

Deep Learning

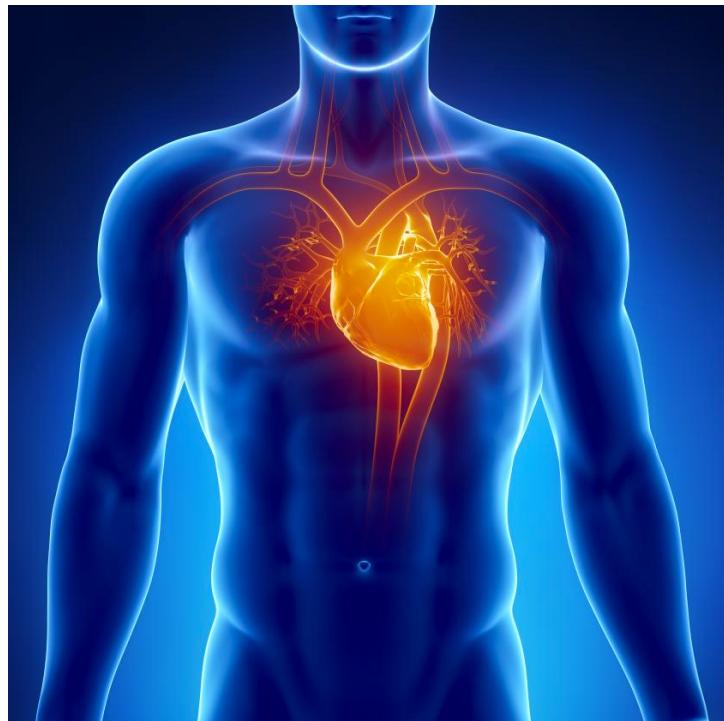
CS 109
Lecture 25
May 23rd, 2016

Today we make a little history

Review

Classification Task

Heart



Ancestry



Netflix

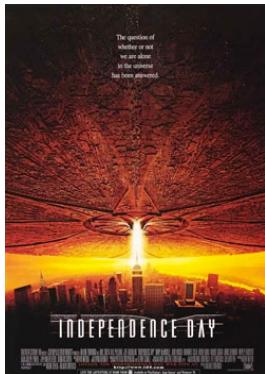


Target Movie “Like” Classification

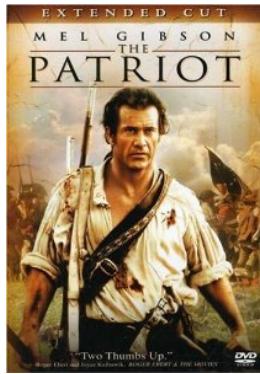
	Movie 1	Movie 2	Movie m	Output
User 1	1	0	1	1
User 2	1	1	0	0
		⋮		⋮
User n	0	0	1	1

Single Instance

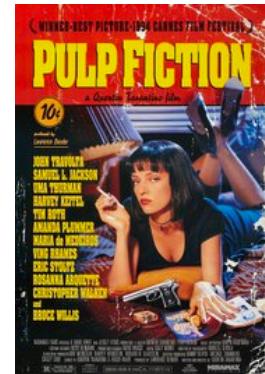
Movie 1



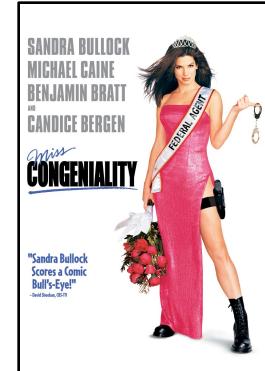
Movie 2



Movie m



Output



User 1

1

0

1

1

User 2

1

1

0

0

:

:

User n

0

0

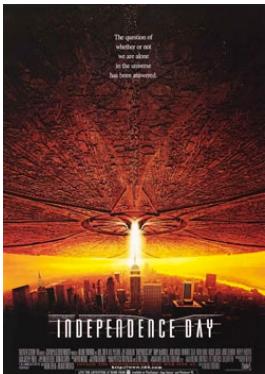
1

1

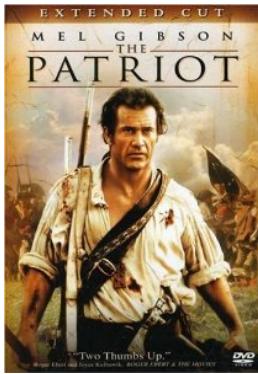
$(\mathbf{x}^{(i)}, y^{(i)})$ such that $1 \leq i \leq n$

Feature Vector

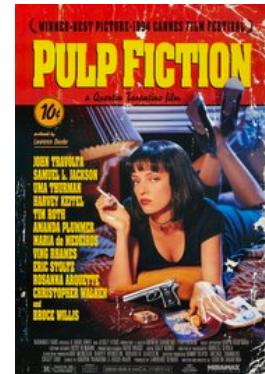
Movie 1



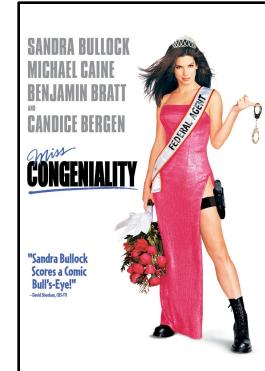
Movie 2



Movie m



Output



User 1

1

0

1

1

User 2

1

1

0

0

⋮

⋮

User n

0

0

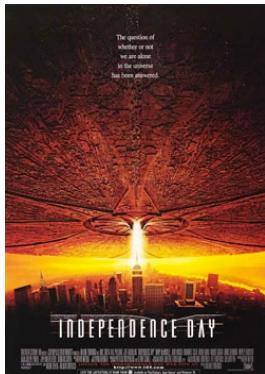
1

1

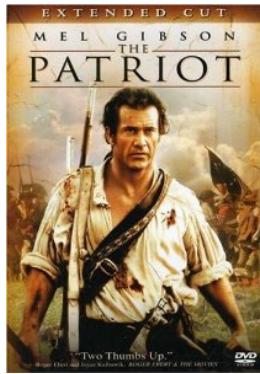
$(\mathbf{x}^{(i)}, y^{(i)})$ such that $1 \leq i \leq n$

Output Value

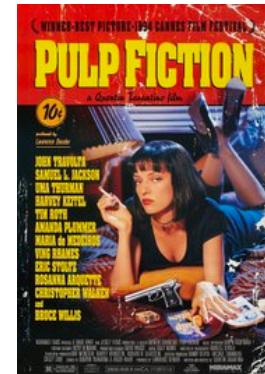
Movie 1



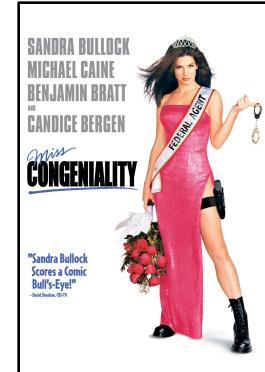
Movie 2



Movie m



Output



User 1

1

0

1

1

User 2

1

1

0

0

⋮

⋮

User n

0

0

1

1

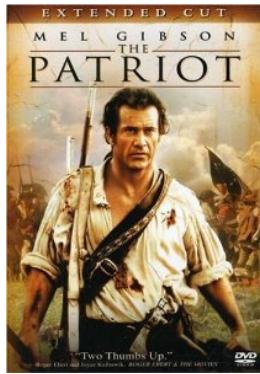
$(\mathbf{x}^{(i)}, y^{(i)})$ such that $1 \leq i \leq n$

Single Feature Value

Movie 1

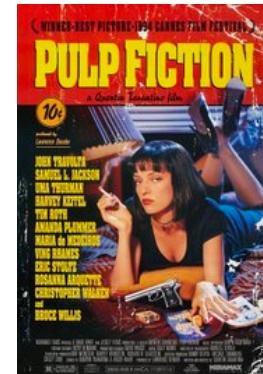


Movie 2

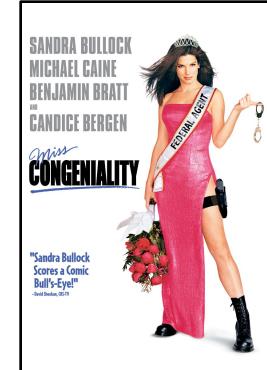


...

Movie m



Output



User 1

1

0

1

1

User 2

1

1

0

0

:

:

User n

0

0

1

1

In general: $\mathbf{x}_j^{(i)}$

In this case: $\mathbf{x}_m^{(2)}$

Healthy Heart Classifier

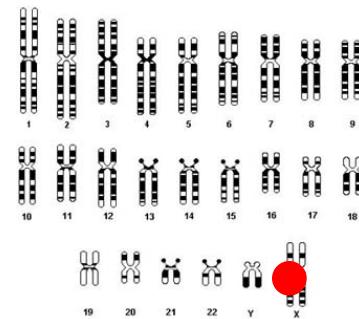
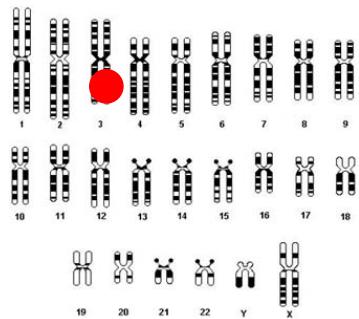
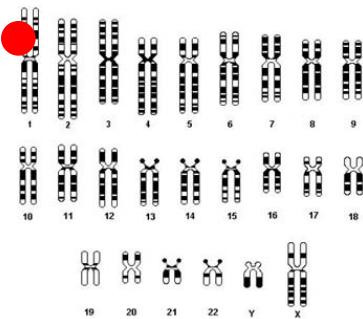
	ROI 1	ROI 2	...	ROI m	Output
Heart 1	0	1		1	0
Heart 2	1	1		1	0
			:		:
Heart n	0	0		0	1

Ancestry Classifier

SNP 1

SNP 2

SNP m



Output



User 1

1

0

1

0

User 2

0

0

1

1

⋮

User n

1

1

0

1

Training Data

Assume IID data:

N training datapoints

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots (\mathbf{x}^{(n)}, y^{(n)})$$

$$m = |\mathbf{x}^{(i)}|$$

Each datapoint has m features and a single output

Notation

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

Sigmoid function

$$\theta^T \mathbf{x} = \sum_{i=1}^n \theta_i x_i$$

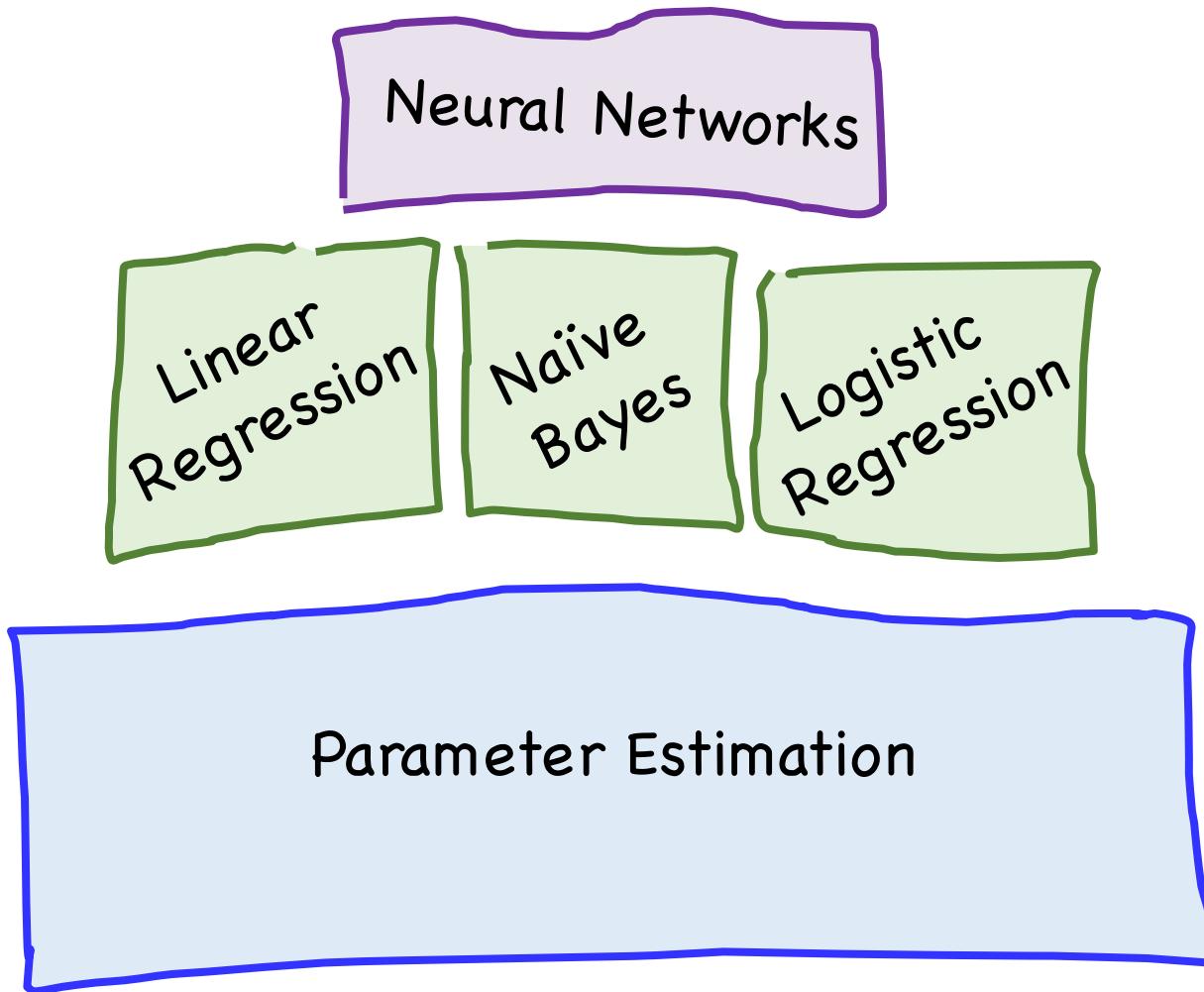
Weighted sum
(aka dot product)

$$= \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

$$\sigma(\theta^T \mathbf{x}) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}}$$

