27: Intro to Deep Learning

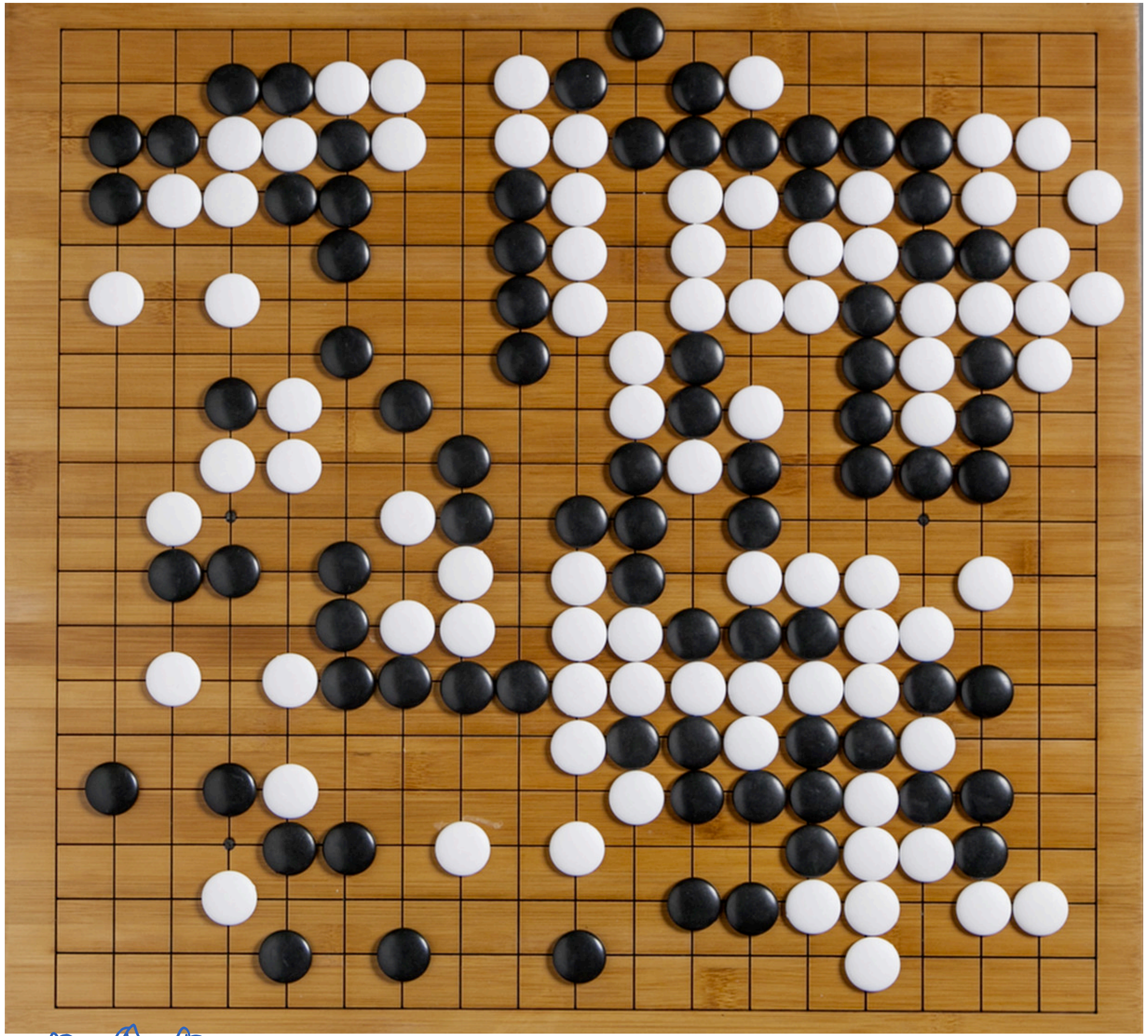
Lisa Yan

November 13, 2020

Quick slide reference

Deep Learning

Innovations in deep learning



AlphaGO (2016)

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

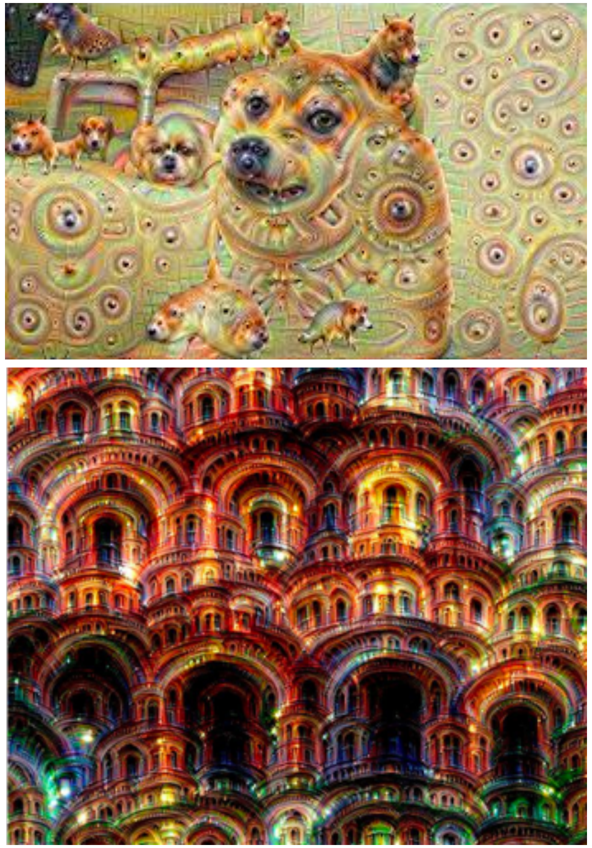
Errata (misspoke):

- 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)

Computers making art

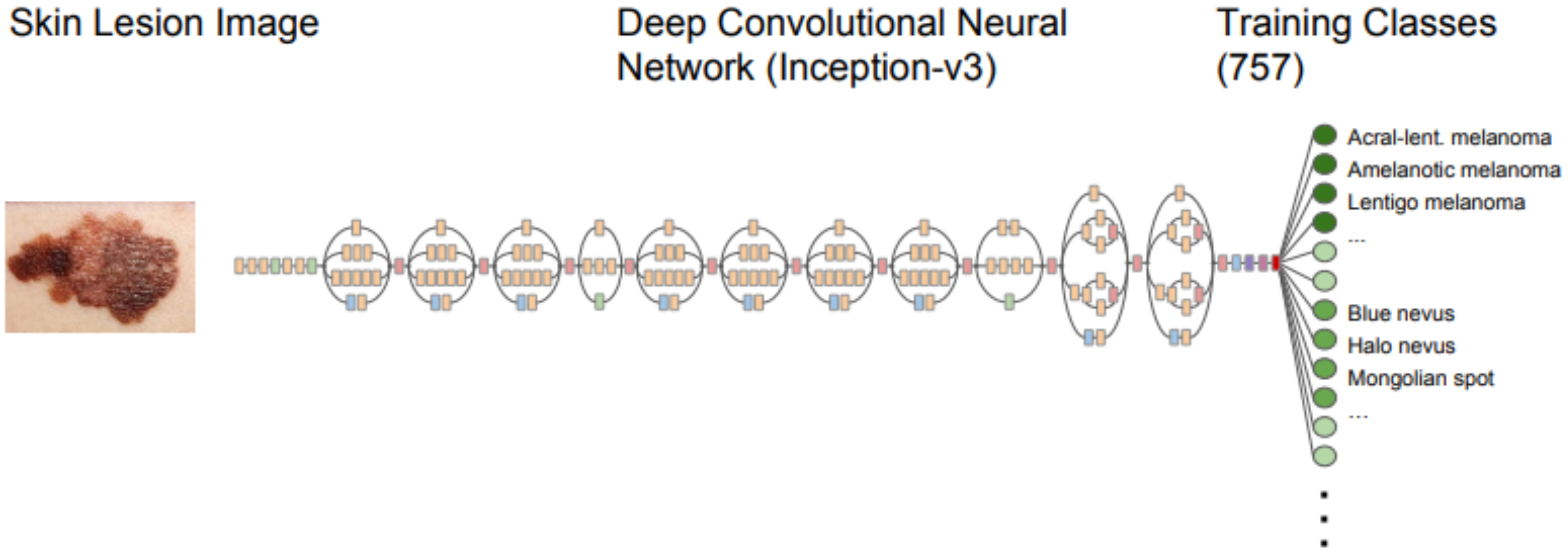
The Next Rembrandt
https://medium.com/@DutchDigital/the-next-rembrandt-bringing-the-old-master-back-to-life-35dfb1653597

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.

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.

$[1,0, \ldots, 1]$
$\boldsymbol{x}$, input

LOL

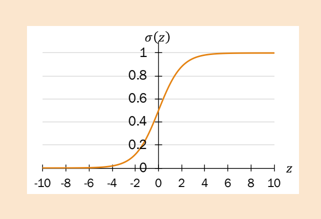
Lots of Logistic
(regressions)

$\hat{y}$, output
$P(Y|\boldsymbol{X} = \boldsymbol{x})$
$P(Y=1|X=x)$

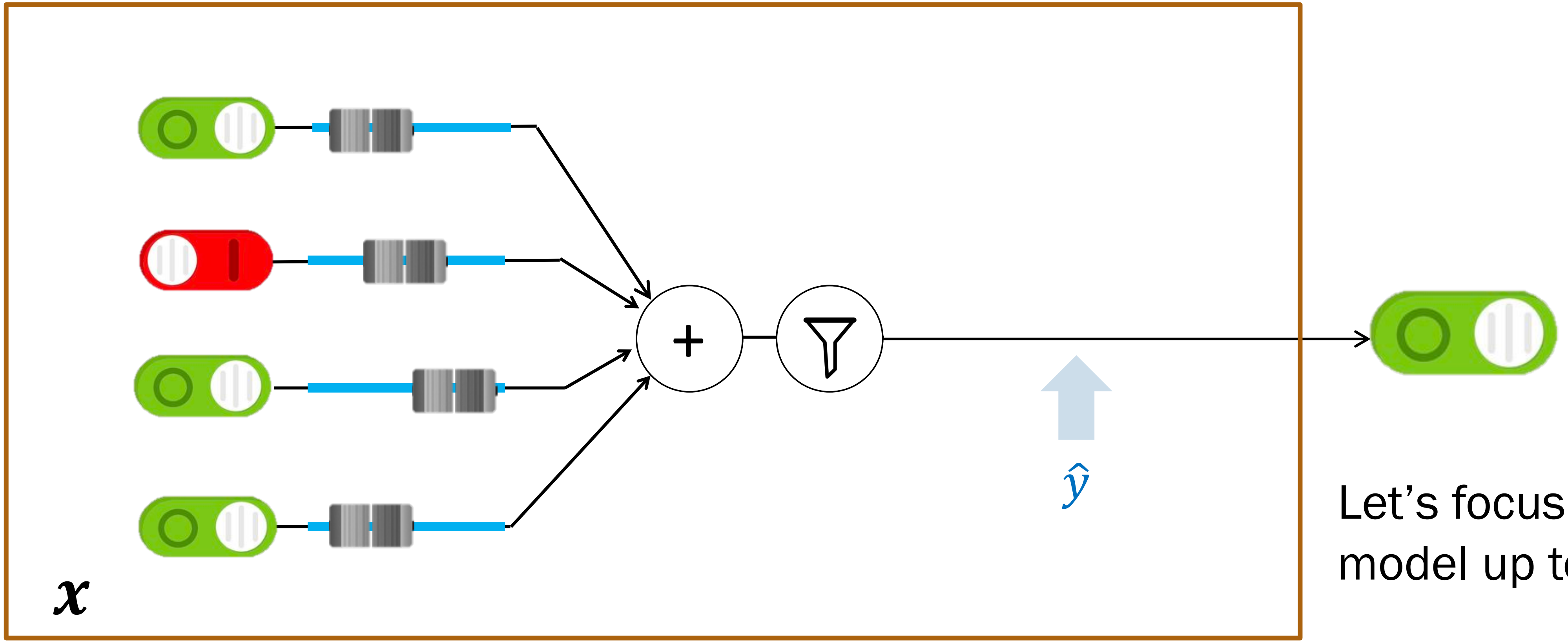
> 0.5?

Yes.
Predict 1

Logistic Regression Model

$\boldsymbol{X}$ ⇒ $\boxed{\theta_0 + \sum_{j=1}^{m} \theta_j X_j}$ ⇒ 

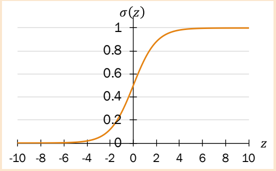
⇒ $\hat{y} = P(Y = 1|\boldsymbol{X})$ ⇒ $\hat{Y} = \arg\max_{y=\{0,1\}} P(Y \mid \boldsymbol{X})$



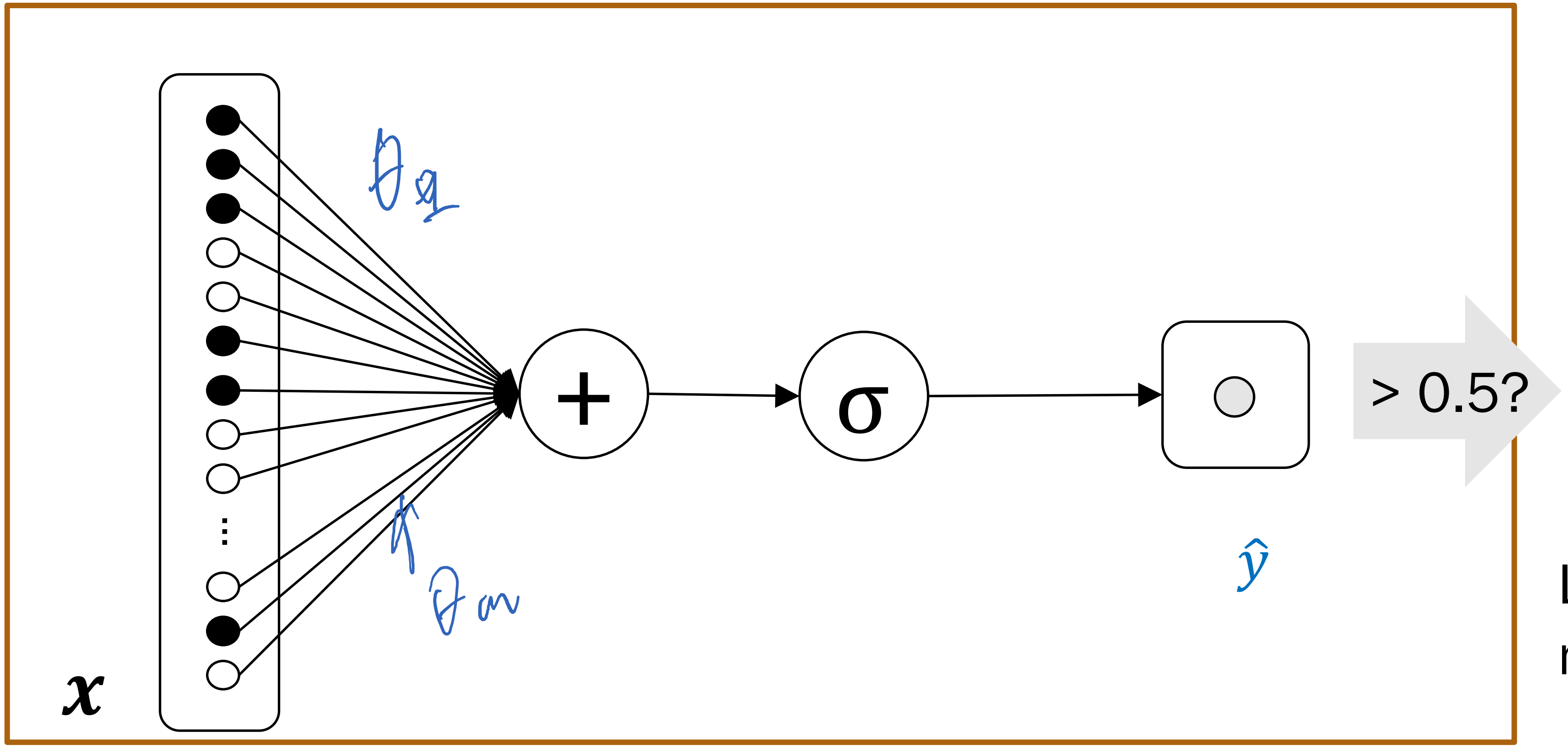
Let's focus on the model up to $\hat{y}$.

