26: Logistic Regression + Deep Learning

Lisa Yan

November 20, 2019

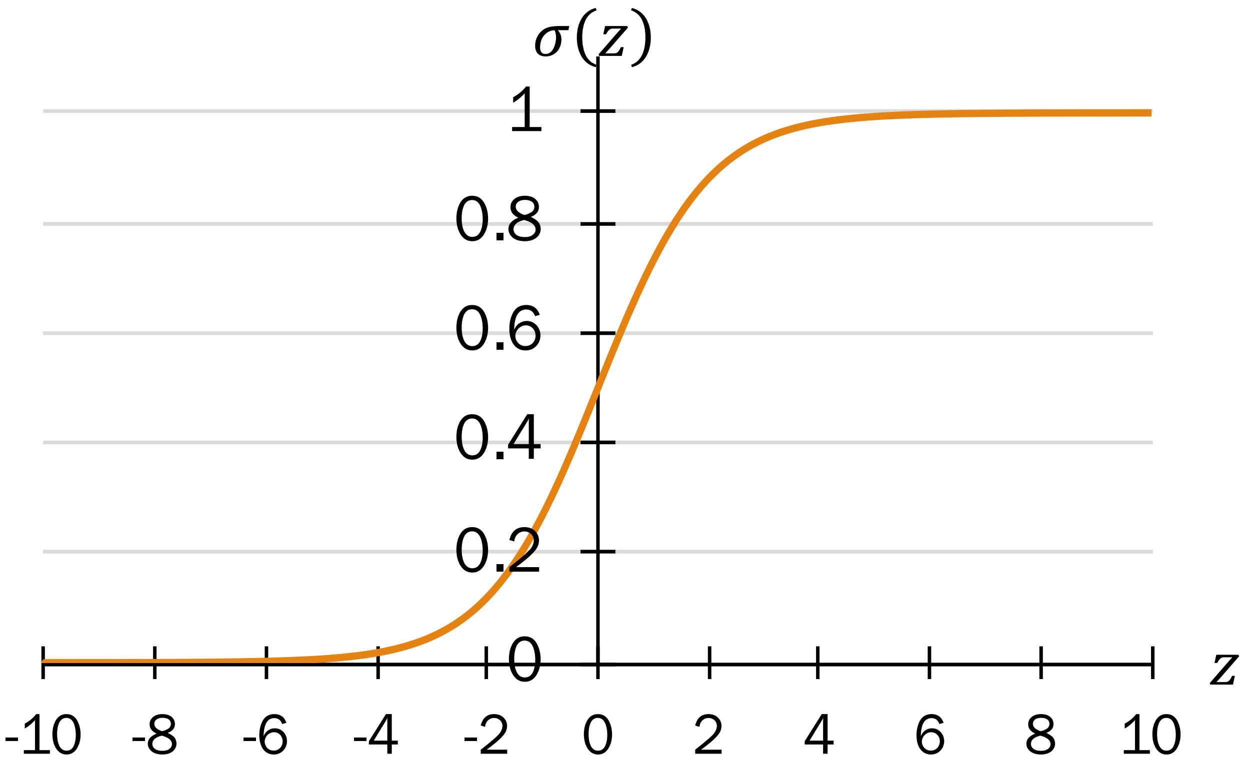
Background: Sigmoid function $\sigma(z)$

Review

- The sigmoid function:

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

- Sigmoid squashes z to a number between 0 and 1.



- Recall definition of probability: A number between 0 and 1

👉 $\sigma(z)$ can represent a probability.

Logistic Regression Model

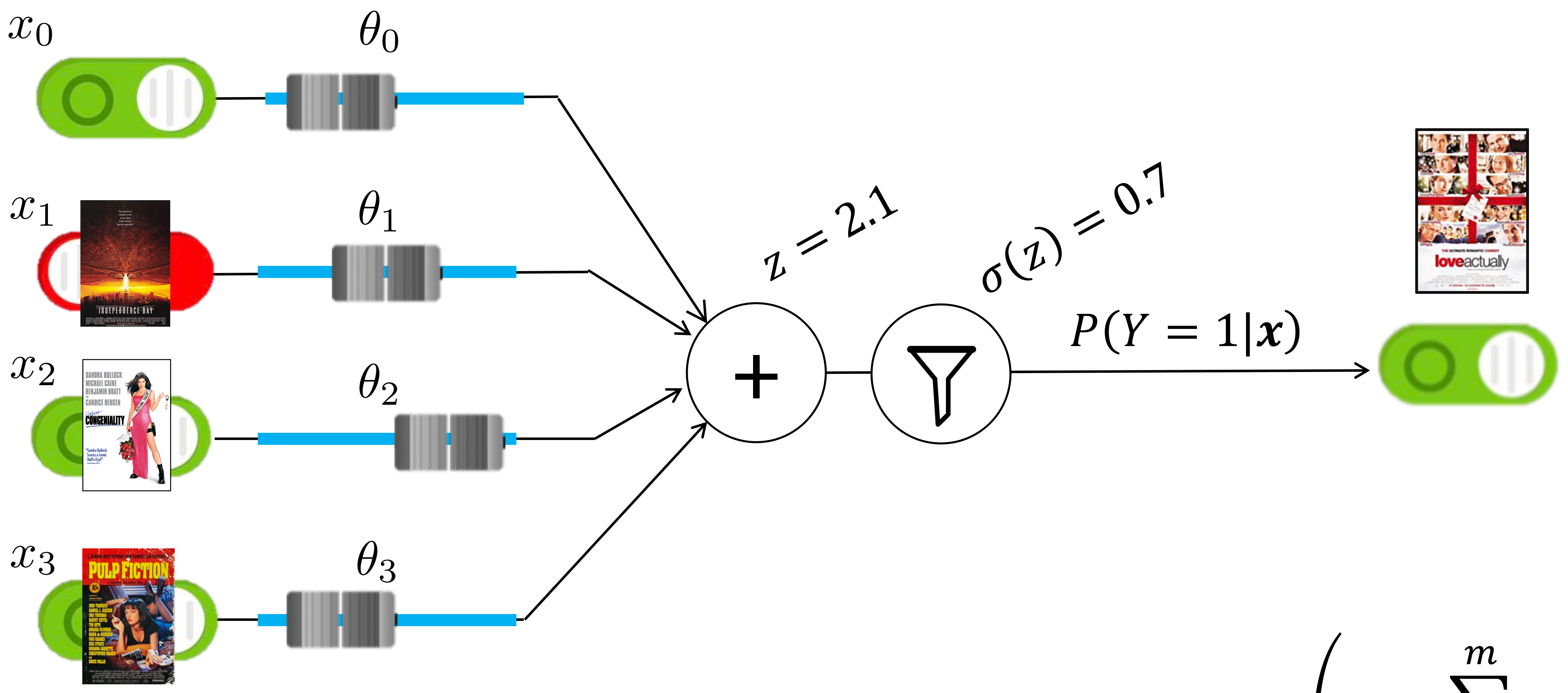
$$\hat{Y} = \arg\max_{y=\{0,1\}} P(Y \mid \boldsymbol{X})$$

Predict the Y that is most likely given our observation $\boldsymbol{X}$

where $$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

models $P(Y \mid \boldsymbol{X})$ directly

Logistic Regression Model



X, input features
[0,1,1]

$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

Logistic Regression Model

Review

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

where

$$P(Y = 1 | \boldsymbol{X} = \boldsymbol{x}) = \sigma\left(\theta_0 + \sum_{j=1}^{m} \theta_j x_j\right)$$

Predict the Y that is most likely given our observation $\boldsymbol{X}$

models $P(Y \mid \boldsymbol{X})$ directly

- $\sigma(z) = \frac{1}{1+e^{-z}}$, the sigmoid function
- For simplicity, define $x_0 = 1$: $P(Y = 1 | \boldsymbol{X} = \boldsymbol{x}) = \sigma(\theta^T \boldsymbol{x})$
- Since $P(Y = 1 | \boldsymbol{X} = \boldsymbol{x}) + P(Y = 0 | \boldsymbol{X} = \boldsymbol{x}) = 1$:

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

Today's plan

Logistic Regression

- Chapter 0: Background
- Chapter 1: Big Picture
- Chapter 2: Details
- Chapter 3: Philosophy

Intro to Deep Learning

- Parameters of a neural network
- Training neural networks

Training: Learning the parameters

Logistic regression gets its **intelligence** from its parameters $\theta = (\theta_0, \theta_1, \ldots, \theta_m)$.

- Logistic Regression Model: $$P(Y = 1 | \boldsymbol{X} = \boldsymbol{x}) = \sigma(\theta^T \boldsymbol{x})$$

- Want to predict training data as correctly as possible: $$\arg\max_{y=\{0,1\}} P\left(Y | \boldsymbol{X} = \boldsymbol{x}^{(i)}\right) = y^{(i)}$$ as often as possible

- Therefore, choose θ that maximizes the conditional likelihood of observing i.i.d. training data: $$L(\theta) = \prod_{i=1}^{n} P\left(Y = y^{(i)} | \boldsymbol{X} = \boldsymbol{x}^{(i)}, \theta\right)$$

During training, find the θ that maximizes log-conditional likelihood of the training data. Use MLE!

Training: Learning the parameters via MLE

0. Add $x_0^{(i)} = 1$ to each $\boldsymbol{x}^{(i)}$
1. Logistic Regression model:

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

2. Compute log-conditional likelihood of training data:

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

3. Compute derivative of log-likelihood with respect to each $\theta_j, j = 0, 1, \dots, m$:

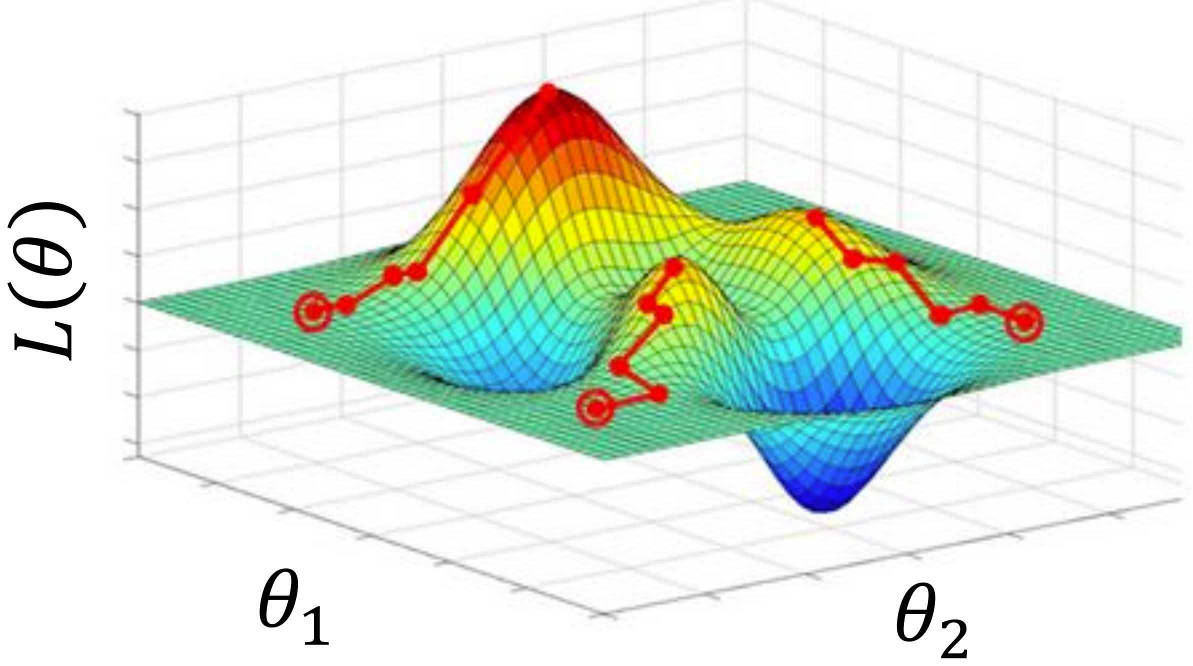
$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right)\right] x_j^{(i)}$$

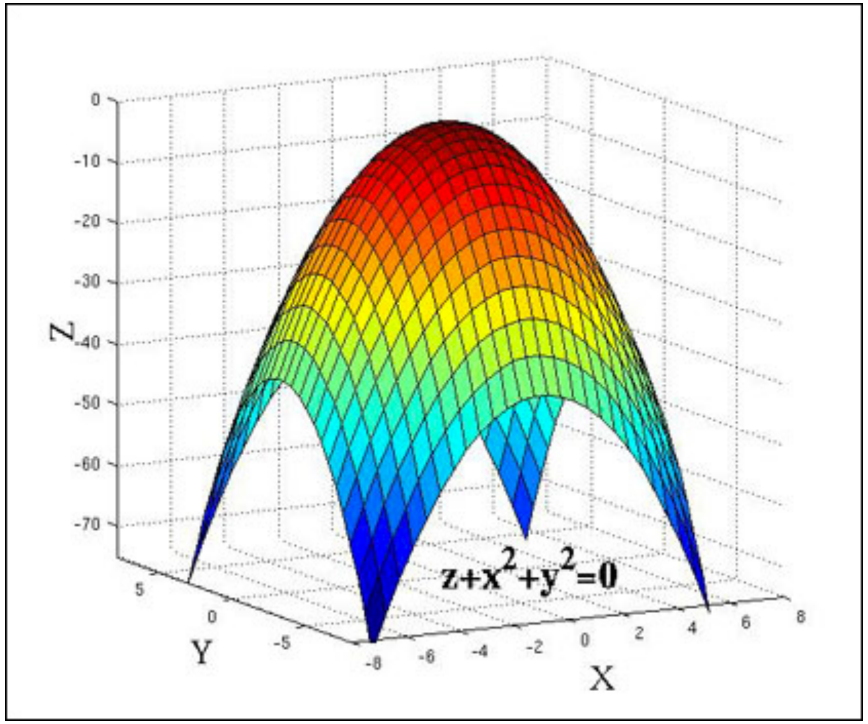
4. Optimize

How did we get this math?? More in Chapter 2…

Gradient Ascent

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





Logistic regression $LL(\theta)$
is convex

Training: Gradient ascent step

4. Optimize.

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

Repeat many times:

For all thetas:

$$\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \frac{\partial LL\left(\theta^{\text{old}}\right)}{\partial \theta_j^{\text{old}}}$$

$$= \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left({\theta^{\text{old}}}^T \boldsymbol{x}^{(i)} \right) \right] x_j^{(i)}$$

What does this look like in code?

