10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Neural Networks and Backpropagation

Neural Net Readings:
Murphy --
Bishop 5
HTF 11
Mitchell 4

Matt Gormley
Lecture 19
March 29, 2017

Reminders

- Homework 6: Unsupervised Learning
  - Release: Wed, Mar. 22
  - Due: Mon, Apr. 03 at 11:59pm
- Homework 5 (Part II): Peer Review
  - Release: Wed, Mar. 29
  - Due: Wed, Apr. 05 at 11:59pm
- Peer Tutoring

Expectation: You should spend at most 1 hour on your reviews

Neural Networks Outline

- **Logistic Regression (Recap)**
  – Data, Model, Learning, Prediction
- **Neural Networks**
  – A Recipe for Machine Learning
  – Visual Notation for Neural Networks
  – Example: Logistic Regression Output Surface
  – 2-Layer Neural Network
  – 3-Layer Neural Network
- **Neural Net Architectures**
  – Objective Functions
  – Activation Functions
- **Backpropagation**
  – Basic Chain Rule (of calculus)
  – Chain Rule for Arbitrary Computation Graph
  – Backpropagation Algorithm
  – Module-based Automatic Differentiation (Autodiff)

RECALL: LOGISTIC REGRESSION

Using gradient ascent for linear classifiers

Recall…

Key idea behind today's lecture:

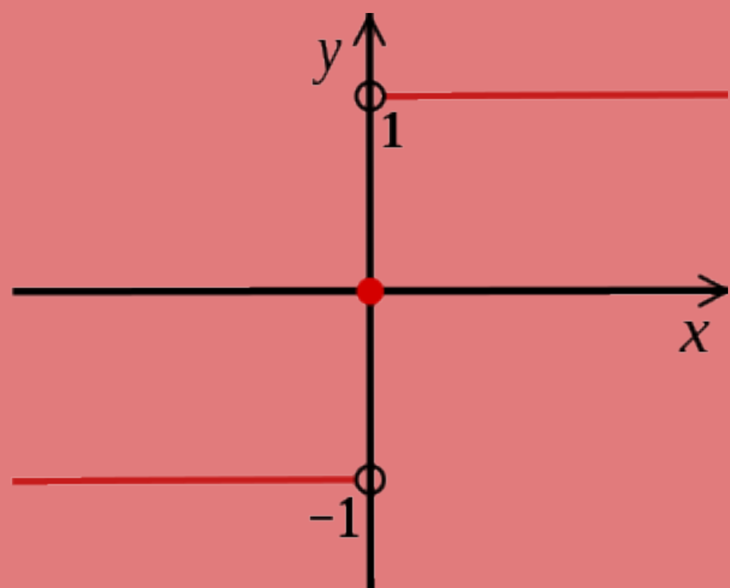
1. Define a linear classifier (logistic regression)
2. Define an objective function (likelihood)
3. Optimize it with gradient descent to learn parameters
4. Predict the class with highest probability under the model

Using gradient ascent for line… classifiers

Recall…

This decision function isn't differentiable:

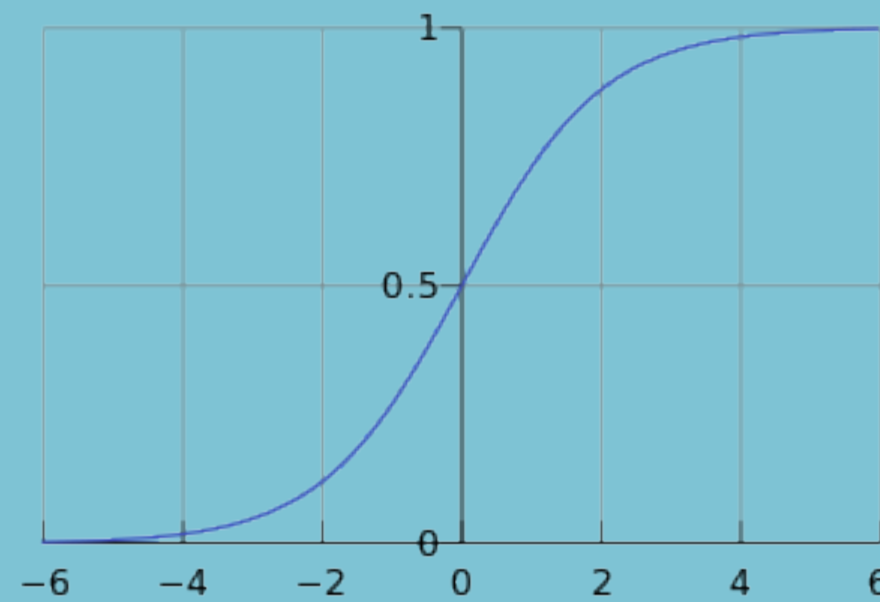
$$h(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$$



sign(x)

Use a differentiable function instead:

$$p_{\boldsymbol{\theta}}(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \mathbf{x})}$$



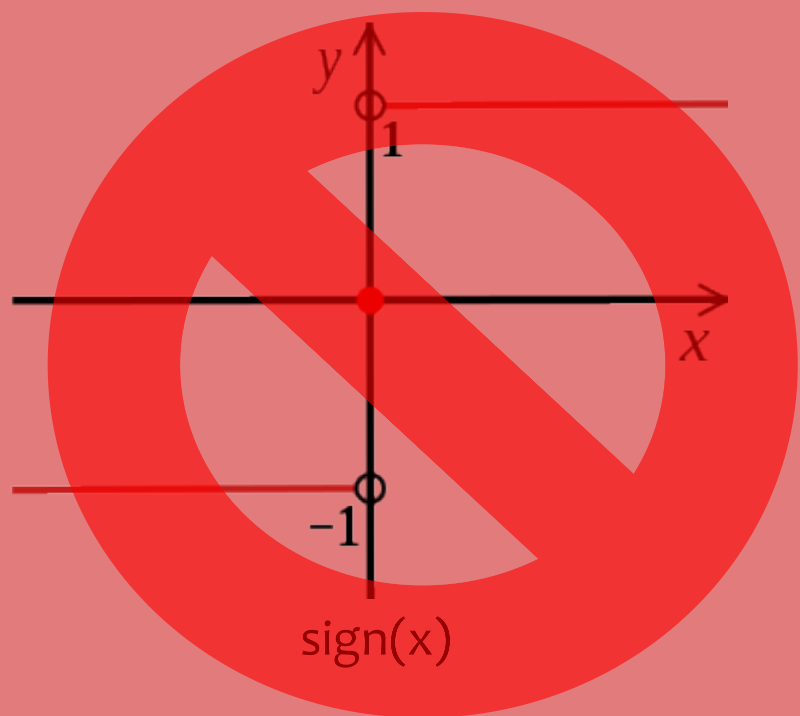
$$\text{logistic}(u) \equiv \frac{1}{1 + e^{-u}}$$

Using gradient ascent for linear classifiers

Recall…

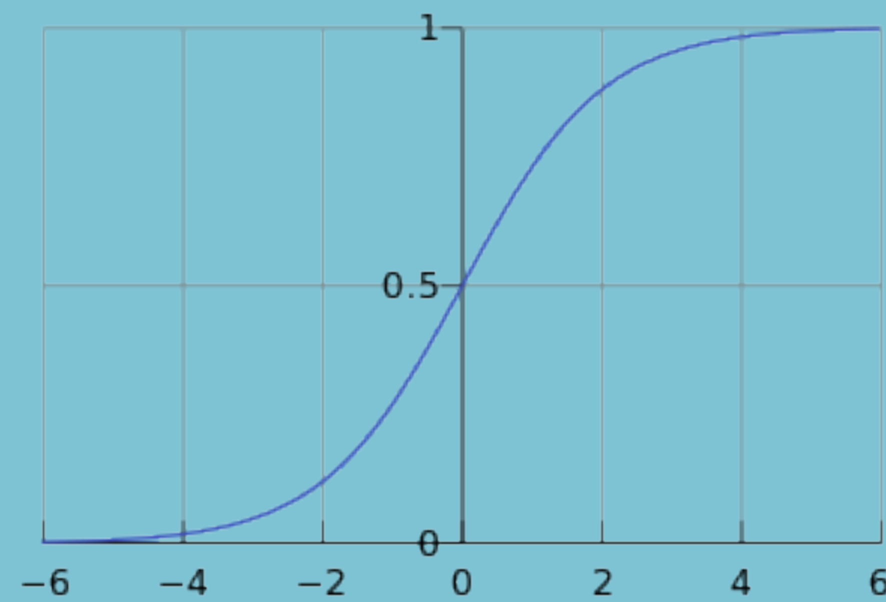
This decision function isn't differentiable:

$$h(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$$



Use a differentiable function instead:

$$p_{\boldsymbol{\theta}}(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \mathbf{x})}$$



$$\text{logistic}(u) \equiv \frac{1}{1 + e^{-u}}$$

Recall...

Logistic Regression

Data: Inputs are continuous vectors of length K. Outputs are discrete.

$$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N} \text{ where } \mathbf{x} \in \mathbb{R}^K \text{ and } y \in \{0, 1\}$$

Model: Logistic function applied to dot product of parameters with input vector.

$$p_{\boldsymbol{\theta}}(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \mathbf{x})}$$

Learning: finds the parameters that minimize some objective function. $\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta})$

Prediction: Output is the most probable class.

$$\hat{y} = \underset{y \in \{0,1\}}{\operatorname{argmax}}\, p_{\boldsymbol{\theta}}(y|\mathbf{x})$$

NEURAL NETWORKS

Background

A Recipe for Machine Learning

1. Given training data:

$$\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$$

Face

Face

Not a face

2. Choose each of these:
   – Decision function

$$\hat{\boldsymbol{y}} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

   – Loss function

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$$

Examples: Linear regression, Logistic regression, Neural Network

Examples: Mean-squared error, Cross Entropy

Background

A Recipe for Machine Learning

1. Given training data:

$$\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$$

2. Choose each of these:
   - Decision function

$$\hat{\boldsymbol{y}} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

   - Loss function

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$$

3. Define goal:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^N \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

4. Train with SGD:
(take small steps opposite the gradient)

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

Background

A Recipe for

Gradients

Backpropagation can compute this gradient!

And it's a **special case of a more general algorithm** called reverse-mode automatic differentiation that can compute the gradient of any differentiable function efficiently!

1. Given training dat

$\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$

2. Choose each of th

– Decision functio

$\hat{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$

– Loss function

$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$

$\boldsymbol{y}_i)$

opposite the gradient)

$\boldsymbol{\theta}^{(t} \quad {}^{(t)} - \eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$

Goals for Today's Lecture

1. Explore a **new class of decision functions** (Neural Networks)
2. Consider **variants of this recipe** for training

A Recipe for

2. Choose each of these:
   - Decision function

$$\hat{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

   - Loss function

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$$

4. Train with SGD:

(take small steps opposite the gradient)

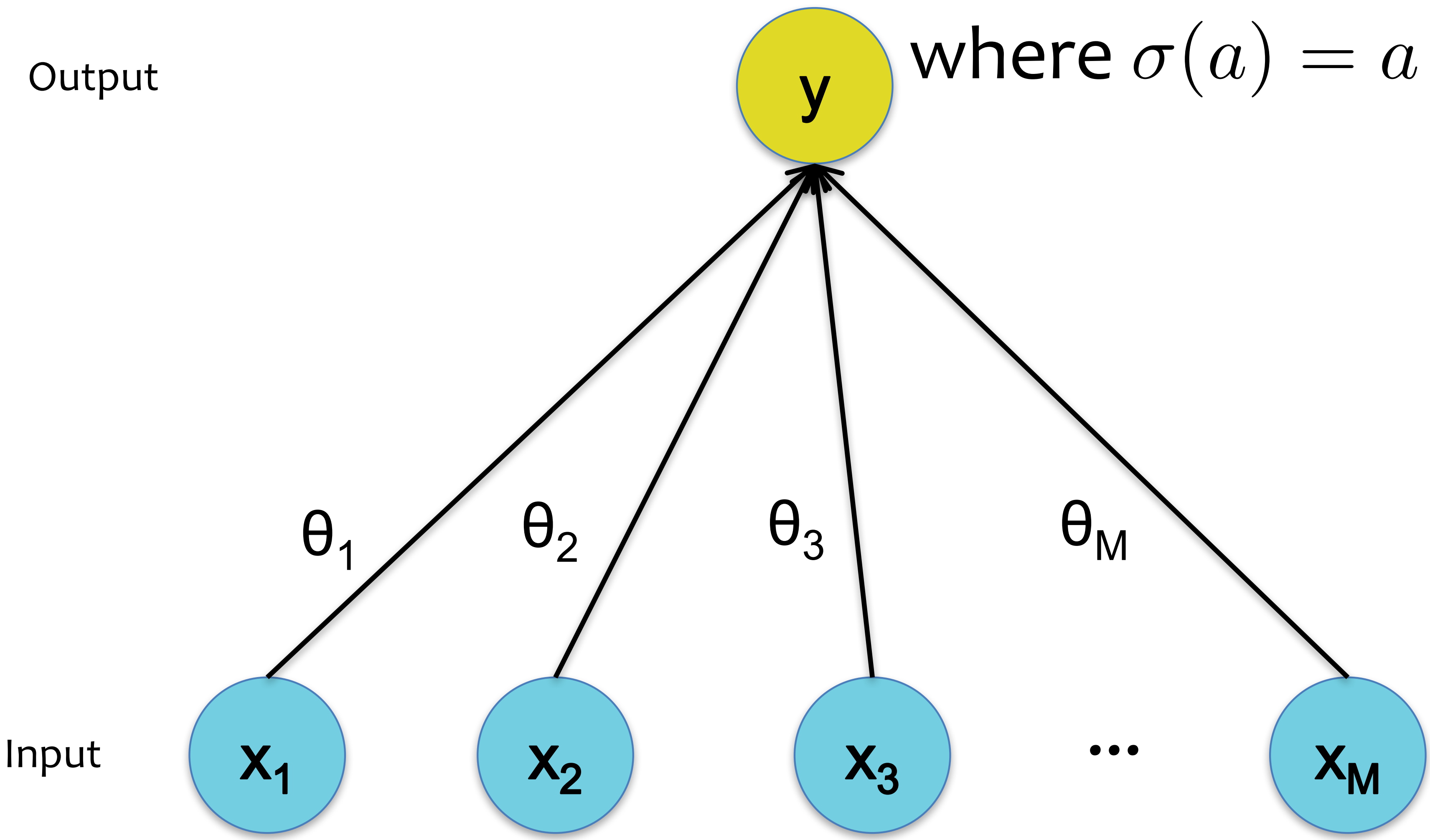
$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

