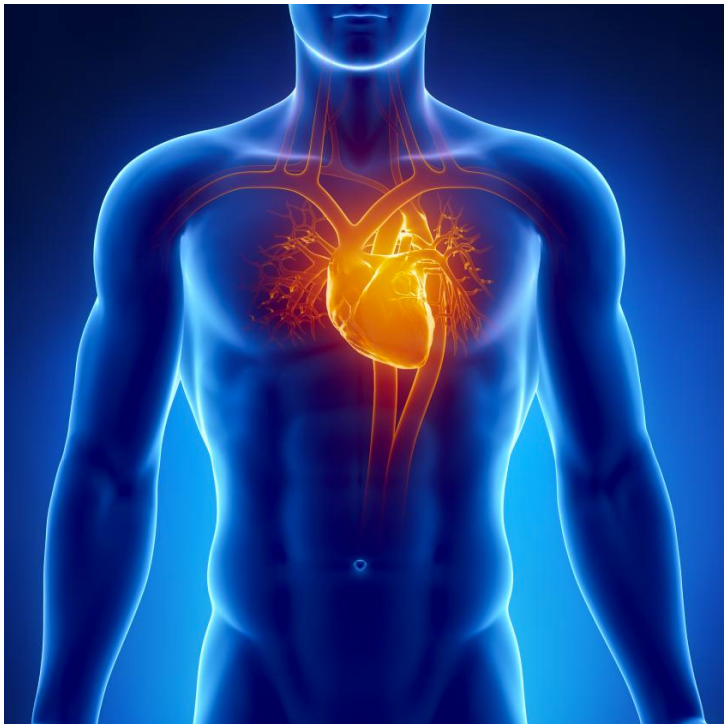
Logistic Regression

Chris Piech
CS109, Stanford University

Classification

Classification Task

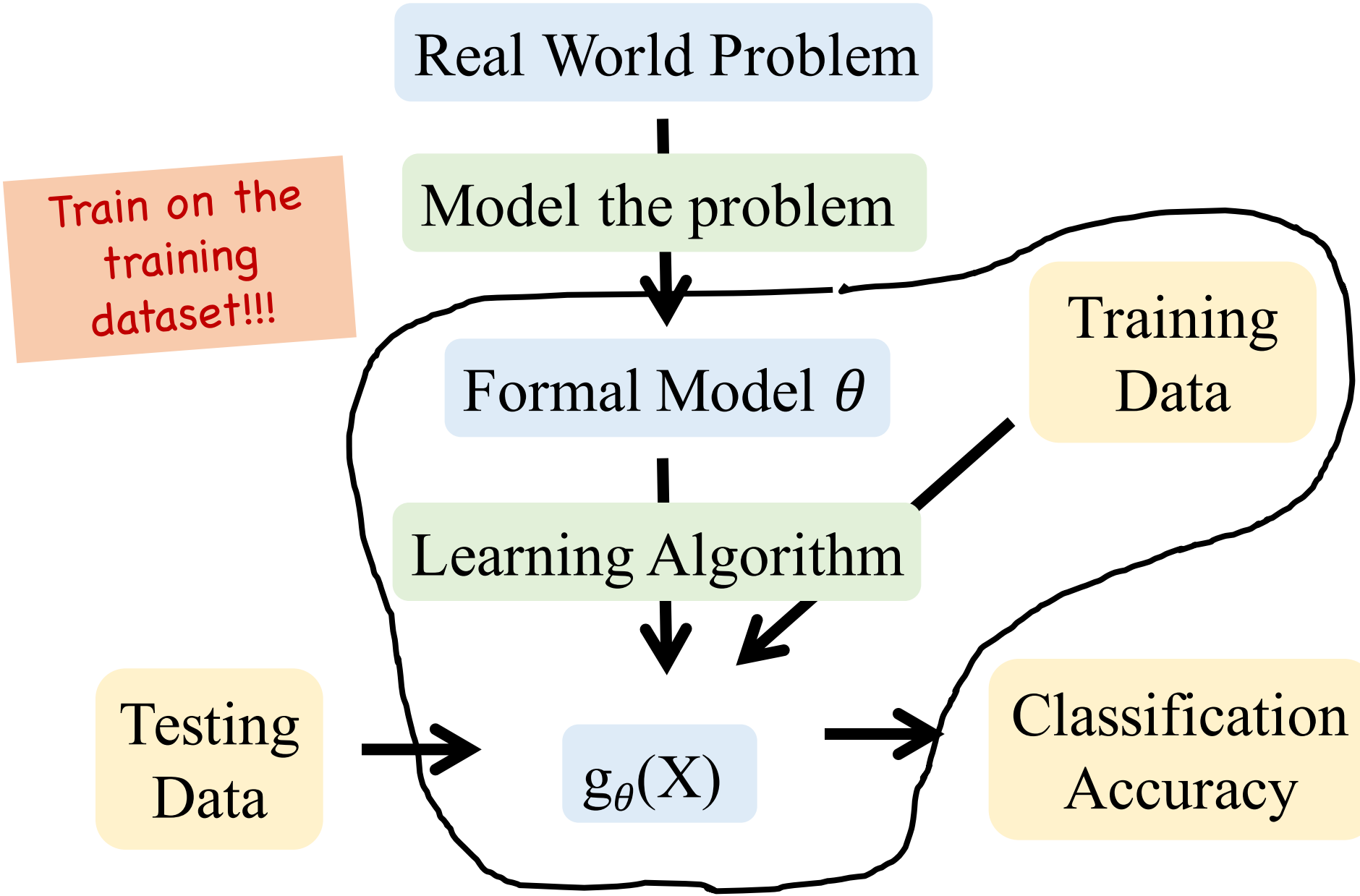
Heart



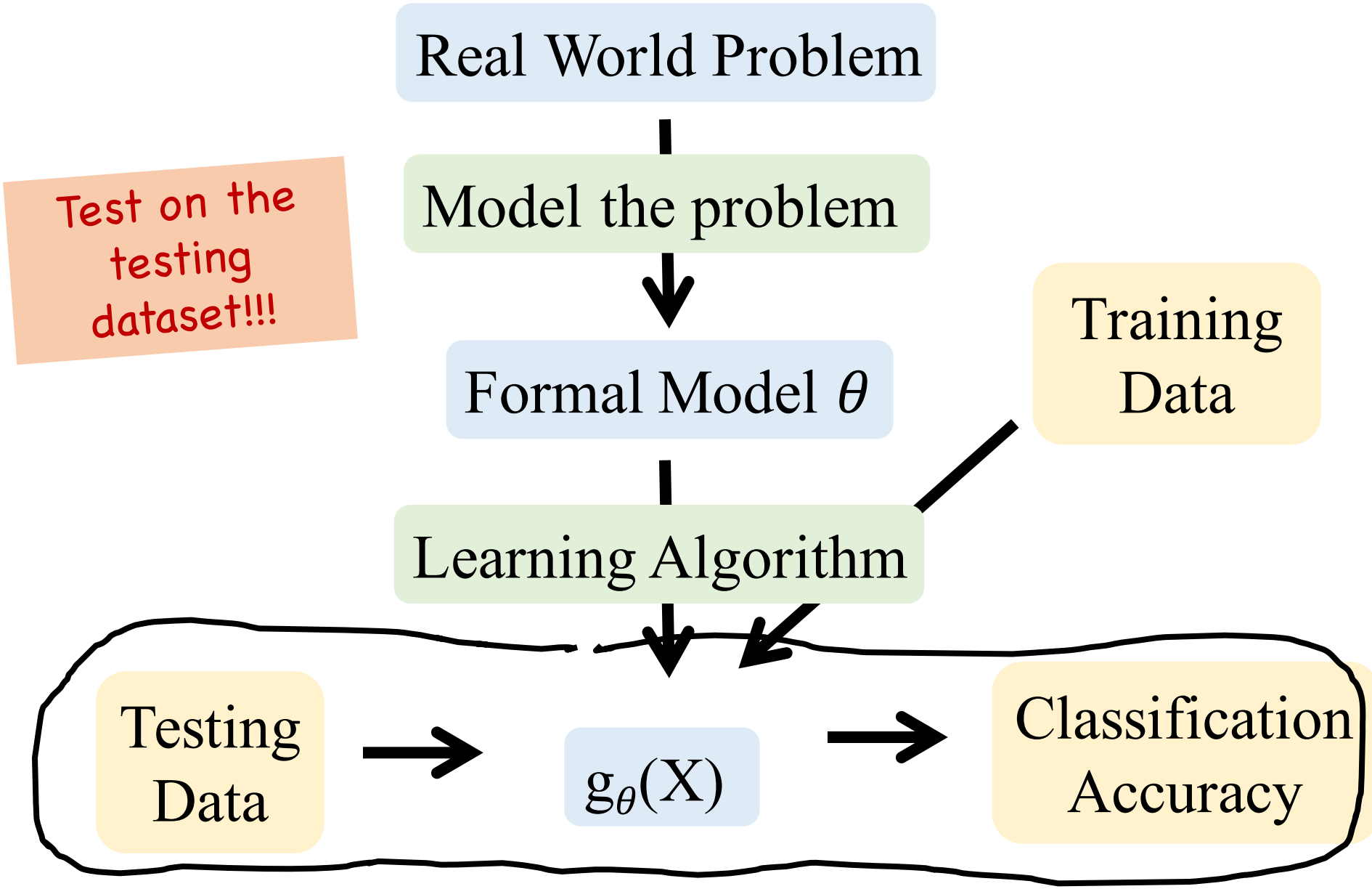
Ancestry

Netflix

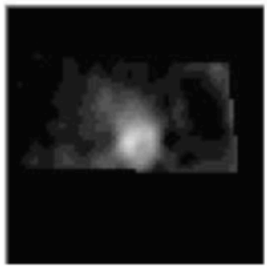
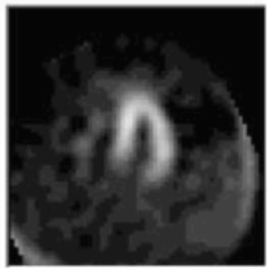
Training


Real World Problem
Model the problem
Train on the training dataset!!!
Formal Model θ
Training Data
Learning Algorithm
Testing Data
$g_\theta(X)$
Classification Accuracy

Testing

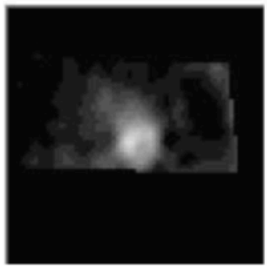
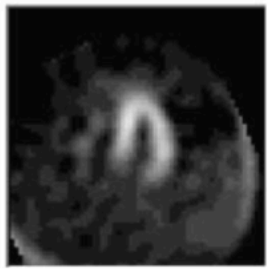

Real World Problem
Model the problem
Formal Model θ
Learning Algorithm
Training Data
Test on the testing dataset!!!
Testing Data
$g_\theta(X)$
Classification Accuracy

