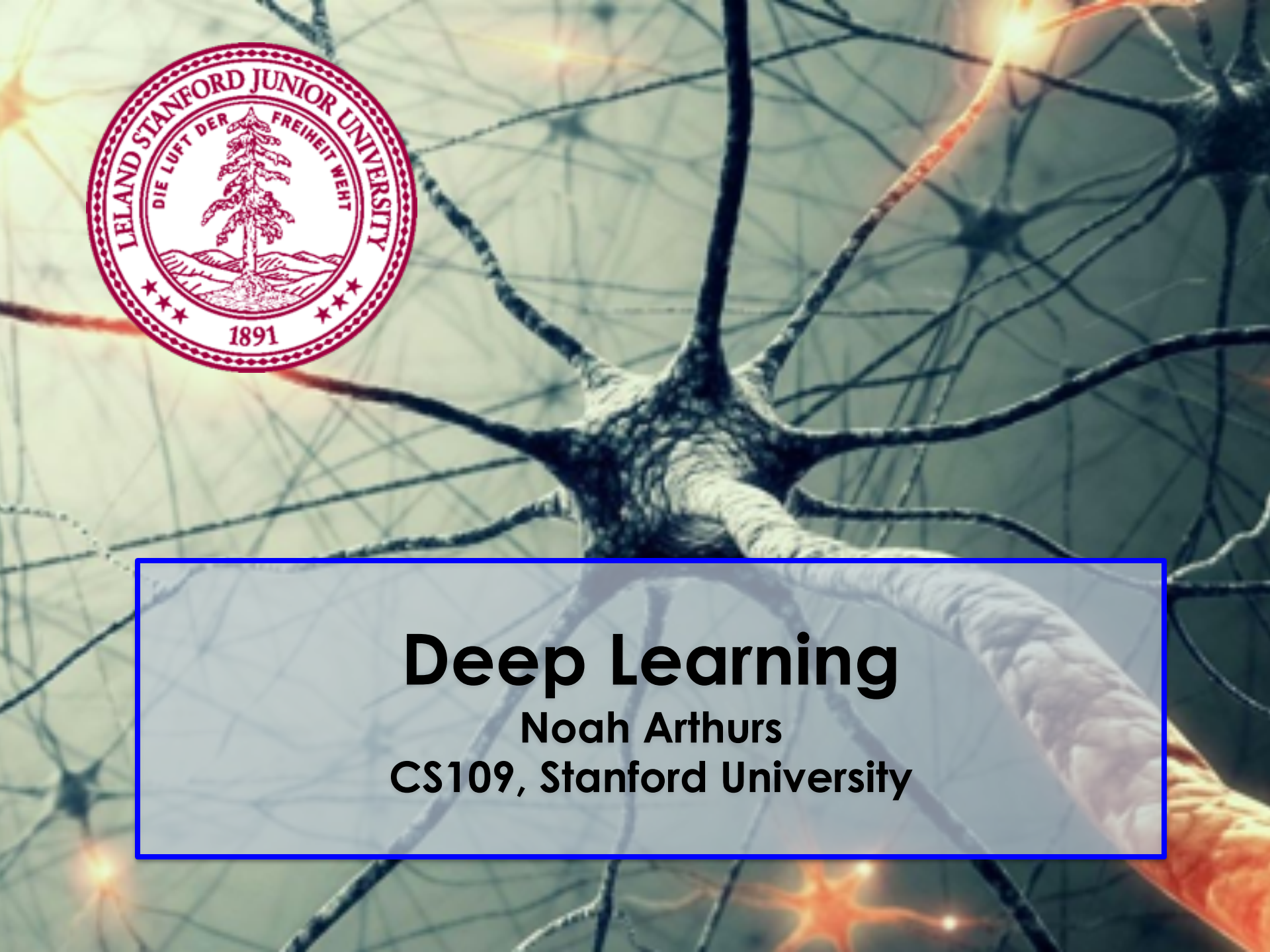


Deep Learning

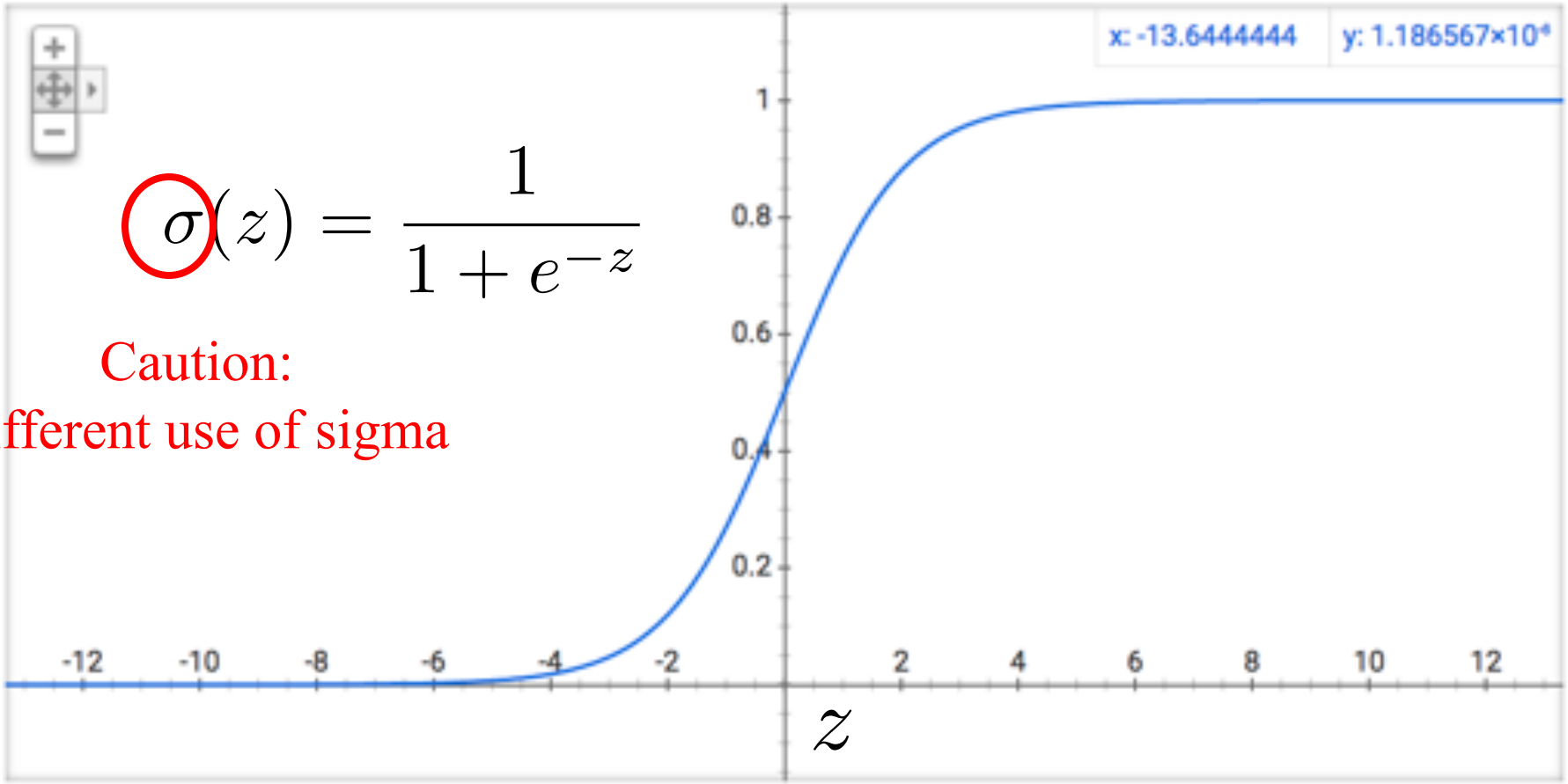
Noah Arthurs
CS109, Stanford University

Review

Background: Sigmoid Function

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

Caution:
different use of sigma



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

Background: Dot Product

If $\theta, x \in \mathcal{R}^n$, then:

$$\theta \cdot x = \theta^T x = \sum_{i=1}^{n} \theta_i x_i = \theta_1 x_1 + \ldots + \theta_n x_n$$

Partial Derivative:

$$\frac{d}{d\theta_i} \theta \cdot x = \frac{d}{d\theta_i} (\theta_1 x_1 + \ldots + \theta_n x_n) = \frac{d}{d\theta_i} \theta_i x_i = x_i$$

Gradient:

$$\nabla_\theta \theta \cdot x = \left(\frac{d}{d\theta_1} \theta \cdot x, \ldots, \frac{d}{d\theta_n} \theta \cdot x \right)$$

$$= (x_1, \ldots, x_n) = x$$

Background: Chain Rule

Who knew calculus would be so useful?

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

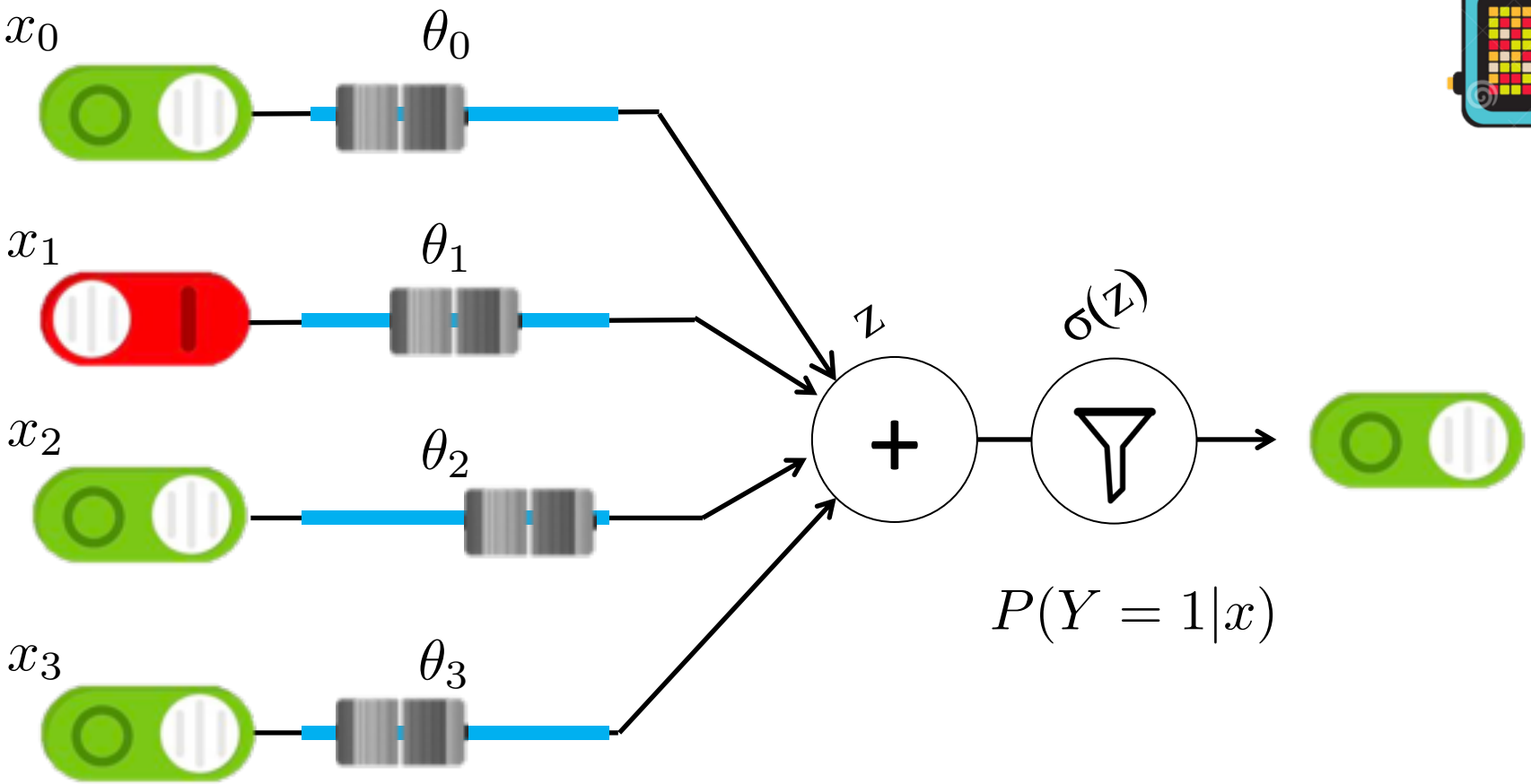
Aka decomposition of composed functions

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

Logistic Regression

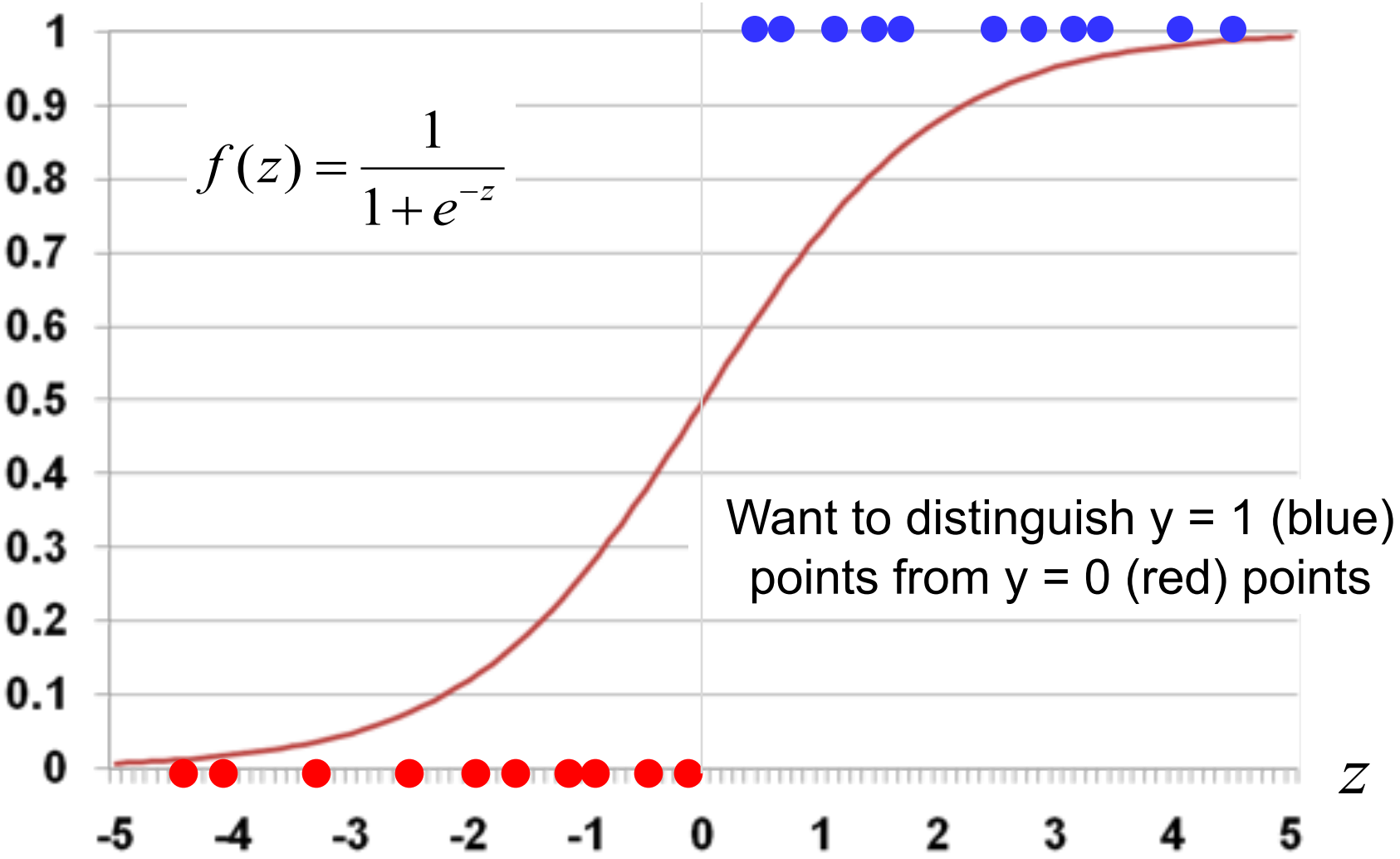
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Logistic Regression Cartoon



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

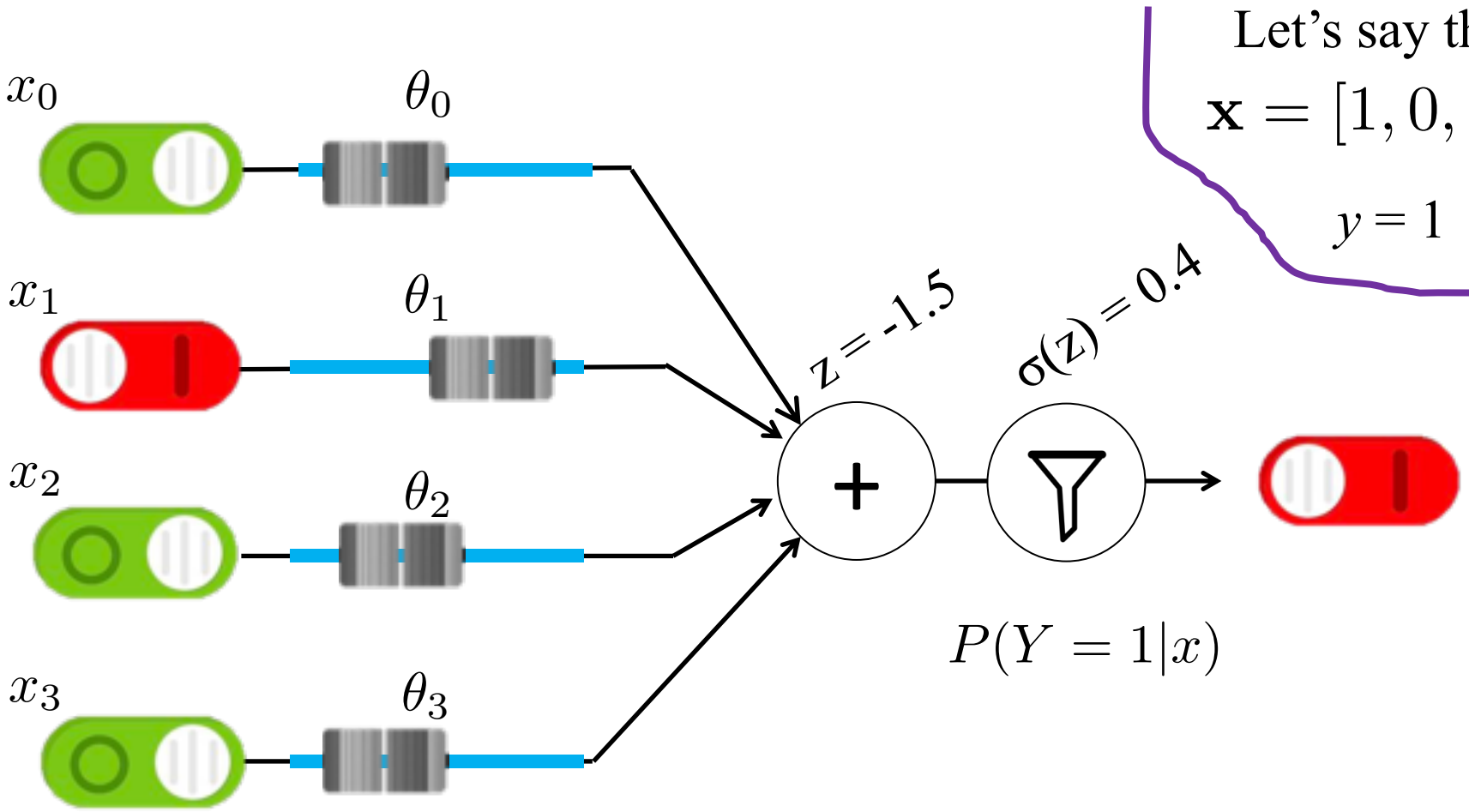
The Sigmoid Function



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

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

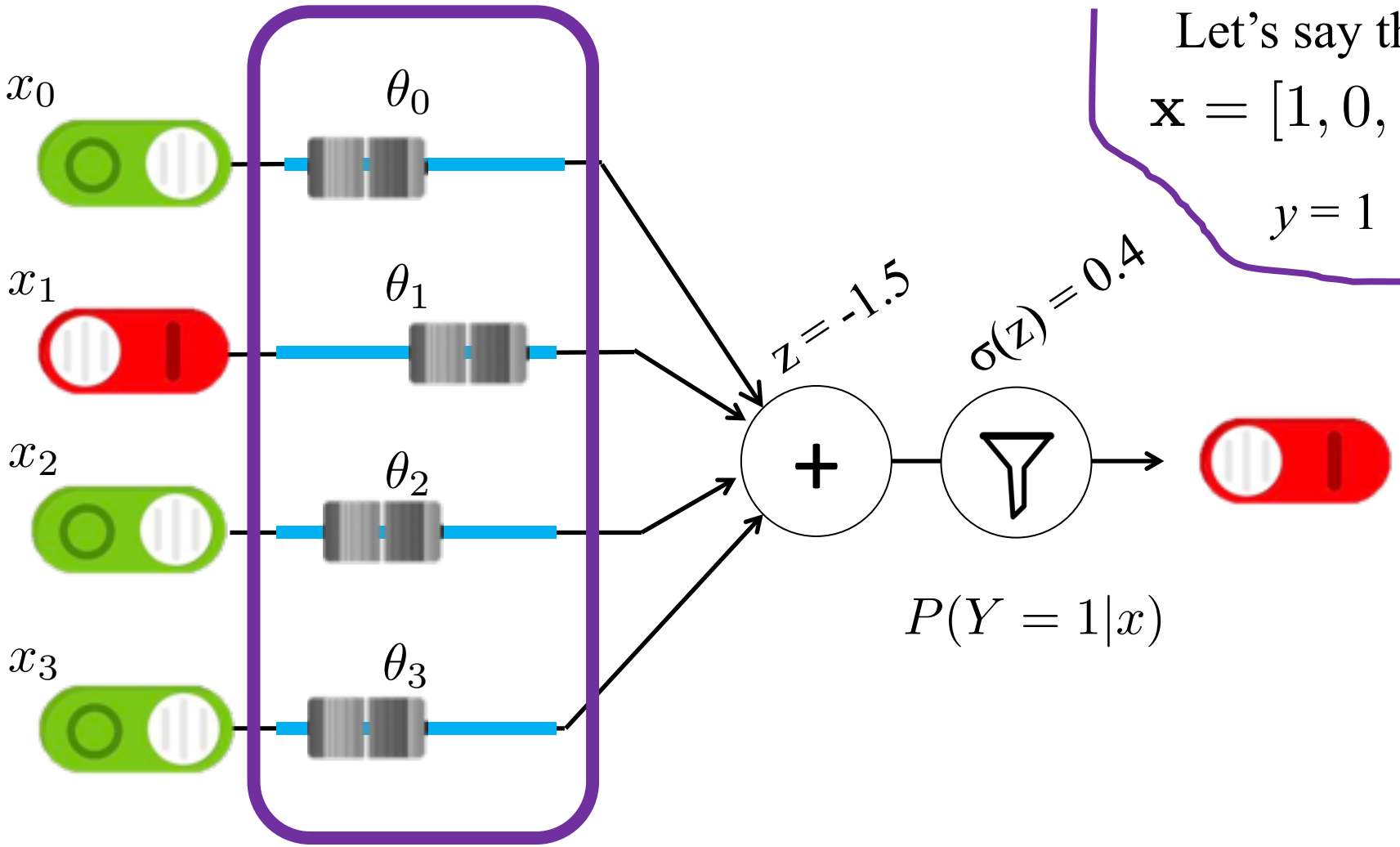
How Do We Learn Parameters?



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

Data looks unlikely

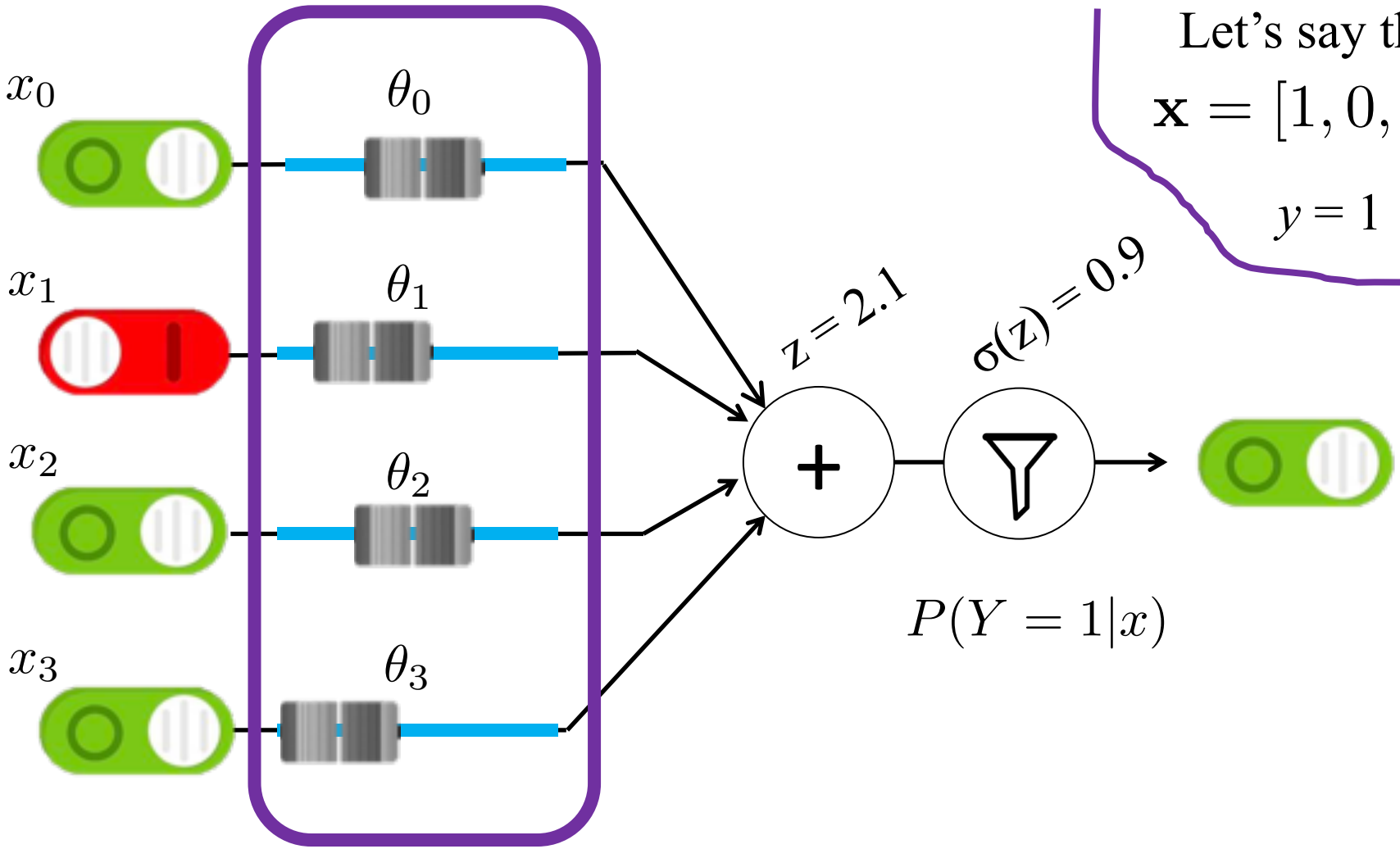
How Do We Learn Parameters?



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

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!

Math for Logistic Regression

1 Make logistic regression assumption

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

Often call this $\hat{y}$

2 Calculate the log likelihood for all data

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

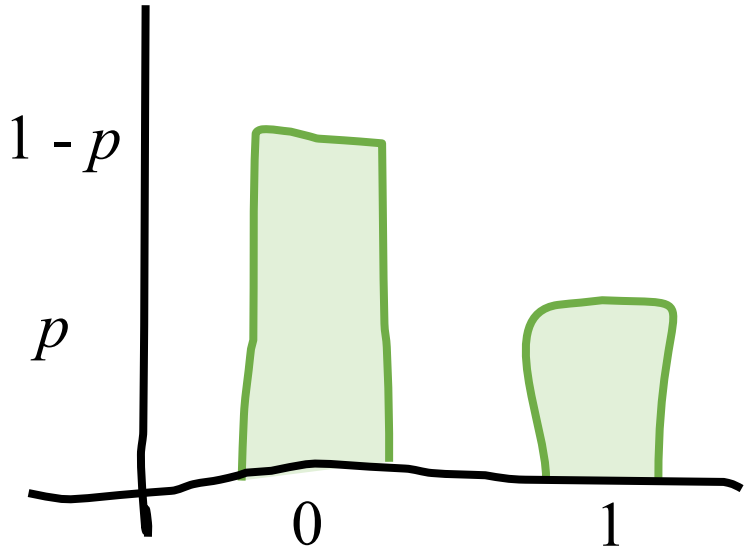
3 Get derivative of log likelihood with respect to thetas

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

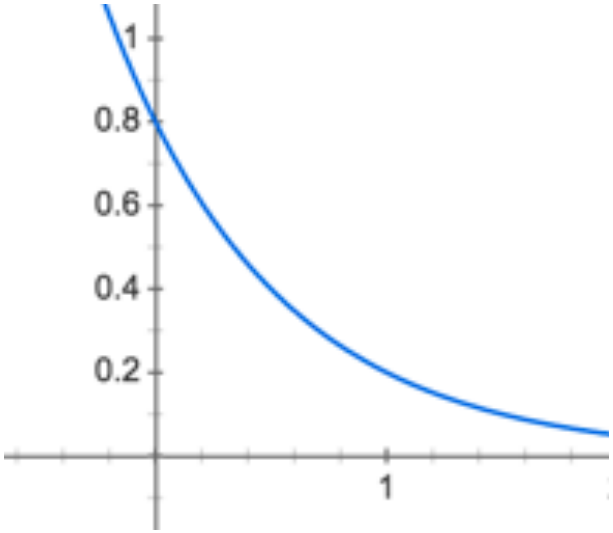
Recall: PMF of Bernoulli

- 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

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

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

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

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

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

First, imagine only one example

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

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

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

CHAIN RULE!

