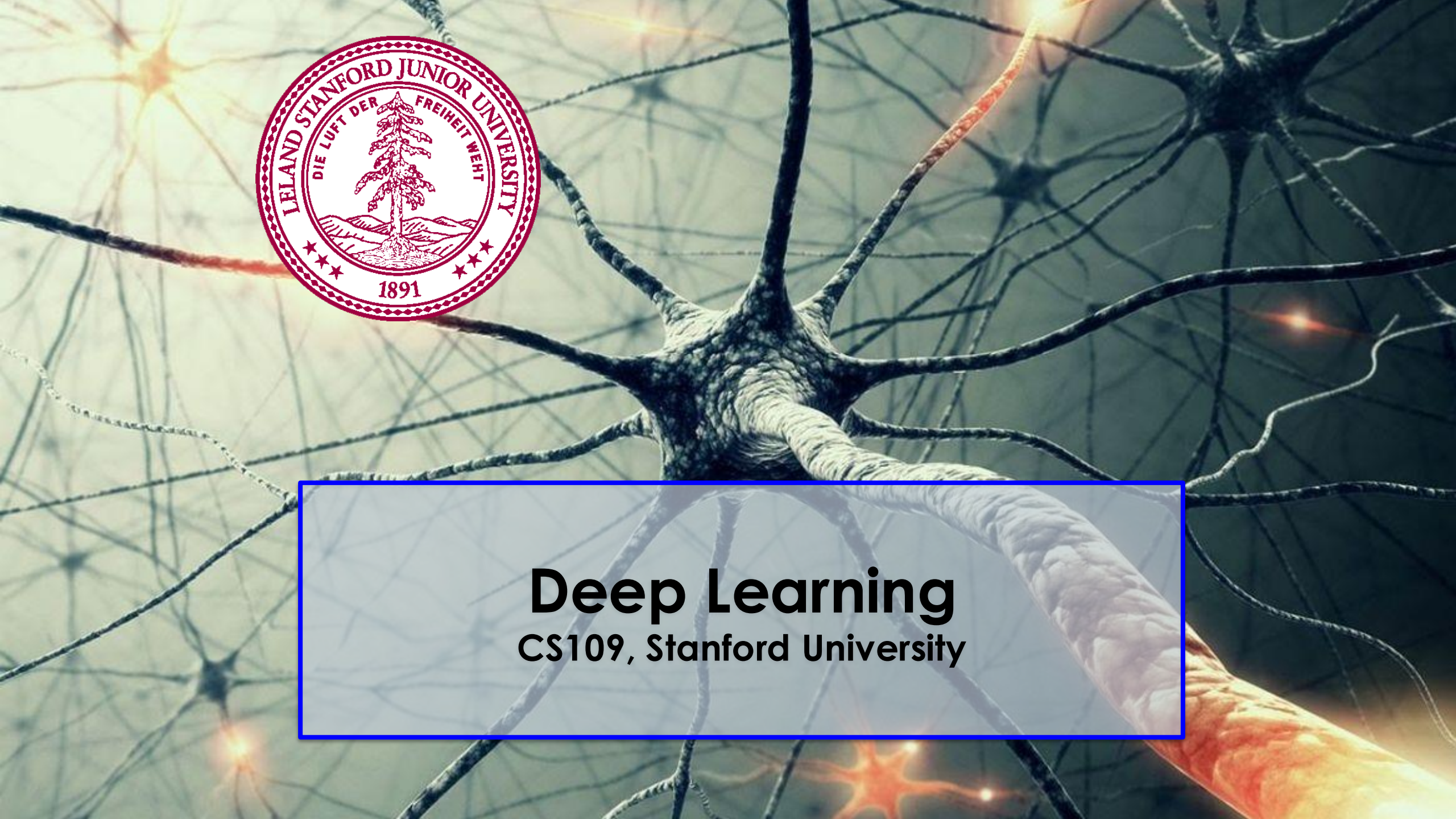
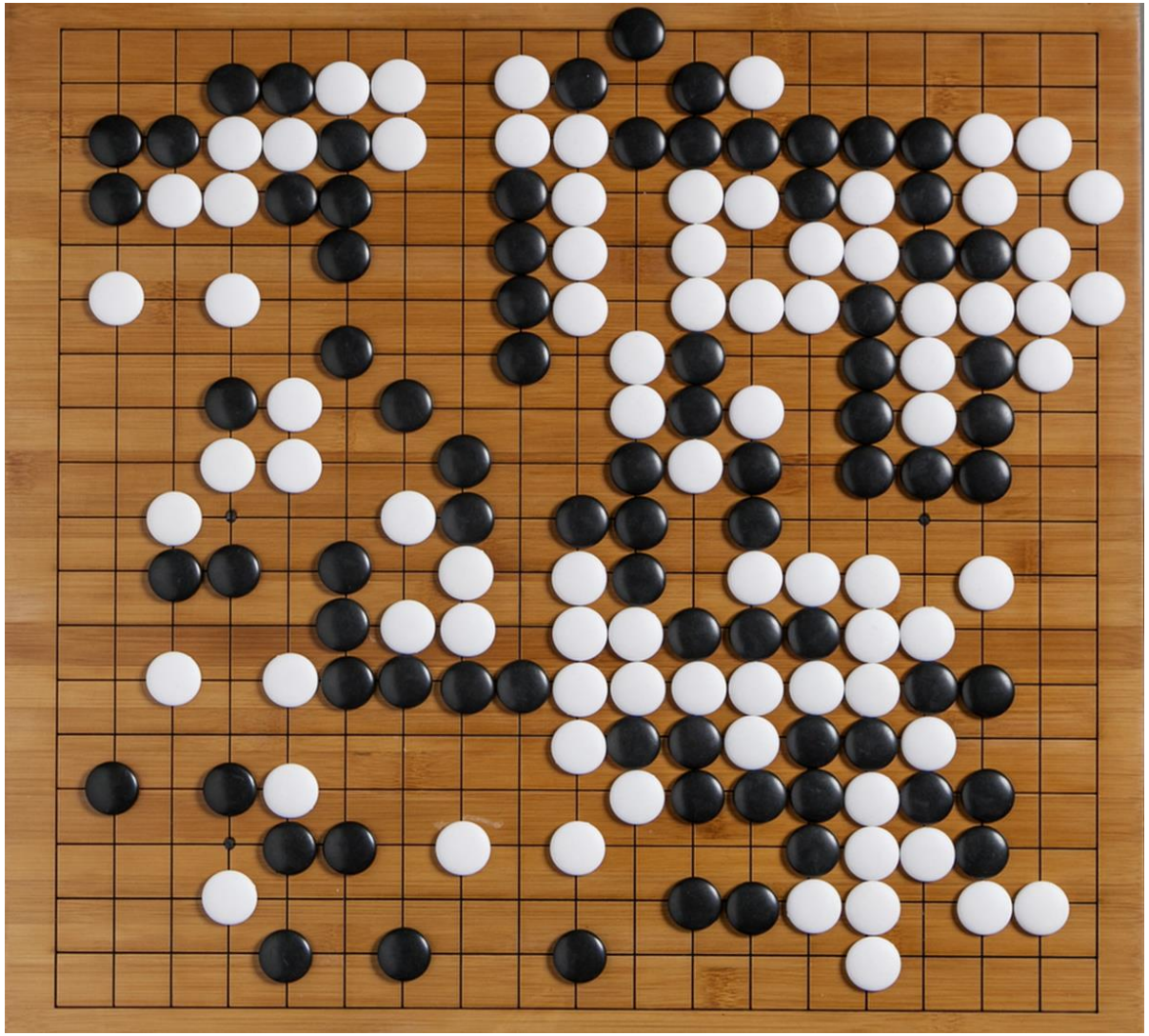


Deep Learning

CS109, Stanford University

Innovations in deep learning



AlphaGO (2016)

Deep learning (neural networks) is the core idea driving the current revolution in AI.

Notes:

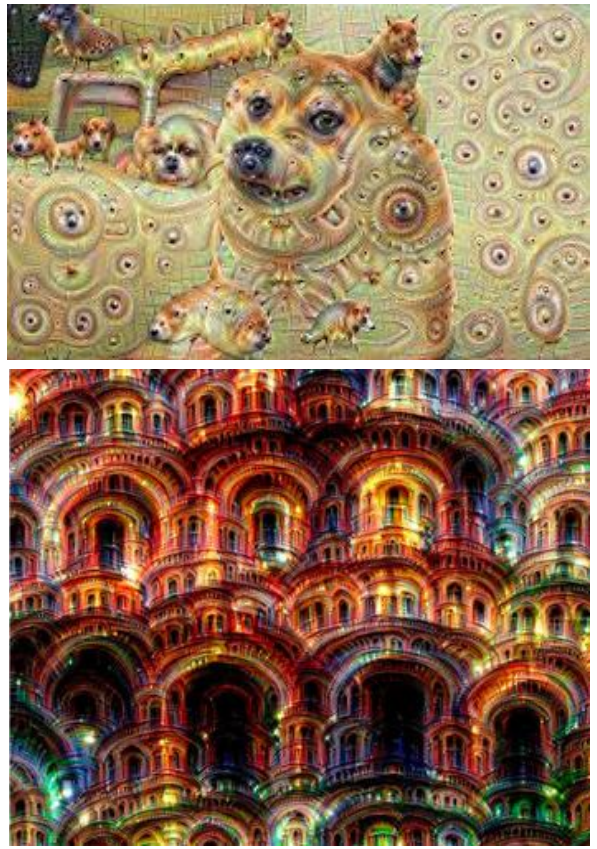
- Checkers is the last **solved** game (from game theory, where perfect player outcomes can be fully predicted from any gameboard).
  https://en.wikipedia.org/wiki/Solved_game
- The first machine learning algorithm defeated a world champion in Chess in 1996.
  https://en.wikipedia.org/wiki/Deep_Blue_(chess_computer)

Self Driving Cars

Computers making art

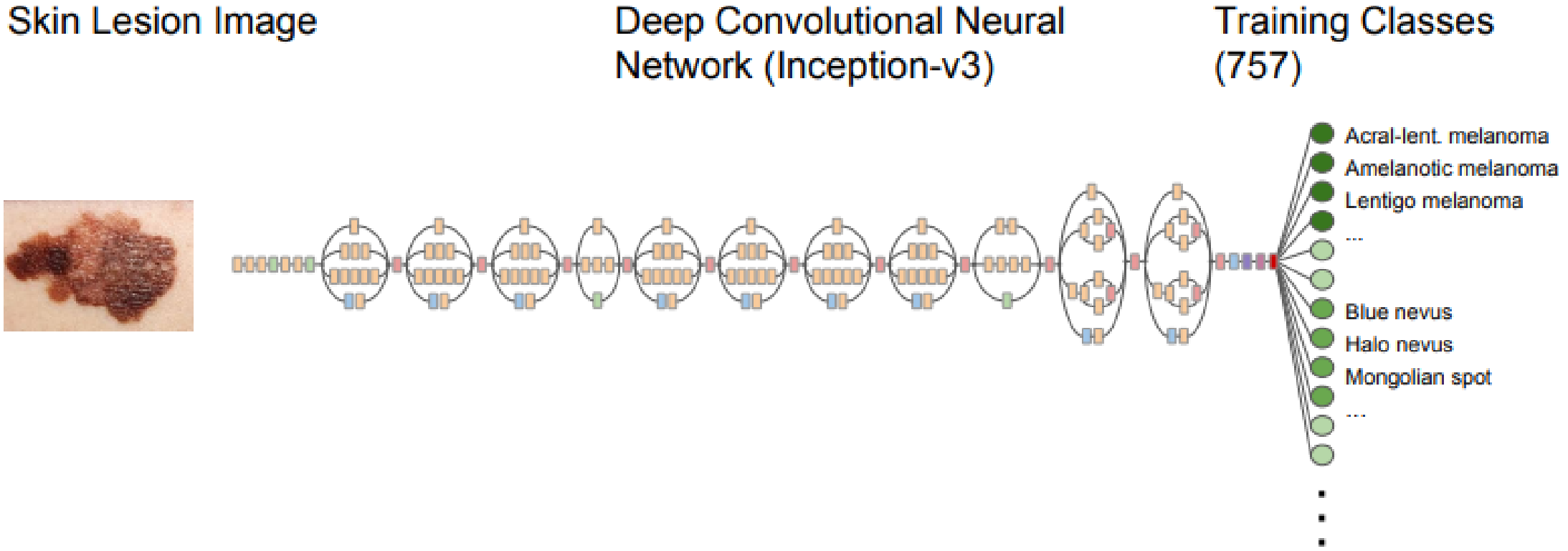
The Next Rembrandt
https://medium.com/@DutchDigital/the-next-rembrandt-bringing-the-old-master-back-to-life-35dfb1653597

A Neural Algorithm of Artistic Style
https://arxiv.org/abs/1508.06576
https://github.com/jcjohnson/neural-style



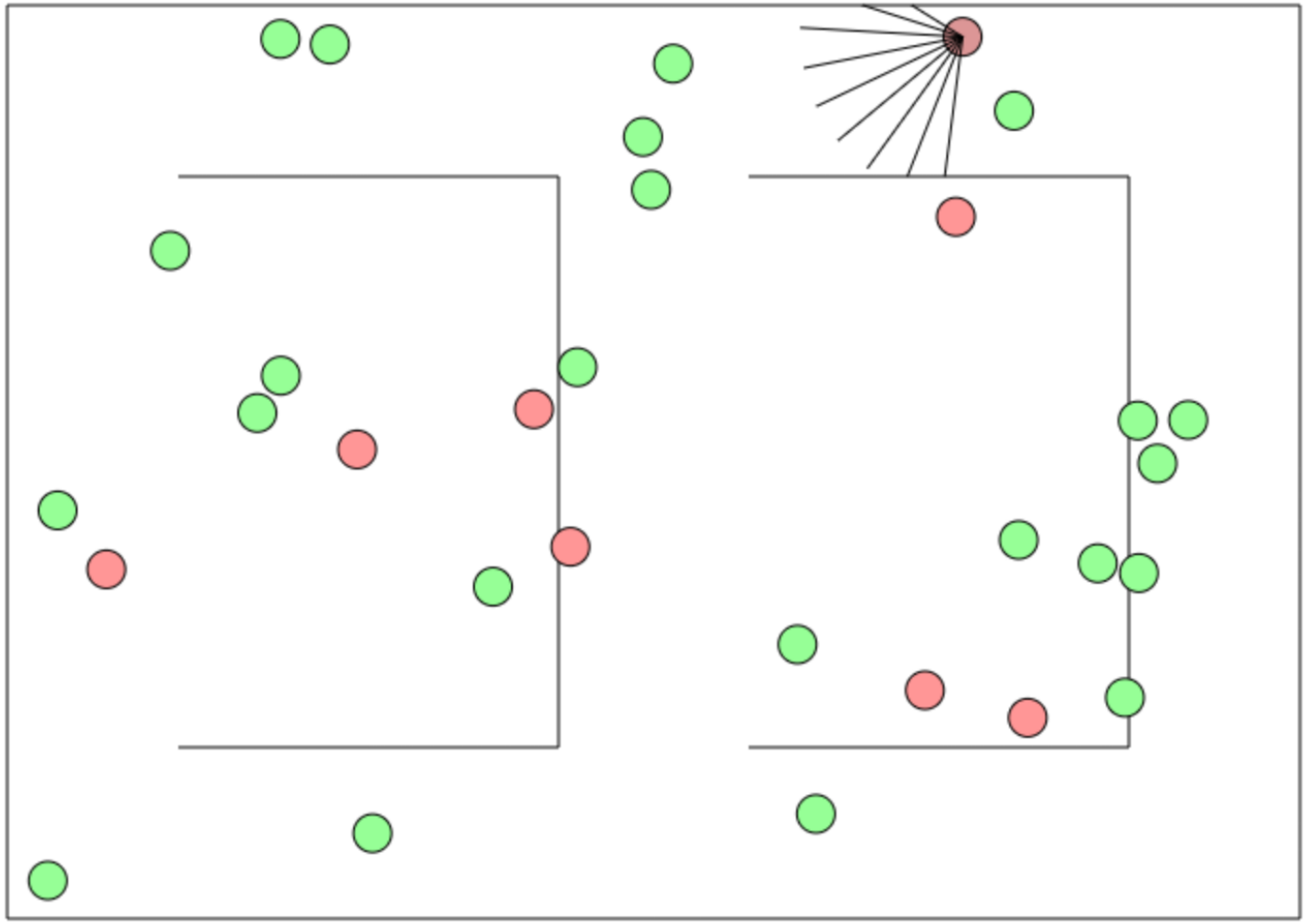
Google Deep Dream
https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

Detecting skin cancer



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

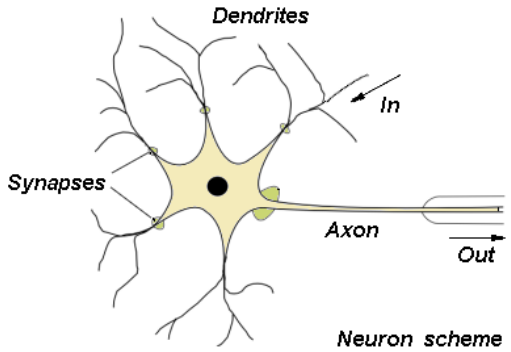
Our Little Buddy

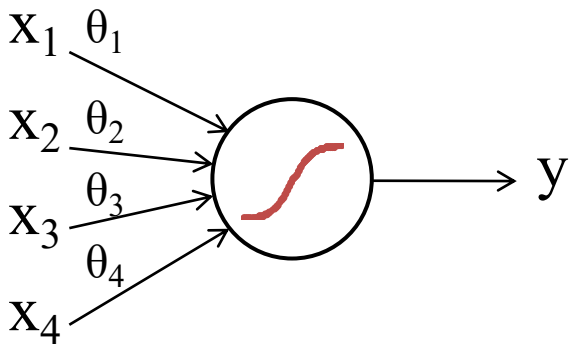


http://cs.stanford.edu/people/karpathy/convnetjs/demo/rldemo.html

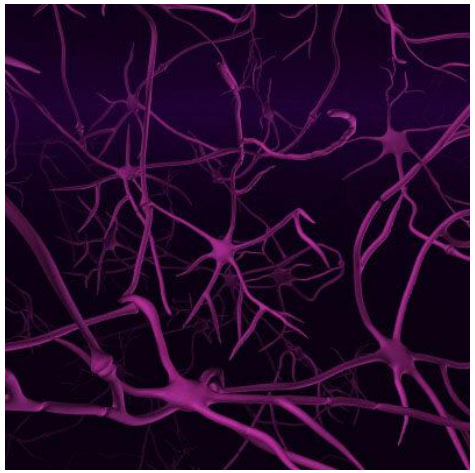
Biological Basis for Neural Networks

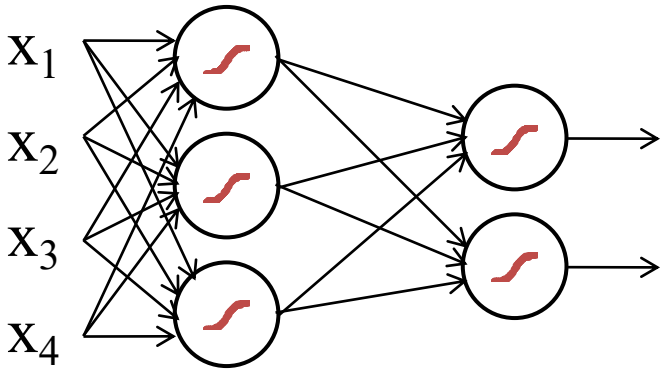
A neuron





Your brain





Actually, it's probably someone else's brain

Logistics

End of Class Schedule

ML PSet

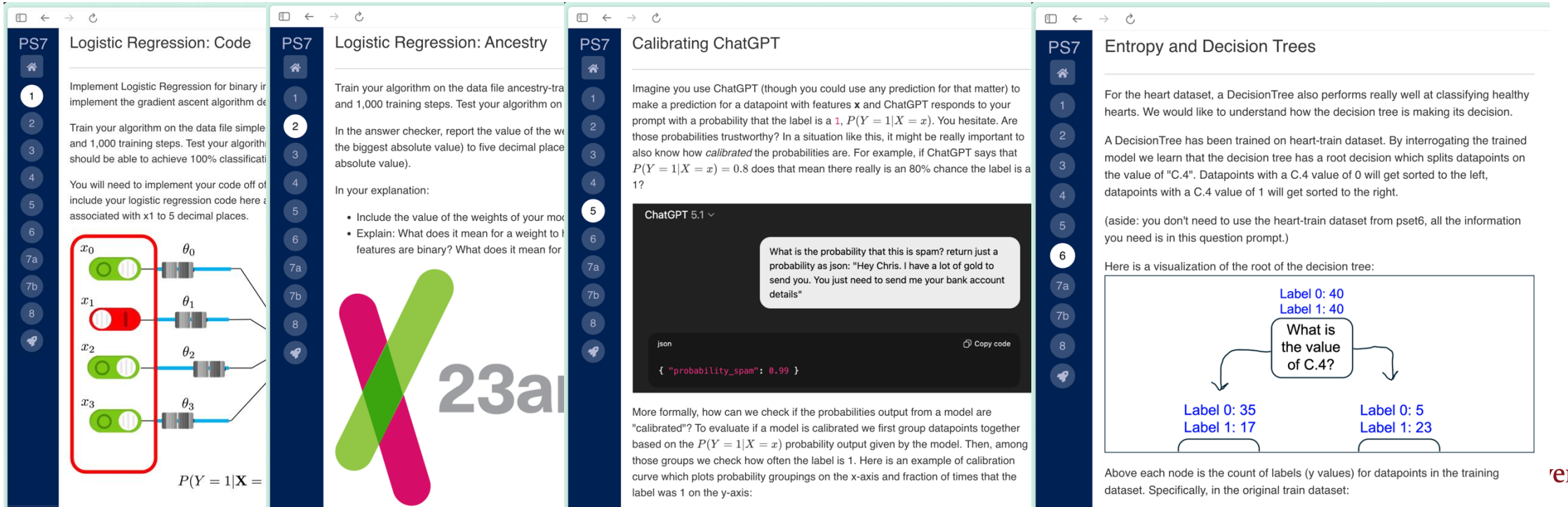
Week	Monday	Wed	Fri
This week (March 2nd)	Deep Learning	Regression	Application / Practice
Last Week of Class! (March 9th)	Application / Practice	Last Class	Exam Review Session

Final PEP

Pset 7 is Live !!

- Due Weds 3/11 (used to be due 3/13 but we cut some problems)
  - Can do problems 1, 2, 3, 4, 5, 6, 8 after today's lecture

- Can use late days on the assignment.



Final PEP Sign Ups

- Personal Exam Prep (PEP) for the final is next week Mon-Weds.

- Signups will be live soon (look out for Ed announcement)

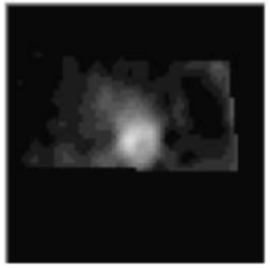
- Slightly different format: random homework question. Make sure to review homeworks! Can focus on Pset 5 and Pset 6.

- Students must do a final PEP for me to give you a grade in the class.

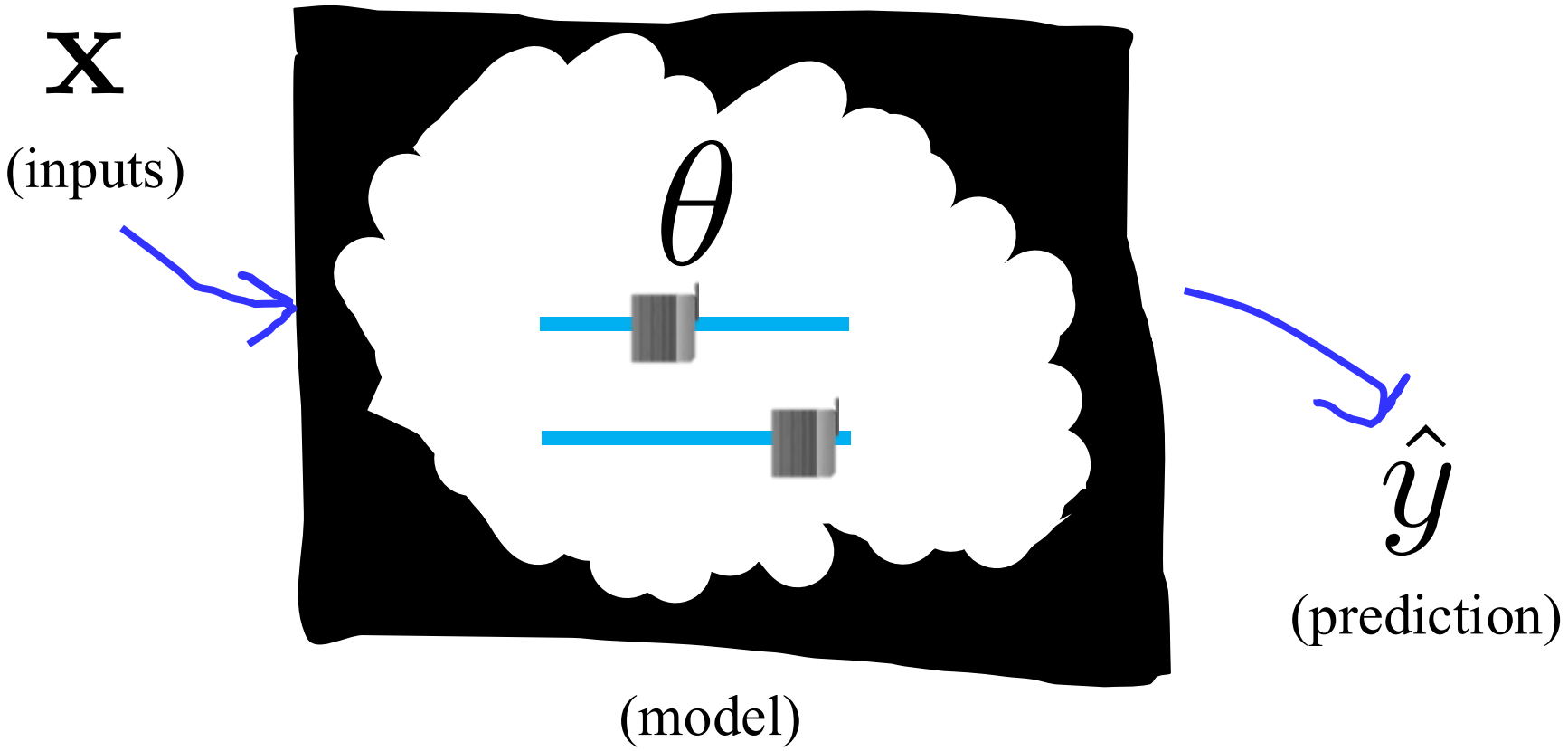
- How are PEPs graded? We will assume you are caught up through Friday's lecture (3/6). Don't need to solve everything perfectly, but need to be able to engage meaningfully.

