad + \frac{\lambda}{2n}\sum_{l=1}^{L-1}\sum_{i=1}^{s_{l-1}}\sum_{j=1}^{s_l}\left(\Theta_{ji}^{(l)}\right)^2$$

Solve via: $\min_{\Theta} J(\Theta)$

$J(\Theta)$ is not convex, so GD on a neural net yields a local optimum

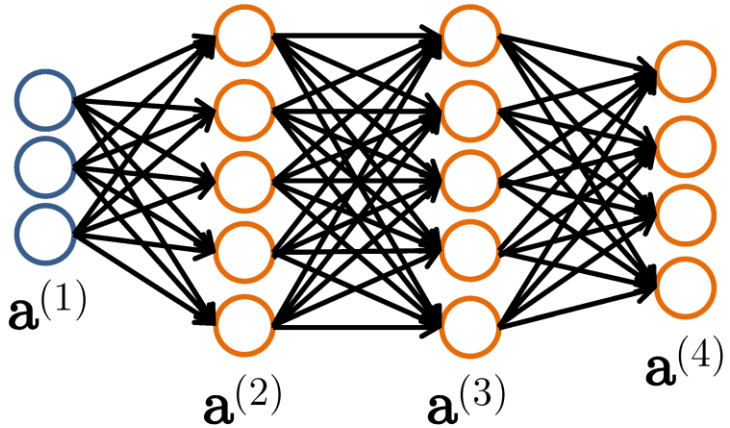
- But, tends to work well in practice

Need code to compute:

- $J(\Theta)$
- $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$

Forward Propagation

- Given one labeled training instance $(\mathbf{x}, y)$:



- $a^{(1)} = x$
- $z^{(2)} = \Theta^{(1)} a^{(1)}$
- $a^{(2)} = g(z^{(2)})$ [add $a_0^{(2)}$]
- $z^{(3)} = \Theta^{(2)} a^{(2)}$
- $a^{(3)} = g(z^{(3)})$ [add $a_0^{(3)}$]
- $z^{(4)} = \Theta^{(3)} a^{(3)}$
- $a^{(4)} = h_\Theta(x) = g(z^{(4)})$

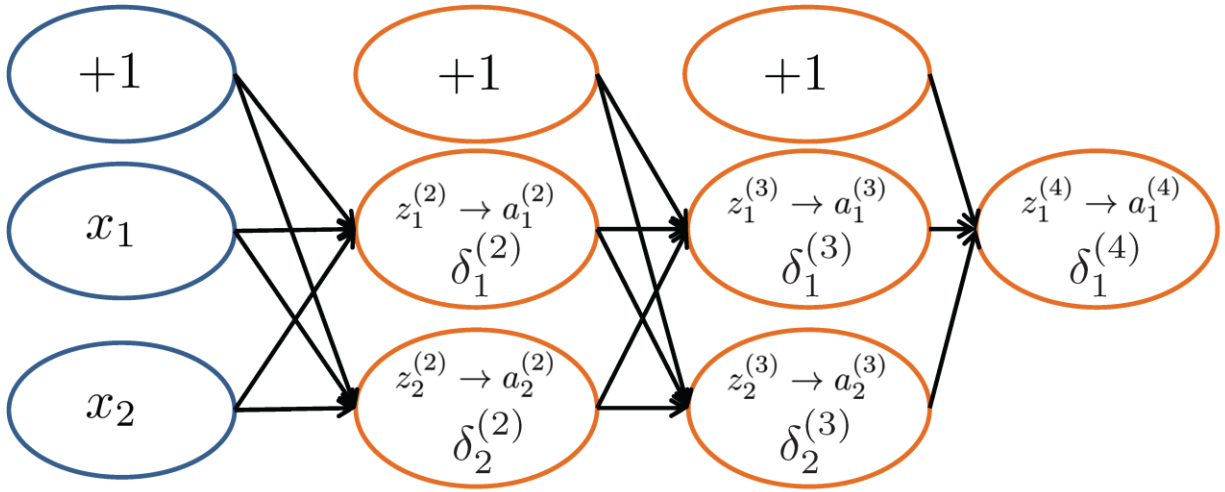
Backpropagation Intuition

- Each hidden node *j* is "responsible" for some fraction of the error $\delta_j^{(l)}$ in each of the output nodes to which it connects

- $\delta_j^{(l)}$ is divided according to the strength of the connection between hidden node and the output node

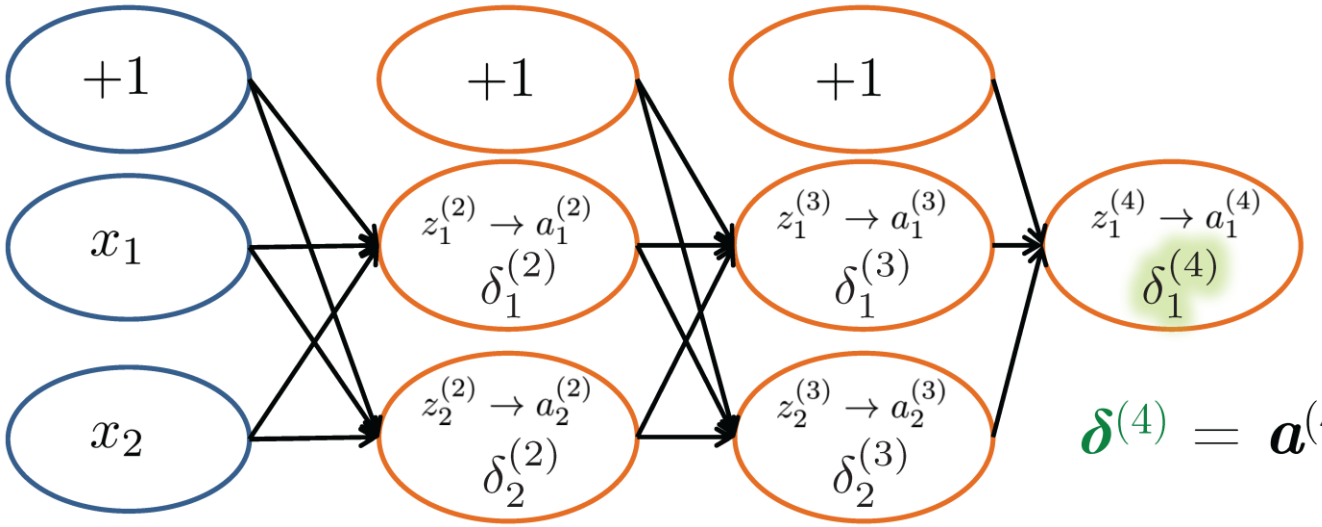
- Then, the "blame" is propagated back to provide the error values for the hidden layer

Backpropagation Intuition (binary classification)



$\delta_j^{(l)} =$ "error" of node j in layer l

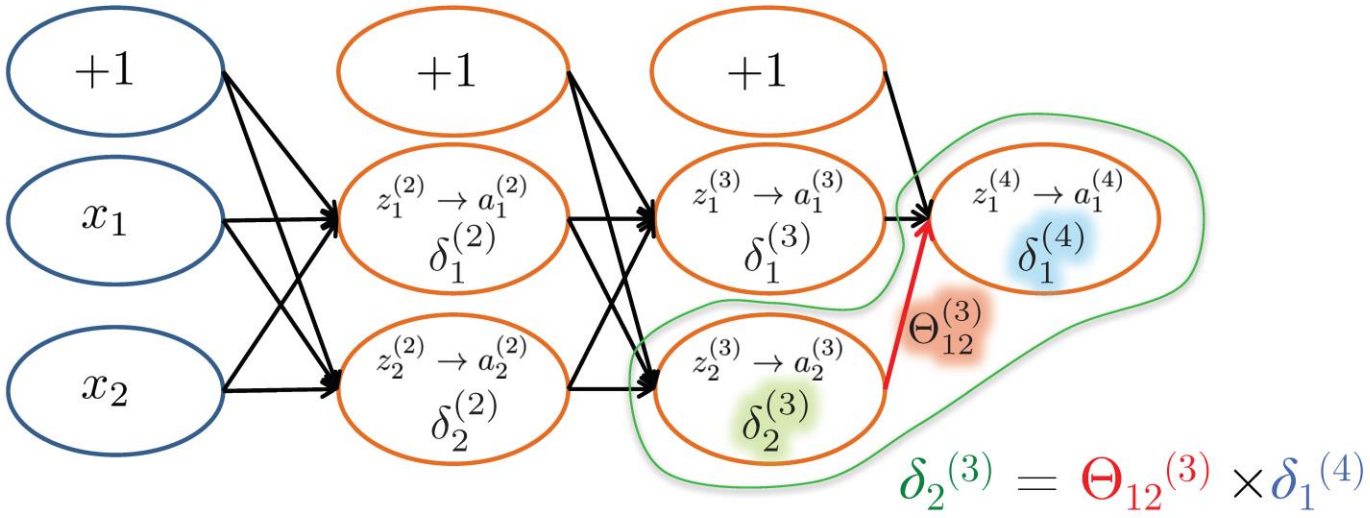
Backpropagation Intuition (binary classification)



$$\boldsymbol{\delta}^{(4)} = \boldsymbol{a}^{(4)} - \mathbf{y}$$

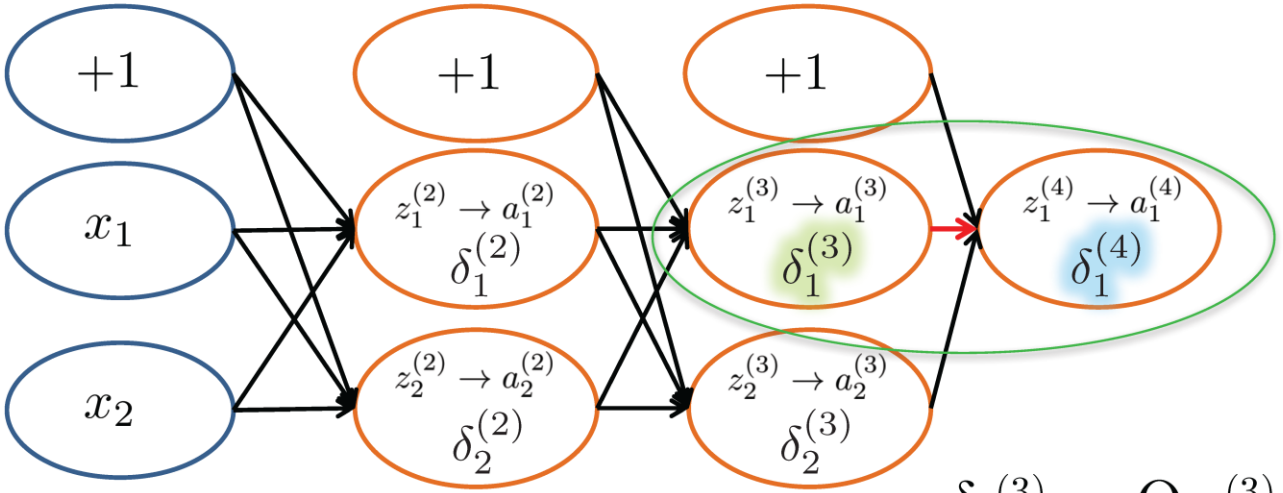
$\delta_j^{(l)} =$ “error” of node j in layer l

Backpropagation Intuition (binary classification)



$\delta_j^{(l)}$ = "error" of node j in layer l

Backpropagation Intuition (binary classification)

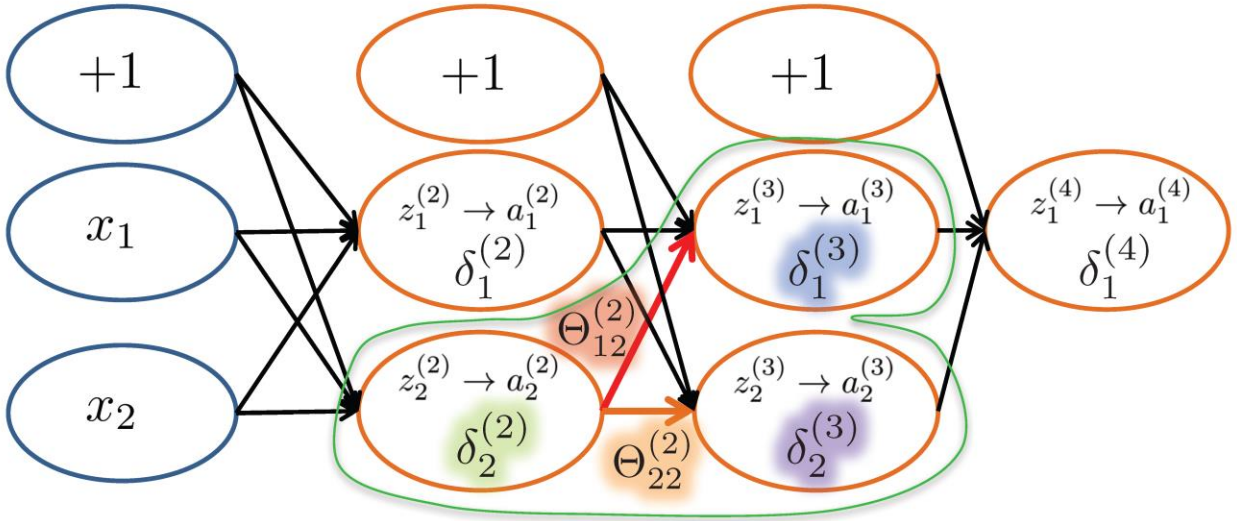


$$\delta_2^{(3)} = \Theta_{12}^{(3)} \times \delta_1^{(4)}$$

$$\delta_1^{(3)} = \Theta_{11}^{(3)} \times \delta_1^{(4)}$$

$\delta_j^{(l)}$ = "error" of node j in layer l

Backpropagation Intuition (binary classification)



$$\delta_2^{(2)} = \Theta_{12}^{(2)} \times \delta_1^{(3)} + \Theta_{22}^{(2)} \times \delta_2^{(3)}$$

$\delta_j^{(l)}$ = "error" of node j in layer l

Formally, $\delta_j^{(l)} = \dfrac{\partial}{\partial z_j^{(l)}} \text{cost}(\mathbf{x}_i) = \dfrac{\partial \text{cost}(x_i)}{\partial a_j^{(2)}} \times \dfrac{\partial a_j^{(2)}}{\partial z_j^{(2)}}$

where $\text{cost}(\mathbf{x}_i) = -y_i \log h_\Theta(\mathbf{x}_i) - (1 - y_i) \log(1 - h_\Theta(\mathbf{x}_i))$

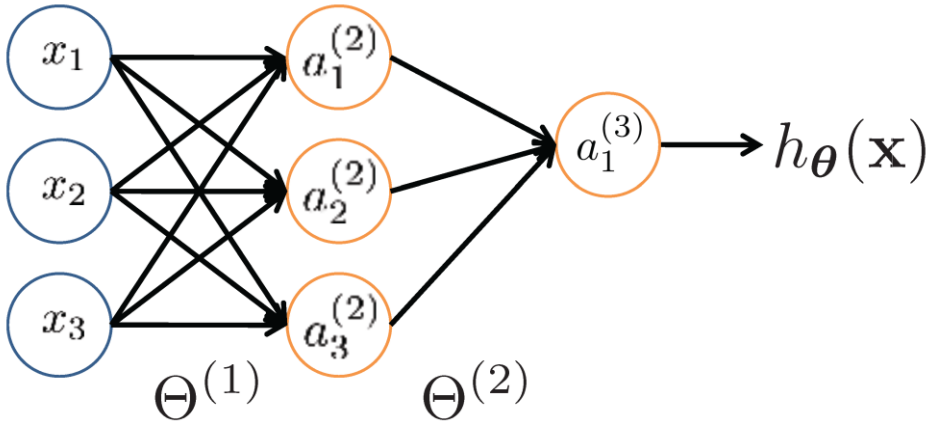
Vectorization

$$a_1^{(2)} = g\left(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3\right) = g\left(z_1^{(2)}\right)$$

$$a_2^{(2)} = g\left(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3\right) = g\left(z_2^{(2)}\right)$$

$$a_3^{(2)} = g\left(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3\right) = g\left(z_3^{(2)}\right)$$

$$h_\Theta(\mathbf{x}) = g\left(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)}\right) = g\left(z_1^{(3)}\right)$$



Feed-Forward Steps:

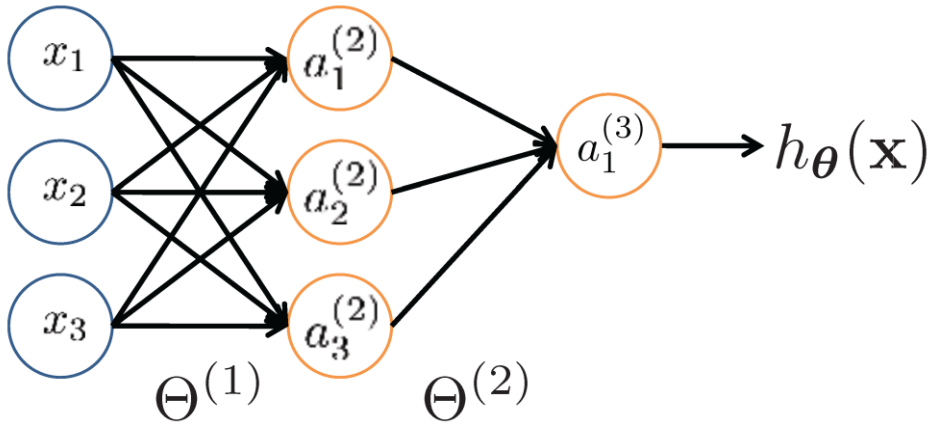
$$\mathbf{z}^{(2)} = \Theta^{(1)}\mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

$$\text{Add } a_0^{(2)} = 1$$

$$\mathbf{z}^{(3)} = \Theta^{(2)}\mathbf{a}^{(2)}$$

$$h_\Theta(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$



Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)}\mathbf{x}$$

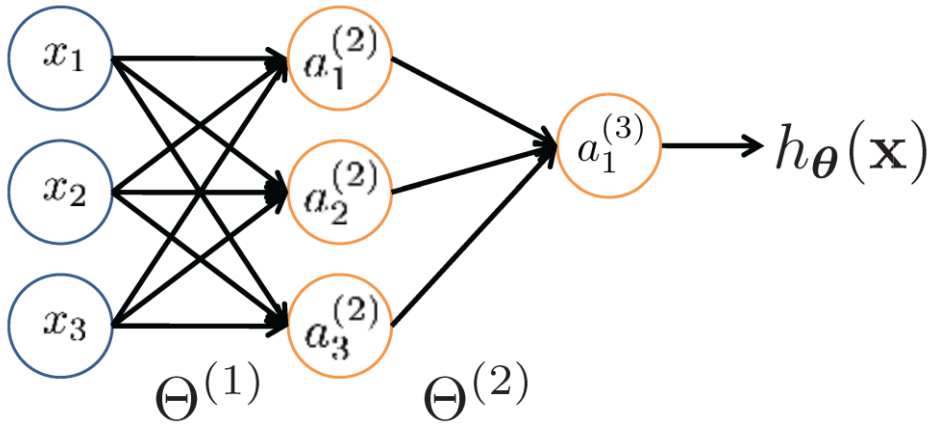
$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

$$\text{Add } a_0^{(2)} = 1$$

$$\mathbf{z}^{(3)} = \Theta^{(2)}\mathbf{a}^{(2)}$$

$$h_\Theta(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

$$\text{where } \text{cost}(\mathbf{x}_i) = -y_i \log h_\Theta(\mathbf{x}_i) - (1 - y_i) \log(1 - h_\Theta(\mathbf{x}_i))$$



Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)}\mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

$$\text{Add } a_0^{(2)} = 1$$

$$\mathbf{z}^{(3)} = \Theta^{(2)}\mathbf{a}^{(2)}$$

$$h_\Theta(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

where $\text{cost}(\mathbf{x}_i) = -y_i \log h_\Theta(\mathbf{x}_i) - (1 - y_i) \log(1 - h_\Theta(\mathbf{x}_i))$

$$\frac{\partial \text{cost}(x_i)}{\partial \Theta^{(2)}} = \frac{\partial \text{cost}(x_i)}{\partial a^{(3)}} \cdot \frac{\partial a^{(3)}}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial \Theta^{(2)}} = \frac{a^{(3)} - y}{a^{(3)}(1-a^{(3)})} \cdot a^{(3)}(1-a^{(3)}) \cdot a^{(2)} = a^{(2)} \cdot (a^{(3)} - y)$$

$$\frac{\partial \text{cost}(x_i)}{\partial \Theta^{(1)}} = \frac{\partial \text{cost}(x_i)}{\partial a^{(3)}} \cdot \frac{\partial a^{(3)}}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial \Theta^{(1)}} = (a^{(3)} - y) \cdot \Theta^{(2)} \cdot a^{(2)}(1 - a^{(2)}) \cdot x_i$$

Training a Neural Network via Gradient Descent with Backprop

Given: training set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
Initialize all $\Theta^{(l)}$ randomly (NOT to 0!)
Loop // each iteration is called an epoch
    Set $\Delta_{ij}^{(l)} = 0 \quad \forall l, i, j$ (Used to accumulate gradient)
    For each training instance $(\mathbf{x}_i, y_i)$:
        Set $\mathbf{a}^{(1)} = \mathbf{x}_i$
        Compute $\{\mathbf{a}^{(2)}, \ldots, \mathbf{a}^{(L)}\}$ via forward propagation
        Compute $\boldsymbol{\delta}^{(L)} = \mathbf{a}^{(L)} - y_i$
        Compute errors $\{\boldsymbol{\delta}^{(L-1)}, \ldots, \boldsymbol{\delta}^{(2)}\}$
        Compute gradients $\Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$
    Compute avg regularized gradient $D_{ij}^{(l)} = \begin{cases} \frac{1}{n}\Delta_{ij}^{(l)} + \lambda\Theta_{ij}^{(l)} & \text{if } j \neq 0 \\ \frac{1}{n}\Delta_{ij}^{(l)} & \text{otherwise} \end{cases}$
    Update weights via gradient step $\Theta_{ij}^{(l)} = \Theta_{ij}^{(l)} - \alpha D_{ij}^{(l)}$
Until weights converge or max #epochs is reached

Backpropagation

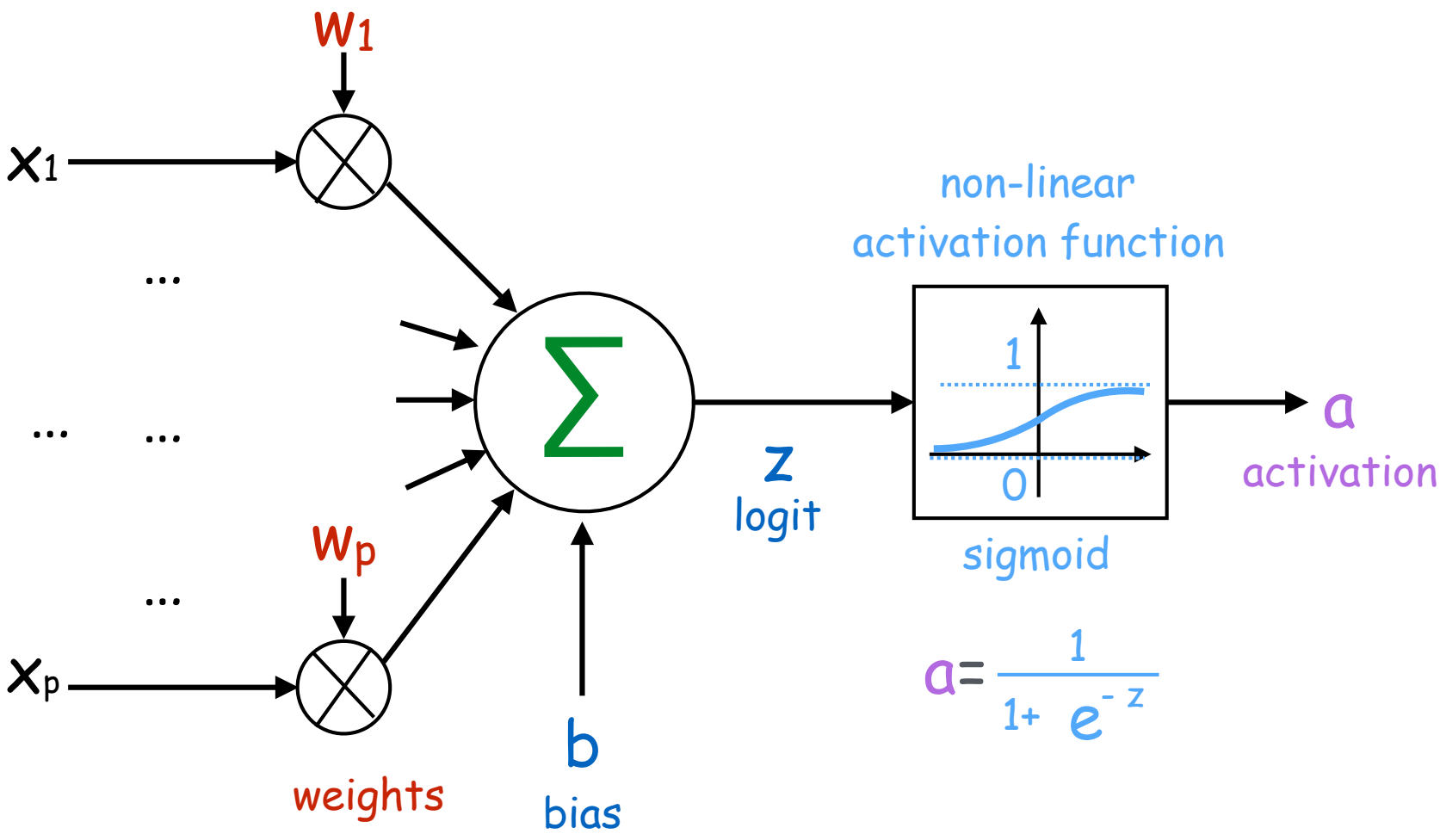
Multi-Class Classification

(Softmax regression and Fully Connected Neural Network)

