10-301/601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Regularization
+
Neural Networks

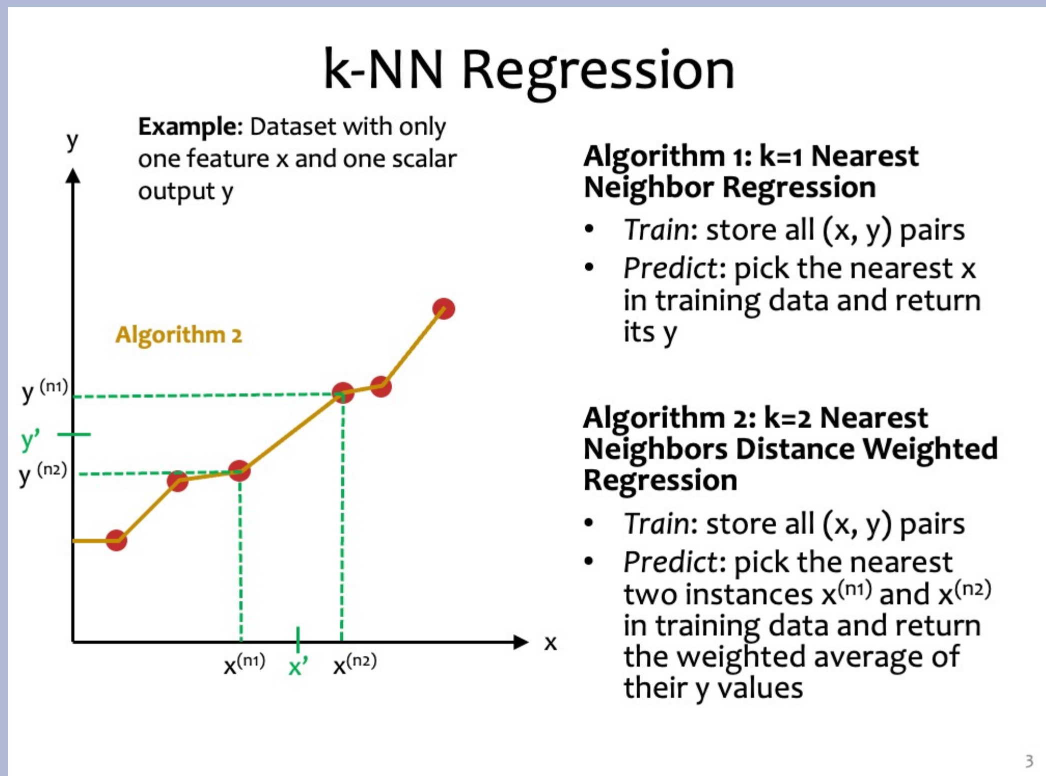
Matt Gormley
Lecture 11
Oct. 3, 2022

Reminders

- **Practice Problems: Exam 1**
- **Exam 1**
  - **Tue, Oct 4, 6:30pm – 8:30pm**
  - **see Piazza for exam location**
- **Homework 4: Logistic Regression**
  - **Out: Tue, Oct 4**
  - **Due: Thu, Oct 13 at 11:59pm**
- Be careful to submit to the correct submission slot on Gradescope!

Q&A

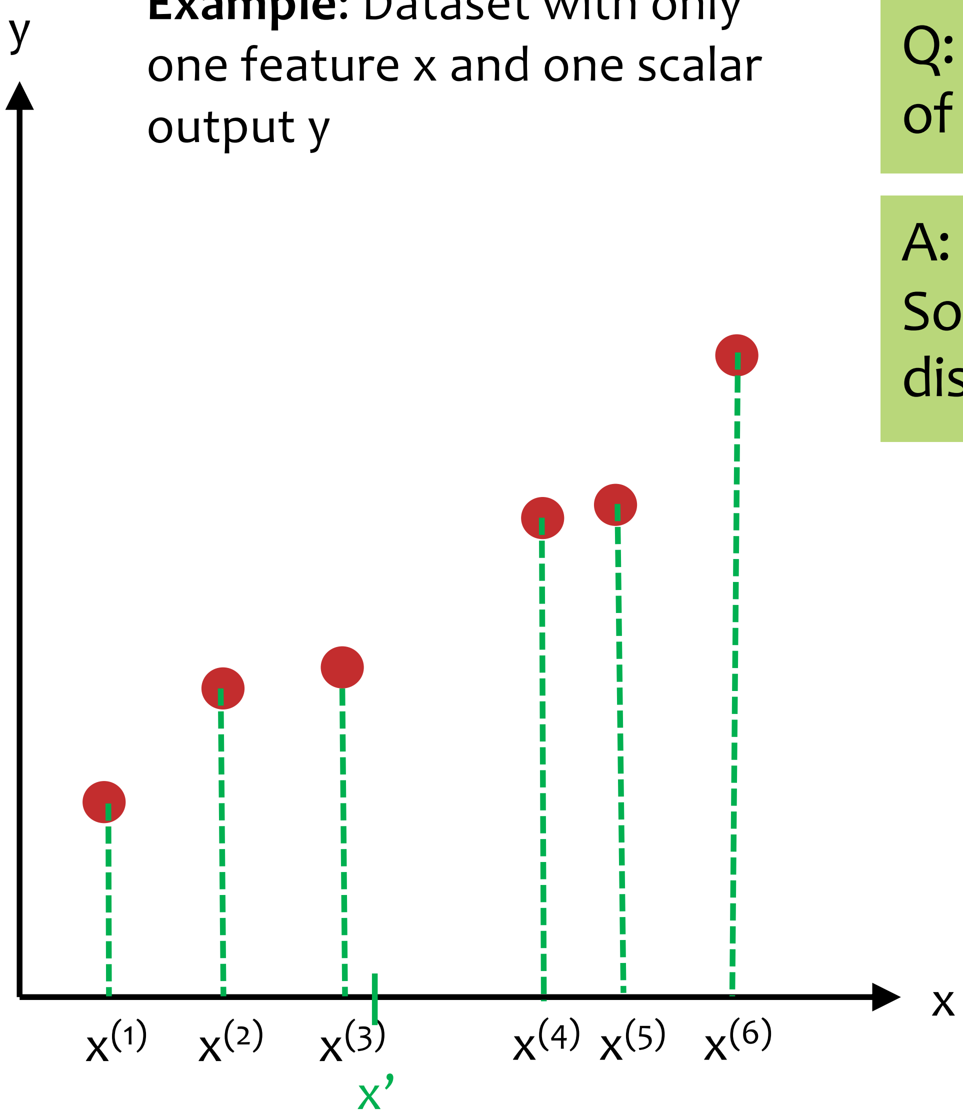
Q: Is this correct?



A: Oops! No.

k-NN Regression

Example: Dataset with only one feature x and one scalar output y



Q: What are the k=2 nearest neighbors of the test point x'?

A: $x^{(2)}$ and $x^{(3)}$.
So the output curve is actually discontinuous!

Algorithm 2: k=2 Nearest Neighbors Distance Weighted Regression

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest two instances $x^{(n1)}$ and $x^{(n2)}$ in training data and return the weighted average of their y values

REGULARIZATION

Overfitting

Definition: The problem of **overfitting** is when the model captures the noise in the training data instead of the underlying structure

Overfitting can occur in all the models we've seen so far:

- Decision Trees (e.g. when tree is too deep)
- KNN (e.g. when k is small)
- Perceptron (e.g. when sample isn't representative)
- Linear Regression (e.g. with nonlinear features)
- Logistic Regression (e.g. with many rare features)

Motivation: Regularization

- **Occam's Razor:** prefer the simplest hypothesis

- What does it mean for a hypothesis (or model) to be **simple**?
  1. small number of features (**model selection**)
  2. small number of "important" features (**shrinkage**)

Regularization

- **Given** objective function: $J(\theta)$
- **Goal** is to find: $\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}}\, J(\boldsymbol{\theta}) + \lambda r(\boldsymbol{\theta})$
- **Key idea:** Define regularizer $r(\theta)$ s.t. we tradeoff between fitting the data and keeping the model simple
- **Choose form of $r(\theta)$:**
  - Example: q-norm (usually p-norm): $\|\boldsymbol{\theta}\|_q = \left(\sum_{m=1}^{M} |\theta_m|^q\right)^{\frac{1}{q}}$

q	$r(\boldsymbol{\theta})$	**yields parameters that are...**	**name**	**optimization notes**
0	$\|\|\boldsymbol{\theta}\|\|_0 = \sum \mathbb{1}(\theta_m \neq 0)$	zero values	L0 reg.	no good computational solutions
1	$\|\|\boldsymbol{\theta}\|\|_1 = \sum \|\theta_m\|$	zero values	L1 reg.	subdifferentiable
2	$(\|\|\boldsymbol{\theta}\|\|_2)^2 = \sum \theta_m^2$	small values	L2 reg.	differentiable

Regularization Examples

Add an **L2 regularizer** to Linear Regression (aka. Ridge Regression)

$$J_{\text{RR}}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \boxed{\lambda ||\boldsymbol{\theta}||_2^2}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)})^2 + \boxed{\lambda \sum_{m=1}^{M} \theta_m^2}$$

Add an **L1 regularizer** to Linear Regression (aka. LASSO)

$$J_{\text{LASSO}}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \boxed{\lambda ||\boldsymbol{\theta}||_1}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)})^2 + \boxed{\lambda \sum_{m=1}^{M} |\theta_m|}$$

Regularization Examples

Add an **L2 regularizer** to Logistic Regression

$$
\begin{aligned}
J'(\boldsymbol{\theta}) &= J(\boldsymbol{\theta}) + \boxed{\lambda ||\boldsymbol{\theta}||_2^2} \\
&= \frac{1}{N} \sum_{i=1}^{N} - \log p(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}) + \boxed{\lambda \sum_{m=1}^{M} \theta_m^2}
\end{aligned}
$$

Add an **L1 regularizer** to Logistic Regression

$$
\begin{aligned}
J'(\boldsymbol{\theta}) &= J(\boldsymbol{\theta}) + \boxed{\lambda ||\boldsymbol{\theta}||_1} \\
&= \frac{1}{N} \sum_{i=1}^{N} - \log p(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}) + \boxed{\lambda \sum_{m=1}^{M} |\theta_m|}
\end{aligned}
$$

Regularization

Question:

Suppose we are minimizing J'(θ) where

$$J'(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \lambda r(\boldsymbol{\theta})$$

As λ increases, the minimum of J'(θ) will…

A. … move towards the midpoint between J(θ) and r(θ)

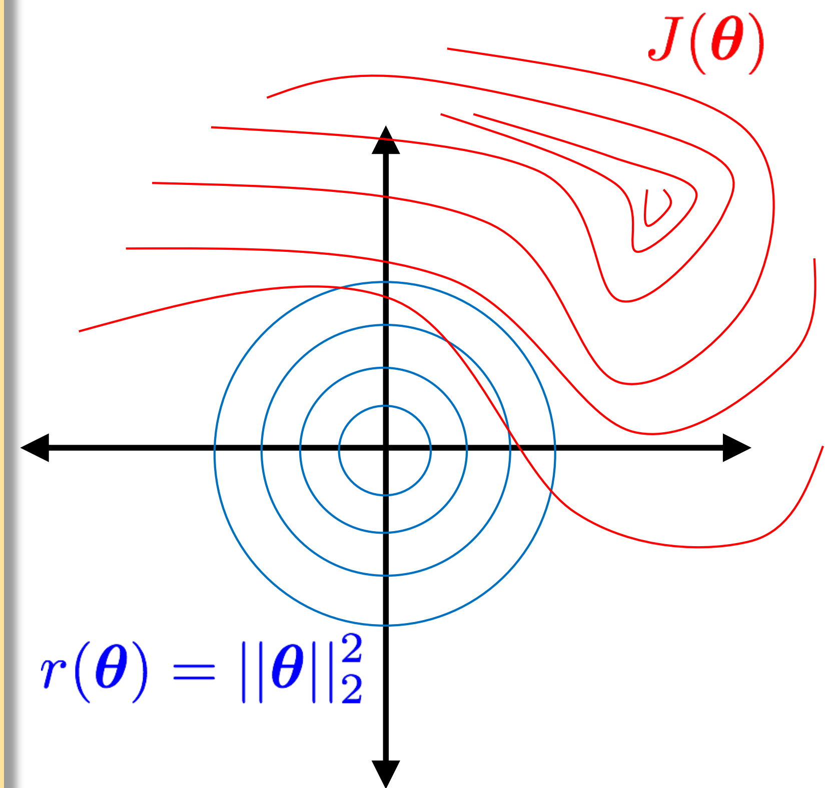
B. … move towards the minimum of J(θ)

C. … move towards the minimum of r(θ)

D. … move towards a theta vector of positive infinities

E. … move towards a theta vector of negative infinities

F. … stay the same



Regularization

Whiteboard

– Why does L2 regularization lead to small numbers and L1 regularization lead to zeros?

Regularization

Don't Regularize the Bias (Intercept) Parameter!

- In our models so far, the bias / intercept parameter is usually denoted by θ_0 -- that is, the parameter for which we fixed $x_0 = 1$
- Regularizers always avoid penalizing this bias / intercept parameter
- Why? Because otherwise the learning algorithms wouldn't be invariant to a shift in the y-values

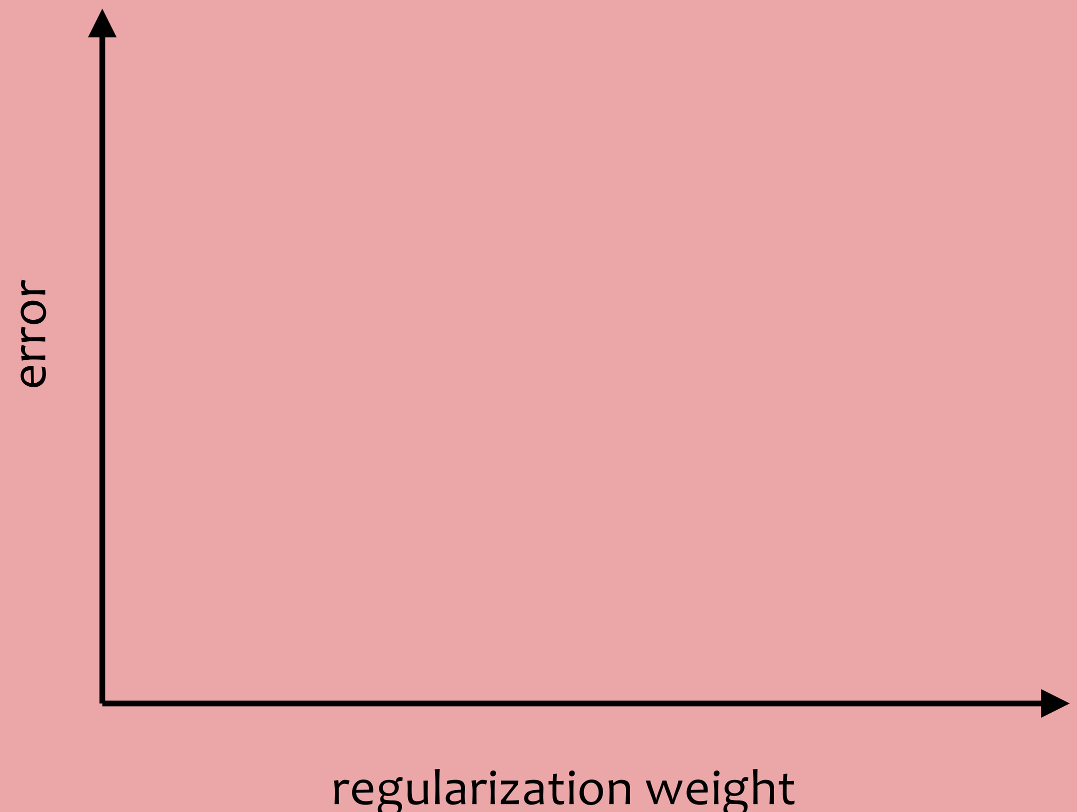
Whitening Data

- It's common to *whiten* each feature by subtracting its mean and dividing by its variance
- For regularization, this helps all the features be penalized in the same units
(e.g. convert both centimeters and kilometers to z-scores)

Regularization Exercise

In-class Exercise

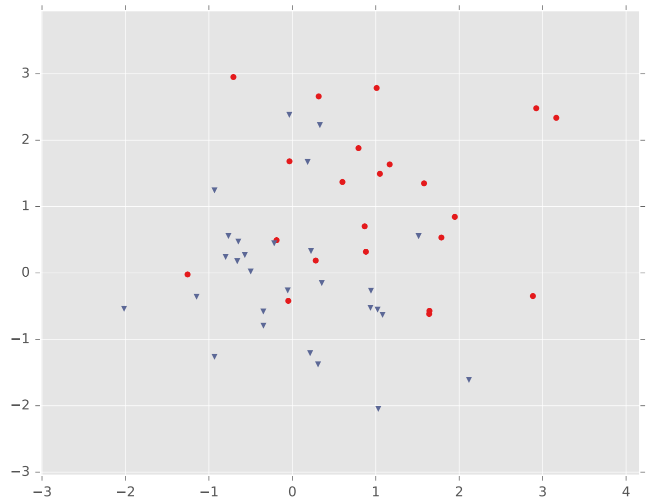
1. Plot train error vs. regularization weight (cartoon)
2. Plot validation error vs . regularization weight (cartoon)



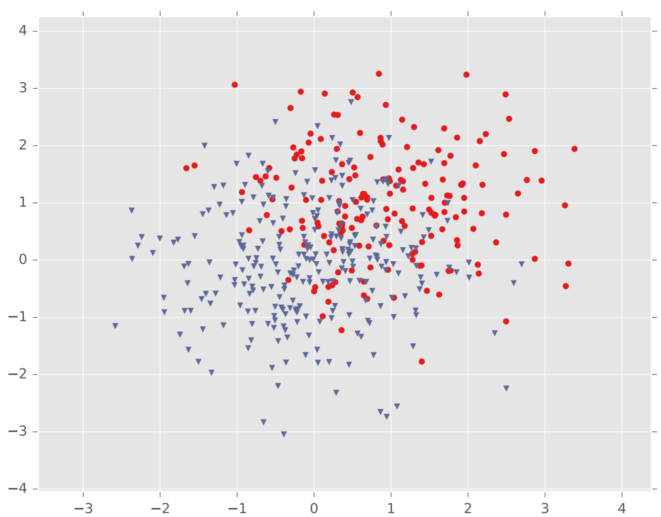
REGULARIZATION EXAMPLE: LOGISTIC REGRESSION