Logistic Regression Model

$\boldsymbol{X}$ → $\theta_0 + \sum_{j=1}^{m} \theta_j X_j$ →

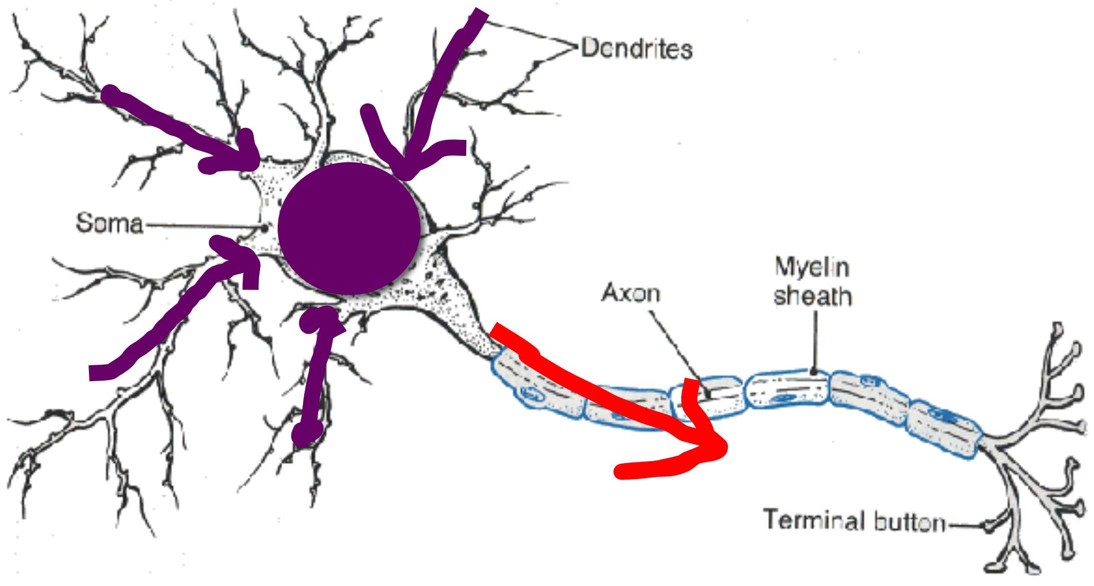


→ $\hat{y} = P(Y = 1|\boldsymbol{X})$ → $\hat{Y} = \arg\max_{y=\{0,1\}} P(Y \mid \boldsymbol{X})$

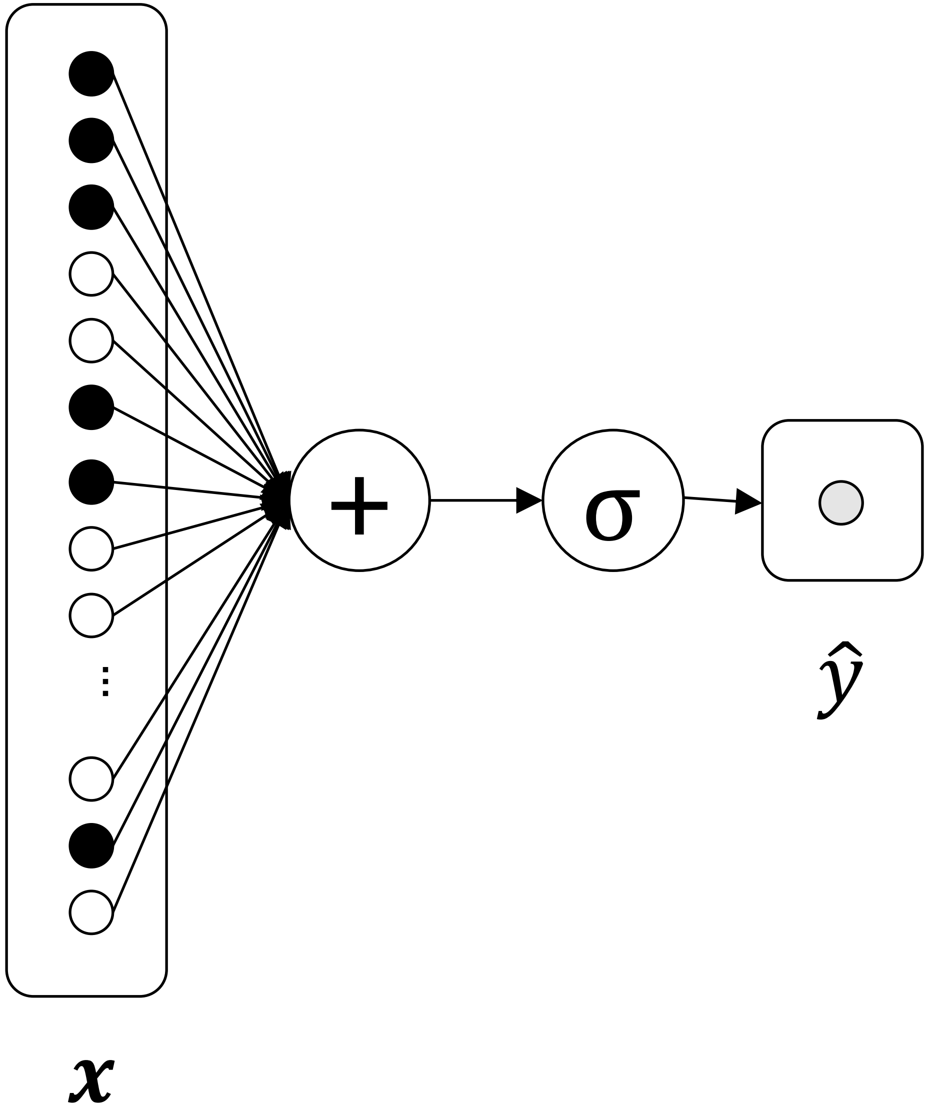


Let's focus on the model up to $\hat{y}$.

One neuron = One logistic regression

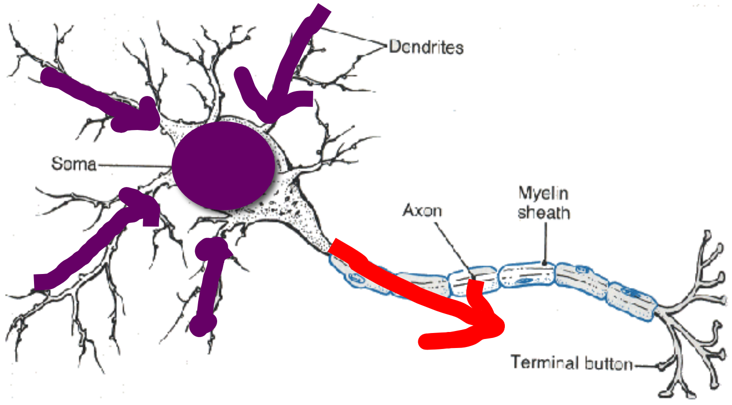


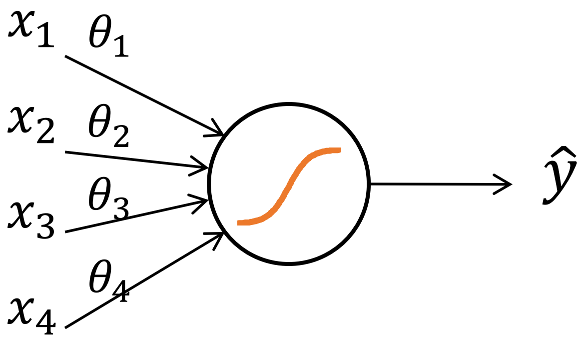
=



Biological basis for neural networks

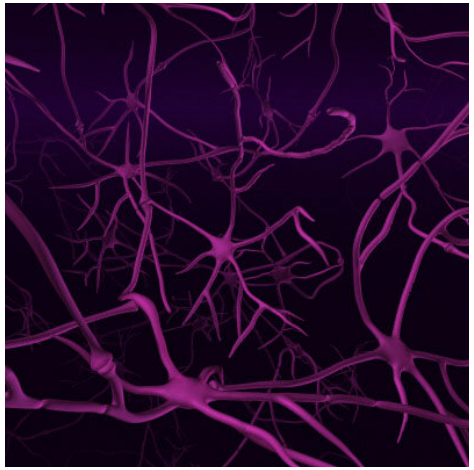
A neuron



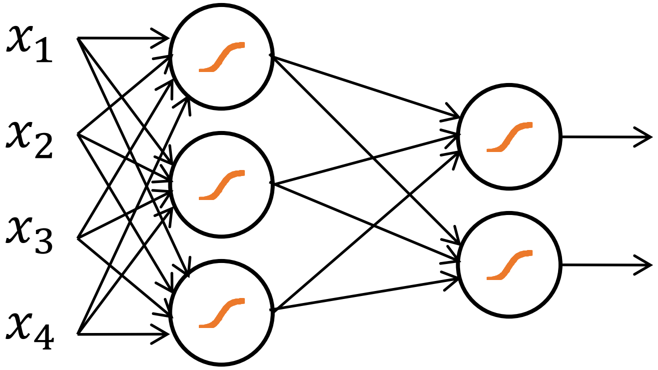


One neuron =
one logistic
regression

Your brain

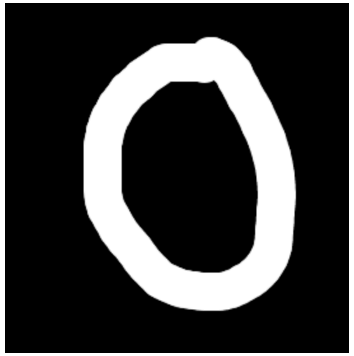


(actually, probably someone else's brain)



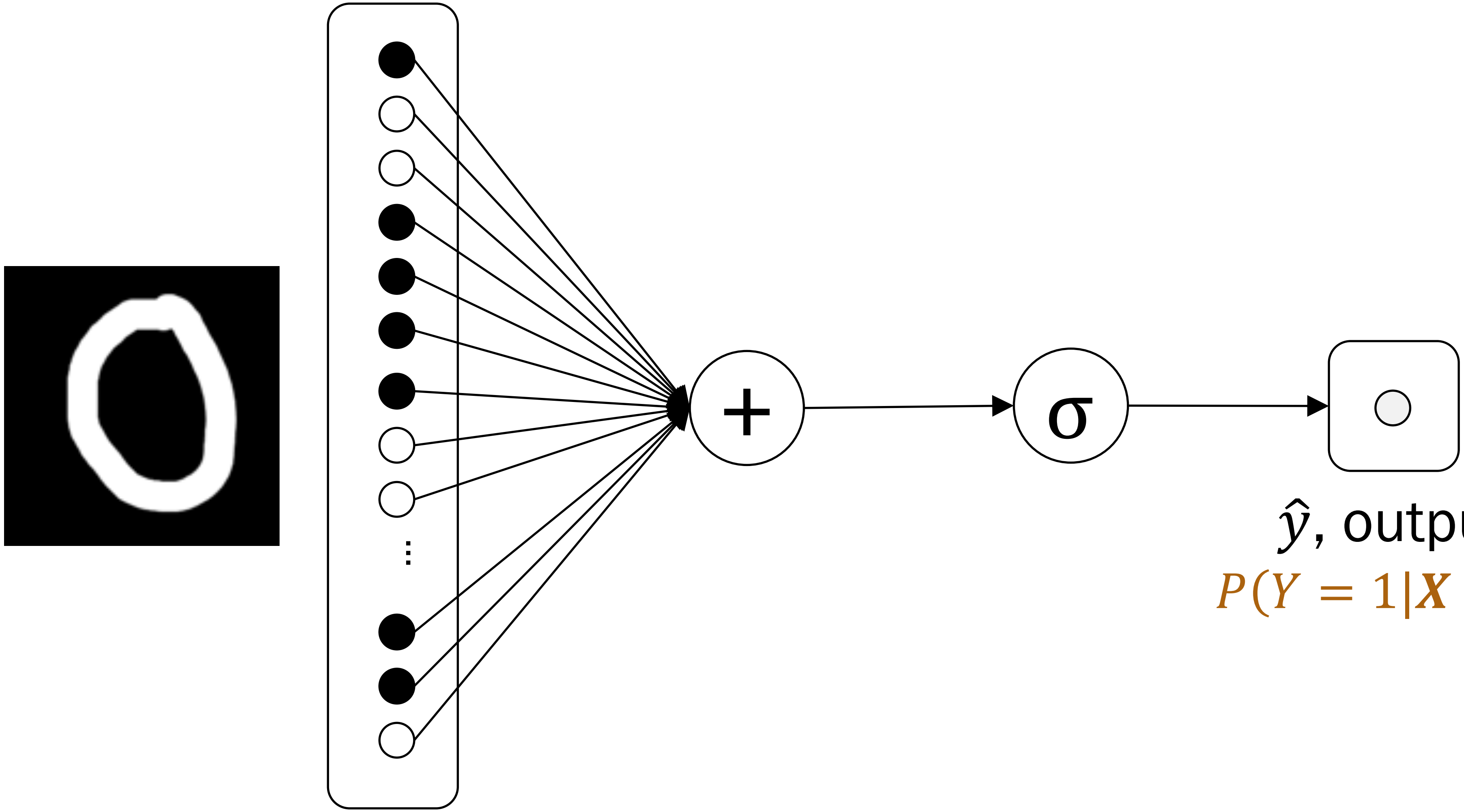
Neural network =
many logistic
regressions

Digit recognition example

Input image	Input feature vector	Output label
	$\boldsymbol{x}^{(i)} = [0,0,0,0, \dots, 1,0,0,1, \dots, 0,0,1,0]$	$y^{(i)} = 0$
	$\boldsymbol{x}^{(i)} = [0,0,1,1, \dots, 0,1,1,0, \dots, 0,1,0,0]$	$y^{(i)} = 1$

We make feature vectors from (digitized) pictures of numbers.

