25: Logistic Regression

David Varodayan
March 6, 2020
Adapted from slides by Lisa Yan

Announcements

Updated CS109 logistics and policies

We are implementing several changes to CS109 logistics and policies in response to growing concern about Covid-19. These changes affect lectures, section, office hours and the final exam.

Read the new policies in the announcements on the course webpage: http://web.stanford.edu/class/cs109/

We welcome your questions on piazza

Announcements

Problem Set 6

Due: Wednesday 3/11
Covers: Up to Lecture 25
Extra Python Office Hours: Saturday 3/7, 3-5PM

Regrades

Pset 1 to 5 and Midterm regrades to close on 3/11 at 1pm

Autograded Coding Problems

Run your code in the command line or install Pycharm following directions on Pset 6 webpage

Late Day Reminder

No late days permitted past last day of the quarter, 3/13

Today’s plan

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

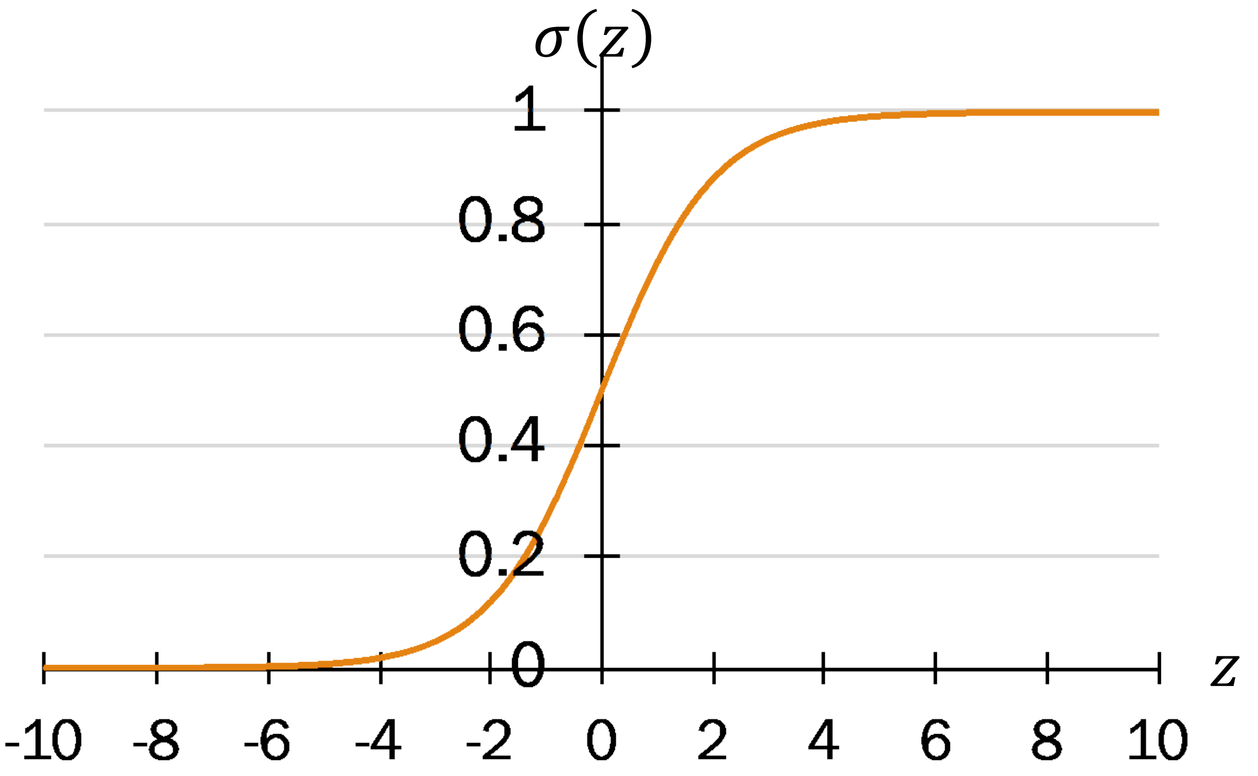
Background: Sigmoid function $\sigma(z)$