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

Already did that one

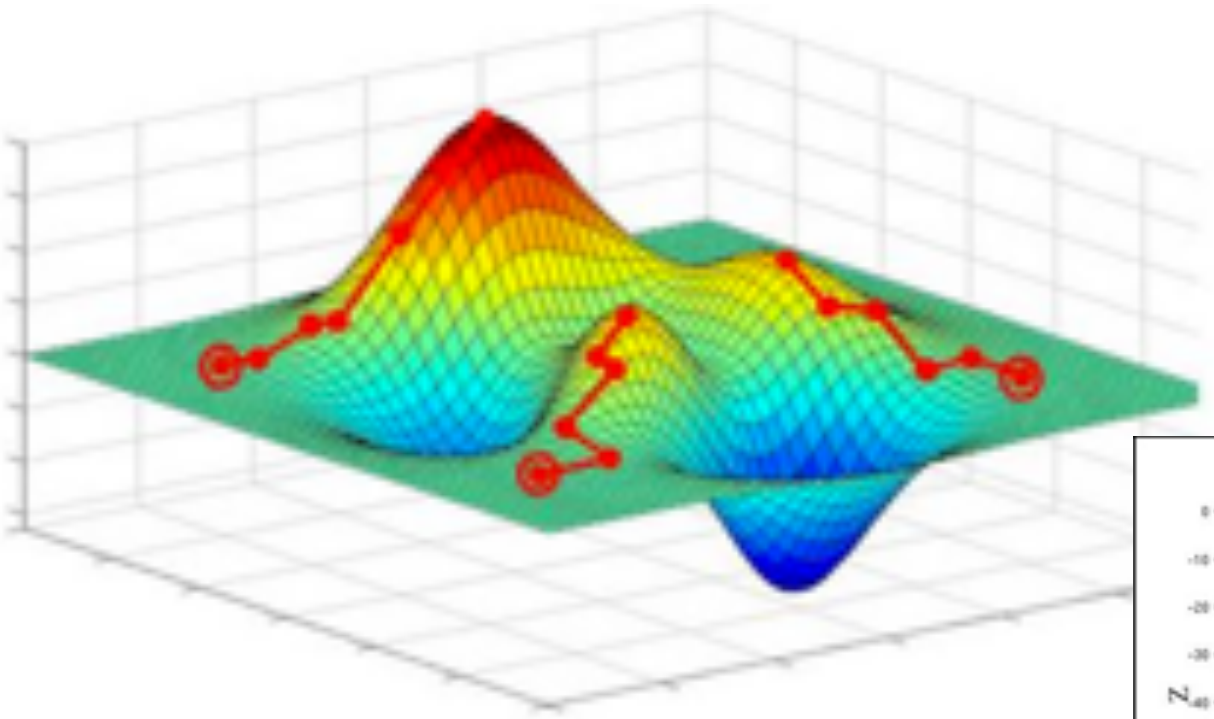
$$= \left[\frac{y}{\hat{y}} - \frac{1 - y}{1 - \hat{y}}\right] \hat{y}(1 - \hat{y}) x_j$$

Derive this one

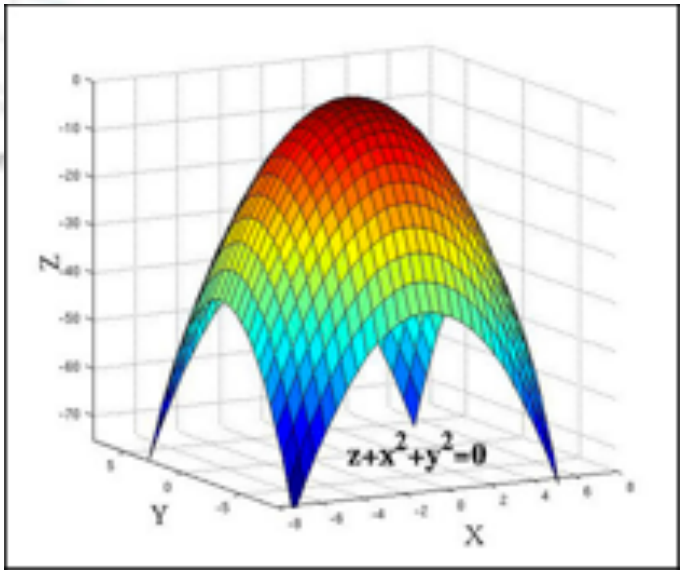
$$= (y - \hat{y}) x_j$$

Simplify

Gradient Ascent



Logistic regression
LL function is
convex



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

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

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

Logistic Regression Training

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

Repeat many times:

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

For each training example (x, y):

For each parameter j:

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

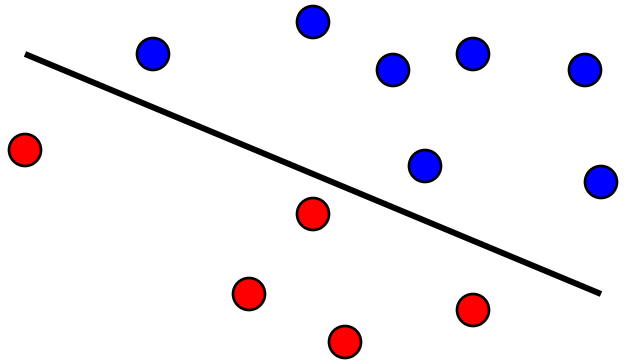
θ_j += η * gradient[j] for all $0 \le j \le m$

Classification with Logistic Regression

- Training: determine parameters θ_j (for all $0 \leq j \leq m$)
  - 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

Discrimination Intuition

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



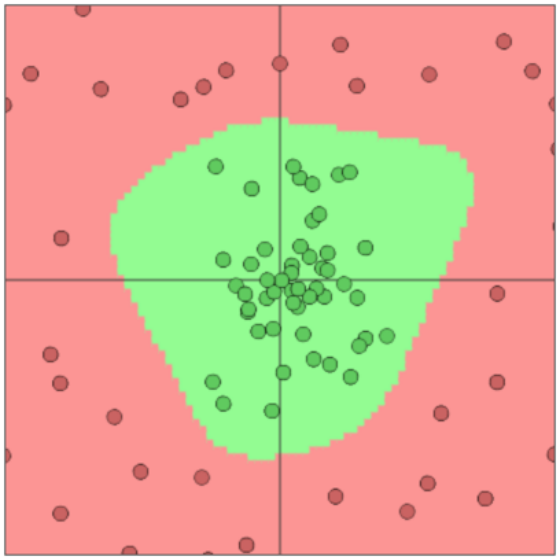
$$\theta^T \mathbf{x} = 0$$

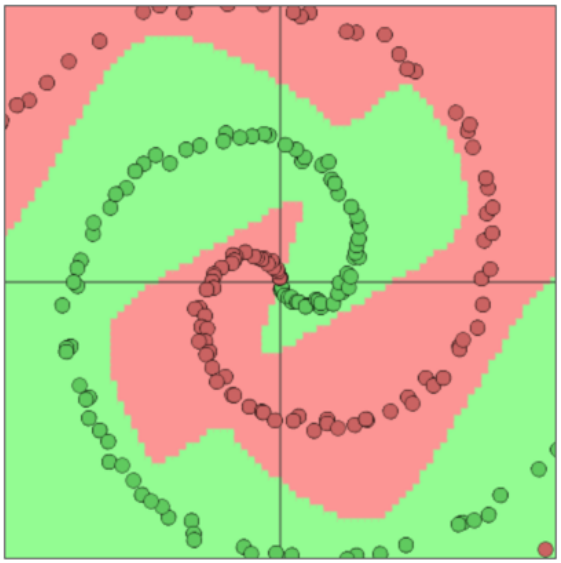
$$\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_m x_m = 0$$

- 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



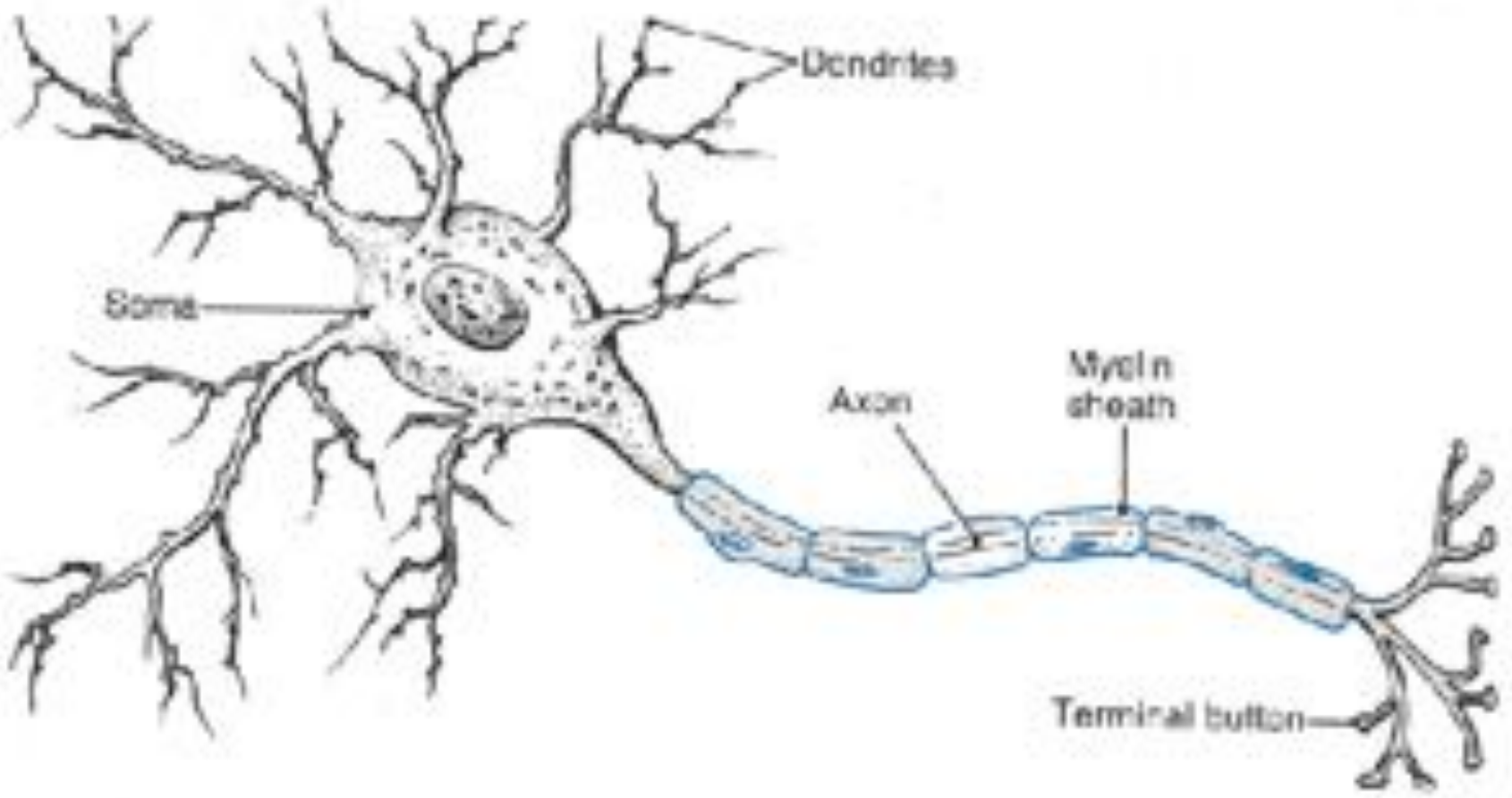


  - Not possible to draw a line that successfully separates all the $y = 1$ points (green) from the $y = 0$ points (red)
  - Despite this fact, logistic regression and Naive Bayes still often work well in practice

