25: Logistic Regression

Lisa Yan

November 18, 2019

Multinomial MLE and MAP

Review

Model:

Multinomial with m outcomes:
p_i probability of outcome i

Observe:

n_i = # of trials with outcome i
Total of $\sum_{i=1}^{m} n_i$ trials

MLE

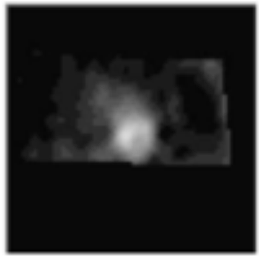
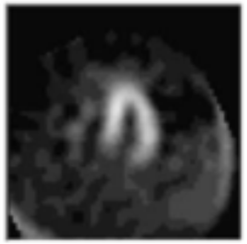
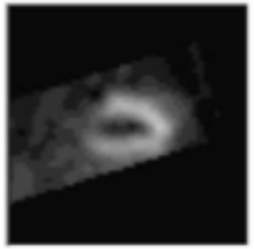
$$p_i = \frac{n_i}{\sum_{i=1}^{m} n_i}$$

MAP with Laplace smoothing
(Laplace estimate)

$$p_i = \frac{n_i + 1}{\sum_{i=1}^{m} n_i + m}$$

Classification problem

Review

	Feature 1	Feature 2		Feature 100	Output
Patient 1	1	0	…	1	1
Patient 2	1	1	…	0	0
…			⋮		⋮
Patient n	0	0	…	1	1

$$\hat{Y} = \arg\max_{y=\{0,1\}} \hat{P}(Y \mid \boldsymbol{X})$$

(Predict the Y that is most likely given our observation $\boldsymbol{X}$)

Training: Train set notation errata

Review

Training data: $(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \ldots, (\boldsymbol{x}^{(n)}, y^{(n)})$ n datapoints

Notation consistent with lecture notes (last lecture has been updated):

i-th observation: $\boldsymbol{x}^{(i)} = \left(x_1^{(i)}, x_2^{(i)}, \ldots, x_m^{(i)}\right)$

j-th feature of i-th observation: $x_j^{(i)}$

Brute force Bayes Classifier

Review

$$\hat{Y} = \arg\max_{y=\{0,1\}} \hat{P}(Y \mid \boldsymbol{X})$$

(Predict the Y that is most likely given our observation $\boldsymbol{X}$)

$$= \arg\max_{y=\{0,1\}} \frac{\hat{P}(\boldsymbol{X}|Y)\hat{P}(Y)}{\hat{P}(\boldsymbol{X})}$$

(Bayes' Theorem)

$$= \arg\max_{y=\{0,1\}} \boxed{\hat{P}(\boldsymbol{X}|Y)}\hat{P}(Y)$$

(eliminate normalization constant $\hat{P}(\boldsymbol{X})$)

$$\rightarrow \hat{P}(X_1, X_2, \ldots, X_m|Y)$$

Use MLE or Laplace estimates to find $\hat{P}(X_1, X_2, \ldots, X_m|Y)$ and Y

- $\hat{P}(X_1, X_2, \ldots, X_m|Y = 1)$: Multinomial, 2^m outcomes
- $\hat{P}(X_1, X_2, \ldots, X_m|Y = 0)$: Multinomial, 2^m outcomes
- $\hat{P}(Y)$: Multinomial, 2 outcomes

Total # parameters: $O(2^m)$

The problem with our Brute force Bayes classifier

Review

$$\hat{Y} = \arg\max_{y=\{0,1\}} \hat{P}(Y \mid \boldsymbol{X})$$

(Predict the Y that is most likely given our observation $\boldsymbol{X}$)

$$= \arg\max_{y=\{0,1\}} \frac{\hat{P}(\boldsymbol{X}|Y)\hat{P}(Y)}{\hat{P}(\boldsymbol{X})}$$

(Bayes' Theorem)

$$= \arg\max_{y=\{0,1\}} \hat{P}(\boldsymbol{X}|Y)\hat{P}(Y)$$

(eliminate normalization constant $\hat{P}(\boldsymbol{X})$)

$$\rightarrow \hat{P}(X_1, X_2, \dots, X_m|Y)$$

too many parameters to estimate

What if we could make a simplifying (but naïve) assumption– that $X_1, \dots, X_m$ are **conditionally independent** given Y?

The Naïve Bayes assumption

$$\hat{Y} = \arg\max_{y=\{0,1\}} \hat{P}(Y \mid \boldsymbol{X})$$

(Predict the Y that is most likely given our observation $\boldsymbol{X}$)

$$= \arg\max_{y=\{0,1\}} \frac{\hat{P}(\boldsymbol{X}|Y)\hat{P}(Y)}{\hat{P}(\boldsymbol{X})}$$

(Bayes' Theorem)

$$= \arg\max_{y=\{0,1\}} \hat{P}(\boldsymbol{X}|Y)\hat{P}(Y)$$

(eliminate normalization constant $\hat{P}(\boldsymbol{X})$)

$$= \arg\max_{y=\{0,1\}} \left(\prod_{i=1}^{m} \hat{P}(X_i|Y)\right)\hat{P}(Y)$$

Naïve Bayes Assumption

$X_1, \ldots, X_m$ are **conditionally independent** given Y.

Today's plan

➡ Naïve Bayes

Logistic Regression

Naïve Bayes Classifier

$$\hat{Y} = \arg\max_{y=\{0,1\}} \left(\prod_{i=1}^{m} \hat{P}(X_i|Y) \right) \hat{P}(Y)$$

Training

Use MLE or Laplace (MAP) $\hat{P}(X_i|Y=0), \hat{P}(X_i|Y=1), \forall i$

$\hat{P}(Y=0), \hat{P}(Y=1)$

Total # params: $O(m)$

Testing

$$\hat{Y} = \arg\max_{y=\{0,1\}} \left(\log \hat{P}(Y) + \sum_{i=1}^{m} \log \hat{P}(X_i|Y) \right)$$

(for numeric stability)

NETFLIX

and Learn

Naïve Bayes for TV shows

Will a user like the Pokémon TV series?

Input indicator variables $\boldsymbol{X} = (X_1, X_2)$:

$X_1 = 1$:
"likes Star Wars"

$X_2 = 1$:
"likes Harry Potter"

Output Y indicator:

$Y = 1$:
"likes Pokémon"

Training: Naïve Bayes for TV shows (**MLE**)

$$\hat{Y} = \underset{y=\{0,1\}}{\arg\max} \left(\prod_{i=1}^{m} \hat{P}(X_i|Y) \right) \hat{P}(Y)$$

Observe indicator vars. $\boldsymbol{X} = (X_1, X_2)$:

- X_1: “likes Star Wars”
- X_2: “likes Harry Potter”

Predict Y: “likes Pokémon”

Y \ X_1	0	1
0	3	10
1	4	13

Y \ X_2	0	1
0	5	8
1	7	10

Training data counts

1. How many datapoints (n) are in our train data?
2. How many parameters do we need to estimate?
3. Compute MLE estimates for $\hat{P}(X_1|Y)$:

🤔

Training: Naïve Bayes for TV shows (MLE)

$$\hat{Y} = \underset{y=\{0,1\}}{\arg\max} \left(\prod_{i=1}^{m} \hat{P}(X_i|Y) \right) \hat{P}(Y)$$

Observe indicator vars. $\boldsymbol{X} = (X_1, X_2)$:

- X_1: “likes Star Wars”
- X_2: “likes Harry Potter”

Predict Y: “likes Pokémon”

Y \ X_1	0	1
0	3	10
1	4	13

Y \ X_2	0	1
0	5	8
1	7	10

Training data counts

1. How many datapoints (n) are in our train data?

$n = 30$

2. How many parameters do we need to estimate?

- $\hat{P}(X_1|Y=0), \hat{P}(X_1|Y=1)$: 4 params
- $\hat{P}(X_2|Y=0), \hat{P}(X_2|Y=1)$: 4 params
- $\hat{P}(Y)$: 2 params

3. Compute MLE estimates for $\hat{P}(X_1|Y)$:

Training: Naïve Bayes for TV shows (MLE)

$$\hat{Y} = \underset{y=\{0,1\}}{\arg\max} \left(\prod_{i=1}^{m} \hat{P}(X_i|Y)\right) \hat{P}(Y)$$

Observe indicator vars. $\boldsymbol{X} = (X_1, X_2)$:

- X_1: “likes Star Wars”
- X_2: “likes Harry Potter”

Predict Y: “likes Pokémon”

$Y \backslash X_1$	0	1
0	3	10
1	4	13

$Y \backslash X_2$	0	1
0	5	8
1	7	10

Training data counts

1. How many datapoints (n) are in our train data?

$n = 30$

2. How many parameters do we need to estimate?

- $\hat{P}(X_1|Y=0), \hat{P}(X_1|Y=1)$: 4 params
- $\hat{P}(X_2|Y=0), \hat{P}(X_2|Y=1)$: 4 params
- $\hat{P}(Y)$: 2 params

3. Compute MLE estimates for $\hat{P}(X_1|Y)$:

$Y \backslash X_1$	0	1
0	$3/13 \approx 0.23$	$10/13 \approx 0.77$
1	$4/17 \approx 0.24$	$13/17 \approx 0.76$

Training: Naïve Bayes for TV shows (**MLE**)

$$\hat{Y} = \underset{y=\{0,1\}}{\arg\max} \left(\prod_{i=1}^{m} \hat{P}(X_i|Y)\right)\hat{P}(Y)$$

Observe indicator vars. $\boldsymbol{X} = (X_1, X_2)$:

- X_1: “likes Star Wars”
- X_2: “likes Harry Potter”

Predict Y: “likes Pokémon”

Y \ X_1	0	1
0	3	10
1	4	13

Y \ X_2	0	1
0	5	8
1	7	10

Training data counts

MLE estimates of $\hat{P}(X_1|Y), \hat{P}(X_2|Y), \hat{P}(Y)$:

Y \ X_1	0	1
0	0.23	0.77
1	0.24	0.76

$\hat{P}(X_1|Y)$

Y \ X_2	0	1
0	$5/13 \approx 0.38$	$8/13 \approx 0.62$
1	$7/17 \approx 0.41$	$10/17 \approx 0.59$

$\hat{P}(X_2|Y)$

Y	
0	$13/30 \approx 0.43$
1	$17/30 \approx 0.57$

$\hat{P}(Y)$

👉 Training MLE estimates: just count.

$$\hat{P}(X_i = x|Y = y) = \frac{\#(X_i = x, Y = y)}{\#(Y = y)}$$

$$\hat{P}(Y = y) = \frac{\#(Y = y)}{n}$$

Training: Naïve Bayes for TV shows (**MLE**)

$$\hat{Y} = \arg\max_{y=\{0,1\}} \left(\prod_{i=1}^{m} \hat{P}(X_i|Y) \right) \hat{P}(Y)$$

Observe indicator vars. $\boldsymbol{X} = (X_1, X_2)$:

- X_1: “likes Star Wars”
- X_2: “likes Harry Potter”

Predict Y: “likes Pokémon”

Y \ X_1	0	1
0	0.23	0.77
1	0.24	0.76

$\hat{P}(X_1|Y)$

Y \ X_2	0	1
0	0.38	0.62
1	0.41	0.59

$\hat{P}(X_2|Y)$

Y	
0	0.43
1	0.57

$\hat{P}(Y)$

Now that we’ve trained and found parameters,
It’s time to classify new users!

Testing: Naïve Bayes for TV shows (MLE)

$$\hat{Y} = \arg\max_{y=\{0,1\}} \left(\prod_{i=1}^{m} \hat{P}(X_i|Y) \right) \hat{P}(Y)$$

Observe indicator vars. $\boldsymbol{X} = (X_1, X_2)$:

- X_1: "likes Star Wars"
- X_2: "likes Harry Potter"

Predict Y: "likes Pokémon"

Y \ X_1	0	1
0	0.23	0.77
1	0.24	0.76

Y \ X_2	0	1
0	0.38	0.62
1	0.41	0.59

Y	
0	0.43
1	0.57

Suppose a new person "likes Star Wars" ($X_1 = 1$) but "dislikes Harry Potter" ($X_2 = 0$).

Will they like Pokemon? Need to predict Y:

$$\hat{Y} = \arg\max_{y=\{0,1\}} \hat{P}(\boldsymbol{X}|Y)\hat{P}(Y) \quad = \arg\max_{y=\{0,1\}} \hat{P}(X_1|Y)\hat{P}(X_2|Y)\hat{P}(Y)$$

If $Y = 0$: $\hat{P}(X_1 = 1|Y = 0)\hat{P}(X_2 = 0|Y = 0)\hat{P}(Y = 0) = 0.77 \cdot 0.38 \cdot 0.43 = 0.126$

If $Y = 1$: $\hat{P}(X_1 = 1|Y = 1)\hat{P}(X_2 = 0|Y = 1)\hat{P}(Y = 1) = 0.76 \cdot 0.41 \cdot 0.57 = 0.178$

Since term is greatest when Y = 1, predict $\hat{Y} = 1$

Naïve Bayes Classifier

$$\hat{Y} = \underset{y=\{0,1\}}{\arg\max} \left(\prod_{i=1}^{m} \hat{P}(X_i|Y) \right) \hat{P}(Y)$$

We can use MLE or MAP to estimate our parameters.

Let’s try using MAP with Laplace smoothing.

Training: Naïve Bayes for TV shows (MAP)

$$\hat{Y} = \underset{y=\{0,1\}}{\arg\max}\left(\prod_{i=1}^{m}\hat{P}(X_i|Y)\right)\hat{P}(Y)$$

Observe indicator vars. $\boldsymbol{X} = (X_1, X_2)$:

- X_1: “likes Star Wars”
- X_2: “likes Harry Potter”

Predict Y: “likes Pokémon”

Y \ X_1	0	1
0	3	10
1	4	13

Y \ X_2	0	1
0	5	8
1	7	10

Training data counts

What are our MAP estimates using Laplace smoothing for $\hat{P}(X_i|Y)$ and $\hat{P}(Y)$?

