

Table of Contents

- 2 Preamble
- 6 Logistic Regression
- 23 Training: Big Picture
- 35 Training: Details
- 49 Extra: Gradient Derivation

25: Logistic Regression

Jerry Cain
March 8, 2024

[Lecture Discussion on Ed](#)

Linear to Logical: Preamble

1. Weighted sum

Dot product: $\theta^T X = \sum_{j=1}^m \theta_j X_j$

Recall the linear regression model, where $X = (X_1, X_2, \dots, X_m)$ and $Y \in \mathbb{R}$:

$$g(X) = \theta_0 + \sum_{j=1}^m \theta_j X_j$$

this is our predictor now

if Z is defined to be

$$Z = \theta_1 X_1 + \theta_2 X_2 + \theta_3 X_3$$

then it can be written more compactly

$$Z = \Theta^T X, \text{ where } \Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m \end{bmatrix}$$

How would you rewrite this expression as a single dot product? and $X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}$

$$g(X) = \theta_0 X_0 + \theta_1 X_1 + \theta_2 X_2 + \dots + \theta_m X_m \quad \text{Define } X_0 = 1$$

$$= \theta^T X$$

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

$$\text{and } \Theta = (\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \dots, \theta_m)$$

Prepending $X_0 = 1$ to each feature vector X makes matrix operators more convenient.

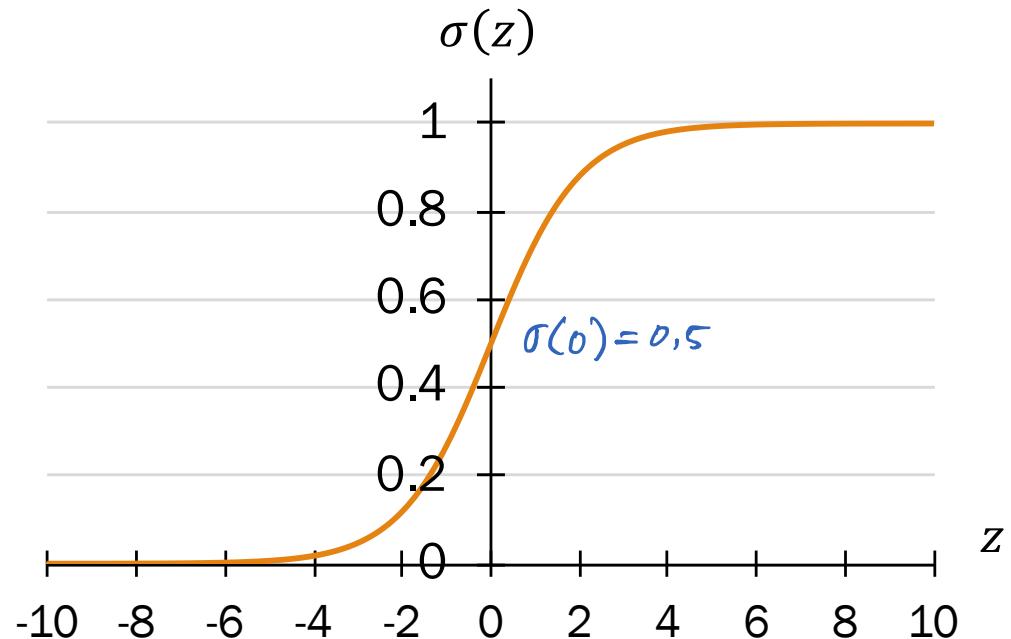


2. Sigmoid function $\sigma(z)$

- 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 that expresses a belief that something is true.

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

3. Conditional likelihood function

Training data (n datapoints):

- $(\mathbf{x}^{(i)}, y^{(i)})$ drawn iid from a distribution $f(\mathbf{X} = \mathbf{x}^{(i)}, Y = y^{(i)} | \theta) = f(\mathbf{x}^{(i)}, y^{(i)} | \theta)$

$$\theta_{MLE} = \arg \max_{\theta} \prod_{i=1}^n f(y^{(i)} | \mathbf{x}^{(i)}, \theta)$$

conditional likelihood
of training data

$$= \arg \max_{\theta} \sum_{i=1}^n \log f(y^{(i)} | \mathbf{x}^{(i)}, \theta)$$

log conditional likelihood

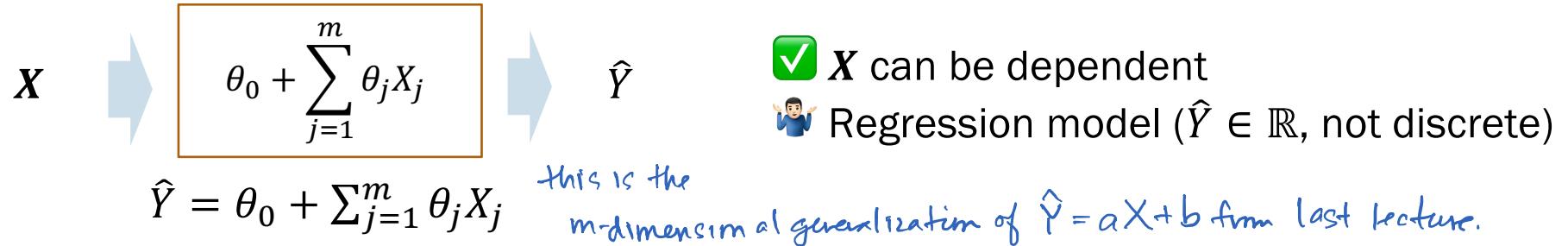
$$= \arg \max_{\theta} LL(\theta)$$

- MLE here is estimator that maximizes **conditional likelihood**
- Confusingly, log conditional likelihood is also written as $LL(\theta)$

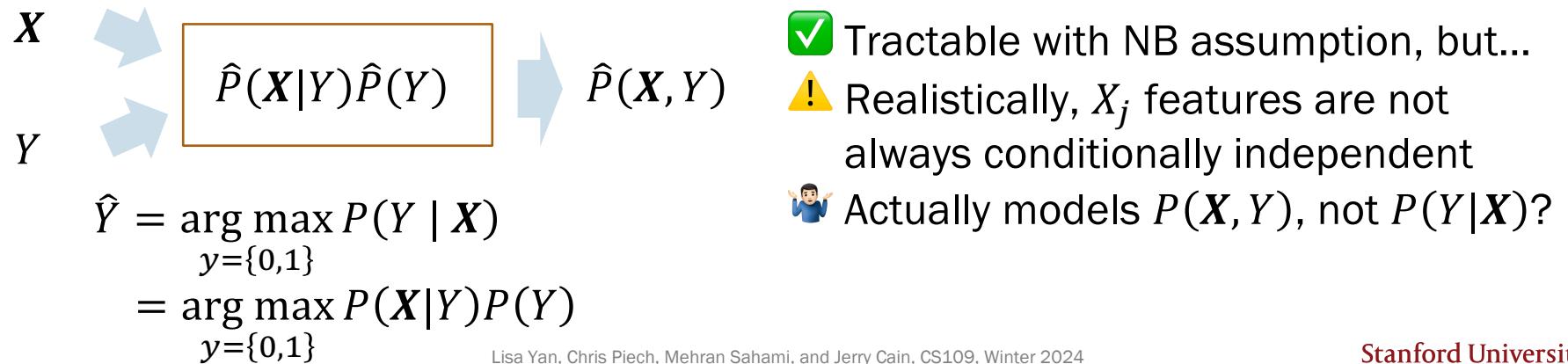
Logistic Regression

Prediction models so far

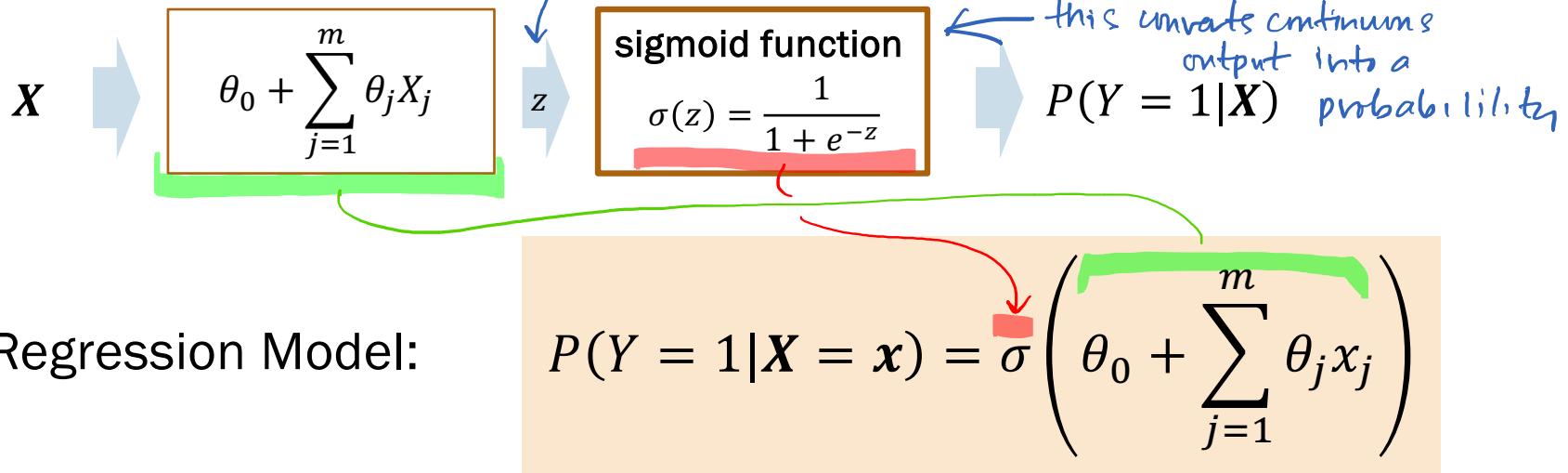
Linear Regression (Regression)



Naïve Bayes (Classification)



