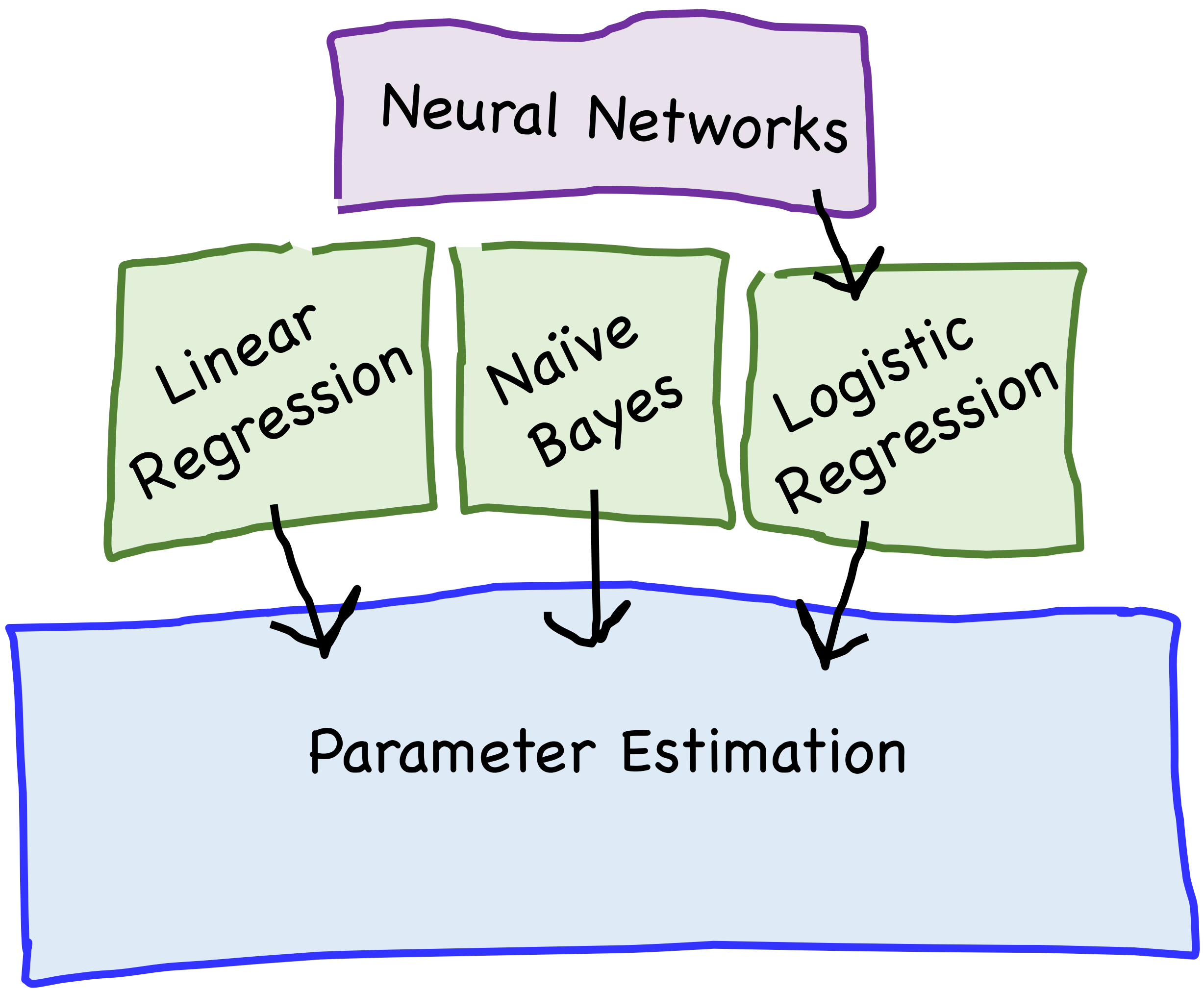
From Logistic Regression to Deep Learning

Chris Piech
CS109, Stanford University

Knowledge Dependency



Important Mathematical Journey

Review

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

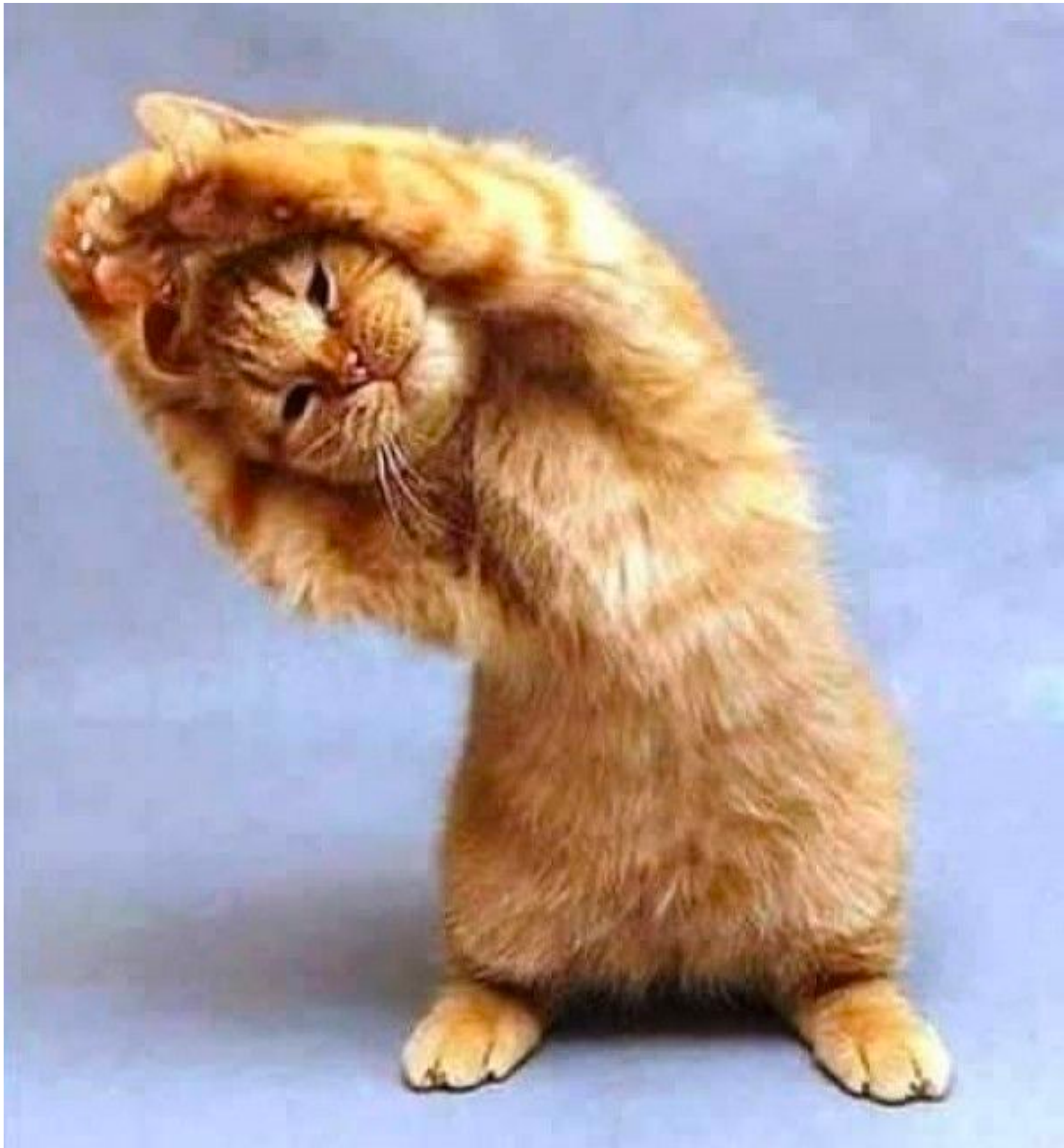
Warmup

$$\frac{\partial}{\partial \theta_j} \theta^T \mathbf{x}?$$

$$= \frac{\partial}{\partial \theta_j} \sum_{i=0}^{n} x_i \theta_i$$

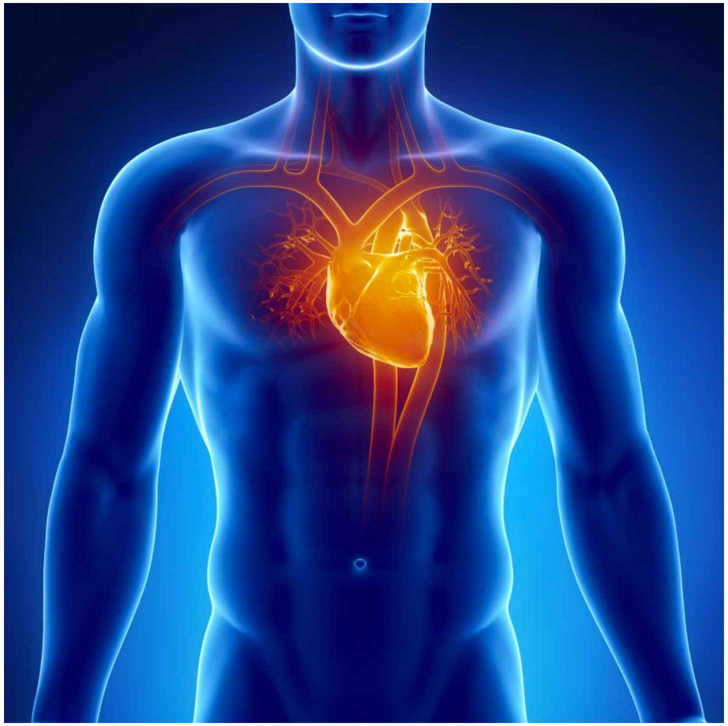
$$= \sum_{i=0}^{n} \frac{\partial}{\partial \theta_j} x_i \theta_i$$

$$= x_j$$



Classification Task

Heart



Ancestry

Netflix

Training Data

Assume IID data:

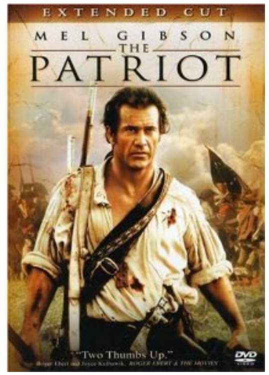
N training datapoints

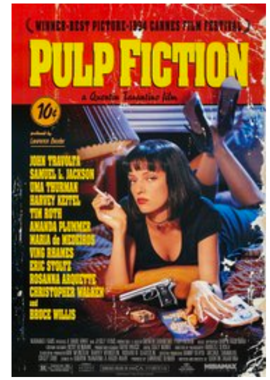
$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots (\mathbf{x}^{(n)}, y^{(n)})$$

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

Each datapoint has m features and a single output

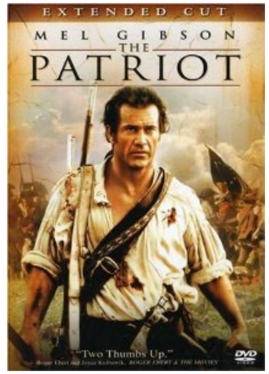
Target Movie "Like" Classification

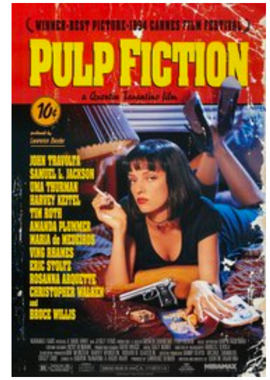


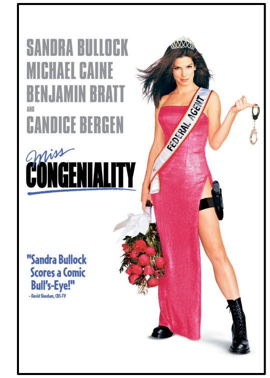


	Movie 1	Movie 2	…	Movie m	Output
User 1	1	0		1	1
User 2	1	1		0	0
			⋮		⋮
User n	0	0		1	1

Single Instance



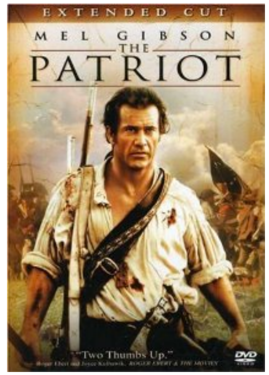


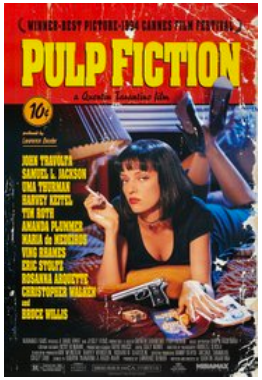


	Movie 1	Movie 2	…	Movie m	Output
User 1	1	0		1	1
User 2	1	1		0	0
			$\vdots$		$\vdots$
User n	0	0		1	1

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

Feature Vector

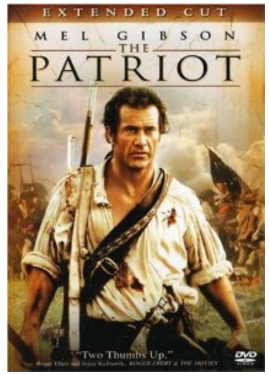
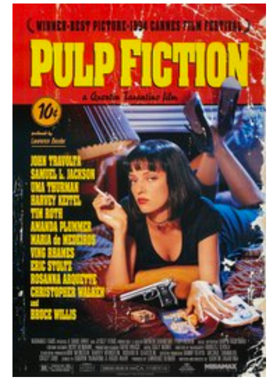




	Movie 1	Movie 2	...	Movie m	Output
User 1	1	0		1	1
User 2	1	1		0	0
⋮			⋮		⋮
User n	0	0		1	1

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

Output Value

	Movie 1	Movie 2	…	Movie m	Output
					
User 1	1	0		1	1
User 2	1	1		0	$\boxed{0}$
			⋮		⋮
User n	0	0		1	1

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

Logistic Regression:

Define a function that predicts $P(Y = 1)$ directly

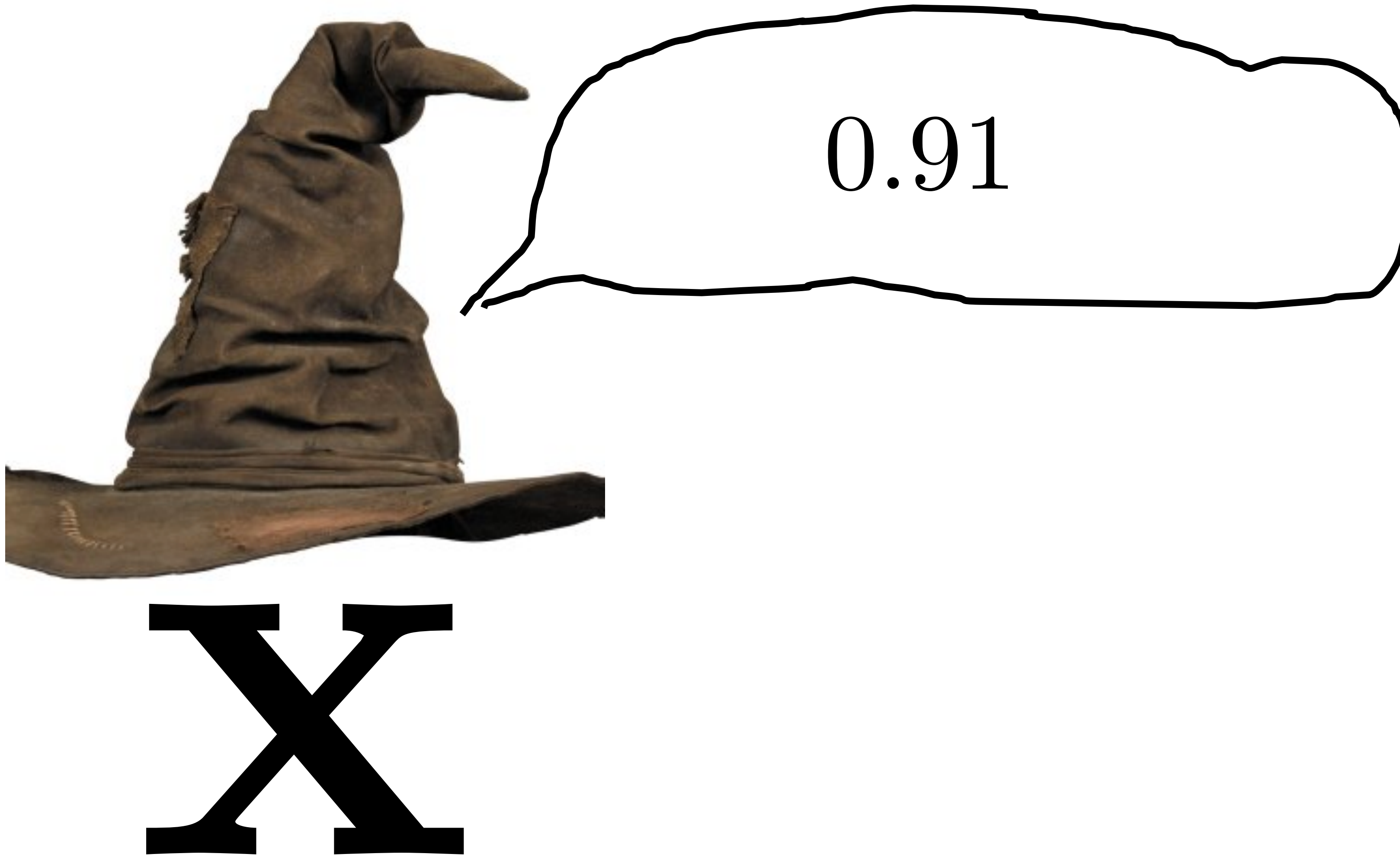
Logistic Regression is like the Harry Pottery Sorting Hat