$\hat{P}(X_i = x|Y = y)$:

A. $\dfrac{\#(X_i{=}x,Y{=}y)}{\#(Y{=}y)}$

B. $\dfrac{\#(X_i{=}x,Y{=}y)+1}{\#(Y{=}y)+2}$

C. $\dfrac{\#(X_i{=}x,Y{=}y)+1}{\#(Y{=}y)+4}$

$\hat{P}(Y = y)$:

A. $\dfrac{\#(Y{=}y)}{\#(Y{=}y)+2}$

B. $\dfrac{\#(Y{=}y)+1}{n}$

C. $\dfrac{\#(Y{=}y)+1}{n+2}$

🤔

Training: Naïve Bayes for TV shows (MAP)

$$\hat{Y} = \underset{y=\{0,1\}}{\arg\max}\left(\prod_{i=1}^{m} \hat{P}(X_i|Y)\right)\hat{P}(Y)$$

Observe indicator vars. $\boldsymbol{X} = (X_1, X_2)$:

- X_1: “likes Star Wars”
- X_2: “likes Harry Potter”

Predict Y: “likes Pokémon”

$Y \backslash X_1$	0	1
0	3	10
1	4	13

$Y \backslash X_2$	0	1
0	5	8
1	7	10

Training data counts

What are our MAP estimates using Laplace smoothing for $\hat{P}(X_i|Y)$ and $\hat{P}(Y)$?

$\hat{P}(X_i = x|Y = y)$:

A. $\frac{\#(X_i=x, Y=y)}{\#(Y=y)}$

[✓ B.] $\frac{\#(X_i=x, Y=y)+1}{\#(Y=y)+2}$

C. $\frac{\#(X_i=x, Y=y)+1}{\#(Y=y)+4}$

$\hat{P}(Y = y)$:

A. $\frac{\#(Y=y)}{\#(Y=y)+2}$

B. $\frac{\#(Y=y)+1}{n}$

[✓ C.] $\frac{\#(Y=y)+1}{n+2}$

Training: Naïve Bayes for TV shows (MAP)

$$\hat{Y} = \arg\max_{y=\{0,1\}} \left(\prod_{i=1}^{m} \hat{P}(X_i|Y)\right) \hat{P}(Y)$$

Observe indicator vars. $\boldsymbol{X} = (X_1, X_2)$:

- X_1: “likes Star Wars”
- X_2: “likes Harry Potter”

Predict Y: “likes Pokémon”

Y \ X_1	0	1
0	3	10
1	4	13

Y \ X_2	0	1
0	5	8
1	7	10

Training data

Y \ X_1	0	1
0	0.27	0.73
1	0.26	0.74

Y \ X_2	0	1
0	0.40	0.60
1	0.42	0.58

Y	
0	$14/32 \approx 0.44$
1	$18/32 \approx 0.56$

👉 Training MAP estimates: just count + imaginary trials.

$$\hat{P}(X_i = x|Y = y) = \frac{\#(X_i = x, Y = y) + 1}{\#(Y = y) + 2}$$

$$\hat{P}(Y = y) = \frac{\#(Y = y) + 1}{n + 2}$$

🤔

Naïve Bayes Classifier

$$\hat{Y} = \arg\max_{y=\{0,1\}} \left(\prod_{i=1}^{m} \hat{P}(X_i|Y) \right) \hat{P}(Y)$$

What is the intuition behind the Naïve Bayes assumption?

Naïve Bayes Model is a Bayesian Network

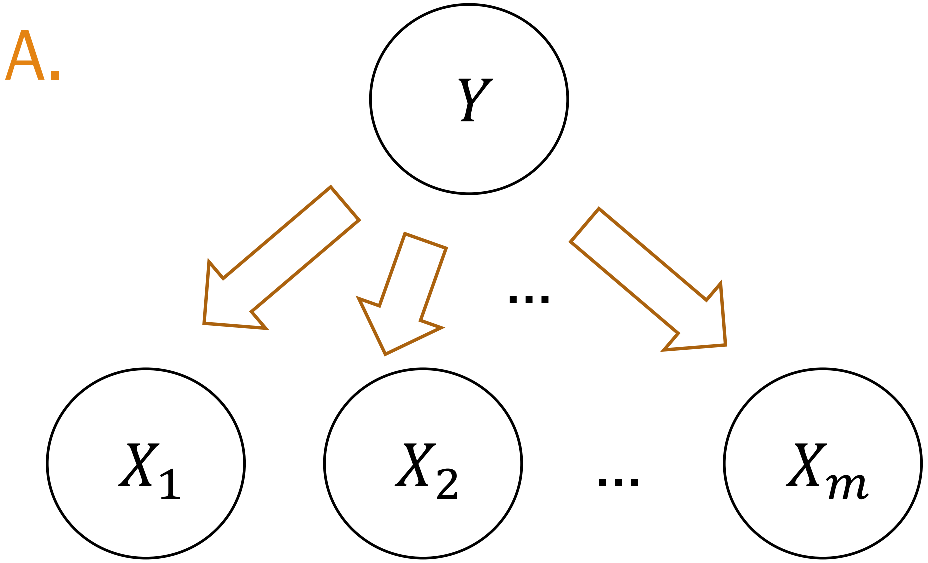
$$\hat{Y} = \arg\max_{y=\{0,1\}} \left(\prod_{i=1}^{m} \hat{P}(X_i|Y) \right) \hat{P}(Y)$$

Naïve Bayes Assumption

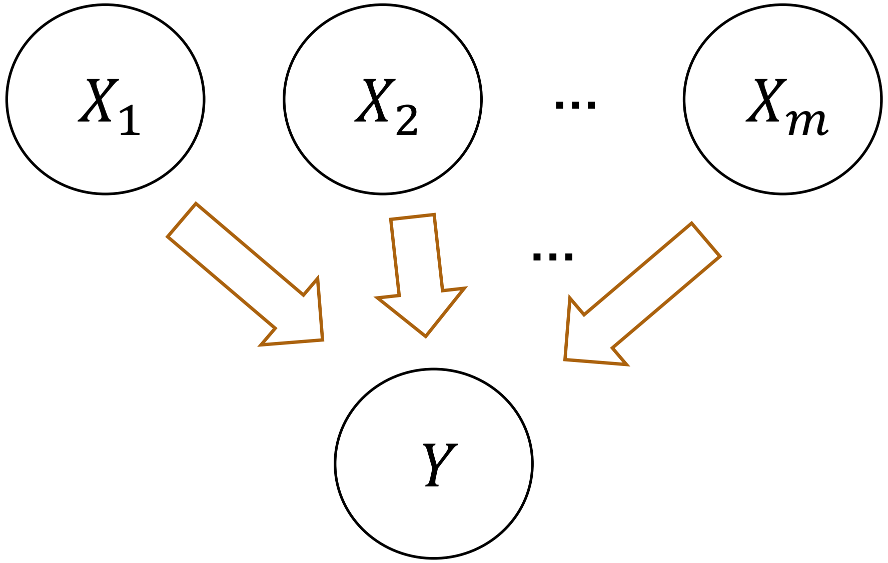
$$P(\boldsymbol{X}|Y) = \prod_{i=1}^{m} P(X_i|Y) \quad \Rightarrow \quad P(\boldsymbol{X}, Y) = P(Y) \prod_{i=1}^{m} P(X_i|Y)$$

Which Bayesian Network encodes this conditional independence?

A.



B.



🤔

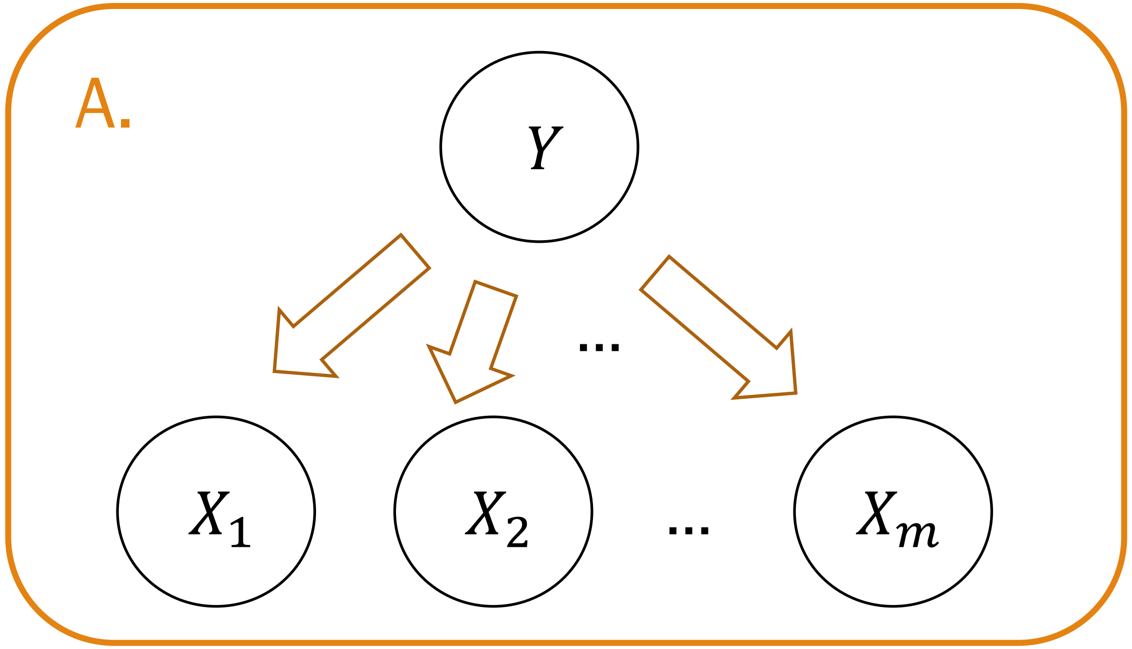
Naïve Bayes Model is a Bayesian Network

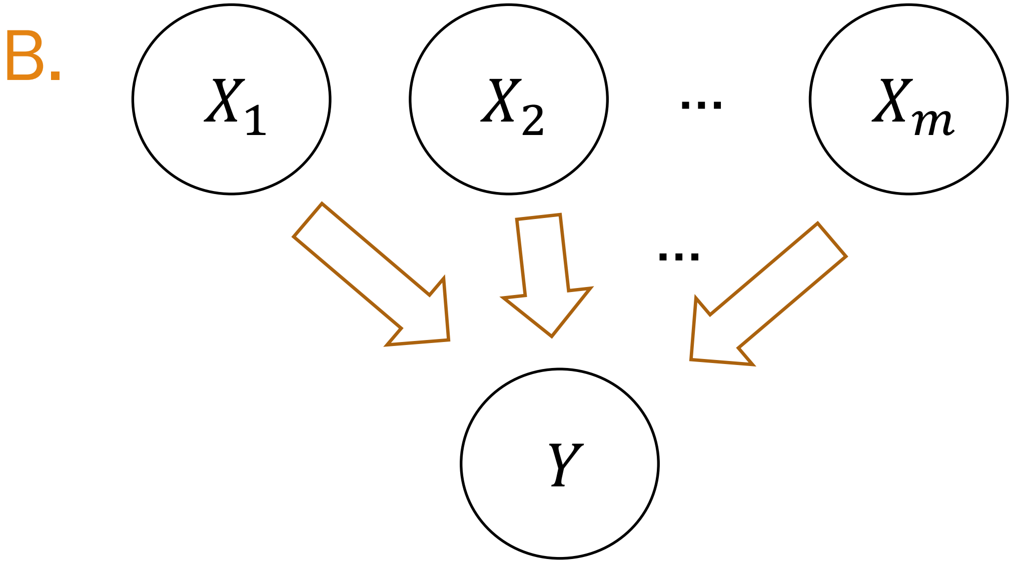
$$\hat{Y} = \underset{y=\{0,1\}}{\arg\max} \left(\prod_{i=1}^{m} \hat{P}(X_i|Y) \right) \hat{P}(Y)$$

Naïve Bayes Assumption

$$P(\mathbf{X}|Y) = \prod_{i=1}^{m} P(X_i|Y) \quad \Rightarrow \quad P(\mathbf{X}, Y) = P(Y) \prod_{i=1}^{m} P(X_i|Y)$$

Which Bayesian Network encodes this conditional independence?