Decision Functions

Linear Regression

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

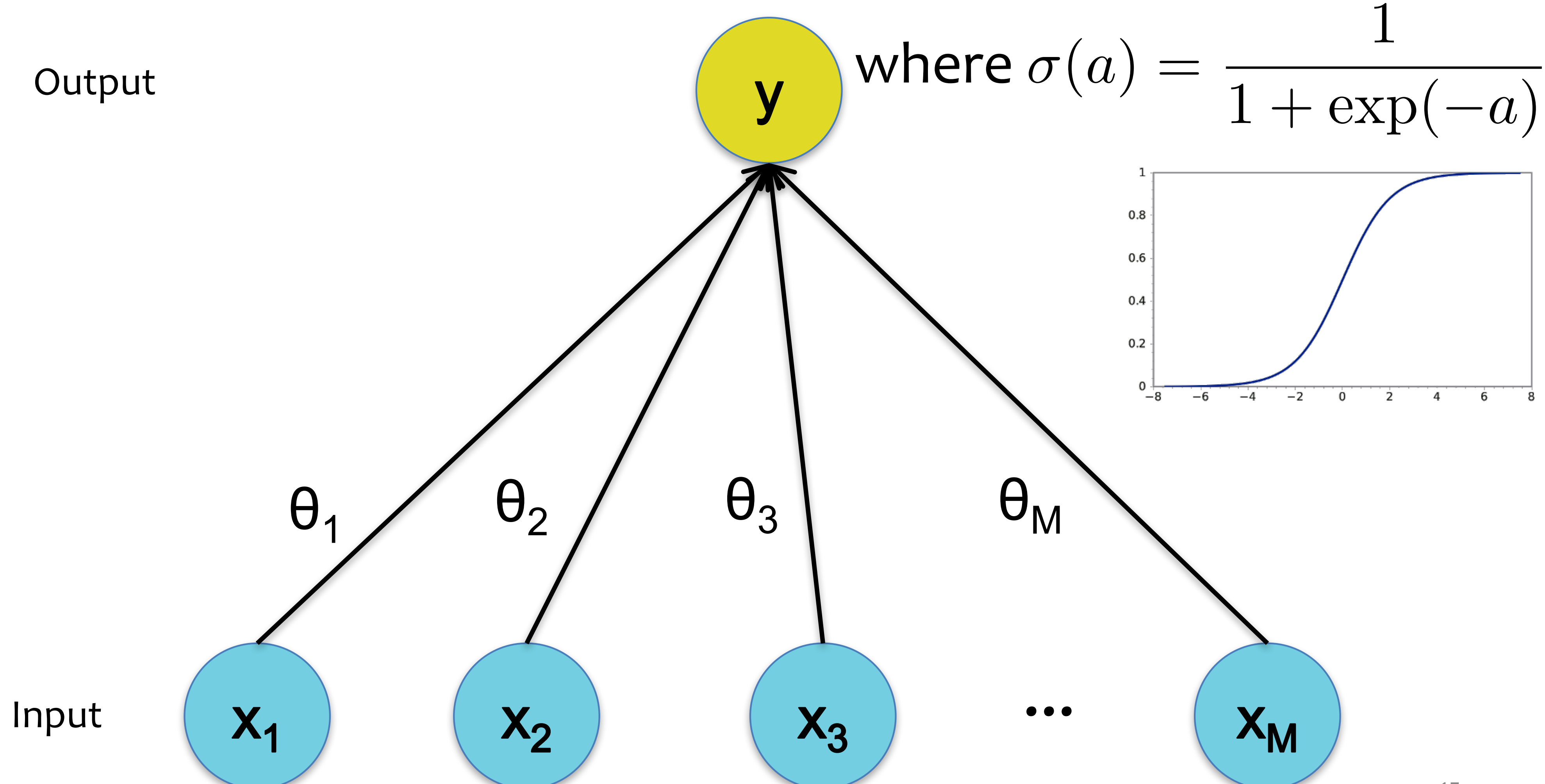
where $\sigma(a) = a$



Logistic Regression

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

where $\sigma(a) = \dfrac{1}{1 + \exp(-a)}$

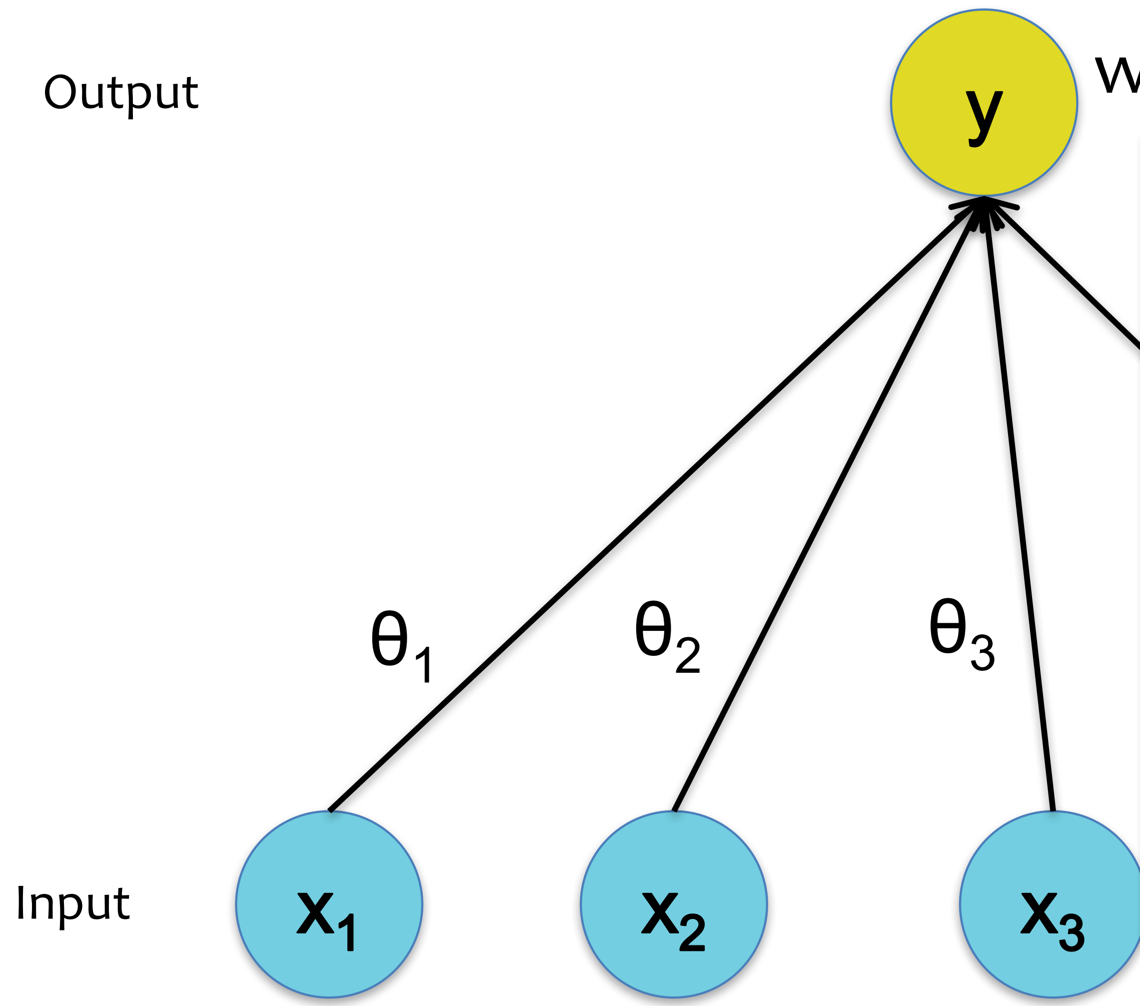


Decision Functions

Logistic Regression

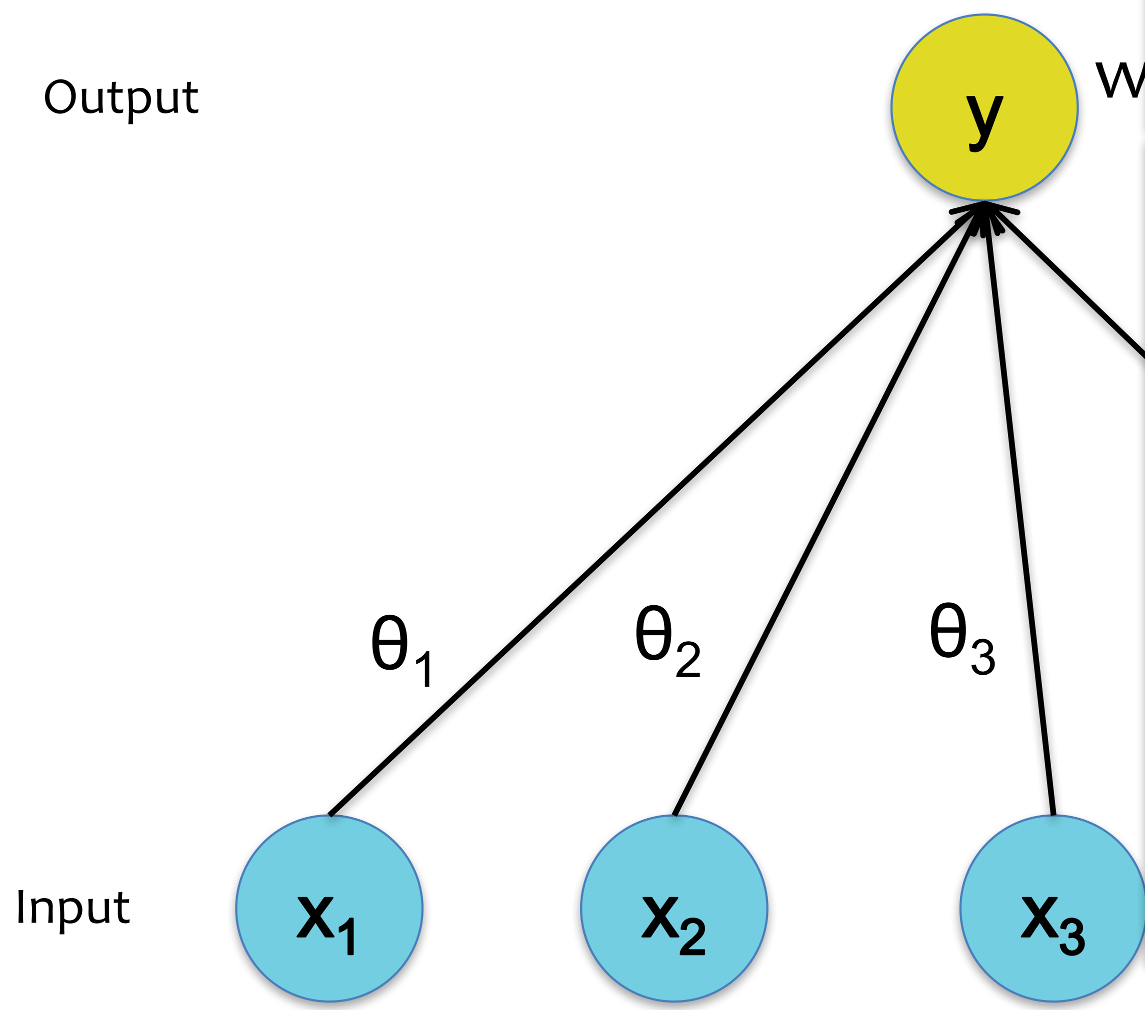
$$y = h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(\boldsymbol{\theta}^T \boldsymbol{x})$$