$$[1, 1, 0, 0]$$

Logistic Regression is like the Harry Pottery Sorting Hat

$$[1, 1, 0, 0]$$

Logistic Regression is like the Harry Pottery Sorting Hat

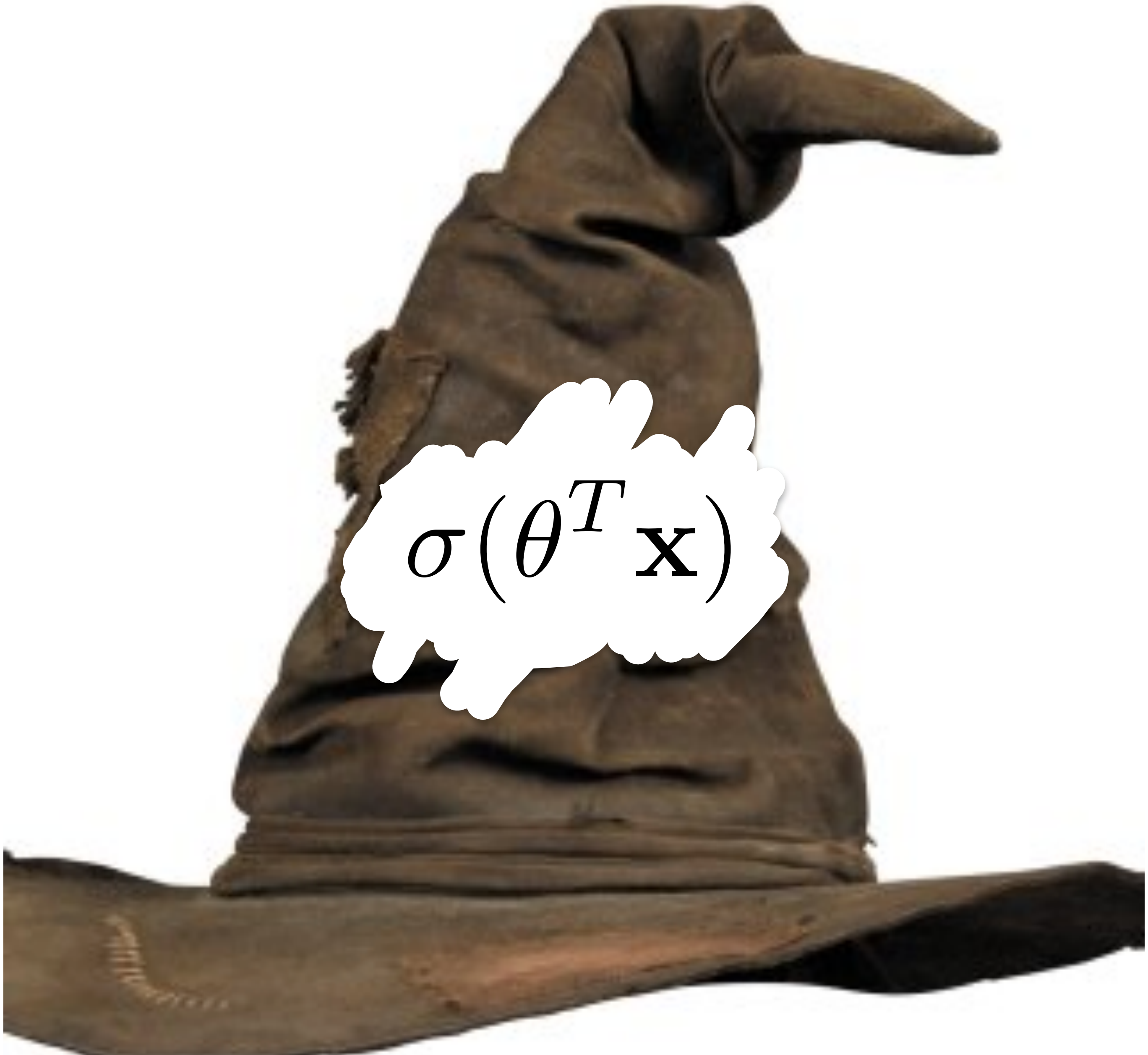


Logistic Regression is like the Harry Pottery Sorting Hat

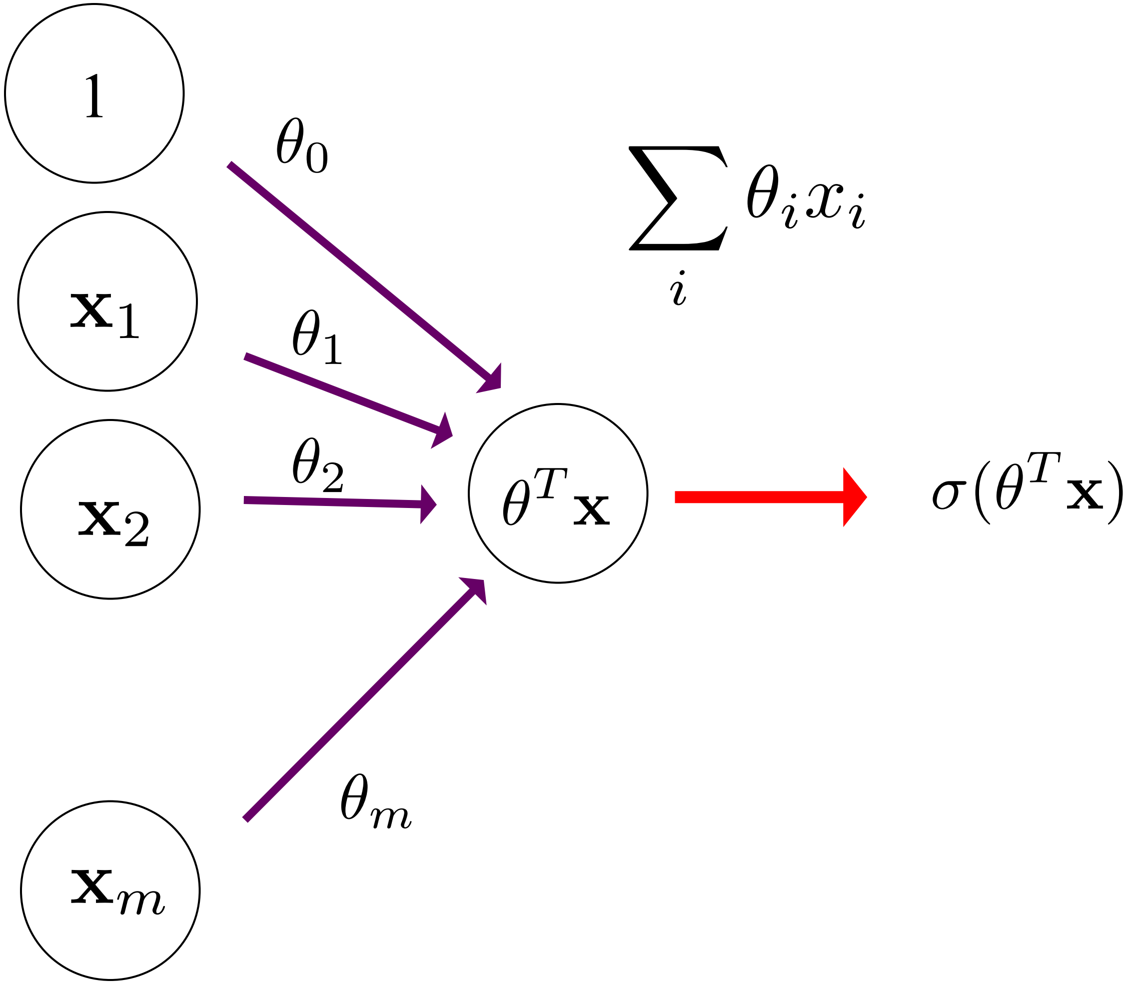
$$P(Y = 1) = 0.91$$

$$\mathbf{X}$$

Logistic Regression is like the Harry Pottery Sorting Hat

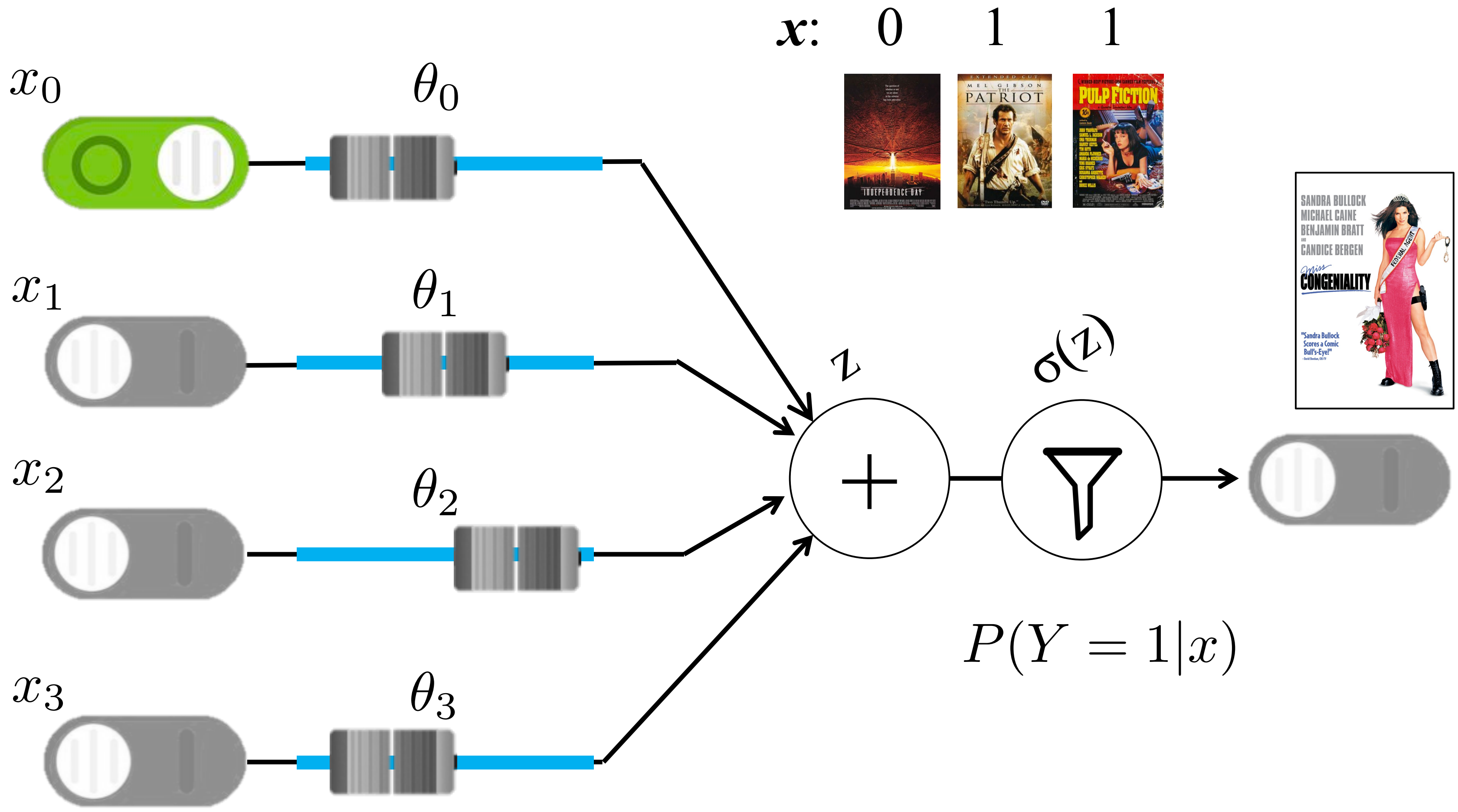


Logistic Regression



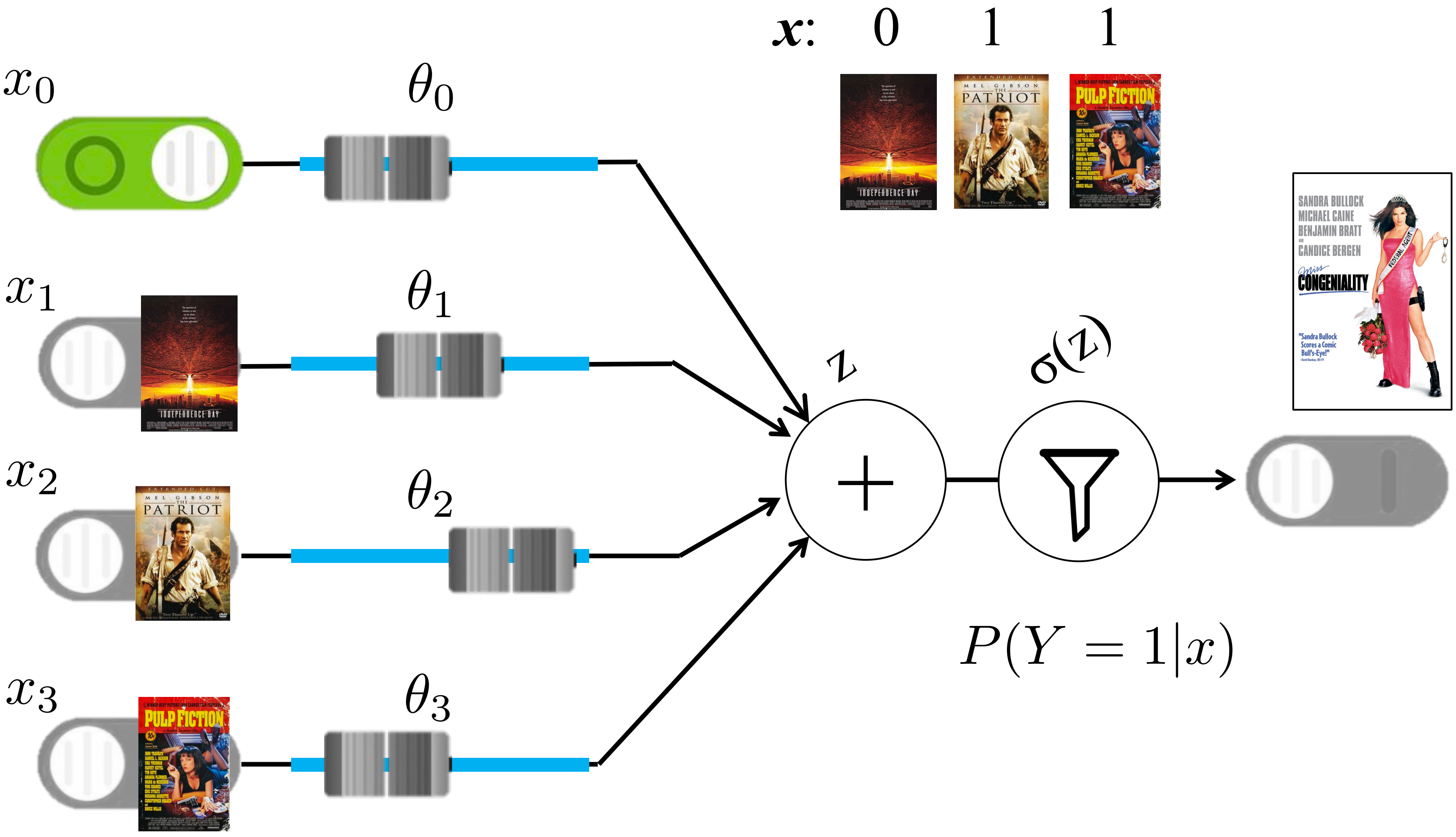
$$P(Y = 1 | X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



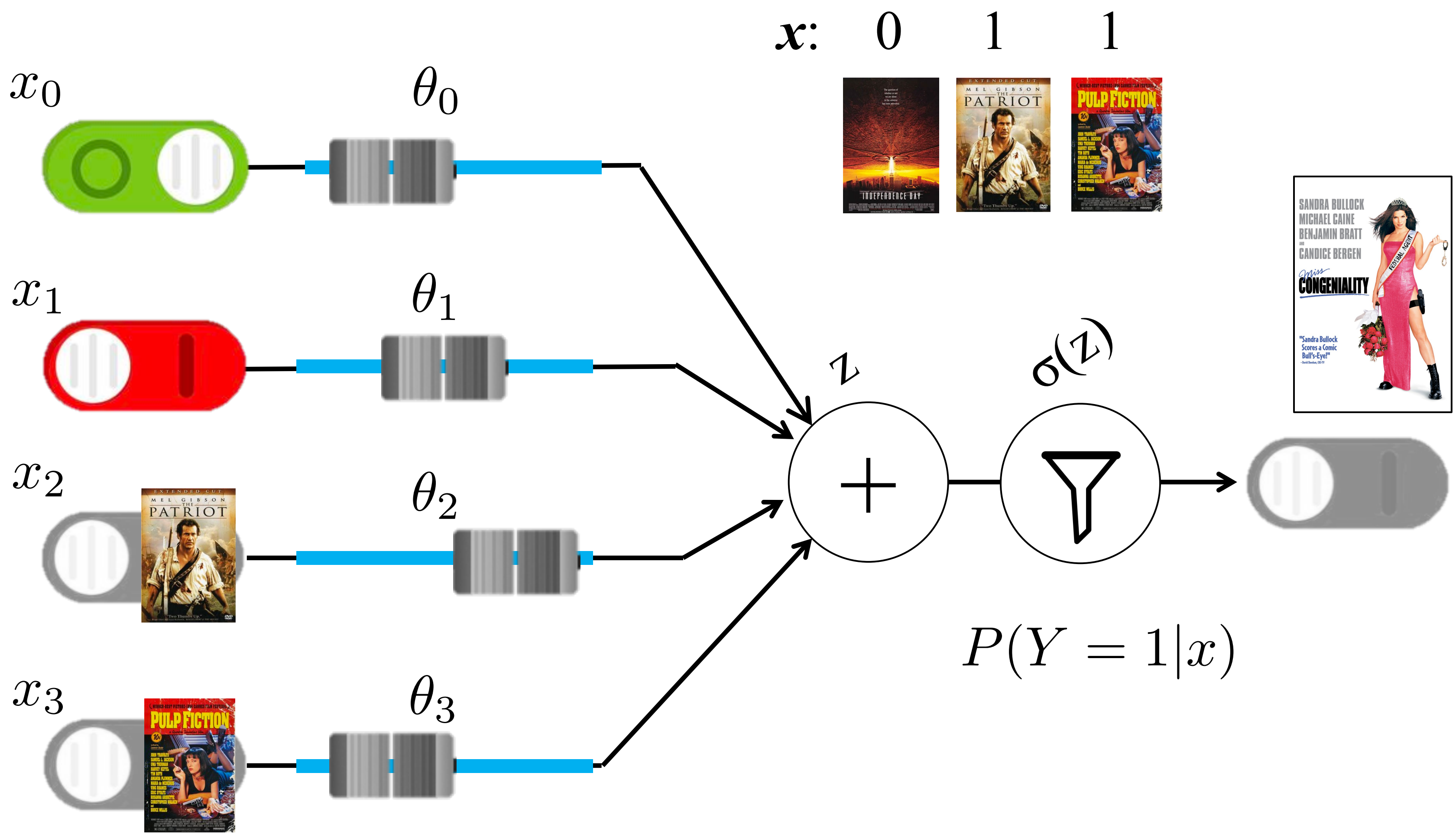
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



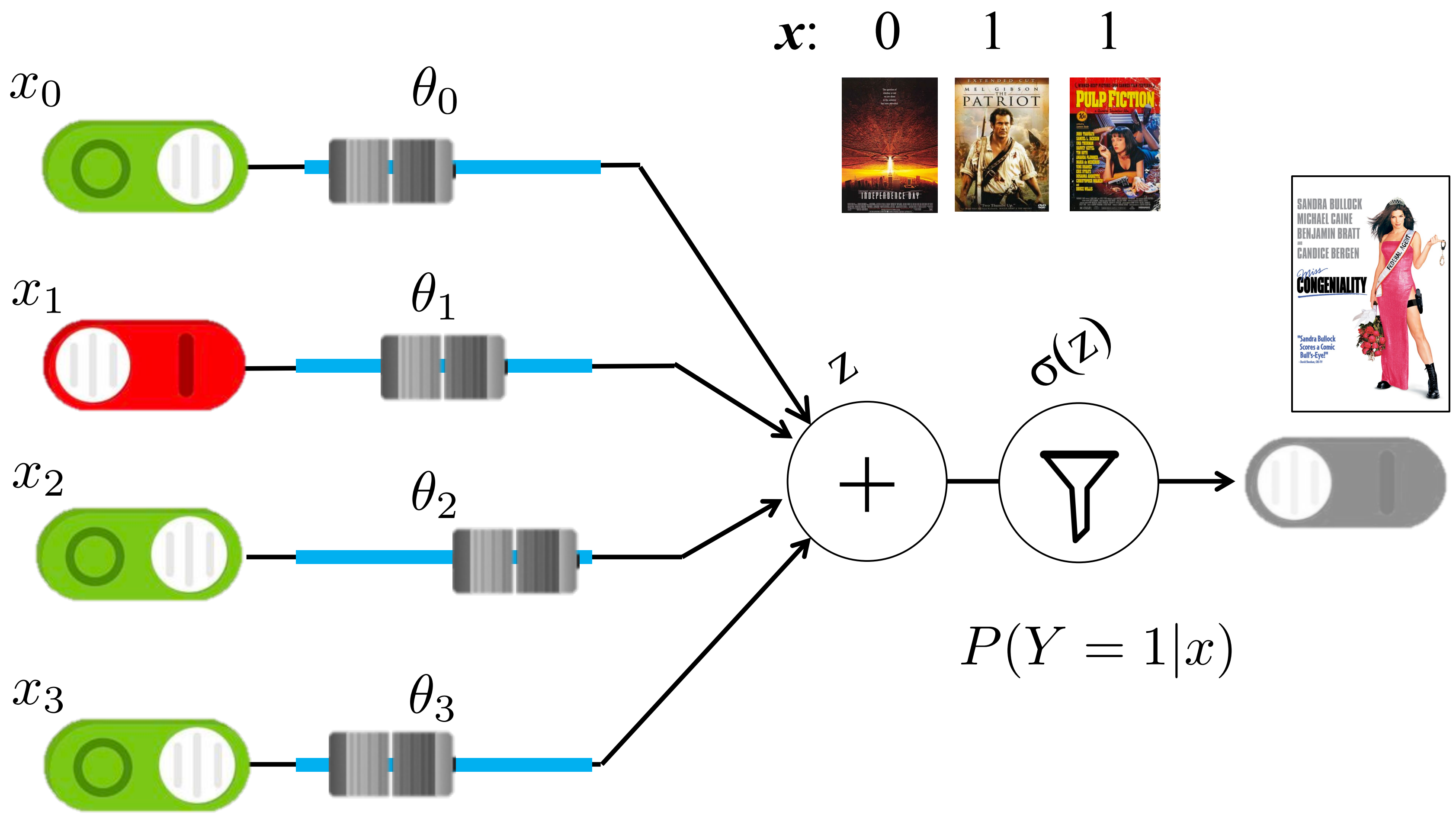
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



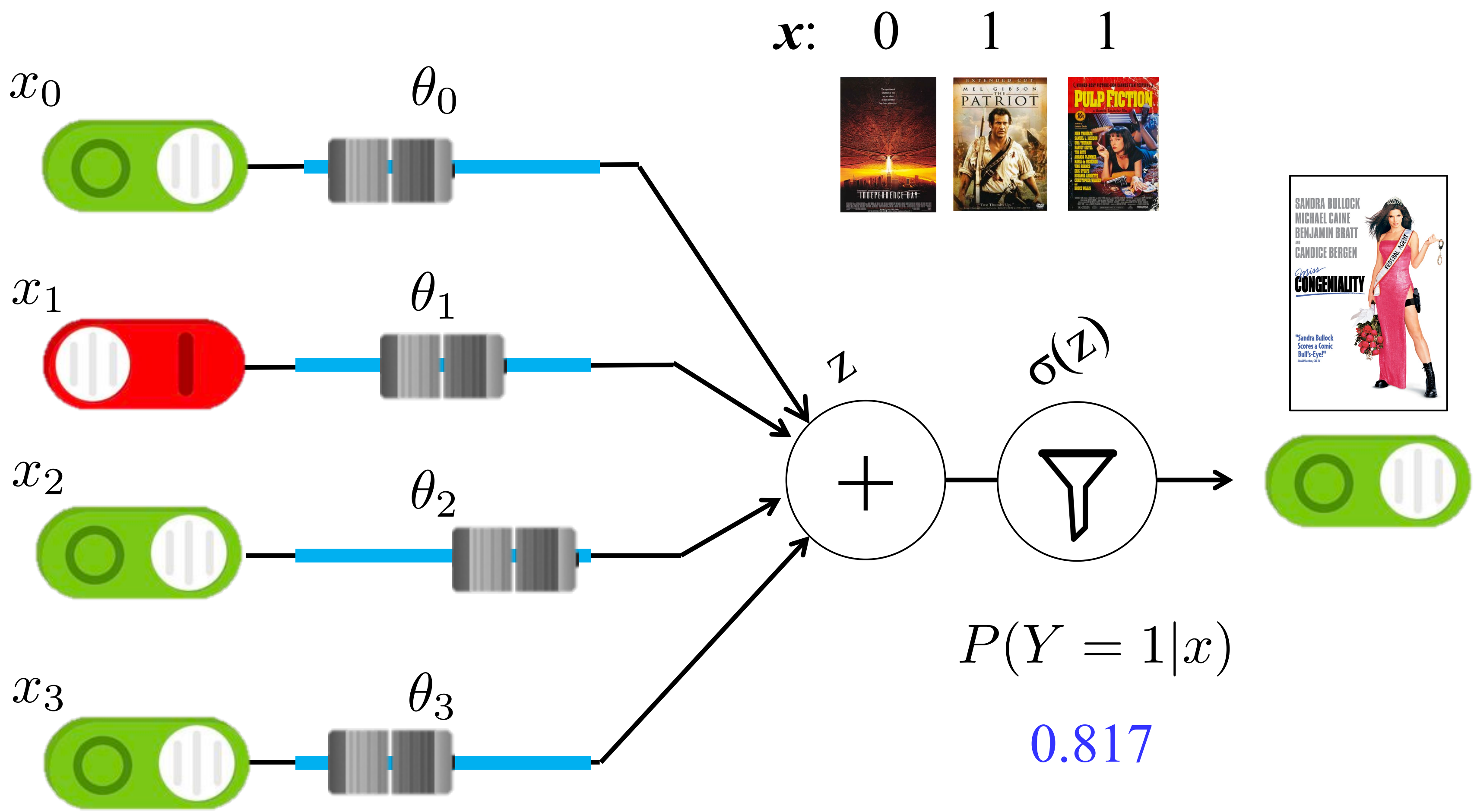
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



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

Logistic Regression

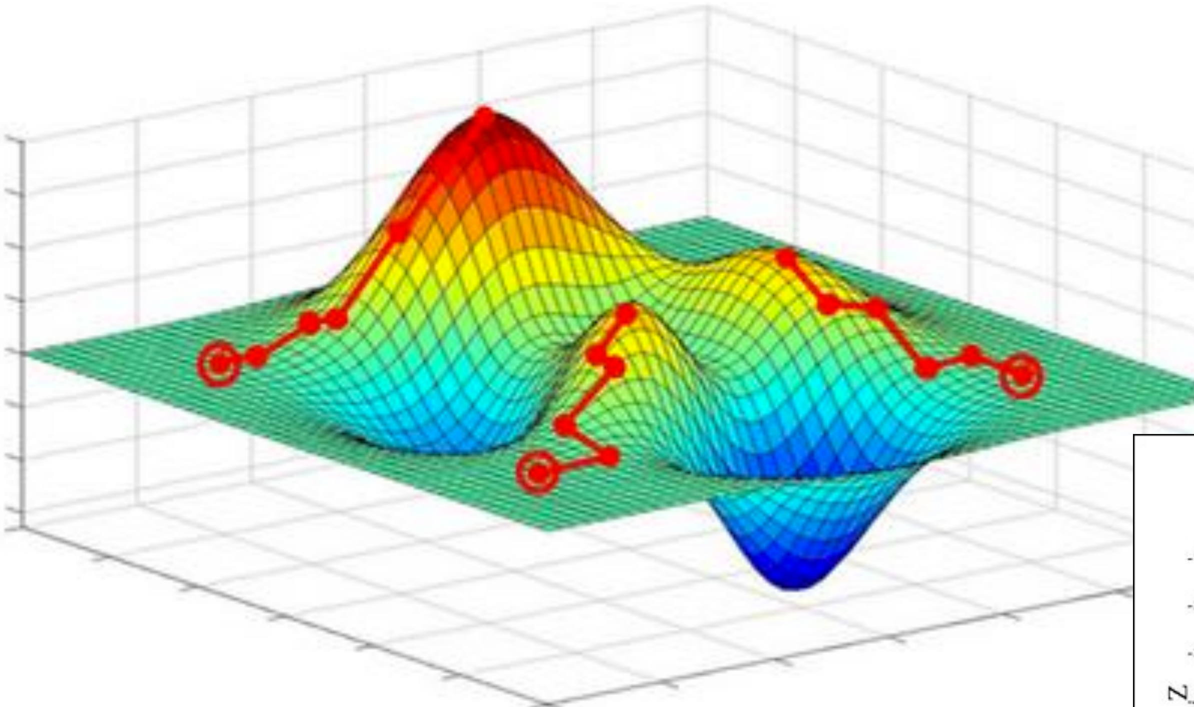


$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

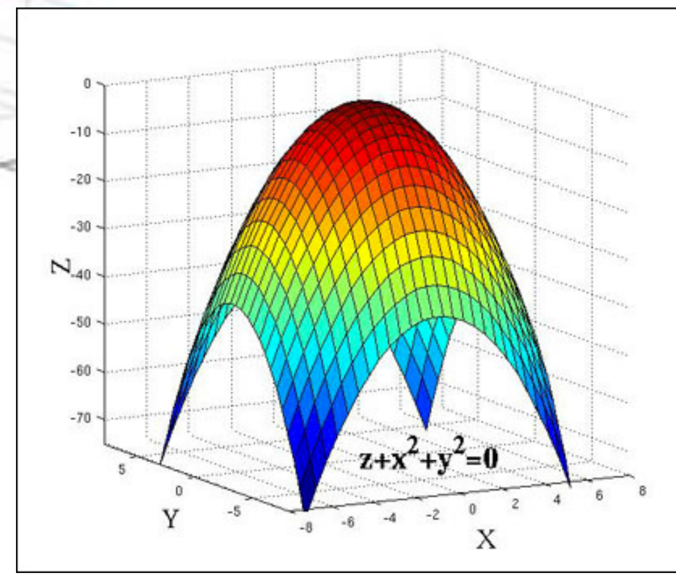
Logistic regression gets its ***intelligence*** from its thetas (aka its parameters)

The hard part is learning the thetas

Gradient Ascent



Logistic regression
LL function is
convex



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

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

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

3 Get derivative of log likelihood with respect to thetas

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

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

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

3. Get derivative of log likelihood with respect to thetas

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

Sanity Check

3 Get partial derivative of log likelihood with respect to each theta

Why?