Review

Classification Task

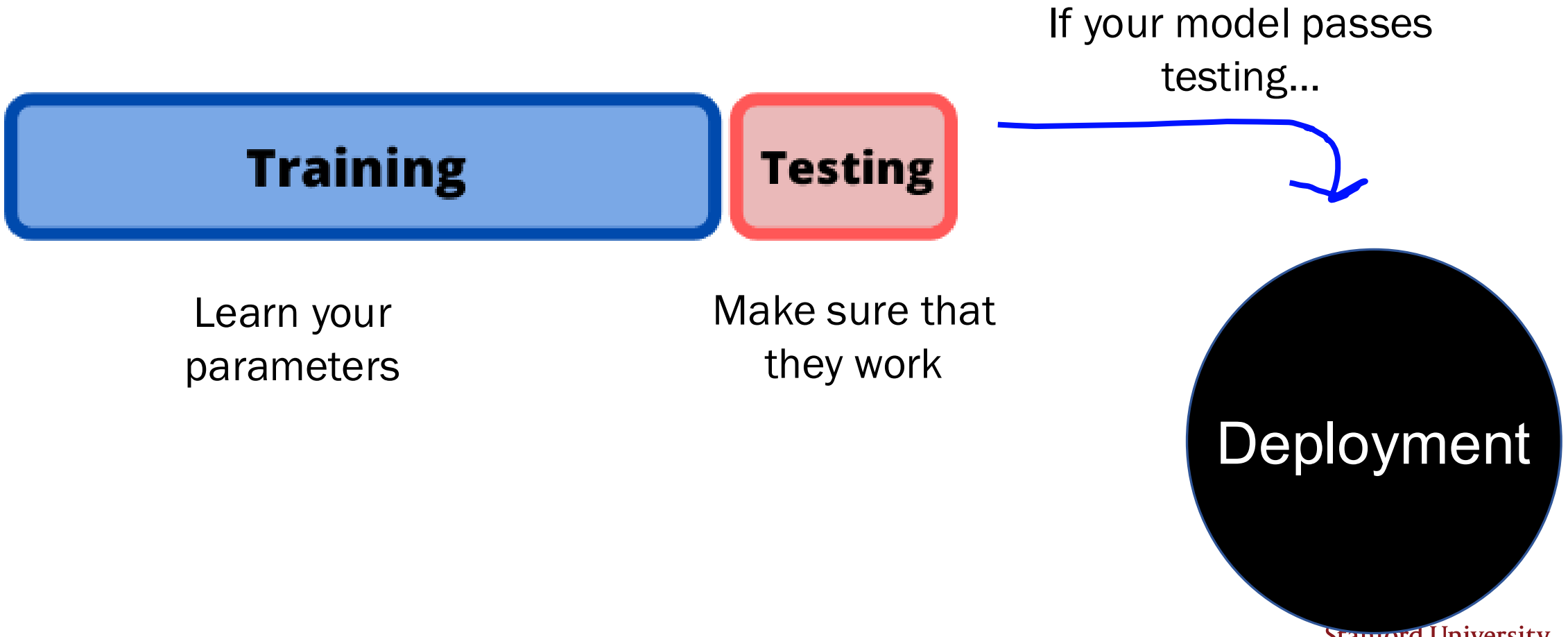
	ROI 1	ROI 2	...	ROI *m*	Output
					
Heart 1	0	1		1	0
Heart 2	1	1		1	0
			⋮		⋮
Heart *n*	0	0		0	1

Machine Learning

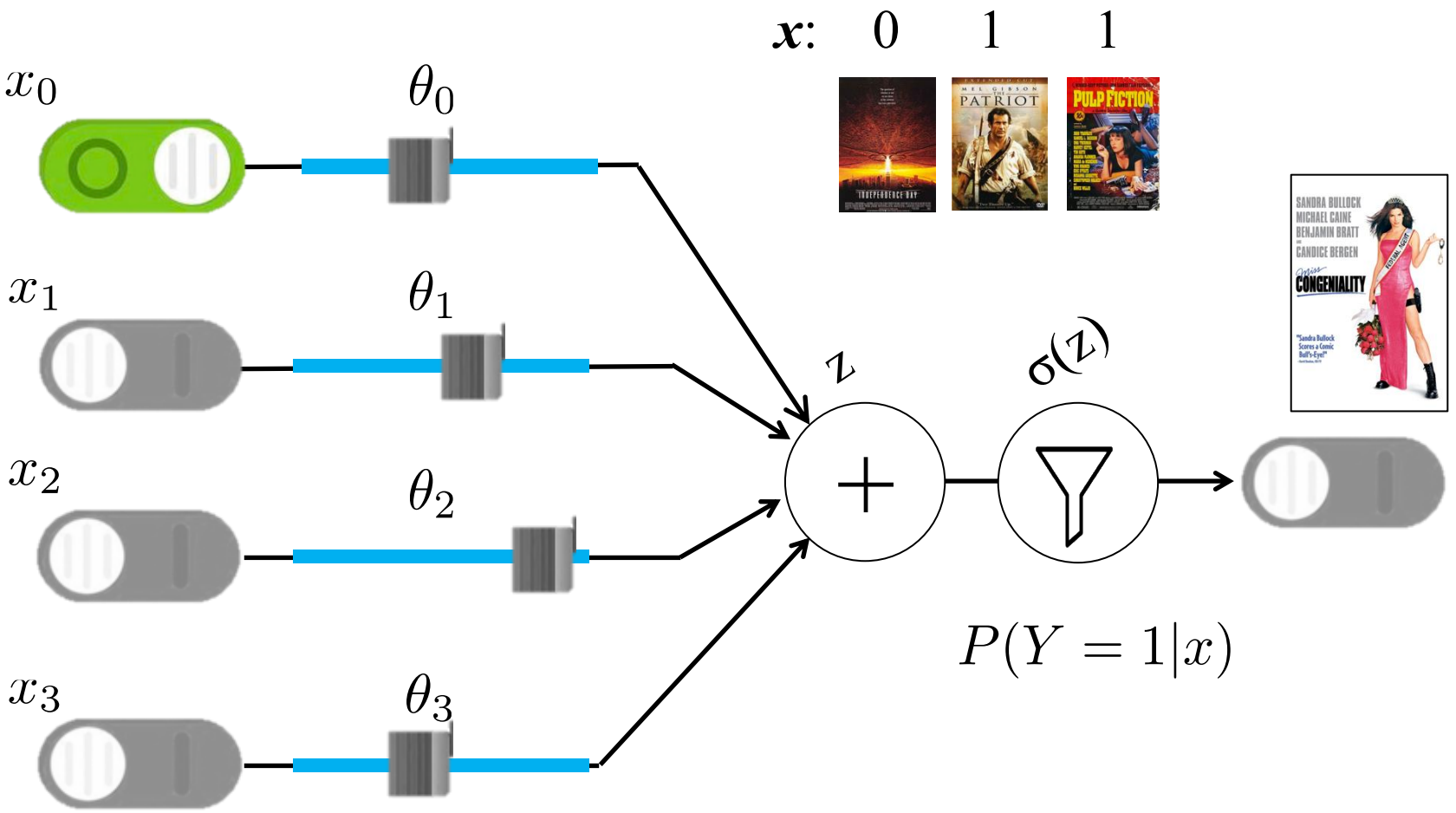


The Training / Testing Paradigm

The Training / Testing Paradigm

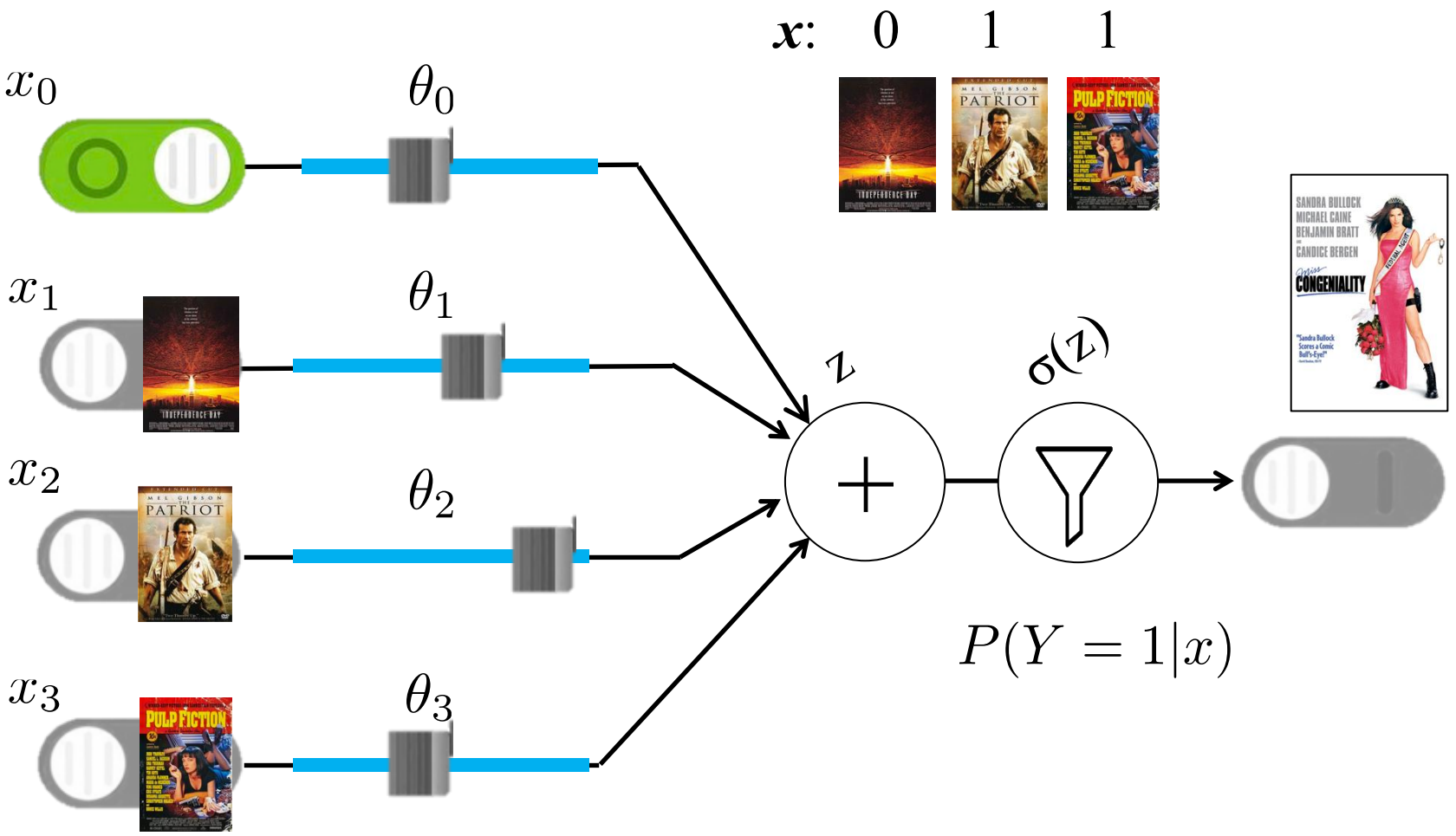


Logistic Regression



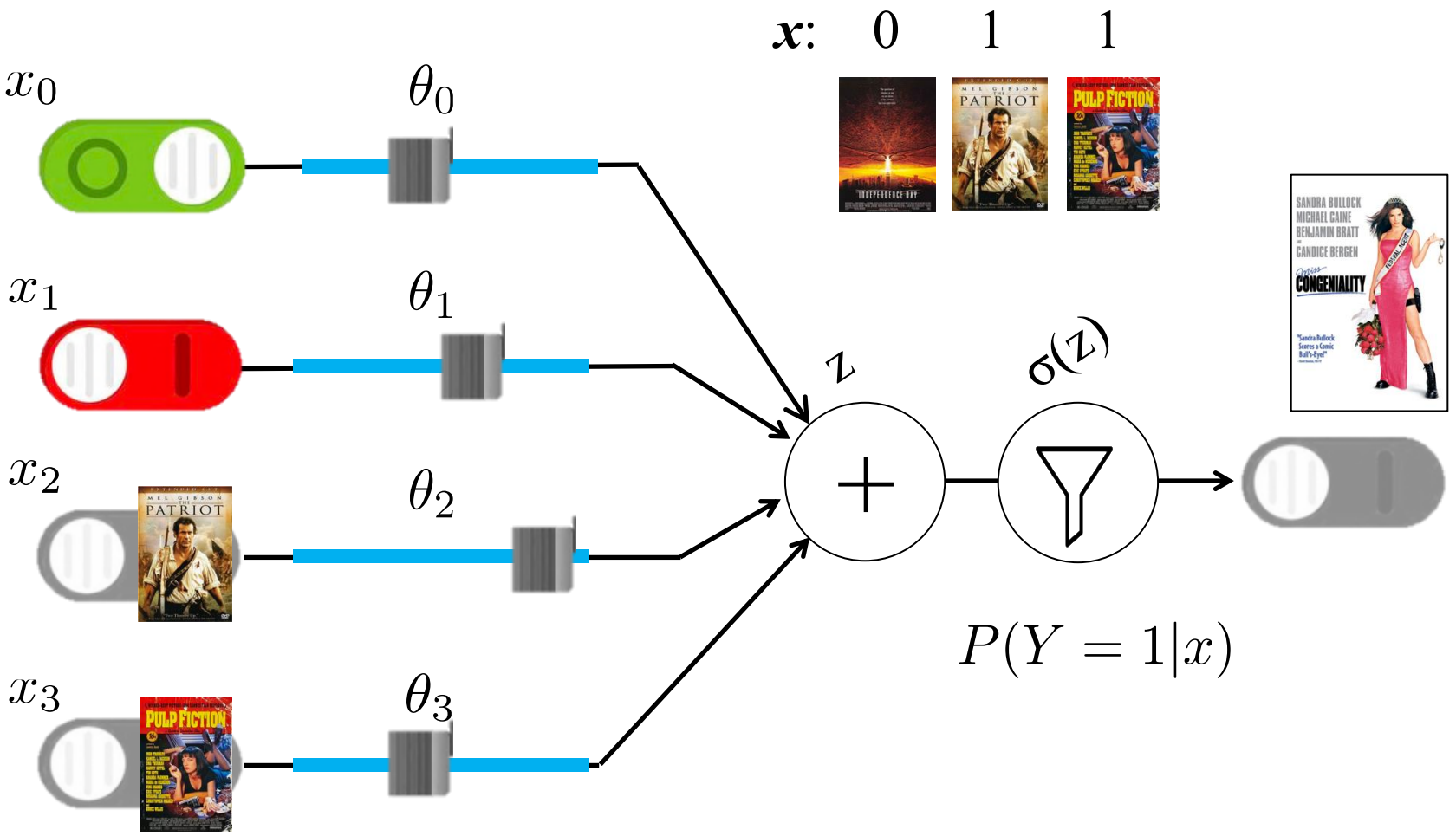
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



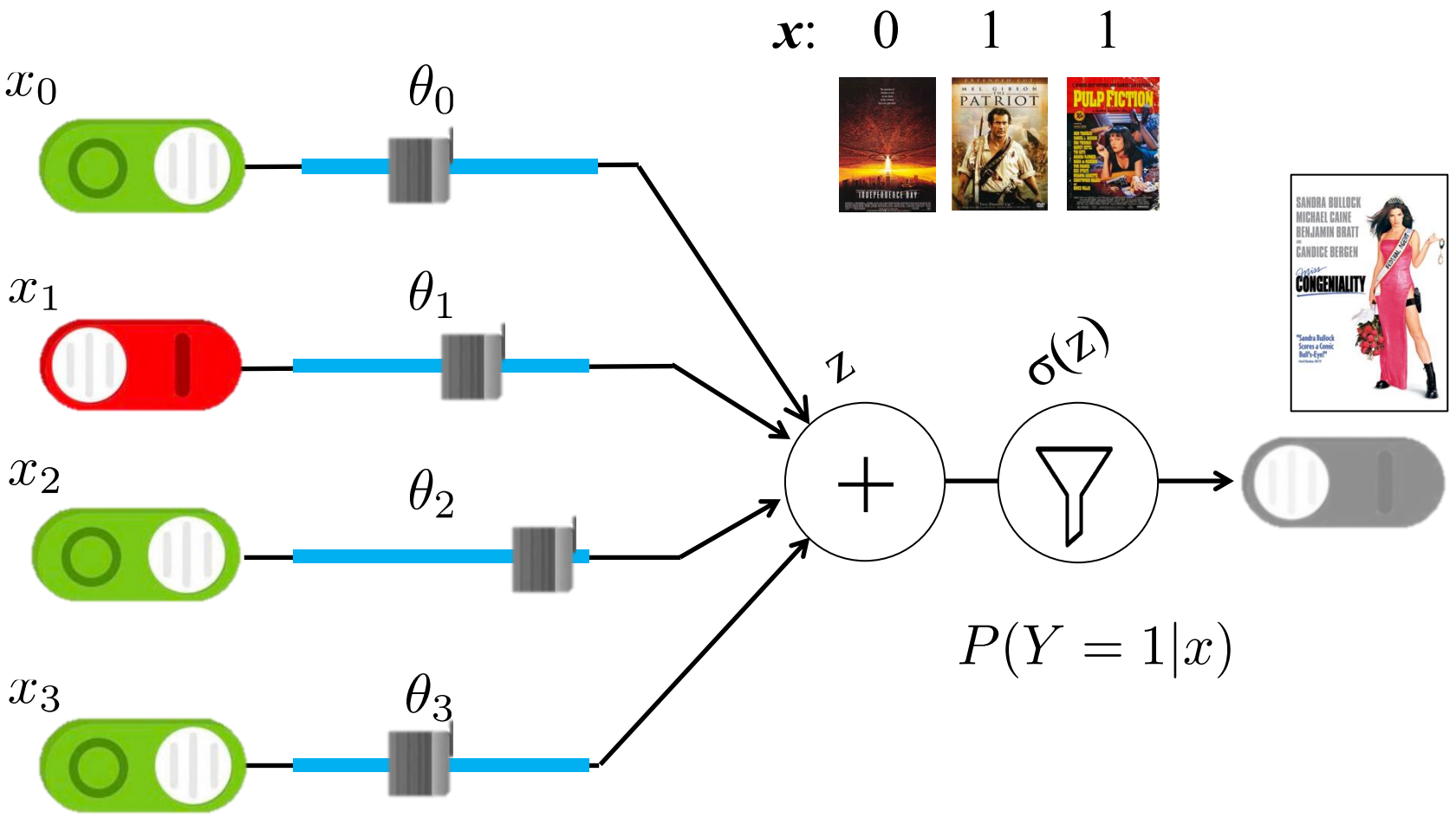
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



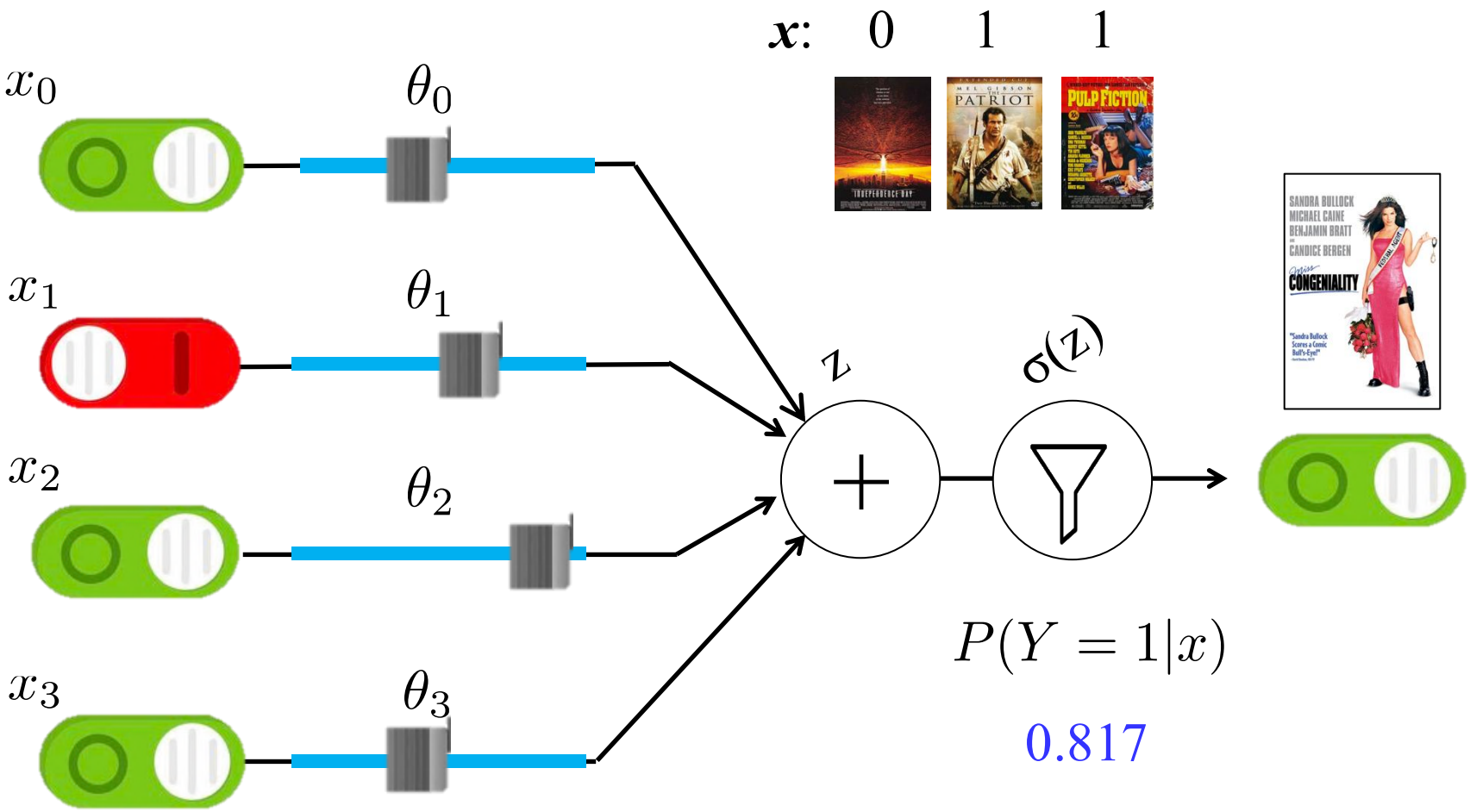
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



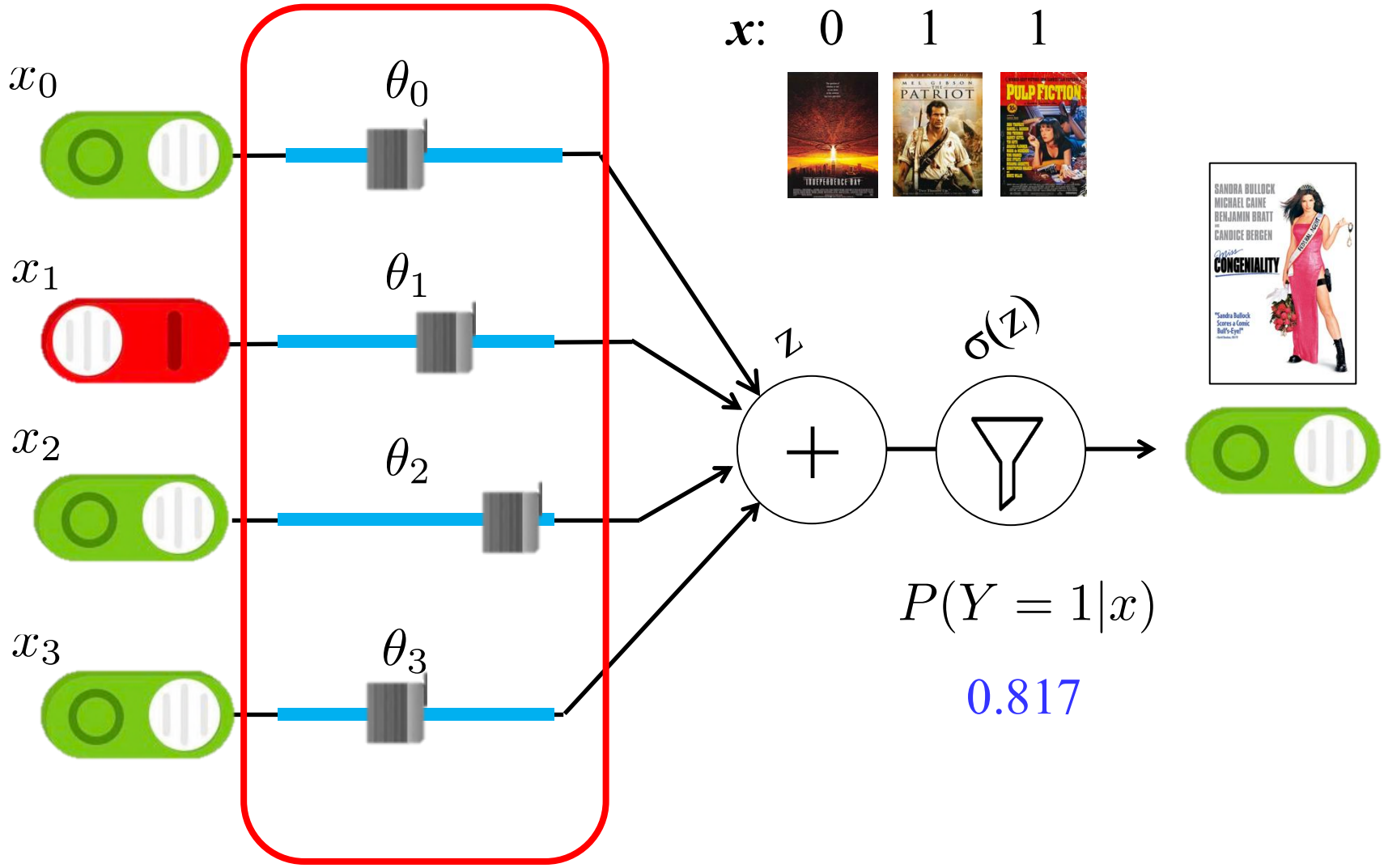
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



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

Logistic Regression



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

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

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

3 Get derivative of log likelihood with respect to thetas

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

Logistic Regression Training

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

Repeat many times:

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

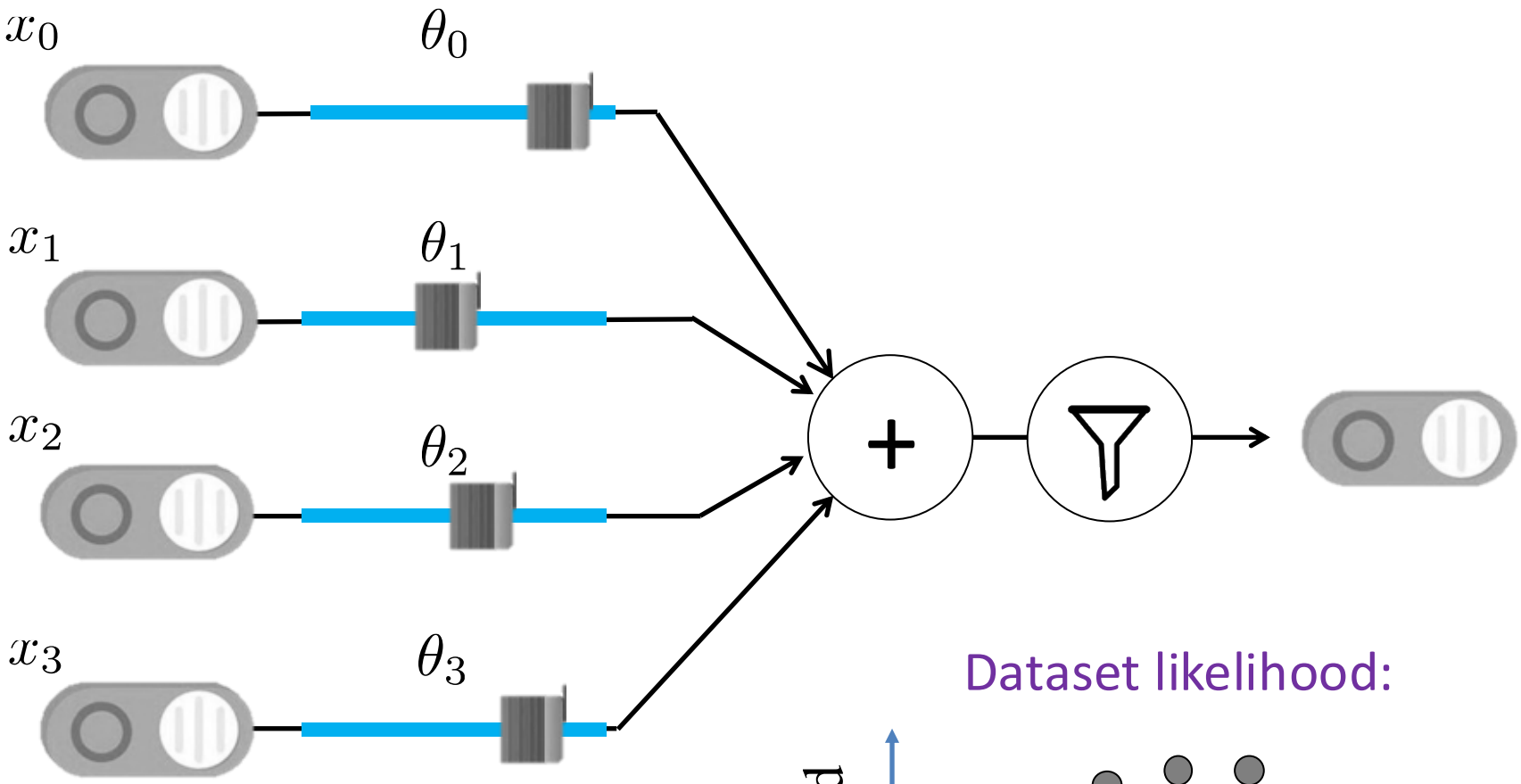
For each parameter j

For each training example (x, y):

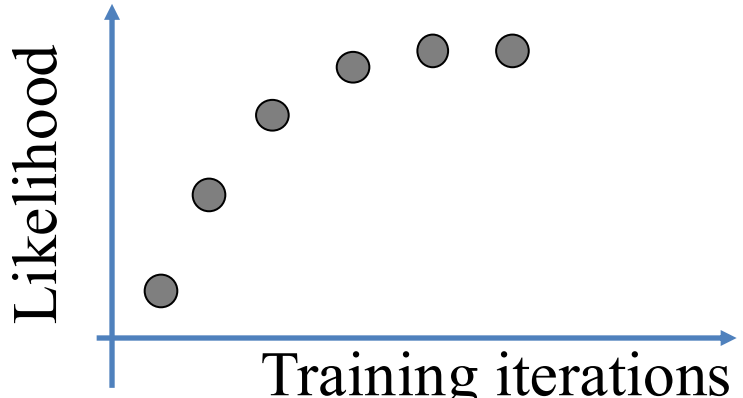
gradient[j] $+= x_j \left(y - \frac{1}{1 + e^{-\theta^T \mathbf{x}}} \right)$

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

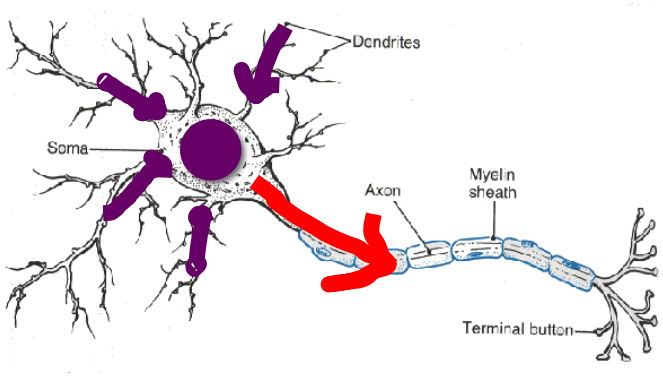
Training

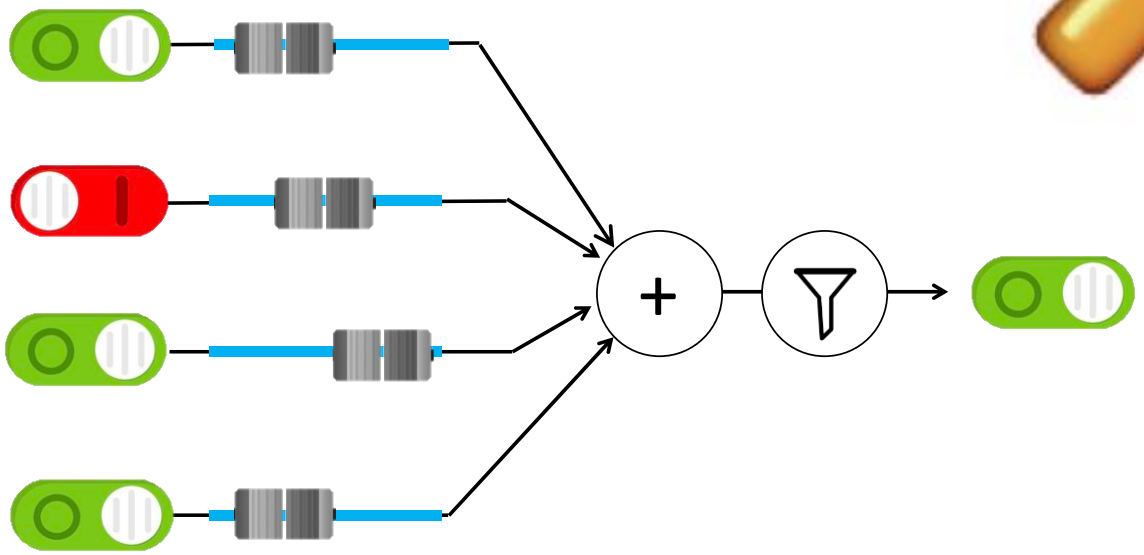


Dataset likelihood:



Artificial Neurons





Last Class: Comparing Classifiers

Comparing Classifiers: Test Accuracy

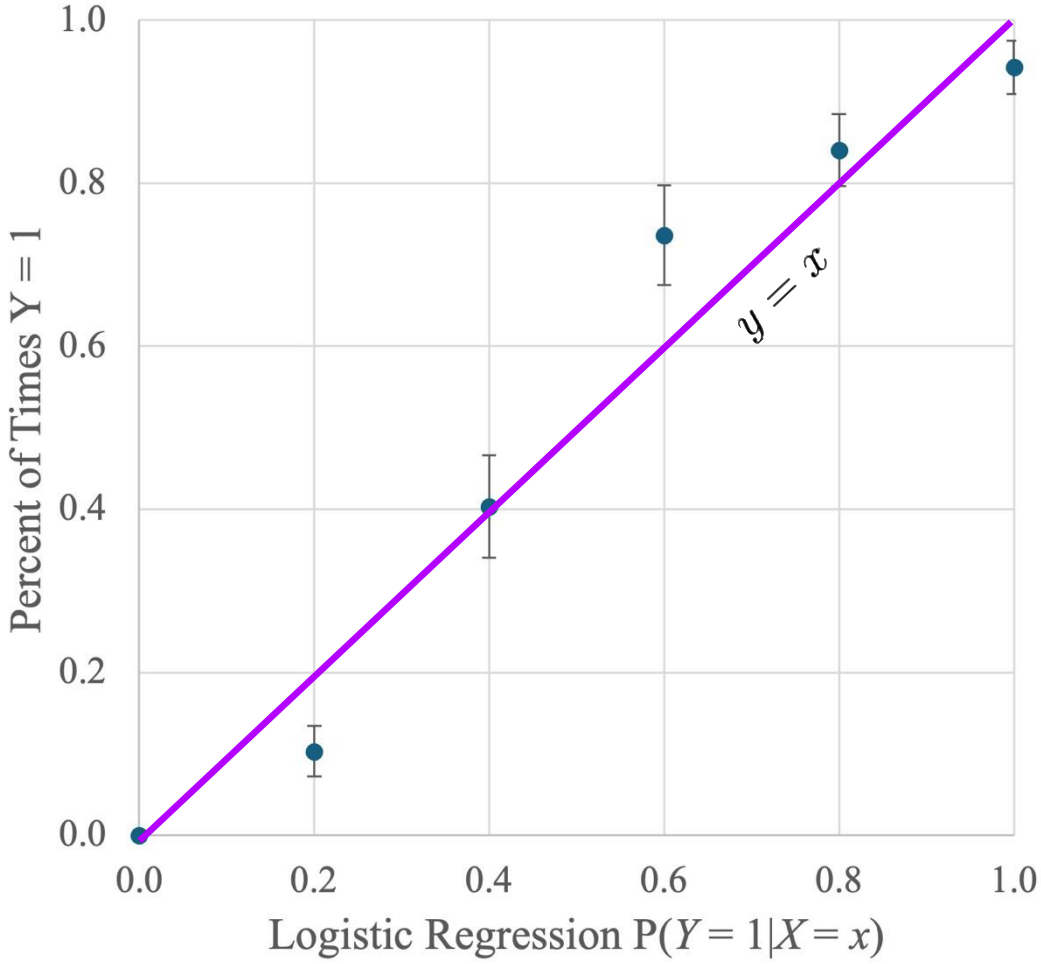
Model	Train Accuracy	Test Accuracy
Baseline	0.6031	0.5887
Naive Bayes	0.7909	0.8067
Logistic Regression	0.8169	0.8307
Decision Tree	0.8514	0.8307
Random Forest	0.8726	0.8500
Gradient Boosting	0.8611	0.8440
AdaBoost	0.8334	0.8353
BayesNet	0.8320	0.8507

