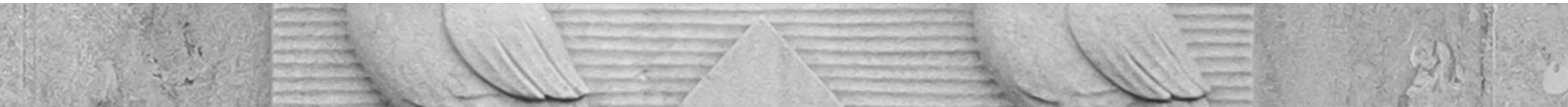


Deep Learning for Vision & Language

SGD and Regularization, Softmax, MLPs

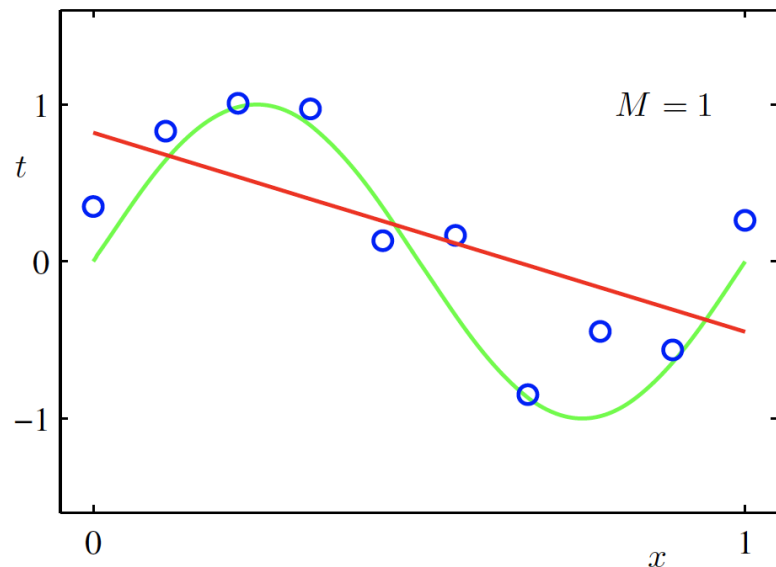


RICE UNIVERSITY



Overfitting

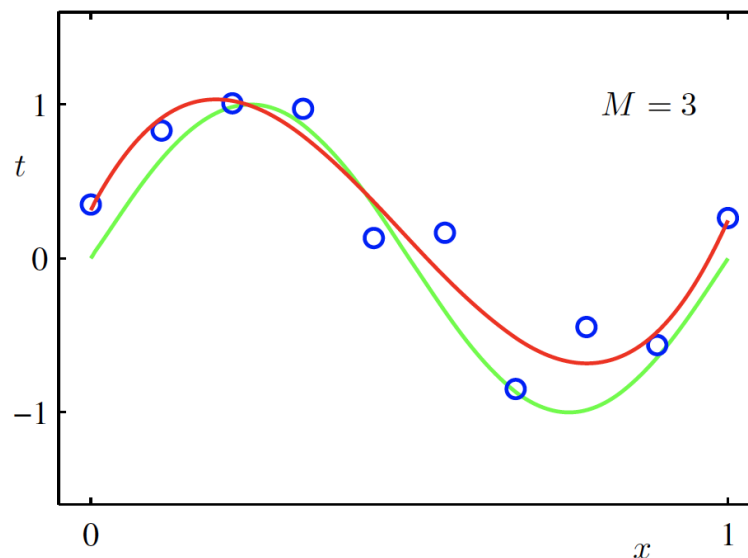
f is linear



$Loss(w)$ is high

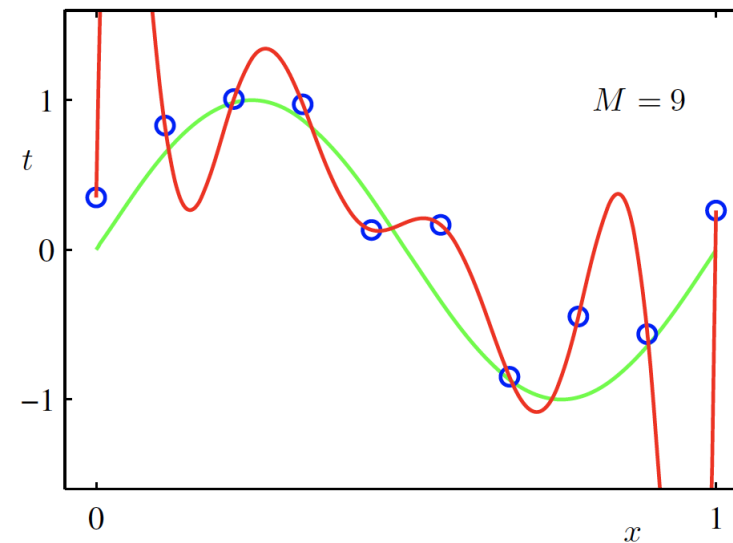
Underfitting

f is cubic



$Loss(w)$ is low

f is a polynomial of degree 9



$Loss(w)$ is zero!

Overfitting

(mini-batch) Stochastic Gradient Descent (SGD)

$\lambda = 0.01$

Initialize w and b randomly

$$l(w, b) = \sum_{i \in B} Cost(w, b)$$

for $e = 0, \text{num_epochs}$ **do**

for $b = 0, \text{num_batches}$ **do**

 Compute: $dl(w, b)/dw$ and $dl(w, b)/db$

 Update w : $w = w - \lambda dl(w, b)/dw$

 Update b : $b = b - \lambda dl(w, b)/db$

 Print: $l(w, b)$ *// Useful to see if this is becoming smaller or not.*

end

end

Regularization

- Large weights lead to large variance. i.e. model fits to the training data too strongly.
- Solution: Minimize the loss but also try to keep the weight values small by doing the following:

$$\text{minimize} \quad L(w, b) + \alpha \sum_i |w_i|^2$$

Regularization

- Large weights lead to large variance. i.e. model fits to the training data too strongly.
- Solution: Minimize the loss but also try to keep the weight values small by doing the following:

minimize $L(w, b) + \alpha \sum_i |w_i|^2$

Regularizer term
e.g. L2- regularizer

SGD with Regularization (L-2)

$$\lambda = 0.01$$

$$l(w, b) = l(w, b) + \alpha \sum_i |w_i|^2$$

Initialize w and b randomly

for $e = 0, \text{num_epochs}$ **do**

for $b = 0, \text{num_batches}$ **do**

Compute: $dl(w, b)/dw$ and $dl(w, b)/db$

Update w : $w = w - \lambda dl(w, b)/dw - \lambda \alpha w$

Update b : $b = b - \lambda dl(w, b)/db$

Print: $l(w, b)$ // Useful to see if this is becoming smaller or not.

end

end

Revisiting Another Problem with SGD

$$\lambda = 0.01$$

$$l(w, b) = l(w, b) + \alpha \sum_i |w_i|^2$$

Initialize w and b randomly

for $e = 0, \text{num_epochs}$ **do**

for $b = 0, \text{num_batches}$ **do**

Compute: $dl(w, b)/dw$ and $dl(w, b)/db$

Update w : $w = w - \lambda dl(w, b)/dw - \lambda \alpha w$

Update b : $b = b - \lambda dl(w, b)/db - \lambda \alpha w$

Print: $l(w, b)$ // Useful to see if this is becoming smaller or not.

end

end

These are only approximations to the true gradient with respect to $L(w, b)$

Revisiting Another Problem with SGD

$\lambda = 0.01$

$$l(w, b) = l(w, b) + \alpha \sum_i |w_i|^2$$

Initialize w and b randomly

for $e = 0, \text{num_epochs}$ **do**

for $b = 0, \text{num_batches}$ **do**

Compute: $dl(w, b)/dw$ and $dl(w, b)/db$

Update w : $w = w - \lambda dl(w, b)/dw - \lambda \alpha w$

Update b : $b = b - \lambda dl(w, b)/db - \lambda \alpha w$

Print: $l(w, b)$ // Useful to see if this is becoming smaller or not.

end

end

This could lead to “un-learning” what has been learned in some previous steps of training.

