

Logistic regression and simple multi-layer neural networks

Roger Levy

9.19: Computational Psycholinguistics

2 November 2023

Agenda for the day

- Review logistic regression (case study: *binomial ordering preferences*)
- Limitations of linear classifiers like logistic regression
- Basic multi-layer neural networks & backpropagation
- Expressing and learning solutions to non-linear classification problems
- Vanishing gradients and activation functions

Recap: binomial ordering preferences

- In each pair, which phrase sounds more natural?

Recap: binomial ordering preferences

- In each pair, which phrase sounds more natural?

pepper and salt

salt and pepper

Recap: binomial ordering preferences

- In each pair, which phrase sounds more natural?

pepper and salt

salt and pepper

hit and run

run and hit

Recap: binomial ordering preferences

- In each pair, which phrase sounds more natural?

pepper and salt

salt and pepper

hit and run

run and hit

gold and silver

silver and gold

Recap: binomial ordering preferences

- In each pair, which phrase sounds more natural?

pepper and salt

salt and pepper

hit and run

run and hit

gold and silver

silver and gold

deer and trees

trees and deer

Recap: binomial ordering preferences

- In each pair, which phrase sounds more natural?

pepper and salt

salt and pepper

hit and run

run and hit

gold and silver

silver and gold

deer and trees

trees and deer

drink and food

food and drink

Recap: binomial ordering preferences

- In each pair, which phrase sounds more natural?

pepper and salt

salt and pepper

hit and run

run and hit

gold and silver

silver and gold

deer and trees

trees and deer

drink and food

food and drink

skirts and sweaters

sweaters and skirts

Recap: binomial ordering preferences

- In each pair, which phrase sounds more natural?

pepper and salt

salt and pepper

hit and run

run and hit

gold and silver

silver and gold

deer and trees

trees and deer

drink and food

food and drink

skirts and sweaters

sweaters and skirts

bishops and seamstresses

seamstresses and bishops

Recap: binomial ordering preferences

- In each pair, which phrase sounds more natural?

pepper and salt

salt and pepper

hit and run

run and hit

gold and silver

silver and gold

deer and trees

trees and deer

drink and food

food and drink

skirts and sweaters

sweaters and skirts

bishops and seamstresses

seamstresses and bishops

few and unfavorable

unfavorable and few

Recap: binomial ordering preferences

- In each pair, which phrase sounds more natural?

pepper and salt

salt and pepper

hit and run

run and hit

gold and silver

silver and gold

deer and trees

trees and deer

drink and food

food and drink

skirts and sweaters

sweaters and skirts

bishops and seamstresses

seamstresses and bishops

few and unfavorable

unfavorable and few

cat and mouse

mouse and cat

Recap: binomial ordering preferences

- In each pair, which phrase sounds more natural?

pepper and salt

salt and pepper

hit and run

run and hit

gold and silver

silver and gold

deer and trees

trees and deer

drink and food

food and drink

skirts and sweaters

sweaters and skirts

bishops and seamstresses

seamstresses and bishops

few and unfavorable

unfavorable and few

cat and mouse

mouse and cat

quilting and sewing

sewing and quilting

Recap: binomial ordering preferences

- In each pair, which phrase sounds more natural?

pepper and salt

salt and pepper

hit and run

run and hit

gold and silver

silver and gold

deer and trees

trees and deer

drink and food

food and drink

skirts and sweaters

sweaters and skirts

bishops and seamstresses

seamstresses and bishops

few and unfavorable

unfavorable and few

cat and mouse

mouse and cat

quilting and sewing

sewing and quilting

interest and principal

principal and interest

Multiple, cross-cutting constraints

Constraint	Example	Strength
Iconic/scalar sequencing	<i>open and read</i>	20
Perceptual markedness	<i>deer and trees</i>	1.7
Formal markedness	<i>change and improve</i>	1.4
Power	<i>food and drink</i>	1
Avoid final stress	<i>confuse and disorient</i>	0.5
Short<Long	<i>cruel and unusual</i>	0.4
Frequent<Infrequent	<i>neatly and sweetly</i>	0.3

Multiple, cross-cutting constraints

Constraint	Example	Strength
Iconic/scalar sequencing	<i>open and read</i>	20
Perceptual markedness	<i>deer and trees</i>	1.7
Formal markedness	<i>change and improve</i>	1.4
Power	<i>food and drink</i>	1
Avoid final stress	<i>confuse and disorient</i>	0.5
Short<Long	<i>cruel and unusual</i>	0.4
Frequent<Infrequent	<i>neatly and sweetly</i>	0.3

- **Logistic regression** to capture effects on ordering preference:

$$\eta = \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_N X_N$$

Multiple, cross-cutting constraints

	Constraint	Example	Strength
{ X_i }	Iconic/scalar sequencing	<i>open and read</i>	20
	Perceptual markedness	<i>deer and trees</i>	1.7
	Formal markedness	<i>change and improve</i>	1.4
	Power	<i>food and drink</i>	1
	Avoid final stress	<i>confuse and disorient</i>	0.5
	Short<Long	<i>cruel and unusual</i>	0.4
	Frequent<Infrequent	<i>neatly and sweetly</i>	0.3

- **Logistic regression** to capture effects on ordering preference:

$$\eta = \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_N X_N$$

Multiple, cross-cutting constraints

	Constraint	Example	Strength
{ X_i }	Iconic/scalar sequencing	<i>open and read</i>	20
	Perceptual markedness	<i>deer and trees</i>	1.7
	Formal markedness	<i>change and improve</i>	1.4
	Power	<i>food and drink</i>	1
	Avoid final stress	<i>confuse and disorient</i>	0.5
	Short<Long	<i>cruel and unusual</i>	0.4
	Frequent<Infrequent	<i>neatly and sweetly</i>	0.3

{ β_i }

- **Logistic regression** to capture effects on ordering preference:

$$\eta = \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_N X_N$$

Multiple, cross-cutting constraints

	Constraint	Example	Strength	
{ X_i }	Iconic/scalar sequencing	<i>open and read</i>	20	{ β_i }
	Perceptual markedness	<i>deer and trees</i>	1.7	
	Formal markedness	<i>change and improve</i>	1.4	
	Power	<i>food and drink</i>	1	
	Avoid final stress	<i>confuse and disorient</i>	0.5	
	Short<Long	<i>cruel and unusual</i>	0.4	
	Frequent<Infrequent	<i>neatly and sweetly</i>	0.3	

- **Logistic regression** to capture effects on ordering preference:

$$\eta = \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_N X_N$$

“goodness score”

Multiple, cross-cutting constraints

	Constraint	Example	Strength	
$\{X_i\}$	Iconic/scalar sequencing	<i>open and read</i>	20	$\{\beta_i\}$
	Perceptual markedness	<i>deer and trees</i>	1.7	
	Formal markedness	<i>change and improve</i>	1.4	
	Power	<i>food and drink</i>	1	
	Avoid final stress	<i>confuse and disorient</i>	0.5	
	Short<Long	<i>cruel and unusual</i>	0.4	
	Frequent<Infrequent	<i>neatly and sweetly</i>	0.3	

- **Logistic regression** to capture effects on ordering preference:

$$\eta = \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_N X_N$$

“goodness score”

$$P(\text{“success”}) = \frac{e^\eta}{1 + e^\eta}$$

Multiple, cross-cutting constraints

	Constraint	Example	Strength
{ X_i }	Iconic/scalar sequencing	<i>open and read</i>	20
	Perceptual markedness	<i>deer and trees</i>	1.7
	Formal markedness	<i>change and improve</i>	1.4
	Power	<i>food and drink</i>	1
	Avoid final stress	<i>confuse and disorient</i>	0.5
	Short<Long	<i>cruel and unusual</i>	0.4
	Frequent<Infrequent	<i>neatly and sweetly</i>	0.3

{ β_i }

- **Logistic regression** to capture effects on ordering preference:

$$\eta = \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_N X_N$$

“goodness score”

$$P(\text{“success”}) = \frac{e^\eta}{1 + e^\eta}$$

a.k.a. **mean μ**

Multiple, cross-cutting constraints

	Constraint	Example	Strength
{ X_i }	Iconic/scalar sequencing	<i>open and read</i>	20
	Perceptual markedness	<i>deer and trees</i>	1.7
	Formal markedness	<i>change and improve</i>	1.4
	Power	<i>food and drink</i>	1
	Avoid final stress	<i>confuse and disorient</i>	0.5
	Short<Long	<i>cruel and unusual</i>	0.4
	Frequent<Infrequent	<i>neatly and sweetly</i>	0.3

{ β_i }

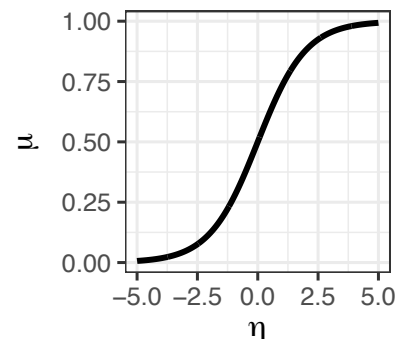
- **Logistic regression** to capture effects on ordering preference:

$$\eta = \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_N X_N$$

“goodness score”

$$P(\text{“success”}) = \frac{e^\eta}{1 + e^\eta}$$

a.k.a. mean μ



Multiple, cross-cutting constraints

	Constraint	Example	Strength
{ X_i }	Iconic/scalar sequencing	<i>open and read</i>	20
	Perceptual markedness	<i>deer and trees</i>	1.7
	Formal markedness	<i>change and improve</i>	1.4
	Power	<i>food and drink</i>	1
	Avoid final stress	<i>confuse and disorient</i>	0.5
	Short<Long	<i>cruel and unusual</i>	0.4
	Frequent<Infrequent	<i>neatly and sweetly</i>	0.3

{ β_i }

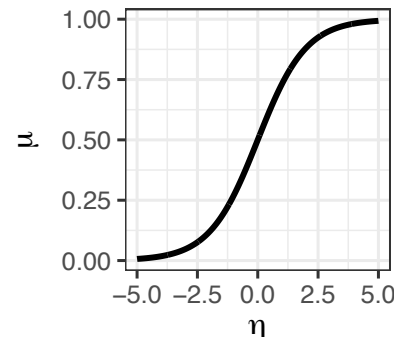
- **Logistic regression** to capture effects on ordering preference:

$$\eta = \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_N X_N$$

“goodness score”

$$P(\text{“success”}) = \frac{e^\eta}{1 + e^\eta}$$

a.k.a. mean μ



Logistic (sigmoid)
activation function

A two-constraint example

A two-constraint example

- Constraints: word **length** (# syllables) and word **frequency**

A two-constraint example

- Constraints: word **length** (# syllables) and word **frequency**

$$\eta = \beta_{Syl} X_{Syl} + \beta_{Freq} X_{Freq}$$

A two-constraint example

- Constraints: word **length** (# syllables) and word **frequency**

$$\eta = \beta_{Syl} X_{Syl} + \beta_{Freq} X_{Freq}$$

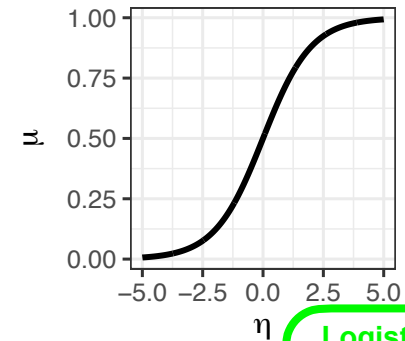
$$P(\text{“success”}) = \frac{e^{\eta}}{1 + e^{\eta}}$$

A two-constraint example

- Constraints: word **length** (# syllables) and word **frequency**

$$\eta = \beta_{Syl} X_{Syl} + \beta_{Freq} X_{Freq}$$

$$P(\text{“success”}) = \frac{e^{\eta}}{1 + e^{\eta}}$$



Logistic (sigmoid)
activation function

A two-constraint example

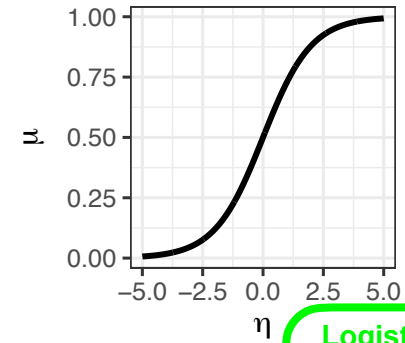
- Constraints: word **length** (# syllables) and word **frequency**

$$\eta = \beta_{Syl} X_{Syl} + \beta_{Freq} X_{Freq}$$

$$P(\text{"success"}) = \frac{e^{\eta}}{1 + e^{\eta}}$$

Arbitrarily define:

"success" ↔ alphabetical ordering



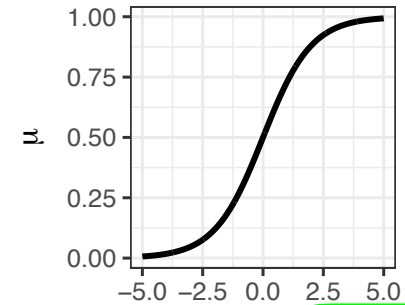
Logistic (sigmoid)
activation function

A two-constraint example

- Constraints: word **length** (# syllables) and word **frequency**

$$\eta = \beta_{Syl} X_{Syl} + \beta_{Freq} X_{Freq}$$

$$P(\text{"success"}) = \frac{e^{\eta}}{1 + e^{\eta}}$$



Logistic (sigmoid) activation function

Arbitrarily define:

"success" ↔ alphabetical ordering

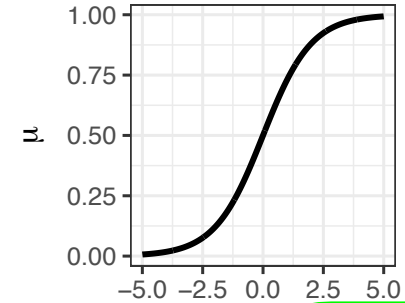
	Short < Long	Freq < Infreq
<i>calm and relaxed</i>	✓	✓
<i>big and thick</i>	n/a	✓
<i>down and out</i>	n/a	✗
<i>cruel and unusual</i>	✓	✗
<i>anger and spite</i>	✗	✓
<i>crochet and knit</i>	✗	✗

A two-constraint example

- Constraints: word **length** (# syllables) and word **frequency**

$$\eta = \beta_{Syl} X_{Syl} + \beta_{Freq} X_{Freq}$$

$$P(\text{"success"}) = \frac{e^\eta}{1 + e^\eta}$$



Logistic (sigmoid) activation function

Arbitrarily define:

"success" ↔ alphabetical ordering

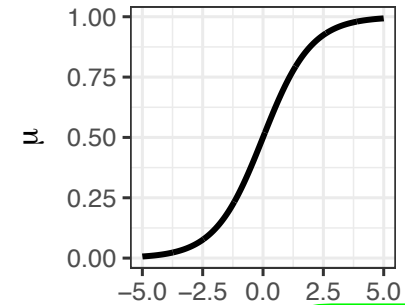
	Short < Long	X_{Syl}	Freq < Infreq
<i>calm and relaxed</i>	✓	1	✓
<i>big and thick</i>	n/a	0	✓
<i>down and out</i>	n/a	0	✗
<i>cruel and unusual</i>	✓	1	✗
<i>anger and spite</i>	✗	-1	✓
<i>crochet and knit</i>	✗	-1	✗

A two-constraint example

- Constraints: word **length** (# syllables) and word **frequency**

$$\eta = \beta_{Syl} X_{Syl} + \beta_{Freq} X_{Freq}$$

$$P(\text{"success"}) = \frac{e^{\eta}}{1 + e^{\eta}}$$



Logistic (sigmoid) activation function

Arbitrarily define:

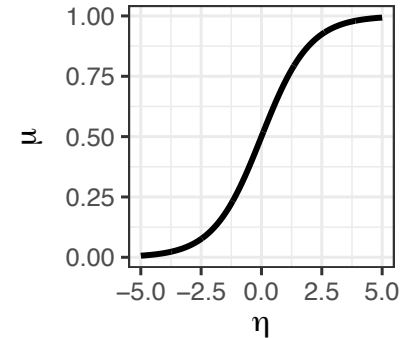
"success" ↔ alphabetical ordering

	Short < Long	X_{Syl}	Freq < Infreq	X_{Freq}
<i>calm and relaxed</i>	✓	1	✓	1
<i>big and thick</i>	n/a	0	✓	1
<i>down and out</i>	n/a	0	✗	-1
<i>cruel and unusual</i>	✓	1	✗	-1
<i>anger and spite</i>	✗	-1	✓	1
<i>crochet and knit</i>	✗	-1	✗	-1

Learning constraint weights

$$\eta = \beta_{Syl} X_{Syl} + \beta_{Freq} X_{Freq}$$

$$P(\text{“success”}) = \frac{e^\eta}{1 + e^\eta}$$



Logistic (sigmoid) activation function

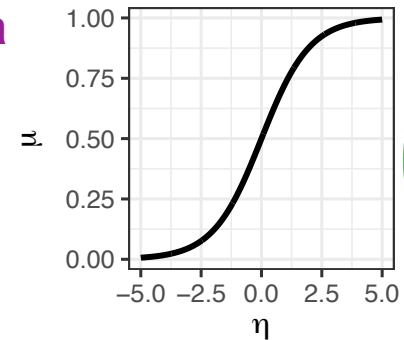
	Short < Long?	X_{Syl}	Freq < Infreq	X_{Freq}
<i>calm and relaxed</i>	✓	1	✓	1
<i>big and thick</i>	n/a	0	✓	1
<i>down and out</i>	n/a	0	✗	-1
<i>cruel and unusual</i>	✓	1	✗	-1
<i>anger and spite</i>	✗	-1	✓	1
<i>crochet and knit</i>	✗	-1	✗	-1

Learning constraint weights

Goal: Estimate good values from data

$$\eta = \beta_{Syl} X_{Syl} + \beta_{Freq} X_{Freq}$$

$$P(\text{“success”}) = \frac{e^\eta}{1 + e^\eta}$$



Logistic (sigmoid) activation function

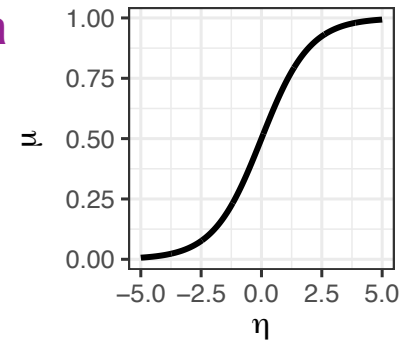
	Short < Long?	X_{Syl}	Freq < Infreq	X_{Freq}
<i>calm and relaxed</i>	✓	1	✓	1
<i>big and thick</i>	n/a	0	✓	1
<i>down and out</i>	n/a	0	✗	-1
<i>cruel and unusual</i>	✓	1	✗	-1
<i>anger and spite</i>	✗	-1	✓	1
<i>crochet and knit</i>	✗	-1	✗	-1

Learning constraint weights

Goal: Estimate good values from data

$$\eta = \beta_{Syl} X_{Syl} + \beta_{Freq} X_{Freq}$$

$$P(\text{“success”}) = \frac{e^\eta}{1 + e^\eta}$$



Logistic (sigmoid) activation function

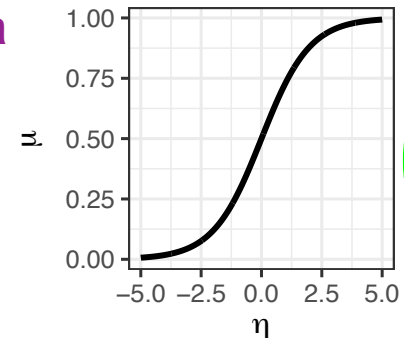
	Short < Long?	X_{Syl}	Freq < Infreq	X_{Freq}
<i>calm and relaxed</i>	✓	1	✓	1
<i>big and thick</i>	n/a	0	✓	1
<i>down and out</i>	n/a	0	✗	-1
<i>cruel and unusual</i>	✓	1	✗	-1
<i>anger and spite</i>	✗	-1	✓	1
<i>crochet and knit</i>	✗	-1	✗	-1
<i>people and soils</i>	✗	-1	✓	1
<i>surprised and dubious</i>	✓	1	✓	1
<i>abused and neglected</i>	✓	1	✗	-1
<i>lurched and stumbled</i>	✓	1	✗	-1
...				

Learning constraint weights

Goal: Estimate good values from data

$$\eta = \beta_{\text{Syl}} X_{\text{Syl}} + \beta_{\text{Freq}} X_{\text{Freq}}$$

$$P(\text{“success”}) = \frac{e^\eta}{1 + e^\eta}$$

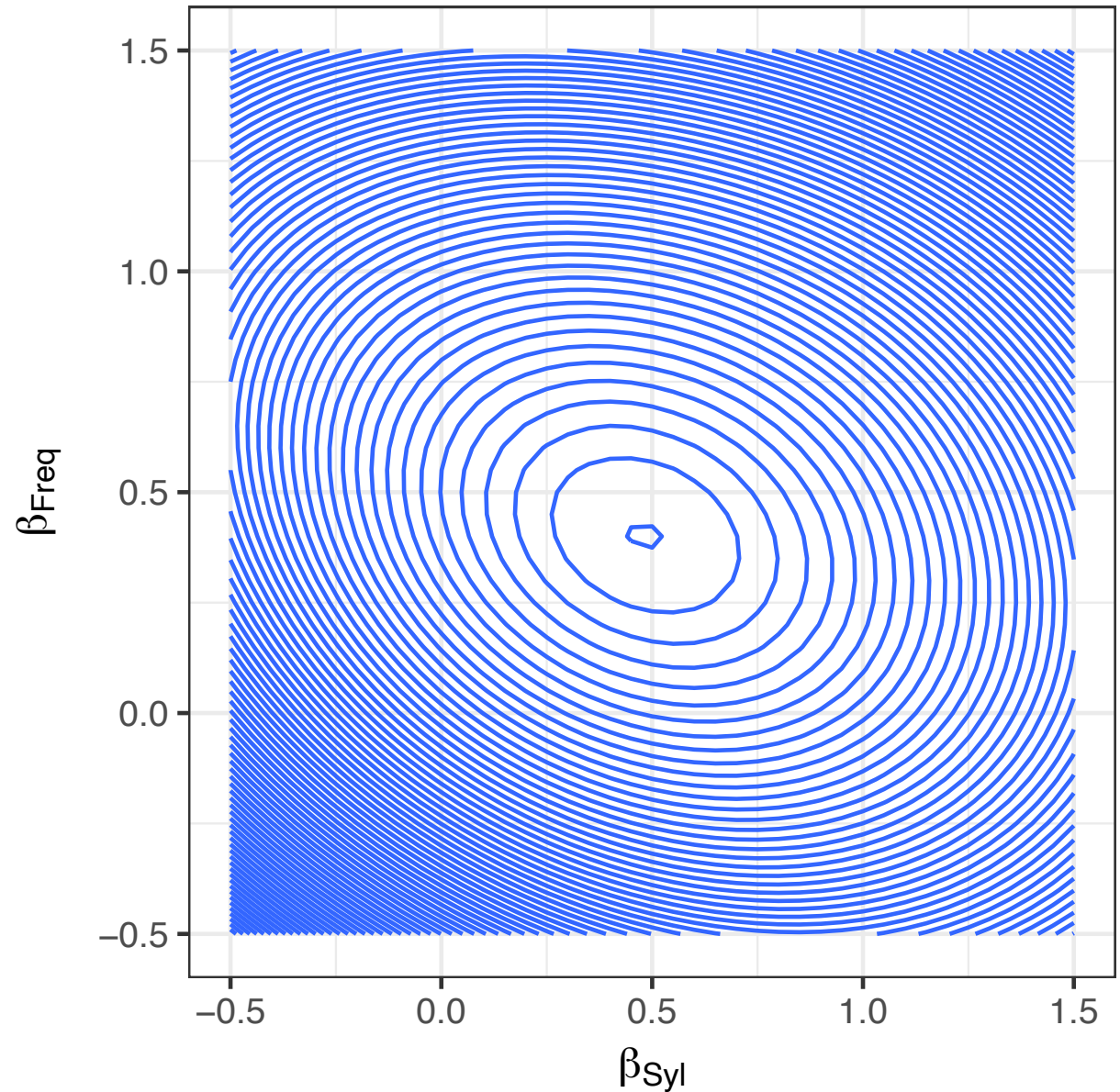


Logistic (sigmoid) activation function

	Short < Long?	X_{Syl}	Freq < Infreq	X_{Freq}
<i>calm and relaxed</i>	✓	1	✓	1
<i>big and thick</i>	n/a	0	✓	1
<i>down and out</i>	n/a	0	✗	-1
<i>cruel and unusual</i>	✓	1	✗	-1
<i>anger and spite</i>	✗	-1	✓	1
<i>crochet and knit</i>	✗	-1	✗	-1
<i>people and soils</i>	✗	-1	✓	1
<i>surprised and dubious</i>	✓	1	✓	1
<i>abused and neglected</i>	✓	1	✗	-1
<i>lurched and stumbled</i>	✓	1	✗	-1
...				