X_i are conditionally independent given parent Y

Break for jokes/ announcements

Announcements

Problem Set 6

Due:	Wednesday 12/4 (after break)
Covers:	Up to end of this week

Late day reminder: No late days permitted past last day of the quarter, 12/6 (Friday)

CS109 Contest

Due:	Monday 12/2 11:59pm
Note:	All serious submissions will get some extra credit

Today's plan

Naïve Bayes

Logistic Regression

- Chapter 0: Background
- Chapter 1: Big Picture
- Chapter 2: Details
- Chapter 3: Philosophy

Background: Weighted sum

If $\boldsymbol{X} = (X_1, X_2, \dots, X_m)$:

$$z = \theta^T \boldsymbol{X} = \sum_{j=1}^{m} \theta_j X_j$$

Weighted sum
(aka dot product)

$$= \theta_1 X_1 + \theta_2 X_2 + \cdots + \theta_m X_m$$

Weighted sum with an intercept term:

$$z = \theta_0 + \sum_{j=1}^{m} \theta_j X_j$$

$$= \theta_0 X_0 + \theta_1 X_1 + \theta_2 X_2 + \cdots + \theta_m X_m$$

Define $X_0 = 1$

$$= \theta^T \boldsymbol{X}$$

New $\boldsymbol{X} = (1, X_1, X_2, \dots, X_m)$

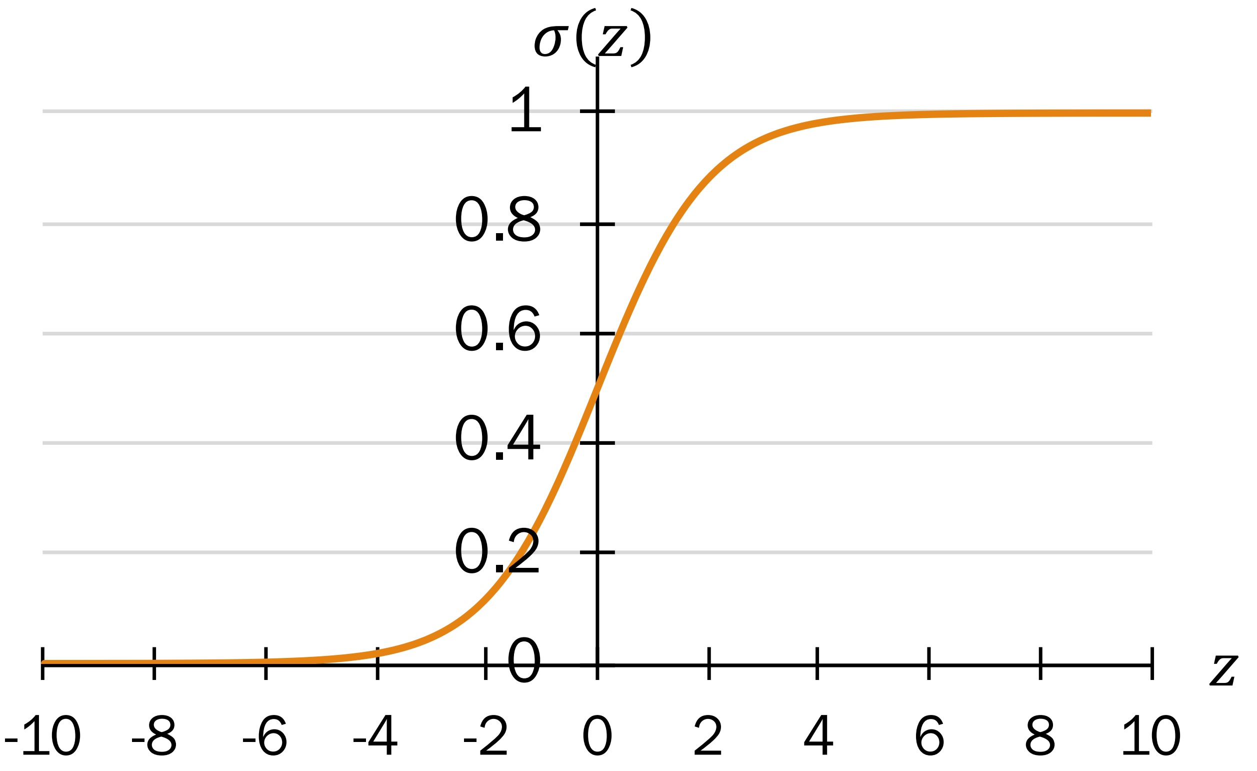
Background: Sigmoid function $\sigma(z)$

- The sigmoid function:

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

- Sigmoid squashes z to a number between 0 and 1.

- Recall definition of probability: A number between 0 and 1



👉 $\sigma(z)$ can represent a probability.

Background: Chain Rule

$$\frac{\partial f(x)}{\partial x} = \frac{\partial f(z)}{\partial z}\frac{\partial z}{\partial x}$$

Calculus
Chain Rule

$$f(x) = f\big(z(x)\big)$$

aka decomposition
of composed functions

Today's plan

Naïve Bayes

Logistic Regression

- Chapter 0: Background
- Chapter 1: Big Picture
- Chapter 2: Details
- Chapter 3: Philosophy

From Naïve Bayes to Logistic Regression

Classification goal: Model $P(Y \mid \boldsymbol{X})$

$$\hat{Y} = \underset{y=\{0,1\}}{\arg\max}\, P(Y \mid \boldsymbol{X})$$

Predict the Y that is most likely given our observation $\boldsymbol{X}$

Naïve Bayes Classifier:

- Estimate $P(\boldsymbol{X} \mid Y)$ and $P(Y)$ because $\underset{y=\{0,1\}}{\arg\max}\, P(Y \mid \boldsymbol{X}) = \underset{y=\{0,1\}}{\arg\max}\, P(\boldsymbol{X}|Y)P(Y)$
- Actually modeling $P(\boldsymbol{X}, Y)$
- Assume $P(\boldsymbol{X}|Y) = P(X_1, X_2, \ldots, X_n|Y) = \prod_{i=1}^{m} P(X_i|Y)$

Can we model $P(Y \mid \boldsymbol{X})$ directly?

- Welcome our friend: Logistic Regression!

Logistic Regression

$$\hat{Y} = \arg\max_{y=\{0,1\}} P(Y \mid \boldsymbol{X})$$

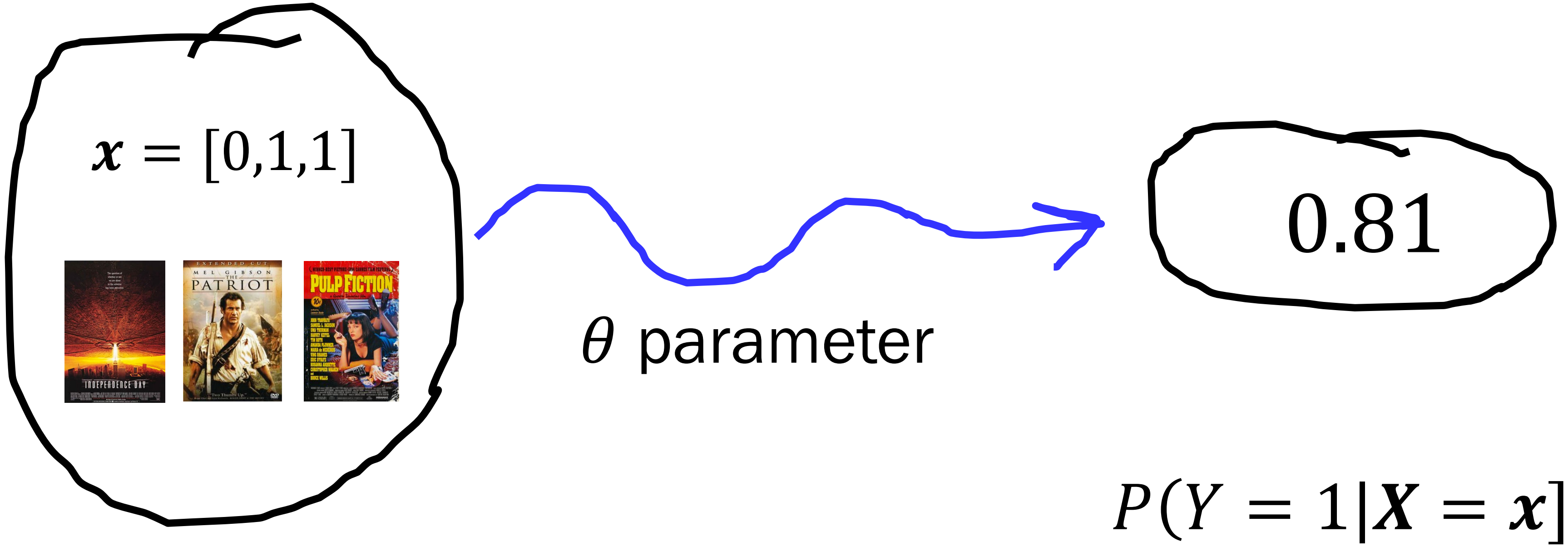
Predict the Y that is most likely given our observation $\boldsymbol{X}$

Logistic Regression Model

$$P(Y = 1 | \boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

models $P(Y \mid \boldsymbol{X})$ directly

Logistic Regression



$\boldsymbol{x} = [0,1,1]$

θ parameter

0.81

$\boldsymbol{X}$

input features

$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}])$

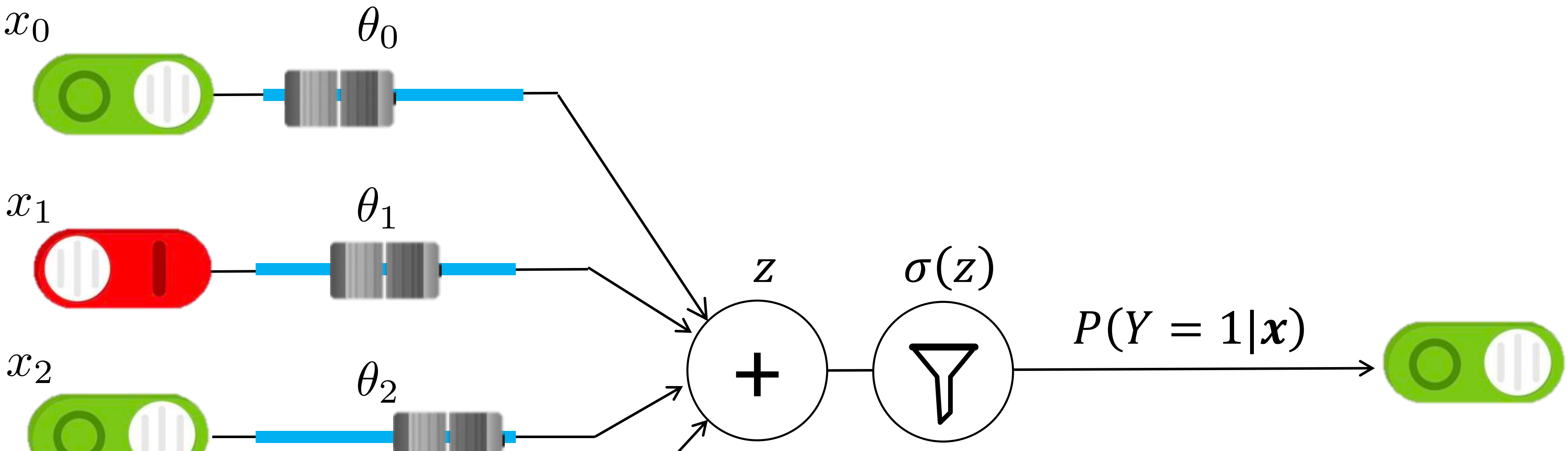
conditional likelihood

$$P(Y = 1|\boldsymbol{X} = x) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