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:

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

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

How did we get that gradient?

That's so derivative…

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

Where

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

Big Idea #1: Chain Rule

Woah Mr Blanton, you were right.
Chain rule is useful!

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

Big Idea #2: Sigmoid Derivative

True fact about sigmoid functions

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

Big Idea #3: Gradient of Sum

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

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

Sigmoid is like a Ski Hill

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

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

True fact about sigmoid functions

$$z = \theta^T \mathbf{x} \qquad \therefore \hat{y} = \sigma(z)$$

First, define z

$$\frac{\partial \hat{y}}{\partial \theta_j} = \frac{\partial}{\partial \theta_j}\sigma(z) = \frac{\partial}{\partial z}\sigma(z) \cdot \frac{\partial z}{\partial \theta_j}$$

Chain rule!

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

Derivative of sigmoid

$$= \sigma(z)[1 - \sigma(z)] \cdot \mathbf{x}_j$$

Derivative of z

$$= \hat{y}[1 - \hat{y}] \cdot \mathbf{x}_j$$

Plug in y hat

Write on board

We are 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

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

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

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

More chain rule!

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

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

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

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

Derivative of sum...

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

$$= \sum_{i=1}^{n} [y^{(i)} - \hat{y}^{(i)}] x_j^{(i)}$$

See last slide

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

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

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

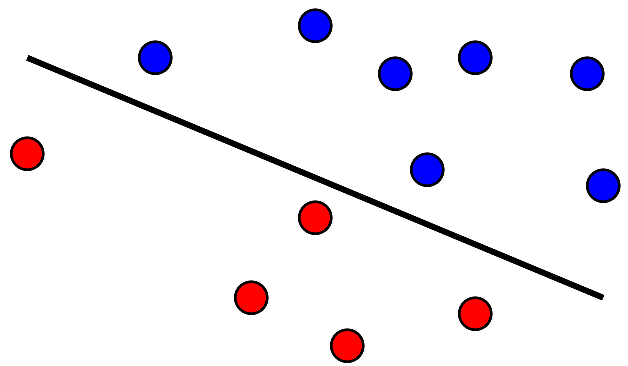
3 Get derivative of log probability with respect to thetas

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

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



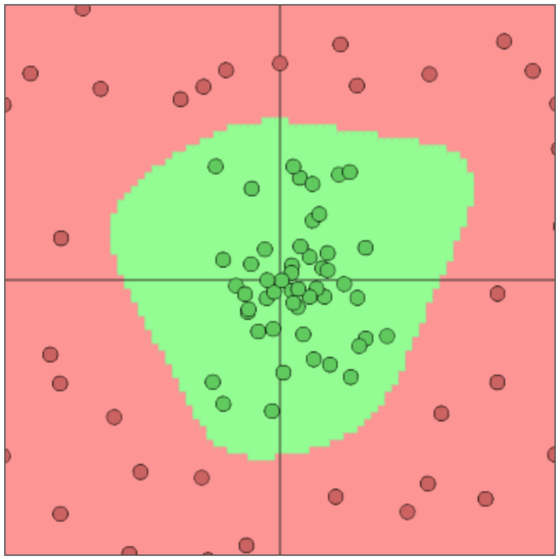
$$\theta^T \mathbf{x} = 0$$

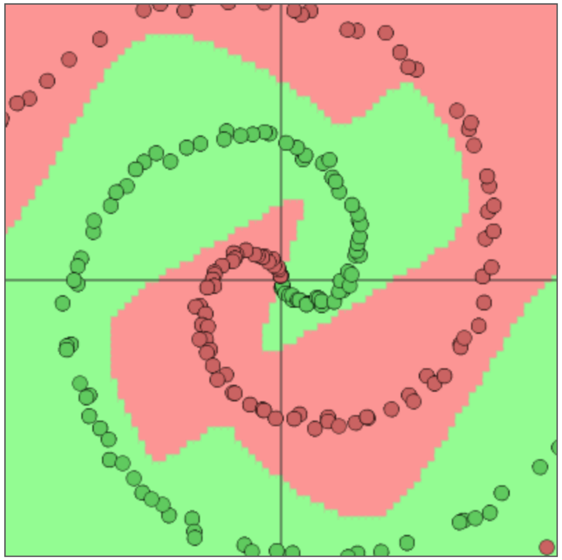
$$\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_m x_m = 0$$

- 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 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

Choosing an Algorithm?

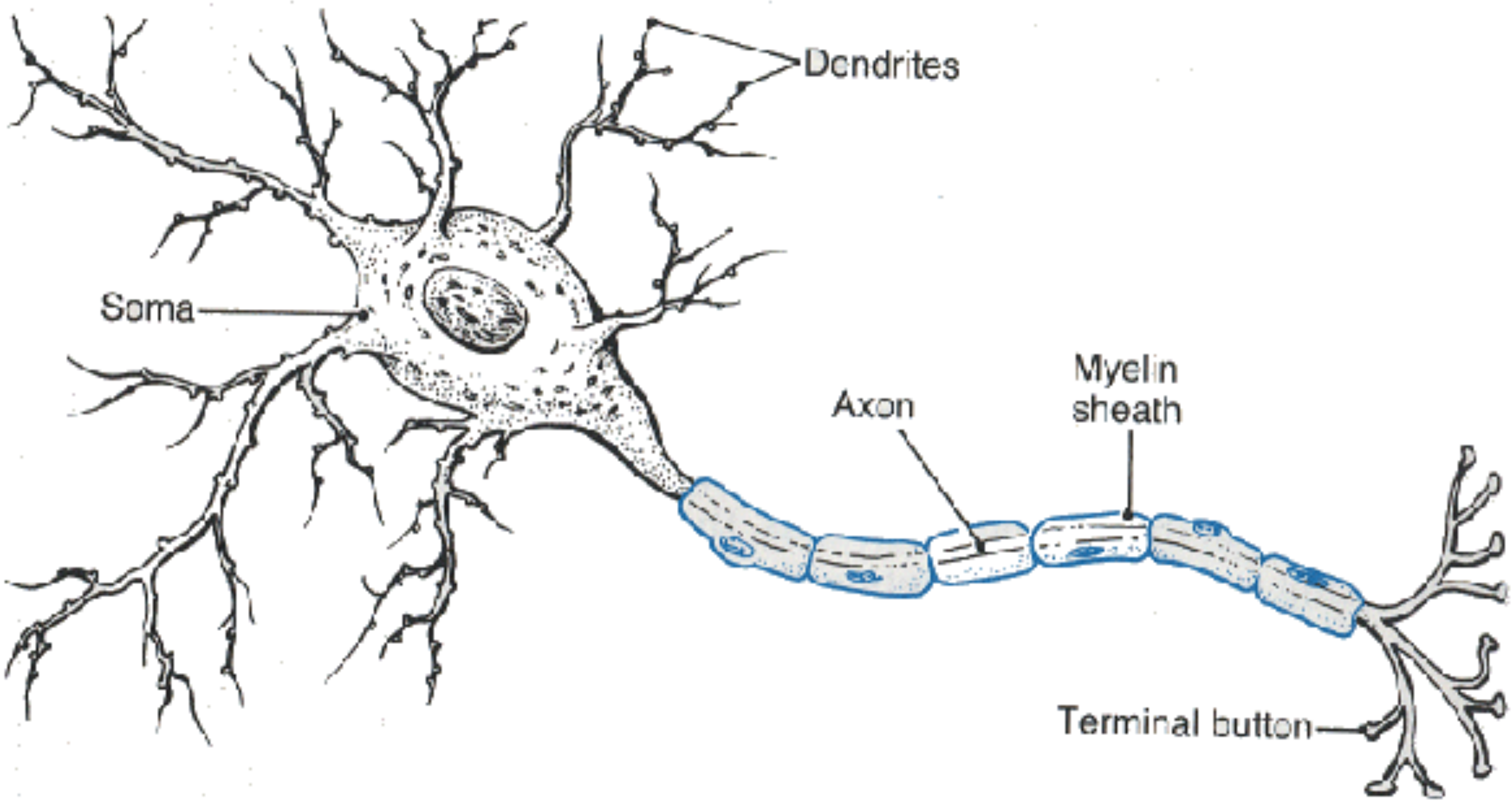
- Many trade-offs in choosing learning algorithm
  - Continuous input variables
    - Logistic Regression easily deals with continuous inputs
    - Naive Bayes needs to use some parametric form for continuous inputs (e.g., Gaussian) or "discretize" continuous values into ranges (e.g., temperature in 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.

Next up: Deep Learning!