Solution: Momentum Updates

$$\lambda = 0.01$$

$$l(w, b) = l(w, b) + \alpha \sum_i |w_i|^2$$

Initialize w and b randomly

for $e = 0, \text{num_epochs}$ **do**

for $b = 0, \text{num_batches}$ **do**

Compute: $dl(w, b)/dw$ and $dl(w, b)/db$

Update w : $w = w - \lambda dl(w, b)/dw - \lambda \alpha w$

Update b : $b = b - \lambda dl(w, b)/db - \lambda \alpha w$

Print: $l(w, b)$ // Useful to see if this is becoming smaller or not.

end

end

Keep track of previous gradients in an accumulator variable! and use a weighted average with current gradient.

Solution: Momentum Updates

$$\lambda = 0.01 \quad \tau = 0.9$$

Initialize w and b randomly

$$l(w, b) = l(w, b) + \alpha \sum_i |w_i|^2$$

global v

for $e = 0, \text{num_epochs}$ **do**

for $b = 0, \text{num_batches}$ **do**

Compute: $dl(w, b)/dw$

Compute: $v = \tau v + dl(w, b)/dw + \alpha w$

Update w : $w = w - \lambda v$

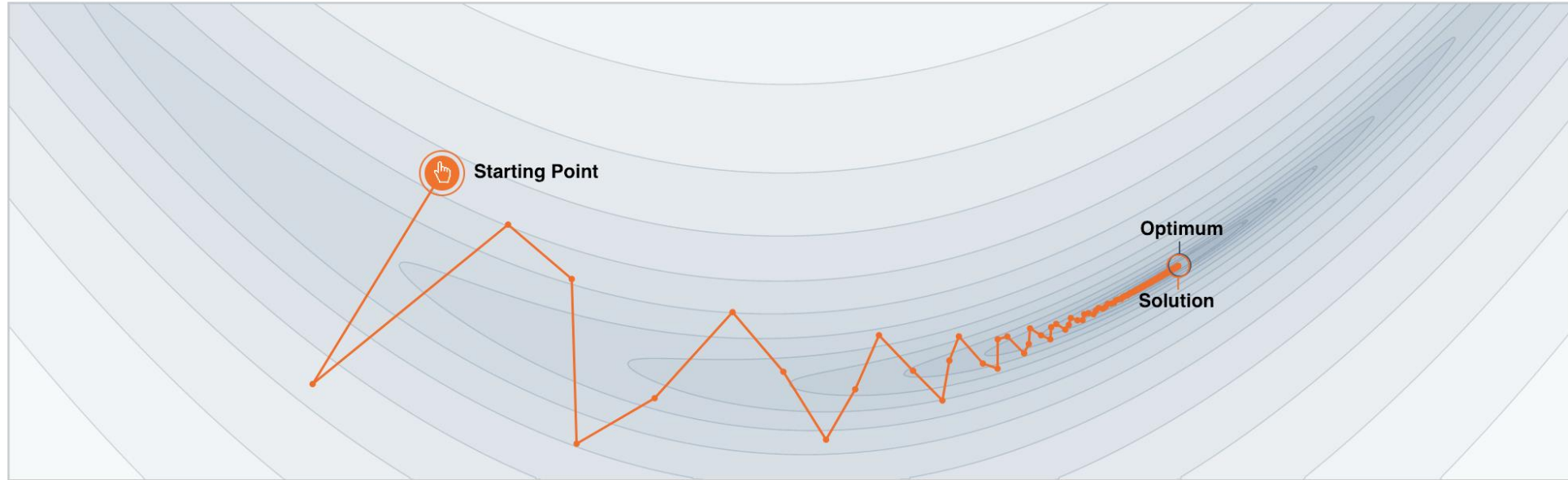
Print: $l(w, b)$ // Useful to see if this is becoming smaller or not.

end

end

Keep track of previous gradients in an accumulator variable! and use a weighted average with current gradient.

More on Momentum



Step-size $\alpha = 0.0050$



Momentum $\beta = 0.77$



We often think of Momentum as a means of dampening oscillations and speeding up the iterations, leading to faster convergence. But it has other interesting behavior. It allows a larger range of step-sizes to be used, and creates its own oscillations. What is going on?

<https://distill.pub/2017/momentum/>

Supervised Learning - Classification

Training Data



cat



dog



cat

.

.

.



bear

Test Data



.

.

.



Supervised Learning - Classification

Training Data

$$x_1 = [\text{] \quad y_1 = [\text{cat}]$$

$$x_2 = [\text{] \quad y_2 = [\text{dog}]$$

$$x_3 = [\text{] \quad y_3 = [\text{cat}]$$

•
•
•

$$x_n = [\text{] \quad y_n = [\text{bear}]$$

Supervised Learning - Classification

Training Data

inputs	targets / labels / ground truth	predictions
$x_1 = [x_{11} \ x_{12} \ x_{13} \ x_{14}]$	$y_1 = 1$	$\hat{y}_1 = 1$
$x_2 = [x_{21} \ x_{22} \ x_{23} \ x_{24}]$	$y_2 = 2$	$\hat{y}_2 = 2$
$x_3 = [x_{31} \ x_{32} \ x_{33} \ x_{34}]$	$y_3 = 1$	$\hat{y}_3 = 2$
•		
•		
•		
$x_n = [x_{n1} \ x_{n2} \ x_{n3} \ x_{n4}]$	$y_n = 3$	$\hat{y}_n = 1$

We need to find a function that maps x and y for any of them.

$$\hat{y}_i = f(x_i; \theta)$$

How do we "learn" the parameters of this function?

We choose ones that makes the following quantity small:

$$\sum_{i=1}^n Cost(\hat{y}_i, y_i)$$

Supervised Learning - Classification

Training Data



cat



dog



cat

•

•

•



bear

Test Data



•

•

•



Supervised Learning - Classification

Training Data

$$x_1 = [\text{img1}] \quad y_1 = [\text{cat}]$$

$$x_2 = [\text{img2}] \quad y_2 = [\text{dog}]$$

$$x_3 = [\text{img3}] \quad y_3 = [\text{cat}]$$

•
•
•

$$x_n = [\text{img4}] \quad y_n = [\text{bear}]$$

Supervised Learning - Classification

Training Data

inputs	targets / labels / ground truth	predictions
$x_1 = [x_{11} \ x_{12} \ x_{13} \ x_{14}]$	$y_1 = 1$	$\hat{y}_1 = 1$
$x_2 = [x_{21} \ x_{22} \ x_{23} \ x_{24}]$	$y_2 = 2$	$\hat{y}_2 = 2$
$x_3 = [x_{31} \ x_{32} \ x_{33} \ x_{34}]$	$y_3 = 1$	$\hat{y}_3 = 2$
$\vdots$		
$\vdots$		
$\vdots$		
$x_n = [x_{n1} \ x_{n2} \ x_{n3} \ x_{n4}]$	$y_n = 3$	$\hat{y}_n = 1$

We need to find a function that maps x and y for any of them.

$$\hat{y}_i = f(x_i; \theta)$$

How do we "learn" the parameters of this function?

