



# 10-601 Introduction to Machine Learning

Machine Learning Department  
School of Computer Science  
Carnegie Mellon University

# Multinomial Logistic Regression + Feature Engineering

Matt Gormley  
Lecture 11  
Sep. 30, 2019

# Reminders

- **Midterm Exam 1**
  - Thu, Oct. 03, 6:30pm – 8:00pm
- **Homework 4: Logistic Regression**
  - Out: Wed, Sep. 25
  - Due: Fri, Oct. 11 at 11:59pm
- **Today's In-Class Poll**
  - <http://p11.mlcourse.org>
- **HW3 grades published**
- **Crowdsourcing Exam Questions**

# **MULTINOMIAL LOGISTIC REGRESSION**



# Multinomial Logistic Regression

## *Chalkboard*

- Background: Multinomial distribution
- Definition: Multi-class classification
- Geometric intuitions
- Multinomial logistic regression model
- Generative story
- Reduction to binary logistic regression
- Partial derivatives and gradients
- Applying Gradient Descent and SGD
- Implementation w/ sparse features

# Debug that Program!

## In-Class Exercise: *Think-Pair-Share*

Debug the following program which is (incorrectly) attempting to run SGD for multinomial logistic regression

### Buggy Program:

```
while not converged:
    for i in shuffle([1,...,N]):
        for k in [1,...,K]:
            theta[k] = theta[k] - lambda * grad(x[i], y[i],
theta, k)
```

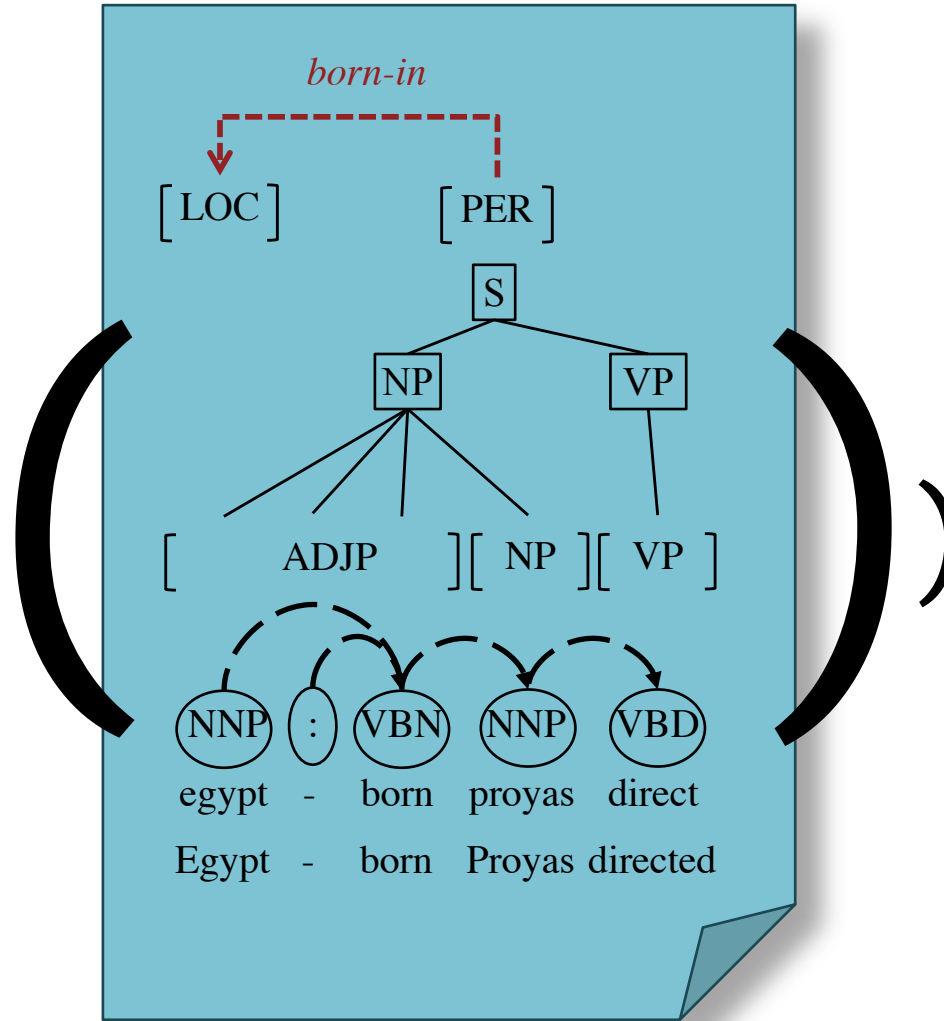
**Assume:** `grad(x[i], y[i], theta, k)` returns the gradient of the negative log-likelihood of the training example  $(x[i], y[i])$  with respect to vector `theta[k]`. `lambda` is the learning rate. `N` = # of examples. `K` = # of output classes. `M` = # of features. `theta` is a `K` by `M` matrix.

# **FEATURE ENGINEERING**

# Handcrafted Features

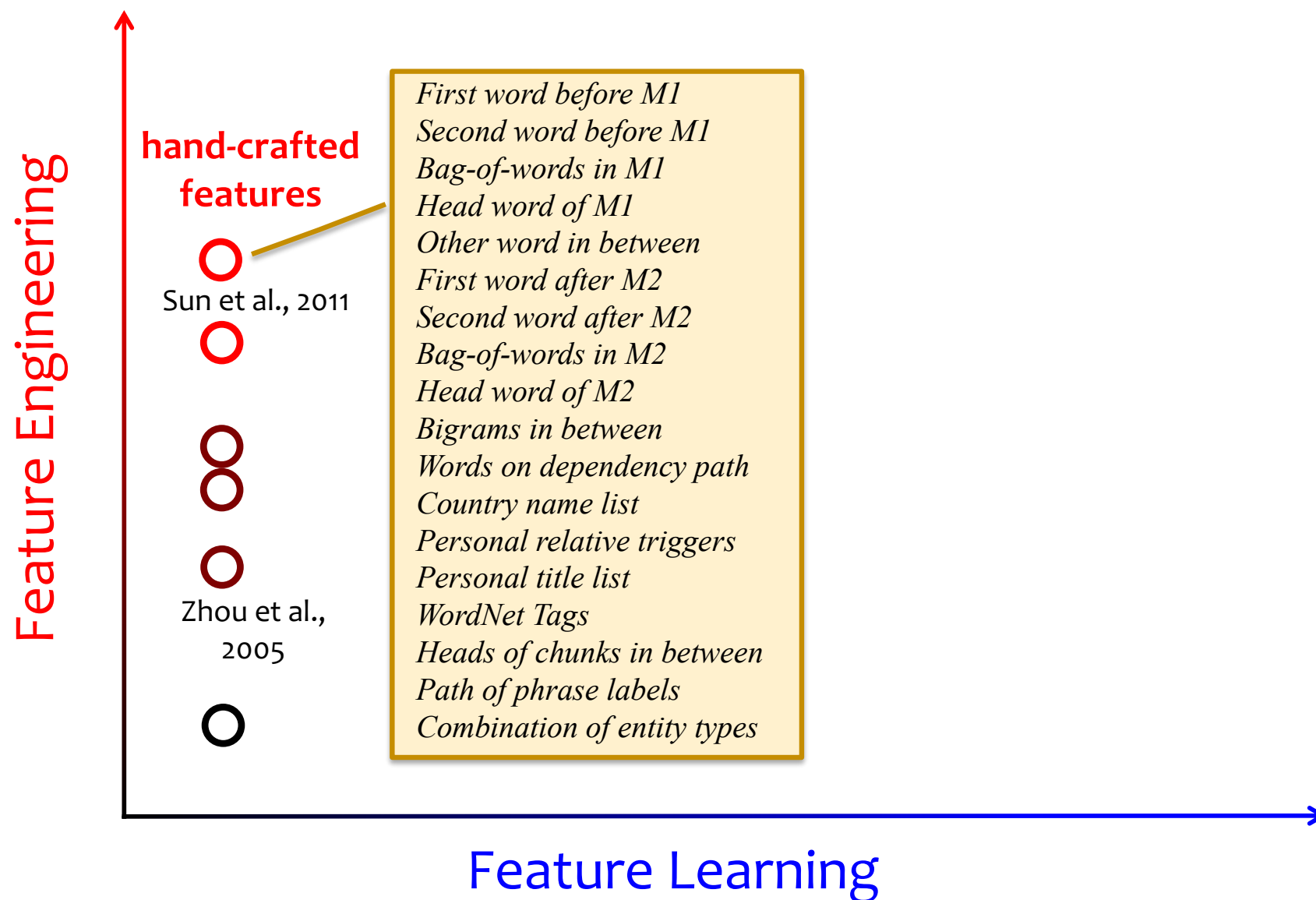
$$p(y|x) \propto$$

$$\exp(\Theta_y \cdot f$$

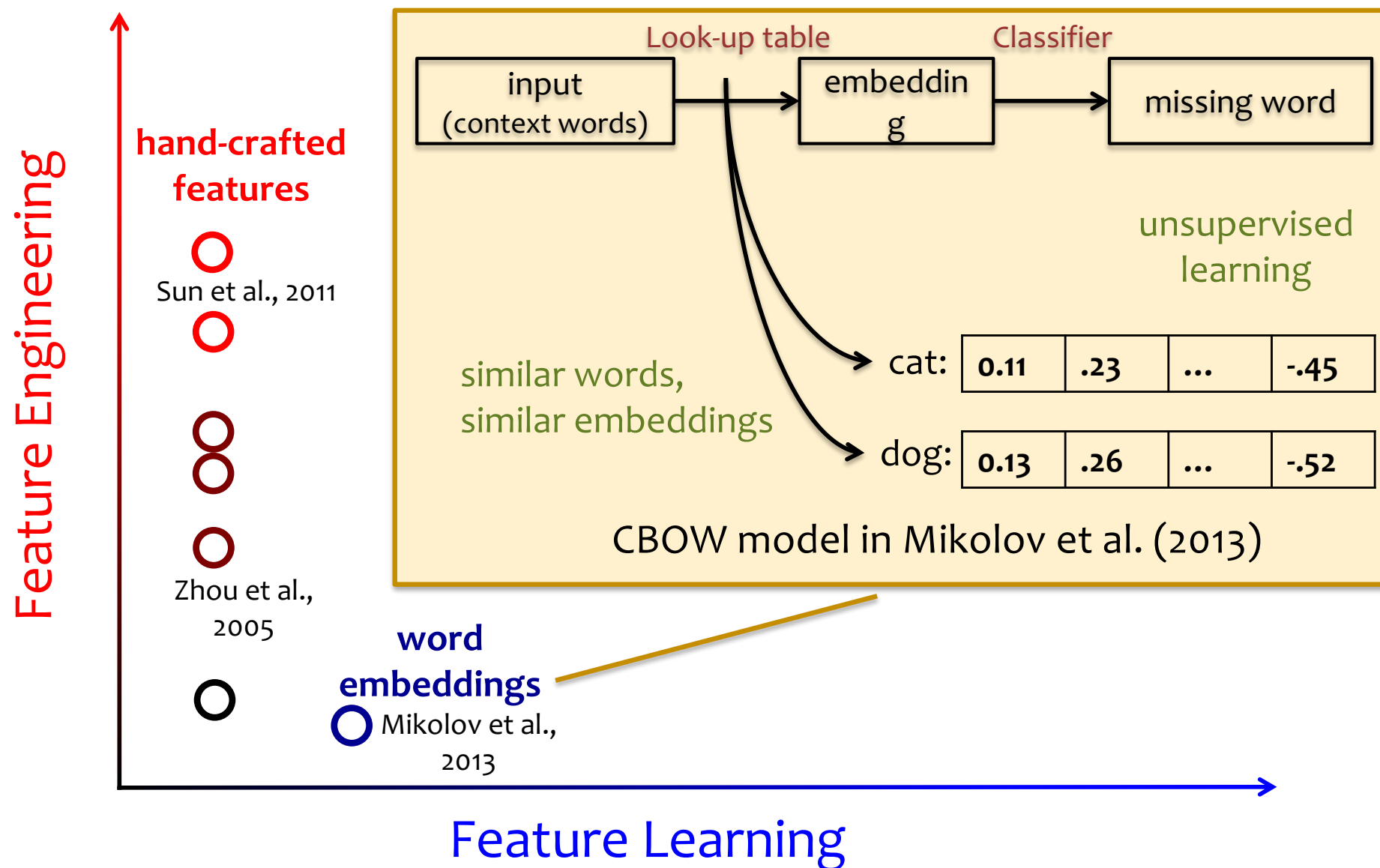




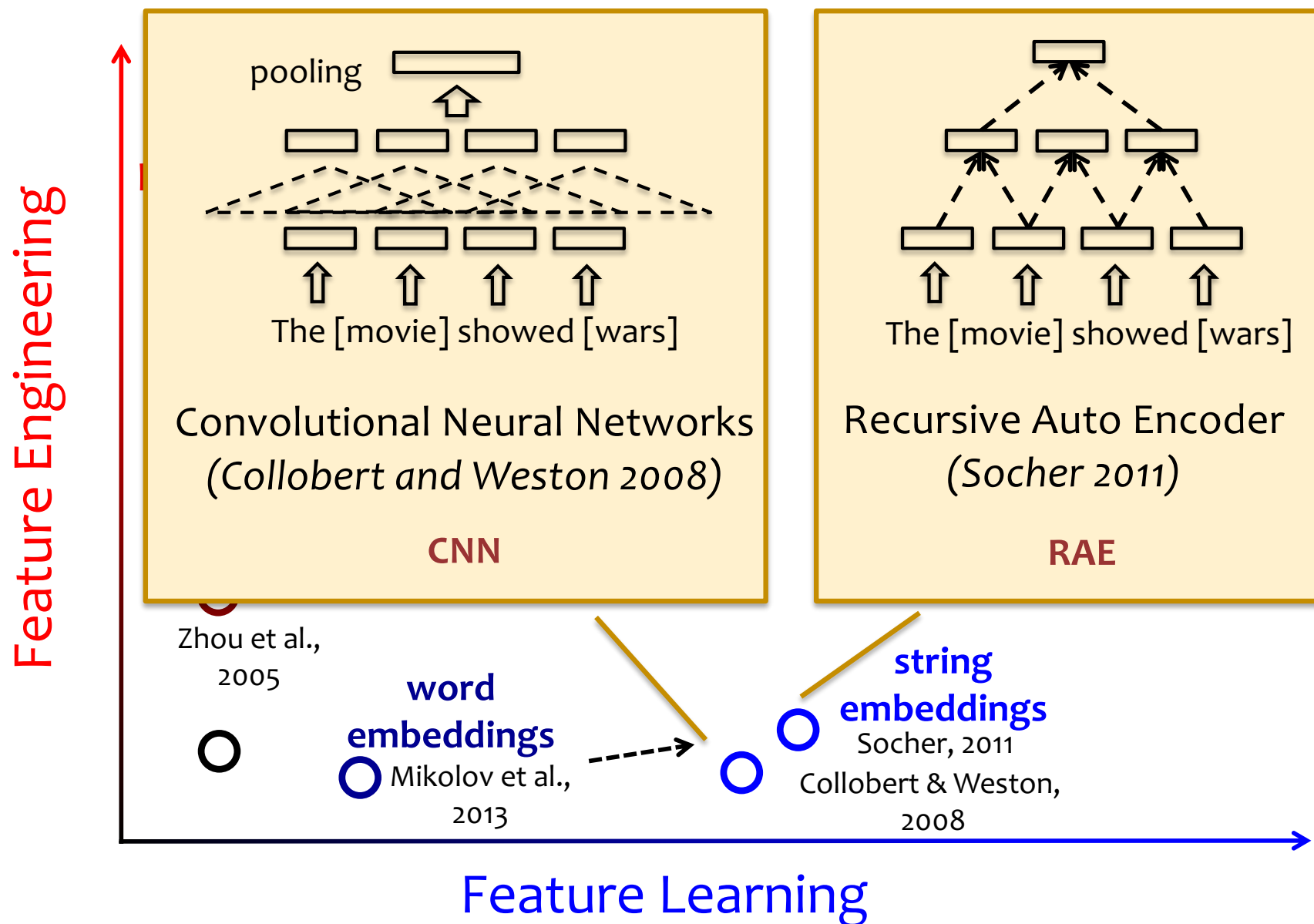
# Where do features come from?



