Deep Learning for Vision & Language

Machine Learning II: SGD, Generalization, Regularization, Softmax, MLPs

RICE UNIVERSITY

About the class

- COMP 646: Deep Learning for Vision and Language
- Instructor: **Vicente** Ordóñez (Vicente Ordóñez Román)
- Website: https://www.cs.rice.edu/~vo9/deep-vislang
- Location: Keck Hall 100
- Times: Tuesdays and Thursdays
  from 4pm to 5:15pm
- Office Hours: Wednesdays 10am to noon (DH2080)
- Teaching Assistants: **Ayush, Jefferson, Jaywon, Zilin**
- Discussion Forum: Piazza (Sign-up Link on Rice Canvas and Class Website)

Assignment 1

- Assignment 1 is released and is available on the class website and to be submitted via Canvas.

- Due: Friday January 26th, midnight (you can and should submit early but not late – do not wait until finishing the whole assignment to have a version uploaded on canvas)

Grading for this class: COMP 646

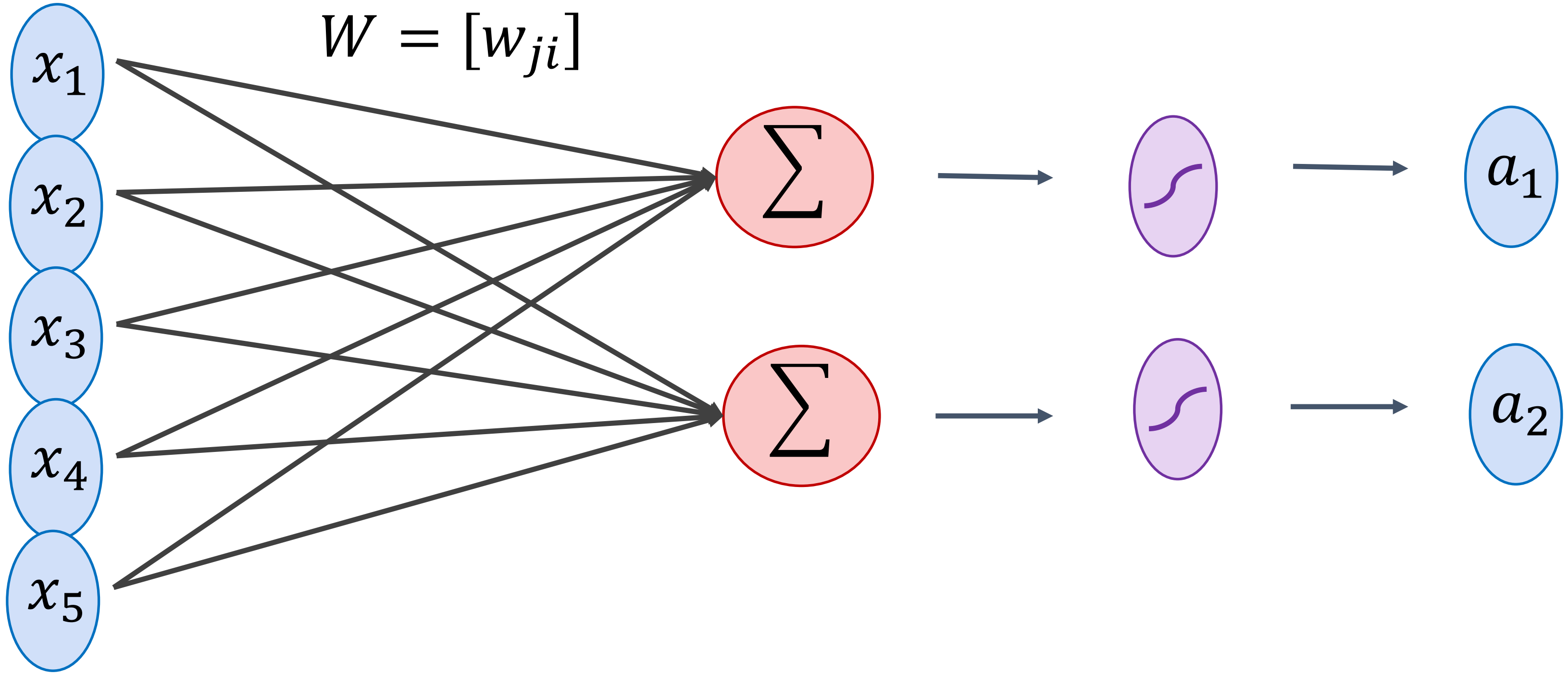
- Assignments: 30pts (3 assignments: 10pts + 10pts + 10pts)
- Class Project: 60pts
- Quiz: 10pts

Total: 100pts

- Grade cutoffs: no stricter than the following:
  A [between 90% and 100%], B [between 80% and 90%),
  C [between 70% and 80%), D [between 55% and 70%),
  F [less than 55%)

Neural Network with One Layer

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



$$a_j = sigmoid(\textstyle\sum_i w_{ji} x_i + b_j)$$

Gradient Descent

$\lambda = 0.01$

$$L(w,b) = \sum_{i=1}^{n} l(w,b)$$

expensive

Initialize w and b randomly

for e = 0, num_epochs **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

Stochastic Gradient Descent (mini-batch)

$\lambda = 0.01$

$$L_B(w,b) = \sum_{i=1}^{B} l(w,b)$$

Initialize w and b randomly

for e = 0, num_epochs **do**

for b = 0, num_batches **do**

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

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

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

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

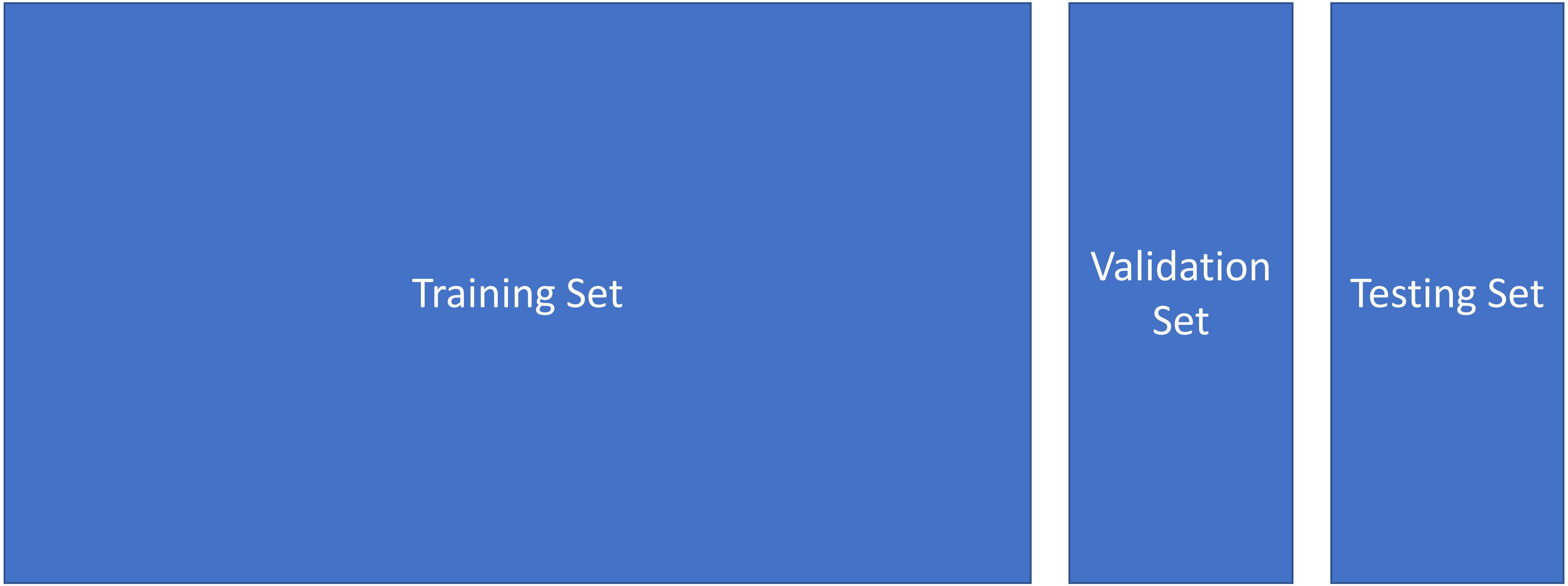
end

end

Stochastic Gradient Descent

- How to choose the right batch size B?
- How to choose the right learning rate lambda?
- How to choose the right loss function, e.g. is least squares good enough?
- How to choose the right function/classifier, e.g. linear, quadratic, neural network with 1 layer, 2 layers, etc?