Comparing Classifiers: Calibration

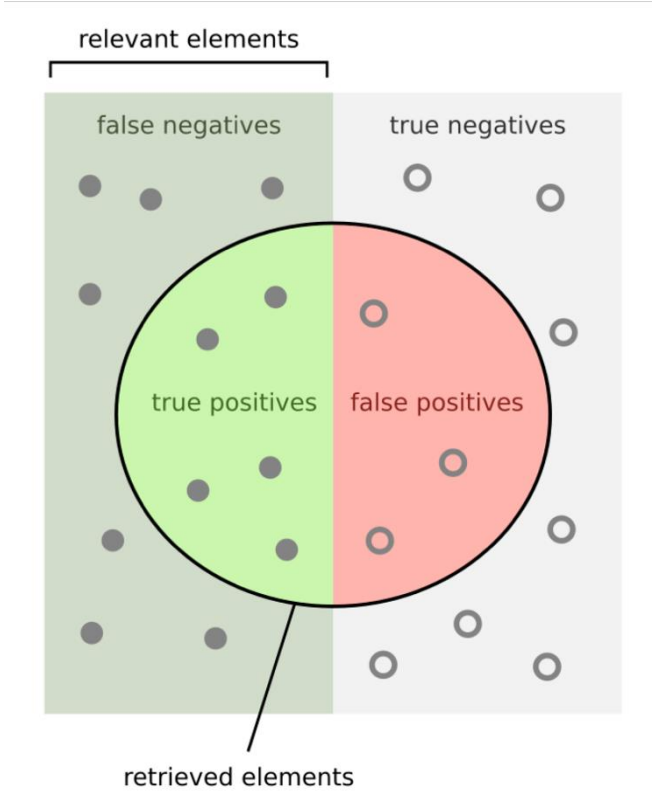
x_1 x_2 x_{19} y

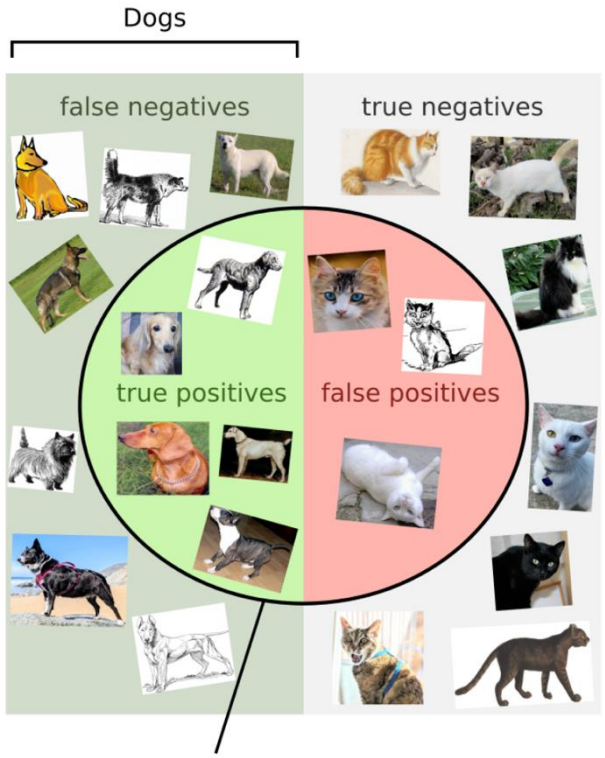
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

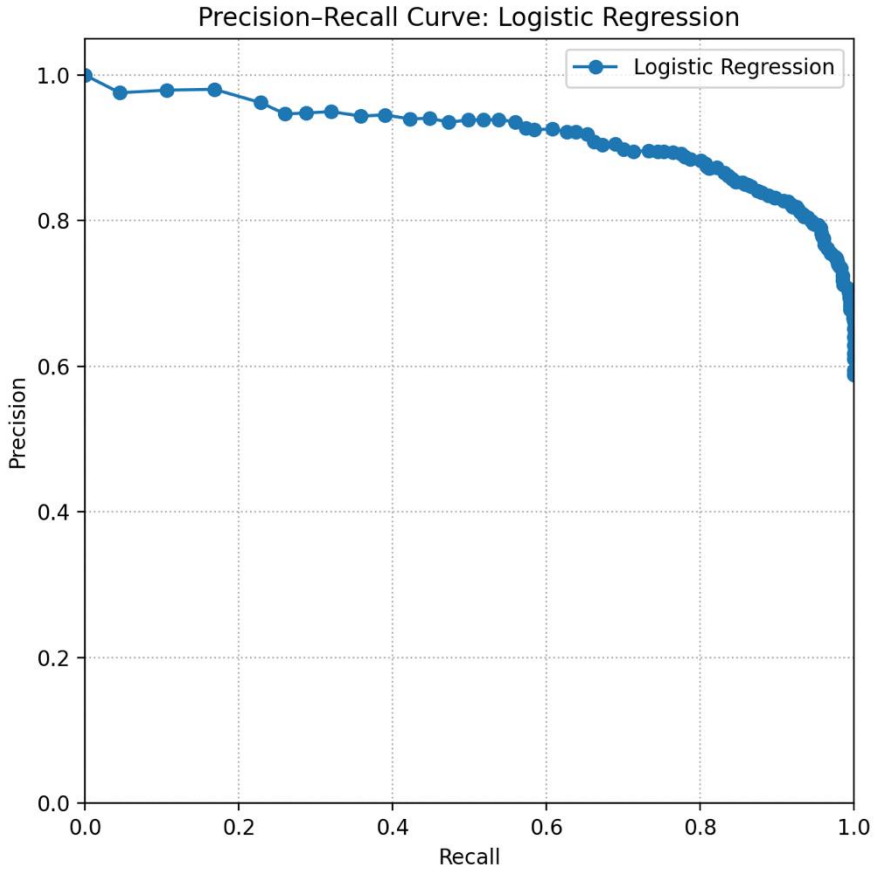
	A	B	T	U	V
1	col1	col2	col19	Label	LogRegPr
2	1	0	1	0	0.237
3	1	1	1	1	0.928
4	1	0	0	1	0.541
5	1	0	1	0	0.003
6	1	0	0	0	0.914
7	1	0	0	1	0.432
8	1	0	1	0	0.001
9	1	0	1	1	0.530
10	0	0	1	0	0.090
11	1	0	1	0	0.439
12	1	0	1	0	0.032
13	1	0	1	0	0.114
14	0	0	1	1	0.848
15	1	0	1	0	0.025
16	1	0	1	0	0.180
17	0	0	1	1	0.672
18	1	0	1	1	0.531
19	1	0	1	0	0.012
20	1	0	1	1	0.560
21	1	0	1	1	0.502



Comparing Classifiers: Precision / Recall Curve







How many retrieved items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{false positives} + \text{true positives}}$$

How many relevant items are retrieved?

$$\text{Recall} = \frac{\text{true positives}}{\text{false negatives} + \text{true positives}}$$

$$\text{Precision} = \frac{5 \text{ true pos.}}{8 \text{ total pos.}} \qquad \text{Recall} = \frac{5 \text{ true pos.}}{12 \text{ total dogs}}$$

$$\text{Prevalence} = \frac{12 \text{ total dogs}}{22 \text{ total images}}$$

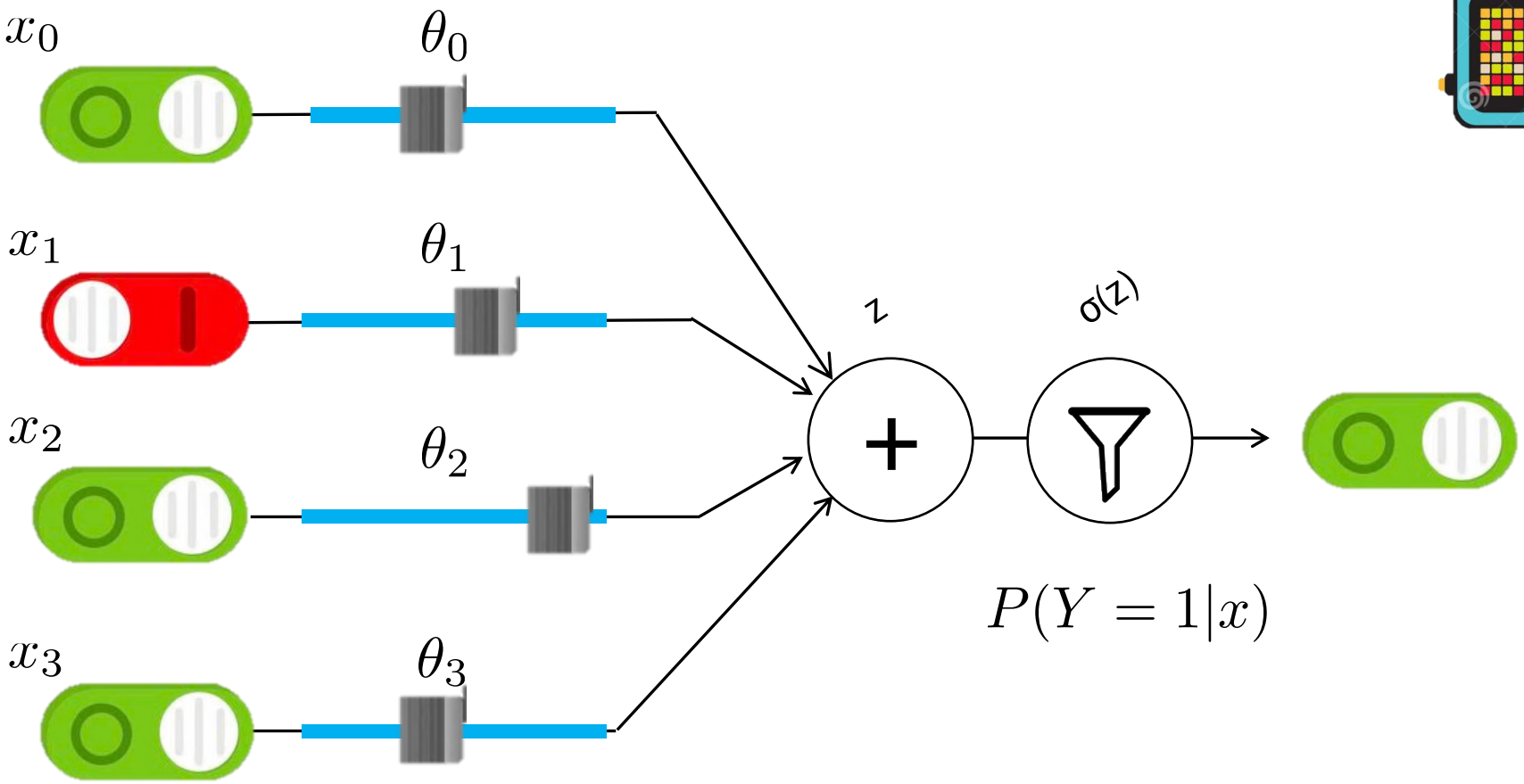
$$\text{Accuracy} = \frac{5 \text{ true pos.} + 7 \text{ true neg.}}{22 \text{ total images}}$$

Dog classifier. Image credit: wikipedia

End Review

Predicting a Categorical

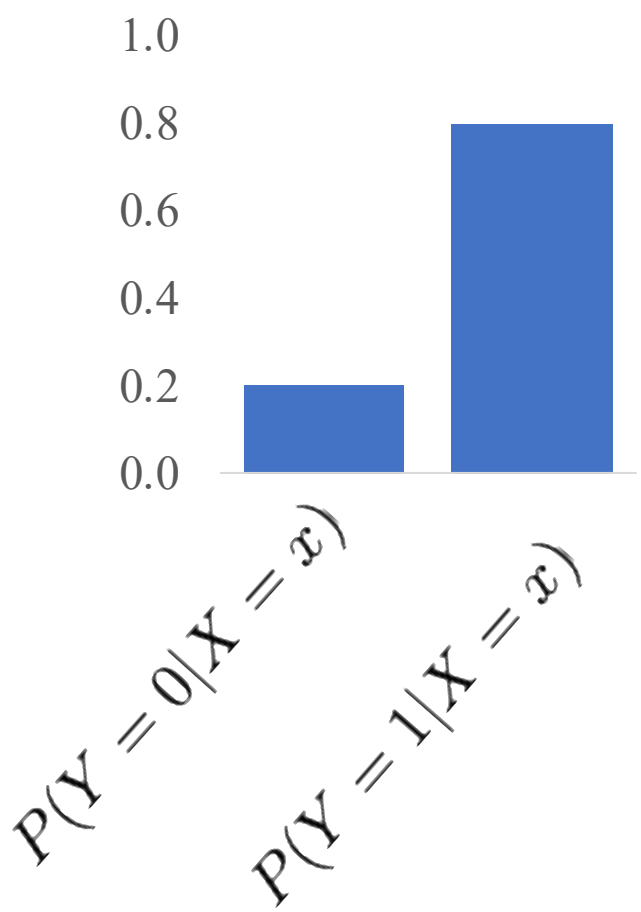
Logistic Regression to Predict a Categorical?



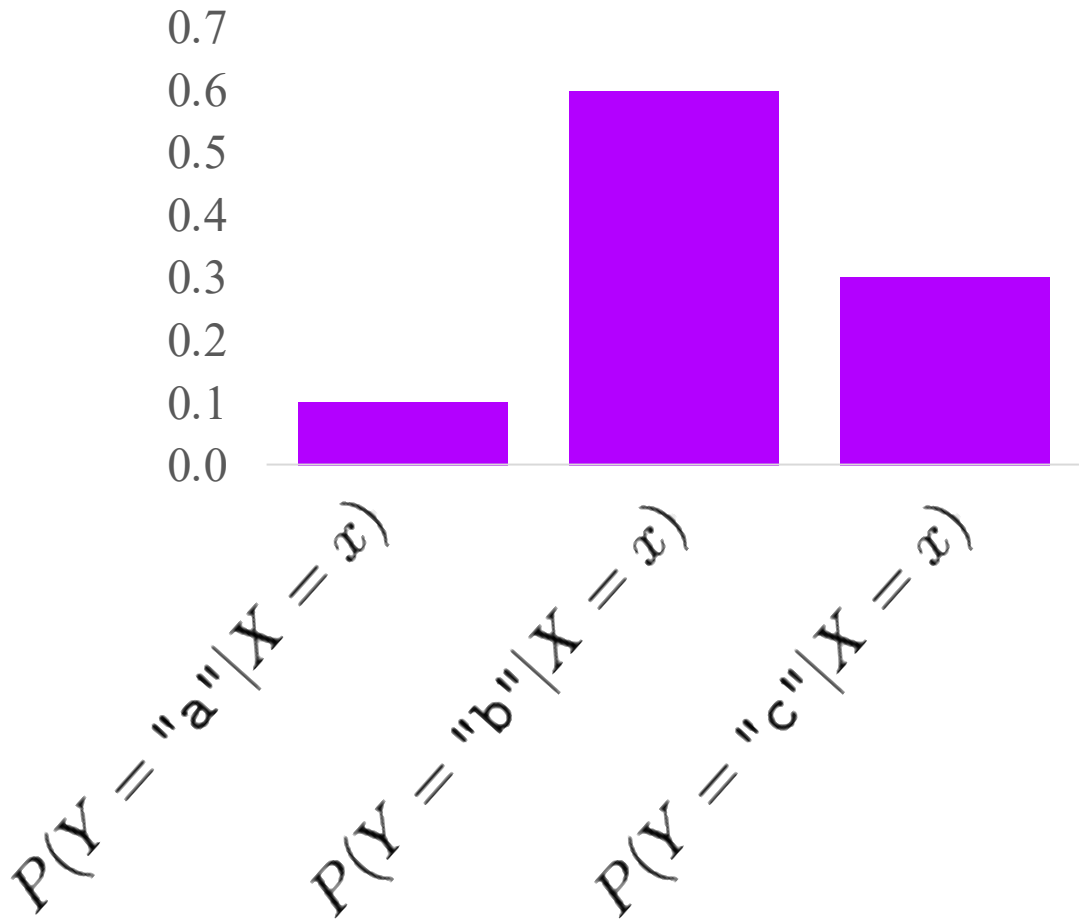
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\Big(\sum_i \theta_i x_i\Big)$$

Logistic Regression to Predict a Categorical?

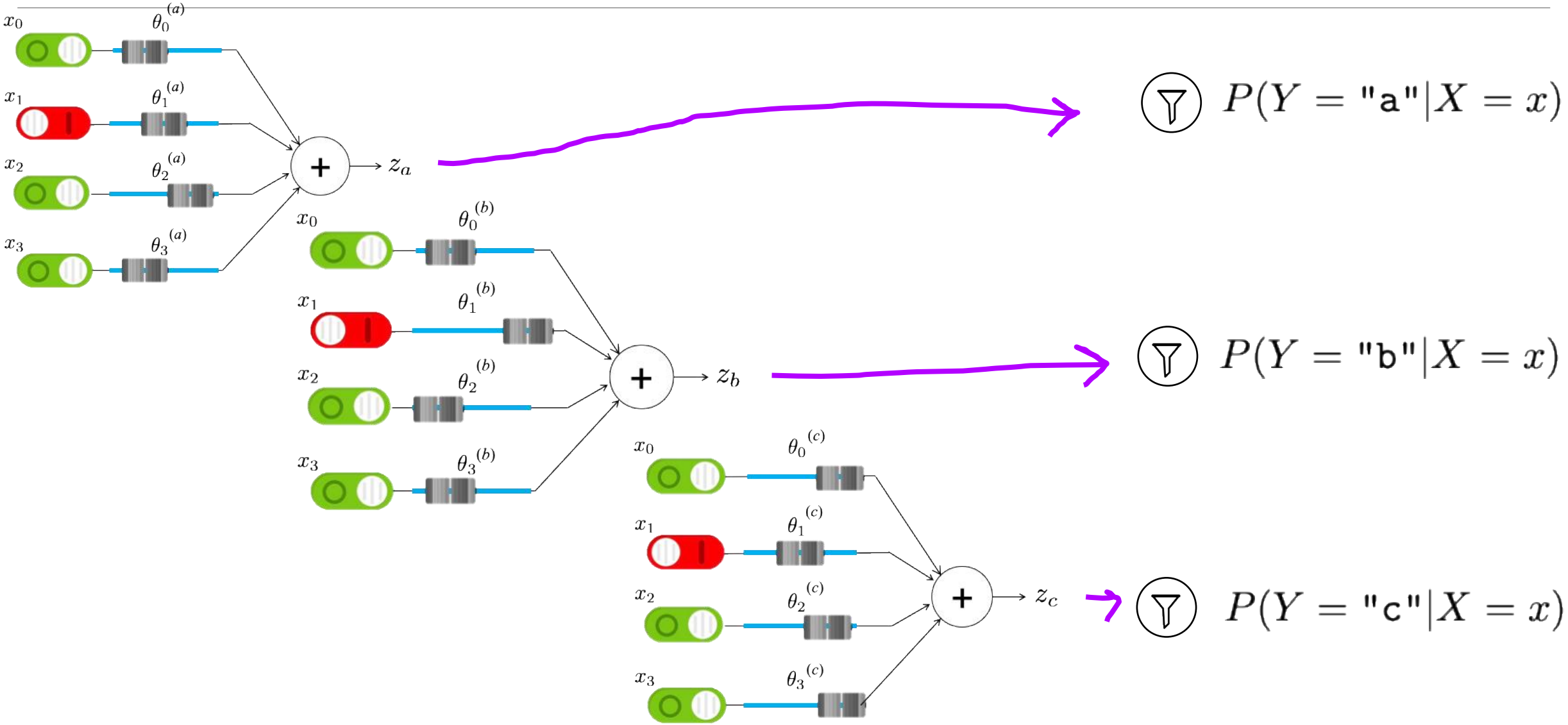
Standard Logistic Regression



Multi-Class Logistic Regression



Logistic Regression to Predict a Categorical



Categorical 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**.

$$P(Y = i|\mathbf{X} = \mathbf{x}) = \text{softmax}(\mathbf{z})_i$$

Sigmoid is to Bernoulli as Softmax is to Categorical

$$\text{softmax}(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_{j=1}^{n} \exp(z_j)}$$

Understanding Softmax

```
 9  def softmax(list_values):
10      """
11      Compute the softmax of a list of numbers.
12      >>> softmax([1, 2, 3])
13      [0.09, 0.24, 0.67]
14      >>> softmax([5, 2, 8])
15      [0.02, 0.00, 0.98]
16      """
17
18      # take the exp of each value in the list
19      raw_exp_values = []
20      for value in list_values:
21          raw_exp_values.append(math.exp(value))
22
23      # normalize the list
24      sum_exp_values = sum(raw_exp_values)
25      softmax_values = []
26      for value in raw_exp_values:
27          softmax_values.append(value / sum_exp_values)
28
29      return softmax_values
```

List: 7 7 7 7
[0.25, 0.25, 0.25, 0.25]

List: 2 8 5 6
[0.0, 0.84, 0.04, 0.11]

Why not this?

$$P(Y = i|X = x) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Softmax

$$P(Y = i|X = x) = \frac{\sigma(z_i)}{\sum_j \sigma(z_j)}$$

Normalized Logistic

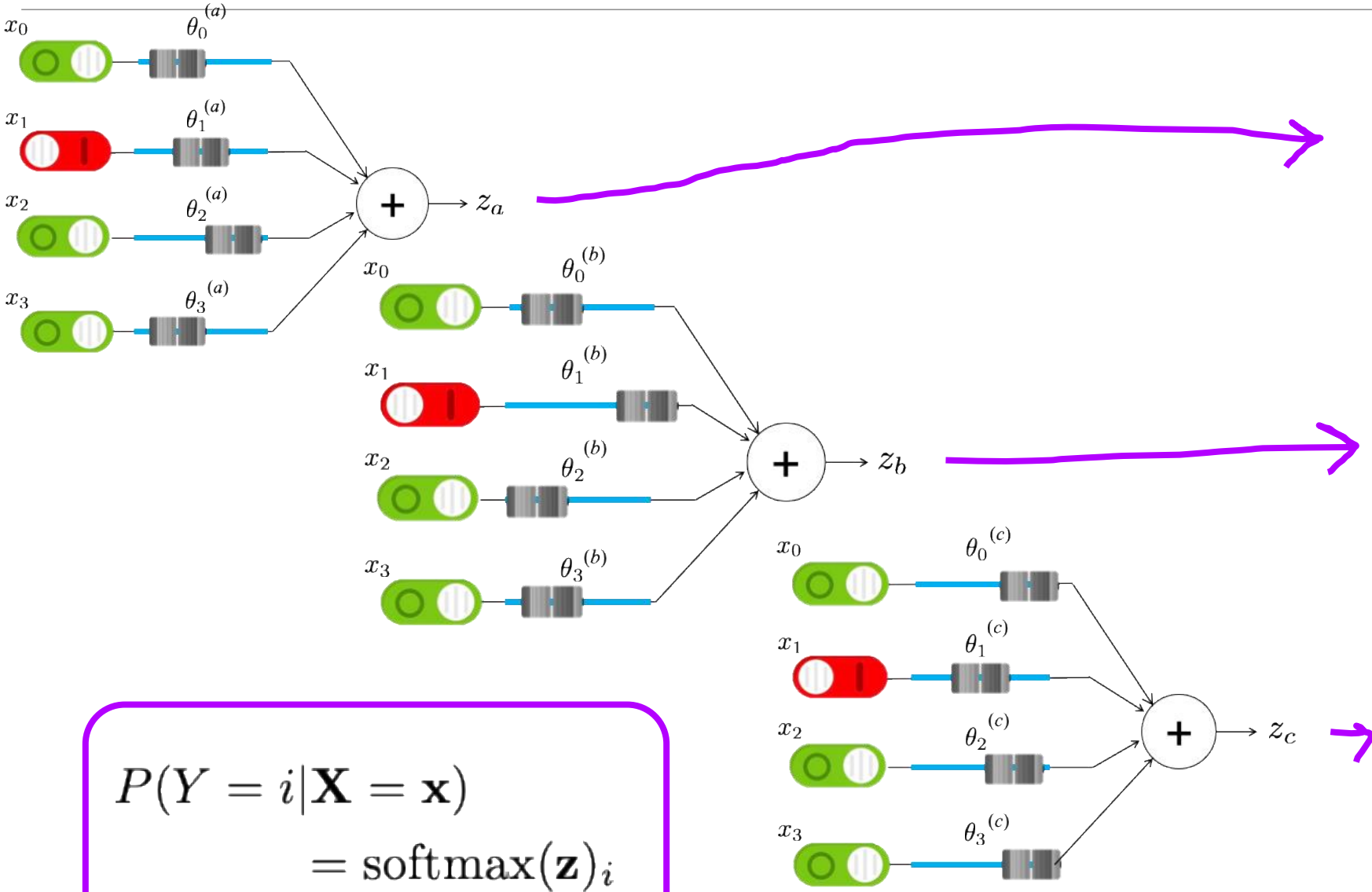
Softmax is Related to Sigma Function!