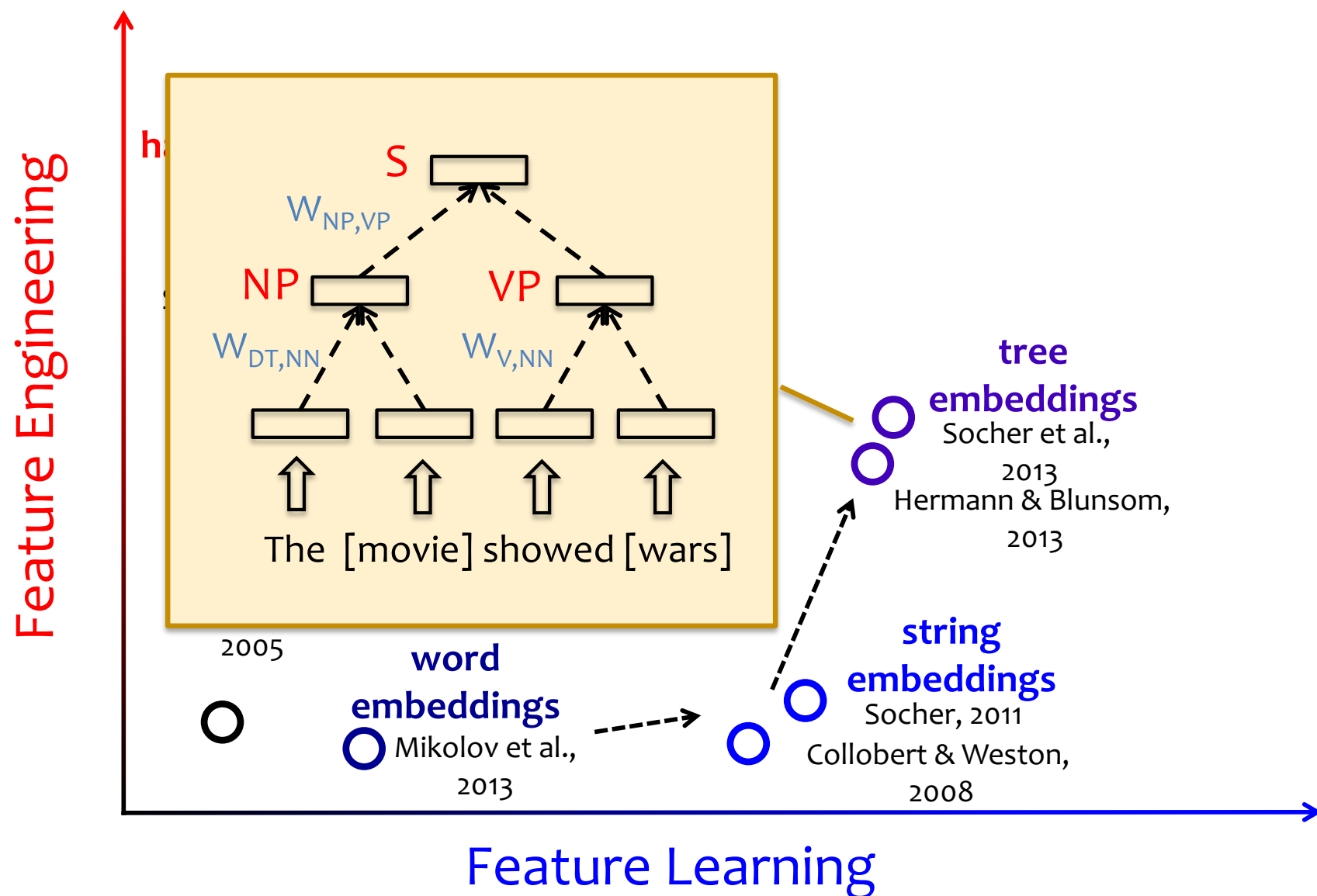
# Where do features come from?



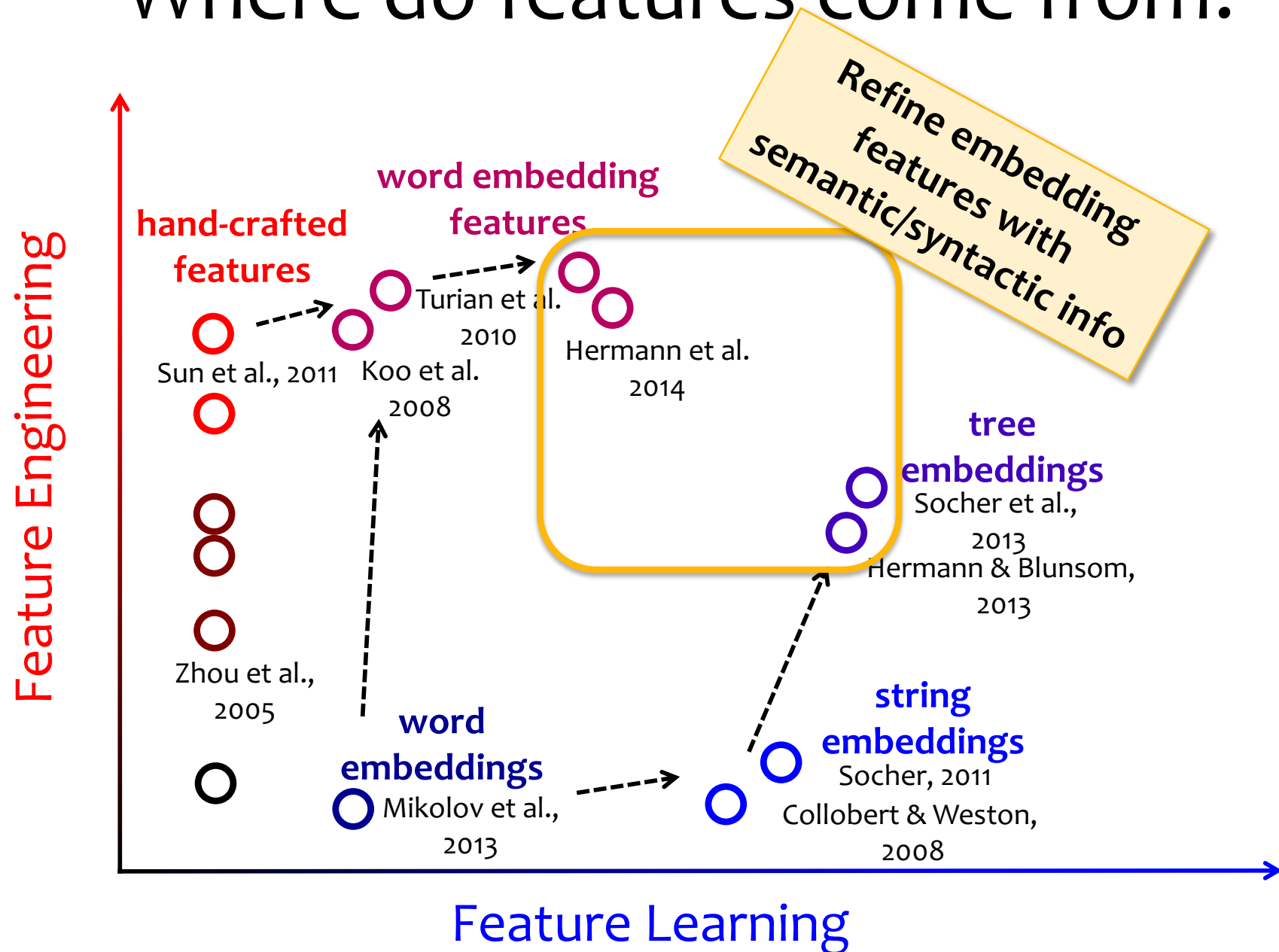
# Where do features come from?



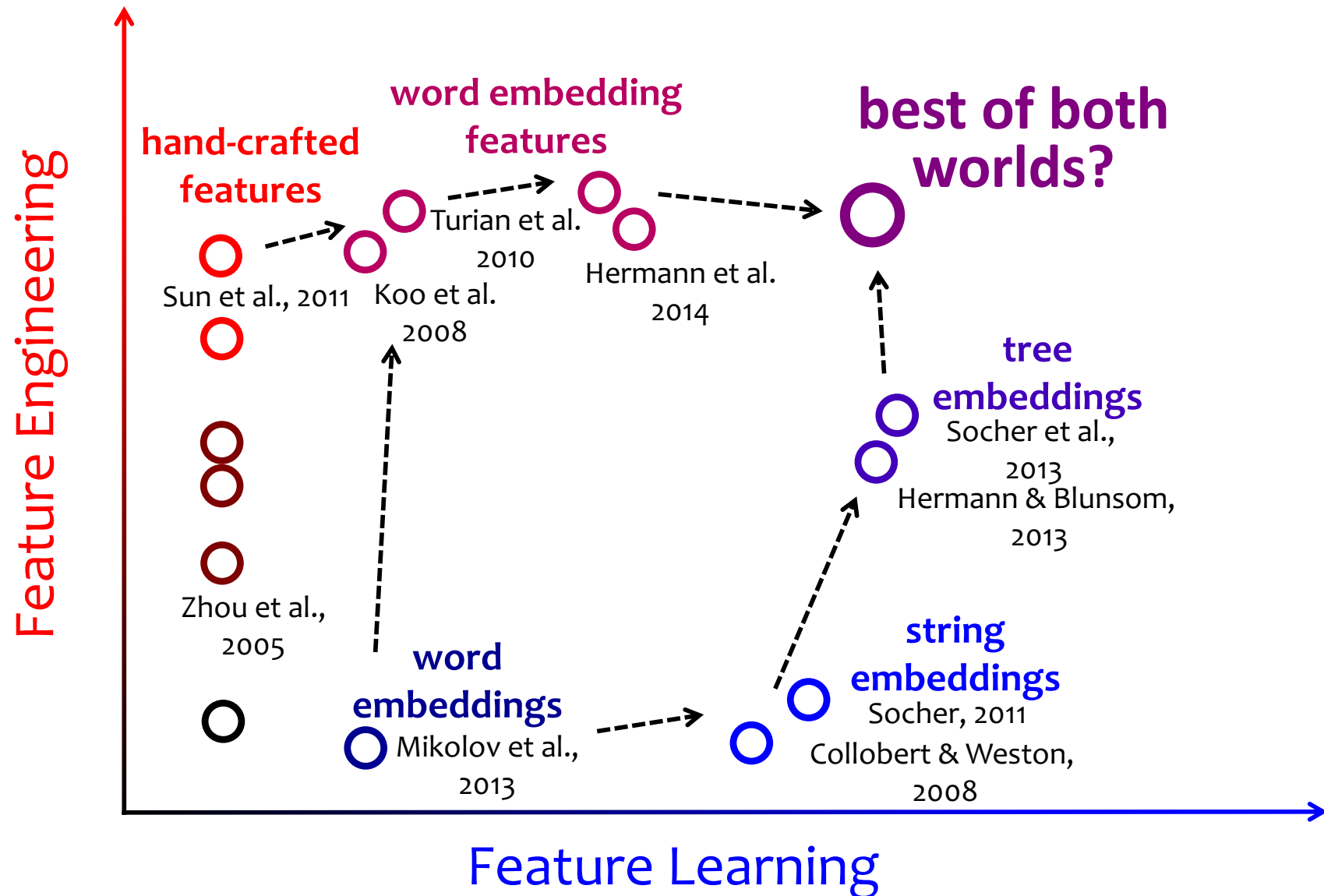
# Where do features come from?



# Where do features come from?



# Where do features come from?



# Feature Engineering for NLP

Suppose you build a logistic regression model to predict a part-of-speech (POS) tag for each word in a sentence.

**What features should you use?**

deter.

noun

noun

verb

verb

noun

*The movie I watched depicted hope*

# Feature Engineering for NLP

## Per-word Features:

	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$
<code>is-capital(<math>w_i</math>)</code>	1	0	1	0	0	0
<code>endswith(<math>w_i</math>, "e")</code>	1	1	0	0	0	1
<code>endswith(<math>w_i</math>, "d")</code>	0	0	0	1	1	0
<code>endswith(<math>w_i</math>, "ed")</code>	0	0	0	1	1	0
<code><math>w_i == \text{"aardvark"}</math></code>	0	0	0	0	0	0
<code><math>w_i == \text{"hope"}</math></code>	0	0	0	0	0	1
...	...	...	...	...	...	...

deter.    noun    noun    verb    verb    noun

*The    movie    I    watched    depicted    hope*



# Feature Engineering for NLP

## Context Features:

	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$
...	...	...	...	...	...	...
$w_i == \text{"watched"}$	0	0	0	1	0	0
$w_{i+1} == \text{"watched"}$	0	0	1	0	0	0
$w_{i-1} == \text{"watched"}$	0	0	0	0	1	0
$w_{i+2} == \text{"watched"}$	0	1	0	0	0	0
$w_{i-2} == \text{"watched"}$	0	0	0	0	0	1
...	...	...	...	...	...	...

deter.    noun    noun    verb    verb    noun

*The    movie    I    watched    depicted    hope*

# Feature Engineering for NLP

## Context Features:

	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$
...	...	...	...	...	...	...
$w_i == \text{"I"}$	0	0	1	0	0	0
$w_{i+1} == \text{"I"}$	0	1	0	0	0	0
$w_{i-1} == \text{"I"}$	0	0	0	1	0	0
$w_{i+2} == \text{"I"}$	1	0	0	0	0	0
$w_{i-2} == \text{"I"}$	0	0	0	0	1	0
...	...	...	...	...	...	...

deter.	noun	noun	verb	verb	noun
--------	------	------	------	------	------

*The movie I watched depicted hope*

# Feature Engineering for NLP

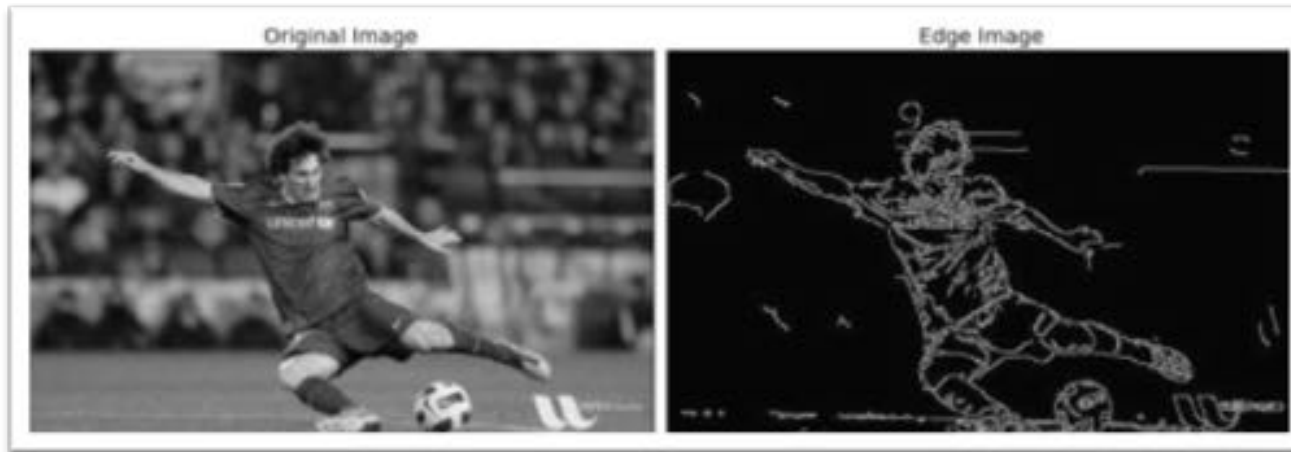
**Table 3.** Tagging accuracies with different feature templates and other changes on the *WSJ* 19-21 development set.

Model	Feature Templates	# Feats	Sent. Acc.	Token Acc.	Unk. Acc.
3GRAMMEMM	See text	248,798	52.07%	96.92%	88.99%
NAACL 2003	See text and [1]	460,552	55.31%	97.15%	88.61%
Replication	See text and [1]	460,551	55.62%	97.18%	88.92%
Replication'	+rareFeatureThresh = 5	482,364	55.67%	97.19%	88.96%
5W	+ $\langle t_0, w_{-2} \rangle, \langle t_0, w_2 \rangle$	730,178	56.23%	97.20%	89.03%
5WSHAPES	+ $\langle t_0, s_{-1} \rangle, \langle t_0, s_0 \rangle, \langle t_0, s_{+1} \rangle$	731,661	56.52%	97.25%	89.81%
5WSHAPESDS	+ distributional similarity	737,955	56.79%	97.28%	90.46%

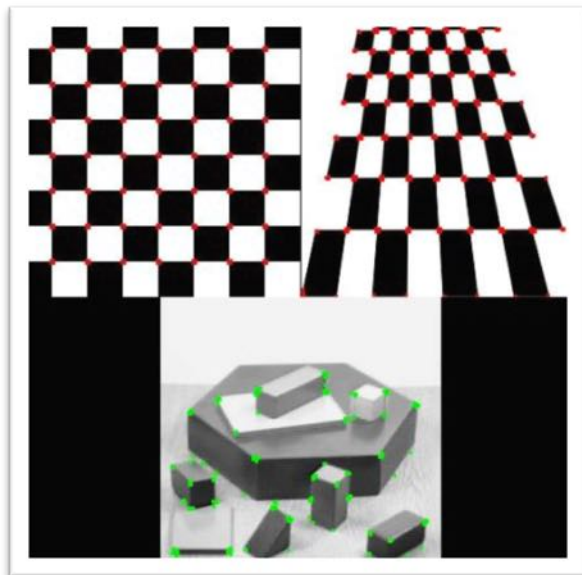
deter.	noun	noun	verb	verb	noun
<i>The</i>	<i>movie</i>	<i>I</i>	<i>watched</i>	<i>depicted</i>	<i>hope</i>