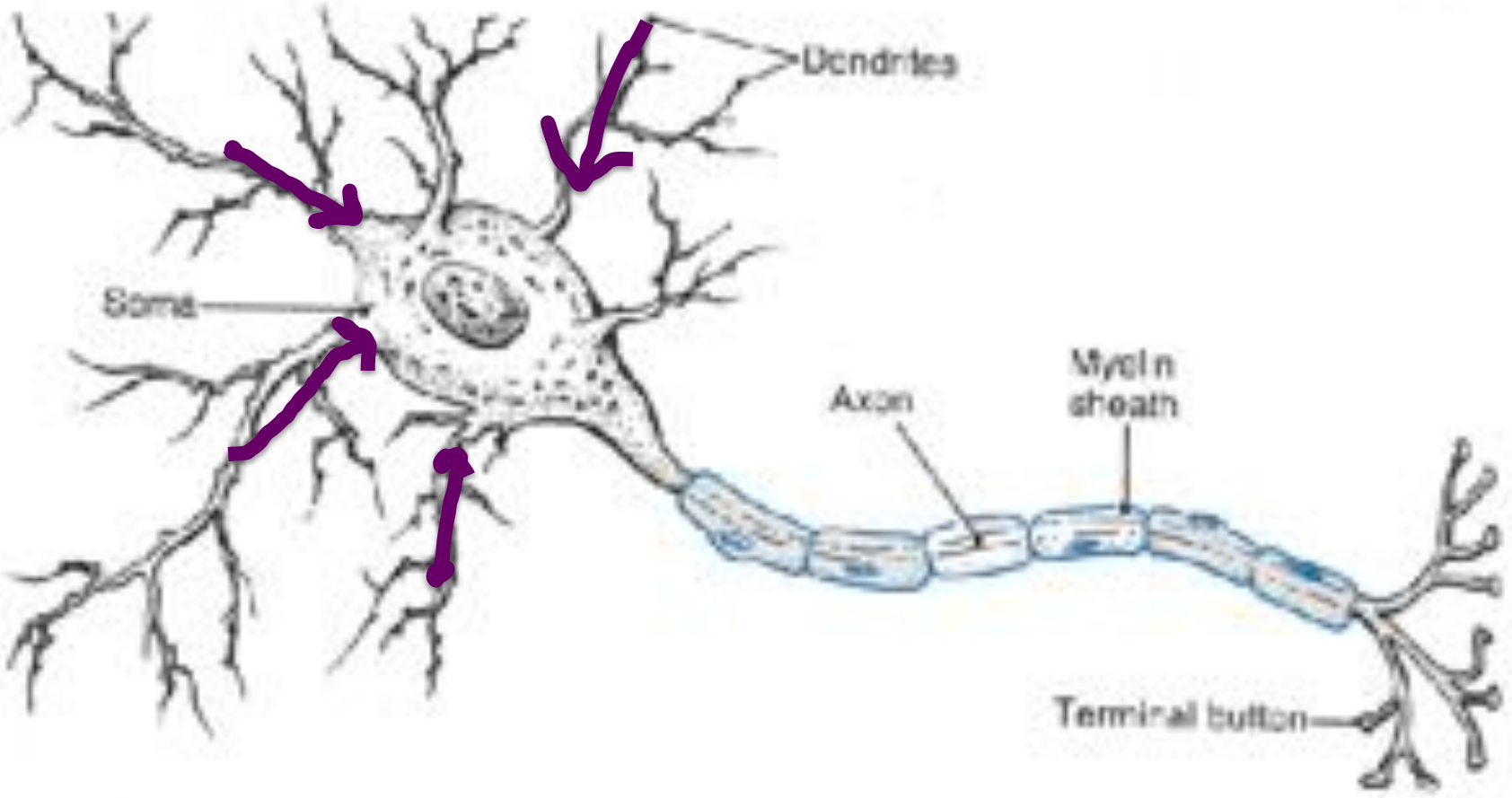
End Review

A Journey From Pure Math to Skin Cancer Detection

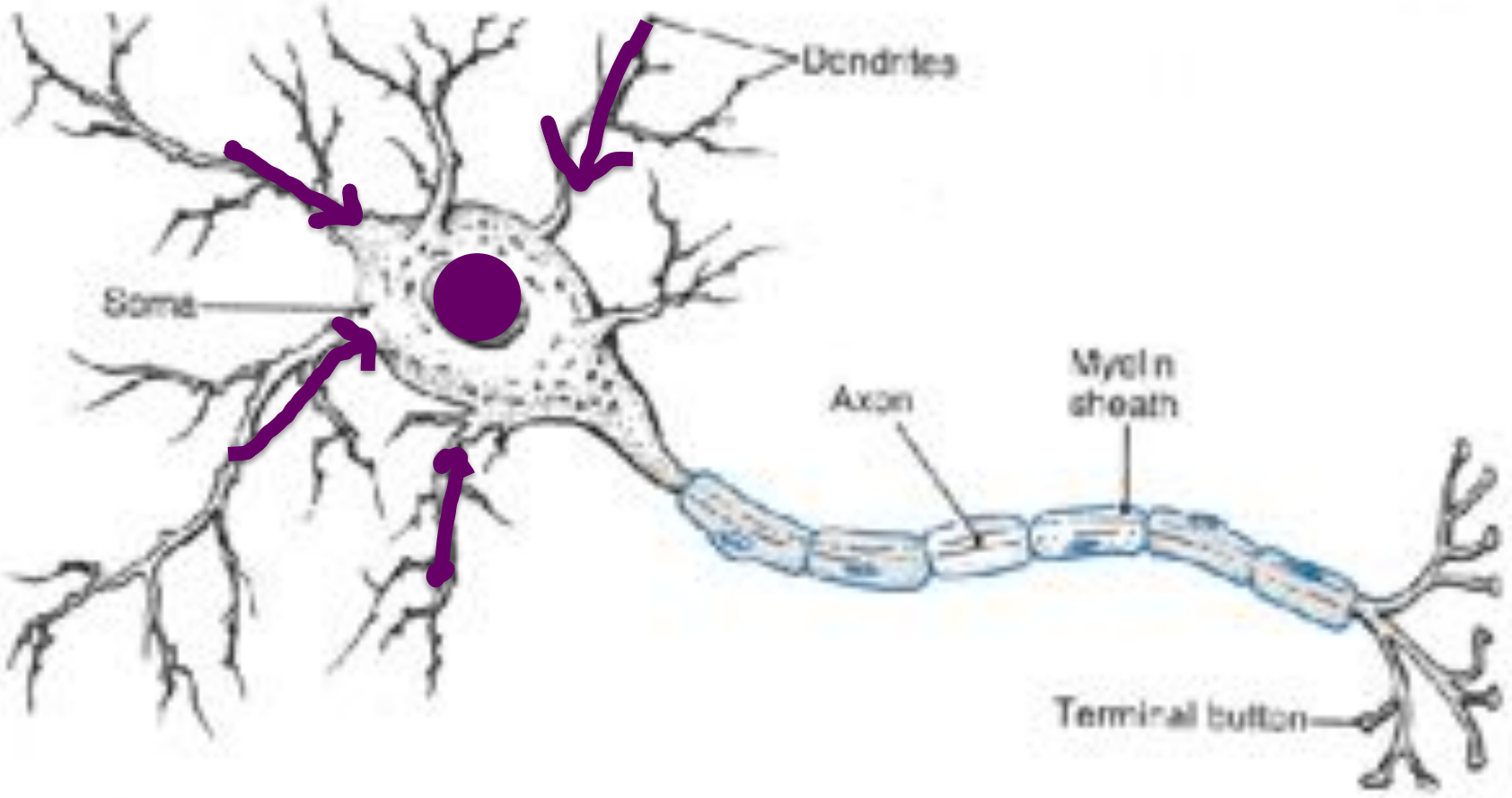
Neuron



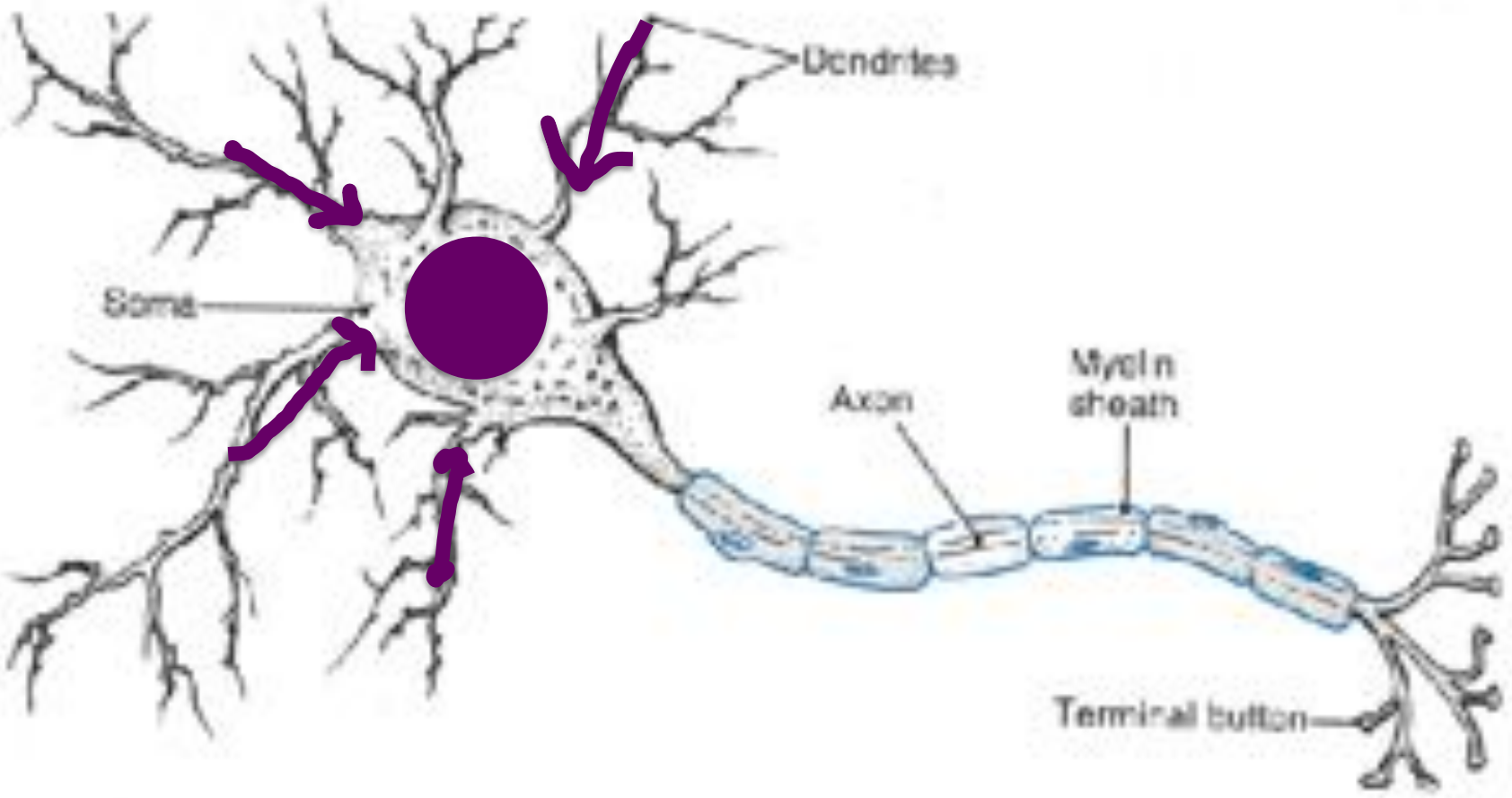
Neuron



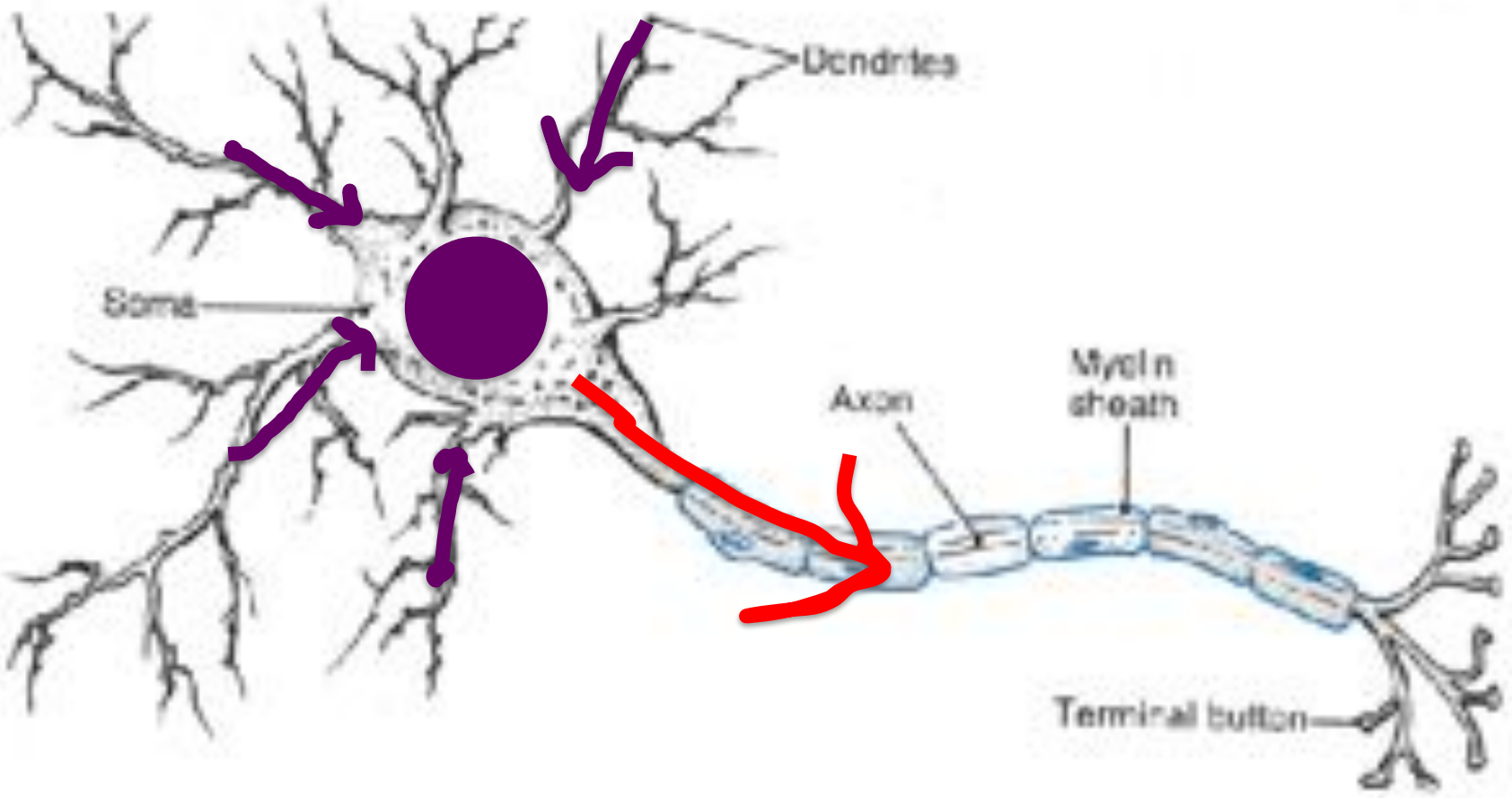
Neuron



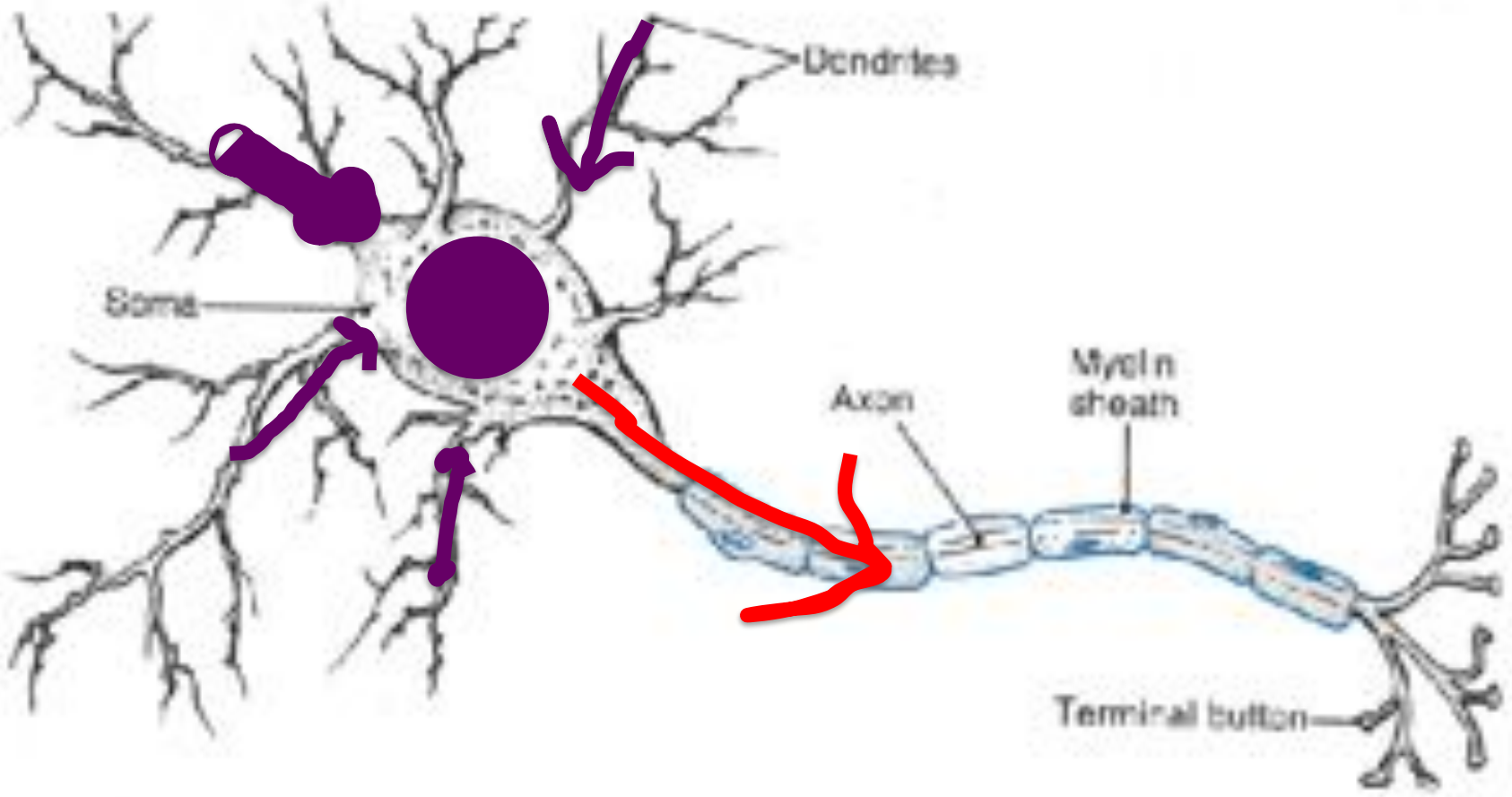
Neuron



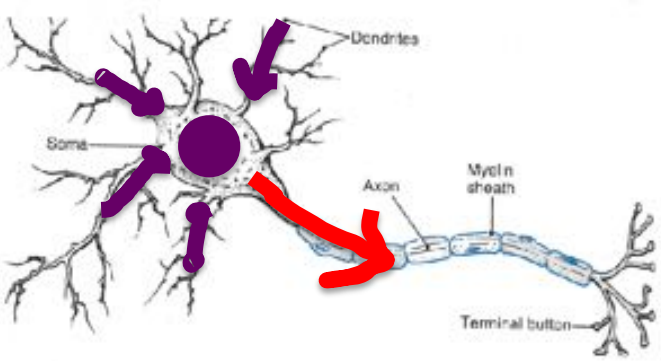
Neuron

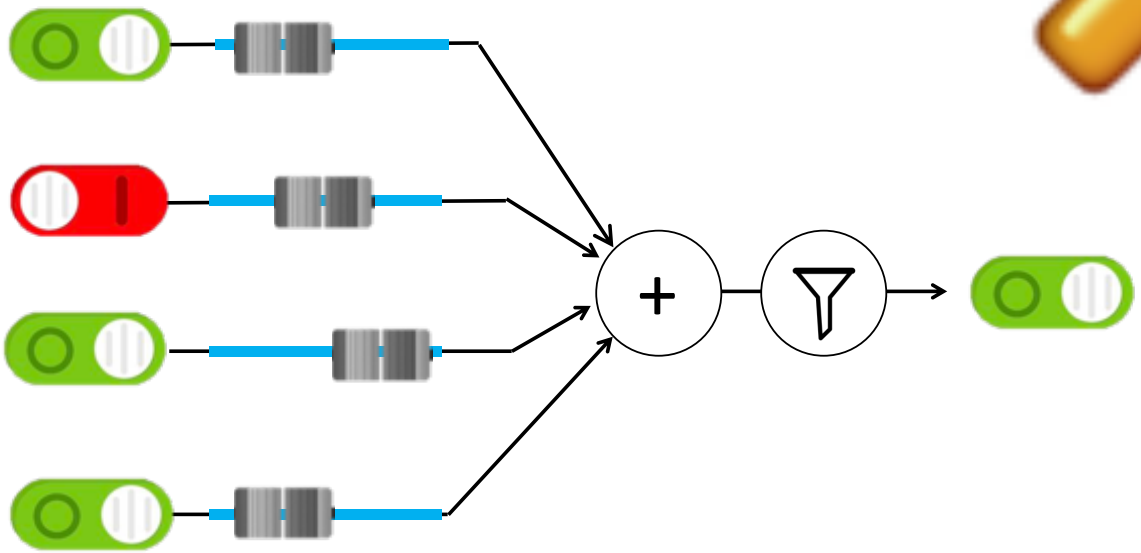


Some inputs are more important



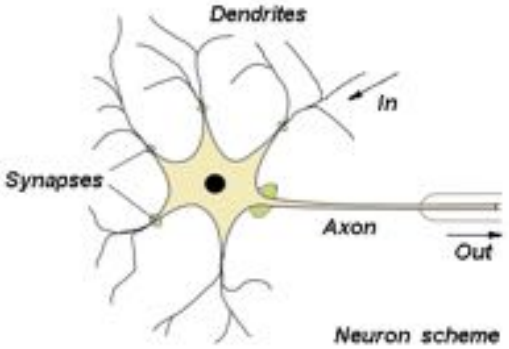
Artificial Neurons

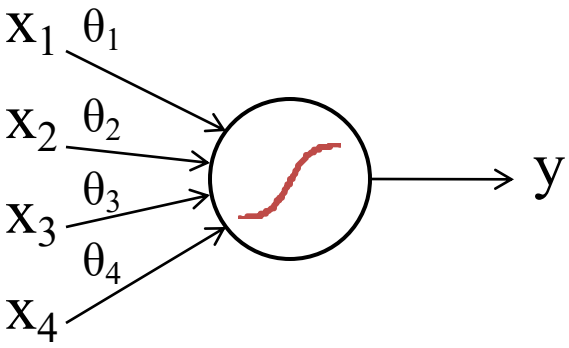




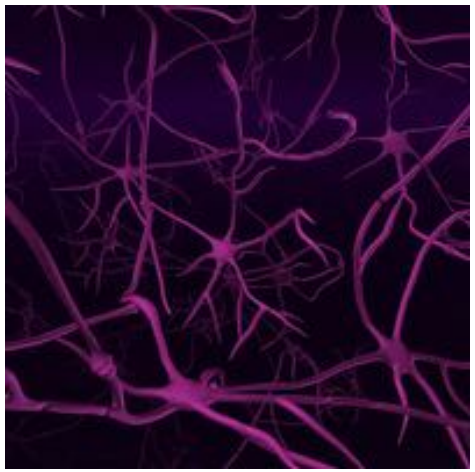
Biological Basis for Neural Networks

- A neuron

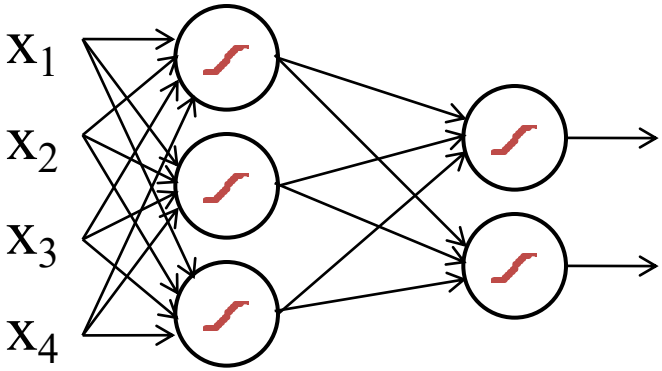




- Your brain



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.

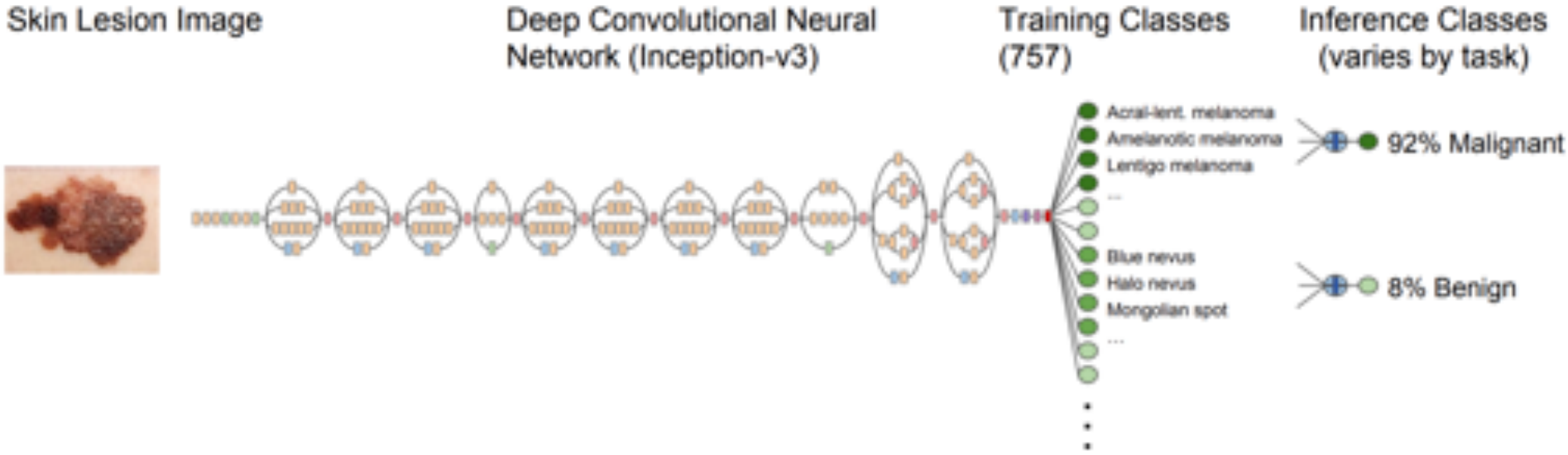
Core idea behind the revolution in AI

Alpha GO

Self Driving Cars

Computers Making Art

Detecting Skin Cancer



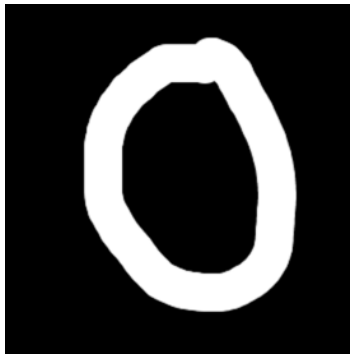
Esteva, Andre, et al. "Dermatologist-level classification of skin cancer with deep neural networks." *Nature* 542.7639 (2017): 115-118.

(aka Neural Networks)

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

Digit Recognition Example

Let's make feature vectors from pictures of numbers

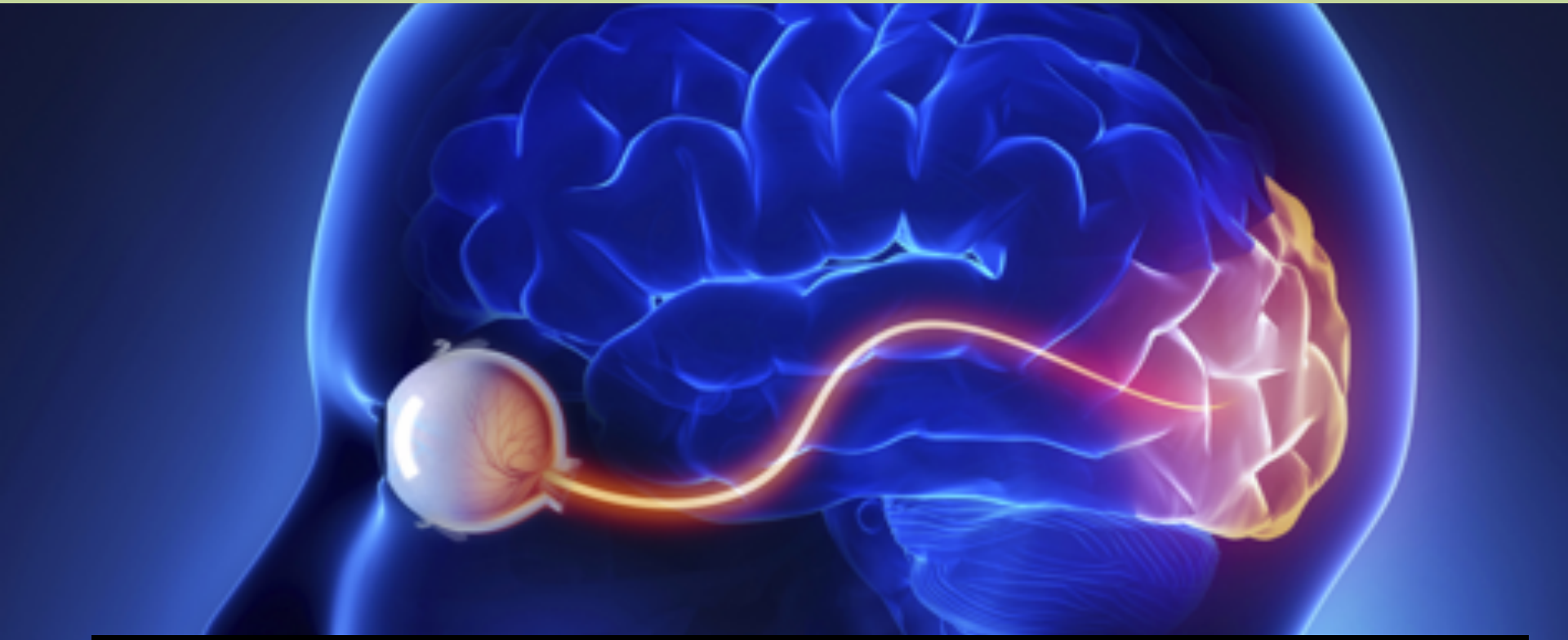


$$\mathbf{x}^{(i)} = [0, 0, 0, 0, \ldots, 1, 0, 0, 1, \ldots 0, 0, 1, 0]$$
$$y^{(i)} = 0$$

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

Computer Vision

Vision in your Brain

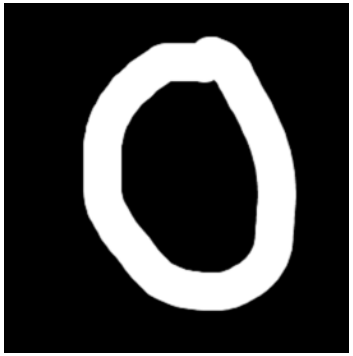


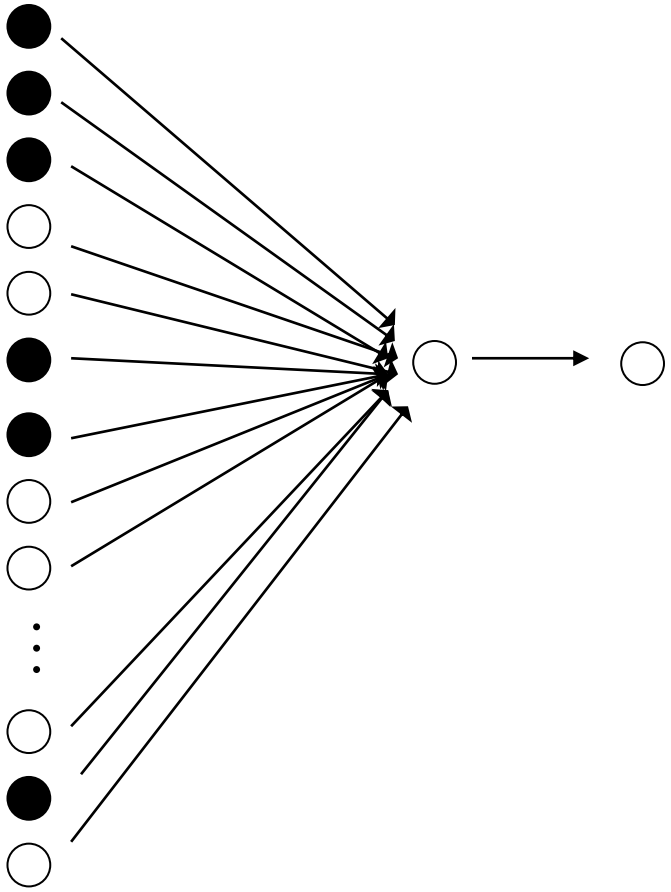
Hundreds of millions of neurons [1]

Visual neurons make up up 30% of your cortex [1]

