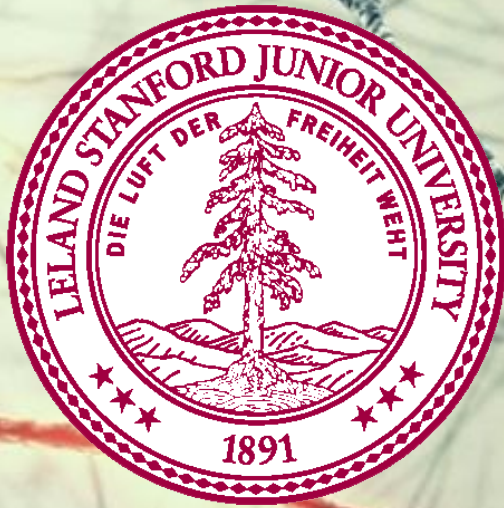




Deep Learning

Chris Piech
CS109, Stanford University

A Journey From Pure Math to Skin Cancer Detection

Logistic Regression is like the Harry Pottery Sorting Hat



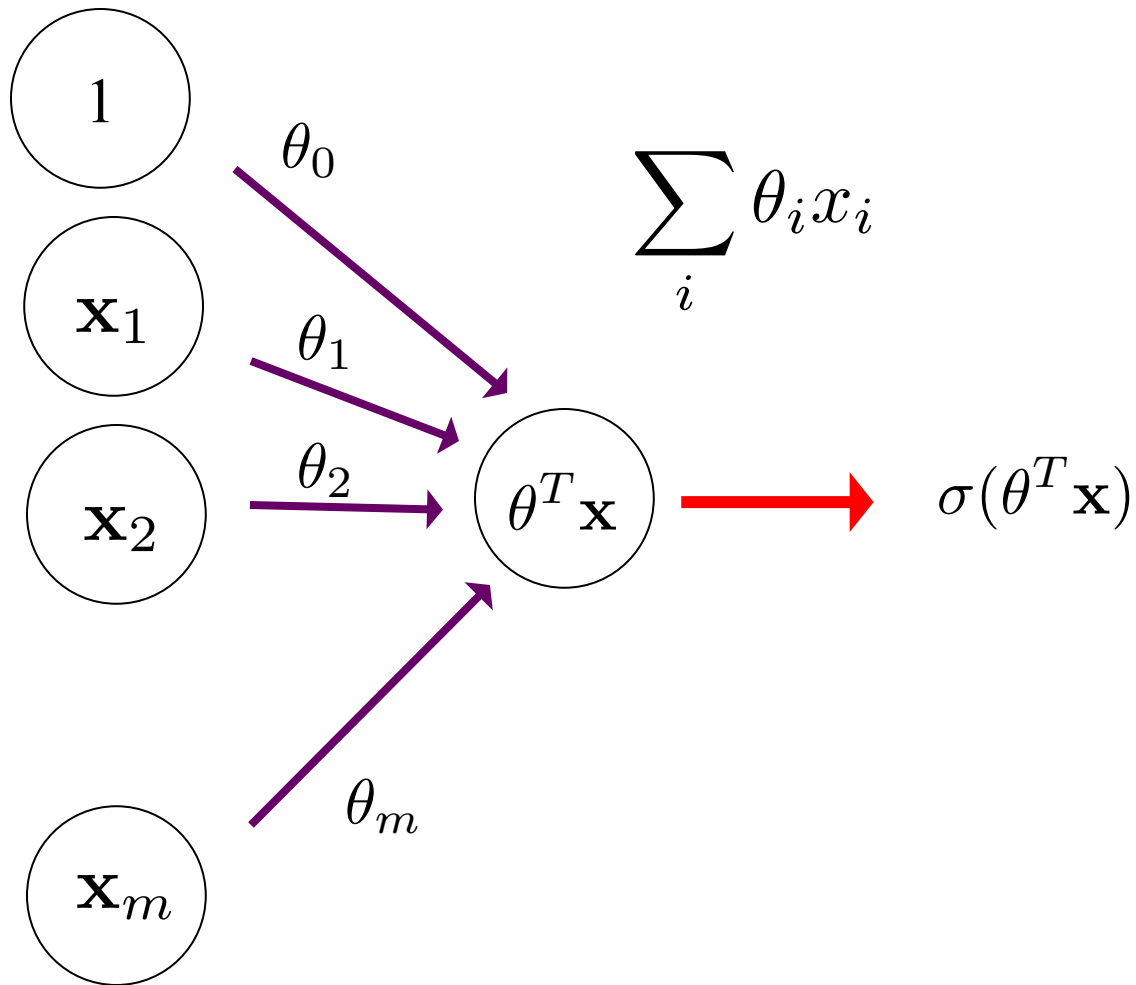
$$P(Y = 1) = 0.91$$

X

Logistic Regression is like the Harry Pottery Sorting Hat

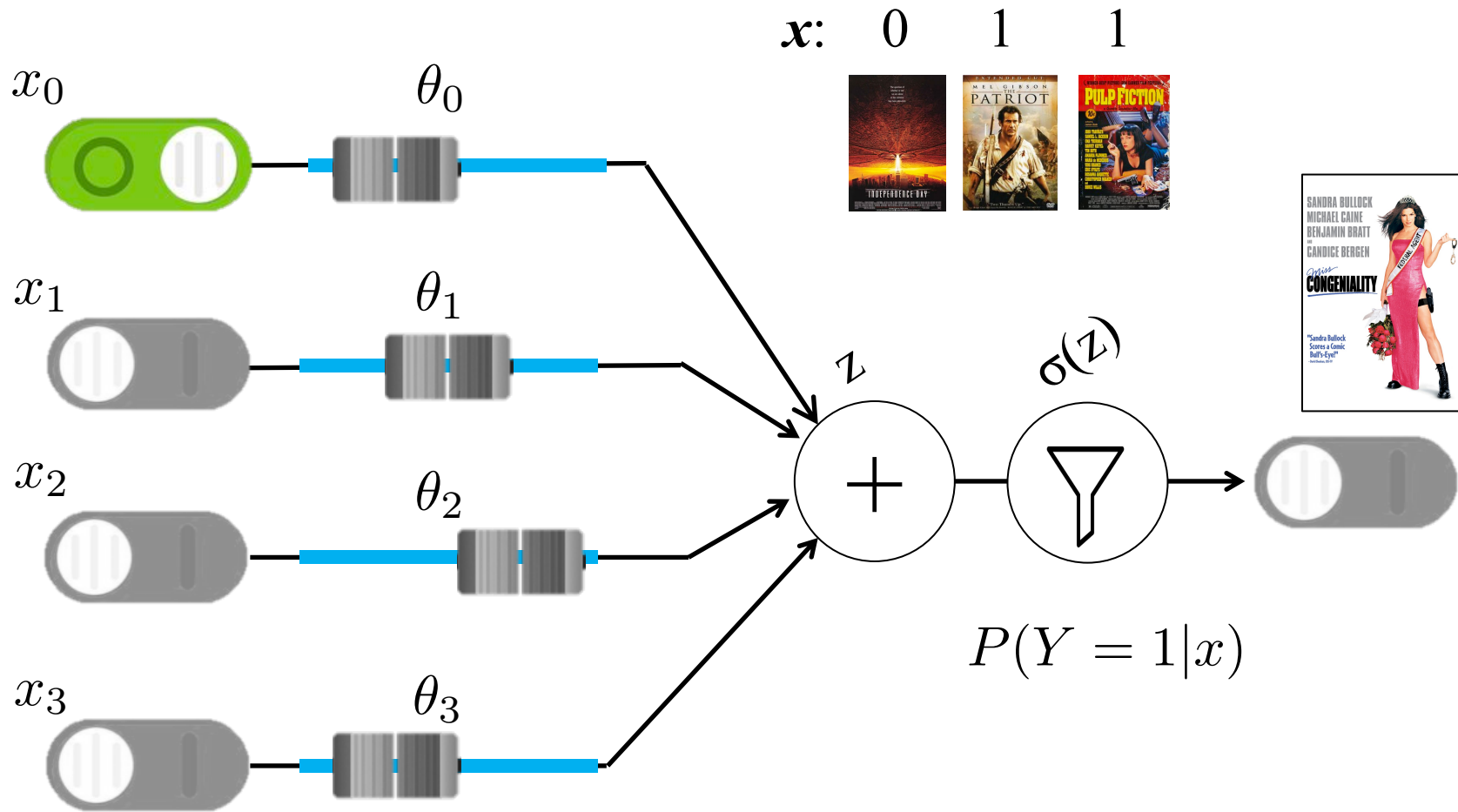


Logistic Regression



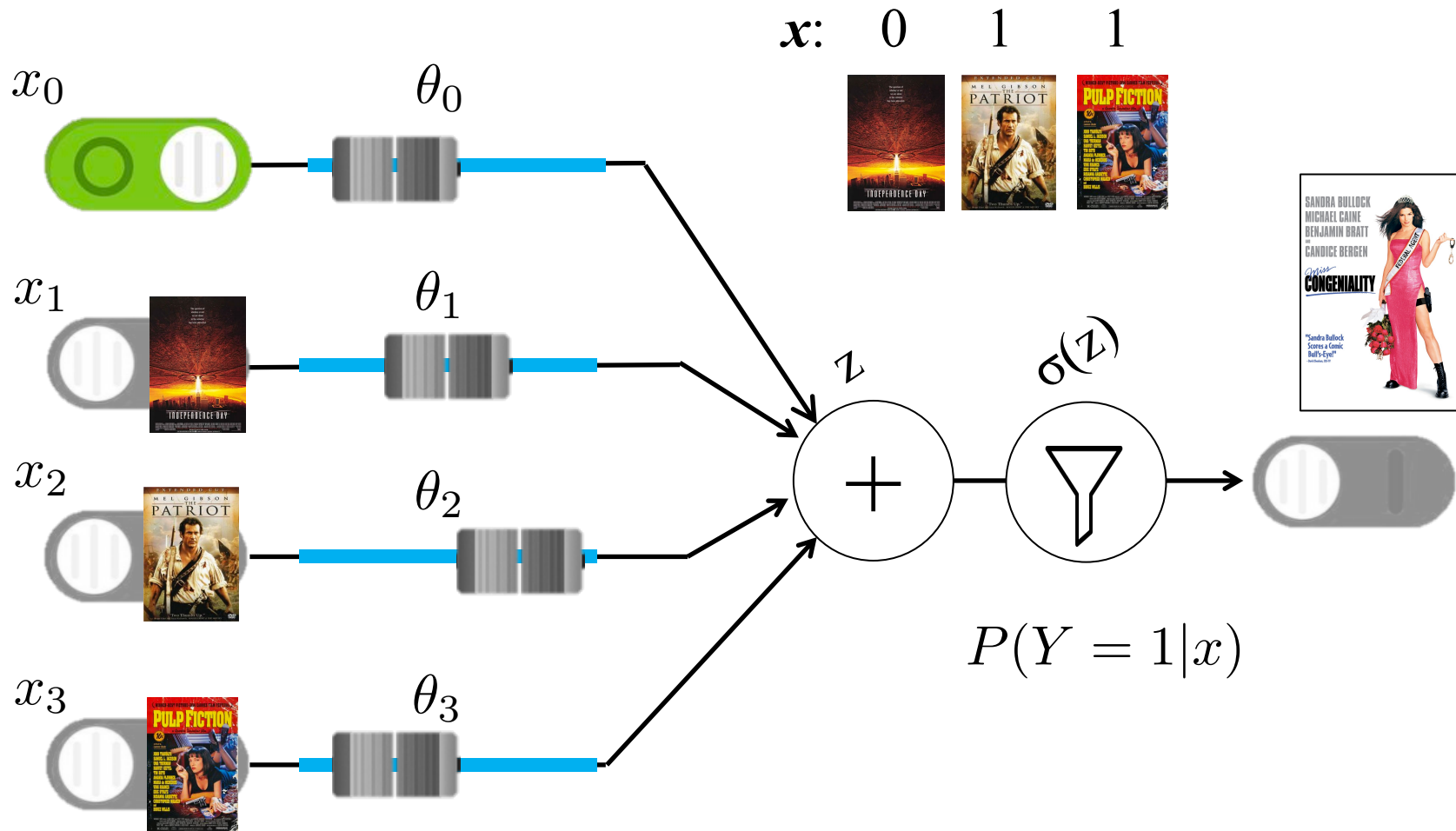
$$P(Y = 1 | X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



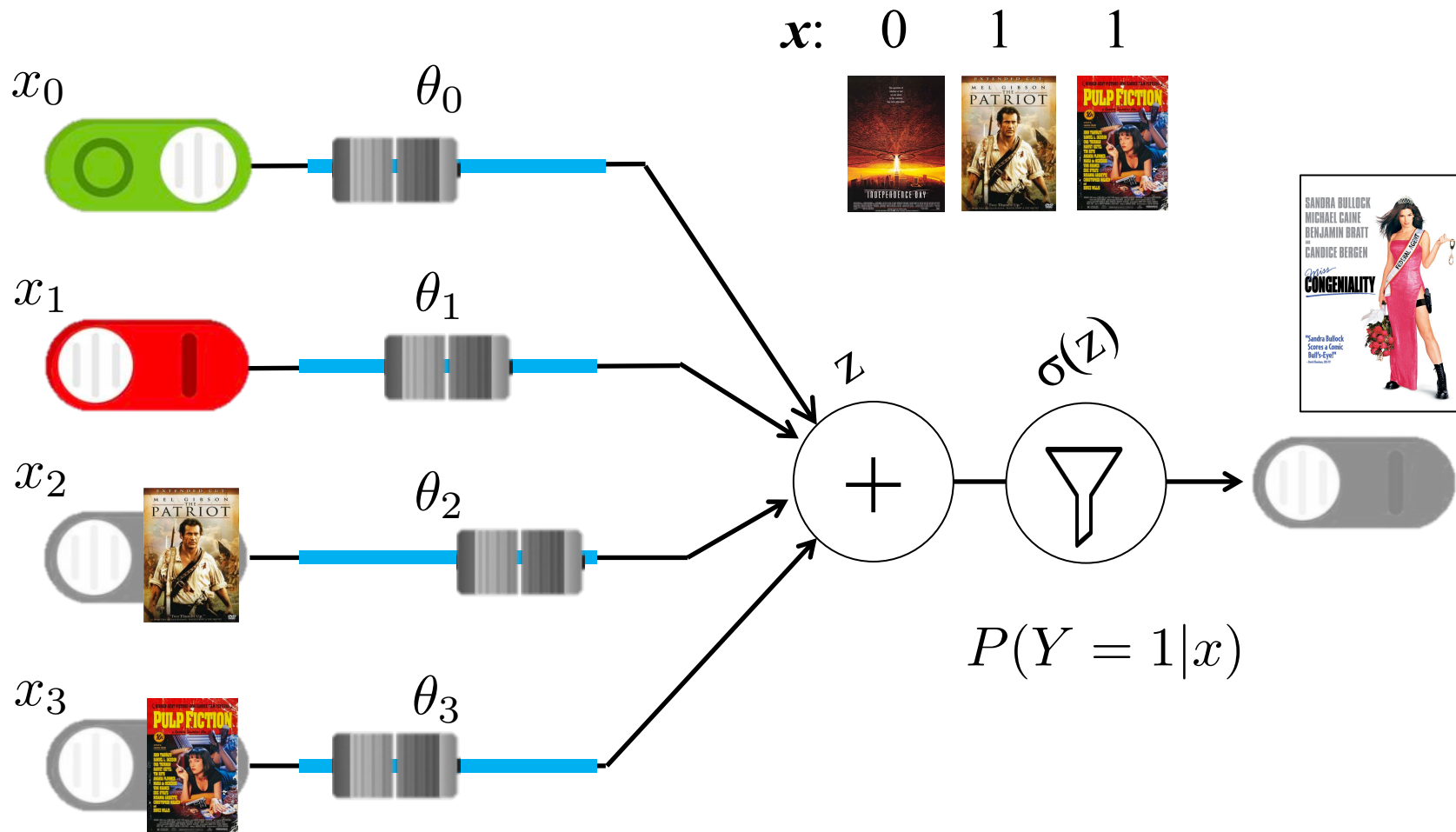
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



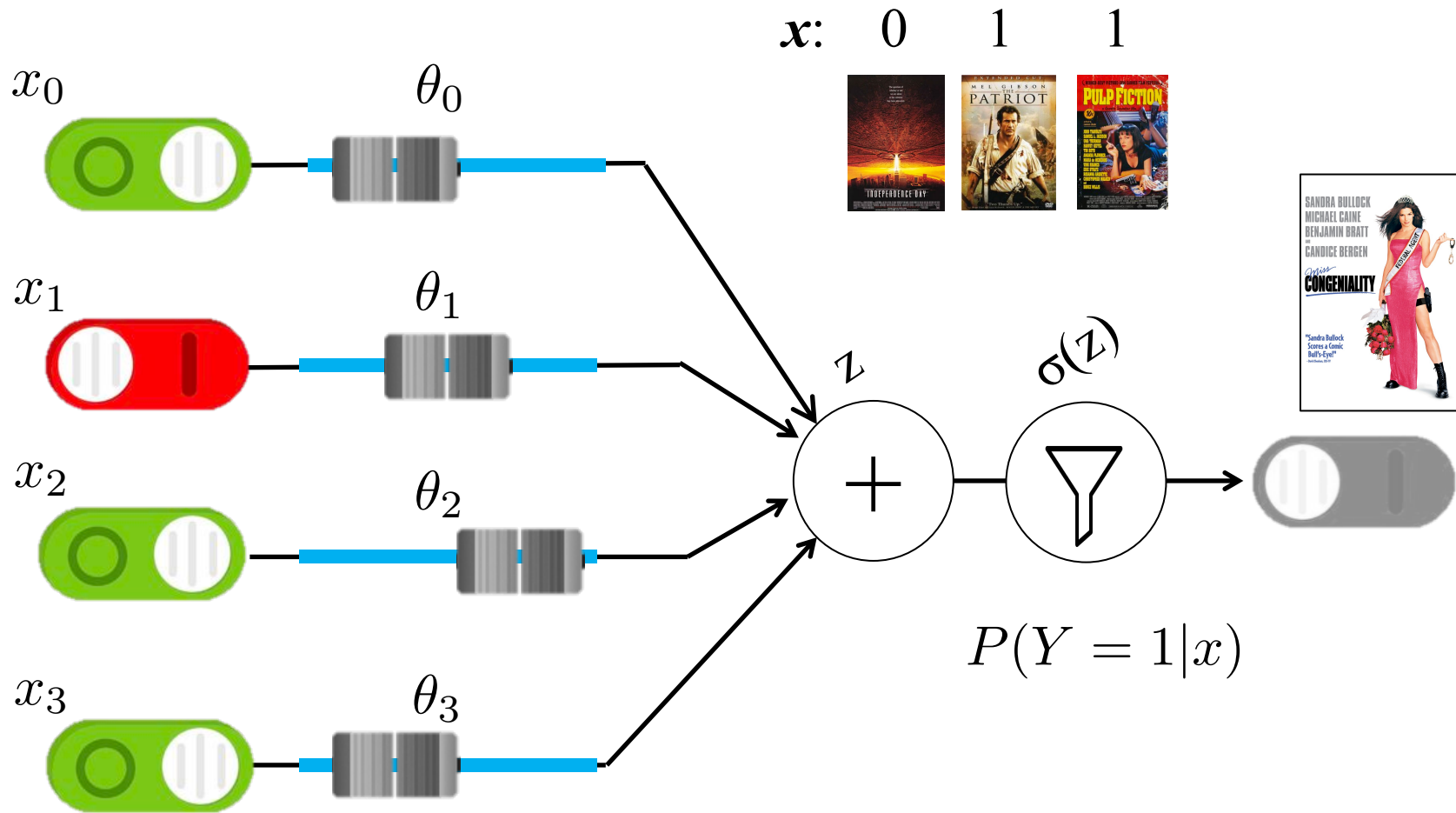
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



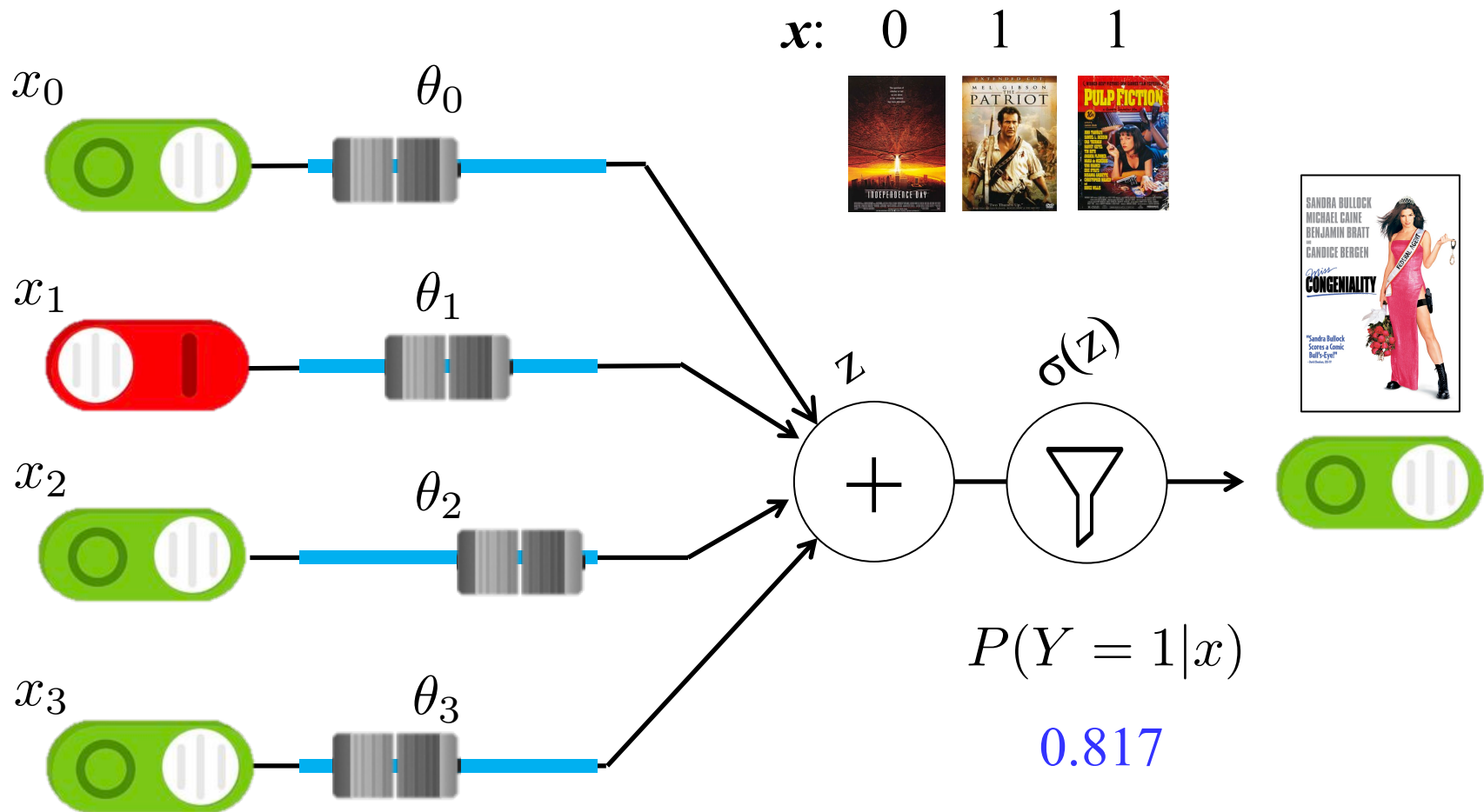
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



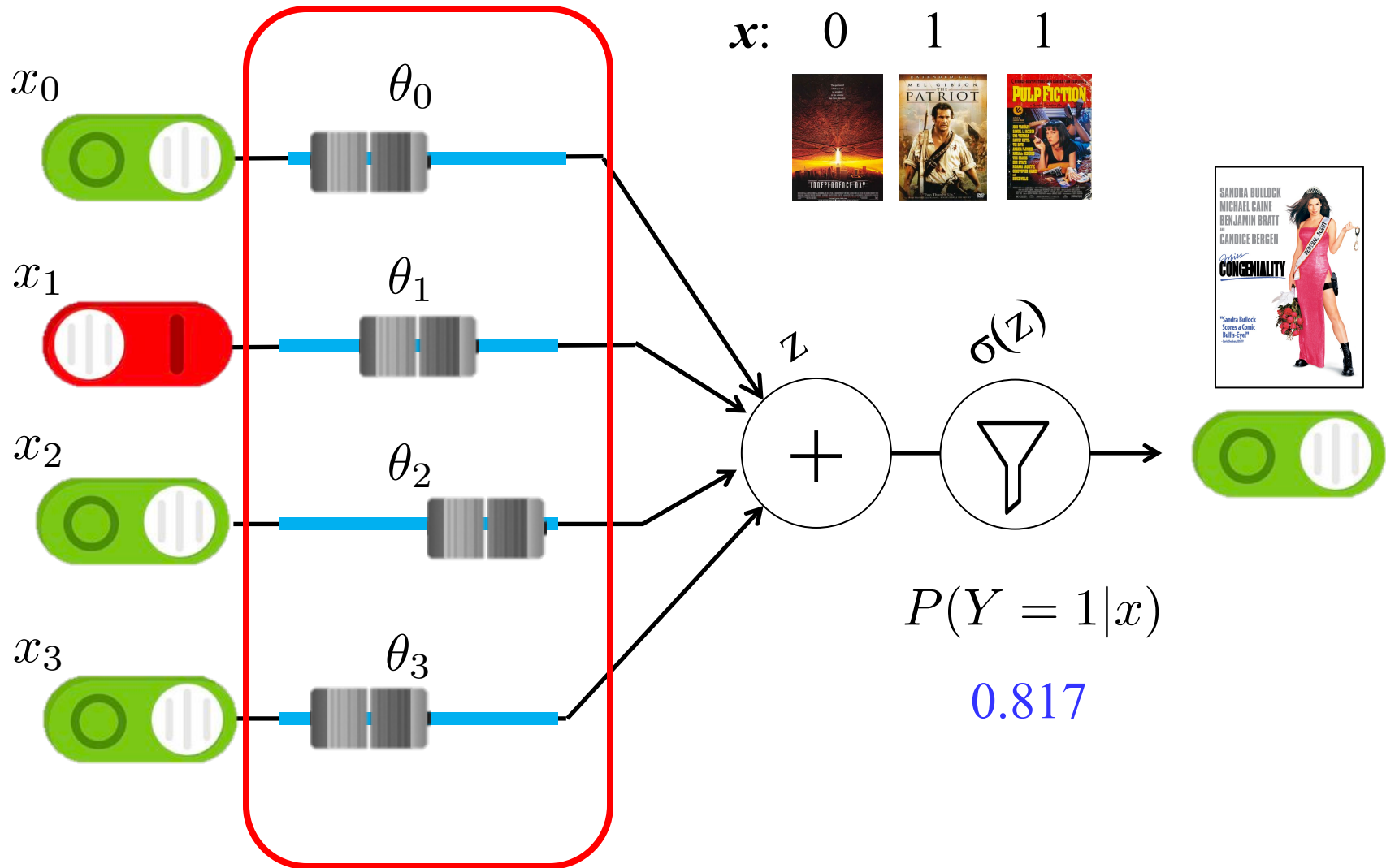
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



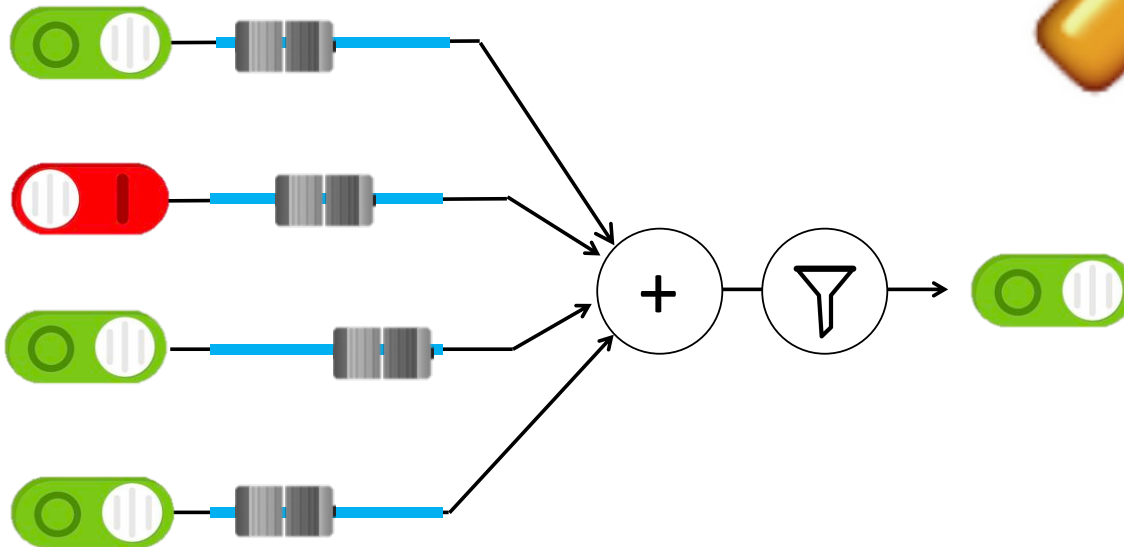
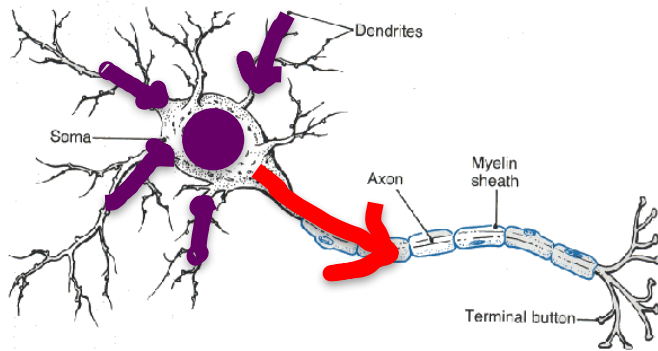
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



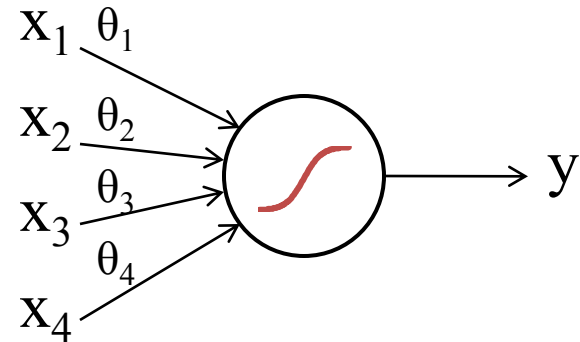
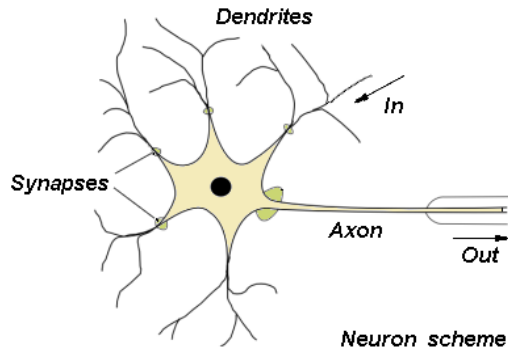
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Artificial Neurons

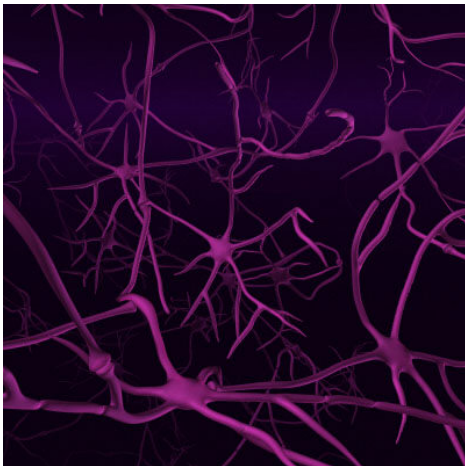


Biological Basis for Neural Networks

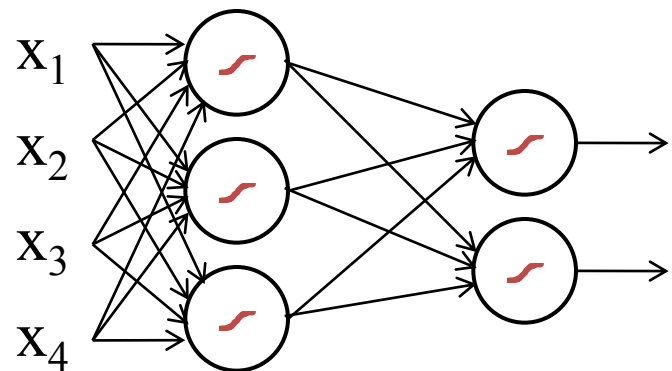
- A neuron



- Your brain



Actually, it's probably someone else's brain



Core idea behind the revolution in AI

Alpha GO



Self Driving Cars



Computers Making Art

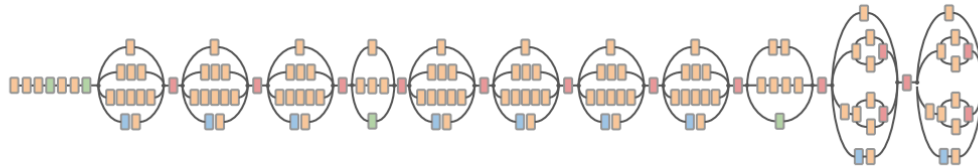


Detecting Skin Cancer

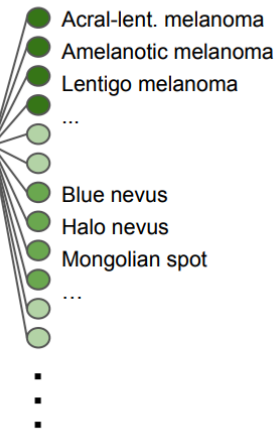
Skin Lesion Image



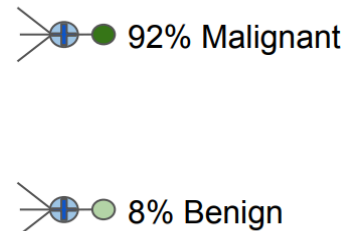
Deep Convolutional Neural Network (Inception-v3)



Training Classes (757)



Inference Classes (varies by task)

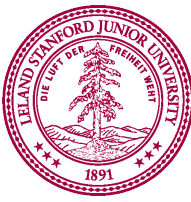


Esteva, Andre, et al. "Dermatologist-level classification of skin cancer with deep neural networks." *Nature* 542.7639 (2017): 115-118.