# Feature Engineering for CV

## Edge detection (Canny)

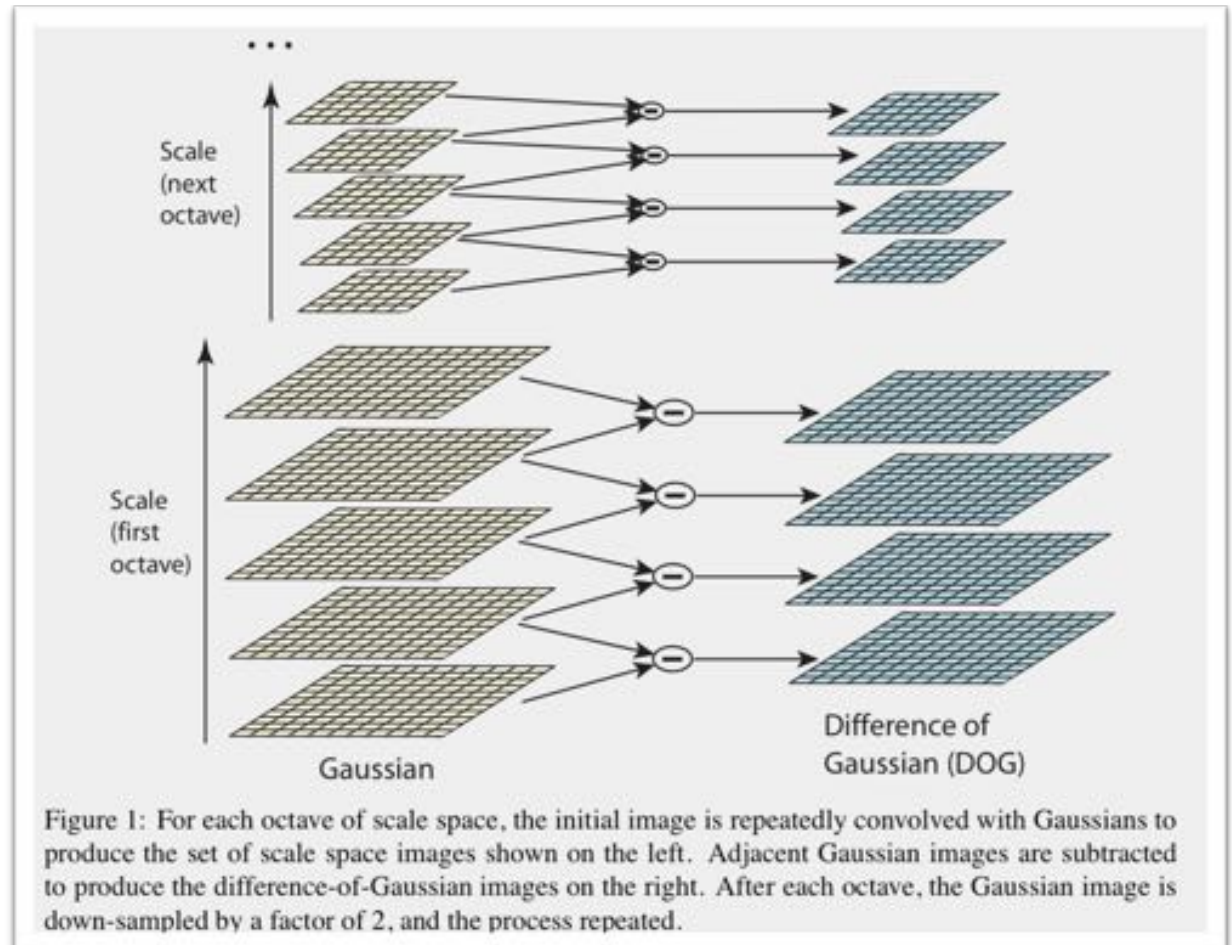


## Corner Detection (Harris)



# Feature Engineering for CV

## Scale Invariant Feature Transform (SIFT)



# **NON-LINEAR FEATURES**

# Nonlinear Features

- aka. “nonlinear basis functions”
- So far, input was always  $\mathbf{x} = [x_1, \dots, x_M]$
- **Key Idea:** let input be some function of  $\mathbf{x}$ 
  - original input:  $\mathbf{x} \in \mathbb{R}^M$
  - new input:  $\mathbf{x}' \in \mathbb{R}^{M'}$  where  $M' > M$  (usually)
  - define  $\mathbf{x}' = b(\mathbf{x}) = [b_1(\mathbf{x}), b_2(\mathbf{x}), \dots, b_{M'}(\mathbf{x})]$   
where  $b_i : \mathbb{R}^M \rightarrow \mathbb{R}$  is any function

- **Examples:** ( $M = 1$ )

polynomial

$$b_j(x) = x^j \quad \forall j \in \{1, \dots, J\}$$

radial basis function

$$b_j(x) = \exp\left(\frac{-(x - \mu_j)^2}{2\sigma_j^2}\right)$$

sigmoid

$$b_j(x) = \frac{1}{1 + \exp(-\omega_j x)}$$

log

$$b_j(x) = \log(x)$$

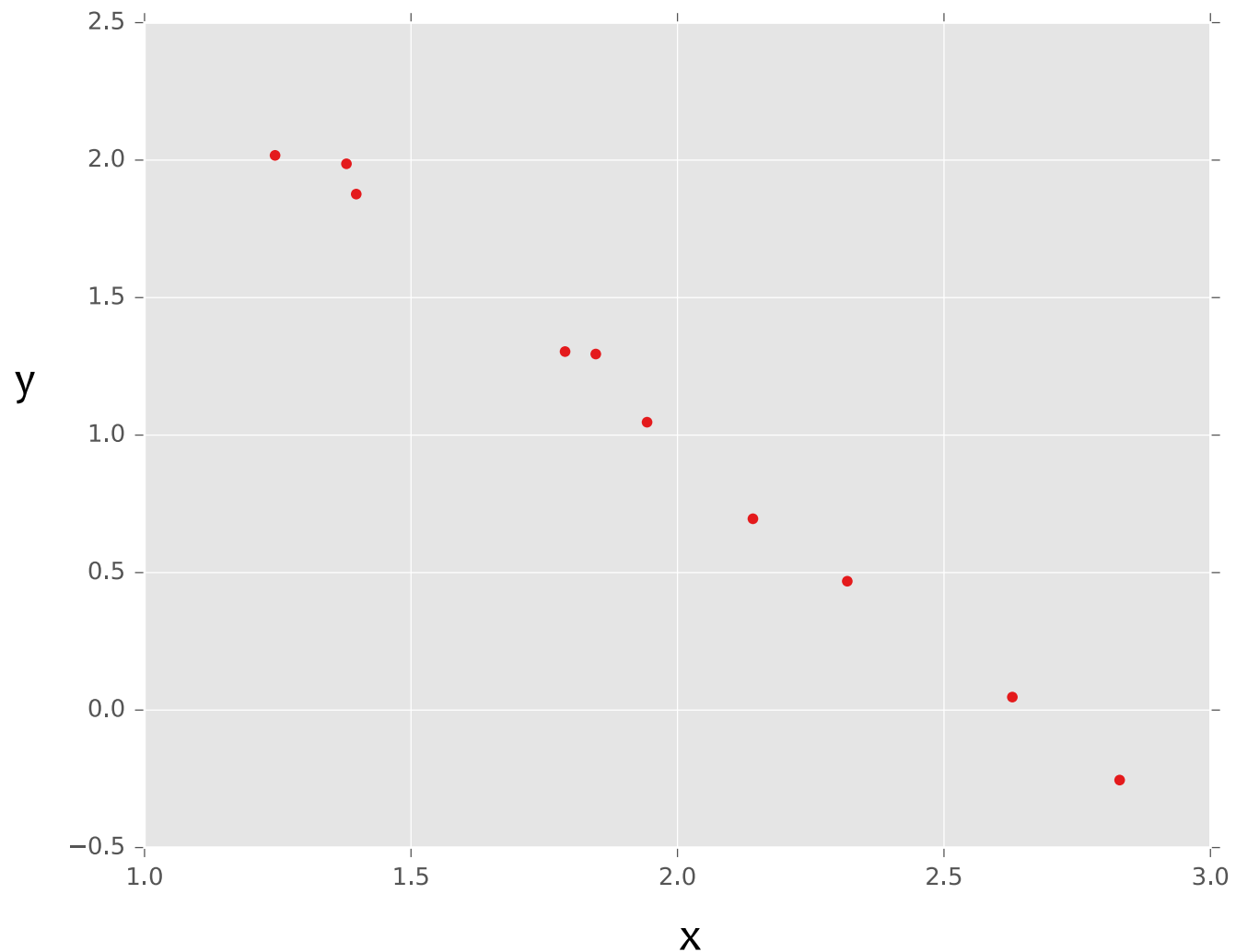
**For a linear model:**  
still a linear function  
of  $b(\mathbf{x})$  even though a  
nonlinear function of  
 $\mathbf{x}$

**Examples:**

- Perceptron
- Linear regression
- Logistic regression

# Example: Linear Regression

**Goal:** Learn  $y = \mathbf{w}^T \mathbf{f}(\mathbf{x}) + b$   
where  $\mathbf{f}(\cdot)$  is a polynomial  
basis function

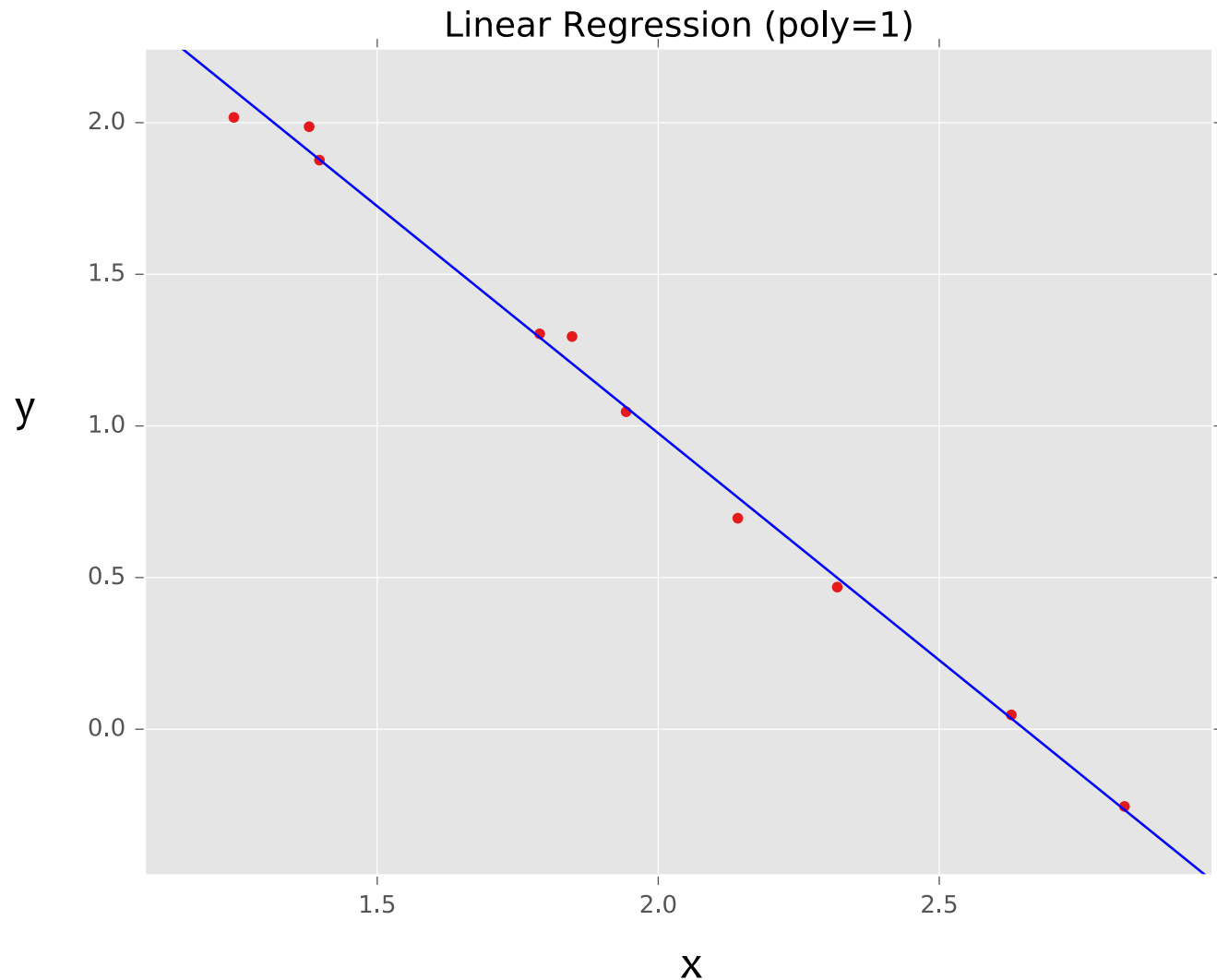


true “unknown”  
target function is  
linear with  
negative slope  
and gaussian  
noise



# Example: Linear Regression

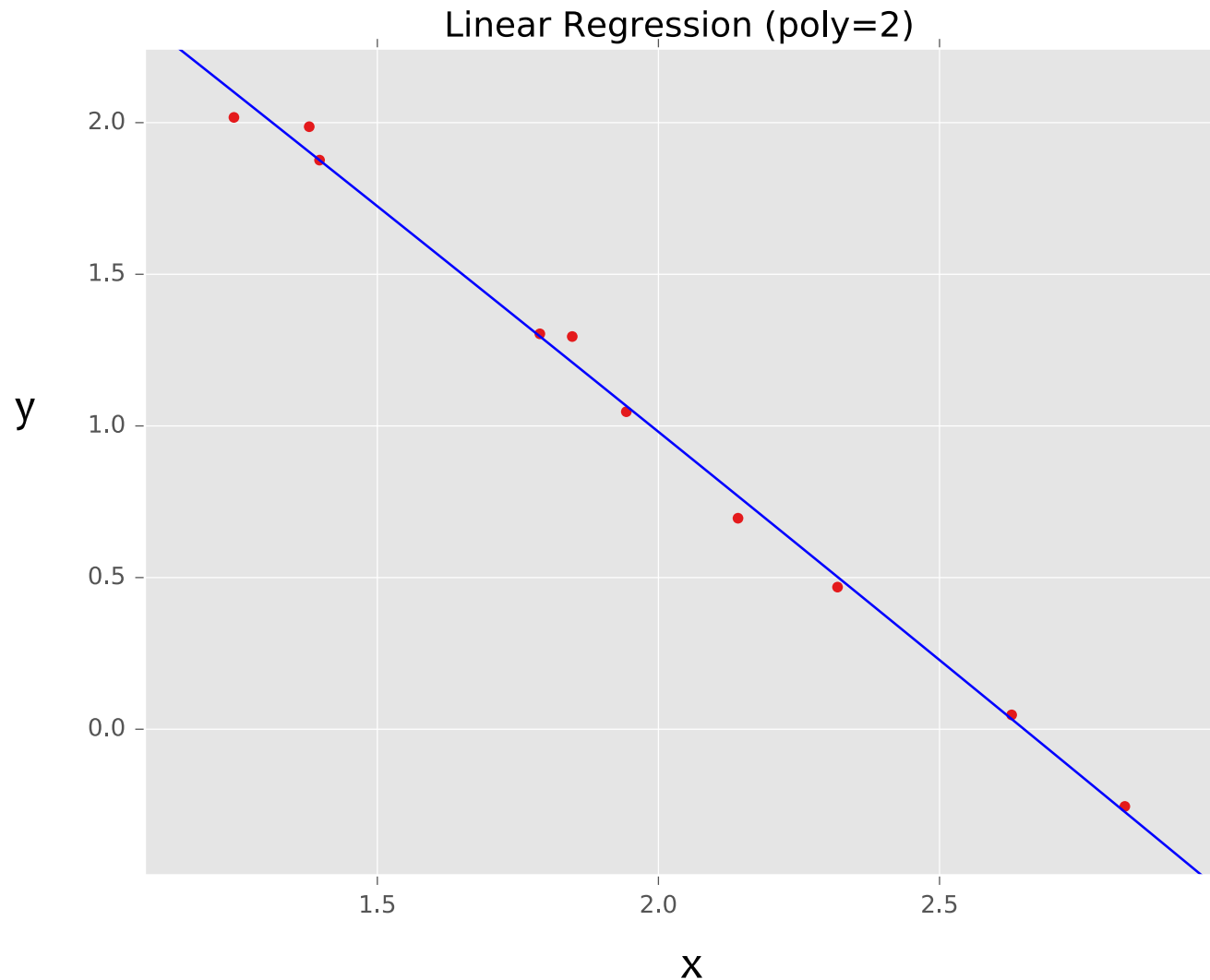
**Goal:** Learn  $y = \mathbf{w}^T \mathbf{f}(\mathbf{x}) + b$   
where  $\mathbf{f}(\cdot)$  is a polynomial  
basis function



true “unknown”  
target function is  
linear with  
negative slope  
and gaussian  
noise

# Example: Linear Regression

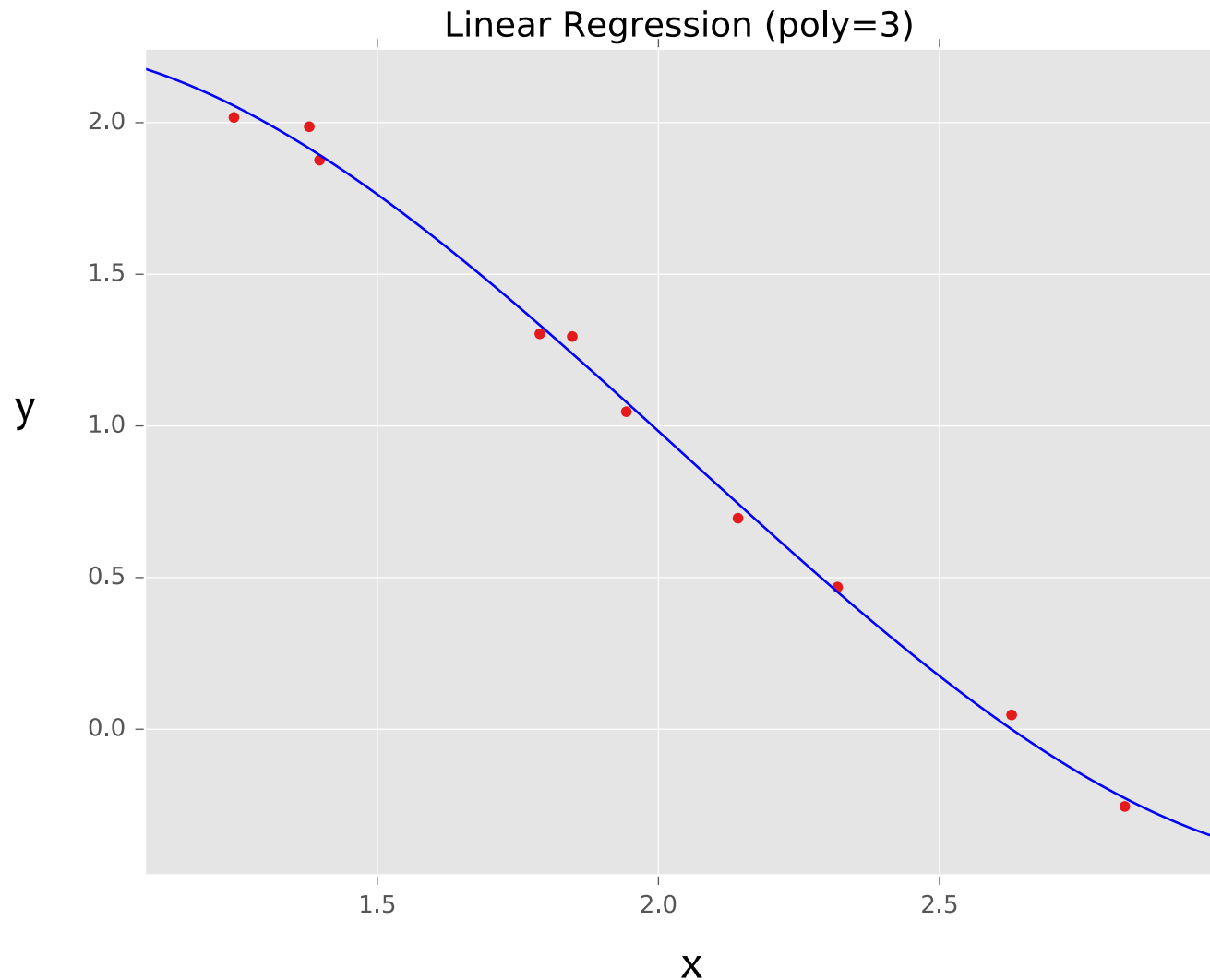
**Goal:** Learn  $y = \mathbf{w}^T \mathbf{f}(\mathbf{x}) + b$   
where  $\mathbf{f}(\cdot)$  is a polynomial  
basis function



true “unknown”  
target function is  
linear with  
negative slope  
and gaussian  
noise