$$\text{where } \sigma(a) = \frac{1}{1 + \exp(-a)}$$

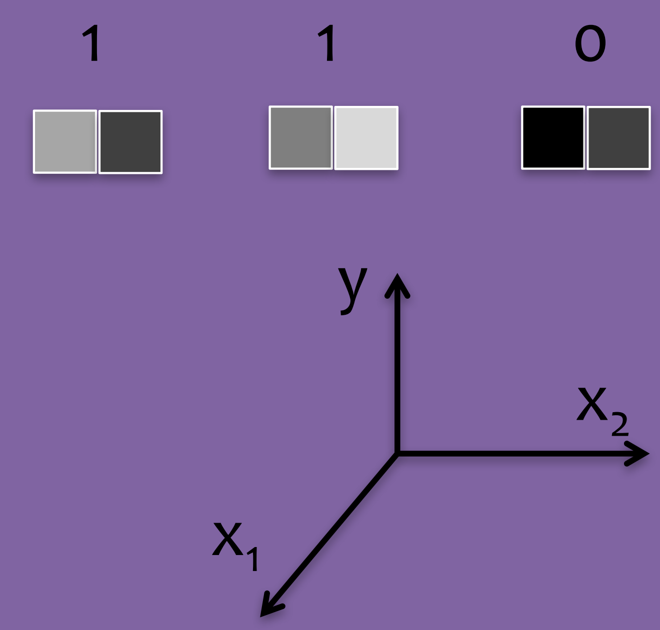


Logistic Regression

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



In-Class Example

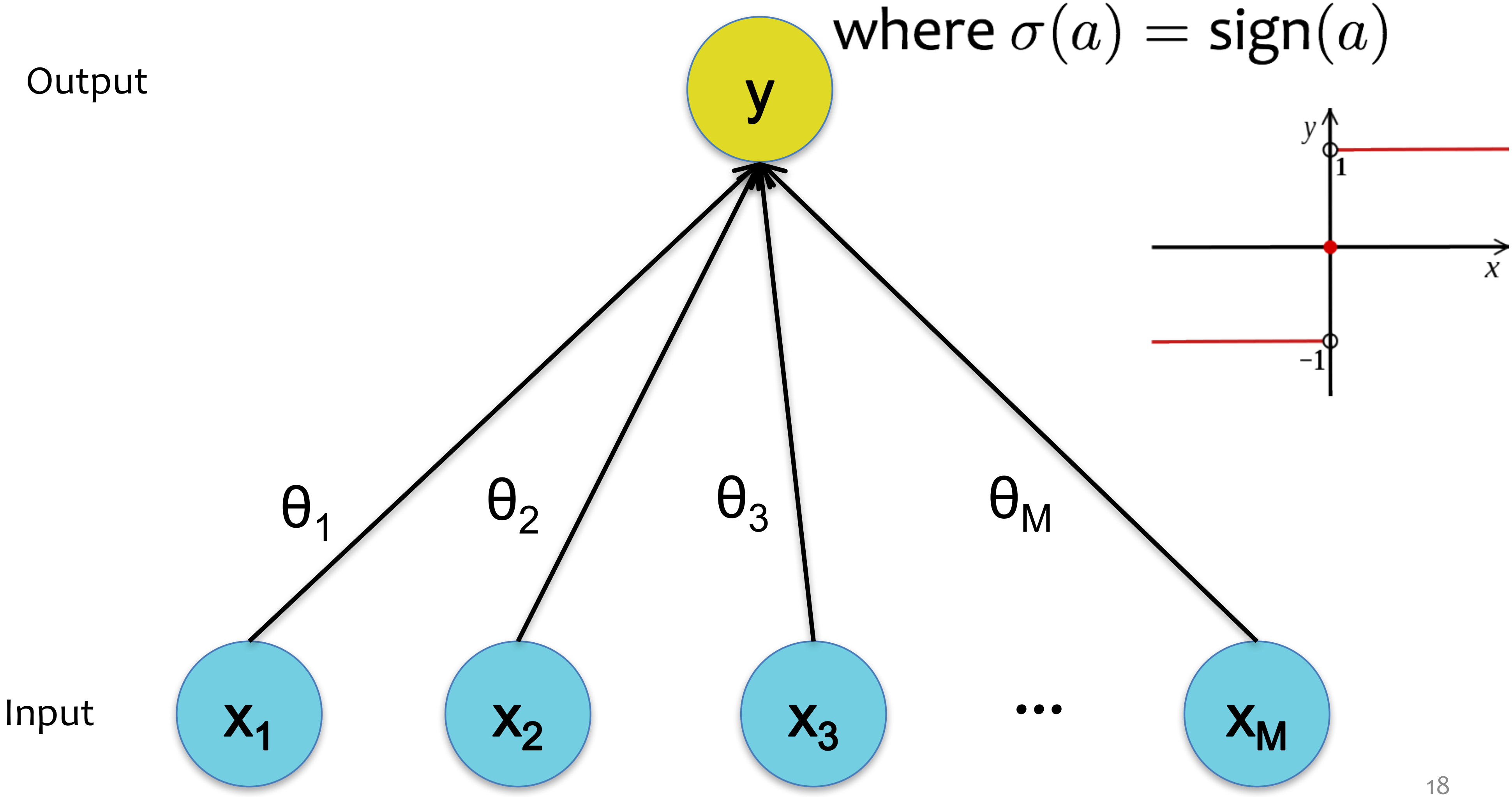


Decision Functions

Perceptron

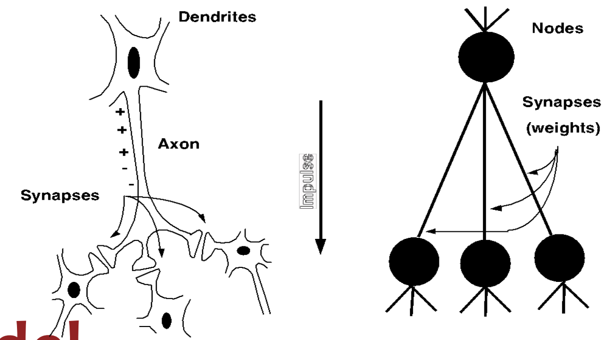
$$y = h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(\boldsymbol{\theta}^T \boldsymbol{x})$$

where $\sigma(a) = \text{sign}(a)$



From Biological to Artificial

The motivation for Artificial Neural Networks comes from biology…



Biological "Model"

- **Neuron:** an excitable cell
- **Synapse:** connection between neurons
- A *neuron* sends an **electrochemical pulse** along its *synapses* when a sufficient voltage change occurs
- **Biological Neural Network:** collection of neurons along some pathway through the brain

Artificial Model

- **Neuron:** node in a directed acyclic graph (DAG)
- **Weight:** multiplier on each edge
- **Activation Function:** nonlinear thresholding function, which allows a neuron to "fire" when the input value is sufficiently high
- **Artificial Neural Network:** collection of neurons into a DAG, which define some differentiable function

Biological "Computation"

- Neuron switching time : ~ 0.001 sec
- Number of neurons: ~ 10^{10}
- Connections per neuron: ~ $10^{4\text{-}5}$
- Scene recognition time: ~ 0.1 sec

Artificial Computation

- Many neuron-like threshold switching units
- Many weighted interconnections among units
- Highly parallel, distributed processes

