



10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Neural Networks

Matt Gormley
Lecture 12
Feb. 24, 2020

Reminders

- **Homework 4: Logistic Regression**
 - Out: Wed, Feb. 19
 - Due: Fri, Feb. 28 at 11:59pm
- **Today's In-Class Poll**
 - <http://p12.mlcourse.org>
- **Swapped lecture/recitation:**
 - Lecture 14: Fri, Feb. 28
 - Recitation HW5: Mon, Mar. 02

Q&A

NEURAL NETWORKS

Background

A Recipe for Machine Learning

1. Given training data:

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

2. Choose each of these:

- Decision function

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

- Loss function

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

Face



Face



Not a face



Examples: Linear regression,
Logistic regression, Neural Network

Examples: Mean-squared error,
Cross Entropy

Background

A Recipe for Machine Learning

1. Given training data:

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

2. Choose each of these:

- Decision function

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

- Loss function

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

3. Define goal:

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

4. Train with SGD:

(take small steps
opposite the gradient)

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

Background

1. Given training data

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

2. Choose each of the

– Decision function

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

– Loss function

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

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!

opposite the gradient)


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

A Recipe for

Goals for Today's Lecture

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

– Decision function

$$\hat{y} = f_{\theta}(x_i)$$

– Loss function

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

4. Train with SGD:

(Take small steps opposite the gradient)

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \nabla \ell(f_{\theta}(x_i), y_i)$$



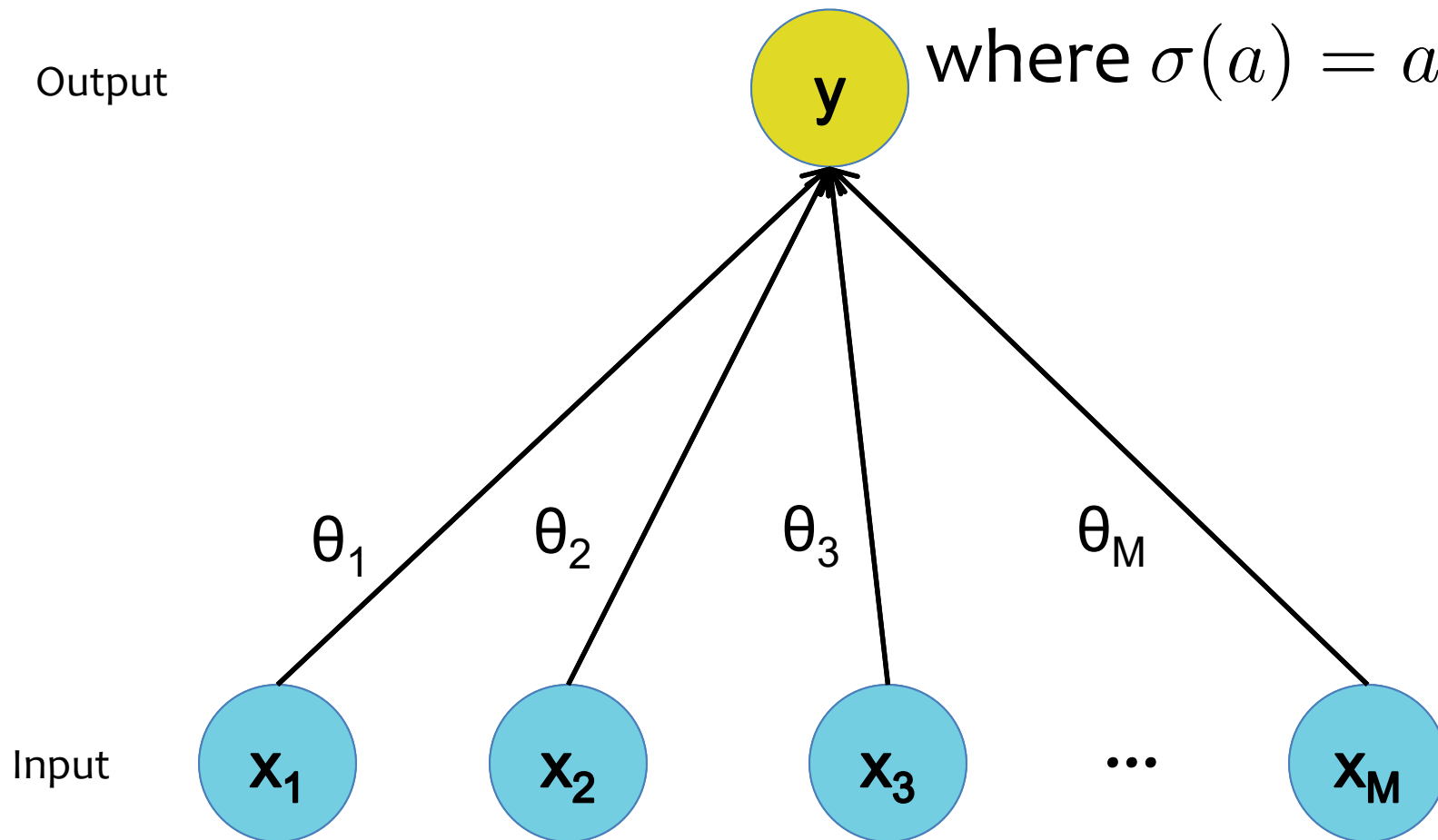
Decision Functions

Linear Regression

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

where $\sigma(a) = a$

Output

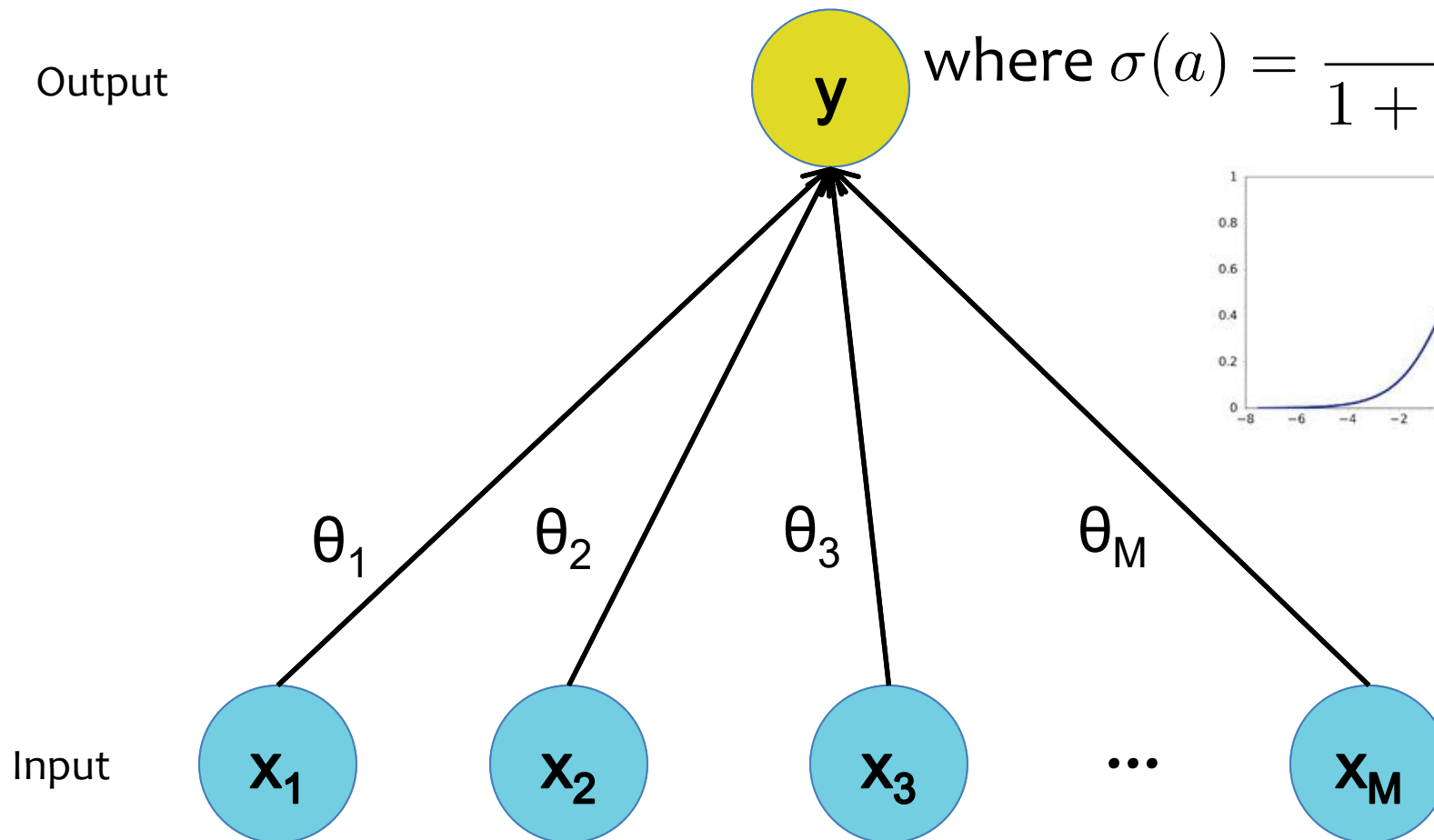
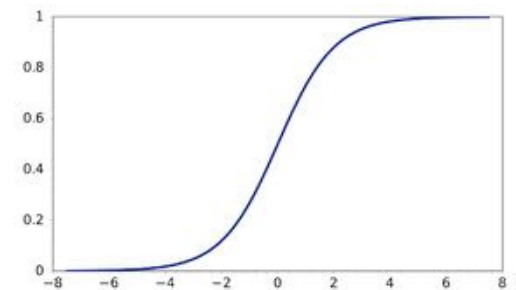


Decision Functions

Logistic Regression

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

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



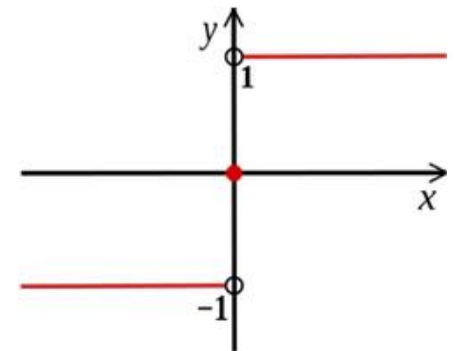
Decision Functions

Perceptron

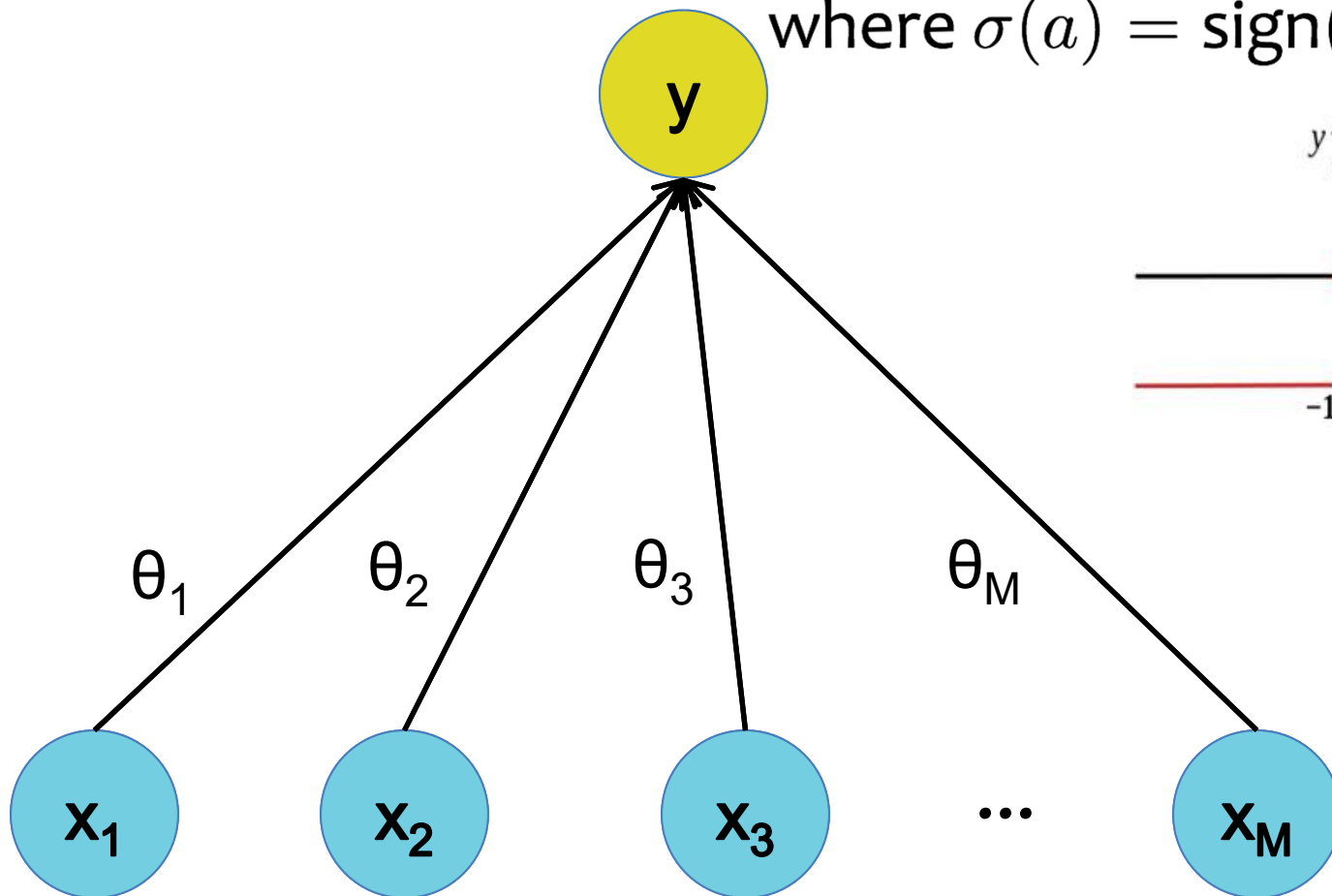
$$y = h_{\theta}(\mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