Logistic Regression



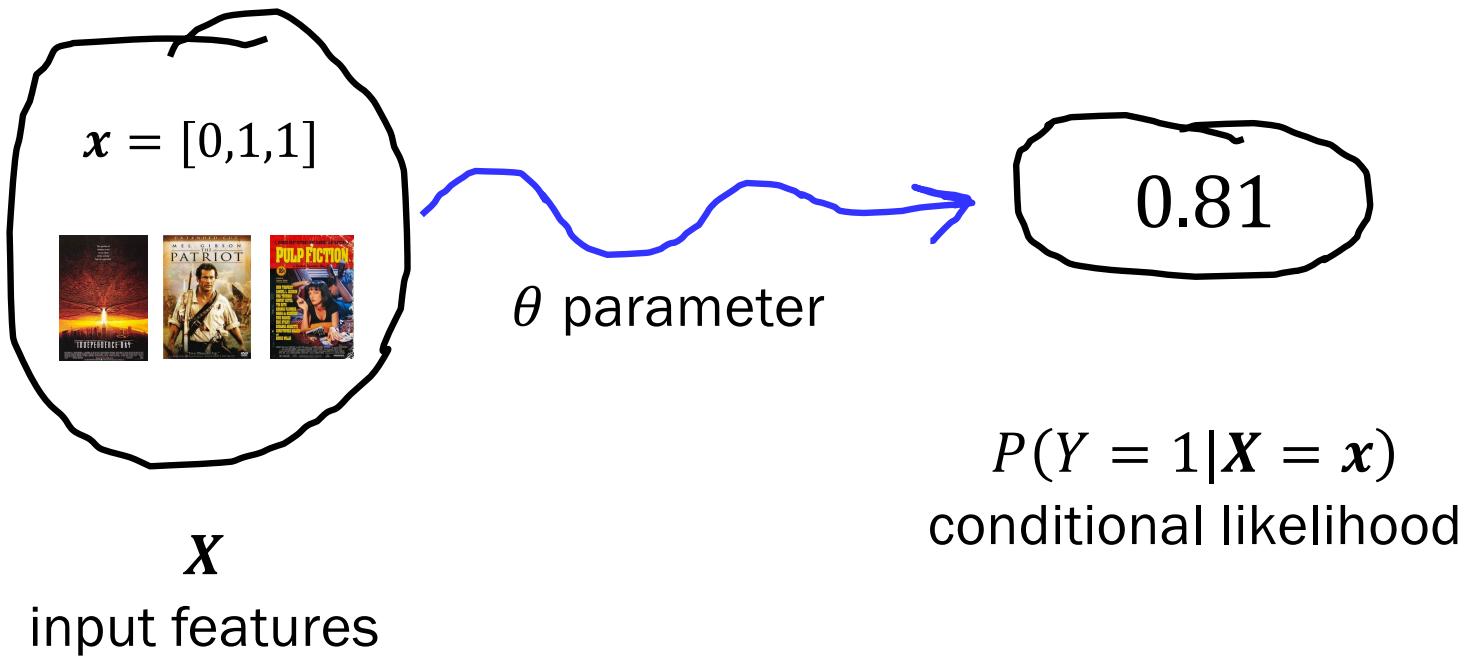
Logistic Regression Model:

Predict \hat{Y} as the more likely Y given our observation $X = x$:

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

- Since $Y \in \{0,1\}$, $P(Y = 0 | X = x) = 1 - \sigma(\theta_0 + \sum_{j=1}^m \theta_j x_j)$
- Sigmoid function also known as **logit function** $= 1 - P(Y=1 | \vec{x} = \vec{x})$

Logistic Regression



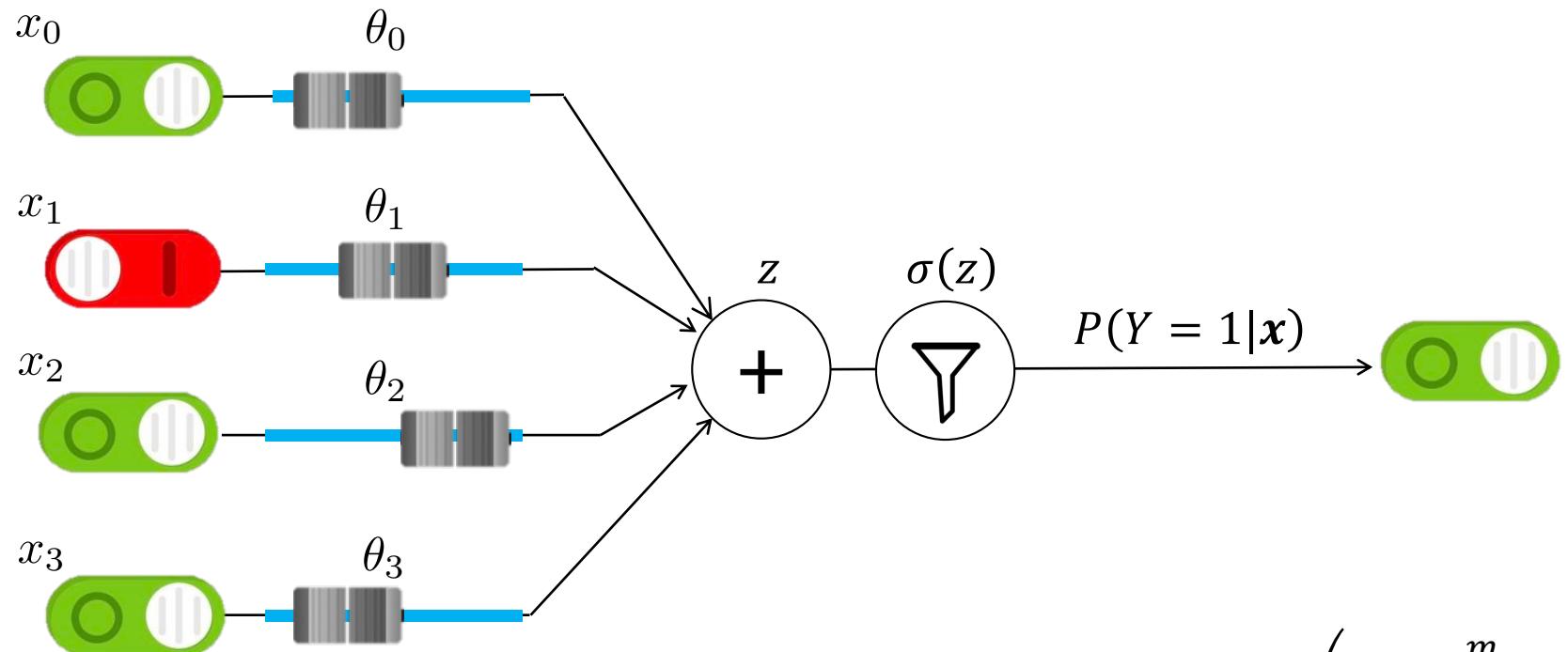
$$P(Y = 1|X = x) = \sigma\left(\theta_0 + \sum_{j=1}^m \theta_j x_j\right)$$