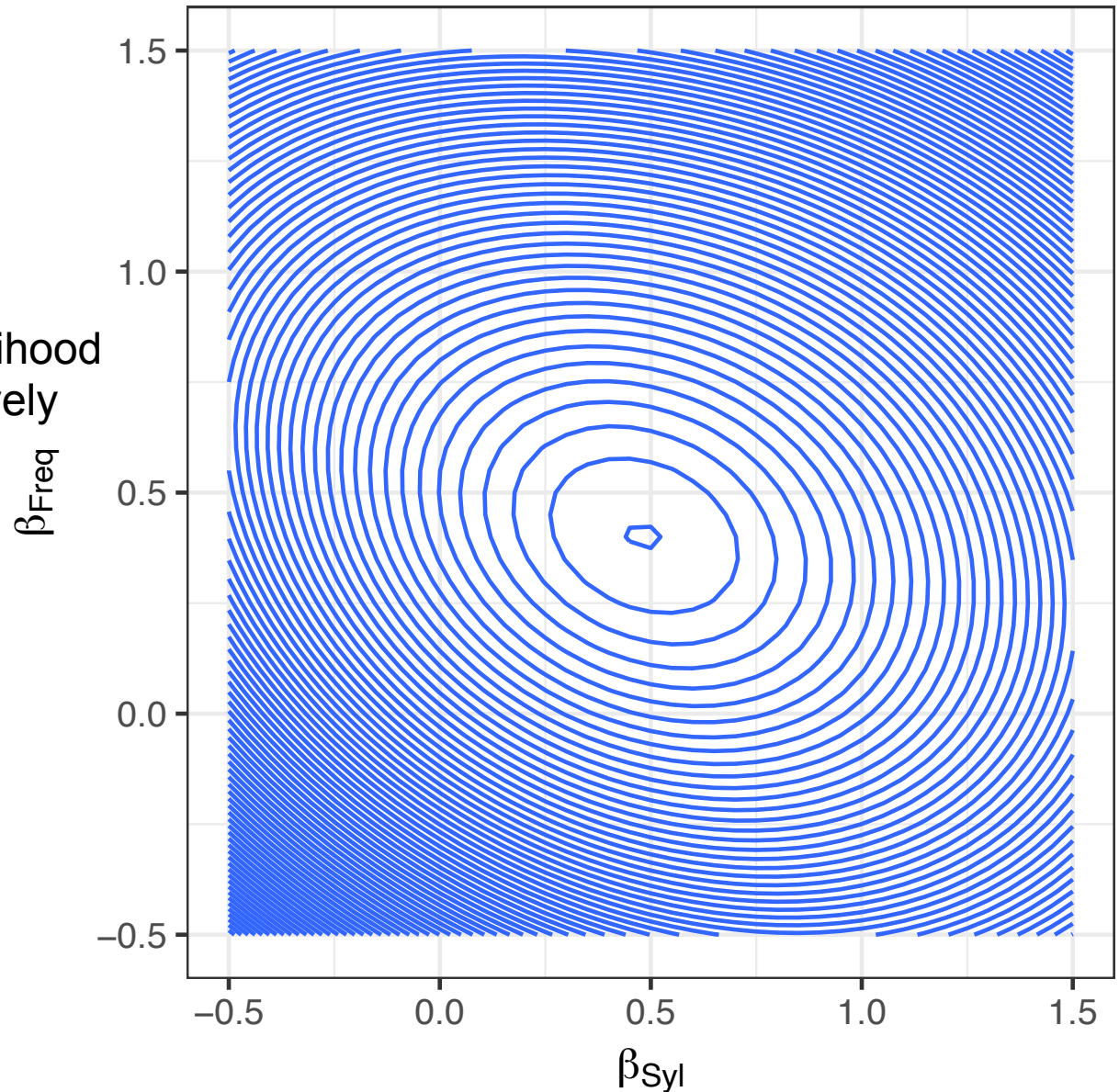
Then, e.g. find maximum-likelihood estimates $\langle \hat{\beta}_{\text{Syl}}, \hat{\beta}_{\text{Freq}} \rangle$

Maximum of the likelihood surface



Maximum of the likelihood surface

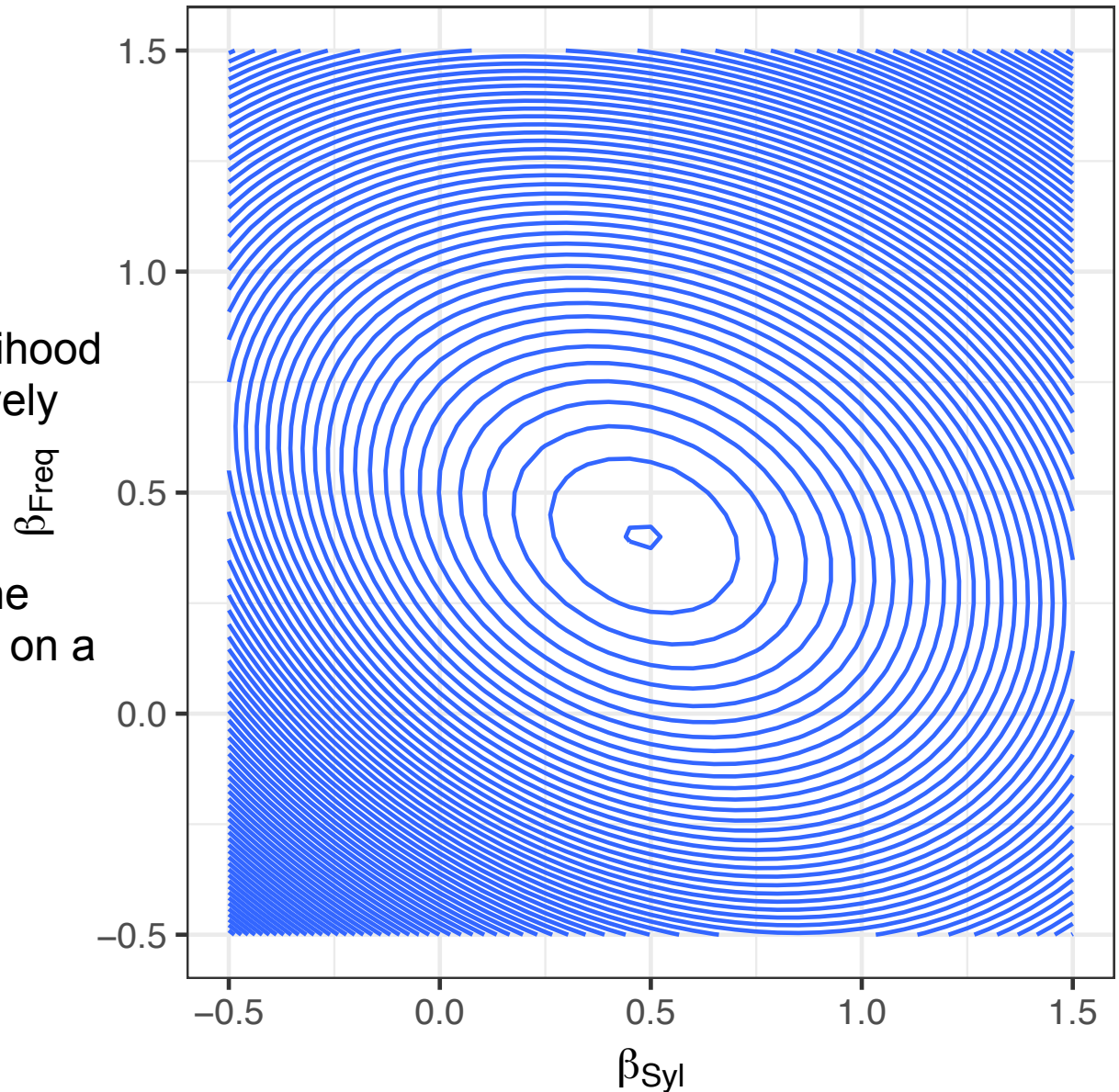
For logistic regression, likelihood surface is **convex** — relatively easy to find optimum



Maximum of the likelihood surface

For logistic regression, likelihood surface is **convex** — relatively easy to find optimum

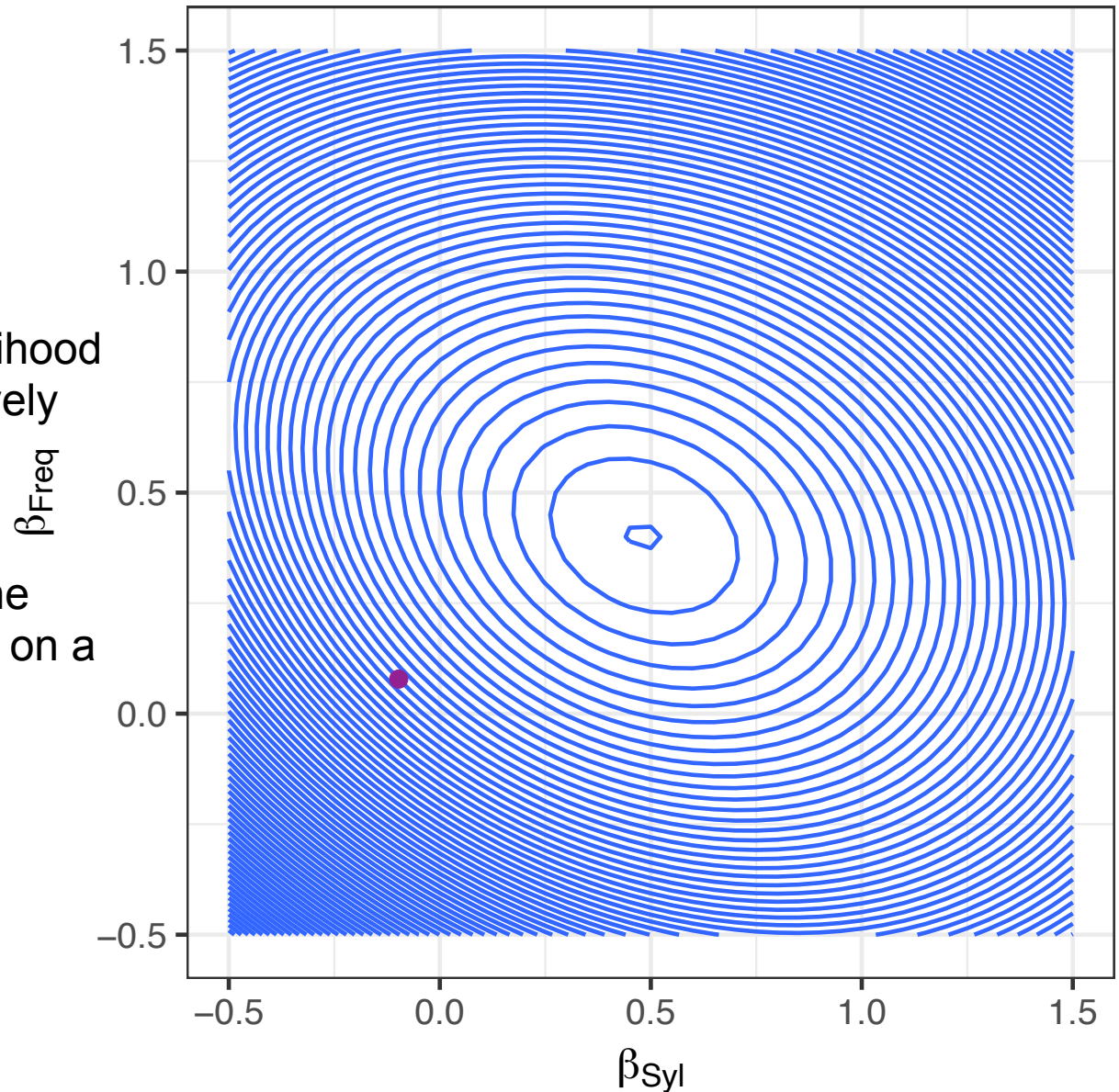
Crucial notion: **gradient**, the “derivative in all directions” on a multidimensional surface



Maximum of the likelihood surface

For logistic regression, likelihood surface is **convex** — relatively easy to find optimum

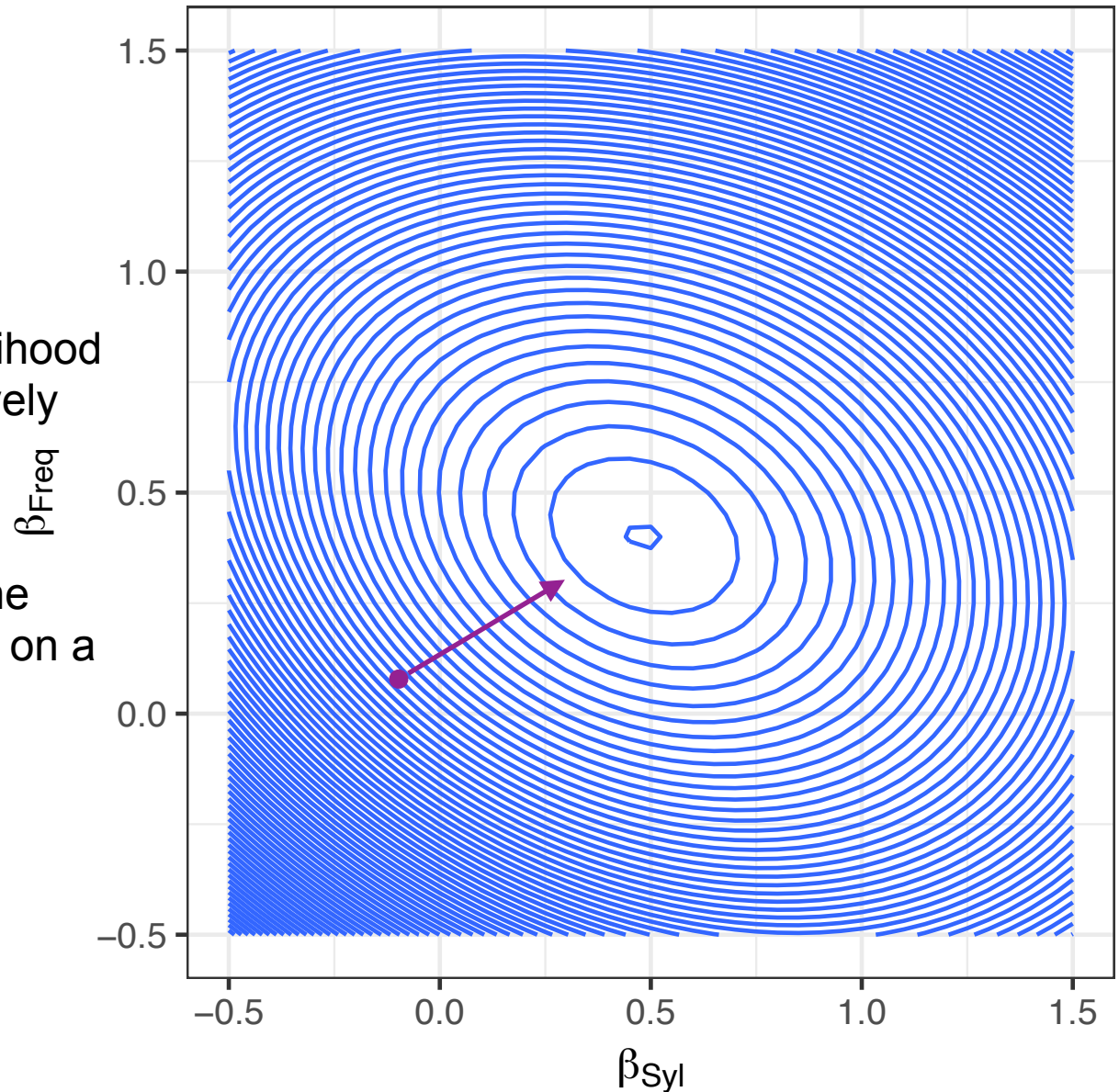
Crucial notion: **gradient**, the “derivative in all directions” on a multidimensional surface



Maximum of the likelihood surface

For logistic regression, likelihood surface is **convex** — relatively easy to find optimum

Crucial notion: **gradient**, the “derivative in all directions” on a multidimensional surface

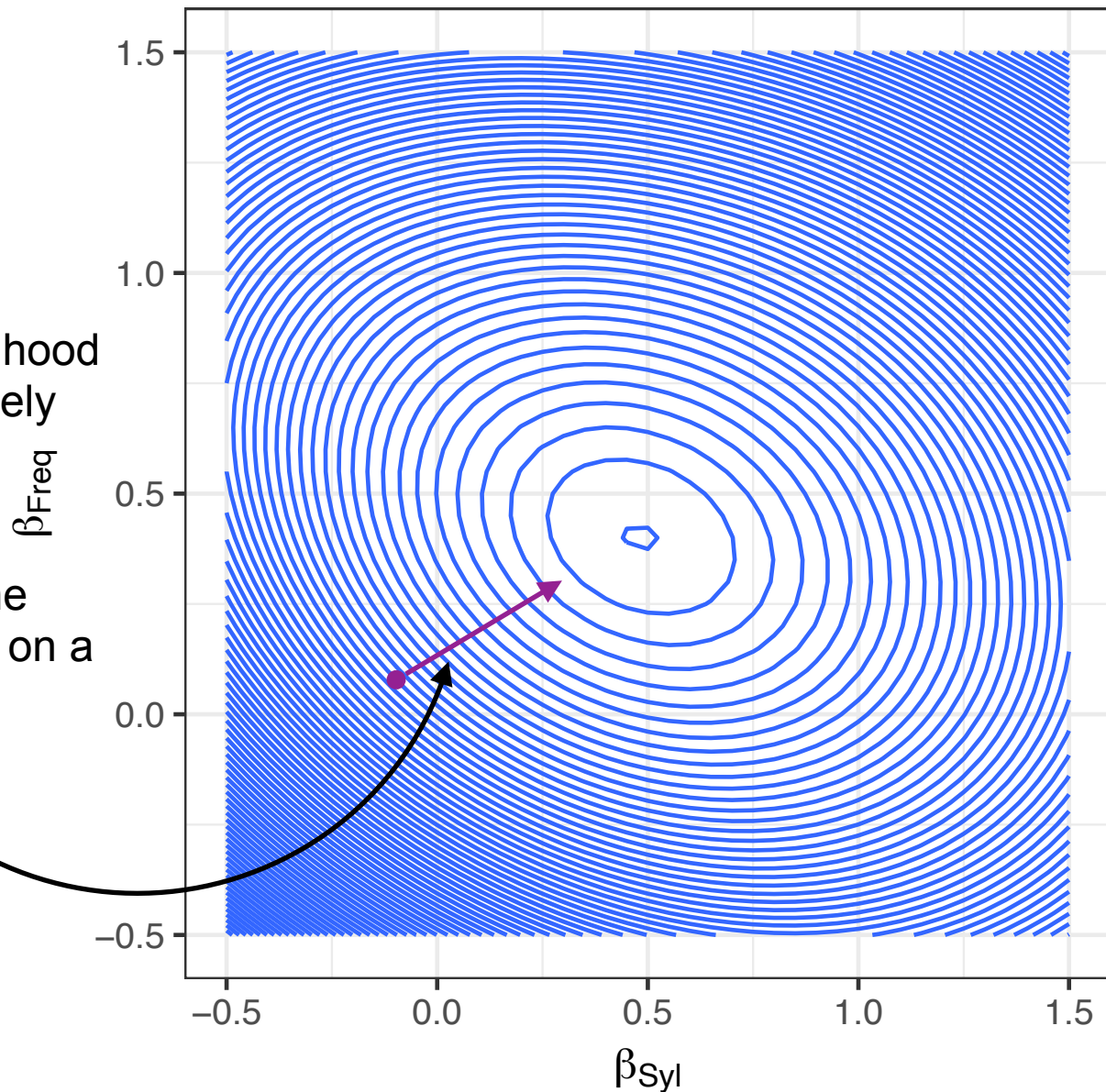


Maximum of the likelihood surface

For logistic regression, likelihood surface is **convex** — relatively easy to find optimum

Crucial notion: **gradient**, the “derivative in all directions” on a multidimensional surface

$$\nabla_{\beta} \text{Lik}(\text{Data}; \beta)$$



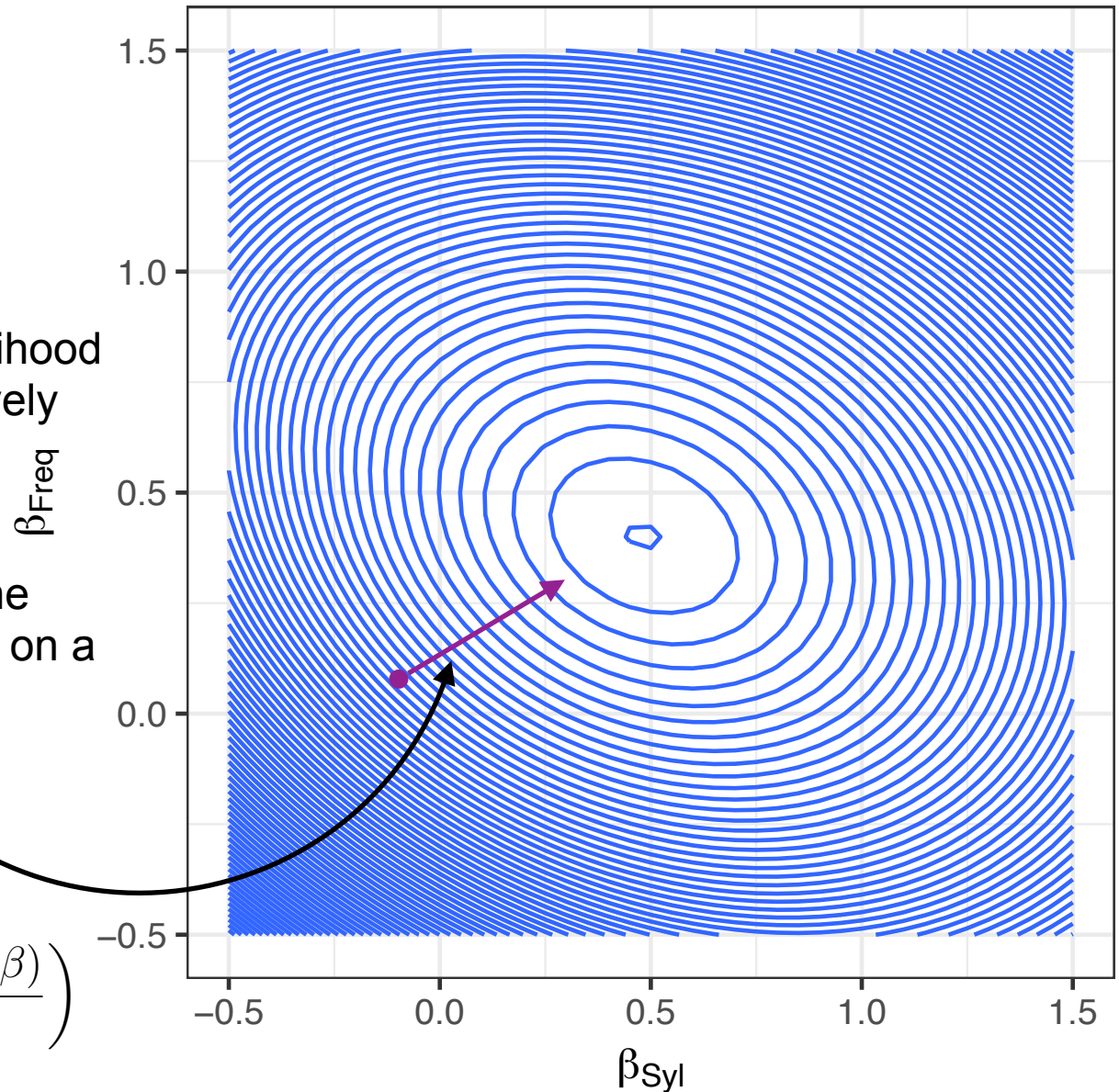
Maximum of the likelihood surface

For logistic regression, likelihood surface is **convex** — relatively easy to find optimum

Crucial notion: **gradient**, the “derivative in all directions” on a multidimensional surface

$$\nabla_{\beta} \text{Lik}(\text{Data}; \beta)$$

$$\left(\frac{\partial \text{Lik}(\text{Data}; \beta)}{\partial \beta_1}, \frac{\partial \text{Lik}(\text{Data}; \beta)}{\partial \beta_2} \right)$$