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

Output

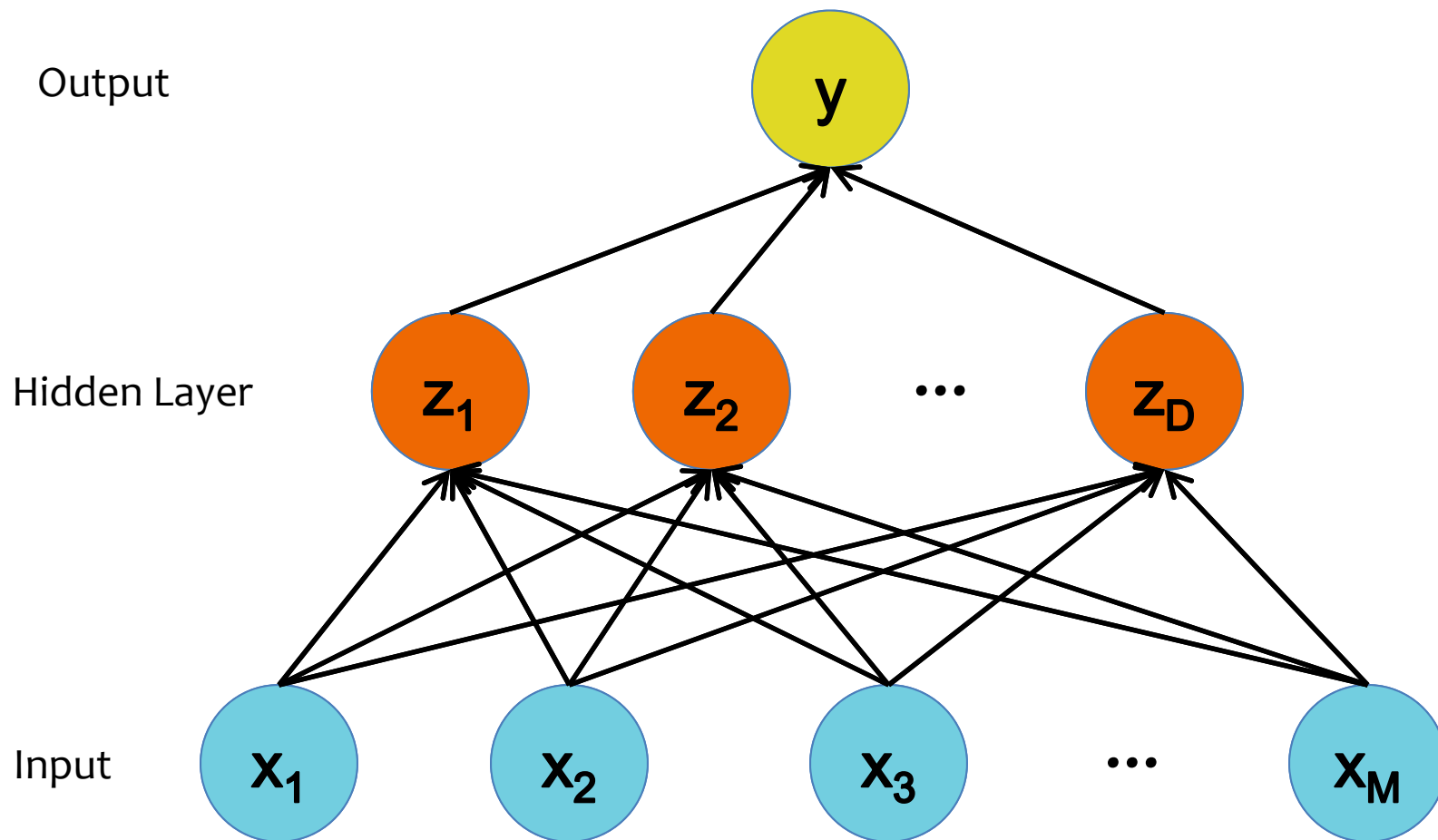


Input

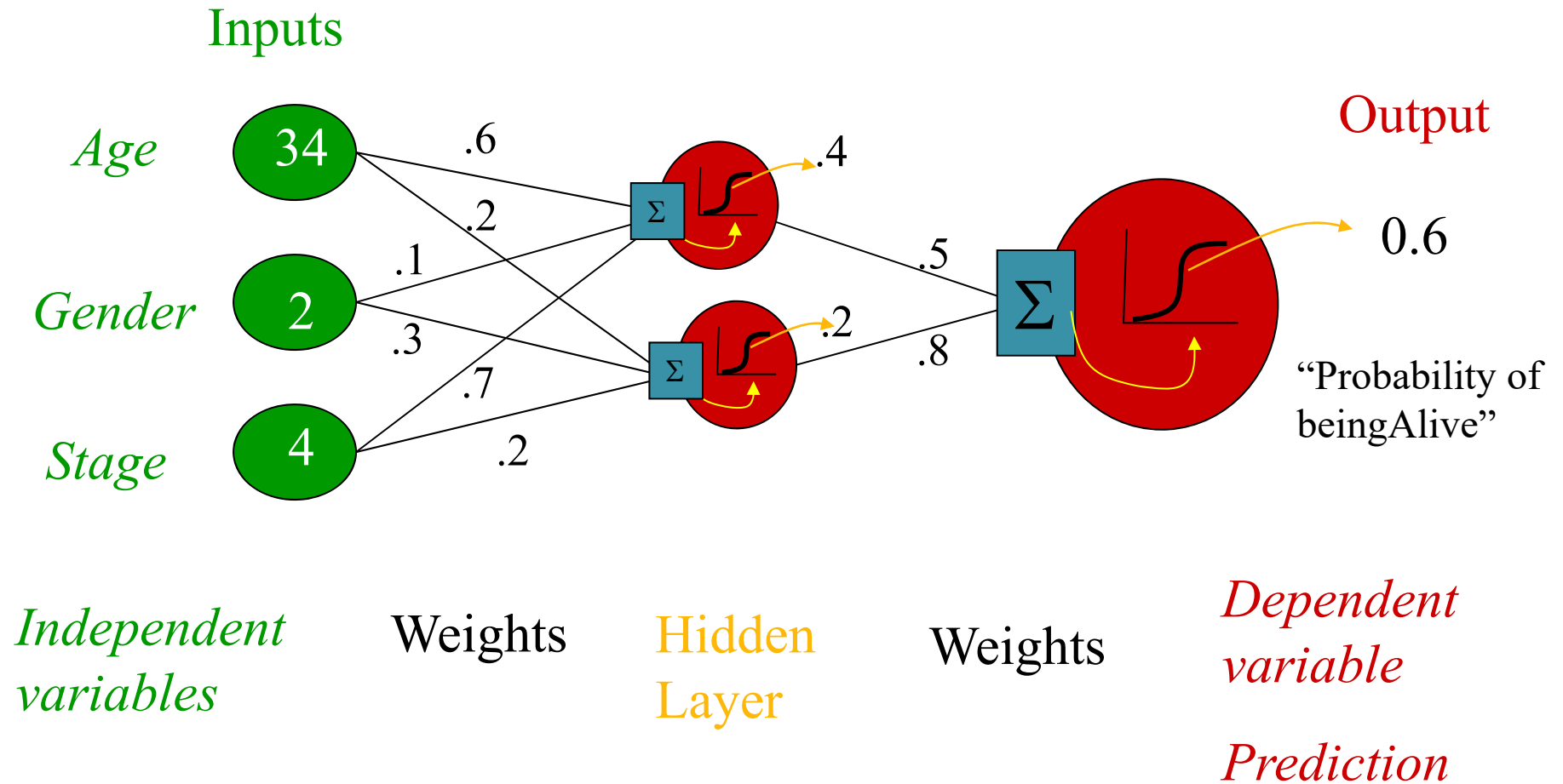


Decision Functions

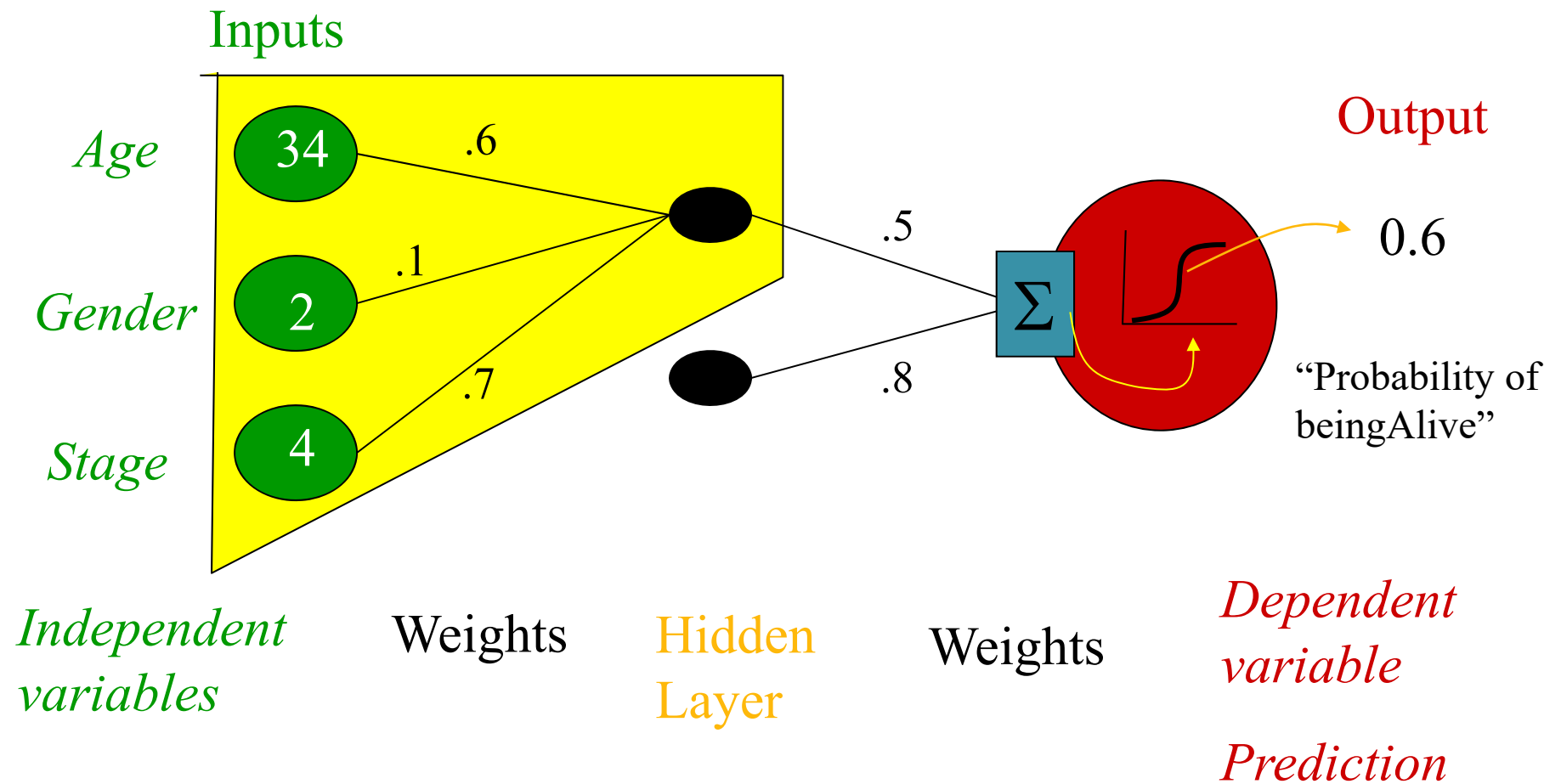
Neural Network

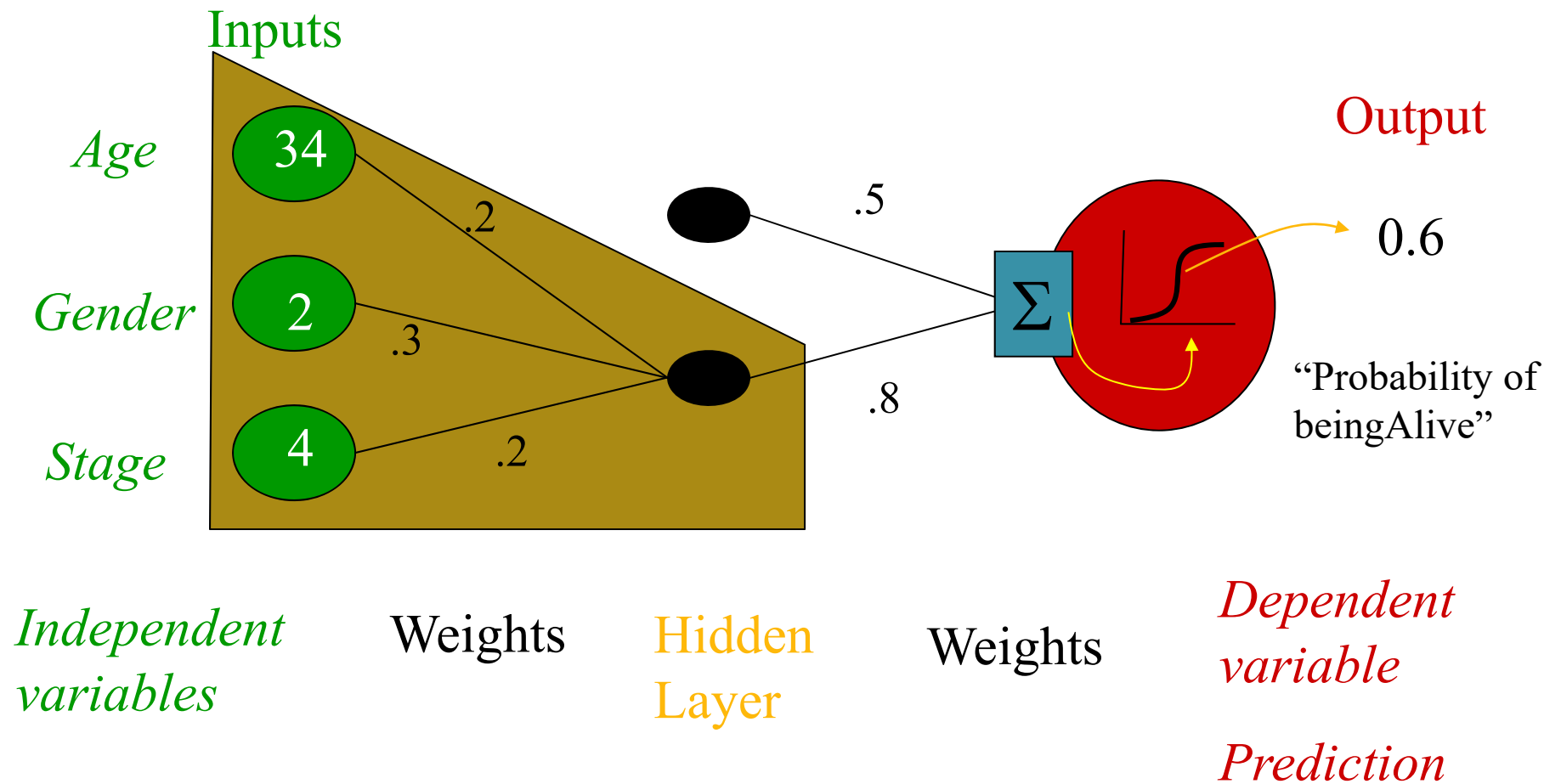


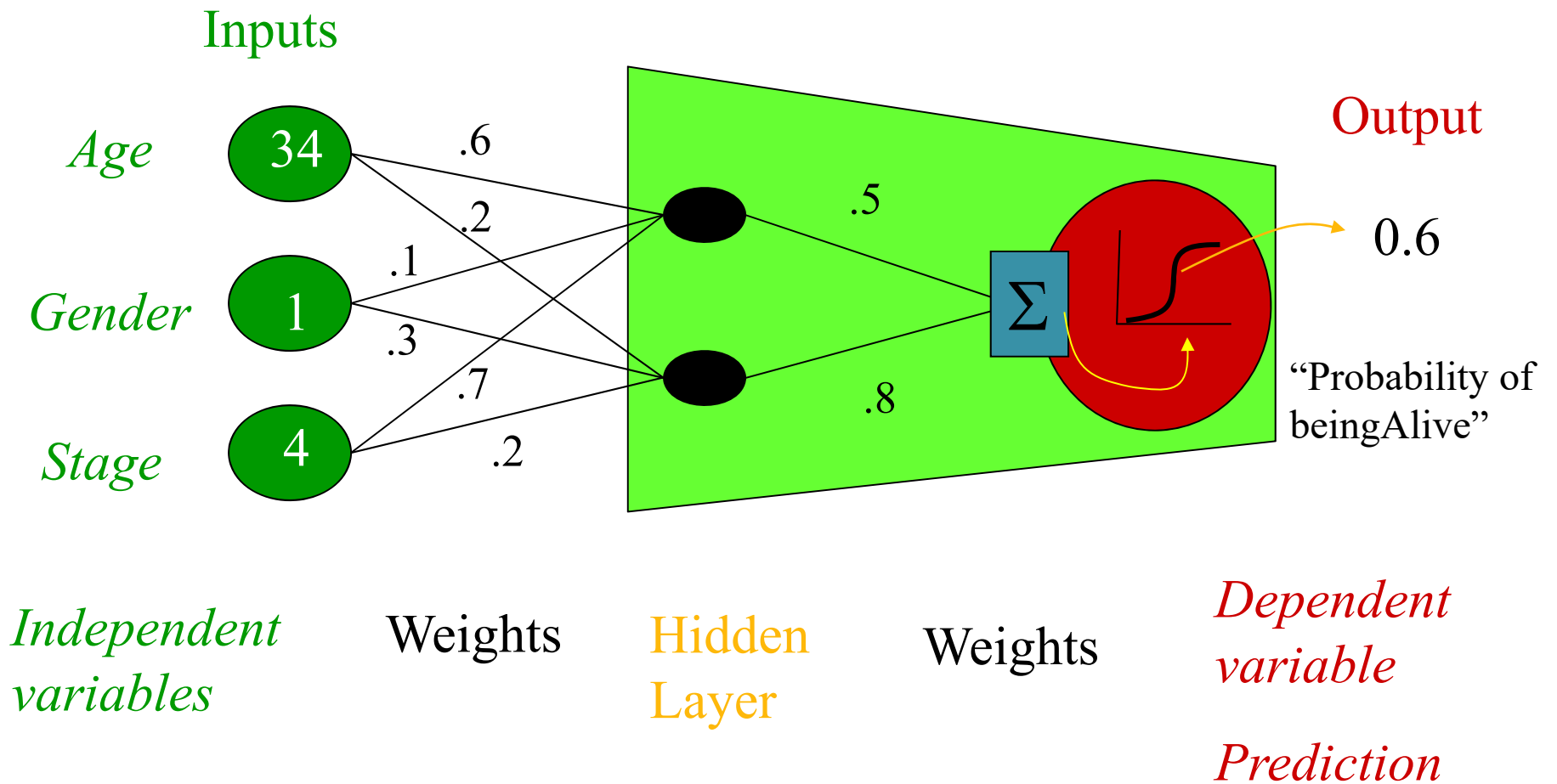
Neural Network Model



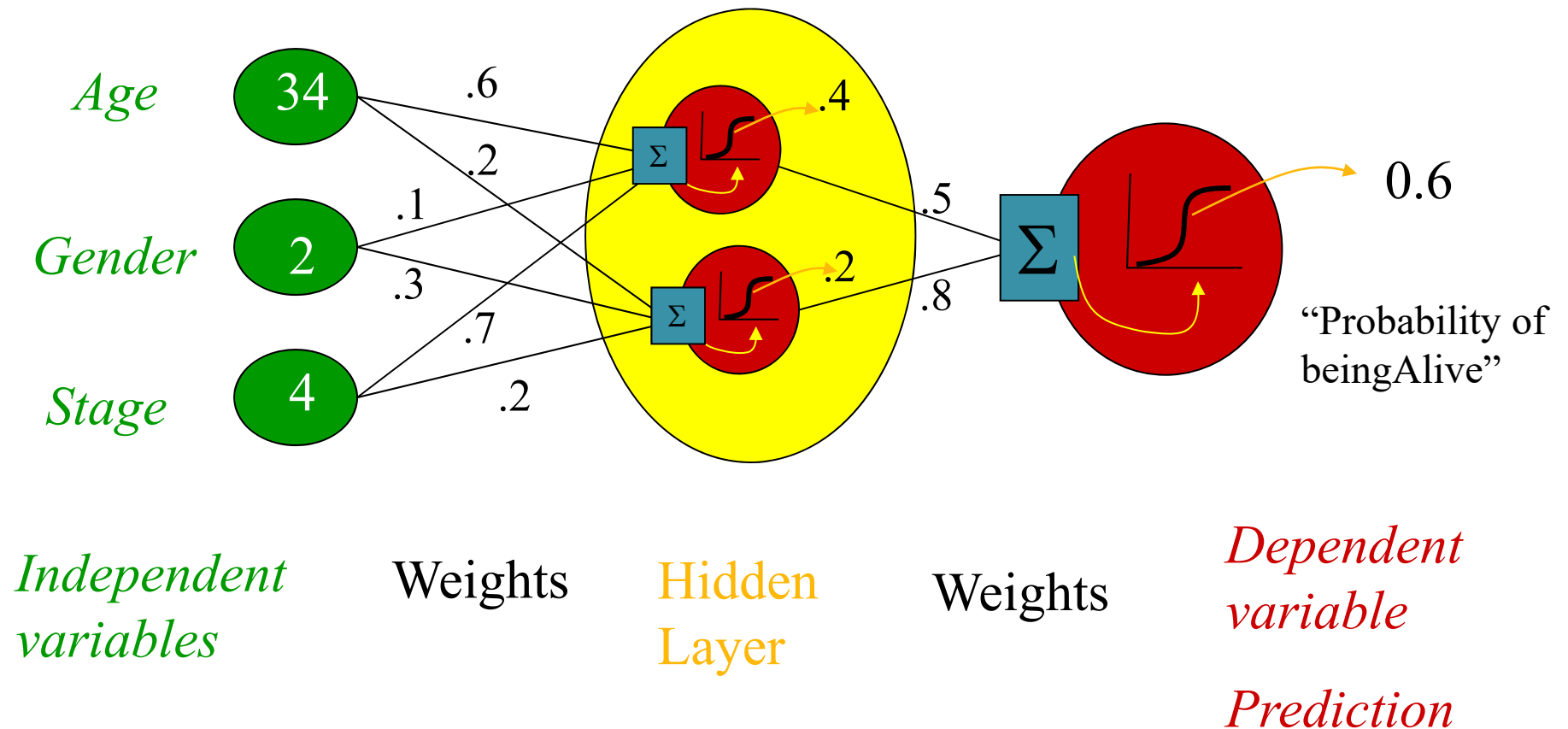
“Combined logistic models”





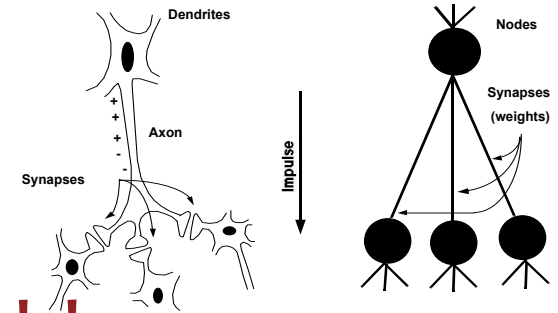


Not really, no target for hidden units...



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

Biological “Computation”

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

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

Artificial Computation

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

Neural Networks

Chalkboard

- Example: Neural Network w/1 Hidden Layer

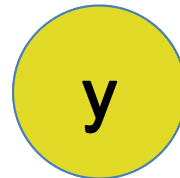
Decision Functions

Logistic Regression

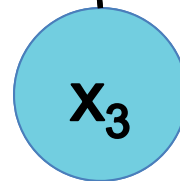
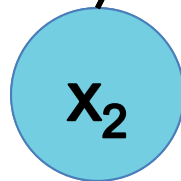
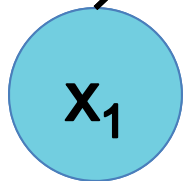
$$y = h_{\theta}(x) = \sigma(\theta^T x)$$

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

Output



Input



θ_1

θ_2

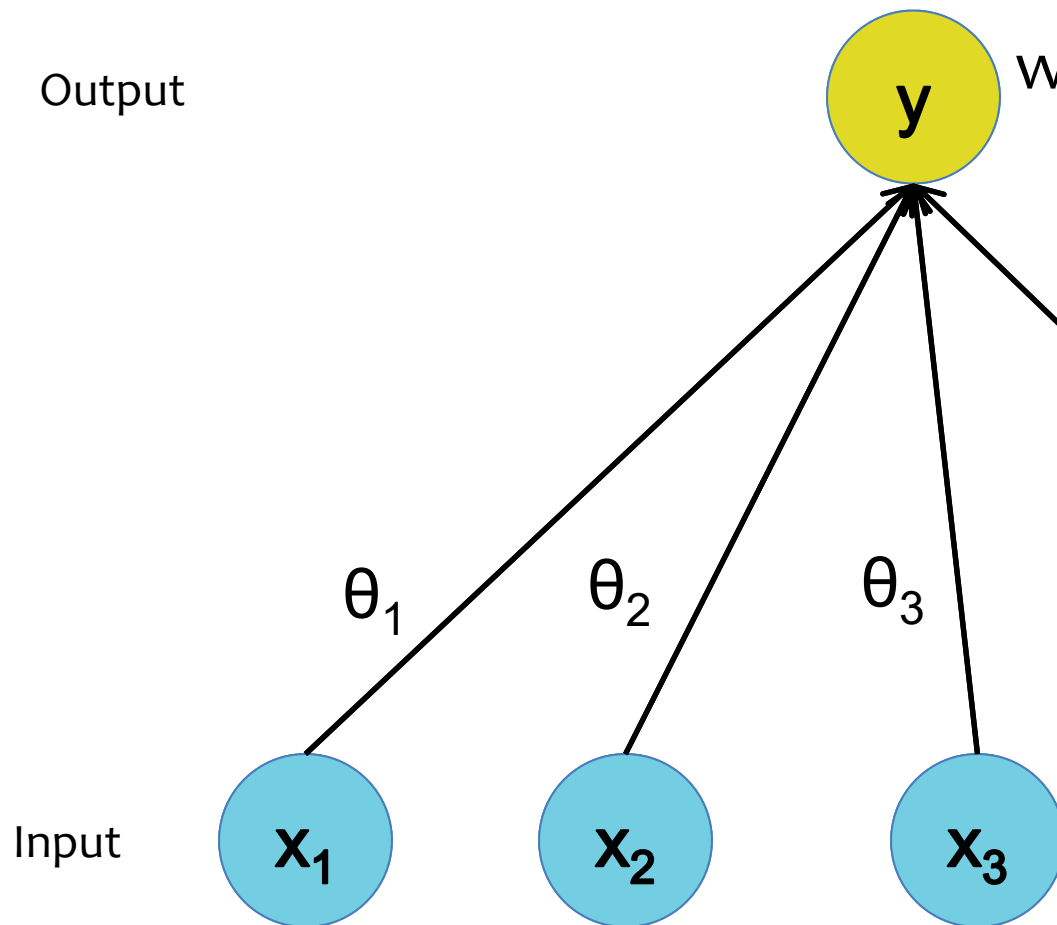
θ_3



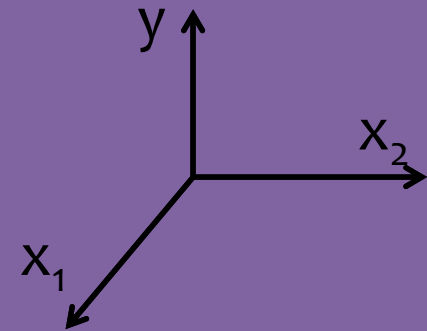
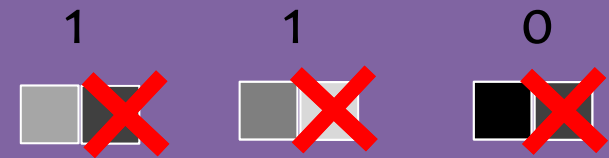
Decision Functions

Logistic Regression

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



In-Class Example



Neural Networks

Chalkboard

- 1D Example from linear regression to logistic regression
- 1D Example from logistic regression to a neural network

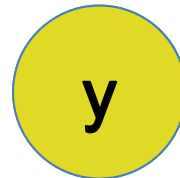
Decision Functions

Logistic Regression

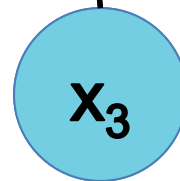
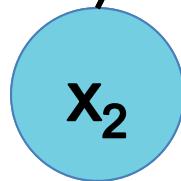
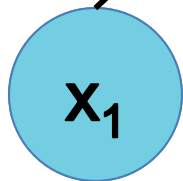
$$y = h_{\theta}(x) = \sigma(\theta^T x)$$

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

Output



Input



θ_1

θ_2

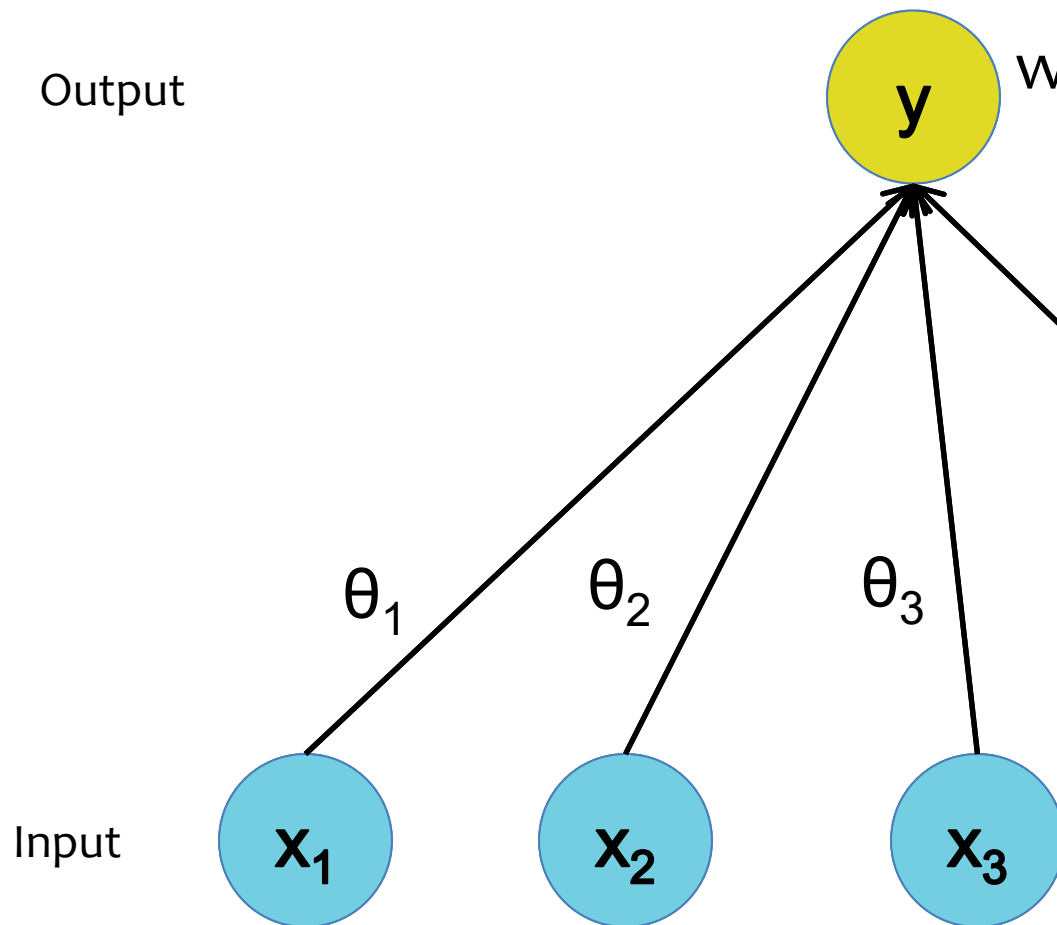
θ_3



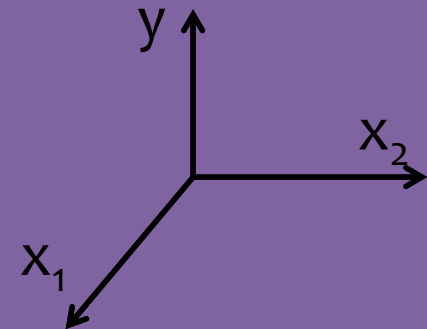
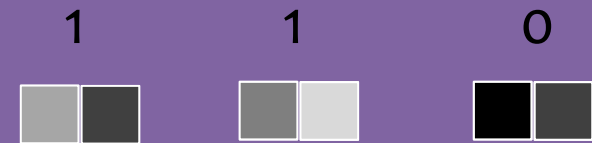
Decision Functions

Logistic Regression

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



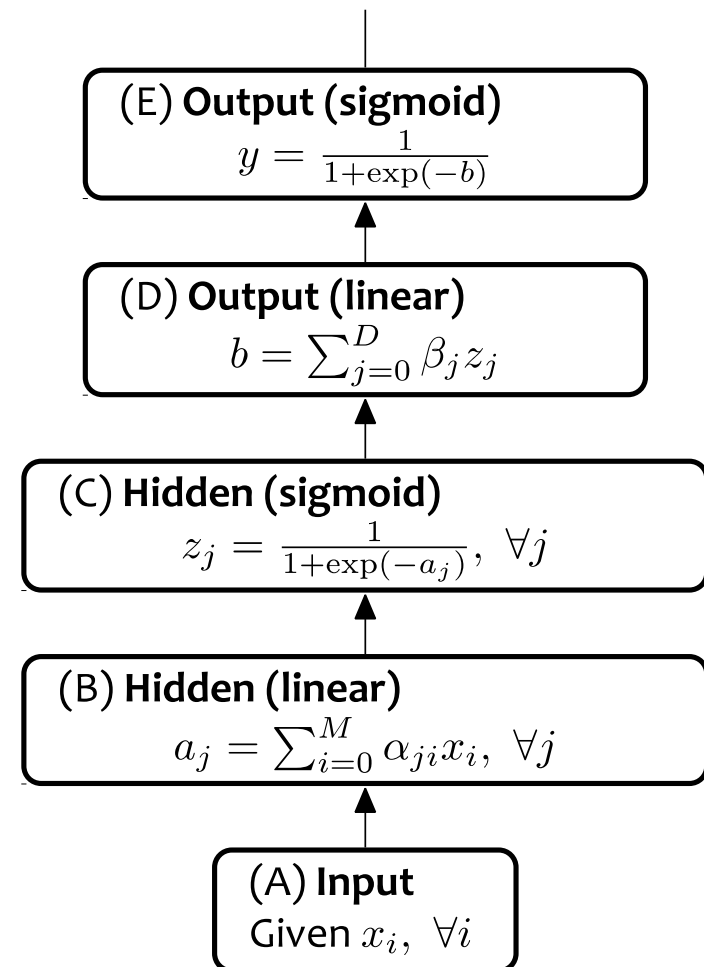
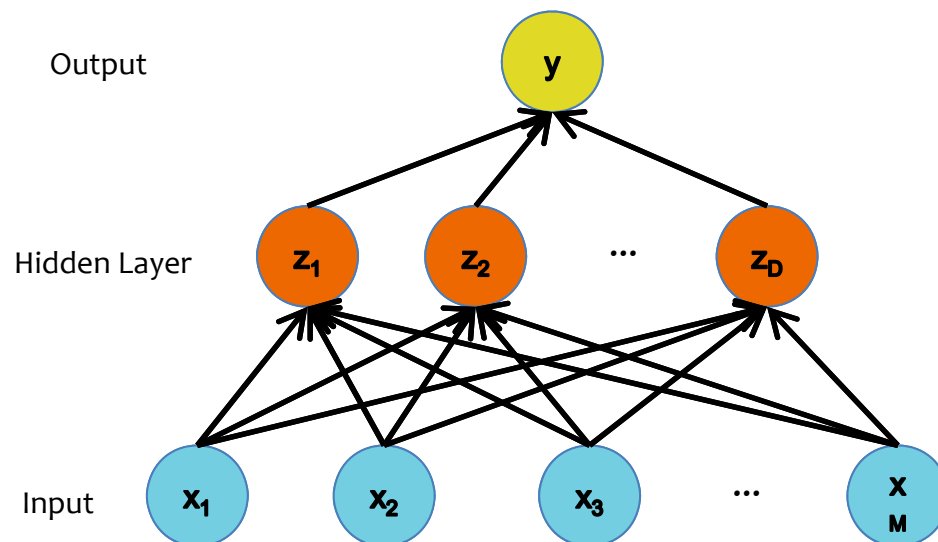
In-Class Example



Decision Functions

Neural Network

Neural Network for **Classification**



pr.mlkawse.219

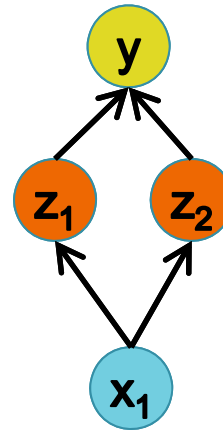
Neural Network Parameters

Question:

Suppose you are training a one-hidden layer neural network with sigmoid activations for binary classification.



True or False: There is a unique set of parameters that maximize the likelihood of the dataset above.



Answer:

$A = \text{True}$
 $B = \text{False}$

ARCHITECTURES

Neural Networks

Chalkboard

- Example: Neural Network w/2 Hidden Layers
- Example: Feed Forward Neural Network
(matrix form)

l

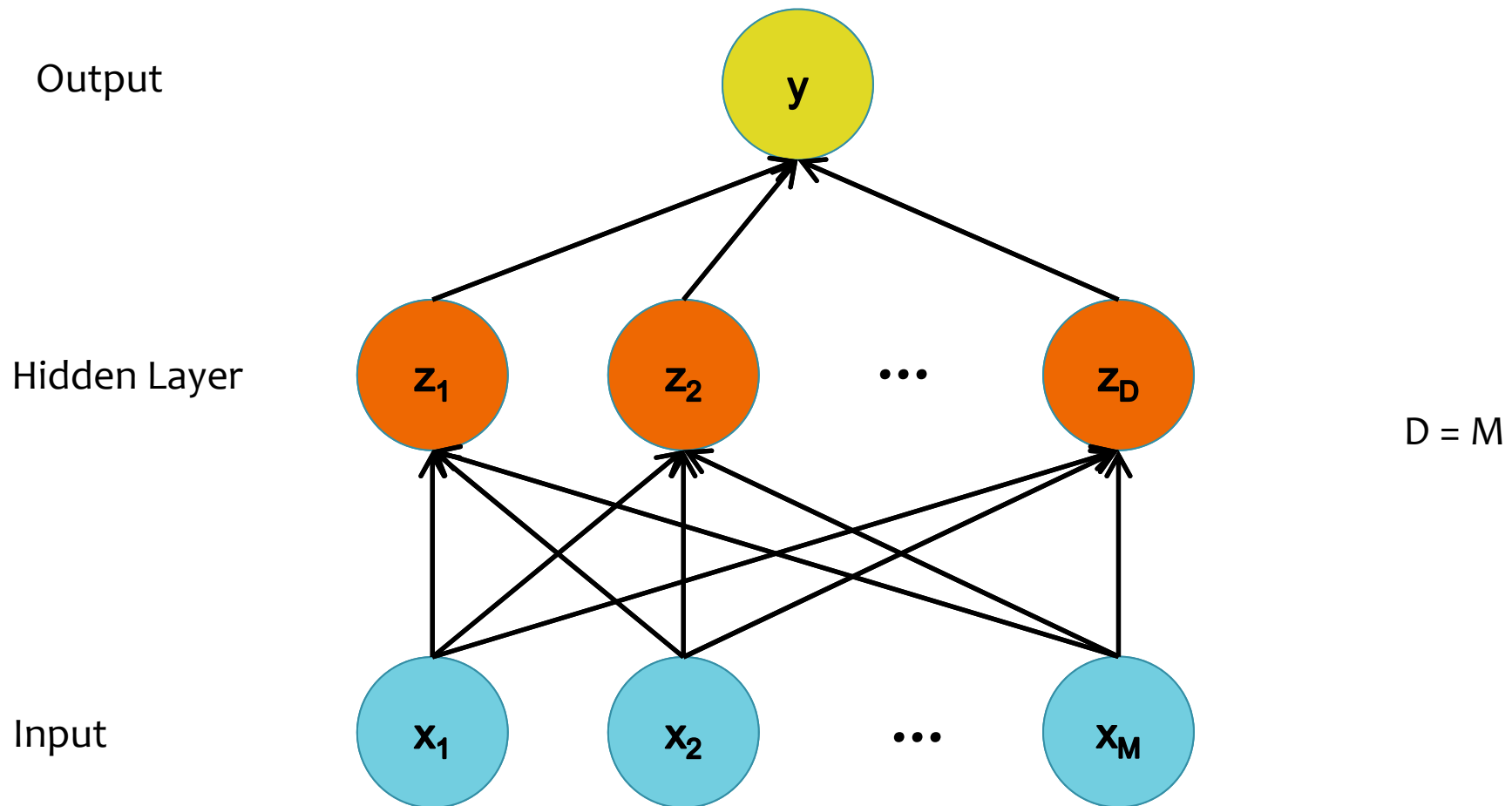
Neural Network Architectures

Even for a basic Neural Network, there are many design decisions to make:

1. # of hidden layers (depth)
2. # of units per hidden layer (width)
3. Type of activation function (nonlinearity)
4. Form of objective function
5. How to initialize parameters