Logistic Regression: Key Metaphor



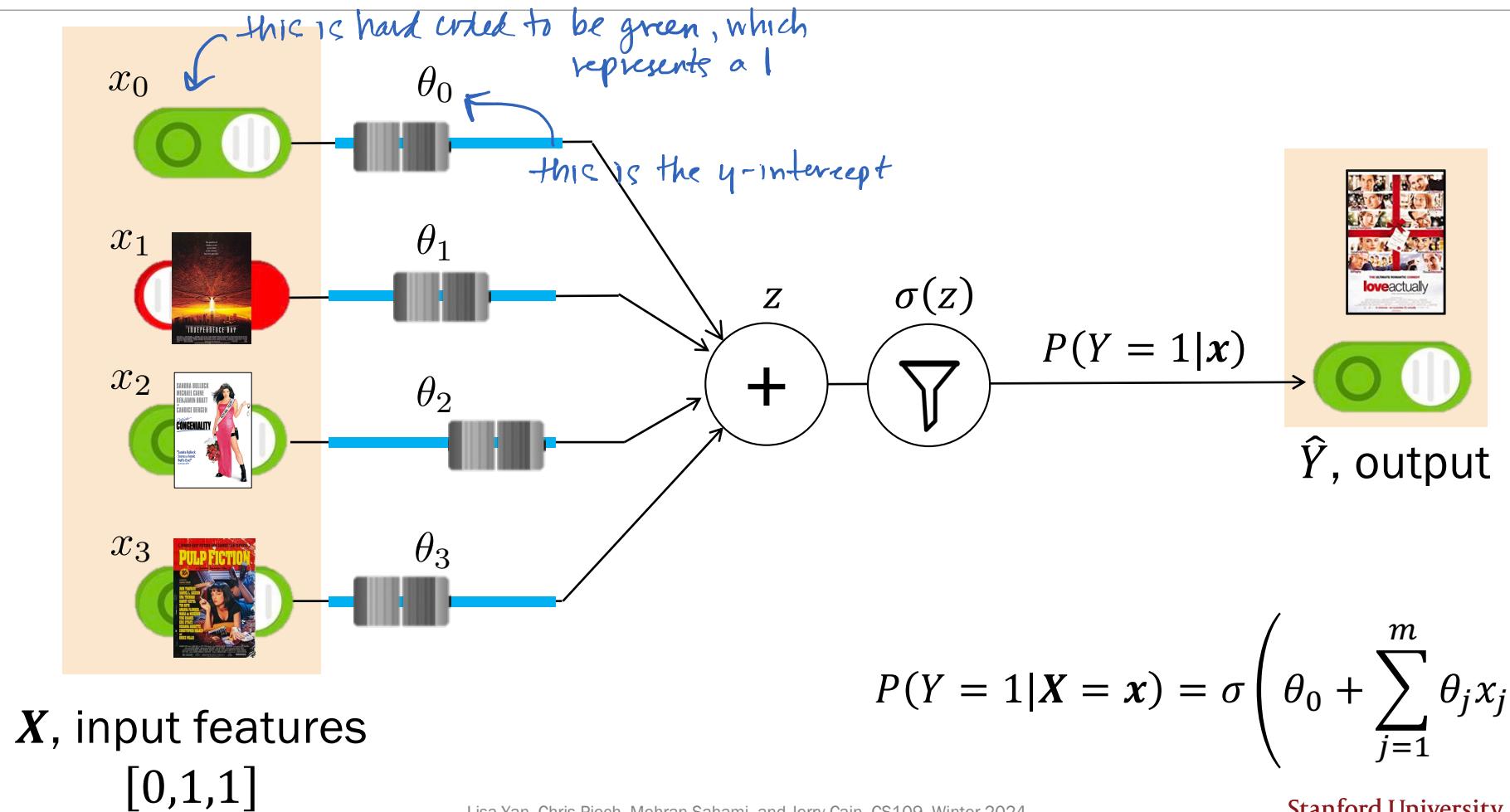
θ parameter

Logistic Regression: Key Metaphor

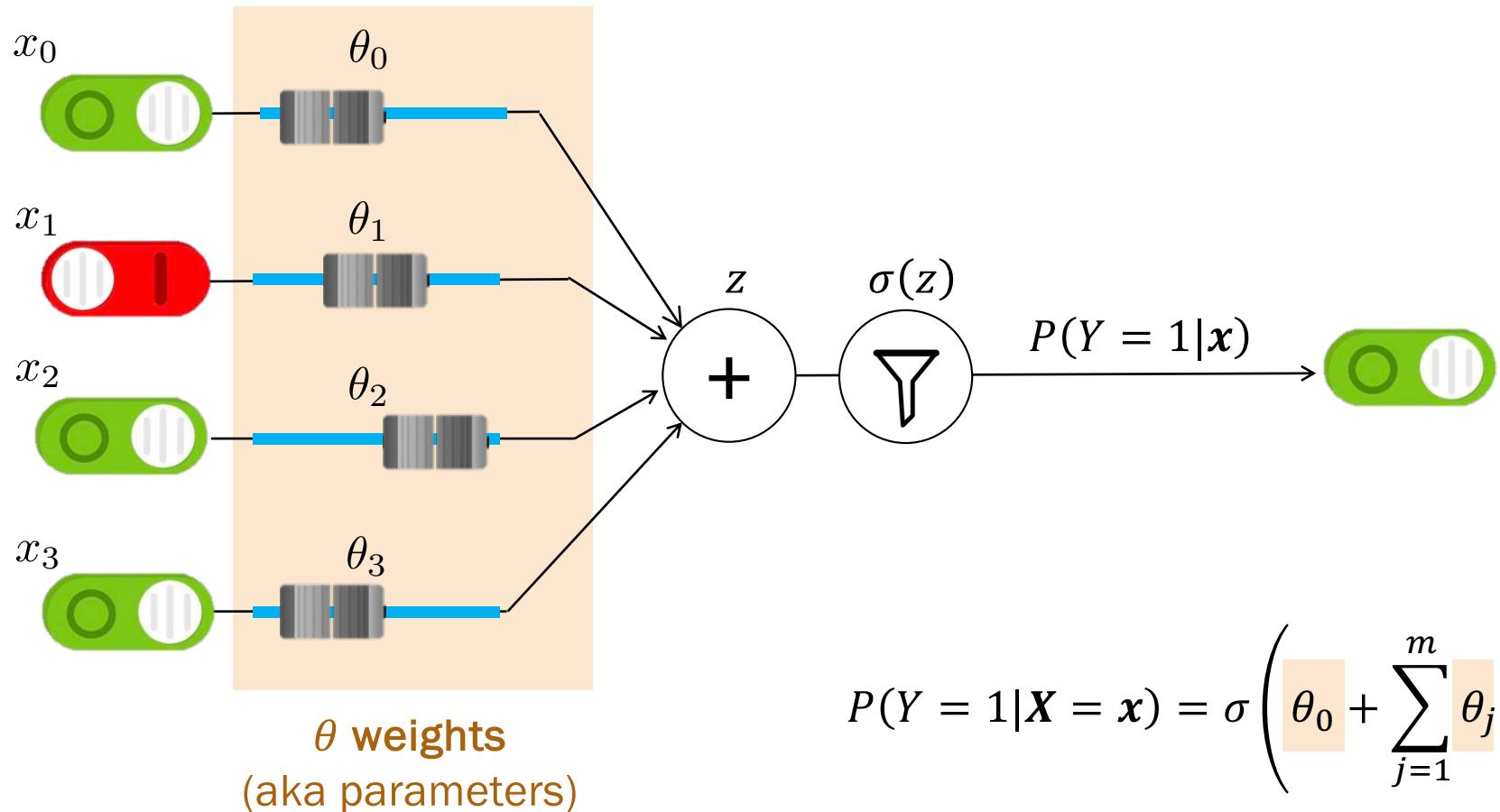


$$P(Y = 1|X = x) = \sigma\left(\theta_0 + \sum_{j=1}^m \theta_j x_j\right)$$

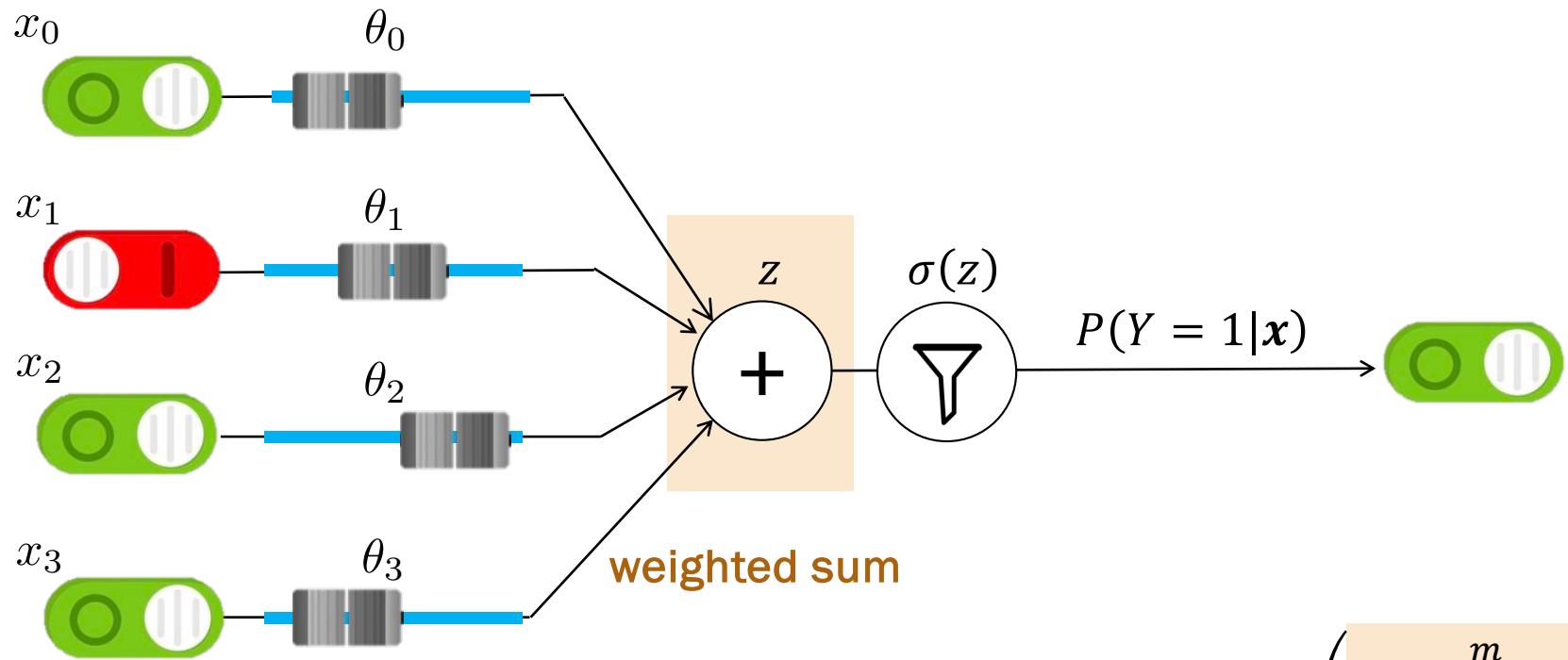
Logistic Regression: Key Metaphor



Components of Logistic Regression

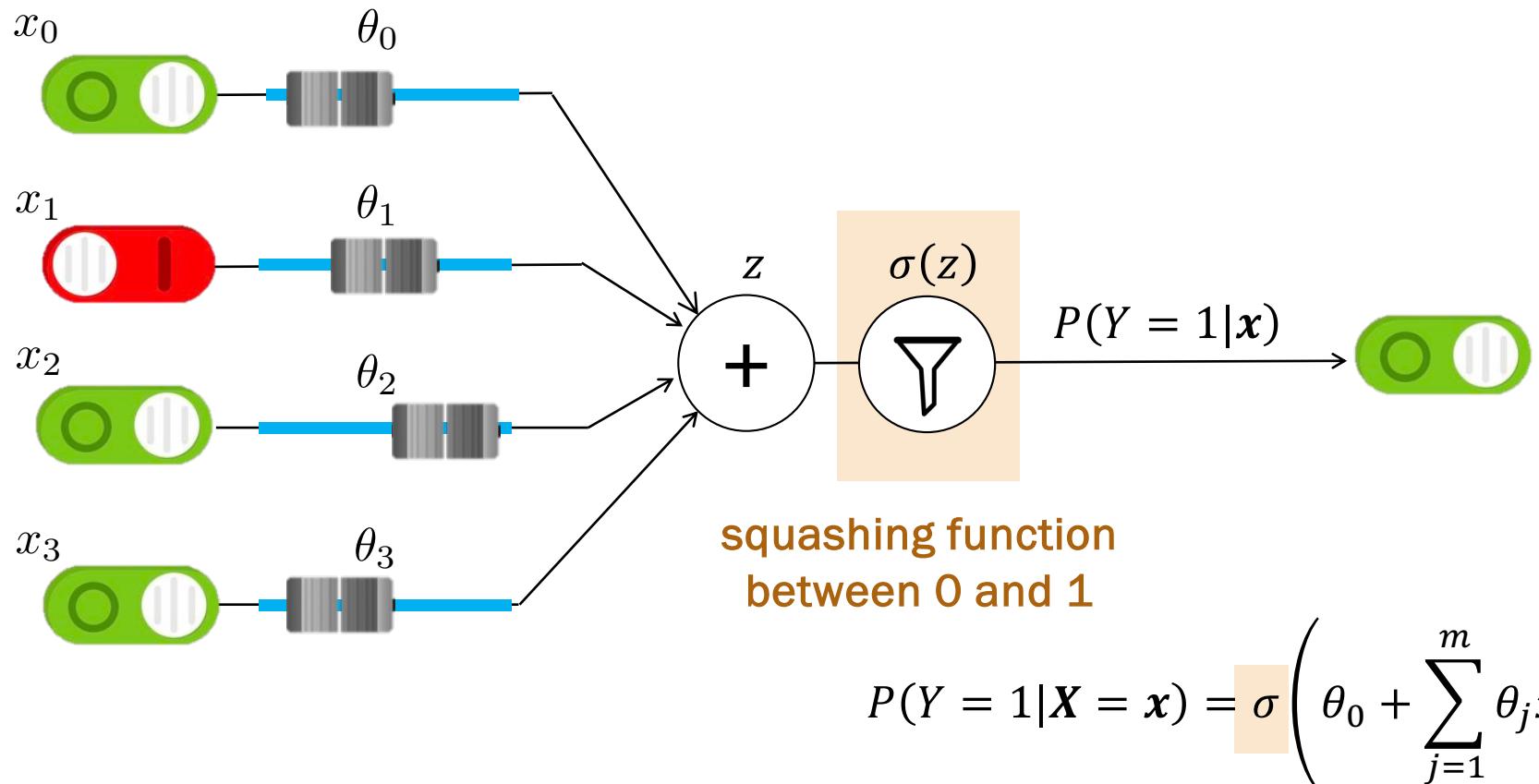