(aka Neural Networks)

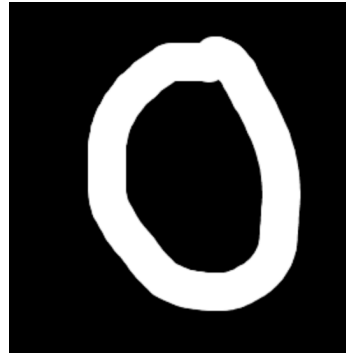


Deep learning is (at its core) many logistic regression pieces stacked on top of each other.



Digit Recognition Example

Let's make feature vectors from pictures of numbers



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

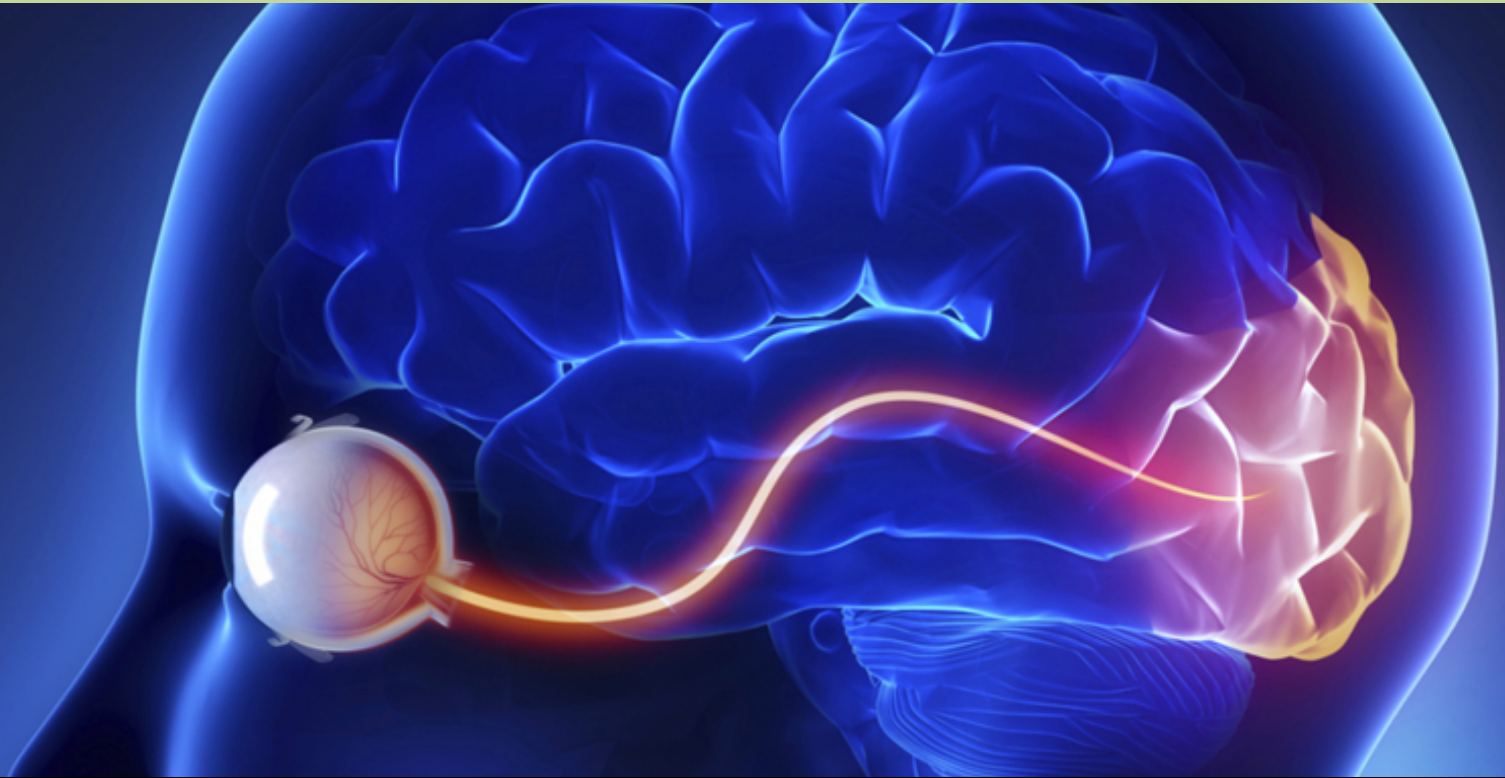


$$\mathbf{x}^{(i)} = [0, 0, 1, 1, \dots, 0, 1, 1, 0, \dots 0, 1, 0, 0]$$
$$y^{(i)} = 1$$

Computer Vision



Vision in your Brain

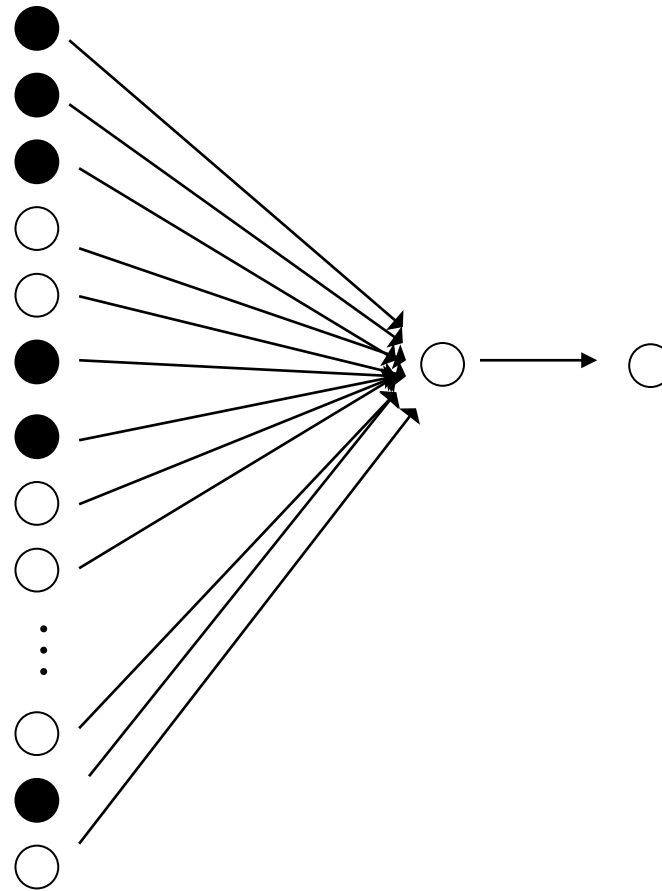
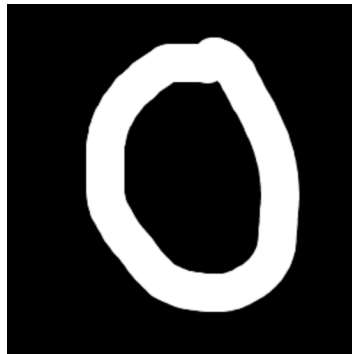


Hundreds of millions of neurons [1]

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

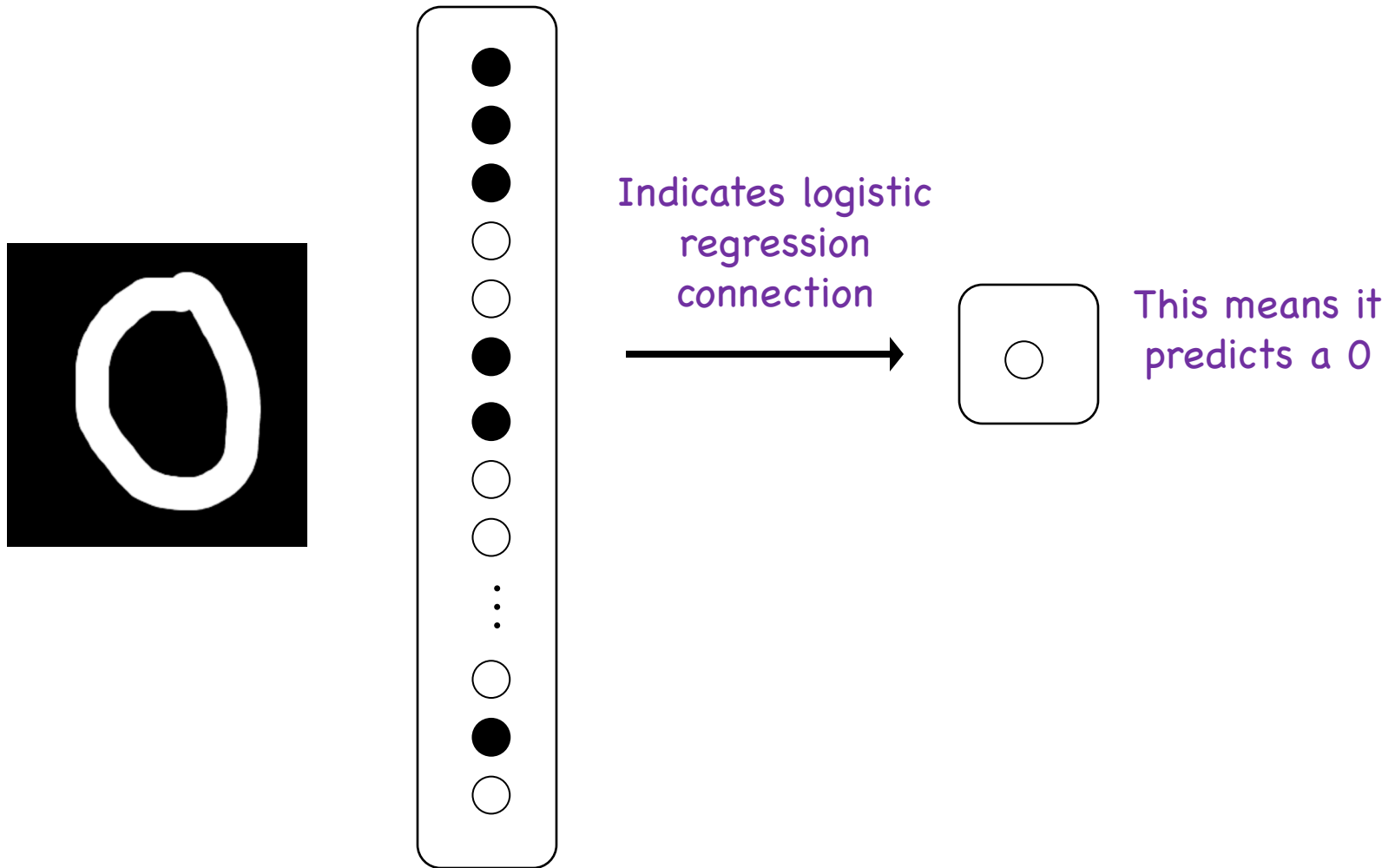
[1] <http://discovermagazine.com/1993/jun/thevisionthingma227>