- 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

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

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

Calculus
Chain Rule

Today's plan

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

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

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

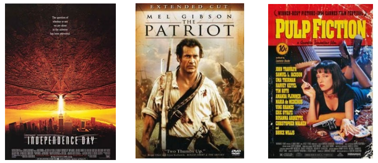
Logistic Regression Model

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

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

Logistic Regression

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



$\boldsymbol{X}$

input features

θ parameter

0.81

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

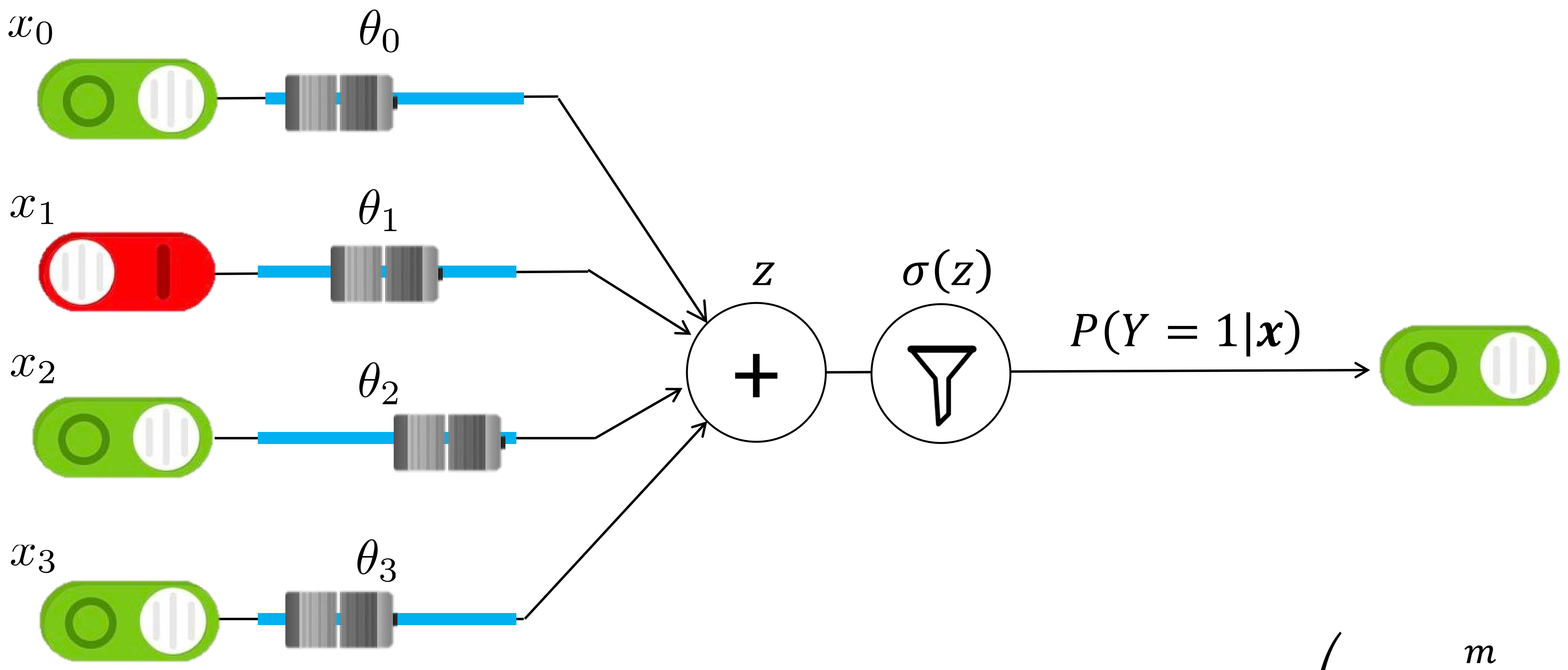
conditional likelihood

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

Logistic Regression Cartoon

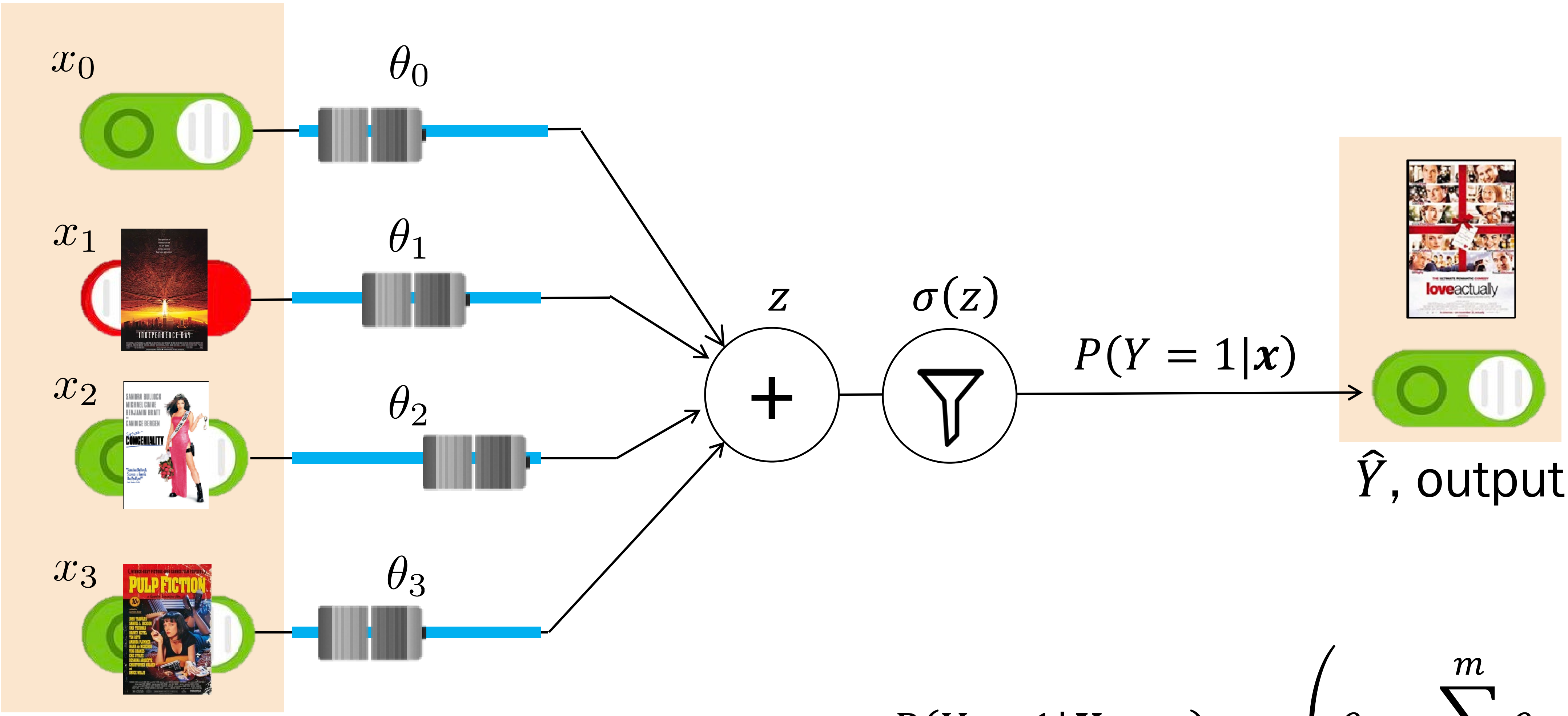
θ parameter

Logistic Regression cartoon



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

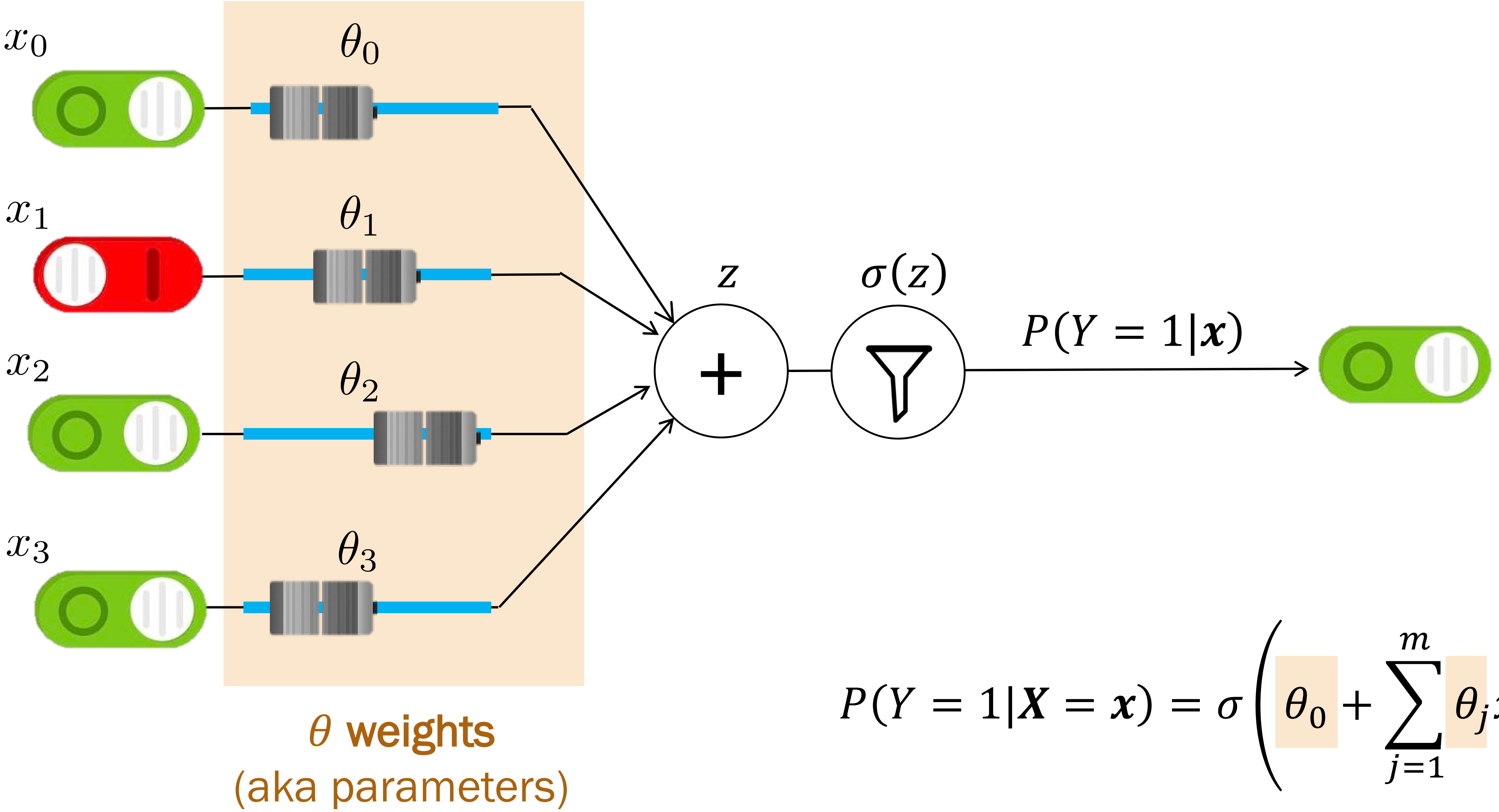
Logistic Regression input/output



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

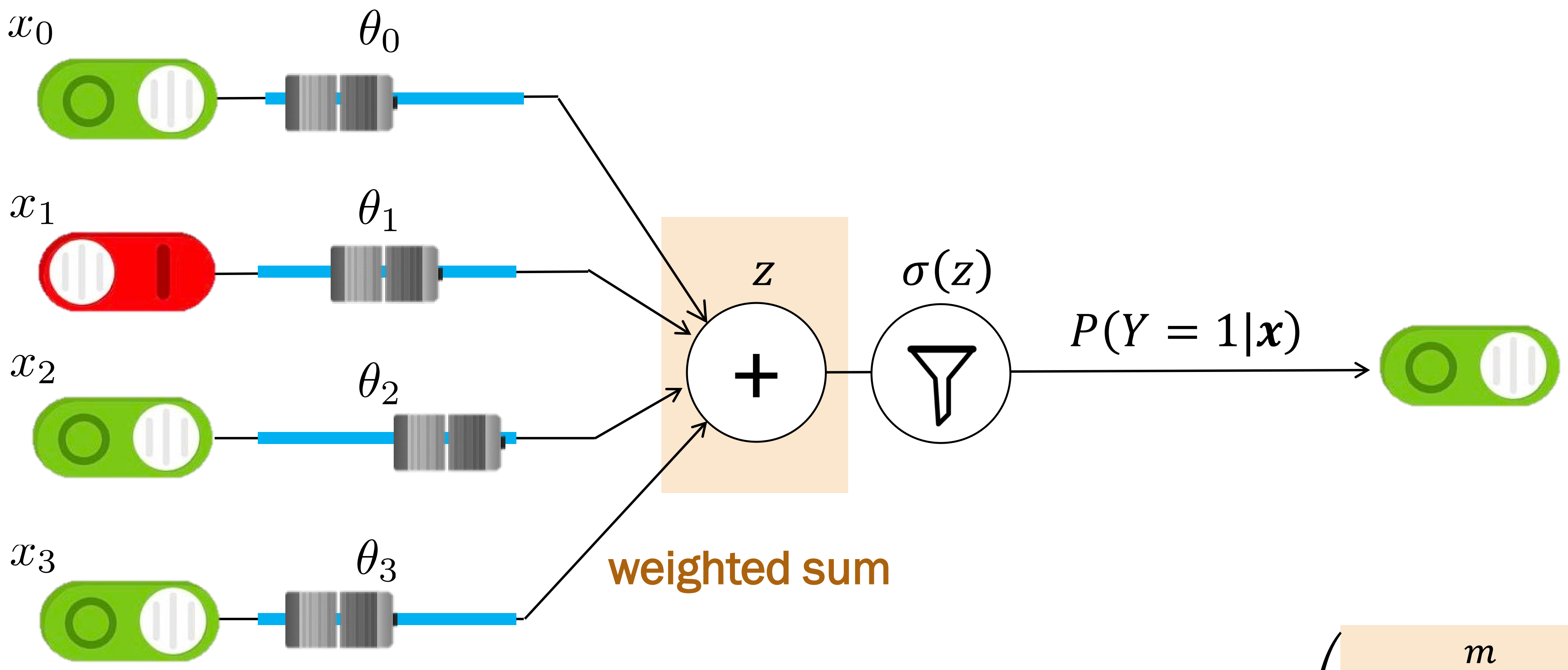
$$P(Y=1|\boldsymbol{X}=\boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

Components of Logistic Regression



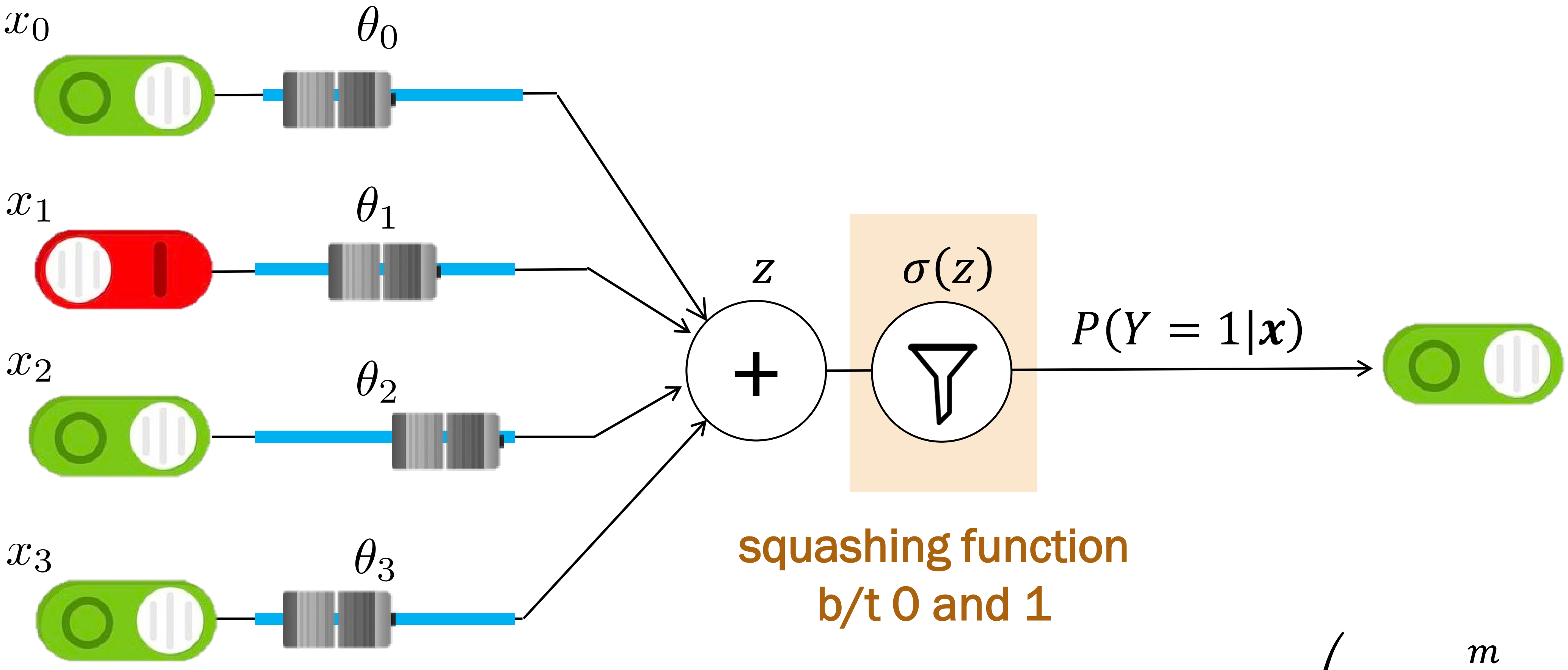
$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