[1] http://discovermagazine.com/1993/jun/thevisionthingma227

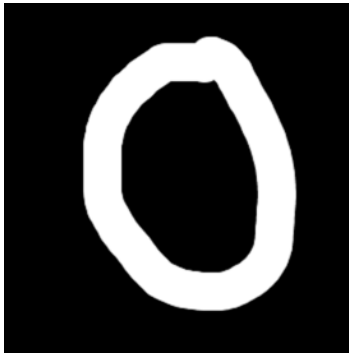
Logistic Regression

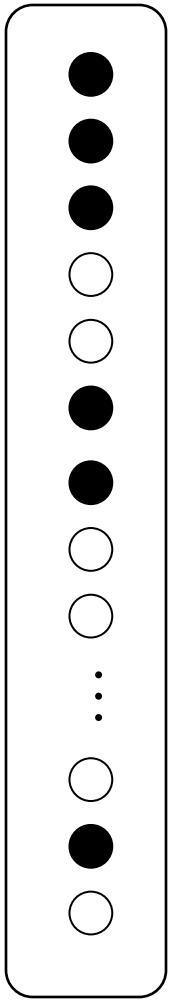




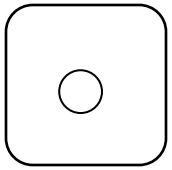
This means it predicts a 0

Logistic Regression



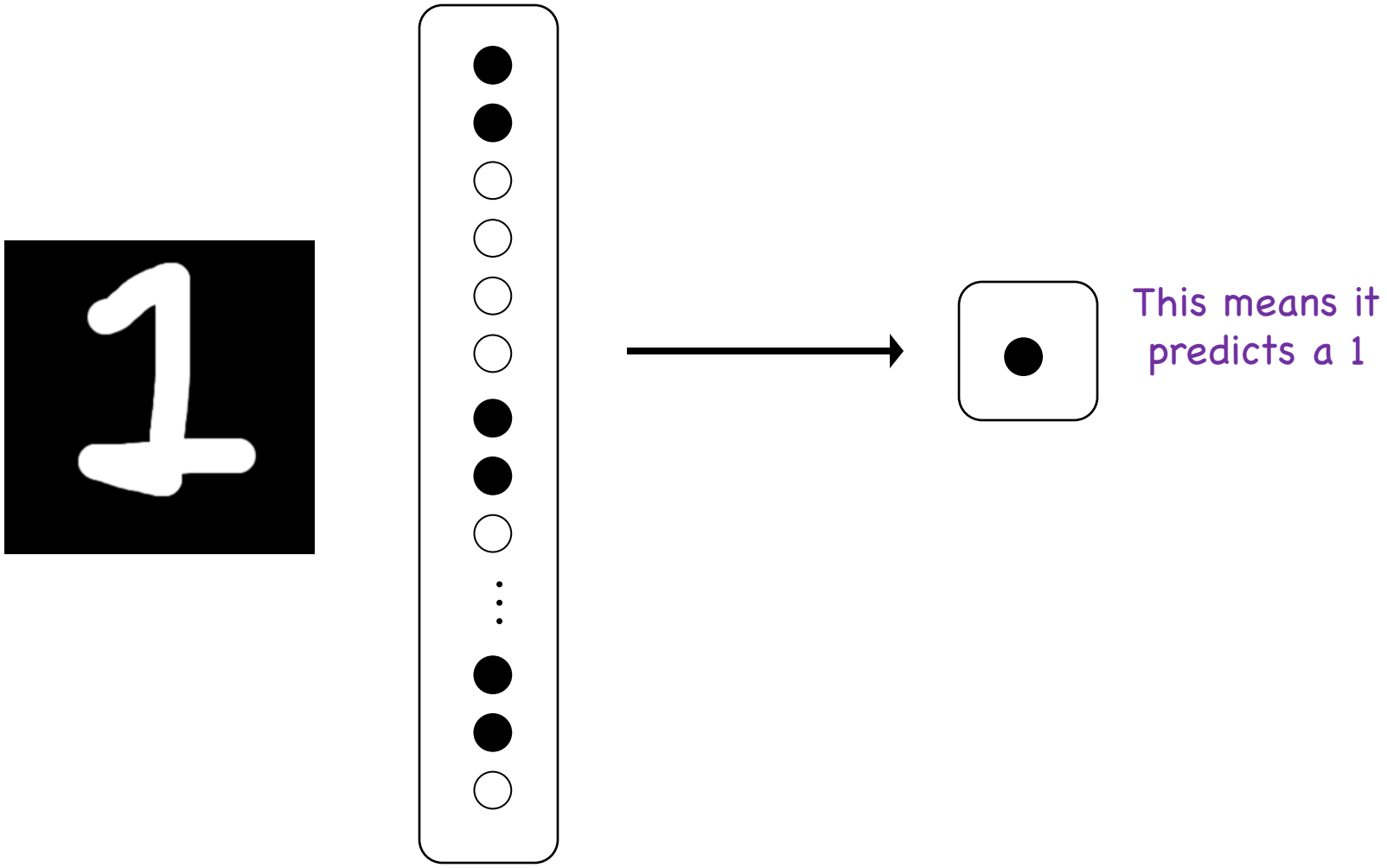


Indicates logistic regression connection

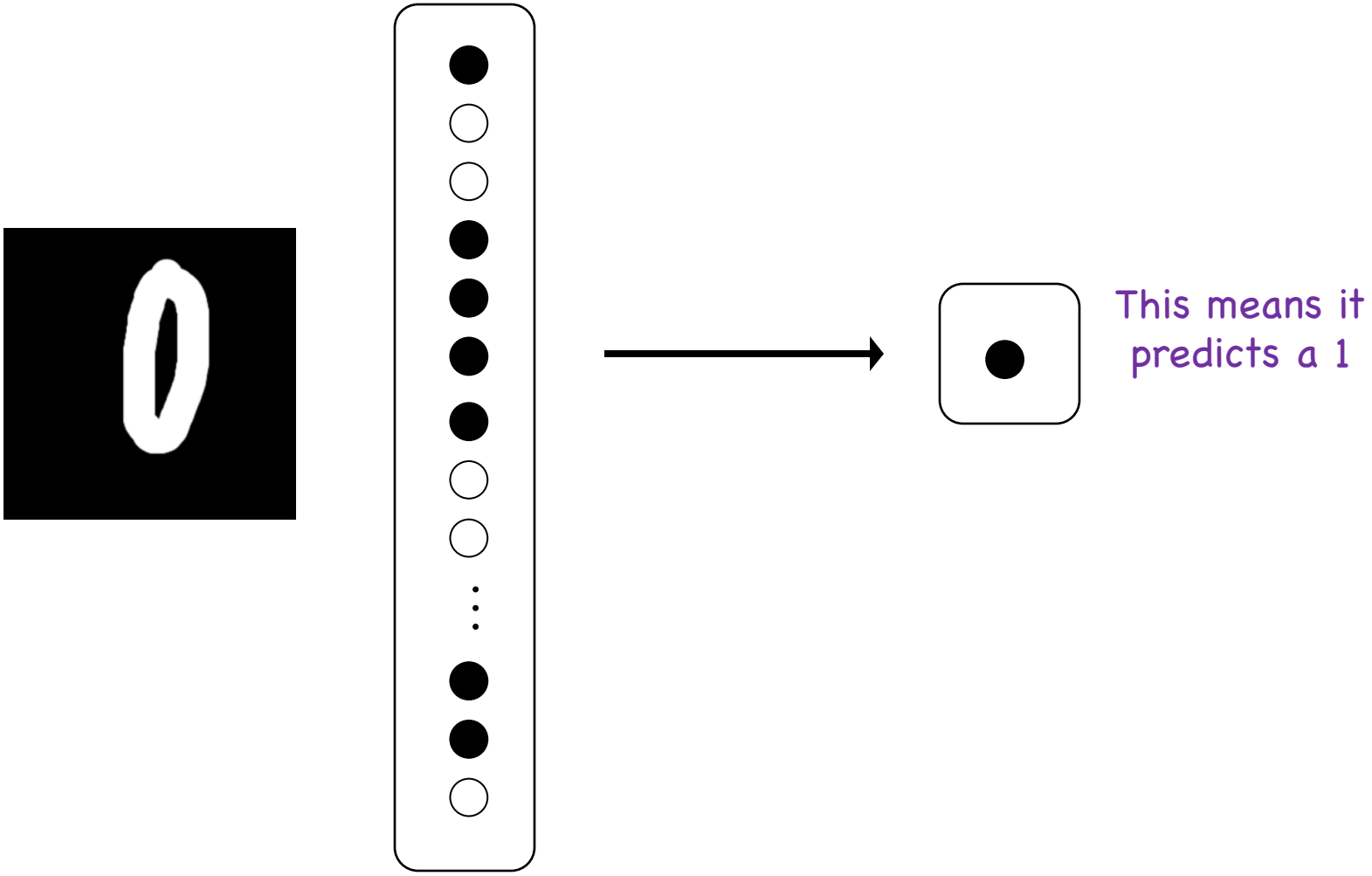


This means it predicts a 0

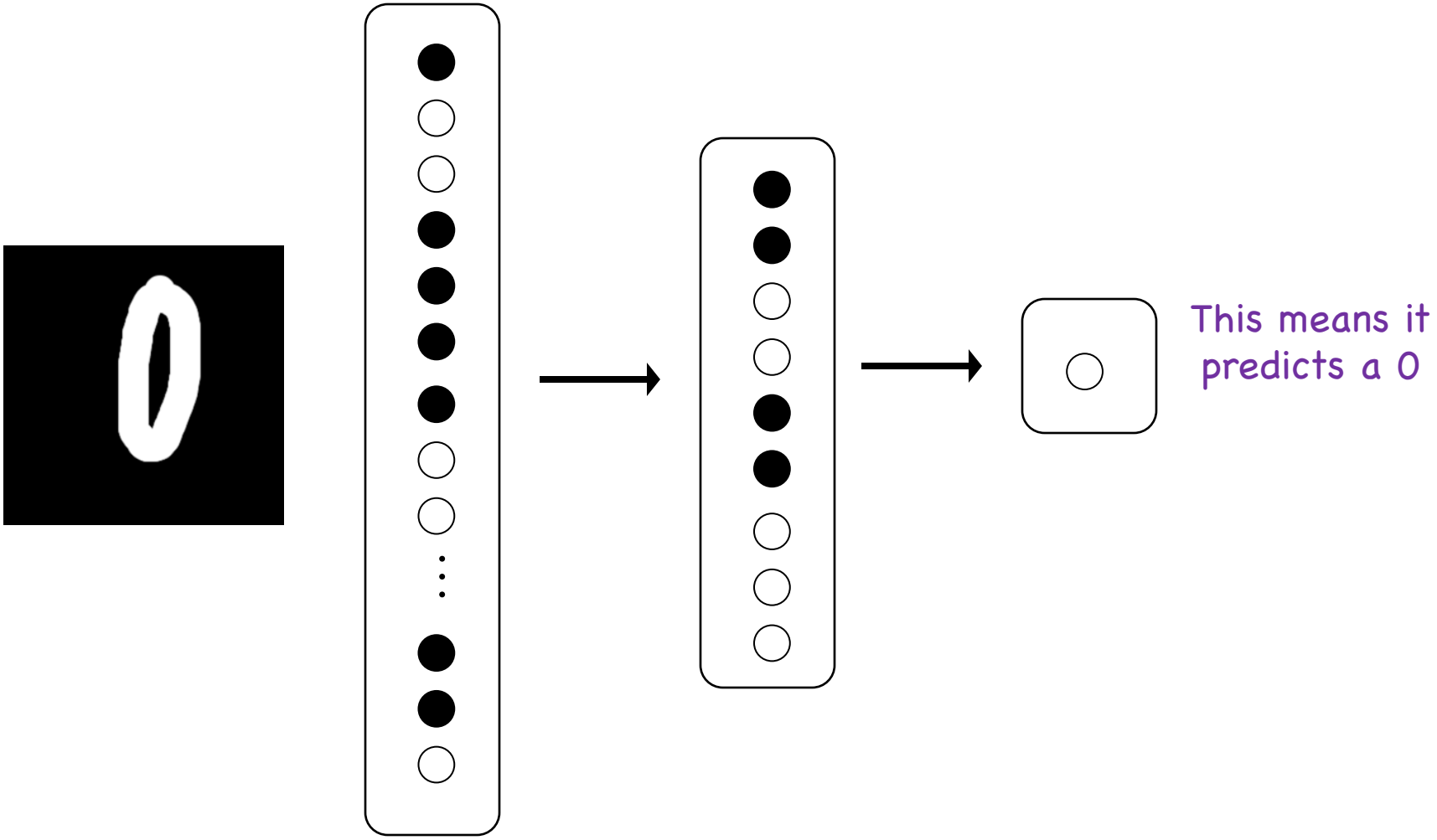
Logistic Regression



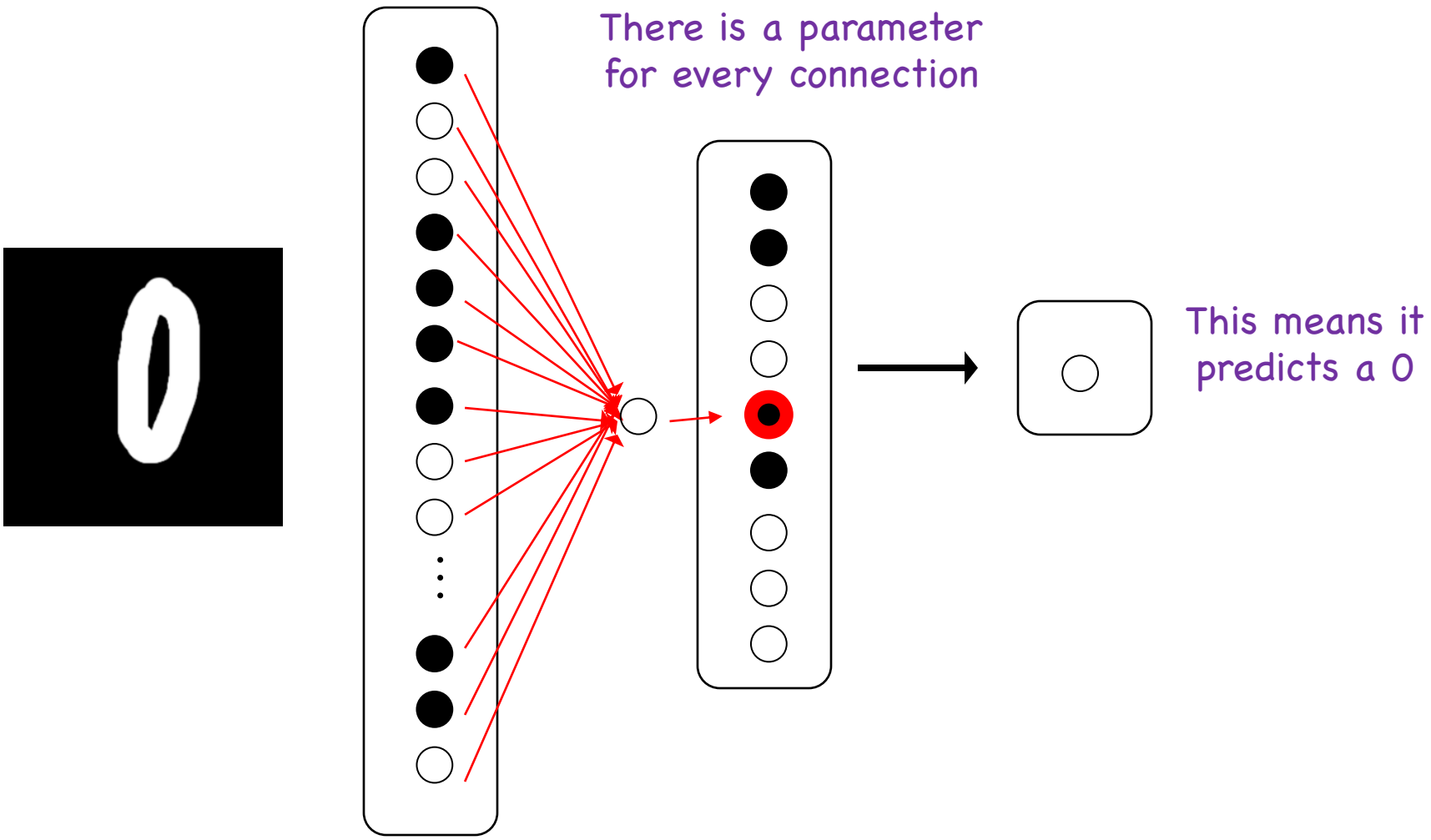
Not So Good



We Can Put Neurons Together

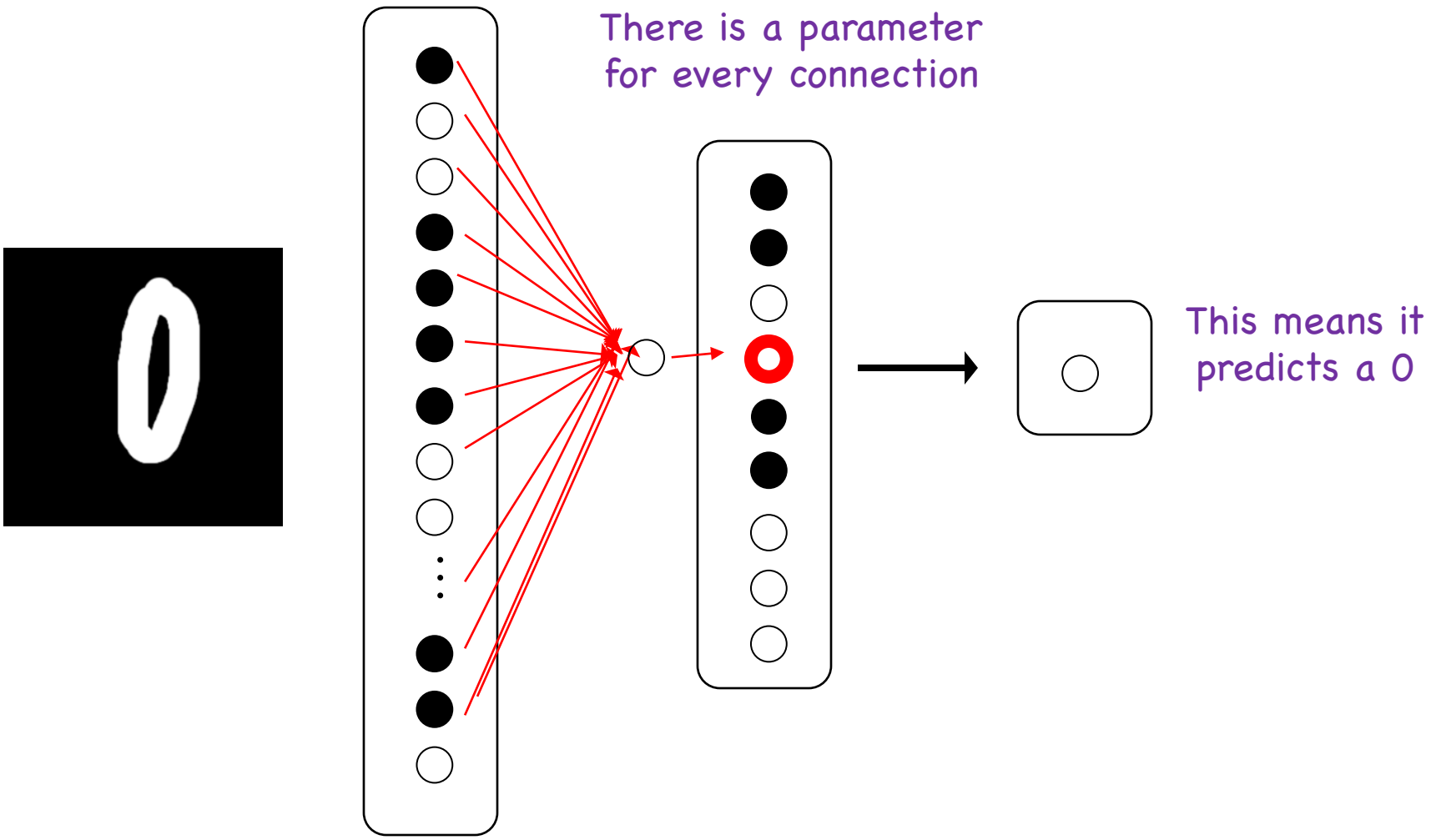


We Can Put Neurons Together



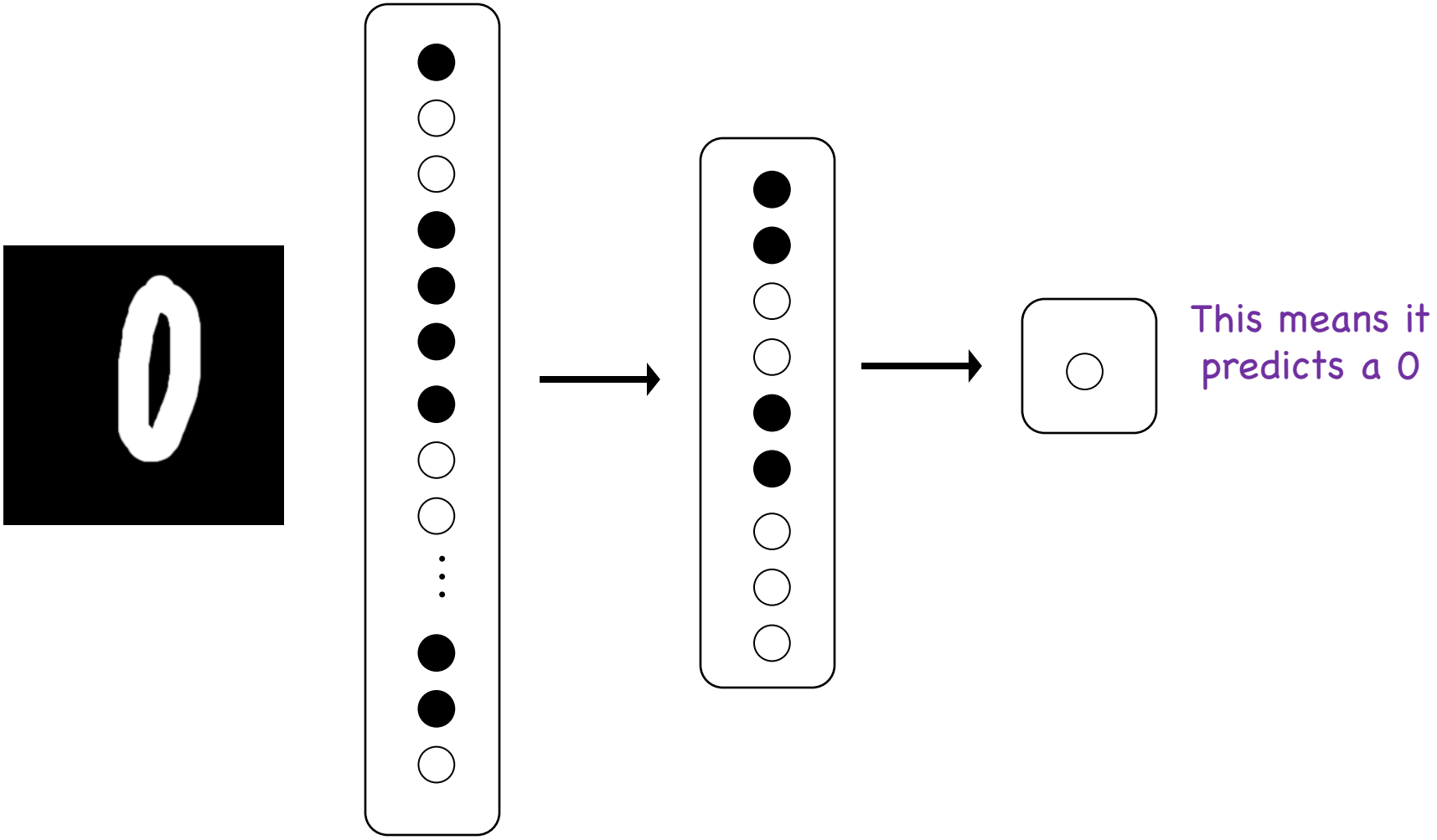
Look at a single “hidden” neuron

We Can Put Neurons Together

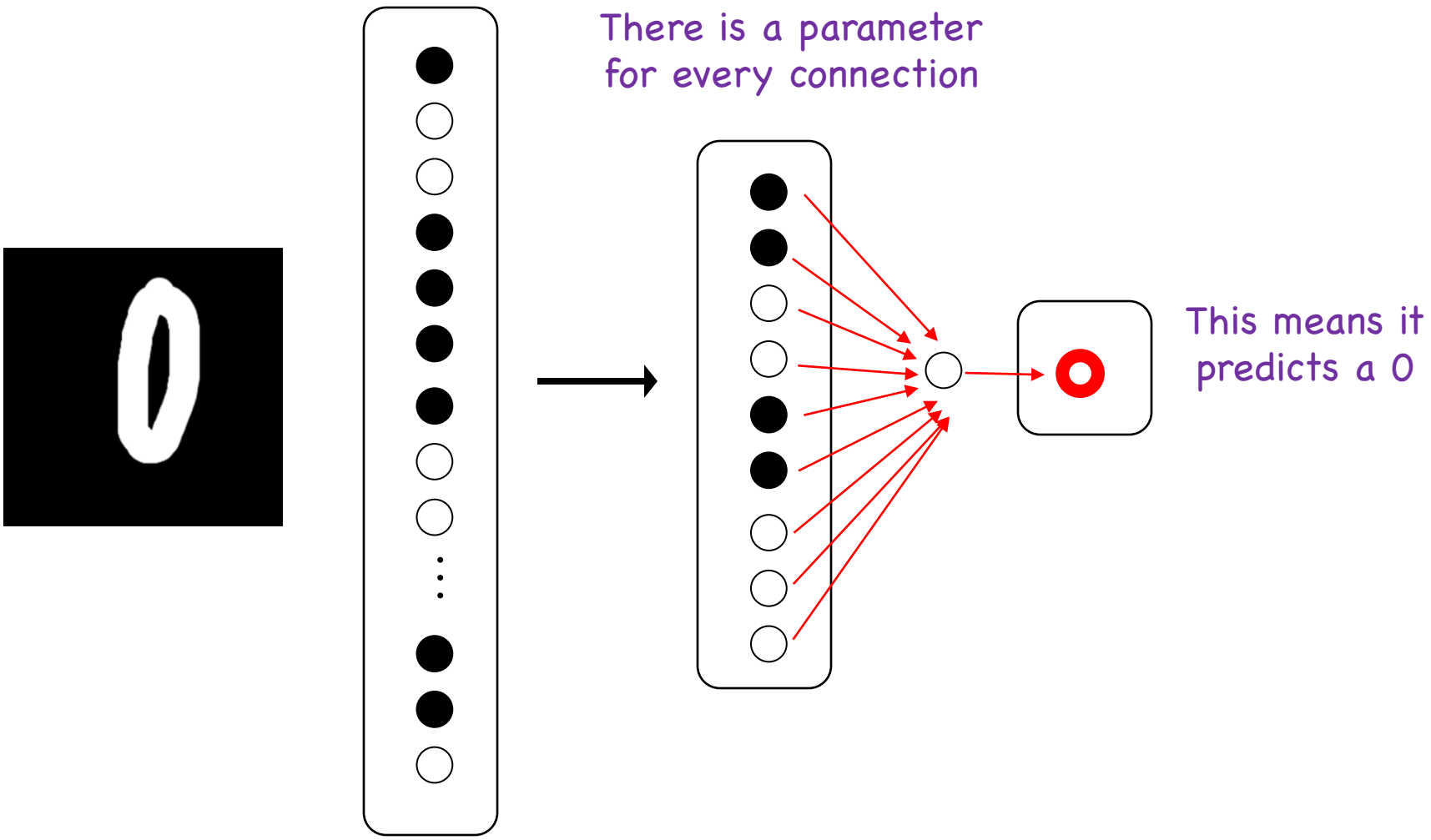


Look at another “hidden” neuron

We Can Put Neurons Together

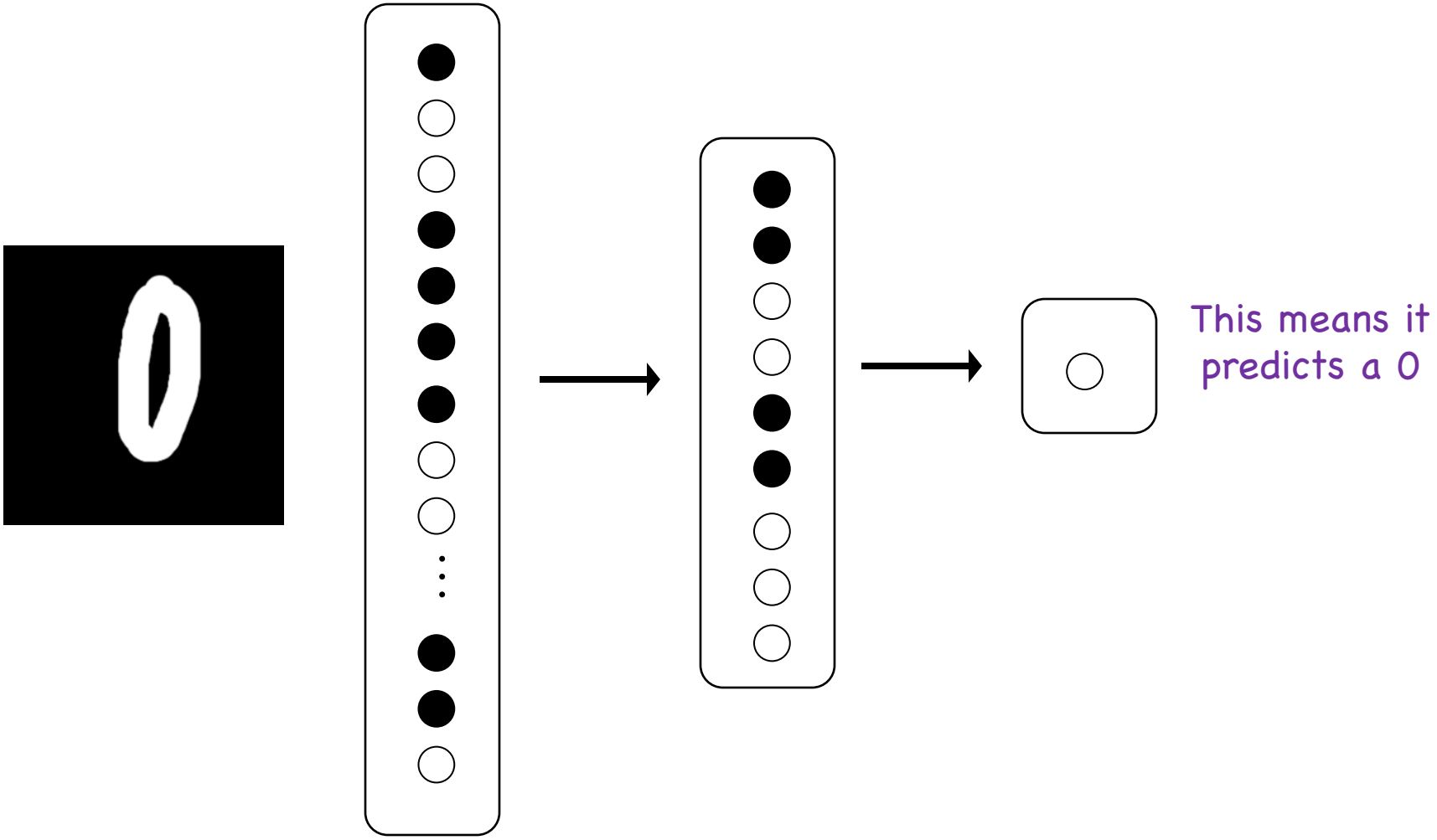


We Can Put Neurons Together

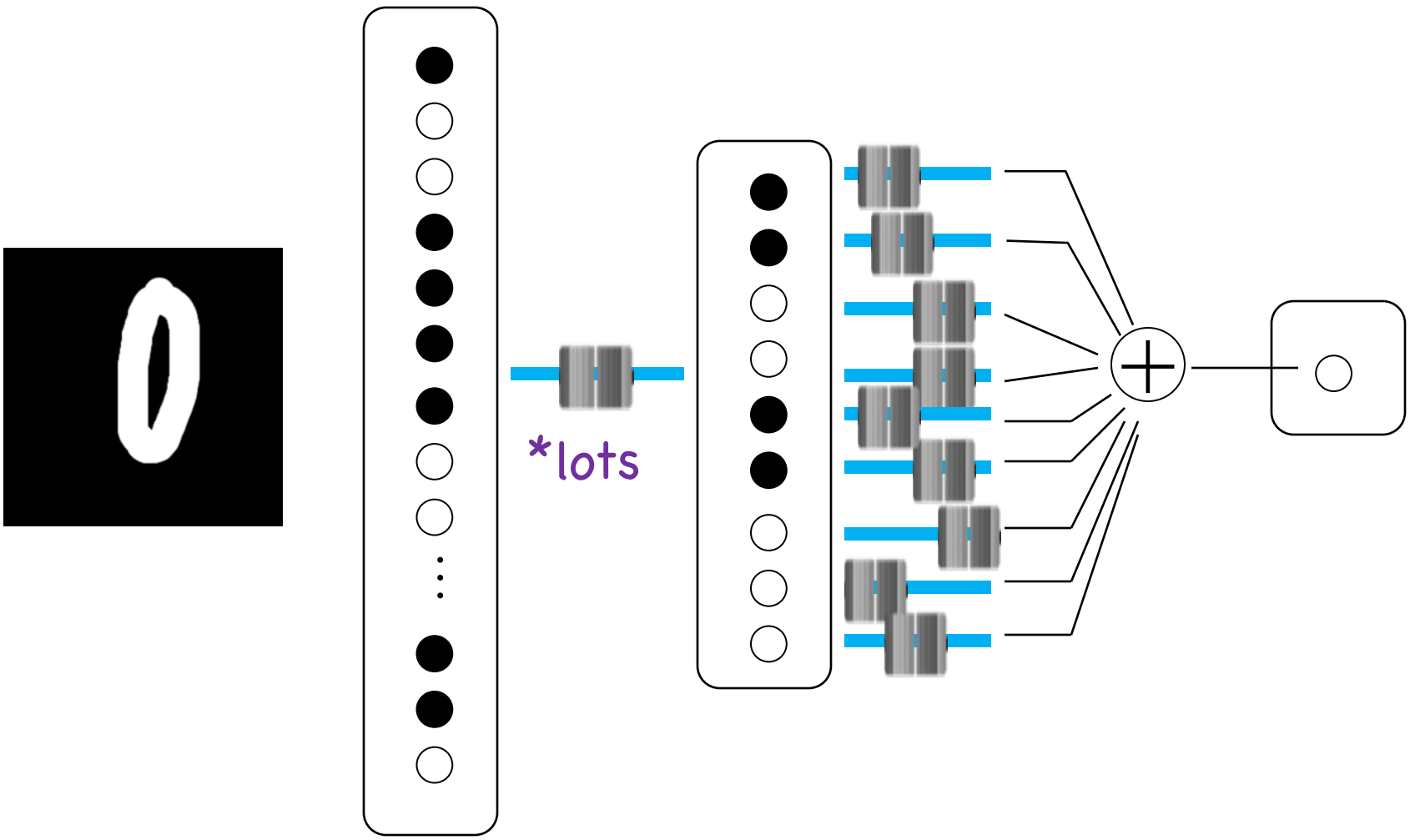


Look at another neuron

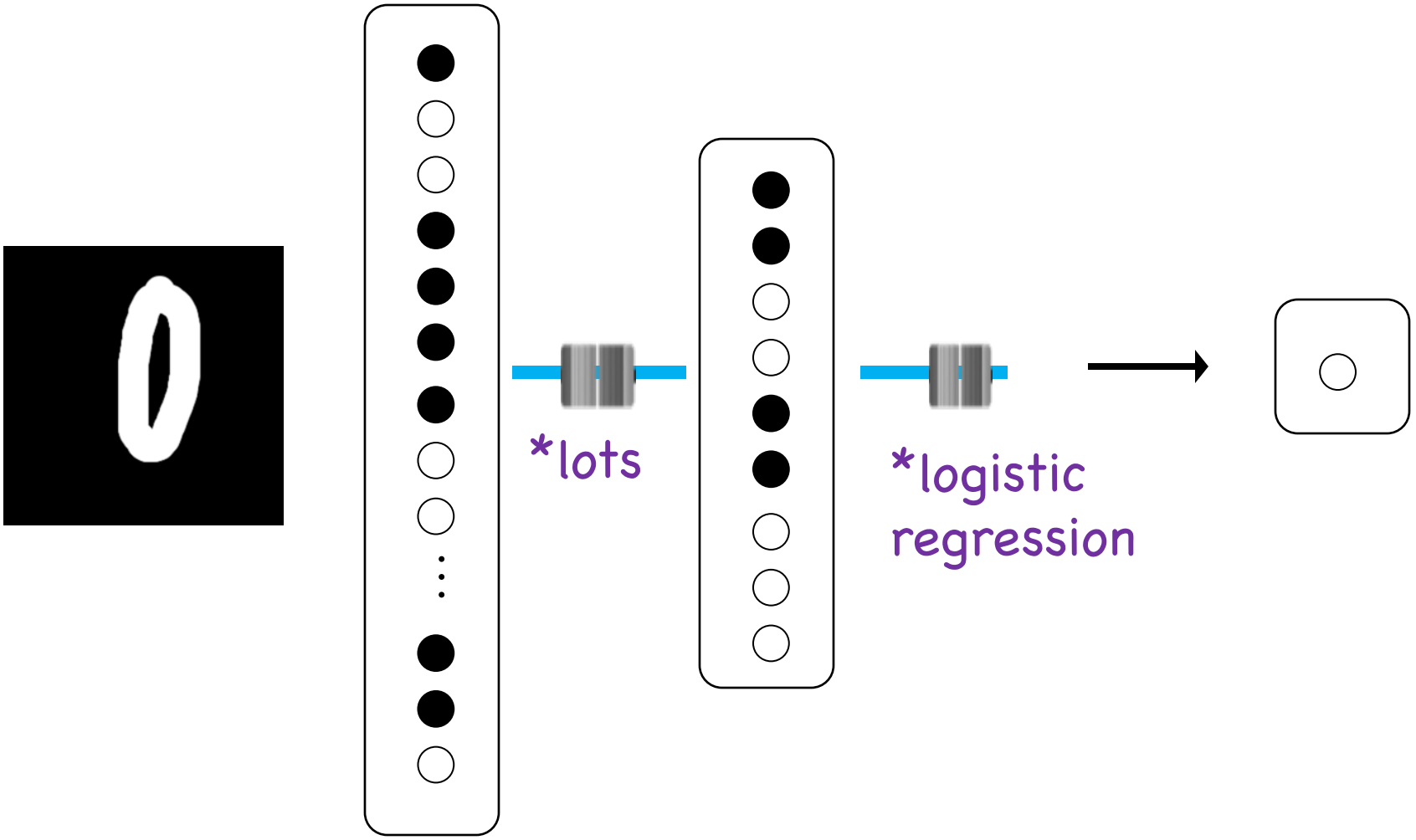
We Can Put Neurons Together



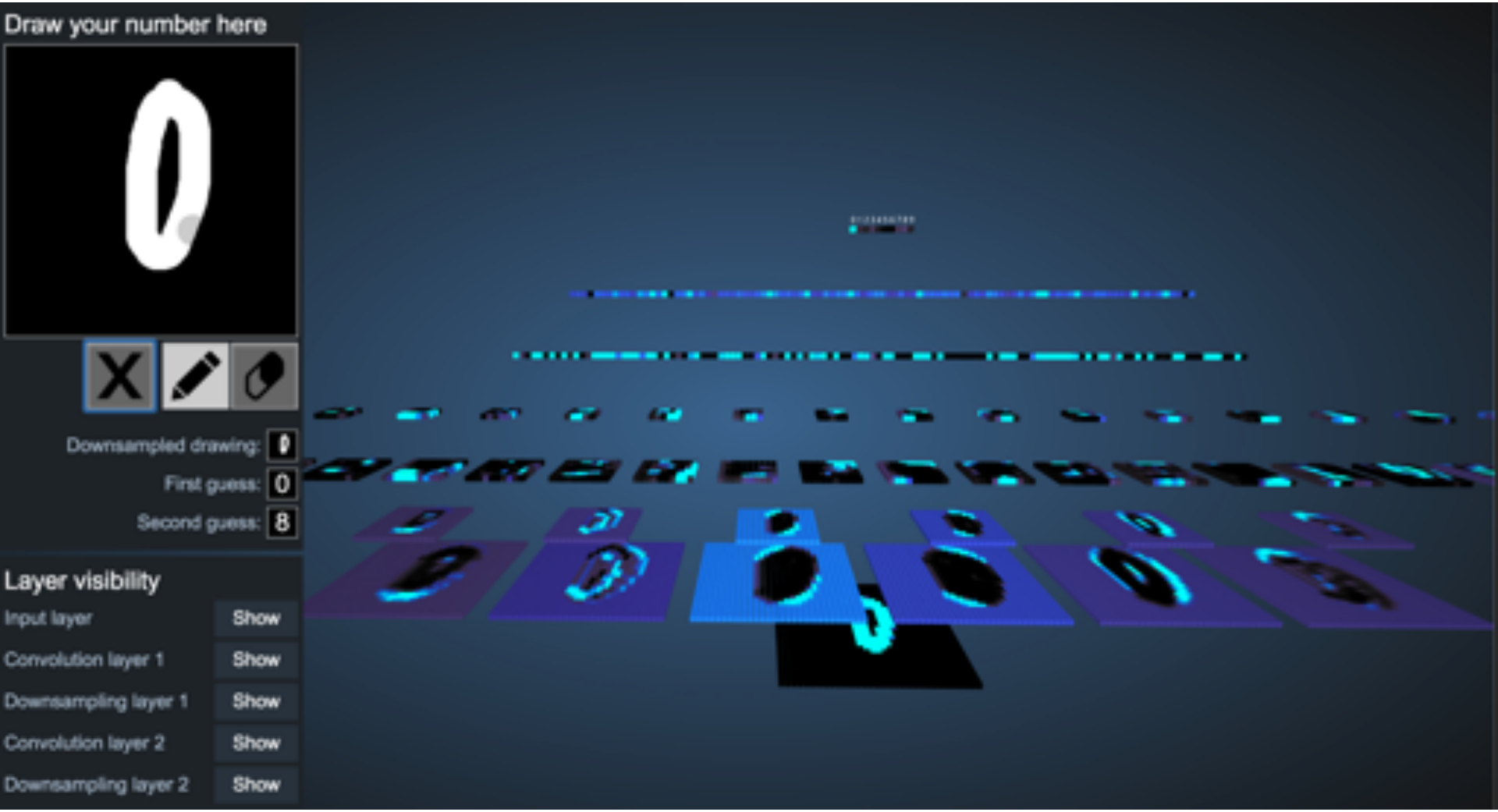
We Can Put Neurons Together



We Can Put Neurons Together



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!