Logistic Regression

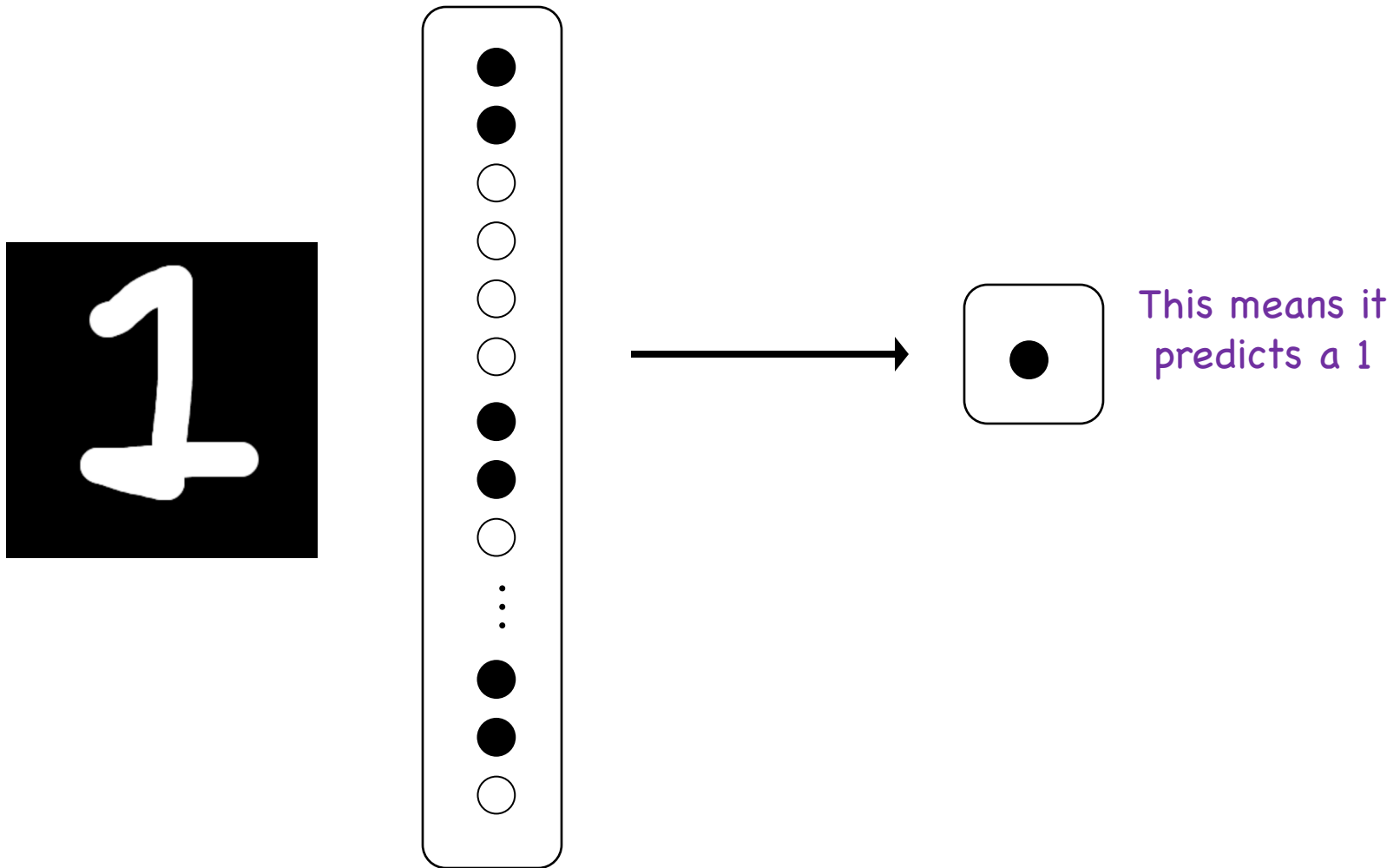


This means it
predicts a 0

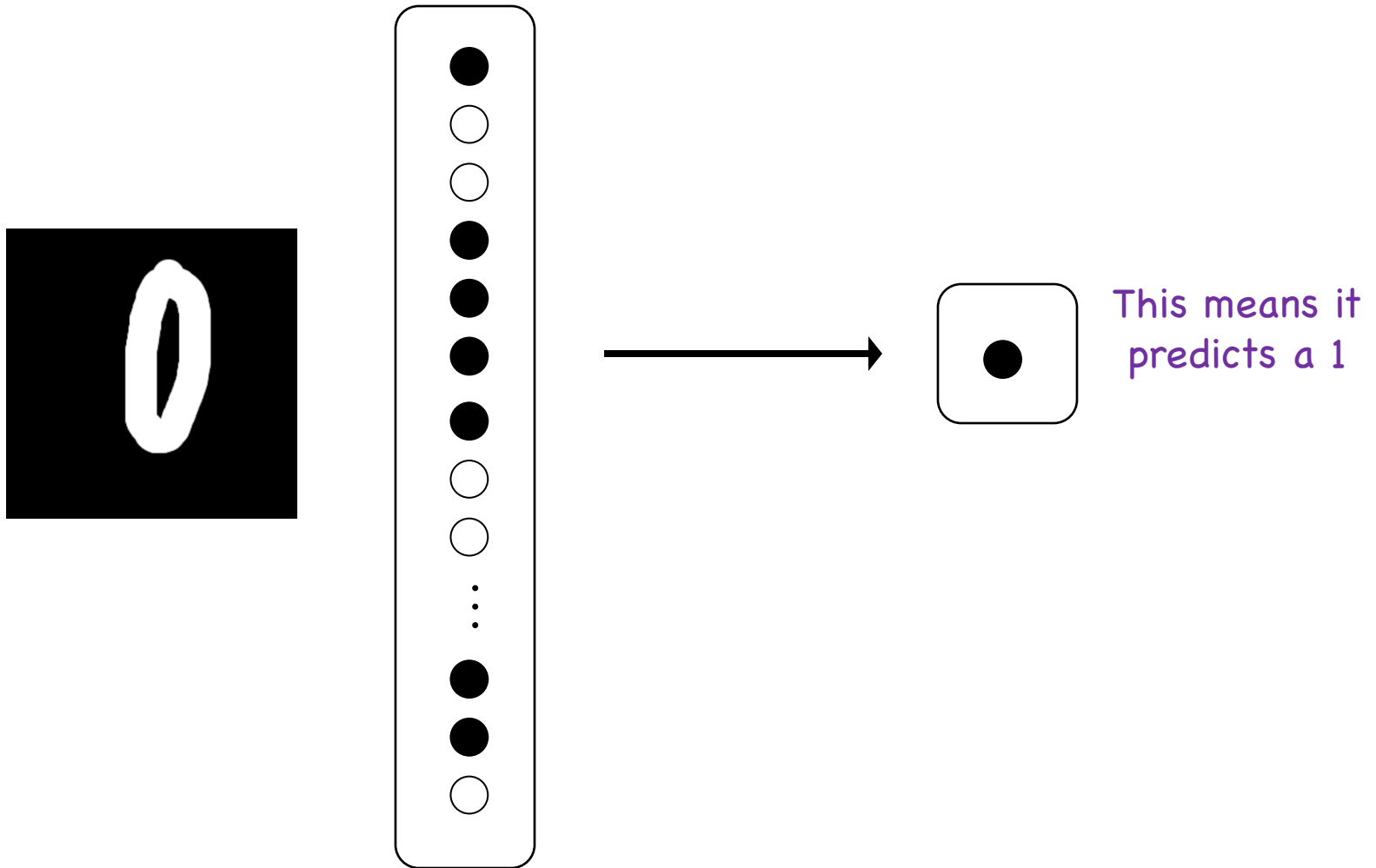
Logistic Regression



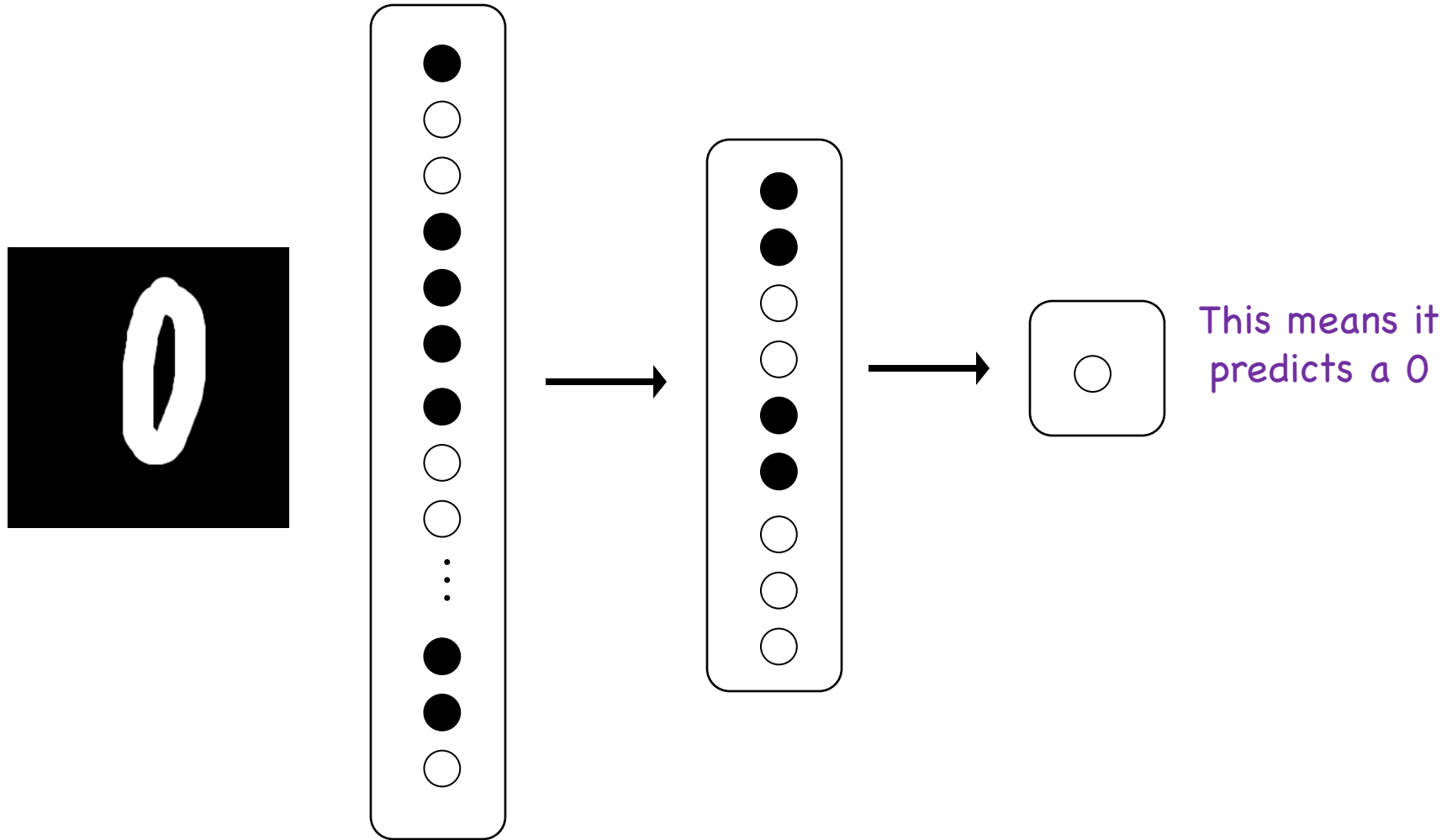
Logistic Regression



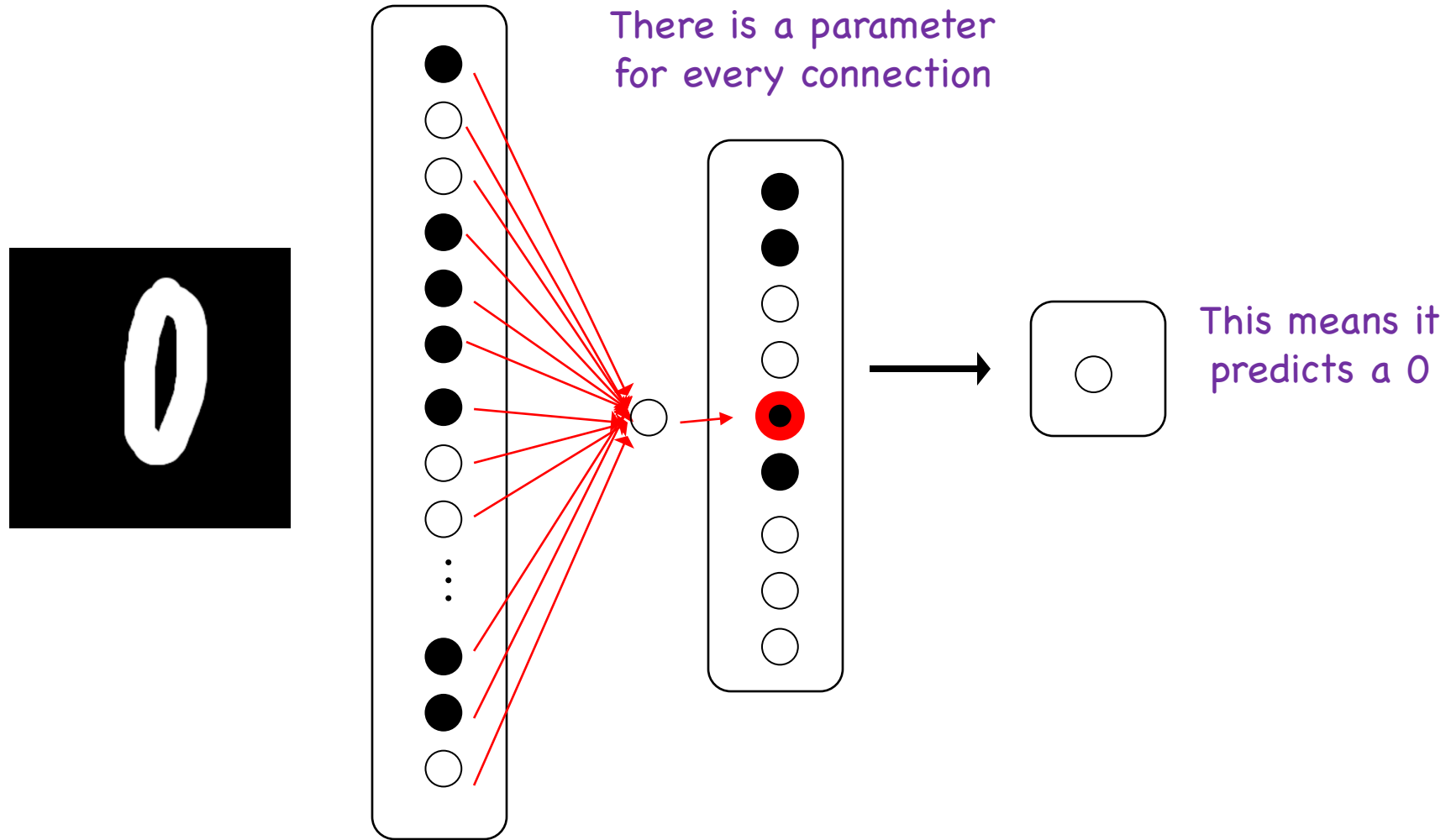
Not So Good



We Can Put Neurons Together

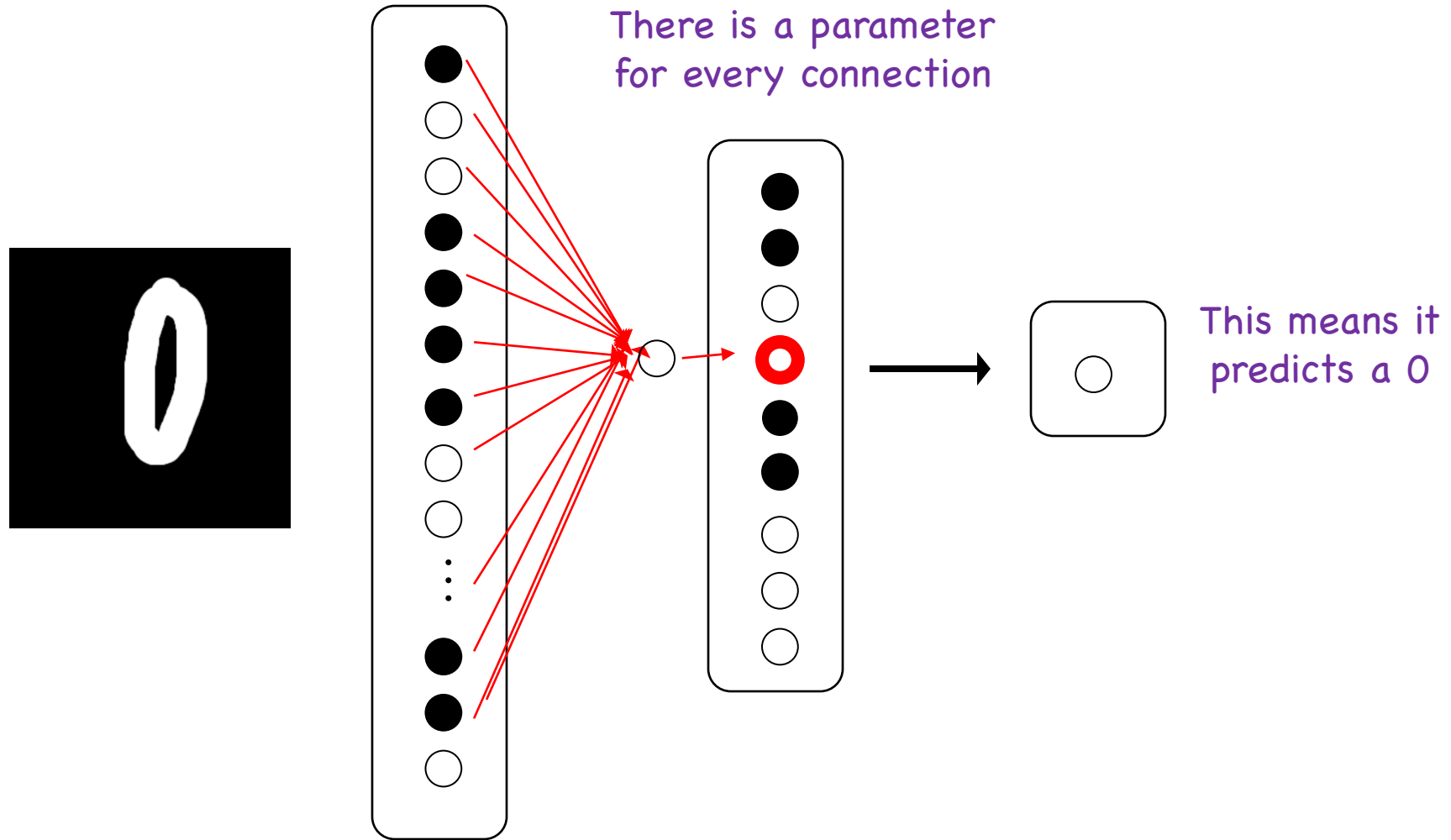


We Can Put Neurons Together



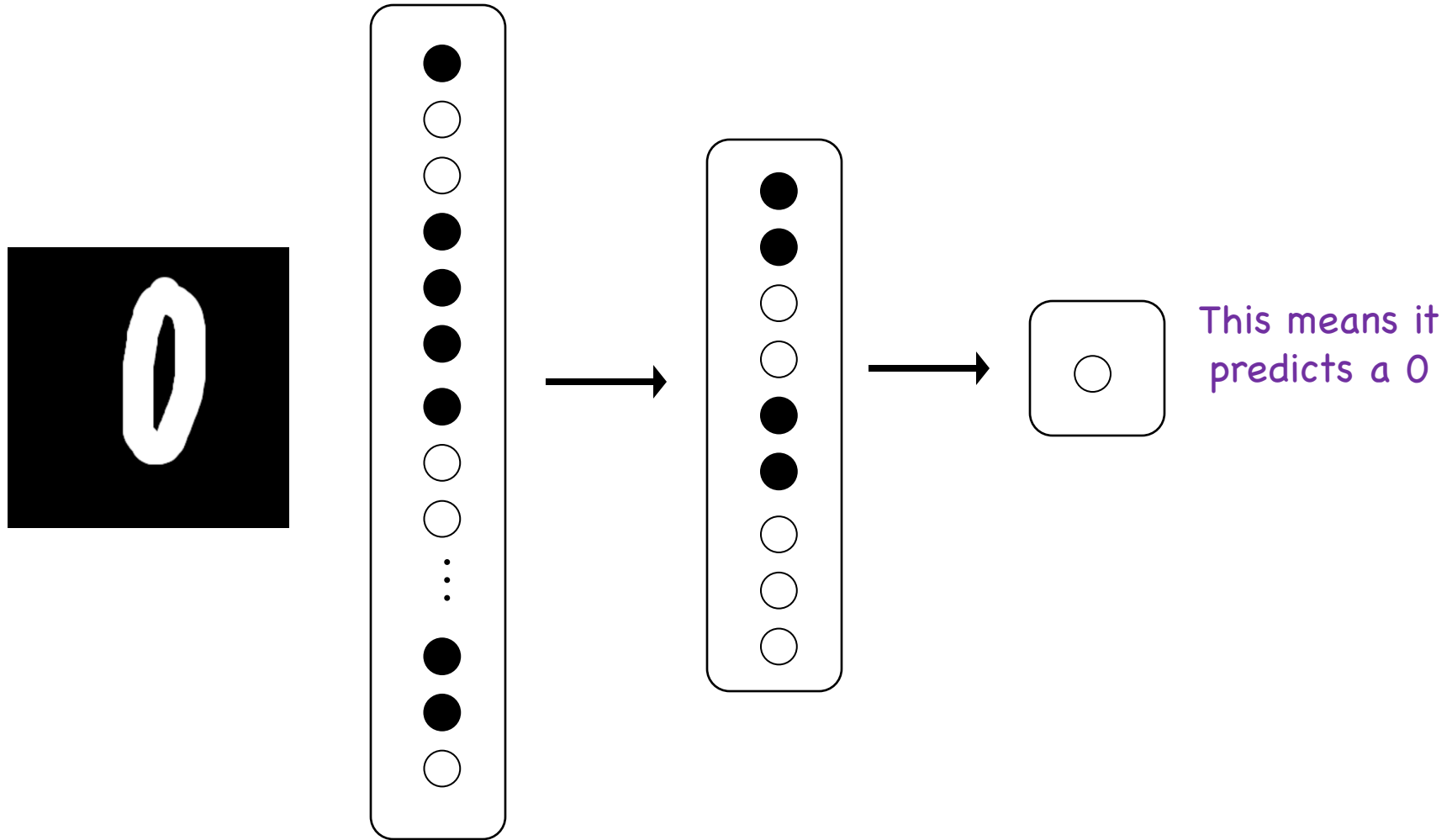
Look at a single “hidden” neuron

We Can Put Neurons Together

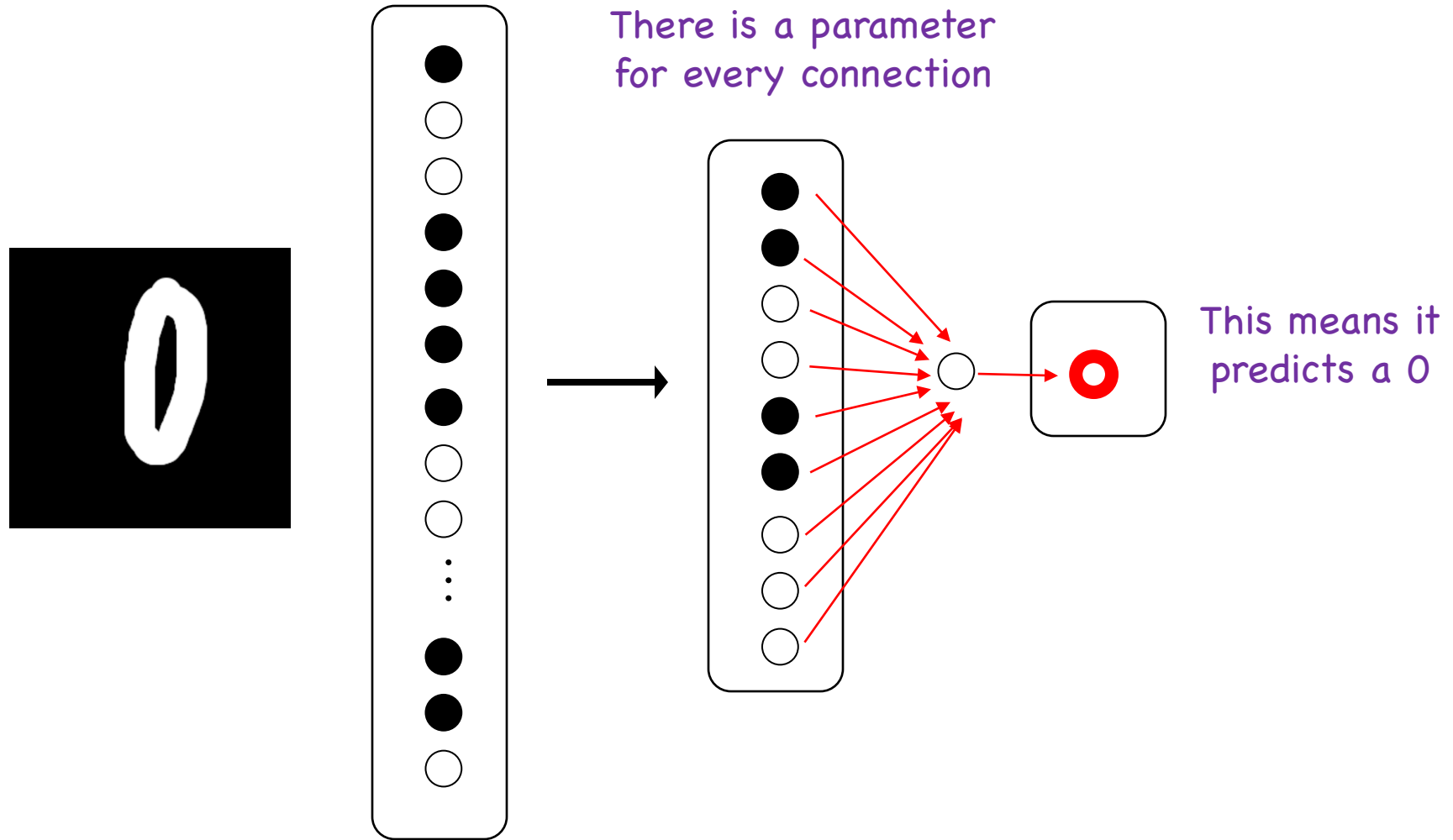


Look at another “hidden” neuron

We Can Put Neurons Together

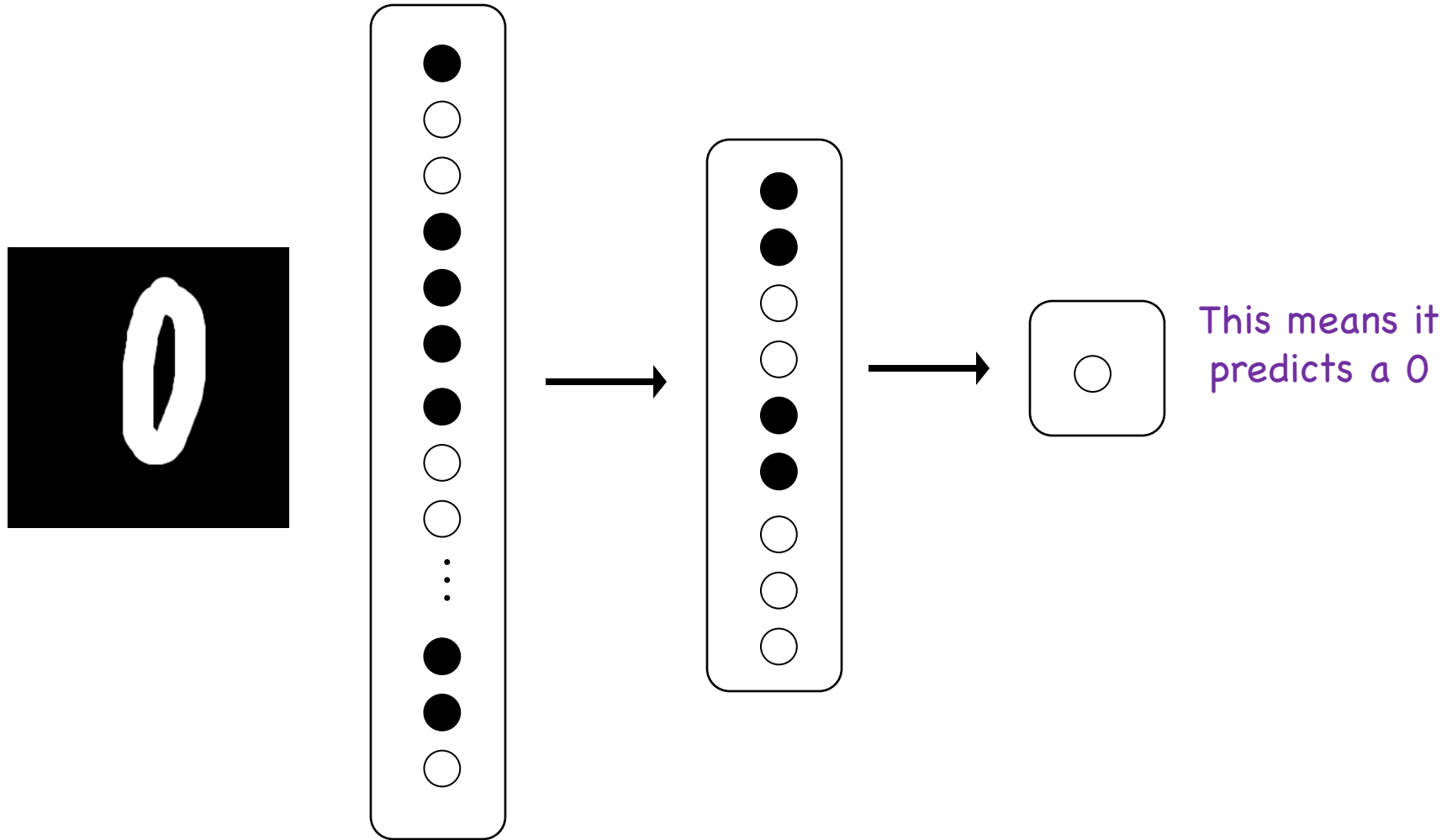


We Can Put Neurons Together

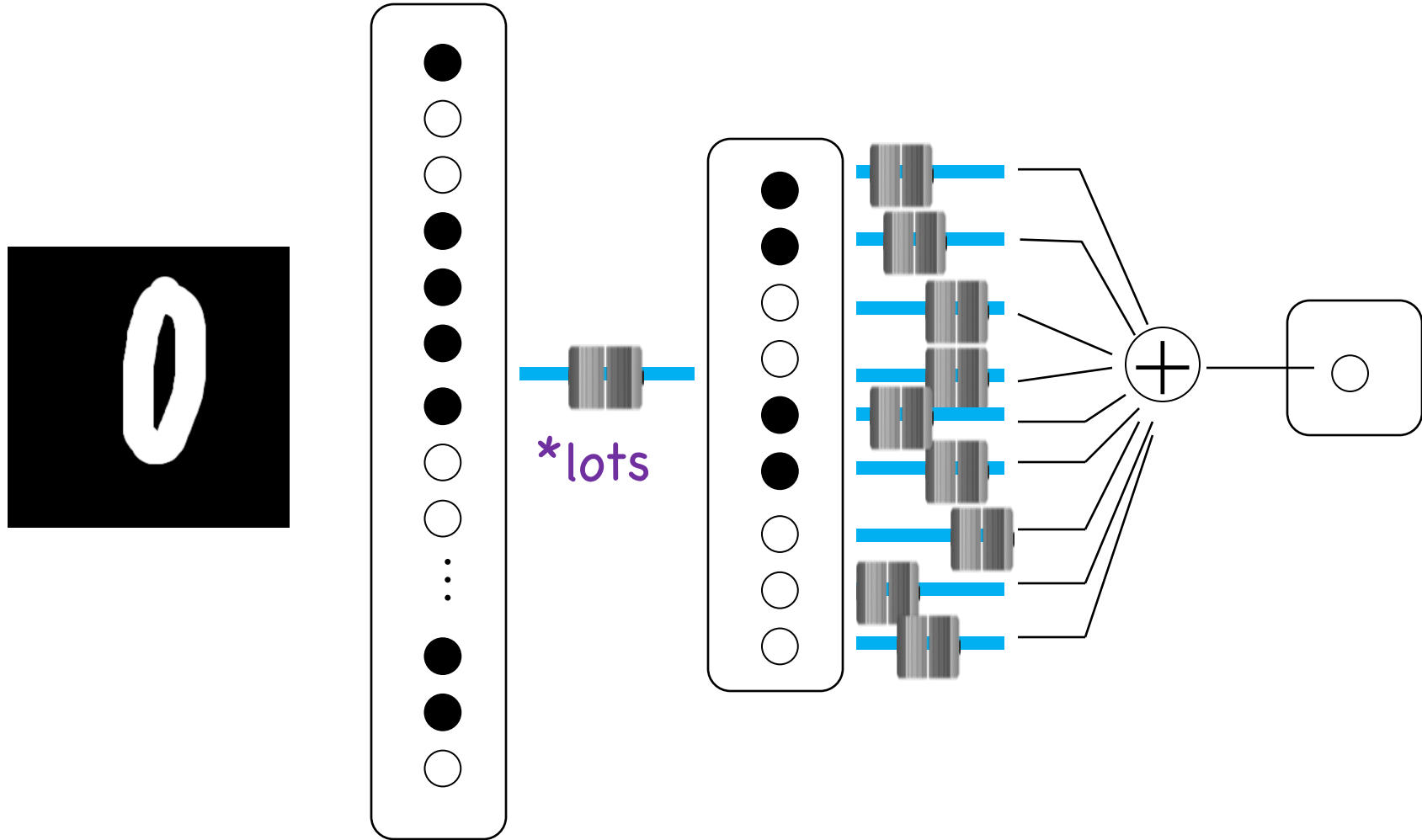


Look at another neuron

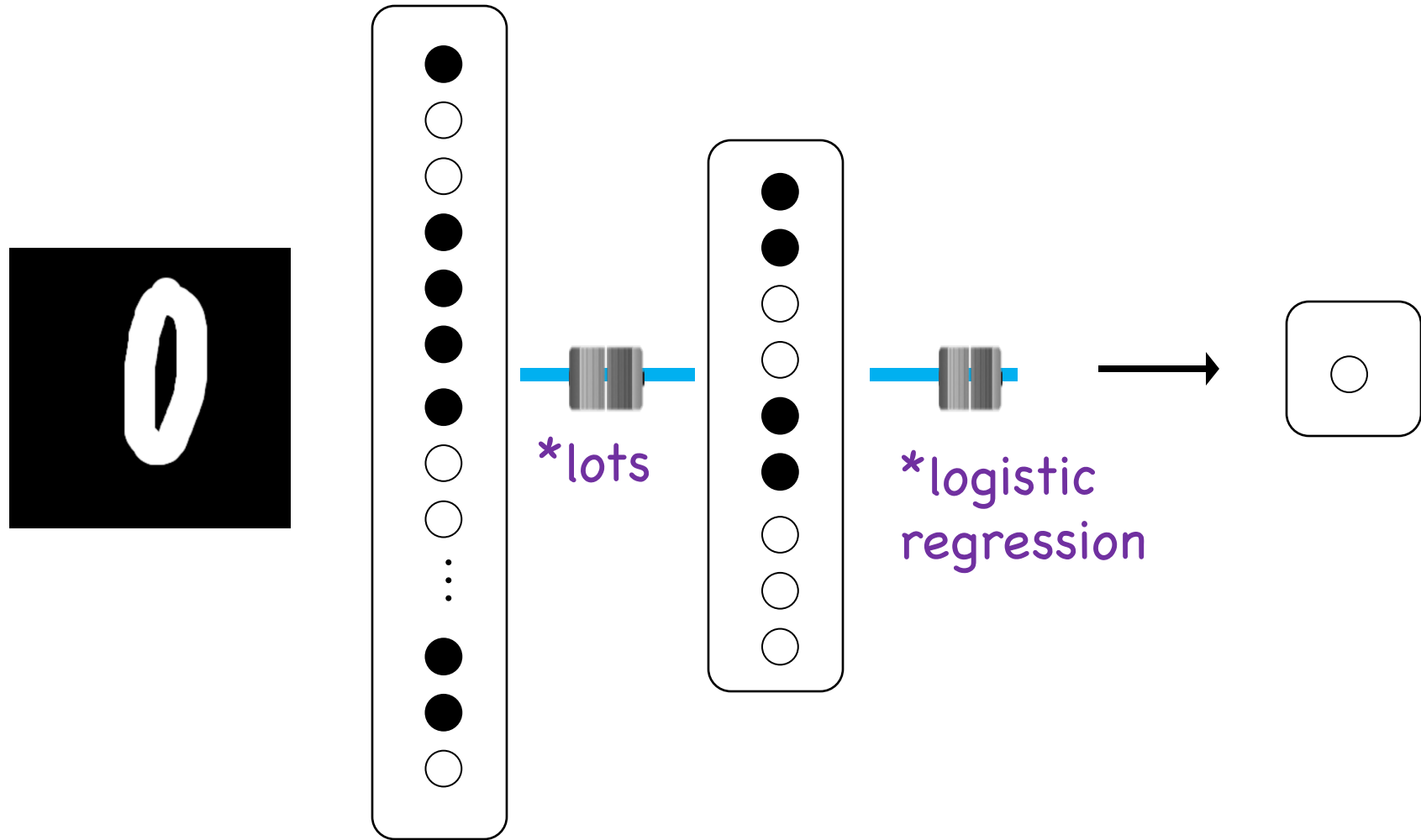
We Can Put Neurons Together



We Can Put Neurons Together

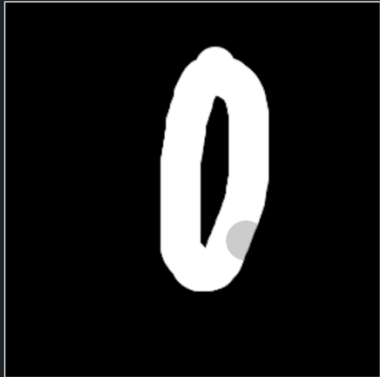


We Can Put Neurons Together



Demonstration

Draw your number here



Downsampled drawing: 0

First guess: 0

Second guess: 8

Layer visibility

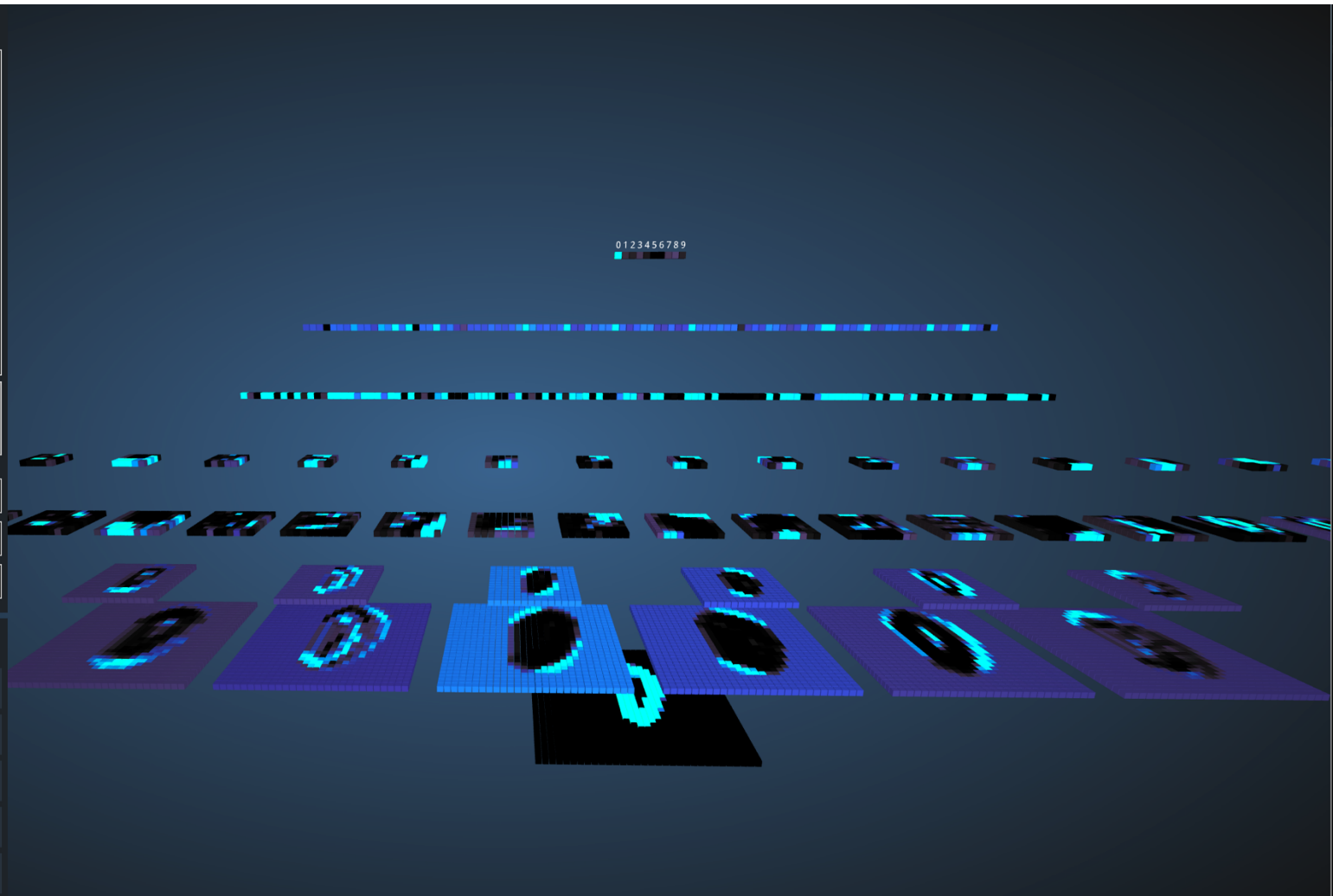
Input layer

Convolution layer 1

Downsampling layer 1

Convolution layer 2

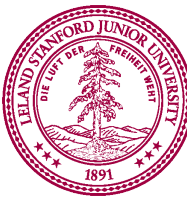
Downsampling layer 2



<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...

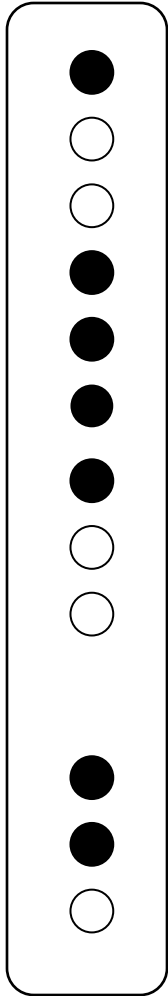
1. Understand Chain Rule as ♥ of Deep Learning

2. Everyone should be able
to do simple derivations

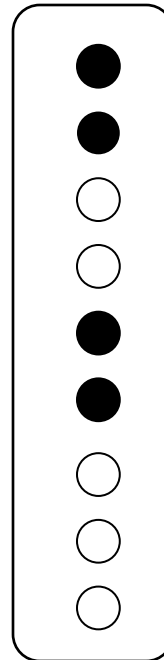
Math worth knowing:

New Notation

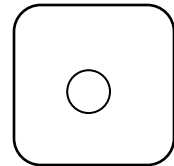
Layer x



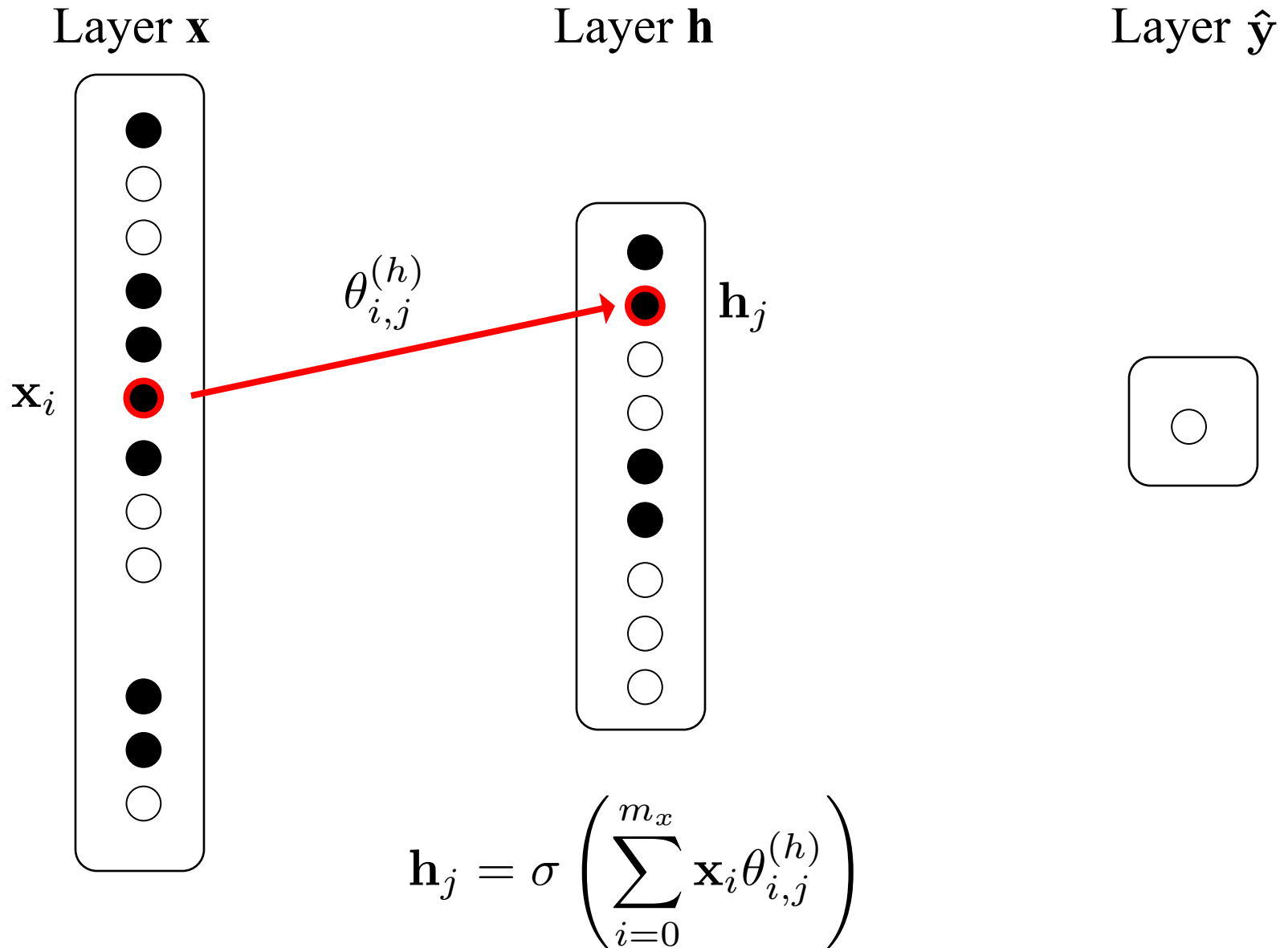
Layer h



Layer $\hat{y}$

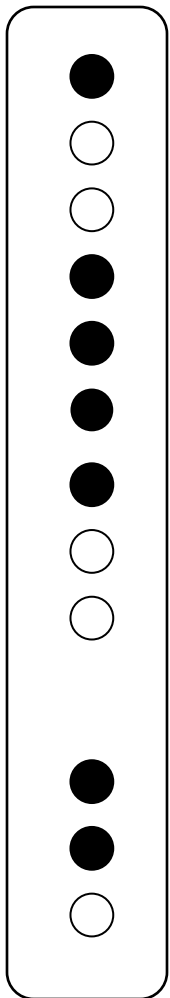


New Notation

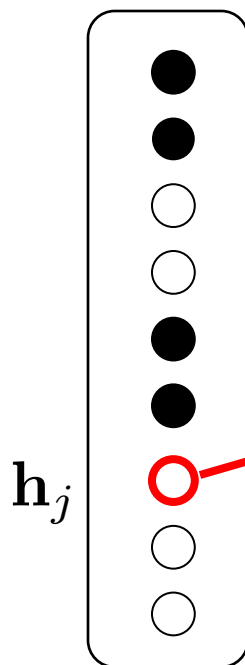


New Notation

Layer $\mathbf{x}$



Layer $\mathbf{h}$



Layer $\hat{\mathbf{y}}$



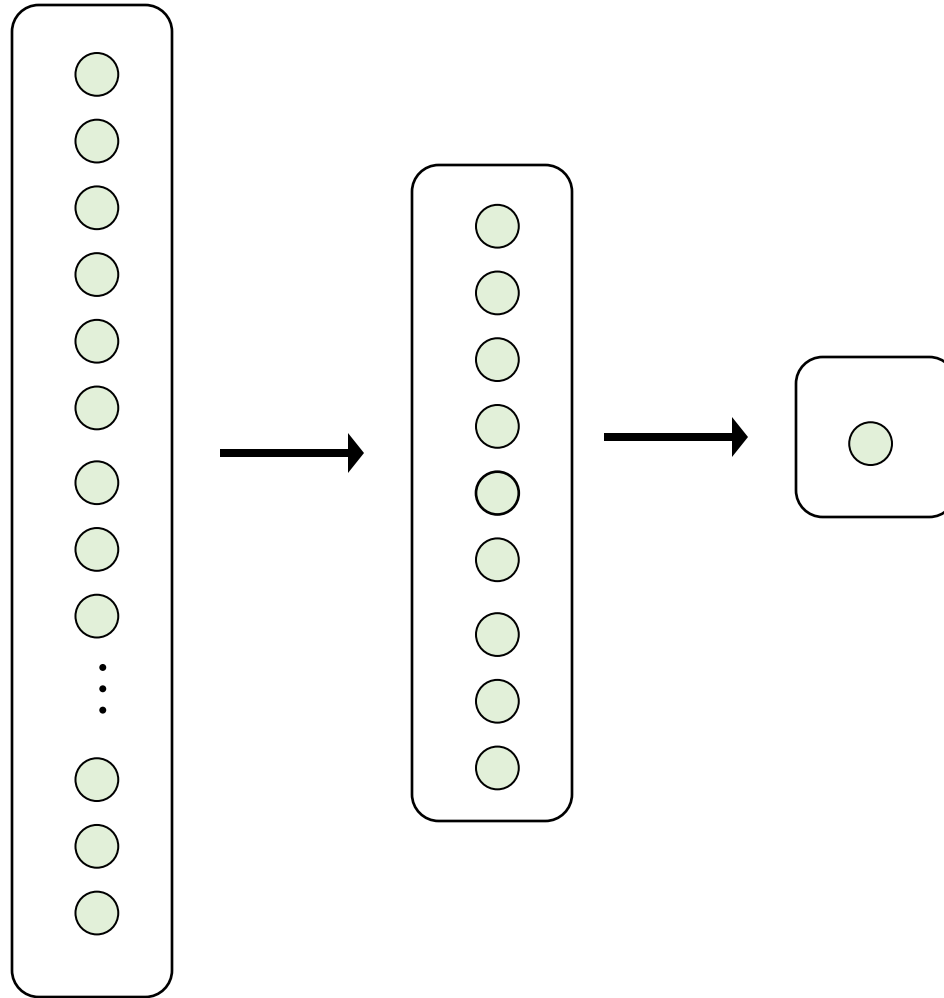
$$\hat{y} = \sigma \left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{\mathbf{y}})} \right)$$