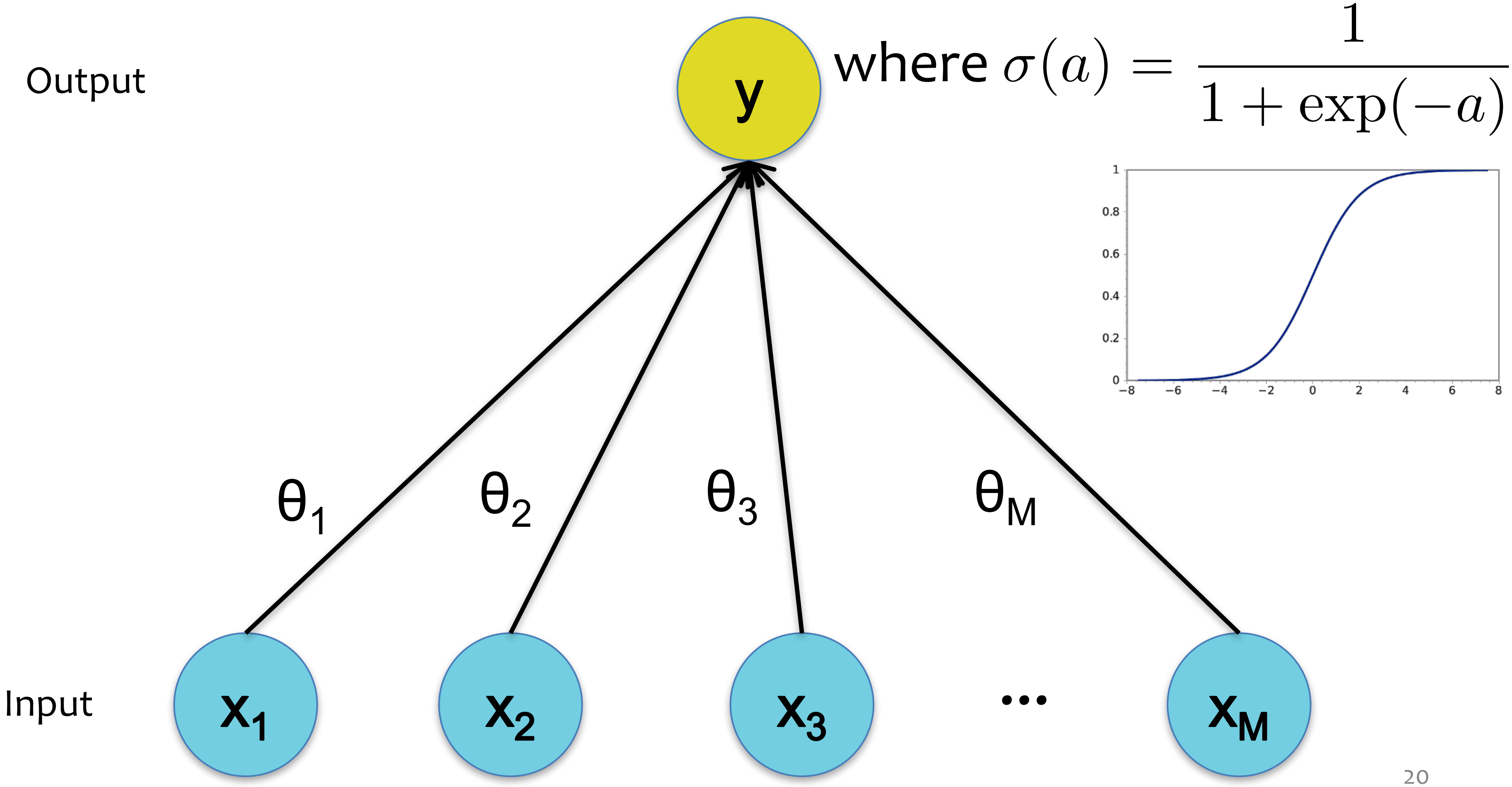
Slide adapted from Eric Xing

Decision Functions

Logistic Regression

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

where $\sigma(a) = \dfrac{1}{1 + \exp(-a)}$

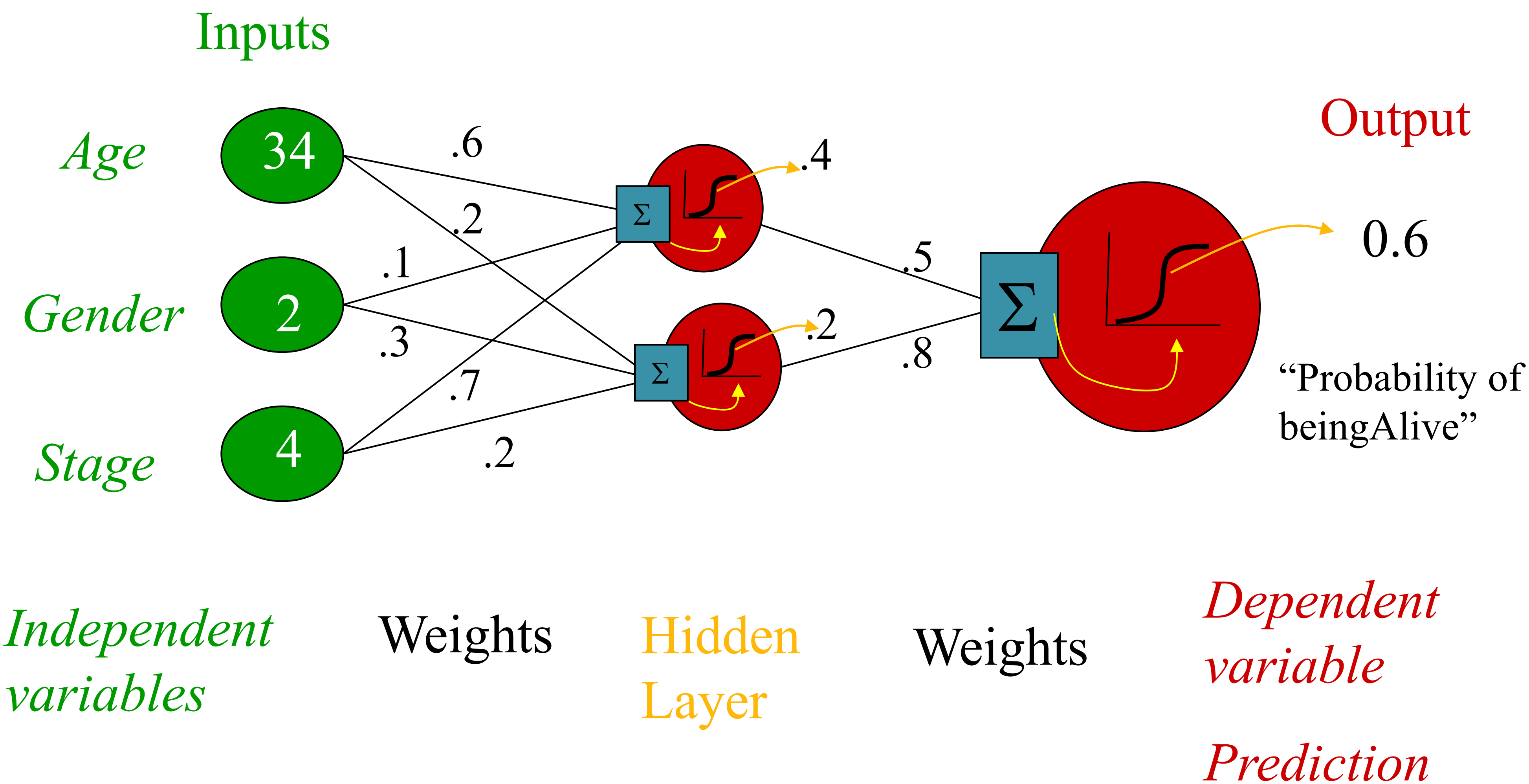


Neural Networks

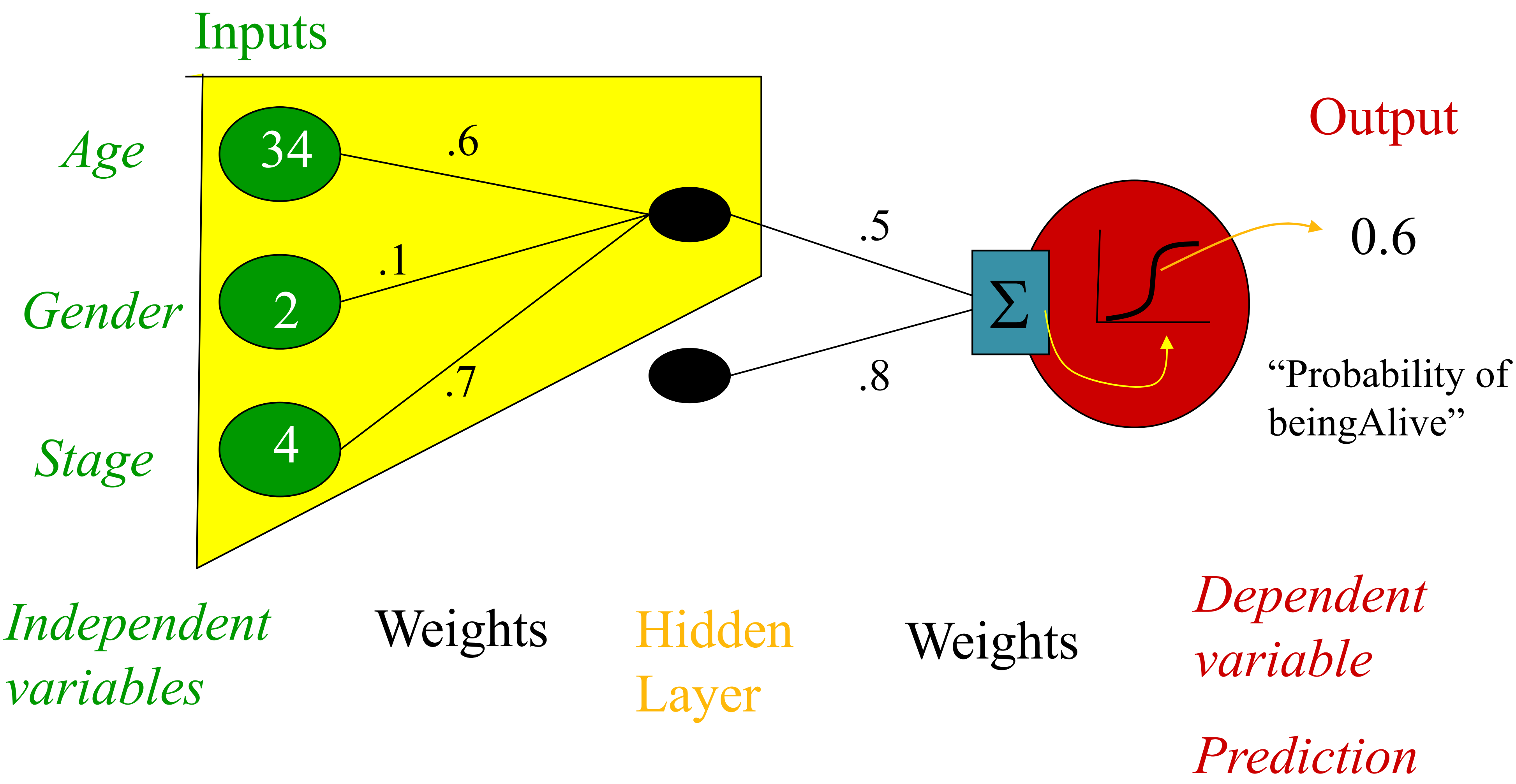
Whiteboard

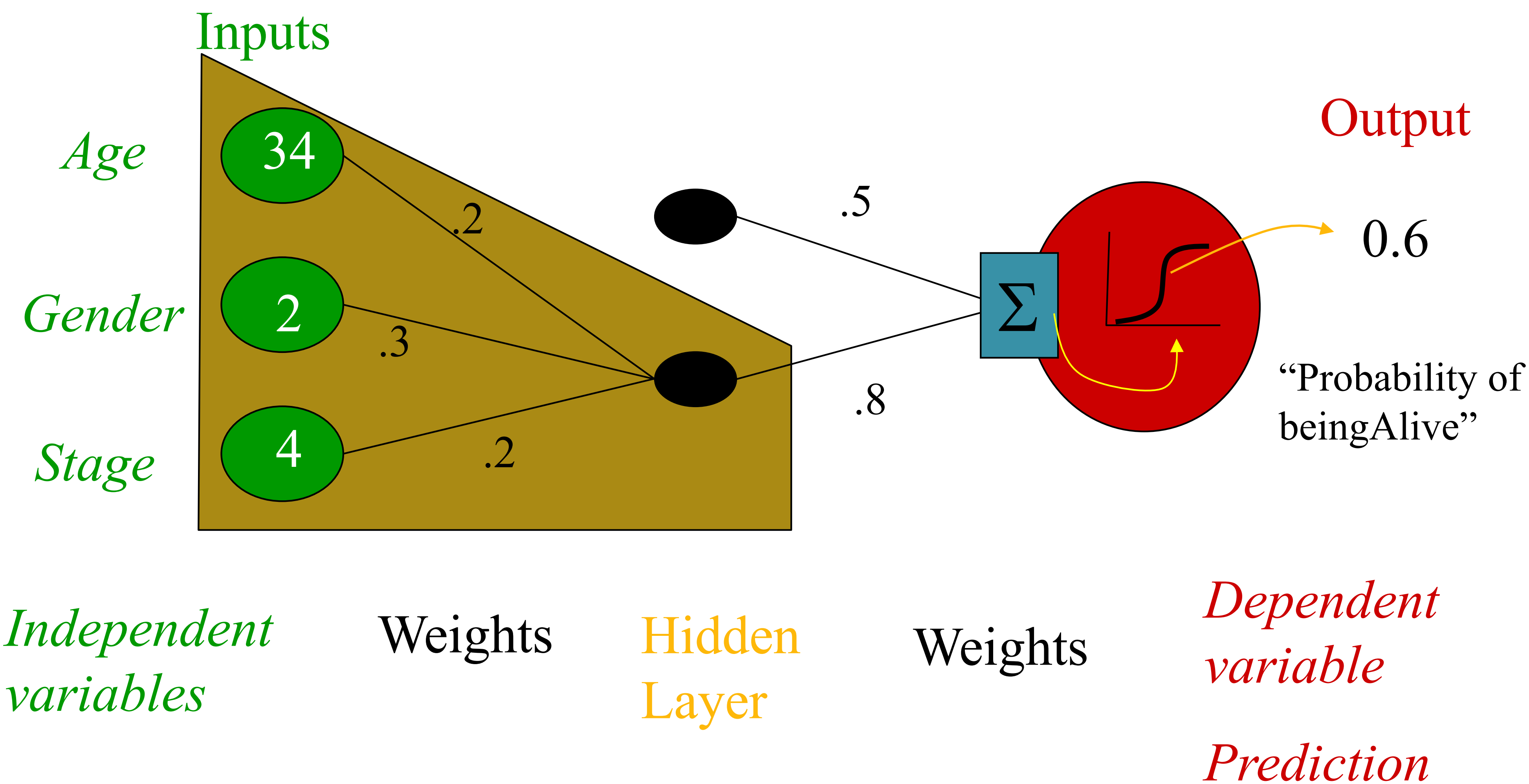
- Example: Neural Network w/1 Hidden Layer
- Example: Neural Network w/2 Hidden Layers
- Example: Feed Forward Neural Network

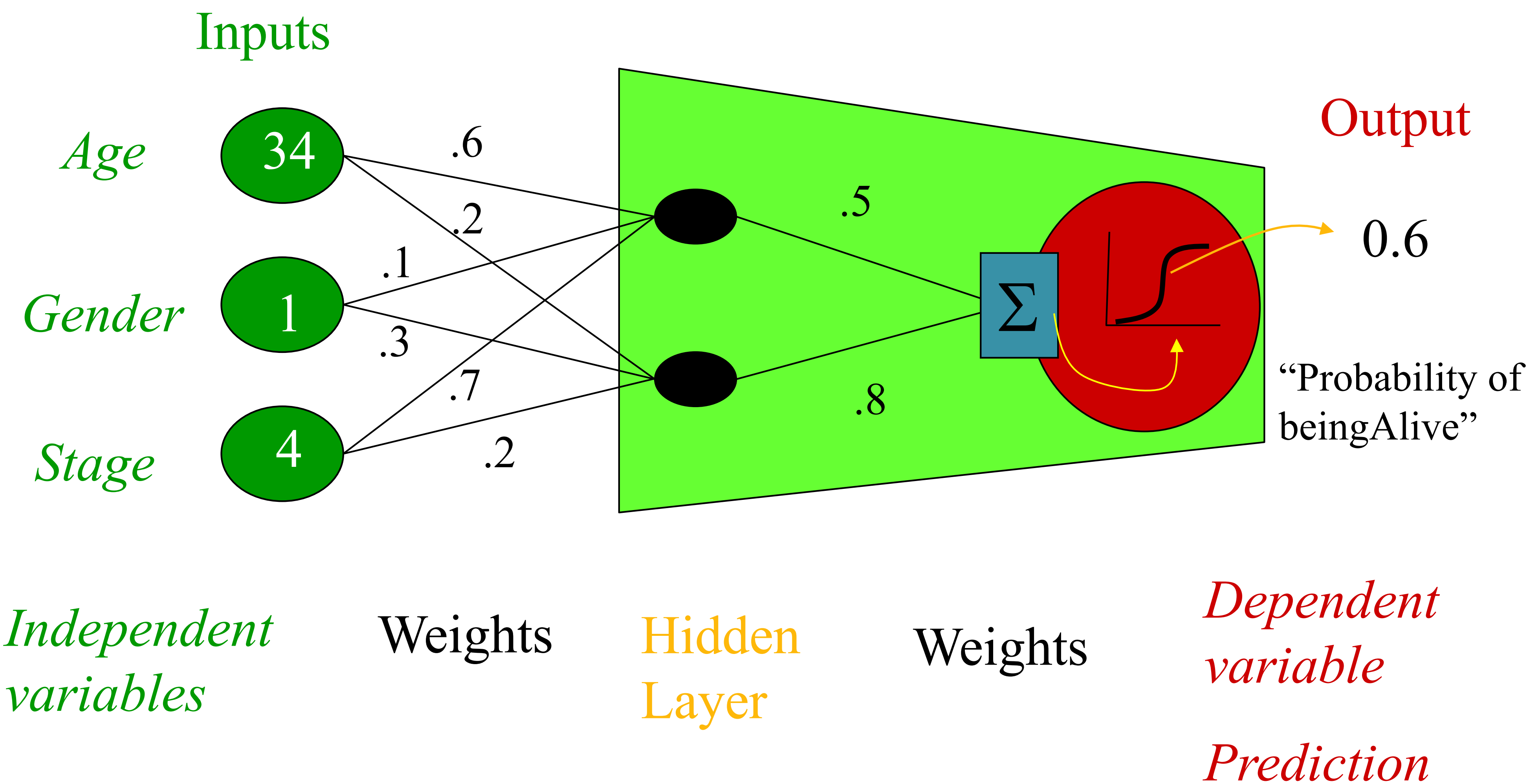
Neural Network Model



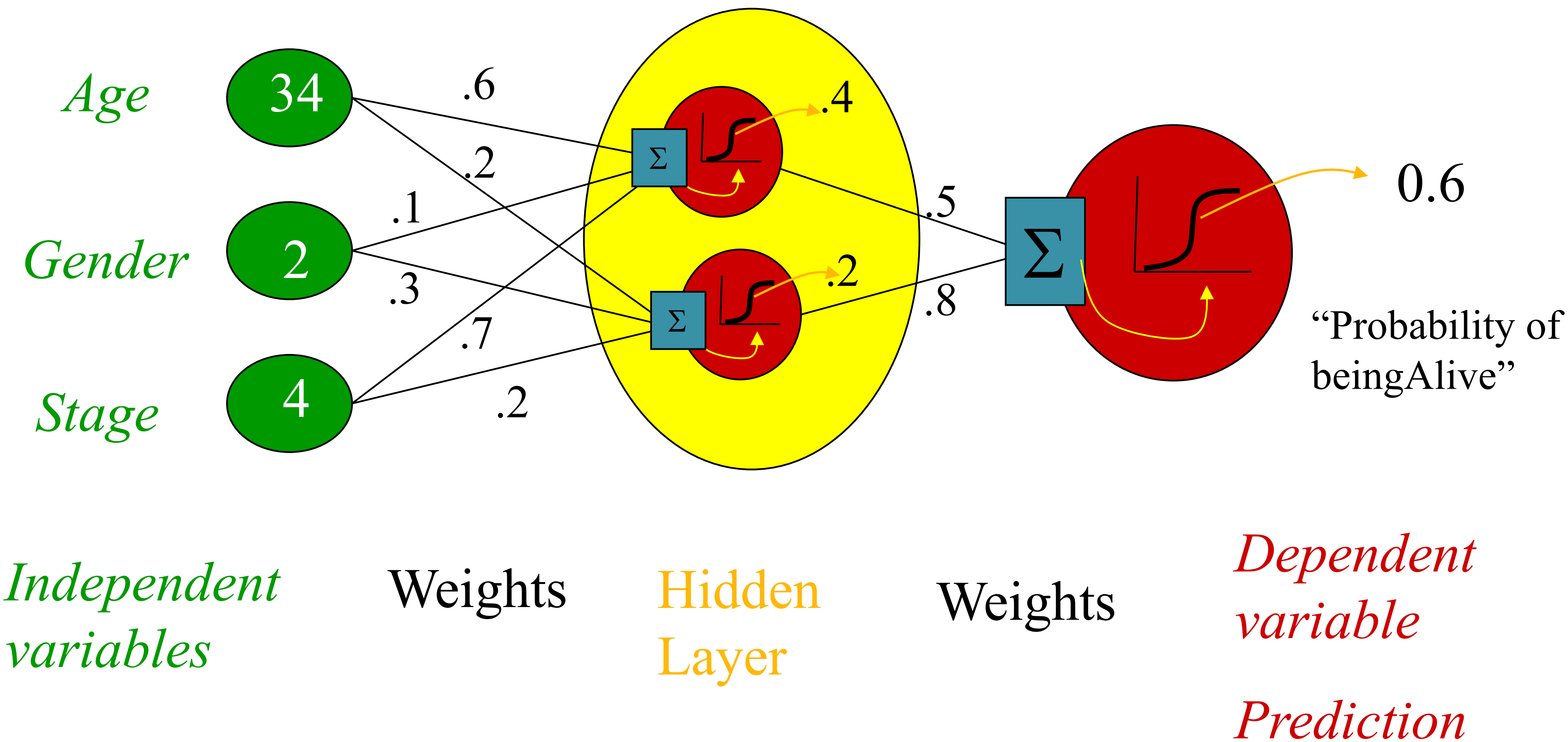
"Combined logistic models"




Inputs
Age
34
Gender
2
Stage
4
.2
.3
.2
.5
.8
Σ
Output
0.6
"Probability of beingAlive"
Independent variables
Weights
Hidden Layer
Weights
Dependent variable
Prediction