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

$$g_\theta(\mathrm{X})$$

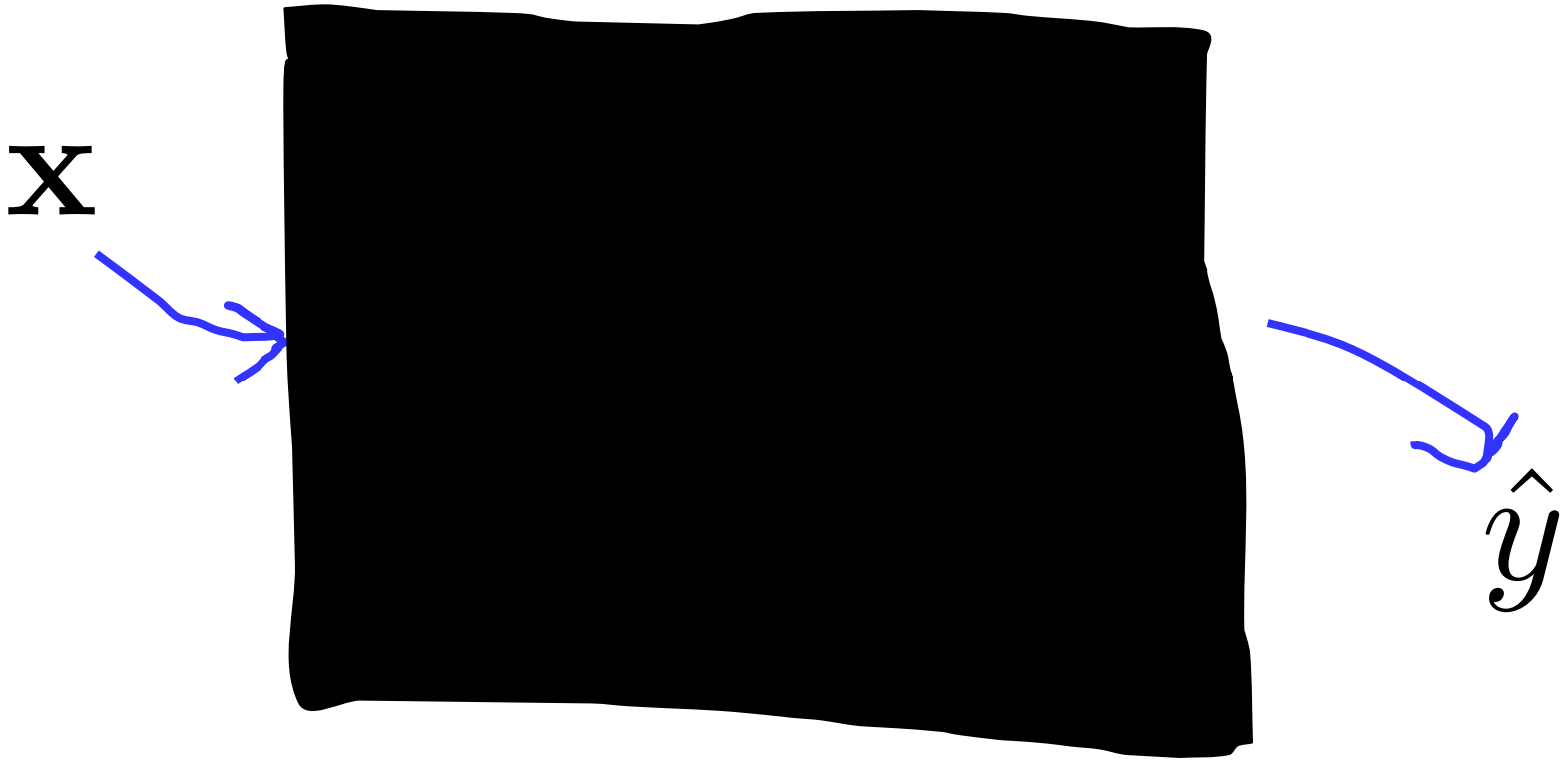
Healthy Heart Classifier

	ROI 1	ROI 2	ROI m	Output
			… 	
New Heart	1	0	1	1

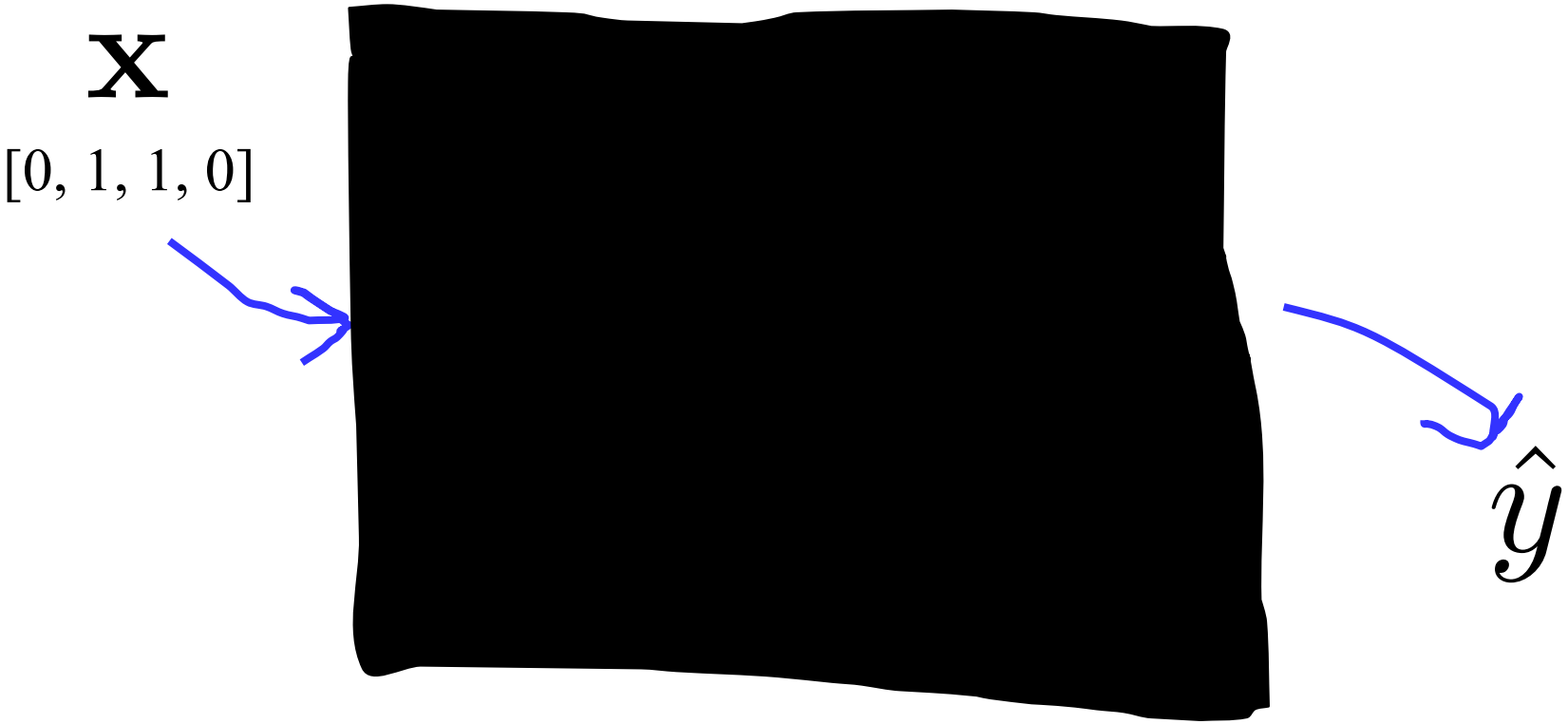
$$g_\theta(X)$$

Naïve Bayes Classification

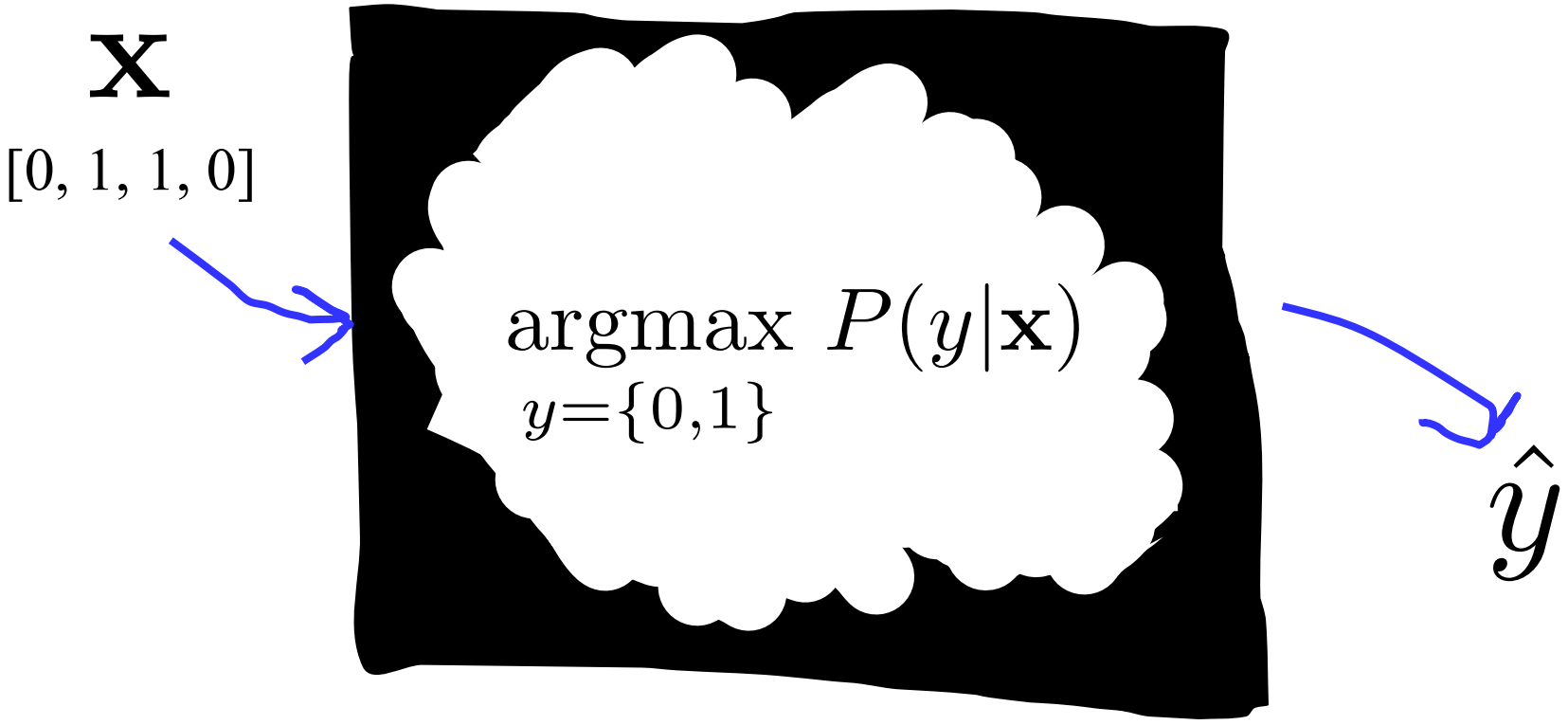
$g_\theta(\mathbf{x})$?