Components of Logistic Regression

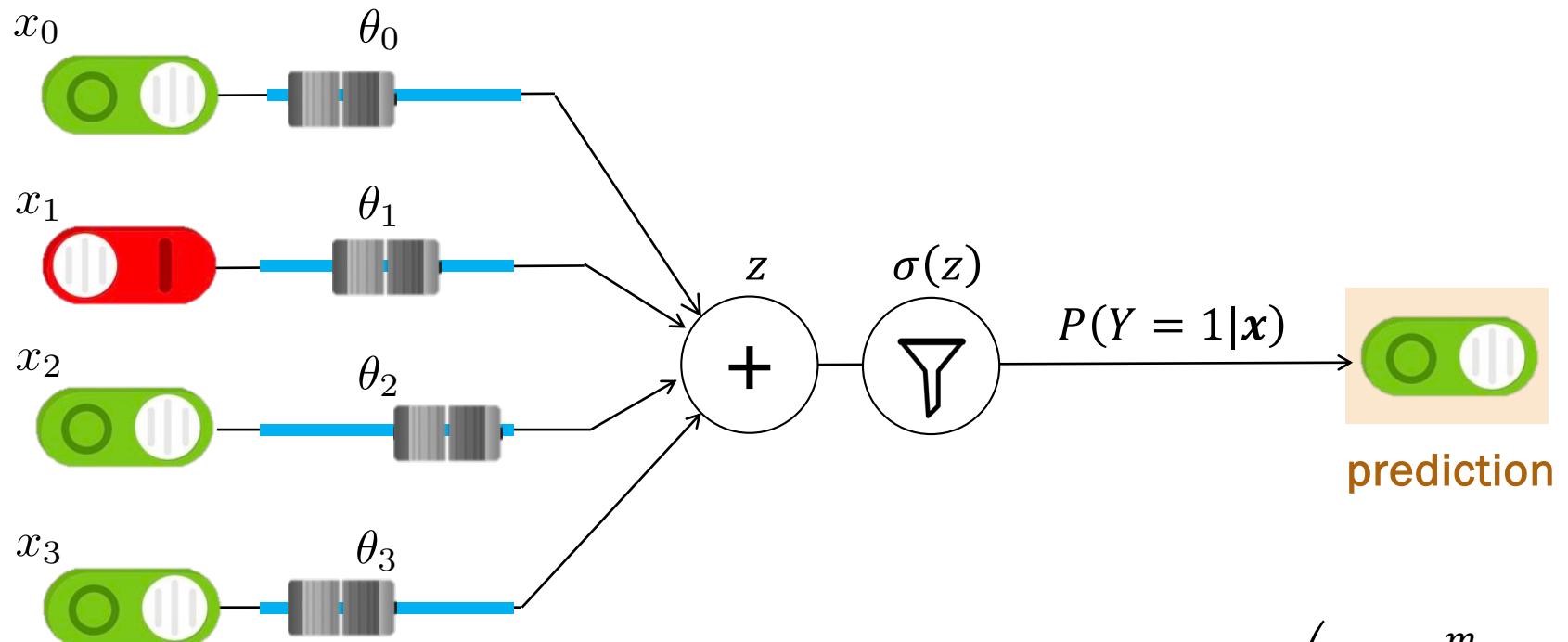


$$P(Y = 1 | X = x) = \sigma \left(\theta_0 + \sum_{j=1}^m \theta_j x_j \right)$$

Components of Logistic Regression

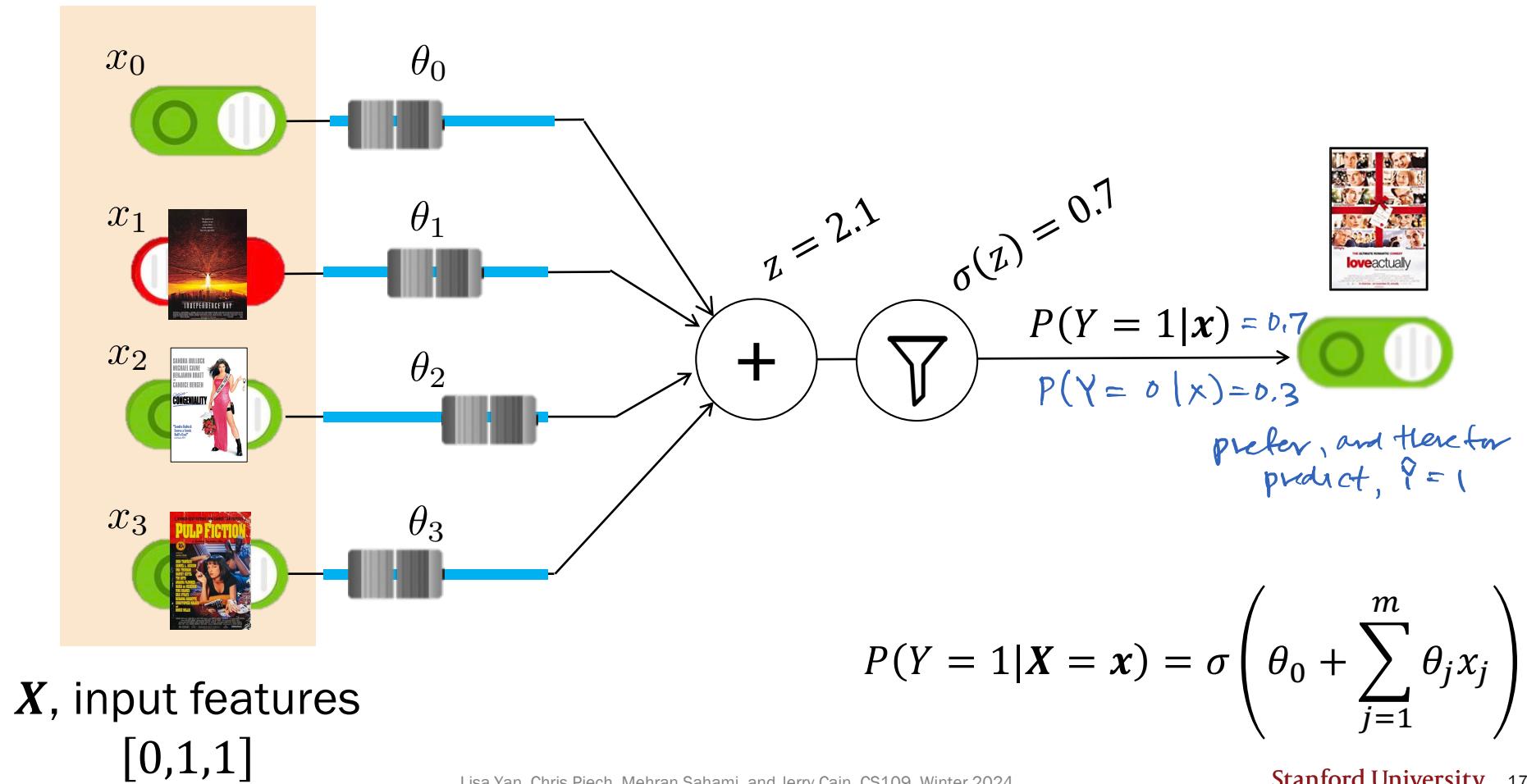


Components of Logistic Regression

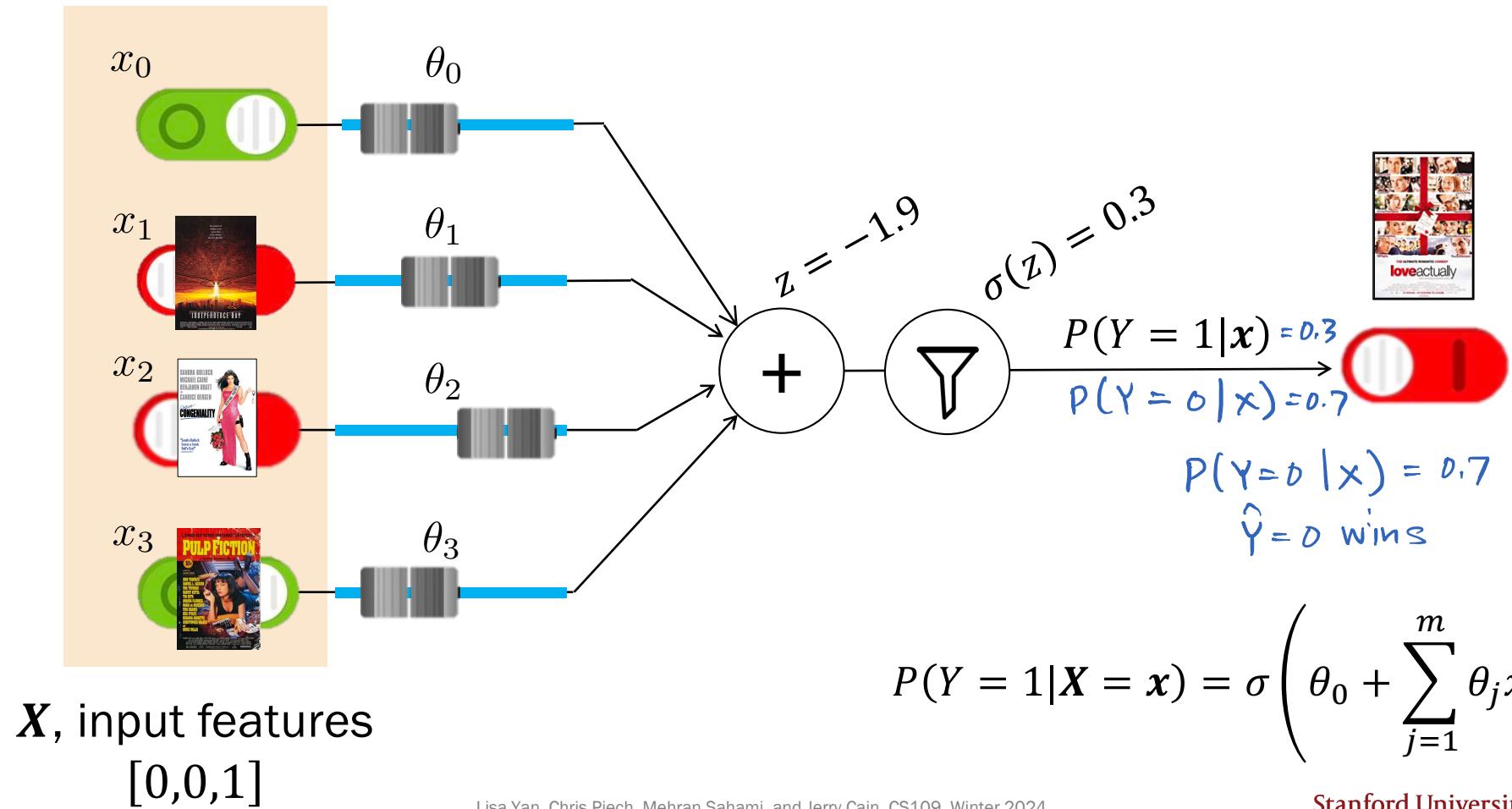


$$P(Y = 1|X = \mathbf{x}) = \sigma\left(\theta_0 + \sum_{j=1}^m \theta_j x_j\right)$$