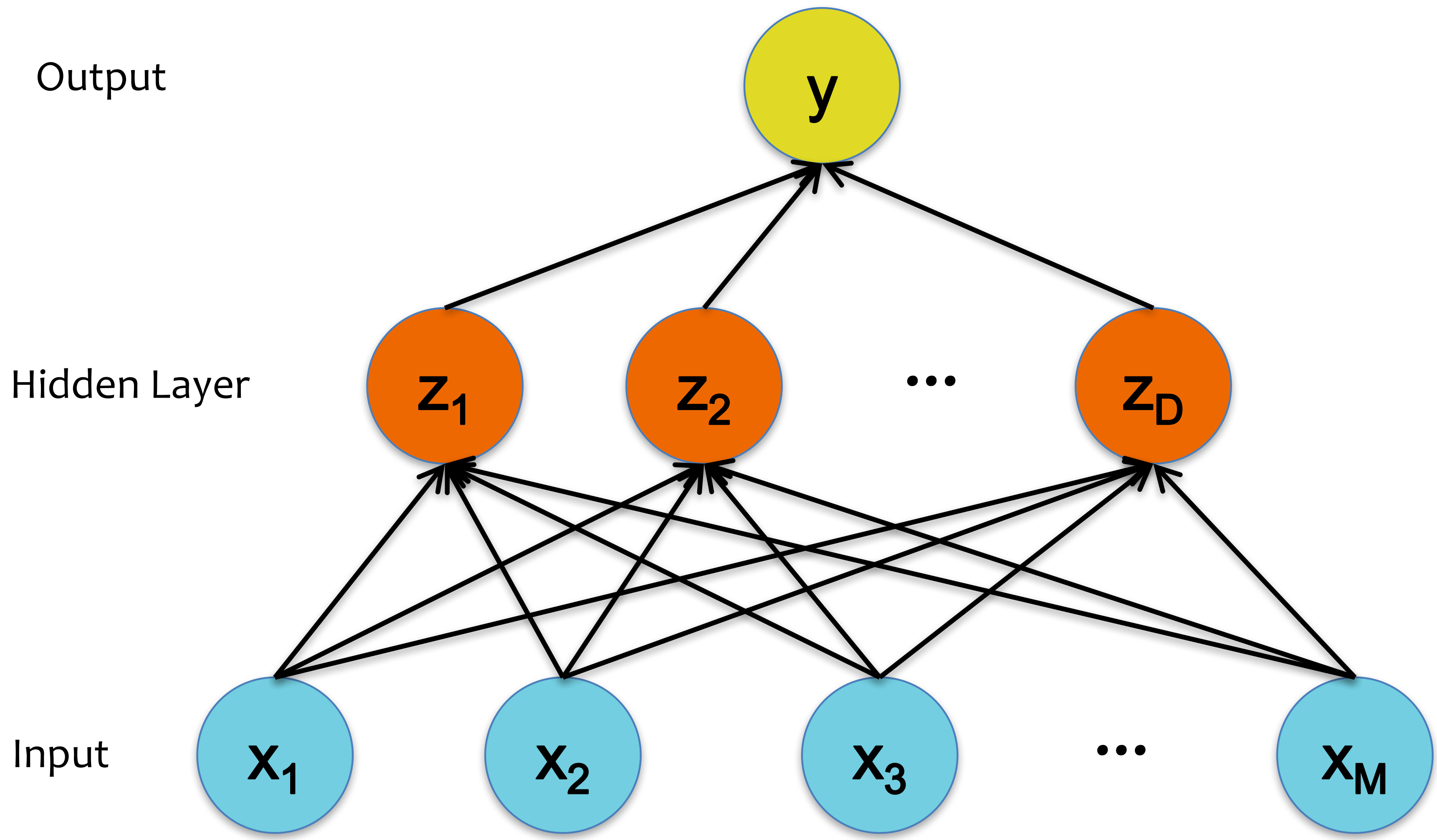

Inputs
Age
Gender
Stage
34
1
4
.6
.2
.1
.3
.7
.2
.5
.8
Σ
Output
0.6
"Probability of beingAlive"
Independent variables
Weights
Hidden Layer
Weights
Dependent variable
Prediction

Not really, no target for hidden units...



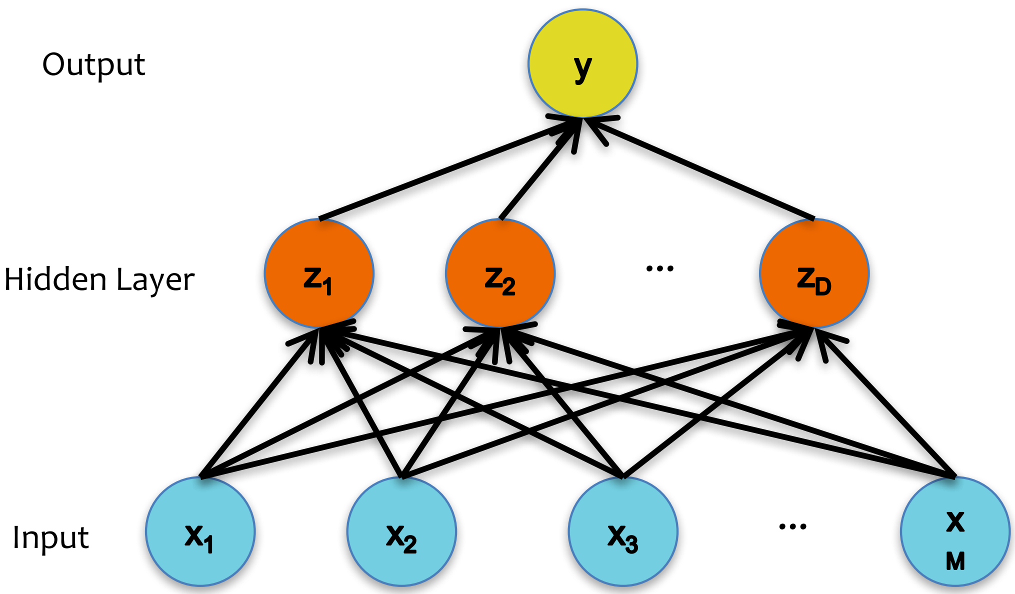
Decision Functions

Neural Network



Decision Functions

Neural Network



(E) Output (sigmoid)
$y = \frac{1}{1+\exp(-b)}$

(D) Output (linear)
$b = \sum_{j=0}^{D} \beta_j z_j$

(C) Hidden (sigmoid)
$z_j = \frac{1}{1+\exp(-a_j)}, \ \forall j$

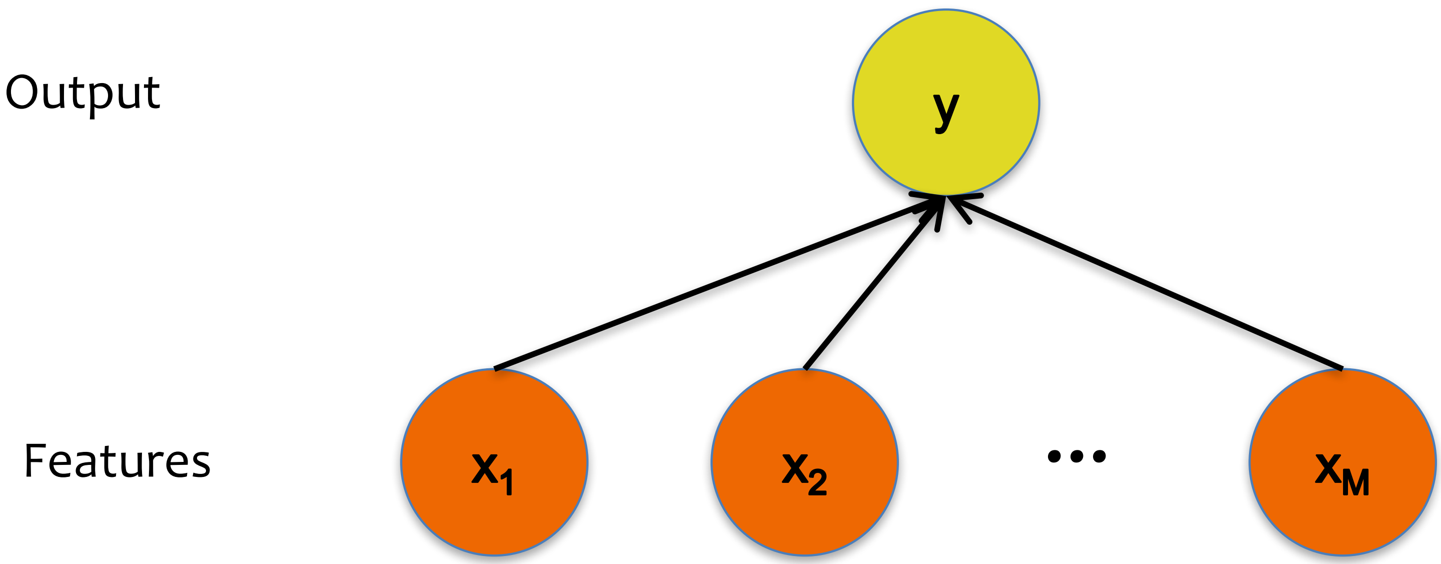
(B) Hidden (linear)
$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \ \forall j$

(A) Input
Given $x_i, \ \forall i$

NEURAL NETWORKS: REPRESENTATIONAL POWER

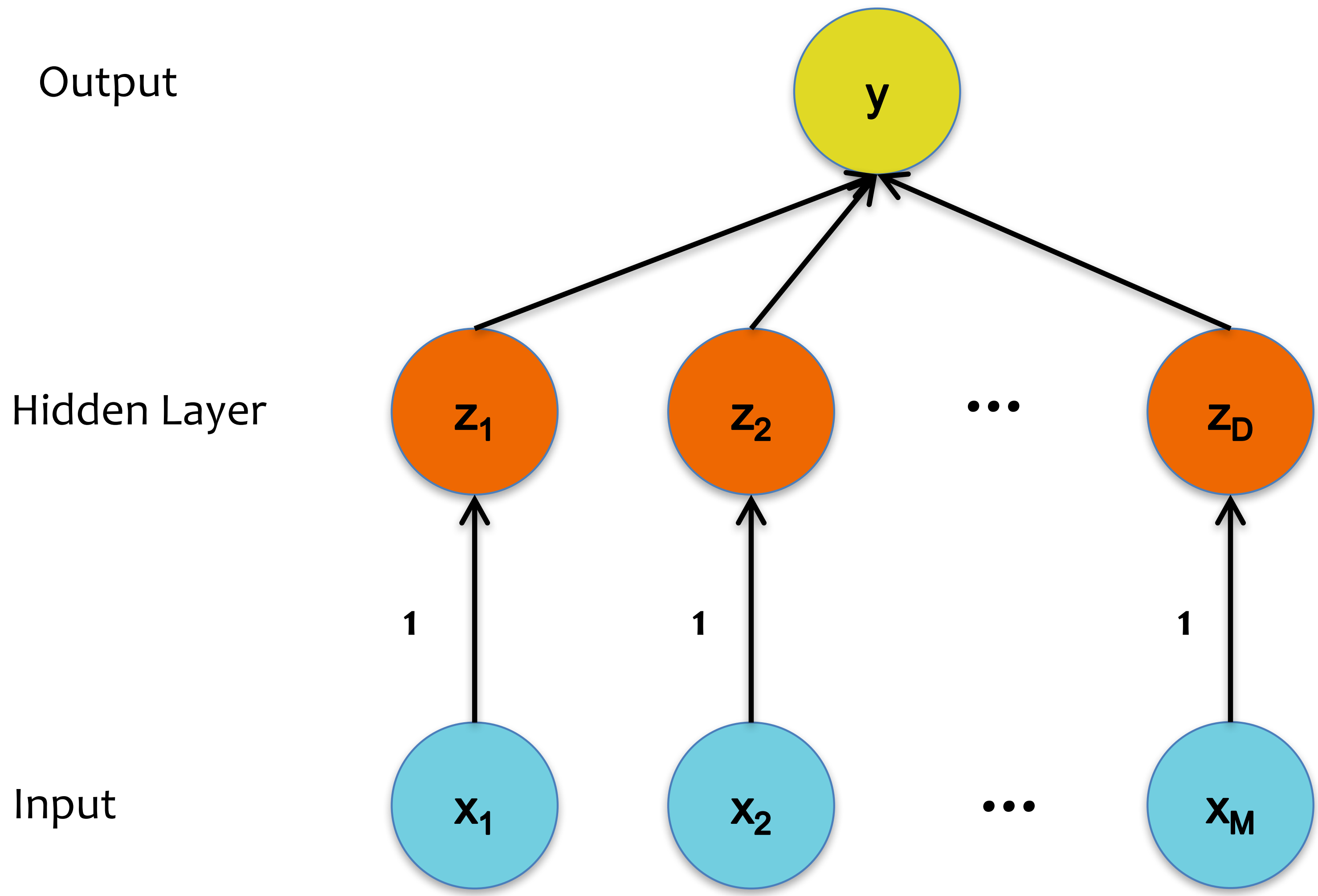
Building a Neural Net

Q: How many hidden units, D, should we use?



Building a Neural Net

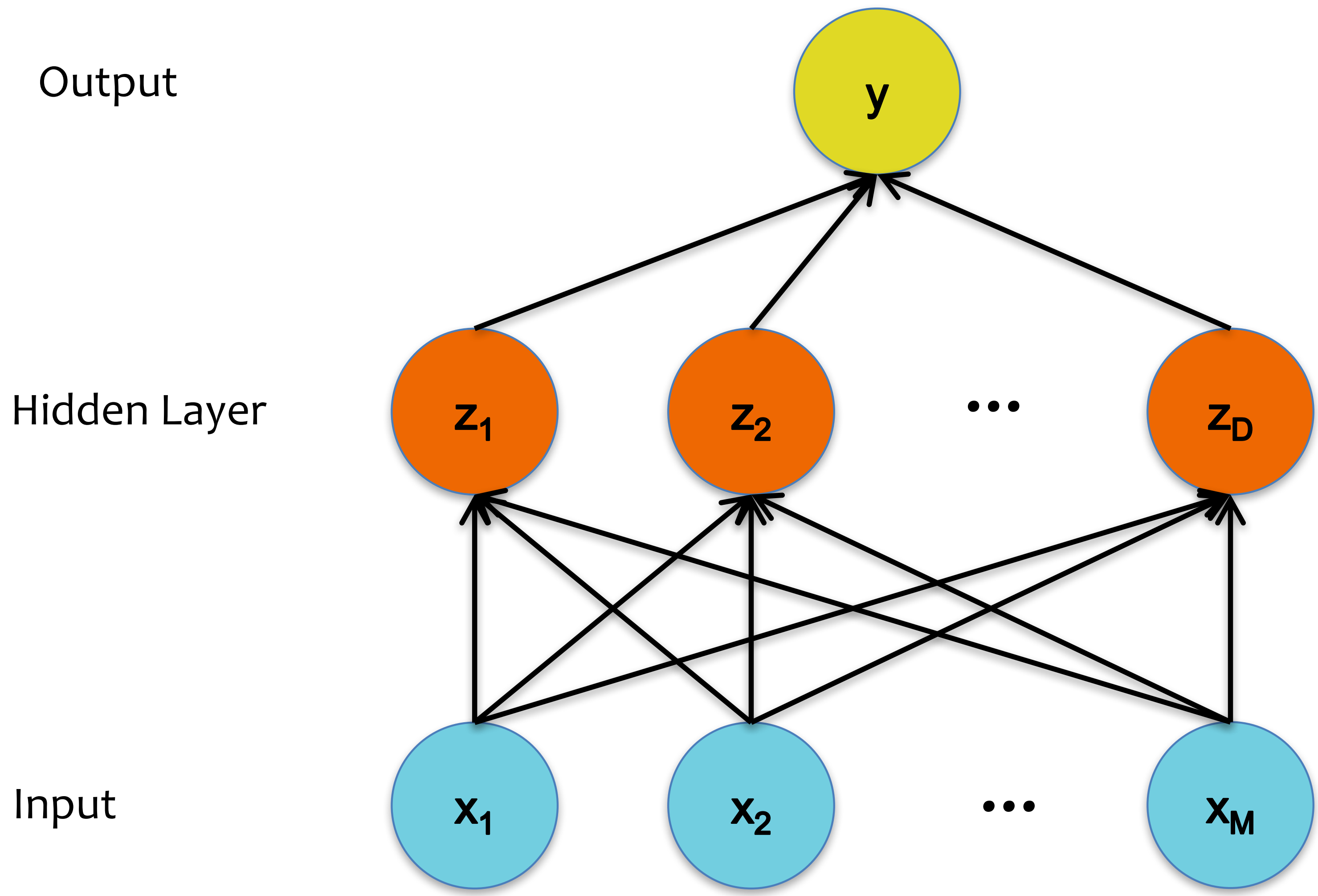
Q: How many hidden units, D, should we use?