Sigmoid function of
weighted sum

Our Path



Logistic Regression Assumption

- Model *conditional* likelihood $P(Y | \mathbf{X})$ directly
 - Model this probability with *logistic* function:

$$P(Y = 1 | \mathbf{X}) = \sigma(z) \text{ where } z = \theta_0 + \sum_{i=1}^m \theta_i x_i$$

- For simplicity define $x_0 = 1$ so $z = \theta^T \mathbf{x}$
- Since $P(Y = 0 | \mathbf{X}) + P(Y = 1 | \mathbf{X}) = 1$:

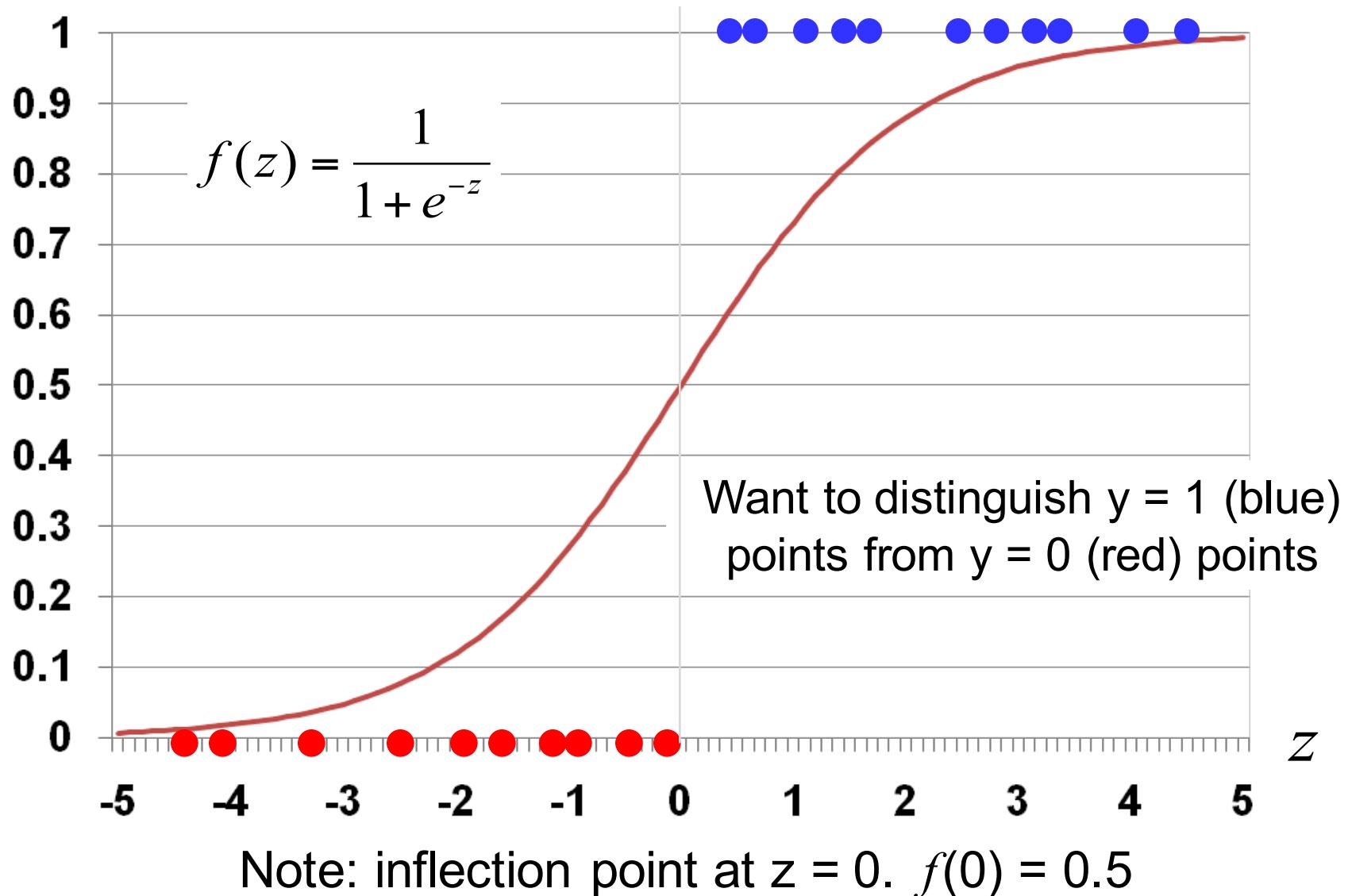
$$P(Y = 1 | X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