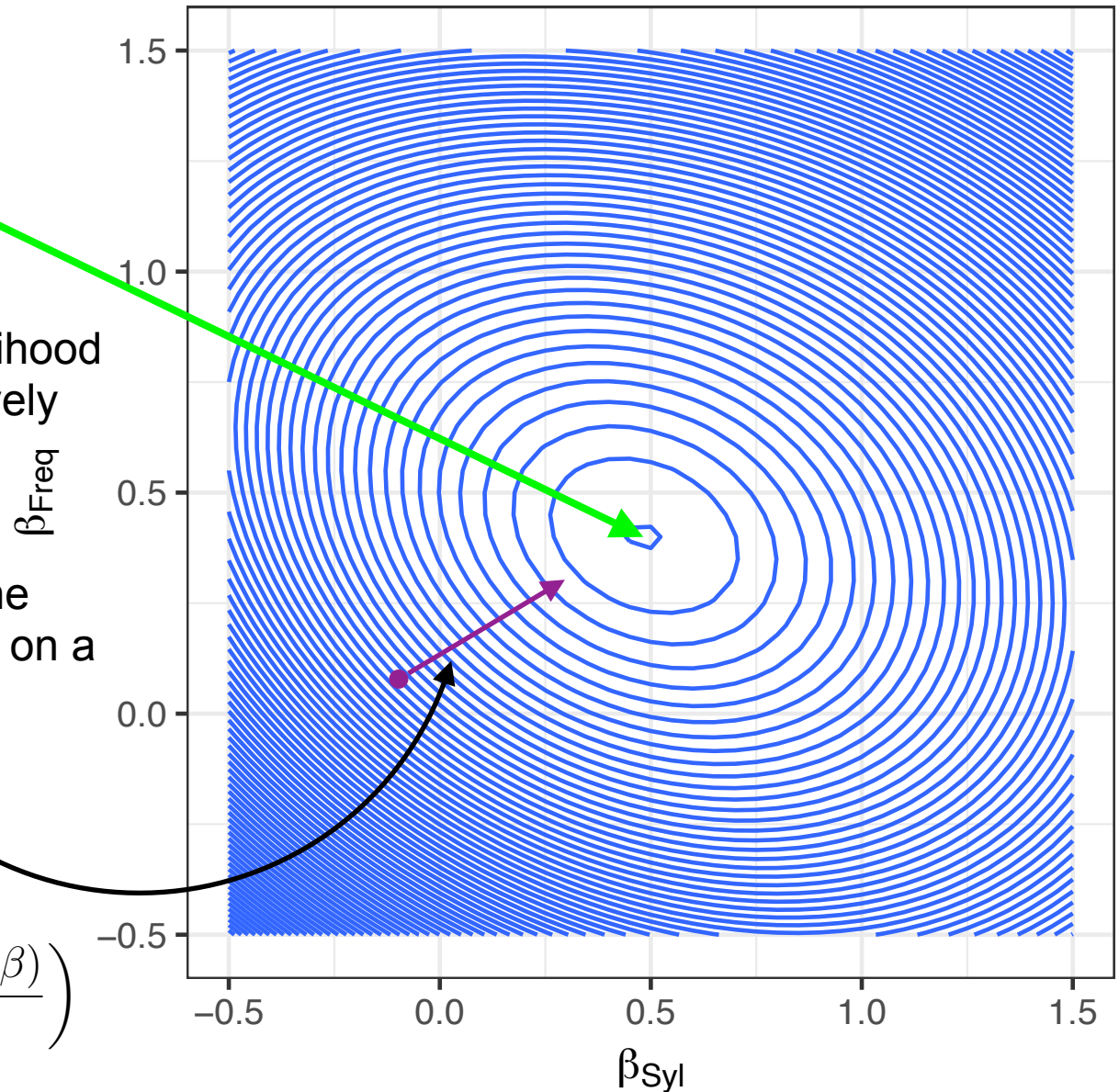
Maximum of the likelihood surface

For logistic regression, likelihood surface is **convex** — relatively easy to find optimum

Crucial notion: **gradient**, the “derivative in all directions” on a multidimensional surface

$$\nabla_{\beta} \text{Lik}(\text{Data}; \beta)$$

$$\left(\frac{\partial \text{Lik}(\text{Data}; \beta)}{\partial \beta_1}, \frac{\partial \text{Lik}(\text{Data}; \beta)}{\partial \beta_2} \right)$$



Maximum of the likelihood surface

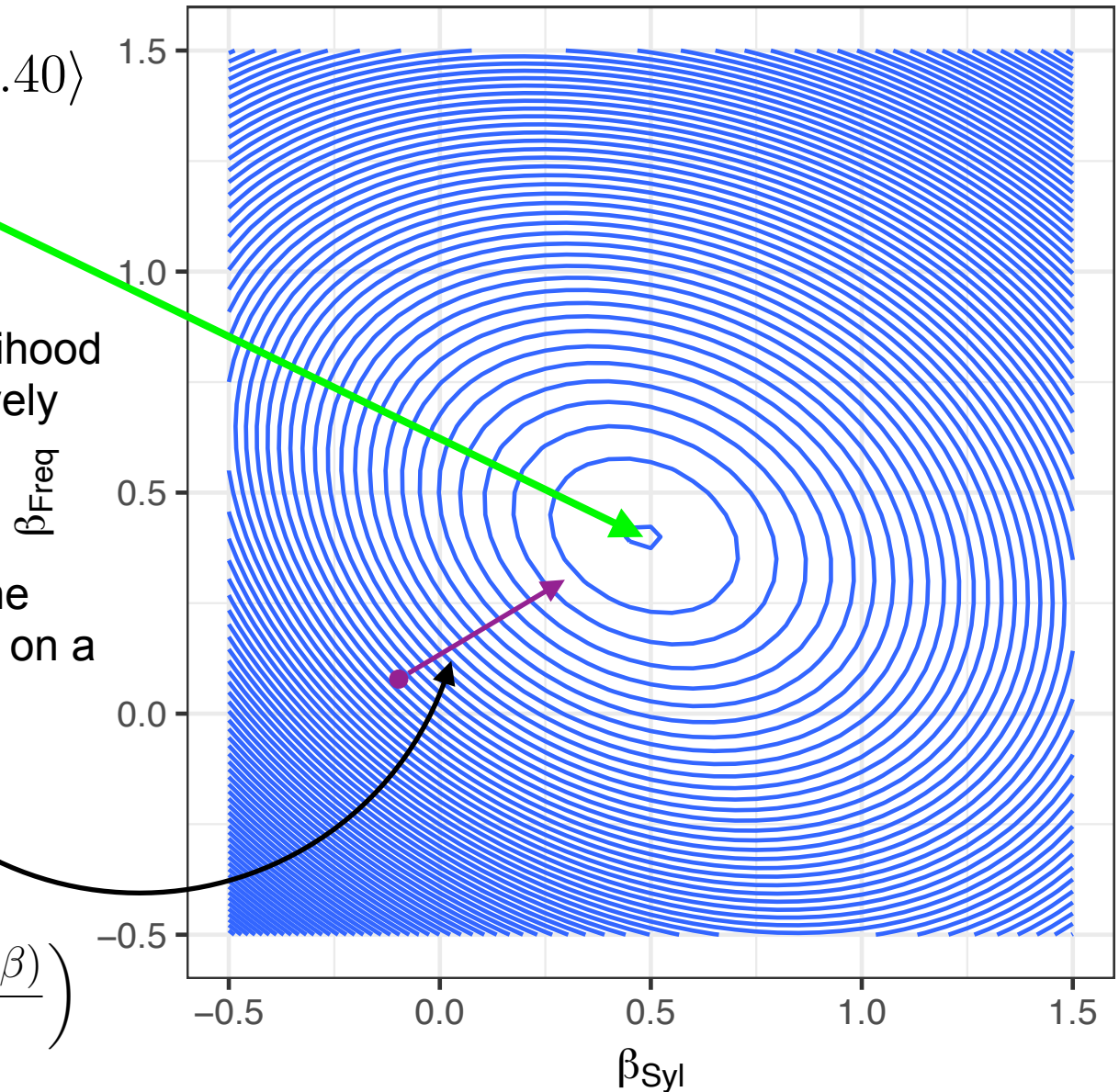
$$\langle \hat{\beta}_{Syl}, \hat{\beta}_{Freq} \rangle = \langle 0.48, 0.40 \rangle$$

For logistic regression, likelihood surface is **convex** — relatively easy to find optimum

Crucial notion: **gradient**, the “derivative in all directions” on a multidimensional surface

$$\nabla_{\beta} \text{Lik}(\text{Data}; \beta)$$

$$\left(\frac{\partial \text{Lik}(\text{Data}; \beta)}{\partial \beta_1}, \frac{\partial \text{Lik}(\text{Data}; \beta)}{\partial \beta_2} \right)$$



Limitations of logistic regression

- Logistic regression defines a ***hyperplane*** boundary separating $P(\text{"success"} | X) > 0.5$ from $P(\text{"success"} | X) < 0.5$

Limitations of logistic regression

- Logistic regression defines a ***hyperplane*** boundary separating $P(\text{"success"} | X) > 0.5$ from $P(\text{"success"} | X) < 0.5$

$$\langle \hat{\beta}_{Syl}, \hat{\beta}_{Freq} \rangle = \langle 0.48, 0.40 \rangle$$

Limitations of logistic regression

- Logistic regression defines a **hyperplane** boundary separating $P(\text{"success"} | X) > 0.5$ from $P(\text{"success"} | X) < 0.5$

$$\langle \hat{\beta}_{Syl}, \hat{\beta}_{Freq} \rangle = \langle 0.48, 0.40 \rangle$$

$$\eta = 0.48X_{Syl} + 0.4X_{Freq}$$

$$P(\text{"success"}) = \frac{e^\eta}{1 + e^\eta}$$

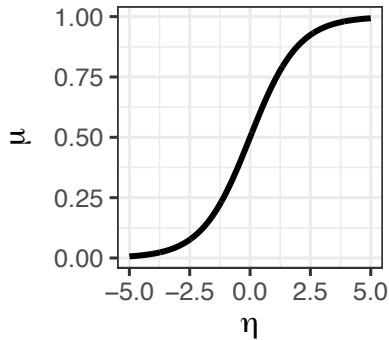
Limitations of logistic regression

- Logistic regression defines a **hyperplane** boundary separating $P(\text{"success"} | X) > 0.5$ from $P(\text{"success"} | X) < 0.5$

$$\langle \hat{\beta}_{Syl}, \hat{\beta}_{Freq} \rangle = \langle 0.48, 0.40 \rangle$$

$$\eta = 0.48X_{Syl} + 0.4X_{Freq}$$

$$P(\text{"success"}) = \frac{e^\eta}{1 + e^\eta}$$



Logistic (sigmoid)
activation function

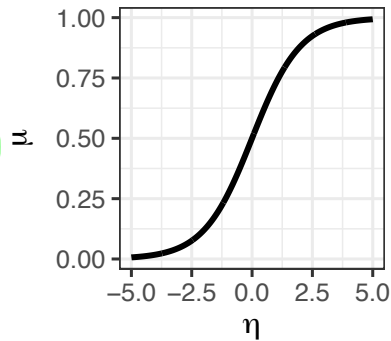
Limitations of logistic regression

- Logistic regression defines a **hyperplane** boundary separating $P(\text{"success"} | X) > 0.5$ from $P(\text{"success"} | X) < 0.5$

$$\langle \hat{\beta}_{Syl}, \hat{\beta}_{Freq} \rangle = \langle 0.48, 0.40 \rangle$$

$$\eta = 0.48X_{Syl} + 0.4X_{Freq}$$

$$P(\text{"success"}) = \frac{e^\eta}{1 + e^\eta}$$



$$0 = 0.48X_{Syl} + 0.4X_{Freq}$$

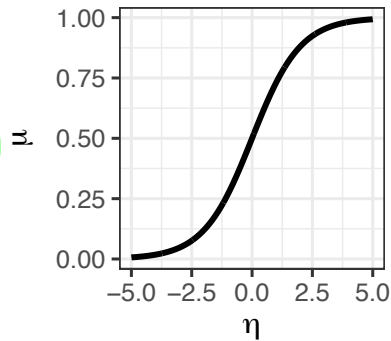
Limitations of logistic regression

- Logistic regression defines a **hyperplane** boundary separating $P(\text{"success"} | X) > 0.5$ from $P(\text{"success"} | X) < 0.5$

$$\langle \hat{\beta}_{Syl}, \hat{\beta}_{Freq} \rangle = \langle 0.48, 0.40 \rangle$$

$$\eta = 0.48X_{Syl} + 0.4X_{Freq}$$

$$P(\text{"success"}) = \frac{e^\eta}{1 + e^\eta}$$



Logistic (sigmoid)
activation function

$$0 = 0.48X_{Syl} + 0.4X_{Freq}$$

$$X_{Freq} = -\frac{0.48}{0.4}X_{Syl}$$

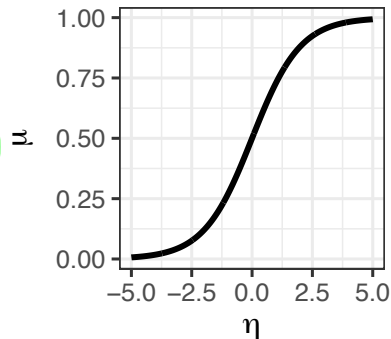
Limitations of logistic regression

- Logistic regression defines a **hyperplane** boundary separating $P(\text{"success"} | X) > 0.5$ from $P(\text{"success"} | X) < 0.5$

$$\langle \hat{\beta}_{Syl}, \hat{\beta}_{Freq} \rangle = \langle 0.48, 0.40 \rangle$$

$$\eta = 0.48X_{Syl} + 0.4X_{Freq}$$

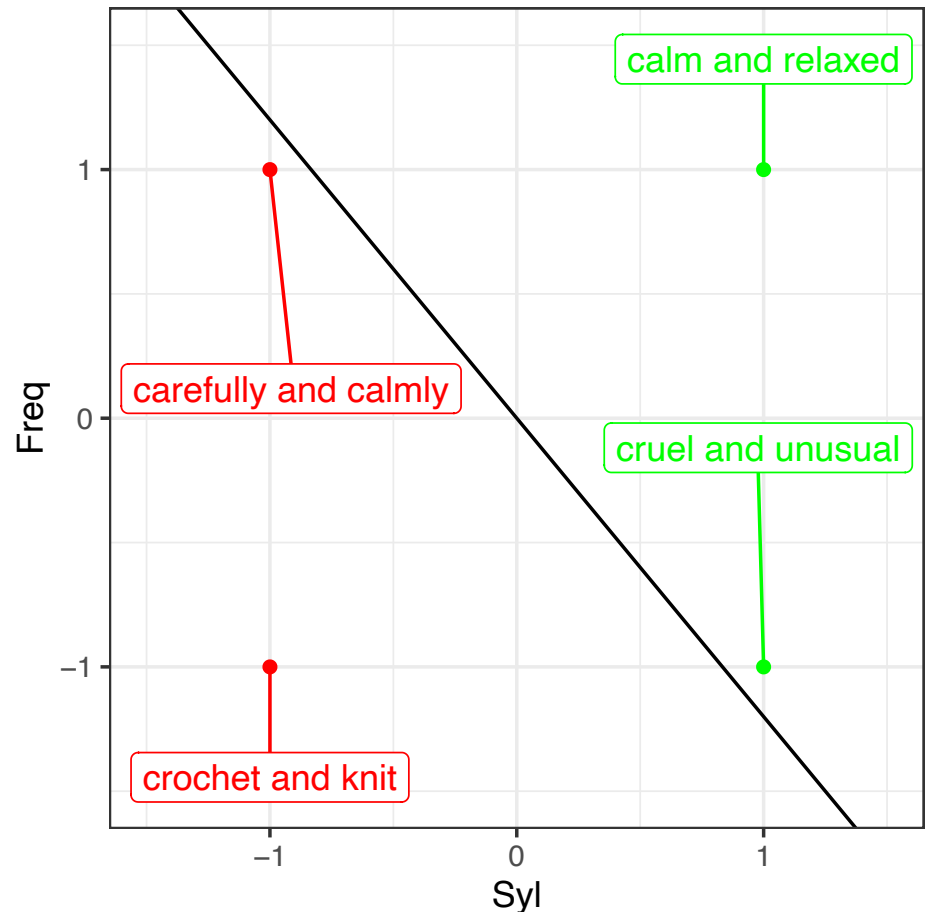
$$P(\text{"success"}) = \frac{e^\eta}{1 + e^\eta}$$



Logistic (sigmoid)
activation function

$$0 = 0.48X_{Syl} + 0.4X_{Freq}$$

$$X_{Freq} = -\frac{0.48}{0.4}X_{Syl}$$



Problems that aren't linearly separable

Problems that aren't linearly separable

- But many prediction problems aren't linearly separable

Problems that aren't linearly separable

- But many prediction problems aren't linearly separable

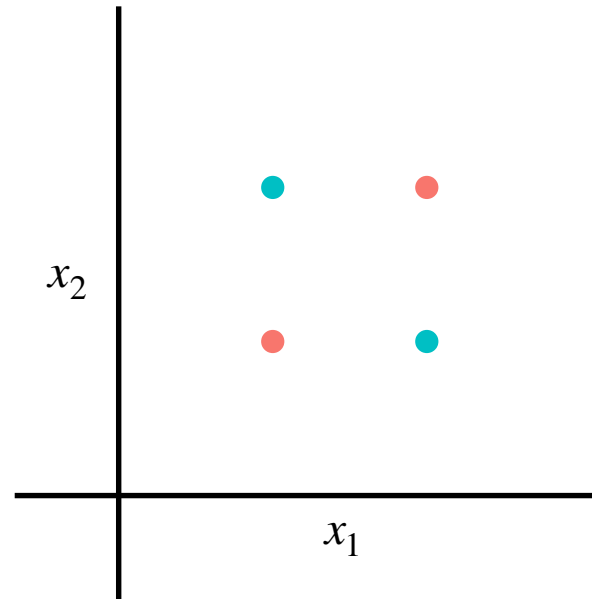
	x_1	x_2	Class
XOR problem	0	0	1
	0	1	0
	1	0	0
	1	1	1

Problems that aren't linearly separable

- But many prediction problems aren't linearly separable

XOR problem

x_1	x_2	Class
0	0	1
0	1	0
1	0	0
1	1	1

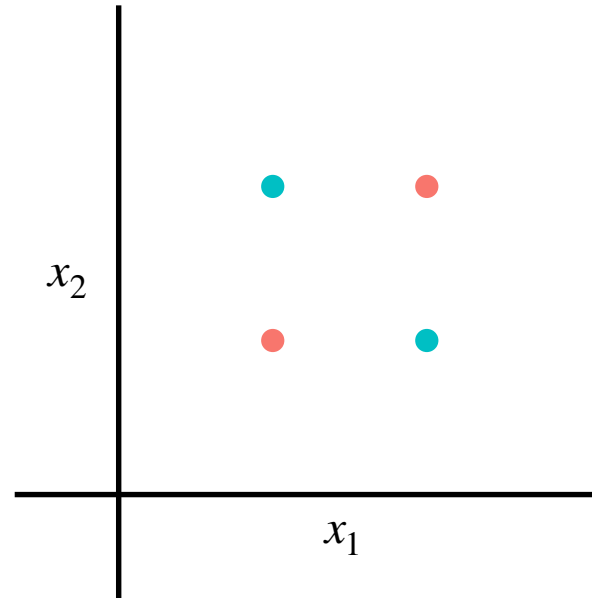


Problems that aren't linearly separable

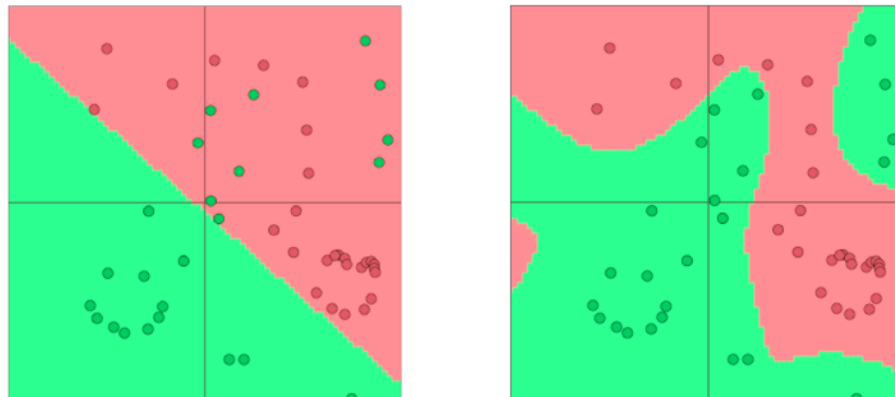
- But many prediction problems aren't linearly separable

XOR problem

x_1	x_2	Class
0	0	1
0	1	0
1	0	0
1	1	1

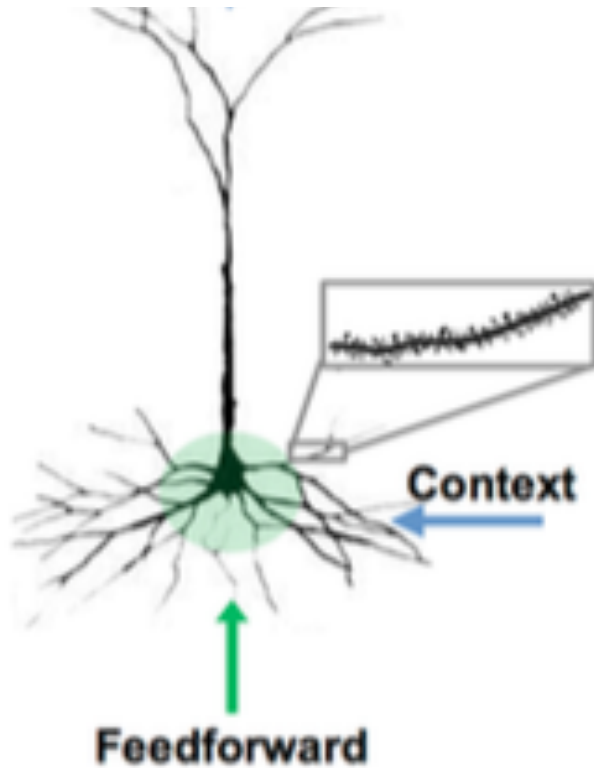


More generally, we want flexibly-shaped class boundaries:



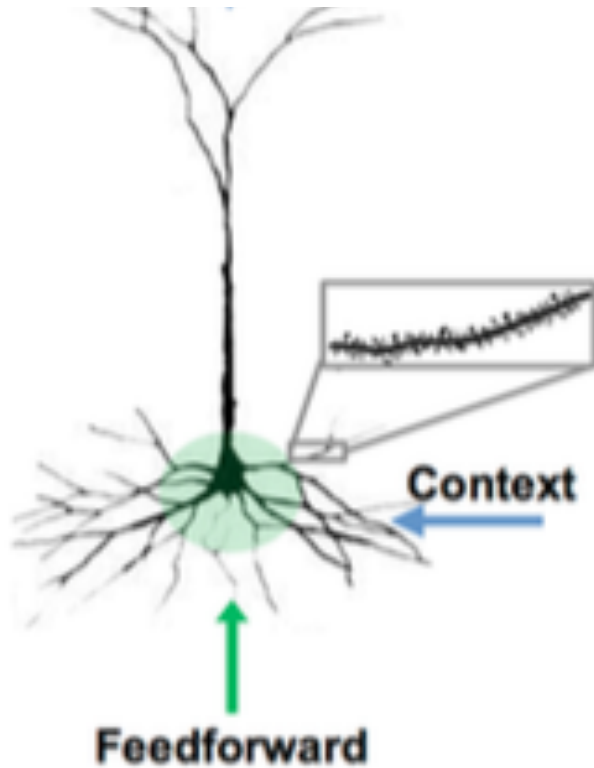
Logistic regression as a “neuron”