D = M

Building a Neural Net

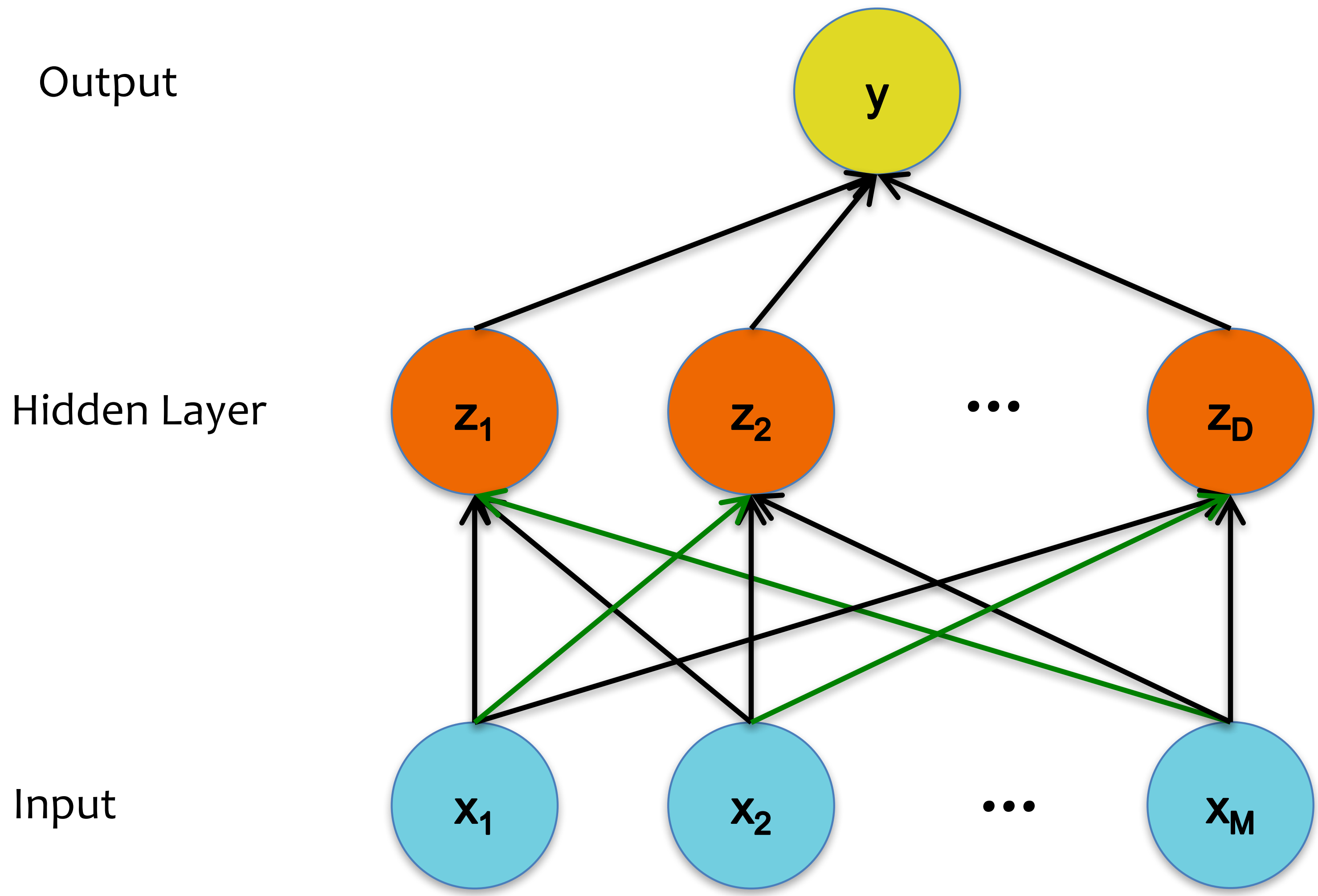
Q: How many hidden units, D, should we use?



D = M

Building a Neural Net

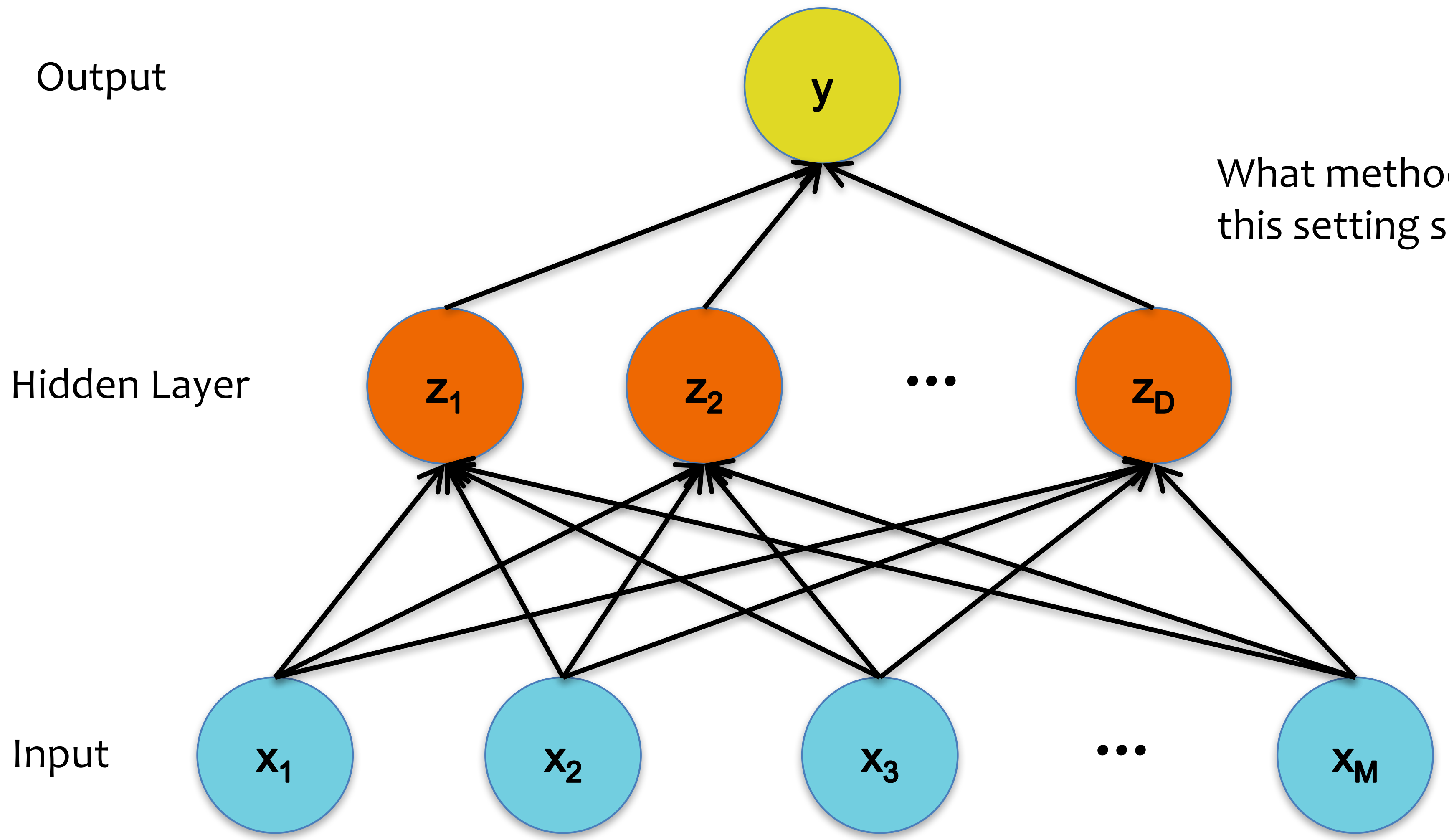
Q: How many hidden units, D, should we use?



D = M

Building a Neural Net

Q: How many hidden units, D, should we use?



What method(s) is this setting similar to?

D < M

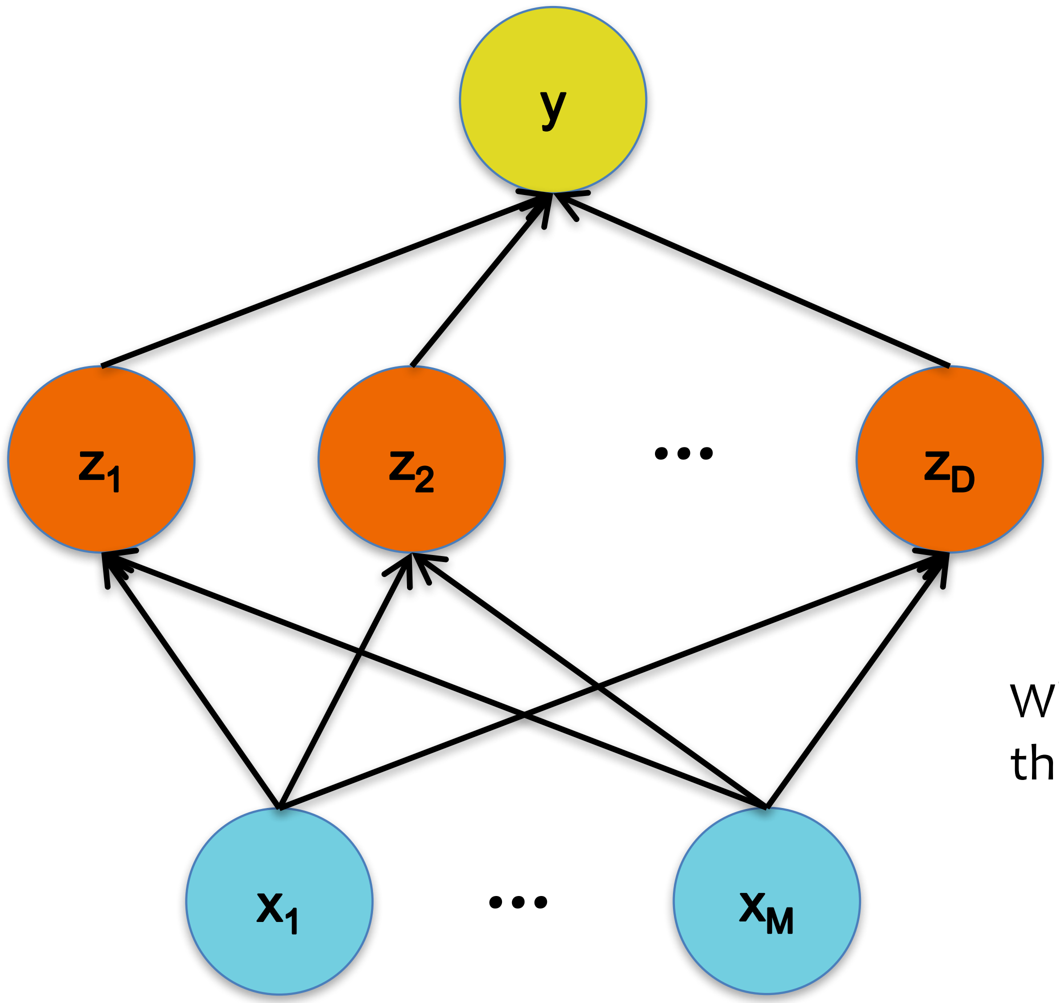
Building a Neural Net

Q: How many hidden units, D, should we use?

Output

Hidden Layer

Input

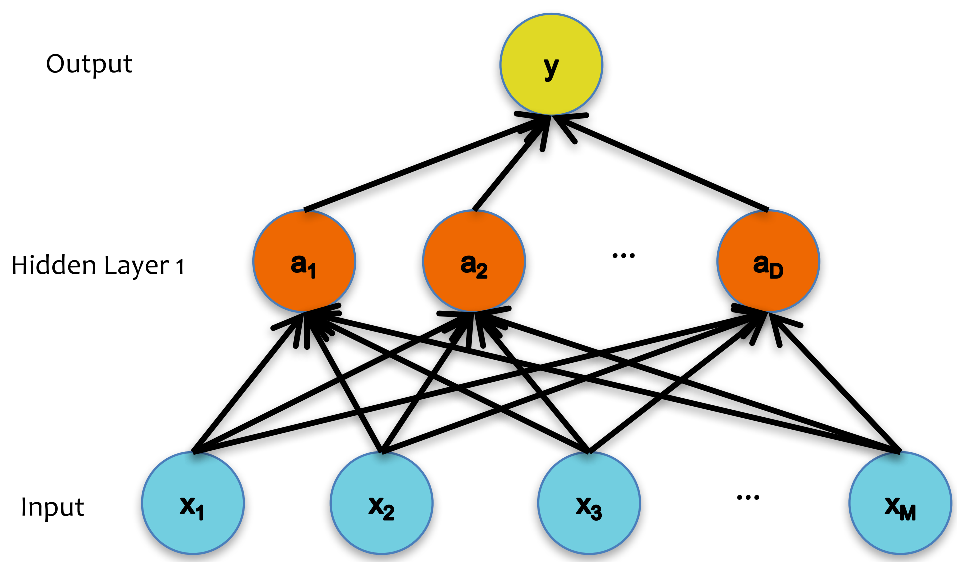


D > M

What method(s) is this setting similar to?