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

$$\text{softmax}(z_1, z_2) = \begin{bmatrix} \frac{e^{z_1}}{e^{z_1}+e^{z_2}} \\ \frac{e^{z_2}}{e^{z_1}+e^{z_2}} \end{bmatrix}$$

In softmax with two classes, $p_1 = \sigma(z_1 - z_2)$ $p_2 = \sigma(z_2 - z_1)$

What is Log Likelihood?



$$P(Y = \texttt{"a"} | X = x)$$

$$P(Y = \texttt{"b"} | X = x)$$

$$P(Y = \texttt{"c"} | X = x)$$

$$P(Y = i | \mathbf{X} = \mathbf{x}) = \text{softmax}(\mathbf{z})_i$$

What is Log Likelihood?

$$L(\theta) = \prod_i P(Y = y | X = x)$$

$$\begin{cases} P(Y = \text{'a'} | X = x) & \text{if truth is 'a'} \\ P(Y = \text{'b'} | X = x) & \text{if truth is 'b'} \\ P(Y = \text{'c'} | X = x) & \text{if truth is 'c'} \end{cases}$$

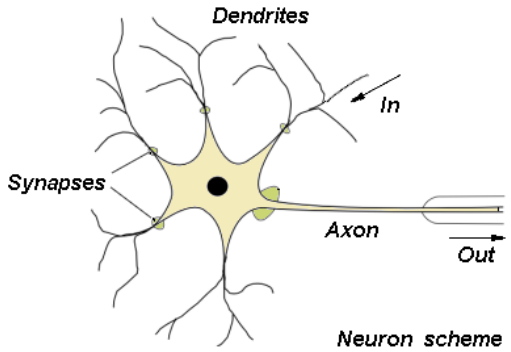
$$LL(\theta) = \sum_i \log P(Y = y | X = x)$$

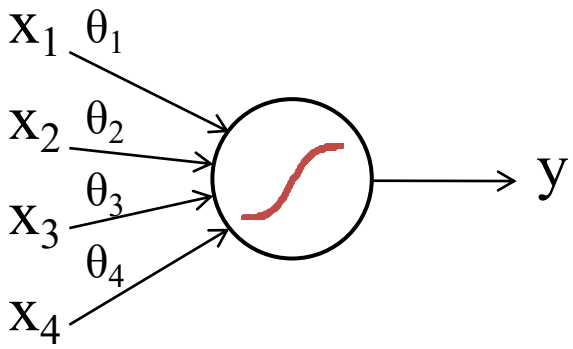
$$= \sum_i \log \text{softmax}(\mathbf{z})_y$$

We are ready…

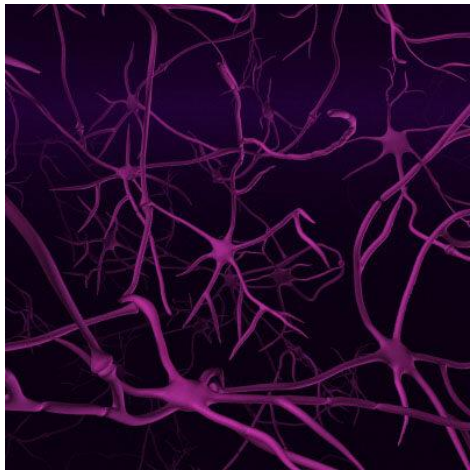
Biological Basis for Neural Networks

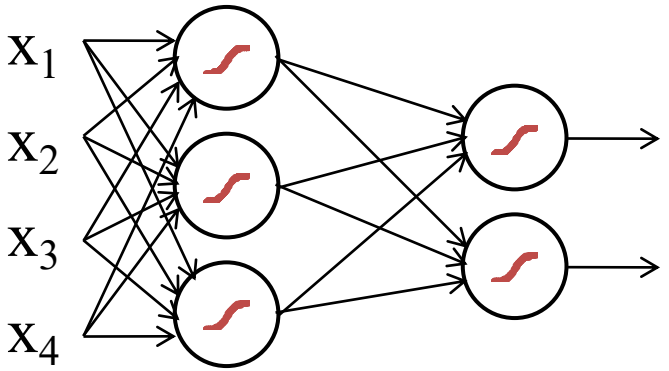
A neuron





Your brain





Actually, it's probably someone else's brain

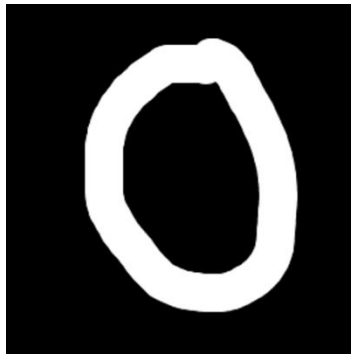
Core idea behind the revolution in AI

(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

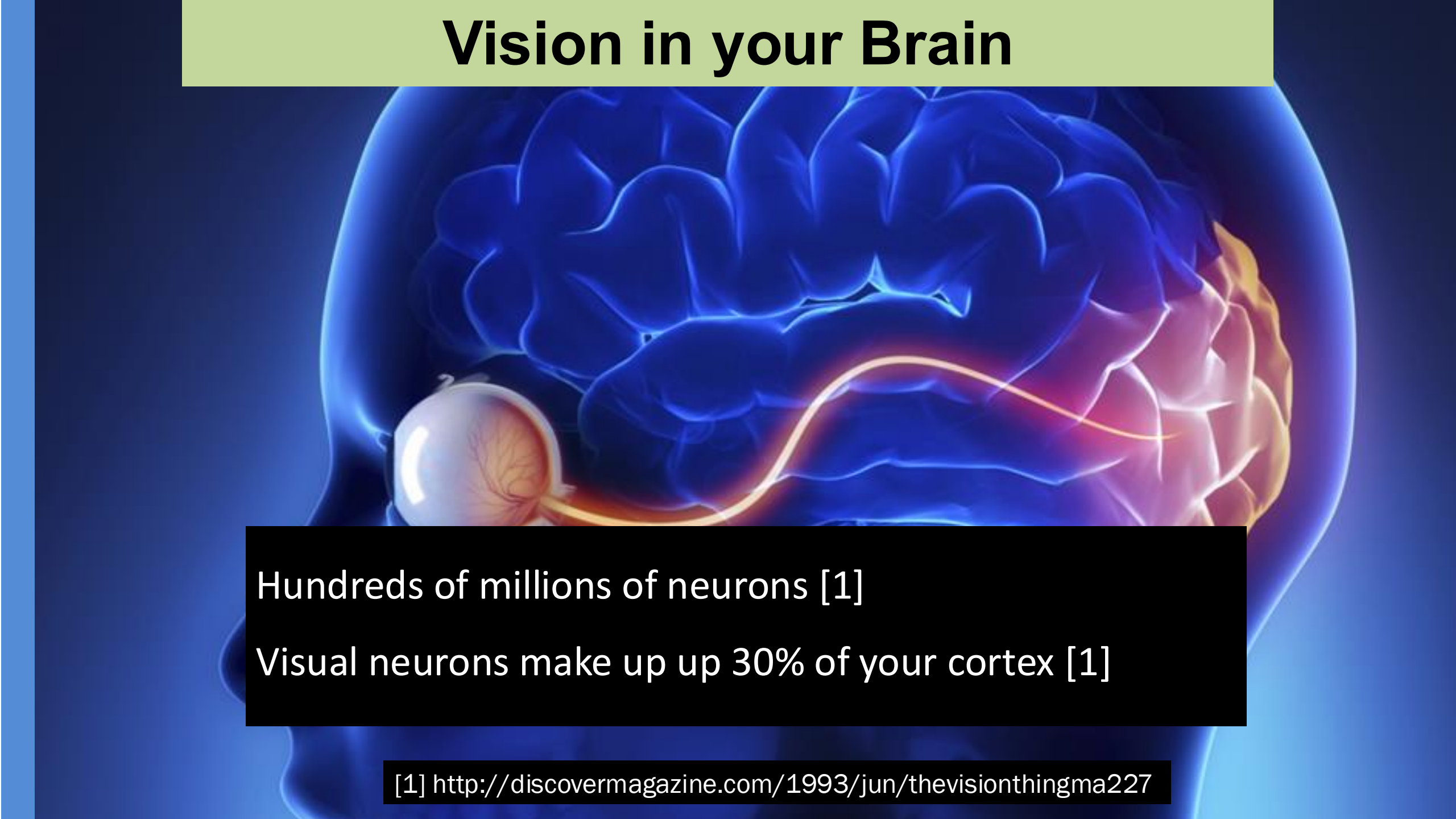


$$\mathbf{x}^{(i)} = [0, 0, 0, 0, \ldots, 1, 0, 0, 1, \ldots 0, 0, 1, 0]$$
$$y^{(i)} = 0$$

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

Computer Vision

Vision in your Brain

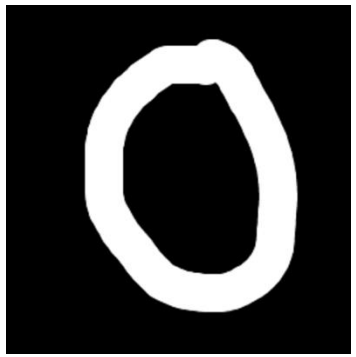


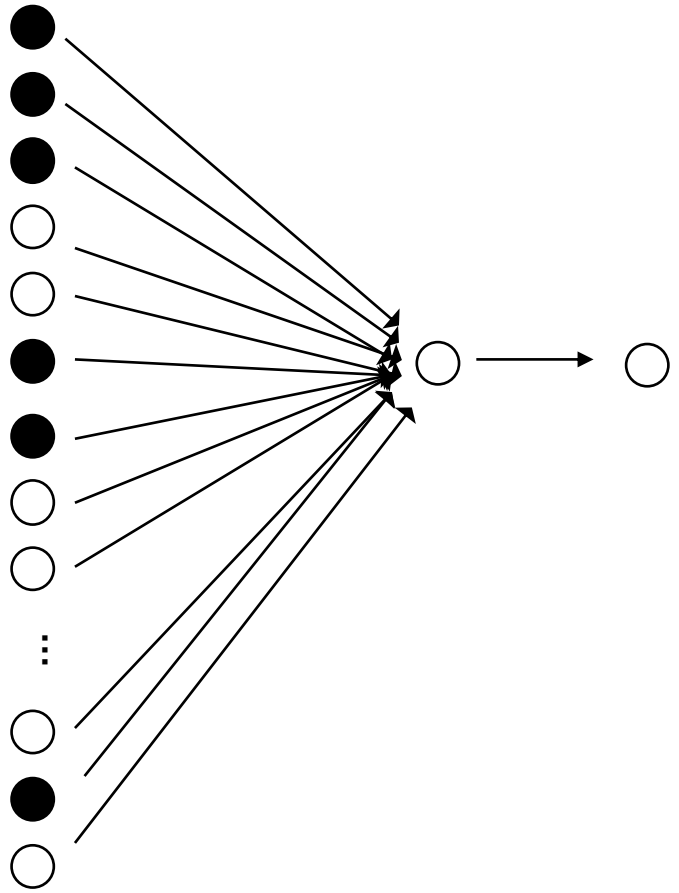
Hundreds of millions of neurons [1]