We choose ones that makes the following quantity small:

$$\sum_{i=1}^n Cost(\hat{y}_i, y_i)$$

Supervised Learning – Linear Softmax

Training Data

inputs

targets /
labels /
ground truth

$$x_1 = [x_{11} \ x_{12} \ x_{13} \ x_{14}] \quad y_1 = 1$$

$$x_2 = [x_{21} \ x_{22} \ x_{23} \ x_{24}] \quad y_2 = 2$$

$$x_3 = [x_{31} \ x_{32} \ x_{33} \ x_{34}] \quad y_3 = 1$$

•
•
•

$$x_n = [x_{n1} \ x_{n2} \ x_{n3} \ x_{n4}] \quad y_n = 3$$

Supervised Learning – Linear Softmax

Training Data

inputs	targets / labels / ground truth	predictions
$x_1 = [x_{11} \ x_{12} \ x_{13} \ x_{14}]$	$y_1 = [1 \ 0 \ 0]$	$\hat{y}_1 = [0.85 \ 0.10 \ 0.05]$
$x_2 = [x_{21} \ x_{22} \ x_{23} \ x_{24}]$	$y_2 = [0 \ 1 \ 0]$	$\hat{y}_2 = [0.20 \ 0.70 \ 0.10]$
$x_3 = [x_{31} \ x_{32} \ x_{33} \ x_{34}]$	$y_3 = [1 \ 0 \ 0]$	$\hat{y}_3 = [0.40 \ 0.45 \ 0.15]$
•		
•		
•		
$x_n = [x_{n1} \ x_{n2} \ x_{n3} \ x_{n4}]$	$y_n = [0 \ 0 \ 1]$	$\hat{y}_n = [0.40 \ 0.25 \ 0.35]$

Supervised Learning – Linear Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}] \quad y_i = [1 \ 0 \ 0] \quad \hat{y}_i = [f_1 \ f_2 \ f_3]$$

$$a_1 = w_{11}x_{i1} + w_{12}x_{i2} + w_{13}x_{i3} + w_{14}x_{i4} + b_c$$

$$a_2 = w_{21}x_{i1} + w_{22}x_{i2} + w_{23}x_{i3} + w_{24}x_{i4} + b_d$$

$$a_3 = w_{31}x_{i1} + w_{32}x_{i2} + w_{33}x_{i3} + w_{34}x_{i4} + b_b$$

$$f_1 = e^{a_1} / (e^{a_1} + e^{a_2} + e^{a_3})$$

$$f_2 = e^{a_2} / (e^{a_1} + e^{a_2} + e^{a_3})$$

$$f_3 = e^{a_3} / (e^{a_1} + e^{a_2} + e^{a_3})$$

How do we find a good w and b?

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}] \quad y_i = [1 \ 0 \ 0] \quad \hat{y}_i = [f_1(w, b) \ f_2(w, b) \ f_3(w, b)]$$

We need to find w, and b that minimize the following:

$$L(w, b) = \sum_{i=1}^n \sum_{j=1}^3 -y_{i,j} \log(\hat{y}_{i,j}) = \sum_{i=1}^n -\log(\hat{y}_{i,label}) = \sum_{i=1}^n -\log f_{i,label}(w, b)$$

Why?

Computing Analytic Gradients

This is what we have:

$$L(w, b) = \sum_{i=1}^n \sum_{j=1}^3 -y_{i,j} \log(\hat{y}_{i,j}) = \sum_{i=1}^n -\log(\hat{y}_{i,label}) = \sum_{i=1}^n -\log f_{i,label}(w, b)$$

To simplify let's assume $n = 1$

$$\ell(W, b) = -\log(\hat{y}_{label}(W, b)) = -\log\left(\frac{\exp(a_{label}(W, b))}{\sum_{k=1}^3 \exp(a_k(W, b))}\right)$$

Supervised Learning – Linear Softmax

$$x = [x_1 \ x_2 \ x_3 \ x_4] \qquad y = [1 \ 0 \ 0] \qquad \hat{y} = [f_1 \ f_2 \ f_3]$$

$$a_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4 + b_c$$

$$a_2 = w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + w_{24}x_4 + b_d$$

$$a_3 = w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + w_{34}x_4 + b_b$$

$$f_1 = e^{a_1} / (e^{a_1} + e^{a_2} + e^{a_3})$$

$$f_2 = e^{a_2} / (e^{a_1} + e^{a_2} + e^{a_3})$$

$$f_3 = e^{a_3} / (e^{a_1} + e^{a_2} + e^{a_3})$$

Computing Analytic Gradients

This is what we have:

$$\ell(W, b) = -\log(\hat{y}_{label}(W, b)) = -\log\left(\frac{\exp(a_{label}(W, b))}{\sum_{k=1}^3 \exp(a_k(W, b))}\right)$$

Computing Analytic Gradients

This is what we have:

$$\ell(W, b) = -\log(\hat{y}_{label}(W, b)) = -\log\left(\frac{\exp(a_{label}(W, b))}{\sum_{k=1}^3 \exp(a_k(W, b))}\right)$$

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^3 \exp(a_k)}\right)$$

Reminder: $a_i = (w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i$

Computing Analytic Gradients

This is what we have:

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^3 \exp(a_k)}\right)$$

Computing Analytic Gradients

This is what we have:

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^3 \exp(a_k)}\right)$$

This is what we need:

$$\frac{\partial \ell}{\partial w_{ij}} \quad \text{for each } w_{ij}$$

$$\frac{\partial \ell}{\partial b_i} \quad \text{for each } b_i$$

Computing Analytic Gradients

This is what we have:

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^3 \exp(a_k)}\right)$$

Step 1: Chain Rule of Calculus

$$\frac{\partial \ell}{\partial w_{ij}} = \frac{\partial \ell}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}}$$

$$\frac{\partial \ell}{\partial b_i} = \frac{\partial \ell}{\partial a_i} \frac{\partial a_i}{\partial b_i}$$

Computing Analytic Gradients

This is what we have:

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^3 \exp(a_k)}\right)$$

Step 1: Chain Rule of Calculus

Let's do these first

$$\frac{\partial \ell}{\partial w_{ij}} = \frac{\partial \ell}{\partial a_i} \boxed{\frac{\partial a_i}{\partial w_{ij}}}$$

$$\frac{\partial \ell}{\partial b_i} = \frac{\partial \ell}{\partial a_i} \boxed{\frac{\partial a_i}{\partial b_i}}$$

Computing Analytic Gradients

$$\frac{\partial a_i}{\partial w_{ij}}$$

$$\frac{\partial a_i}{\partial b_i}$$

$$a_i = (w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i$$

$$\frac{\partial a_i}{\partial w_{i,3}} = \frac{\partial}{\partial w_{i,3}} (w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i$$

$$\frac{\partial a_i}{\partial w_{i,3}} = x_3$$

$$\frac{\partial a_i}{\partial w_{i,j}} = x_j$$

Computing Analytic Gradients

$$\frac{\partial a_i}{\partial w_{i,j}} = x_j$$

$$\frac{\partial a_i}{\partial b_i}$$

$$a_i = (w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i$$

$$\frac{\partial a_i}{\partial b_i} = \frac{\partial}{\partial b_i} (w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i$$

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

Computing Analytic Gradients

$$\frac{\partial a_i}{\partial w_{i,j}} = x_j$$

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

Computing Analytic Gradients

This is what we have:

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^3 \exp(a_k)}\right)$$

Step 1: Chain Rule of Calculus

Now let's do this one (same for both!)

$$\frac{\partial \ell}{\partial w_{ij}} = \boxed{\frac{\partial \ell}{\partial a_i}} \frac{\partial a_i}{\partial w_{ij}}$$

$$\frac{\partial \ell}{\partial b_i} = \boxed{\frac{\partial \ell}{\partial a_i}} \frac{\partial a_i}{\partial b_i}$$

Computing Analytic Gradients

$$\begin{aligned}\frac{\partial \ell}{\partial a_i} &= \frac{\partial}{\partial a_i} \left[-\log \left(\frac{\exp(a_{label})}{\sum_{k=1}^3 \exp(a_k)} \right) \right] \\ &= \frac{\partial}{\partial a_i} \left[\log \left(\sum_{k=1}^3 \exp(a_k) \right) - a_{label} \right]\end{aligned}$$

In our cat, dog, bear classification example: $i = \{1, 2, 3\}$

Computing Analytic Gradients

$$\begin{aligned}\frac{\partial \ell}{\partial a_i} &= \frac{\partial}{\partial a_i} \left[-\log \left(\frac{\exp(a_{\text{label}})}{\sum_{k=1}^3 \exp(a_k)} \right) \right] \\ &= \frac{\partial}{\partial a_i} \left[\log \left(\sum_{k=1}^3 \exp(a_k) \right) - a_{\text{label}} \right]\end{aligned}$$

