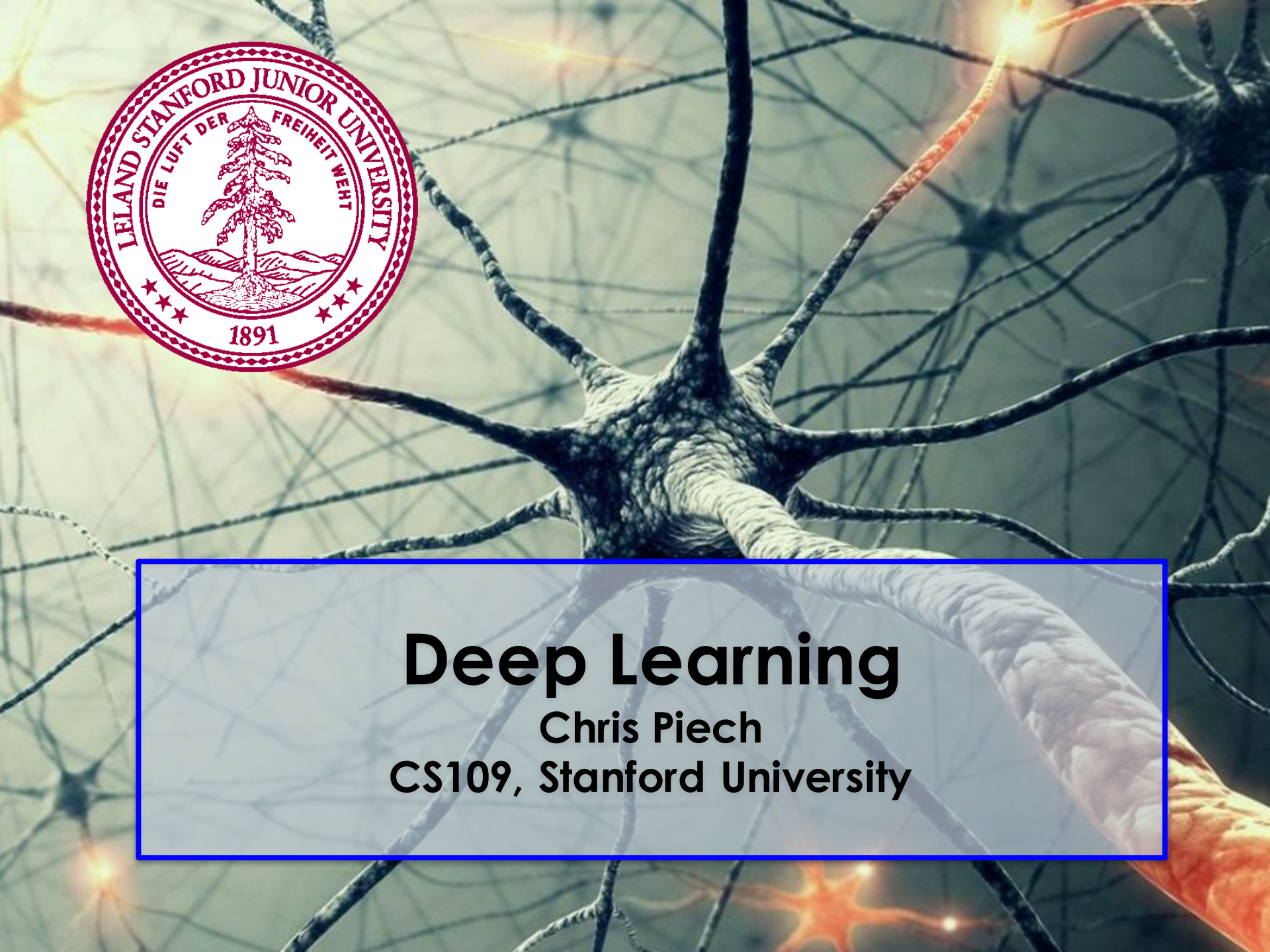


Deep Learning

Chris Piech
CS109, Stanford University

Important Mathematical Journey

Logistic Regression:

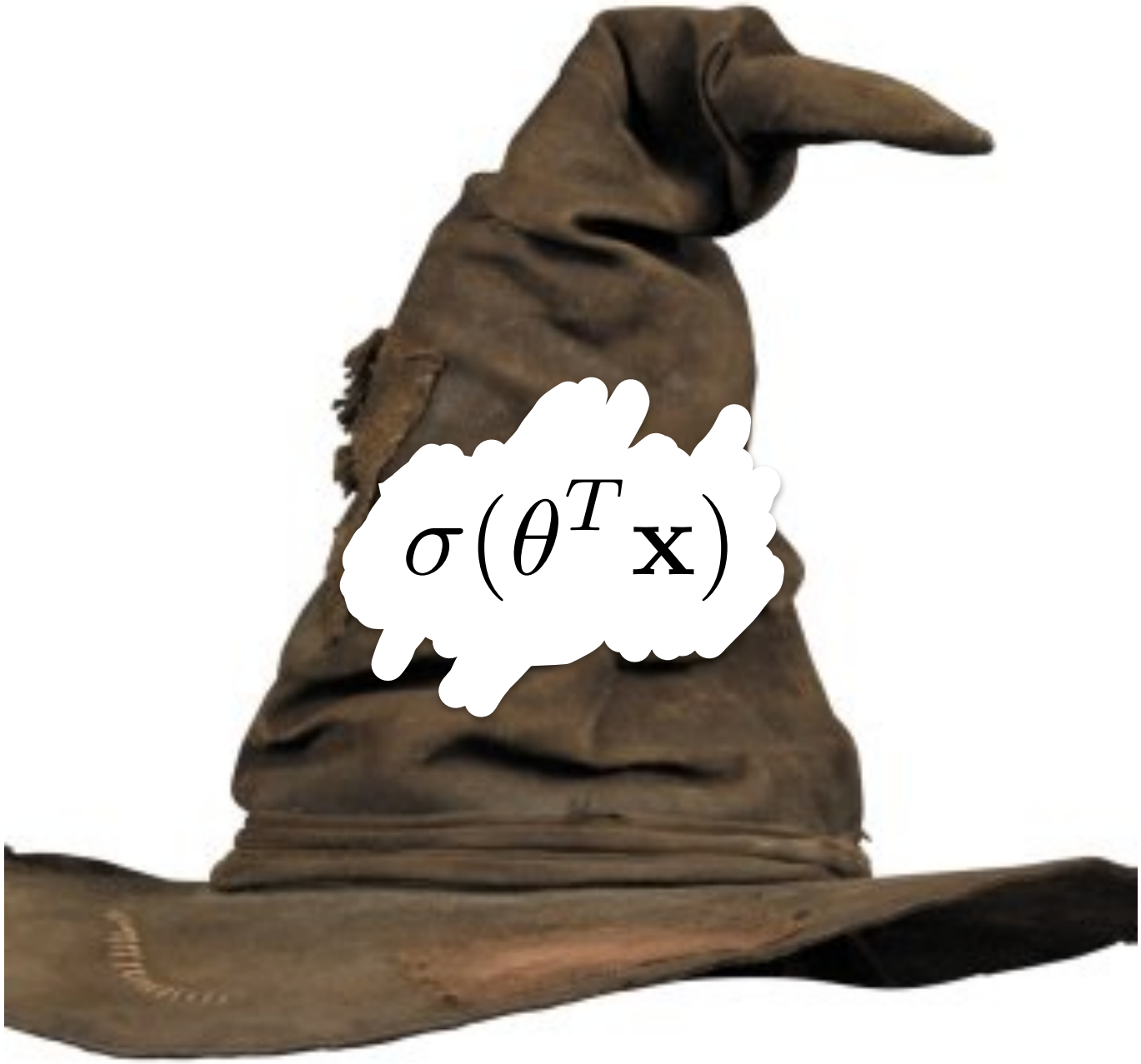
Define a function that predicts P(Y = 1) directly

Logistic Regression is like the Harry Pottery Sorting Hat

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

$$\mathbf{X}$$

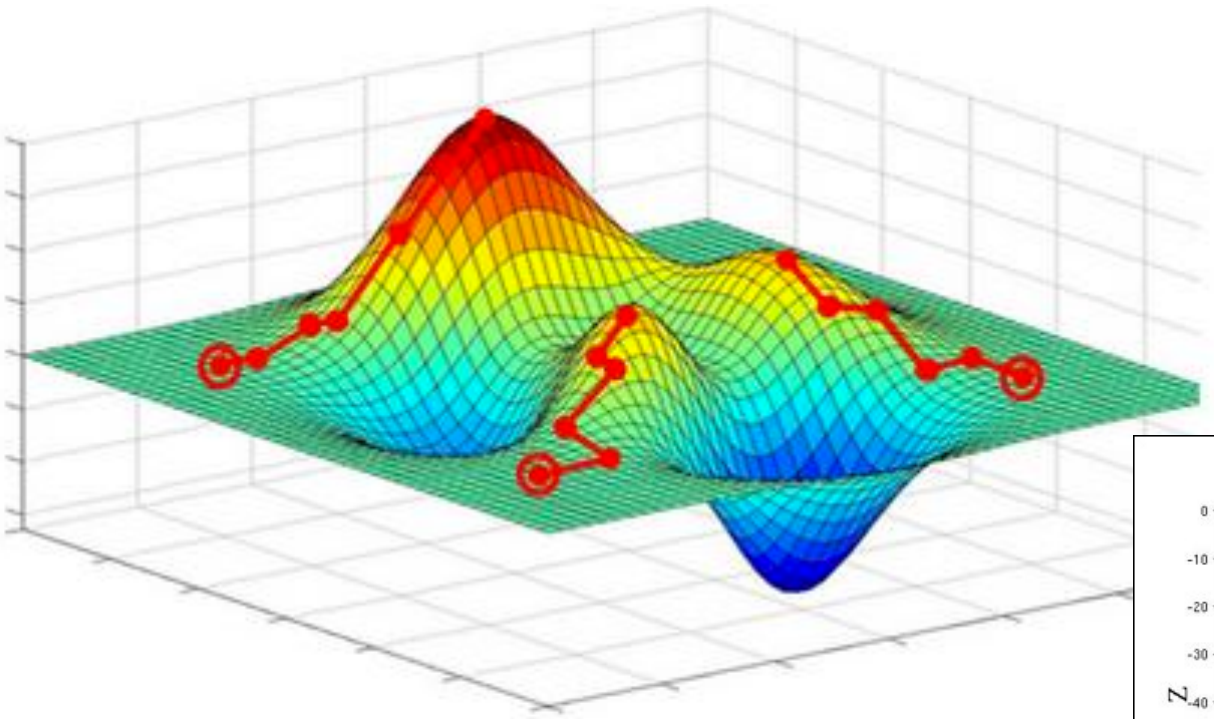
Logistic Regression is like the Harry Pottery Sorting Hat



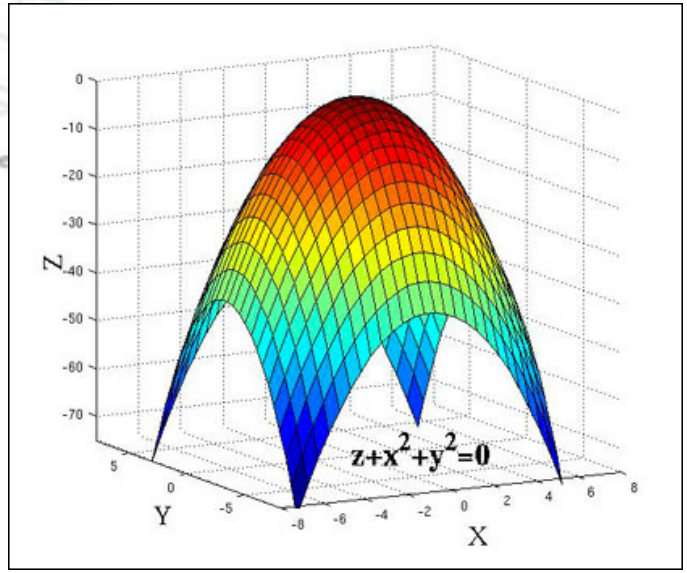
Logistic regression gets its ***intelligence*** from its thetas (aka its parameters)

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

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

3 Get derivative of log likelihood with respect to thetas

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

Logistic Regression Training

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

Repeat many times:

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

For each training example (x, y):

For each parameter j:

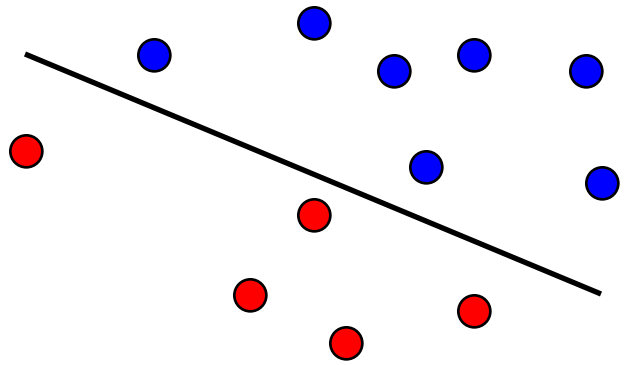
gradient[j] $+= \left[y - \sigma(\theta^T \mathbf{x})\right] \mathbf{x}_j$

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

Chapter 3: Philosophy

Discrimination Intuition

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



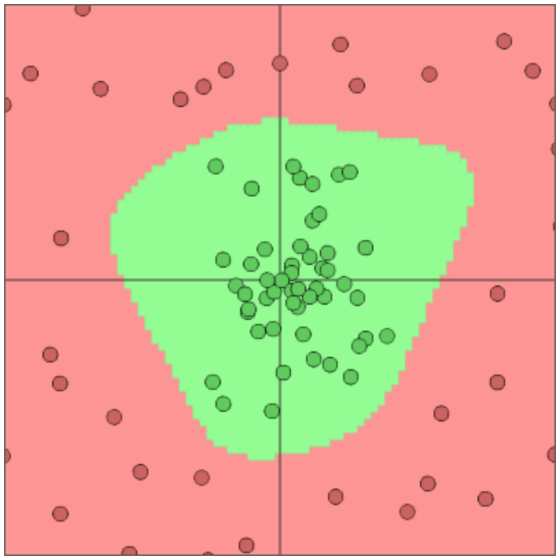
$$\theta^T \mathbf{x} = 0$$

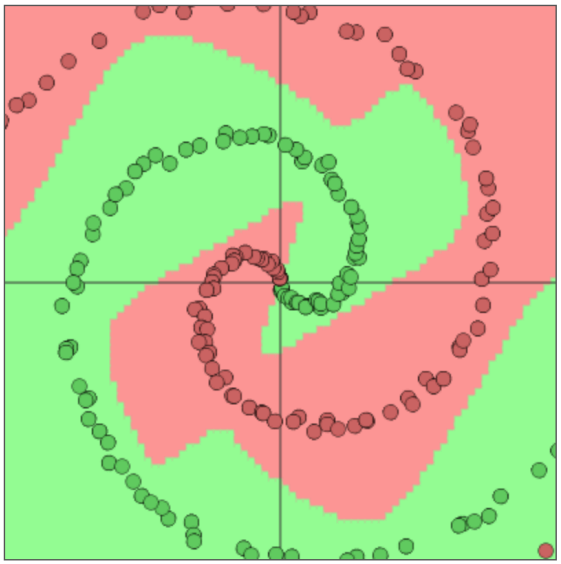
$$\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_m x_m = 0$$

- 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

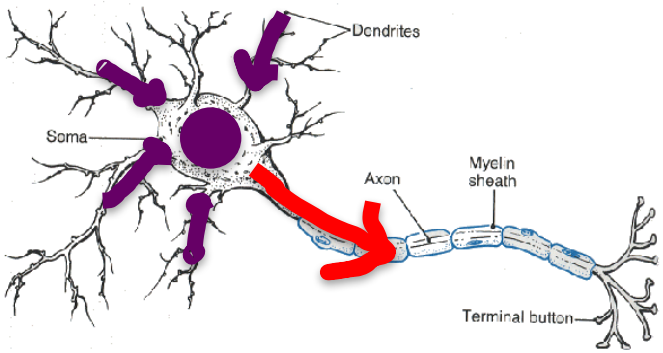
Logistic Regression vs Naïve Bayes

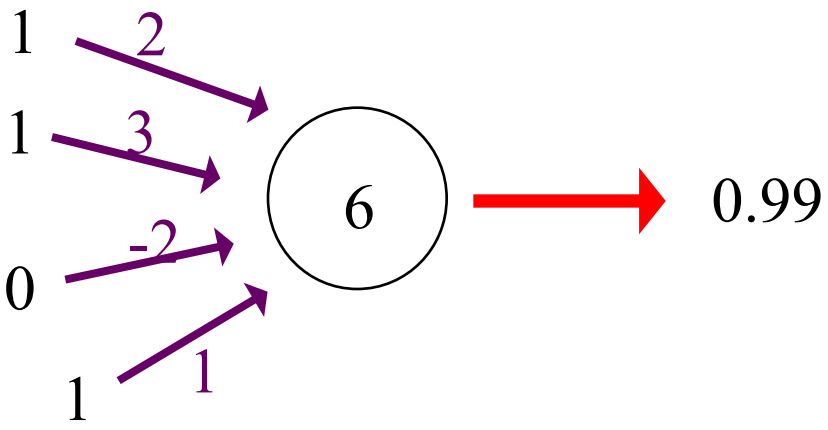
- Compare Naive Bayes and Logistic Regression
  - Recall that Naive Bayes models P(**X**, Y) = P(**X** | Y) P(Y)
  - Logistic Regression directly models P(Y | **X**)
  - We call Naive Bayes a "generative model"
    - Tries to model joint distribution of how data is "generated"
    - I.e., could use P(X,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

Choosing an Algorithm?

- Many trade-offs in choosing learning algorithm
  - Continuous input variables
    - Logistic Regression easily deals with continuous inputs
    - Naive Bayes needs to use some parametric form for continuous inputs (e.g., Gaussian) or "discretize" continuous values into ranges (e.g., temperature in range: <50, 50-60, 60-70, >70)
  - Discrete input variables
    - Naive Bayes naturally handles multi-valued discrete data by using multinomial distribution for $P(X_i \mid Y)$
    - Logistic Regression requires some sort of representation of multi-valued discrete data (e.g., one hot vector)
    - Say $X_i \in \{A, B, C\}$. Not necessarily a good idea to encode X_i as taking on input values 1, 2, or 3 corresponding to A, B, or C.

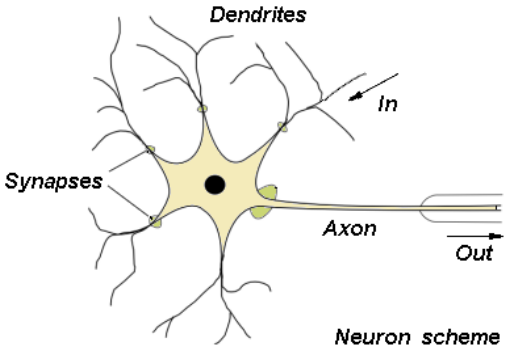
Artificial Neuron

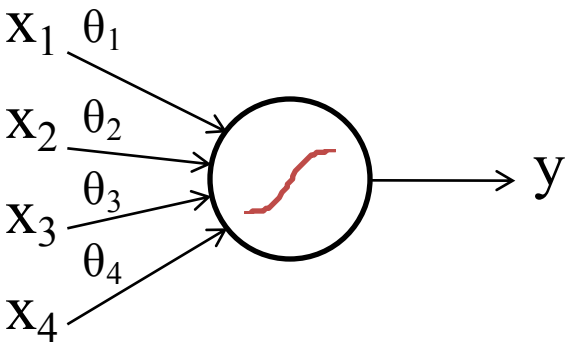




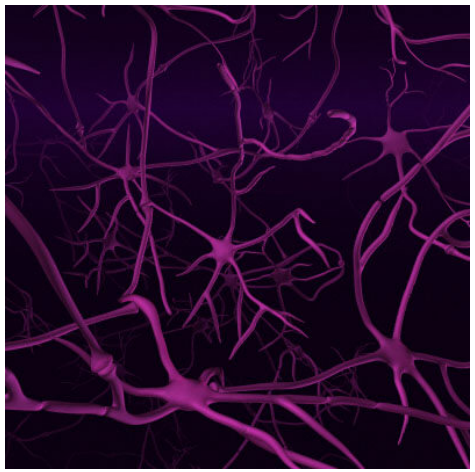
Biological Basis for Neural Networks

- A neuron

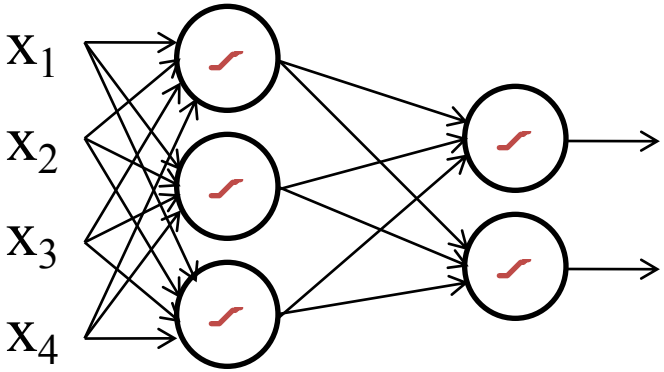




- Your brain



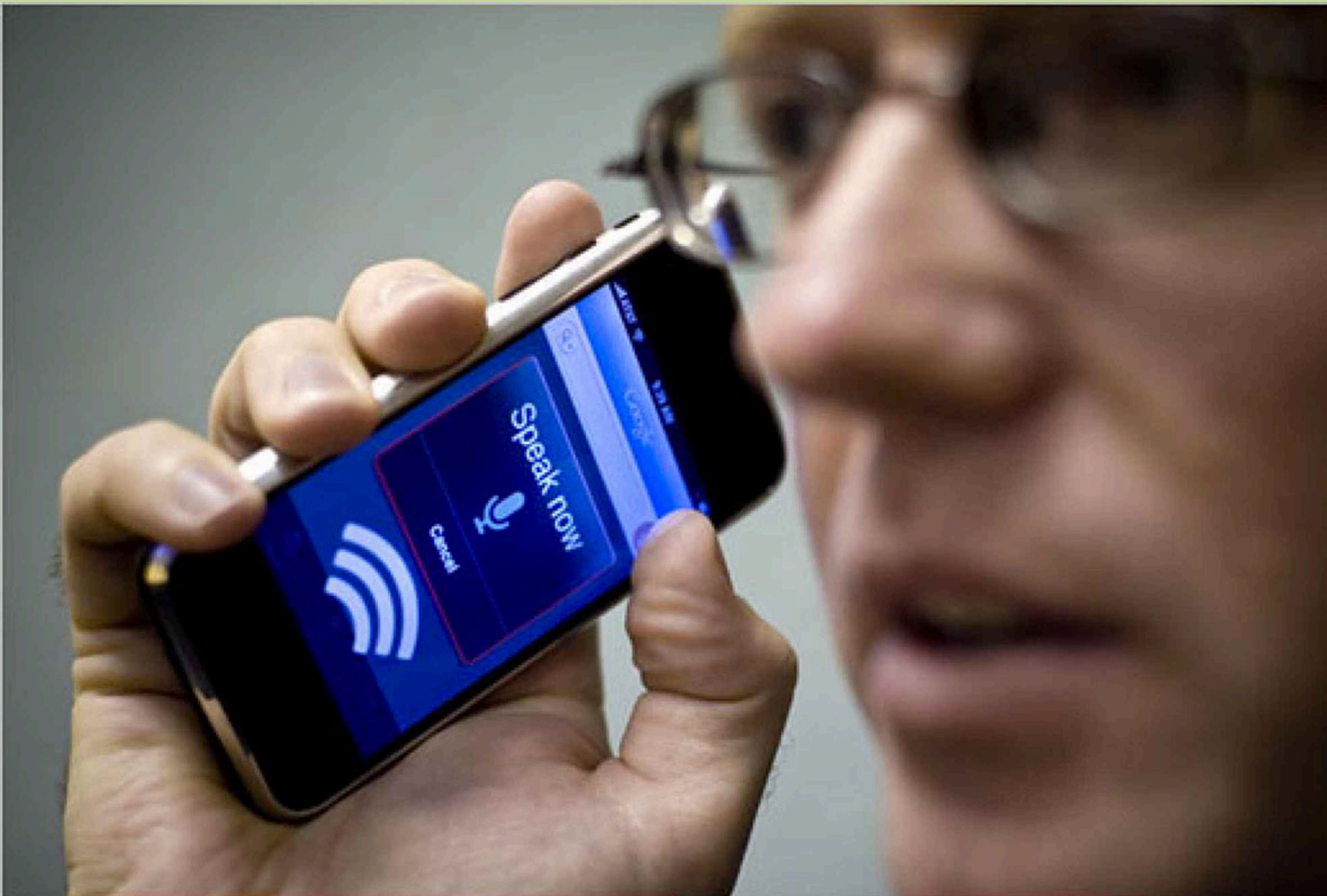
Actually, it's probably someone else's brain