Logistic Regression



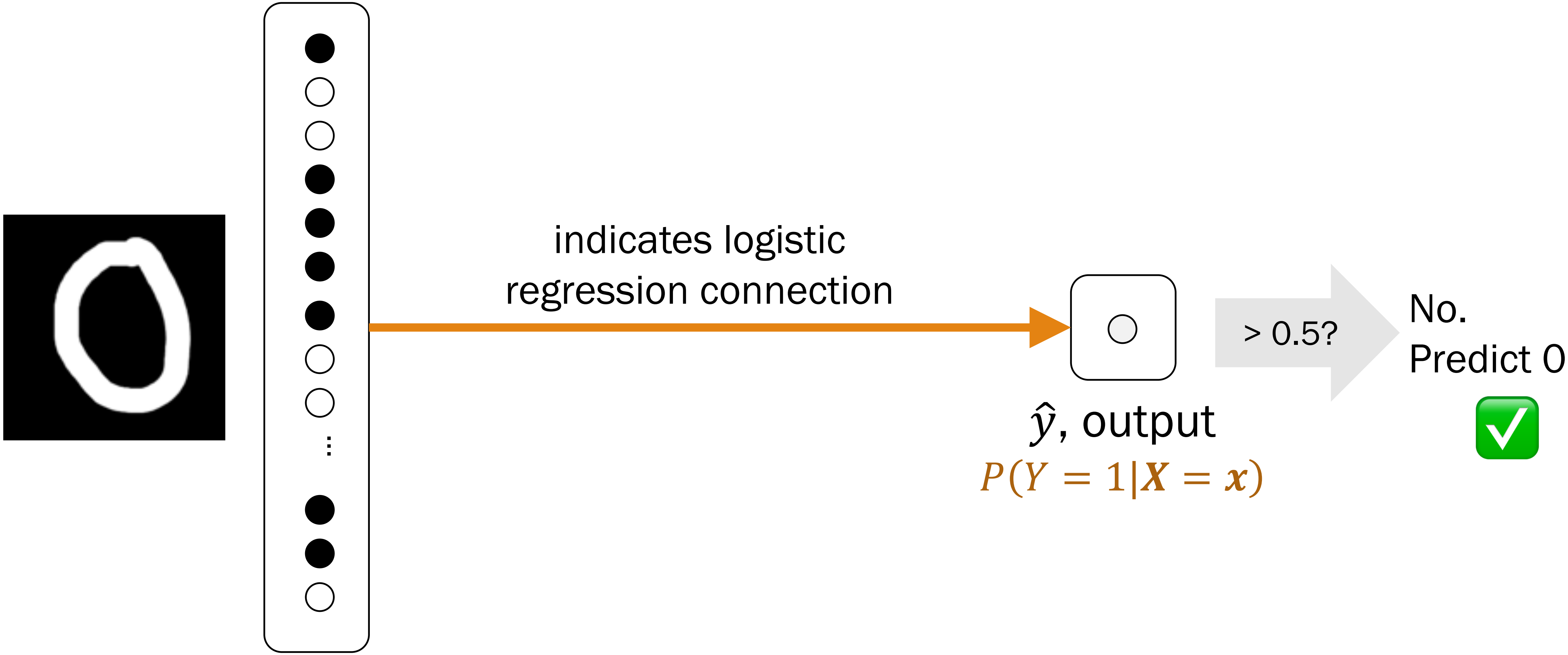
$\hat{y}$, output

$P(Y = 1|\boldsymbol{X} = \boldsymbol{x})$

$\boldsymbol{x}$, input features

(pixels, on/off)

Logistic Regression



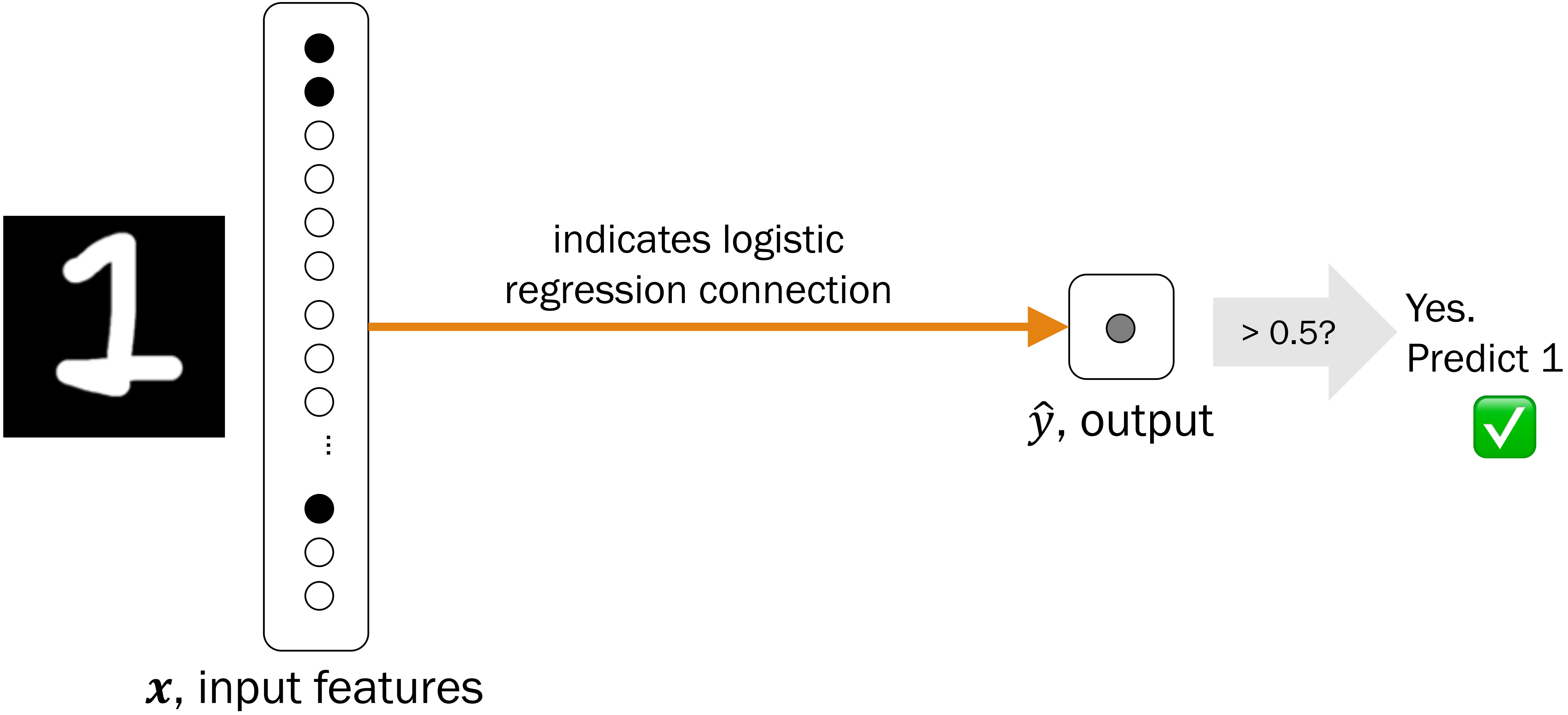
$\hat{y}$, output

$P(Y = 1|\boldsymbol{X} = \boldsymbol{x})$

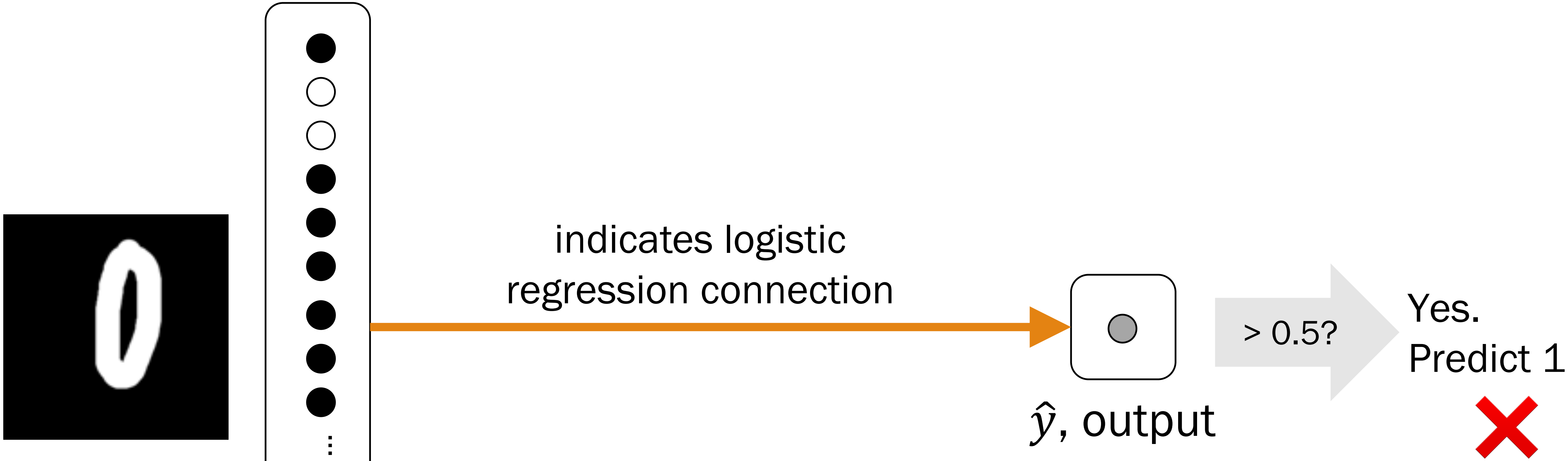
$\boldsymbol{x}$, input features

(pixels, on/off)

Logistic Regression



Logistic Regression



$\boldsymbol{x}$, input features

$\hat{y}$, output

What can we do to **increase** complexity of our model?

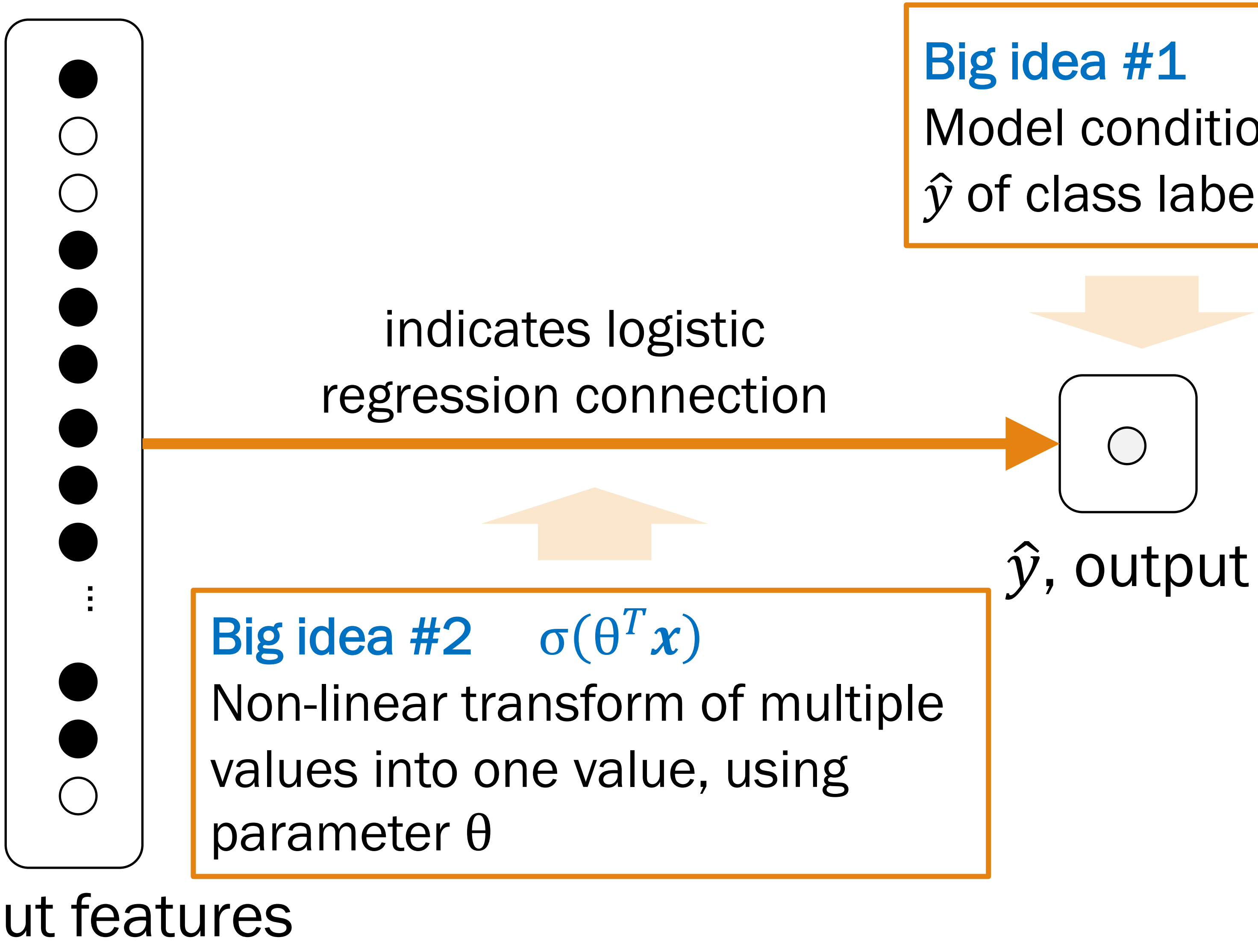
Take two big ideas from Logistic Regression