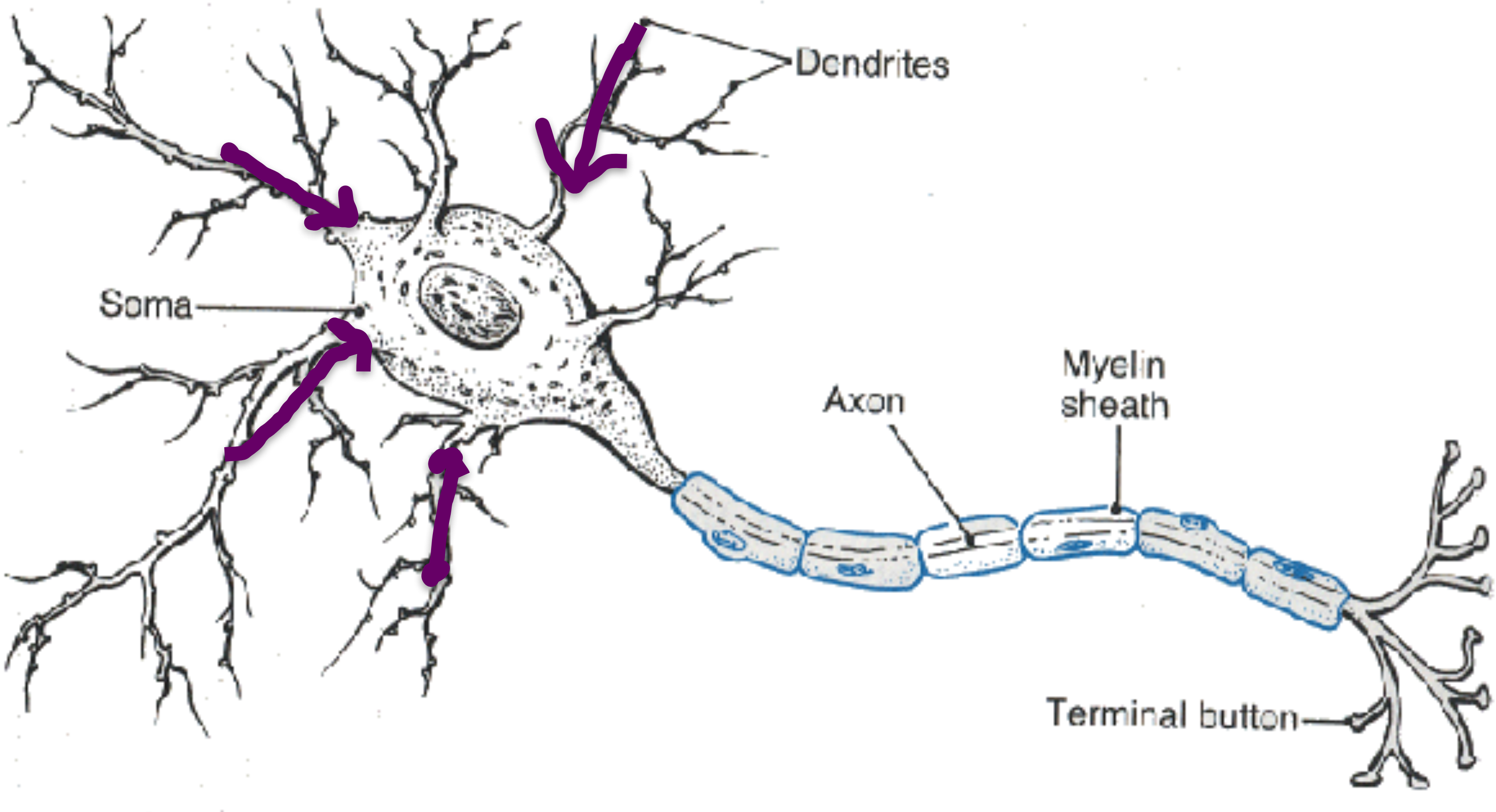
20 second pedagogical pause

Summarize what we have learned

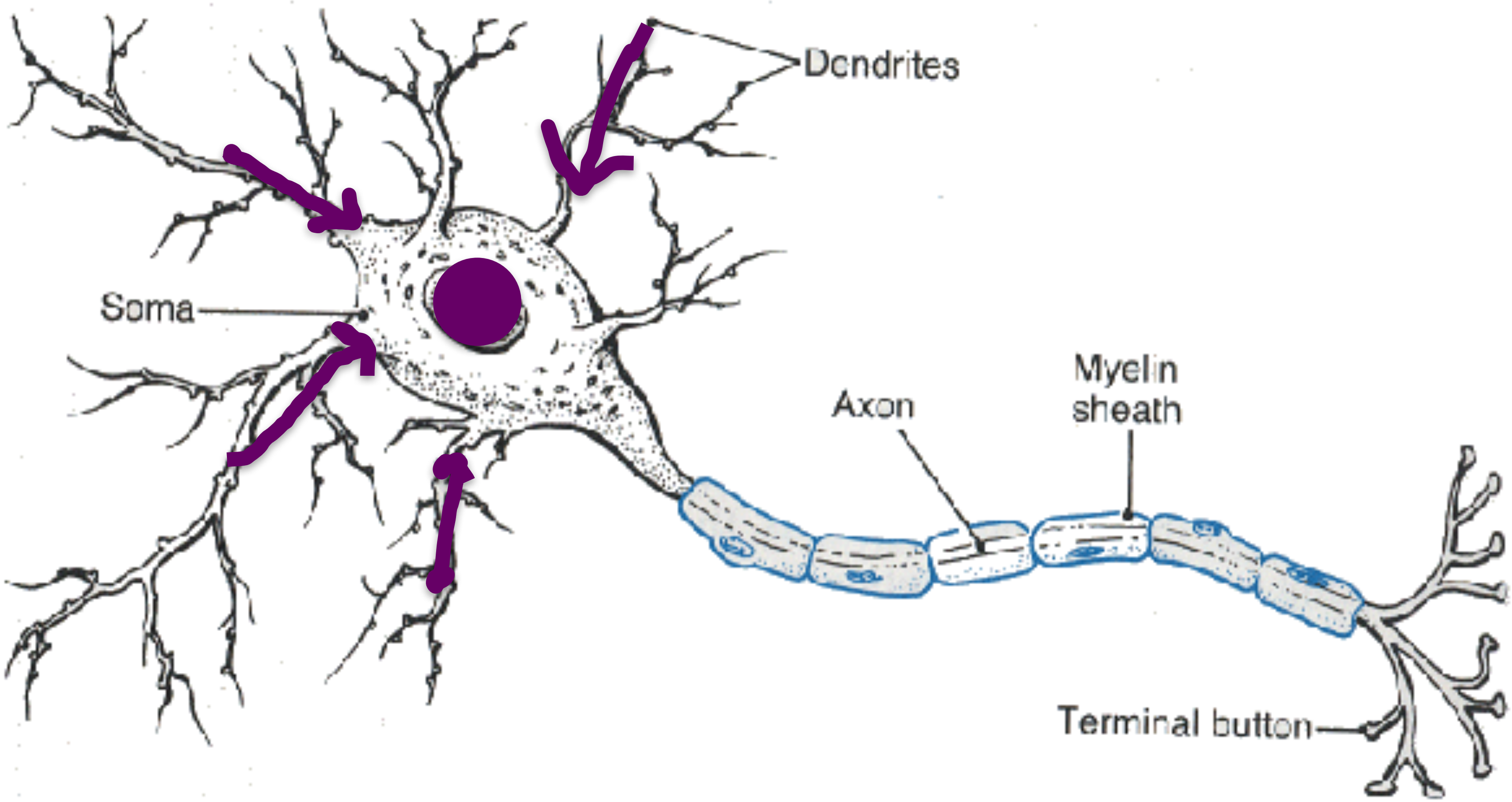
Neuron



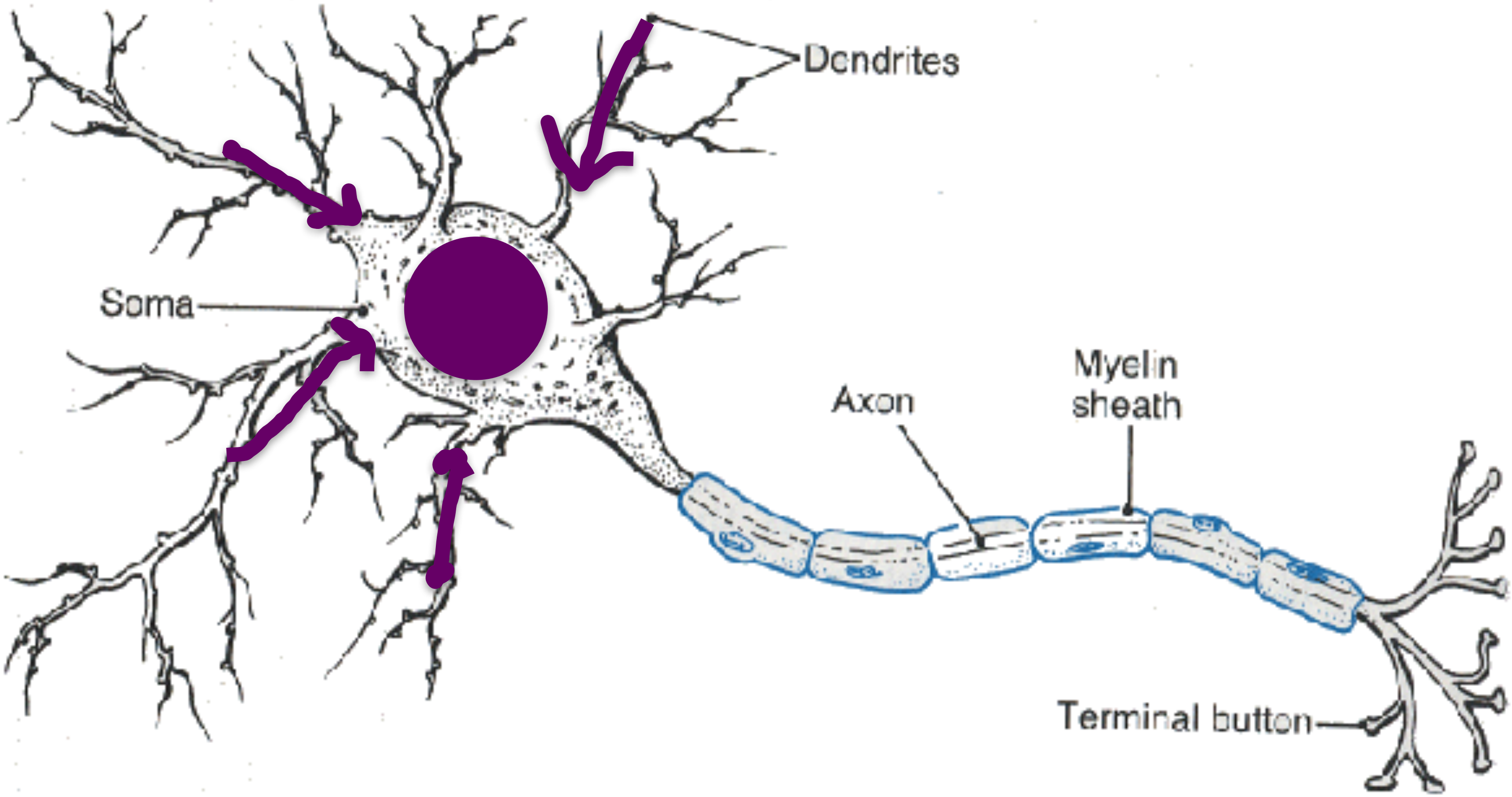
Neuron



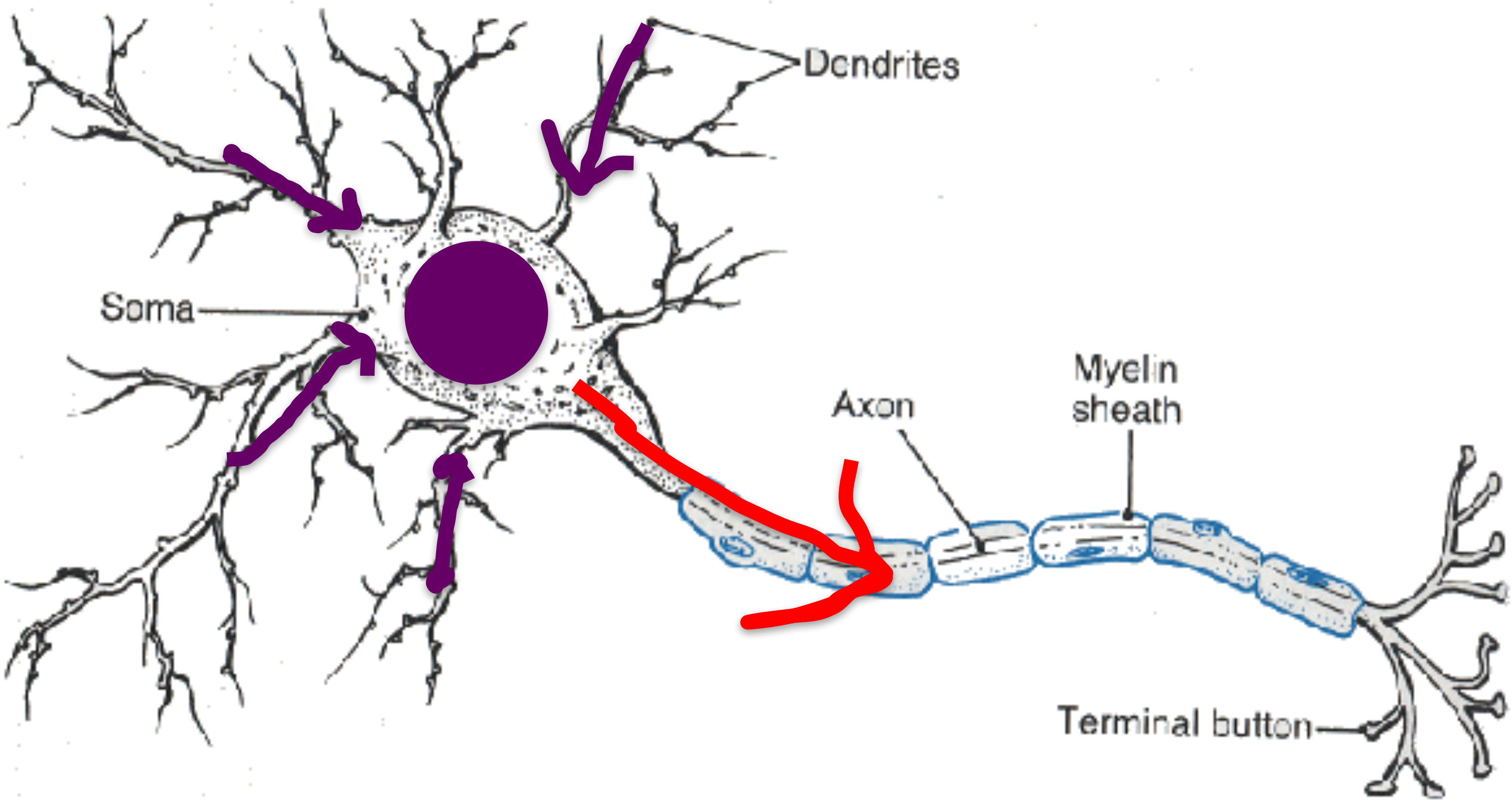
Neuron



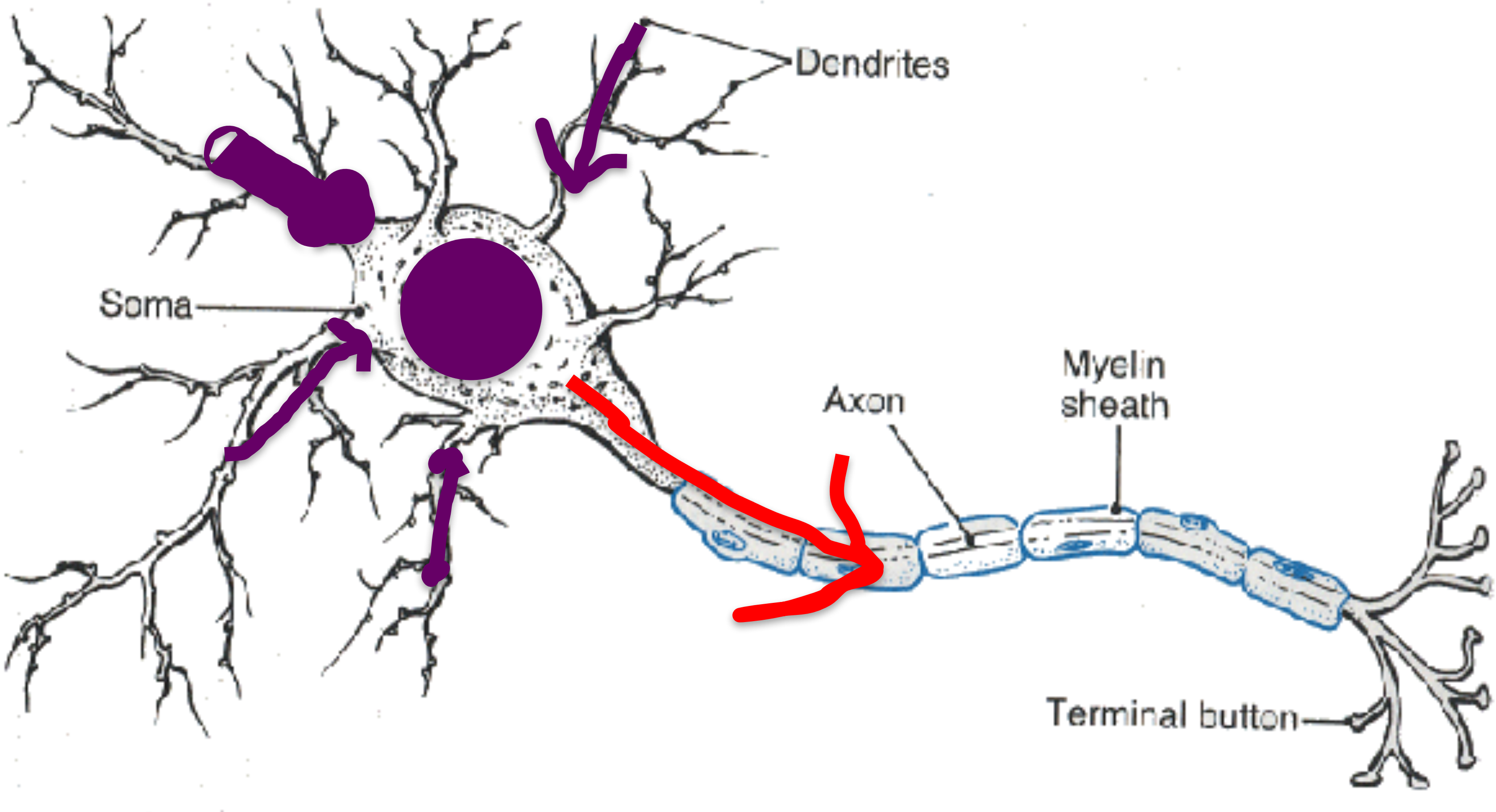
Neuron



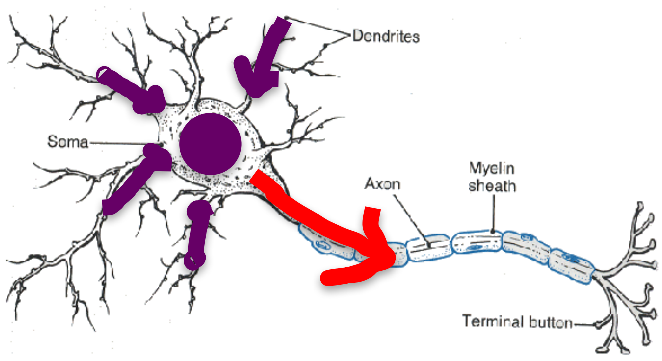
Neuron

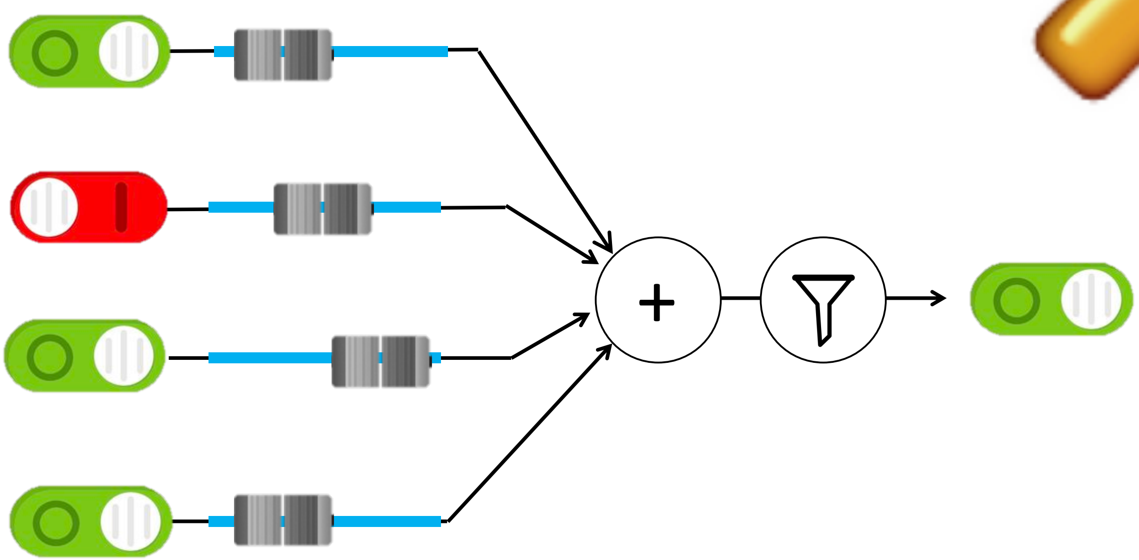


Some inputs are more important



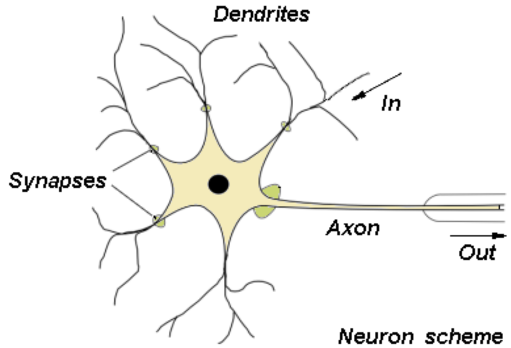
Artificial Neurons

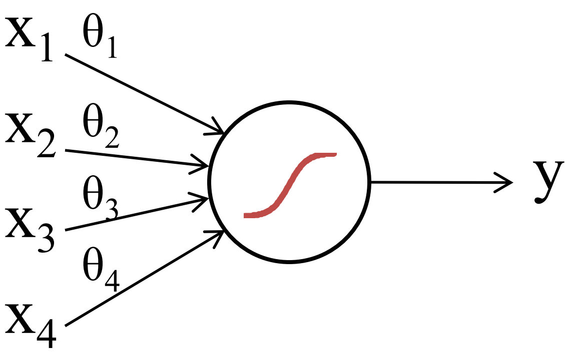




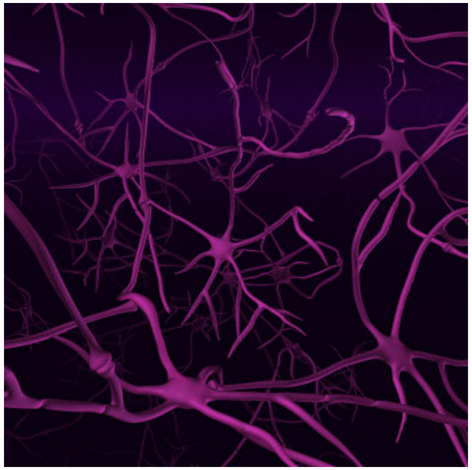
Biological Basis for Neural Networks

- A neuron

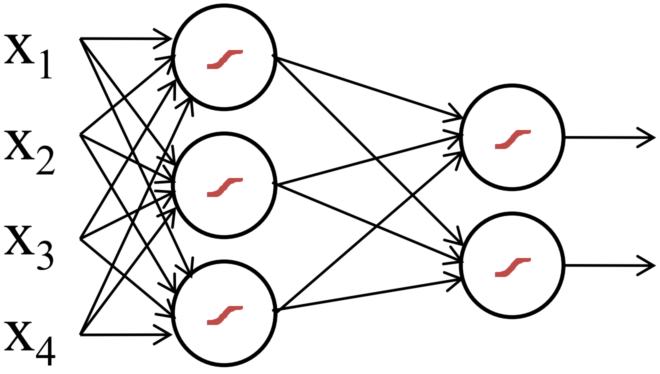




- Your brain



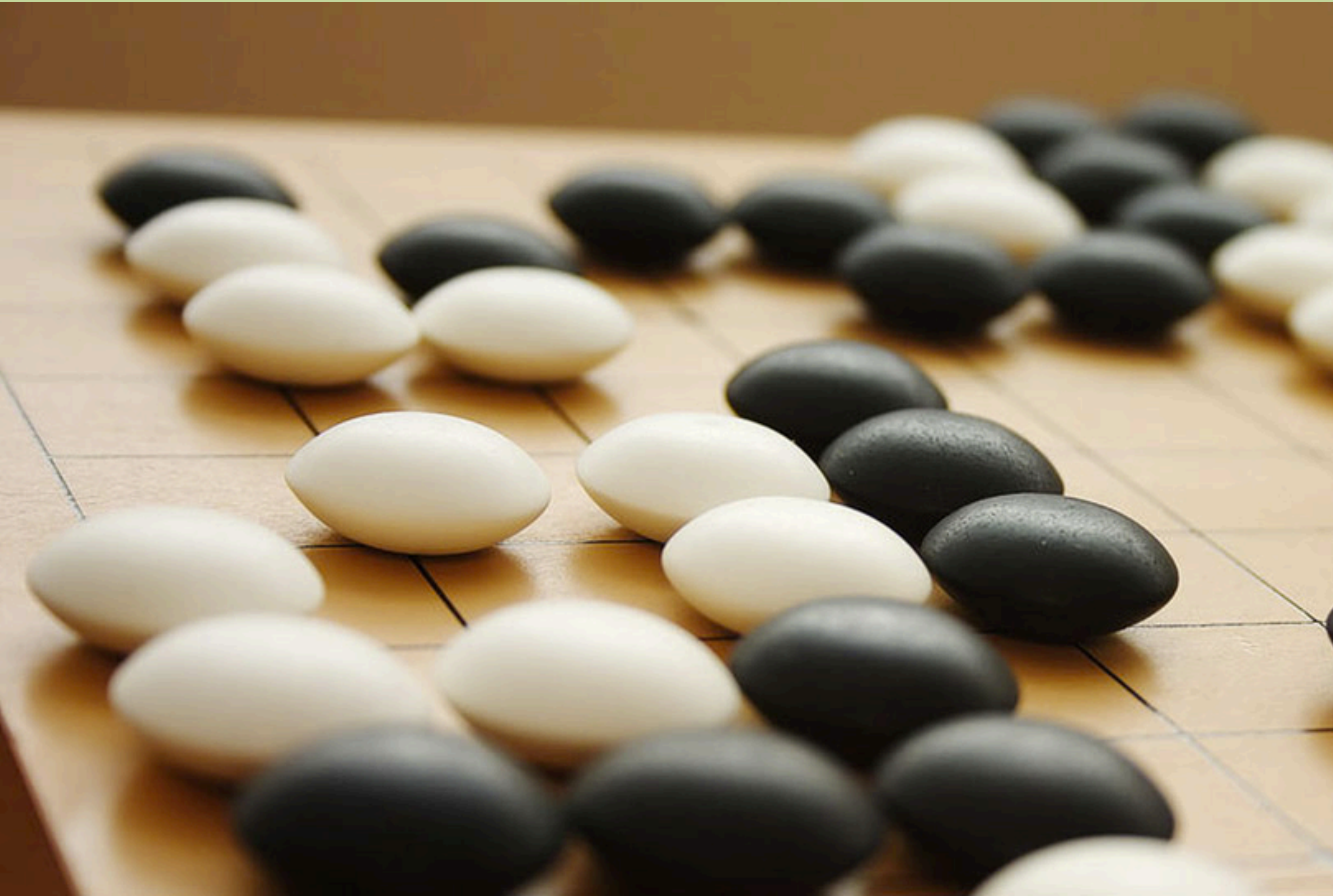
Actually, it's probably someone else's brain



(aka Neural Networks)

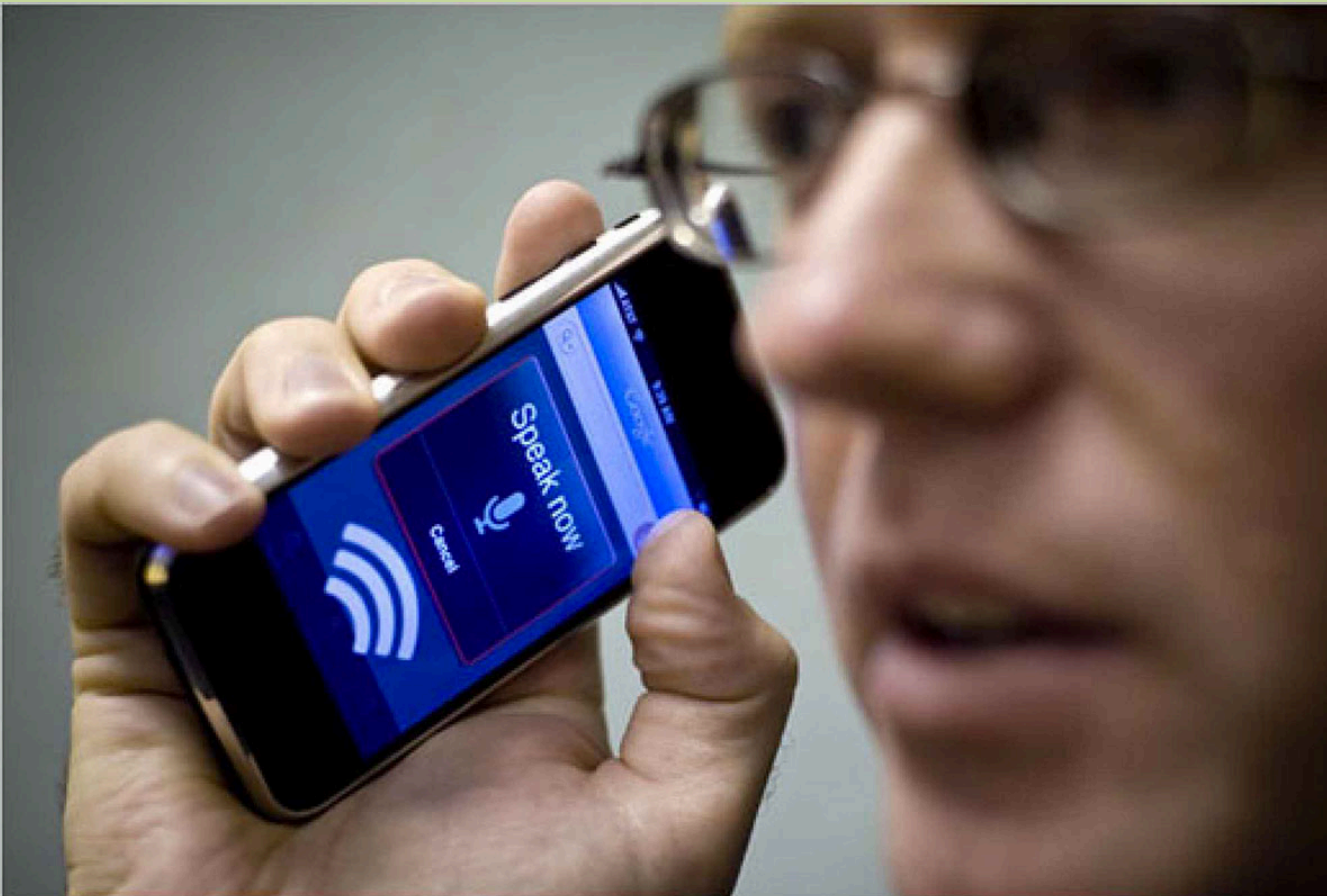
Deep learning is (at its core) many logistic regression pieces stacked on top of each other.