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

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

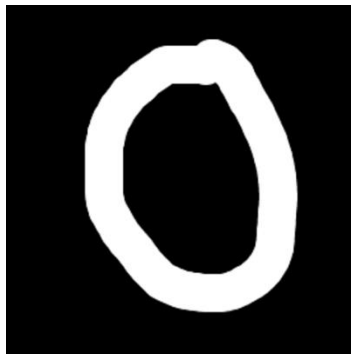
Logistic Regression

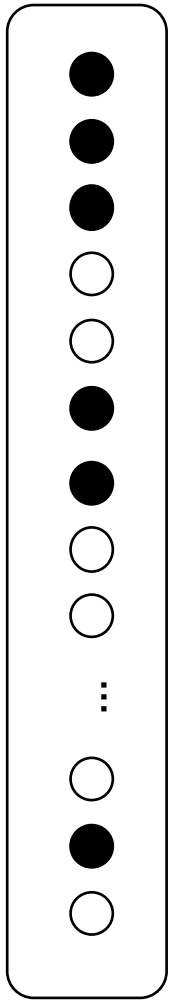




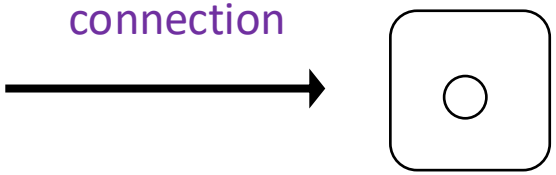
This means it predicts a 0

Logistic Regression



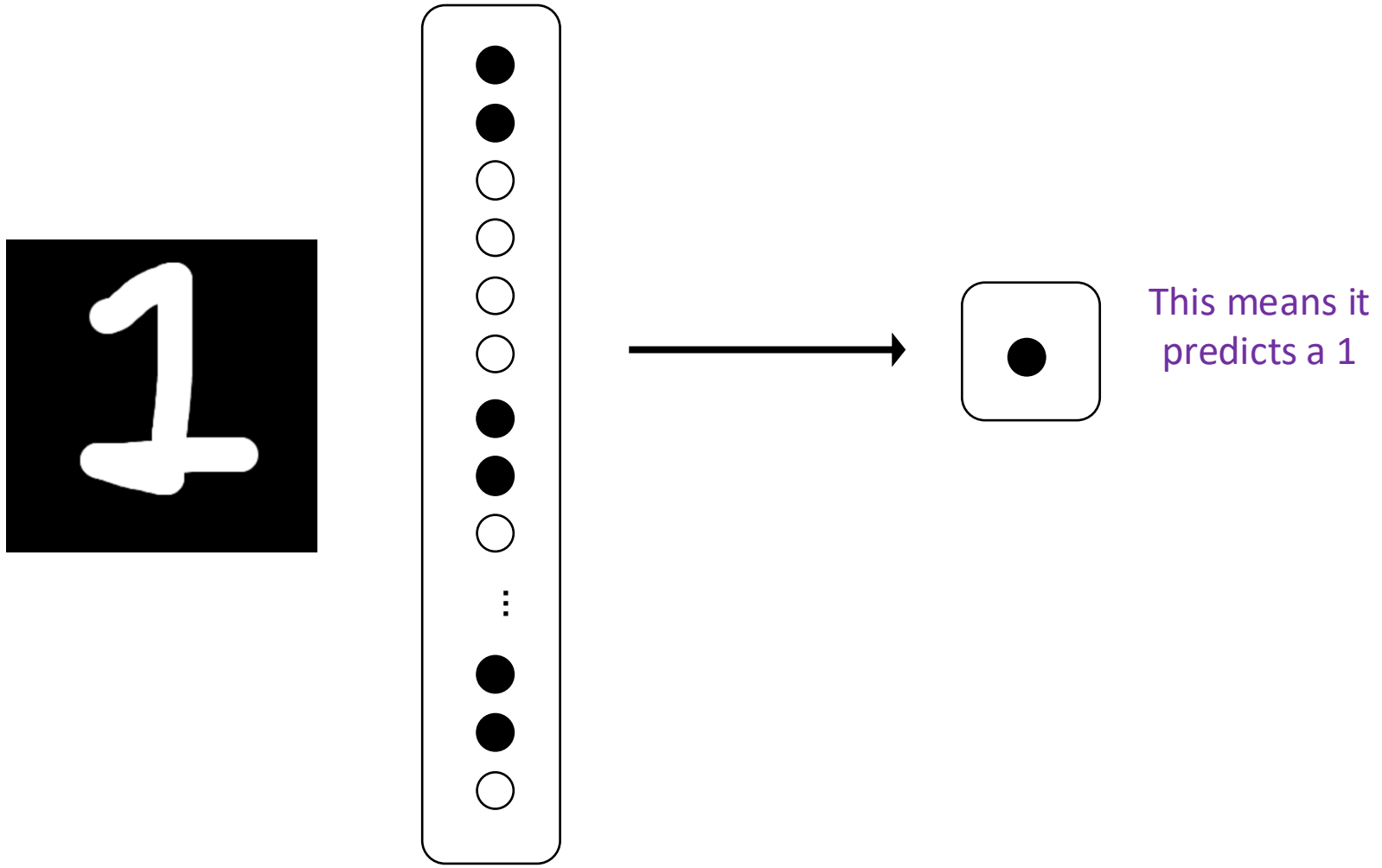


Indicates logistic regression connection

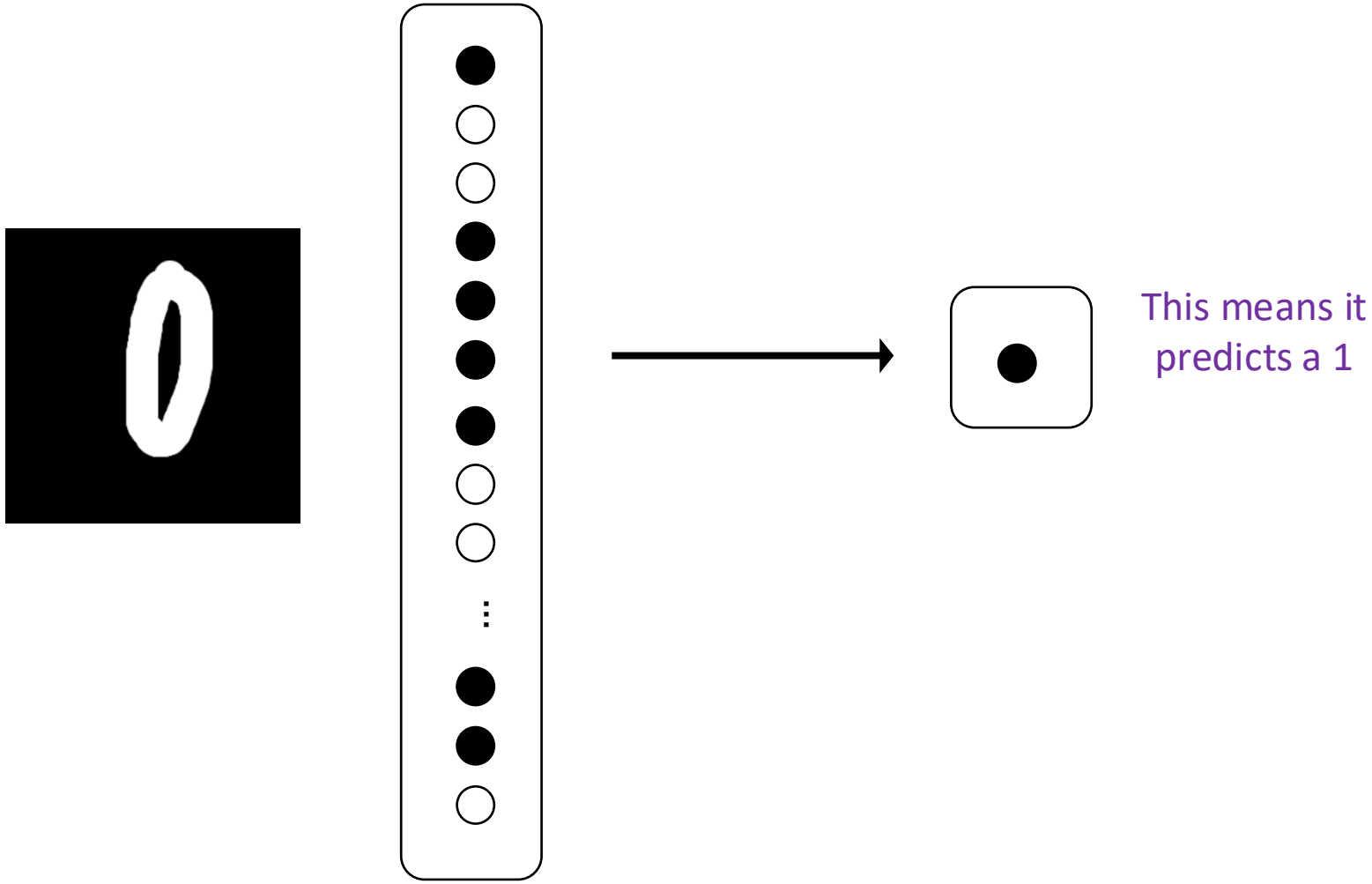


This means it predicts a 0

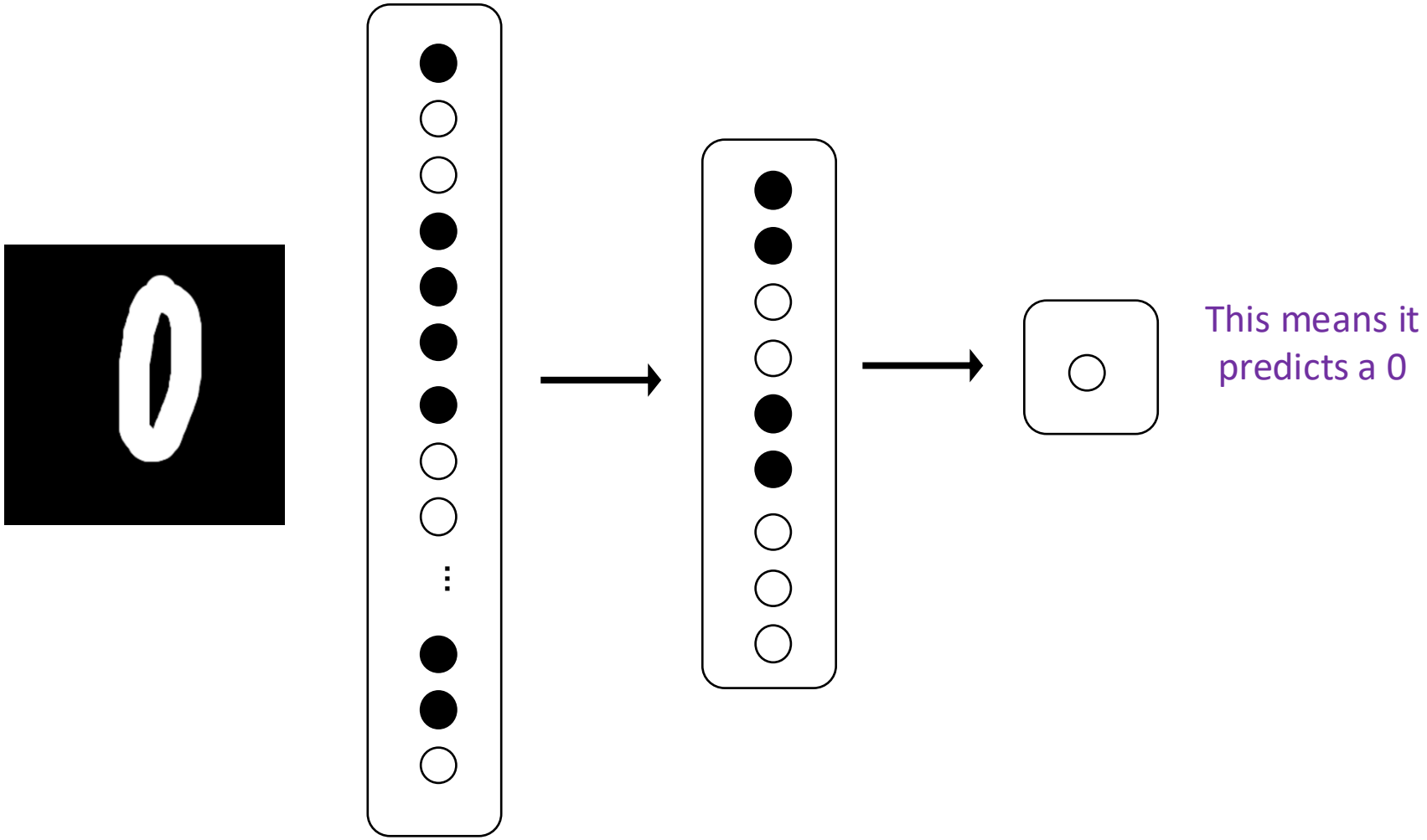
Logistic Regression



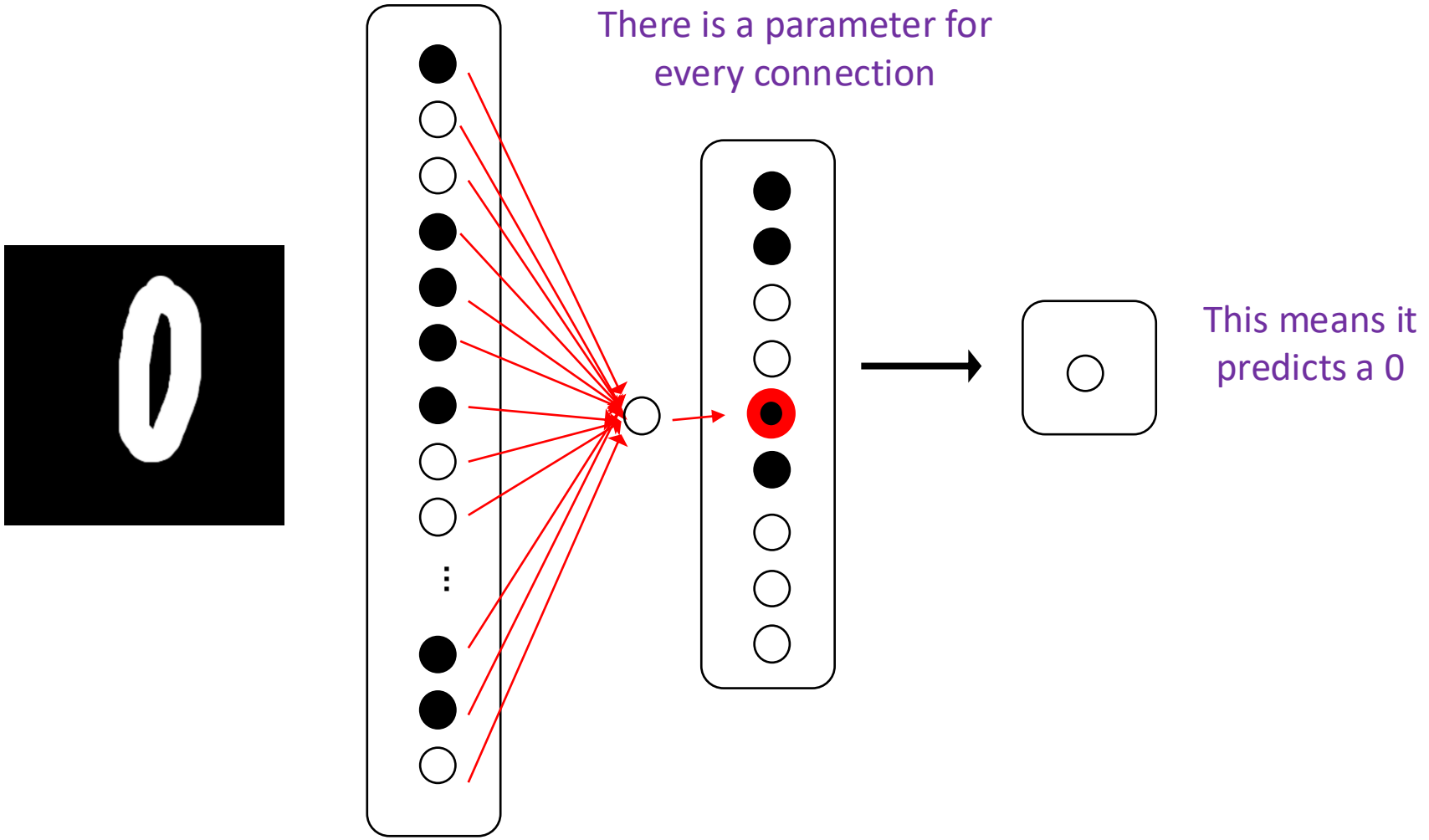
Not So Good



We Can Put Neurons Together

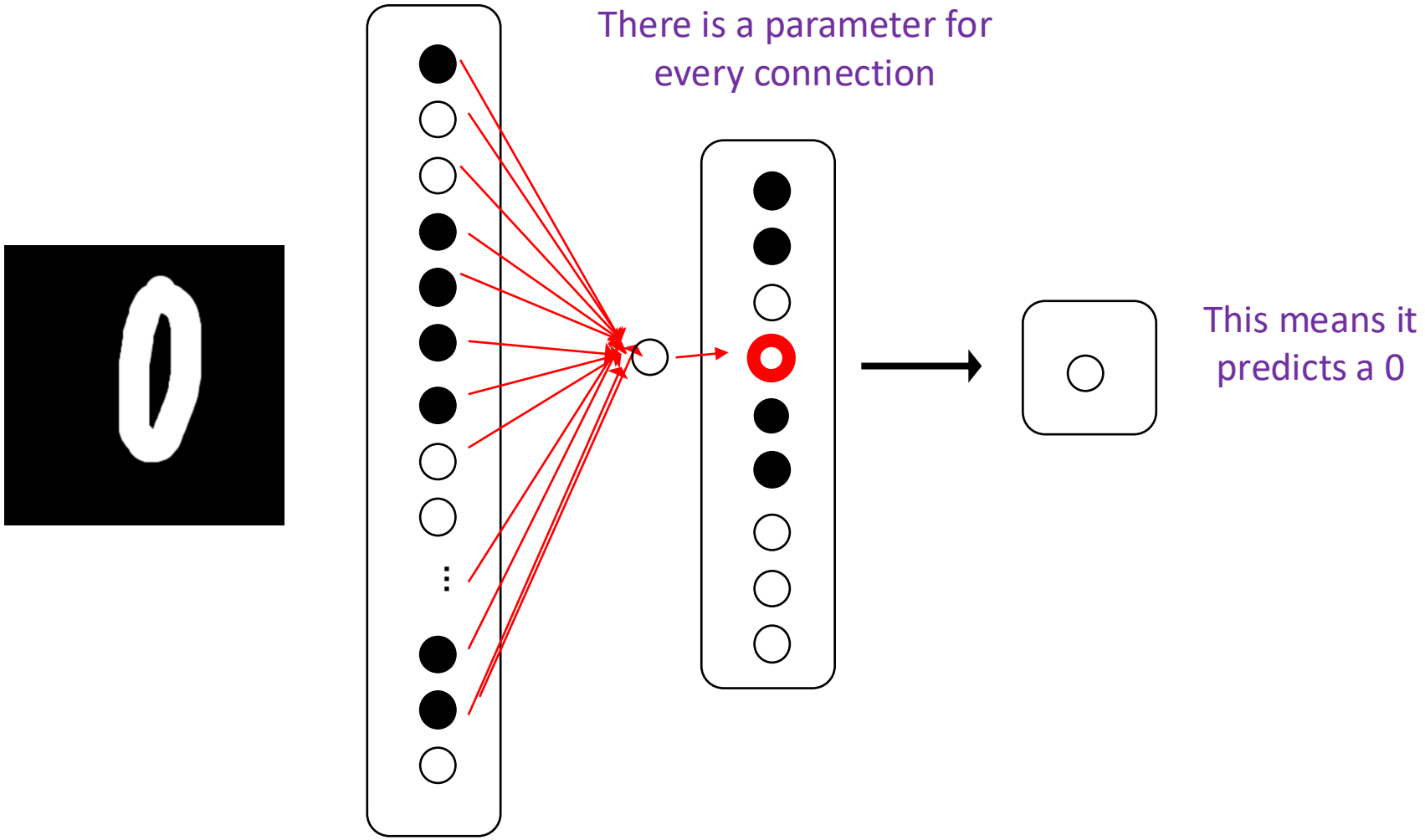


We Can Put Neurons Together



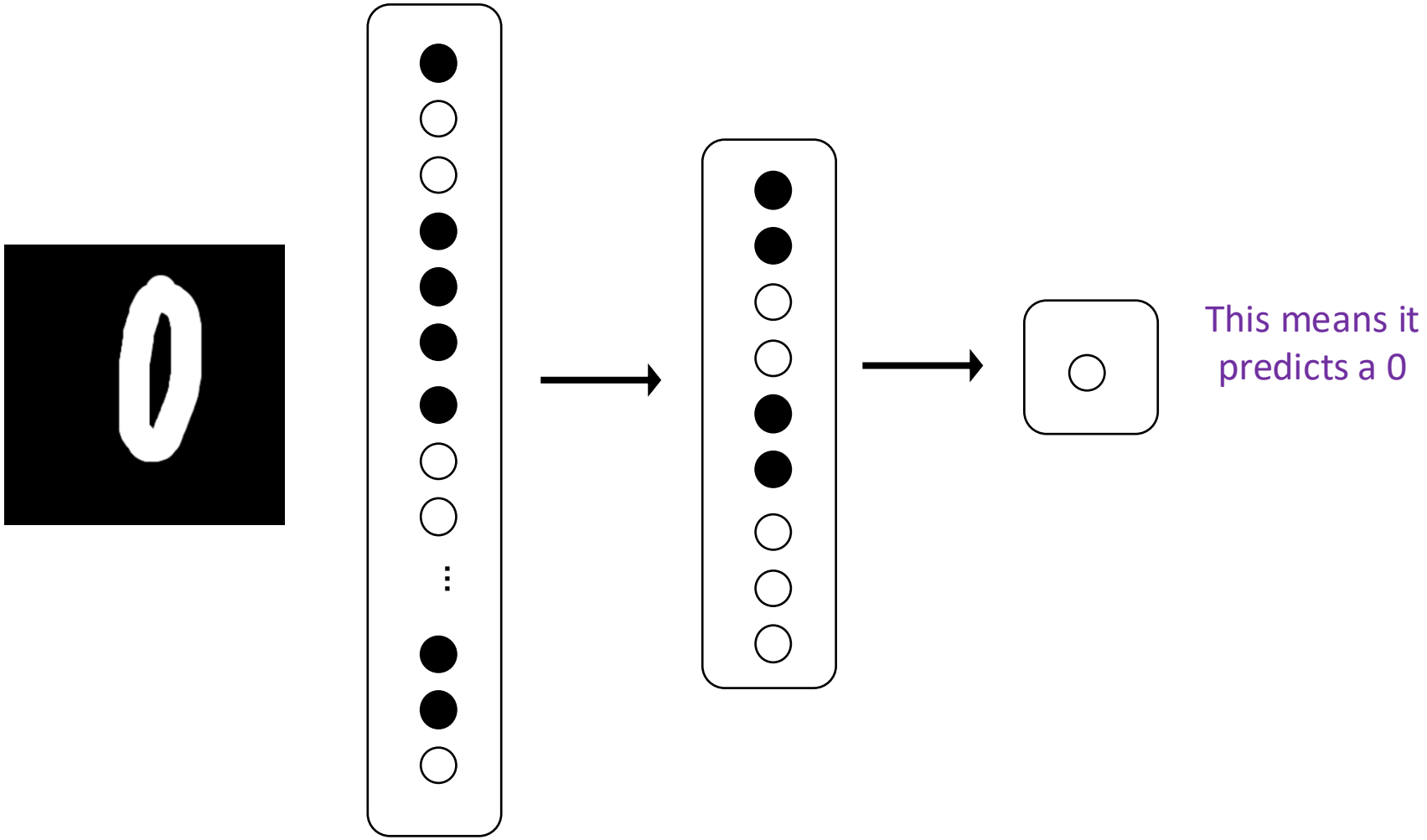
Look at a single "hidden" neuron

We Can Put Neurons Together

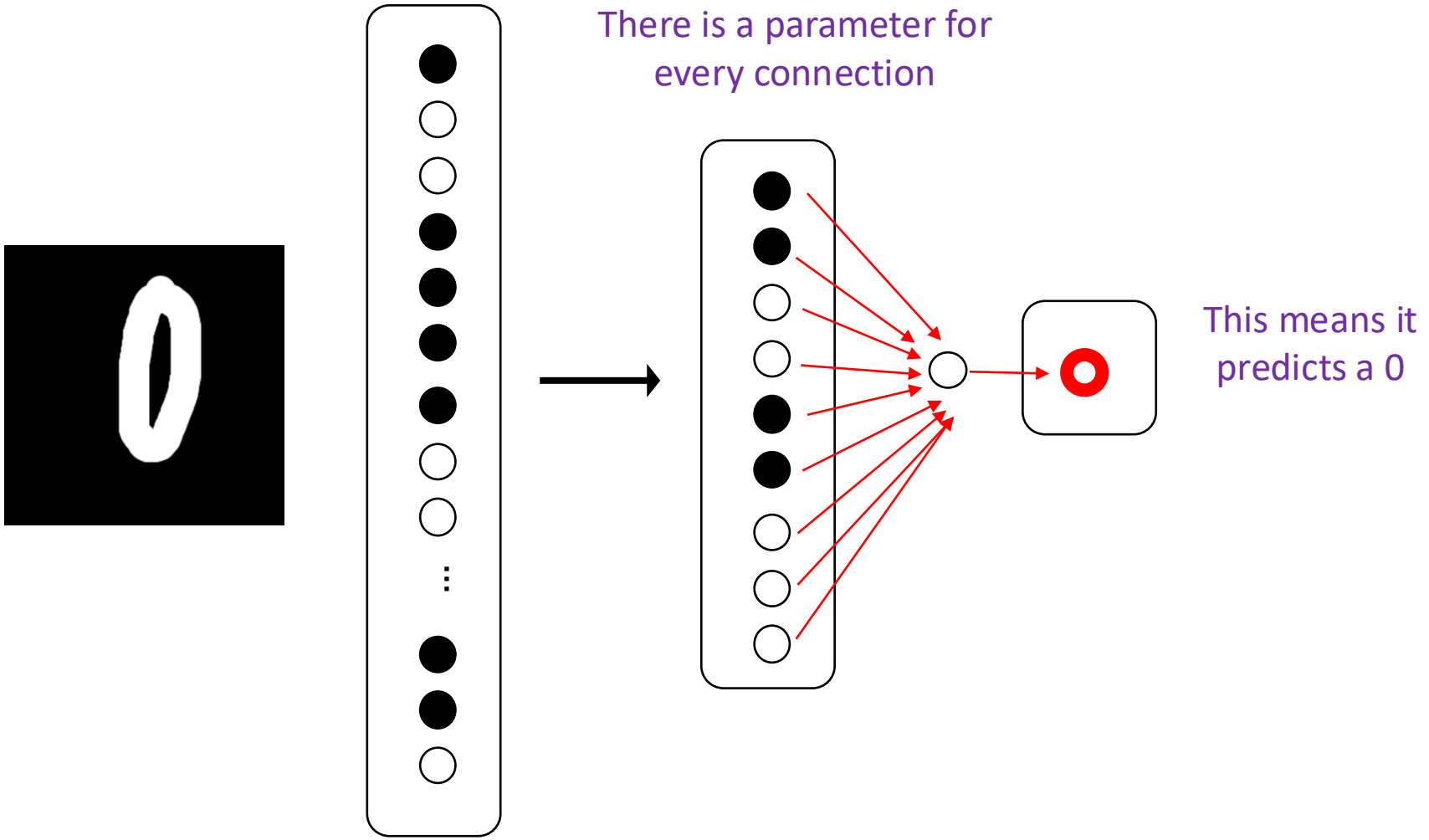


Look at another "hidden" neuron

We Can Put Neurons Together

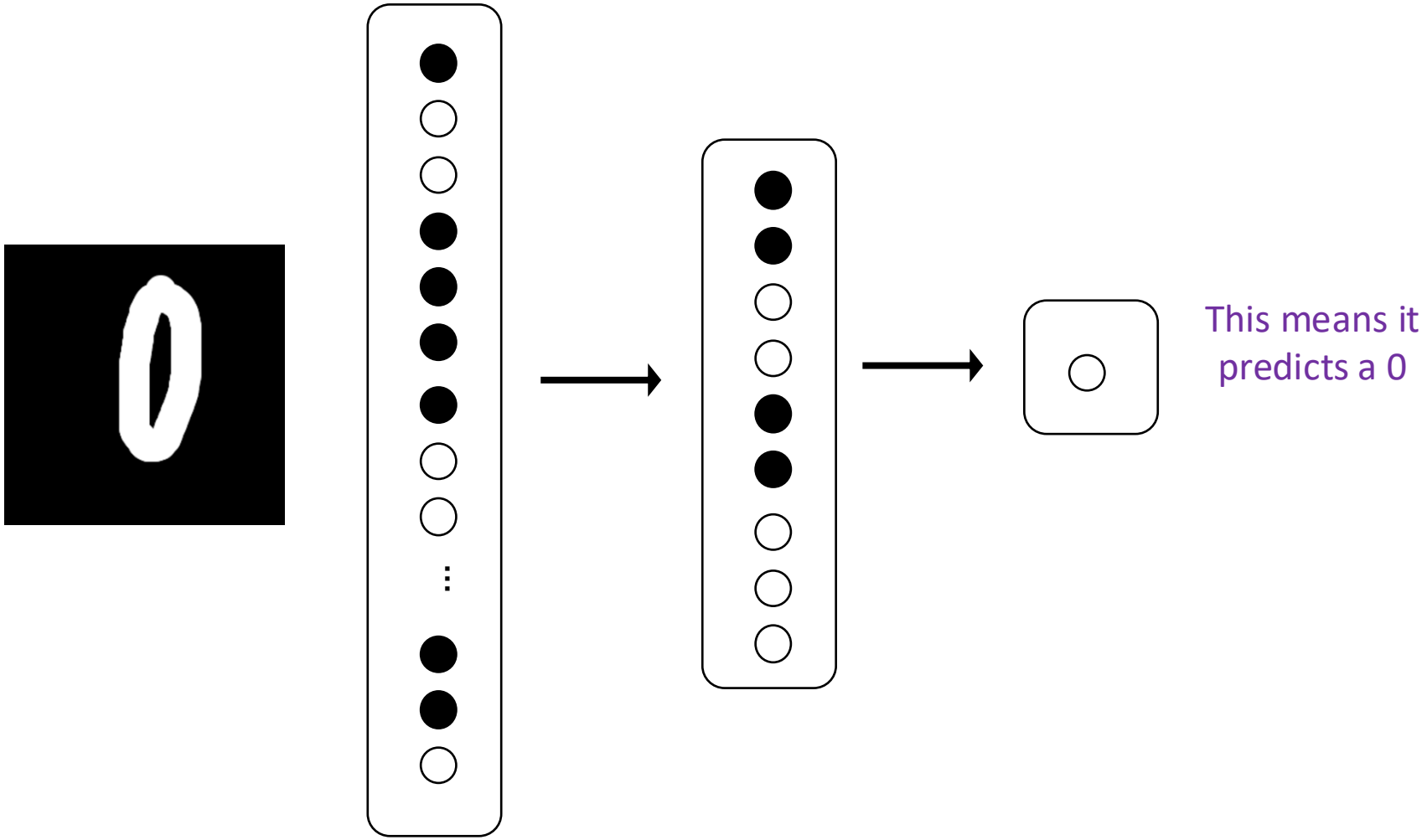


We Can Put Neurons Together

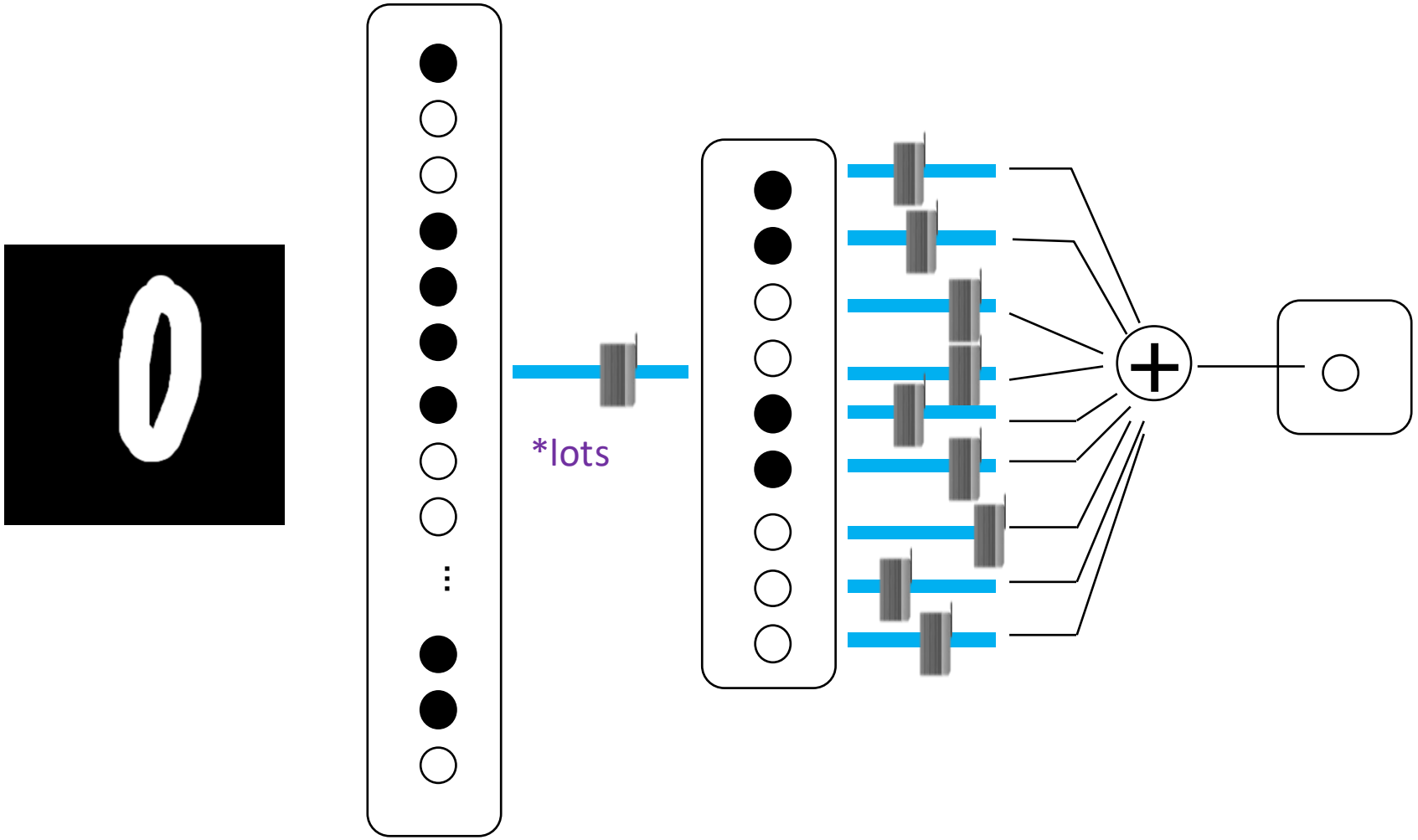


Look at another neuron

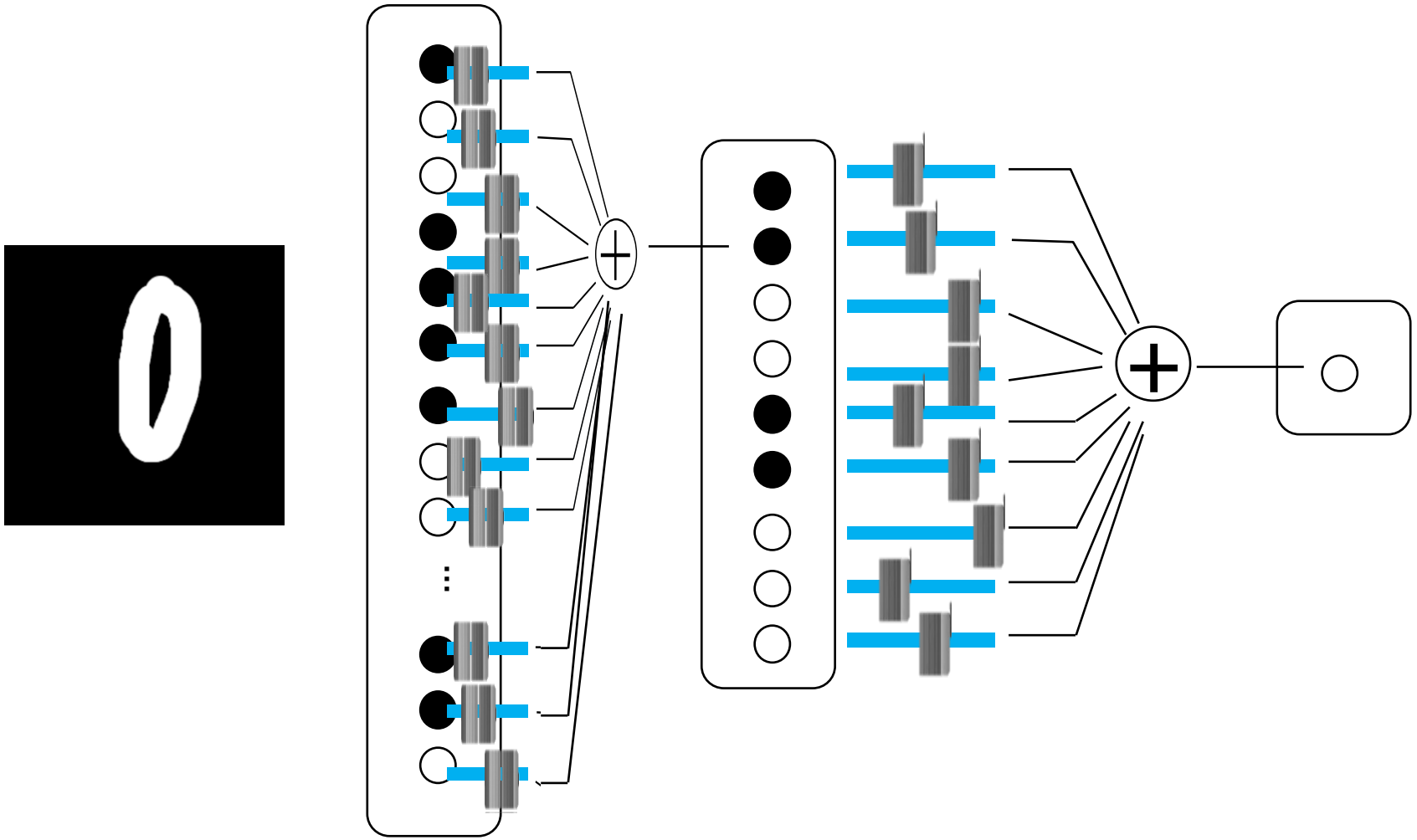
We Can Put Neurons Together



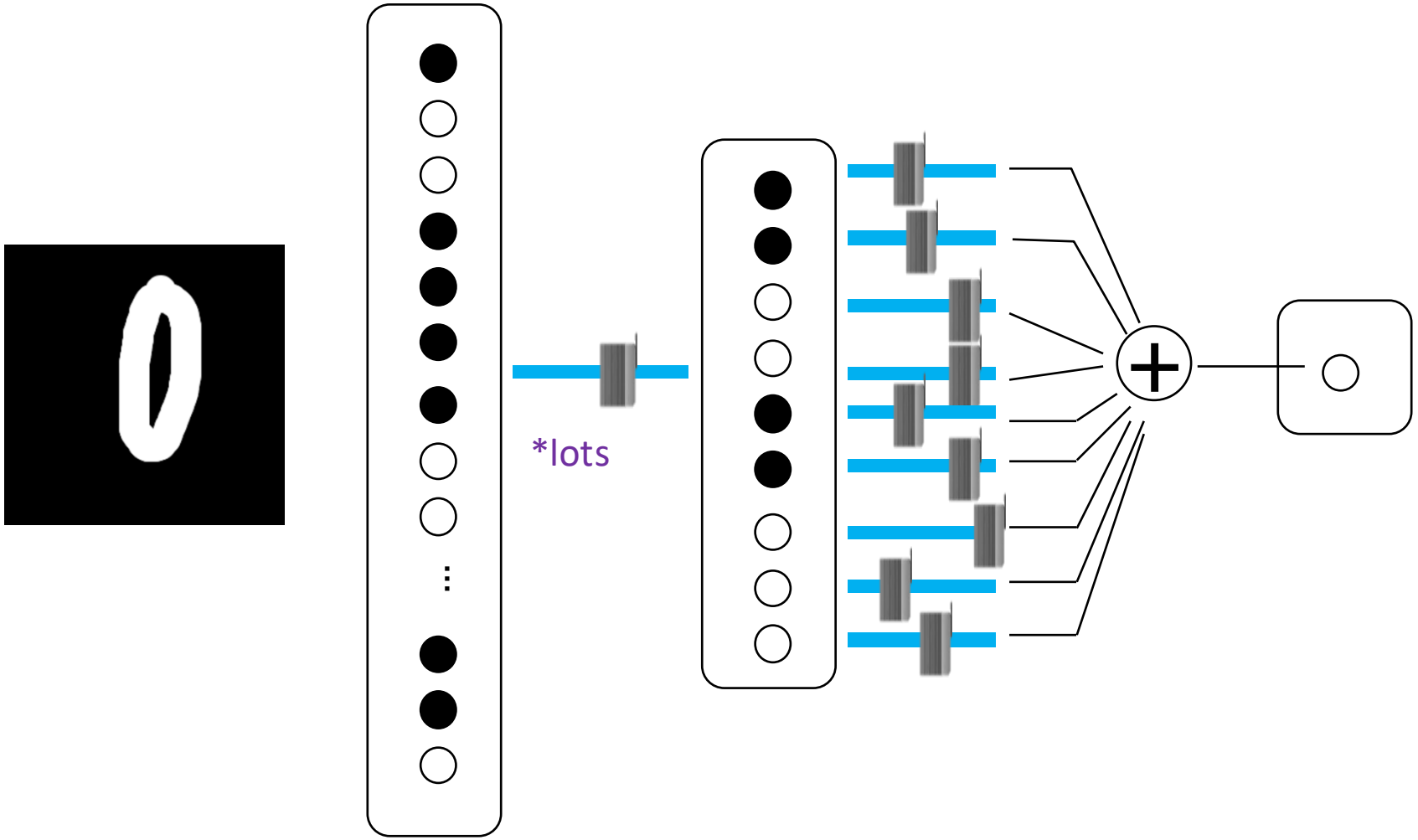
We Can Put Neurons Together



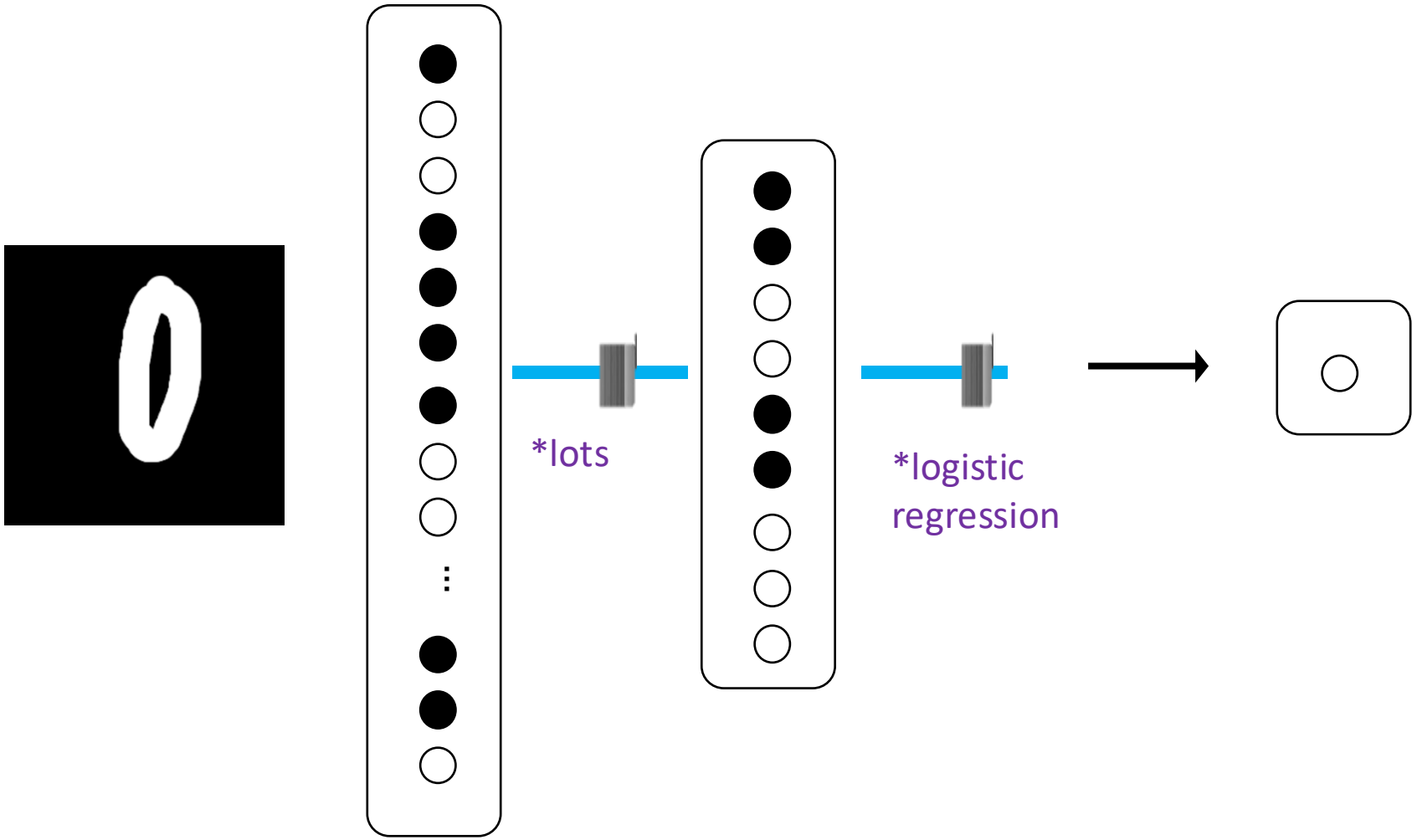
We Can Put Neurons Together



We Can Put Neurons Together



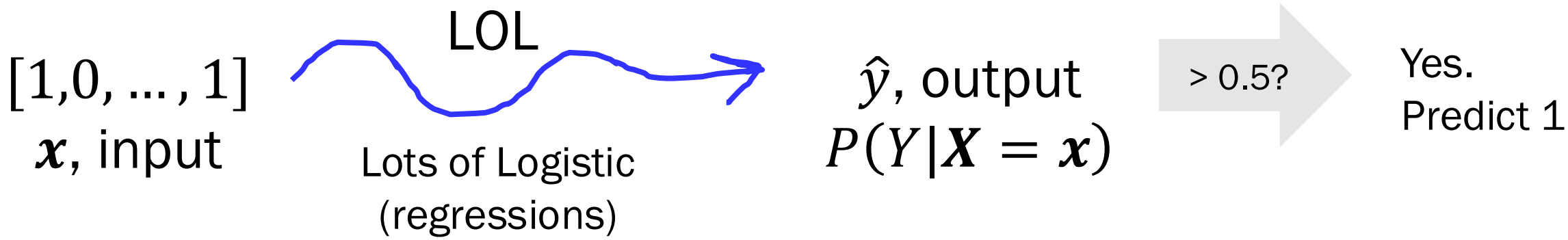
We Can Put Neurons Together



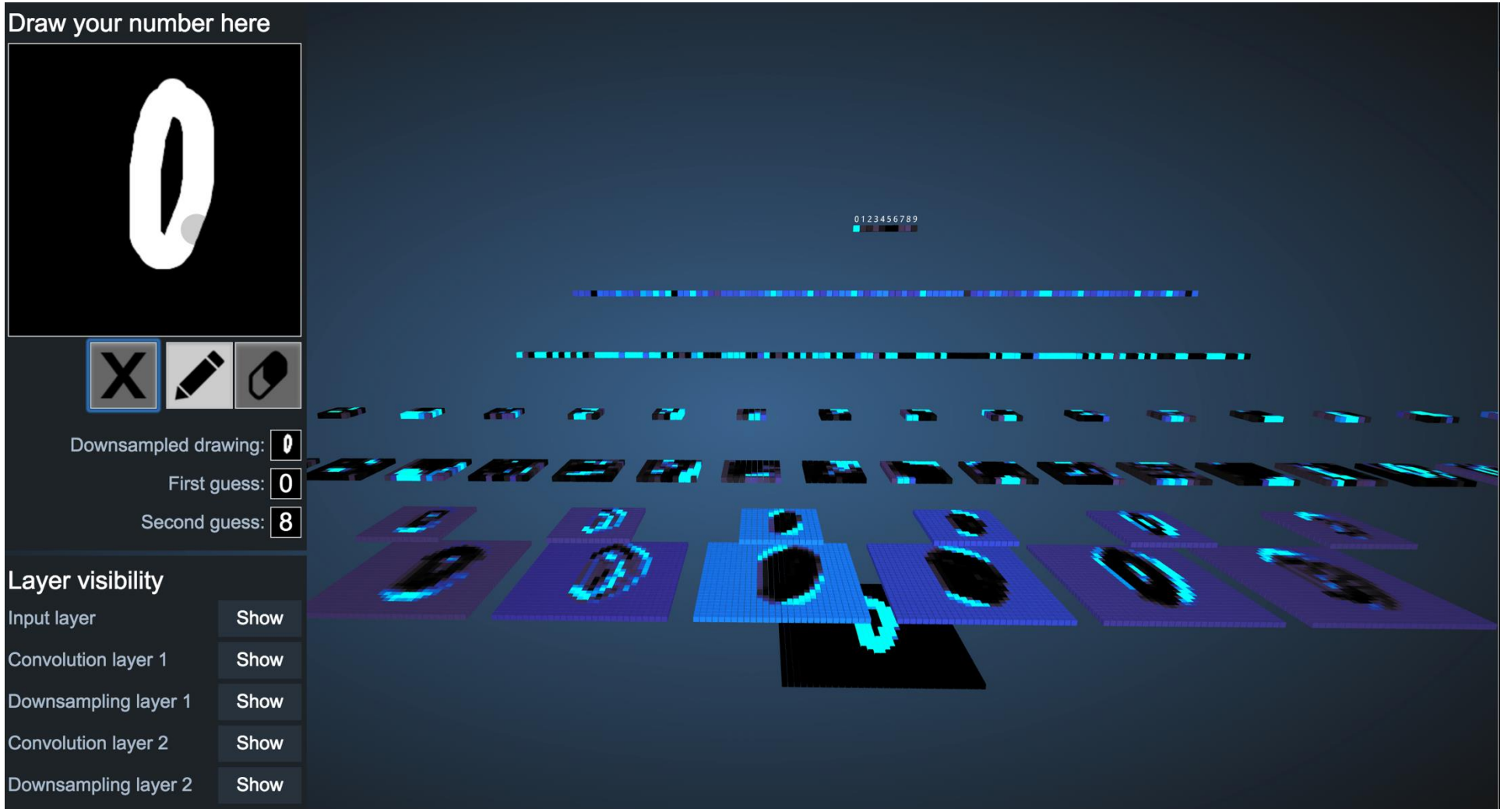
Deep learning

def **Deep learning** is
maximum likelihood estimation
with neural networks.

def A **neural network** is
(at its core) many logistic
regression pieces stacked on
top of each other.



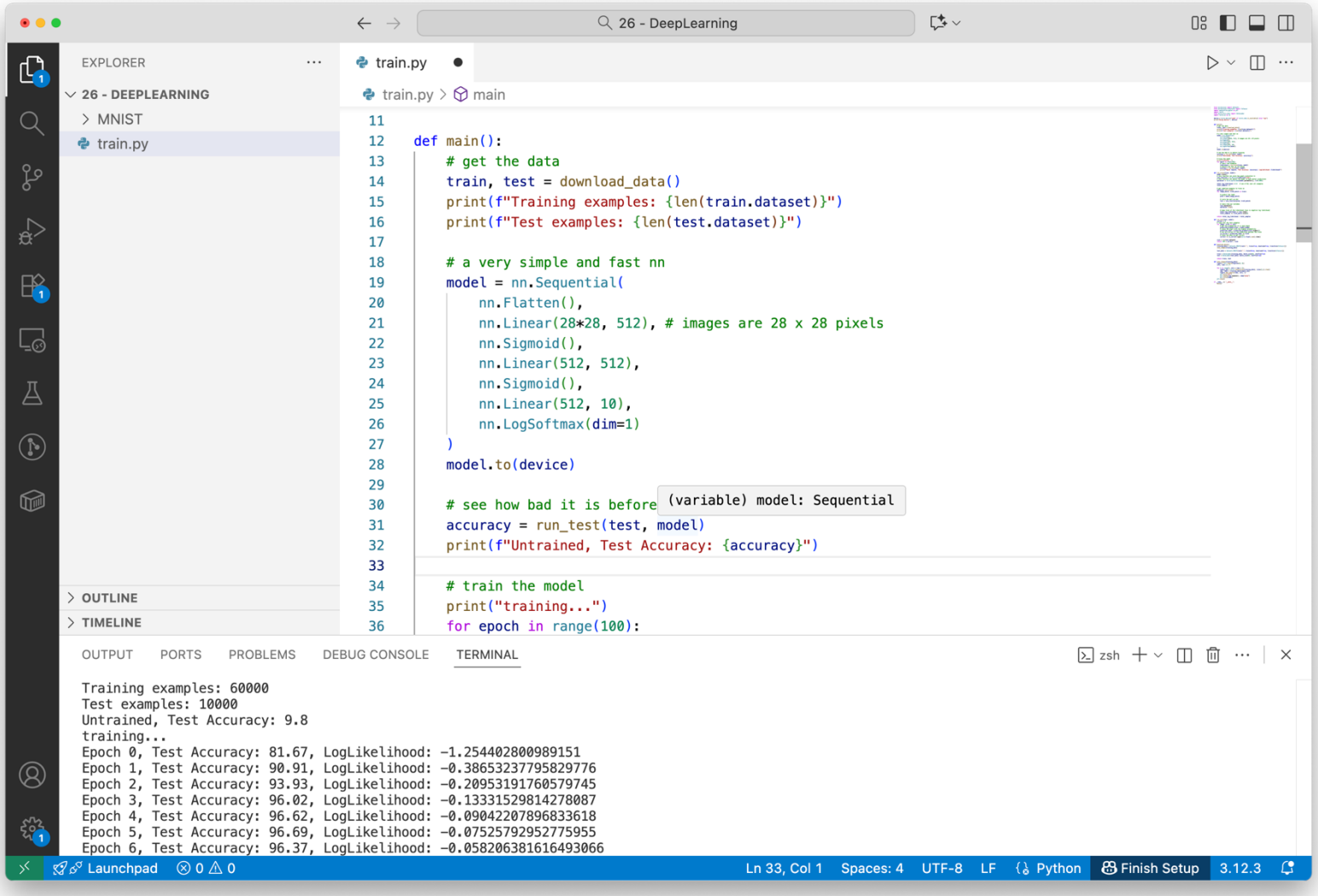
Demonstration



https://adamharley.com/nn_vis/cnn/2d.html

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

Lets Build a Neural Network!



How do we train?

MLE of Thetas!