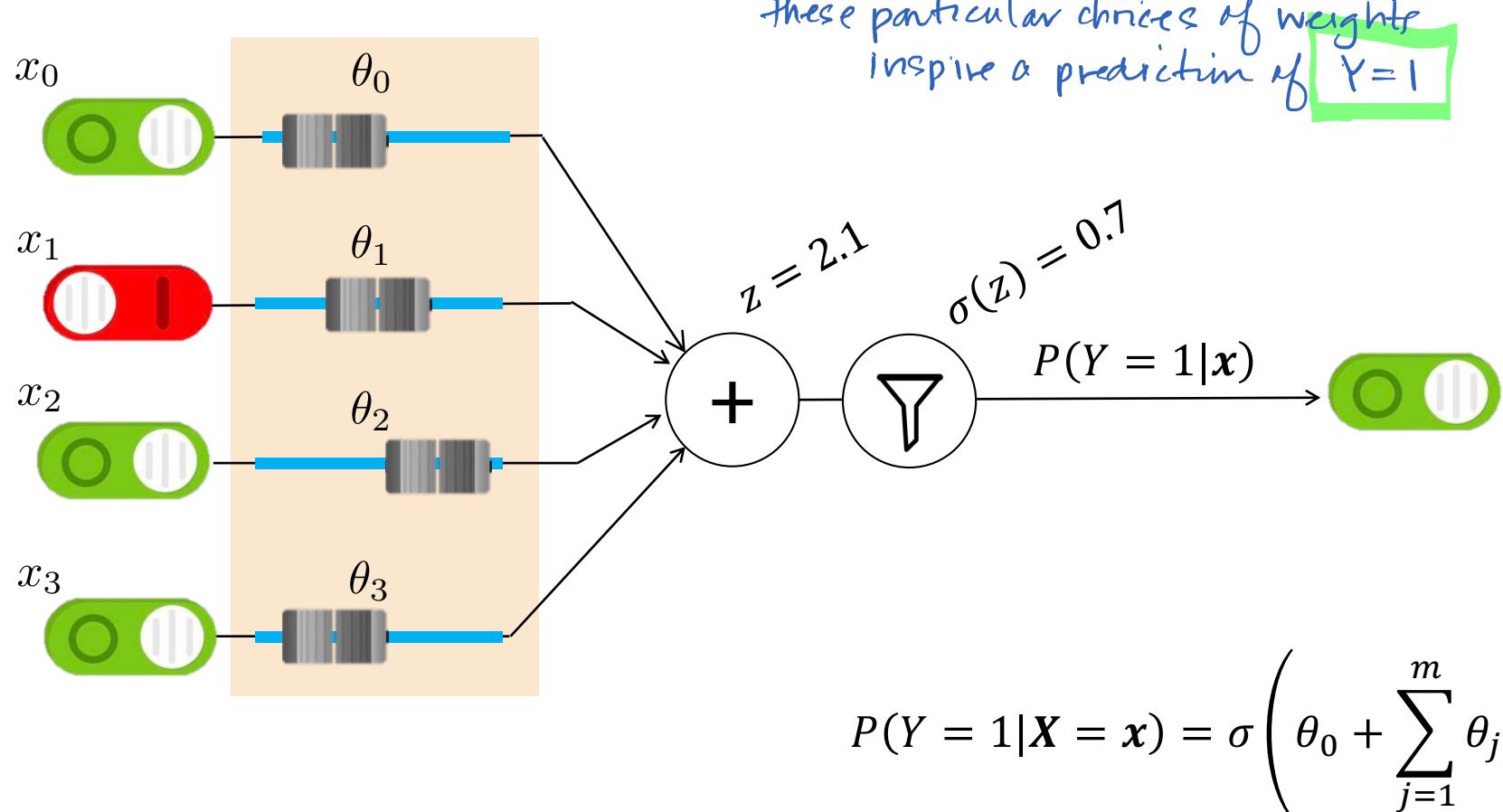
Different predictions for different inputs



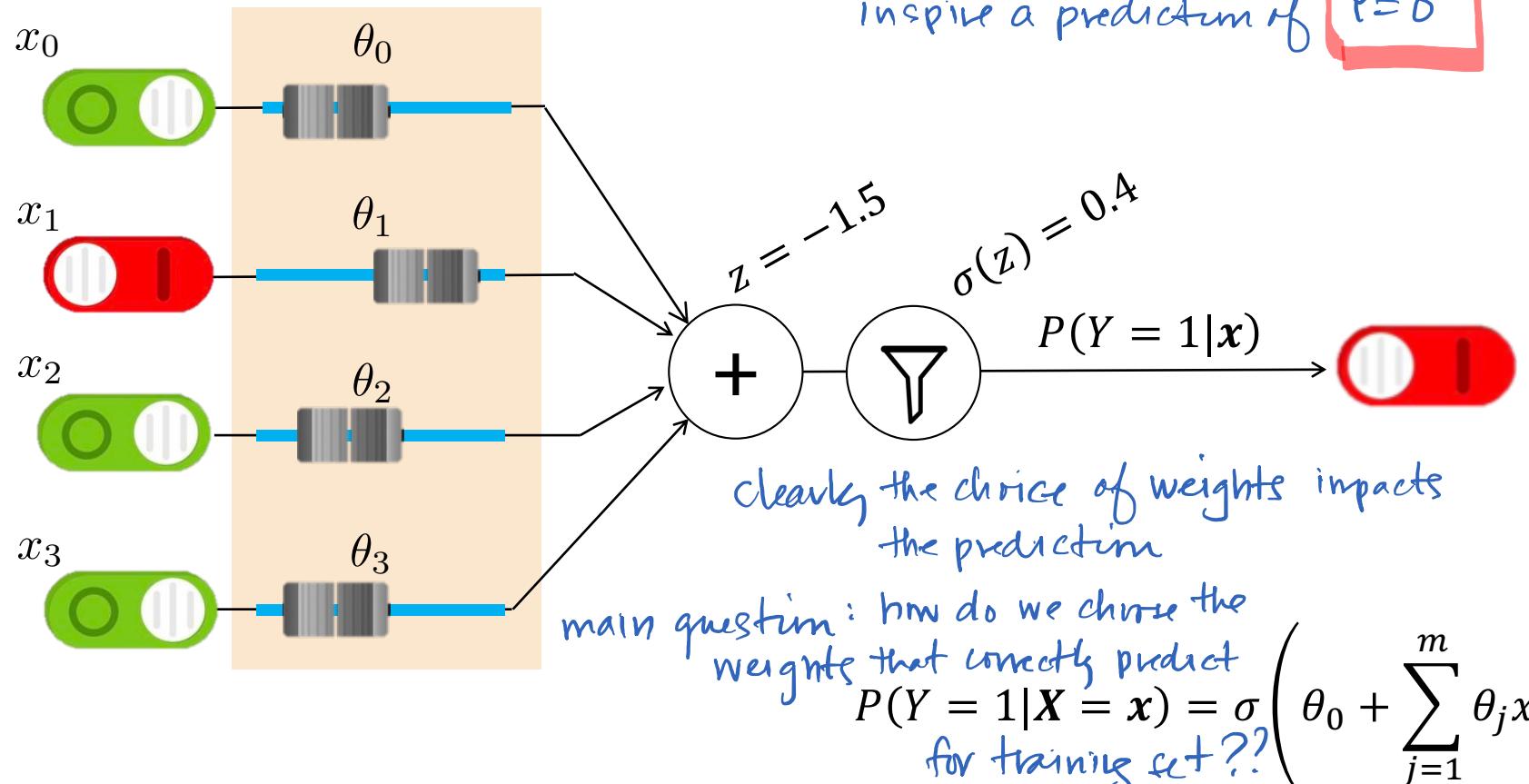
Different predictions for different inputs



Parameters affect prediction



Parameters affect prediction



Parameters affect prediction

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

merge

this is precisely the same training and testing approach used for linear regression, but we extend the process by applying a sigmoid to deliver a probability.

$$P(Y = 1|X = \mathbf{x}) = \sigma \left(\sum_{j=0}^m \theta_j x_j \right) = \sigma(\boldsymbol{\theta}^T \mathbf{x}) \quad \text{where } x_0 = 1$$

Logistic regression classifier

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

$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_{j=0}^m \theta_j x_j\right) = \sigma(\boldsymbol{\theta}^T \mathbf{x})$$