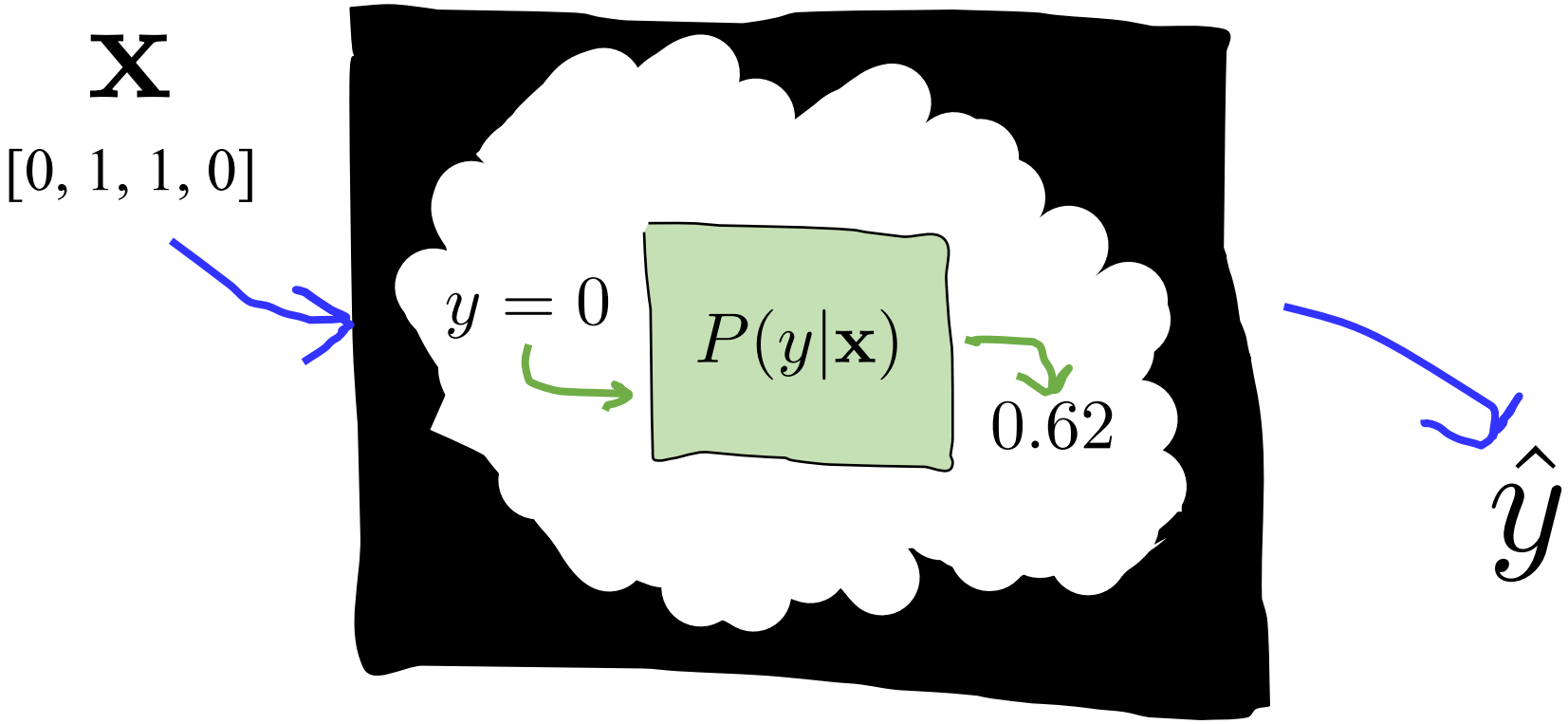
$g_\theta(\mathbf{x})$?



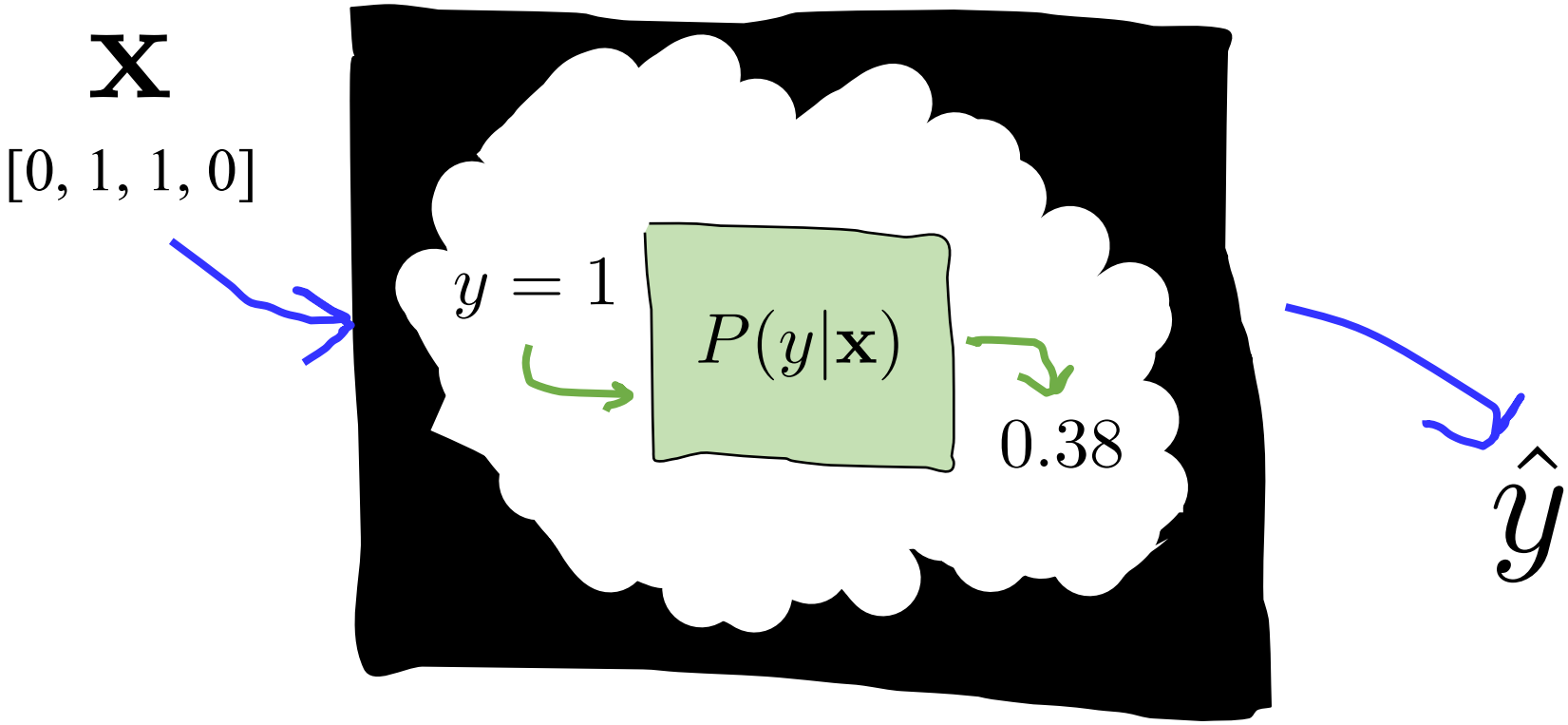
$\mathrm{g}_\theta(\mathbf{x})$?



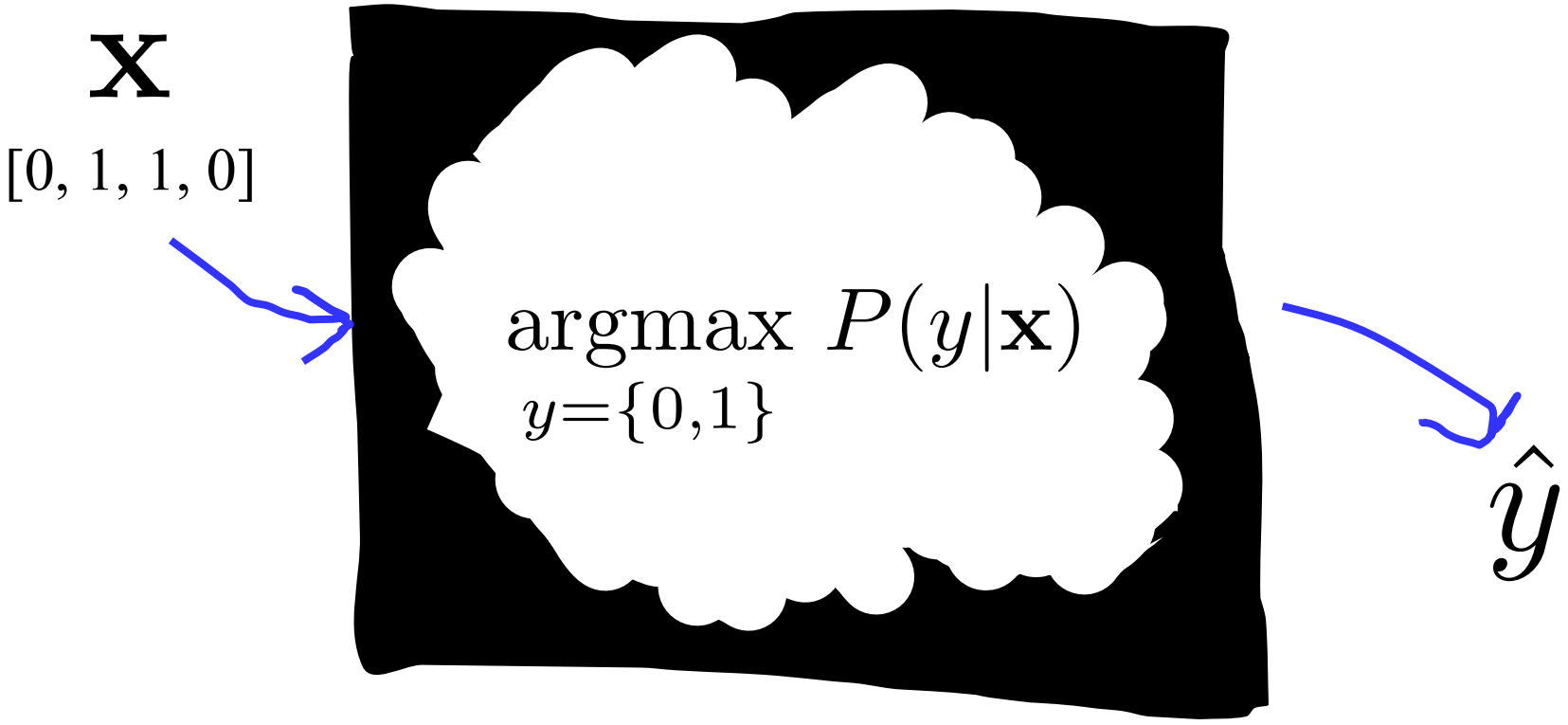
$\mathbf{g}_\theta(\mathbf{x})$?