Training: Gradient Ascent

Gradient Ascent Step $\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left({\theta^{\text{old}}}^T \boldsymbol{x}^{(i)} \right) \right] x_j^{(i)}$

```
initialize θ_j = 0 for 0 ≤ j ≤ m
repeat many times:
    gradient[j] = 0 for 0 ≤ j ≤ m

    // compute all gradient[j]'s
    // based on n training examples

    θ_j += η * gradient[j] for all 0 ≤ j ≤ m
```

Training: Gradient Ascent

Gradient Ascent Step $\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left({\theta^{\text{old}}}^T \boldsymbol{x}^{(i)} \right) \right] x_j^{(i)}$

```
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:
            // update gradient[j] for
            // current (x,y) example
    θ_j += η * gradient[j] for all 0 ≤ j ≤ m
```

Training: Gradient Ascent

Gradient Ascent Step $\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left({\theta^{\text{old}}}^T \boldsymbol{x}^{(i)} \right) \right] x_j^{(i)}$

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:
            gradient[j] += $\left[y - \frac{1}{1 + e^{-\theta^T \boldsymbol{x}}} \right] x_j$
    θ_j += η * gradient[j] for all 0 ≤ j ≤ m

What are important implementation details? 🤔

Training: Gradient Ascent

Gradient Ascent Step $\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left({\theta^{\text{old}}}^T \boldsymbol{x}^{(i)} \right) \right] x_j^{(i)}$

```
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:
            gradient[j] += [y − 1/(1 + e^(−θ^T x))] x_j
    θ_j += η * gradient[j] for all 0 ≤ j ≤ m
```

$$\text{gradient}[j] \mathrel{+}= \left[y - \frac{1}{1 + e^{-\theta^T \boldsymbol{x}}} \right] x_j$$

- x_j is j-th feature of input var $x = (x_1, \dots, x_m)$

Training: Gradient Ascent

Gradient Ascent Step $\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left({\theta^{\text{old}}}^T \boldsymbol{x}^{(i)} \right) \right] x_j^{(i)}$

```
initialize θj = 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:
         gradient[j] +=  [y − 1/(1 + e^(−θ^T x))] xj
   θj += η * gradient[j] for all 0 ≤ j ≤ m
```

$$\text{gradient[j]} \mathrel{+}= \left[y - \frac{1}{1 + e^{-\theta^T \boldsymbol{x}}} \right] x_j$$

- x_j is j-th feature of input var $x = (x_1, \ldots, x_m)$
- Insert $x_0 = 1$ before training

Training: Gradient Ascent

Gradient Ascent Step $\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left(\theta^{\text{old}^T} \boldsymbol{x}^{(i)} \right) \right] x_j^{(i)}$

```
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:
            gradient[j] += [y - 1/(1 + e^(-θ^T x))] x_j
    θ_j += η * gradient[j] for all 0 ≤ j ≤ m
```

- x_j is j-th feature of input var $x = (x_1, \ldots, x_m)$
- Insert $x_0 = 1$ before training
- Finish computing gradient before updating any part of θ

Training: Gradient Ascent

Gradient Ascent Step $\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left({\theta^{\text{old}}}^T \boldsymbol{x}^{(i)} \right) \right] x_j^{(i)}$

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:

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

    θ_j += η * gradient[j] for all 0 ≤ j ≤ m

- x_j is j-th feature of input var $x = (x_1, \dots, x_m)$
- Insert $x_0 = 1$ before training
- Finish computing gradient before updating any part of θ
- Learning rate η is a constant you set before training

Training: Gradient Ascent

Gradient Ascent Step $\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left(\theta^{\text{old}^T} \boldsymbol{x}^{(i)} \right) \right] x_j^{(i)}$

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:
            gradient[j] += $\left[y - \frac{1}{1 + e^{-\theta^T \boldsymbol{x}}} \right] x_j$
    θ_j += η * gradient[j] for all 0 ≤ j ≤ m

- x_j is j-th feature of input var $x = (x_1, \ldots, x_m)$
- Insert $x_0 = 1$ before training
- Finish computing gradient before updating any part of θ
- Learning rate η is a constant you set before training

Testing: Classification with Logistic Regression

Training

Learn parameters $\theta = (\theta_0, \theta_1, \dots, \theta_m)$

via gradient ascent: $$\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left(\theta^{\text{old}^T} \boldsymbol{x}^{(\boldsymbol{i})} \right) \right] x_j^{(i)}$$

Testing

- Compute $\hat{y} = P(Y = 1 | \boldsymbol{X} = \boldsymbol{x}) = \sigma(\theta^T \boldsymbol{x}) = \dfrac{1}{1 + e^{-\theta^T \boldsymbol{x}}}$
- Classify instance as:

$$\begin{cases} 1 & \hat{y} > 0.5, \text{ equivalently } \theta^T \boldsymbol{x} > 0 \\ 0 & \text{otherwise} \end{cases}$$

👉 ⚠️ Parameters θ_j are **not** updated during testing phase

Today's plan

Logistic Regression

- Chapter 0: Background
- Chapter 1: Big Picture
- Chapter 2: Details
- Chapter 3: Philosophy

Intro to Deep Learning

- Parameters of a neural network
- Training neural networks

Introducing notation $\hat{y}$

Logistic Regression model:

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

$$P(Y = y|\boldsymbol{X} = \boldsymbol{x}) = \begin{cases} \hat{y} & \text{if } y = 1 \\ 1 - \hat{y} & \text{if } y = 0 \end{cases}$$

Prediction:

$$\hat{Y} = \arg\max_{y=\{0,1\}} P(Y|\boldsymbol{X} = \boldsymbol{x}) = \begin{cases} 1 & \text{if } \hat{y} > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Training: Learning the parameters via MLE

0. Add $x_0^{(i)} = 1$ to each $\boldsymbol{x}^{(i)}$

1. Logistic Regression model:

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

2. Compute log-likelihood of training data:

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

3. Compute derivative of log-likelihood with respect to each $\theta_j, j = 0, 1, \ldots, m$:

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

Training: Learning the parameters via MLE

0. Add $x_0^{(i)} = 1$ to each $\boldsymbol{x}^{(i)}$
1. Logistic Regression model:

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

2. Compute log-likelihood of training data:

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

🤔 How did we get this likelihood function?

3. Compute derivative of log-likelihood with respect to each $\theta_j, j = 0, 1, \ldots, m$:

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

Log-likelihood of data

Logistic Regression model:

$$P(Y = y|\boldsymbol{X} = \boldsymbol{x}) = \begin{cases} \hat{y} & \text{if } y = 1 \\ 1 - \hat{y} & \text{if } y = 0 \end{cases}$$

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

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

(see Bernoulli MLE PMF)

Likelihood of training data:

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

Notes:

- Actually **conditional likelihood**
- Still correctly gets correct θ_{MLE} since $\boldsymbol{X}, \theta$ independent
- See lecture notes

Log-likelihood of data

Logistic Regression model:

$$P(Y = y|\boldsymbol{X} = \boldsymbol{x}) = \begin{cases} \hat{y} & \text{if } y = 1 \\ 1 - \hat{y} & \text{if } y = 0 \end{cases}$$

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

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

(see Bernoulli MLE PMF)

Likelihood of training data:

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

Log-likelihood:

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

$$= \sum_{i=1}^{n} y^{(i)} \log \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right)\right)$$ ✅

Training: Learning the parameters via MLE

0. Add $x_0^{(i)} = 1$ to each $\boldsymbol{x}^{(i)}$
1. Logistic Regression model: $P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma(\theta^T \boldsymbol{x})$
2. Compute log-likelihood of training data:

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

3. Compute derivative of log-likelihood with respect to each $\theta_j, j = 0, 1, \ldots, m$:

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

How did we get this gradient?

Aside: Sigmoid has a beautiful derivative

Sigmoid function:

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

Derivative:

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

What is $\frac{\partial}{\partial \theta_j}\sigma(\theta^T \boldsymbol{x})$?

A. $\sigma(x_j)[1 - \sigma(x_j)]x_j$
B. $\sigma(\theta^T \boldsymbol{x})[1 - \sigma(\theta^T \boldsymbol{x})]\boldsymbol{x}$
C. $\sigma(\theta^T \boldsymbol{x})[1 - \sigma(\theta^T \boldsymbol{x})]x_j$
D. $\sigma(\theta^T \boldsymbol{x})x_j[1 - \sigma(\theta^T \boldsymbol{x})x_j]$
E. None/other

Aside: Sigmoid has a beautiful derivative

Sigmoid function:

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

Derivative:

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

What is $\frac{\partial}{\partial \theta_j}\sigma(\theta^T \boldsymbol{x})$?

A. $\sigma(x_j)[1 - \sigma(x_j)]x_j$

B. $\sigma(\theta^T \boldsymbol{x})[1 - \sigma(\theta^T \boldsymbol{x})]\boldsymbol{x}$

C. $\sigma(\theta^T \boldsymbol{x})[1 - \sigma(\theta^T \boldsymbol{x})]x_j$

D. $\sigma(\theta^T \boldsymbol{x})x_j[1 - \sigma(\theta^T \boldsymbol{x})x_j]$

E. None/other

Let $z = \theta^T \boldsymbol{x} = \sum_{k=0}^{m} \theta_k x_k$.

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

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

Compute gradient of log-conditional likelihood

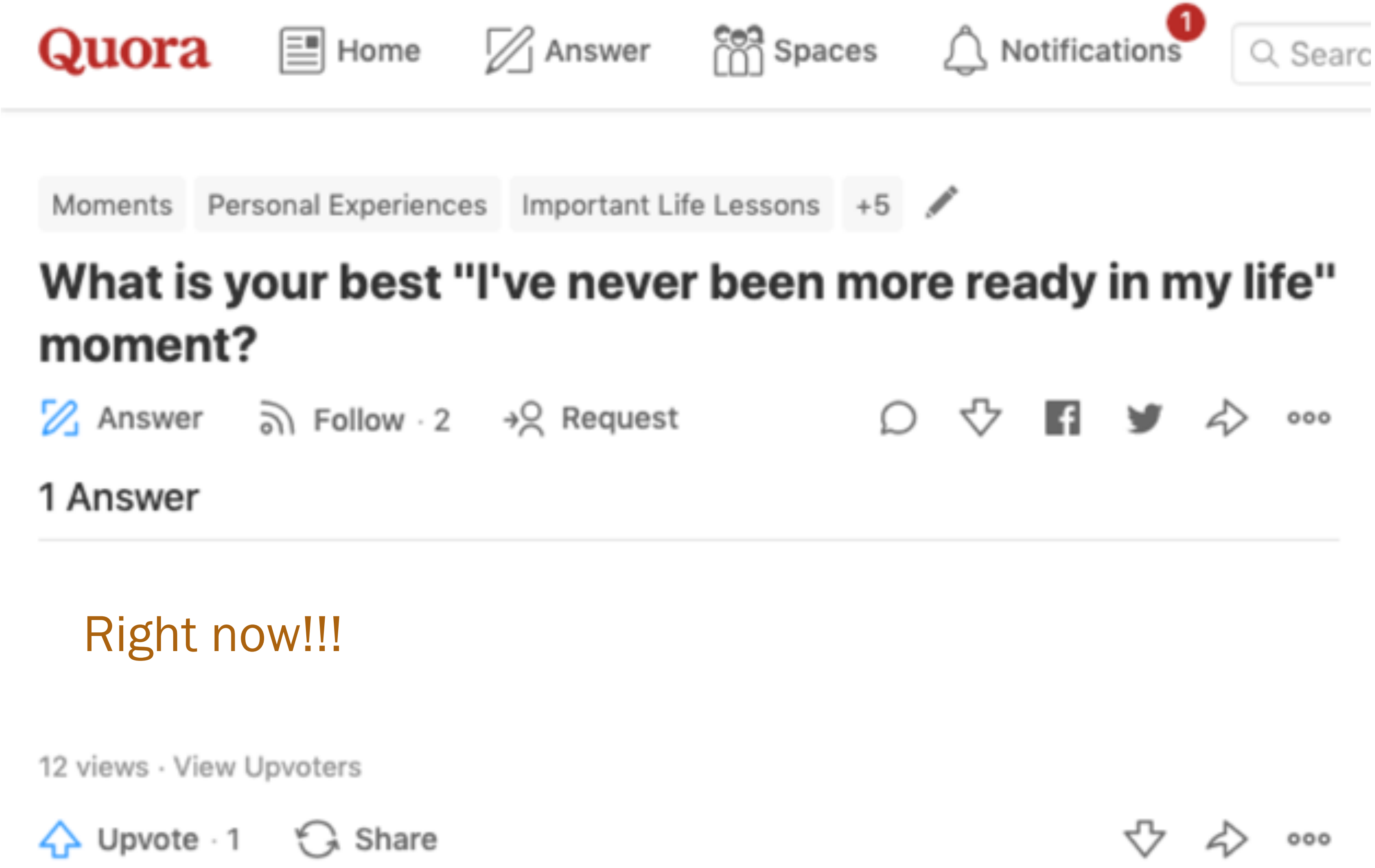
Find: $$\frac{\partial LL(\theta)}{\partial \theta_j}$$

where

Log-conditional Likelihood:

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

Are you ready?



Right now!!!

Compute gradient of log-likelihood

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

Let $\hat{y}^{(i)} = \sigma\left(\theta^T \boldsymbol{x}^{(\boldsymbol{i})}\right)$

$$= \sum_{i=1}^{n} \frac{\partial}{\partial \hat{y}^{(i)}} \left[y^{(i)} \log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \right] \cdot \frac{\partial \hat{y}^{(i)}}{\partial \theta_j}$$

(Chain Rule)

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

(calculus)

$$= \sum_{i=1}^{n} \left[y^{(i)} - \hat{y}^{(i)} \right] x_j^{(i)} \qquad = \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right) \right] x_j^{(i)}$$

(simplify)

Compute gradient of log-likelihood

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

Let $\hat{y}^{(i)} = \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right)$

$$= \sum_{i=1}^{n} \frac{\partial}{\partial \hat{y}^{(i)}} \left[y^{(i)} \log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \right] \cdot \frac{\partial \hat{y}^{(i)}}{\partial \theta_j}$$

(Chain Rule)

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

(calculus)

$$= \sum_{i=1}^{n} \left[y^{(i)} - \hat{y}^{(i)} \right] x_j^{(i)} \qquad = \sum_{i=1}^{n} \left[y^{(i)} - \sigma\left(\theta^T \boldsymbol{x}^{(i)}\right) \right] x_j^{(i)}$$

(simplify)

Today's plan

Naïve Bayes

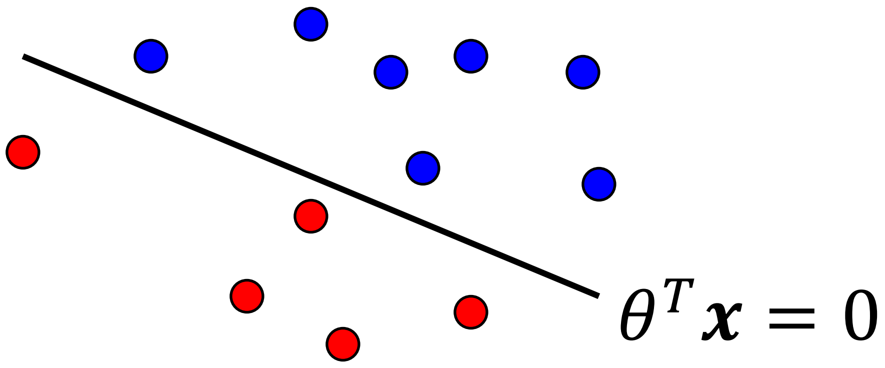
Logistic Regression

- Chapter 0: Background
- Chapter 1: Big Picture
- Chapter 2: Details
- Chapter 3: Philosophy

Intuition about Logistic Regression

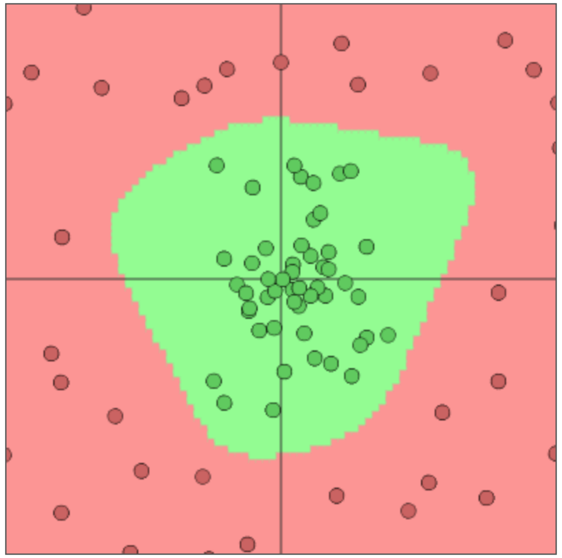
Logistic Regression Model $P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma(\theta^T \boldsymbol{x})$ where $\theta^T \boldsymbol{x} = \sum_{j=0}^{m} \theta_j x_j$

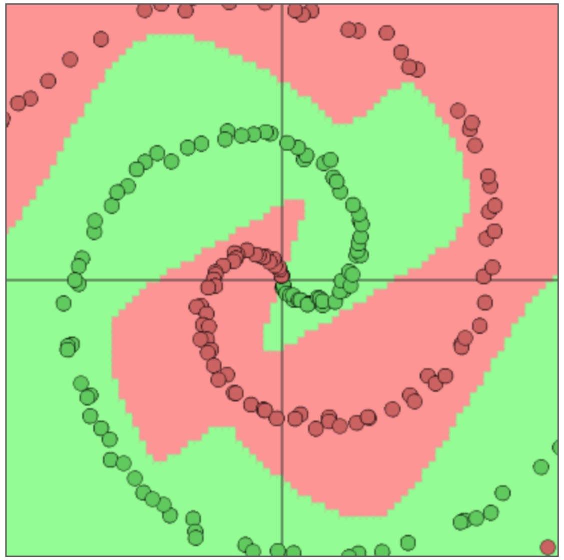
Logistic Regression is trying to fit a <u>line</u> that separates data instances where $y = 1$ from those where $y = 0$:



- We call such data (or functions generating the data <u>linearly separable</u>.
- Naïve Bayes is linear too, because there is no interaction between different features.