Training

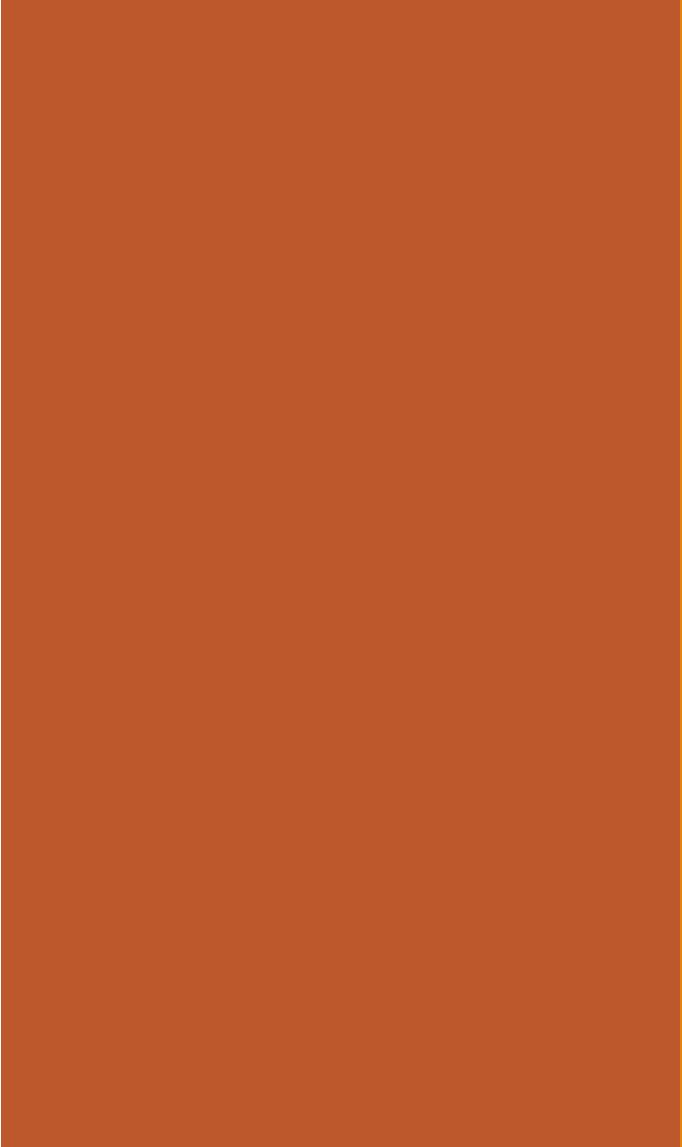
Estimate parameters
from training data

$$\boldsymbol{\theta} = (\theta_0, \theta_1, \theta_2, \dots, \theta_m)$$

Testing

Given an observation $\mathbf{X} = (X_1, X_2, \dots, X_m)$, predict

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



Training: The big picture

Logistic regression classifier

$$\hat{Y} = \arg \max_{y=\{0,1\}} P(Y|X) \quad \left. \begin{array}{l} \text{compute probabilities of } Y=0 \text{ versus } Y=1, \text{ and} \\ \text{choose the } Y \text{ value that comes with the} \\ \text{higher probability.} \end{array} \right\}$$

$$P(Y = 1|X = \mathbf{x}) = \sigma\left(\sum_{j=0}^m \theta_j x_j\right) = \sigma(\theta^T \mathbf{x})$$

Training

Estimate parameters
from training data

$$\theta = (\theta_0, \theta_1, \theta_2, \dots, \theta_m)$$

Choose θ that optimizes some objective:

1. Determine objective function
2. Find gradient with respect to each θ
3. Solve analytically by setting to 0, or solve computationally with gradient ascent

(all m of them!)

We are modeling $P(Y|X)$ directly, so we maximize the conditional likelihood of training data.

Estimating θ

1. Determine objective function

$$\theta_{MLE} = \arg \max_{\theta} \prod_{i=1}^n f(y^{(i)} | \mathbf{x}^{(i)}, \theta)$$

2. Gradient wrt θ_j , for $j = 0, 1, \dots, m$

3. Solve

- No analytical derivation of θ_{MLE} ...
- ...but can still determine θ_{MLE} via gradient ascent!

```
initialize x  
repeat many times:  
    compute gradient  
    x += η * gradient
```

1. Determine objective function

$$\theta_{MLE} = \arg \max_{\theta} \prod_{i=1}^n f(y^{(i)} | \mathbf{x}^{(i)}, \theta) = \arg \max_{\theta} LL(\theta)$$

$$P(Y = 1 | \mathbf{X} = \mathbf{x}) = \sigma(\sum_{j=0}^m \theta_j x_j) \\ = \sigma(\theta^T \mathbf{x})$$

First: Interpret
conditional likelihood
with Logistic Regression

Second: Write a differentiable
expression for log conditional
likelihood

1. Determine objective function (interpret)

$$\theta_{MLE} = \arg \max_{\theta} \prod_{i=1}^n f(y^{(i)} | \mathbf{x}^{(i)}, \theta) = \arg \max_{\theta} LL(\theta)$$

$$P(Y = 1 | \mathbf{X} = \mathbf{x}) = \sigma\left(\sum_{j=0}^m \theta_j x_j\right) \\ = \sigma(\theta^T \mathbf{x})$$

Suppose you have $n = 2$ training datapoints:

$$(\mathbf{x}^{(1)}, 1), (\mathbf{x}^{(2)}, 0)$$

Consider the following expressions for a given θ :

A. $\sigma(\theta^T \mathbf{x}^{(1)}) \sigma(\theta^T \mathbf{x}^{(2)})$
 $P(Y=1 | \mathbf{X}=\mathbf{x}^{(1)}) P(Y=1 | \mathbf{X}=\mathbf{x}^{(2)})$

B. $(1 - \sigma(\theta^T \mathbf{x}^{(1)})) \sigma(\theta^T \mathbf{x}^{(2)})$
 $P(Y=0 | \mathbf{X}=\mathbf{x}^{(1)}) P(Y=1 | \mathbf{X}=\mathbf{x}^{(2)})$

C. $\sigma(\theta^T \mathbf{x}^{(1)}) (1 - \sigma(\theta^T \mathbf{x}^{(2)}))$
 $P(Y=1 | \mathbf{X}=\mathbf{x}^{(1)}) P(Y=0 | \mathbf{X}=\mathbf{x}^{(2)})$

D. $(1 - \sigma(\theta^T \mathbf{x}^{(1)})) (1 - \sigma(\theta^T \mathbf{x}^{(2)}))$
 $P(Y=0 | \mathbf{X}=\mathbf{x}^{(1)}) P(Y=0 | \mathbf{X}=\mathbf{x}^{(2)})$

1. Interpret the above expressions as probabilities.
2. If we let $\theta = \theta_{MLE}$, which probability should be the highest?

C shnld be highest.

1. Determine objective function (write)

$$\theta_{MLE} = \arg \max_{\theta} \prod_{i=1}^n f(y^{(i)} | x^{(i)}, \theta) = \arg \max_{\theta} LL(\theta)$$

$$P(Y = 1 | X = x) = \sigma\left(\sum_{j=0}^m \theta_j x_j\right) \\ = \sigma(\theta^T x)$$

1. What is a differentiable expression for $P(Y = y | X = x)$?
 remember for $B \sim Ber(p)$

Recall
 Bern(B=b) = $\begin{cases} p & b=1 \\ 1-p & b=0 \end{cases}$
 MLE!

was
 written
 to be
 $p(B=b) = p^b (1-p)^{1-b}$

$$P(Y = y | X = x) = \begin{cases} \sigma(\theta^T x) & \text{if } y = 1 \\ 1 - \sigma(\theta^T x) & \text{if } y = 0 \end{cases}$$

similarly, rewrite the above

$$p(y|x) = \sigma(\theta^T x)^y (1 - \sigma(\theta^T x))^{1-y}$$

this is a continuous form that
differentiable

$$LL(\theta) = \log \prod_{i=1}^n f(y^{(i)} | x^{(i)}, \theta)$$

2. What is a differentiable expression for $LL(\theta)$, log conditional likelihood?

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

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



1. Determine objective function (write)

$$\theta_{MLE} = \arg \max_{\theta} \prod_{i=1}^n f(y^{(i)} | \mathbf{x}^{(i)}, \theta) = \arg \max_{\theta} LL(\theta)$$
$$P(Y = 1 | \mathbf{X} = \mathbf{x}) = \sigma\left(\sum_{j=0}^m \theta_j x_j\right) \\ = \sigma(\theta^T \mathbf{x})$$

1. What is a differentiable expression for $P(Y = y | \mathbf{X} = \mathbf{x})$?

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

2. What is a differentiable expression for $LL(\theta)$, log conditional likelihood?

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

2. Find gradient with respect to θ

Optimization problem:

$$\theta_{MLE} = \arg \max_{\theta} \prod_{i=1}^n f(y^{(i)} | \mathbf{x}^{(i)}, \theta) = \arg \max_{\theta} LL(\theta)$$
$$LL(\theta) = \sum_{i=1}^n y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log (1 - \sigma(\theta^T \mathbf{x}^{(i)}))$$

Gradient wrt θ_j , for $j = 0, 1, \dots, m$:

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

🦄 presented here without proof, though it is 🦄
a generalization of the gradient of a in the
 $Y = aX + b + Z$ derivation from last lecture

How do we interpret the gradient contribution of the i^{th} training datapoint?



2. Find gradient with respect to θ

Optimization
problem:

$$\theta_{MLE} = \arg \max_{\theta} \prod_{i=1}^n f(y^{(i)} | \mathbf{x}^{(i)}, \theta) = \arg \max_{\theta} LL(\theta)$$
$$LL(\theta) = \sum_{i=1}^n y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log (1 - \sigma(\theta^T \mathbf{x}^{(i)}))$$

Gradient wrt θ_j , for $j = 0, 1, \dots, m$:

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



scale by j-th feature

2. Find gradient with respect to θ

Optimization
problem:

$$\theta_{MLE} = \arg \max_{\theta} \prod_{i=1}^n f(y^{(i)} | \mathbf{x}^{(i)}, \theta) = \arg \max_{\theta} LL(\theta)$$
$$LL(\theta) = \sum_{i=1}^n y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log (1 - \sigma(\theta^T \mathbf{x}^{(i)}))$$

Gradient wrt θ_j , for $j = 0, 1, \dots, m$:

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


1 or 0 $P(Y = 1 | \mathbf{X} = \mathbf{x}^{(i)})$

2. Find gradient with respect to θ

Optimization problem:

$$\theta_{MLE} = \arg \max_{\theta} \prod_{i=1}^n f(y^{(i)} | \mathbf{x}^{(i)}, \theta) = \arg \max_{\theta} LL(\theta)$$
$$LL(\theta) = \sum_{i=1}^n y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log (1 - \sigma(\theta^T \mathbf{x}^{(i)}))$$

Gradient wrt θ_j , for $j = 0, 1, \dots, m$:

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^n [y^{(i)} - \underbrace{\sigma(\theta^T \mathbf{x}^{(i)})}_{\text{orange bracket}}] x_j^{(i)}$$

Suppose $y^{(i)} = 1$ (the true class label for the i^{th} datapoint):

- If $\sigma(\theta^T \mathbf{x}^{(i)}) \geq 0.5$, correct, so $y^{(i)} - \hat{y}^{(i)}$ is small
- If $\sigma(\theta^T \mathbf{x}^{(i)}) < 0.5$, incorrect → change θ_j more