Something happening in the world of AI

Alpha GO

Revolution in AI



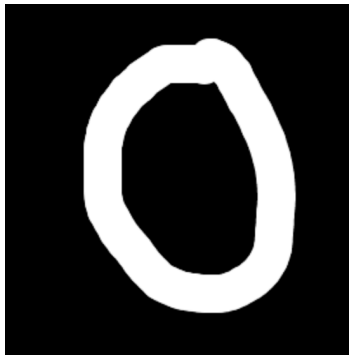
Computers Making Art

(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

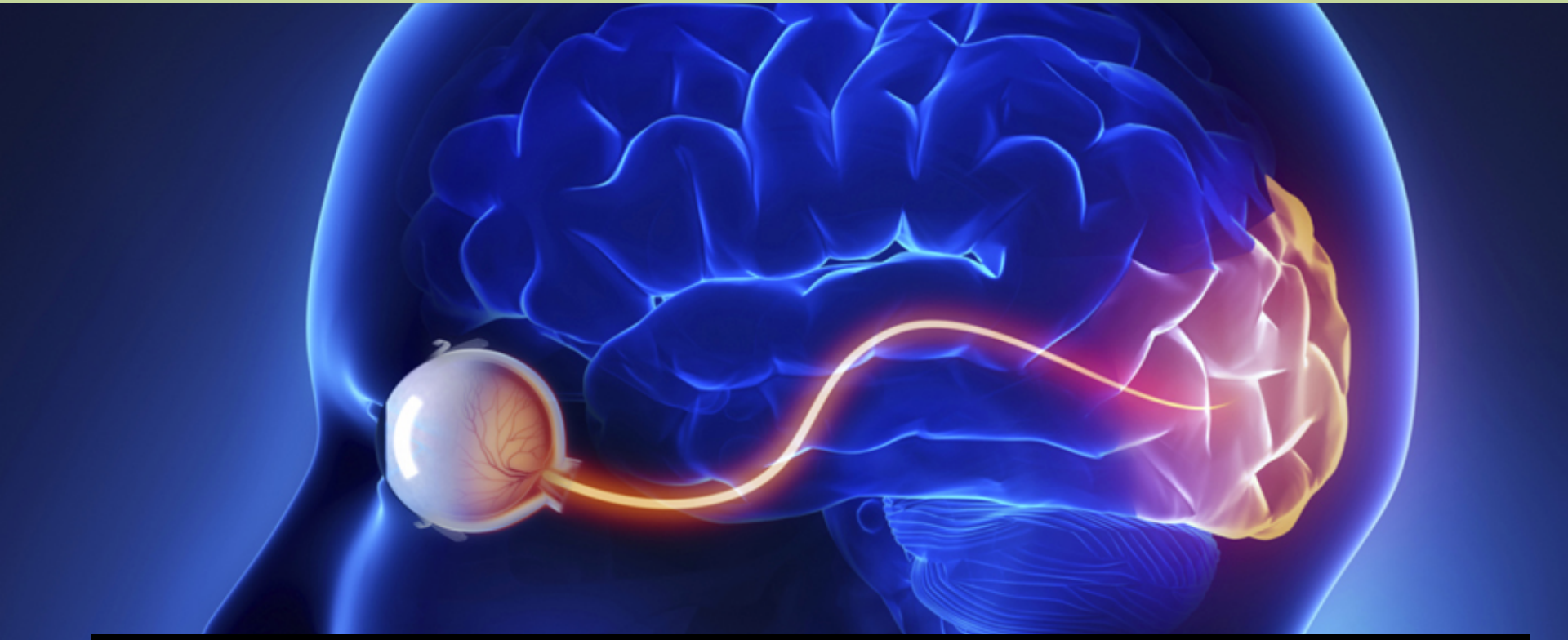


$$\mathbf{x}^{(i)} = [0, 0, 0, 0, \ldots, 1, 0, 0, 1, \ldots 0, 0, 1, 0]$$
$$y^{(i)} = 0$$

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

Computer Vision

Vision in your Brain

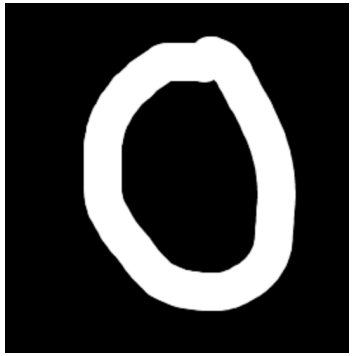


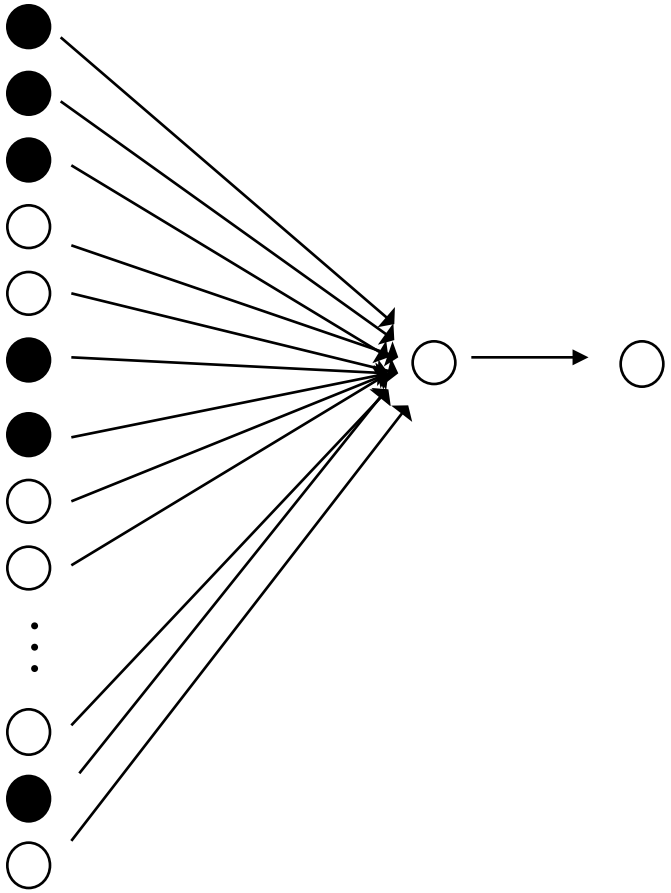
Hundreds of millions of neurons [1]

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

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

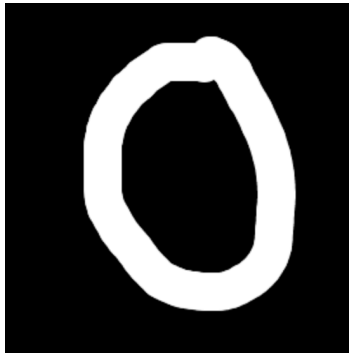
Logistic Regression

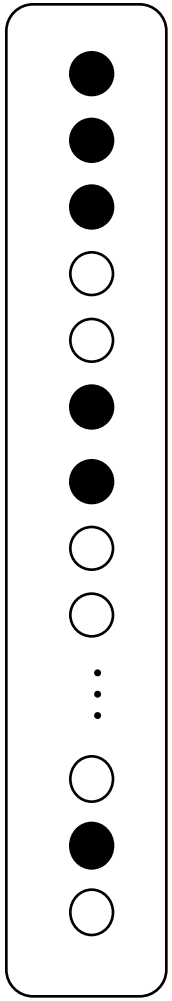




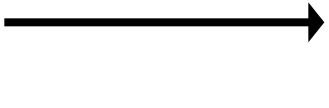
This means it predicts a 0

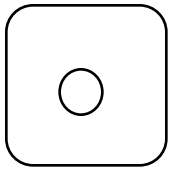
Logistic Regression





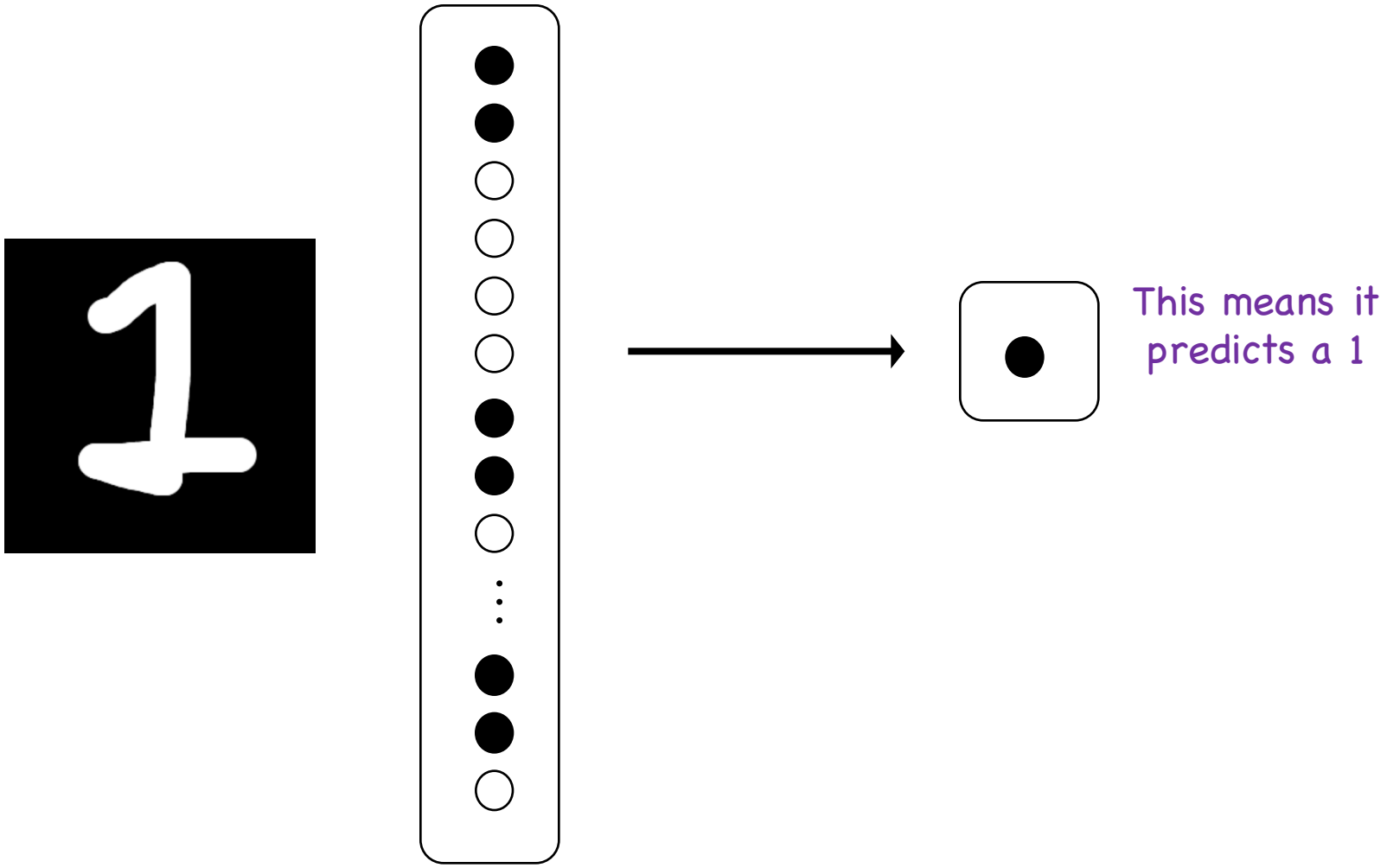
Indicates logistic regression connection



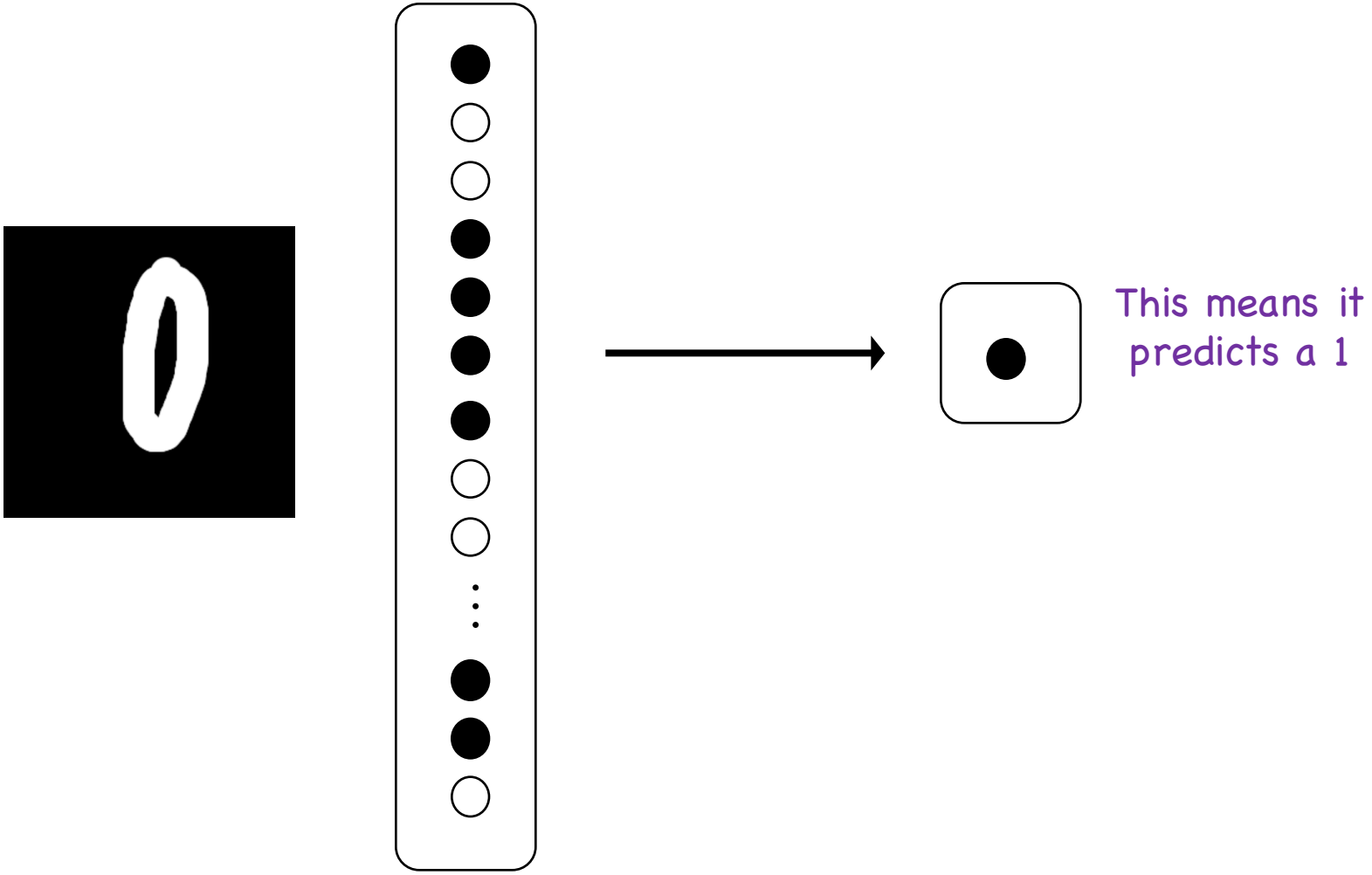


This means it predicts a 0

Logistic Regression

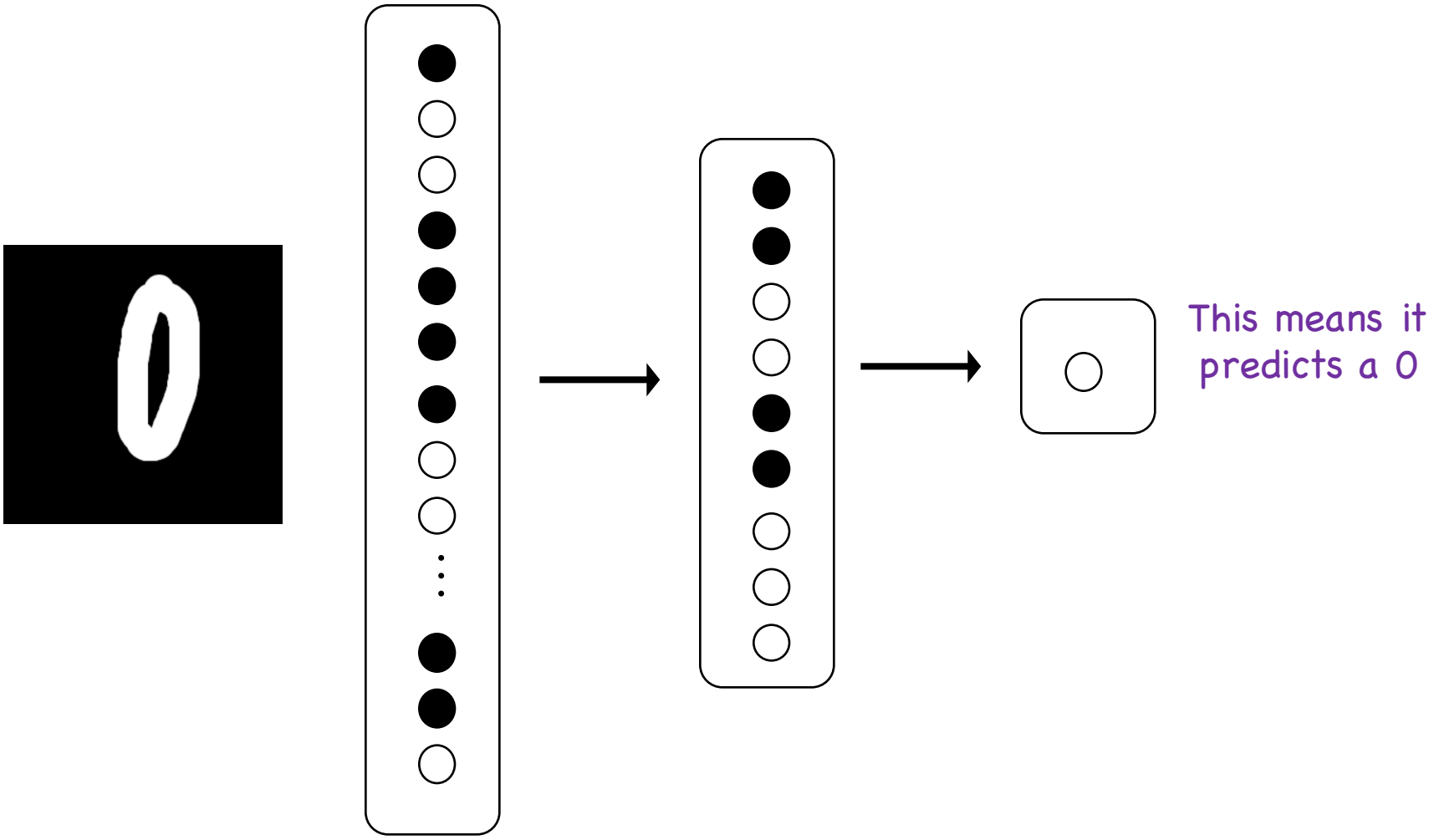


Not So Good

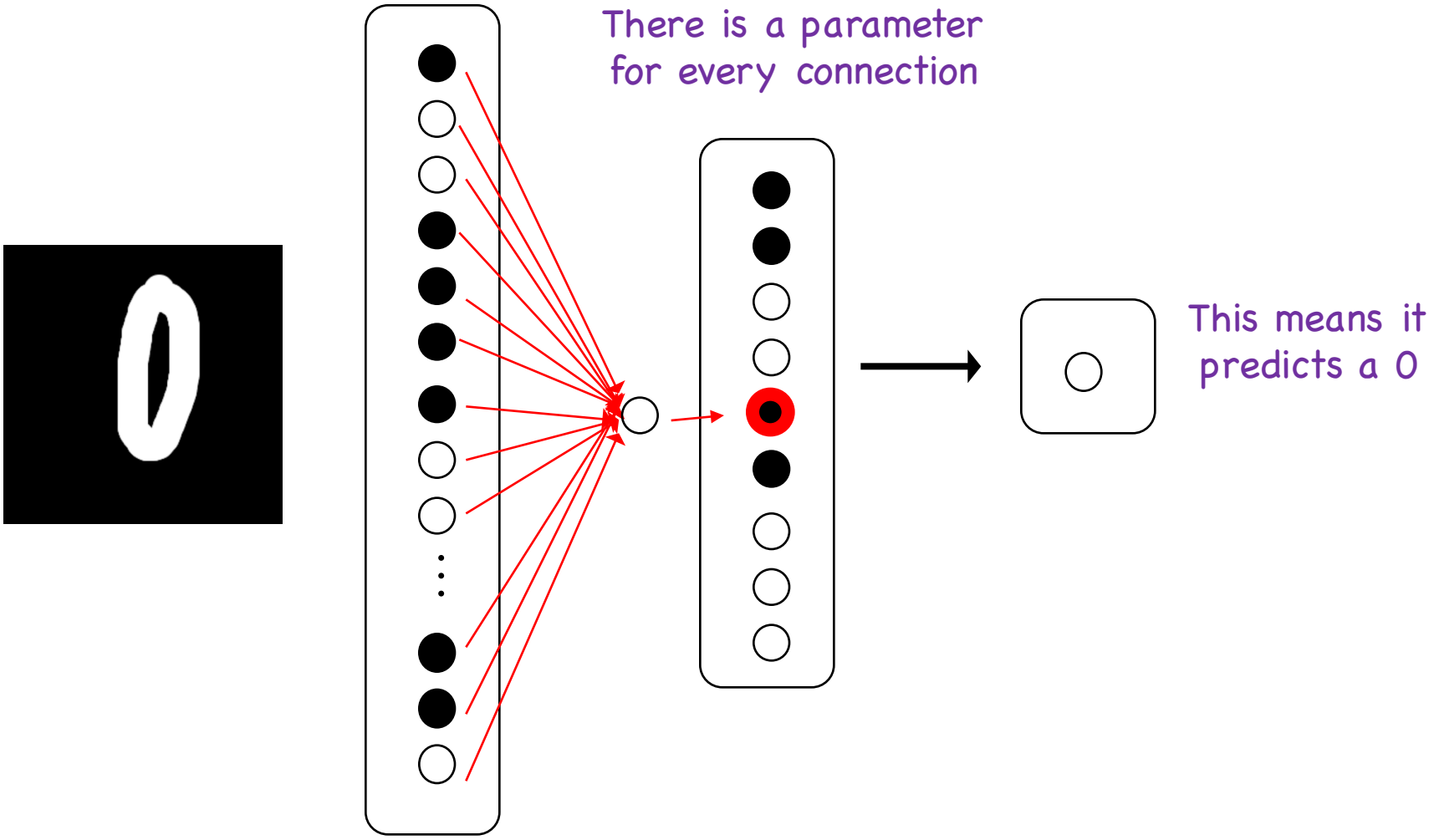


What can we do?