Alpha GO



Computer Vision

Revolution in AI

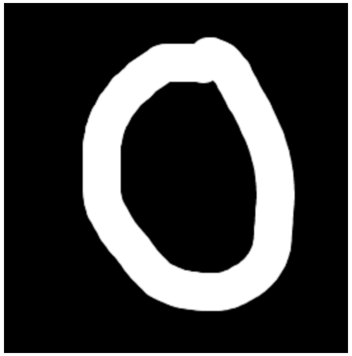


Computers Making Art

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

Digit Recognition Example

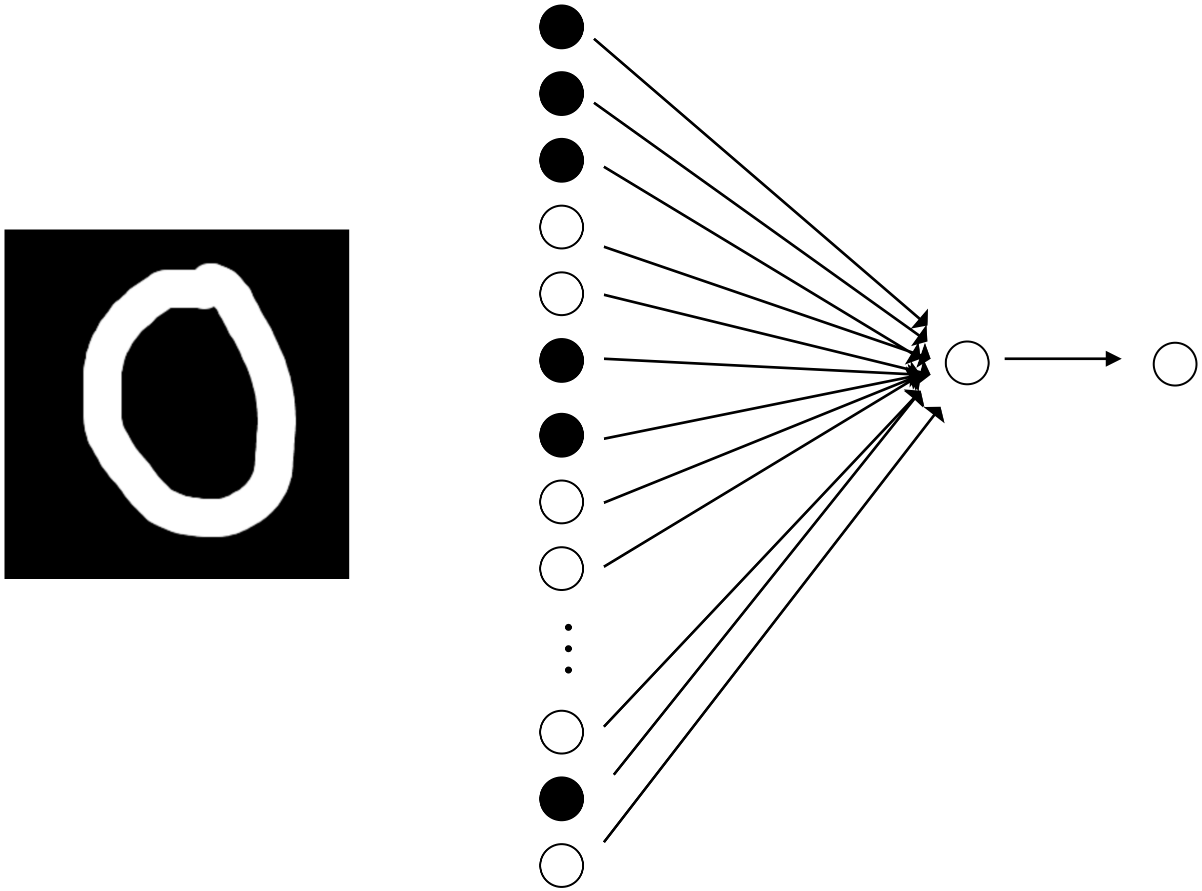
Let's make feature vectors from pictures of numbers



$$\mathbf{x}^{(i)} = [0,0,0,0,\ldots,1,0,0,1,\ldots 0,0,1,0]$$
$$y^{(i)} = 0$$

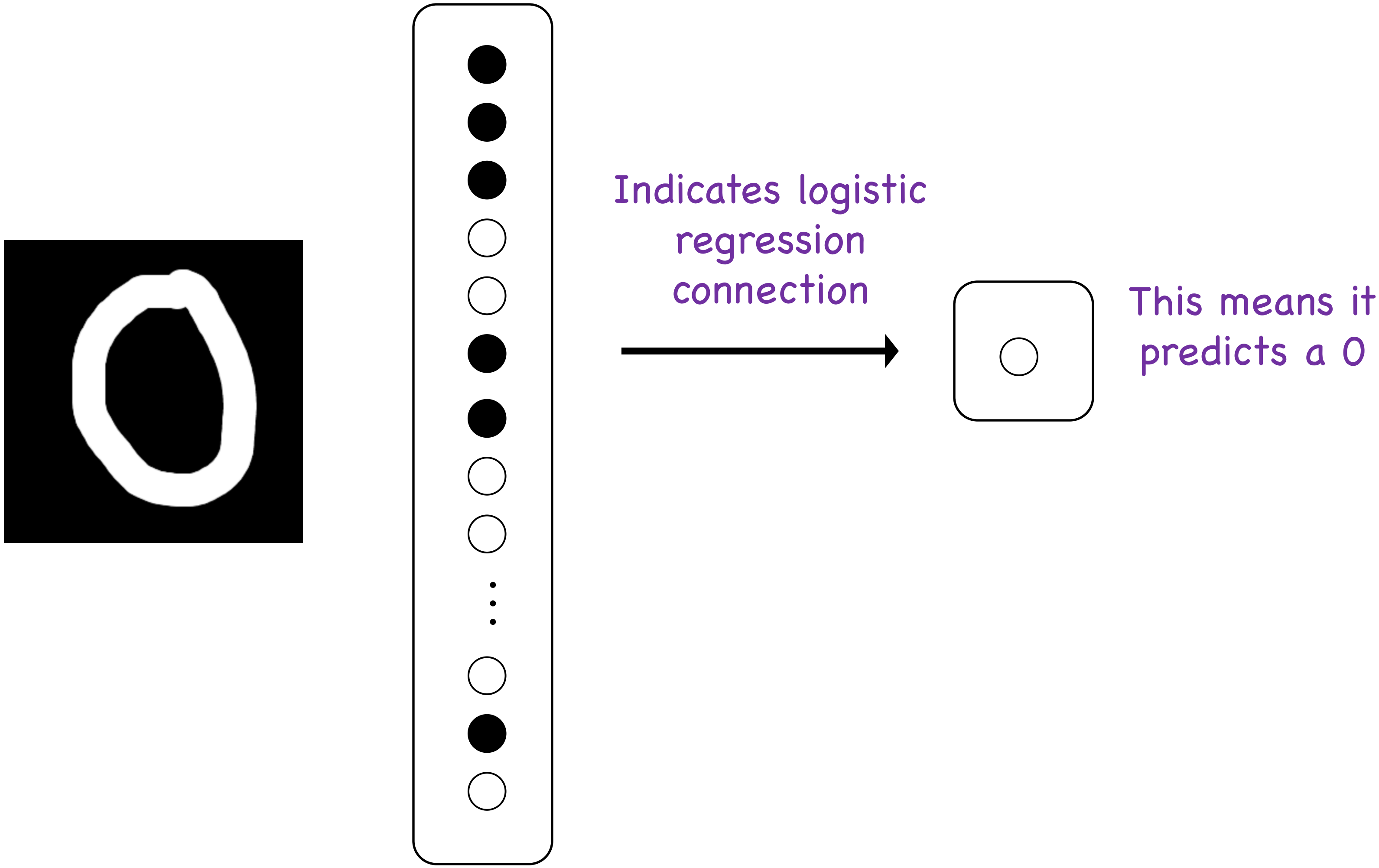
$$\mathbf{x}^{(i)} = [0,0,1,1,\ldots,0,1,1,0,\ldots 0,1,0,0]$$
$$y^{(i)} = 1$$

Logistic Regression

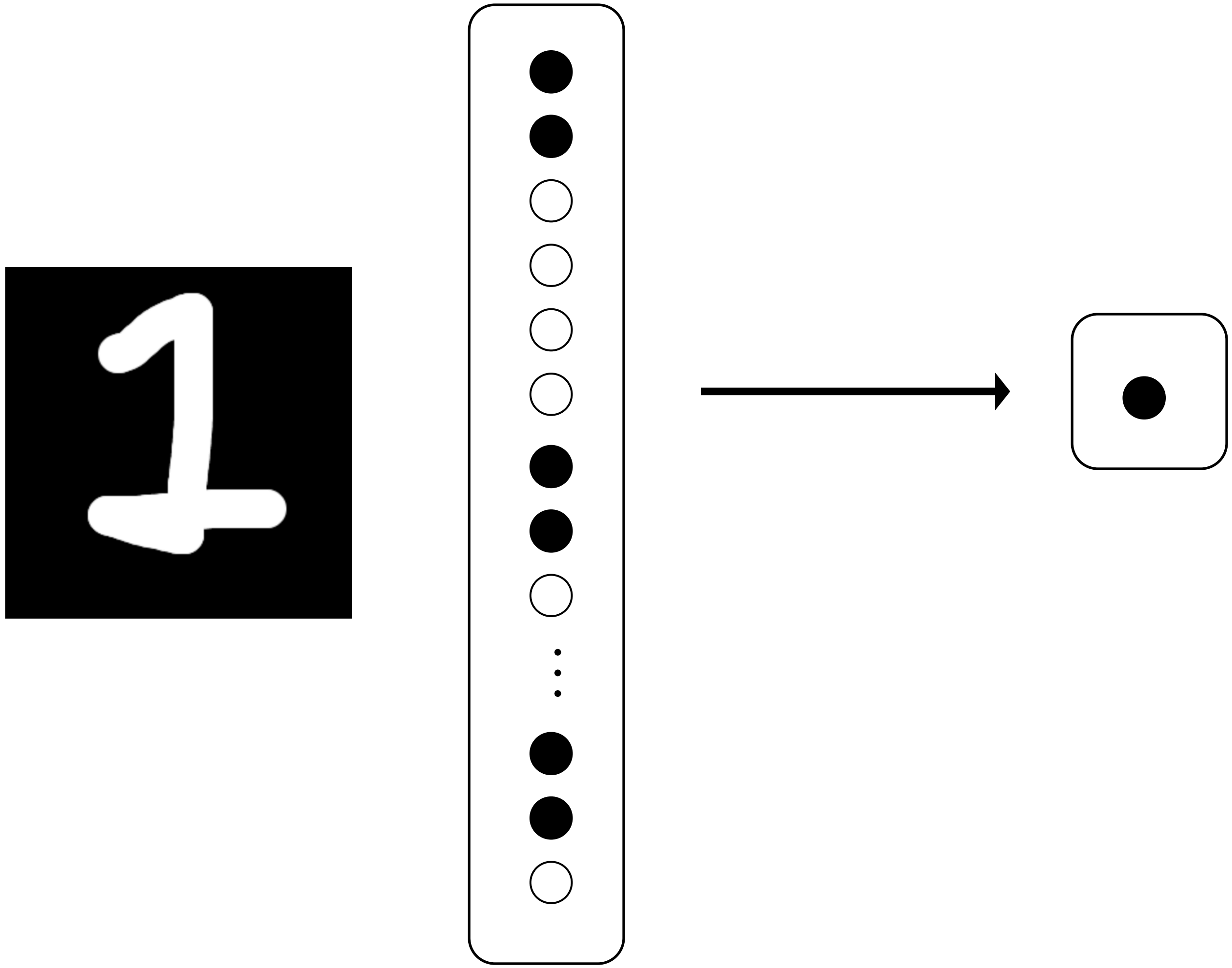


This means it predicts a 0

Logistic Regression

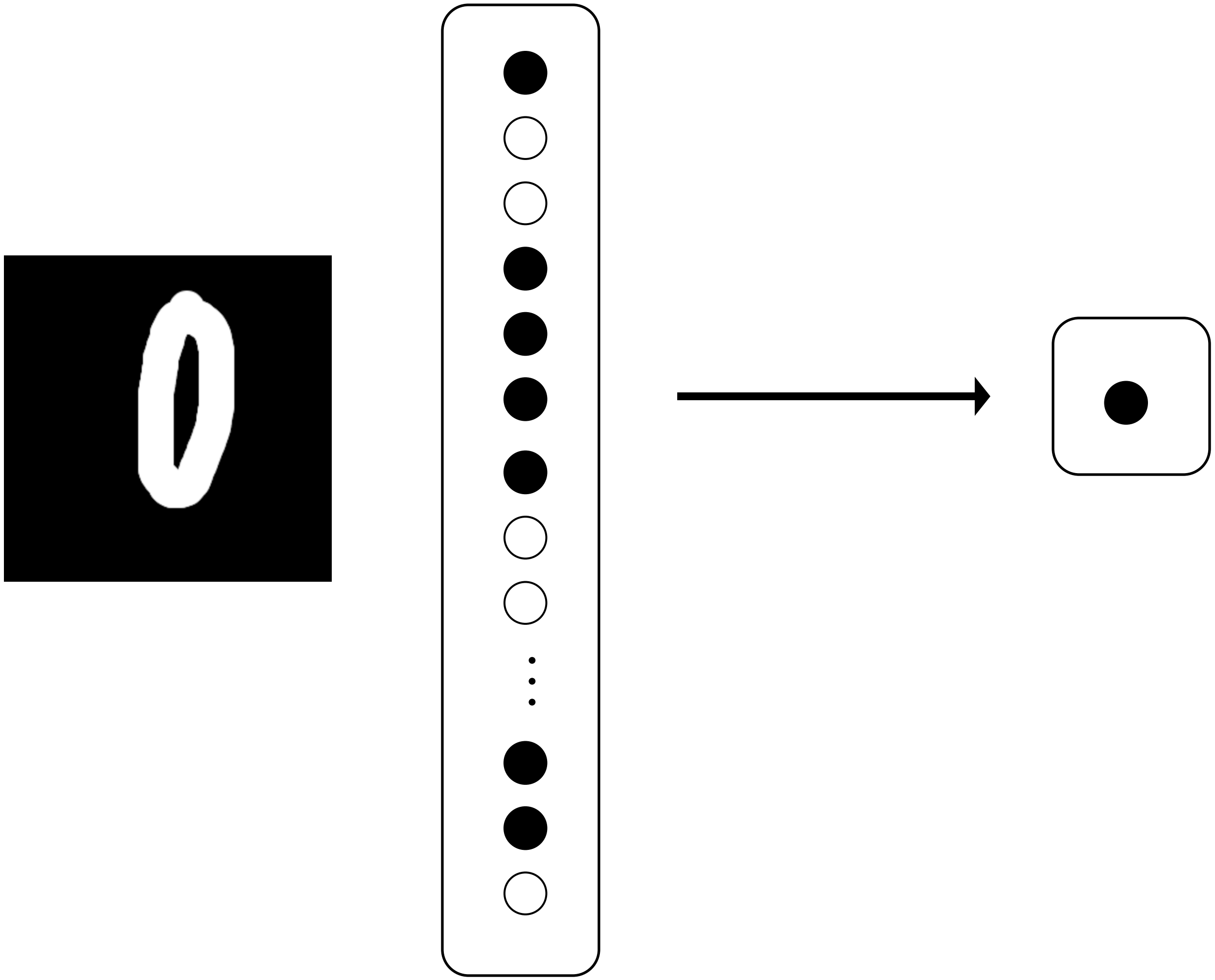


Logistic Regression



This means it predicts a 1

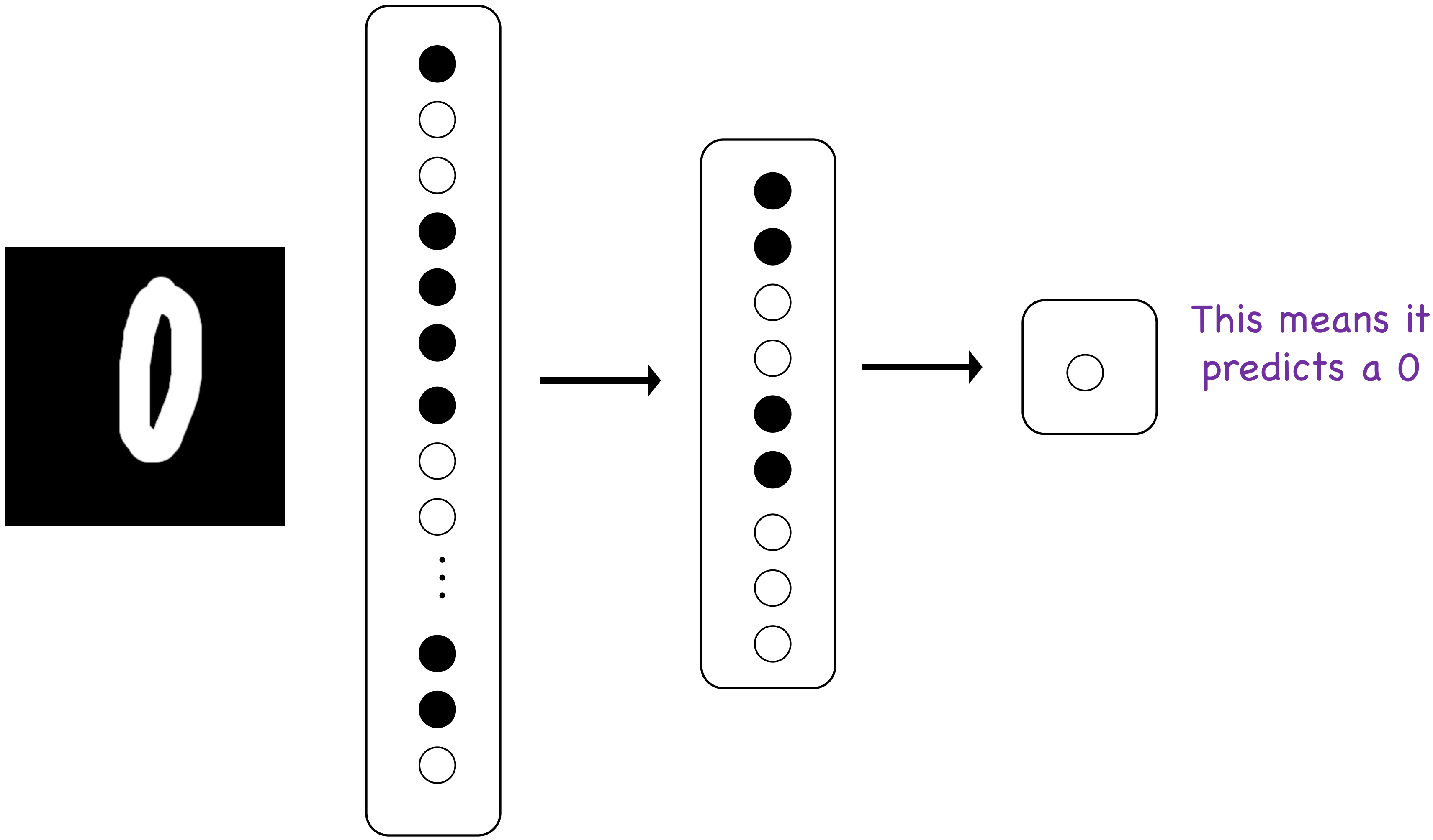
Not So Good



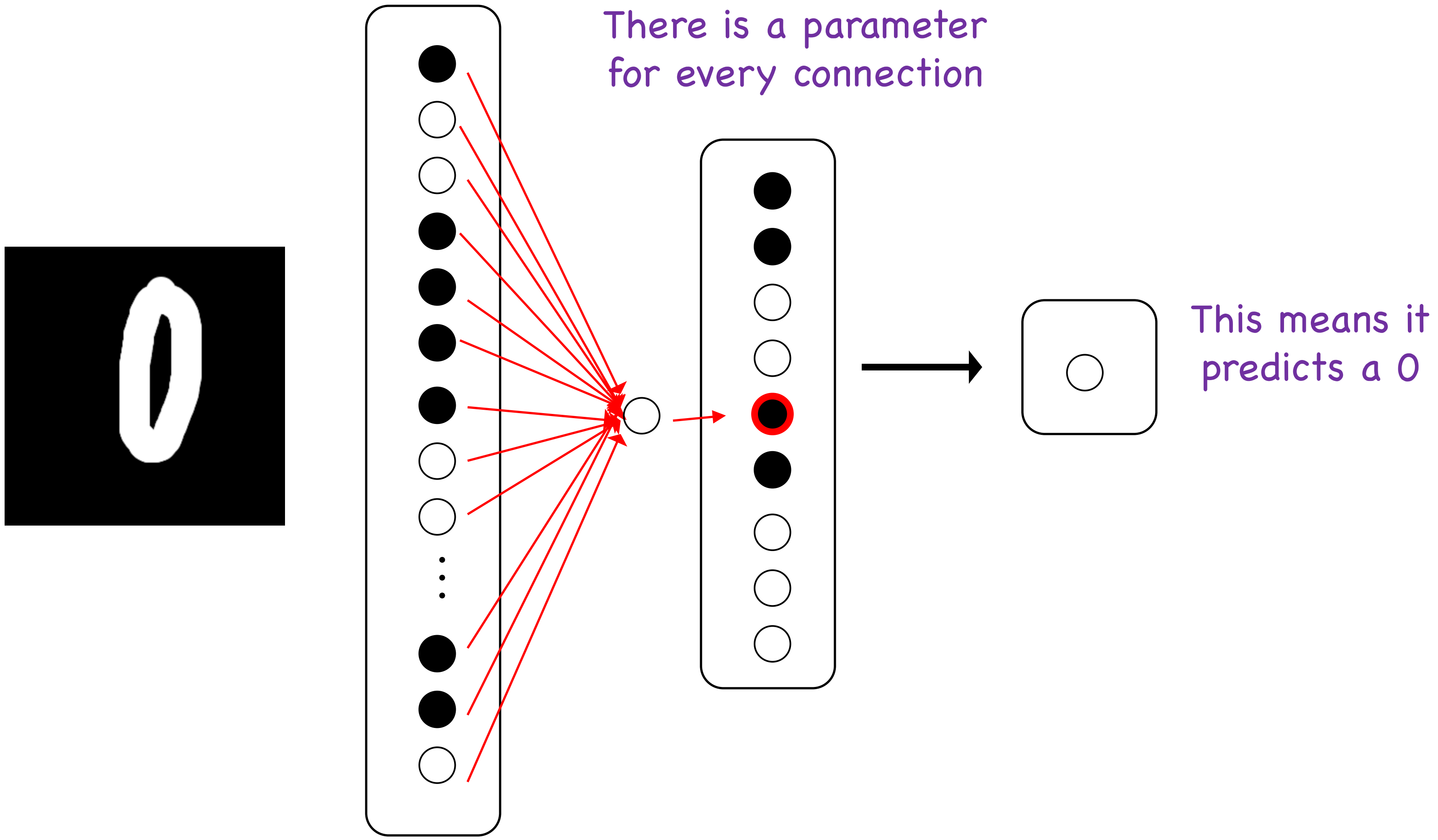
This means it
predicts a 1

What can we do?