Forward Pass

Layer x

Layer h

Layer $\hat{y}$

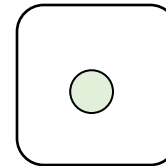
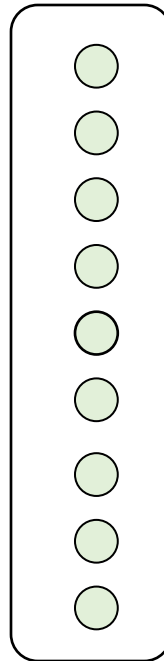
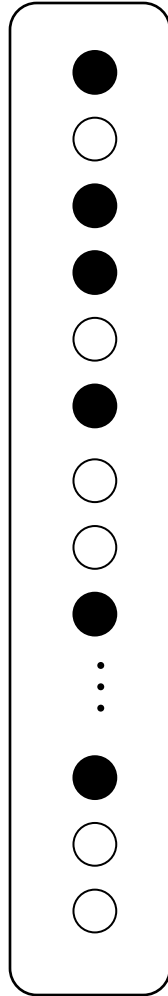


Forward Pass

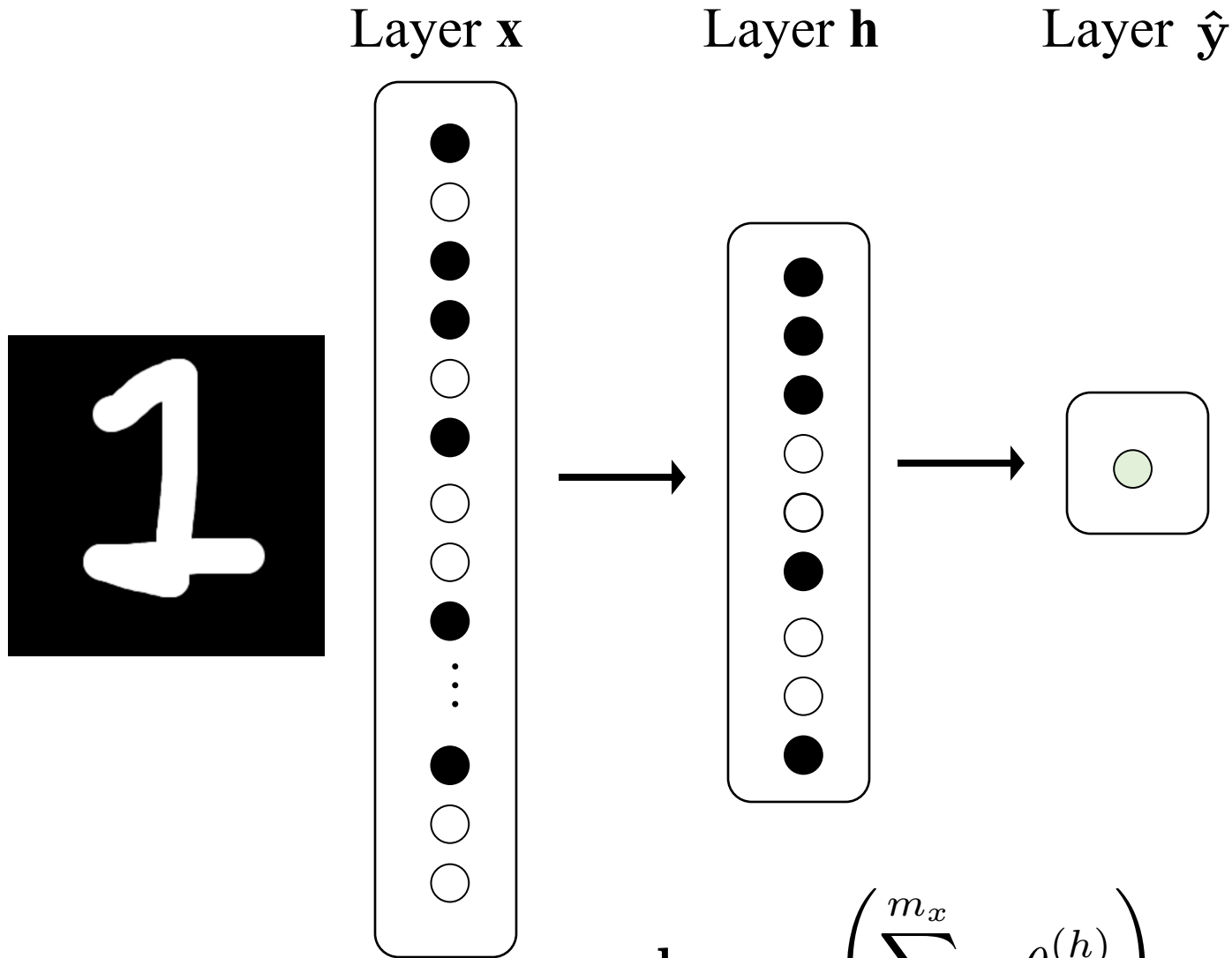
Layer x

Layer h

Layer $\hat{y}$

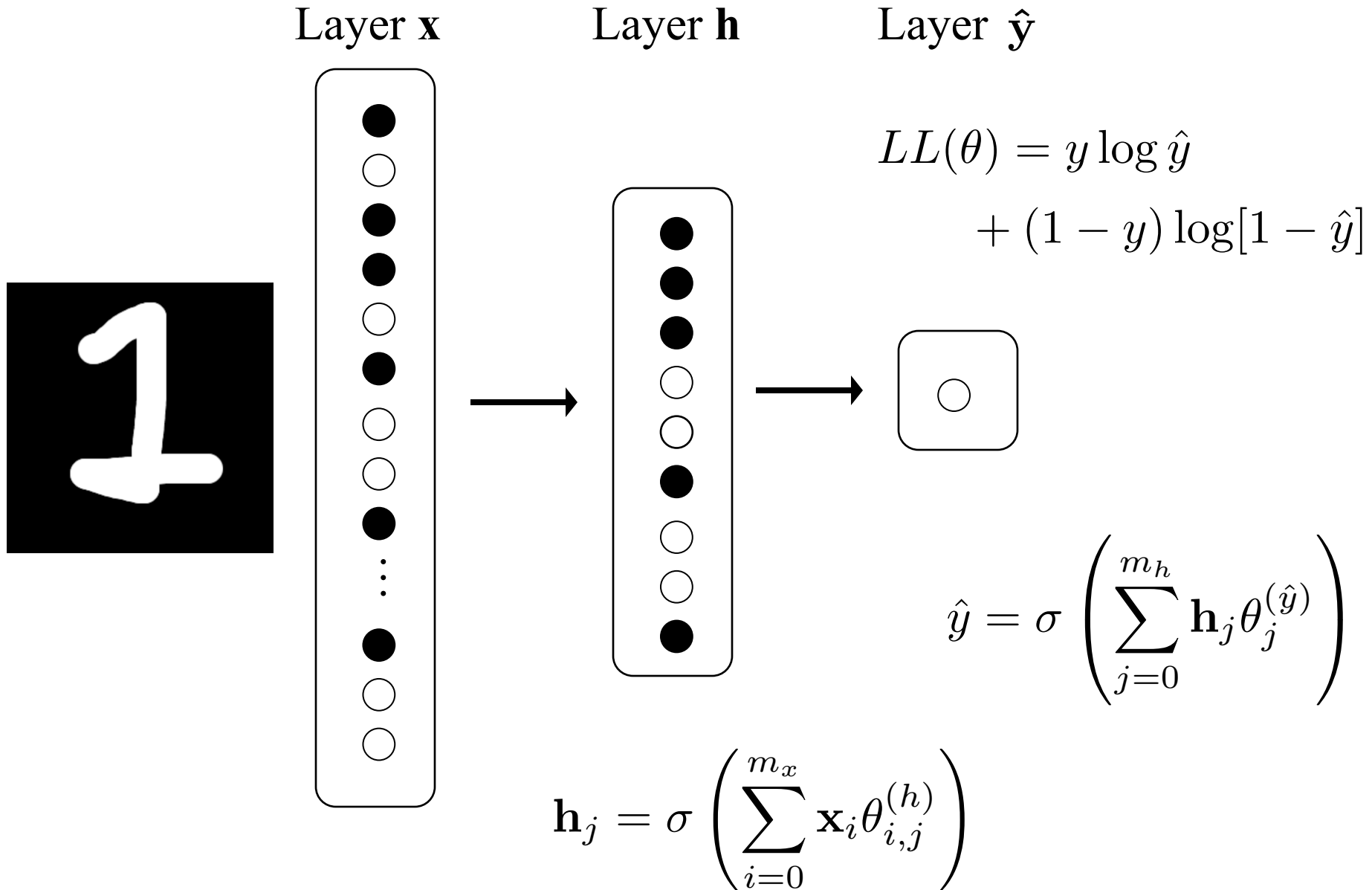


Forward Pass

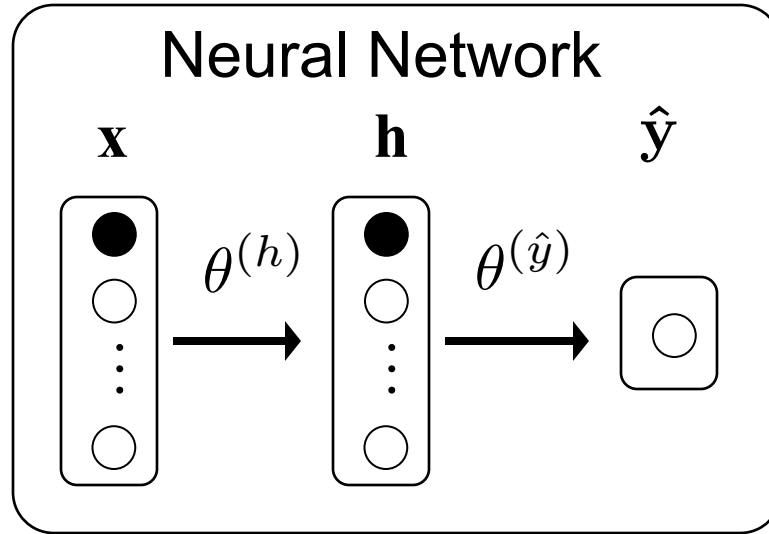


$$\mathbf{h}_j = \sigma \left(\sum_{i=0}^{m_x} \mathbf{x}_i \theta_{i,j}^{(h)} \right)$$

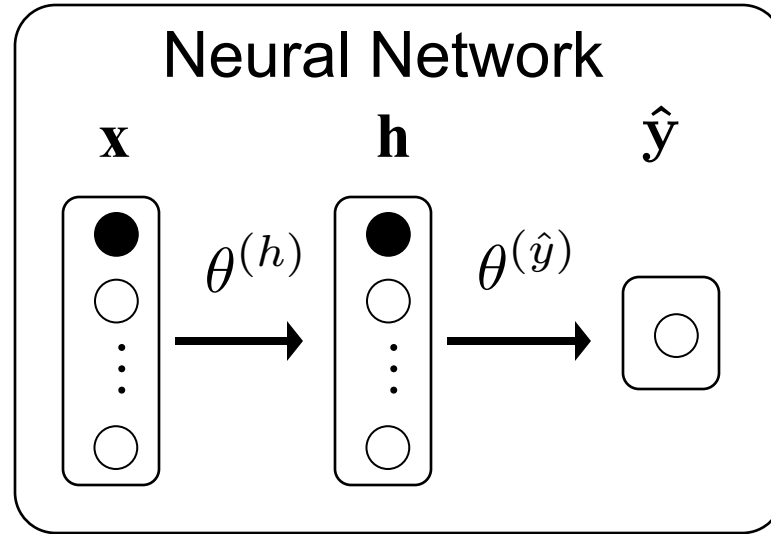
Forward Pass



All Together



Sanity Check 1



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

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

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

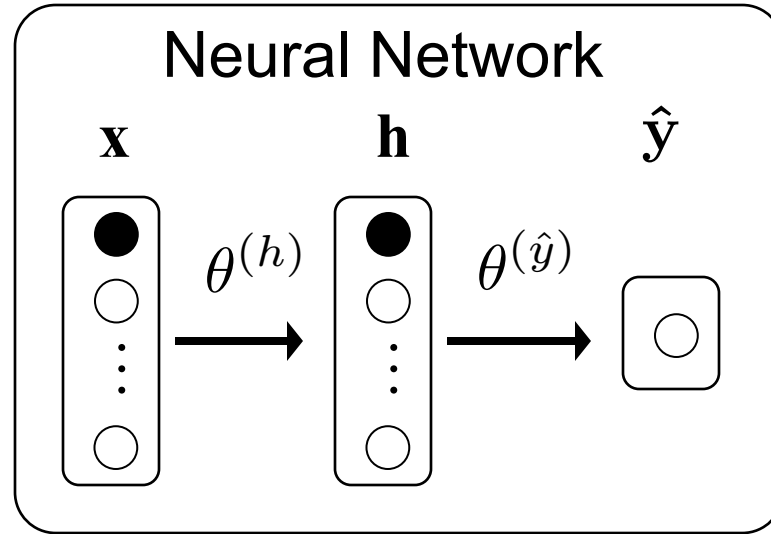
a) 2

b) 20

c) 40

d) 800

Sanity Check 2



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

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

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

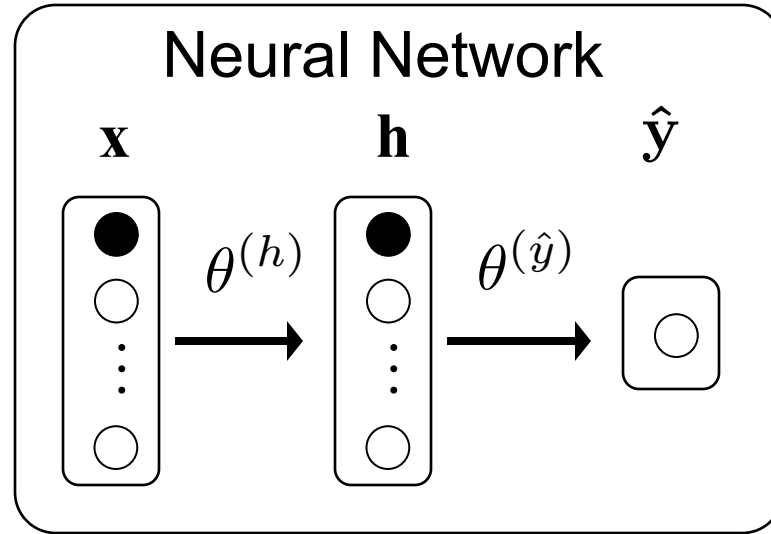
a) 2

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

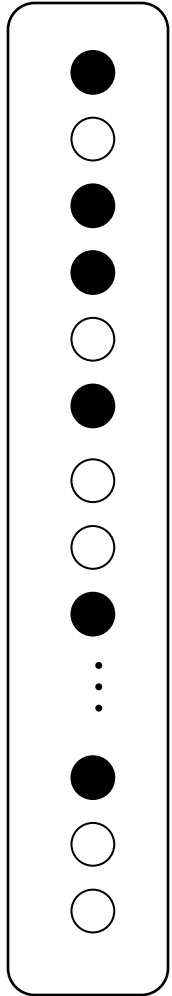
d) 16000

Today: Do Something Brave

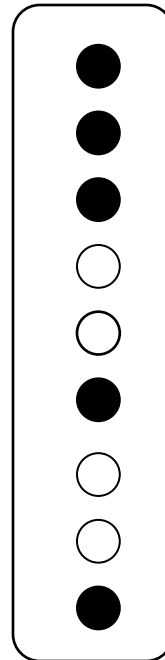


Forward Pass

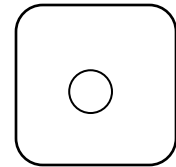
Layer x



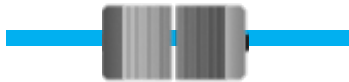
Layer h



Layer $\hat{y}$



800 parameters
need setting



20 parameters
need setting



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.

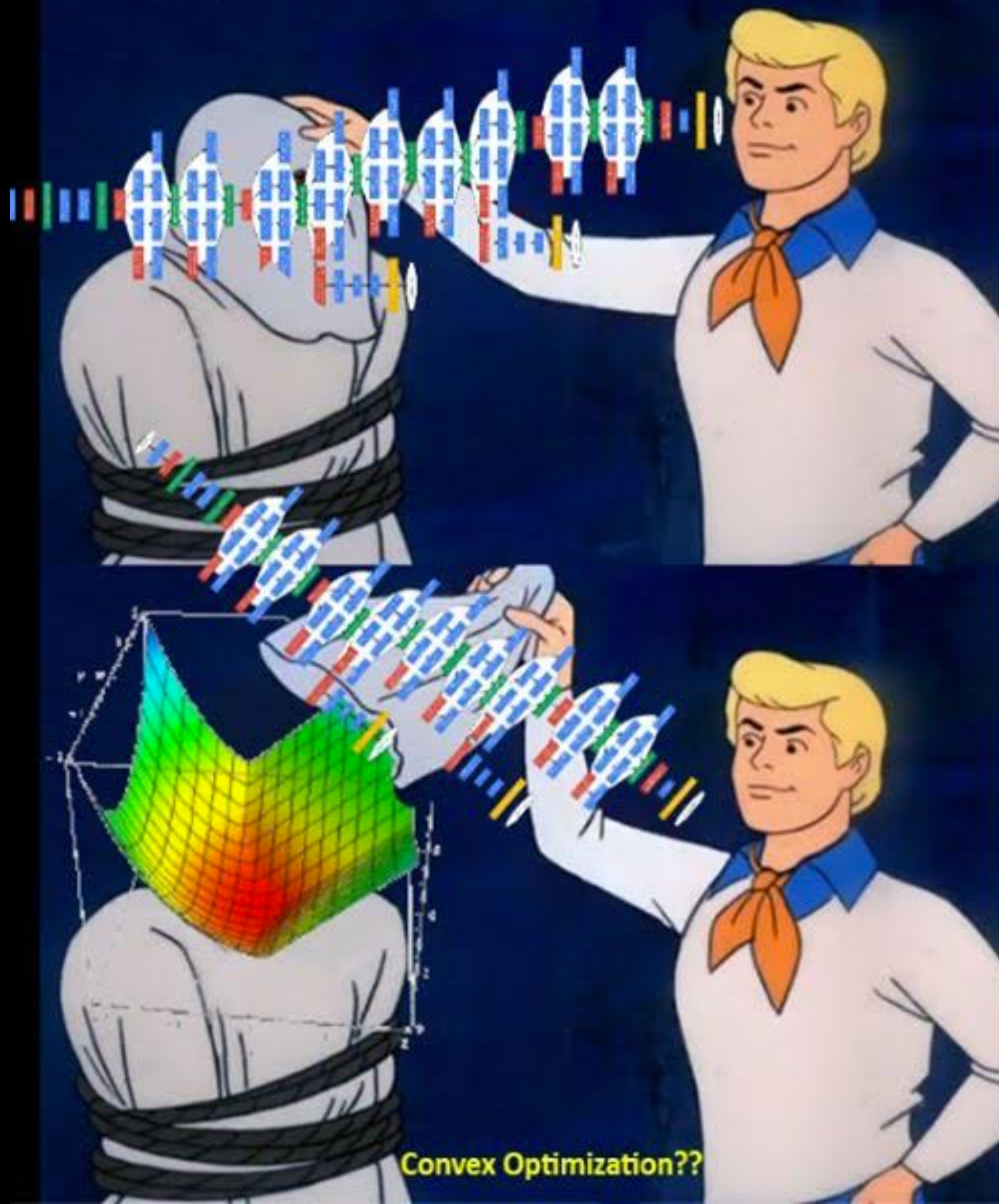
Basically MLE is hard because
it has so many details





Okay gang, let's see what deep learning really is.

Thanks to Keith Eicher



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

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

Feel the Bern!
 $Y \sim \text{Bern}(\hat{y})$

For IID data

$$\begin{aligned} L(\theta) &= \prod_{i=1}^n P(Y = y^{(i)}|X = \mathbf{x}^{(i)}) \\ &= \prod_{i=1}^n (\hat{y}^{(i)})^{y^{(i)}} \cdot \left[1 - (\hat{y}^{(i)})\right]^{(1-y^{(i)})} \end{aligned}$$

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

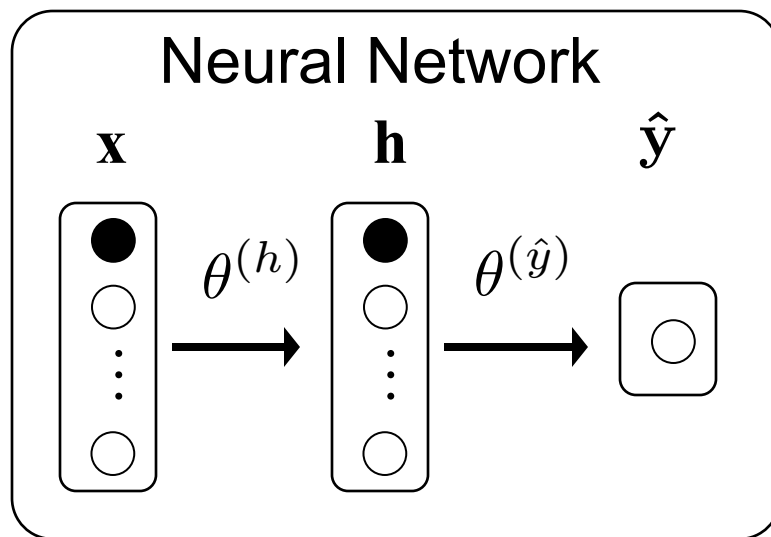
Derivative Goals

Loss with respect to
output layer params

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

Loss with respect to
hidden layer params

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

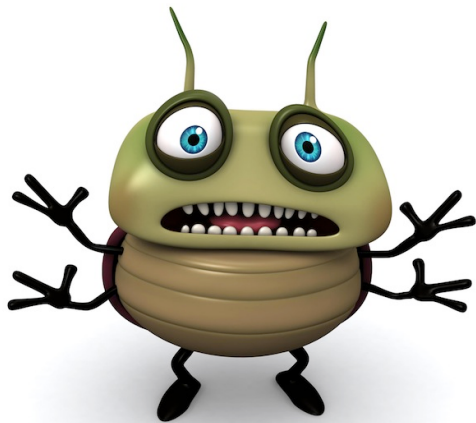


Bad Approach

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

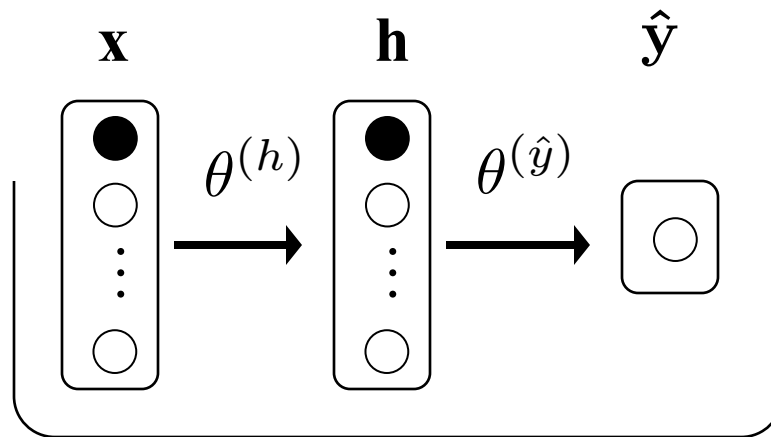
$$\hat{y} = \sigma \left(\sum_{i=0}^{m_h} \mathbf{h}_i \theta_i^{(\hat{y})} \right)$$

Math bug

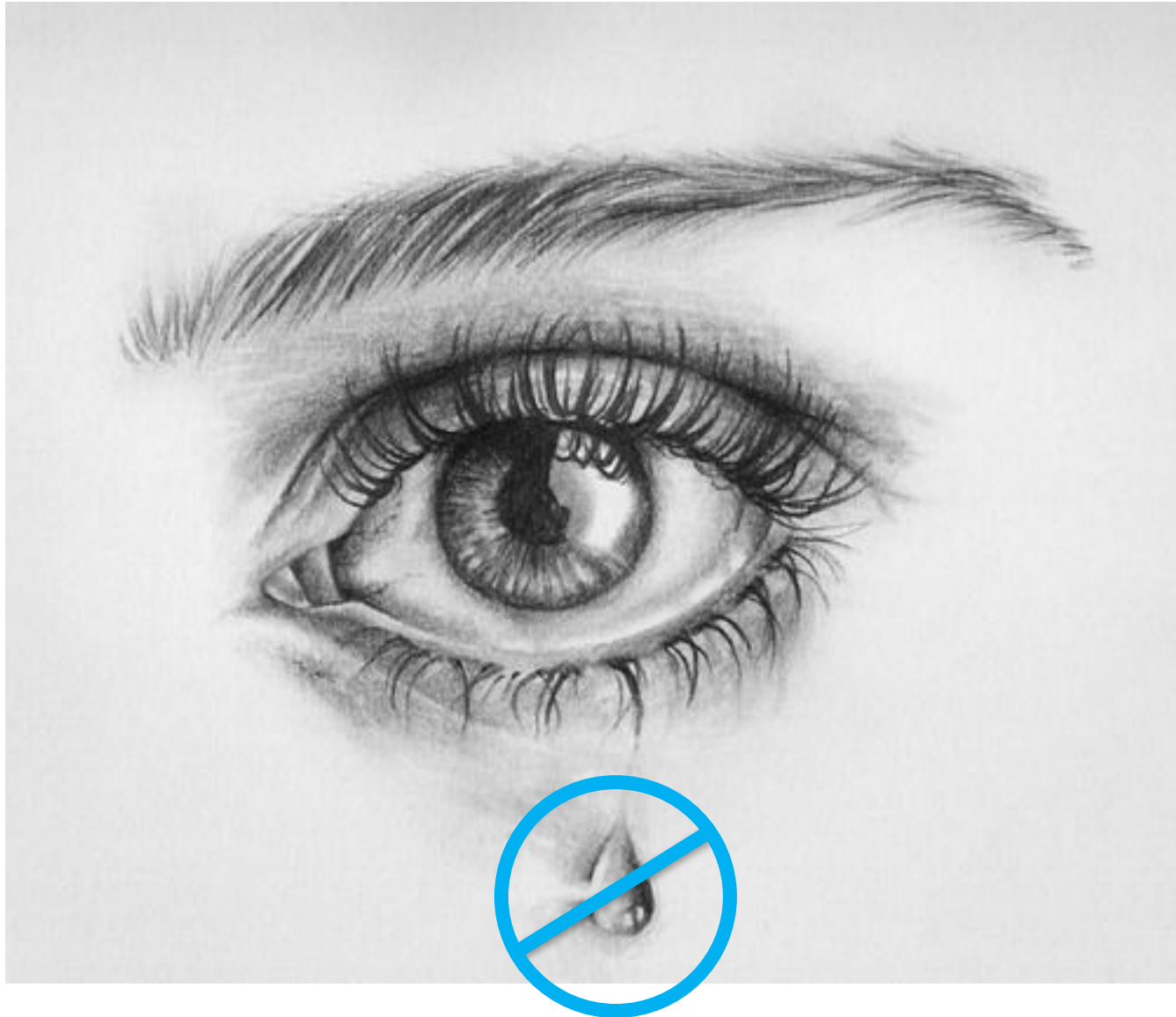


$$= \sigma \left(\sum_{i=0}^{m_h} \left[\sigma \left(\sum_{j=0}^{m_x} \mathbf{x}_j \theta_{i,j}^{(\mathbf{h})} \right) \right] \theta_i^{(\hat{y})} \right)$$

Neural Network



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:

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

Future Chris: Write this on the board ☺ – Thanks, Past Chris



This is ~~Sparta~~!!!!