Data is often not linearly 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

Many tradeoffs in choosing an algorithm

	Naïve Bayes	Logistic Regression
Modeling goal	$P(\boldsymbol{X}, Y)$	$P(Y\|\boldsymbol{X})$
Generative or **discriminative**?	**Generative:** could use joint distribution to generate new points (⚠️but you might not need this extra effort)	**Discriminative:** just tries to discriminate $y = 0$ vs $y = 1$ (❌ cannot generate new points b/c no $P(\boldsymbol{X}, Y)$)
Continuous input features	⚠️ Needs parametric form (e.g., Gaussian) or discretized buckets (for multinomial features)	✅ Yes, easily
Discrete input features	Yes, multi-value discrete data = multinomial $P(X_i\|Y)$	⚠️ Multi-valued discrete data hard (e.g., if $X_i \in \{A, B, C\}$, not necessarily good to encode as $\{1, 2, 3\}$

30-second pedagogical pause

Summarize what we have learned

Break for jokes/ announcements

Announcements

Problem Set 6

Due: Wednesday 12/4 (after break)

Note: Skip Problem 3 (neural net) for now, we will finish covering it on Friday

Office Hours

During Thanksgiving break: None

Last weekly concept check

Due: Tuesday 12/3 (after break)

Today's plan

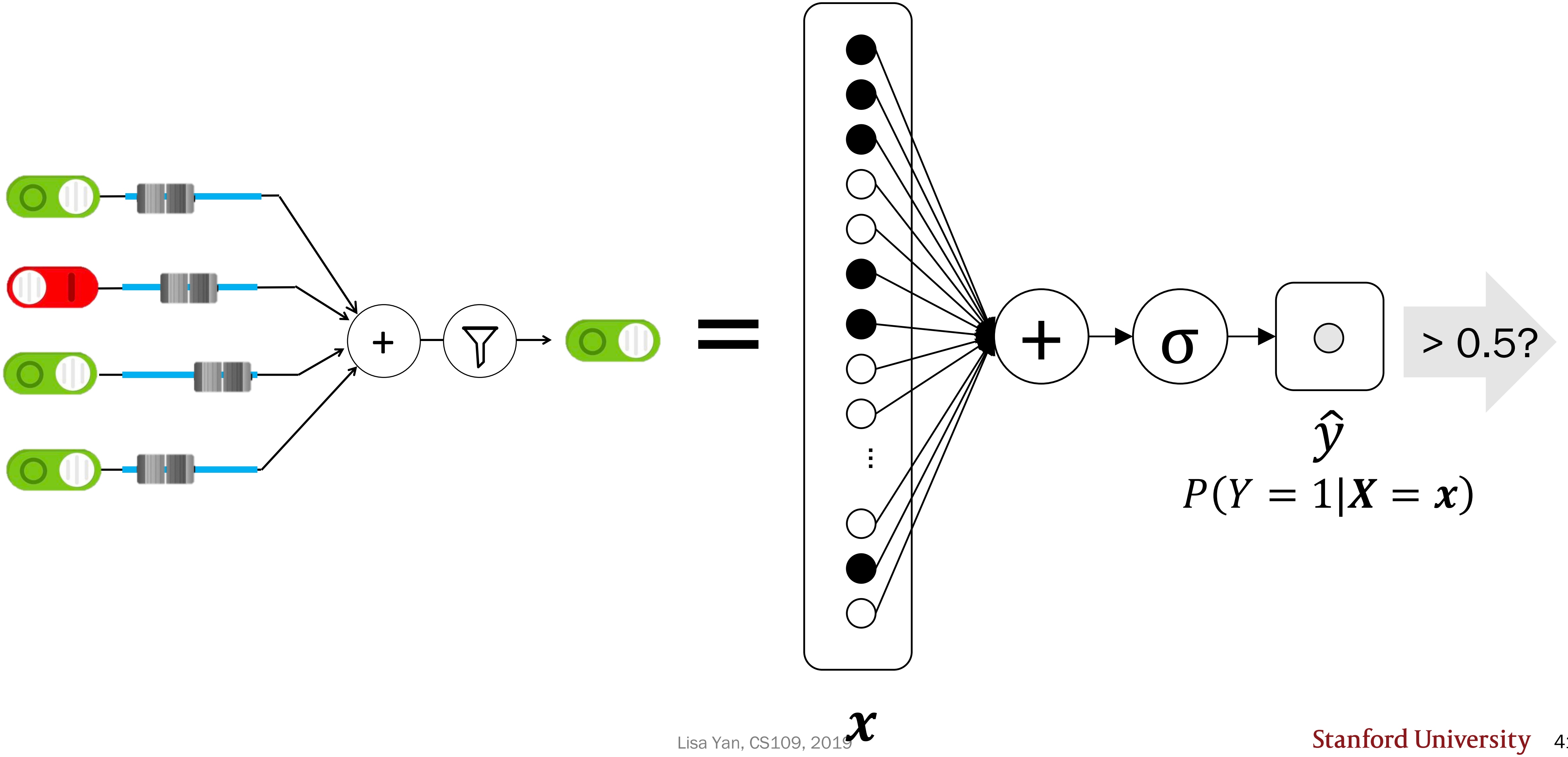
Logistic Regression

- Chapter 0: Background
- Chapter 1: Big Picture
- Chapter 2: Details
- Chapter 3: Philosophy