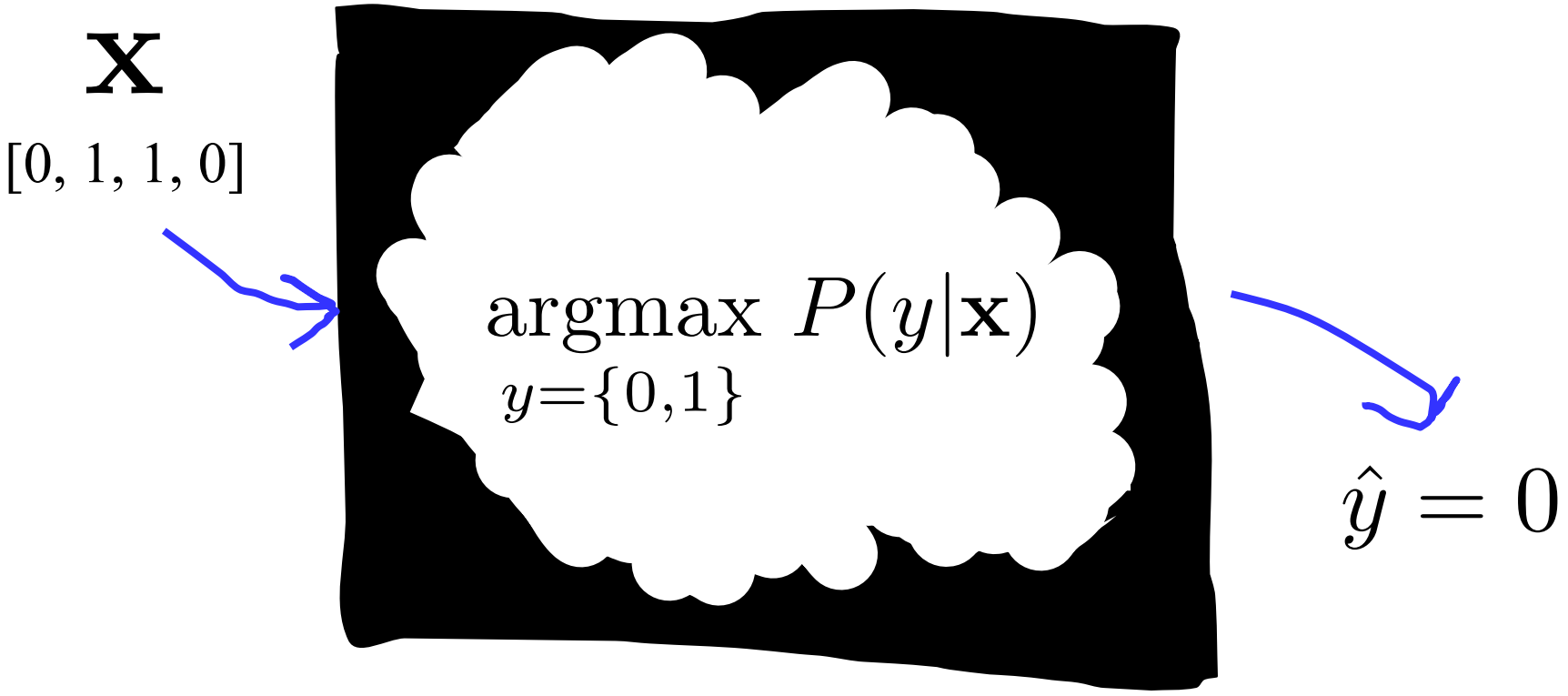
$\mathbf{g}_\theta(\mathbf{x})$?



$\mathrm{g}_\theta(\mathbf{x})$?



$\mathbf{g}_\theta(\mathbf{x})$?



Brute Force Bayes

Simply chose the class label that is the most likely given the data

$$\hat{y} = \underset{y=\{0,1\}}{\operatorname{argmax}} \; P(y|\mathbf{x})$$

$$= \underset{y=\{0,1\}}{\operatorname{argmax}} \; \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}$$

$$= \underset{y=\{0,1\}}{\operatorname{argmax}} \; P(\mathbf{x}|y)P(y)$$

$$P(x_1, x_2, x_3, \ldots, x_{100}|y)$$

Naïve Bayes

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

$$
\begin{aligned}
\hat{y} &= \mathrm{g}_\theta(\mathbf{x}) \\
&= \underset{y=\{0,1\}}{\operatorname{argmax}} \; P(y|\mathbf{x}) \\
&= \underset{y=\{0,1\}}{\operatorname{argmax}} \; \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} \\
&= \underset{y=\{0,1\}}{\operatorname{argmax}} \; P(\mathbf{x}|y)P(y)
\end{aligned}
$$

Naïve Bayes

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

$$
\begin{aligned}
\hat{y} &= \mathrm{g}_\theta(\mathbf{x}) \\
&= \underset{y=\{0,1\}}{\operatorname{argmax}} \; P(y|\mathbf{x}) \\
&= \underset{y=\{0,1\}}{\operatorname{argmax}} \; \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} \\
&= \underset{y=\{0,1\}}{\operatorname{argmax}} \; P(\mathbf{x}|y)P(y) \\
&= \underset{y=\{0,1\}}{\operatorname{argmax}} \; \prod_i P(x_i|y)P(y)
\end{aligned}
$$

By Naïve Bayes Assumption

Naïve Bayes

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

$$
\begin{aligned}
\hat{y} &= \mathrm{g}_\theta(\mathbf{x}) \\
&= \underset{y=\{0,1\}}{\operatorname{argmax}} \; P(y|\mathbf{x}) \\
&= \underset{y=\{0,1\}}{\operatorname{argmax}} \; \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} \\
&= \underset{y=\{0,1\}}{\operatorname{argmax}} \; P(\mathbf{x}|y)P(y) \\
&= \underset{y=\{0,1\}}{\operatorname{argmax}} \; \prod_i P(x_i|y)P(y) \\
&= \underset{y=\{0,1\}}{\operatorname{argmax}} \; \log P(y) + \sum_i \log P(x_i|y)
\end{aligned}
$$

Argmax of log

Computing Probabilities from Data

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

$$\hat{P}(Y=0) = \frac{\#\text{instances in class} = 0}{\text{total}\,\#\,\text{instances}}$$

$$\hat{P}(X_i = 0, Y = 0) = \frac{\#\text{instances where } X_i = 0 \text{ and class} = 0}{\text{total}\,\#\,\text{instances}}$$

$$\hat{P}(X_i = 0 \mid Y = 0) = \frac{\hat{P}(X_i = 0, Y = 0)}{\hat{P}(Y = 0)} \qquad \hat{P}(X_i = 0 \mid Y = 1) = \frac{\hat{P}(X_i = 0, Y = 1)}{\hat{P}(Y = 1)}$$

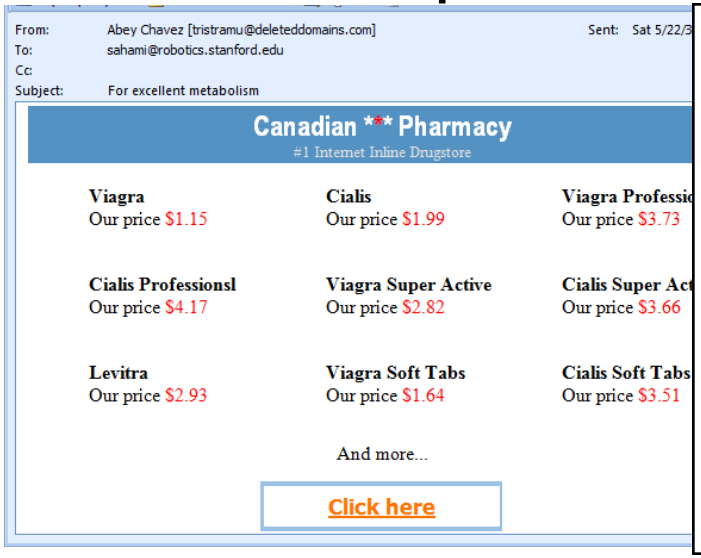
$$\hat{P}(X_i = 1 \mid Y = 0) = 1 - \hat{P}(X_i = 0 \mid Y = 0)$$

Training Naïve Bayes, is estimating parameters for a multinomial.

Thus training is just counting.

What is Bayes Doing in my Mail Server

- This is spam:


From: Abey Chavez [tristramu@deleteddomains.com] Sent: Sat 5/22/3
To: sahami@robotics.stanford.edu
Cc:
Subject: For excellent metabolism

Canadian * Pharmacy**
#1 Internet Inline Drugstore

Viagra Our price $1.15	**Cialis** Our price $1.99	**Viagra Professio** Our price $3.73
Cialis Professionsl Our price $4.17	**Viagra Super Active** Our price $2.82	**Cialis Super Act** Our price $3.66
Levitra Our price $2.93	**Viagra Soft Tabs** Our price $1.64	**Cialis Soft Tabs** Our price $3.51

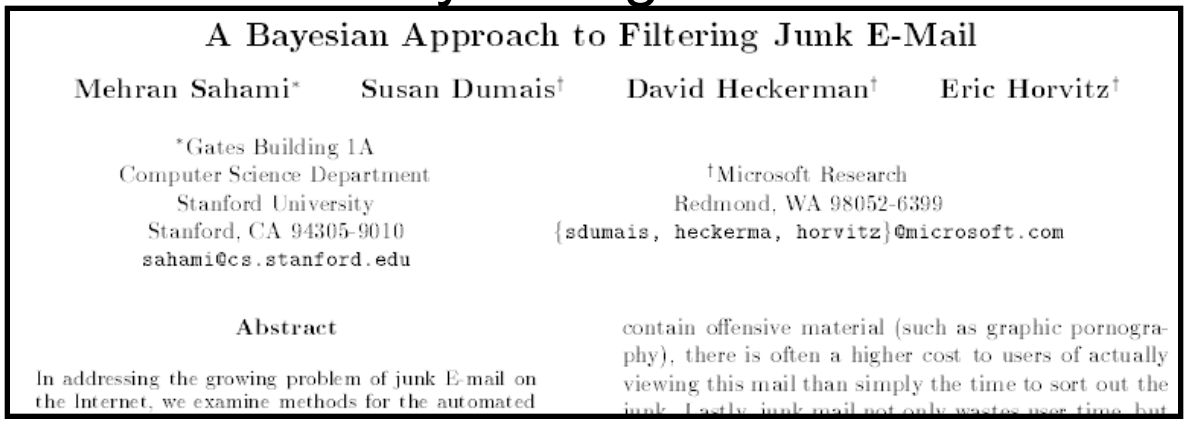
And more...

Click here

Let's get Bayesian on your spam:

```
Content analysis details:    (49.5 hits, 7.0 required)
 0.9 RCVD_IN_PBL             RBL: Received via a relay in Spamhaus PBL
                             [93.40.189.29 listed in zen.spamhaus.org]
 1.5 URIBL_WS_SURBL          Contains an URL listed in the WS SURBL blocklist
                             [URIs: recragas.cn]
 5.0 URIBL_JP_SURBL          Contains an URL listed in the JP SURBL blocklist
                             [URIs: recragas.cn]
 5.0 URIBL_OB_SURBL          Contains an URL listed in the OB SURBL blocklist
                             [URIs: recragas.cn]
 5.0 URIBL_SC_SURBL          Contains an URL listed in the SC SURBL blocklist
                             [URIs: recragas.cn]
 2.0 URIBL_BLACK             Contains an URL listed in the URIBL blacklist
                             [URIs: recragas.cn]
 8.0 BAYES_99                BODY: Bayesian spam probability is 99 to 100%
                             [score: 1.0000]
```

Who was crazy enough to think of that?