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

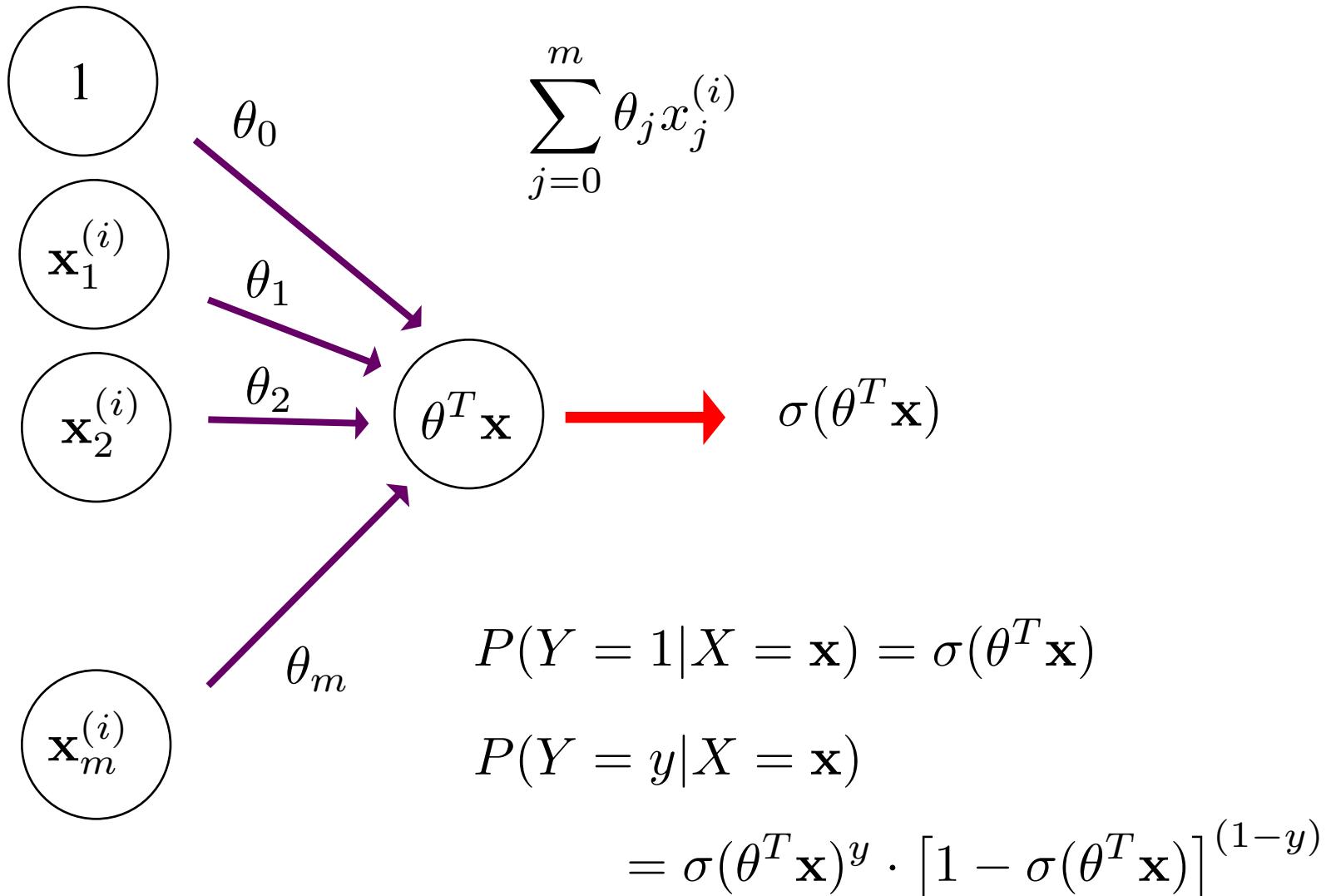
Recall:
Sigmoid function

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

The Sigmoid Function

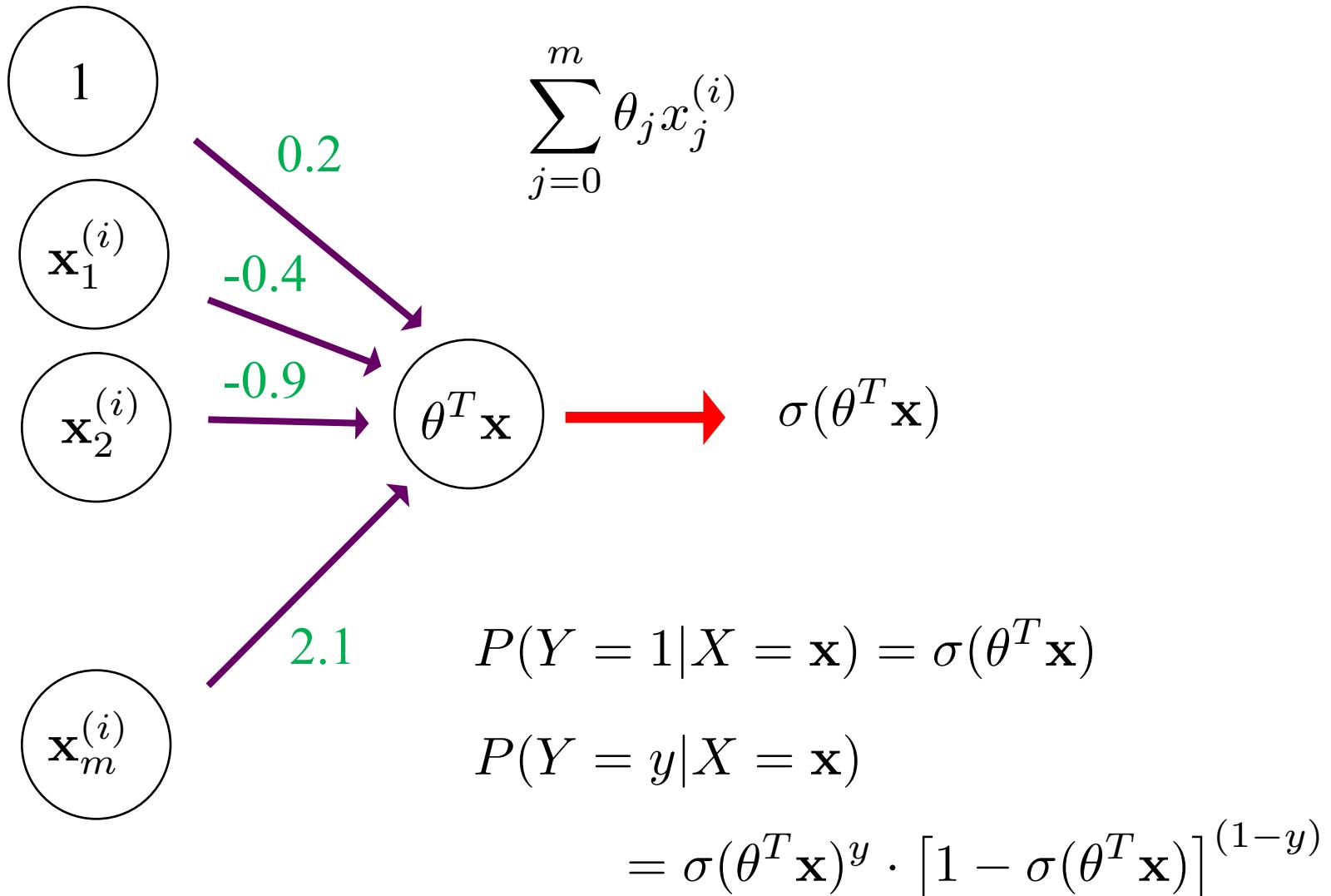


Logistic Regression



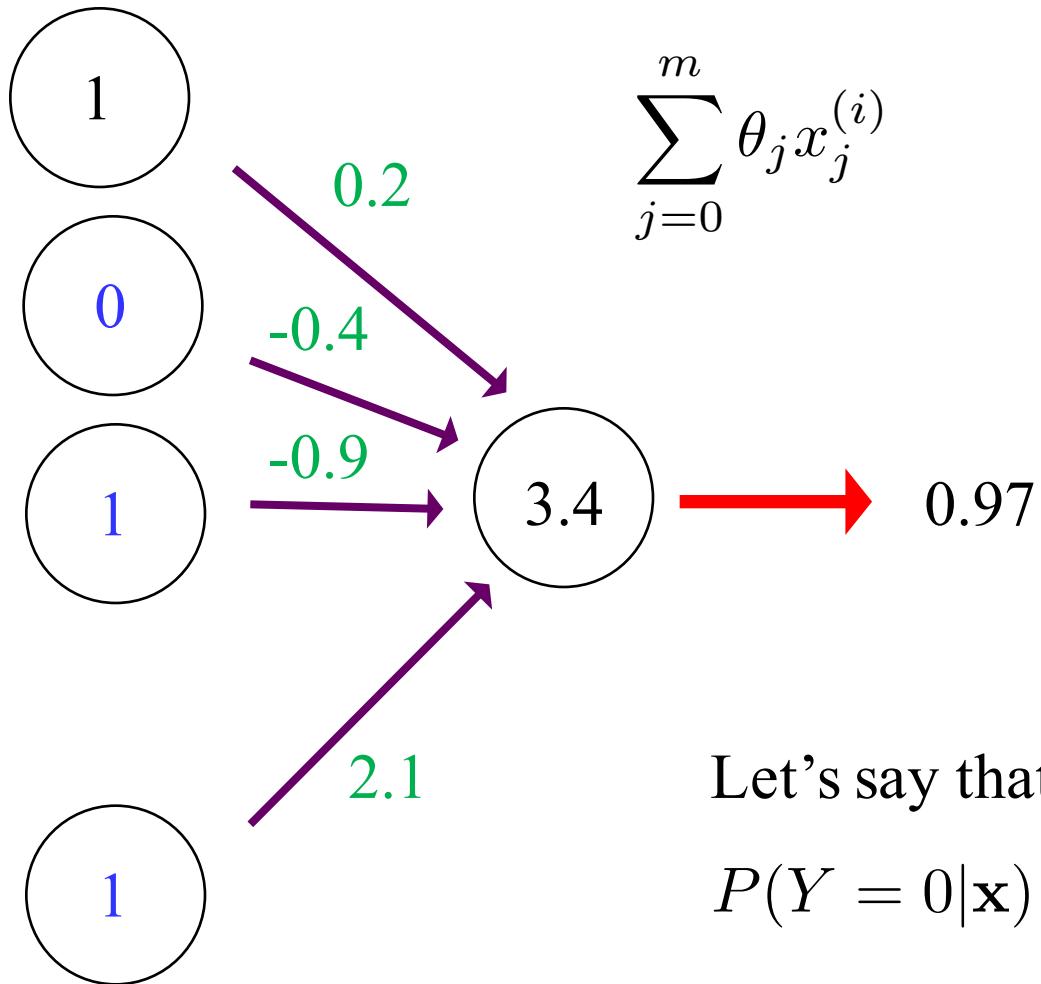
You can calculate likelihood of data, θ

Logistic Regression



You can calculate likelihood of data, θ

Logistic Regression

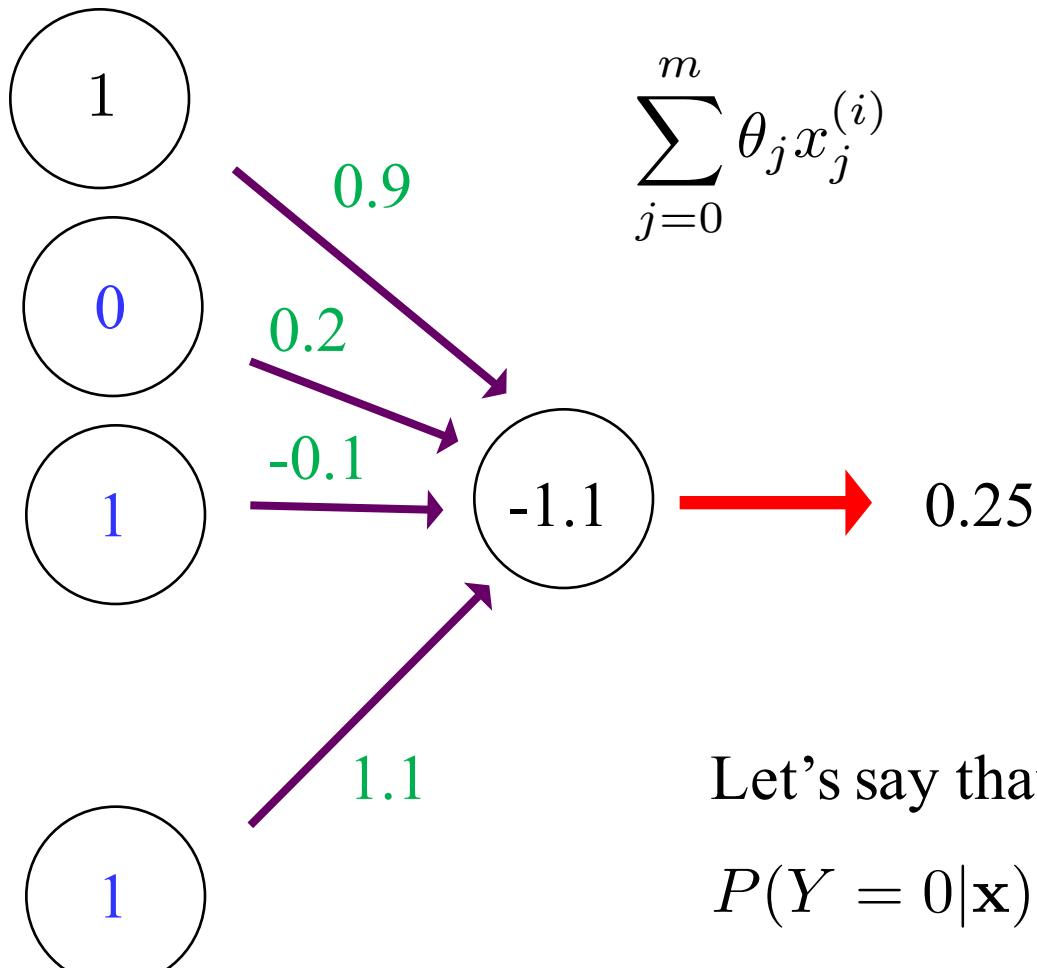


Let's say that $y = 0$:

$$\begin{aligned} P(Y = 0 | \mathbf{x}) &= (1 - \theta^T \mathbf{x}) \\ &= 0.03 \end{aligned}$$

We should change θ

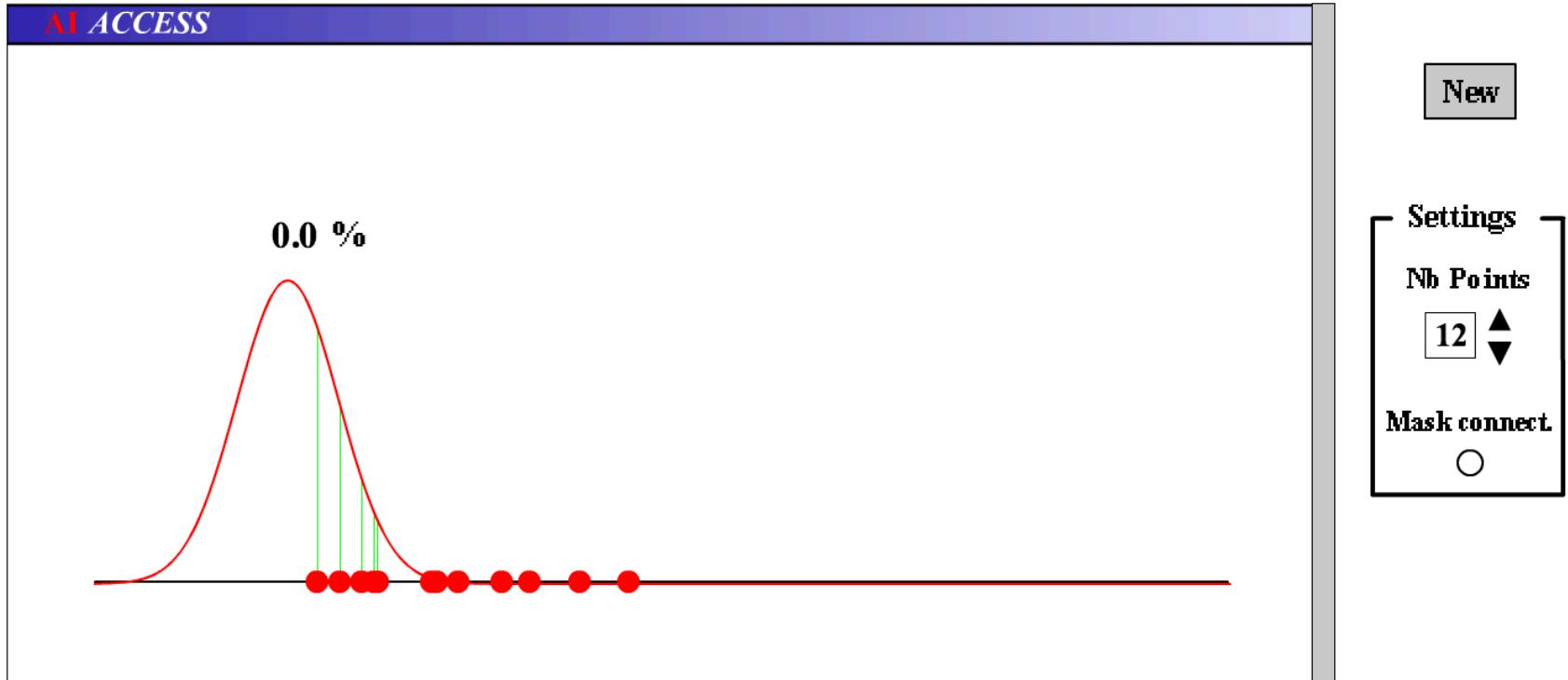
These Parameters are Better



Better...