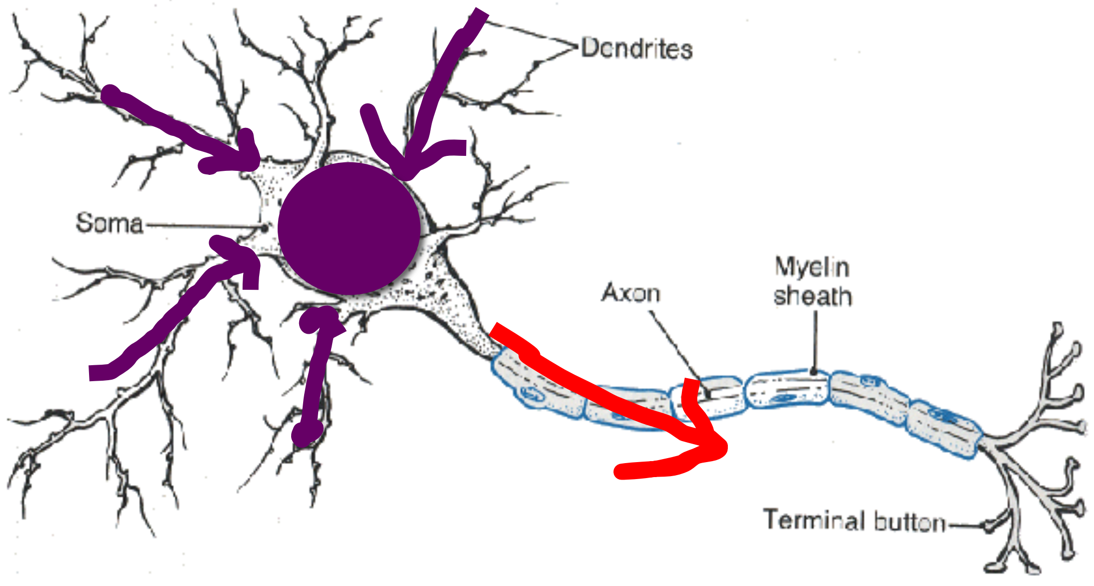
Intro to Deep Learning

- Parameters of a neural network
- Training neural networks

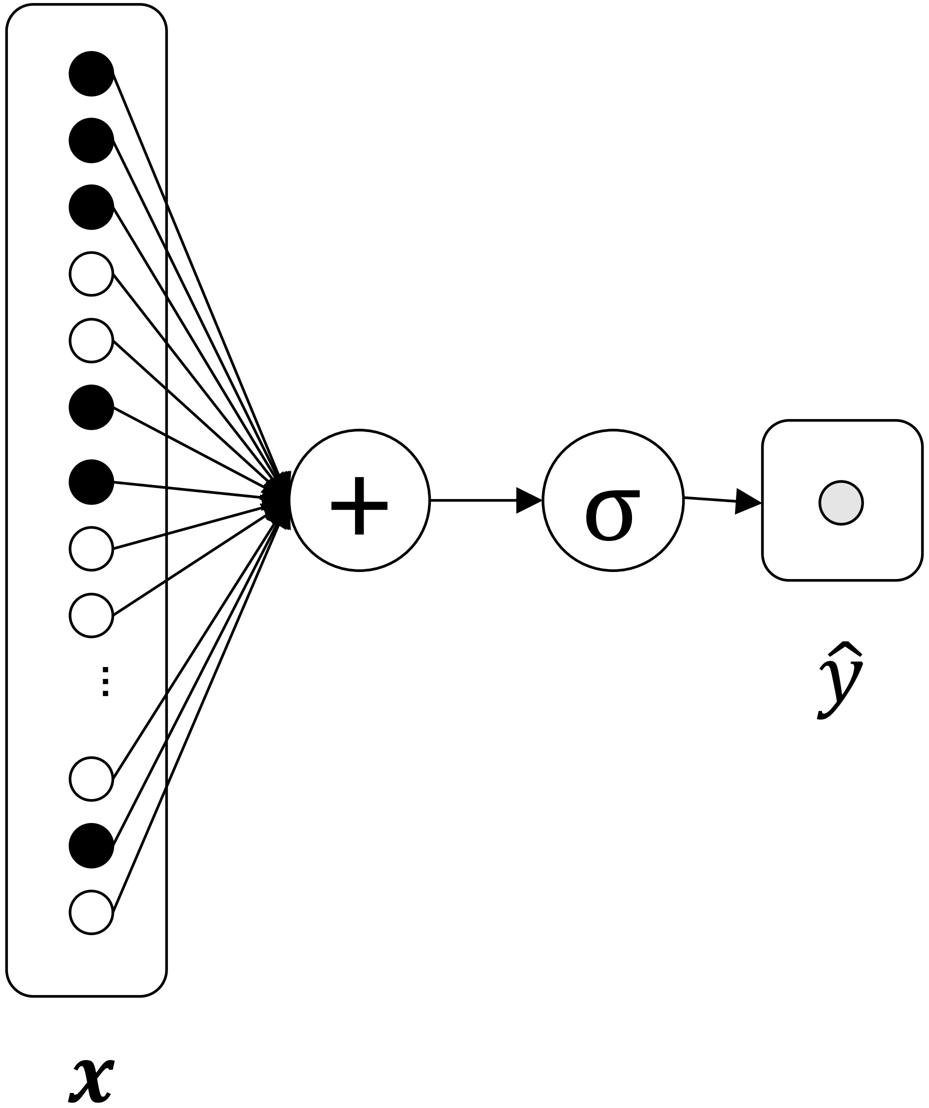
One logistic regression



One neuron = One logistic regression

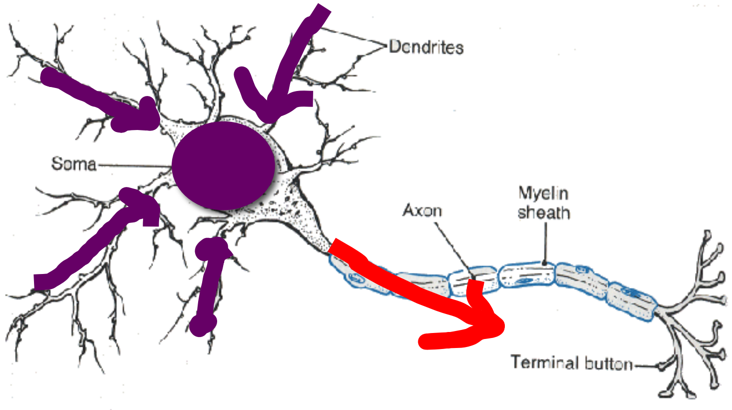


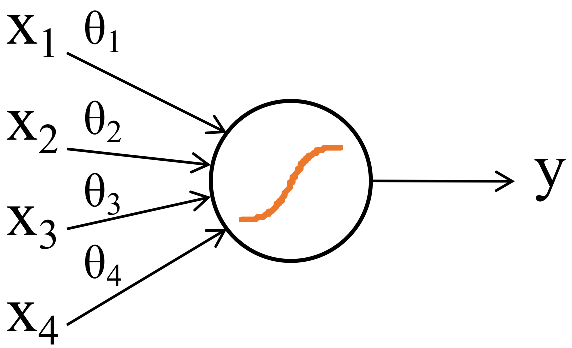
=



Biological basis for neural networks

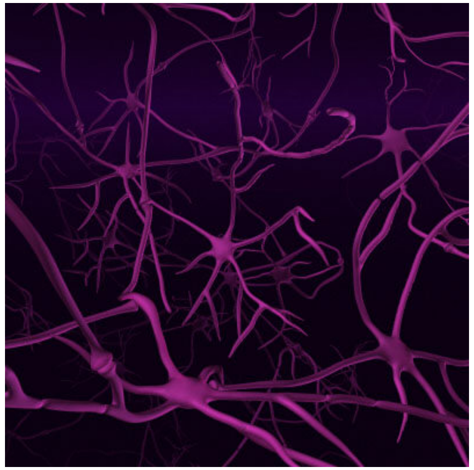
A neuron



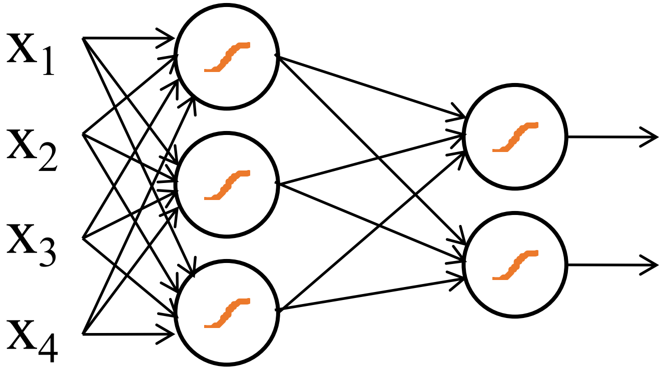


One neuron =
one logistic
regression

Your brain

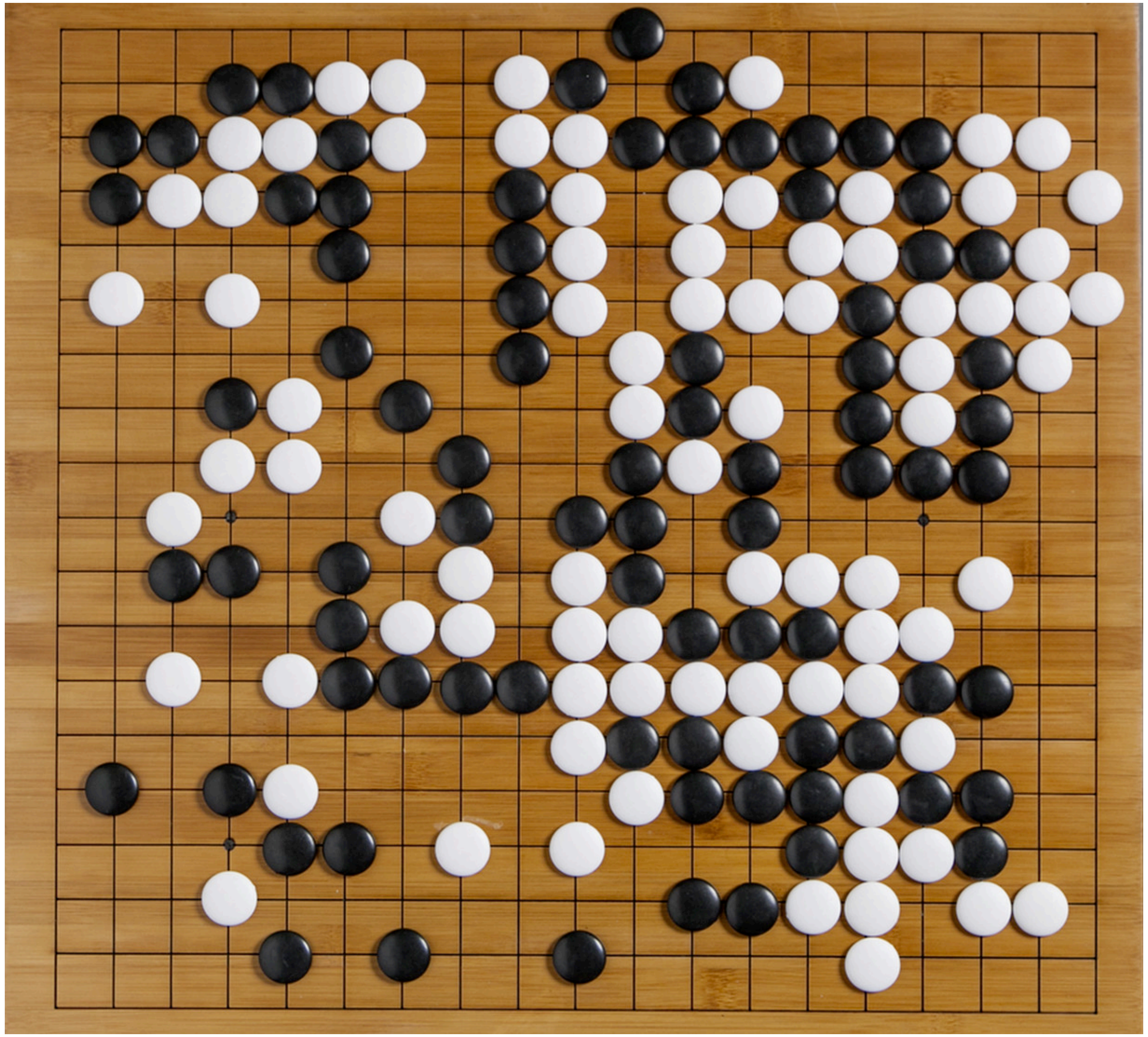


(actually, probably someone else's brain)



Neural network =
many logistic
regressions

Innovations in deep learning



AlphaGO (2016)

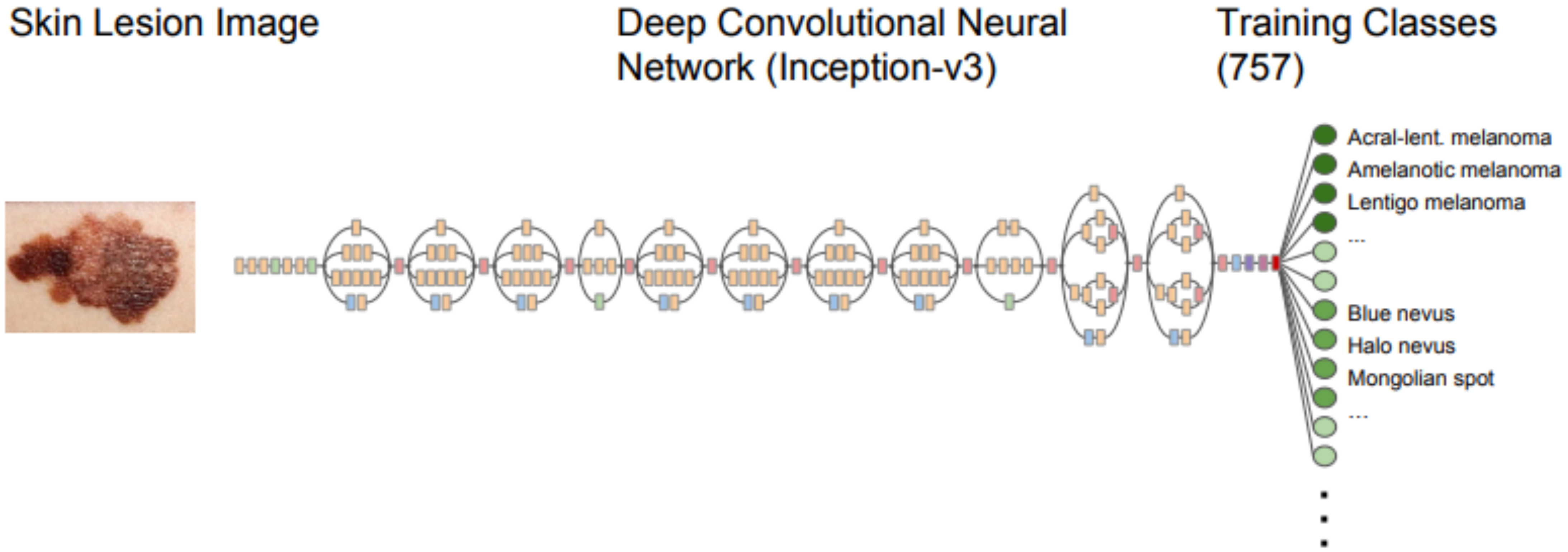
Deep learning (neural networks) is the core idea behind the current revolution in AI.

Computers making art






Detecting skin cancer



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

Deep learning

<u>def</u> **Deep learning** is
maximum likelihood estimation with neural networks.

<u>def</u> 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

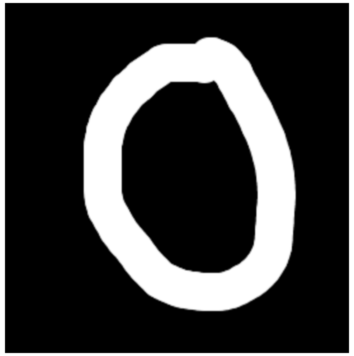
Lots of Logistic (regressions)

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

> 0.5?

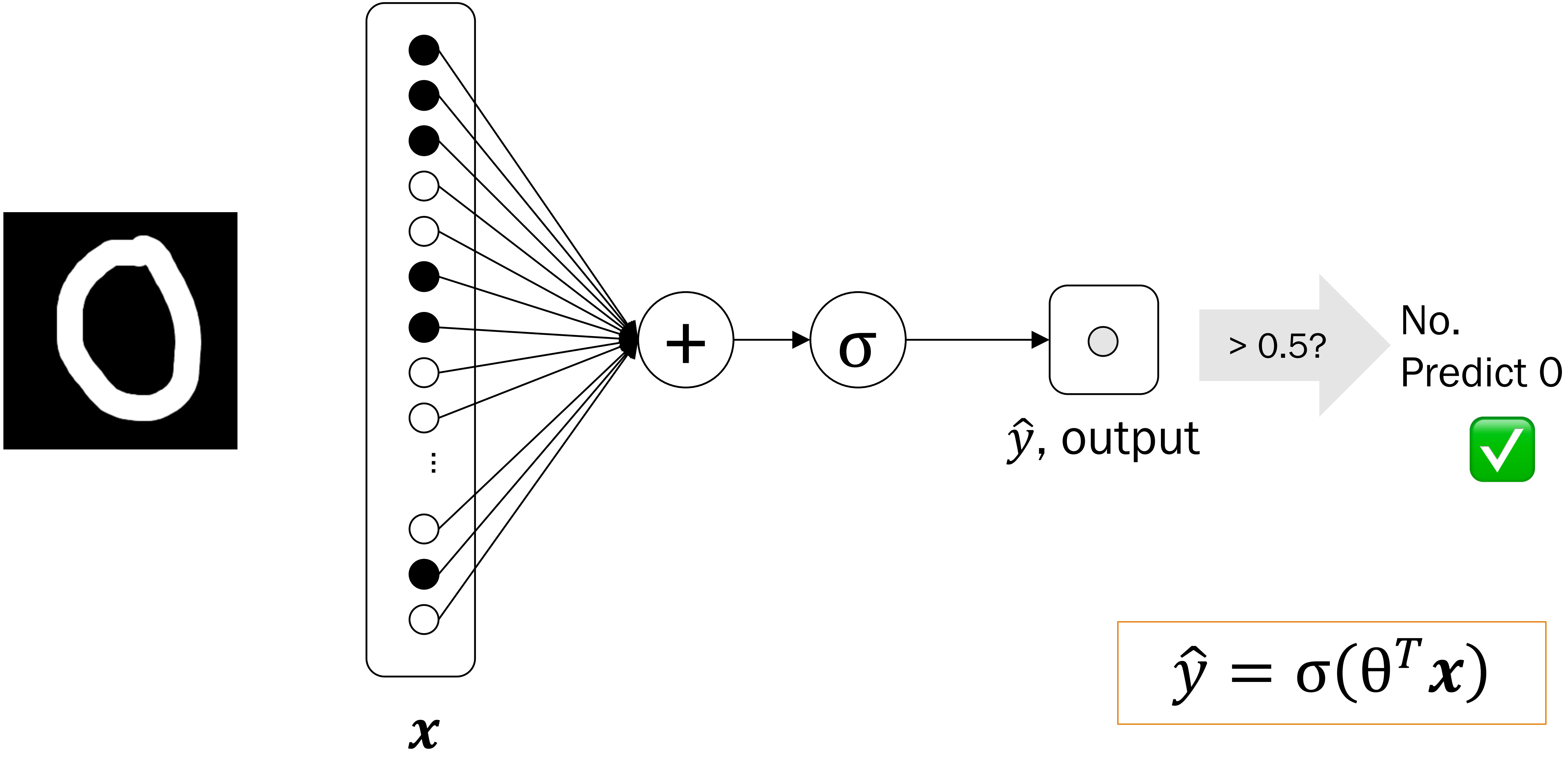
Yes.
Predict 1

Digit recognition example

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

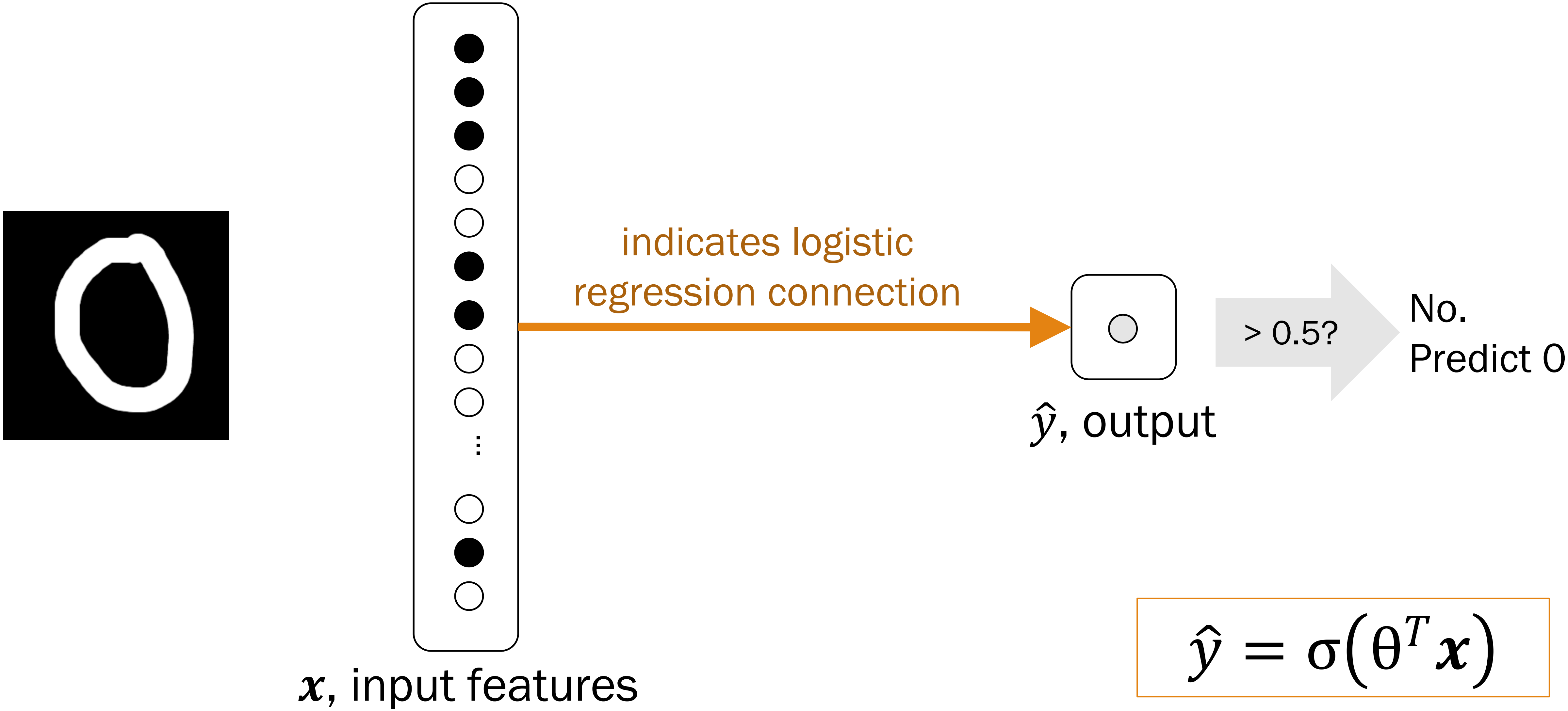
We make feature vectors from (digitized) pictures of numbers.