First: Learning Goals…

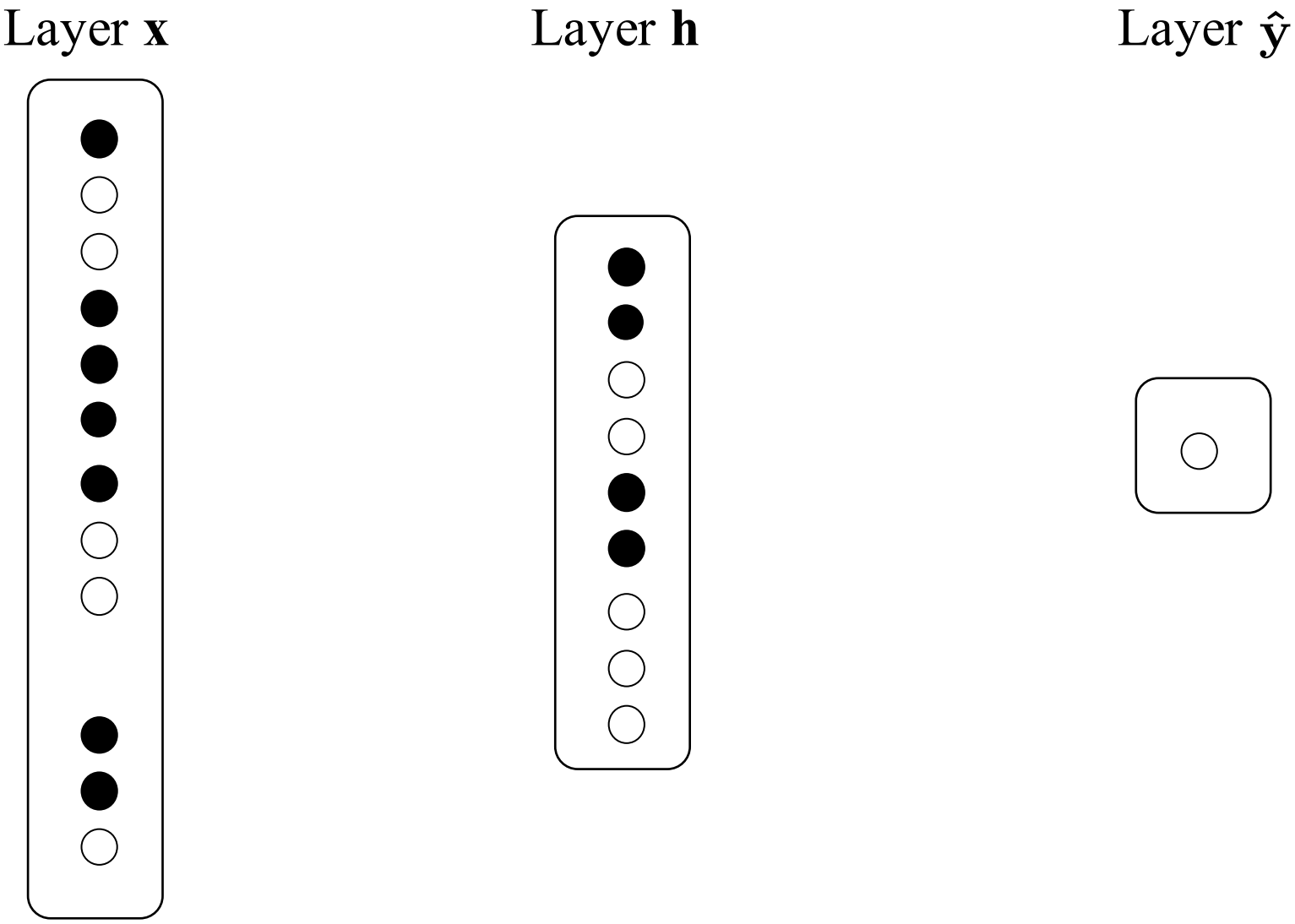
1. Understand Chain Rule as ♡ of Deep Learning

2. Demystify:
Deep Learning is MLE

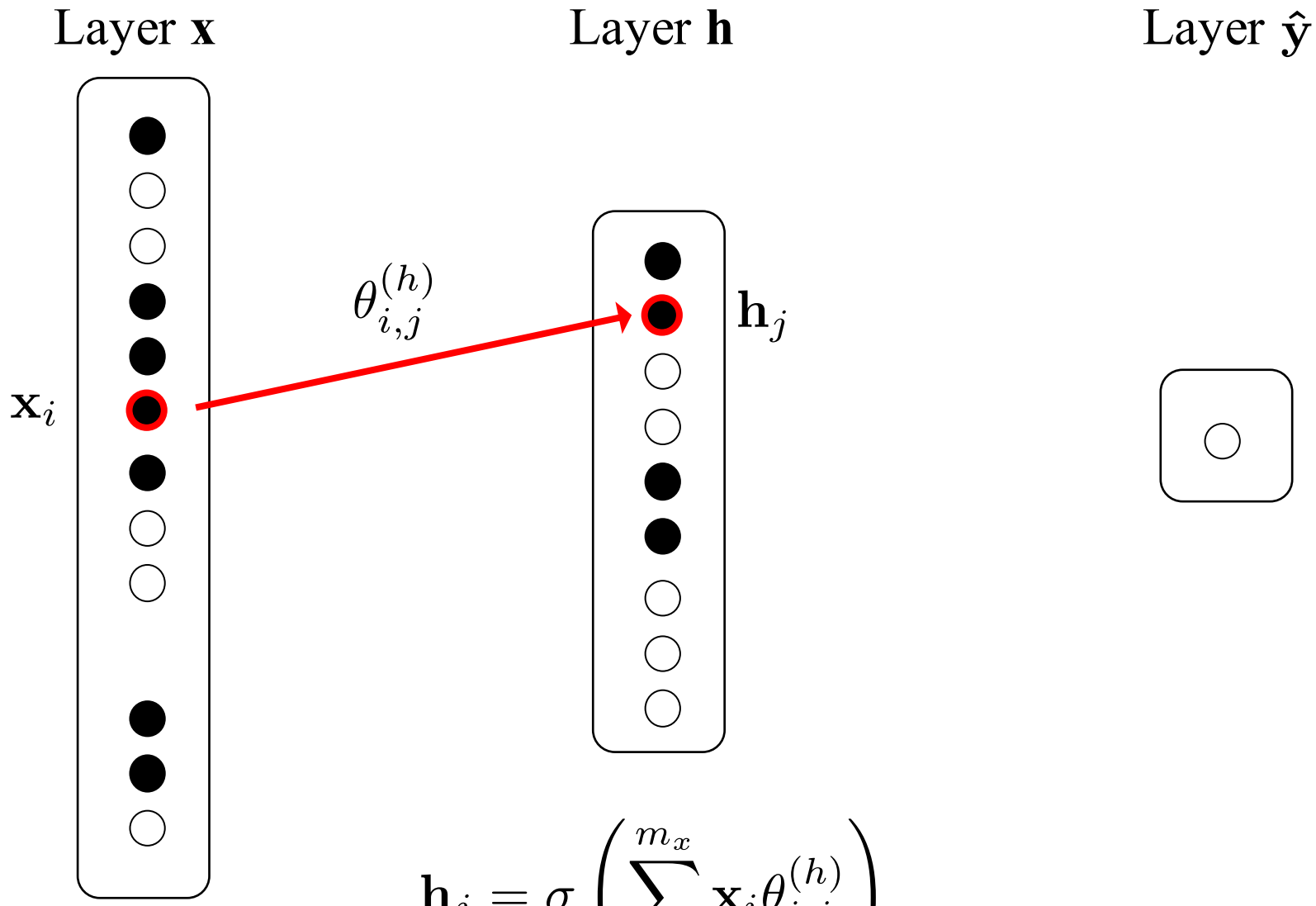
3. Become experts of logistic regression

Math worth knowing:

New Notation

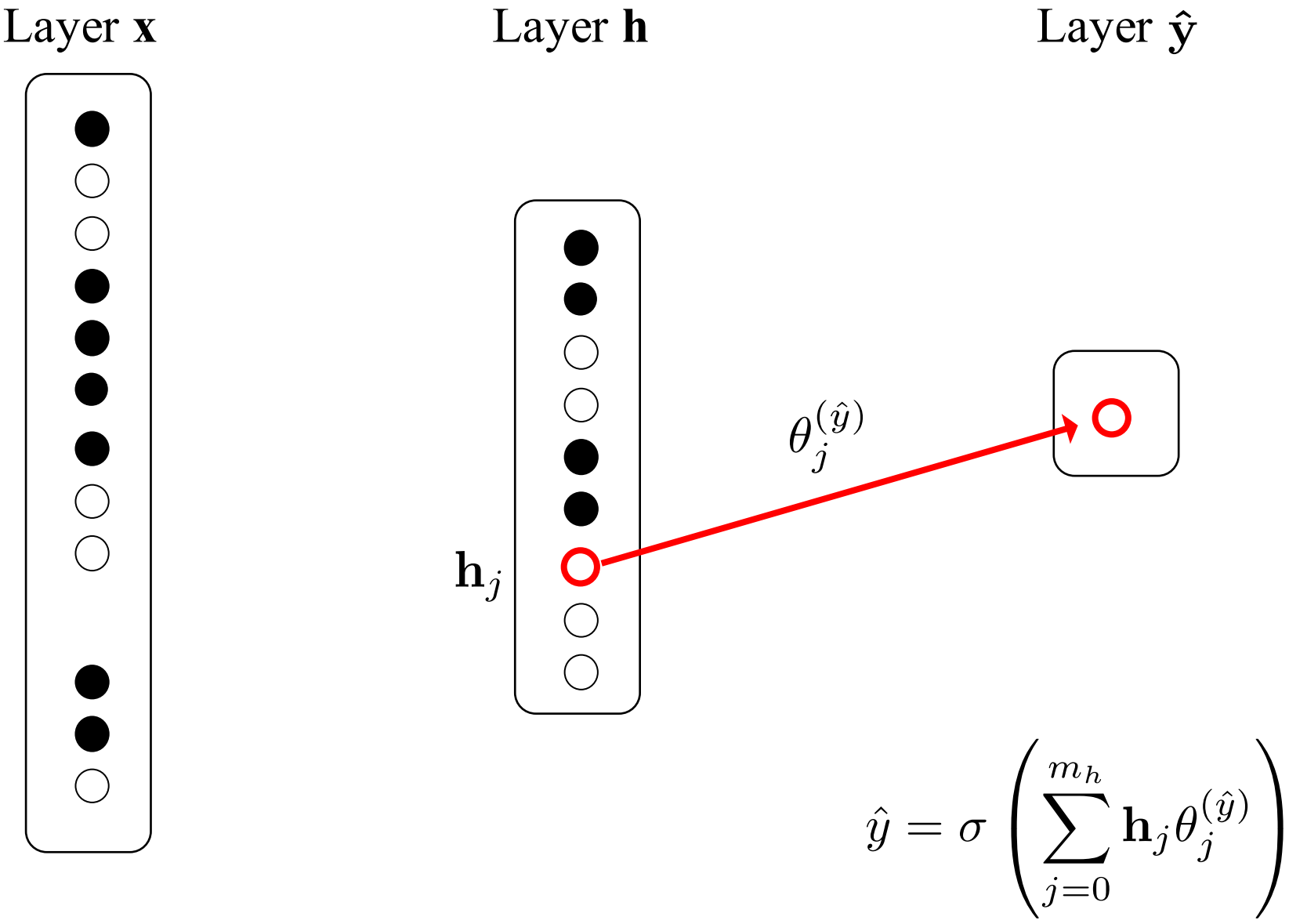


New Notation



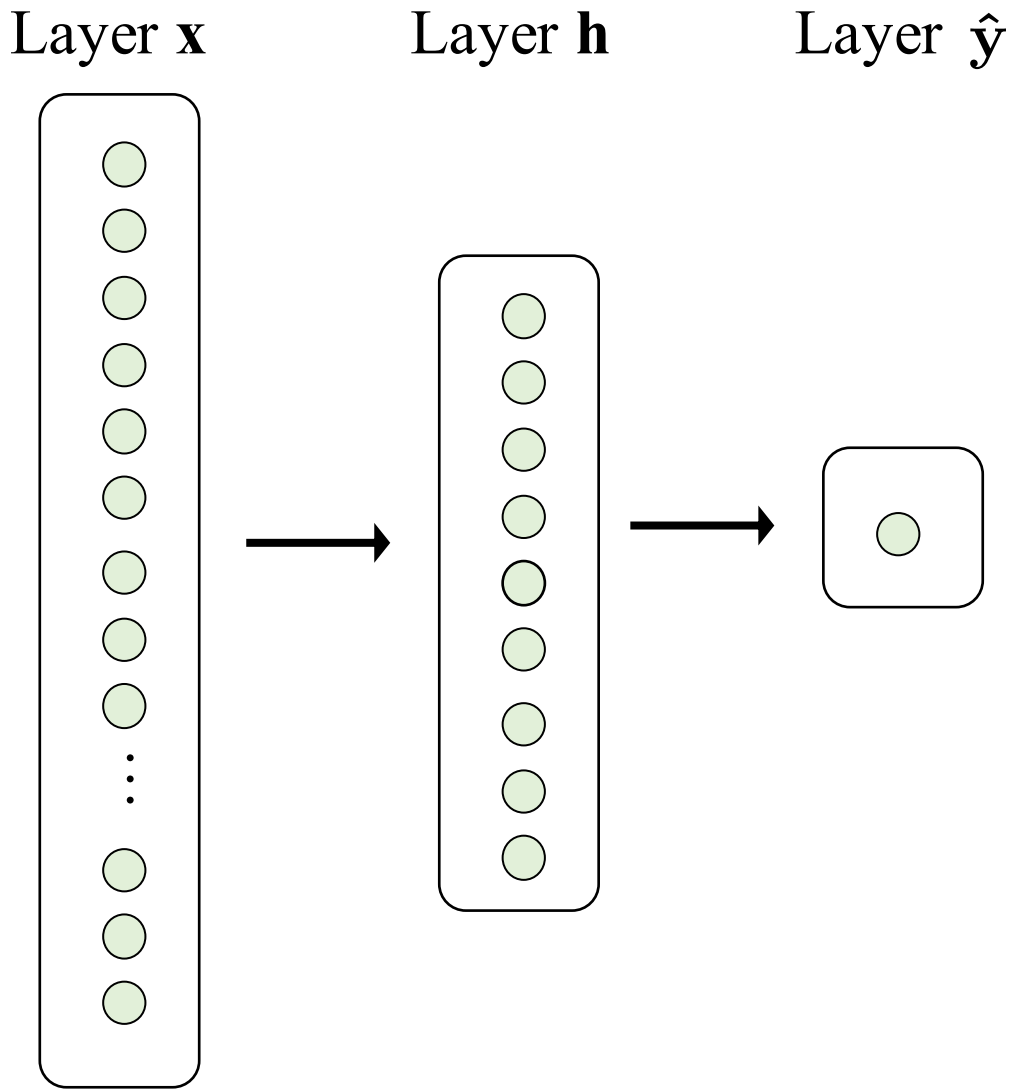
$$\mathbf{h}_j = \sigma\left(\sum_{i=0}^{m_x} \mathbf{x}_i \theta_{i,j}^{(h)}\right)$$

New Notation

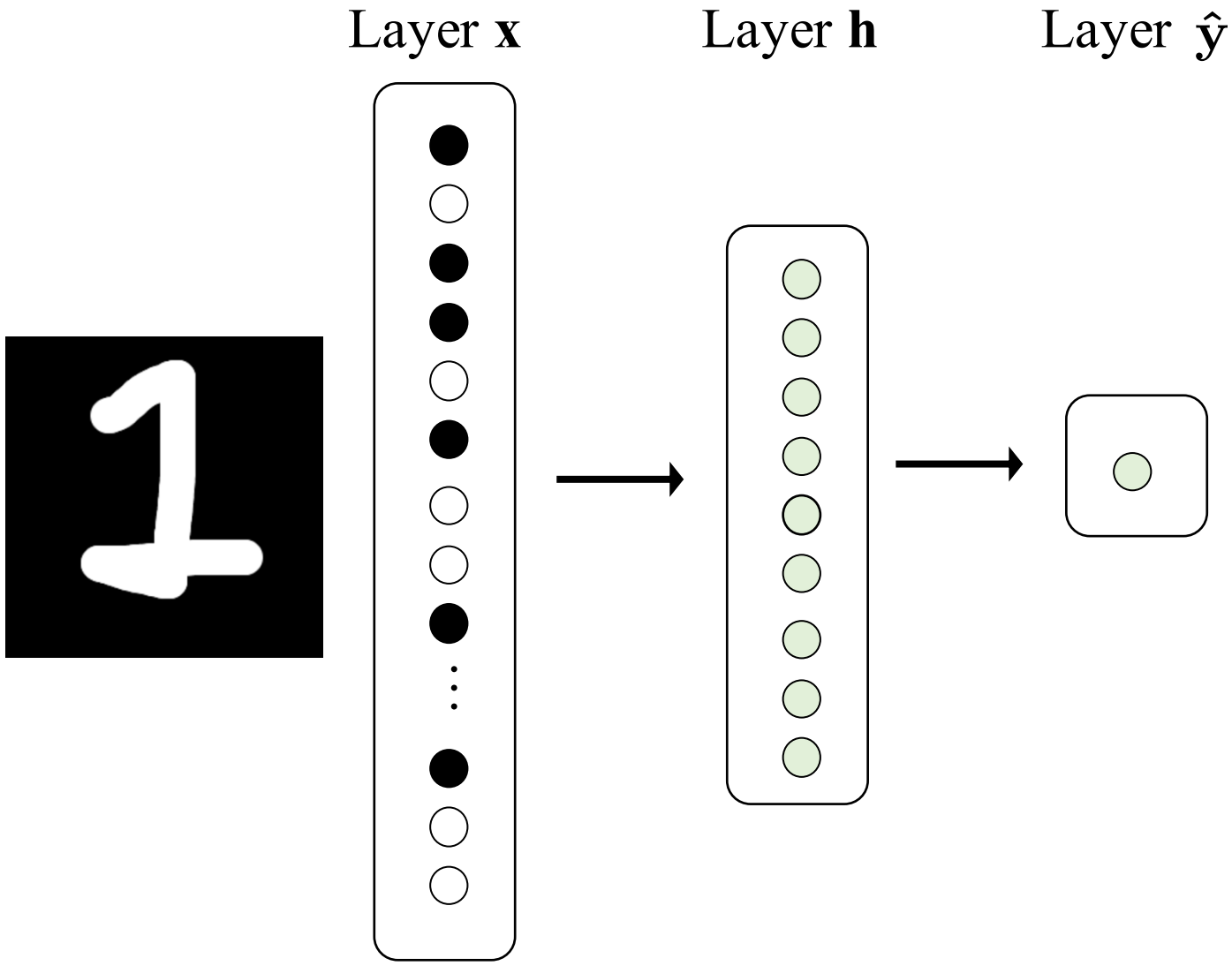


$$\hat{y} = \sigma\left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}\right)$$

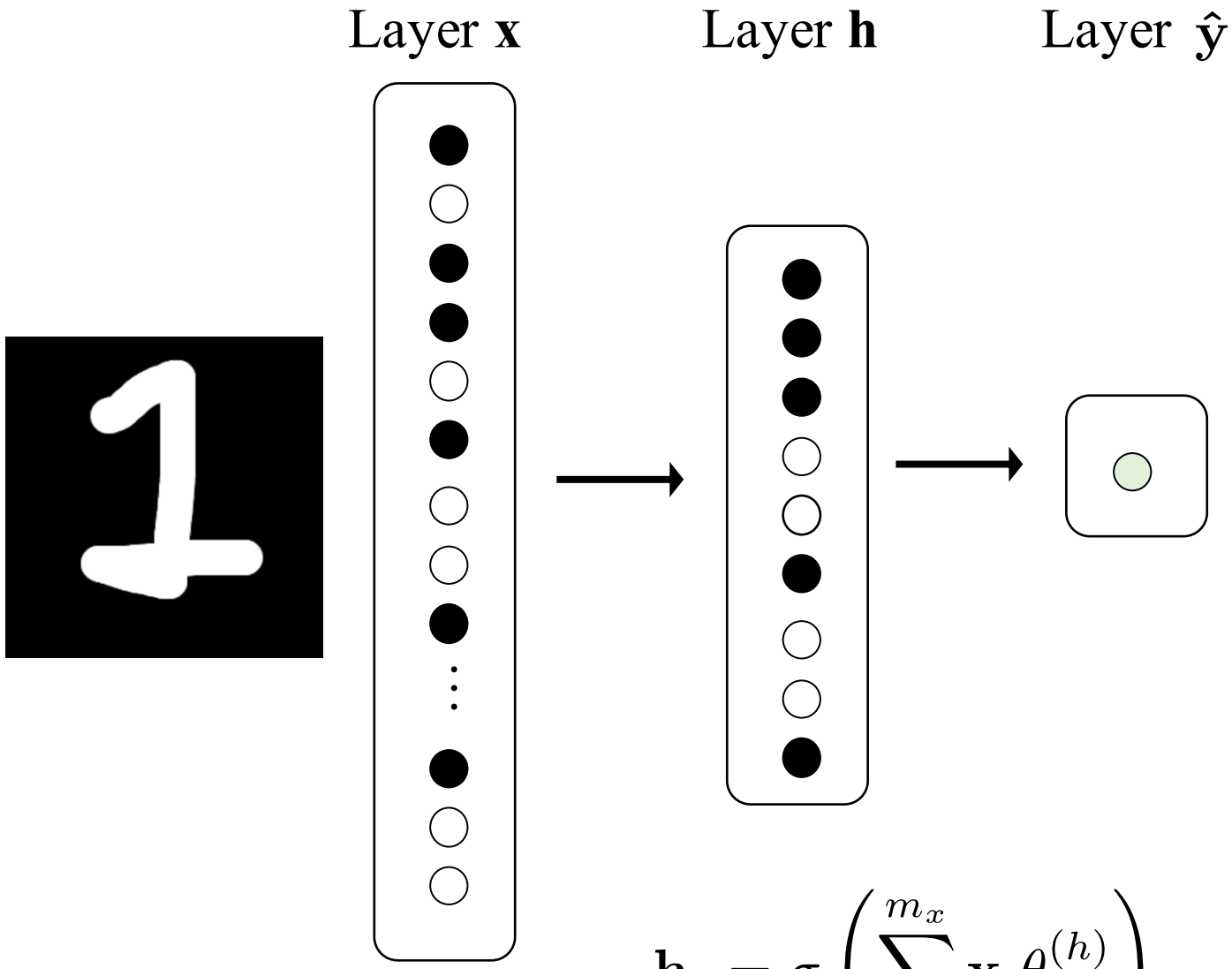
Forward Pass



Forward Pass

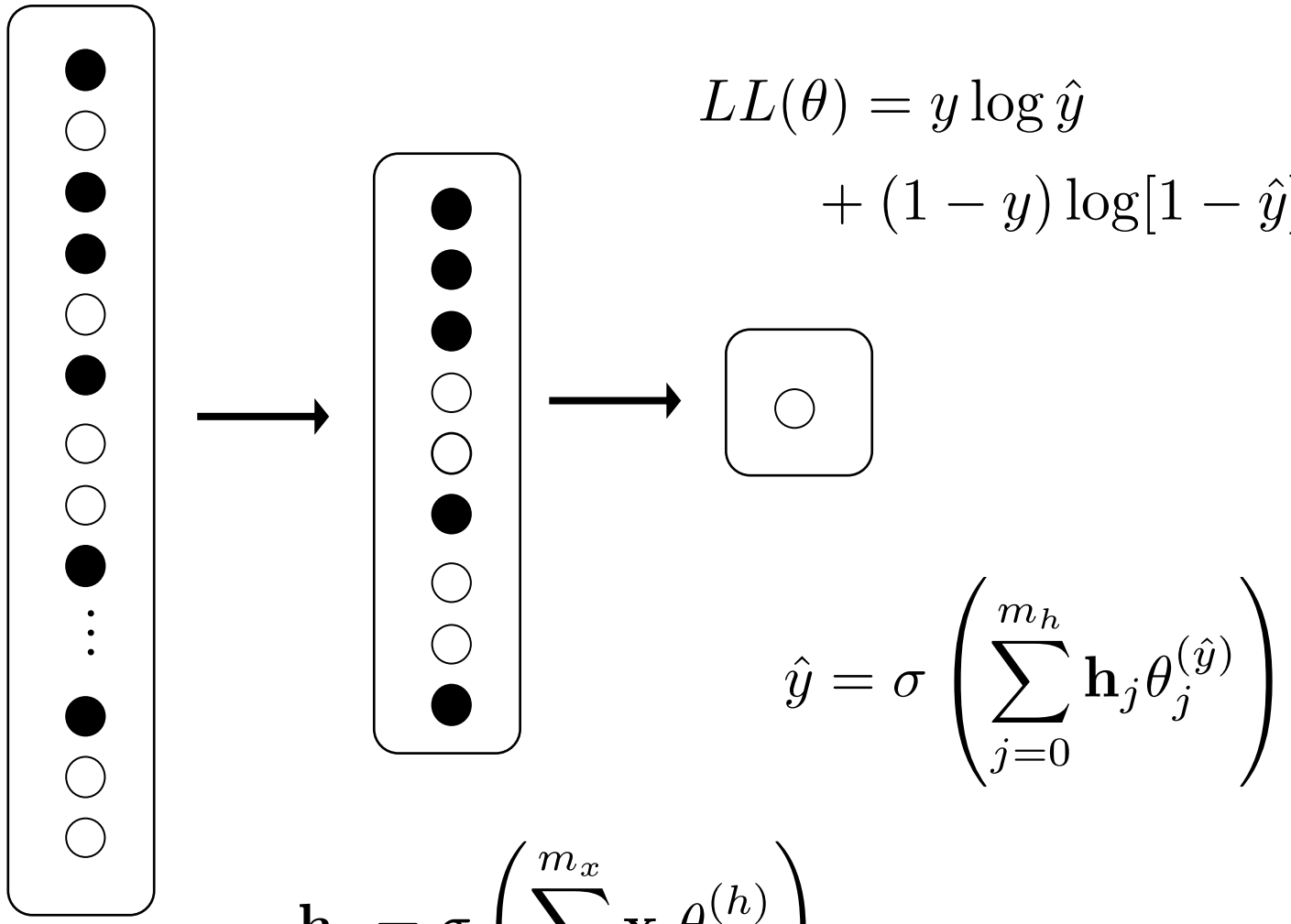


Forward Pass



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

Forward Pass

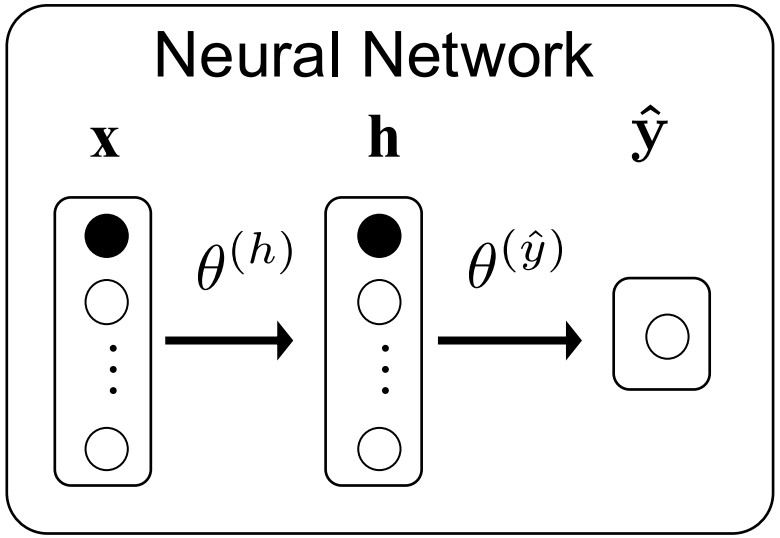


$$LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}]$$

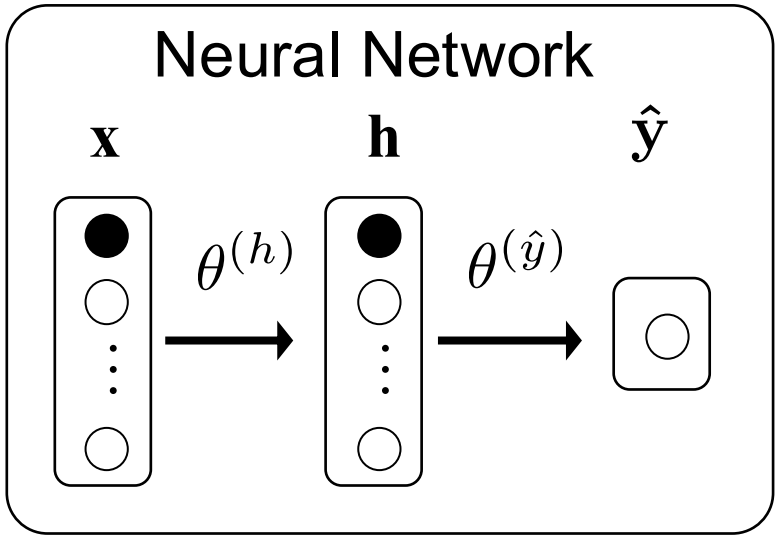
$$\hat{y} = \sigma\left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}\right)$$

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

All Together



Smoke Check 1



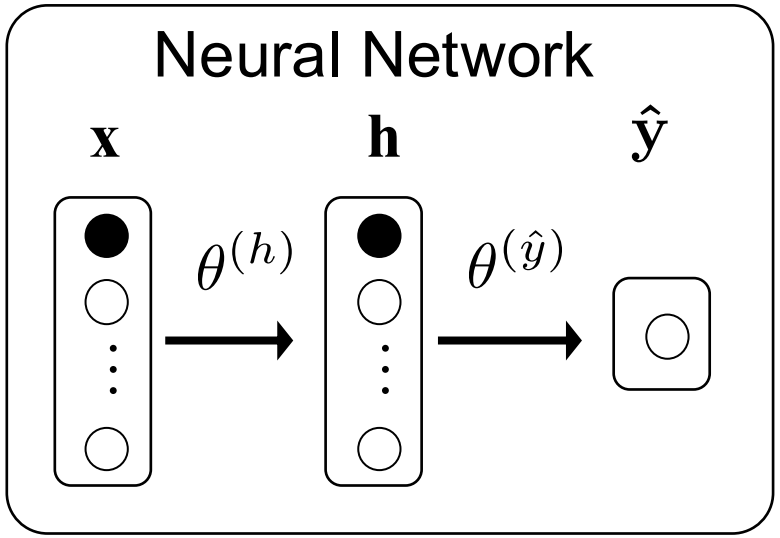
$$|\mathbf{x}| = 40$$

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

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

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

Smoke Check 2



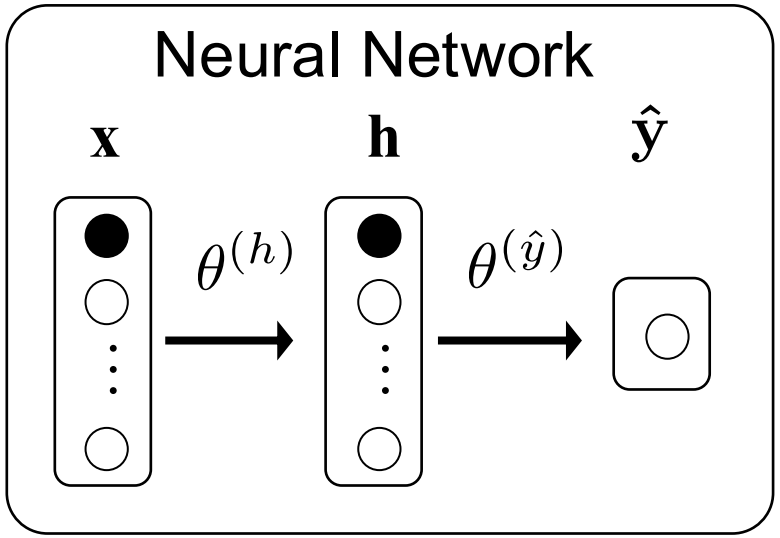
$$|\mathbf{x}| = 40$$

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

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

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

Smoke Check 3



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

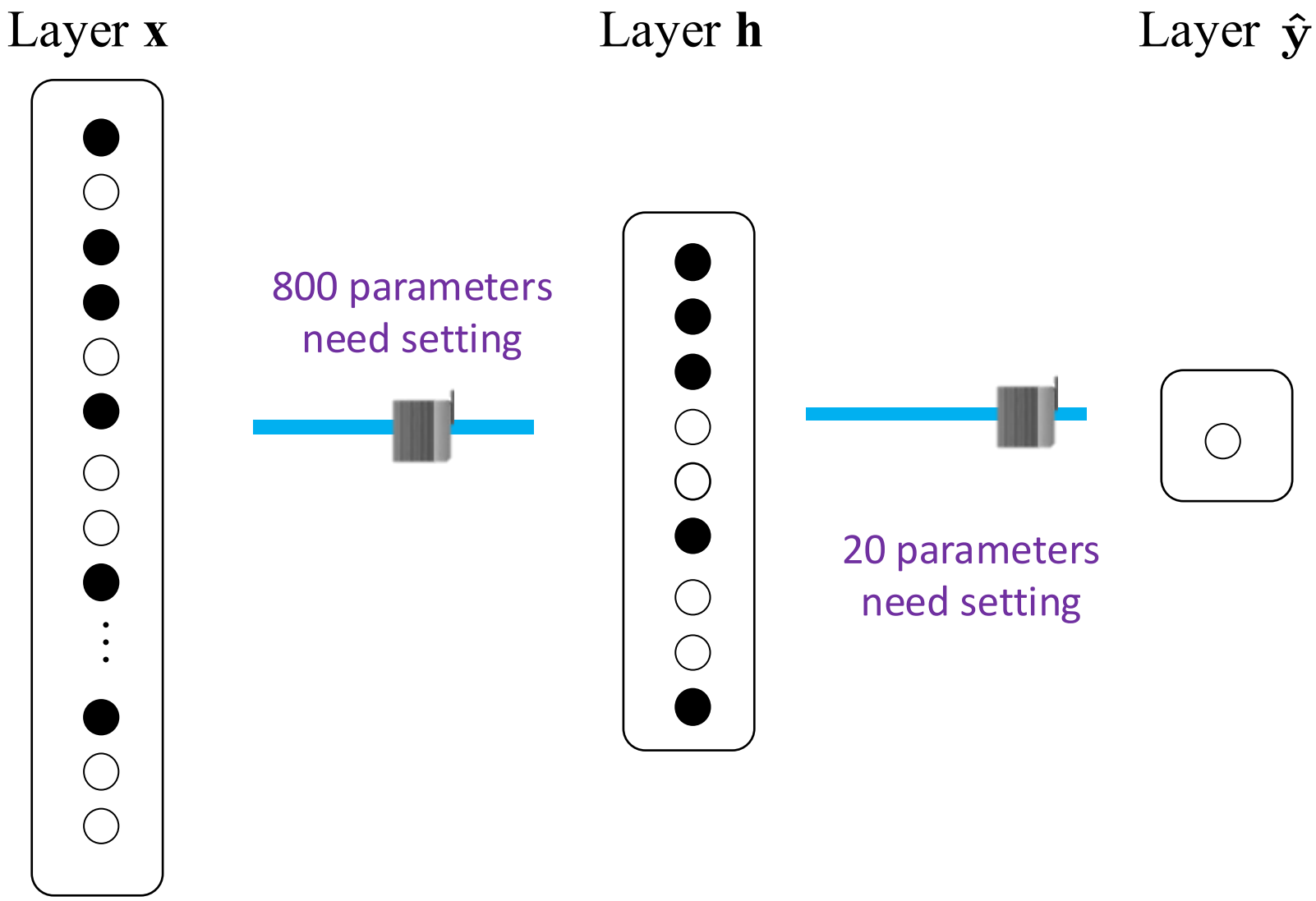
$$|\mathbf{h}| = 20$$

How many parameters in total?