In our cat, dog, bear classification example: $i = \{1, 2, 3\}$

Let's say: label = 2

We need:

$$\frac{\partial \ell}{\partial a_1} \quad \frac{\partial \ell}{\partial a_2} \quad \frac{\partial \ell}{\partial a_3}$$

Computing Analytic Gradients

$$= \frac{\partial}{\partial a_i} \left[\log \left(\sum_{k=1}^3 \exp(a_k) \right) - a_{label} \right]$$

$$\frac{\partial \ell}{\partial a_1} \quad \frac{\partial \ell}{\partial a_3} \quad \text{when } i \neq label:$$

$$\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i} \left[\log \left(\sum_{k=1}^3 \exp(a_k) \right) - a_{label} \right]$$

$$\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i} \log \left(\sum_{k=1}^3 \exp(a_k) \right)$$

$$\frac{\partial \ell}{\partial a_i} = \left(\frac{1}{\sum_{k=1}^3 \exp(a_k)} \right) \left(\frac{\partial}{\partial a_i} \sum_{k=1}^3 \exp(a_k) \right)$$

$$\frac{\partial \ell}{\partial a_i} = \frac{\exp(a_i)}{\sum_{k=1}^3 \exp(a_k)} = \hat{y}_i$$

Supervised Learning – Linear Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}] \quad y_i = [1 \ 0 \ 0] \quad \hat{y}_i = [f_1 \ f_2 \ f_3]$$

$$a_1 = w_{11}x_{i1} + w_{12}x_{i2} + w_{13}x_{i3} + w_{14}x_{i4} + b_c$$

$$a_2 = w_{21}x_{i1} + w_{22}x_{i2} + w_{23}x_{i3} + w_{24}x_{i4} + b_d$$

$$a_3 = w_{31}x_{i1} + w_{32}x_{i2} + w_{33}x_{i3} + w_{34}x_{i4} + b_b$$

$$f_1 = e^{a_1} / (e^{a_1} + e^{a_2} + e^{a_3})$$

$$f_2 = e^{a_2} / (e^{a_1} + e^{a_2} + e^{a_3})$$

$$f_3 = e^{a_3} / (e^{a_1} + e^{a_2} + e^{a_3})$$

Computing Analytic Gradients

$$= \frac{\partial}{\partial a_i} \left[\log \left(\sum_{k=1}^3 \exp(a_k) \right) - a_{label} \right]$$

$$\frac{\partial \ell}{\partial a_1} \quad \frac{\partial \ell}{\partial a_3} \quad \text{when } i \neq label:$$

$$\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i} \left[\log \left(\sum_{k=1}^3 \exp(a_k) \right) - a_{label} \right]$$

$$\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i} \log \left(\sum_{k=1}^3 \exp(a_k) \right)$$

$$\frac{\partial \ell}{\partial a_i} = \left(\frac{1}{\sum_{k=1}^3 \exp(a_k)} \right) \left(\frac{\partial}{\partial a_i} \sum_{k=1}^3 \exp(a_k) \right)$$

$$\frac{\partial \ell}{\partial a_i} = \frac{\exp(a_i)}{\sum_{k=1}^3 \exp(a_k)} = \hat{y}_i$$

Computing Analytic Gradients

$$= \frac{\partial}{\partial a_i} \left[\log \left(\sum_{k=1}^3 \exp(a_k) \right) - a_{label} \right]$$

$$\frac{\partial \ell}{\partial a_2}$$

when $i = label$:

$$\frac{\partial \ell}{\partial a_{label}} = \frac{\partial}{\partial a_{label}} \left[\log \left(\sum_{k=1}^3 \exp(a_k) \right) - a_{label} \right]$$

$$\frac{\partial \ell}{\partial a_{label}} = \frac{\partial}{\partial a_{label}} \log \left(\sum_{k=1}^3 \exp(a_k) \right) - 1$$

$$\frac{\partial \ell}{\partial a_{label}} = \left(\frac{1}{\sum_{k=1}^3 \exp(a_k)} \right) \left(\frac{\partial}{\partial a_{label}} \sum_{k=1}^3 \exp(a_k) \right) - 1$$

$$\frac{\partial \ell}{\partial a_{label}} = \frac{\exp(a_{label})}{\sum_{k=1}^3 \exp(a_k)} - 1 \quad \hat{y}_i - 1$$

Computing Analytic Gradients

label = 2

$$\frac{\partial \ell}{\partial a_1} = \hat{y}_1$$

$$\frac{\partial \ell}{\partial a_2} = \hat{y}_2 - 1$$

$$\frac{\partial \ell}{\partial a_3} = \hat{y}_3$$

$$\frac{\partial \ell}{\partial a} = \begin{bmatrix} \frac{\partial \ell}{\partial a_1} \\ \frac{\partial \ell}{\partial a_2} \\ \frac{\partial \ell}{\partial a_3} \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 - 1 \\ \hat{y}_3 \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \hat{y} - y$$

$$\frac{\partial \ell}{\partial a_i} = \hat{y}_i - y_i$$

Computing Analytic Gradients

$$\frac{\partial \ell}{\partial w_{ij}} = \frac{\partial \ell}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}}$$

$$\frac{\partial \ell}{\partial b_i} = \frac{\partial \ell}{\partial a_i} \frac{\partial a_i}{\partial b_i}$$

$$\frac{\partial a_i}{\partial w_{i,j}} = x_j$$

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

$$\frac{\partial \ell}{\partial a_i} = \hat{y}_i - y_i$$

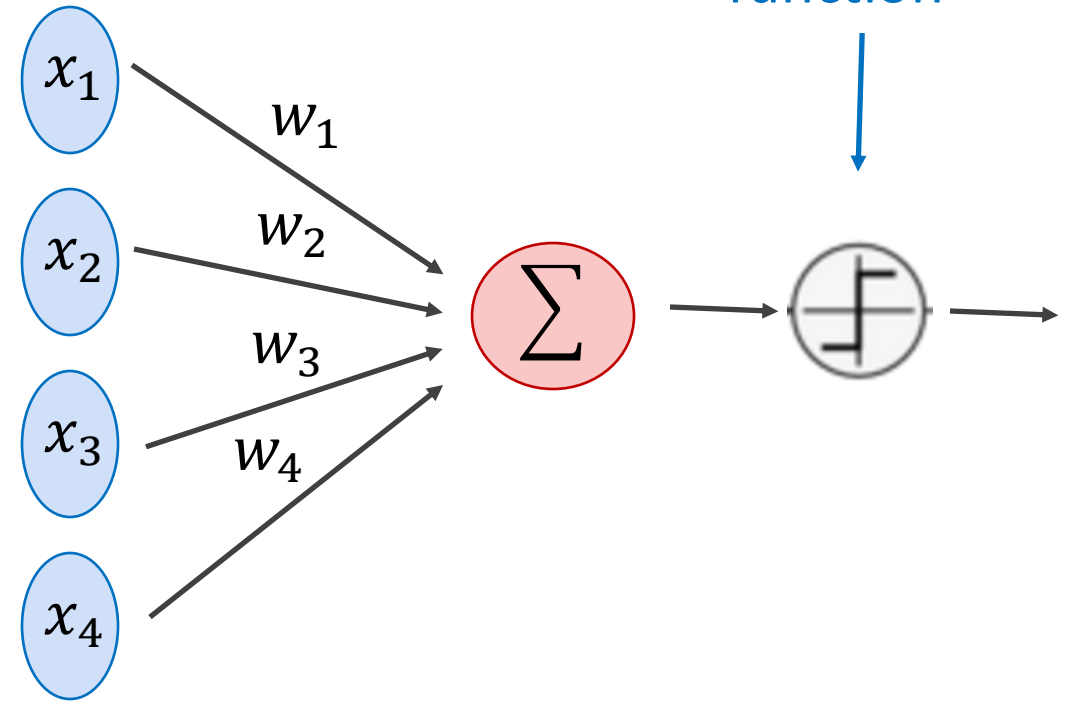
$$\frac{\partial \ell}{\partial w_{i,j}} = (\hat{y}_i - y_i)x_j$$

$$\frac{\partial \ell}{\partial b_i} = (\hat{y}_i - y_i)$$

Perceptron Model

Frank Rosenblatt (1957) - Cornell University

$$f(x) = \begin{cases} 1, & \text{if } \sum_{i=0}^n w_i x_i + b > 0 \\ 0, & \text{otherwise} \end{cases}$$