Logistic Regression



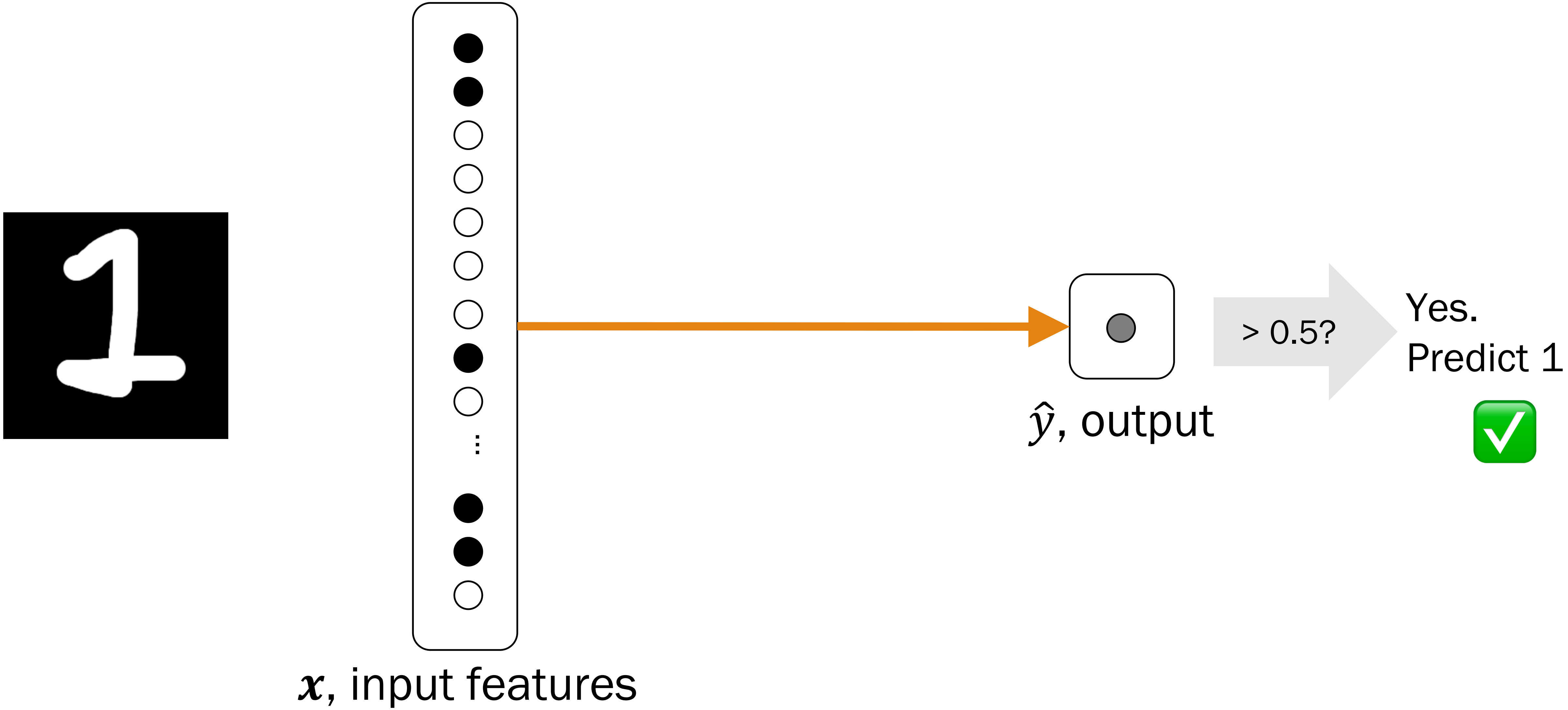
$$\hat{y} = \sigma(\theta^T \boldsymbol{x})$$