Example: Logistic Regression

Training Data

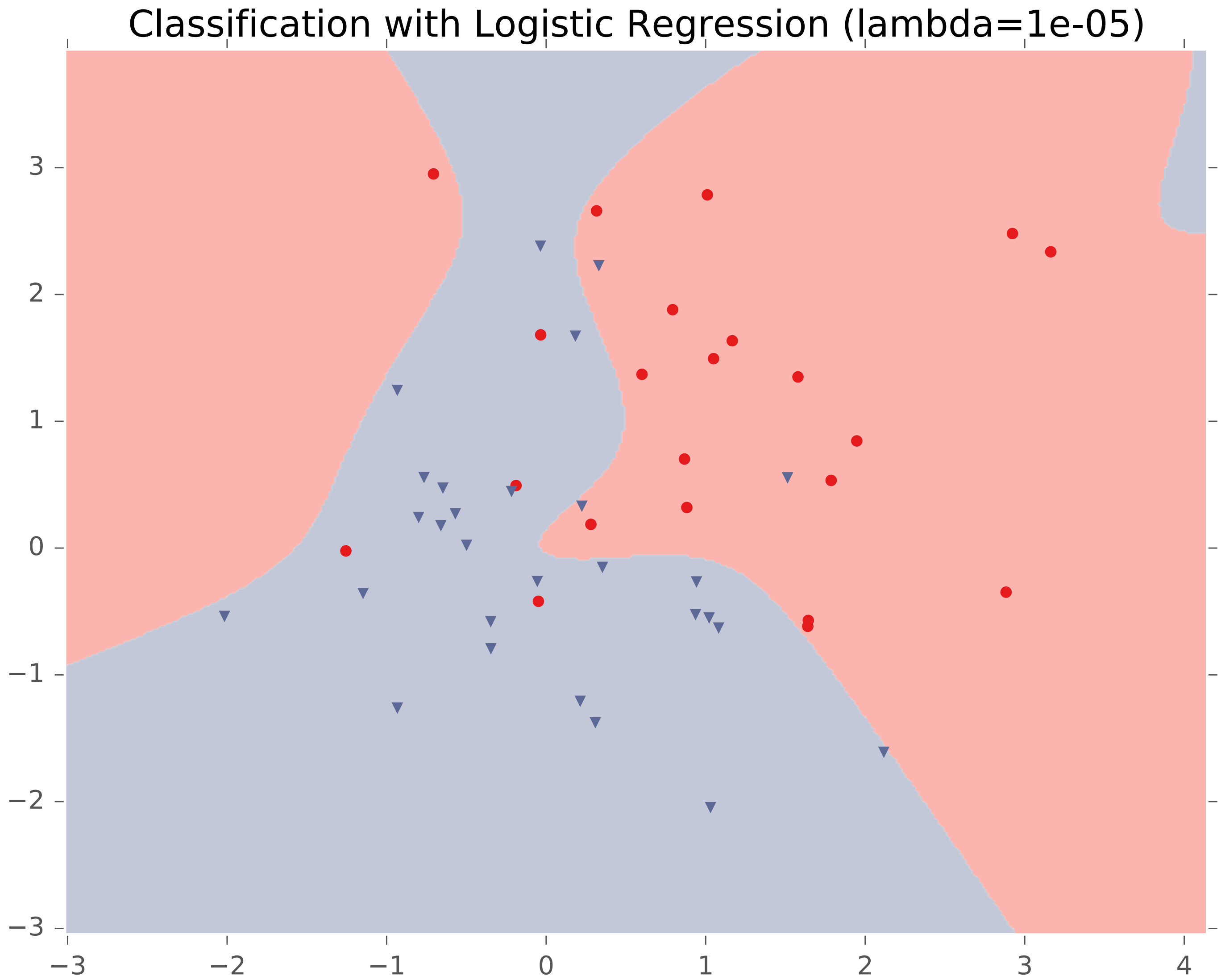


Test Data

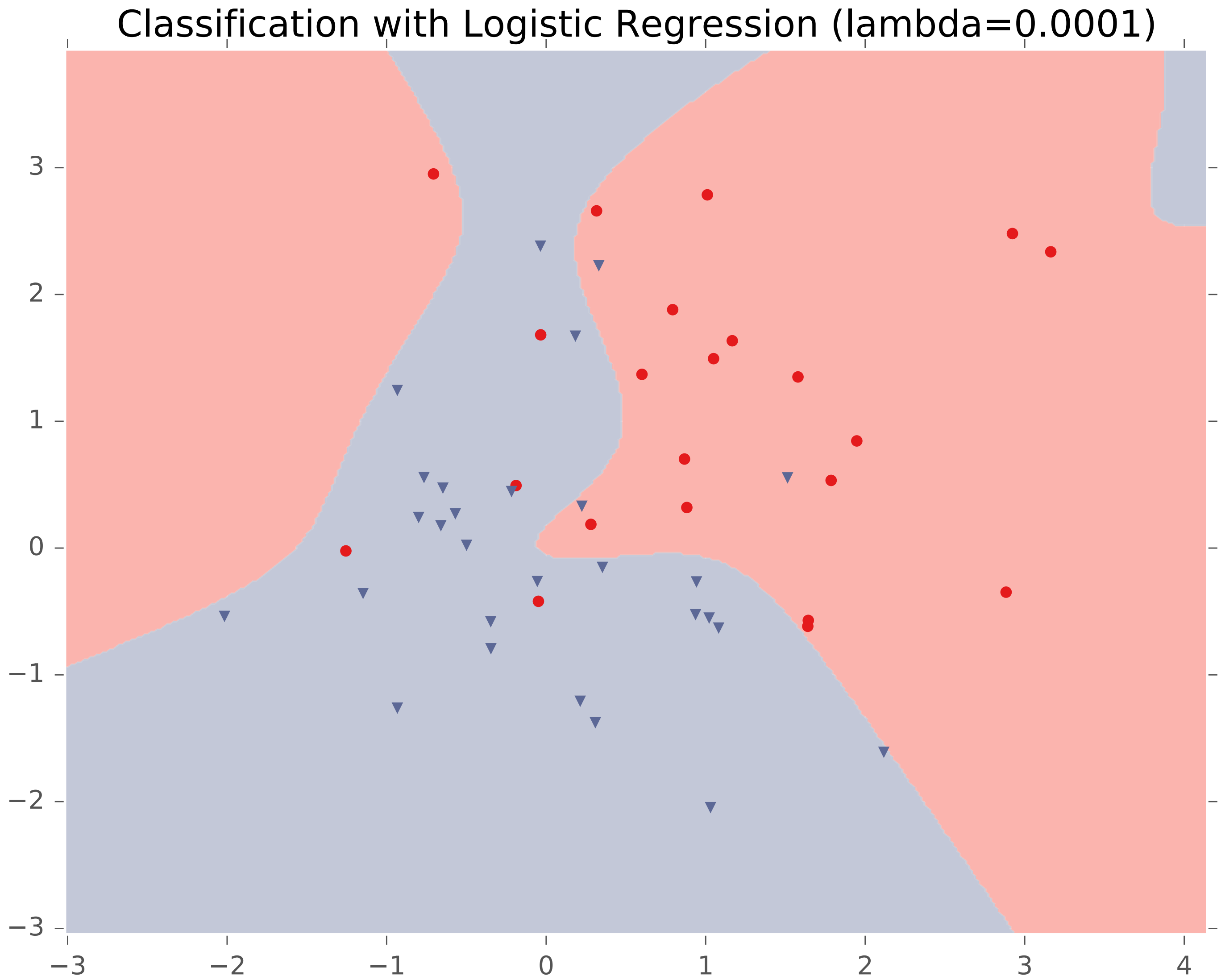


- For this example, we construct **nonlinear features** (i.e. feature engineering)
- Specifically, we add **polynomials up to order 9** of the two original features x_1 and x_2
- Thus our classifier is **linear** in the **high-dimensional feature space**, but the decision boundary is **nonlinear** when visualized in **low-dimensions** (i.e. the original two dimensions)