Biological neuron

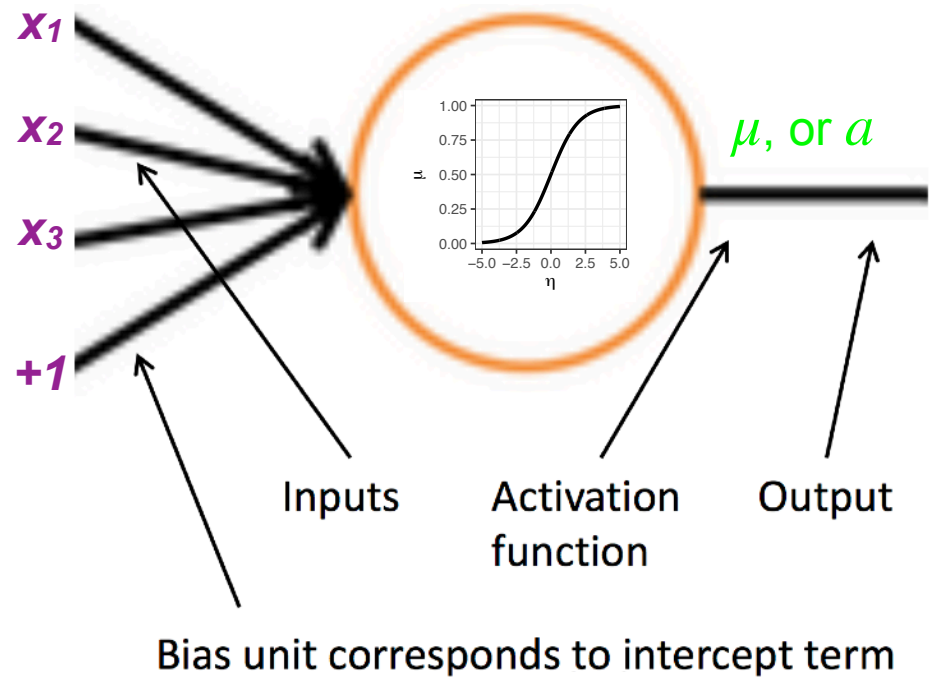


Logistic regression as a “neuron”

Biological neuron

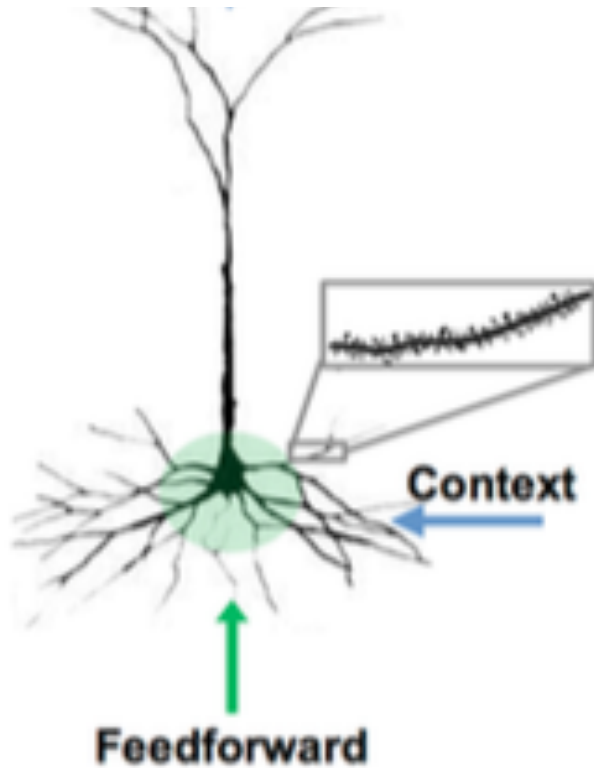


Artificial neuron

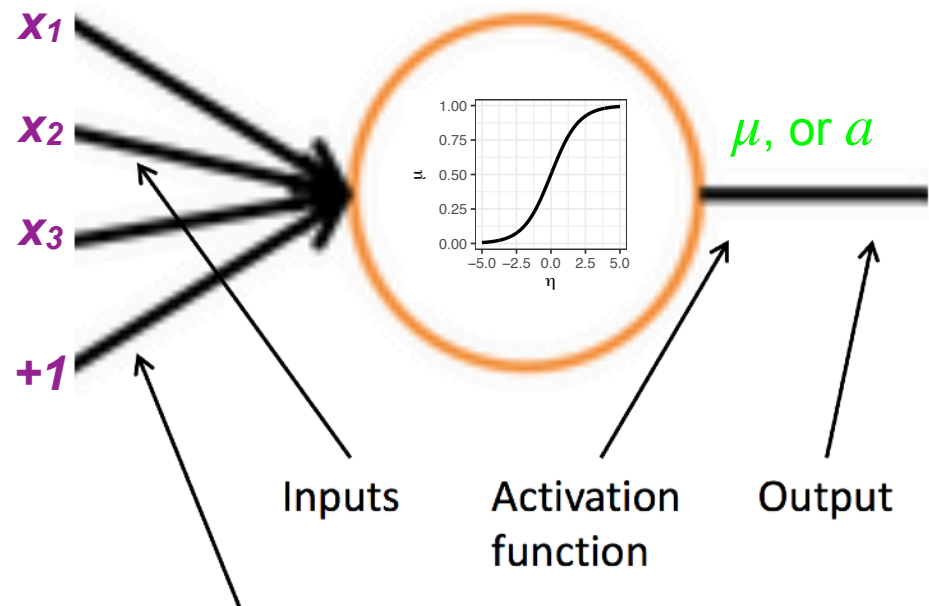


Logistic regression as a “neuron”

Biological neuron



Artificial neuron



Bias unit corresponds to intercept term

$$\eta = \sum_i \beta_i X_i \quad \mu = \frac{e^\eta}{1 + e^\eta}$$

Neurons are organized in networks!

BULLETIN OF
MATHEMATICAL BIOPHYSICS
VOLUME 5, 1943

A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY

WARREN S. MCCULLOCH AND WALTER PITTS

FROM THE UNIVERSITY OF ILLINOIS, COLLEGE OF MEDICINE,
DEPARTMENT OF PSYCHIATRY AT THE ILLINOIS NEUROPSYCHIATRIC INSTITUTE,
AND THE UNIVERSITY OF CHICAGO

Because of the "all-or-none" character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms, with the addition of more complicated logical means for nets containing circles; and that for any logical expression satisfying certain conditions, one can find a net behaving in the fashion it describes. It is shown that many particular choices among possible neurophysiological assumptions are equivalent, in the sense that for every net behaving under one assumption, there exists another net which behaves under the other and gives the same results, although perhaps not in the same time. Various applications of the calculus are discussed.

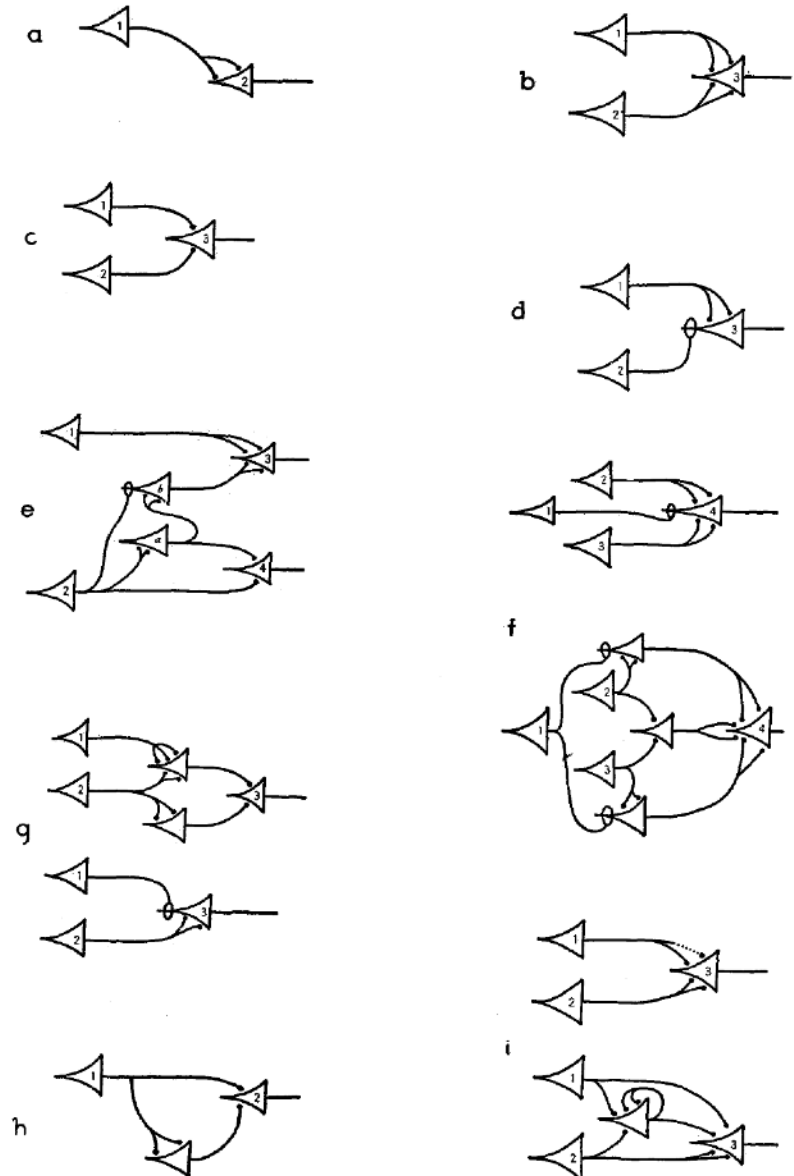
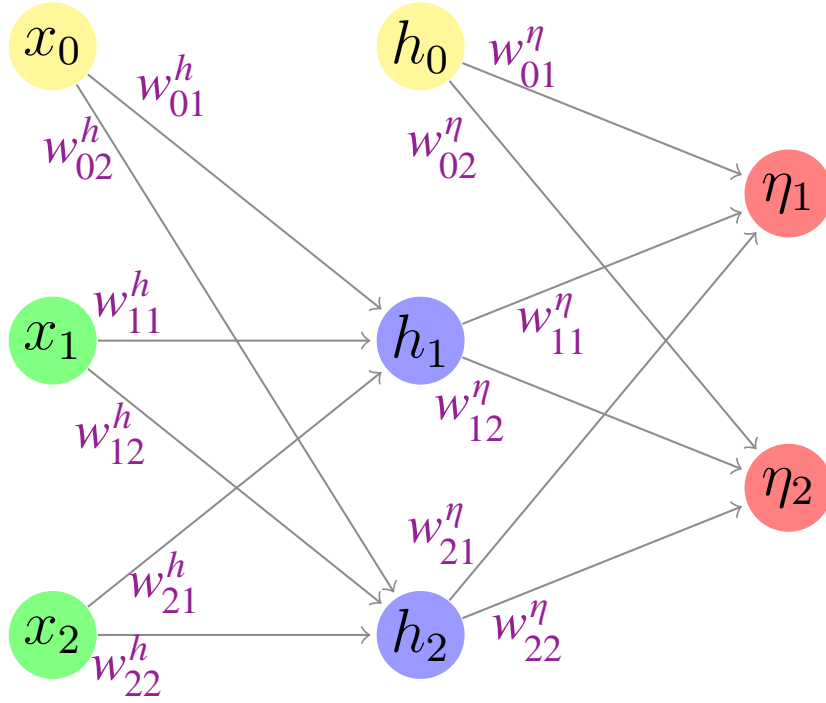


FIGURE 1

A simple single-hidden-layer neural network

Input Hidden Output



Predict: is y class 1 or class 2?

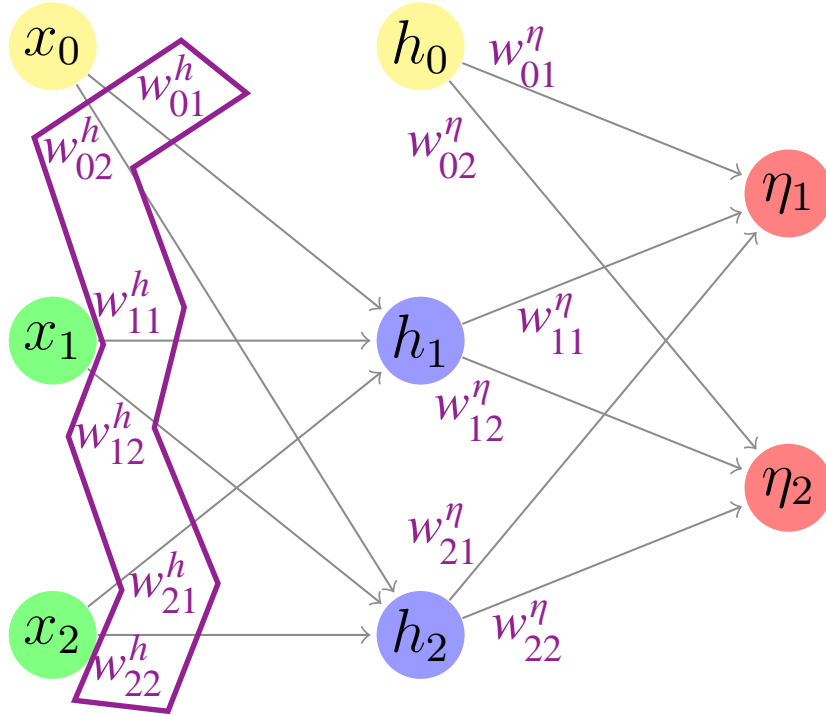
y

A simple single-hidden-layer neural network

Input

Hidden

Output



Predict: is y class 1 or class 2?

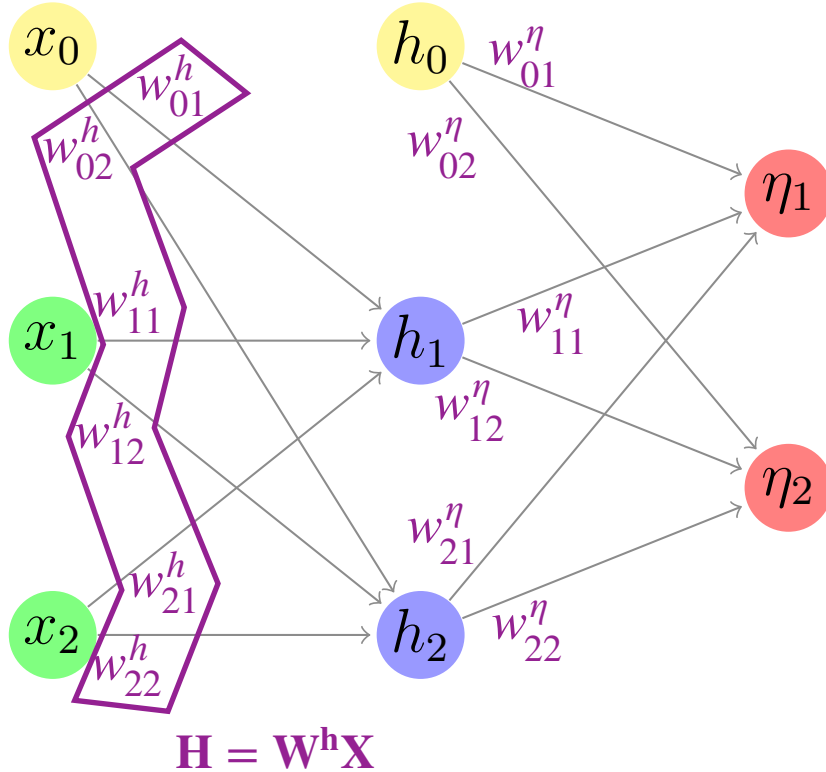
y

A simple single-hidden-layer neural network

Input

Hidden

Output



Predict: is y class 1 or class 2?

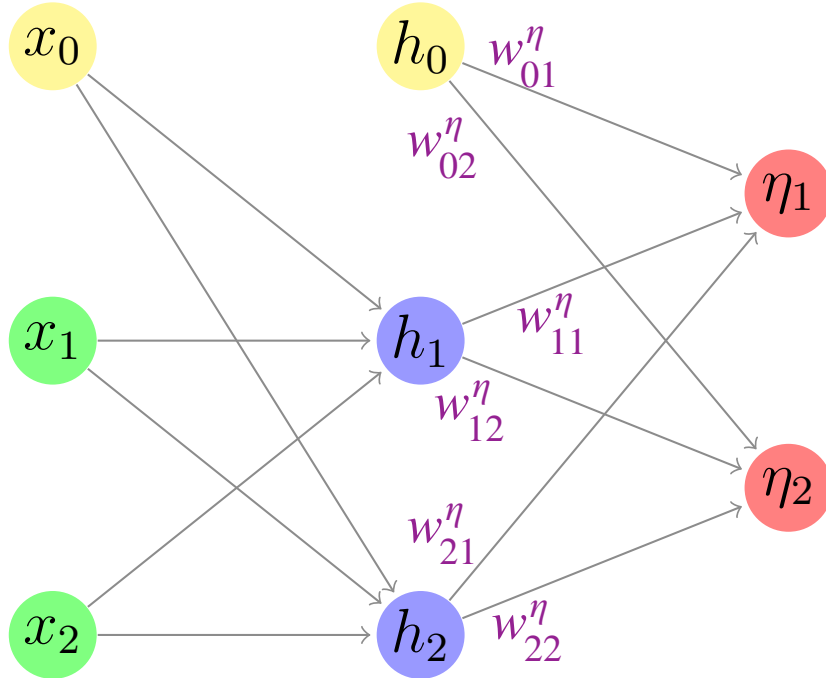
y

A simple single-hidden-layer neural network

Input

Hidden

Output



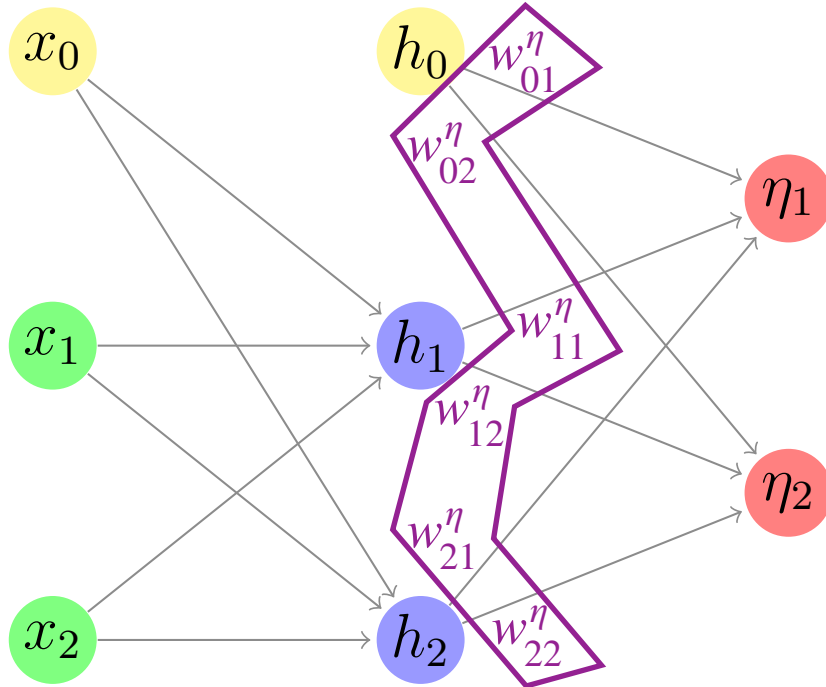
Predict: is y class 1 or class 2?

y

$$\mathbf{H} = \mathbf{W}^h \mathbf{X}$$

A simple single-hidden-layer neural network

Input Hidden Output



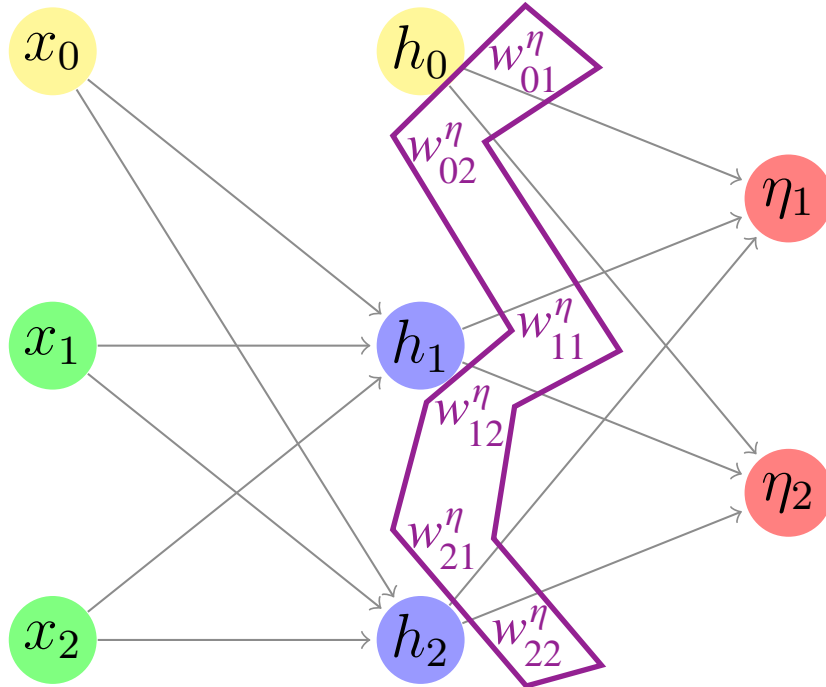
$$\mathbf{H} = \mathbf{W}^h \mathbf{X}$$

Predict: is y class 1 or class 2?

y

A simple single-hidden-layer neural network

Input Hidden Output



Predict: is y class 1 or class 2?

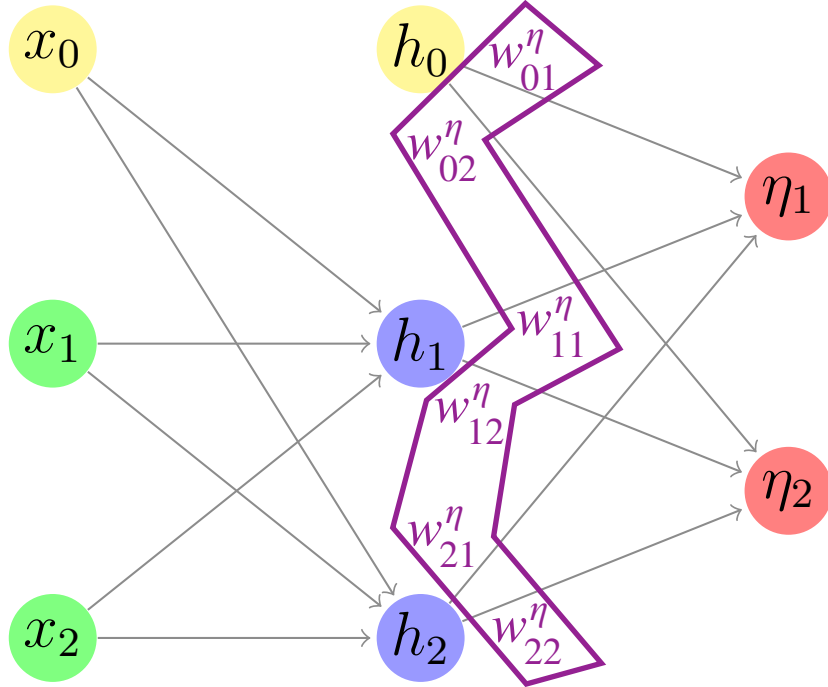
y

$$\mathbf{H} = \mathbf{W}^h \mathbf{X}$$

$$\boldsymbol{\eta} = \mathbf{W}^\eta f(\mathbf{H})$$

A simple single-hidden-layer neural network

Input Hidden Output



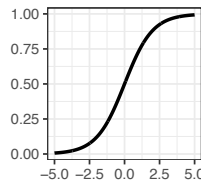
Predict: is y class 1 or class 2?

y

$$\mathbf{H} = \mathbf{W}^h \mathbf{X}$$

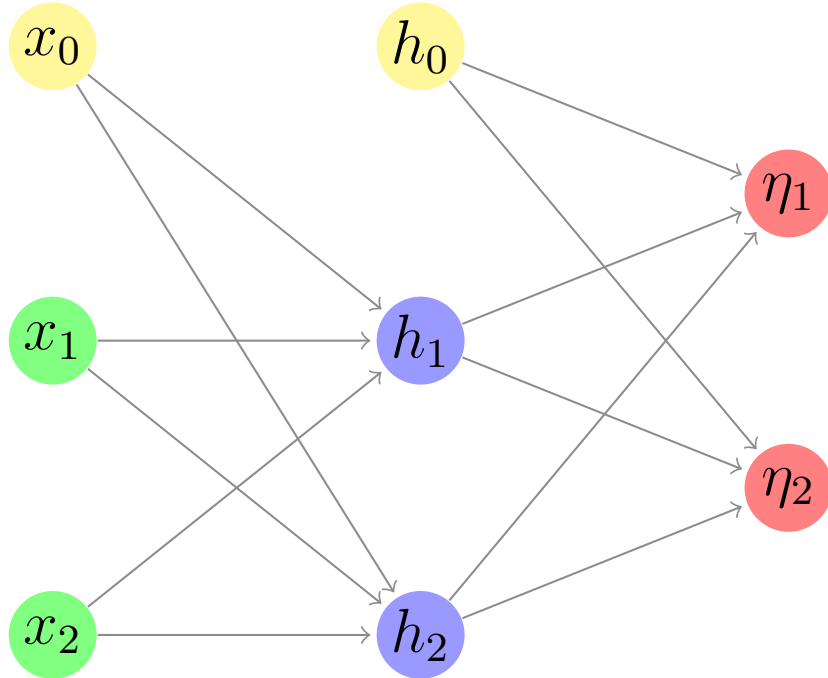
$$\eta = \mathbf{W}^\eta f(\mathbf{H})$$

nonlinear activation function
(e.g., sigmoid)