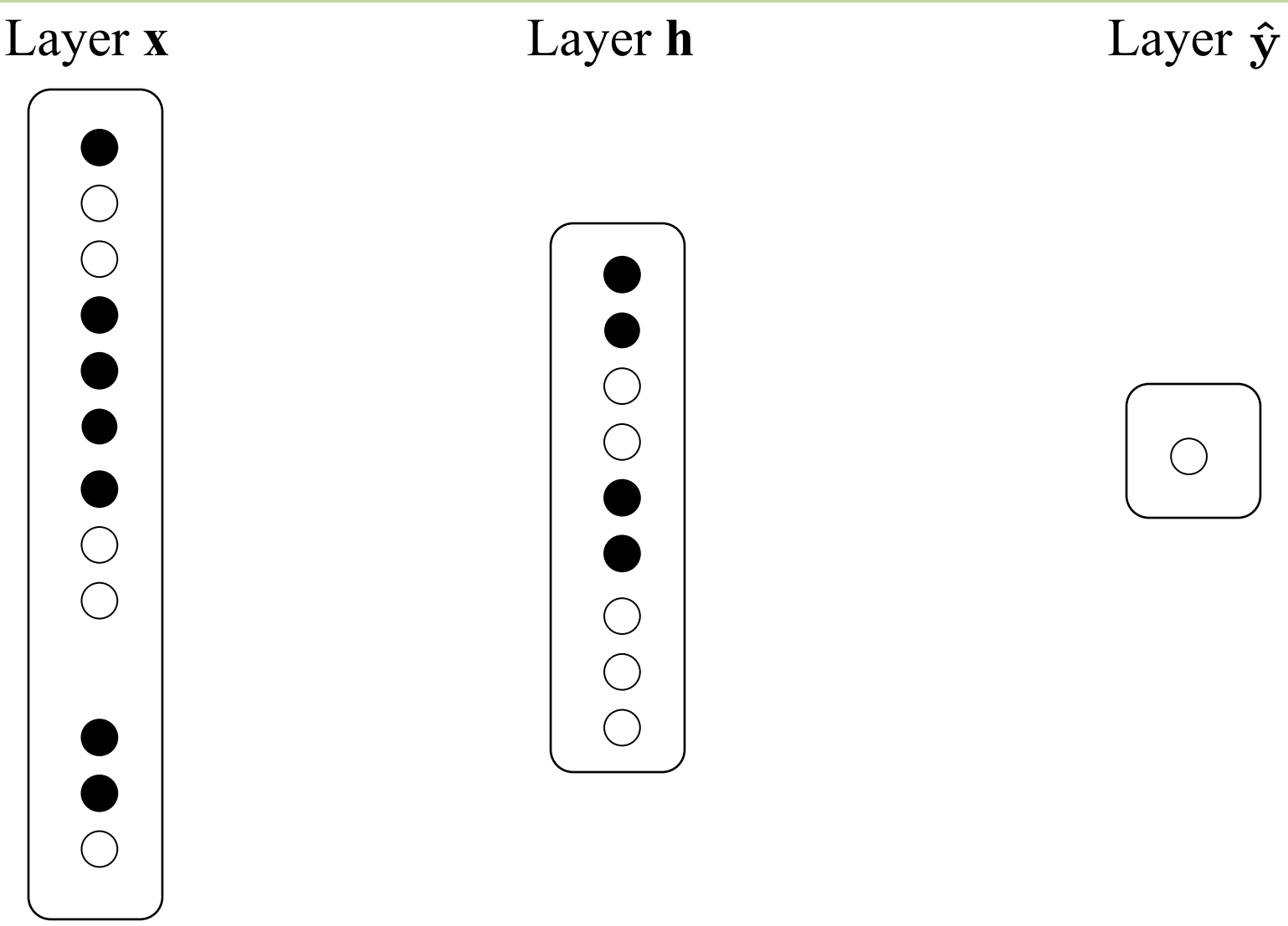
First: Learning Goals…

1. Understand Chain Rule as ♡ of Deep Learning

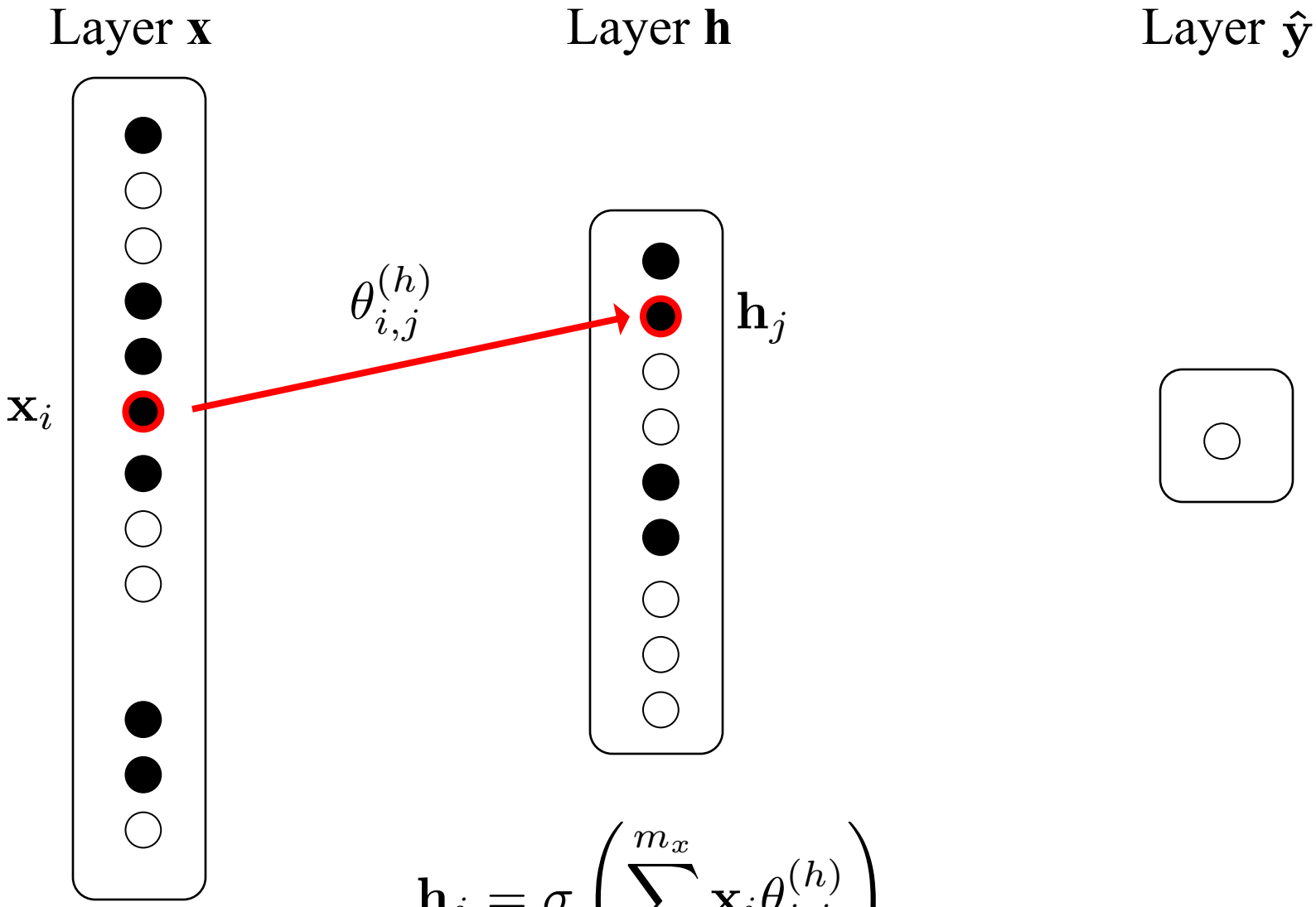
2. Everyone should be able to do simple derivations

Math worth knowing:

New Notation

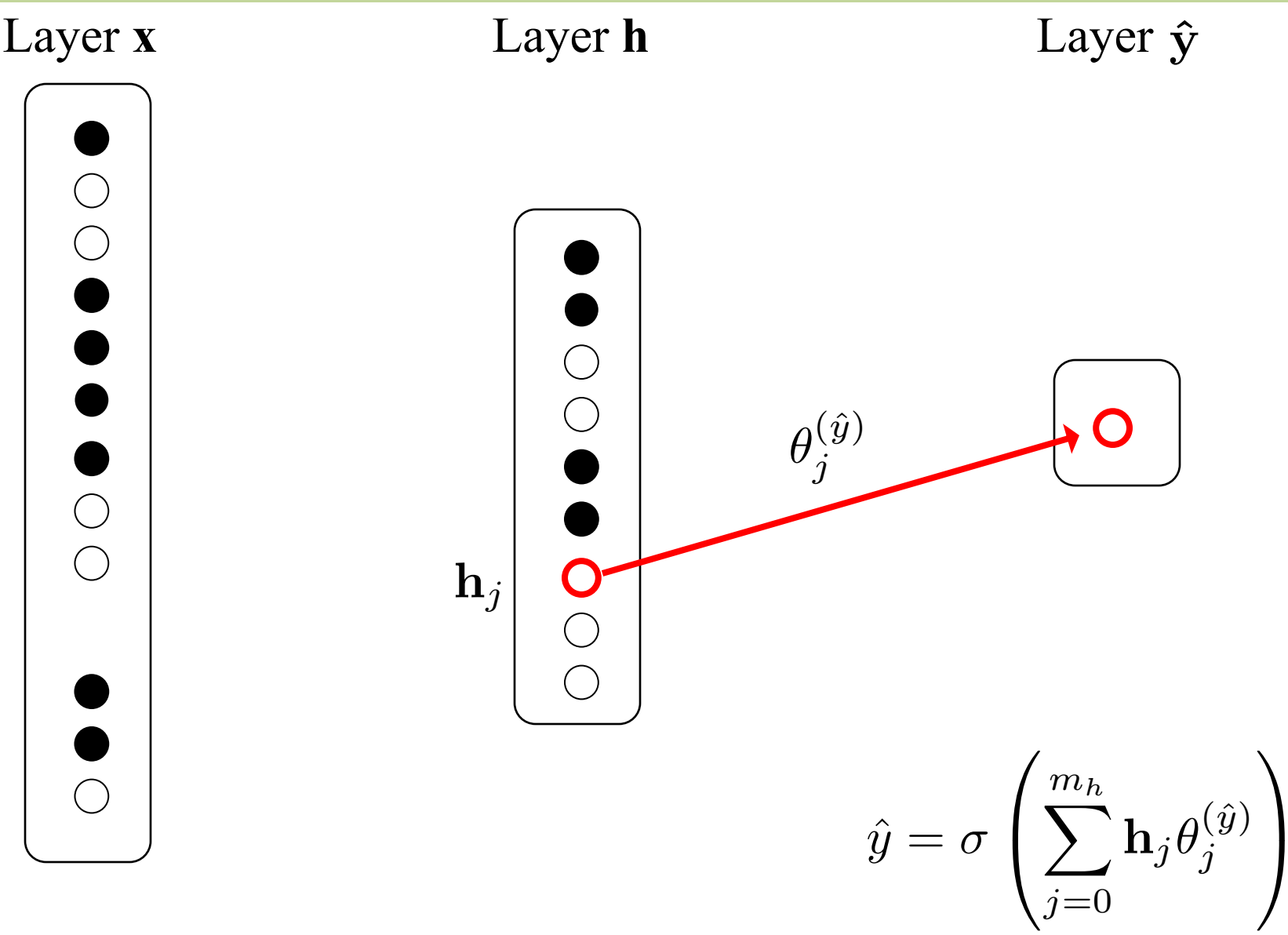


New Notation



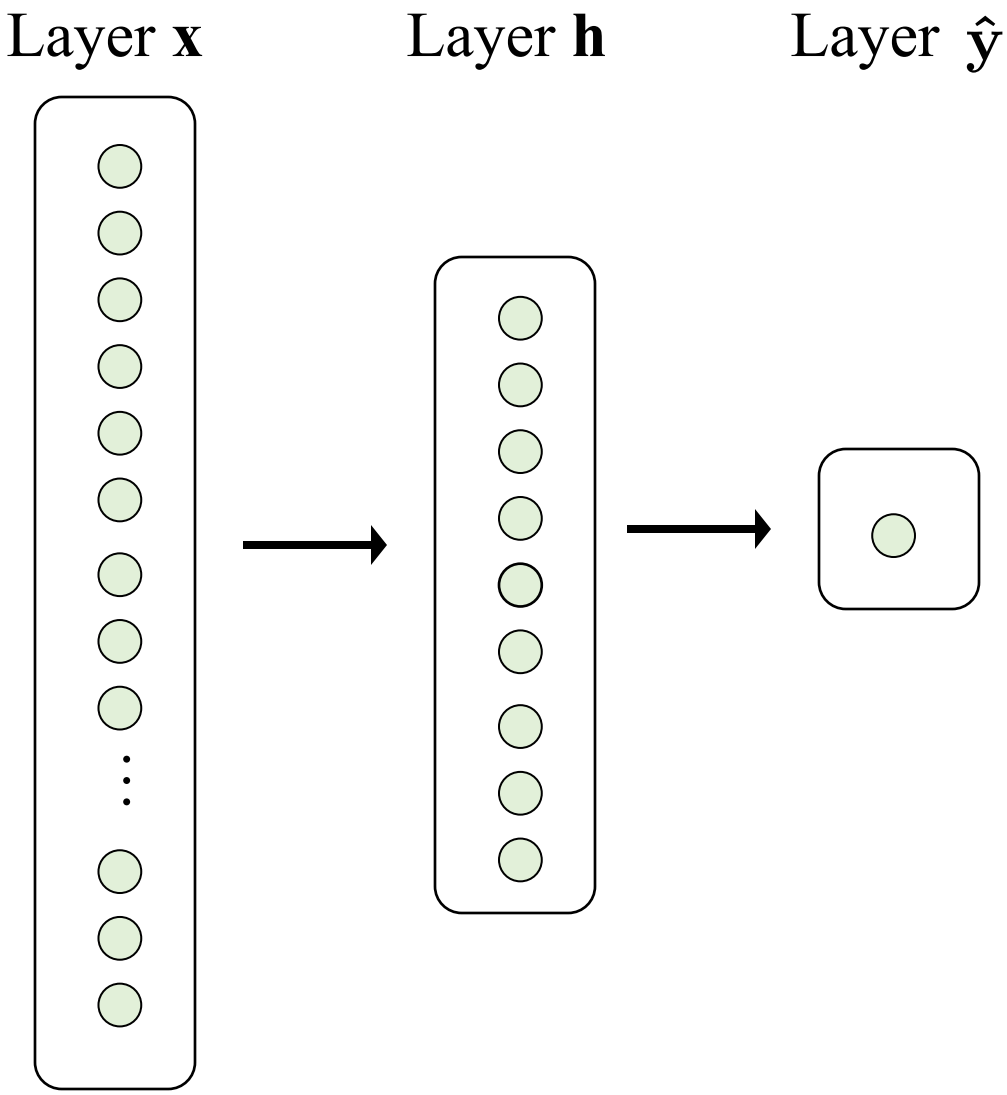
$$\mathbf{h}_j = \sigma\left(\sum_{i=0}^{m_x} \mathbf{x}_i \theta_{i,j}^{(h)}\right)$$

New Notation

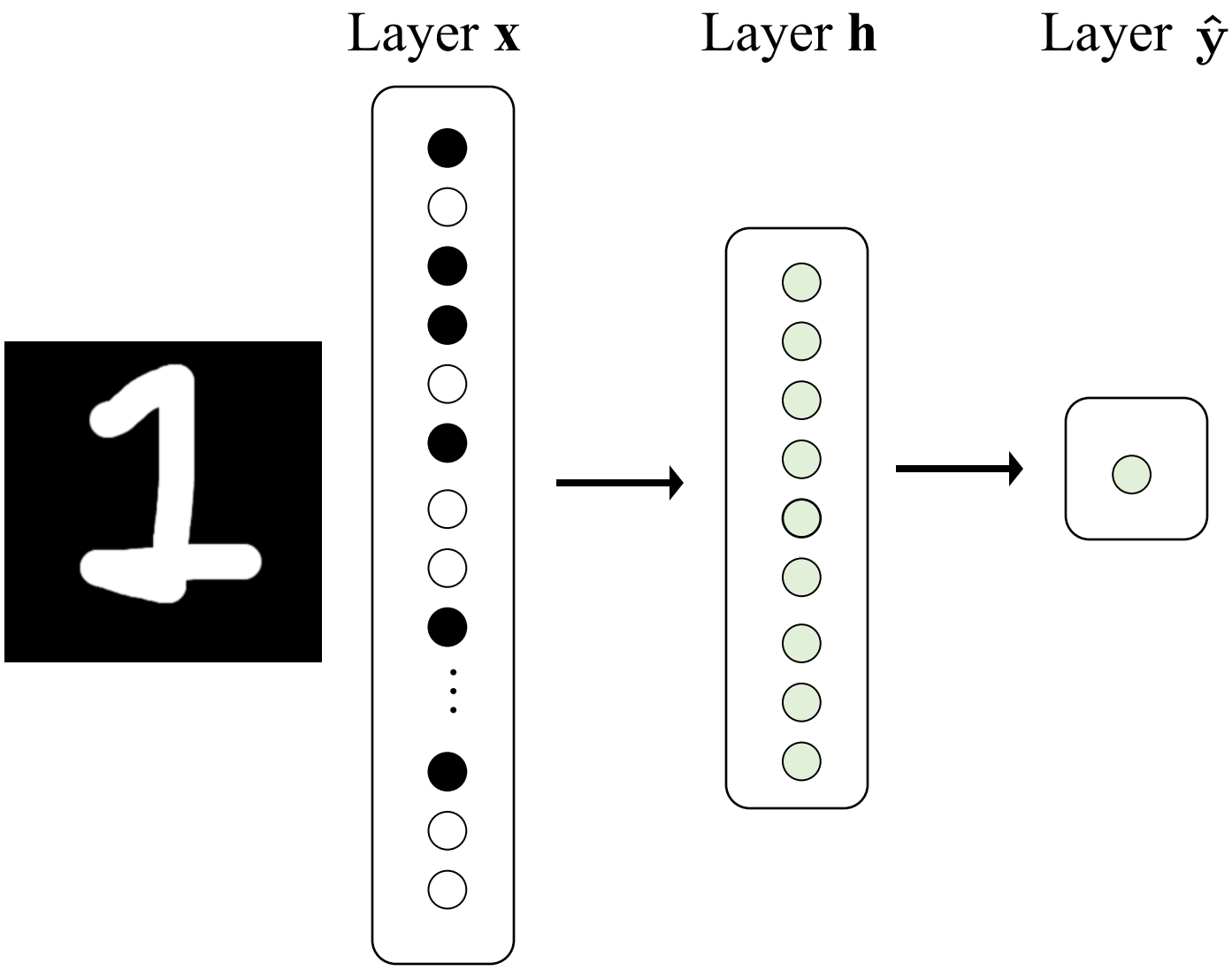


$$\hat{y} = \sigma\left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}\right)$$

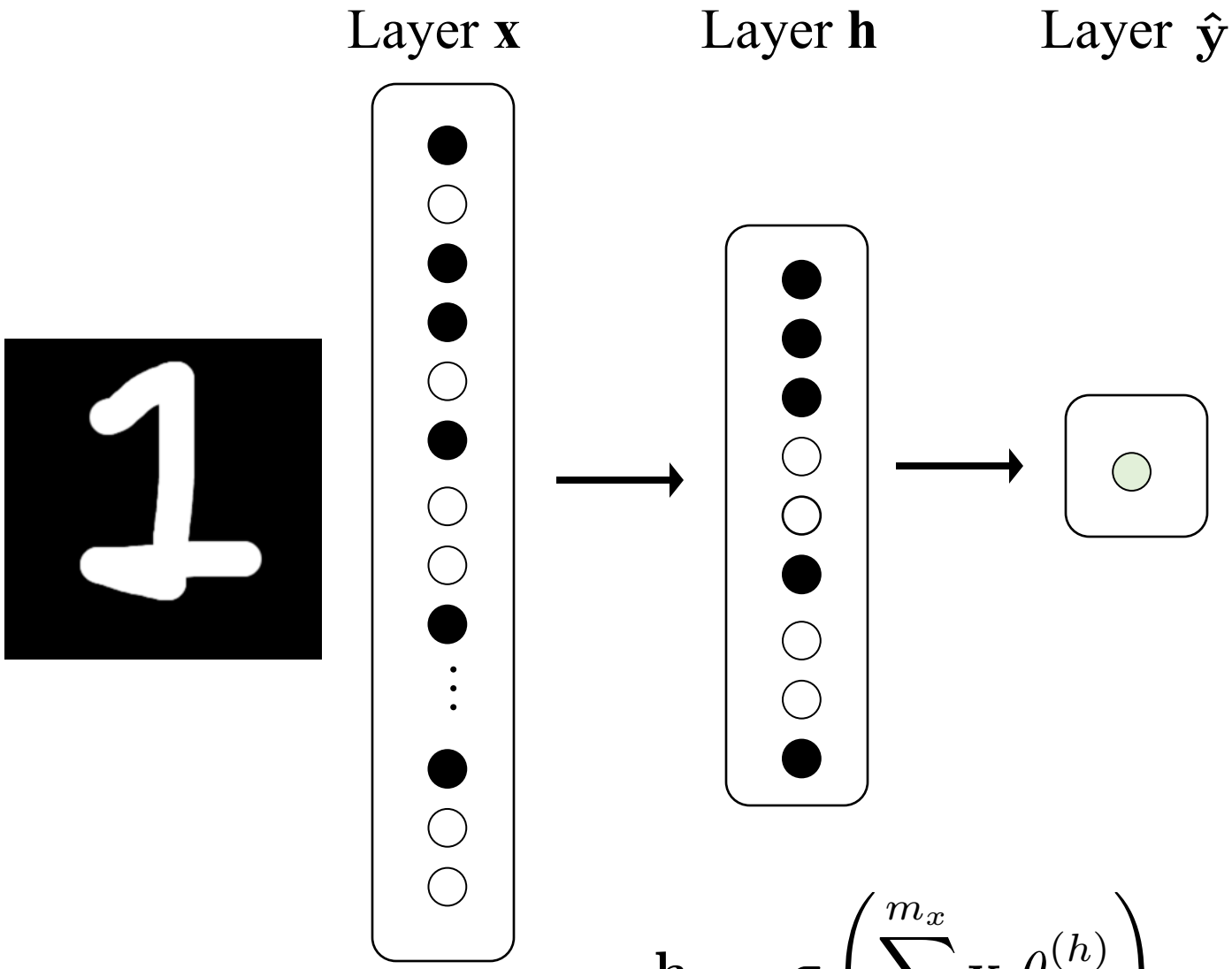
Forward Pass



Forward Pass

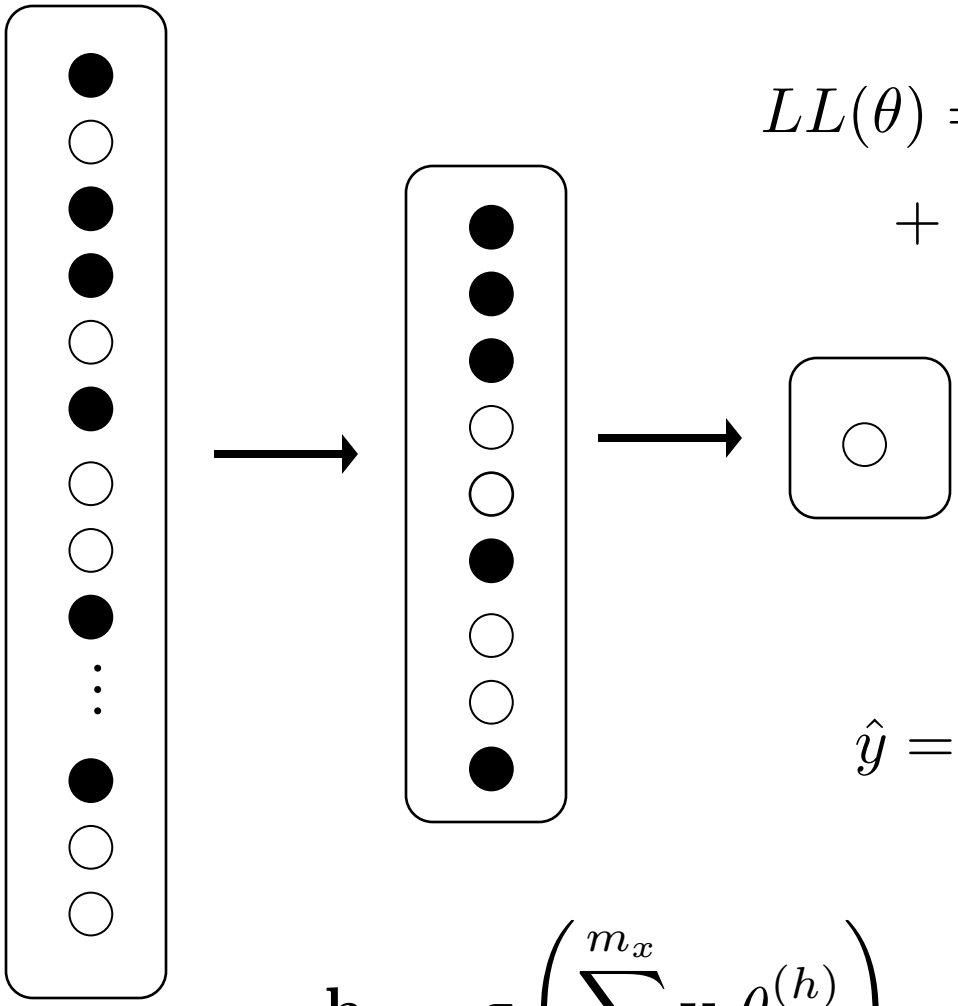


Forward Pass



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

Forward Pass

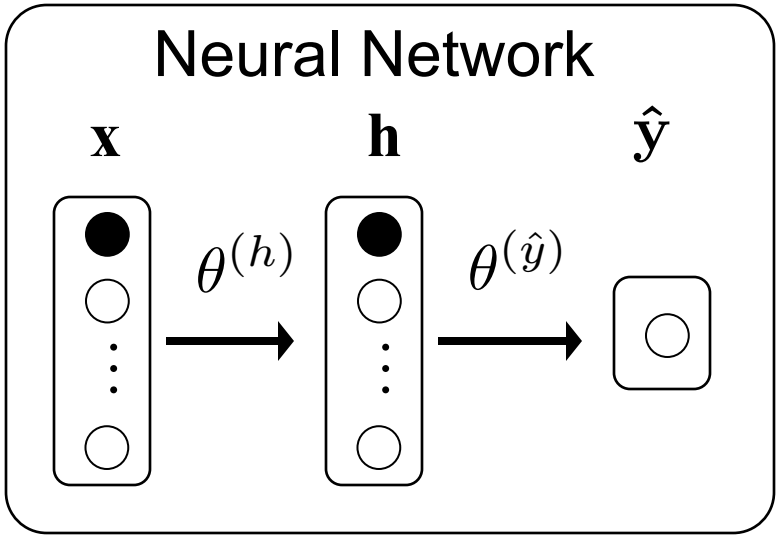


$$LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}]$$

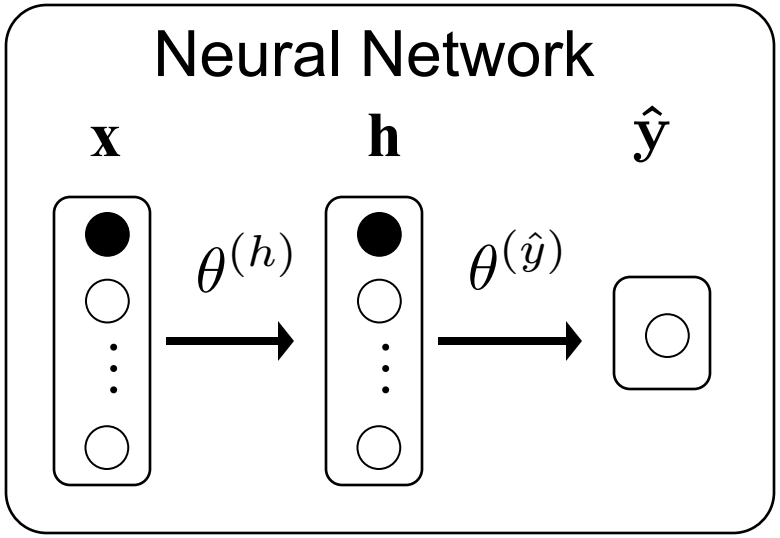
$$\hat{y} = \sigma\left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}\right)$$

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

All Together



Sanity Check 1



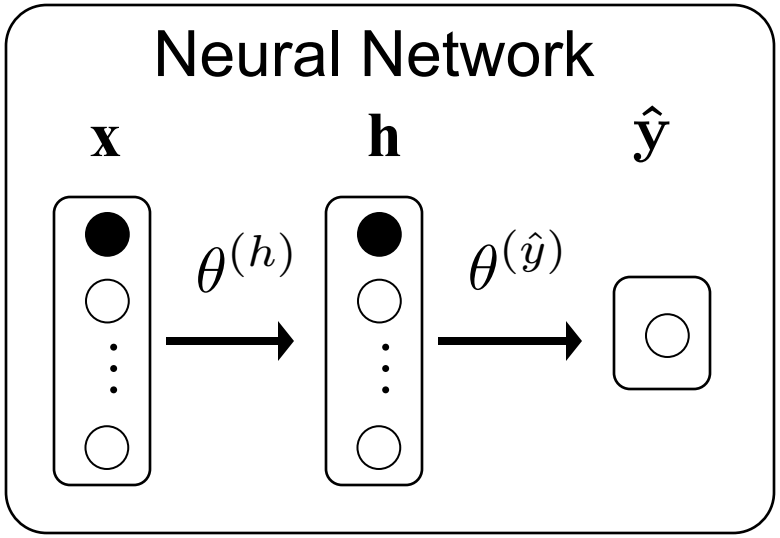
$$|\mathbf{x}| = 40$$

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

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

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

Sanity Check 2



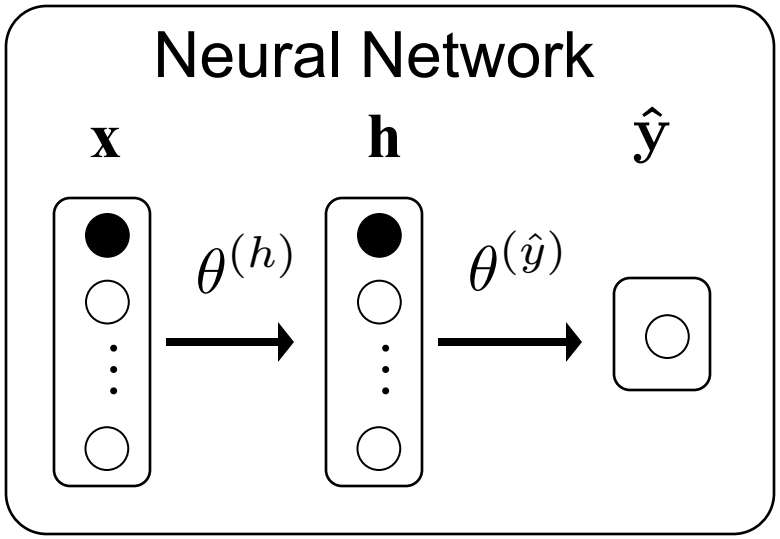
$$|\mathbf{x}| = 40$$

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

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

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

Sanity Check 3



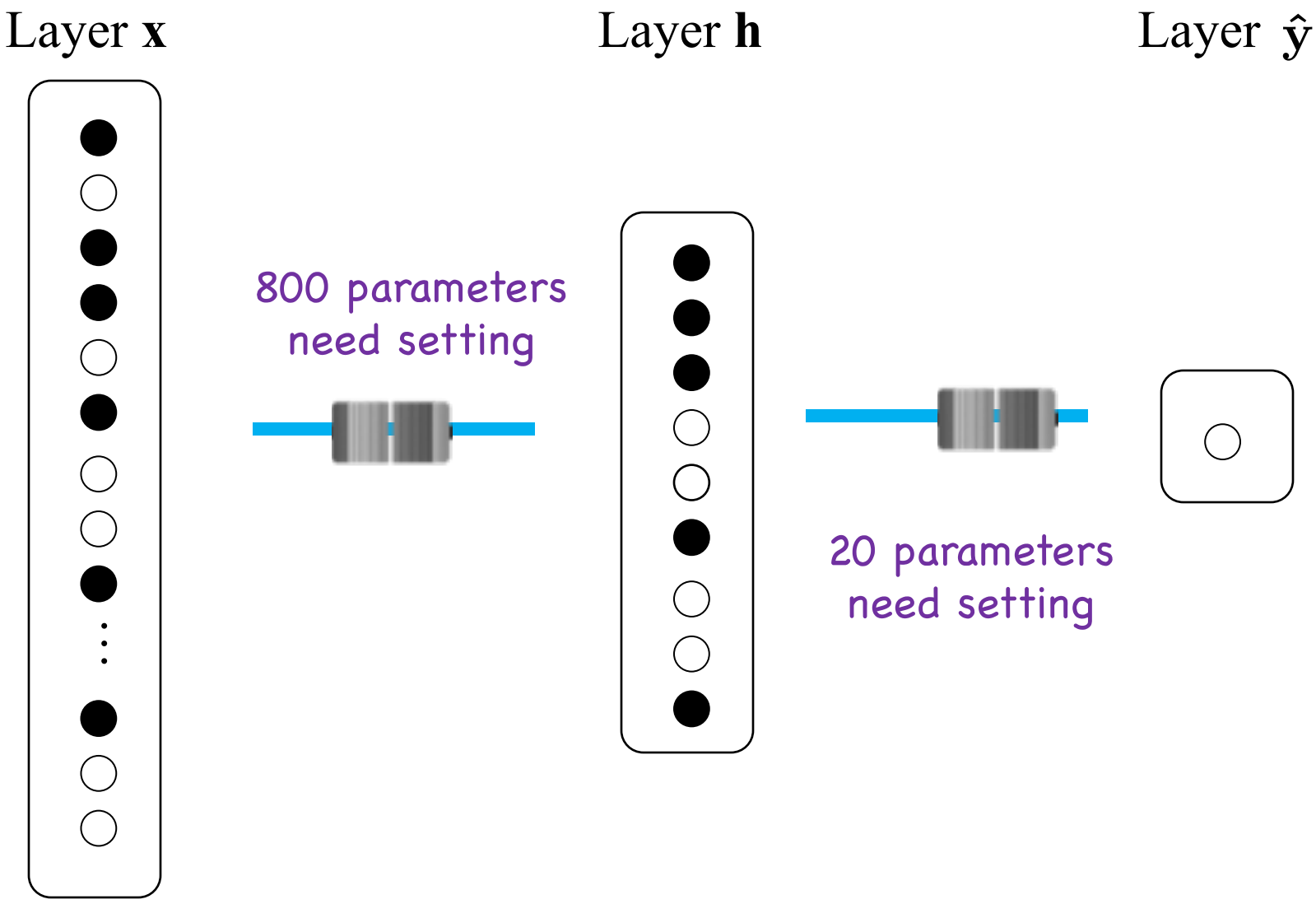
$$|\mathbf{x}| = 40$$

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

How many parameters in total?

a) 800 b) 20 c) 820 d) 16000

Forward Pass



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?