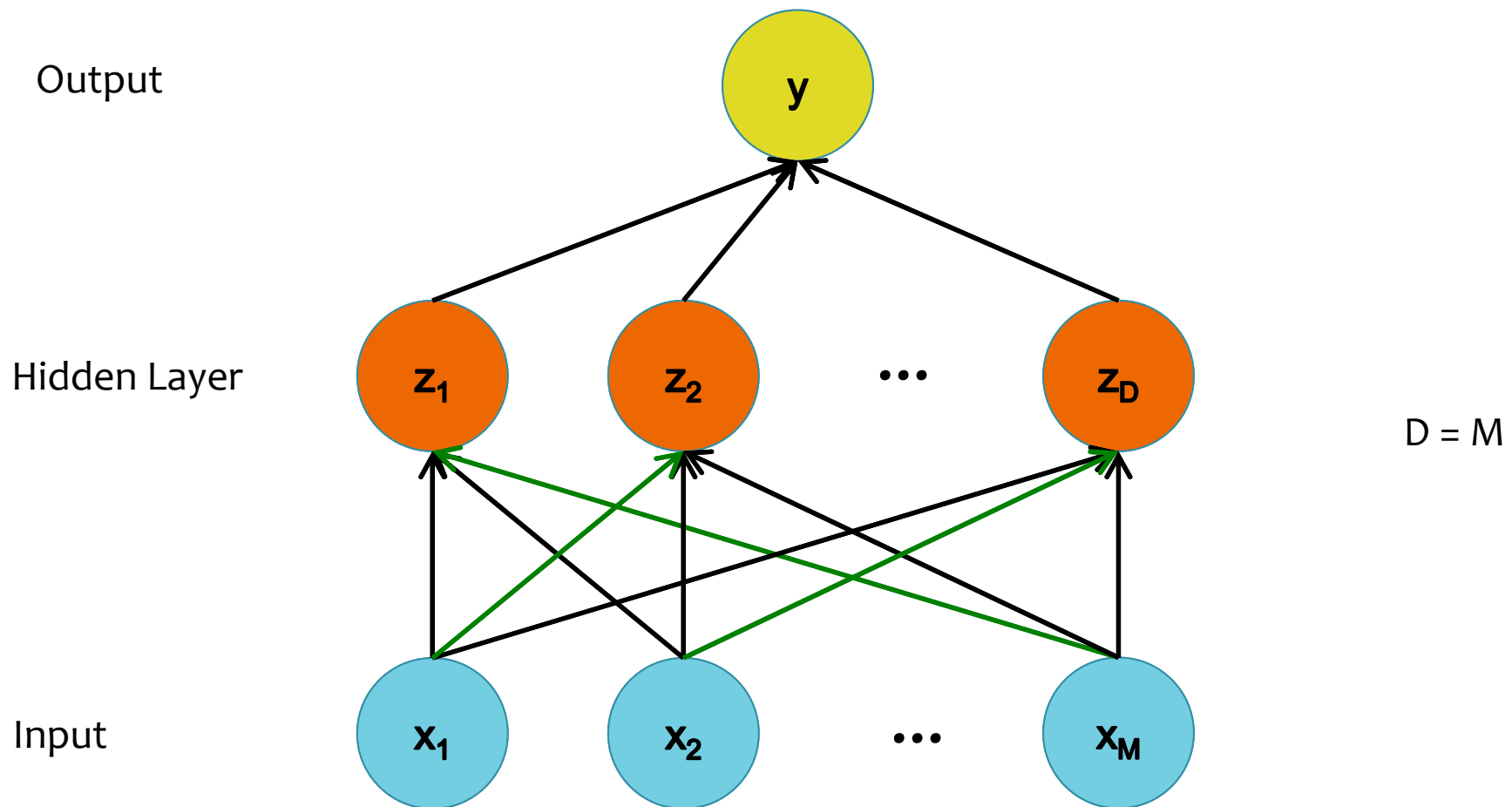
Building a Neural Net

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



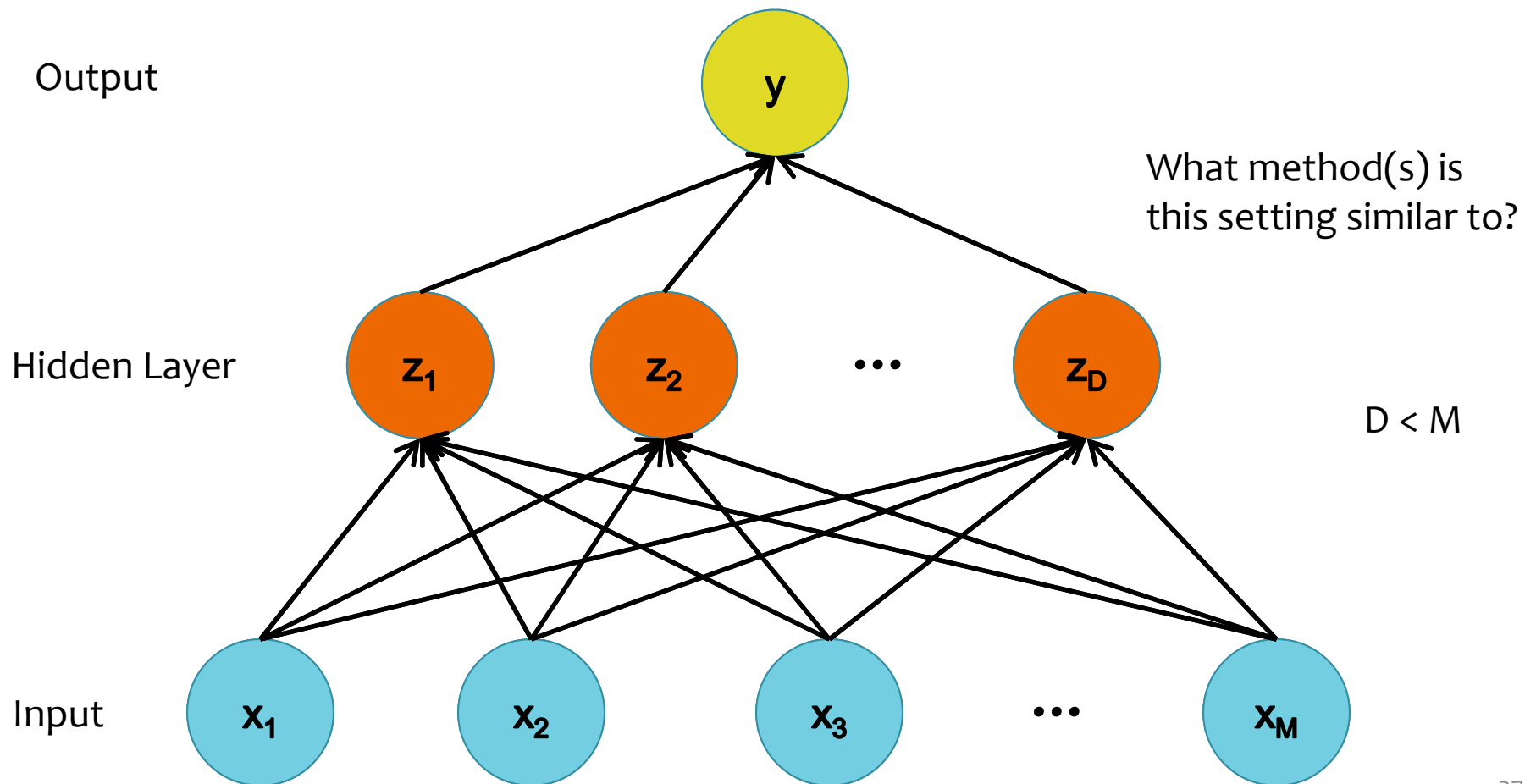
Building a Neural Net

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



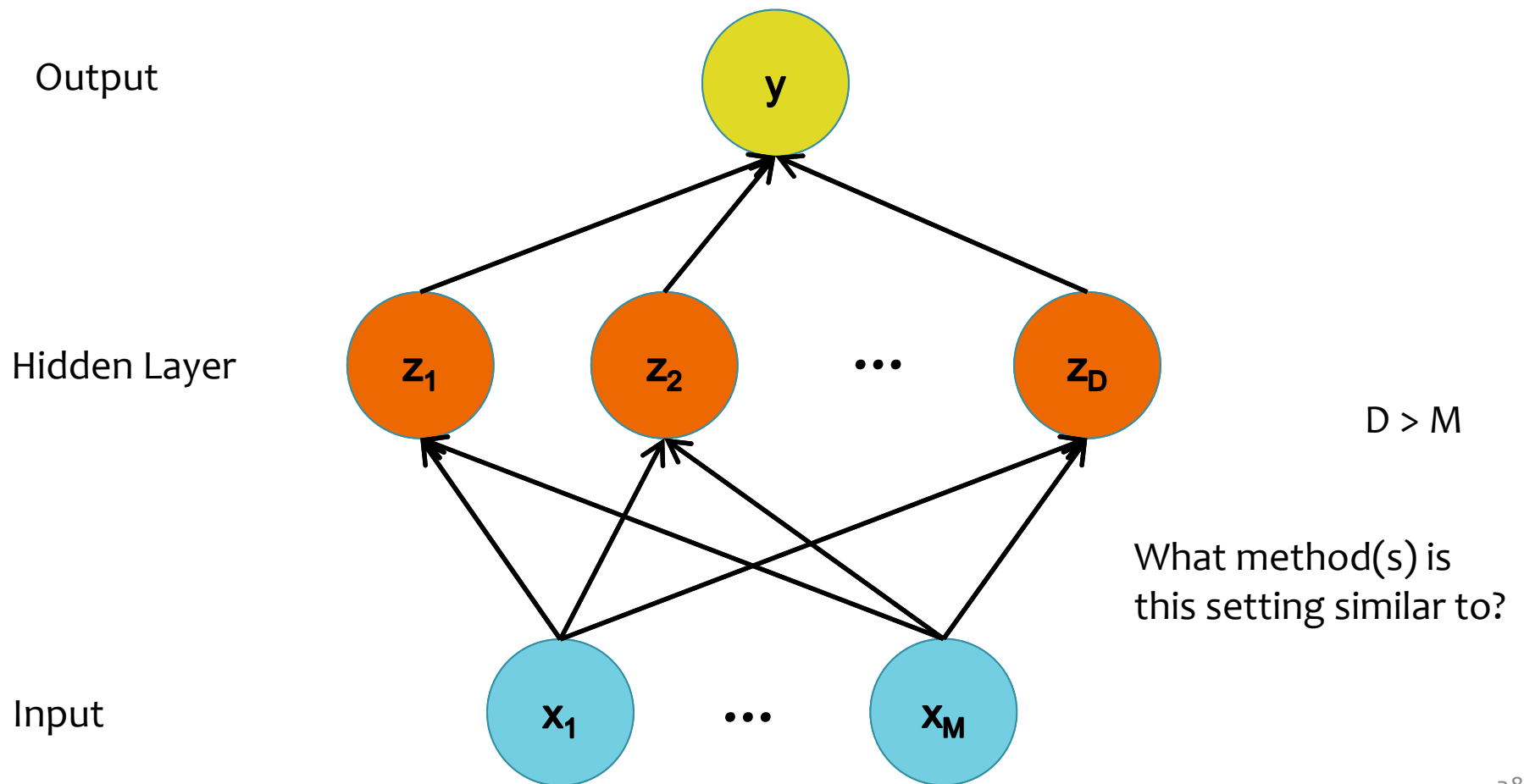
Building a Neural Net

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



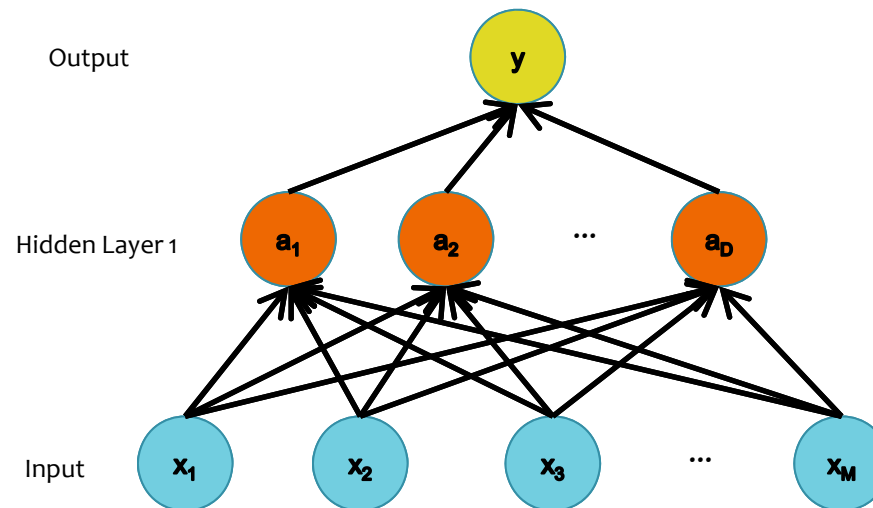
Building a Neural Net

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



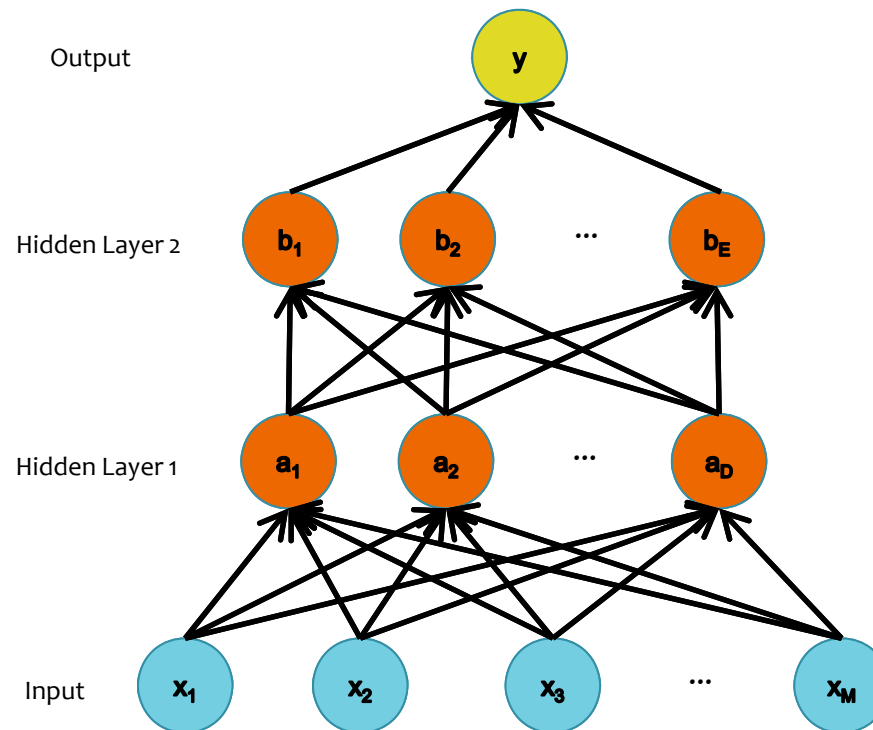
Deeper Networks

Q: How many layers should we use?



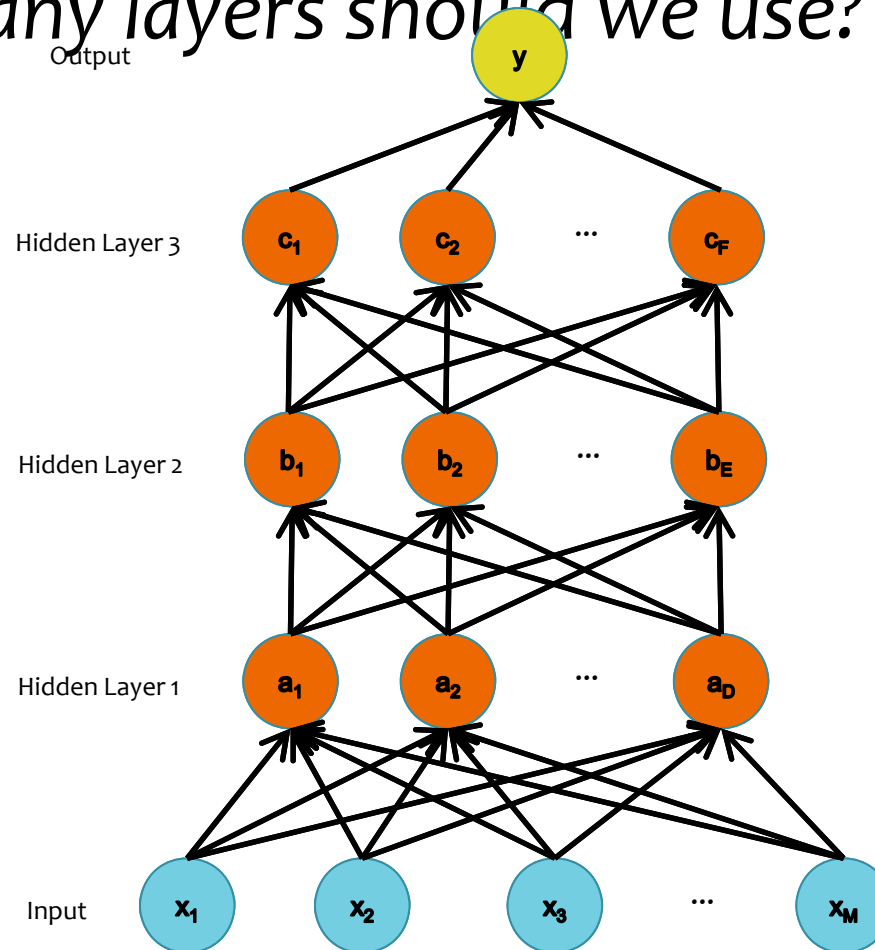
Deeper Networks

Q: How many layers should we use?



Deeper Networks

Q: How many layers should we use?



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.

Different Levels of Abstraction

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

Feature representation



3rd layer
“Objects”



2nd layer
“Object parts”



1st layer
“Edges”



Pixels

Different Levels of Abstraction

Face Recognition:

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

Feature representation



3rd layer
“Objects”



2nd layer
“Object parts”

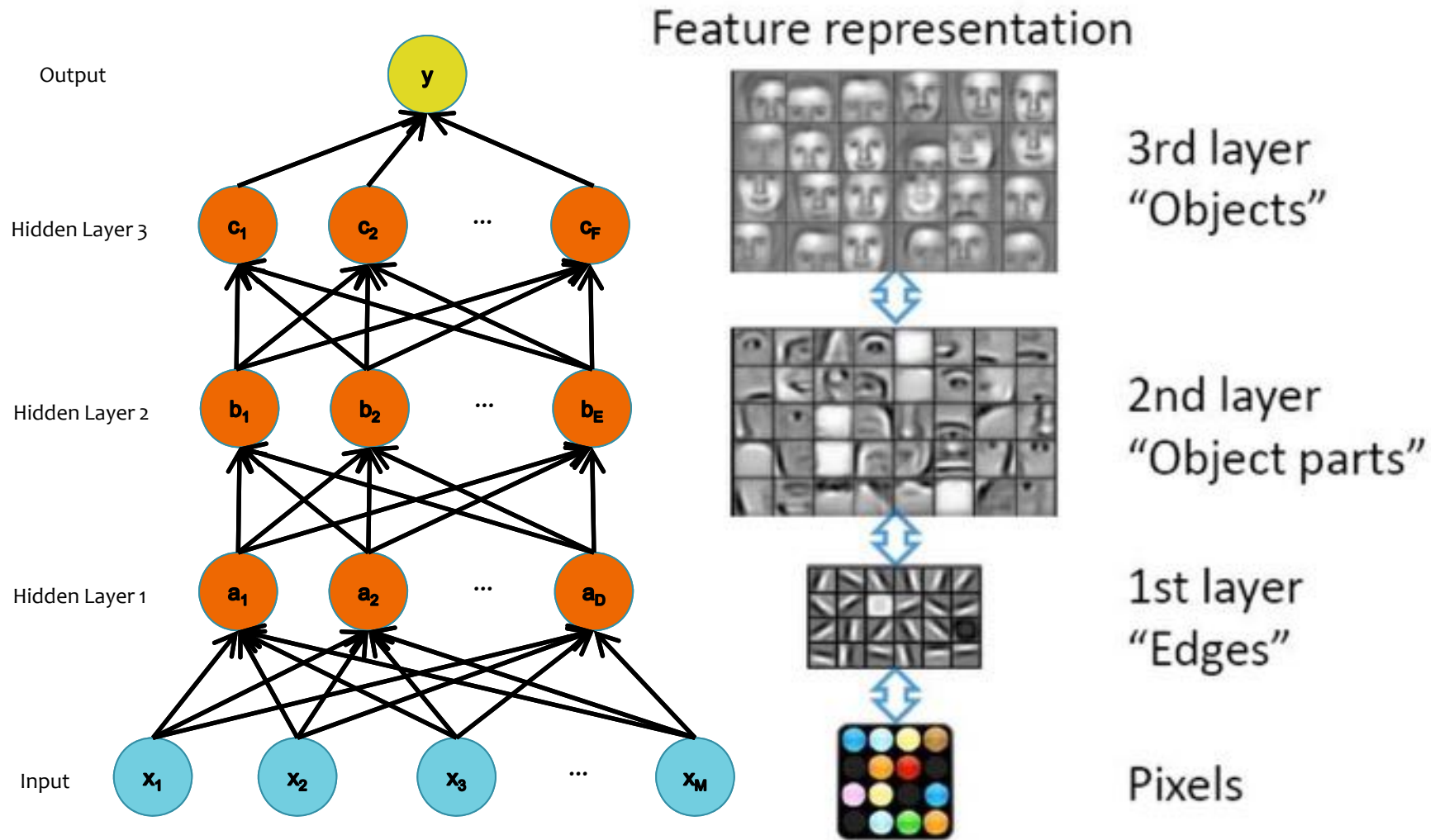


1st layer
“Edges”



Pixels

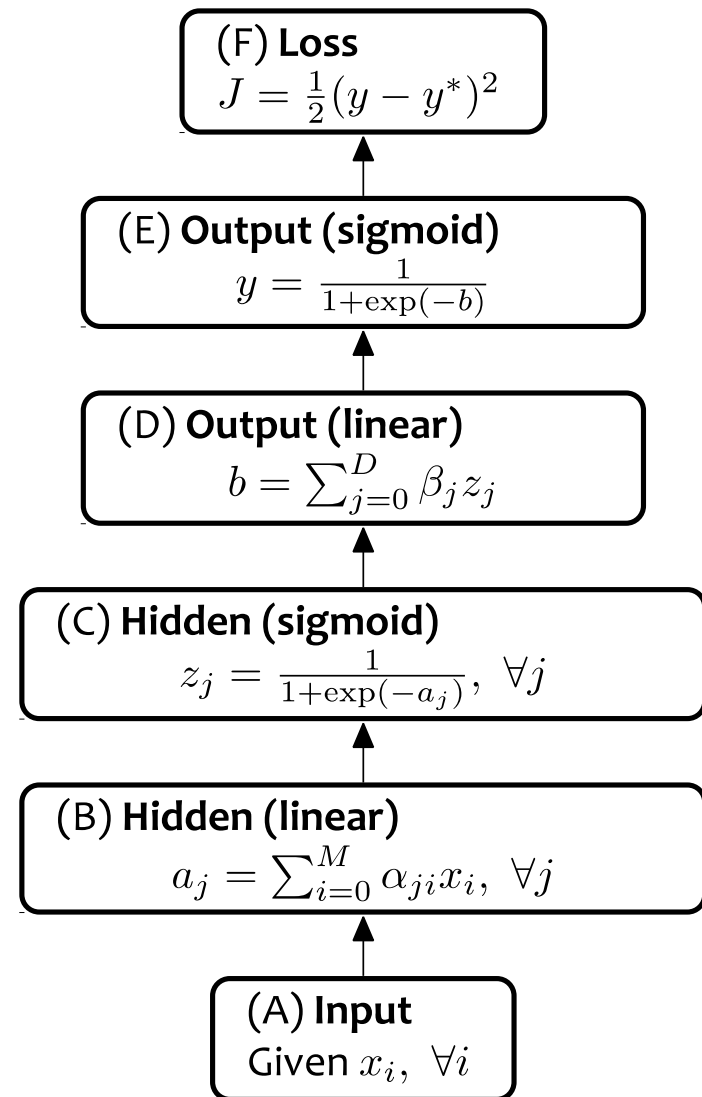
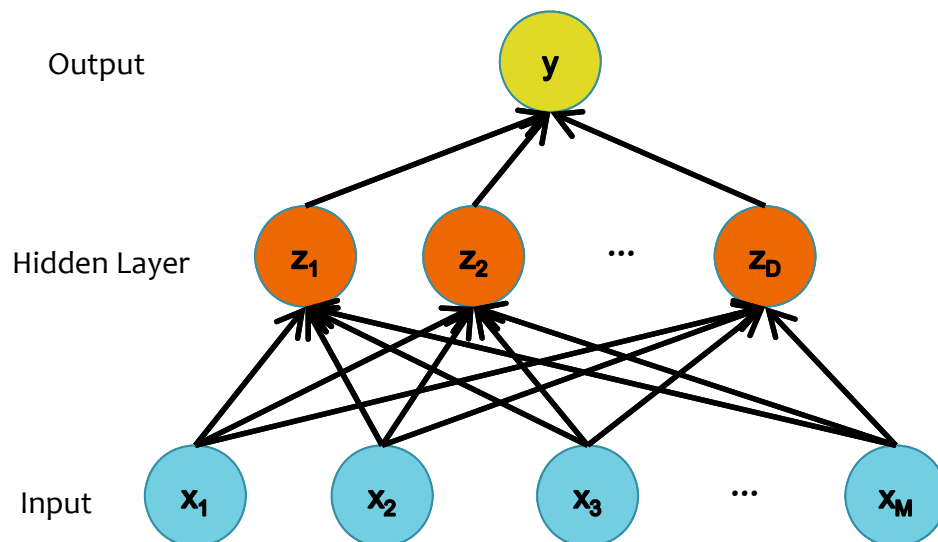
Different Levels of Abstraction



Example from Honglak Lee (NIPS 2010)

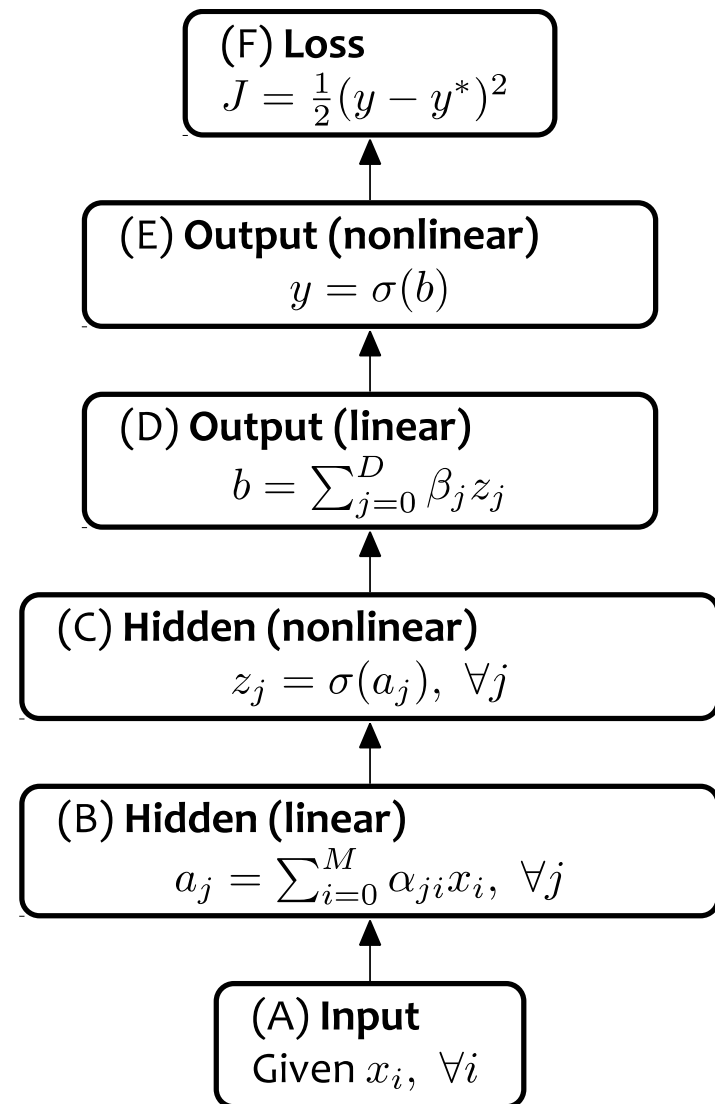
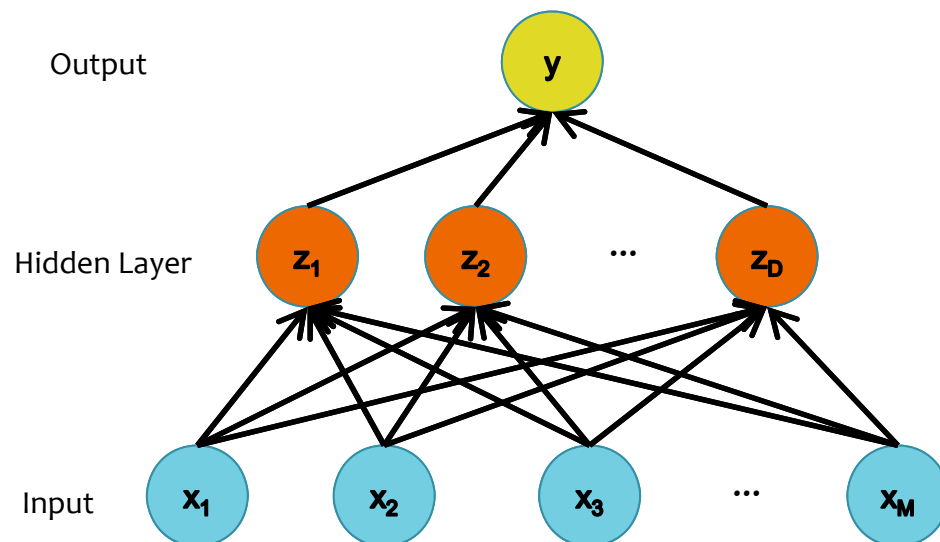
Activation Functions

Neural Network with sigmoid
activation functions



Activation Functions

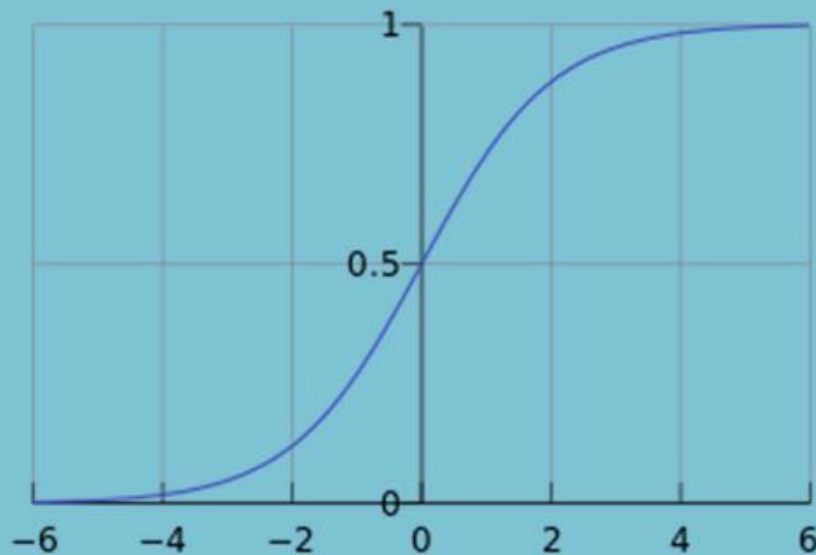
Neural Network with arbitrary nonlinear activation functions



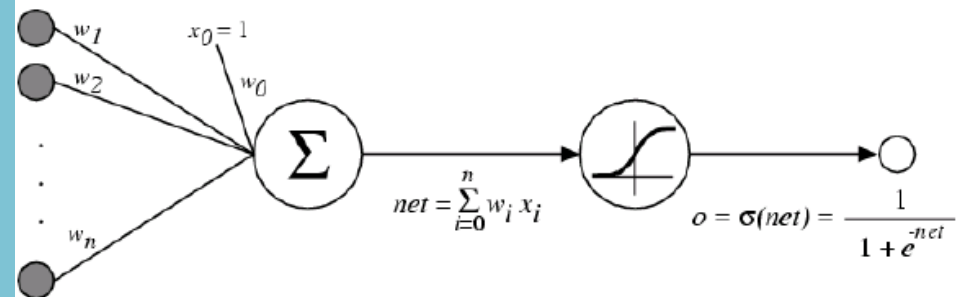
Activation Functions

Sigmoid / Logistic Function

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

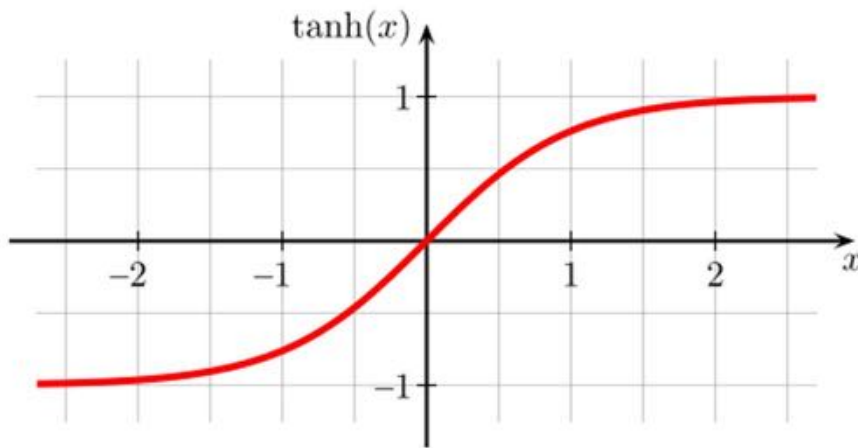


So far, we've assumed that the activation function (nonlinearity) is always the sigmoid function...