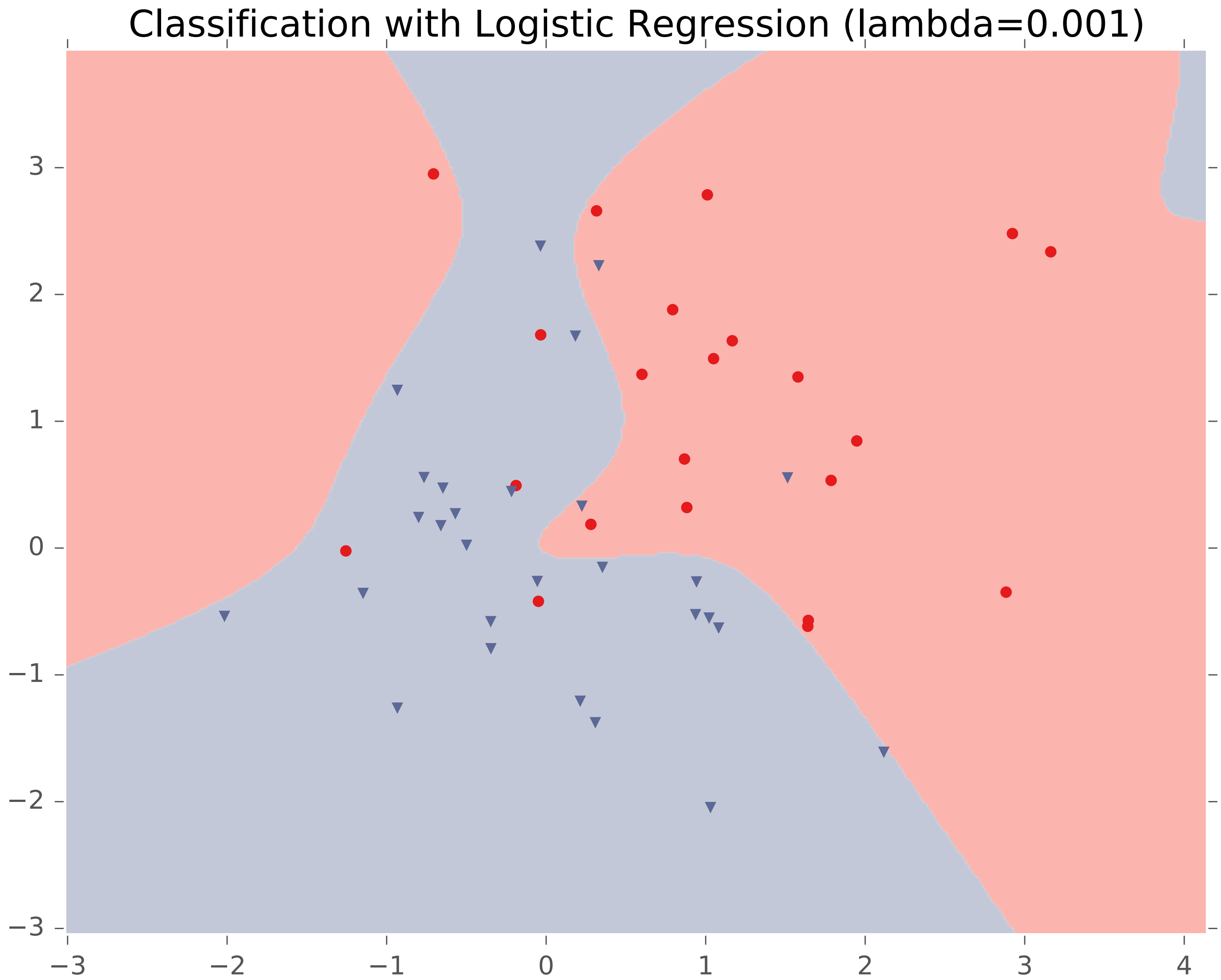
Example: Logistic Regression



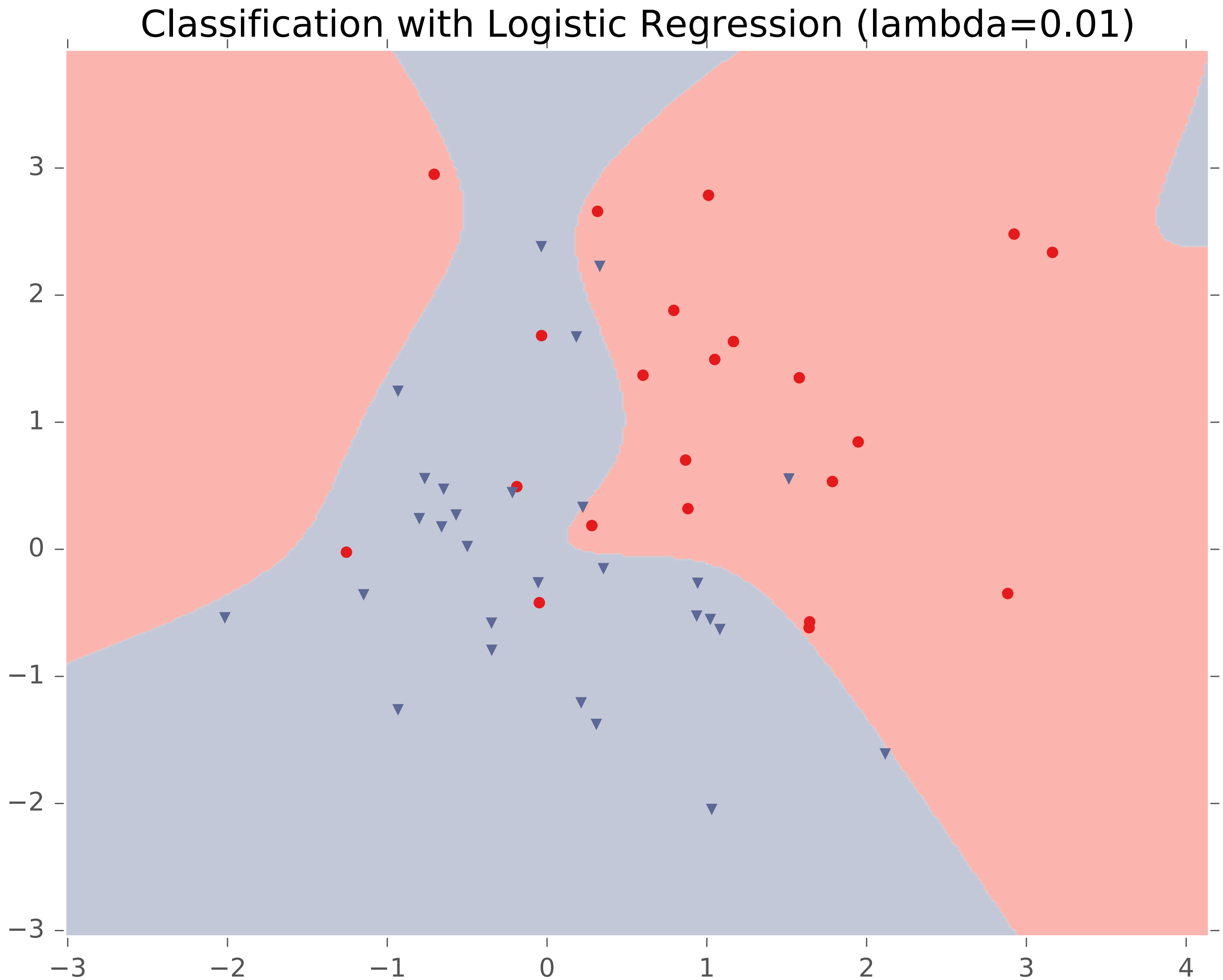
Example: Logistic Regression



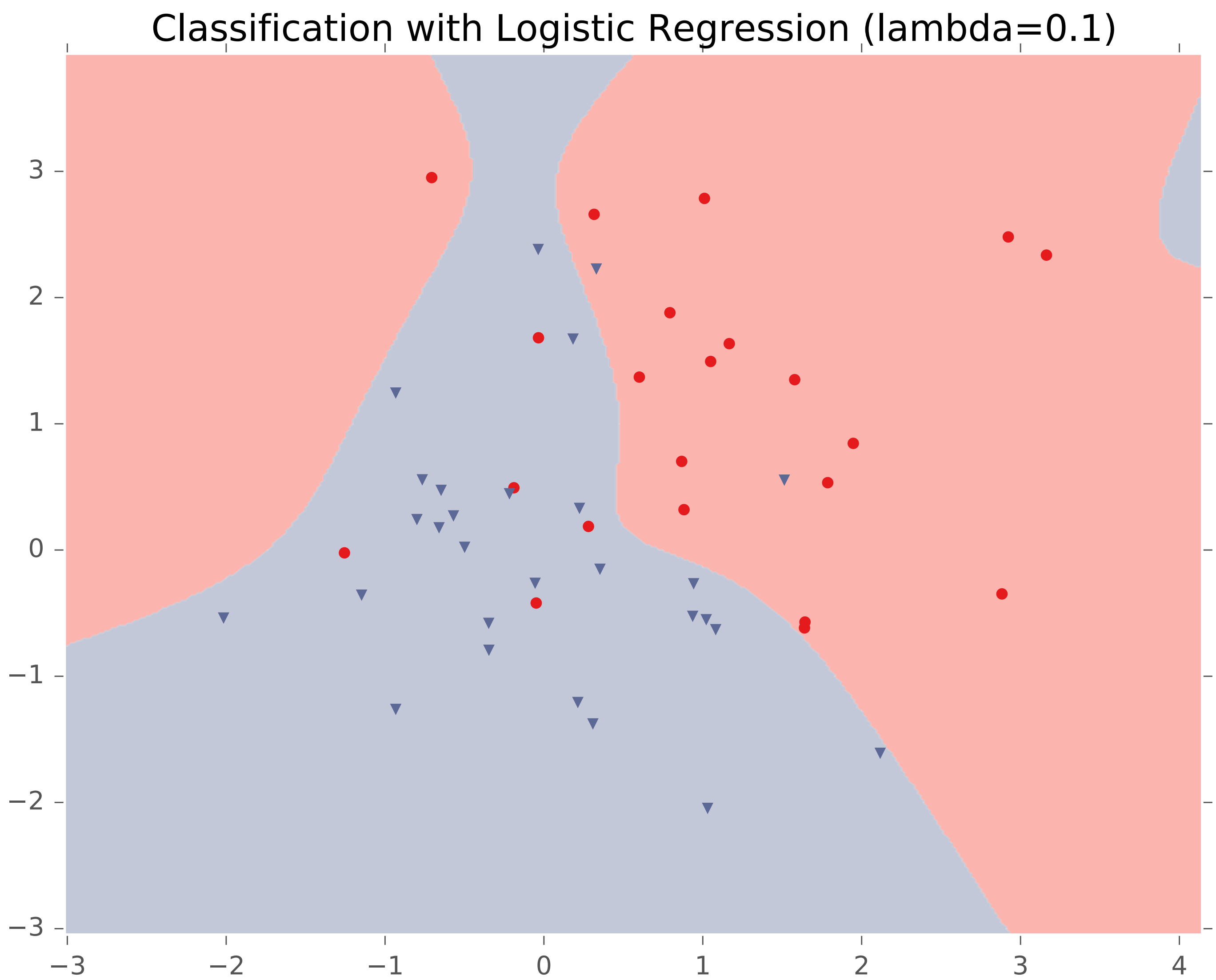
Example: Logistic Regression



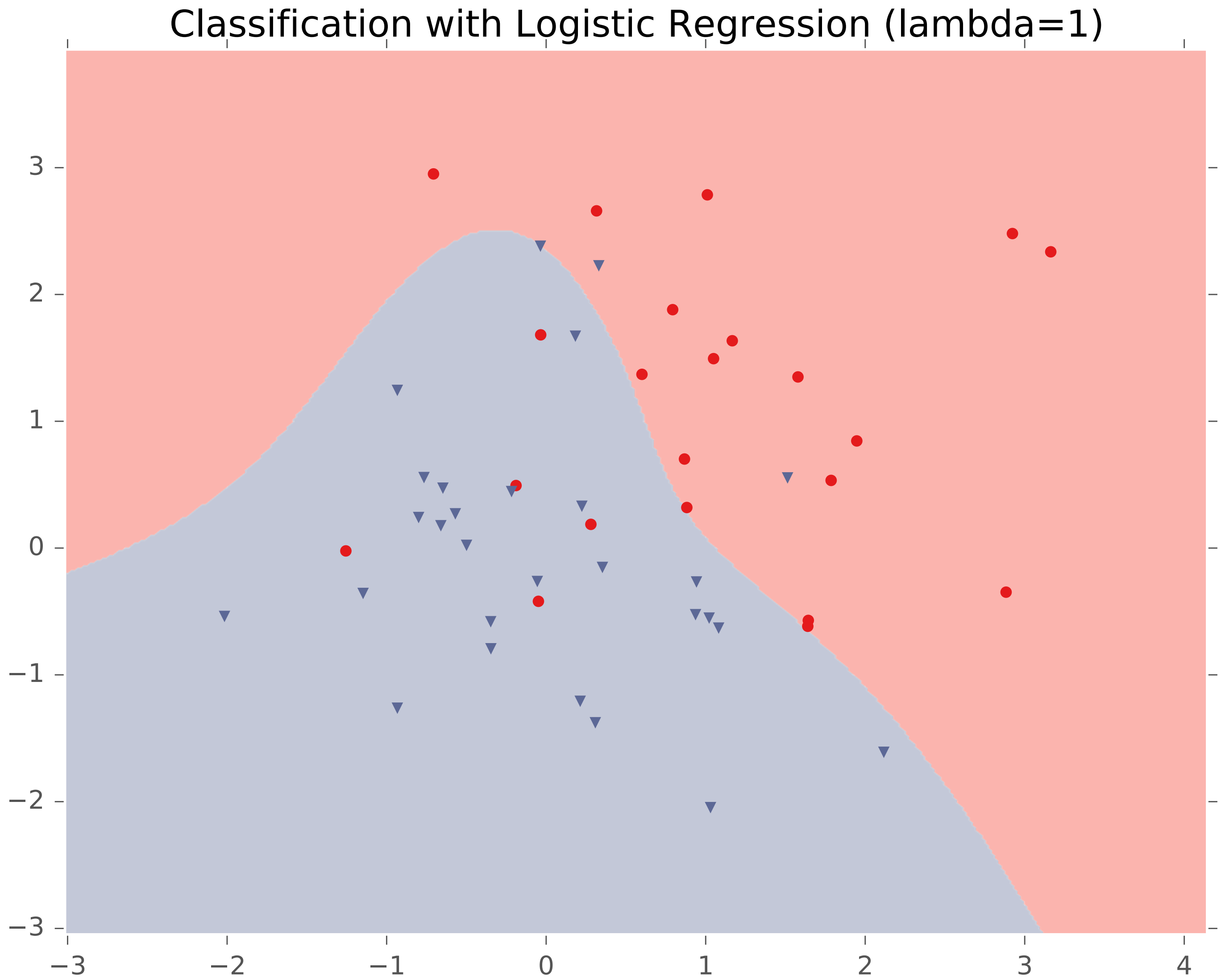
Example: Logistic Regression



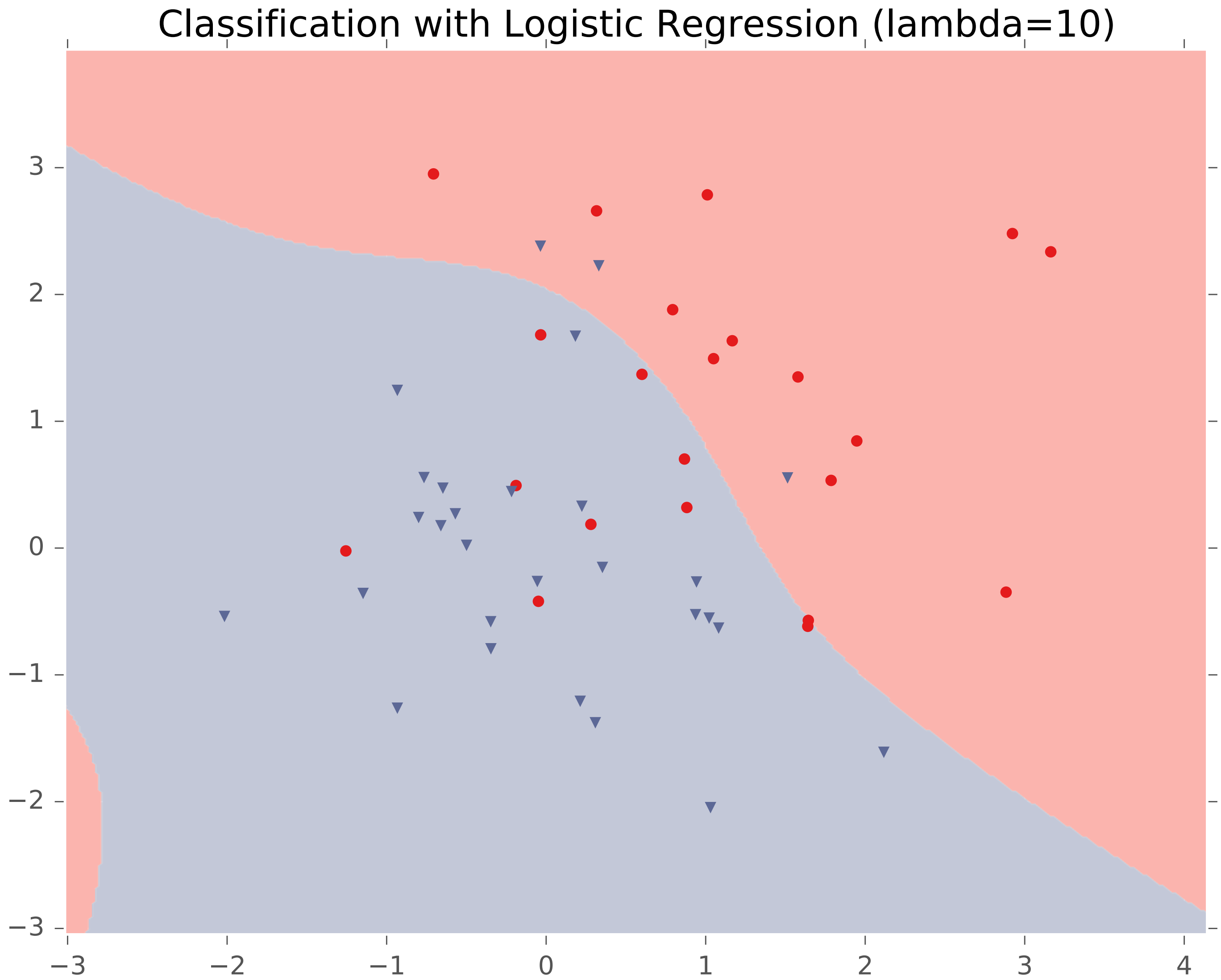
Example: Logistic Regression



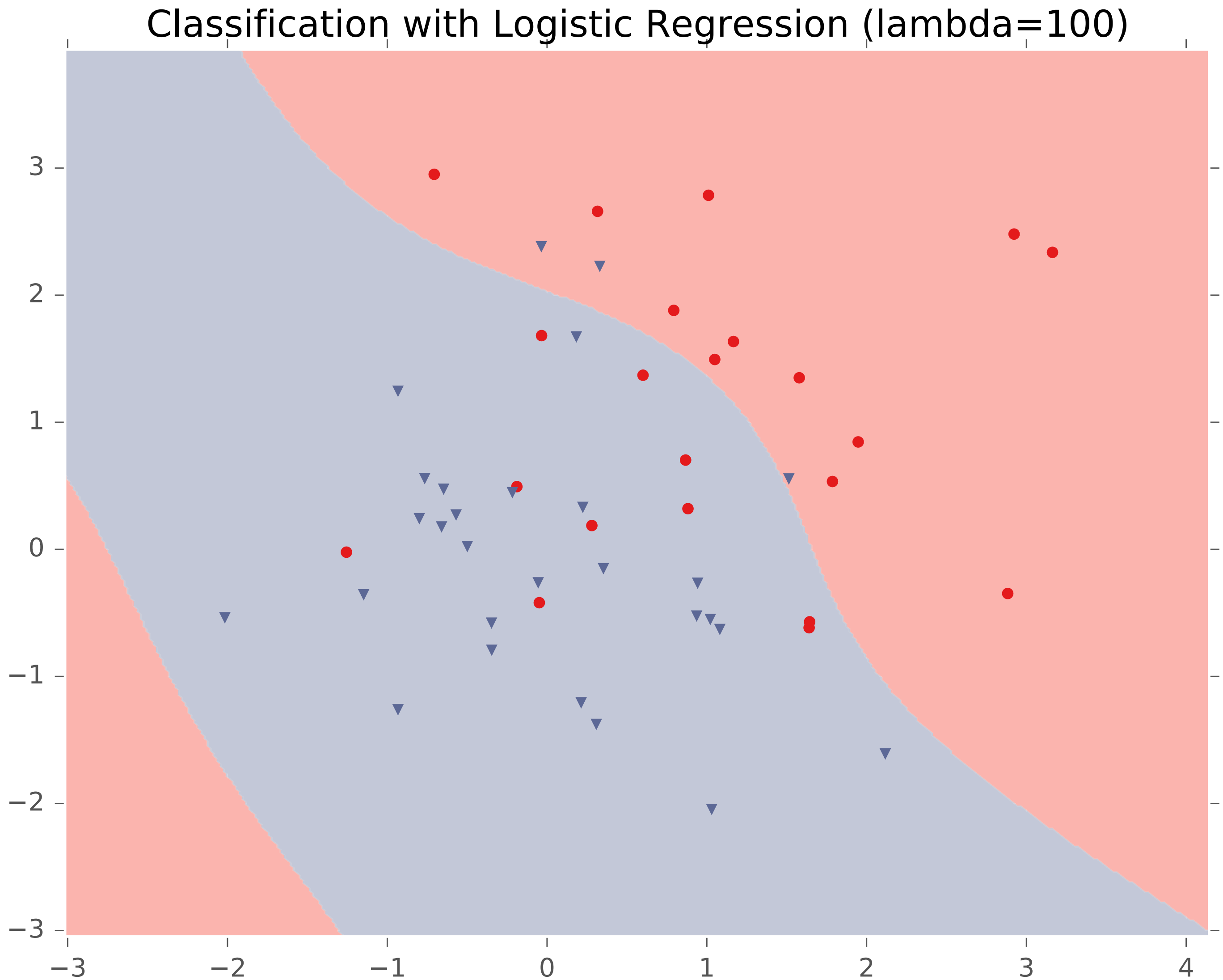
Example: Logistic Regression



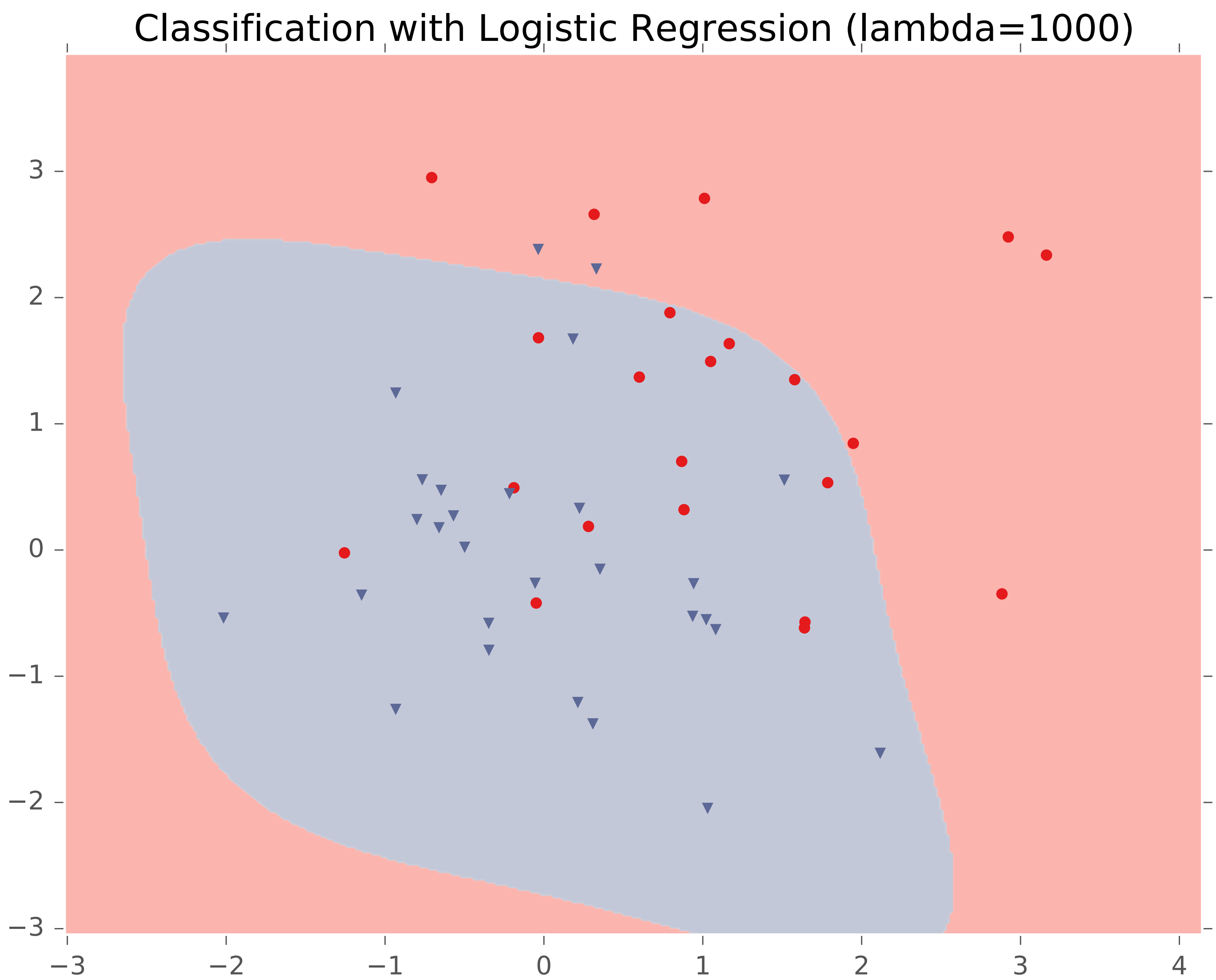
Example: Logistic Regression



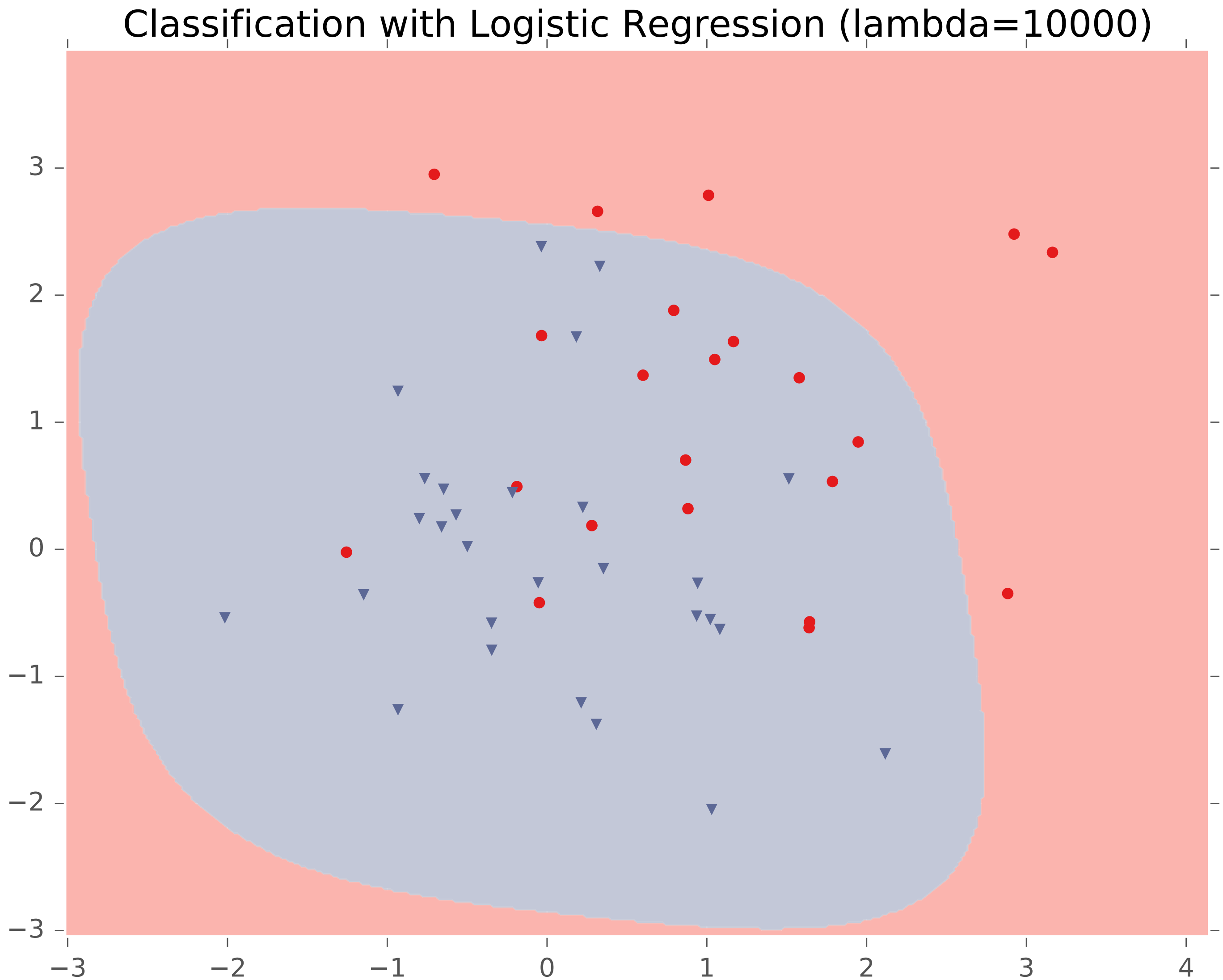
Example: Logistic Regression



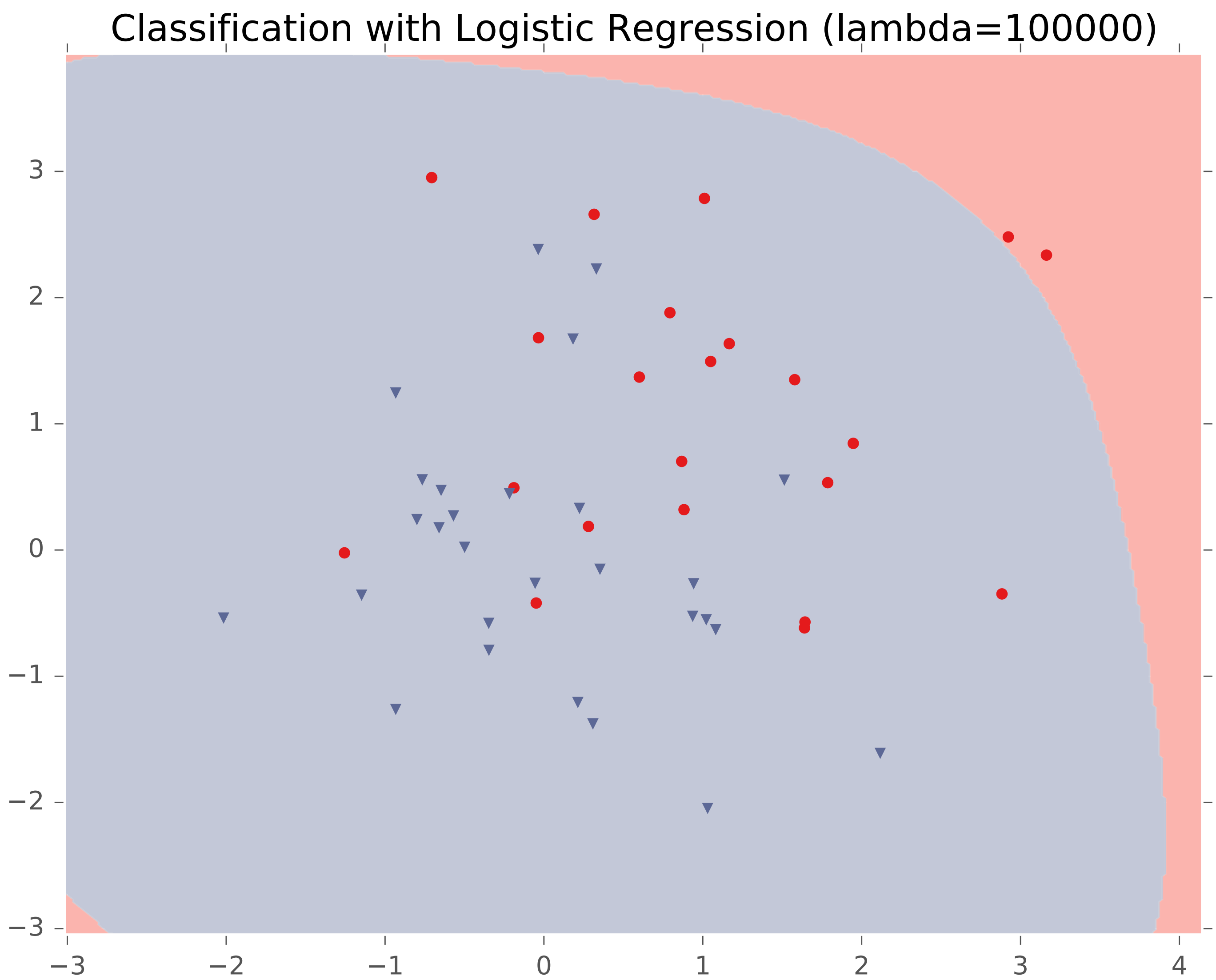
Example: Logistic Regression



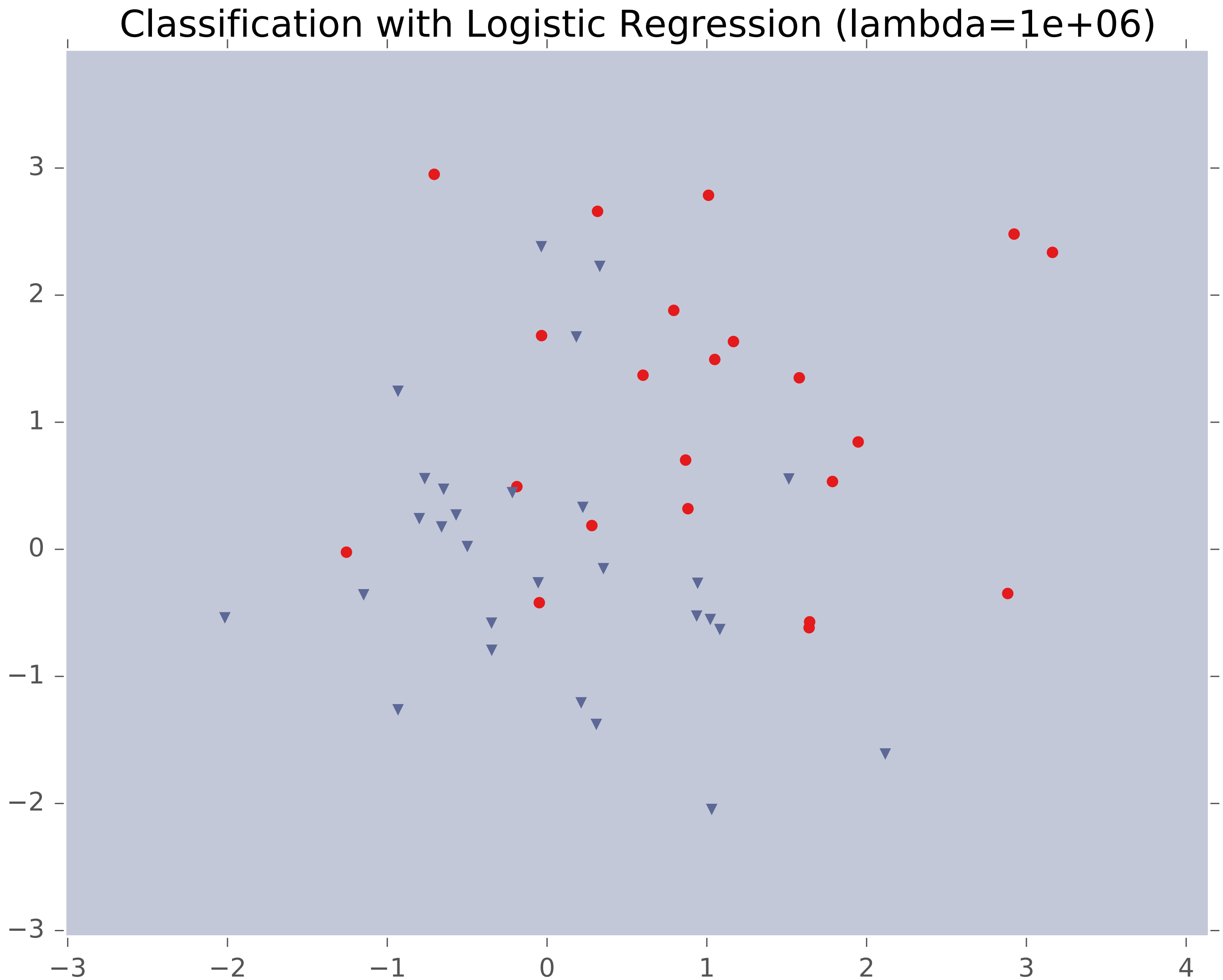
Example: Logistic Regression



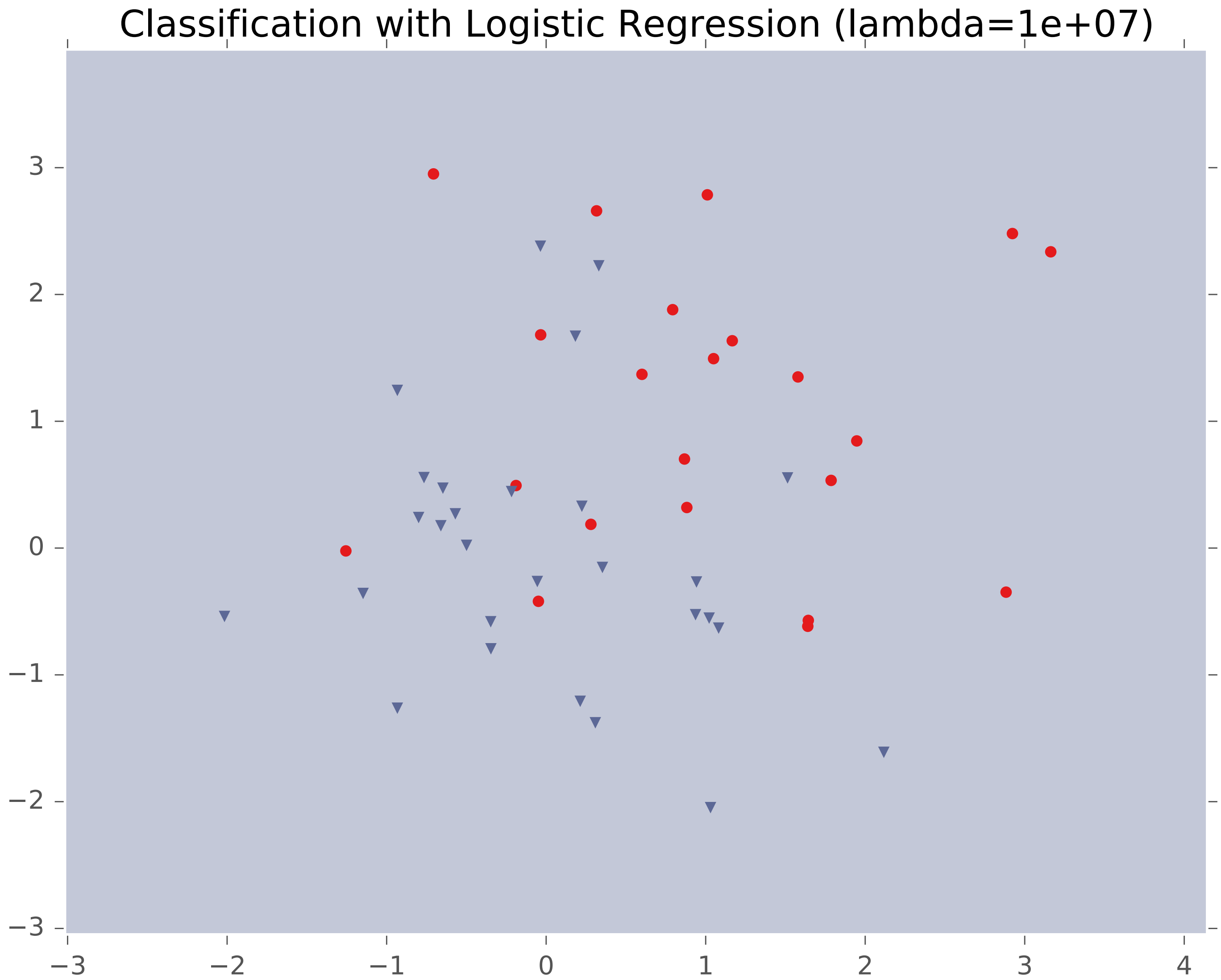
Example: Logistic Regression



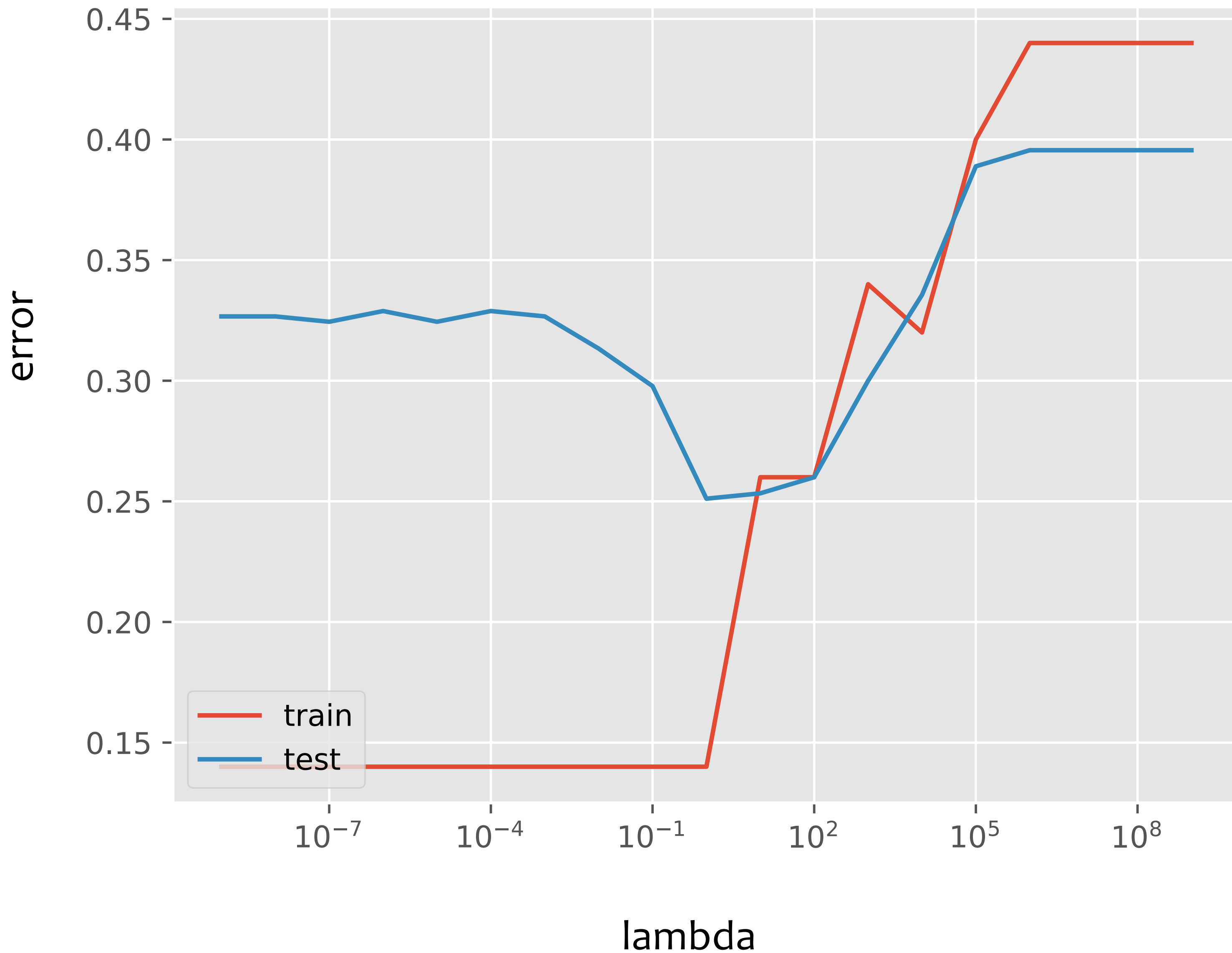
Example: Logistic Regression



Example: Logistic Regression



Example: Logistic Regression



OPTIMIZATION FOR L1 REGULARIZATION

Optimization for L1 Regularization

Can we apply SGD to the LASSO learning problem?

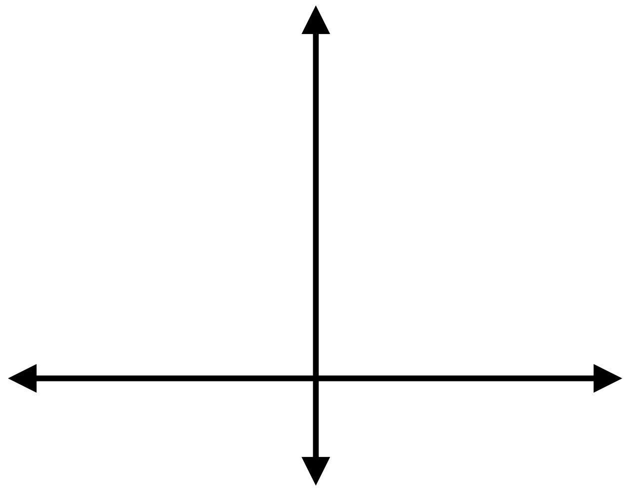
$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \quad J_{\text{LASSO}}(\boldsymbol{\theta})$$

$$\begin{aligned} J_{\text{LASSO}}(\boldsymbol{\theta}) &= J(\boldsymbol{\theta}) + \boxed{\lambda ||\boldsymbol{\theta}||_1} \\ &= \frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2 + \boxed{\lambda \sum_{k=1}^{K} |\theta_k|} \end{aligned}$$

Optimization for L1 Regularization

- Consider the absolute value function:

$$r(\boldsymbol{\theta}) = \lambda \sum_{k=1}^{K} |\theta_k|$$



- The L1 penalty is subdifferentiable (i.e. not differentiable at 0)

Def: A vector $g \in \mathbb{R}^M$ is called a **subgradient** of a function $f(\mathbf{x}) : \mathbb{R}^M \to \mathbb{R}$ at the point $\mathbf{x}$ if, for all $\mathbf{x}' \in \mathbb{R}^M$, we have:

$$f(\mathbf{x}') \geq f(\mathbf{x}) + \mathbf{g}^T(\mathbf{x}' - \mathbf{x})$$

Optimization for L1 Regularization

- The L1 penalty is subdifferentiable (i.e. not differentiable at 0)