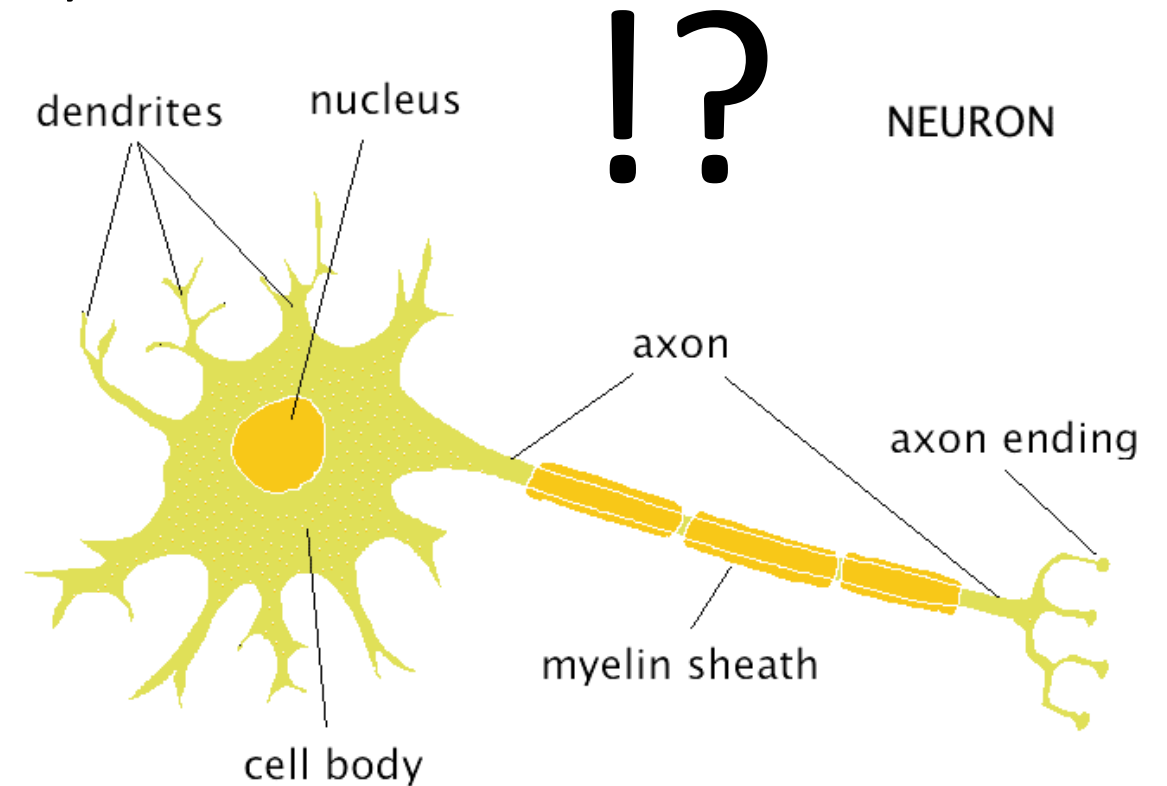


More: <https://en.wikipedia.org/wiki/Perceptron>

Perceptron Model

Frank Rosenblatt (1957) - Cornell University

$$f(x) = \begin{cases} 1, & \text{if } \sum_{i=0}^n w_i x_i + b > 0 \\ 0, & \text{otherwise} \end{cases}$$

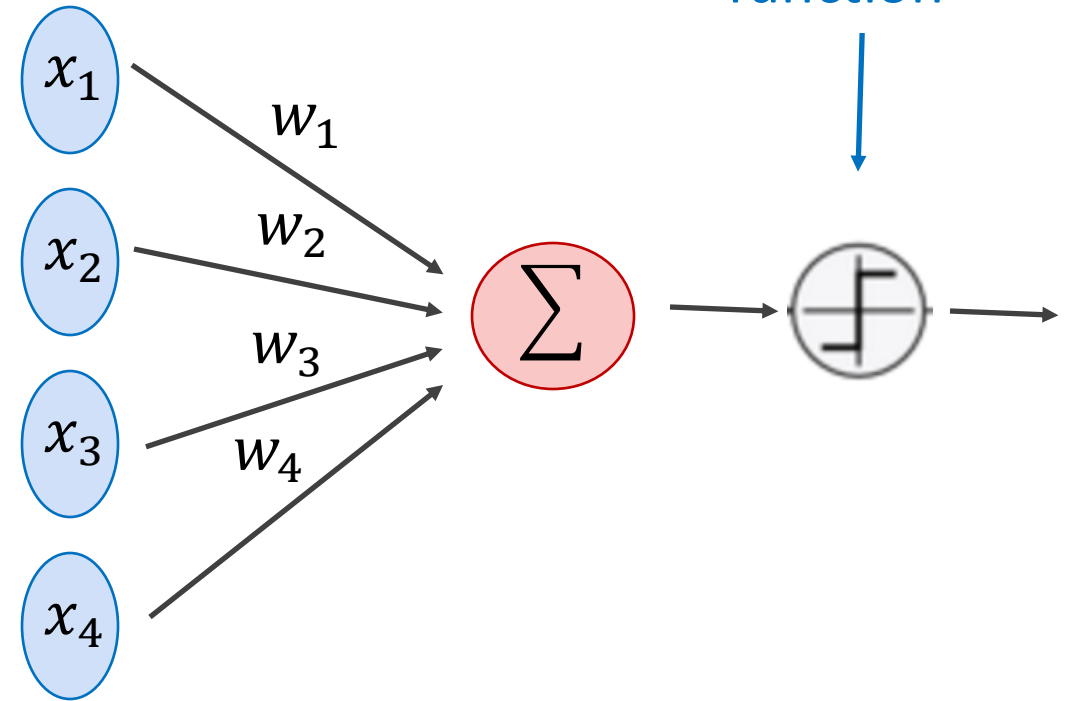


More: <https://en.wikipedia.org/wiki/Perceptron>

Perceptron Model

Frank Rosenblatt (1957) - Cornell University

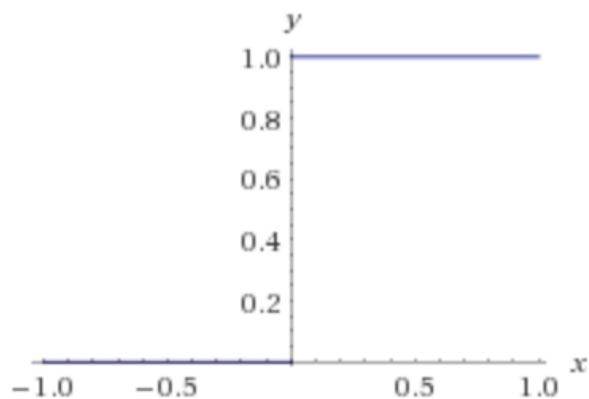
$$f(x) = \begin{cases} 1, & \text{if } \sum_{i=0}^n w_i x_i + b > 0 \\ 0, & \text{otherwise} \end{cases}$$