# Example: Linear Regression

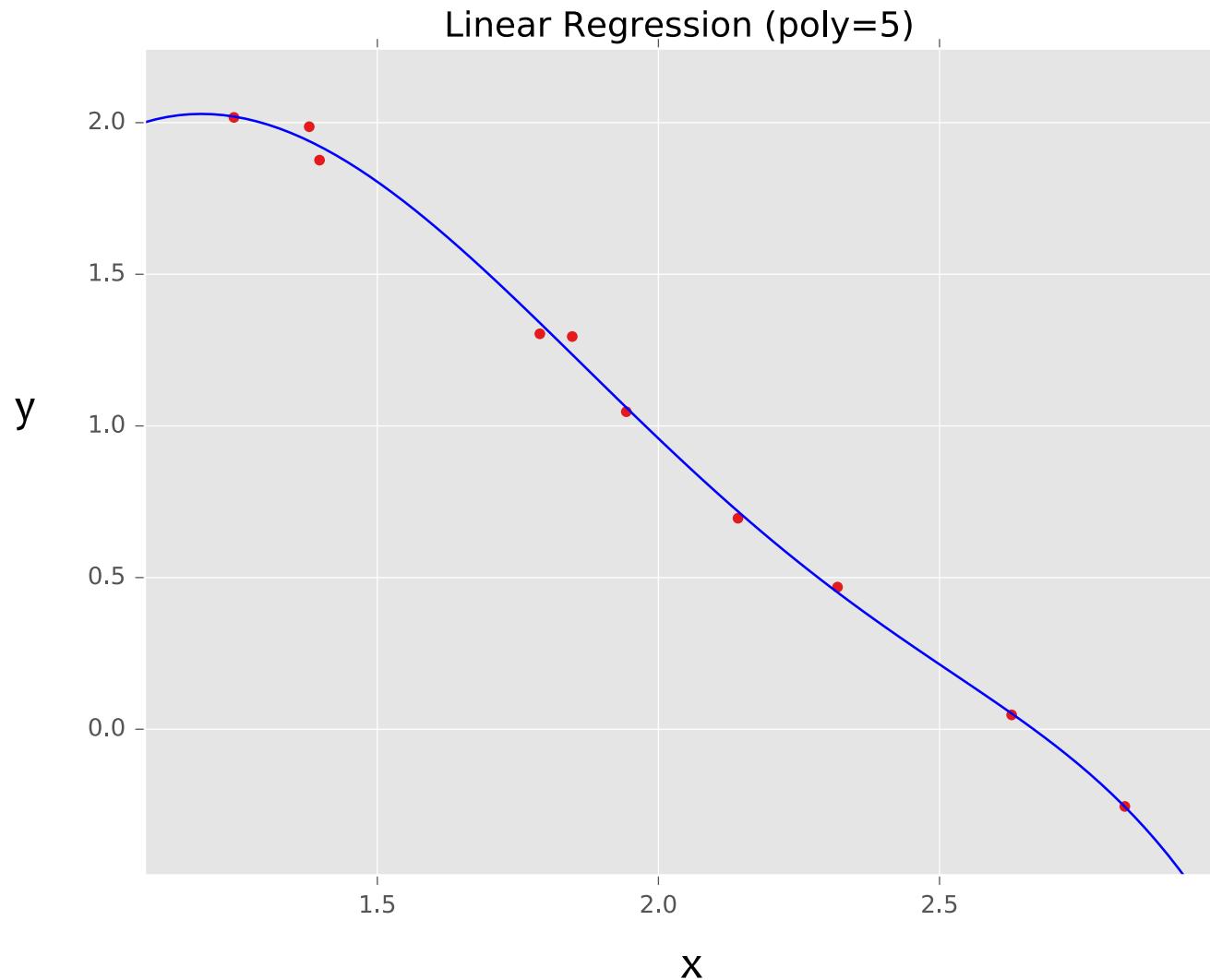
**Goal:** Learn  $y = \mathbf{w}^T \mathbf{f}(\mathbf{x}) + b$   
where  $\mathbf{f}(\cdot)$  is a polynomial  
basis function



true “unknown”  
target function is  
linear with  
negative slope  
and gaussian  
noise

# Example: Linear Regression

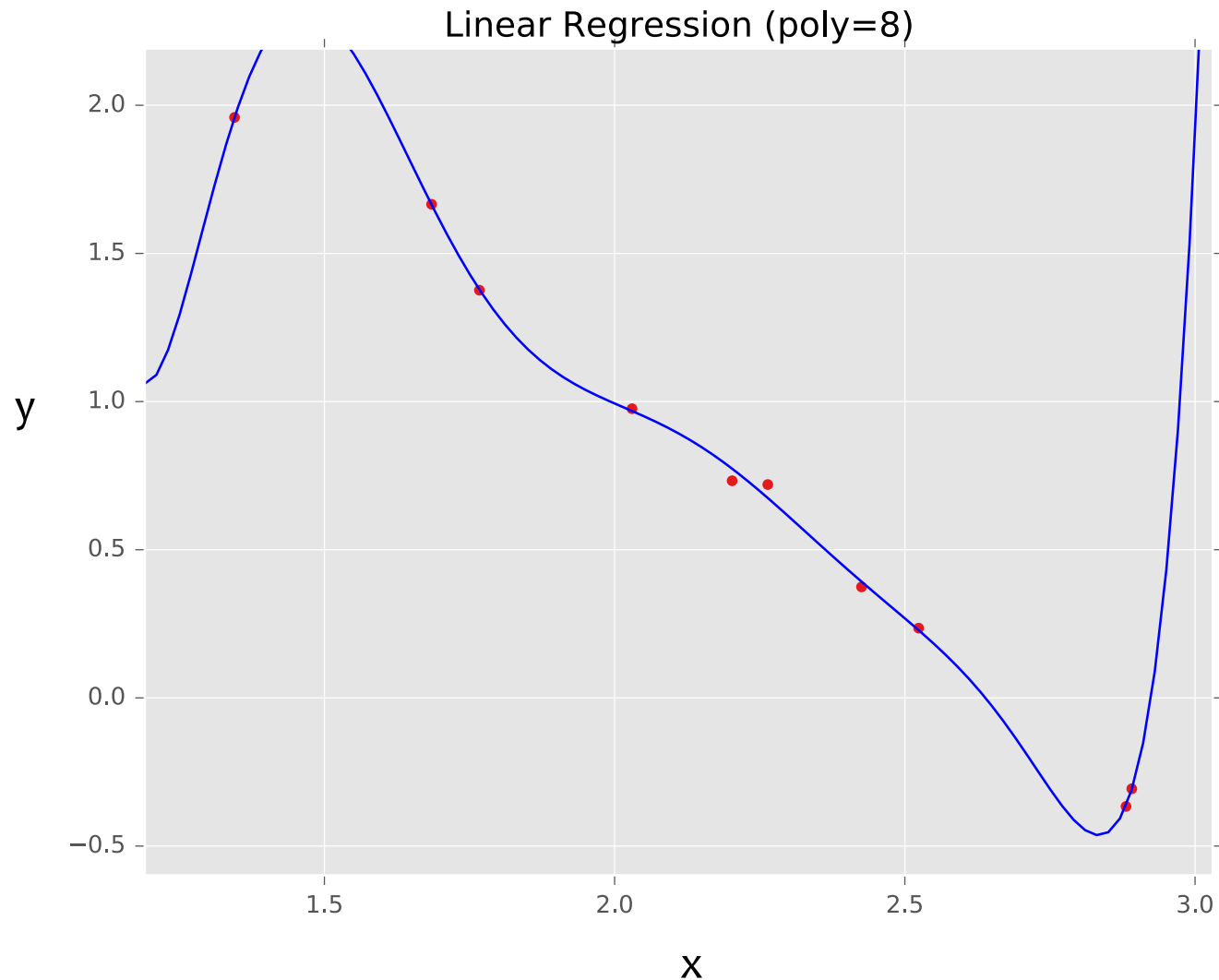
**Goal:** Learn  $y = \mathbf{w}^T \mathbf{f}(\mathbf{x}) + b$   
where  $\mathbf{f}(\cdot)$  is a polynomial  
basis function



true “unknown”  
target function is  
linear with  
negative slope  
and gaussian  
noise

# Example: Linear Regression

**Goal:** Learn  $y = \mathbf{w}^T \mathbf{f}(\mathbf{x}) + b$   
where  $\mathbf{f}(\cdot)$  is a polynomial  
basis function



true “unknown”  
target function is  
linear with  
negative slope  
and gaussian  
noise

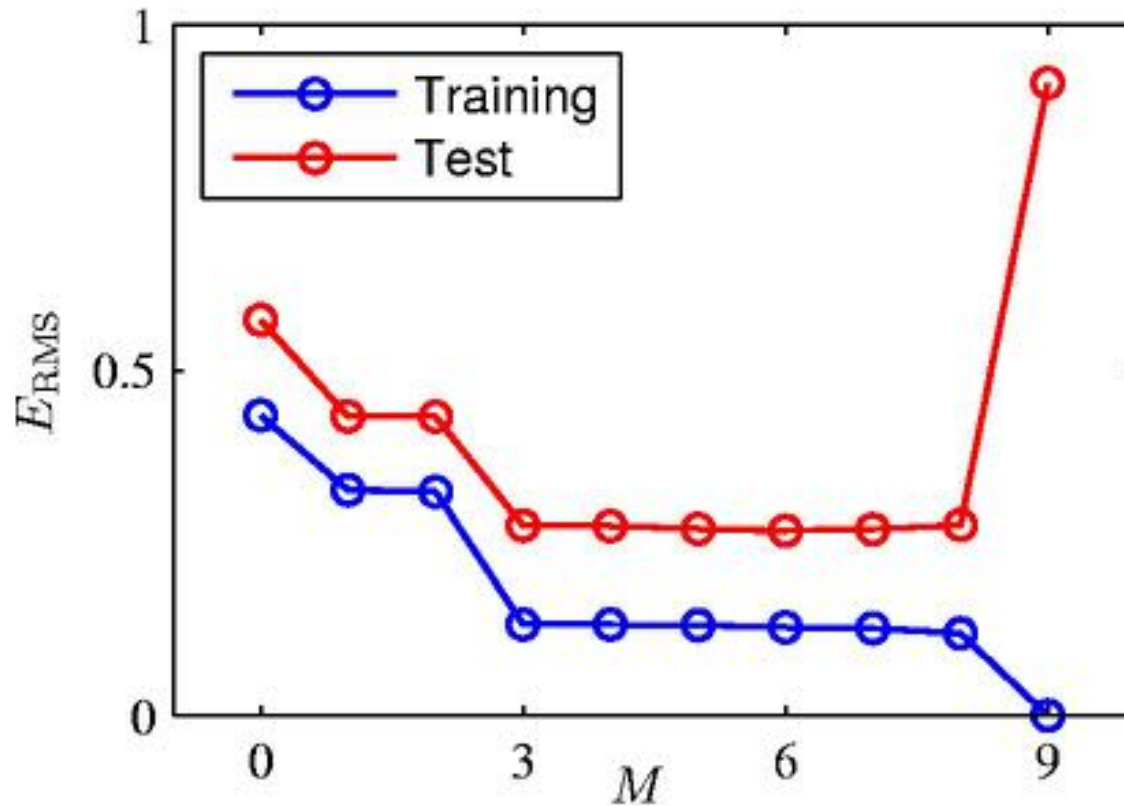
# Example: Linear Regression

**Goal:** Learn  $y = \mathbf{w}^T \mathbf{f}(\mathbf{x}) + b$   
where  $\mathbf{f}(\cdot)$  is a polynomial  
basis function



true “unknown”  
target function is  
linear with  
negative slope  
and gaussian  
noise

# Over-fitting



Root-Mean-Square (RMS) Error:  $E_{\text{RMS}} = \sqrt{2E(\mathbf{w}^*)/N}$

# Polynomial Coefficients

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$\theta_0$	0.19	0.82	0.31	0.35
$\theta_1$		-1.27	7.99	232.37
$\theta_2$			-25.43	-5321.83
$\theta_3$			17.37	48568.31
$\theta_4$				-231639.30
$\theta_5$				640042.26
$\theta_6$				-1061800.52
$\theta_7$				1042400.18
$\theta_8$				-557682.99
$\theta_9$				125201.43



# Example: Linear Regression

**Goal:** Learn  $y = \mathbf{w}^T \mathbf{f}(\mathbf{x}) + b$   
where  $\mathbf{f}(\cdot)$  is a polynomial  
basis function

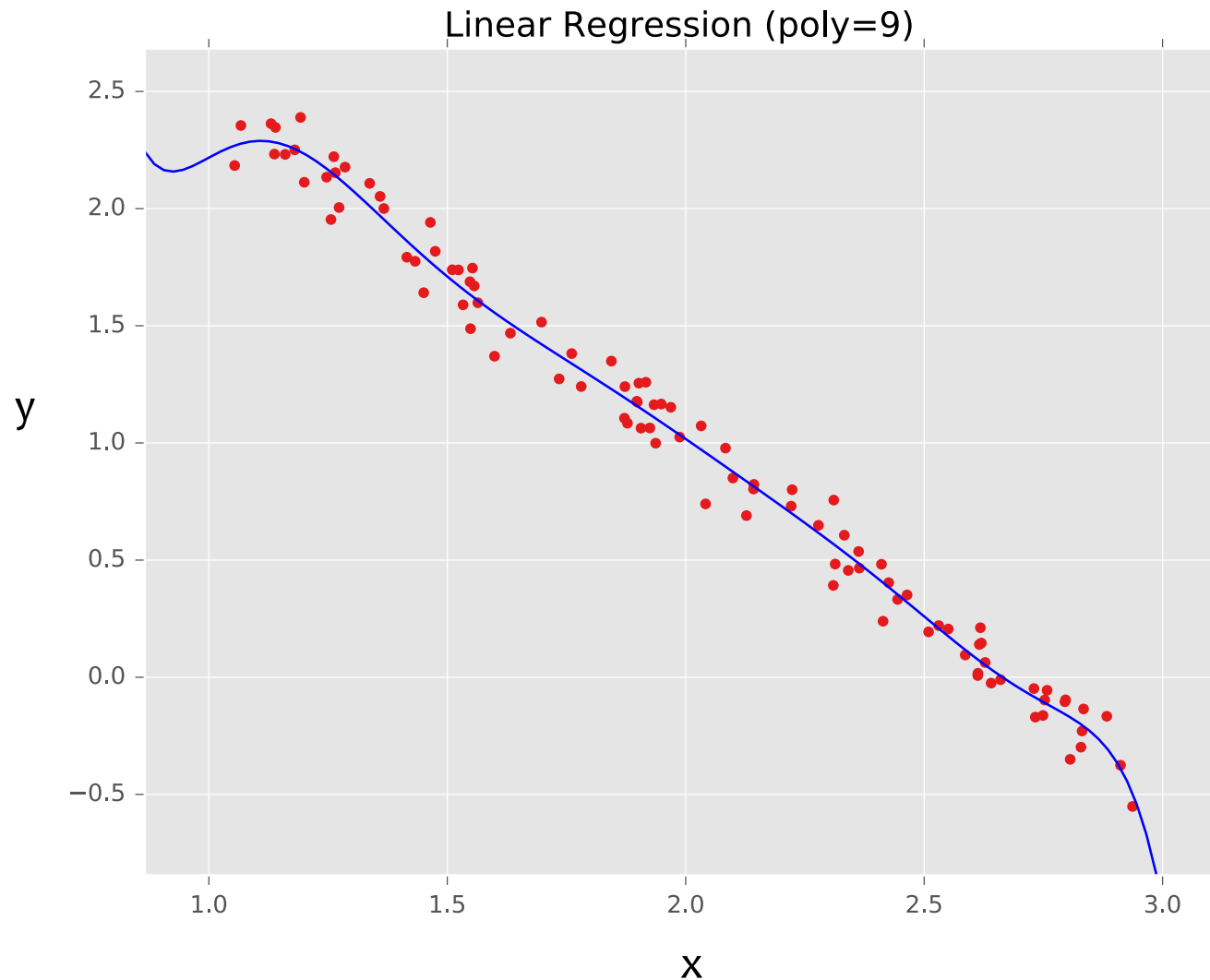


true “unknown”  
target function is  
linear with  
negative slope  
and gaussian  
noise

# Example: Linear Regression

**Goal:** Learn  $y = \mathbf{w}^T \mathbf{f}(\mathbf{x}) + b$   
where  $\mathbf{f}(\cdot)$  is a polynomial  
basis function

Same as before, but now  
with  $N = 100$  points



true “unknown”  
target function is  
linear with  
negative slope  
and gaussian  
noise

# 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

## Example: Stock Prices

- Suppose we wish to predict Google's stock price at time  $t+1$
- **What features should we use?**  
(putting all computational concerns aside)
  - Stock prices of all other stocks at times  $t, t-1, t-2, \dots, t-k$
  - Mentions of Google with positive / negative sentiment words in all newspapers and social media outlets
- Do we believe that **all** of these features are going to be useful?