A simple single-hidden-layer neural network

Input Hidden Output



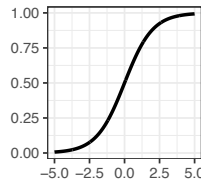
Predict: is y class 1 or class 2?

y

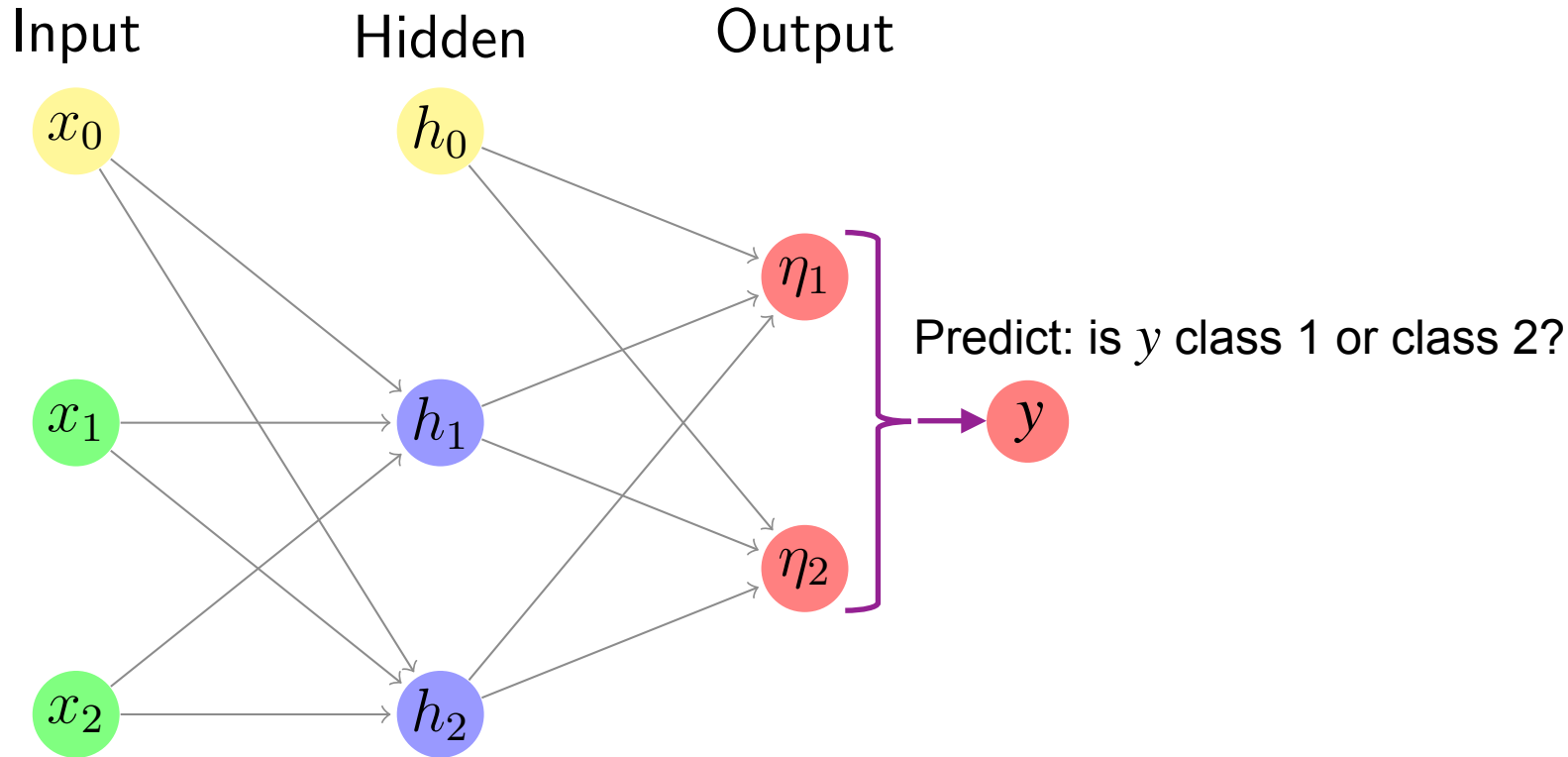
$$\mathbf{H} = \mathbf{W}^h \mathbf{X}$$

$$\boldsymbol{\eta} = \mathbf{W}^o f(\mathbf{H})$$

nonlinear activation function
(e.g., sigmoid)



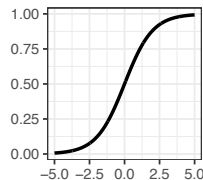
A simple single-hidden-layer neural network



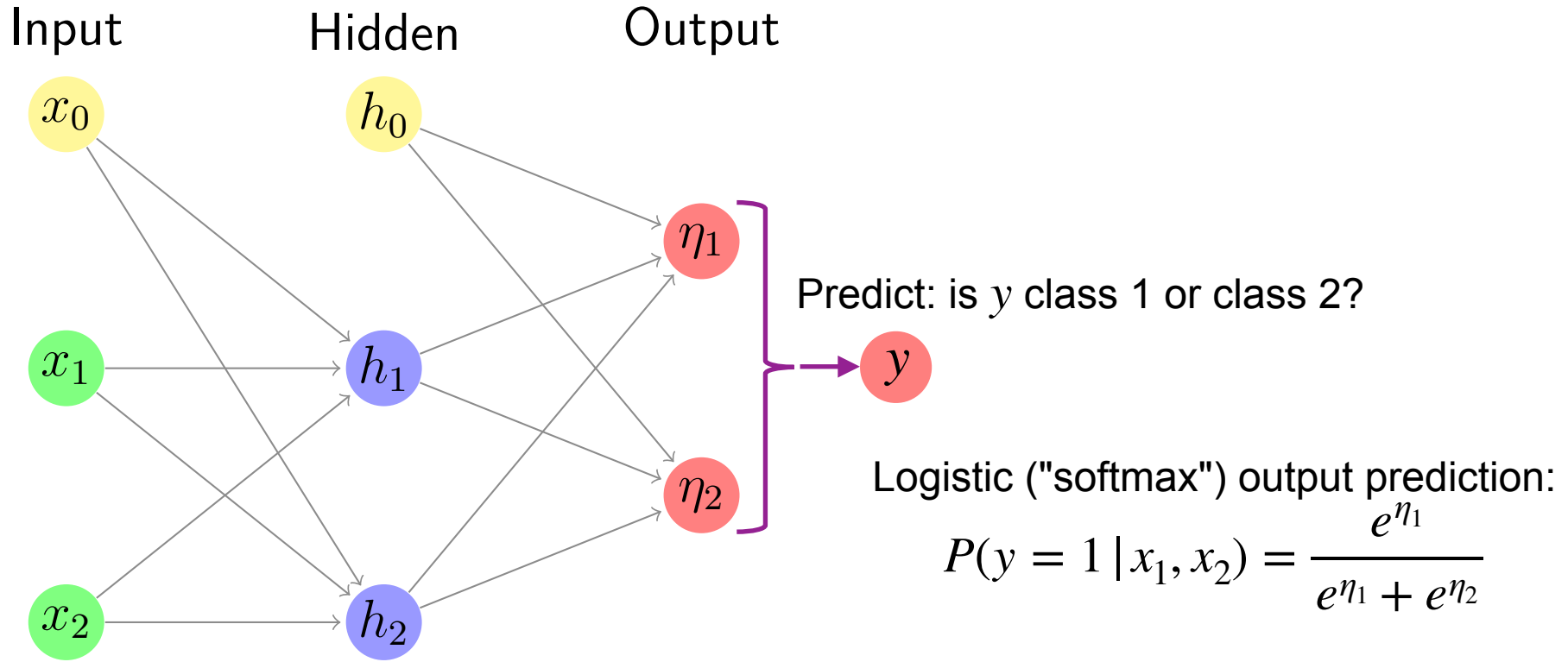
$$\mathbf{H} = \mathbf{W}^h \mathbf{X}$$

$$\boldsymbol{\eta} = \mathbf{W}^o f(\mathbf{H})$$

nonlinear activation function
(e.g., sigmoid)



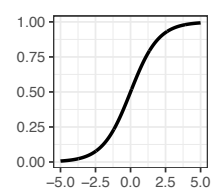
A simple single-hidden-layer neural network



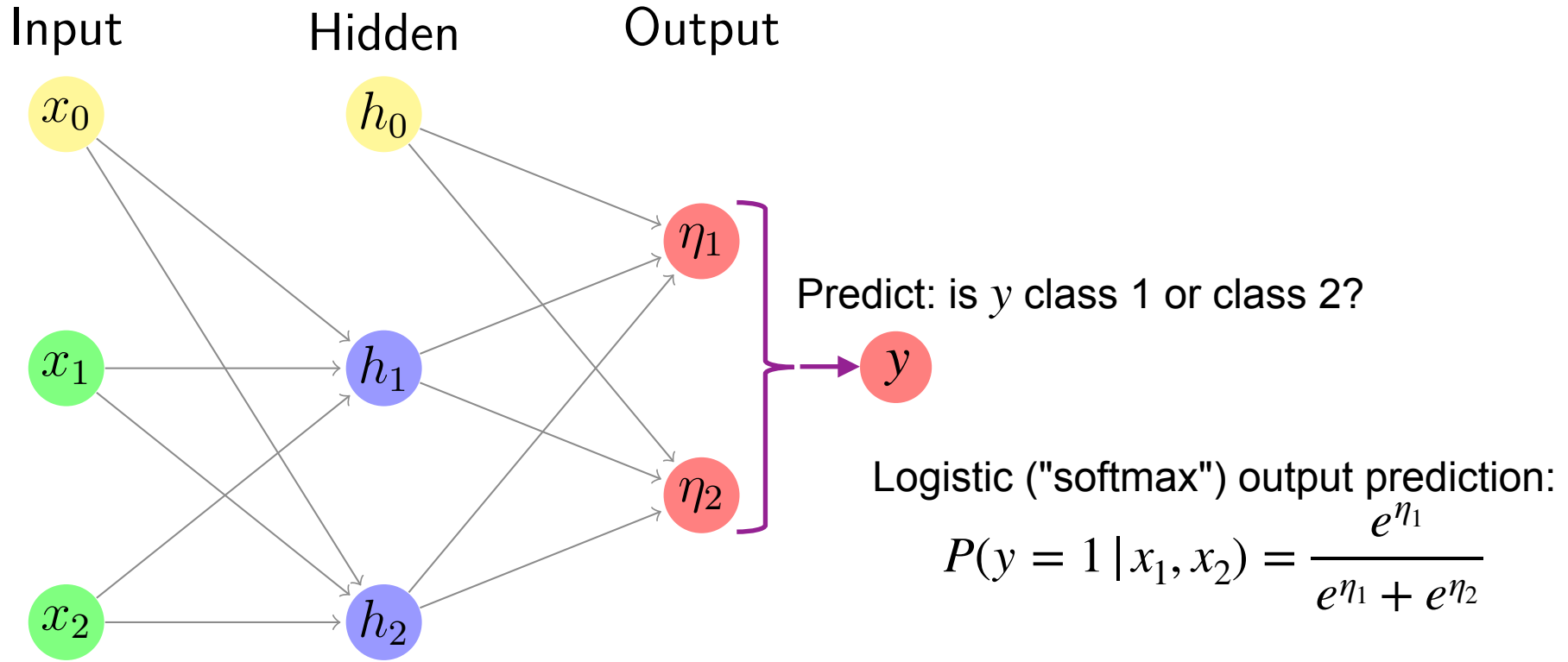
Logistic ("softmax") output prediction:

$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$

nonlinear activation function
(e.g., sigmoid)



A simple single-hidden-layer neural network



Logistic ("softmax") output prediction:

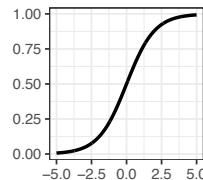
$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$

$$\mathbf{H} = \mathbf{W}^h \mathbf{X}$$

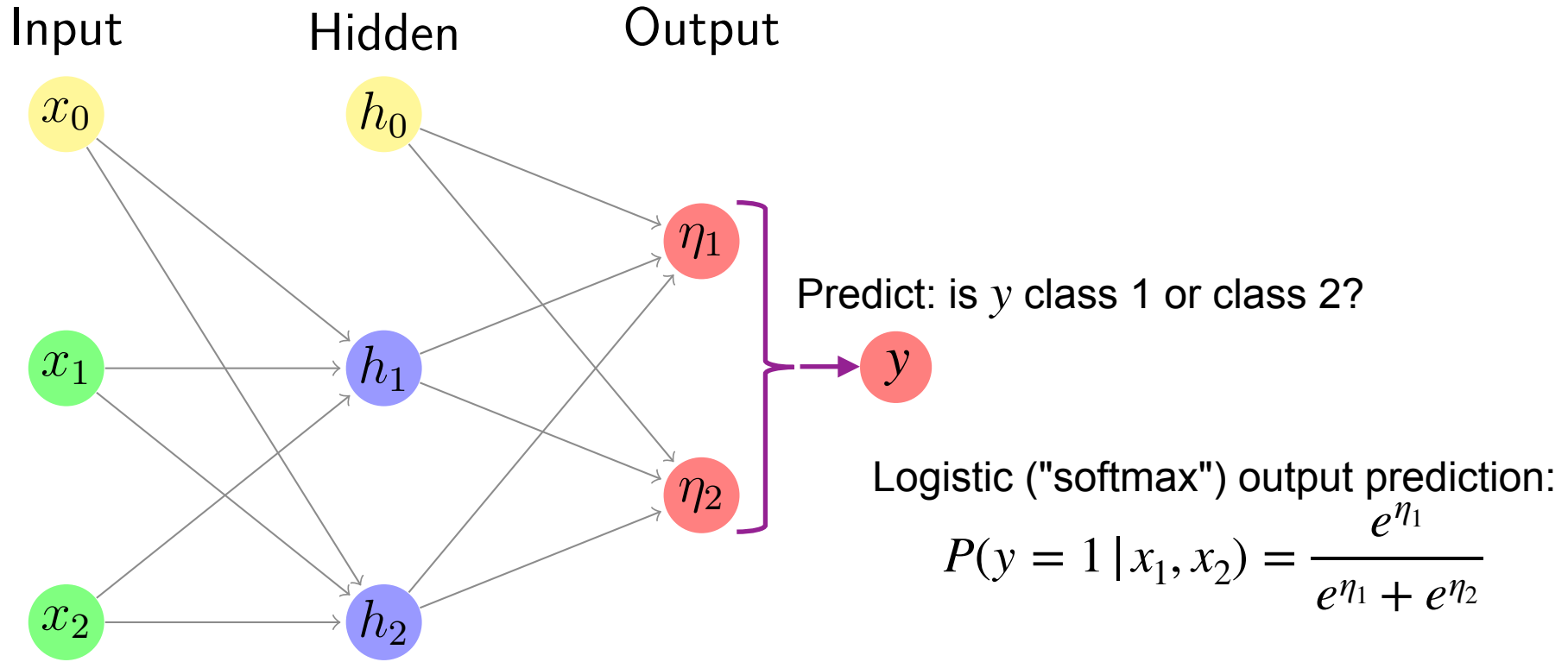
$$\boldsymbol{\eta} = \mathbf{W}^o f(\mathbf{H})$$

$$P(\mathbf{y} | \mathbf{X}) = \text{softmax}(\boldsymbol{\eta})$$

nonlinear activation function
(e.g., sigmoid)



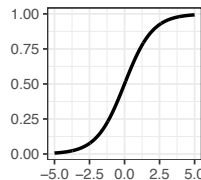
A simple single-hidden-layer neural network



$$\mathbf{H} = \mathbf{W}^h \mathbf{X}$$

$$\eta = \mathbf{W}^\eta f(\mathbf{H})$$

nonlinear activation function
(e.g., sigmoid)



Predict: is y class 1 or class 2?

Logistic ("softmax") output prediction:

$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$

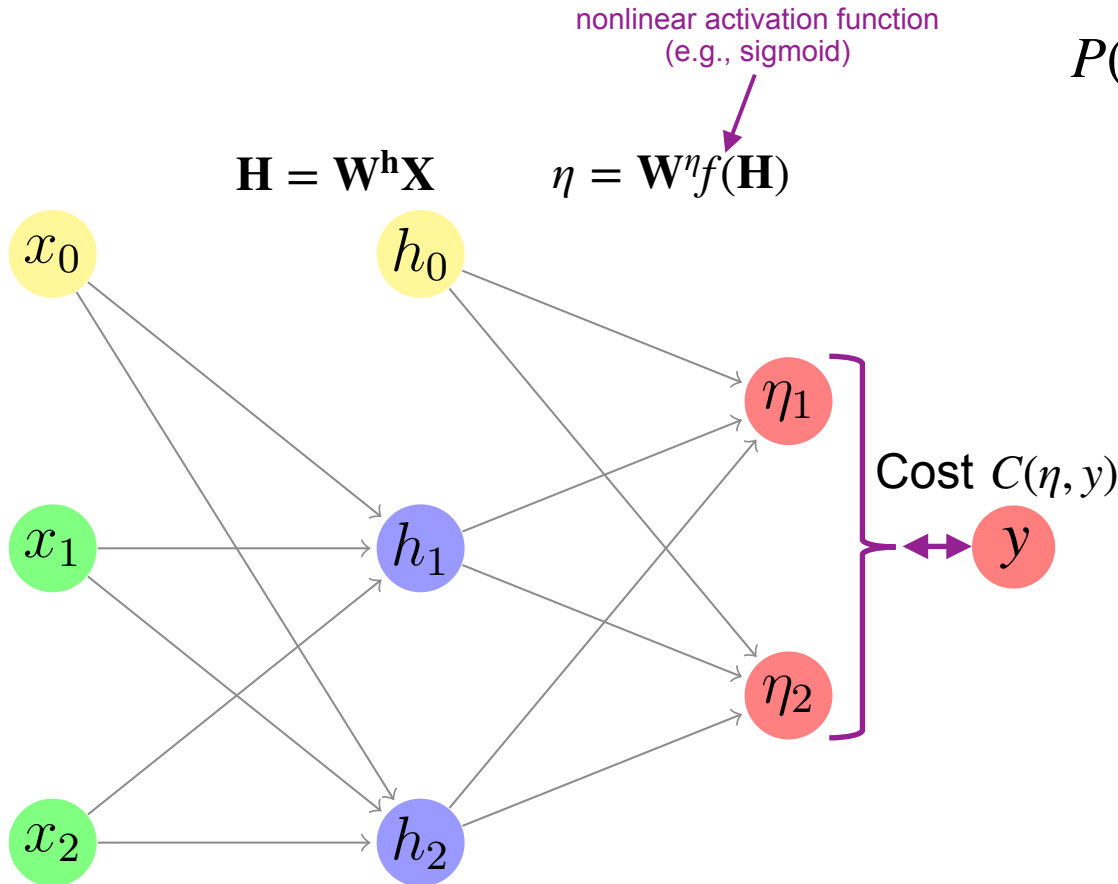
$$P(\mathbf{y} | \mathbf{X}) = \text{softmax}(\eta)$$

$$P(\mathbf{y} | \mathbf{X}) = \text{softmax}(\mathbf{W}^\eta f(\mathbf{W}^h \mathbf{X}))$$

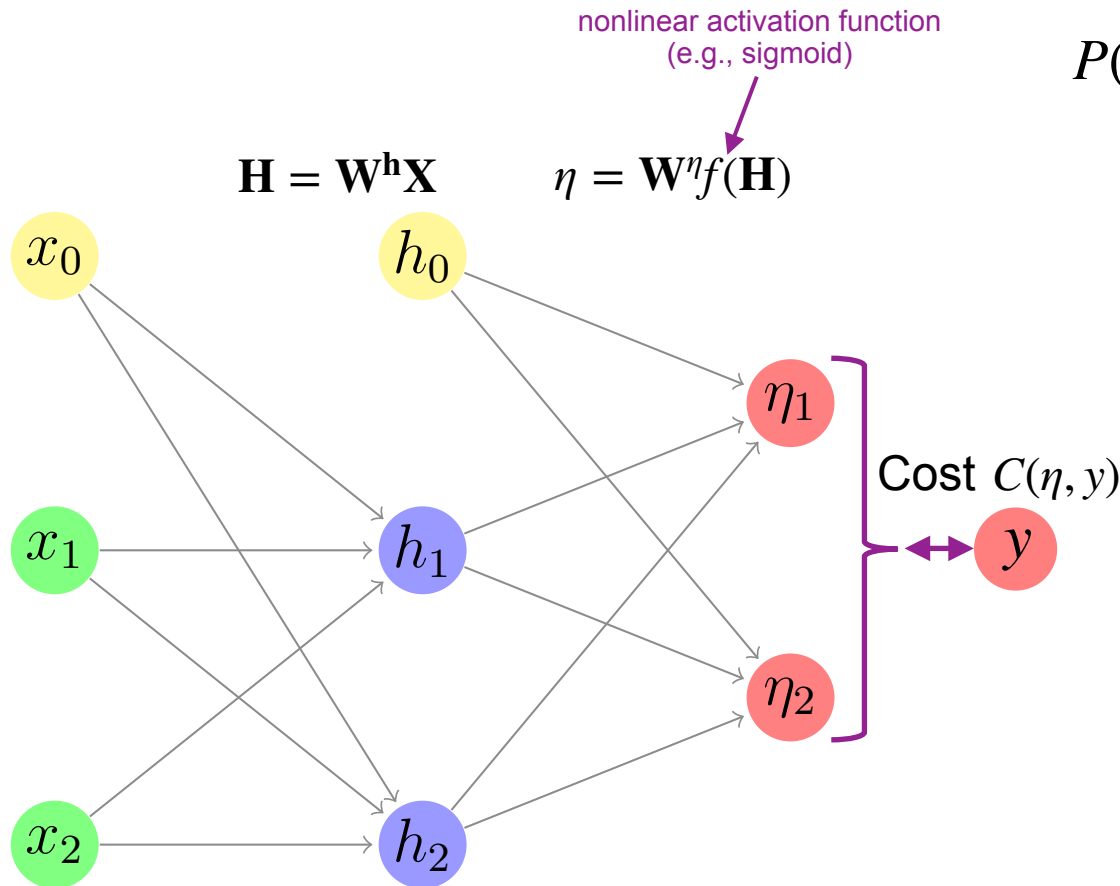
Gradient descent with neural networks

Logistic ("softmax") output prediction:

$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$



Gradient descent with neural networks

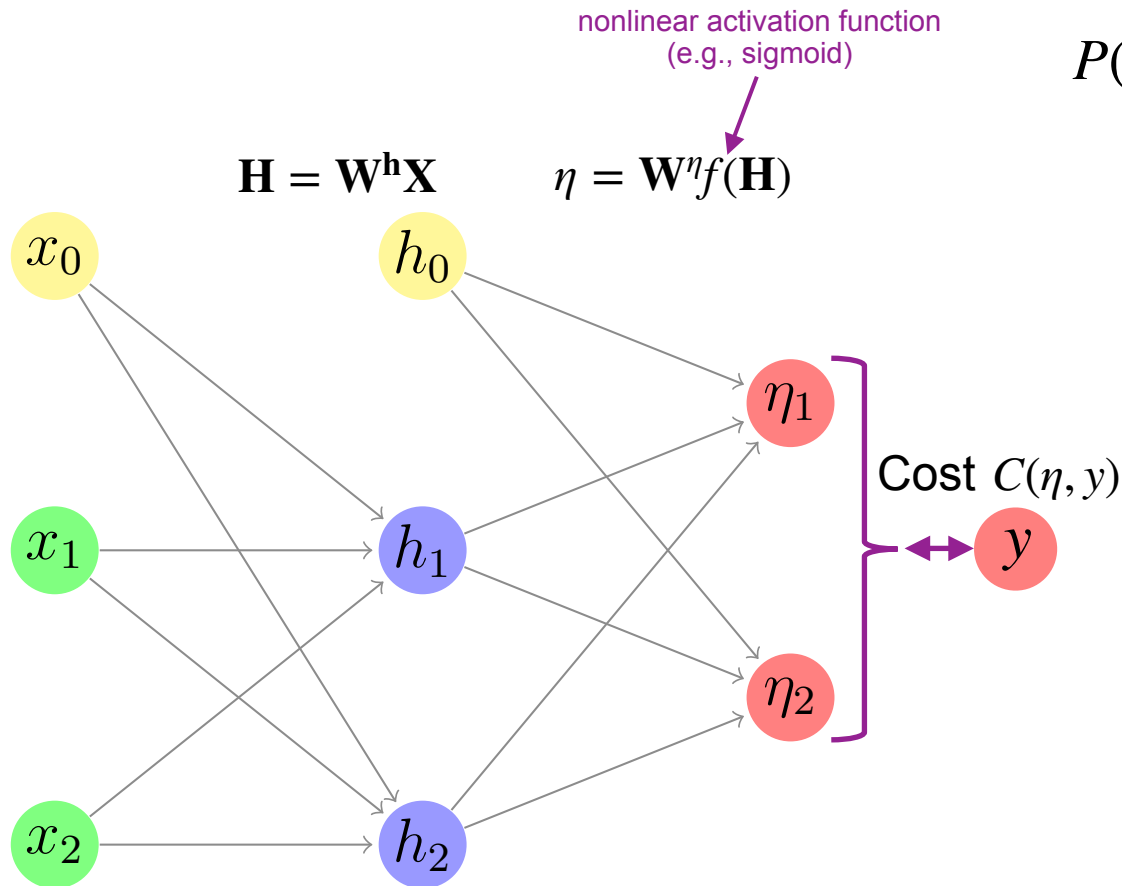


Logistic ("softmax") output prediction:

$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$

To improve the model's weights, we iteratively compute $\nabla_{\mathbf{w}} C(\eta, y)$ and move the weights in that direction

Gradient descent with neural networks



Logistic ("softmax") output prediction:

$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$

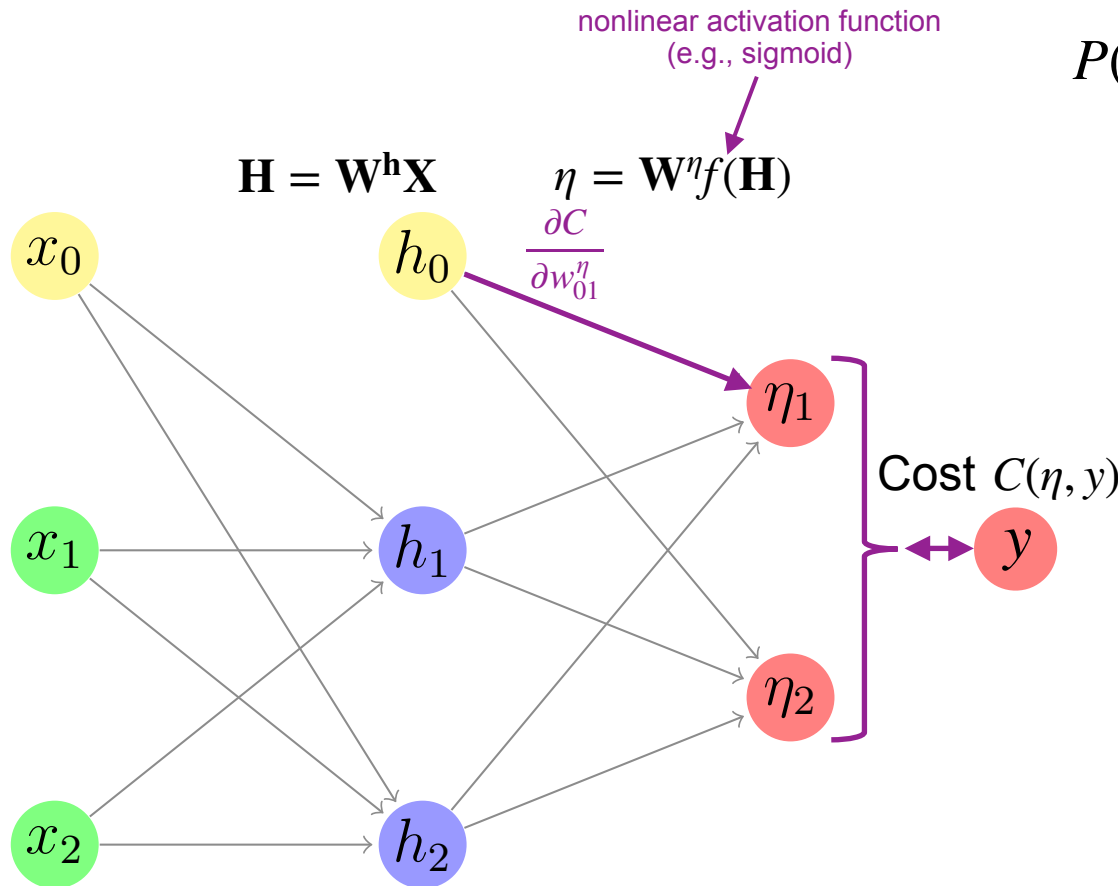
To improve the model's weights, we iteratively compute $\nabla_{\mathbf{w}} C(\eta, y)$ and move the weights in that direction

Chain rule of calculus:

if $y = f(u_1, \dots, u_n)$ and $u_i = g_i(x)$,

$$\text{then } \frac{\partial y}{\partial x} = \sum_{i=1}^n \frac{\partial y}{\partial u_i} \frac{\partial u_i}{\partial x}$$

Gradient descent with neural networks



Logistic ("softmax") output prediction:

$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$

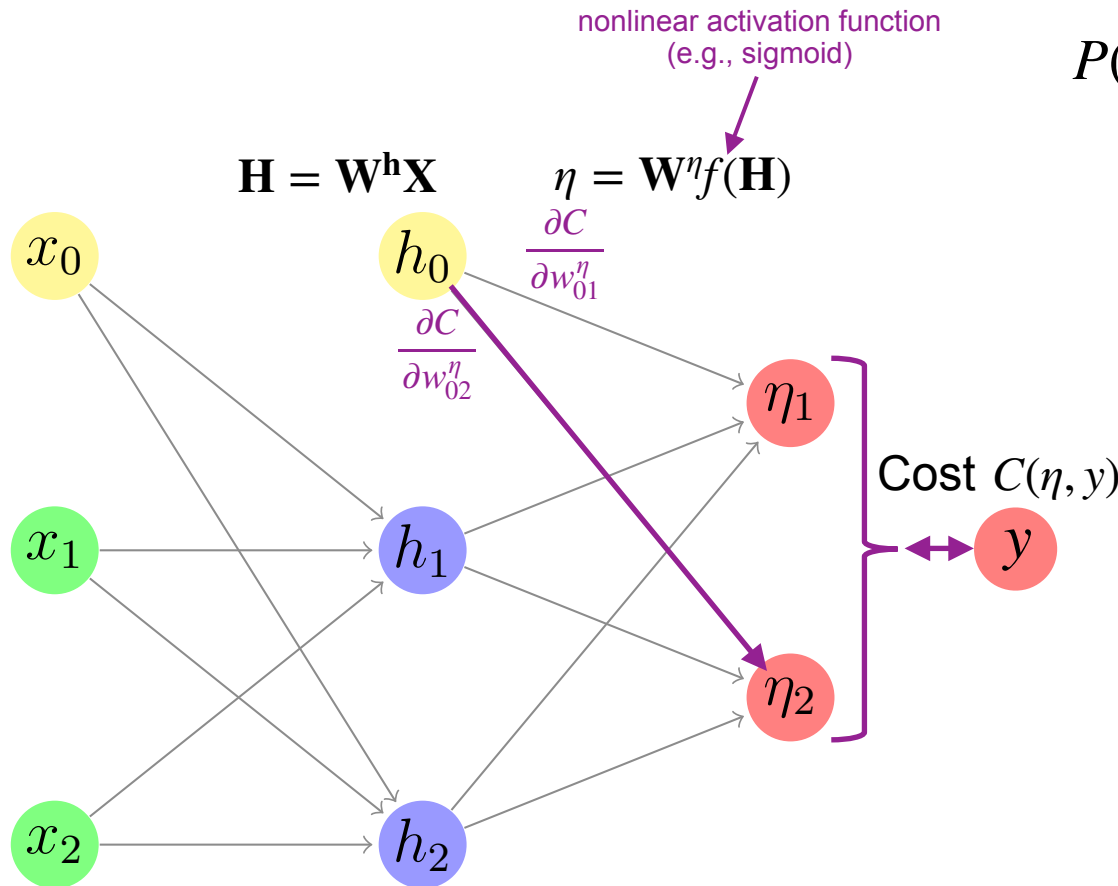
To improve the model's weights, we iteratively compute $\nabla_{\mathbf{w}} C(\eta, y)$ and move the weights in that direction

Chain rule of calculus:

if $y = f(u_1, \dots, u_n)$ and $u_i = g_i(x)$,

$$\text{then } \frac{\partial y}{\partial x} = \sum_{i=1}^n \frac{\partial y}{\partial u_i} \frac{\partial u_i}{\partial x}$$

Gradient descent with neural networks



Logistic ("softmax") output prediction:

$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$

To improve the model's weights, we iteratively compute $\nabla_{\mathbf{w}} C(\eta, y)$ and move the weights in that direction

Chain rule of calculus:

if $y = f(u_1, \dots, u_n)$ and $u_i = g_i(x)$,

$$\text{then } \frac{\partial y}{\partial x} = \sum_{i=1}^n \frac{\partial y}{\partial u_i} \frac{\partial u_i}{\partial x}$$

Gradient descent with neural networks

Logistic ("softmax") output prediction:

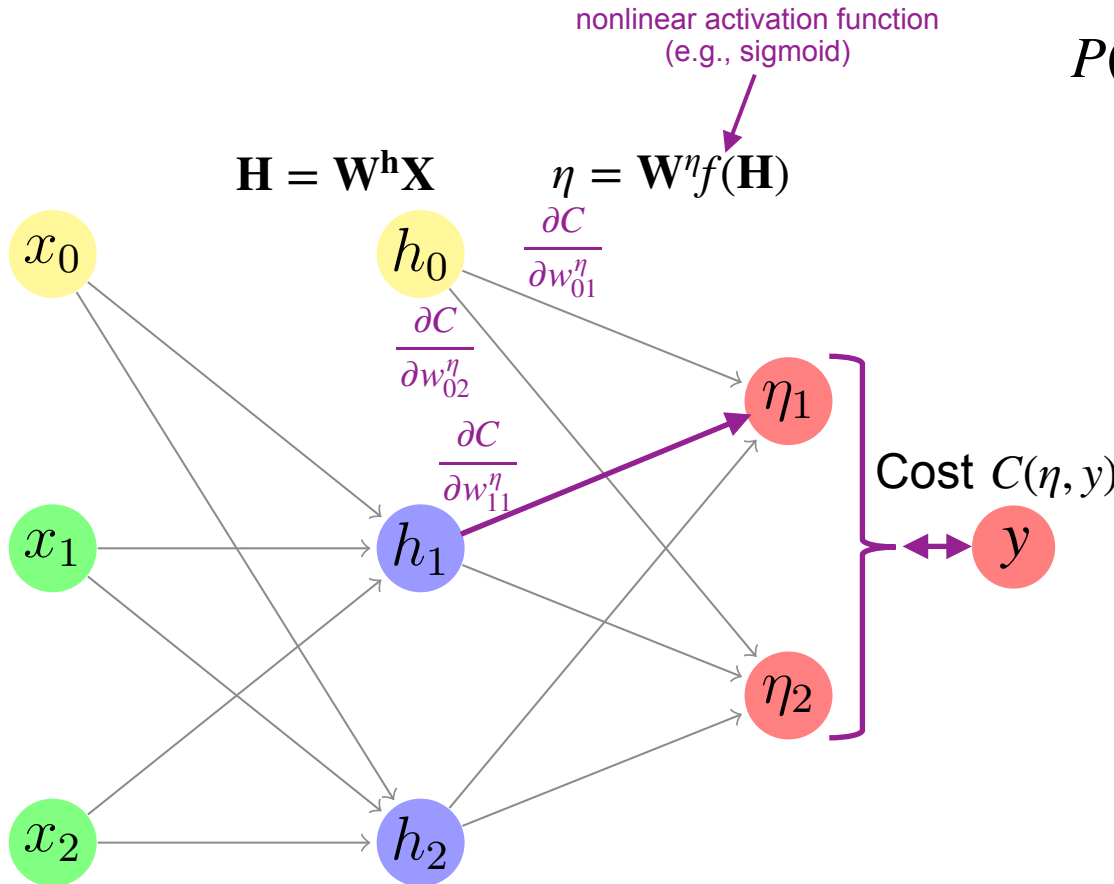
$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$

To improve the model's weights, we iteratively compute $\nabla_{\mathbf{w}} C(\eta, y)$ and move the weights in that direction

Chain rule of calculus:

if $y = f(u_1, \dots, u_n)$ and $u_i = g_i(x)$,

$$\text{then } \frac{\partial y}{\partial x} = \sum_{i=1}^n \frac{\partial y}{\partial u_i} \frac{\partial u_i}{\partial x}$$



Gradient descent with neural networks

Logistic ("softmax") output prediction:

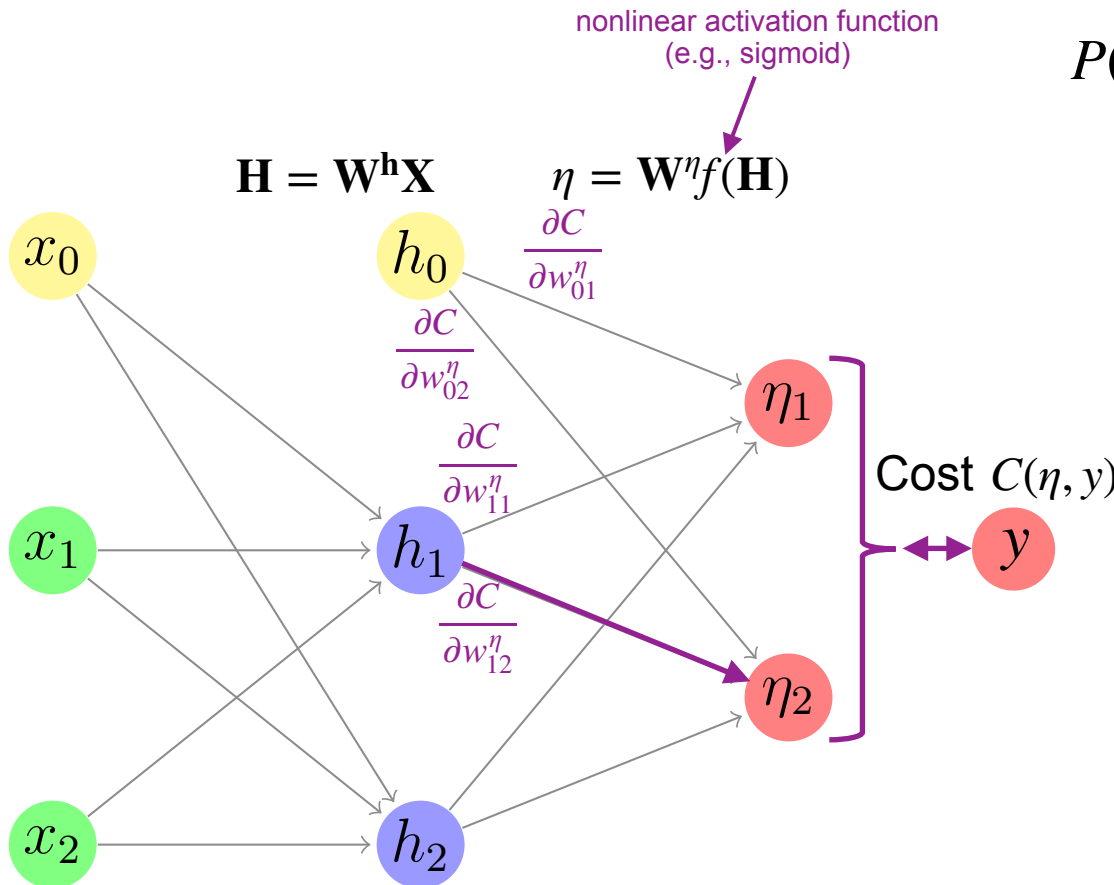
$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$

To improve the model's weights, we iteratively compute $\nabla_{\mathbf{w}} C(\eta, y)$ and move the weights in that direction

Chain rule of calculus:

if $y = f(u_1, \dots, u_n)$ and $u_i = g_i(x)$,

$$\text{then } \frac{\partial y}{\partial x} = \sum_{i=1}^n \frac{\partial y}{\partial u_i} \frac{\partial u_i}{\partial x}$$



Gradient descent with neural networks

Logistic ("softmax") output prediction:

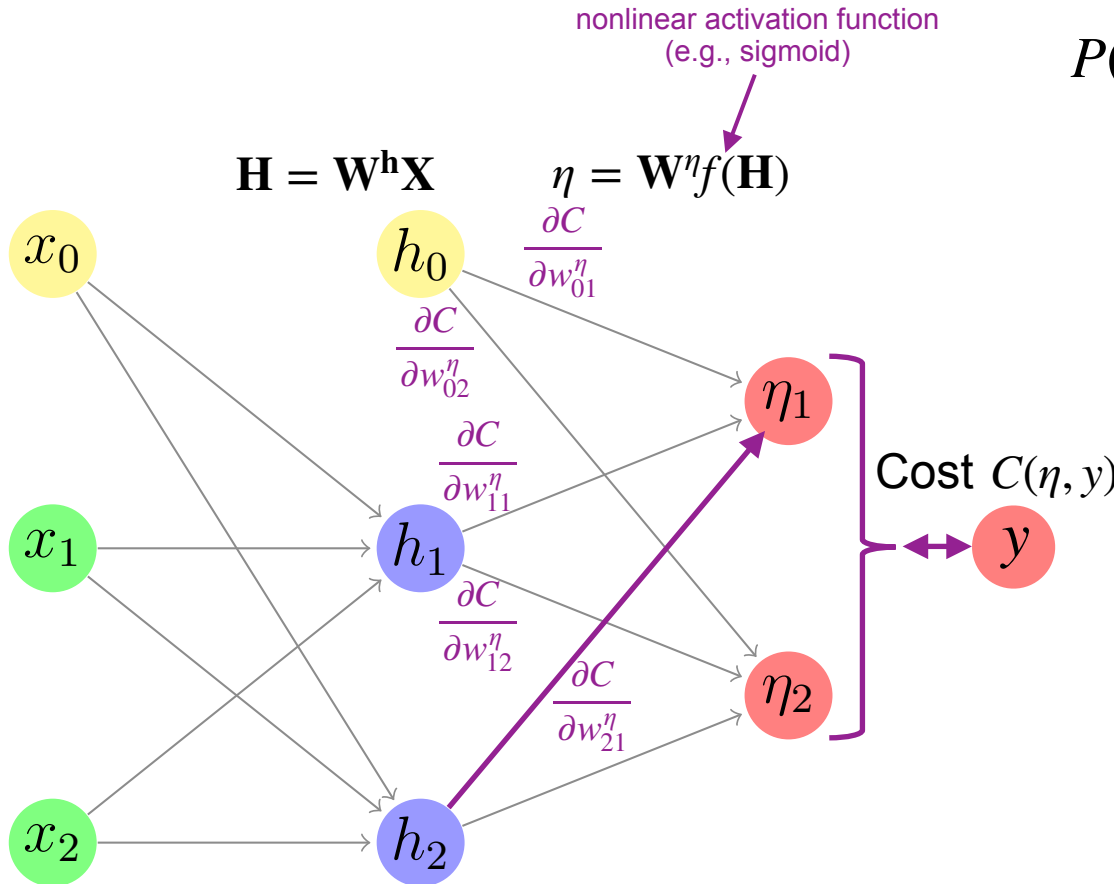
$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$

To improve the model's weights, we iteratively compute $\nabla_{\mathbf{w}} C(\eta, y)$ and move the weights in that direction

Chain rule of calculus:

if $y = f(u_1, \dots, u_n)$ and $u_i = g_i(x)$,

$$\text{then } \frac{\partial y}{\partial x} = \sum_{i=1}^n \frac{\partial y}{\partial u_i} \frac{\partial u_i}{\partial x}$$



Gradient descent with neural networks

Logistic ("softmax") output prediction:

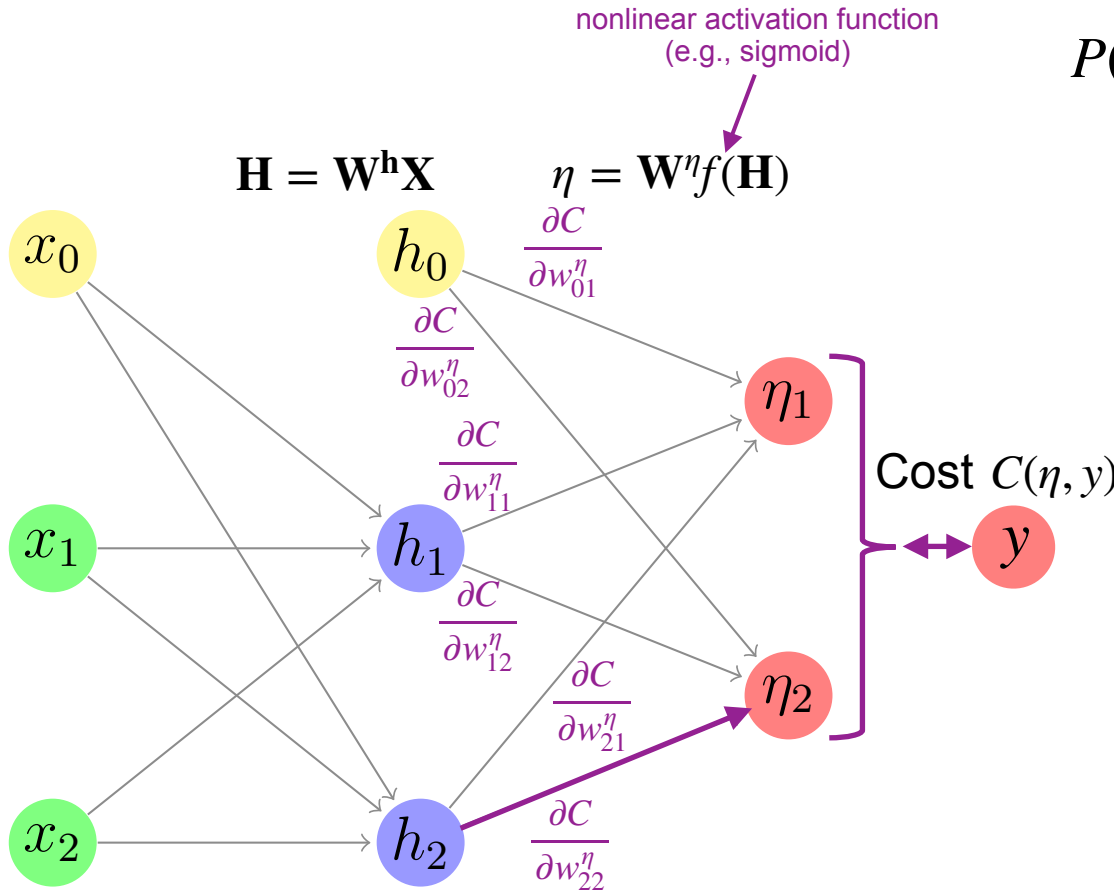
$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$

To improve the model's weights, we iteratively compute $\nabla_{\mathbf{w}} C(\eta, y)$ and move the weights in that direction

Chain rule of calculus:

if $y = f(u_1, \dots, u_n)$ and $u_i = g_i(x)$,

$$\text{then } \frac{\partial y}{\partial x} = \sum_{i=1}^n \frac{\partial y}{\partial u_i} \frac{\partial u_i}{\partial x}$$



Gradient descent with neural networks

Logistic ("softmax") output prediction:

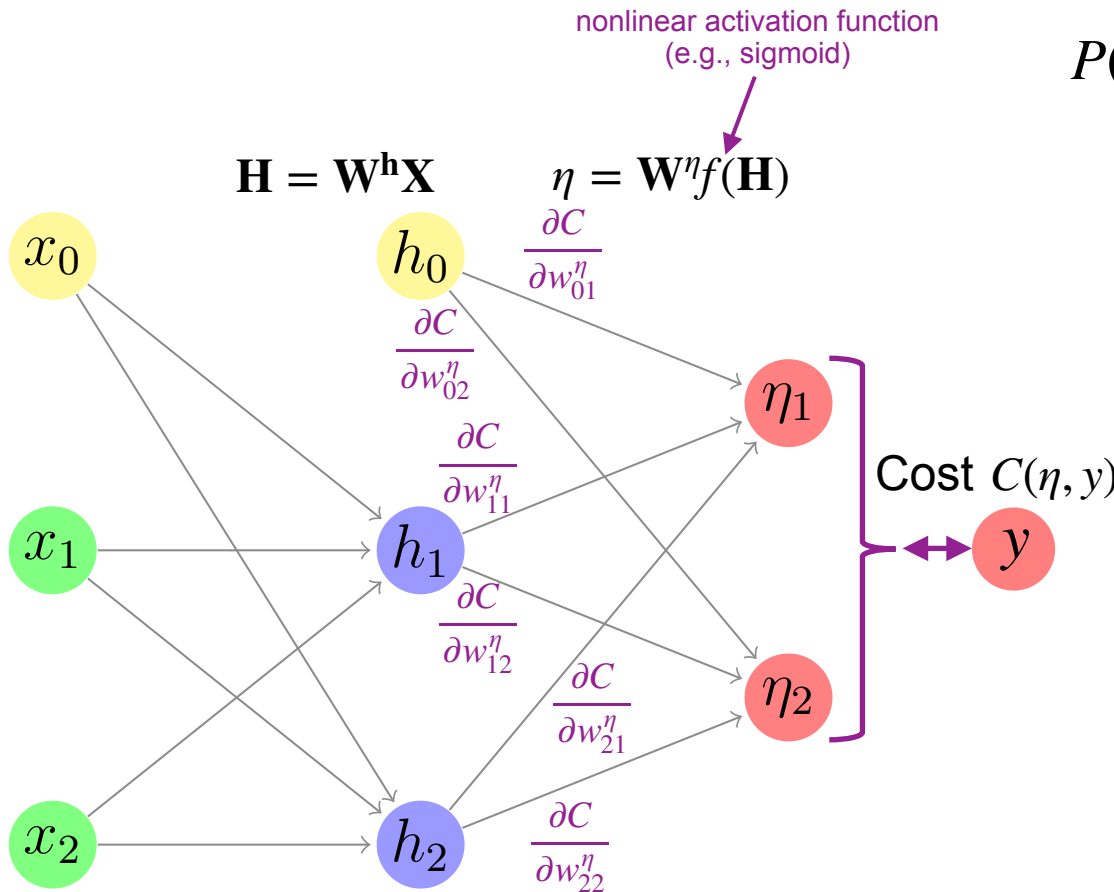
$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$

To improve the model's weights, we iteratively compute $\nabla_{\mathbf{w}} C(\eta, y)$ and move the weights in that direction

Chain rule of calculus:

if $y = f(u_1, \dots, u_n)$ and $u_i = g_i(x)$,

$$\text{then } \frac{\partial y}{\partial x} = \sum_{i=1}^n \frac{\partial y}{\partial u_i} \frac{\partial u_i}{\partial x}$$



Gradient descent with neural networks

Logistic ("softmax") output prediction:

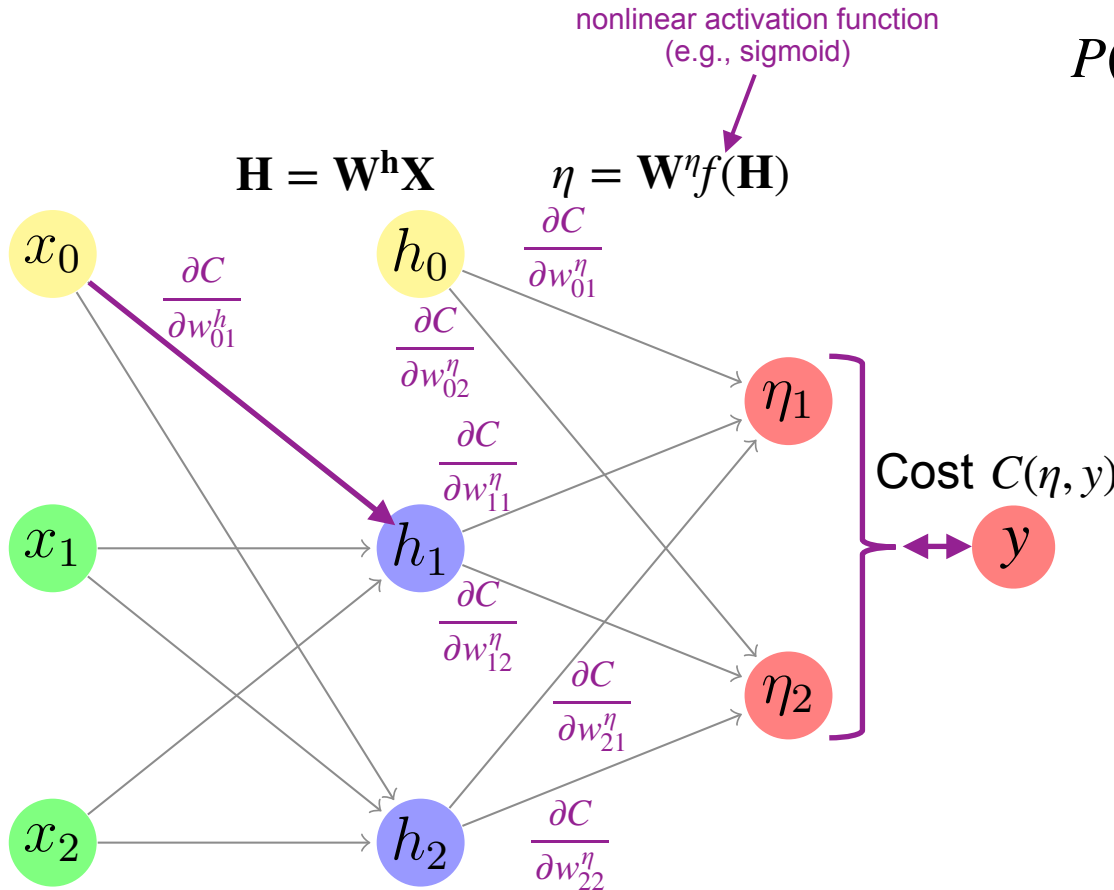
$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$

To improve the model's weights, we iteratively compute $\nabla_{\mathbf{w}} C(\eta, y)$ and move the weights in that direction

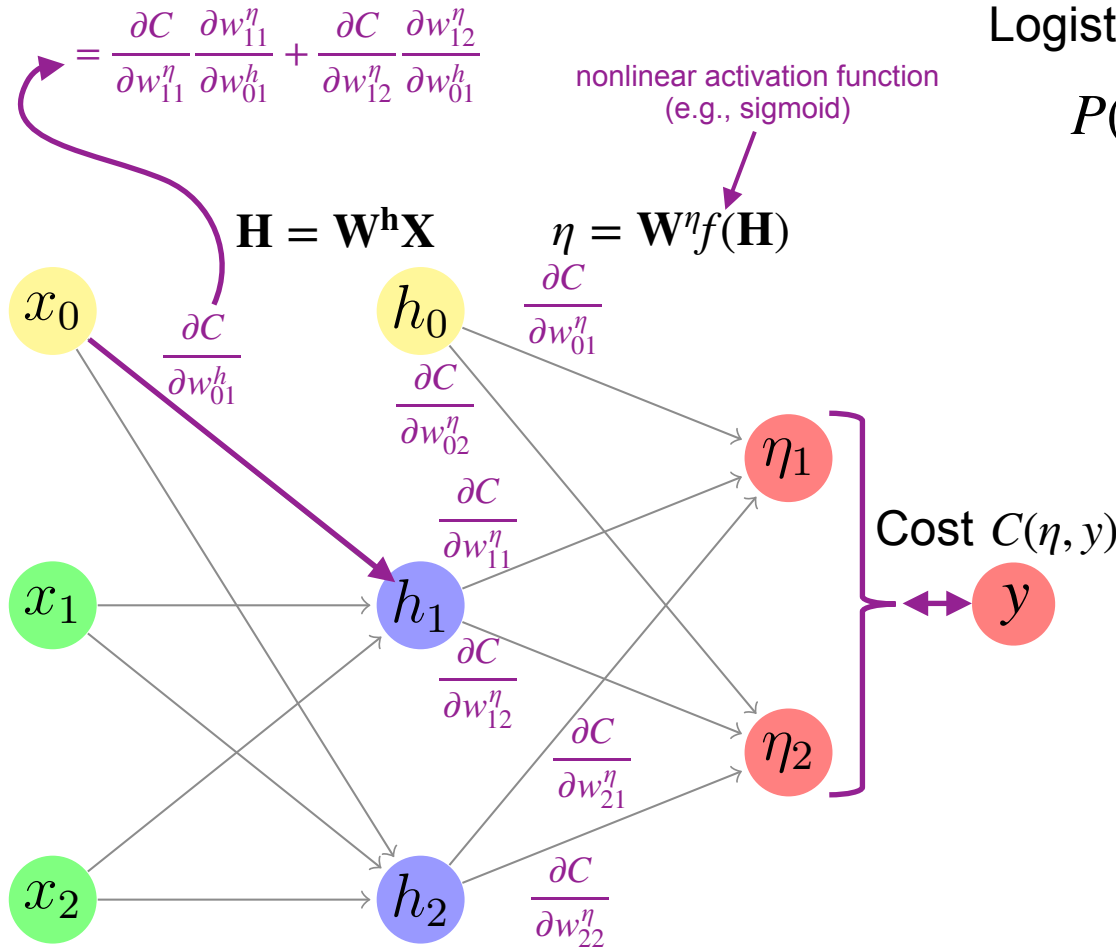
Chain rule of calculus:

if $y = f(u_1, \dots, u_n)$ and $u_i = g_i(x)$,

$$\text{then } \frac{\partial y}{\partial x} = \sum_{i=1}^n \frac{\partial y}{\partial u_i} \frac{\partial u_i}{\partial x}$$



Gradient descent with neural networks



Logistic ("softmax") output prediction:

$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$

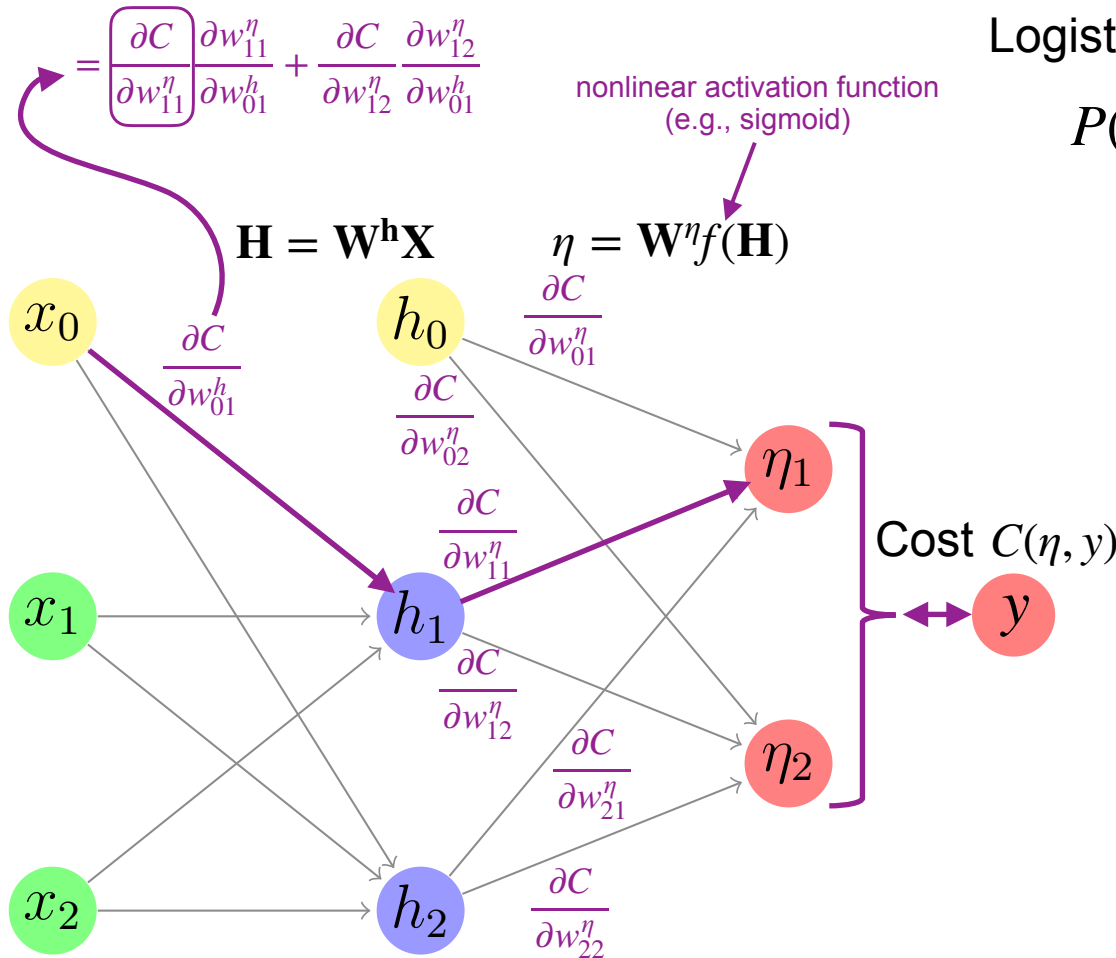
To improve the model's weights, we iteratively compute $\nabla_{\mathbf{w}} C(\eta, y)$ and move the weights in that direction

Chain rule of calculus:

if $y = f(u_1, \dots, u_n)$ and $u_i = g_i(x)$,

$$\text{then } \frac{\partial y}{\partial x} = \sum_{i=1}^n \frac{\partial y}{\partial u_i} \frac{\partial u_i}{\partial x}$$

Gradient descent with neural networks



Logistic ("softmax") output prediction:

$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$

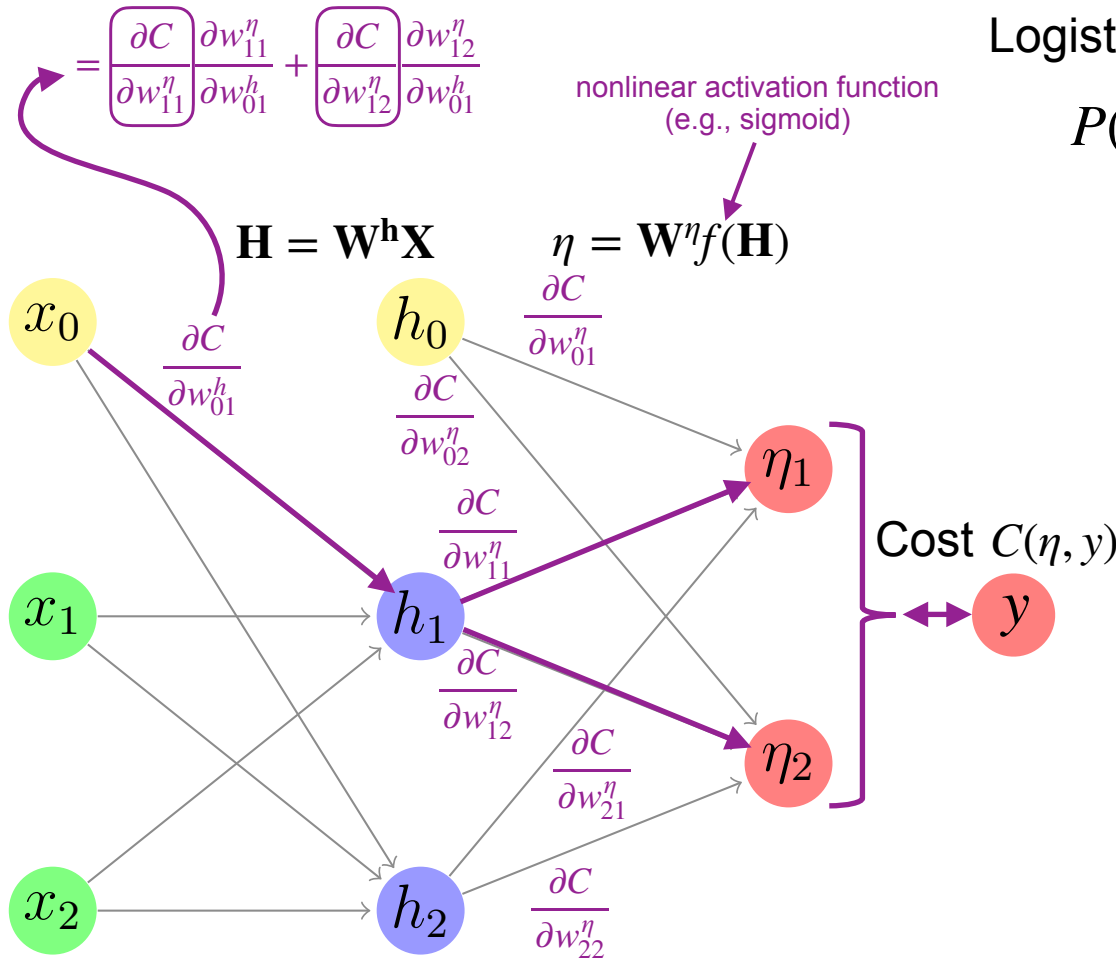
To improve the model's weights, we iteratively compute $\nabla_{\mathbf{w}} C(\eta, y)$ and move the weights in that direction

Chain rule of calculus:

if $y = f(u_1, \dots, u_n)$ and $u_i = g_i(x)$,

$$\text{then } \frac{\partial y}{\partial x} = \sum_{i=1}^n \frac{\partial y}{\partial u_i} \frac{\partial u_i}{\partial x}$$

Gradient descent with neural networks



Logistic ("softmax") output prediction:

$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$

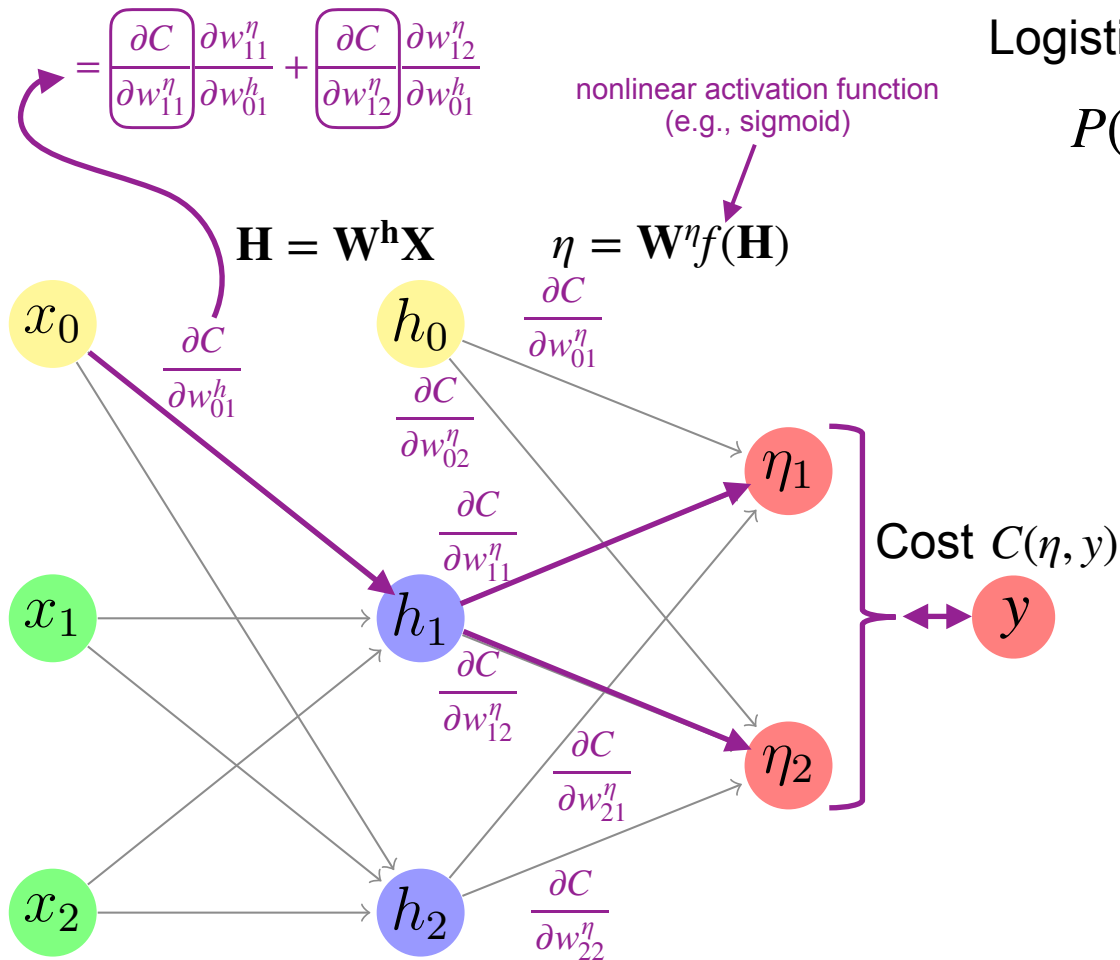
To improve the model's weights, we iteratively compute $\nabla_{\mathbf{w}} C(\eta, y)$ and move the weights in that direction

Chain rule of calculus:

if $y = f(u_1, \dots, u_n)$ and $u_i = g_i(x)$,

$$\text{then } \frac{\partial y}{\partial x} = \sum_{i=1}^n \frac{\partial y}{\partial u_i} \frac{\partial u_i}{\partial x}$$

Gradient descent with neural networks



Logistic ("softmax") output prediction:

$$P(y = 1 | x_1, x_2) = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}}$$

To improve the model's weights, we iteratively compute $\nabla_{\mathbf{w}} C(\eta, y)$ and move the weights in that direction

Chain rule of calculus:

if $y = f(u_1, \dots, u_n)$ and $u_i = g_i(x)$,

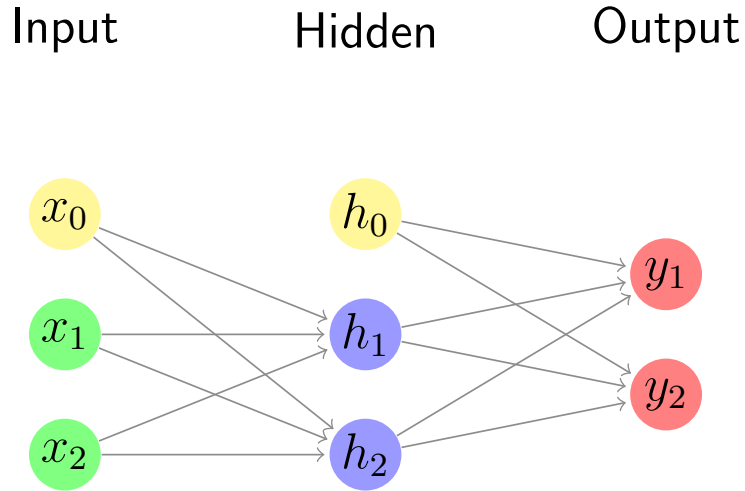
$$\text{then } \frac{\partial y}{\partial x} = \sum_{i=1}^n \frac{\partial y}{\partial u_i} \frac{\partial u_i}{\partial x}$$

This reuse of partially computed results (here, $\frac{\partial C}{\partial w_{11}^{\eta}}$ and $\frac{\partial C}{\partial w_{12}^{\eta}}$) is what is called

BACKPROPAGATION*

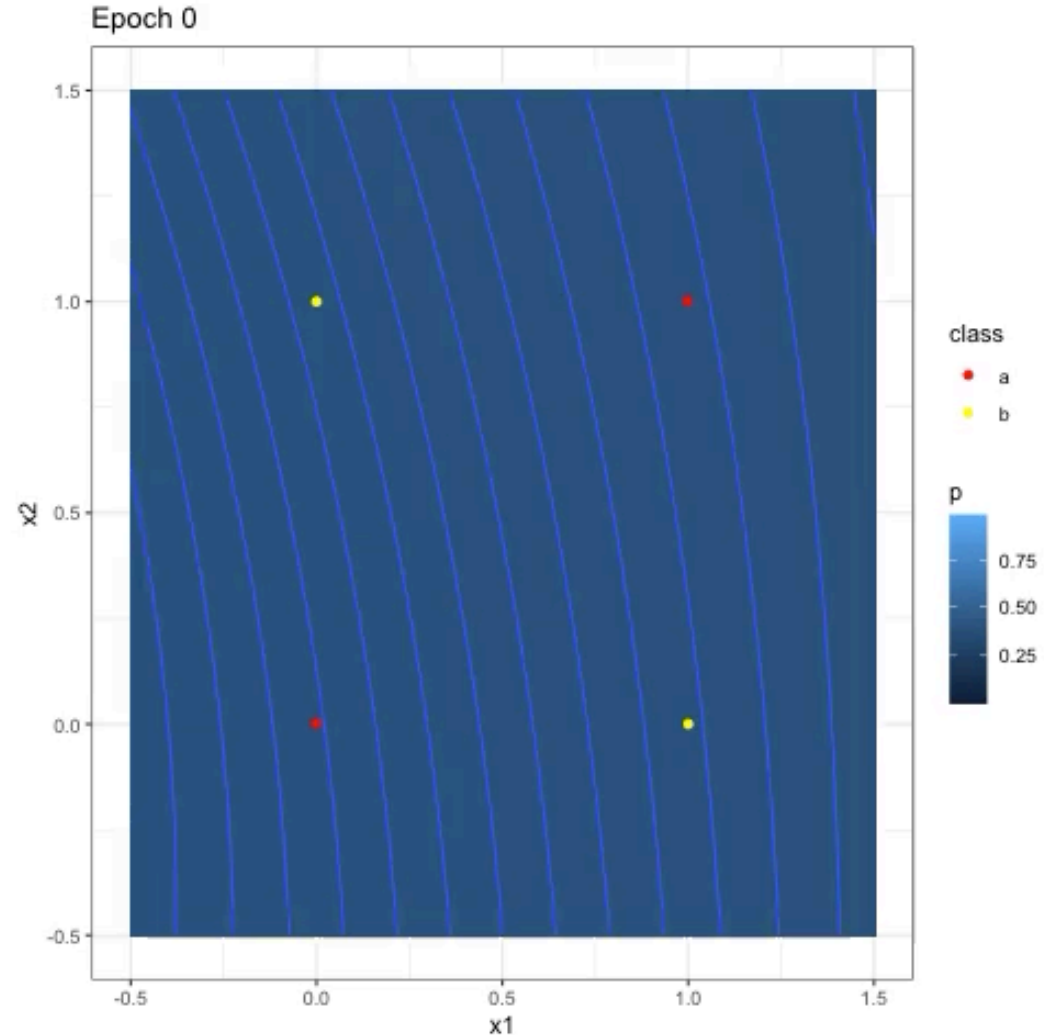
(*An instance of **dynamic programming**. Technically, the stored outputs of intermediate computations are not $\frac{\partial C}{\partial w}$ terms themselves, but gradients for node values, from which the weight gradients can be easily computed.)