3. Solve

1. Optimization problem:

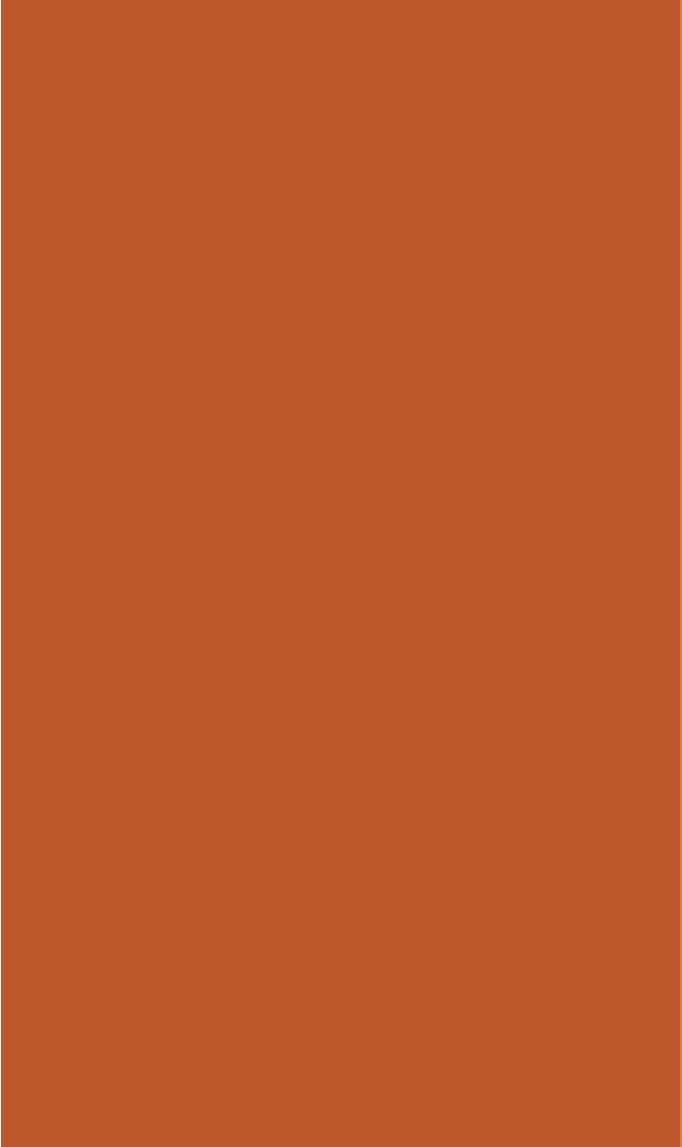
$$\theta_{MLE} = \arg \max_{\theta} \prod_{i=1}^n f(y^{(i)} | \mathbf{x}^{(i)}, \theta) = \arg \max_{\theta} LL(\theta)$$

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

2. Gradient wrt θ_j , for $j = 0, 1, \dots, m$:

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

3. Solve using gradient ascent!



Training: The details

Training: Gradient ascent step

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^n [y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)})] x_j^{(i)} \quad \text{for } j = 0, 1, \dots, m$$

repeat until convergence:

for all thetas:

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

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

What does
this look like
in code?

Training: Gradient Ascent

for $j = 0, 1, \dots, m$:

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

initialize $\theta_j = 0$ for $0 \leq j \leq m$ e.g. $\Theta = [0, 0, 0, 0]$ for $m=3$
repeat until convergence:

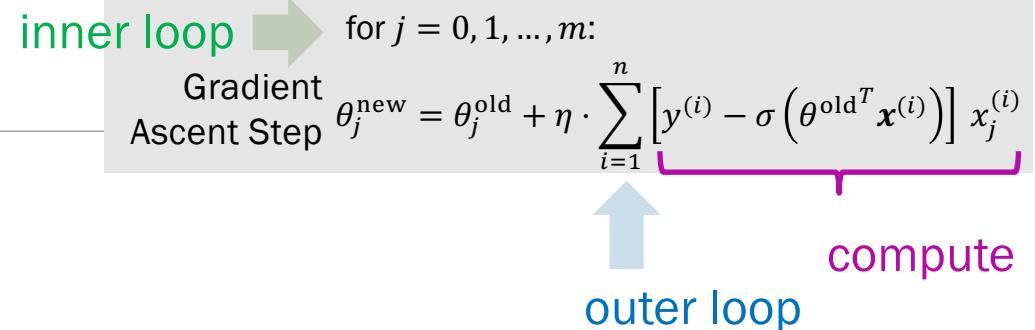
gradient[j] = 0 for $0 \leq j \leq m$ e.g. gradient = [0, 0, 0, 0] for $m=3$

// TODO: your code here

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

$\theta_j += \eta * \text{gradient}[j]$ for all $0 \leq j \leq m$

Training: Gradient Ascent



```
initialize  $\theta_j = 0$  for  $0 \leq j \leq m$ 
repeat until convergence:
```

```
    gradient[j] = 0 for  $0 \leq j \leq m$ 
```

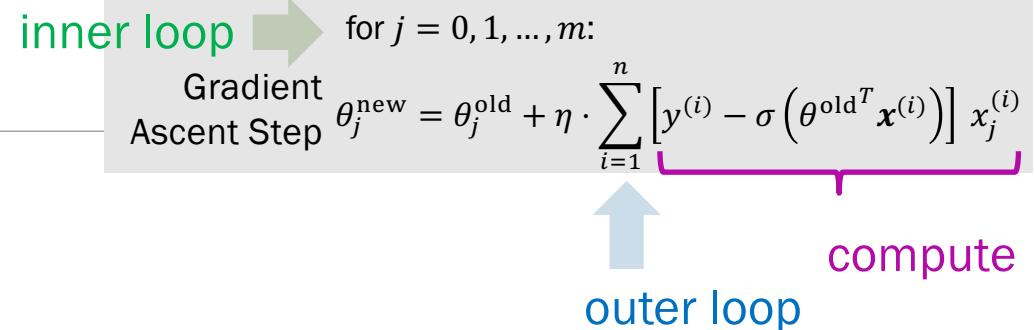
```
    for each training example  $(x, y)$ :
```

```
        for each  $0 \leq j \leq m$ :
```

```
            // update gradient[j] for
            // current  $(x, y)$  example
```

```
     $\theta_j += \eta * \text{gradient}[j]$  for all  $0 \leq j \leq m$ 
```

Training: Gradient Ascent



```
initialize  $\theta_j = 0$  for  $0 \leq j \leq m$ 
repeat until convergence :
```

```
    gradient[j] = 0 for  $0 \leq j \leq m$ 
```

```
        for each training example  $(x, y)$ :
```

```
            for each  $0 \leq j \leq m$ :
```

$$\text{gradient}[j] += \left[y - \frac{1}{1 + e^{-\theta^T x}} \right] x_j$$

```
             $\theta_j += \eta * \text{gradient}[j]$  for all  $0 \leq j \leq m$ 
```

Some important details...

Training: Gradient Ascent

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

initialize $\theta_j = 0$ for $0 \leq j \leq m$

repeat until convergence:

gradient[j] = 0 for $0 \leq j \leq m$

for each training example (x, y) :

for each $0 \leq j \leq m$:

$$\text{gradient}[j] += \left[y - \frac{1}{1 + e^{-\theta^T x}} \right] x_j$$

$\theta_j += \eta * \text{gradient}[j]$ for all $0 \leq j \leq m$

- Finish computing gradient with θ^{old} prior to any θ update

Training: Gradient Ascent

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

initialize $\theta_j = 0$ for $0 \leq j \leq m$

repeat until convergence:

 gradient[j] = 0 for $0 \leq j \leq m$

 for each training example (x, y) :

 for each $0 \leq j \leq m$:

$$\text{gradient}[j] += \left[y - \frac{1}{1 + e^{-\theta^T x}} \right] x_j$$

$\theta_j += n * \text{gradient}[j]$ for all $0 \leq j \leq m$

- Finish computing gradient with θ^{old} prior to any θ update
- Learning rate η is a constant you set before training

Training: Gradient Ascent

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

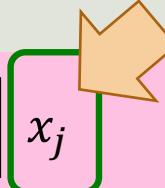
initialize $\theta_j = 0$ for $0 \leq j \leq m$

repeat until convergence:

gradient[j] = 0 for $0 \leq j \leq m$

for each training example (\mathbf{x}, y) :

for each $0 \leq j \leq m$:

$$\text{gradient}[j] += \left[y - \frac{1}{1 + e^{-\theta^T \mathbf{x}}} \right] x_j$$


$\theta_j += \eta * \text{gradient}[j]$ for all $0 \leq j \leq m$

- Finish computing gradient with θ^{old} prior to any θ update
- Learning rate η is a constant you set before training
- x_j is the j^{th} feature of input $\mathbf{x} = (x_1, \dots, x_m)$

Training: Gradient Ascent

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

initialize $\theta_j = 0$ for $0 \leq j \leq m$

repeat until convergence:

gradient[j] = 0 for $0 \leq j \leq m$

for each training example (x, y) :

for each $0 \leq j \leq m$:

$$\text{gradient}[j] += \left[y - \frac{1}{1 + e^{-\theta^T x}} \right] x_j$$

$\theta_j += \eta * \text{gradient}[j]$ for all $0 \leq j \leq m$

- Finish computing gradient with θ^{old} prior to any θ update
- Learning rate η is a constant you set before training
- x_j is the j^{th} feature of input $x = (x_1, \dots, x_m)$
- Insert $x_0 = 1$ before training

Training: Gradient Ascent

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

initialize $\theta_j = 0$ for $0 \leq j \leq m$

repeat until convergence:

gradient[j] = 0 for $0 \leq j \leq m$

for each training example (\mathbf{x}, y) :

for each $0 \leq j \leq m$:

$$\text{gradient}[j] += \left[y - \frac{1}{1 + e^{-\theta^T \mathbf{x}}} \right] x_j$$

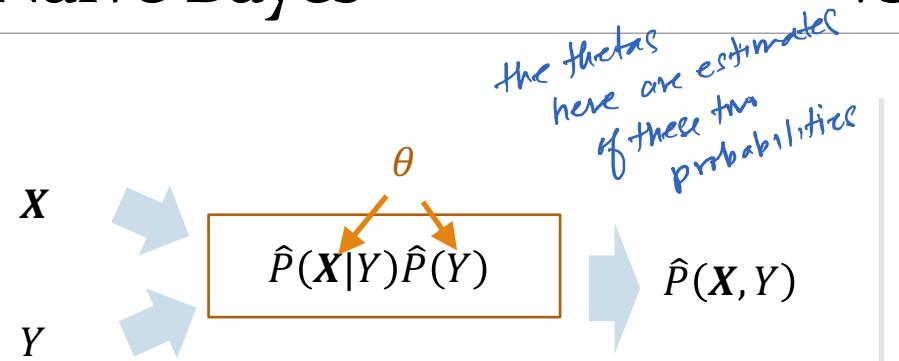
$\theta_j += \eta * \text{gradient}[j]$ for all $0 \leq j \leq m$

- Finish computing gradient with θ^{old} prior to any θ update
- Learning rate η is a constant you set before training
- x_j is the j^{th} feature of input $\mathbf{x} = (x_1, \dots, x_m)$
- Insert $x_0 = 1$ before training

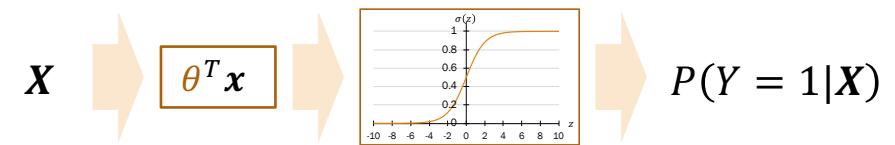


Naïve Bayes

Logistic Regression



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



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

Compare/contrast:

1. What distributions are we modeling?
2. After learning our parameters, could we randomly generate a new datapoint (x, y) ?
3. Could we model a continuous X_j feature (e.g., $X_j \sim \text{Normal}$, or $X_j \sim \text{Unknown}$)?
4. Could we model a non-binary discrete X_j (e.g., $X_j \in \{1, 2, \dots, 6\}$)?