# 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:  $\hat{\theta} = \underset{\theta}{\operatorname{argmin}} J(\theta) + \lambda r(\theta)$
- **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)  $r(\theta) = \|\theta\|_q = \left[ \sum_{m=1}^M \|\theta_m\|^q \right]^{\left(\frac{1}{q}\right)}$

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

# Regularization

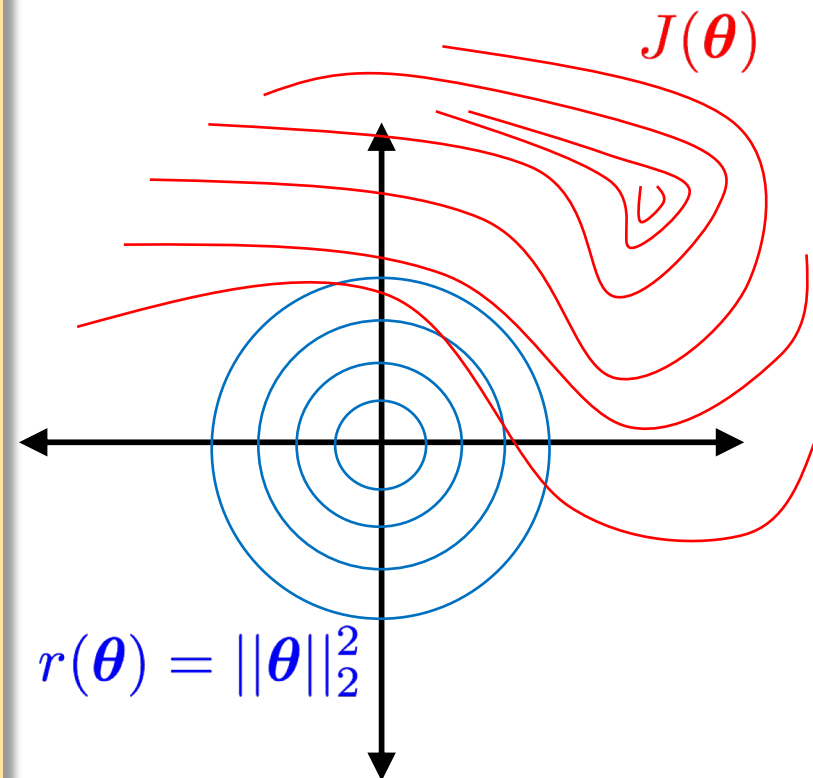
## Question:

Suppose we are minimizing  $J'(\theta)$  where

$$J'(\theta) = J(\theta) + \lambda r(\theta)$$

As  $\lambda$  increases, the minimum of  $J'(\theta)$  will move...

- A. ...towards the midpoint between  $J'(\theta)$  and  $r(\theta)$
- B. ...towards the minimum of  $J(\theta)$
- C. ...towards the minimum of  $r(\theta)$
- D. ...towards a theta vector of positive infinities
- E. ...towards a theta vector of negative infinities





# Regularization Exercise

## *In-class Exercise*

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



# Regularization

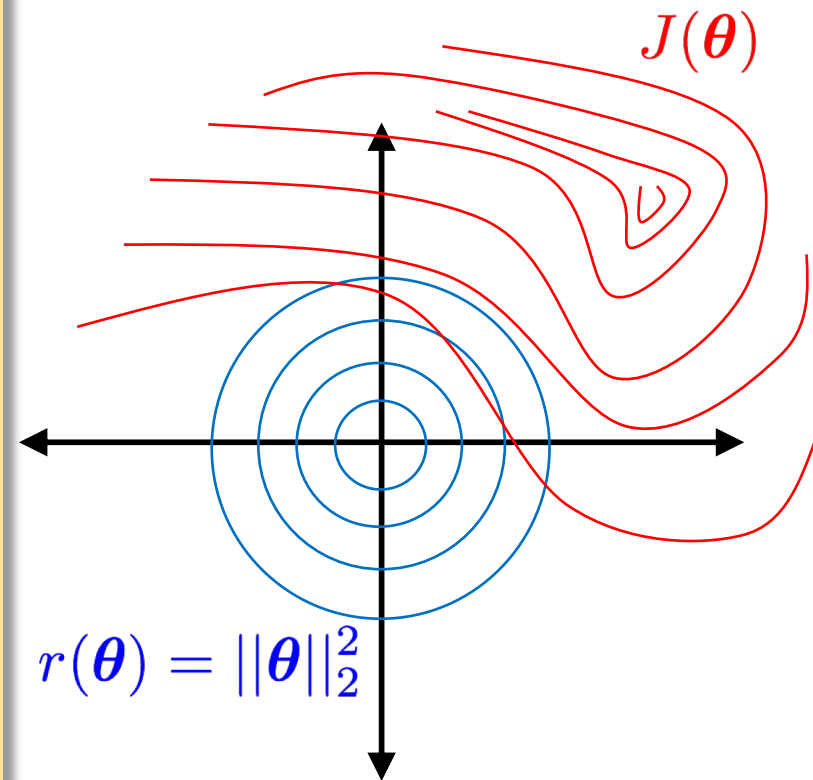
## Question:

Suppose we are minimizing  $J'(\theta)$  where

$$J'(\theta) = J(\theta) + \lambda r(\theta)$$

As we increase  $\lambda$  from 0, the the validation error will...

- A. ...increase
- B. ...decrease
- C. ...first increase, then decrease
- D. ...first decrease, then increase
- E. ...stay the same



# Regularization

## Don't Regularize the Bias (Intercept) Parameter!

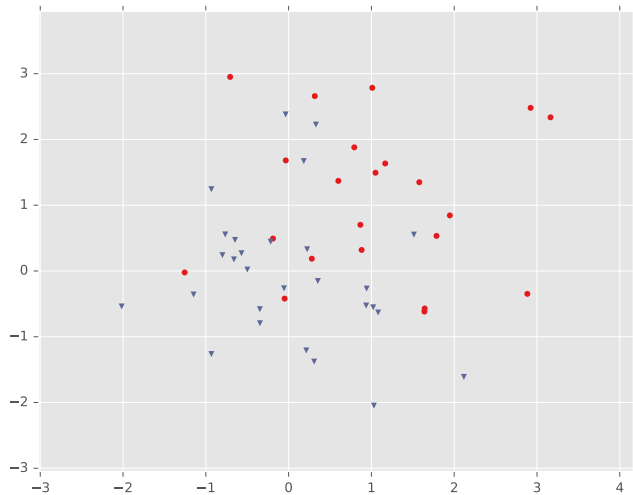
- In our models so far, the bias / intercept parameter is usually denoted by  $\theta_0$  -- that is, the parameter for which we fixed  $x_0 = 1$
- Regularizers always avoid penalizing this bias / intercept parameter
- Why? Because otherwise the learning algorithms wouldn't be invariant to a shift in the y-values

## Whitening Data

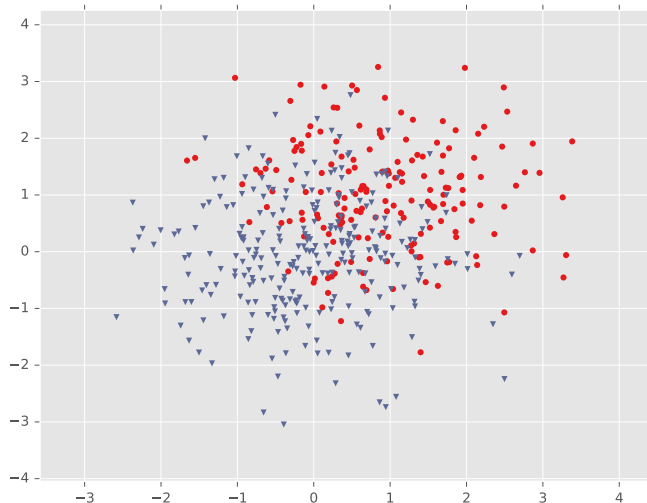
- It's common to *whiten* each feature by subtracting its mean and dividing by its variance
- For regularization, this helps all the features be penalized in the same units  
(e.g. convert both centimeters and kilometers to z-scores)

# Example: Logistic Regression

Training  
Data

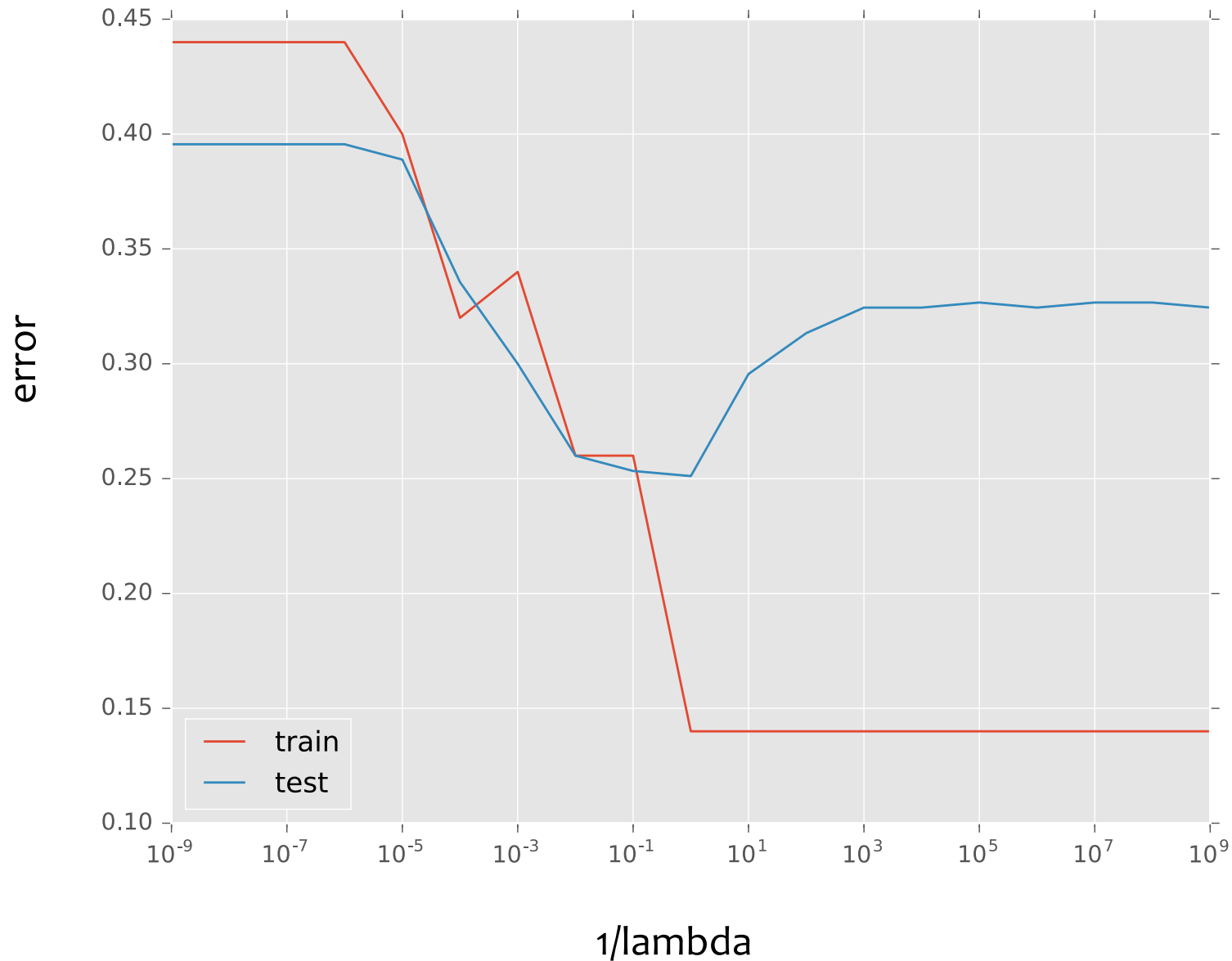


Test  
Data

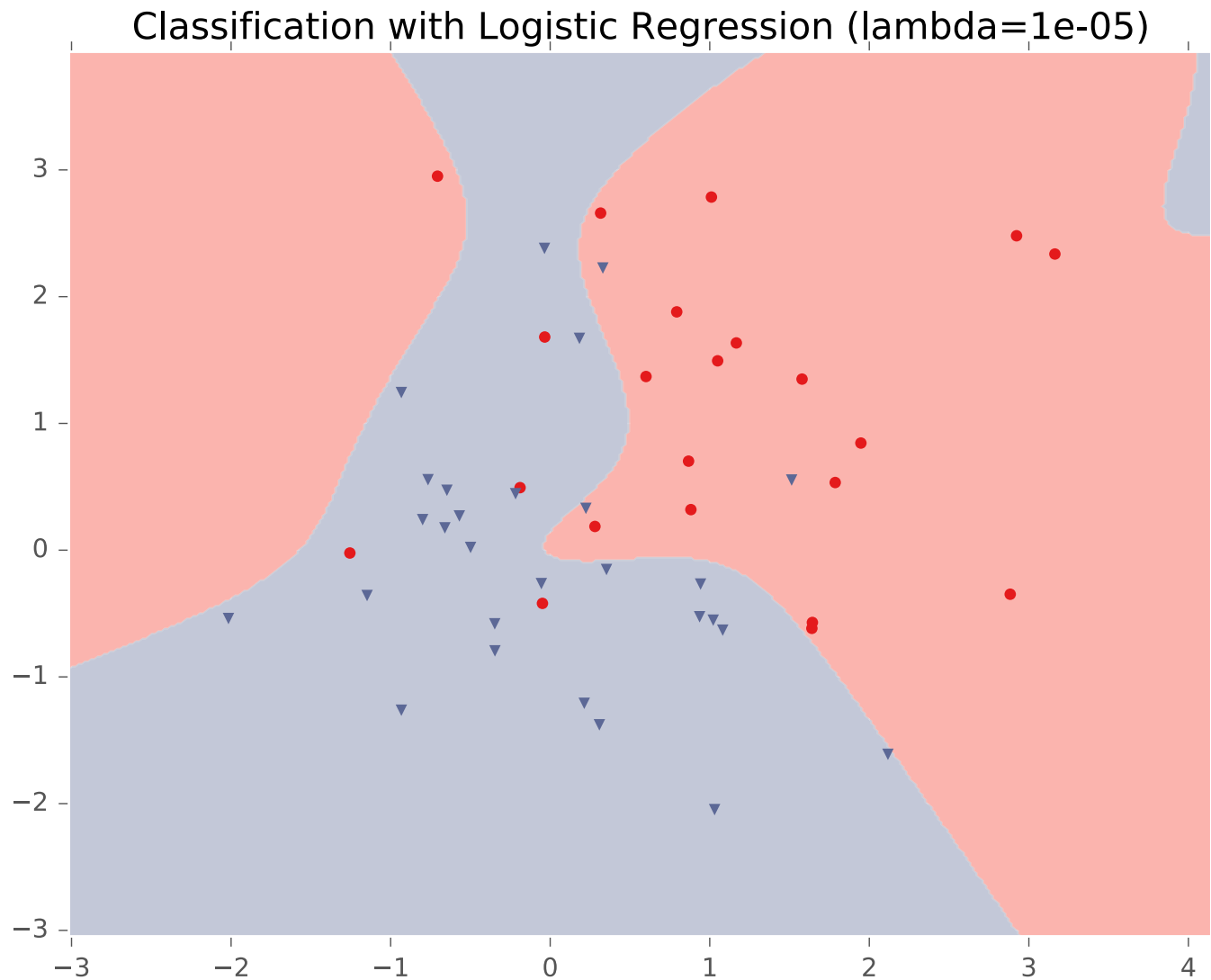


- For this example, we construct **nonlinear features** (i.e. feature engineering)
- Specifically, we add **polynomials up to order 9** of the two original features  $x_1$  and  $x_2$
- Thus our classifier is **linear** in the **high-dimensional feature space**, but the decision boundary is **nonlinear**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