Learning XOR with one hidden layer

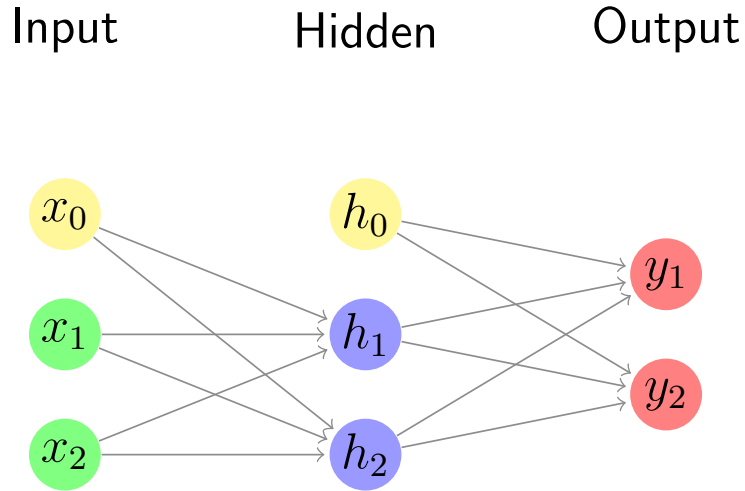


Initialize weights **randomly**

In each learning **epoch**,
collect gradient from the 4
datapoints, and move weights
"a bit" in direction of gradient

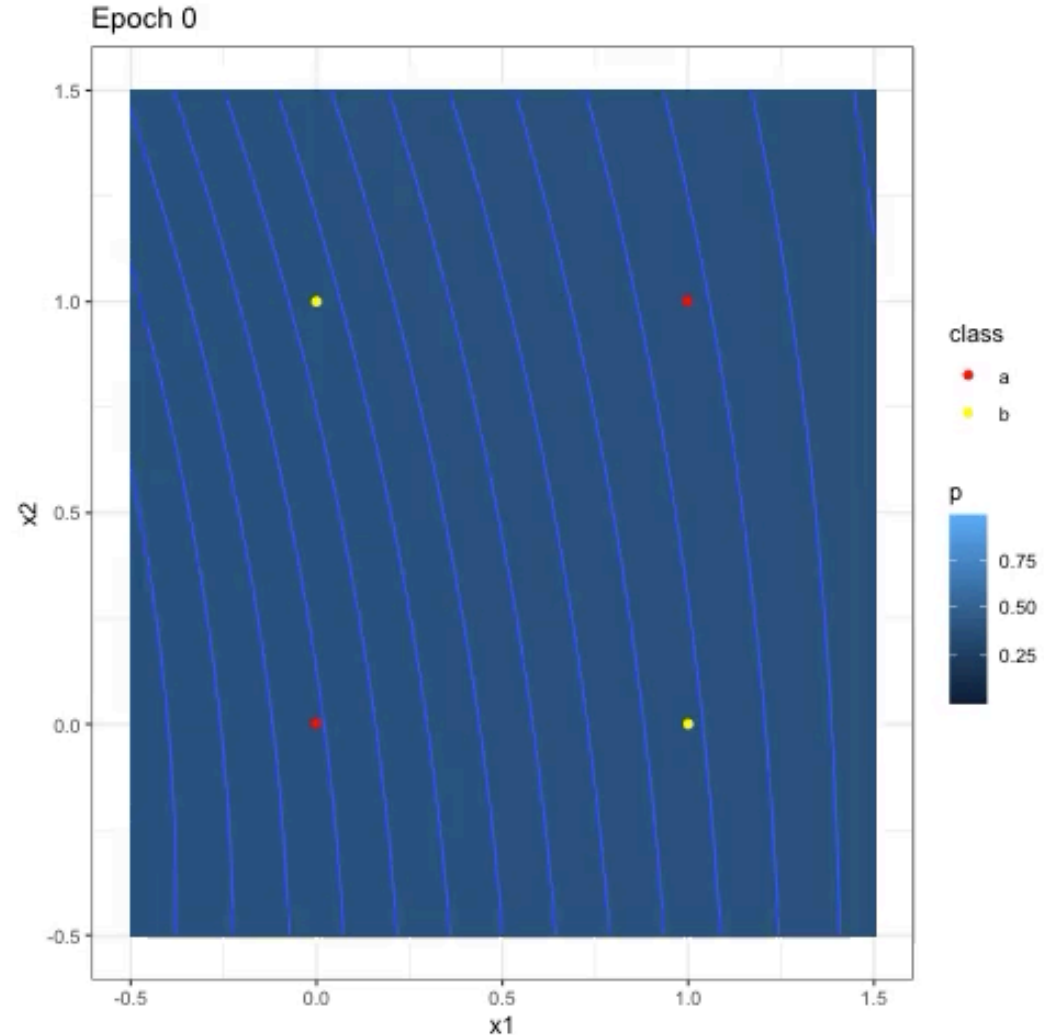


Learning XOR with one hidden layer

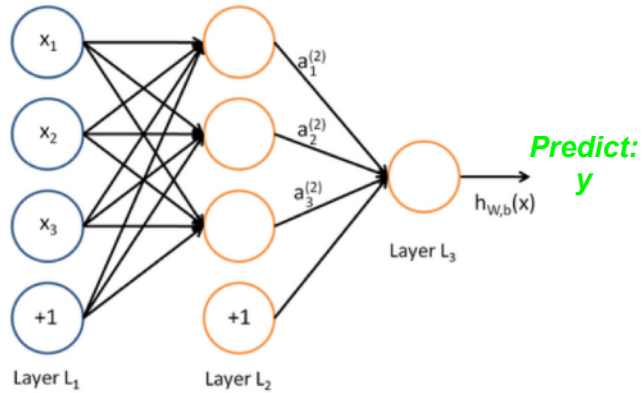


Initialize weights **randomly**

In each learning **epoch**,
collect gradient from the 4
datapoints, and move weights
"a bit" in direction of gradient

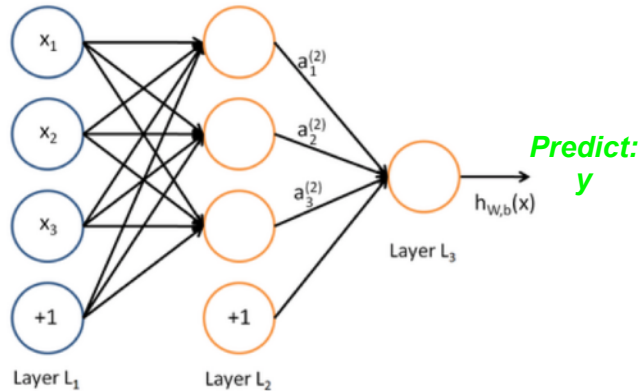


Expressive power of multilayer network



$$g(x_1, \dots, x_n) = y$$

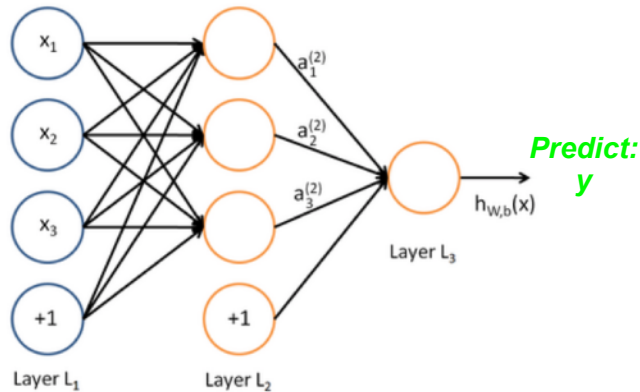
Expressive power of multilayer network



$$g(x_1, \dots, x_n) = y$$

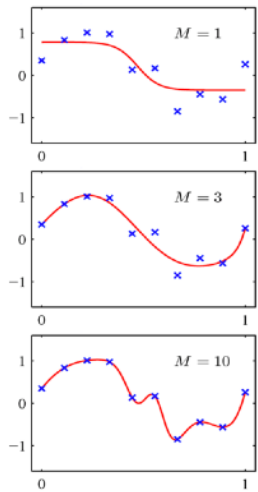
- Even just one hidden layer makes a neural network a ***universal function approximator*** (Hornik et al., 1989)

Expressive power of multilayer network

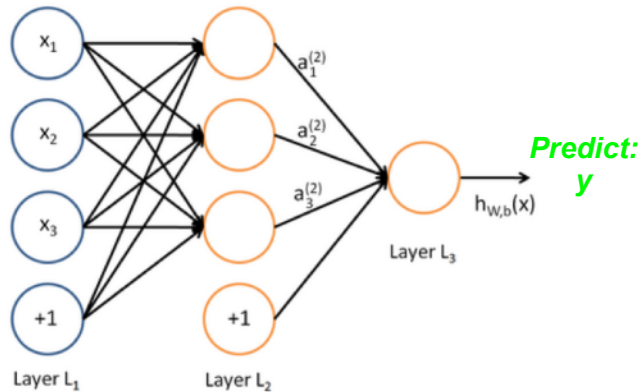


$$g(x_1, \dots, x_n) = y$$

- Even just one hidden layer makes a neural network a **universal function approximator** (Hornik et al., 1989)

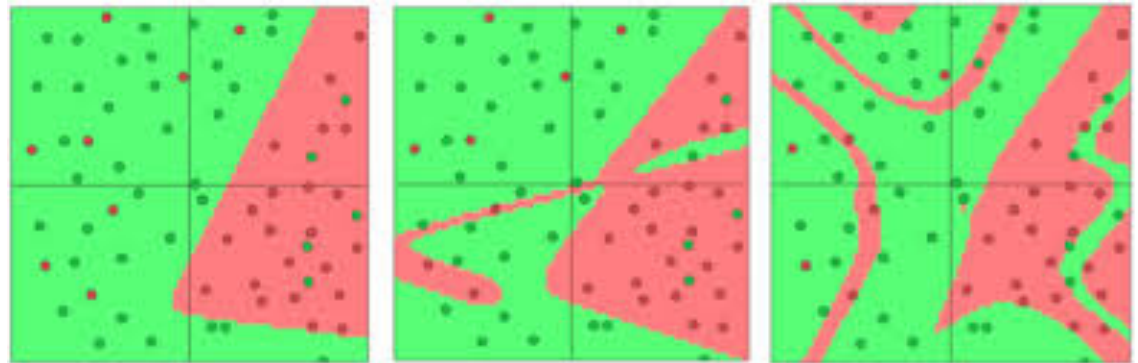
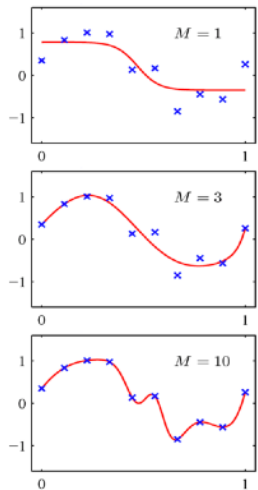


Expressive power of multilayer network

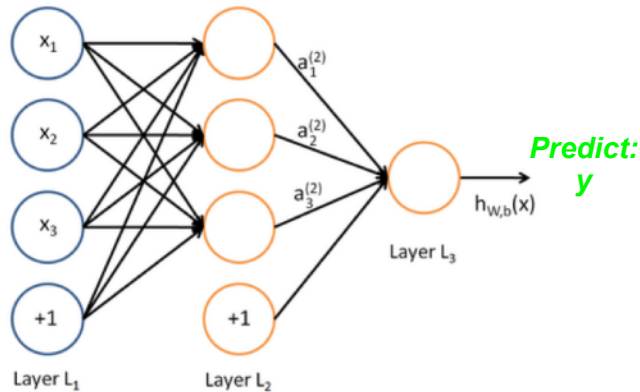


$$g(x_1, \dots, x_n) = y$$

- Even just one hidden layer makes a neural network a **universal function approximator** (Hornik et al., 1989)

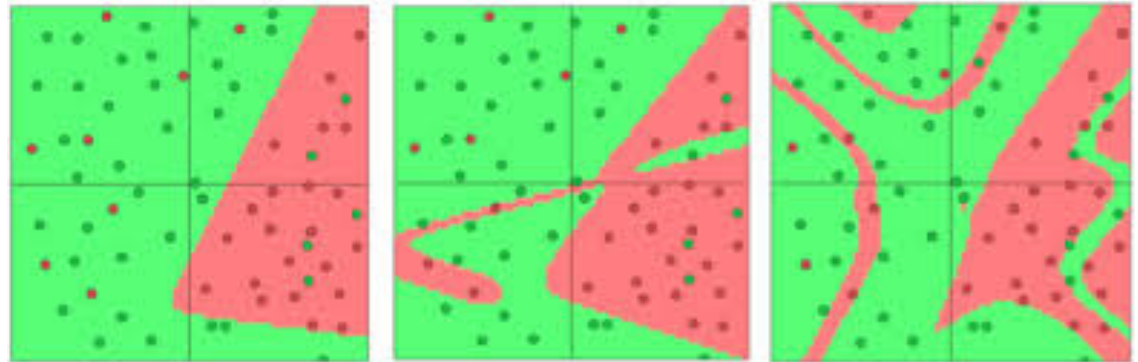
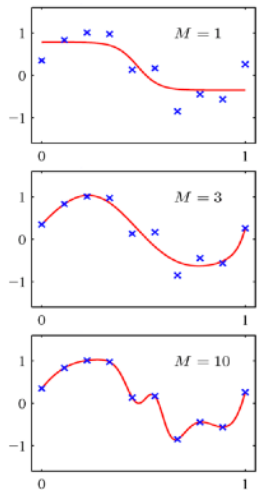


Expressive power of multilayer network



$$g(x_1, \dots, x_n) = y$$

- Even just one hidden layer makes a neural network a **universal function approximator** (Hornik et al., 1989)



- Challenge: **how to learn best function approximation?**

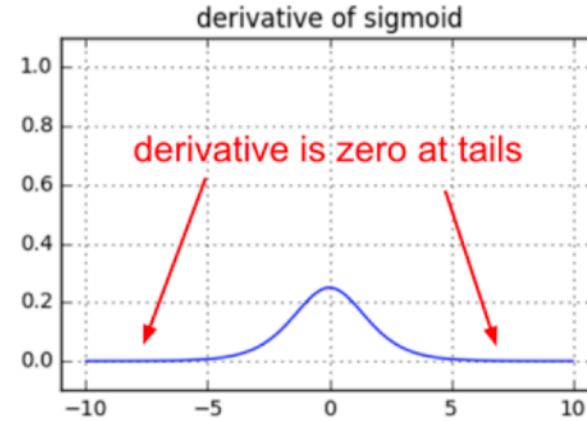
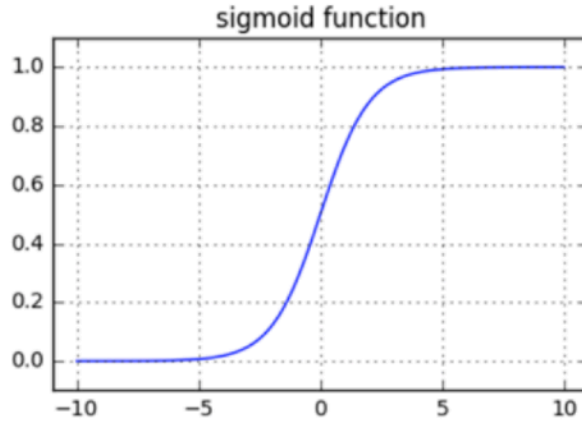
Changing activation functions

Changing activation functions

- Using sigmoid as non-linear activation function $f(\mathbf{H})$ has problems when you add more network layers

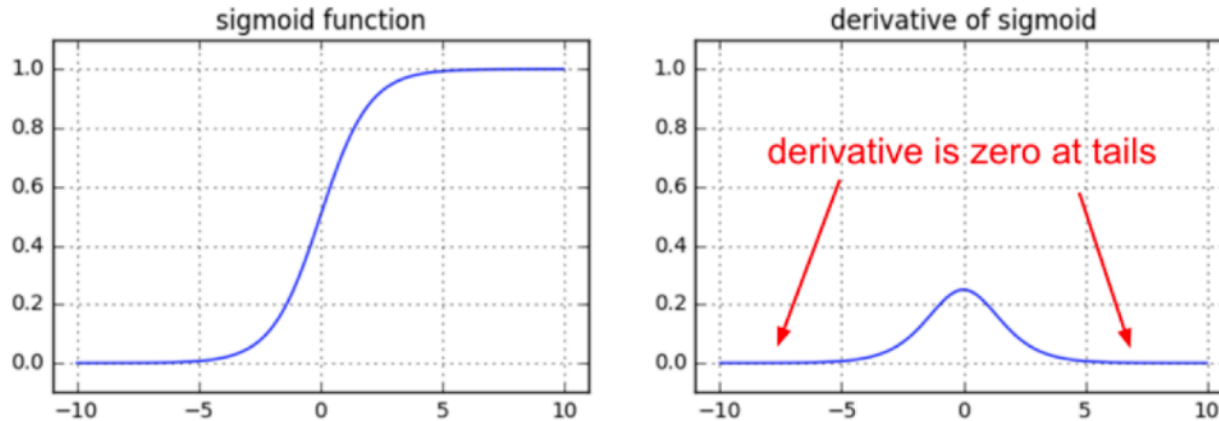
Changing activation functions

- Using sigmoid as non-linear activation function $f(\mathbf{H})$ has problems when you add more network layers



Changing activation functions

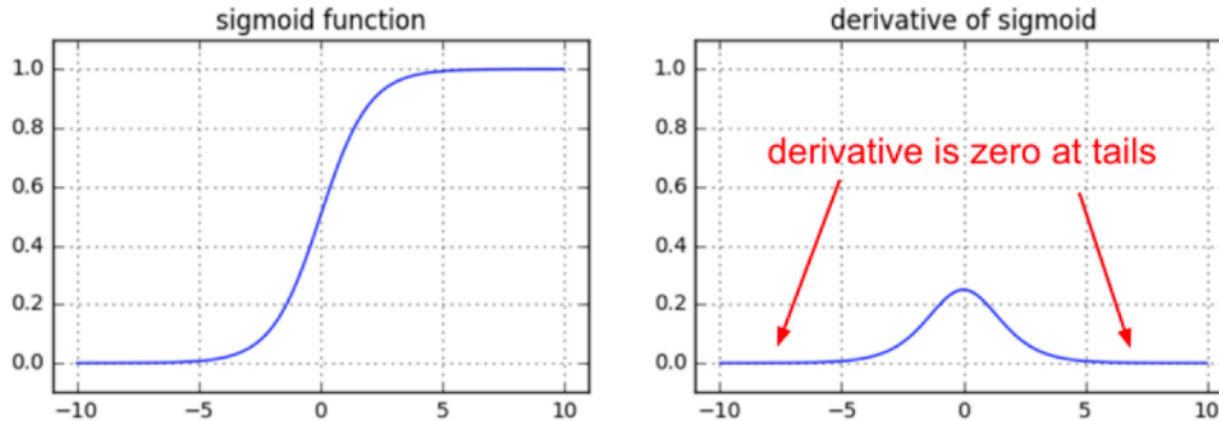
- Using sigmoid as non-linear activation function $f(\mathbf{H})$ has problems when you add more network layers



- Thus other functions for $f(\mathbf{H})$ have become more popular

Changing activation functions

- Using sigmoid as non-linear activation function $f(\mathbf{H})$ has problems when you add more network layers

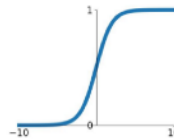


- Thus other functions for $f(\mathbf{H})$ have become more popular

Activation Functions

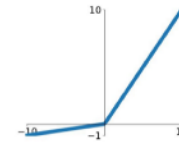
Sigmoid

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



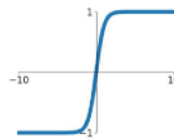
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

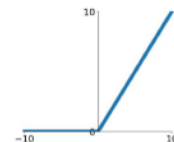


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

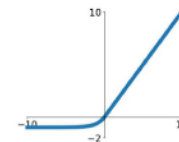
ReLU

$$\max(0, x)$$



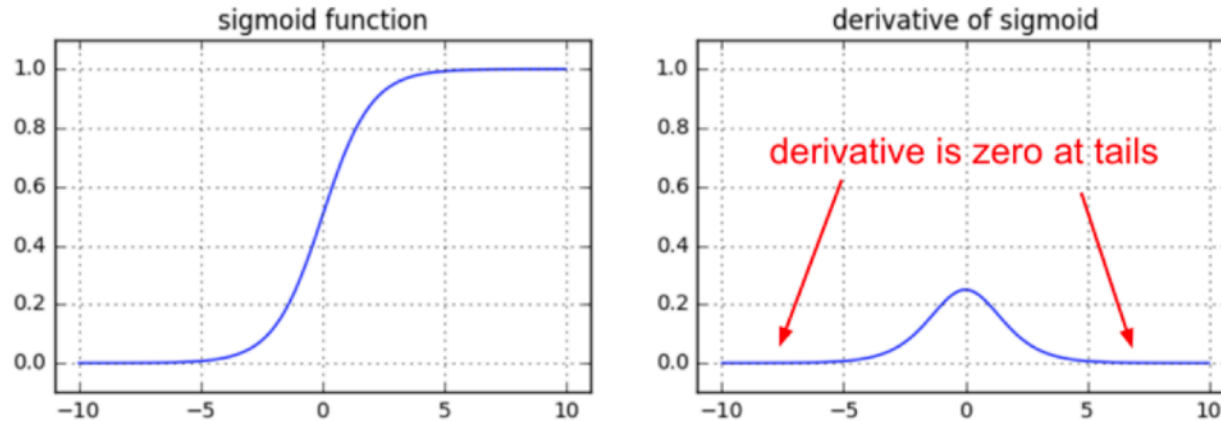
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Changing activation functions

- Using sigmoid as non-linear activation function $f(\mathbf{H})$ has problems when you add more network layers

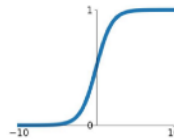


- Thus other functions for $f(\mathbf{H})$ have become more popular

Activation Functions

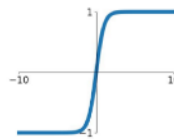
Sigmoid

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



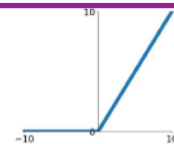
tanh

$$\tanh(x)$$



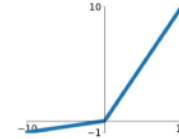
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

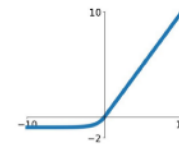


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Online resources for learning more

Backpropagation:

Backprop as derivatives on computation graphs: <http://colah.github.io/posts/2015-08-Backprop/>

Lecture by Richard Socher (especially first ~18min) at https://www.youtube.com/watch?v=isPiE-DBagM&list=PL3FW7Lu3i5Jsnh1rnUwq_TcyINr7EkRe6

Worked numerical example: <https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>

More generally, RNNs in natural language processing:

<https://learning-modules.mit.edu/class/index.html?uuid=/course/6/fa17/6.864#info>

<http://web.stanford.edu/class/cs224n/>

<http://cs231n.github.io>

<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1), 1-309. [Available for PDF download through MIT Libraries]

(And if you recommend another resource not listed here, let me know at rplevy@mit.edu!)