Softmax Regression

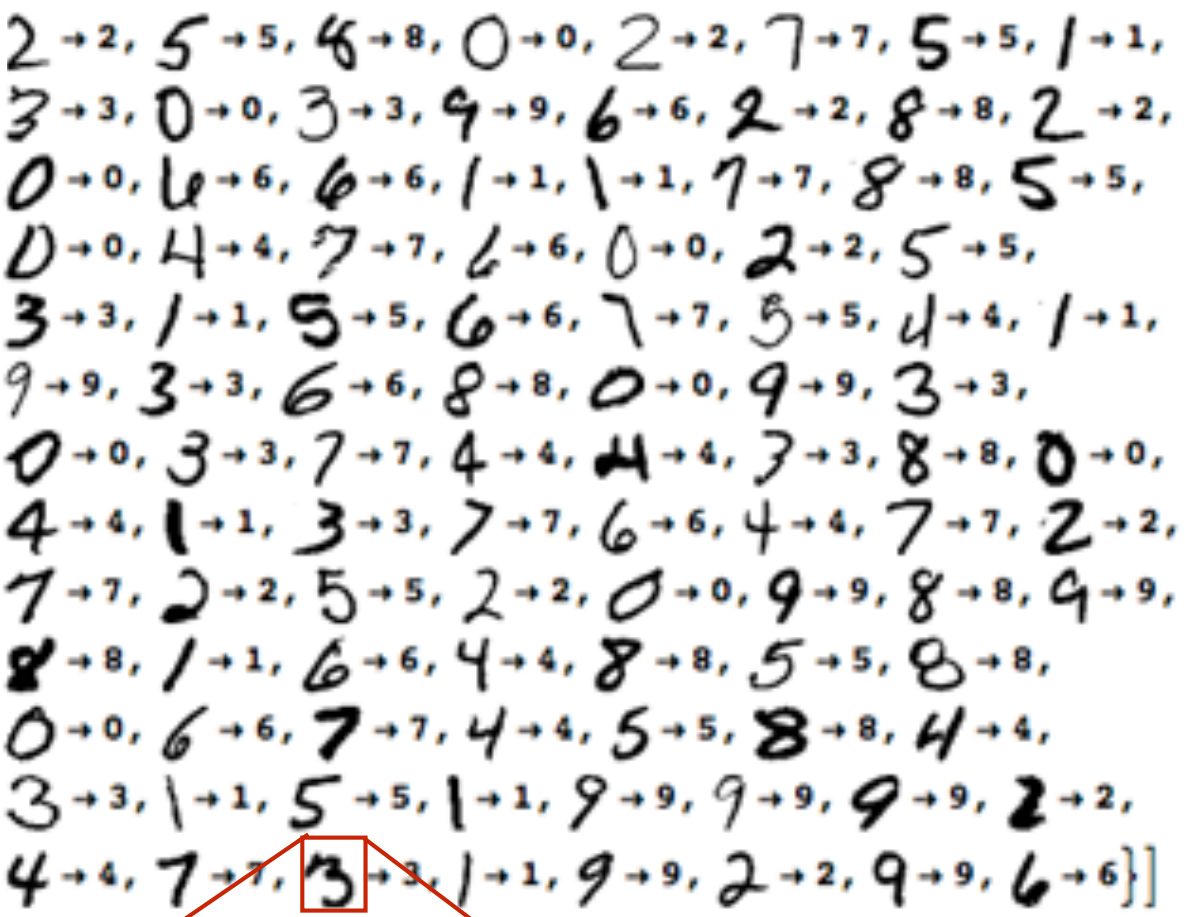
:a generalization of logistic regression to the case where we want to handle multiple classes

Logistic Regression



Multi-class Classification Problem

data set



candidate labels (classes)

0

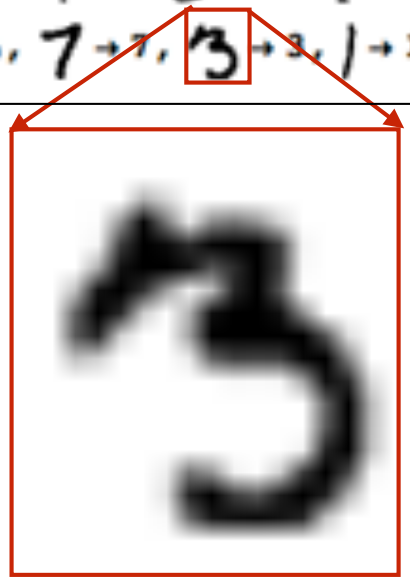
1

2

⋮

9

input (instance)



output (label)

0 or 1 or … or 9

Feature Matrix X

(n by p)

Feature (pixel)

Instance

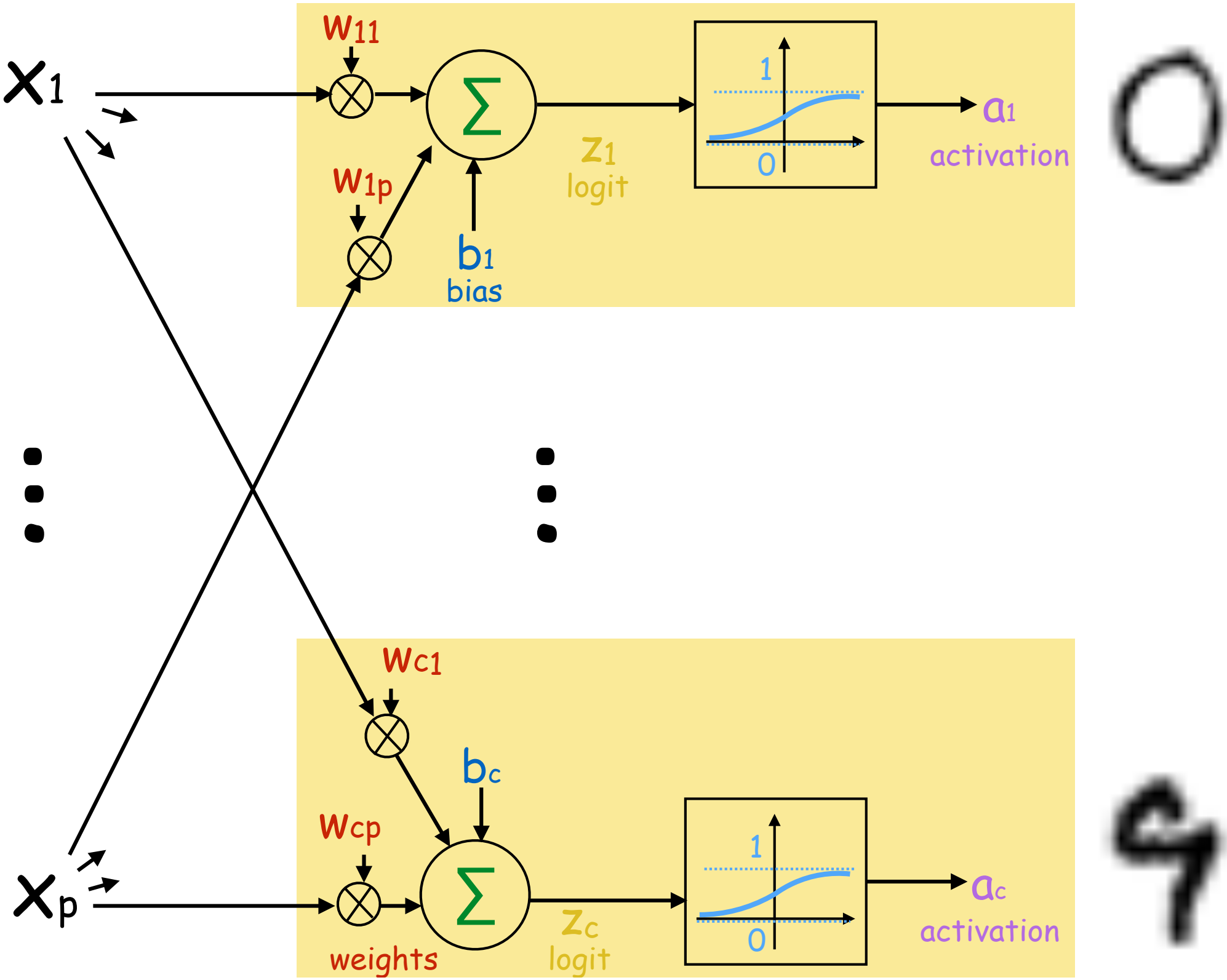
		1	2				…					p
x_1	2	1	3	4	3	8	3	5	7	9	7	3	4
x_2	5	3	3	5	7	7	0	4	1	2	1	9	7
	0	7	0	4	1	1	4	3	7	8	6	2	7
	3	1	4	3	7	9	7	3	2	7	0	4	1
	8	7	7	3	2	2	1	9	8	1	4	3	7
…	0	2	1	9	8	8	6	2	0	7	7	3	2
	9	8	6	2	0	0	4	1	1	4	1	9	8
	6	0	2	1	4	1	3	7	9	7	6	2	0
	7	3	5	3	3	7	3	2	2	1	2	1	3
x_n	6	1	7	2	3	2	2	1	2	3	5	3	1

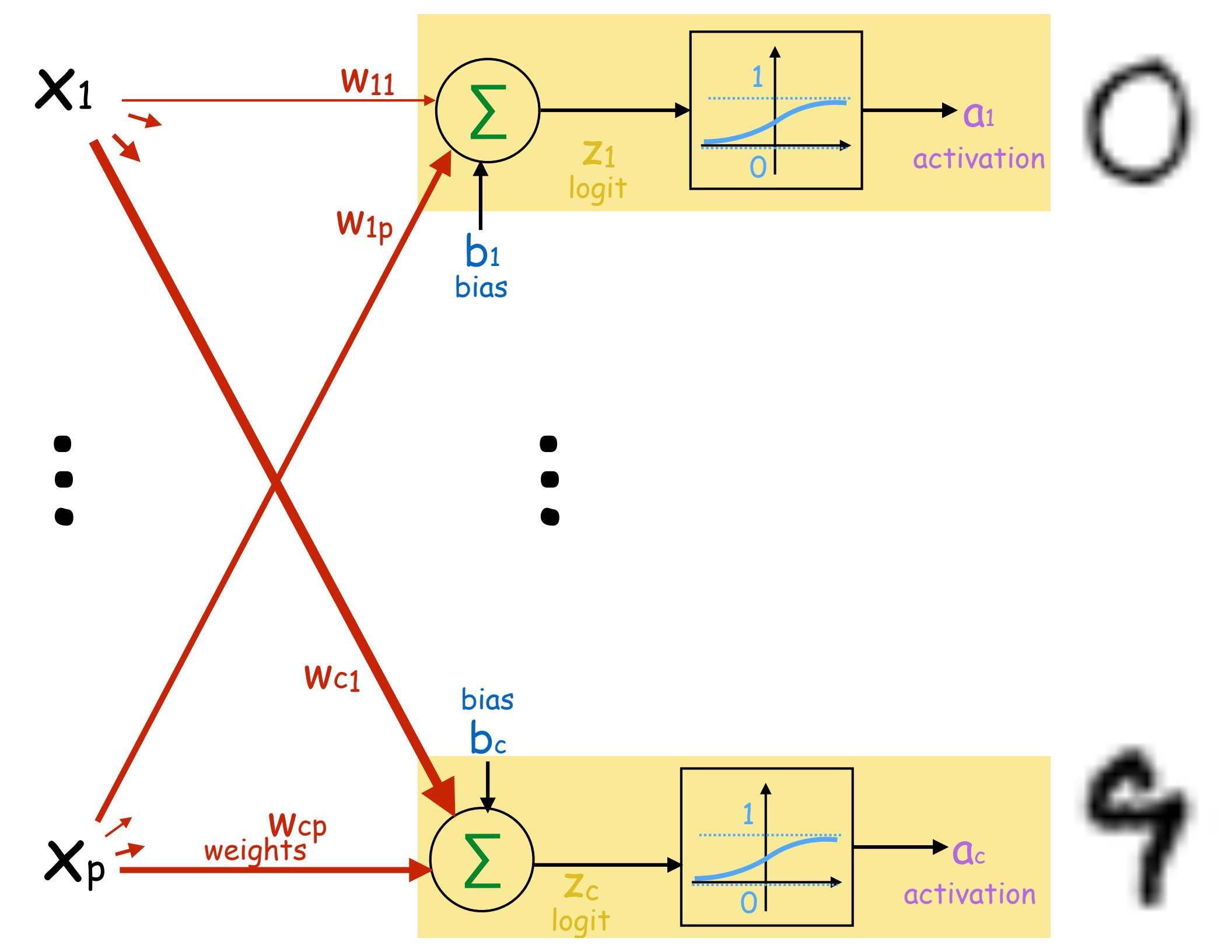
n - # instances p - # features

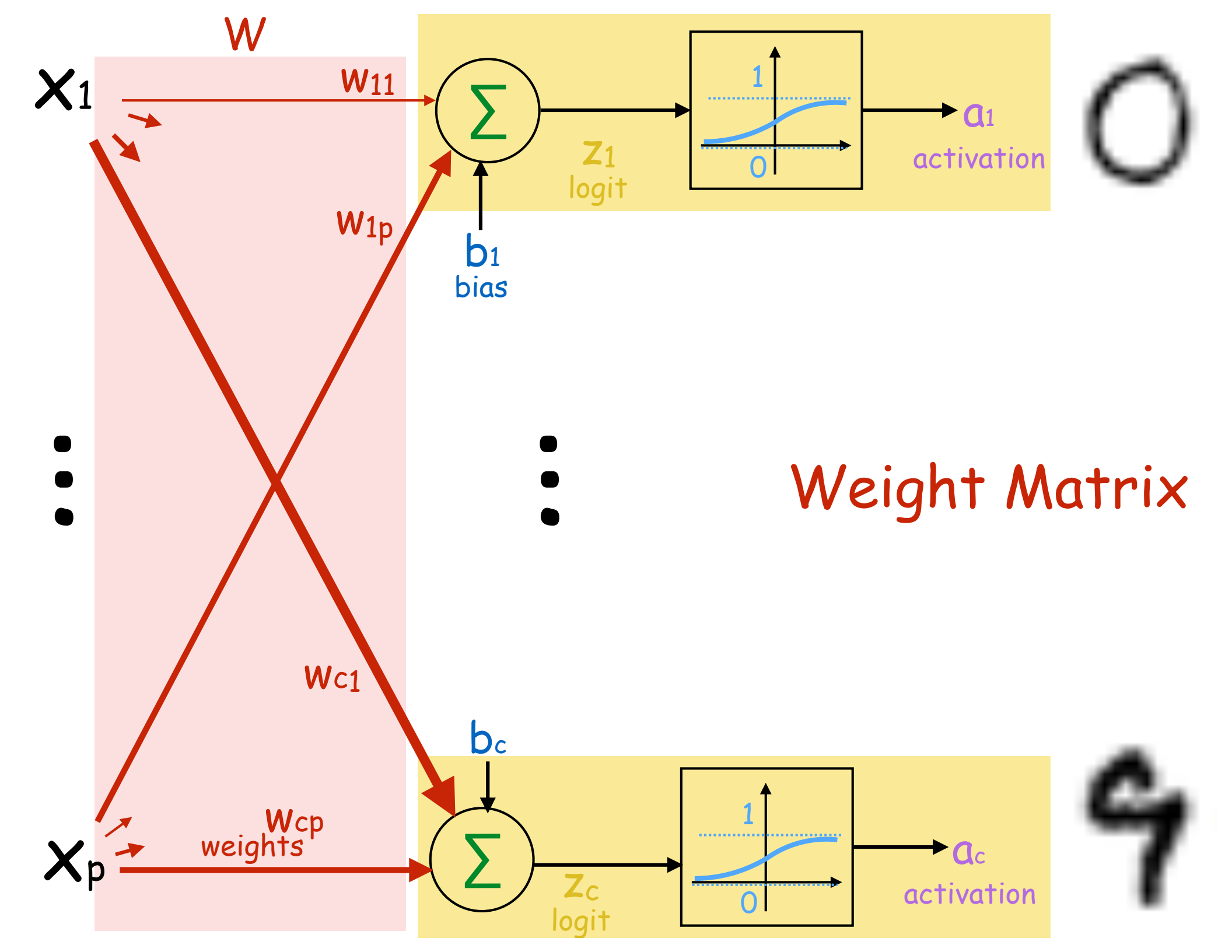
Label Vector y

(length n)

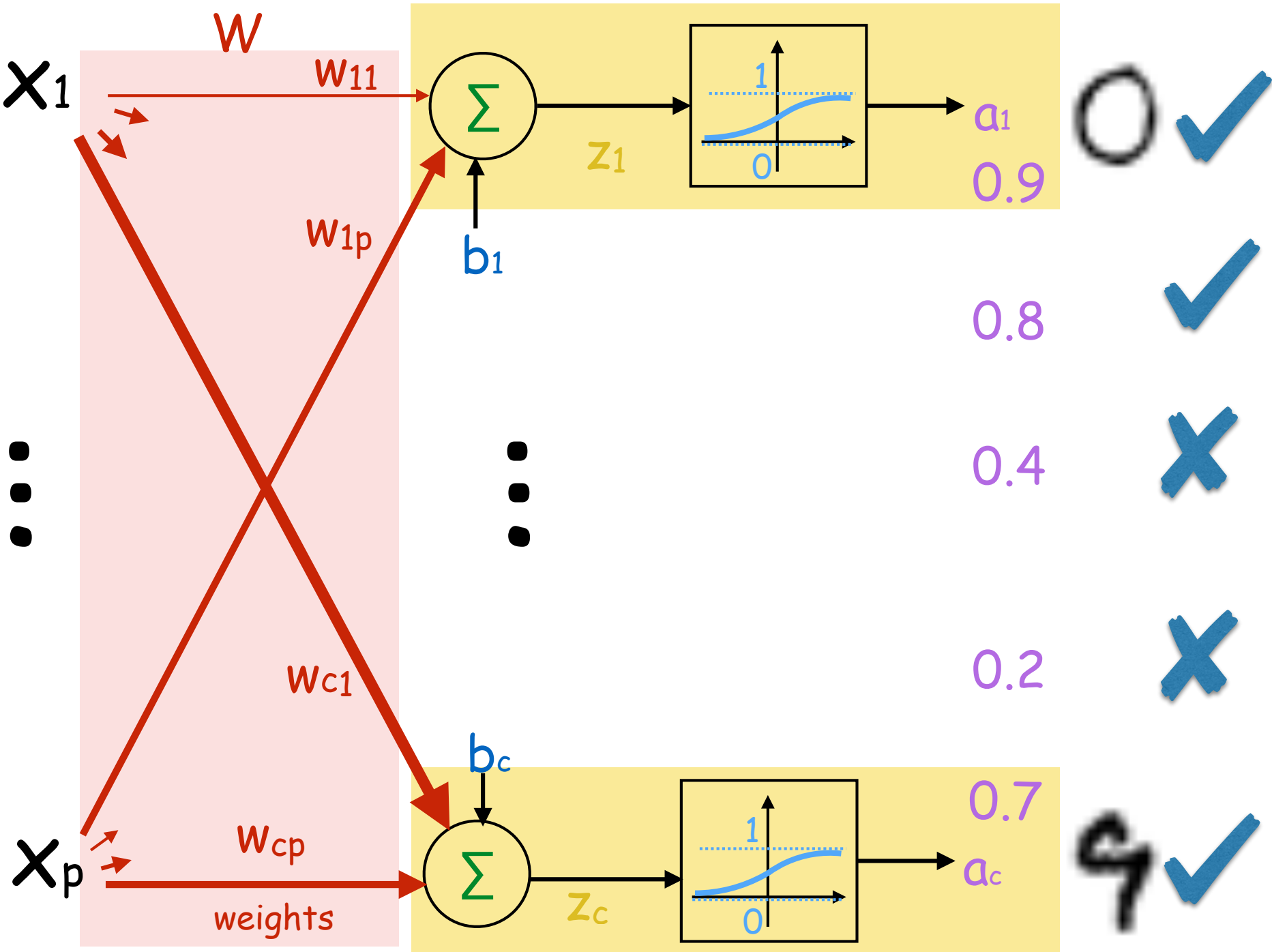
y_1	2
y_2	5
	0
	3
	8
…	0
	9
	6
	7
y_n	6


x_1
w_{11}
w_{1p}
b_1
bias
z_1
logit
1
0
a_1
activation
x_p
w_{c1}
b_c
w_{cp}
weights
z_c
logit
1
0
a_c
activation

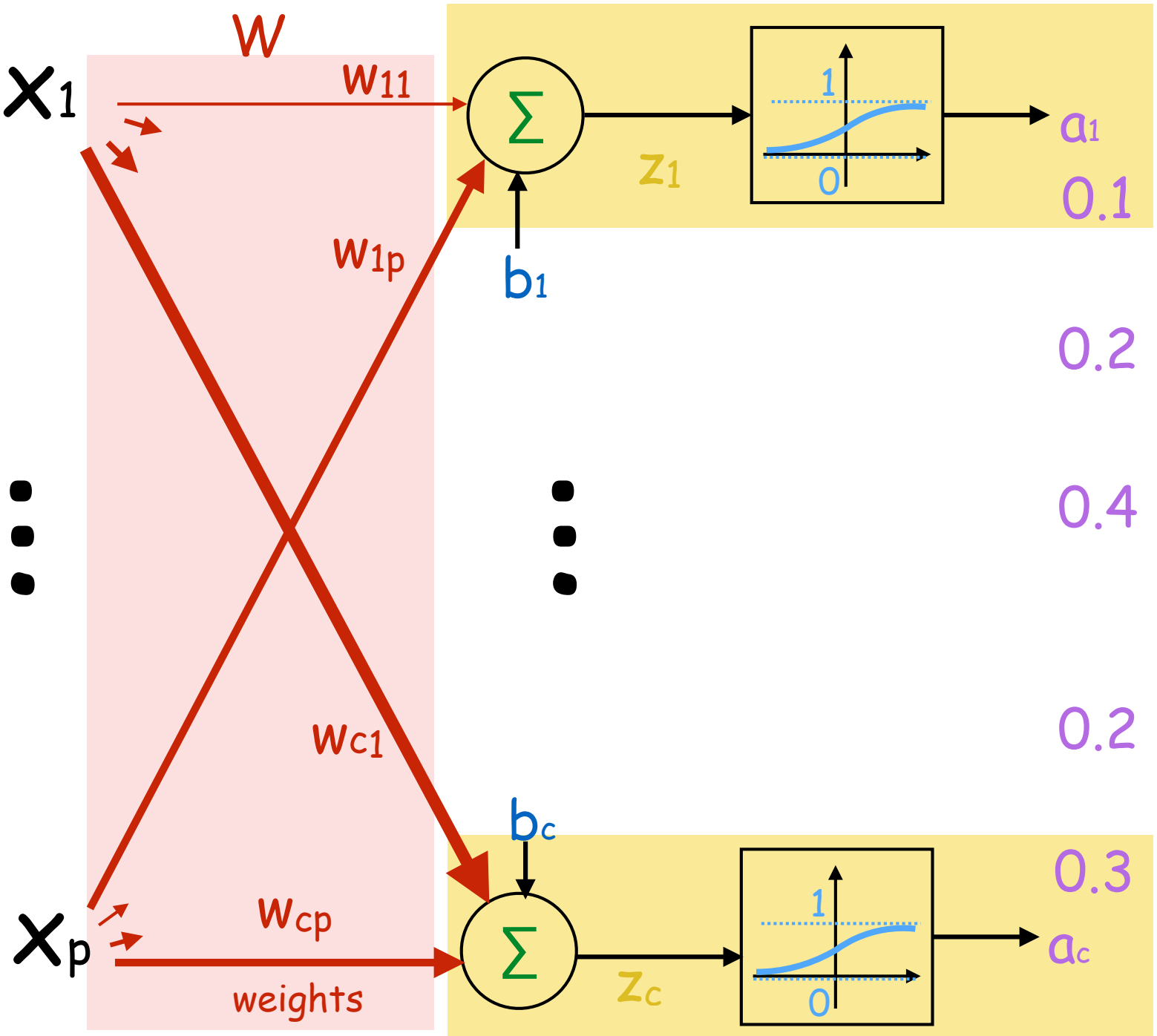

x_1
w_{11}
w_{1p}
b_1
bias
z_1
logit
1
0
a_1
activation
w_{c1}
bias
b_c
x_p
w_{cp}
weights
z_c
logit
1
0
a_c
activation