Logistic Regression Cartoon

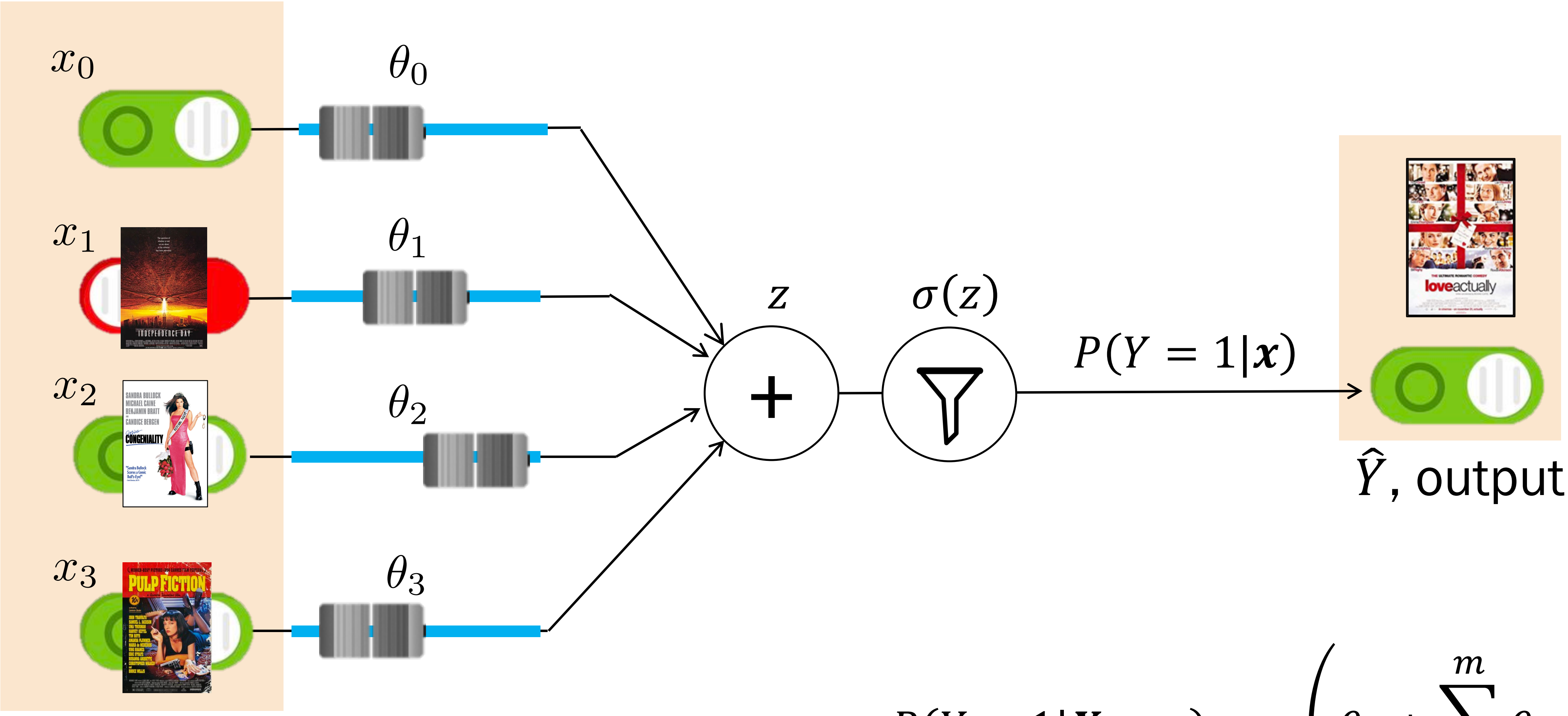
θ parameter

Logistic Regression cartoon



$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

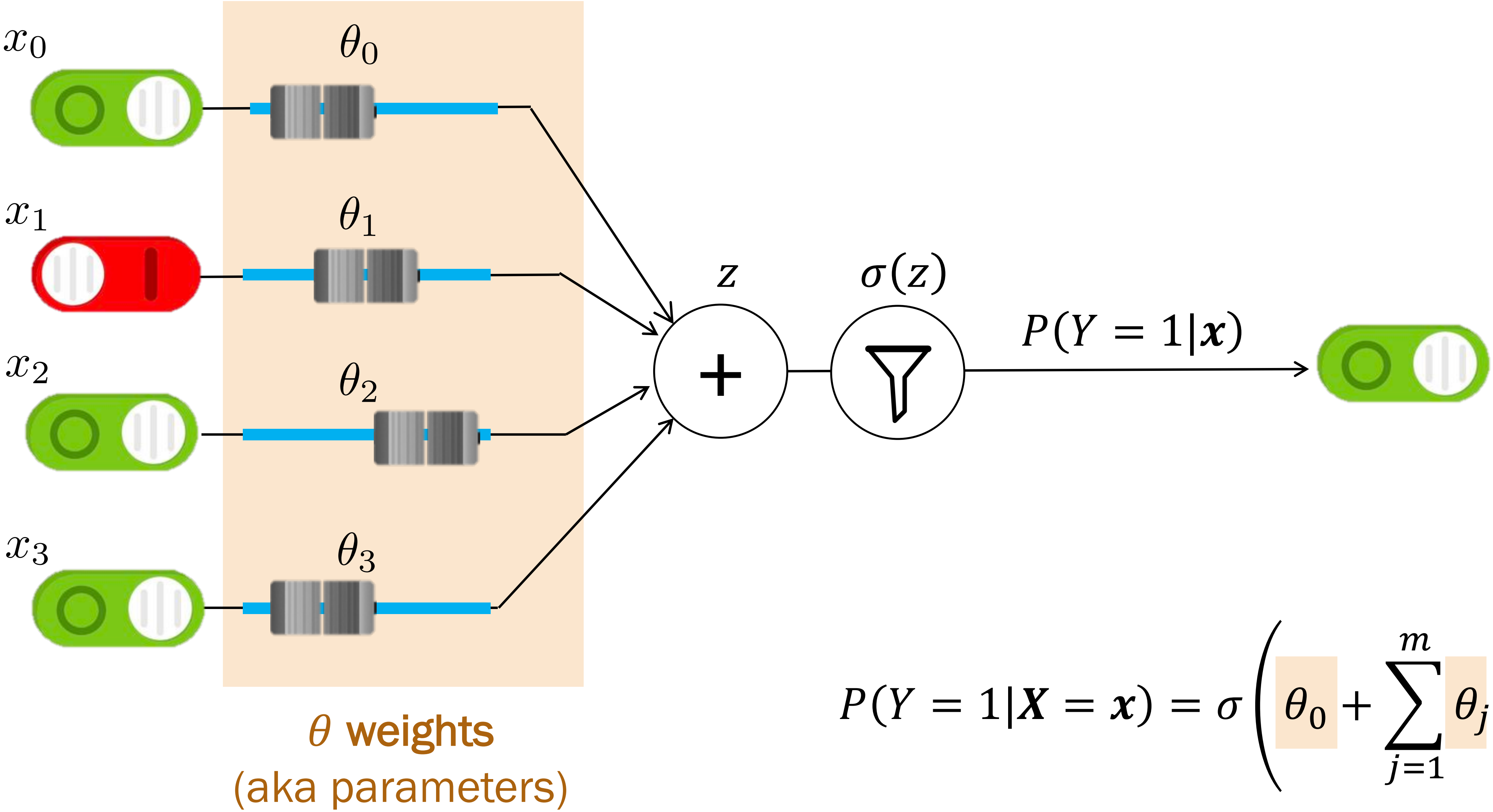
Logistic Regression input/output



$\boldsymbol{X}$, input features
[0,1,1]

$$P(Y=1|\boldsymbol{X}=\boldsymbol{x})=\sigma\left(\theta_0+\sum_{j=1}^{m}\theta_j x_j\right)$$

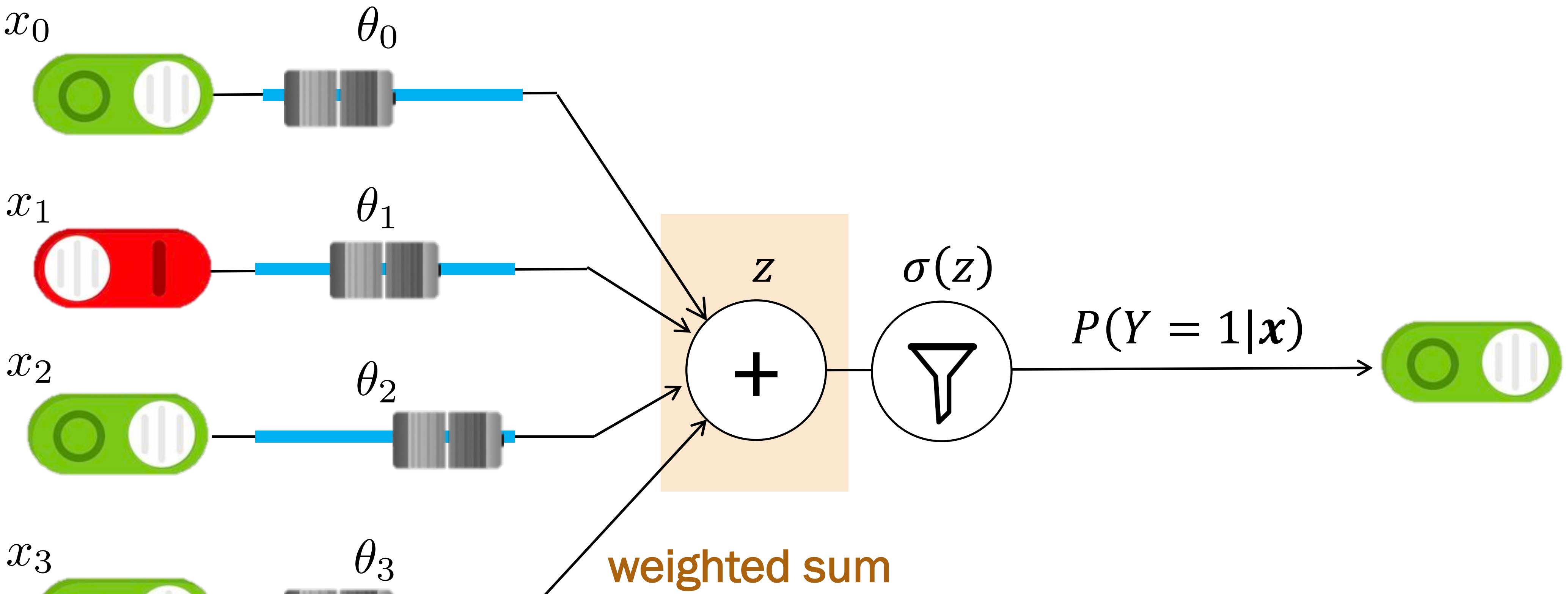
Components of Logistic Regression



θ weights
(aka parameters)

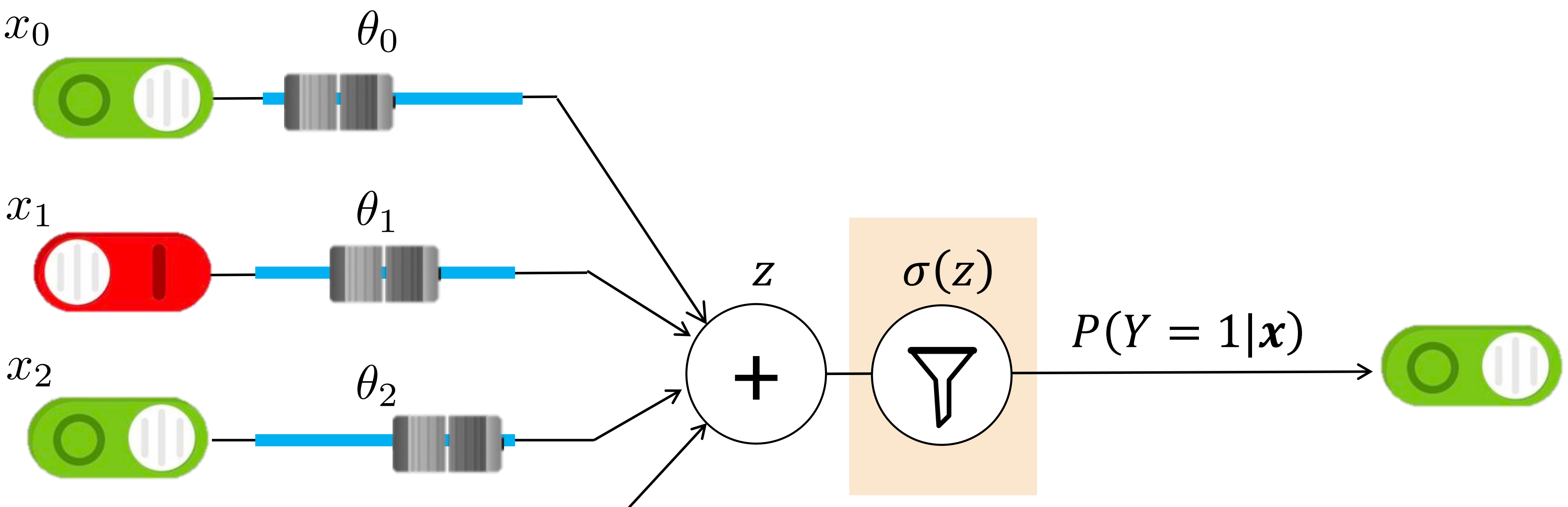
$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

Components of Logistic Regression



$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

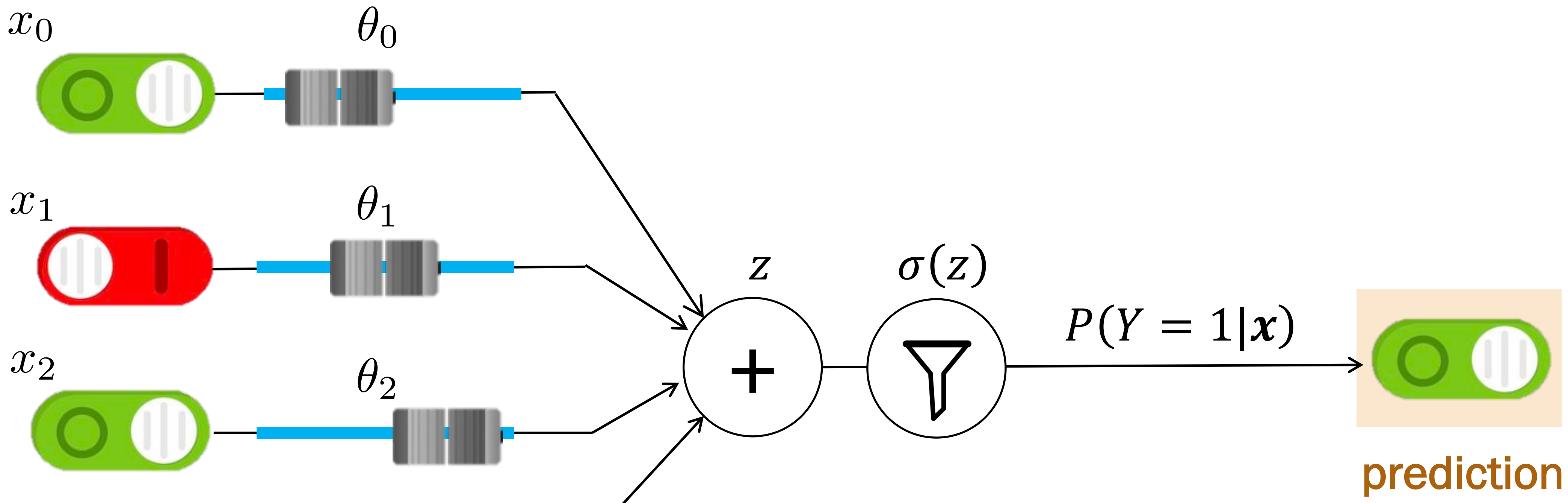
Components of Logistic Regression



squashing function
b/t 0 and 1

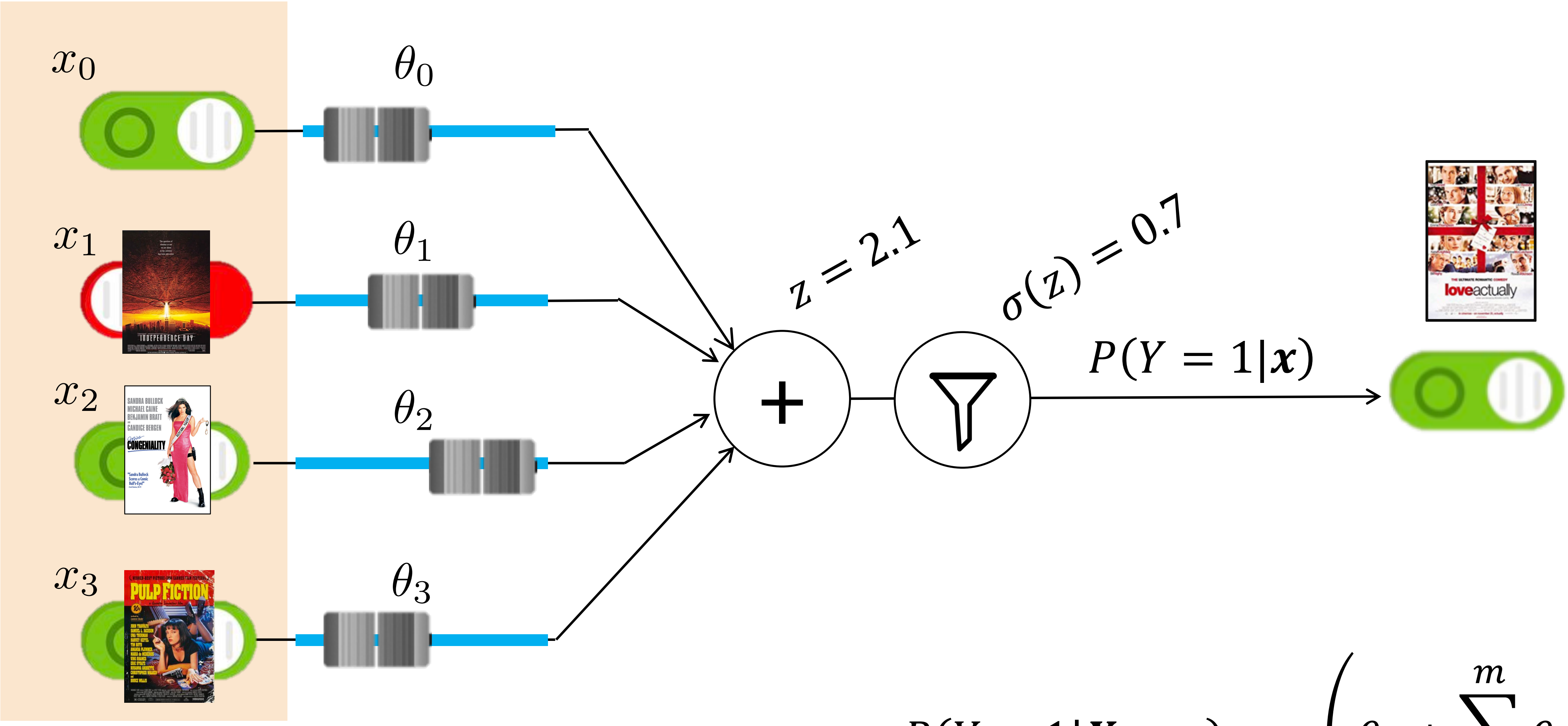
$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