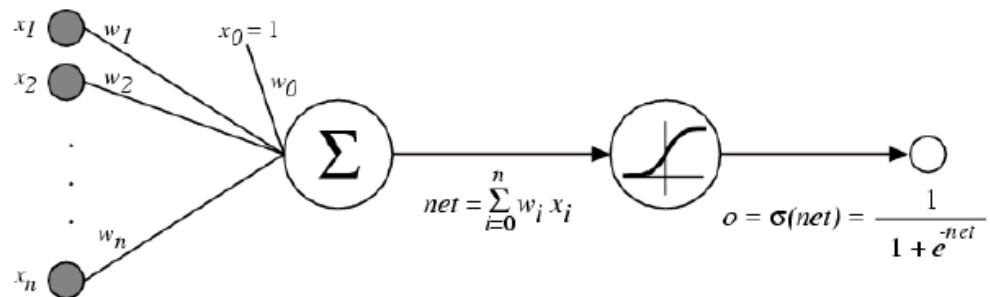
Activation Functions

- A new change: modifying the nonlinearity
 - The logistic is not widely used in modern ANNs



Alternate 1:
tanh

Like logistic function but
shifted to range $[-1, +1]$



Understanding the difficulty of training deep feedforward neural networks

AI Stats 2010

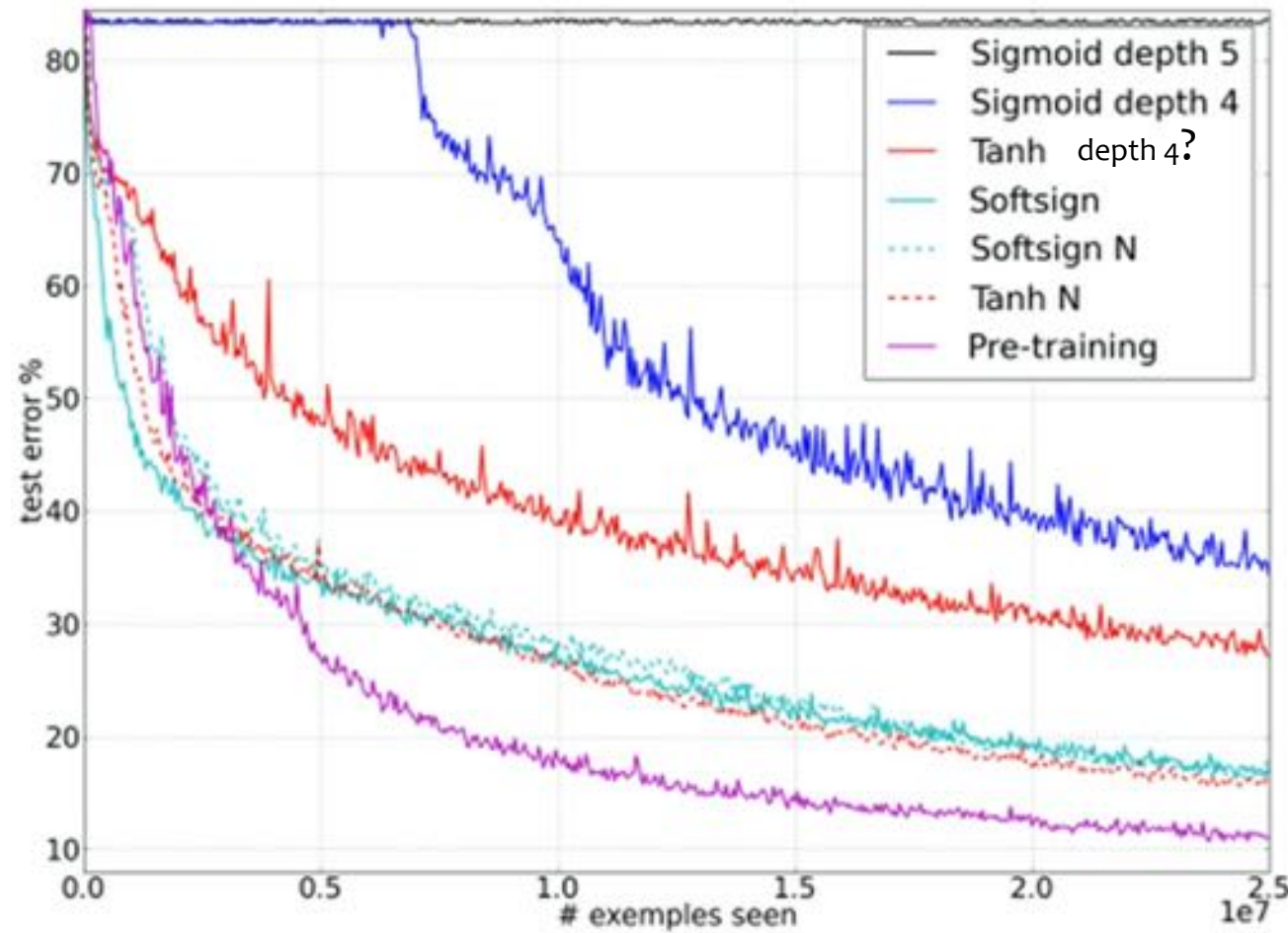
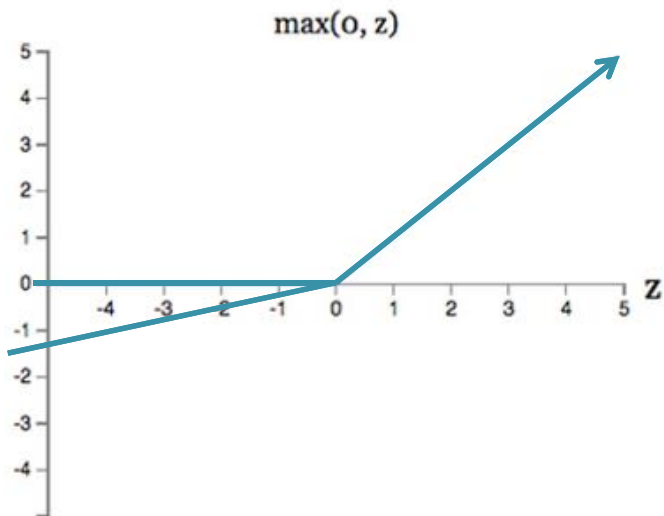


Figure from Glorot & Bentio (2010)

Activation Functions

- A new change: modifying the nonlinearity
 - reLU often used in vision tasks

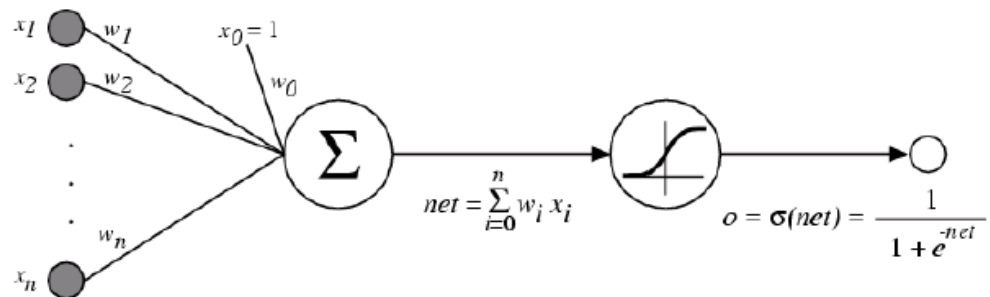


$$\max(0, w \cdot x + b).$$

Alternate 2: rectified linear unit

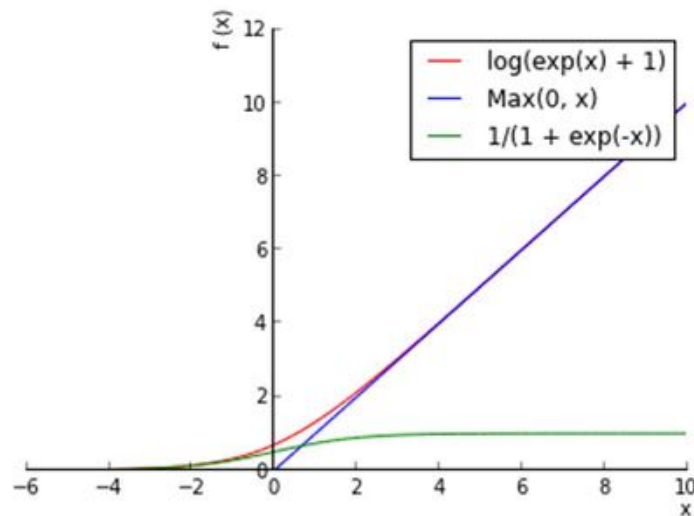
Linear with a cutoff at zero

(Implementation: clip the gradient when you pass zero)



Activation Functions

- A new change: modifying the nonlinearity
 - reLU often used in vision tasks



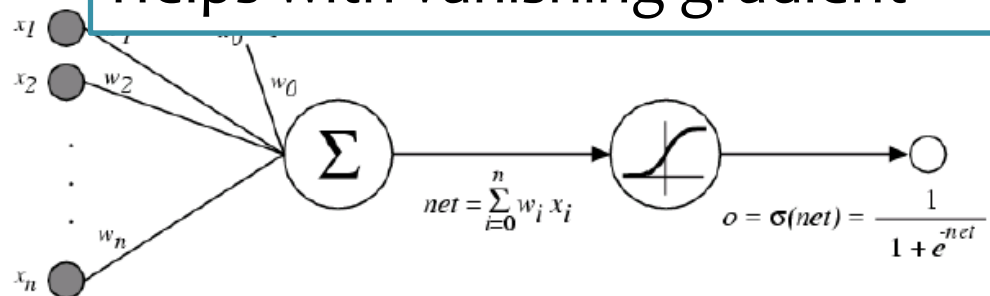
Alternate 2: rectified linear unit

Soft version: $\log(\exp(x)+1)$

Doesn't saturate (at one end)

Sparsifies outputs

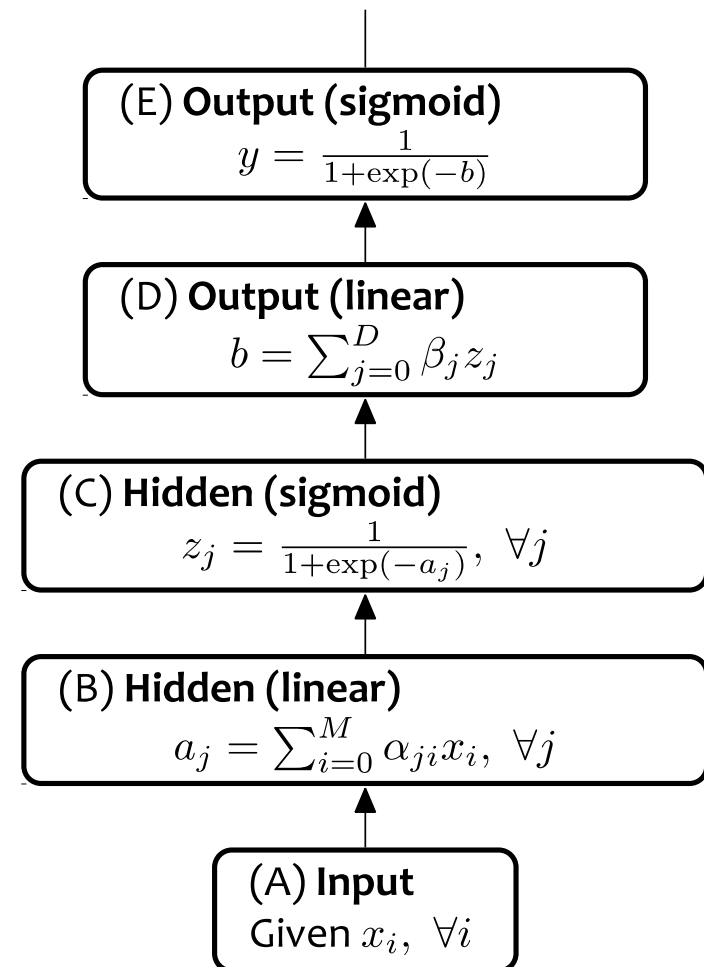
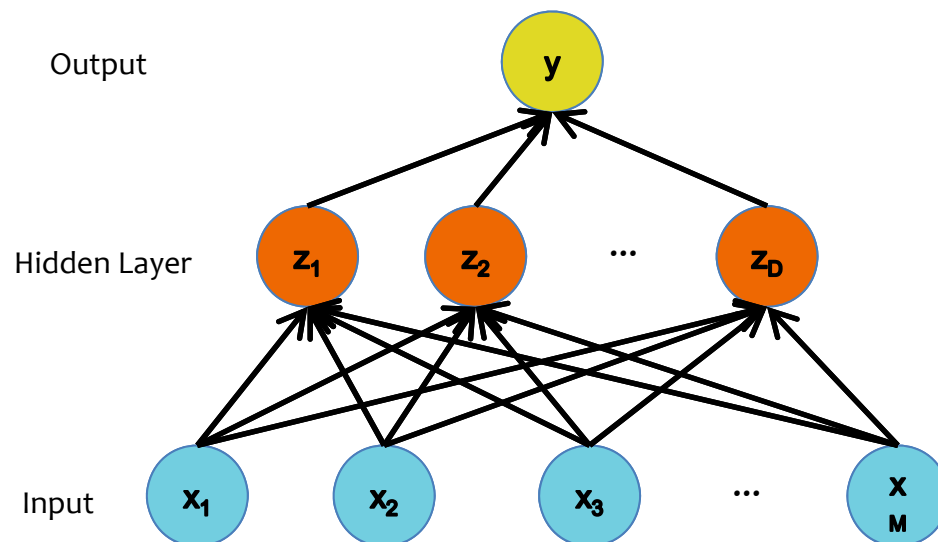
Helps with vanishing gradient



Decision Functions

Neural Network

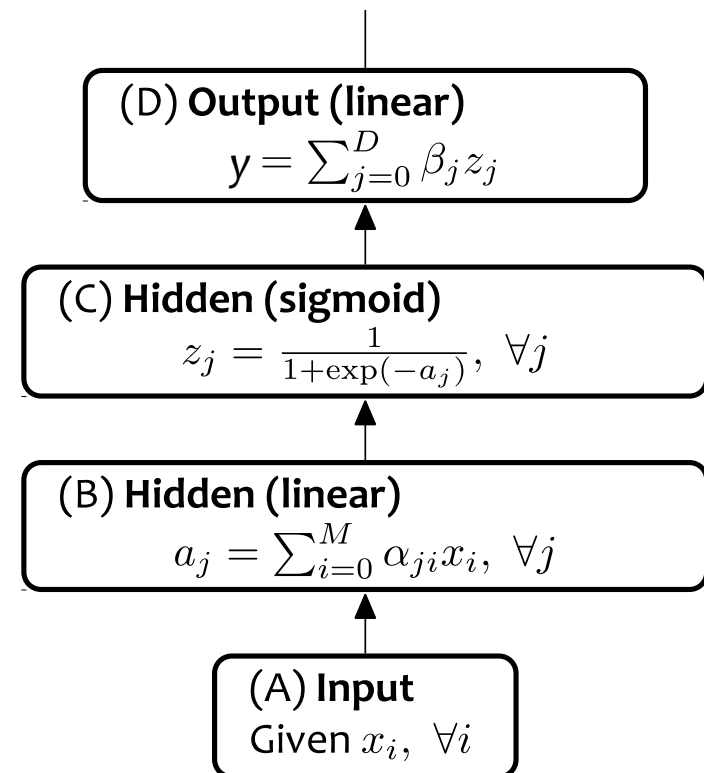
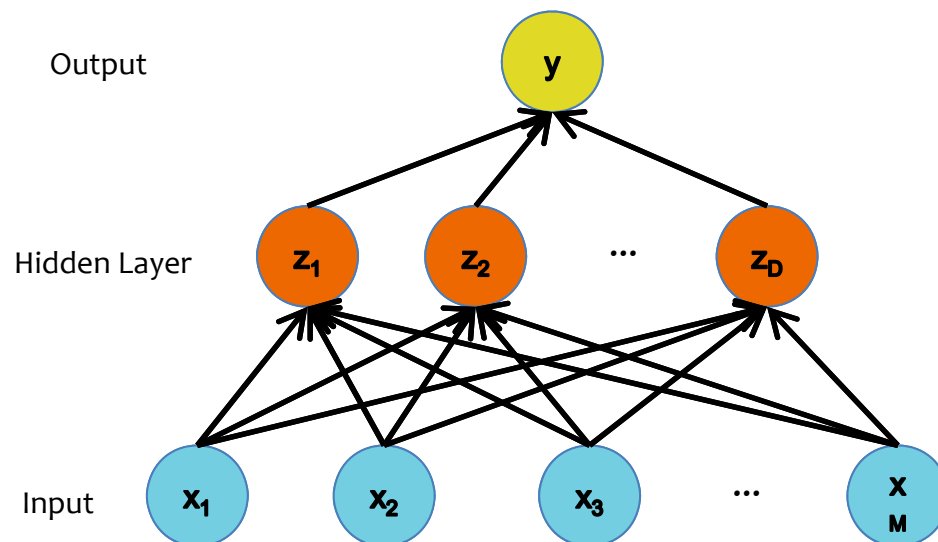
Neural Network for **Classification**



Decision Functions

Neural Network

Neural Network for Regression



Objective Functions for NNs

1. Quadratic Loss:
 - the same objective as Linear Regression
 - i.e. mean squared error
2. Cross-Entropy:
 - the same objective as Logistic Regression
 - i.e. negative log likelihood
 - This requires probabilities, so we add an additional “softmax” layer at the end of our network

	Forward	Backward
Quadratic	$J = \frac{1}{2}(y - y^*)^2$	$\frac{dJ}{dy} = y - y^*$
Cross Entropy	$J = y^* \log(y) + (1 - y^*) \log(1 - y)$	$\frac{dJ}{dy} = y^* \frac{1}{y} + (1 - y^*) \frac{1}{y - 1}$

Objective Functions for NNs

Cross-entropy vs. Quadratic loss

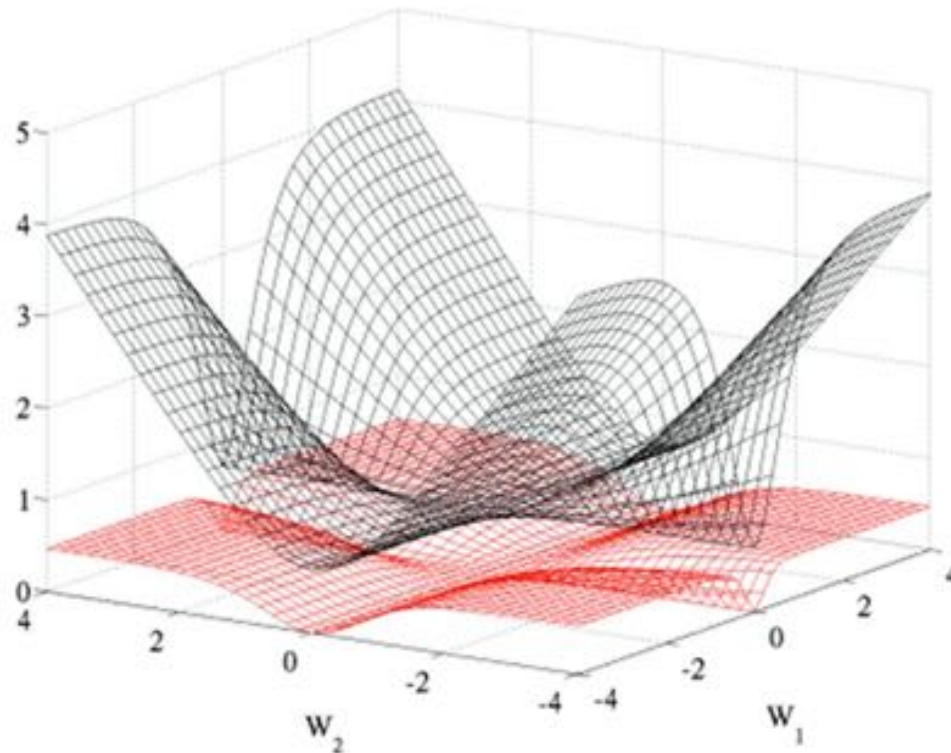
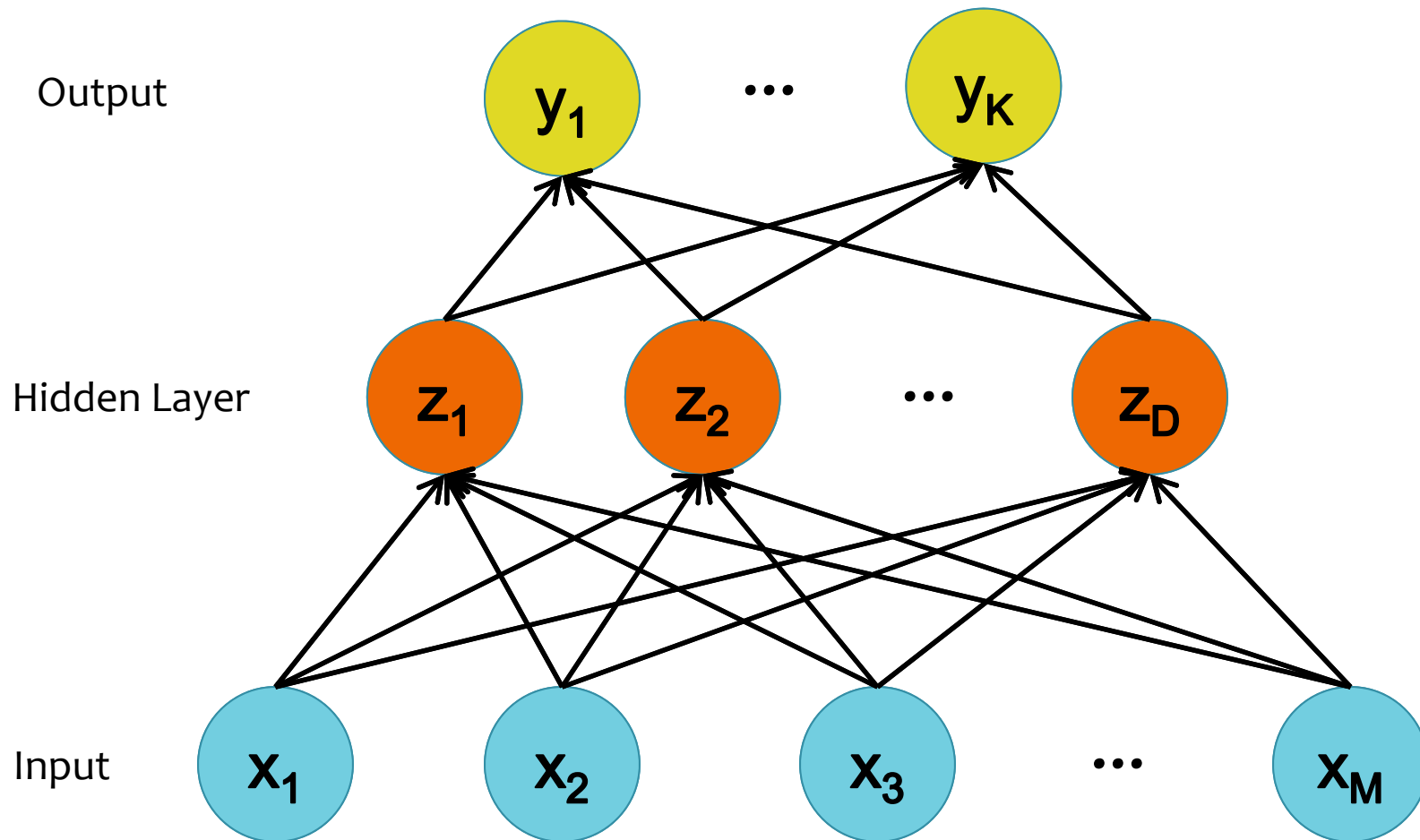


Figure 5: Cross entropy (black, surface on top) and quadratic (red, bottom surface) cost as a function of two weights (one at each layer) of a network with two layers, w_1 respectively on the first layer and w_2 on the second, output layer.