Training, Validation (Dev), Test Sets



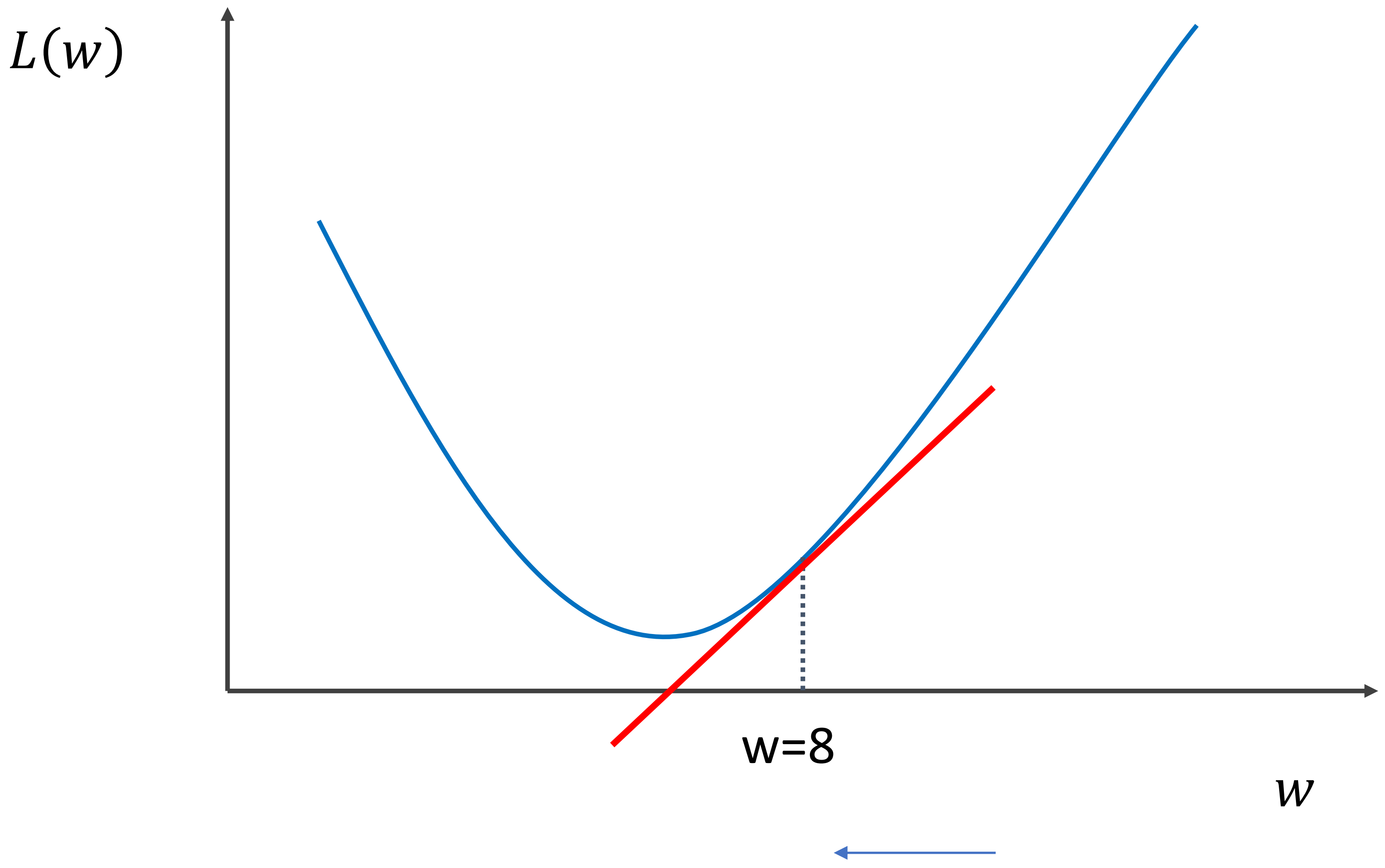
Training, Validation (Dev), Test Sets

Used during development

Training, Validation (Dev), Test Sets

Only to be used for evaluating the model at the very end of development and any changes to the model after running it on the test set, could be influenced by what you saw happened on the test set, which would invalidate any future evaluation.

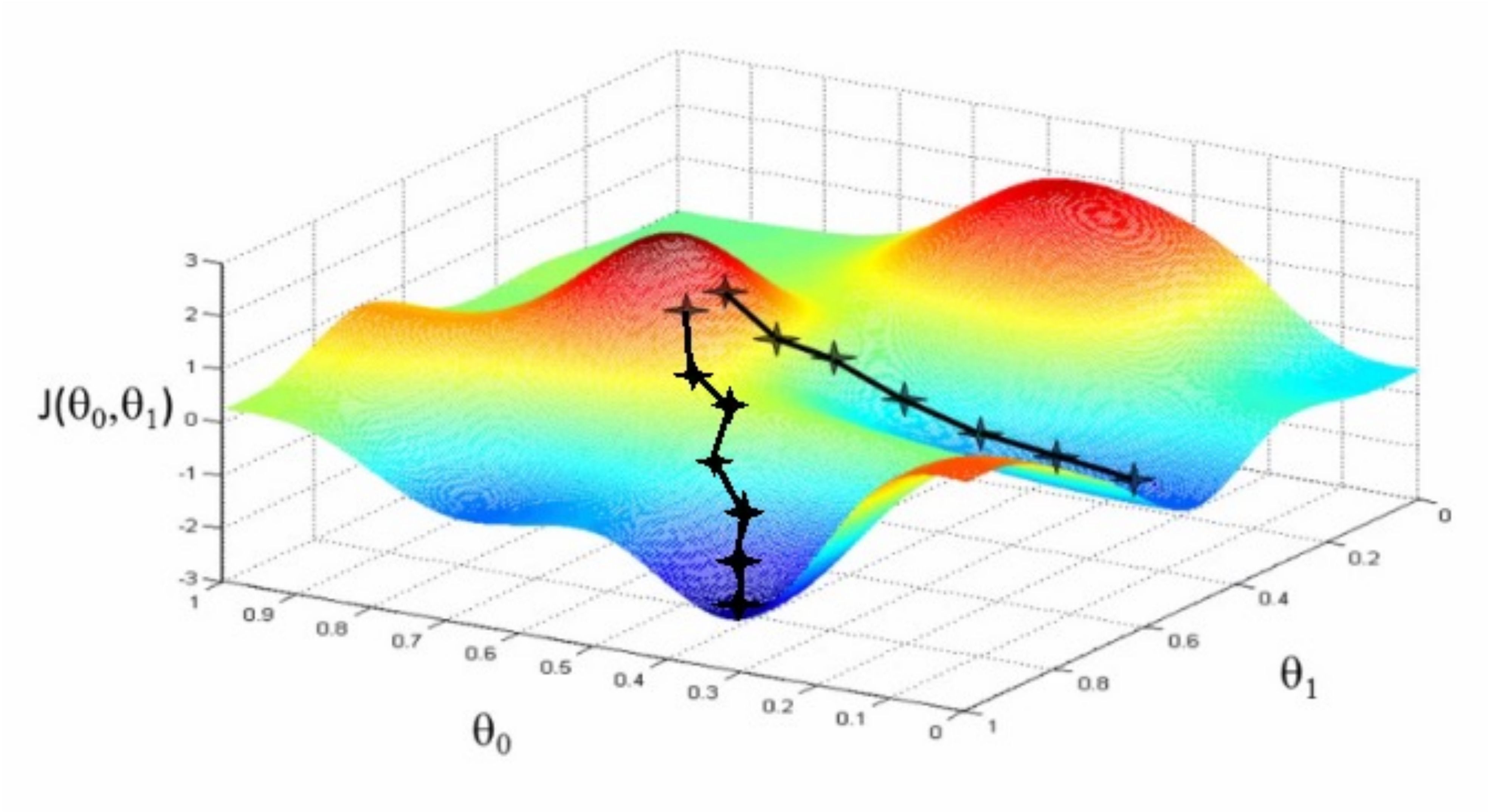
Gradient Descent



2. Compute the gradient (derivative) of L(w) at point w = 12. (e.g. dL/dw = 6)

3. Recompute w as:

w = w – lambda * (dL / dw)



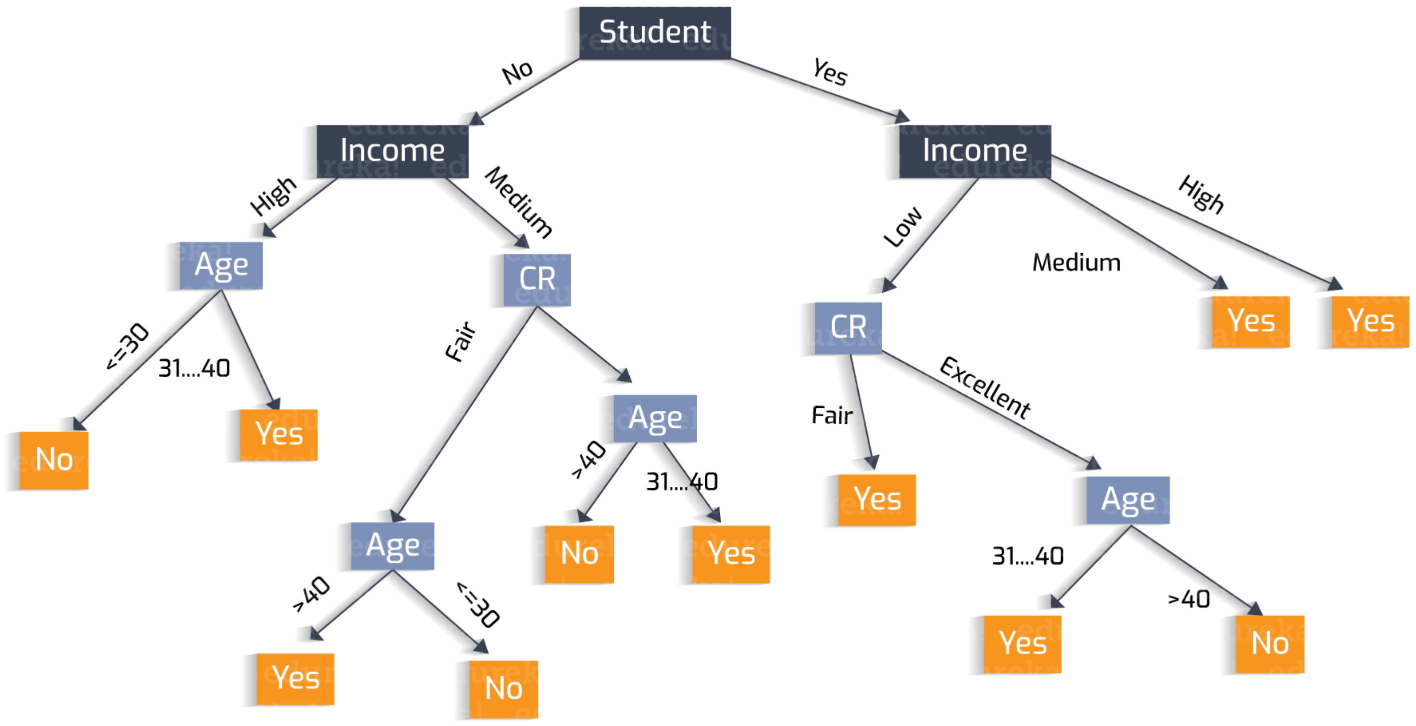
Source: Andrew Ng

In this class we will mostly rely on…

- K-nearest neighbors
- Linear classifiers
- Naïve Bayes classifiers
- Decision Trees
- Random Forests
- Boosted Decision Trees
- Neural Networks

Why?