W
x_1
w_{11}
w_{1p}
b_1
bias
z_1
logit
1
0
a_1
activation
w_{c1}
b_c
x_p
w_{cp}
weights
z_c
logit
1
0
a_c
activation
Weight Matrix

Independent Outputs



Independent Outputs



0.1

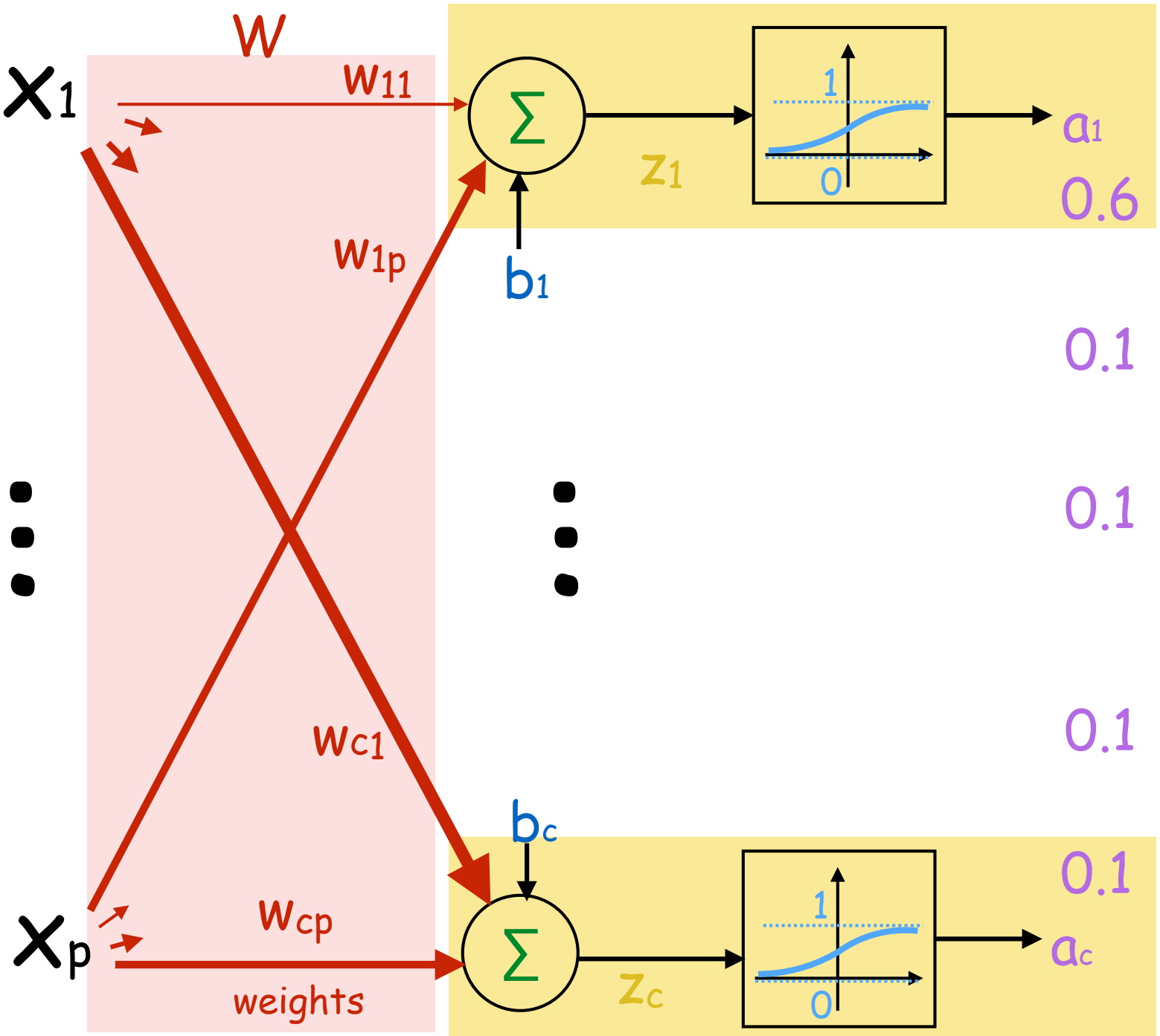
0.2

0.4

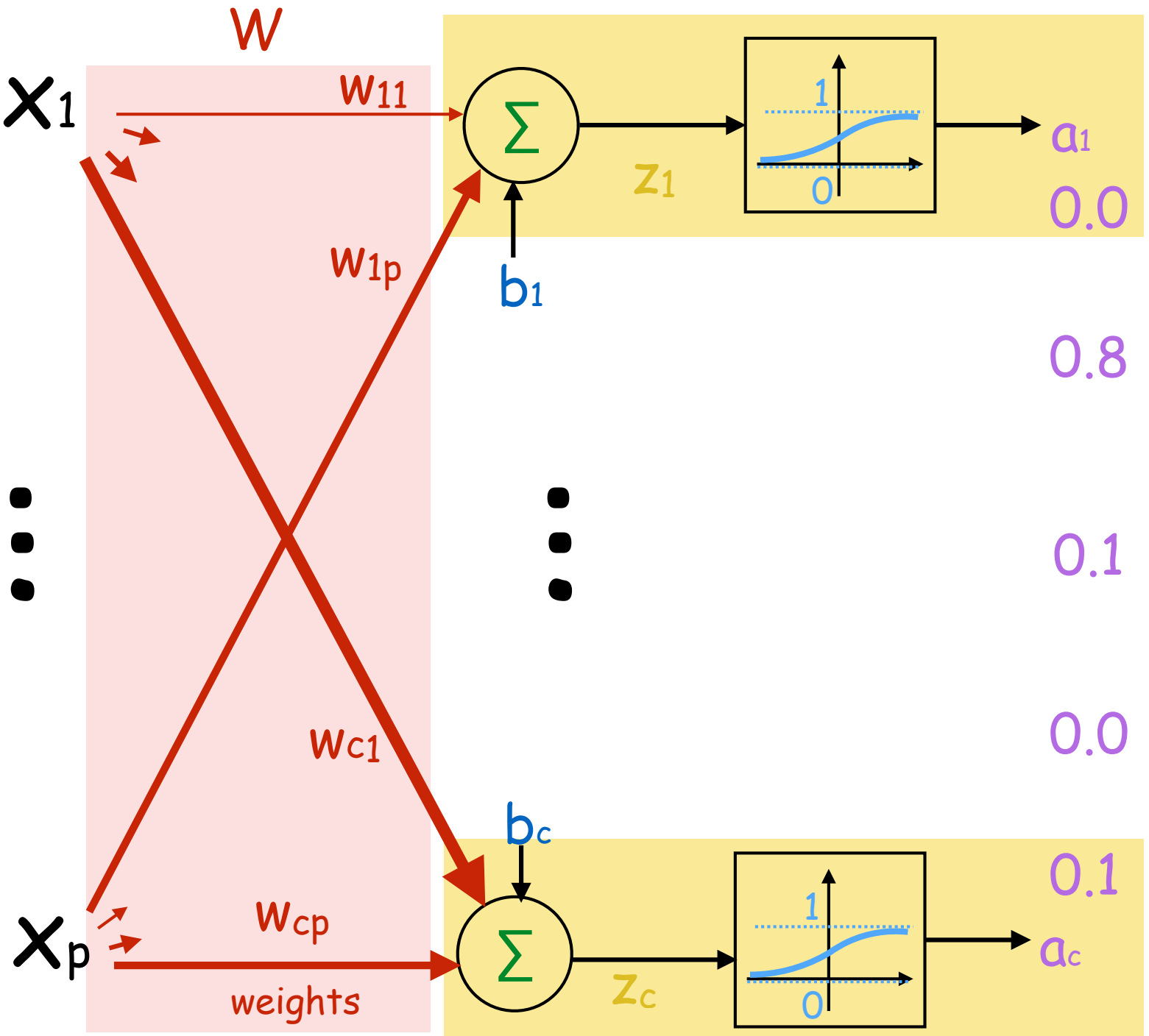
0.2

0.3

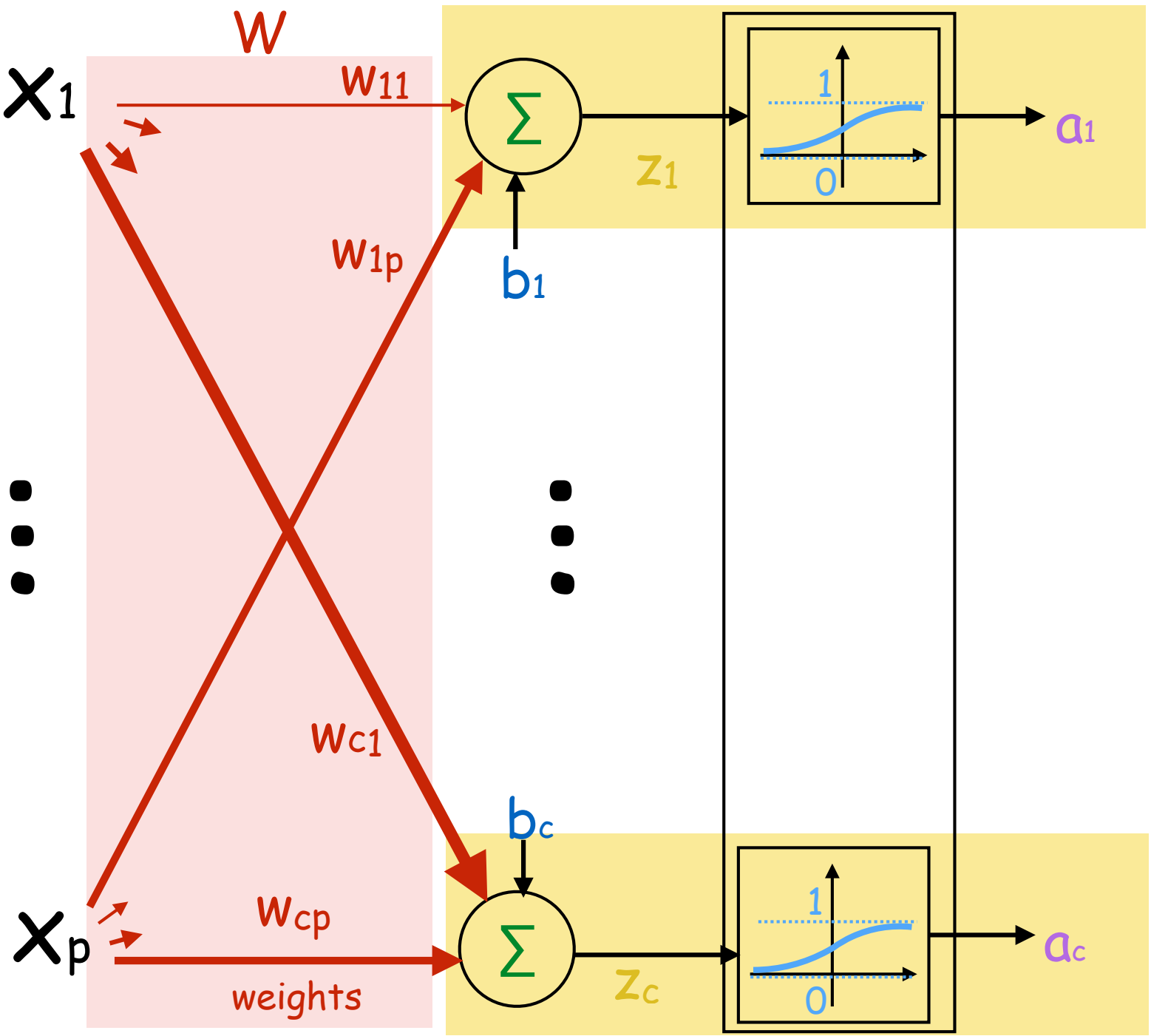
The outputs we need



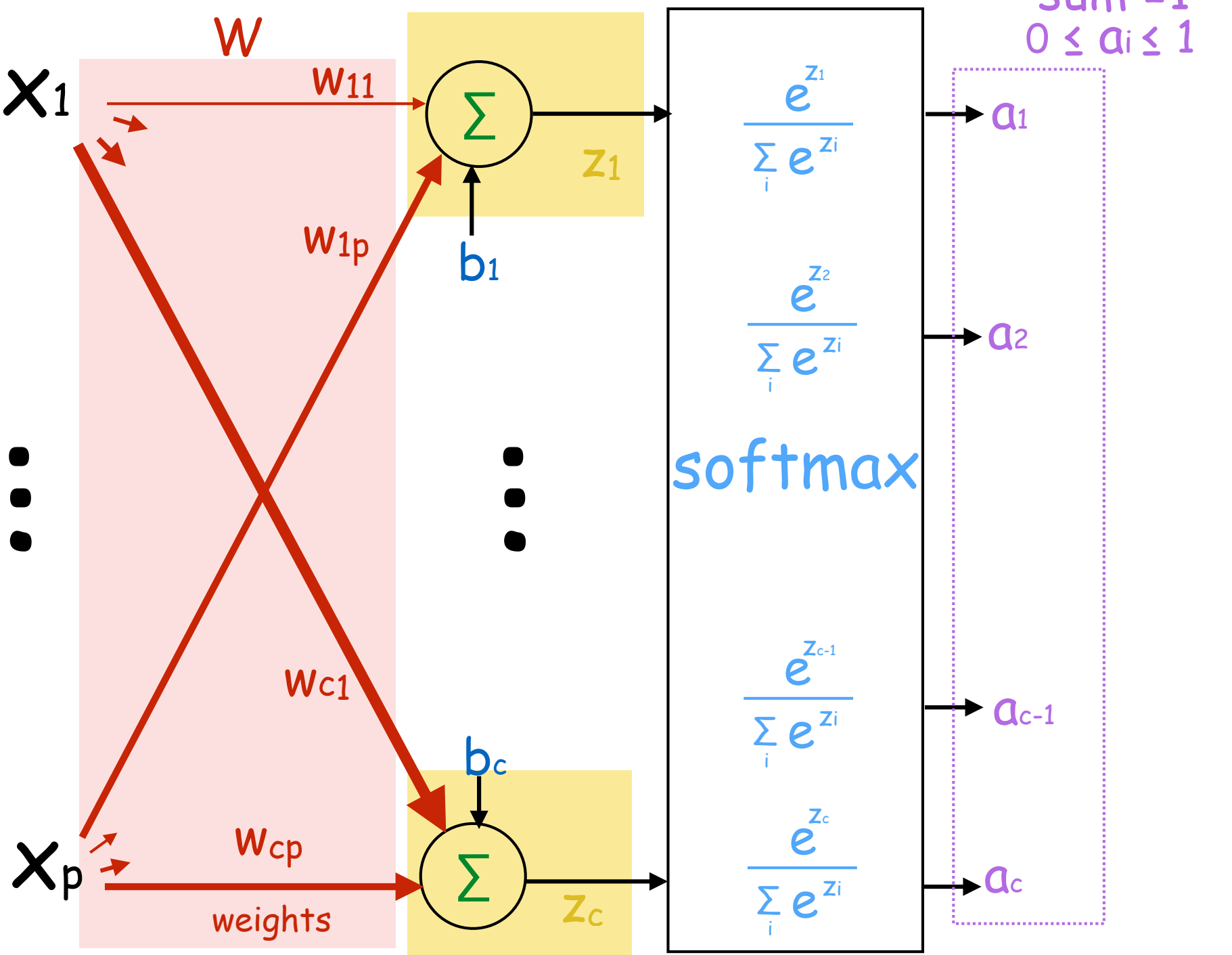
The outputs we need



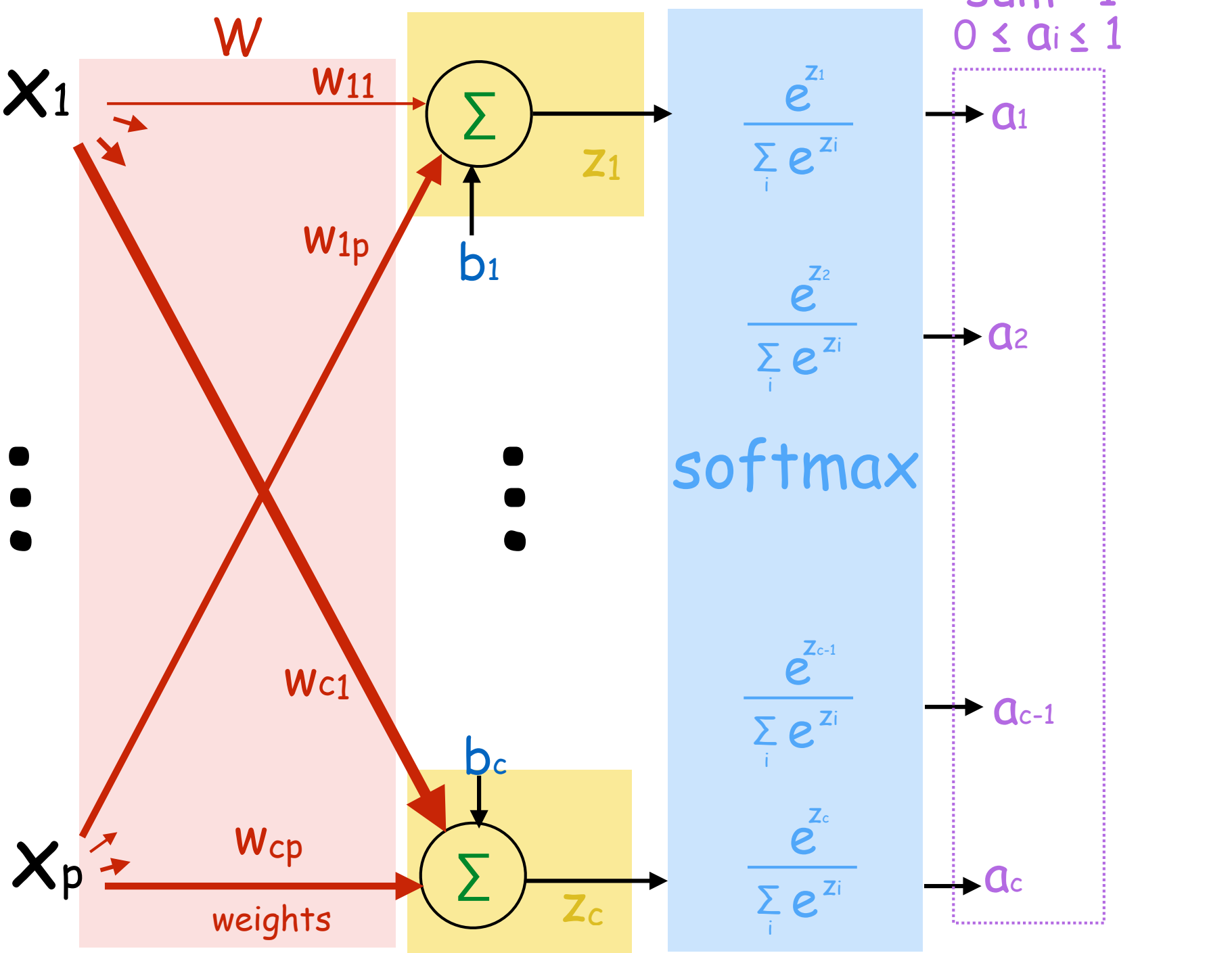
Coordinated Outputs



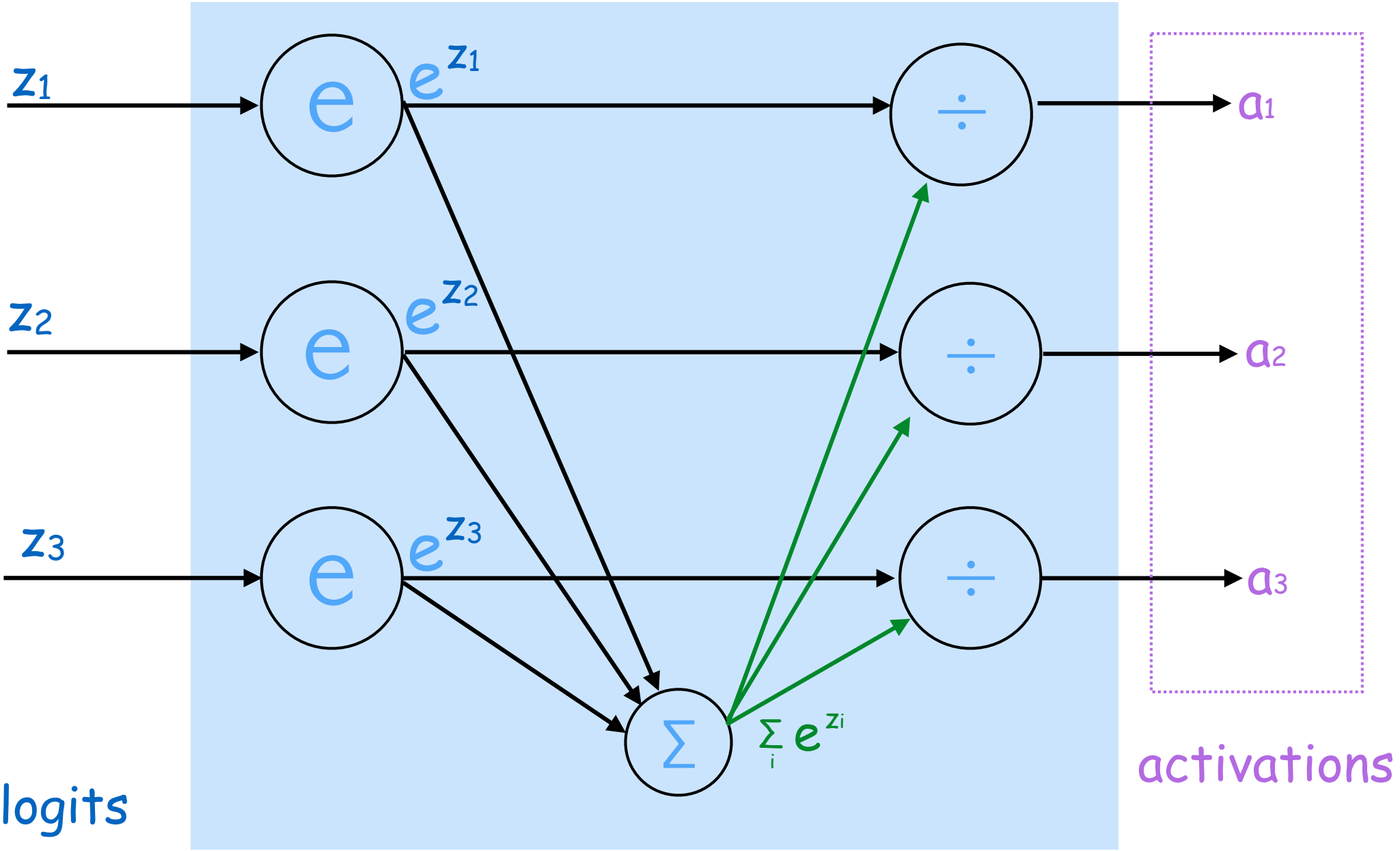
Softmax Activation



Softmax Activation

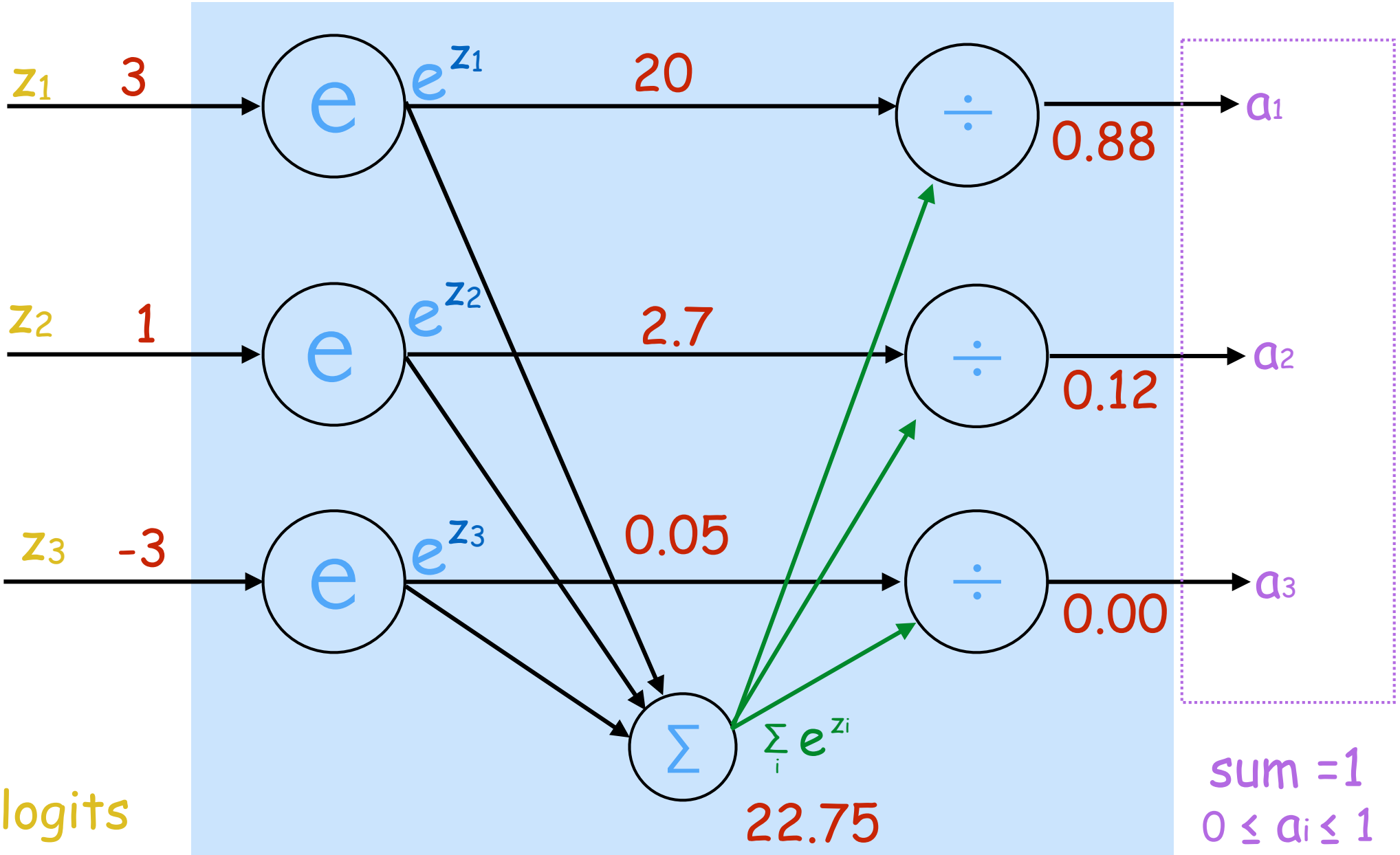


Softmax Activation



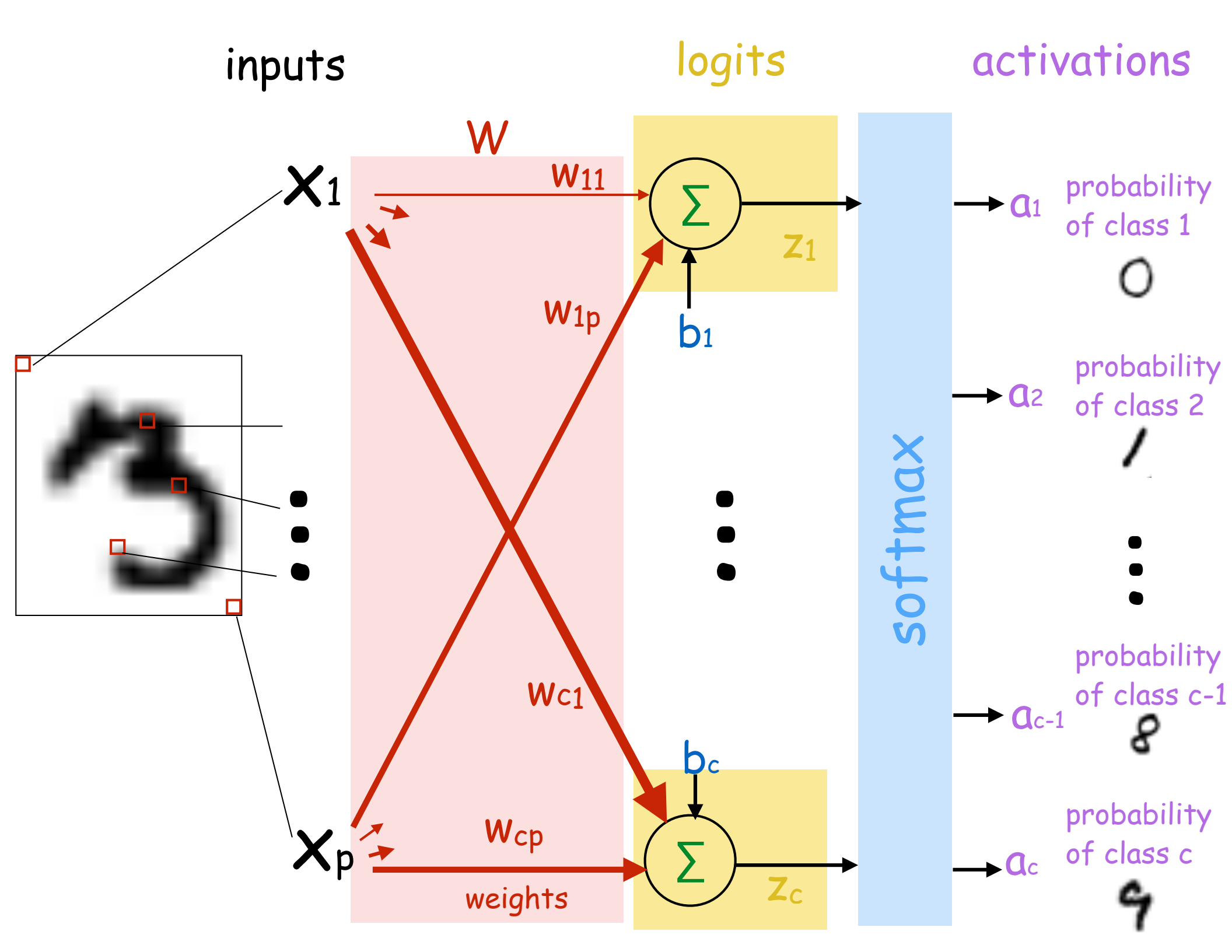
Softmax

Softmax Activation

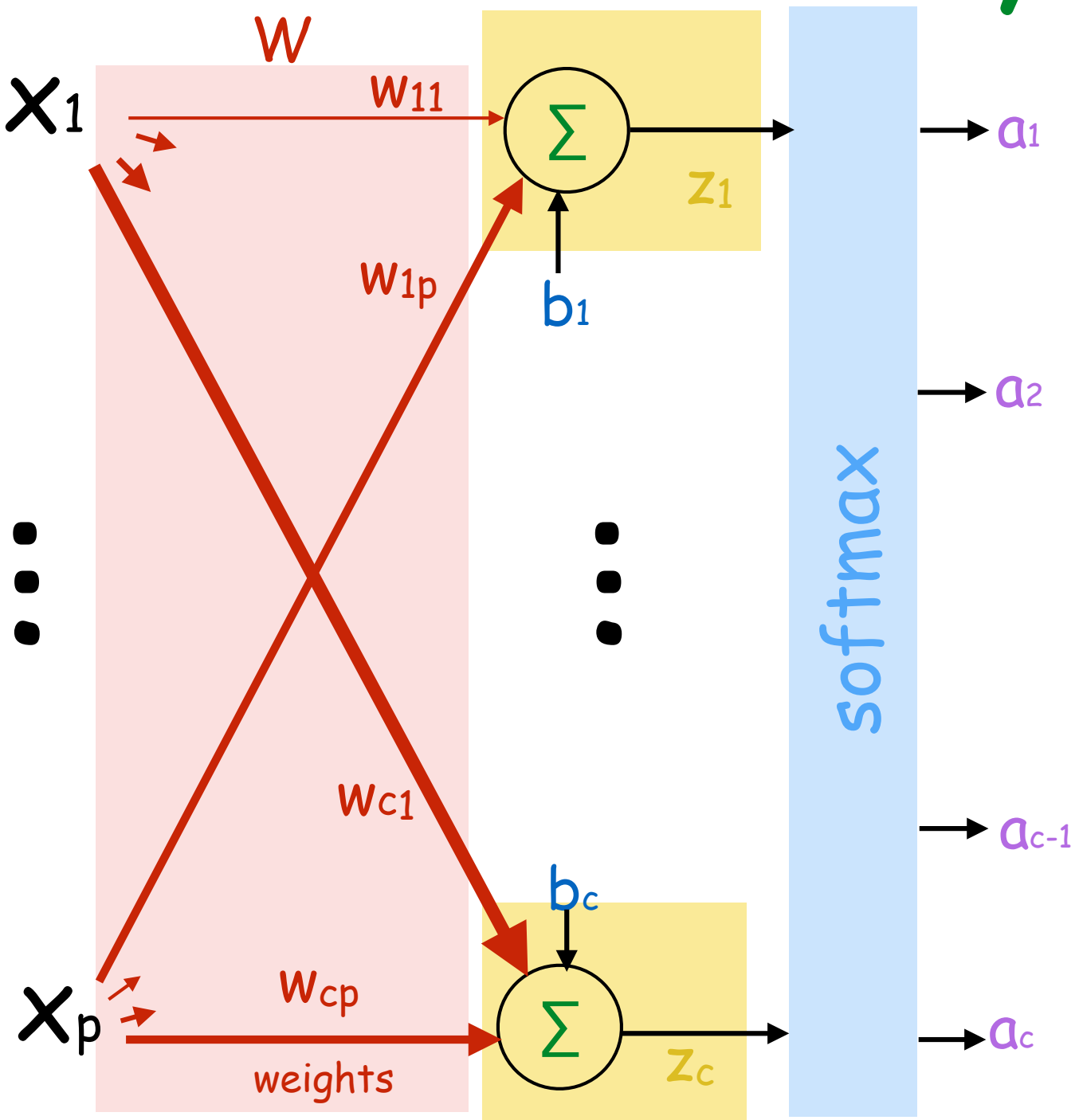


Softmax

Probabilities

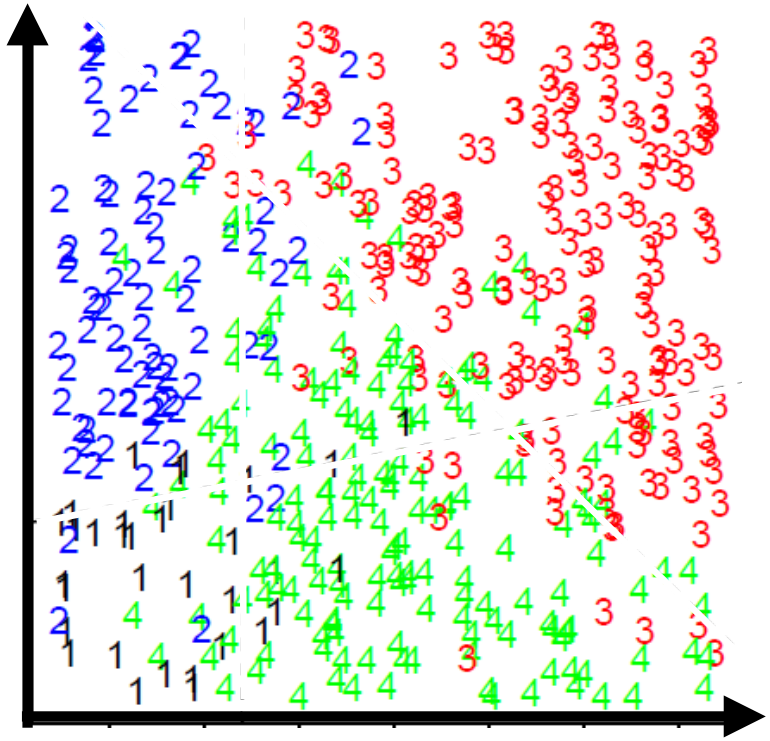

inputs
logits
activations
W
x_1
x_p
w_{11}
w_{1p}
w_{c1}
w_{cp}
weights
b_1
b_c
z_1
z_c
softmax
a_1
a_2
a_{c-1}
a_c
probability of class 1
probability of class 2
probability of class c-1