Logistic Regression

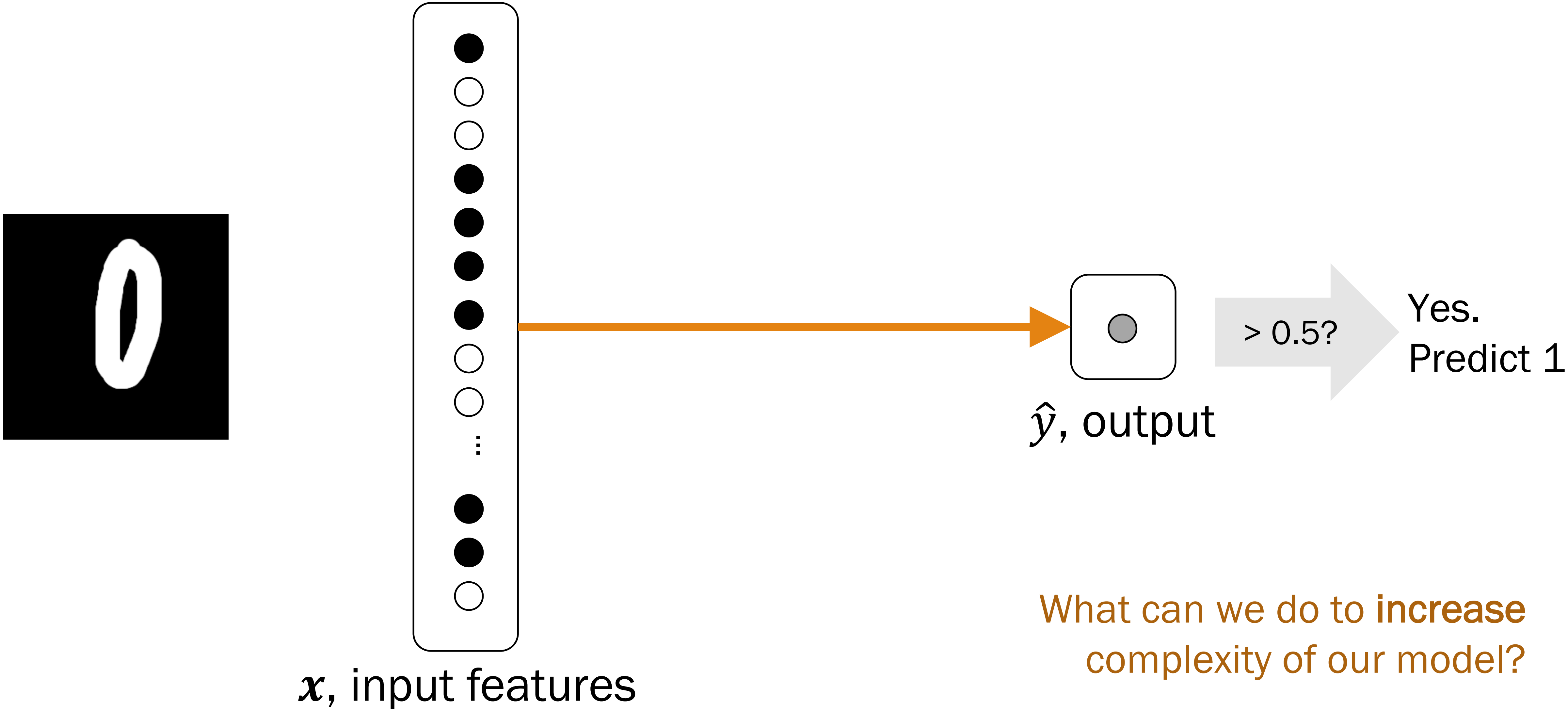


$$\hat{y} = \sigma(\theta^T \boldsymbol{x})$$

Logistic Regression

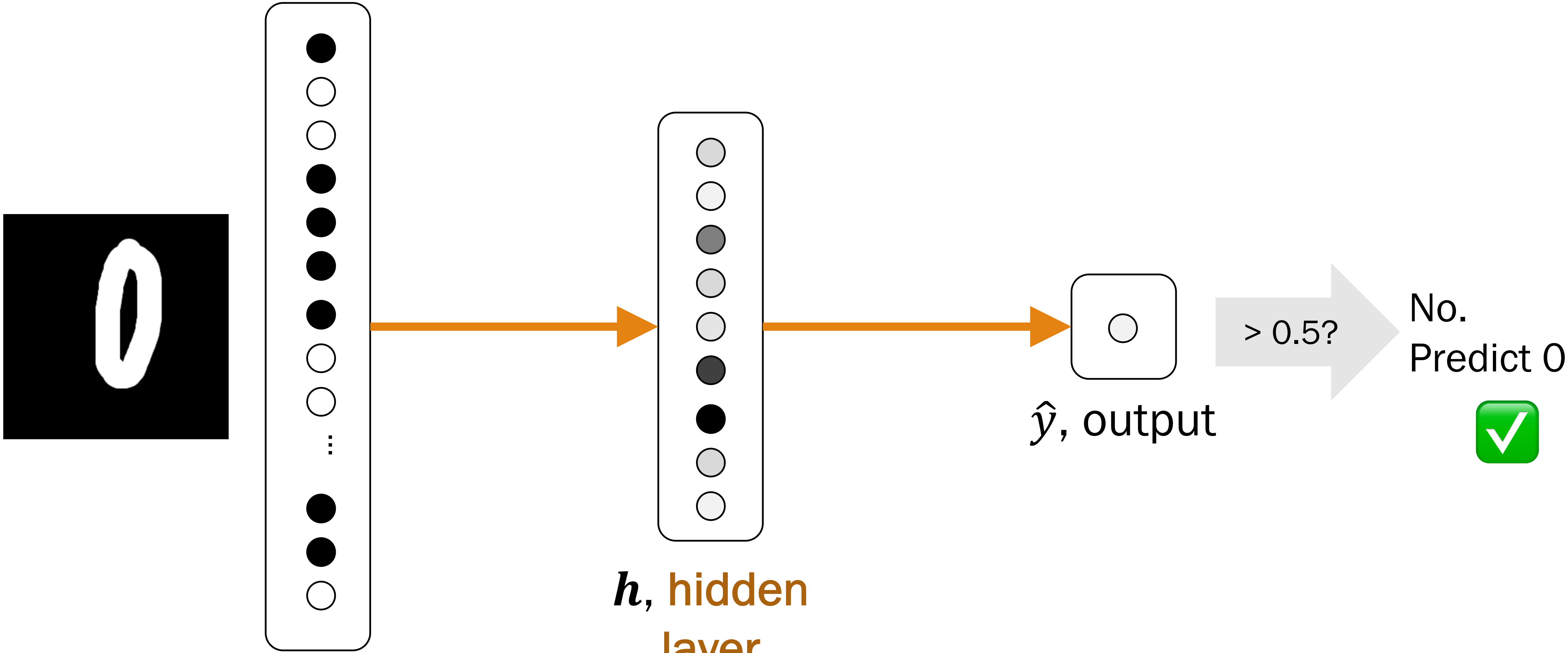


Logistic Regression: not so good



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

Feed neurons into other neurons

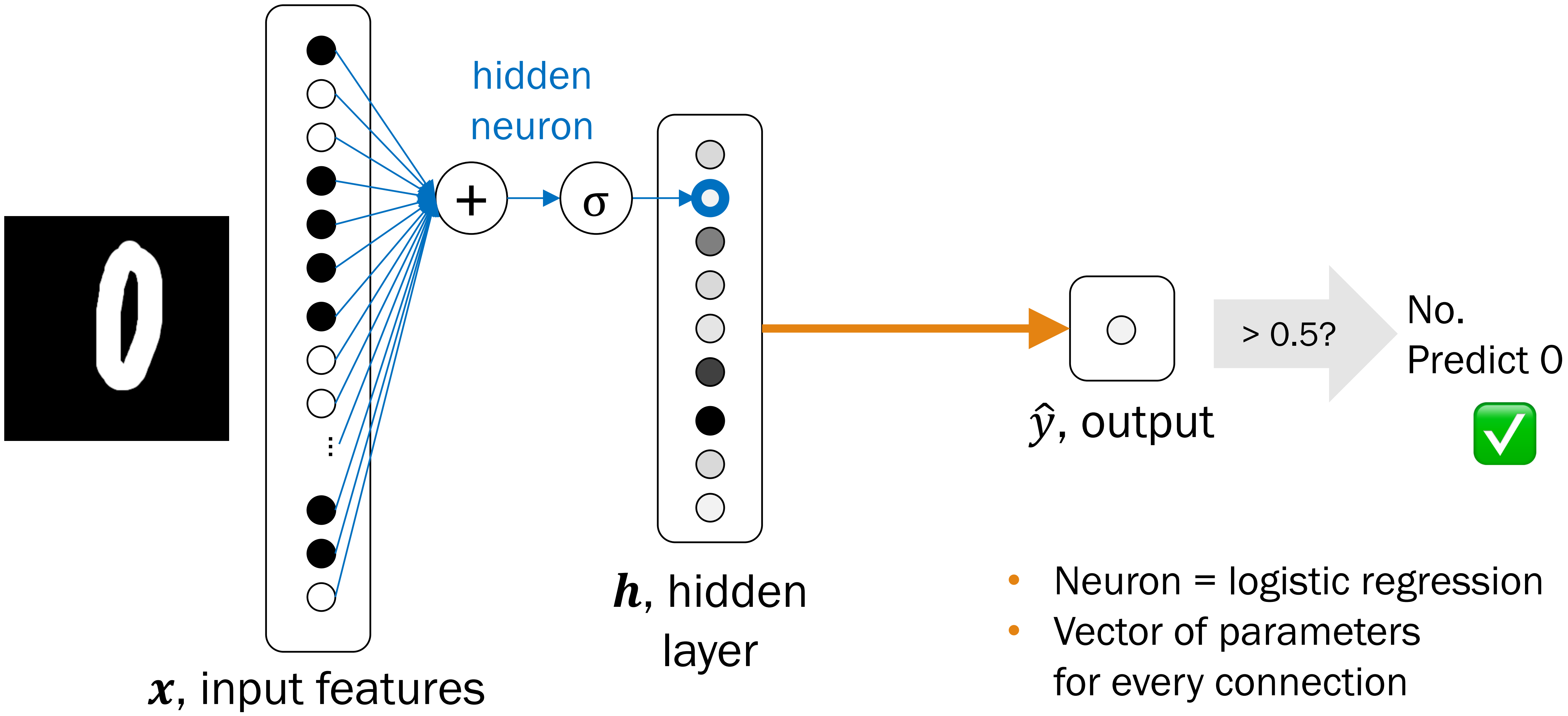


$\boldsymbol{x}$, input features

$\boldsymbol{h}$, hidden layer

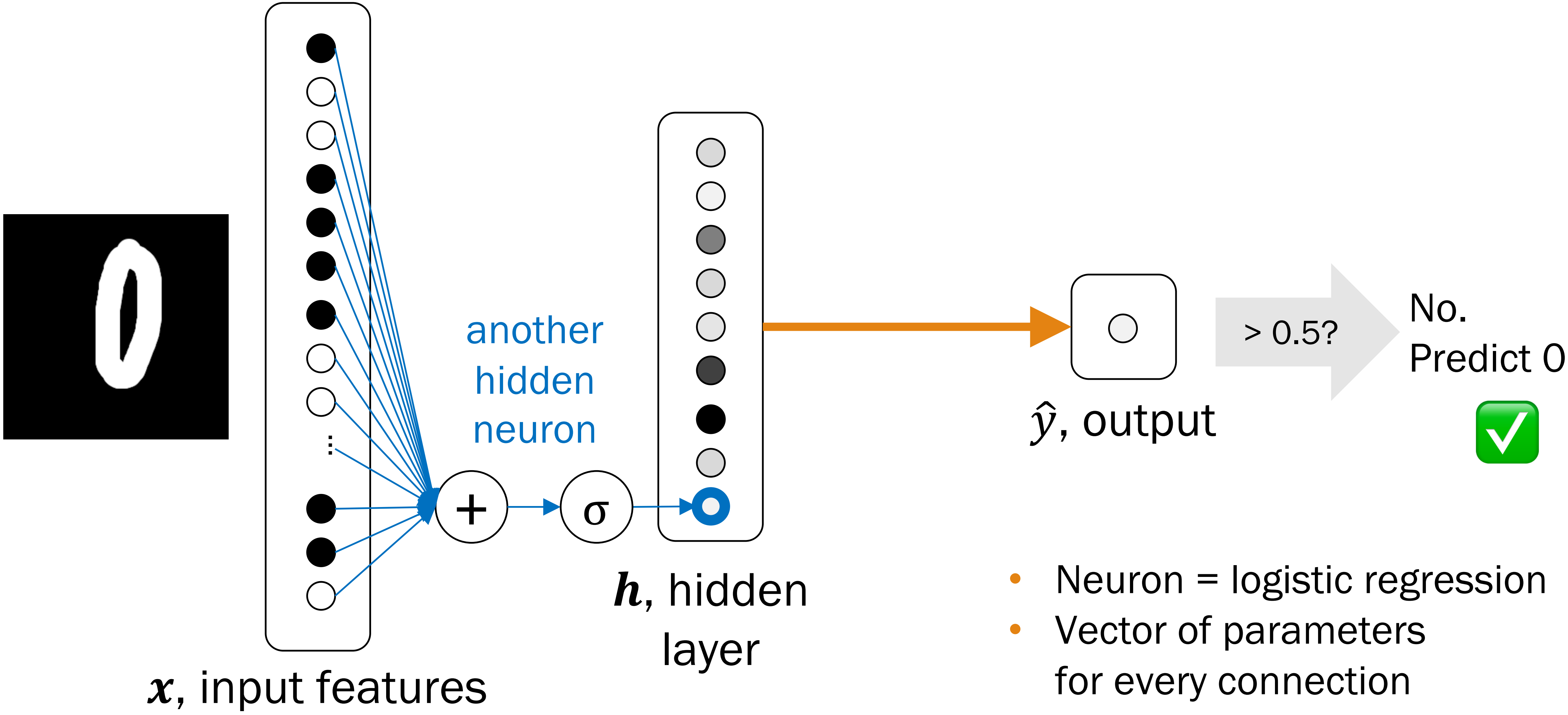
$\hat{y}$, output

Feed neurons into other neurons



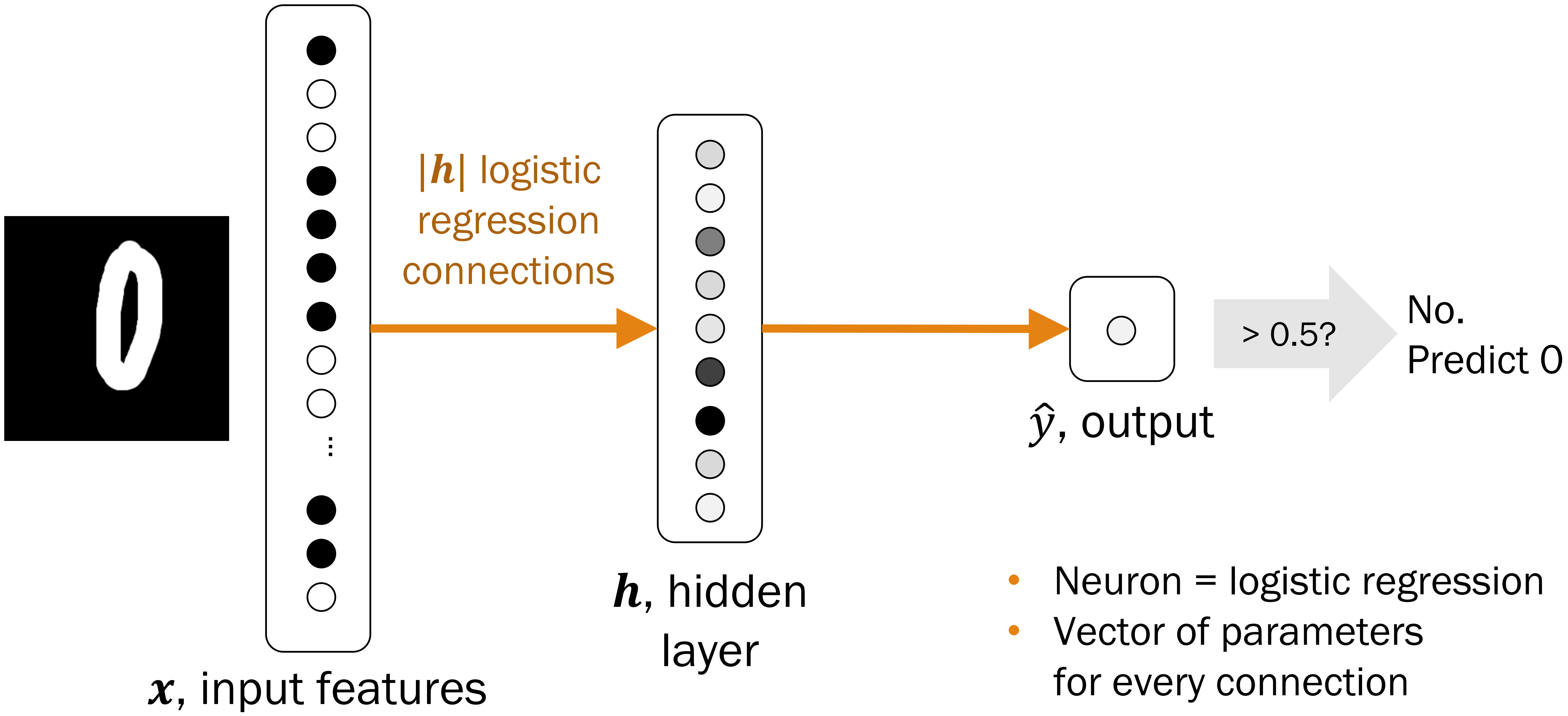
- Neuron = logistic regression
- Vector of parameters for every connection

Feed neurons into other neurons



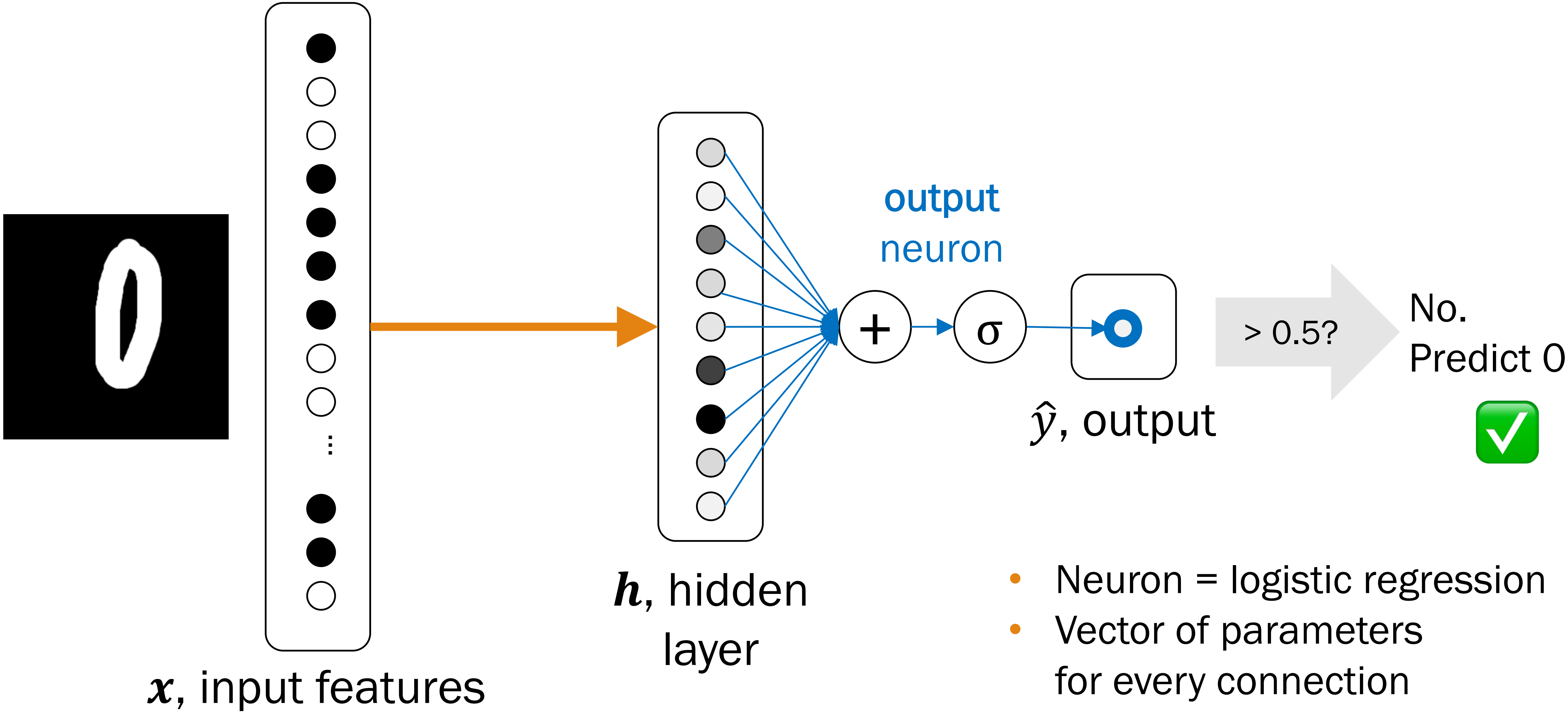
- Neuron = logistic regression
- Vector of parameters for every connection

Feed neurons into other neurons



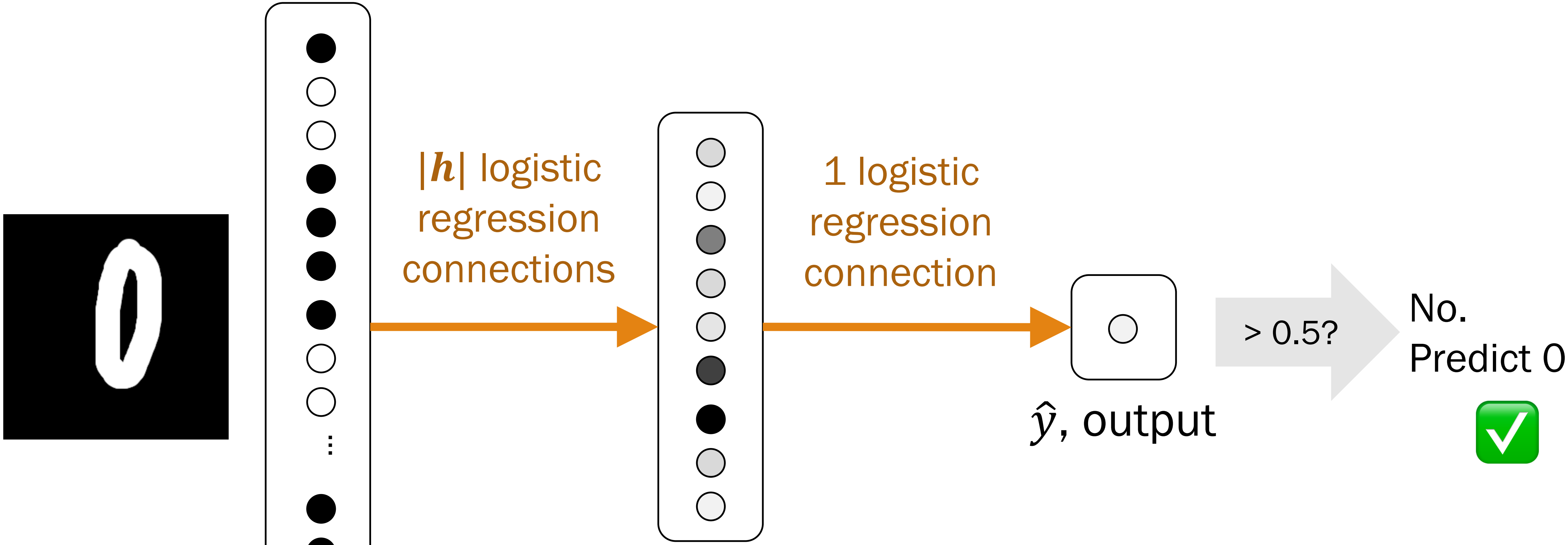
- Neuron = logistic regression
- Vector of parameters for every connection

Feed neurons into other neurons



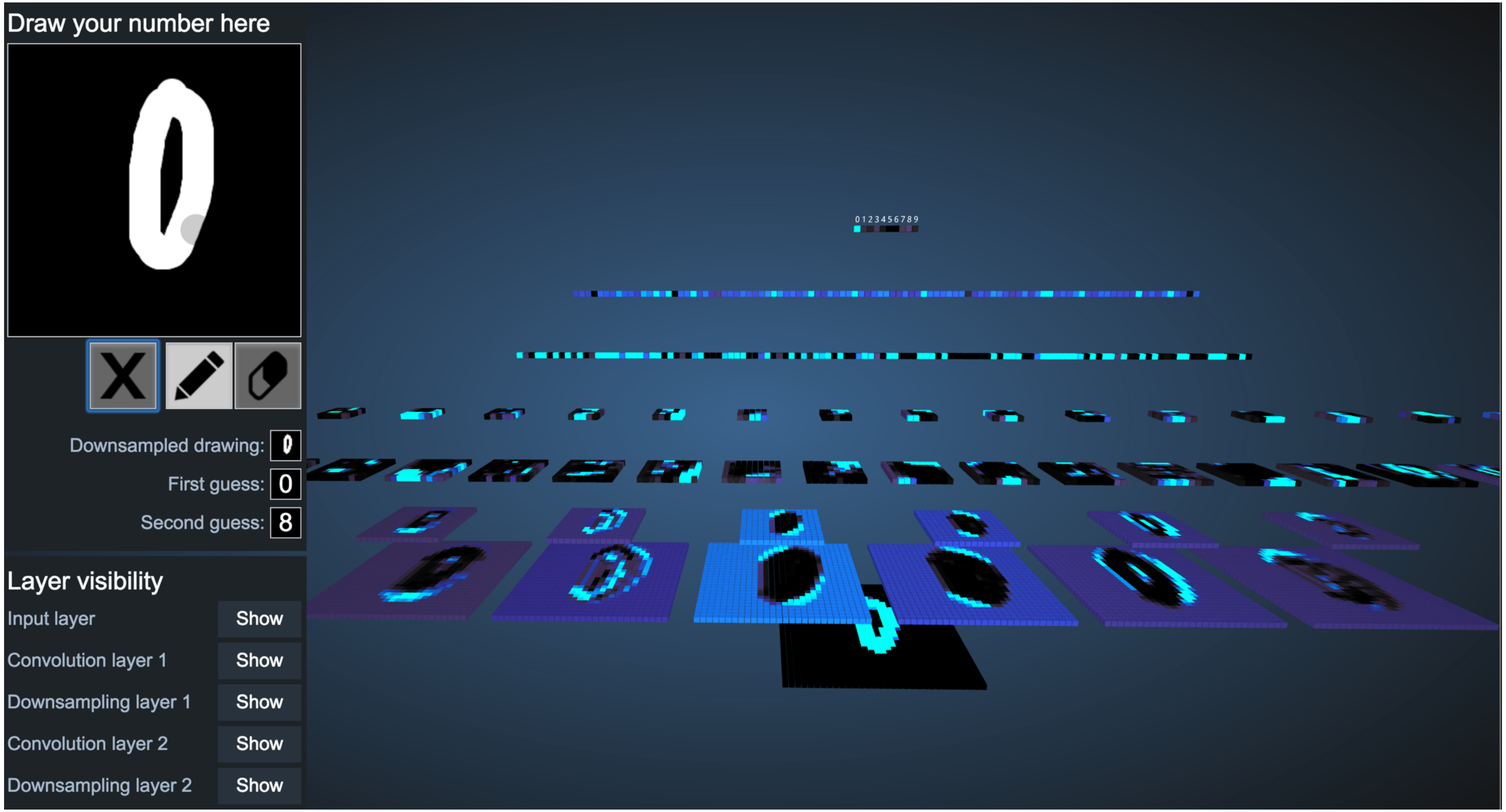
- Neuron = logistic regression
- Vector of parameters for every connection

Feed neurons into other neurons



Neural networks are simply composed logistic regression units.
1 neuron = 1 logistic regression.