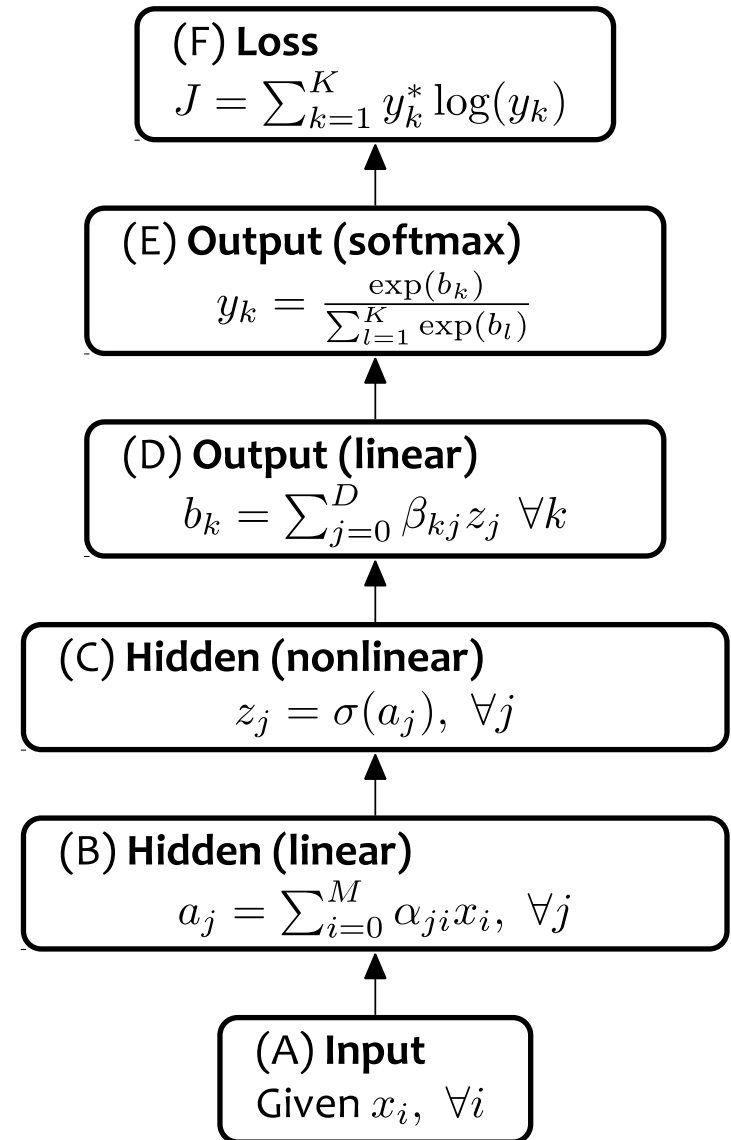
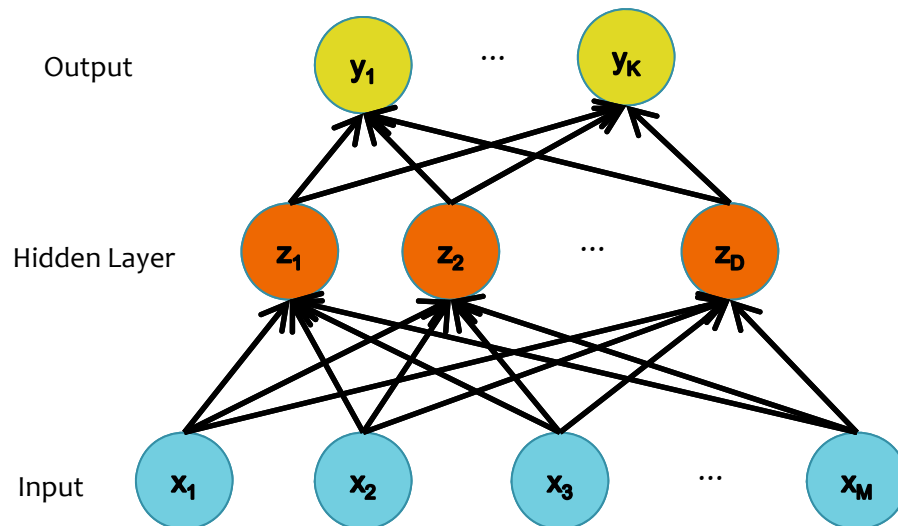
Multi-Class Output



Multi-Class Output

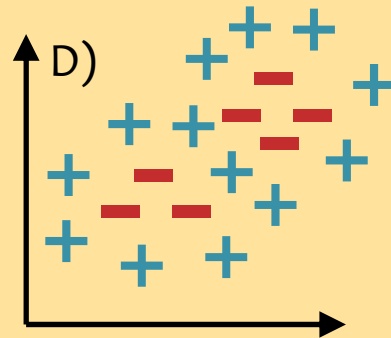
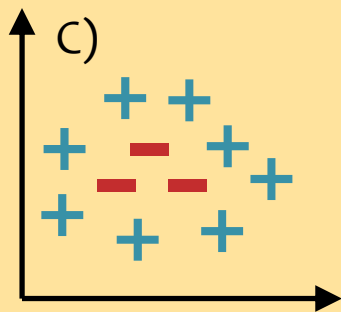
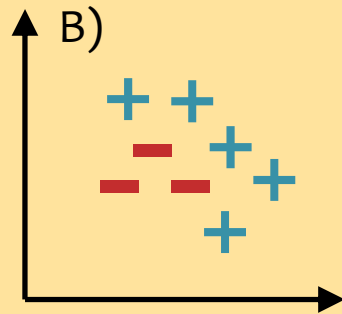
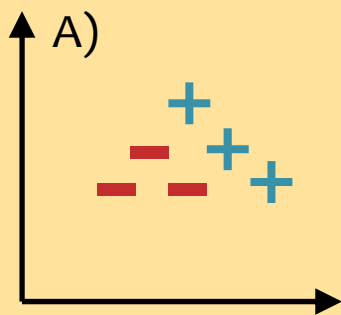
Softmax:

$$y_k = \frac{\exp(b_k)}{\sum_{l=1}^K \exp(b_l)}$$

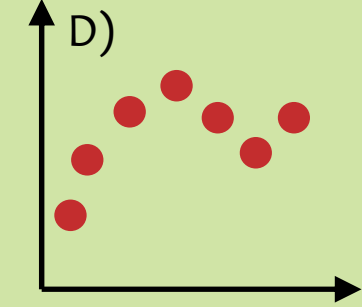
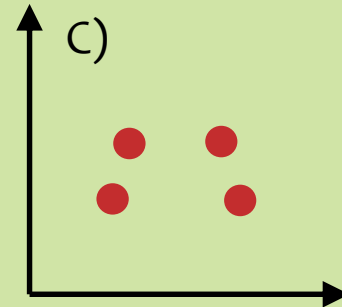
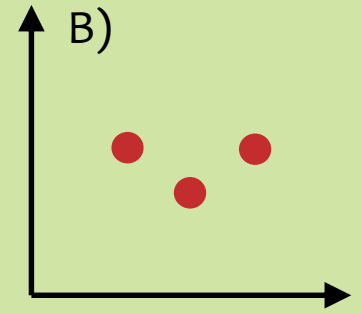
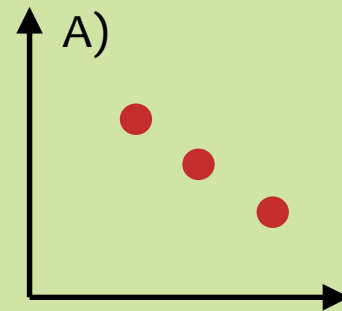


Neural Network Errors

Question A: For which of the datasets below does there exist a one-hidden layer neural network that achieves zero *classification* error? **Select all that apply.**

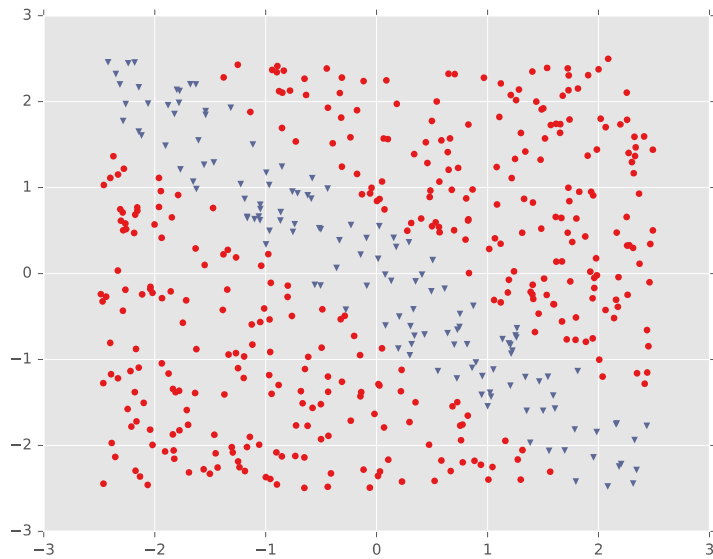


Question B: For which of the datasets below does there exist a one-hidden layer neural network for regression that achieves *nearly* zero MSE? **Select all that apply.**

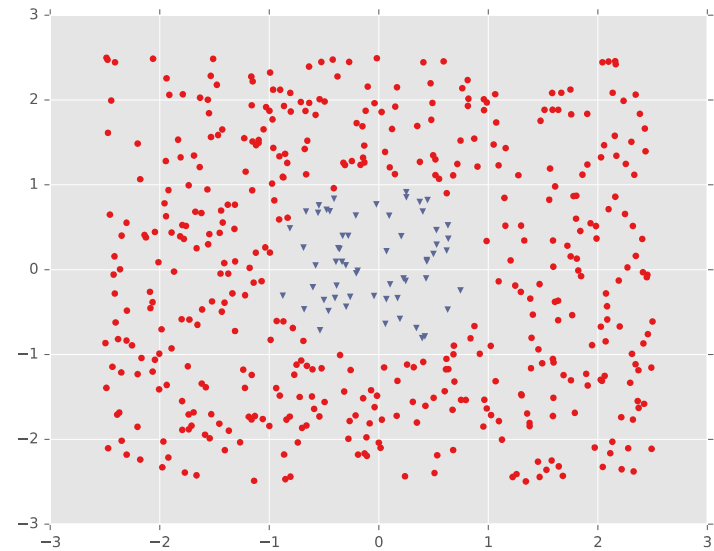


DECISION BOUNDARY EXAMPLES

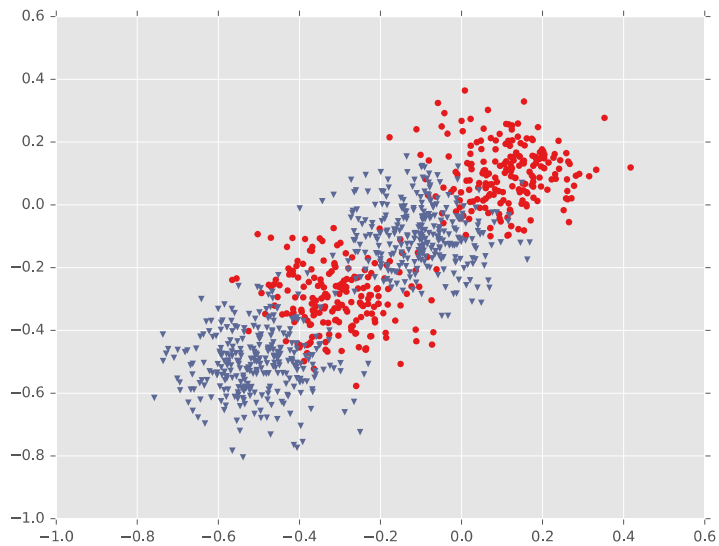
Example #1: Diagonal Band



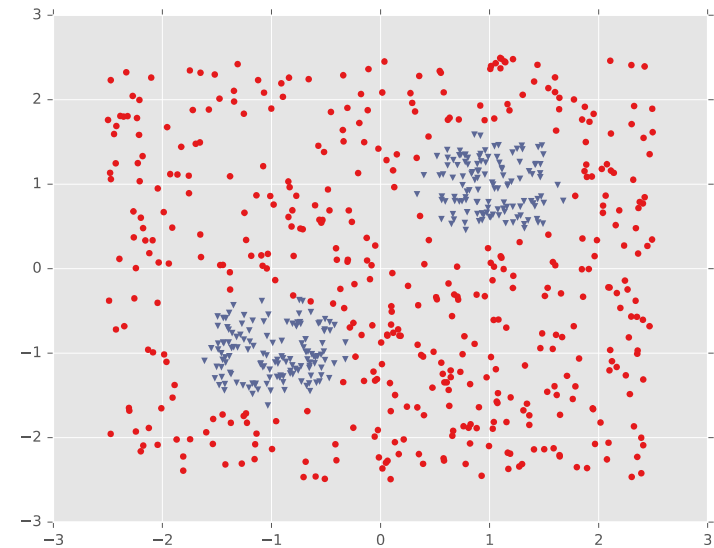
Example #2: One Pocket



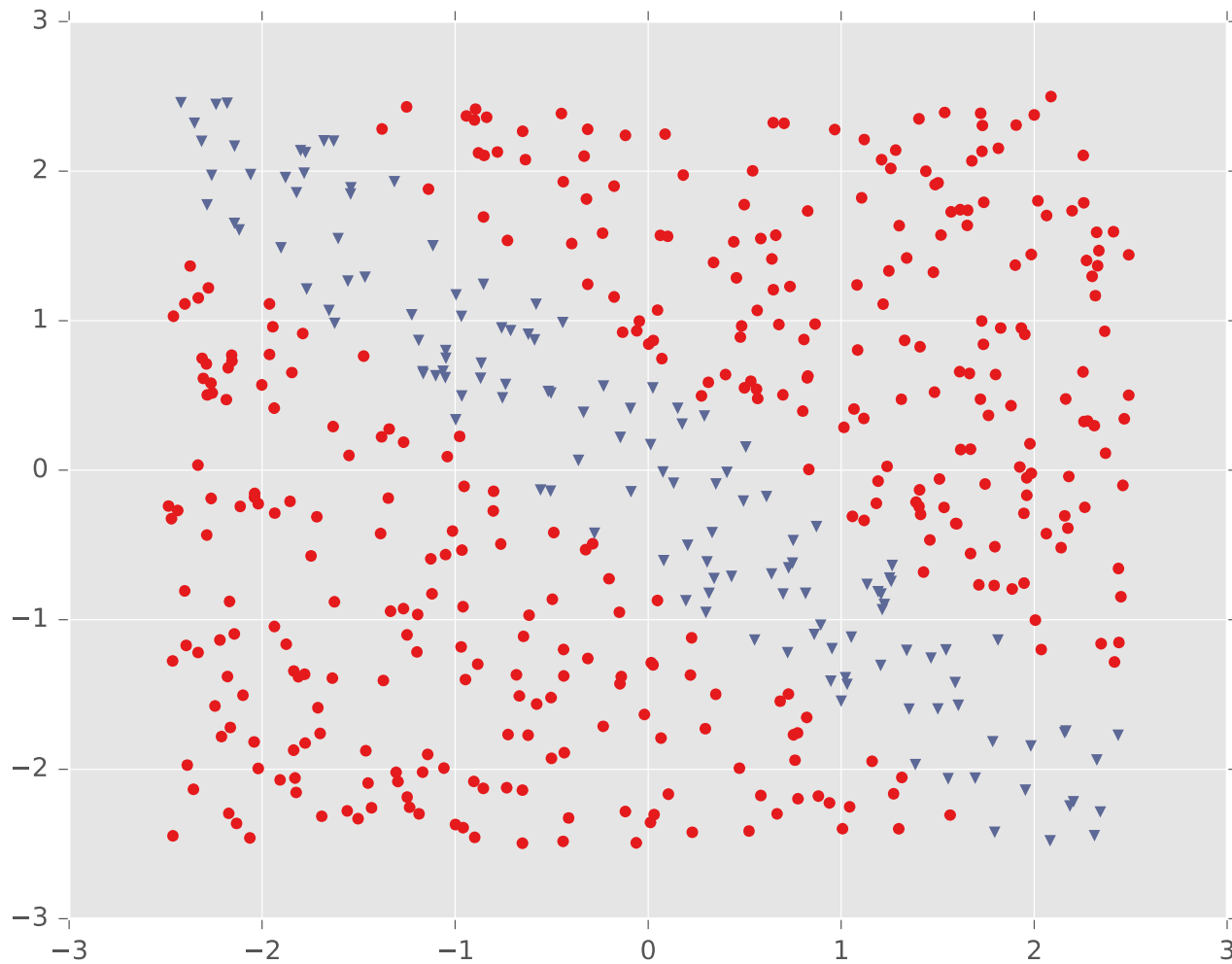
Example #3: Four Gaussians



Example #4: Two Pockets



Example #1: Diagonal Band

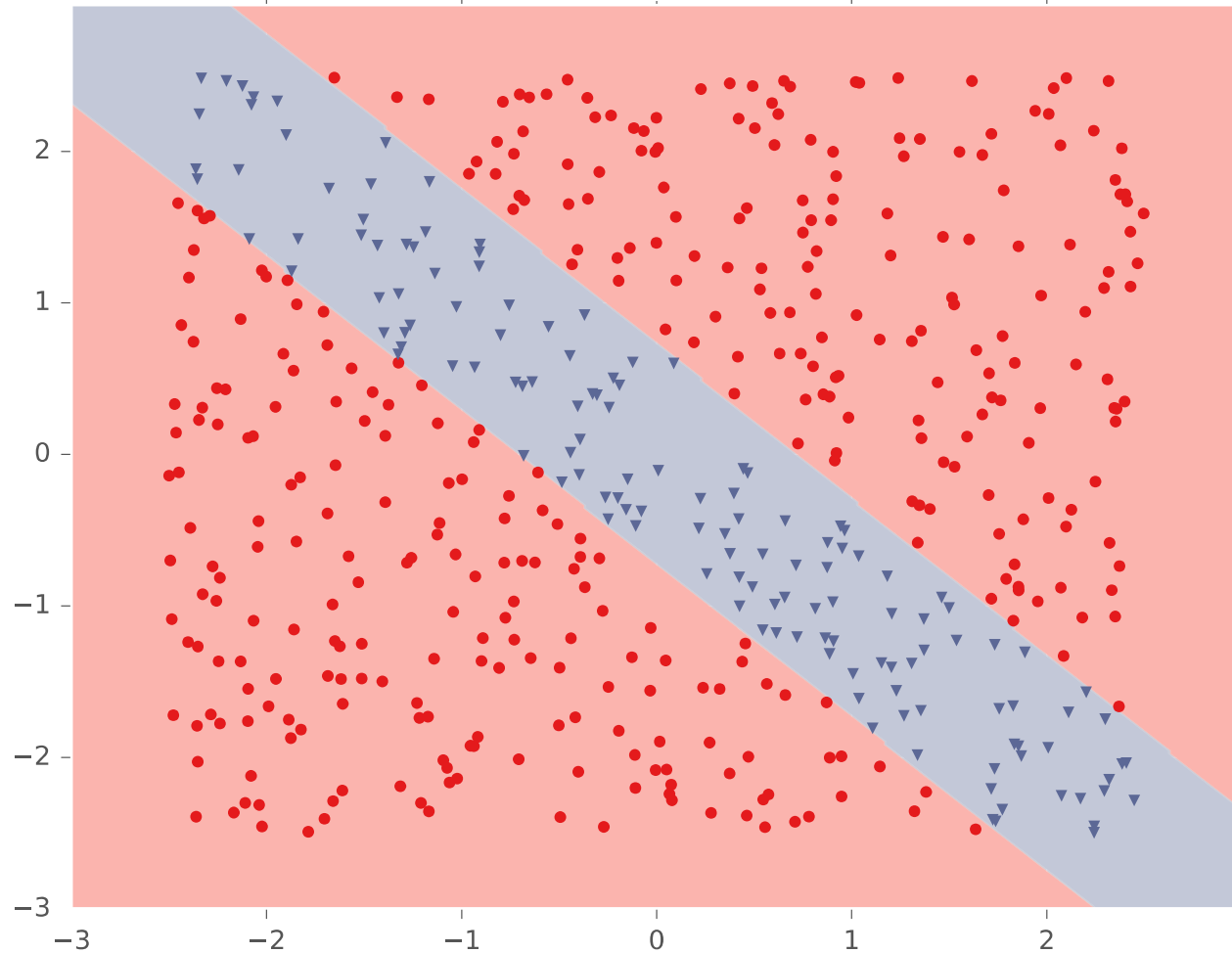


Example #1: Diagonal Band



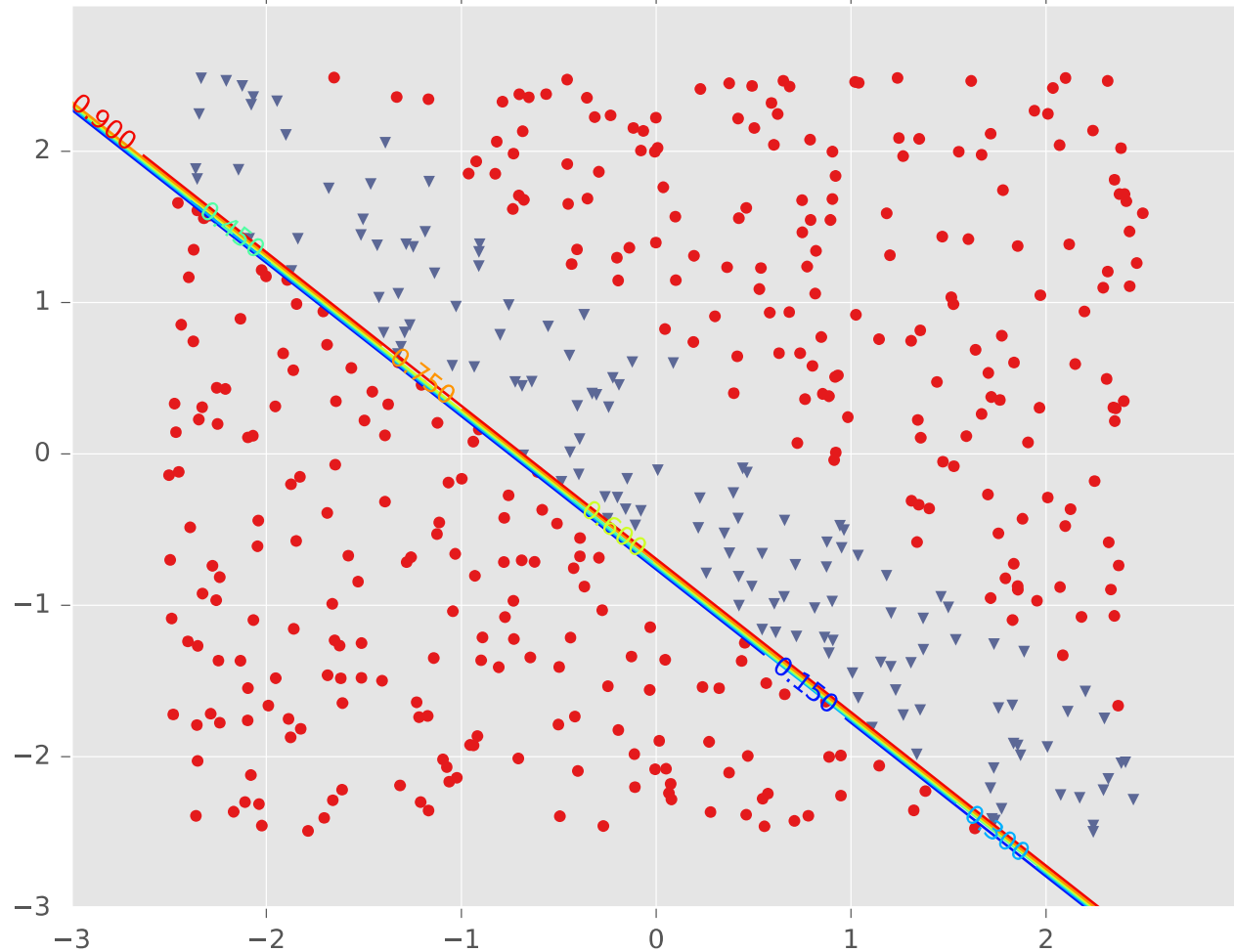
Example #1: Diagonal Band

Tuned Neural Network (hidden=2, activation=logistic)



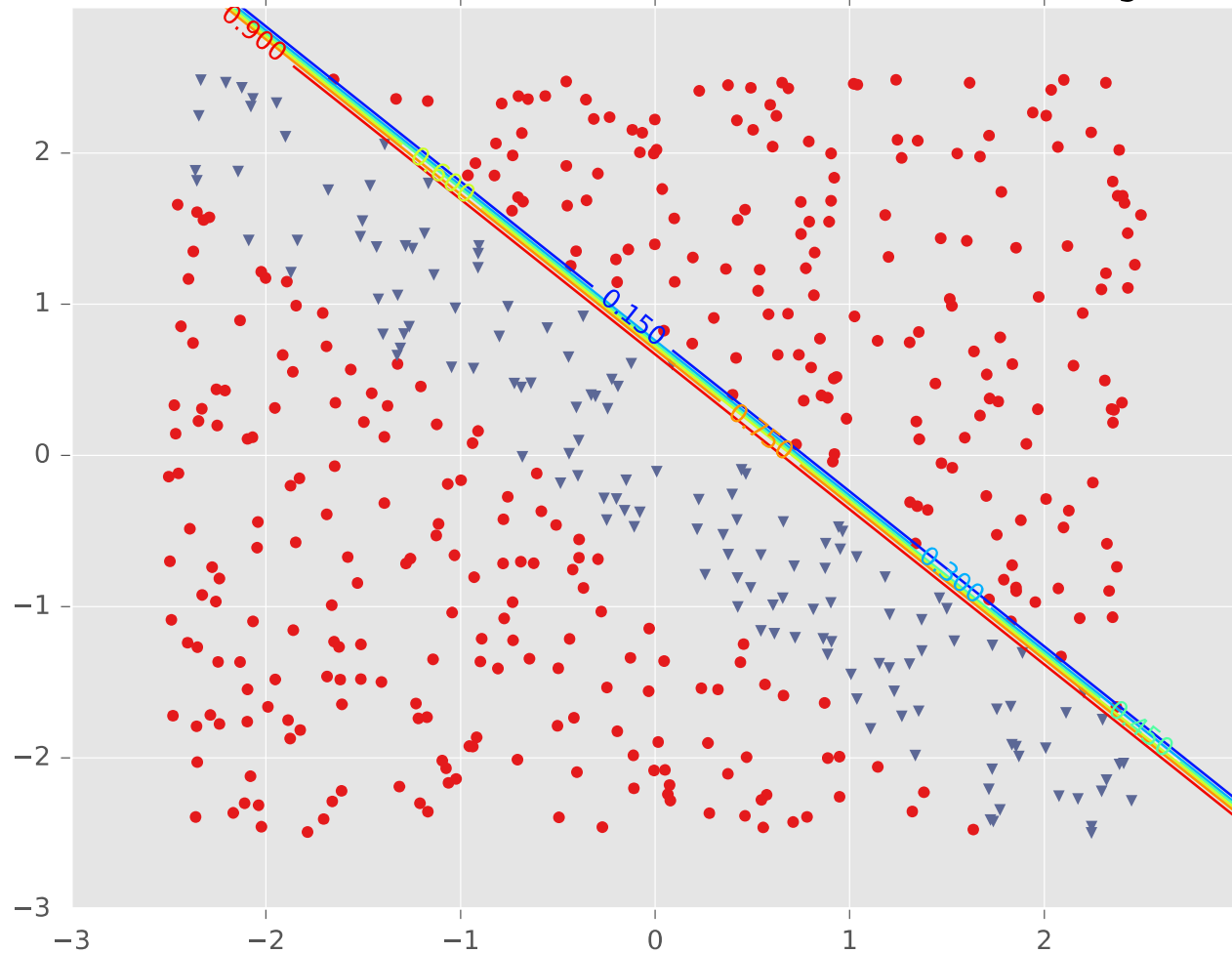
Example #1: Diagonal Band

LR1 for Tuned Neural Network (hidden=2, activation=logistic)



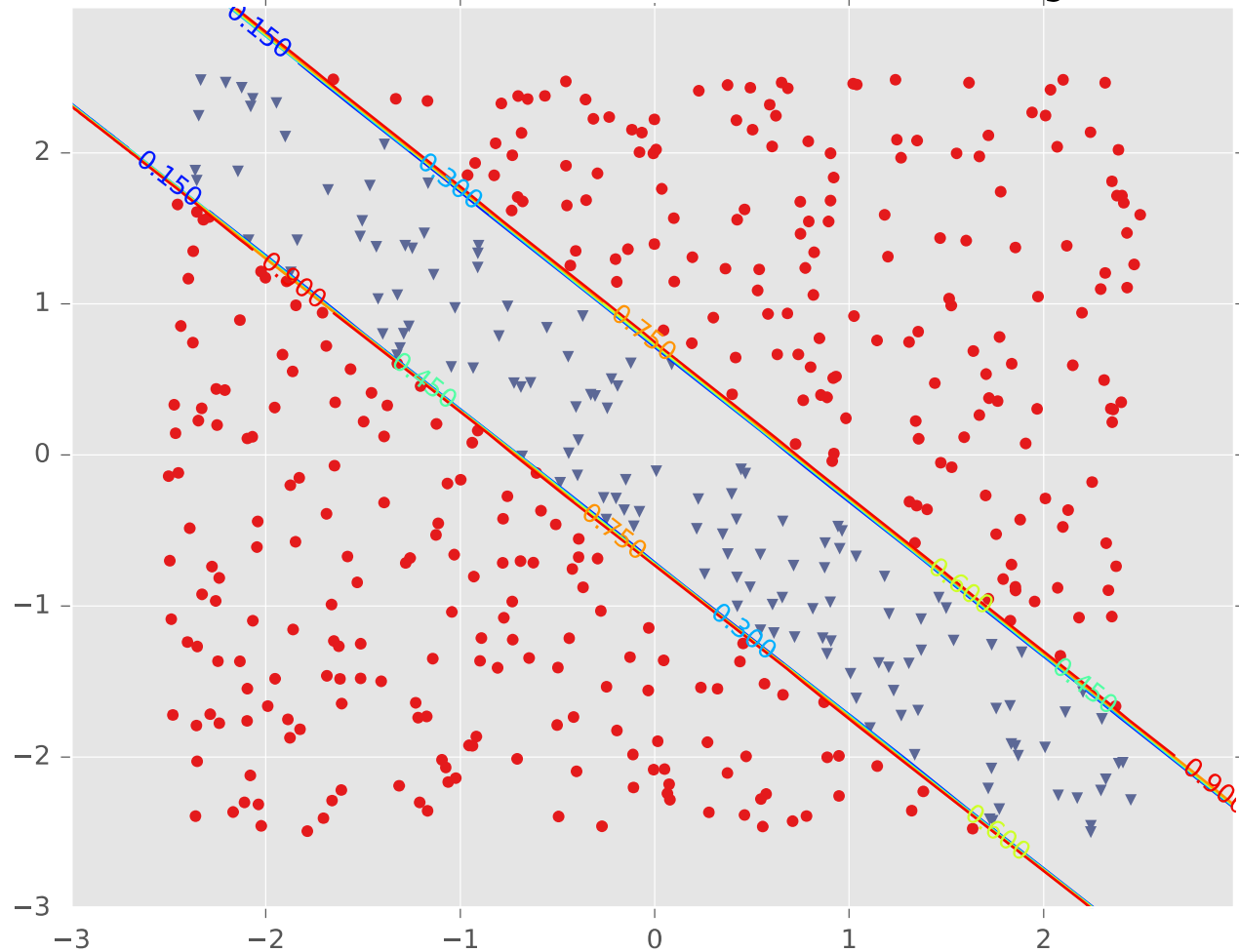
Example #1: Diagonal Band

LR2 for Tuned Neural Network (hidden=2, activation=logistic)