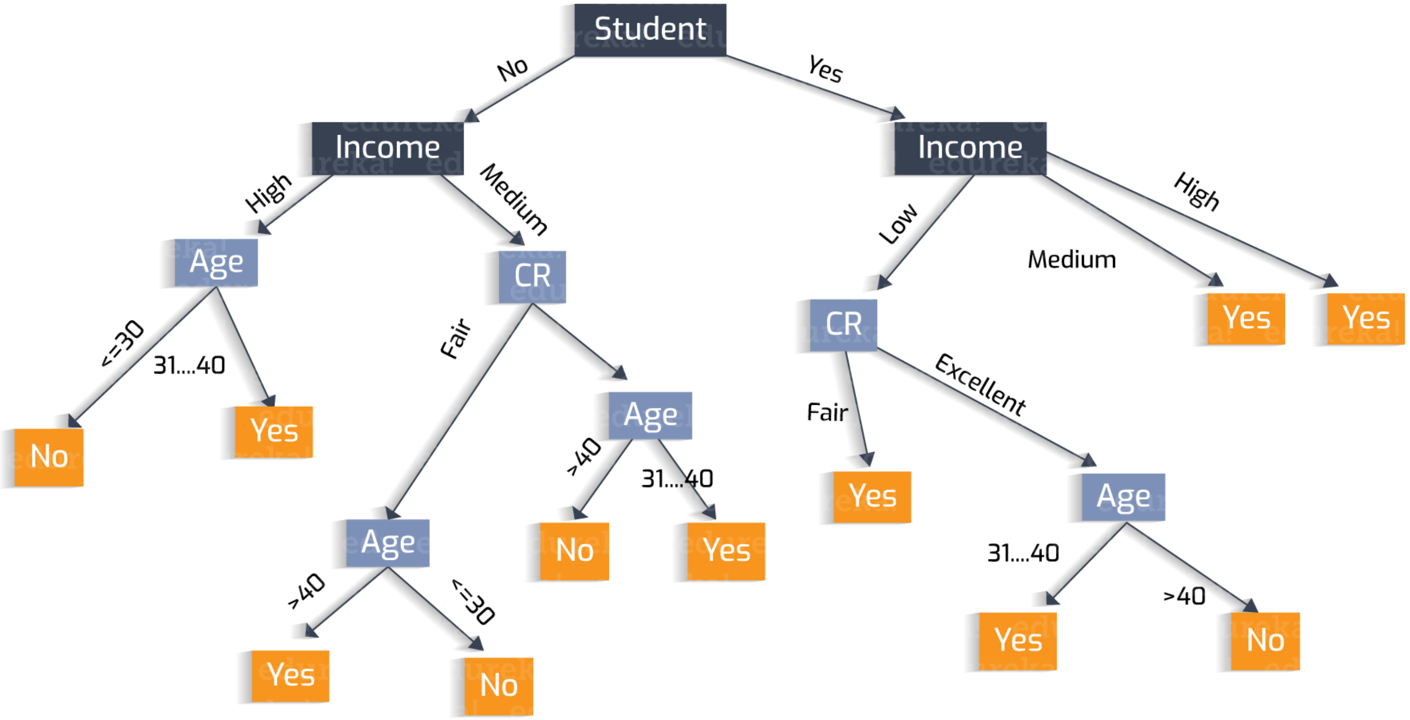
- Decisions Trees



https://heartbeat.fritz.ai/understanding-the-mathematics-behind-decision-trees-22d86d55906 by Nikita Sharma

Why?

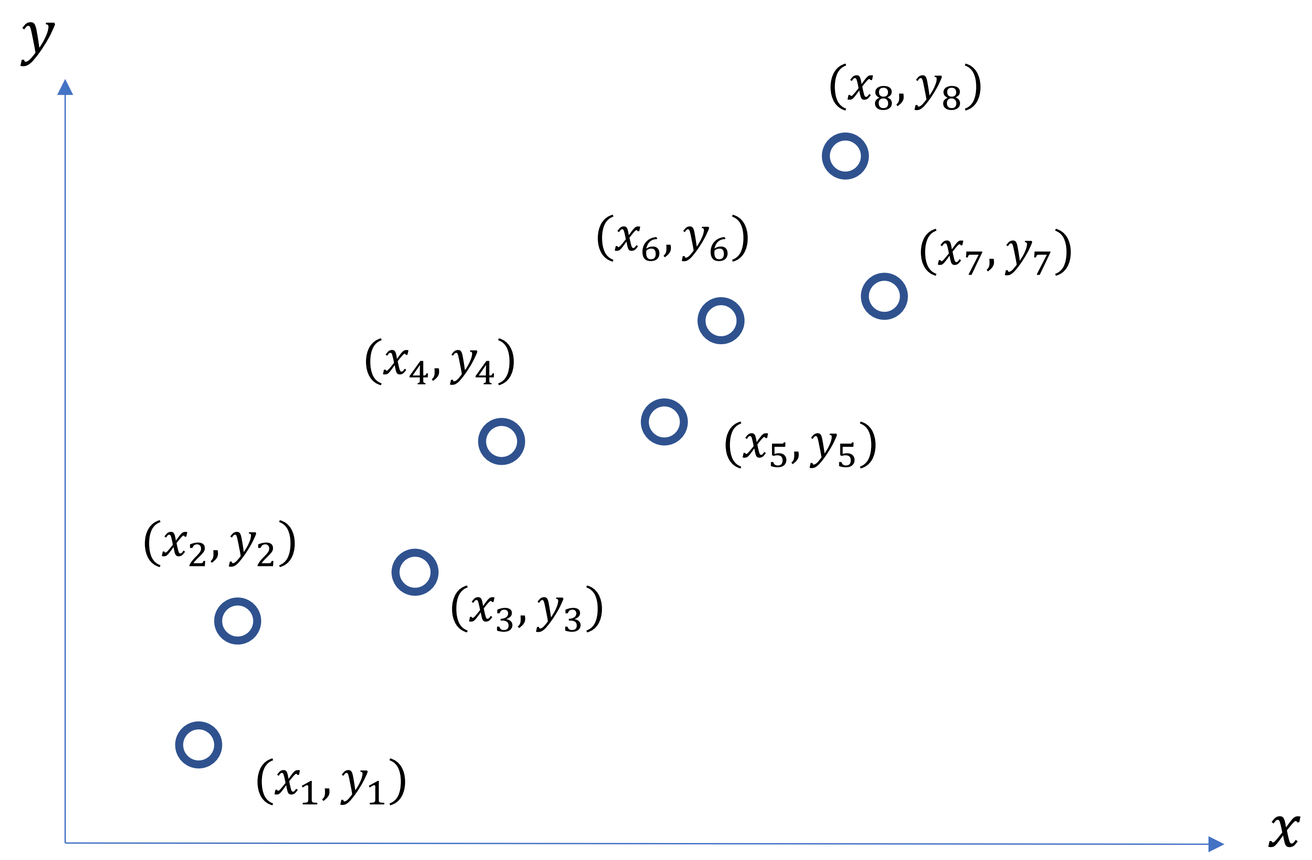
- Decisions Trees are great because they are often interpretable.
- However, they usually deal better with categorical data – not input pixel data.



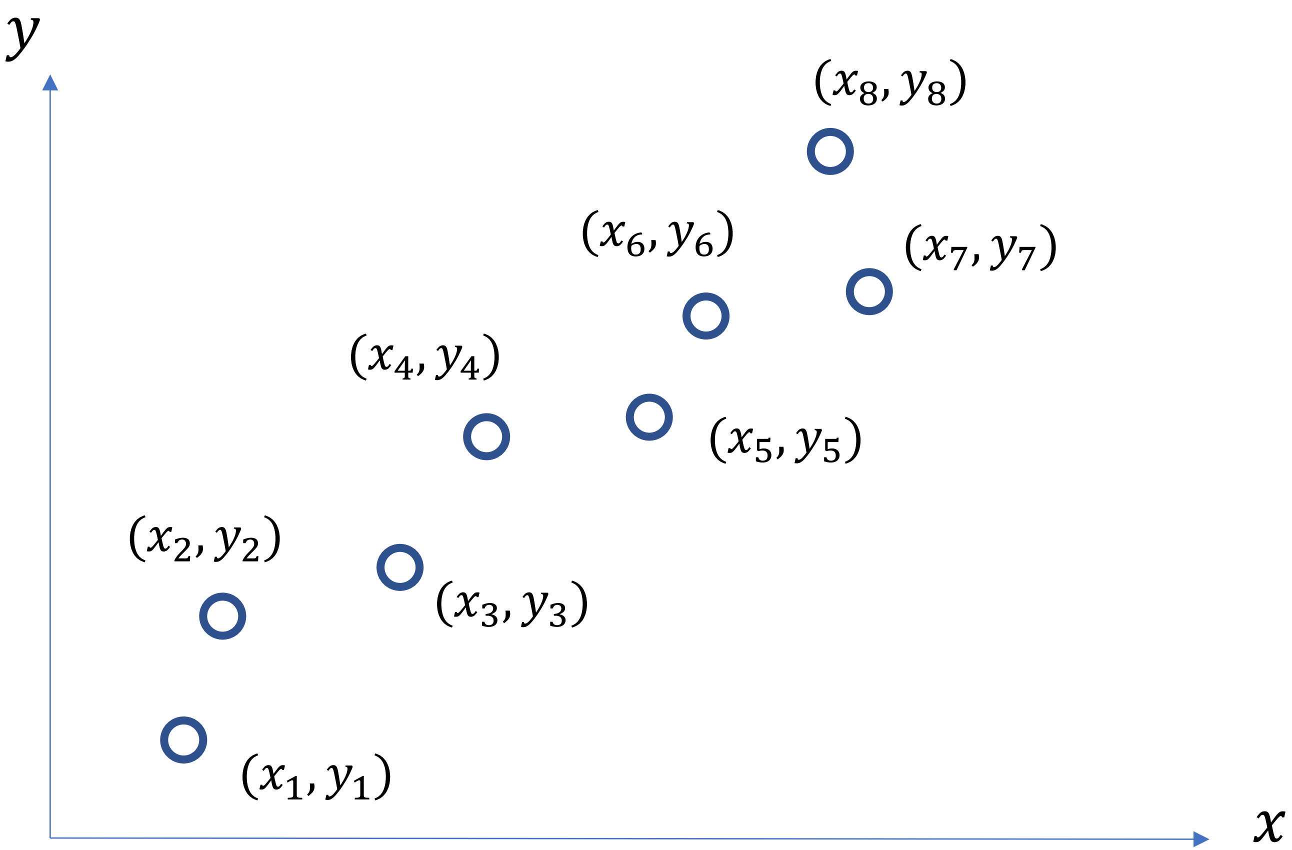
https://heartbeat.fritz.ai/understanding-the-mathematics-behind-decision-trees-22d86d55906 by Nikita Sharma

How to pick the right model?