probability of class c

Parameters W, b



Training Model

training set

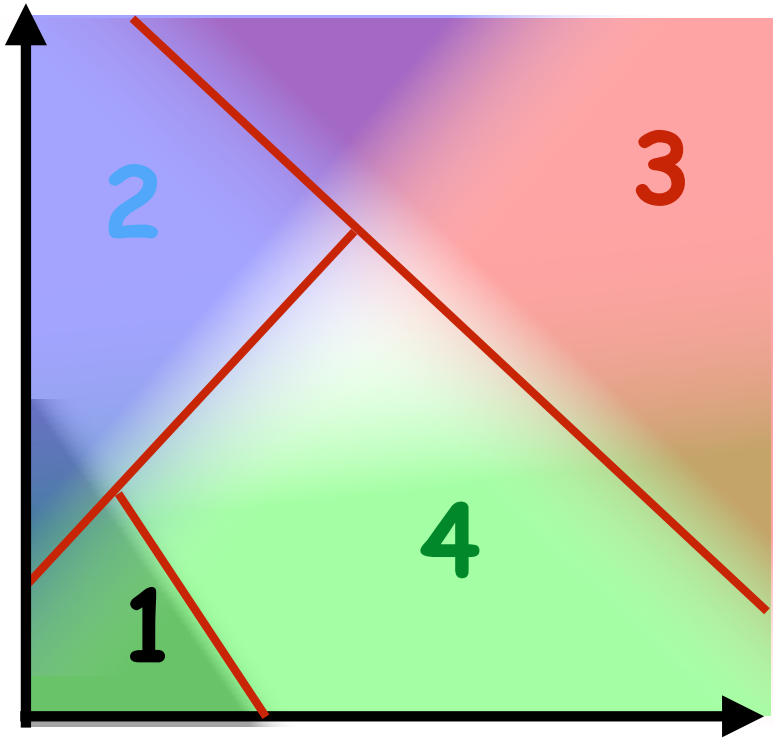


Feature Matrix $X_{(n \text{ by } p)}$

Label Vector Y

(length n) (0, 1, 2,... value)

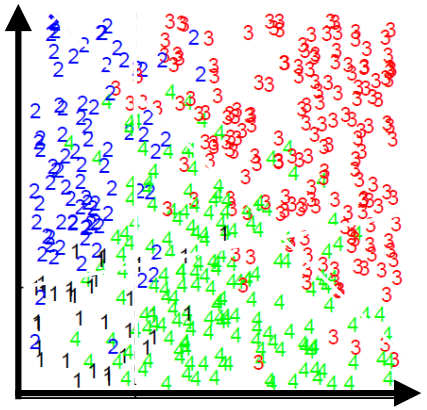
learn parameters



Weight Matrix **W** (shape c by p)

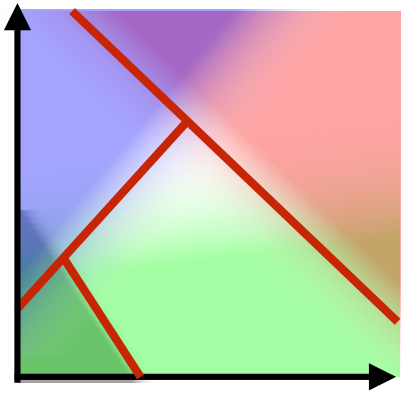
bias vector b (length c)

negative log Likelihood



Label: y 3

observed data



Weights **w**

biases b

Outputs: a_1 .4

probabilities a_2 .2

a_3 .1

a_c .1

y 1

a_1 .4

a_2 .2

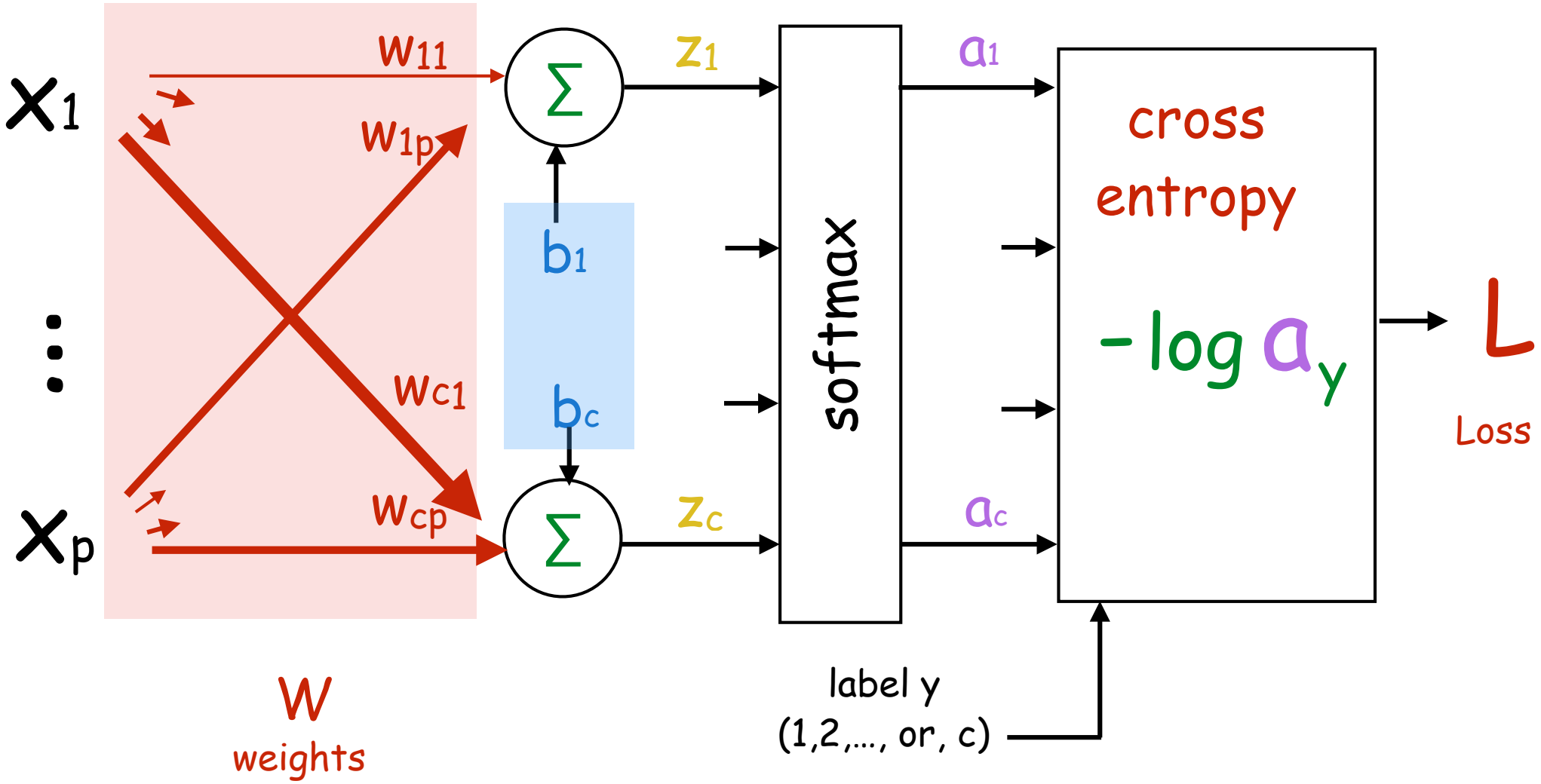
a_3 .1

a_c .1

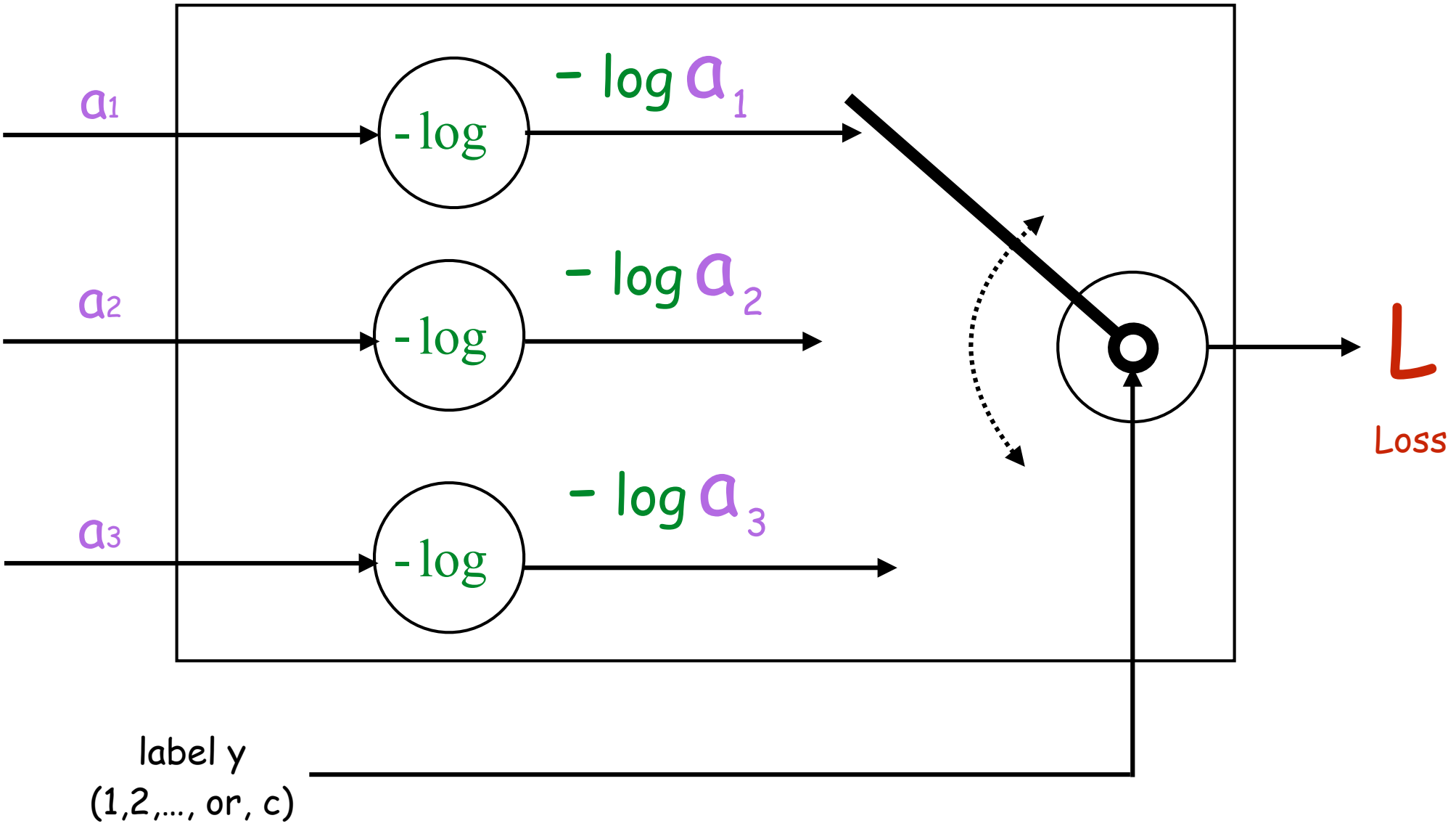
- log Likelihood = $-\log a_y$

Multi-class cross entropy loss

Multi-Class Cross Entropy Loss

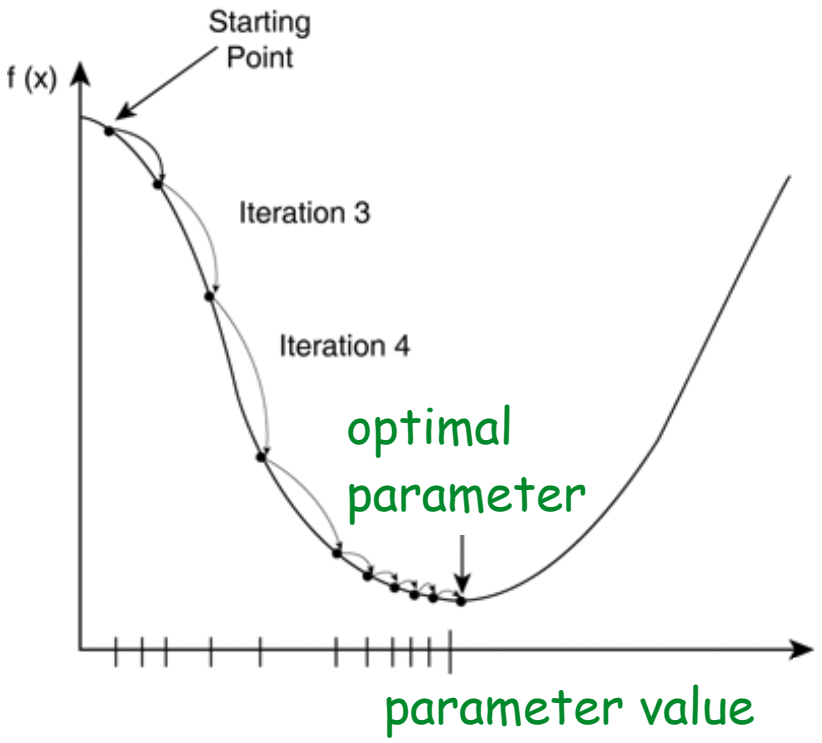


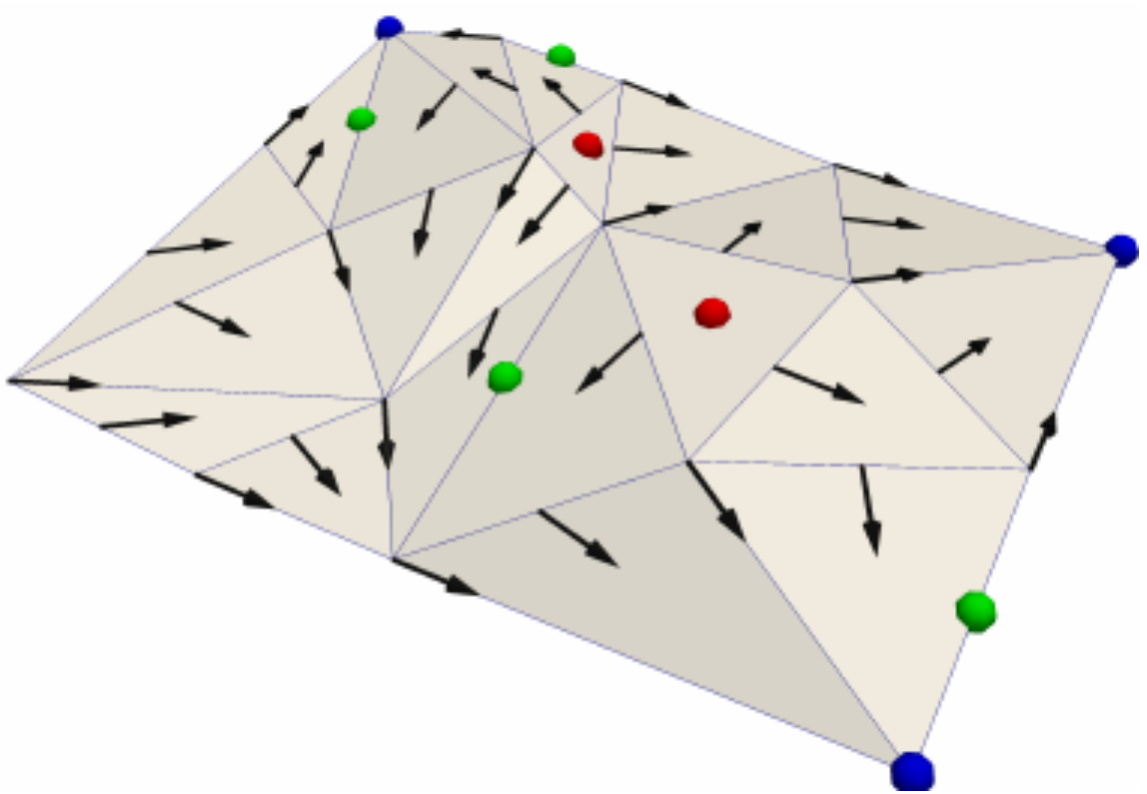
Multi-Class Cross Entropy Loss



Gradient Descent

L= loss (w)





$$W \leftarrow W - a \frac{\partial L}{\partial W}$$

$$b \leftarrow b - a \frac{\partial L}{\partial b}$$

a step size (a constant scalar)