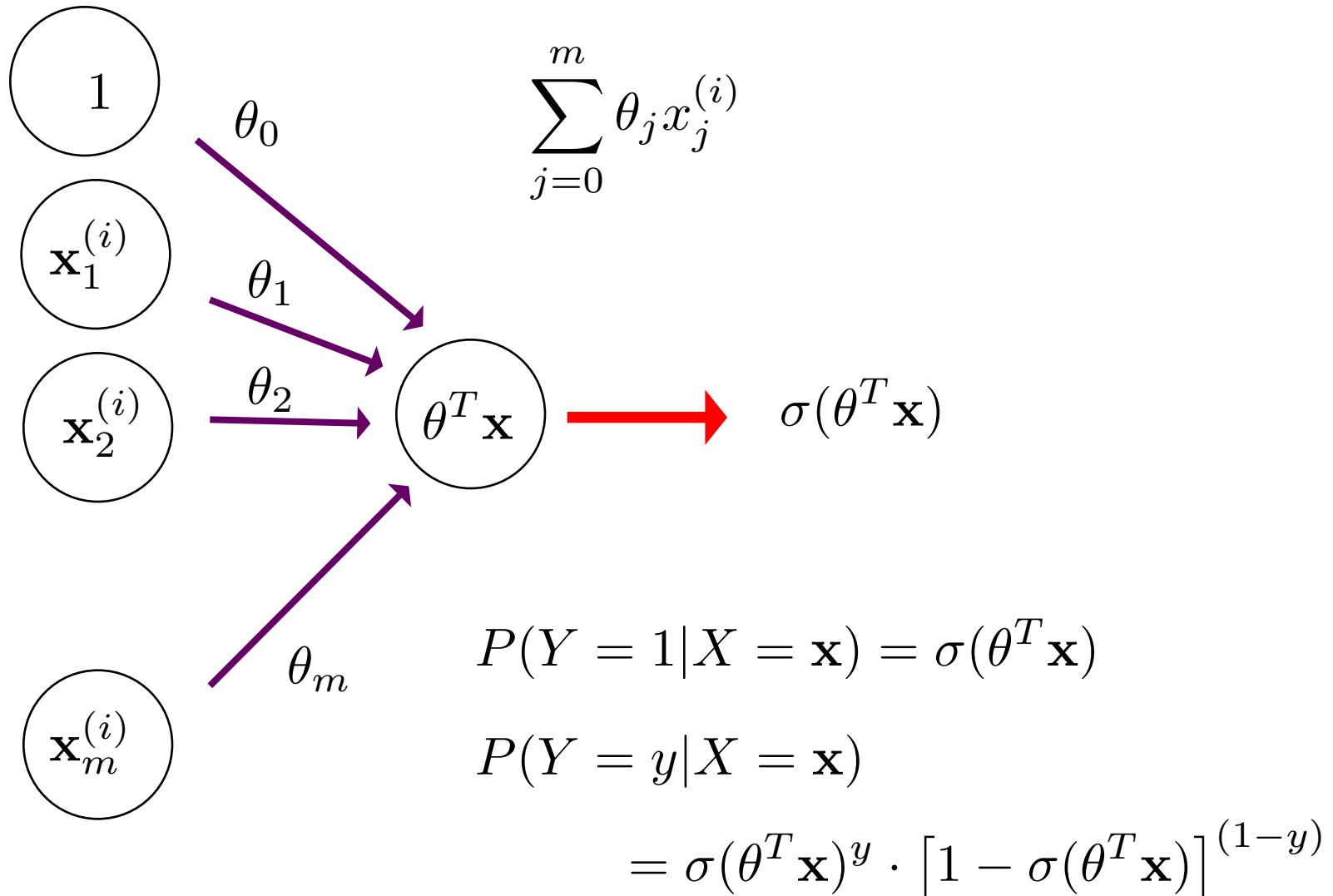
Maximum Likelihood Estimation

Likelihood of Data from a Normal



Remember this?

Logistic Regression



You can calculate likelihood of data, θ

Logistic Regression

1

Make logistic regression assumption

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

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

2

Write the log probability for all data

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

3

Optimize likelihood of data given theta

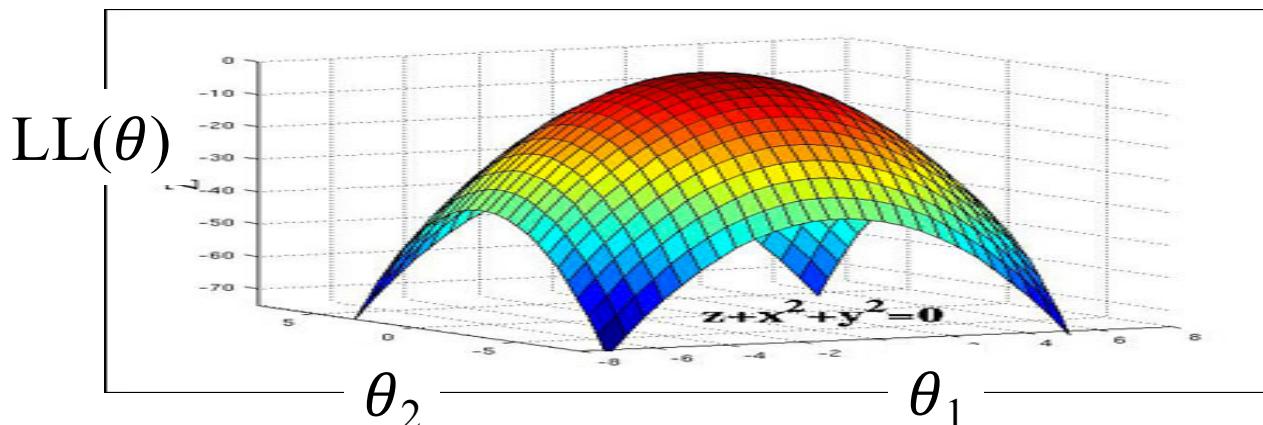
$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=0}^n \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

Gradient Ascent Step

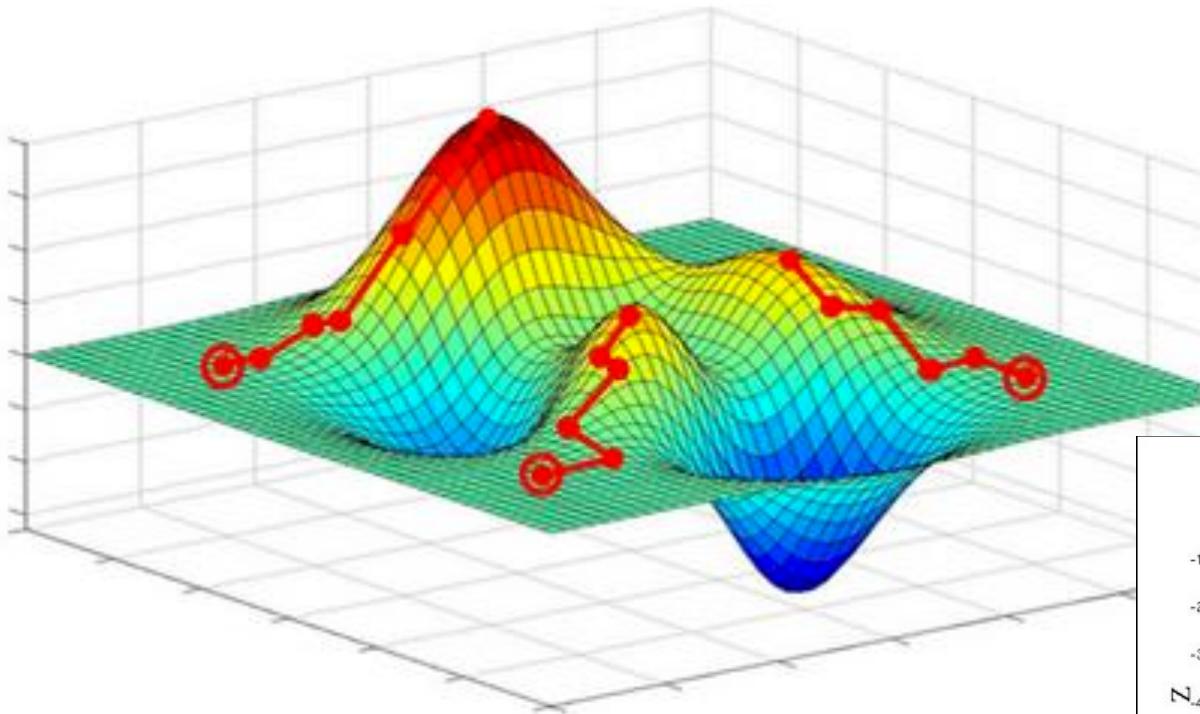
$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=0}^n \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

$$\begin{aligned}\theta_j^{\text{new}} &= \theta_j^{\text{old}} + \eta \cdot \frac{\partial LL(\theta^{\text{old}})}{\partial \theta_j^{\text{old}}} \\ &= \theta_j^{\text{old}} + \eta \cdot \sum_{i=0}^n \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}\end{aligned}$$

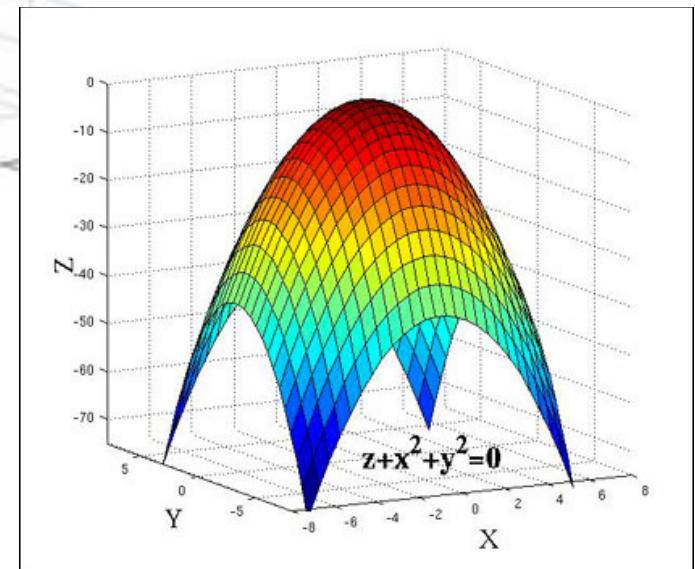
Do this
for all
thetas!



Gradient Ascent



Logistic regression
LL function is convex



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

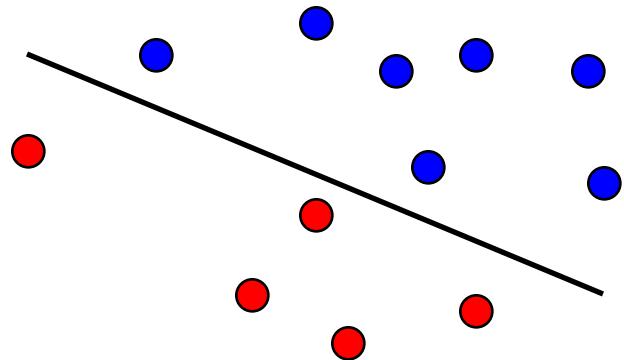
End Review

Logistic Regression Continued

Intuition

Discrimination Intuition

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



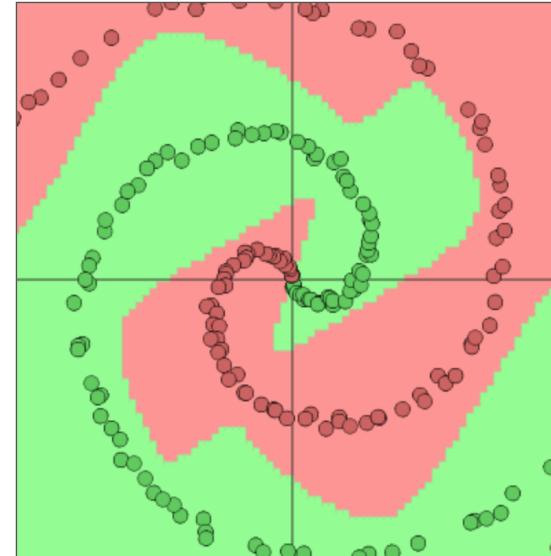
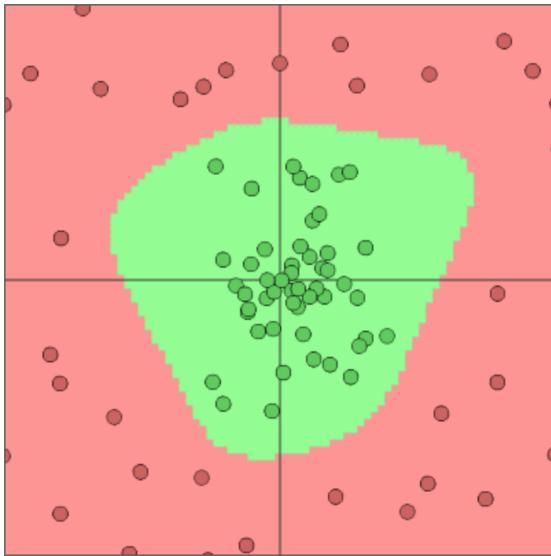
$$\theta^T \mathbf{x} = 0$$

$$\theta_0 + \theta_1 \mathbf{x}_1 + \theta_2 \mathbf{x}_2 = 0$$

- We call such data (or the functions generating the data) "linearly separable"
- Naïve bayes is linear too as there is no interaction between different features.