Linear Regression – 1 output, 1 input

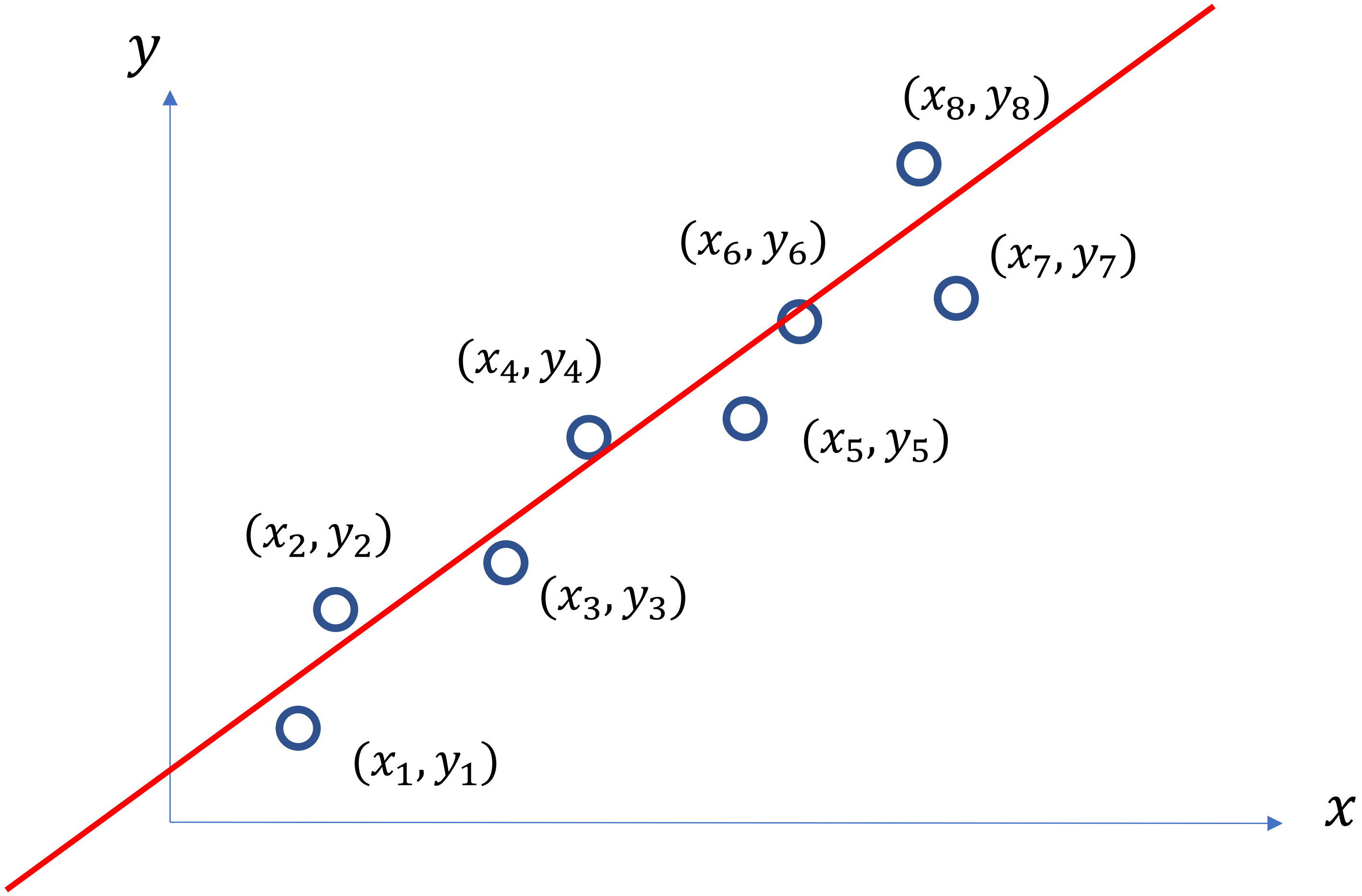


Linear Regression – 1 output, 1 input



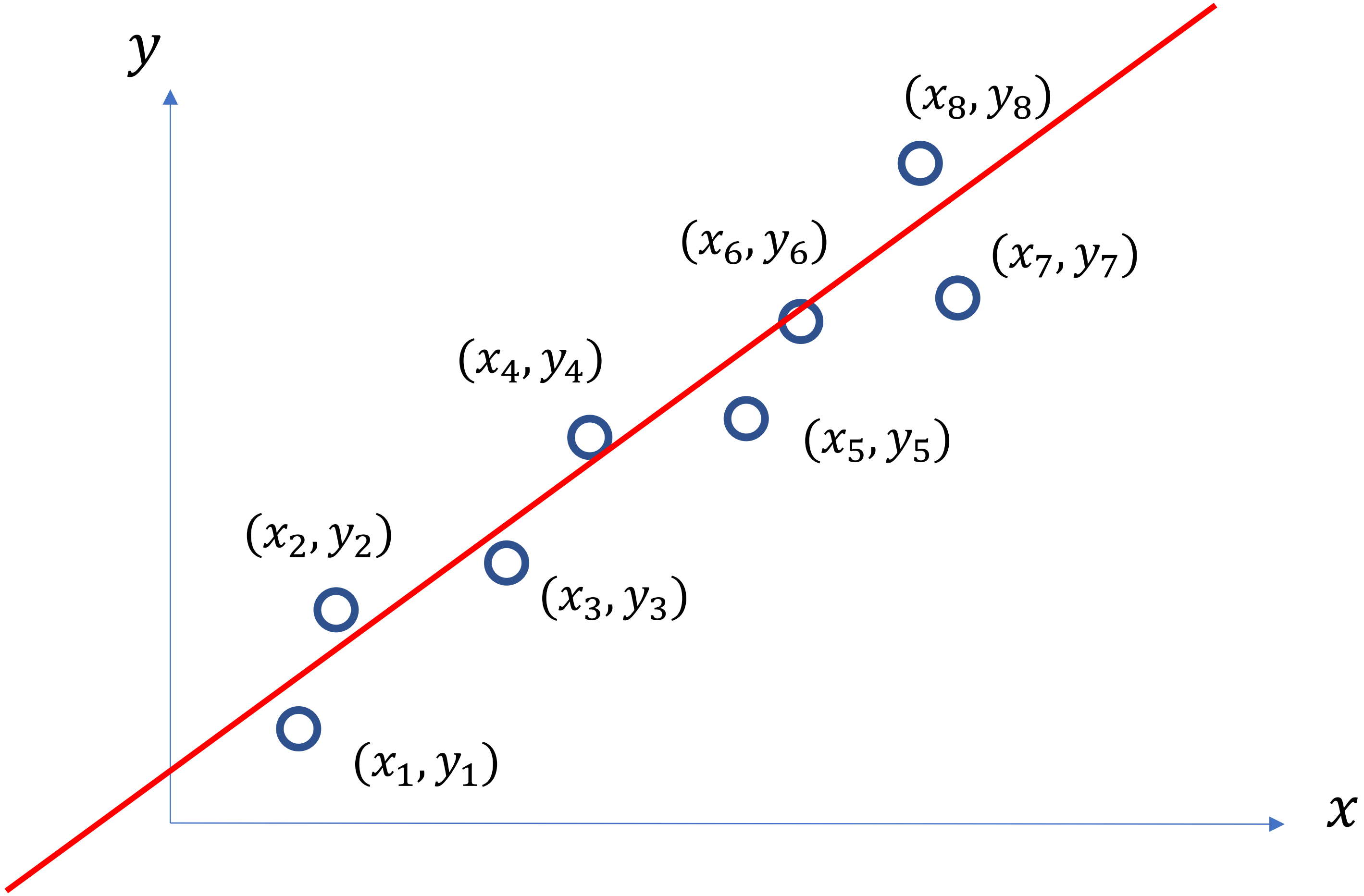
Model: $\hat{y} = wx + b$

Linear Regression – 1 output, 1 input



Model: $\hat{y} = wx + b$

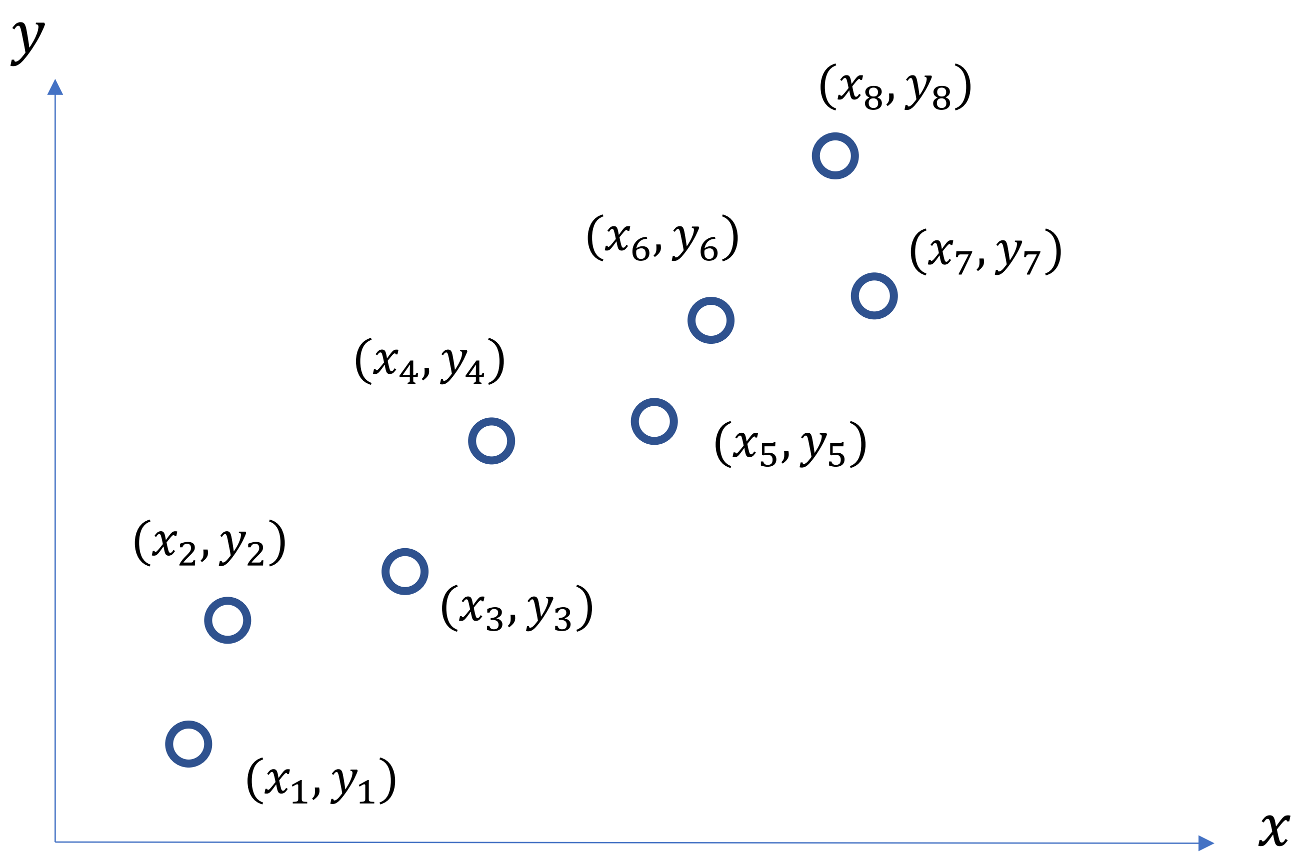
Linear Regression – 1 output, 1 input



Model: $\hat{y} = wx + b$

Loss: $L(w, b) = \sum_{i=1}^{i=8} (\hat{y}_i - y_i)^2$

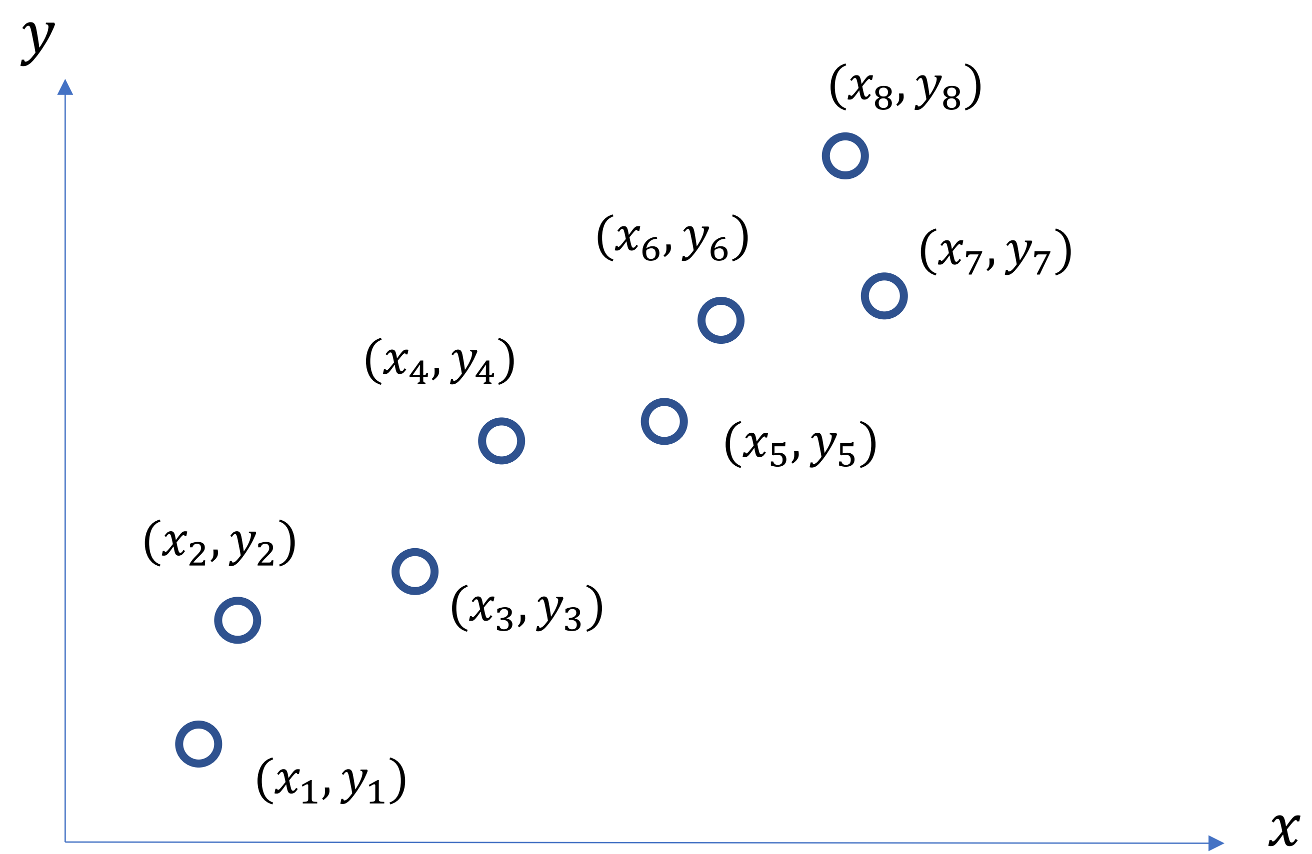
Quadratic Regression



Model: $\hat{y} = w_1 x^2 + w_2 x + b$

Loss: $L(w, b) = \sum_{i=1}^{i=8} (\hat{y}_i - y_i)^2$

n-polynomial Regression

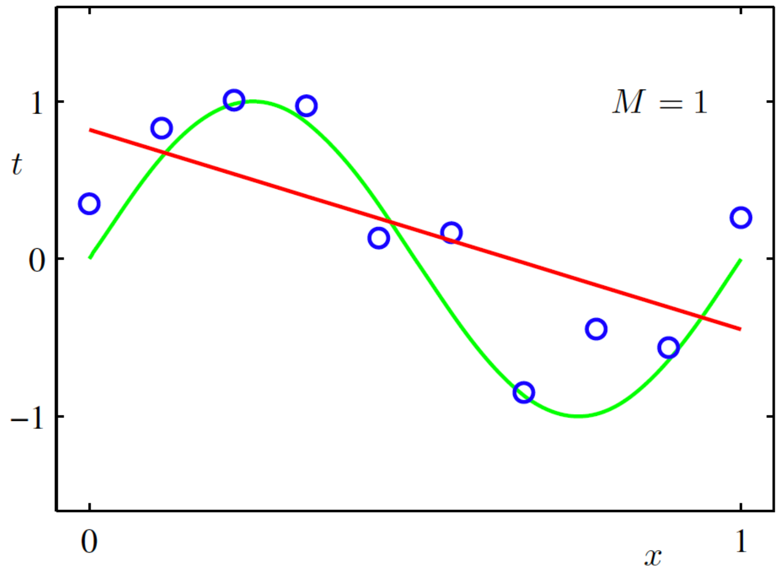


Model: $\hat{y} = w_n x^n + \cdots + w_1 x + b$

Loss: $L(w, b) = \sum_{i=1}^{i=8} (\hat{y}_i - y_i)^2$

Overfitting

f is linear

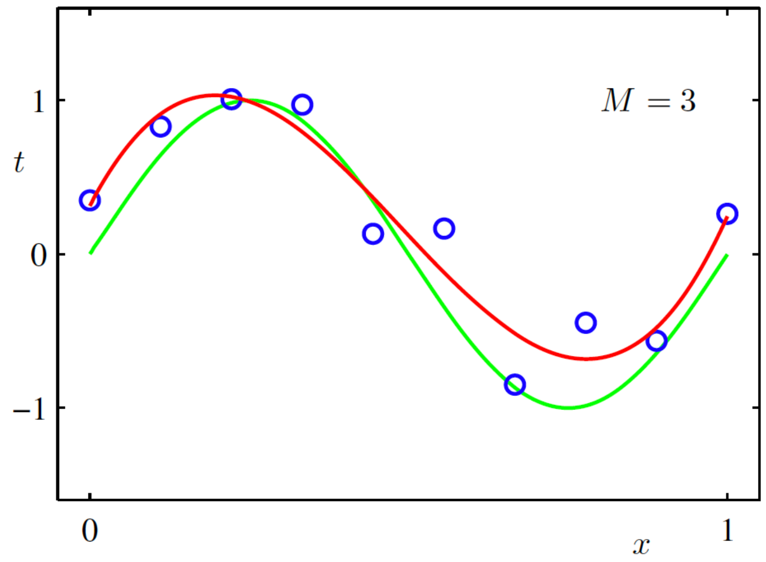


$Loss(w)$ is high

Underfitting

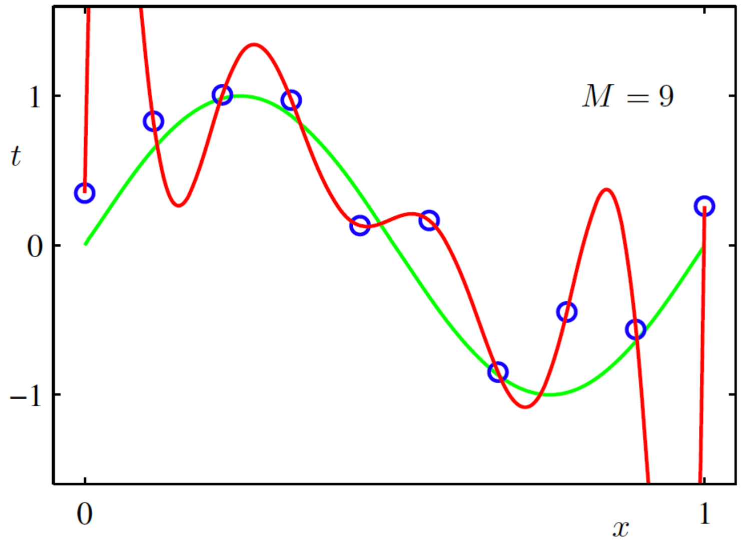
High Bias

f is cubic



$Loss(w)$ is low

f is a polynomial of degree 9



$Loss(w)$ is zero!

Overfitting

High Variance

Christopher M. Bishop – Pattern Recognition and Machine Learning

(mini-batch) Stochastic Gradient Descent (SGD)

$\lambda = 0.01$

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

Initialize w and b randomly

for e = 0, num_epochs **do**

for b = 0, 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:

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

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, num_epochs **do**

for b = 0, num_batches **do**

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

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

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

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, num_epochs **do**

for b = 0, 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, num_epochs **do**

for b = 0, 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, num_epochs **do**

for b = 0, 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

$\lambda = 0.01 \qquad \tau = 0.9$

Initialize w and b randomly

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

global v

for e = 0, num_epochs **do**

for b = 0, num_batches **do**

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

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

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

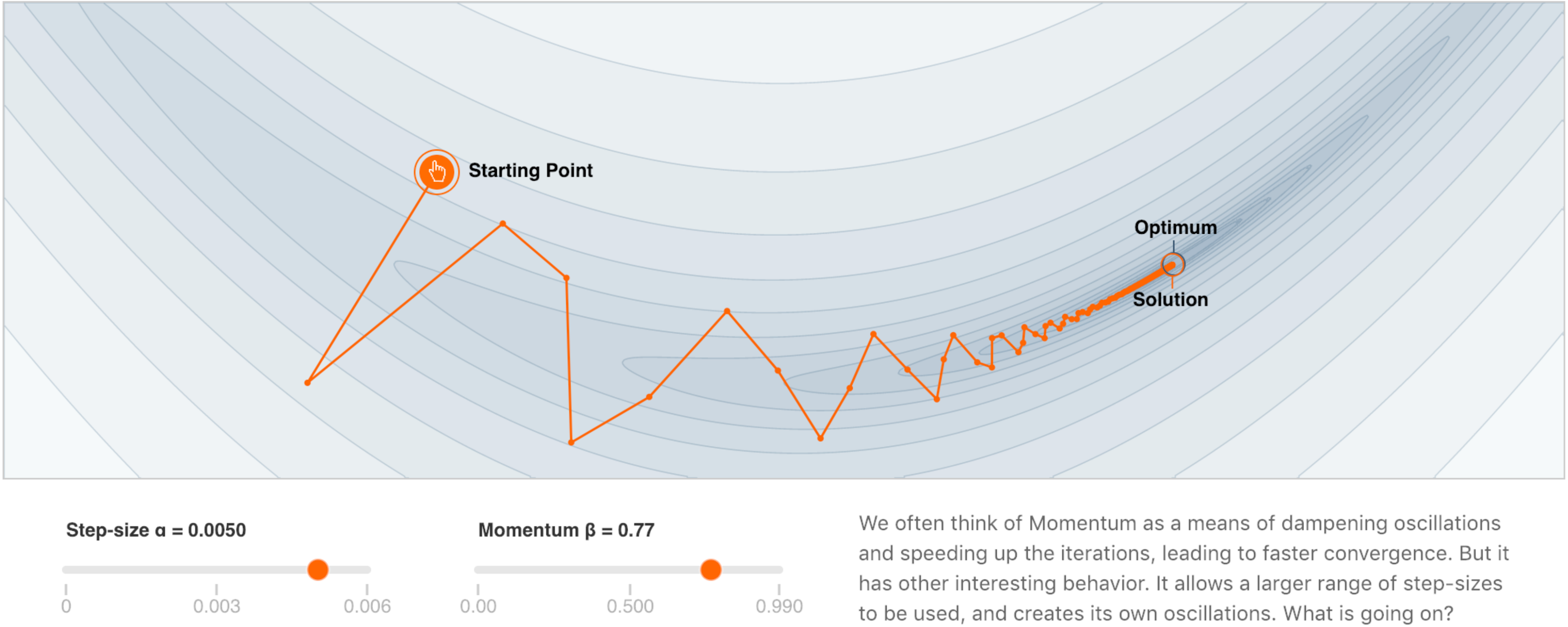
Print: $l(w, b)$ // Useful to see if this is becoming smaller or not.

end

end

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

More on Momentum



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

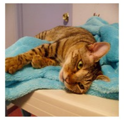

dog

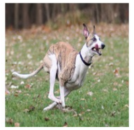

cat

.
.
.


bear

Test Data





.
.
.