- An array of optimization algorithms exist to handle this issue:
  - Subgradient descent
  - Stochastic subgradient descent
  - Coordinate Descent
  - Othant-Wise Limited memory Quasi-Newton (OWL-QN) (Andrew & Gao, 2007) and provably convergent variants
  - Block coordinate Descent (Tseng & Yun, 2009)
  - Sparse Reconstruction by Separable Approximation (SpaRSA) (Wright et al., 2009)
  - Fast Iterative Shrinkage Thresholding Algorithm (FISTA) (Beck & Teboulle, 2009)

Basically the same as GD and SGD, but you use one of the subgradients when necessary

Regularization as MAP

- L1 and L2 regularization can be interpreted as **maximum a-posteriori (MAP) estimation** of the parameters
- To be discussed later in the course…

Takeaways

1. **Nonlinear basis functions** allow **linear models** (e.g. Linear Regression, Logistic Regression) to capture **nonlinear** aspects of the original input
2. Nonlinear features are **require no changes to the model** (i.e. just preprocessing)
3. **Regularization** helps to avoid **overfitting**
4. **Regularization** and **MAP estimation** are equivalent for appropriately chosen priors

Feature Engineering / Regularization Objectives

You should be able to…

- Engineer appropriate features for a new task
- Use feature selection techniques to identify and remove irrelevant features
- Identify when a model is overfitting
- Add a regularizer to an existing objective in order to combat overfitting
- Explain why we should **not** regularize the bias term
- Convert linearly inseparable dataset to a linearly separable dataset in higher dimensions
- Describe feature engineering in common application areas

NEURAL NETWORKS

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

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

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:

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

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)

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

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!