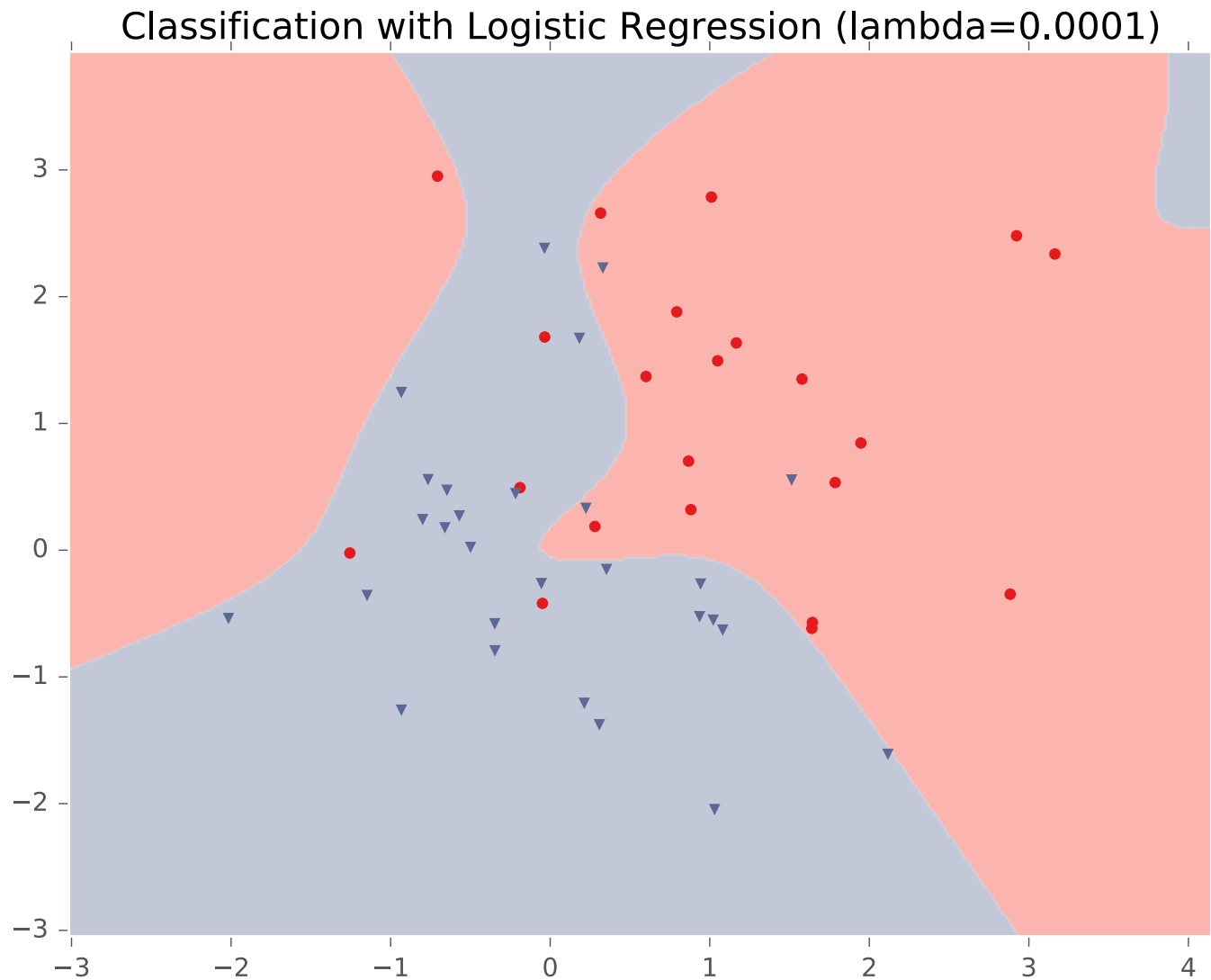
# Example: Logistic Regression



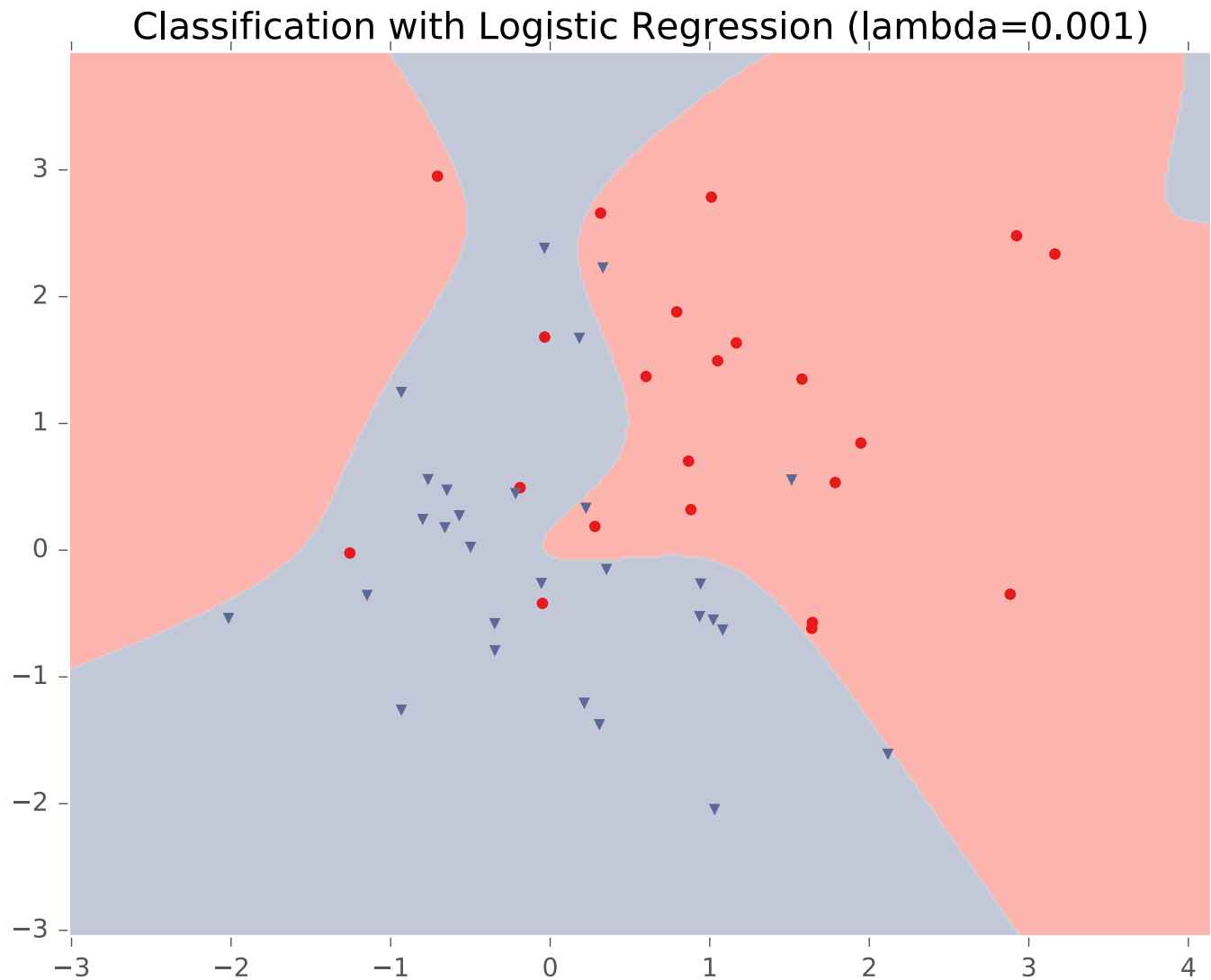
# Example: Logistic Regression



# Example: Logistic Regression

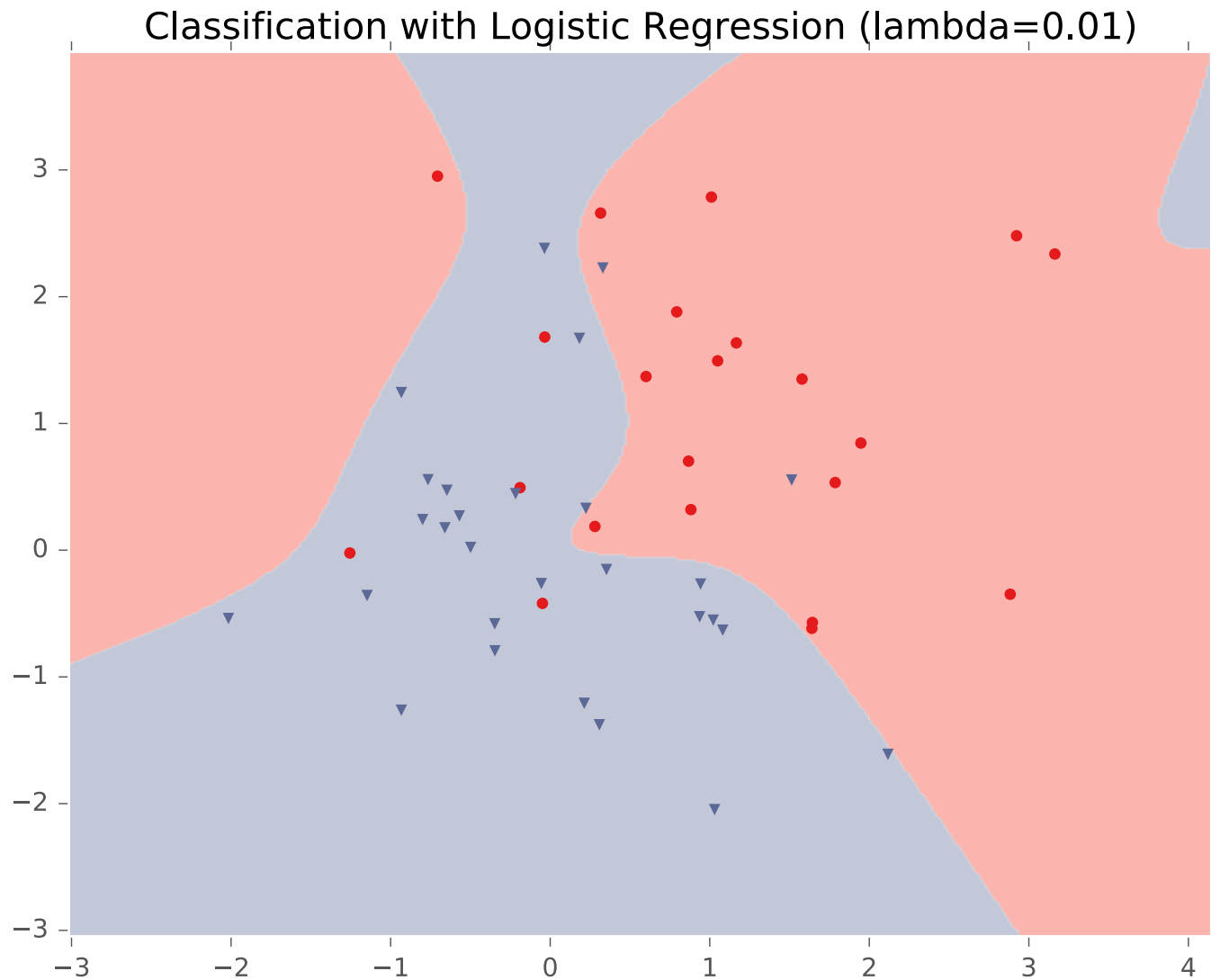


# Example: Logistic Regression

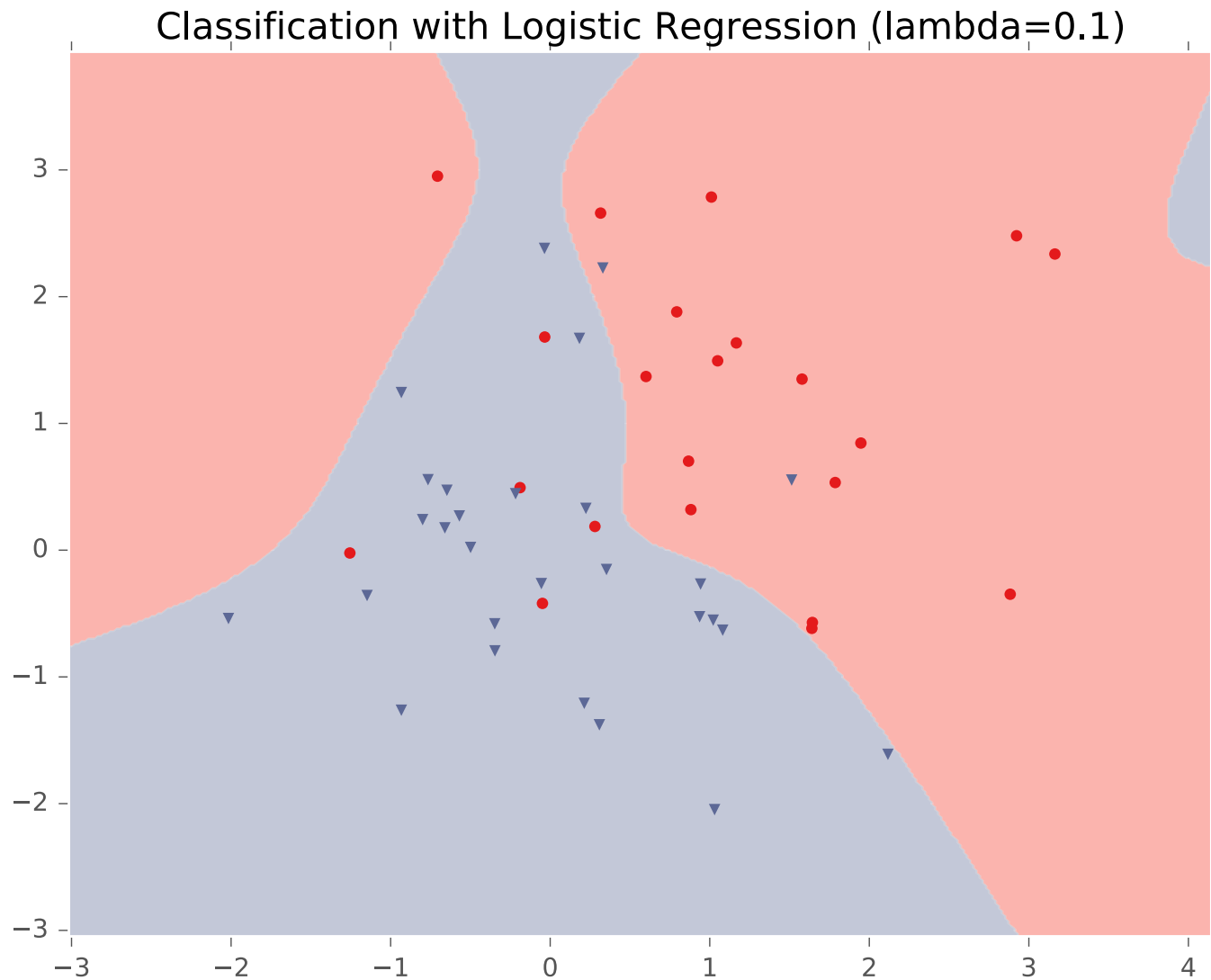




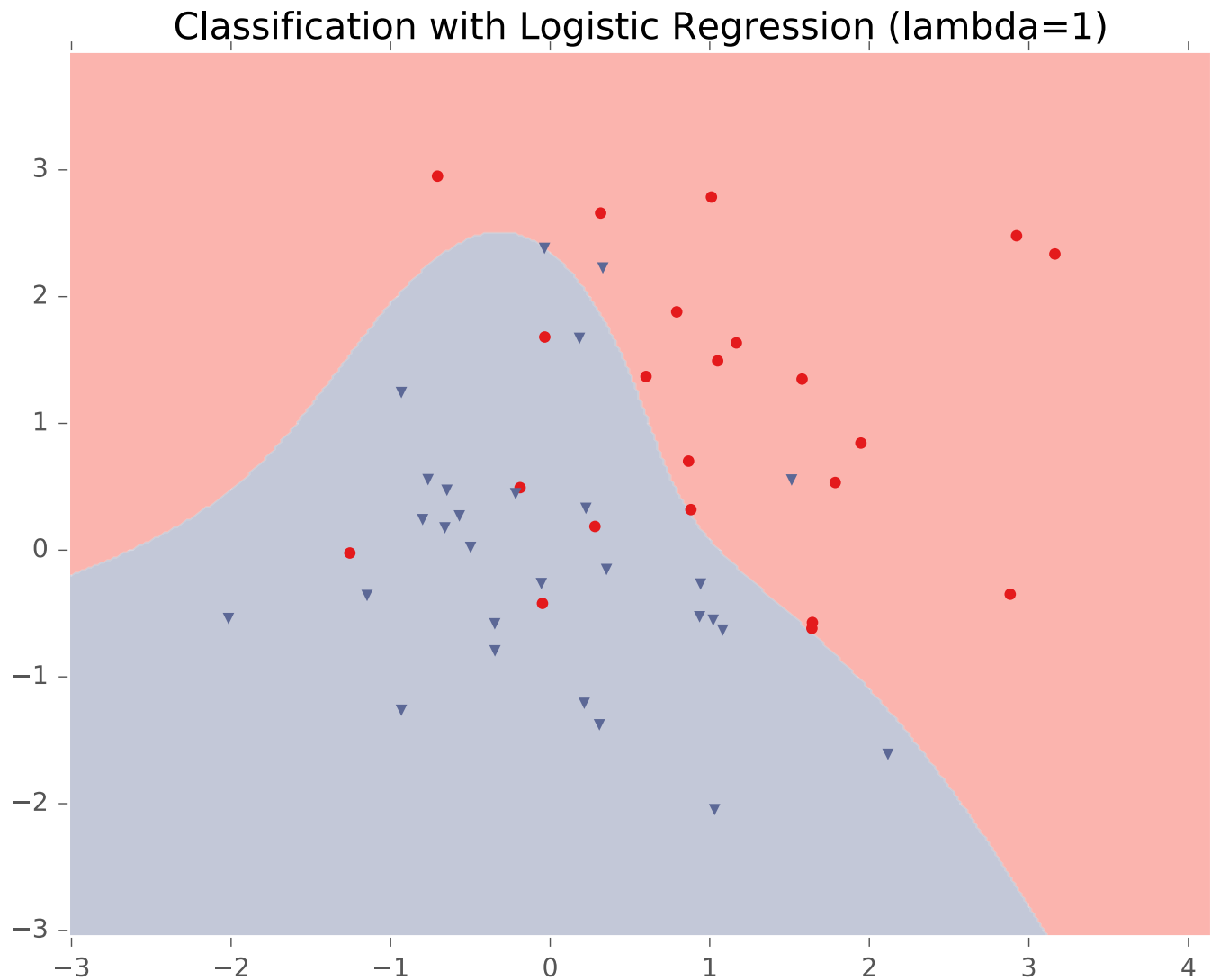
# Example: Logistic Regression



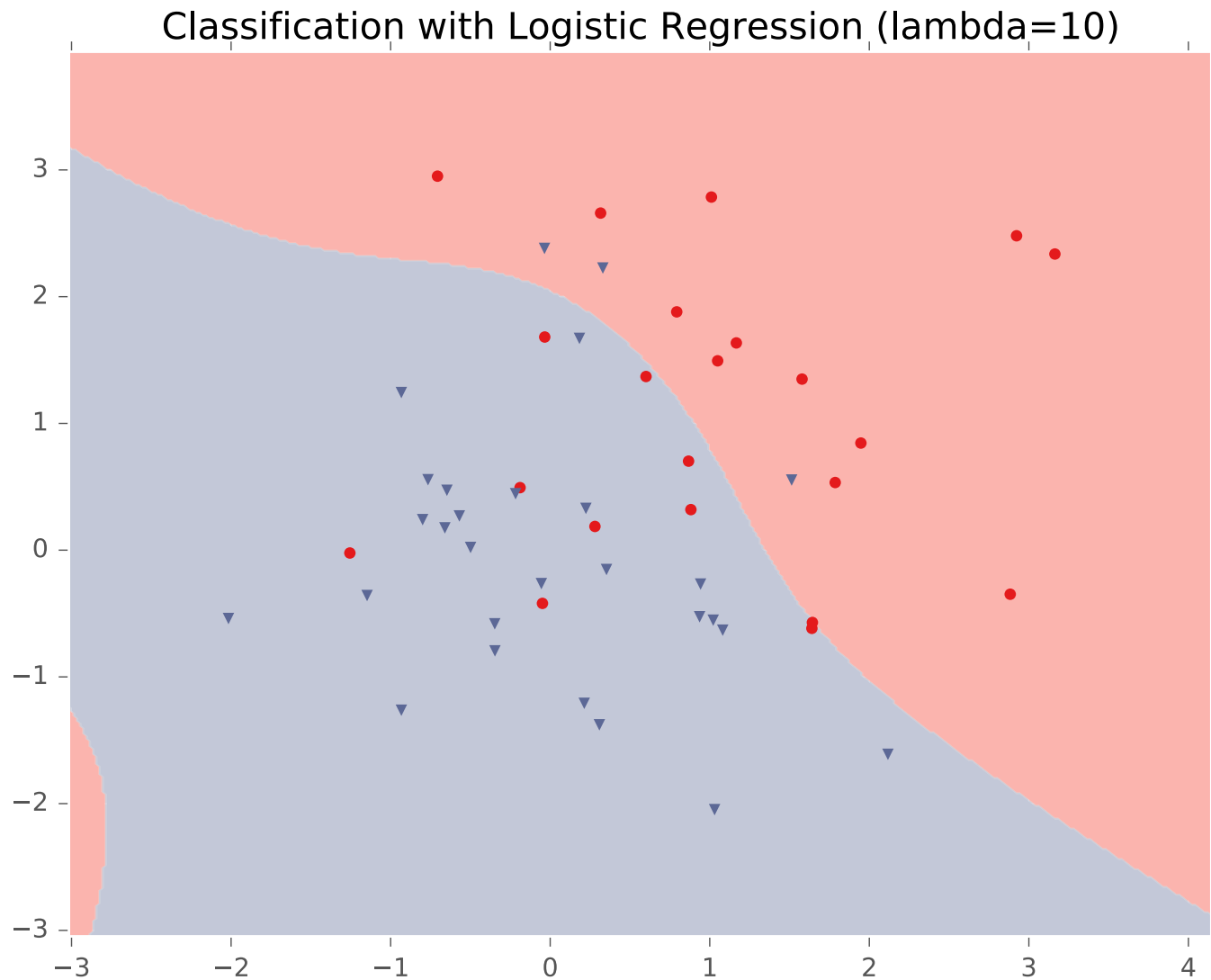
# Example: Logistic Regression



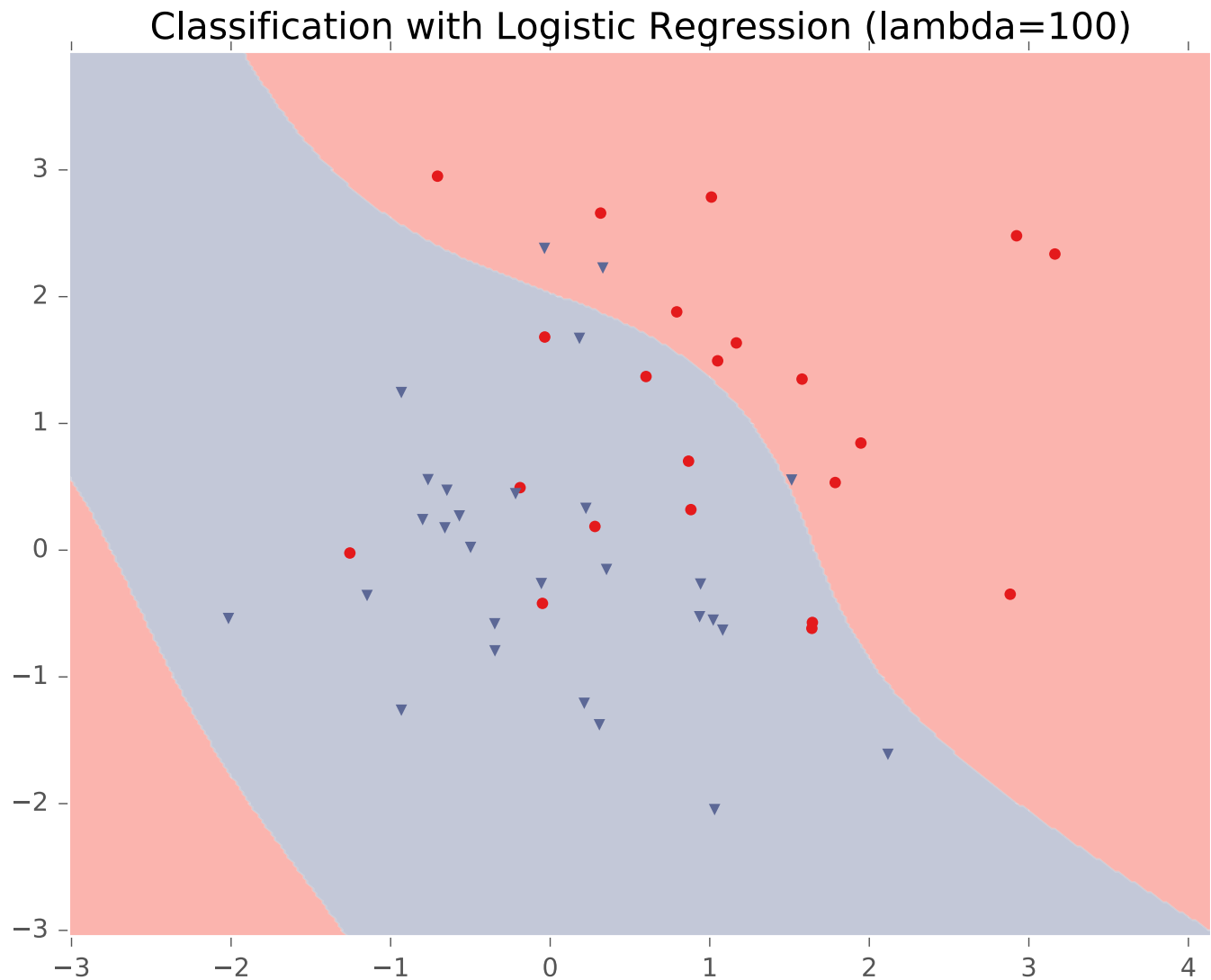
# Example: Logistic Regression



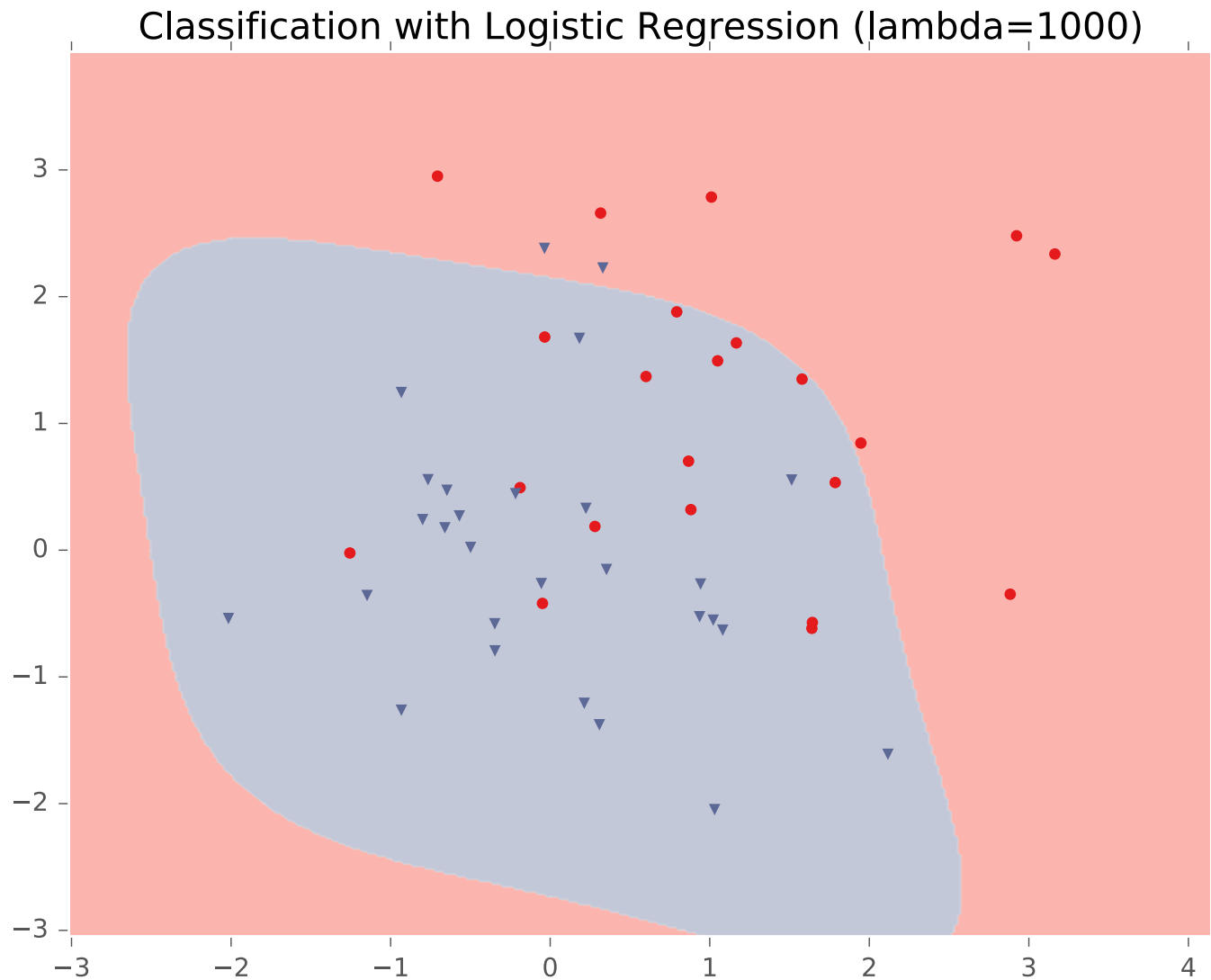