Components of Logistic Regression



$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

Different predictions for different inputs

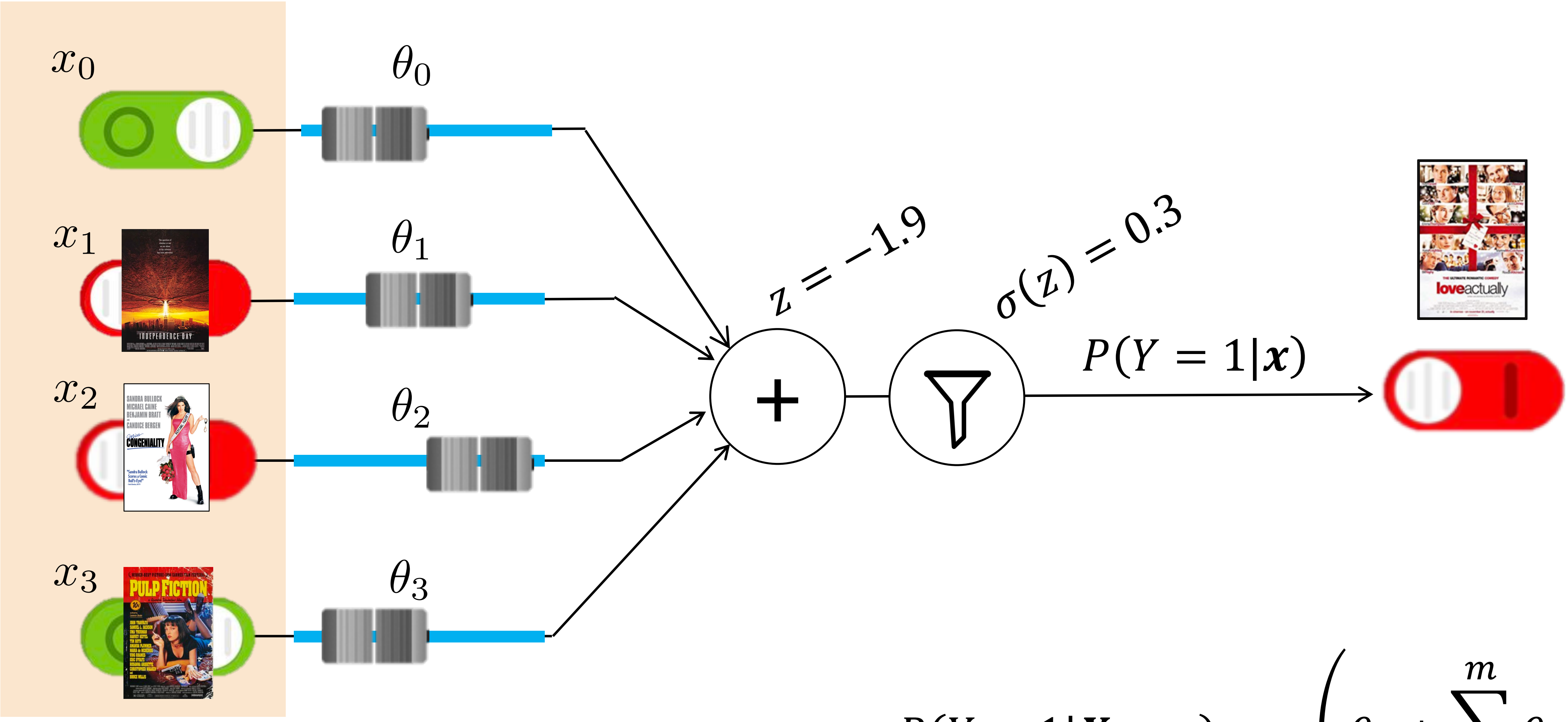


$\boldsymbol{X}$, input features
[0,1,1]

$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

Slides courtesy of Chris Piech

Different predictions for different inputs

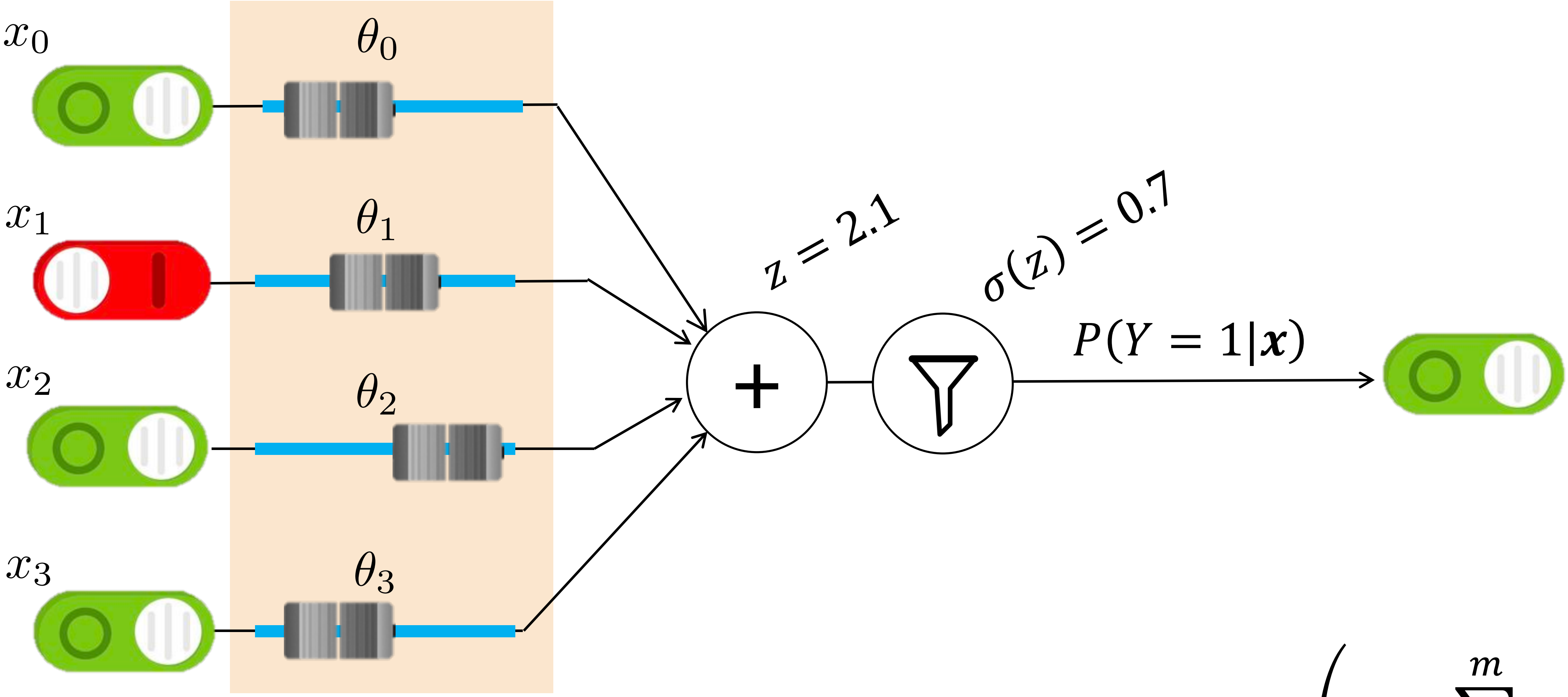


$\boldsymbol{X}$, input features

[0,0,1]

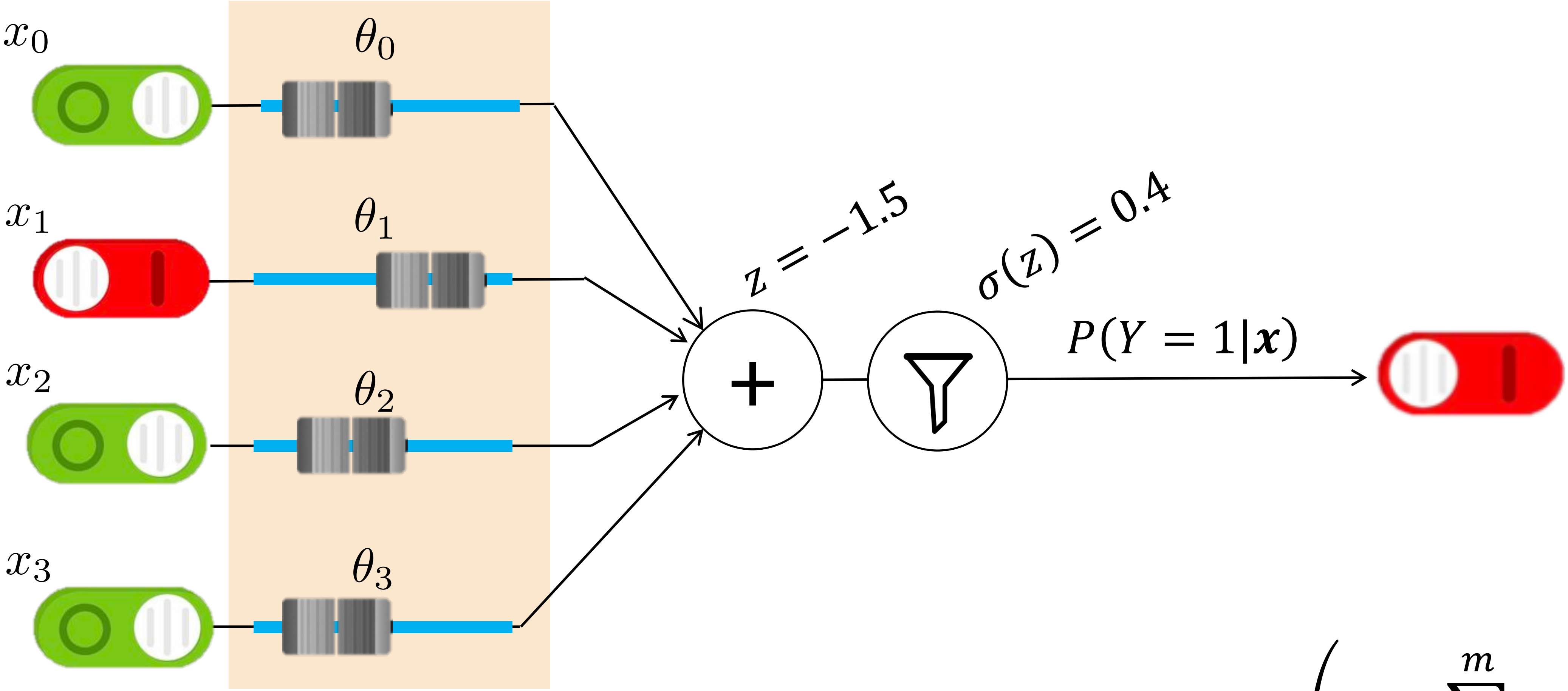
$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

Parameters affect prediction



$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

Parameters affect prediction



$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

Logistic Regression Model

$$\hat{Y} = \arg\max_{y=\{0,1\}} P(Y \mid \boldsymbol{X})$$

where

$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

Predict the Y that is most likely given our observation $\boldsymbol{X}$

models $P(Y \mid \boldsymbol{X})$ directly

- $\sigma(z) = \frac{1}{1+e^{-z}}$, the sigmoid function
- For simplicity, define $x_0 = 1$: $P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma(\theta^T \boldsymbol{x})$
- Since $P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) + P(Y = 0|\boldsymbol{X} = \boldsymbol{x}) = 1$:

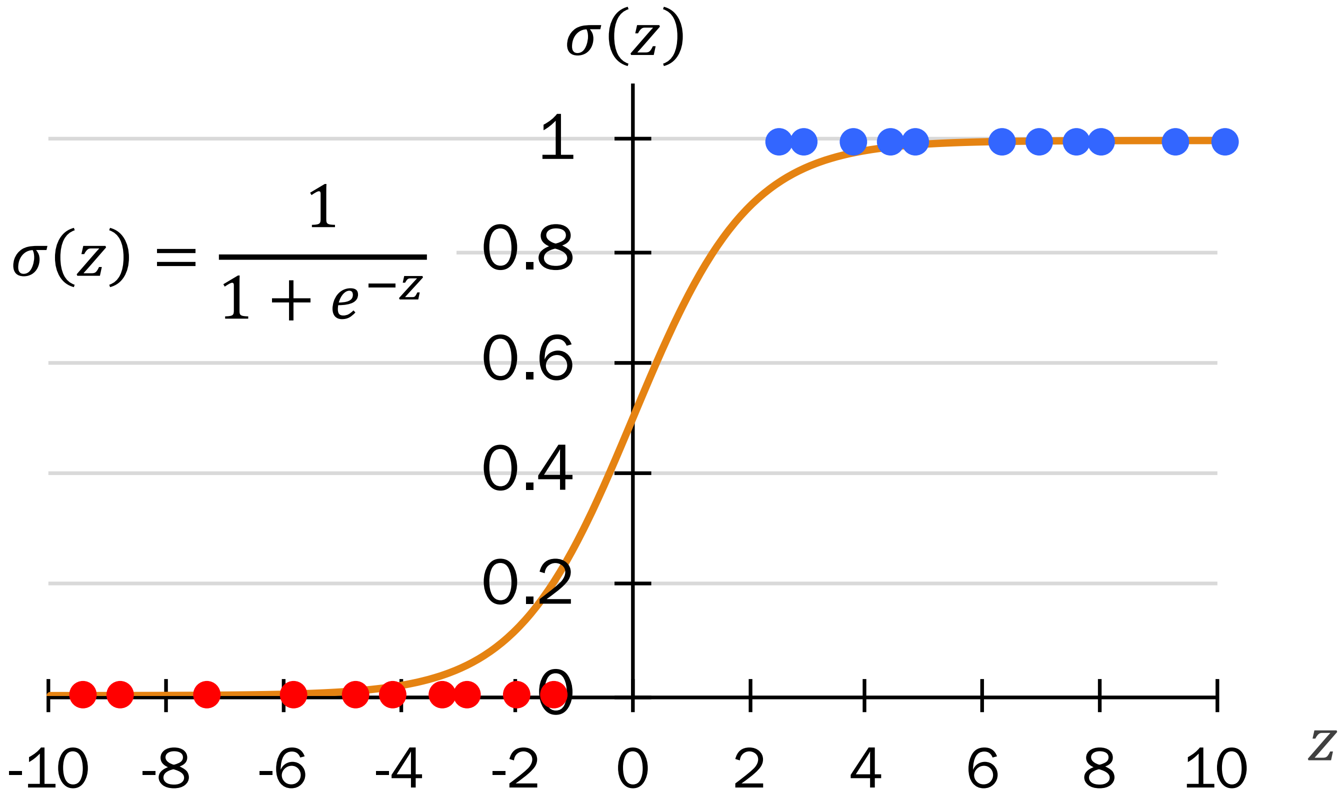
$$P(Y = 0|\boldsymbol{X} = \boldsymbol{x}) = 1 - \sigma(\theta^T \boldsymbol{x})$$

Classifying using the sigmoid function

Logistic Regression Model

$$\hat{Y} = \arg\max_{y=\{0,1\}} P(Y \mid \boldsymbol{X}) \quad \text{where} \quad P(Y = 1 | \boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

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



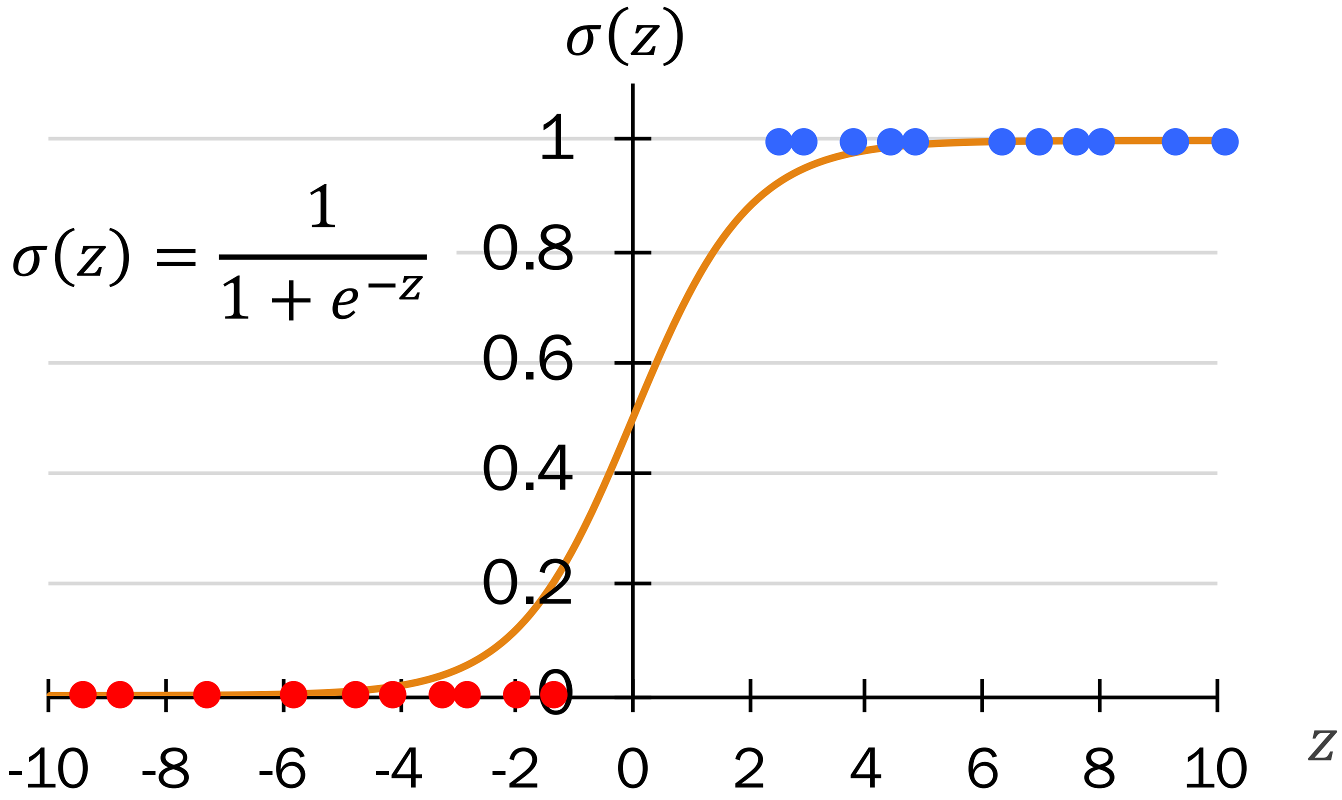
👉 Logistic Regression uses the sigmoid function to try and distinguish $y = 1$ (blue) points from $y = 0$ (red) points.

Classifying using the sigmoid function

Logistic Regression Model

$$\hat{Y} = \underset{y=\{0,1\}}{\arg\max}\, P(Y \mid \boldsymbol{X}) \quad \text{where} \quad P(Y=1|\boldsymbol{X}=\boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

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



When do we predict $\hat{Y} = 1$?

A. If $\sigma(\theta^T \boldsymbol{x}) > 1 - \sigma(\theta^T \boldsymbol{x})$

B. If $\sigma(\theta^T \boldsymbol{x}) > 0.5$

C. If $\theta^T \boldsymbol{x} > 0$

D. All are valid, but C is easiest

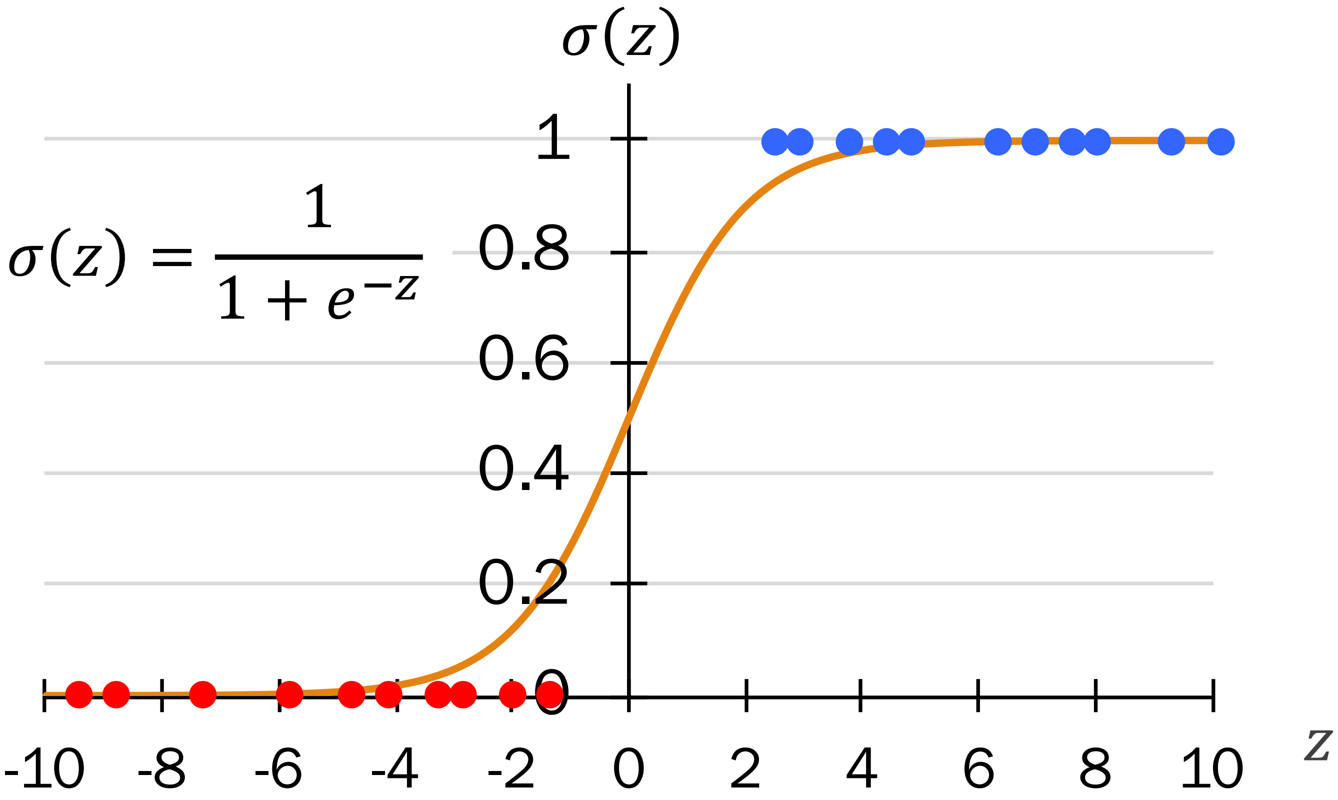
E. None/Other

Classifying using the sigmoid function

Logistic Regression Model

$$\hat{Y} = \underset{y=\{0,1\}}{\arg\max}\, P(Y \mid \boldsymbol{X}) \quad \text{where} \quad P(Y=1|\boldsymbol{X}=\boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

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



When do we predict $\hat{Y} = 1$?

A. If $\sigma(\theta^T \boldsymbol{x}) > 1 - \sigma(\theta^T \boldsymbol{x})$

B. If $\sigma(\theta^T \boldsymbol{x}) > 0.5$

C. If $\theta^T \boldsymbol{x} > 0$

[✓ D.] All are valid, but C is easiest

E. None/Other

Naming algorithms

Regression Algorithms

Linear Regression 🍅

Classification Algorithms

Naïve Bayes 🍅

Logistic Regression ✳

Awesome classifier, terrible name

If Lisa could rename it, she would call it: Sigmoidal Classification

Training: Learning the parameters

Logistic regression gets its **intelligence** from its parameters $\theta = (\theta_0, \theta_1, \dots, \theta_m)$.

- Logistic Regression Model:

$$P(Y = 1 | \boldsymbol{X} = \boldsymbol{x}) = \sigma(\theta^T \boldsymbol{x})$$

- Want to predict training data as correctly as possible:

$$\underset{y=\{0,1\}}{\arg\max}\, P\left(Y | \boldsymbol{X} = \boldsymbol{x}^{(i)}\right) = y^{(i)}$$

as often as possible

- Therefore, choose θ that maximizes the conditional likelihood of observing i.i.d. training data:

$$L(\theta) = \prod_{i=1}^{n} P\left(Y = y^{(i)} | \boldsymbol{X} = \boldsymbol{x}^{(i)}, \theta\right)$$

👉 During training, find the θ that maximizes log-conditional likelihood of the training data. Use MLE!