Background

A Recipe for

1. Given training dat

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

2. Choose each of t

– Decision functio

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

– Loss function

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

$\boldsymbol{y}_i)$

opp site 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)

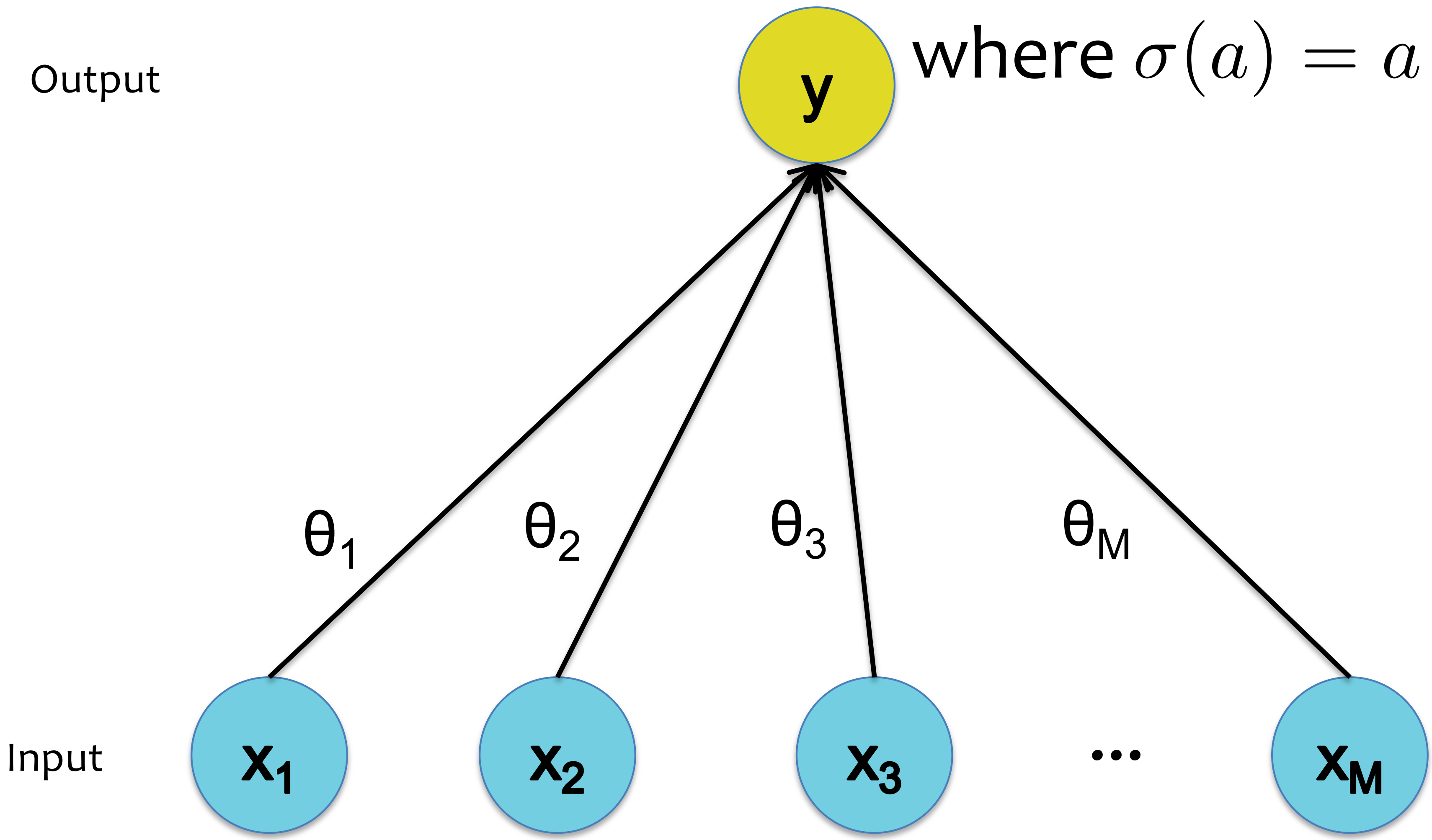
$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

Decision Functions

Linear Regression

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

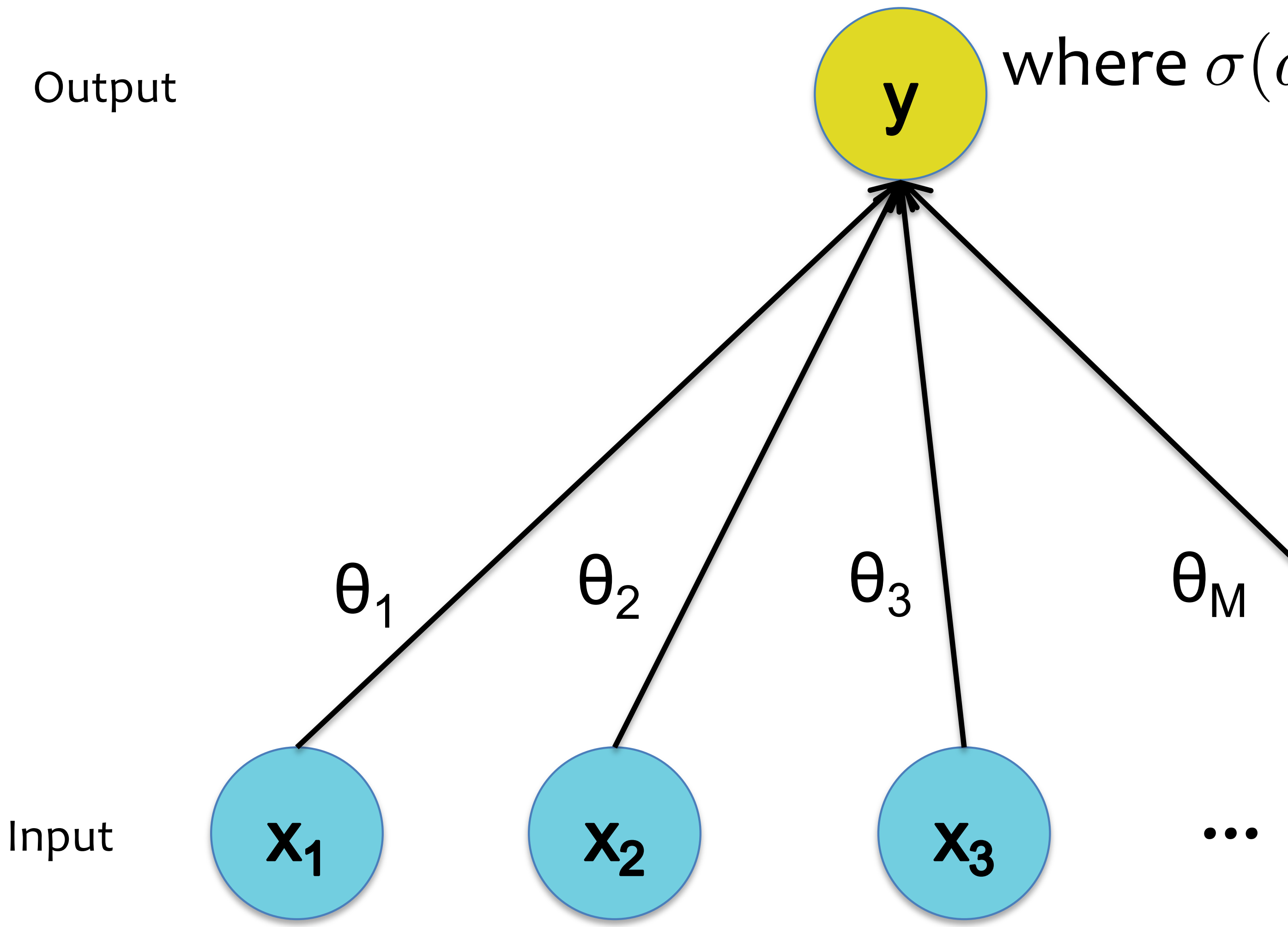
where $\sigma(a) = a$

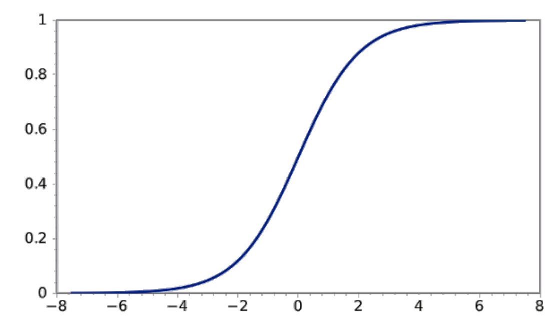


Logistic Regression

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

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



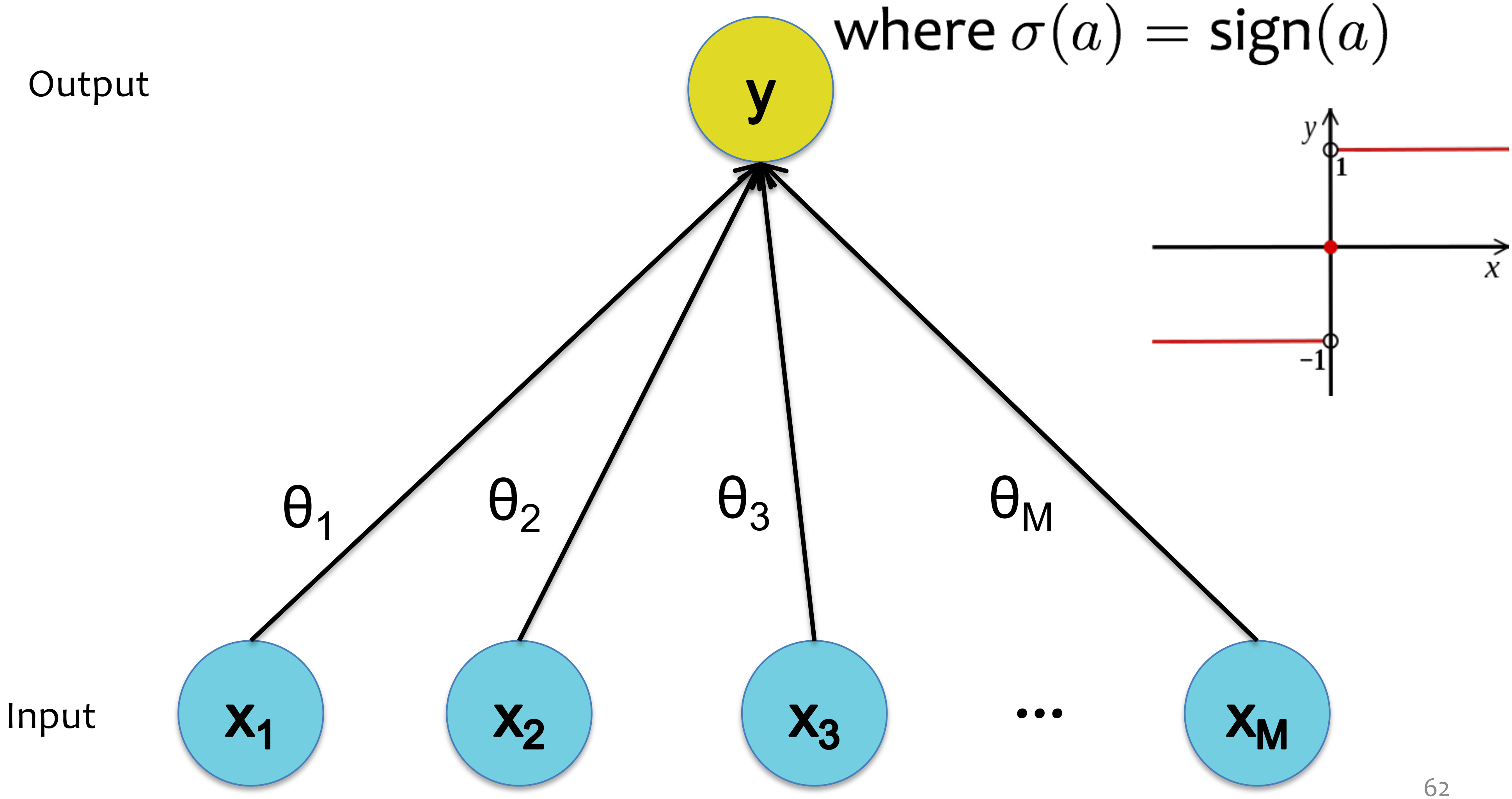


Decision Functions

Perceptron

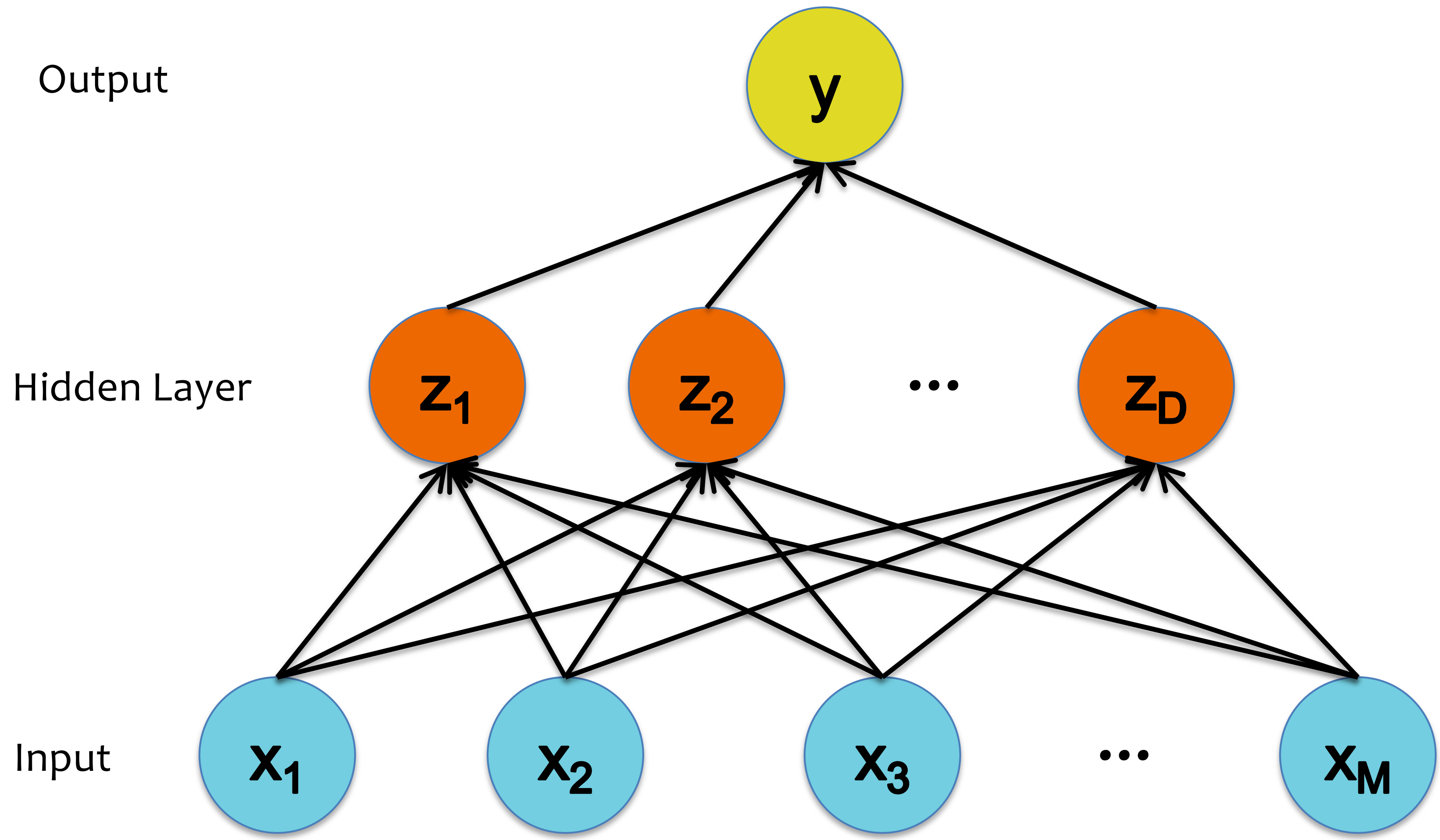
$$y = h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(\boldsymbol{\theta}^T \boldsymbol{x})$$

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



Decision Functions

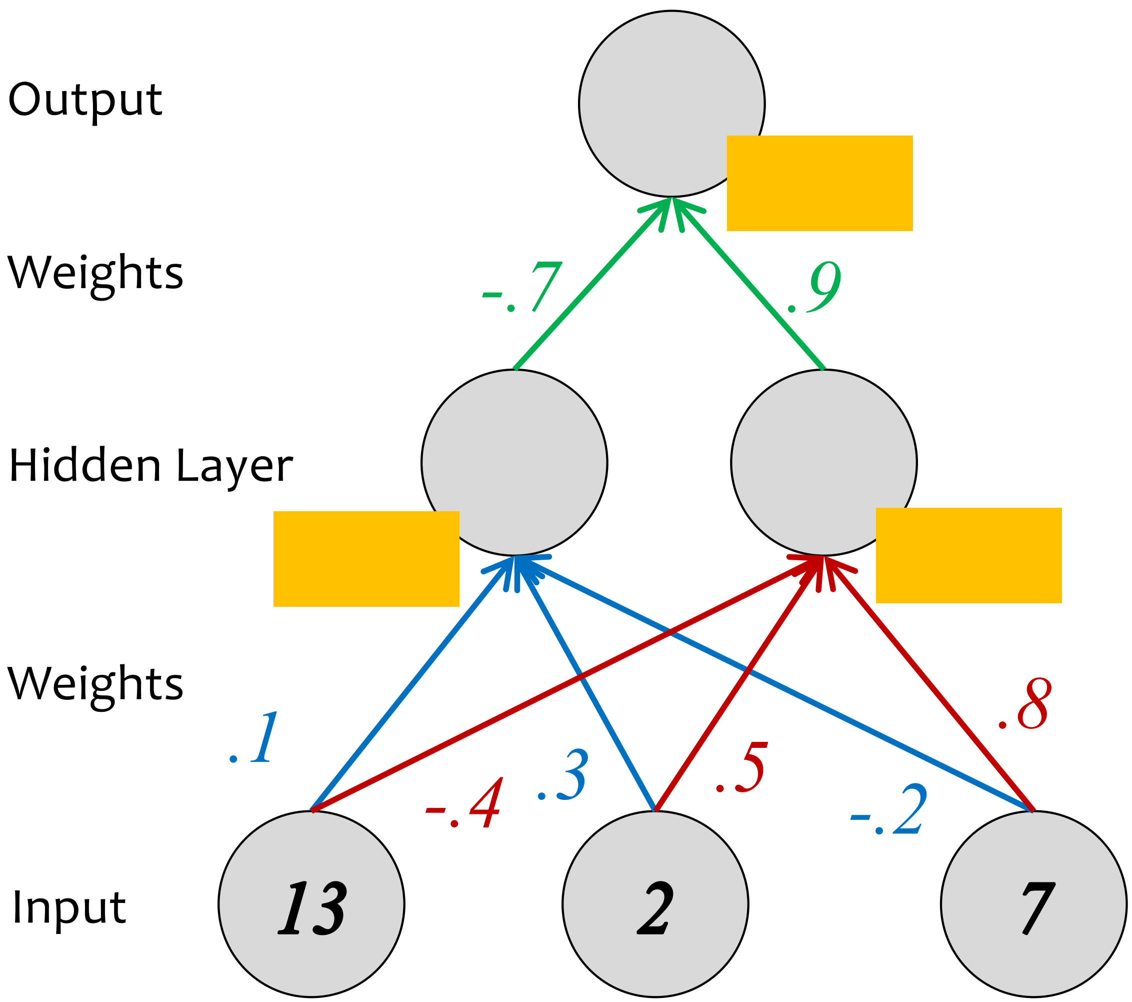
Neural Network



COMPONENTS OF A NEURAL NETWORK

Decision Functions

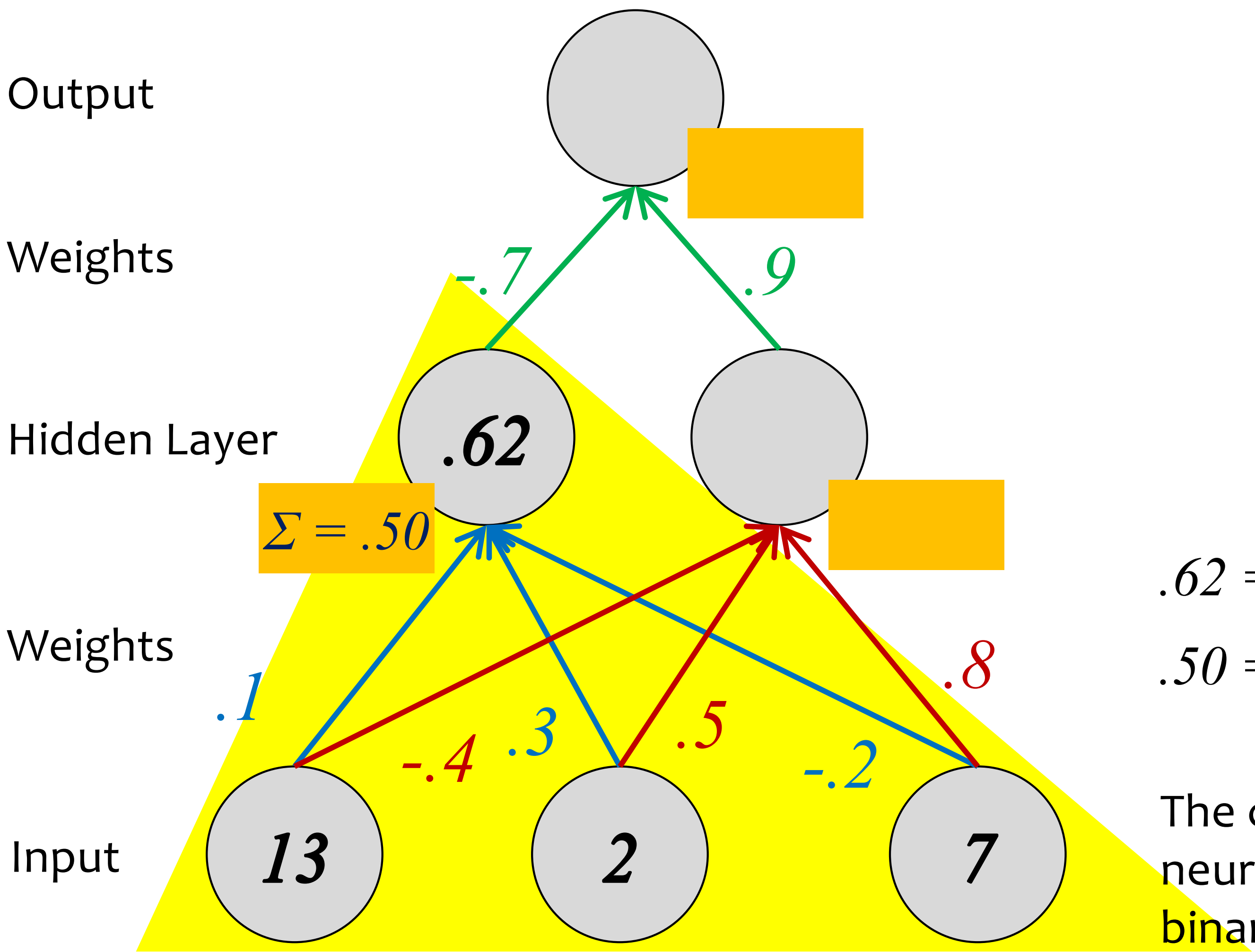
Neural Network



Suppose we already learned the weights of the neural network.