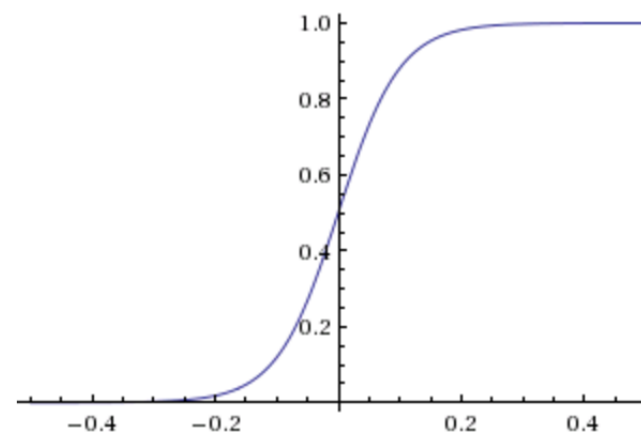
More: <https://en.wikipedia.org/wiki/Perceptron>

Activation Functions

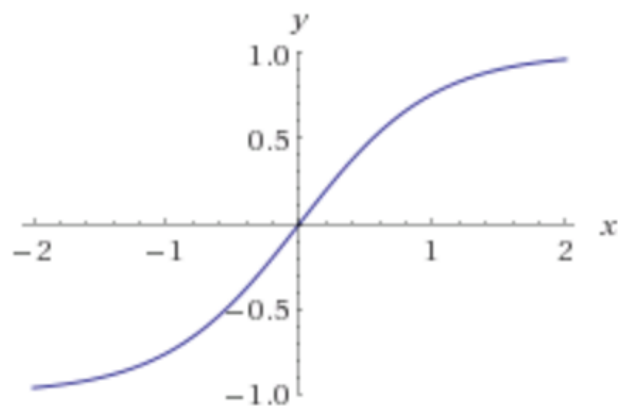
Step(x)



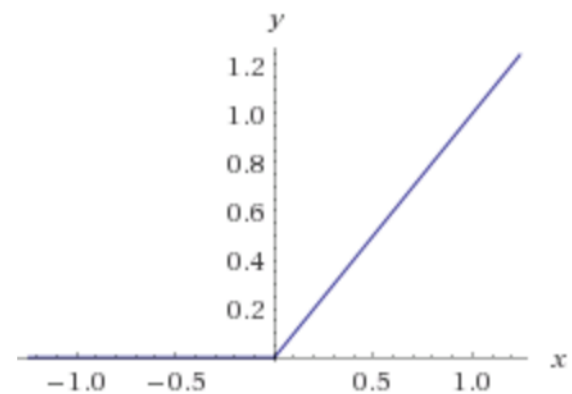
Sigmoid(x)



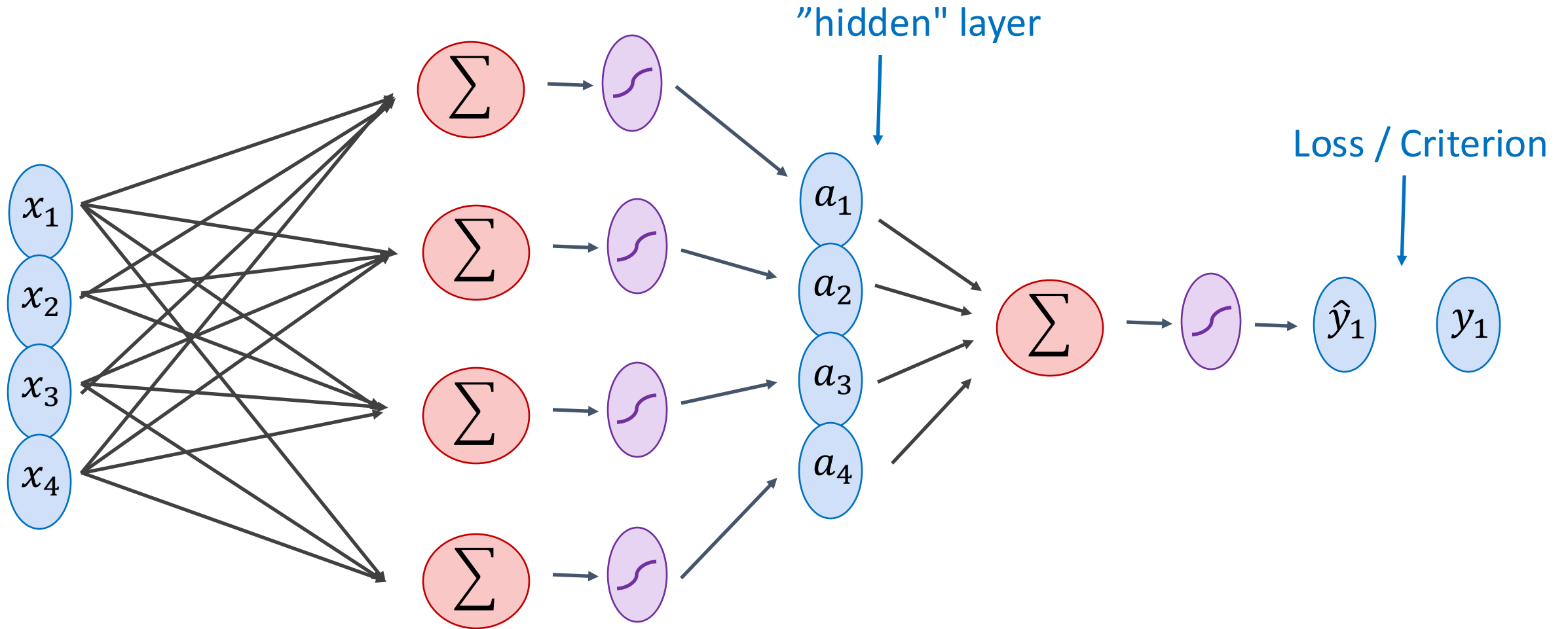
Tanh(x)



$\text{ReLU}(x) = \max(0, x)$



Two-layer Multi-layer Perceptron (MLP)



Linear Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_d \ f_b]$$

$$g_c = w_{c1}x_{i1} + w_{c2}x_{i2} + w_{c3}x_{i3} + w_{c4}x_{i4} + b_c$$

$$g_d = w_{d1}x_{i1} + w_{d2}x_{i2} + w_{d3}x_{i3} + w_{d4}x_{i4} + b_d$$

$$g_b = w_{b1}x_{i1} + w_{b2}x_{i2} + w_{b3}x_{i3} + w_{b4}x_{i4} + b_b$$

$$f_c = e^{g_c} / (e^{g_c} + e^{g_d} + e^{g_b})$$

$$f_d = e^{g_d} / (e^{g_c} + e^{g_d} + e^{g_b})$$

$$f_b = e^{g_b} / (e^{g_c} + e^{g_d} + e^{g_b})$$

Linear Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_d \ f_b]$$

$$g_c = w_{c1}x_{i1} + w_{c2}x_{i2} + w_{c3}x_{i3} + w_{c4}x_{i4} + b_c$$

$$g_d = w_{d1}x_{i1} + w_{d2}x_{i2} + w_{d3}x_{i3} + w_{d4}x_{i4} + b_d$$

$$g_b = w_{b1}x_{i1} + w_{b2}x_{i2} + w_{b3}x_{i3} + w_{b4}x_{i4} + b_b$$

$$W = \begin{bmatrix} w_{c1} & w_{c2} & w_{c3} & w_{c4} \\ w_{d1} & w_{d2} & w_{d3} & w_{d4} \\ w_{b1} & w_{b2} & w_{b3} & w_{b4} \end{bmatrix}$$

$$b = [b_c \ b_d \ b_b]$$

$$f_c = e^{g_c} / (e^{g_c} + e^{g_d} + e^{g_b})$$

$$f_d = e^{g_d} / (e^{g_c} + e^{g_d} + e^{g_b})$$

$$f_b = e^{g_b} / (e^{g_c} + e^{g_d} + e^{g_b})$$

Linear Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_d \ f_b]$$

$$g = wx^T + b^T$$

$$w = \begin{bmatrix} w_{c1} & w_{c2} & w_{c3} & w_{c4} \\ w_{d1} & w_{d2} & w_{d3} & w_{d4} \\ w_{b1} & w_{b2} & w_{b3} & w_{b4} \end{bmatrix}$$