Review

$=1$

Big idea #1 $P(Y|\boldsymbol{X} = \boldsymbol{x})$

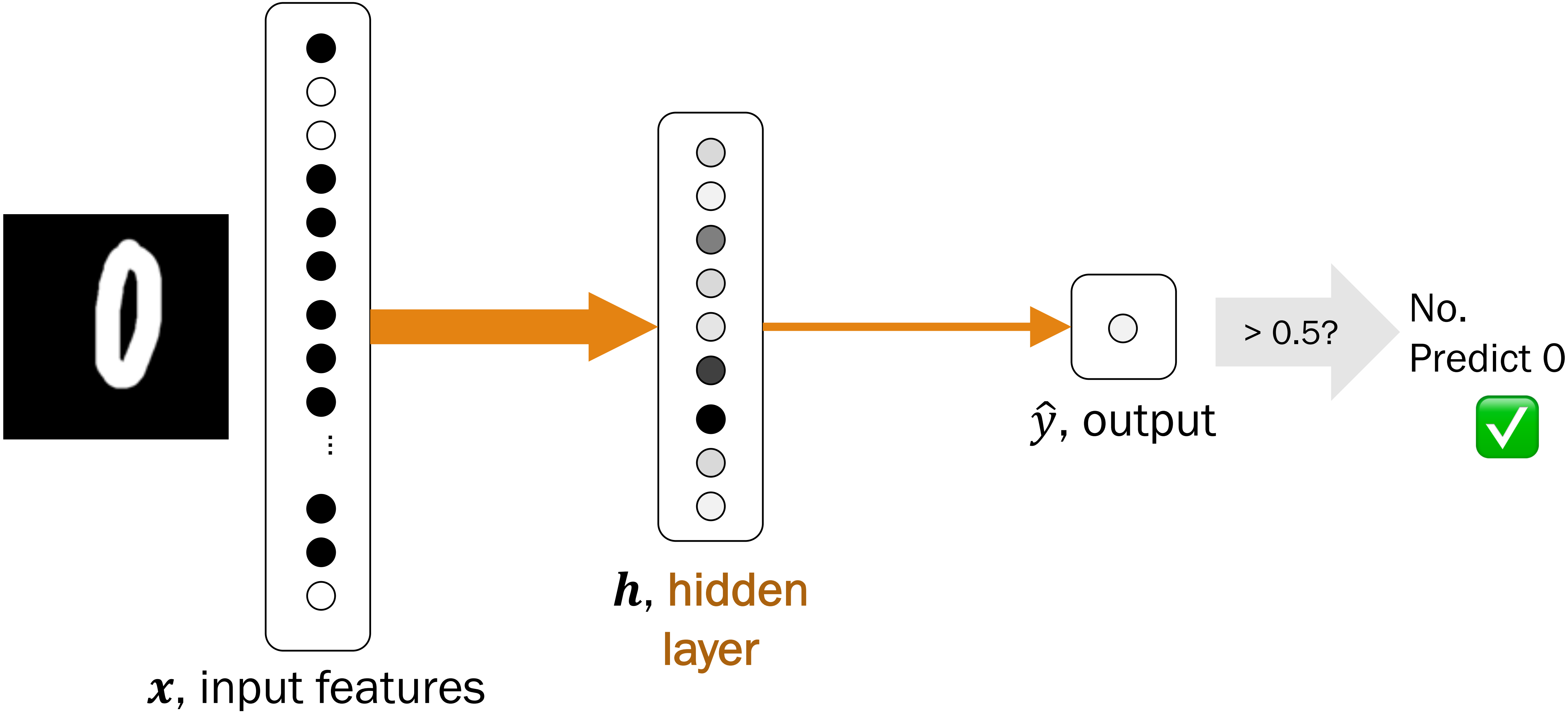
Model conditional probability $\hat{y}$ of class label given input



Big idea #2 $\sigma(\theta^T \boldsymbol{x})$

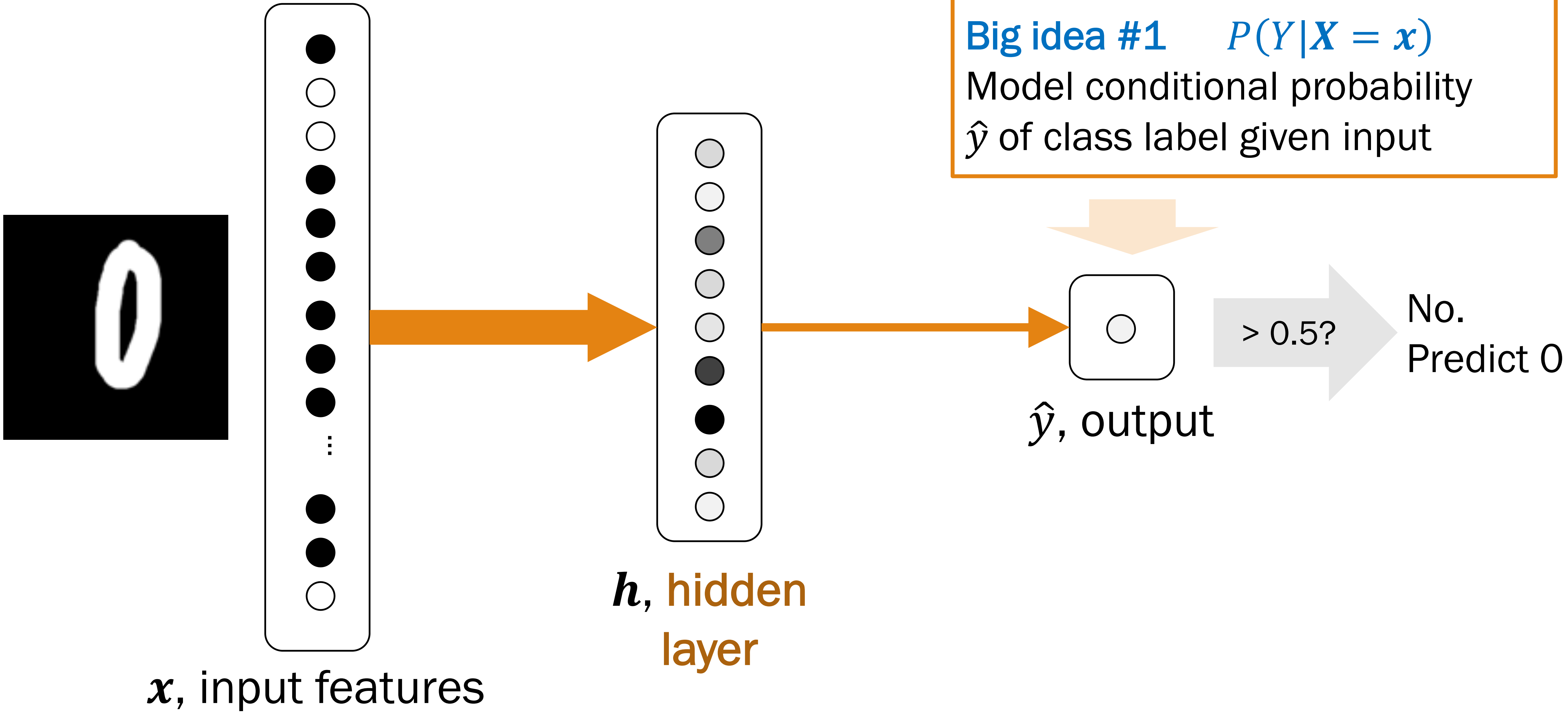
Non-linear transform of multiple values into one value, using parameter θ

Introducing: The Neural network

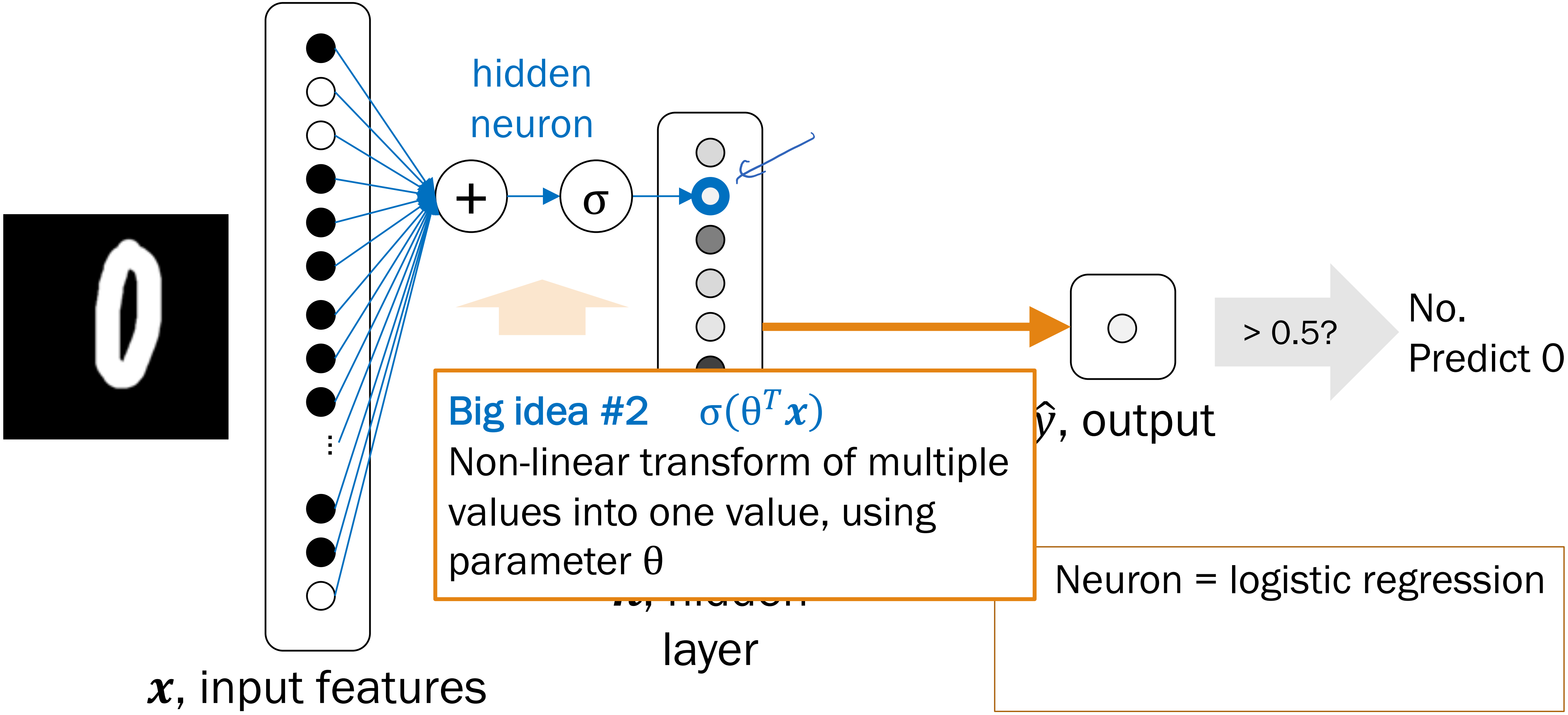


Neural network

Big idea #1 $P(Y|\boldsymbol{X} = \boldsymbol{x})$
Model conditional probability $\hat{y}$ of class label given input