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

Today: Do Something Brave

Forward Pass



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

Smoke 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**.

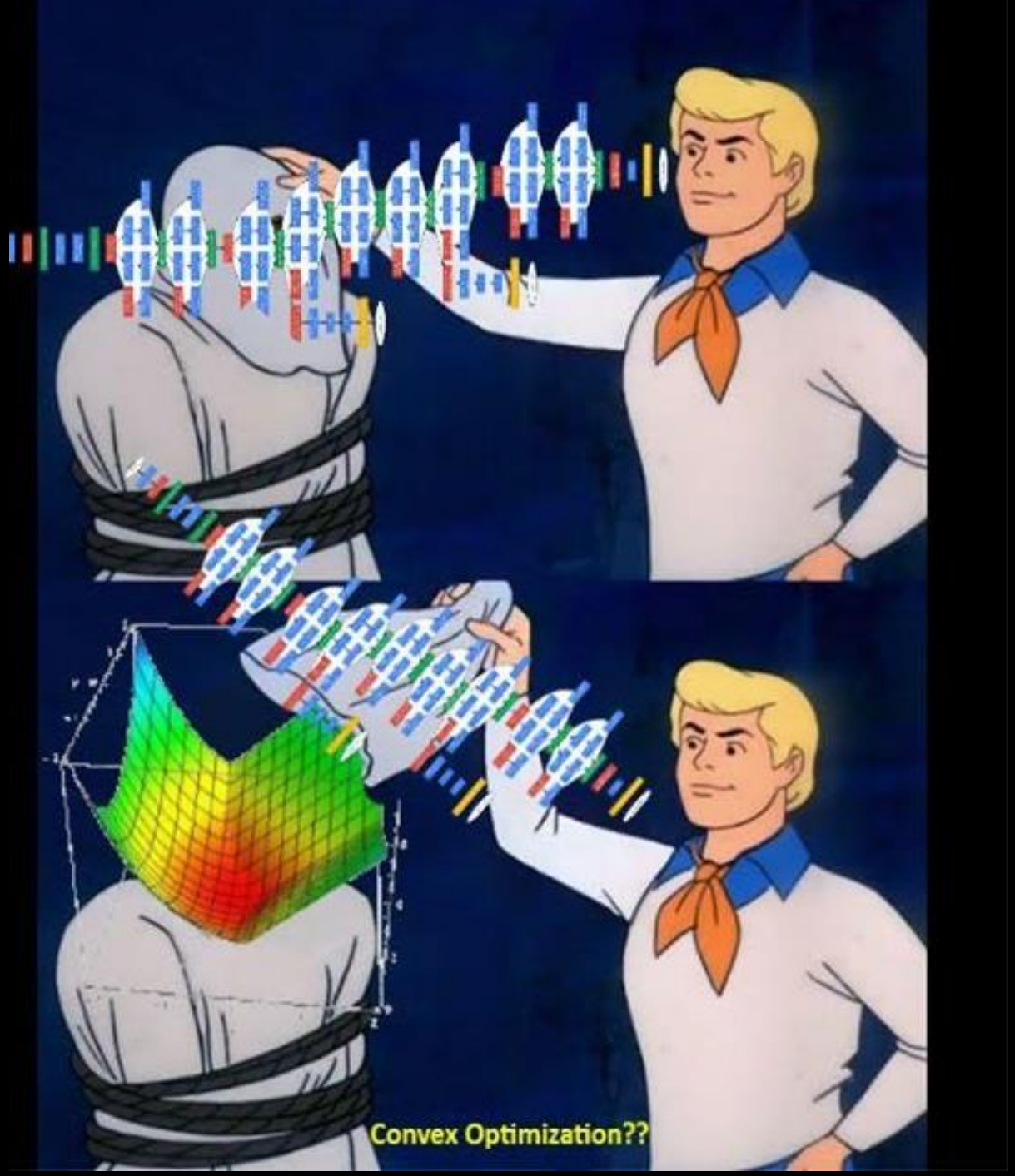
We can find **useful thetas**, by searching for ones that **maximize likelihood** of our training data

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

$$P(Y = 1|X = \mathbf{x}) = \hat{y}$$
$$P(Y = 0|X = \mathbf{x}) = 1 - \hat{y}$$

2. Calculate the log probability for all data

Same Assumption, Same LL

$$P(Y = 1|X = \mathbf{x}) = \hat{y} \qquad \hat{y} = \sigma\left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}\right) \qquad \mathbf{h}_j = \sigma\left(\sum_{i=0}^{m_x} \mathbf{x}_i \theta_{i,j}^{(h)}\right)$$

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

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

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

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

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

$$P(Y = 1|X = \mathbf{x}) = \hat{y}$$
$$P(Y = 0|X = \mathbf{x}) = 1 - \hat{y}$$

2. Calculate the log probability for all data

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

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

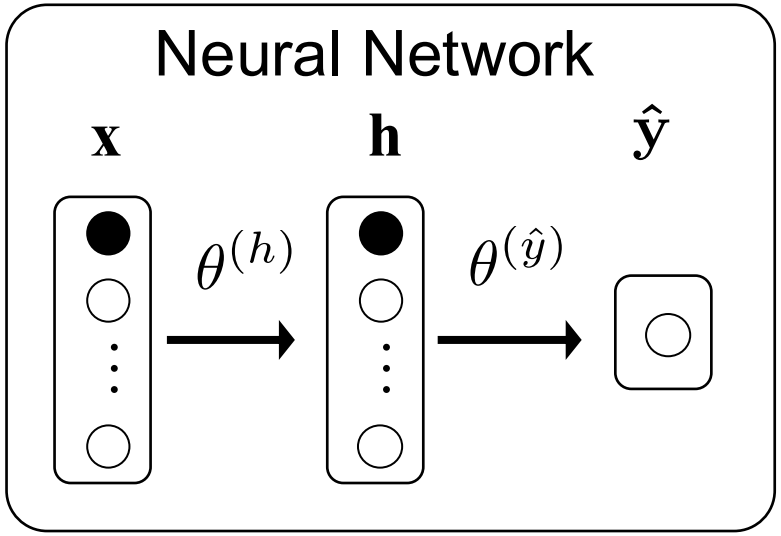
Derivative Goals

Loss with respect to output layer params

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

Loss with respect to hidden layer params

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



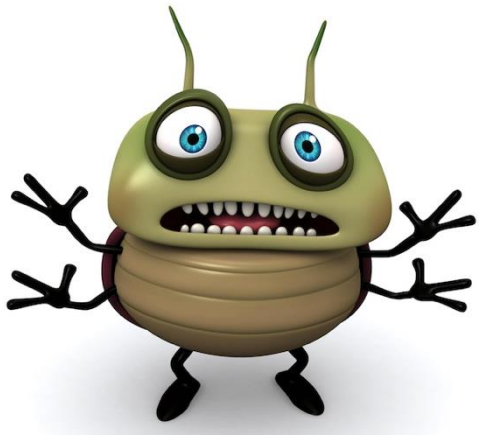
Bad Approach

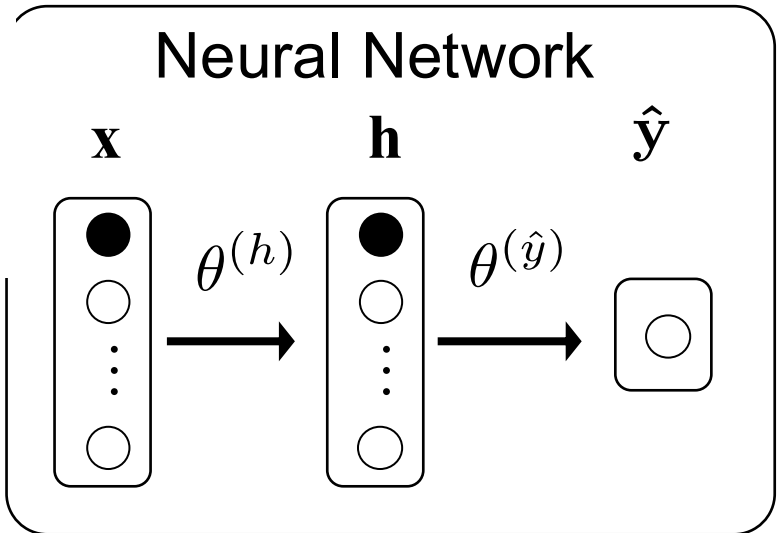
$$LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}]$$

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

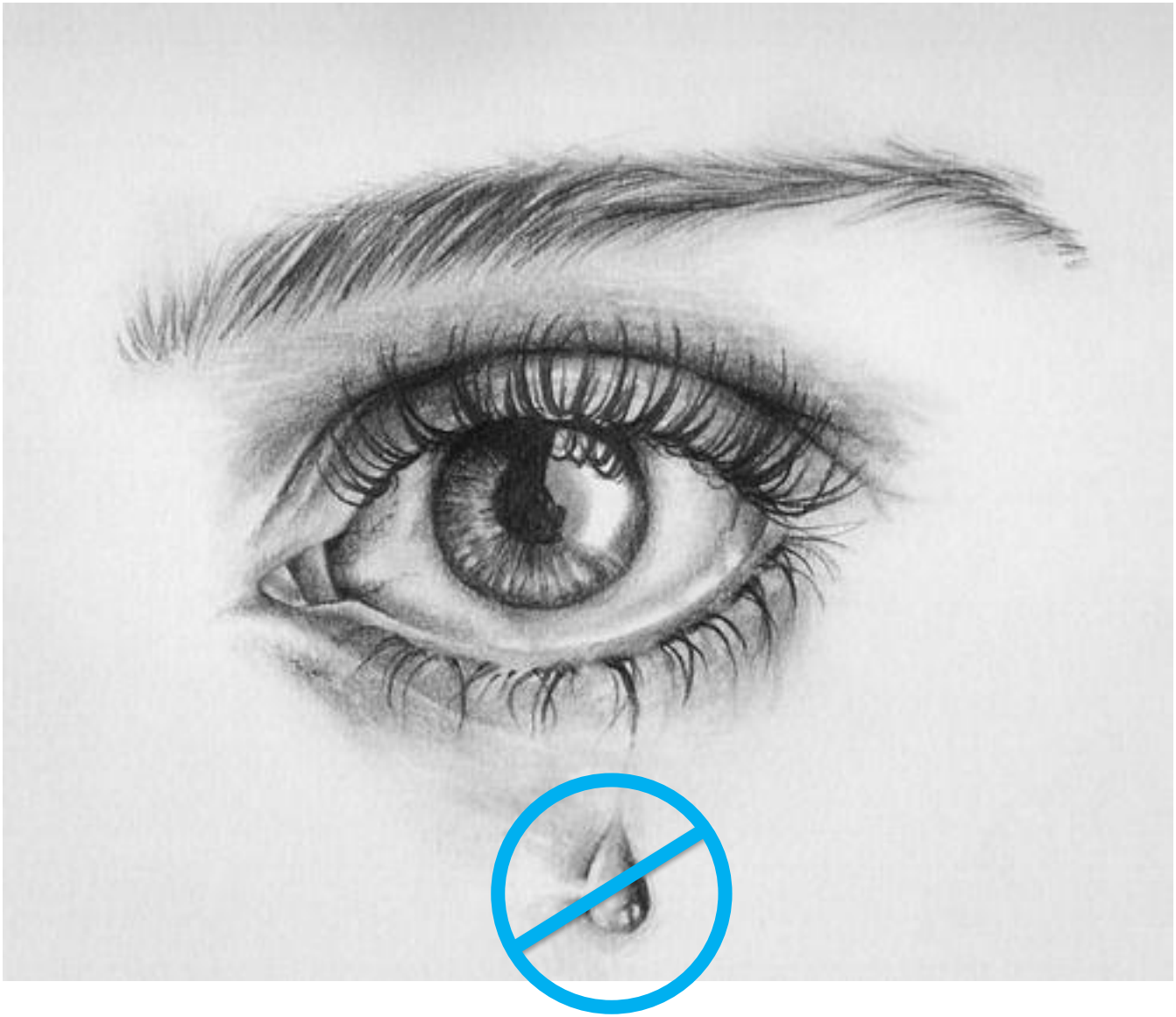
Pain, Tears

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





Derivatives Without Tears



Big Idea #1: Chain Rule

Woah Mrs. Noonan, 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

Recall

Sigmoid has a Beautiful Slope

$$\frac{\partial}{\partial \theta_j}\sigma(\theta^T x)? \qquad \frac{\partial}{\partial z}\sigma(z) = \sigma(z)[1-\sigma(z)]$$

---

where $z = \theta^T x$

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

Chain rule!

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

Plug and chug

Sigmoid, you should be a ski hill

This is ~~Sparta~~!!!!!

↑

Stanford

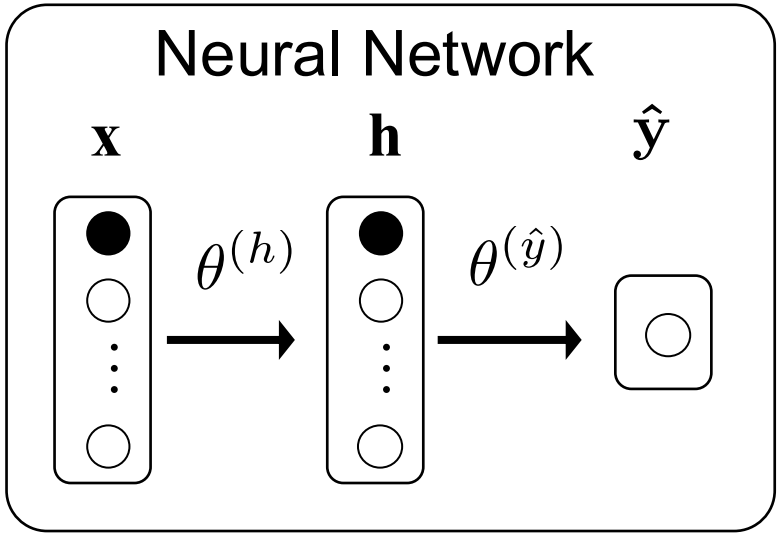
Derivative Goals

Loss with respect to output layer params

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

Loss with respect to hidden layer params

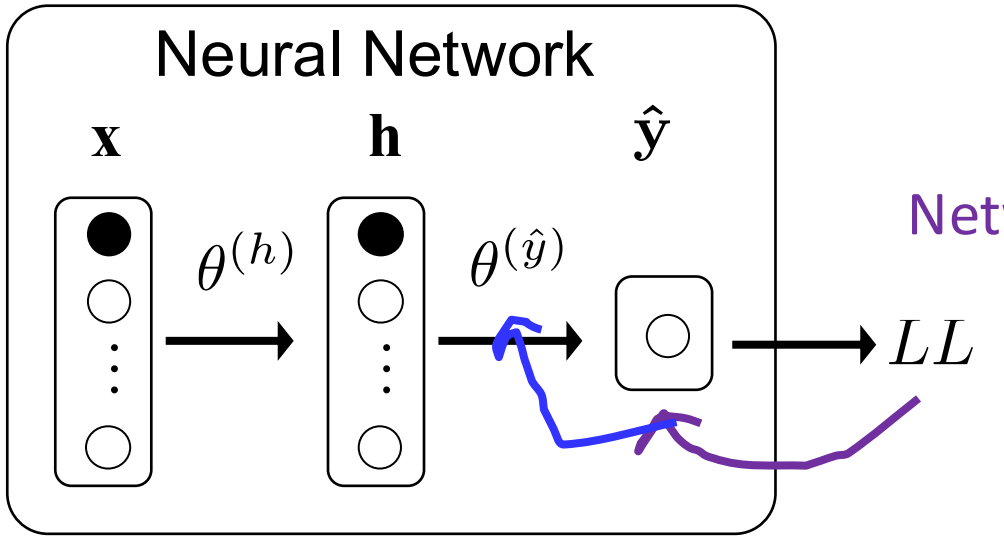
$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}}$$



Chain Rule Example 1

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

Goal



Network

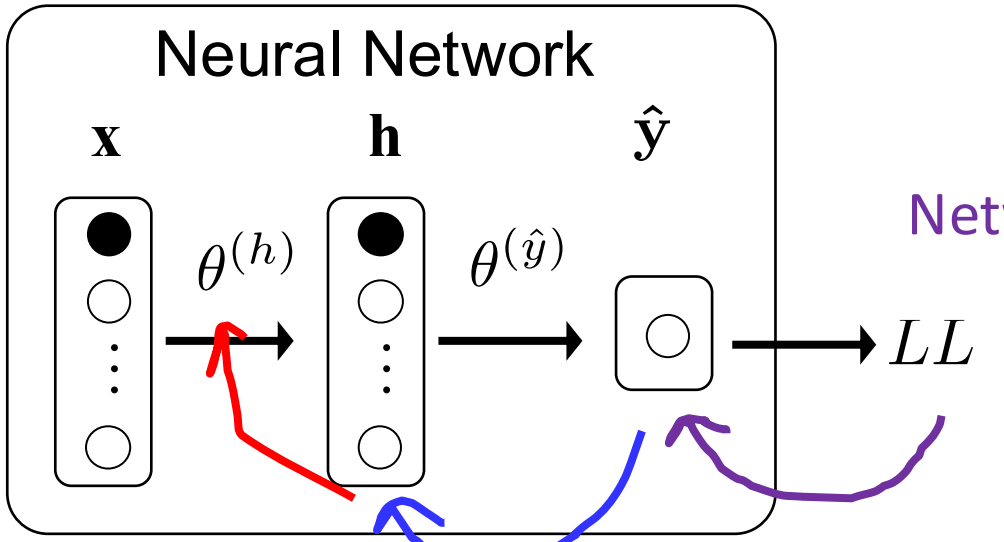
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}}$$

Decomposition

Chain Rule Example 2

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

Goal



Network

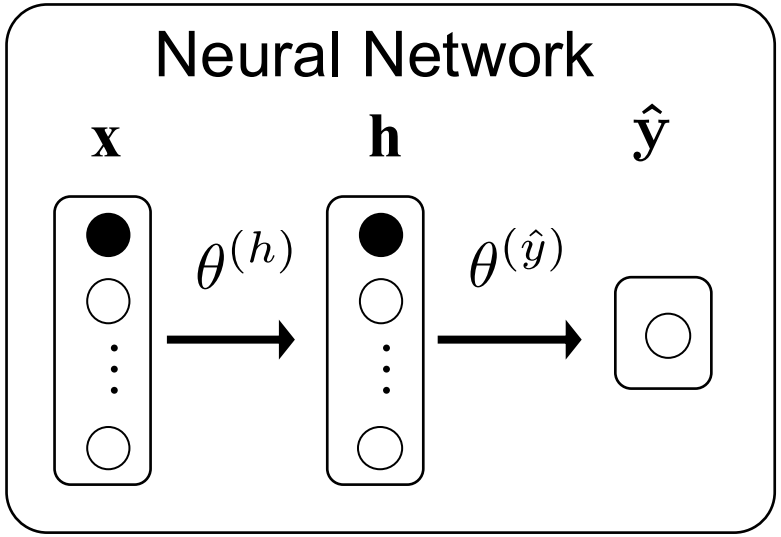
$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{h}_j} \cdot \frac{\partial \mathbf{h}_j}{\partial \theta_{i,j}^{(h)}}$$

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

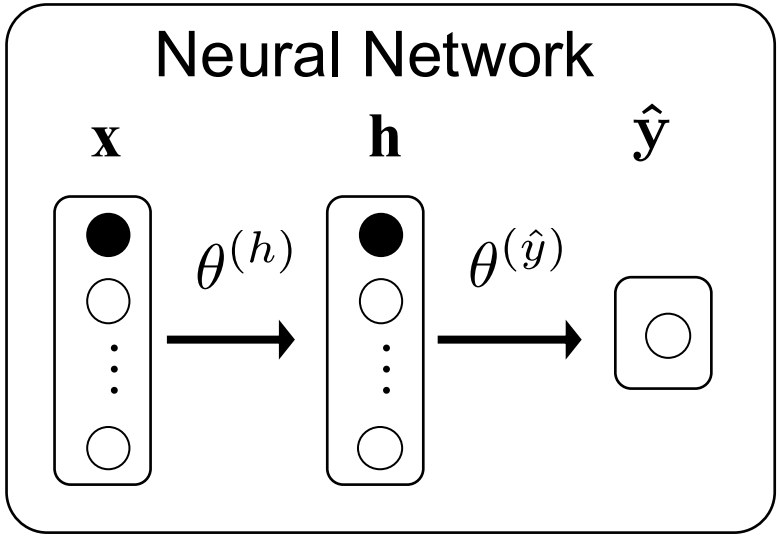
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \text{  } \cdot \text{  }$$

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

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

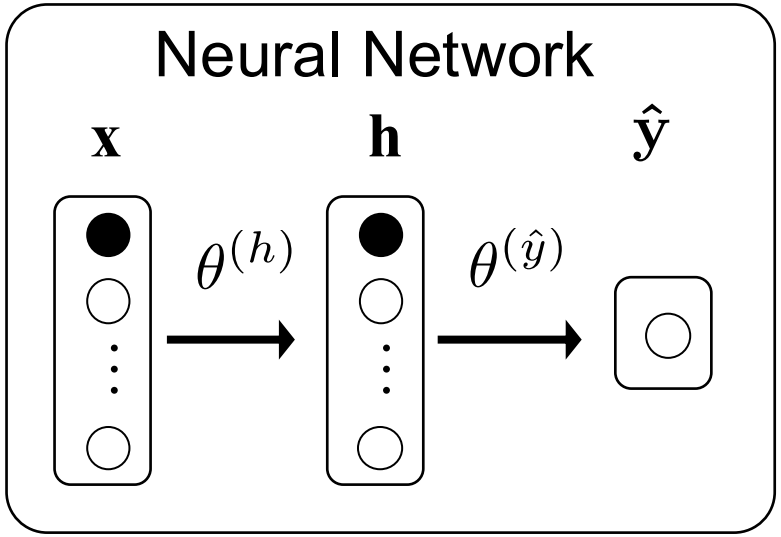
Boom!

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



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

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

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

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

That one too?

Make it Simple

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

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

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

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

Congrats. You now know Backpropagation

Moment of silence

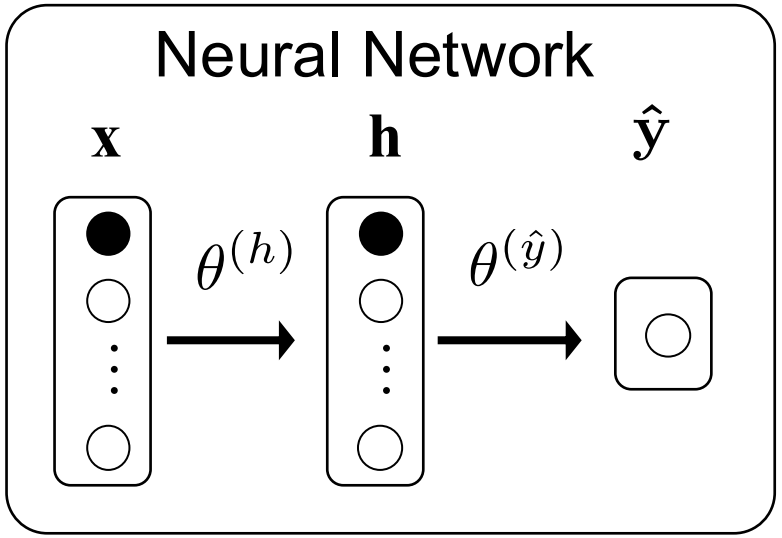
Summary: Simple Calculations For

Loss with respect to output layer params

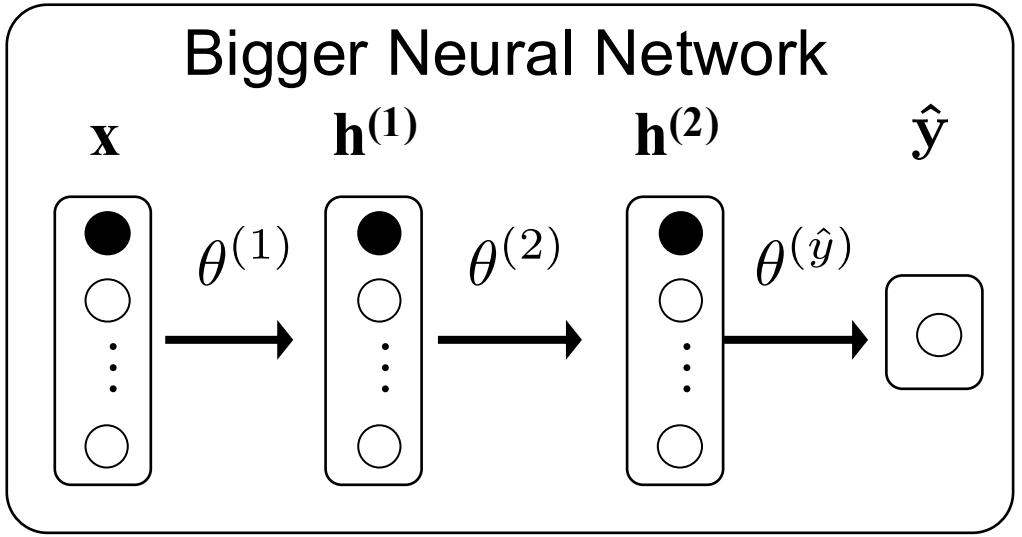
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}}$$

Loss with respect to hidden layer params

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



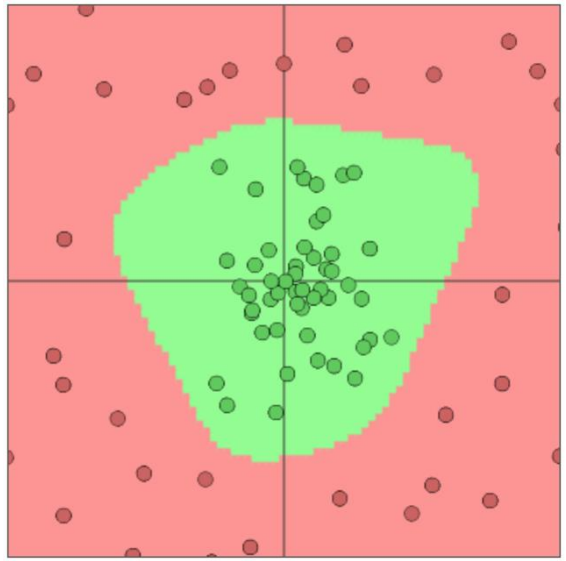
What Would You Do Here?