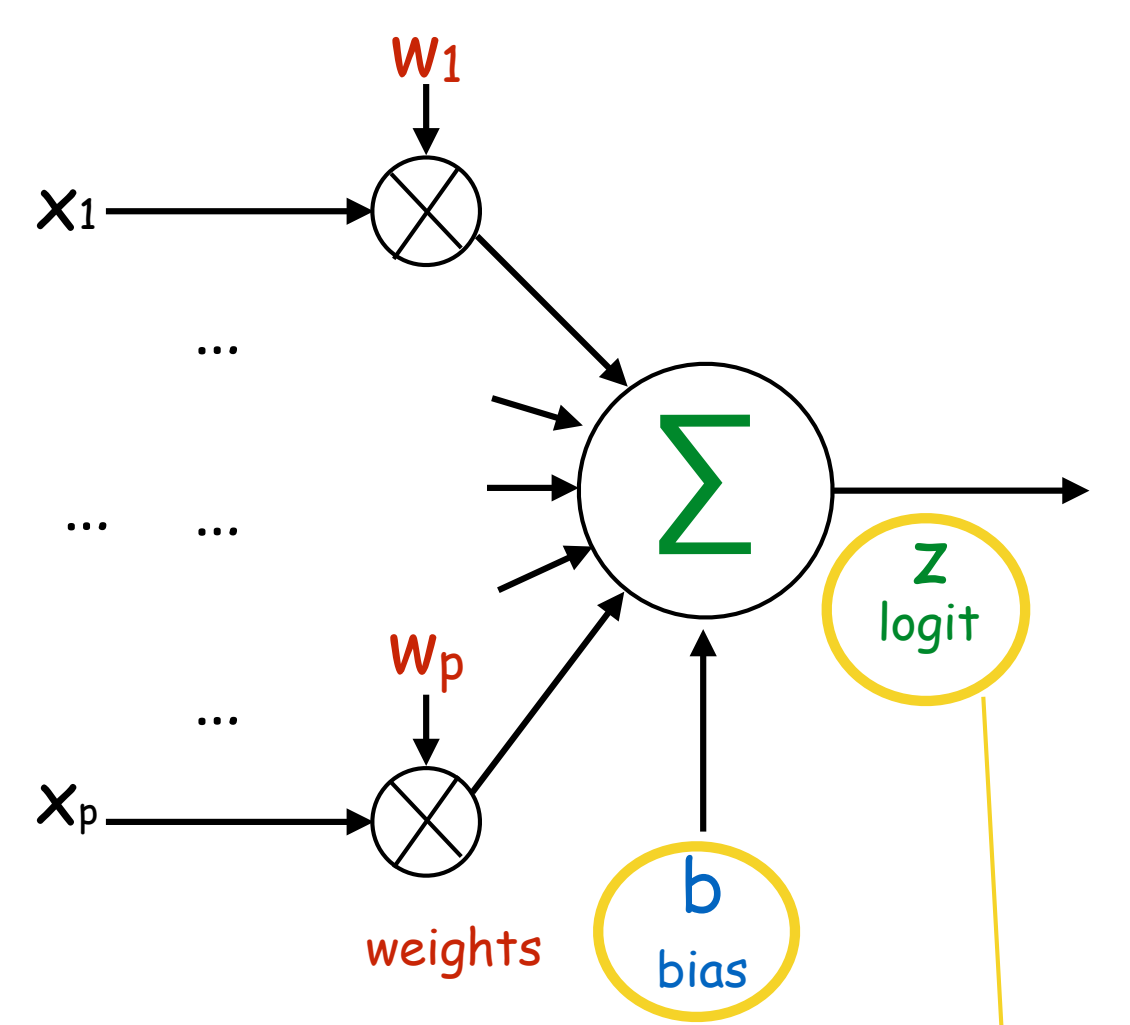
Gradient of L w.r.t. a **vector**?

$$\frac{\partial L}{\partial b}$$

Gradient of L w.r.t. a **matrix**?

$$\frac{\partial L}{\partial W}$$

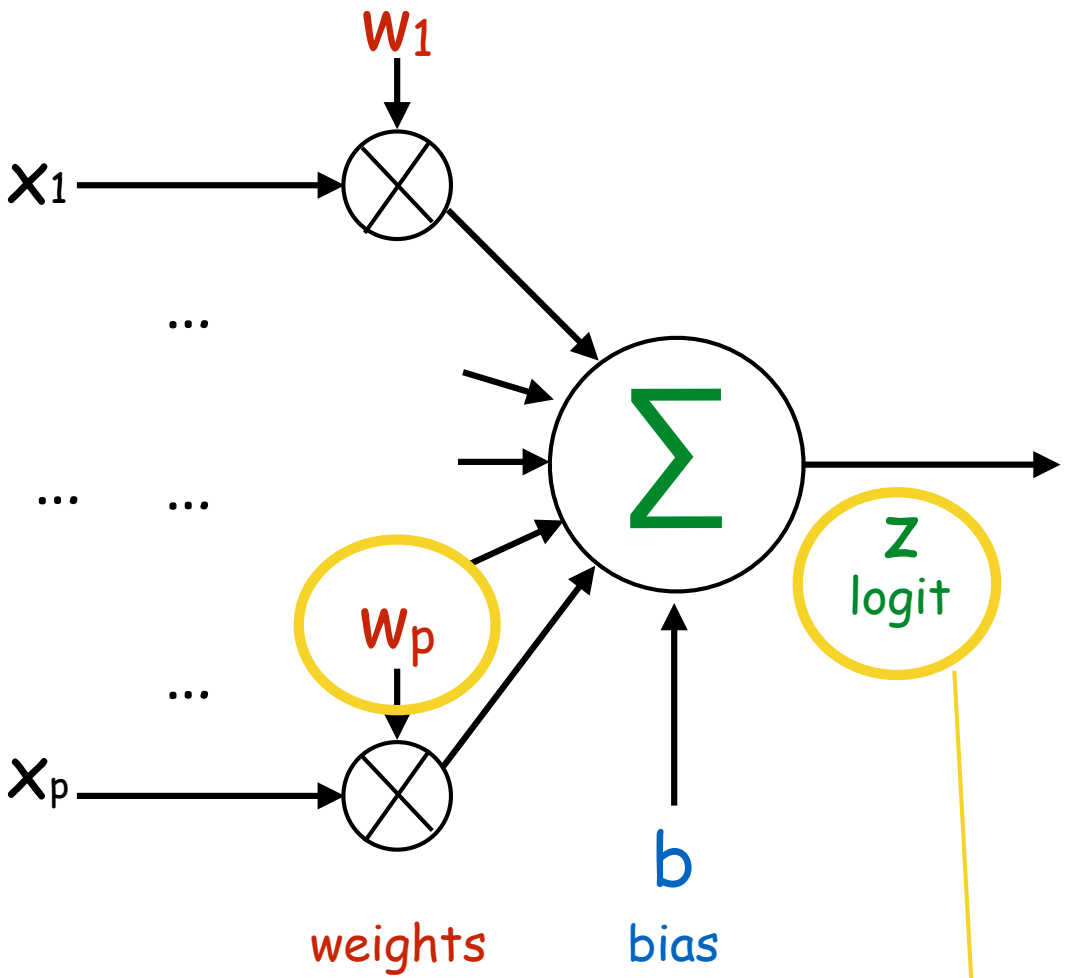
Example



gradient

$$\frac{\partial z}{\partial b} = 1$$

Example

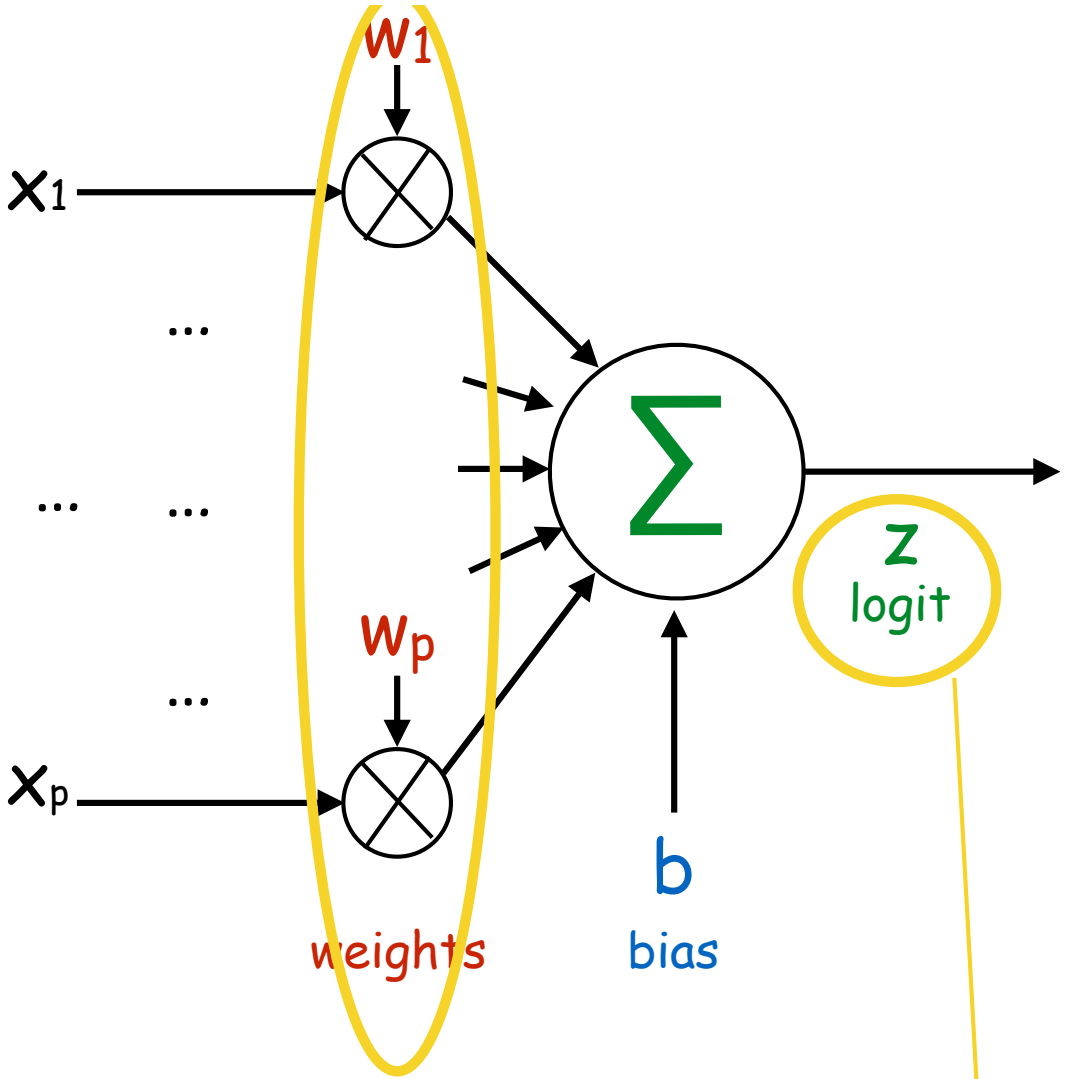


gradient

$$\frac{\partial z}{\partial w_p} = x_p$$

is a function of x_p

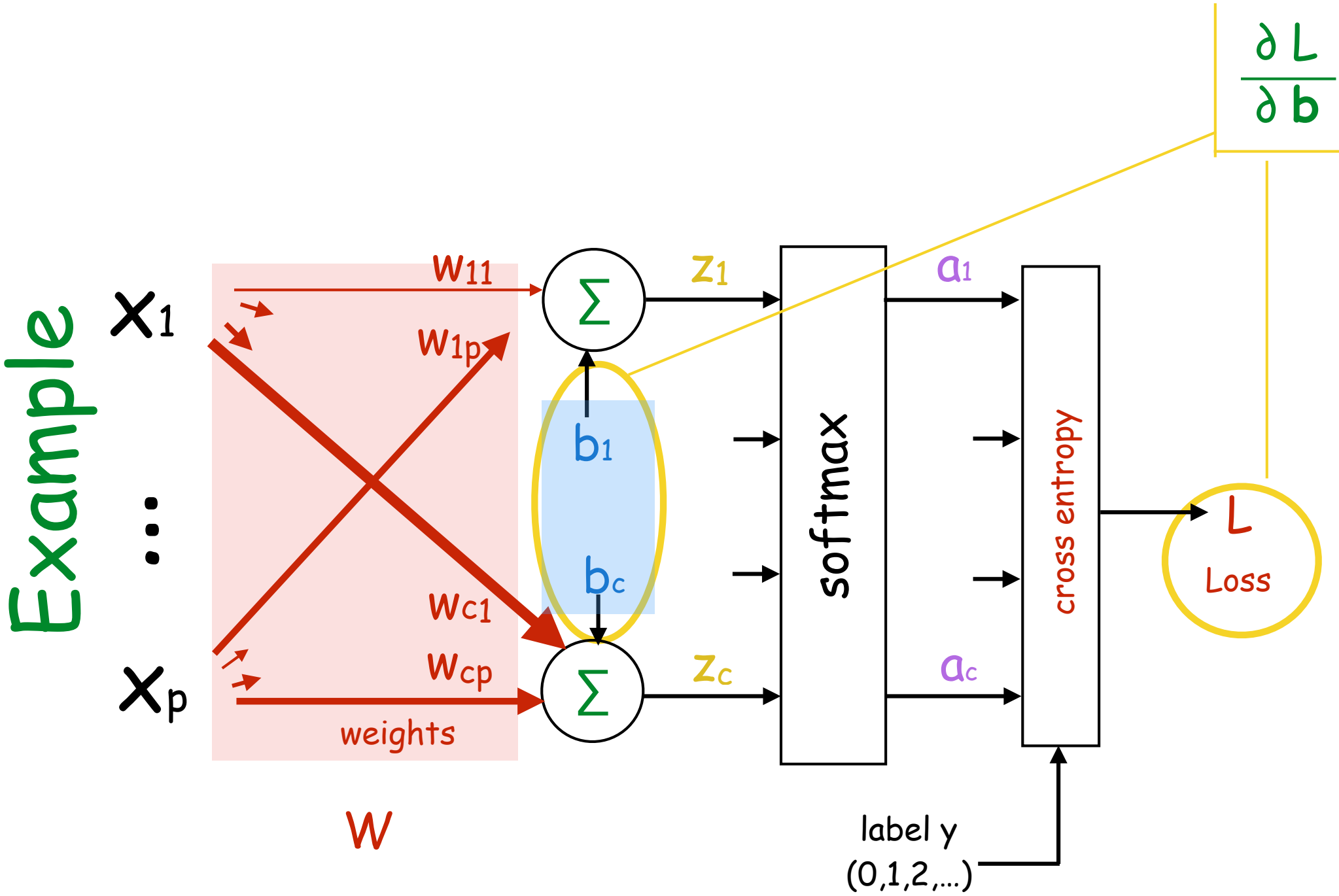
Example

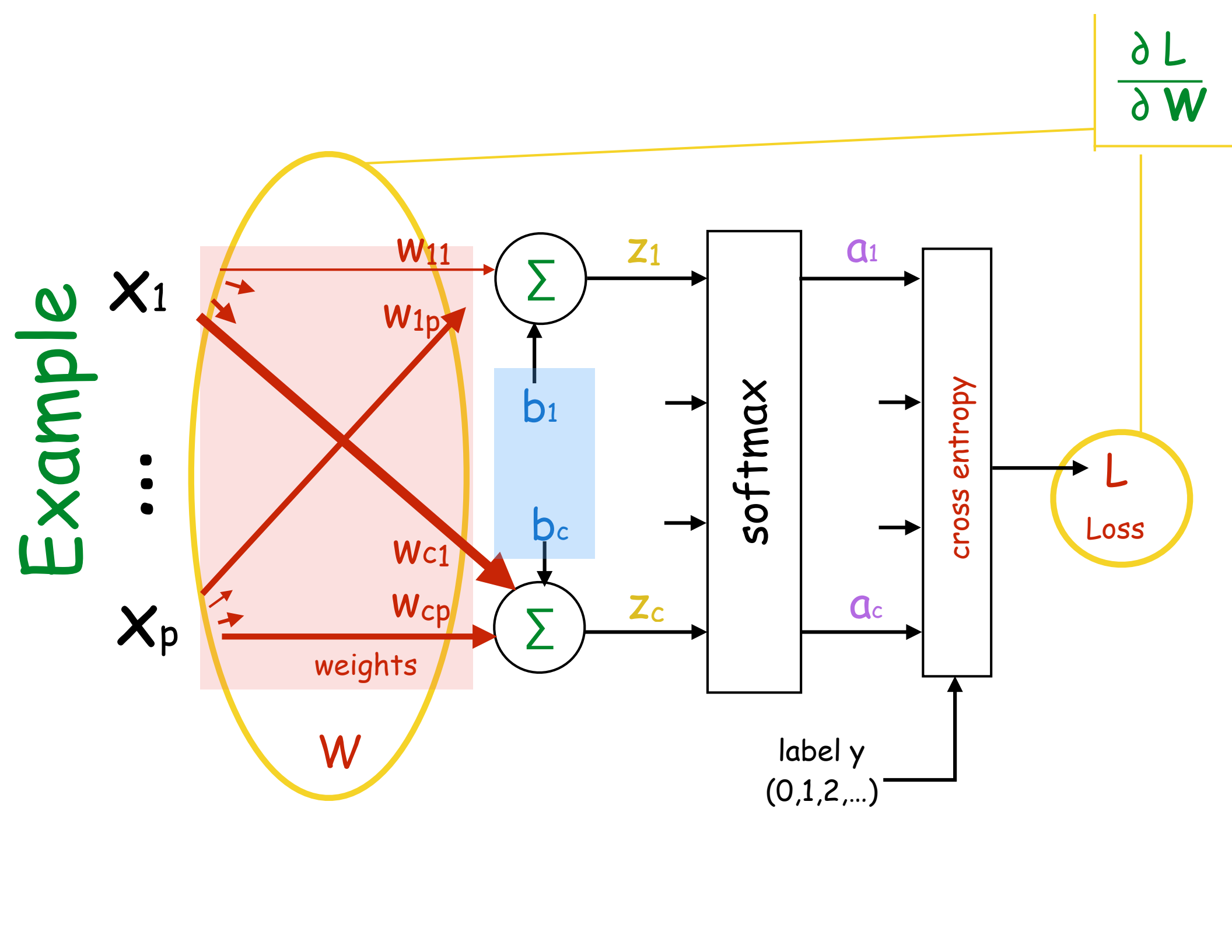


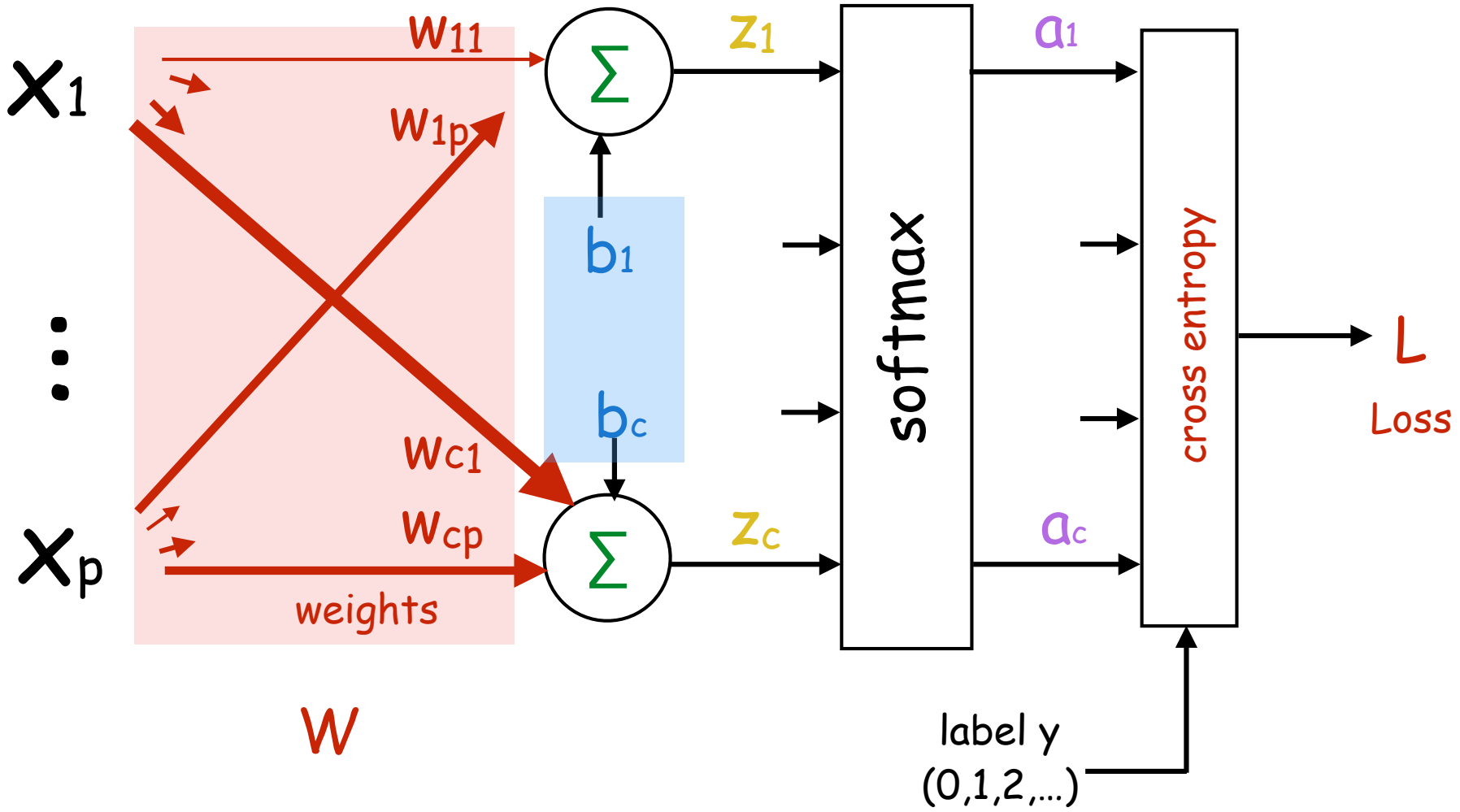
gradient

$$\frac{\partial z}{\partial \mathbf{w}} = \left(\frac{\partial z}{\partial w_1}, \frac{\partial z}{\partial w_2}, \cdots, \frac{\partial z}{\partial w_p}\right)$$

$$= \left(x_1 , x_2 , \cdots, x_p \right) = \mathbf{x}$$


Example
x_1
x_p
w_{11}
w_{1p}
w_{c1}
w_{cp}
weights
W
b_1
b_c
z_1
z_c
softmax
a_1
a_c
cross entropy
L
Loss
label y
(0,1,2,...)
$\frac{\partial L}{\partial b}$