Feed neurons into other neurons

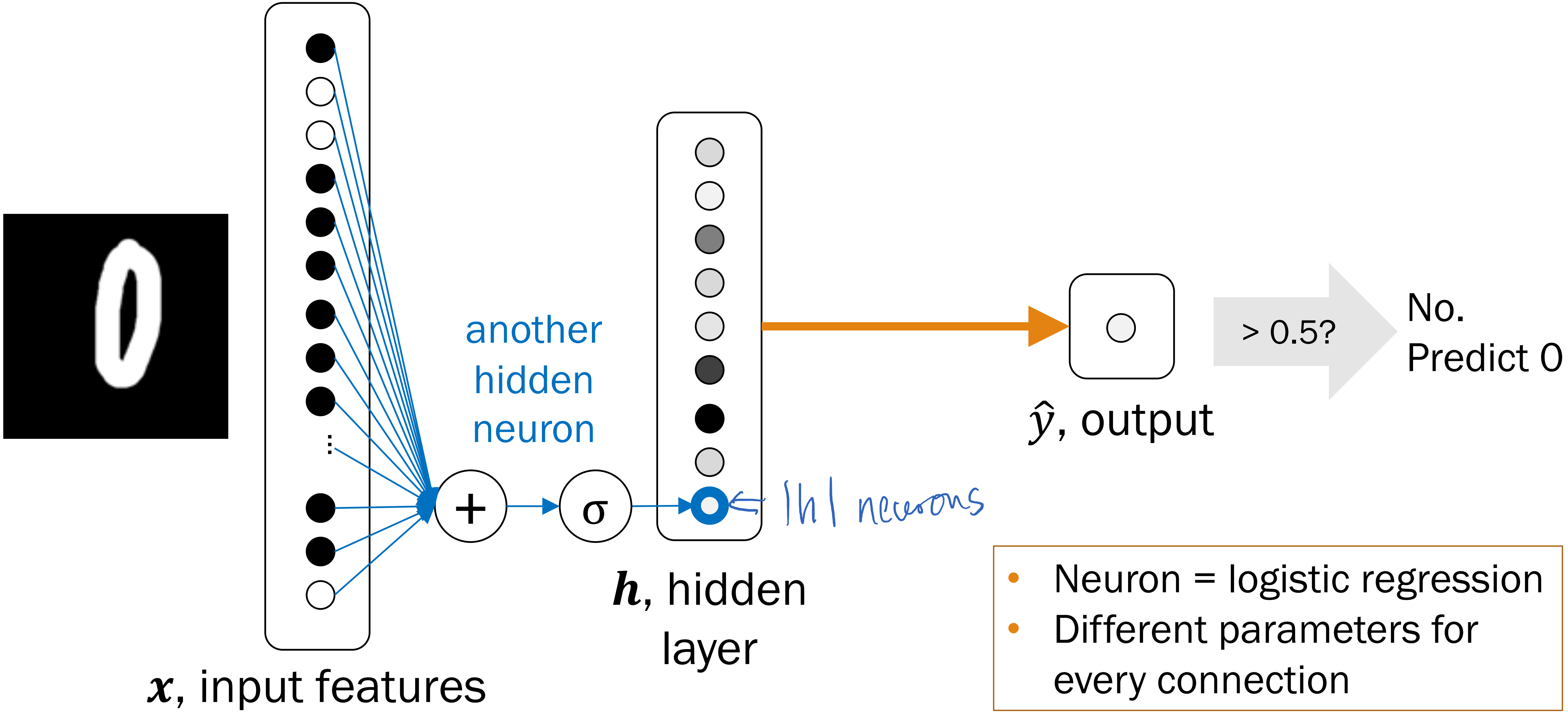


Big idea #2 $\sigma(\theta^T \boldsymbol{x})$

Non-linear transform of multiple values into one value, using parameter θ

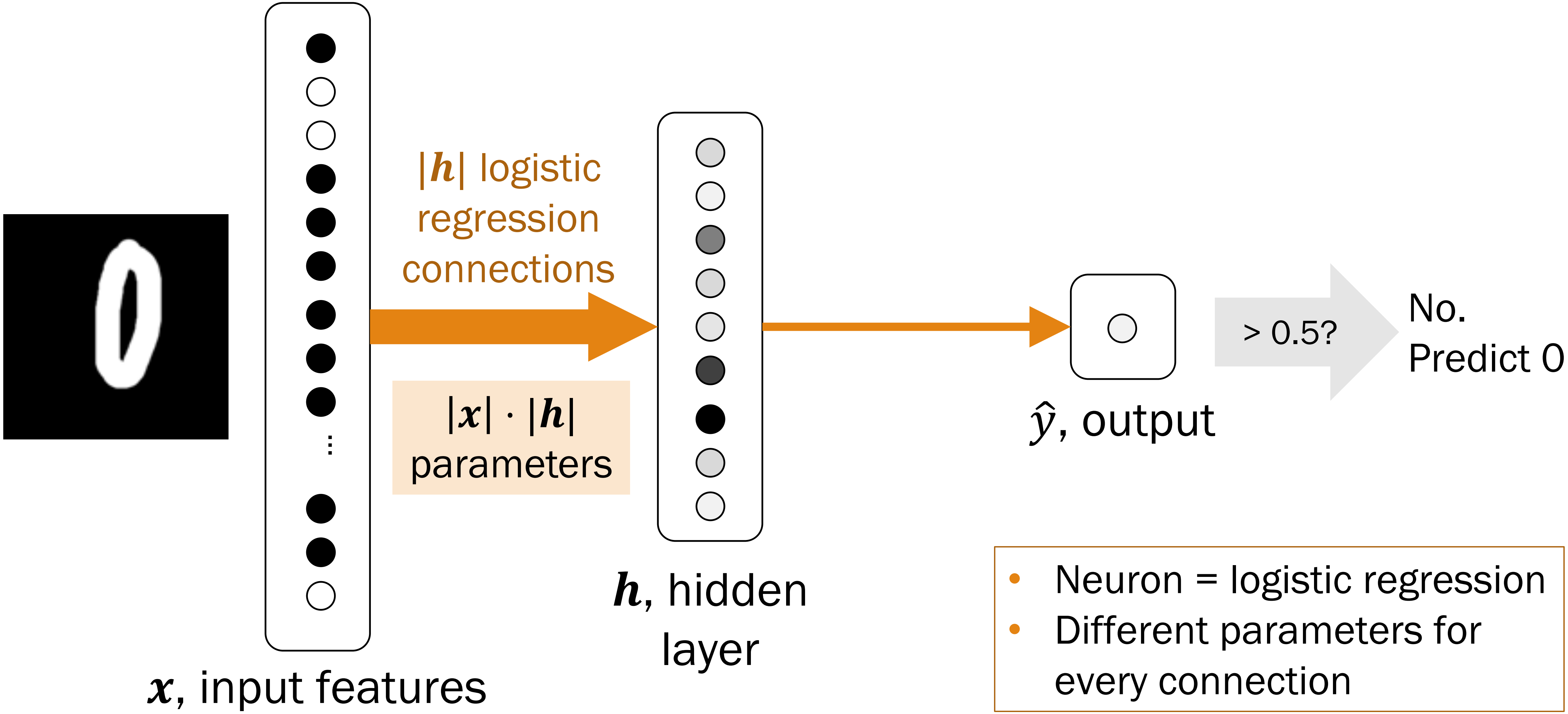
Neuron = logistic regression

Feed neurons into other neurons



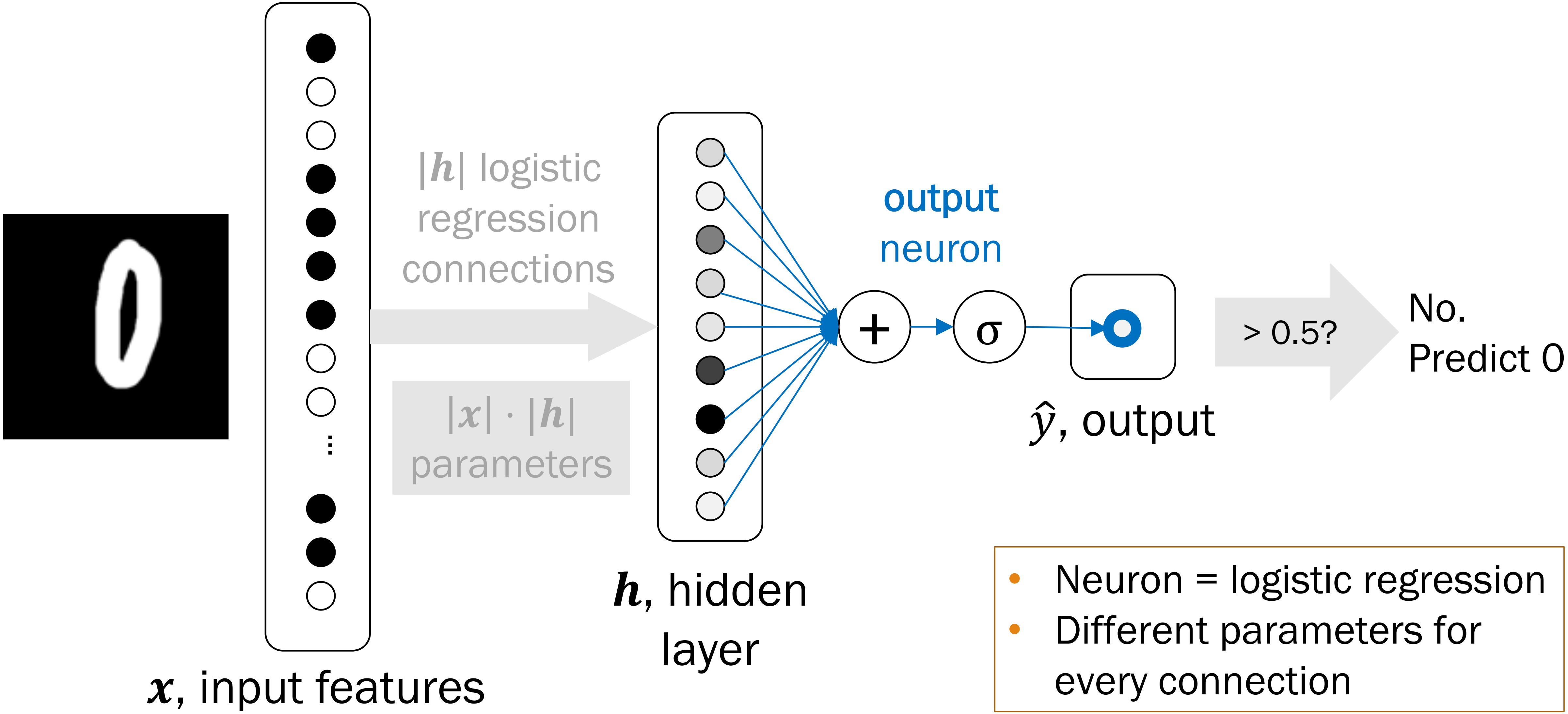
- Neuron = logistic regression
- Different parameters for every connection

Feed neurons into other neurons



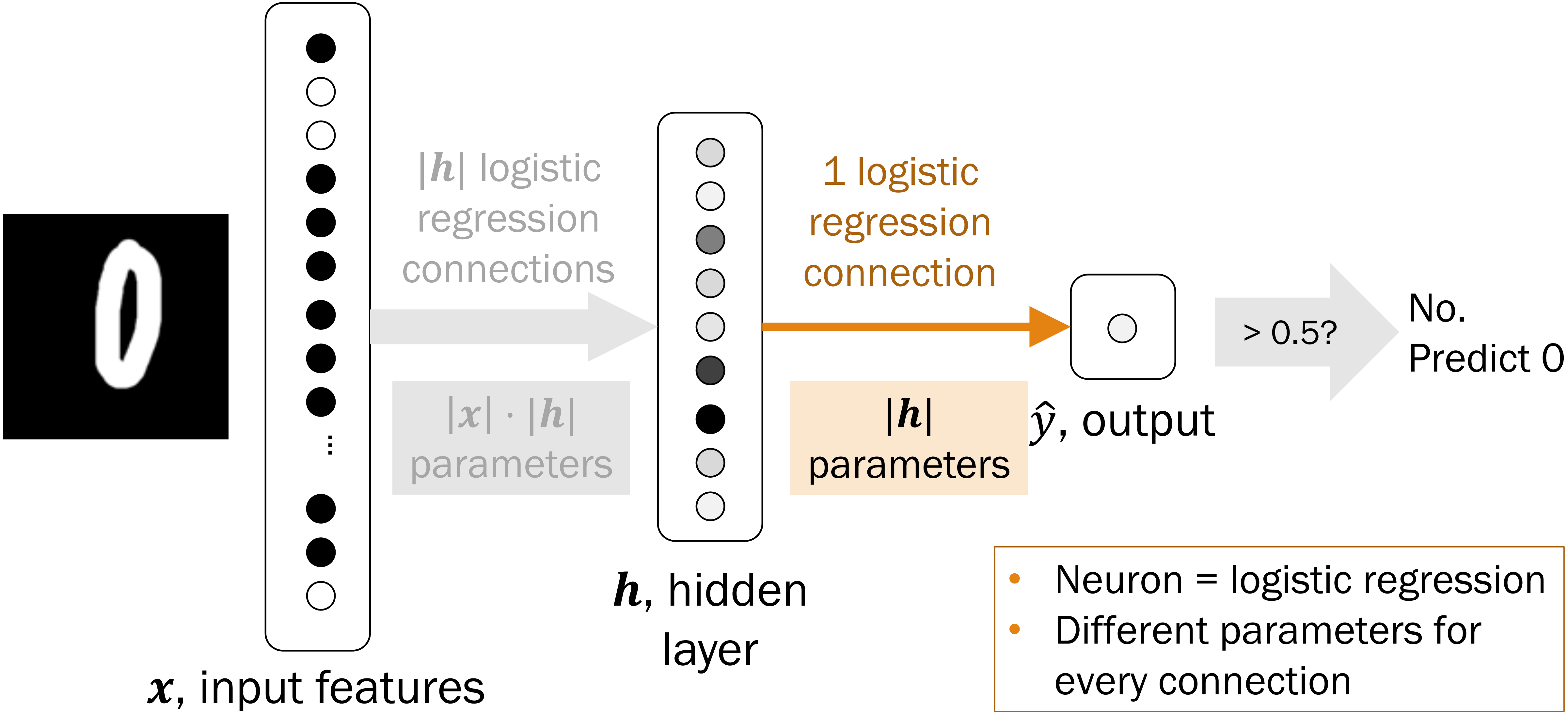
- Neuron = logistic regression
- Different parameters for every connection

Feed neurons into other neurons



- Neuron = logistic regression
- Different parameters for every connection

Feed neurons into other neurons



- Neuron = logistic regression
- Different parameters for every connection

Think

Slide 26 asks you to think over by yourself.

Post any clarifications in chat!

Think by yourself: 2 min

(by yourself)

Why doesn't a linear model introduce "complexity"?

Neural network:

1. for $j = 1, \dots, |\boldsymbol{h}|$:

$$\boxed{h_j = \sigma\left(\theta_j^{(h)^T} \boldsymbol{x}\right)}$$

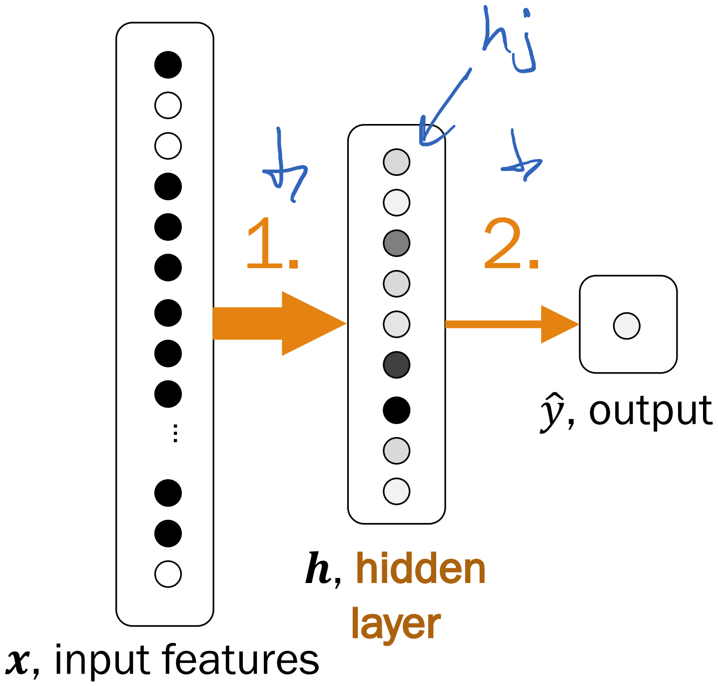
2. $\hat{y} = \sigma\left(\theta^{(\hat{y})^T} \boldsymbol{h}\right) = P(Y = 1 | \boldsymbol{X} = \boldsymbol{x})$

Linear network:

1. for $j = 1, \dots, |\boldsymbol{h}|$:

$$\boxed{h_j = \theta_j^{(h)^T} \boldsymbol{x}}$$

2. $\hat{y} = \sigma\left(\theta^{(\hat{y})^T} \boldsymbol{h}\right) = P(Y = 1 | \boldsymbol{X} = \boldsymbol{x})$



(by yourself)

Why doesn't a linear model introduce "complexity"?