# Example: Logistic Regression



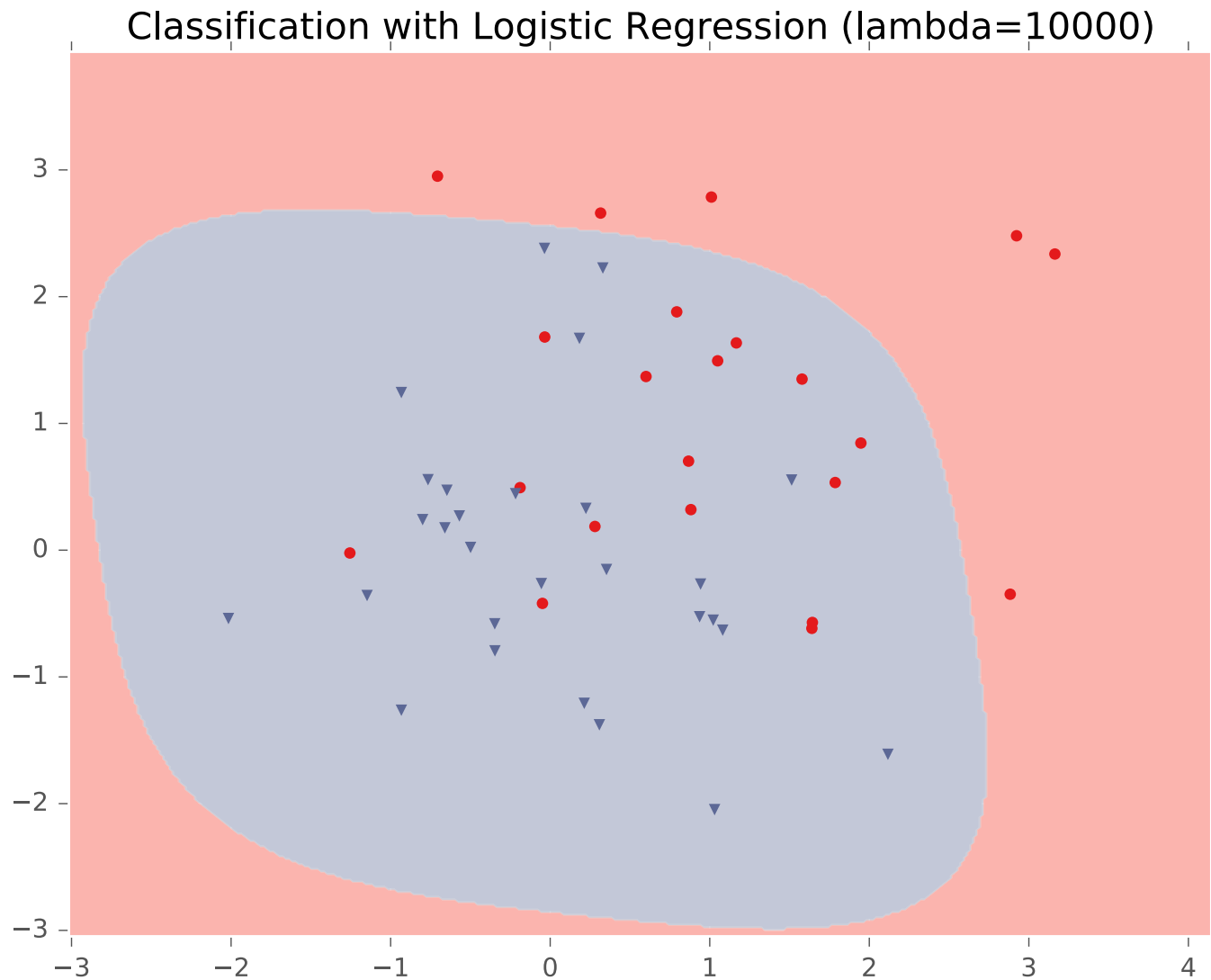
# Example: Logistic Regression



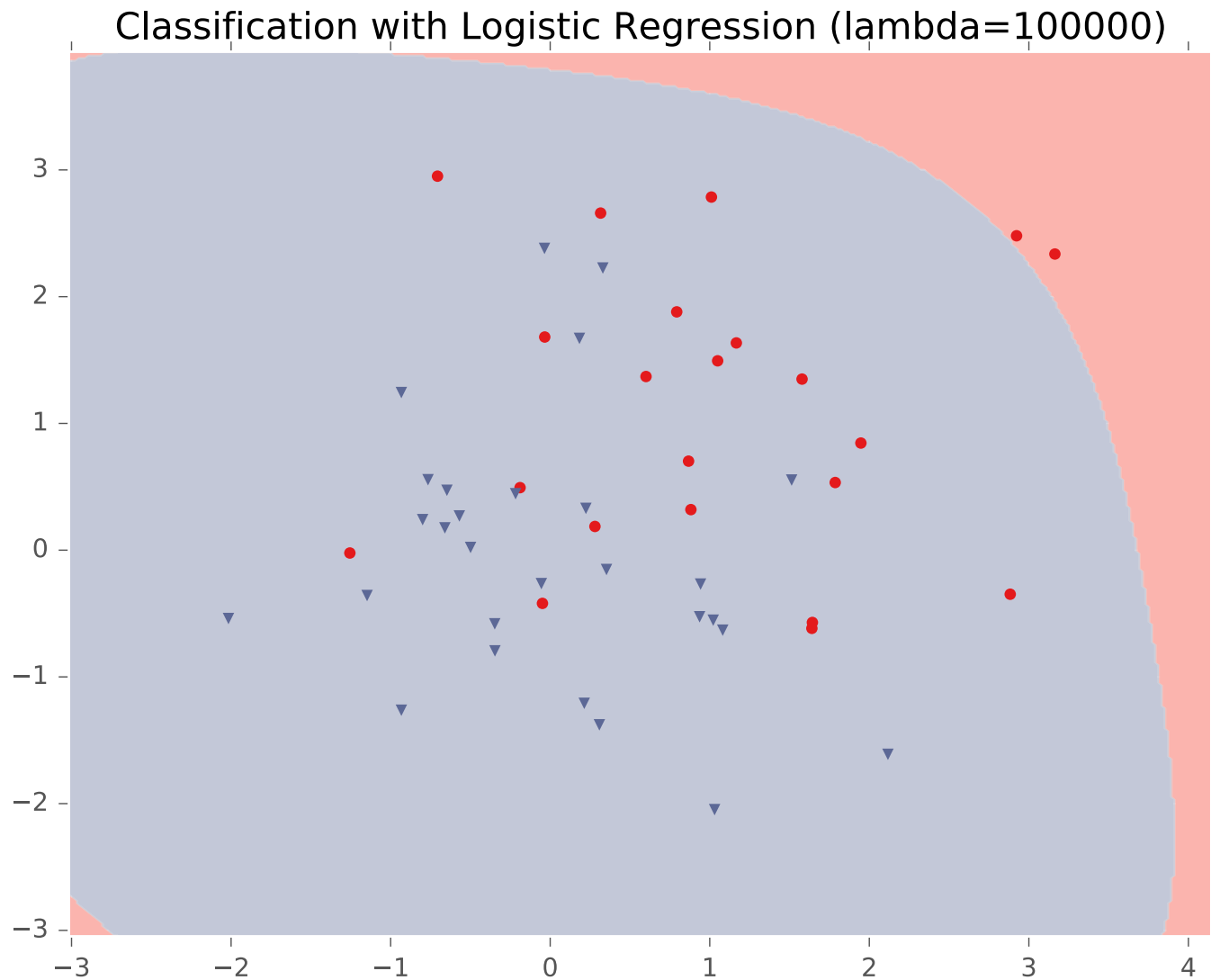
# Example: Logistic Regression



# Example: Logistic Regression

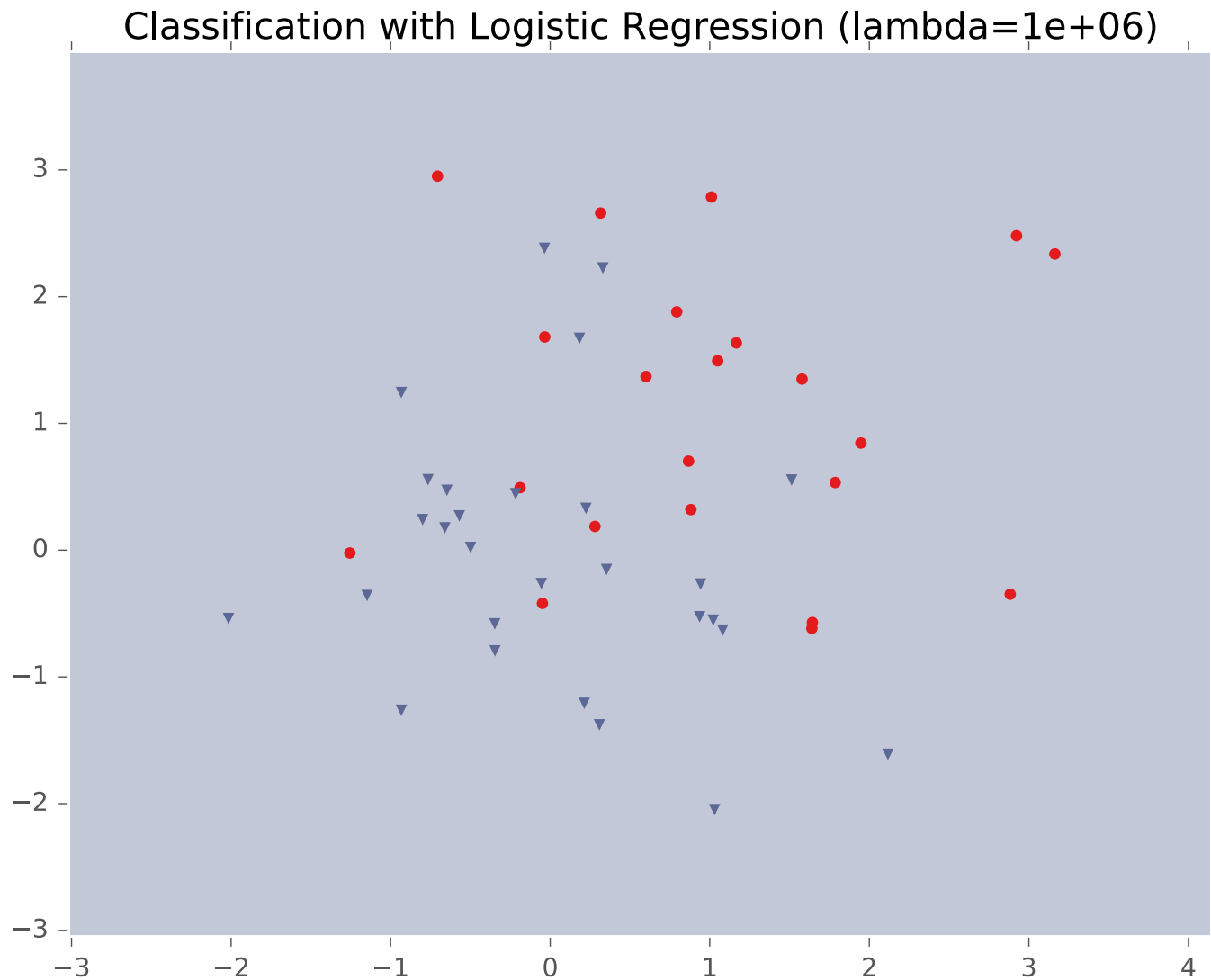


# Example: Logistic Regression

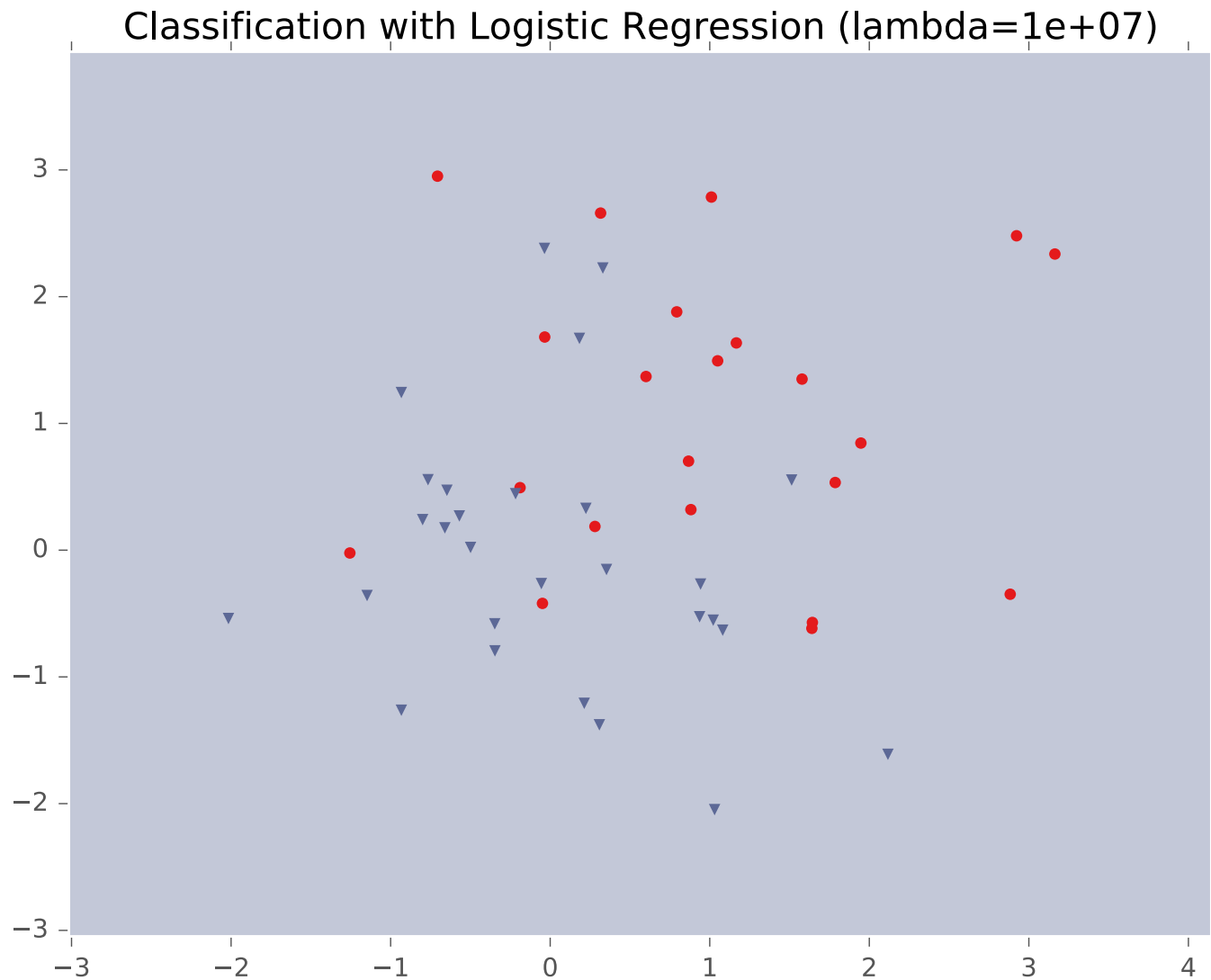




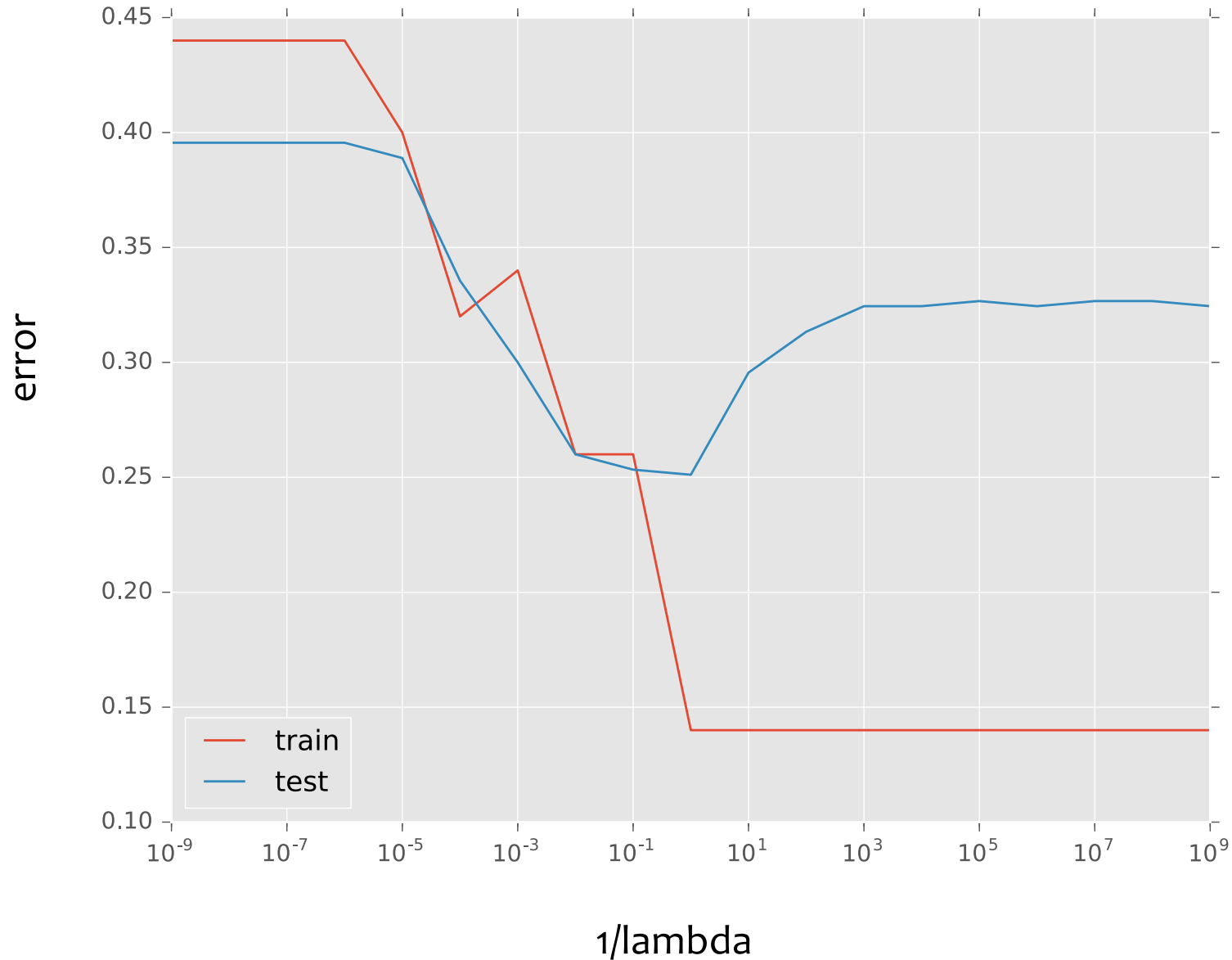
# Example: Logistic Regression



# Example: Logistic Regression



# Example: Logistic Regression



# 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 **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 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)