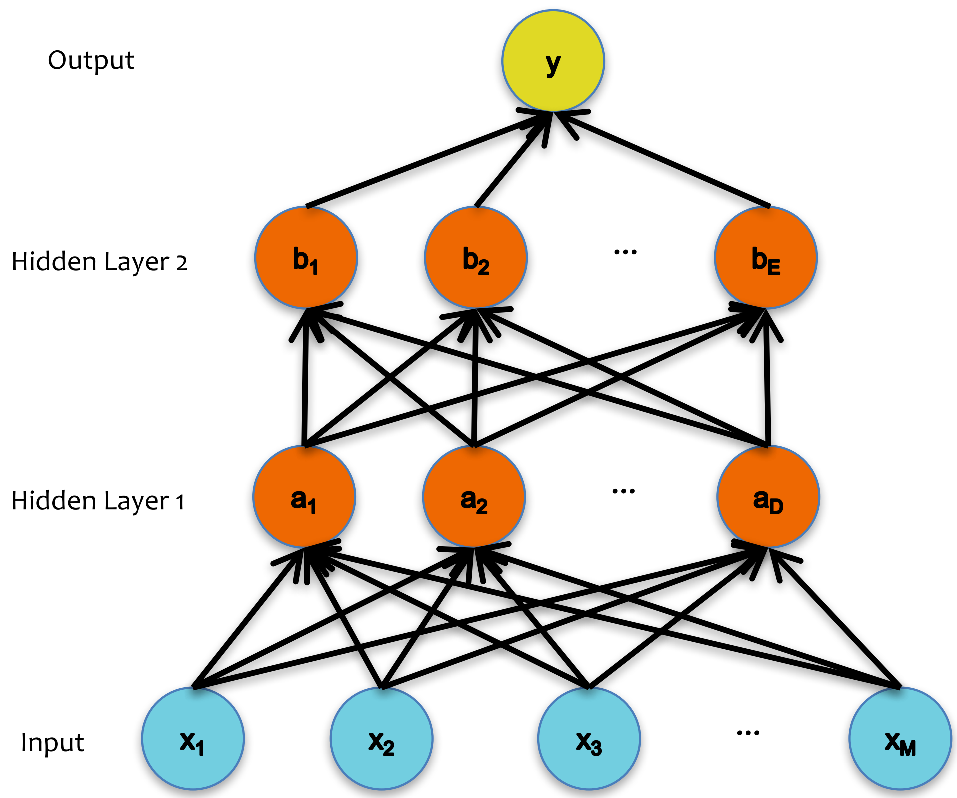
Deeper Networks

Q: How many layers should we use?



Deeper Networks

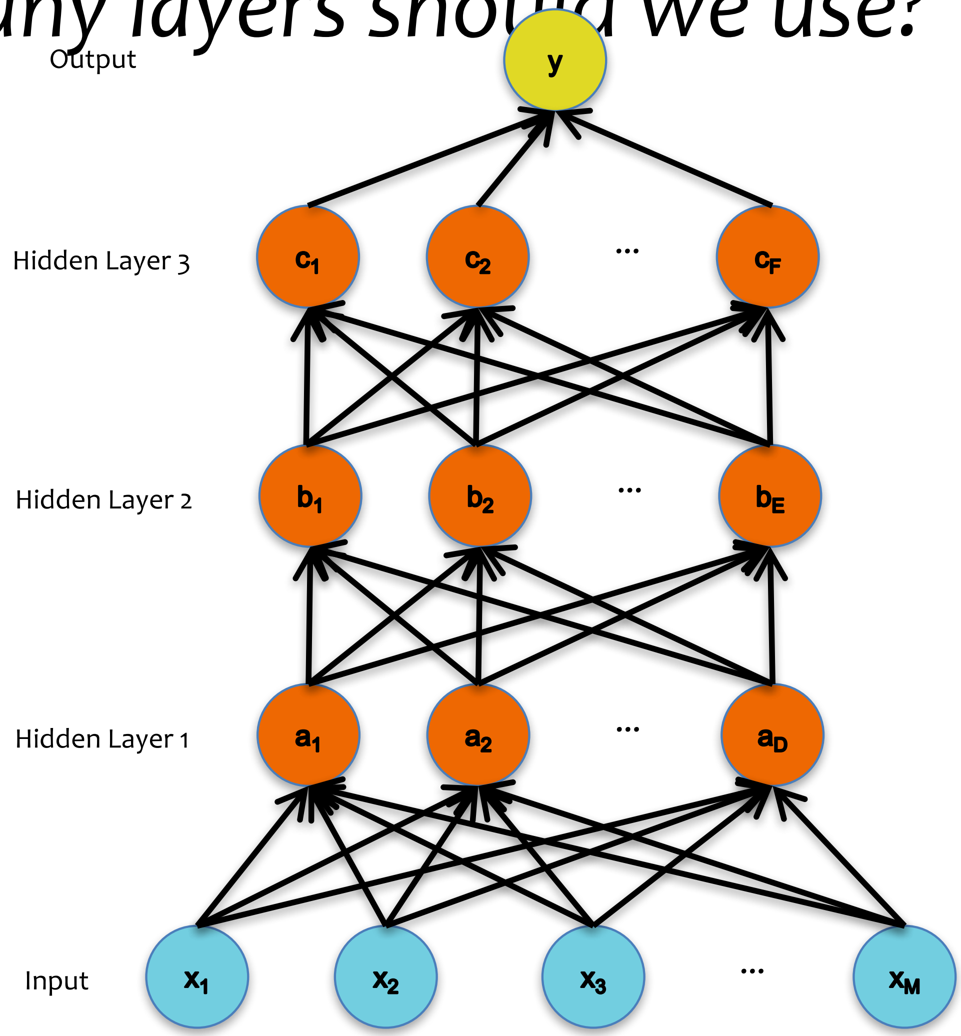
Q: How many layers should we use?



Decision Functions

Deeper Networks

Q: How many layers should we use?



Decision Functions

Deeper Networks

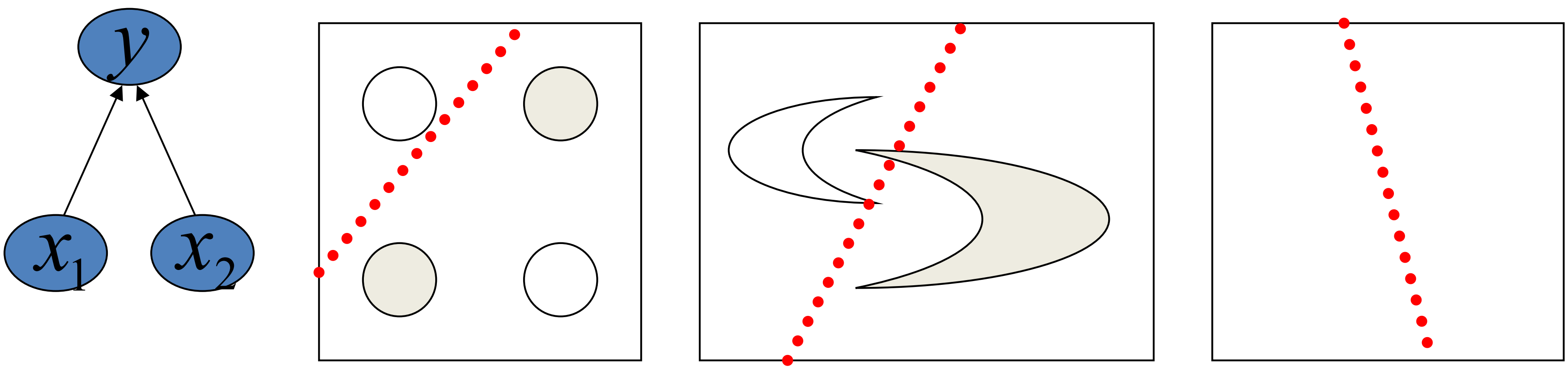
Q: How many layers should we use?

- **Theoretical answer:**
  - A neural network with 1 hidden layer is a **universal function approximator**
  - Cybenko (1989): For any continuous function $g(\mathbf{x})$, there exists a 1-hidden-layer neural net $h_\theta(\mathbf{x})$ s.t. $|\, h_\theta(\mathbf{x}) - g(\mathbf{x}) \,| < \epsilon$ for all $\mathbf{x}$, assuming sigmoid activation functions
- **Empirical answer:**
  - Before 2006: "Deep networks (e.g. 3 or more hidden layers) are too hard to train"
  - After 2006: "Deep networks are easier to train than shallow networks (e.g. 2 or fewer layers) for many problems"

    Big caveat: You need to know and use the right tricks.

Decision Boundary

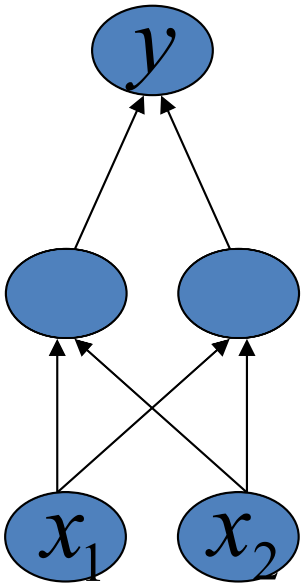
- 0 hidden layers: linear classifier
  - Hyperplanes

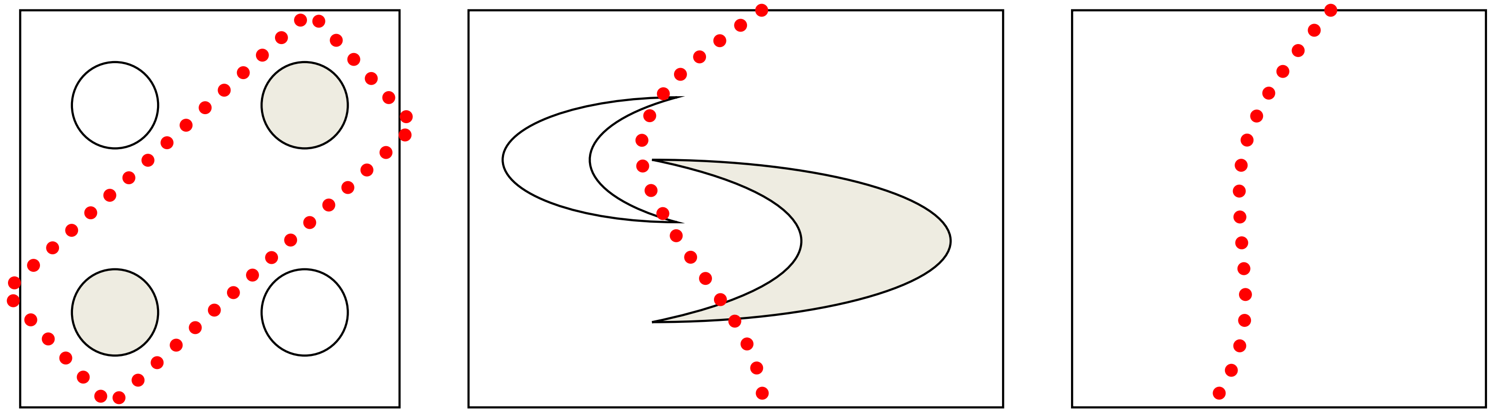


Example from to Eric Postma via Jason Eisner

Decision Boundary

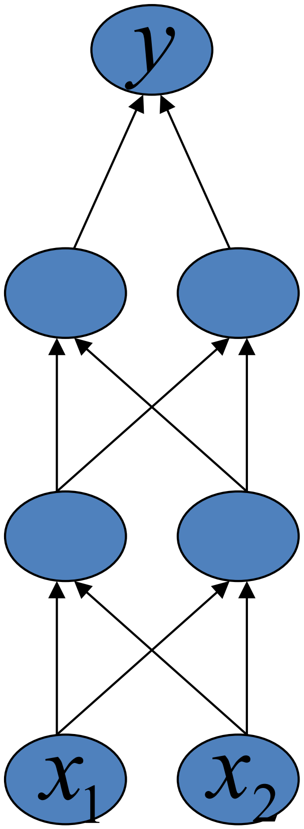
- 1 hidden layer
  - Boundary of convex region (open or closed)