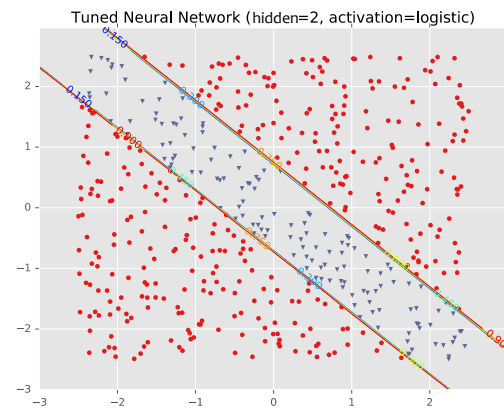
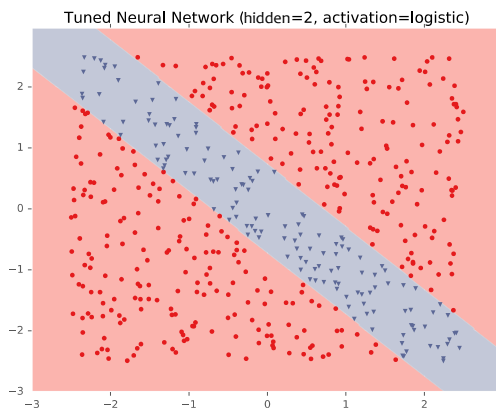
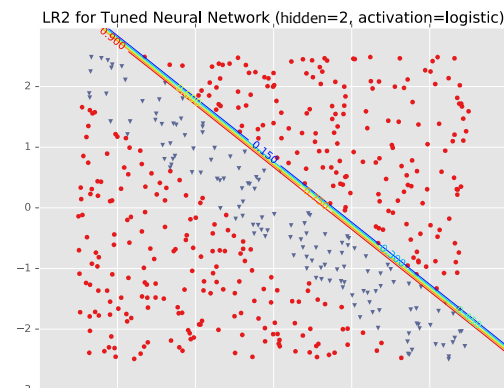
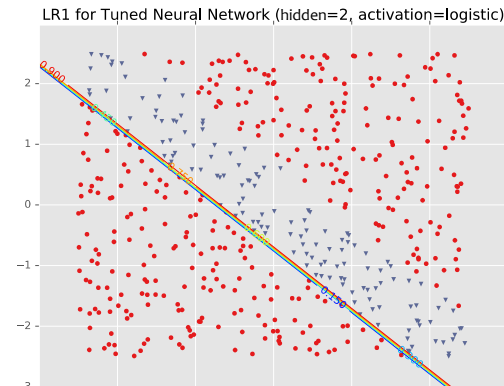


Example #1: Diagonal Band

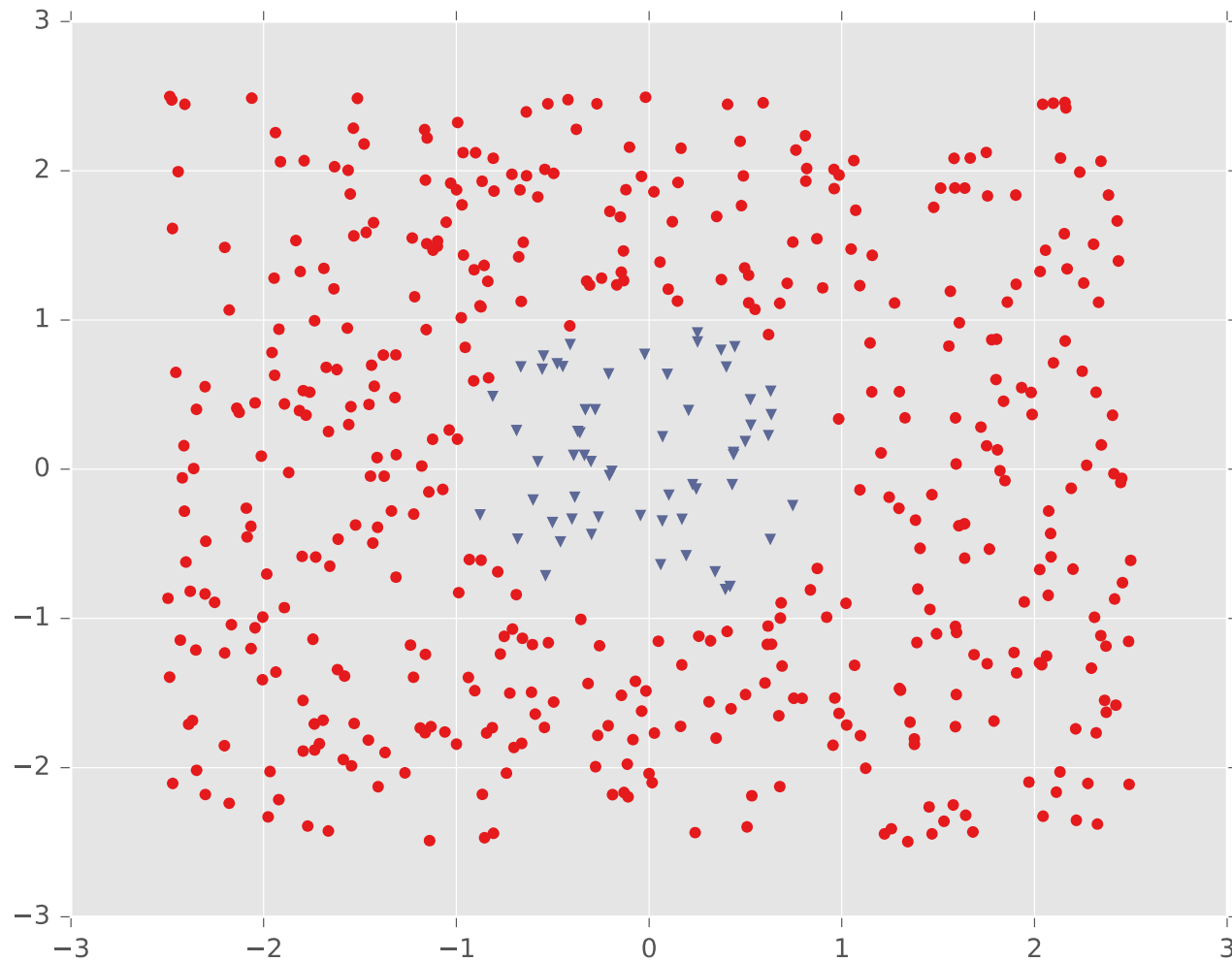
Tuned Neural Network (hidden=2, activation=logistic)



Example #1: Diagonal Band



Example #2: One Pocket

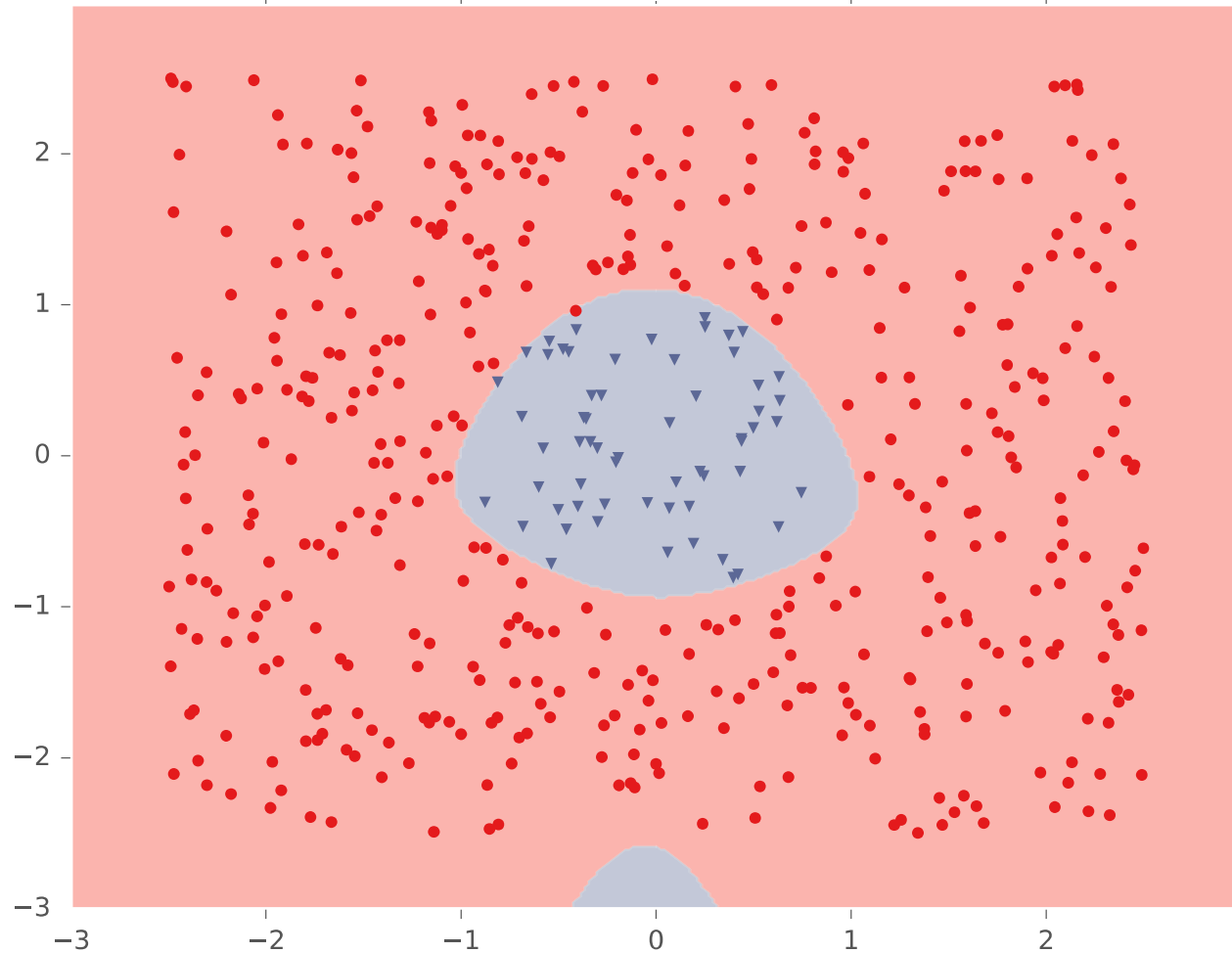


Example #2: One Pocket



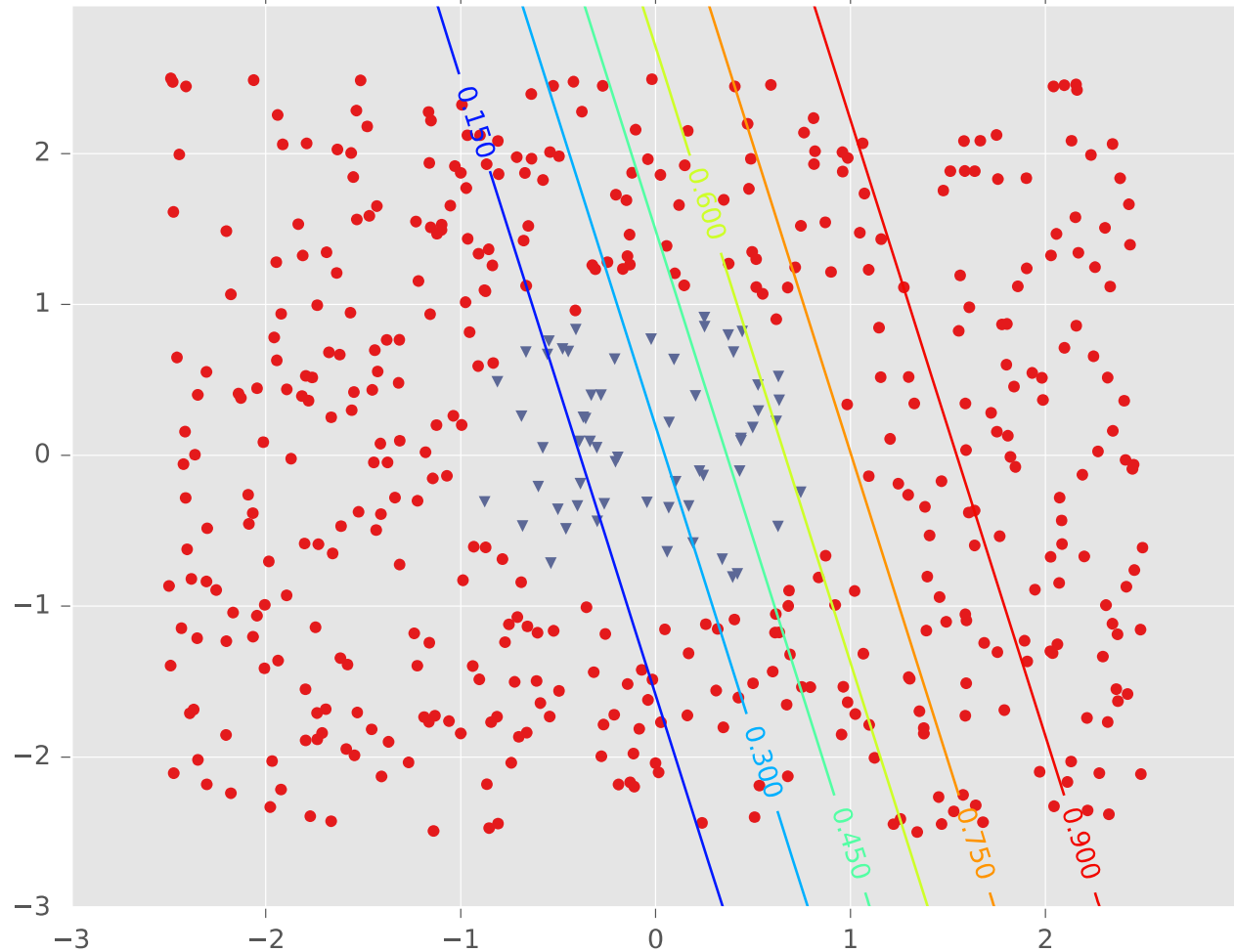
Example #2: One Pocket

Tuned Neural Network (hidden=3, activation=logistic)



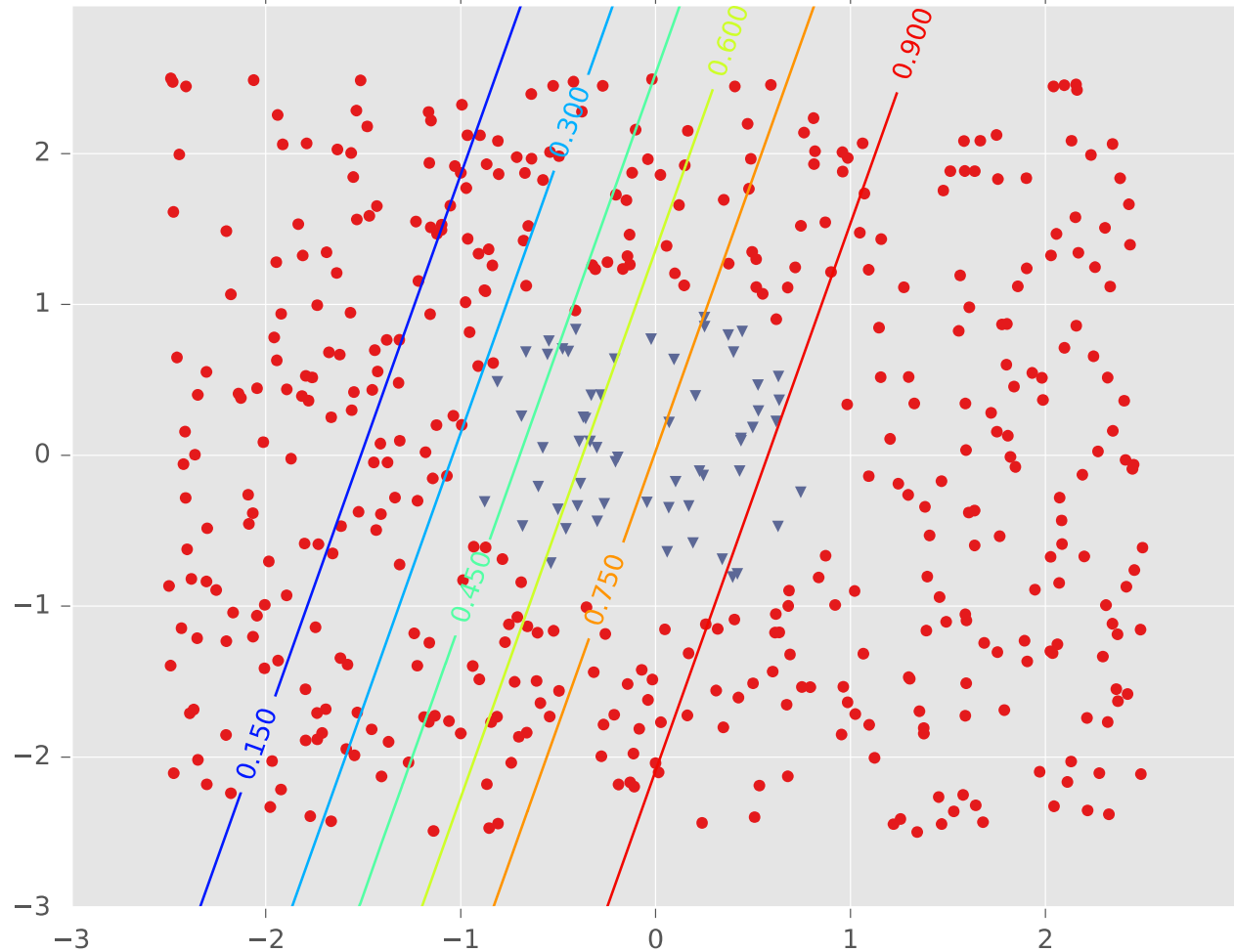
Example #2: One Pocket

LR1 for Tuned Neural Network (hidden=3, activation=logistic)



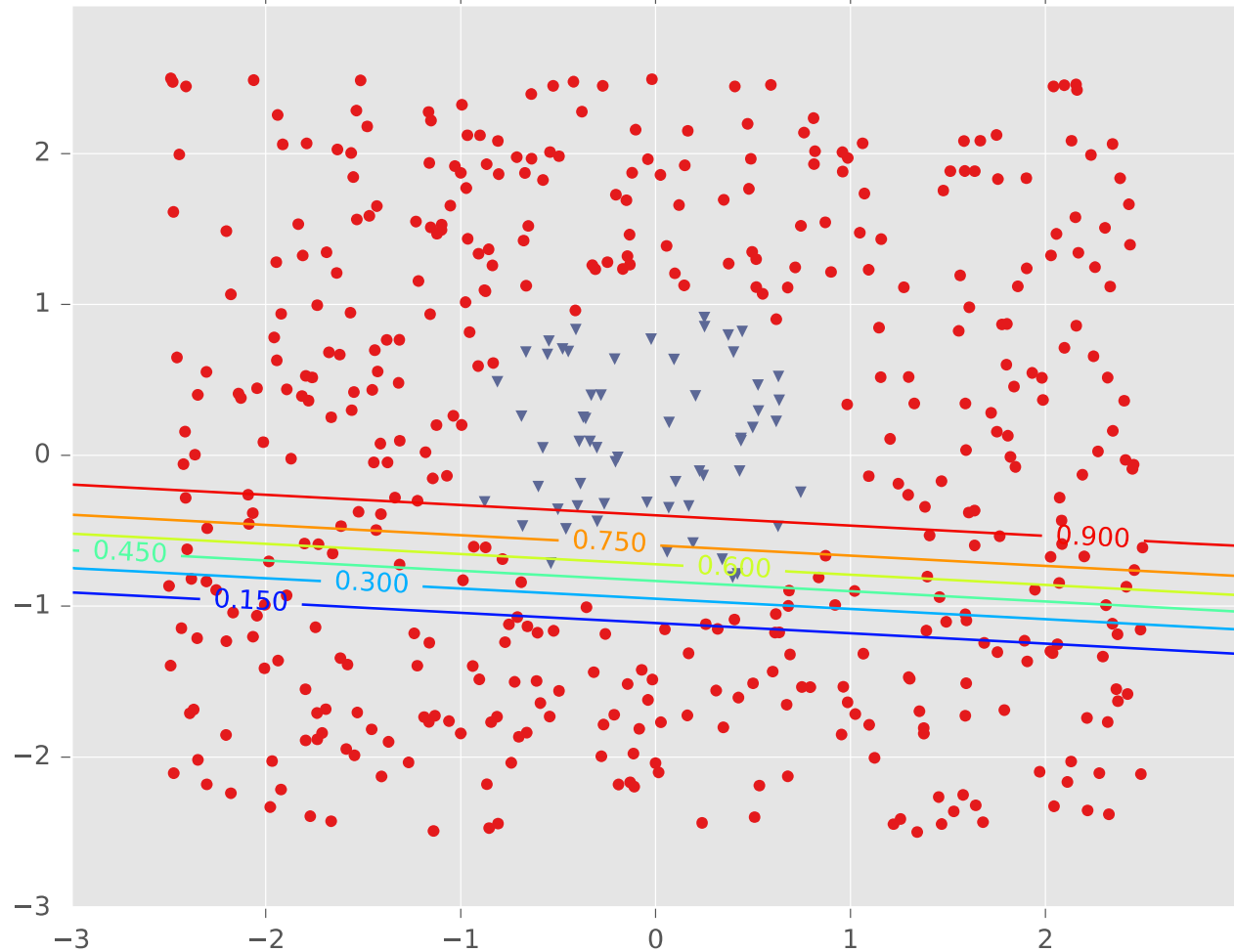
Example #2: One Pocket

LR2 for Tuned Neural Network (hidden=3, activation=logistic)



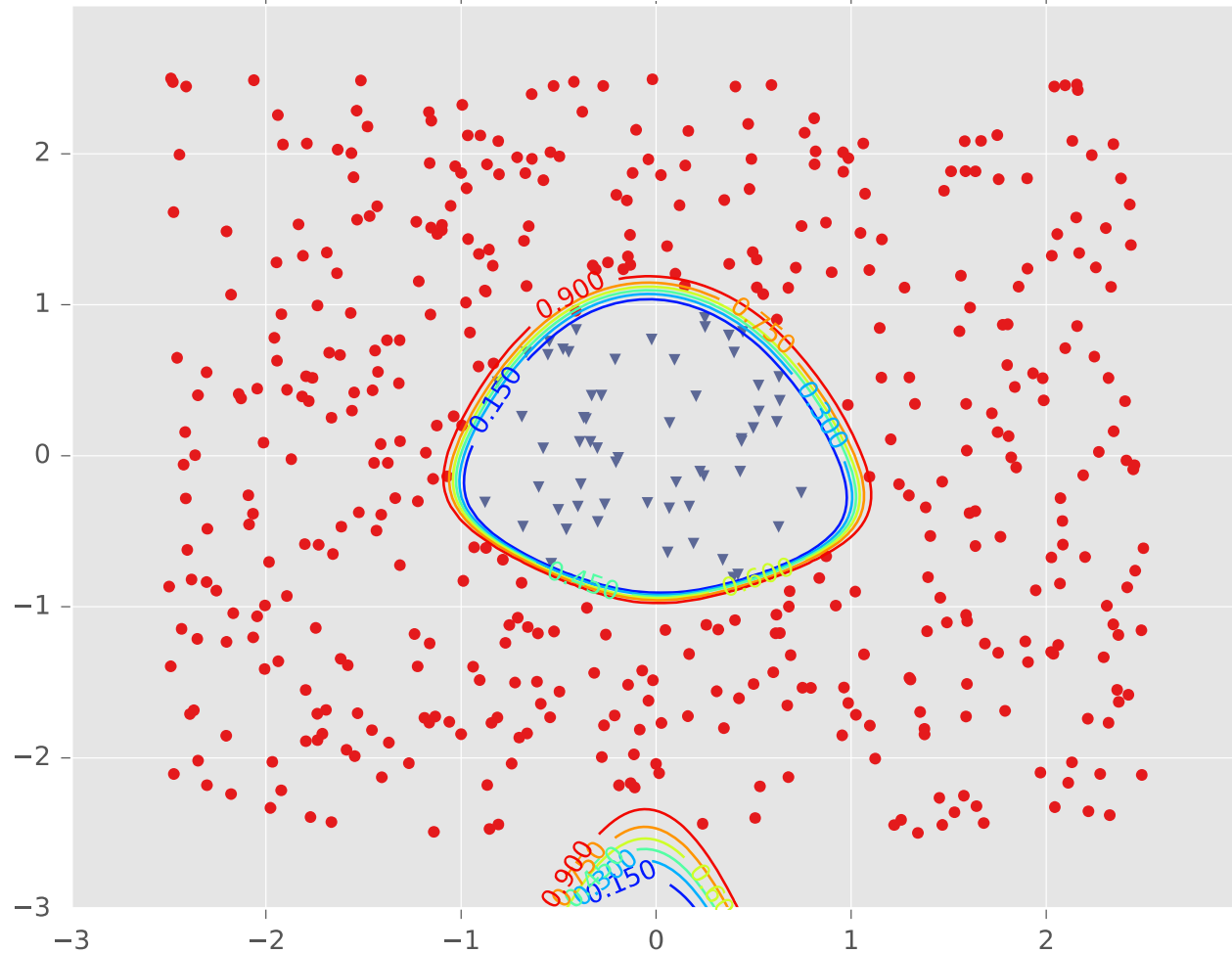
Example #2: One Pocket

LR3 for Tuned Neural Network (hidden=3, activation=logistic)



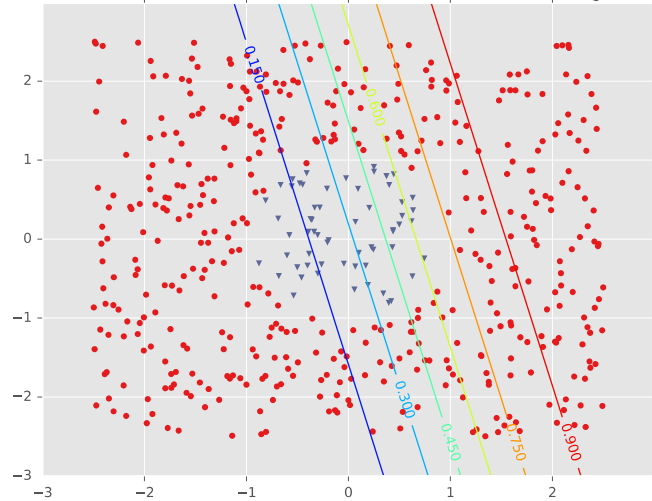
Example #2: One Pocket

Tuned Neural Network (hidden=3, activation=logistic)

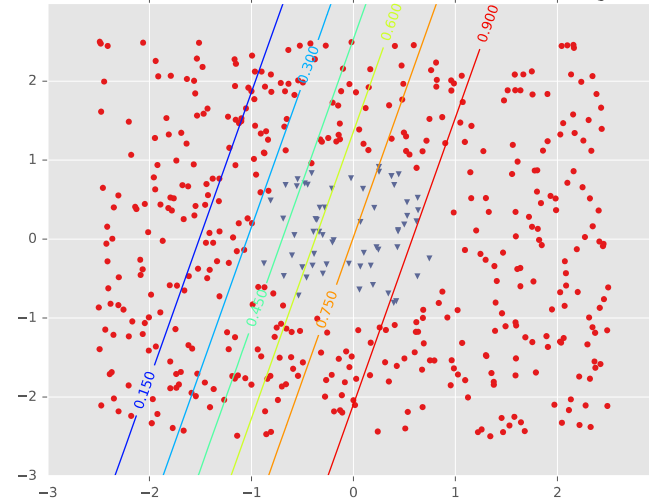


Example #2: One Pocket

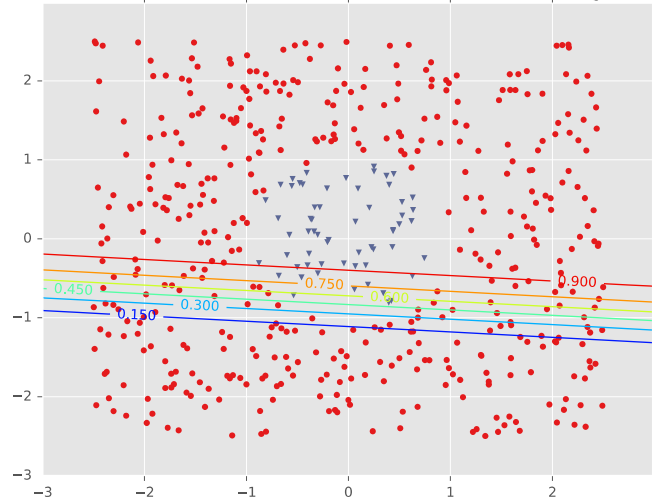
LR1 for Tuned Neural Network (hidden=3, activation=logistic)