Some Data Not Linearly Separable

- Some data sets/functions are not separable



- Not possible to draw a line that successfully separates all the $y = 1$ points (green) from the $y = 0$ points (red)
- Despite this fact, logistic regression and Naive Bayes still often work well in practice

Descriptive vs Generative

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

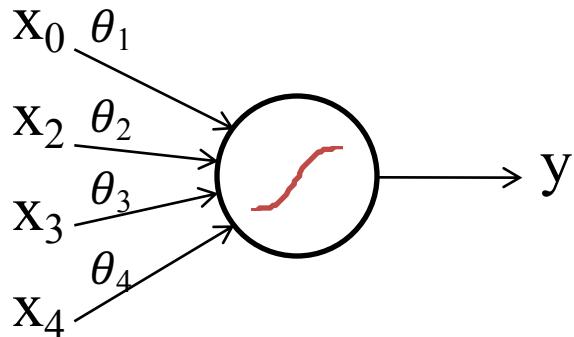
Beyond Discrete Input

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

Neural Networks

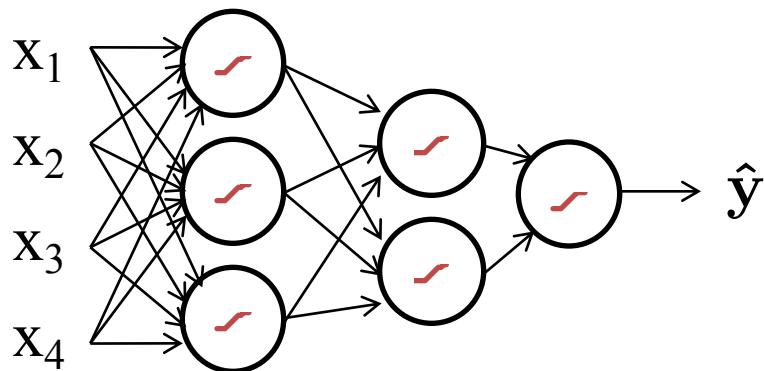
Logistic Regression and Neural Networks

- Consider logistic regression as:



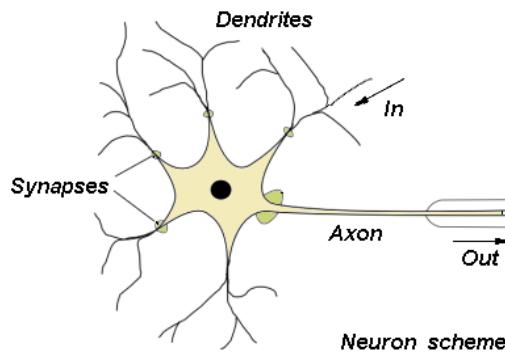
Logistic regression is same as a one node neural network

- Neural network

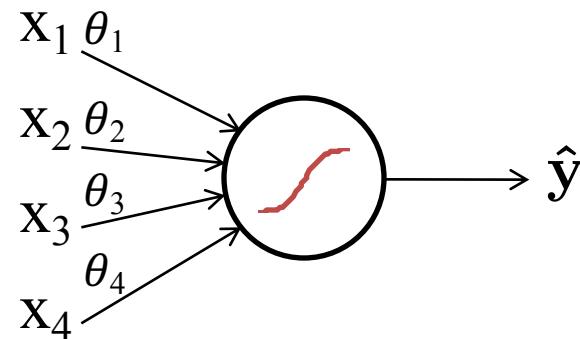


Biological Basis for Neural Networks

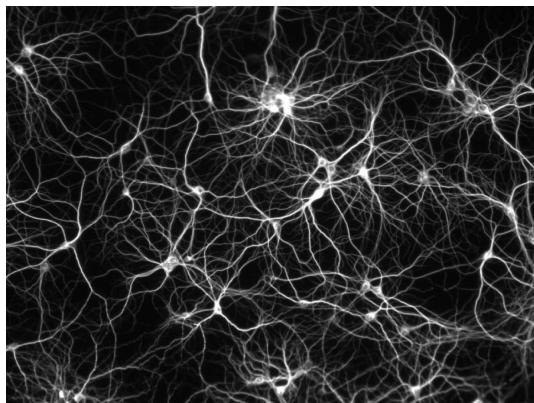
A neuron



Logistic Regression

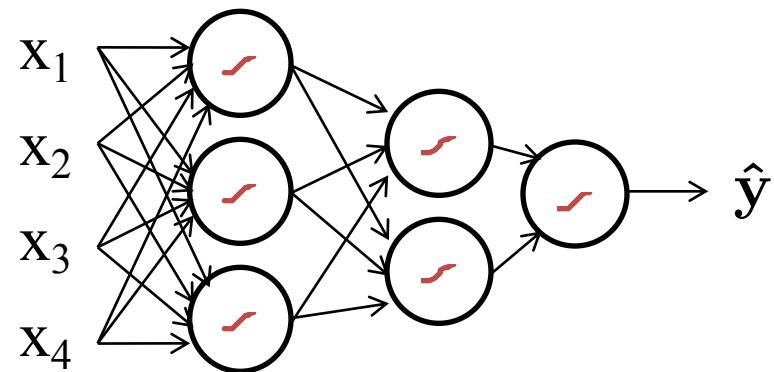


Your brain



Actually, it's probably someone else's brain

Neural Network



Alpha GO



Computer Vision



Revolution in AI

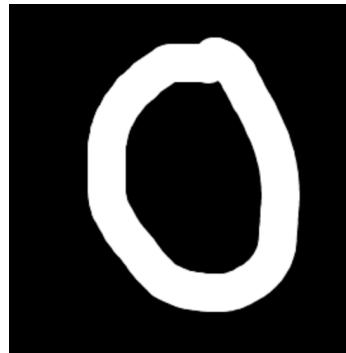


Computers Making Art



Digit Recognition Example

Let's make feature vectors from pictures of numbers

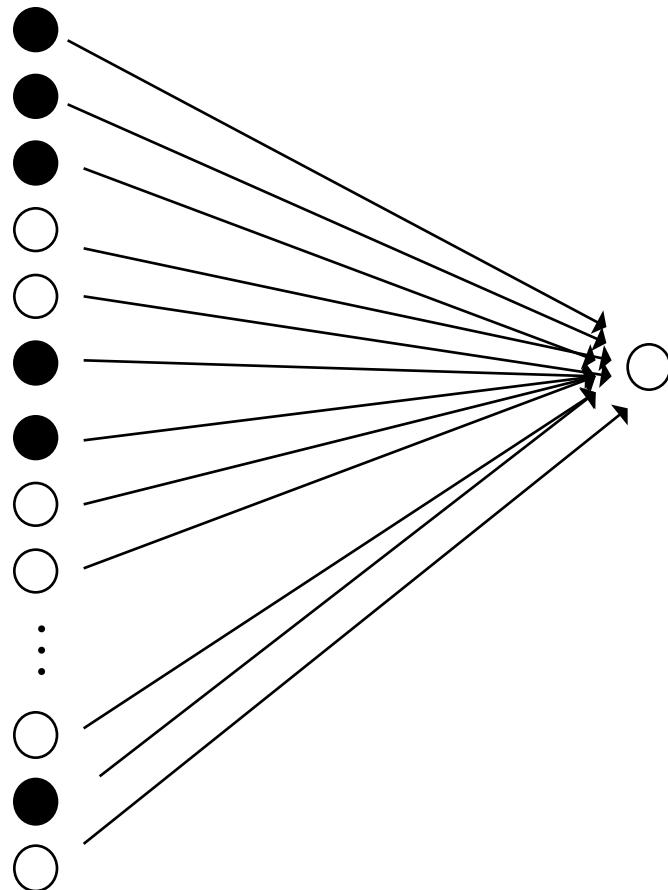
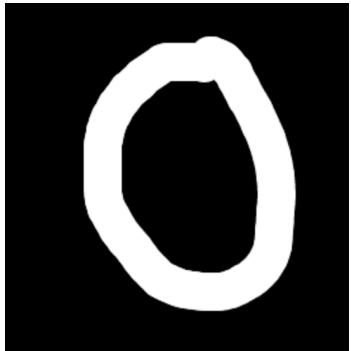


$$\mathbf{x}^{(i)} = [0, 0, 0, 0, \dots, 1, 0, 0, 1, \dots, 0, 0, 1, 0]$$
$$y^{(i)} = 0$$



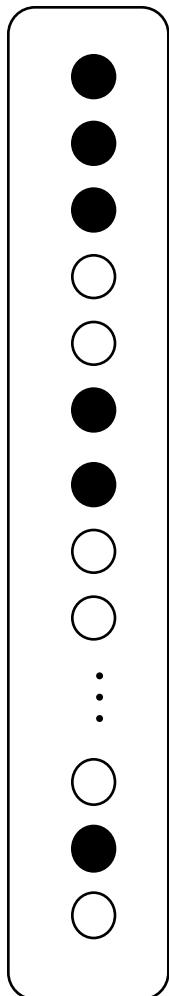
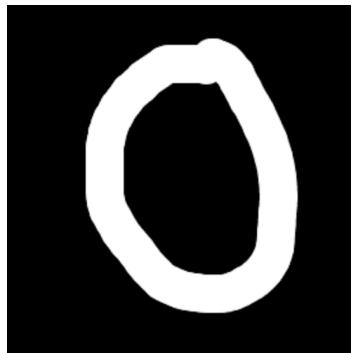
$$\mathbf{x}^{(i)} = [0, 0, 1, 1, \dots, 0, 1, 1, 0, \dots, 0, 1, 0, 0]$$
$$y^{(i)} = 1$$

Logistic Regression

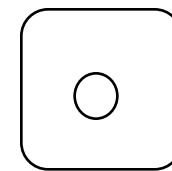


This means it
predicts a 0

Logistic Regression

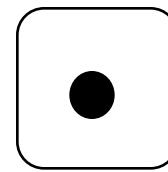
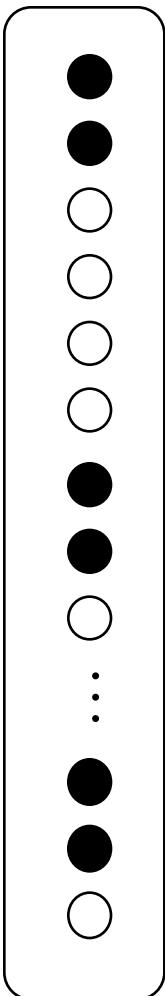


Indicates fully
connected



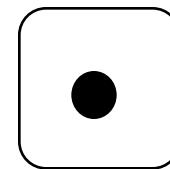
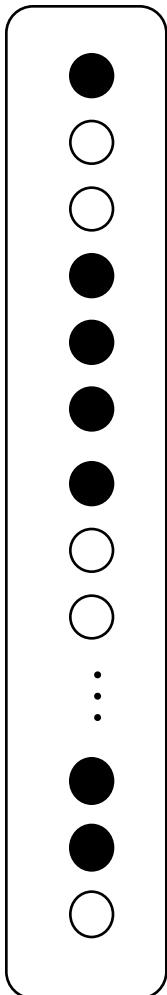
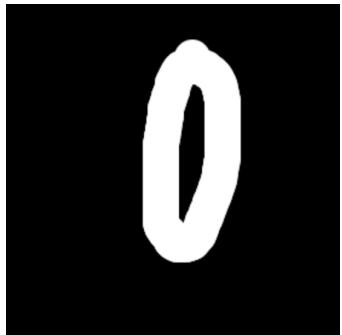
This means it
predicts a 0

Logistic Regression



This means it
predicts a 1

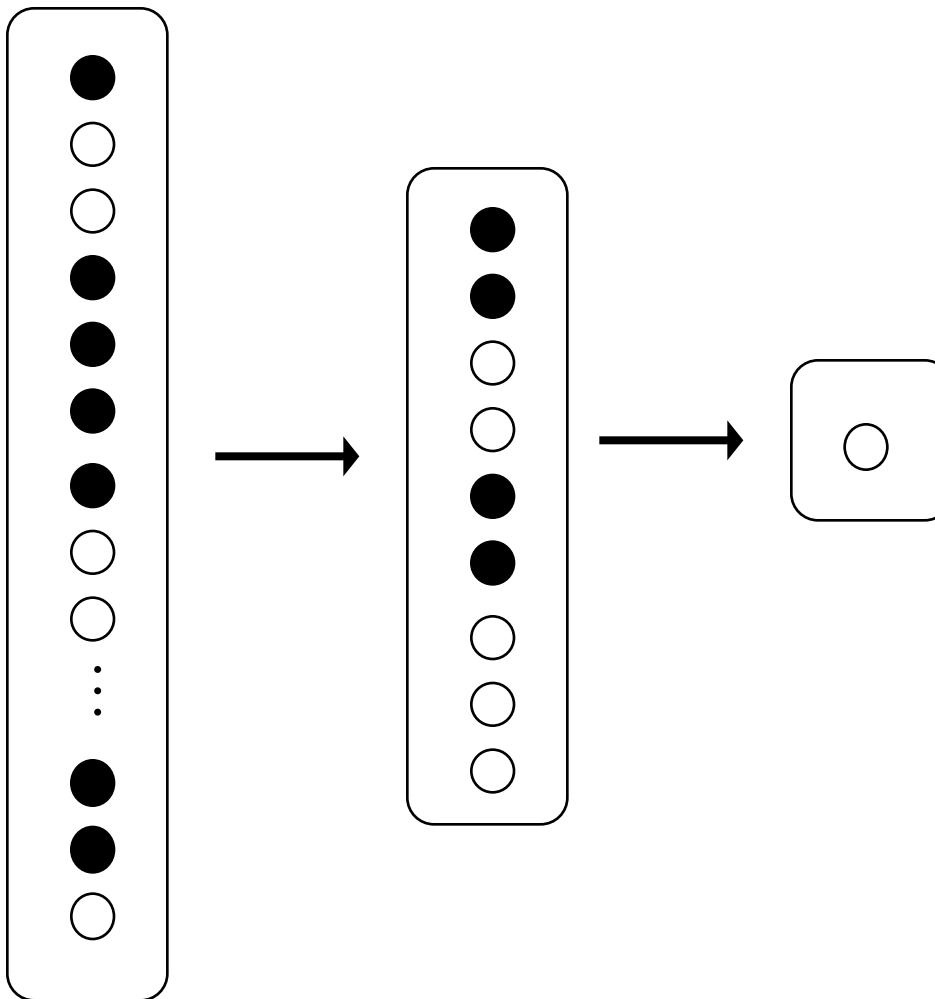
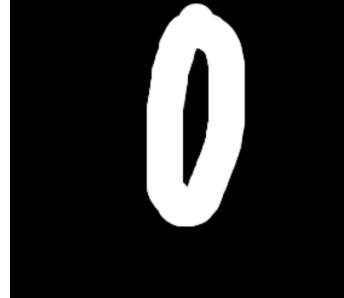
Not So Good