$$b = [b_c \ b_d \ b_b]$$

$$f_c = e^{g_c} / (e^{g_c} + e^{g_d} + e^{g_b})$$

$$f_d = e^{g_d} / (e^{g_c} + e^{g_d} + e^{g_b})$$

$$f_b = e^{g_b} / (e^{g_c} + e^{g_d} + e^{g_b})$$

Linear Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_d \ f_b]$$

$$g = wx^T + b^T$$

$$w = \begin{bmatrix} w_{c1} & w_{c2} & w_{c3} & w_{c4} \\ w_{d1} & w_{d2} & w_{d3} & w_{d4} \\ w_{b1} & w_{b2} & w_{b3} & w_{b4} \end{bmatrix}$$

$$b = [b_c \ b_d \ b_b]$$

$$f = \text{softmax}(g)$$

Linear Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_a \ f_b]$$

$$f = \text{softmax}(wx^T + b^T)$$

Two-layer MLP + Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_a \ f_b]$$

$$a_1 = \text{sigmoid}(w_{[1]}x^T + b_{[1]}^T)$$

$$f = \text{softmax}(w_{[2]}a_1^T + b_{[2]}^T)$$

N-layer MLP + Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_a \ f_b]$$

$$a_1 = \text{sigmoid}(w_{[1]}x^T + b_{[1]}^T)$$

$$a_2 = \text{sigmoid}(w_{[2]}a_1^T + b_{[2]}^T)$$

...

$$a_k = \text{sigmoid}(w_{[k]}a_{k-1}^T + b_{[k]}^T)$$

...

$$f = \text{softmax}(w_{[n]}a_{n-1}^T + b_{[n]}^T)$$

How to train the parameters?

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_a \ f_b]$$

$$a_1 = \textit{sigmoid}(w_{[1]}x^T + b_{[1]}^T)$$

$$a_2 = \textit{sigmoid}(w_{[2]}a_1^T + b_{[2]}^T)$$

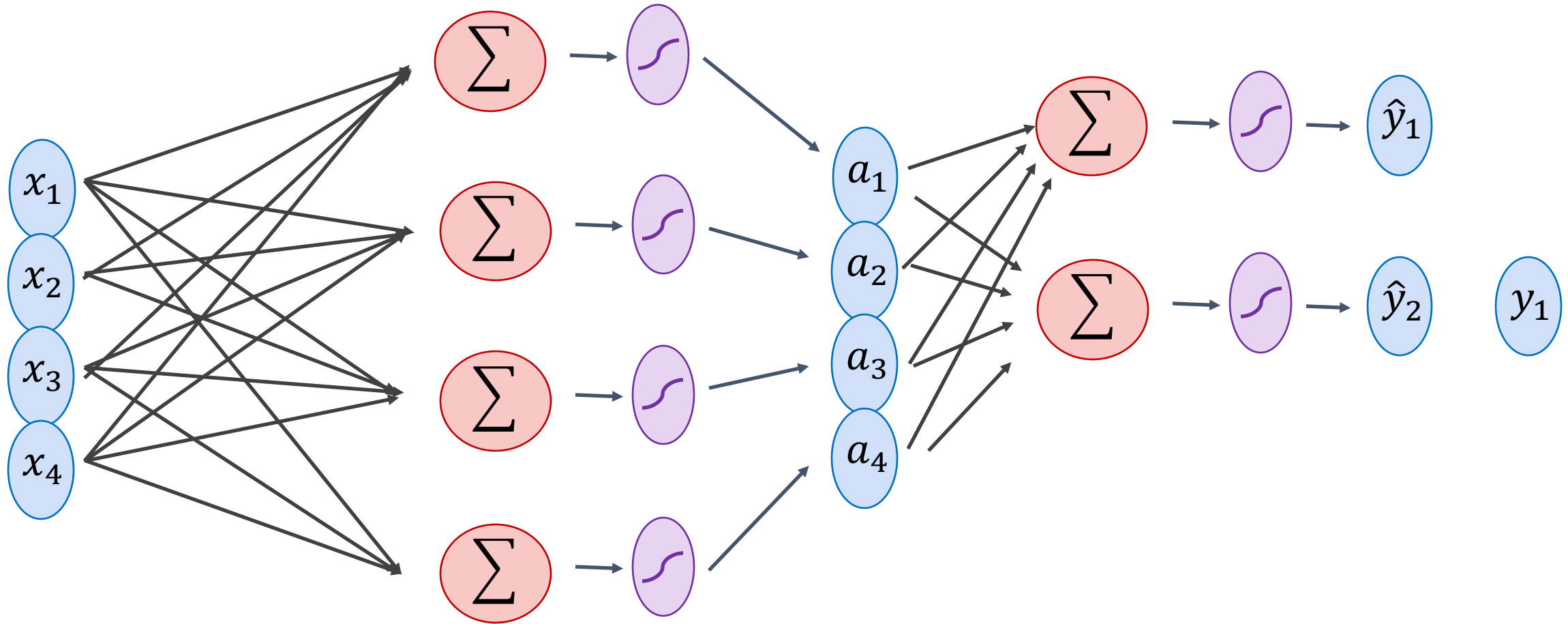
...

$$a_k = \textit{sigmoid}(w_{[k]}a_{k-1}^T + b_{[k]}^T)$$

...

$$f = \textit{softmax}(w_{[n]}a_{n-1}^T + b_{[n]}^T)$$

Forward pass (Forward-propagation)



Forward pass (Forward-propagation)