Why We Calculate Partial Derivatives

A deep learning model gets its **intelligence** by having **useful thetas**.

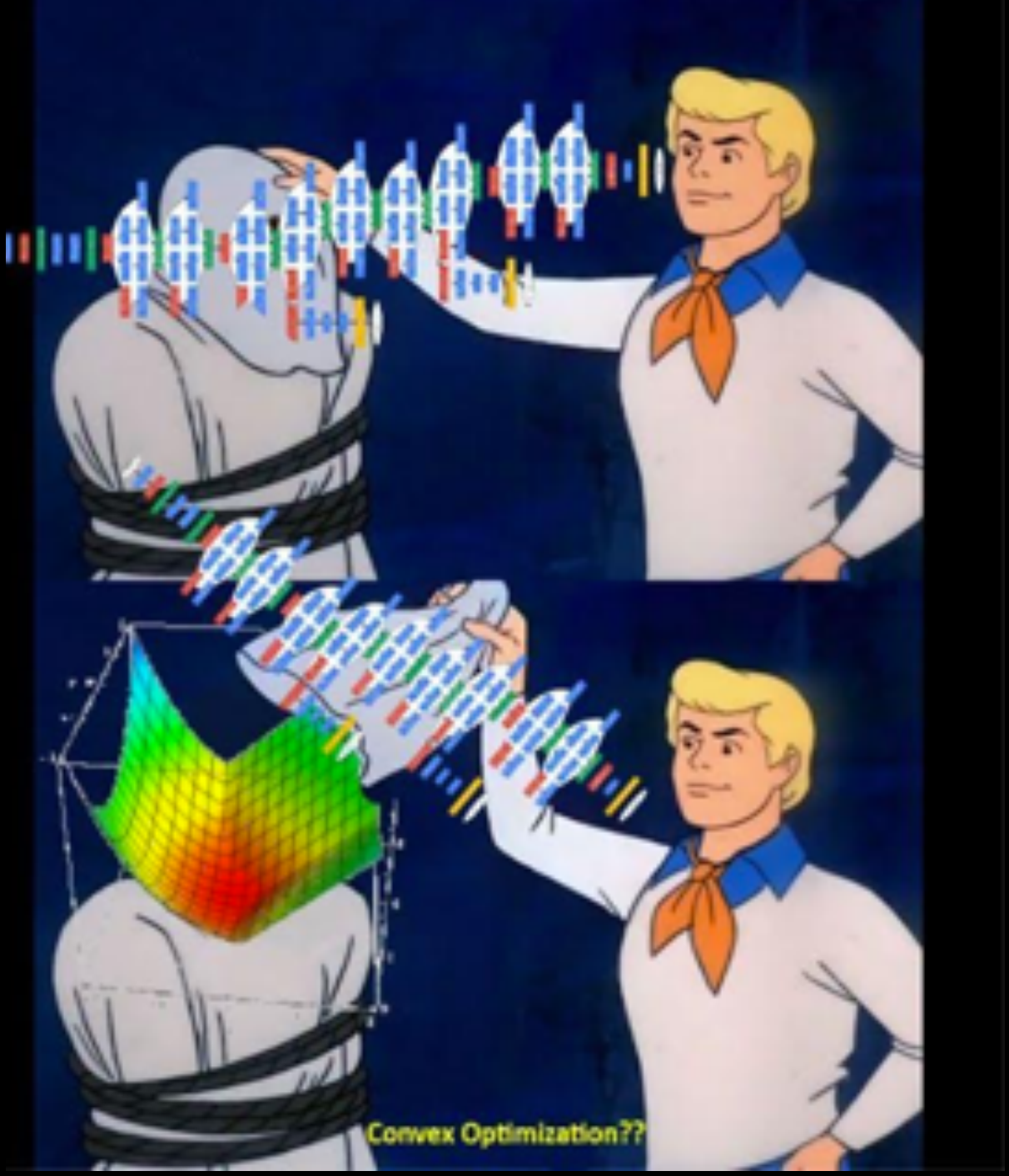
We can find **useful thetas**, by searching for ones that **maximize likelihood** of our training data

We can **maximize likelihood** using **optimization techniques** (such as gradient ascent).

In order to use **optimization techniques**, we need to calculate the **partial derivative** of likelihood with respect to thetas.

Basically MLE is hard because it has so many details

Thanks to Keith Eicher


Convex Optimization??

Only Have to Do Three Things

1. Make deep learning 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

Same Assumption, Same LL

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

For one datum

$$P(Y = y|\mathbf{X} = \mathbf{x}) = (\hat{y})^y (1 - \hat{y})^{1-y} \qquad Y \sim \text{Bern}(\hat{y})$$

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

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

Only Have to Do Three Things

1. Make deep learning 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 partial derivative of log likelihood with respect to each theta

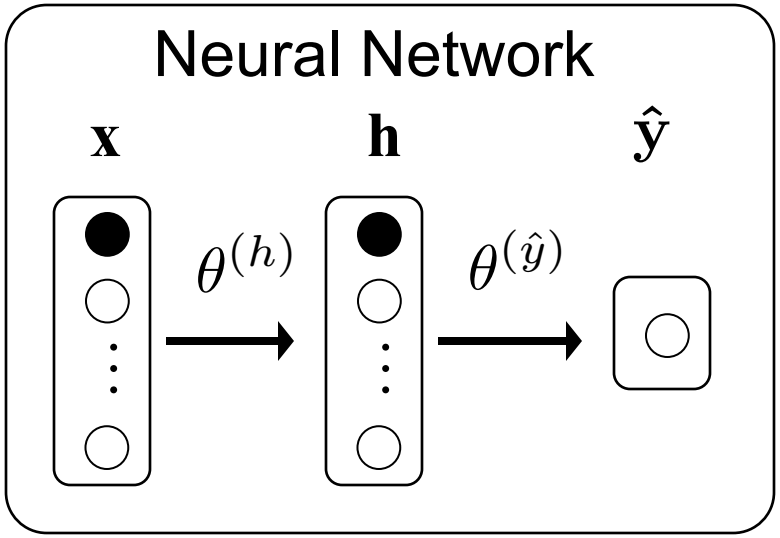
Derivative Goals

Loss with respect to output layer params

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

Loss with respect to hidden layer params

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

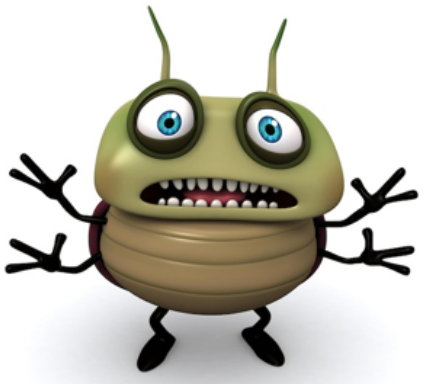


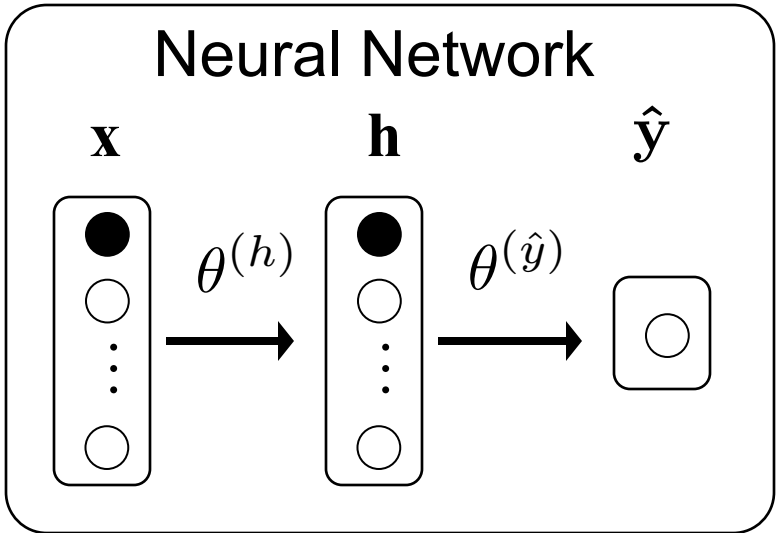
Bad Approach

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

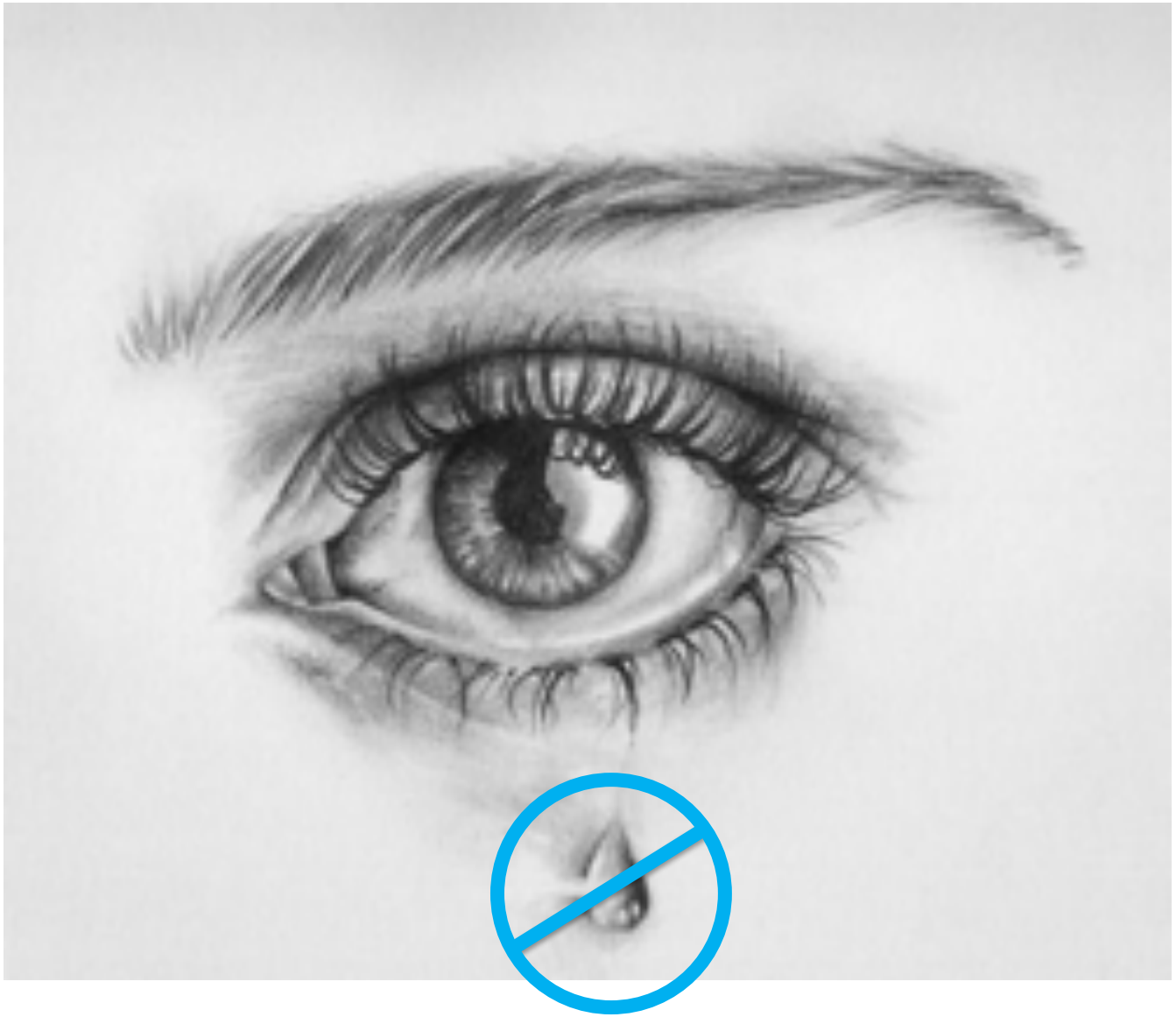
$$\hat{\mathbf{y}} = \sigma\left(\sum_{i=0}^{m_h} \mathbf{h}_i \theta_i^{(\hat{\mathbf{y}})}\right)$$

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





Derivatives Without Tears



Big Idea #1: Chain Rule

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

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

First use:

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

Big Idea #2: Sigmoid Derivative

True fact about sigmoid functions

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

Big Idea #3: Derivative of Sum

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

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

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

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

Warmup

Warmup

Compute:

$$\frac{\partial}{\partial \theta_j} \sigma(z)$$

Assume you can easily calculate:

$$\frac{\partial}{\partial \theta_j} z$$

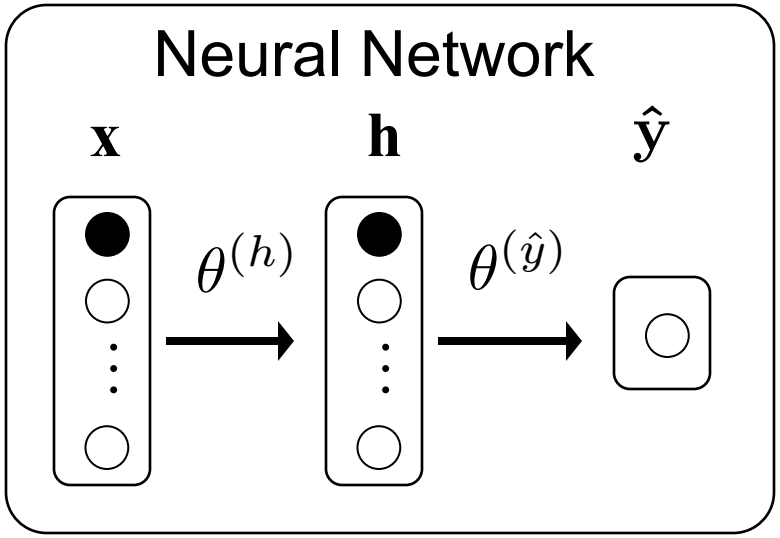
Derivative Goals

Loss with respect to output layer params

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

Loss with respect to hidden layer params

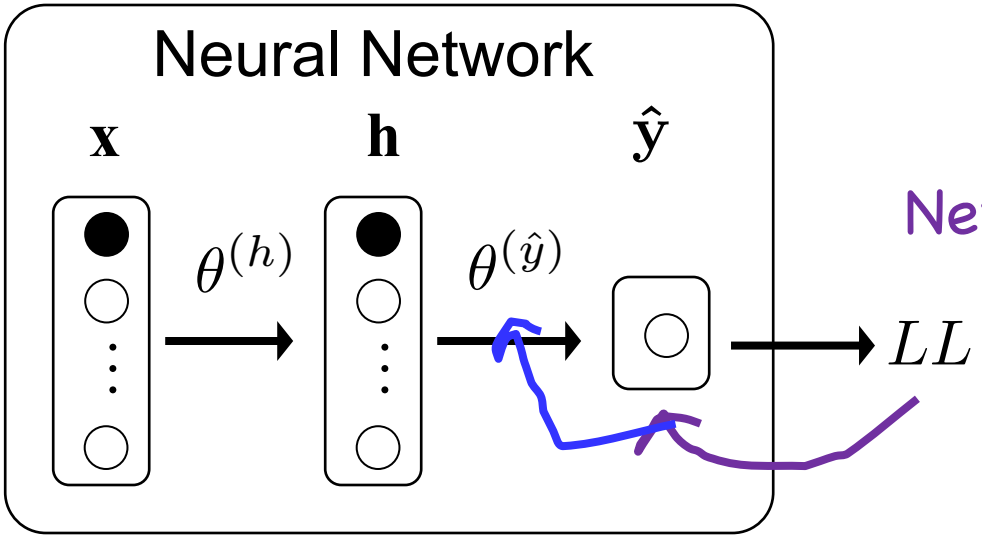
$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}}$$



Chain Rule Example 1

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

Goal



Network

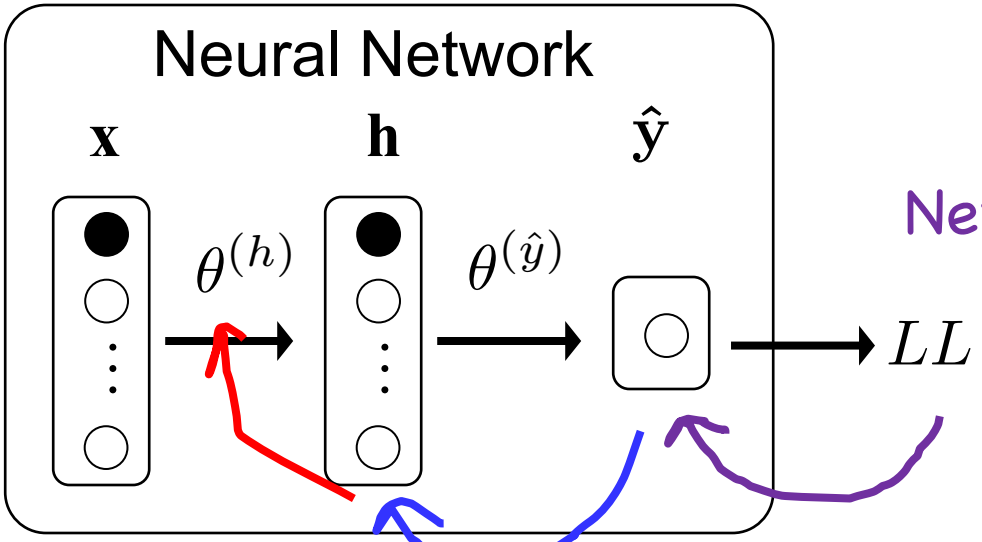
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}}$$

Decomposition

Chain Rule Example 2

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

Goal



Network

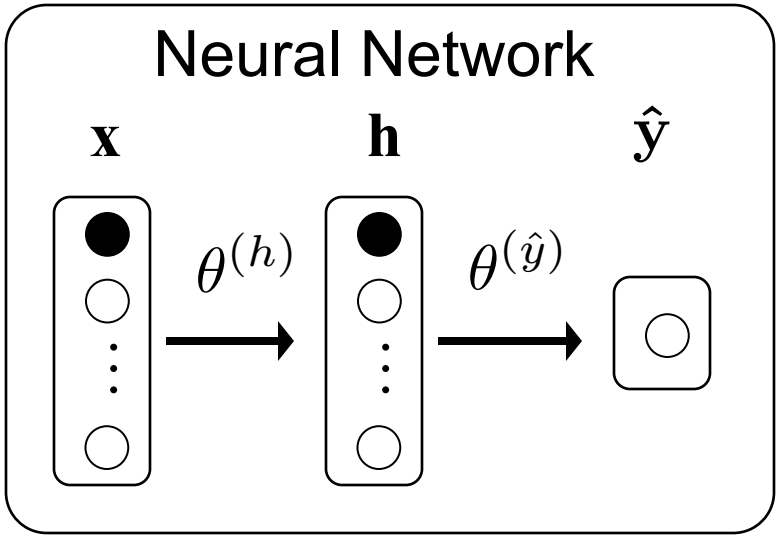
$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{h}_j} \cdot \frac{\partial \mathbf{h}_j}{\partial \theta_{i,j}^{(h)}}$$

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

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

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

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

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

Gradient of output layer params

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

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

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

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

What! That's not scary!

Make it Simple

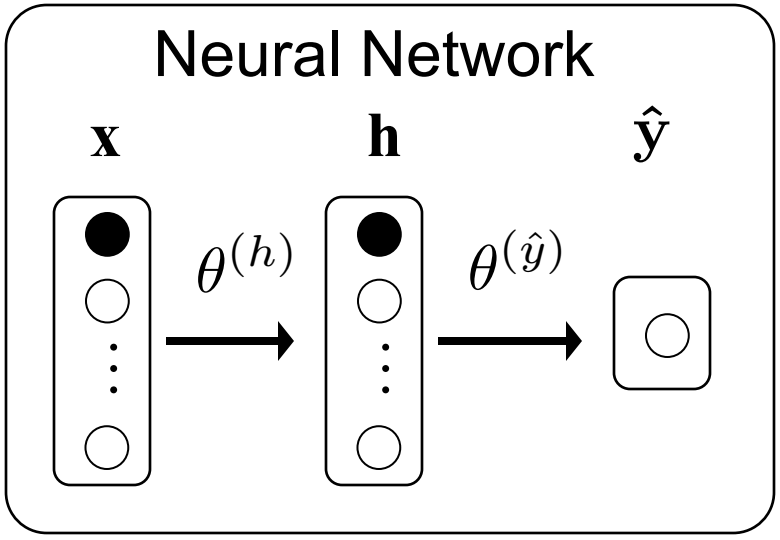
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \text{  } \cdot \text{  }$$

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

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

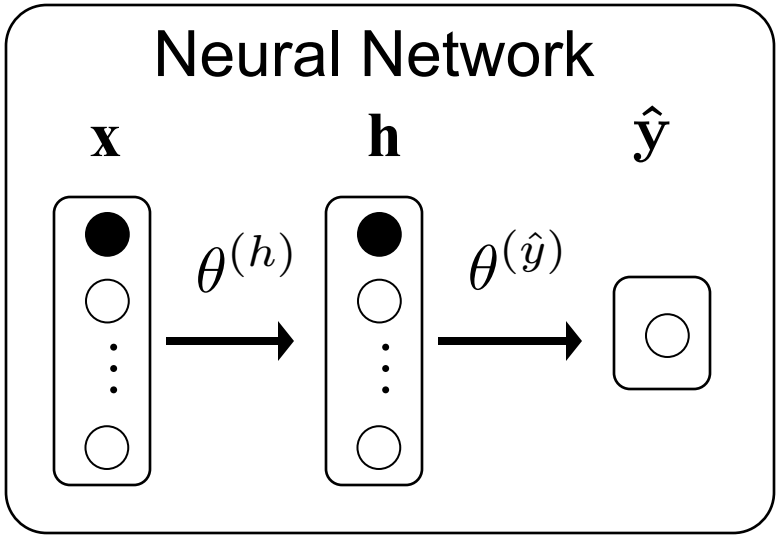
Boom!

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



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

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

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

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

Wait is it over?

Gradient of hidden layer params

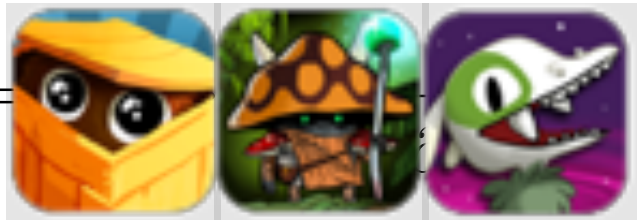
$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{h}_j} \cdot \frac{\partial \mathbf{h}_j}{\partial \theta_{i,j}^{(h)}}$$

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

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

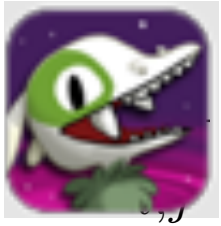
That one too?

Make it Simple

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

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

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



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

Congrats. You now know Backpropagation

Stretch!

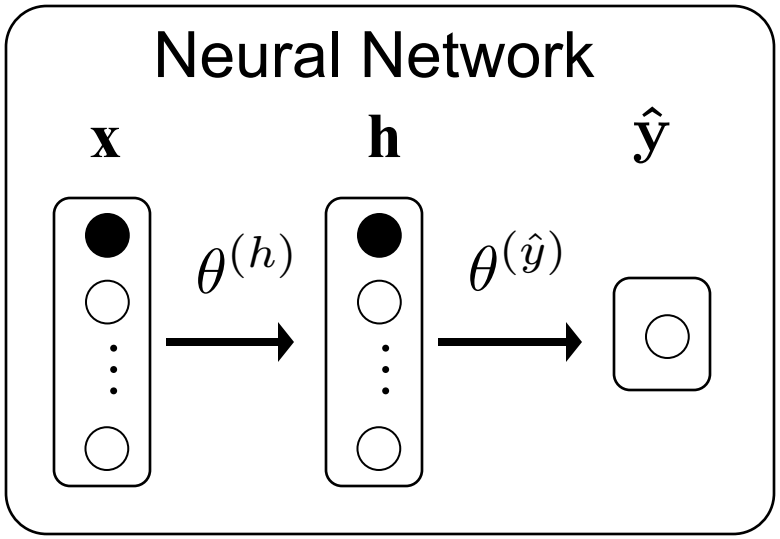
Summary: Simple Calculations For

Loss with respect to output layer params

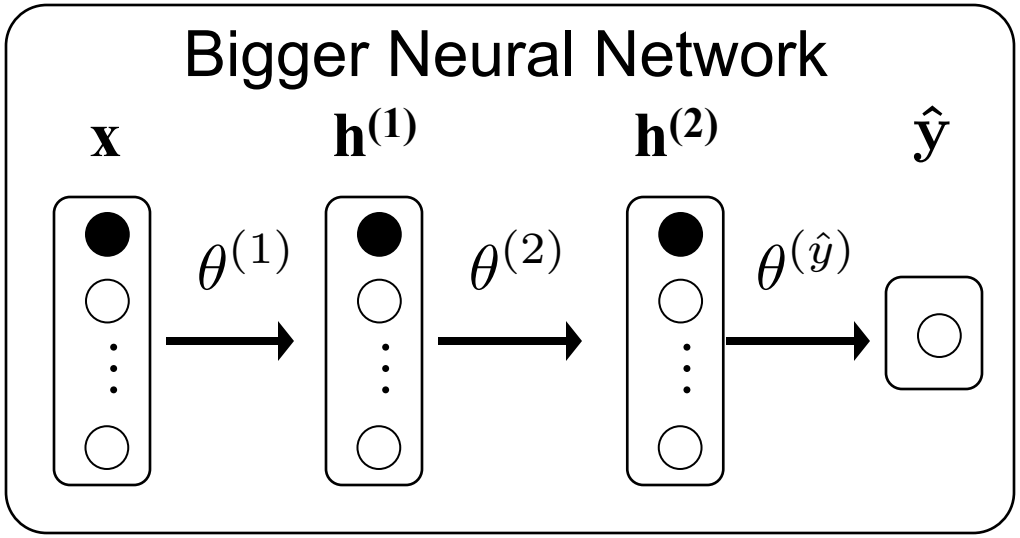
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}}$$

Loss with respect to hidden layer params

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



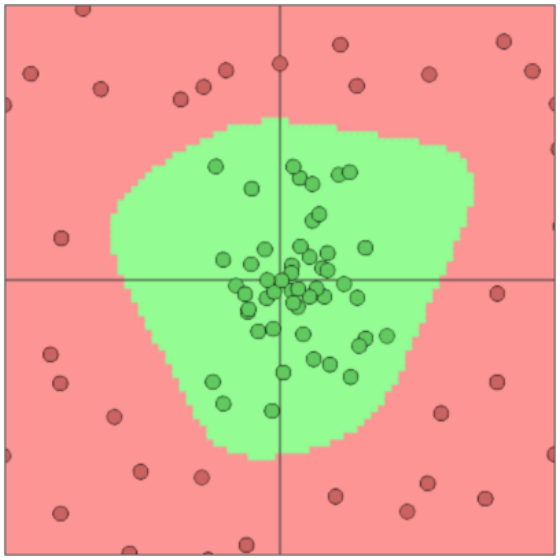
What Would You Do Here?

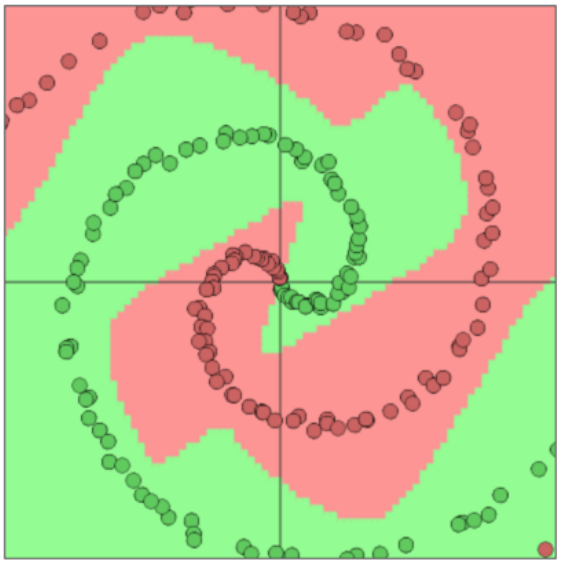


Chain rule:
Game changer for
artificial intelligence

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

Some Extra Ideas!

Multiple Outputs

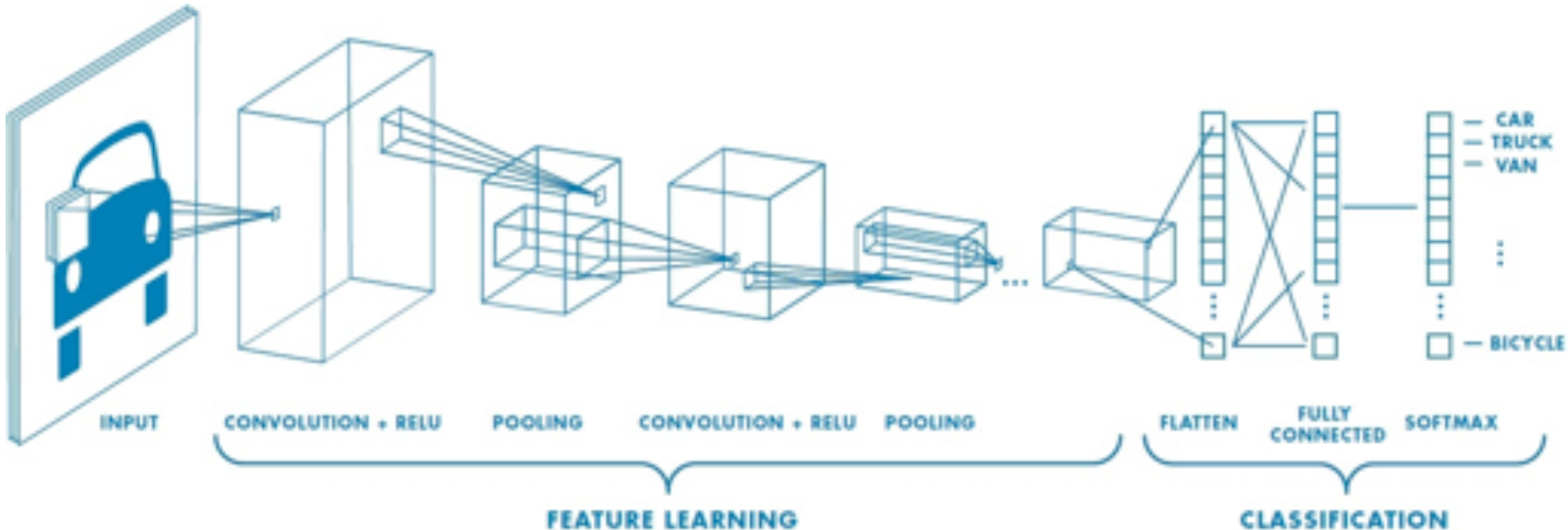
Multiple Output Classification?