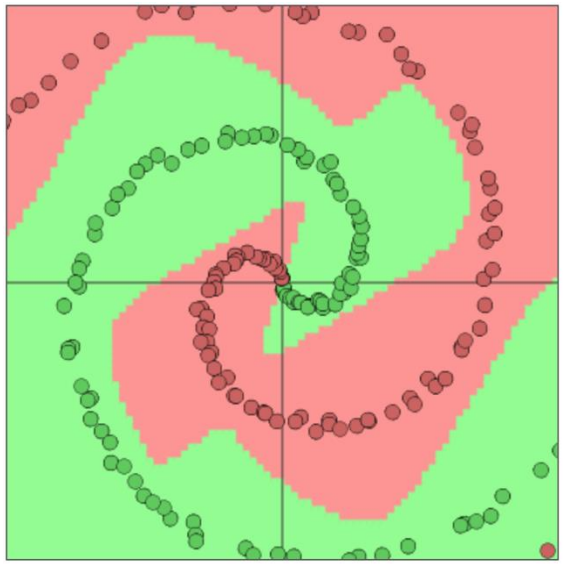


Chain rule:
Game changer for artificial intelligence

Neural Networks Can Learn Complex Functions

- Some data sets/functions are not separable

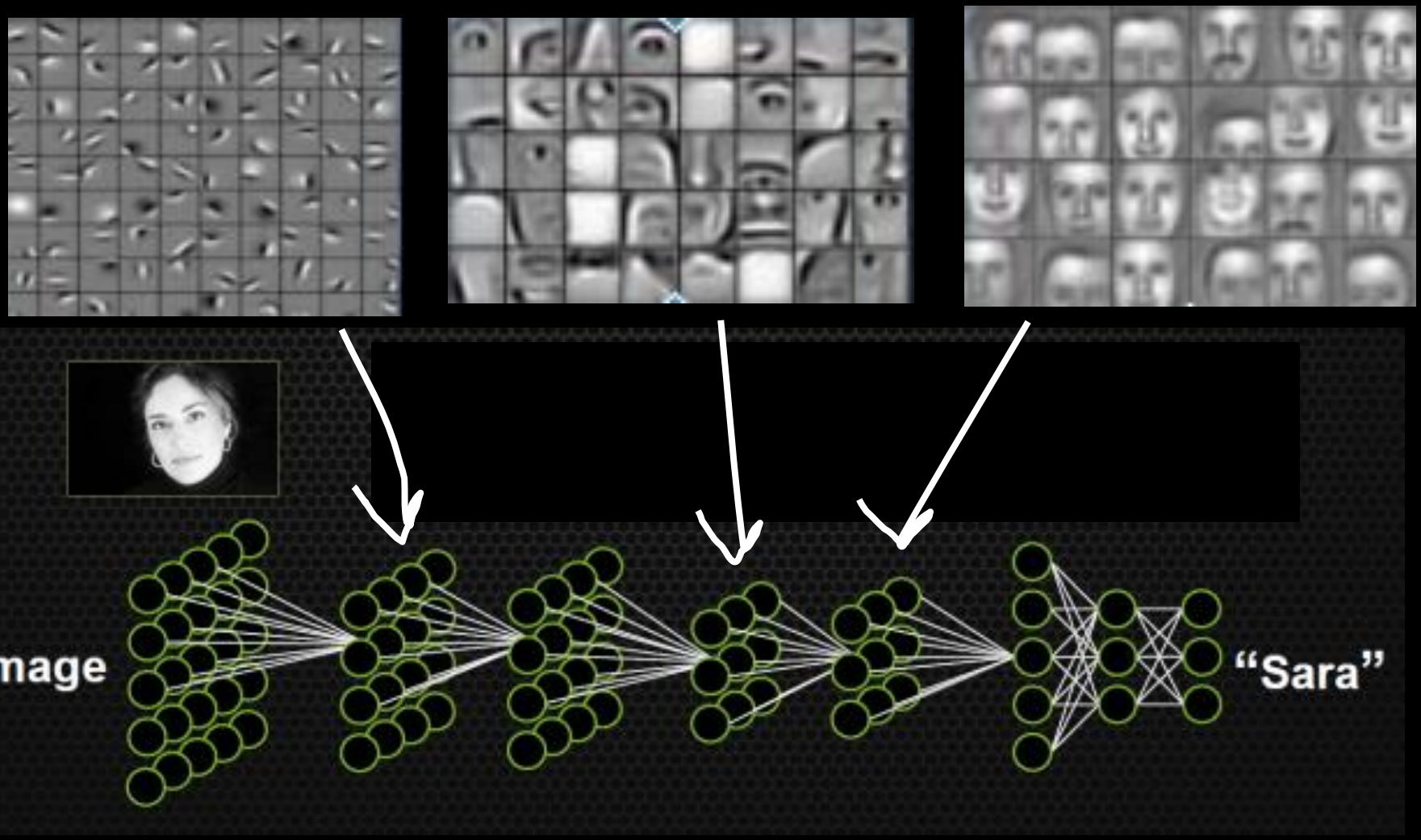




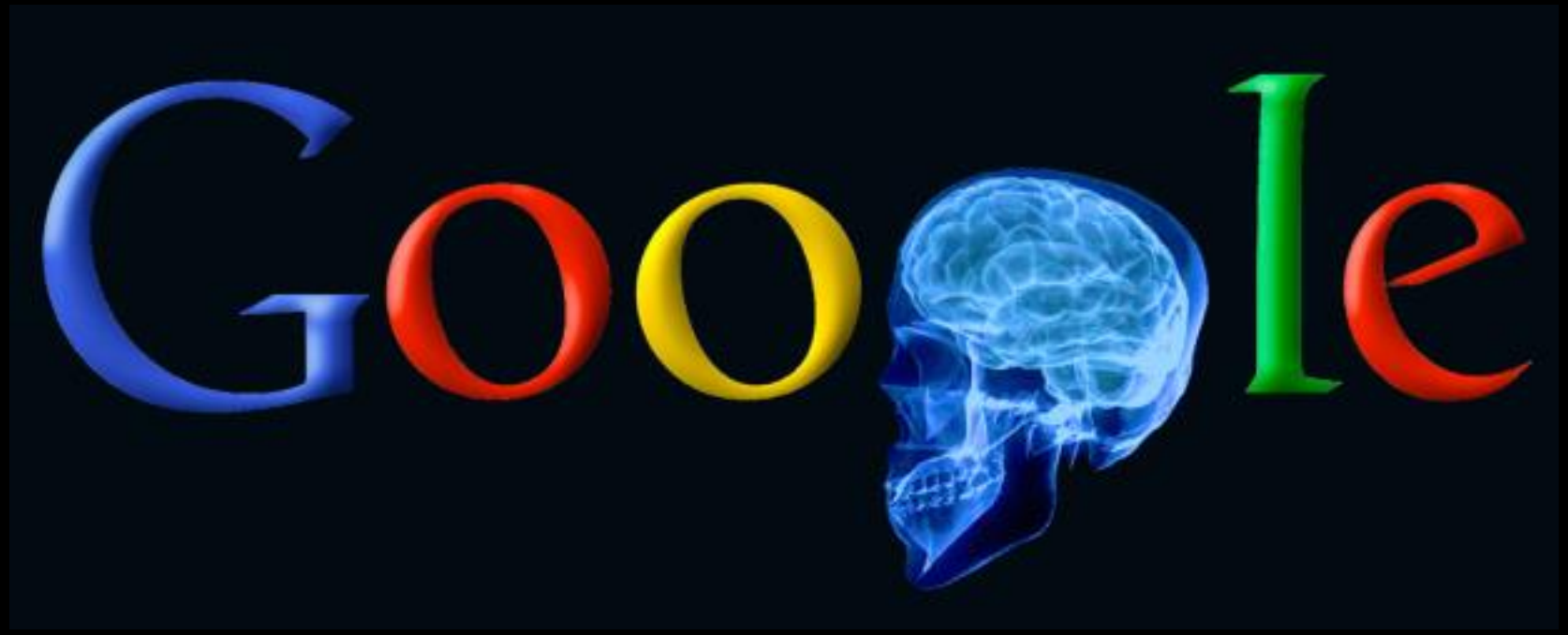
  - These are classifiers learned by neural networks

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

Works for any number of layers



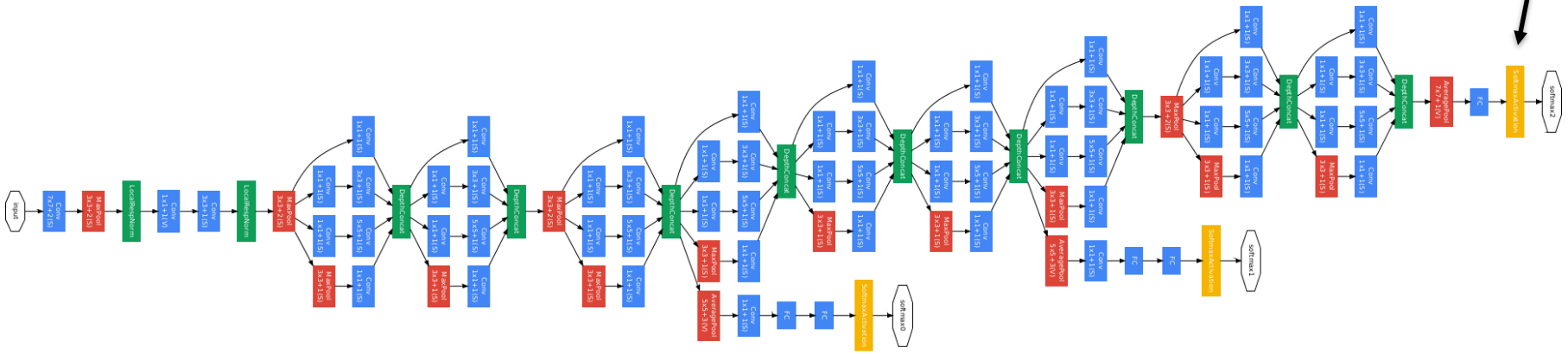
GoogLeNet Brain



1 Trillion Artificial Neurons

GoogLeNet Brain

Multiple,
Multi class output



22 layers deep

Piech

The Cat Neuron

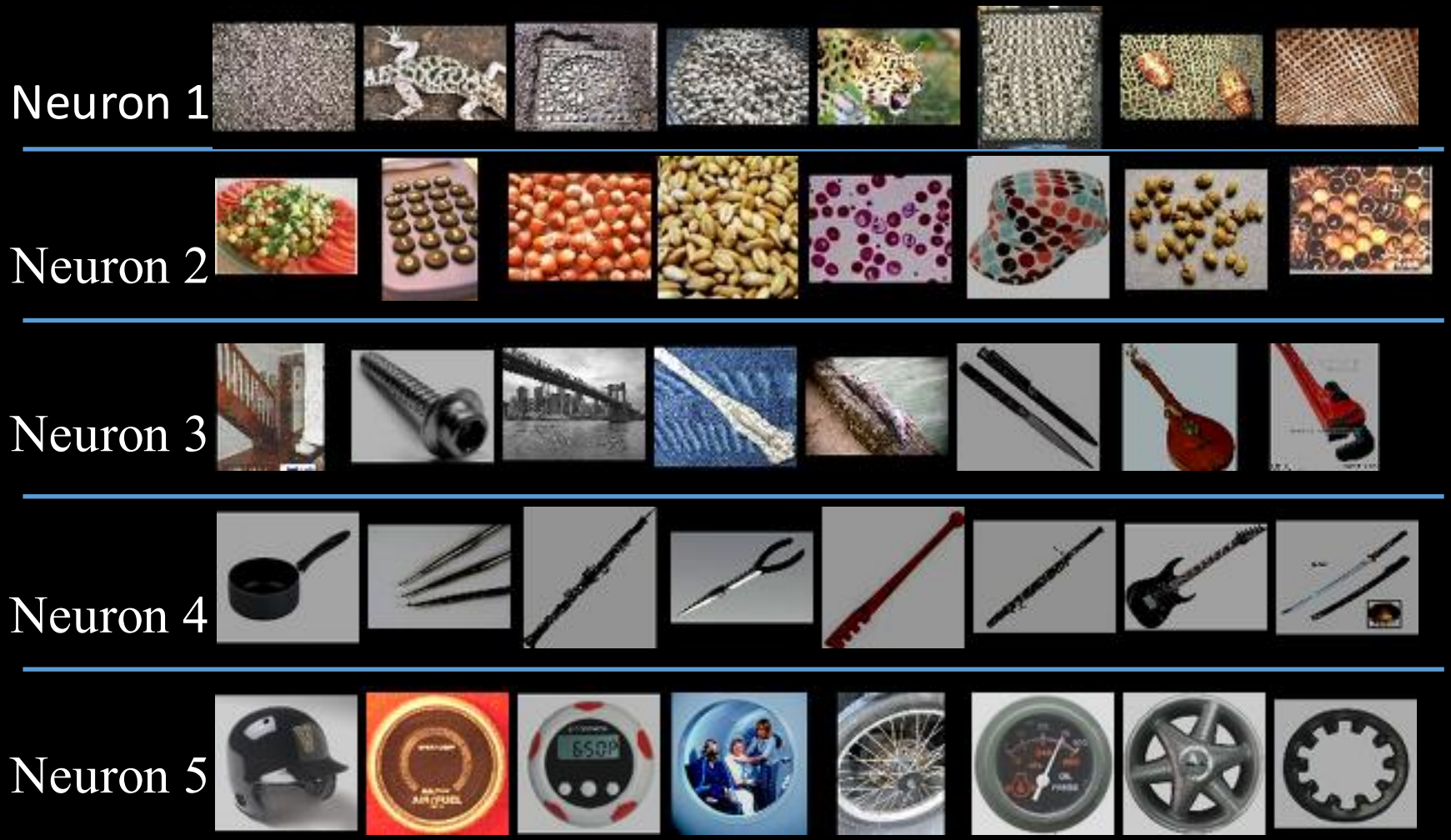
Top stimuli from the test set

Optimal stimulus
by numerical optimization

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

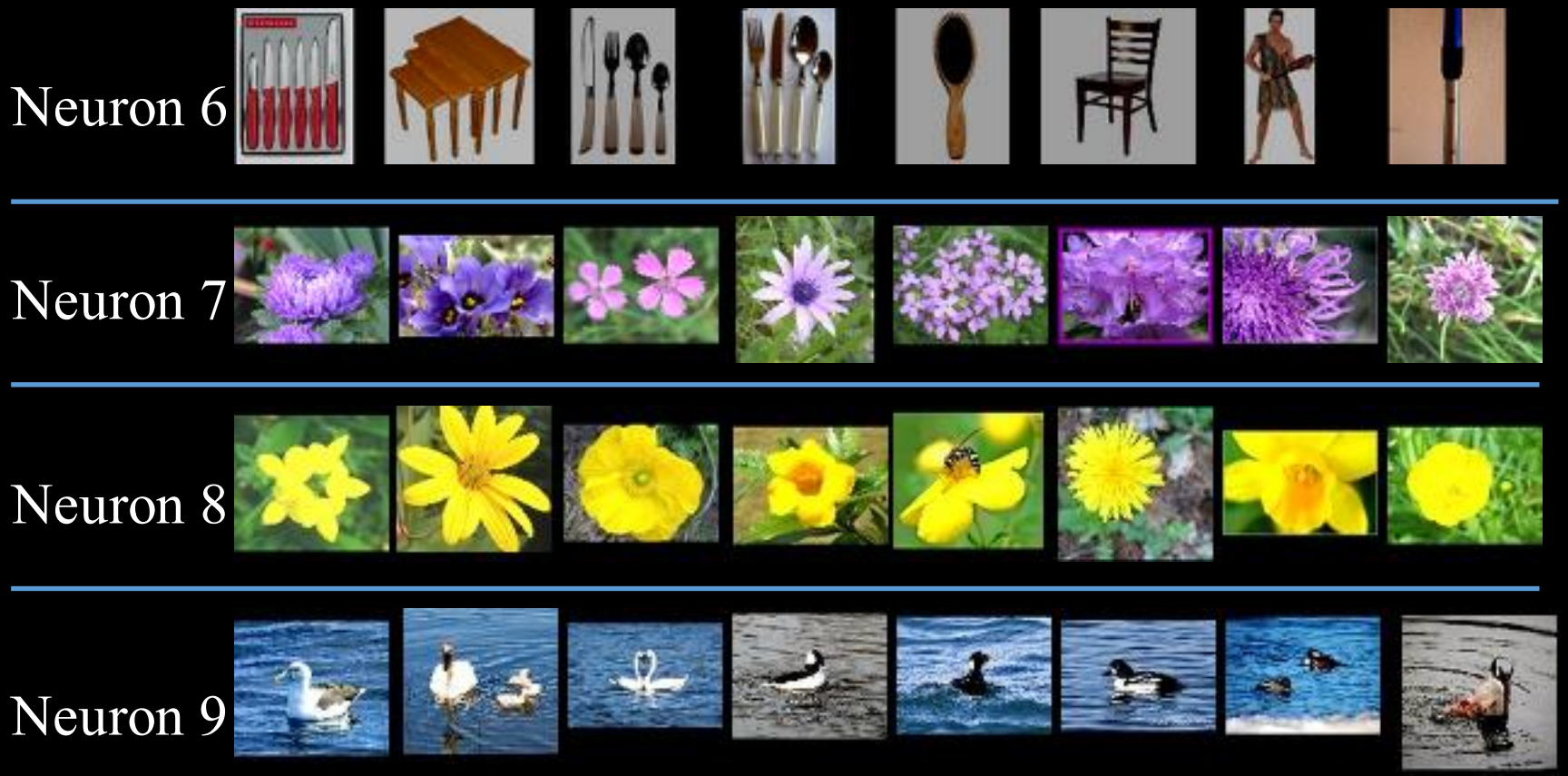

Hire the smartest people in the world
Invent cat detector

Best Neuron Stimuli



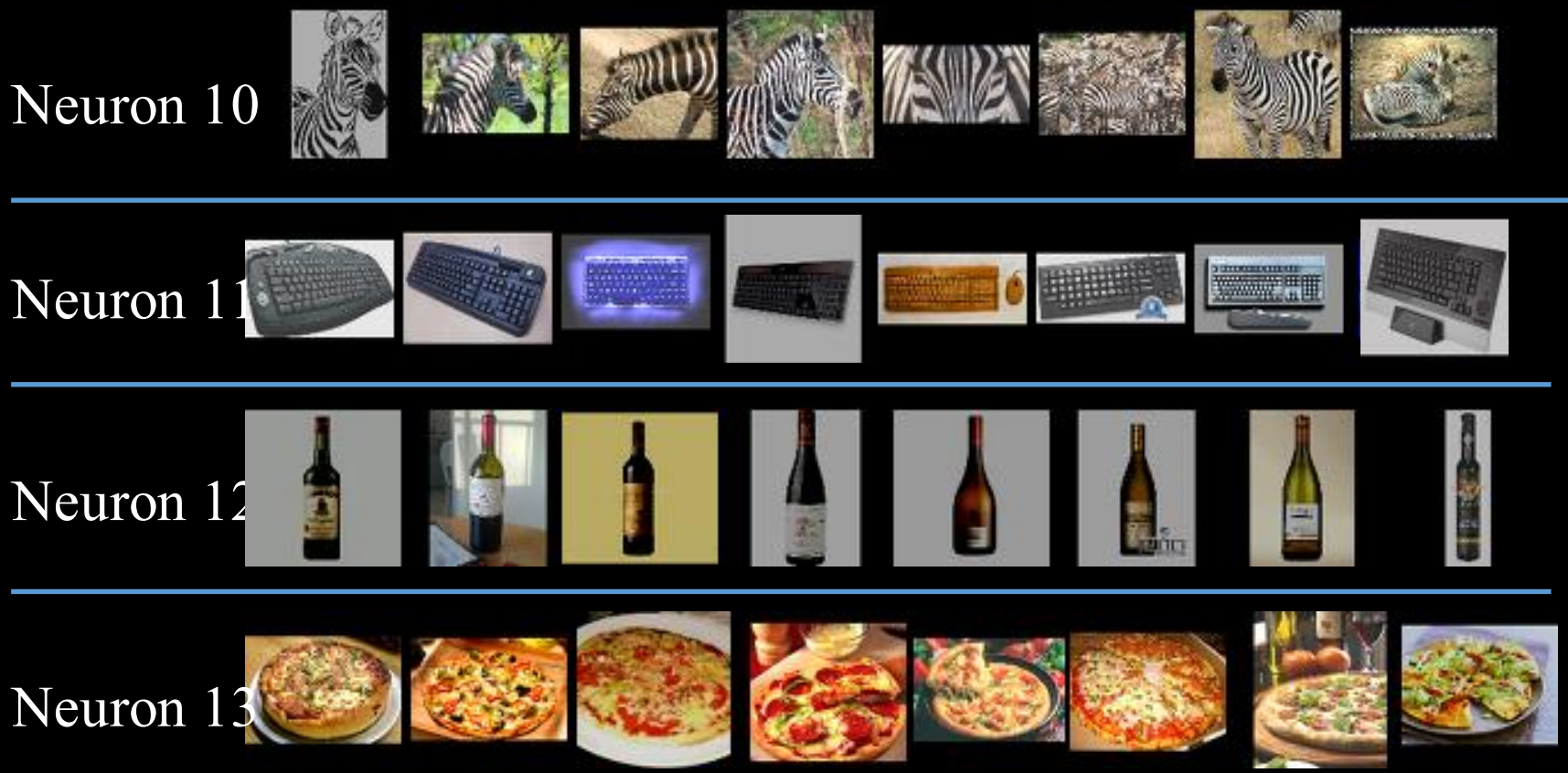
Le, et al., *Building high-level features using large-scale unsupervised*

Best Neuron Stimuli



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

Best Neuron Stimuli



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

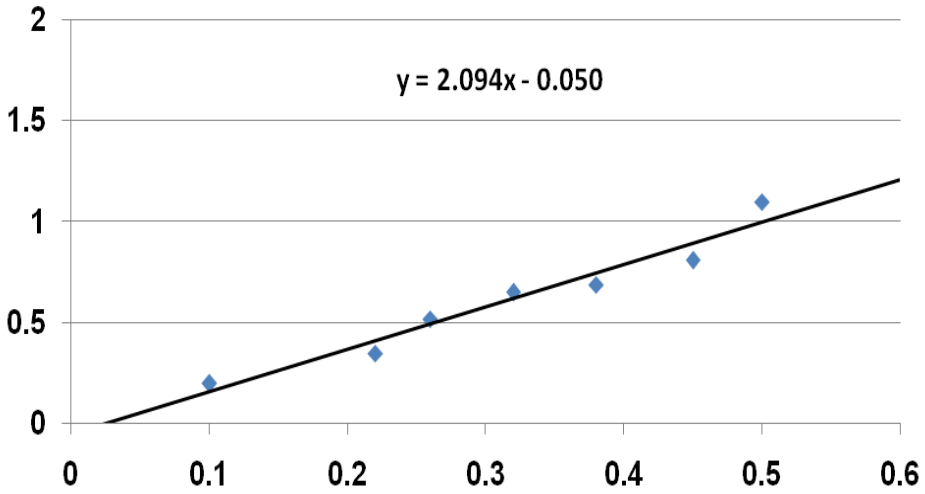
Szegedy et al, Going Deeper With Convolutions, CVPR 2015

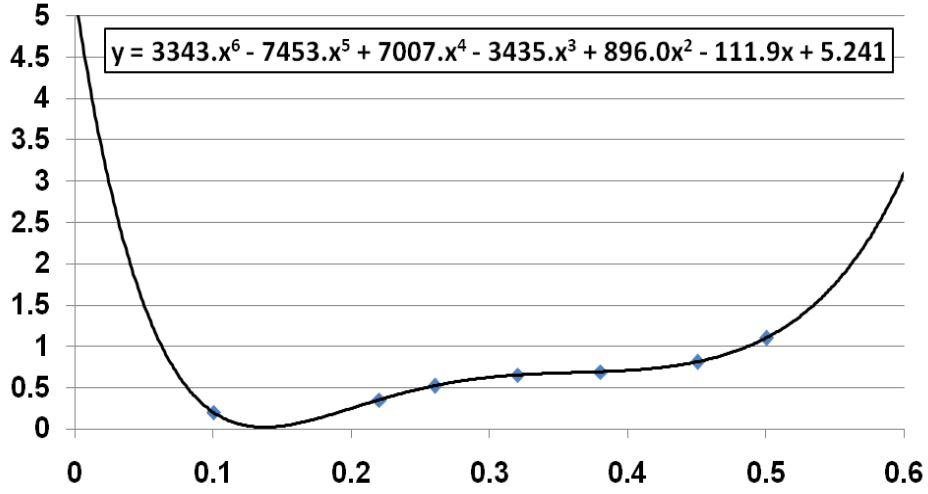
0.005%	1.5%	95.1%
Random guess	Pre Neural Networks	SE-ResNet

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

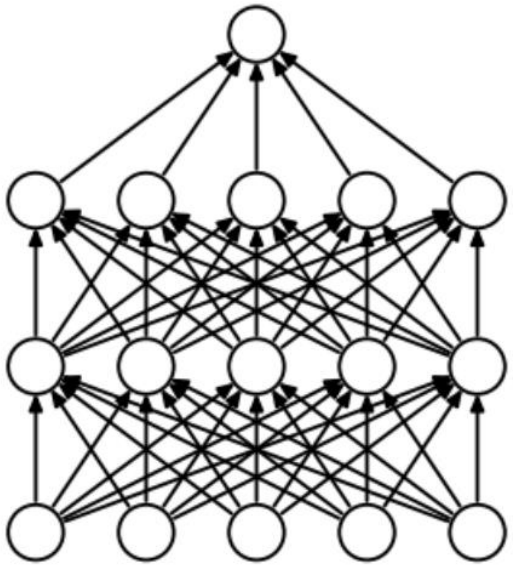




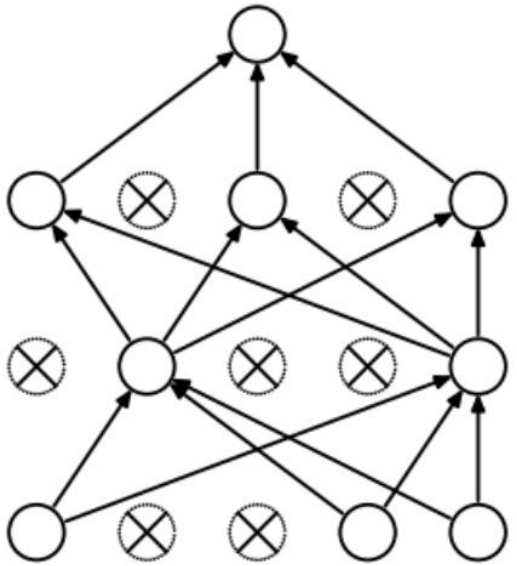
  - Polynomial on the right fits training data perfectly!
  - Which would you rather use to predict a new data point?

Prevent Overfitting?

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



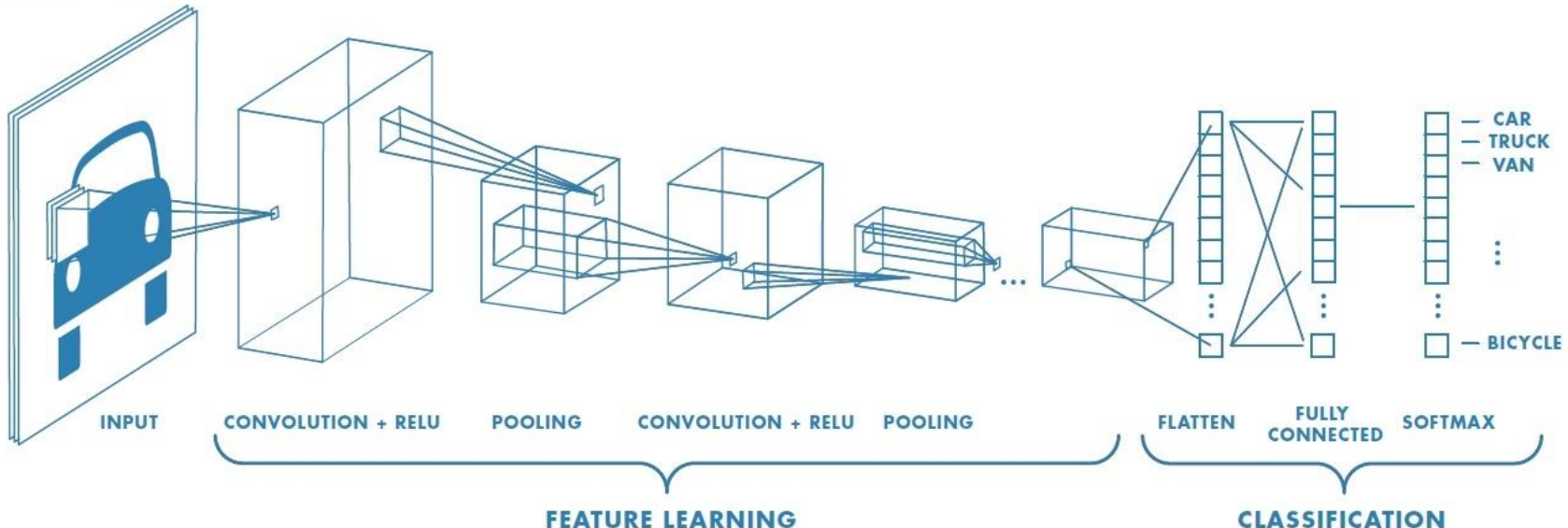
(a) Standard Neural Net



(b) After applying dropout.

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).



Not everything is classification

Come to Wednesday’s Lecture!

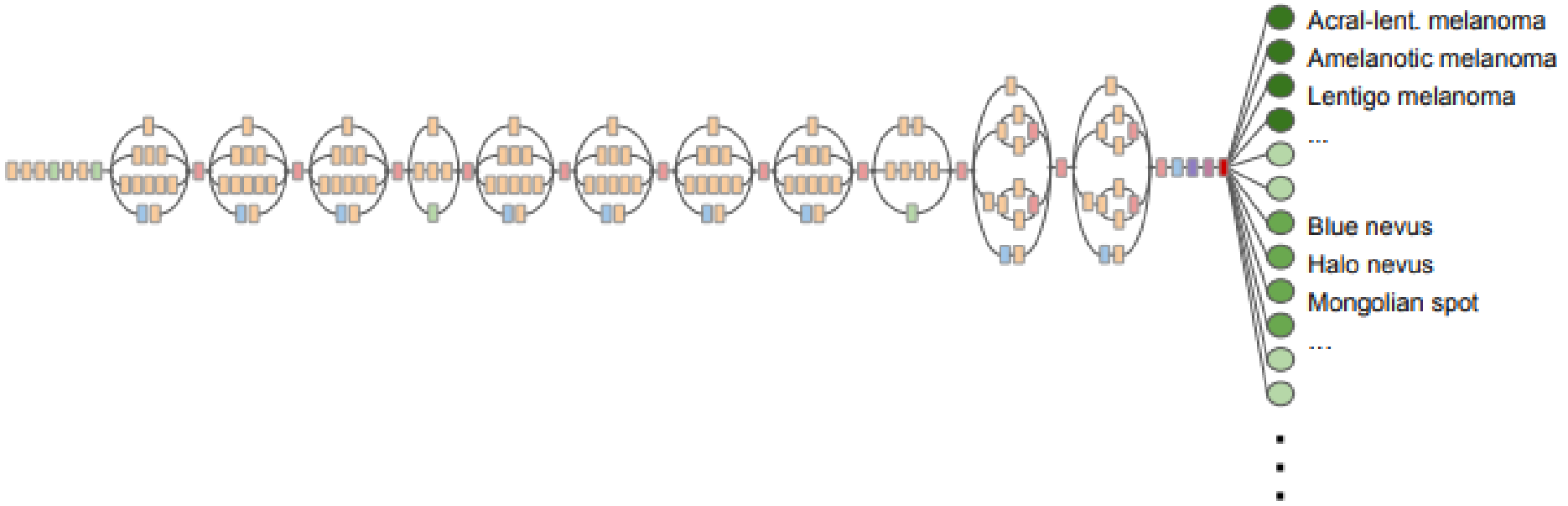
Piech

Detecting skin cancer

Skin Lesion Image

Deep Convolutional Neural Network (Inception-v3)

Training Classes (757)



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


ChatGPT