We Can Put Neurons Together

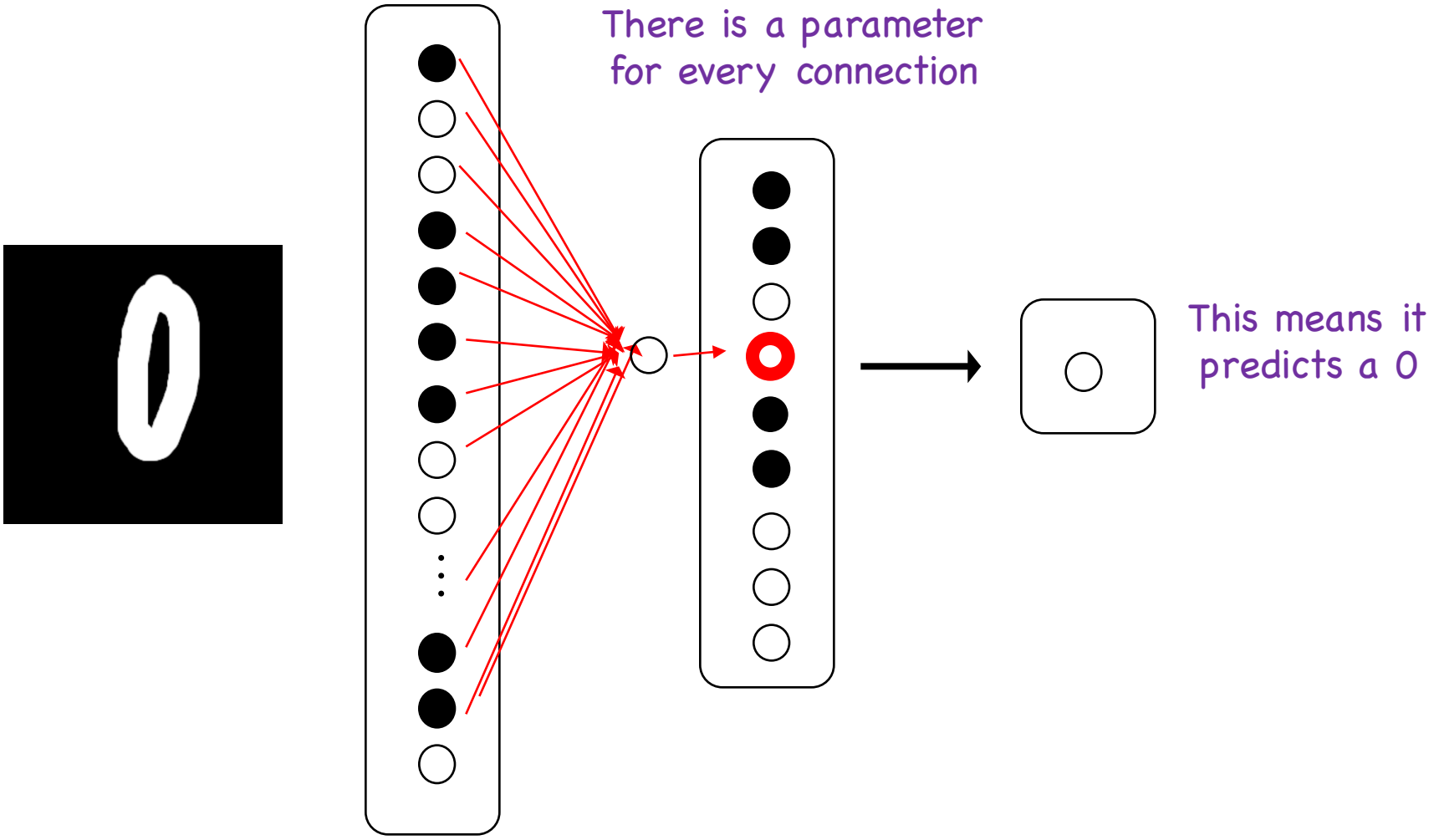


We Can Put Neurons Together



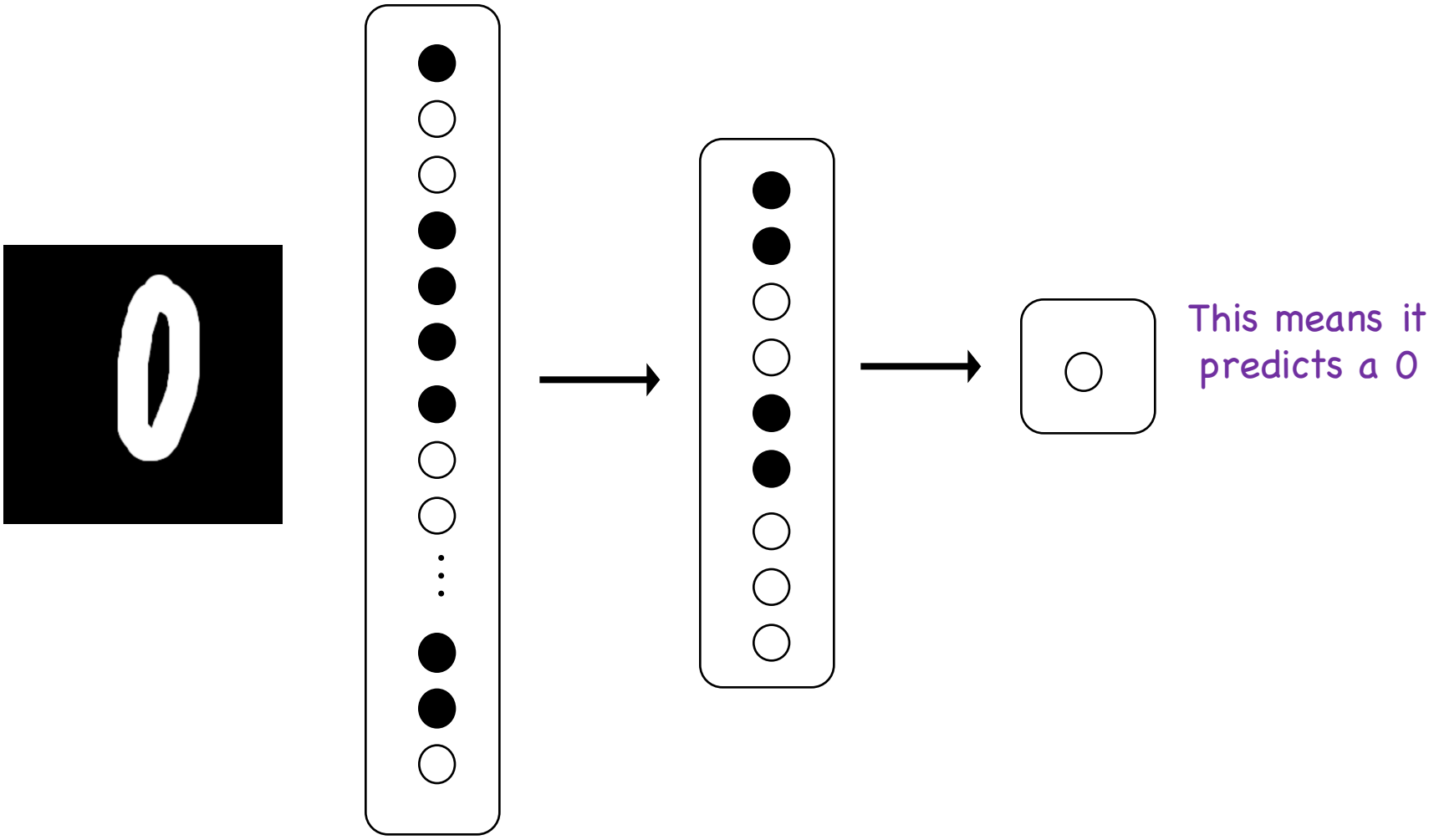
Look at a single "hidden" neuron

We Can Put Neurons Together

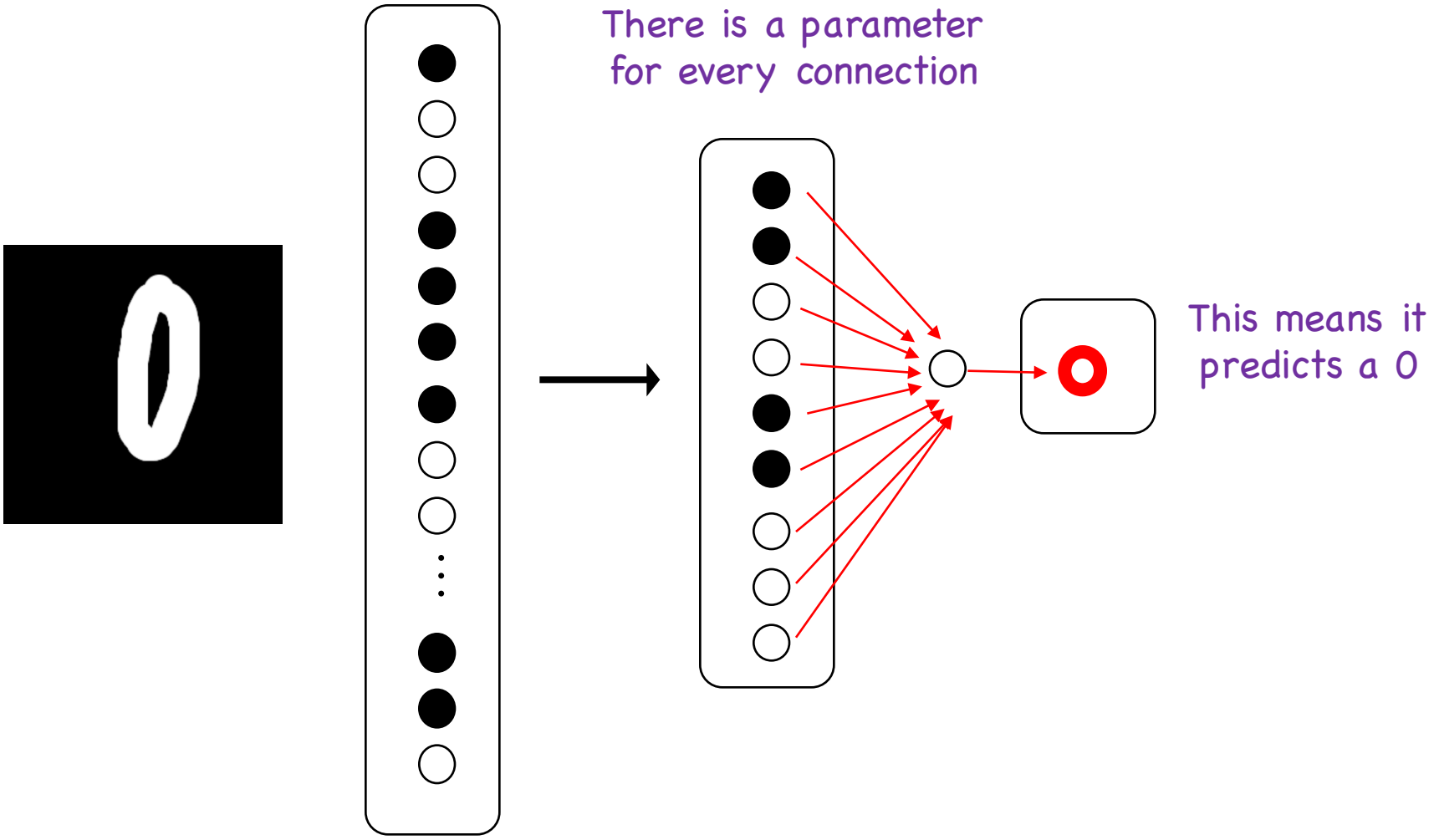


Look at another "hidden" neuron

We Can Put Neurons Together

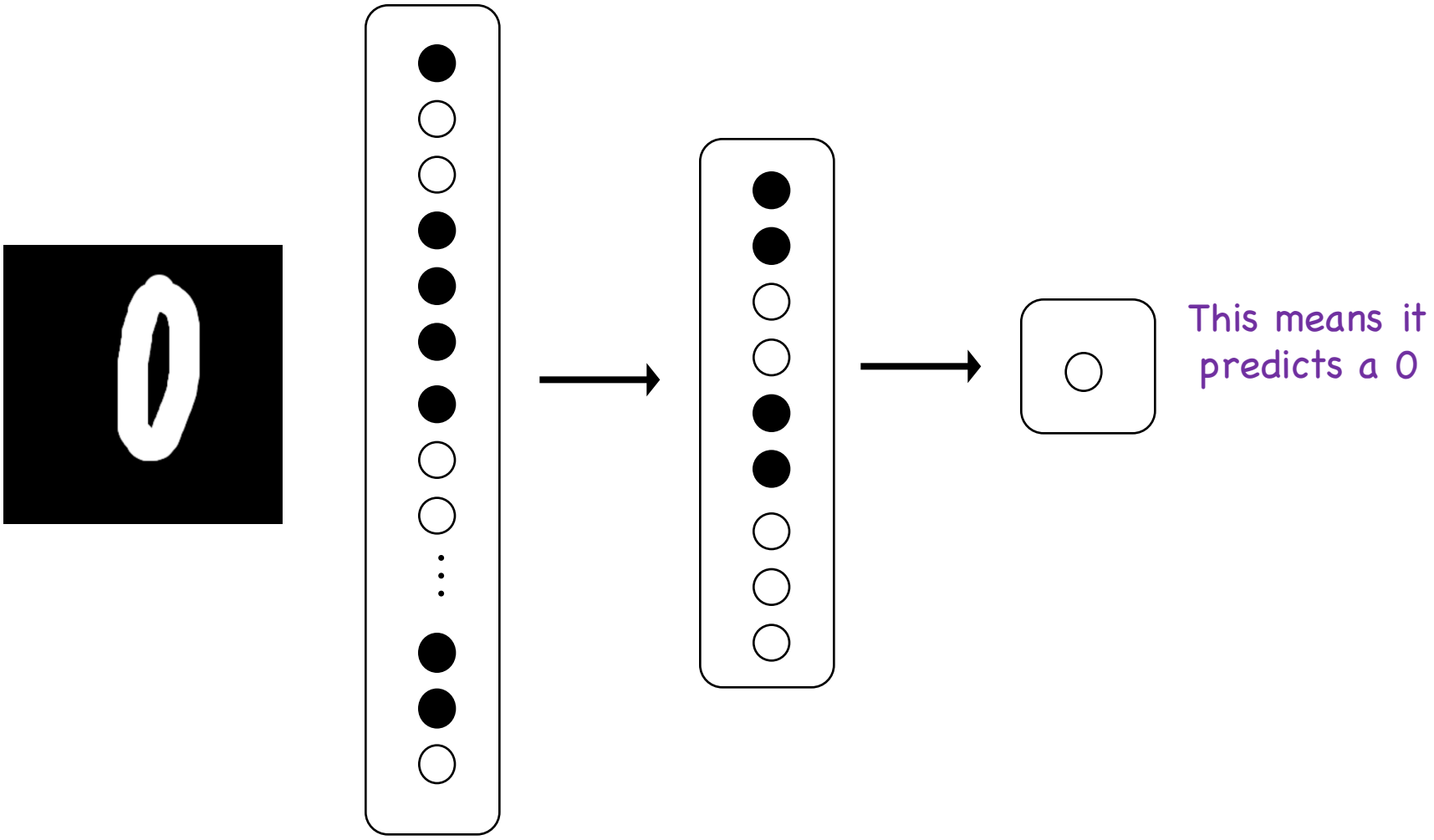


We Can Put Neurons Together



Look at another neuron

We Can Put Neurons Together

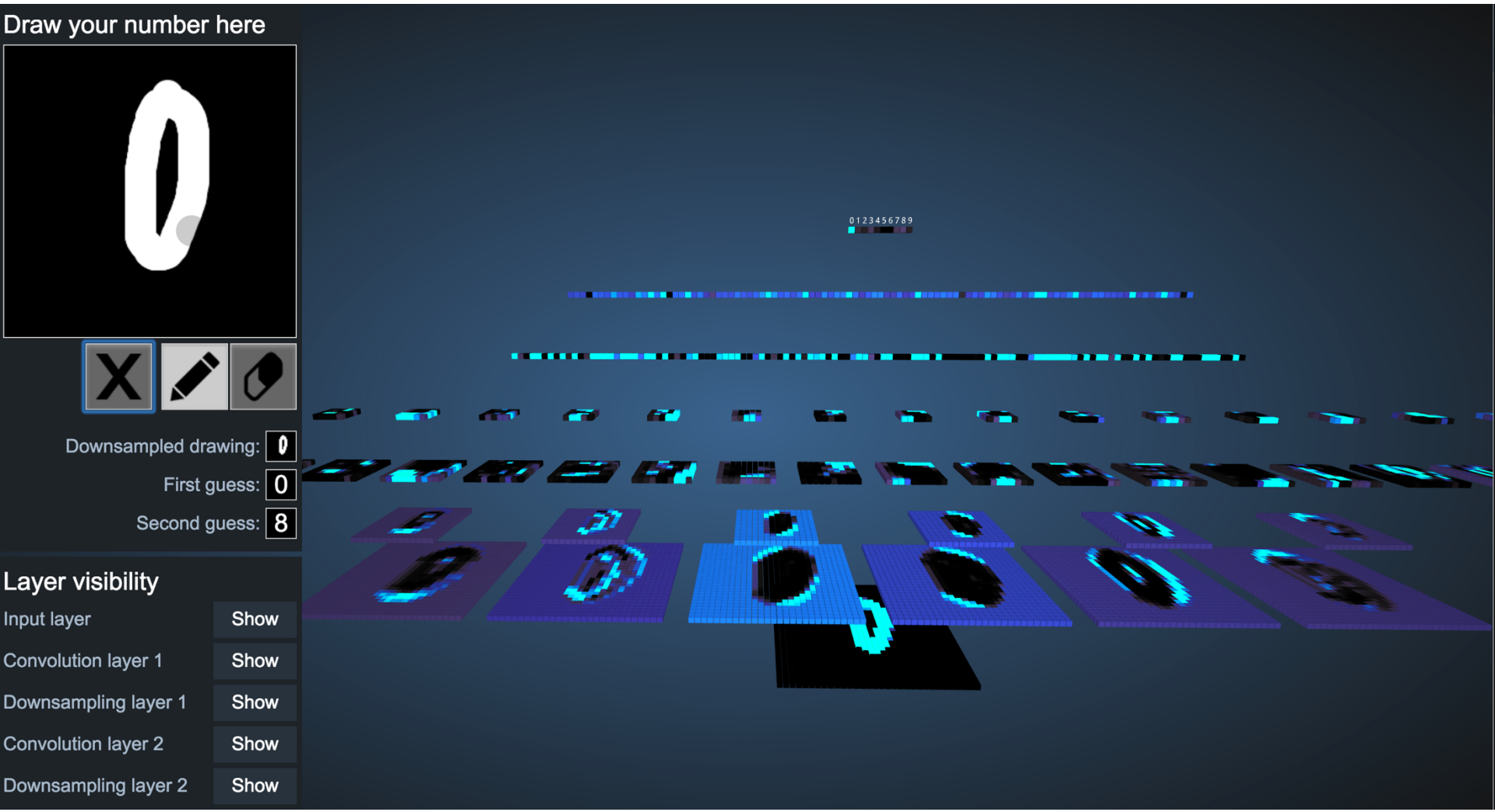


Deep Learning Assumption

- 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!

2 Min Pause

Summarize what we have learned

MLE of Thetas!

First: Learning Goals…

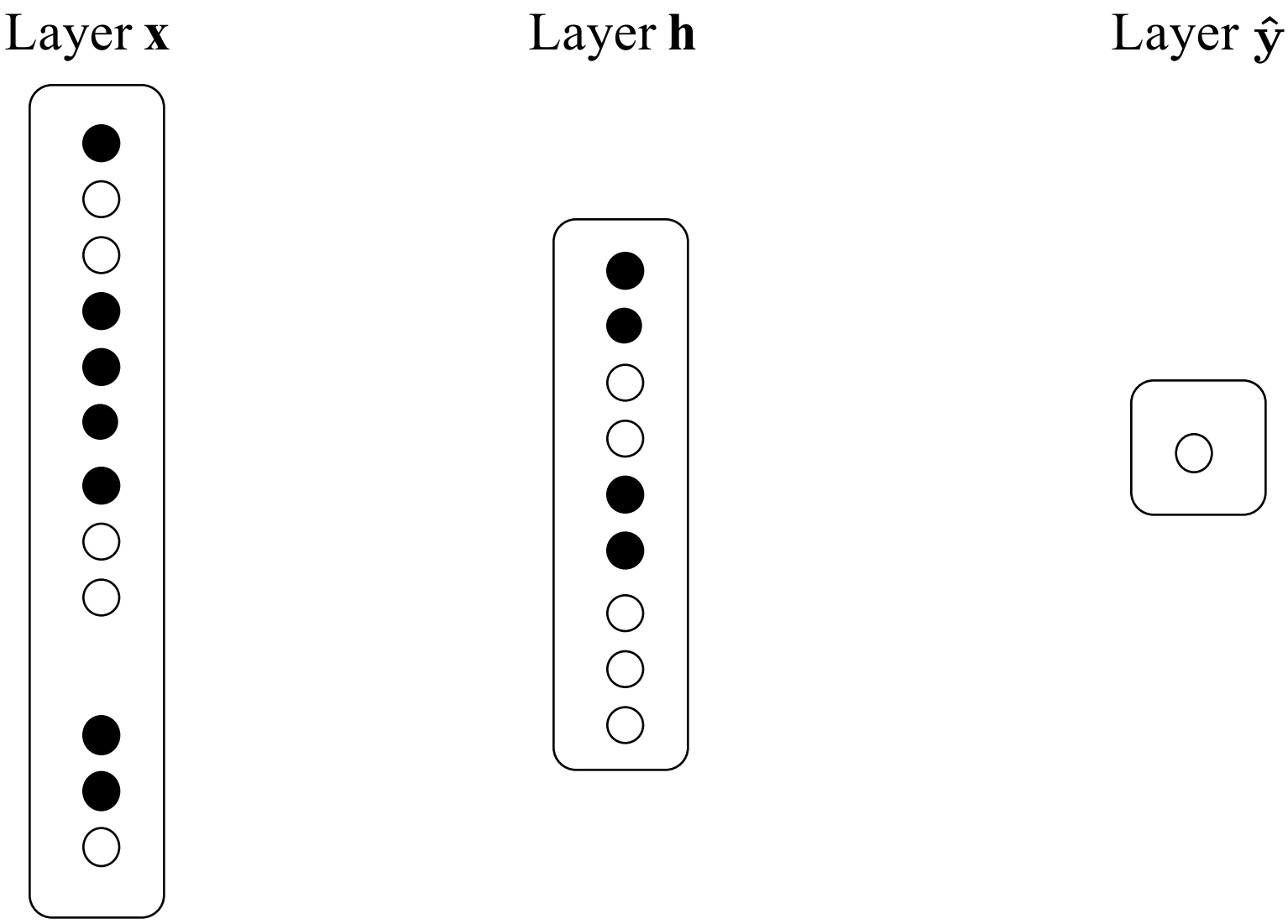
1. Understand Chain Rule as ♡ of Deep Learning

2. Everyone should be able to do simple derivations

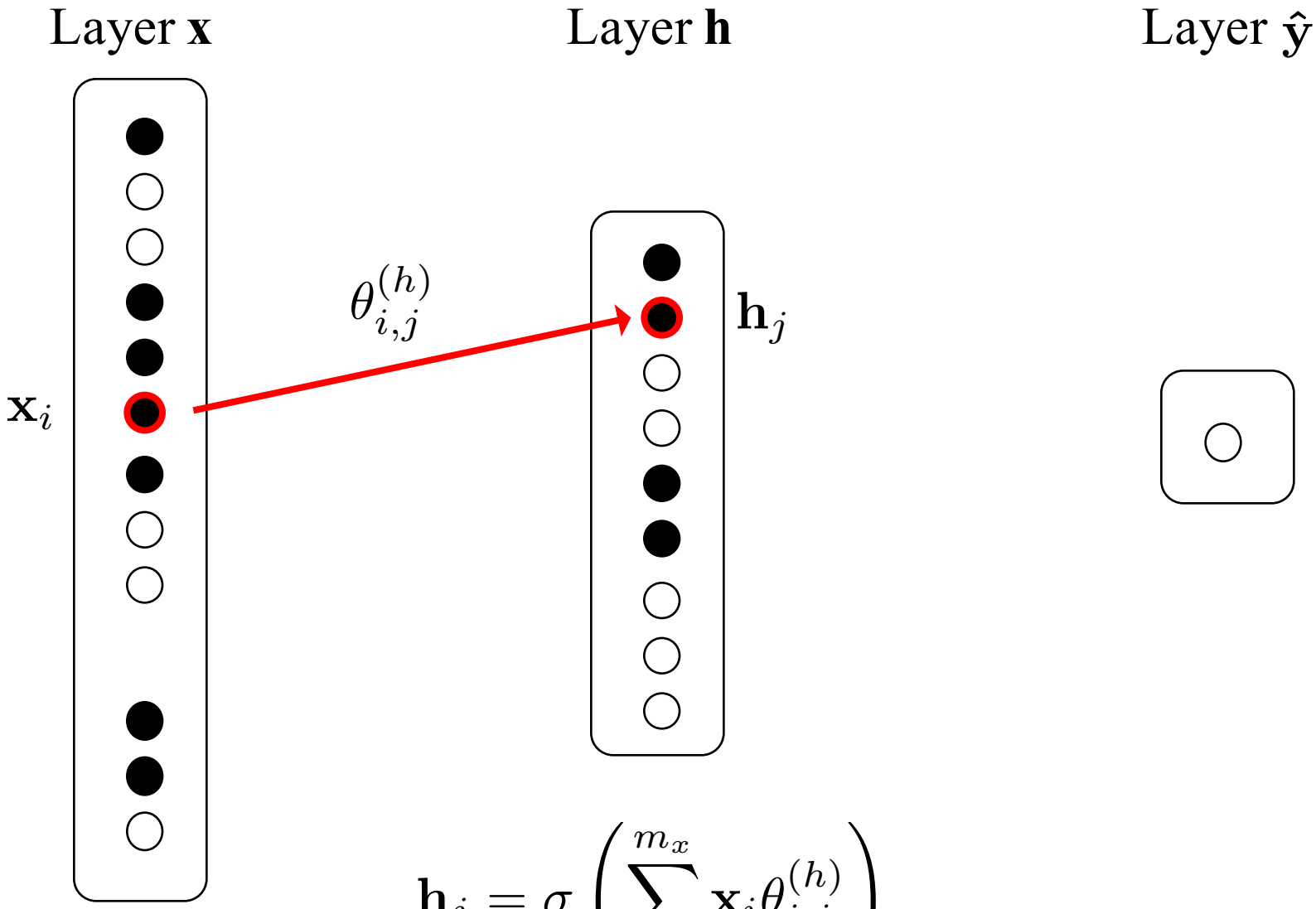
3. Be ready to rock
the socks off of CS221 and CS224N

Math worth knowing:

New Notation

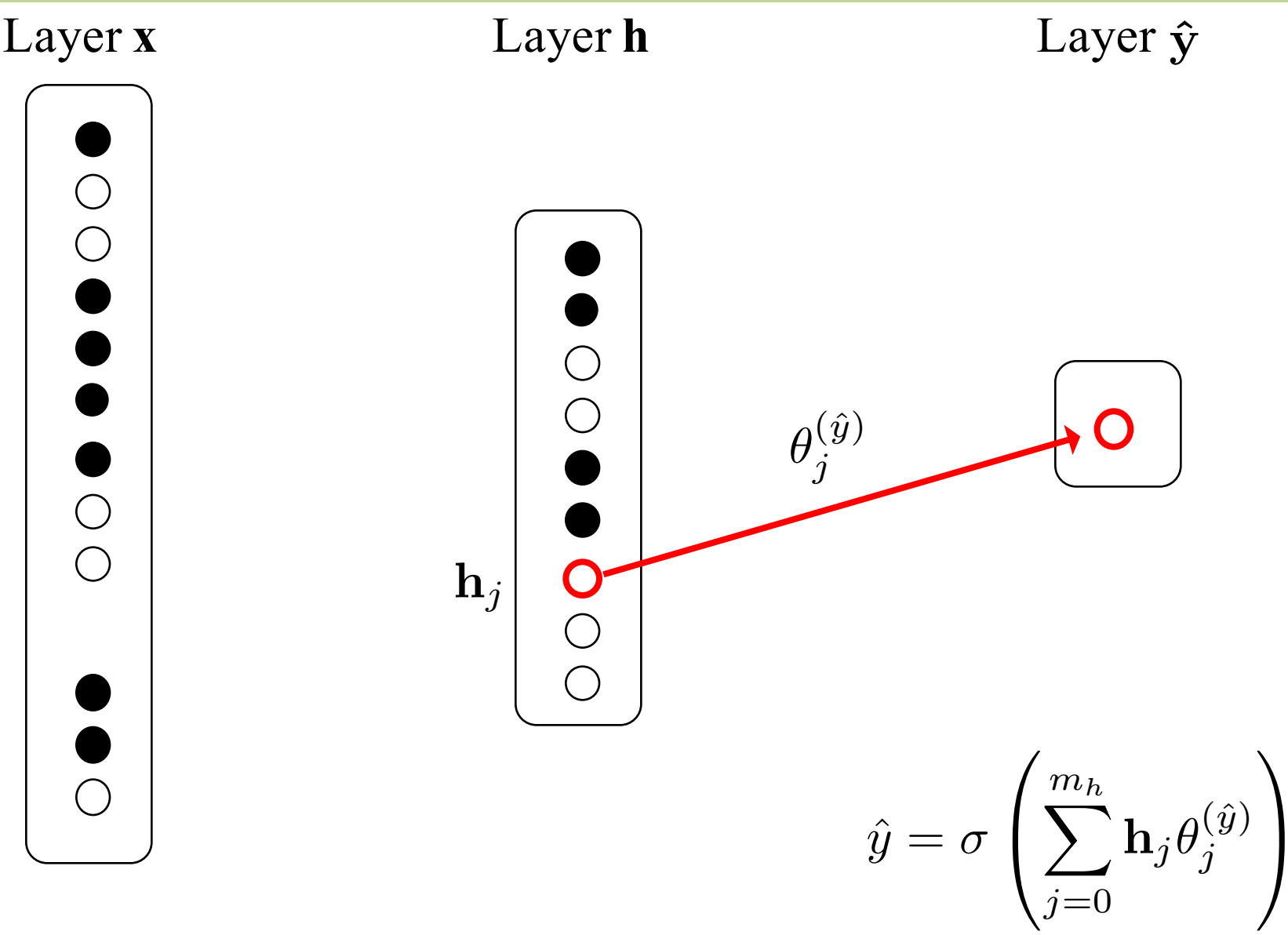


New Notation



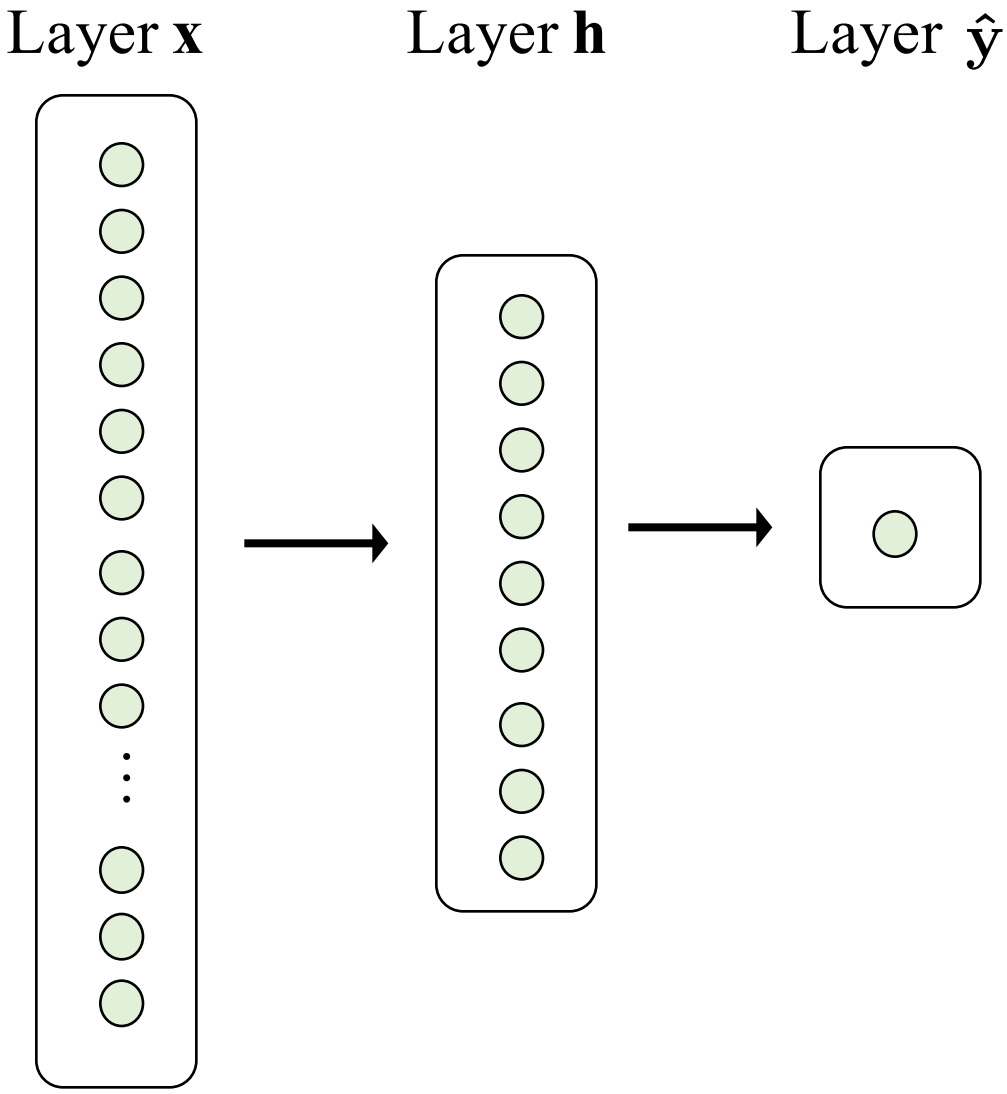
$$\mathbf{h}_j = \sigma\left(\sum_{i=0}^{m_x} \mathbf{x}_i \theta_{i,j}^{(h)}\right)$$