A Bayesian Approach to Filtering Junk E-Mail

Mehran Sahami* Susan Dumais† David Heckerman† Eric Horvitz†

*Gates Building 1A
Computer Science Department
Stanford University
Stanford, CA 94305-9010
sahami@cs.stanford.edu

†Microsoft Research
Redmond, WA 98052-6399
{sdumais, heckerma, horvitz}@microsoft.com

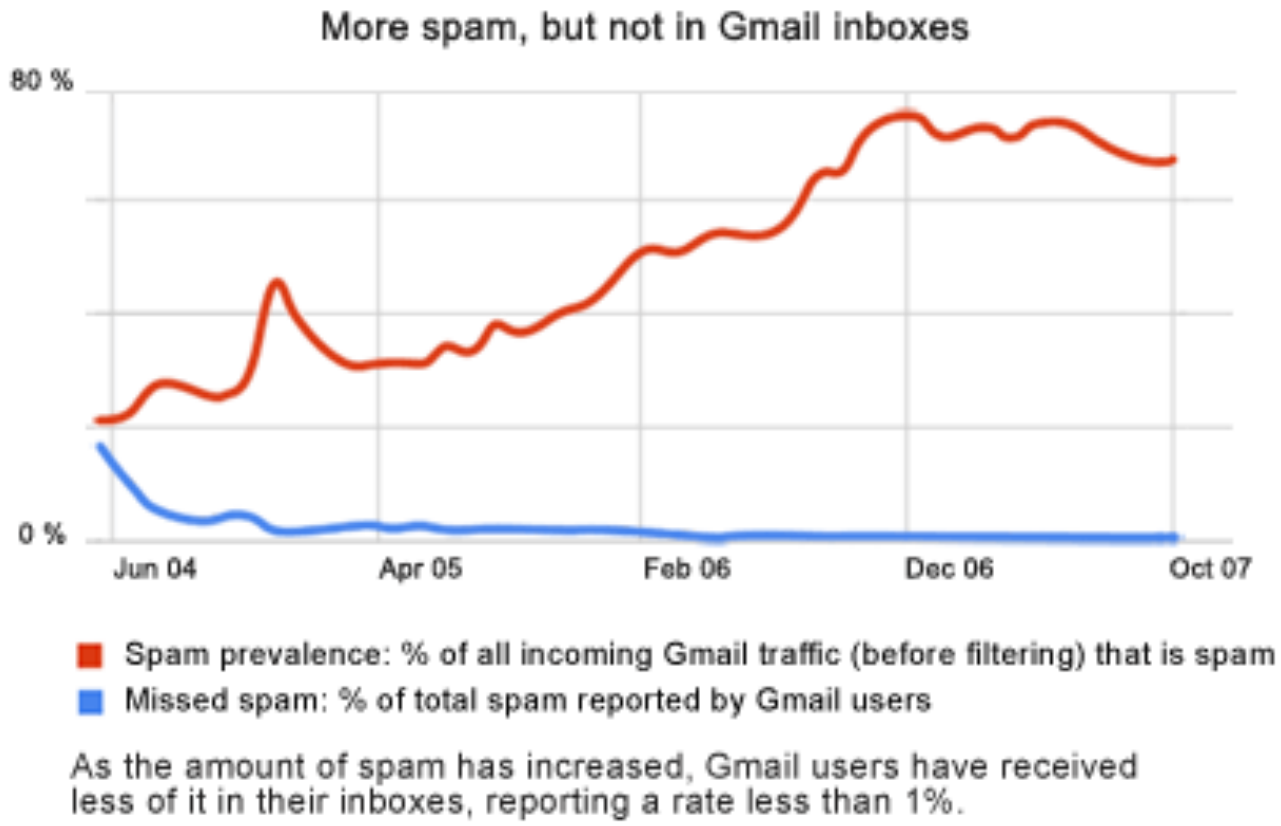
Abstract

In addressing the growing problem of junk E-mail on the Internet, we examine methods for the automated

contain offensive material (such as graphic pornography), there is often a higher cost to users of actually viewing this mail than simply the time to sort out the junk. Lastly, junk mail not only wastes user time, but

Spam, Spam… Go Away!

- The constant battle with spam



Email Classification

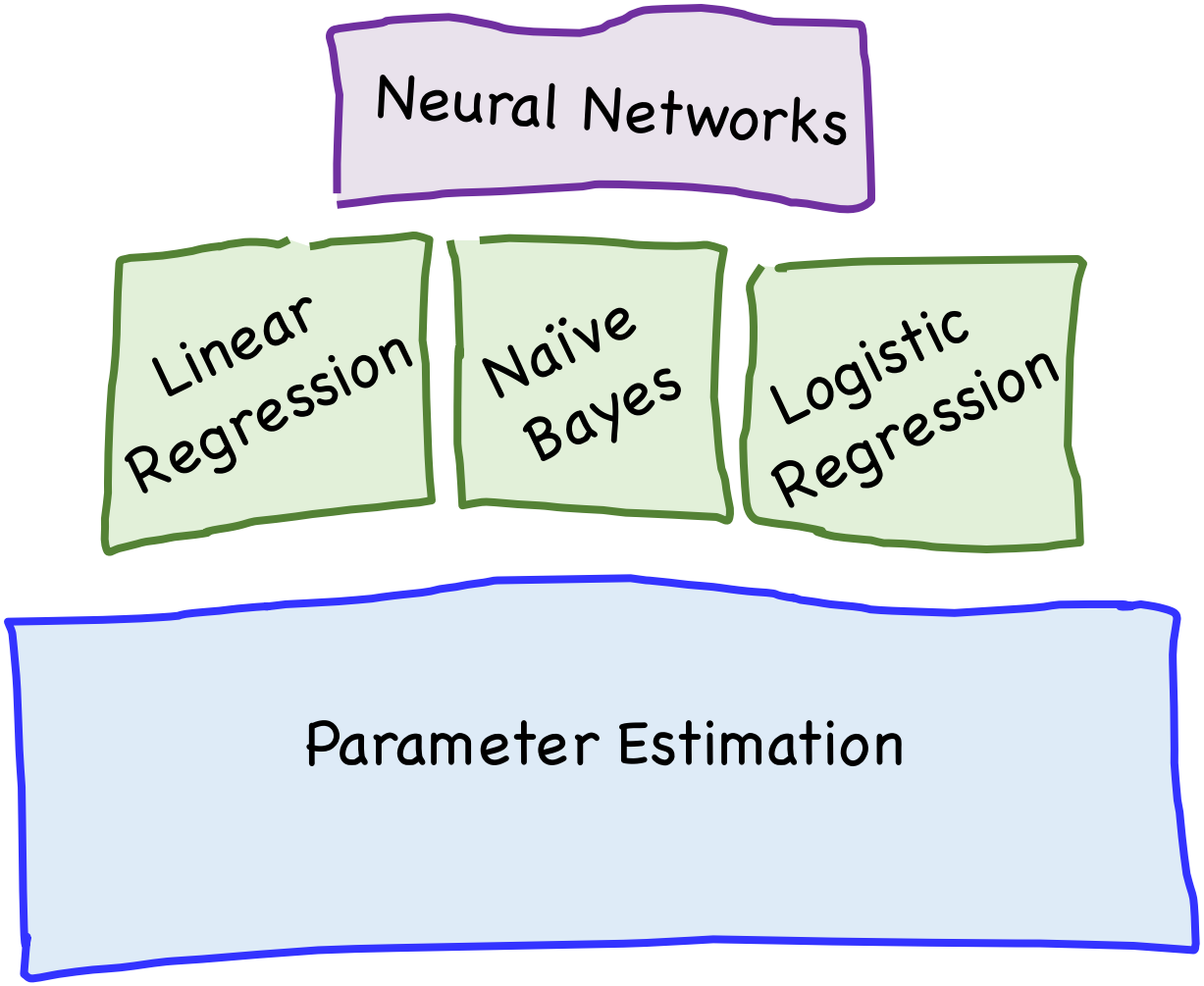
- Want to predict if an email is spam or not
  - Start with the input data
    - Consider a lexicon of *m* words (Note: in English $m \approx 100{,}000$)
    - Define *m* indicator variables $\mathbf{X} = \langle X_1, X_2, \ldots, X_m \rangle$
    - Each variable X_i denotes if word *i* appeared in a document or not
    - Note: *m* is huge, so make "Naive Bayes" assumption
  - Define output classes Y to be: {spam, non-spam}
  - Given training set of *N* previous emails
    - For each email message, we have a training instance:
      $\mathbf{X} = \langle X_1, X_2, \ldots, X_m \rangle$ noting for each word, if it appeared in email
    - Each email message is also marked as spam or not (value of Y)

How Does This Do?

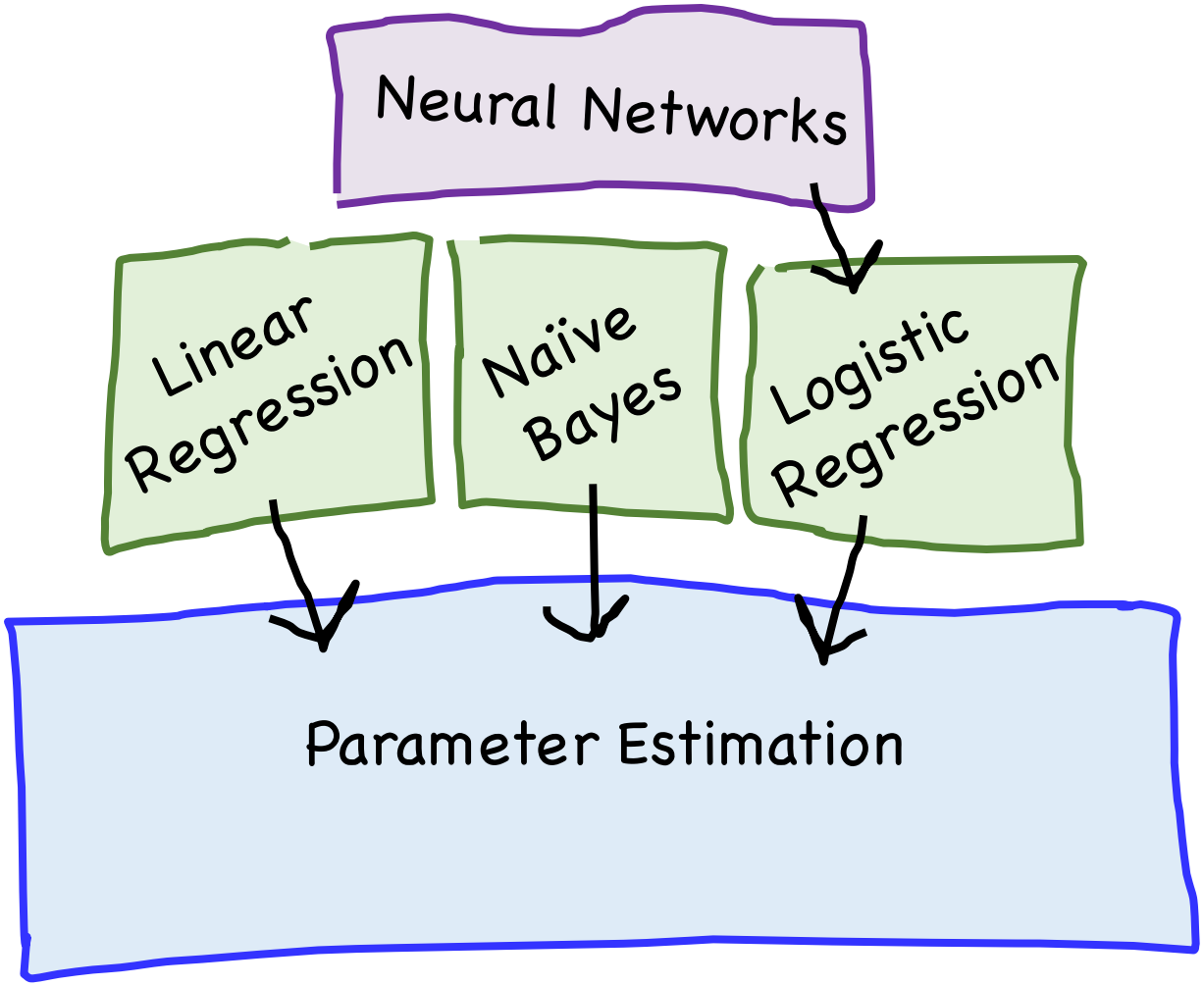
- After training, can test with another set of data
  - "Testing" set also has known values for Y, so we can see how often we were right/wrong in predictions for Y
  - Spam data
    - Email data set: 1789 emails (1578 spam, 211 non-spam)
    - First, 1538 email messages (by time) used for training
    - Next 251 messages used to test learned classifier
  - Criteria:
    - Precision = # *correctly* predicted class Y/ # predicted class Y
    - Recall = # *correctly* predicted class Y / # real class Y messages

	Spam		Non-spam	
	Precision	**Recall**	**Precision**	**Recall**
Words only	**97.1%**	**94.3%**	**87.7%**	**93.4%**
Words + add'l features	**100%**	**98.3%**	**96.2%**	**100%**

Our Path


Neural Networks
Linear Regression
Naïve Bayes
Logistic Regression
Parameter Estimation