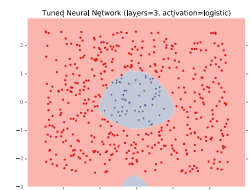
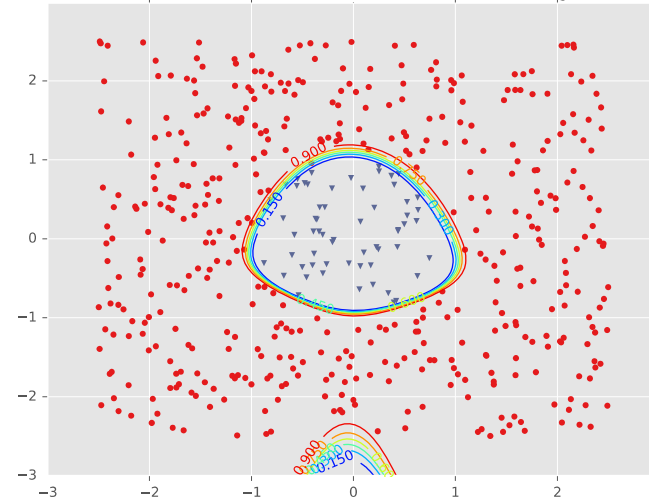
LR2 for Tuned Neural Network (hidden=3, activation=logistic)



LR3 for Tuned Neural Network (hidden=3, activation=logistic)



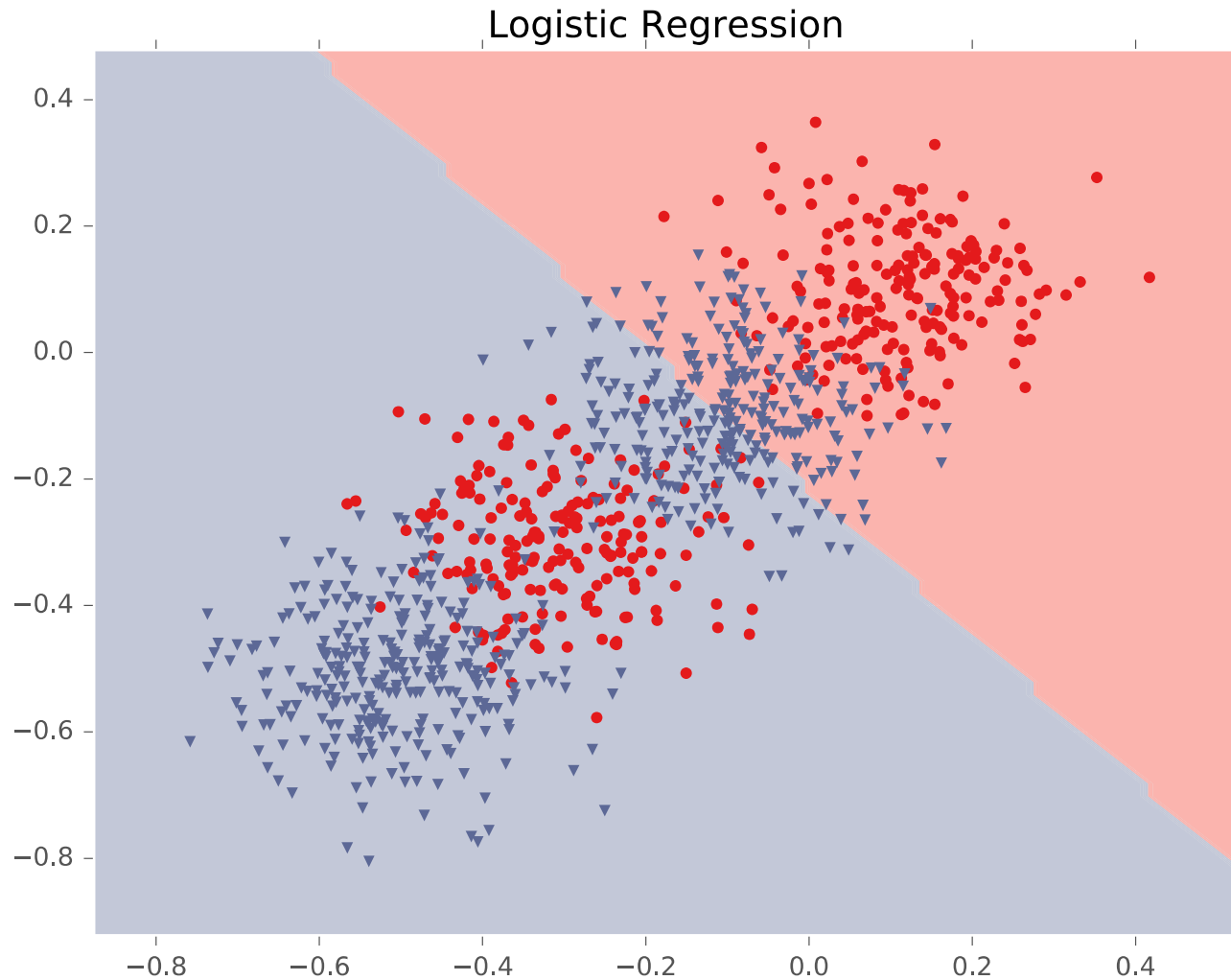
Tuned Neural Network (hidden=3, activation=logistic)