Demonstration



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

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 $\hat{y} = P(Y = 1|\boldsymbol{X} = \boldsymbol{x})$
- If $\hat{y} > 0.5$, predict 1. Else, predict 0.

Today's plan

Logistic Regression

- Chapter 0: Background
- Chapter 1: Big Picture
- Chapter 2: Details
- Chapter 3: Philosophy

Intro to Deep Learning

- Parameters of a neural network
- Training neural networks

Learning Goals

- Deep learning (like Logistic Regression) gets its intelligence from its parameters, θ.
- Training a neural network (like Logistic Regression) is finding θ_{MLE}.

Learning goals:

1. Understand Chain Rule as the heart of neural networks
2. Demystifying deep learning as MLE
3. Become experts of logistic regression

Predict: **Forward Pass**

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)

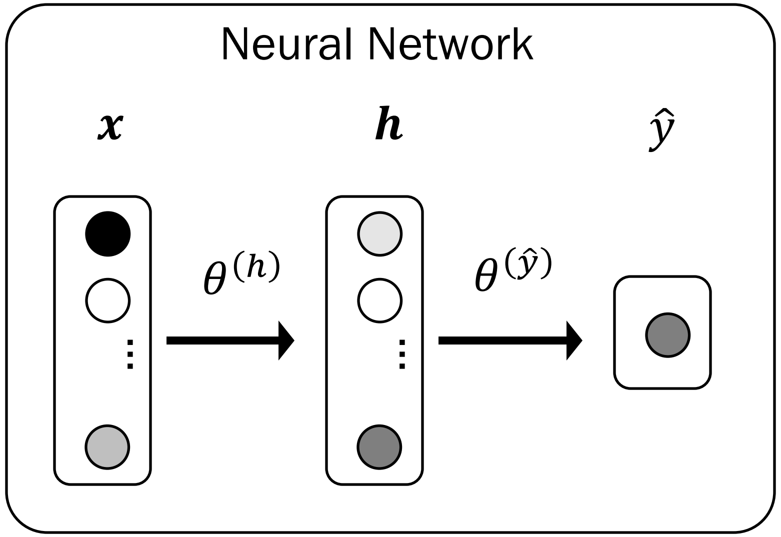
Testing/ Prediction

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

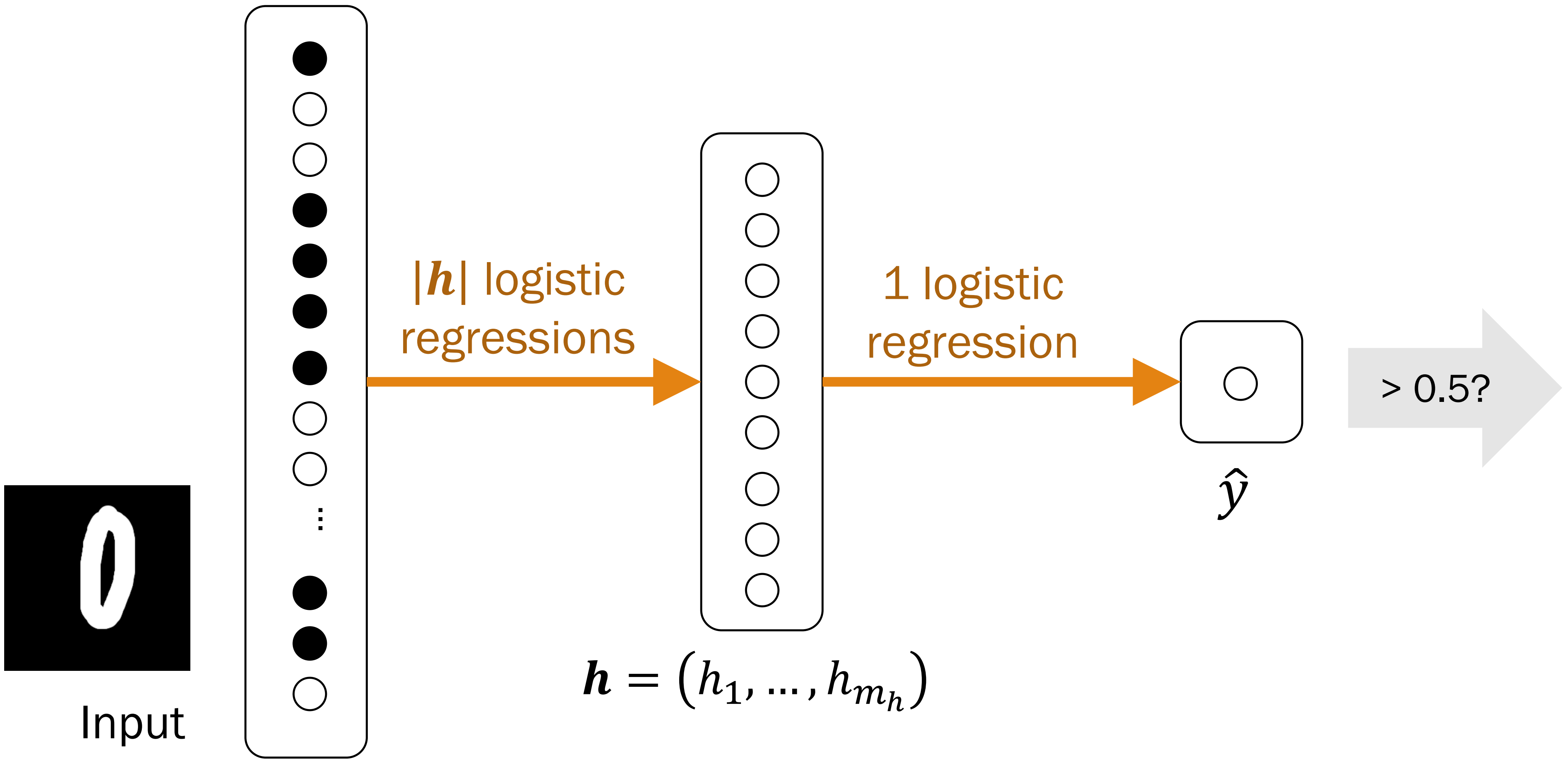
- Use parameters to compute $\hat{y} = P(Y = 1|\boldsymbol{X} = \boldsymbol{x})$
- If $\hat{y} > 0.5$, predict 1. Else, predict 0.

Predict: **Forward Pass**

To predict, make a **forward pass** through the network.