Knowledge Dependency



Logistic Regression

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

Chapter 1: Big Picture

From Naïve Bayes to Logistic Regression

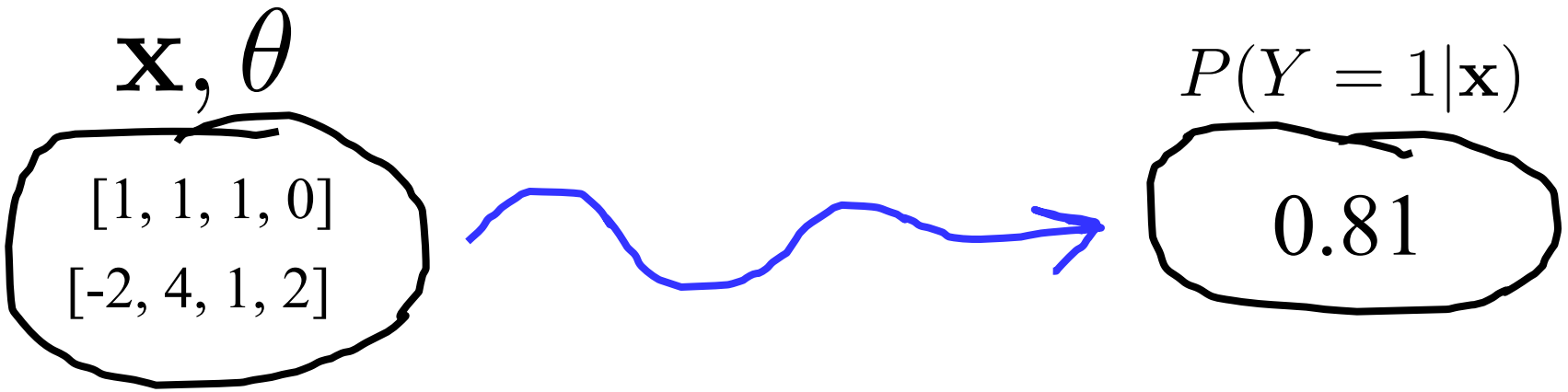
- For classification we care about P(Y | **X**)

- Recall the Naive Bayes Classifier
  - Predict $\hat{Y} = \arg\max_y P(\boldsymbol{X}, Y) = \arg\max_y P(\boldsymbol{X} \mid Y) P(Y)$
  - 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!

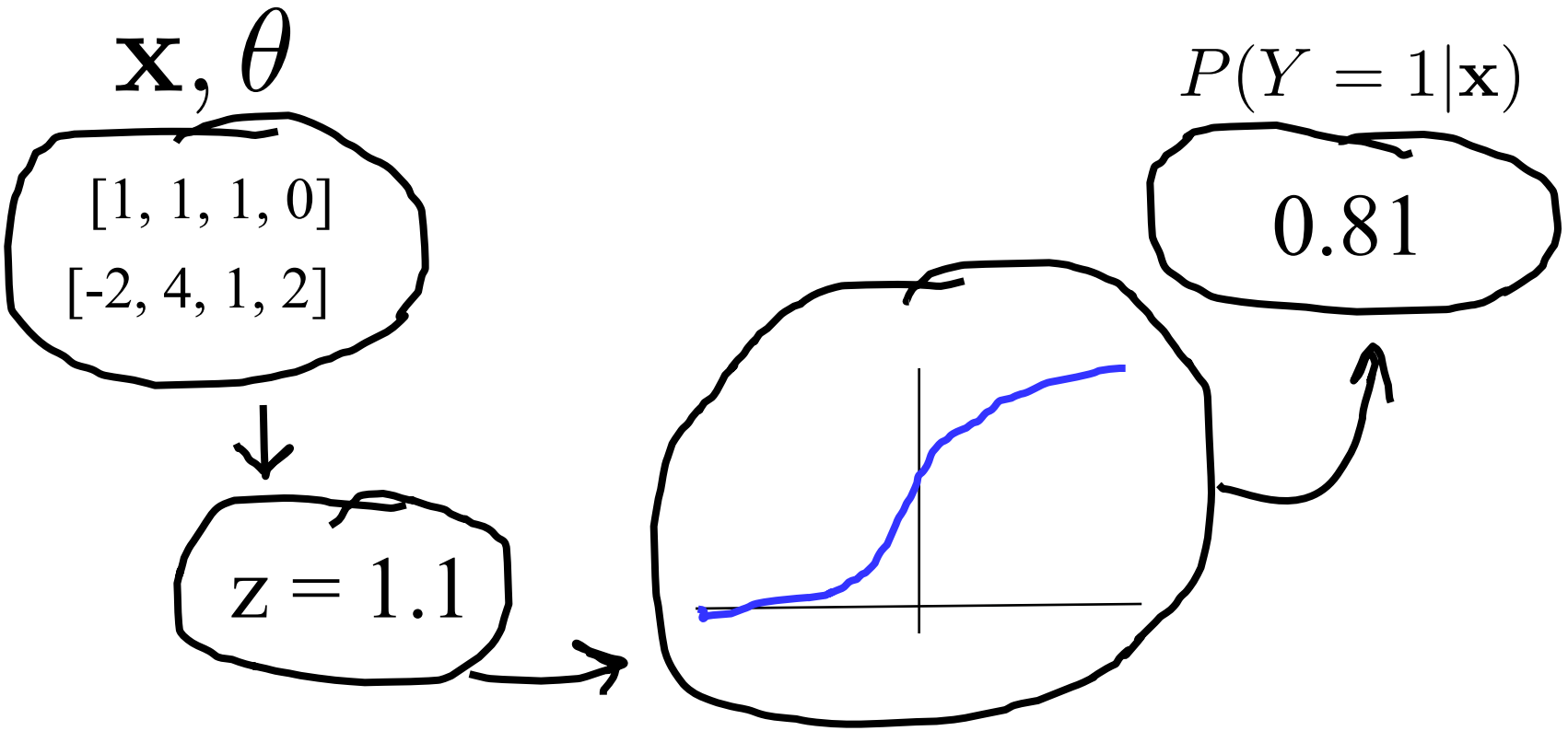
Logistic Regression Assumption

- 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!



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

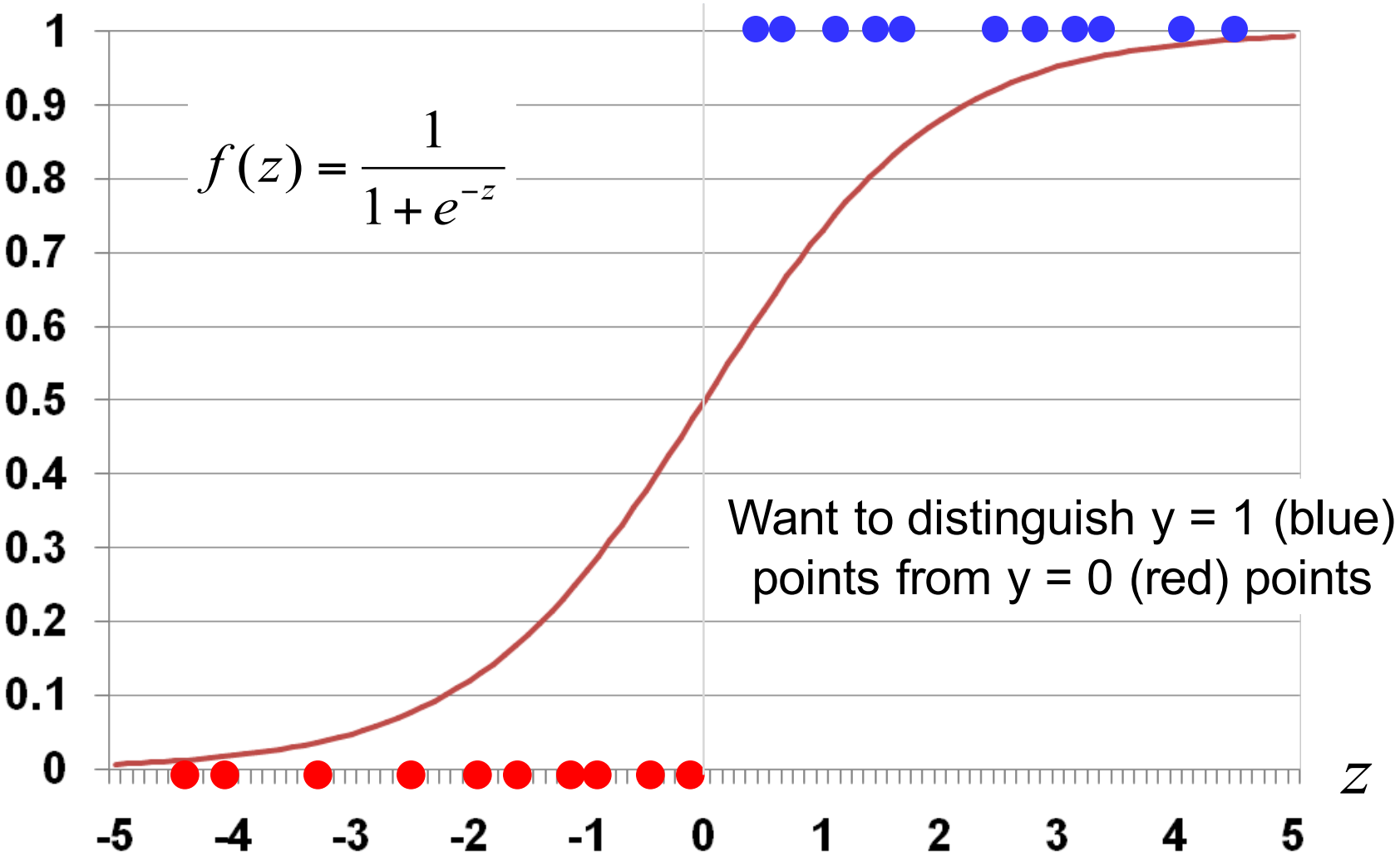
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