↑
Stanford

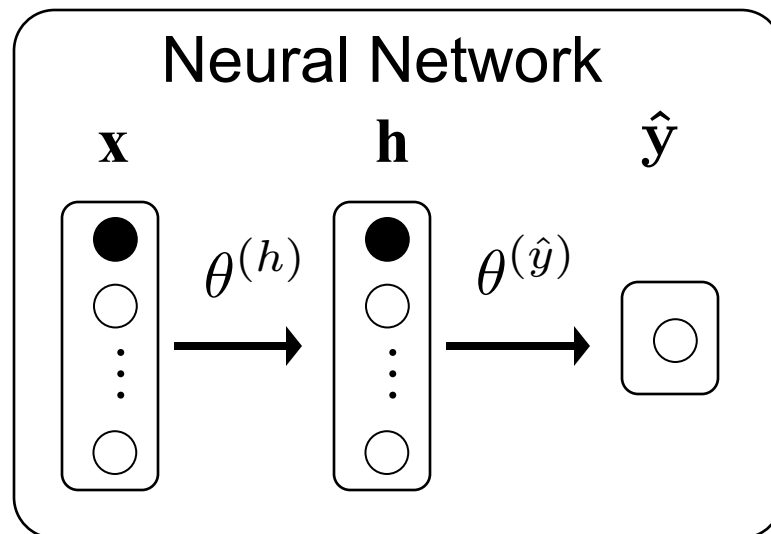
Derivative Goals

Loss with respect to
output layer params

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

Loss with respect to
hidden layer params

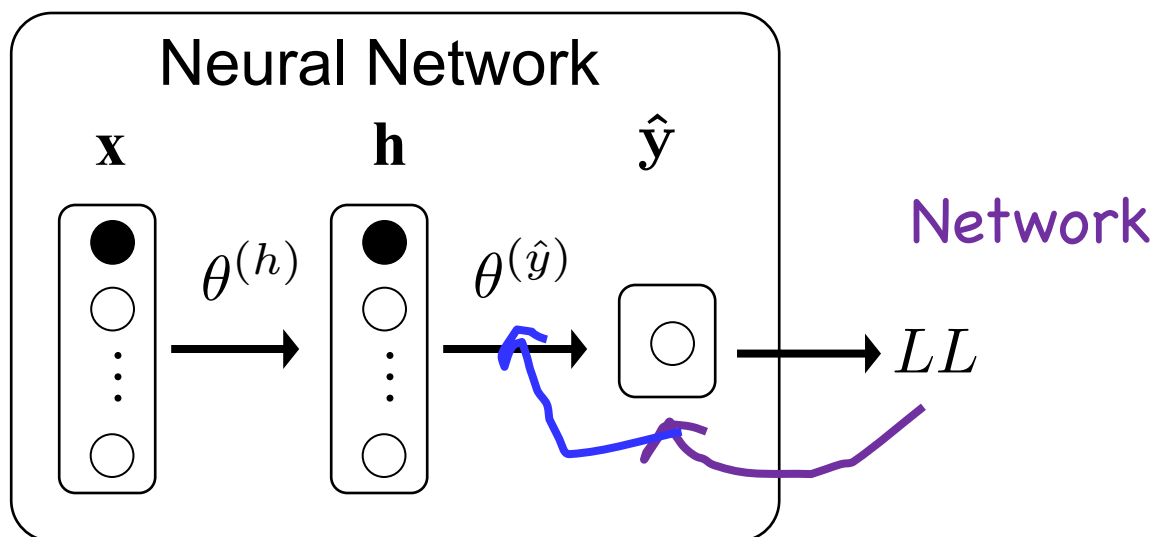
$$\frac{\partial L L(\theta)}{\partial \theta_{i,j}^{(h)}}$$



Chain Rule Example 1

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

Goal



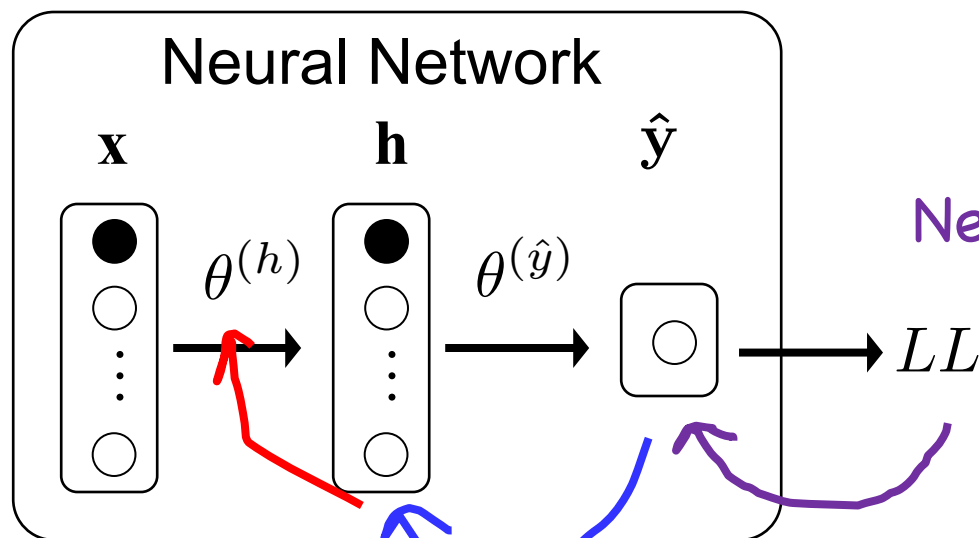
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}}$$

Decomposition

Chain Rule Example 2

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

Goal



Network

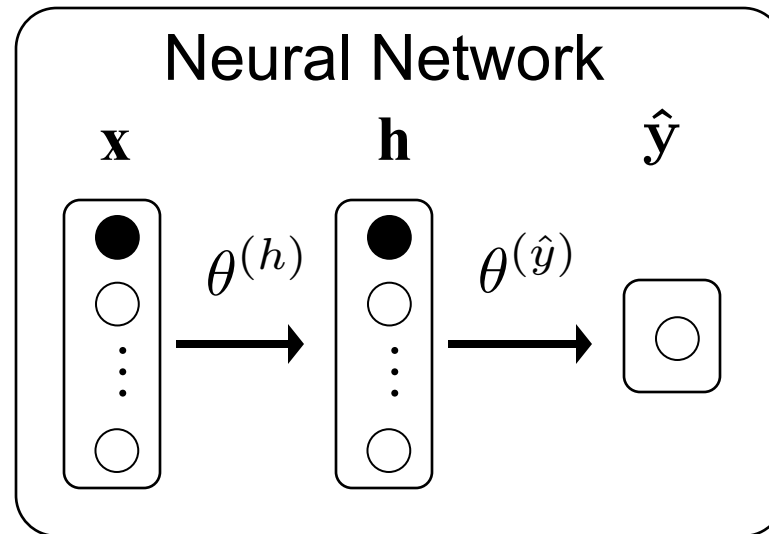
$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{h}_j} \cdot \frac{\partial \mathbf{h}_j}{\partial \theta_{i,j}^{(h)}}$$

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

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

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

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

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

Gradient of output layer params

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

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

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

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

What! That's not scary!

Make it Simple

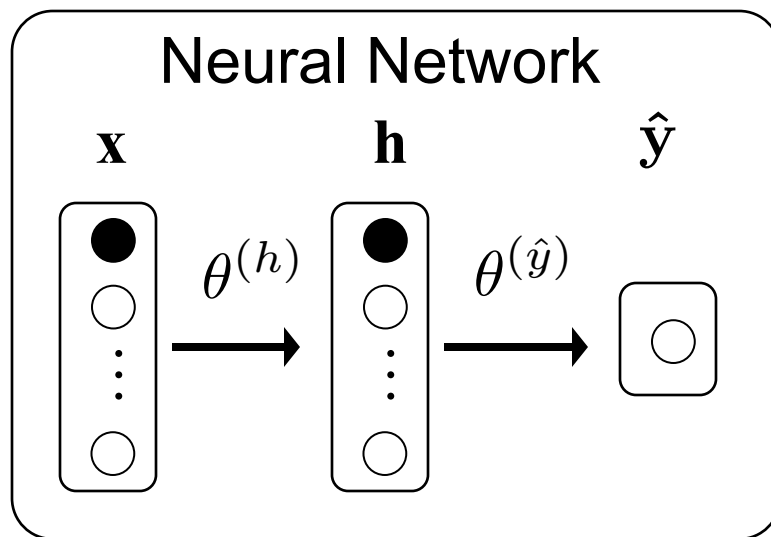
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \text{[Goomba icon]} \cdot \text{[Koopa icon]}$$

$$\text{[Goomba icon]} = \frac{y}{\hat{y}} - \frac{(1 - y)}{(1 - \hat{y})}$$

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

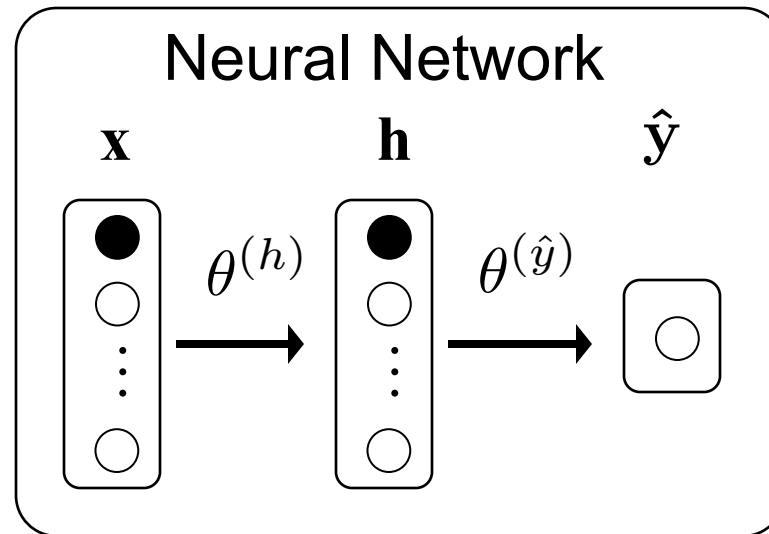
Boom!

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



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

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

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

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

Wait is it over?

Gradient of hidden layer params

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

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

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

That one too?

Make it Simple

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



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


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


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



Congrats. You now know
Backpropagation

Moment of silence

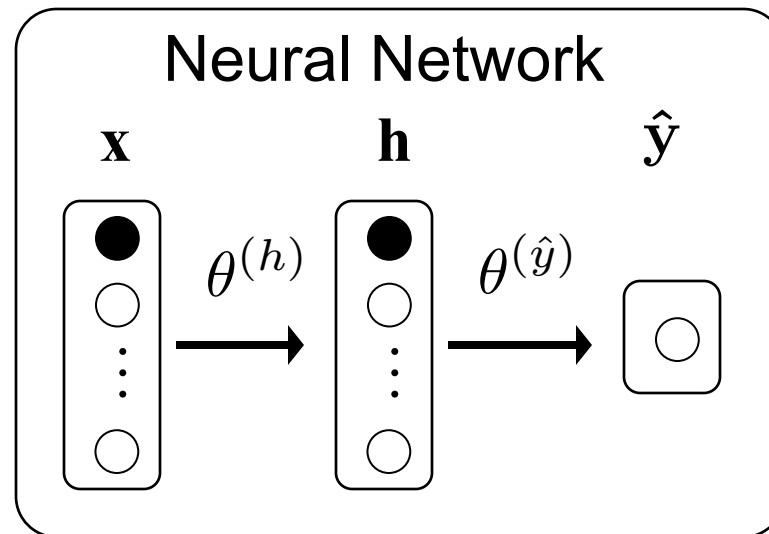
Summary: Simple Calculations For

Loss with respect to
output layer params

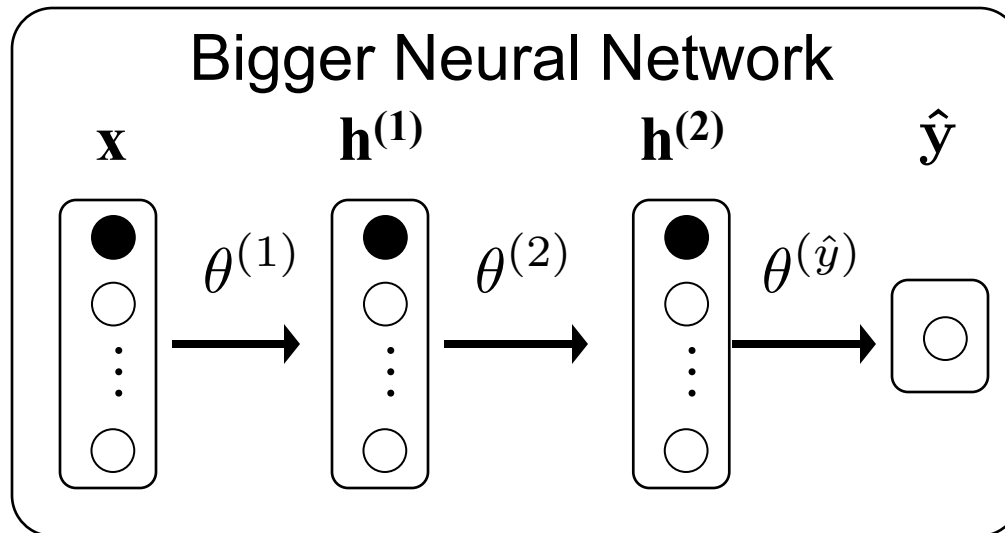
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}}$$

Loss with respect to
hidden layer params

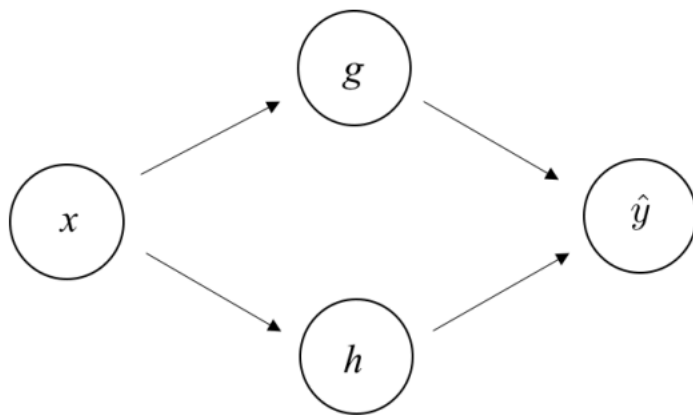
$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}}$$



What Would You Do Here?



What If You Had a Neural Network Like This?



$$g = \text{sigmoid}(\theta_1 \cdot x)$$

$$h = \text{sigmoid}(\theta_2 \cdot x)$$

$$\hat{y} = \text{sigmoid}(\theta_3 \cdot g + \theta_4 \cdot h)$$

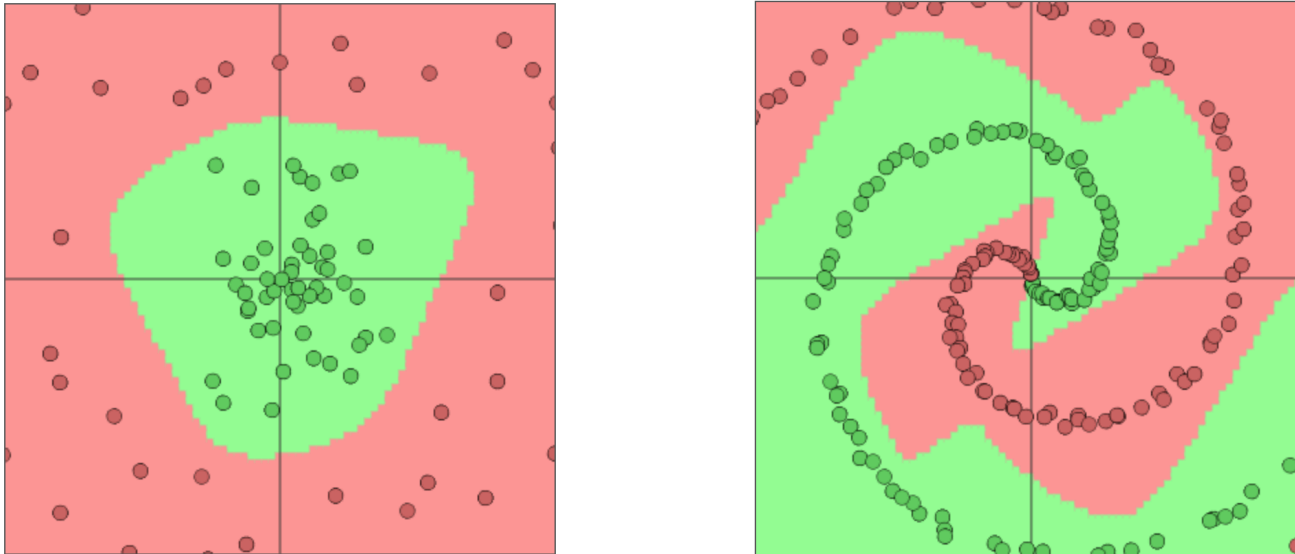
$$LL(\theta) = \sum_{i=1}^n y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

1. Calculate partial derivative for one data instance
2. Use chain rule!
3. Sigmoid derivatives come out simple if you use the right decomp.
4. You don't need to give the most reduced answer

Chain rule:
Game changer for
artificial intelligence

Neural Networks Can Learn Complex Functions

- Some data sets/functions are not separable

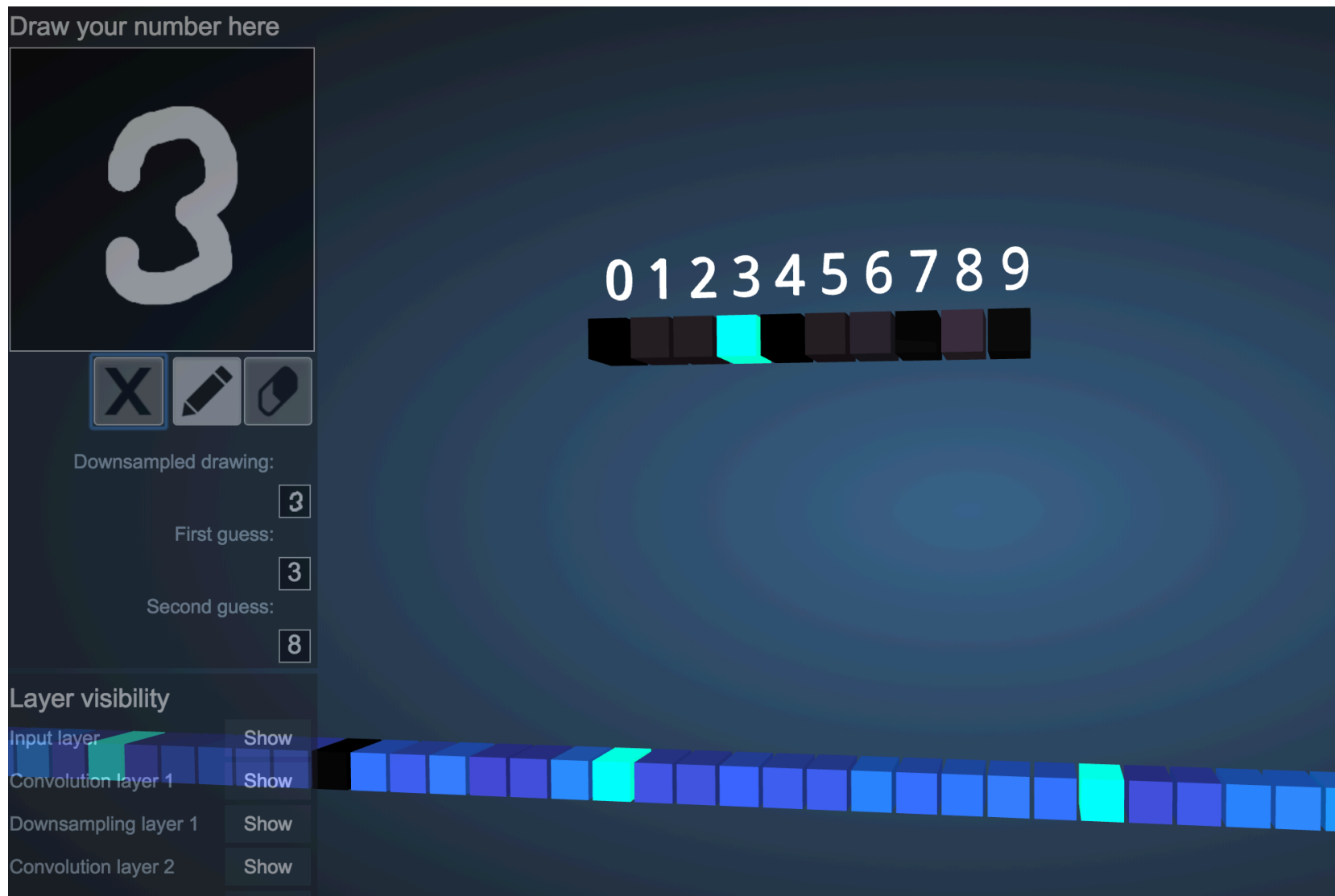


- These are classifiers learned by neural networks

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

Some Extra Ideas!

Multiple Outputs

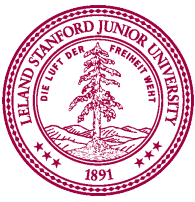


Multiple Output Classification?



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 $\text{softmax}(\mathbf{z})$ of real values in the range $[0, 1]$ that add up to 1.

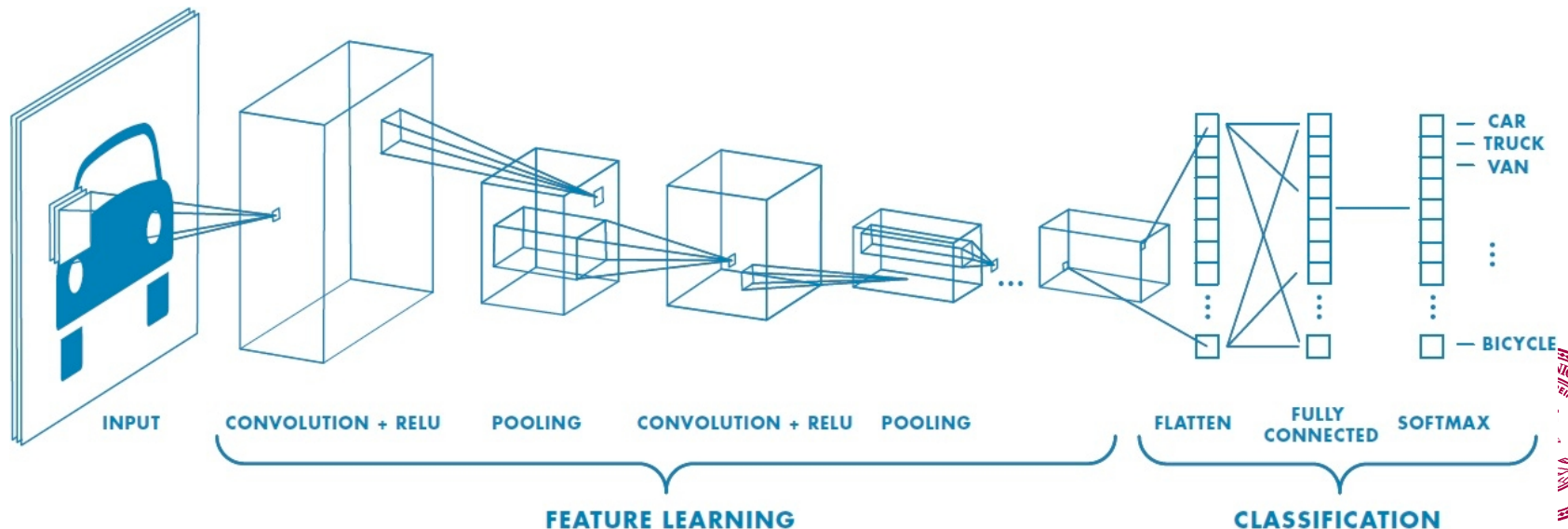
$$P(Y = j | \mathbf{X} = \mathbf{x}) = \text{softmax}(f(\mathbf{x}))_j$$



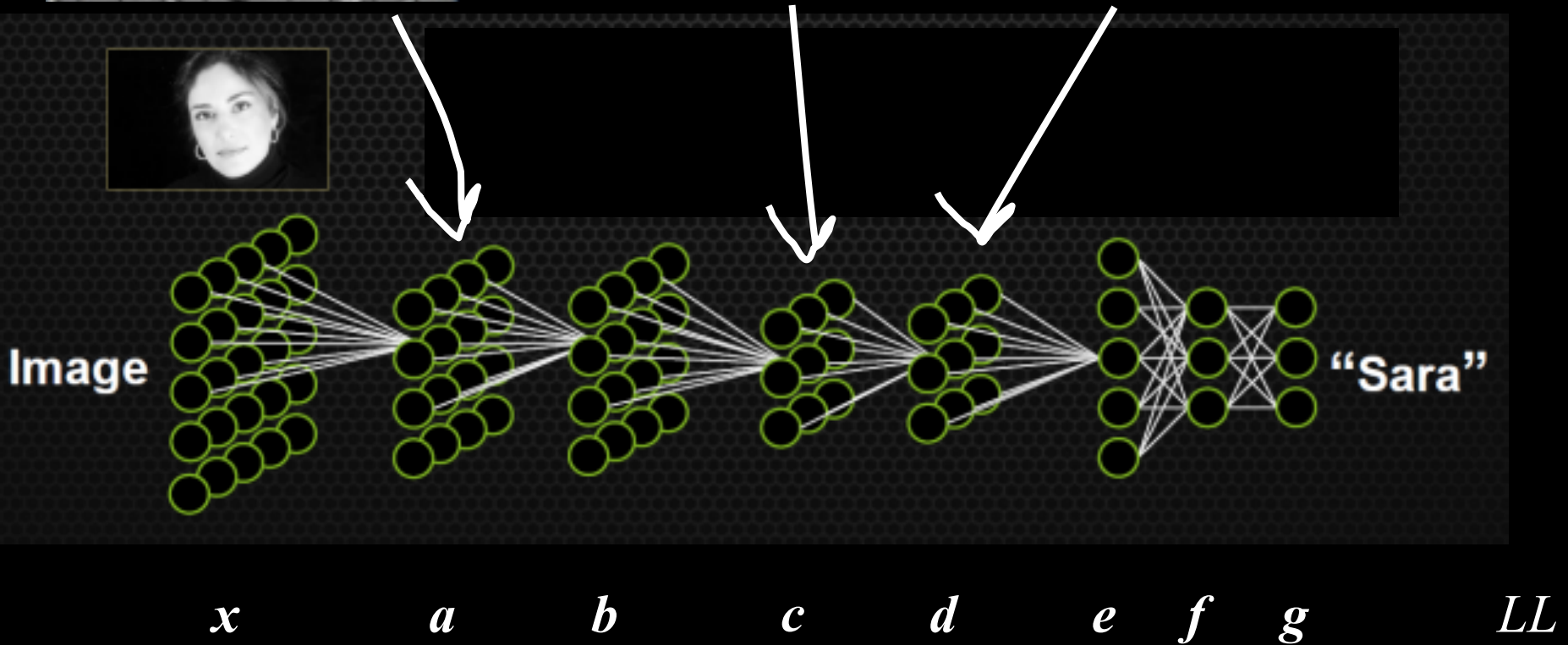
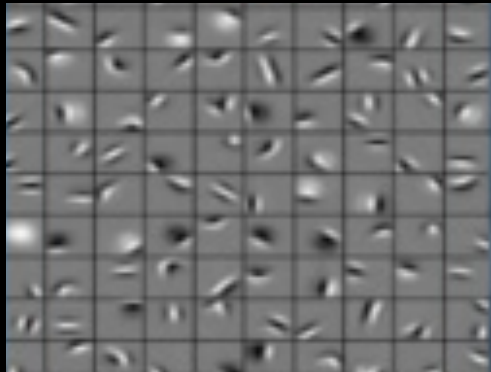
Shared Weights?



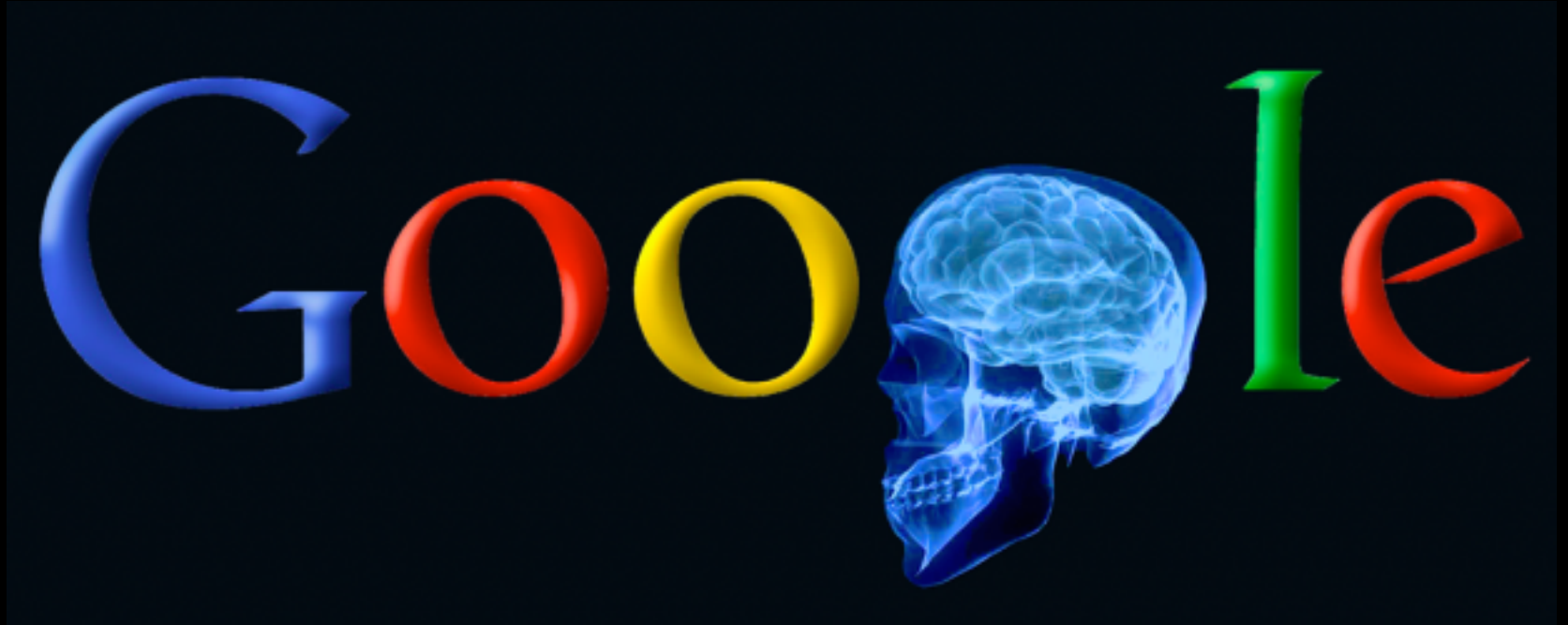
Convolution it turns out if you want to force some of your weights to be shared for different neurons, the math isn't that much harder. This is used a lot for vision (CNN).