To make a new prediction, we take in some new features (aka. the input layer) and perform the feed-forward computation.

Neural Network

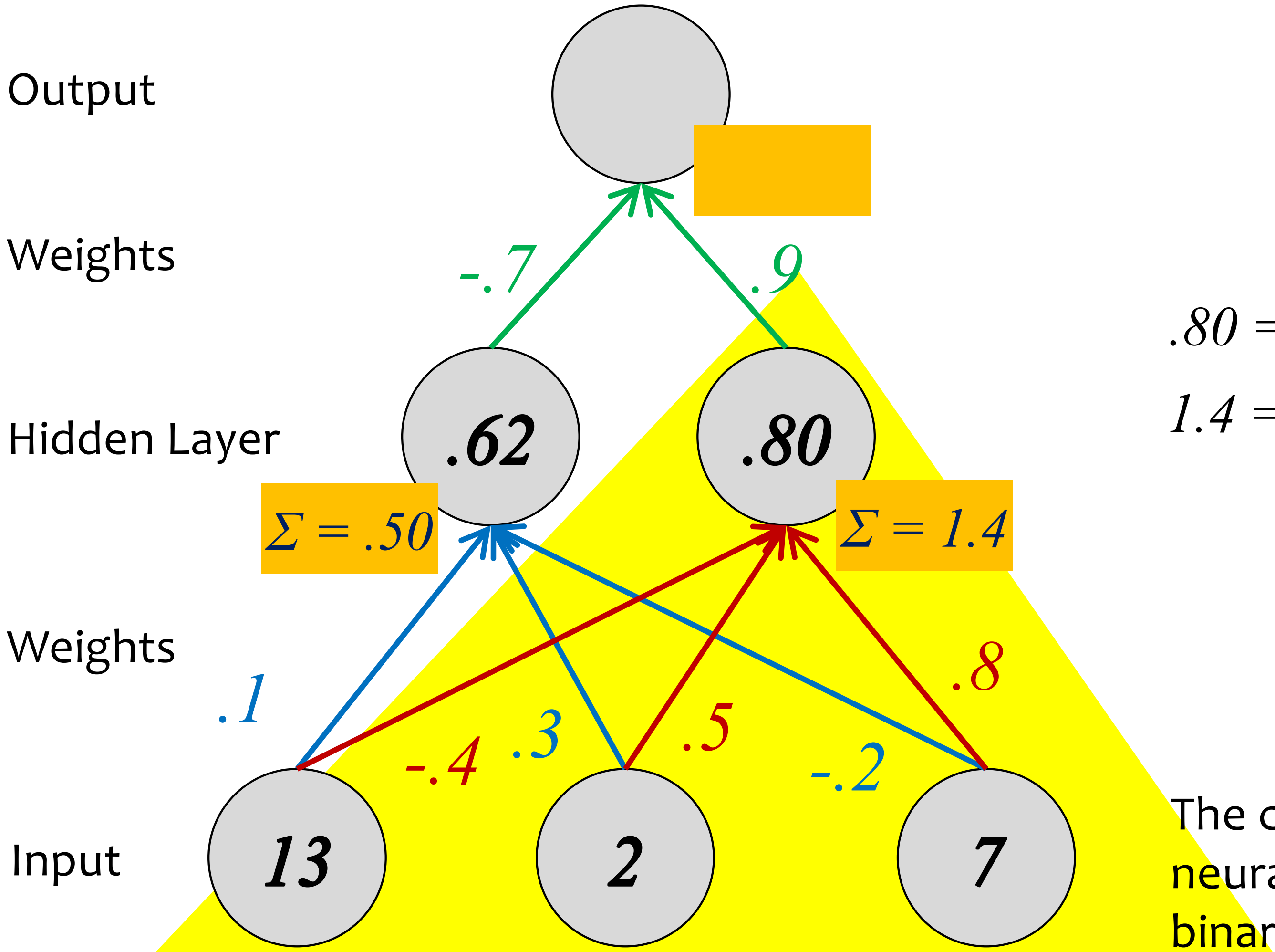


$.62 = \sigma(.50)$

$.50 = 13(.1) + 2(.3) + 7(-.2)$

The computation of each neural network unit resembles binary logistic regression.

Neural Network



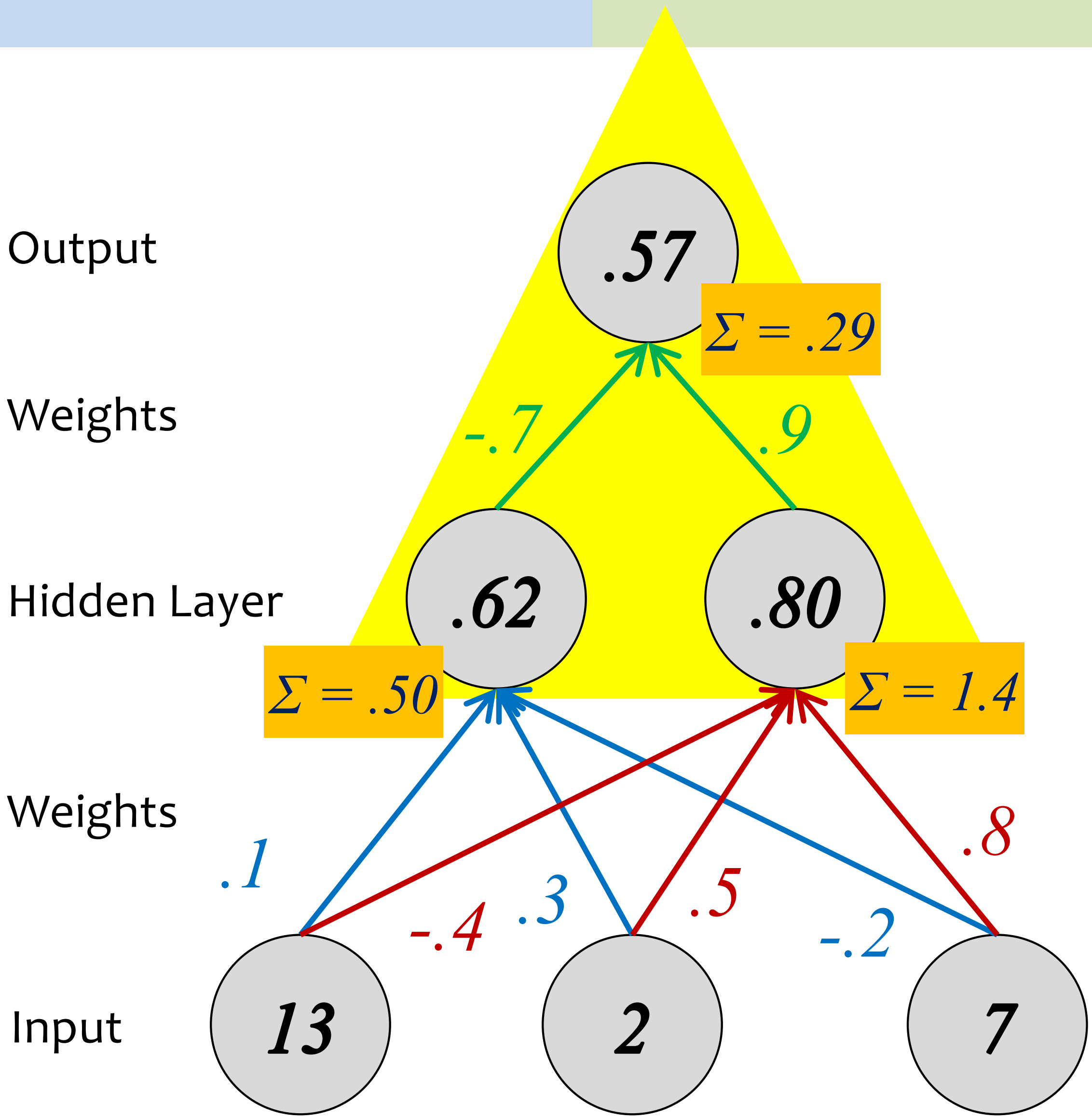
$.80 = \sigma(1.4)$

$1.4 = 13(-.4) + 2(.5) + 7(.8)$

The computation of each neural network unit resembles binary logistic regression.

Decision Functions

Neural Network



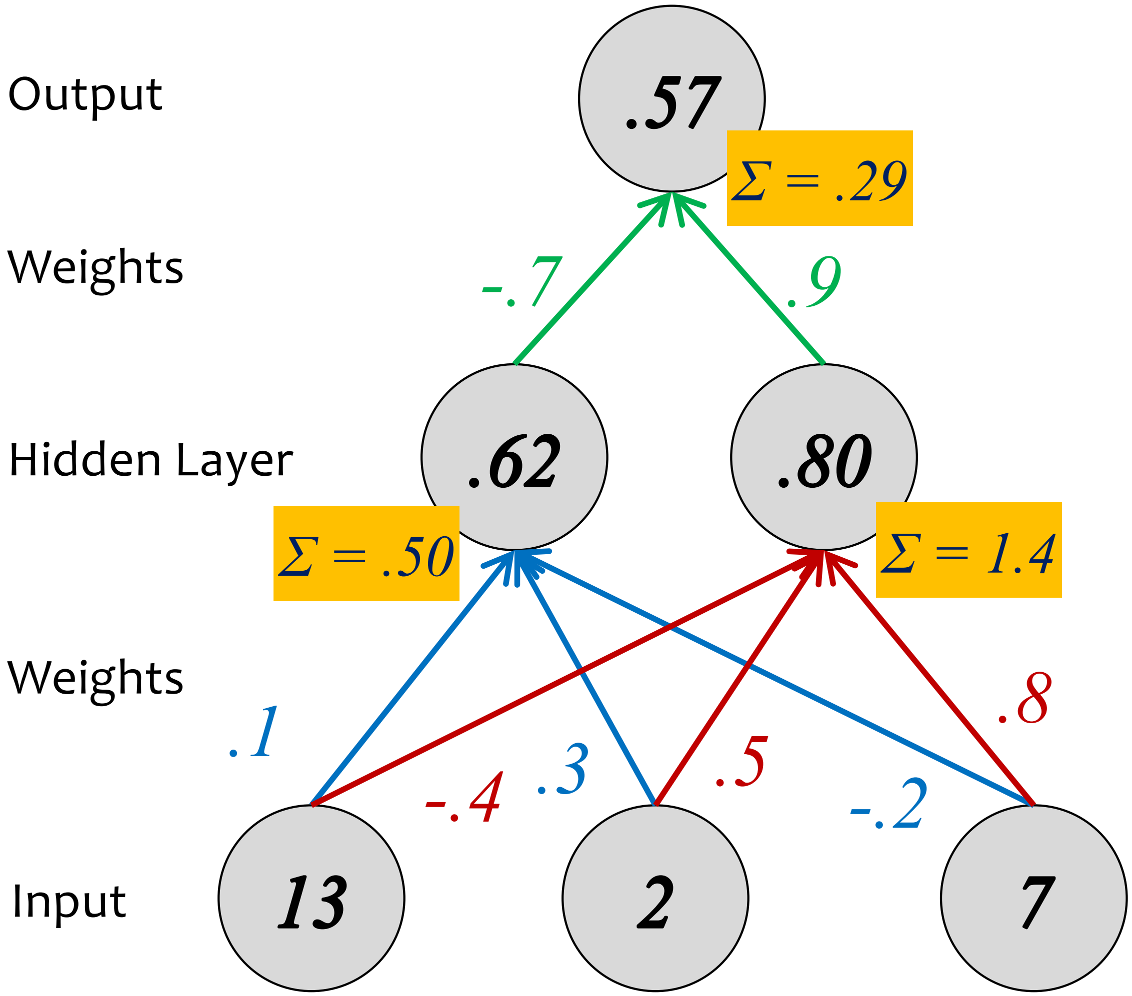
$.57 = \sigma(.29)$

$.29 = .62(-.7) + .80(.9)$

The computation of each neural network unit resembles binary logistic regression.

Decision Functions

Neural Network



$.57 = \sigma(.29)$

$.29 = .62(-.7) + .80(.9)$

$.80 = \sigma(1.4)$

$1.4 = 13(-.4) + 2(.5) + 7(.8)$

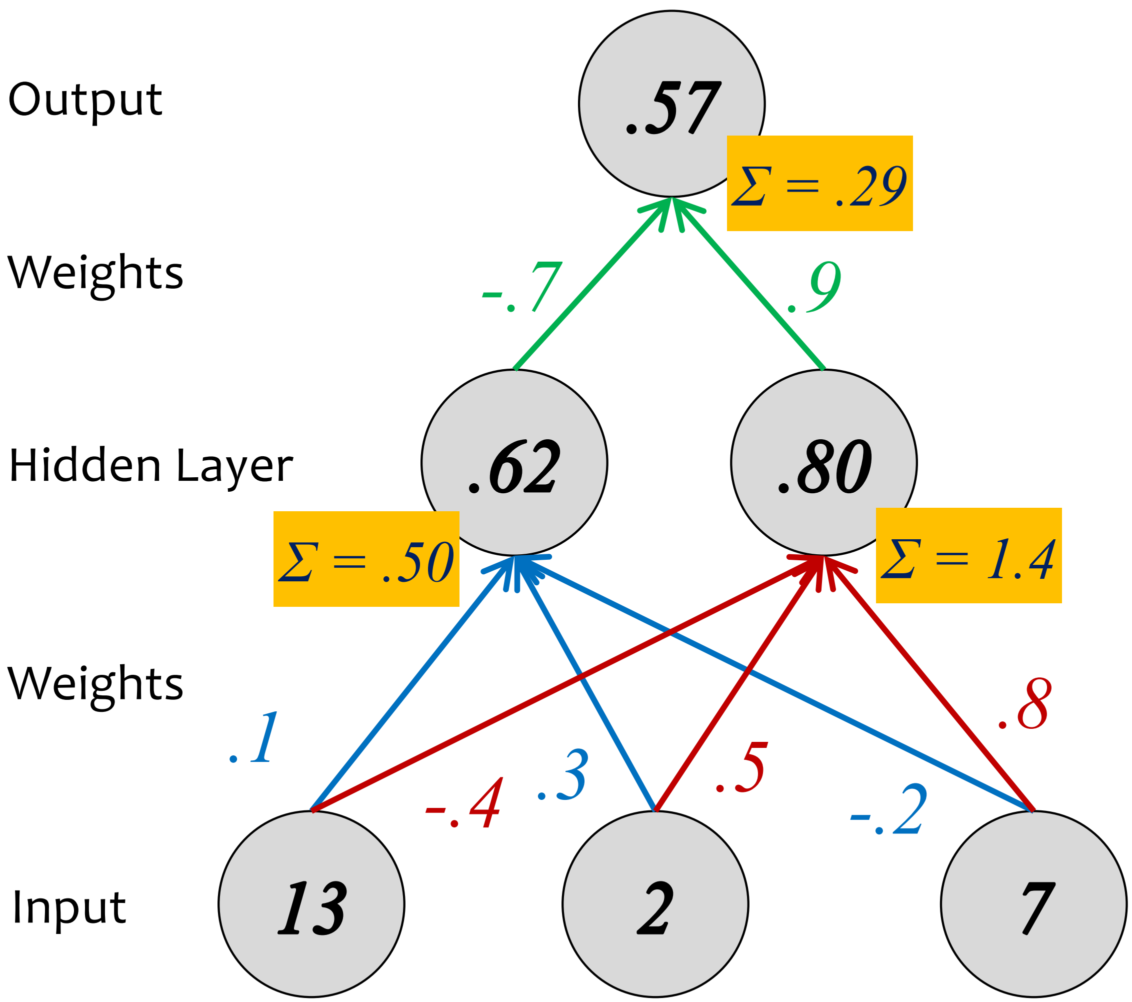
$.62 = \sigma(.50)$

$.50 = 13(.1) + 2(.3) + 7(-.2)$

The computation of each neural network unit resembles binary logistic regression.

Decision Functions

Neural Network



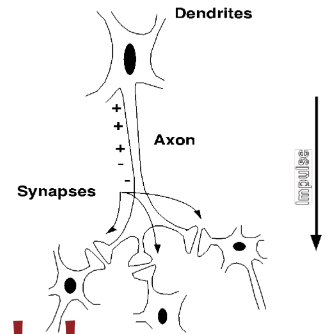
Except we only have the target value for y at training time!

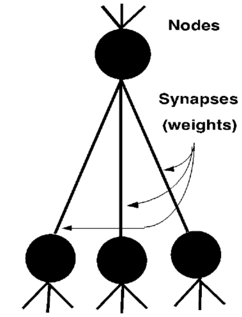
We have to learn to create "useful" values of z_1 and z_2 in the hidden layer.

The computation of each neural network unit resembles binary logistic regression.