This means it
predicts a 1

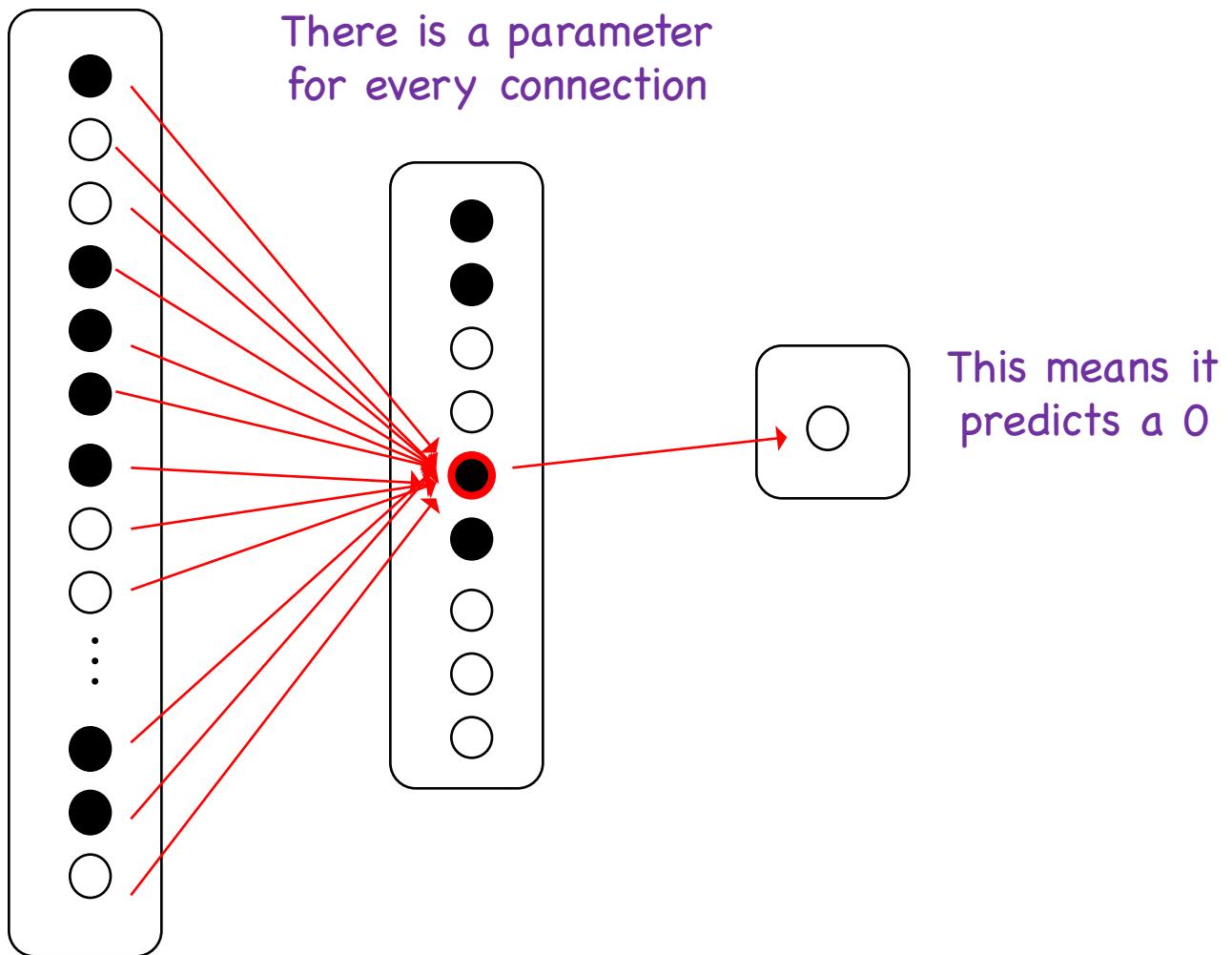
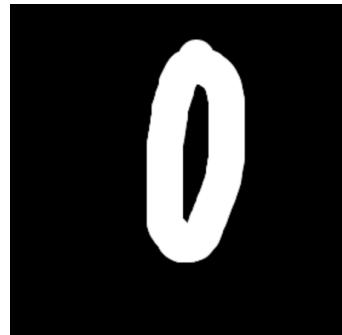
And neither is nearest neighbors

We Can Put Neurons Together



And neither is nearest neighbors

We Can Put Neurons Together



Look at a single “hidden” neuron

Demonstration

Draw your number here



Downsampled drawing: 0

First guess: 0

Second guess: 8

Layer visibility

Input layer

Show

Convolution layer 1

Show

Downsampling layer 1

Show

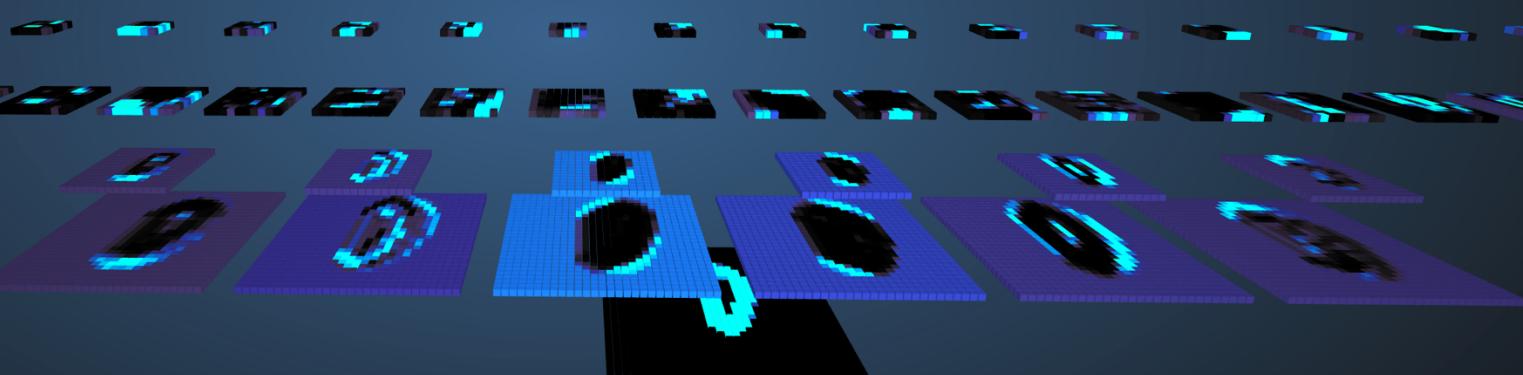
Convolution layer 2

Show

Downsampling layer 2

Show

0 1 2 3 4 5 6 7 8 9



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

How do we train?

MLE of Thetas!

2 Min Pause
Summarize what we have learned

MLE of Thetas!

First: Learning Goals...

1. Understand Chain Rule as ❤ of Deep Learning

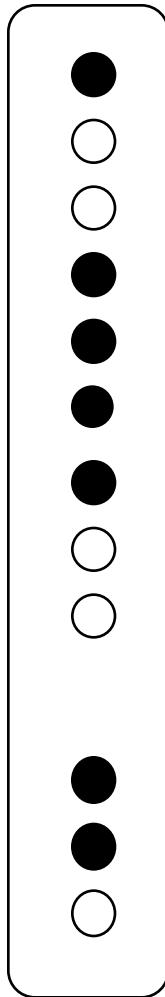
2. Everyone should be able
to do simple derivations

3. Be ready to rock
the socks off of CS229

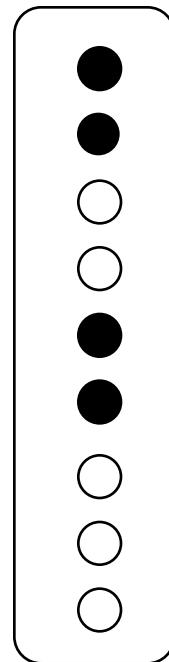
Math worth knowing:

New Notation

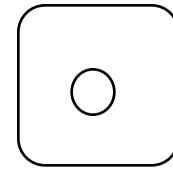
Layer x



Layer h

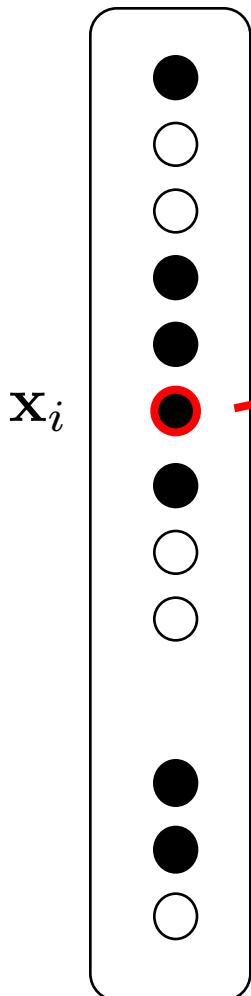


Layer \hat{y}

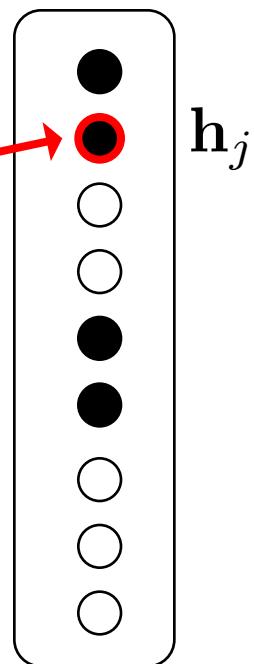


New Notation

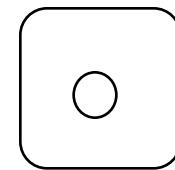
Layer \mathbf{x}



Layer \mathbf{h}



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

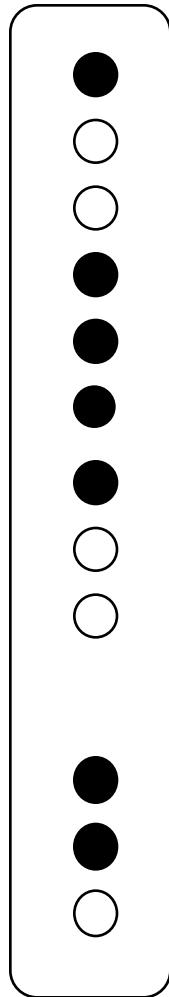


$$\theta_{i,j}^{(h)}$$

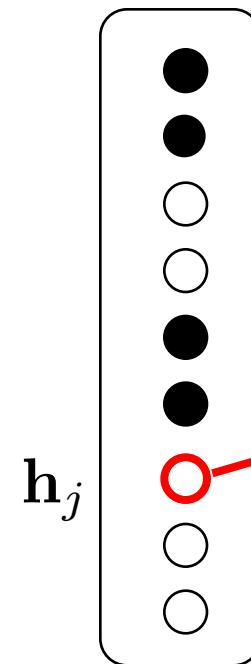
$$\mathbf{h}_j = \sigma \left(\sum_{i=0}^{m_x} \mathbf{x}_i \theta_{i,j}^{(h)} \right)$$

New Notation

Layer \mathbf{x}



Layer \mathbf{h}



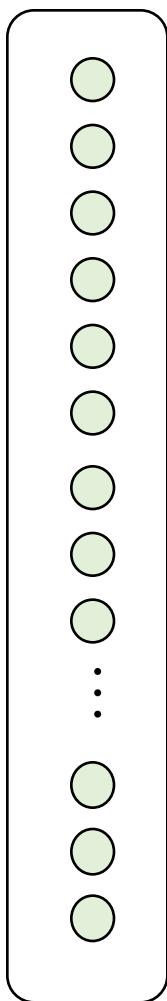
Layer $\hat{\mathbf{y}}$



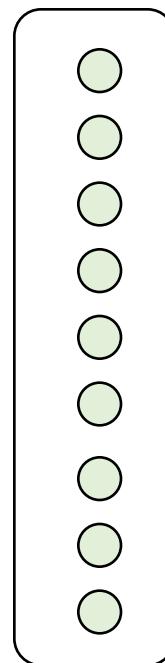
$$\hat{y} = \sigma \left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})} \right)$$