Works for any number of layers



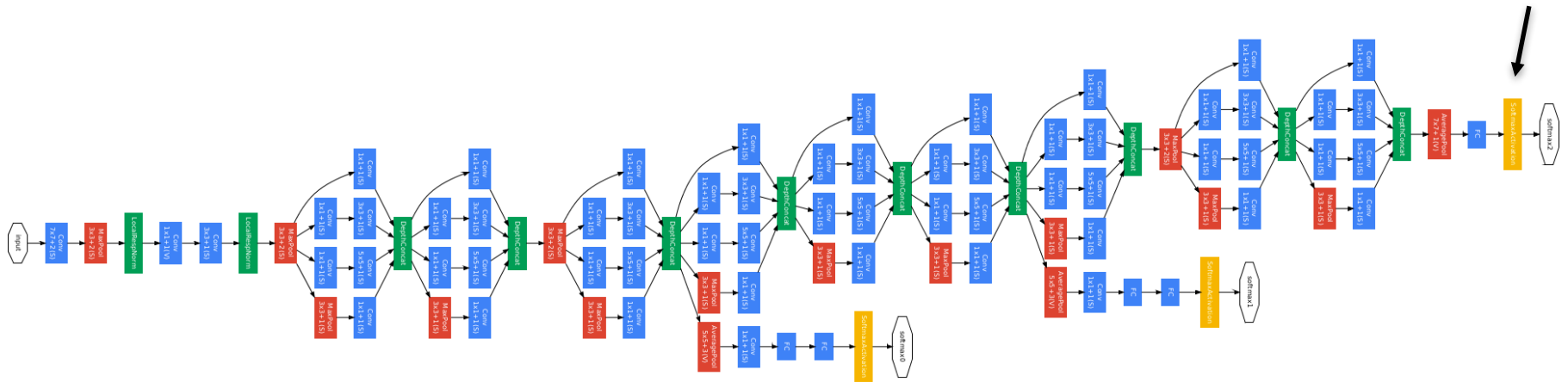
GoogLeNet Brain



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

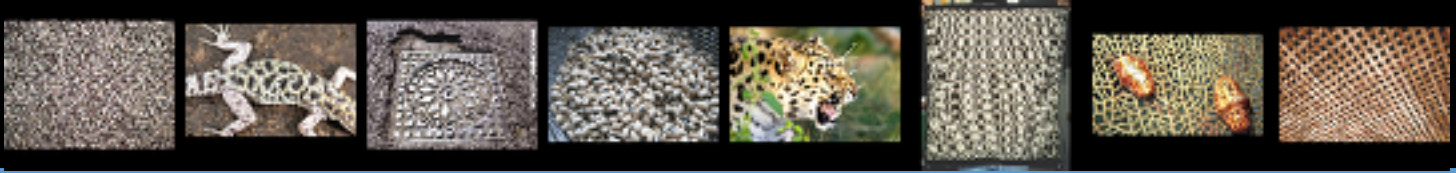
Hire the smartest people in the world



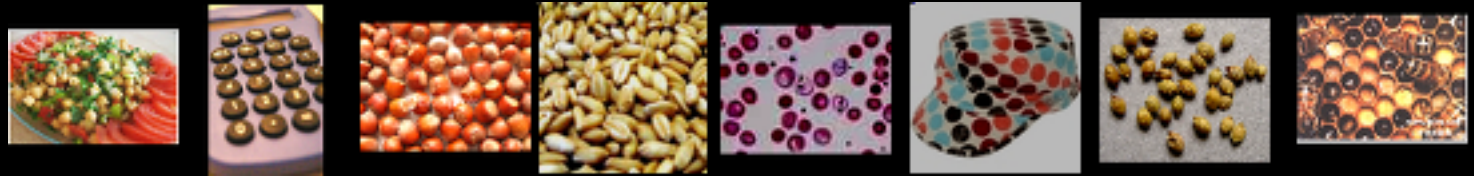
Invent cat detector

Best Neuron Stimuli

Neuron 1



Neuron 2



Neuron 3



Neuron 4



Neuron 5



Best Neuron Stimuli

Neuron 6



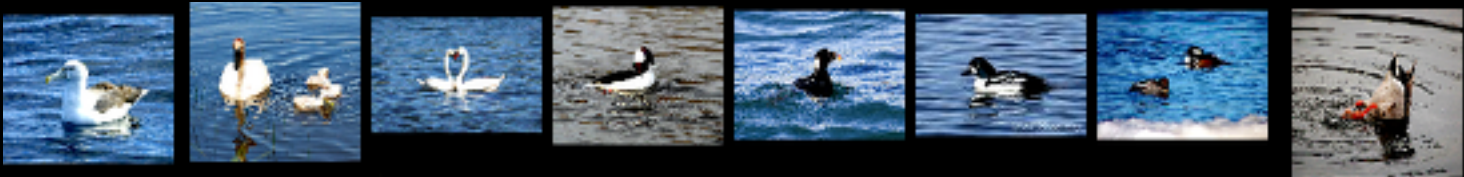
Neuron 7



Neuron 8



Neuron 9



Best Neuron Stimuli

Neuron 10



Neuron 11



Neuron 12



Neuron 13



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

rougtail 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

0.005%

Random guess

1.5%

Pre Neural Networks

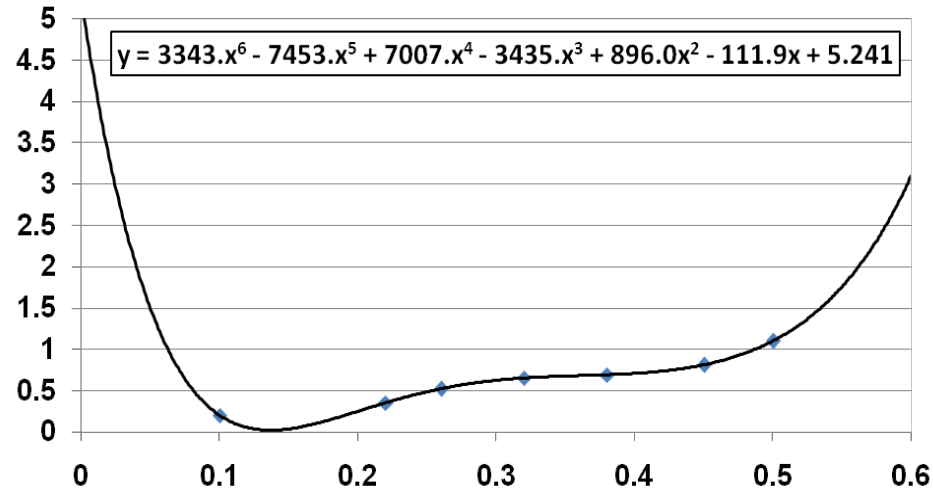
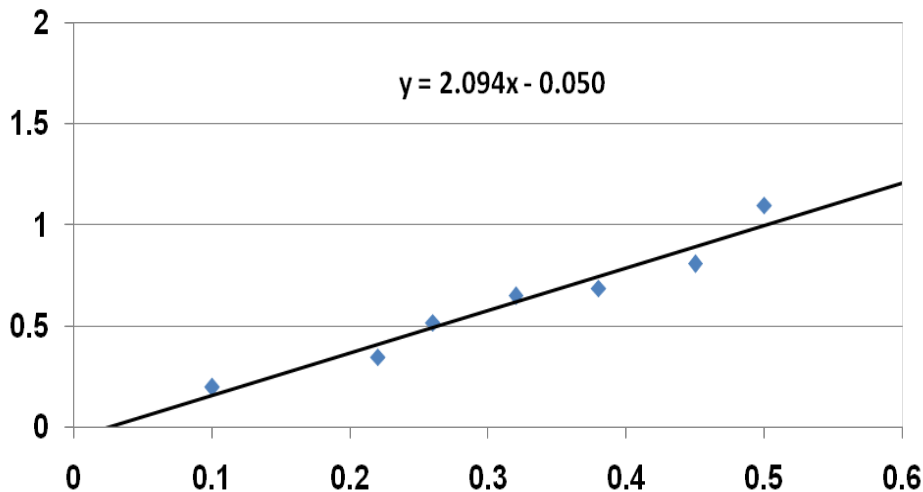
43.9%

GoogLeNet

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

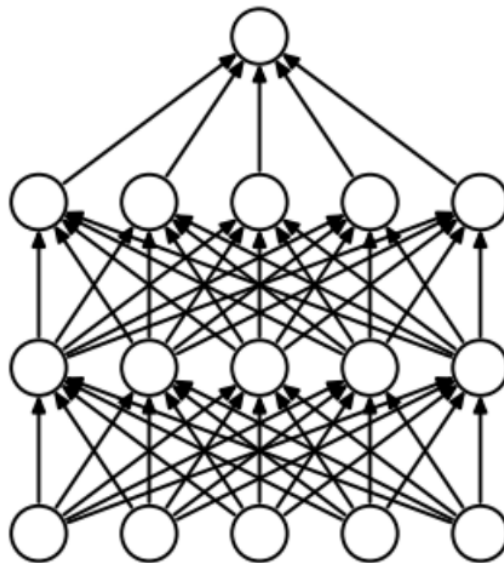


- Polynomial on the right fits training data perfectly!
- Which would you rather use to predict a new data point?

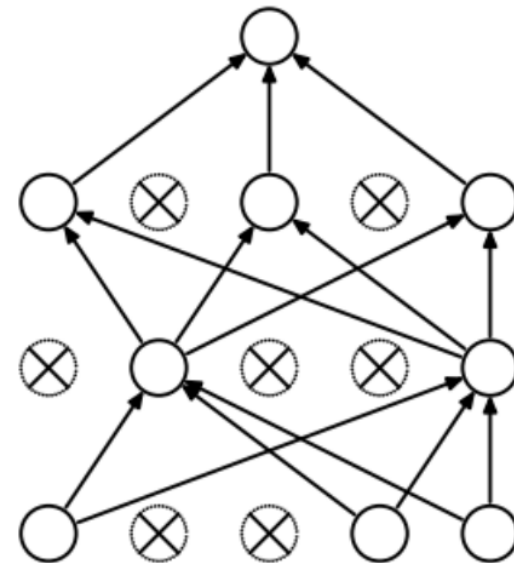
Prevent Overfitting?



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.

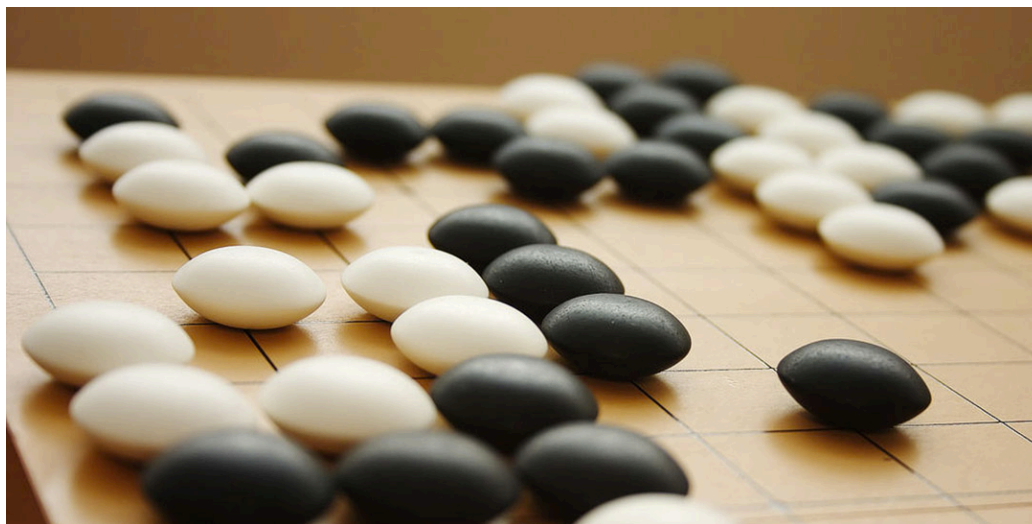
Not everything is classification

Making Decisions?

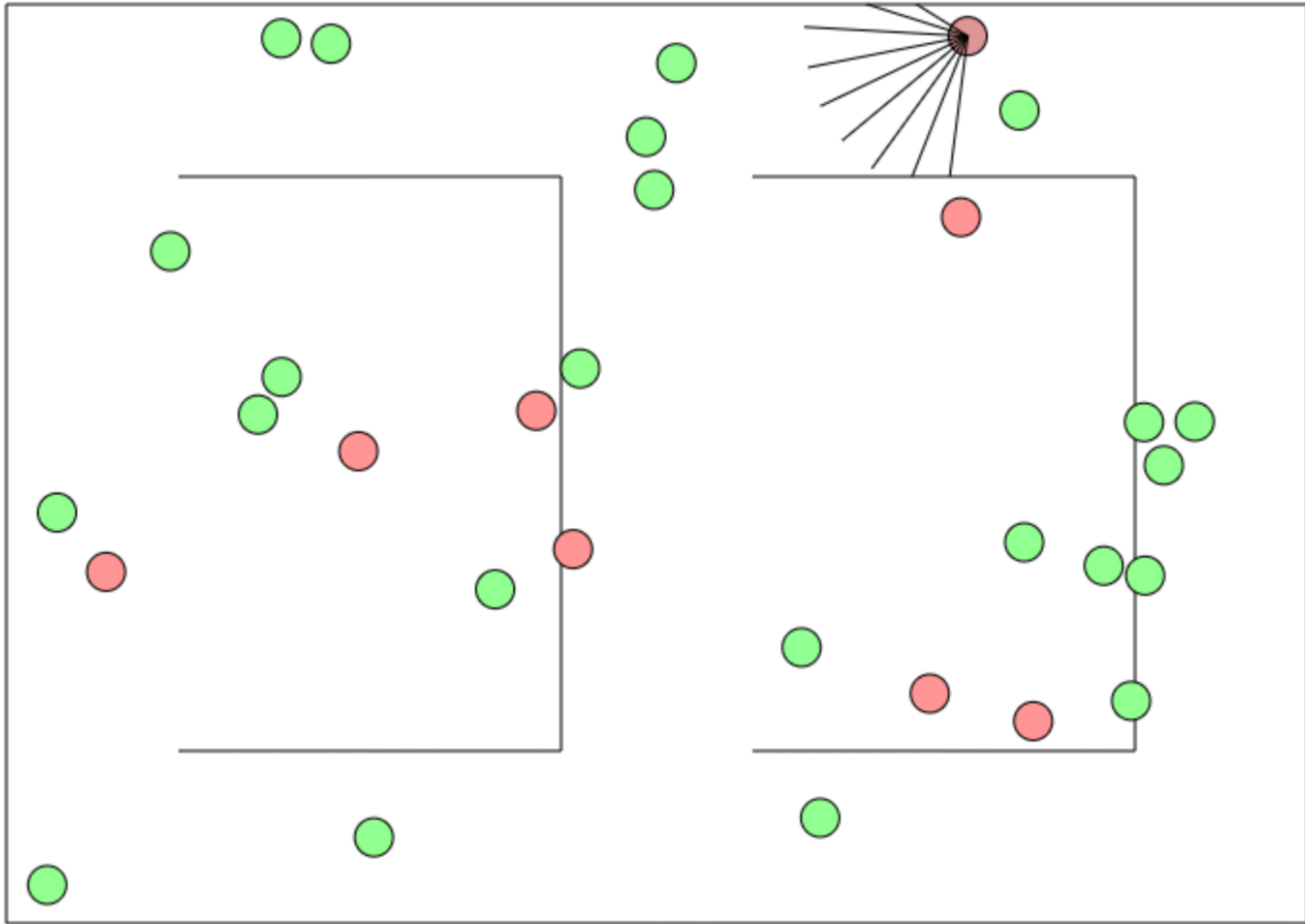


Deep Reinforcement Learning

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



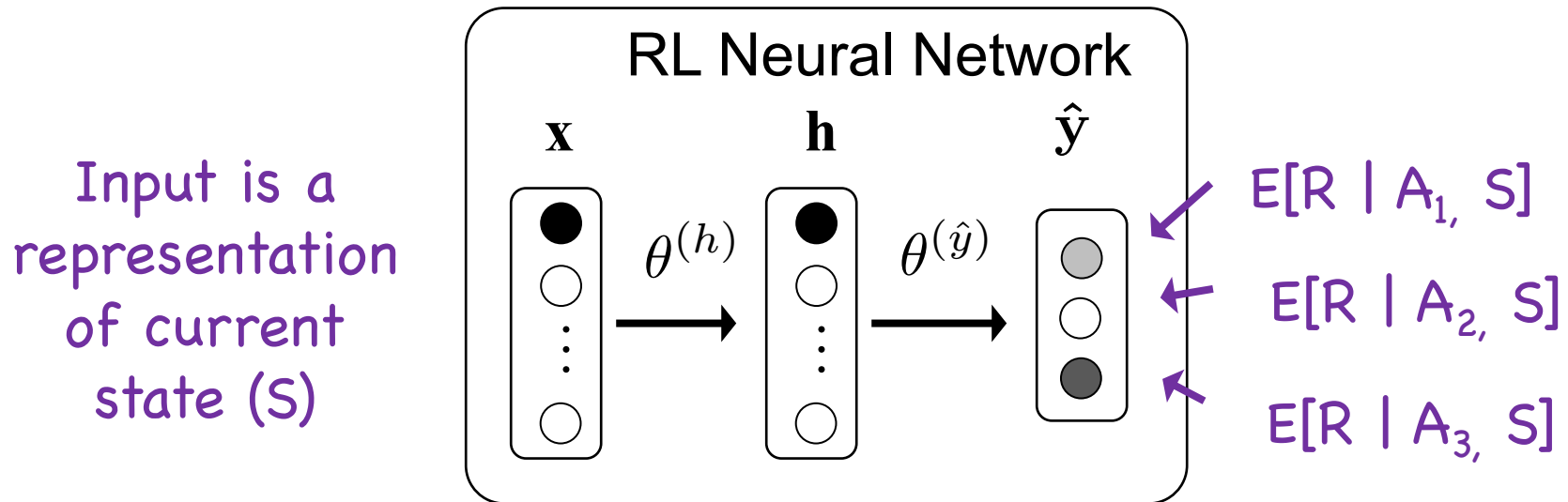
Deep Reinforcement Learning



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

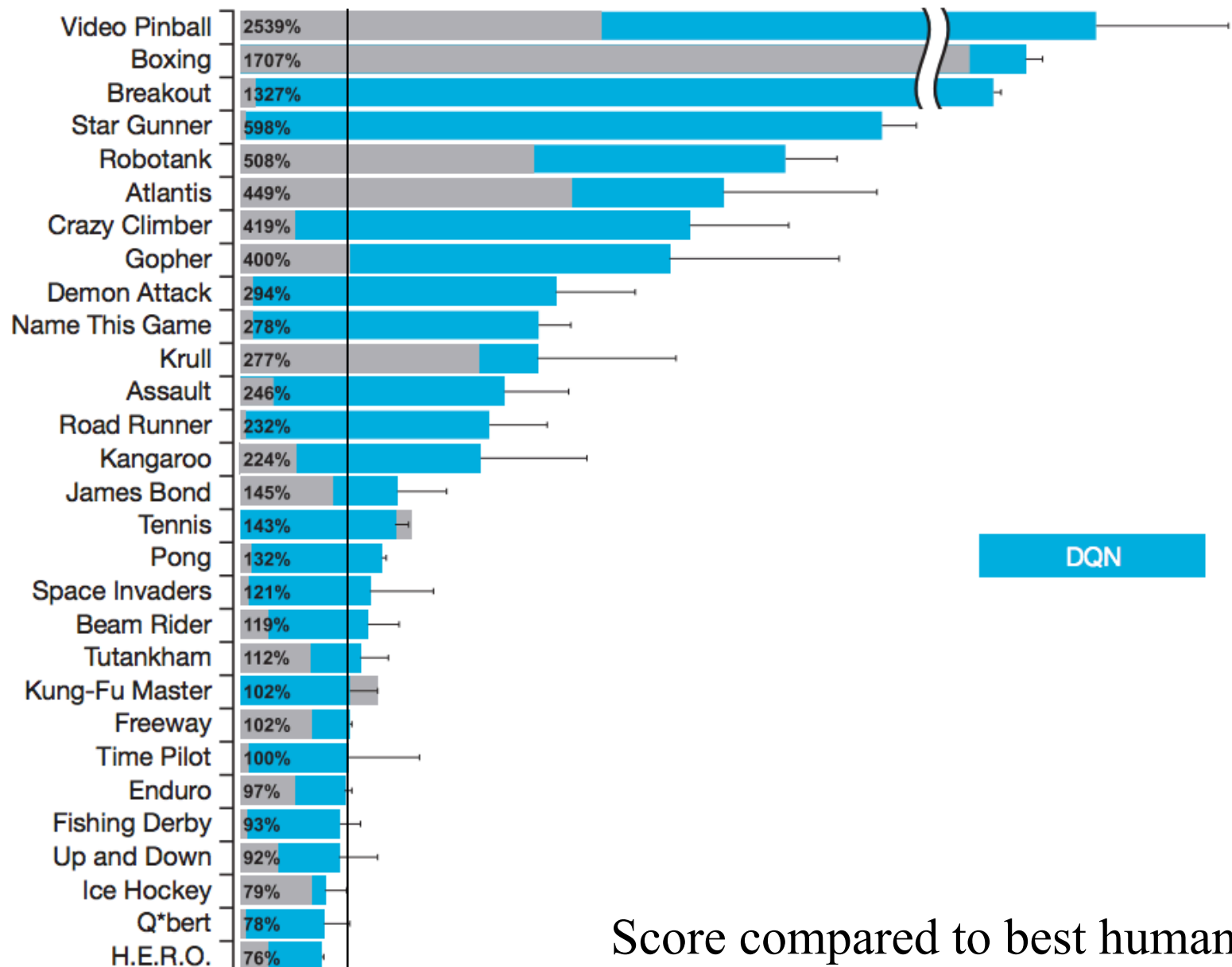
Deep Reinforcement Learning

R is a reward and A_i is a legal action



Interpret outputs as expected reward for a given action

Deep Mind Atari Games





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

Story of Riley



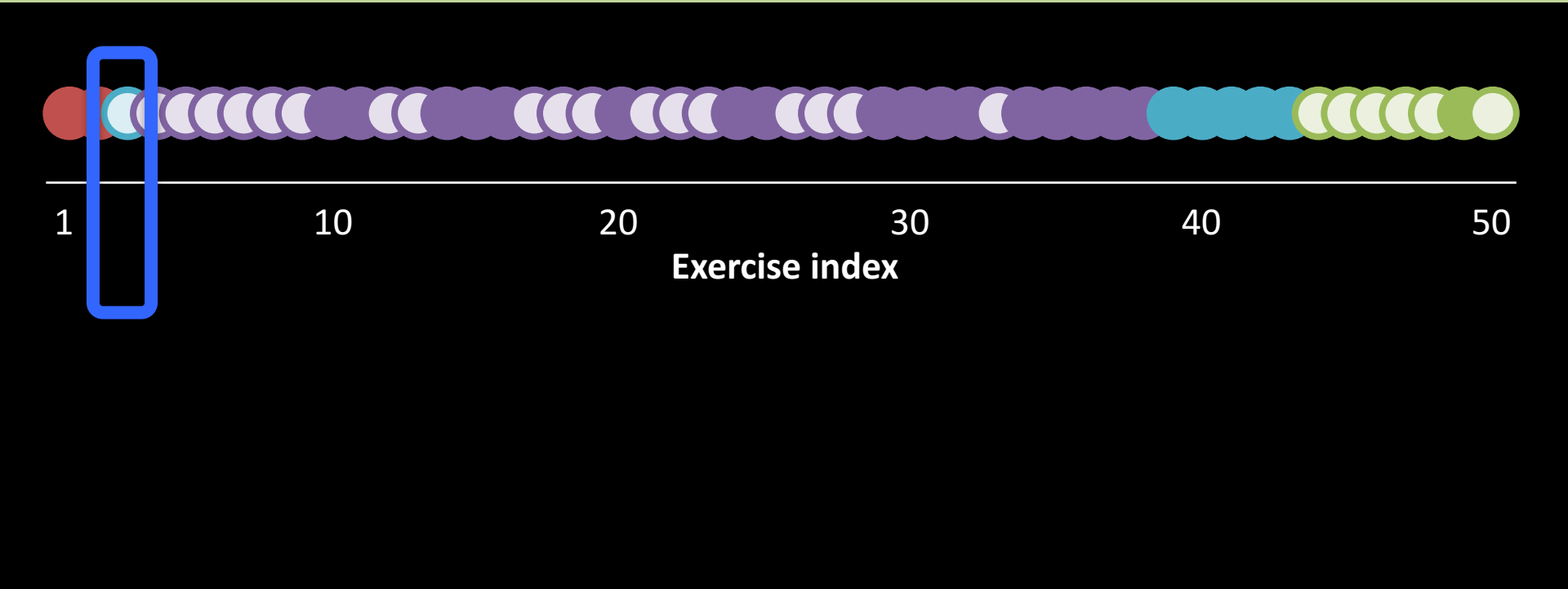
Exercise Type:

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

Answer:

-  Correct
-  Incorrect

Story of Riley



Exercise Type:	Answer:
Solving for x-intercept	Correct
Solving for y-intercept	Incorrect
Graphing linear equations	
Square roots	
Slope of a line	

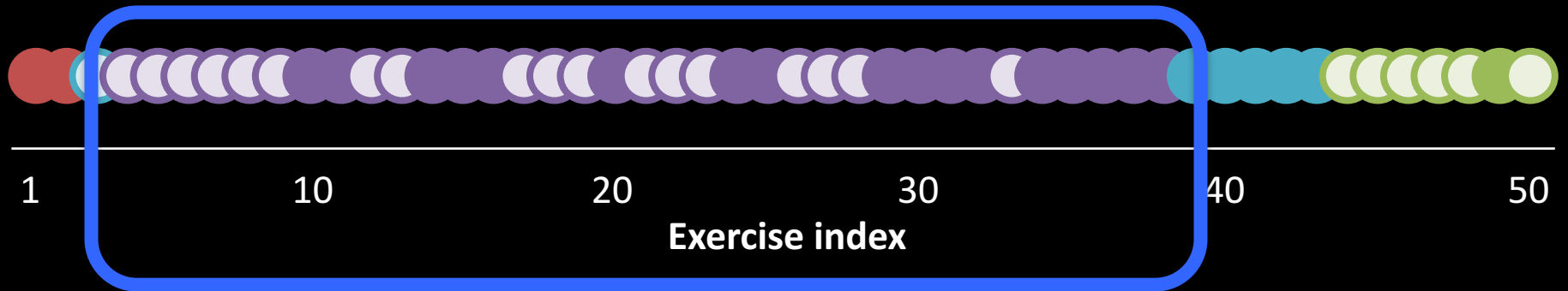
Exercise Type:	Answer:
 Solving for x-intercept	 Correct
 Solving for y-intercept	 Incorrect
 Graphing linear equations	
 Square roots	
 Slope of a line	

Exercise Type:	Answer:
Solving for x-intercept	Correct
Solving for y-intercept	Incorrect
Graphing linear equations	
Square roots	
Slope of a line	

Exercise Type:	Answer:
 Solving for x-intercept	 Correct
 Solving for y-intercept	 Incorrect
 Graphing linear equations	
 Square roots	
 Slope of a line	



Story of Riley



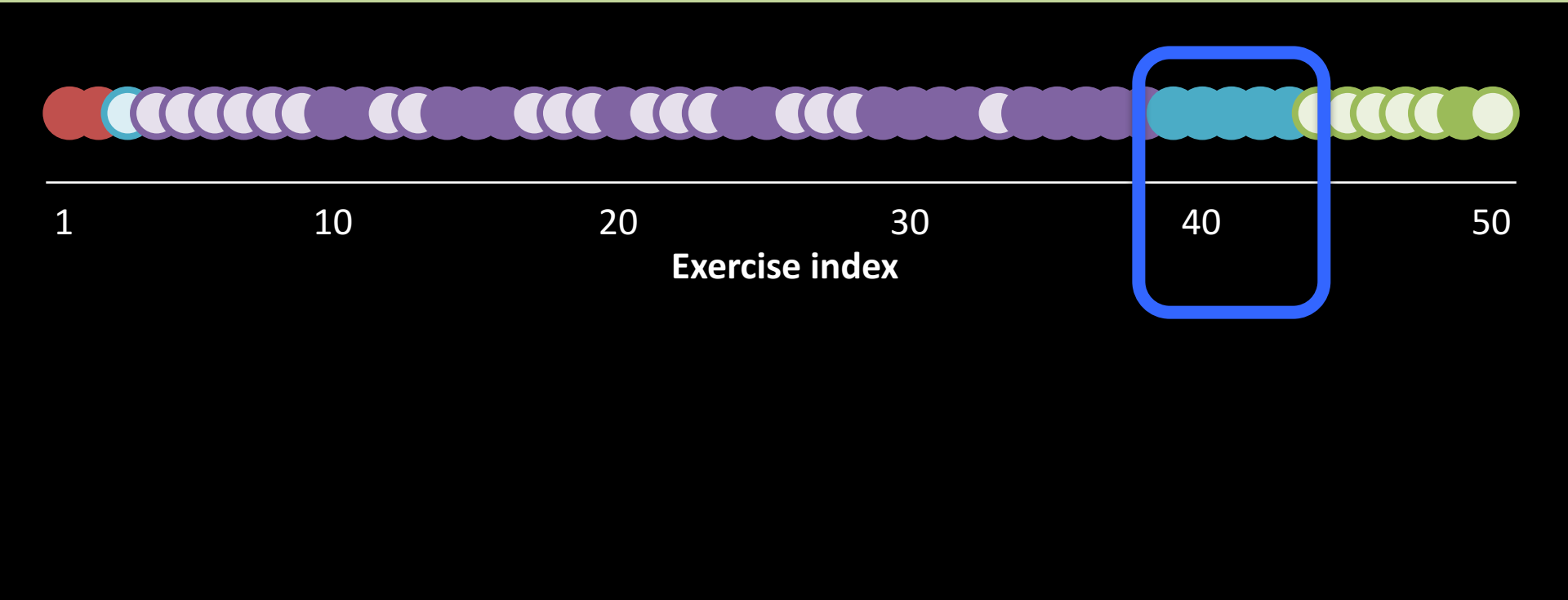
Exercise Type:

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

Answer:

-  Correct
-  Incorrect

Story of Riley



Exercise Type:	Answer:
Solving for x-intercept	Correct
Solving for y-intercept	Incorrect
Graphing linear equations	
Square roots	
Slope of a line	

Exercise Type:	Answer:
 Solving for x-intercept	 Correct
 Solving for y-intercept	 Incorrect
 Graphing linear equations	
 Square roots	
 Slope of a line	

Exercise Type:	Answer:
Solving for x-intercept	Correct
Solving for y-intercept	Incorrect
Graphing linear equations	
Square roots	
Slope of a line	

Exercise Type:	Answer:
 Solving for x-intercept	 Correct
 Solving for y-intercept	 Incorrect
 Graphing linear equations	
 Square roots	
 Slope of a line	



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

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

Story of Riley



What should Riley do next?