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

Recall:
Sigmoid function

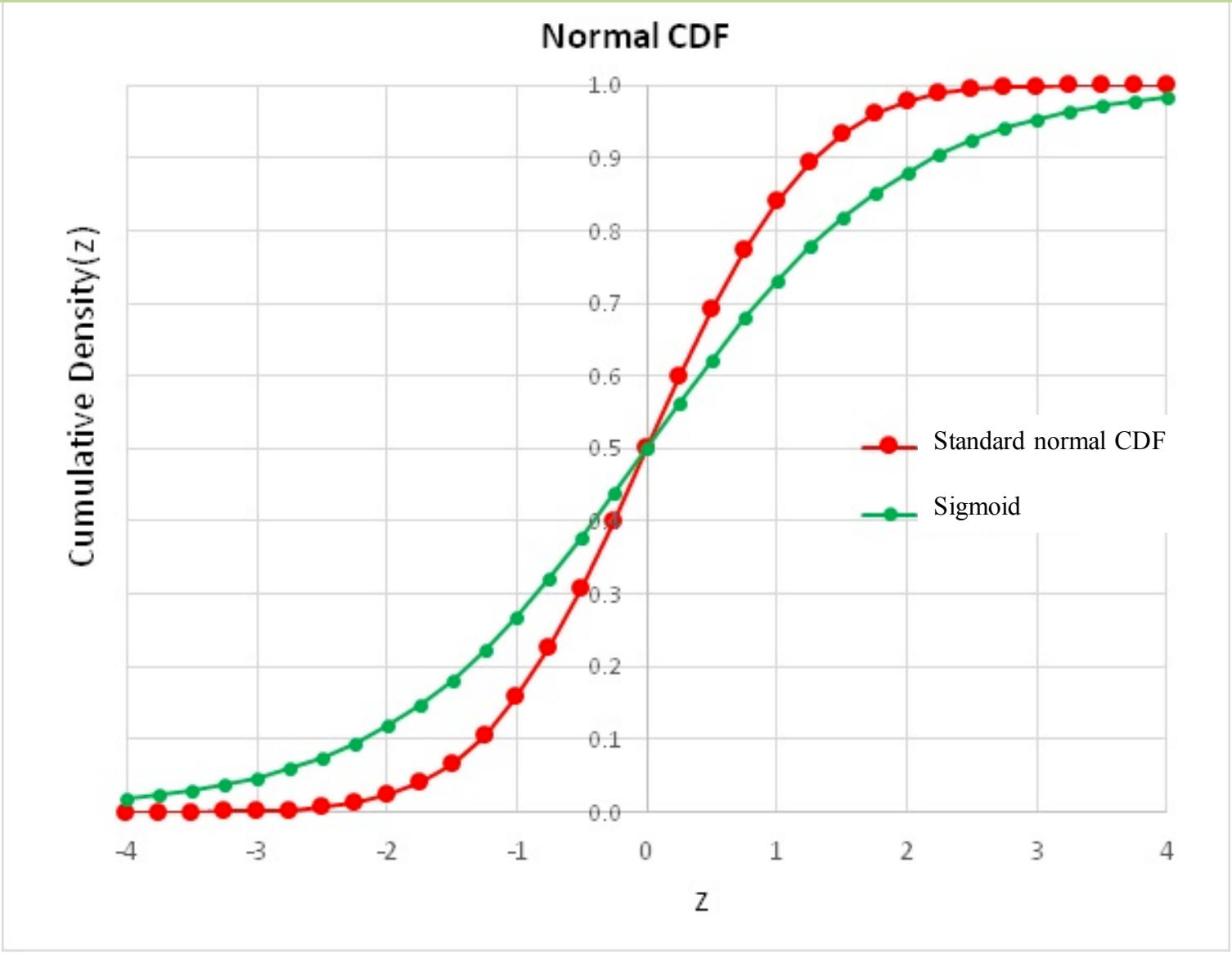
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

The Sigmoid Function

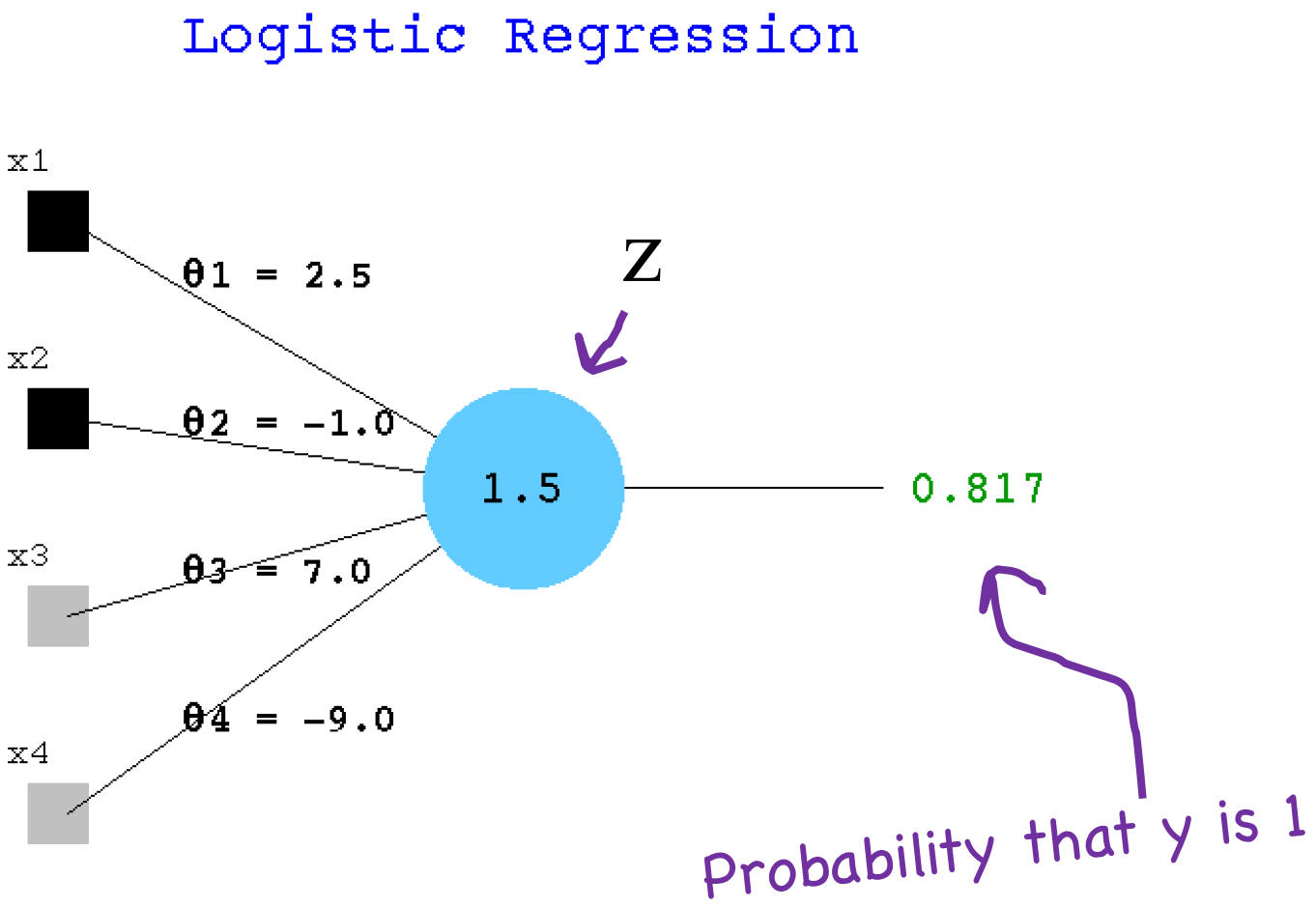


Note: inflection point at z = 0. $f(0) = 0.5$

The Sigmoid Function



Logistic Regression Example



What is in a Name

Regression Algorithms

Linear Regression 

Classification Algorithms

Naïve Bayes 

Logistic Regression 

Awesome classifier, terrible name

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

Math for Logistic Regression

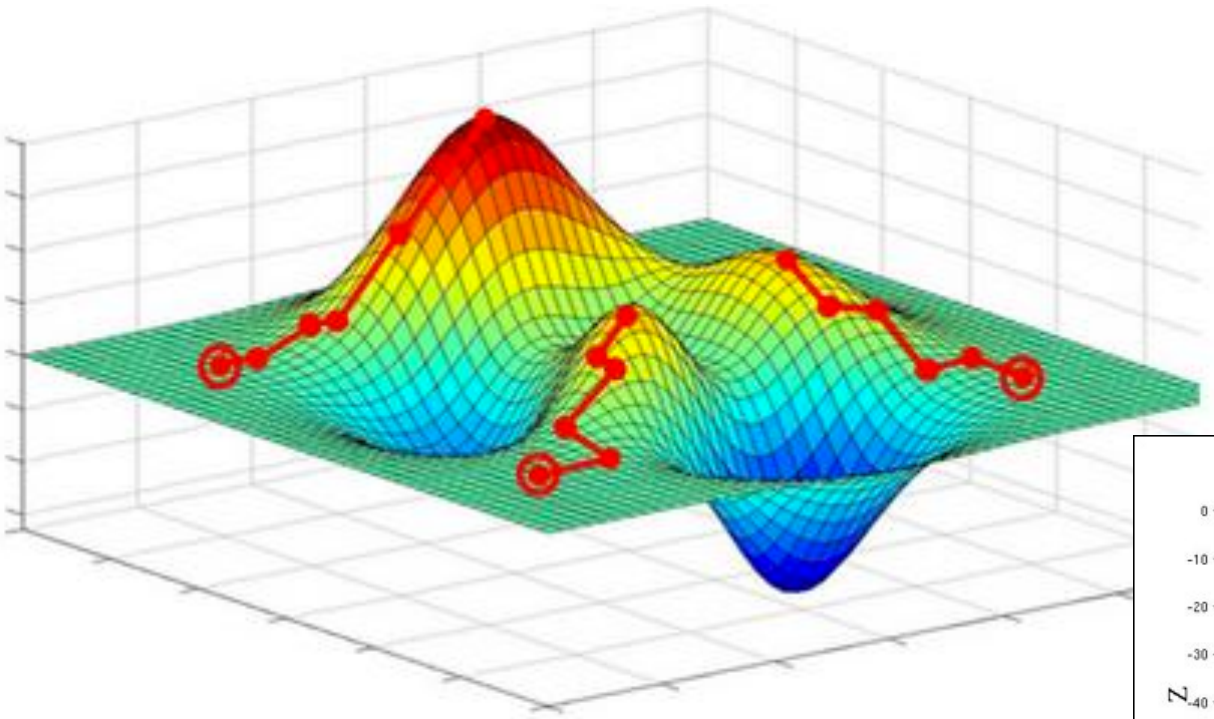
1 Make logistic regression assumption

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

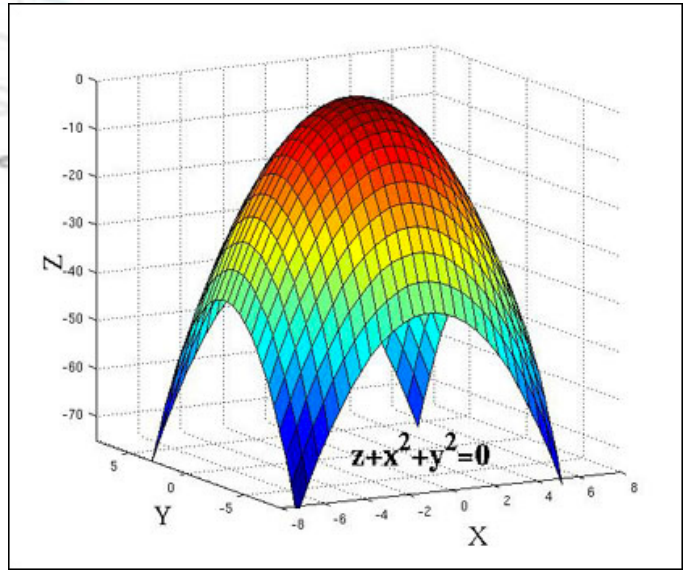
2 Calculate the log likelihood for all data

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

Gradient Ascent



Logistic regression LL function is convex



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

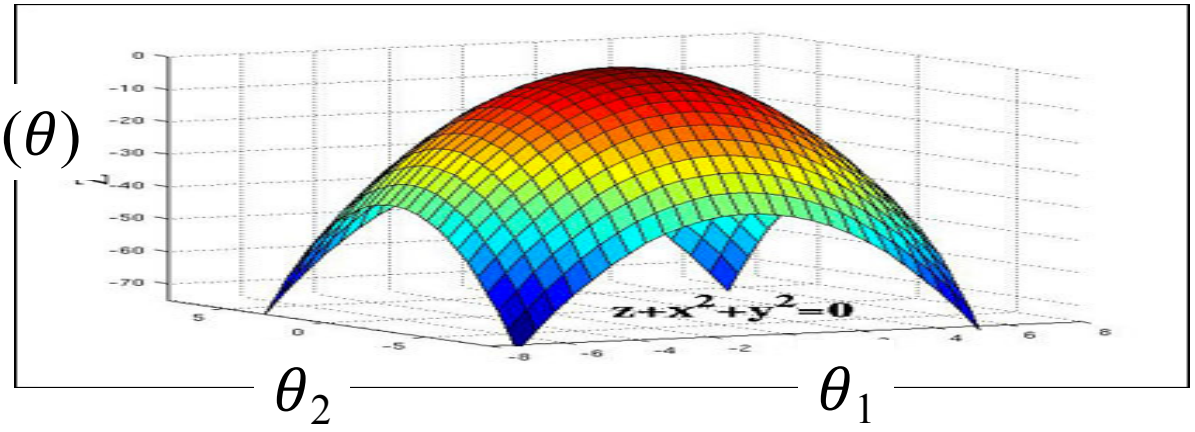
Gradient Ascent Step

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

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

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

Do this for all thetas!