Neural network:

$|x| \cdot |h| + |h|$

1. for $j = 1, \dots, |\boldsymbol{h}|$:

   $$\boxed{h_j = \sigma\left(\theta_j^{(h)^T} \boldsymbol{x}\right)}$$

2. $\hat{y} = \sigma\left(\theta^{(\hat{y})^T} \boldsymbol{h}\right) = P(Y = 1 | \boldsymbol{X} = \boldsymbol{x})$

Linear network:

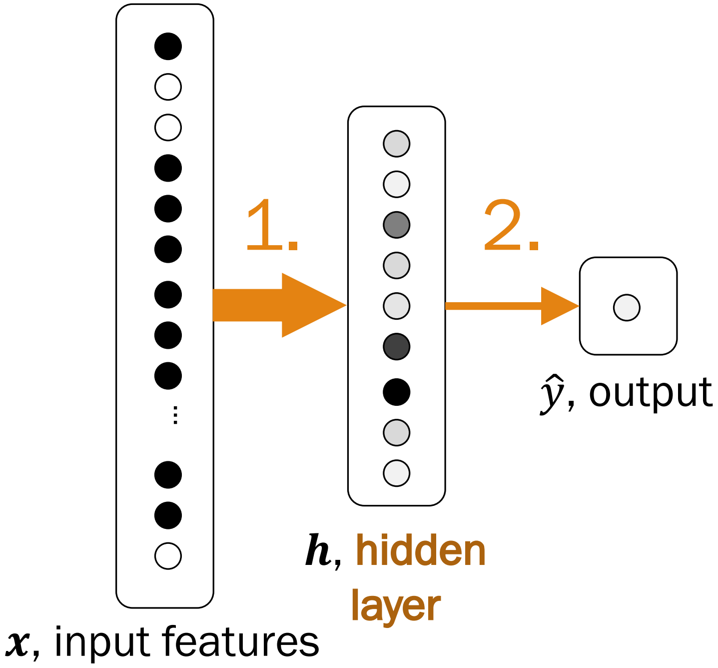
$$\underbrace{\left[\theta^{(\hat{y})T}\right]}_{1 \times |h|} \overbrace{\begin{bmatrix} \theta_1^{(h)T} \\ \theta_2^{(h)T} \\ \vdots \\ \theta_{|h|}^{(h)T} \end{bmatrix}}^{|h| \times |x|} x$$

h

1. for $j = 1, \dots, |\boldsymbol{h}|$:

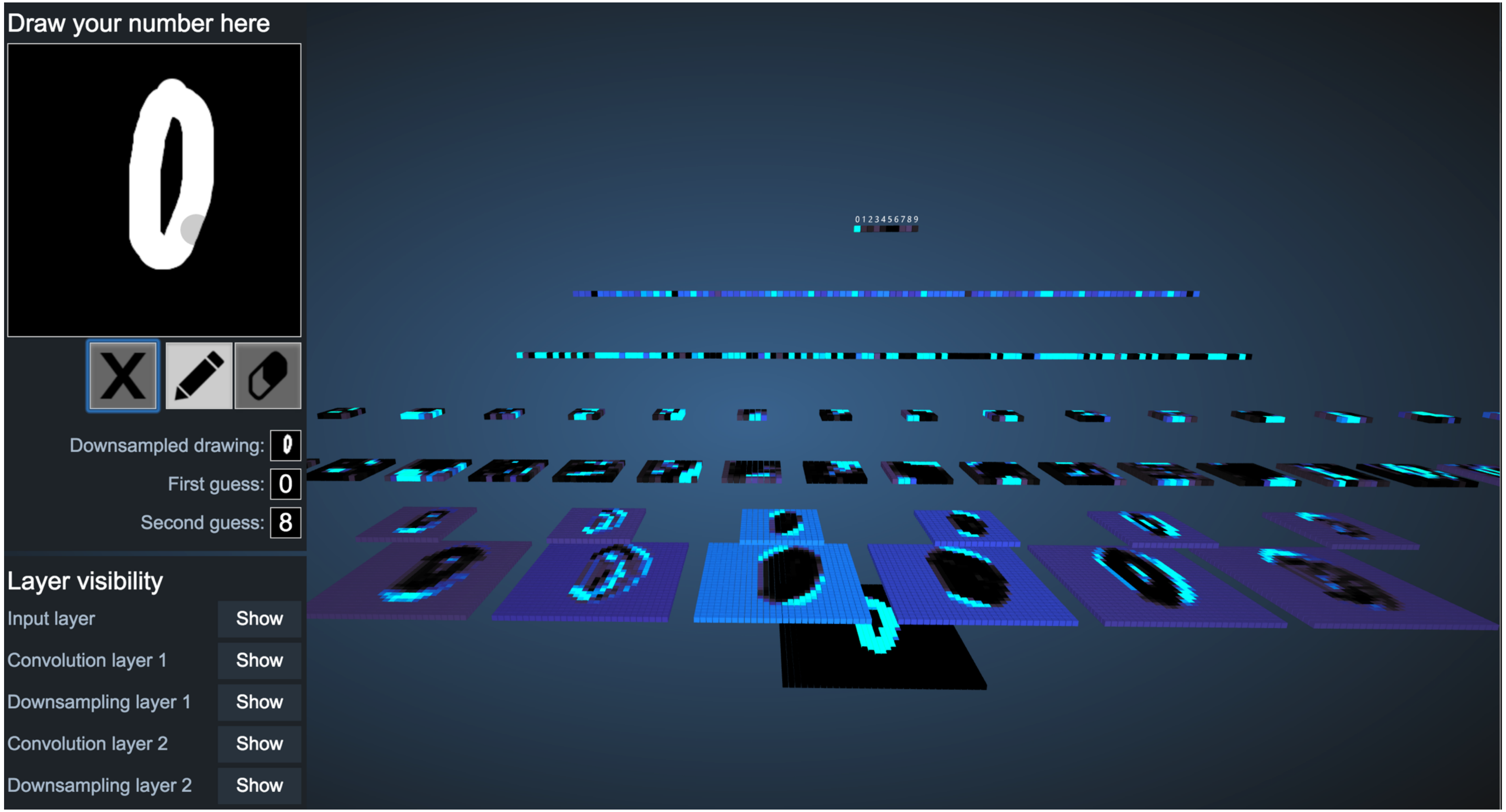
   $$\boxed{h_j = \theta_j^{(h)^T} \boldsymbol{x}}$$

2. $\hat{y} = \sigma\left(\theta^{(\hat{y})^T} \boldsymbol{h}\right) = P(Y = 1 | \boldsymbol{X} = \boldsymbol{x})$

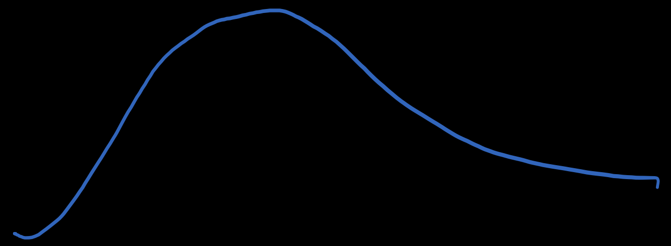


The linear model is effectively a single logistic regression with $|\boldsymbol{x}|$ parameters.

Demonstration



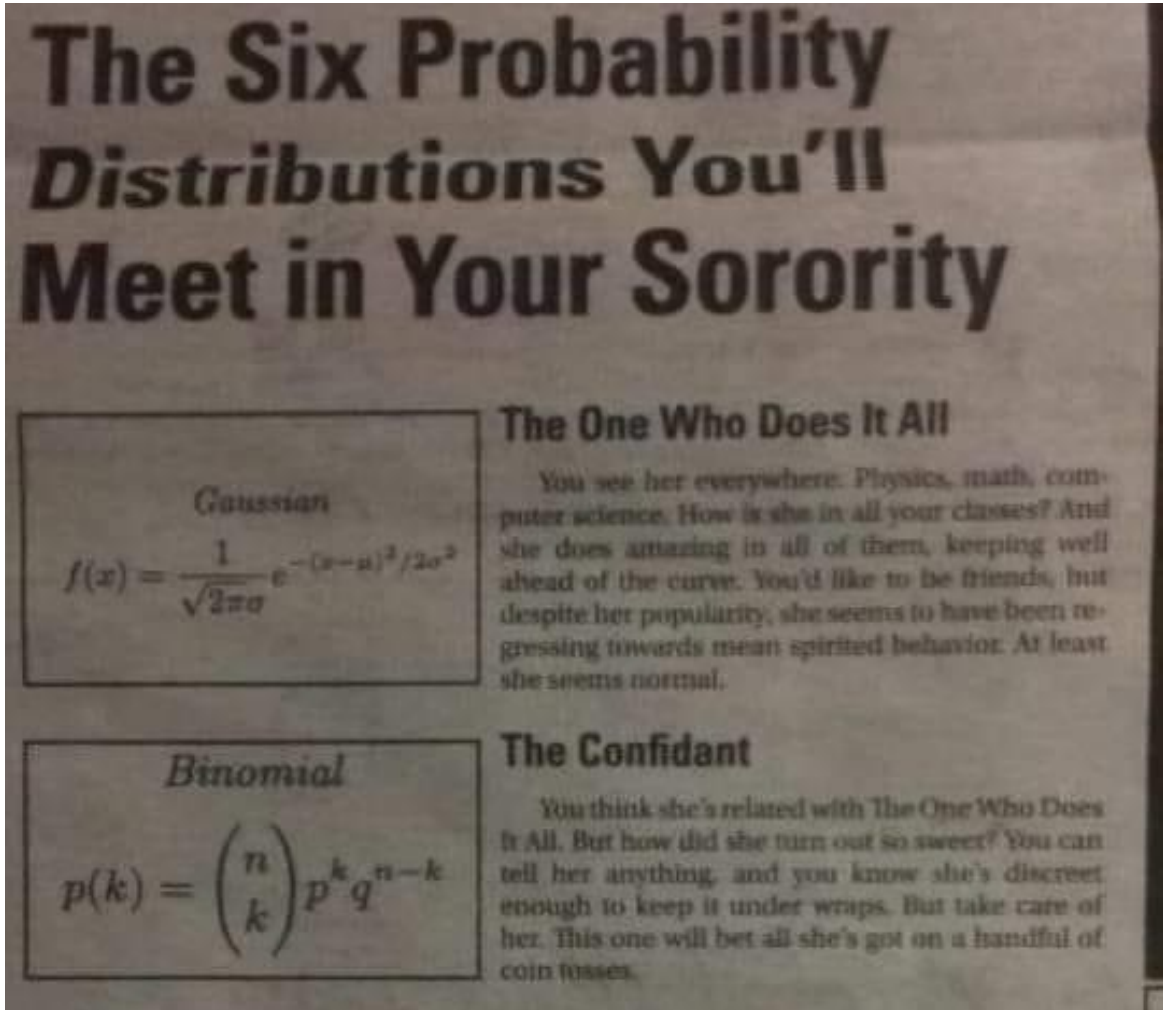
http://scs.ryerson.ca/~aharley/vis/conv/



para NORMAL

Interlude for jokes

Probability as college students



The Six Probability Distributions You'll Meet in Your Sorority

Gaussian

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}$$

The One Who Does It All

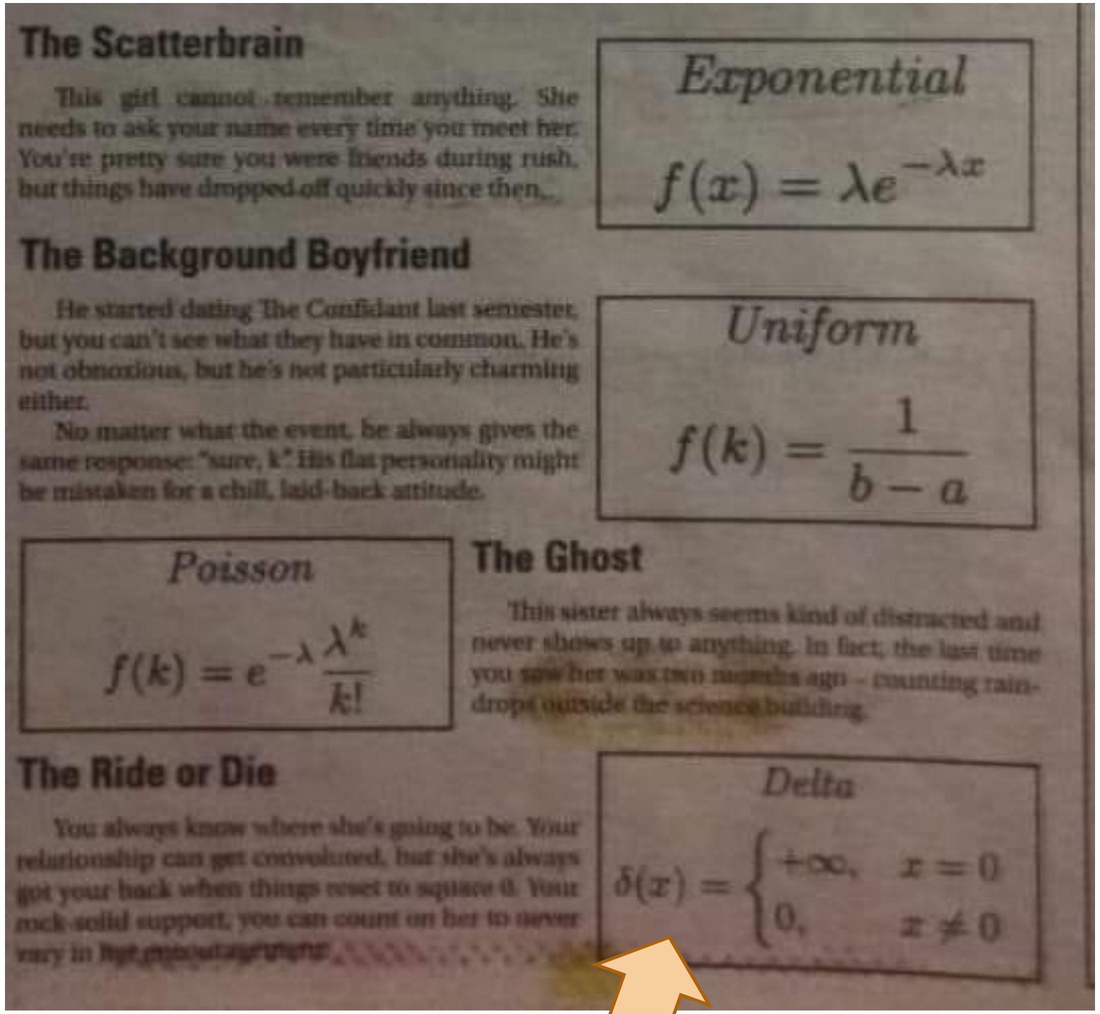
You see her everywhere: Physics, math, computer science. How is she in all your classes? And she does amazing in all of them, keeping well ahead of the curve. You'd like to be friends, but despite her popularity, she seems to have been regressing towards mean spirited behavior. At least she seems normal.

Binomial

$$p(k) = \binom{n}{k} p^k q^{n-k}$$

The Confidant

You think she's related with The One Who Does It All. But how did she turn out so sweet? You can tell her anything, and you know she's discreet enough to keep it under wraps. But take care of her. This one will bet all she's got on a handful of coin tosses.



The Scatterbrain

This girl cannot remember anything. She needs to ask your name every time you meet her. You're pretty sure you were friends during rush, but things have dropped off quickly since then.

Exponential

$$f(x) = \lambda e^{-\lambda x}$$

The Background Boyfriend

He started dating The Confidant last semester, but you can't see what they have in common. He's not obnoxious, but he's not particularly charming either.

No matter what the event, he always gives the same response: "sure, k." His flat personality might be mistaken for a chill, laid-back attitude.

Uniform

$$f(k) = \frac{1}{b-a}$$

Poisson

$$f(k) = e^{-\lambda} \frac{\lambda^k}{k!}$$

The Ghost

This sister always seems kind of distracted and never shows up to anything. In fact, the last time you saw her was two months ago – counting raindrops outside the science building.

The Ride or Die

You always know where she's going to be. Your relationship can get convoluted, but she's always got your back when things reset to square 0. Your rock-solid support, you can count on her to never vary in [illegible]

Delta

$$\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases}$$

(A useful construct that connects discrete PMF to continuous PDF)

Neural networks

A neural network (like logistic regression) gets intelligence from its parameters θ.

Training

- Learn parameters θ
- Find θ_{MLE} that maximizes likelihood of training data (MLE)

Testing/ Prediction

For input feature vector $\boldsymbol{X} = \boldsymbol{x}$:

- Use parameters to compute $\hat{y} = P(Y = 1 | \boldsymbol{X} = \boldsymbol{x})$
- Classify instance as: $\begin{cases} 1 & \hat{y} > 0.5 \\ 0 & \text{otherwise} \end{cases}$

Neural networks

A neural network (like logistic regression) gets intelligence from its parameters θ.

Training

- Learn parameters θ
- Find θ_{MLE} that maximizes likelihood of training data (MLE)

How do we learn the $|\boldsymbol{x}| \cdot |\boldsymbol{h}| + |\boldsymbol{h}|$ parameters?