We Can Put Neurons Together

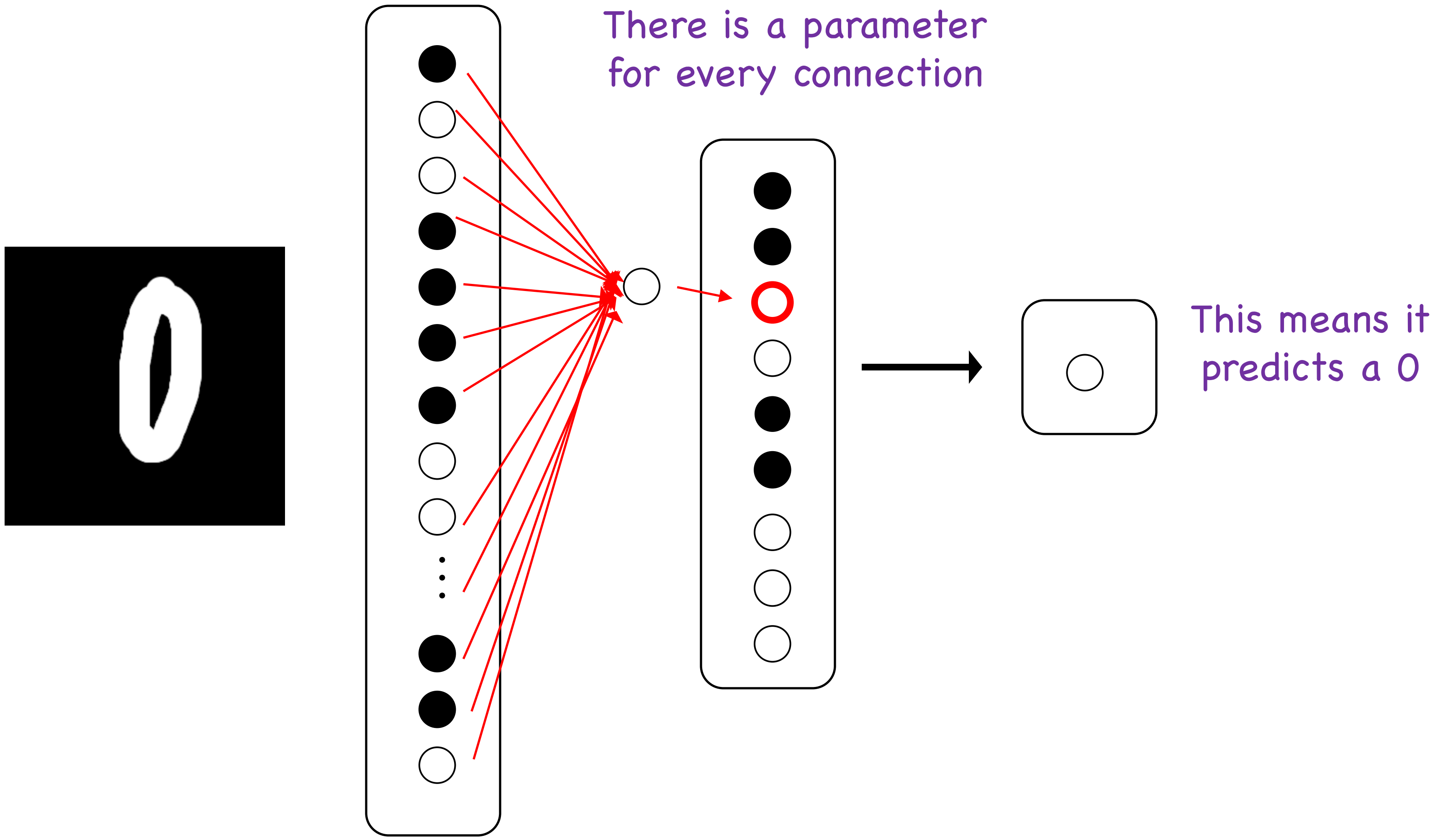


We Can Put Neurons Together



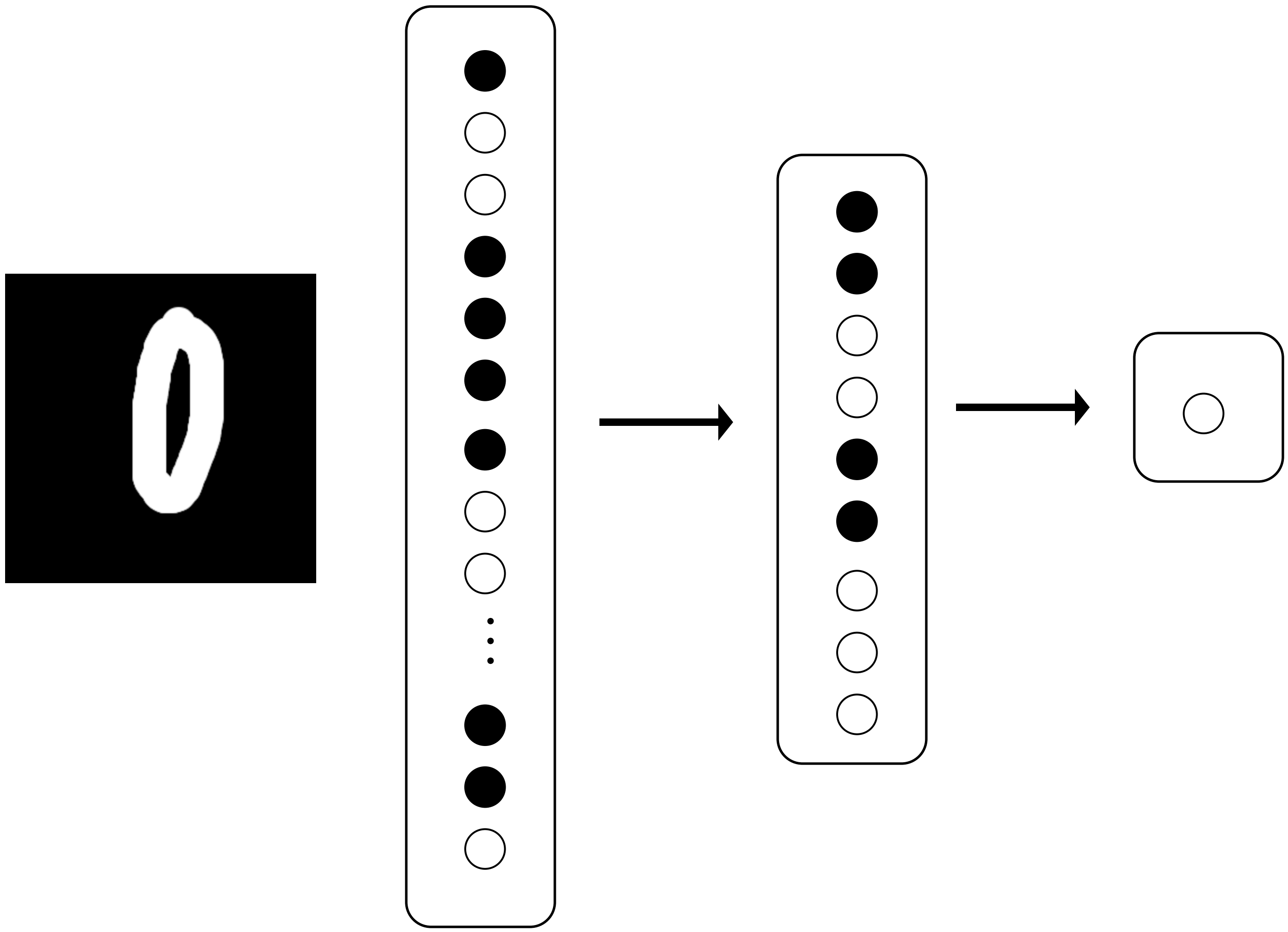
Look at a single "hidden" neuron

We Can Put Neurons Together



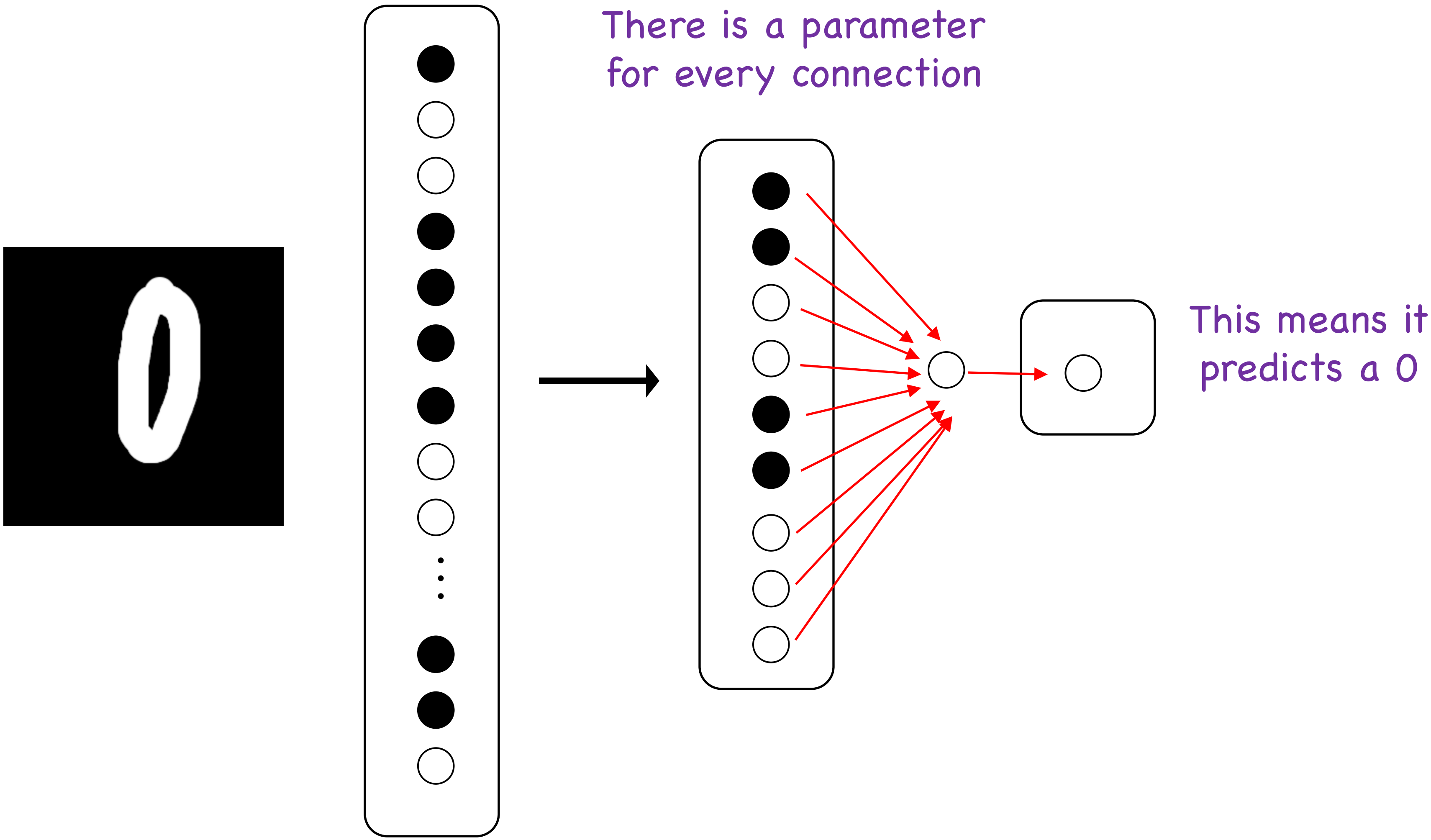
Look at a single “hidden” neuron

We Can Put Neurons Together



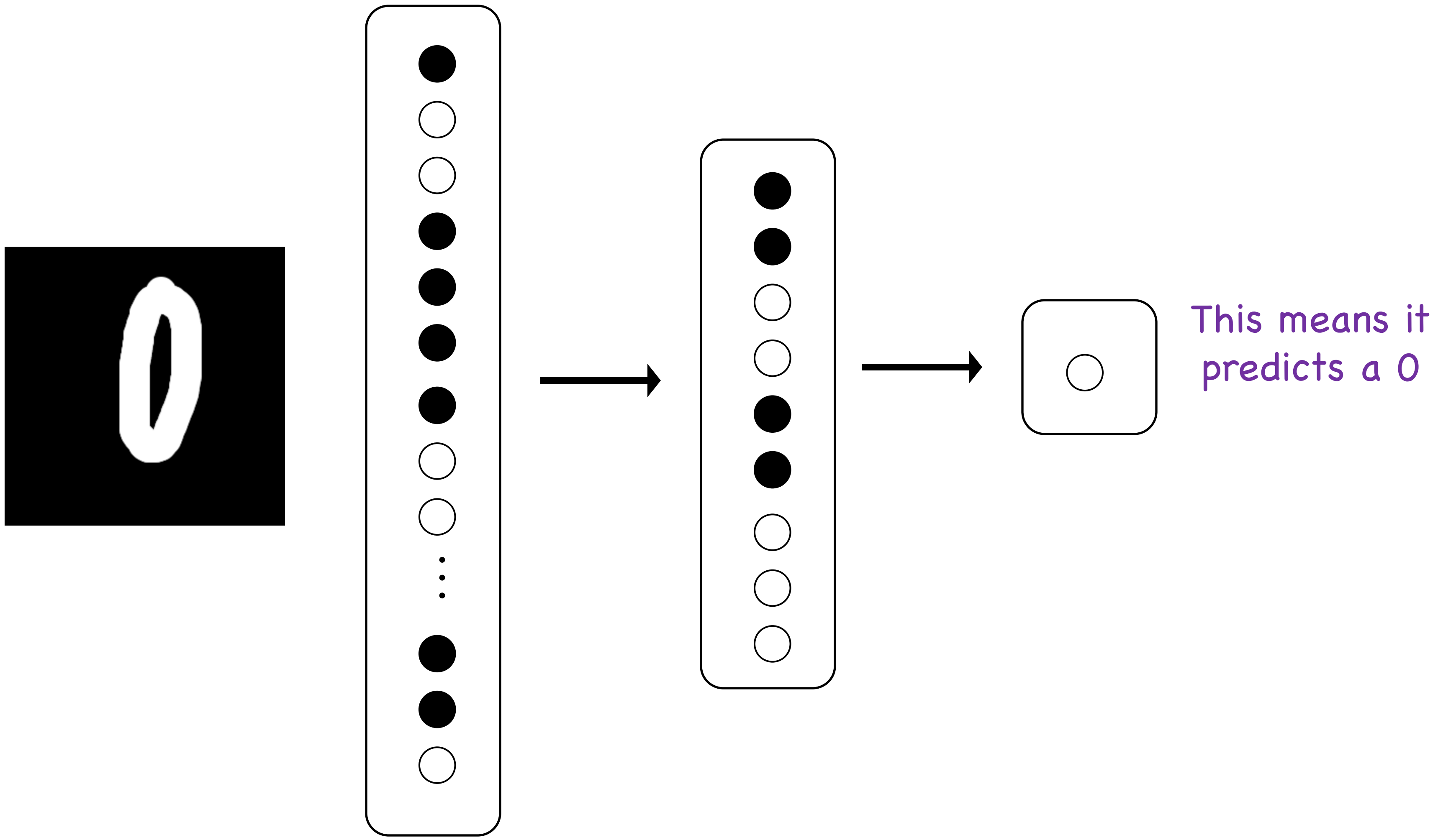
This means it
predicts a 0

We Can Put Neurons Together



Look at another neuron

We Can Put Neurons Together

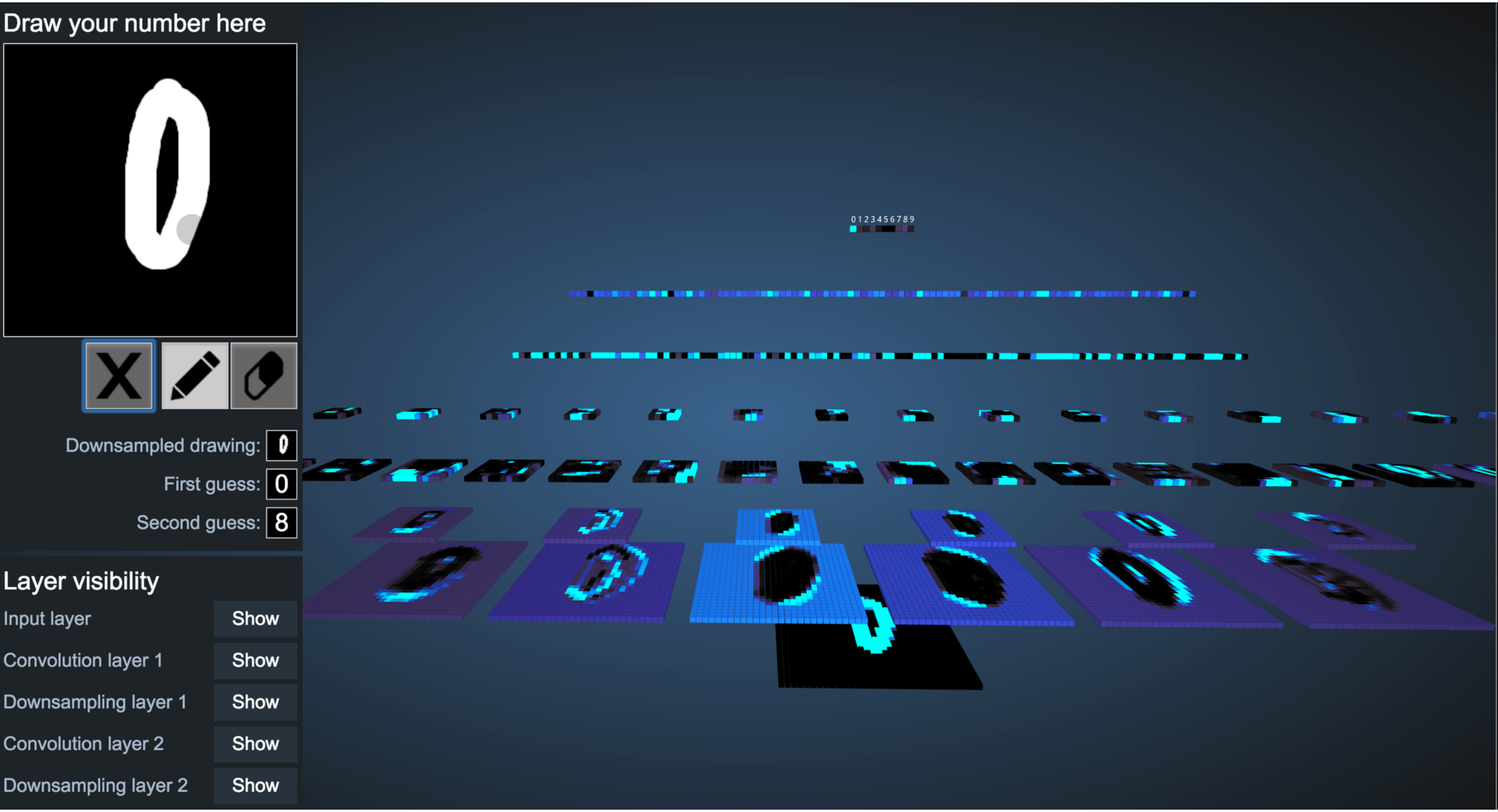


Deep Learning Assumption

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

$P(Y = 1|\mathbf{X}) =$ The output of a chain of logistic regressions

Demonstration



http://scs.ryerson.ca/~aharley/vis/conv/

Deep learning gets its ***intelligence*** from its thetas (aka its parameters)

How do we train?

MLE of Thetas!

First: Learning Goals…

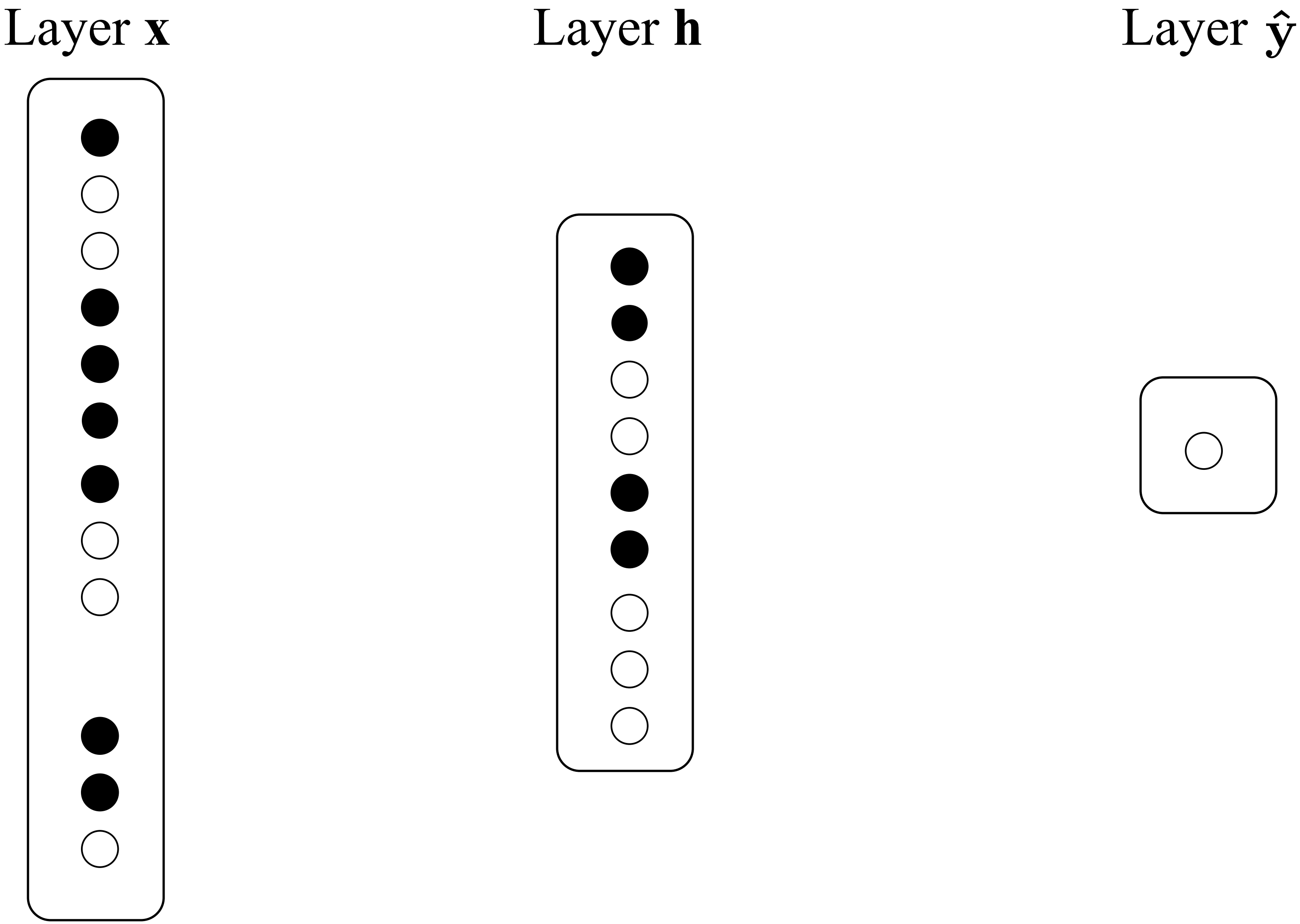
1. Understand Chain Rule as ♡ of Deep Learning

2. Everyone should be able to do simple derivations

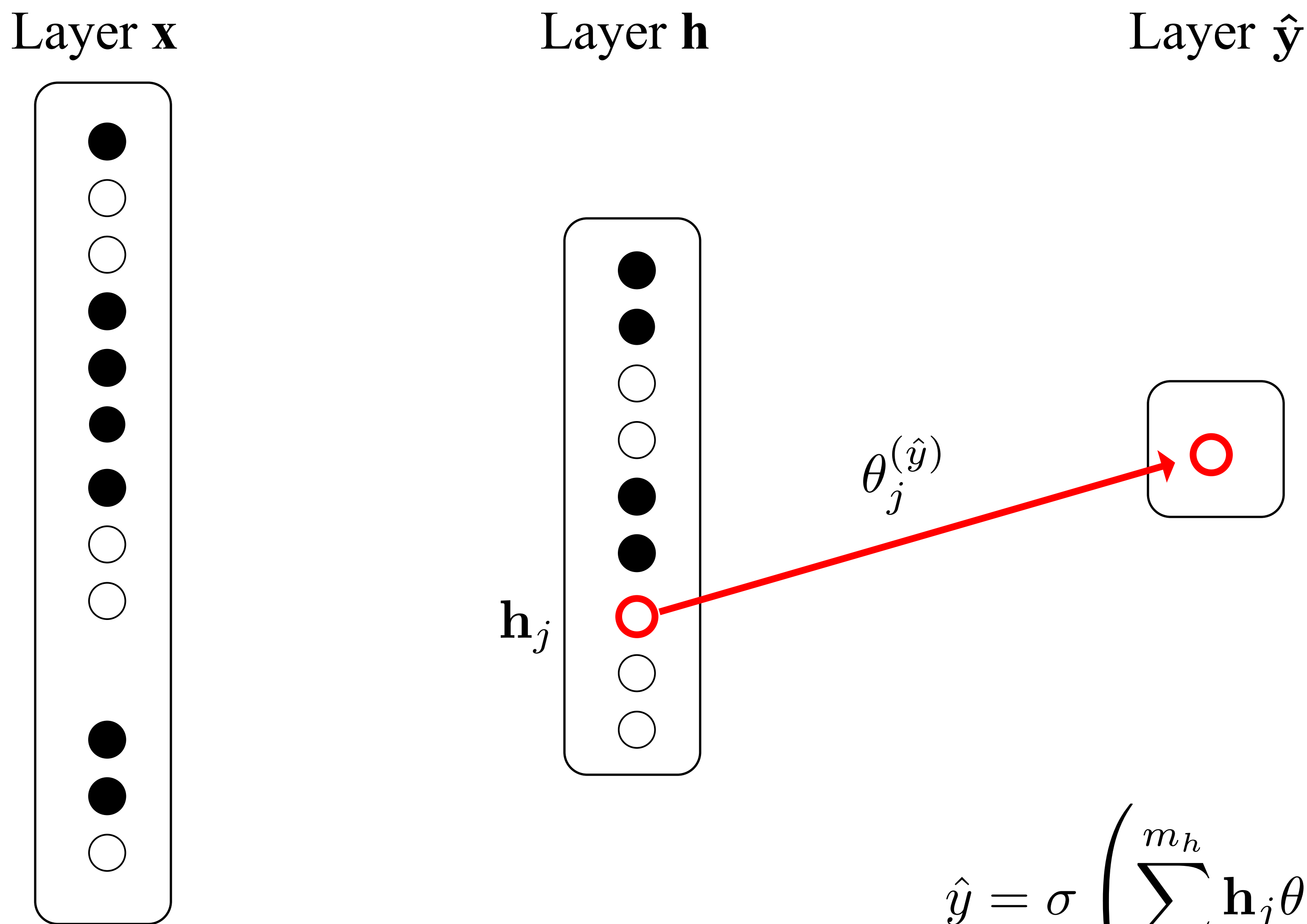
3. Be ready to rock
the socks off of CS221 and CS224N