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

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

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

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 $0 \le j \le m$

Repeat many times:

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

For each training example (x, y):

For each parameter j:

Update gradient[j] *for current training example*

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

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

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

Don’t forget:

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

Allows for θ_0 to be an intercept.

Classification with Logistic Regression

- Training: determine parameters θ_j (for all $0 \le j \le m$)
  - After parameters θ_j have been learned, test classifier
- To test classifier, for each new (test) instance ***X***:
  - Compute: $p = P(Y = 1 \mid \boldsymbol{X}) = \dfrac{1}{1 + e^{-z}}$, where $z = \theta^T \mathbf{x}$
  - Classify instance as: $\hat{y} = \begin{cases} 1 & p > 0.5 \\ 0 & \text{otherwise} \end{cases}$
  - Note about evaluation set-up: parameters θ_j are **not** updated during "testing" phase

Chapter 2: How Come?

Logistic Regression

1 Make logistic regression assumption

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

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

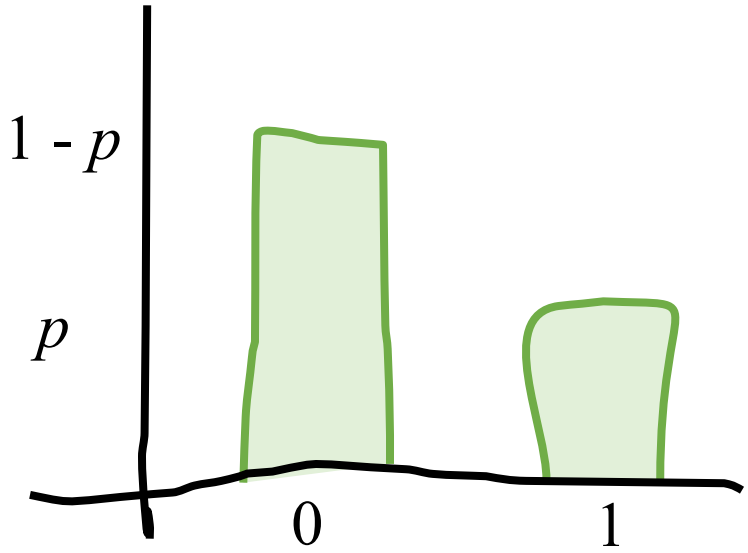
$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=0}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

How did we get that LL function?

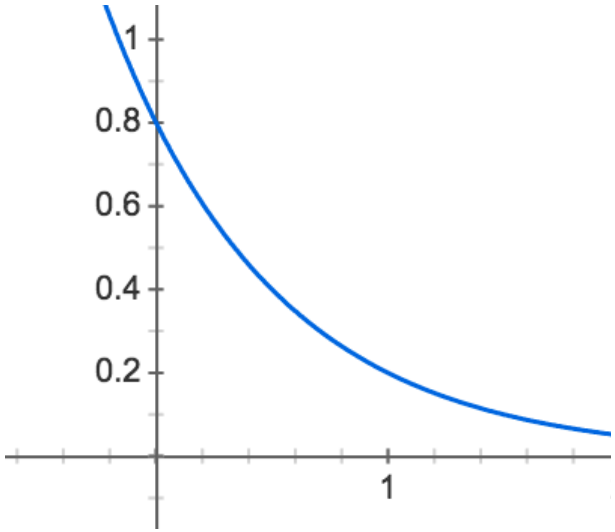
Recall: PMF of Bernoulli

- 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

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

How did we get that gradient?

Sigmoid has a Beautiful Slope

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

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

True fact about sigmoid functions

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

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

Gradient Update

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

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

Imagine only one data point

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

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

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

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

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

For many data points

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

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

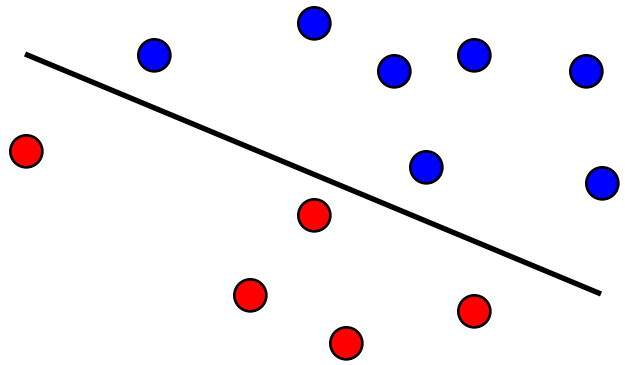
3 Get derivative of log probability with respect to thetas

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

Chapter 3: Philosophy

Discrimination Intuition

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



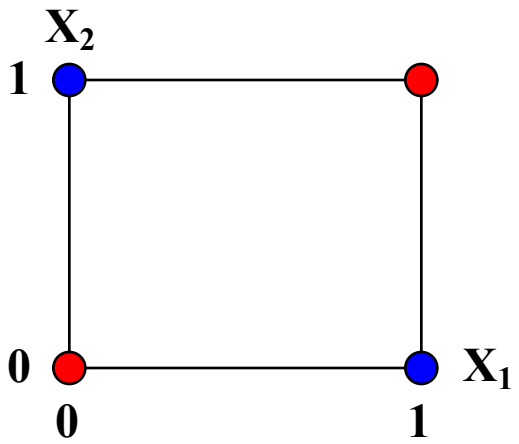
$$\theta^T \mathbf{x} = 0$$

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

- 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 Separable

- Some data sets/functions are not separable
  - Consider function: $y = x_1$ XOR x_2
  - Note: y =1 iff **one** of either x_1 or x_2 = 1



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

Logistic Regression vs Naïve Bayes

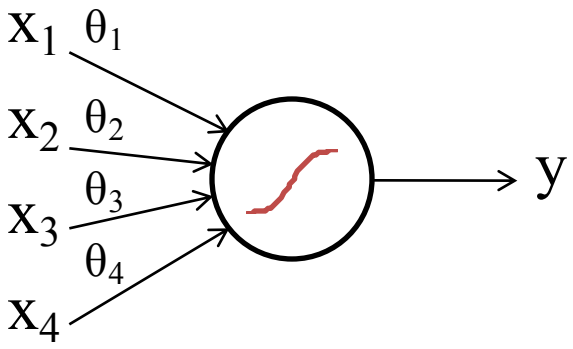
- Compare Naive Bayes and Logistic Regression
  - Recall that Naive Bayes models P(**X**, Y) = P(**X** | Y) P(Y)
  - Logistic Regression directly models P(Y | **X**)
  - We call Naive Bayes a "generative model"
    - Tries to model joint distribution of how data is "generated"
    - I.e., could use P(X, Y) to generate new data points if we wanted
    - But lots of effort to model something that may not be needed
  - We call Logistic Regression a "discriminative model"
    - Just tries to model way to discriminate y = 0 vs. y = 1 cases
    - Cannot use model to generate new data points (no P(X, Y))
    - Note: Logistic Regression can be generalized to more than two output values for y (have multiple sets of parameters β_j)

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 data 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.

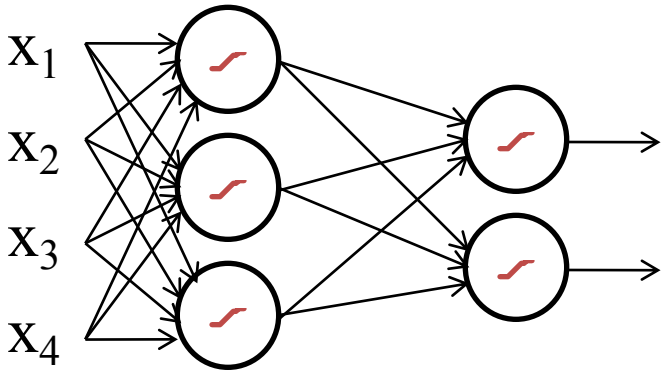
Logistic Regression and Neural Networks

- Consider logistic regression as:



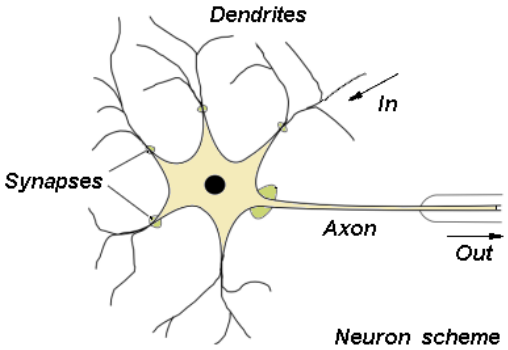
Logistic regression is same as a one node neural network

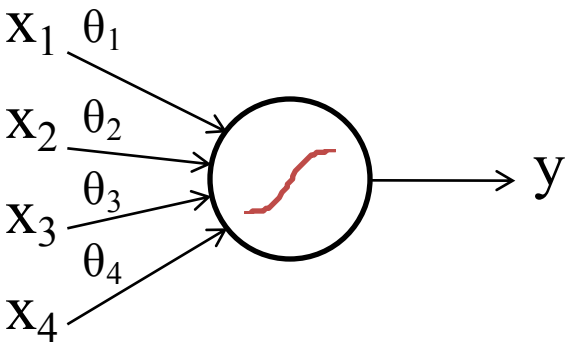
- Neural network



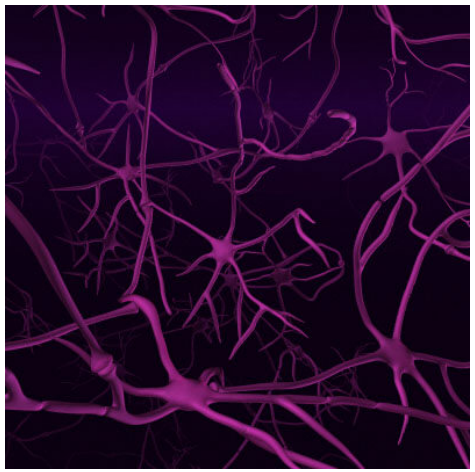
Biological Basis for Neural Networks

- A neuron

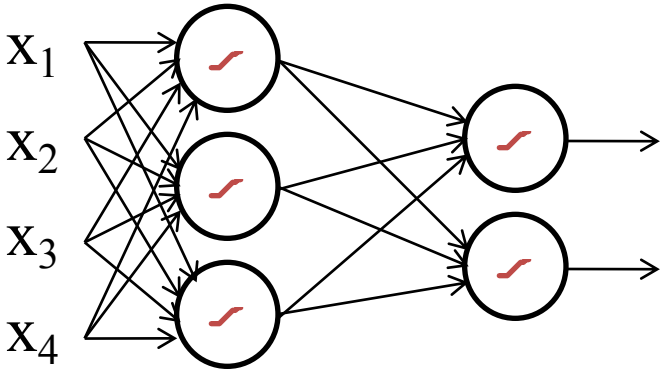




- Your brain



Actually, it's probably someone else's brain



Next up: Deep Learning!