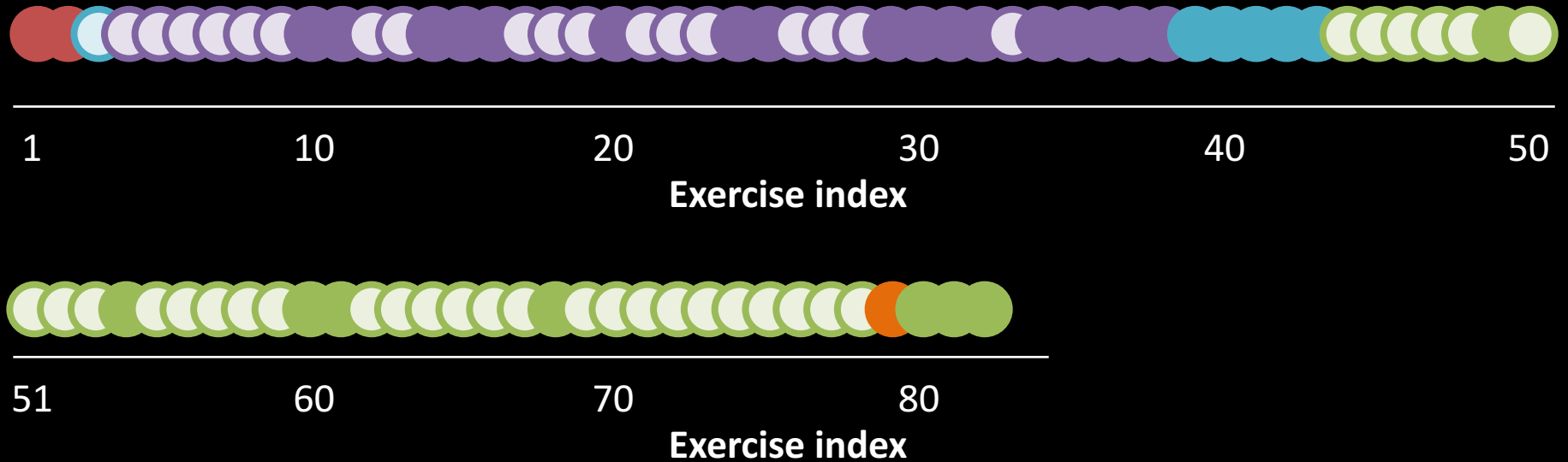
Exercise Type:

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

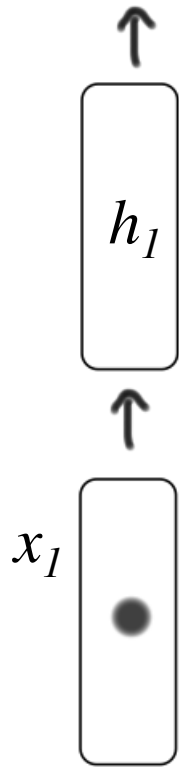
Answer:

-  Correct
-  Incorrect

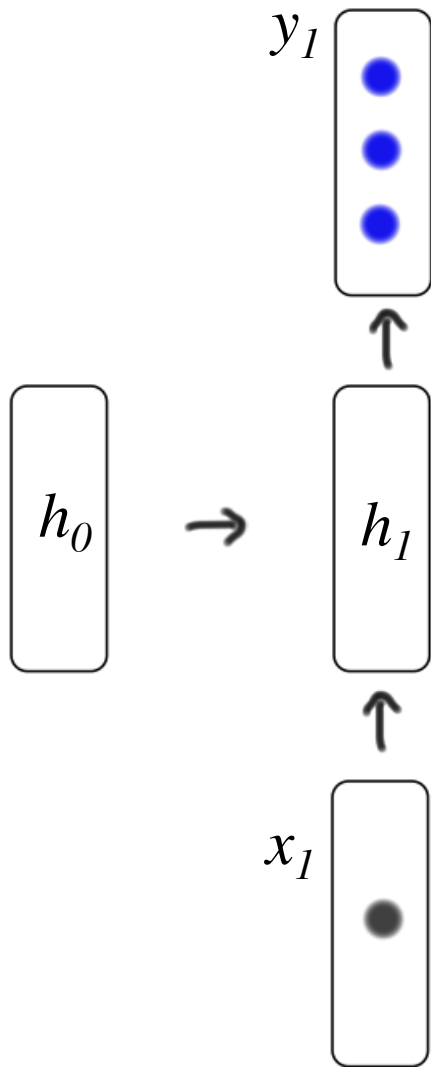
Story of Riley



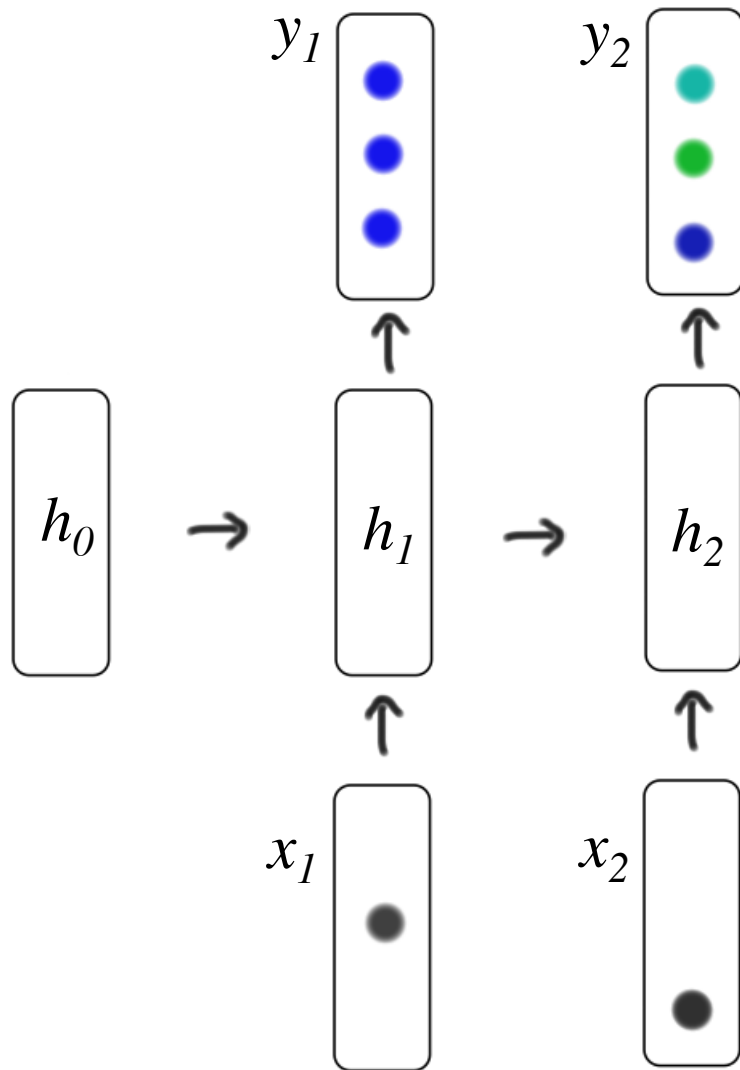
Recurrent Neural Network



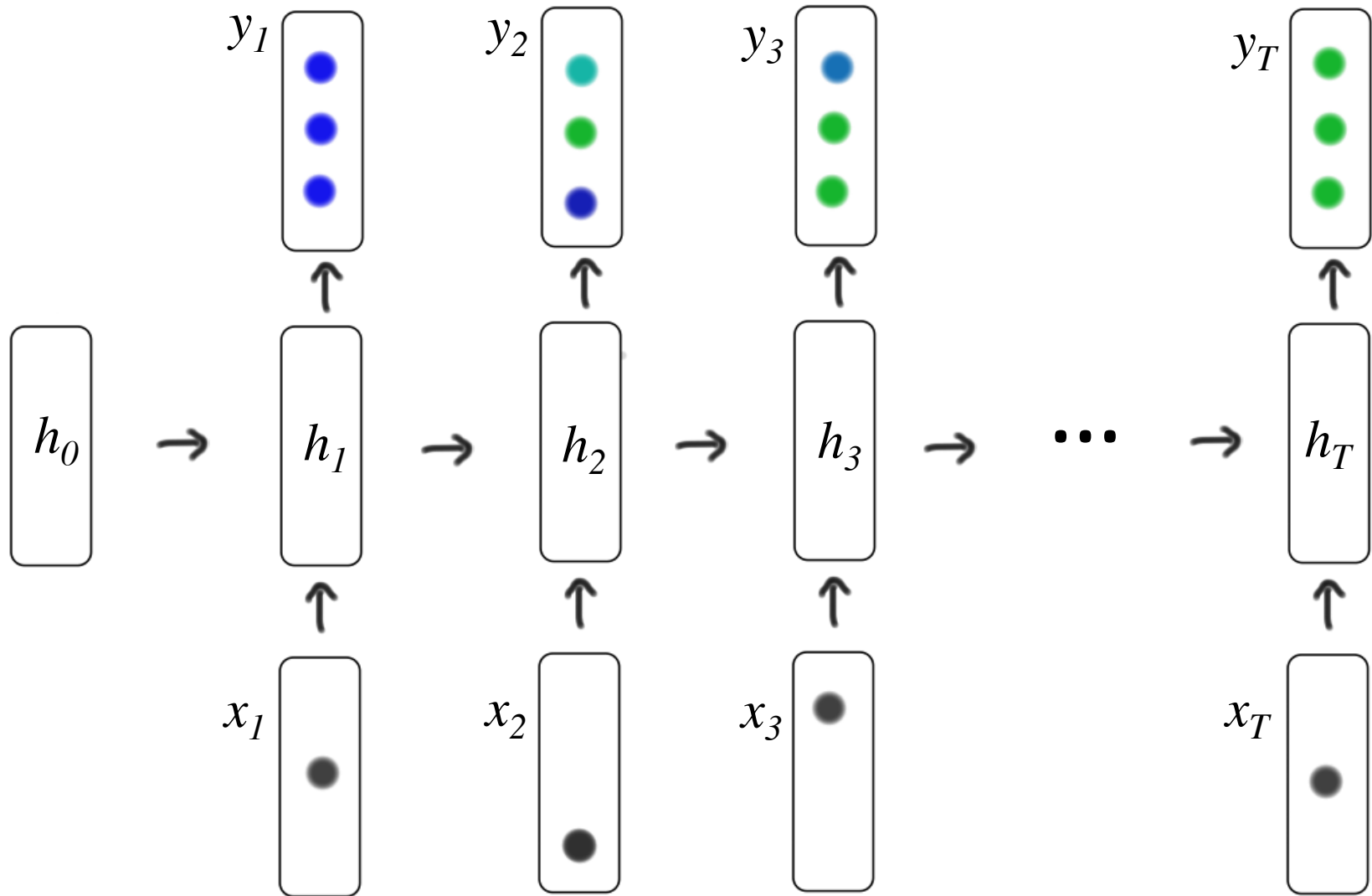
Recurrent Neural Network



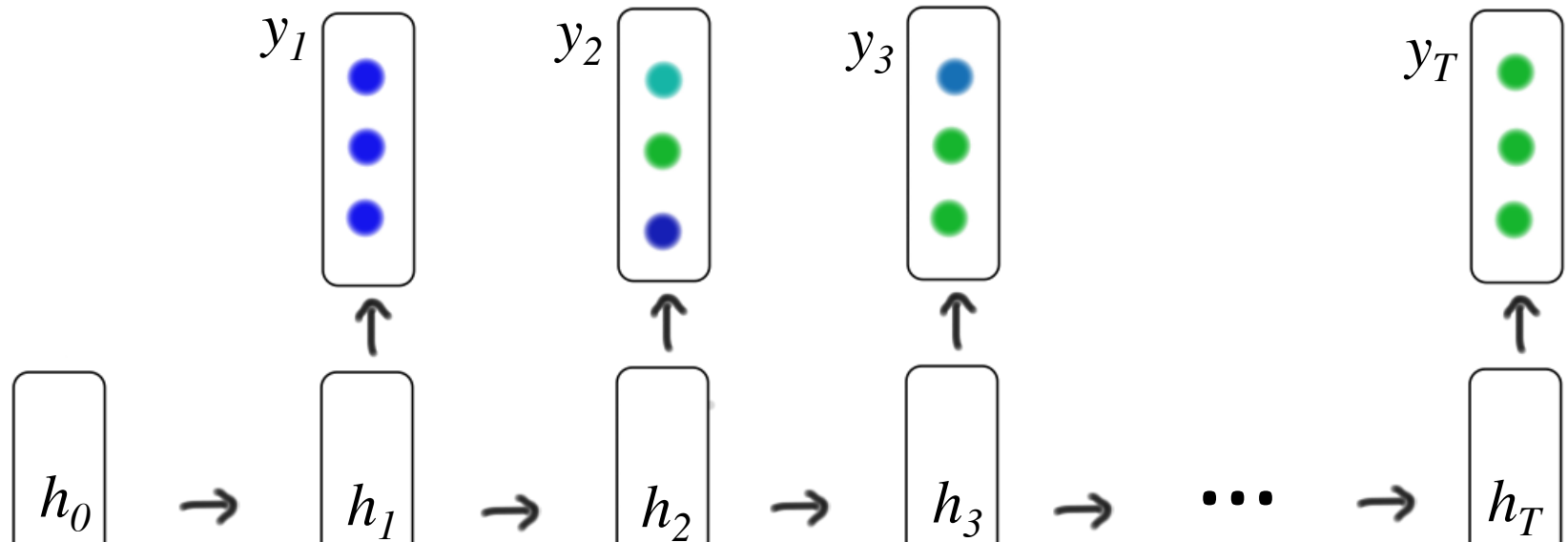
Recurrent Neural Network



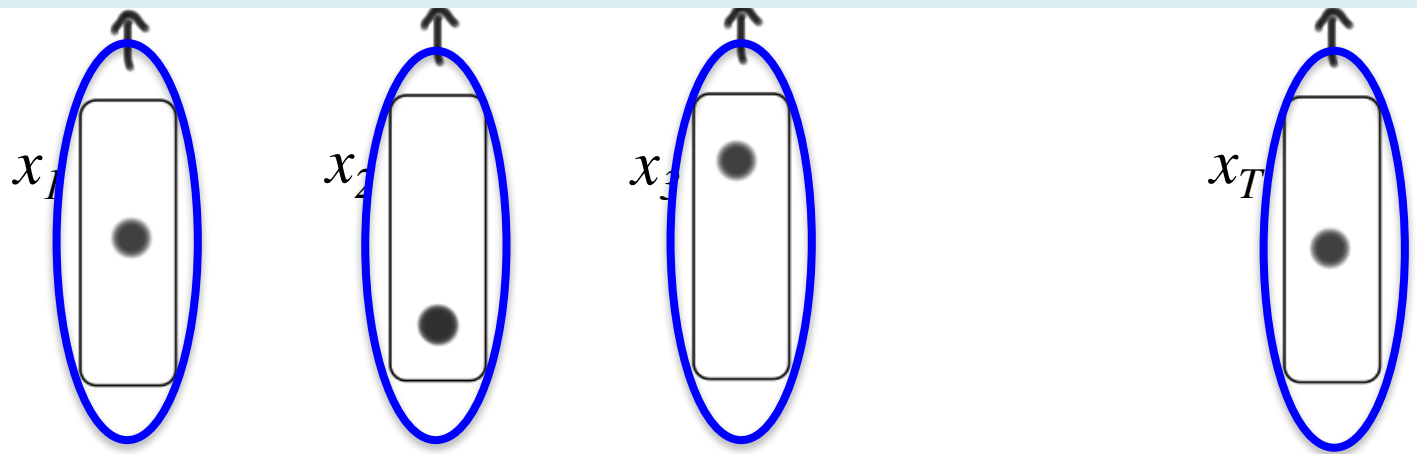
Recurrent Neural Network



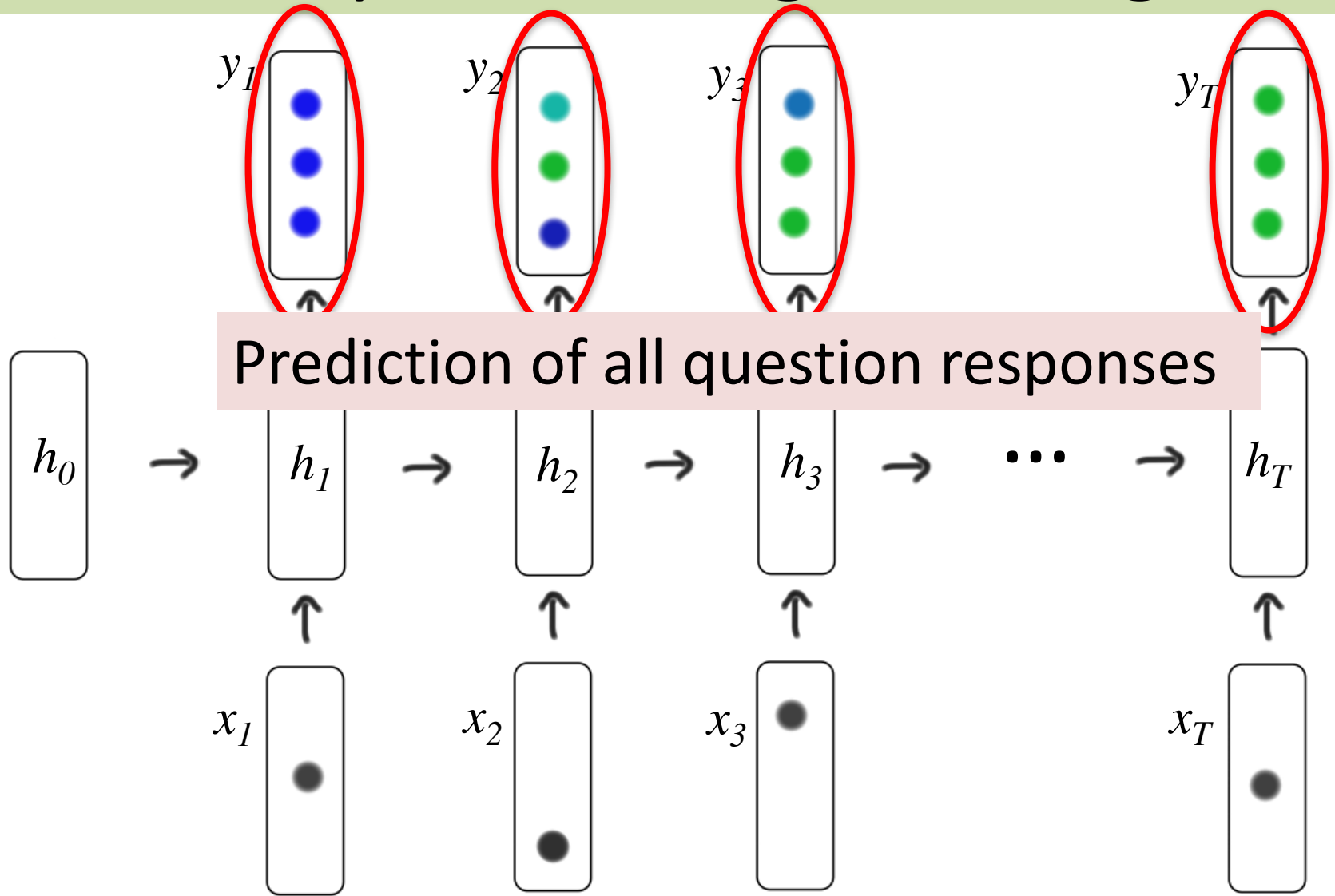
Deep Knowledge Tracing



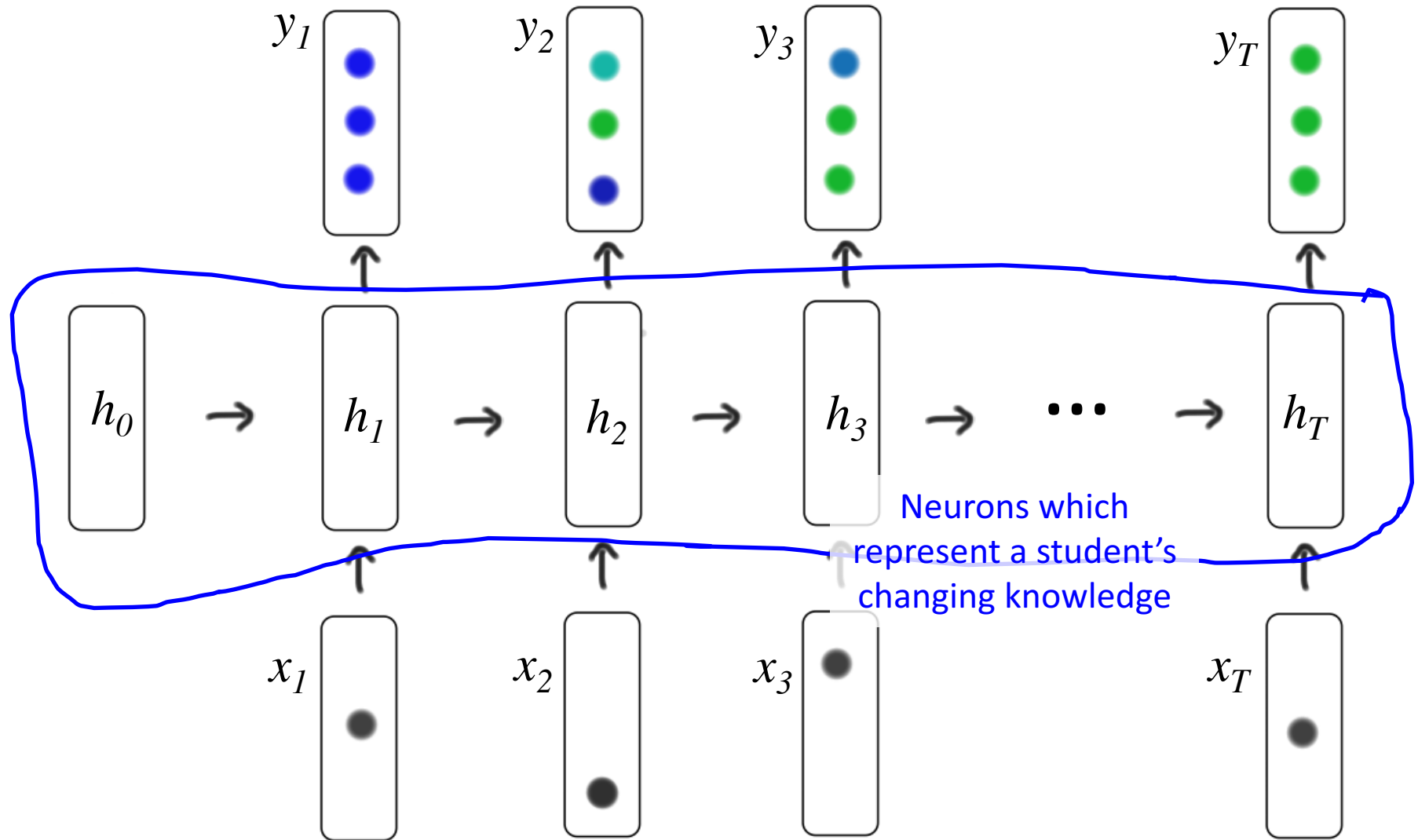
One hot encoding of question id and correct (q_i, a_i)



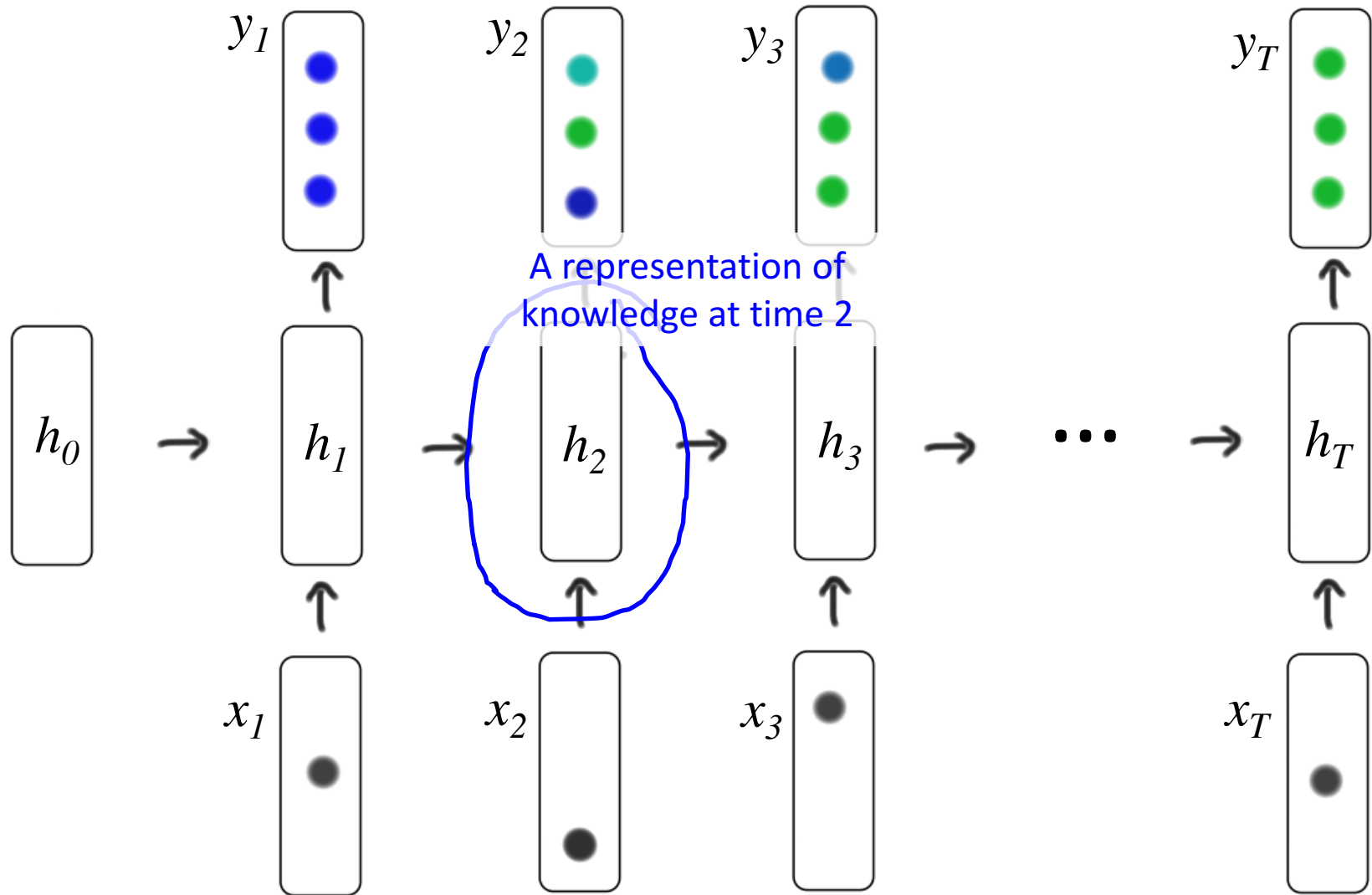
Deep Knowledge Tracing



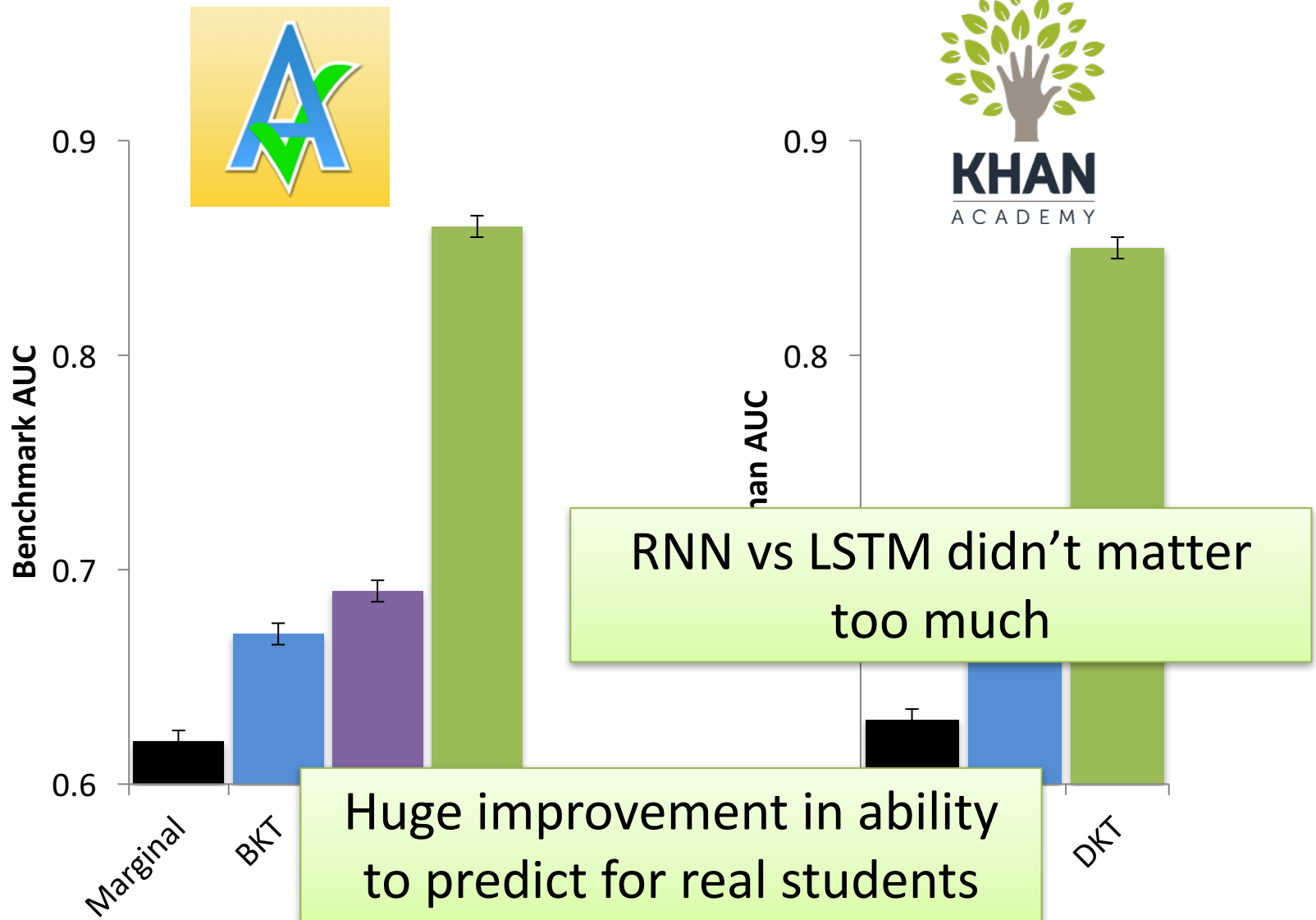
Recurrent Neural Network



Recurrent Neural Network



Prediction Results



Interpretation

If you can't do
problem X

Then you can't do
problem Y either.

But, if you can do
problem X

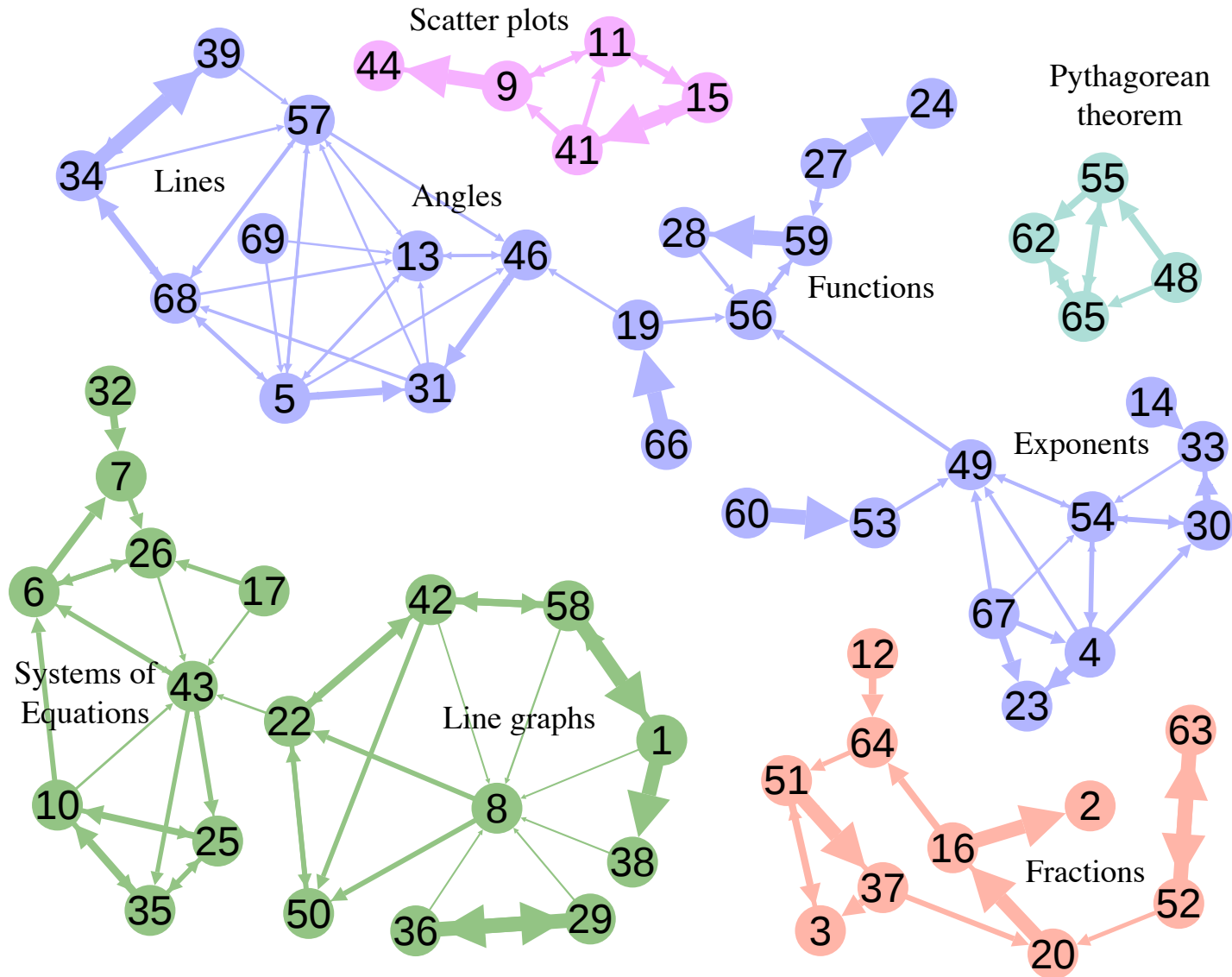
Then you can do
problem Y too.

$$\rho(X, Y) = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X)\text{Var}(Y)}}$$

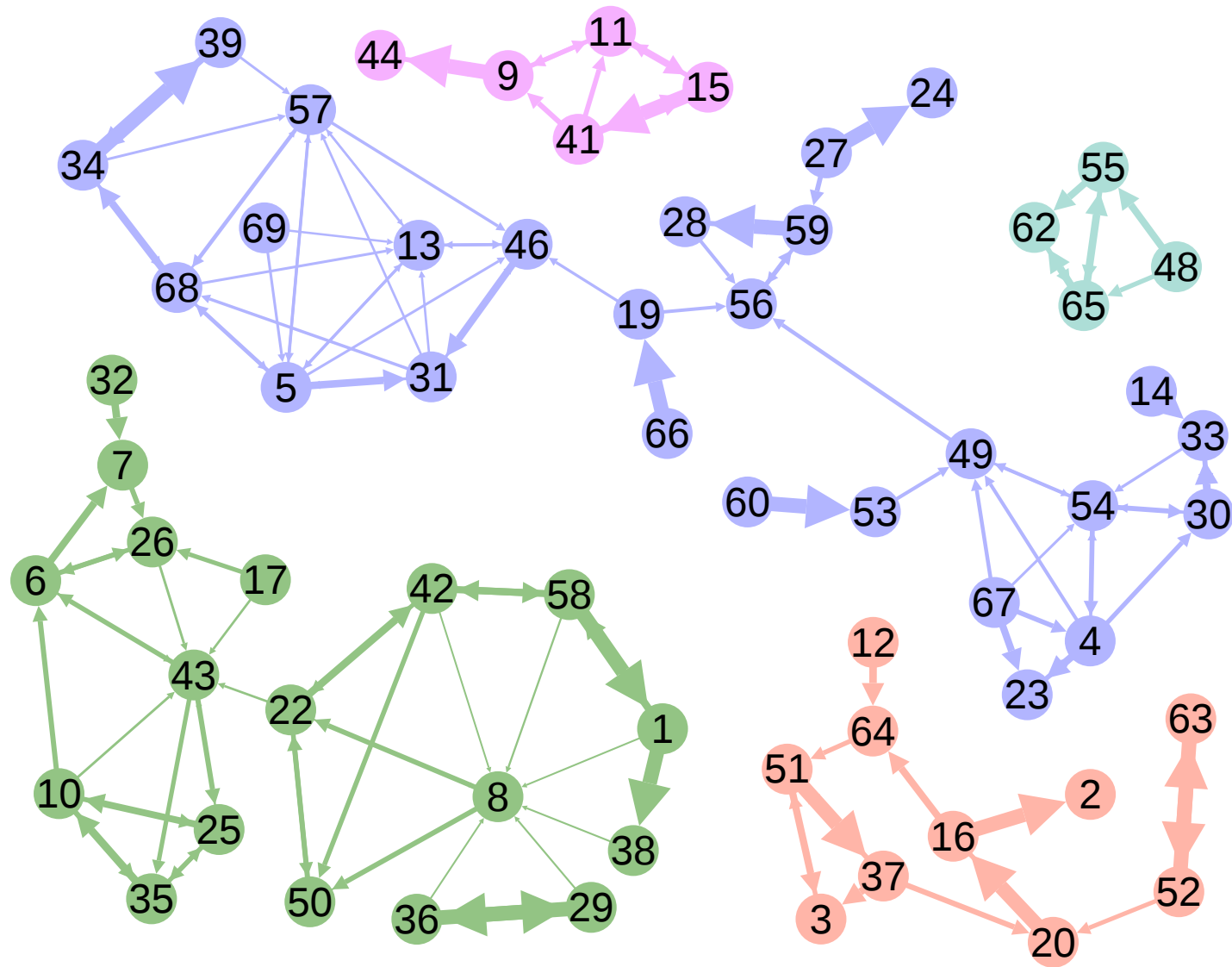
$$\begin{aligned}\text{Cov}(X, Y) &= E[XY] - E[X]E[Y] \\ &= P(XY) - P(X)P(Y)\end{aligned}$$



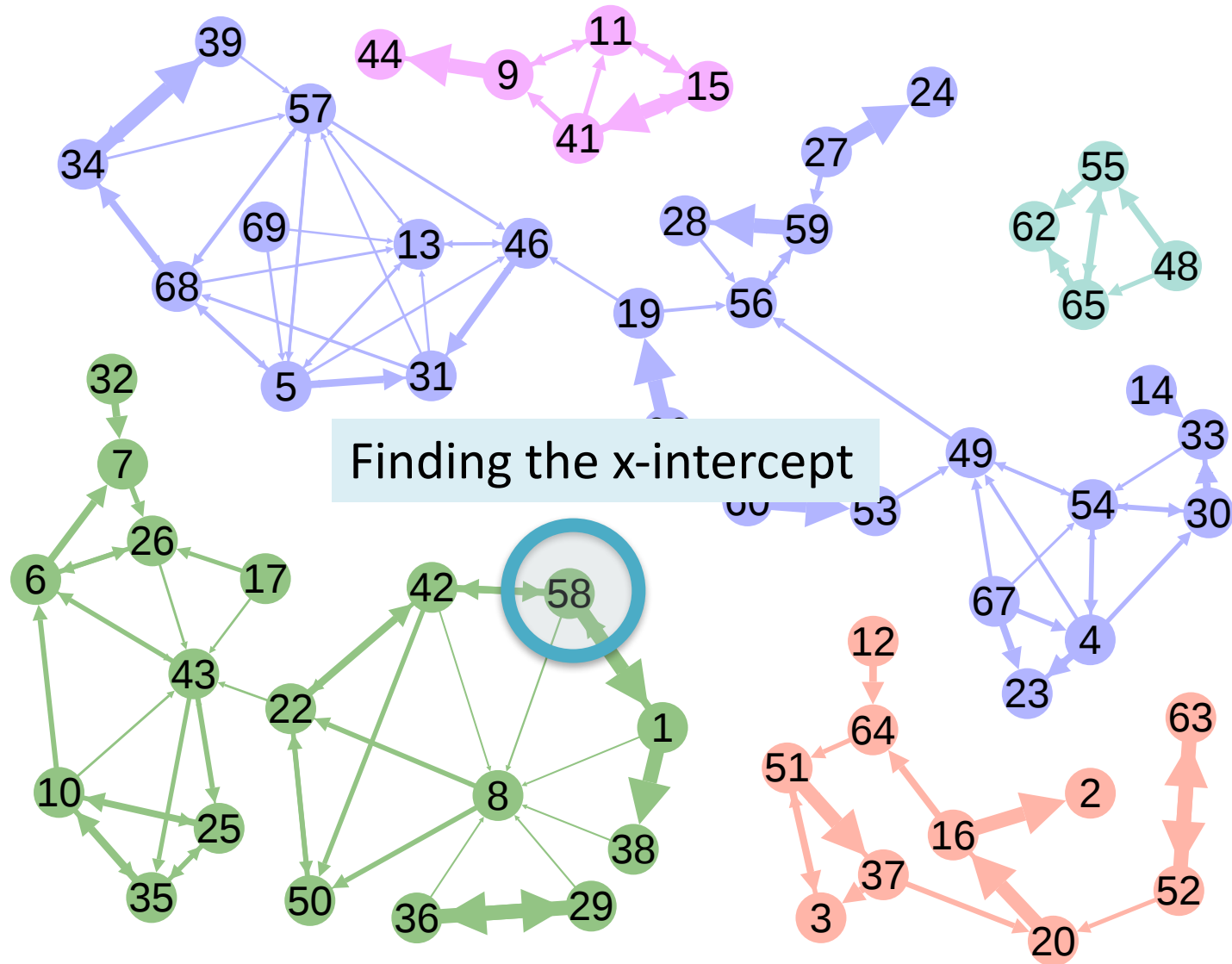
Learns Concept Relationships



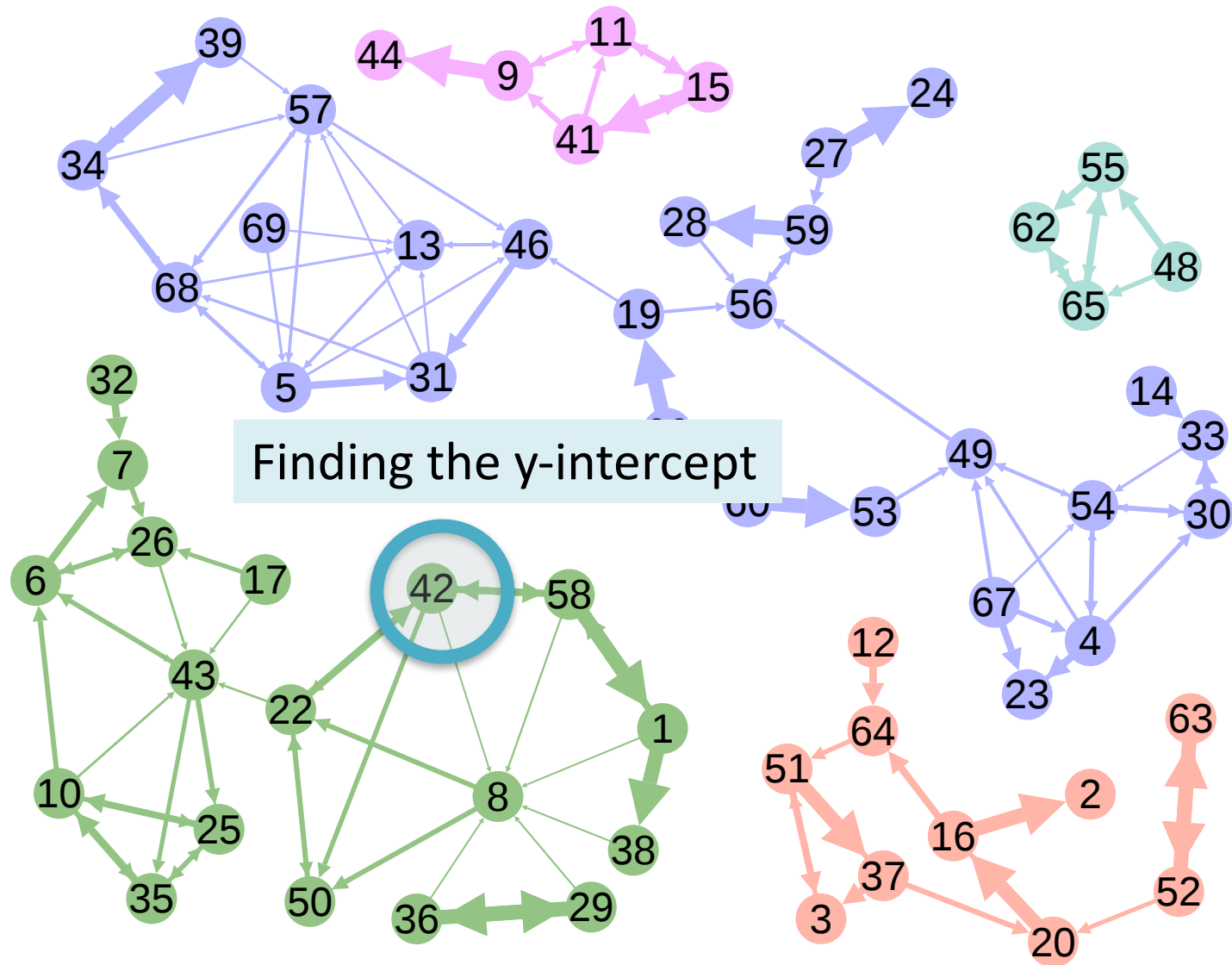
Learns Concept Relationships



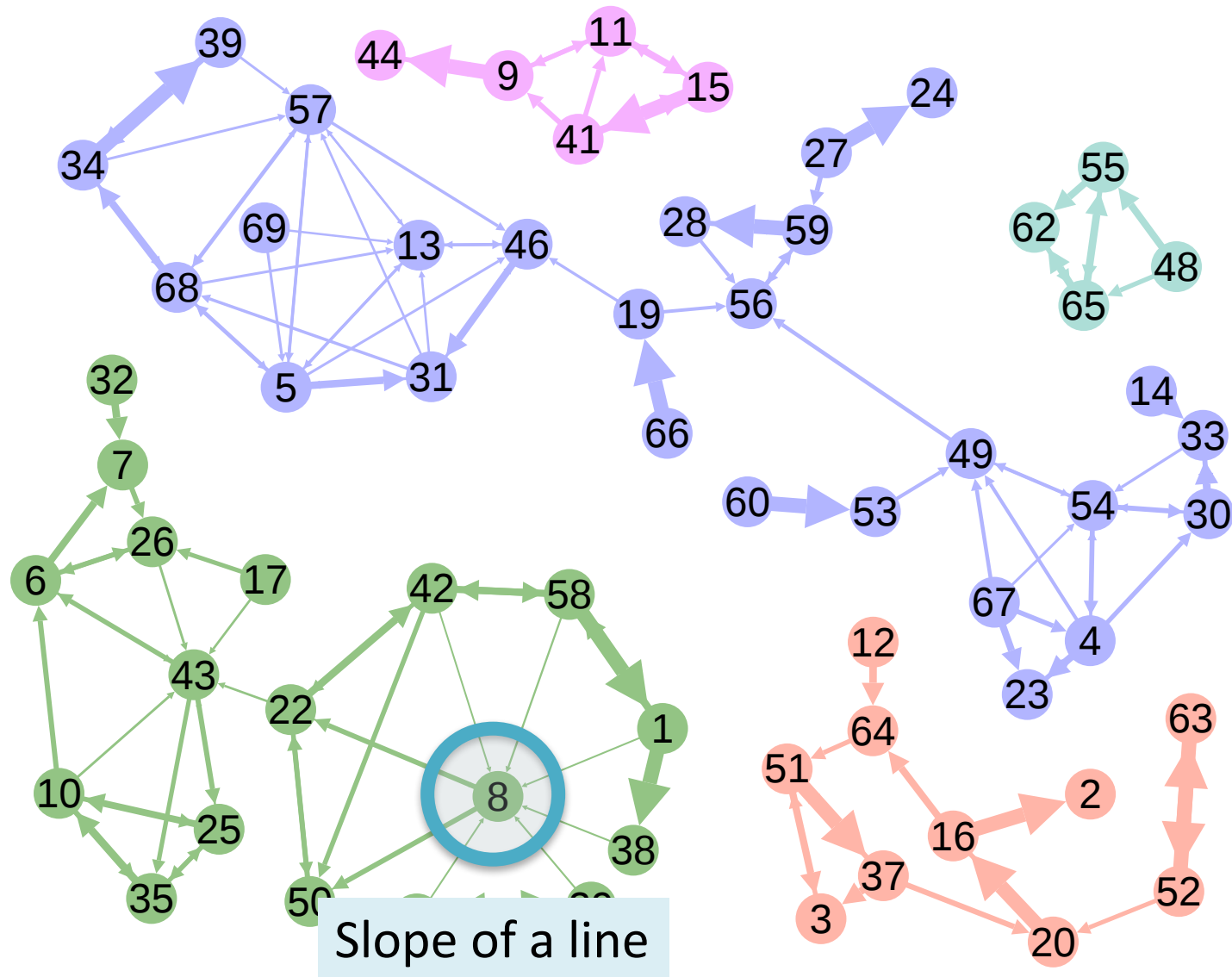
Learns Concept Relationships



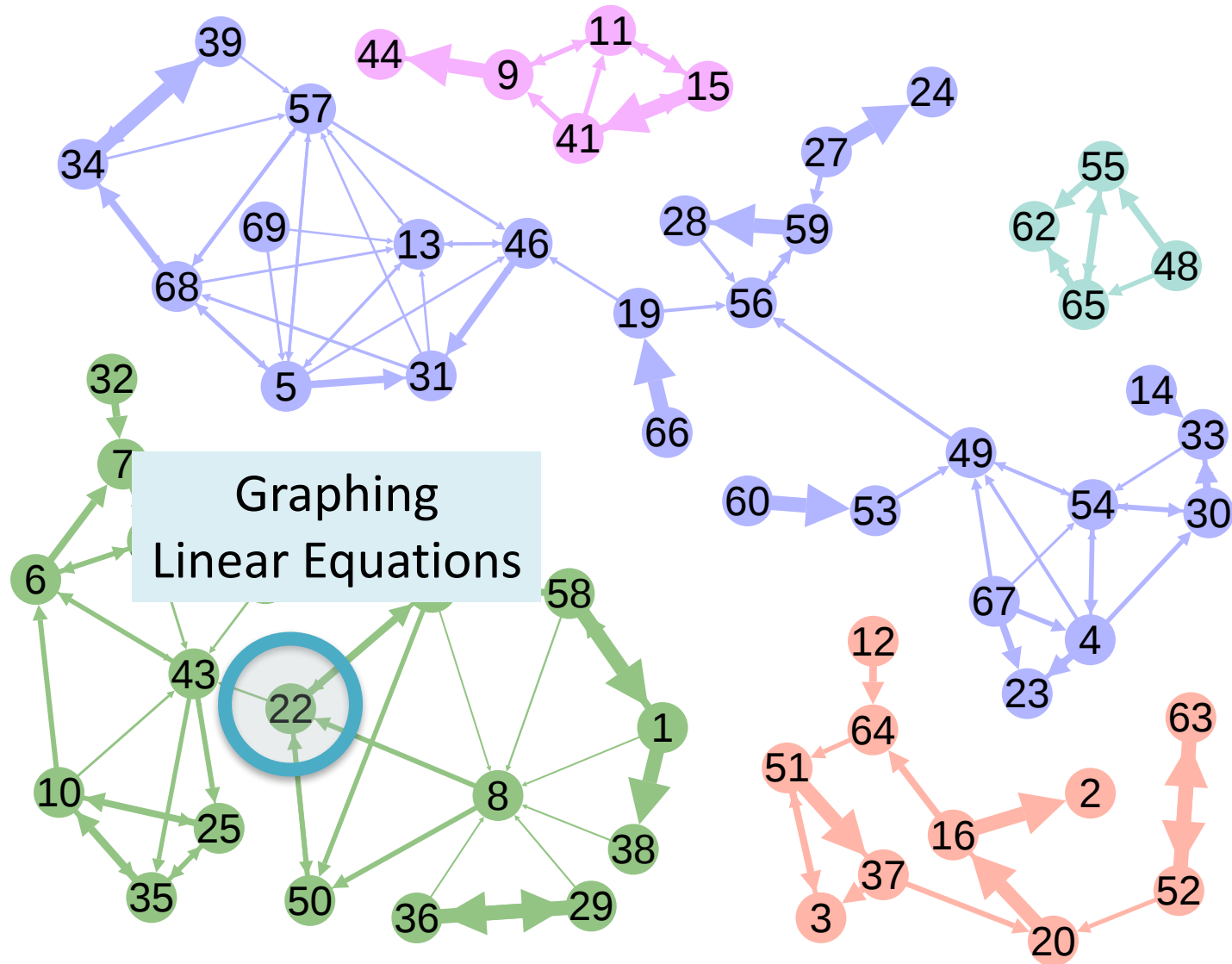
Learns Concept Relationships



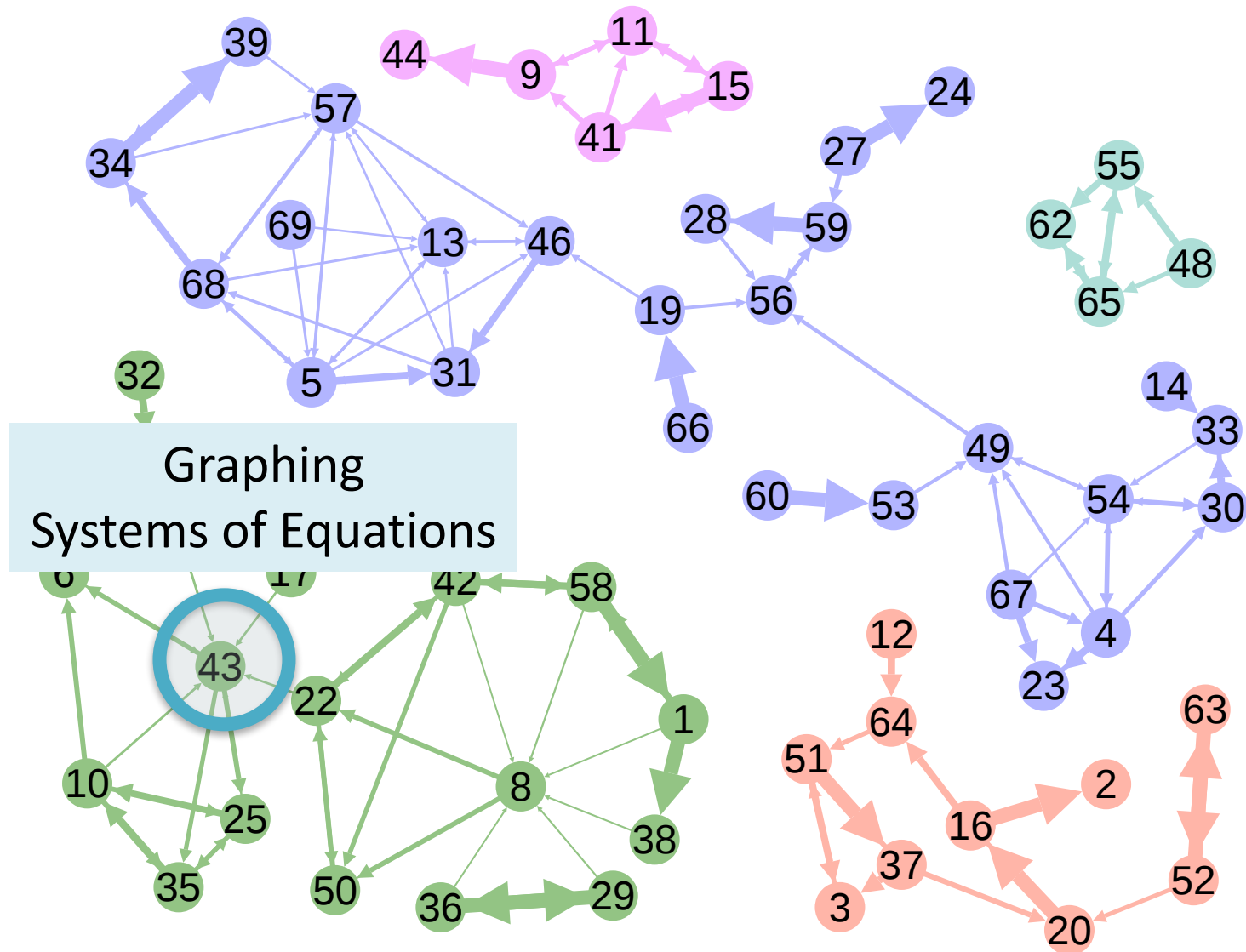
Learns Concept Relationships



Learns Concept Relationships



Learns Concept Relationships



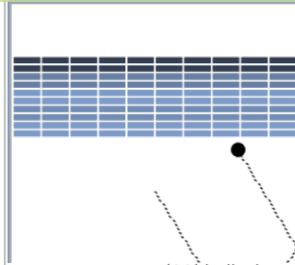
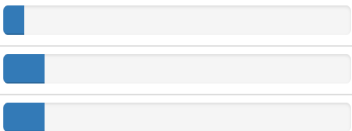
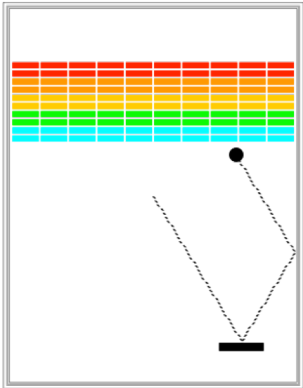
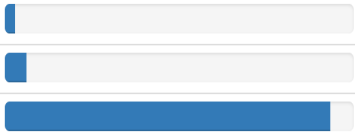
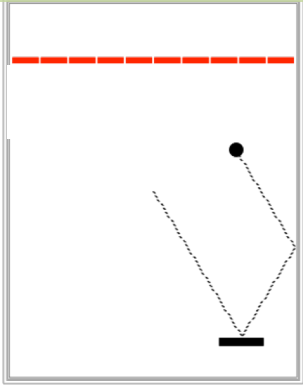
Let's try it!



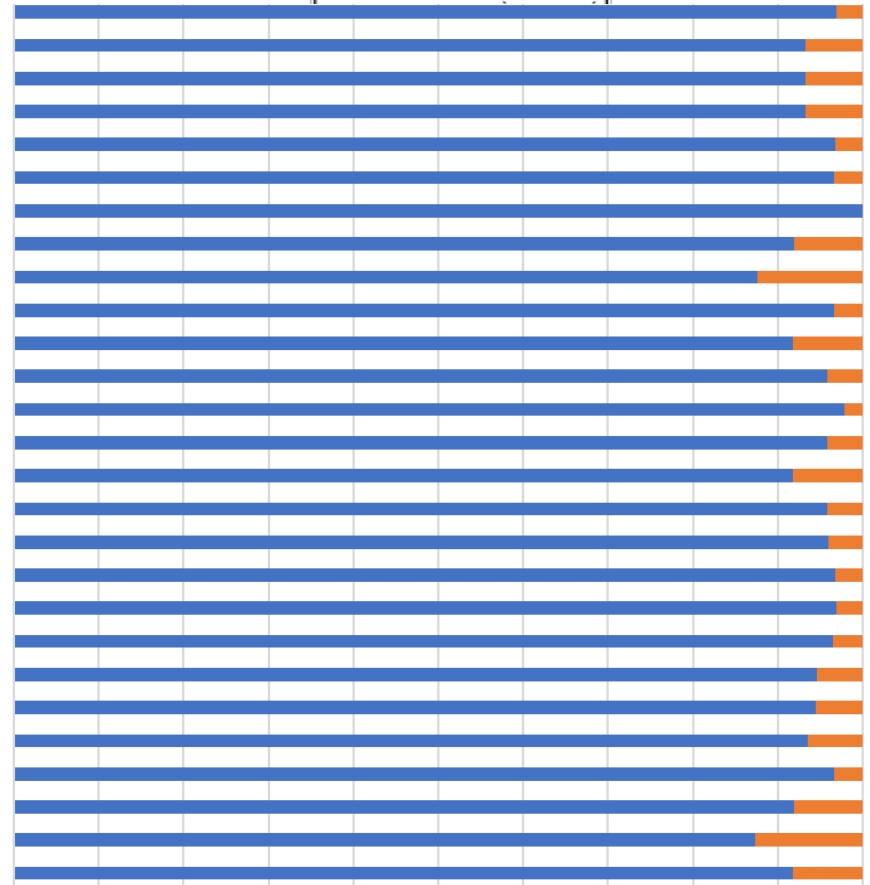
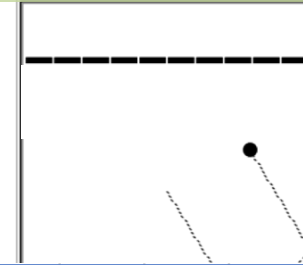
iHDWA



In a Computer Near You



- (32) ball x bounce
- (33) ball top off
- (34) ball right off
- (35) ball left off
- (36) paddle bounce wrong
- (37) paddle bounce off
- (38) paddle bounce hard-coded
- (39) ball miss at paddle y
- (40) ball stays after miss
- (41) sticky paddle
- (42) one corner fail
- (43) two corner fail
- (44) 3 corner fail
- (45) $v_y \neq -v_y$ bounce
- (46) double collision wrong
- (47) single collision wrong
- (48) brick not removed
- (49) brick sometimes not removed
- (50) paddle removed
- (51) no replay
- (52) < 3 lives
- (53) does not stop after 3 lives
- (54) bricks reset always
- (55) ball not random direction
- (56) no win condition
- (57) sometimes win condition
- (58) doesn't stop animation



■ Correct ■ Incorrect



Ethics in AI