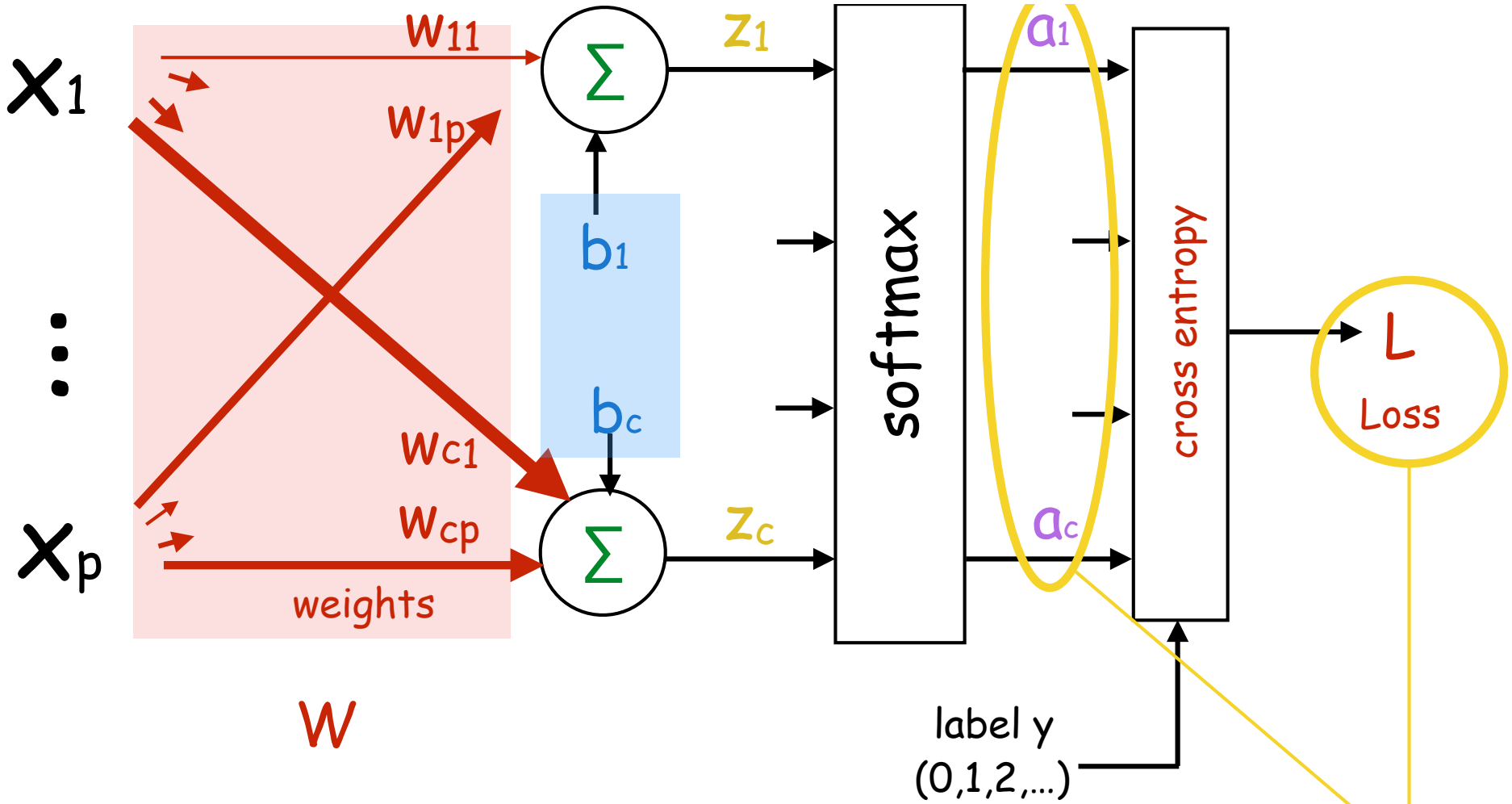

Example
$\frac{\partial L}{\partial W}$
x_1
x_p
w_{11}
w_{1p}
w_{c1}
w_{cp}
weights
W
b_1
b_c
z_1
z_c
softmax
a_1
a_c
cross entropy
L
Loss
label y
(0,1,2,...)



$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial W}$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial b}$$

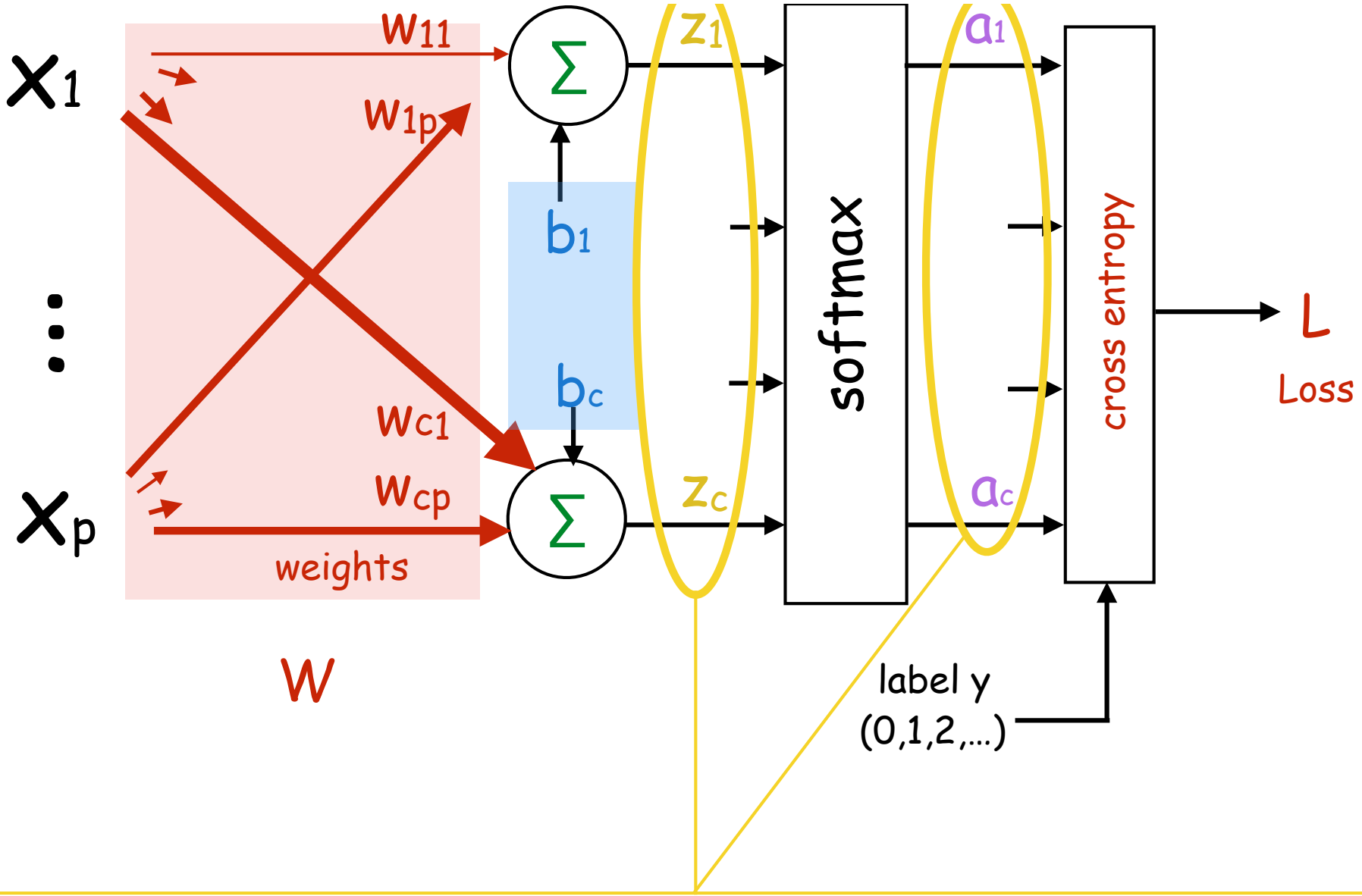
Chain Rule



$$\frac{\partial L}{\partial a} = \left(\frac{\partial L}{\partial a_1}, \ldots, \frac{\partial L}{\partial a_y}, \ldots, \frac{\partial L}{\partial a_c}\right)$$

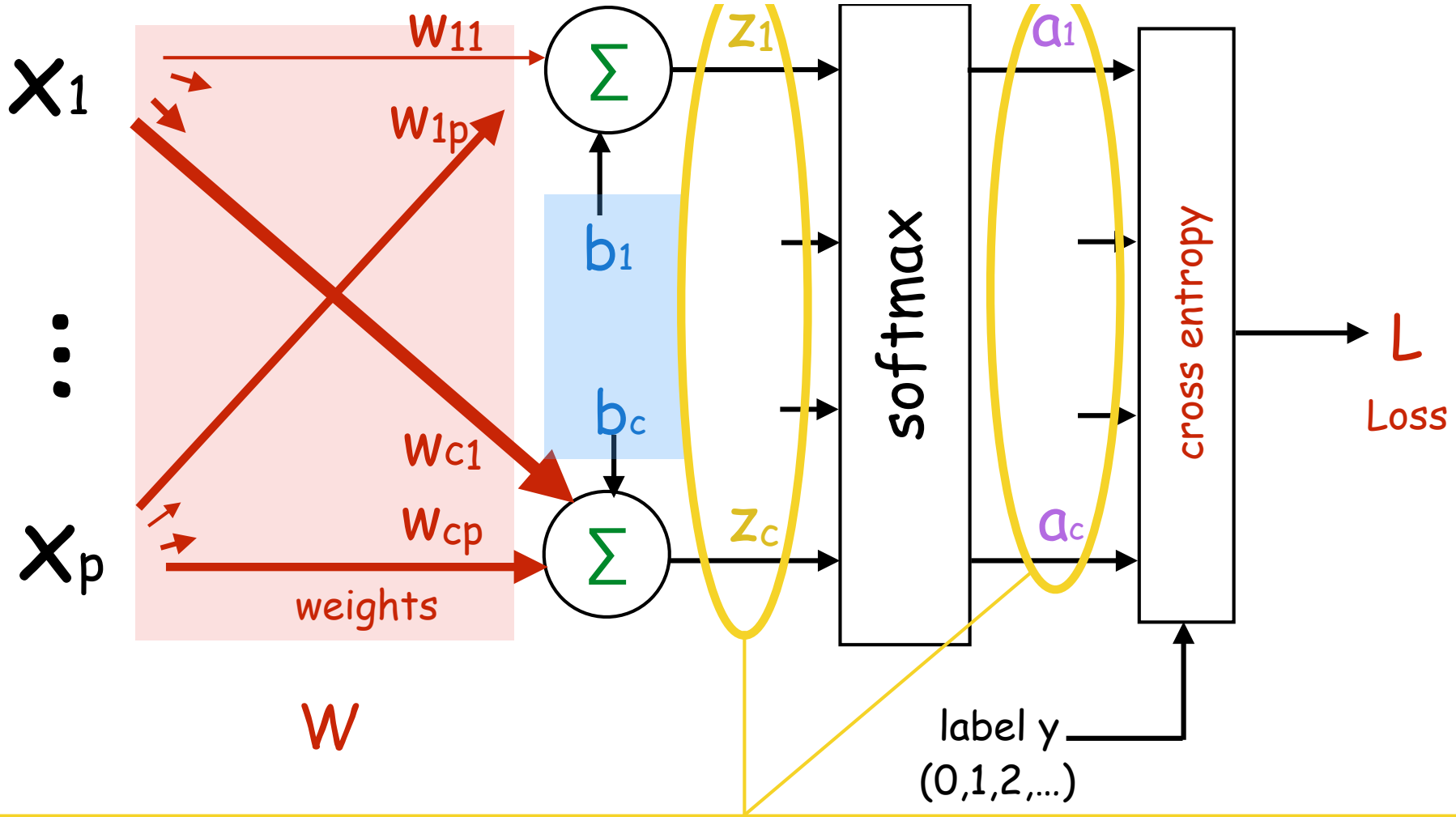
$$= \left(0, \ldots, -\frac{1}{a_y}, \ldots, 0\right)$$

y



$$\frac{\partial \mathbf{a}}{\partial \mathbf{z}} = ?$$

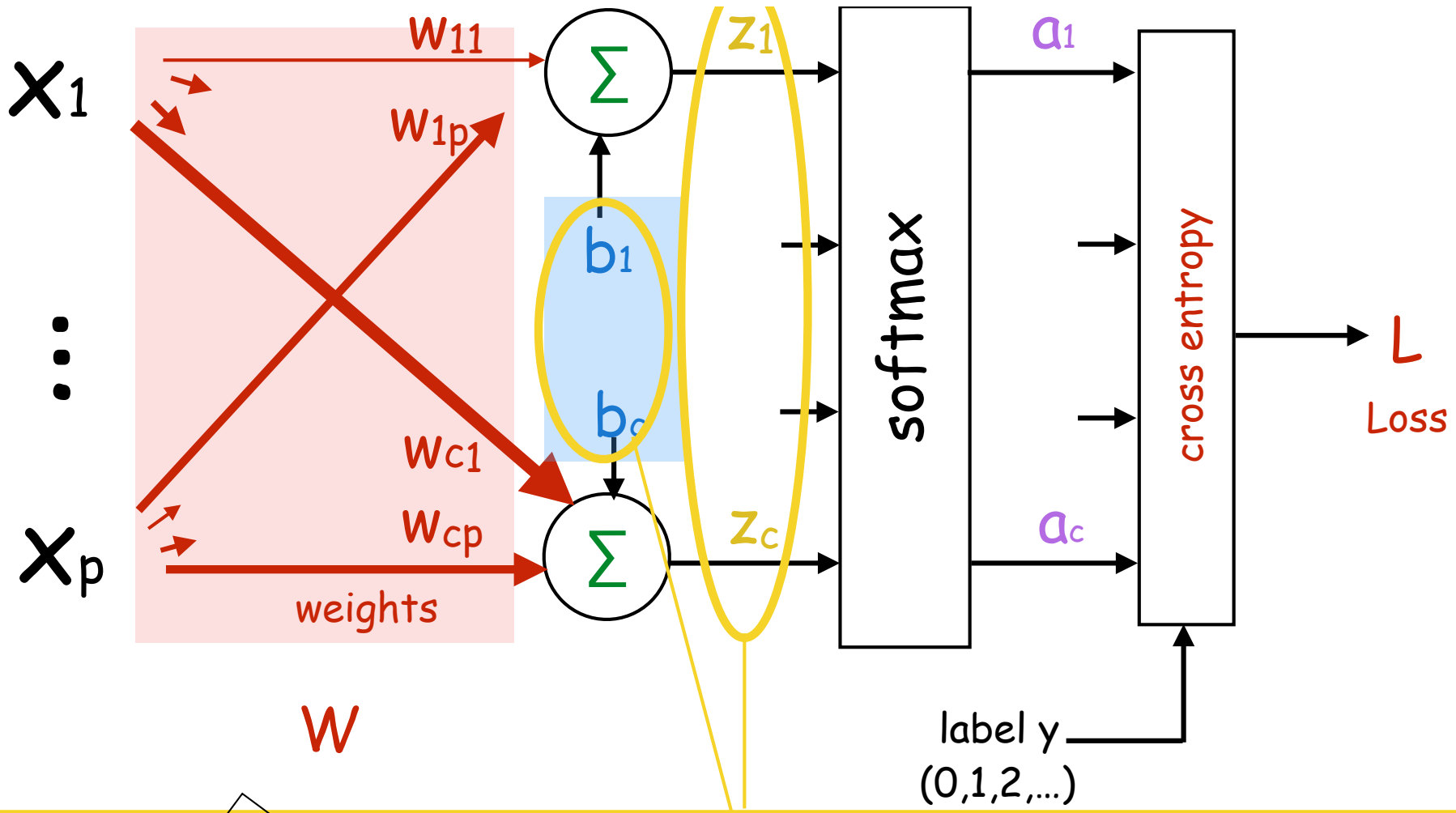
Gradient of a vector w.r.t. a vector !!!



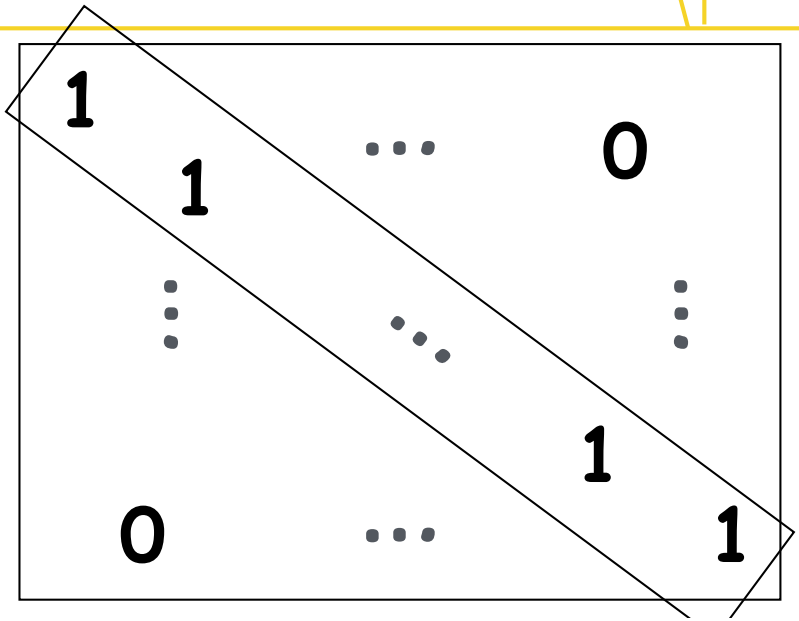
$$\frac{\partial a}{\partial z} = \begin{bmatrix} \frac{\partial a_1}{\partial z_1} & \cdots & \frac{\partial a_1}{\partial z_c} \\ \vdots & & \vdots \\ \frac{\partial a_c}{\partial z_1} & & \frac{\partial a_c}{\partial z_c} \end{bmatrix}$$

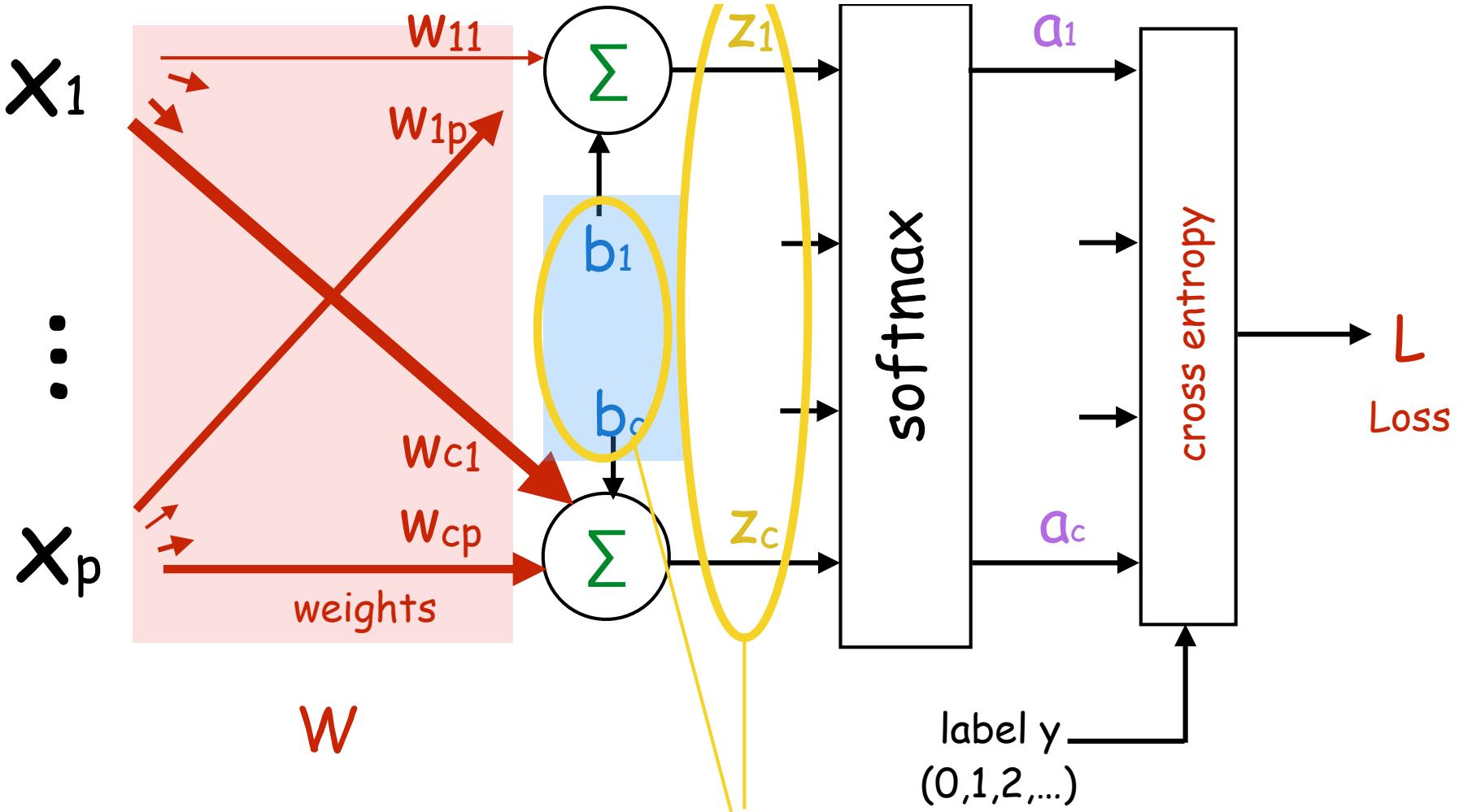
$$\frac{\partial a_i}{\partial z_j} = \begin{cases} a_i (1 - a_i) & \text{if } i=j \\ -a_i a_j & \text{if } i \neq j \end{cases}$$