Tradeoffs:

1. Modeling goal

Naïve Bayes

$$P(\mathbf{X}, Y)$$

2. Generative or discriminative?

Generative: could use joint distribution to generate new points (⚠️ but you might not need this extra effort)

3. Continuous input features

⚠️ Needs parametric form (e.g., Gaussian) or discretized buckets (for multinomial features)

4. Discrete input features

✓ Yes, multi-value discrete data = multinomial $P(X_i|Y)$

Logistic Regression

$$P(Y|\mathbf{X})$$

Discriminative: just tries to discriminate $y = 0$ vs $y = 1$ (✗ cannot generate new points b/c no $P(\mathbf{X}, Y)$)

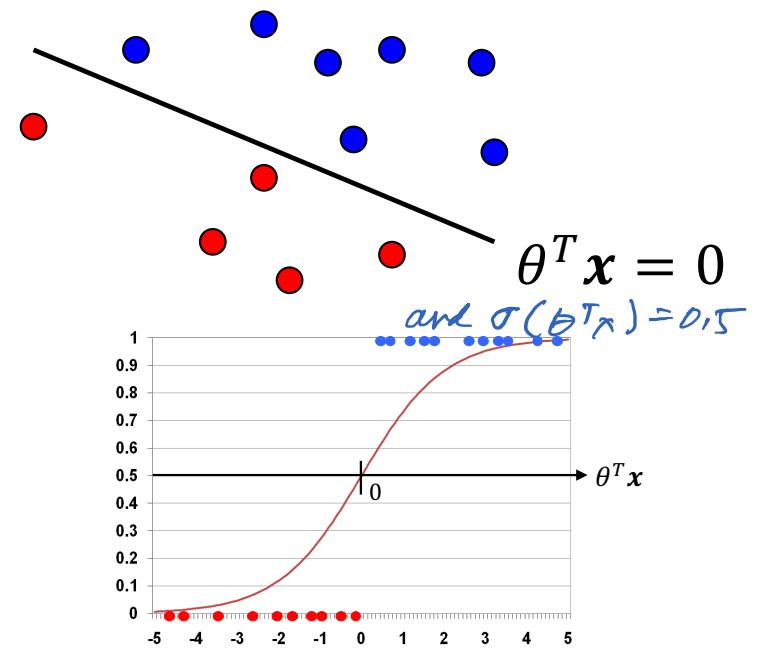
✓ Yes, easily

⚠️ Multi-valued discrete data hard (e.g., if $X_i \in \{A, B, C\}$, not necessarily good to encode as $\{1, 2, 3\}$)

Linearly separable data

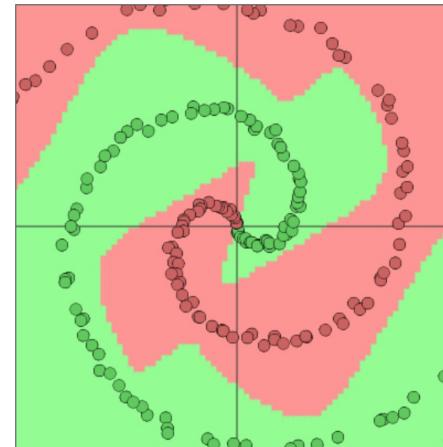
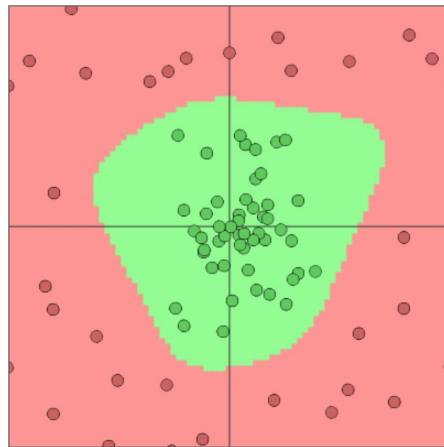
Logistic Regression is trying to find the line that separates data instances where $y = 1$ from those where $y = 0$:

- We call such data (or functions generating that data) linearly separable.
- Naïve Bayes is linear too, because there is one parameter for each feature (and no parameters that involve multiple features).



$$\hat{P}(\mathbf{X}|Y) = \prod_{j=1}^m \hat{P}(X_j|Y)$$

Data is not always 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, Logistic Regression and Naive Bayes still often work well in practice

Extra: Gradient Derivation

Background: Calculus

Calculus refresher

#1:

Derivative(sum) =
sum(derivative)

$$\frac{\partial}{\partial x} \sum_{i=1}^n f_i(x) = \sum_{i=1}^n \frac{\partial f_i(x)}{\partial x}$$

Calculus refresher

#2:

Chain rule 

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

Calculus Chain Rule

$$f(x) = f(z(x))$$

aka decomposition
of composed functions

Our goal

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

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

log conditional likelihood

Two "pre-processing" steps to prepare for chain rule

1. Rewrite $LL(\theta)$ with \hat{y}
2. Compute gradient of \hat{y}

1. Rewriting $LL(\theta)$ with \hat{y}

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

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

log conditional likelihood

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

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

2. Compute gradient of $\hat{y} = \sigma(\theta^T x)$

Aside: Sigmoid has a beautiful derivative!

Sigmoid function:

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

Derivative:

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

2. Compute gradient of $\hat{y} = \sigma(\theta^T \mathbf{x})$

Sigmoid function:

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

Derivative:

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

What is $\frac{\partial}{\partial \theta_j} \hat{y} = \frac{\partial}{\partial \theta_j} \sigma(\theta^T \mathbf{x})$?

- A. $\sigma(x_j)[1 - \sigma(x_j)]x_j$
- B. $\sigma(\theta^T \mathbf{x})[1 - \sigma(\theta^T \mathbf{x})]\mathbf{x}$
- C. $\sigma(\theta^T \mathbf{x})[1 - \sigma(\theta^T \mathbf{x})]x_j$
- D. $\sigma(\theta^T \mathbf{x})x_j[1 - \sigma(\theta^T \mathbf{x})x_j]$
- E. None/other



2. Compute gradient of $\hat{y} = \sigma(\theta^T \mathbf{x})$

Sigmoid function:

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

Derivative:

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

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

- A. $\sigma(x_j)[1 - \sigma(x_j)]x_j$
- B. $\sigma(\theta^T \mathbf{x})[1 - \sigma(\theta^T \mathbf{x})]\mathbf{x}$
- C. $\sigma(\theta^T \mathbf{x})[1 - \sigma(\theta^T \mathbf{x})]x_j$
- D. $\sigma(\theta^T \mathbf{x})x_j[1 - \sigma(\theta^T \mathbf{x})x_j]$
- E. None/other

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

$$\frac{\partial}{\partial \theta_j} \sigma(\theta^T \mathbf{x}) = \frac{\partial}{\partial z} \sigma(z) \cdot \frac{\partial z}{\partial \theta_j} \quad (\text{Chain Rule})$$

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

Compute gradient of log conditional likelihood

$$\begin{aligned}\frac{\partial LL(\theta)}{\partial \theta_j} &= \sum_{i=1}^n \frac{\partial}{\partial \theta_j} [y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})] && \text{Let } \hat{y}^{(i)} = \sigma(\theta^T \mathbf{x}^{(i)}) \\ &= \sum_{i=1}^n \frac{\partial}{\partial \hat{y}^{(i)}} [y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})] \cdot \frac{\partial \hat{y}^{(i)}}{\partial \theta_j} && \text{(Chain Rule)} \\ &= \sum_{i=1}^n \left[y^{(i)} \frac{1}{\hat{y}^{(i)}} - (1 - y^{(i)}) \frac{1}{1 - \hat{y}^{(i)}} \right] \cdot \hat{y}^{(i)} (1 - \hat{y}^{(i)}) x_j^{(i)} && \text{(calculus)} \\ &= \sum_{i=1}^n [y^{(i)} - \hat{y}^{(i)}] x_j^{(i)} && = \sum_{i=1}^n [y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)})] x_j^{(i)} && \text{(simplify)}\end{aligned}$$



Compute gradient of log conditional likelihood

$$\begin{aligned}\frac{\partial LL(\theta)}{\partial \theta_j} &= \sum_{i=1}^n \frac{\partial}{\partial \theta_j} [y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})] && \text{Let } \hat{y}^{(i)} = \sigma(\theta^T \mathbf{x}^{(i)}) \\ &= \sum_{i=1}^n \frac{\partial}{\partial \hat{y}^{(i)}} [y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})] \cdot \frac{\partial \hat{y}^{(i)}}{\partial \theta_j} && \text{(Chain Rule)} \\ &= \sum_{i=1}^n \left[y^{(i)} \frac{1}{\hat{y}^{(i)}} - (1 - y^{(i)}) \frac{1}{1 - \hat{y}^{(i)}} \right] \cdot \hat{y}^{(i)} (1 - \hat{y}^{(i)}) x_j^{(i)} && \text{(calculus)} \\ &= \sum_{i=1}^n [y^{(i)} - \hat{y}^{(i)}] x_j^{(i)} && = \sum_{i=1}^n [y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)})] x_j^{(i)} \quad \begin{array}{c} \text{party hat emoji} \\ \text{(simplify)} \end{array}\end{aligned}$$