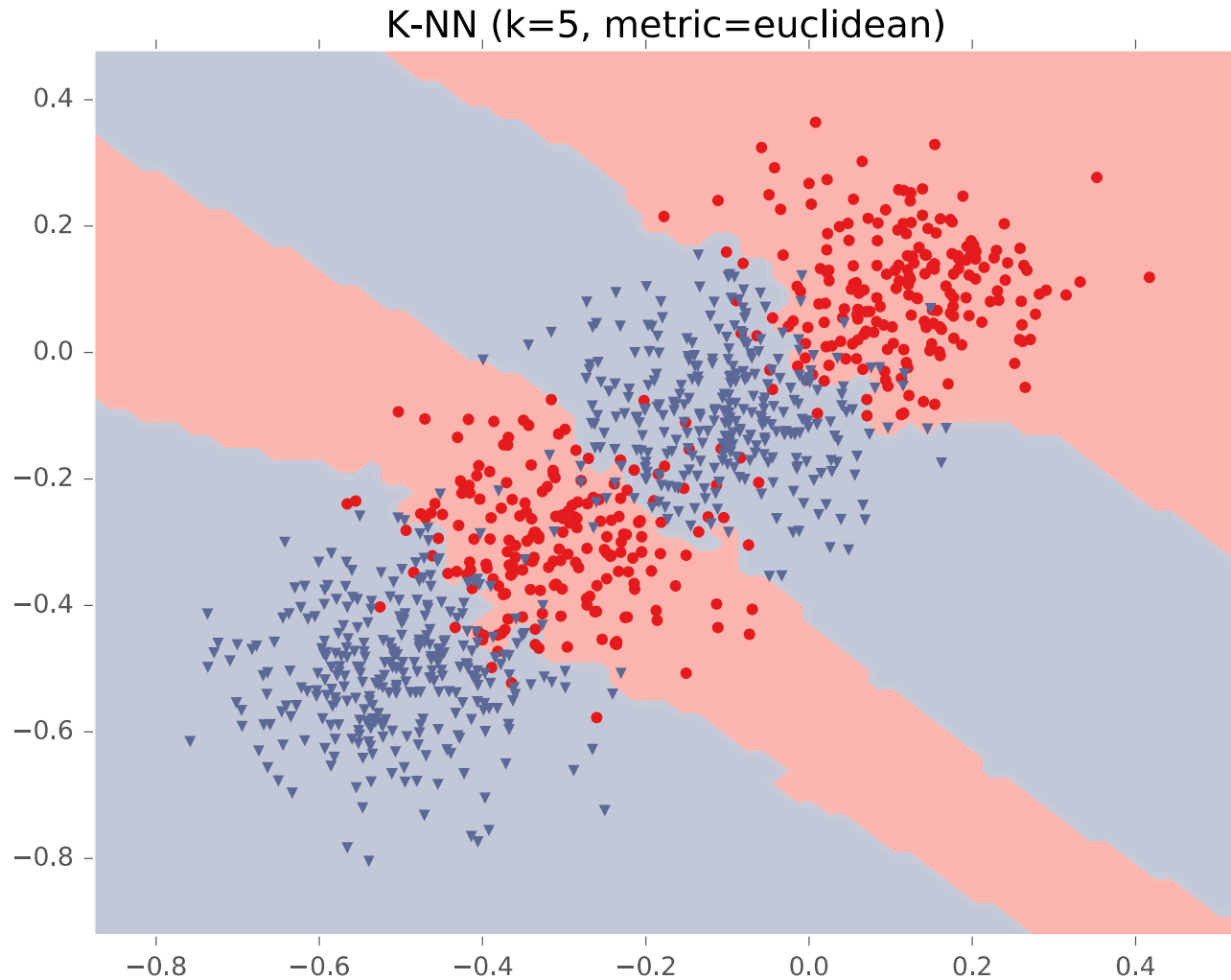
Example #3: Four Gaussians



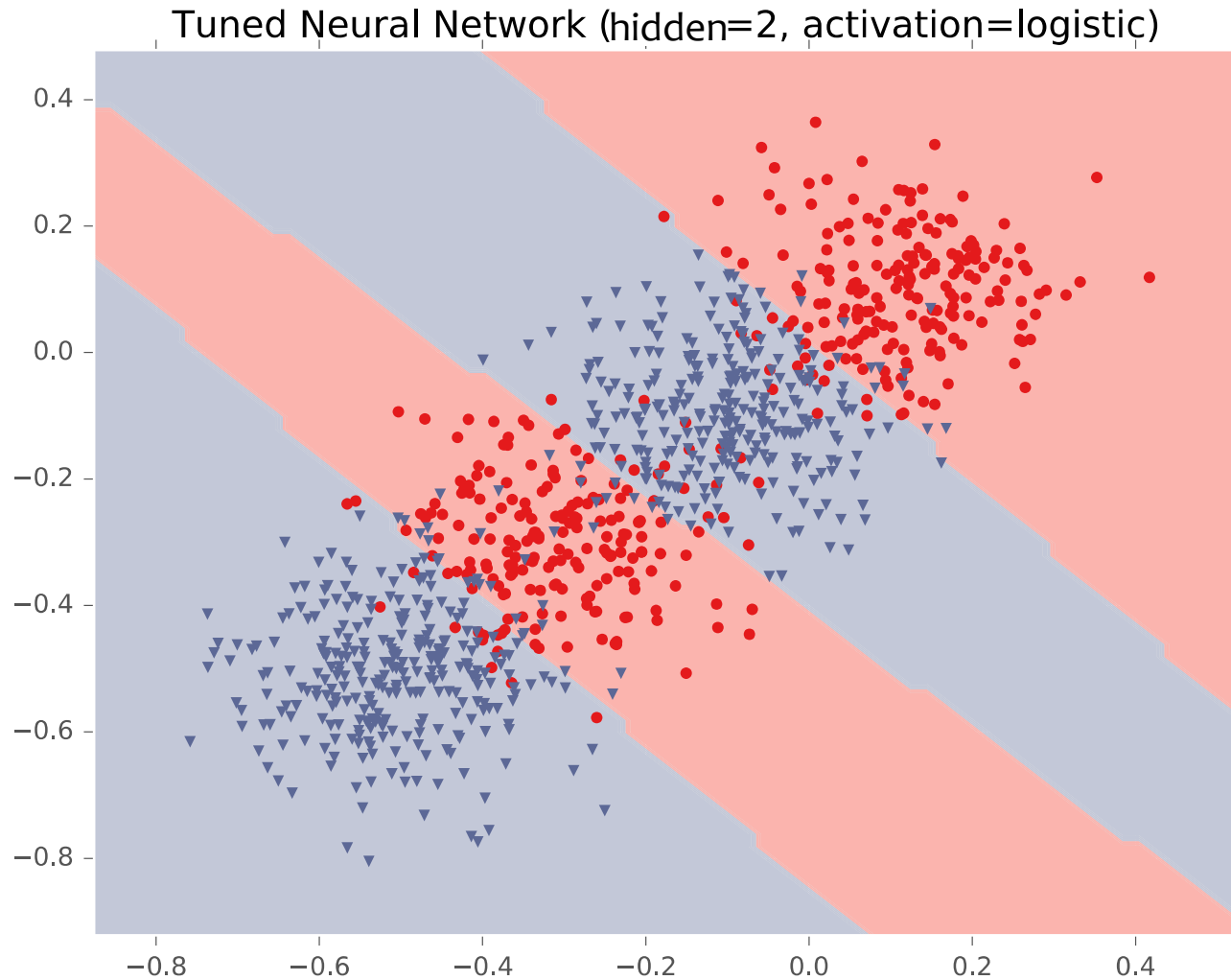
Example #3: Four Gaussians



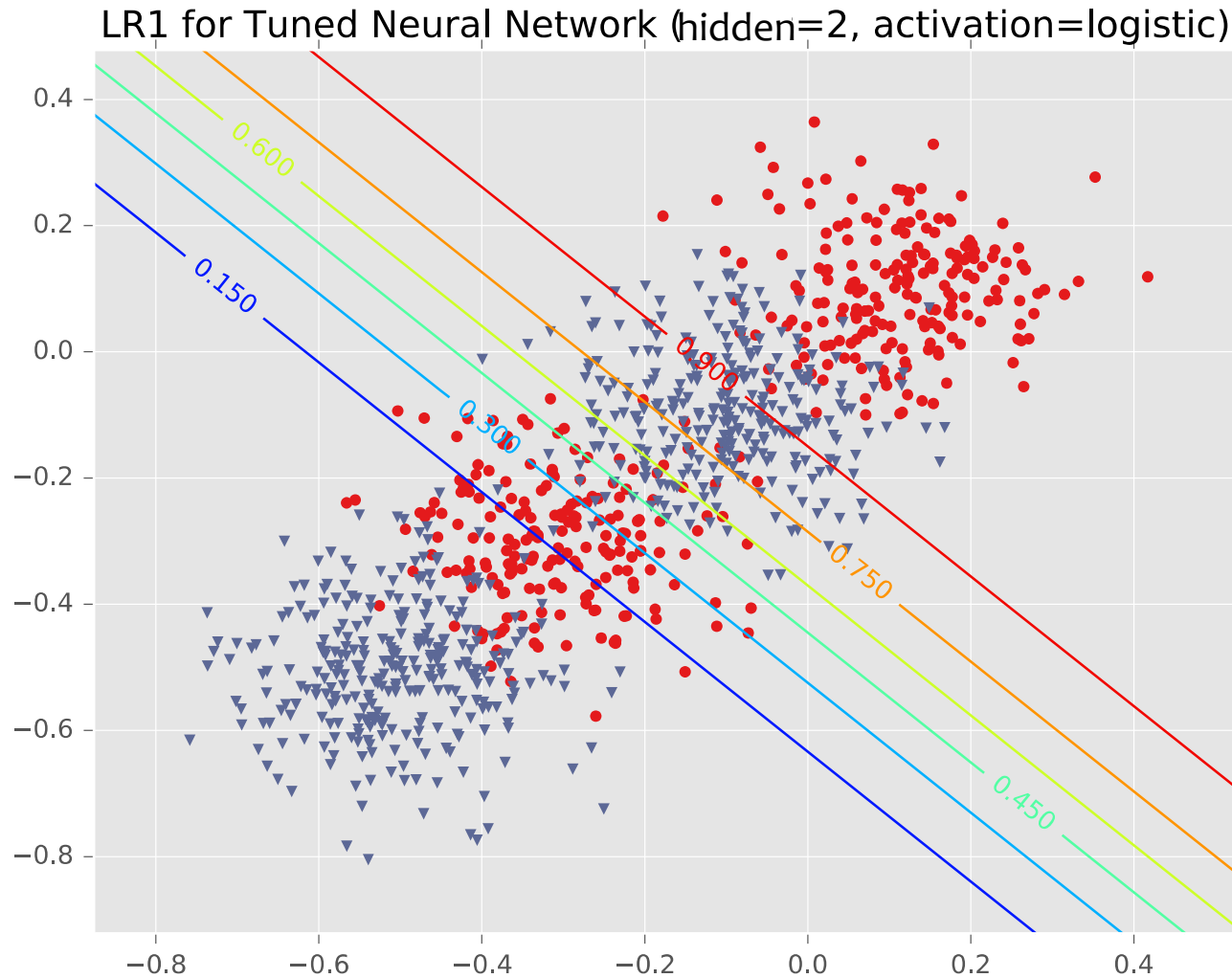
Example #3: Four Gaussians



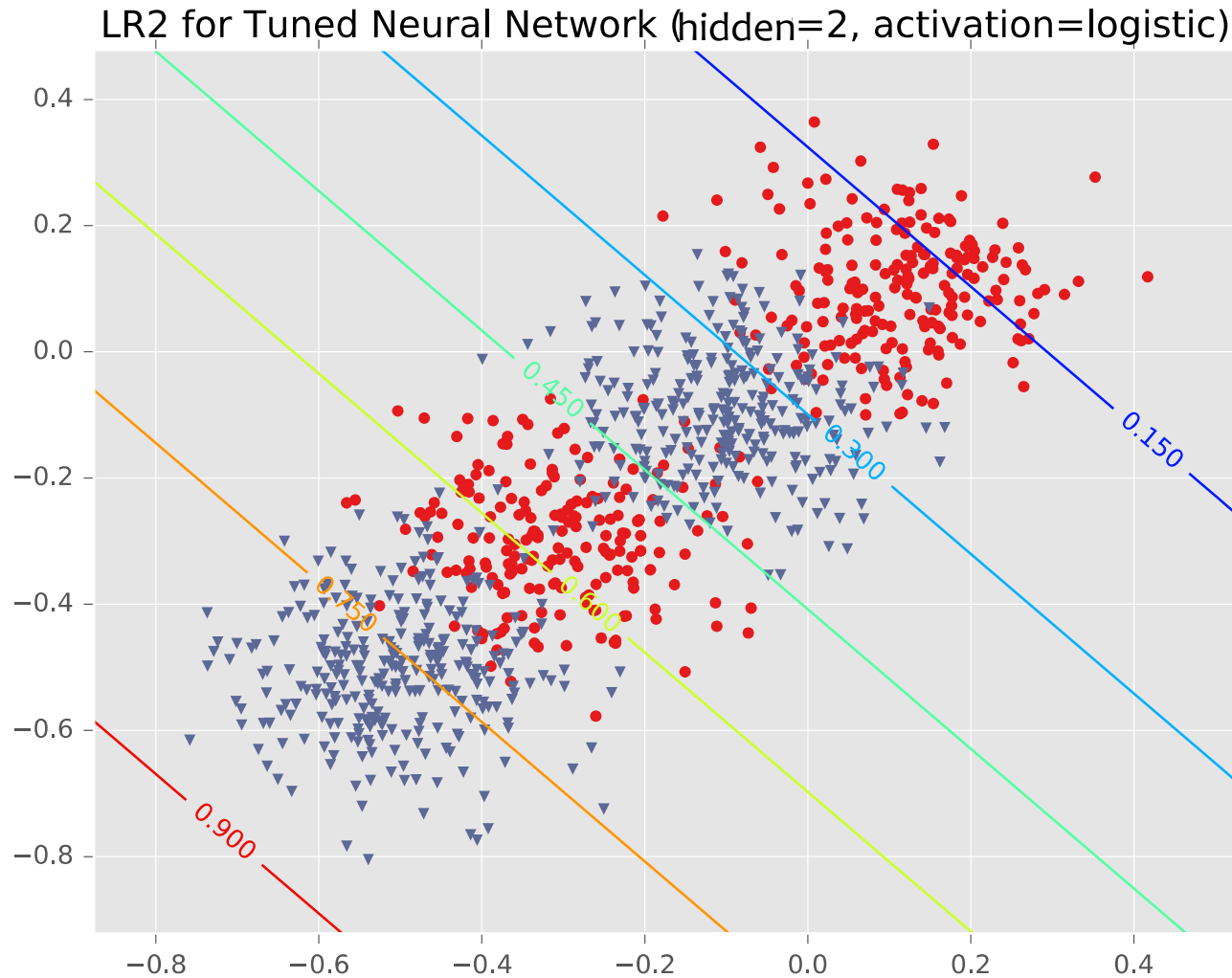
Example #3: Four Gaussians



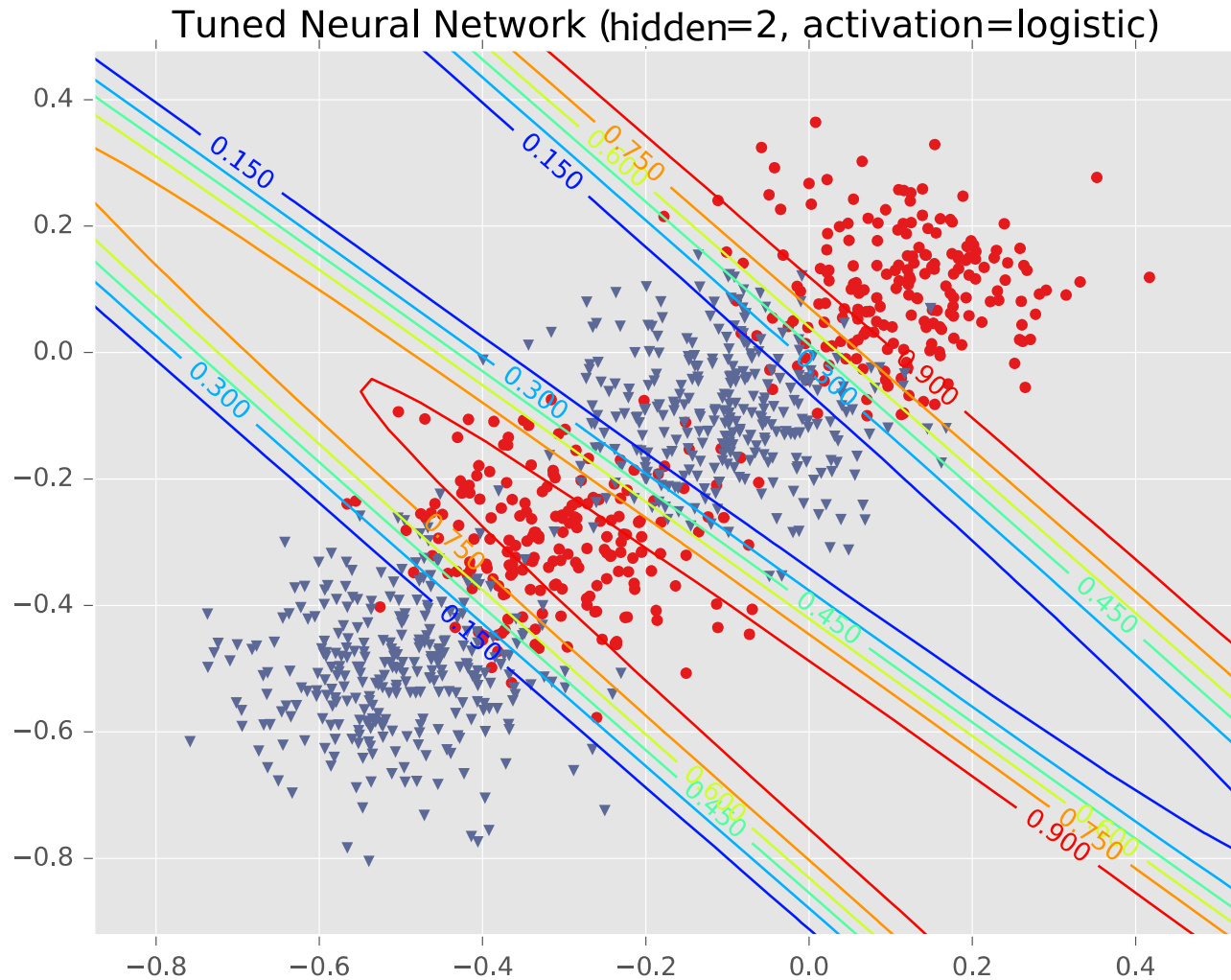
Example #3: Four Gaussians



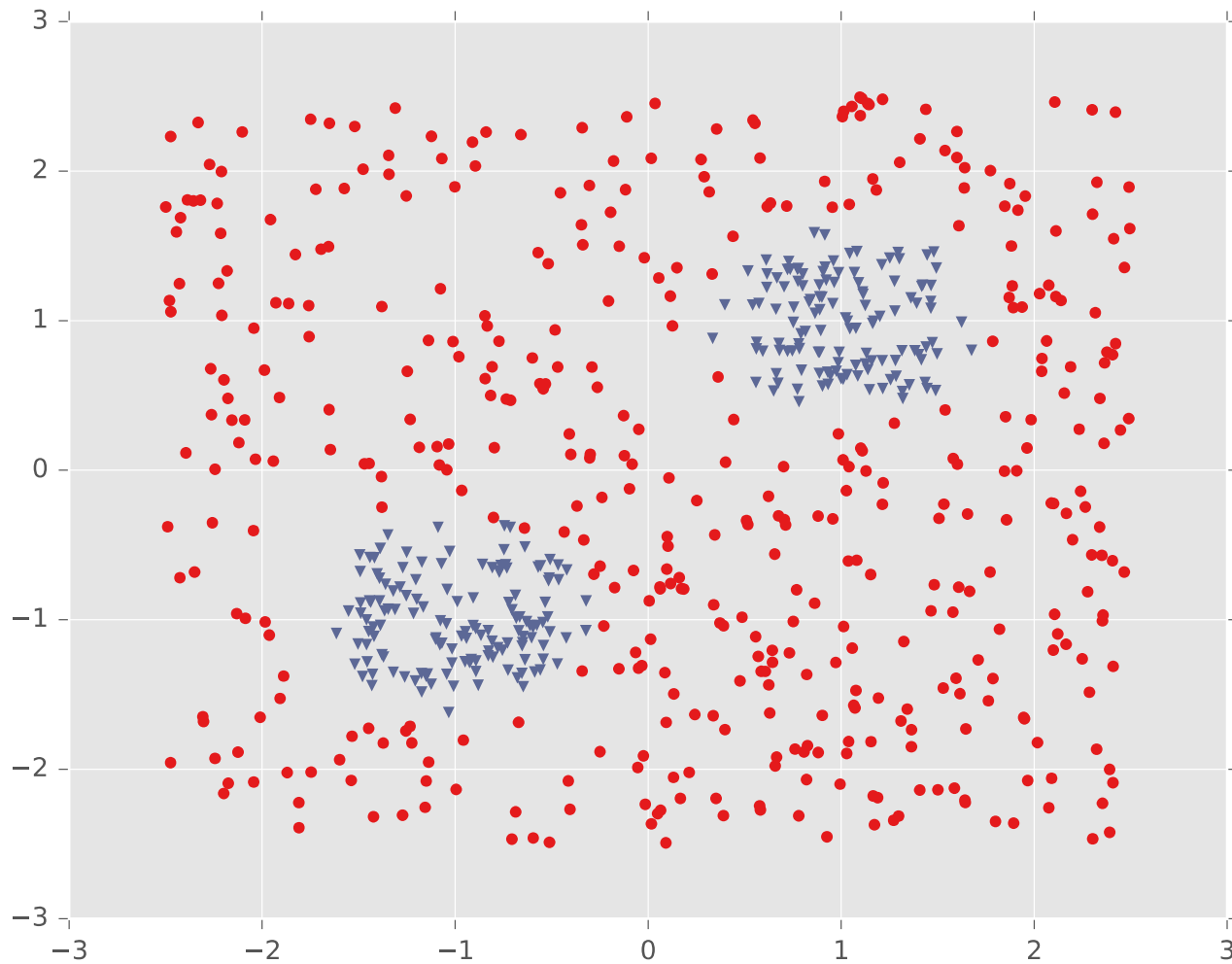
Example #3: Four Gaussians



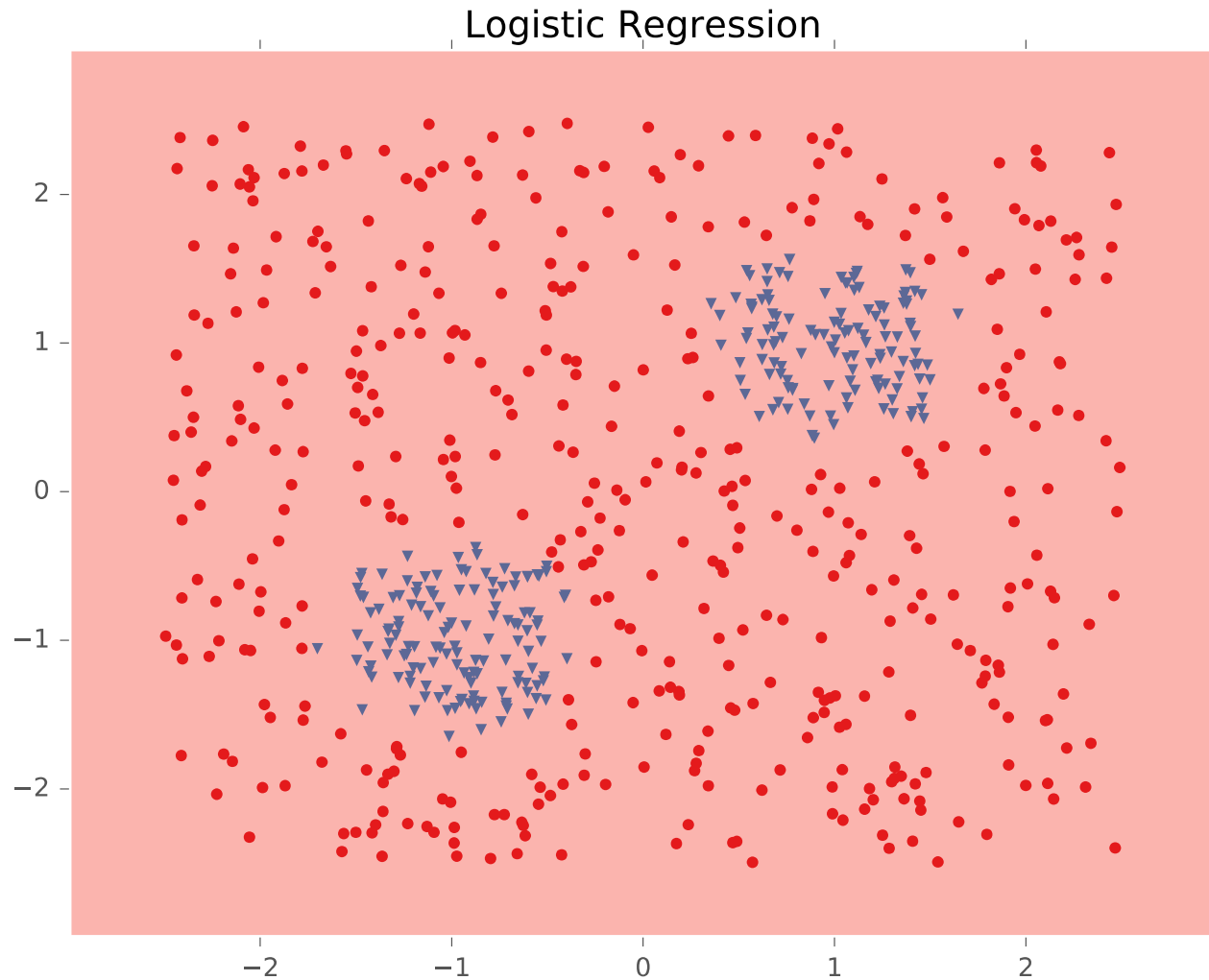
Example #3: Four Gaussians



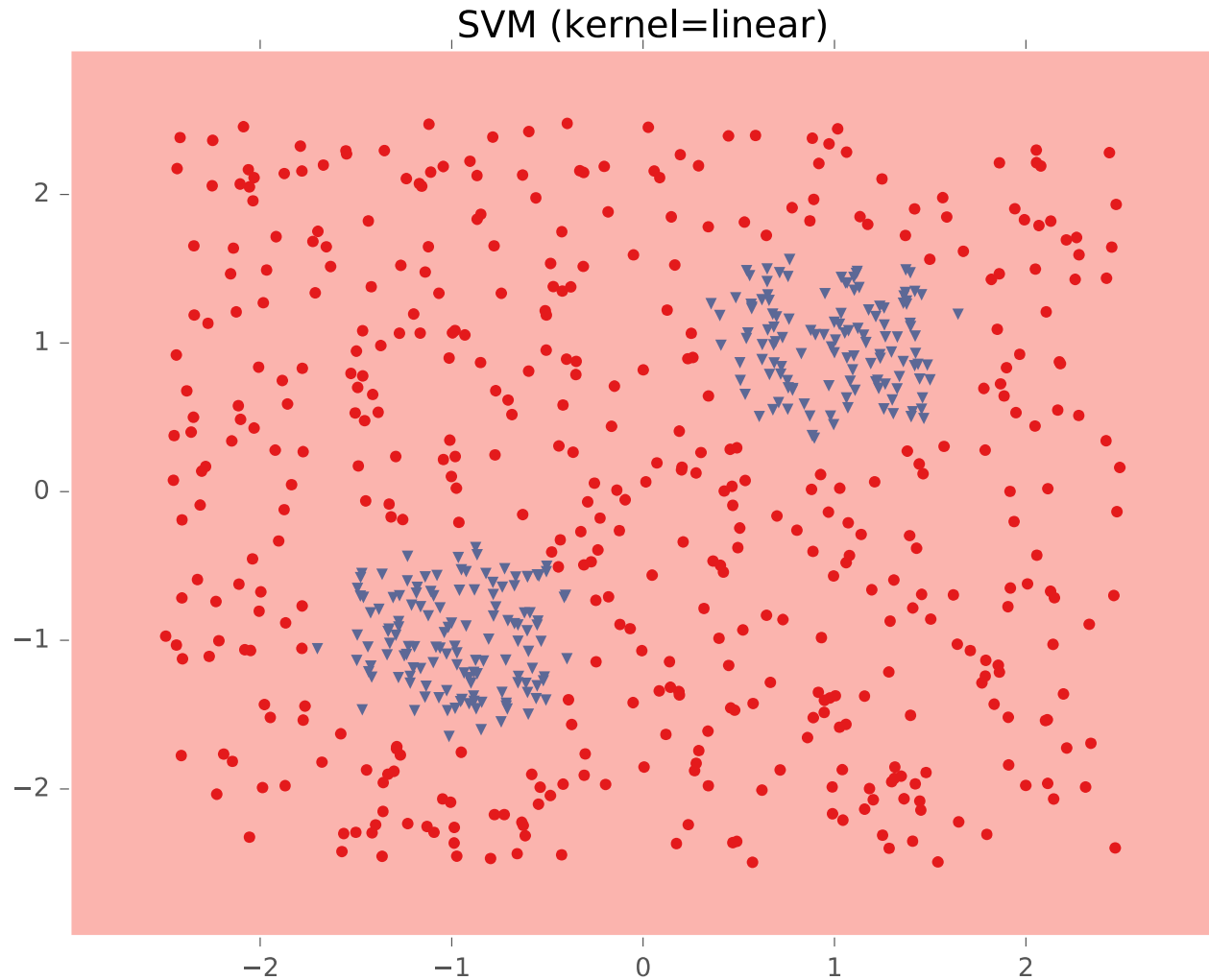
Example #4: Two Pockets



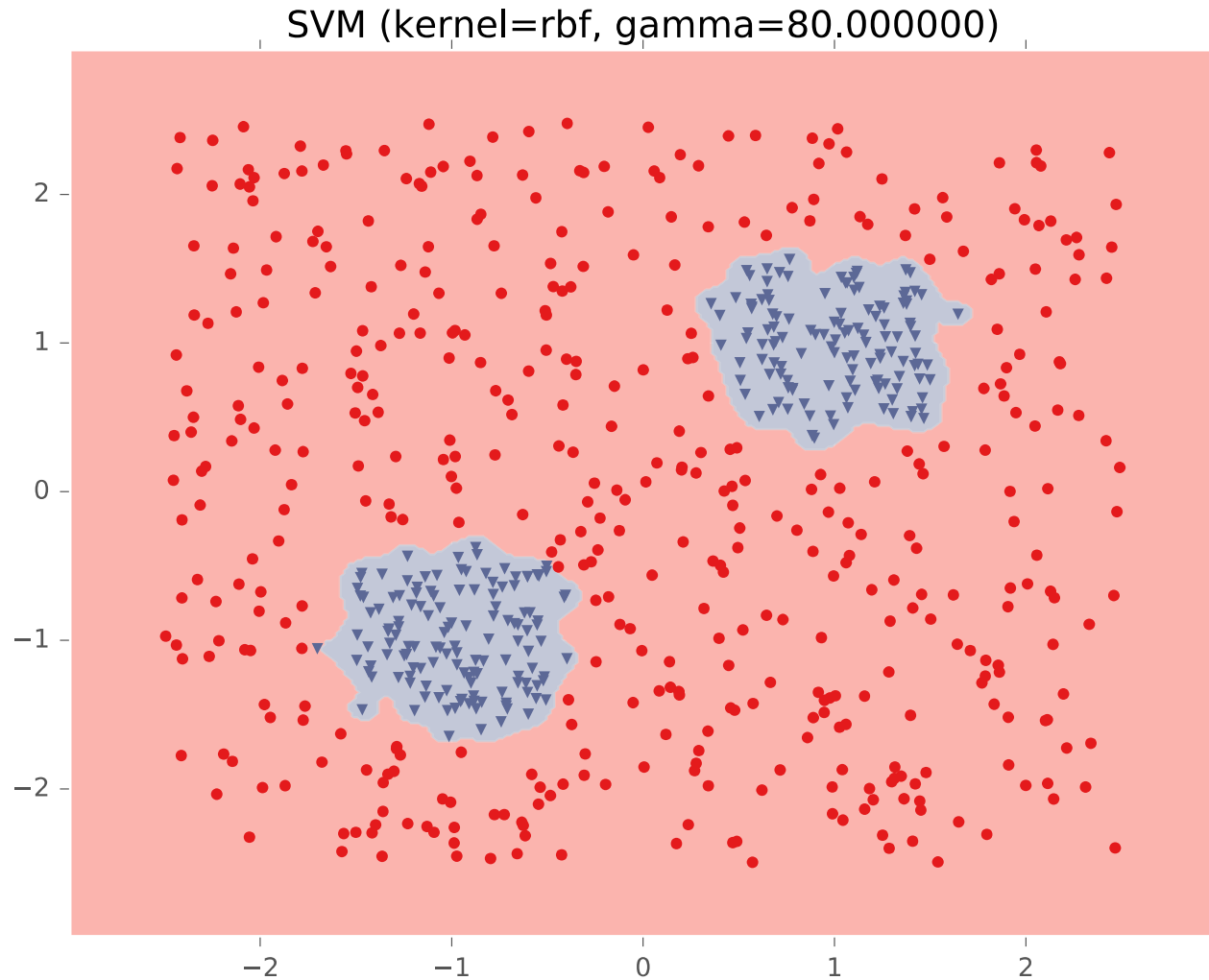
Example #4: Two Pockets



Example #4: Two Pockets

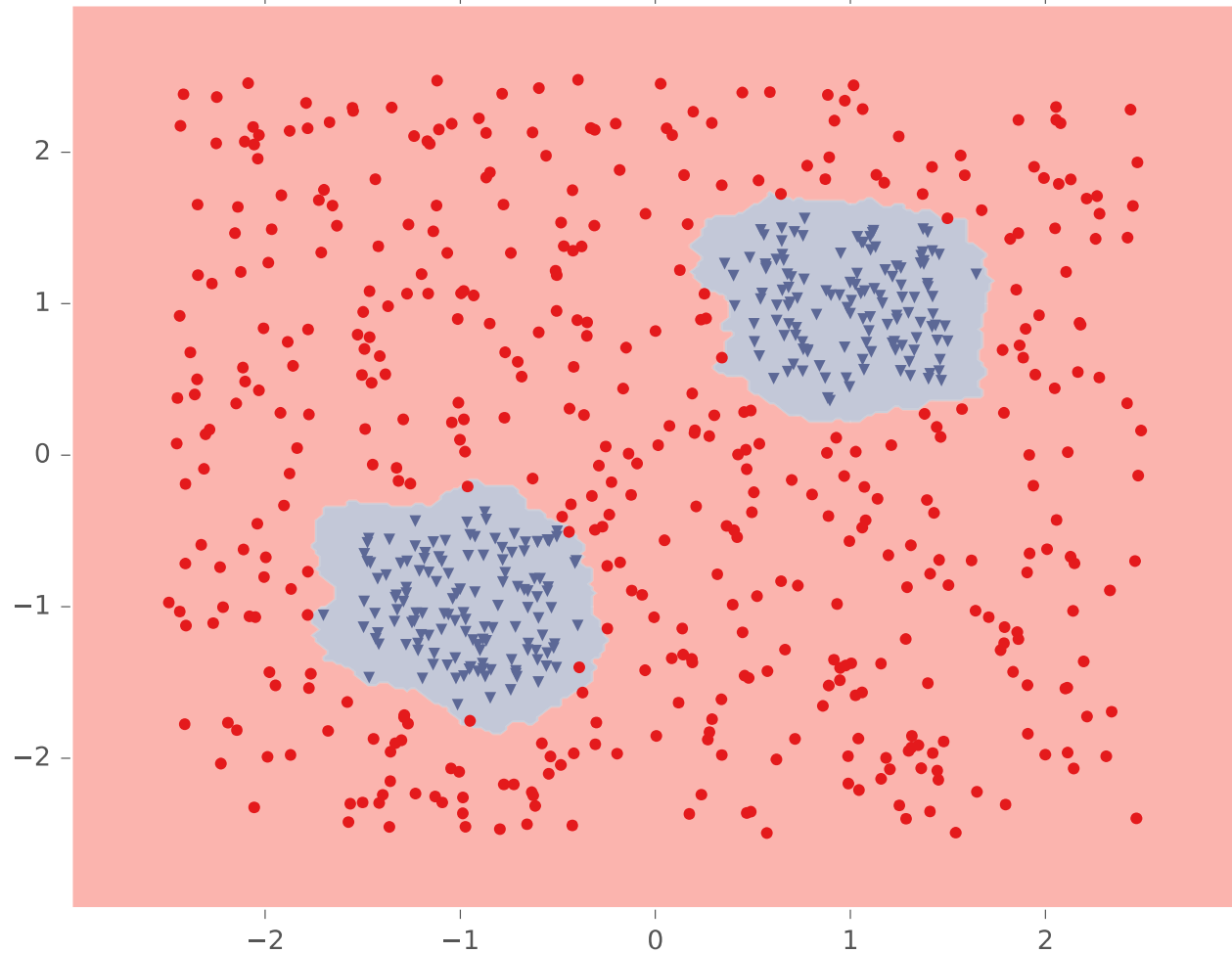


Example #4: Two Pockets



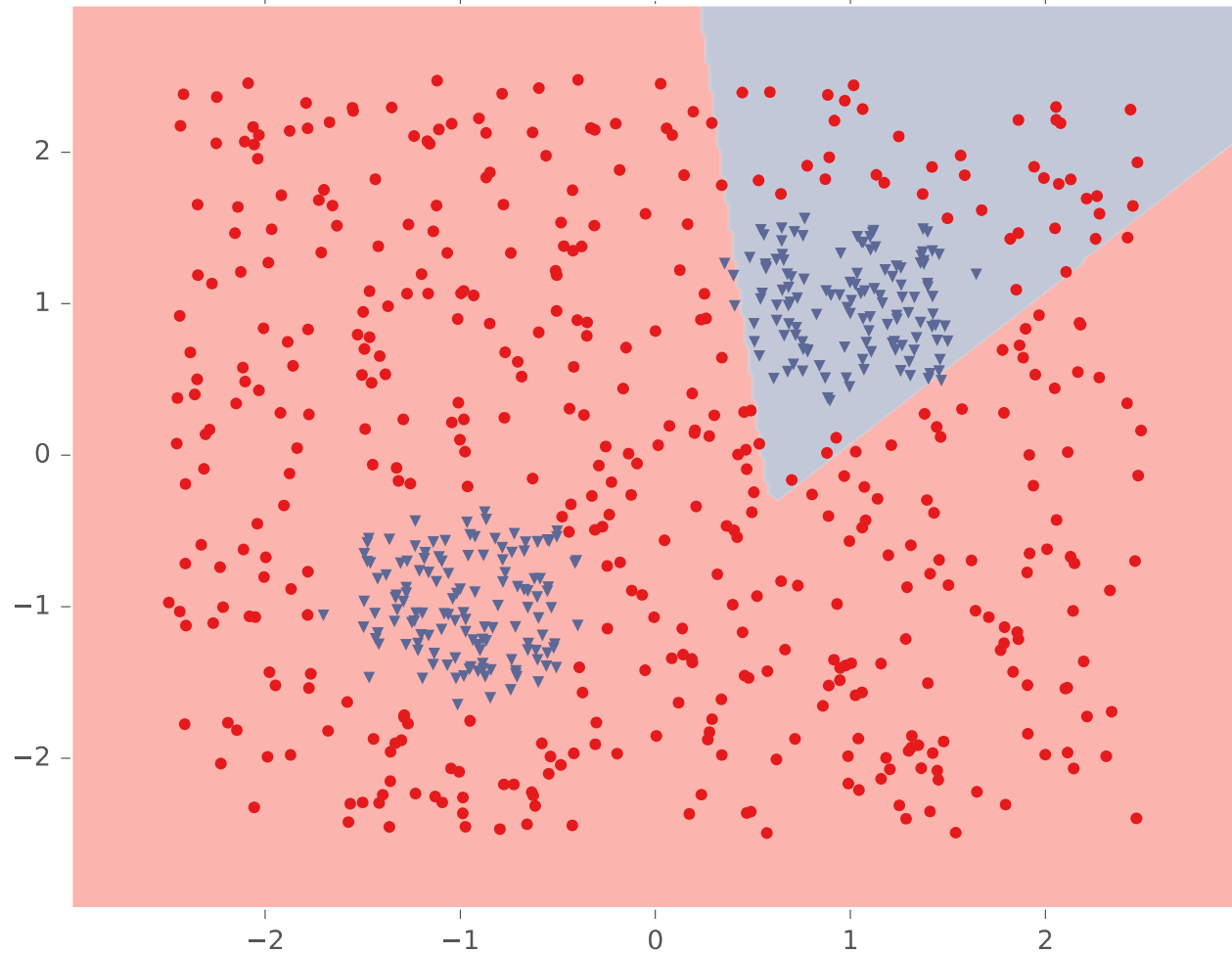
Example #4: Two Pockets

K-NN (k=5, metric=euclidean)



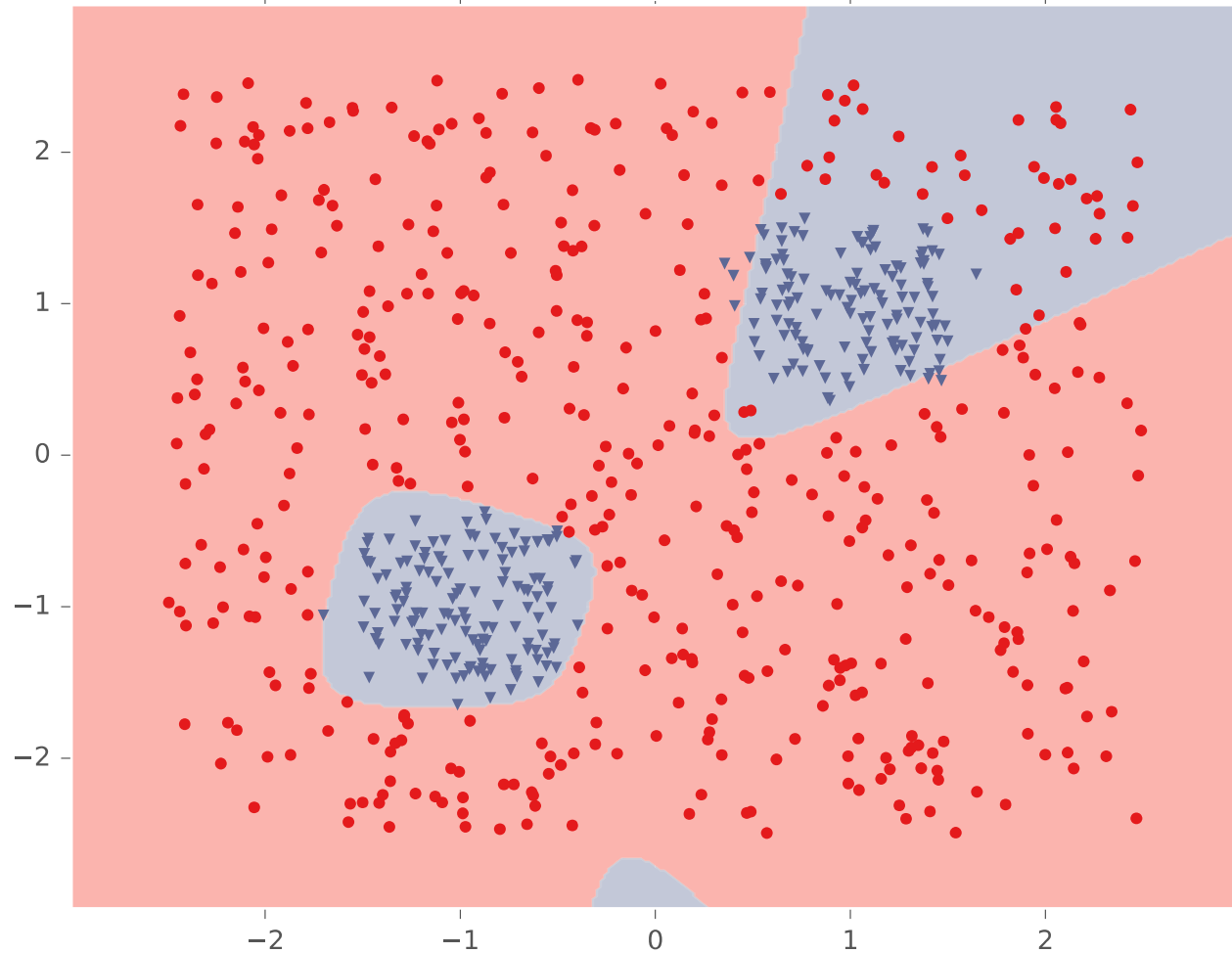
Example #4: Two Pockets

Tuned Neural Network (hidden=2, activation=logistic)



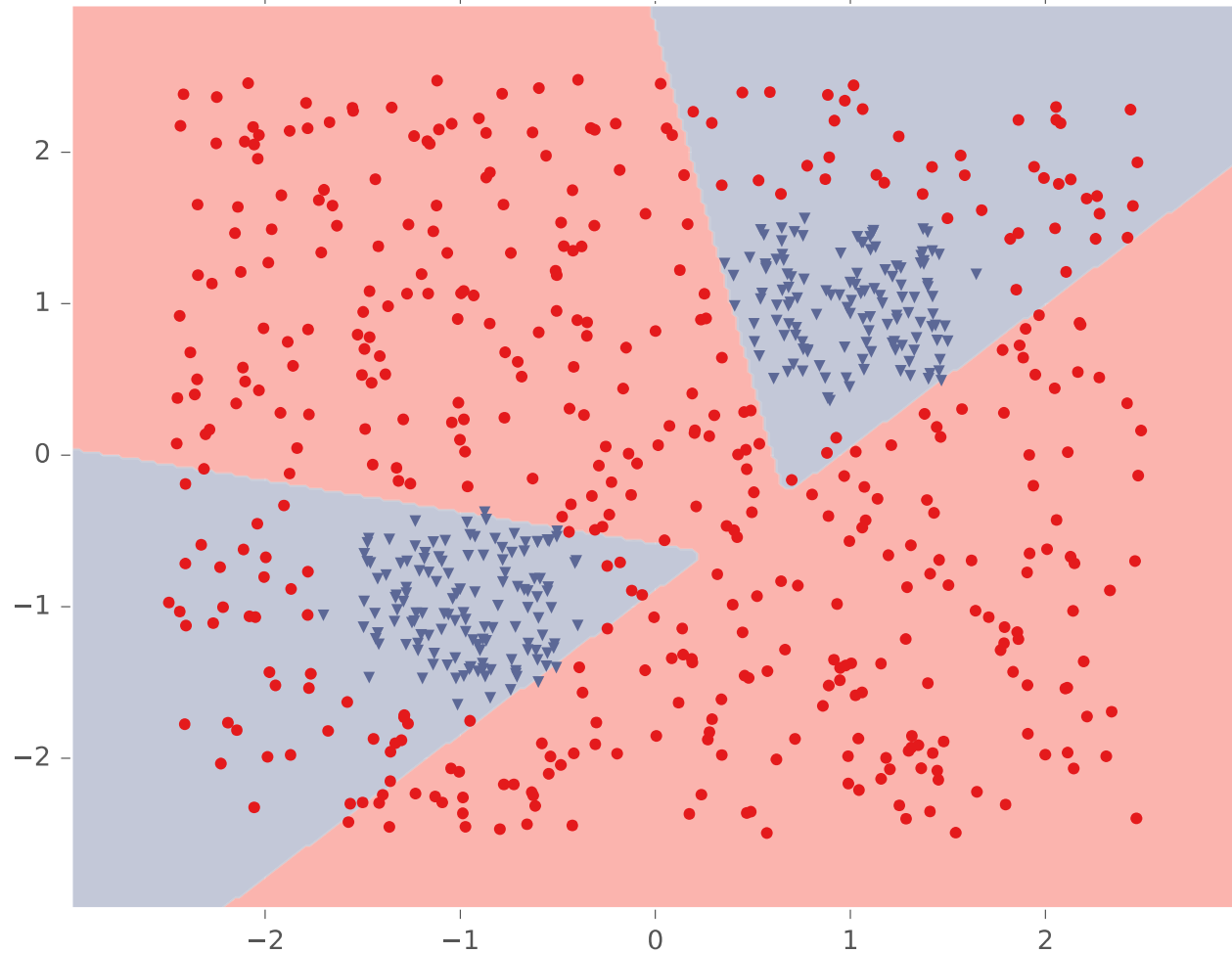
Example #4: Two Pockets

Tuned Neural Network (hidden=3, activation=logistic)



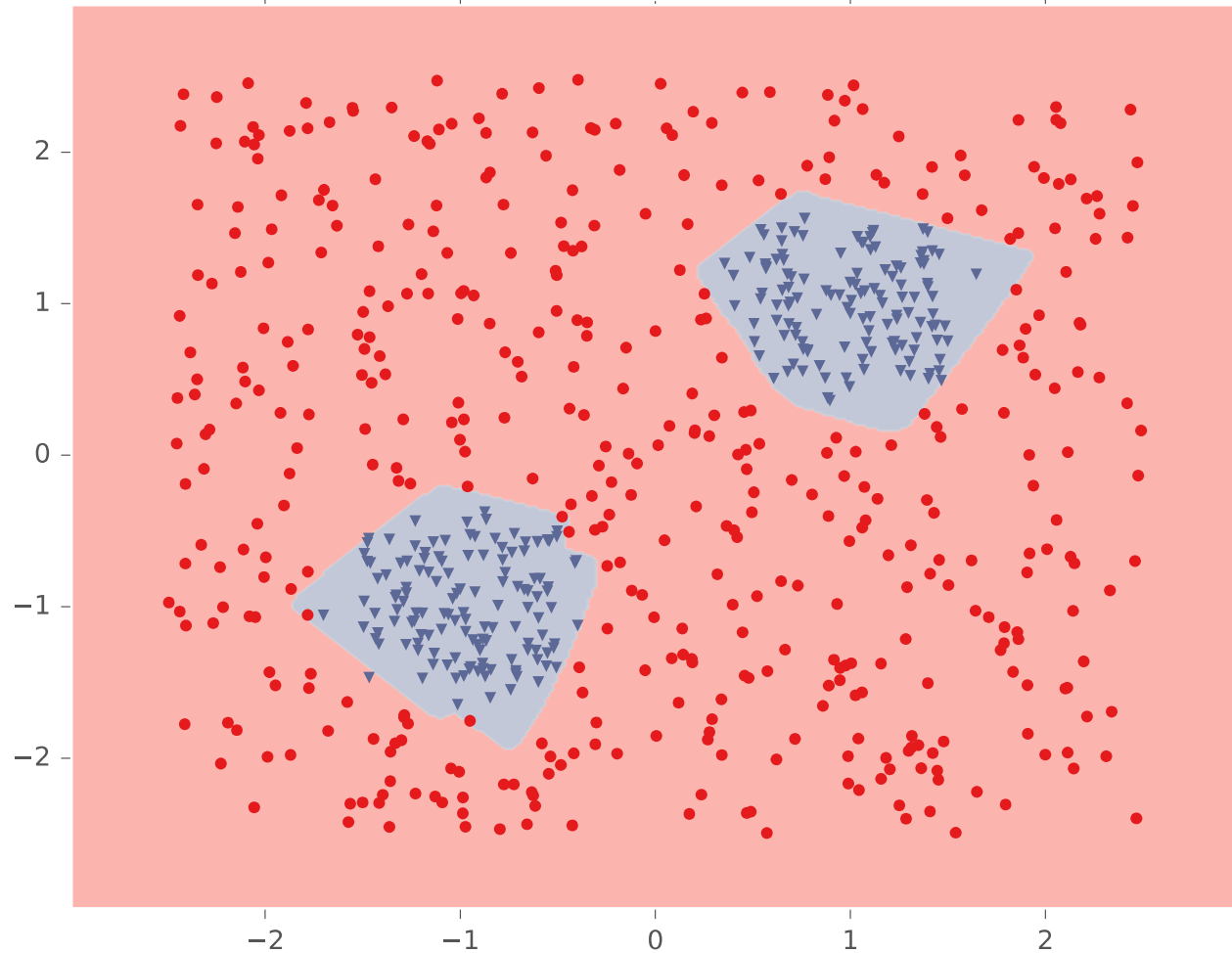
Example #4: Two Pockets

Tuned Neural Network (hidden=4, activation=logistic)



Example #4: Two Pockets

Tuned Neural Network (hidden=10, activation=logistic)



Neural Networks Objectives

You should be able to...

- Explain the biological motivations for a neural network
- Combine simpler models (e.g. linear regression, binary logistic regression, multinomial logistic regression) as components to build up feed-forward neural network architectures
- Explain the reasons why a neural network can model nonlinear decision boundaries for classification
- Compare and contrast feature engineering with learning features
- Identify (some of) the options available when designing the architecture of a neural network
- Implement a feed-forward neural network