https://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/



$$\frac{\partial z}{\partial b} =$$



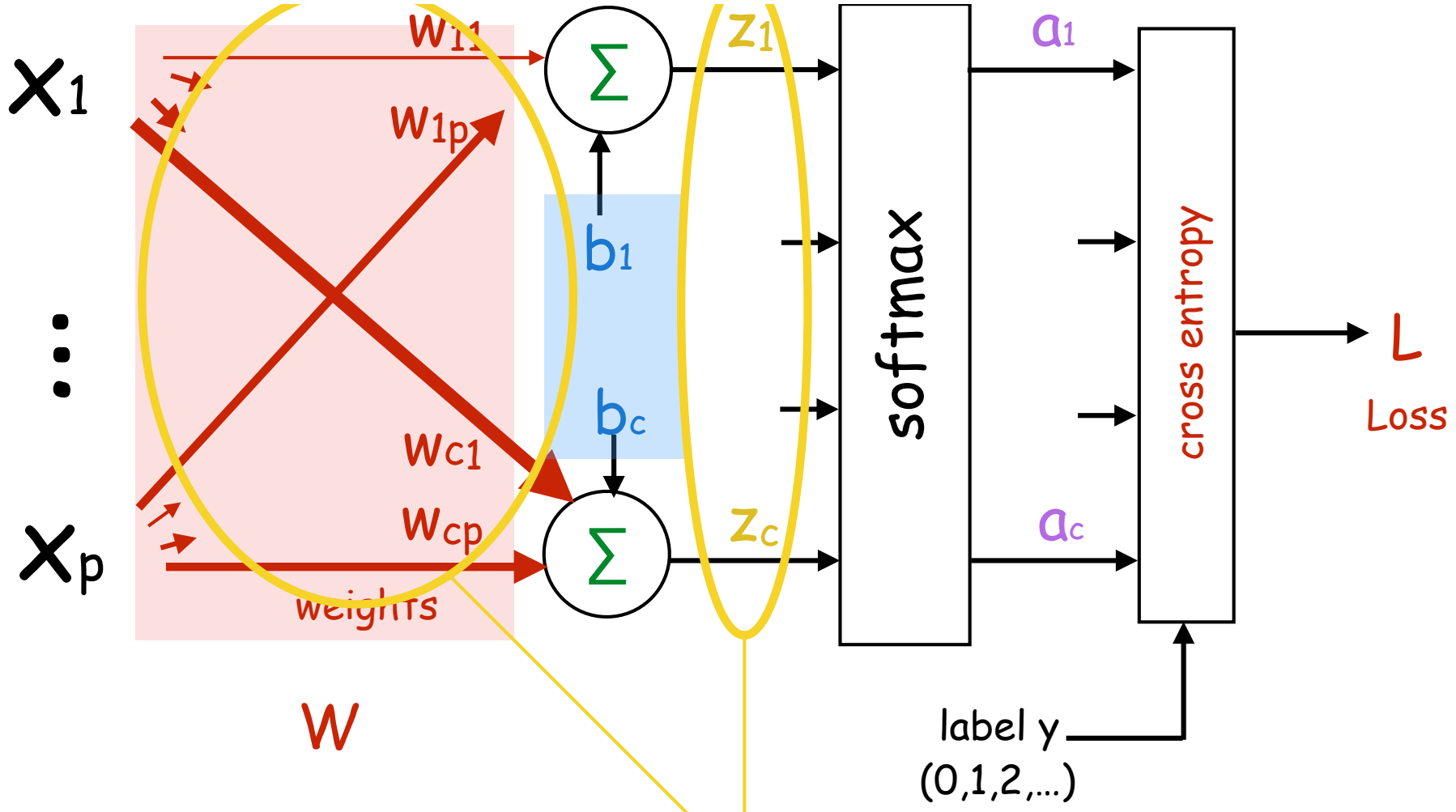


$$\frac{\partial z}{\partial b}$$

$$\frac{\partial z_1}{\partial b_1} = 1$$

$$\frac{\partial z_i}{\partial b_i} = 1$$

$$\frac{\partial z_c}{\partial b_c} = 1$$



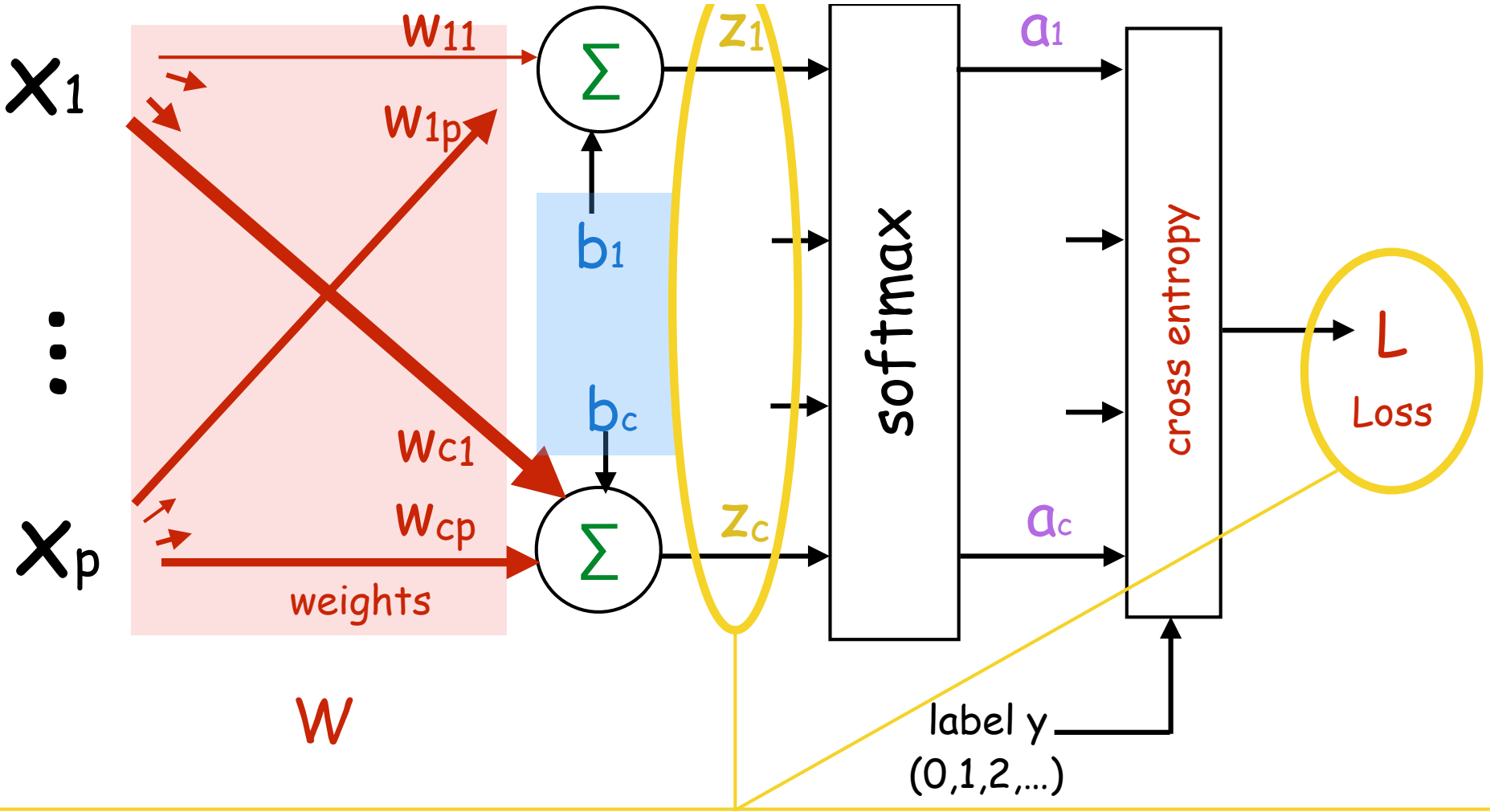
$$\frac{\partial z}{\partial W}$$

$$\frac{\partial z_1}{\partial w_{1*}} = \frac{\partial z_1}{\partial w_{11}}, \cdots, \frac{\partial z_1}{\partial w_{1p}}$$

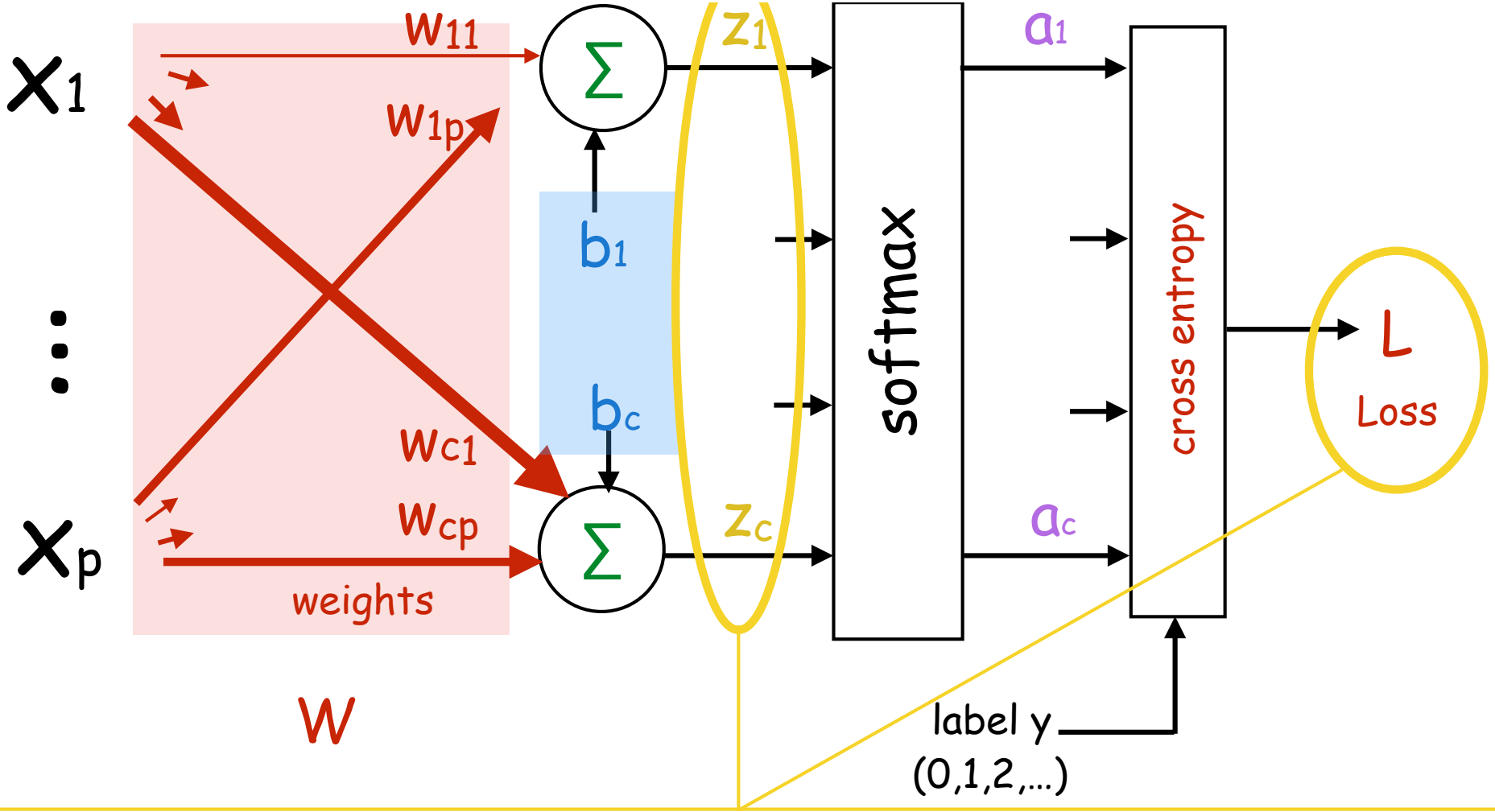
$$\frac{\partial z_i}{\partial w_{i*}} = \frac{\partial z_i}{\partial w_{i1}}, \cdots, \frac{\partial z_i}{\partial w_{ip}}$$

$$\frac{\partial z_c}{\partial w_{c*}} = \frac{\partial z_c}{\partial w_{c1}}, \cdots, \frac{\partial z_c}{\partial w_{cp}}$$

$$= \begin{matrix} x_1 & x_2 & \cdots & x_p \\ x_1 & x_2 & \cdots & x_p \\ x_1 & x_2 & \cdots & x_p \end{matrix}$$



$$\frac{\partial L}{\partial z} = \left(\frac{\partial L}{\partial z_1}, \ldots, \frac{\partial L}{\partial z_i}, \ldots, \frac{\partial L}{\partial z_c}\right) = ?$$



Chain Rule

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial a} \times \frac{\partial a}{\partial z}$$

vector (1 by c) vector (1 by c) matrix (c by c)

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial a} \times \frac{\partial a}{\partial z}$$

vector (1 by c) vector (1 by c) matrix (c by c)

$$\left(\frac{\partial L}{\partial z_1}, \ldots, \frac{\partial L}{\partial z_i}, \ldots, \frac{\partial L}{\partial z_c}\right) =$$

$$\left(\frac{\partial L}{\partial a_1}, \ldots, \frac{\partial L}{\partial a_i}, \ldots, \frac{\partial L}{\partial a_c}\right) \times \begin{bmatrix} \frac{\partial a_1}{\partial z_1} & \cdots & \frac{\partial a_1}{\partial z_c} \\ \vdots & & \vdots \\ \frac{\partial a_c}{\partial z_1} & & \frac{\partial a_c}{\partial z_c} \end{bmatrix}$$

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial a} \times \frac{\partial a}{\partial z}$$

vector (1 by c) vector (1 by c) matrix (c by c)

$$\left(\frac{\partial L}{\partial z_1}, \ldots, \frac{\partial L}{\partial z_i}, \ldots, \frac{\partial L}{\partial z_c}\right) =$$

$$\left(\frac{\partial L}{\partial a_1}, \ldots, \frac{\partial L}{\partial a_i}, \ldots, \frac{\partial L}{\partial a_c}\right) \times \begin{bmatrix} \frac{\partial a_1}{\partial z_1} & \cdots & \frac{\partial a_1}{\partial z_c} \\ \vdots & & \vdots \\ \frac{\partial a_c}{\partial z_1} & & \frac{\partial a_c}{\partial z_c} \end{bmatrix}$$

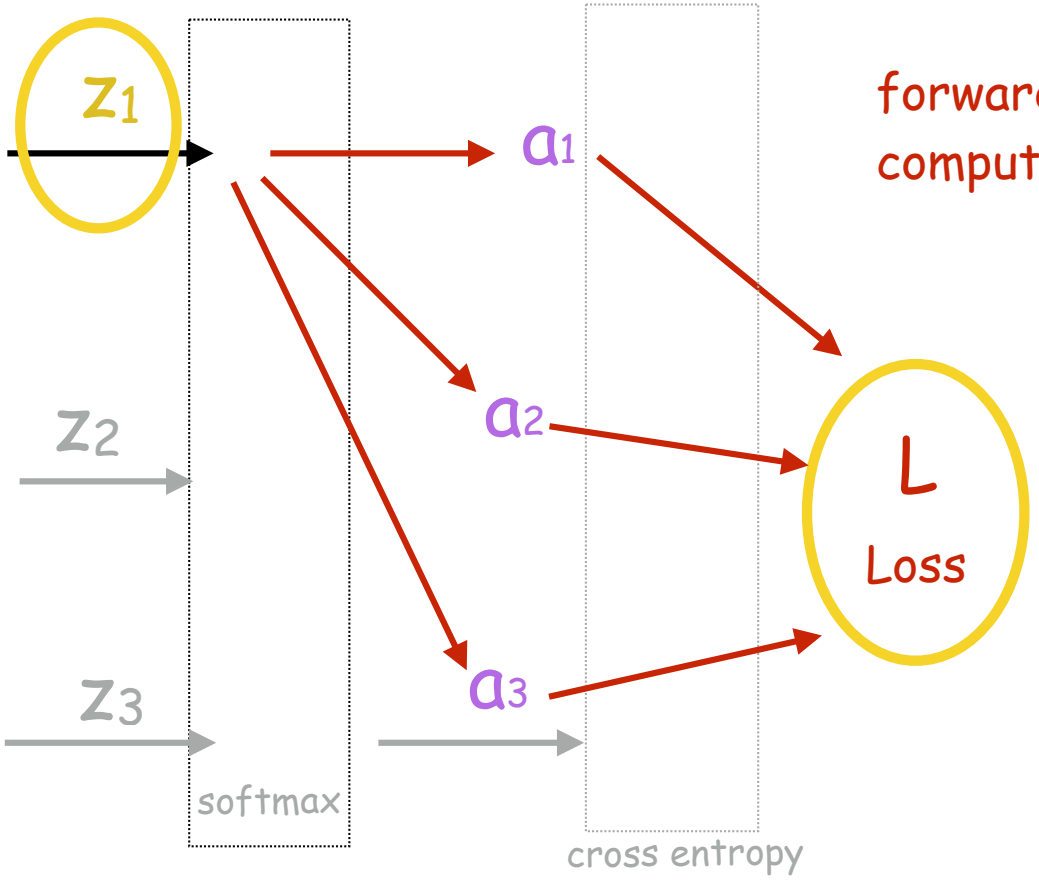
$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial a} \times \frac{\partial a}{\partial z}$$

vector (1 by c) vector (1 by c) matrix (c by c)

$$\left(\frac{\partial L}{\partial z_1}, \ldots, \frac{\partial L}{\partial z_i}, \ldots, \frac{\partial L}{\partial z_c}\right) =$$

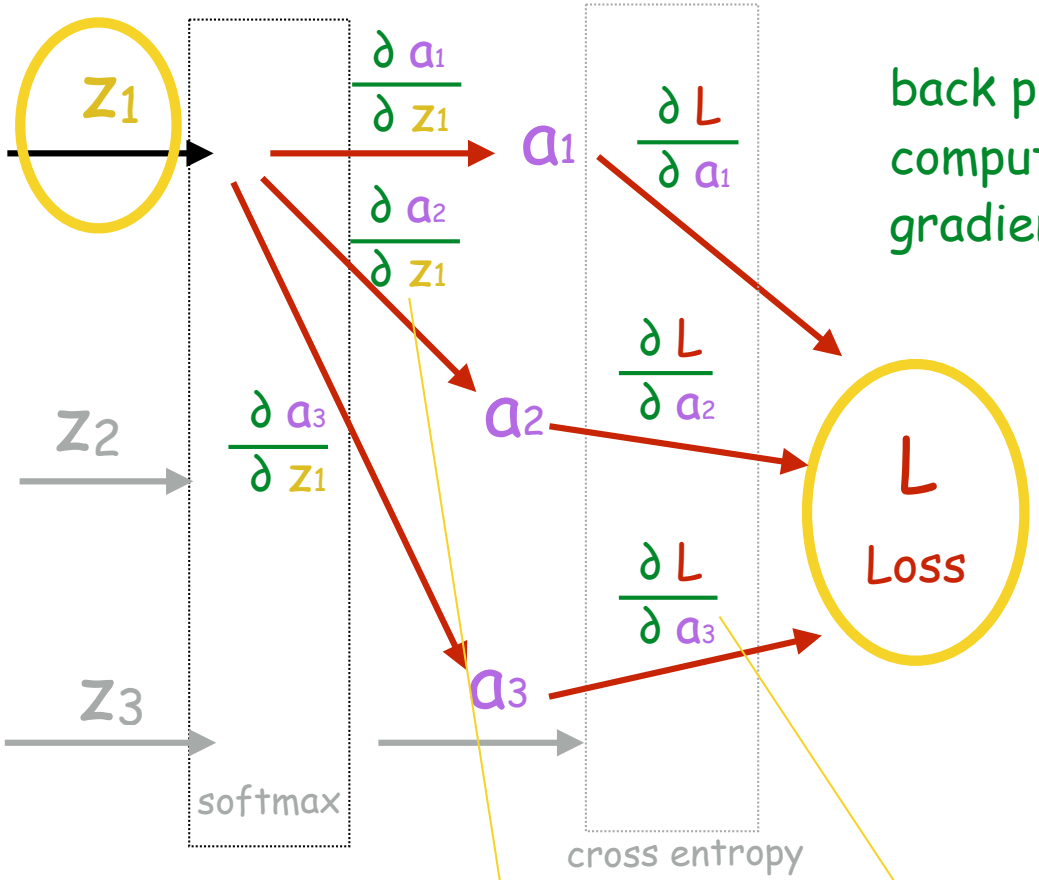
$$\left(\frac{\partial L}{\partial a_1}, \ldots, \frac{\partial L}{\partial a_i}, \ldots, \frac{\partial L}{\partial a_c}\right) \times \begin{bmatrix} \frac{\partial a_1}{\partial z_1} & \cdots & \frac{\partial a_1}{\partial z_c} \\ \vdots & & \vdots \\ \frac{\partial a_c}{\partial z_1} & & \frac{\partial a_c}{\partial z_c} \end{bmatrix}$$

$\frac{\partial L}{\partial z_1}$



$$\left(\frac{\partial L}{\partial a_1}, \ldots, \frac{\partial L}{\partial a_i}, \ldots, \frac{\partial L}{\partial a_c} \right) \times \begin{bmatrix} \frac{\partial a_1}{\partial z_1} & \cdots & \frac{\partial a_1}{\partial z_c} \\ \vdots & & \vdots \\ \frac{\partial a_c}{\partial z_1} & & \frac{\partial a_c}{\partial z_c} \end{bmatrix}$$

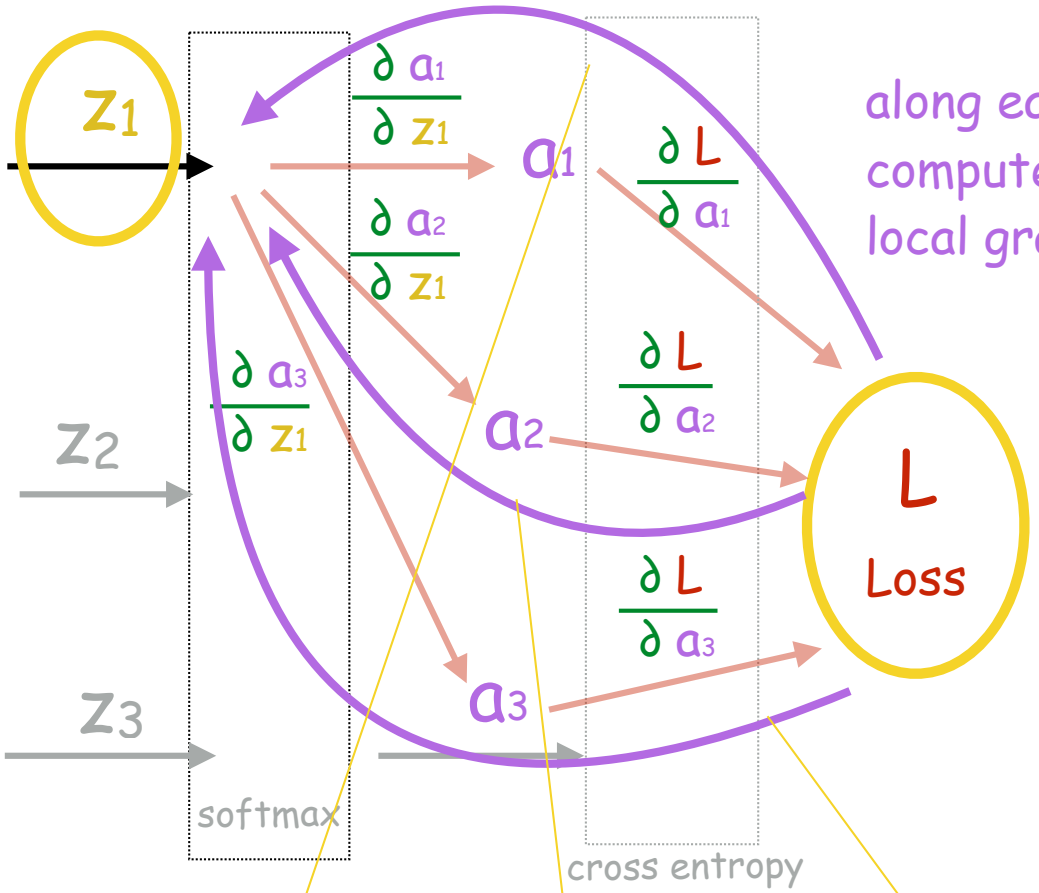
$\frac{\partial L}{\partial z_1}$



$$\frac{\partial L}{\partial a} = \left(\frac{\partial L}{\partial a_1} , \frac{\partial L}{\partial a_2} , \frac{\partial L}{\partial a_3} \right)$$

$$\frac{\partial a}{\partial z_1} = \left(\frac{\partial a_1}{\partial z_1} , \frac{\partial a_2}{\partial z_1} , \frac{\partial a_3}{\partial z_1} \right)$$

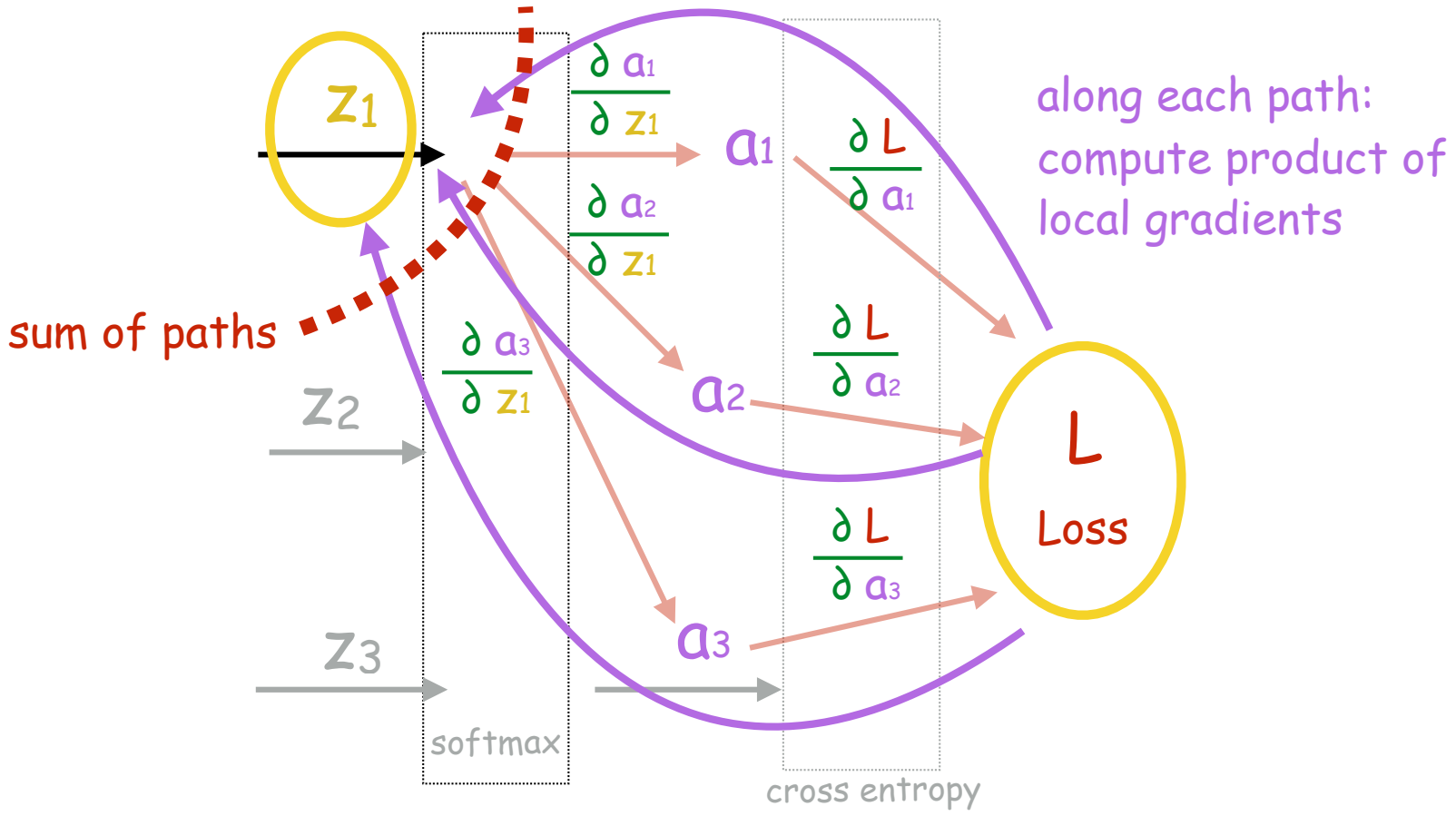
$\frac{\partial L}{\partial z_1}$



$$\frac{\partial L}{\partial a} = \left(\frac{\partial L}{\partial a_1}, \frac{\partial L}{\partial a_2}, \frac{\partial L}{\partial a_3}\right) \times$$

$$\frac{\partial a}{\partial z_1} = \left(\frac{\partial a_1}{\partial z_1}, \frac{\partial a_2}{\partial z_1}, \frac{\partial a_3}{\partial z_1}\right)$$

