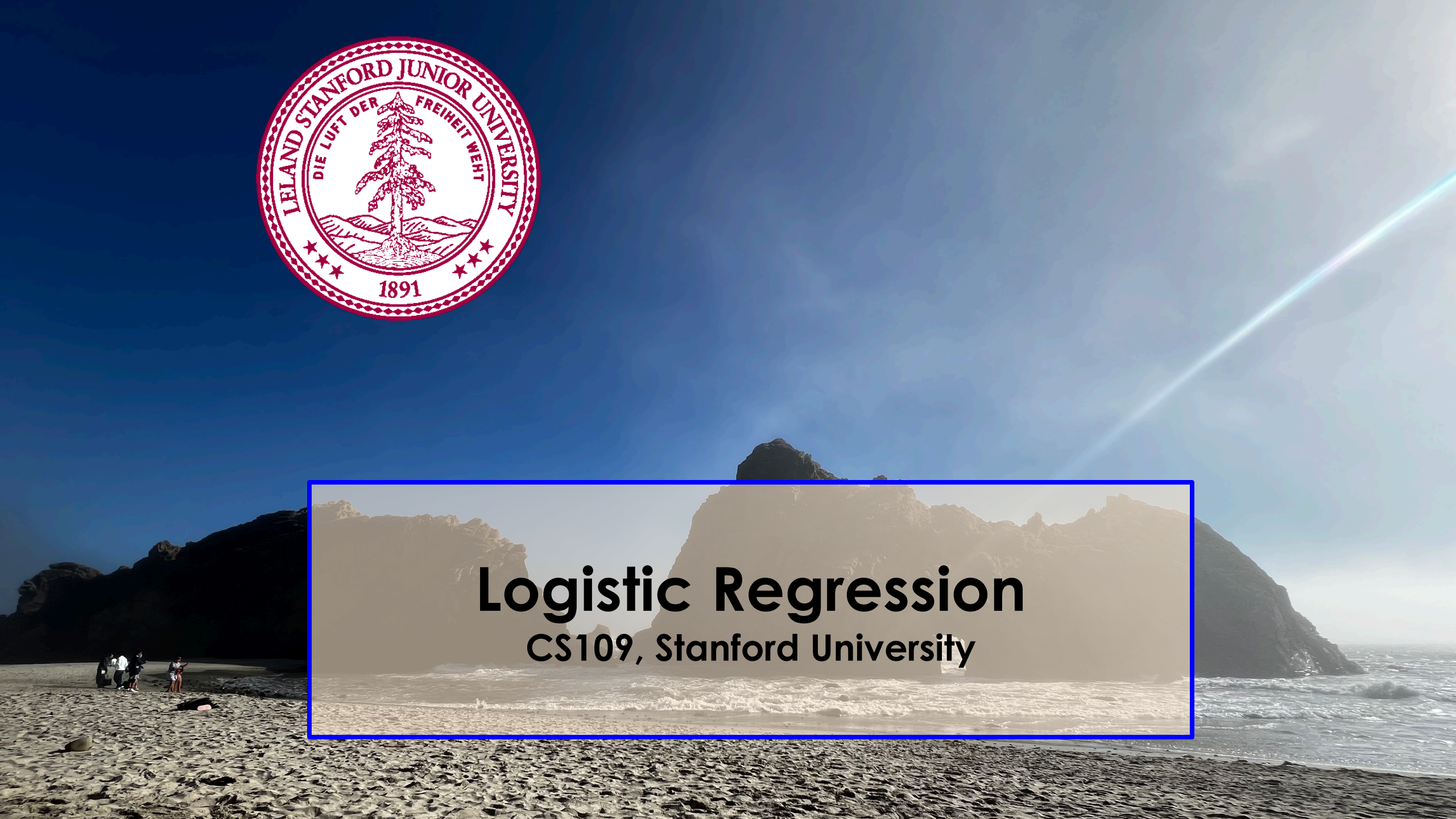
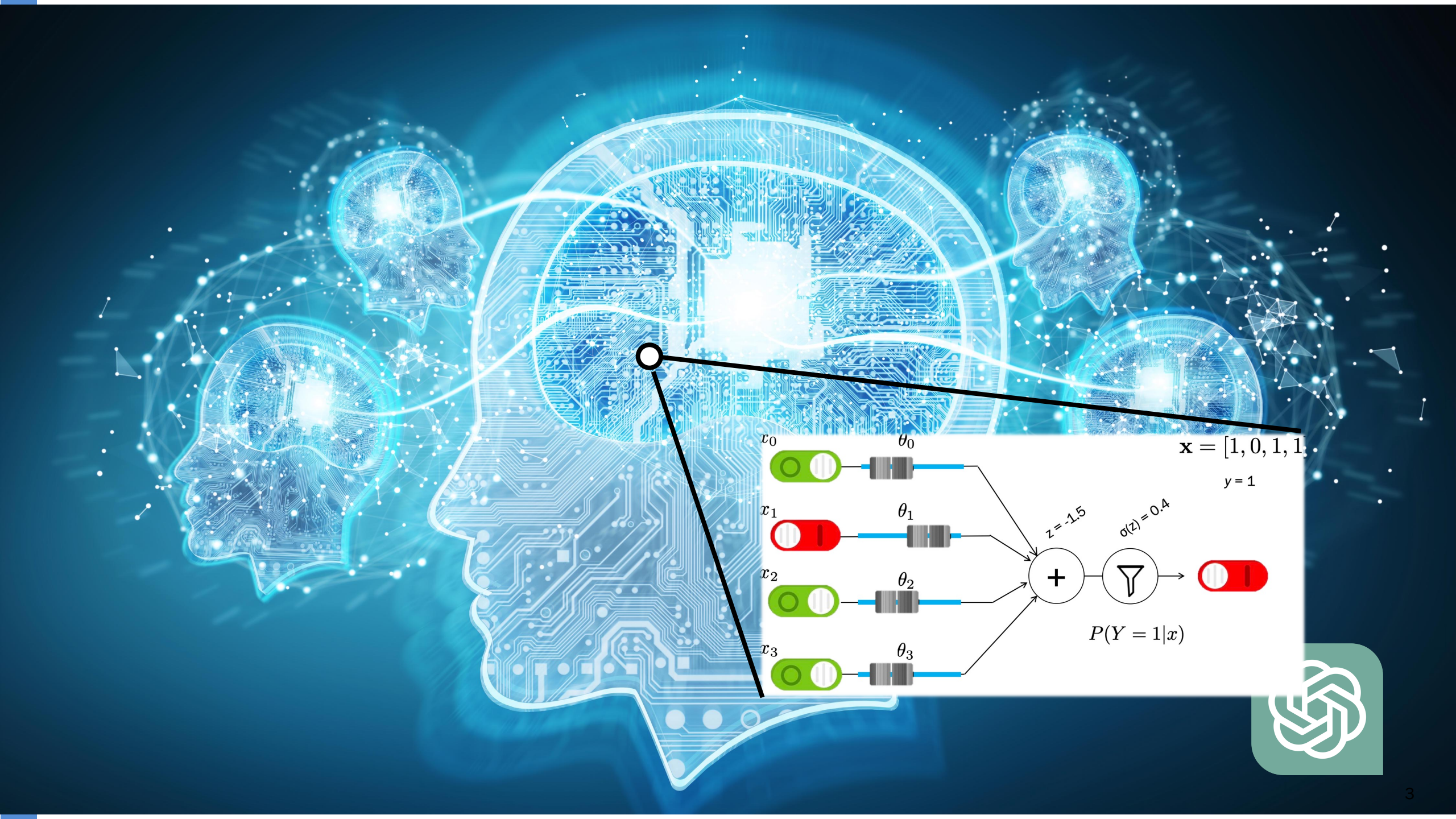


Logistic Regression

CS109, Stanford University

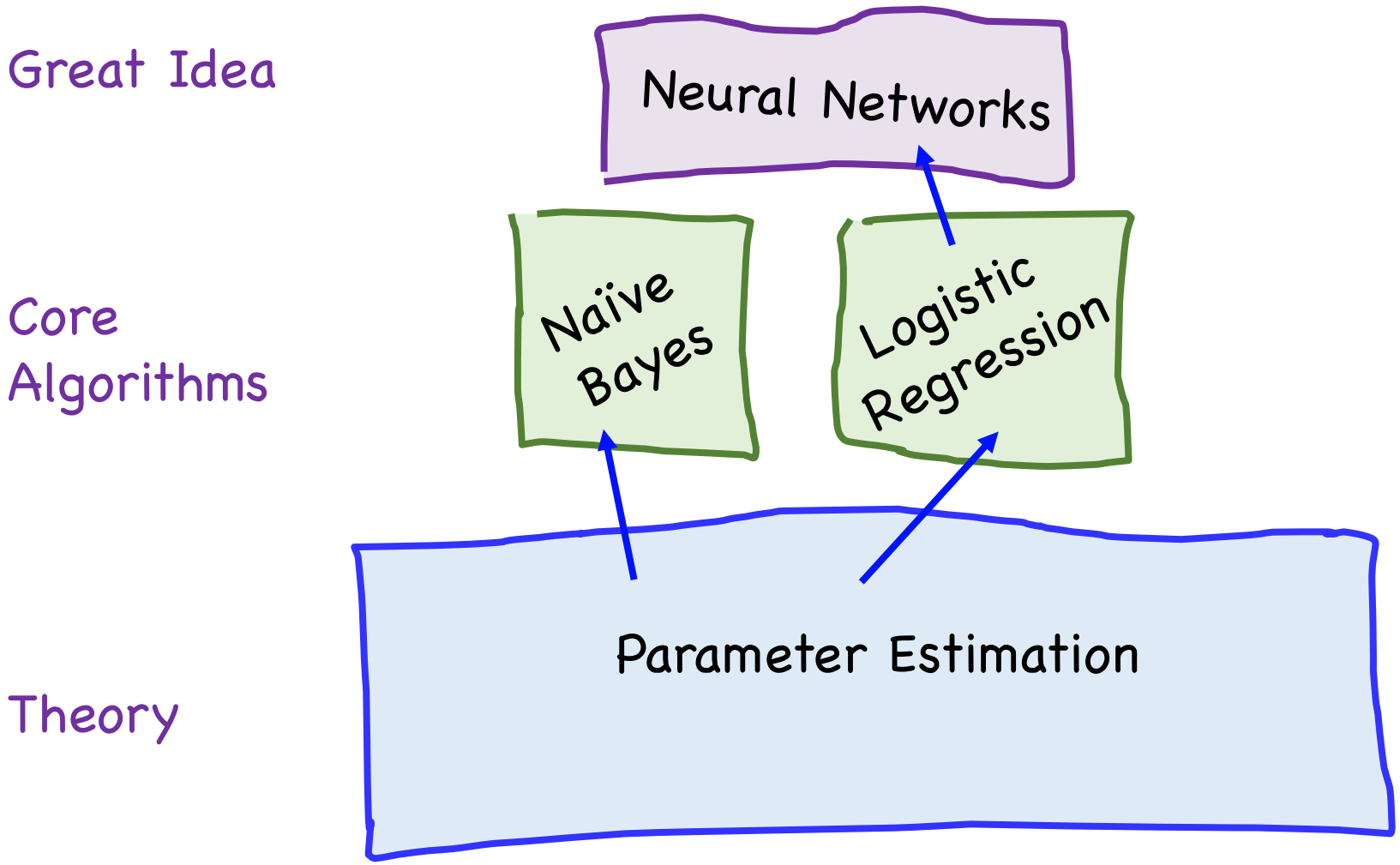

x_0
θ_0
$\mathbf{x} = [1, 0, 1, 1$
$y = 1$
x_1
θ_1
$z = -1.5$
$\sigma(z) = 0.4$
x_2
θ_2
x_3
θ_3
$P(Y = 1|x)$

Creative Challenge

Due this Wed at Midnight!

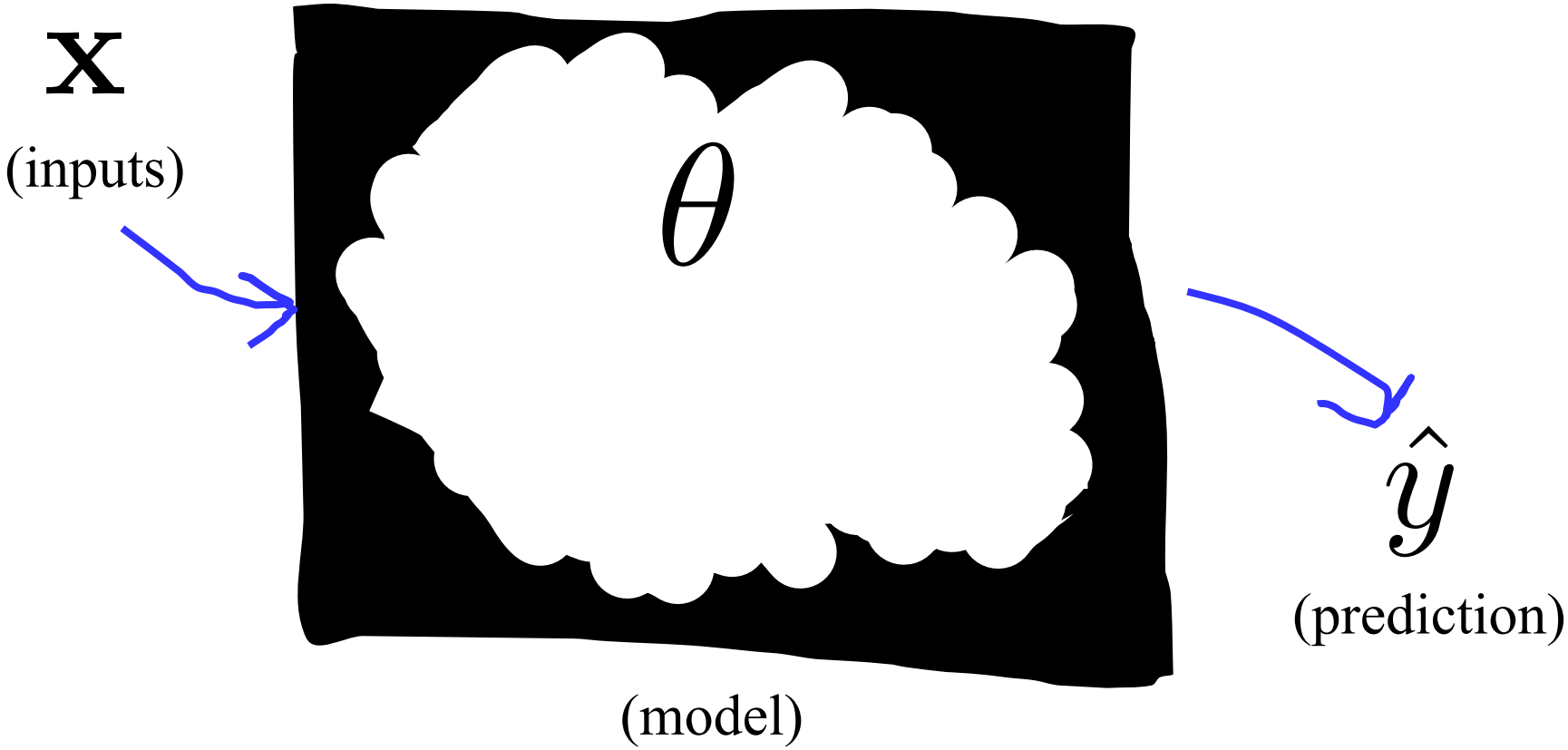
Review

Machine Learning in CS109

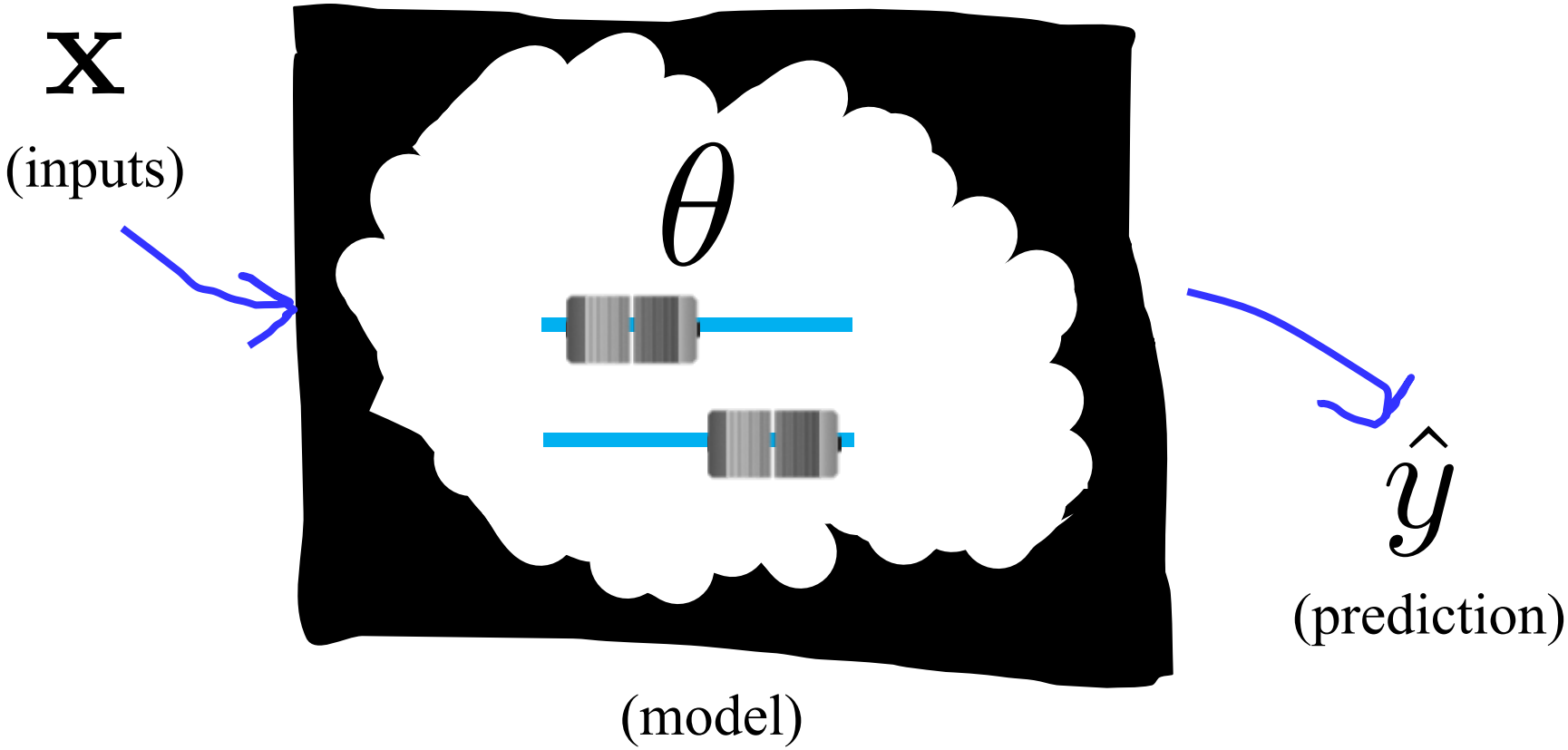


Machine Learning
(aka Applied Probability)

Machine Learning



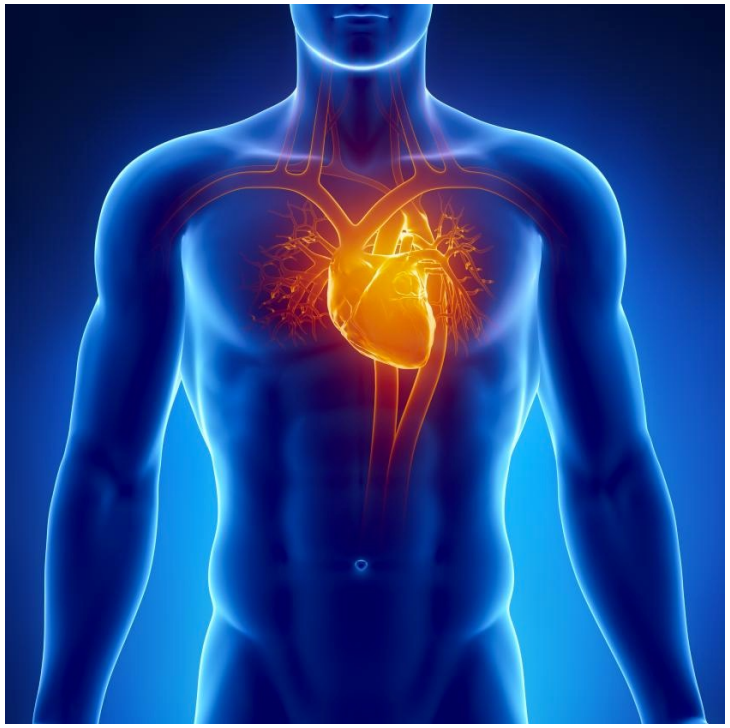
Machine Learning



Classification

Classification Task

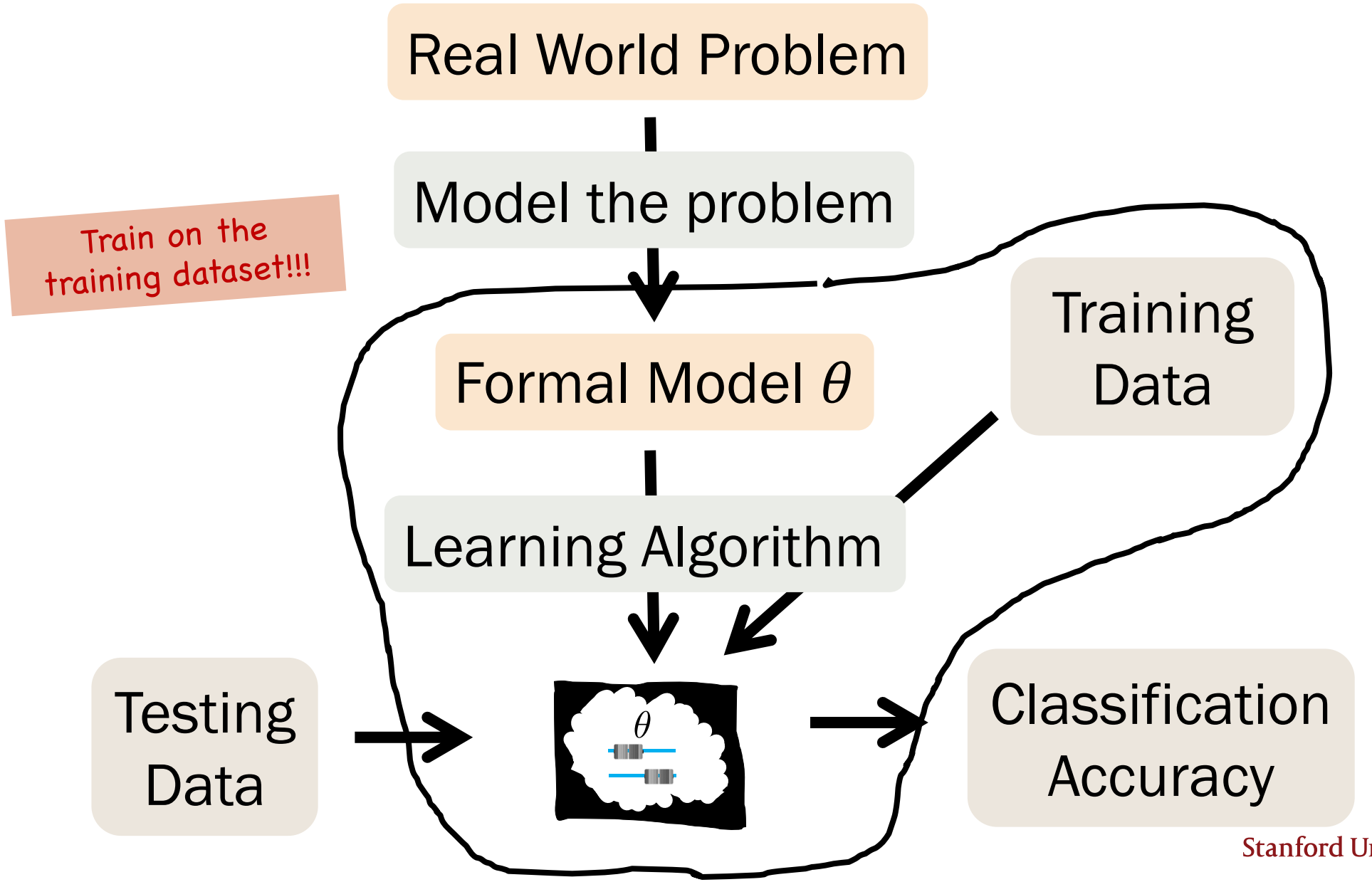
Heart



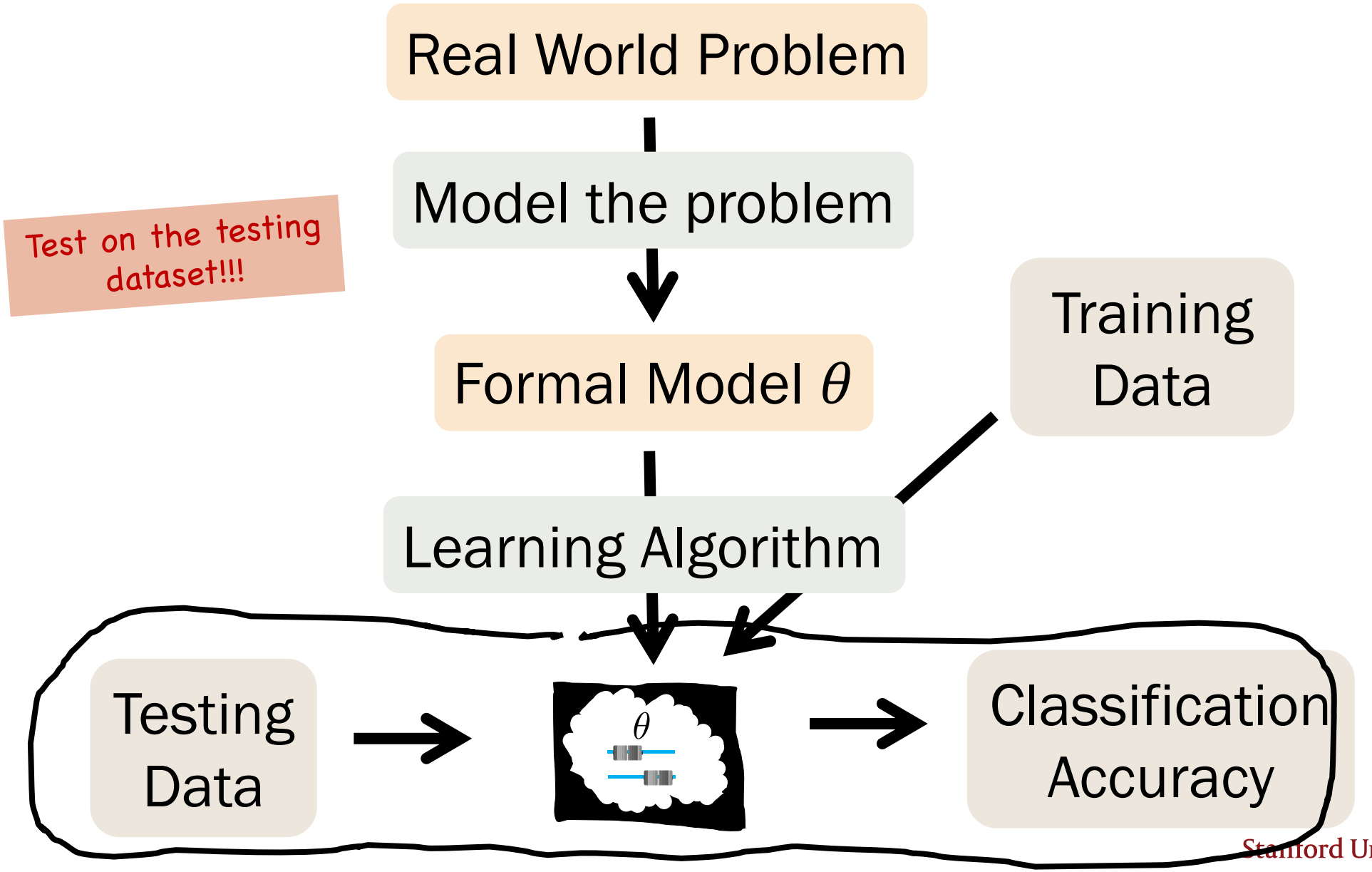
Ancestry

Netflix

Training


Real World Problem
Model the problem
Train on the training dataset!!!
Formal Model θ
Training Data
Learning Algorithm
Testing Data
θ
Classification Accuracy

Testing


Real World Problem
Model the problem
Test on the testing dataset!!!
Formal Model θ
Training Data
Learning Algorithm
Testing Data
θ
Classification Accuracy

Training Data

Assume IID data:

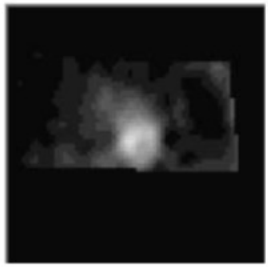
n training datapoints

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots (\mathbf{x}^{(n)}, y^{(n)})$$

$$m = |\mathbf{x}^{(i)}|$$

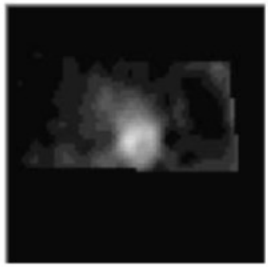
Each datapoint has m features and a single output

Healthy Heart Classifier

	ROI 1	ROI 2	...	ROI *m*	Output
					
Heart 1	0	1		1	0
Heart 2	1	1		1	0
	⋮				⋮
Heart *n*	0	0		0	1

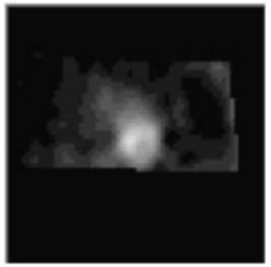
$$(\mathbf{x}^{(2)}, y^{(2)})$$

Healthy Heart Classifier

	ROI 1	ROI 2	...	ROI *m*	Output
					
Heart 1	0	1		1	0
Heart 2	1	1		1	0
	⋮				⋮
Heart *n*	0	0		0	1

$\mathbf{x}^{(2)}$

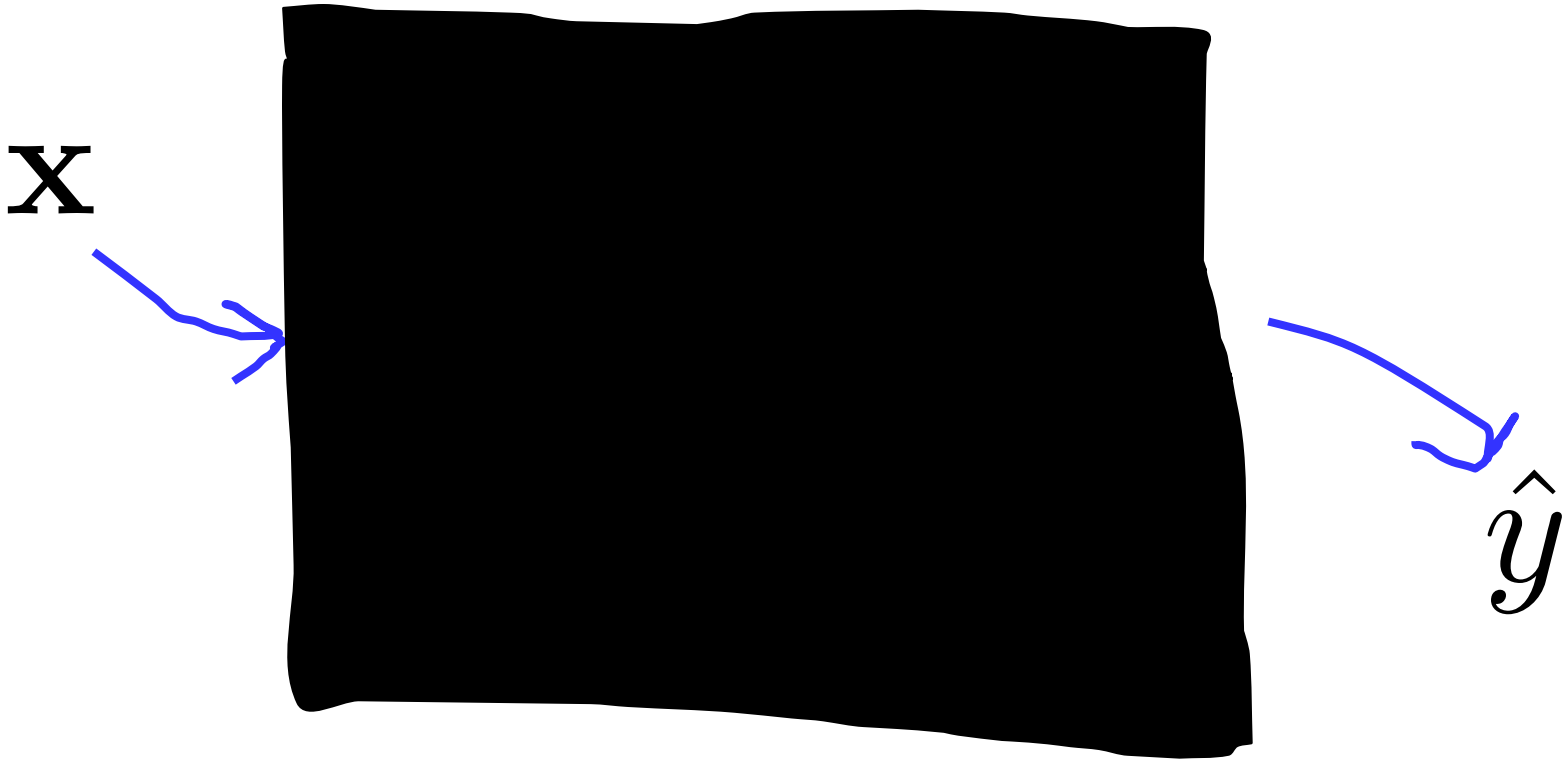
Healthy Heart Classifier

	ROI 1	ROI 2	...	ROI *m*	Output
					
Heart 1	0	1		1	0
Heart 2	1	1		1	0
			⋮		⋮
Heart *n*	0	0		0	1

$y^{(2)}$

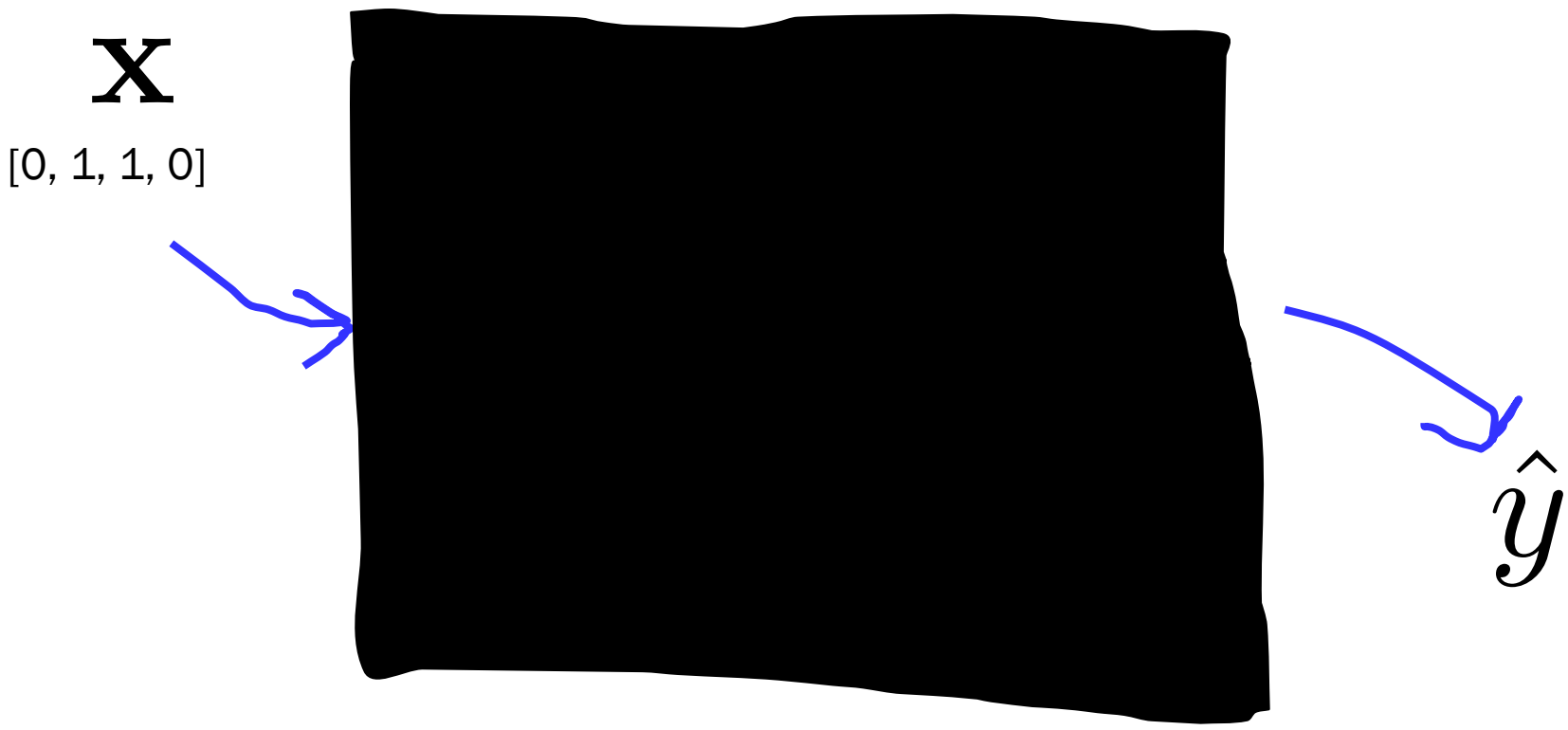
Naïve Bayes Classification

$g_\theta(\mathbf{x})$?



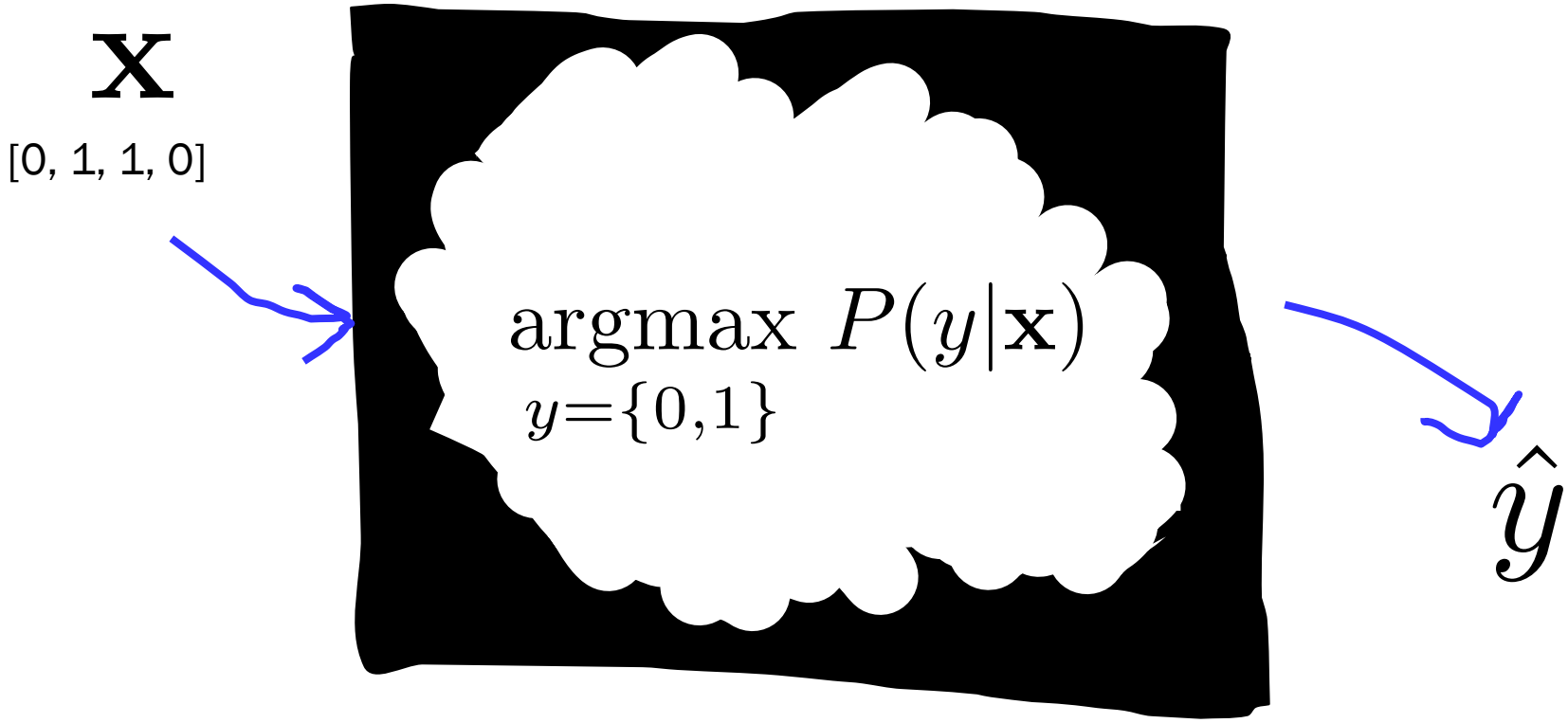
Making a prediction…

$g_\theta(\mathbf{x})$?



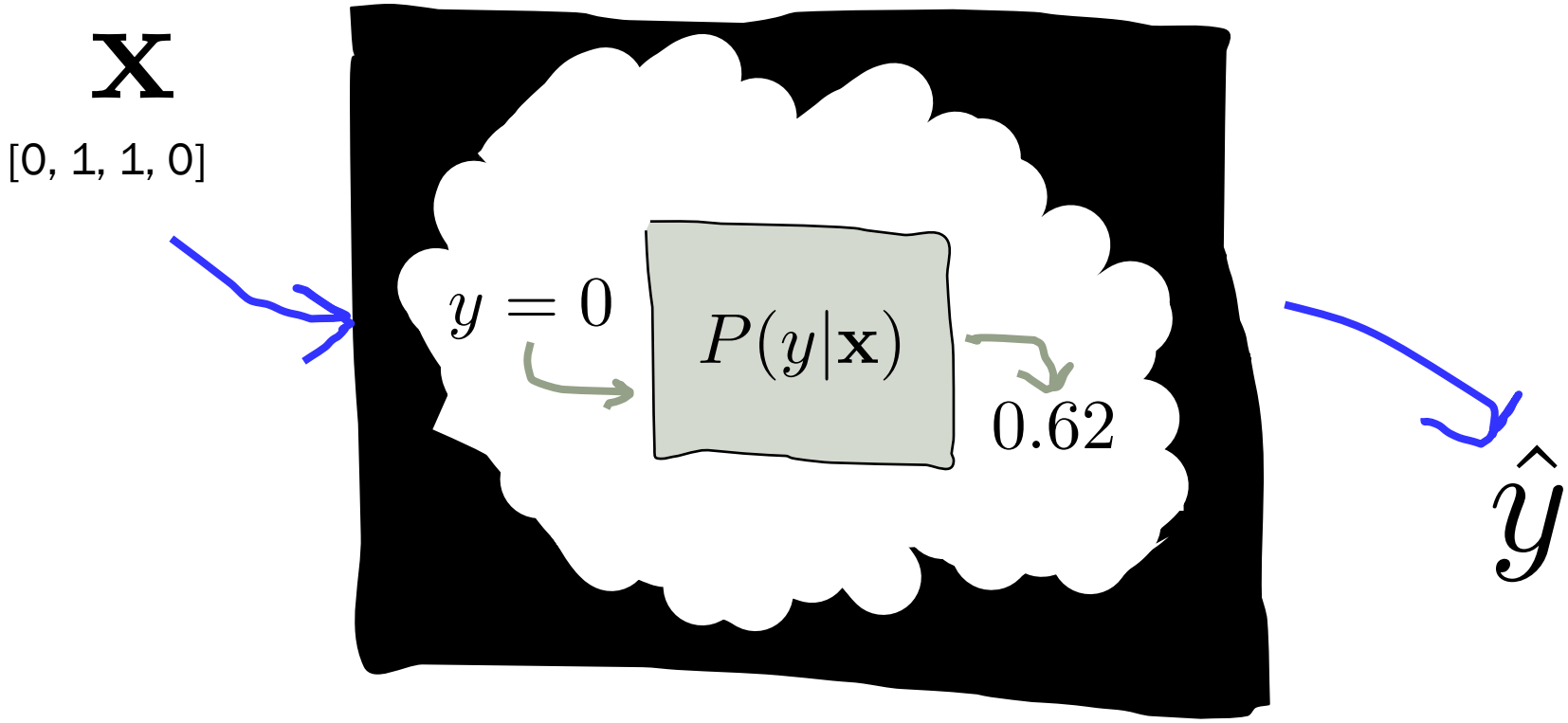
Making a prediction…

$g_\theta(\mathbf{x})$?



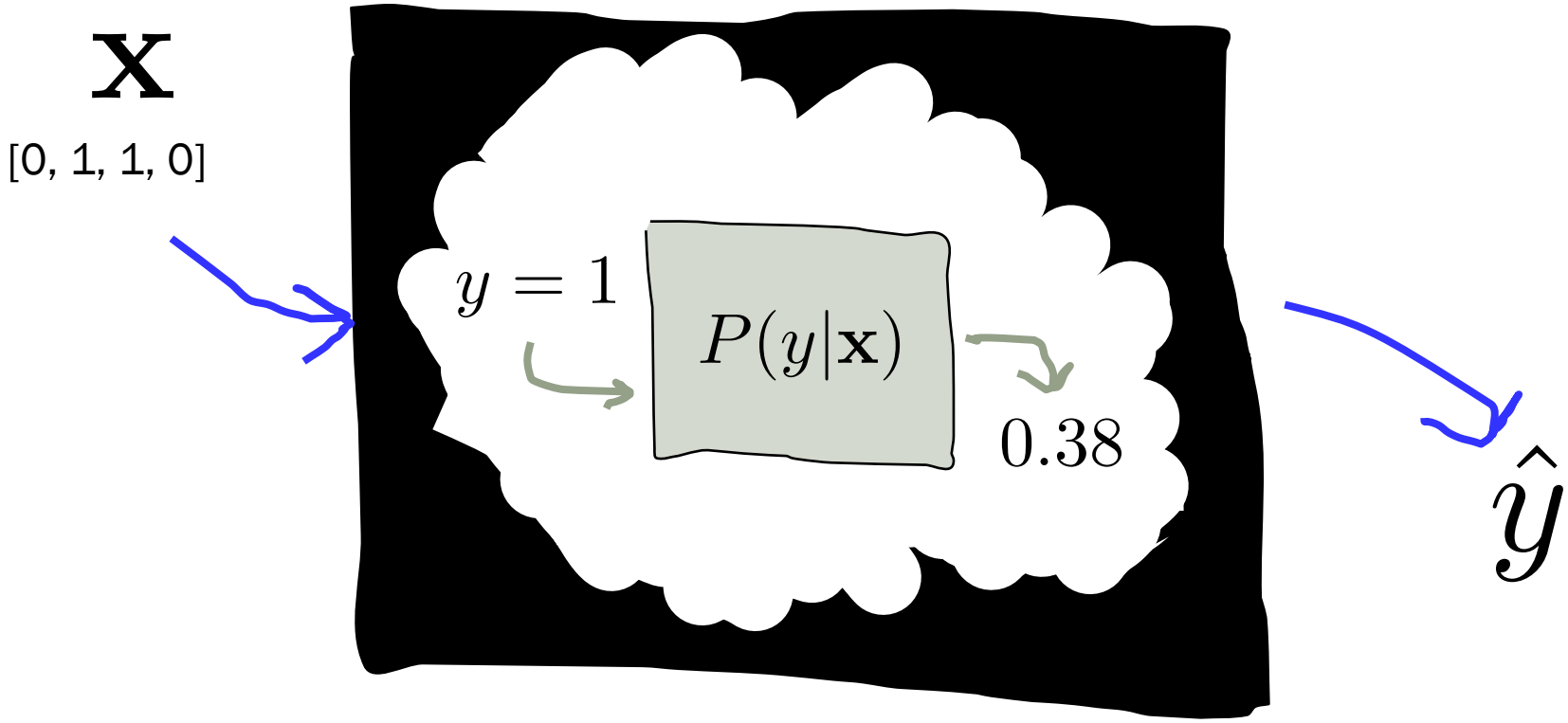
Making a prediction…

$g_\theta(\mathbf{x})$?



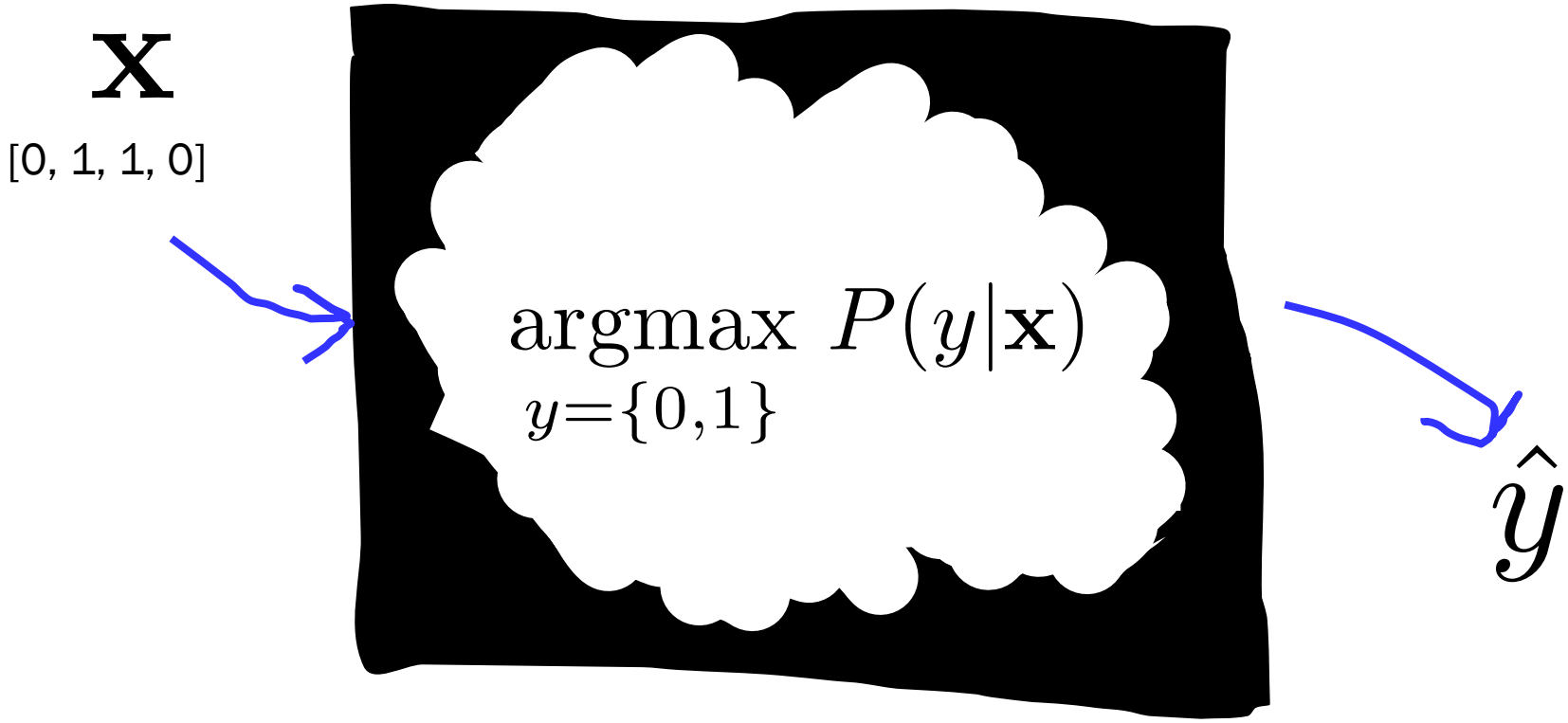
Making a prediction…

$g_\theta(\mathbf{x})$?



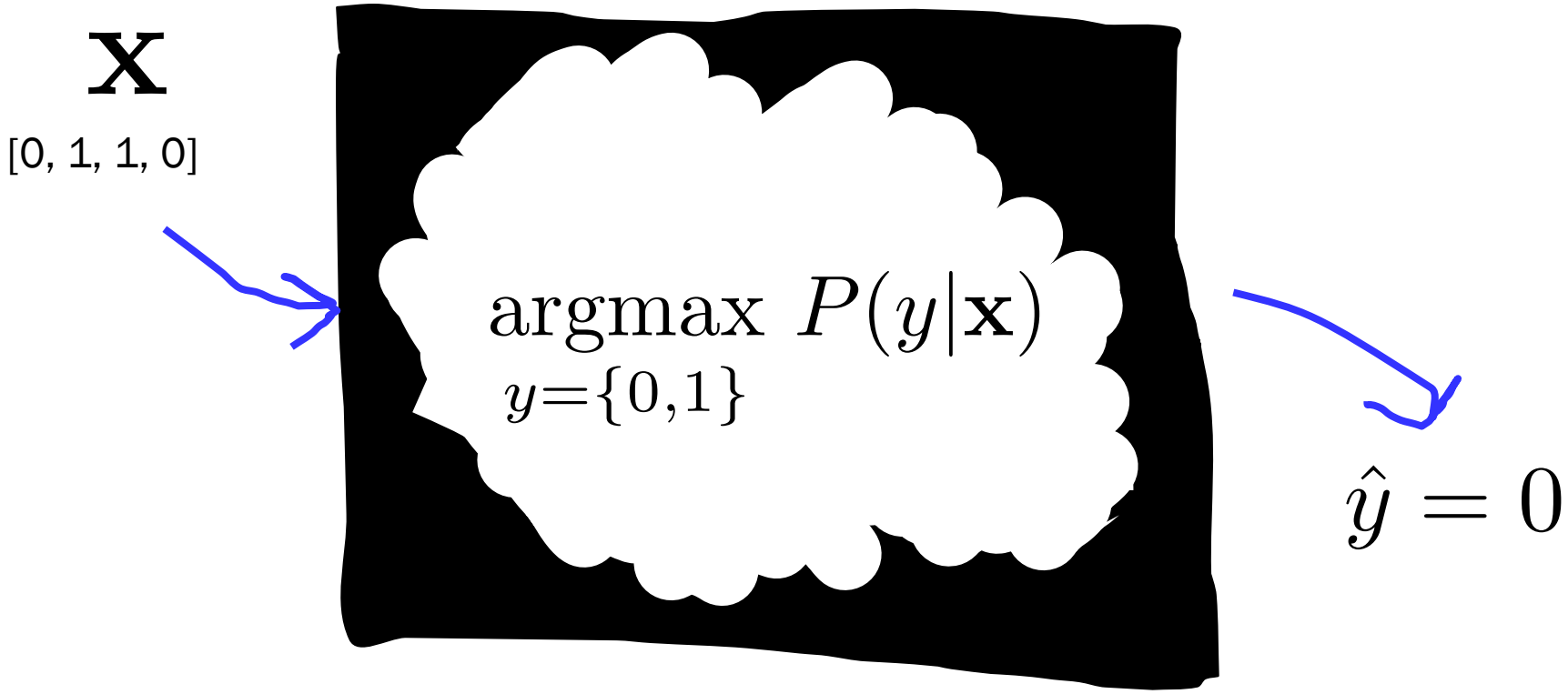
Making a prediction…

$g_\theta(\mathbf{x})$?



Making a prediction…

$g_\theta(\mathbf{x})$?



Making a prediction…

Big Assumption

Naïve Bayes Assumption:

$$P(\mathbf{x}|y) = \prod_i P(x_i|y)$$

Prediction

Simply chose the class label that is the most likely given the data. Make Naïve Bayes assumption

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

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

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

$$= \arg\max_{y=\{0,1\}} P(Y = y) \prod_i P(X_i = x_i | Y = y)$$

$$= \arg\max_{y=\{0,1\}} \log P(Y = y) + \sum_i \log P(X_i = x_i | Y = y)$$

Must learn params for this

Must learn params for this

Training Naïve Bayes, is estimating parameters for bernoullis

Thus training is just counting.

Learning Probabilities from Data

Various probabilities you will need to compute for Naive Bayesian Classifier (using **MLE** here):

$$\hat{p}(X_i = 1|Y = 0) = \frac{(\#\text{ training examples where } X_i = 1 \text{ and } Y = 0)}{(\#\text{ training examples where } Y = 0)}$$

$$\hat{p}(Y = 1) = \frac{(\#\text{ training examples where } Y = 1)}{(\#\text{ training examples})}$$

Learning Probabilities from Data

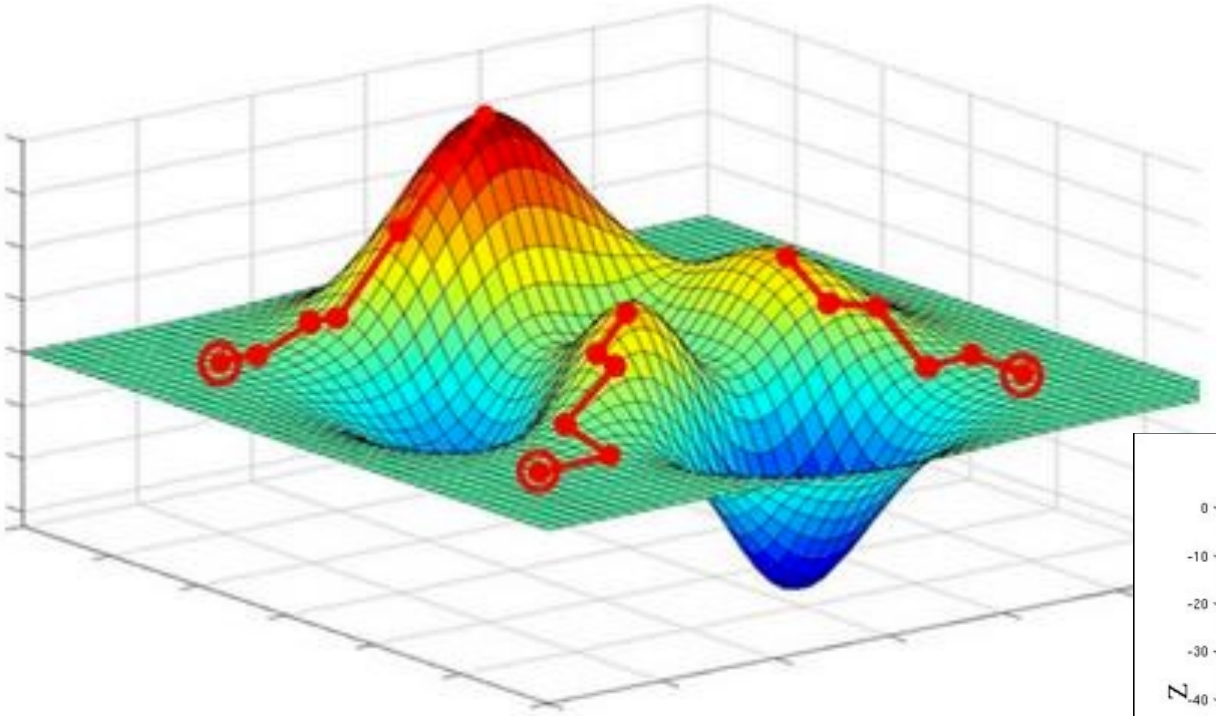
Various probabilities you will need to compute for Naive Bayesian Classifier (using **MAP with Laplace prior** here):

$$\hat{p}(X_i = 1|Y = 0) = \frac{(\#\text{ training examples where } X_i = 1 \text{ and } Y = 0) + 1}{(\#\text{ training examples where } Y = 0) + 2}$$

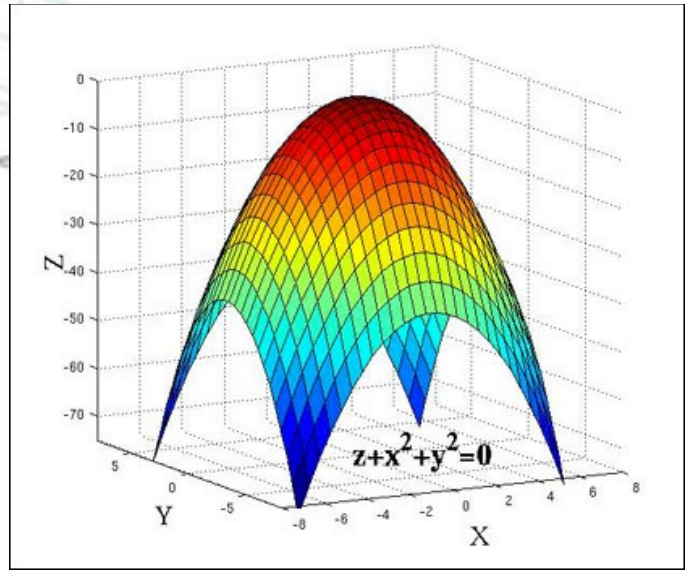
$$\hat{p}(Y = 1) = \frac{(\#\text{ training examples where } Y = 1) + 1}{(\#\text{ training examples}) + 2}$$

Optimization

Gradient Ascent



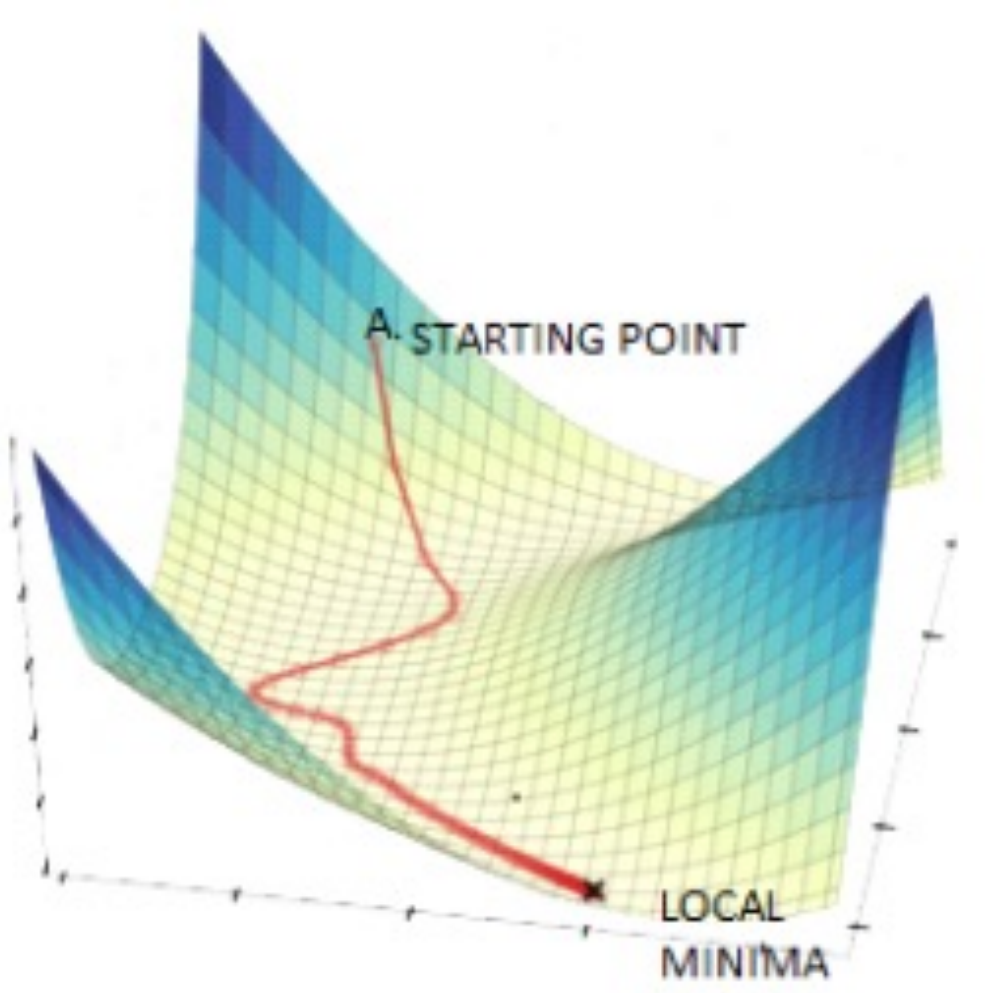
Logistic regression
LL function is
convex



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

Gradient ascent is your bread and butter algorithm for optimization (eg argmax)

Gradient Decent



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

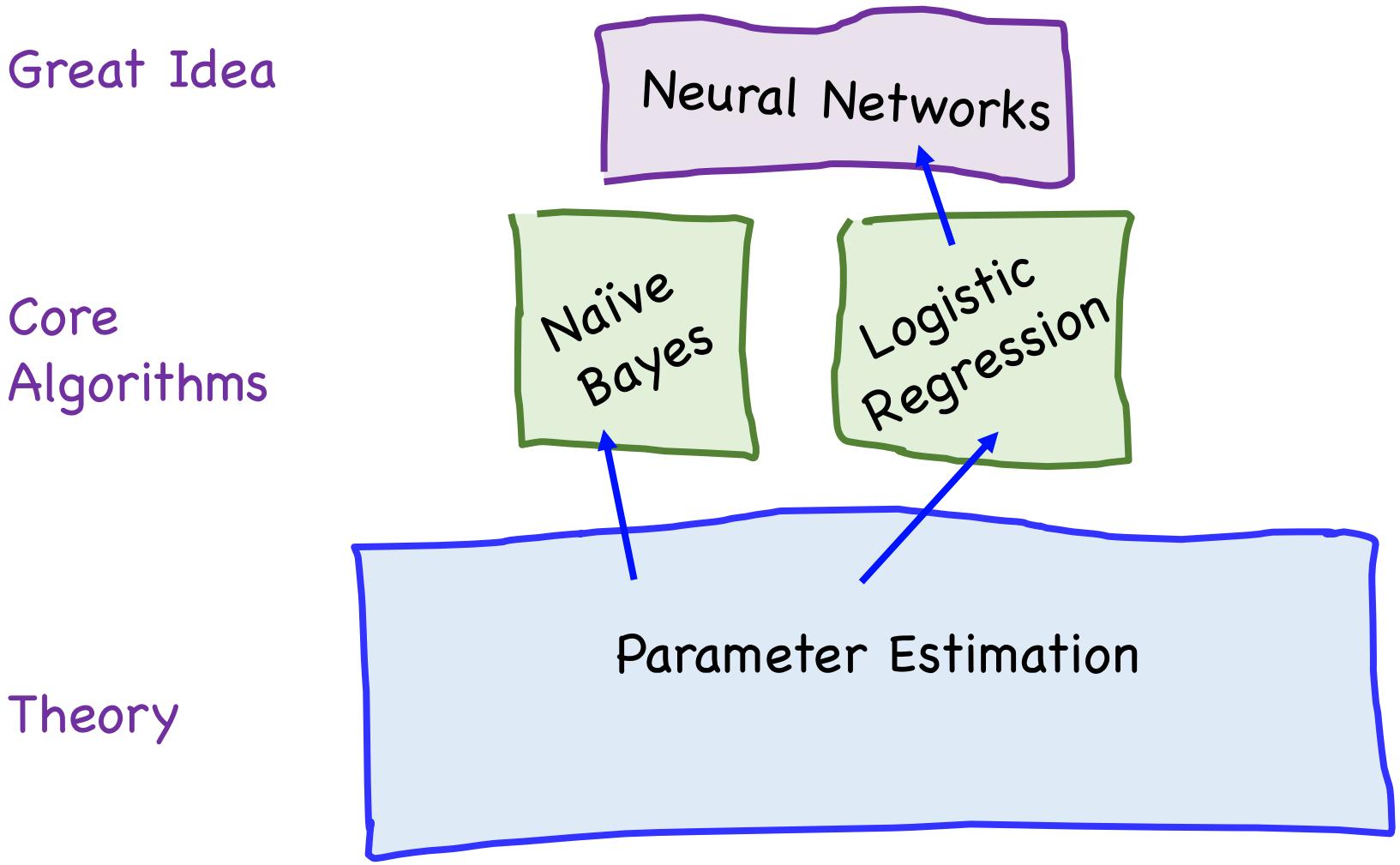
If someone gives you a gradient descent package, you should minimize **negative** log likelihood.

If you are writing an optimization yourself, feel free to gradient **ascent** on log likelihood :-)

End Review

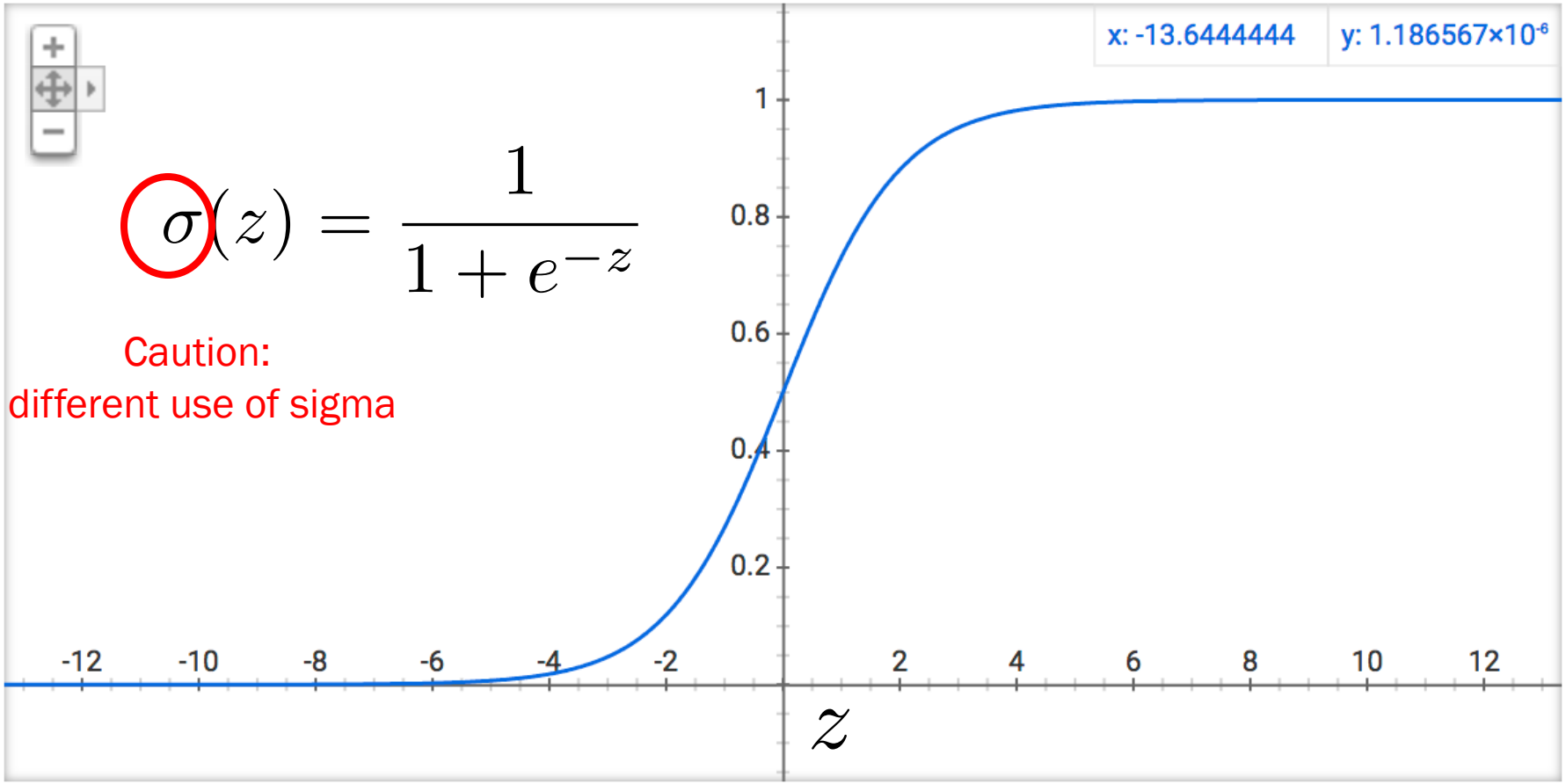
Logistic Regression

Machine Learning in CS109



Chapter 0: Background

Background: Sigmoid Function



The sigmoid function squashes z to be a number between 0 and 1

Background: Key Notation

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

Sigmoid function

$$\theta^T \mathbf{x} = \sum_{i=1}^{n} \theta_i x_i$$

$$= \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Weighted sum
(aka dot product)

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

Sigmoid function of
weighted sum

Concept Check: Weighted Sum

$$\theta^T \mathbf{x} = \sum_{i=1}^{n} \theta_i x_i$$

Weighted sum
(aka dot product)

$$= \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Let $\theta = [1, 3, 4, 5]$ and $\mathbf{x} = [0, 1, 1, 0]$.

What is the corresponding weighted sum between the feature vector and parameters (weights)?

Background: Chain Rule

Who knew calculus would be so useful?

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

Aka decomposition of composed functions

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

Chapter 1: Big Picture

From Naïve Bayes to Logistic Regression

In classification we care about P(Y | **X**)

Recall the Naive Bayes Classifier

- Predict P(Y | **X**)
- Use assumption that $P(\boldsymbol{X} \mid Y) = P(X_1, X_2, \ldots X_m \mid Y) = \prod_{i=1}^{m} P(X_i \mid Y)$
- That is a pretty big assumption…

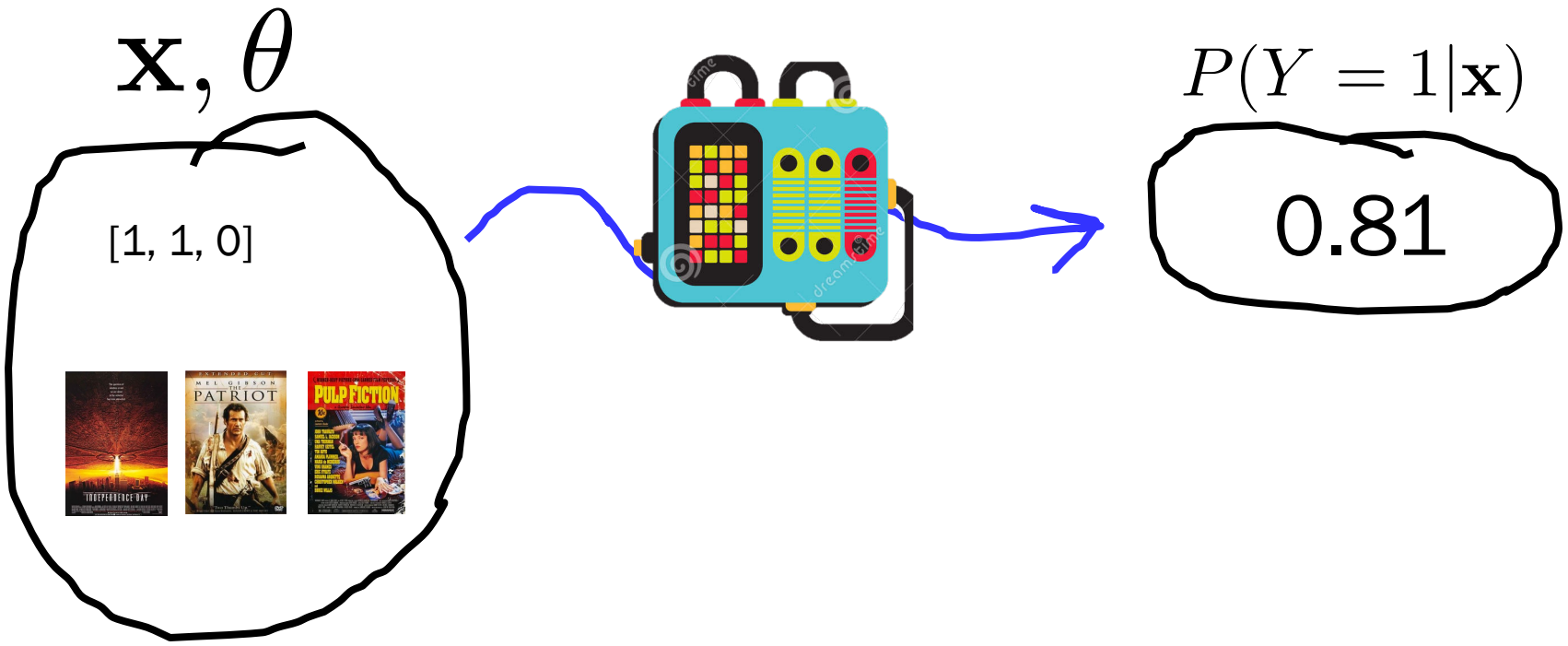
Could we model P(Y | **X**) directly?

- Welcome our friend: logistic regression!

Logistic Regression Assumption

Could we model P(Y | **X**) directly?

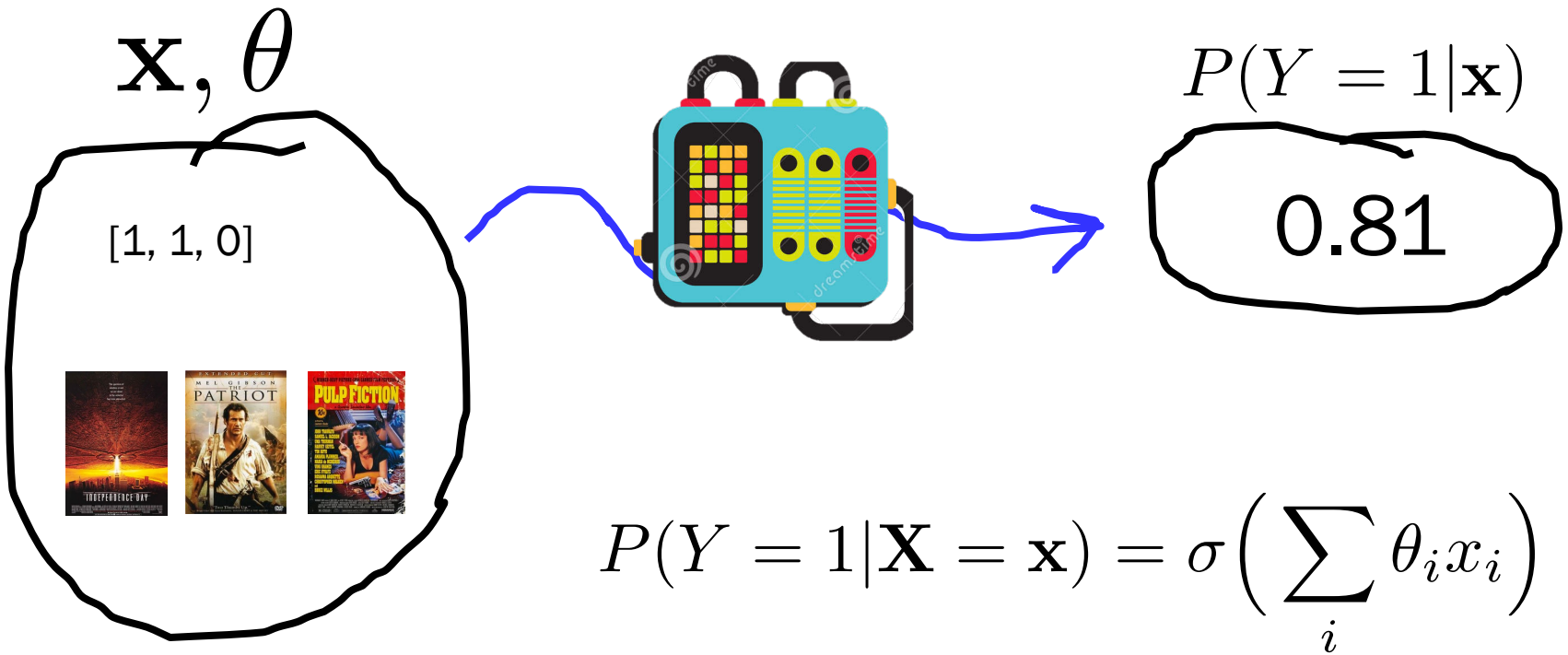
- Welcome our friend: logistic regression!



Logistic Regression Assumption

Could we model P(Y | **X**) directly?

- Welcome our friend: logistic regression!

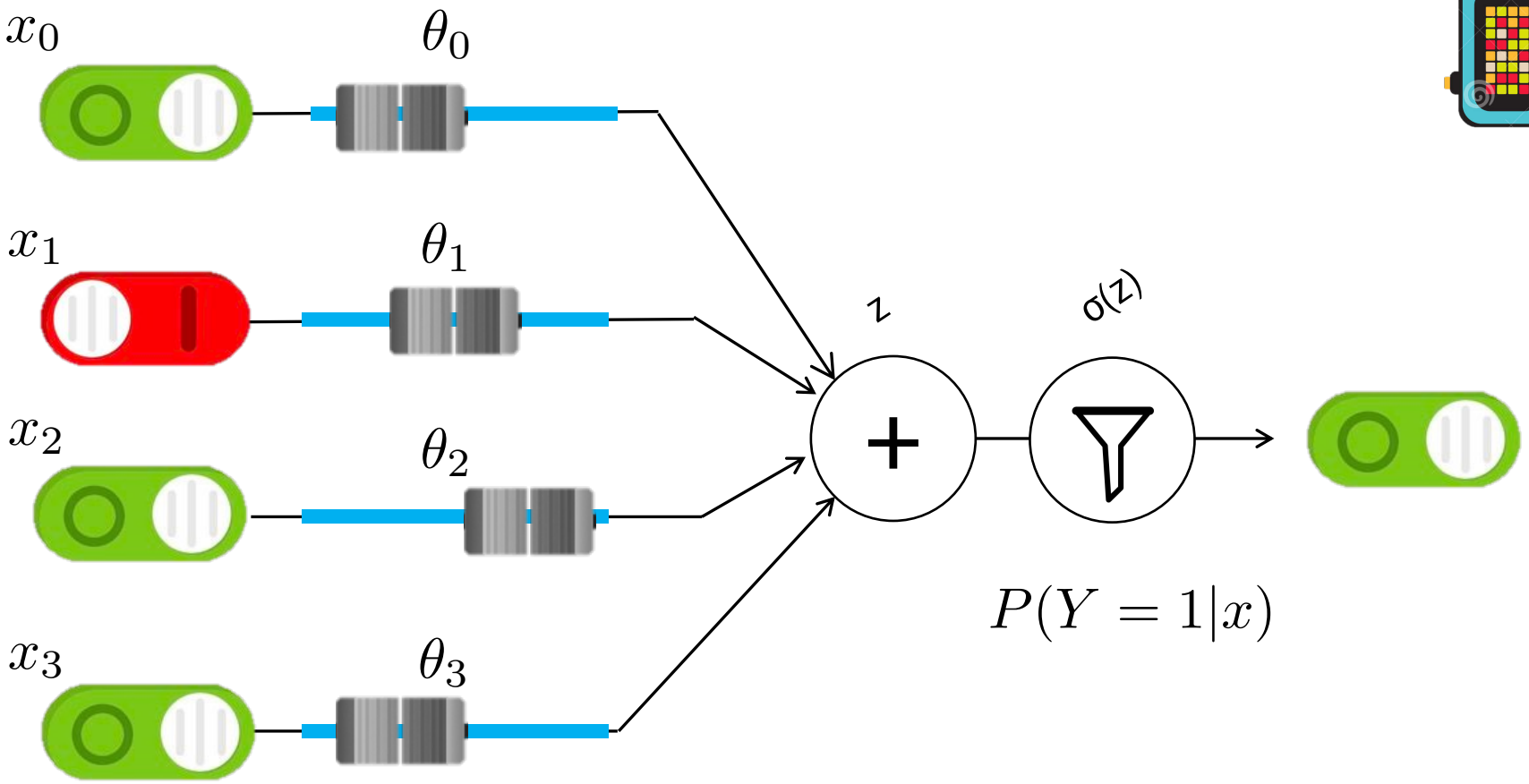


$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Logistic Regression Assumption

$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Logistic Regression

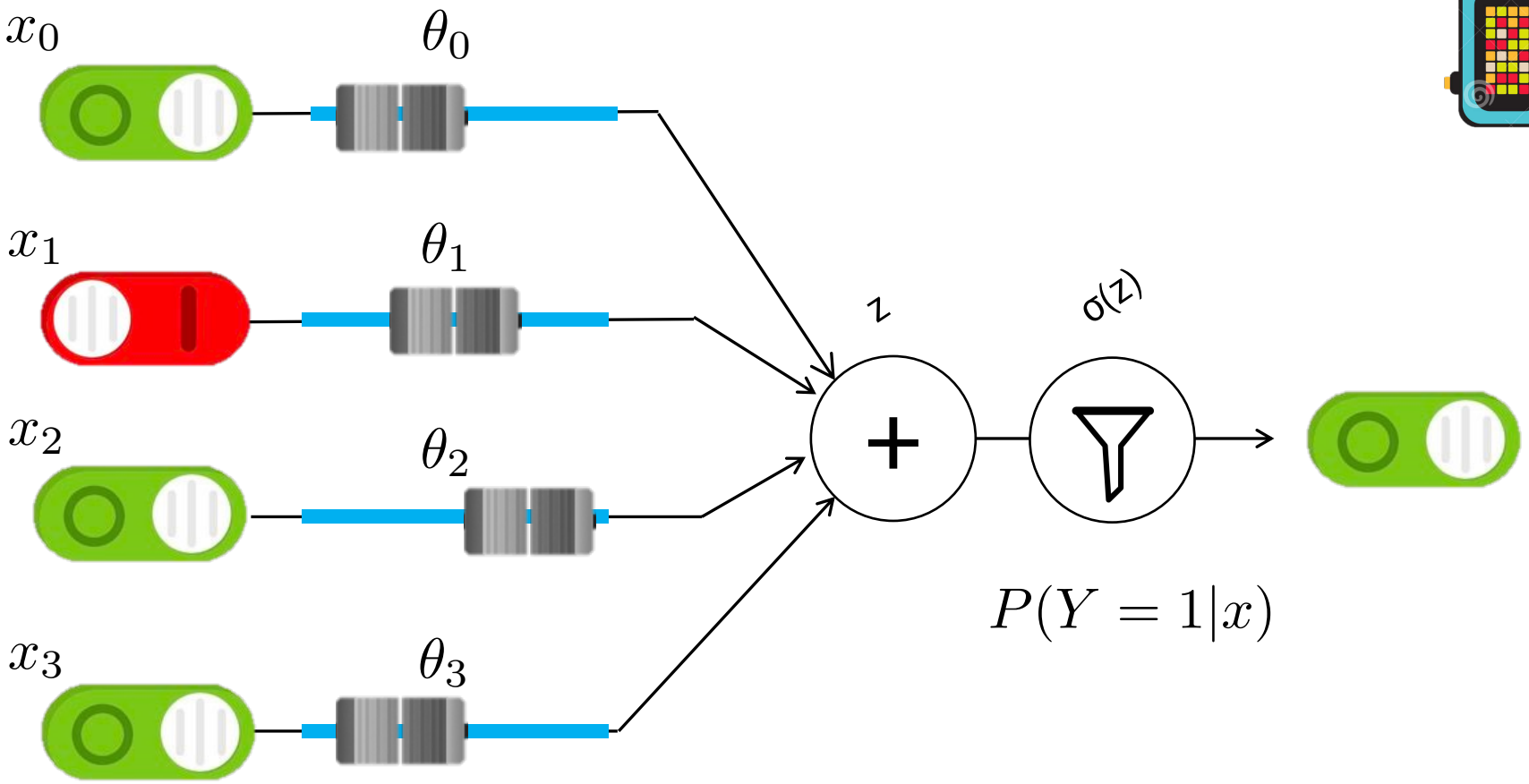


$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Logistic Regression

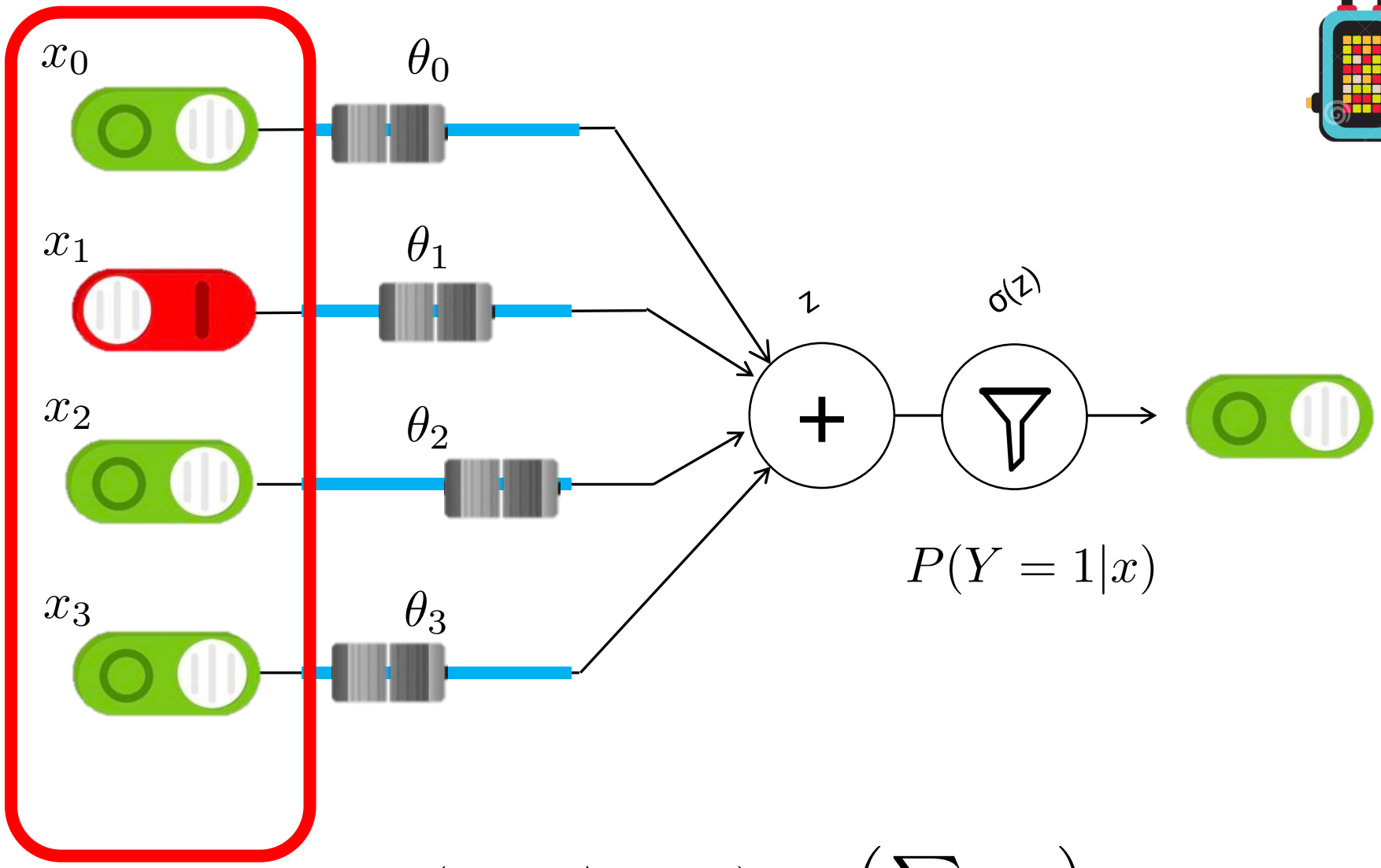
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\Big(\sum_i \theta_i x_i\Big)$$

Logistic Regression



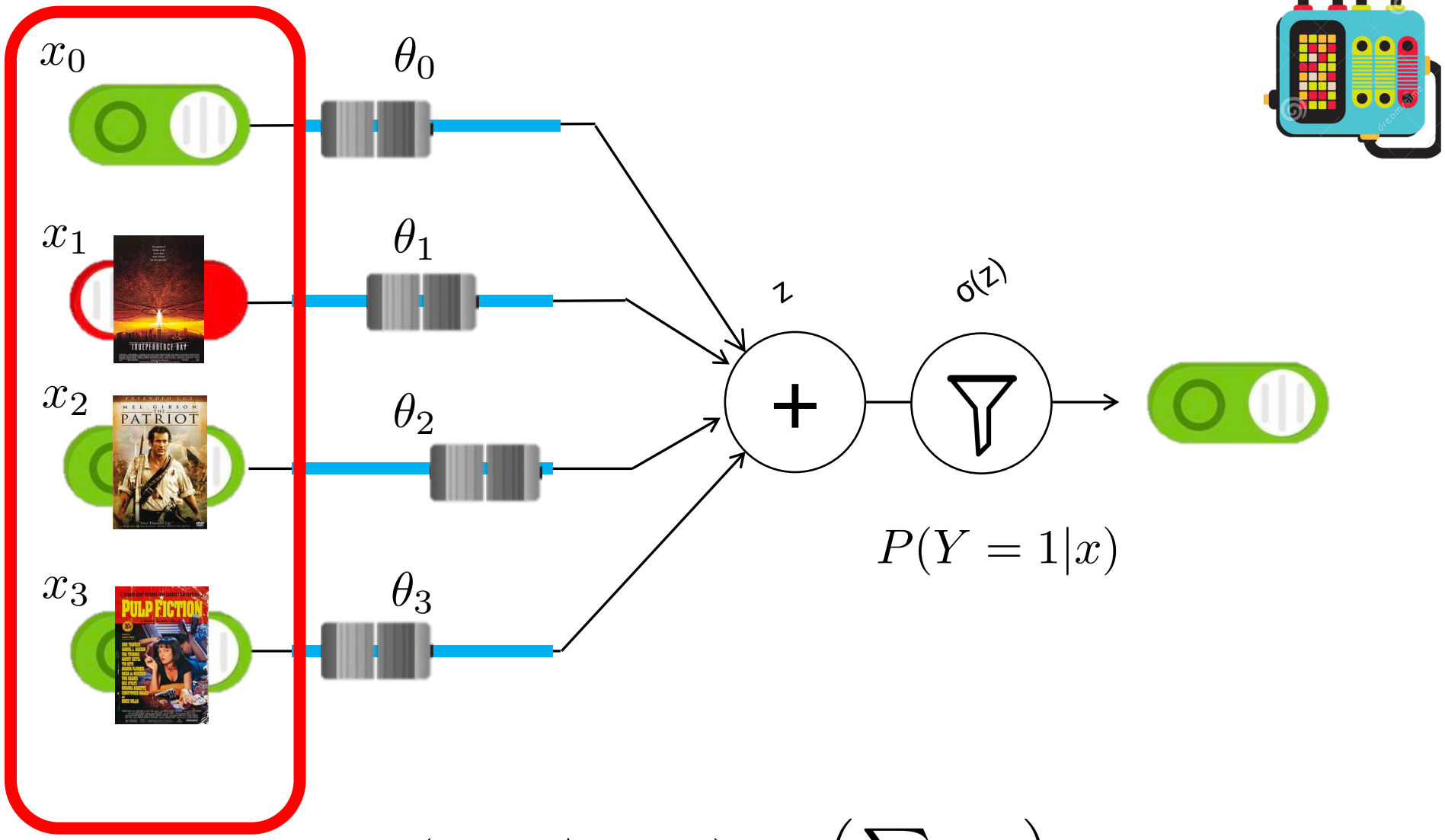
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Inputs $x = [0, 1, 1]$



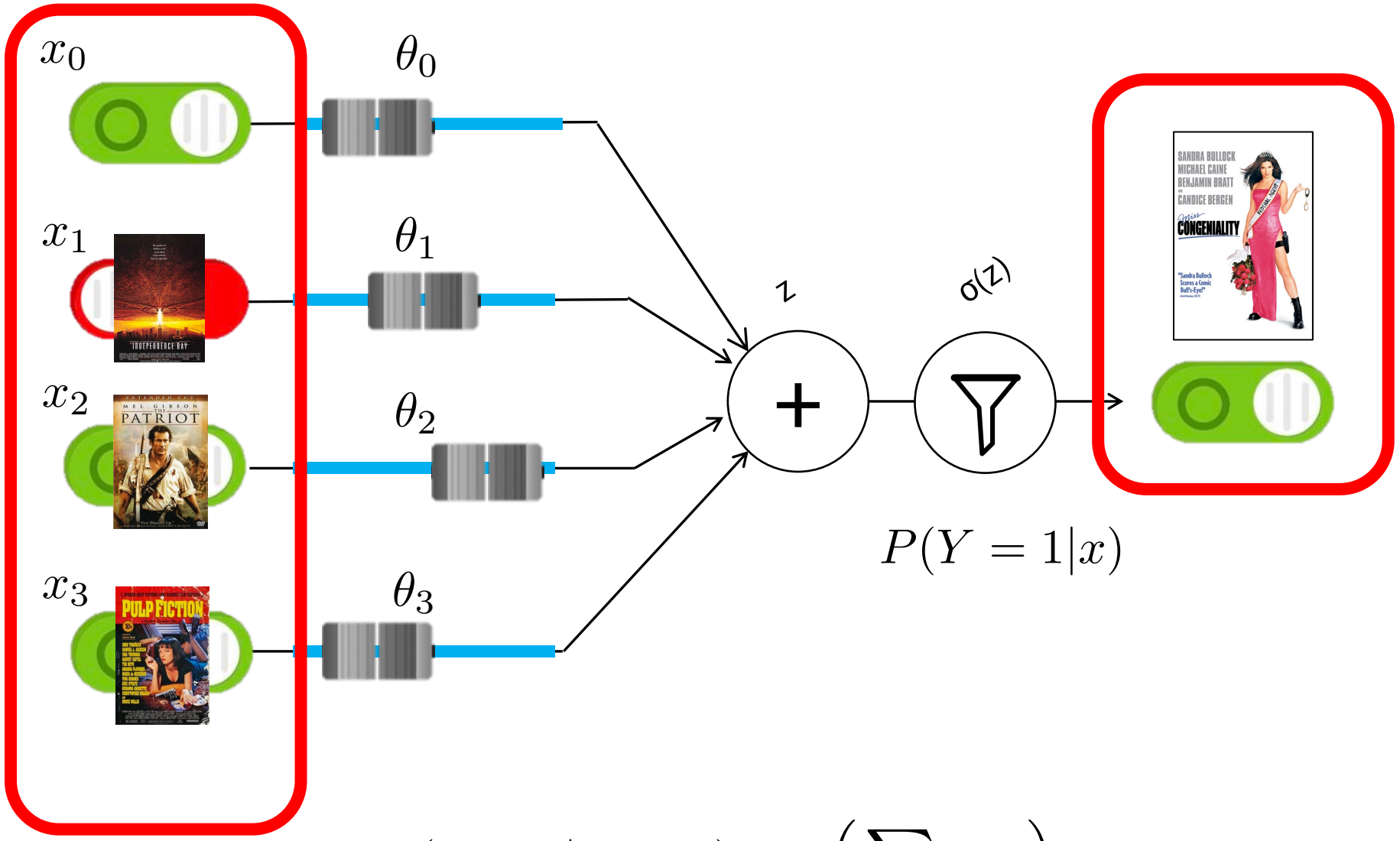
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Inputs



$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

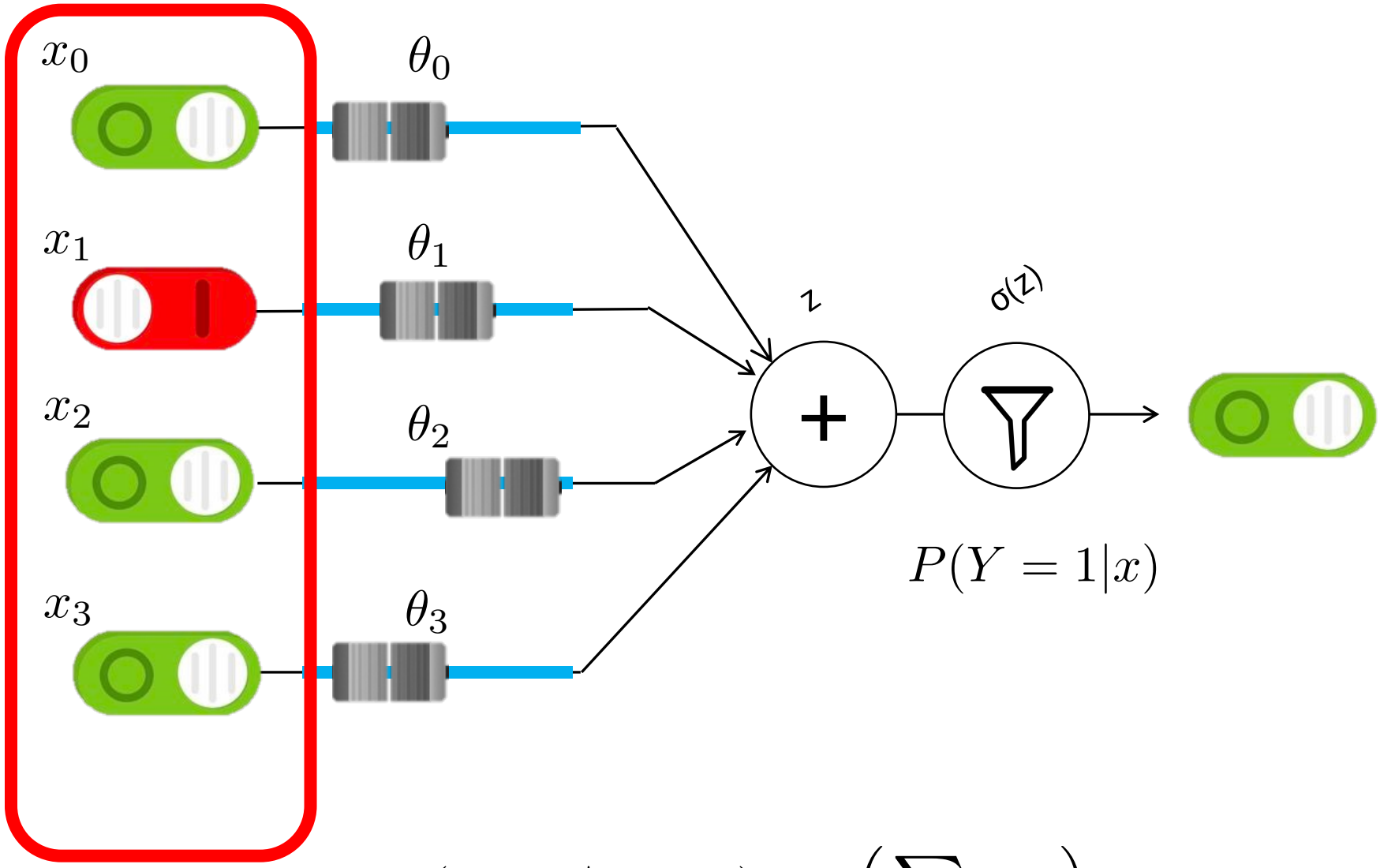
Inputs + Output



$$P(Y = 1|x)$$

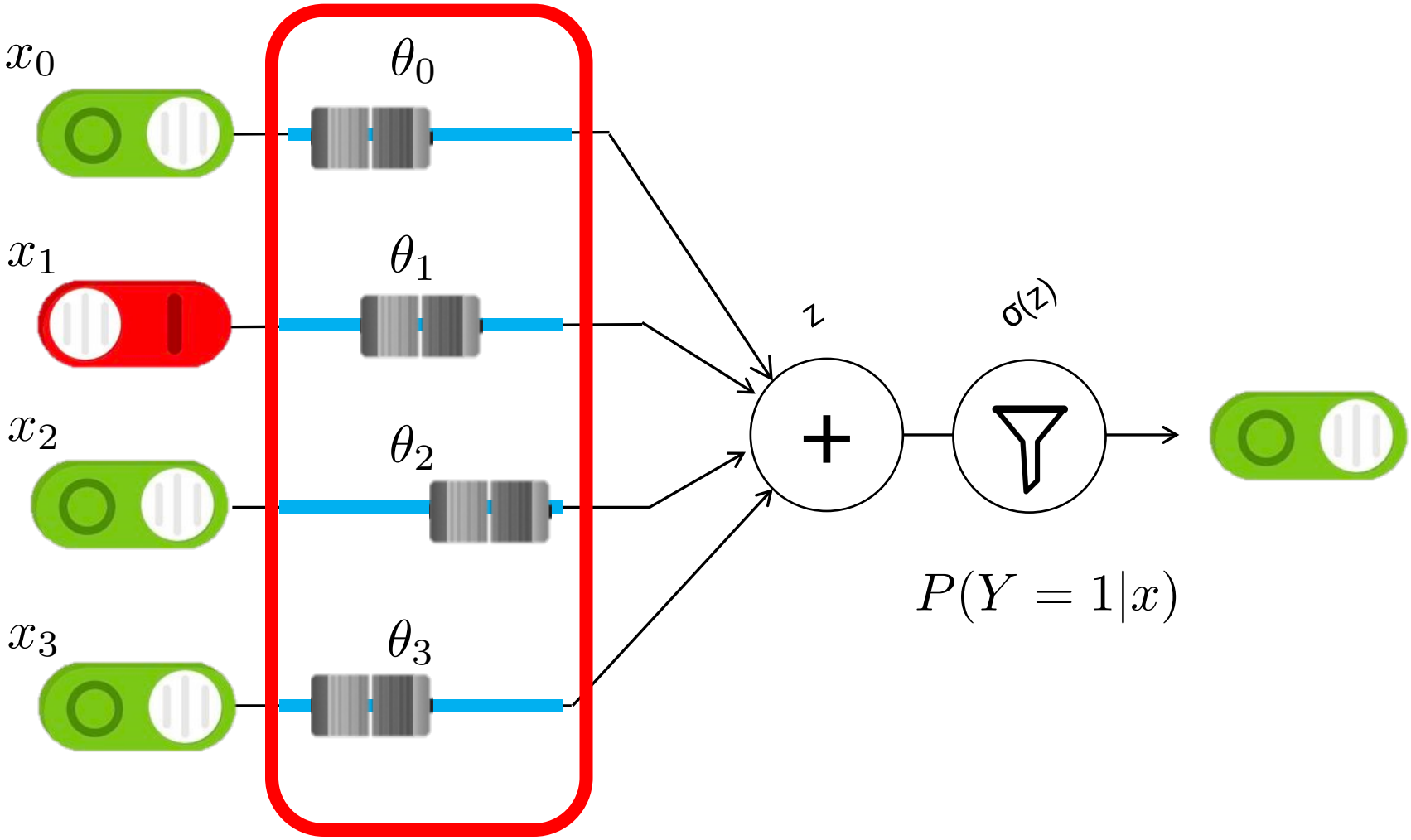
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\Big(\sum_i \theta_i x_i\Big)$$

Inputs



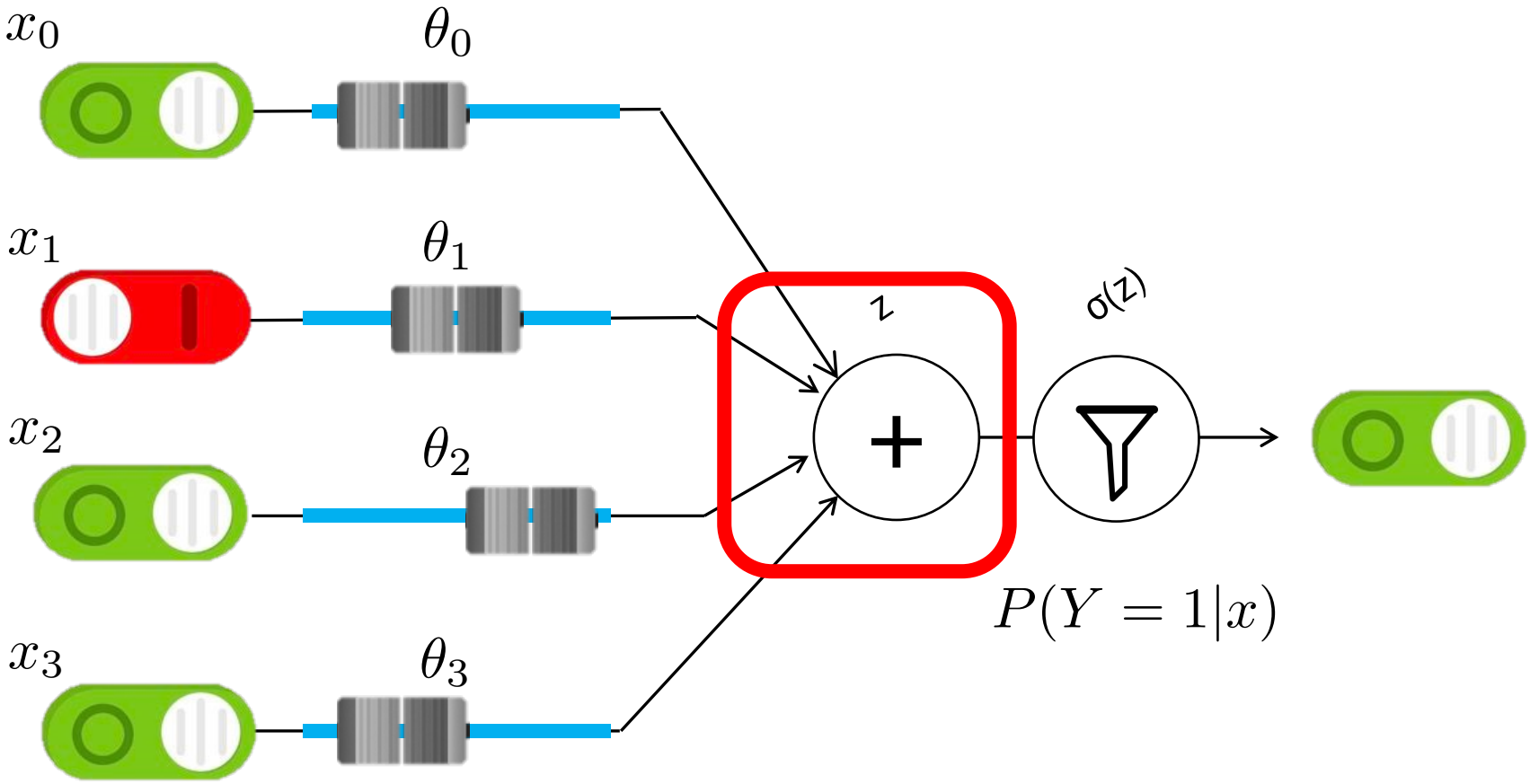
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Weights



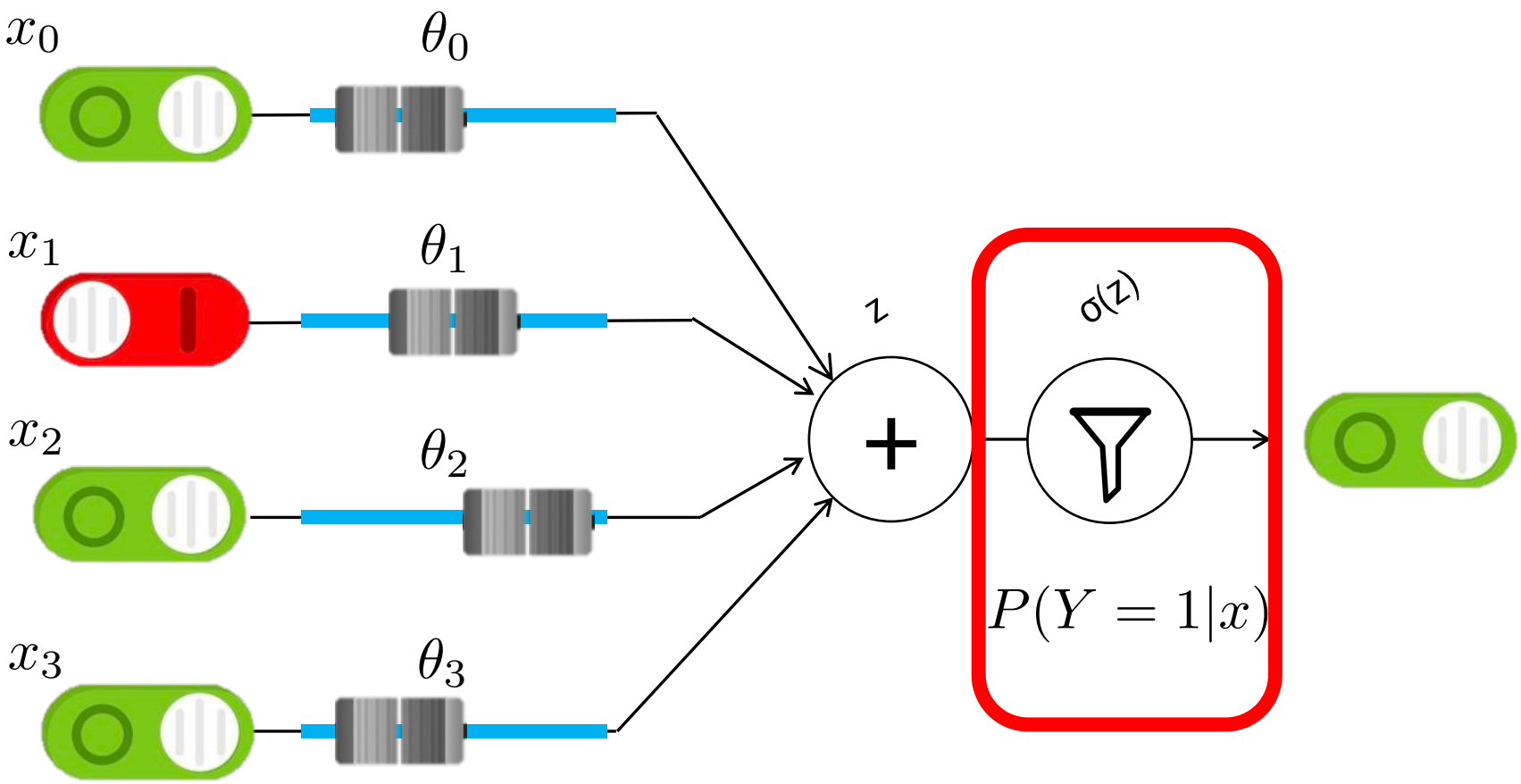
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\Big(\sum_i \theta_i x_i\Big)$$

Weighed Sum



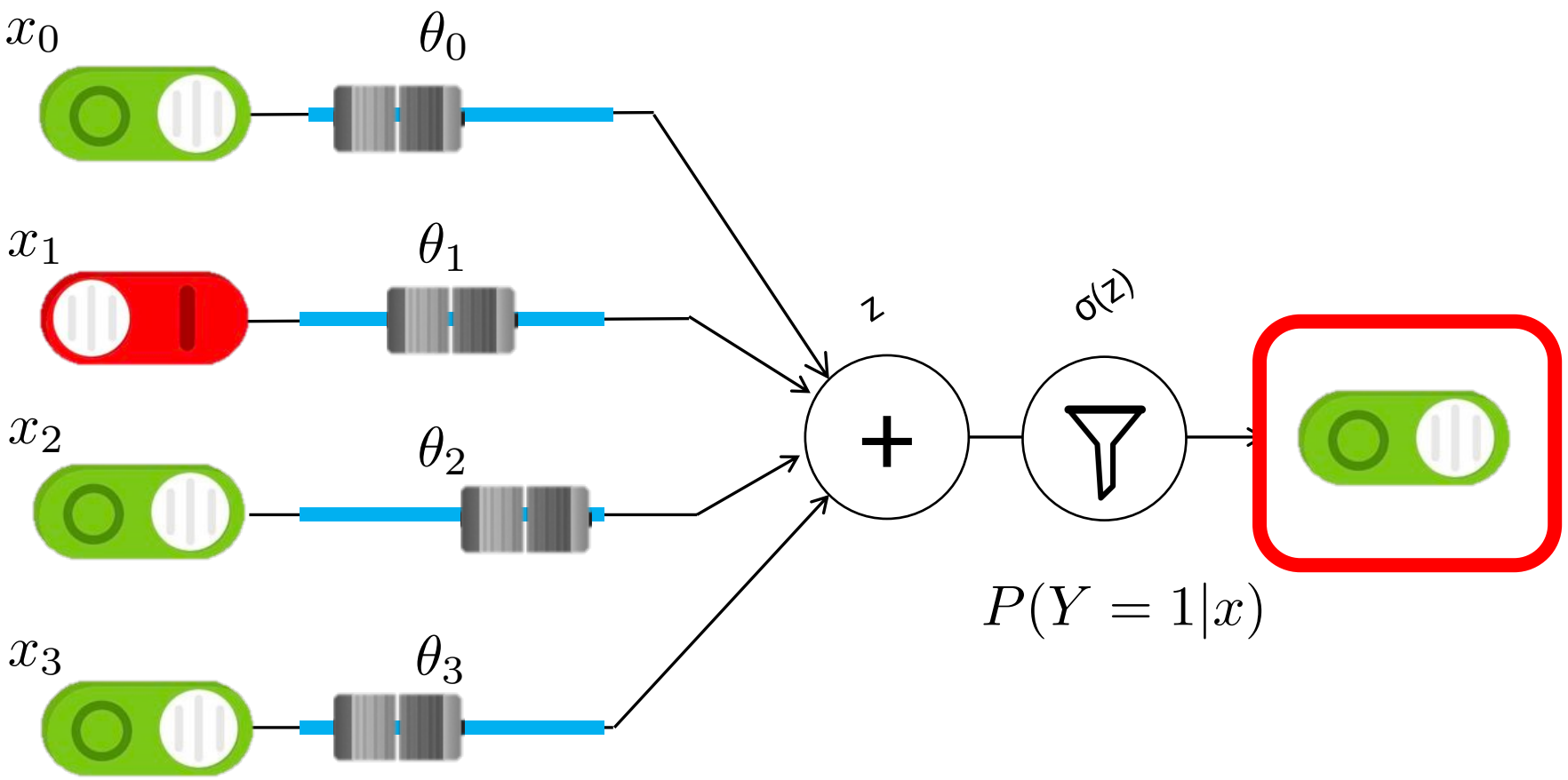
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\Big(\sum_i \theta_i x_i\Big)$$

Squashing Function



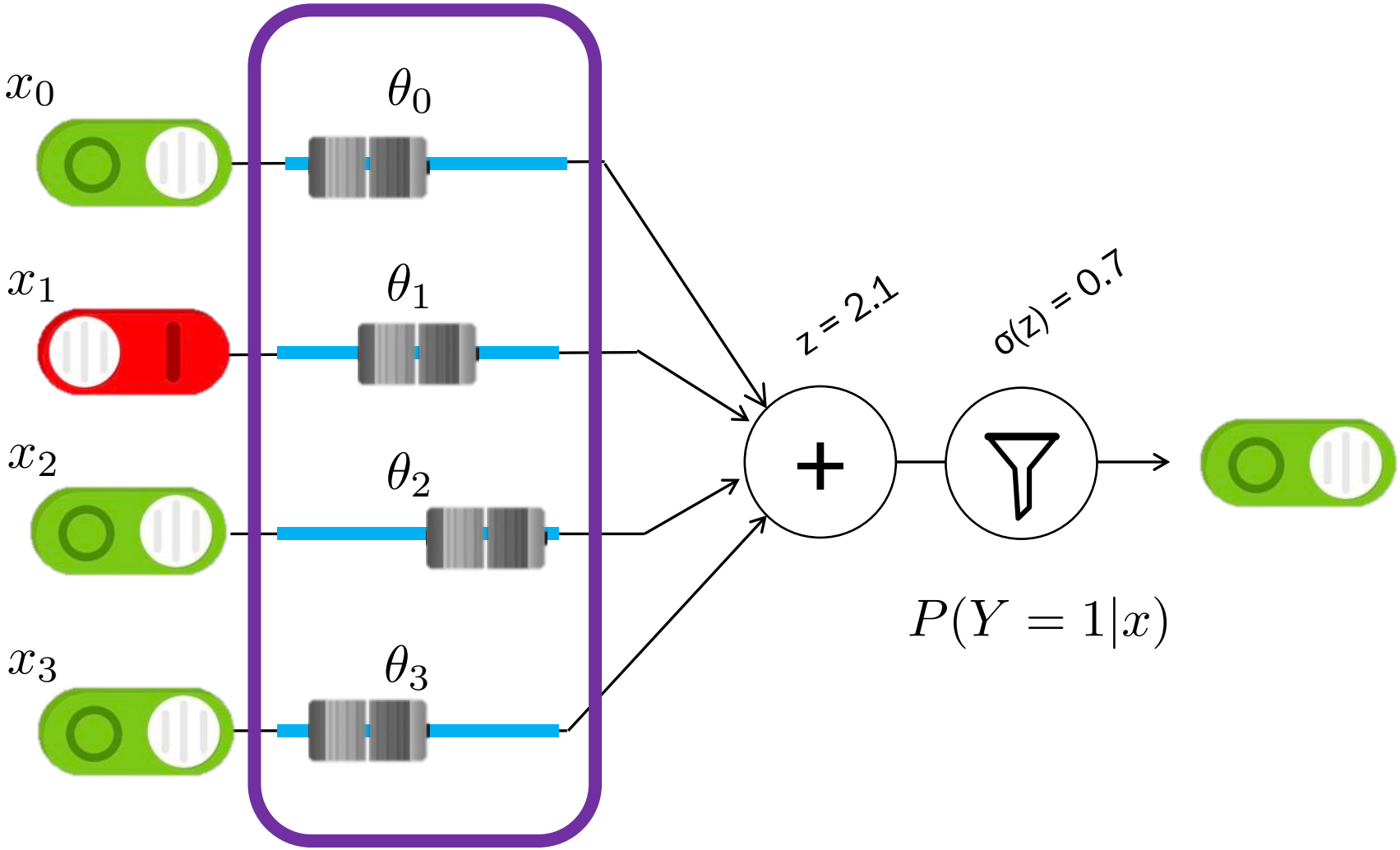
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\Big(\sum_i \theta_i x_i\Big)$$

Prediction



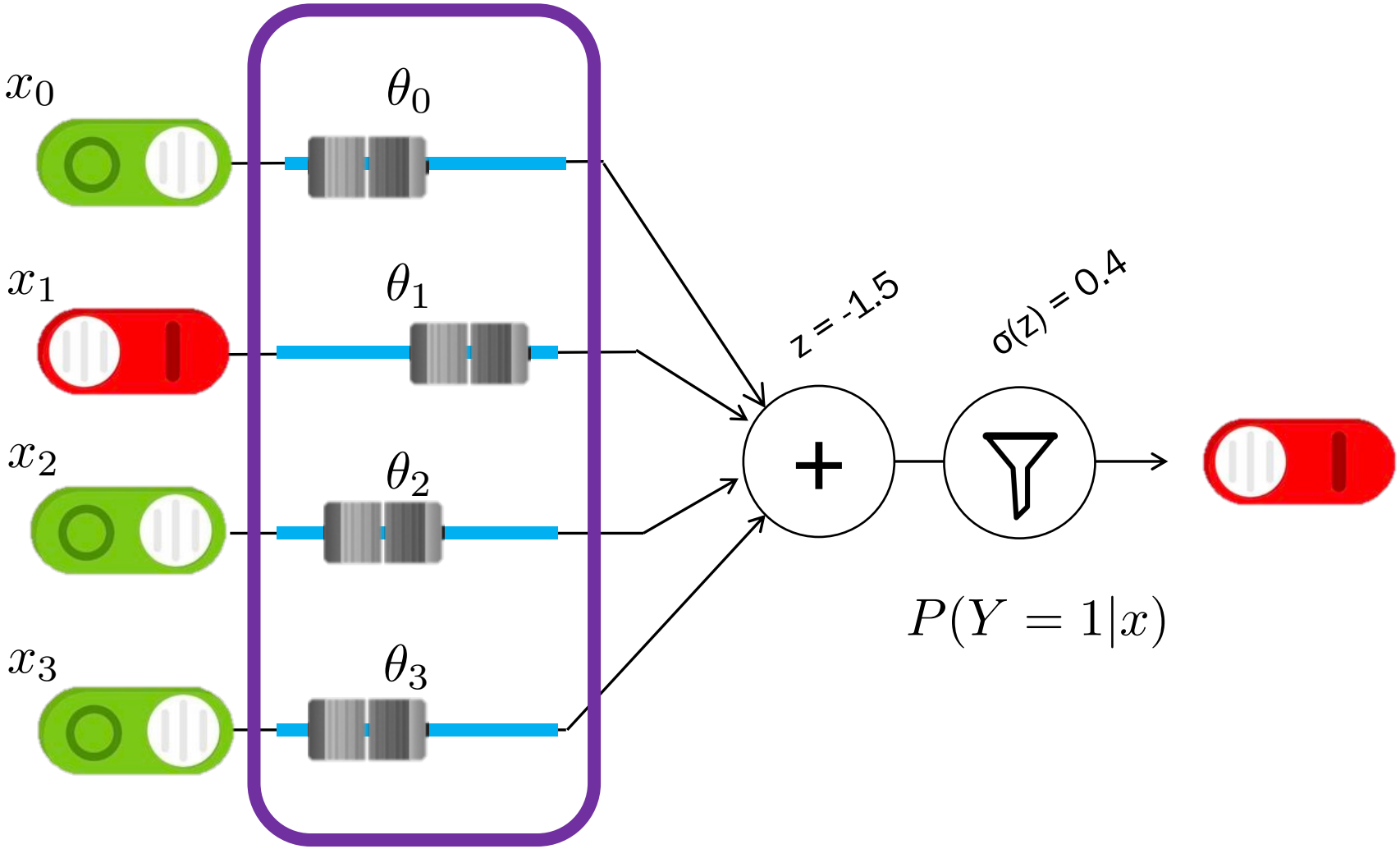
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\Big(\sum_i \theta_i x_i\Big)$$

Parameters Affect Prediction



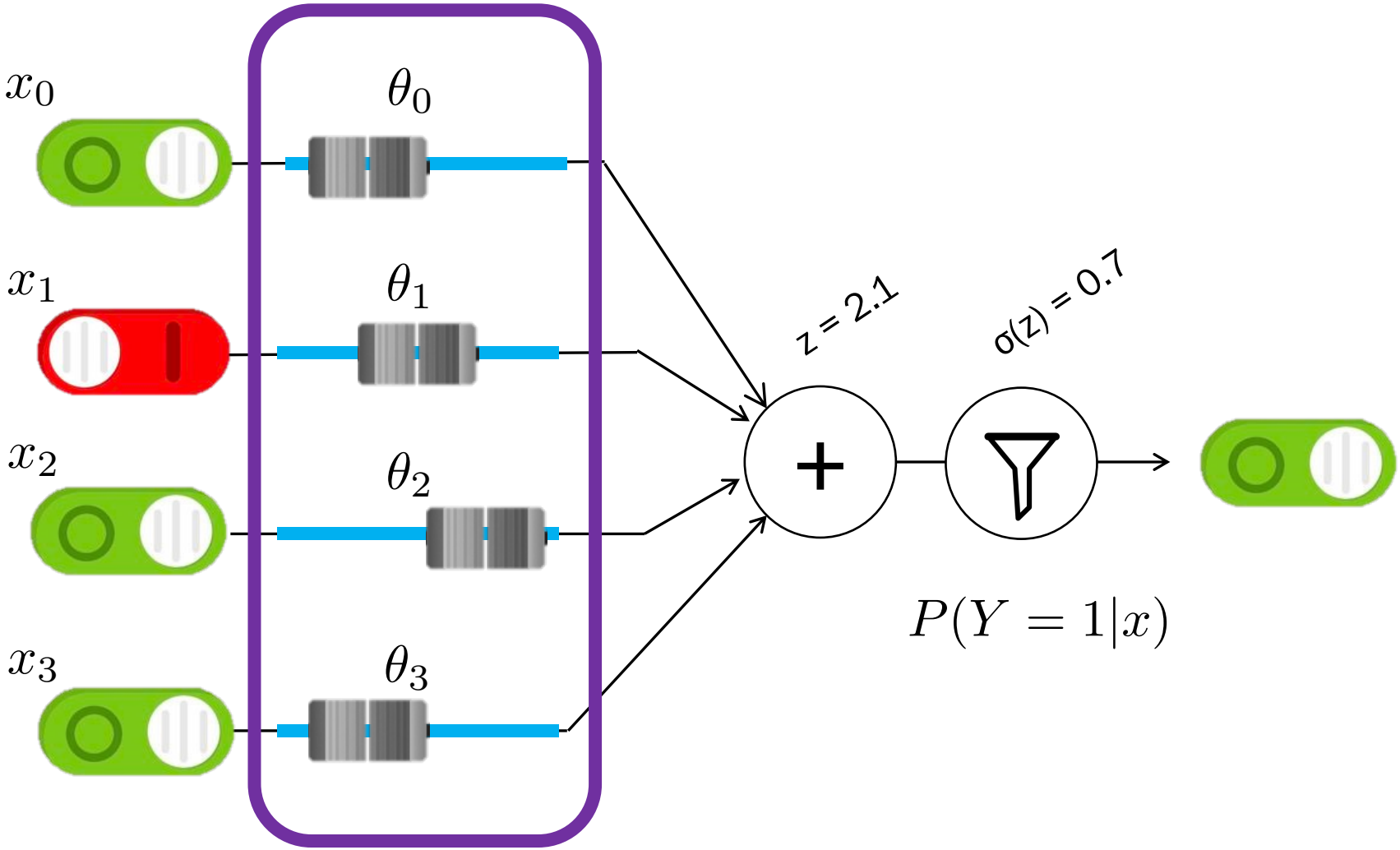
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\Big(\sum_i \theta_i x_i\Big)$$

Parameters Affect Prediction



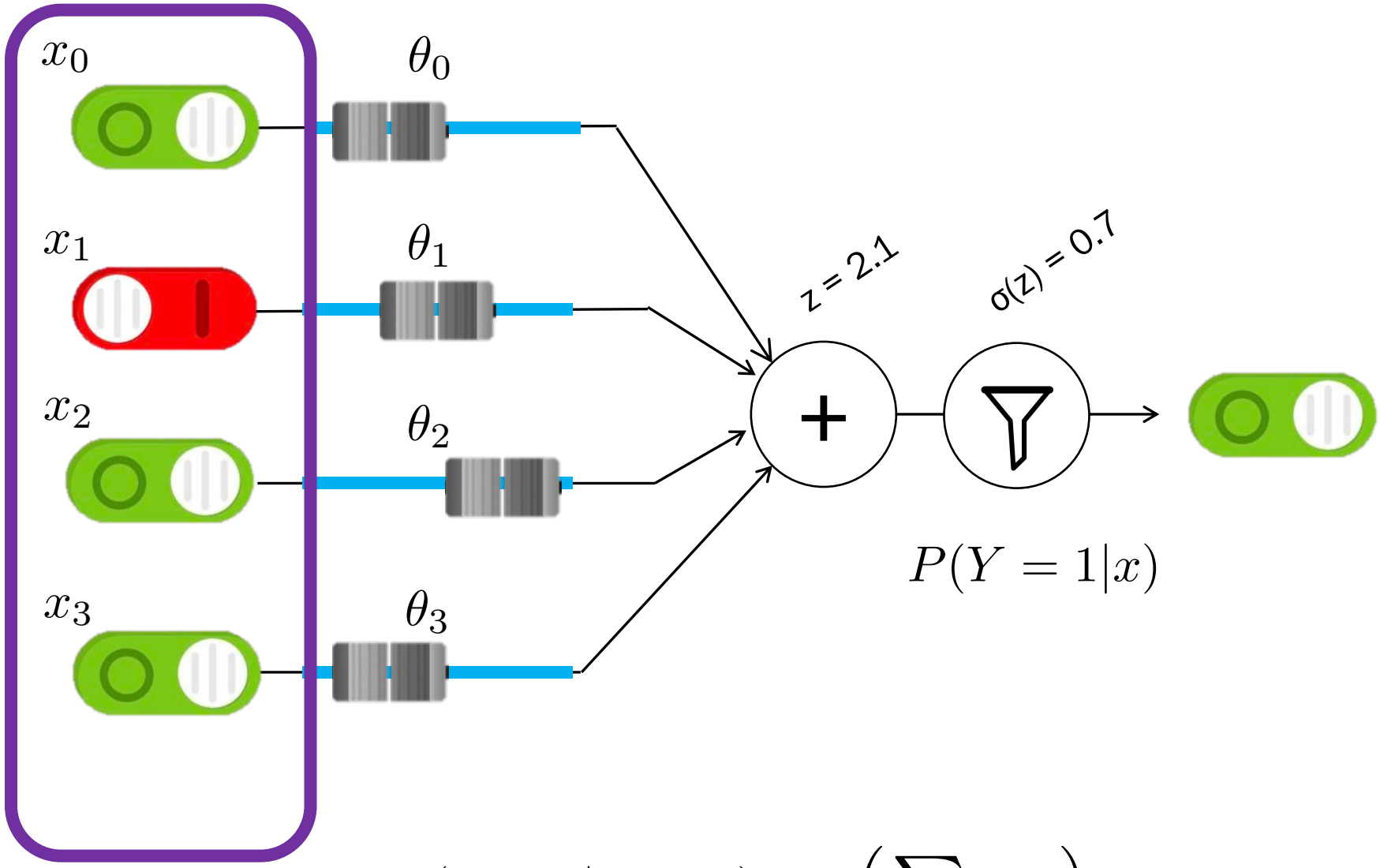
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\Big(\sum_i \theta_i x_i\Big)$$

Parameters Affect Prediction



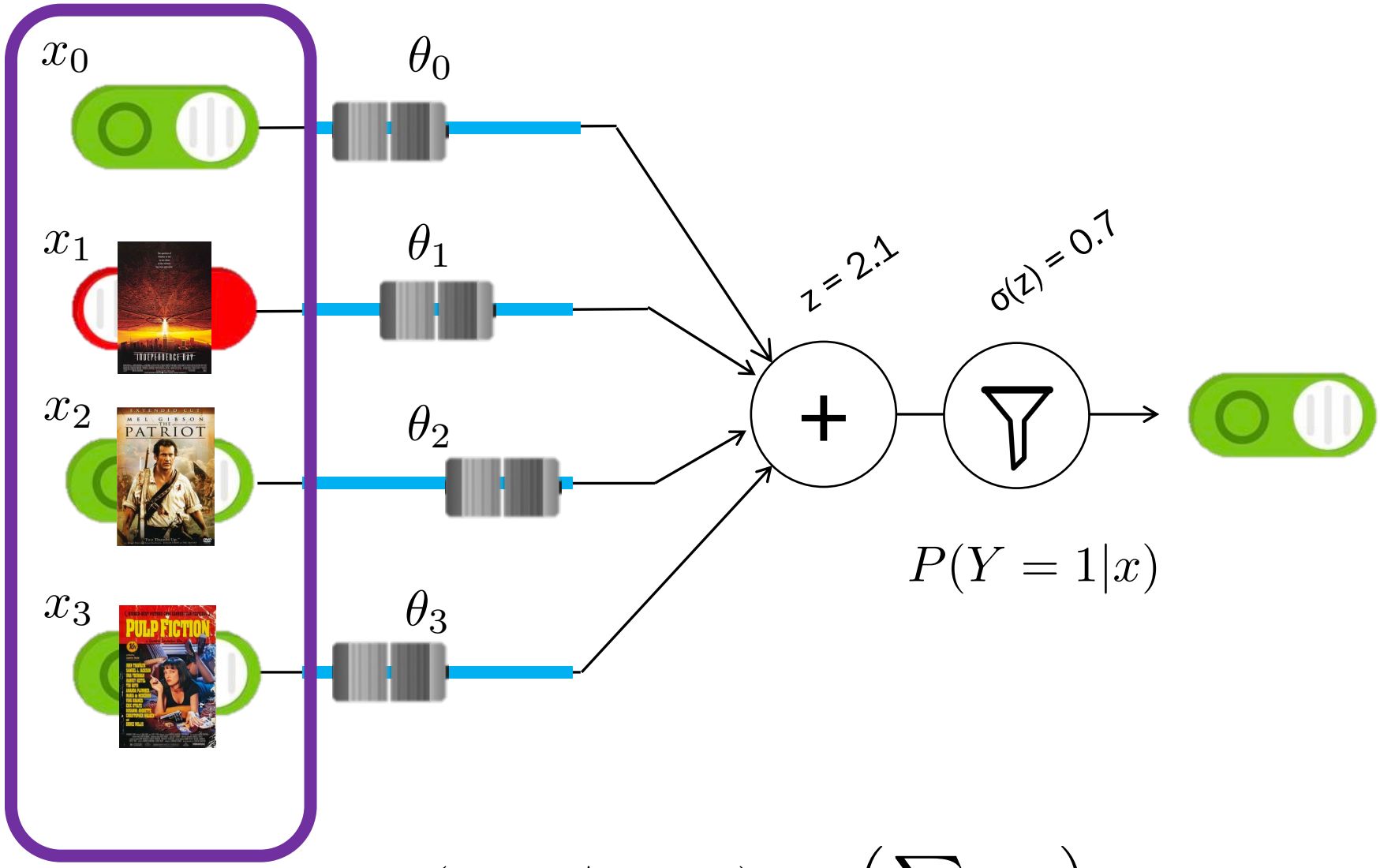
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\Big(\sum_i \theta_i x_i\Big)$$

Different Predictions for Different Inputs



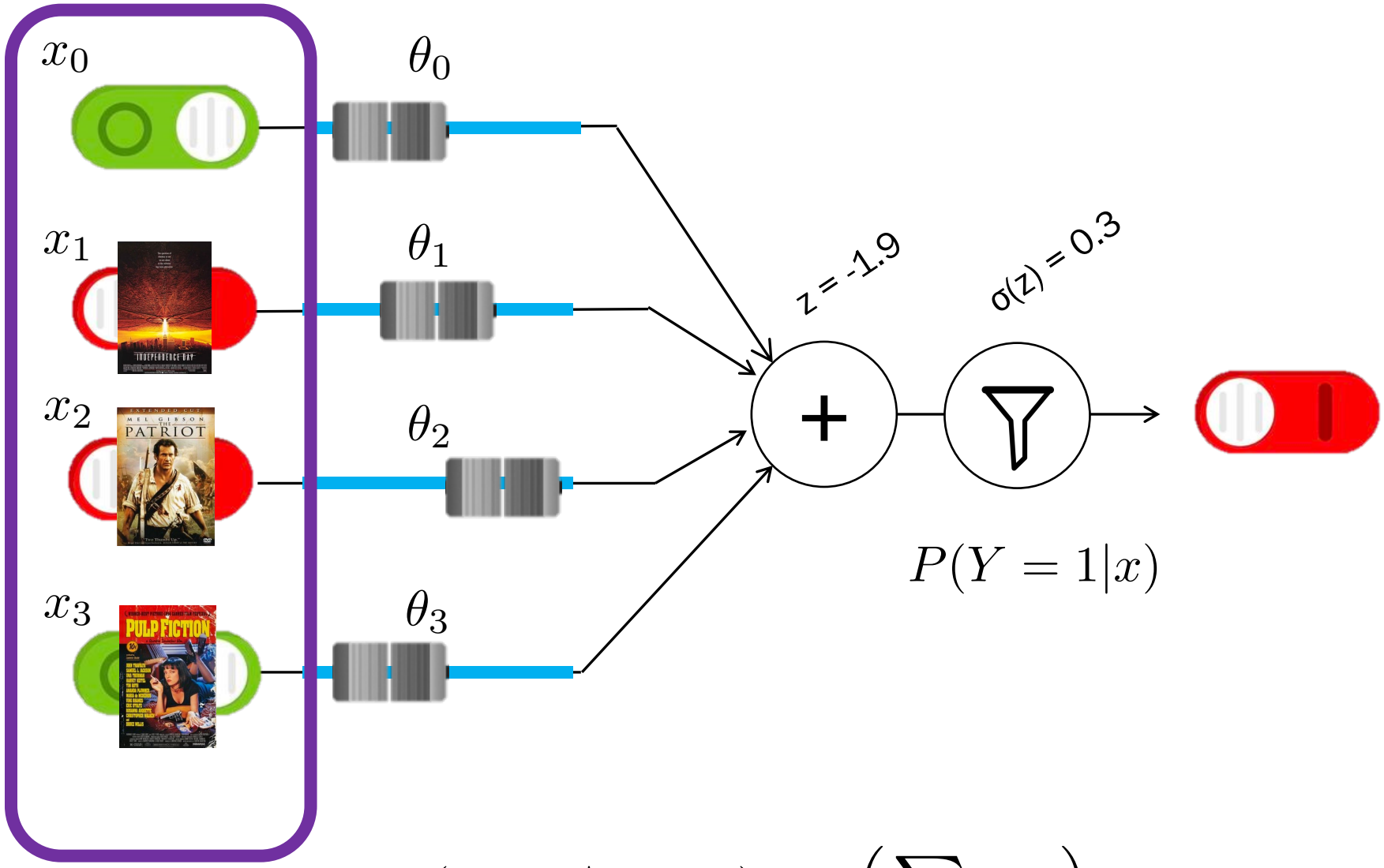
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\Big(\sum_i \theta_i x_i\Big)$$

Different Predictions for Different Inputs



$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\Big(\sum_i \theta_i x_i\Big)$$

Different Predictions for Different Inputs



$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\Big(\sum_i \theta_i x_i\Big)$$

Logistic Regression Assumption

Model *conditional* likelihood P(Y | **X**) directly

- Model this probability with *logistic* function:

$$P(Y = 1|\mathbf{X}) = \sigma(z) \text{ where } z = \theta_0 + \sum_{i=1}^{m} \theta_i x_i$$

- For simplicity define $x_0 = 1$ so $z = \theta^T \mathbf{x}$

- Since $\mathrm{P}(Y = 0 \mid \mathbf{X}) + \mathrm{P}(Y = 1 \mid \mathbf{X}) = 1$:

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

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

Recall:
Sigmoid function

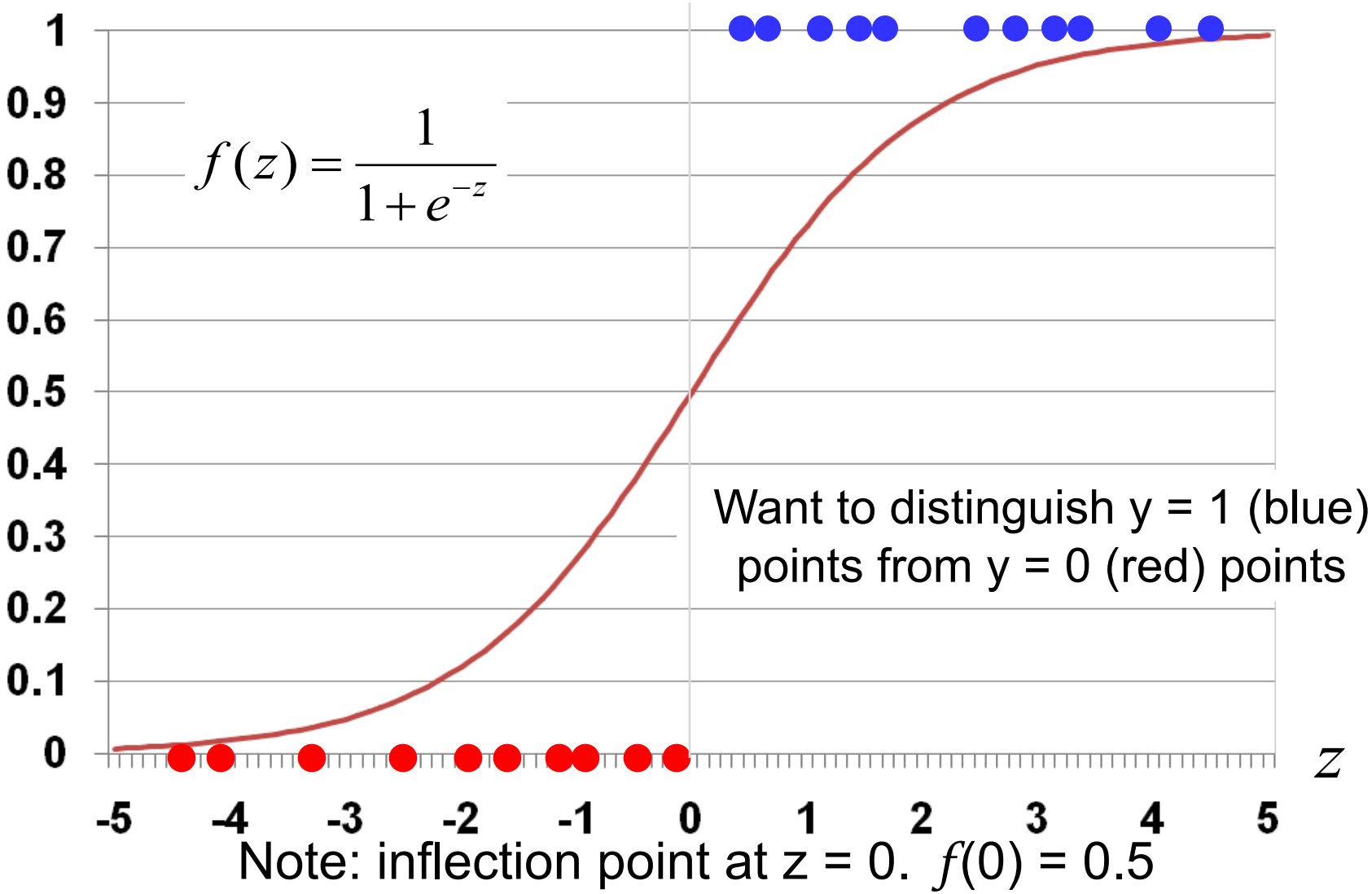
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Big Assumption

Logistic Regression Assumption:

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

The Sigmoid Function



Note: inflection point at z = 0. $f(0) = 0.5$

What is in a Name

Regression Algorithms

Linear Regression 

Classification Algorithms

Naïve Bayes 

Logistic Regression 

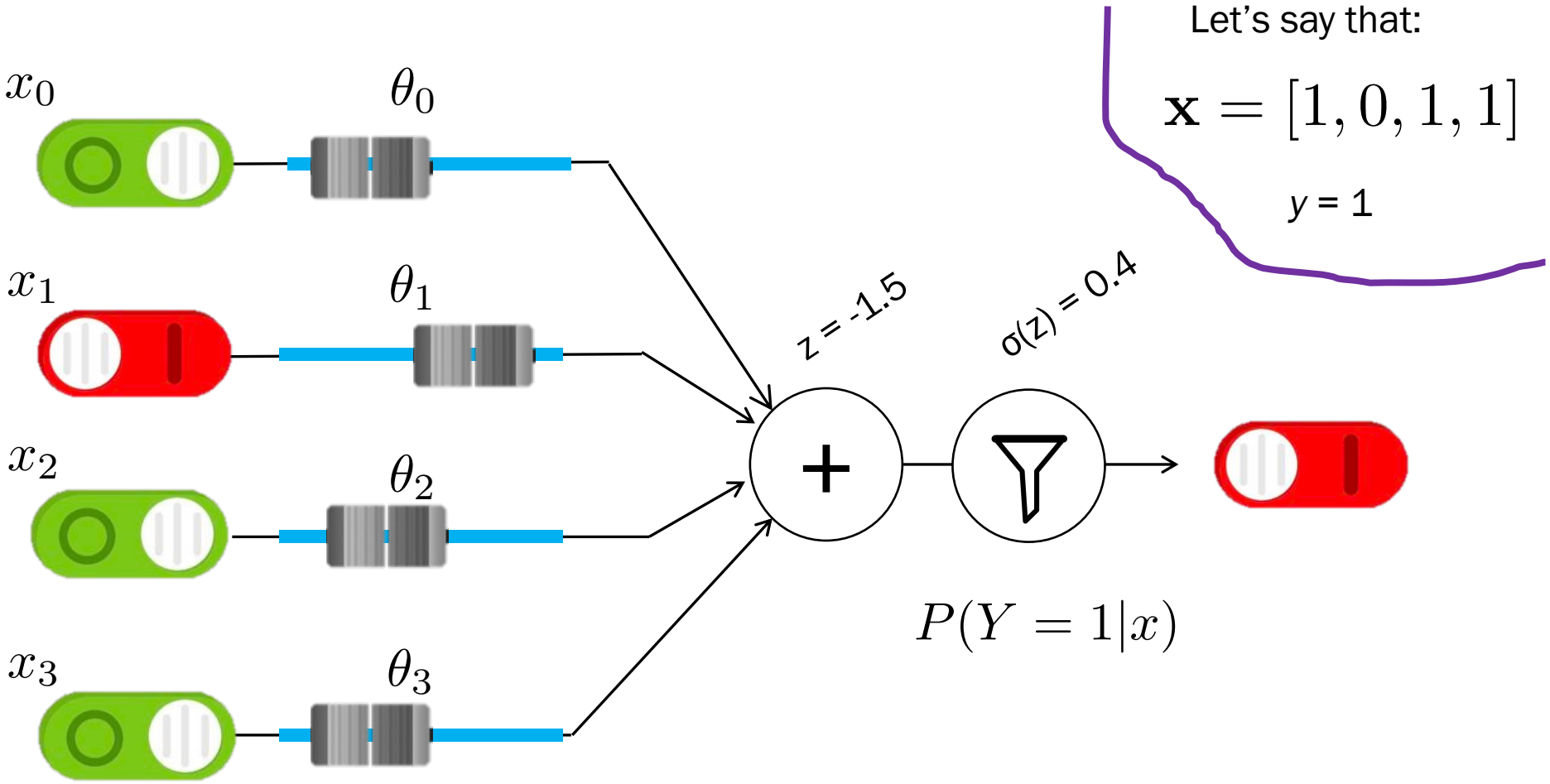
Awesome classifier, terrible name

If Joel could rename it he would call it: Siggy Classification

What makes for a “smart” logistic regression algorithm?

Logistic regression gets its ***intelligence*** from its thetas (aka its parameters)

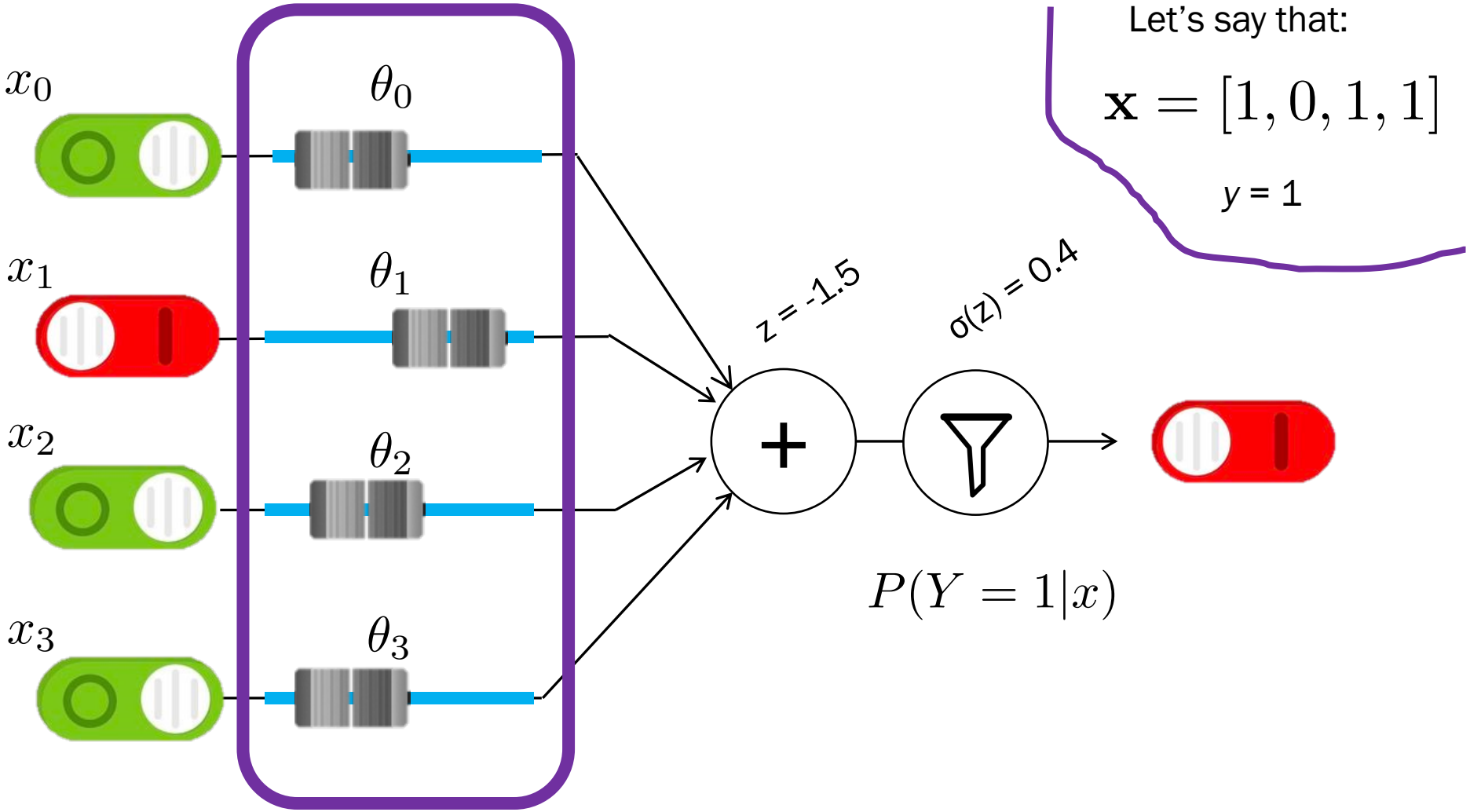
How Do We Learn Parameters?



$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\Big(\sum_i \theta_i x_i\Big) = 0.4$$

Data looks unlikely

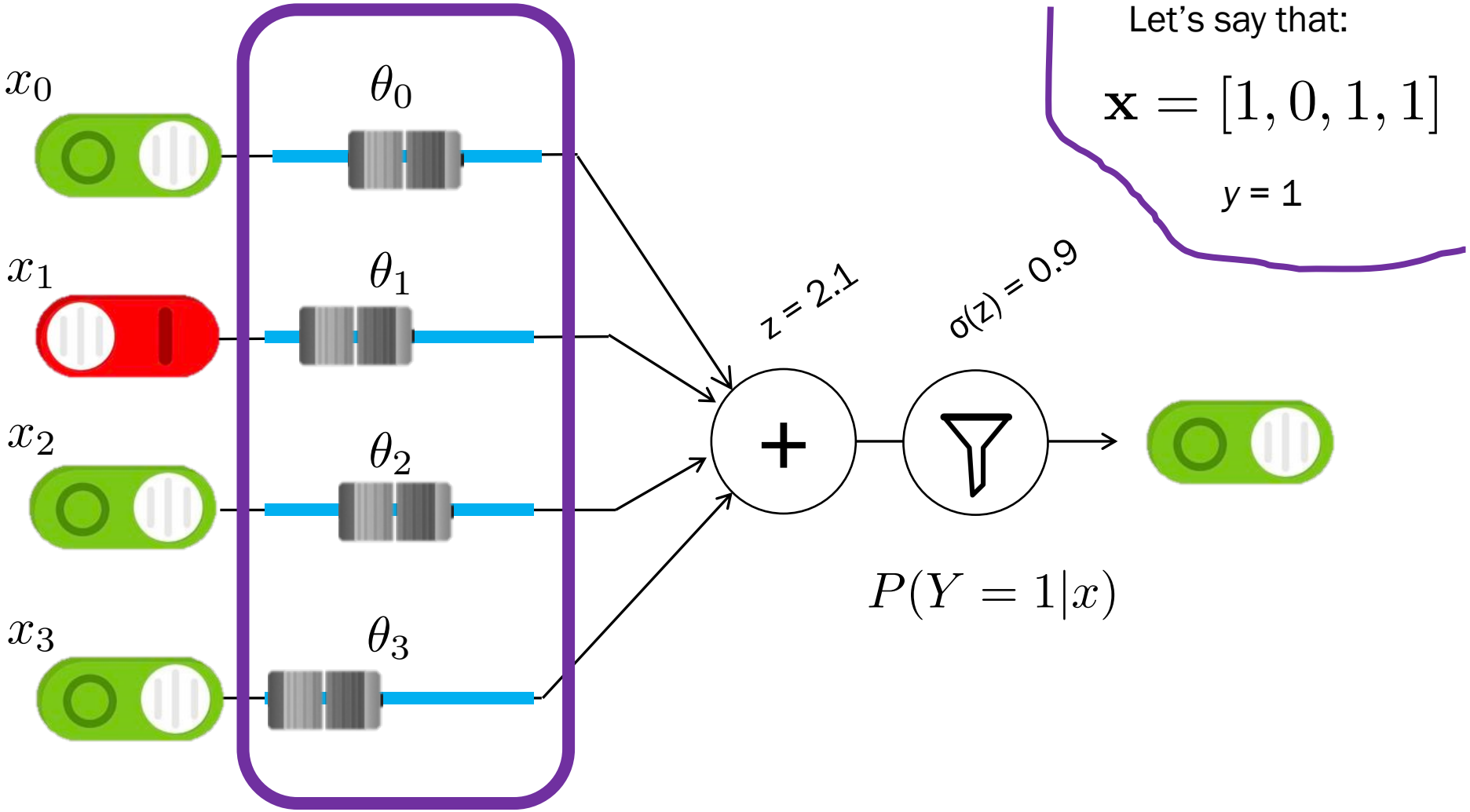
How Do We Learn Parameters?



$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\Big(\sum_i \theta_i x_i\Big) = 0.4$$

Data looks unlikely

How Do We Learn Parameters?



$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right) = 0.9$$

Data is much more likely!

Maximum Likelihood Estimation

Chose your parameter estimates

Parameter μ: 5 Parameter σ: 1.1

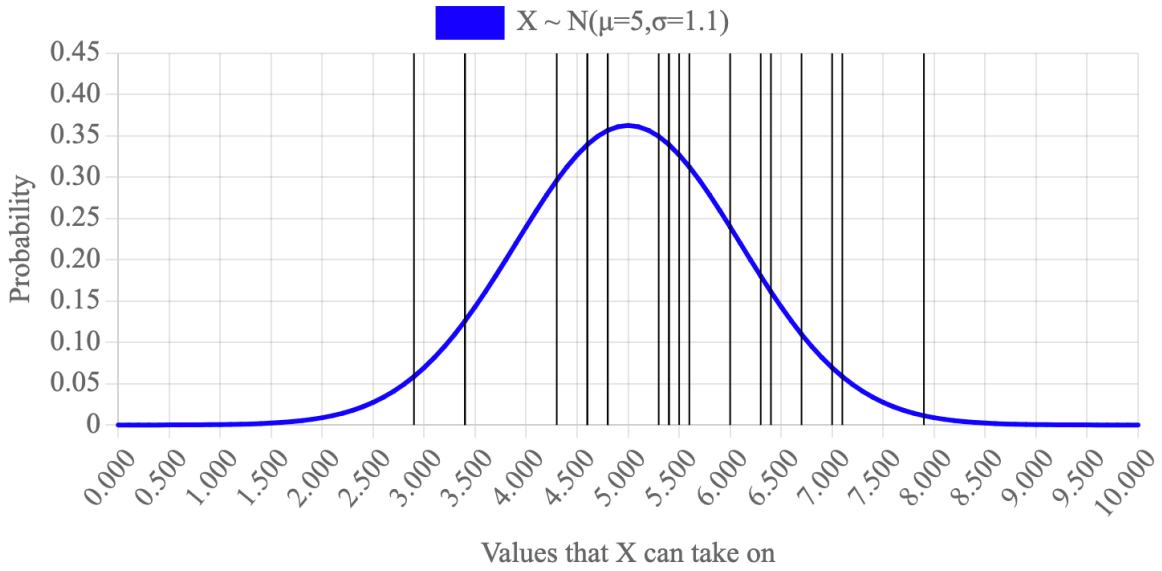
Likelihood of the data given your params

Likelihood: 5.204152095194613e-16

Log Likelihood: -314.1

Best Seen: -311.2

Your Gaussian



Pedagogy: show you the big picture,
then we can derive it!

Math for Logistic Regression

1 Make logistic regression assumption

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

Often call this $\hat{y}$

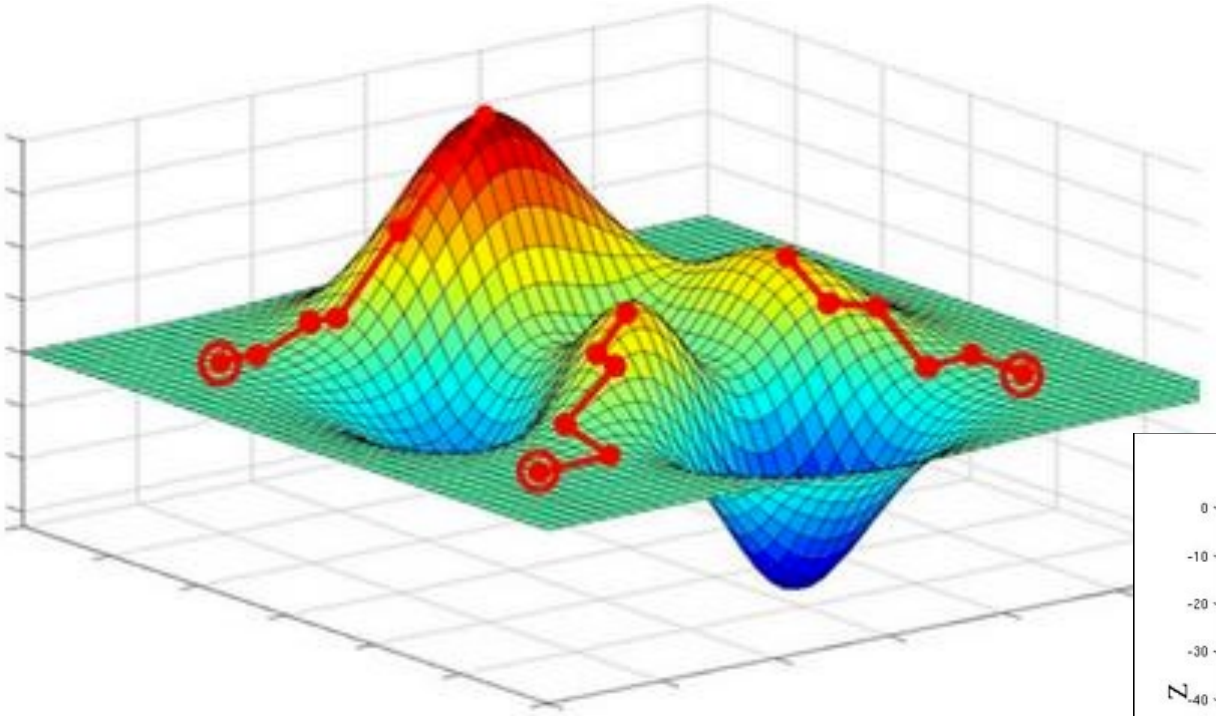
2 Calculate the log likelihood for all data

$$LL(\theta) = \sum_{i=0}^{n} y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$

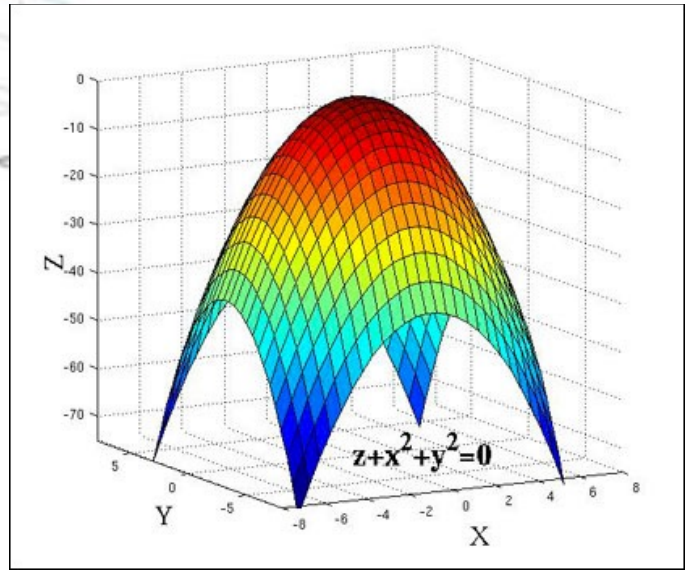
3 Get derivative of log likelihood with respect to thetas

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

Gradient Ascent



Logistic regression
LL function is
convex



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

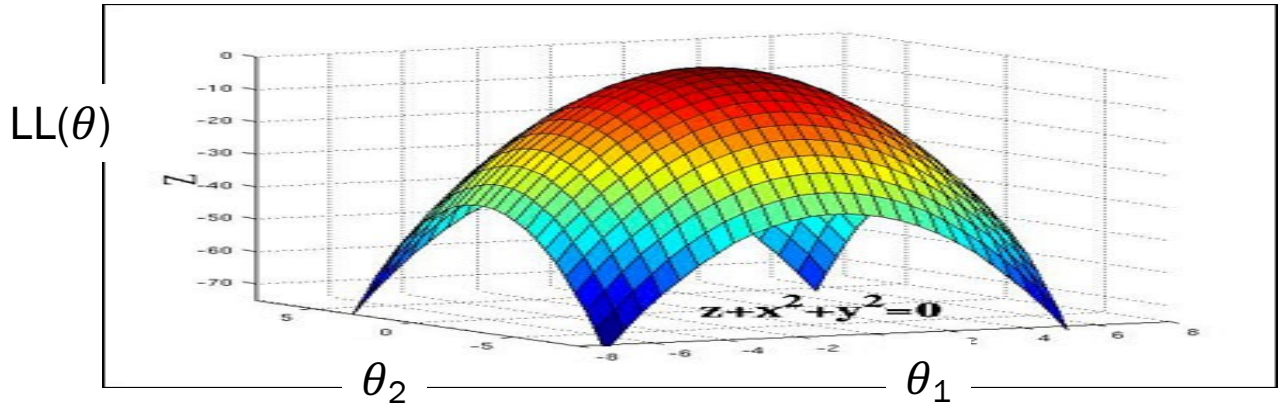
Gradient Ascent Step

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=0}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

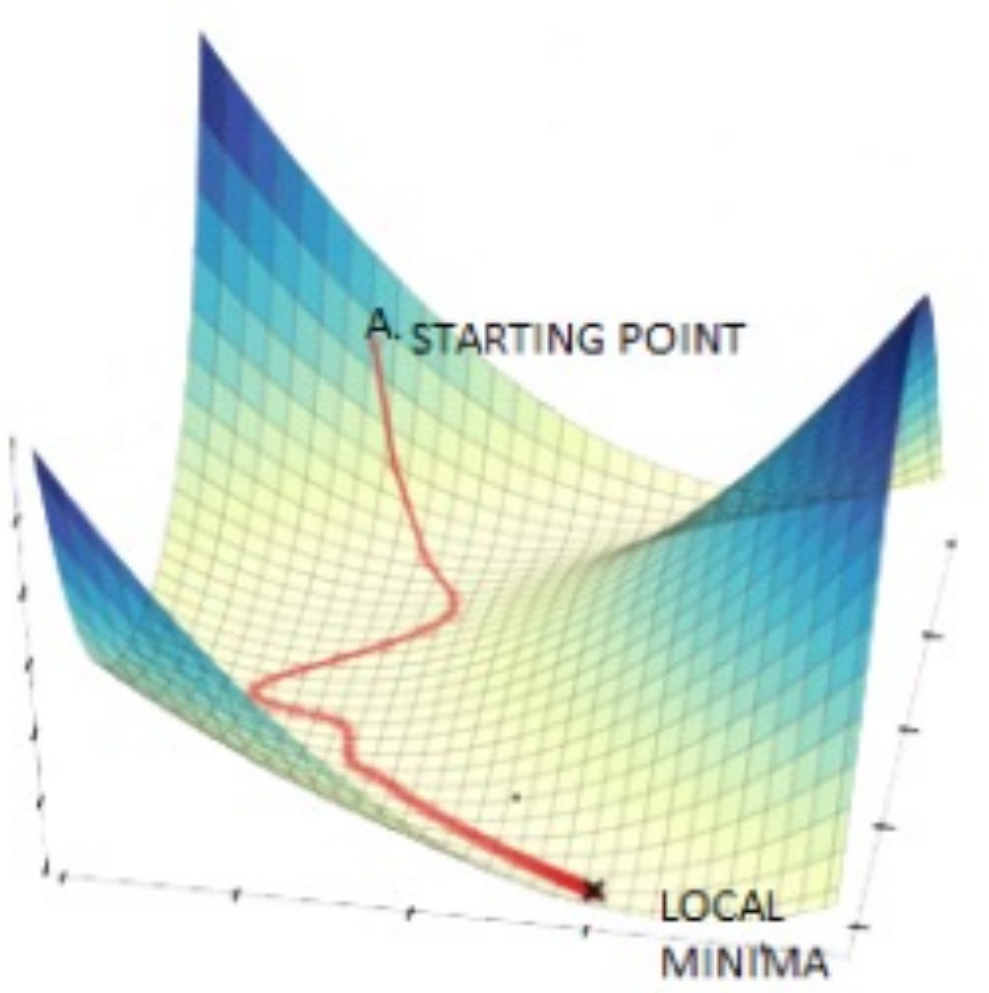
$$\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \frac{\partial LL(\theta^{\text{old}})}{\partial \theta_j^{\text{old}}}$$

$$= \theta_j^{\text{old}} + \eta \cdot \sum_{i=0}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

Do this for all thetas!



Gradient Decent



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

Gradient Descent with Negative LL

Assume some loss function with known derivative $\frac{\partial \text{ Loss}}{\partial \theta_j}$

$$\theta_j^{\text{new}} = \theta_j^{\text{old}} - \eta \cdot \frac{\partial \text{ Loss}}{\partial \theta_j}$$

Gradient Descent with Negative LL

Assume some loss function with known derivative $\frac{\partial \text{ Loss}}{\partial \theta_j}$

$$\theta_j^{\text{new}} = \theta_j^{\text{old}} - \eta \cdot \frac{\partial \text{ Loss}}{\partial \theta_j}$$

$$= \theta_j^{\text{old}} - \eta \cdot \frac{\partial \text{ Negative}LL}{\partial \theta_j}$$

$$= \theta_j^{\text{old}} + \eta \cdot \sum_{i=0}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

Gradient Descent with Negative LL

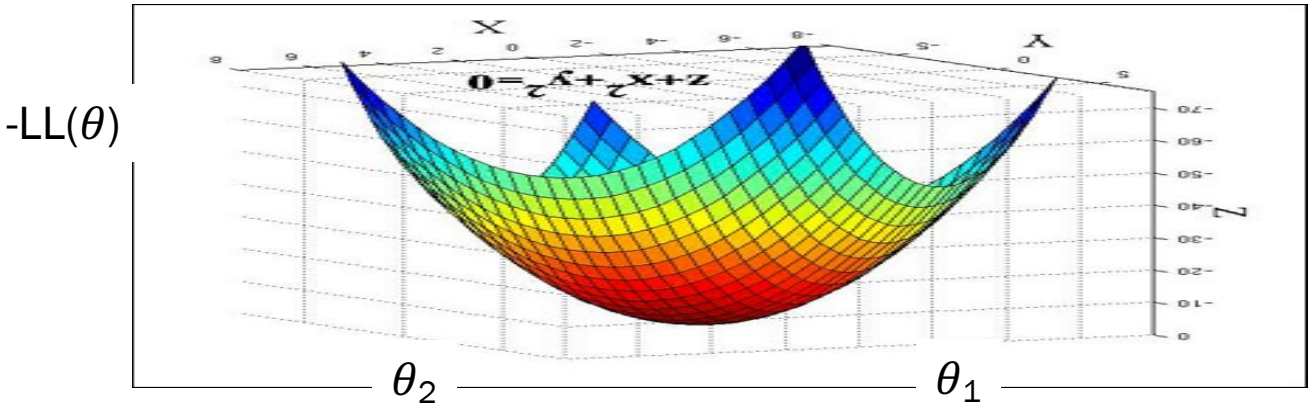
Assume some loss function with known derivative $\frac{\partial \text{ Loss}}{\partial \theta_j}$

$$\theta_j^{\text{new}} = \theta_j^{\text{old}} - \eta \cdot \frac{\partial \text{ Loss}}{\partial \theta_j}$$

...exactly the same

$$= \theta_j^{\text{old}} - \eta \cdot \frac{\partial \text{ Negative} LL}{\partial \theta_j}$$

$$= \theta_j^{\text{old}} + \eta \cdot \sum_{i=0}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$



What does this look like in code?

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

$$= \theta_j^{\text{old}} + \eta \cdot \sum_{i=0}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

Logistic Regression Training

Initialize: $\theta_j = 0$ **for all** $0 \leq j \leq m$

Calculate all θ_j

Logistic Regression Training

Initialize: $\theta_j = 0$ for all $0 \leq j \leq m$

Repeat many times:

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

Calculate all gradient[j]*'s based on data*

θ_j += η * gradient[j] for all $0 \leq j \leq m$

Logistic Regression Training

Initialize: $\theta_j = 0$ for all $0 \leq j \leq m$

Repeat many times:

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

For each training example (x, y):

For each parameter j:

Update gradient[j] *for current training example (x, y)*

θ_j += η * gradient[j] for all $0 \leq j \leq m$

Logistic Regression Training

Initialize: $\theta_j = 0$ for all $0 \le j \le m$

Repeat many times:

gradient[j] = 0 for all $0 \le j \le m$

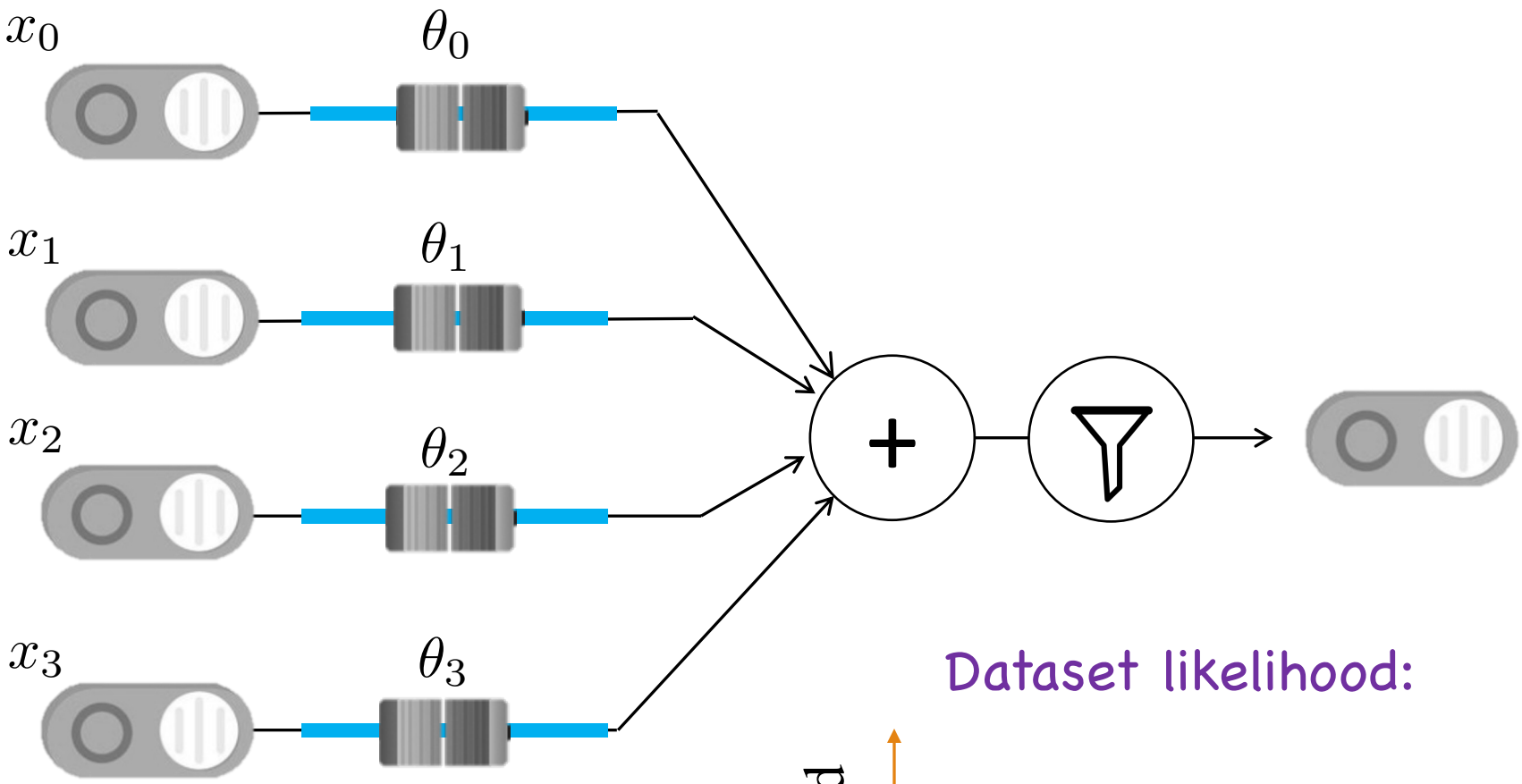
For each training example (x, y):

For each parameter j:

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

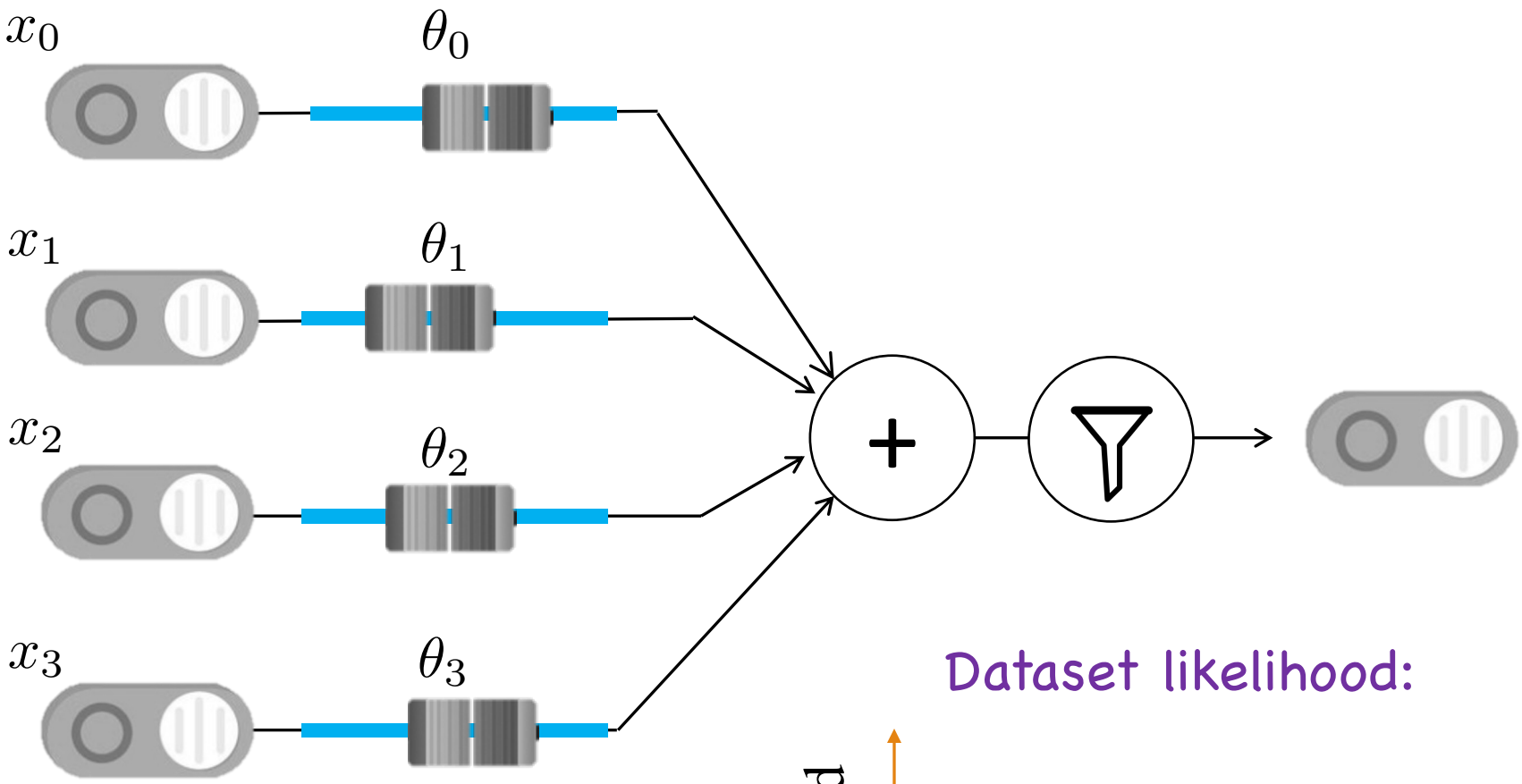
θ_j += η * gradient[j] for all $0 \le j \le m$

Training

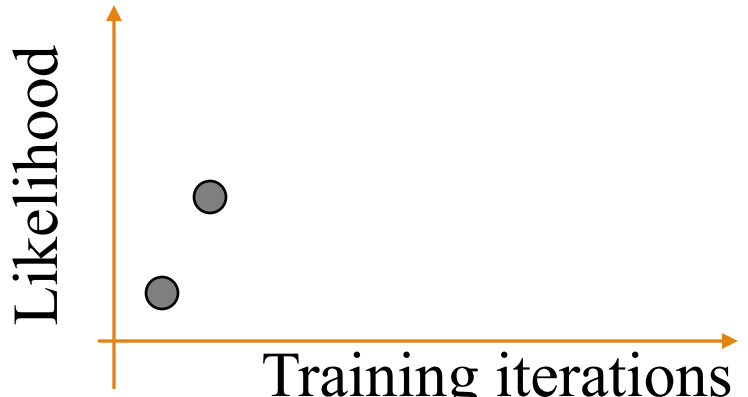


Dataset likelihood:

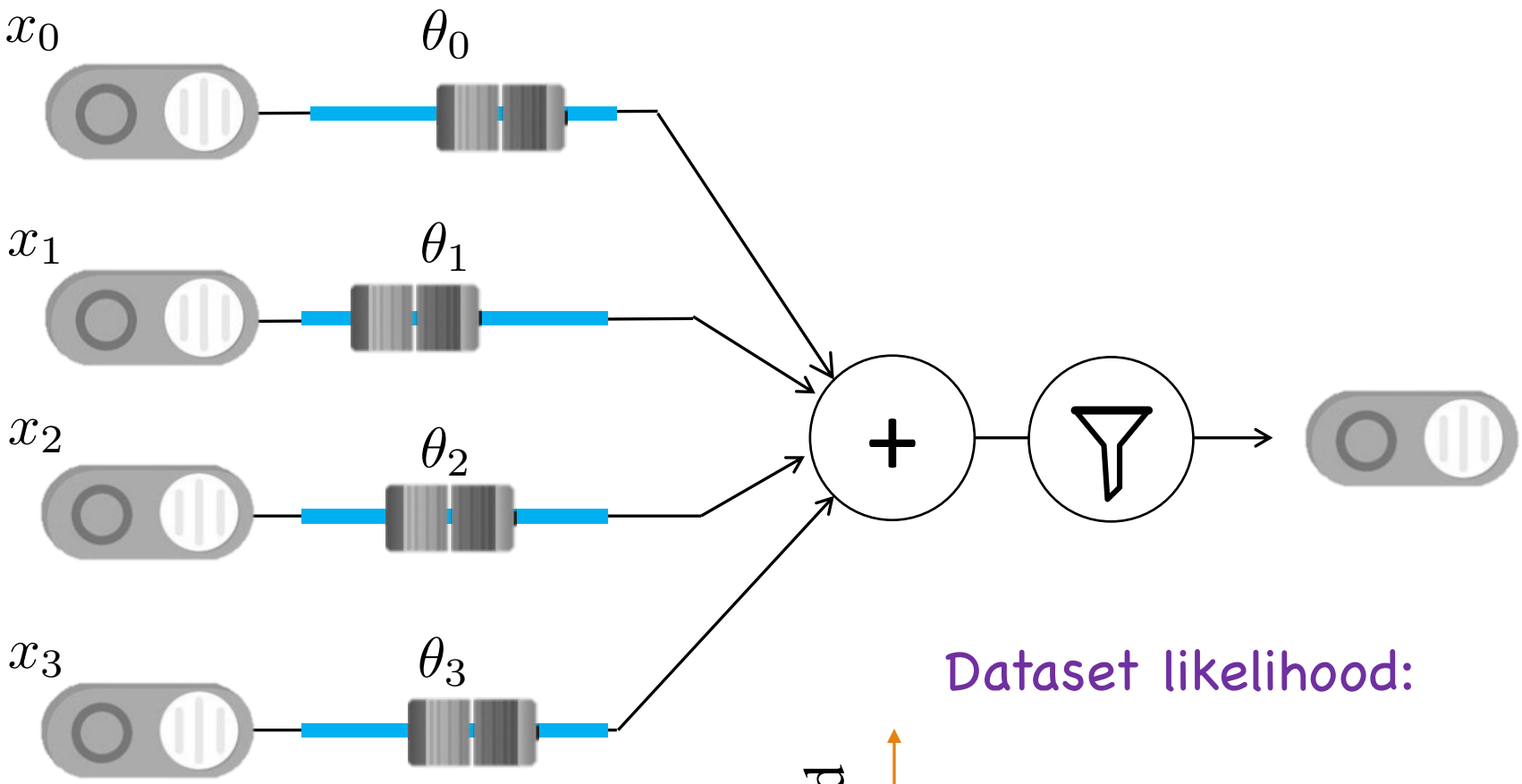
Training



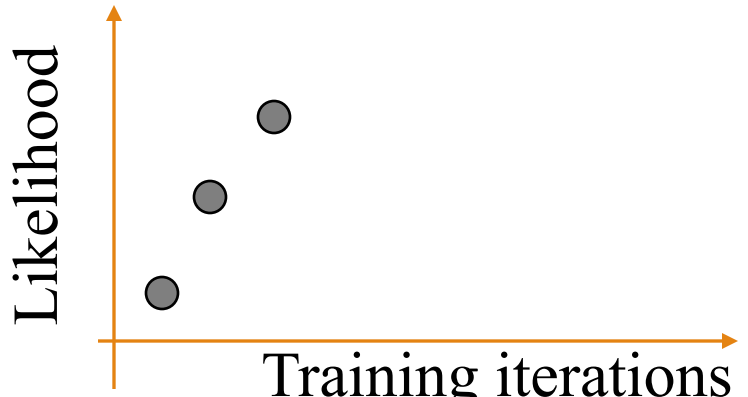
Dataset likelihood:



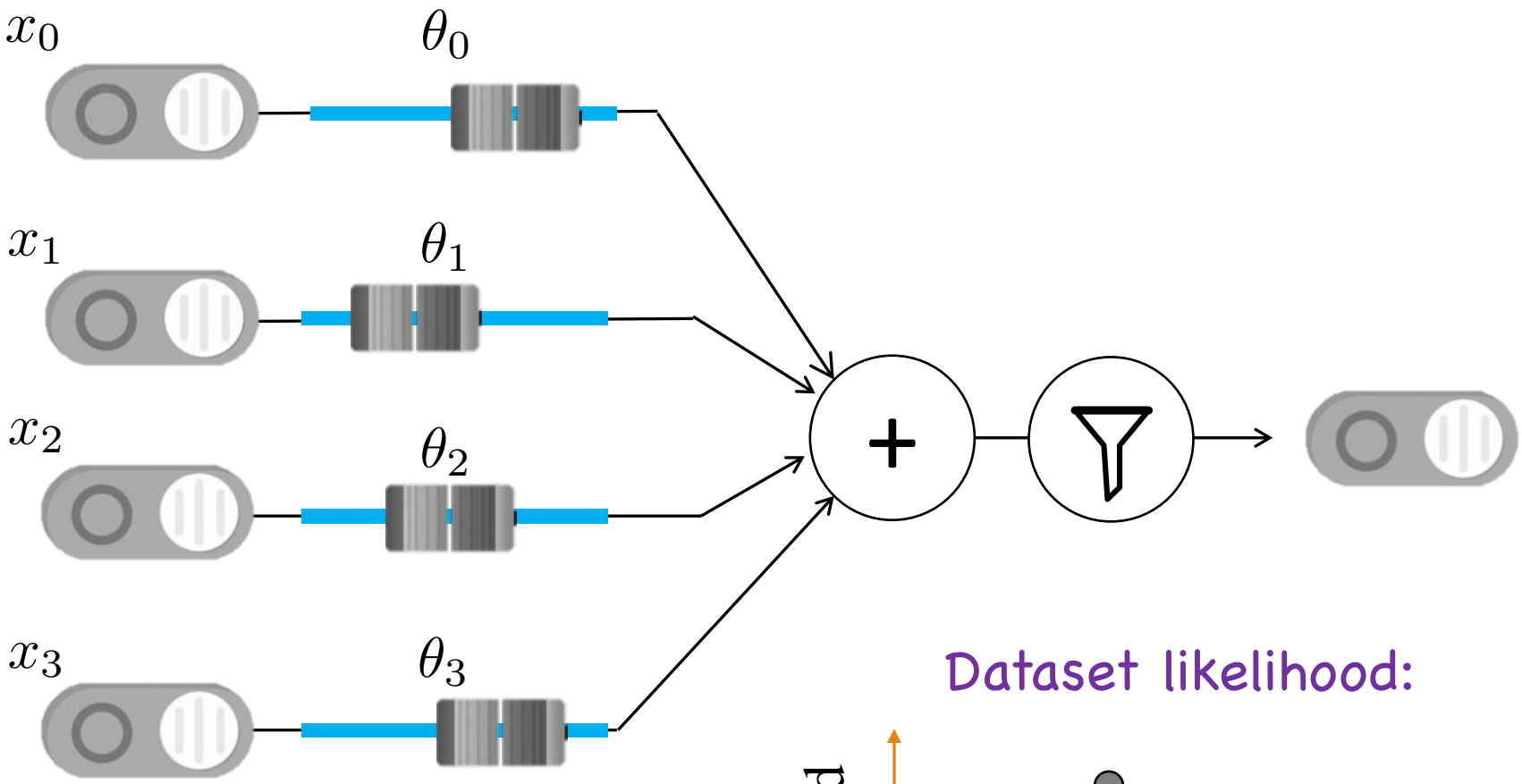
Training



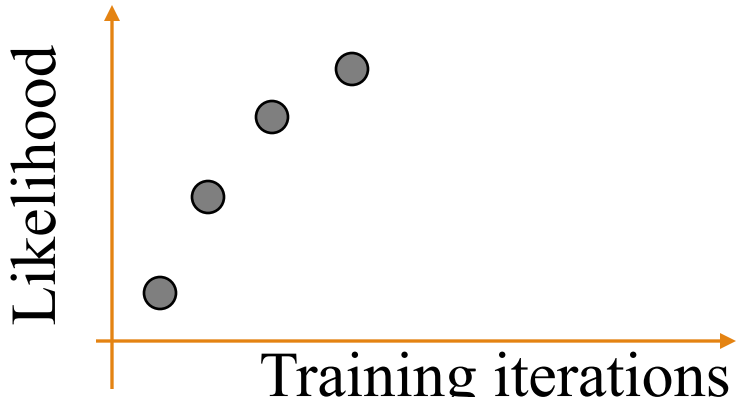
Dataset likelihood:



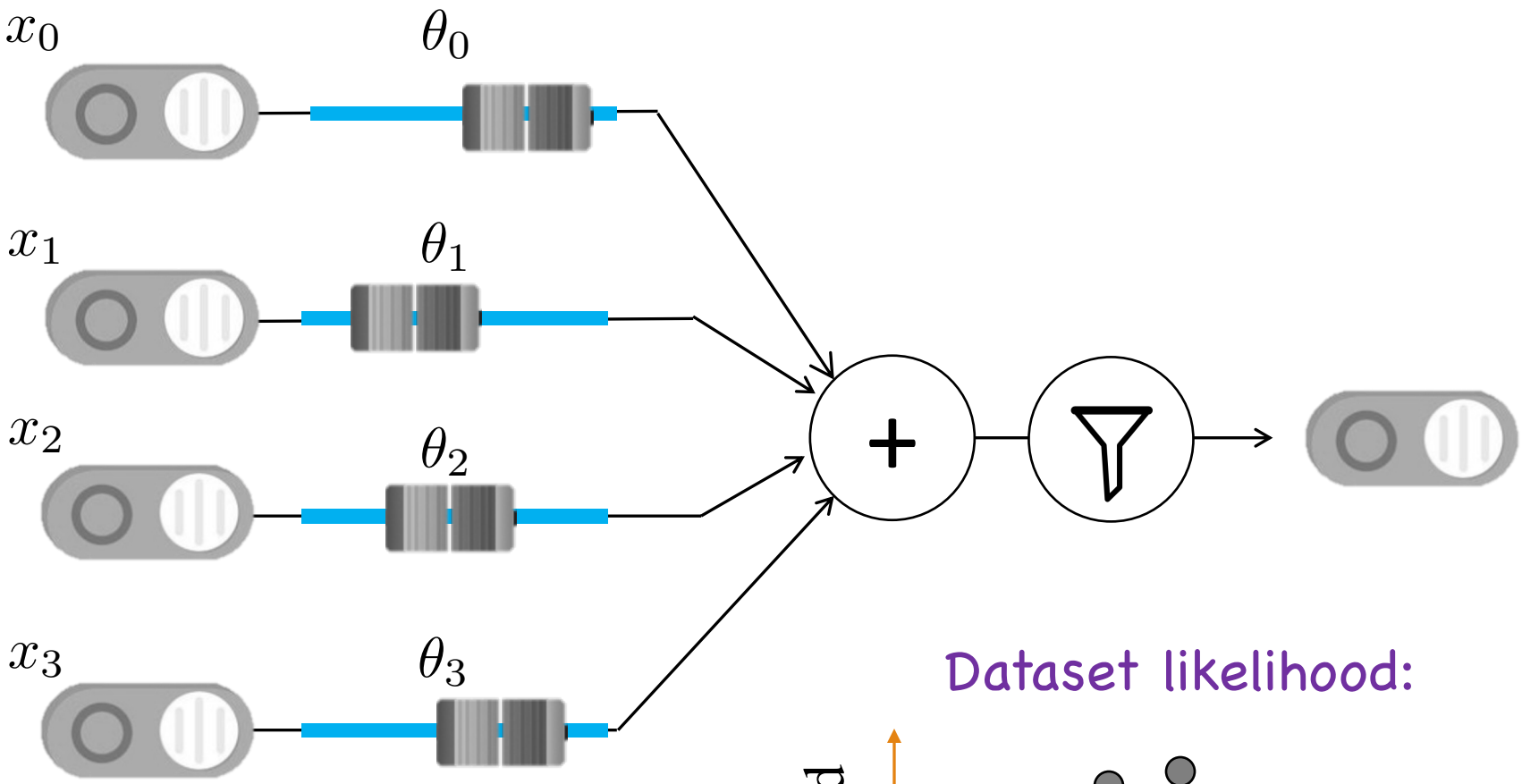
Training



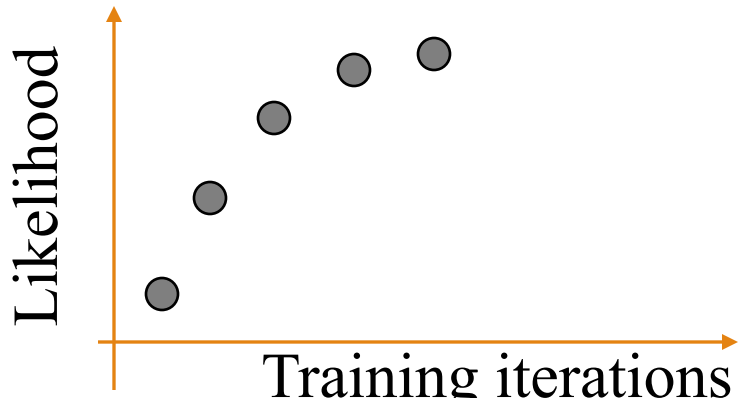
Dataset likelihood:



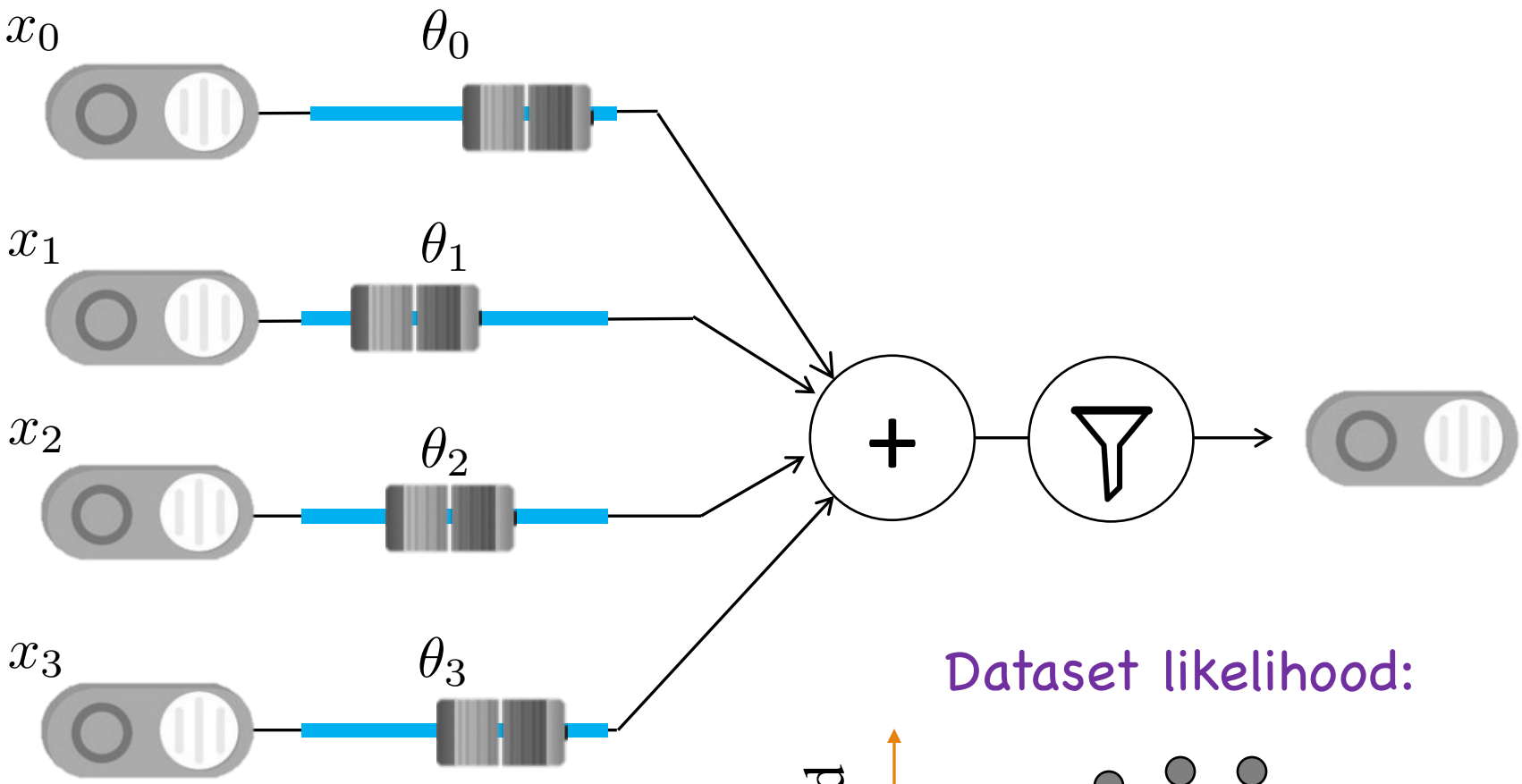
Training



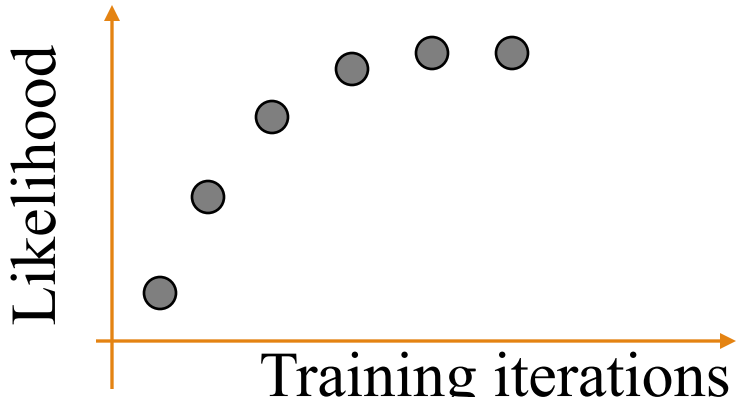
Dataset likelihood:



Training



Dataset likelihood:

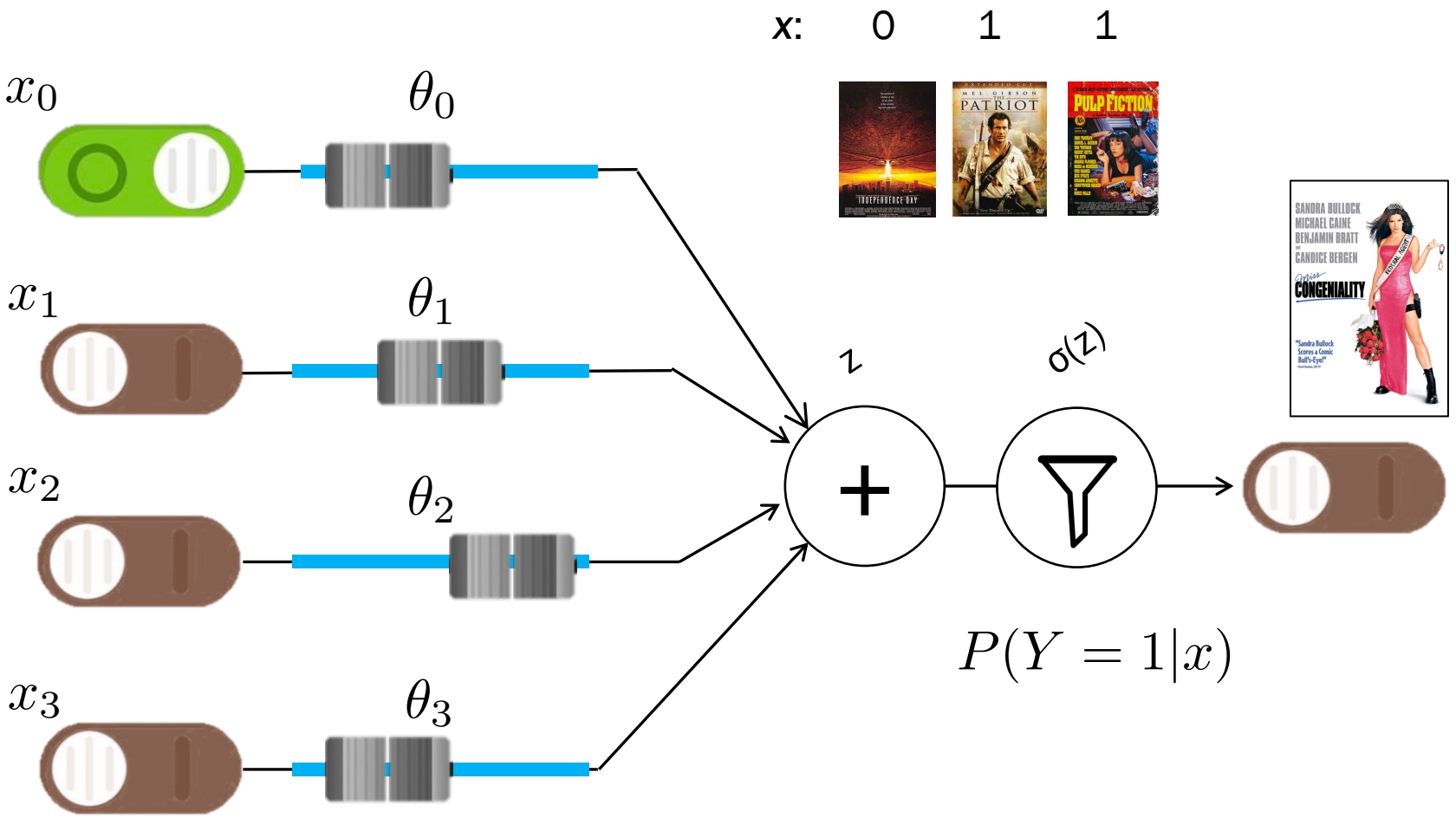


Don’t forget:

x_j is j-th input variable and $x_0 = 1$.

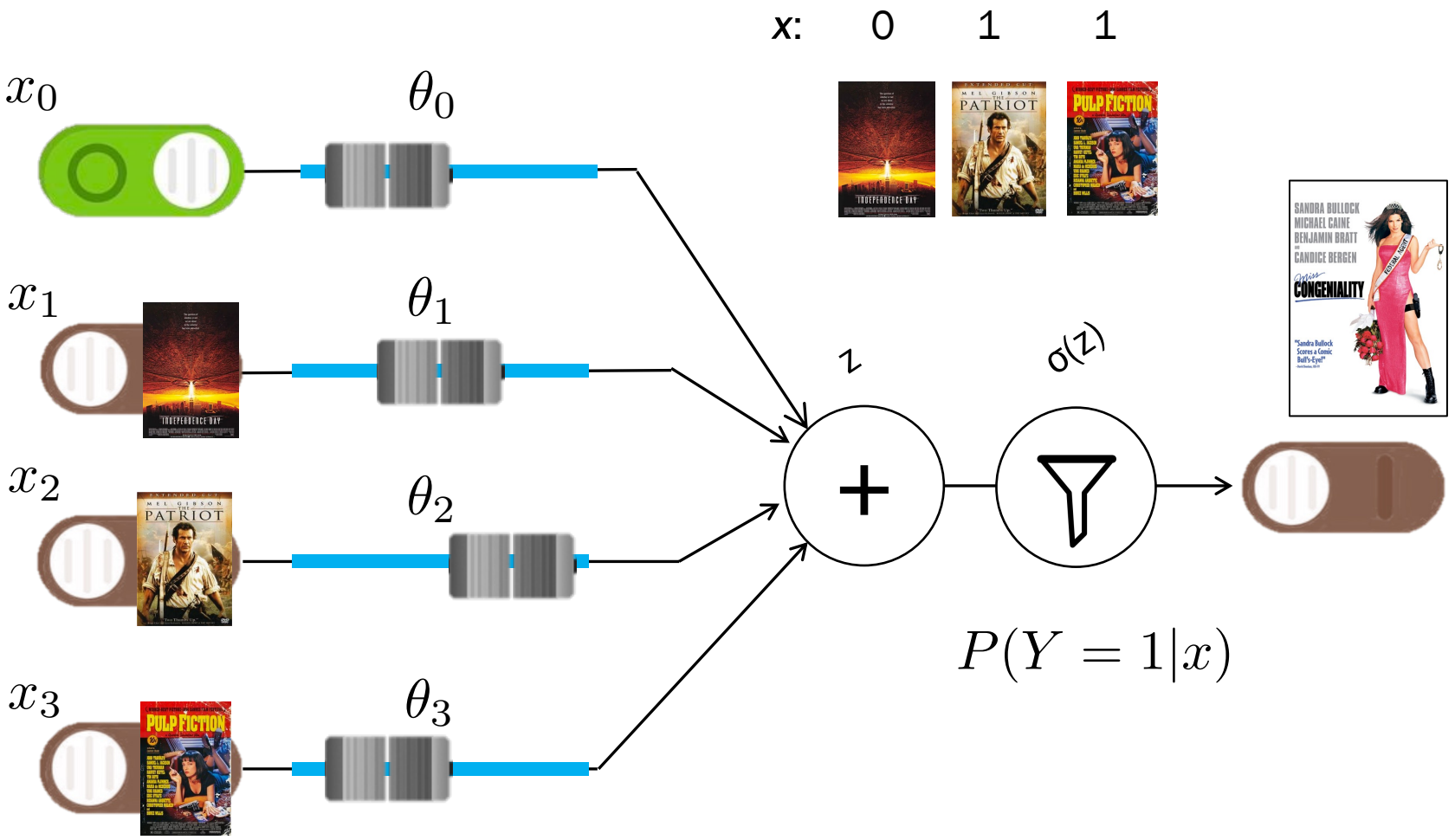
Allows for θ_0 to be an intercept.

Prediction



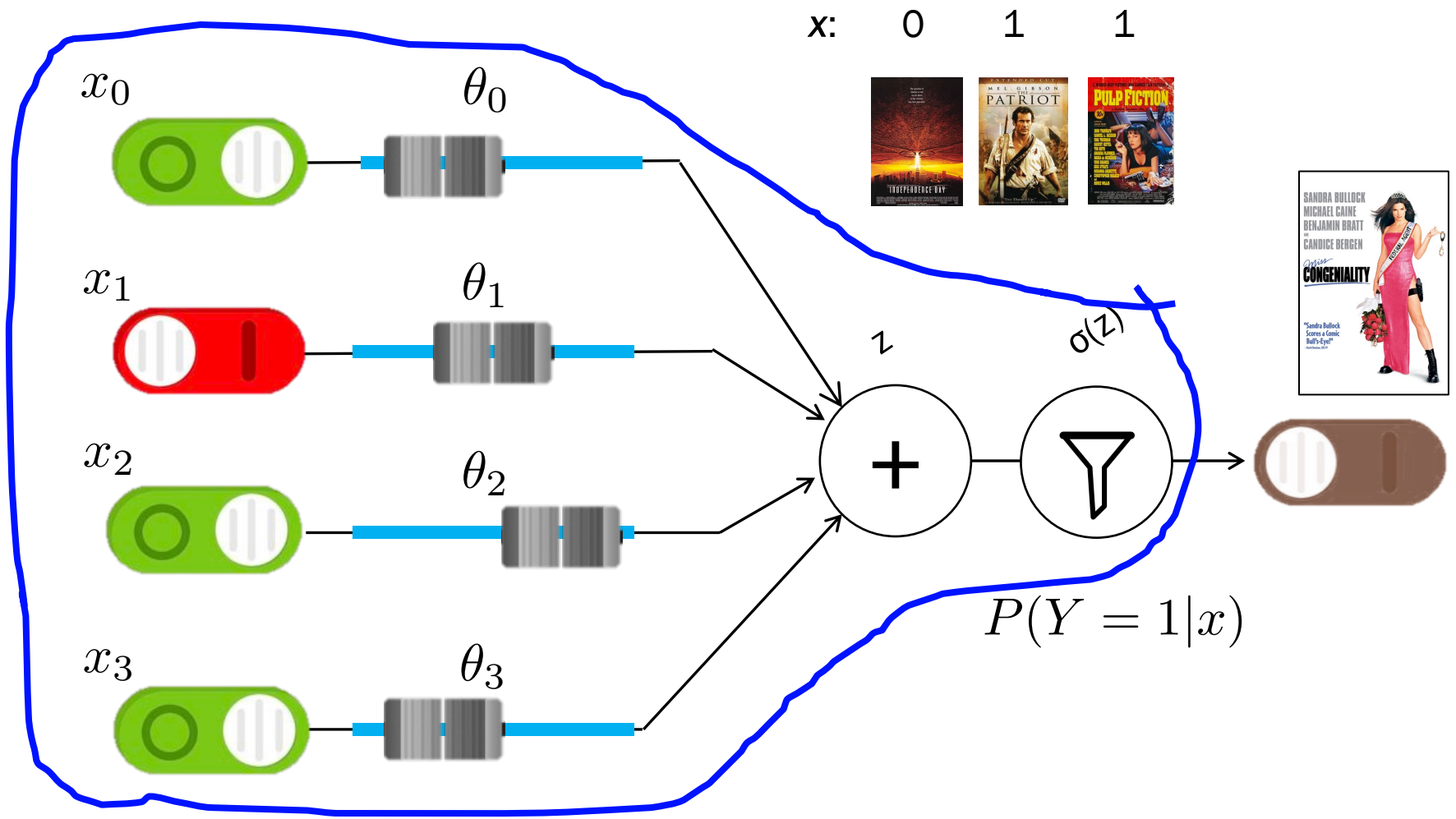
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Prediction



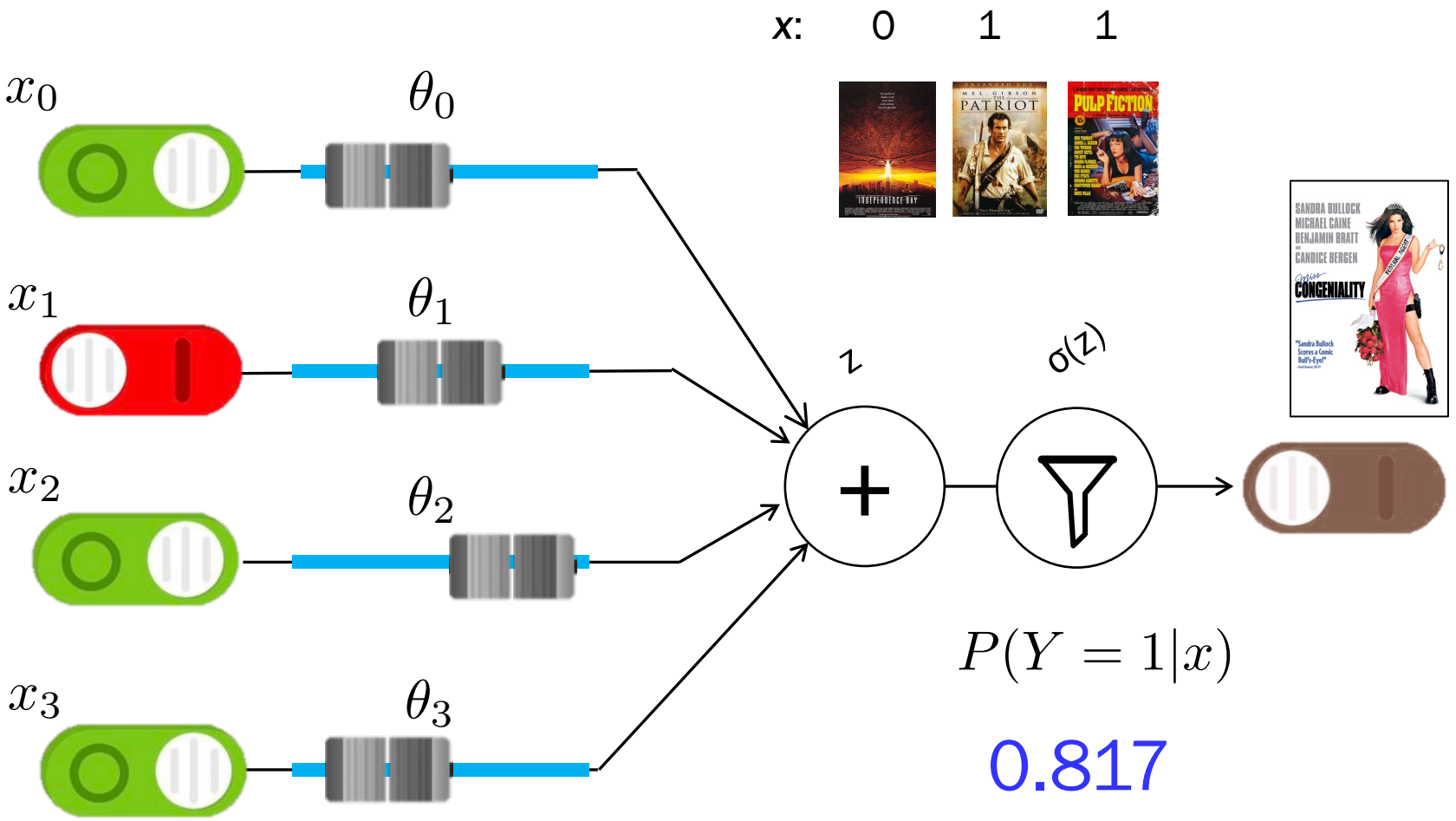
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Prediction



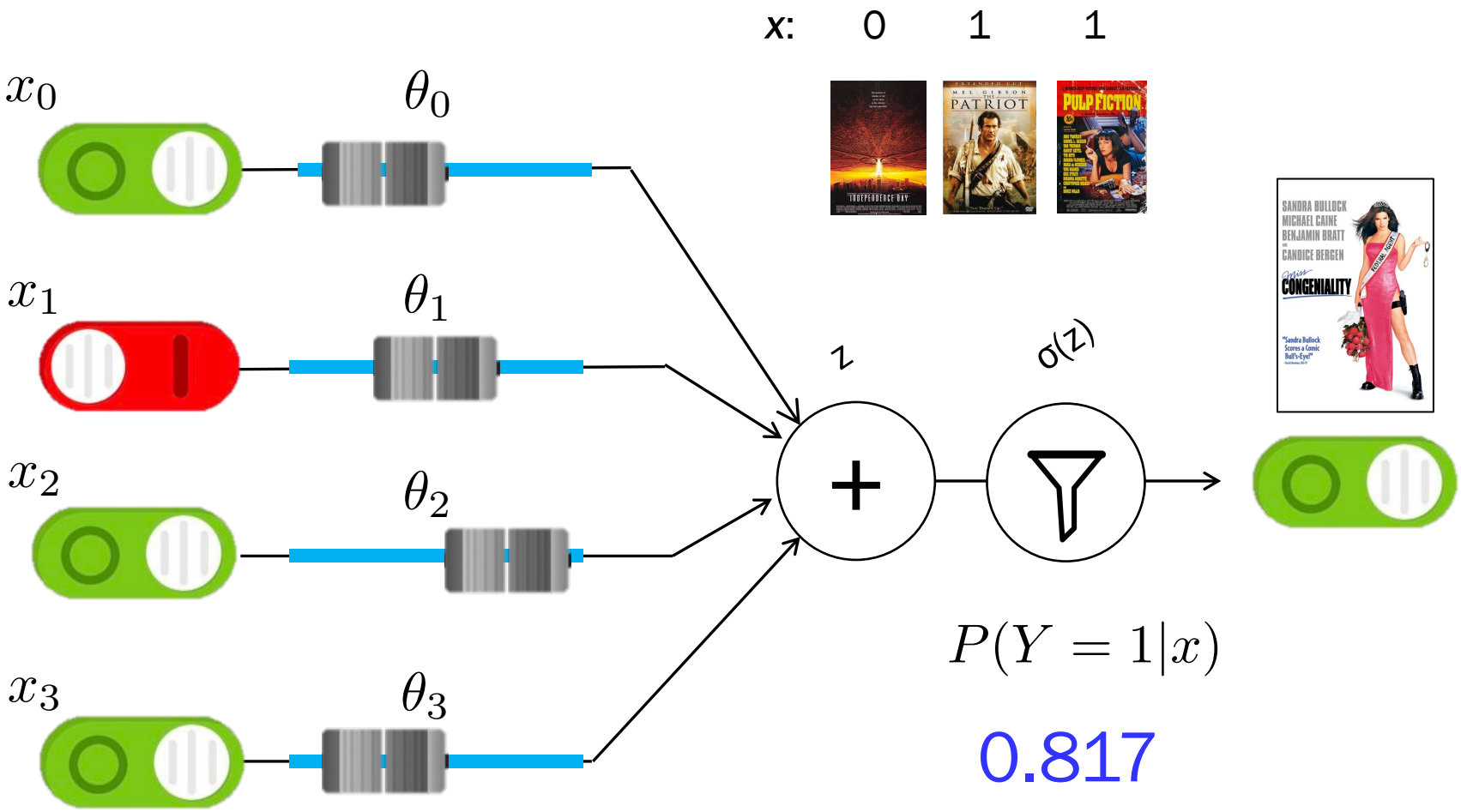
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Prediction



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

Prediction



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

Chapter 2: How Come?

Logistic Regression

1 Make logistic regression assumption

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

Often call this $\hat{y}$

2 Calculate the log probability for all data

$$LL(\theta) = \sum_{i=0}^{n} y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$

3 Get derivative of log probability with respect to thetas

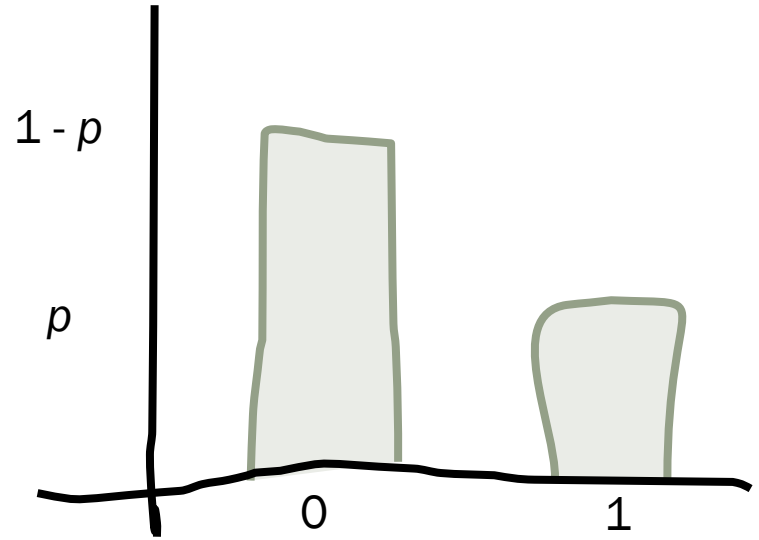
$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

How did we get that LL function?

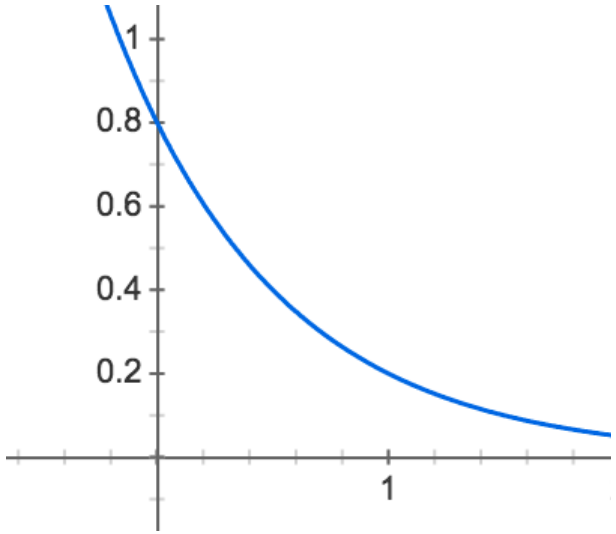
Recall: PMF of Bernoulli

- Y ~ Ber(p)
- Probability mass function: $P(Y = y)$

PMF of Bernoulli



PMF of Bernoulli (p = 0.2)



$$P(Y = y) = p^y(1-p)^{1-y}$$

$$P(Y = y) = 0.2^y(0.8)^{1-y}$$

Log Probability of Data

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

Implies

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

For IID data

$$L(\theta) = \prod_{i=1}^{n} P(Y = y^{(i)}|X = \mathbf{x}^{(i)})$$
$$= \prod_{i=1}^{n} \sigma(\theta^T \mathbf{x}^{(i)})^{y^{(i)}} \cdot \left[1 - \sigma(\theta^T \mathbf{x}^{(i)})\right]^{(1-y^{(i)})}$$

Take the log

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

How did we get that gradient?

Sigmoid has a Beautiful Slope

True fact about
sigmoid functions

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

Sigmoid has a Beautiful Slope

$$\frac{\partial}{\partial \theta_j}\sigma(\theta^T x)? \qquad \frac{\partial}{\partial z}\sigma(z) = \sigma(z)[1-\sigma(z)]$$

---

where $z = \theta^T x$

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

Chain rule!

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

Plug and chug

Sigmoid, you should be a ski hill

Sigmoid has a Beautiful Slope

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

$$\frac{\partial \hat{y}}{\partial \theta_j} = \sigma(\theta^T x)[1 - \sigma(\theta^T x)]x_j$$

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

[pedagogical pause]

ARE YOU READY???

I think I'm Ready…

$$\frac{\partial LL(\theta)}{\partial \theta_j}$$

Where

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

This is Sparta!!!!!

This is ~~Sparta~~!!!!!

↑

Stanford

Think About Only One Training Instance

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

We only need to calculate the gradient for one training example!

$$\frac{\partial}{\partial x} \sum_i f(x, i) = \sum_i \frac{\partial}{\partial x} f(x, i)$$

We will pretend we only have one example

$$LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}]$$

We can sum up the gradients of each example to get the correct answer

First, imagine only one example

$$LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}]$$

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

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \frac{\partial LL(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \theta_j}$$

CHAIN RULZ!

$$= \frac{\partial LL(\theta)}{\partial \hat{y}} \hat{y}(1 - \hat{y}) x_j$$

Already did that one

$$= \left[\frac{y}{\hat{y}} - \frac{1 - y}{1 - \hat{y}}\right] \hat{y}(1 - \hat{y}) x_j$$

Derive this one

$$= (y - \hat{y}) x_j$$

Simplify

Now, all the data

$$LL(\theta) = \sum_{i=0}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log[1 - \hat{y}^{(i)}]$$

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

Derivative of sum...

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=0}^{n} \frac{\partial}{\partial \theta_j} \left[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log[1 - \hat{y}^{(i)}] \right]$$

$$= \sum_{i=0}^{n} [y^{(i)} - \hat{y}^{(i)}] x_j^{(i)}$$

See last slide

$$= \sum_{i=0}^{n} [y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)})] x_j^{(i)}$$

Some people don't like hats...

Now, all the data

$$\frac{\partial LL(\theta)}{\partial \theta_j}$$

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

Logistic Regression

1 Make logistic regression assumption

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

2 Calculate the log probability for all data

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

3 Get derivative of log probability with respect to thetas

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

The Hard Way

$$LL(\theta) = y \log \sigma(\theta^T \mathbf{x}) + (1 - y) \log[1 - \sigma(\theta^T \mathbf{x})]$$

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j} y \log \sigma(\theta^T \mathbf{x}) + \frac{\partial}{\partial \theta_j} (1 - y) \log[1 - \sigma(\theta^T \mathbf{x}]$$

$$= \left[\frac{y}{\sigma(\theta^T x)} - \frac{1 - y}{1 - \sigma(\theta^T x)} \right] \frac{\partial}{\partial \theta_j} \sigma(\theta^T x)$$

$$= \left[\frac{y}{\sigma(\theta^T x)} - \frac{1 - y}{1 - \sigma(\theta^T x)} \right] \frac{\partial}{\partial \theta_j} \sigma(\theta^T x)$$

$$= \left[\frac{y - \sigma(\theta^T x)}{\sigma(\theta^T x)[1 - \sigma(\theta^T x)]} \right] \sigma(\theta^T x)[1 - \sigma(\theta^T x)] x_j$$

$$= \left[y - \sigma(\theta^T x) \right] x_j$$

Phew!

Chapter 3: Philosophy (if time)

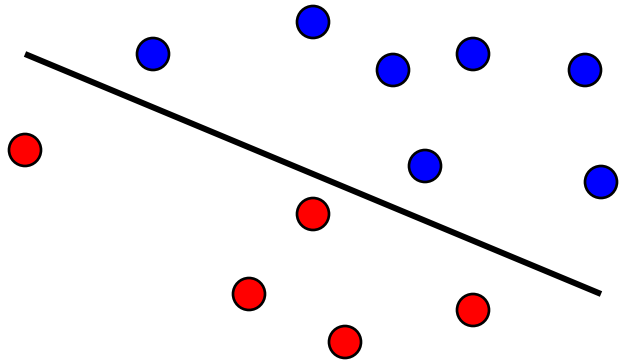
Choosing an Algorithm?

Many trade-offs in choosing learning algorithm

- Continuous input variables
  - Logistic Regression easily deals with continuous inputs
  - Naive Bayes needs to use some parametric form for continuous inputs (e.g., Gaussian) or "discretize" continuous values into ranges (e.g., temperature in range: <50, 50-60, 60-70, >70)
- Discrete input variables
  - Naive Bayes naturally handles multi-valued discrete features by using multinomial distribution for $P(X_i \mid Y)$
  - Logistic Regression requires some sort of representation of multi-valued discrete data (e.g., one hot vector)
  - Say $X_i \in \{A, B, C\}$. Not necessarily a good idea to encode X_i as taking on input values 1, 2, or 3 corresponding to A, B, or C.

Discrimination Intuition

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



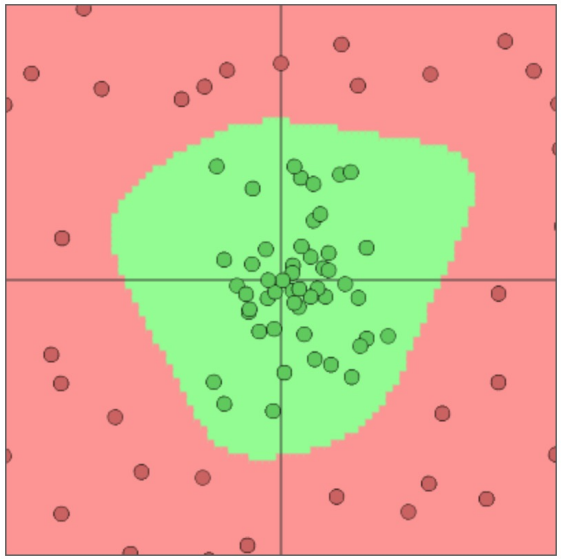
$$\theta^T \mathbf{x} = 0$$

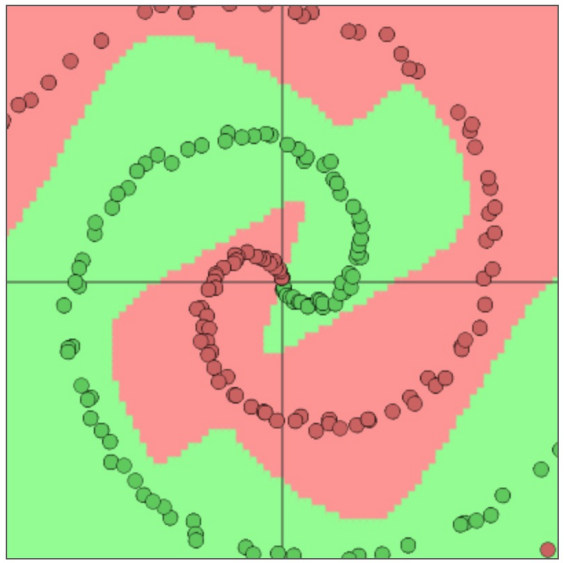
$$\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_m x_m = 0$$

- We call such data (or the functions generating the data) "**linearly separable**"
- **Naïve bayes is linear too** as there is no interaction between different features.

Some Data Not Linearly Seperable

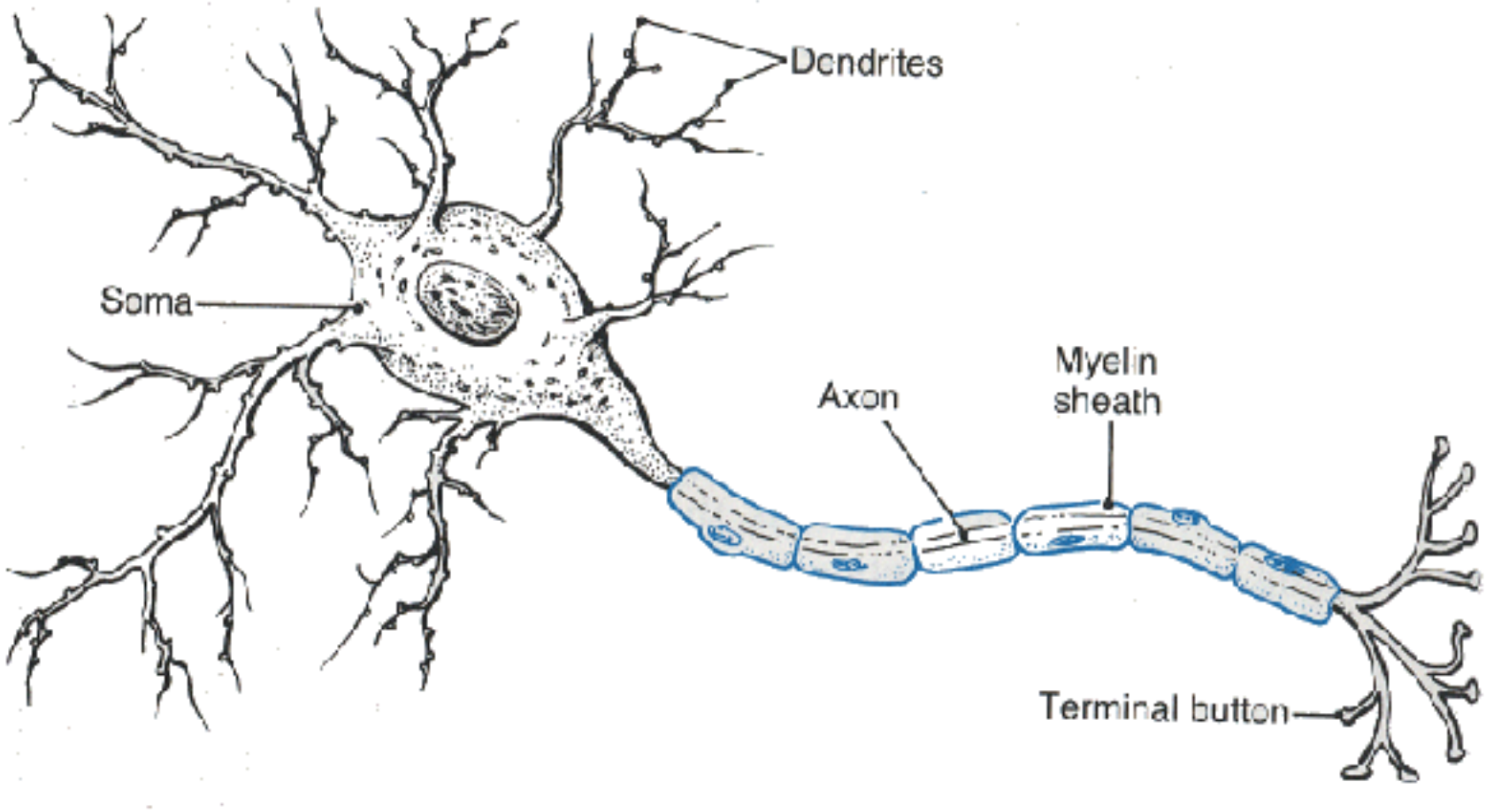
Some data sets/functions are not separable



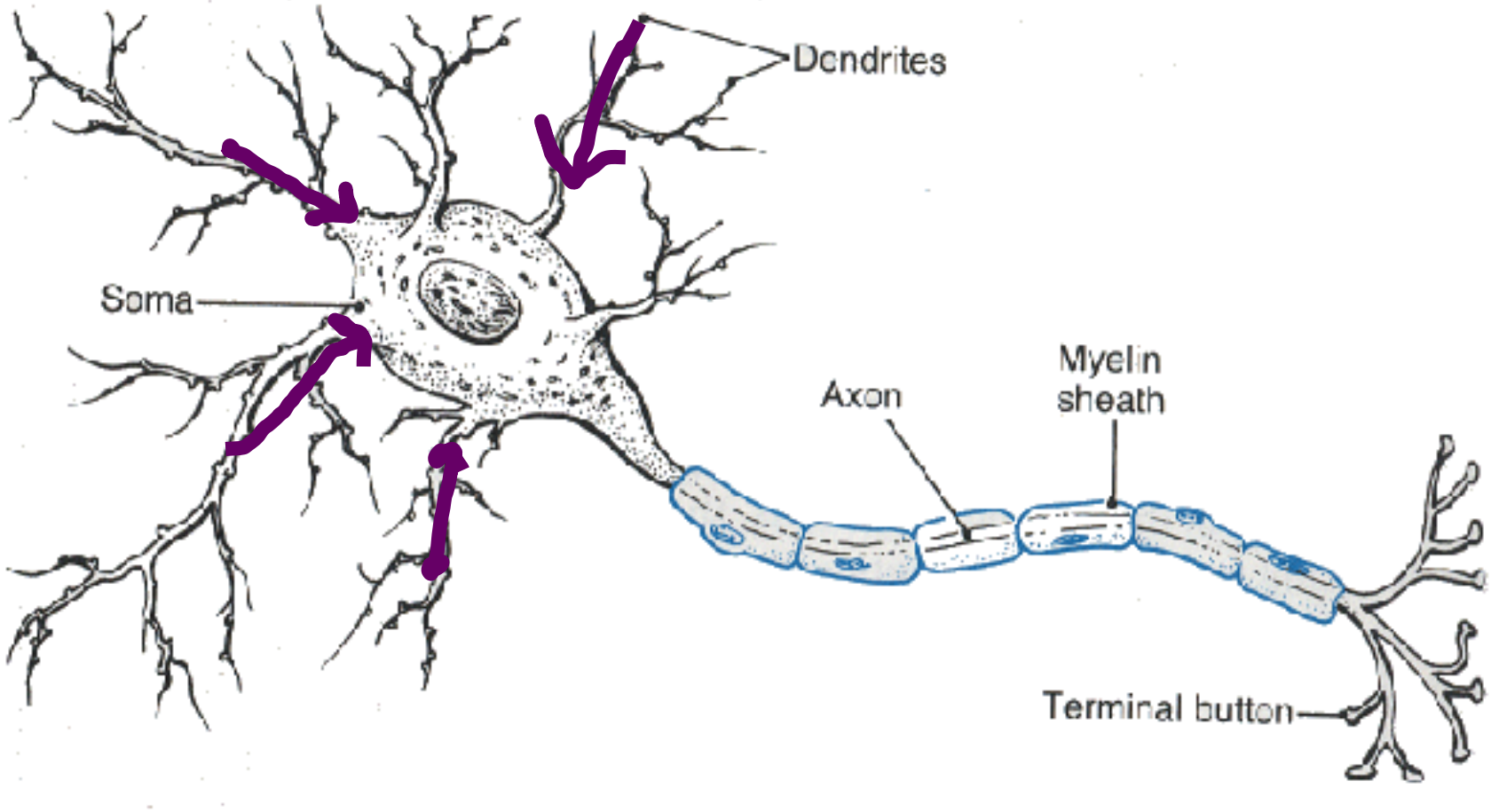


- Not possible to draw a line that successfully separates all the $y = 1$ points (green) from the $y = 0$ points (red)
- Despite this fact, logistic regression and Naive Bayes still often work well in practice

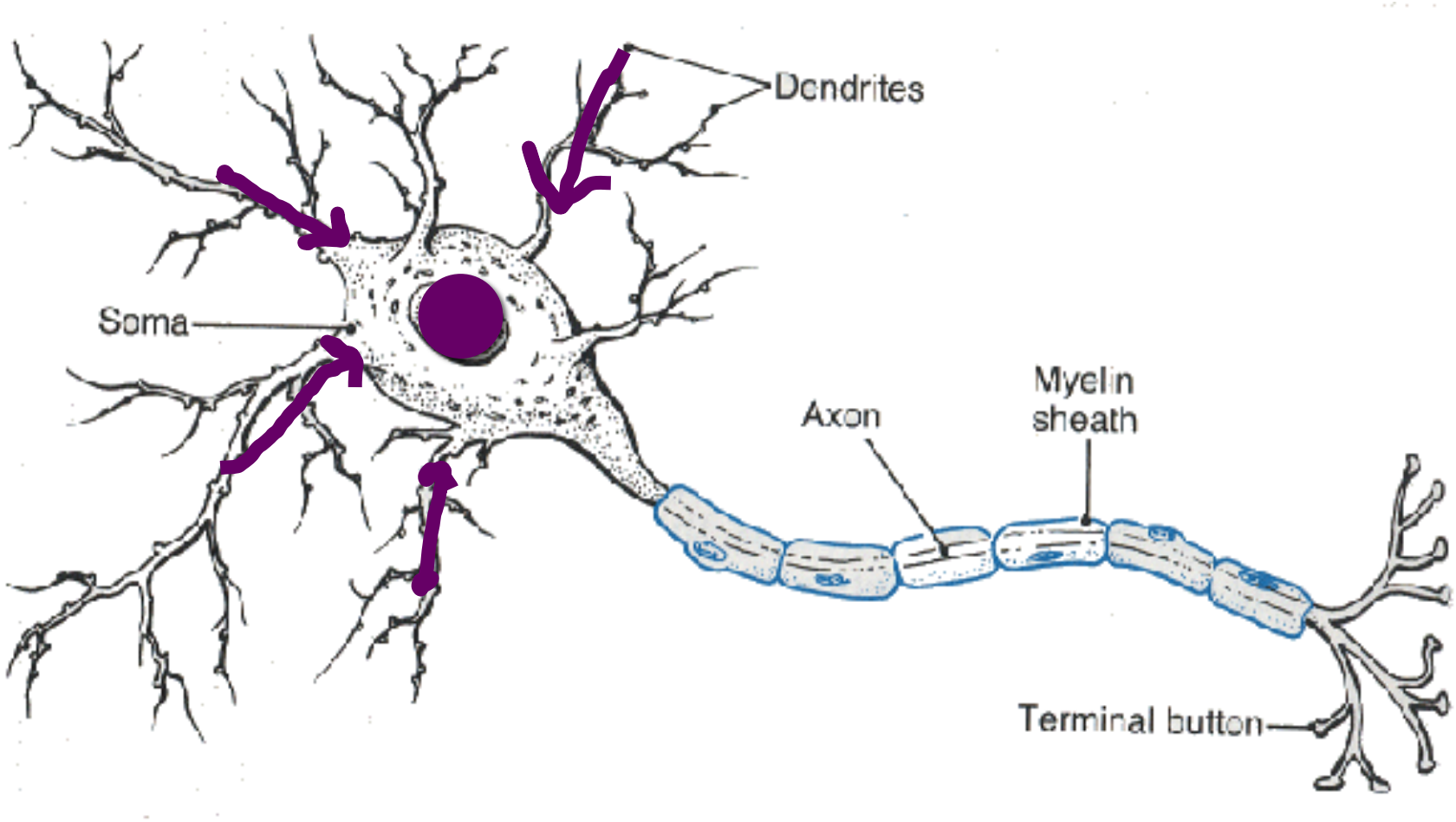
Neuron



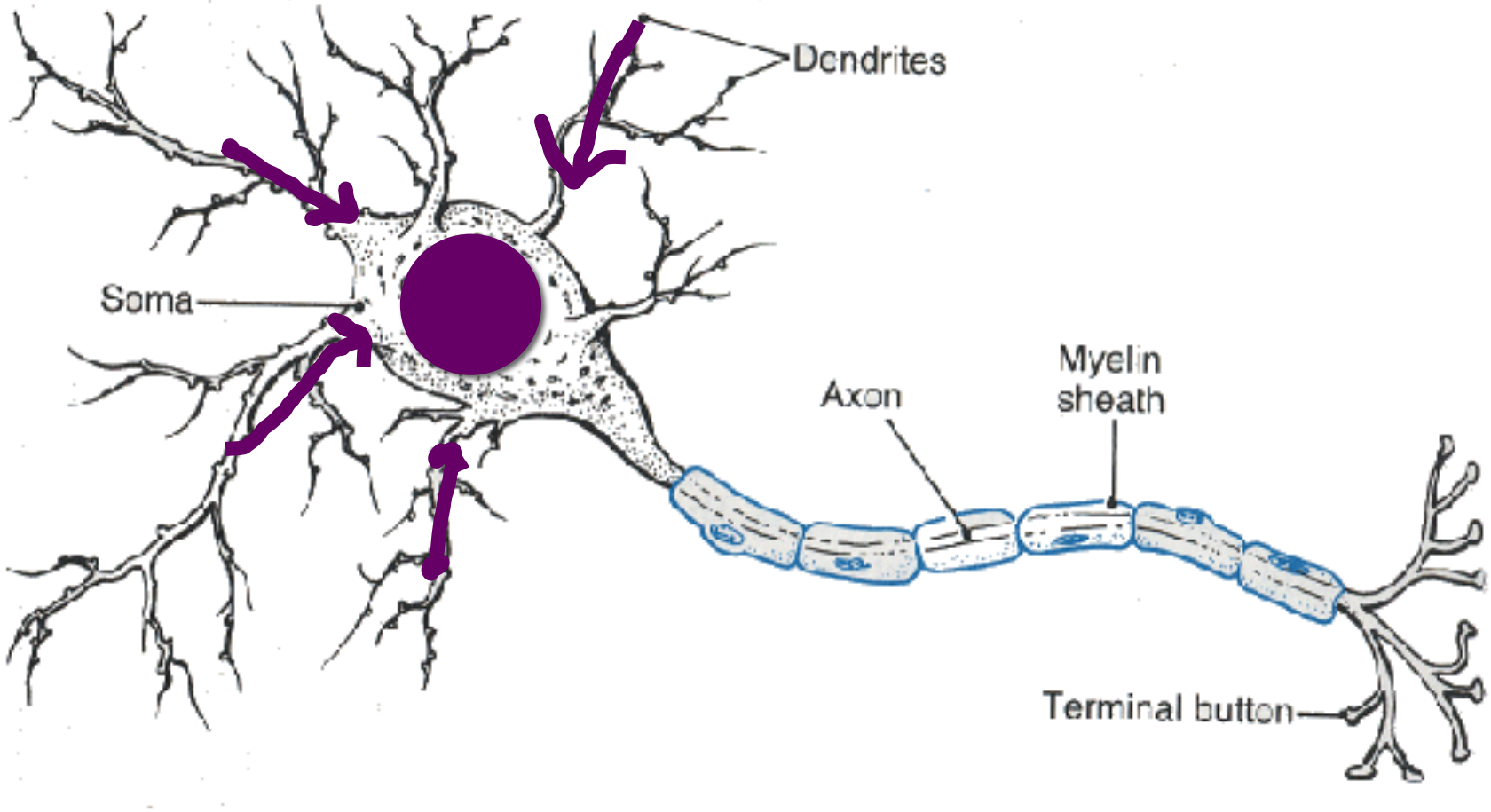
Neuron



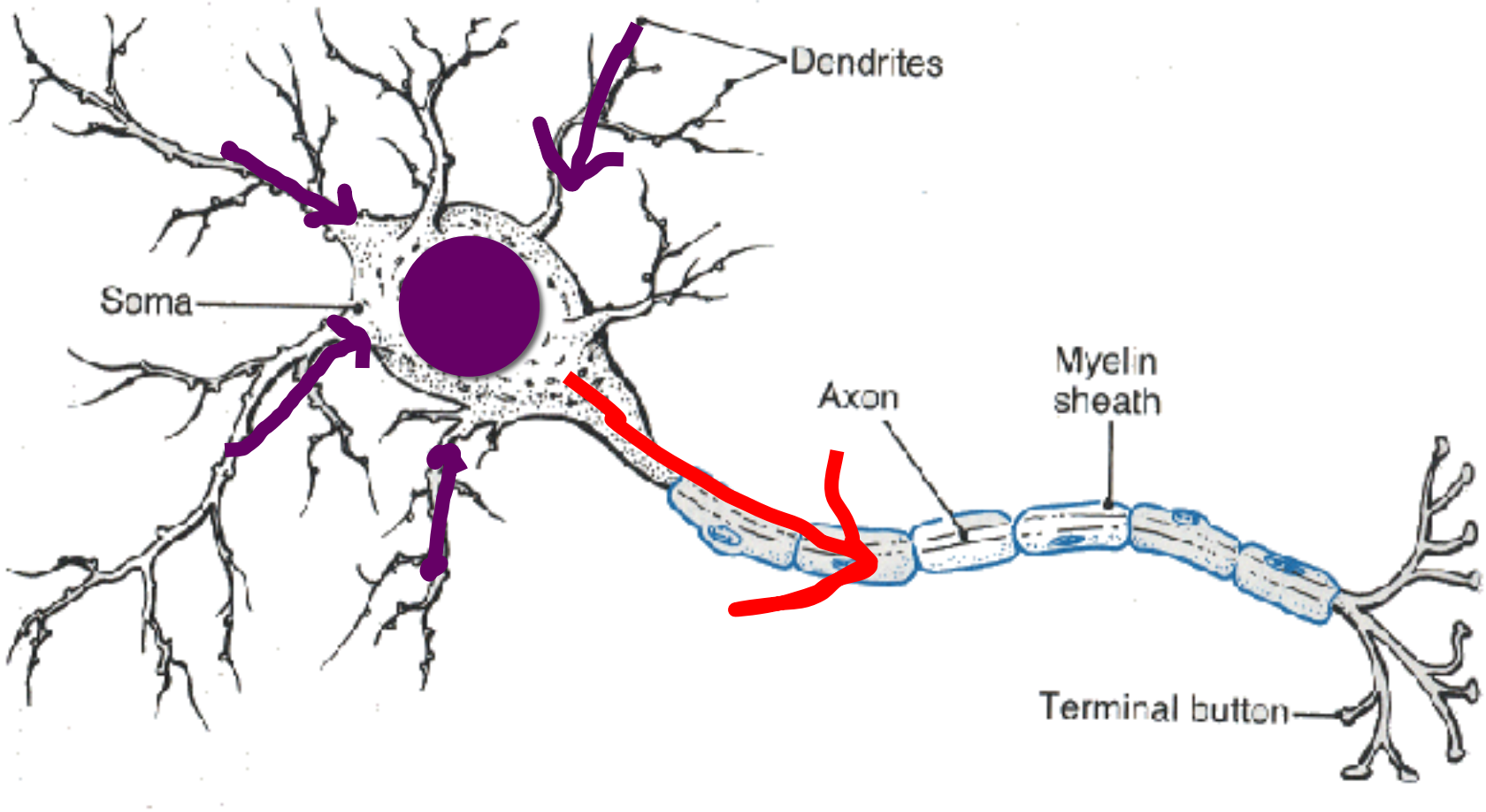
Neuron



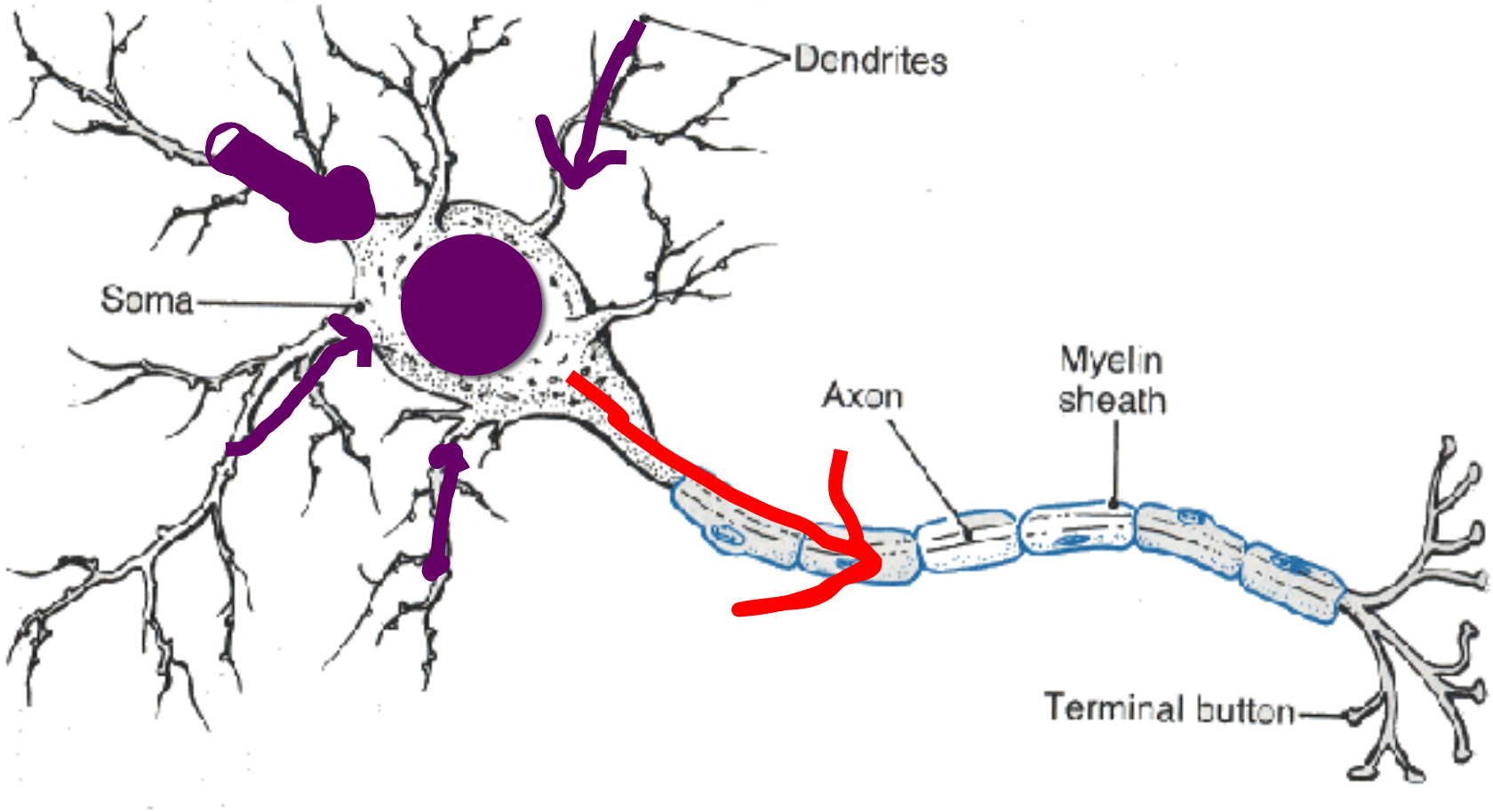
Neuron



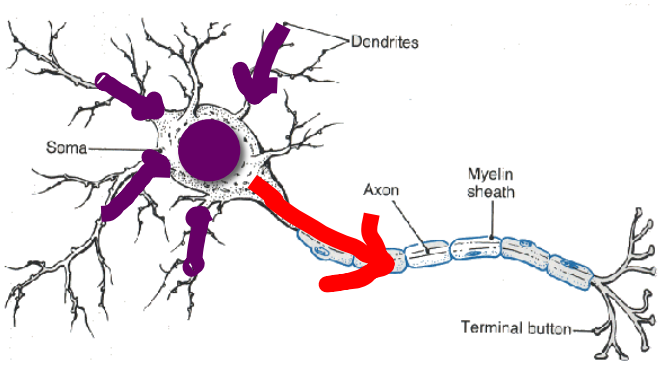
Neuron

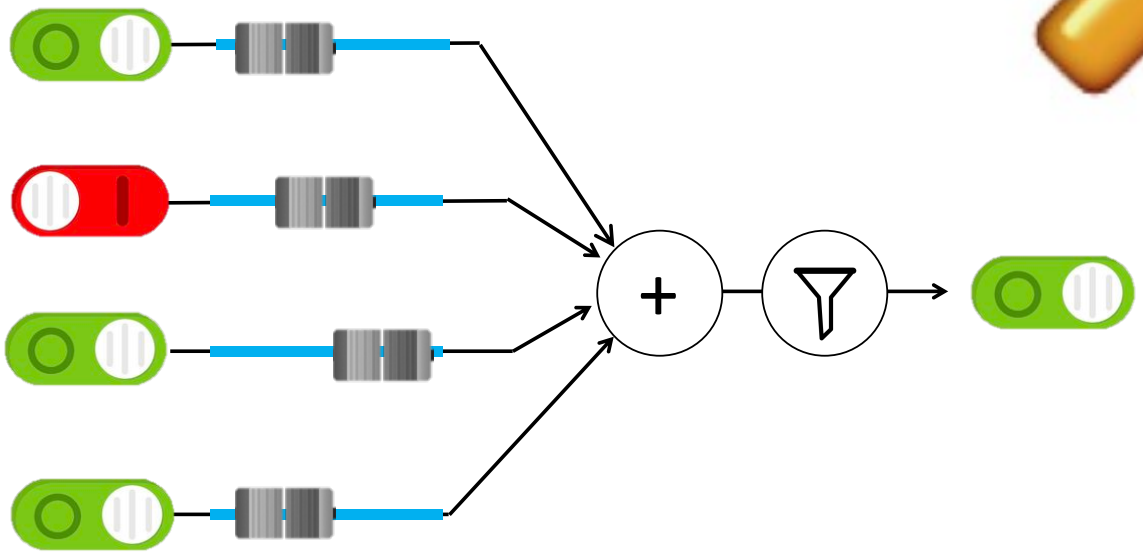


Some inputs are more important



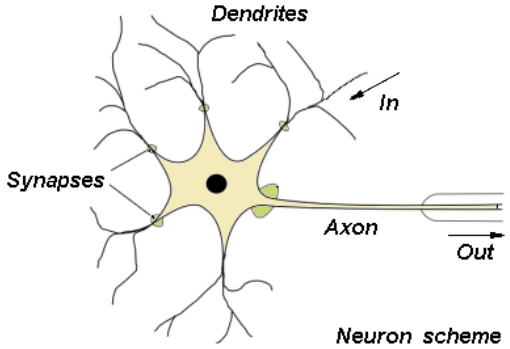
Artificial Neurons

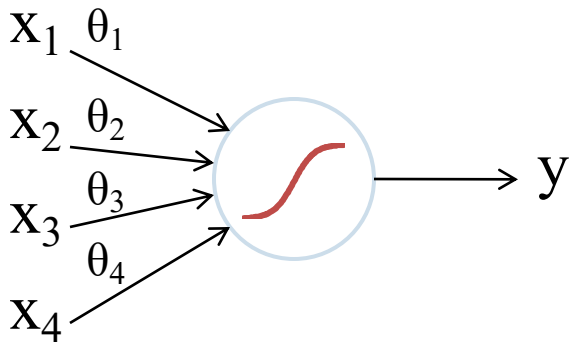




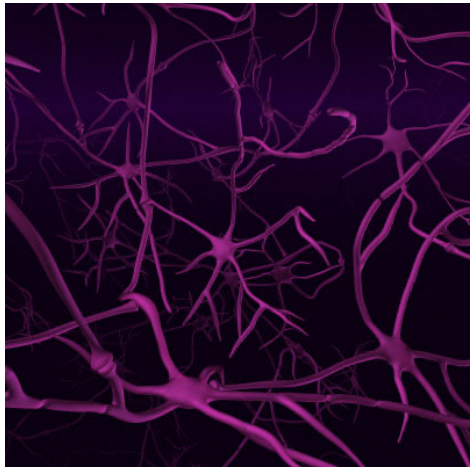
Biological Basis for Neural Networks

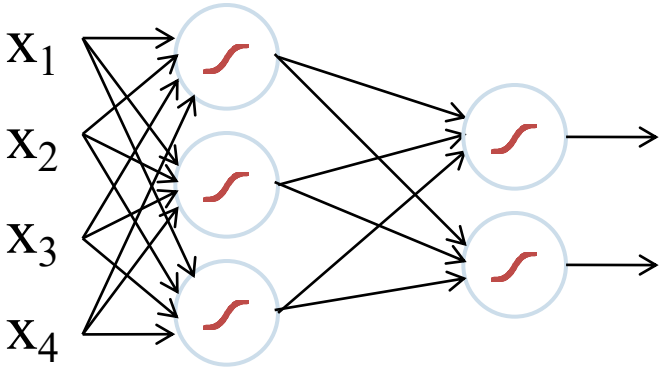
A neuron





Your brain





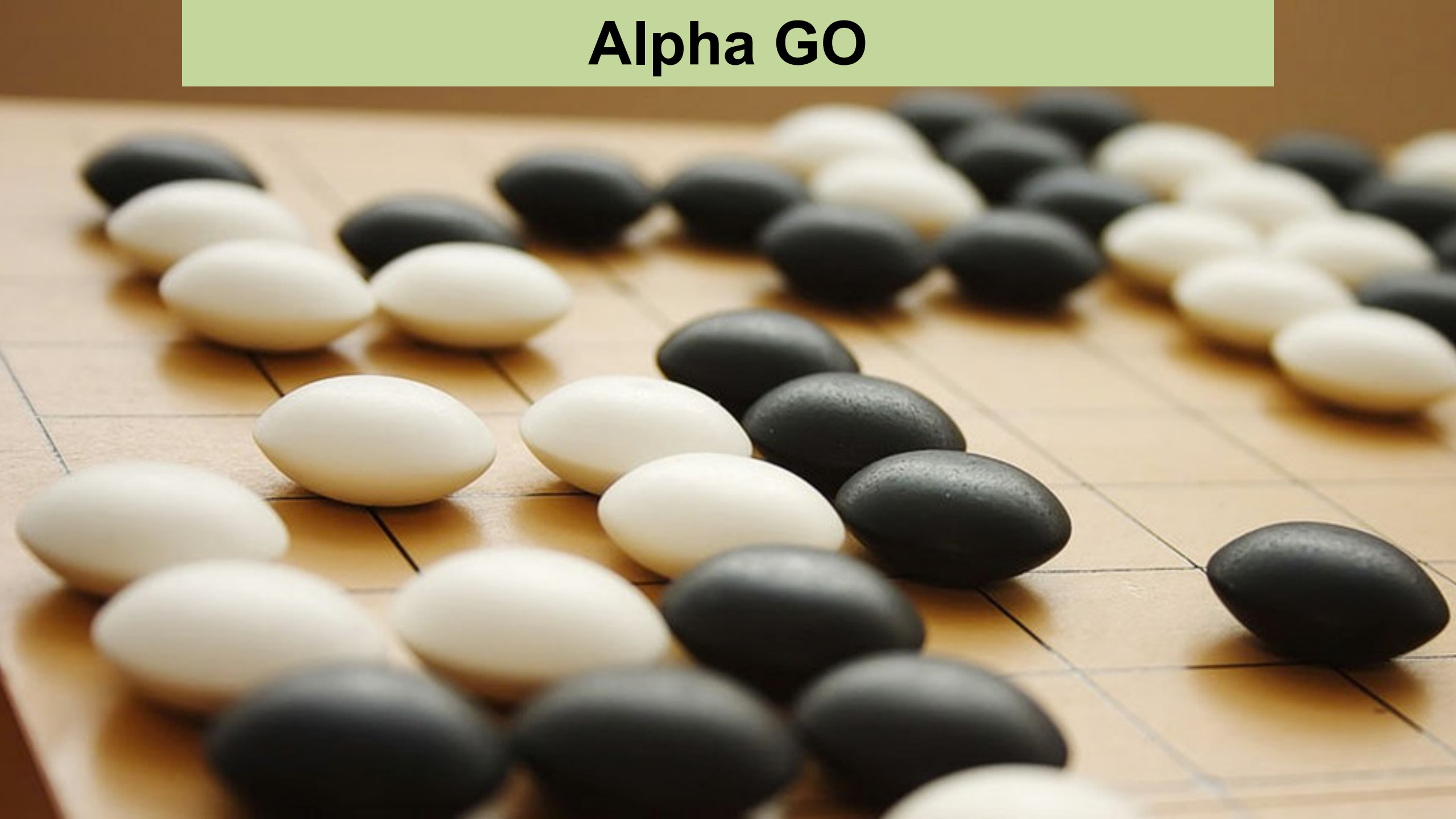
*** Actually, it's probably someone else's brain**

(aka Neural Networks)

Deep learning is (at its core) many logistic regression pieces stacked on top of each other.

Computer Vision

Alpha GO



Revolution in AI

Computers Making Art

Basically just many logistic regression cells
And lots of chain rule…