Components of Logistic Regression



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

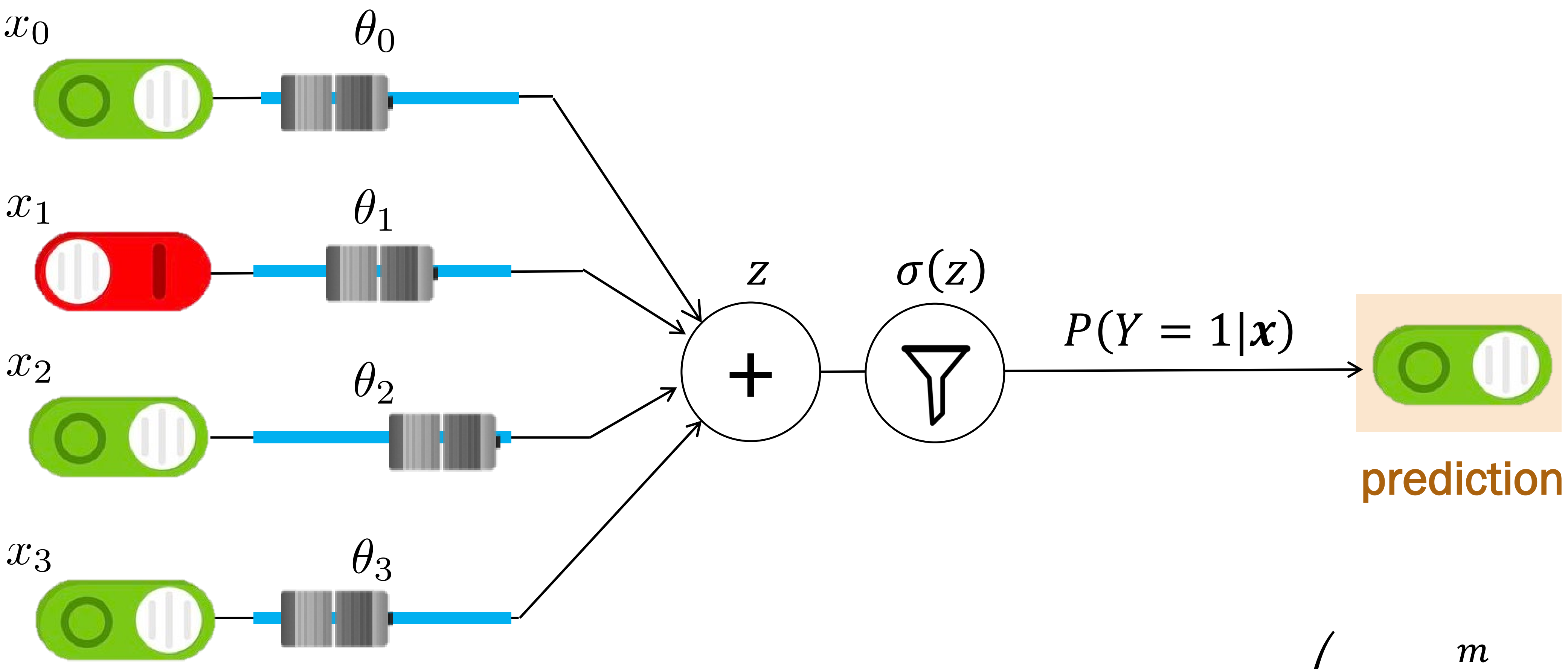
Components of Logistic Regression



squashing function
b/t 0 and 1

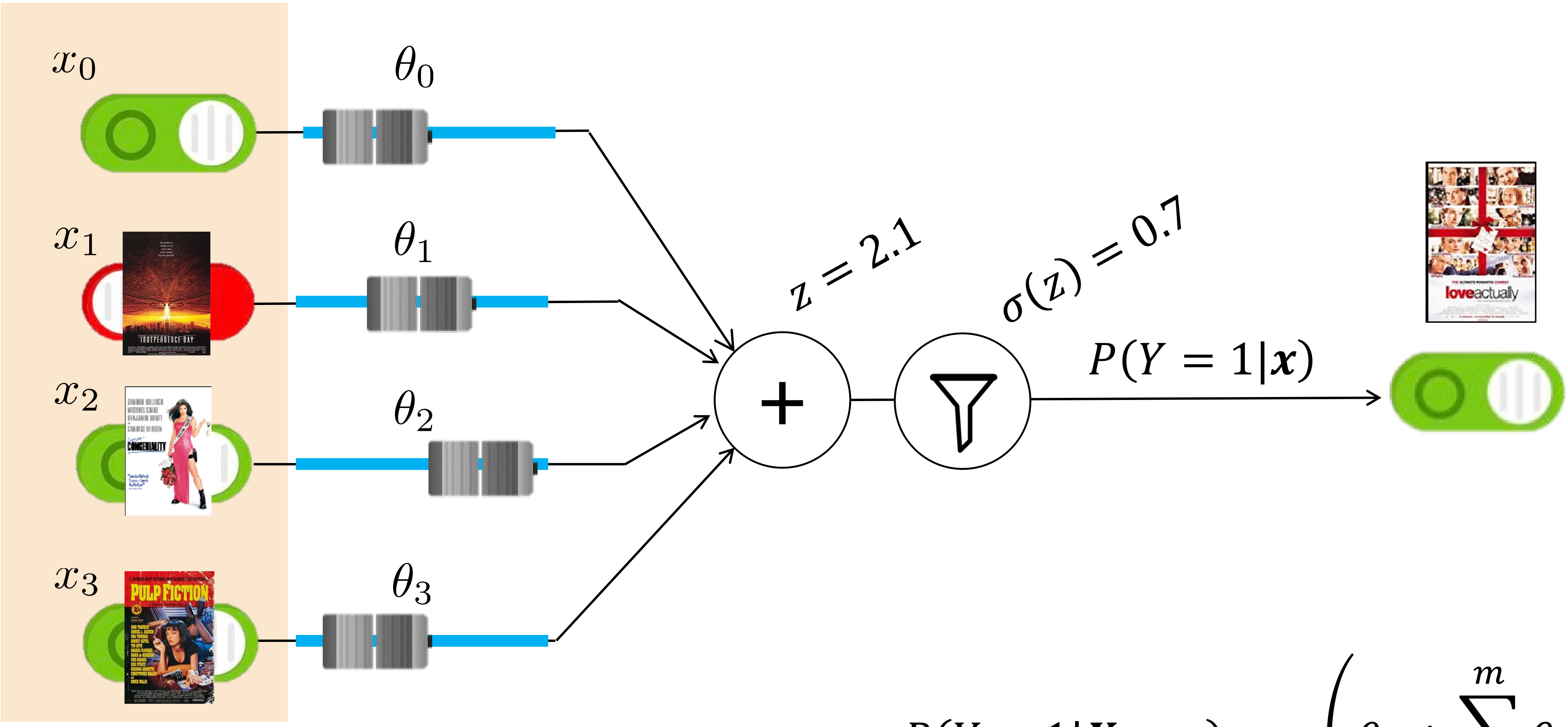
$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

Components of Logistic Regression



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

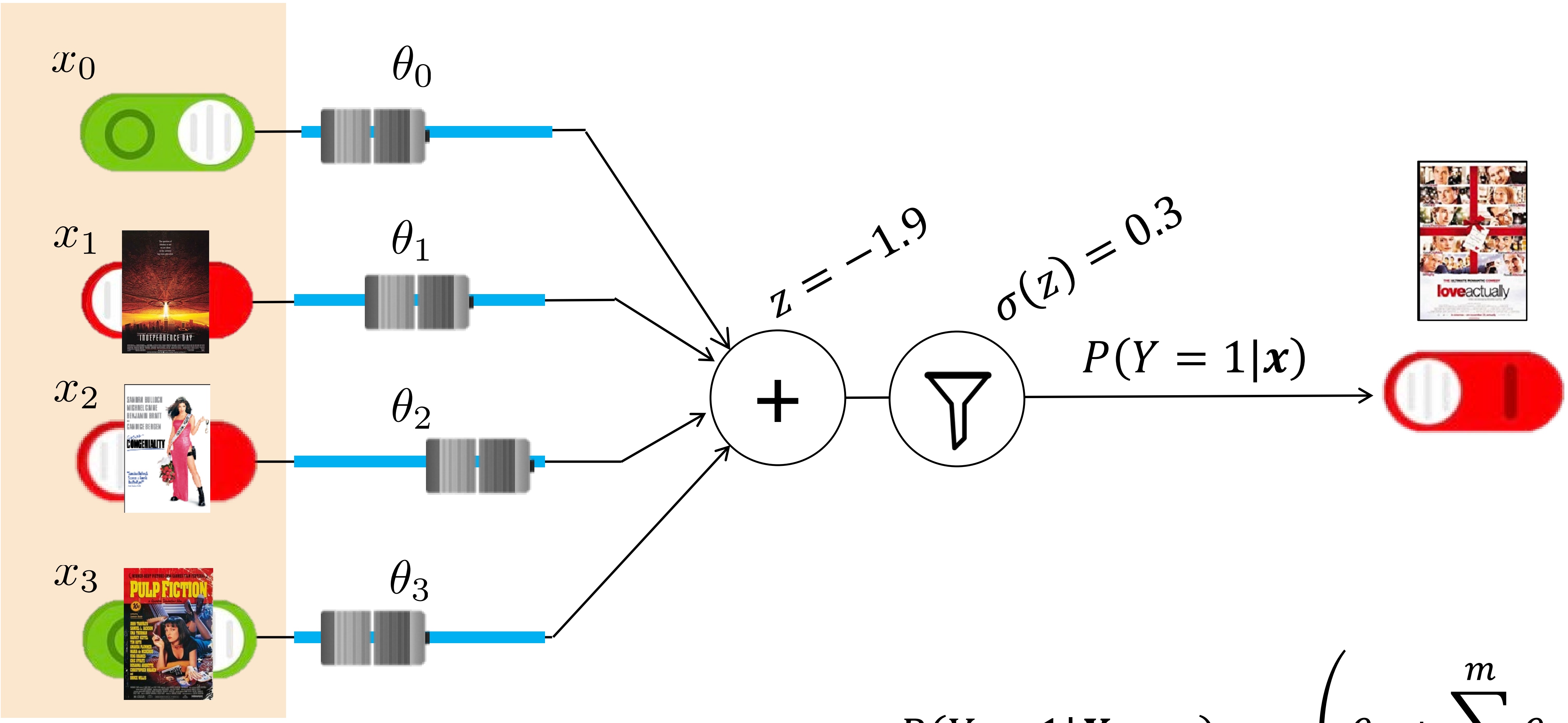
Different predictions for different inputs



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

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

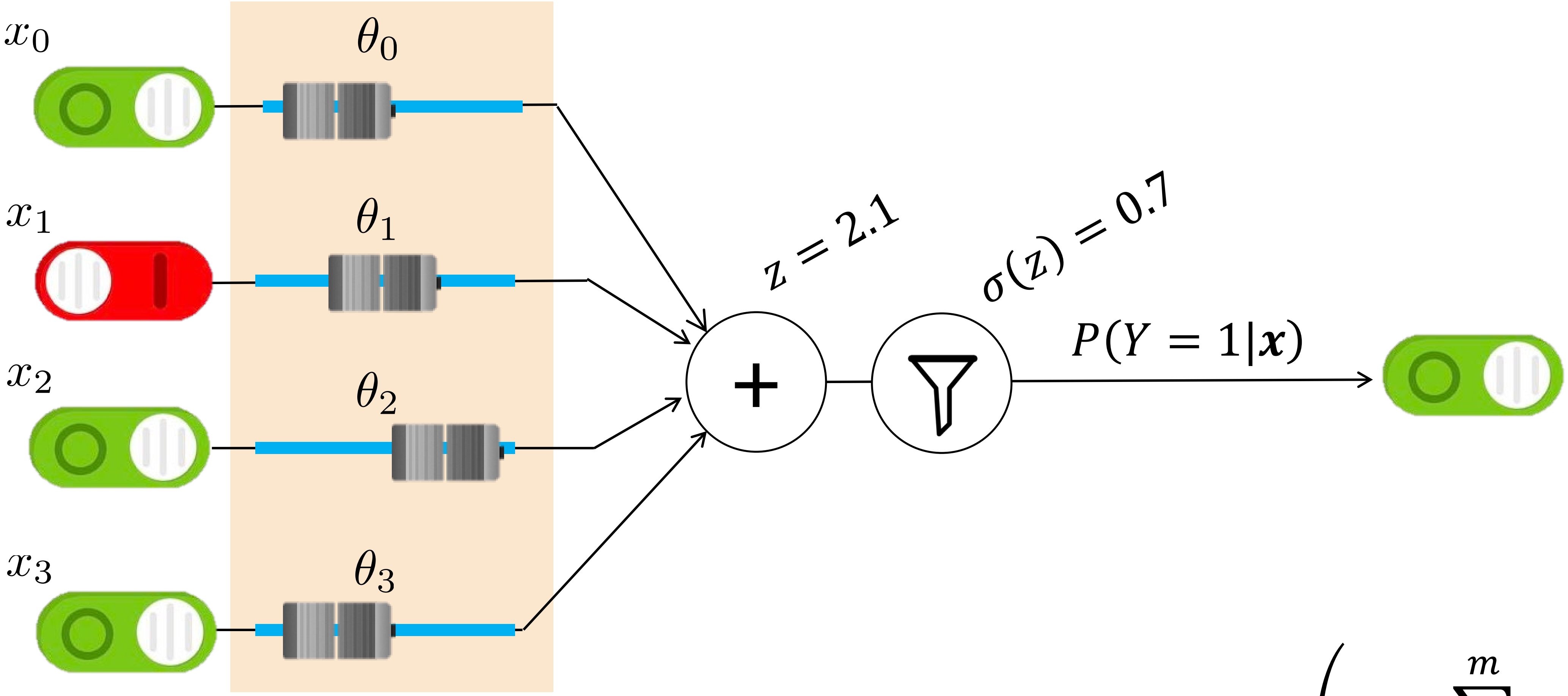
Different predictions for different inputs



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

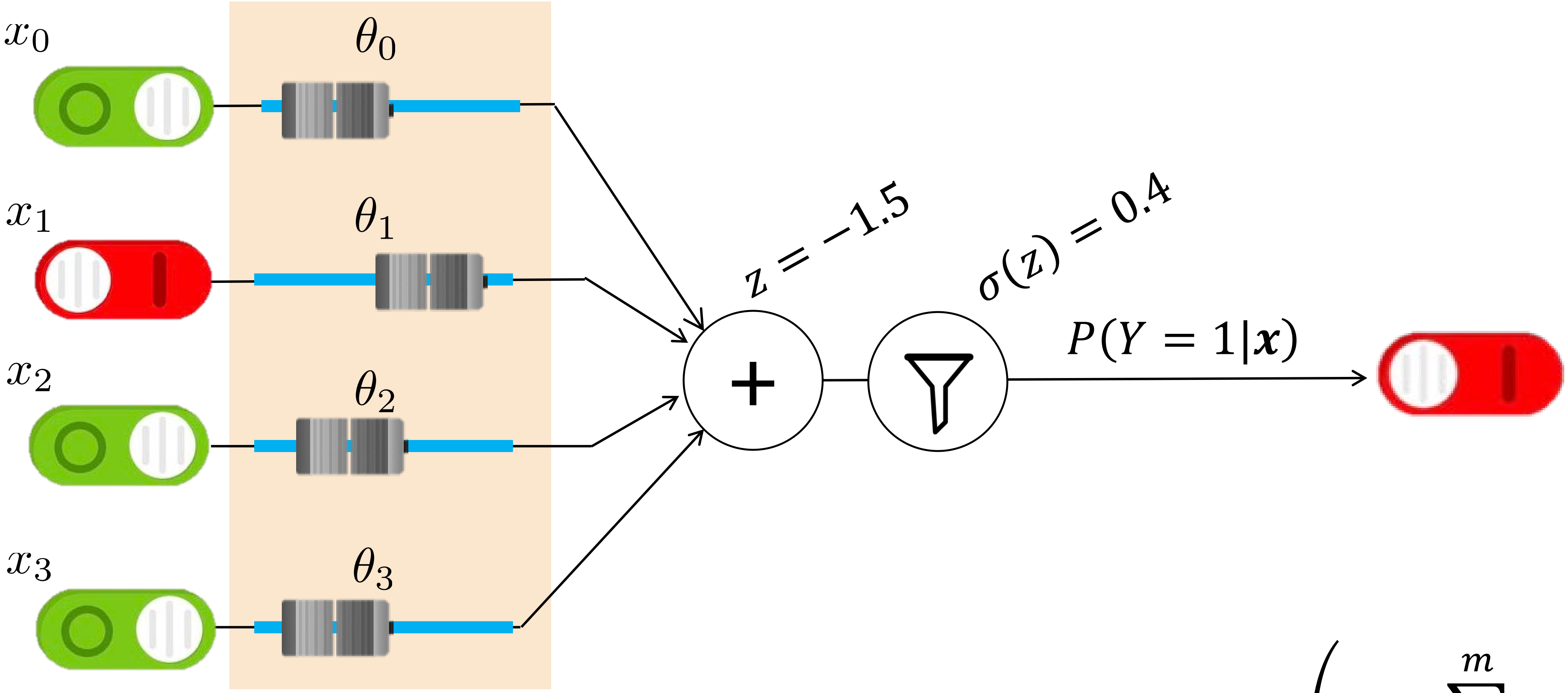
$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

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

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

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

where

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

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

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

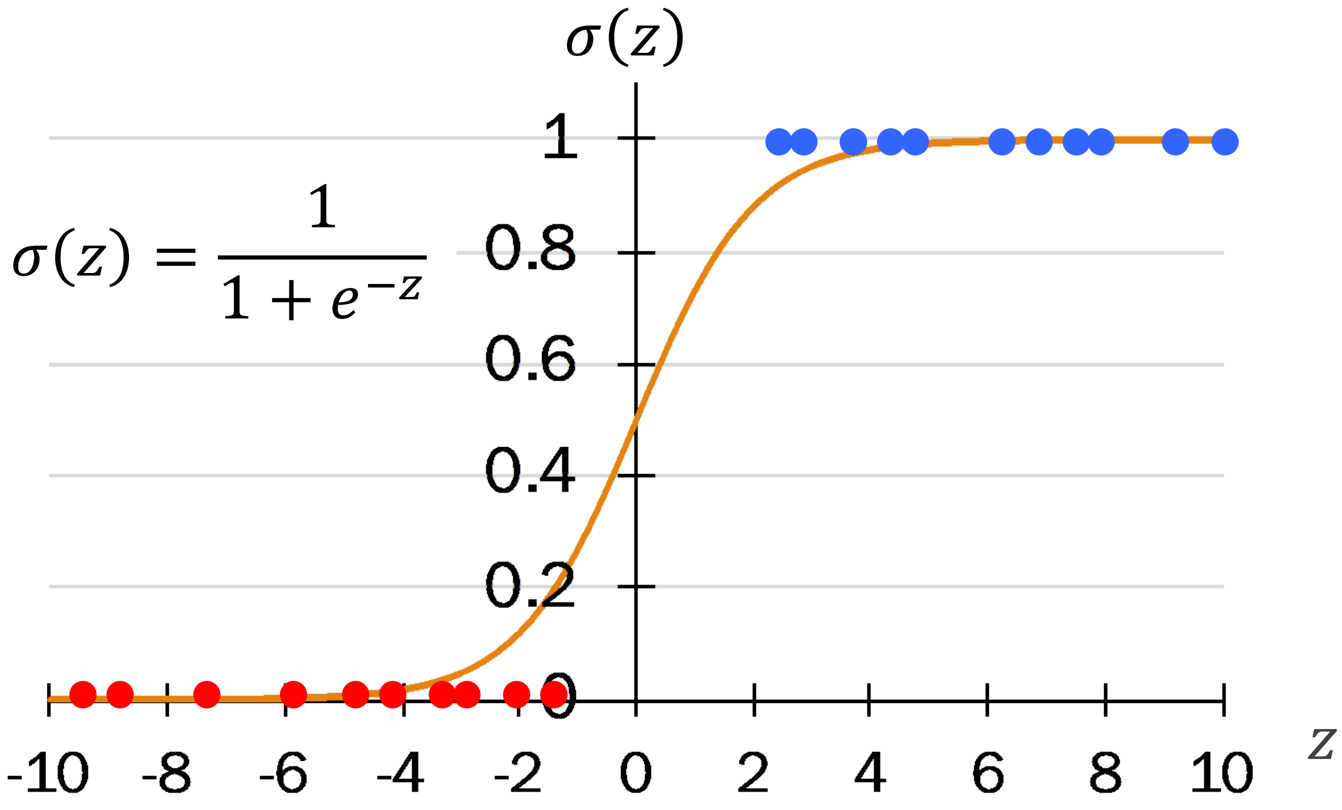
$$P(Y = 0|\boldsymbol{X} = \boldsymbol{x}) = 1 - \sigma(\theta^T \boldsymbol{x})$$

Classifying using the sigmoid function

Logistic Regression Model

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

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

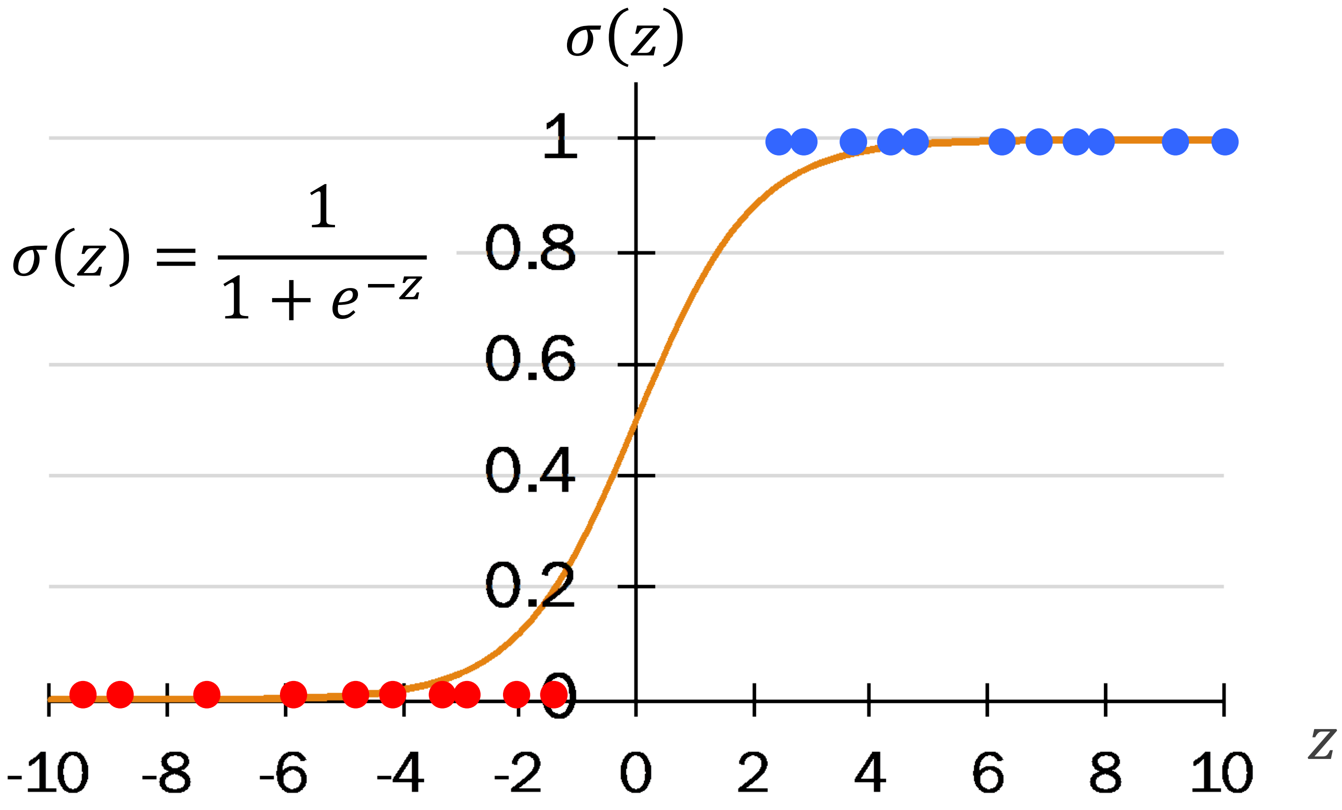


Logistic Regression uses the sigmoid function to try and distinguish $y = 1$ (blue) points from $y = 0$ (red) points.

Classifying using the sigmoid function

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

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



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

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

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

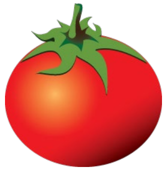
C. If $\theta^T \boldsymbol{x} > 0$

D. All are valid, but C is easiest

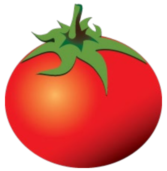
E. None/Other

Naming algorithms

Regression Algorithms

Linear Regression 

Classification Algorithms

Naïve Bayes 

Logistic Regression 

Awesome classifier,
terrible name

Training: Learning the parameters

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

- Logistic Regression Model:

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

- Want to predict training data as correctly as possible:

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

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

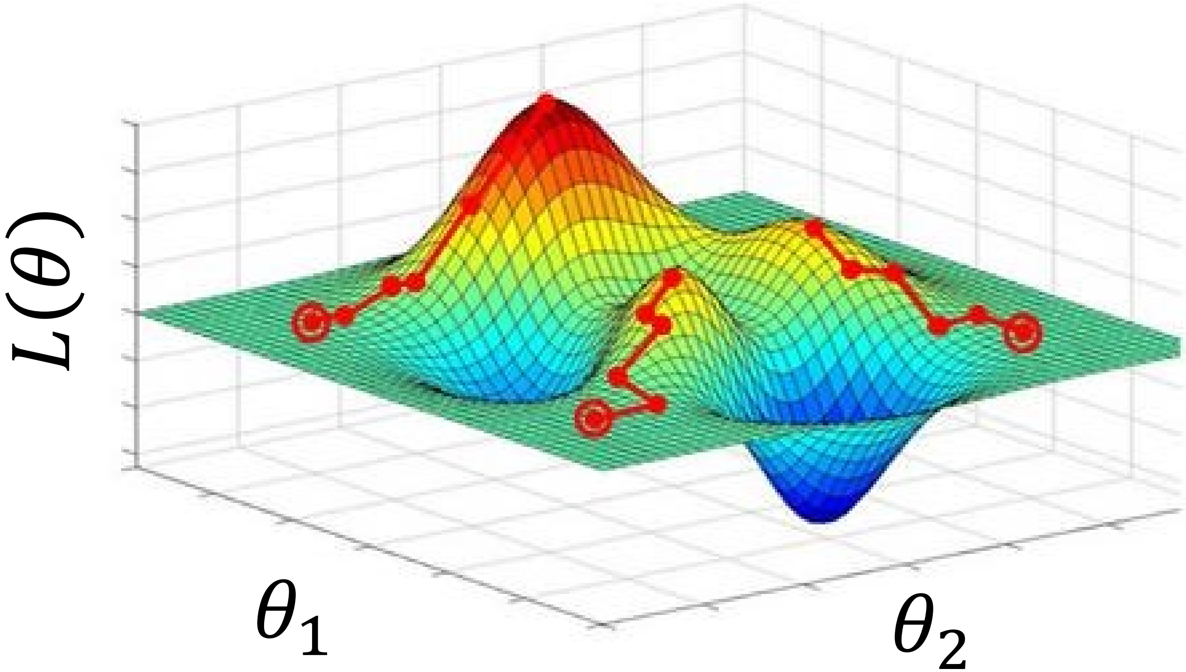
3. Compute derivative of log-likelihood with respect to each $\theta_j, j = 0, 1, \ldots, m$:

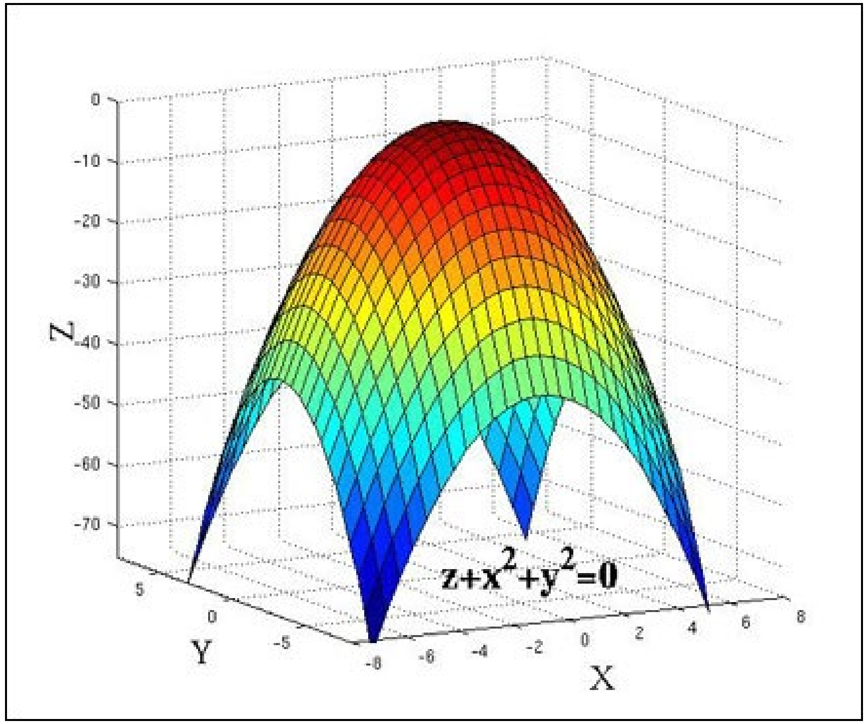
$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right)\right] x_j^{(i)}$$

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.

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

Repeat many times:

For all thetas:

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

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

What does this look like in code?

Training: Gradient Ascent

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

initialize θ_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

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

θ_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 θ_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] += $\left[y - \frac{1}{1 + e^{-\theta^T \boldsymbol{x}}} \right] \boxed{x_j}$

θ_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 θ_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] += $\left[y - \frac{1}{1 + \boxed{e^{-\theta^T \boldsymbol{x}}}} \right] x_j$

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

initialize θ_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] += $\left[y - \frac{1}{1 + e^{-\theta^T \boldsymbol{x}}} \right] x_j$

θ_j += η * 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

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

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

    θ_j += η * 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

Testing: Classification with Logistic Regression

Training

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

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 <u>not</u> updated during testing phase

Today's plan

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:

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

How did we get this log-likelihood function?

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

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

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:

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

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

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 $\theta_j, j = 0, 1, \ldots, m$:

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

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

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)

Today's plan

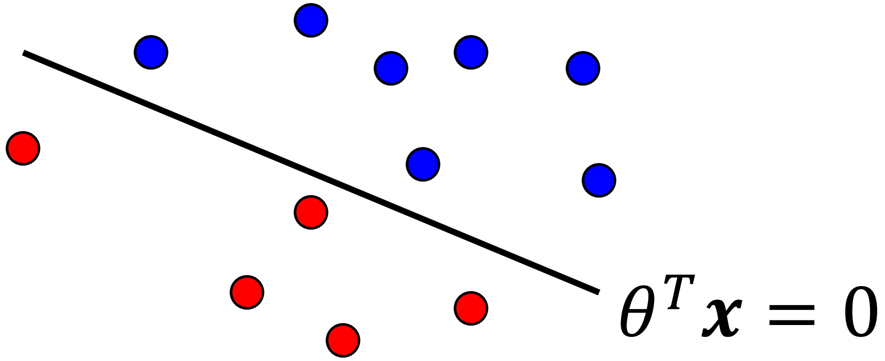
Logistic Regression

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

Intuition about Logistic Regression

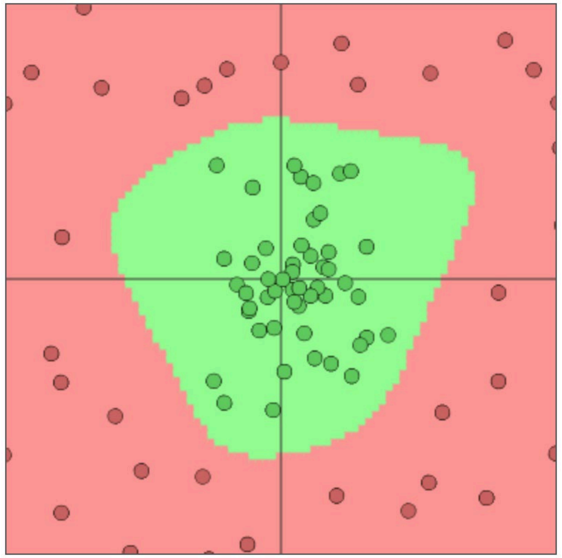
Logistic Regression Model $P(Y=1|\boldsymbol{X}=\boldsymbol{x})=\sigma(\theta^T\boldsymbol{x})$ where $\theta^T\boldsymbol{x}=\sum_{j=0}^{m}\theta_j x_j$

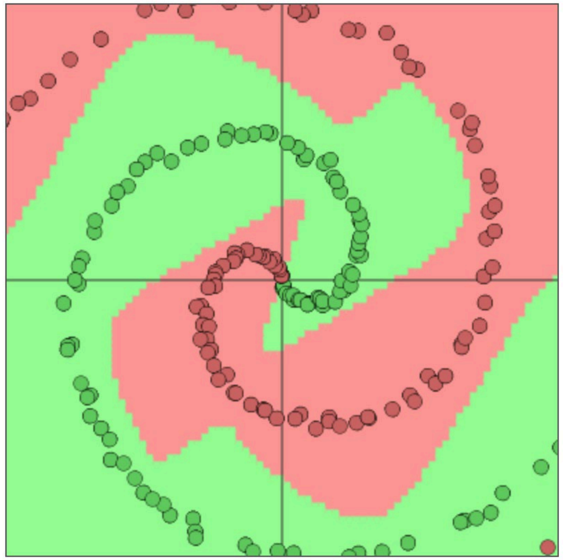
Logistic Regression is trying to fit a **line** that separates data instances where $y=1$ from those where $y=0$:



- We call such data (or functions generating the data **linearly separable**.
- 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	$P(\boldsymbol{X}, Y)$	$P(Y \mid \boldsymbol{X})$
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 $P(X_i \mid Y)$	Multi-valued discrete data hard (e.g., if $X_i \in \{A, B, C\}$, not necessarily good to encode as $\{1, 2, 3\}$