Softmax is a generalization of the sigmoid function that squashes a K -dimensional vector $\mathbf{z}$ of arbitrary real values to a K-dimensional vector $\mathrm{softmax}(\mathbf{z})$ of real values in the range [0, 1] that add up to 1.

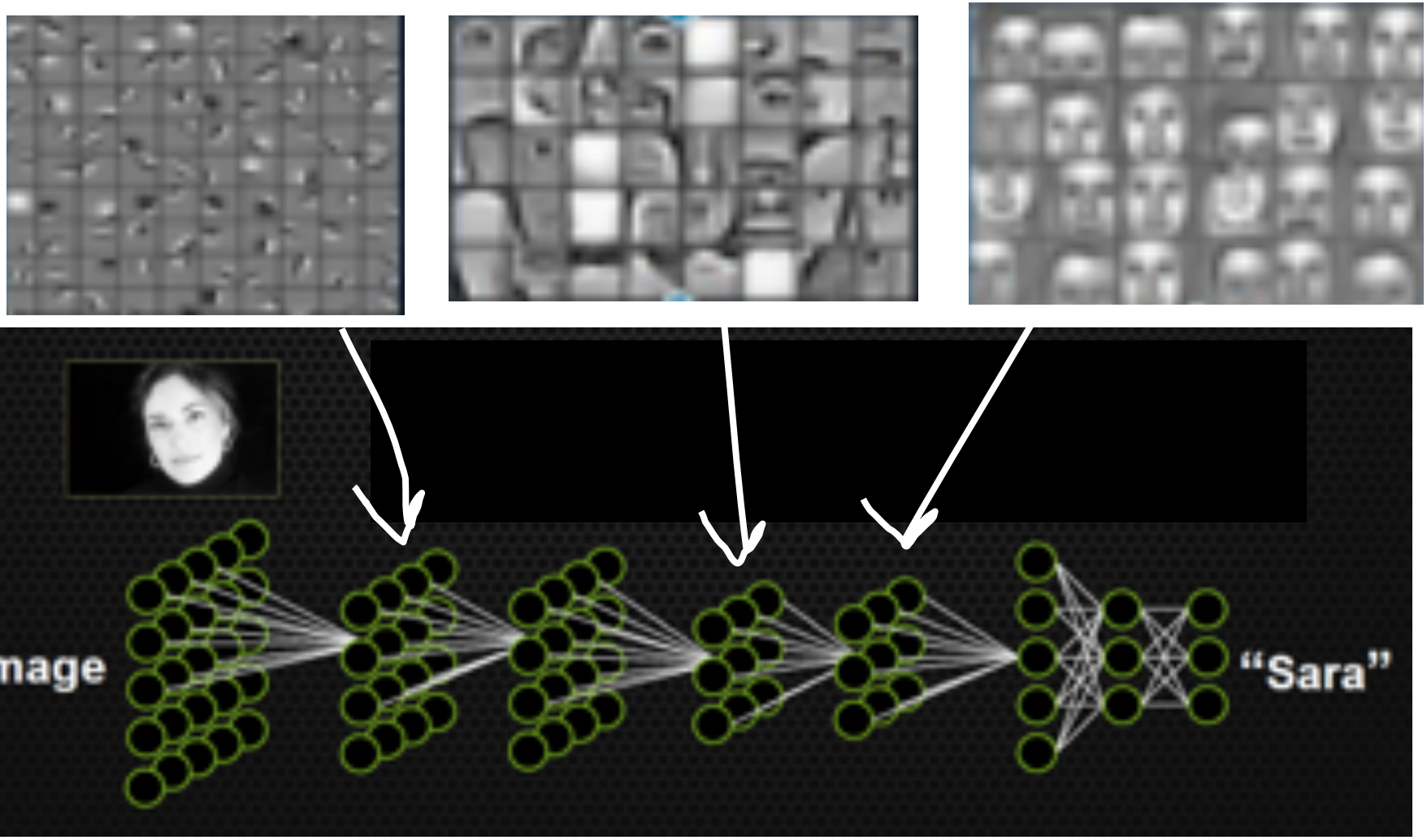
$$P(Y = j|\mathbf{X} = \mathbf{x}) = \mathrm{softmax}(f(\mathbf{x}))_j$$

Shared Weights?

Convolution it turns out if you want to force some of your weights to be shared for different neurons, the math isn't that much harder. This is used a lot for vision (CNN).



Works for any number of layers



$\hat{y}$

GoogLeNet Brain

1 Trillion Artificial Neurons

GoogLeNet Brain

Multiple,
Multi class output

22 layers deep

The Cat Neuron

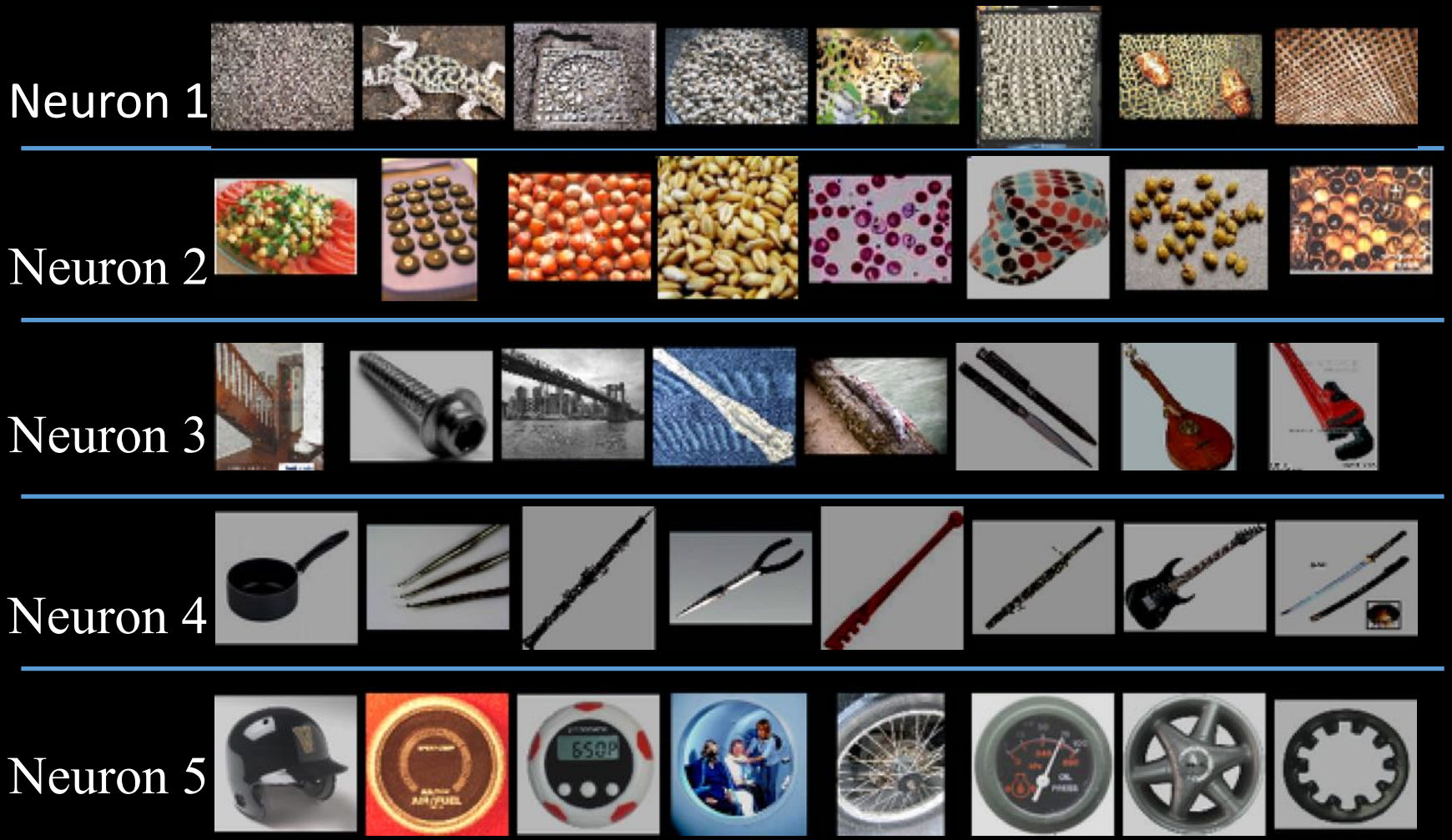
Top stimuli from the test set

Optimal stimulus
by numerical optimization

Le, et al., *Building high-level features using large-scale unsupervised*


Hire the smartest people in the world
Invent cat detector

Best Neuron Stimuli



Le, et al., *Building high-level features using large-scale unsupervised*

Best Neuron Stimuli

Neuron 6

Neuron 7

Neuron 8

Neuron 9

Le, et al., *Building high-level features using large-scale unsupervised*

Best Neuron Stimuli

Neuron 10

Neuron 11

Neuron 12

Neuron 13

Le, et al., *Building high-level features using large-scale unsupervised*

ImageNet Classification

22,000 categories

14,000,000 images

Hand-engineered features (SIFT, HOG, LBP),
Spatial pyramid, SparseCoding/Compression

Le, et al., *Building high-level features using large-scale unsupervised*

22,000 is a lot!

…
smoothhound, smoothhound shark, Mustelus mustelus
American smooth dogfish, Mustelus canis
Florida smoothhound, Mustelus norrisi
whitetip shark, reef whitetip shark, Triaenodon obseus
Atlantic spiny dogfish, Squalus acanthias
Pacific spiny dogfish, Squalus suckleyi
hammerhead, hammerhead shark
smooth hammerhead, Sphyrna zygaena
smalleye hammerhead, Sphyrna tudes
shovelhead, bonnethead, bonnet shark, Sphyrna tiburo
angel shark, angelfish, Squatina squatina, monkfish
electric ray, crampfish, numbfish, torpedo
smalltooth sawfish, Pristis pectinatus
guitarfish
roughtail stingray, Dasyatis centroura
butterfly ray
eagle ray
spotted eagle ray, spotted ray, Aetobatus narinari
cownose ray, cow-nosed ray, Rhinoptera bonasus
manta, manta ray, devilfish
Atlantic manta, Manta birostris
devil ray, Mobula hypostoma
grey skate, gray skate, Raja batis
little skate, Raja erinacea
…

Stingray

Mantaray

0.005%

Random guess

1.5%

Pre Neural Networks

?

GoogLeNet

Le, et al., *Building high-level features using large-scale unsupervised*

0.005%

Random guess

1.5%

Pre Neural Networks

43.9%

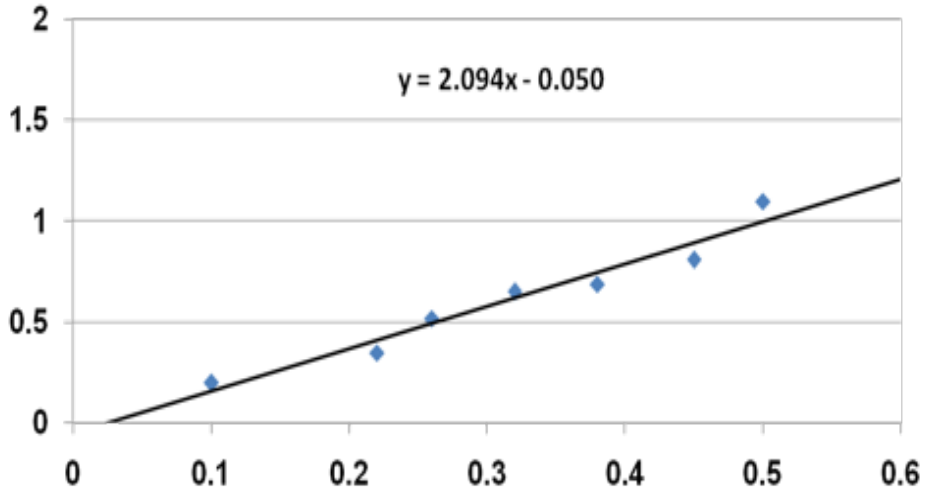
GoogLeNet

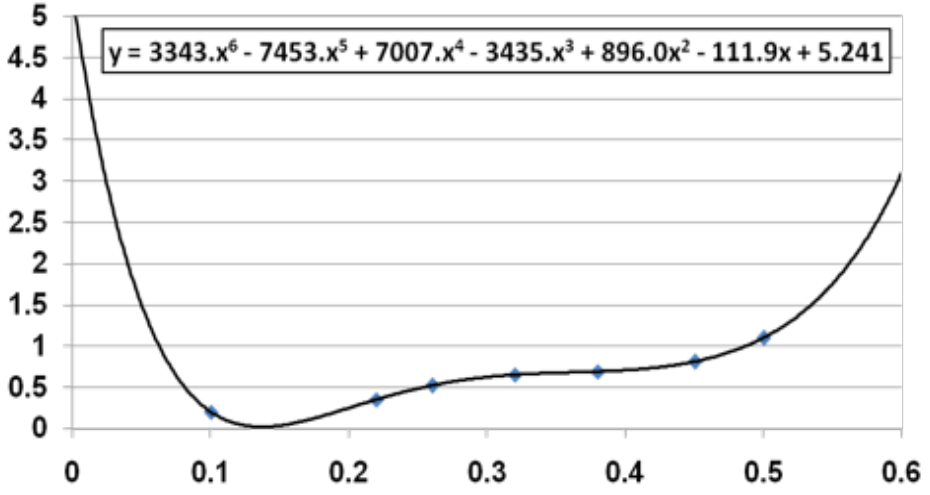
Szegedy et al, Going Deeper With Convolutions, CVPR 2015

How many parameters
is too many?

Good ML = Generalization

- Goal of machine learning: build models that ***generalize*** well to predicting new data
  - "Overfitting": fitting the training data too well, so we lose generality of model

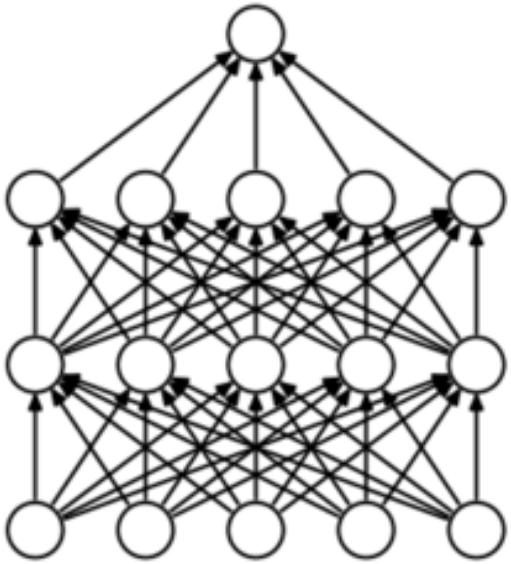




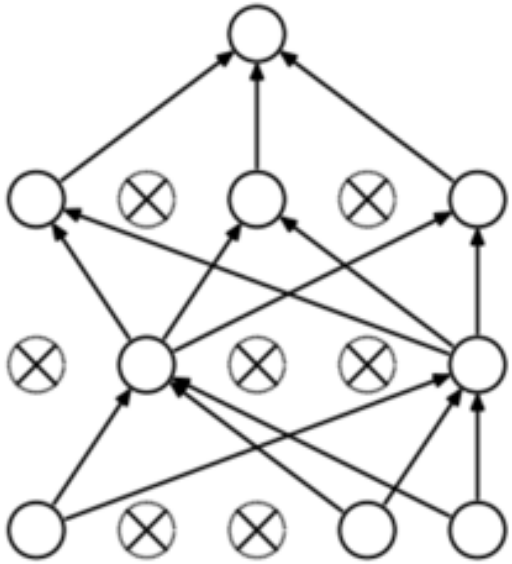
  - Polynomial on the right fits training data perfectly!
  - Which would you rather use to predict a new data point?

Prevent Overfitting?

Dropout when your model is training, randomly turn off your neurons with probability 0.5. It will make your network more robust.



(a) Standard Neural Net



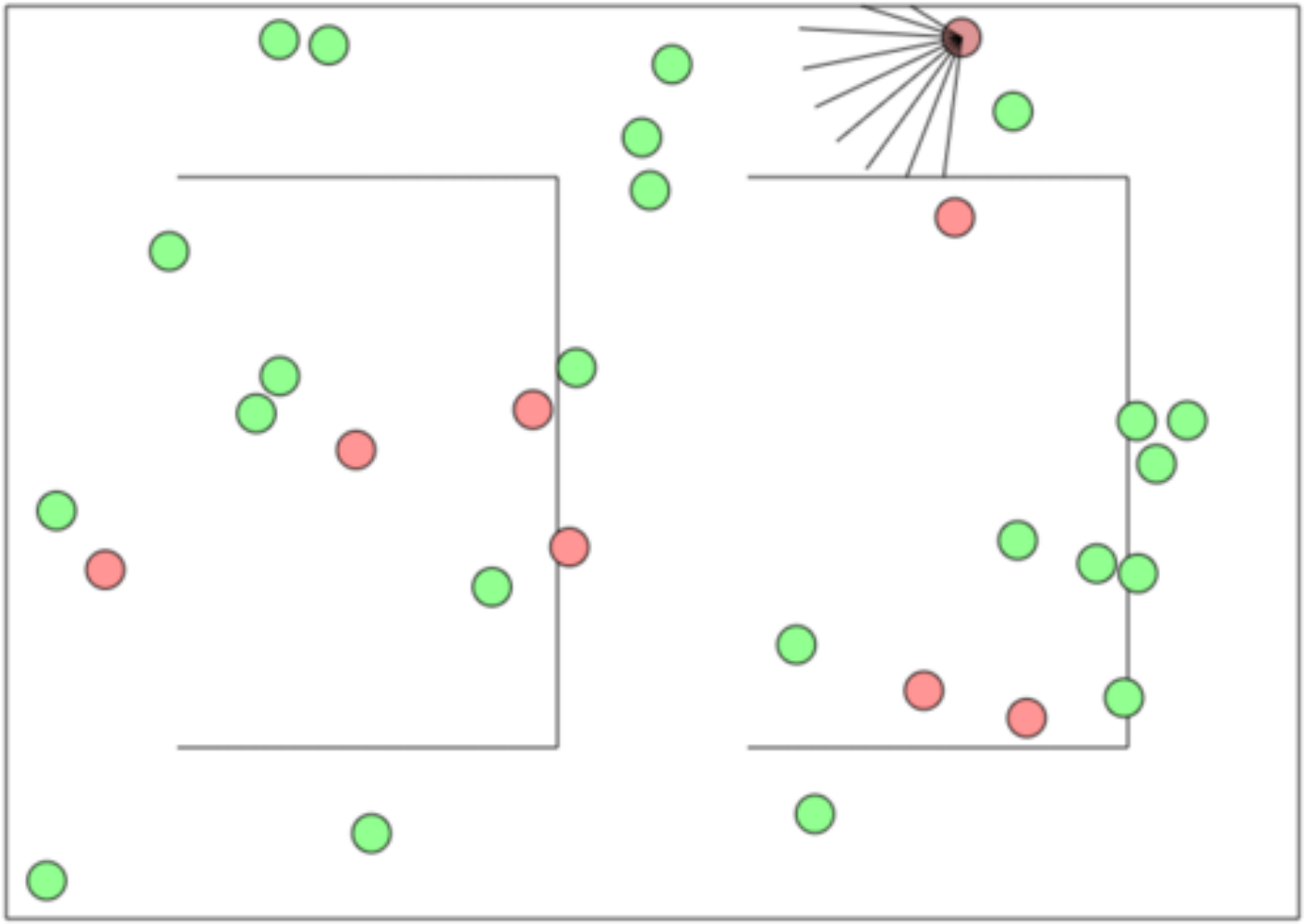
(b) After applying dropout.

Not everything is classification

Making Decisions?

Deep Reinforcement Learning
Instead of having the output of a model be a probability you can make it an expectation.

Deep Reinforcement Learning

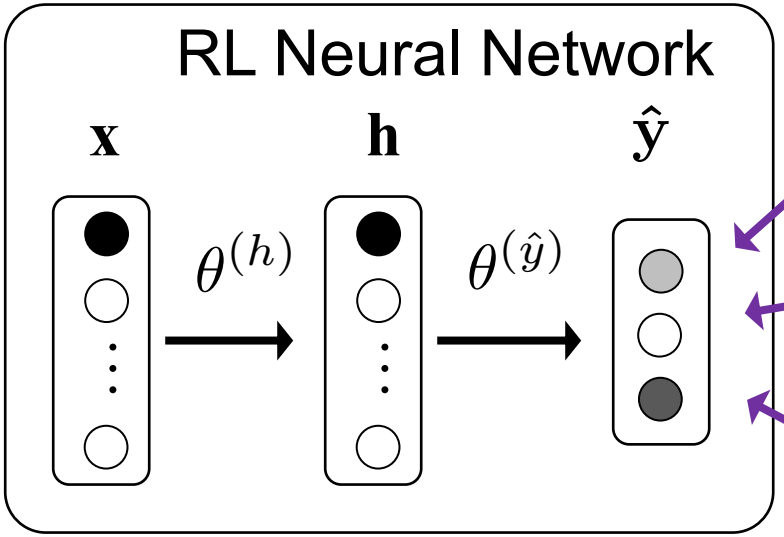


http://cs.stanford.edu/people/karpathy/convnetjs/demo/rldemo.html

Deep Reinforcement Learning

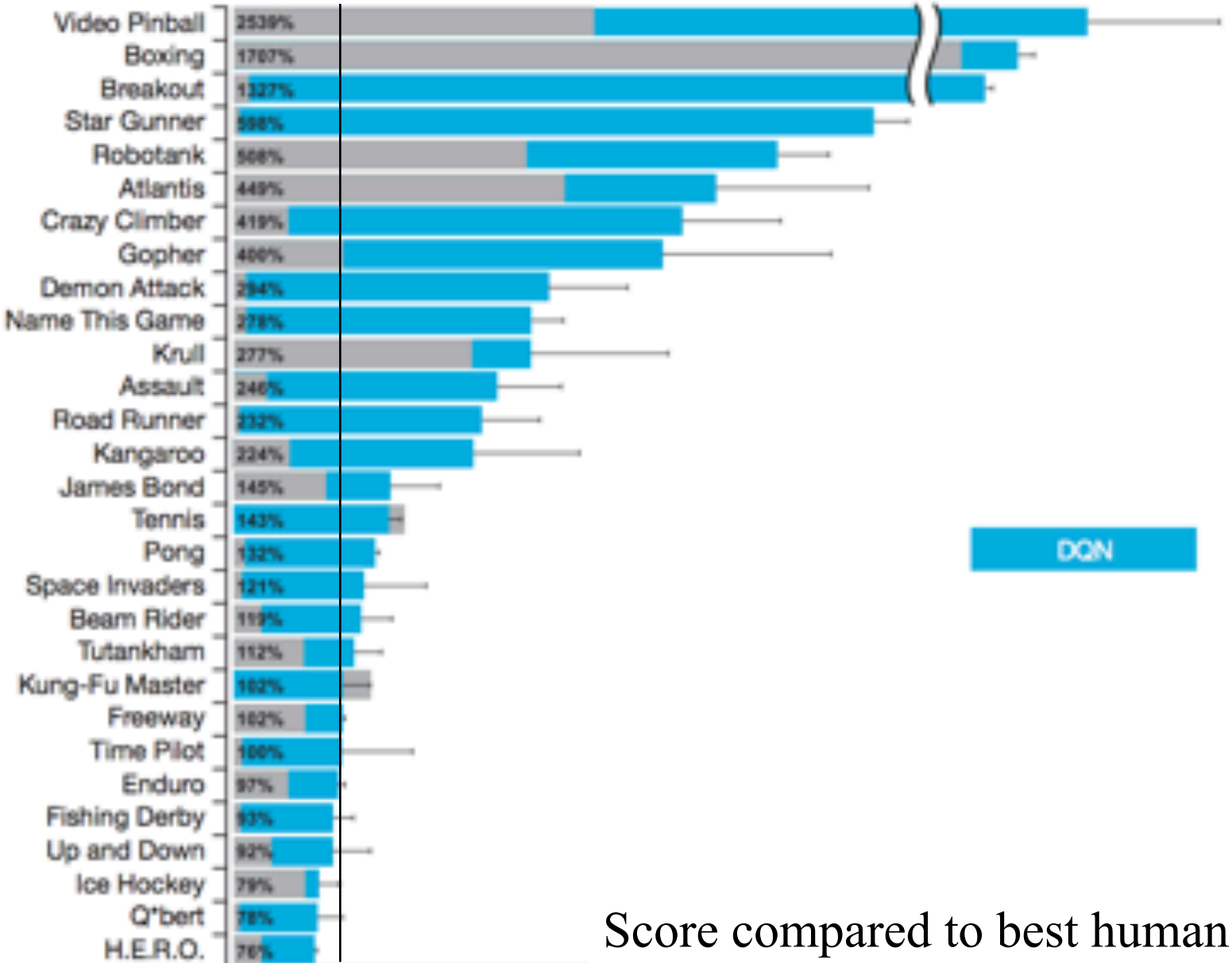
R is a reward and A_i is a legal action

Input is a representation of current state (S)



Interpret outputs as expected reward for a given action

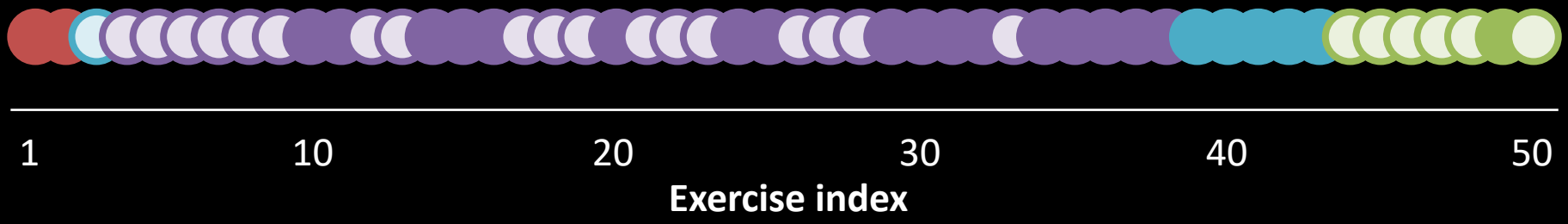
Deep Mind Atari Games



Score compared to best human

When Do I Get a Robo Tutor?

Story of Riley



Exercise Type:

- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

Answer:

- Correct
- Incorrect

KHAN ACADEMY

Story of Riley

Exercise Type:

- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

Answer:

- Correct
- Incorrect

KHAN ACADEMY

Story of Riley

Exercise Type:

- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

Answer:

- Correct
- Incorrect

KHAN ACADEMY

Story of Riley

Exercise Type:

- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

Answer:

- Correct
- Incorrect

KHAN ACADEMY

Story of Riley

Exercise Type:

- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

Answer:

- Correct
- Incorrect

KHAN ACADEMY

Story of Riley

Exercise Type:

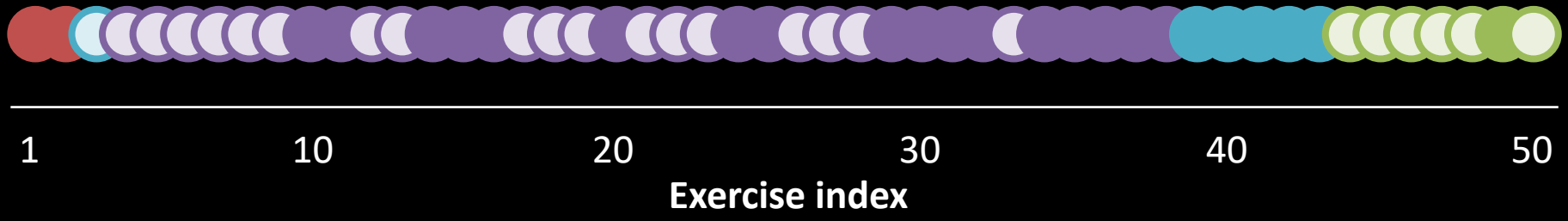
- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

Answer:

- Correct
- Incorrect

KHAN ACADEMY

Story of Riley



What does Riley know?

Exercise Type:

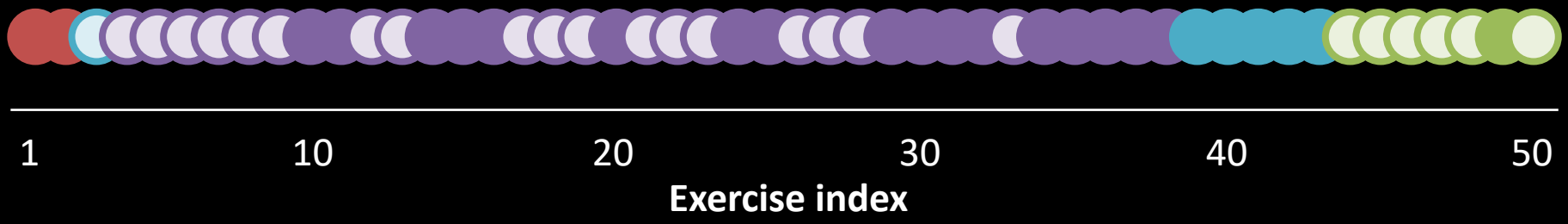
- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

Answer:

- Correct
- Incorrect

KHAN ACADEMY

Story of Riley



What should Riley do next?