Forward Pass

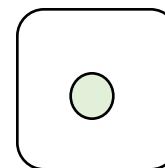
Layer x



Layer h



Layer \hat{y}

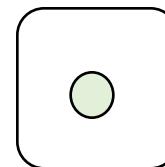
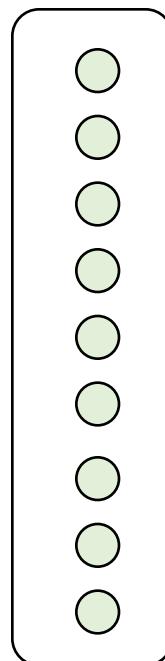
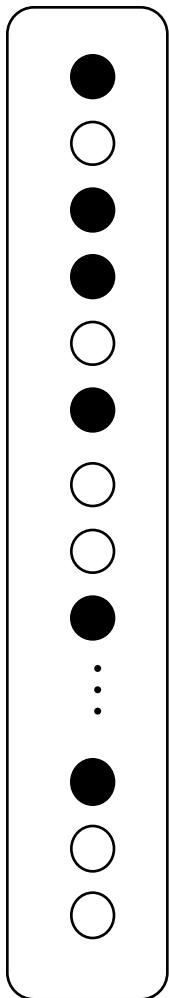


Forward Pass

Layer x

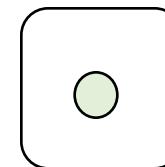
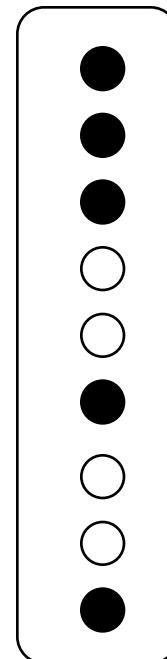
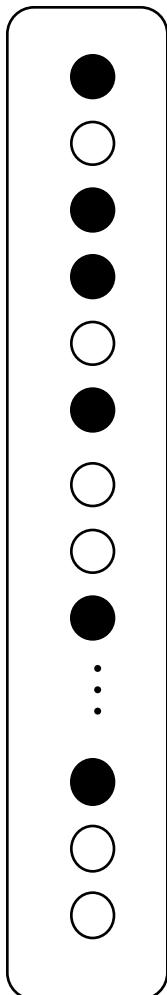
Layer h

Layer \hat{y}



Forward Pass

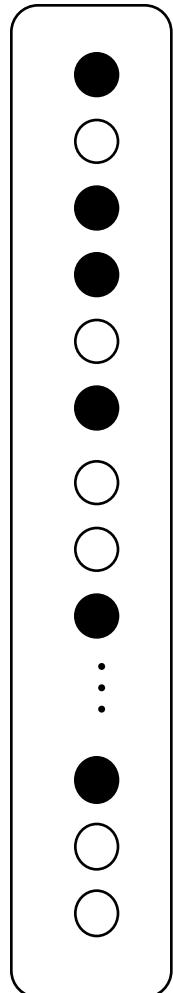
Layer \mathbf{x} Layer \mathbf{h} Layer $\hat{\mathbf{y}}$



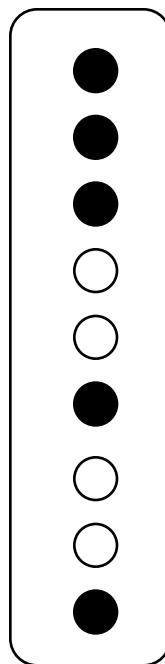
$$\mathbf{h}_j = \sigma \left(\sum_{i=0}^{m_x} \mathbf{x}_i \theta_{i,j}^{(h)} \right)$$

Forward Pass

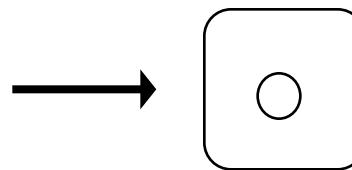
Layer \mathbf{x}



Layer \mathbf{h}



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

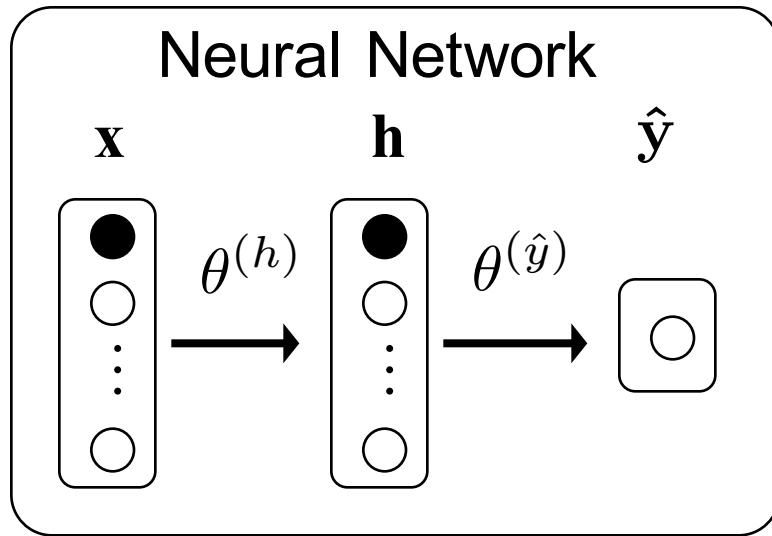


$$\begin{aligned} LL(\theta) = & y \log \hat{y} \\ & + (1 - y) \log [1 - \hat{y}] \end{aligned}$$

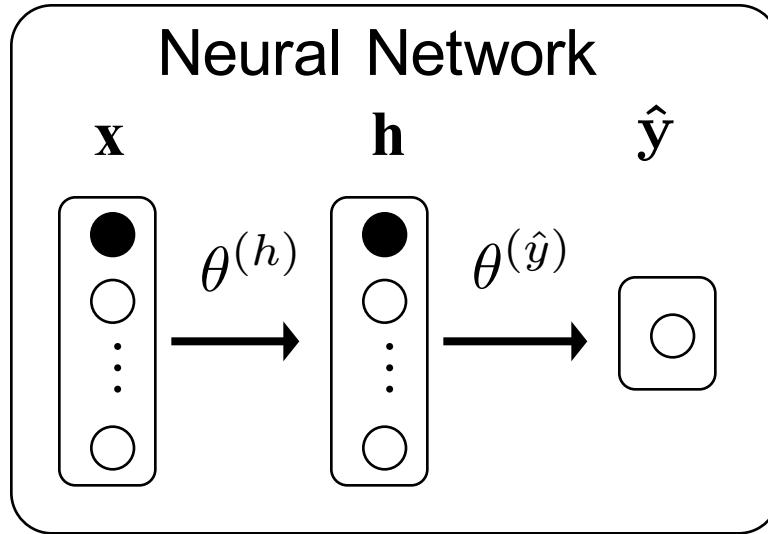
$$\hat{y} = \sigma \left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})} \right)$$

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

All Together



Sanity Check



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

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

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

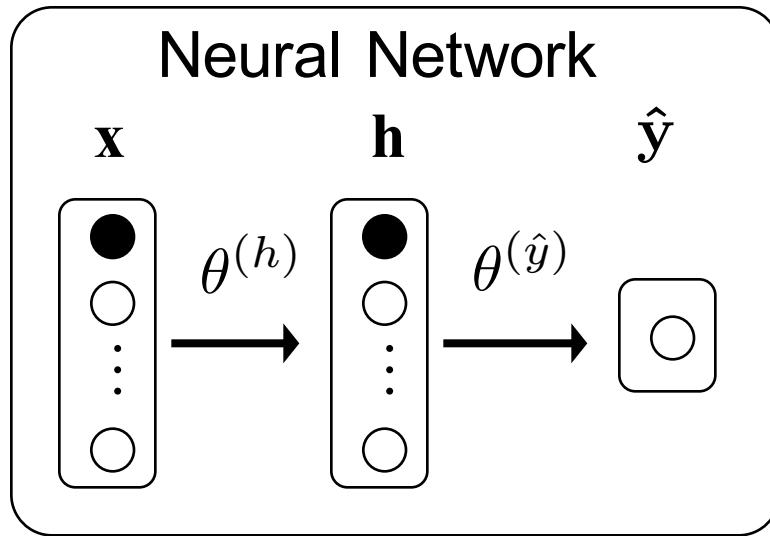
c) 2

a) 20

b) 40

c) 800

Sanity Check 2



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

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

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

c) 2

a) 20

b) 40

c) 800

Today: Do Something Brave



Only Have to Do Three Things

- 1 Make logistic regression assumption
- 2 Calculate the log probability for all data
- 3 Get derivative of log probability with respect to each theta

Same Assumption, Same LL

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

For one datum

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

For IID data

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

Take the log

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

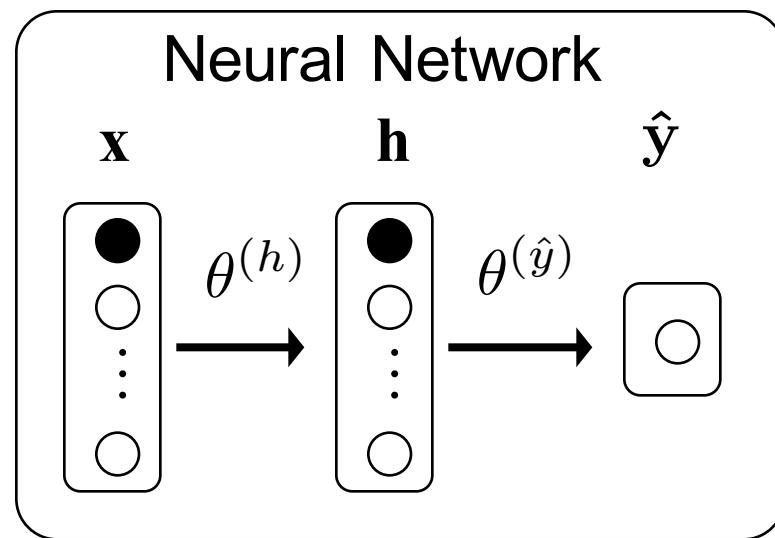
Derivative Goals

Loss with respect to
output layer params

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

Loss with respect to
hidden layer params

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



Think About Only One Training Instance

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

We only need to calculate the gradient for one training example!

$$\frac{\partial}{\partial x} \sum f(x) = \sum \frac{\partial}{\partial x} f(x)$$

We will pretend we only have one example

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

We can sum up the gradients of each example to get the correct answer

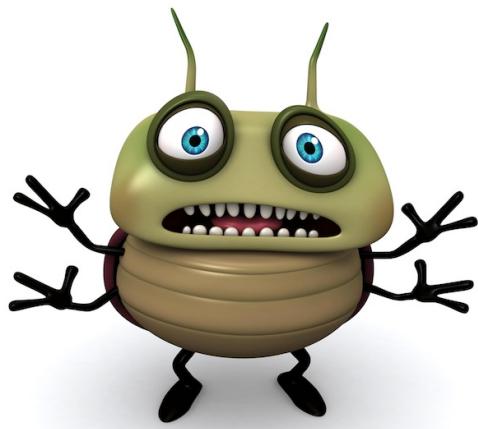
Bad Approach

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

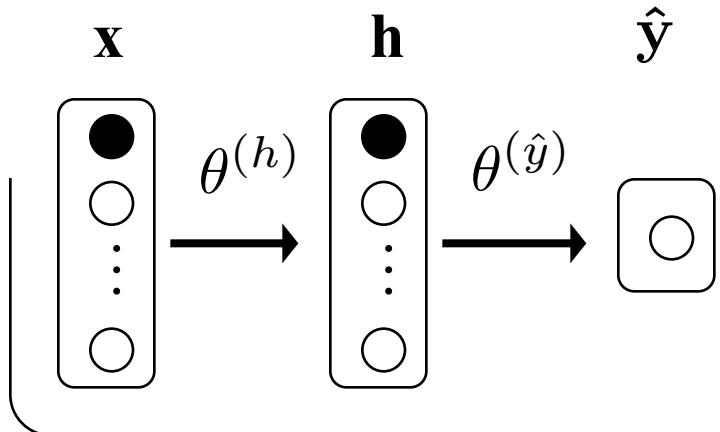
$$\hat{y} = \sigma \left(\sum_{i=0}^{m_h} h_i \theta_i^{(\hat{y})} \right)$$