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

DEFINING A 1-HIDDEN LAYER NEURAL NETWORK

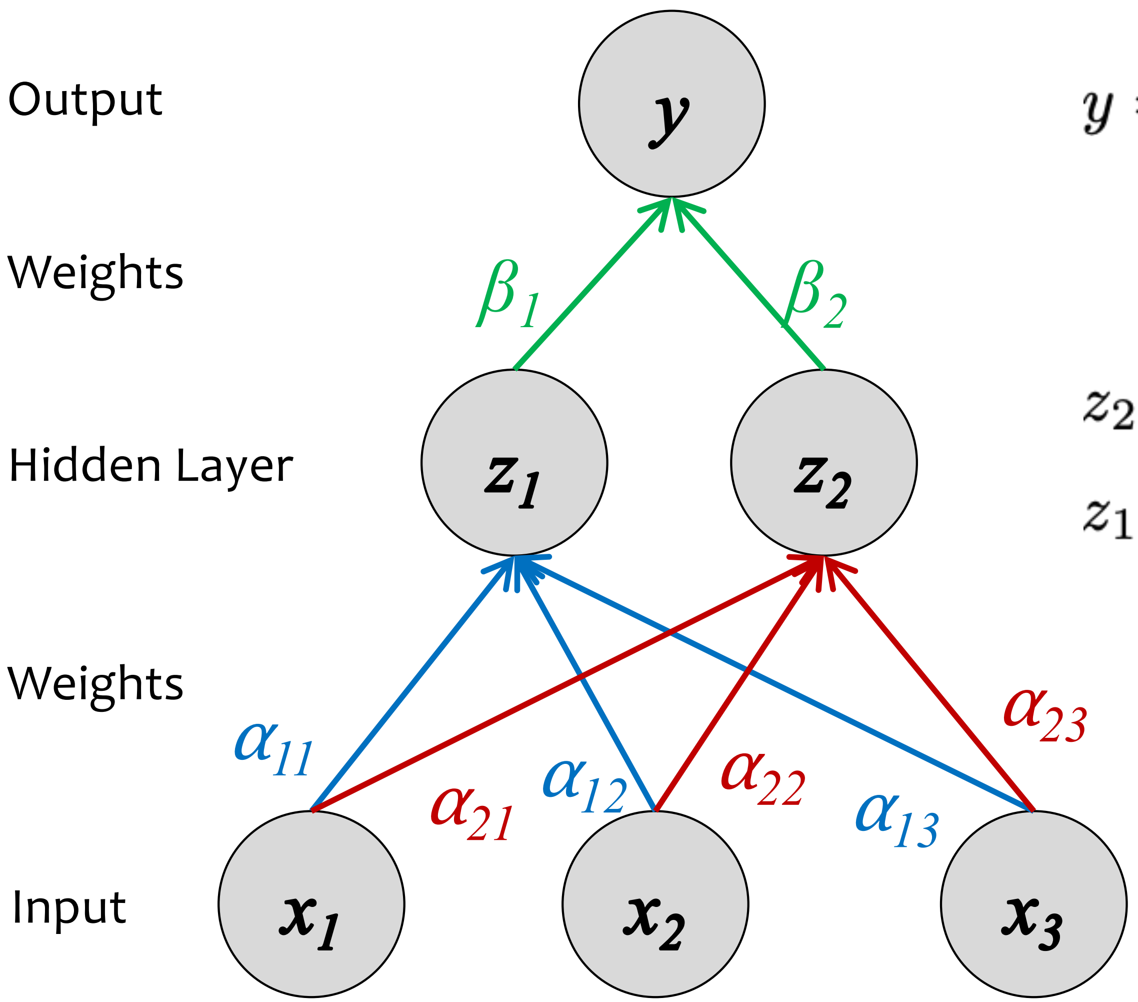
Neural Networks

Chalkboard

– Example: Neural Network w/1 Hidden Layer

Decision Functions

Neural Network



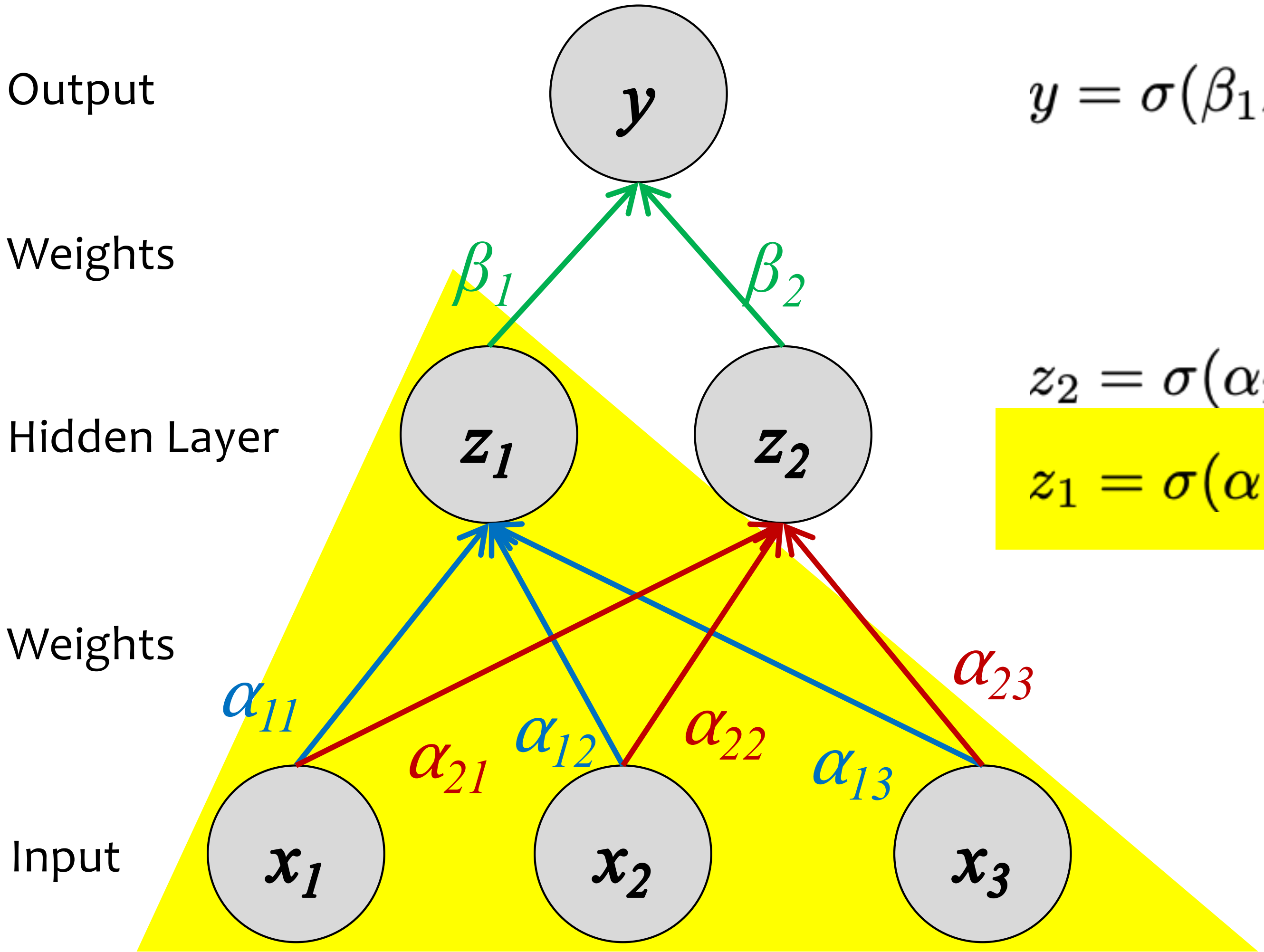
$$y = \sigma(\beta_1 z_1 + \beta_2 z_2)$$

$$z_2 = \sigma(\alpha_{21}x_1 + \alpha_{22}x_2 + \alpha_{23}x_3)$$

$$z_1 = \sigma(\alpha_{11}x_1 + \alpha_{12}x_2 + \alpha_{13}x_3)$$

Decision Functions

Neural Network



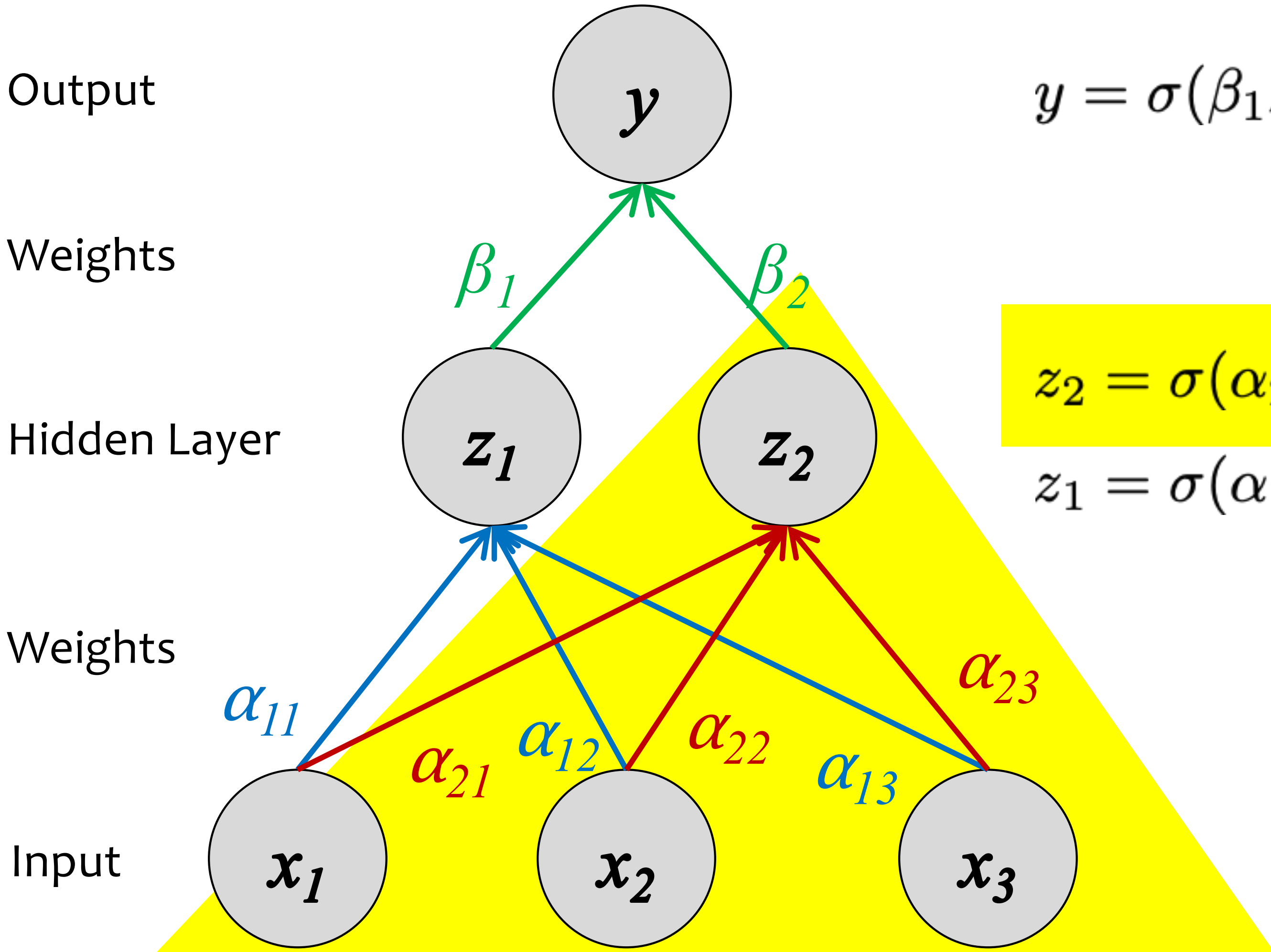
$$y = \sigma(\beta_1 z_1 + \beta_2 z_2)$$

$$z_2 = \sigma(\alpha_{21} x_1 + \alpha_{22} x_2 + \alpha_{23} x_3)$$

$$z_1 = \sigma(\alpha_{11} x_1 + \alpha_{12} x_2 + \alpha_{13} x_3)$$

Decision Functions

Neural Network



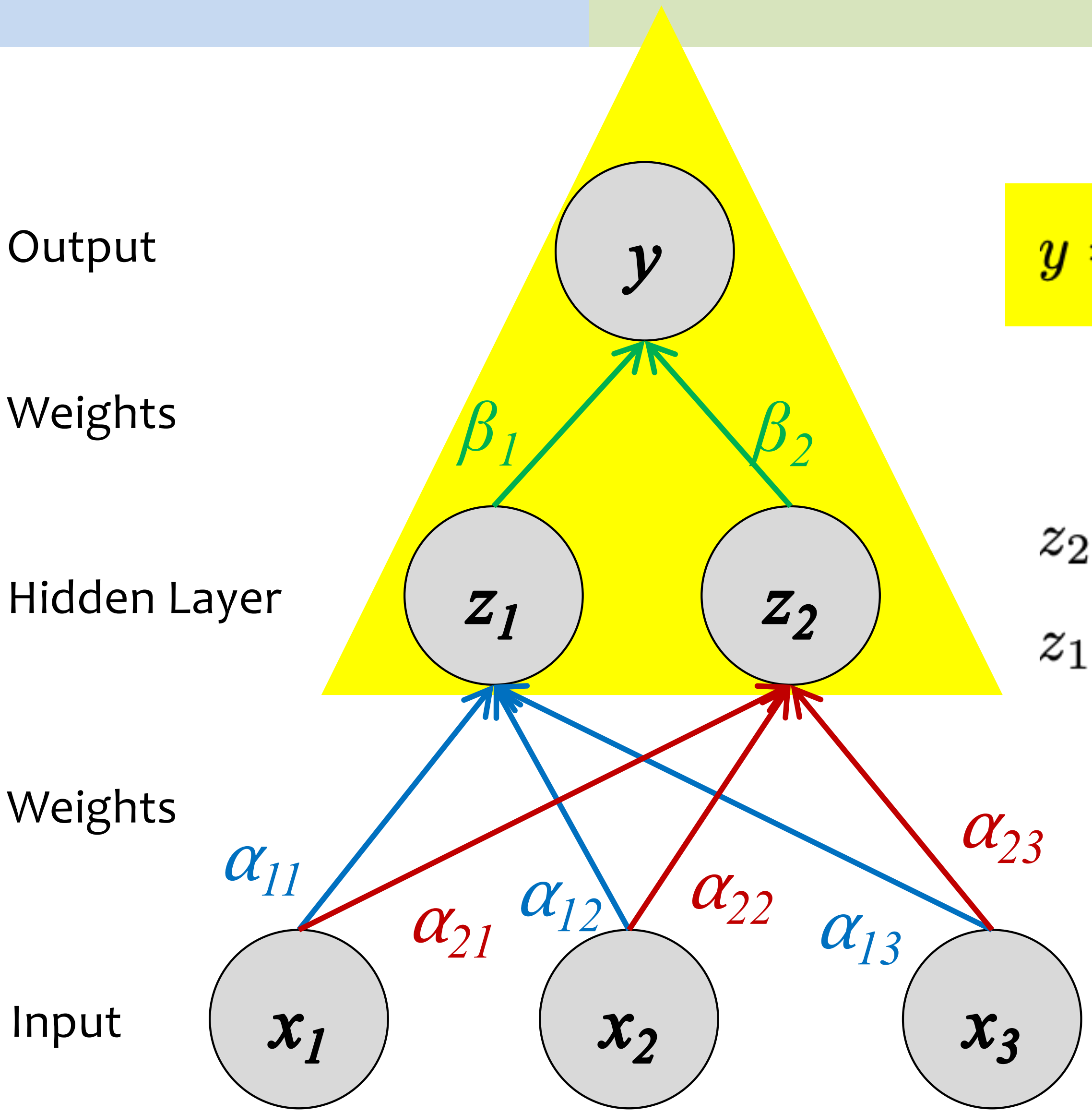
$$y = \sigma(\beta_1 z_1 + \beta_2 z_2)$$

$$z_2 = \sigma(\alpha_{21} x_1 + \alpha_{22} x_2 + \alpha_{23} x_3)$$

$$z_1 = \sigma(\alpha_{11} x_1 + \alpha_{12} x_2 + \alpha_{13} x_3)$$

Decision Functions

Neural Network



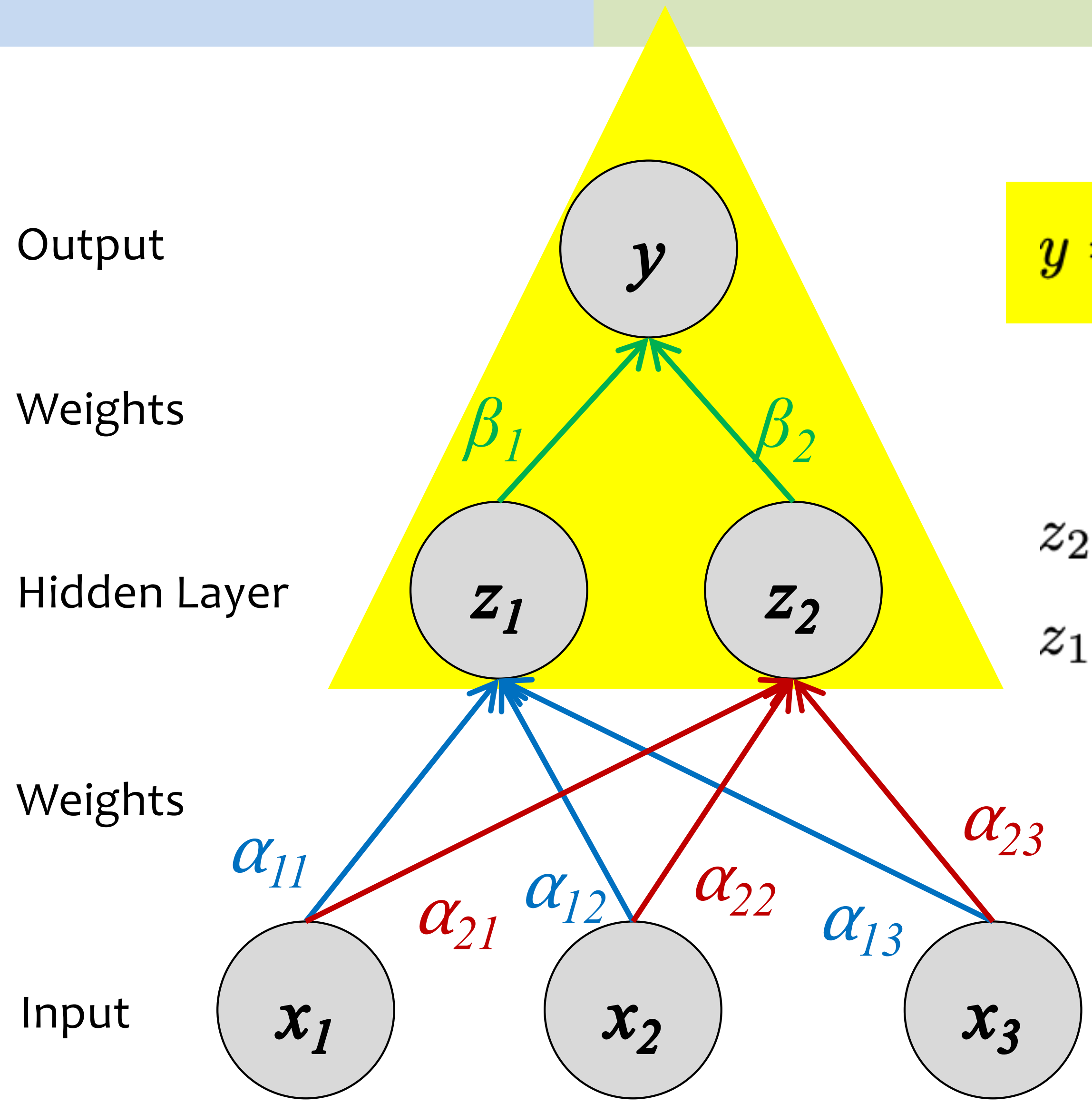
$$y = \sigma(\beta_1 z_1 + \beta_2 z_2)$$