Gradient ascent + chain rule!

Training: Logistic Regression

Review

1. Optimization problem:

$$\theta_{MLE} = \arg\max_{\theta} \prod_{i=1}^{n} f\left(y^{(i)} \mid \boldsymbol{x}^{(i)}, \theta\right) = \arg\max_{\theta} LL(\theta)$$

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

$$\hat{y} = \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right) = P(Y = 1 \mid \boldsymbol{X} = \boldsymbol{x})$$

2. Compute gradient

Find $|\boldsymbol{x}|$ parameters

3. Optimize

```
initialize params
repeat many times:
  compute gradient
   params += η * gradient
```

Training: Neural networks

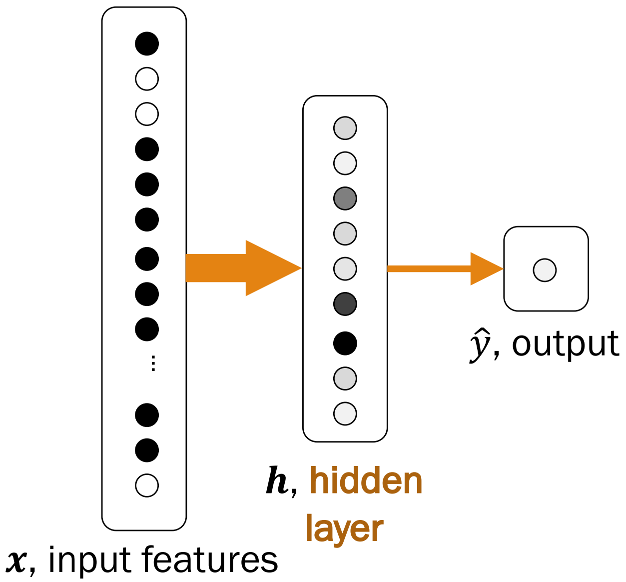
1. Optimization problem:

$$\theta_{MLE} = \arg\max_{\theta} \prod_{i=1}^{n} f\left(y^{(i)} \mid \boldsymbol{x}^{(i)}, \theta\right) = \arg\max_{\theta} \; LL(\theta)$$

2. Compute gradient

3. Optimize

1. Same output $\hat{y}$, same log conditional likelihood



$$\theta_{MLE} = \arg\max_{\theta} \prod_{i=1}^{n} f\left(y^{(i)} \mid \boldsymbol{x}^{(i)}, \theta\right) = \arg\max_{\theta} LL(\theta)$$

$$L(\theta) = \prod_{i=1}^{n} P\left(Y = y^{(i)} \mid \boldsymbol{X} = \boldsymbol{x}^{(i)}, \theta\right)$$

Binary class labels:
$Y \in \{0, 1\}$

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

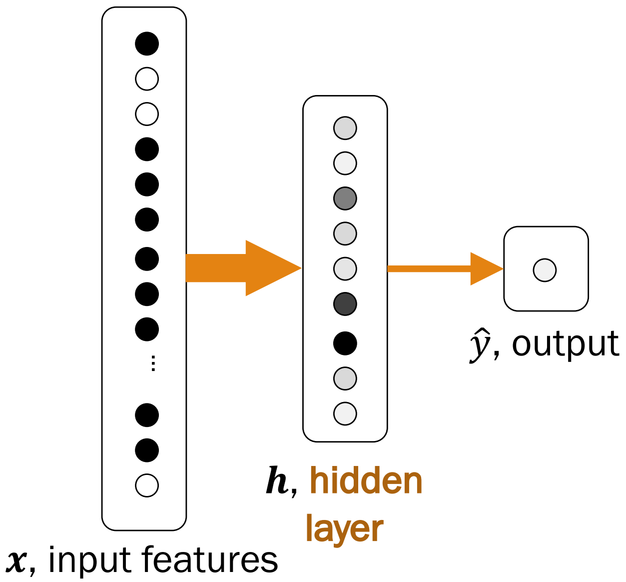
$$LL(\theta) = \sum_{i=1}^{n} y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)$$

for $j = 1, \dots, |\boldsymbol{h}|$:

$$h_j = \sigma\left({\theta_j^{(h)}}^T \boldsymbol{x}\right)$$

$$\hat{y} = \sigma\left({\theta^{(\hat{y})}}^T \boldsymbol{h}\right) = P(Y = 1 \mid \boldsymbol{X} = \boldsymbol{x})$$

(model is a little more complicated)



$$\theta_{MLE} = \arg\max_{\theta} \prod_{i=1}^{n} f\left(y^{(i)} \mid \boldsymbol{x}^{(i)}, \theta\right) = \arg\max_{\theta} \ LL(\theta)$$

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

for $j = 1, \ldots, |\boldsymbol{h}|$:

$$h_j = \sigma\left(\boxed{\theta_j^{(h)}}^T \boldsymbol{x}\right)$$ dimension $|\boldsymbol{x}|$

$$\hat{y} = \sigma\left(\boxed{\theta^{(\hat{y})}}^T \boldsymbol{h}\right) = P(Y = 1 \mid \boldsymbol{X} = \boldsymbol{x})$$ dimension $|\boldsymbol{h}|$

To optimize for log conditional likelihood, we now need to find:

$|\boldsymbol{h}| \cdot |\boldsymbol{x}| + |\cancel{\boldsymbol{x}}\, h|$ parameters

2. Compute gradient

1. Optimization problem:

$$\theta_{MLE} = \arg\max_{\theta} \prod_{i=1}^{n} f\left(y^{(i)} \mid \boldsymbol{x}^{(i)}, \theta\right) = \arg\max_{\theta} LL(\theta)$$

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

$$h_j = \sigma\left(\boxed{\theta_j^{(h)}}^T \boldsymbol{x}\right) \quad \text{for } j = 1, \ldots, |\boldsymbol{h}| \qquad \hat{y} = \sigma\left(\boxed{\theta^{(\hat{y})}}^T \boldsymbol{h}\right)$$

2. Compute gradient

Take gradient with respect to all θ parameters

3. Optimize

Calculus refresher #1:
Derivative(sum) =
sum(derivative)

Calculus refresher #2:
Chain rule 🌟 🌟 🌟

3. Optimize

1. Optimization problem:

$$\theta_{MLE} = \arg\max_{\theta} \prod_{i=1}^{n} f\left(y^{(i)} \mid \boldsymbol{x}^{(i)}, \theta\right) = \arg\max_{\theta} LL(\theta)$$

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

$$h_j = \sigma\left(\boxed{\theta_j^{(h)}}^T \boldsymbol{x}\right) \quad \text{for } j = 1, \dots, |\boldsymbol{h}| \qquad \hat{y} = \sigma\left(\boxed{\theta^{(\hat{y})}}^T \boldsymbol{h}\right)$$

2. Compute gradient

Take gradient with respect to all θ parameters

3. Optimize

```
initialize params
repeat many times:
  compute gradient
   params += η * gradient
```

Training a neural net

1. Optimization problem:

$$\theta_{MLE} = \arg\max_{\theta} \prod_{i=1}^{n} f\left(y^{(i)} \mid \boldsymbol{x}^{(i)}, \theta\right) = \arg\max_{\theta} LL(\theta)$$

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

$$\theta^{(\hat{y})^T} \boldsymbol{h}\Big)$$

2. Compute ... ers

Wait, did we just skip something difficult?

3. Optimize

```
initialize params
repeat many times:
  compute gradient
   params += η * gradient
```

2. Compute gradient via backpropagation

1. Optimization problem:

$$\theta_{MLE} = \arg\max_{\theta} \prod_{i=1}^{n} f\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}, \theta\right) = \arg\max_{\theta} LL(\theta)$$

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

$$h_j = \sigma\left(\boxed{\theta_j^{(h)}}^T \boldsymbol{x}\right) \quad \text{for } j = 1, \ldots, |\boldsymbol{h}| \qquad \hat{y} = \sigma\left(\boxed{\theta^{(\hat{y})}}^T \boldsymbol{h}\right)$$

2. Compute gradient

Take gradient with respect to all θ parameters

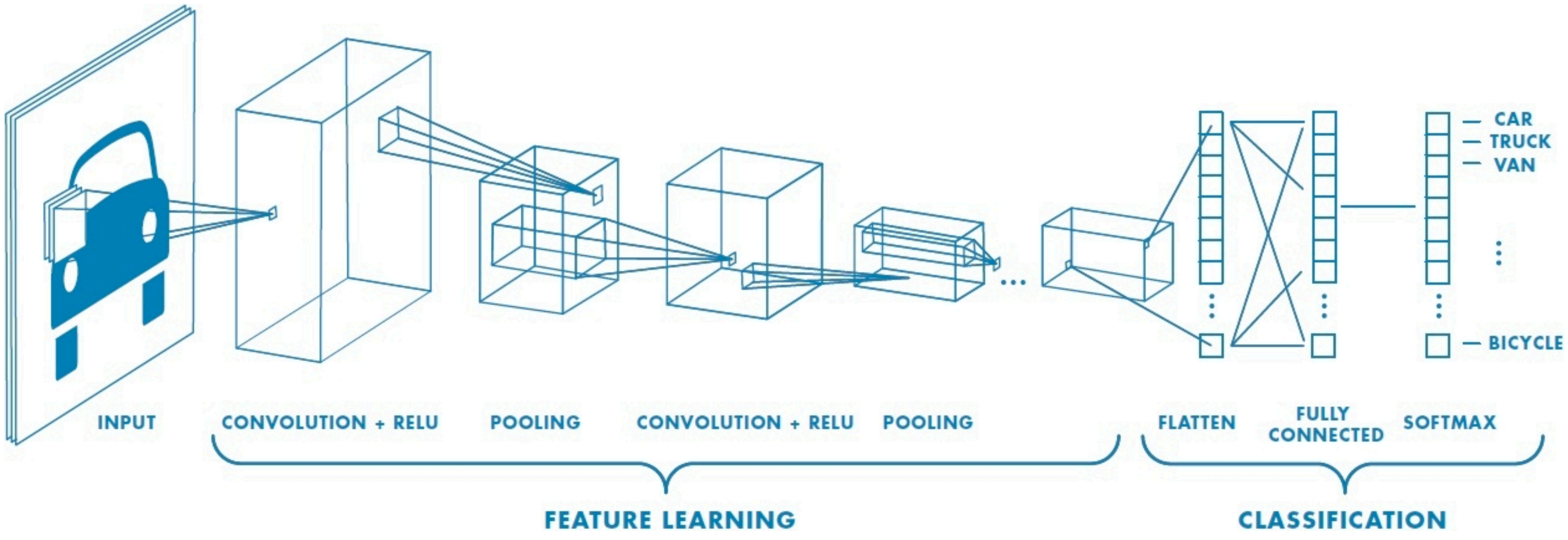
3. Optimize

```
initia
repeat
   comp
    par
```

Learn the tricks behind **backpropagation** in CS229, CS231N, CS224N, etc.

Beyond the basics

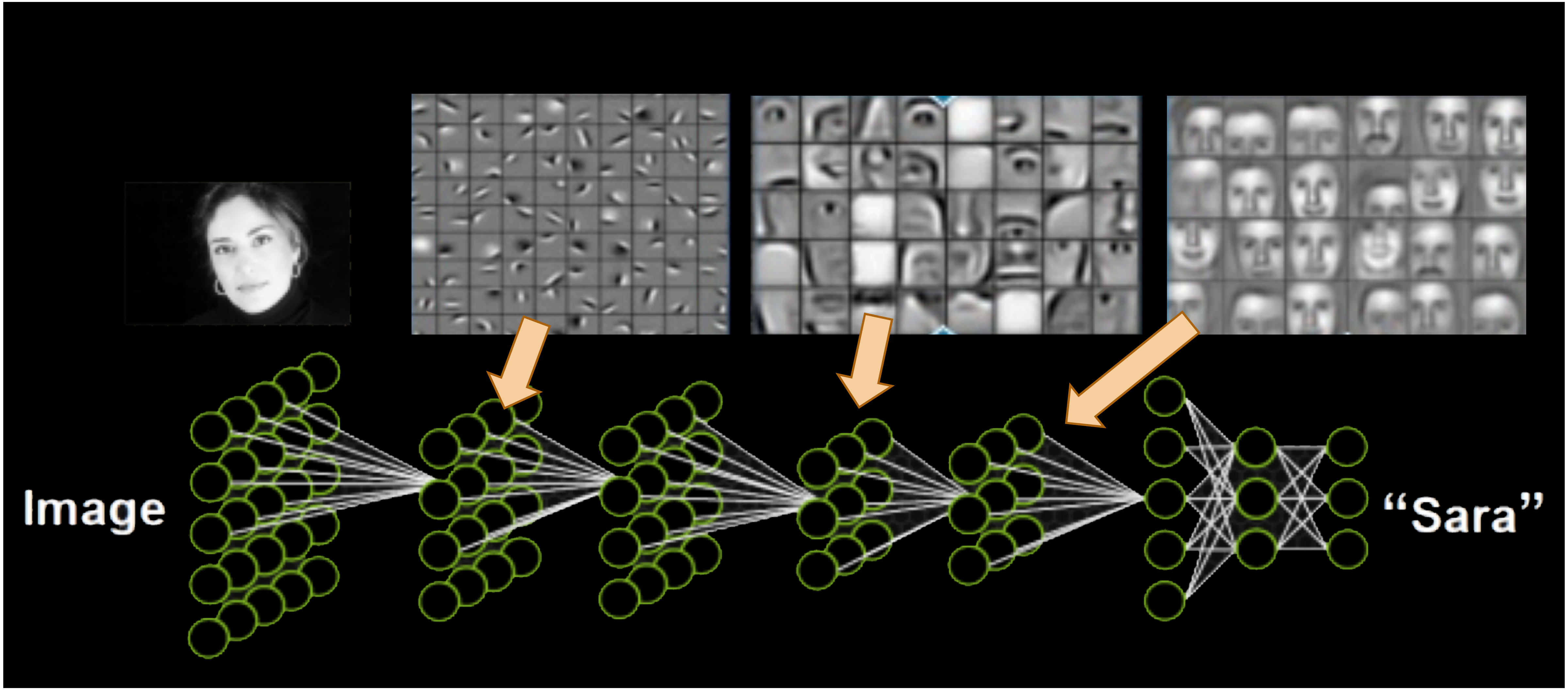
Shared weights?



It turns out if you want to force some of your weights to be shared over different neurons, the math isn't much harder.

Convolution is an example of such weight-sharing and is used a lot for vision (Convolutional Neural Networks, CNN).

Neural networks with multiple layers



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

Neurons learn features of the dataset

Neurons in later layers will respond strongly to high-level features of your training data.

If your training data is faces, you will get lots of face neurons.

If your training data is all of YouTube…

YouTube

…you get a cat neuron.