Supervised Learning - Classification

Training Data

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

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

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

$\vdots$

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

Supervised Learning - Classification

Training Data

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

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

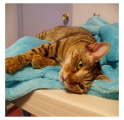
dog

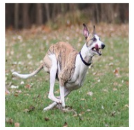
cat

.
.
.

bear

Test Data





.
.
.

Supervised Learning - Classification

Training Data

x_1 = [] y_1 = [cat]

x_2 = [] y_2 = [dog]

x_3 = [] y_3 = [cat]

.
.
.

x_n = [] y_n = [bear]

Supervised Learning - Classification

Training Data

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

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

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

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

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

$\vdots$

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

Supervised Learning – Linear Softmax

Training Data

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

Supervised Learning – Linear Softmax

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

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

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

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

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

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

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

How do we find a good w and b?

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

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:

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

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

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

$$a_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4 + b_c$$
$$a_2 = w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + w_{24}x_4 + b_d$$
$$a_3 = w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + w_{34}x_4 + b_b$$

$$f_1 = e^{a_1}/(e^{a_1}+e^{a_2} + e^{a_3})$$
$$f_2 = e^{a_2}/(e^{a_1}+e^{a_2} + e^{a_3})$$
$$f_3 = e^{a_3}/(e^{a_1}+e^{a_2} + e^{a_3})$$

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:

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

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

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

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!)

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

Computing Analytic Gradients

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

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

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

Computing Analytic Gradients

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

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

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

Let's say: label = 2

We need: $\frac{\partial \ell}{\partial a_1}$ $\frac{\partial \ell}{\partial a_2}$ $\frac{\partial \ell}{\partial a_3}$

Computing Analytic Gradients

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

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

when $i \neq label$:

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

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

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

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

Supervised Learning – Linear Softmax

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

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

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

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

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

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

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

Computing Analytic Gradients

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

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

when $i \neq label$:

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

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

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

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

Computing Analytic Gradients

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

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

when $i = label$:

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

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

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

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

$$\hat{y}_i - 1$$

Computing Analytic Gradients

label = 2

$$\frac{\partial \ell}{\partial a_1} = \hat{y}_1 \qquad \frac{\partial \ell}{\partial a_2} = \hat{y}_2 - 1 \qquad \frac{\partial \ell}{\partial a_3} = \hat{y}_3$$

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

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

Computing Analytic Gradients

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

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

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

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

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

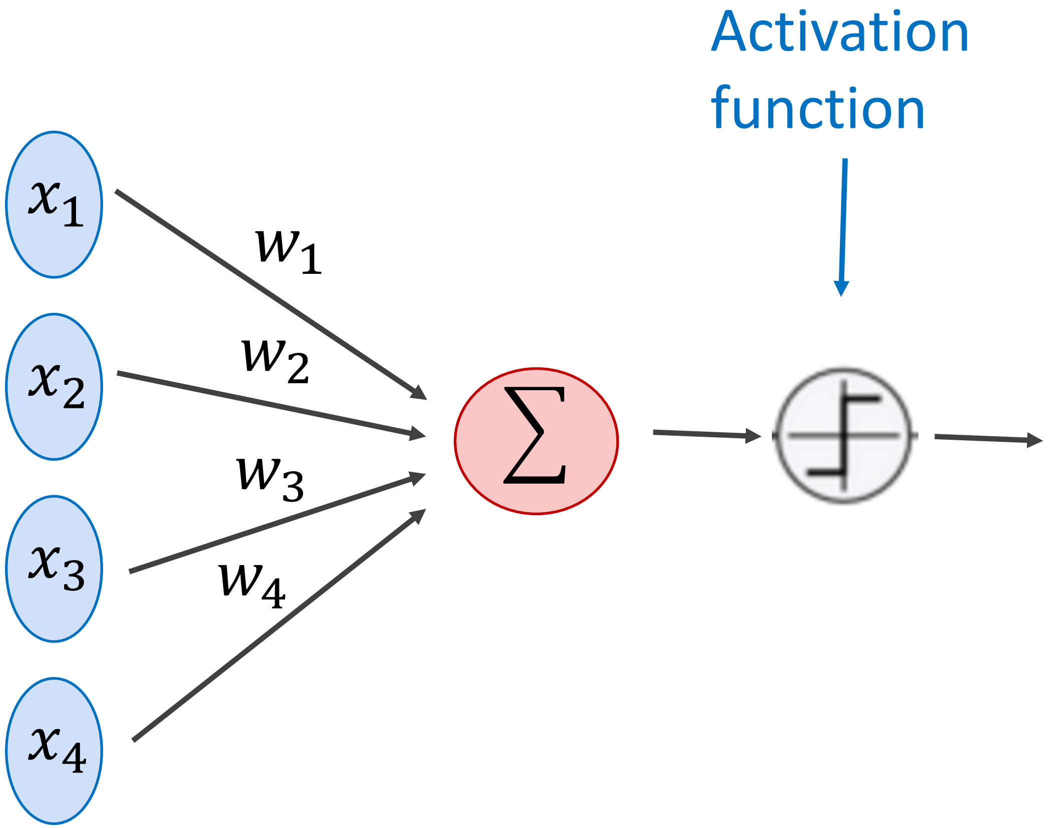
$$\boxed{\frac{\partial \ell}{\partial w_{i,j}} = (\hat{y}_i - y_i)x_j}$$

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

Perceptron Model

Frank Rosenblatt (1957) - Cornell University

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



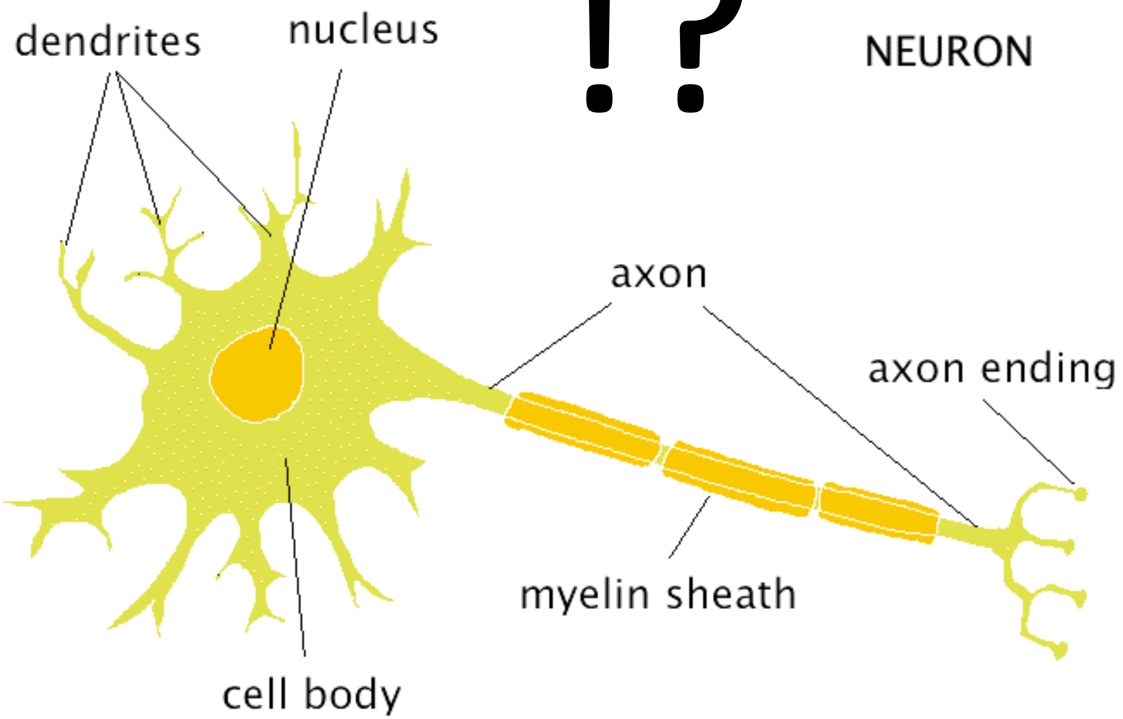
More: https://en.wikipedia.org/wiki/Perceptron

Perceptron Model

Frank Rosenblatt (1957) - Cornell University

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

!?