$$z_2 = \sigma(\alpha_{21} x_1 + \alpha_{22} x_2 + \alpha_{23} x_3)$$

$$z_1 = \sigma(\alpha_{11} x_1 + \alpha_{12} x_2 + \alpha_{13} x_3)$$

Decision Functions

Neural Network



$$y = \sigma(\beta_1 z_1 + \beta_2 z_2)$$

$$z_2 = \sigma(\alpha_{21}x_1 + \alpha_{22}x_2 + \alpha_{23}x_3)$$

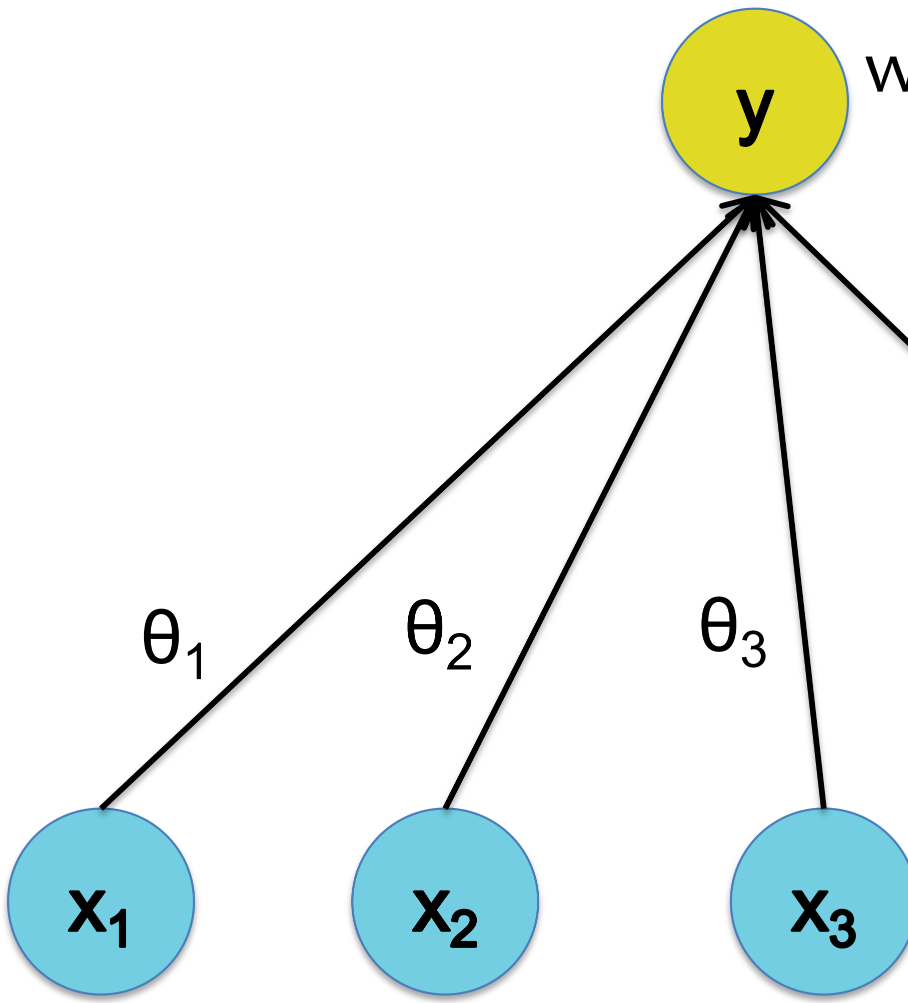
$$z_1 = \sigma(\alpha_{11}x_1 + \alpha_{12}x_2 + \alpha_{13}x_3)$$

NONLINEAR DECISION BOUNDARIES AND NEURAL NETWORKS

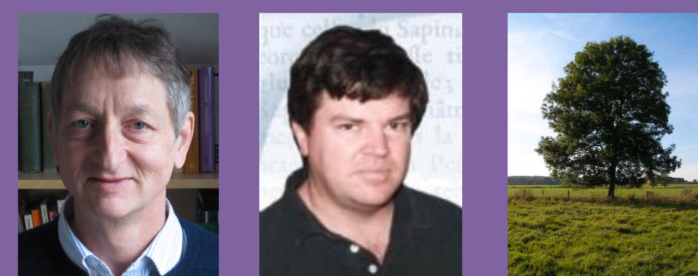
Logistic Regression

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

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



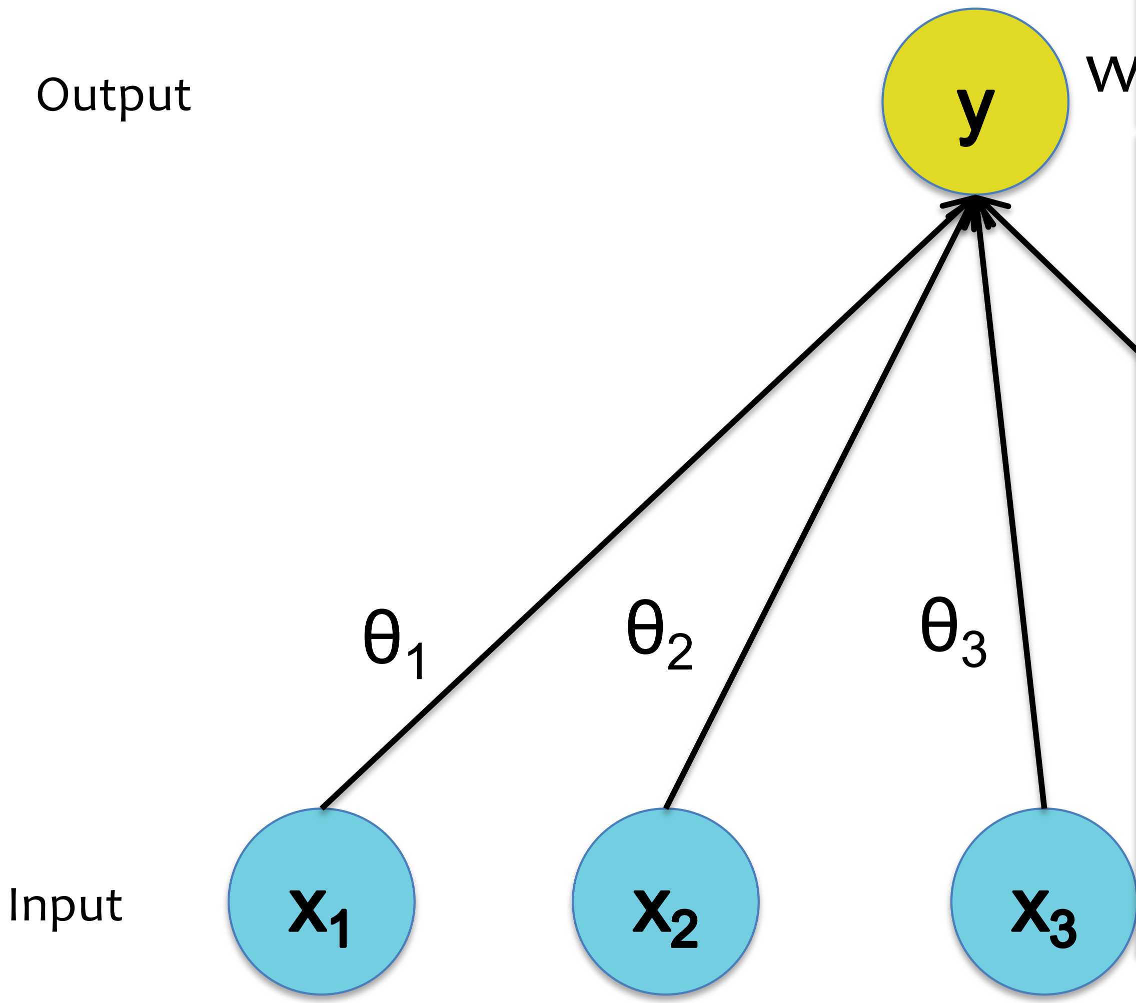
Face *Face* *Not a face*



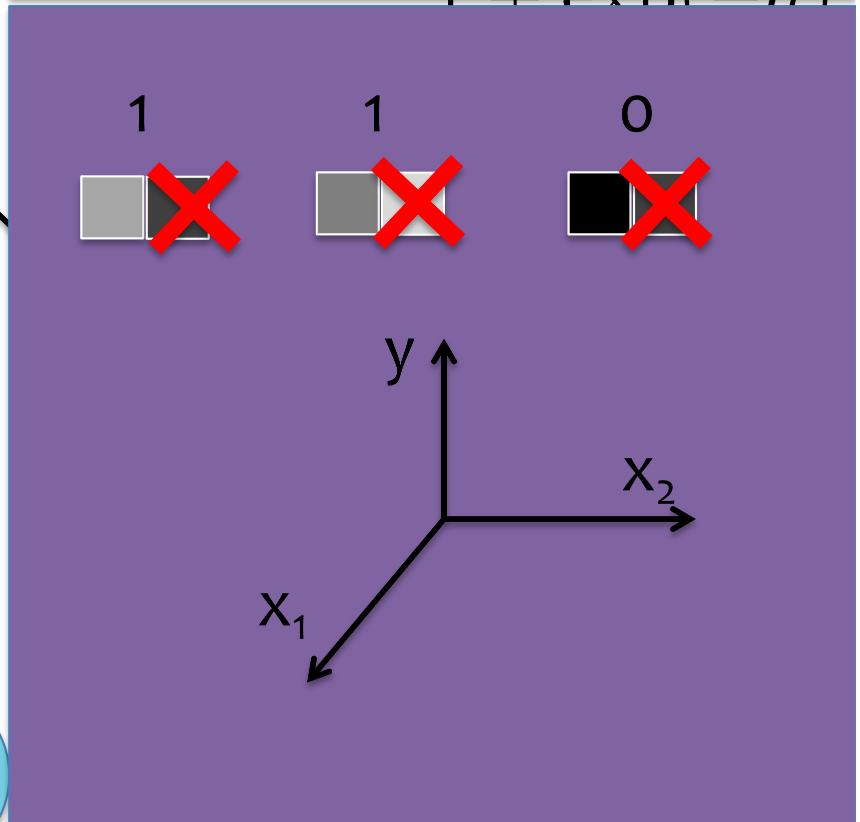
Decision Functions

Logistic Regression

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



In-Class Example



Neural Networks

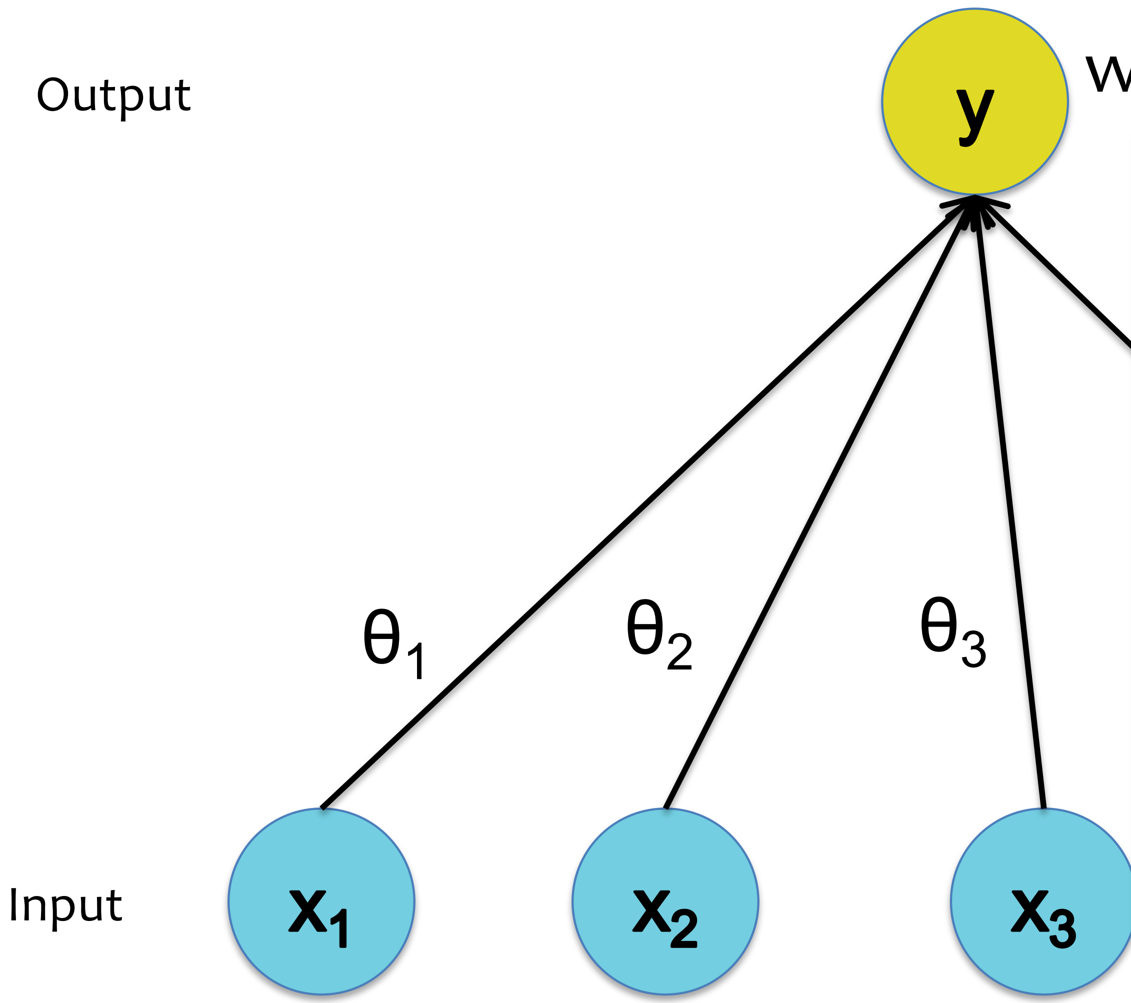
Chalkboard

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

Logistic Regression

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

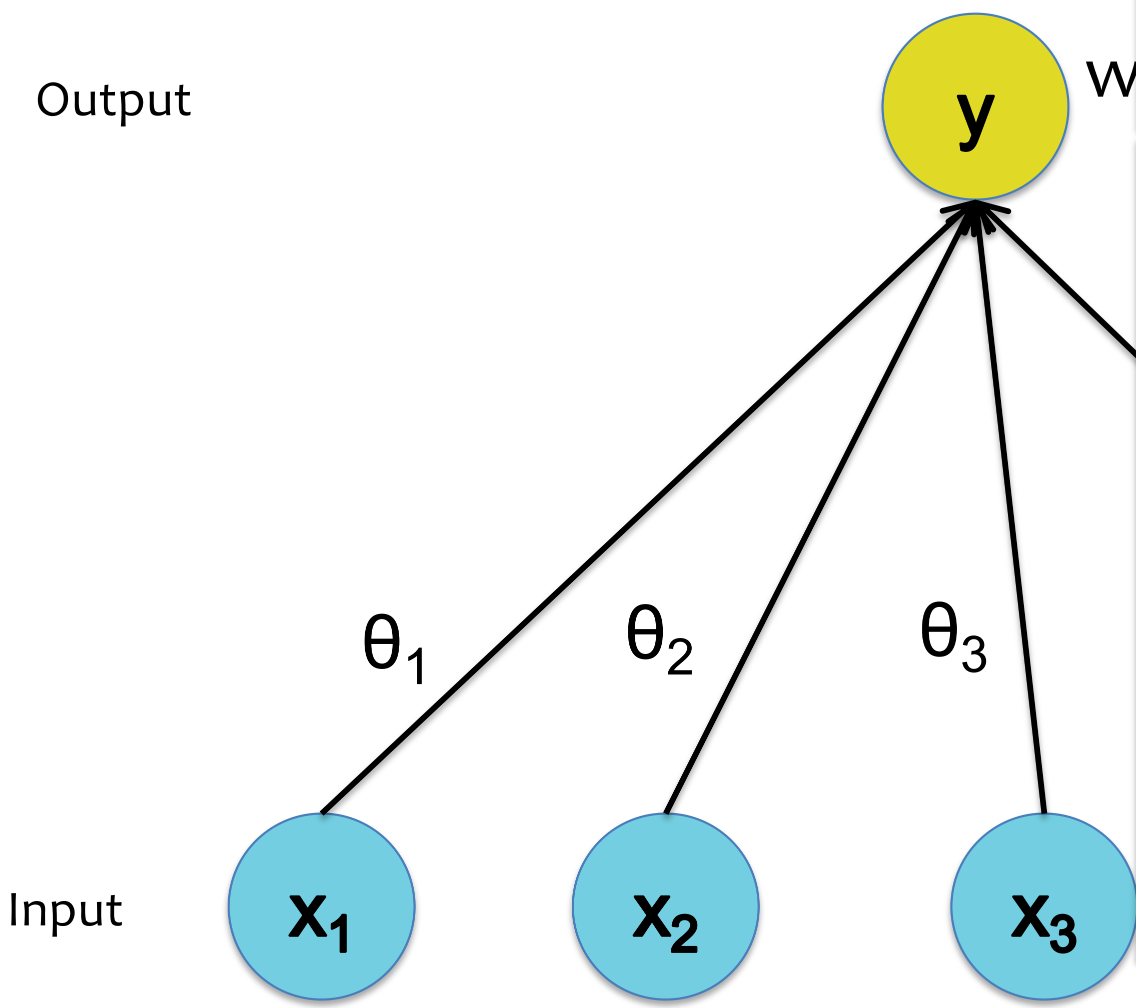
where $\sigma(a) = \frac{1}{1 + \exp(-a)}$



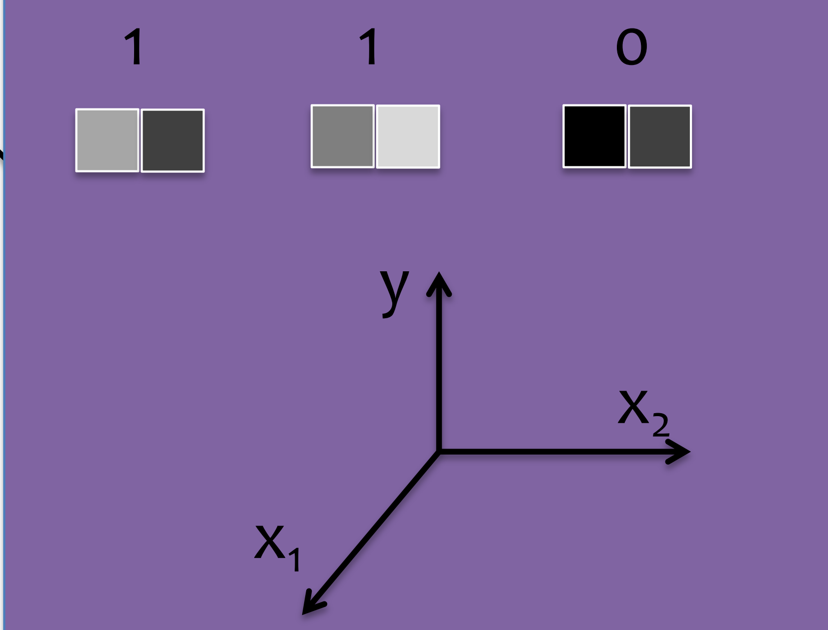
Decision Functions

Logistic Regression

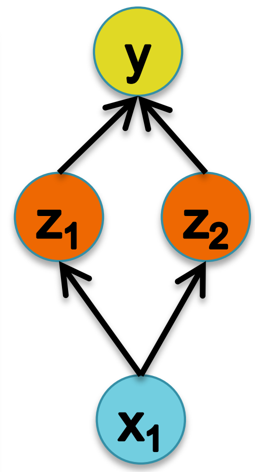
$$y = h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(\boldsymbol{\theta}^T \boldsymbol{x})$$



In-Class Example



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: