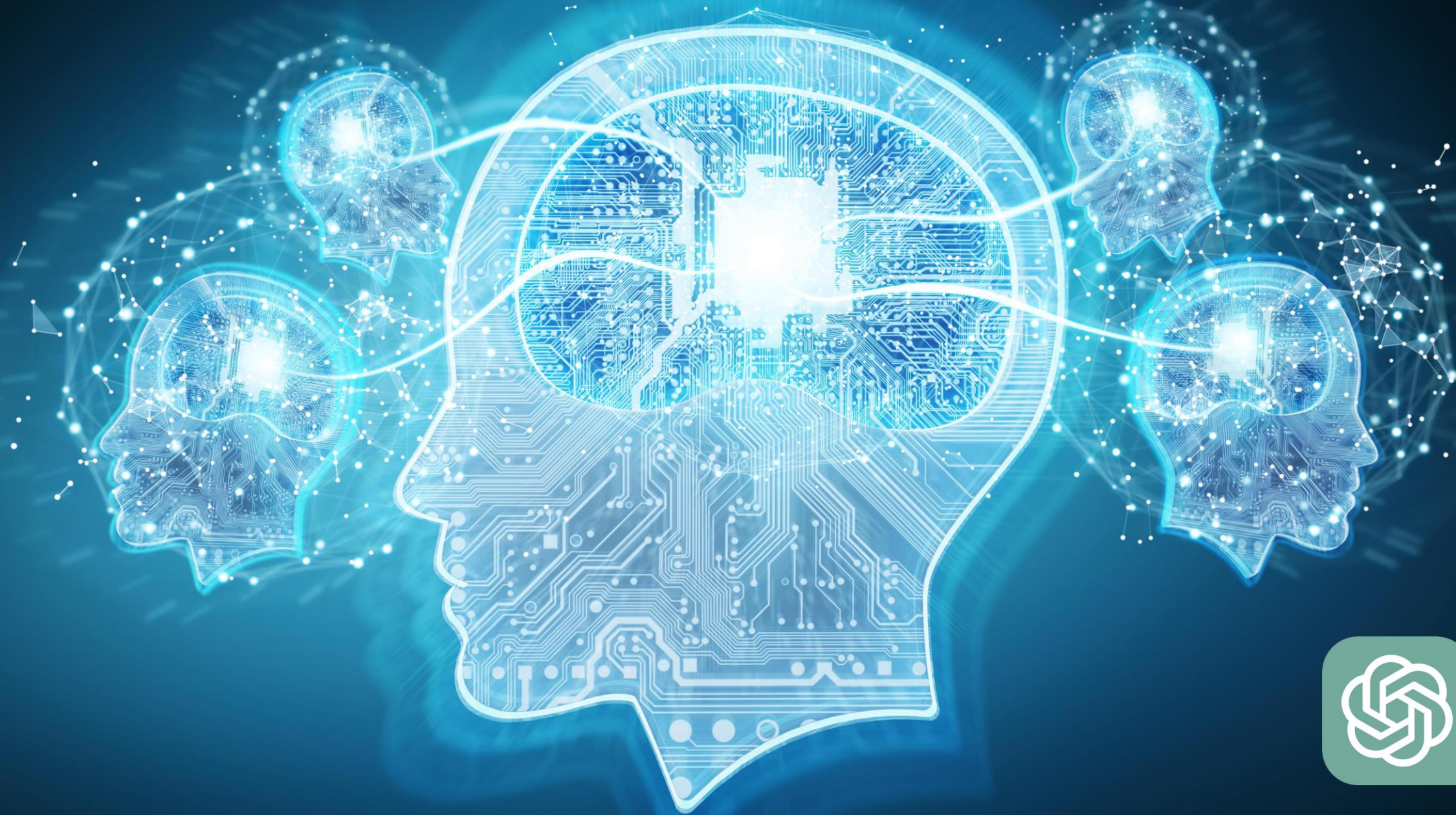
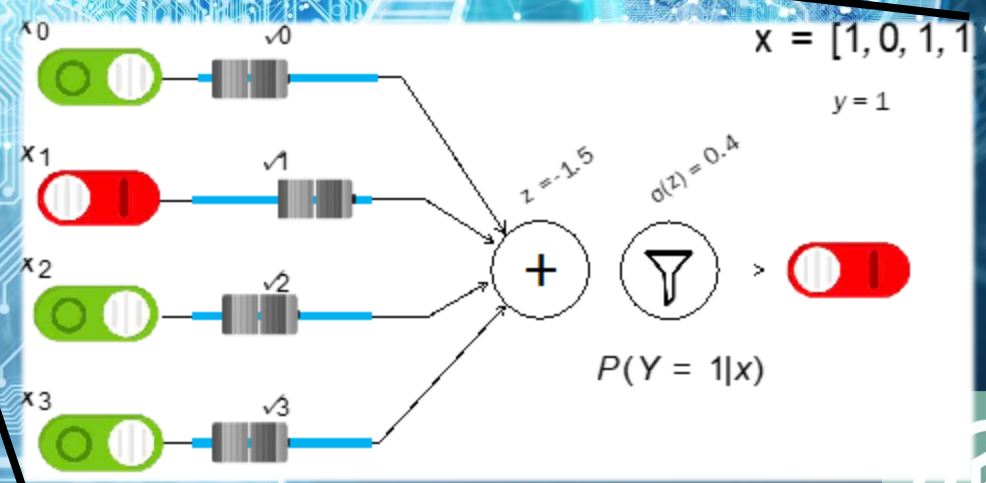
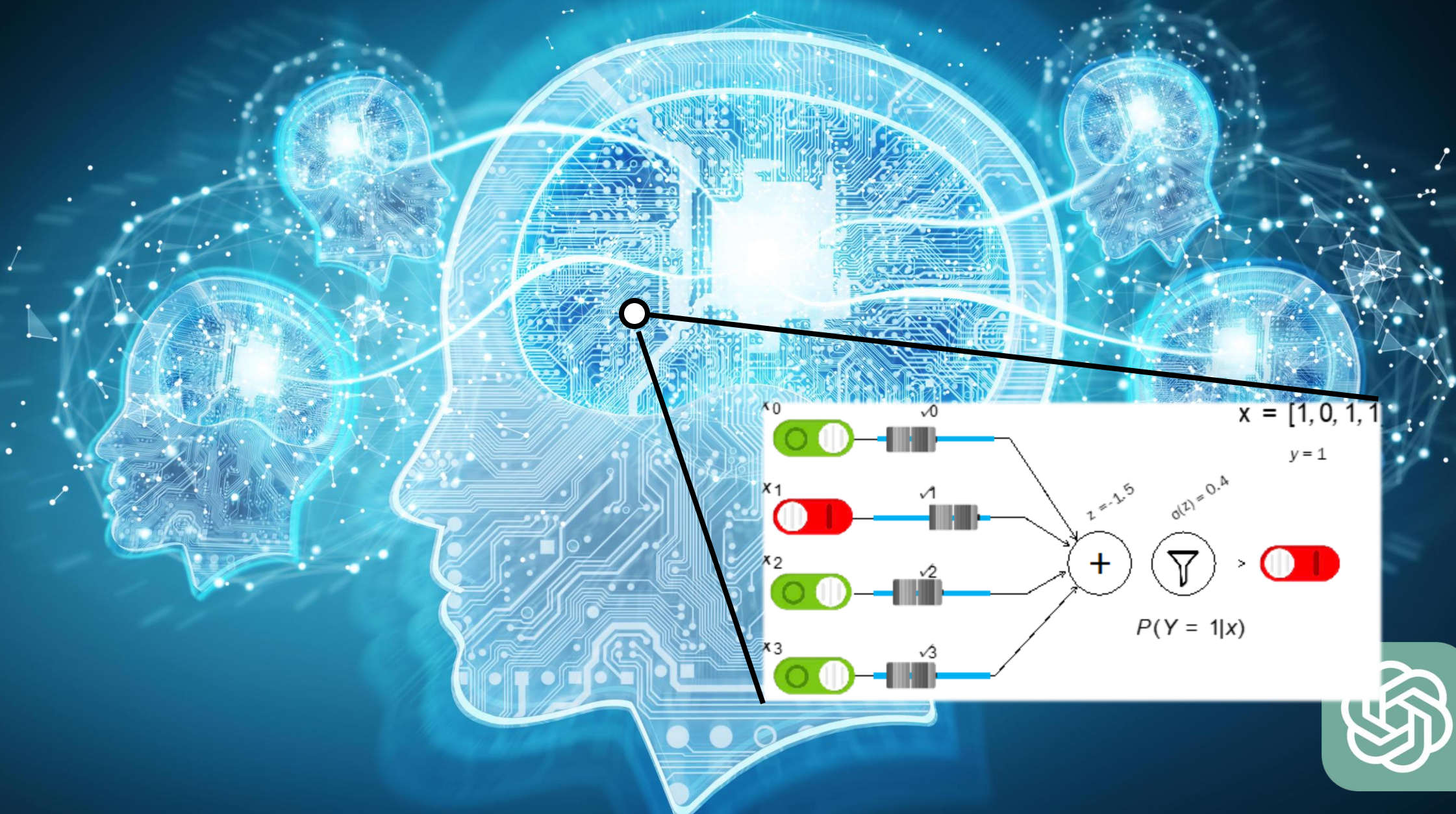


Logistic Regression

Chris Piech

CS109, Stanford University





Where are we in CS109?



Core
Probability

X_2

Random
Variables



Probabilistic
Models



Uncertainty
Theory



Machine
Learning

Where are we in CS109?

Week 8				
21	Mon	Nov 10	Logistic Regression	
22	Wed	Nov 12	Comparing Classifiers	
23	Fri	Nov 14	Beyond Classification	
Week 9				
24	Mon	Nov 17	Deep Learning	PSet 6 Due
25	Wed	Nov 19	Application/Practice	
26	Fri	Nov 21	Application/Practice	
Week 10				
27	Mon	Dec 1	Application/Practice	Final PEP
28	Wed	Dec 3	Last Class	Pset 7 Due/Challenge In
-	Fri	Dec 5	No Class	

Two free Late Days

Pset 6 - Machine Learning For Chris Piech

Get Started

Due Date: Monday, Nov 17, 10:00 PM Pacific Standard Time (in 7 days).

Grace Period Date: Monday, Nov 17, 11:59 PM Pacific Standard Time (in 7 days).

Solutions Posted: Friday, Nov 21, 11:59 PM Pacific Standard Time (in 11 days).

Extension Request Forms ▾

Grace period extension

26 hour extension (1 Late Day)

50 hour extension (2 Late Days)

74 hour extension (3 Late Days)

98 hour extension (4 Late Days)

Over 98 hour extension

You now have 7 late
days total for the
quarter

Earlier the start, the better, for PSet #6

The screenshot shows a web browser window with the URL `cs109psets.netlify.app/fall25/pset6/logistic_regression`. The page title is "PSet 6 - Machine Learning Ge". The main content area is titled "Logistic Regression: Code" and contains the following text:

Implement Logistic Regression for binary input/output data. Specifically, you should implement the gradient ascent algorithm described in class.

Train your algorithm on the data file `simple-train.csv`. Use learning rate $\eta = 0.0001$ and 1,000 training steps. Test your algorithm on the data file `simple-test.csv`. You should be able to achieve 100% classification accuracy on the testing data.

You will need to implement your code off of the pset app. When you are done, include your logistic regression code here and report the value of the weight associated with x_1 to 5 decimal places.

The diagram illustrates a simple neural network for binary classification. It has four input nodes: x_0 , x_1 , x_2 , and x_3 . Each input node is connected to a corresponding weight node: θ_0 , θ_1 , θ_2 , and θ_3 . The outputs of these weight nodes are summed at a node labeled with a plus sign (+). The result of the summation is passed through an activation function node labeled $\sigma(z)$. The final output is a green toggle switch, representing the probability $P(Y = 1|x)$. The equation below the diagram is
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

The right side of the page features an "Answer Editor" and "Solution" tabs. Below these is a "Numeric Answer:" field with "Enter your answer" and a "Check Answer" button. A "Python:" code editor is also present, containing the following text:

```
1 """
2 Paste your logistic regression code here. Note that you shouldn't
3 expect to run the code here. Rather, you're just sharing your implementation
4 so we can review it, but you're expected to run this classifier outside of the
5 problem set app.
6 """
7
```

At the bottom of the code editor is a blue "Run" button.

Earlier the start, the better, for PSet #6

cs109psets.netlify.app/fall25/pset6/mle_wind

PS6 MLE of the Wind

Climate sensitivity suggests that there is a fierce urgency to developing clean energy solutions. Wind is a powerful yet unpredictable source of clean energy and thus requires probability theory. The speed of the wind at a windfarm is a random variable that varies as a *Rayleigh Distribution*. A Rayleigh distribution is parameterized by a single scale parameter θ and has the following probability density function.


$$f_X(x) = \begin{cases} \frac{x}{\theta} e^{-x^2/2\theta} & x \geq 0 \\ 0 & \text{else} \end{cases}$$

We wish to model the wind speed on a wind farm. To this end we collect N independent measurements of wind speeds w_1, w_2, \dots, w_N .

Your Task: Derive an equation for the maximum likelihood estimate of θ if we are modeling the wind speed as coming from a Rayleigh distribution. Make sure to include the equation in your answer. Then use the equation to estimate θ for observed 10 speeds:

```
[7.55, 8.15, 8.91, 1.17, 6.77, 3.03, 8.43, 5.56, 3.26, 2.55]
```

Give your answer to three decimal places.



Previous Question Next Question

https://cs109psets.netlify.app/fall25/pset6/mle_wind

cs109psets.netlify.app/fall25/pset6/mle_neg_bin

PS6 MLE of Negative Binomial

The Negative Binomial random variable is a generalization of the Geometric random variable and counts **the number of experiments until r successes** where each experiment is independent with probability of success p . The PMF is

$$P(X = x) = \binom{x-1}{r-1} p^r (1-p)^{x-r}$$

Given that $r = 5$ and 100 samples of X , use MLE to estimate the parameter p .

```
X = [9, 23, 19, 10, 21, 27, 6, 10, 12, 16, 10, 20, 18, 12, 13, 9, 7, 23, 17, 23, 7, 11, 19, 15, 16, 14, 20, 10, 23, 9, 18, 10, 10, 11, 14, 19, 13, 7, 15, 17, 10, 19, 14, 10, 14, 16, 17, 15, 19, 14, 17, 16, 11, 17, 22, 12, 10, 9, 22, 11, 6, 14, 12, 17, 15, 15, 7, 23, 9, 12, 17, 21, 10, 17, 18, 7, 15, 23, 8, 6, 12, 18, 7, 25, 11, 16, 9, 11, 13, 30, 28, 14, 10, 11, 15, 11, 11, 14, 16, 18]
```

Previous Question Next Question

cs109psets.netlify.app/fall25/pset6/mle_truncated_normal

PS6 MLE of Truncated Normal

A normal distribution can allow negative values. A positive truncated normal is a version of the normal distribution that only supports values of x that are greater than 0. If $X \sim \text{PosTruncNorm}(\mu)$ then it has the following PDF:

$$f(X = x) = \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\Phi(\mu)} \quad \text{for } x > 0.$$

Where Φ is the Cumulative Distribution Function for the Standard Normal.

You have 200 i.i.d. samples from a PosTruncNorm:

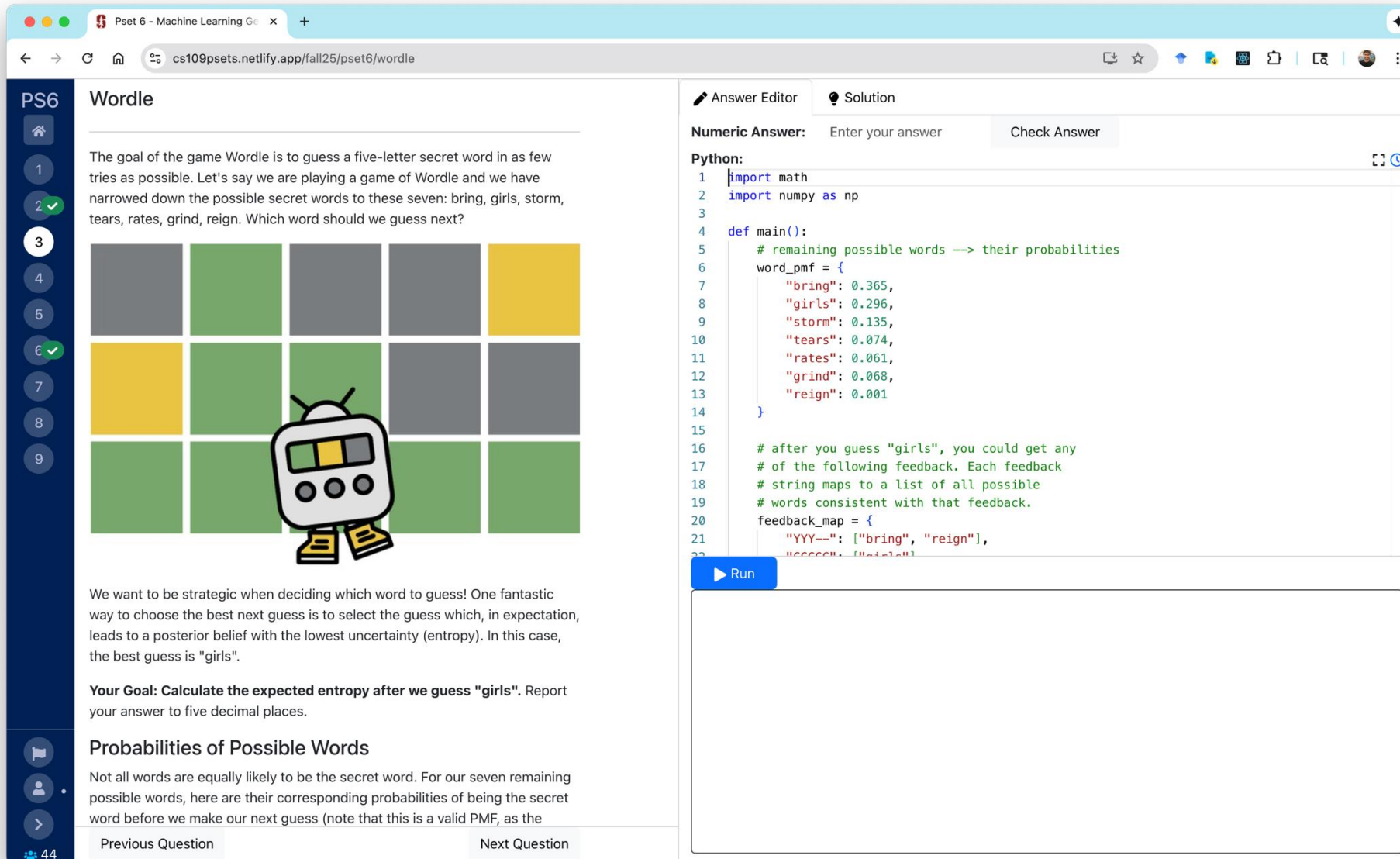
```
[0.76, 2.48, 1.53, 1.21, 0.34, 0.34, 0.13, 1.97, 1.22, 1.47, 0.05, 2.70, 1.84, 0.45, 0.39, 0.40, 0.63, 1.06, 0.87, 0.61, 1.24, 0.31, 0.61, 0.75, 0.92, 1.68, 0.43, 1.04, 1.2, 0.11, 1.23, 0.37, 0.15, 2.47, 2.64, 1.76, 0.63, 0.22, 1.41, 0.89, 0.27, 1.00, 0.08, 2.19, 0.54, 1.36, 0.65, 1.05, 1.10, 0.40, 2.69, 1.65, 2.39, 2.11, 1.21, 2.26, 0.2, 0.42, 0.11, 0.67, 0.79, 0.57, 1.83, 0.73, 0.59, 1.09, 0.31, 1.74, 0.17, 3.03, 1.64, 0.43, 0.01, 1.78, 1.46, 1.52, 1.64, 0.17, 0.73, 0.26, 1.96, 1.27, 0.68, 0.15, 0.64, 0.67, 1.52, 1.3, 2.07, 0.95, 0.27, 1.48, 1.61, 1.13, 1.64, 1.0, 1.05, 0.87, 0.06, 0.24, 0.07, 1.3, 0.65, 1.03, 2.18, 0.53, 0.83, 1.6, 0.49, 0.18, 0.6, 0.35, 2.31, 1.76, 1.29, 2.0, 1.74, 0.4, 2.1, 1.09, 1.75, 2.11, 0.66, 0.25, 0.48, 0.87, 1.79, 1.95, 0.02, 1.03, 0.85, 0.47, 0.27, 0.69, 2.41, 0.67, 1.05, 1.46, 0.74, 2.73, 2.6, 0.53, 1.0, 0.62, 0.59, 0.09, 1.24, 1.01, 0.12, 0.58, 2.18, 0.51, 0.32, 0.99, 2.99, 0.51, 1.38, 1.61, 0.5, 1.52, 0.75, 1.29, 1.29, 1.08, 0.21, 1.85, 0.66, 0.4, 0.1, 1.2, 1.39, 0.04, 1.03, 0.48, 1.32, 0.38, 1.42, 0.79, 2.36, 0.3, 0.7, 0.25, 2.28, 2.02, 0.54, 1.35, 1.79, 1.12, 1.07, 0.51, 0.21, 2.12, 2.14, 1.29, 0.7, 0.72, 1.51, 2.12, 2.07, 1.67]
```

Let x_i be the i th value.

Estimate the parameter μ using gradient descent starting from an estimated μ .

Previous Question Next Question

Earlier the start, the better, for PSet #6



The screenshot shows a web browser window with the URL `cs109psets.netlify.app/fall25/pset6/wordle`. The page is titled "Wordle" and contains the following content:

PS6

1
2 ✓
3
4
5
6 ✓
7
8
9

The goal of the game Wordle is to guess a five-letter secret word in as few tries as possible. Let's say we are playing a game of Wordle and we have narrowed down the possible secret words to these seven: bring, girls, storm, tears, rates, grind, reign. Which word should we guess next?

We want to be strategic when deciding which word to guess! One fantastic way to choose the best next guess is to select the guess which, in expectation, leads to a posterior belief with the lowest uncertainty (entropy). In this case, the best guess is "girls".

Your Goal: Calculate the expected entropy after we guess "girls". Report your answer to five decimal places.

Probabilities of Possible Words

Not all words are equally likely to be the secret word. For our seven remaining possible words, here are their corresponding probabilities of being the secret word before we make our next guess (note that this is a valid PMF, as the

Previous Question Next Question

44

Answer Editor **Solution**

Numeric Answer: Check Answer

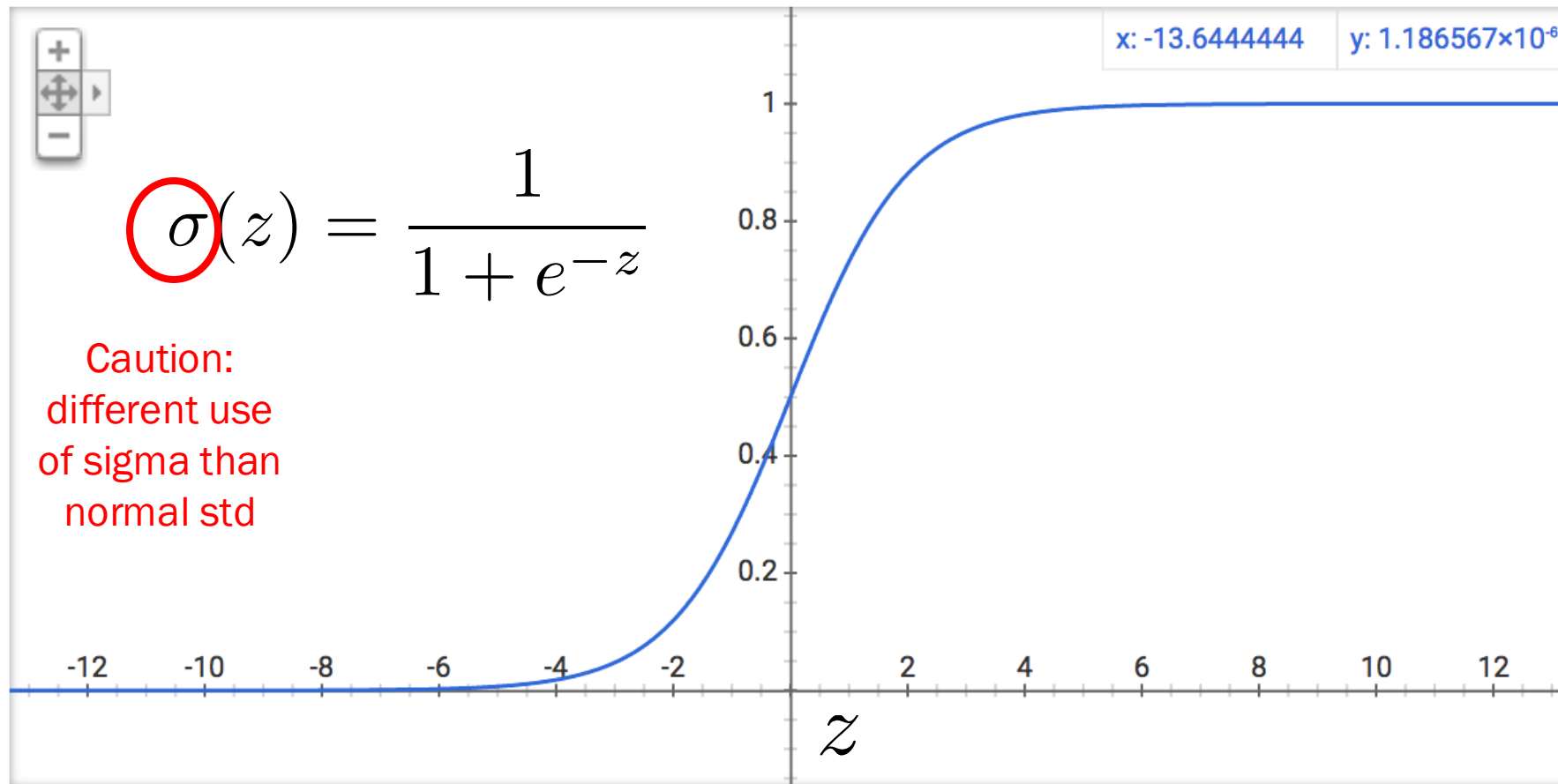
Python:

```
1 import math
2 import numpy as np
3
4 def main():
5     # remaining possible words --> their probabilities
6     word_pmf = {
7         "bring": 0.365,
8         "girls": 0.296,
9         "storm": 0.135,
10        "tears": 0.074,
11        "rates": 0.061,
12        "grind": 0.068,
13        "reign": 0.001
14    }
15
16    # after you guess "girls", you could get any
17    # of the following feedback. Each feedback
18    # string maps to a list of all possible
19    # words consistent with that feedback.
20    feedback_map = {
21        "YYY--": ["bring", "reign"],
22        "CCCC-": ["girls"],
23    }
```

Run

Review

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

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

Weighted sum
(aka dot product)

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

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

Sigmoid function of
weighted sum

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

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

Machine Learning (aka Applied Probability)

Machine Learning in CS109

Great Idea

Neural Networks

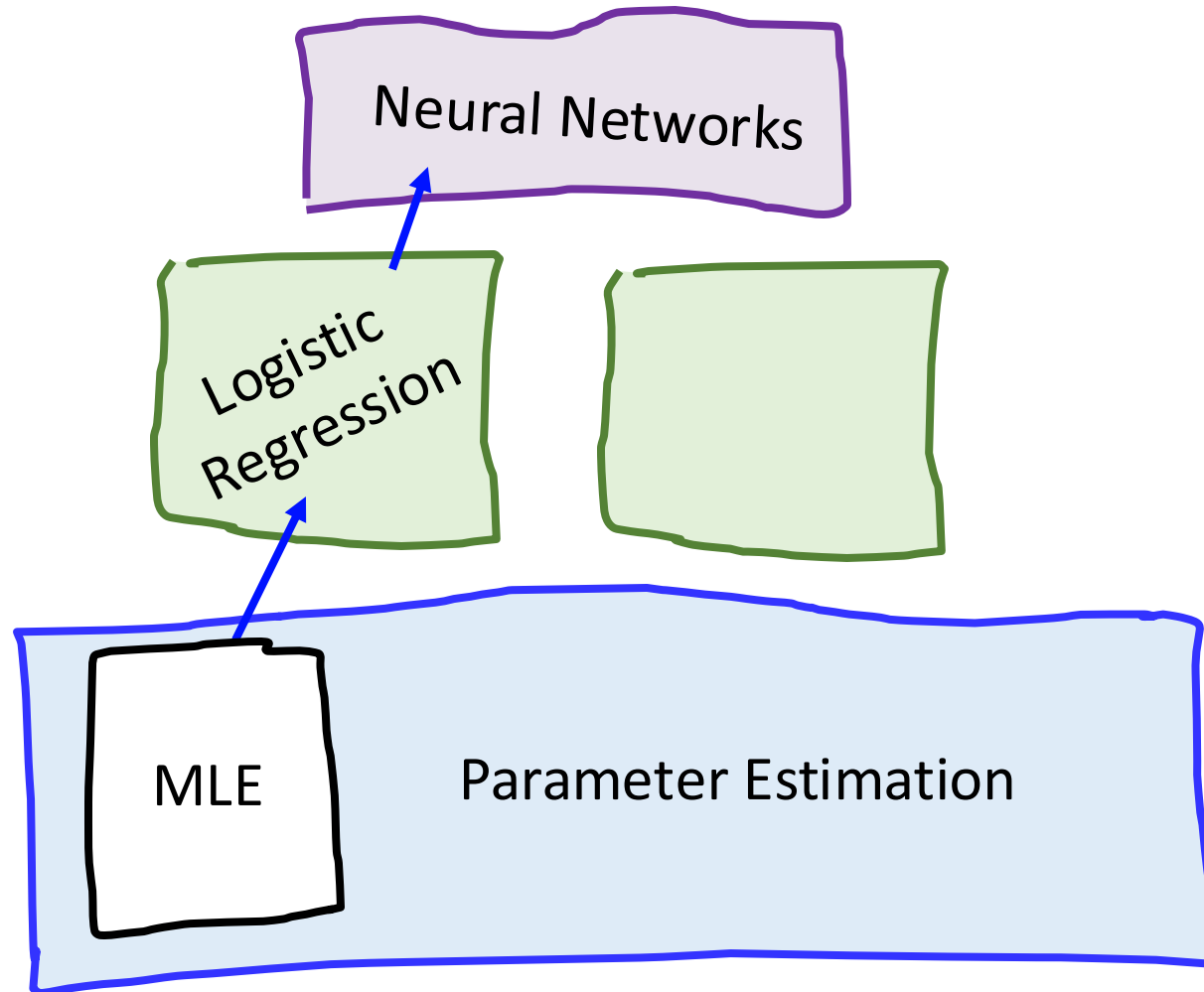
Core Algorithms

Logistic Regression

Theory

MLE

Parameter Estimation



MLE Idea: Chose params that make the data look likely

Data = [6.3 , 5.5 , 5.4, 7.1, 4.6, 6.7, 5.3 , 4.8, 5.6, 3.4, 5.4, 3.4, 4.8, 7.9, 4.6, 7.0, 2.9, 6.4, 6.0 , 4.3]

Estimate the Parameters

Parameter μ :

Parameter σ :

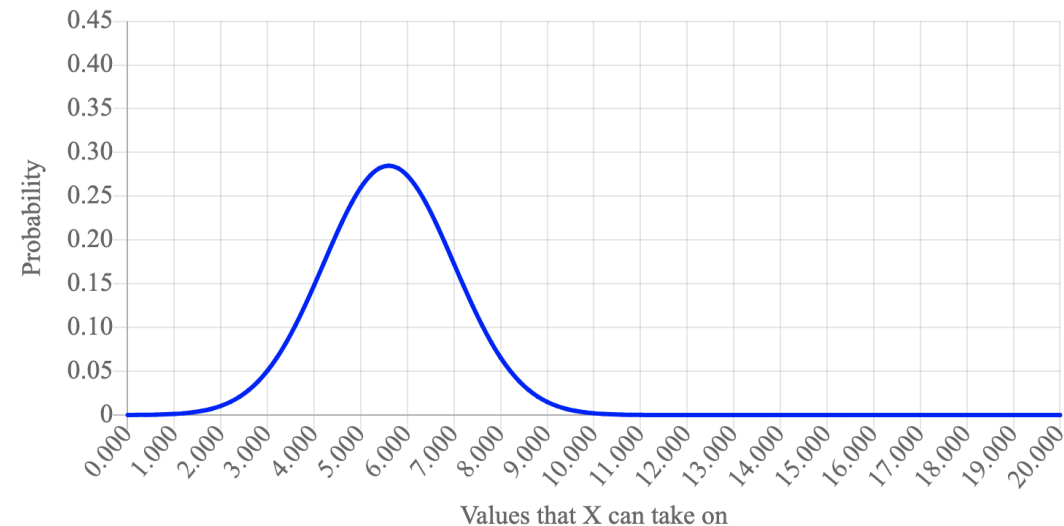
Likelihood

Likelihood: 1.9542923784106326e-15

Log Likelihood: -301.9

Best Seen: -301.9

PDF Graph



Likelihood Definition

Wikipedia:

Likelihood function

[Article](#) [Talk](#)

From Wikipedia, the free encyclopedia

The **likelihood function** (often simply called the **likelihood**) is the [joint probability](#) (or probability density) of [observed data](#) viewed as a function of the [parameters](#) of a [statistical model](#).^{[1] [2] [3]}

A generalized term for “PDF / PMF / Joint”
of data as a function of parameters

Maximum Likelihood

That maximize likelihood



$$\hat{\theta}_{MLE} = \operatorname{argmax}_{\theta} f(x^{(1)}, \dots, x^{(n)} | \theta)$$

Chose the params



$$L(\theta) = \prod_{i=1}^n f(x_i | \theta)$$

Define likelihood, use independence.

$$LL(\theta) = \sum_{i=1}^n \log f(x_i | \theta)$$

Define the loglikelihood

$$\hat{\theta}_{MLE} = \operatorname{argmax}_{\theta} LL(\theta)$$

Use LL to chose params

MLE for a Pareto

```
observations = [1.677, 3.812, 1.463, 2.641, 1.256, 1.678, 1.157,  
1.146, 1.323, 1.029, 1.238, 1.018, 1.171, 1.123, 1.074, 1.652,  
1.873, 1.314, 1.309, 3.325, 1.045, 2.271, 1.305, 1.277, 1.114,  
1.391, 3.728, 1.405, 1.054, 2.789, 1.019, 1.218, 1.033, 1.362,  
1.058, 2.037, 1.171, 1.457, 1.518, 1.117, 1.153, 2.257, 1.022,  
1.839, 1.706, 1.139, 1.501, 1.238, 2.53, 1.414, 1.064, 1.097,  
1.261, 1.784, 1.196, 1.169, 2.101, 1.132, 1.193, 1.239, 1.518,  
2.764, 1.053, 1.267, 1.015, 1.789, 1.099, 1.25, 1.253, 1.418,  
1.494, 1.015, 1.459, 2.175, 2.044, 1.551, 4.095, 1.396, 1.262,  
1.351, 1.121, 1.196, 1.391, 1.305, 1.141, 1.157, 1.155, 1.103,  
1.048, 1.918, 1.889, 1.068, 1.811, 1.198, 1.361, 1.261, 4.093,  
2.925, 1.133, 1.573]
```

```
def estimate_alpha(observations):  
    print('your code here')
```



We know sand is distributed as a pareto with PDF

$$f(x) = \frac{\alpha}{x^{\alpha+1}}$$

$$\alpha_{\text{mle}} = \frac{n}{\sum_i \log x_i}$$

MLE for a Pareto

Consider I.I.D. random variables X_1, X_2, \dots, X_n

- $X_i \sim \text{Pareto}(\alpha)$. **Use Maximum Likelihood to estimate α .**

1. What is the likelihood of all the *data*

2. What is the log-likelihood all the *data*

3. Find the value of α which maximizes log likelihood

MLE for a Pareto

Consider I.I.D. random variables X_1, X_2, \dots, X_n

- $X_i \sim \text{Pareto}(\alpha)$. **Use Maximum Likelihood to estimate α .**
- Likelihood:

$$L(\alpha) = \prod_{i=1}^n \frac{\alpha}{x_i^{\alpha+1}}$$

2. What is the log-likelihood all the *data*

3. Find the value of α which maximizes log likelihood

MLE for a Pareto

Consider I.I.D. random variables X_1, X_2, \dots, X_n

- $X_i \sim \text{Pareto}(\alpha)$. **Use Maximum Likelihood to estimate α .**

- Likelihood:

$$L(\alpha) = \prod_{i=1}^n \frac{\alpha}{x_i^{\alpha+1}}$$

- Log-likelihood:

$$LL(\alpha) = \sum_{i=1}^n \log \alpha - (\alpha + 1) \log x_i = n \log \alpha - (\alpha + 1) \sum_{i=1}^n \log x_i$$

3. Find the value of α which maximizes log likelihood

MLE for a Pareto

Consider I.I.D. random variables X_1, X_2, \dots, X_n

- $X_i \sim \text{Pareto}(\alpha)$. **Use Maximum Likelihood to estimate α .**

- Likelihood:

$$L(\alpha) = \prod_{i=1}^n \frac{\alpha}{x_i^{\alpha+1}}$$

- Log-likelihood:

$$LL(\alpha) = \sum_{i=1}^n \log \alpha - (\alpha + 1) \log x_i = n \log \alpha - (\alpha + 1) \sum_{i=1}^n \log x_i$$

- Chose α to be the argmax of LL:

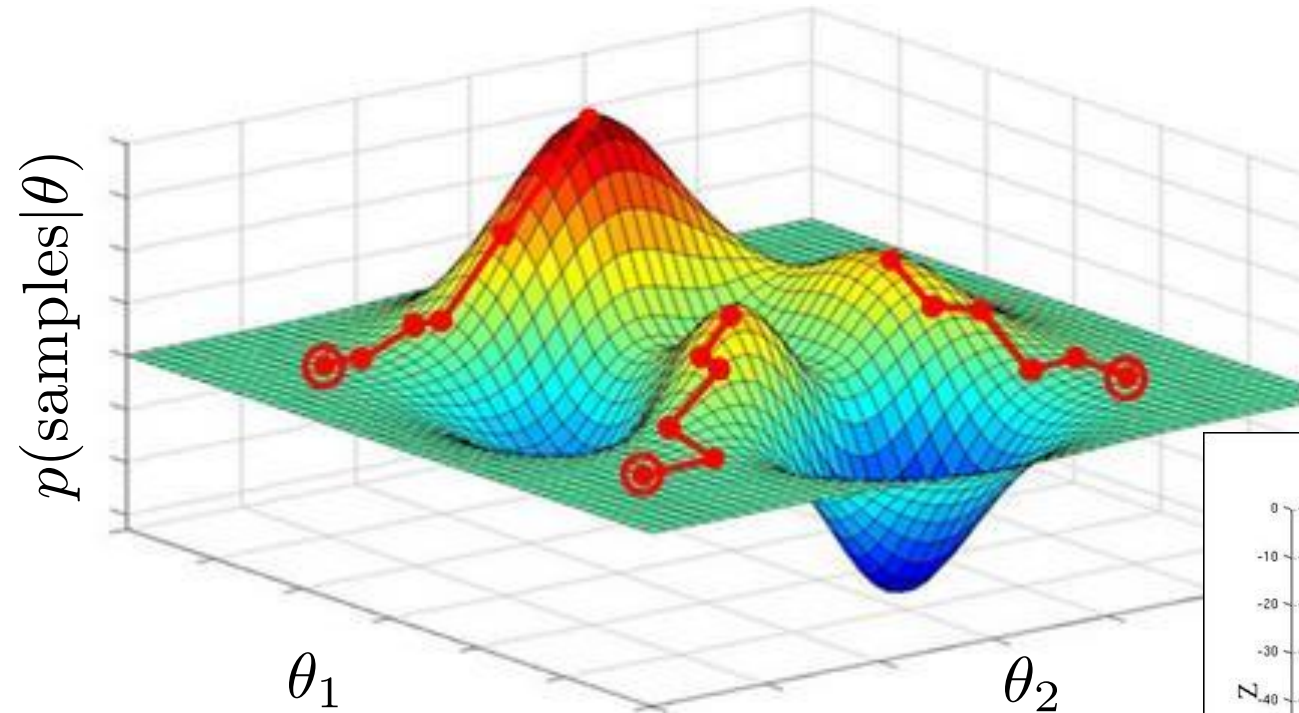
$$\frac{\partial LL(\alpha)}{\partial \alpha} = \frac{n}{\alpha} - \sum_{i=1}^n \log x_i$$

Argmax Option #1: set the derivative to 0, and solve for alpha

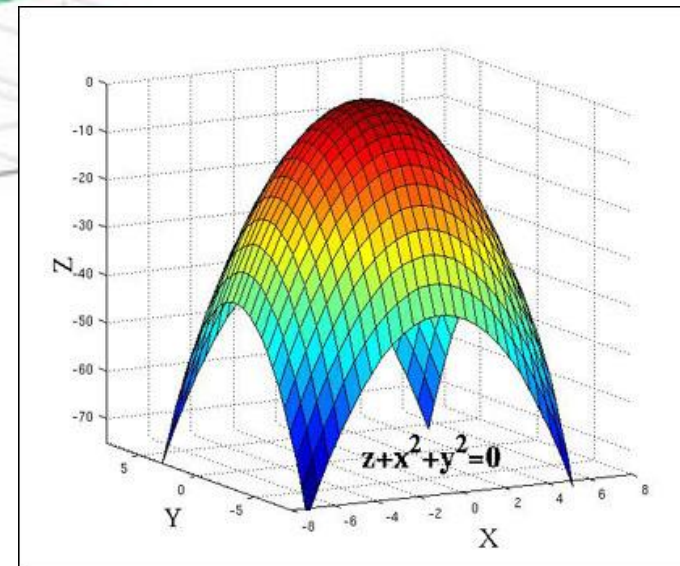


arg max

Gradient Ascent



Especially good if
function is convex



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

Gradient Ascent

Repeat many times

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



Step size constant

This is some **profound** life philosophy

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

Gradient Ascent

Initialize: $\theta_j = \text{random}$ for all $0 \leq j \leq m$

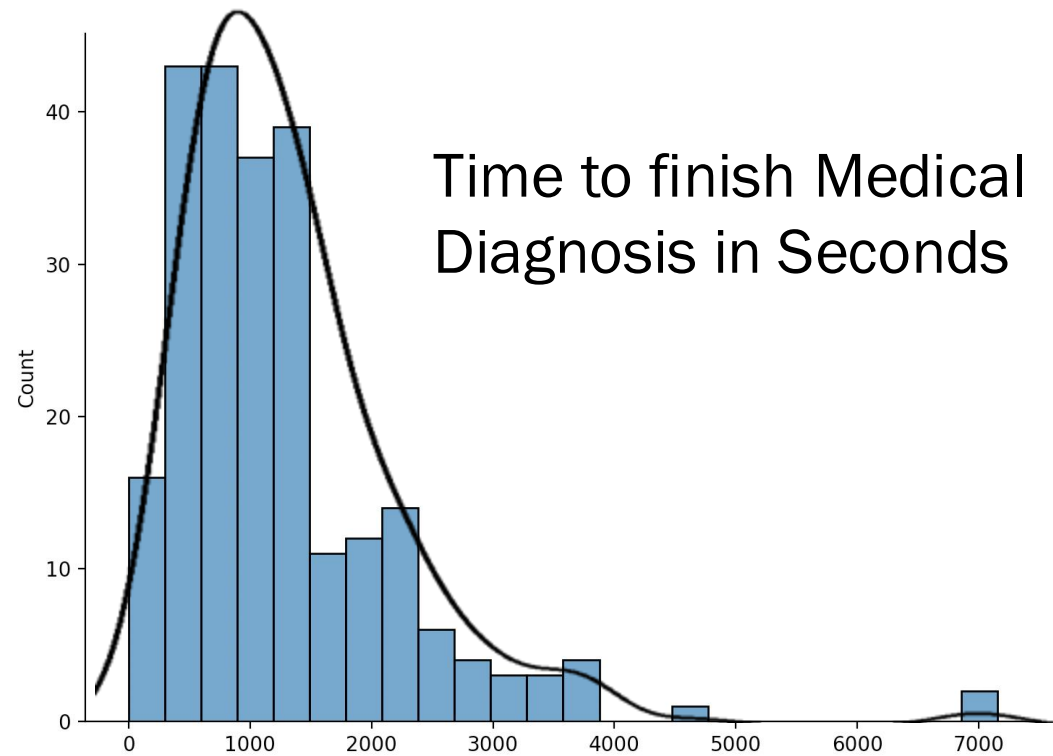
Repeat many times:

Calculate all gradient[j]'s based on data

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

MLE of Erlang

```
[3.002, 0.983, 2.186, 1.624, 3.997, 1.777,
2.809, 0.42, 0.515, 1.582, 0.948, 0.458, 1.
066, 0.8, 2.398, 0.794, 2.561, 2.61, 0.
595, 3.897, 1.852, 1.182, 3.043, 0.905, 1.
45, 0.405, 0.445, 2.103, 1.425, 3.12, 0.
973, 1.056, 3.715, 2.952, 1.817, 2.686, 4.
173, 0.358, 2.185, 2.581, 7.134, 0.206, 2.
049, 0.896, 2.095, 4.39, 2.199, 3.434, 5.
696, 0.819, 0.416, 1.571, 1.337, 2.79, 2.
701, 3.061, 4.677, 0.671, 1.594, 3.586, 2.
708, 1.417, 1.799, 1.137, 1.771, 2.12, 0.
93, 6.835, 3.213, 2.541, 2.505, 1.257, 1.
99, 1.5, 0.014, 3.856, 0.979, 2.413, 2.
596, 1.653, 0.881, 4.457, 0.717, 3.305, 2.
456, 3.462, 1.737, 0.968, 0.528, 0.18, 1.
626, 2.224, 1.466, 1.6, 1.572, 0.12, 2.86,
1.062, 2.139, 1.217]
```



$$f(x) = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!} = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{\Gamma(k)}$$

End Review

Warmup

$$X \sim \text{Bern}(p)$$

Don't we already have the Beta?

Yes! But this example is critical for developing
towards deep learning.

Maximum Likelihood with Bernoulli

Consider I.I.D. random variables X_1, X_2, \dots, X_n

- $X_i \sim \text{Bern}(p)$ **Use Maximum Likelihood to estimate p**

- Probability mass function can be written as:
$$f(x_i|p) = \begin{cases} p & \text{if } x_i = 1 \\ 1 - p & \text{if } x_i = 0 \end{cases}$$

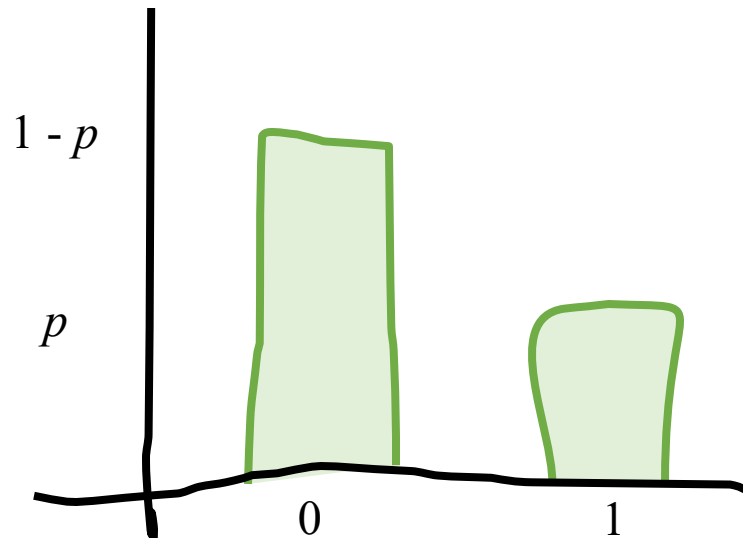


Differentiable PMF for Bernoulli

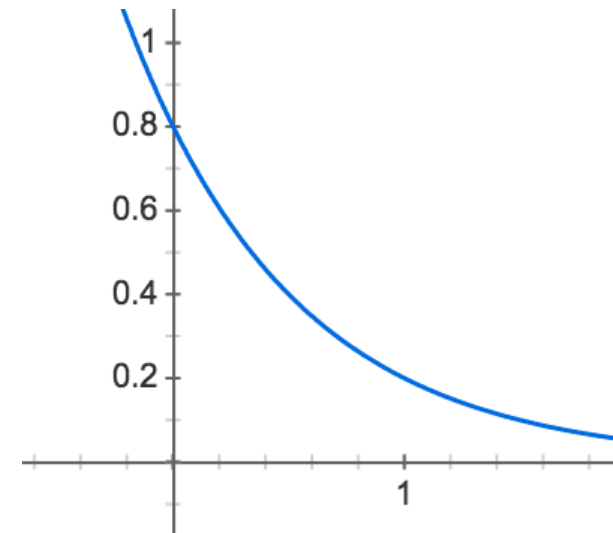
Consider I.I.D. random variables X_1, X_2, \dots, X_n

- $X_i \sim \text{Ber}(p)$
- Probability mass function, $f(X_i = x_i | P = p)$

PMF of Bernoulli



PMF of Bernoulli ($p = 0.2$)



$$f(x_i | p) = p^{x_i} (1 - p)^{1 - x_i}$$
$$f(x_i | p = 0.2) = 0.2^{x_i} (1 - 0.2)^{1 - x_i}$$

Bernoulli PMF

$$X \sim \text{Ber}(p)$$



$$f(X = x|p) = p^x (1 - p)^{1-x}$$

Maximum Likelihood with Bernoulli

Consider I.I.D. random variables X_1, X_2, \dots, X_n

- $X_i \sim \text{Bern}(p)$ **Use Maximum Likelihood to estimate p**

1. What is the likelihood of one X_i

2. What is the likelihood of all the *data*

3. What is the log-likelihood all the *data*

4. Find the value of p which maximizes log likelihood

Maximum Likelihood with Bernoulli

Consider I.I.D. random variables X_1, X_2, \dots, X_n

- $X_i \sim \text{Bern}(p)$ **Use Maximum Likelihood to estimate p**
- Probability mass function can be written as: $f(x_i|p) = p^{x_i}(1-p)^{1-x_i}$

2. What is the likelihood of all the *data*

3. What is the log-likelihood all the *data*

4. Find the value of p which maximizes log likelihood

Maximum Likelihood with Bernoulli

Consider I.I.D. random variables X_1, X_2, \dots, X_n

- $X_i \sim \text{Bern}(p)$ **Use Maximum Likelihood to estimate p**

- Probability mass function can be written as: $f(x_i|p) = p^{x_i}(1-p)^{1-x_i}$

- Likelihood:
$$L(p) = \prod_{i=1}^n f(x_i|p) = \prod_{i=1}^n p^{x_i}(1-p)^{1-x_i}$$

3. What is the log-likelihood all the *data*

4. Find the value of p which maximizes log likelihood

Maximum Likelihood with Bernoulli

Consider I.I.D. random variables X_1, X_2, \dots, X_n

- $X_i \sim \text{Bern}(p)$ **Use Maximum Likelihood to estimate p**

- Probability mass function can be written as: $f(x_i|p) = p^{x_i}(1-p)^{1-x_i}$

- Likelihood:
$$L(p) = \prod_{i=1}^n f(x_i|p) = \prod_{i=1}^n p^{x_i}(1-p)^{1-x_i}$$

- Log-likelihood:

$$LL(p) = \sum_{i=1}^n x_i \log p + (1 - x_i) \log(1 - p)$$

4. Find the value of p which maximizes log likelihood

Take Derivative:

$$LL(p) = \sum_{i=1}^n x_i \log p + (1 - x_i) \log(1 - p)$$

$$\frac{\partial LL(p)}{\partial p} = \frac{\partial}{\partial p} \sum_{i=1}^n x_i \log p + (1 - x_i) \log(1 - p)$$

Take the derivative wrt p

$$= \sum_{i=1}^n \frac{\partial}{\partial p} \left[x_i \log p + (1 - x_i) \log(1 - p) \right]$$

Derivative of a sum!

$$= \sum_{i=1}^n \left[\frac{\partial}{\partial p} x_i \log p \right] + \frac{\partial}{\partial p} (1 - x_i) \log(1 - p)$$

Derivative of a sum!

$$= \sum_{i=1}^n \frac{x_i}{p} + \frac{\partial}{\partial p} (1 - x_i) \log(1 - p)$$

Derivative of log p

$$= \sum_{i=1}^n \frac{x_i}{p} - \frac{1 - x_i}{1 - p}$$

Derivative of log (1-p)

Set to Zero:

$$\frac{\partial LL(p)}{\partial p} = \sum_{i=1}^n \frac{x_i}{p} - \frac{1 - x_i}{1 - p}$$

$$\begin{aligned} 0 &= \sum_{i=1}^n \frac{x_i}{\hat{p}} - \frac{1 - x_i}{1 - \hat{p}} \\ &= \sum_{i=1}^n \frac{x_i}{\hat{p}} - \sum_{i=1}^n \frac{1 - x_i}{1 - \hat{p}} \\ &= \frac{y}{\hat{p}} - \frac{n - y}{1 - \hat{p}} \end{aligned}$$

$$\frac{n - y}{1 - \hat{p}} = \frac{y}{\hat{p}}$$

$$\hat{p}(n - y) = y(1 - \hat{p})$$

$$\hat{p}n - \hat{p}y = y - \hat{p}y$$

$$\hat{p}n = y$$

Let $\sum_{i=1}^n x_i = y$ To make life easier

And $\sum_{i=1}^n 1 - x_i = \sum_{i=1}^n 1 - \sum_{i=1}^n x_i = n - y$

$$\begin{aligned} \hat{p} &= \frac{1}{n}y \\ &= \frac{1}{n} \sum_{i=1}^n x_i \end{aligned}$$

Isn't that the same as
unbiased estimator?

Yes. For Bernoulli.

MLE of Bernoulli is the sample mean



MLE vs Beta

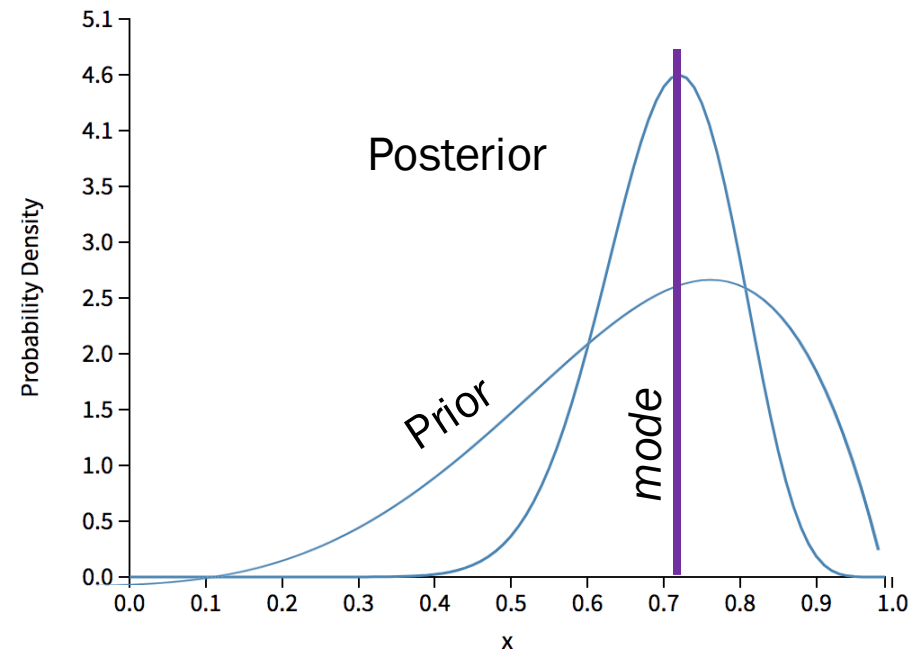
The medicine is tried on 20 patients. It “works” for 14 and “doesn’t work” for 6. What is your new belief that the drug works?

In other words I have 20 IID samples from a Bernoulli. Estimate p . The data is $[1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0]$

MLE estimate:

$$p \approx \frac{14}{20} = 0.7$$

Beta estimate:

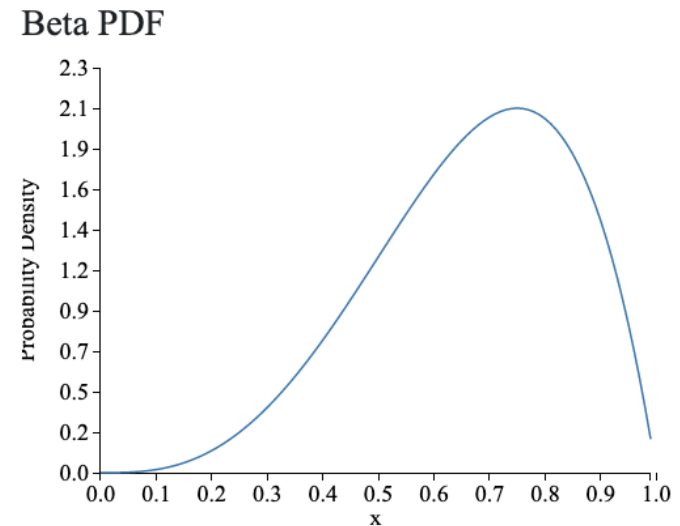




Think about the difference between a **point estimate** and a **distribution**

$$p = 0.75$$

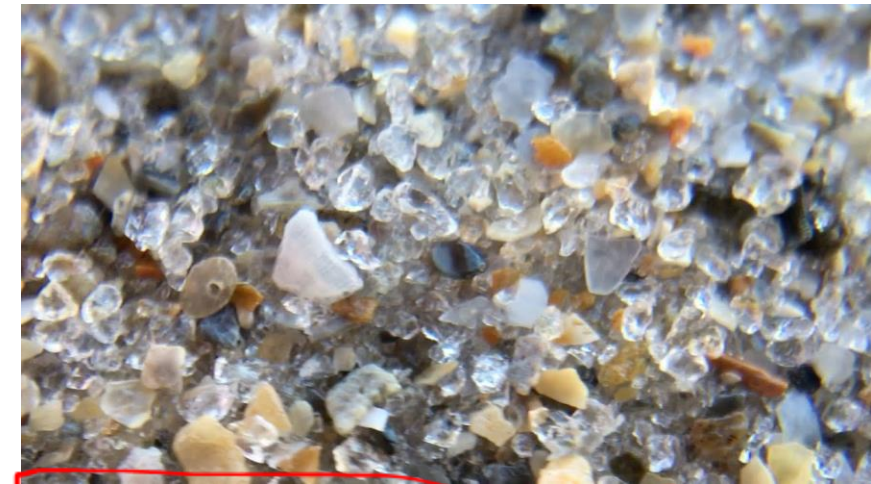
$$p =$$



Param Estimation With a Prior == Inference

```
observations = [1.677, 3.812, 1.463, 2.641, 1.256, 1.678, 1.157,  
1.146, 1.323, 1.029, 1.238, 1.018, 1.171, 1.123, 1.074, 1.652,  
1.873, 1.314, 1.309, 3.325, 1.045, 2.271, 1.305, 1.277, 1.114,  
1.391, 3.728, 1.405, 1.054, 2.789, 1.019, 1.218, 1.033, 1.362,  
1.058, 2.037, 1.171, 1.457, 1.518, 1.117, 1.153, 2.257, 1.022,  
1.839, 1.706, 1.139, 1.501, 1.238, 2.53, 1.414, 1.064, 1.097,  
1.261, 1.784, 1.196, 1.169, 2.101, 1.132, 1.193, 1.239, 1.518,  
2.764, 1.053, 1.267, 1.015, 1.789, 1.099, 1.25, 1.253, 1.418,  
1.494, 1.015, 1.459, 2.175, 2.044, 1.551, 4.095, 1.396, 1.262,  
1.351, 1.121, 1.196, 1.391, 1.305, 1.141, 1.157, 1.155, 1.103,  
1.048, 1.918, 1.889, 1.068, 1.811, 1.198, 1.361, 1.261, 4.093,  
2.925, 1.133, 1.573]
```

```
def estimate_alpha(observations):  
    print('your code here')
```



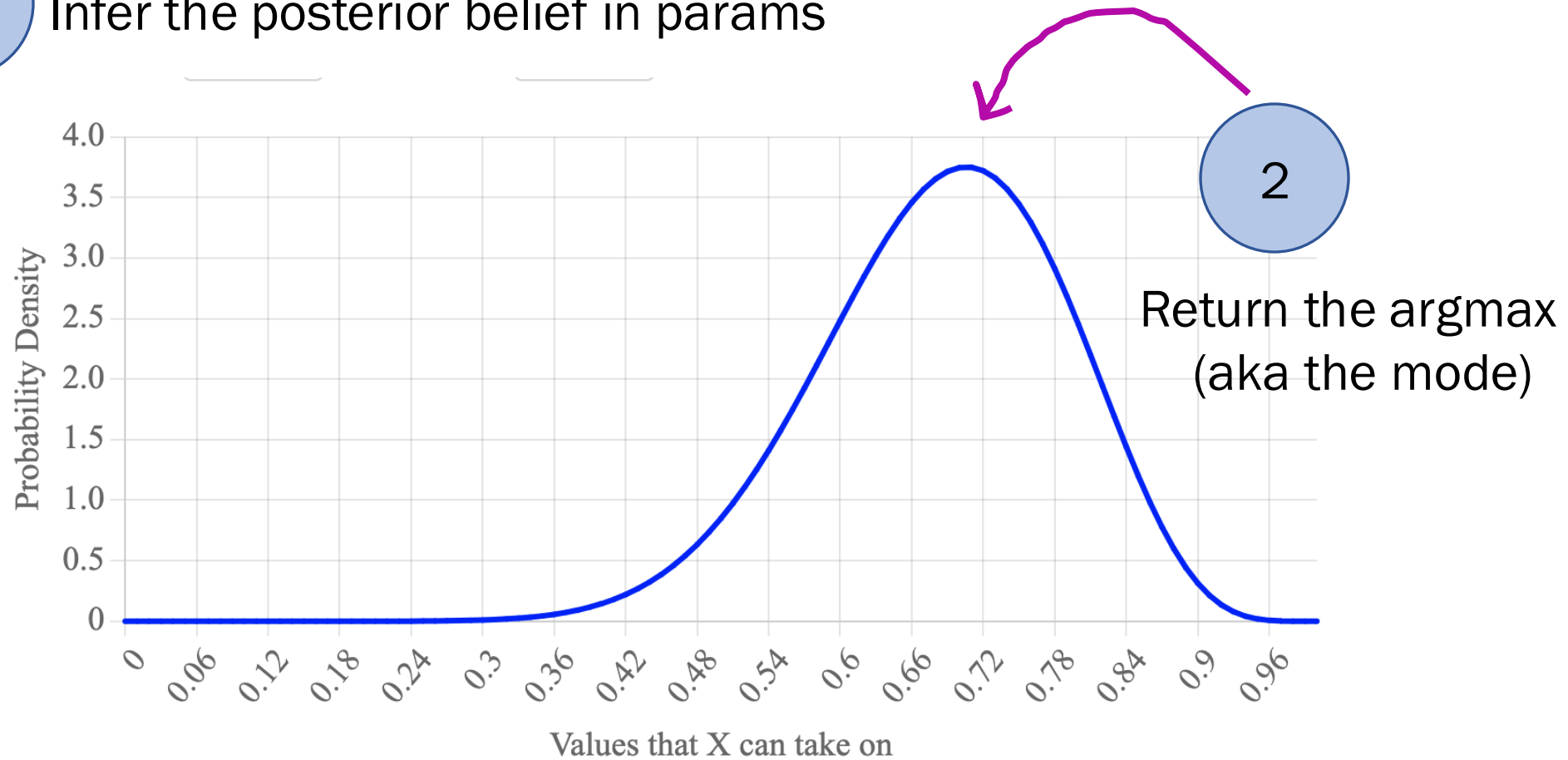
We know sand is distributed as a pareto with PDF

$$f(x) = \frac{\alpha}{x^{\alpha+1}}$$

Prior: $\alpha \sim N(\mu = 2.5, \sigma^2 = 3)$

Maximum A Posteriori (MAP)

1 Infer the posterior belief in params



You need to know MLE and Inference.
MAP is something you should recognize!

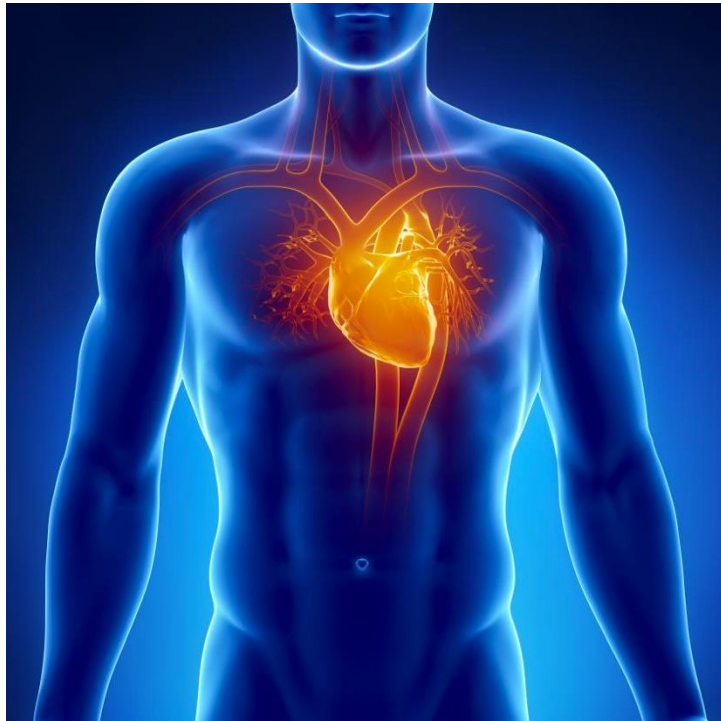
It is time....

Today a very special MLE problem

Classification

Example Datasets

Heart



Ancestry



Netflix



Training Data

Training Data: assignments all random variables \mathbf{X} and Y

Assume IID data:


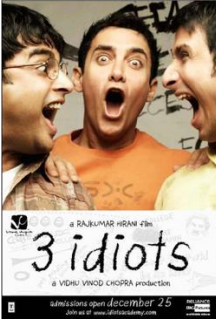

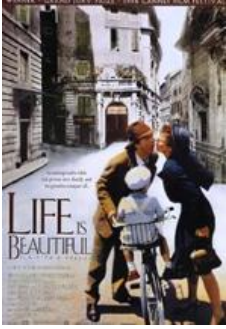
n training datapoints

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

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


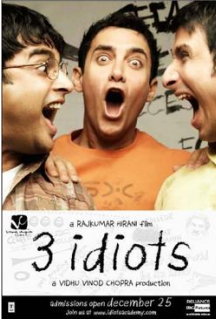

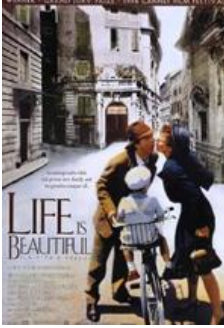
Each datapoint has m features and a single output

Single Feature Value

	Movie 1	Movie 2	...	Movie m	Output
			...		
User 1	1	0		1	1
User 2	1	1		0	0
			⋮		⋮
User n	0	0		1	1


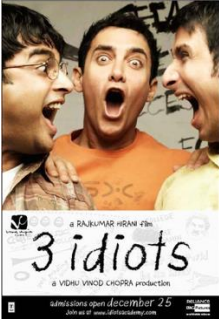


$(\mathbf{x}^{(i)}, y^{(i)})$ such that $1 \leq i \leq n$

Single Feature Value

	Movie 1	Movie 2	...	Movie m	Output
			...		
User 1	1	0		1	1
User 2	1	1		0	0
			⋮		⋮
User n	0	0		1	1


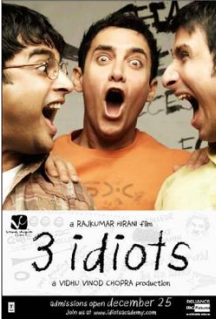


$(\mathbf{x}^{(i)}, y^{(i)})$ such that $1 \leq i \leq n$

Single Feature Value

	Movie 1	Movie 2	...	Movie m	Output
			...		
User 1	1	0		1	1
User 2	1	1		0	0
			⋮		⋮
User n	0	0		1	1


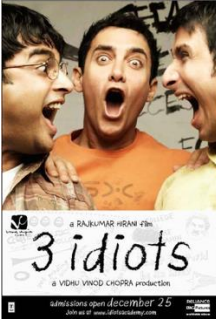

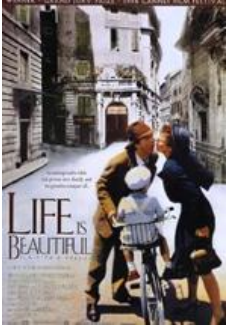
$(\mathbf{x}^{(i)}, y^{(i)})$ such that $1 \leq i \leq n$

Single Feature Value

	Movie 1	Movie 2	...	Movie m	Output
			...		
User 1	1	0		1	1
User 2	1	1		0	0
			⋮		⋮
User n	0	0		1	1

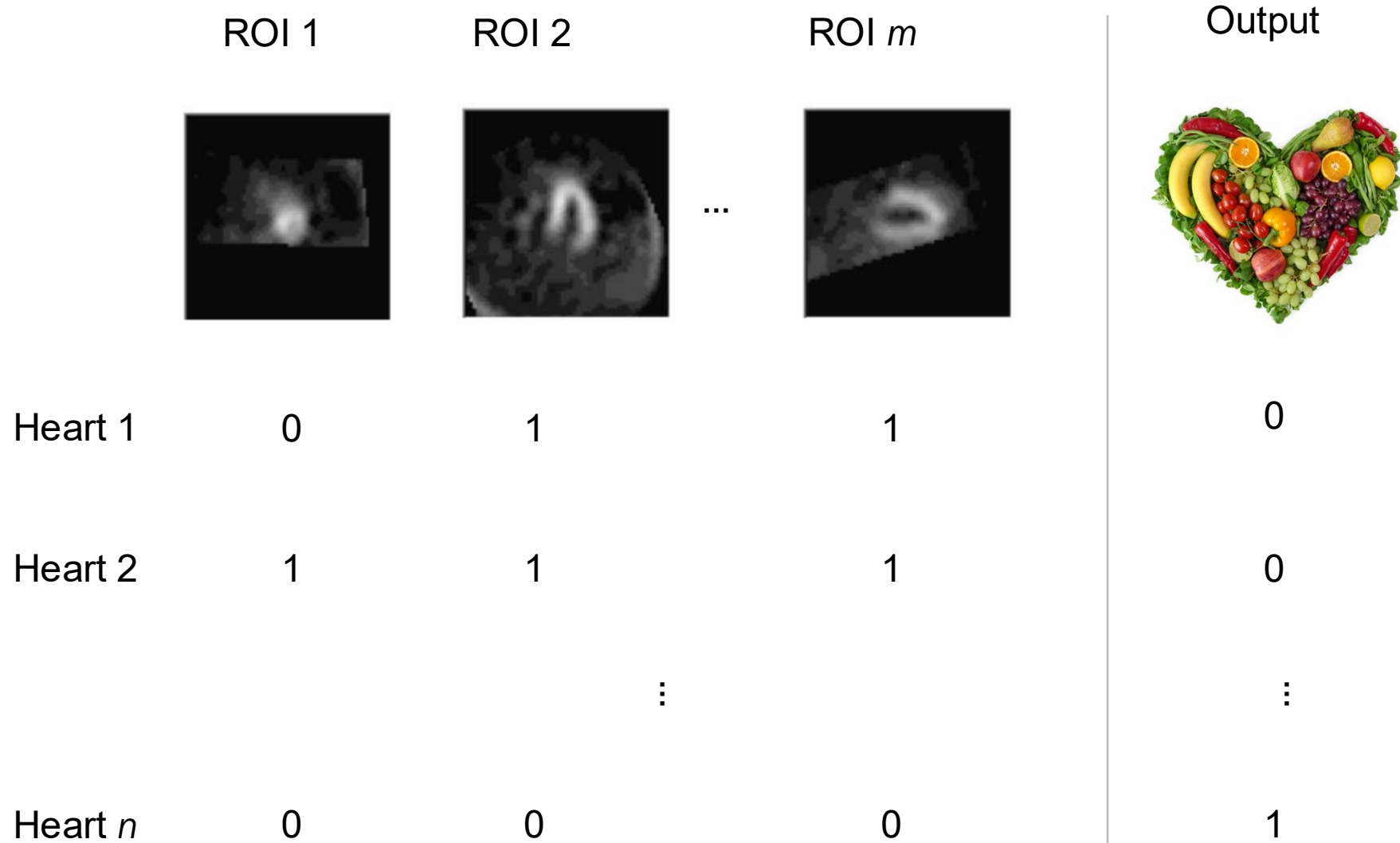
$(\mathbf{x}^{(i)} \quad y^{(i)})$ such that $1 \leq i \leq n$

Single Feature Value

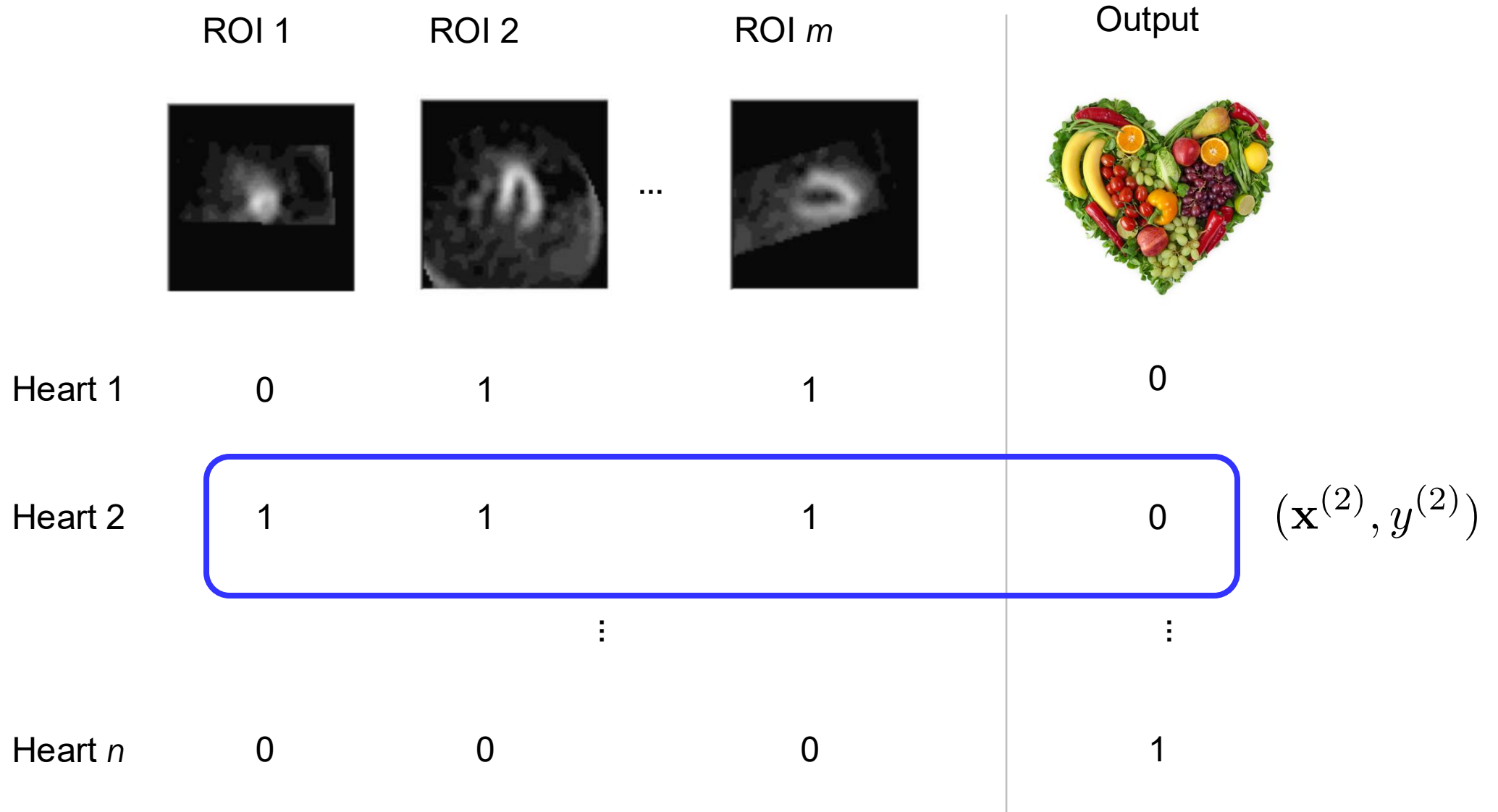
	Movie 1	Movie 2	...	Movie m	Output
			...		
User 1	1	0		1	1
User 2	1	1		0	0
			⋮		⋮
User n	0	0		1	1

In general: $\mathbf{x}_j^{(i)}$ In this case: $\mathbf{x}_m^{(2)}$

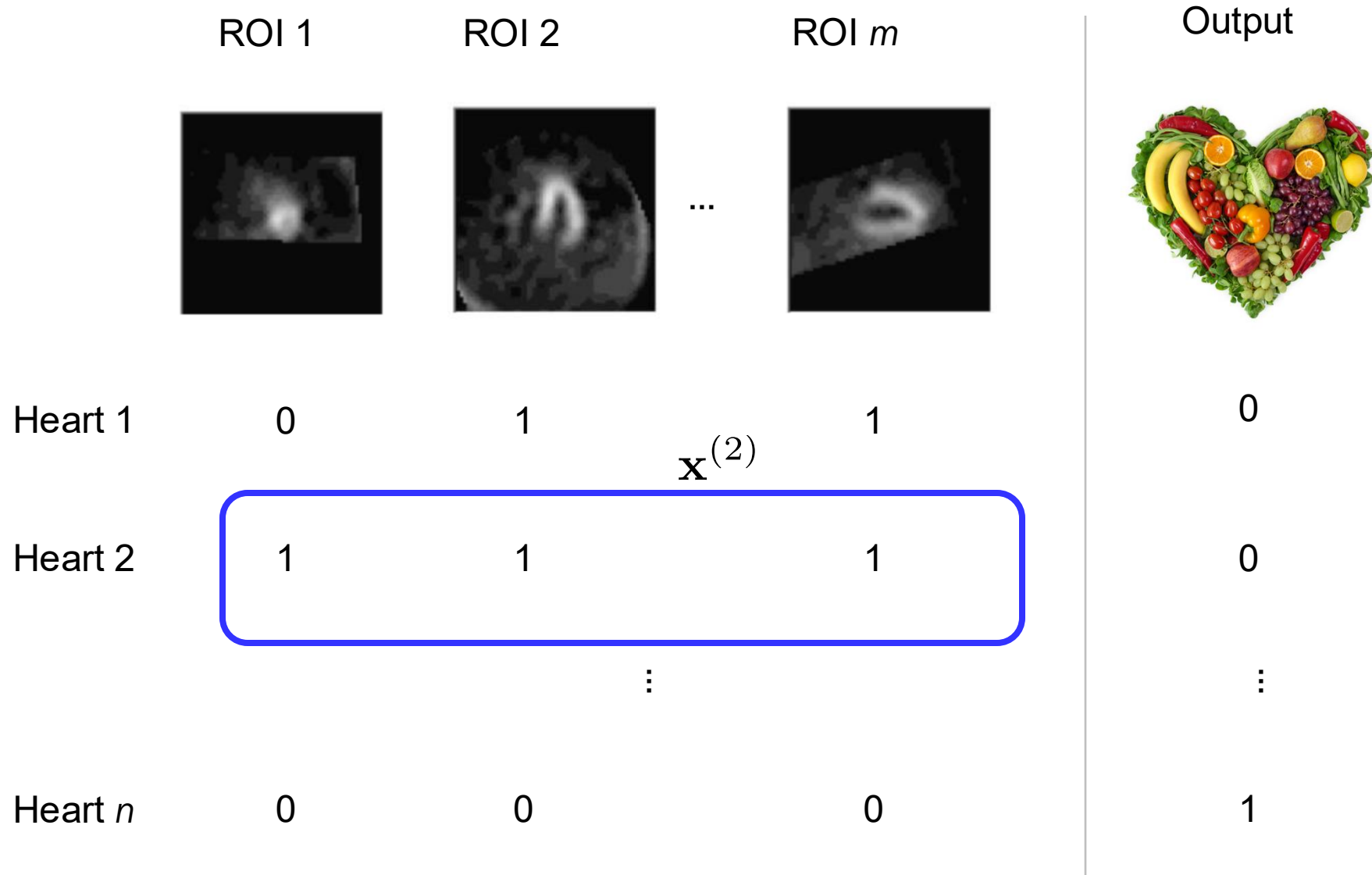
Healthy Heart Classifier



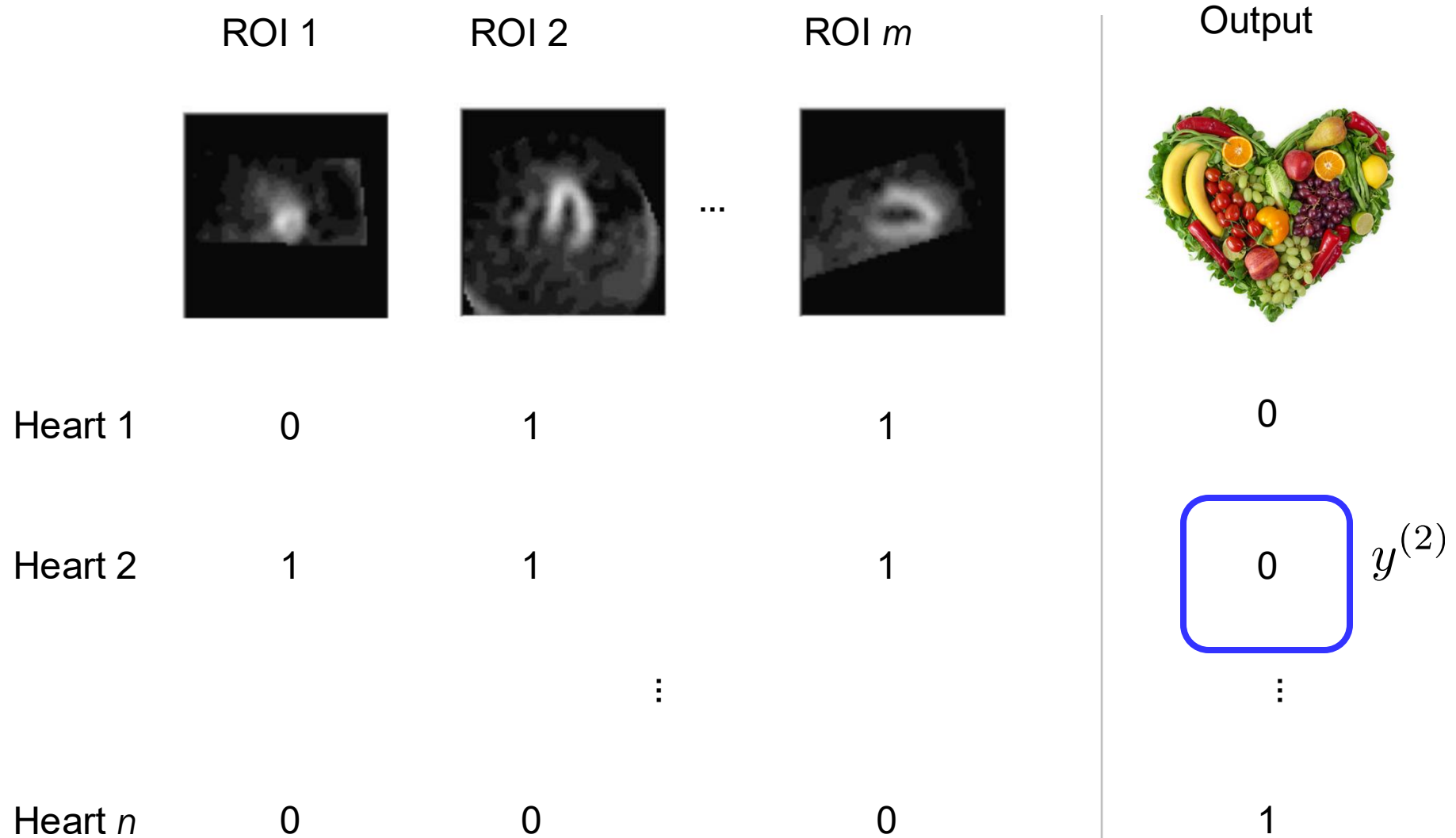
Healthy Heart Classifier



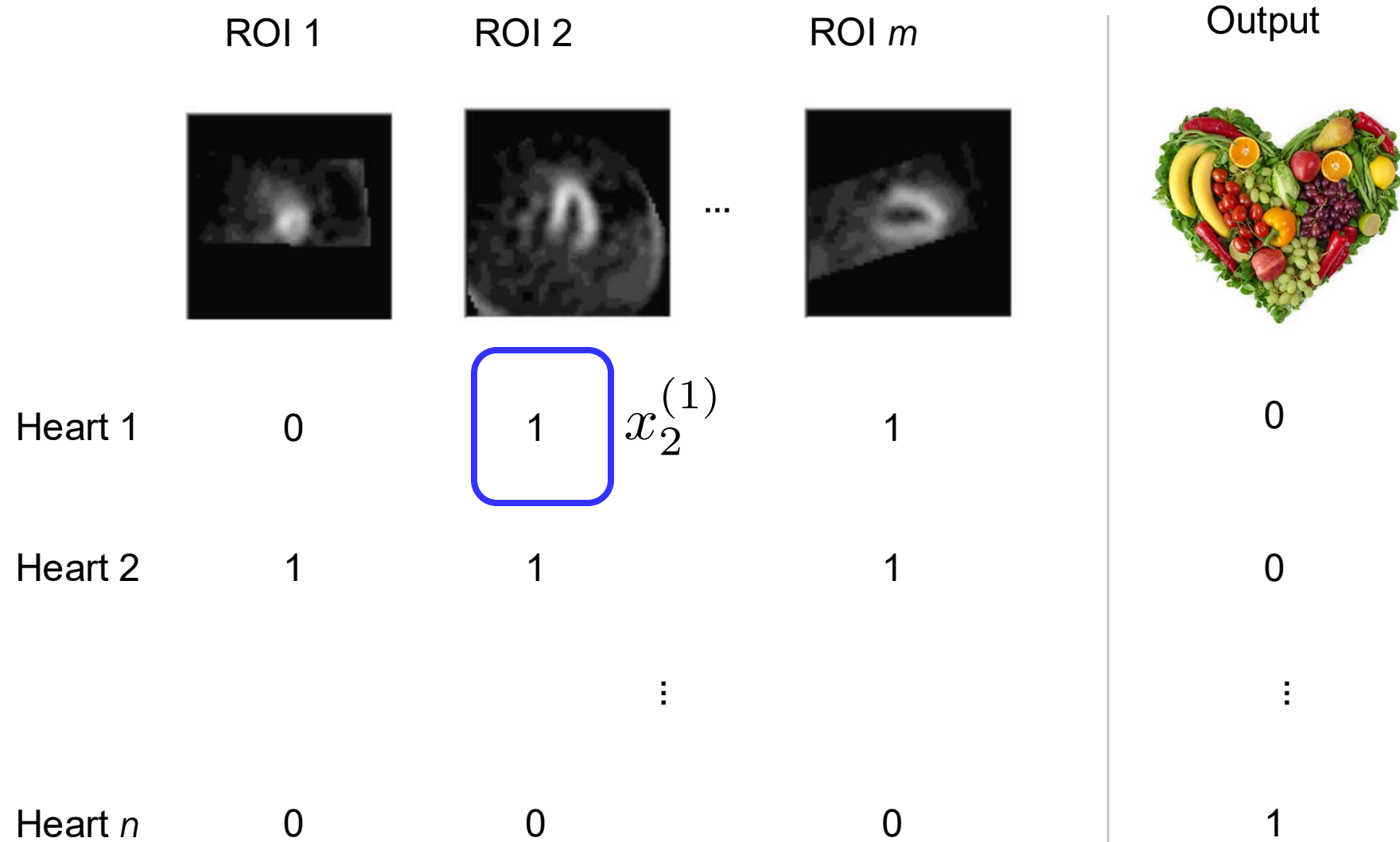
Healthy Heart Classifier



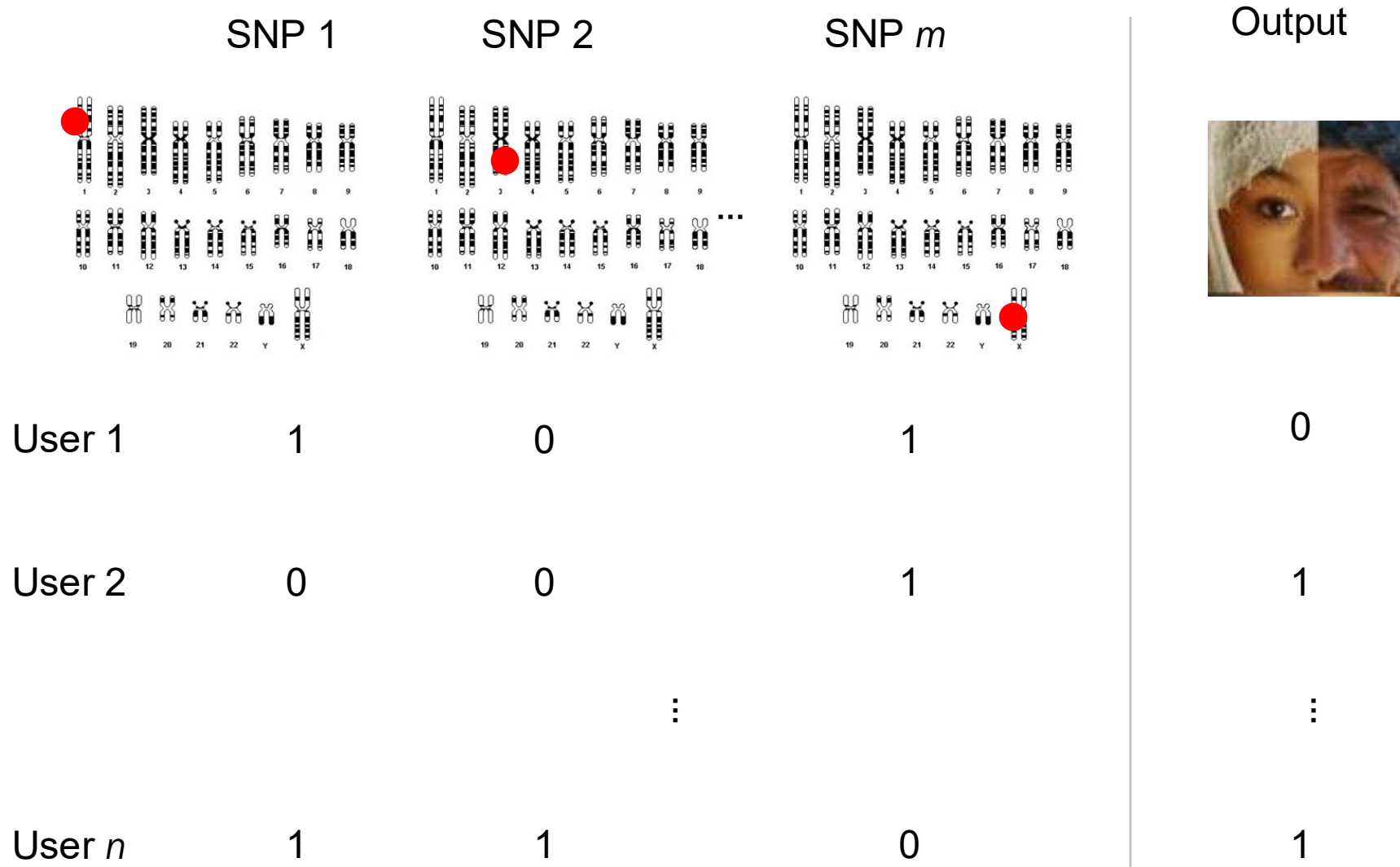
Healthy Heart Classifier







Healthy Heart Classifier



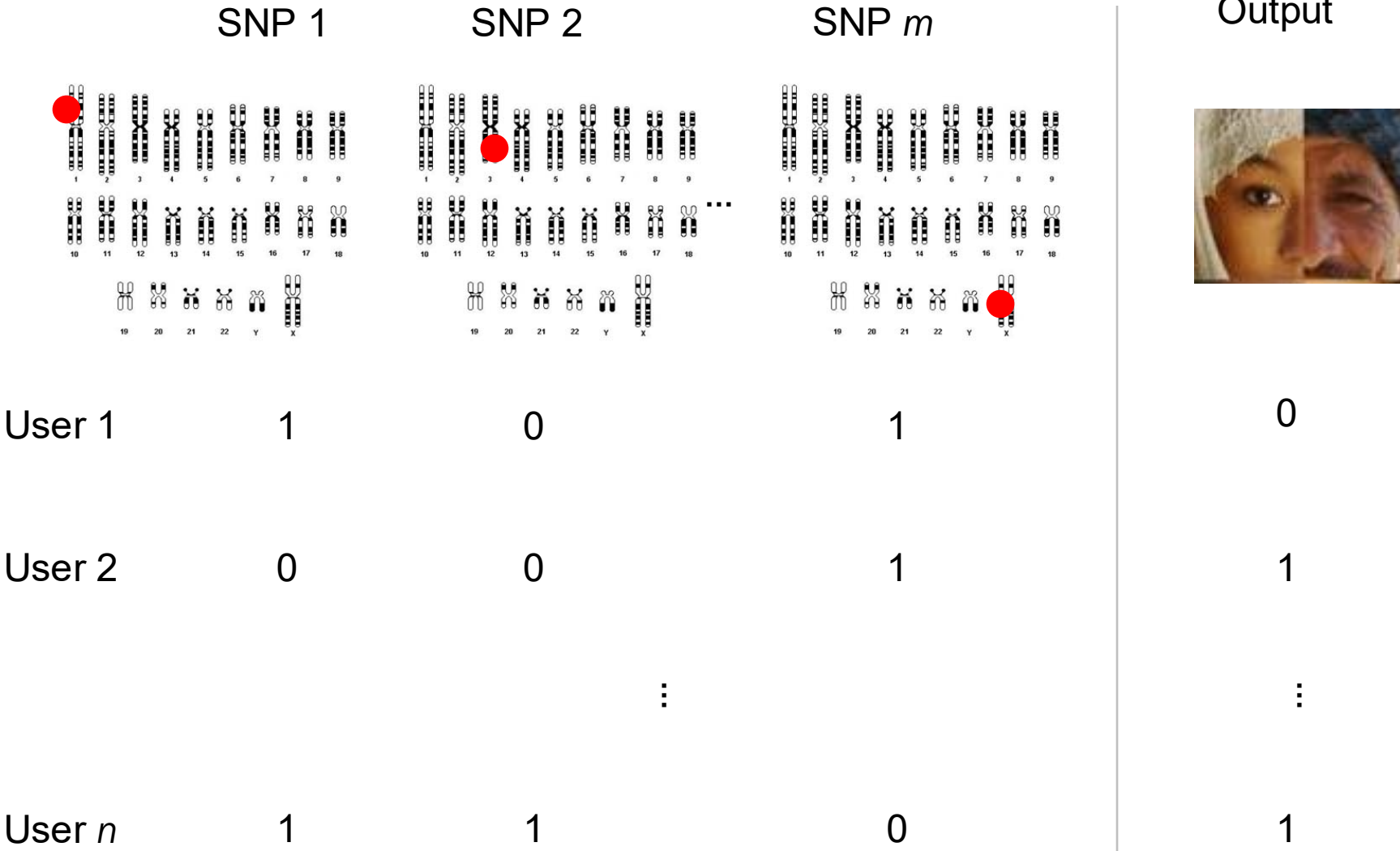
Ancestry Classifier



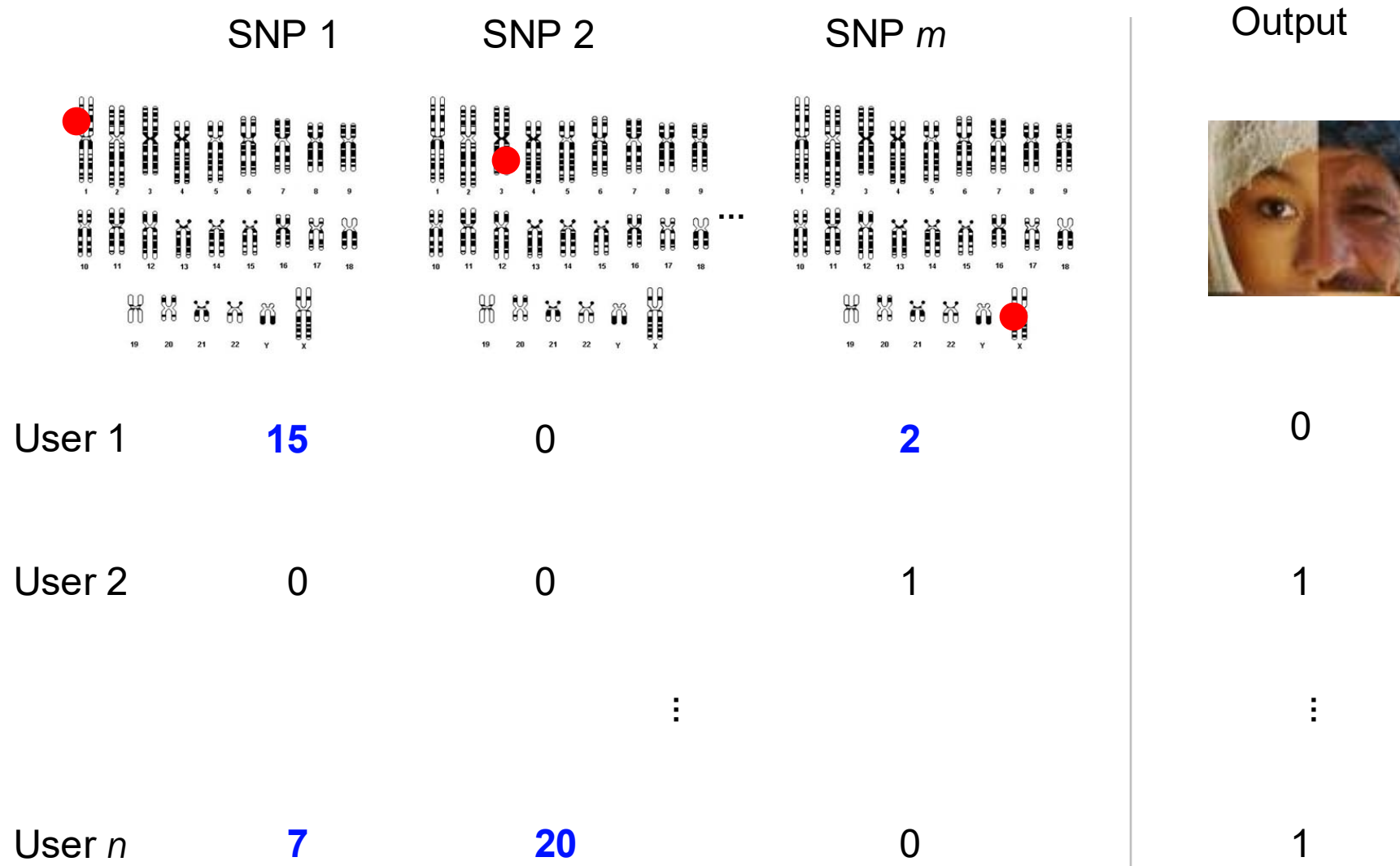
Aside Predicting Real Numbers is Regression

	Opposing team ELO	Points in last game	At Home?	Output
				 # Points
Game 1	84	105	1	120
Game 2	90	102	0	95
		⋮		⋮
Game n	74	120	0	115

Ancestry Classifier

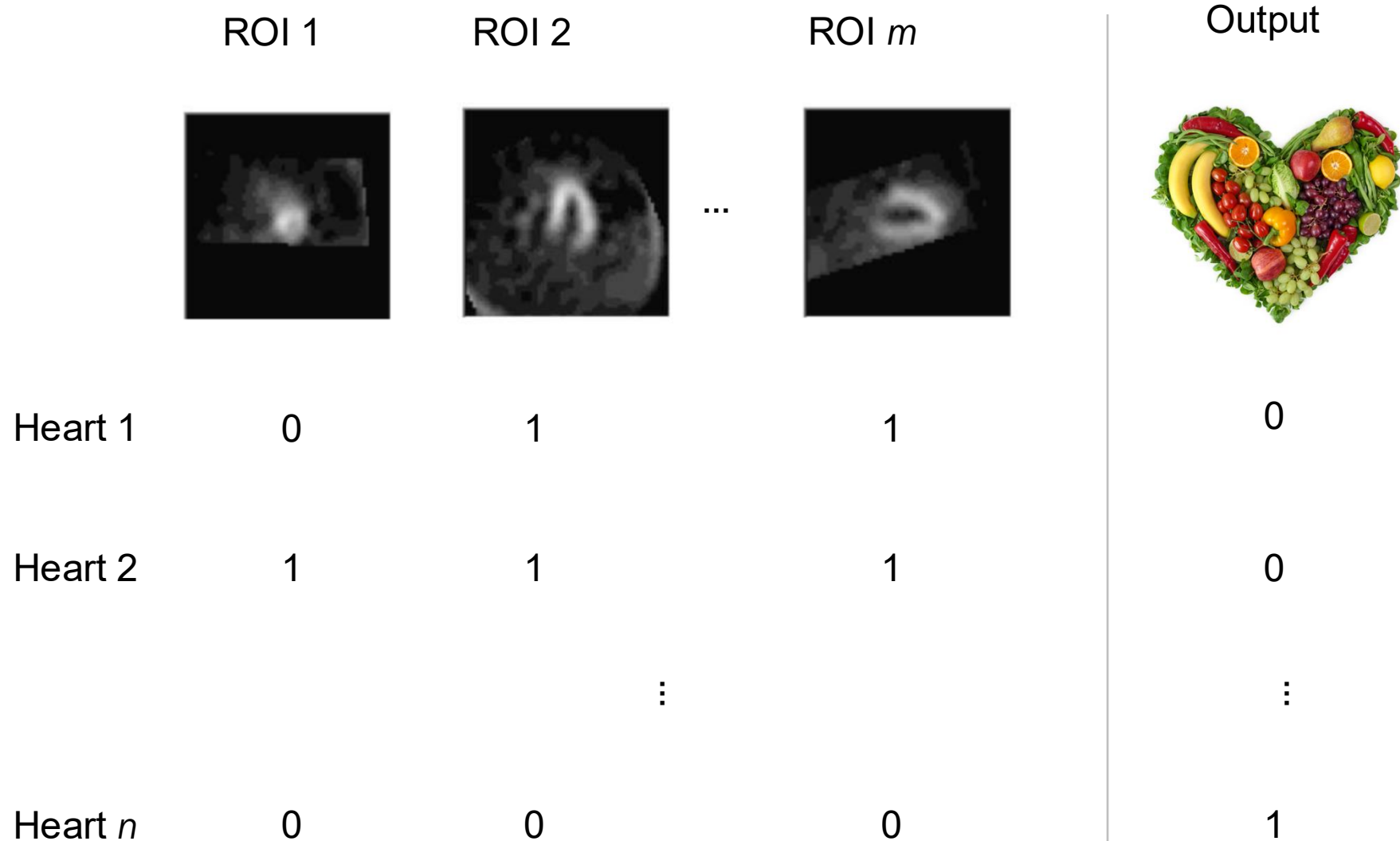


Still Classification



Classification

Healthy Heart Classifier

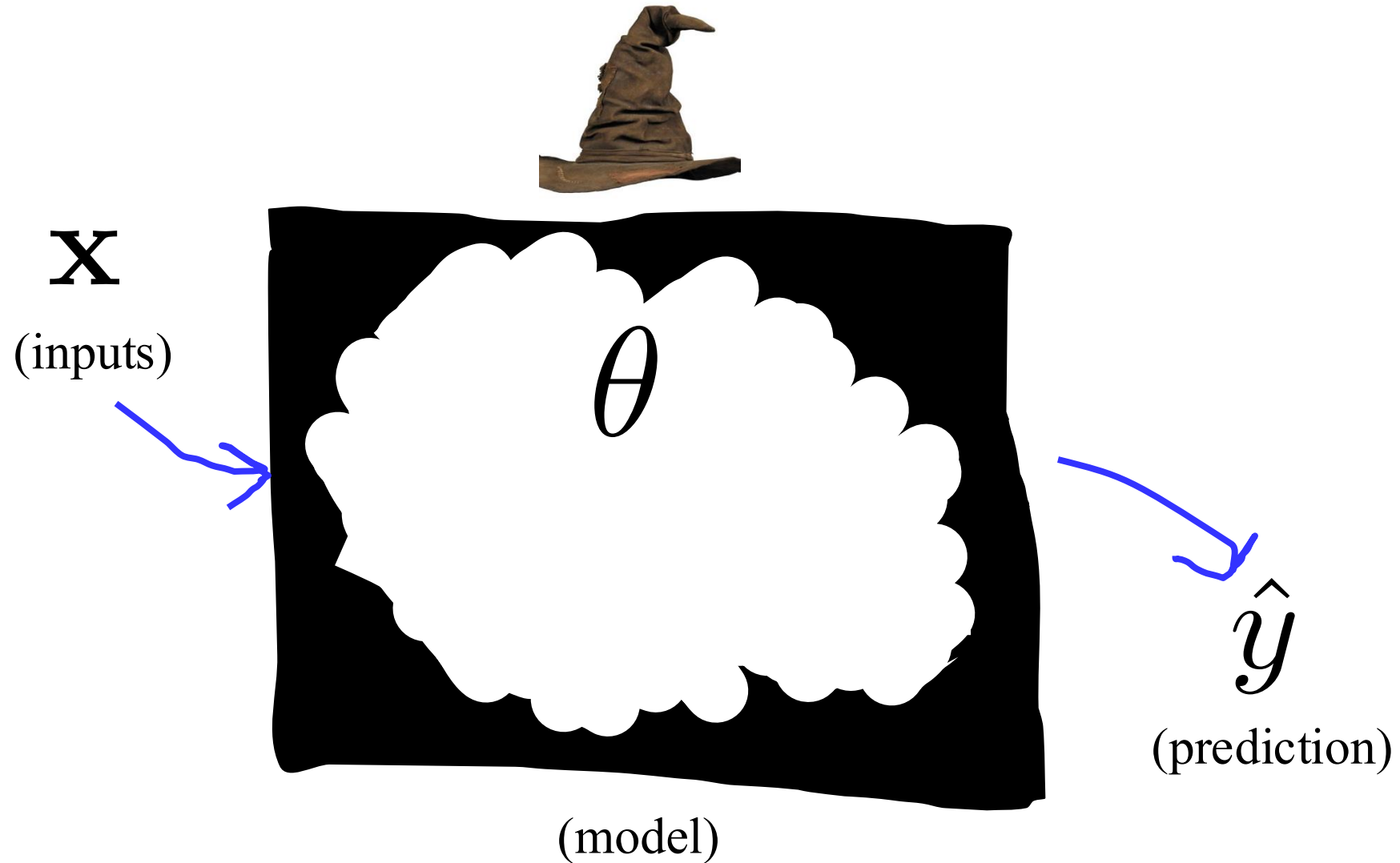


Classification is Building a Harry Potter Hat

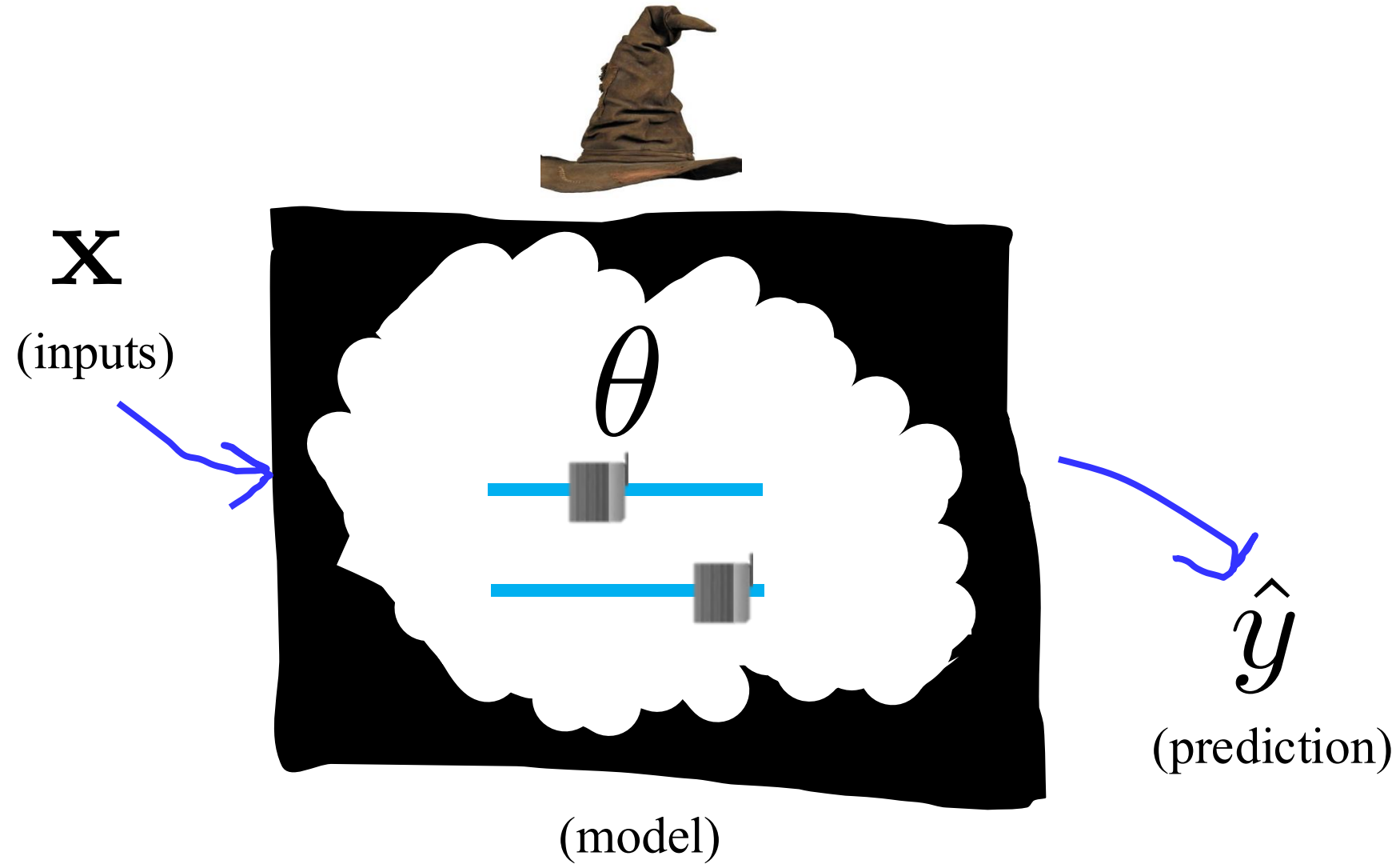


$$\mathbf{x} = [0, 1, \dots, 1]$$

Machine Learning for Classification



Machine Learning for Classification



Logistic Regression

Chapter 1: Big Picture

From Naïve Bayes to Logistic Regression

In classification we care about $P(Y = 1 | \mathbf{X} = \mathbf{x})$

Lets build a machine
that can you can put
 \mathbf{x} into, which then
spits out

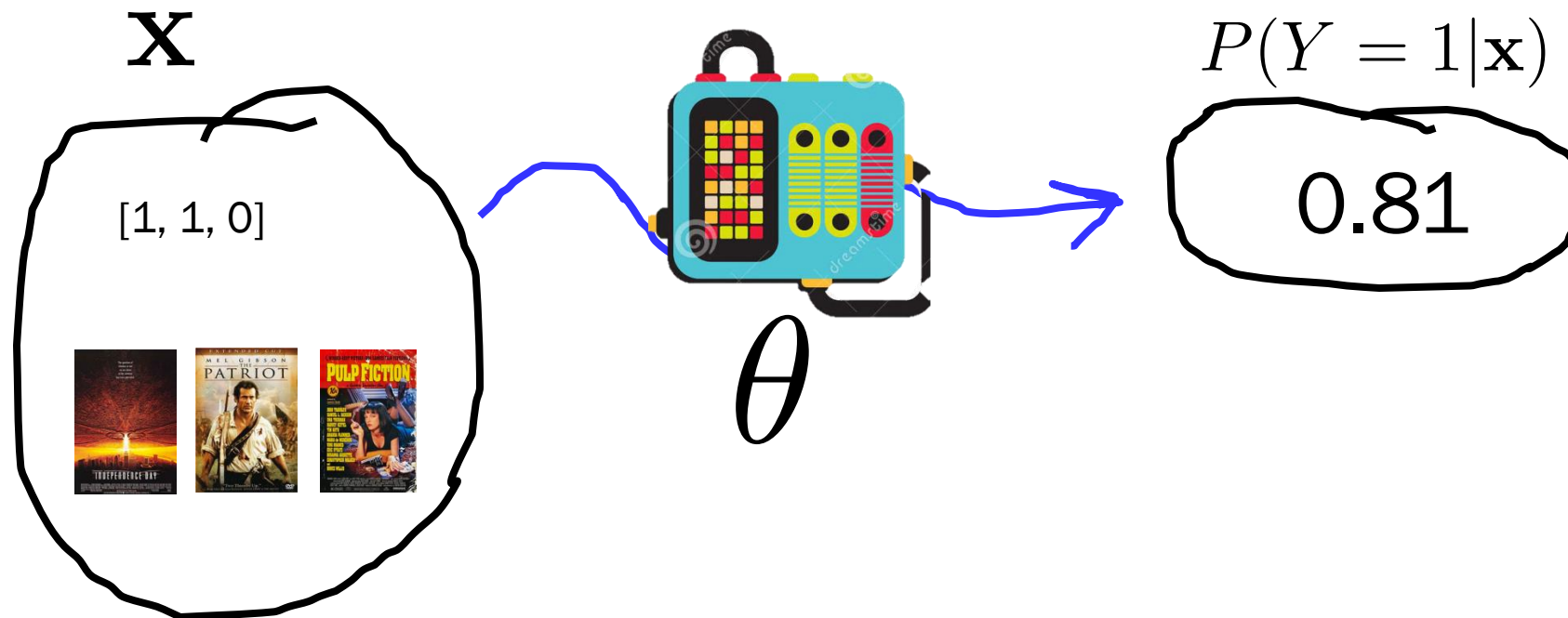
$$P(Y = 1 | \mathbf{X} = \mathbf{x})$$



Logistic Regression Assumption

Could we compute $P(Y = 1 | \mathbf{X} = \mathbf{x})$ via a machine?

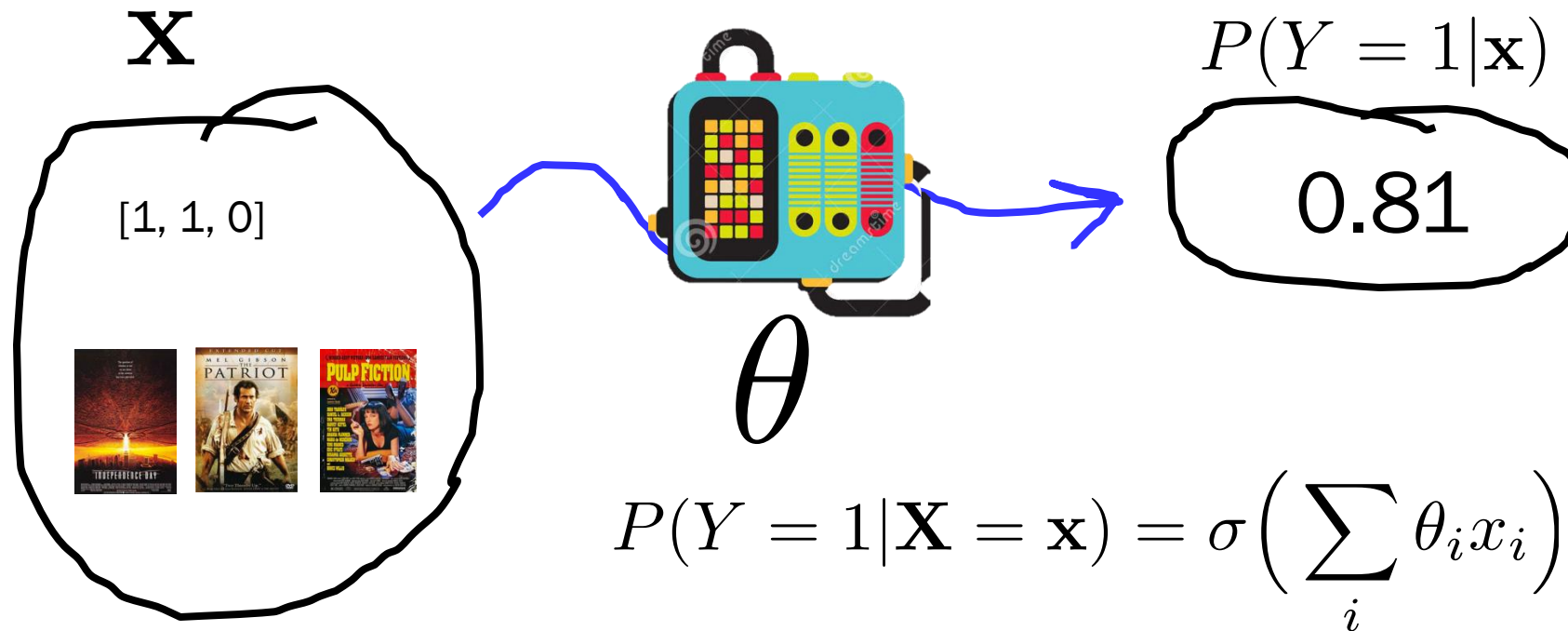
Welcome our friend: logistic regression!



Logistic Regression Assumption

Could we compute $P(Y = 1|\mathbf{X} = \mathbf{x})$ via a machine?

Welcome our friend: logistic regression!

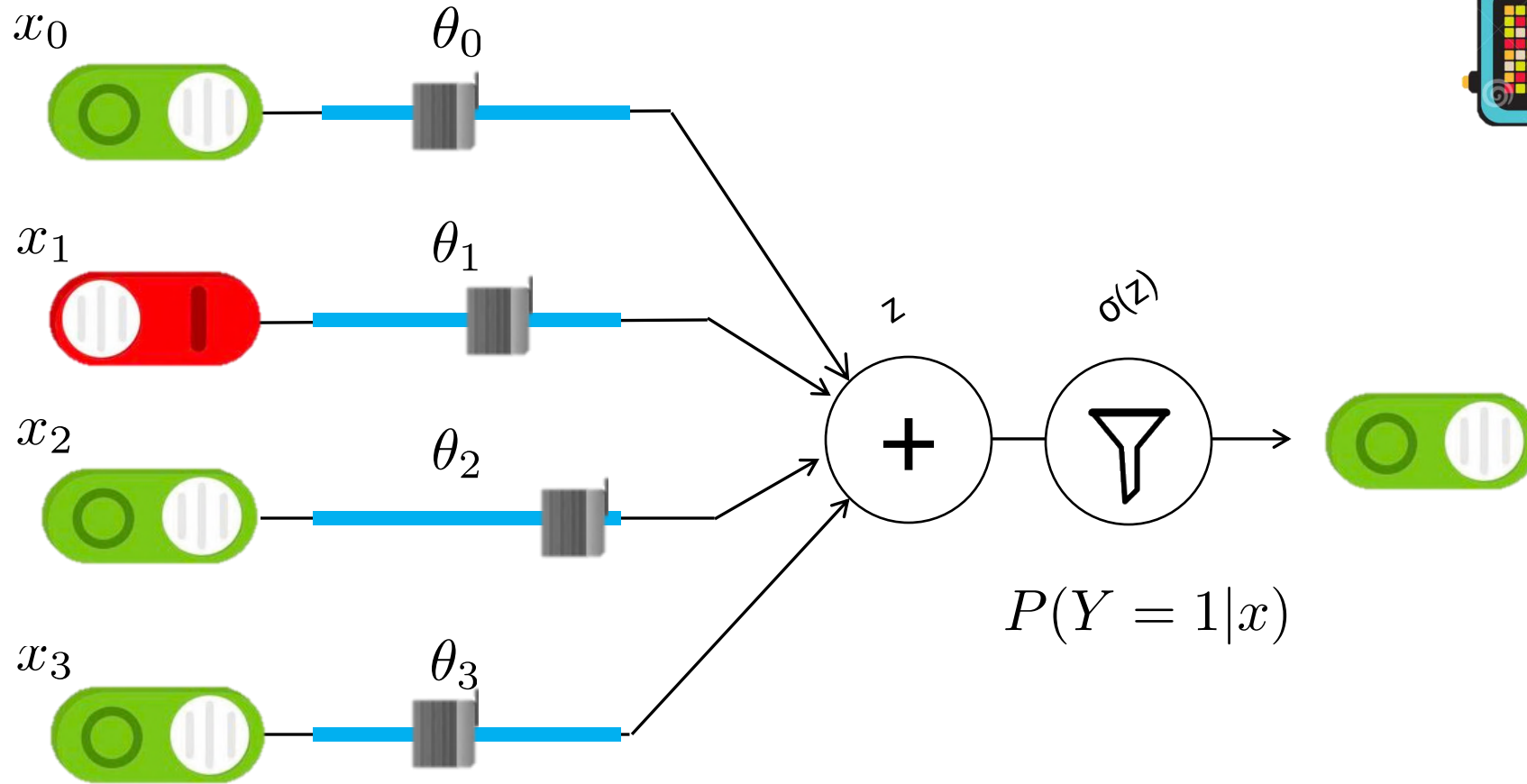


Logistic Regression Assumption



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

Logistic Regression



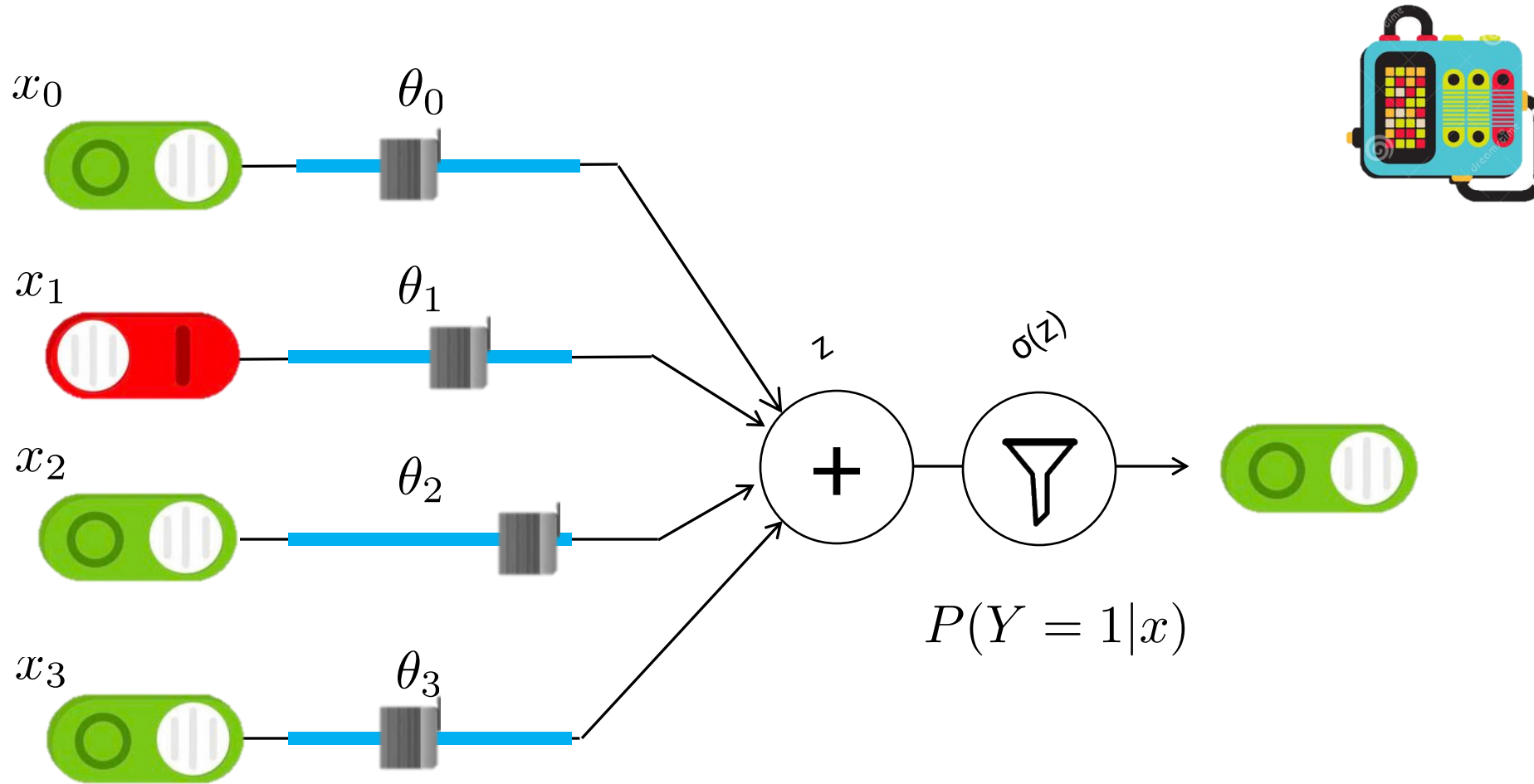
$$P(Y = 1 | \mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Logistic Regression



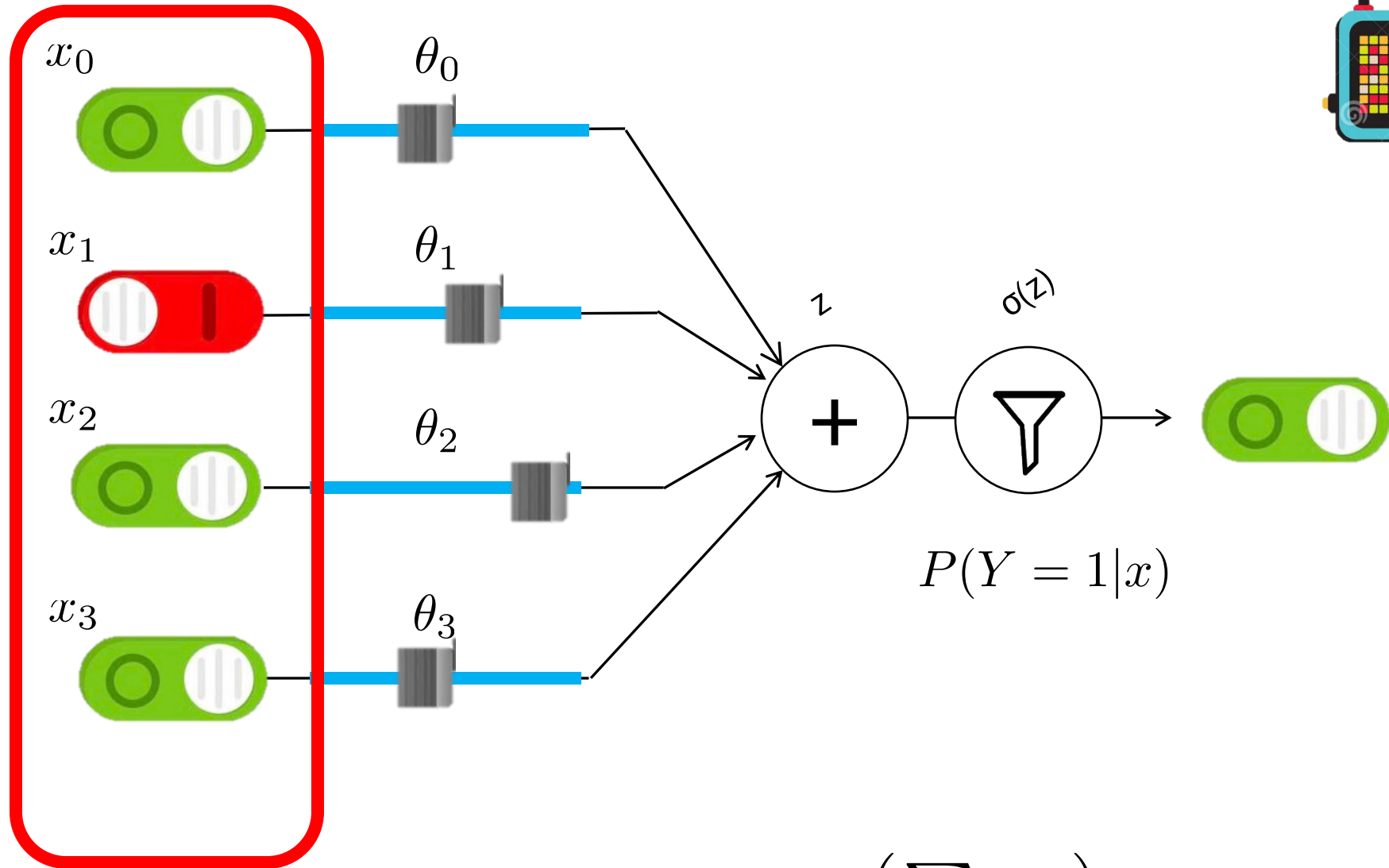
$$P(Y = 1 | \mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Logistic Regression Cartoon



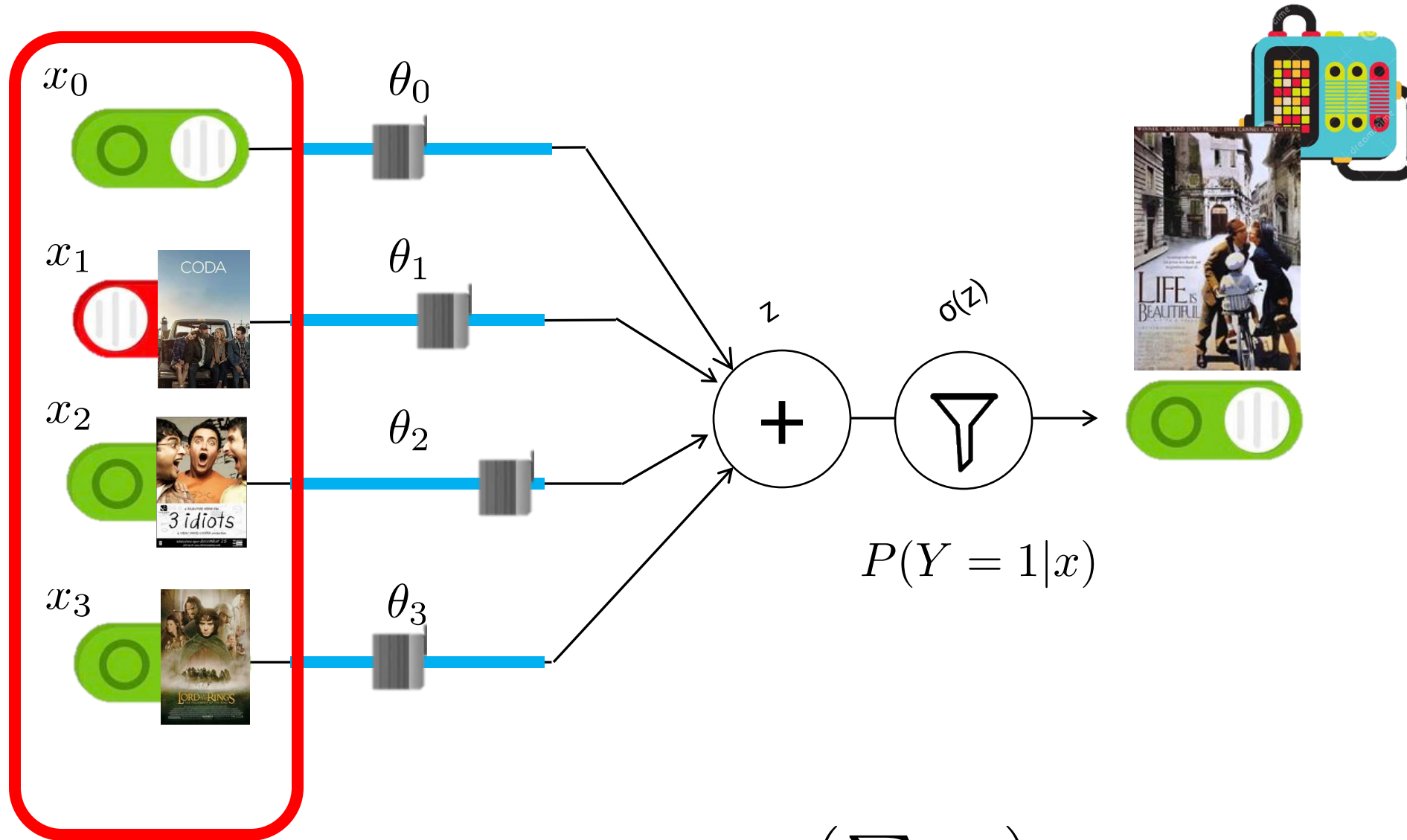
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

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



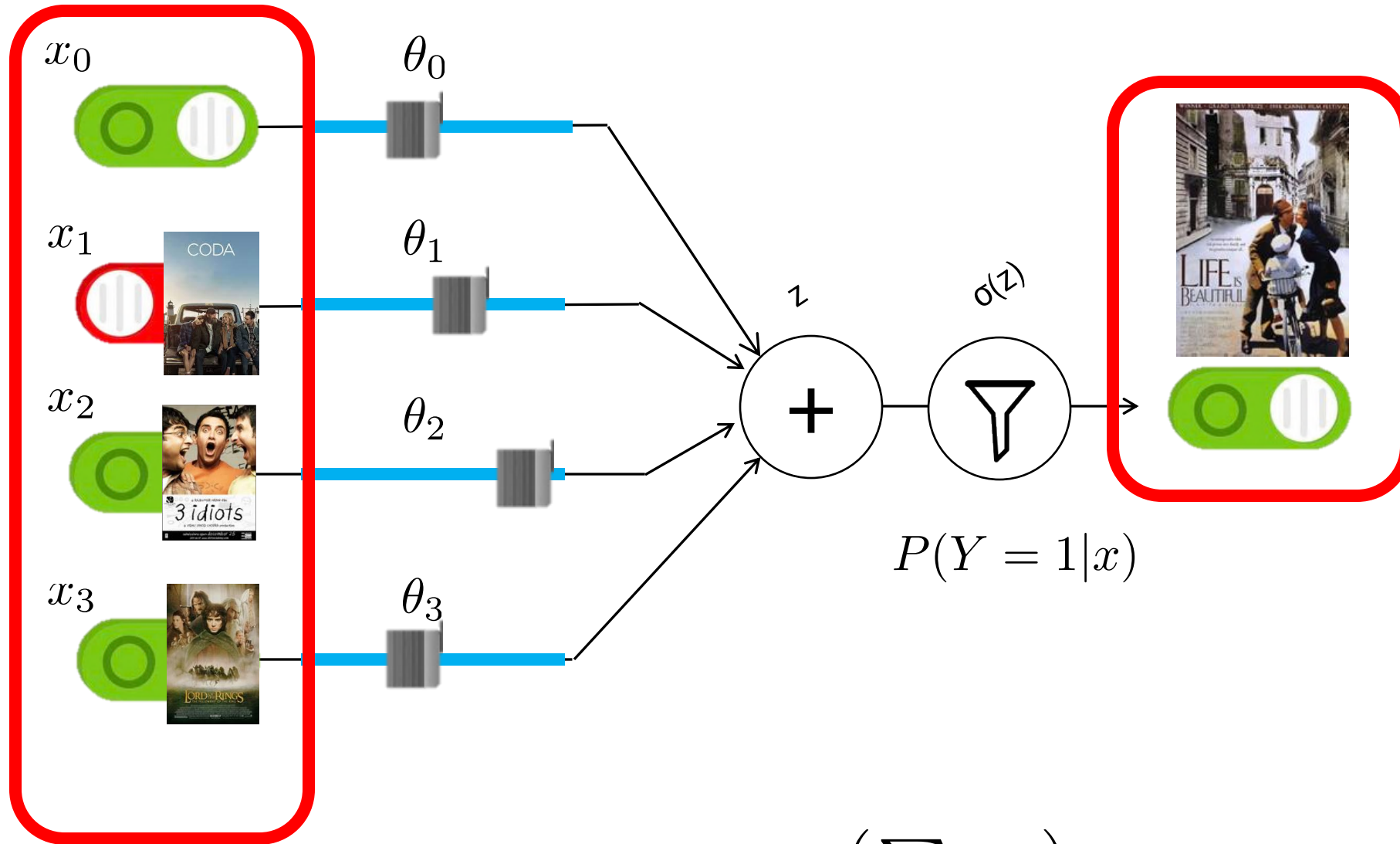
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Inputs



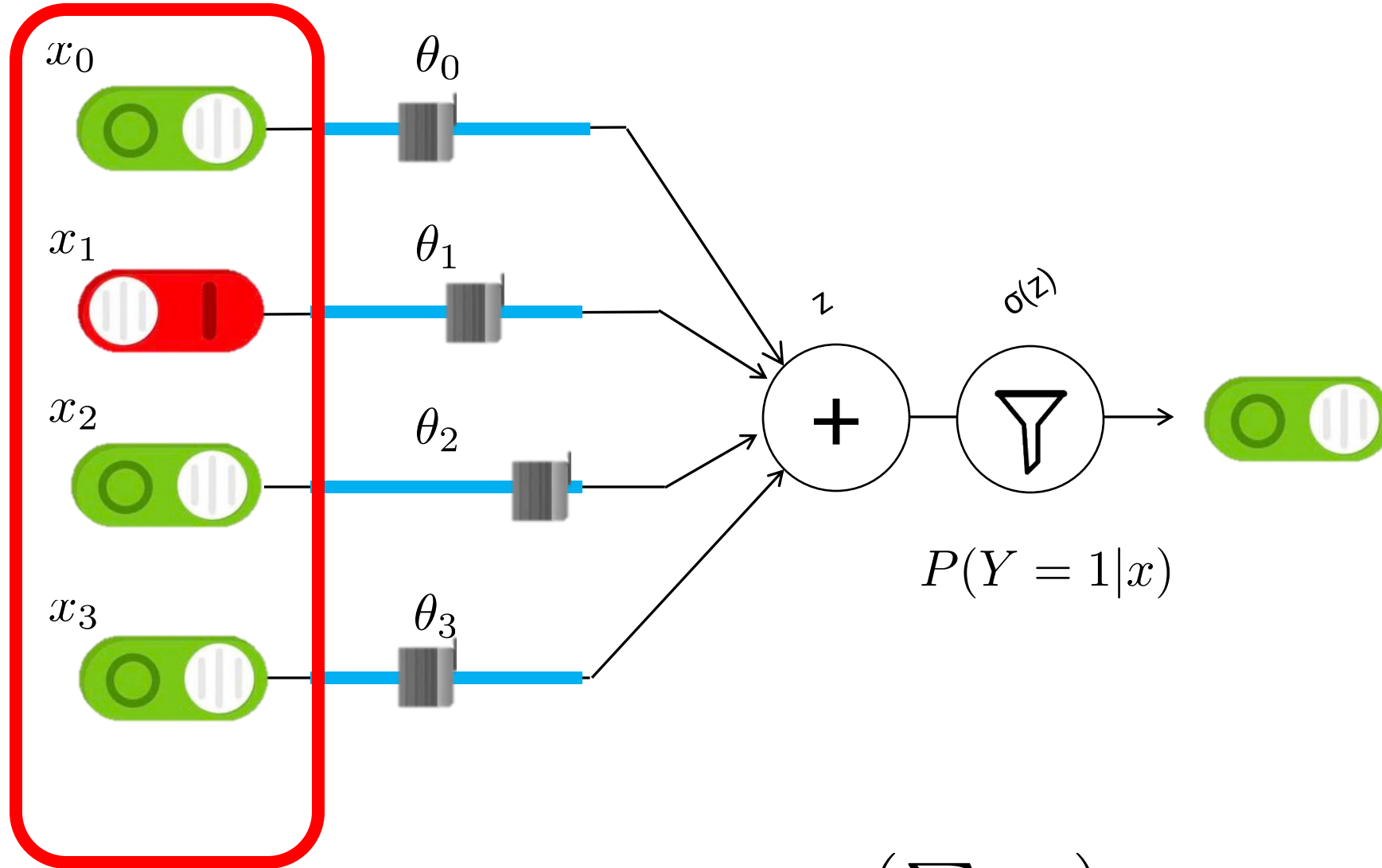
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Inputs + Output



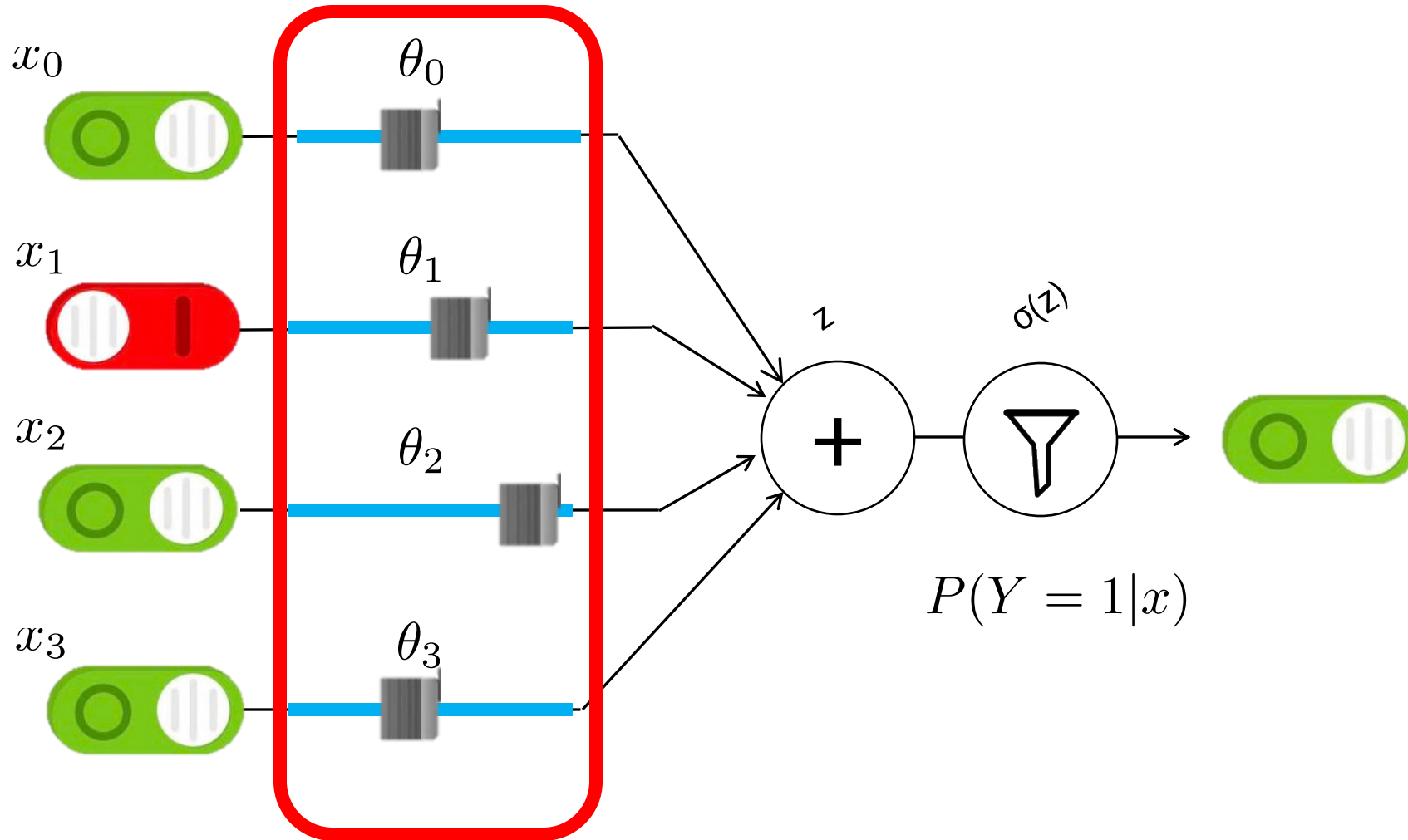
$$P(Y = 1 | \mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Inputs



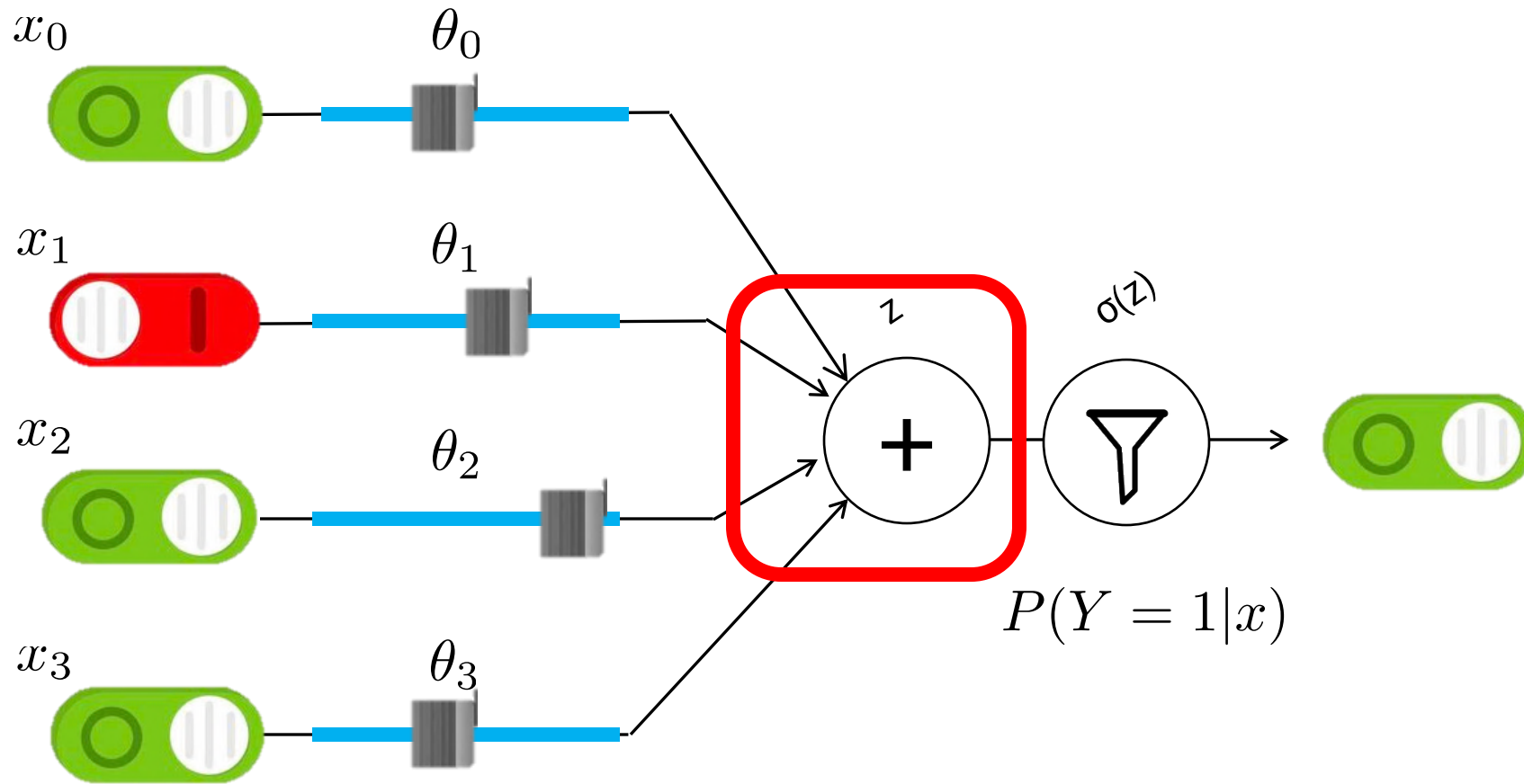
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Weights



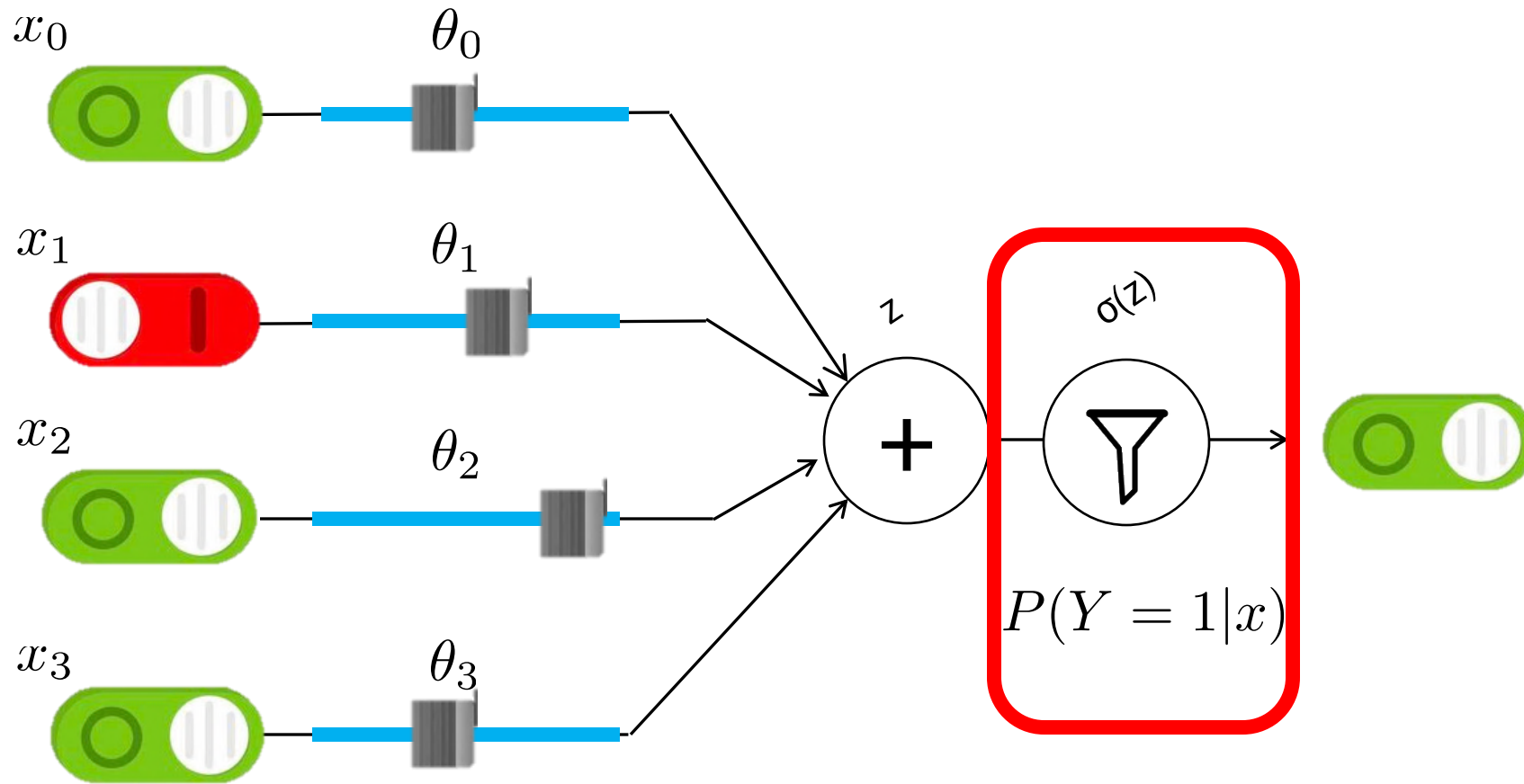
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Weighted Sum



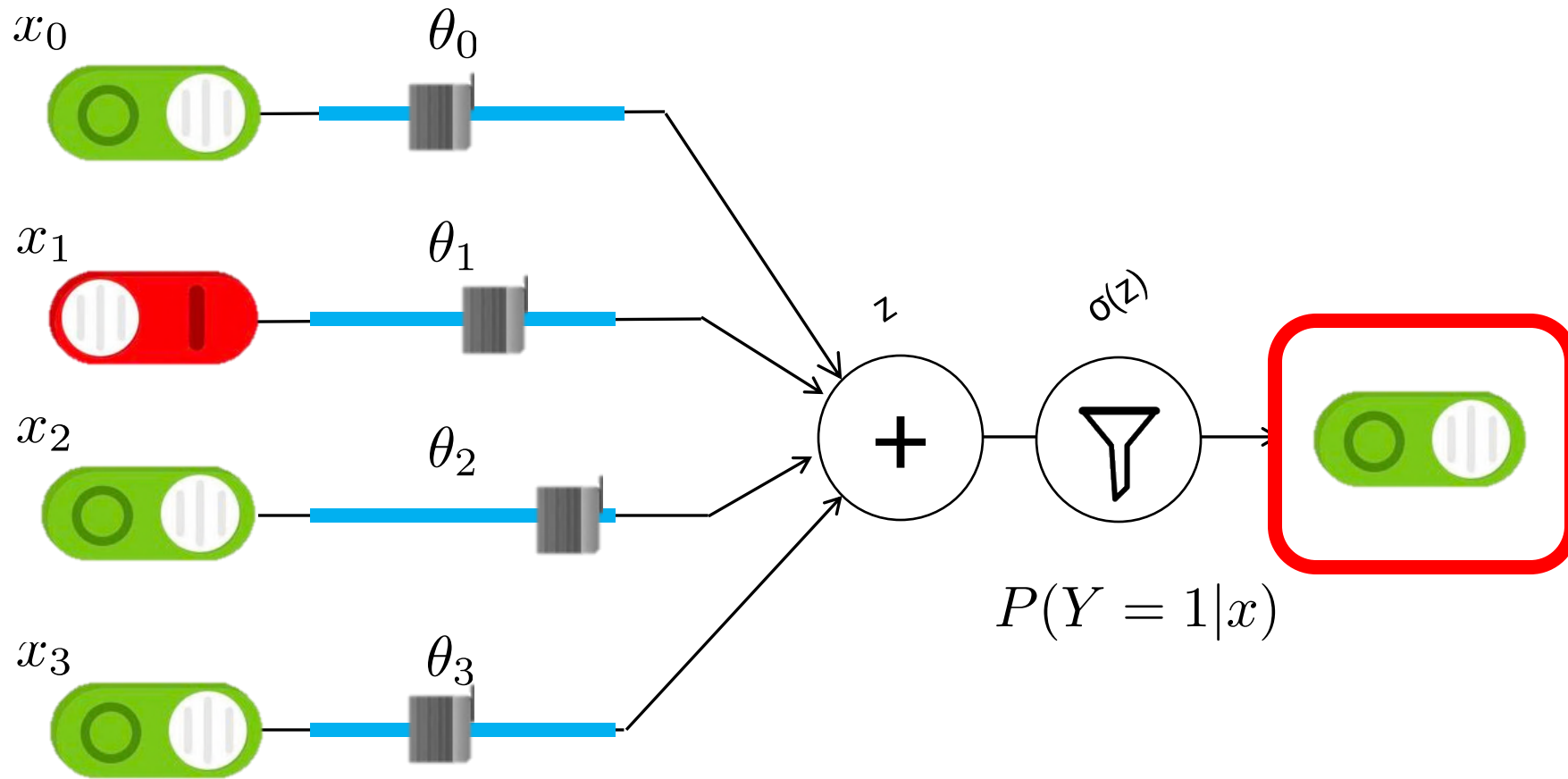
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Squashing Function



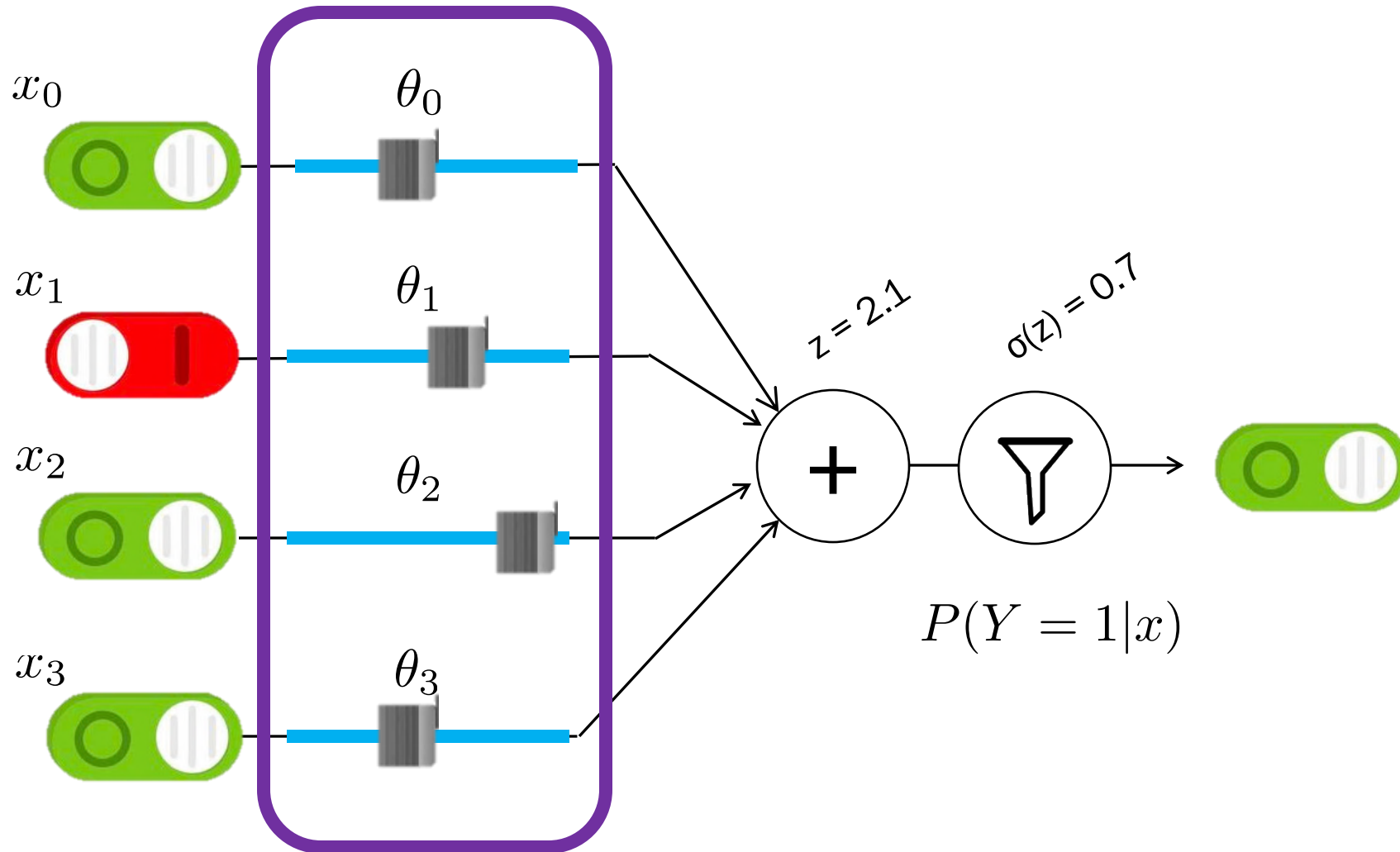
$$P(Y = 1 | \mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Prediction



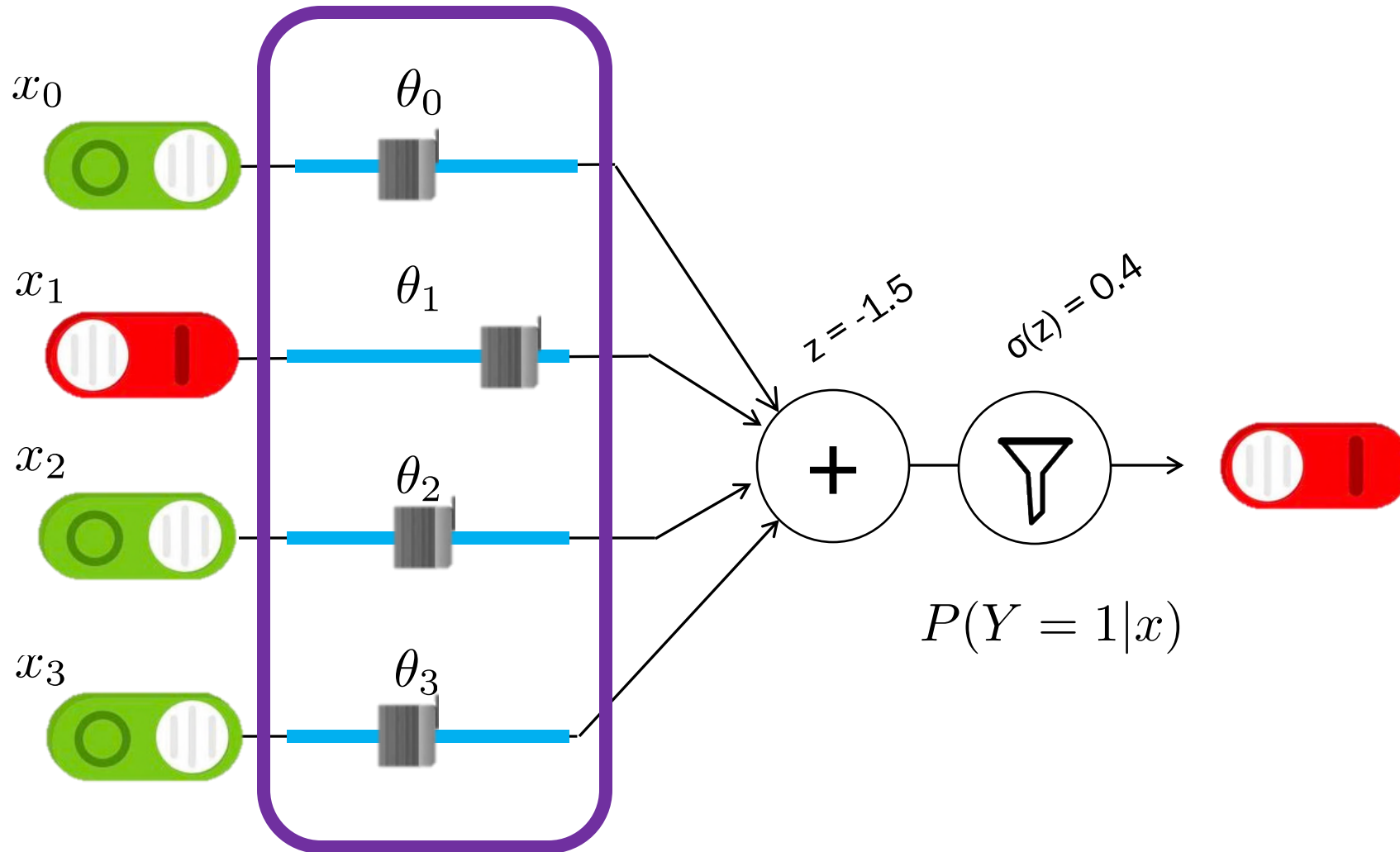
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Parameters Affect Prediction



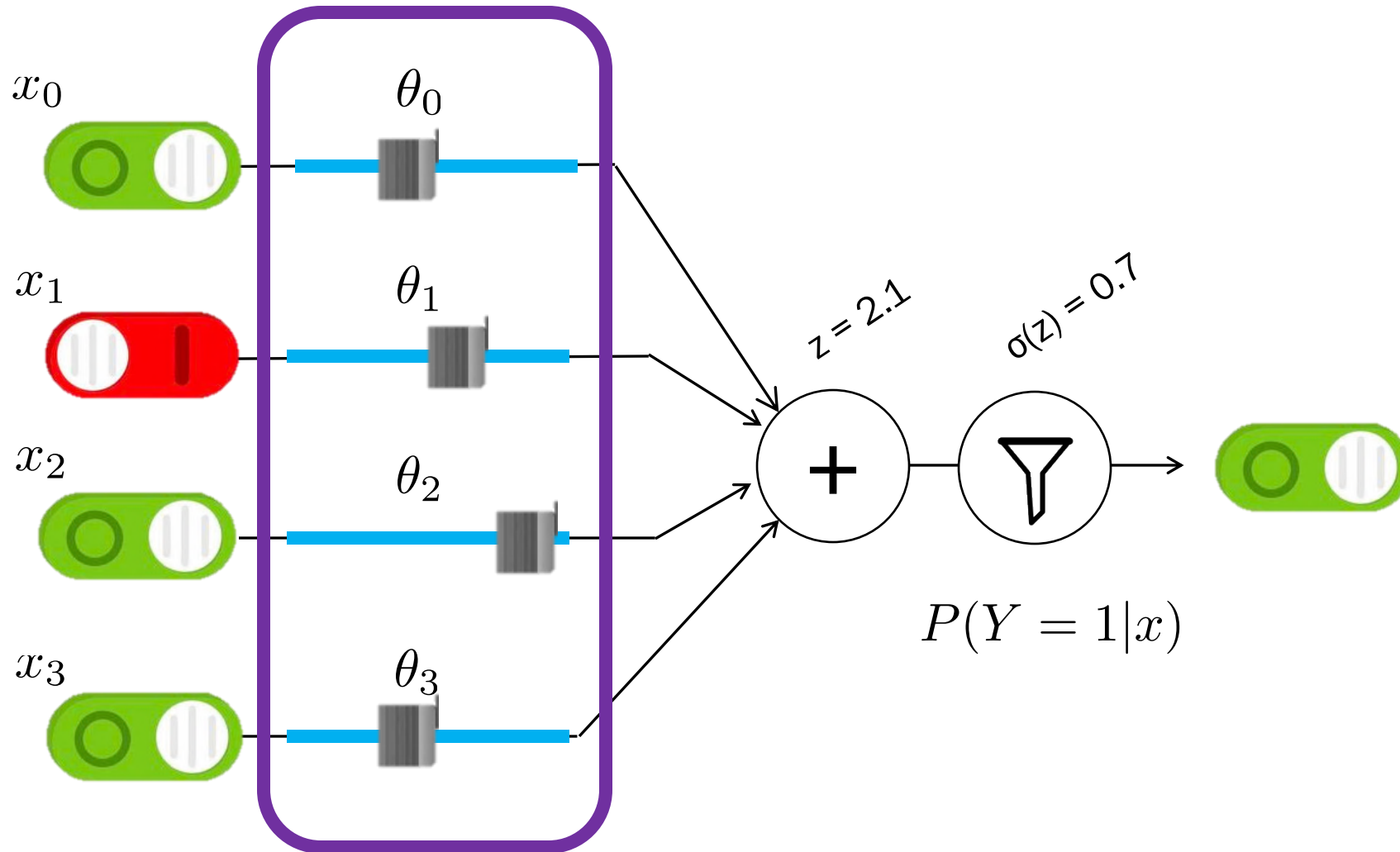
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Parameters Affect Prediction



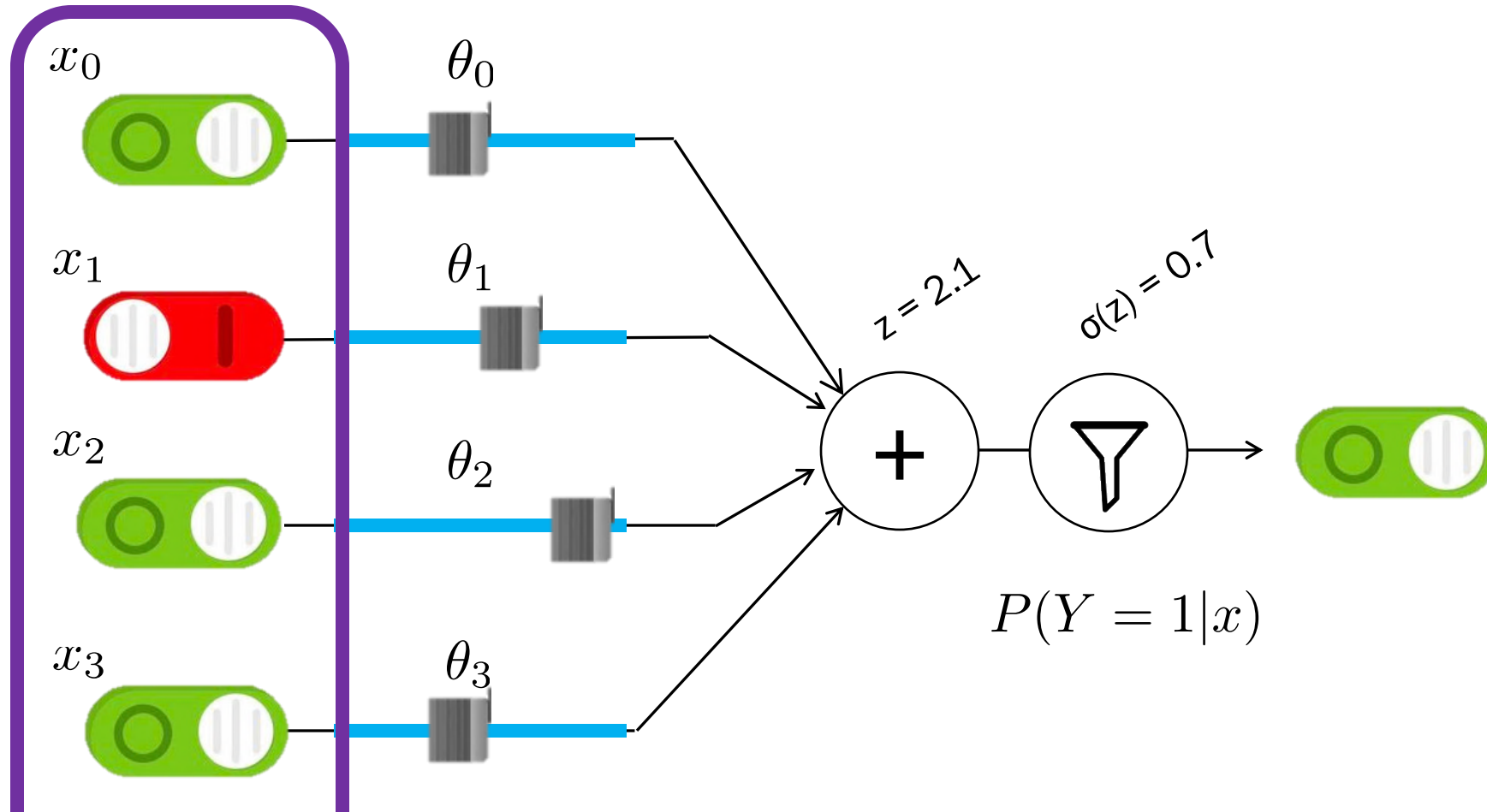
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Parameters Affect Prediction



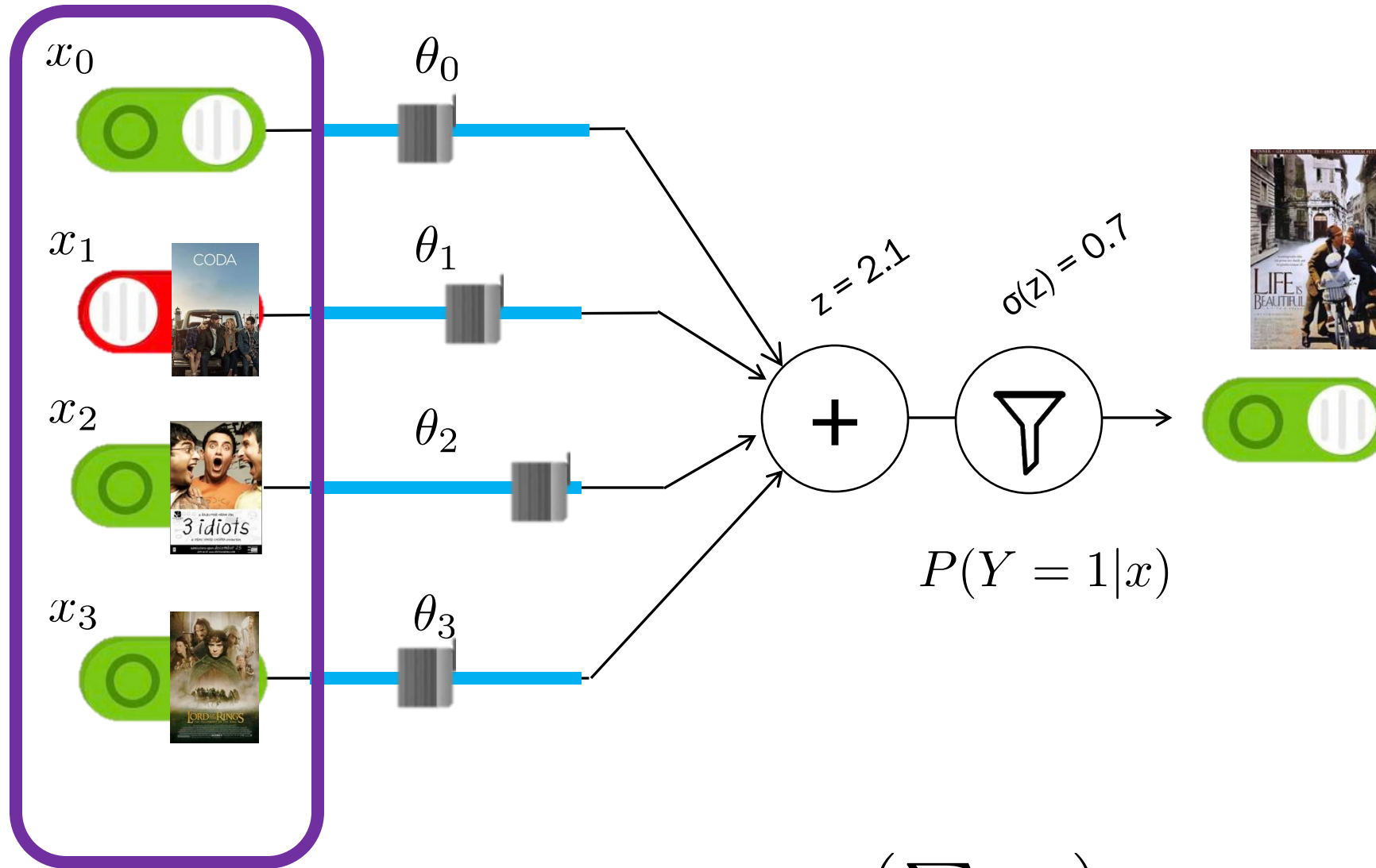
$$P(Y = 1 | \mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Different Predictions for Different Inputs



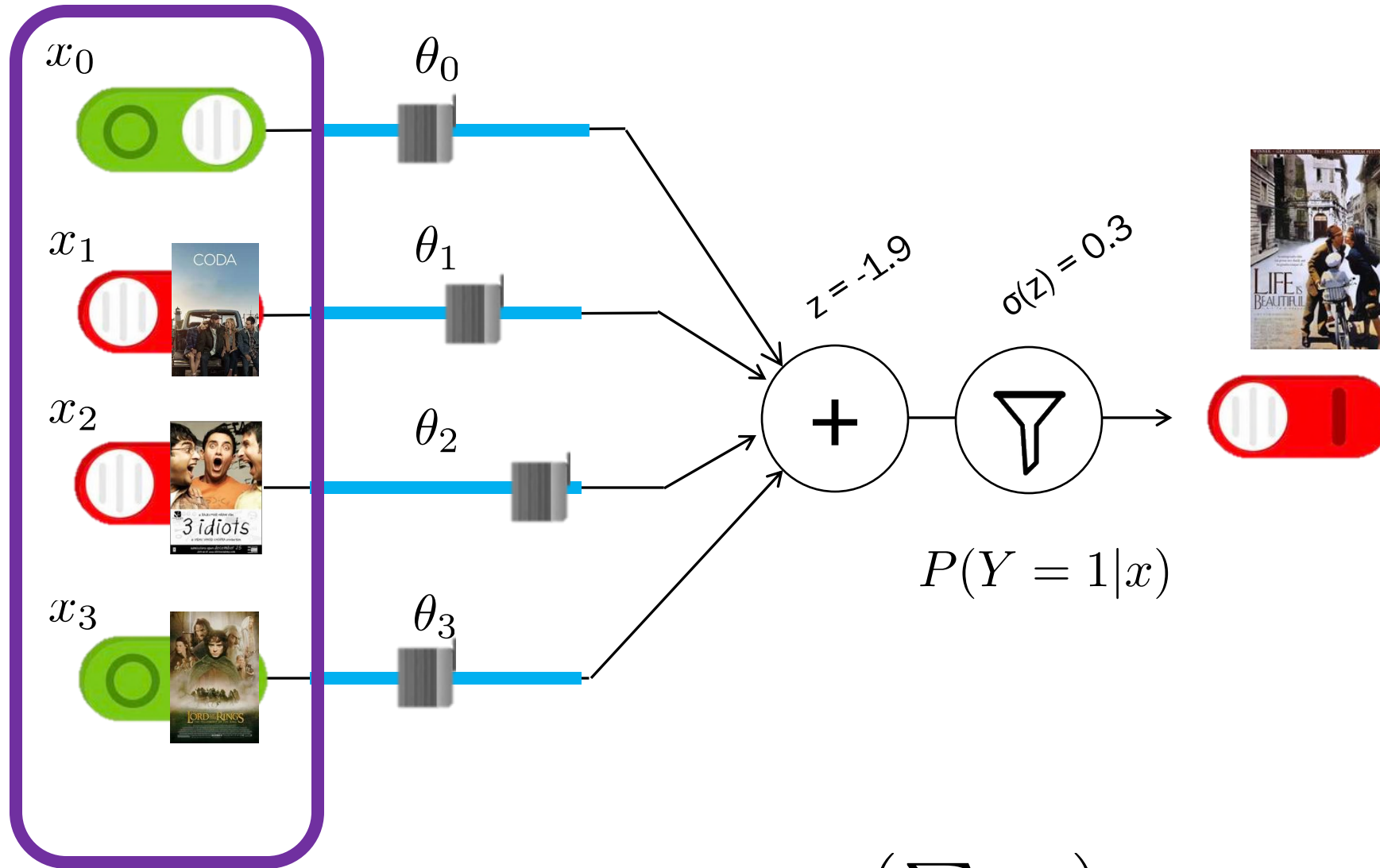
$$P(Y = 1 | \mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Different Predictions for Different Inputs



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

Different Predictions for Different Inputs



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

Handling the Intercept

Model *conditional* likelihood $P(Y = 1 | \mathbf{X} = \mathbf{x})$
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 $P(Y = 0 | \mathbf{X} = \mathbf{x}) + P(Y = 1 | \mathbf{X} = \mathbf{x}) = 1$:

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

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

Recall:
Sigmoid function

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

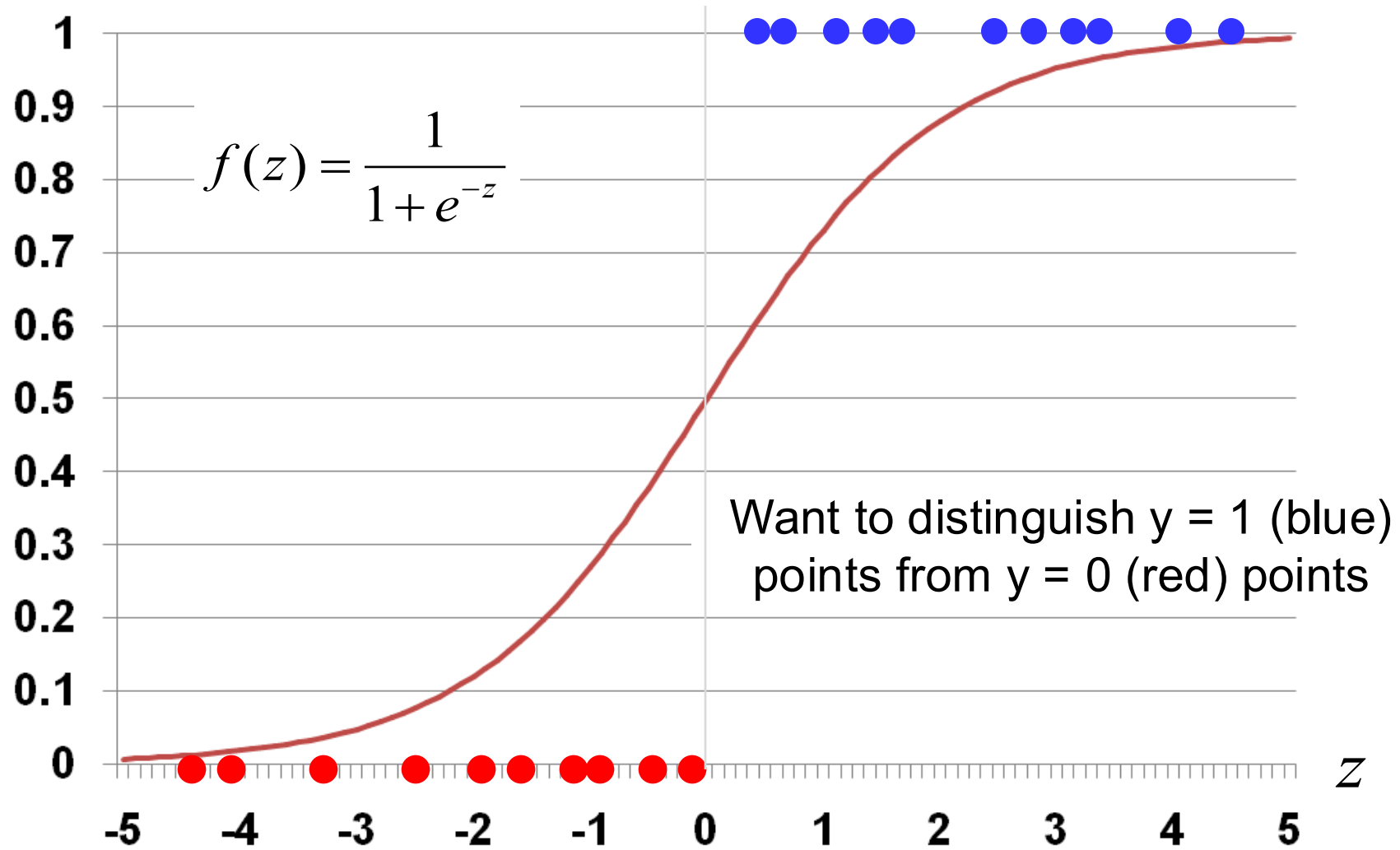
Big Assumption



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

Decision Tree Classifier



Logistic Regression



Awesome classifier, terrible name



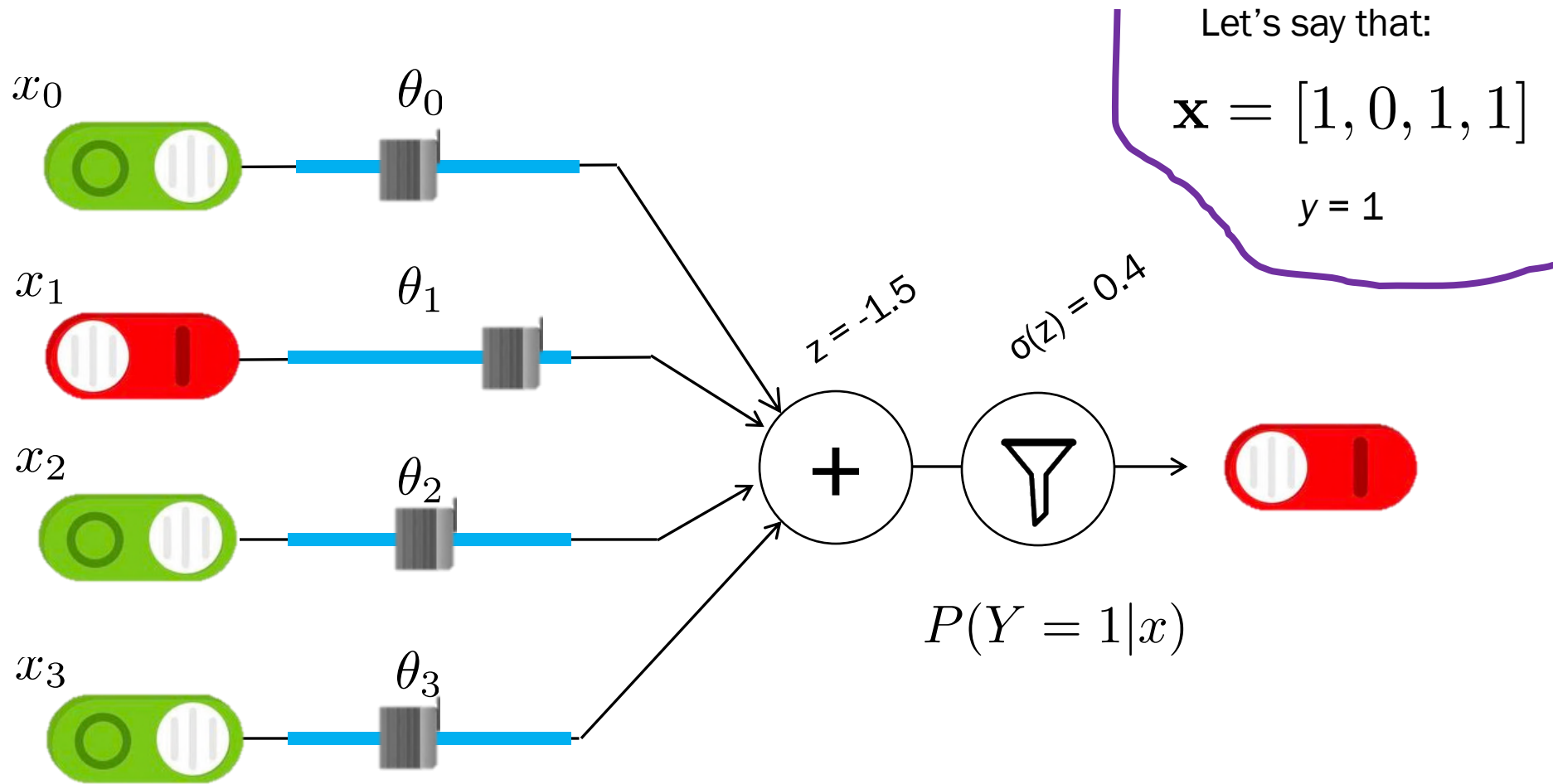
If Chris could rename it he would call it: Sigmoidal Classification

What makes for a “smart”
logistic regression algorithm?



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

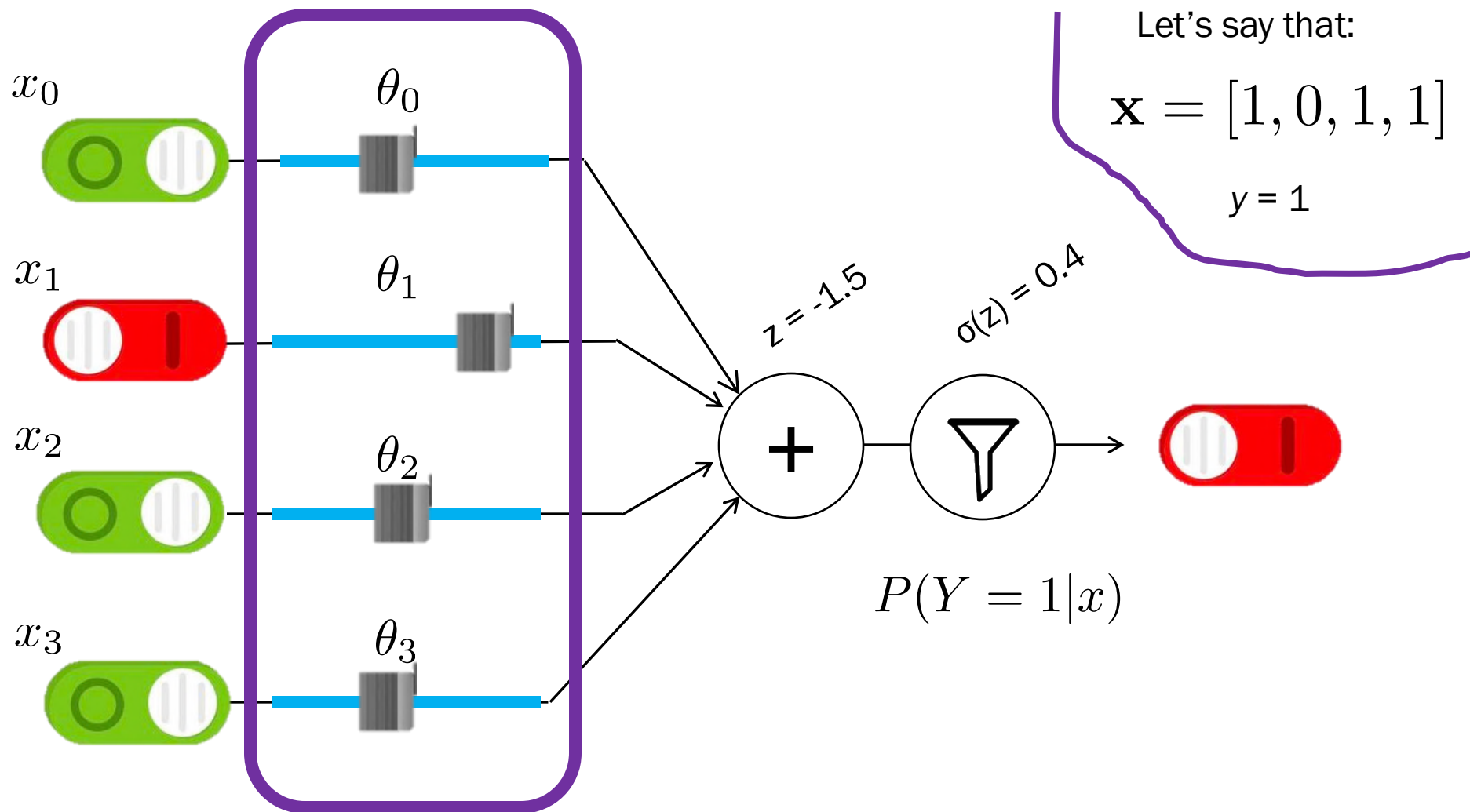
How Do We Learn Parameters?



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

Data looks unlikely

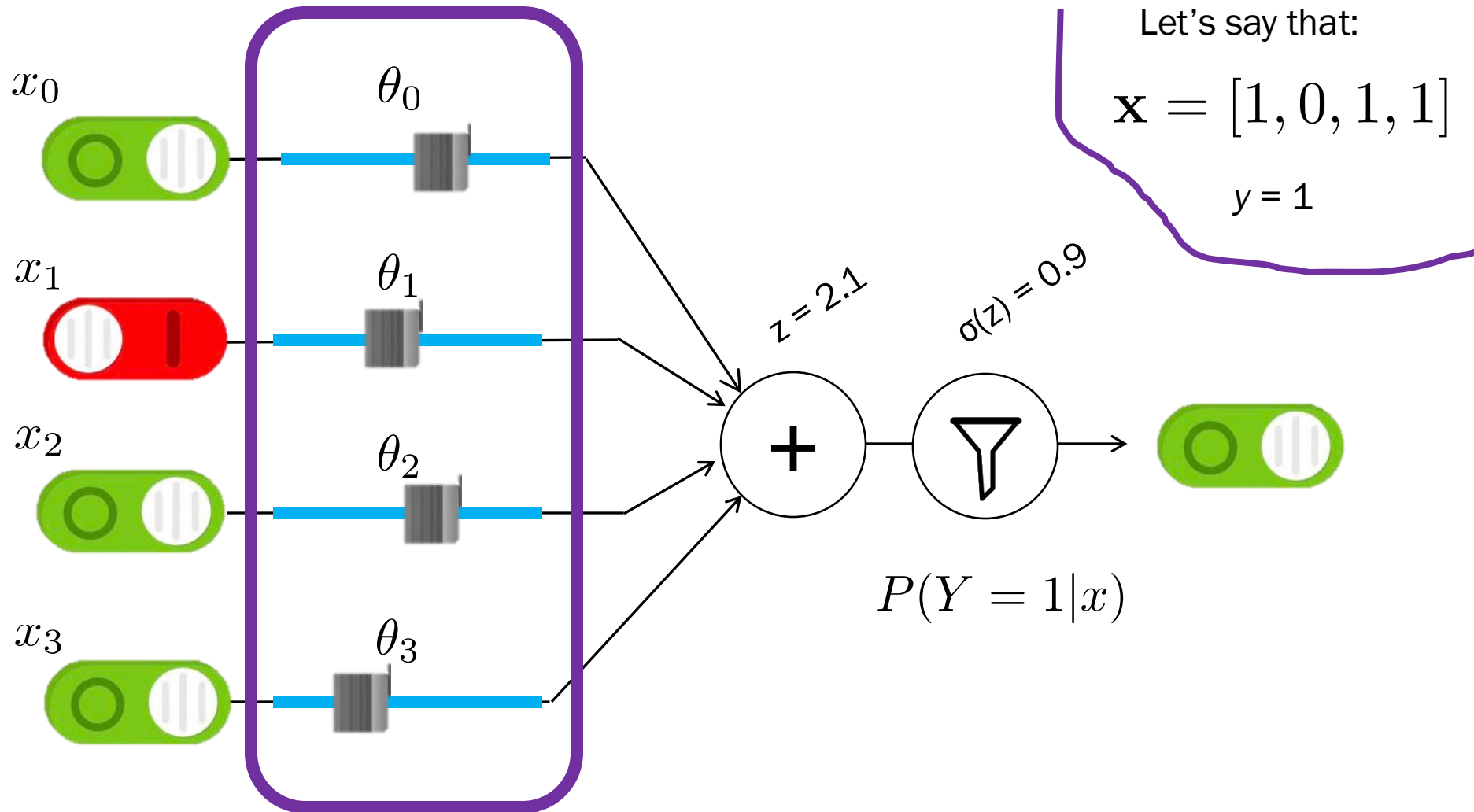
How Do We Learn Parameters?



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

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

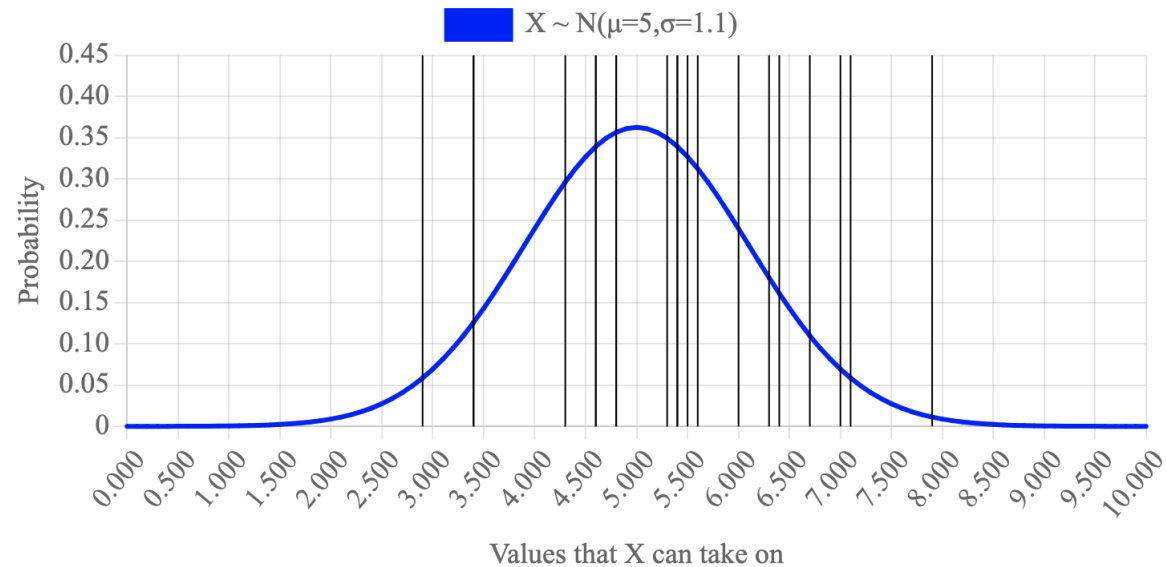
Likelihood of the data given your params

Likelihood: 5.204152095194613e-16

Log Likelihood: -314.1

Best Seen: -311.2

Your Gaussian



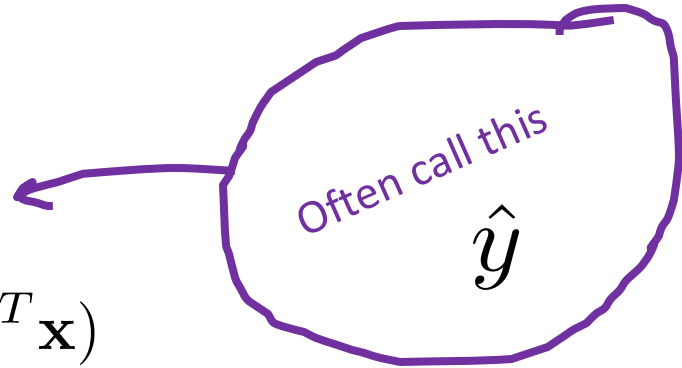
Pedagogy: show you the big picture,
then we can derive it!

Math for Logistic Regression

- 1 Make logistic regression assumption

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

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



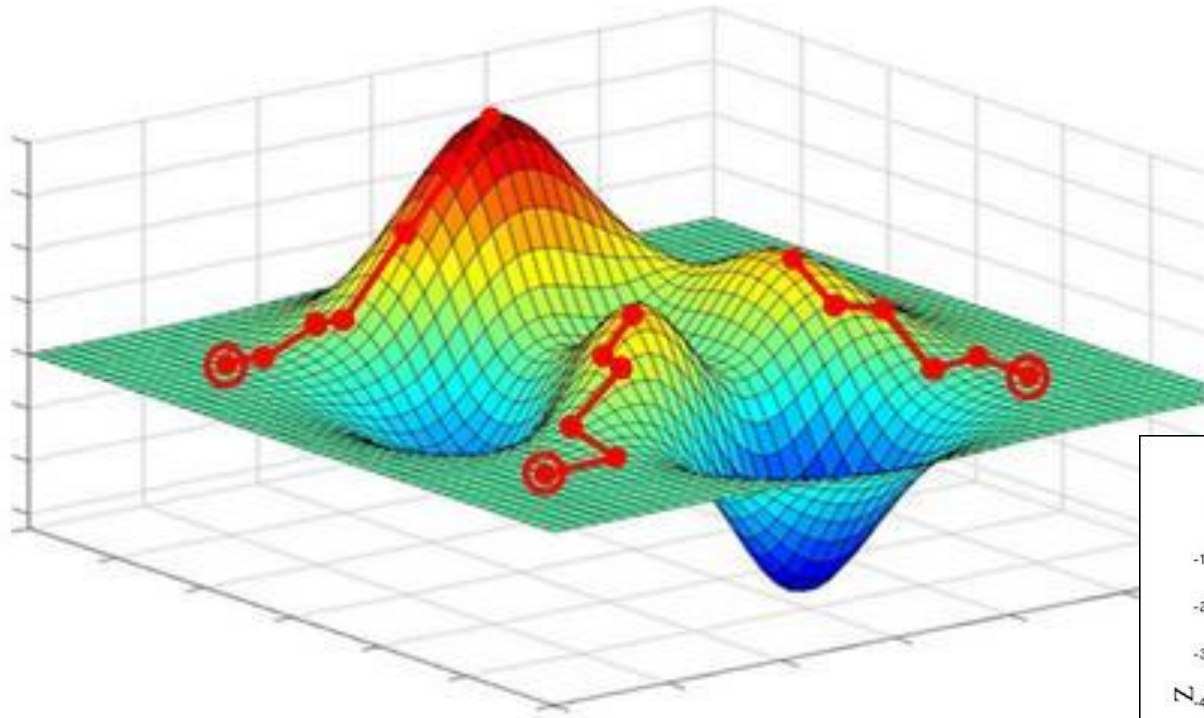
- 2 Calculate the log likelihood for all data

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

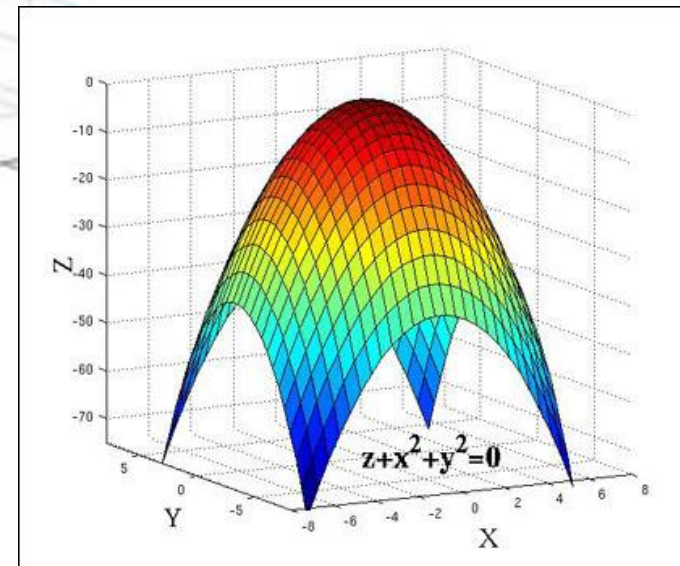
- 3 Get derivative of log likelihood with respect to thetas

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

Gradient Ascent



Logistic regression LL
function is convex



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

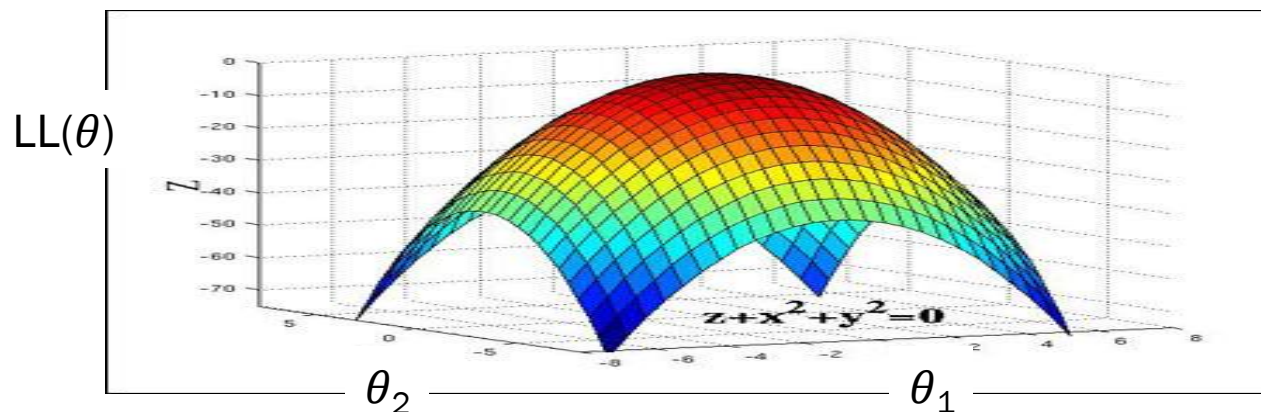
Gradient Ascent Step

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

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

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

Do this
for all
thetas!



What does this look like in code?

$$\begin{aligned}\theta_j^{\text{new}} &= \theta_j^{\text{old}} + \eta \cdot \frac{\partial LL(\theta^{\text{old}})}{\partial \theta_j^{\text{old}}} \\ &= \theta_j^{\text{old}} + \eta \cdot \sum_{i=0}^n \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}\end{aligned}$$

Real Code!!!

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 (\mathbf{x}, y) :

For each parameter j :

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

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

Step by Step

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:

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

Calculate all $\text{gradient}[j]$'s based on data

$\theta_j += \eta * \text{gradient}[j]$ 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)

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

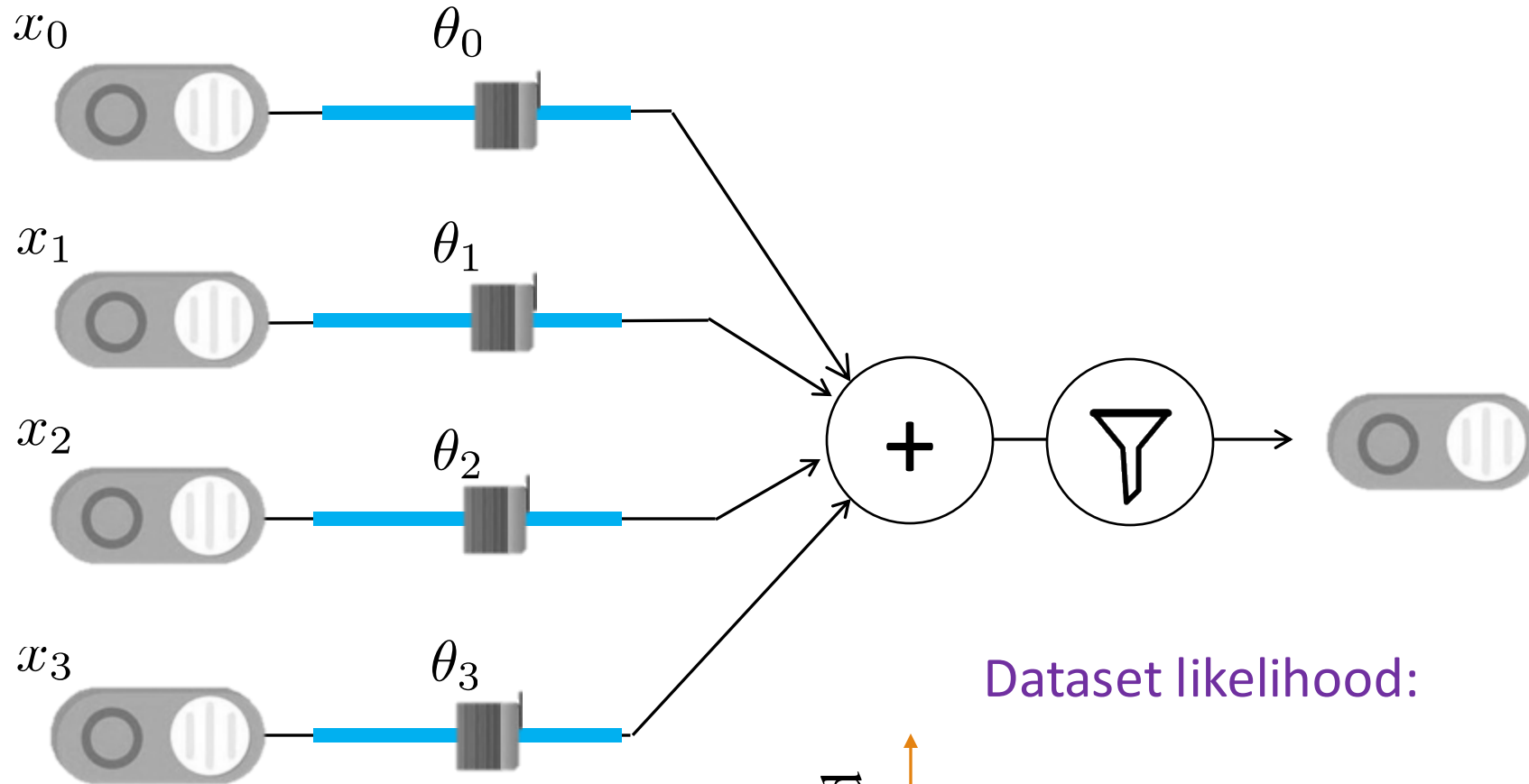
For each training example (\mathbf{x}, y) :

For each parameter j :

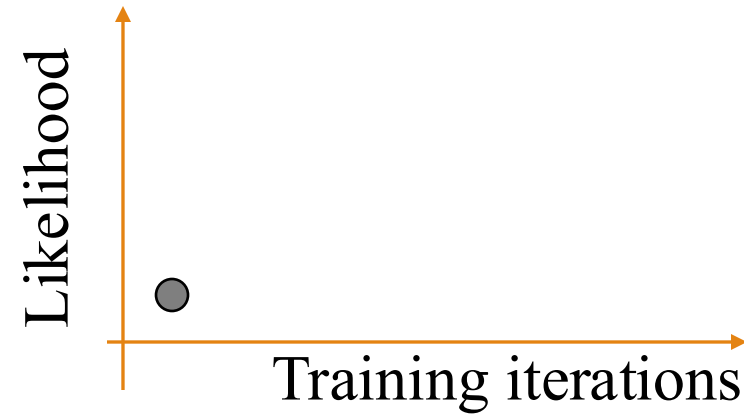
$$\text{gradient}[j] += x_j \left(y - \frac{1}{1 + e^{-\theta^T \mathbf{x}}} \right)$$

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

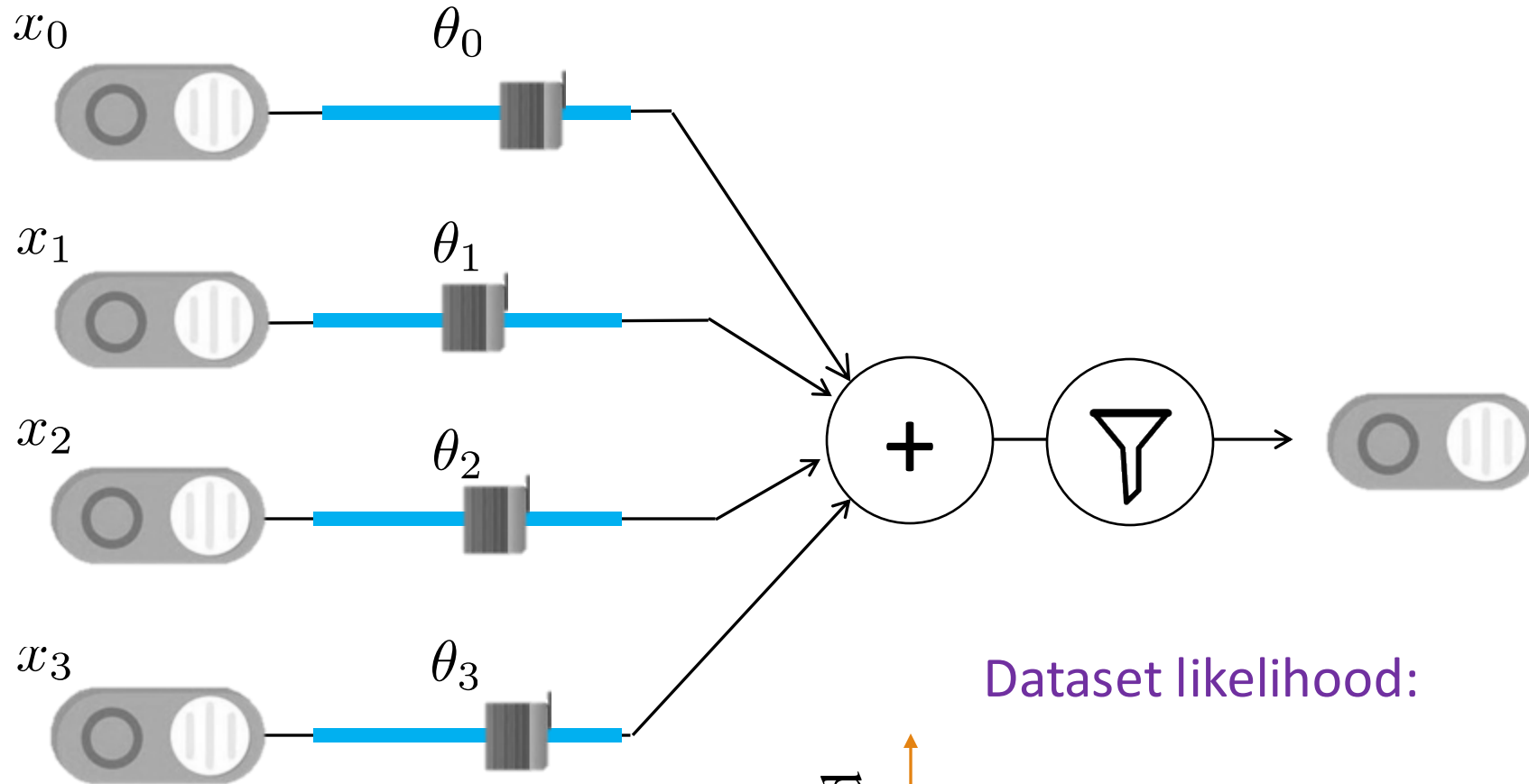
Training



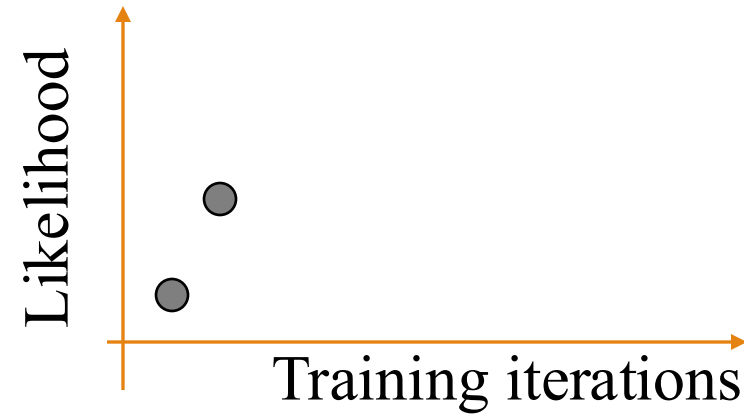
Dataset likelihood:



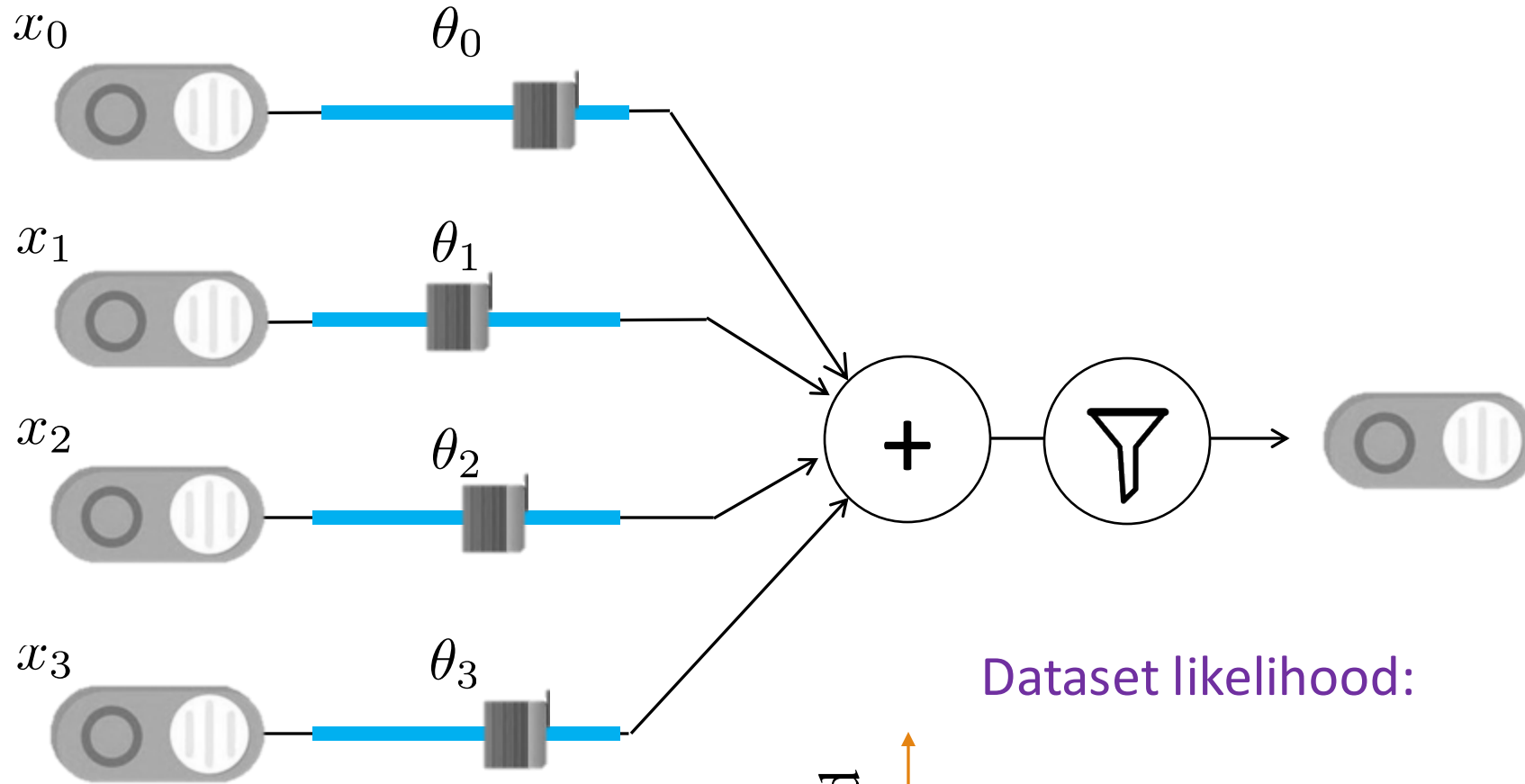
Training



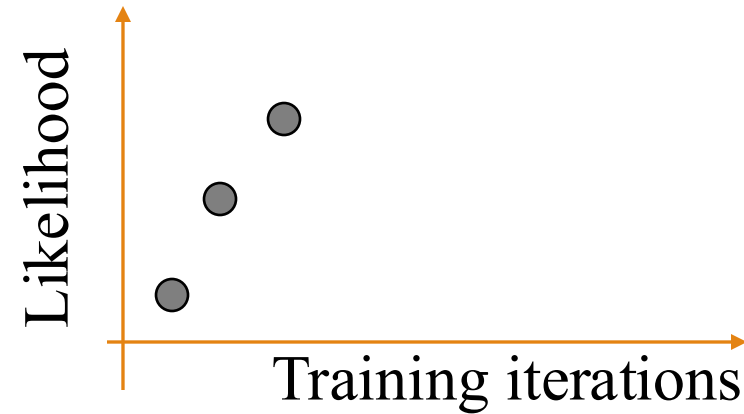
Dataset likelihood:



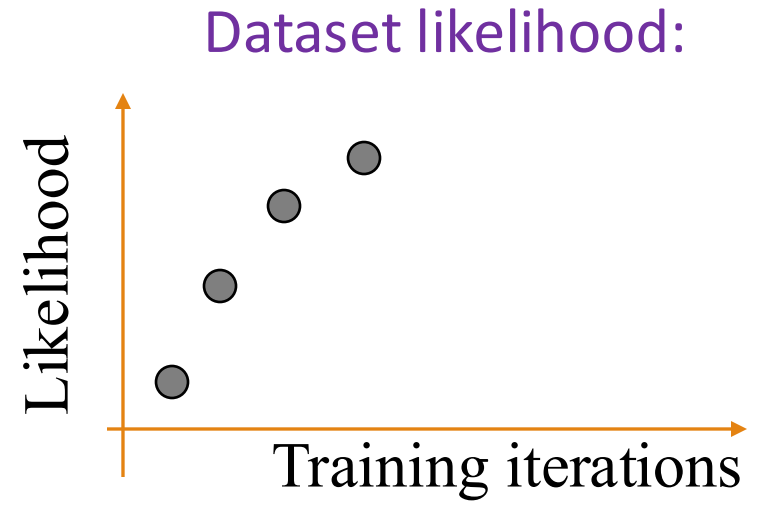
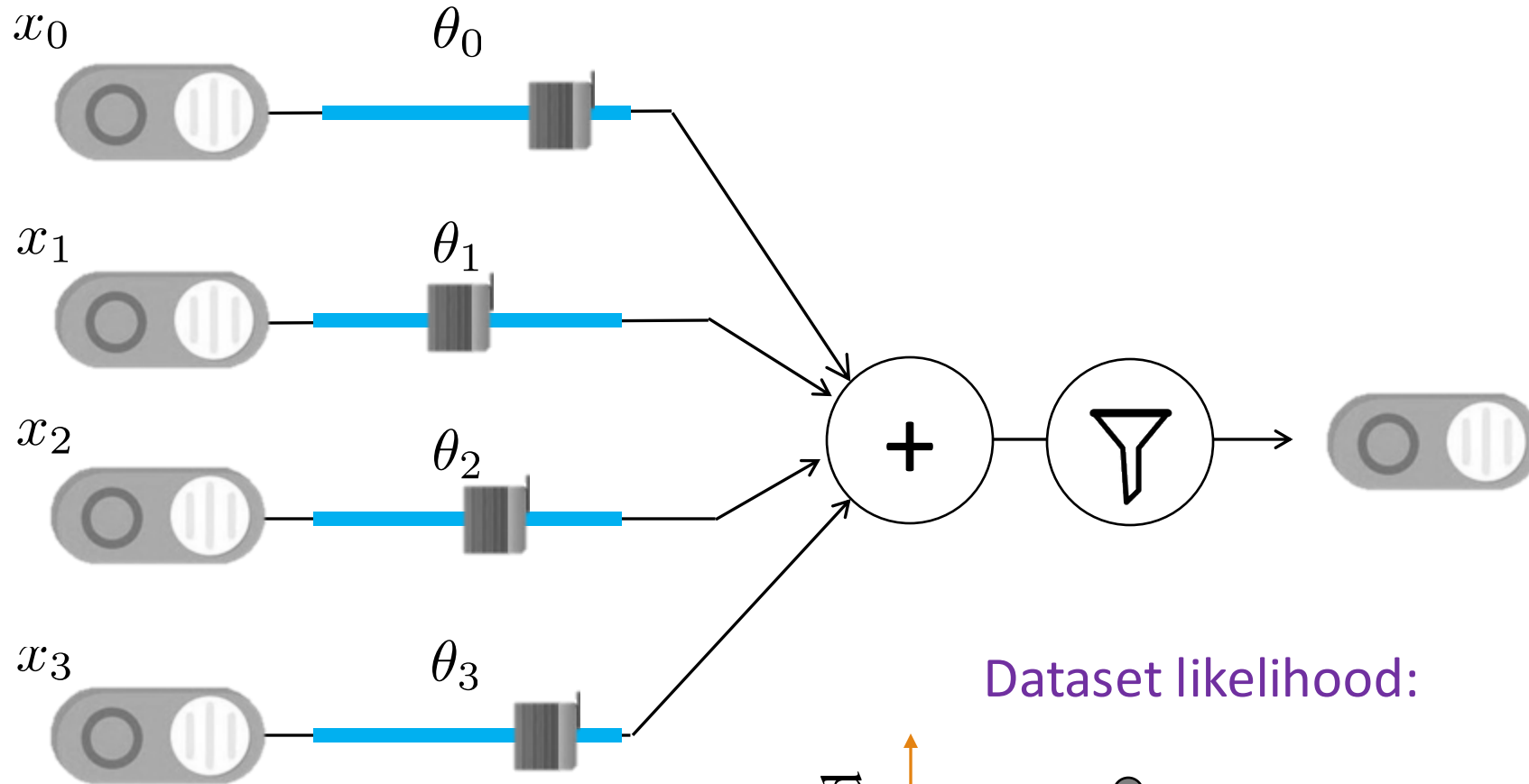
Training



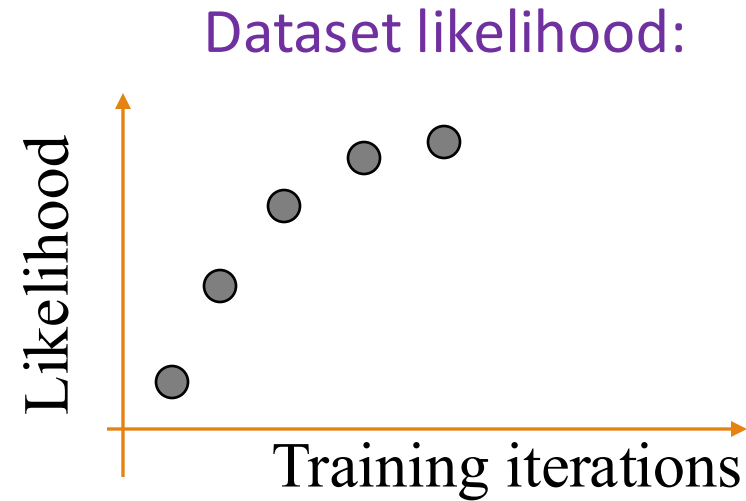
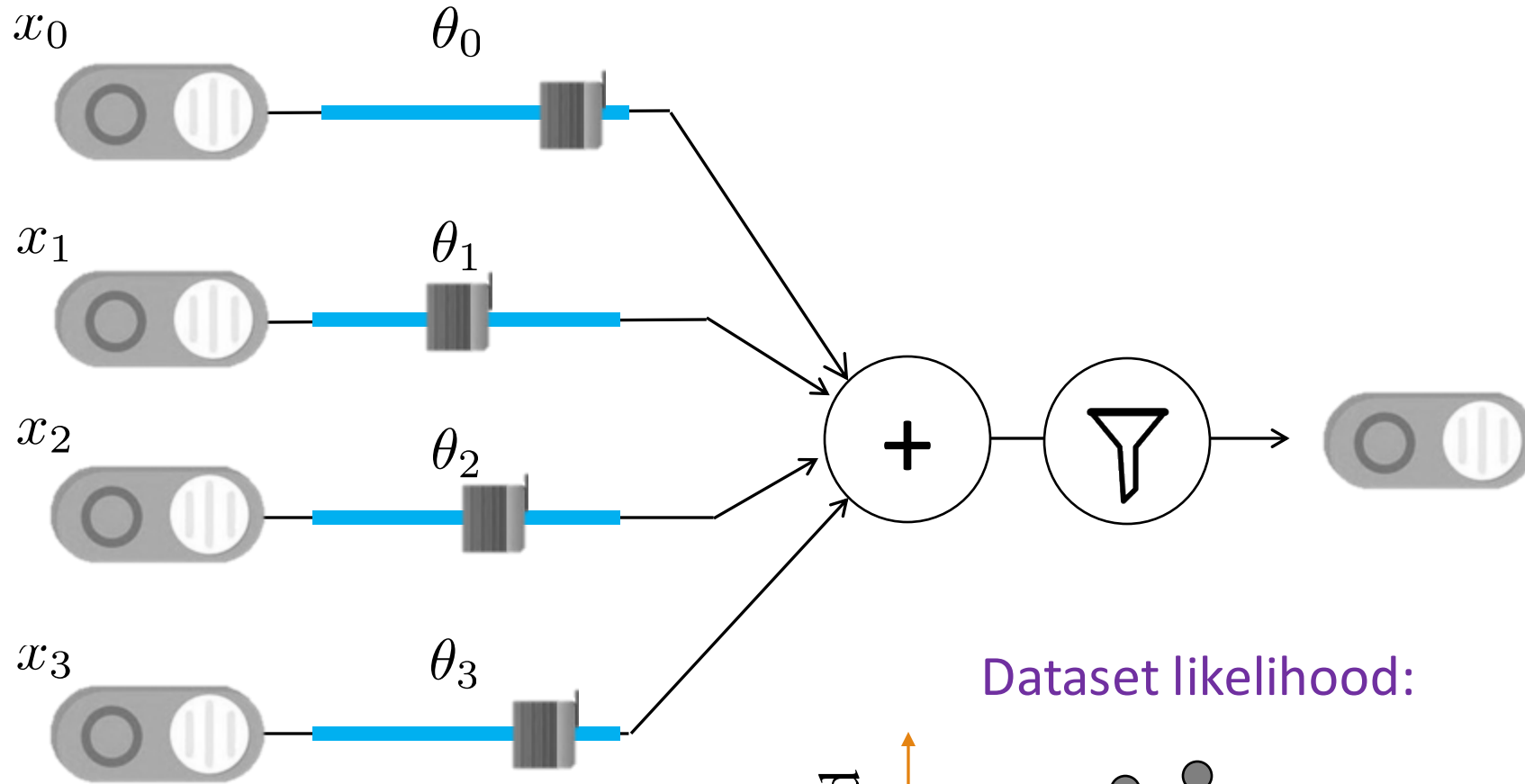
Dataset likelihood:



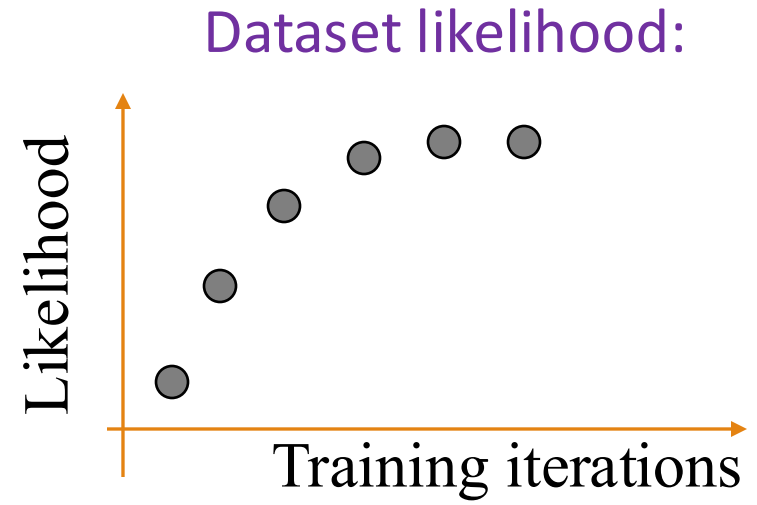
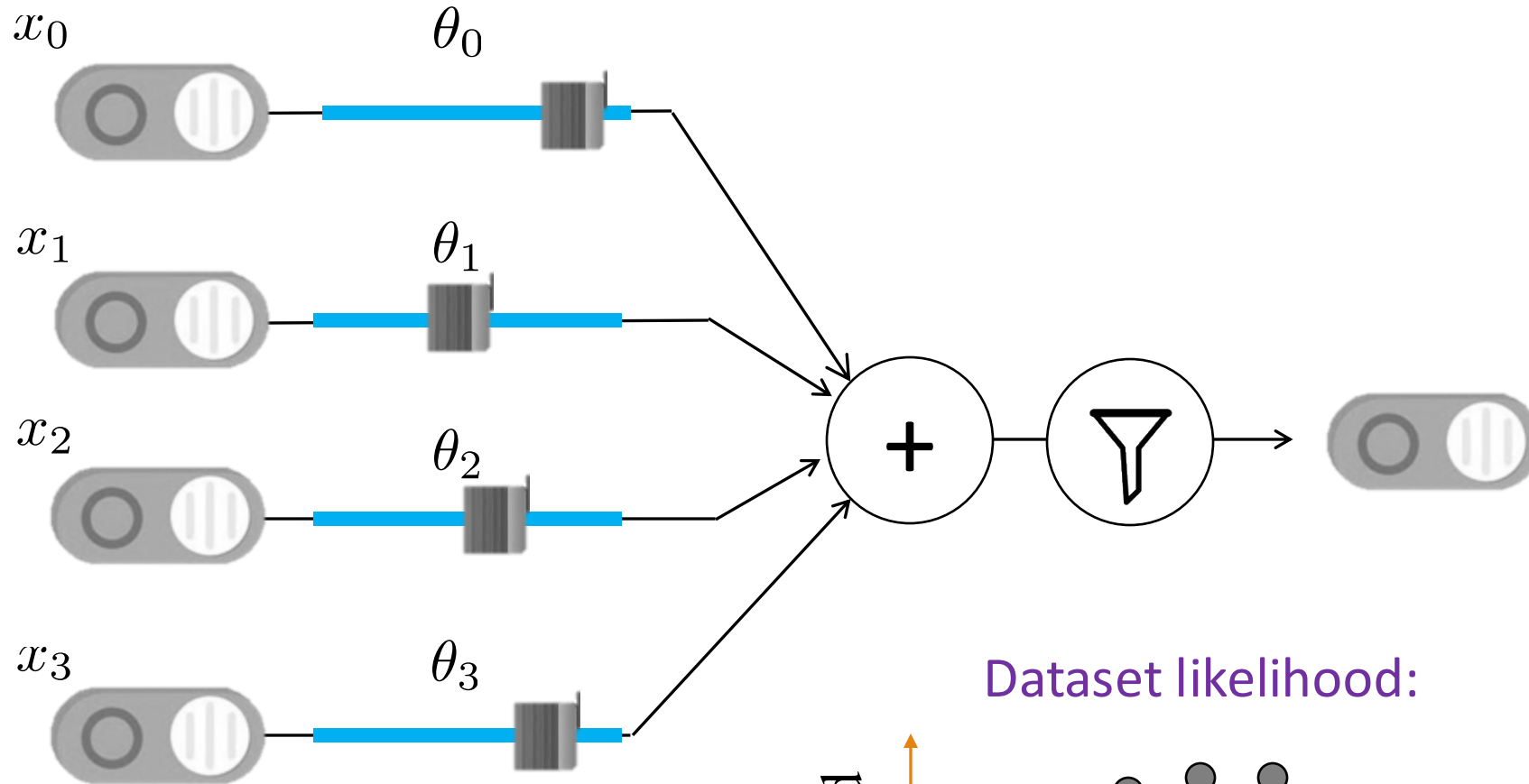
Training



Training



Training



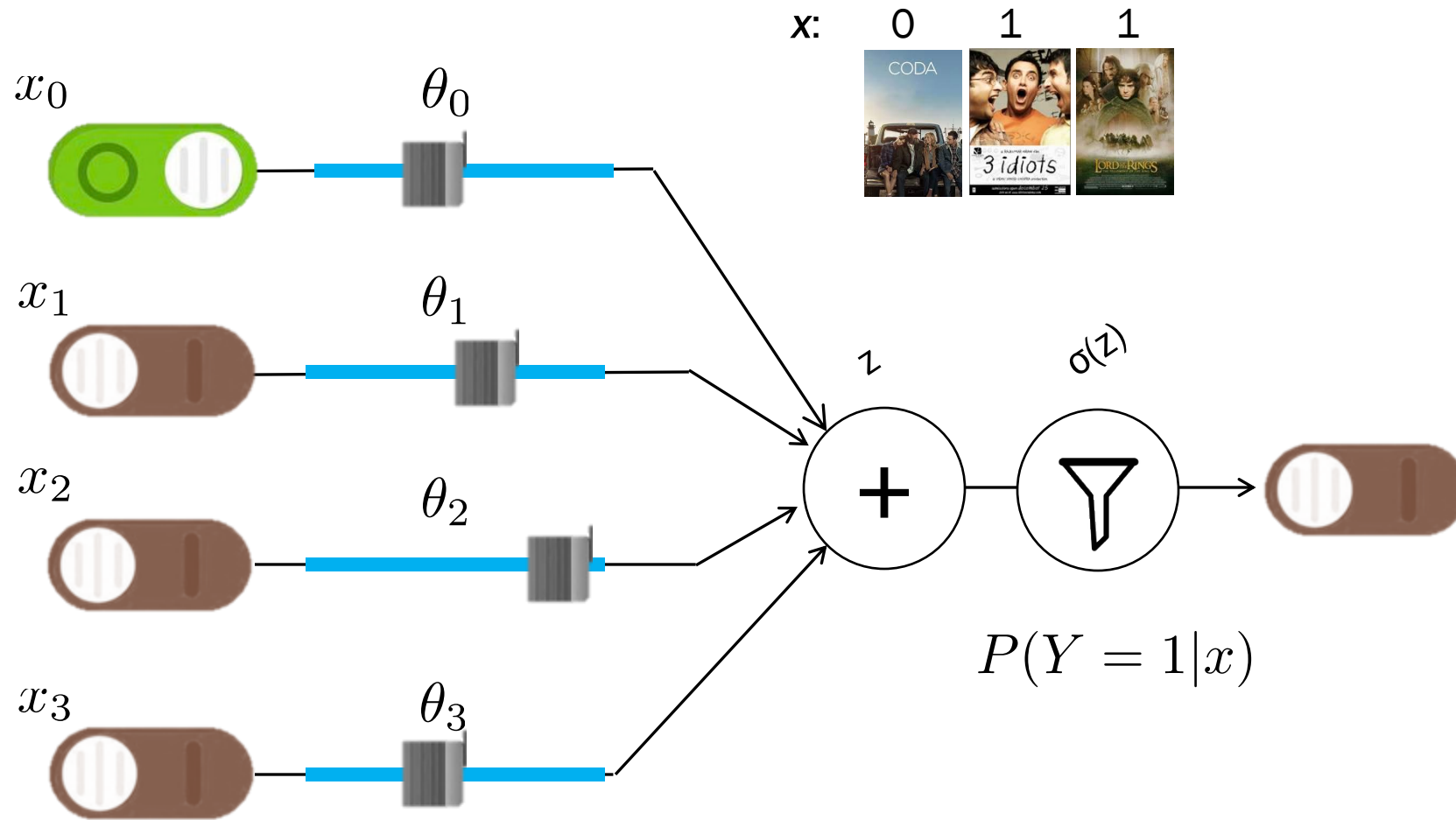


Don't forget:

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

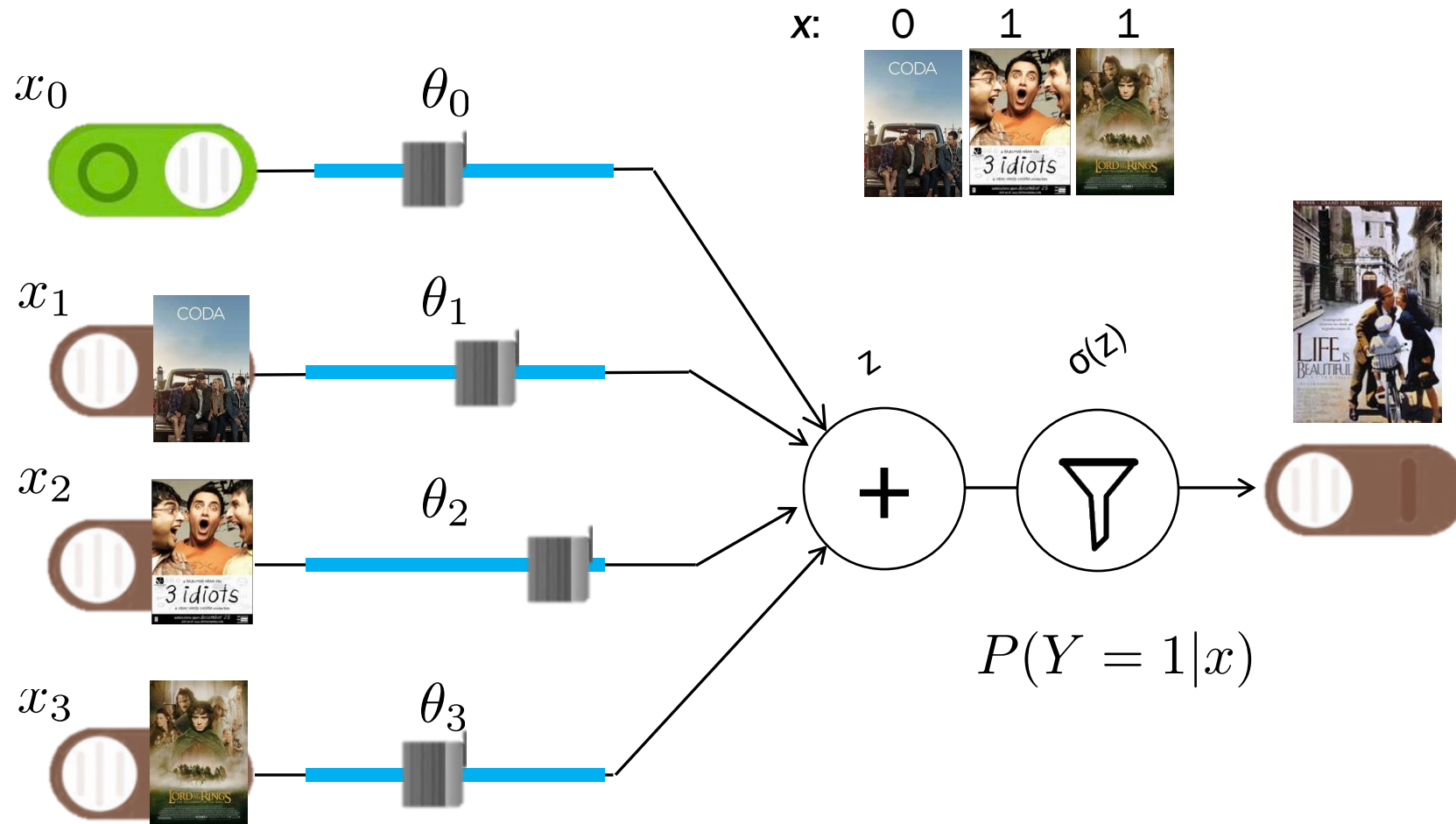
Allows for θ_0 to be an
intercept.

Prediction



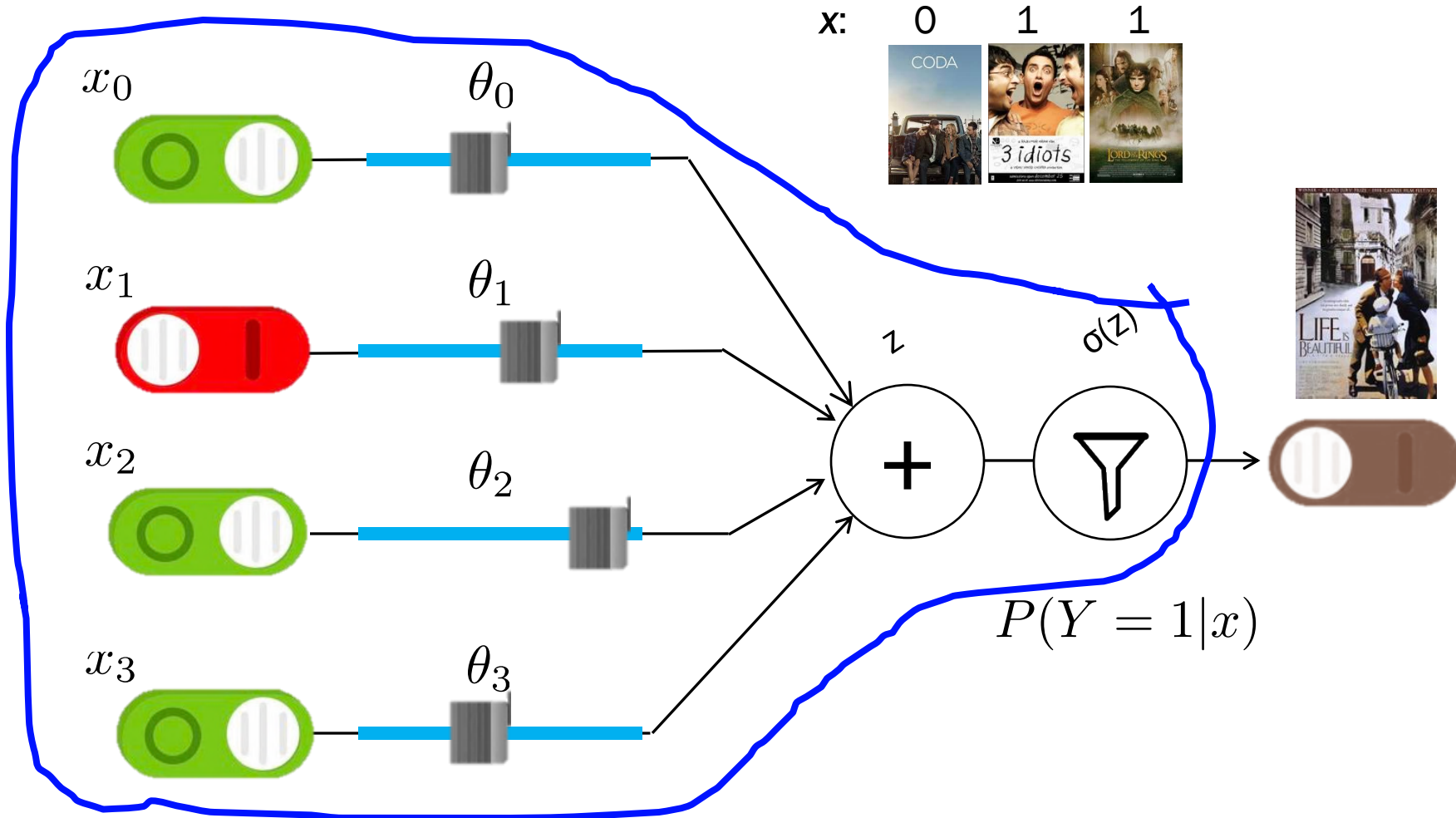
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Prediction



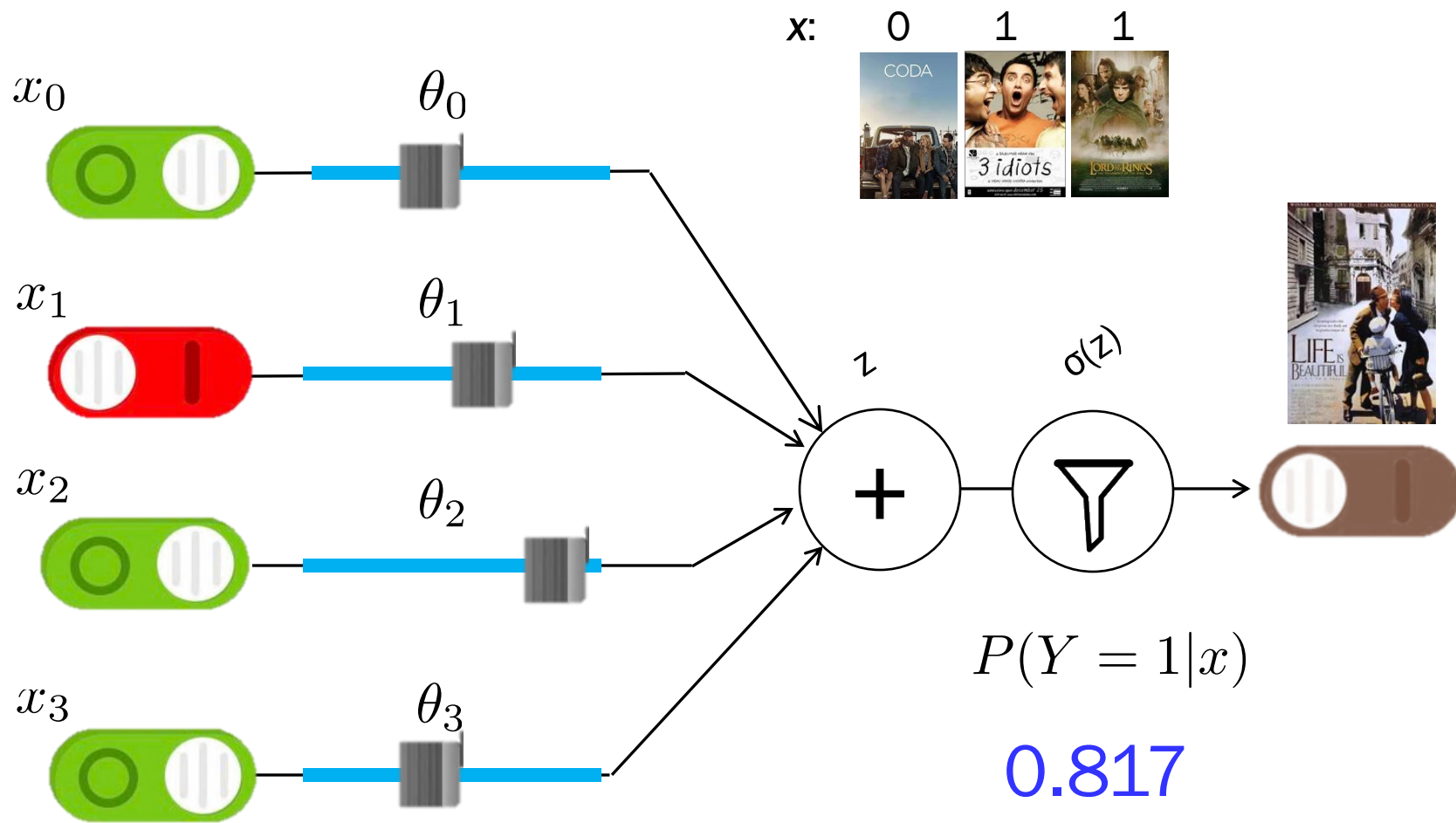
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Prediction



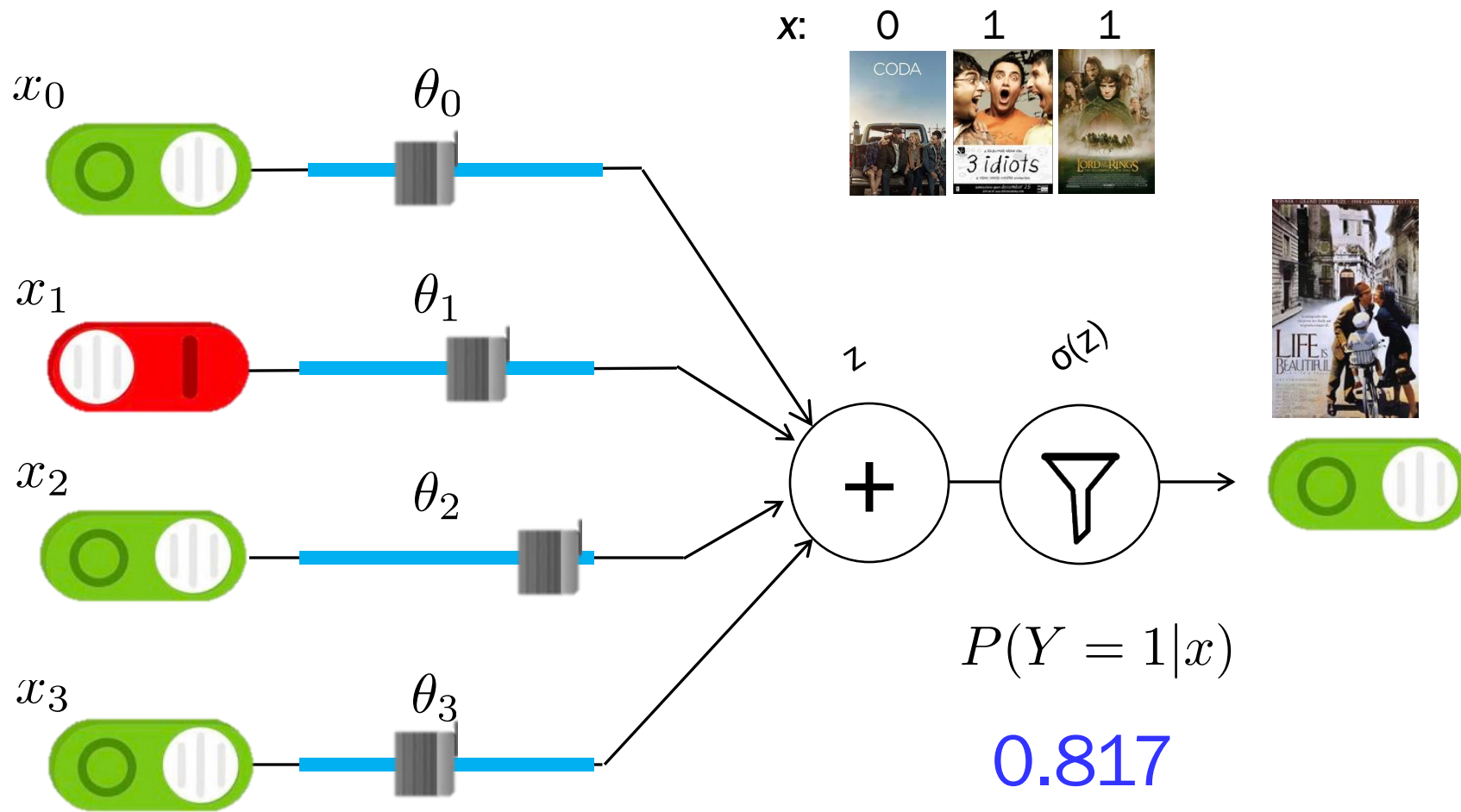
$$P(Y = 1 | X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Prediction



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

Prediction



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

Live Demo!!

iris versicolor



petal

sepal

iris virginica



petal

sepal

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

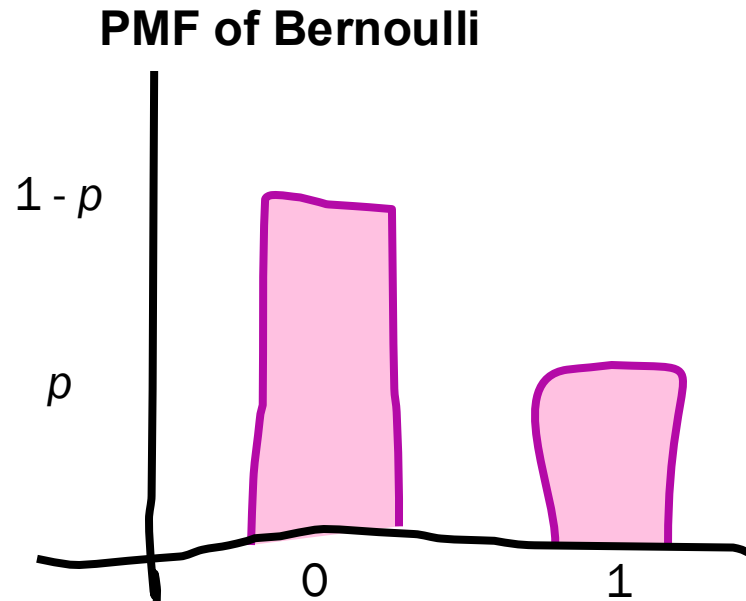
$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^n \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

How did we get that LL function?

Recall: PMF of Bernoulli

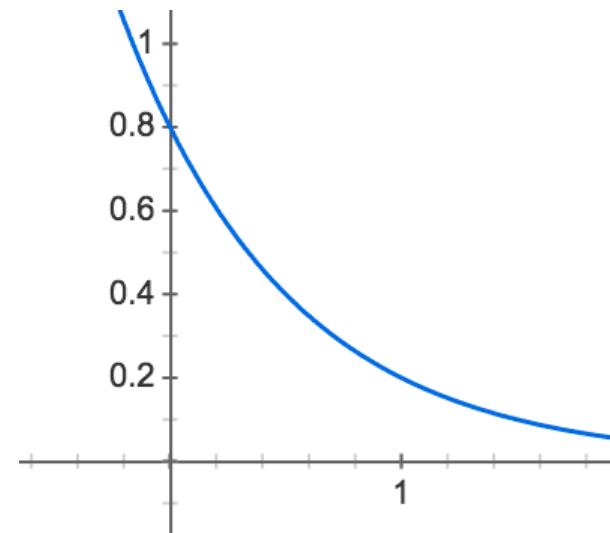
$$Y \sim \text{Bern}(p)$$

Probability mass function: $P(Y = y)$



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

PMF of Bernoulli ($p = 0.2$)



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

Recall:

$Y \sim \text{Bern}(p)$

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

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

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

Implies

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

For IID data

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

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

Take the log

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

Ask Chris:
Why not

$$P(Y = y^{(i)}, X = x^{(i)})$$

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

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

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

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

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

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

Derivative of sum...

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

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

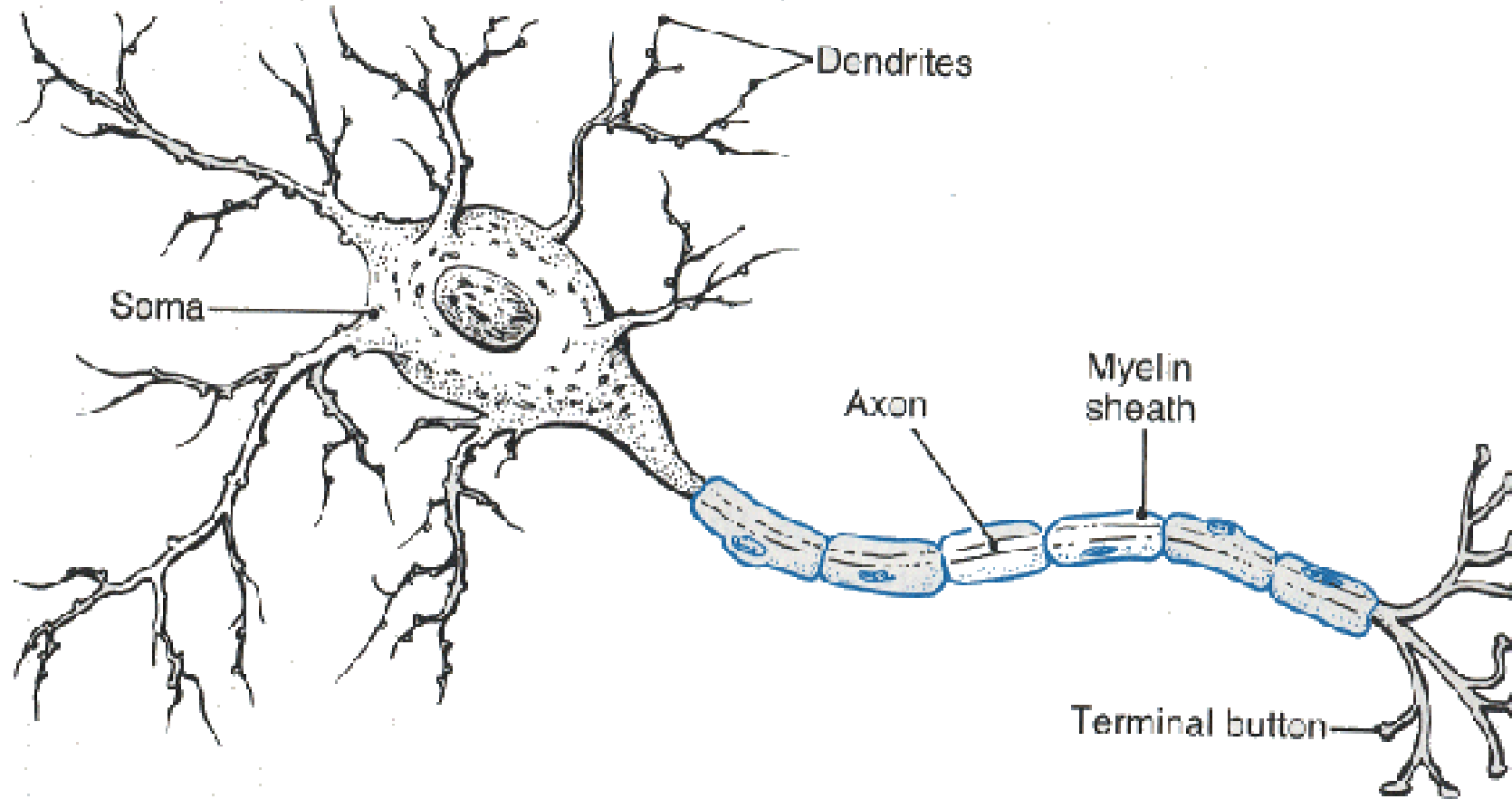
$$LL(\theta) = y \log \sigma(\theta^T \mathbf{x}) + (1 - y) \log[1 - \sigma(\theta^T \mathbf{x})]$$

$$\begin{aligned} \frac{\partial LL(\theta)}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j} y \log \sigma(\theta^T \mathbf{x}) + \frac{\partial}{\partial \theta_j} (1 - y) \log[1 - \sigma(\theta^T \mathbf{x})] \\ &= \left[\frac{y}{\sigma(\theta^T x)} - \frac{1 - y}{1 - \sigma(\theta^T x)} \right] \frac{\partial}{\partial \theta_j} \sigma(\theta^T x) \\ &= \left[\frac{y}{\sigma(\theta^T x)} - \frac{1 - y}{1 - \sigma(\theta^T x)} \right] \frac{\partial}{\partial \theta_j} \sigma(\theta^T x) \\ &= \left[\frac{y - \sigma(\theta^T x)}{\sigma(\theta^T x)[1 - \sigma(\theta^T x)]} \right] \sigma(\theta^T x)[1 - \sigma(\theta^T x)] x_j \\ &= [y - \sigma(\theta^T x)] x_j \end{aligned}$$

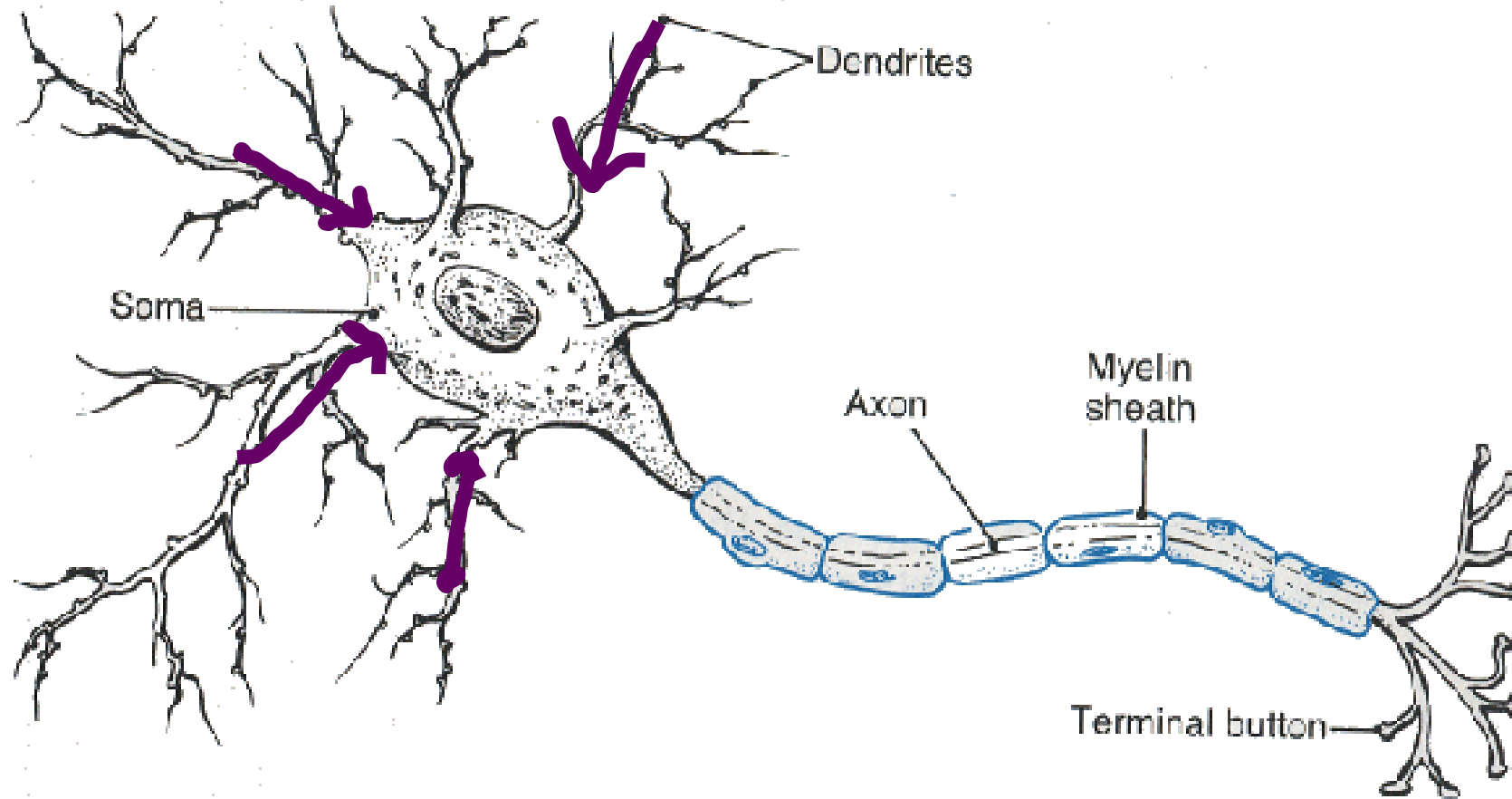
Phew!

Chapter 3: Philosophy (if time)

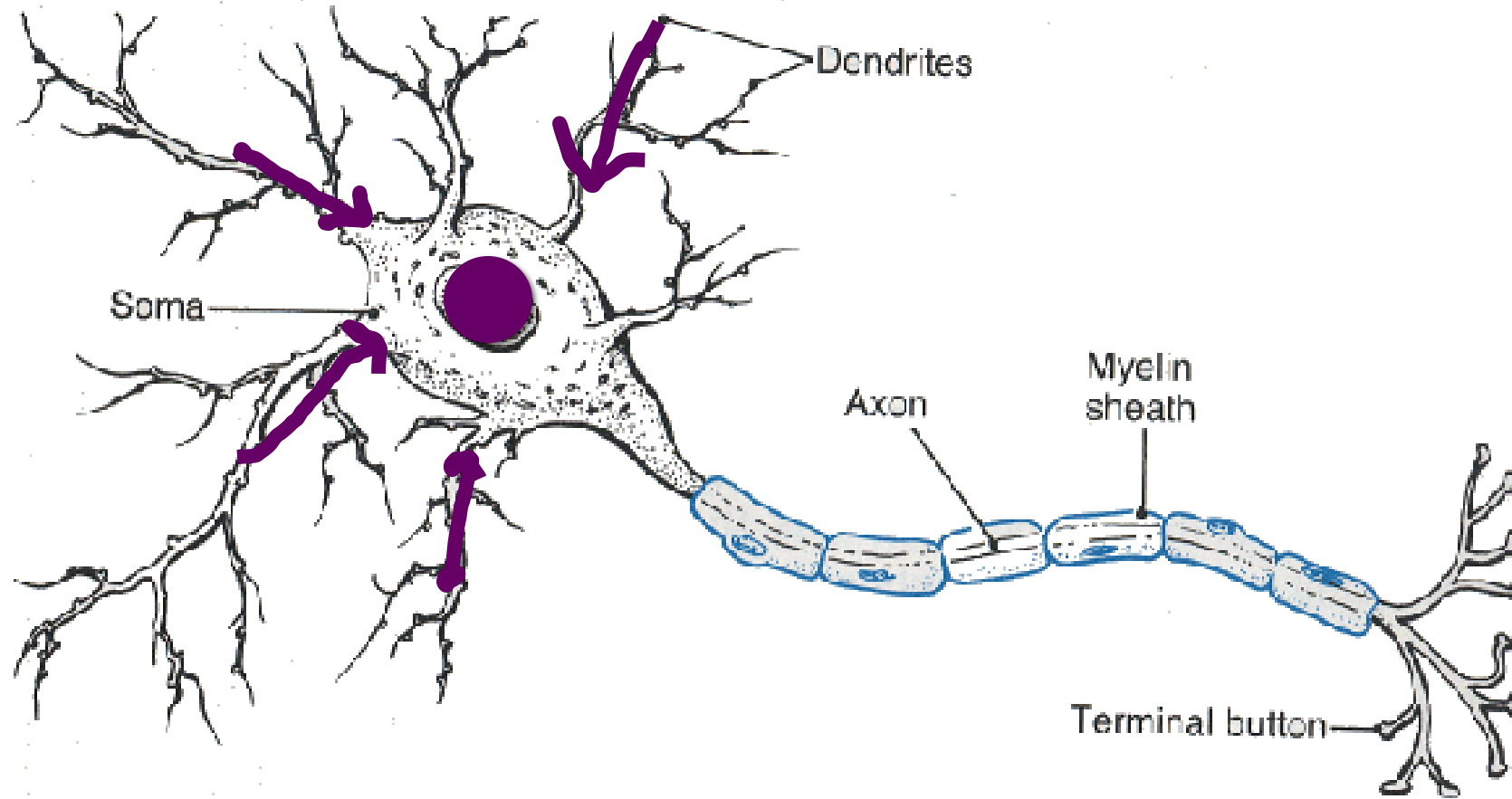
Neuron



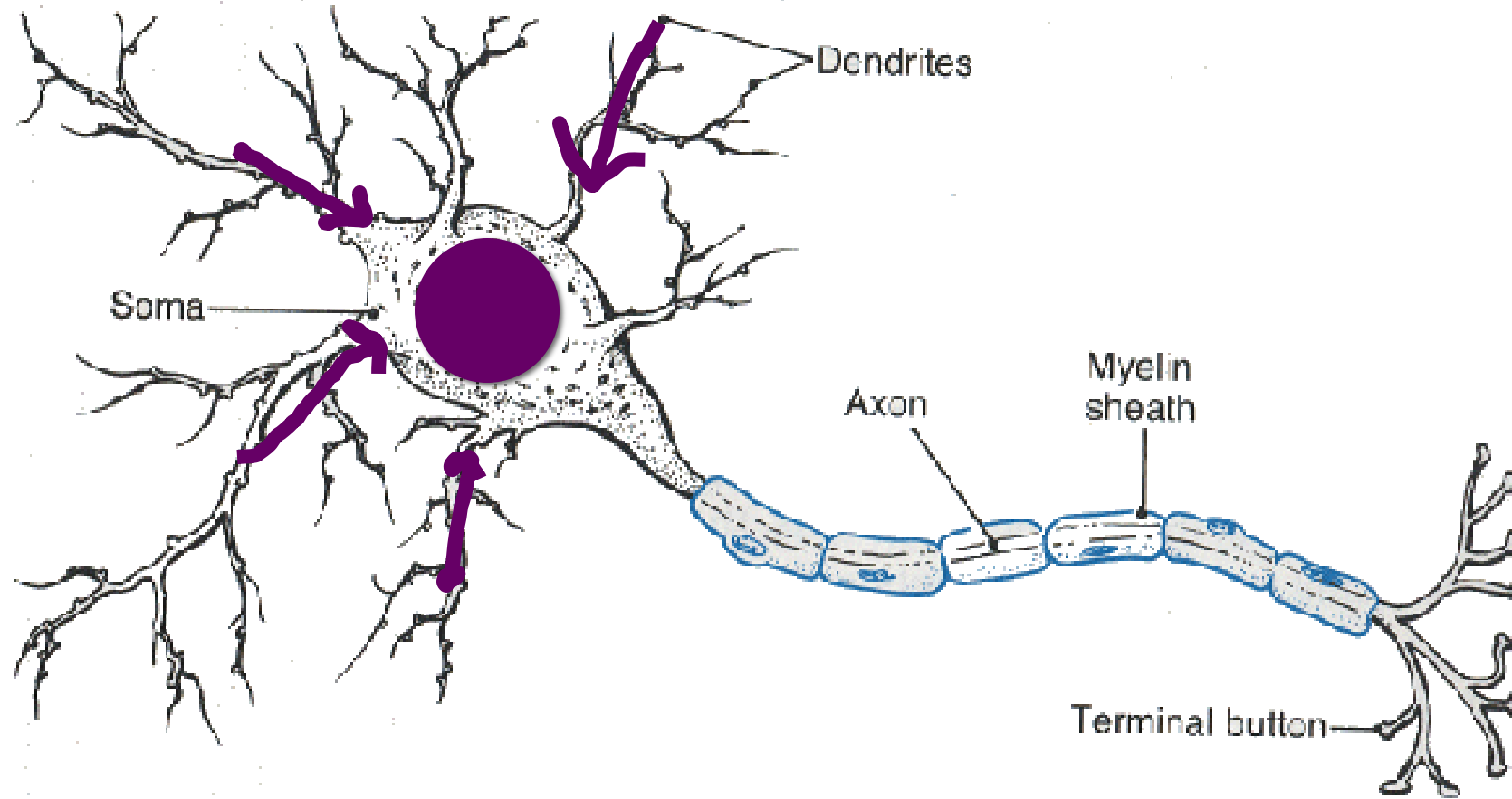
Neuron



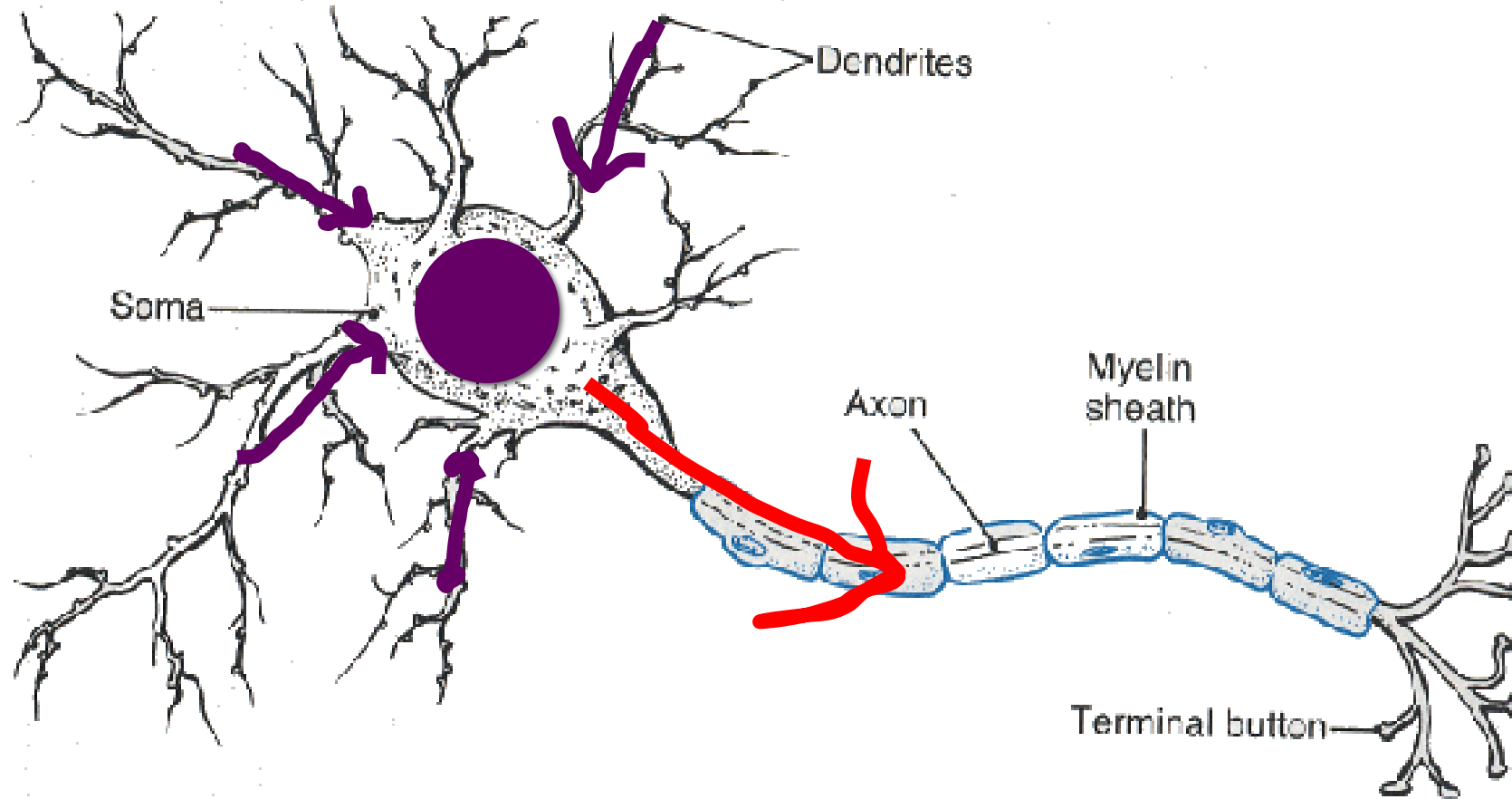
Neuron



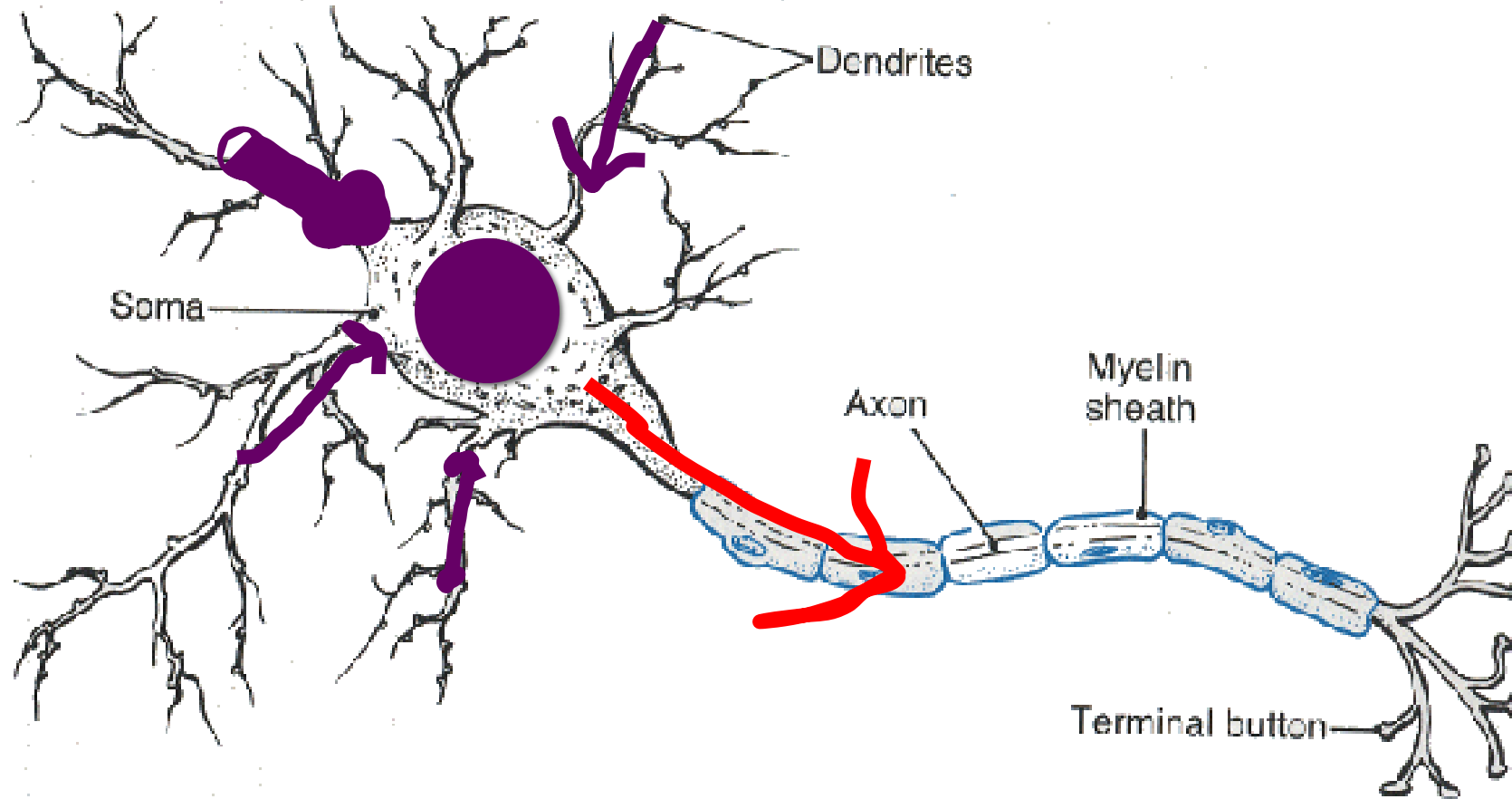
Neuron



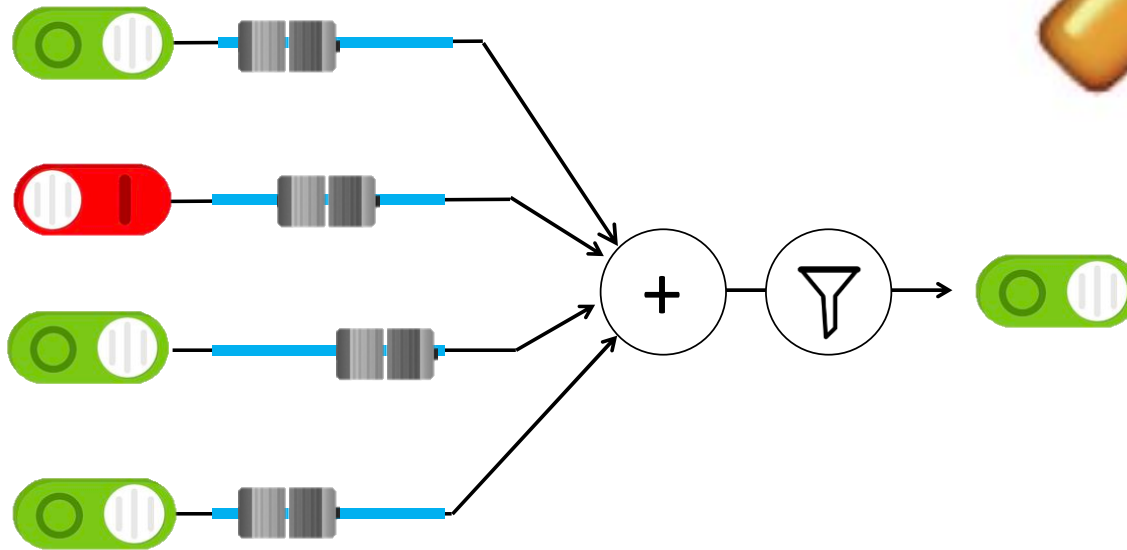
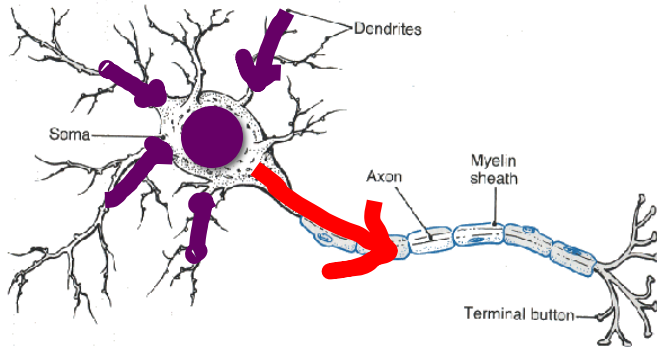
Neuron



Some inputs are more important

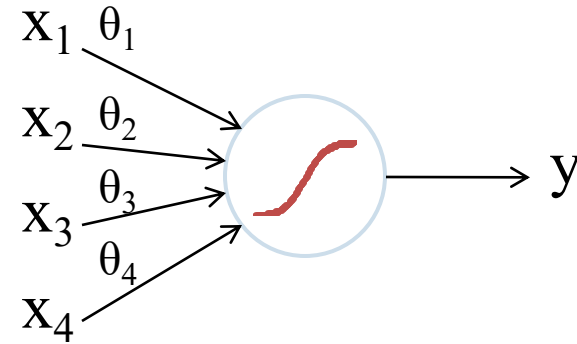
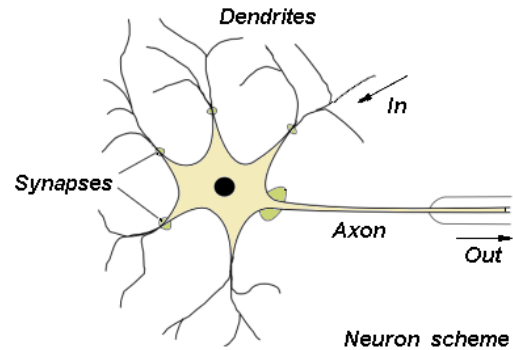


Artificial Neurons

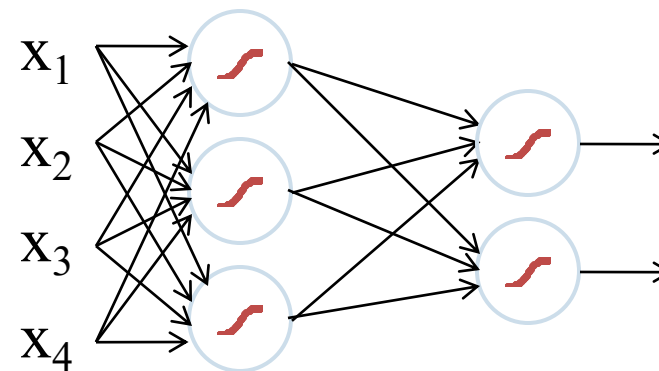
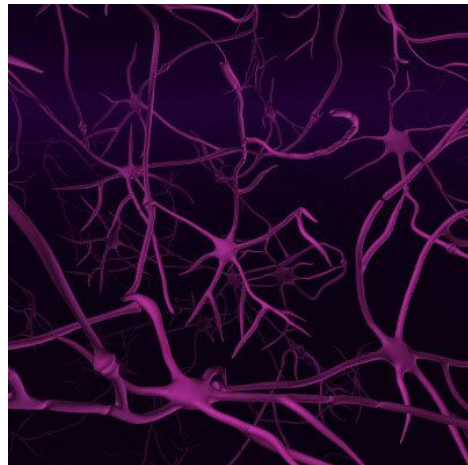


Biological Basis for Neural Networks

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...