Math worth knowing:

New Notation

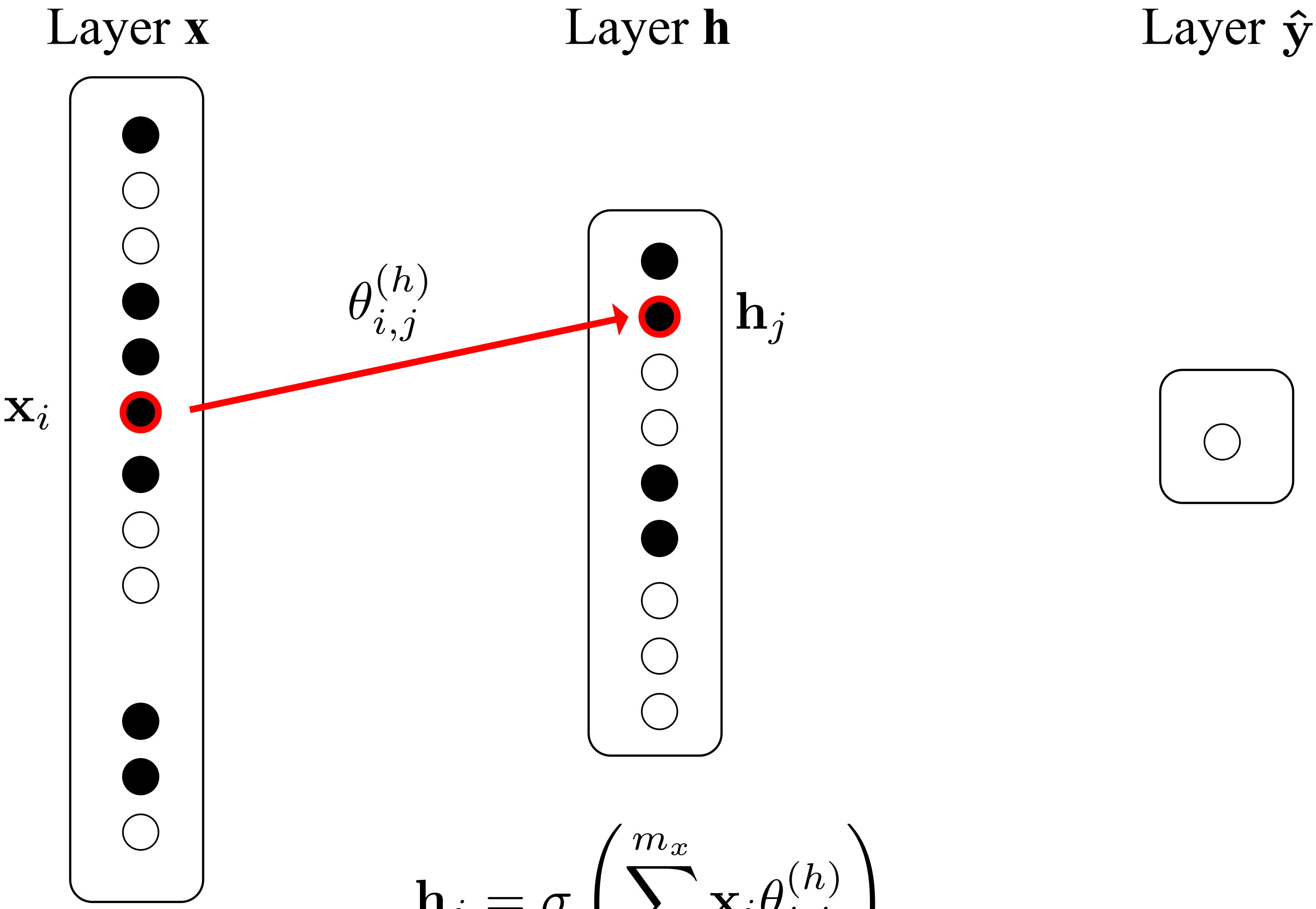


New Notation



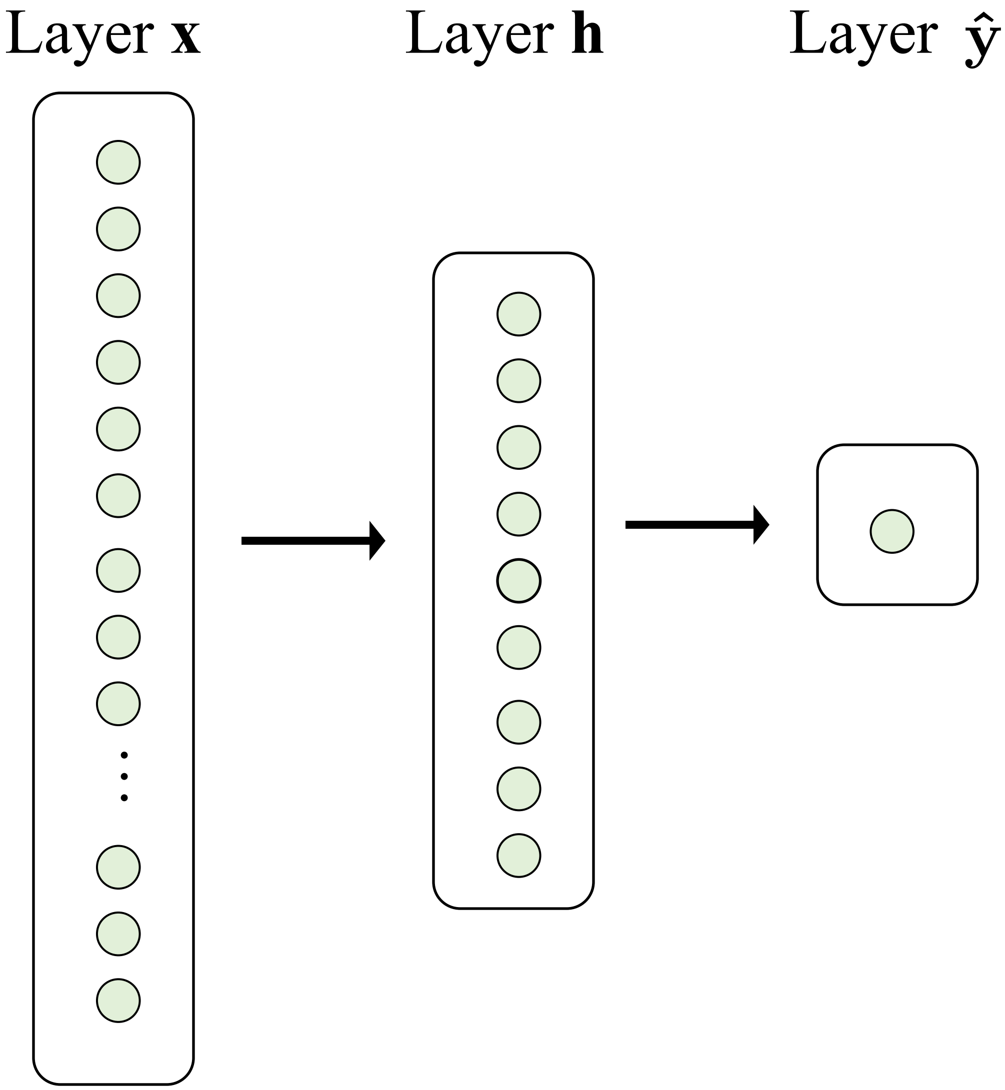
$$\hat{y} = \sigma\left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}\right)$$

New Notation

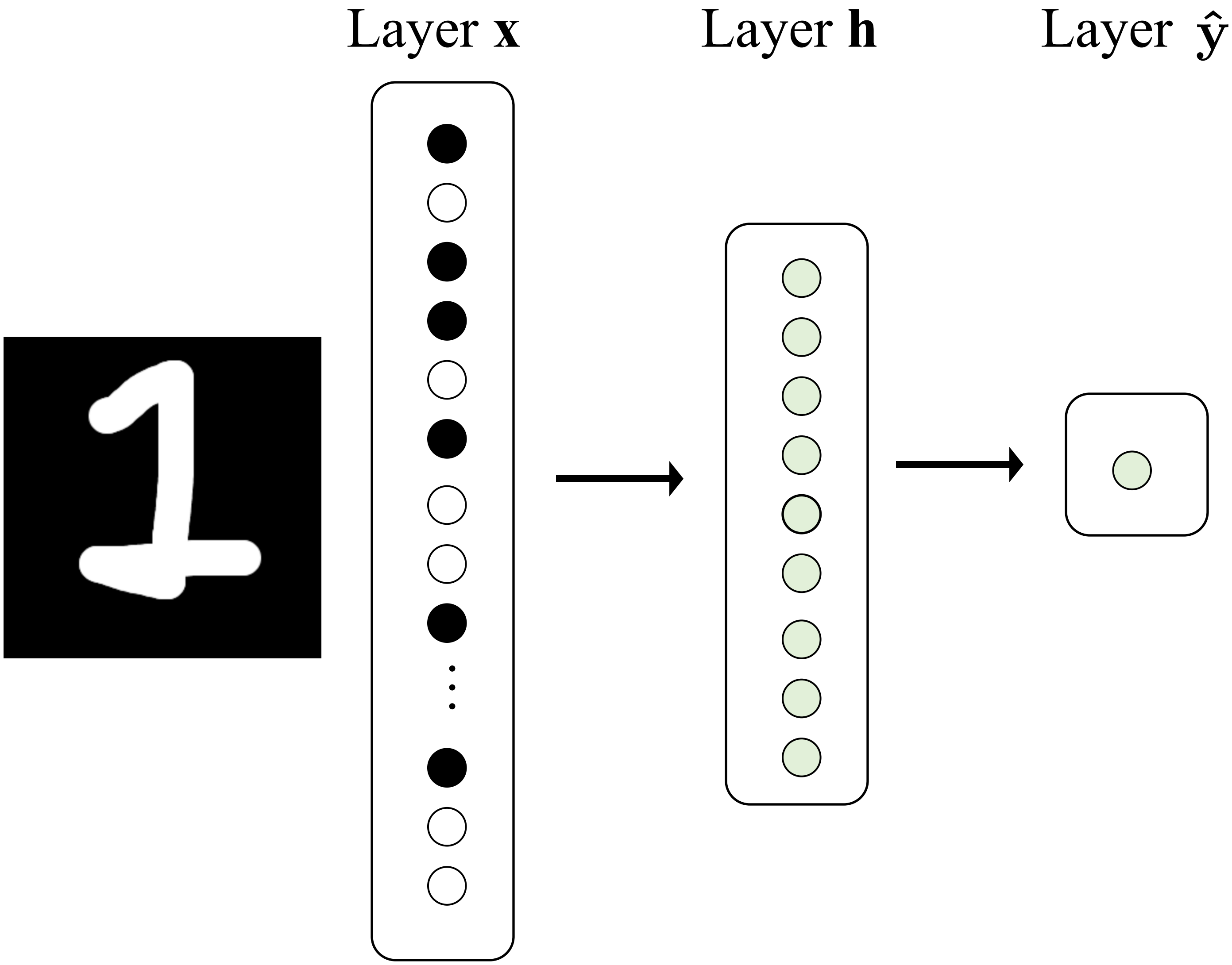


$$\mathbf{h}_j = \sigma\left(\sum_{i=0}^{m_x} \mathbf{x}_i \theta_{i,j}^{(h)}\right)$$

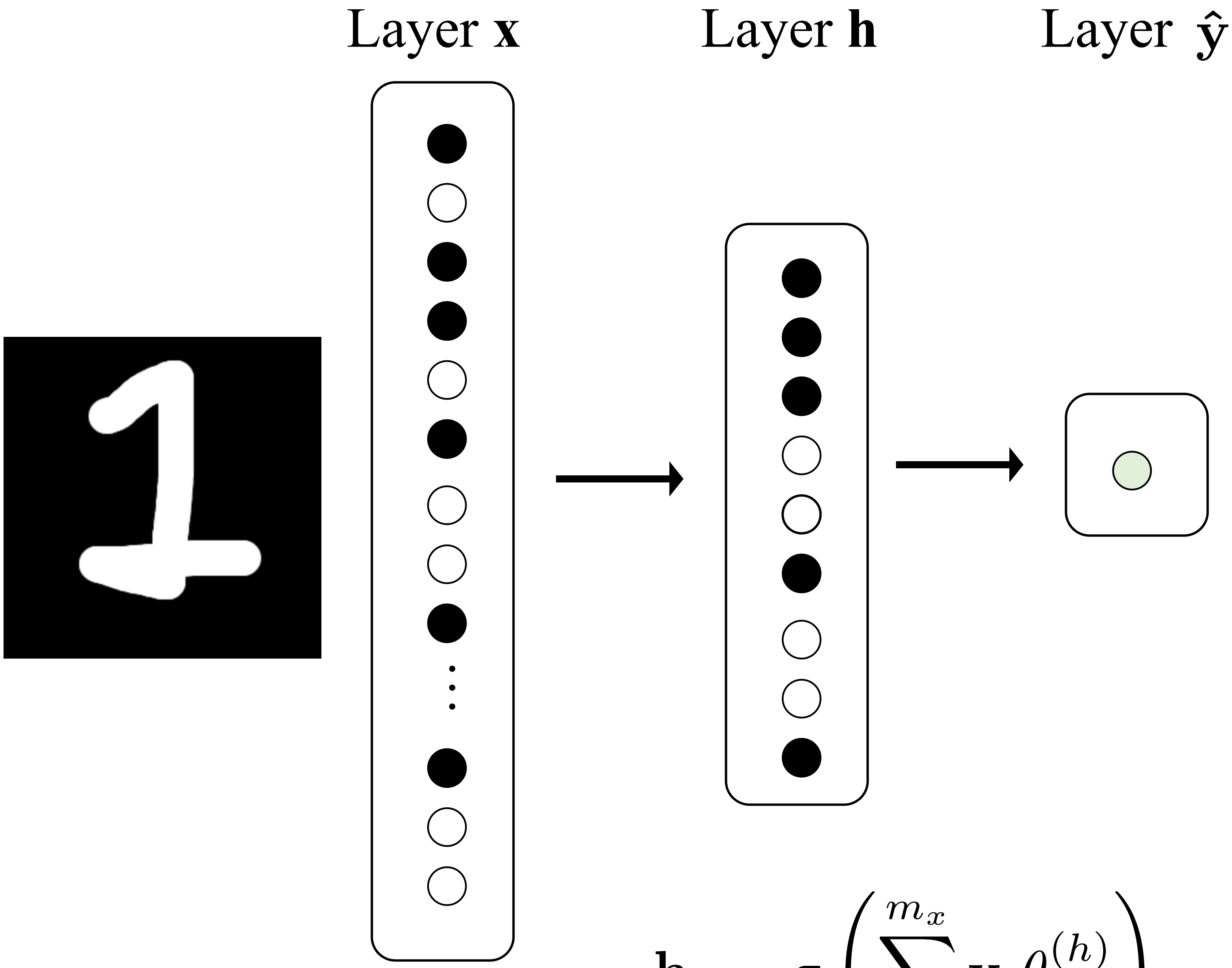
Forward Pass



Forward Pass

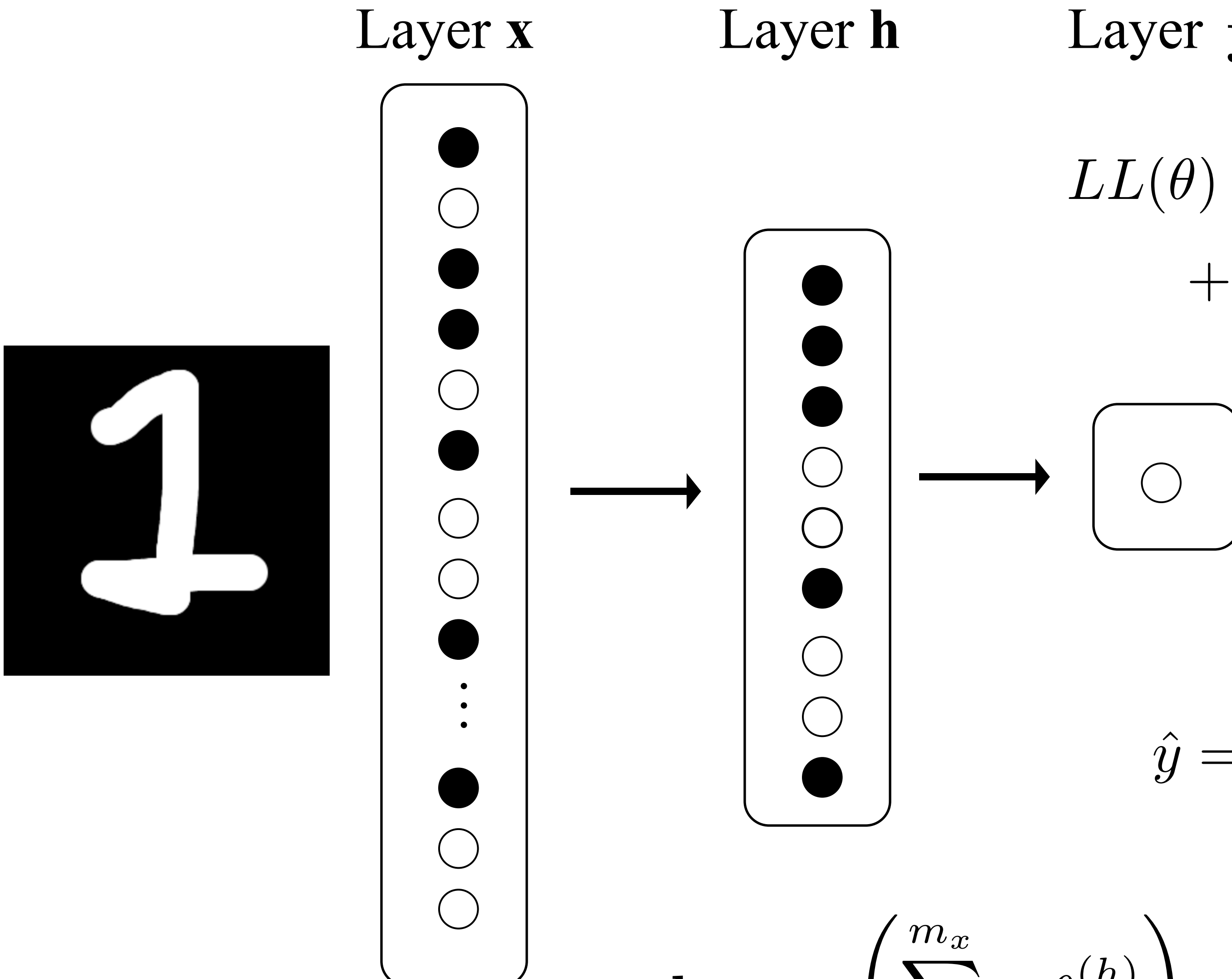


Forward Pass



$$\mathbf{h}_j = \sigma\left(\sum_{i=0}^{m_x} \mathbf{x}_i \theta_{i,j}^{(h)}\right)$$

Forward Pass



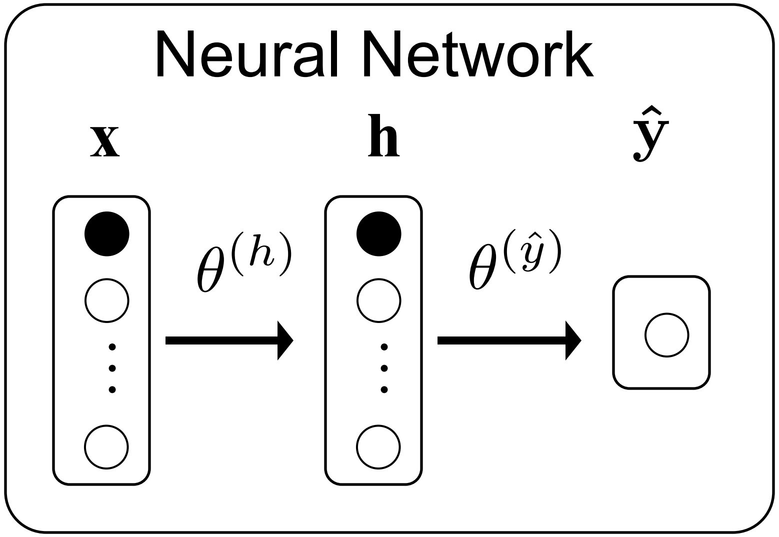
$$LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}]$$

$$\hat{y} = \sigma\left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}\right)$$

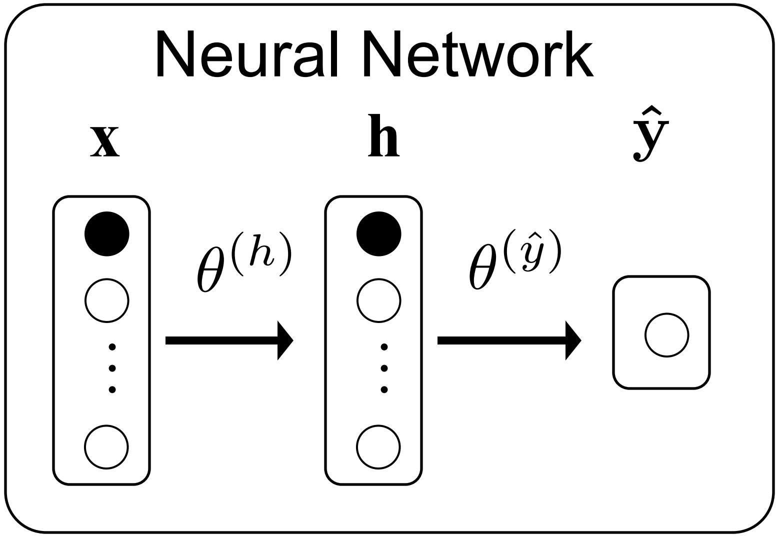
$$\mathbf{h}_j = \sigma\left(\sum_{i=0}^{m_x} \mathbf{x}_i \theta_{i,j}^{(h)}\right)$$

Congrats. You now know
Forward Propagation

All Together



Sanity Check



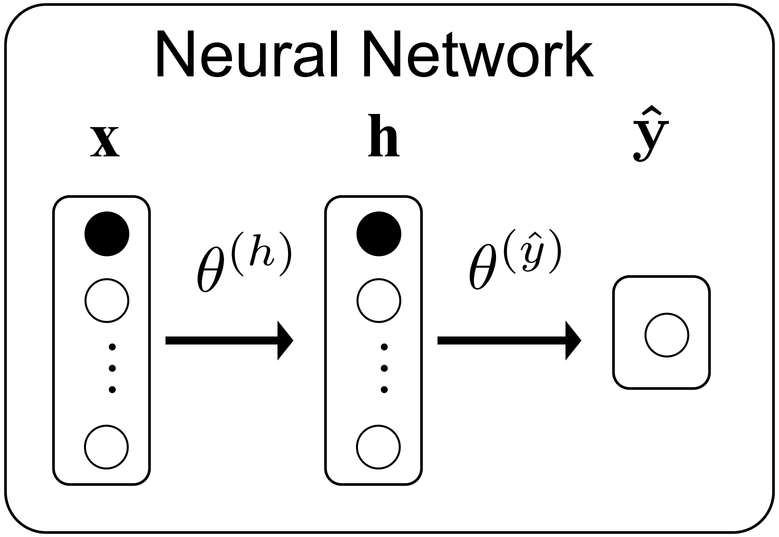
$$|\mathbf{x}| = 40$$

$$|\mathbf{h}| = 20$$

How many parameters in $\theta^{(h)}$?

c) 2 a) 20 b) 40 c) 800

Sanity Check 2



$$|\mathbf{x}| = 40$$

$$|\mathbf{h}| = 20$$

How many parameters in $\theta^{(\hat{y})}$?

c) 2 a) 20 b) 40 c) 800

Today: Do Something Brave

Only Have to Do Three Things

1. Make deep learning assumption

2. Calculate the log probability for all data

3. Get partial derivative of log likelihood with respect to each theta

Sanity Check

3 Get partial derivative of log likelihood with respect to each theta

Why?