Predict: **Forward Pass**

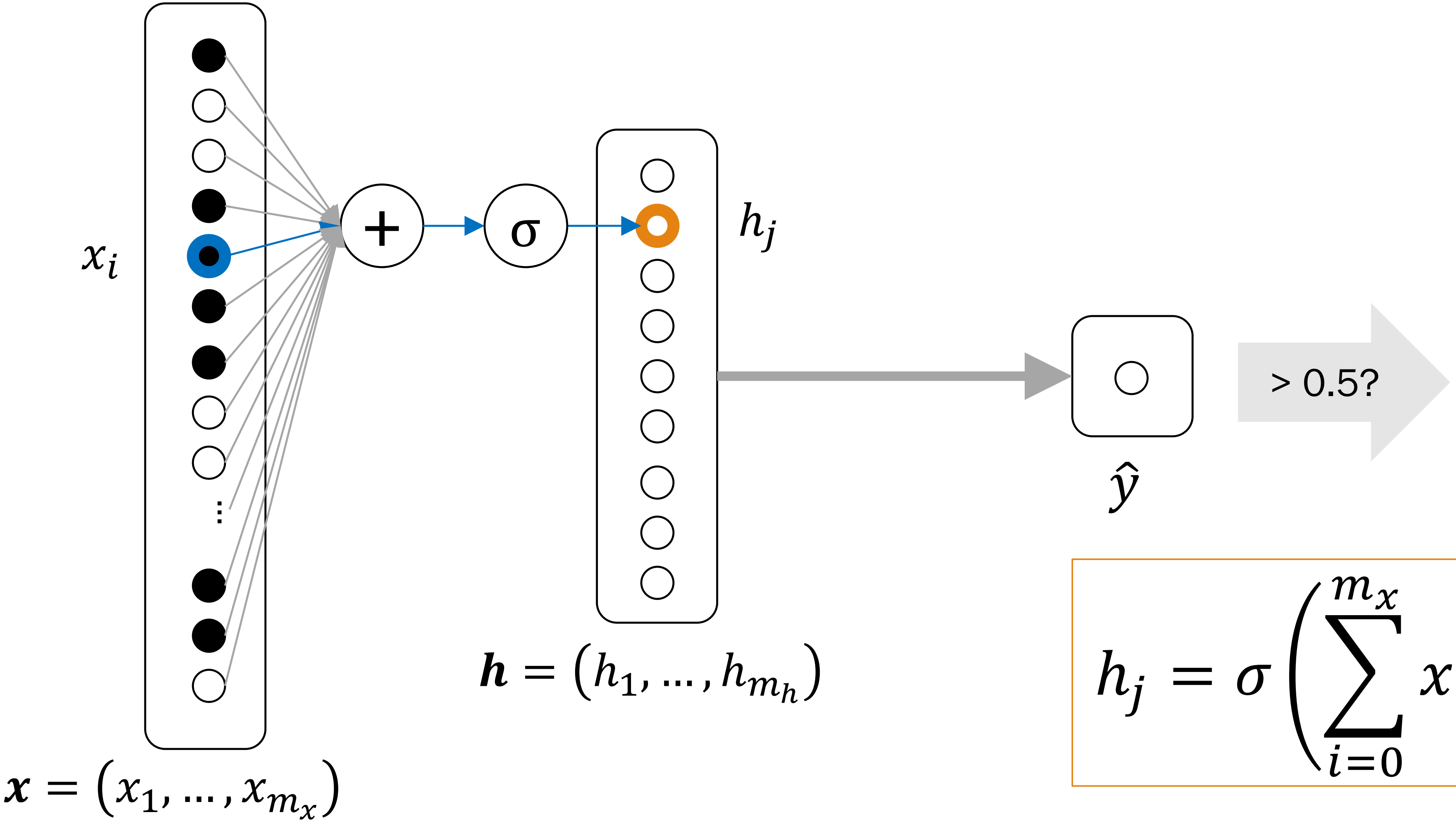


$\boldsymbol{x} = (x_1, \ldots, x_{m_x})$

$\boldsymbol{h} = (h_1, \ldots, h_{m_h})$

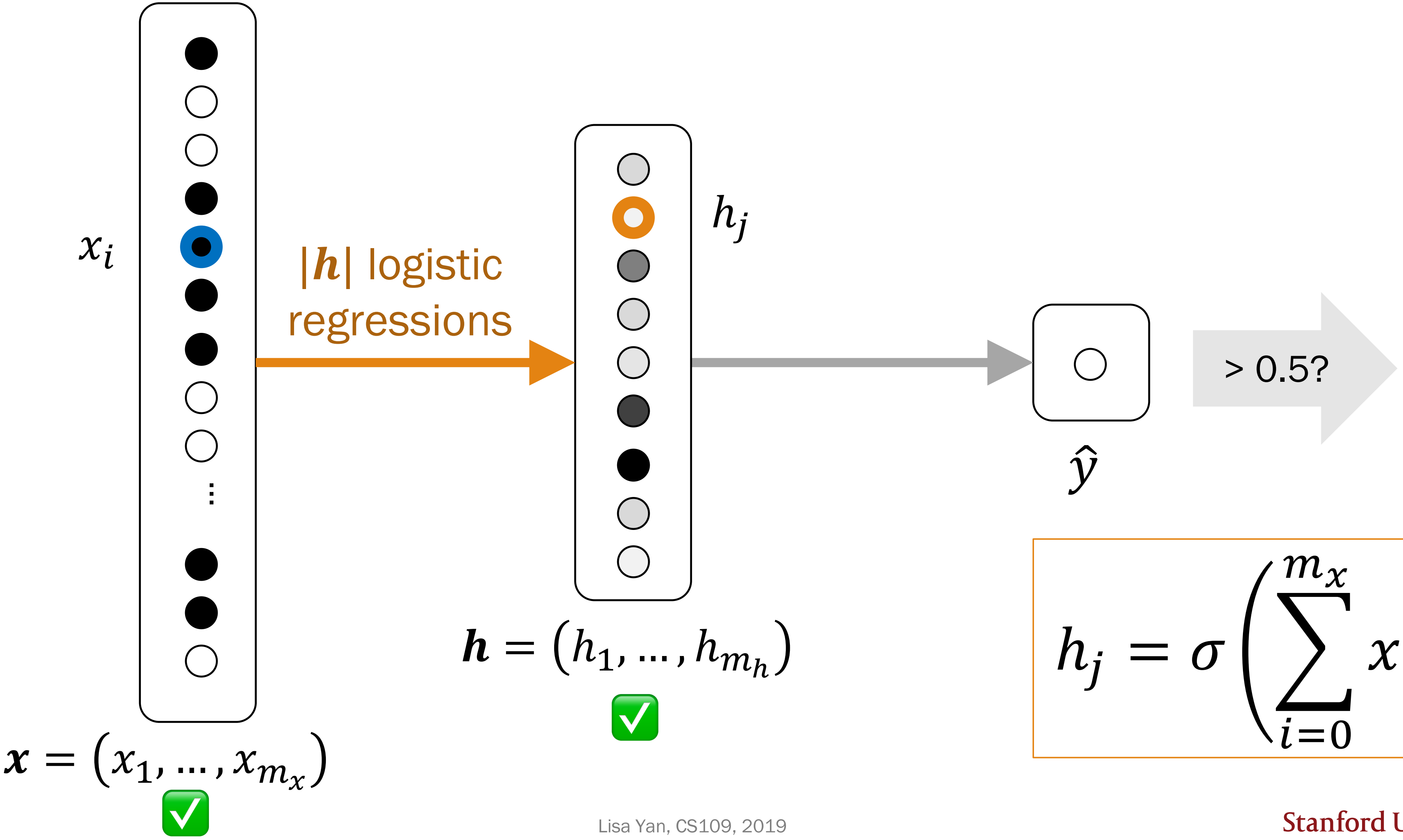
$\hat{y}$

Predict: **Forward Pass**



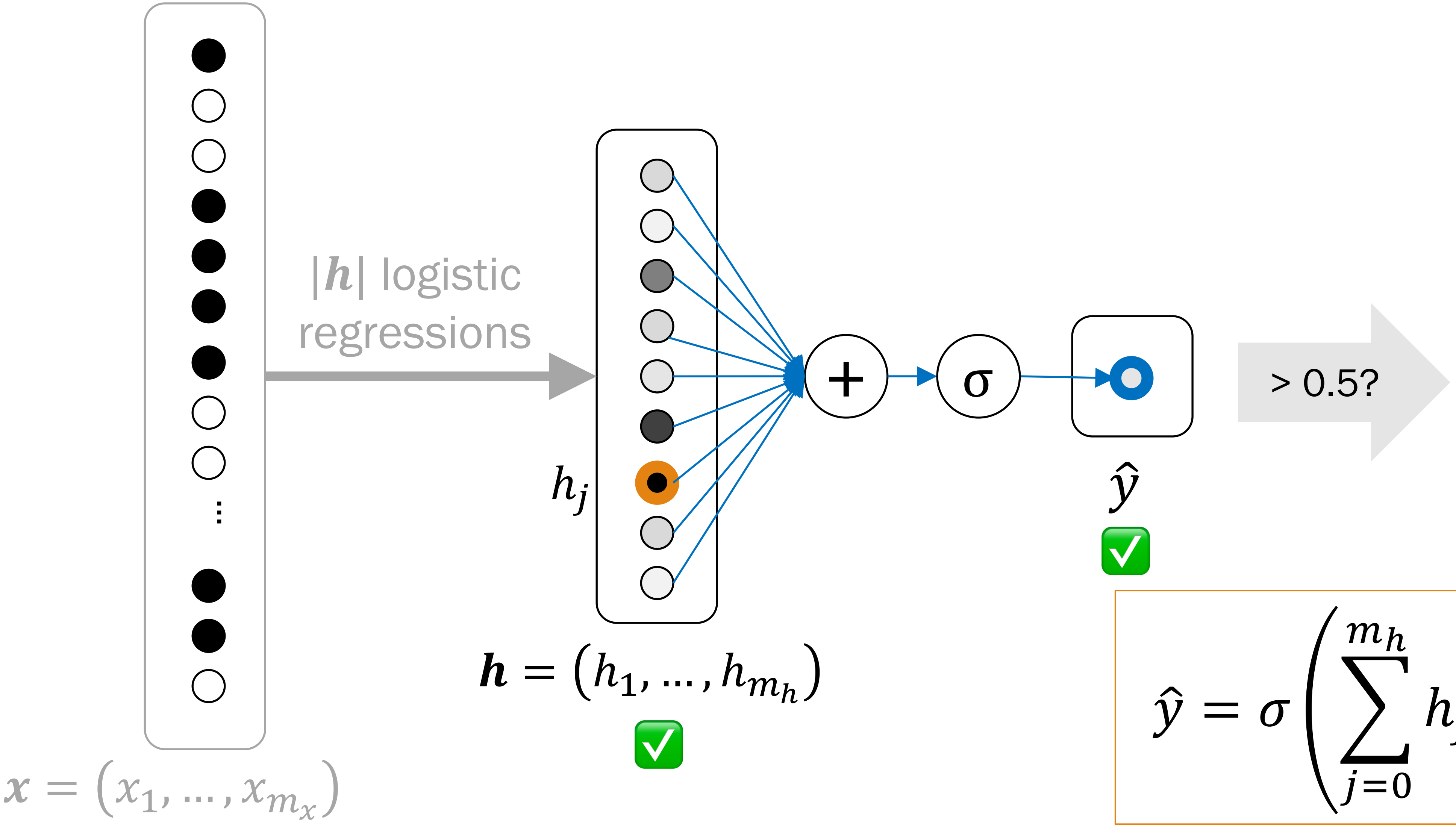
$$h_j = \sigma\left(\sum_{i=0}^{m_x} x_i \theta_{i,j}^{(h)}\right)$$

Predict: **Forward Pass**



$$h_j = \sigma\left(\sum_{i=0}^{m_x} x_i \theta_{i,j}^{(h)}\right)$$

Predict: **Forward Pass**

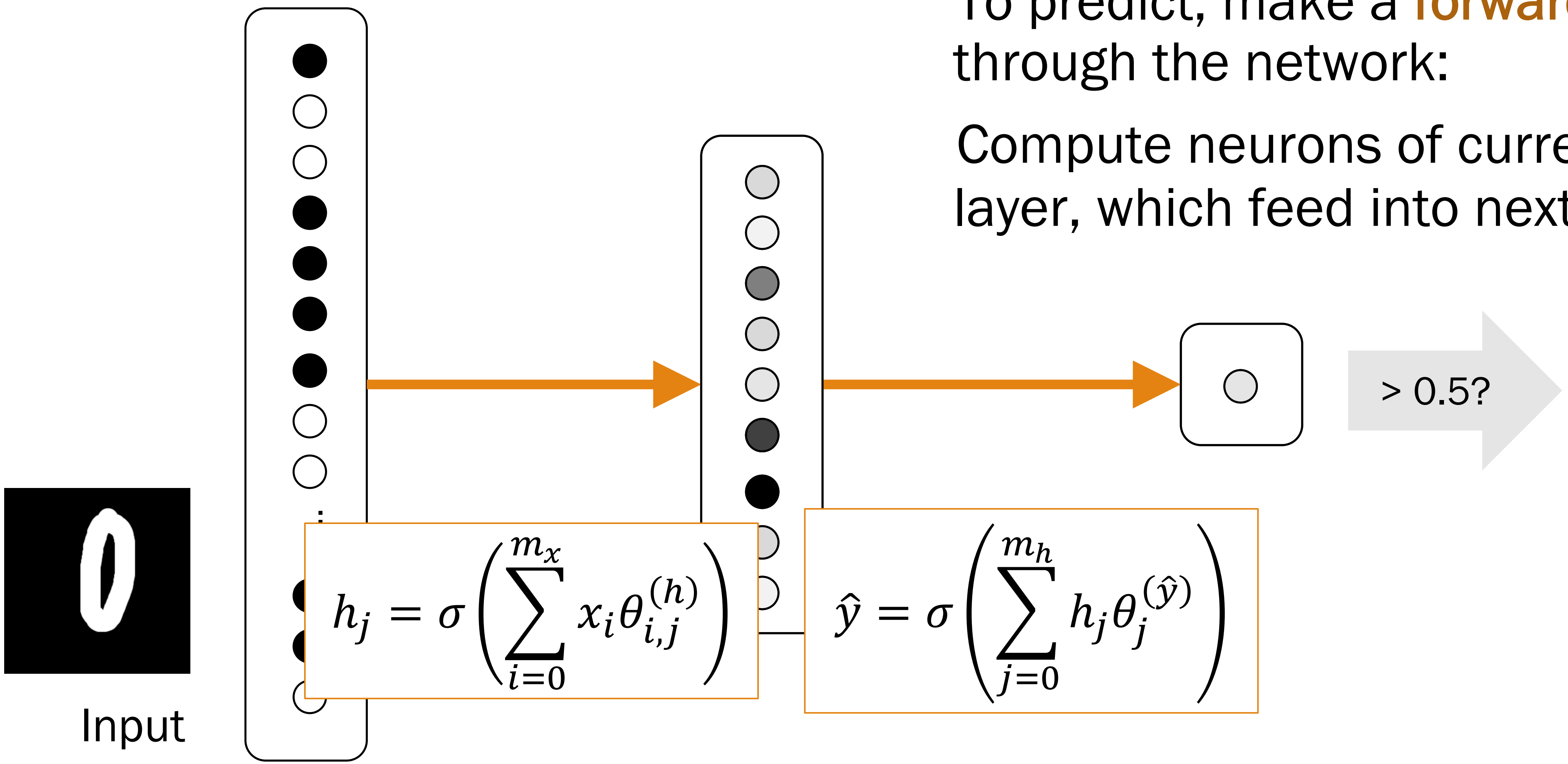


$$\hat{y} = \sigma\left(\sum_{j=0}^{m_h} h_j \theta_j^{(\hat{y})}\right)$$

Predict: **Forward Pass**

To predict, make a **forward pass** through the network:

Compute neurons of current layer, which feed into next layer

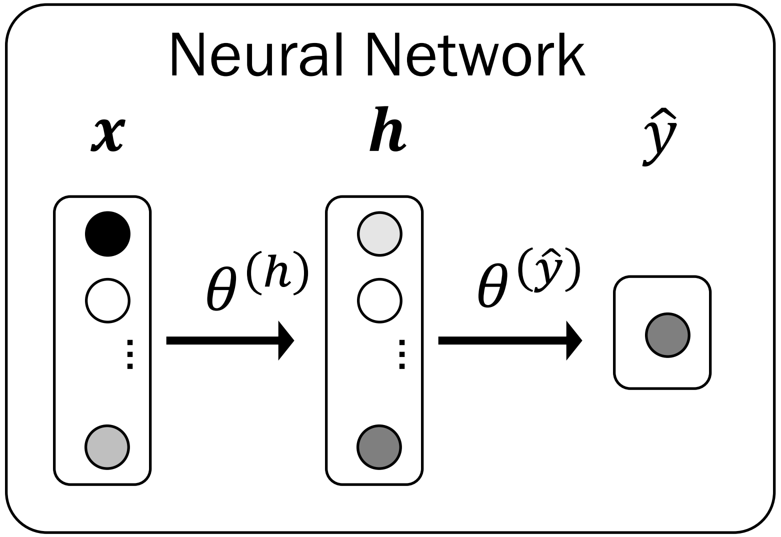


$$h_j = \sigma\left(\sum_{i=0}^{m_x} x_i \theta_{i,j}^{(h)}\right)$$

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

$\boldsymbol{x} = (x_1, \dots, x_{m_x})$ $\quad$ $\boldsymbol{h} = (h_1, \dots, h_{m_h})$ $\quad$ $\hat{y}$

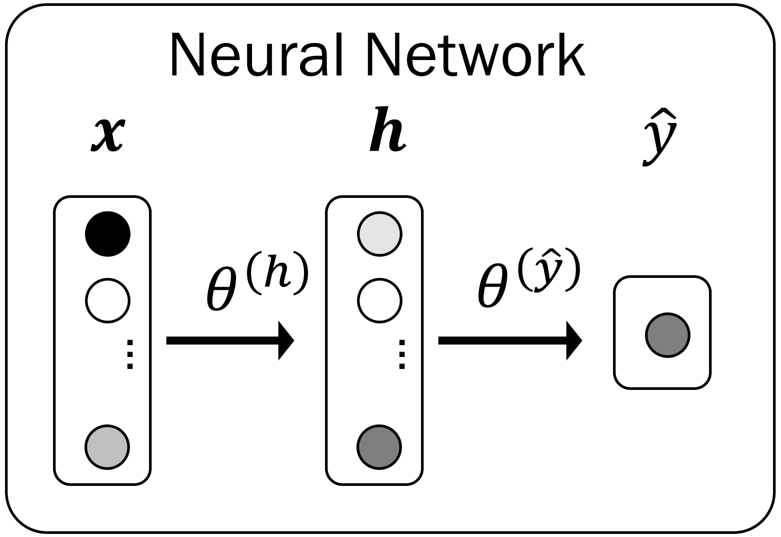
Neural network model



$$h_j = \sigma\left(\sum_{i=0}^{m_x} x_i \theta_{i,j}^{(h)}\right) \qquad \hat{y} = \sigma\left(\sum_{j=0}^{m_h} h_j \theta_j^{(\hat{y})}\right)$$

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

Quick check



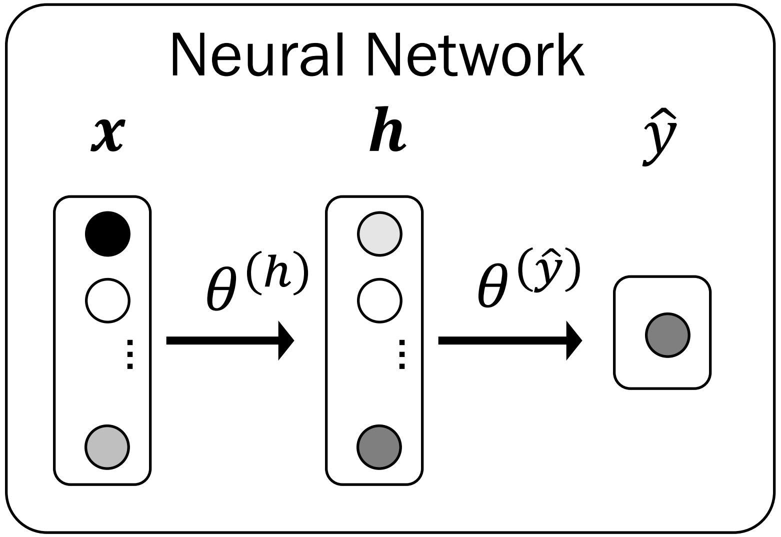
$$h_j = \sigma\left(\sum_{i=0}^{m_x} x_i \theta_{i,j}^{(h)}\right) \qquad \hat{y} = \sigma\left(\sum_{j=0}^{m_h} h_j \theta_j^{(\hat{y})}\right)$$

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

Let $|\boldsymbol{x}| = 40$ and $|\boldsymbol{h}| = 20$.

1. How many parameters are in $\theta^{(\hat{y})}$?
   A. 2
   B. 20
   C. 40
   D. 800
2. How many parameters are in $\theta^{(h)}$?
   A. 2
   B. 20
   C. 40
   D. 800
3. How many parameters in total?
   A. 800
   B. 20
   C. 820
   D. 16000

Quick check



$$h_j = \sigma\left(\sum_{i=0}^{m_x} x_i \theta_{i,j}^{(h)}\right) \qquad \hat{y} = \sigma\left(\sum_{j=0}^{m_h} h_j \theta_j^{(\hat{y})}\right)$$

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

Let $|\boldsymbol{x}| = 40$ and $|\boldsymbol{h}| = 20$.

1. How many parameters are in $\theta^{(\hat{y})}$?
   A. 2
   B. 20
   C. 40
   D. 800
2. How many parameters are in $\theta^{(h)}$?
   A. 2
   B. 20
   C. 40
   D. 800
3. How many parameters in total?
   A. 800
   B. 20
   C. 820
   D. 16000