Math bug

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



Neural Network



Big Idea

Big Idea

It's called chain rule

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

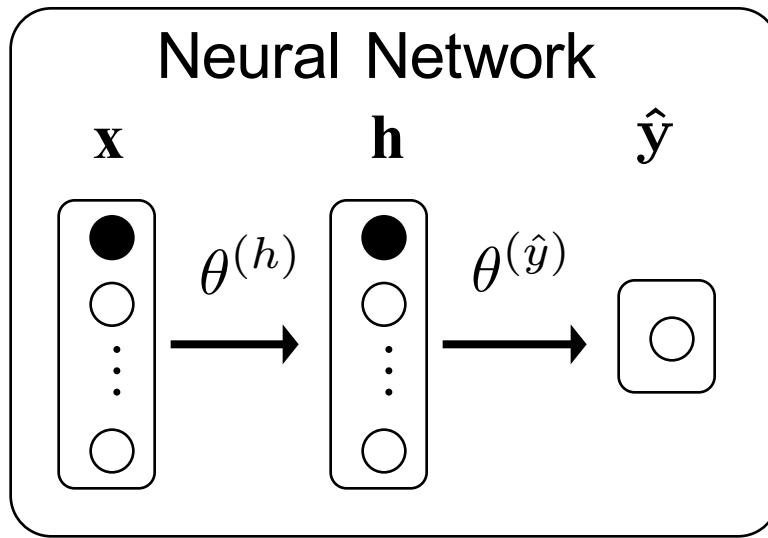
First use:

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

Chain Rule Example 1

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

Goal



Network

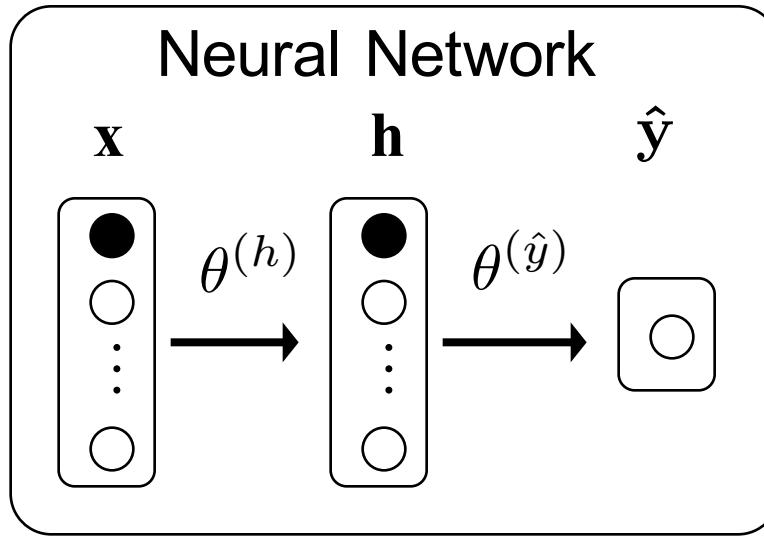
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}}$$

Decomposition

Chain Rule Example 2

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

Goal



Network

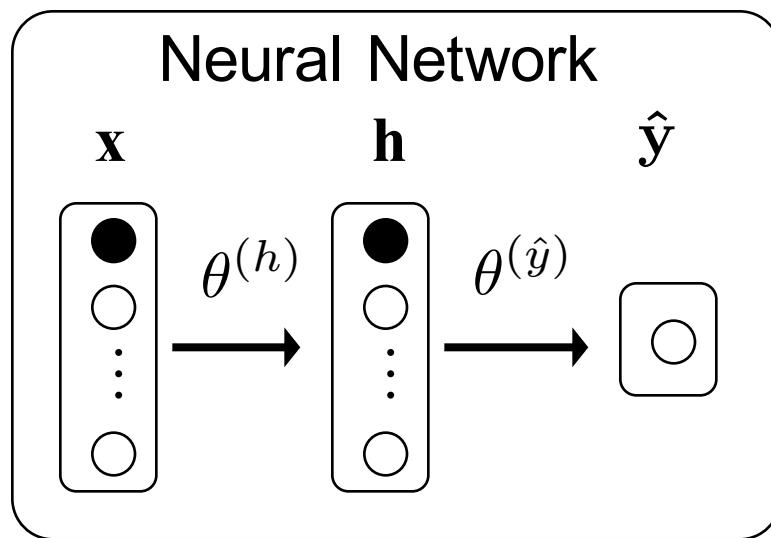
$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{h}_j} \cdot \frac{\partial \mathbf{h}_j}{\partial \theta_{i,j}^{(h)}}$$

Decomposition

Decomposition

Gradient of output layer params

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



Gradient of output layer params

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

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

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

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

Recall: Sigmoid has a Beautiful Slope

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

*True fact about
sigmoid functions*

Sigmoid, you should be a ski hill

Gradient of output layer params

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

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

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

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

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

What! That's not scary!

Make it Simple

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



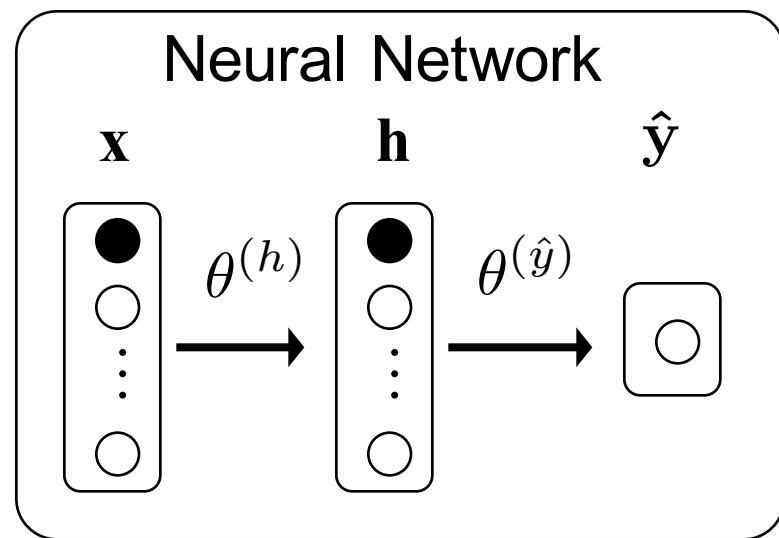
$$= \frac{y}{\hat{y}} - \frac{(1-y)}{(1-\hat{y})}$$



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

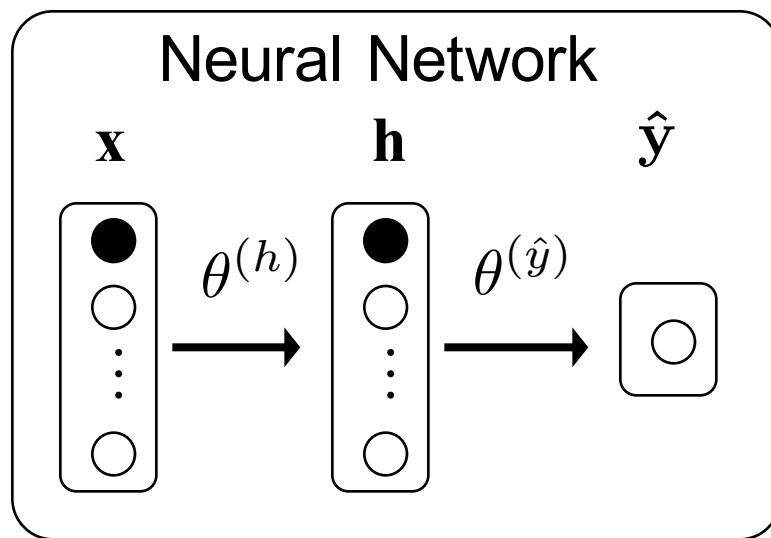
Boom!

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



Gradient of hidden layer params

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



Gradient of hidden layer params

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

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

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

Wait is it over?

Gradient of hidden layer params

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

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

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

That one too?

Make it Simple

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



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


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


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

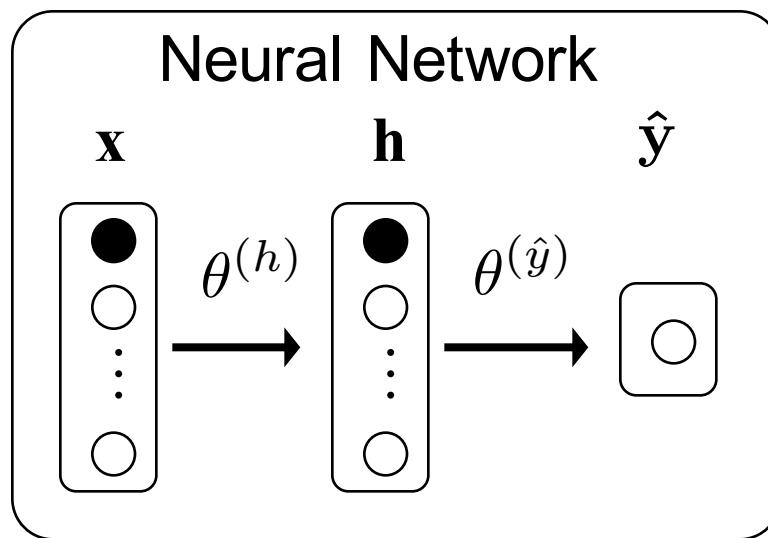
Summary: Simple Calculations For

Loss with respect to
output layer params

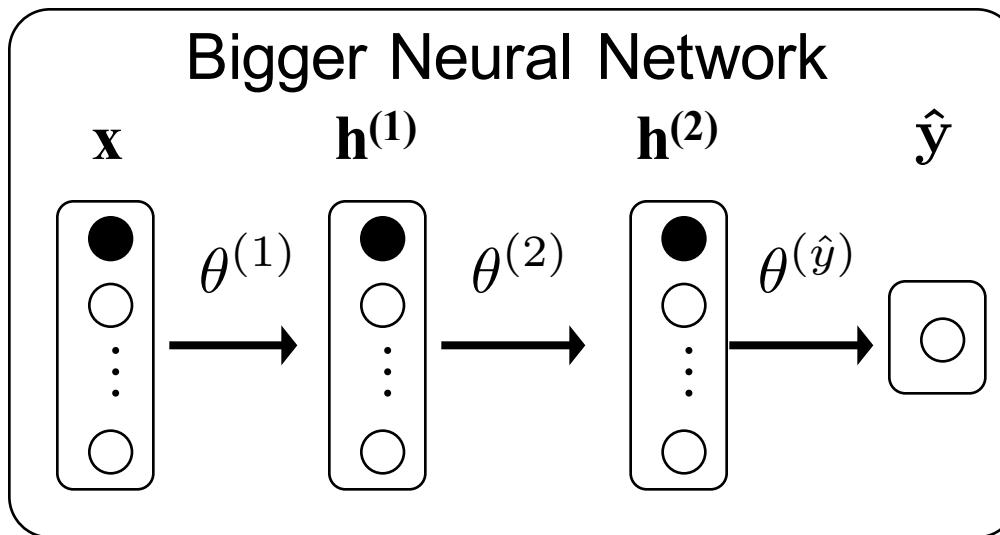
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}}$$

Loss with respect to
hidden layer params

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



What Would You Do Here?

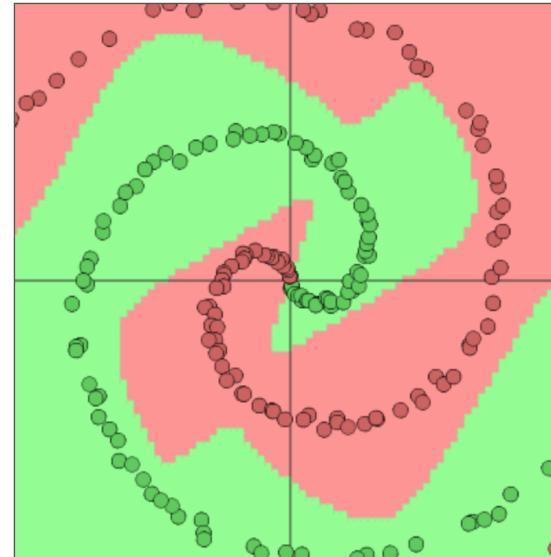
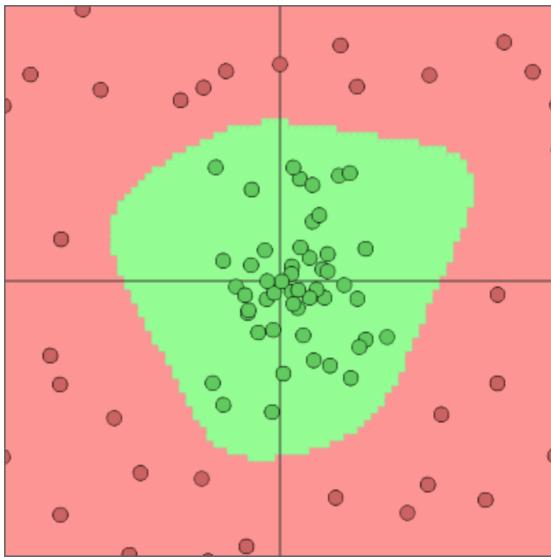


Chain rule:
Game changer for
artificial intelligence

Congrats. You now know
backpropagation

Neural Networks Can Learn Complex Functions

- Some data sets/functions are not separable



- These are classifiers learned by neural networks

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