New Notation

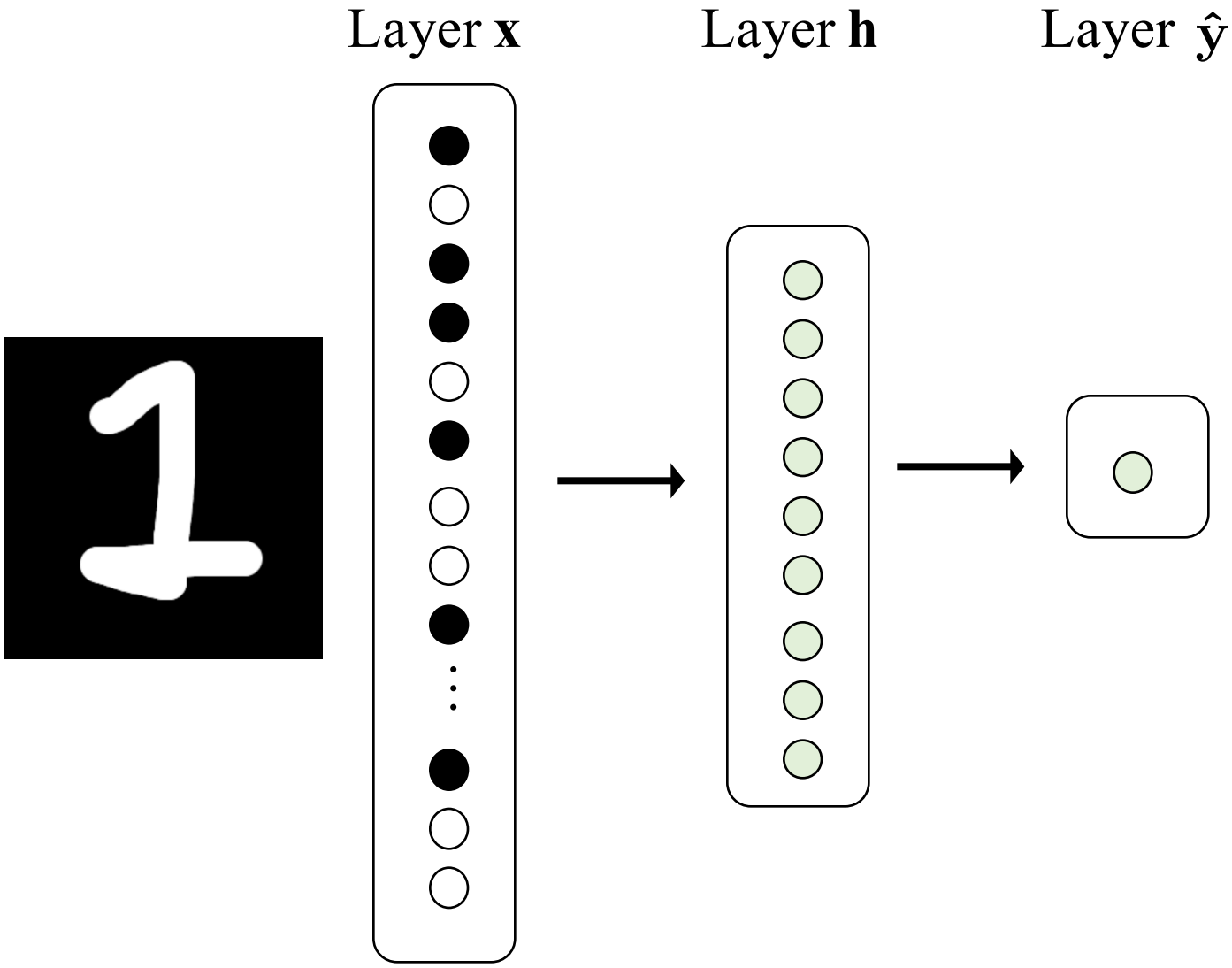


$$\hat{y} = \sigma\left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}\right)$$

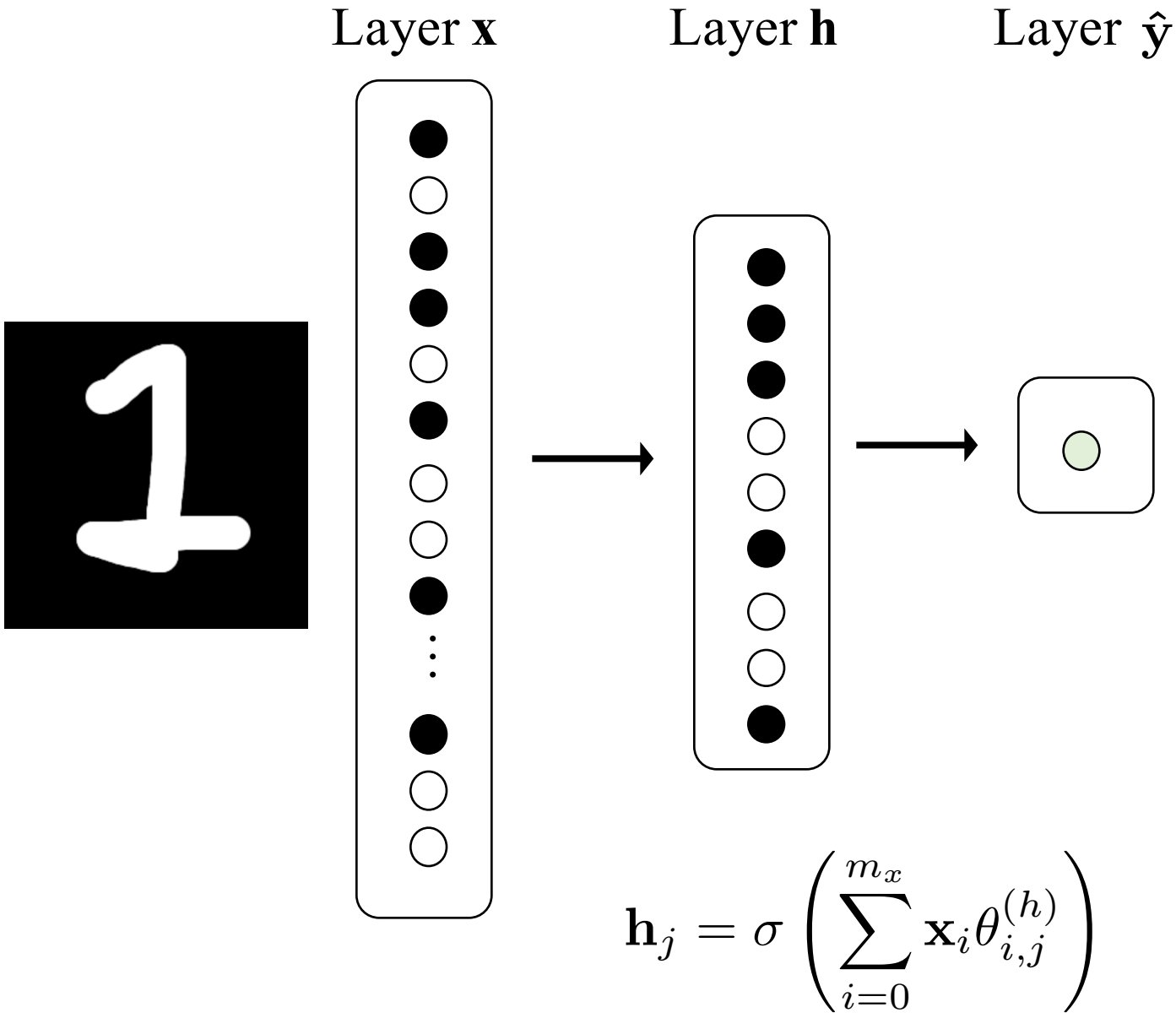
Forward Pass



Forward Pass

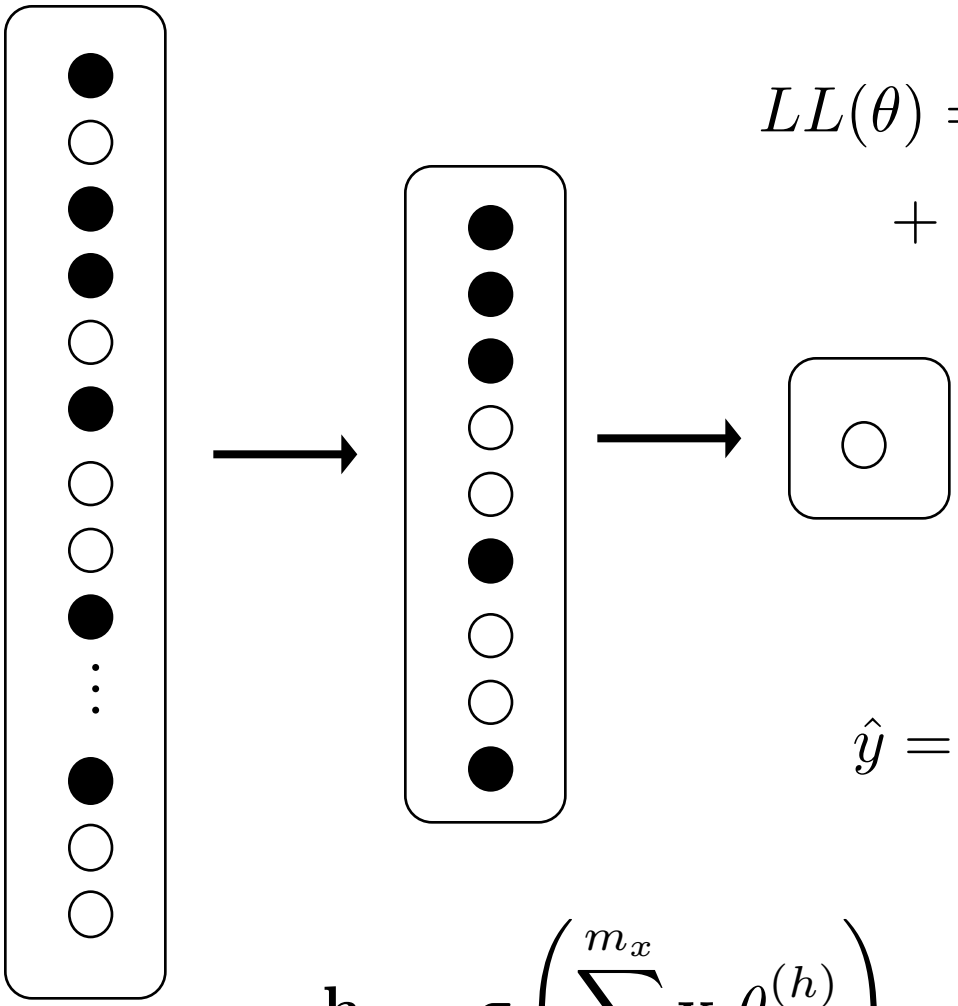


Forward Pass



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

Forward Pass

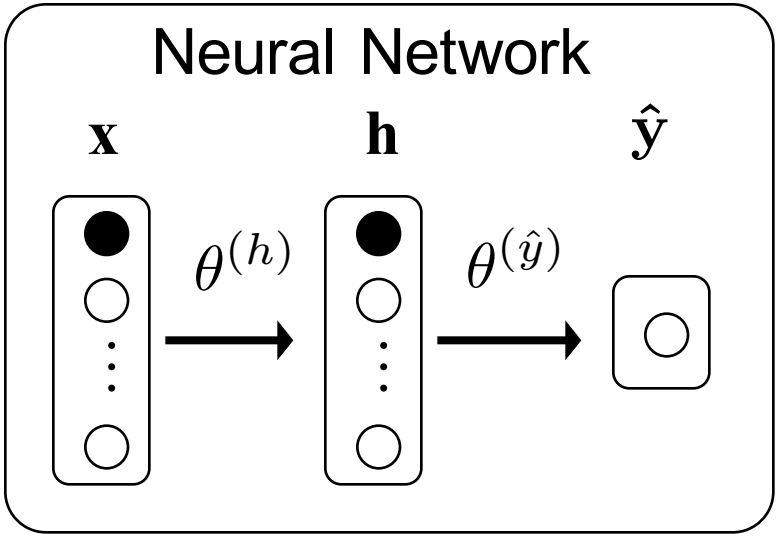


$$LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}]$$

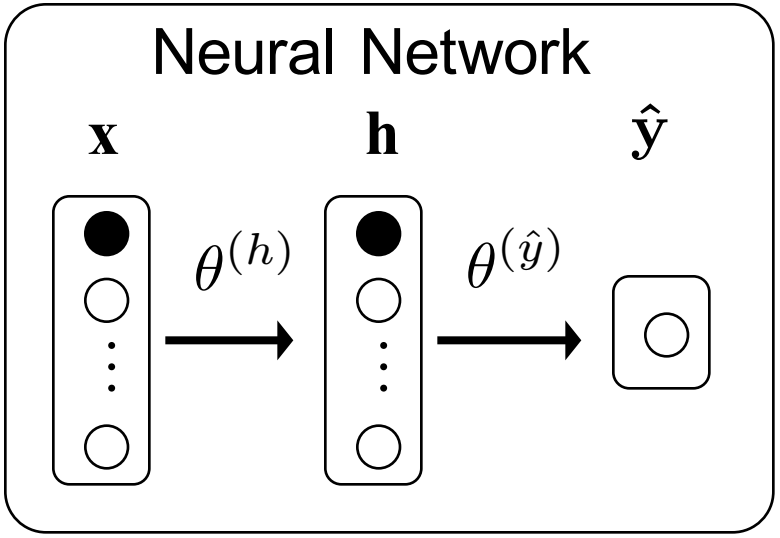
$$\hat{y} = \sigma\left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}\right)$$

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

All Together



Sanity Check



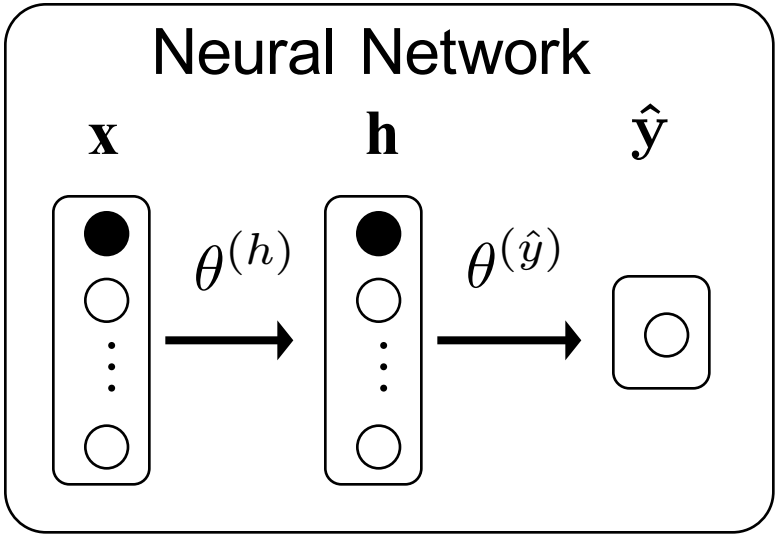
$$|\mathbf{x}| = 40$$

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

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?

Why We Calculate Partial Derivatives

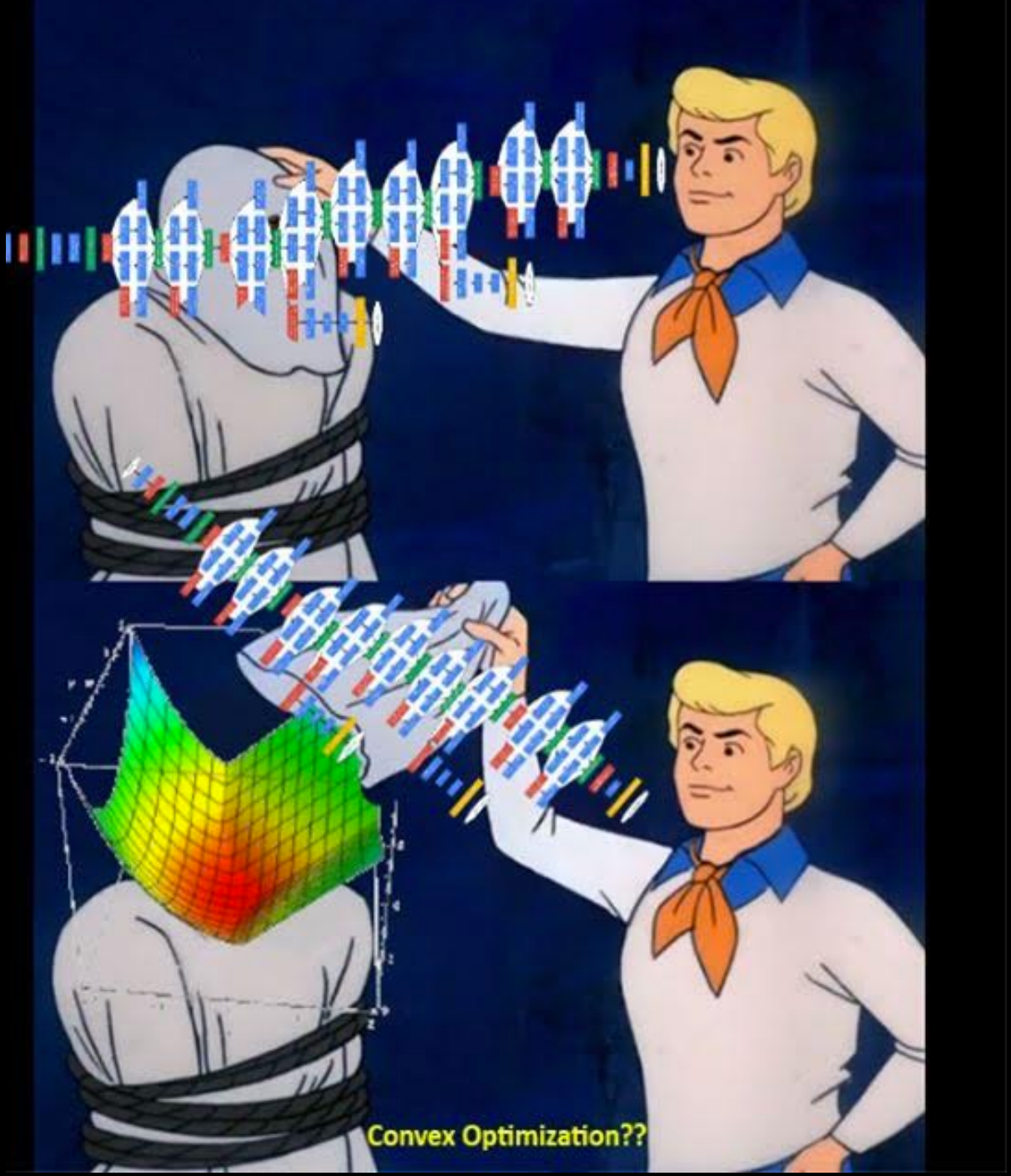
A deep learning model gets its **intelligence** by having **useful thetas**.

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.

Thanks to Keith Eicher


Convex Optimization??

Goal

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

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

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

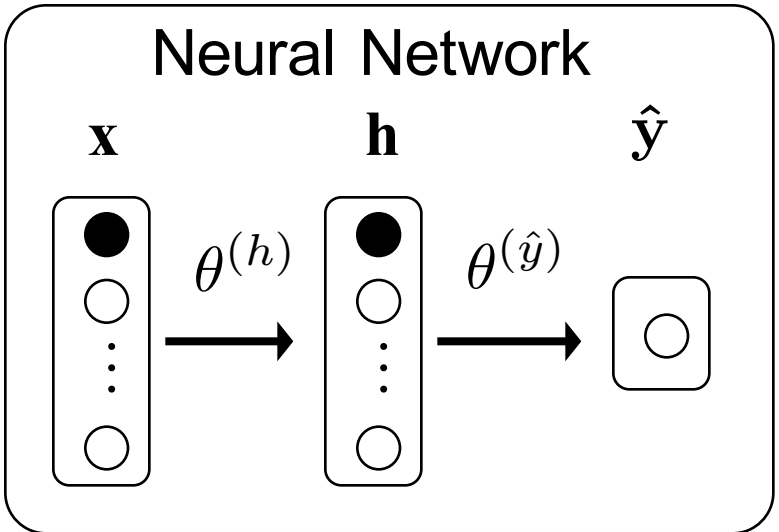
Derivative Goals

Loss with respect to output layer params

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

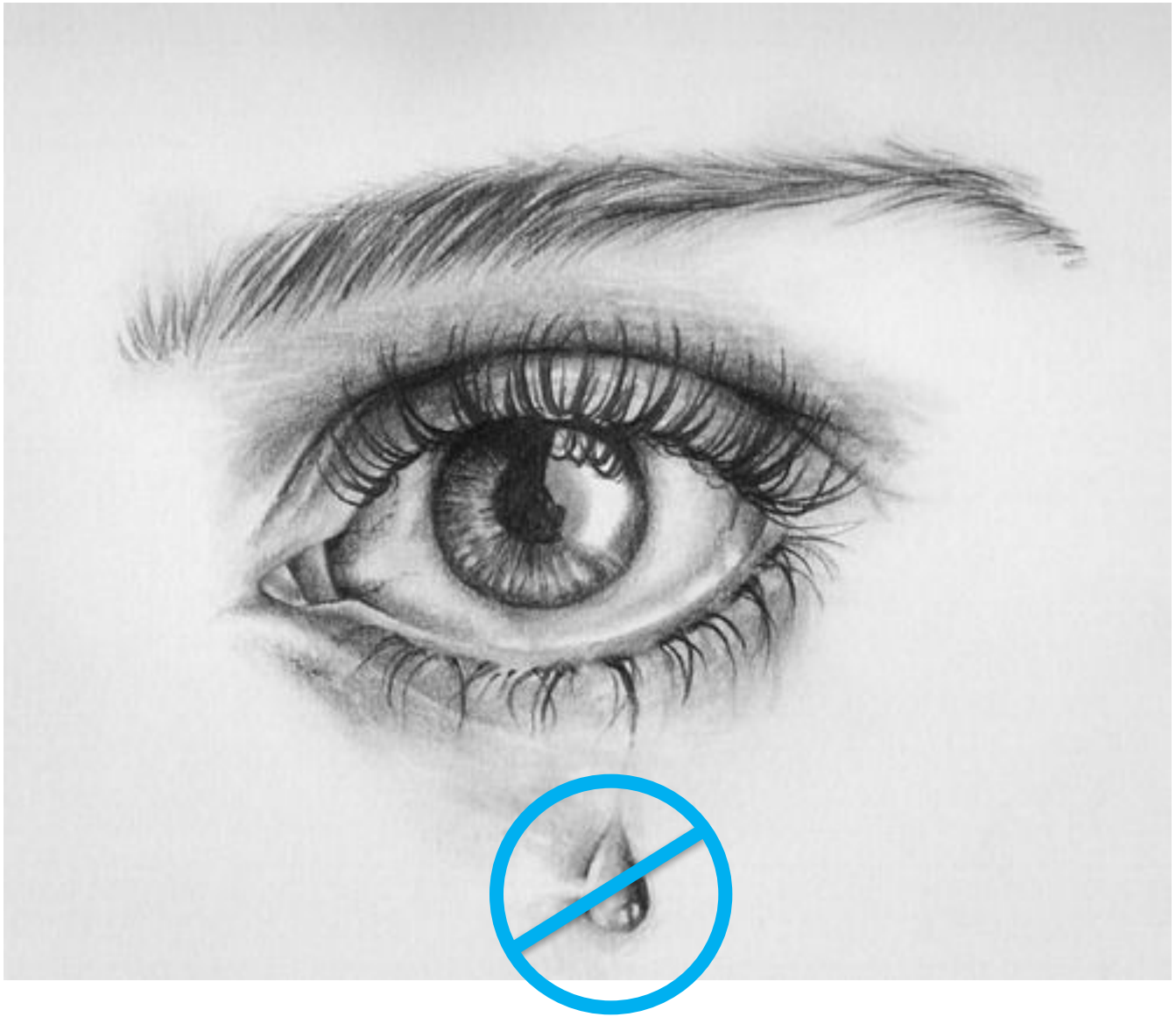
Loss with respect to hidden layer params

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



Derivatives without tears

Derivatives Without Tears



Think About Only One Training Instance

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

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

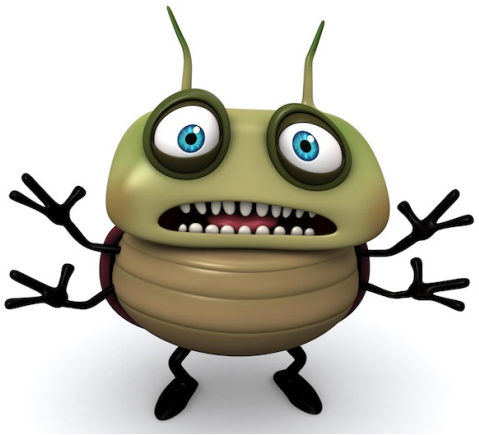
Bad Approach

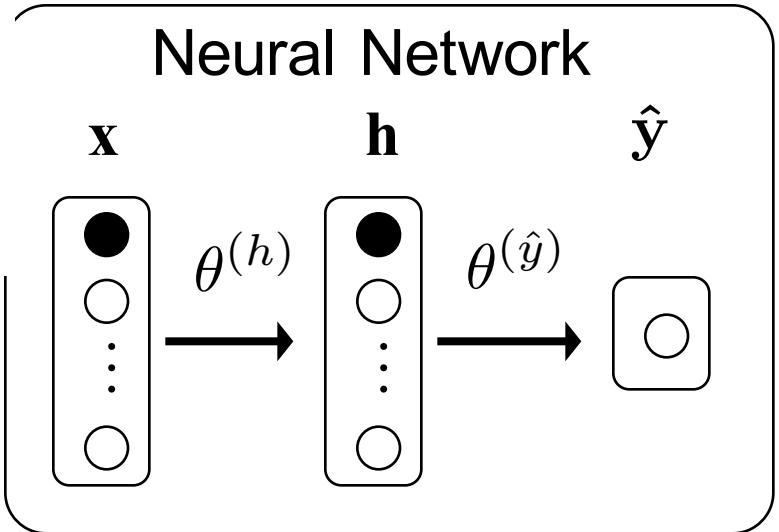
$$LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}]$$

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

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

It's called chain rule

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

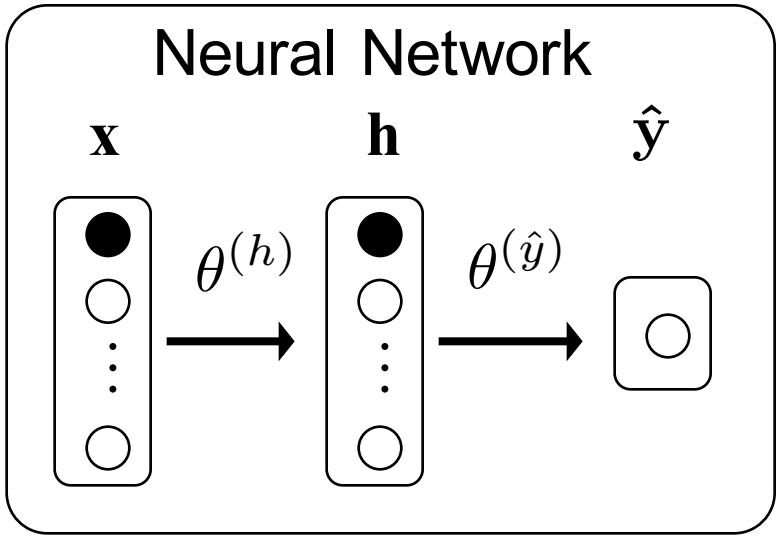
First use:

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

Chain Rule Example 1

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

Goal



Network

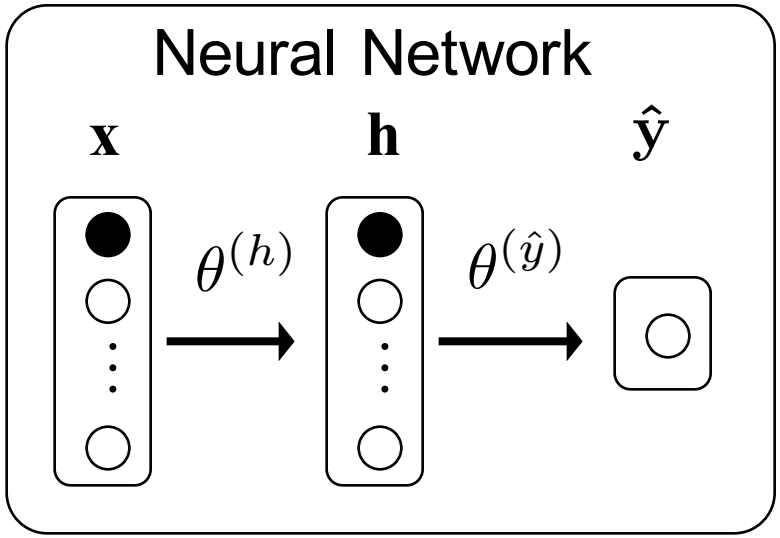
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}}$$

Decomposition

Chain Rule Example 2

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

Goal



Network

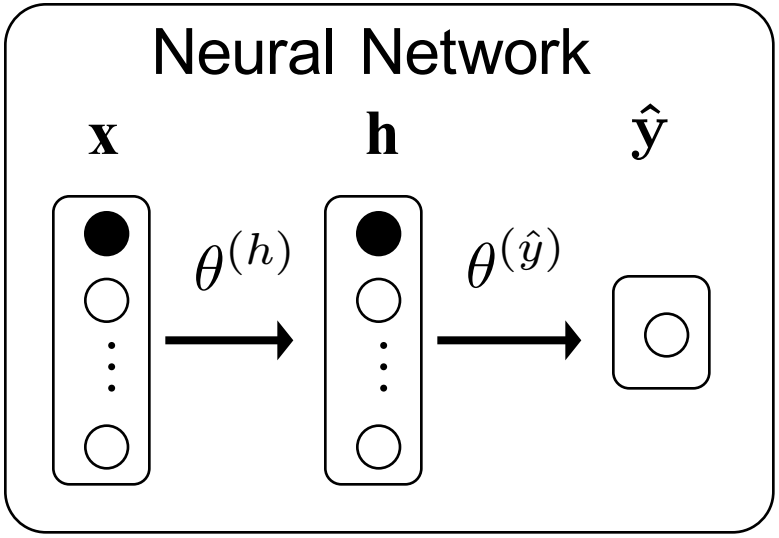
$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{h}_j} \cdot \frac{\partial \mathbf{h}_j}{\partial \theta_{i,j}^{(h)}}$$

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

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

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

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

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

Recall: Sigmoid has a Beautiful Slope

True fact about sigmoid functions

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

Sigmoid, you should be a ski hill

Gradient of output layer params

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

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

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

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

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

What! That's not scary!

Make it Simple

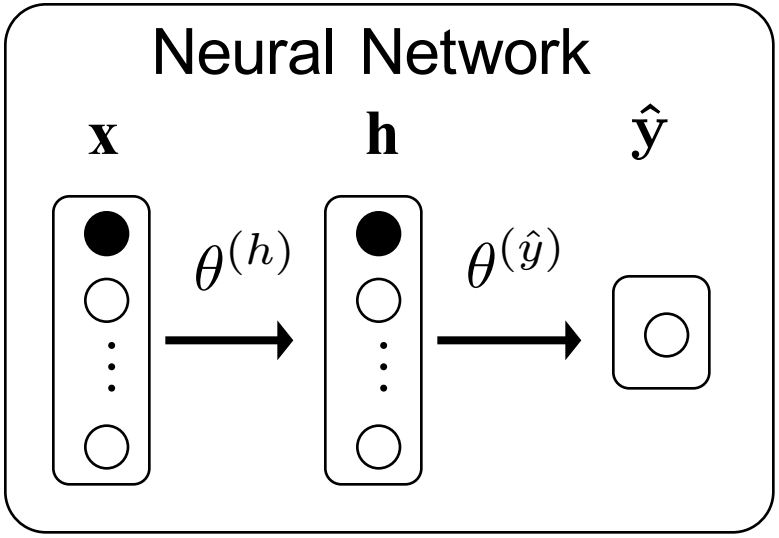
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \text{  } \cdot \text{  }$$

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

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

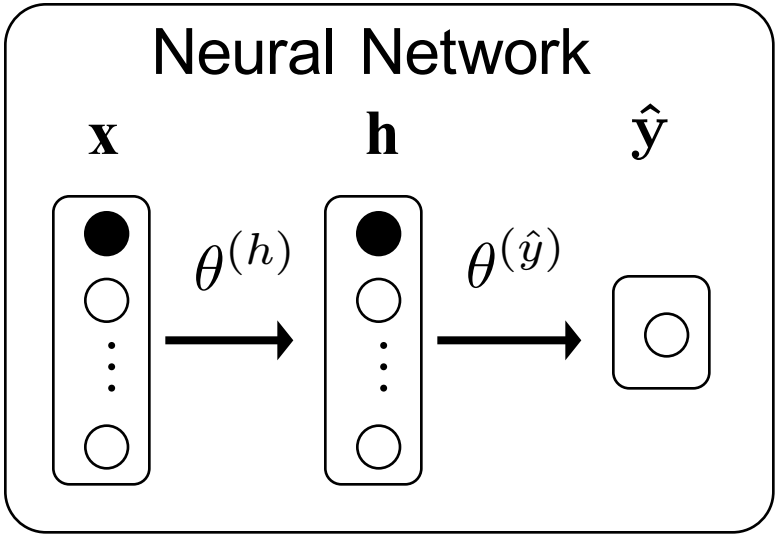
Boom!

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



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

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

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

Based on derivative of sigmoid

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

For all other values of i, the term in the sum is independent of h_j

Wait is it over?

Gradient of hidden layer params

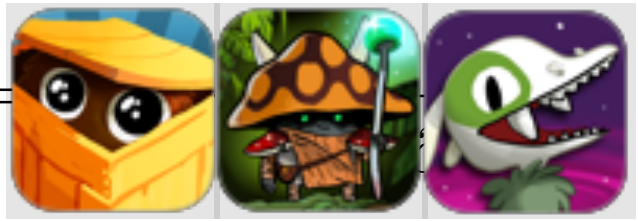
$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{h}_j} \cdot \frac{\partial \mathbf{h}_j}{\partial \theta_{i,j}^{(h)}}$$

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

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

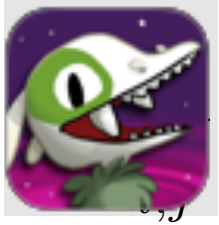
That one too?

Make it Simple

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

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

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



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

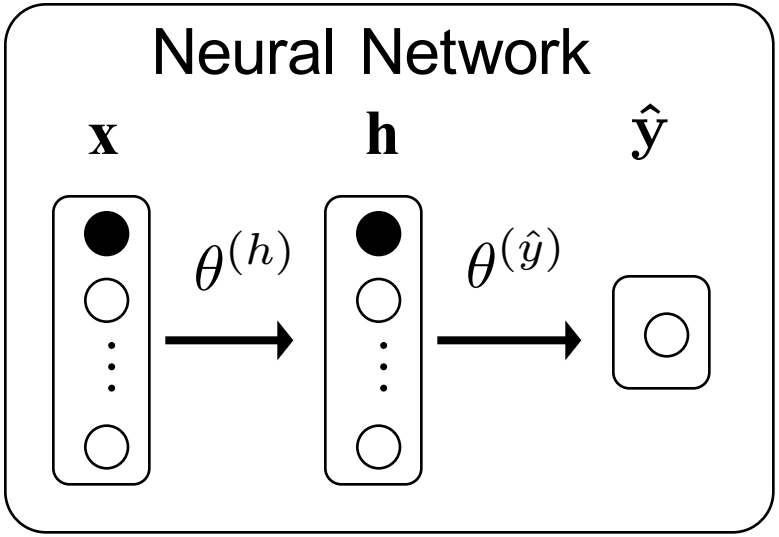
Summary: Simple Calculations For

Loss with respect to output layer params

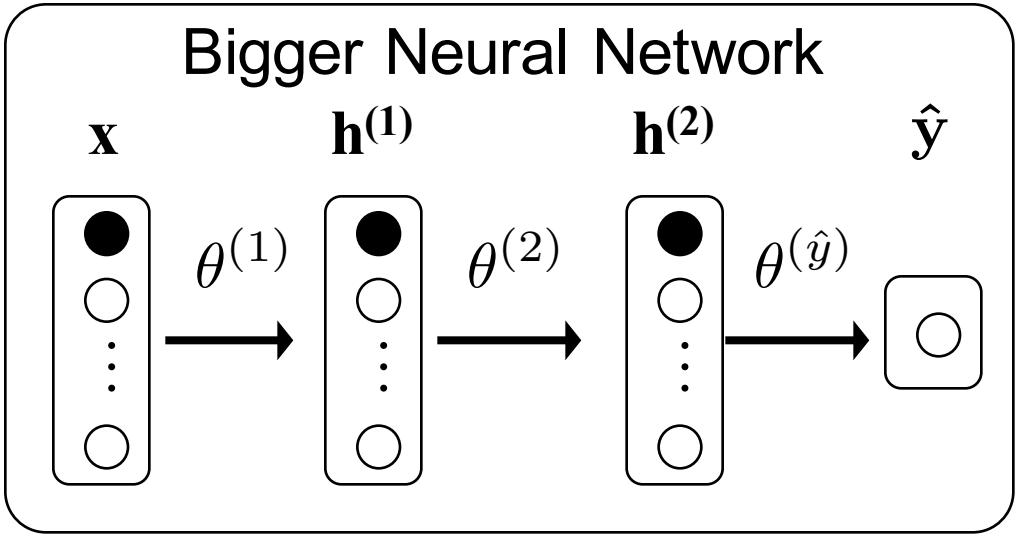
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}}$$

Loss with respect to hidden layer params

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



What Would You Do Here?

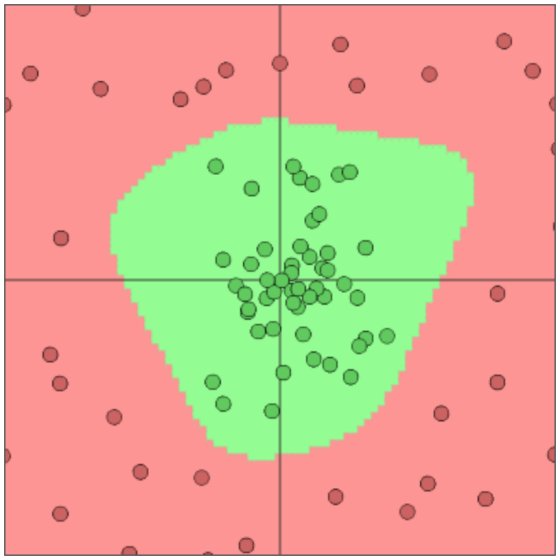


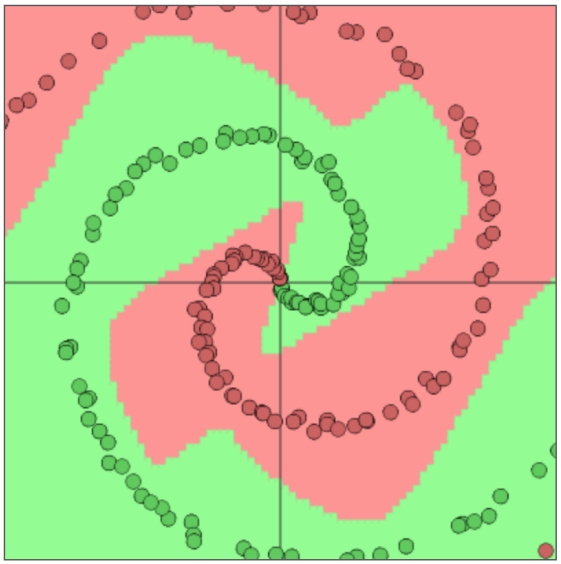
Chain rule:
Game changer for
artificial intelligence

Congrats. You now know backpropagation

Neural Networks Can Learn Complex Functions

- Some data sets/functions are not separable

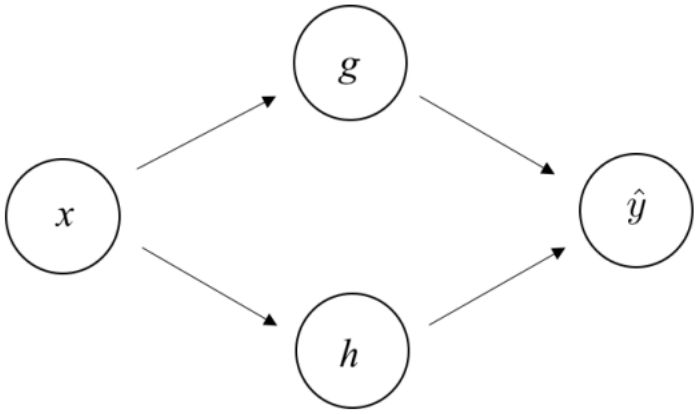




  - These are classifiers learned by neural networks

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

What If You Had a Neural Network Like This?



$$g = \text{sigmoid}(\theta_1 \cdot x)$$
$$h = \text{sigmoid}(\theta_2 \cdot x)$$
$$\hat{y} = \text{sigmoid}(\theta_3 \cdot g + \theta_4 \cdot h)$$
$$LL(\theta) = \sum_{i=1}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

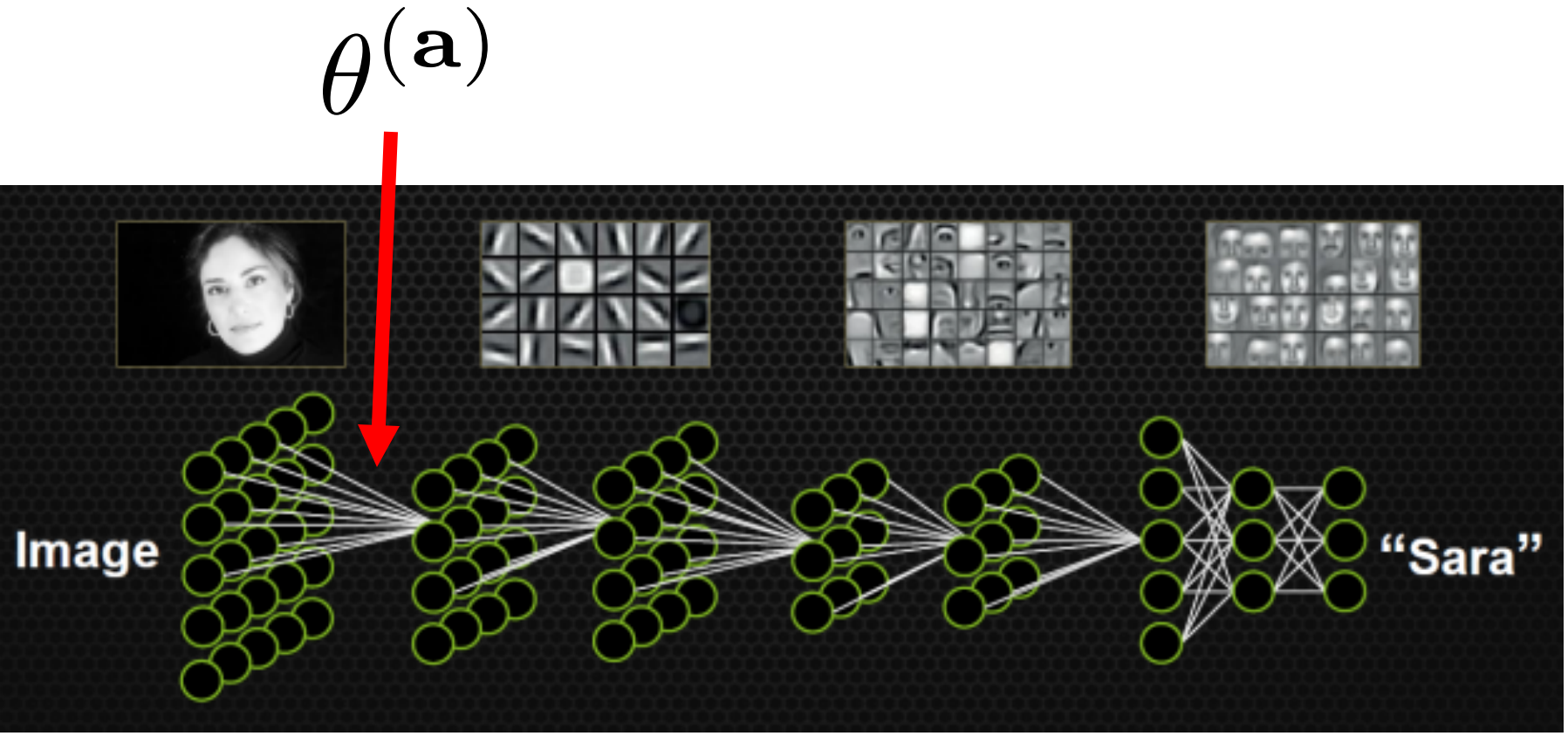
1. Calculate partial derivative for one data instance
2. Use chain rule!
3. Sigmoid derivatives come out simple if you use the right notation
4. You don't need to give the most reduced answer

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.

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

Works for any number of layers



x a b c d e f g $\hat{y}$ LL

model.update(data)

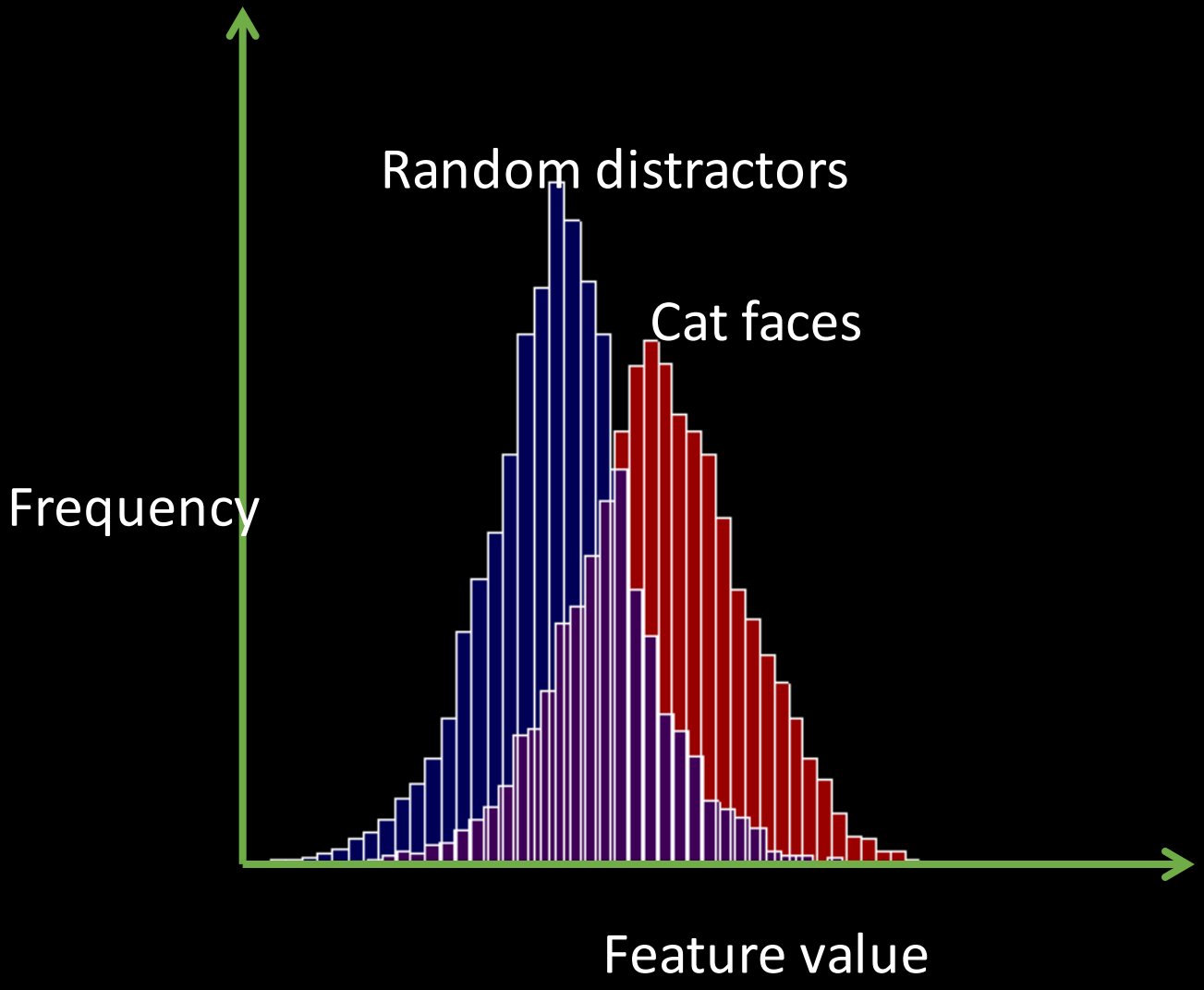
The Cat Neuron

Top stimuli from the test set

Optimal stimulus
by numerical optimization

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

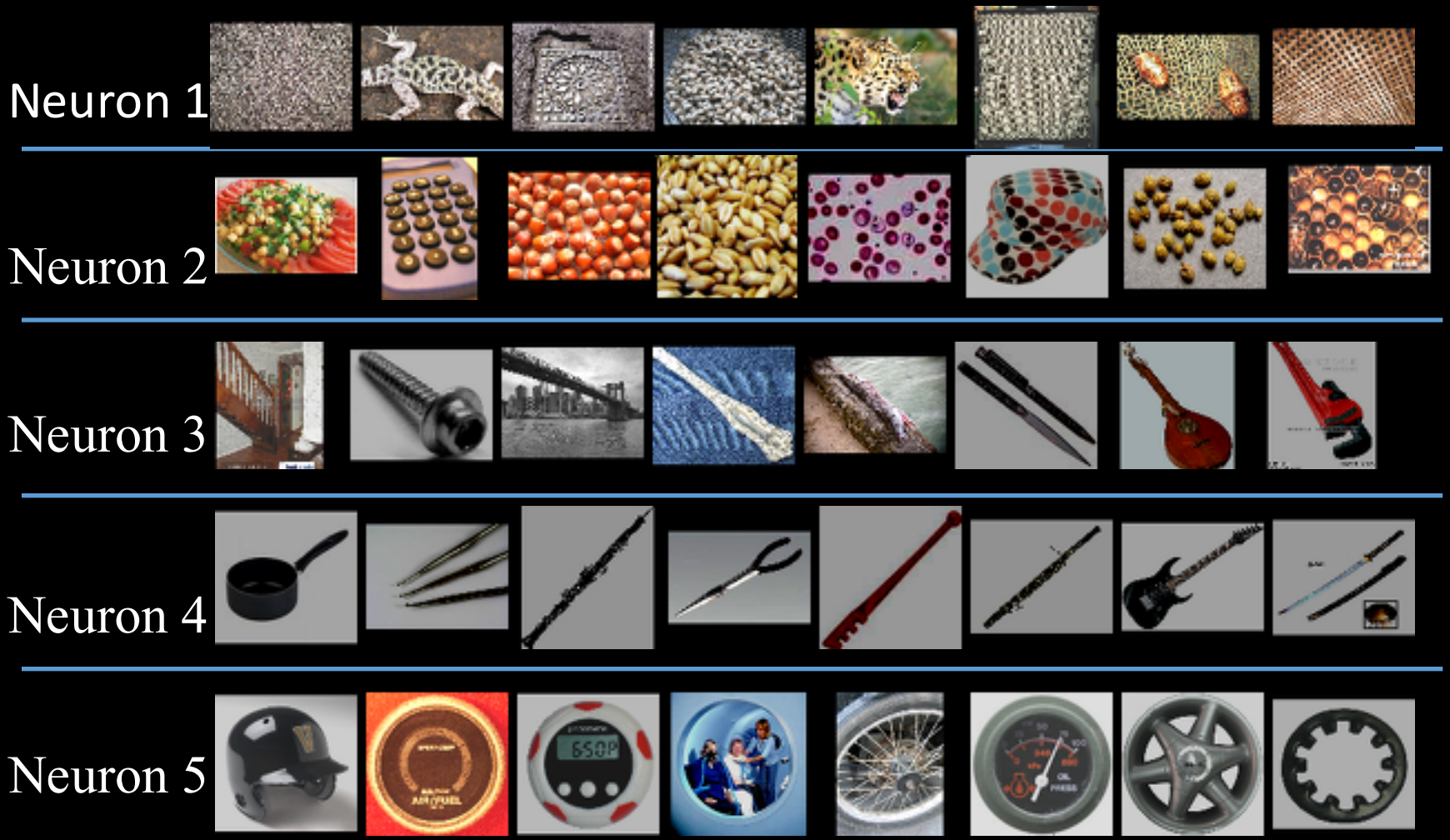
The Cat Neuron



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*

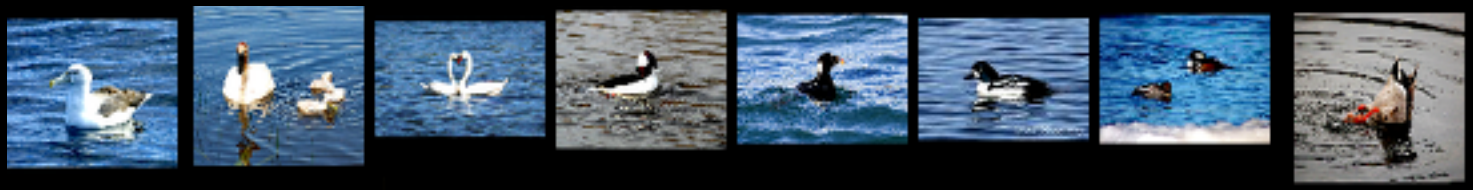
Best Neuron Stimuli

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*

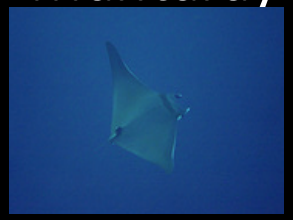
22,000 is a lot!

…
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%

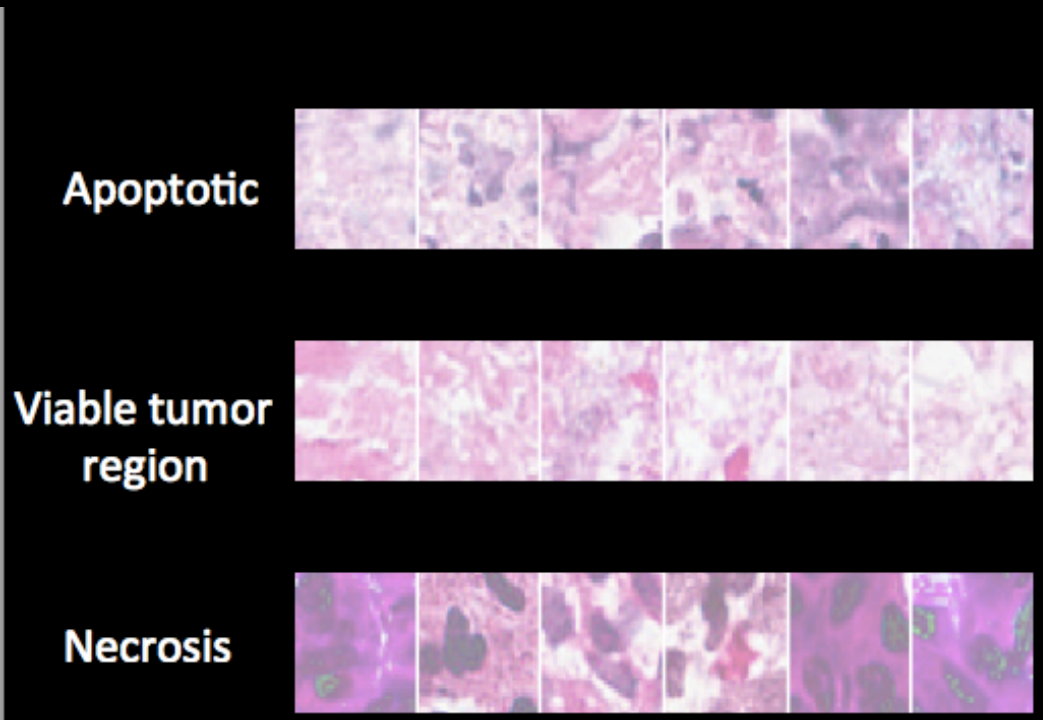
Pre Neural Networks

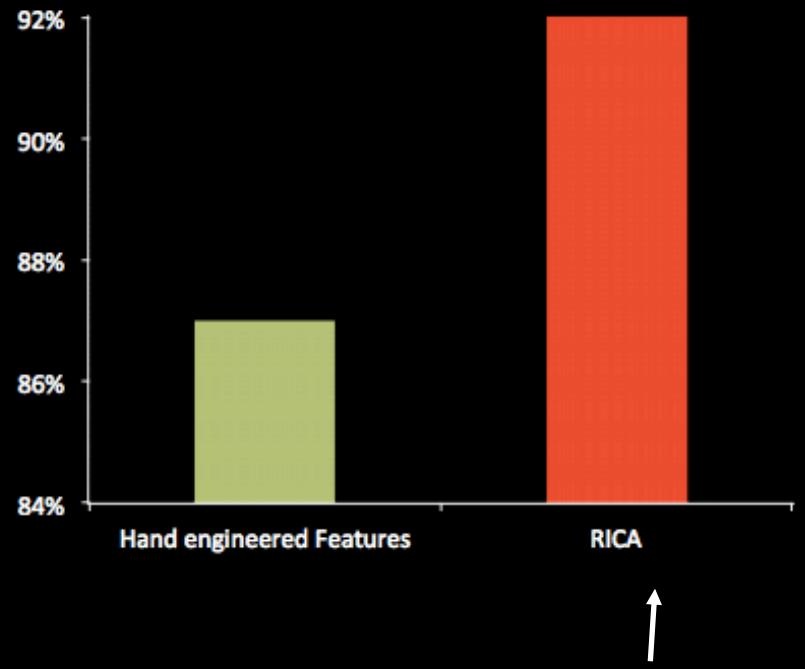
43.9%

GoogLeNet

Szegedy et al, Going Deeper With Convolutions, CVPR 2015

Vision has Social Implications

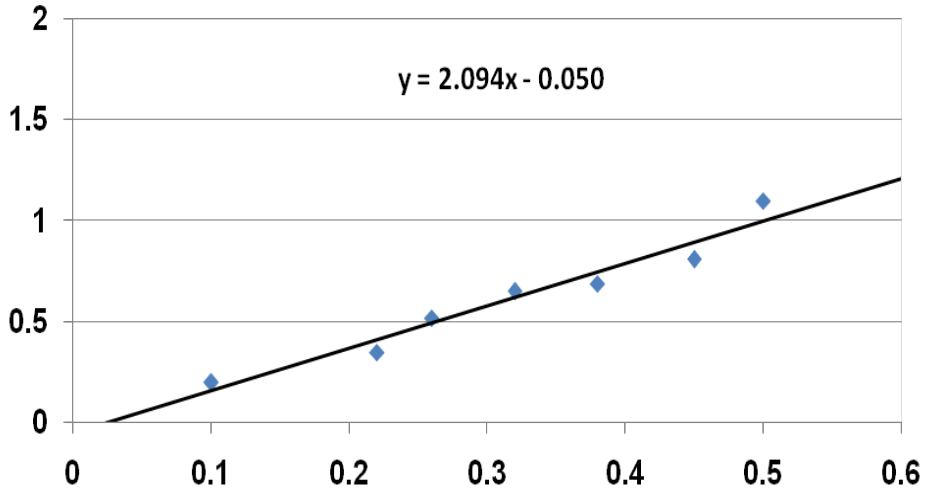


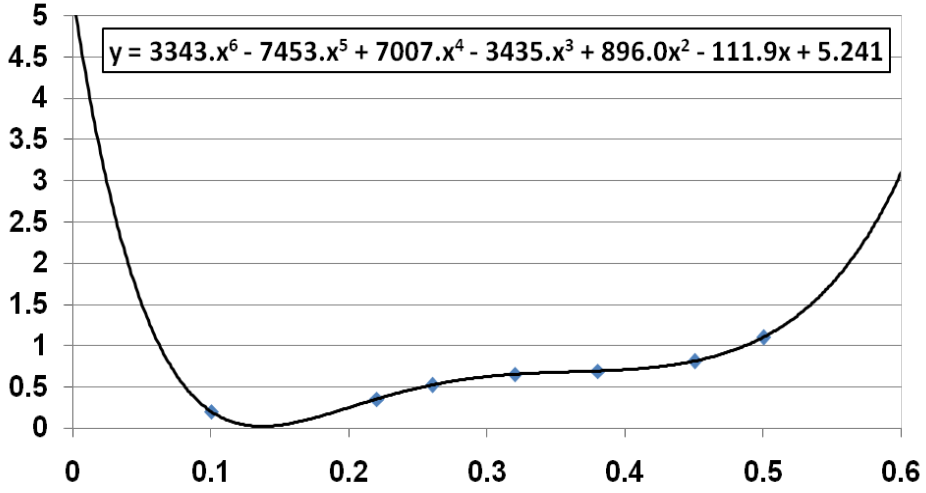


Neural network

Good ML = Generalization

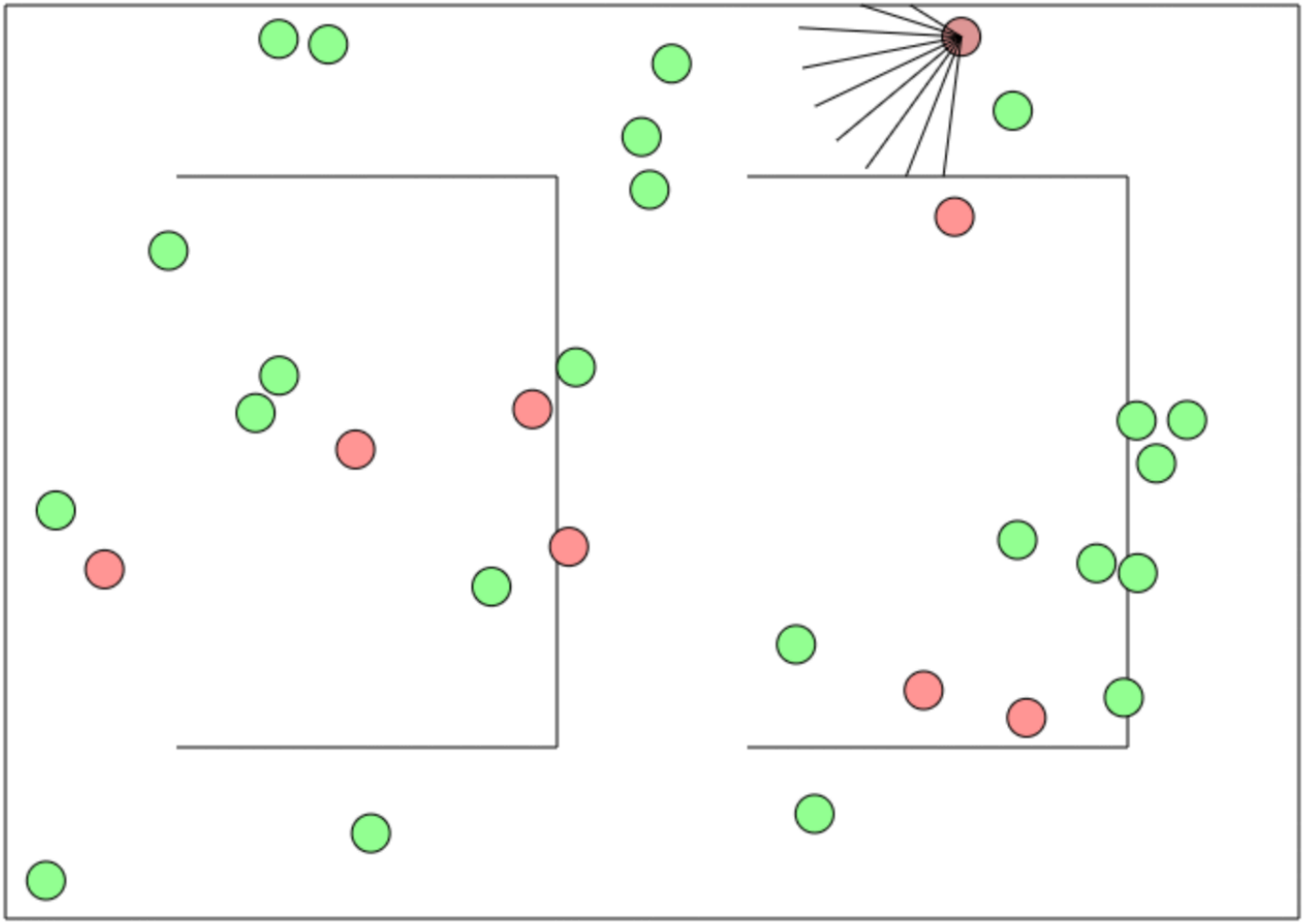
- Goal of machine learning: build models that ***generalize*** well to predicting new data
  - "Overfitting": fitting the training data too well, so we lose generality of model





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

Deep Reinforcement Learning

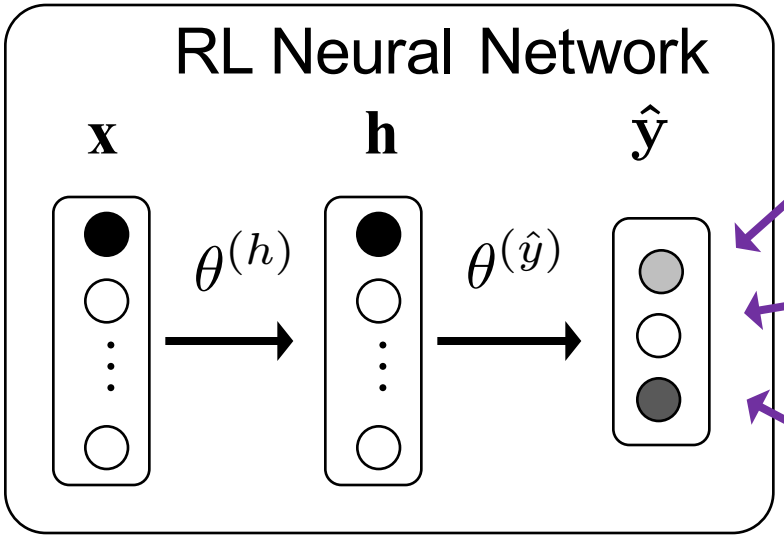


http://cs.stanford.edu/people/karpathy/convnetjs/demo/rldemo.html

Deep Reinforcement Learning

R is a reward and A_i is a legal action

Input is a representation of current state (S)



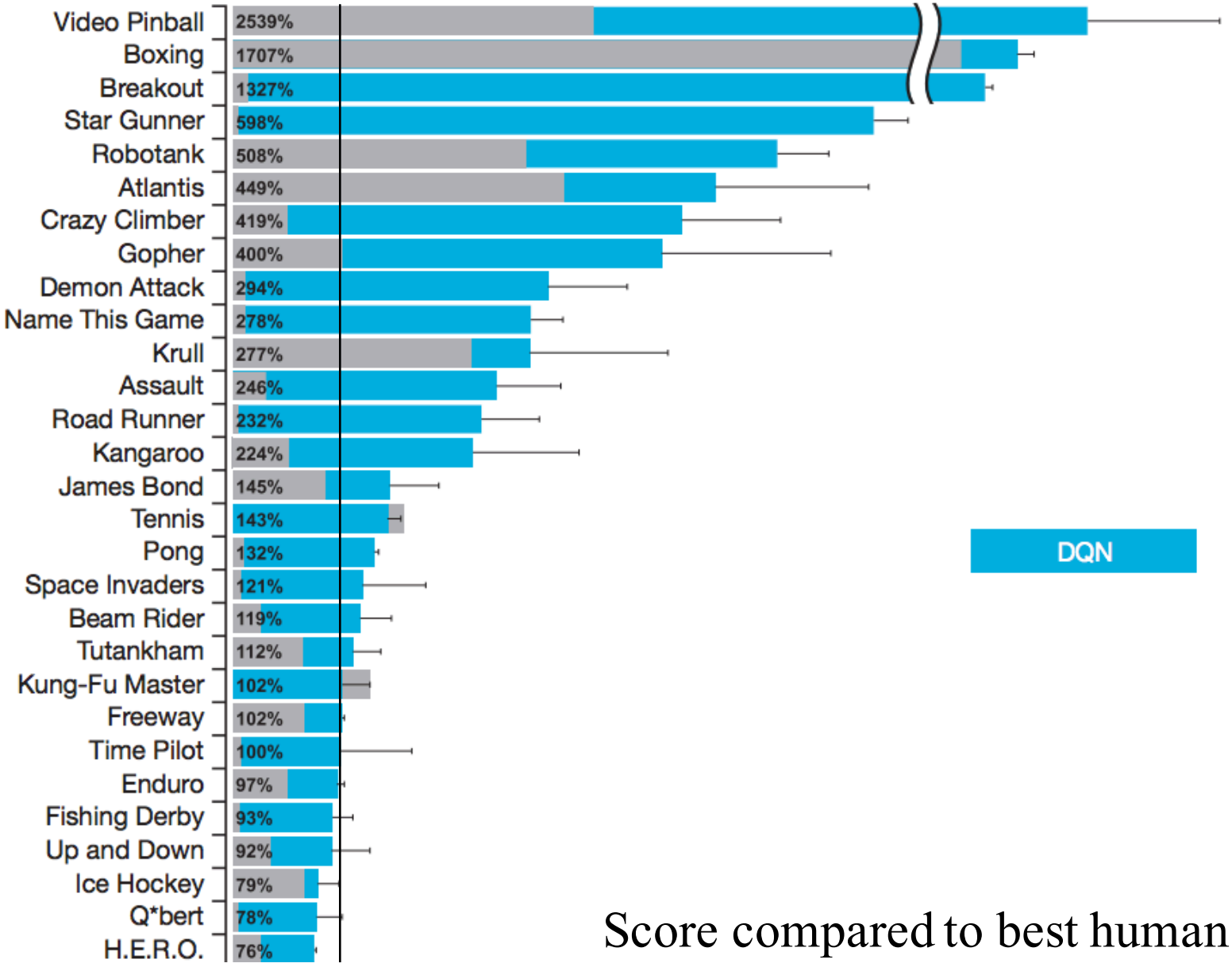
E[R | A_1, S]

E[R | A_2, S]

E[R | A_3, S]

Interpret outputs as expected reward for a given action

Deep Mind Atari Games



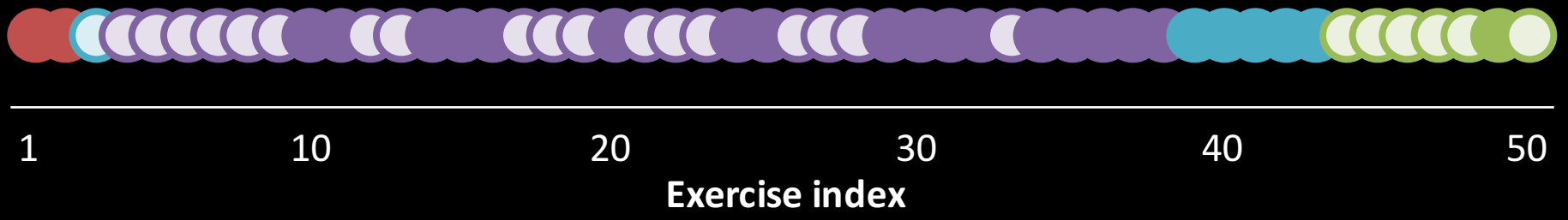
Score compared to best human

Alpha GO

When Do I Get a Robo Tutor?

Piech

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:

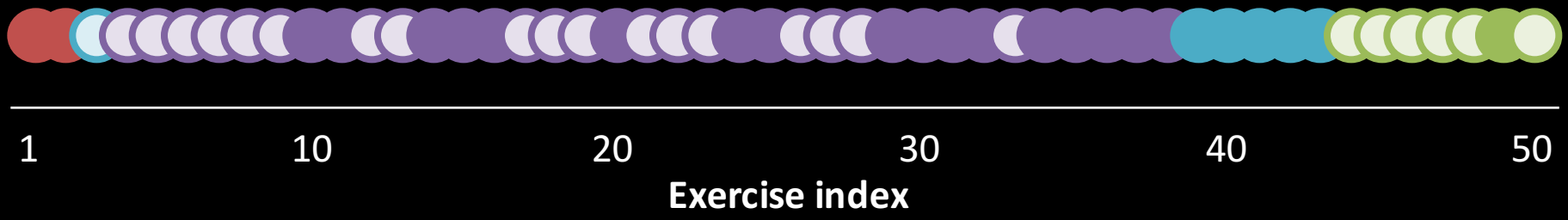
- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

Answer:

- Correct
- Incorrect

KHAN ACADEMY

Story of Riley



What does Riley know?

Exercise Type:

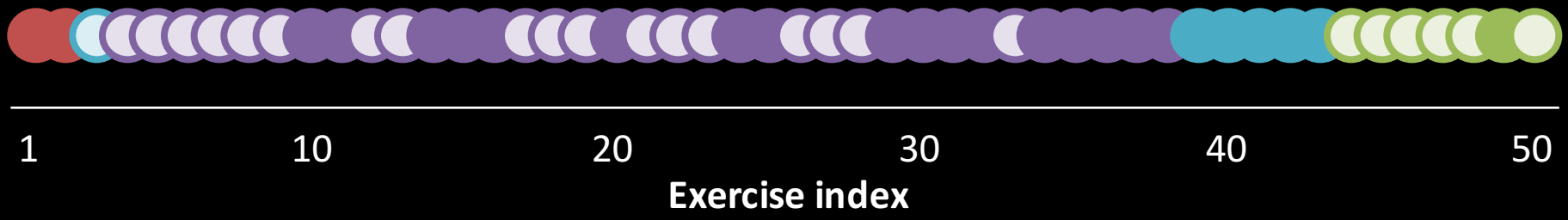
- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

Answer:

- Correct
- Incorrect

KHAN ACADEMY

Story of Riley



What should Riley do next?

Exercise Type:

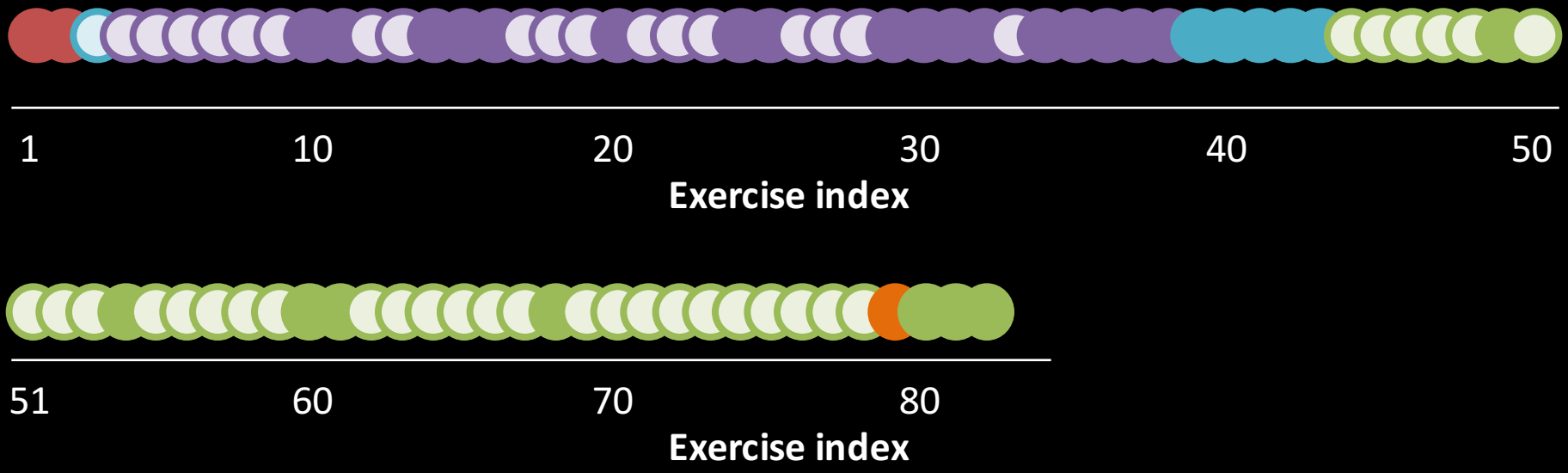
- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

Answer:

- Correct
- Incorrect

KHAN ACADEMY

Story of Riley



Exercise Type:

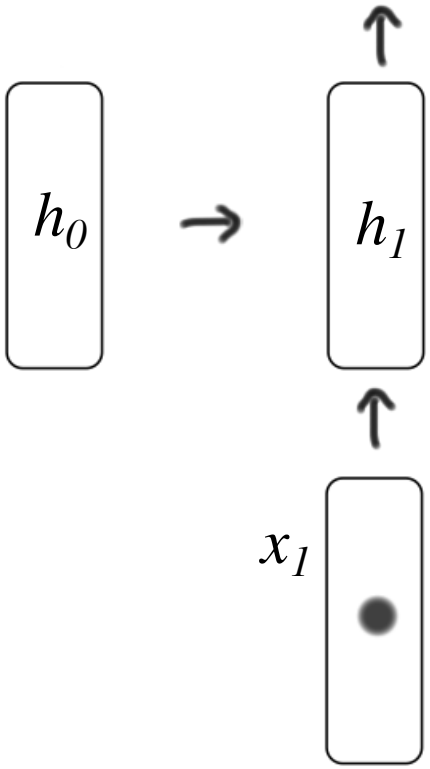
- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

Answer:

- Correct
- Incorrect

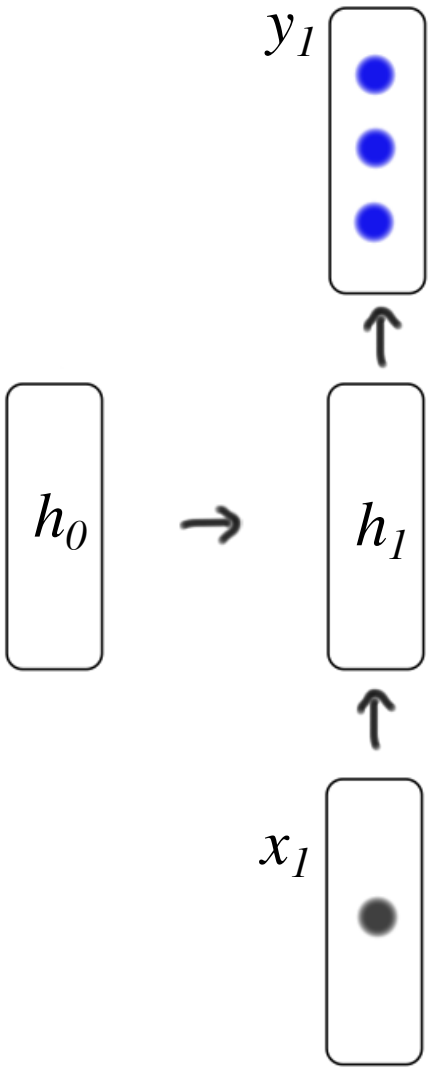
KHAN ACADEMY

Recurrent Neural Network



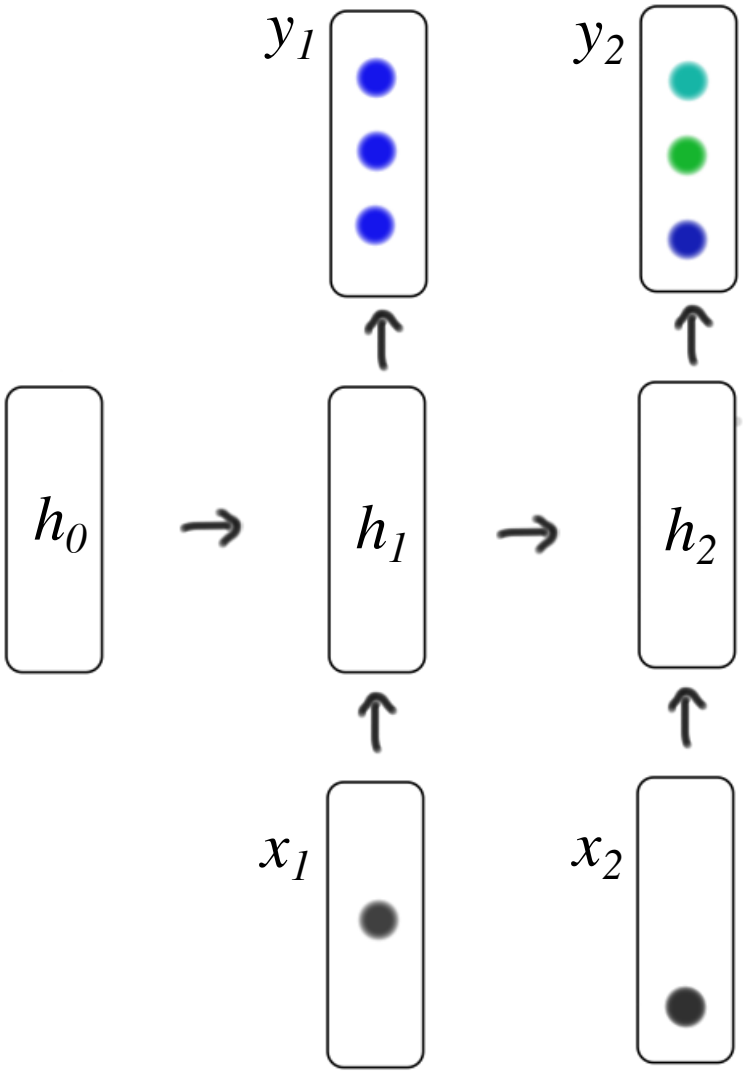
Piech

Recurrent Neural Network



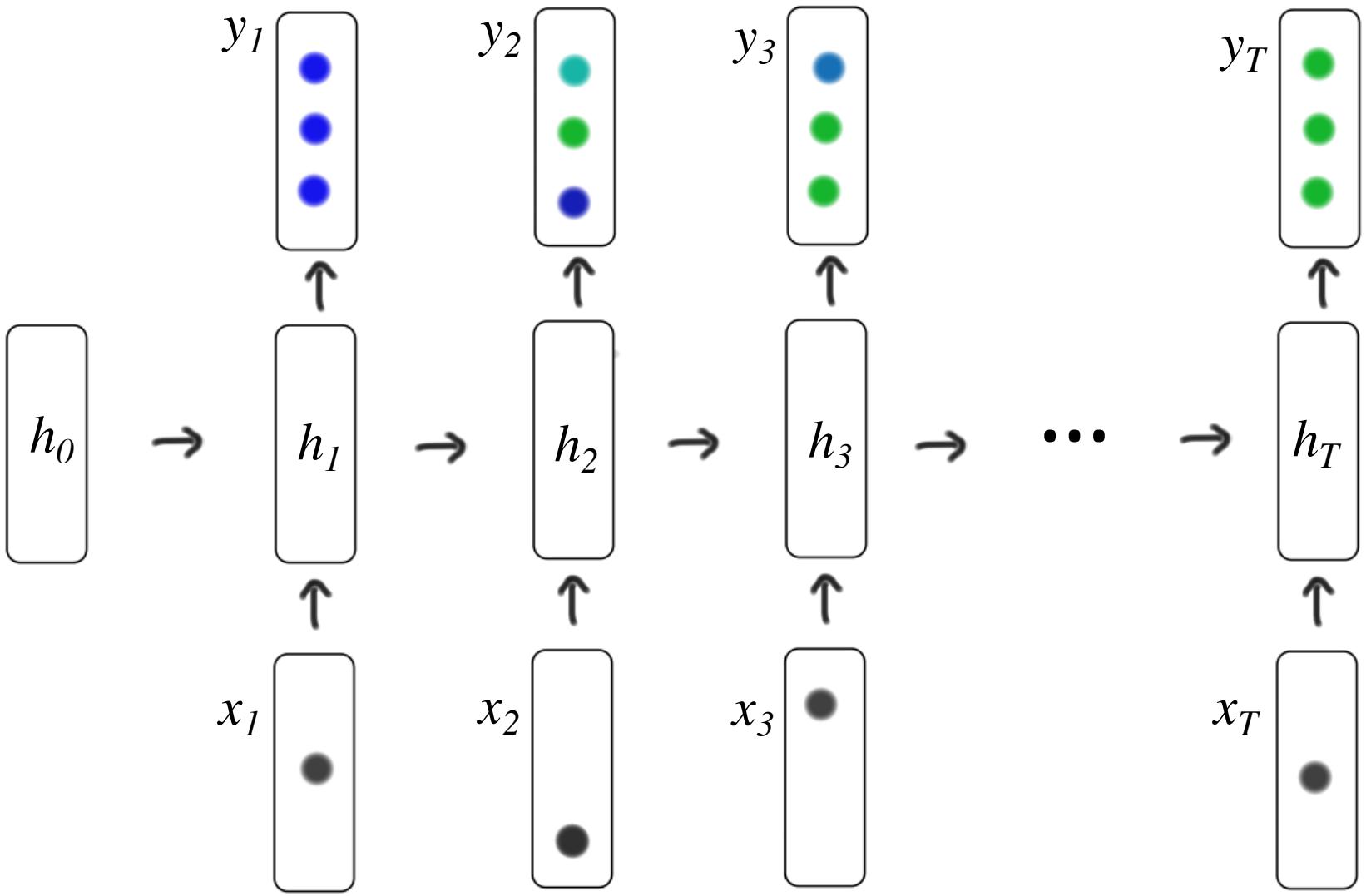
Piech

Recurrent Neural Network



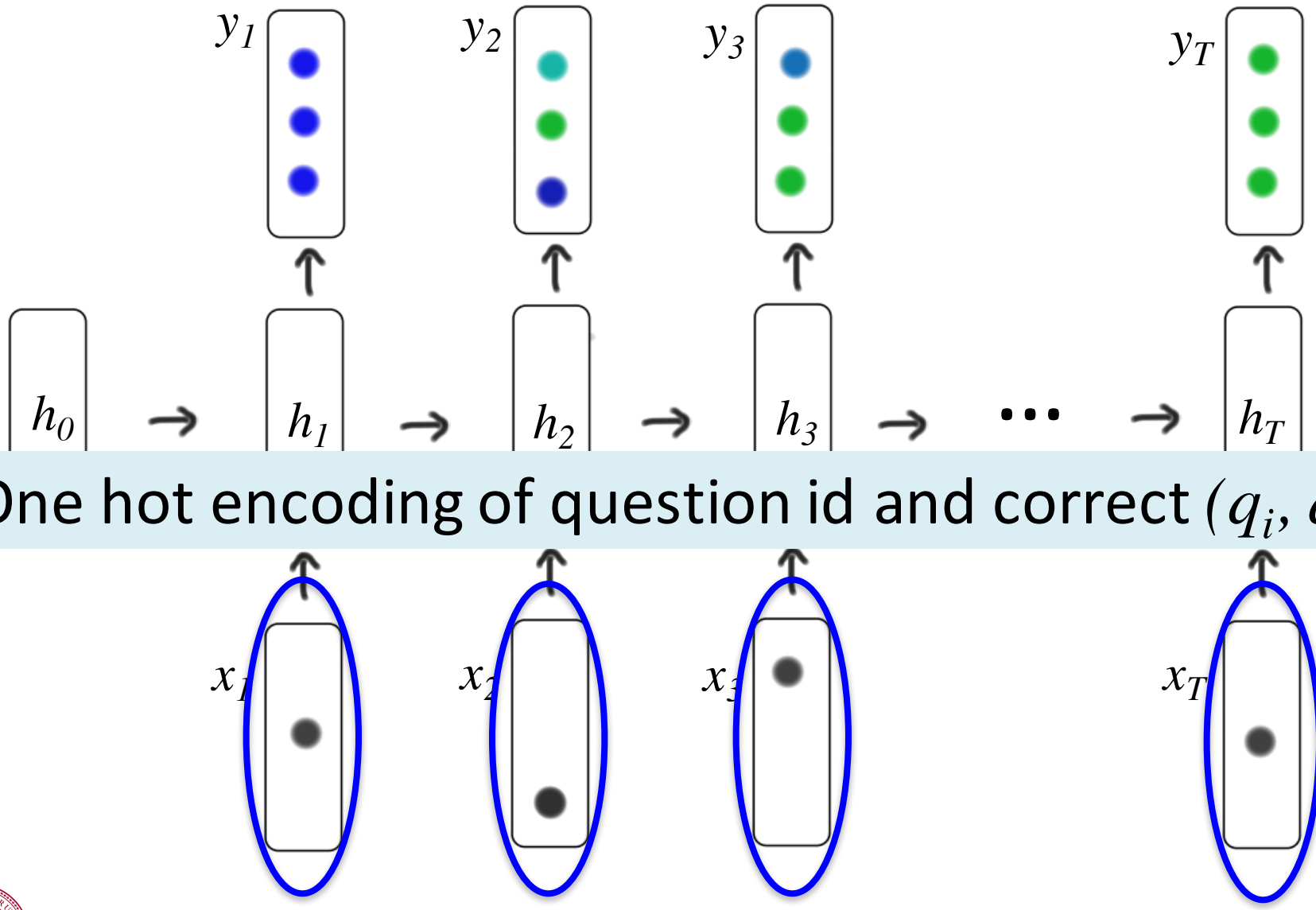
Piech

Recurrent Neural Network



Piech

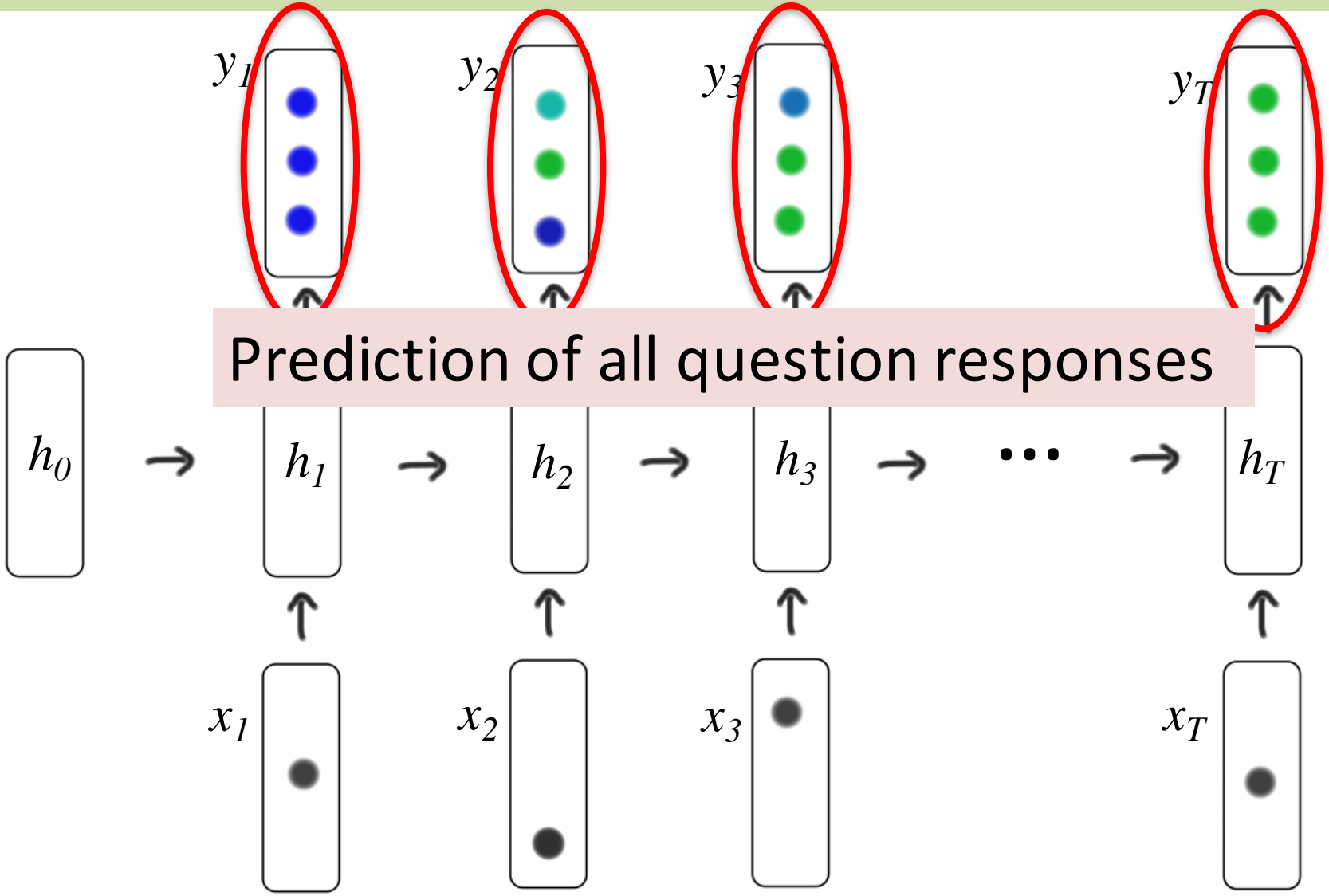
Deep Knowledge Tracing



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

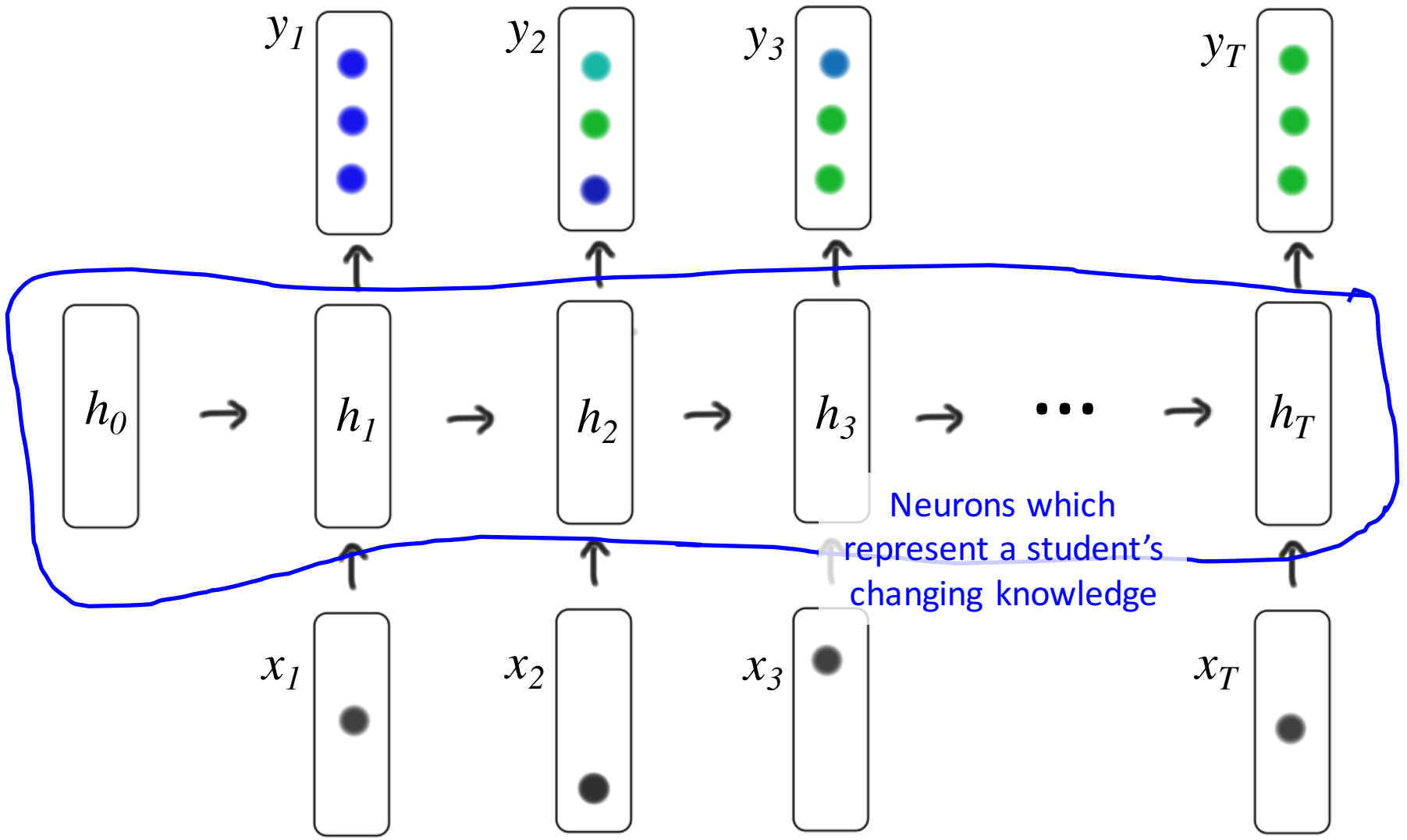
Piech

Deep Knowledge Tracing



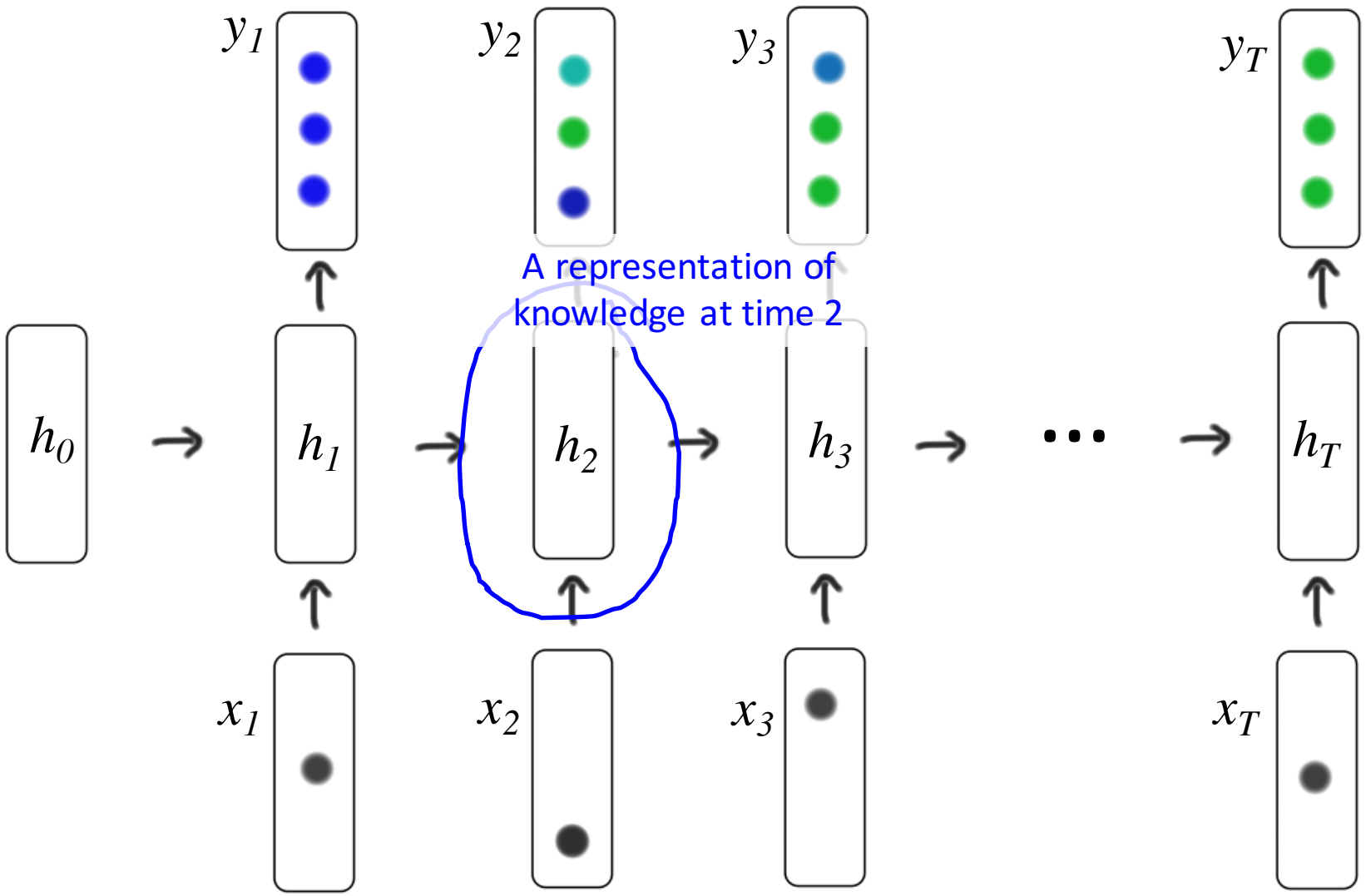
Piech

Recurrent Neural Network



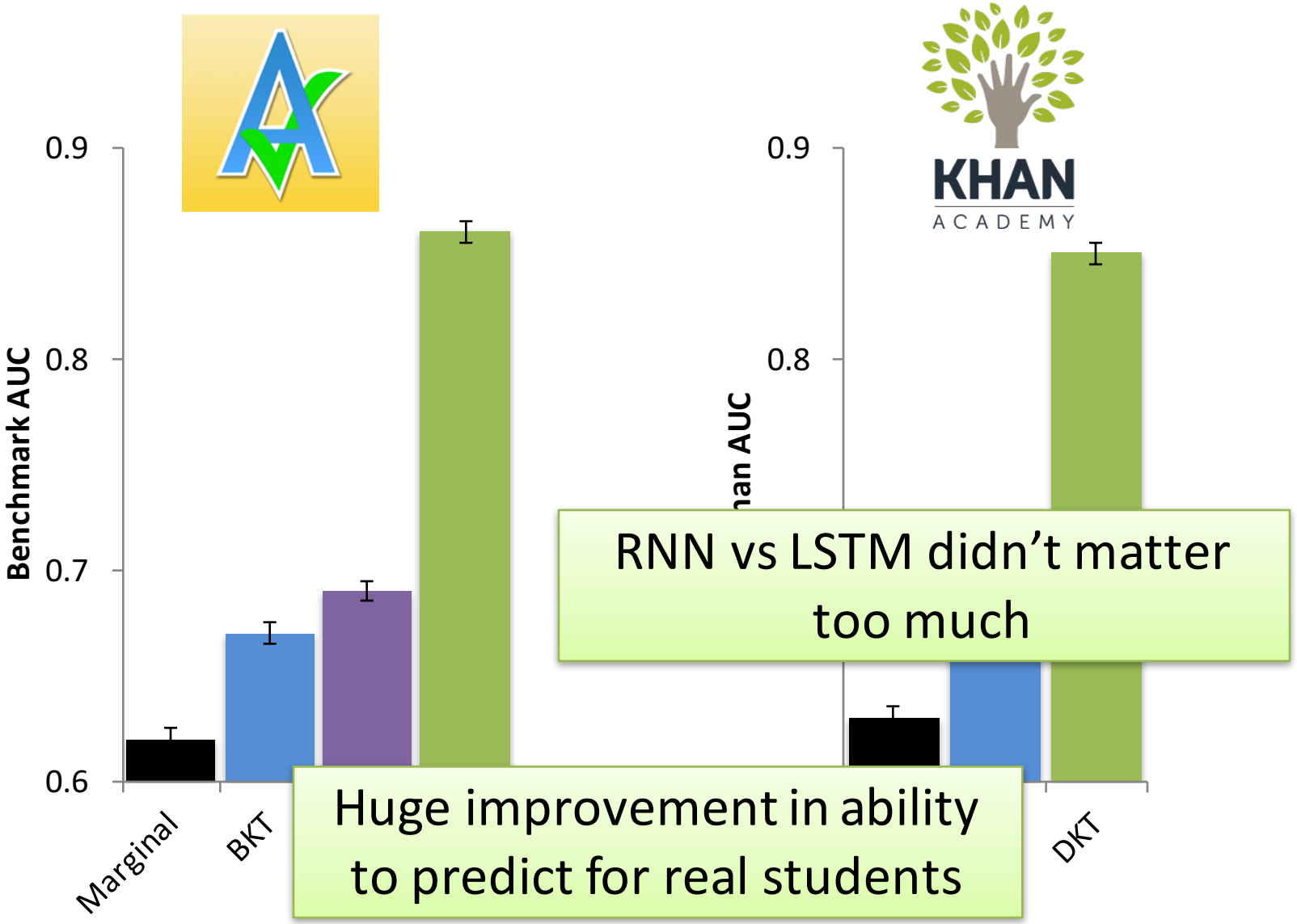
Piech

Recurrent Neural Network



Piech

Prediction Results



RNN vs LSTM didn't matter too much

Huge improvement in ability to predict for real students

Piech

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.

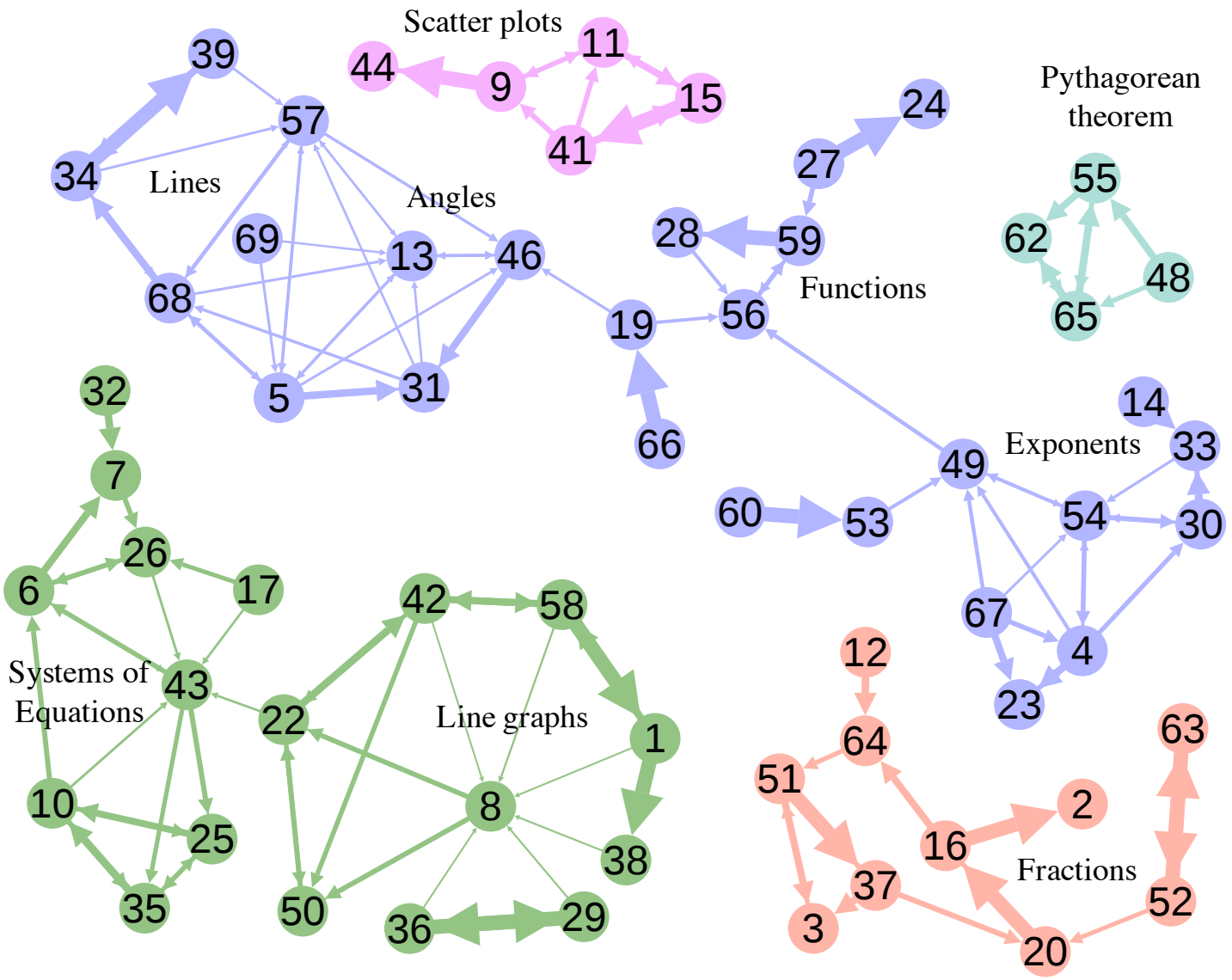
$$\rho(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sqrt{\mathrm{Var(X)Var(Y)}}}$$

$$\mathrm{Cov}(X,Y) = E[XY] - E[X]E[Y]$$

$$= P(XY) - P(X)P(Y)$$

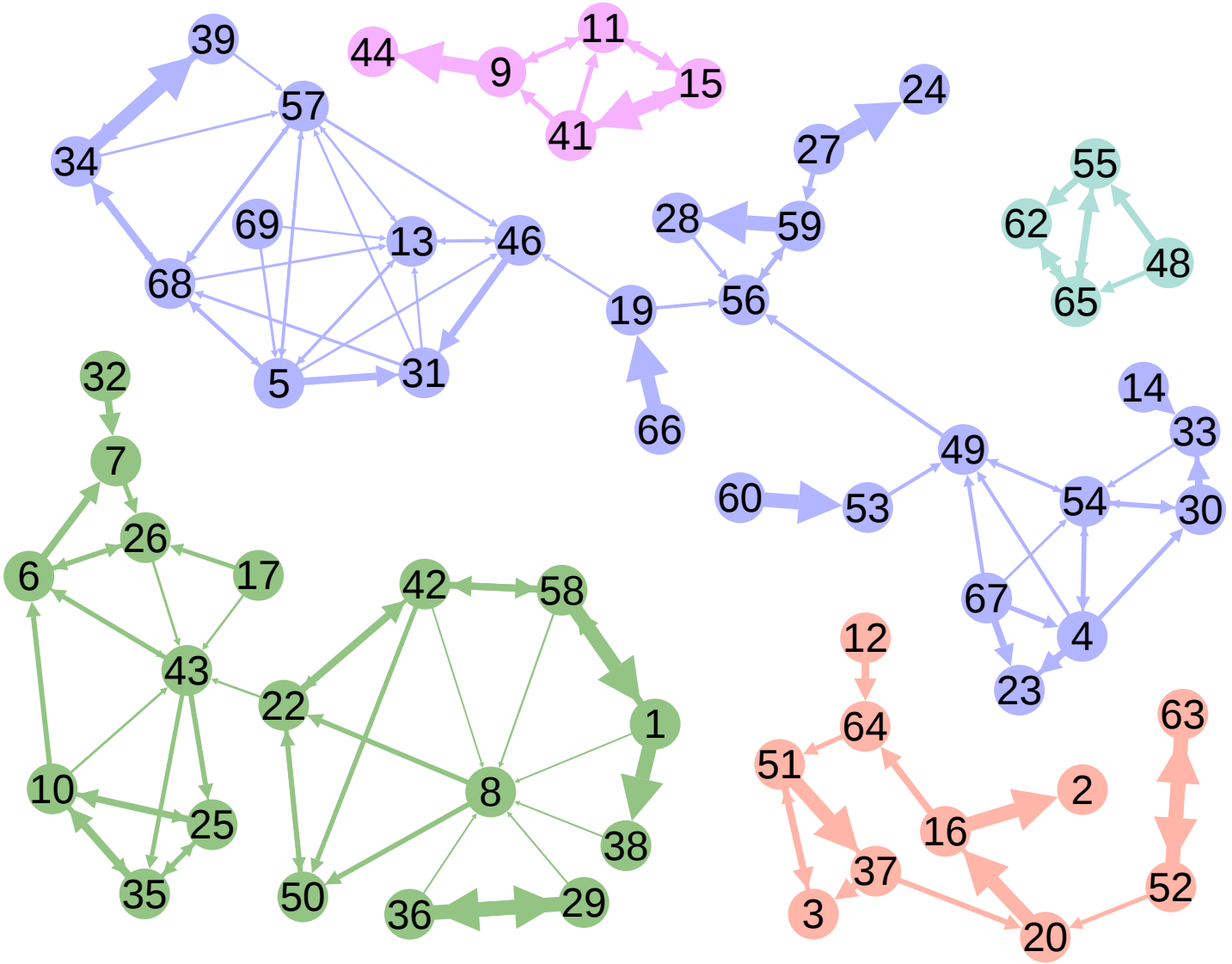
Piech

Learns Concept Relationships



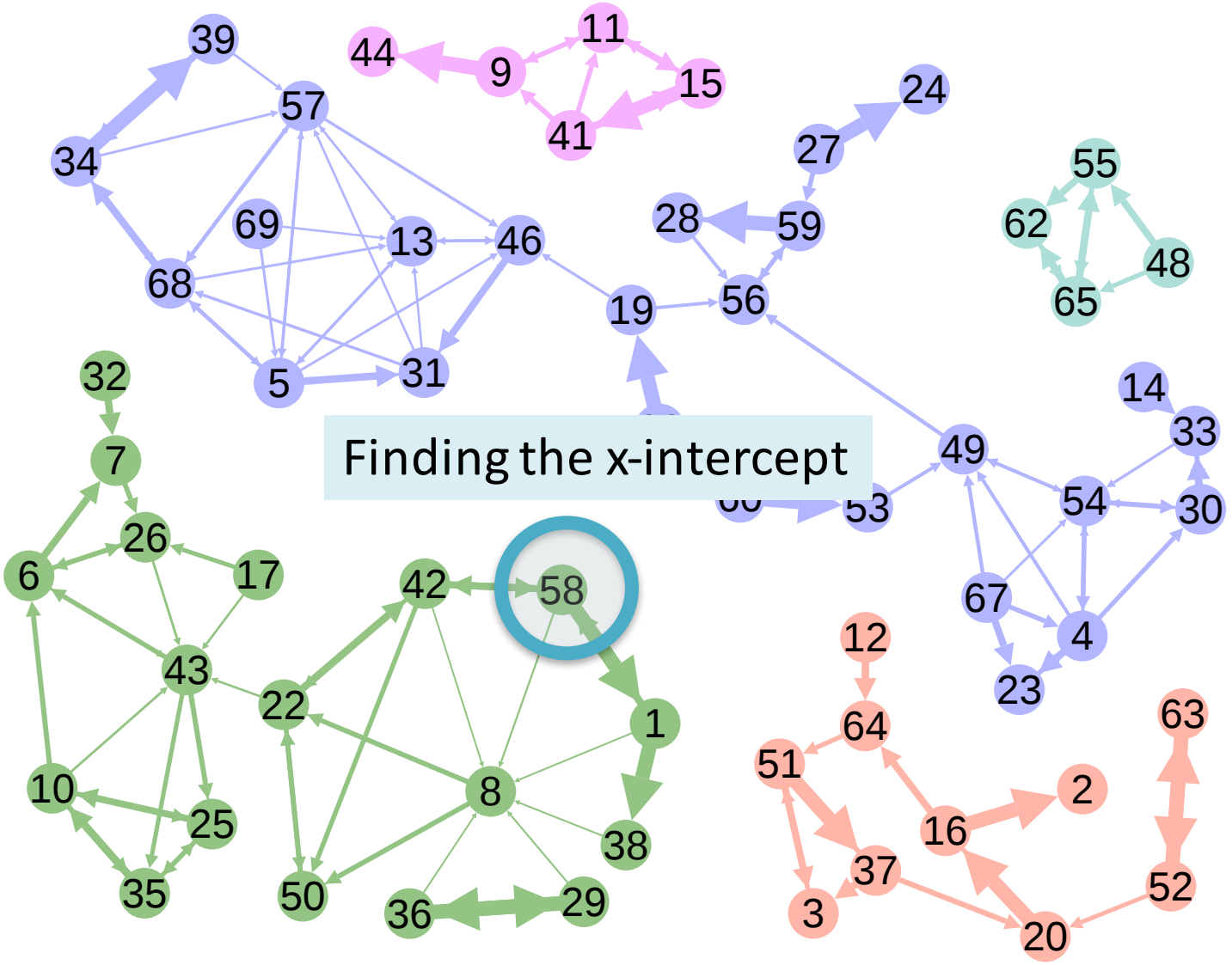
Piech

Learns Concept Relationships



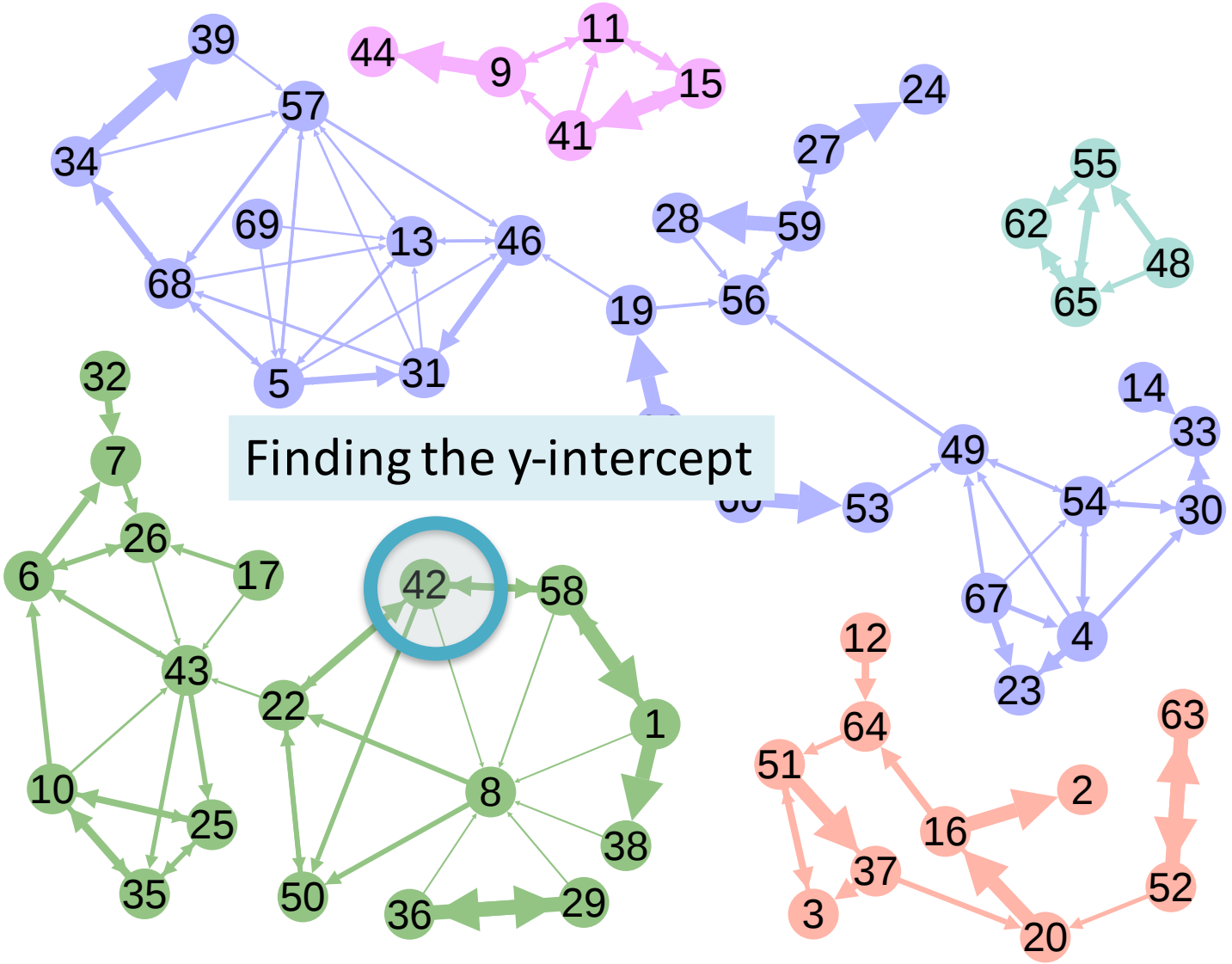
Piech

Learns Concept Relationships



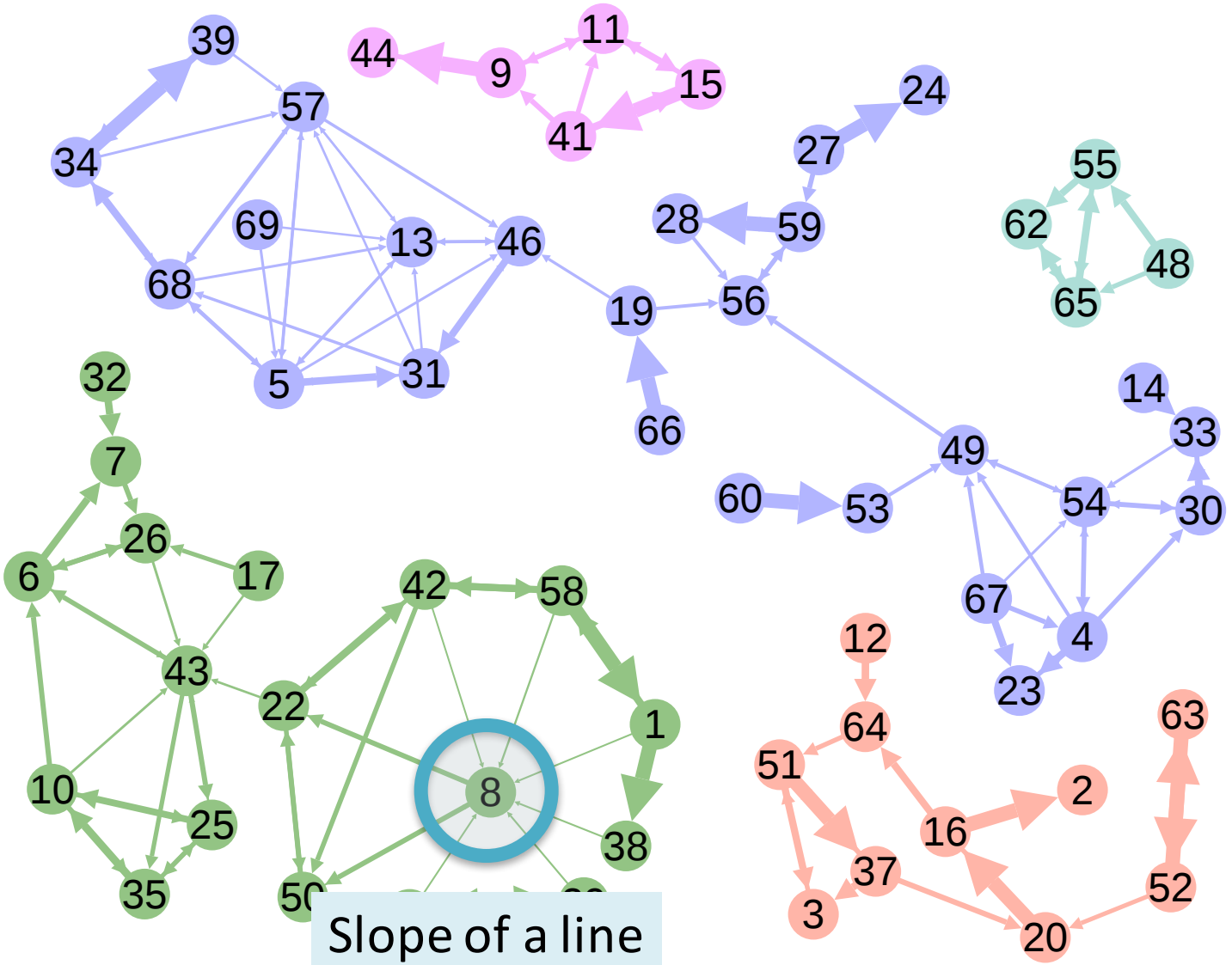
Piech

Learns Concept Relationships



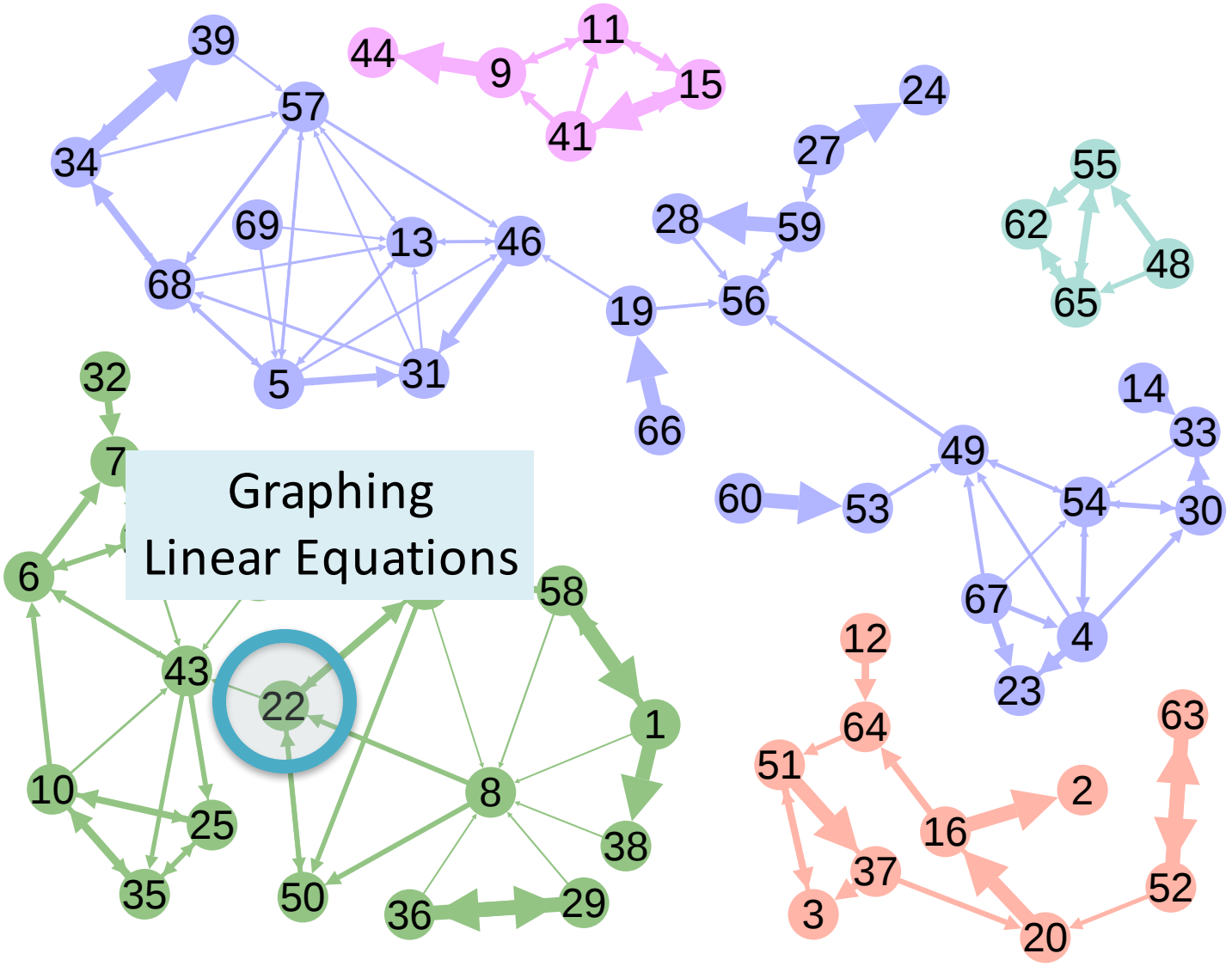
Piech

Learns Concept Relationships



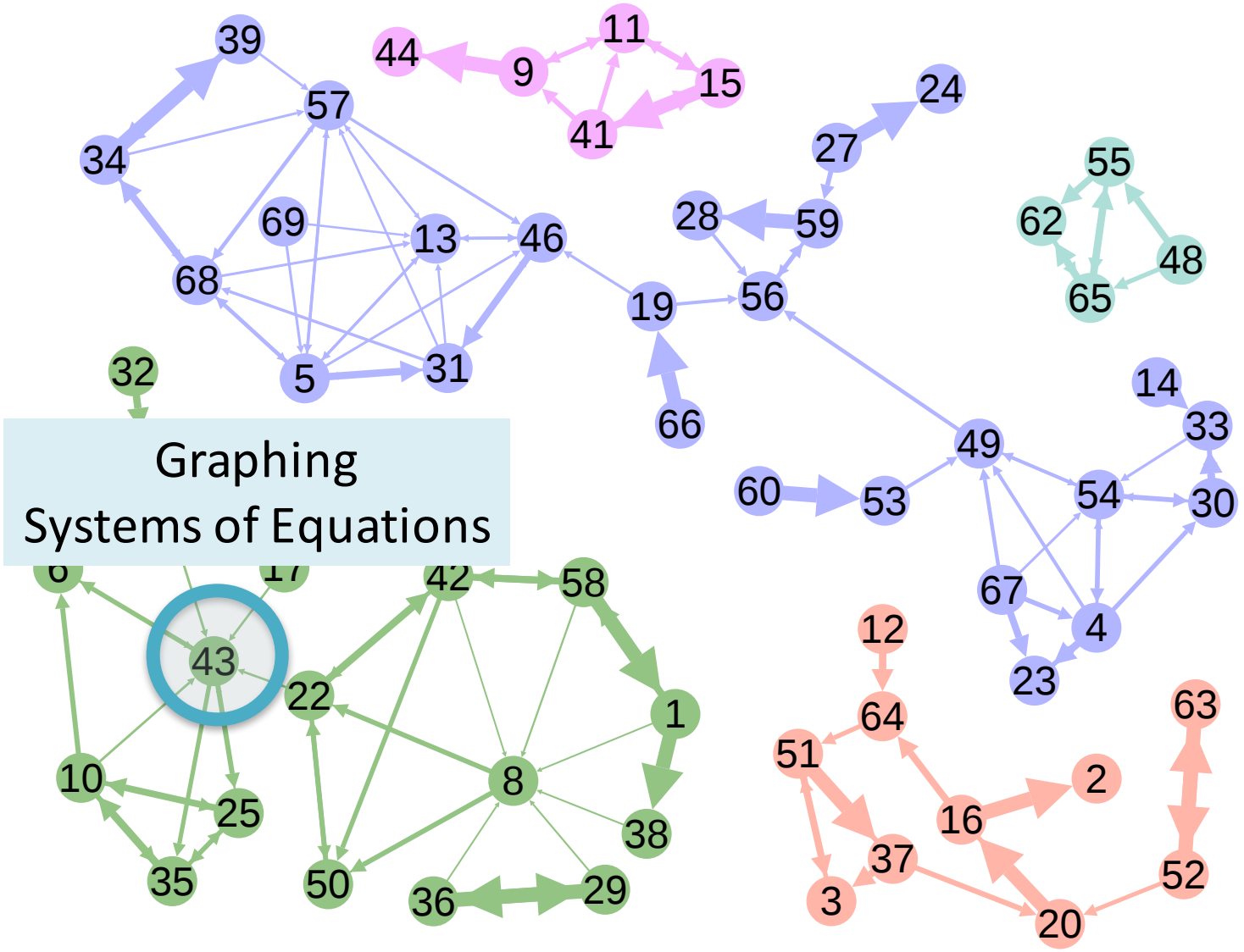
Piech

Learns Concept Relationships



Piech

Learns Concept Relationships



Graphing
Systems of Equations

Piech

Let’s try it!

Piech