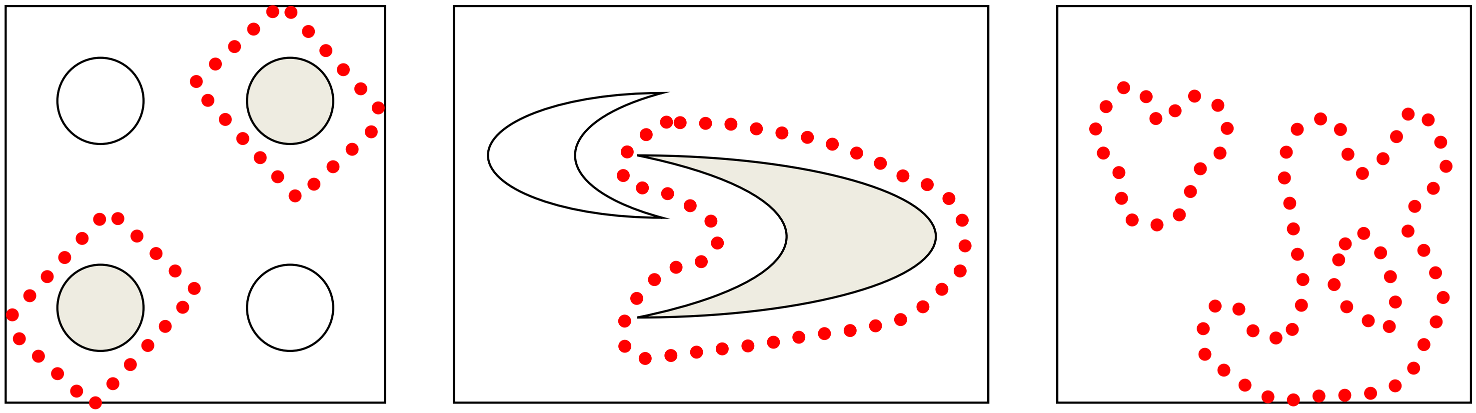


Example from to Eric Postma via Jason Eisner

Decision Boundary



- 2 hidden layers
  - Combinations of convex regions

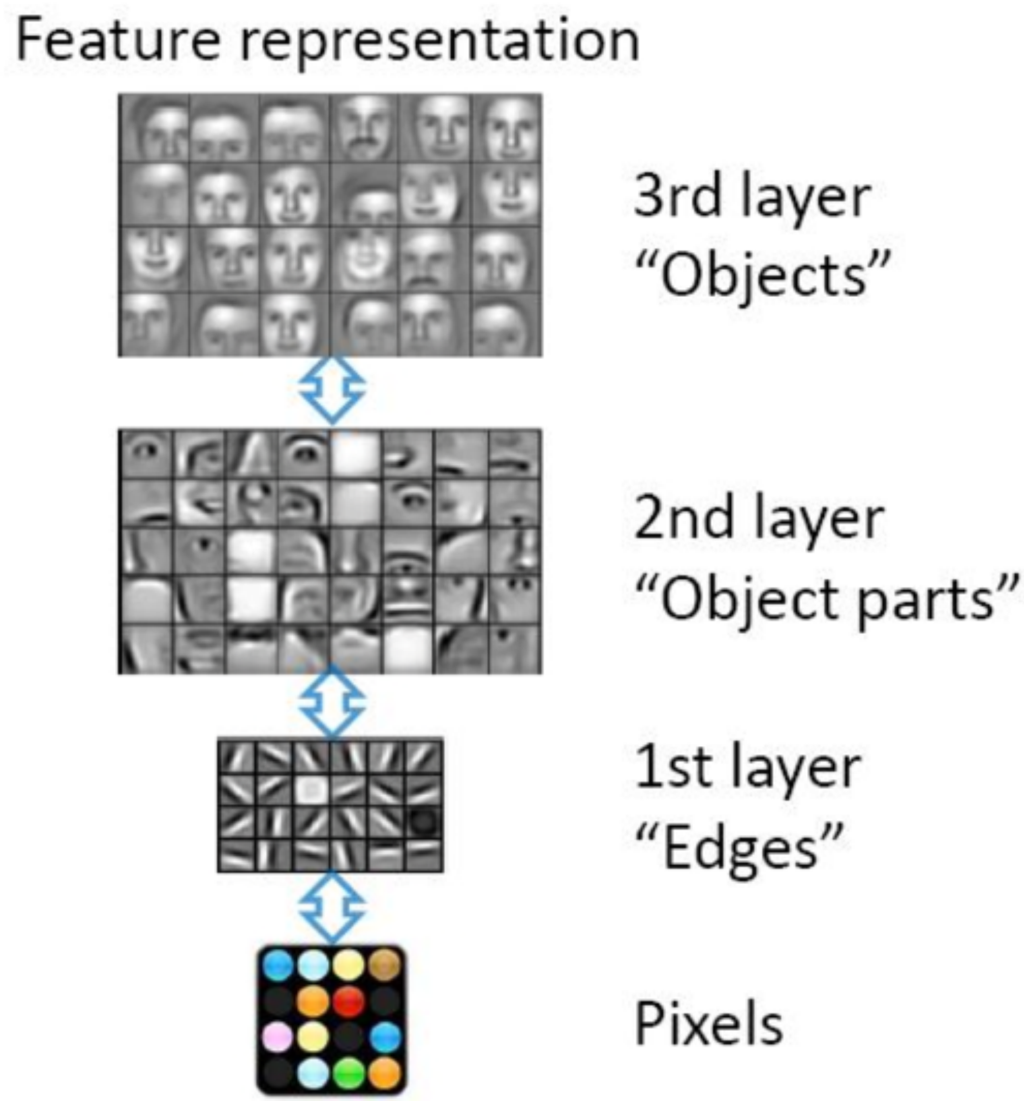


Example from to Eric Postma via Jason Eisner

Decision Functions

Different Levels of Abstraction

- We don't know the "right" levels of abstraction
- So let the model figure it out!



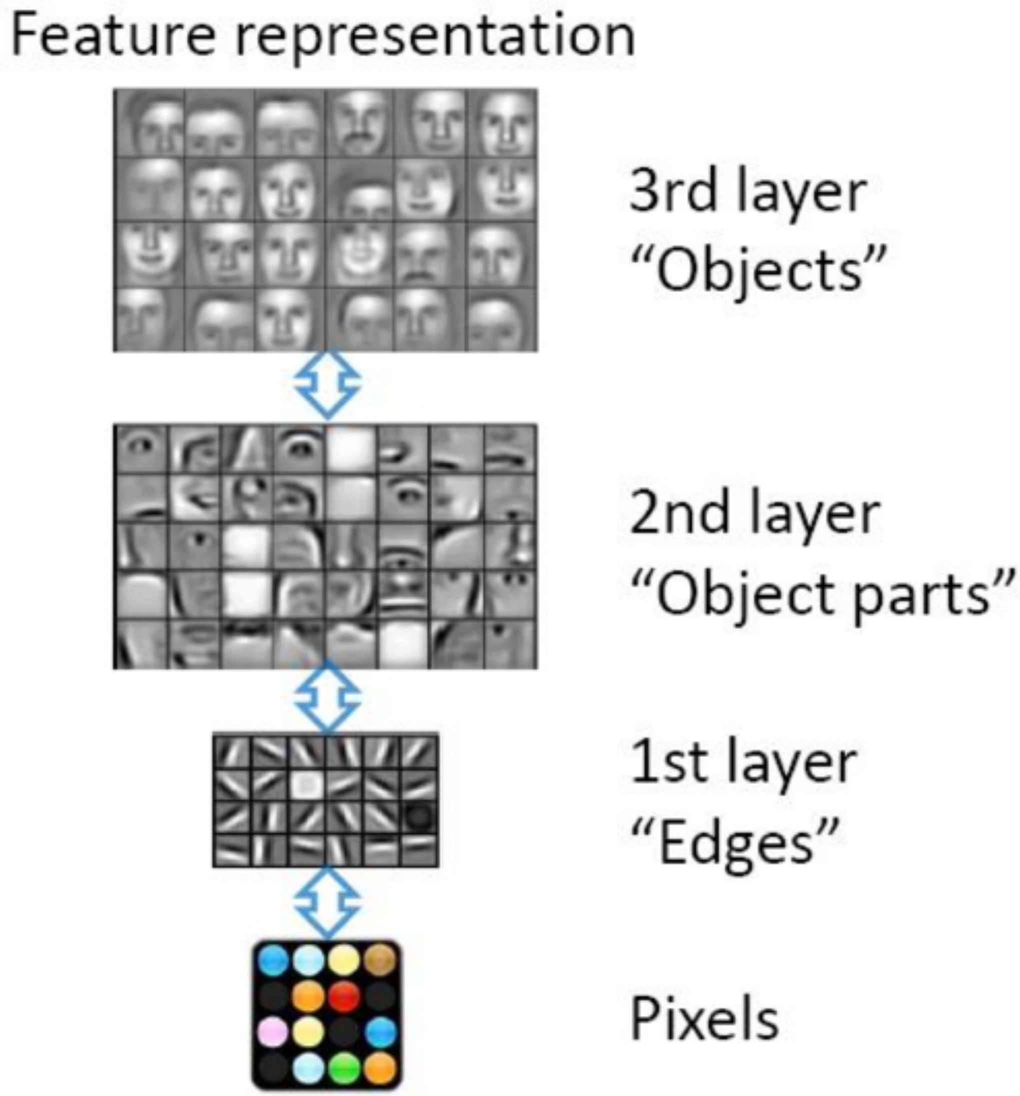
Example from Honglak Lee (NIPS 2010)

Decision Functions

Different Levels of Abstraction

Face Recognition:

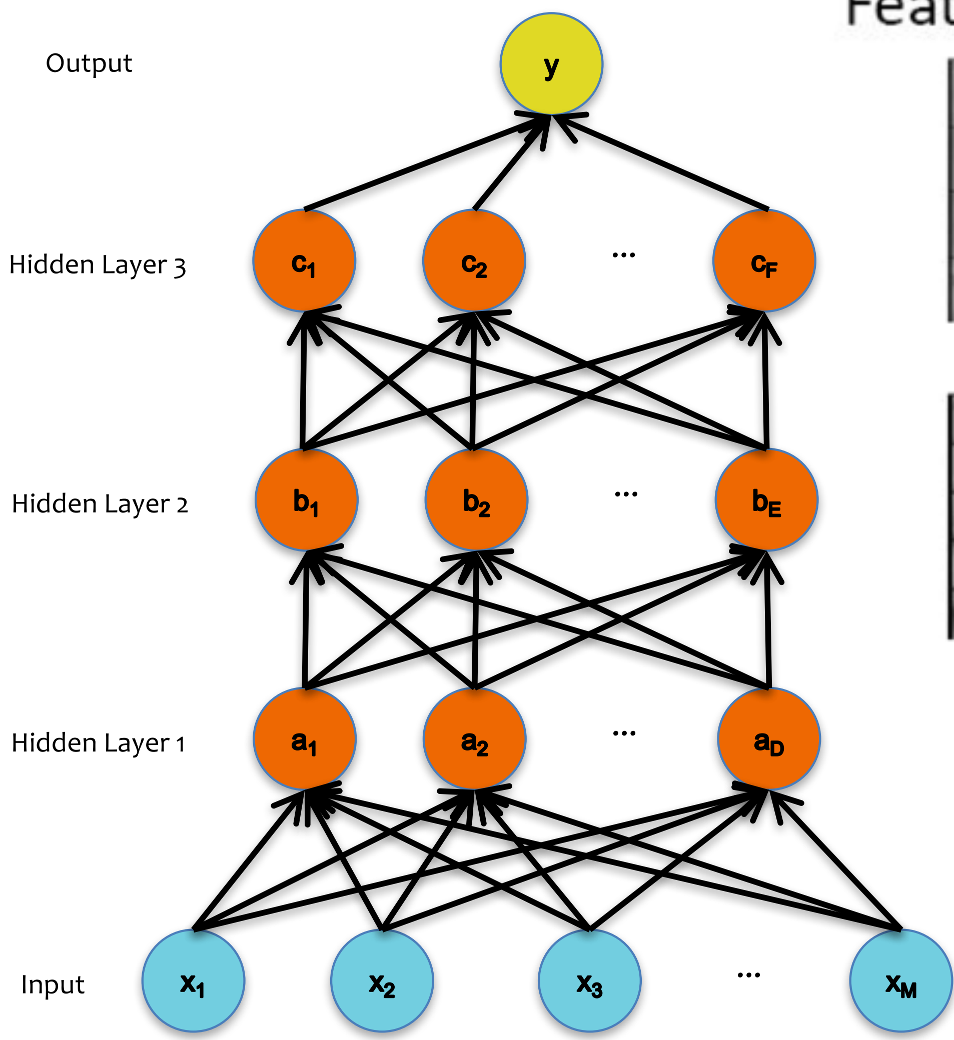
- Deep Network can build up increasingly higher levels of abstraction
- Lines, parts, regions

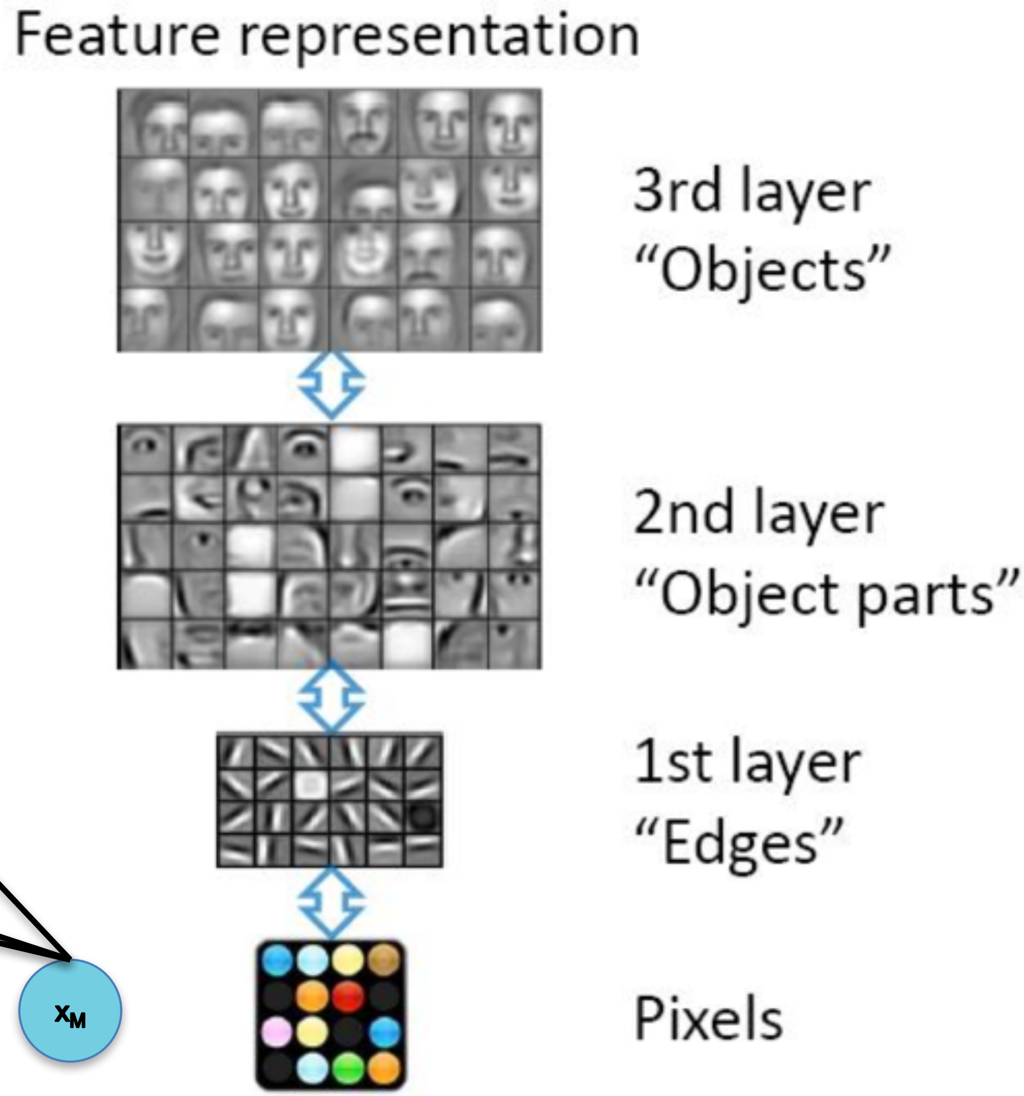


Example from Honglak Lee (NIPS 2010)

Decision Functions

Different Levels of Abstraction





Example from Honglak Lee (NIPS 2010)