Training: Learning the parameters via MLE

0. Add $x_0^{(i)} = 1$ to each $\boldsymbol{x}^{(i)}$

1. Logistic Regression model: $P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma(\theta^T \boldsymbol{x})$

2. Compute log-likelihood of training data:

$$LL(\theta) = \sum_{i=1}^{n} y^{(i)} \log \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right)\right)$$

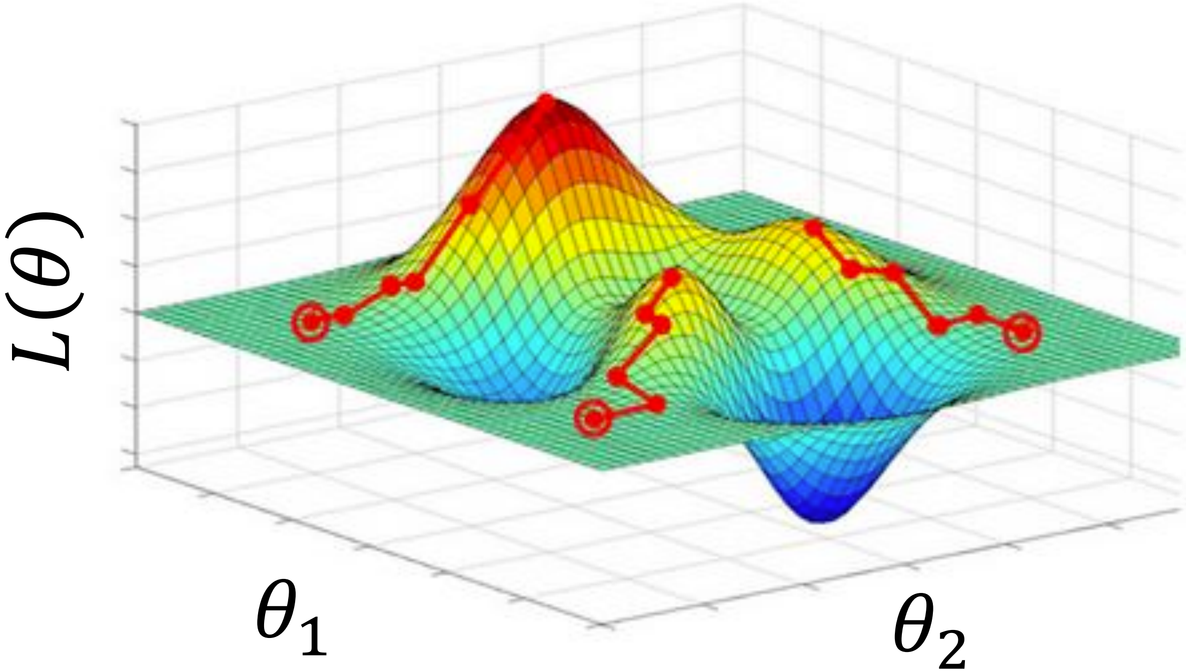
3. Compute derivative of log-likelihood with respect to each $\theta_j, j = 0, 1, \dots, m$:

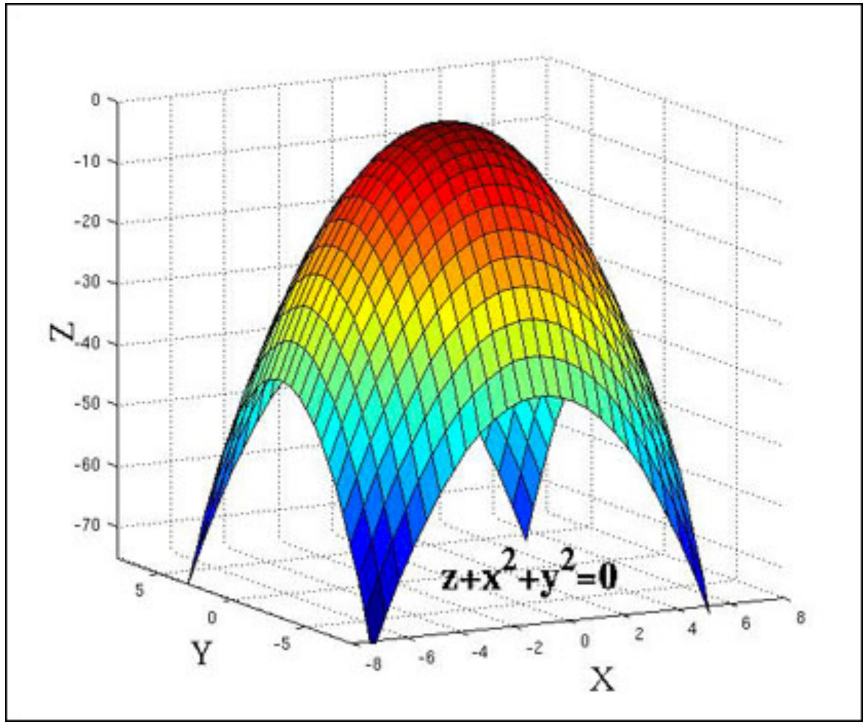
$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right)\right] x_j^{(i)}$$

Gradient Ascent

Review

Walk uphill and you will find a local maxima
(if your step is small enough).





Logistic regression $LL(\theta)$
is convex

Training: Gradient ascent step

4. Optimize.

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right) \right] x_j^{(i)}$$

Repeat many times:

For all thetas:

$$\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \frac{\partial LL\left(\theta^{\text{old}}\right)}{\partial \theta_j^{\text{old}}}$$

$$= \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left({\theta^{\text{old}}}^T \boldsymbol{x}^{(i)} \right) \right] x_j^{(i)}$$

What does this look like in code?

Training: Gradient Ascent

Gradient Ascent Step $\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left({\theta^{\text{old}}}^T \boldsymbol{x}^{(\boldsymbol{i})} \right) \right] x_j^{(i)}$

```
initialize $\theta_j$ = 0 for 0 ≤ j ≤ m
repeat many times:
    gradient[j] = 0 for 0 ≤ j ≤ m

    // compute all gradient[j]'s
    // based on n training examples

    $\theta_j$ += η * gradient[j] for all 0 ≤ j ≤ m
```

Training: Gradient Ascent

Gradient Ascent Step $\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left({\theta^{\text{old}}}^T \boldsymbol{x}^{(\boldsymbol{i})}\right)\right] x_j^{(i)}$

```
initialize $\theta_j$ = 0 for 0 ≤ j ≤ m
repeat many times:
  gradient[j] = 0 for 0 ≤ j ≤ m
  for each training example (x, y):
    for each 0 ≤ j ≤ m:
      // update gradient[j] for
      // current (x,y) example
  $\theta_j$ += η * gradient[j] for all 0 ≤ j ≤ m
```

Training: Gradient Ascent

Gradient Ascent Step $\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left({\theta^{\text{old}}}^T \boldsymbol{x}^{(i)}\right)\right] x_j^{(i)}$

```
initialize $\theta_j$ = 0 for 0 ≤ j ≤ m
repeat many times:
    gradient[j] = 0 for 0 ≤ j ≤ m
    for each training example (x, y):
        for each 0 ≤ j ≤ m:
            gradient[j] += $\left[y - \frac{1}{1 + e^{-\theta^T \boldsymbol{x}}}\right] x_j$
    $\theta_j$ += η * gradient[j] for all 0 ≤ j ≤ m
```

What are important implementation details? 🤔

Training: Gradient Ascent

Gradient Ascent Step $\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left({\theta^{\text{old}}}^T \boldsymbol{x}^{(i)} \right) \right] x_j^{(i)}$

```
initialize $\theta_j$ = 0 for 0 ≤ j ≤ m
repeat many times:
    gradient[j] = 0 for 0 ≤ j ≤ m
    for each training example (x, y):
        for each 0 ≤ j ≤ m:
            gradient[j] += $\left[ y - \frac{1}{1 + e^{-\theta^T \boldsymbol{x}}} \right] \boxed{x_j}$
    $\theta_j$ += η * gradient[j] for all 0 ≤ j ≤ m
```

- x_j is j-th feature of input var $x = (x_1, \ldots, x_m)$

Training: Gradient Ascent

Gradient Ascent Step $\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left({\theta^{\text{old}}}^T \boldsymbol{x}^{(i)} \right) \right] x_j^{(i)}$

```
initialize $\theta_j$ = 0 for 0 ≤ j ≤ m
repeat many times:
    gradient[j] = 0 for 0 ≤ j ≤ m
    for each training example (x, y):
        for each 0 ≤ j ≤ m:
            gradient[j] += $\left[ y - \frac{1}{1 + \boxed{e^{-\theta^T \boldsymbol{x}}}} \right] x_j$
    $\theta_j$ += η * gradient[j] for all 0 ≤ j ≤ m
```

- x_j is j-th feature of input var $x = (x_1, \dots, x_m)$
- Insert $x_0 = 1$ before training

Training: Gradient Ascent

Gradient Ascent Step $\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left({\theta^{\text{old}}}^{T} \boldsymbol{x}^{(\boldsymbol{i})}\right)\right] x_j^{(i)}$

```
initialize $\theta_j$ = 0 for 0 ≤ j ≤ m
repeat many times:
    gradient[j] = 0 for 0 ≤ j ≤ m
    for each training example (x, y):
        for each 0 ≤ j ≤ m:
            gradient[j] += $\left[y - \frac{1}{1 + e^{-\theta^T x}}\right] x_j$
    $\theta_j$ += η * gradient[j] for all 0 ≤ j ≤ m
```

- x_j is j-th feature of input var $x = (x_1, \dots, x_m)$
- Insert $x_0 = 1$ before training
- Finish computing gradient before updating any part of θ

Training: Gradient Ascent

Gradient Ascent Step $\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left({\theta^{\text{old}}}^T \boldsymbol{x}^{(i)} \right) \right] x_j^{(i)}$

```
initialize $\theta_j$ = 0 for 0 ≤ j ≤ m
repeat many times:
    gradient[j] = 0 for 0 ≤ j ≤ m
    for each training example (x, y):
        for each 0 ≤ j ≤ m:
            gradient[j] += $\left[ y - \frac{1}{1 + e^{-\theta^T \boldsymbol{x}}} \right] x_j$
    $\theta_j$ += $\boxed{\eta}$ * gradient[j] for all 0 ≤ j ≤ m
```

- x_j is j-th feature of input var $x = (x_1, \ldots, x_m)$
- Insert $x_0 = 1$ before training
- Finish computing gradient before updating any part of θ
- Learning rate η is a constant you set before training

Training: Gradient Ascent

Gradient Ascent Step $\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left(\theta^{\text{old}^T} \boldsymbol{x}^{(i)} \right) \right] x_j^{(i)}$

```
initialize $\theta_j$ = 0 for 0 ≤ j ≤ m
repeat many times:
    gradient[j] = 0 for 0 ≤ j ≤ m
    for each training example (x, y):
        for each 0 ≤ j ≤ m:
            gradient[j] += $\left[ y - \frac{1}{1 + e^{-\theta^T \boldsymbol{x}}} \right] x_j$
    $\theta_j$ += η * gradient[j] for all 0 ≤ j ≤ m
```

- x_j is j-th feature of input var $x = (x_1, \dots, x_m)$
- Insert $x_0 = 1$ before training
- Finish computing gradient before updating any part of θ
- Learning rate η is a constant you set before training

Testing: Classification with Logistic Regression

Training

Learn parameters $\theta = (\theta_0, \theta_1, \dots, \theta_m)$

via gradient ascent:

$$\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left({\theta^{\text{old}}}^T \boldsymbol{x}^{(\boldsymbol{i})} \right) \right] x_j^{(i)}$$

Testing

- Compute $\hat{y} = P(Y = 1 | \boldsymbol{X} = \boldsymbol{x}) = \sigma(\theta^T \boldsymbol{x}) = \dfrac{1}{1 + e^{-\theta^T \boldsymbol{x}}}$
- Classify instance as:

$$\begin{cases} 1 & \hat{y} > 0.5\text{, equivalently } \theta^T \boldsymbol{x} > 0 \\ 0 & \text{otherwise} \end{cases}$$

⚠ Parameters θ_j are **not** updated during testing phase

Today's plan

Naïve Bayes

Logistic Regression

- Chapter 0: Background
- Chapter 1: Big Picture
- Chapter 2: Details
- Chapter 3: Philosophy

Introducing notation $\hat{y}$

Logistic Regression model:

$$\hat{y} = P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma(\theta^T \boldsymbol{x})$$

$$P(Y = y|\boldsymbol{X} = \boldsymbol{x}) = \begin{cases} \hat{y} & \text{if } y = 1 \\ 1 - \hat{y} & \text{if } y = 0 \end{cases}$$

Prediction:

$$\hat{Y} = \underset{y=\{0,1\}}{\arg\max}\, P(Y|\boldsymbol{X} = \boldsymbol{x}) \quad = \begin{cases} 1 & \text{if } \hat{y} > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Training: Learning the parameters via MLE

0. Add $x_0^{(i)} = 1$ to each $\boldsymbol{x}^{(i)}$

1. Logistic Regression model:

$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \hat{y}$$
$$\hat{y} = \sigma(\theta^T \boldsymbol{x})$$

2. Compute log-likelihood of training data:

$$LL(\theta) = \sum_{i=1}^{n} y^{(i)} \log \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right)\right)$$

3. Compute derivative of log-likelihood with respect to each $\theta_j, j = 0, 1, \dots, m$:

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right)\right] x_j^{(i)}$$

Training: Learning the parameters via MLE

0. Add $x_0^{(i)} = 1$ to each $\boldsymbol{x}^{(i)}$
1. Logistic Regression model:

$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \hat{y}$$
$$\hat{y} = \sigma(\theta^T \boldsymbol{x})$$

2. Compute log-likelihood of training data:

$$LL(\theta) = \sum_{i=1}^{n} y^{(i)} \log \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right)\right)$$

🤔 How did we get this likelihood function?

3. Compute derivative of log-likelihood with respect to each $\theta_j, j = 0, 1, \dots, m$:

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right)\right] x_j^{(i)}$$

Log-likelihood of data

Logistic Regression model:

$$P(Y = y | \boldsymbol{X} = \boldsymbol{x}) = \begin{cases} \hat{y} & \text{if } y = 1 \\ 1 - \hat{y} & \text{if } y = 0 \end{cases}$$

where $\hat{y} = \sigma(\theta^T \boldsymbol{x})$

$$= (\hat{y})^y (1 - \hat{y})^{1-y}$$

(see Bernoulli MLE PMF)

Likelihood of training data:

$$L(\theta) = \prod_{i=1}^{n} P\left(Y = y^{(i)} | \boldsymbol{X} = \boldsymbol{x}^{(i)}, \theta\right)$$

Notes:

- Actually **conditional likelihood**
- Still correctly gets correct θ_{MLE} since $\boldsymbol{X}, \theta$ independent
- See lecture notes

Log-likelihood of data

Logistic Regression model:

$$P(Y = y|\boldsymbol{X} = \boldsymbol{x}) = \begin{cases} \hat{y} & \text{if } y = 1 \\ 1 - \hat{y} & \text{if } y = 0 \end{cases}$$

where $\hat{y} = \sigma(\theta^T \boldsymbol{x})$

$$= (\hat{y})^y (1 - \hat{y})^{1-y}$$

(see Bernoulli MLE PMF)

Likelihood of training data:

$$L(\theta) = \prod_{i=1}^{n} P\left(Y = y^{(i)} | \boldsymbol{X} = \boldsymbol{x}^{(i)}, \theta\right) = \prod_{i=1}^{n} \left(\hat{y}^{(i)}\right)^{y^{(i)}} \left(1 - \hat{y}^{(i)}\right)^{1-y^{(i)}}$$

Log-likelihood:

$$LL(\theta) = \sum_{i=1}^{n} y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)$$

$$= \sum_{i=1}^{n} y^{(i)} \log \sigma\left(\theta^T \boldsymbol{x}^{(\boldsymbol{i})}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right)\right)$$

Training: Learning the parameters via MLE

0. Add $x_0^{(i)} = 1$ to each $\boldsymbol{x}^{(i)}$
1. Logistic Regression model: $P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma(\theta^T \boldsymbol{x})$
2. Compute log-likelihood of training data: $$LL(\theta) = \sum_{i=1}^{n} y^{(i)} \log \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right)\right)$$
3. Compute derivative of log-likelihood with respect to each $\theta_j, j = 0, 1, \dots, m$: $$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right)\right] x_j^{(i)}$$

🤔 How did we get this gradient?