$\frac{\partial L}{\partial z_1}$



product along each path

sum of paths

$$\frac{\partial L}{\partial z_1} = \left(\frac{\partial L}{\partial a_1} \times \frac{\partial a_1}{\partial z_1}\right) + \left(\frac{\partial L}{\partial a_2} \times \frac{\partial a_2}{\partial z_1}\right) + \left(\frac{\partial L}{\partial a_3} \times \frac{\partial a_3}{\partial z_1}\right)$$

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial z}^{T} \times \frac{\partial z}{\partial W}$$

matrix (c by p) vector (c by 1) matrix (c by p)

$$\left(\frac{\partial L}{\partial z_1}, \ldots, \frac{\partial L}{\partial z_i}, \ldots, \frac{\partial L}{\partial z_c}\right)^{T} \times$$

transpose

element-wise product

$$\begin{bmatrix} \frac{\partial z_1}{\partial w_{11}} & \cdots & \frac{\partial z_1}{\partial w_{1p}} \\ \frac{\partial z_i}{\partial w_{i1}} & \cdots & \frac{\partial z_i}{\partial w_{ip}} \\ \frac{\partial z_c}{\partial w_{c1}} & \cdots & \frac{\partial z_c}{\partial w_{cp}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial w_{11}} & \cdots & \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial w_{1p}} \\ \frac{\partial L}{\partial z_i}\frac{\partial z_i}{\partial w_{i1}} & \cdots & \frac{\partial L}{\partial z_i}\frac{\partial z_i}{\partial w_{ip}} \\ \frac{\partial L}{\partial z_c}\frac{\partial z_c}{\partial w_{c1}} & \cdots & \frac{\partial L}{\partial z_c}\frac{\partial z_c}{\partial w_{cp}} \end{bmatrix}$$

Softmax Regression (train)

initialize **W** and **b**

Loop for n_epoch iterations:

Loop for each training instance (**x**, y) in training set

forward pass to compute **z**, **a** and L for the instance

backward pass to compute local gradients

$$\frac{\partial L}{\partial a}\ \frac{\partial a}{\partial z}\ \frac{\partial z}{\partial b}\ \frac{\partial z}{\partial W}$$

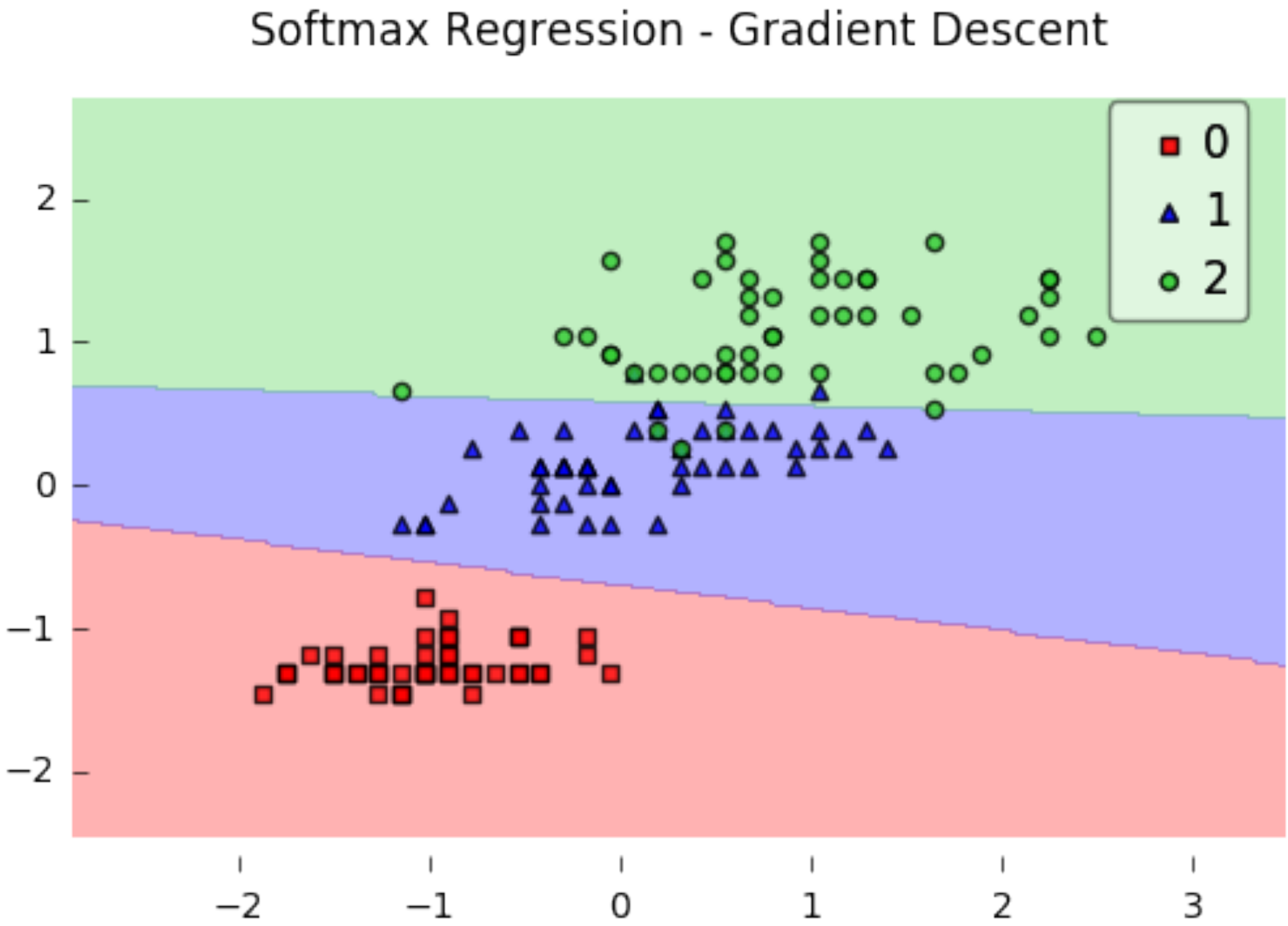
compute global gradients using chain rule

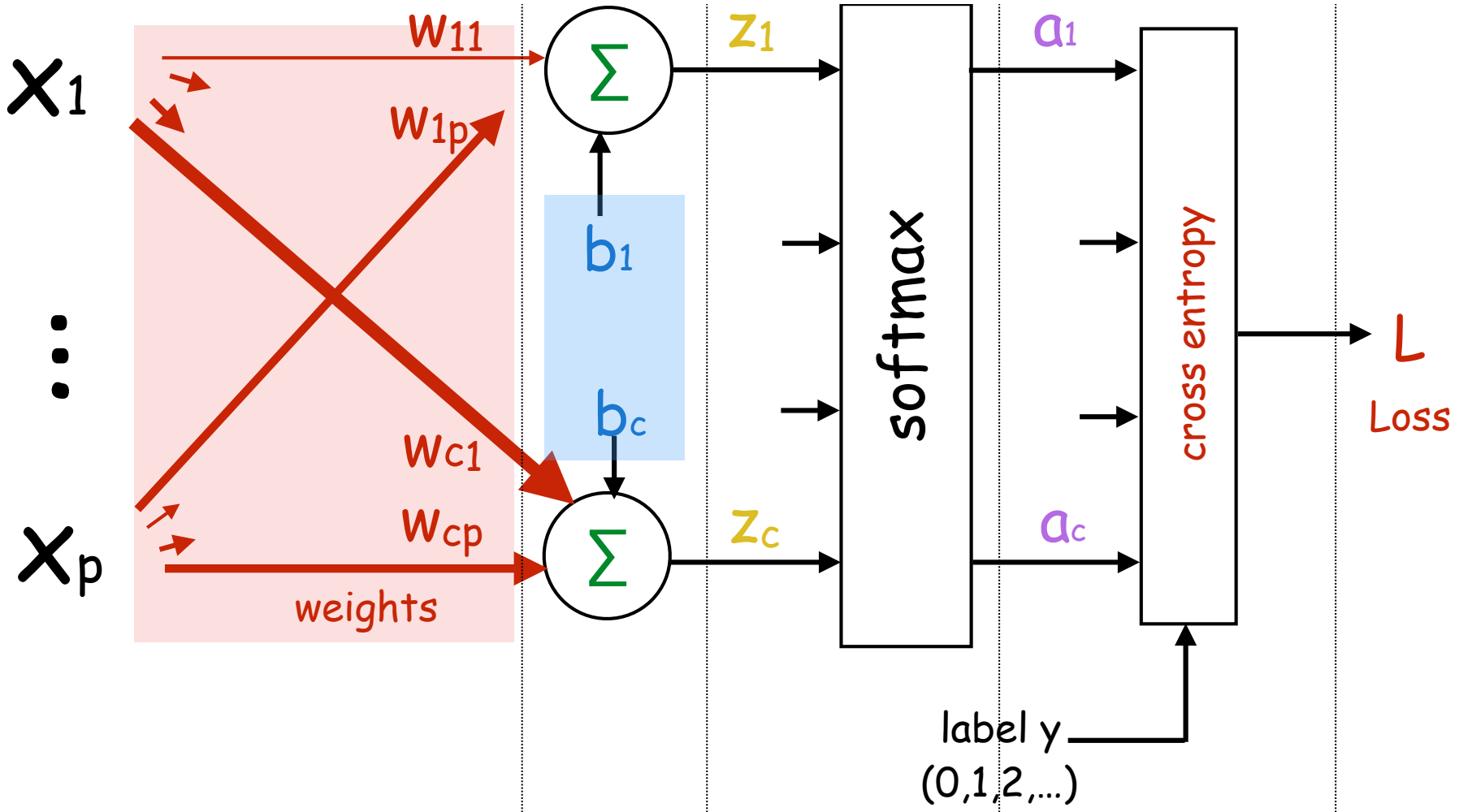
$$\frac{\partial L}{\partial W}\ \frac{\partial L}{\partial b}$$

update the parameters **W** and **b**

$$W \leftarrow W - a\frac{\partial L}{\partial W}$$

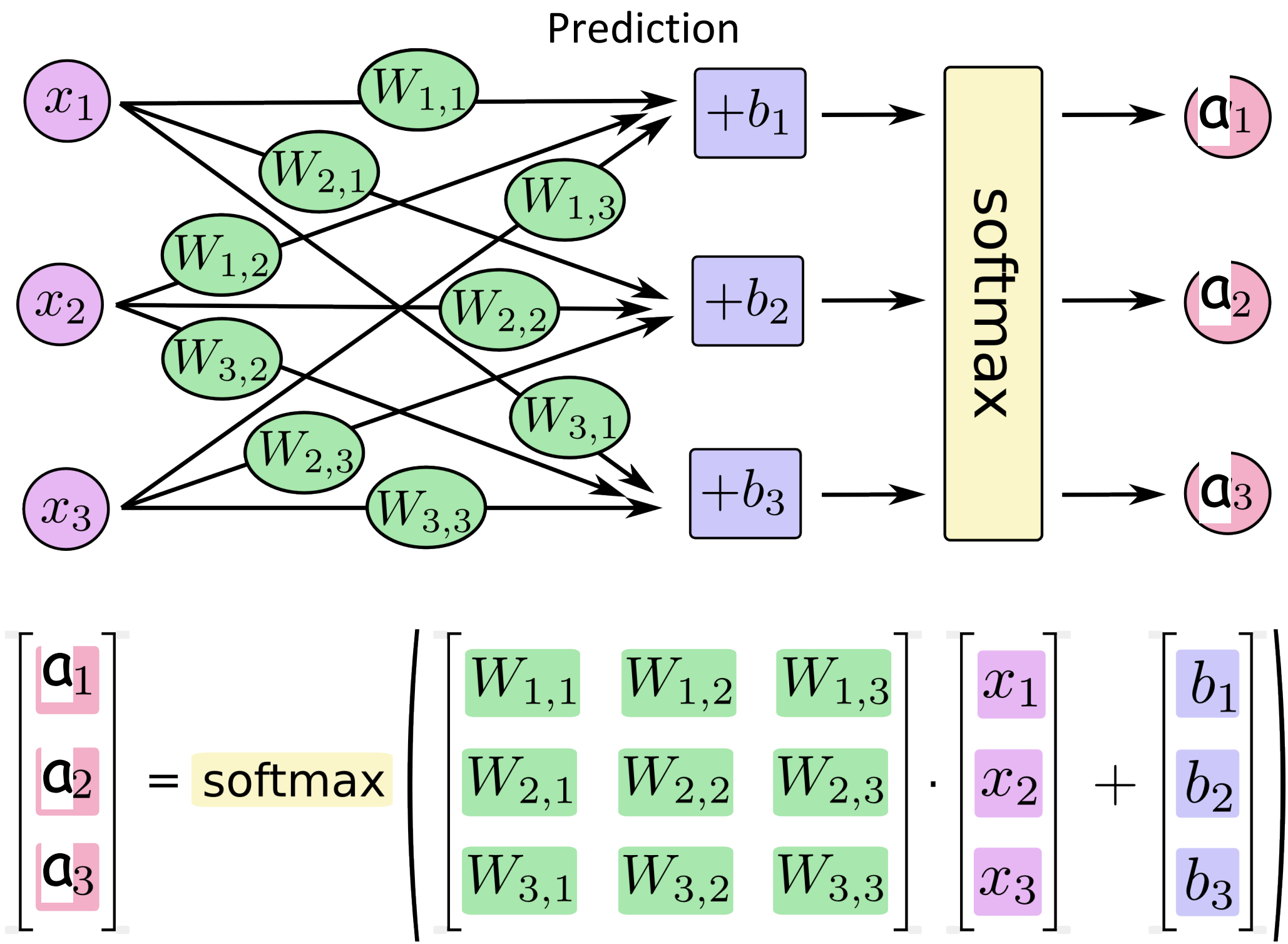
$$b \leftarrow b - a\frac{\partial L}{\partial b}$$


Softmax Regression - Gradient Descent
0
1
2
2
1
0
−1
−2
−2
−1
0
1
2
3



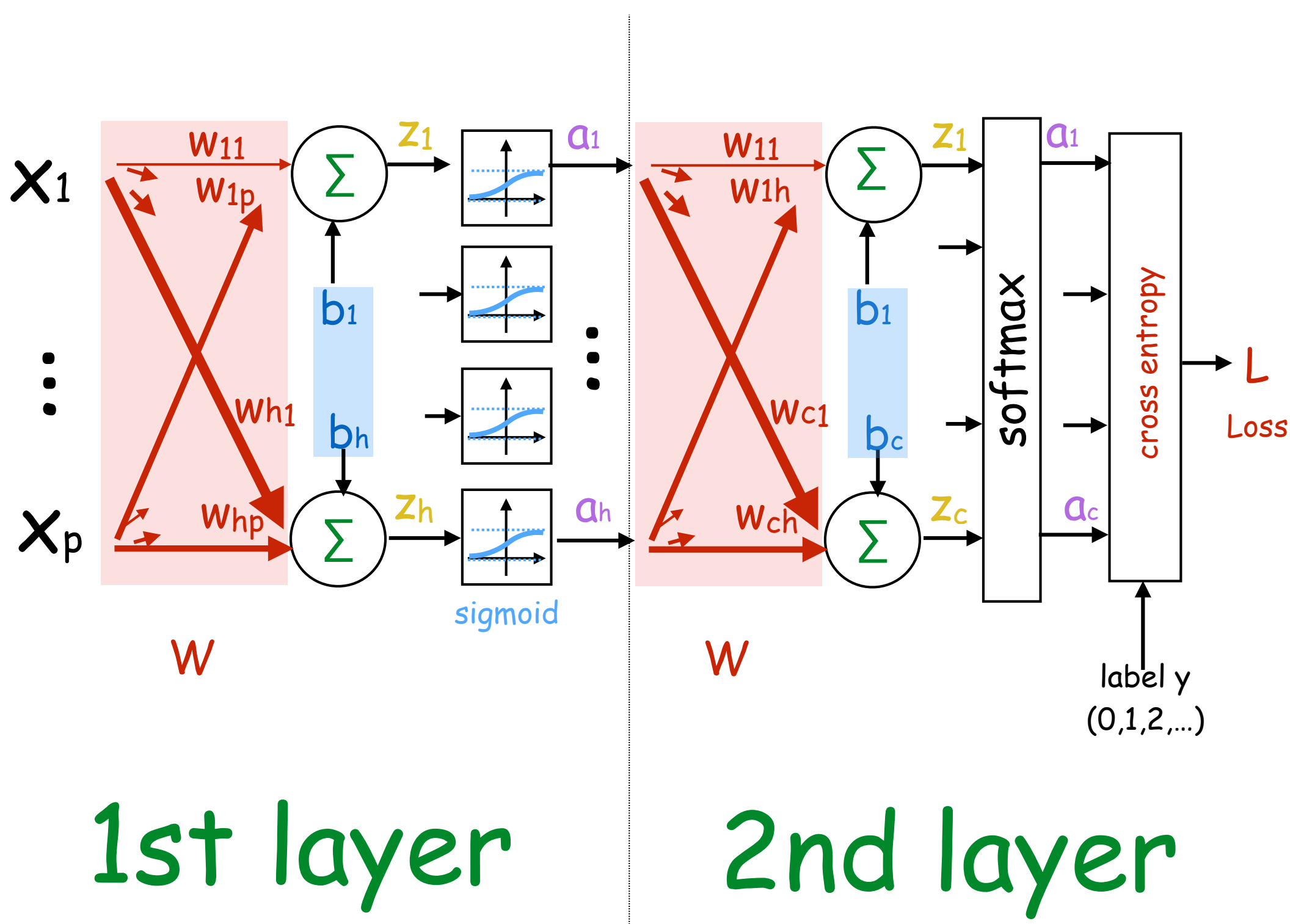
weights	biases	logits	activations	loss
W	**b**	**z**	**a**	**L**

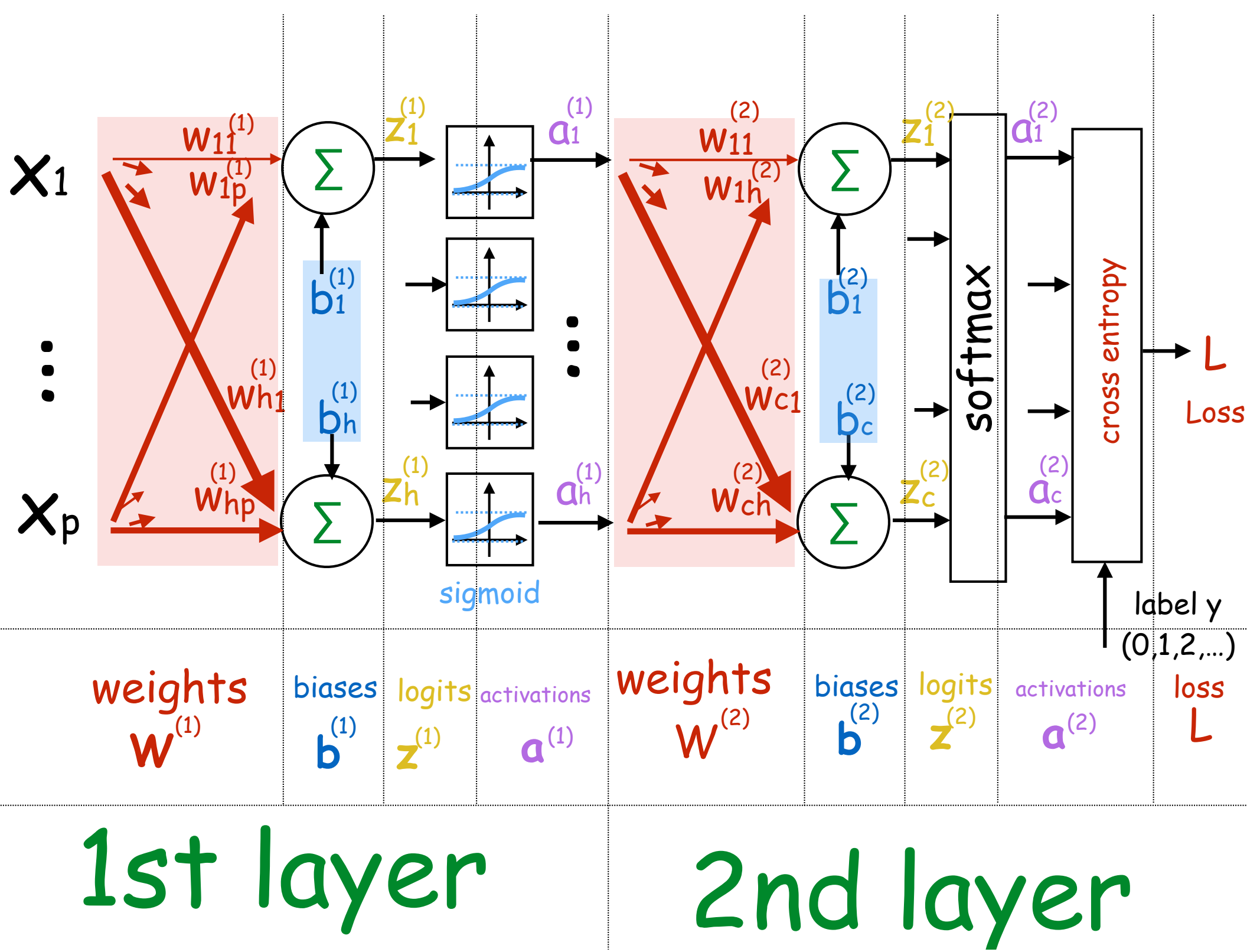
Softmax Regression



$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

Fully Connected Neural Network


x1
xp
w11
w1p
wh1
whp
b1
bh
z1
zh
a1
ah
sigmoid
W
w11
w1h
wc1
wch
b1
bc
z1
zc
a1
ac
W
softmax
cross entropy
L
Loss
label y
(0,1,2,...)
1st layer
2nd layer


x1
xp
w11(1)
w1p(1)
whp(1)
wh1(1)
b1(1)
bh(1)
z1(1)
zh(1)
a1(1)
ah(1)
sigmoid
w11(2)
w1h(2)
wc1(2)
wch(2)
b1(2)
bc(2)
z1(2)
zc(2)
a1(2)
ac(2)
softmax
cross entropy
L
Loss
label y
(0,1,2,...)
weights
w(1)
biases
b(1)
logits
z(1)
activations
a(1)
weights
w(2)
biases
b(2)
logits
z(2)
activations
a(2)
loss
L
1st layer
2nd layer

$$\frac{\partial L}{\partial a^{(1)}} = \frac{\partial L}{\partial z^{(2)}} \times \frac{\partial z^{(2)}}{\partial a^{(1)}}$$

vector length h | vector of length c | matrix (c by h)

$$\left(\frac{\partial L}{\partial a_1^{(1)}}, \ldots, \frac{\partial L}{\partial a_c^{(1)}}\right)$$

$$= \left(\frac{\partial L}{\partial z_1^{(2)}}, \ldots, \frac{\partial L}{\partial z_c^{(2)}}\right) \times \begin{bmatrix} \frac{\partial z_1^{(2)}}{\partial a_1^{(1)}} & \cdots & \frac{\partial z_1^{(2)}}{\partial a_h^{(1)}} \\ \cdots & & \\ \frac{\partial z_c^{(2)}}{\partial a_1^{(1)}} & \cdots & \frac{\partial z_c^{(2)}}{\partial a_h^{(1)}} \end{bmatrix}$$