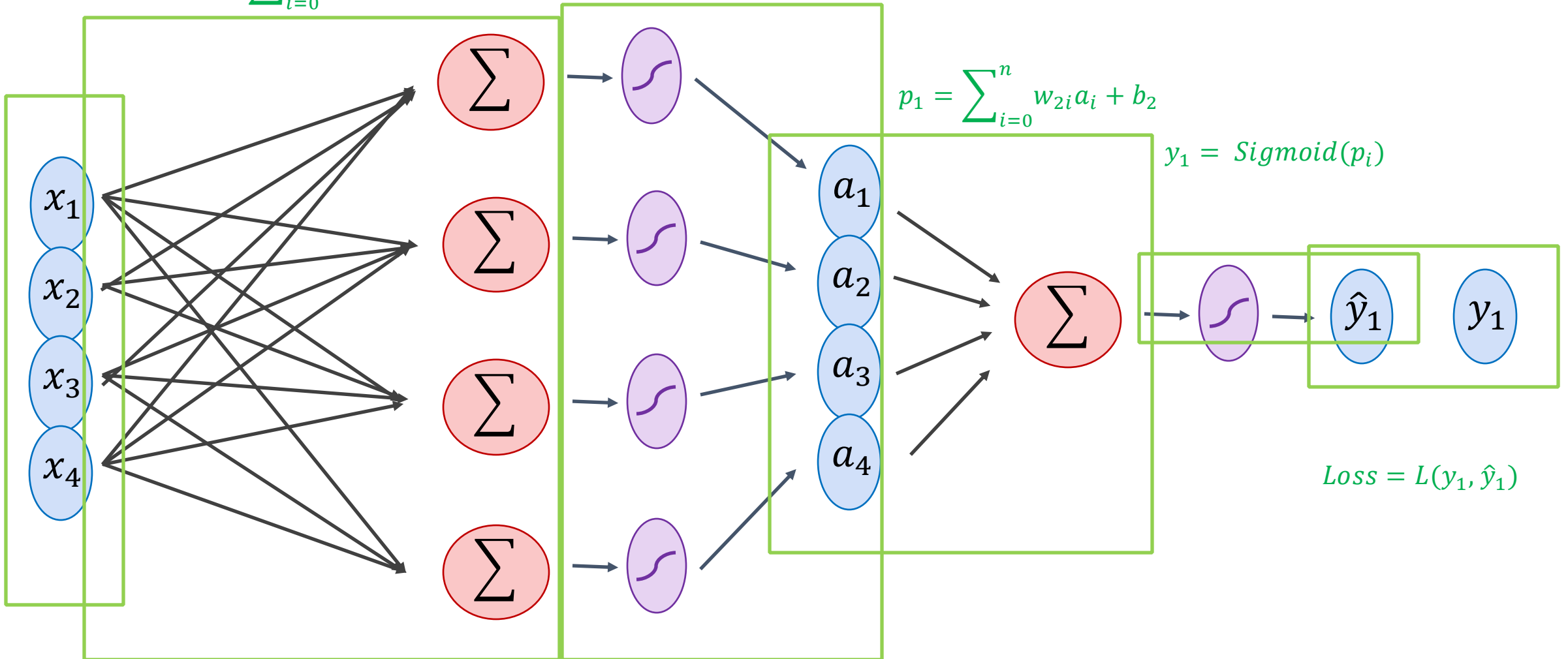
$$z_i = \sum_{i=0}^n w_{1ij}x_i + b_1$$

$$a_i = \text{Sigmoid}(z_i)$$

$$p_1 = \sum_{i=0}^n w_{2i}a_i + b_2$$

$$y_1 = \text{Sigmoid}(p_1)$$

$$\text{Loss} = L(y_1, \hat{y}_1)$$



How to train the parameters?

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_a \ f_b]$$

$$a_1 = \text{sigmoid}(w_{[1]}x^T + b_{[1]}^T)$$

$$a_2 = \text{sigmoid}(w_{[2]}a_1^T + b_{[2]}^T)$$

...

$$a_k = \text{sigmoid}(w_{[k]}a_{k-1}^T + b_{[i]}^T)$$

...

$$f = \text{softmax}(w_{[n]}a_{n-1}^T + b_{[n]}^T)$$

We can still use SGD

We need!

$$\frac{\partial l}{\partial w_{[k]ij}}$$

$$\frac{\partial l}{\partial b_{[k]i}}$$

How to train the parameters?

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_a \ f_b]$$

$$a_1 = \text{sigmoid}(w_{[1]}x^T + b_{[1]}^T)$$

$$a_2 = \text{sigmoid}(w_{[2]}a_1^T + b_{[2]}^T)$$

...

$$a_i = \text{sigmoid}(w_{[k]}a_{k-1}^T + b_{[k]}^T)$$

...

$$f = \text{softmax}(w_{[n]}a_{n-1}^T + b_{[n]}^T)$$

$$l = \text{loss}(f, y)$$

We can still use SGD

We need!

$$\frac{\partial l}{\partial w_{[k]ij}}$$

$$\frac{\partial l}{\partial b_{[k]i}}$$

How to train the parameters?

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_a \ f_b]$$

$$a_1 = \text{sigmoid}(w_{[1]}x^T + b_{[1]}^T)$$

$$a_2 = \text{sigmoid}(w_{[2]}a_1^T + b_{[2]}^T)$$

...

$$a_i = \text{sigmoid}(w_{[k]}a_{k-1}^T + b_{[k]}^T)$$

...

$$f = \text{softmax}(w_{[n]}a_{n-1}^T + b_{[n]}^T)$$

$$l = \text{loss}(f, y)$$

We can still use SGD

We need!

$$\frac{\partial l}{\partial w_{[k]ij}}$$

$$\frac{\partial l}{\partial b_{[k]i}}$$

How to train the parameters?

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_a \ f_b]$$

$$a_1 = \text{sigmoid}(w_{[1]}x^T + b_{[1]}^T)$$

$$a_2 = \text{sigmoid}(w_{[2]}a_1^T + b_{[2]}^T)$$

...

$$a_i = \text{sigmoid}(w_{[k]}a_{k-1}^T + b_{[k]}^T)$$

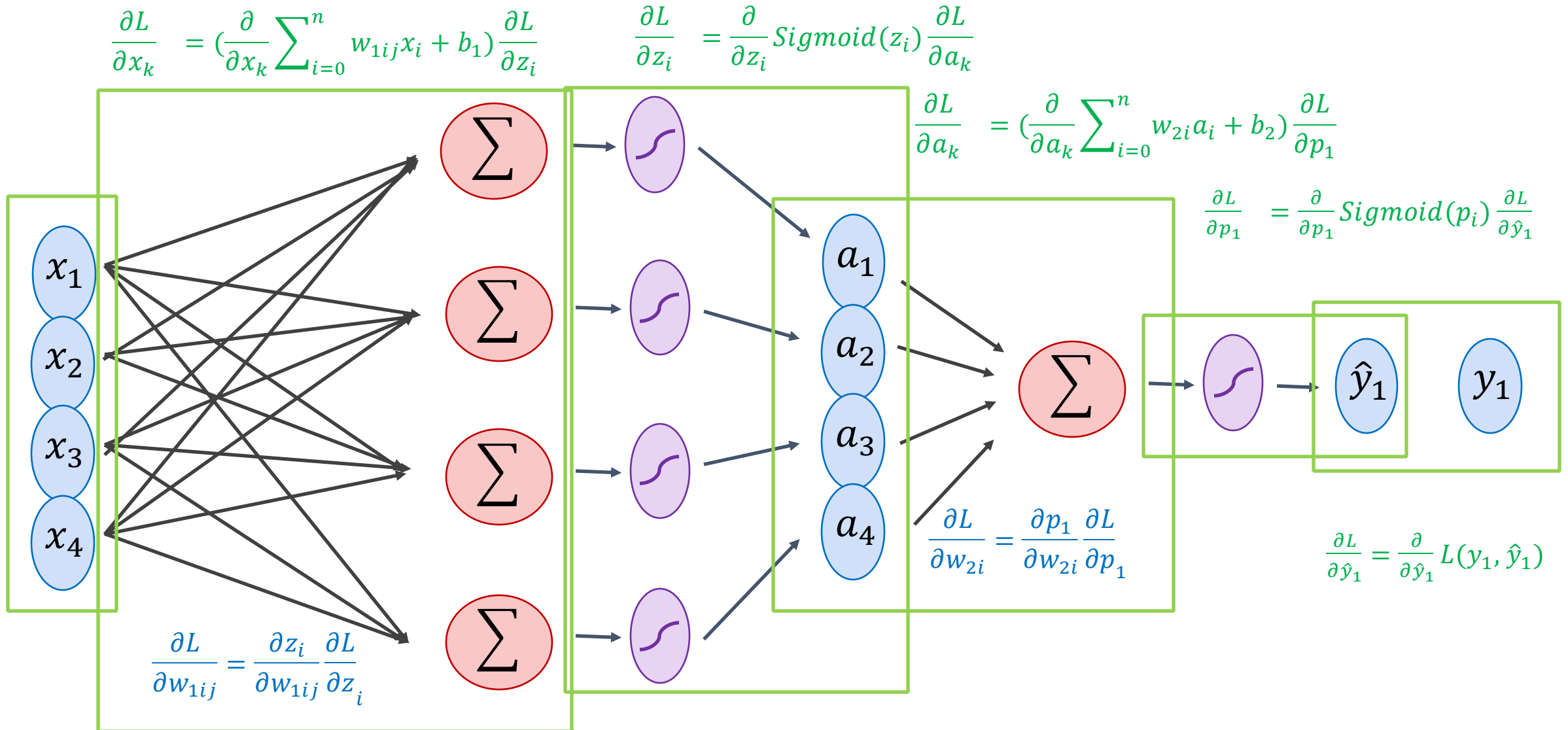
...

$$f = \text{softmax}(w_{[n]}a_{n-1}^T + b_{[n]}^T)$$

$$l = \text{loss}(f, y)$$

$$\frac{\partial l}{\partial w_{[k]ij}} = \frac{\partial l}{\partial a_{n-1}} \frac{\partial a_{n-1}}{\partial a_{n-2}} \cdots \frac{\partial a_{k-2}}{\partial a_{k-1}} \frac{\partial a_{k-1}}{\partial w_{[k]ij}}$$

Backward pass (Back-propagation)



Questions?