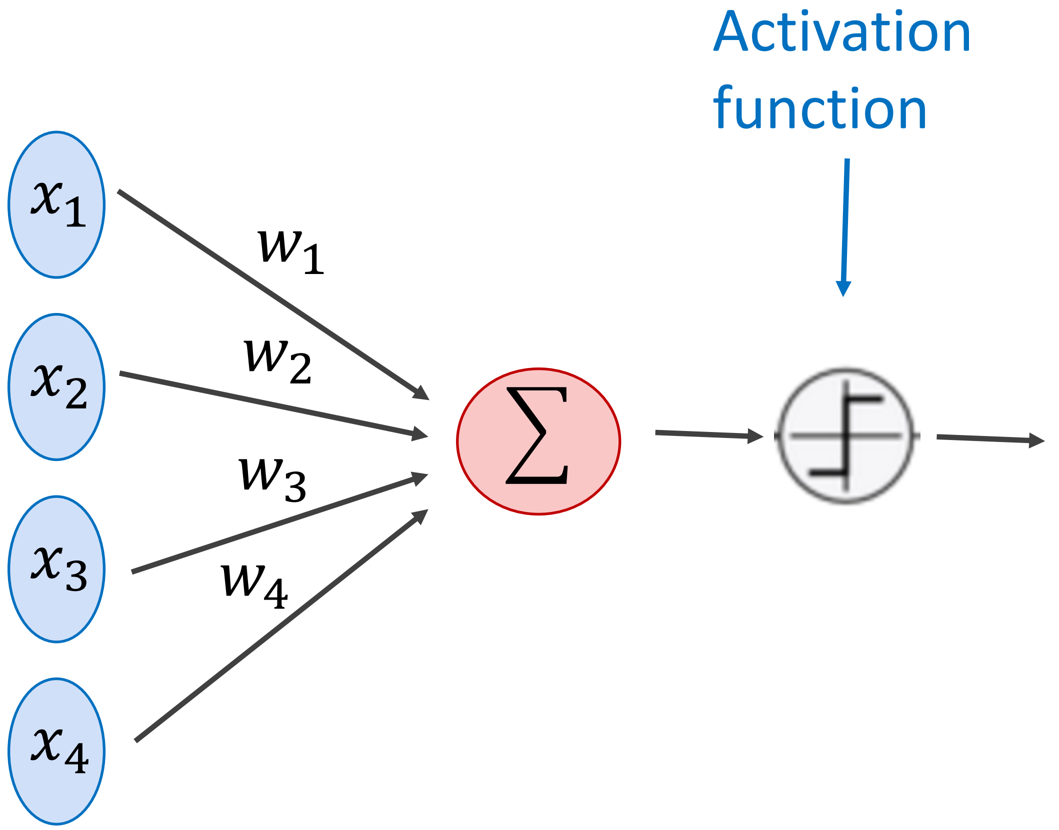


More: https://en.wikipedia.org/wiki/Perceptron

Perceptron Model

Frank Rosenblatt (1957) - Cornell University

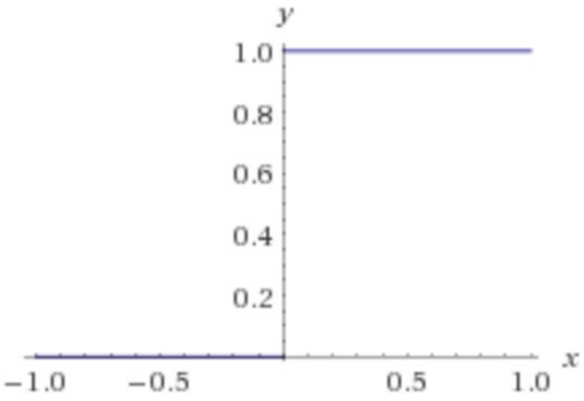
$$f(x) = \begin{cases} 1, & \text{if } \sum_{i=0}^{n} w_i x_i + b > 0 \\ 0, & \text{otherwise} \end{cases}$$



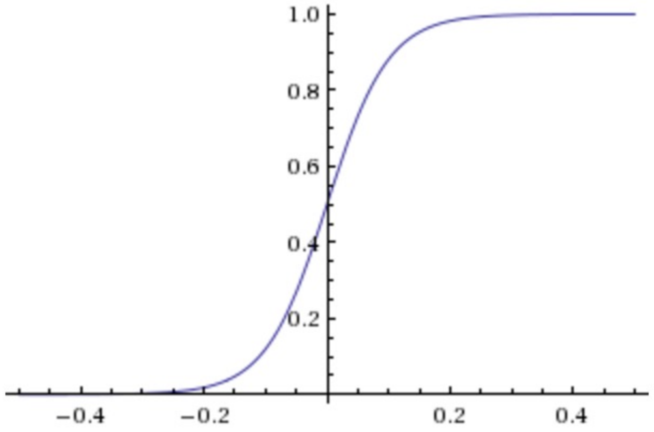
More: https://en.wikipedia.org/wiki/Perceptron

Activation Functions

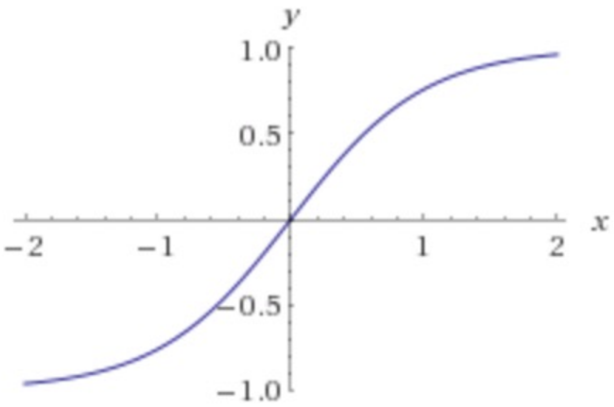
Step(x)



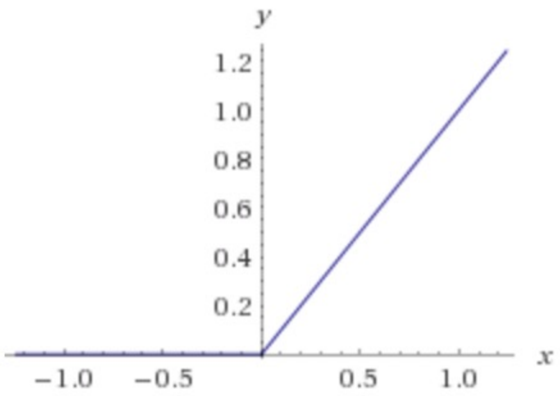
Sigmoid(x)



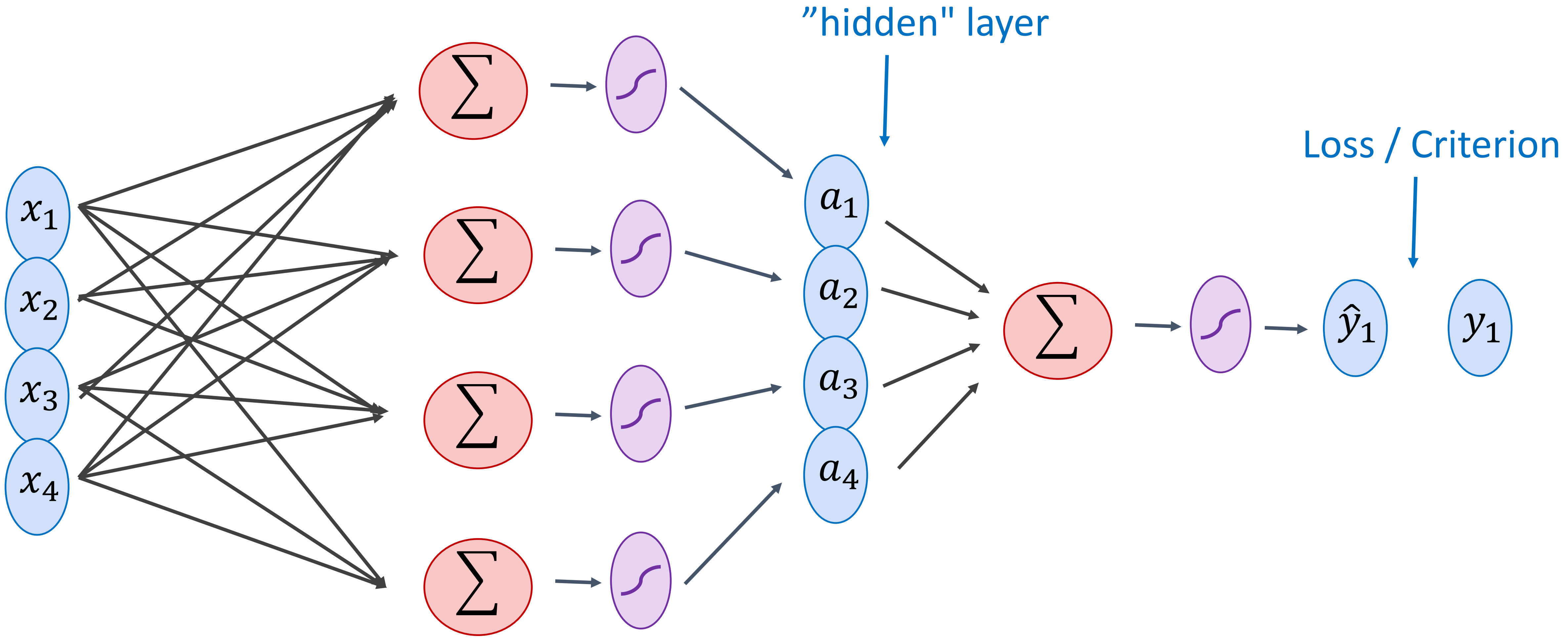
Tanh(x)



ReLU(x) = max(0, x)



Two-layer Multi-layer Perceptron (MLP)



Linear Softmax

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

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

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

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

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

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

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

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

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

Linear Softmax

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

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

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

$$g_c = w_{c1}x_{i1} + w_{c2}x_{i2} + w_{c3}x_{i3} + w_{c4}x_{i4} + b_c$$
$$g_d = w_{d1}x_{i1} + w_{d2}x_{i2} + w_{d3}x_{i3} + w_{d4}x_{i4} + b_d$$
$$g_b = w_{b1}x_{i1} + w_{b2}x_{i2} + w_{b3}x_{i3} + w_{b4}x_{i4} + b_b$$

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

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

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

Linear Softmax

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

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

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

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

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

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

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

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

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

Linear Softmax

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

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

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

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

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

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

$$f = softmax(g)$$

Linear Softmax

$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]$ $y_i = [1 \quad 0 \quad 0]$ $\hat{y}_i = [f_c \quad f_d \quad f_b]$

$$f = softmax(wx^T + b^T)$$

Two-layer MLP + Softmax

$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]$ $y_i = [1 \quad 0 \quad 0]$ $\hat{y}_i = [f_c \quad f_d \quad f_b]$

$$a_1 = sigmoid(w_{[1]}x^T + b_{[1]}^T)$$
$$f = softmax(w_{[2]}a_1{}^T + b_{[2]}^T)$$

N-layer MLP + Softmax

$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]$ $y_i =$ [1 0 0] $\hat{y}_i = [f_c \quad f_d \quad f_b]$

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

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

...

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

...

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

How to train the parameters?