Aside: Sigmoid has a beautiful derivative

Sigmoid function:

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

Derivative:

$$\frac{d}{dz}\sigma(z) = \sigma(z)[1 - \sigma(z)]$$

What is $\frac{\partial}{\partial \theta_j}\sigma(\theta^T \boldsymbol{x})$?

A. $\sigma(x_j)[1 - \sigma(x_j)]x_j$

B. $\sigma(\theta^T \boldsymbol{x})[1 - \sigma(\theta^T \boldsymbol{x})]\boldsymbol{x}$

C. $\sigma(\theta^T \boldsymbol{x})[1 - \sigma(\theta^T \boldsymbol{x})]x_j$

D. $\sigma(\theta^T \boldsymbol{x})x_j[1 - \sigma(\theta^T \boldsymbol{x})x_j]$

E. None/other

Aside: Sigmoid has a beautiful derivative

Sigmoid function:

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

Derivative:

$$\frac{d}{dz}\sigma(z) = \sigma(z)[1 - \sigma(z)]$$

What is $\frac{\partial}{\partial \theta_j}\sigma(\theta^T \boldsymbol{x})$?

A. $\sigma(x_j)[1 - \sigma(x_j)]x_j$

B. $\sigma(\theta^T \boldsymbol{x})[1 - \sigma(\theta^T \boldsymbol{x})]\boldsymbol{x}$

[✓ C.] $\sigma(\theta^T \boldsymbol{x})[1 - \sigma(\theta^T \boldsymbol{x})]x_j$

D. $\sigma(\theta^T \boldsymbol{x})x_j[1 - \sigma(\theta^T \boldsymbol{x})x_j]$

E. None/other

Let $z = \theta^T \boldsymbol{x} = \sum_{k=0}^{m} \theta_k x_k$.

$$\frac{\partial}{\partial \theta_j}\sigma(\theta^T \boldsymbol{x}) = \frac{\partial}{\partial z}\sigma(z) \cdot \frac{\partial z}{\partial \theta_j}$$

(Chain Rule)

$$= \sigma(\theta^T \boldsymbol{x})[1 - \sigma(\theta^T \boldsymbol{x})]x_j$$

🤔

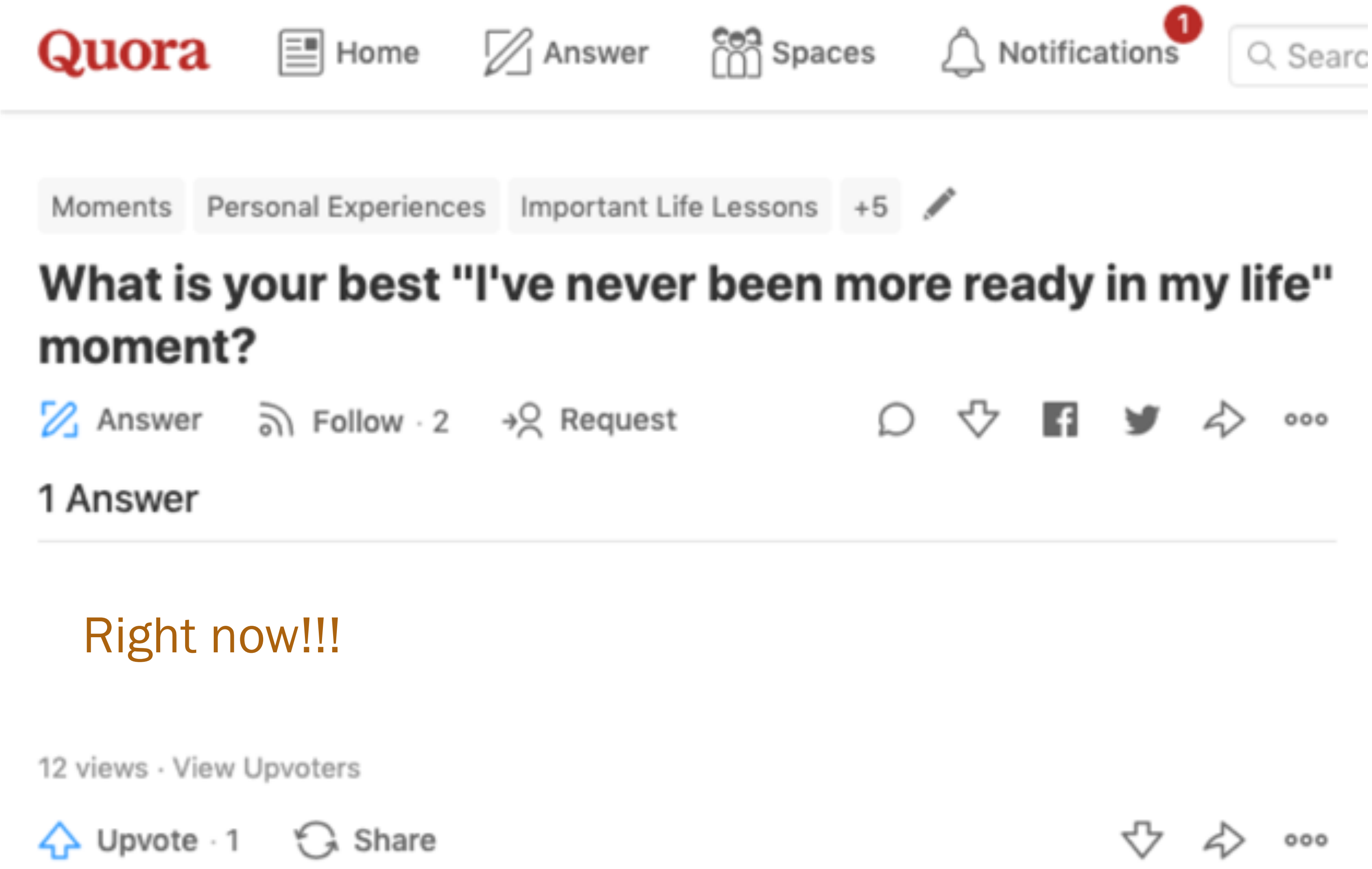
Compute gradient of log-conditional likelihood

Find: $\dfrac{\partial LL(\theta)}{\partial \theta_j}$

where

Log-conditional Likelihood: $LL(\theta) = \sum_{i=1}^{n} y^{(i)} \log \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right)\right)$

Are you ready?



What is your best "I've never been more ready in my life" moment?

1 Answer

Right now!!!

Compute gradient of log-likelihood

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} \frac{\partial}{\partial \theta_j} \left[y^{(i)} \log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \right]$$

Let $\hat{y}^{(i)} = \sigma\left(\theta^T \boldsymbol{x}^{(\boldsymbol{i})}\right)$

$$= \sum_{i=1}^{n} \frac{\partial}{\partial \hat{y}^{(i)}} \left[y^{(i)} \log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \right] \cdot \frac{\partial \hat{y}^{(i)}}{\partial \theta_j}$$

(Chain Rule)

$$= \sum_{i=1}^{n} \left[y^{(i)} \frac{1}{\hat{y}^{(i)}} + \left(1 - y^{(i)}\right) \frac{1}{1 - \hat{y}^{(i)}} \right] \cdot \hat{y}^{(i)}\left(1 - \hat{y}^{(i)}\right) x_j^{(i)}$$

(calculus)

$$= \sum_{i=1}^{n} \left[y^{(i)} - \hat{y}^{(i)} \right] x_j^{(i)} \qquad = \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right) \right] x_j^{(i)}$$

(simplify)

Compute gradient of log-likelihood

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} \frac{\partial}{\partial \theta_j}\left[y^{(i)} \log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)\right]$$

Let $\hat{y}^{(i)} = \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right)$

$$= \sum_{i=1}^{n} \frac{\partial}{\partial \hat{y}^{(i)}}\left[y^{(i)} \log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)\right] \cdot \frac{\partial \hat{y}^{(i)}}{\partial \theta_j}$$

(Chain Rule)

$$= \sum_{i=1}^{n} \left[y^{(i)} \frac{1}{\hat{y}^{(i)}} + \left(1 - y^{(i)}\right) \frac{1}{1 - \hat{y}^{(i)}}\right] \cdot \hat{y}^{(i)}\left(1 - \hat{y}^{(i)}\right) x_j^{(i)}$$

(calculus)

$$= \sum_{i=1}^{n} \left[y^{(i)} - \hat{y}^{(i)}\right] x_j^{(i)} \qquad = \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right)\right] x_j^{(i)}$$

🎉 (simplify)

Today's plan

Naïve Bayes

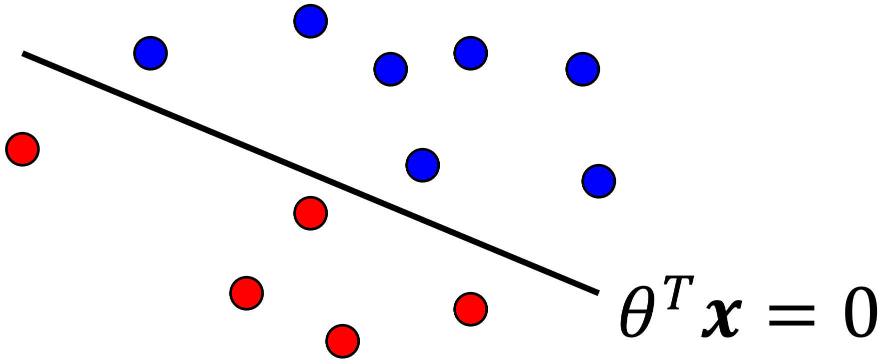
Logistic Regression

- Chapter 0: Background
- Chapter 1: Big Picture
- Chapter 2: Details
- Chapter 3: Philosophy

Intuition about Logistic Regression

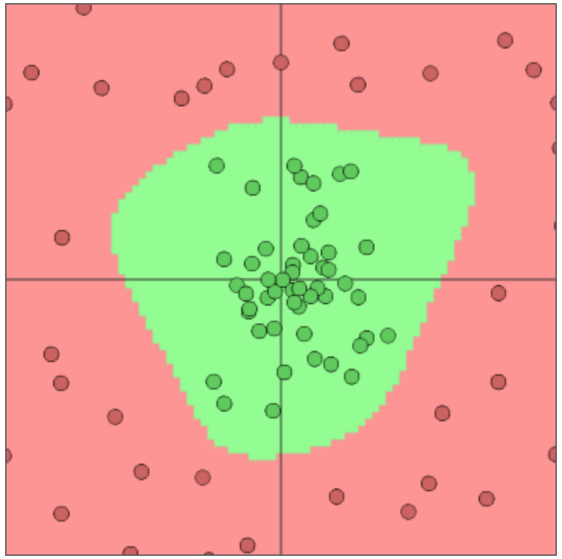
Logistic Regression Model $P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma(\theta^T \boldsymbol{x})$ where $\theta^T \boldsymbol{x} = \sum_{j=0}^{m} \theta_j x_j$

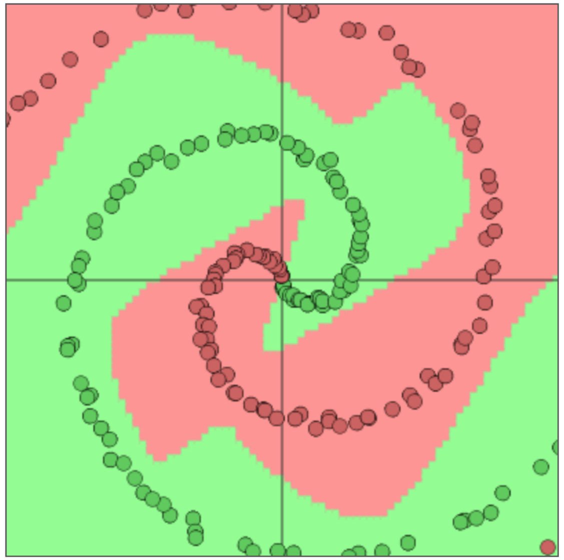
Logistic Regression is trying to fit a <u>line</u> that separates data instances where $y = 1$ from those where $y = 0$:



- We call such data (or functions generating the data <u>linearly separable</u>.
- Naïve Bayes is linear too, because there is no interaction between different features.

Data is often not linearly separable





- Not possible to draw a line that successfully separates all the $y = 1$ points (green) from the $y = 0$ points (red)
- Despite this fact, Logistic Regression and Naive Bayes still often work well in practice

Many tradeoffs in choosing an algorithm

	Naïve Bayes	Logistic Regression
Modeling goal	$P(\boldsymbol{X}, Y)$	$P(Y\|\boldsymbol{X})$
Generative or **discriminative**?	**Generative:** could use joint distribution to generate new points (⚠️but you might not need this extra effort)	**Discriminative:** just tries to discriminate $y = 0$ vs $y = 1$ (cannot generate new points b/c no $P(\boldsymbol{X}, Y)$)
Continuous input features	⚠️ Needs parametric form (e.g., Gaussian) or discretized buckets (for multinomial features)	✅ Yes, easily
Discrete input features	Yes, multi-value discrete data = multinomial $P(X_i\|Y)$	⚠️ Multi-valued discrete data hard (e.g., if $X_i \in \{A, B, C\}$, not necessarily good to encode as $\{1, 2, 3\}$