Exercise Type:

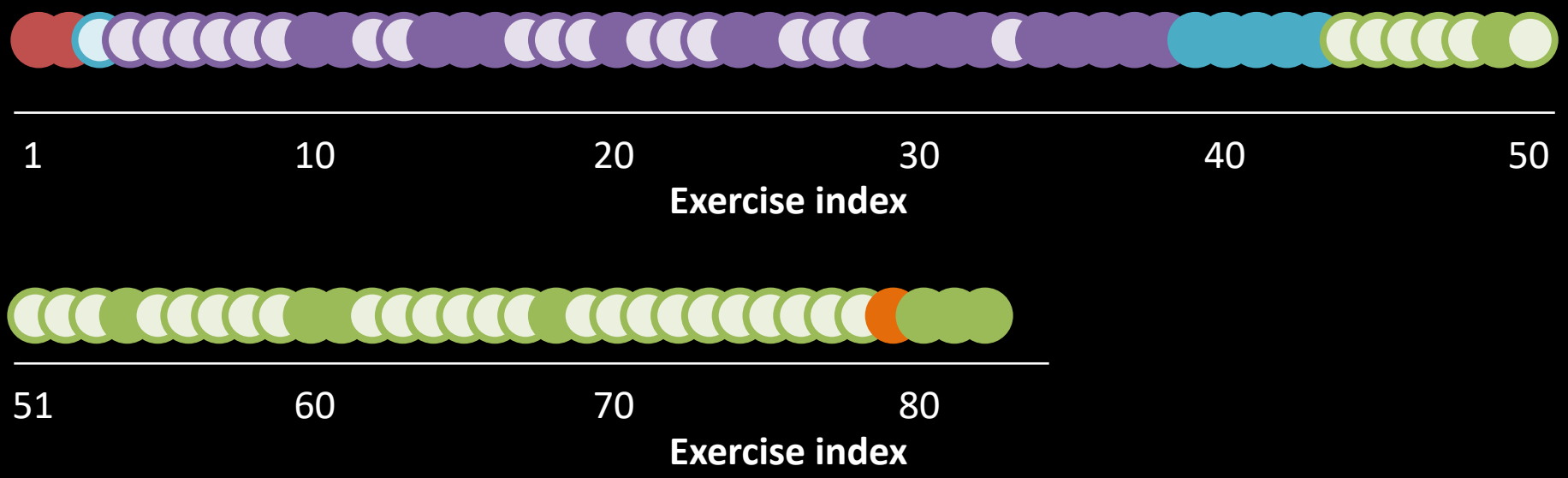
- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

Answer:

- Correct
- Incorrect

KHAN ACADEMY

Story of Riley



Exercise Type:

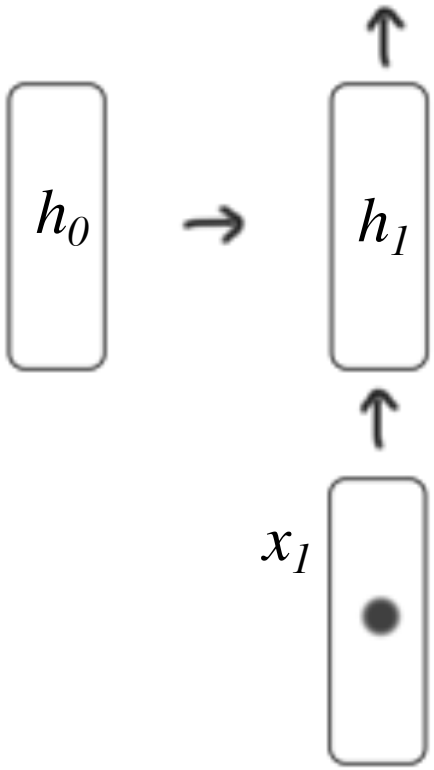
- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

Answer:

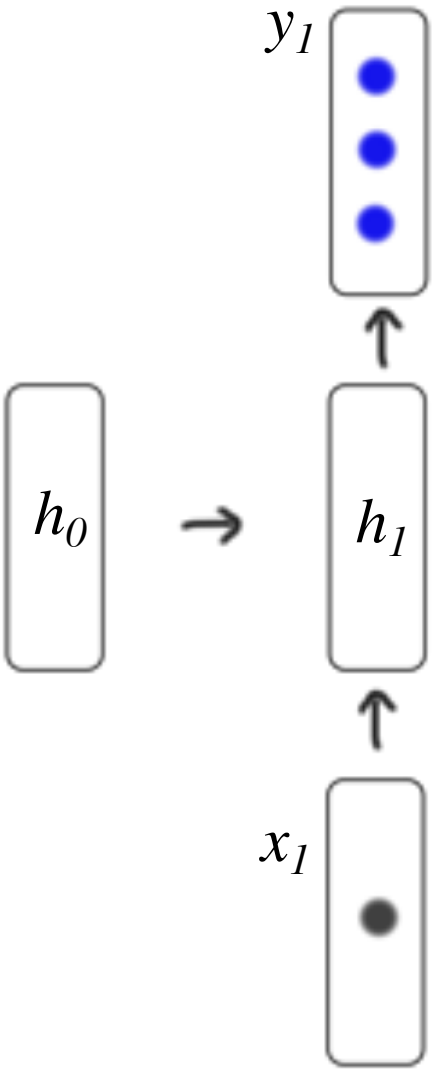
- Correct
- Incorrect

KHAN ACADEMY

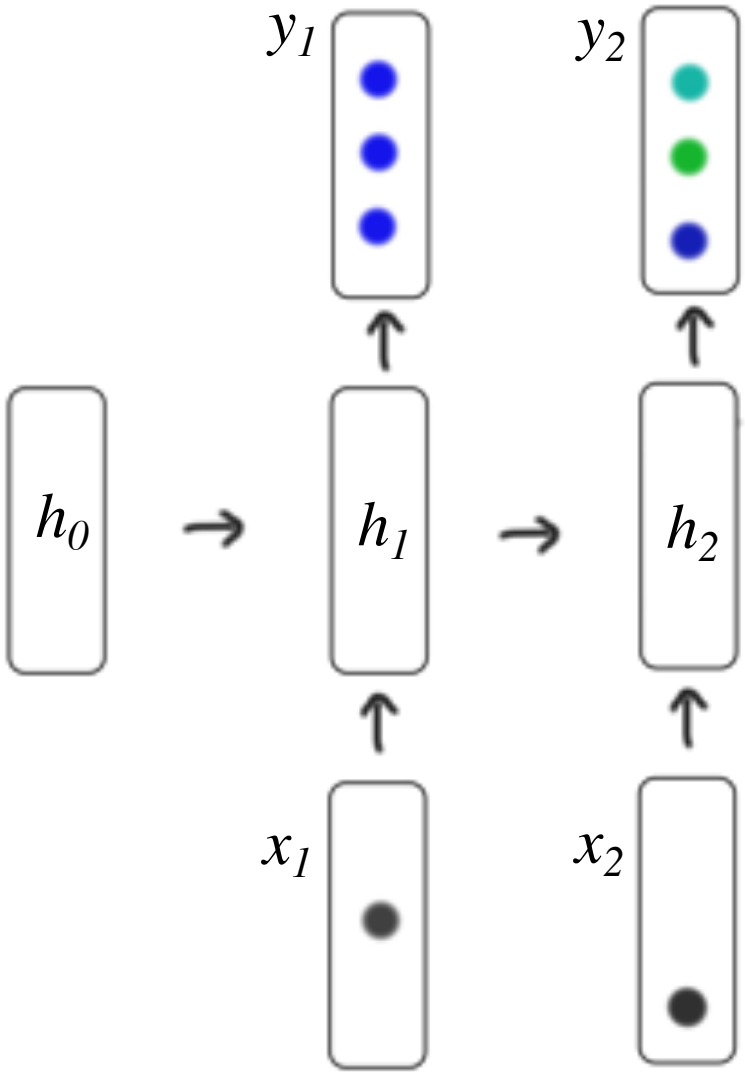
Recurrent Neural Network



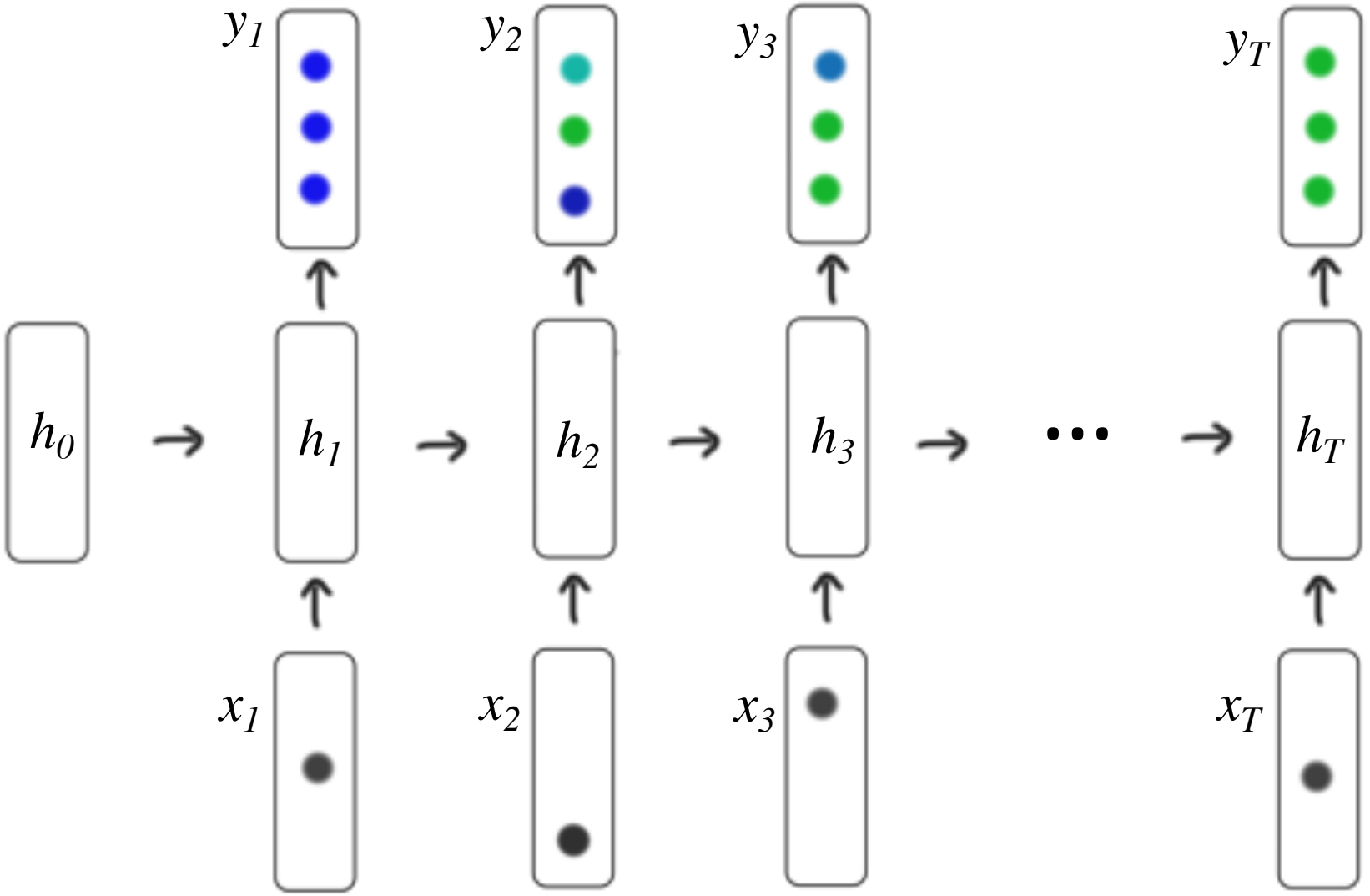
Recurrent Neural Network



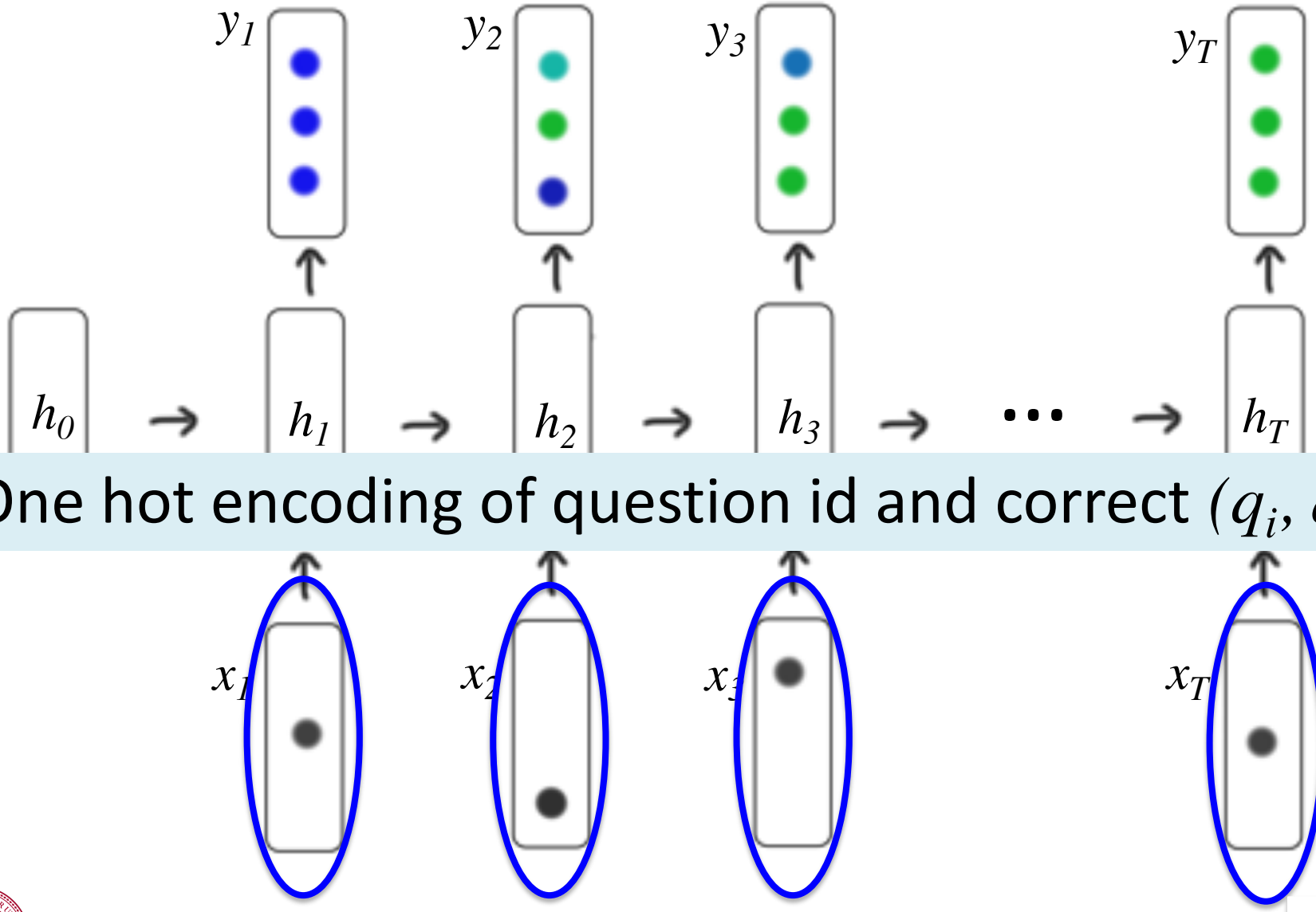
Recurrent Neural Network



Recurrent Neural Network

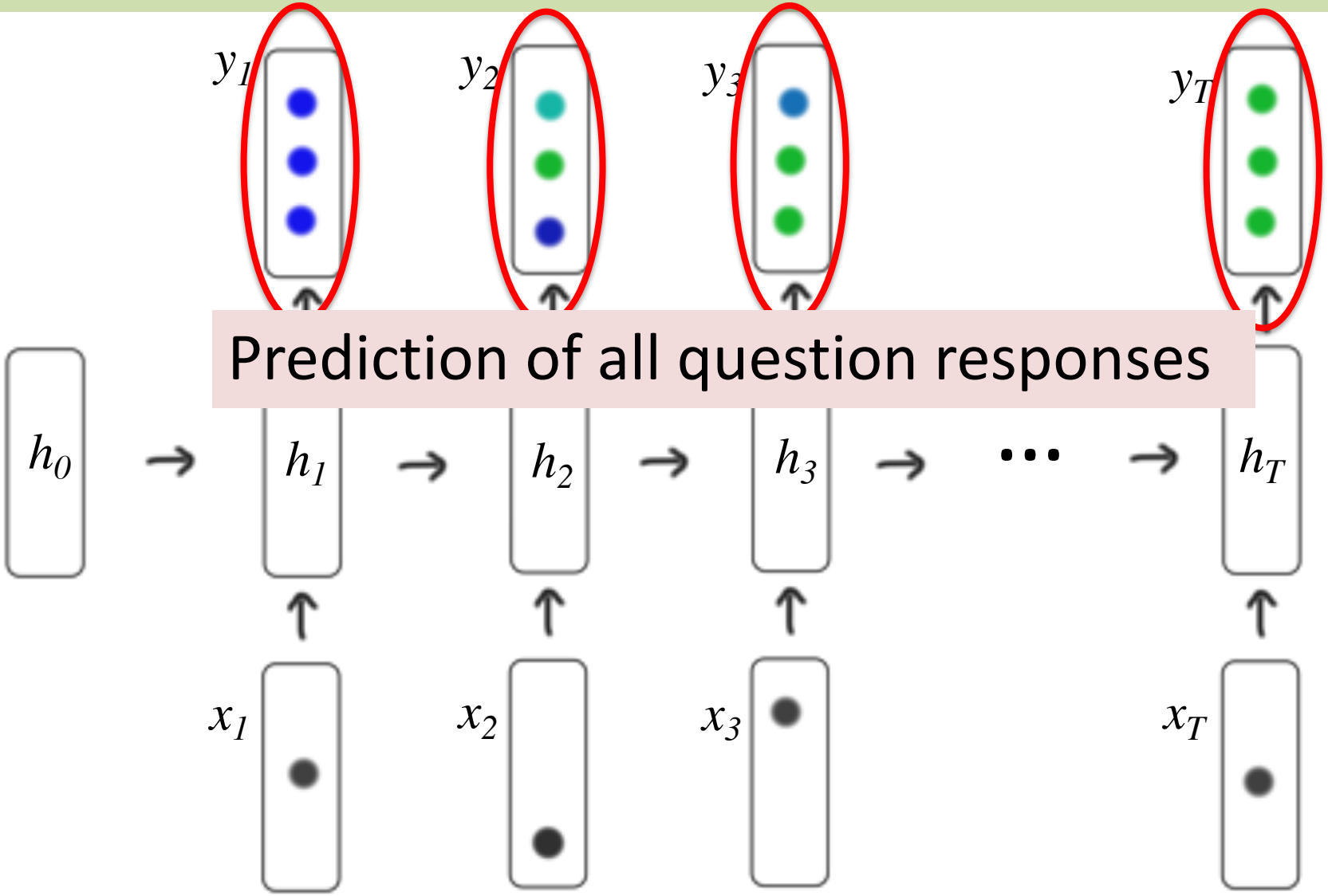


Deep Knowledge Tracing

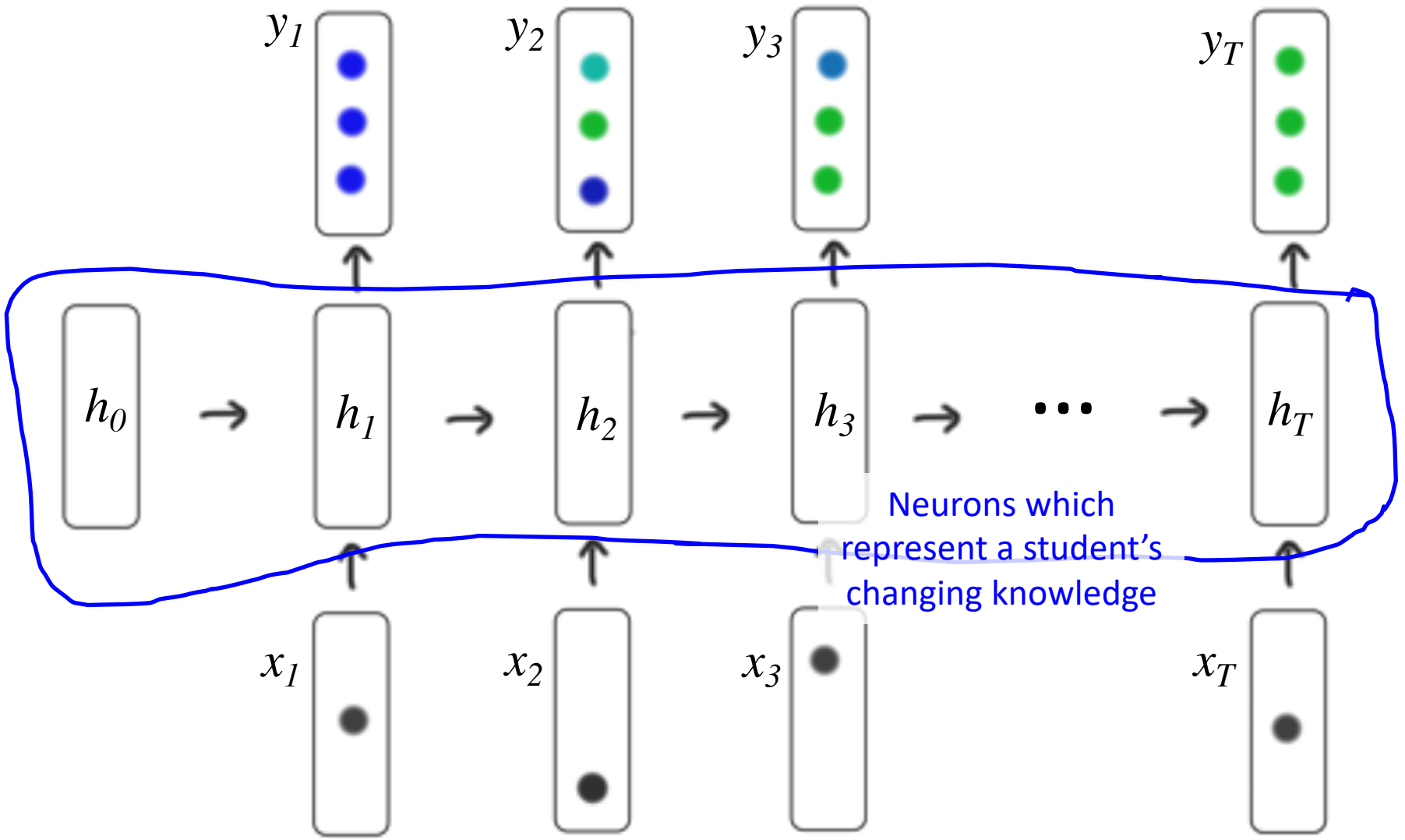


One hot encoding of question id and correct (q_i, a_i)

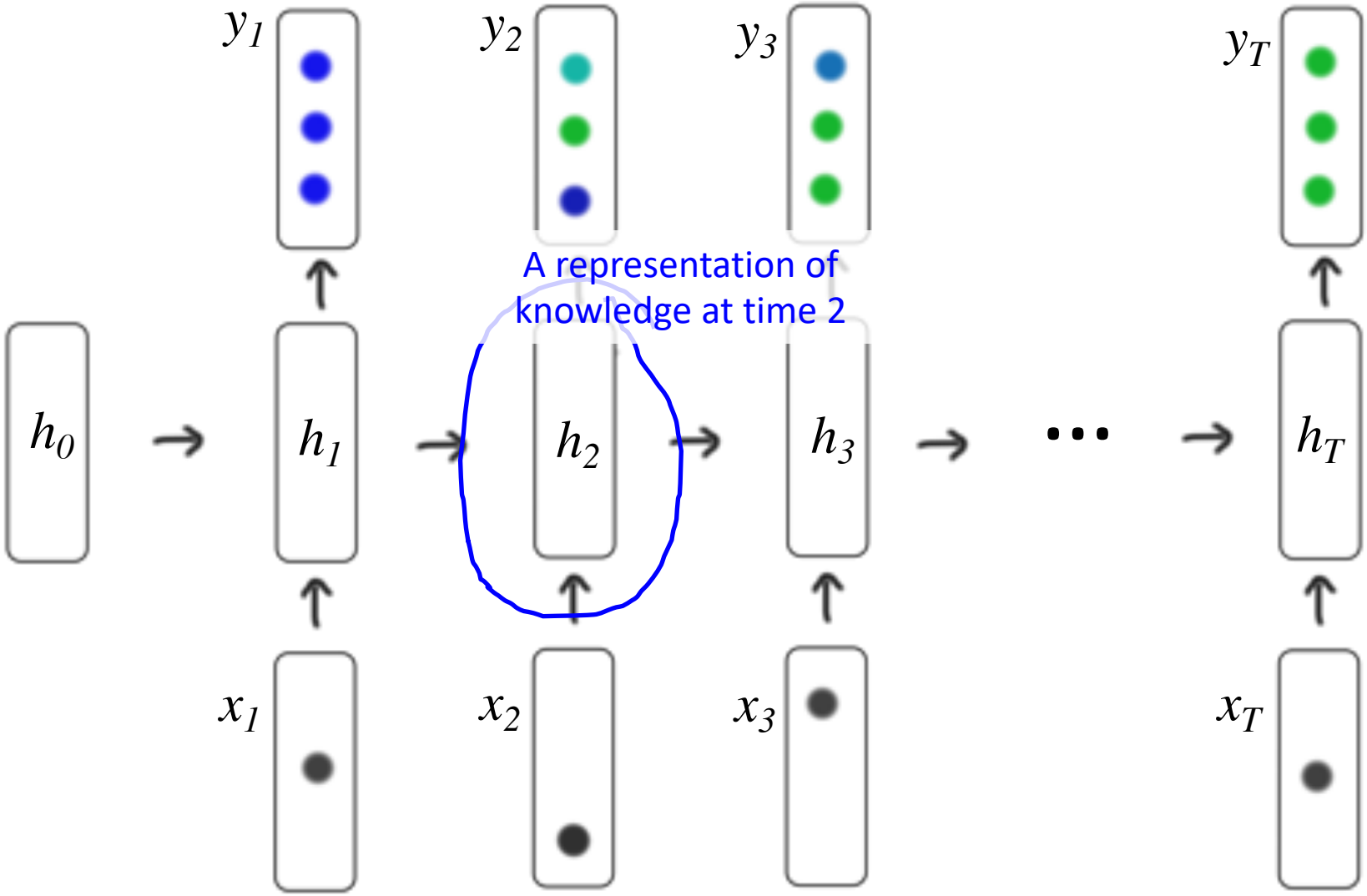
Deep Knowledge Tracing



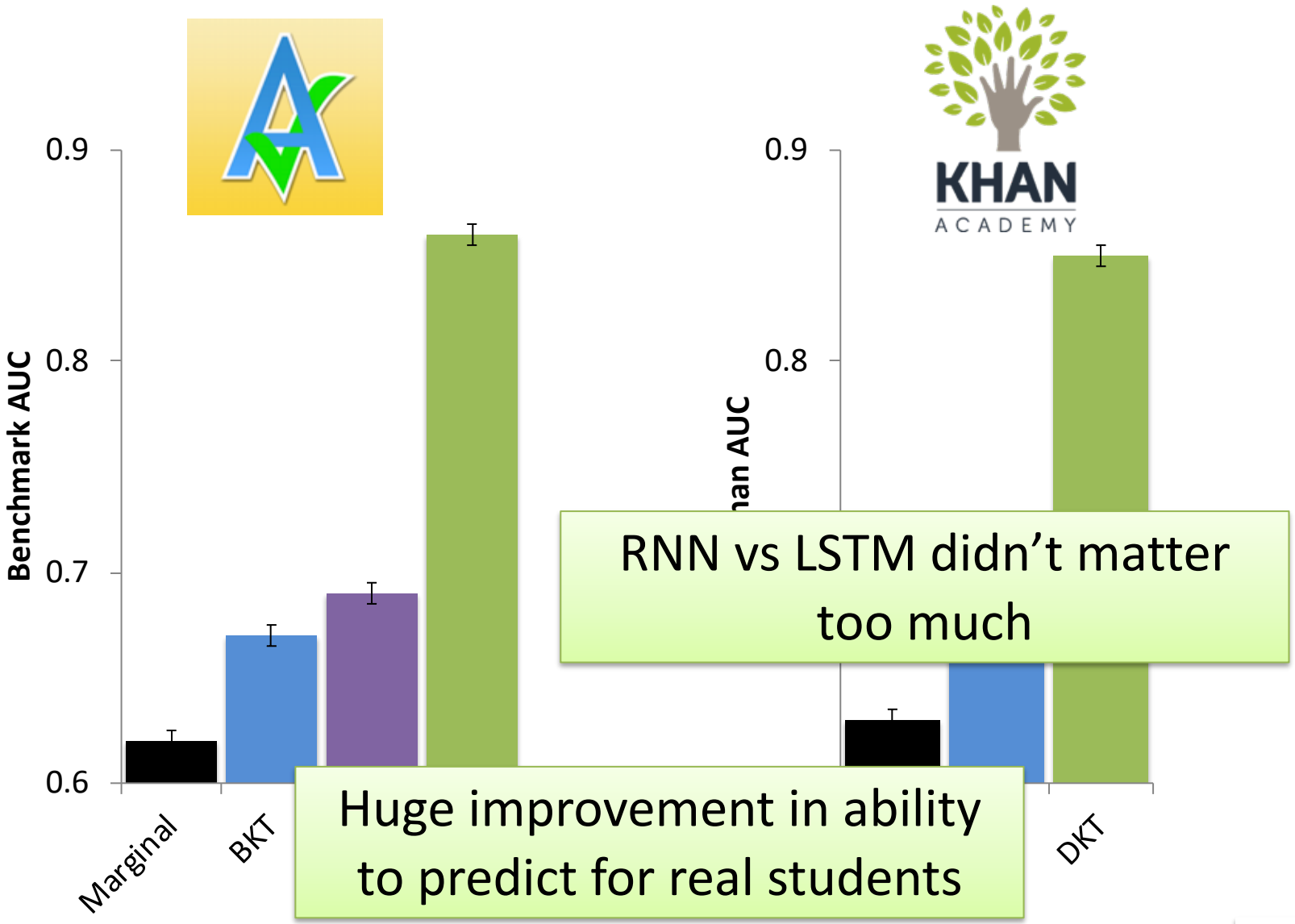
Recurrent Neural Network



Recurrent Neural Network



Prediction Results



Ethics in AI