$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]$ $y_i = [1 \quad 0 \quad 0]$ $\hat{y}_i = [f_c \quad f_d \quad f_b]$

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

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

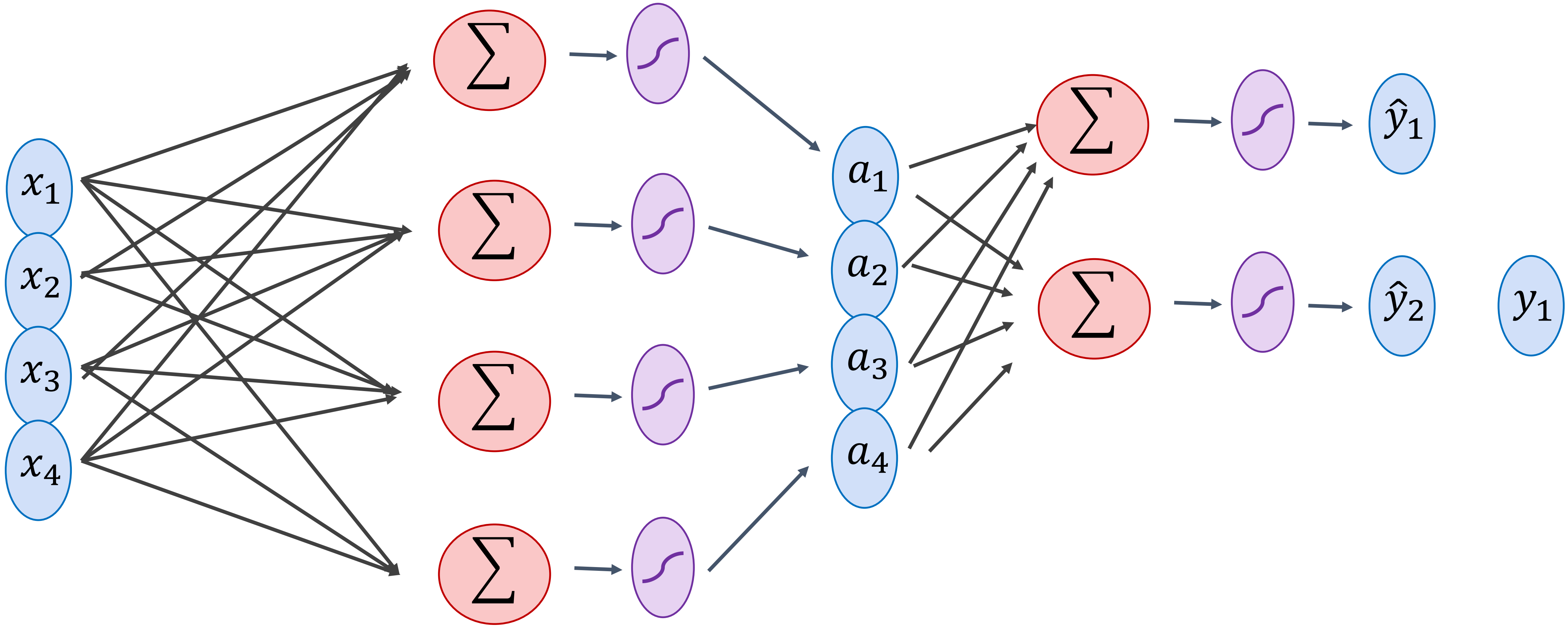
...

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

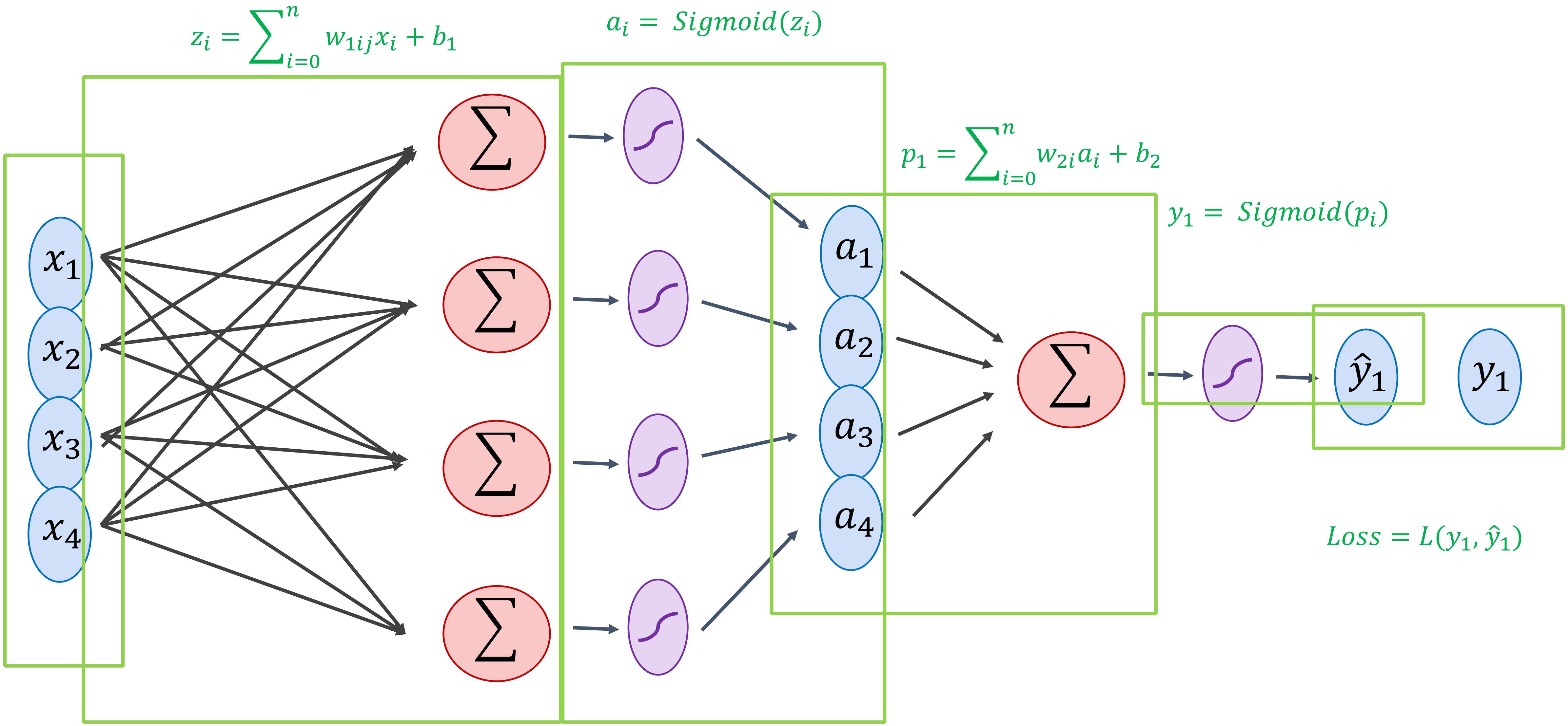
...

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

Forward pass (Forward-propagation)



Forward pass (Forward-propagation)



How to train the parameters?

$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]$ $\qquad$ $y_i = [1 \quad 0 \quad 0]$ $\qquad$ $\hat{y}_i = [f_c \quad f_d \quad f_b]$

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

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

...

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

...

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

We can still use SGD

We need!

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

How to train the parameters?

$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]$ $y_i = \ [1 \quad 0 \quad 0]$ $\hat{y}_i = \ [f_c \quad f_d \quad f_b]$

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

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

$$\cdots$$

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

$$\cdots$$

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

$$l = loss(f, y)$$

We can still use SGD

We need!

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

How to train the parameters?

$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]$ $y_i = [1 \quad 0 \quad 0]$ $\hat{y}_i = [f_c \quad f_d \quad f_b]$

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

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

...

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

...

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

$$l = loss(f, y)$$

We can still use SGD

We need!

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

How to train the parameters?

$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]$ $\quad y_i = [1 \quad 0 \quad 0]$ $\quad \hat{y}_i = [f_c \quad f_d \quad f_b]$

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

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

$$\dots$$

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

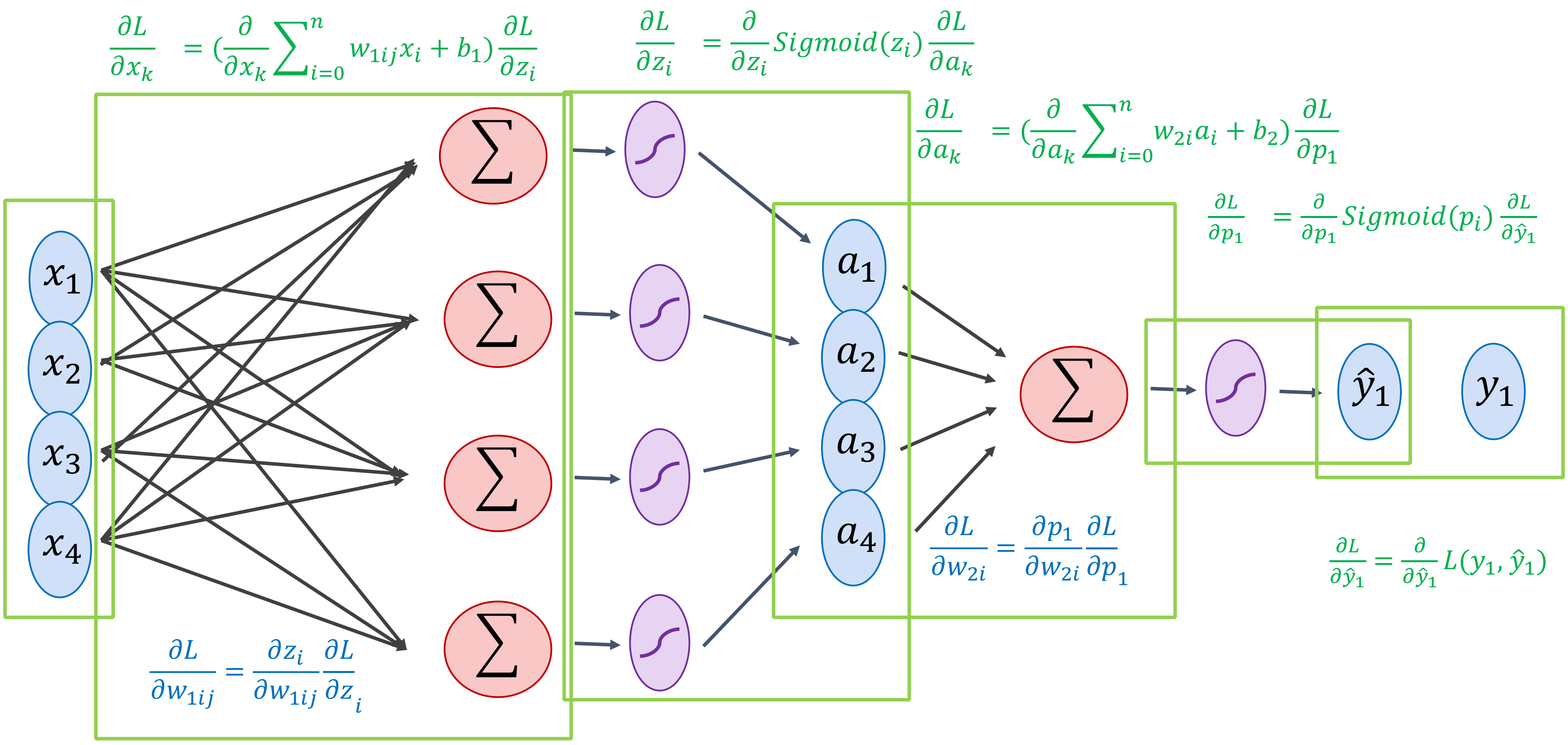
$$\dots$$

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

$$l = loss(f, y)$$

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

Backward pass (Back-propagation)



Questions?