Top stimuli in test set

Optimal stimulus found by numerical optimization

Le, et al., Building high-level features using large-scale unsupervised learning. ICML 2012


Hire the smartest people in the world
Invent cat detector

Multiple outputs?

Softmax is a generalization of the sigmoid function.

sigmoid(z): value in range [0, 1]

$z \in \mathbb{R}$:

$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma(z)$$

(equivalent: Bernoulli p)

softmax(z): k-dimensional values in range[0,1] that add up to 1

$\boldsymbol{z} \in \mathbb{R}^k$:

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

(equivalent: Multinomial $p_1, \ldots, p_k$)

Softmax test metric: Top-5 error

$Y = y$	$P(Y = y \mid \boldsymbol{X} = \boldsymbol{x})$
5	0.14
8	0.13
7	0.12
2	0.10
9	0.10
4	0.09
1	0.09
0	0.09
6	0.08
3	0.05

Top-5 classification error
What % of datapoints did *not* have the correct class label in the top-5 predictions?

ImageNet classification

22,000 categories

14,000,000 images

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

…
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, S
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

Le, et al., Building high-level features using large-scale unsupervised learning. ICML 2012

ImageNet classification challenge

~~22,000 categories~~

1000 categories

14,000,000 images

1,200,000 images in train set

200,000 images in test set

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

moothhound shark, Mustelus mustelus
n dogfish, Mustelus canis
Florida smoothhound, Mustelus norrisi
odon obseus
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
…

Le, et al., Building high-level features using large-scale unsupervised learning. ICML 2012

ImageNet challenge: Top-5 classification error

(lower is better)

99.5%

Random guess

$$P(\text{true class label not in 5 guesses}) = \frac{\binom{999}{5}}{\binom{1000}{5}} = \frac{995}{1000}$$

ImageNet challenge: Top-5 classification error

(lower is better)

99.5%
Random guess

25.8%
Pre-Neural Networks

5.1%
Humans
(2014)

16.4%
GoogLe Net
(2015)

Russakovsky et al., ImageNet Large Scale Visual Recognition Challenge. IJCV 2015
Szegedy et al., Going Deeper With Convolutions. CVPR 2015
Hu et al., Squeeze-and-Excitation Networks. Preprint arXiV 2017

ImageNet challenge: Top-5 classification error

(lower is better)

99.5%	25.8%	5.1%
Random guess	Pre-Neural Networks	Humans (2014)

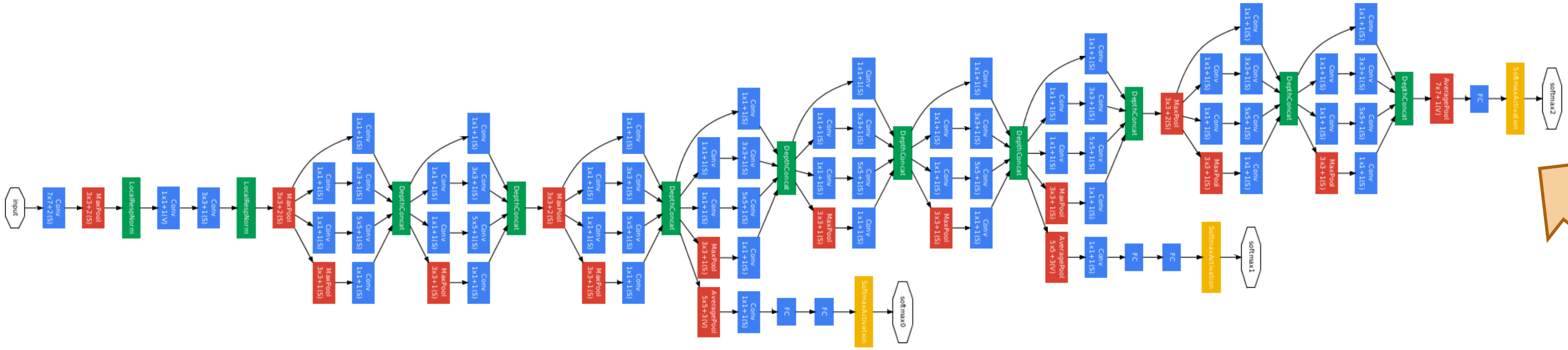
16.4%	2.25%
GoogLe Net (2015)	SENet (2017)

Russakovsky et al., ImageNet Large Scale Visual Recognition Challenge. IJCV 2015
Szegedy et al., Going Deeper With Convolutions. CVPR 2015
Hu et al., Squeeze-and-Excitation Networks. Preprint arXiV 2017

GoogLeNet (2015)

1 Trillion Artificial Neurons

(btw human brains have 1 billion neurons)



Multiple,
Multi class output

22 layers deep!

Szegedy et al., Going Deeper With Convolutions. CVPR 2015

Speeding up gradient descent

minimizes loss (a function of prediction error)

```
initialize θ_j = 0 for 0 ≤ j ≤ m
repeat many times:
    gradient[j] = 0 for 0 ≤ j ≤ m
    for each training example (x,y):
        for each 0 ≤ j ≤ m:
            compute gradient
    θ_j -= η * gradient[j] for all 0 ≤ j ≤ m
```

1. What if we have 1,200,000 images in our training set?
2. How can we speed up the update?

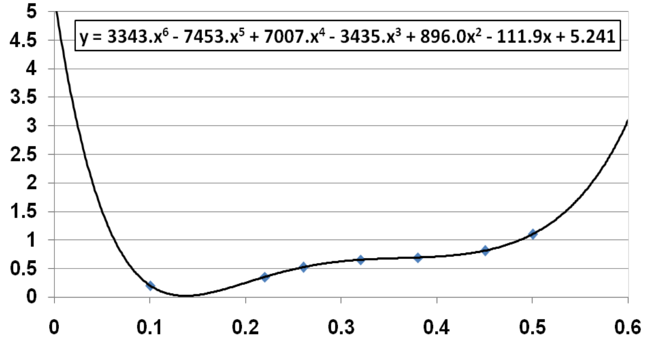
Our **batch gradient descent** (over the entire training set) will be slow + expensive.

1. Use **stochastic gradient descent** (randomly select training examples with replacement).
2. **Momentum update** (Incorporate "acceleration" or "deceleration" of gradient updates so far)

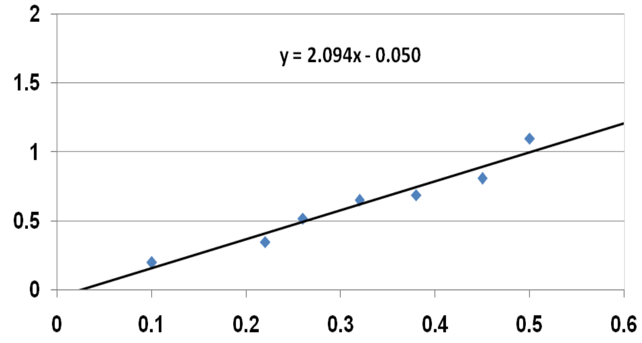
Good ML = Generalization

Overfitting

Fitting the training data too well, such that we lose generality of model for predicting new data



perfect fit, but bad predictor for new data

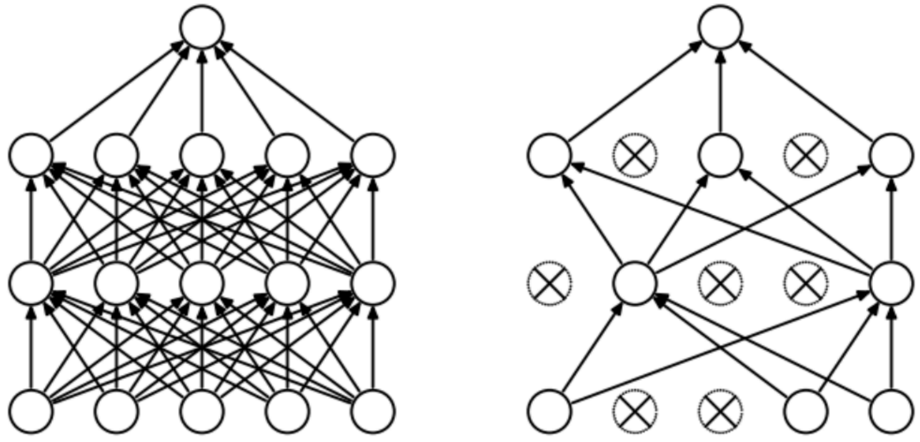


more general fit + better predictor for new data

Dropout

During training, randomly leave out some neurons each training step.

It will make your network more robust.



(a) Standard Neural Net

(b) After applying dropout.

Making decisions?

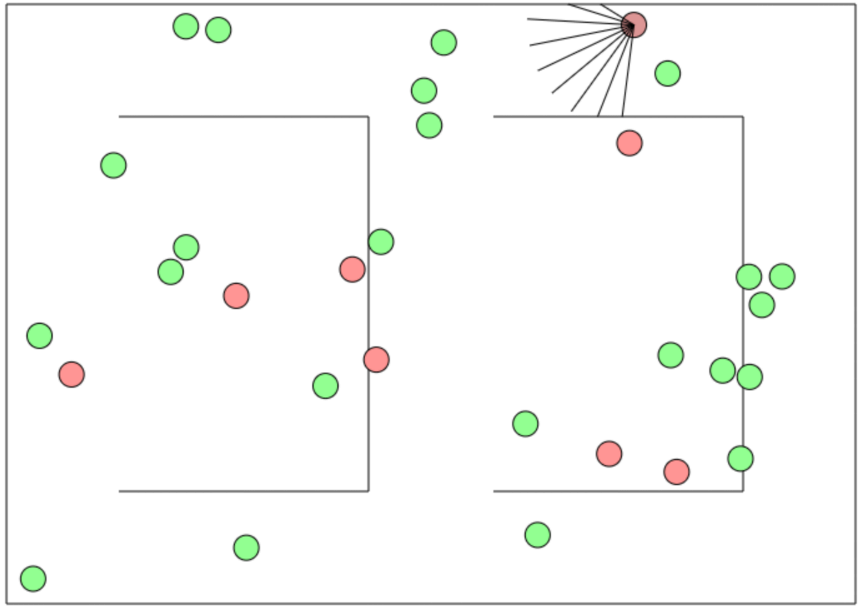
Not everything is classification.

Deep Reinforcement Learning

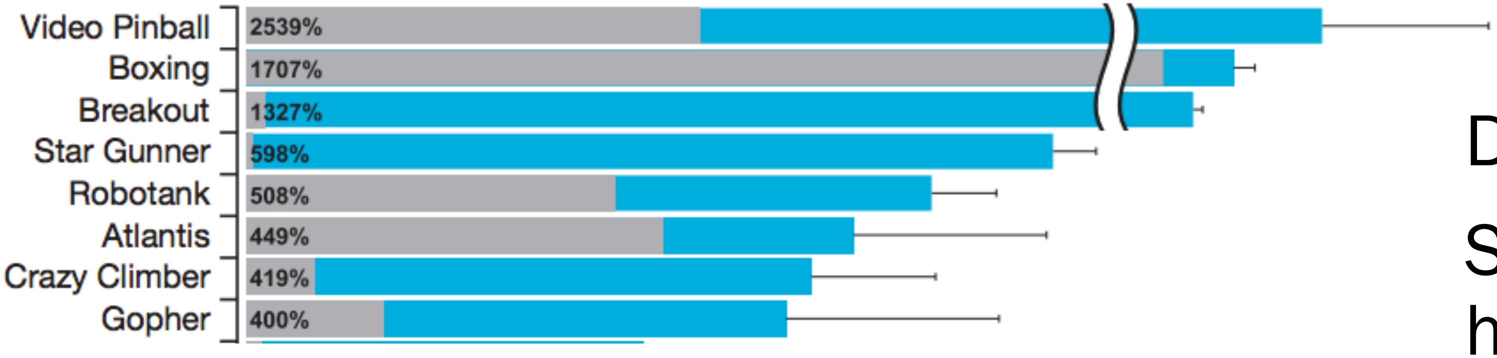
Instead of having the output of a model be a probability, you make output an expectation.

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

Deep Reinforcement Learning



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



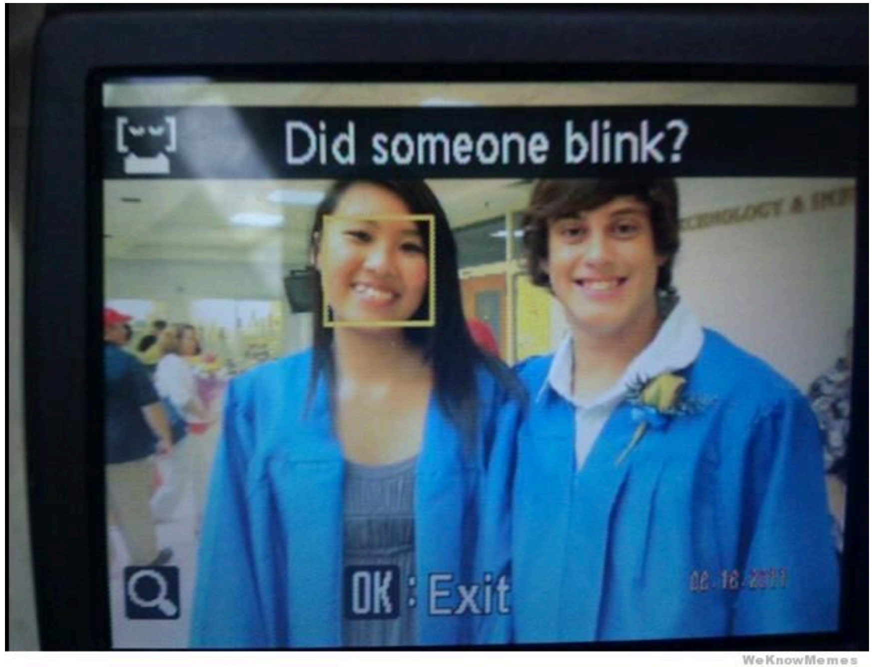
Deep Mind Atari Games

Score compared to best human

What's missing?

How are you getting your data?

Ethics and datasets



Sometimes machine learning feels universally unbiased.

We can even prove our estimators are “unbiased” (mathematically).

Google/Nikon/HP had biased datasets.

Should your data be unbiased?

Dataset: Google News

$$\overrightarrow{\text{man}} - \overrightarrow{\text{woman}} \approx \overrightarrow{\text{king}} - \overrightarrow{\text{queen}}$$

$$\overrightarrow{\text{man}} - \overrightarrow{\text{woman}} \approx \overrightarrow{\text{computer programmer}} - \overrightarrow{\text{homemaker}}.$$

Should our unbiased data collection reflect society's systemic bias?

Bolukbasi et al., Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings. NIPS 2016

How can we explain decisions?

If your task is image classification, reasoning about high-level features is relatively easy.

Everything can be visualized.

What if you are trying to classify social outcomes?

- Criminal recidivism
- Job performance
- Policing
- Terrorist risk
- At-risk kids

Ethics in Machine Learning is a whole new field. 🙂