Same Assumption, Same LL

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

For one datum

$$P(Y = y | X = \mathbf{X}) = (\hat{y})^y (1 - \hat{y})^{1-y}$$

For IID data

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

$$= \prod_{i=1}^{n} (\hat{y}^{(i)})^{y^{(i)}} \cdot \left[1 - (\hat{y}^{(i)})\right]^{(1 - y^{(i)})}$$

Take the log

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

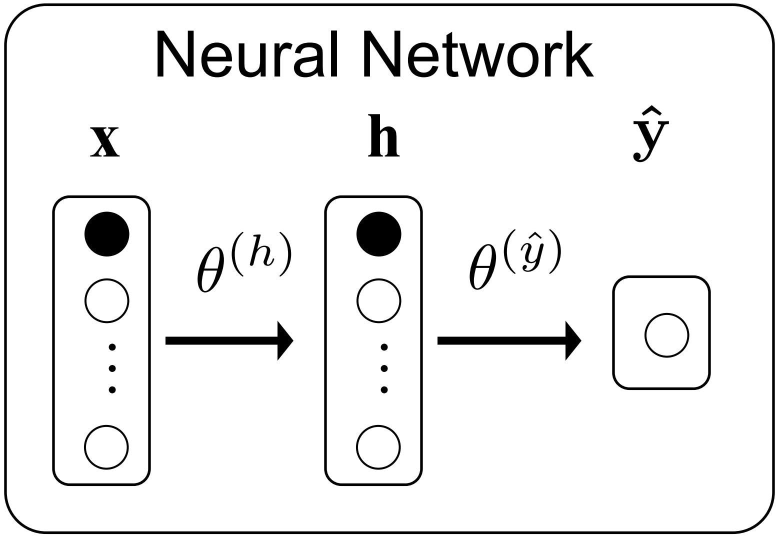
Derivative Goals

Loss with respect to output layer params

$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}}$$

Loss with respect to hidden layer params

$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}}$$



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!

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

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

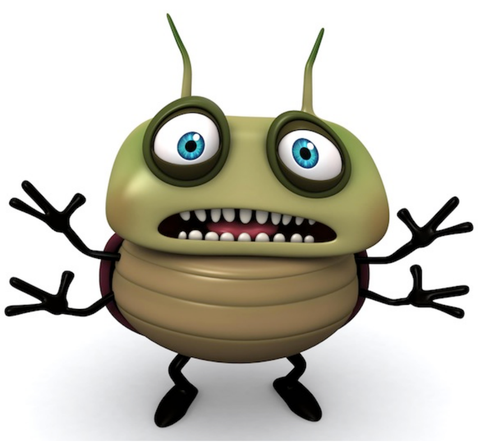
Bad Approach

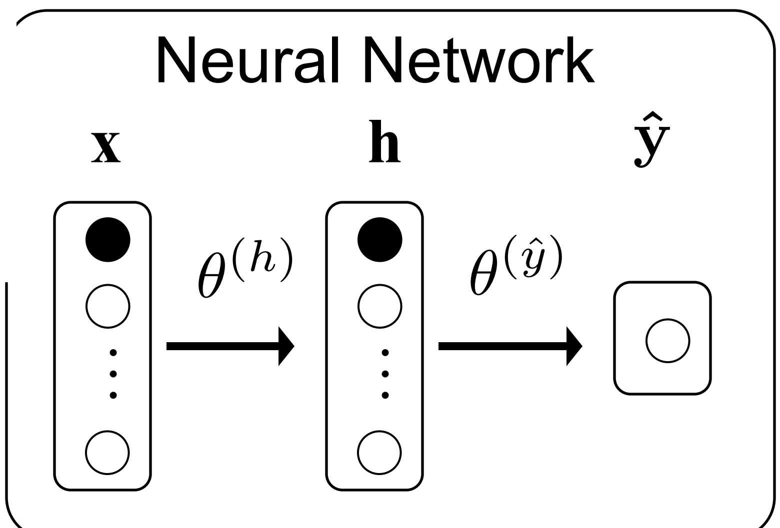
$$LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}]$$

$$\hat{\mathbf{y}} = \sigma\left(\sum_{i=0}^{m_h} \mathbf{h}_i \theta_i^{(\hat{\mathbf{y}})}\right)$$

Math bug

$$= \sigma\left(\sum_{i=0}^{m_h} \left[\sigma\left(\sum_{i=0}^{m_x} \mathbf{x}_i \theta_{i,j}^{(\mathbf{h})}\right)\right] \theta_i^{(\hat{\mathbf{y}})}\right)$$





Big Idea

Big Idea

People who knew chain rule revolutionized AI

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

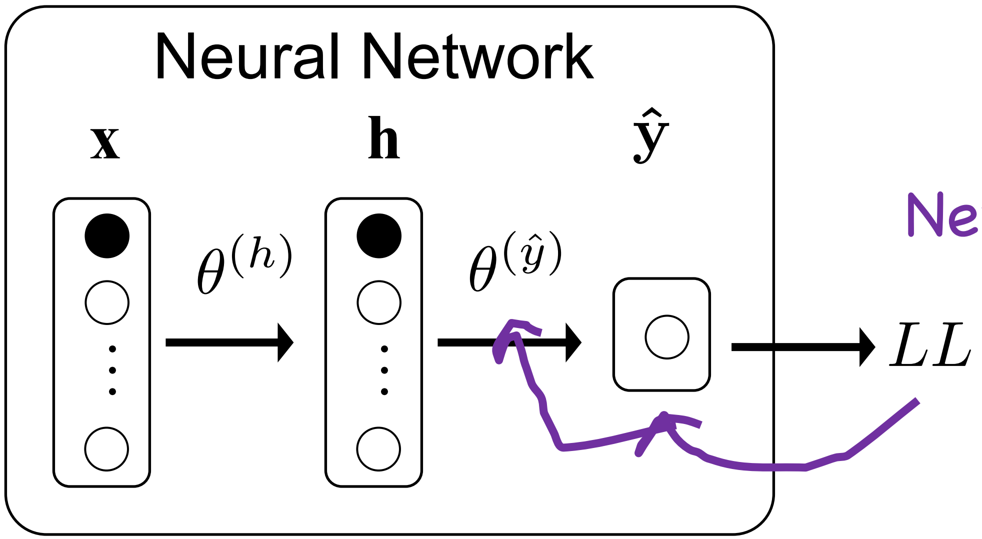
First use:

$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}}$$

Chain Rule Example 1

$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}}$$

Goal



Network

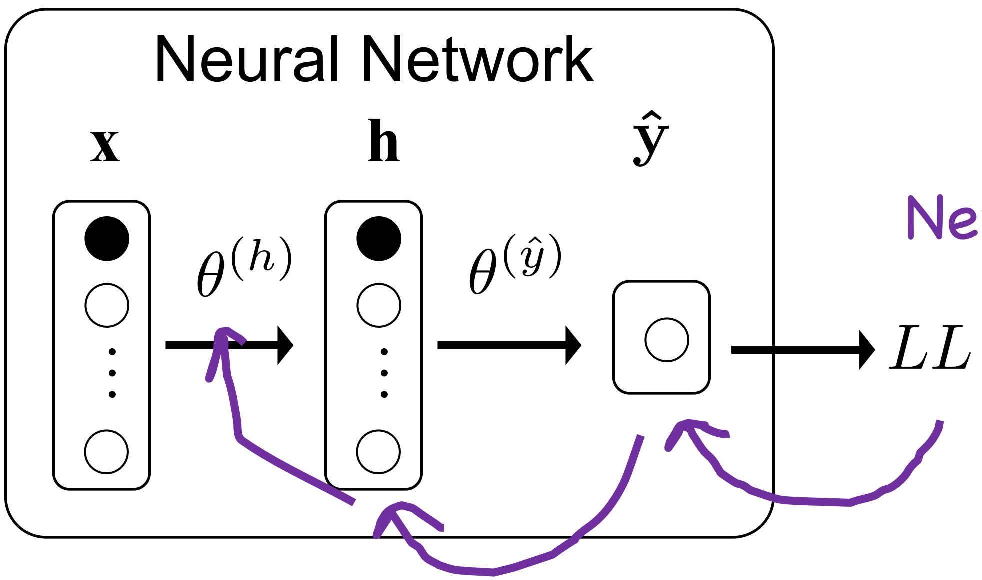
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}}$$

Decomposition

Chain Rule Example 2

$$\frac{\partial LL(\theta)}{\partial \theta^{(h)}_{i,j}}$$

Goal



Network

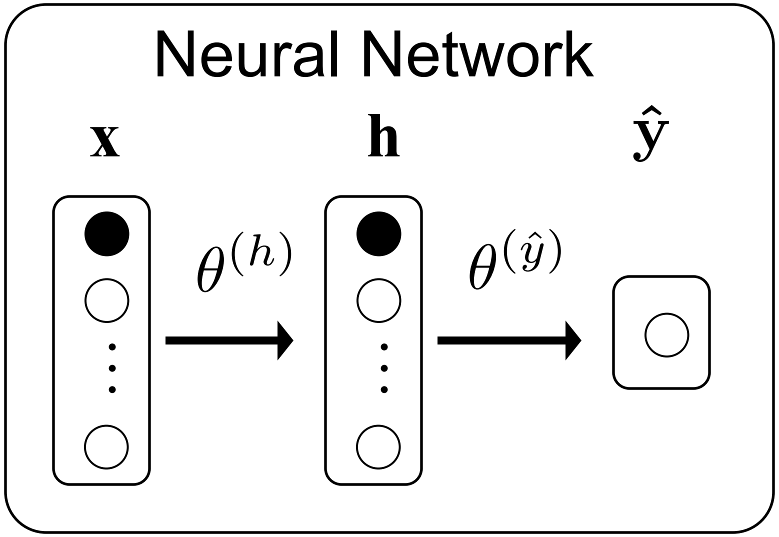
$$\frac{\partial LL(\theta)}{\partial \theta^{(h)}_{i,j}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{h}_j} \cdot \frac{\partial \mathbf{h}_j}{\partial \theta^{(h)}_{i,j}}$$

Decomposition

Decomposition

Gradient of output layer params

$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}}$$



Gradient of output layer params

$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \boxed{\frac{\partial LL}{\partial \hat{y}}} \cdot \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}}$$

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

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

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

Gradient of output layer params

$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \boxed{\frac{\partial LL}{\partial \hat{y}}} \cdot \boxed{\frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}}}$$

$$\hat{y} = \sigma\left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}\right) = \sigma(z) \quad \text{where} \quad z = \sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}$$

$$\frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}} = \hat{y}[1-\hat{y}] \cdot \frac{\partial}{\partial \theta_i^{(\hat{y})}} \sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}$$

$$= \hat{y}[1-\hat{y}] \cdot h_i$$

What! That's not scary!

Make it Simple

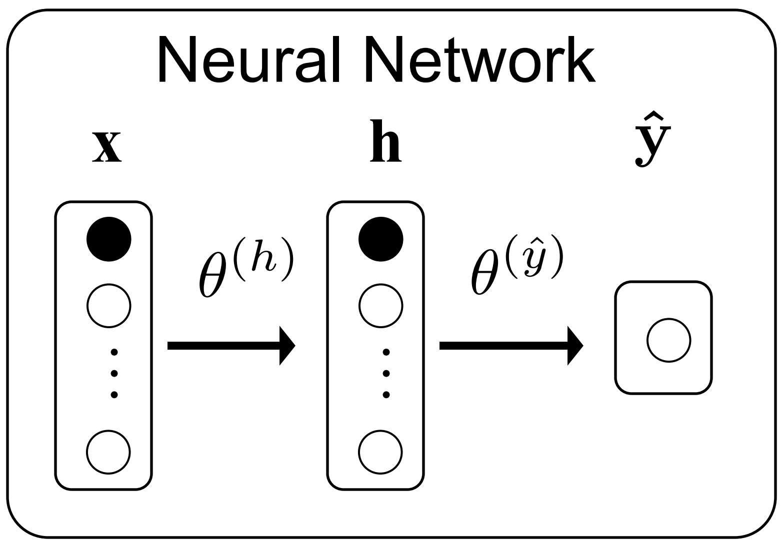
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} =$$  $\cdot$ 

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

 $$= \hat{y}[1-\hat{y}] \cdot h_i$$

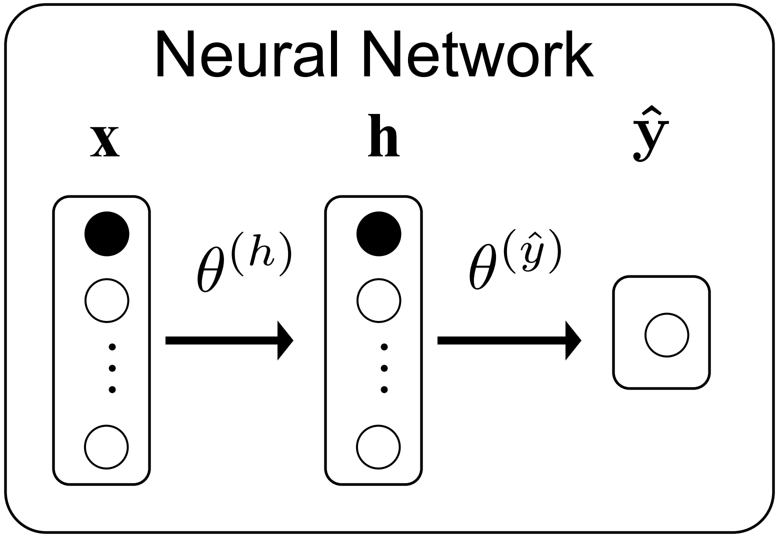
Boom!

$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}}$$



Gradient of hidden layer params

$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{h}_j} \cdot \frac{\partial \mathbf{h}_j}{\partial \theta_{i,j}^{(h)}}$$



Gradient of hidden layer params

$$\frac{\partial LL(\theta)}{\partial \theta^{(h)}_{i,j}} = \boxed{\frac{\partial LL}{\partial \hat{y}}} \cdot \boxed{\frac{\partial \hat{y}}{\partial \mathbf{h}_j}} \cdot \frac{\partial \mathbf{h}_j}{\partial \theta^{(h)}_{i,j}}$$

$$\hat{y} = \sigma\left(\sum_{i=0}^{m_h} \mathbf{h}_i \theta_i^{(\hat{y})}\right)$$

$$\frac{\partial \hat{y}}{\partial \mathbf{h}_j} = \hat{y}[1 - \hat{y}]\theta_j^{(\hat{y})}$$

Wait is it over?

Gradient of hidden layer params

$$\frac{\partial LL(\theta)}{\partial \theta^{(h)}_{i,j}} = \boxed{\frac{\partial LL}{\partial \hat{y}}} \cdot \boxed{\frac{\partial \hat{y}}{\partial \mathbf{h}_j}} \cdot \boxed{\frac{\partial \mathbf{h}_j}{\partial \theta^{(h)}_{i,j}}}$$

$$\mathbf{h}_j = \sigma\left(\sum_{k=0}^{m_x} \mathbf{x}_k \theta_{k,j}\right)$$

$$\frac{\partial \mathbf{h}_j}{\partial \theta^{(h)}_{i,j}} = \mathbf{h}_j[1 - \mathbf{h}_j]\mathbf{x}_j$$

That one too?

Make it Simple

$$\frac{\partial LL(\theta)}{\partial \theta^{(h)}_{i,j}} =$$

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

$$= \hat{y}[1-\hat{y}]\theta_j^{(\hat{y})}$$

$$= \mathbf{h}_j[1-\mathbf{h}_j]\mathbf{x}_j$$

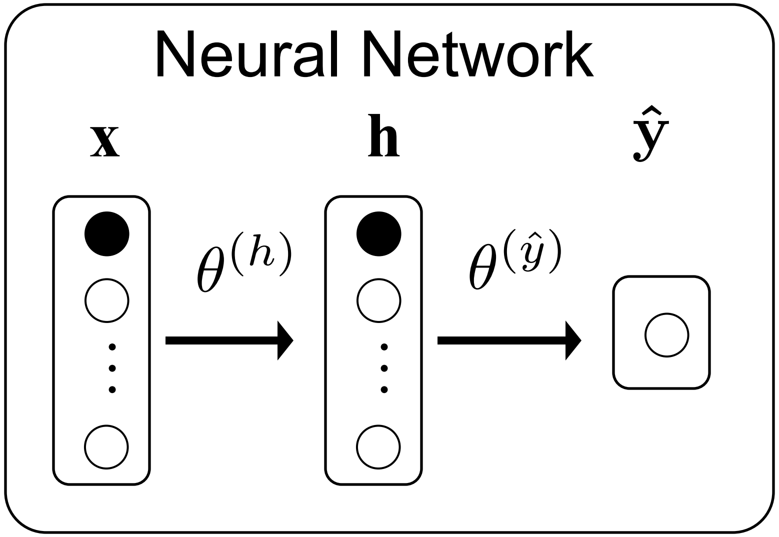
Summary: Simple Calculations For

Loss with respect to output layer params

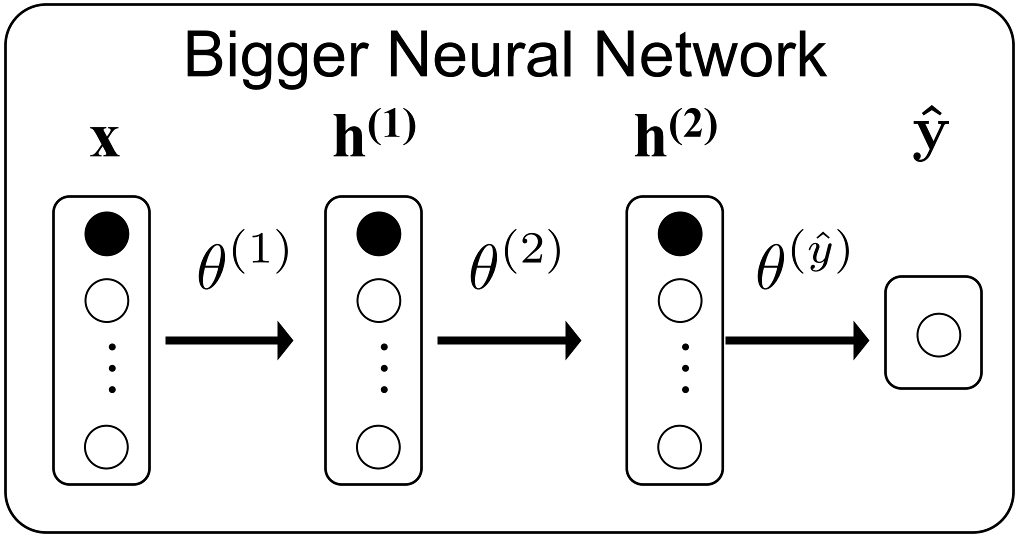
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}}$$

Loss with respect to hidden layer params

$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}}$$



What Would You Do Here?

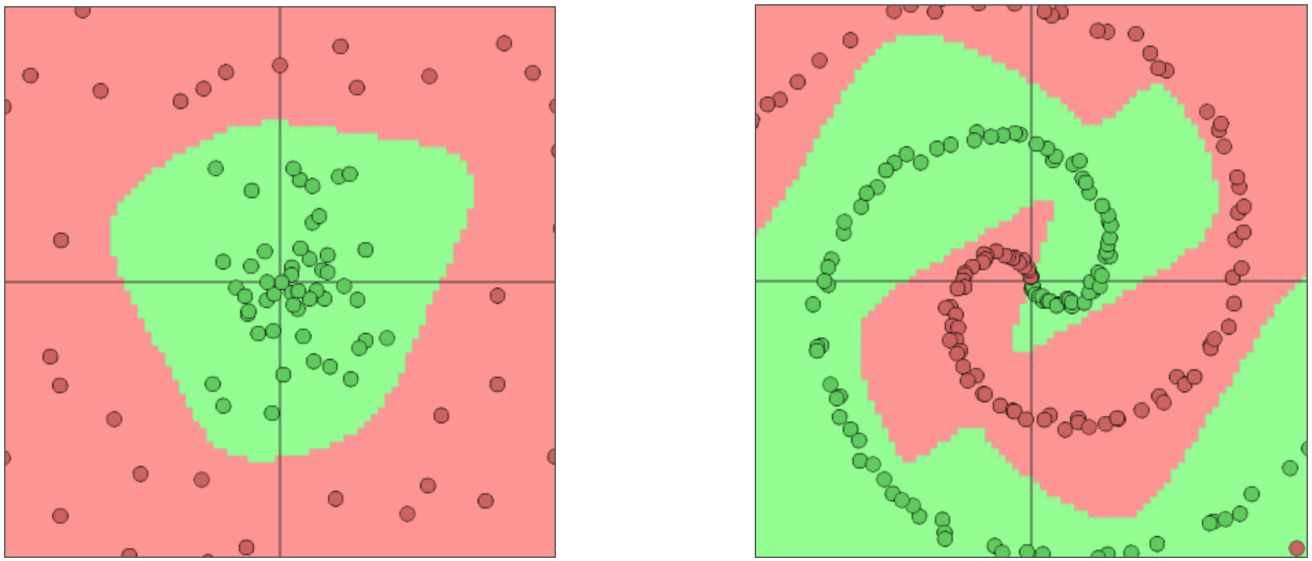


Chain rule:
Game changer for
artificial intelligence

Congrats. You now know Backpropagation

Neural Networks Can Learn Complex Functions

- Some data sets/functions are not separable



  - These are classifiers learned by neural networks

http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html