# 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


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

## 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)


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

# 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)

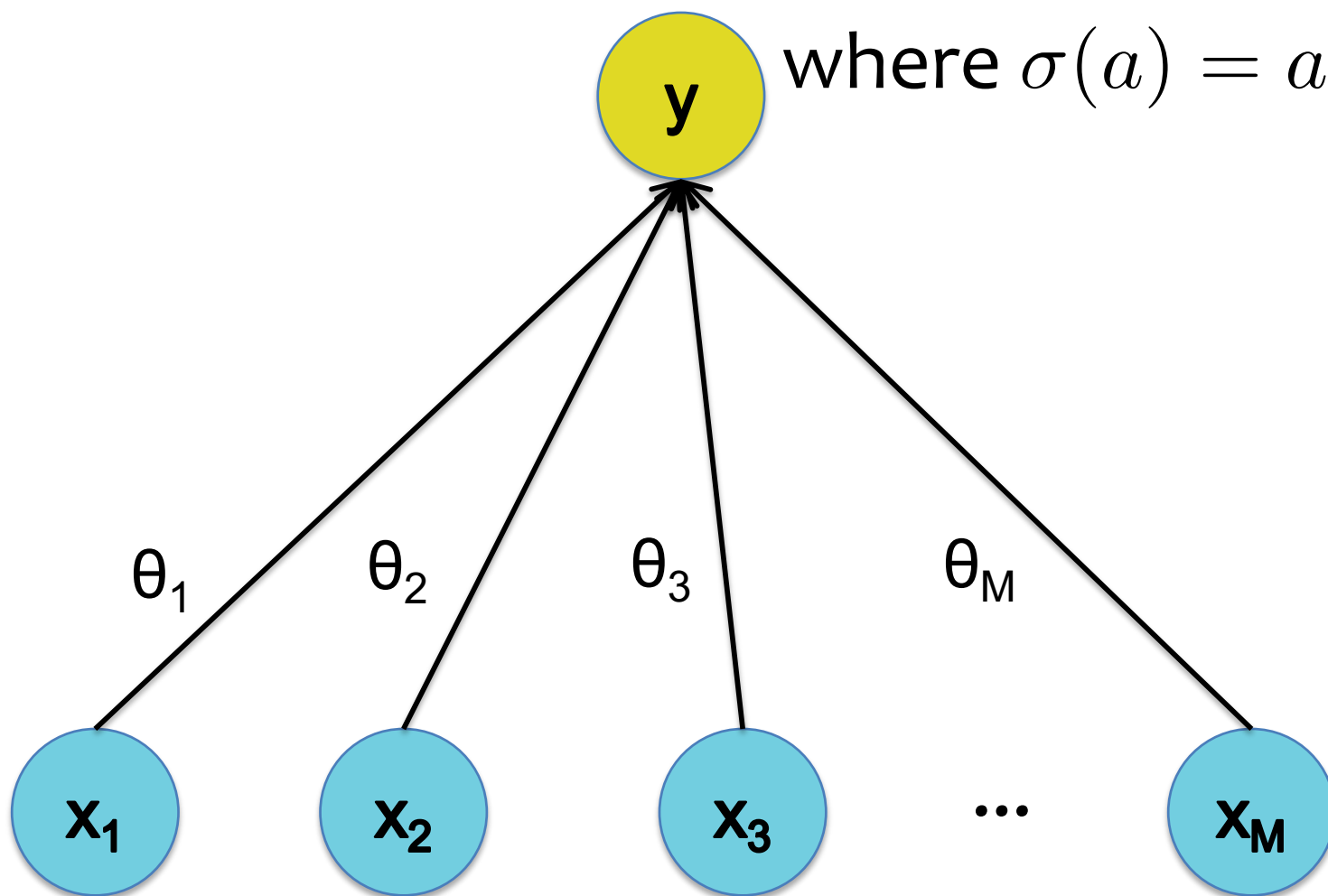
$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \nabla \ell(f_{\theta}(x_i), y_i)$$

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

where  $\sigma(a) = a$

Output

Input



## Decision Functions

# Logistic Regression

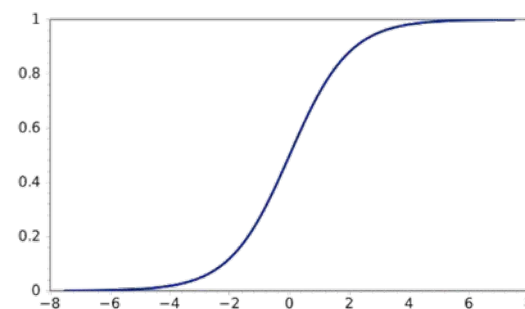
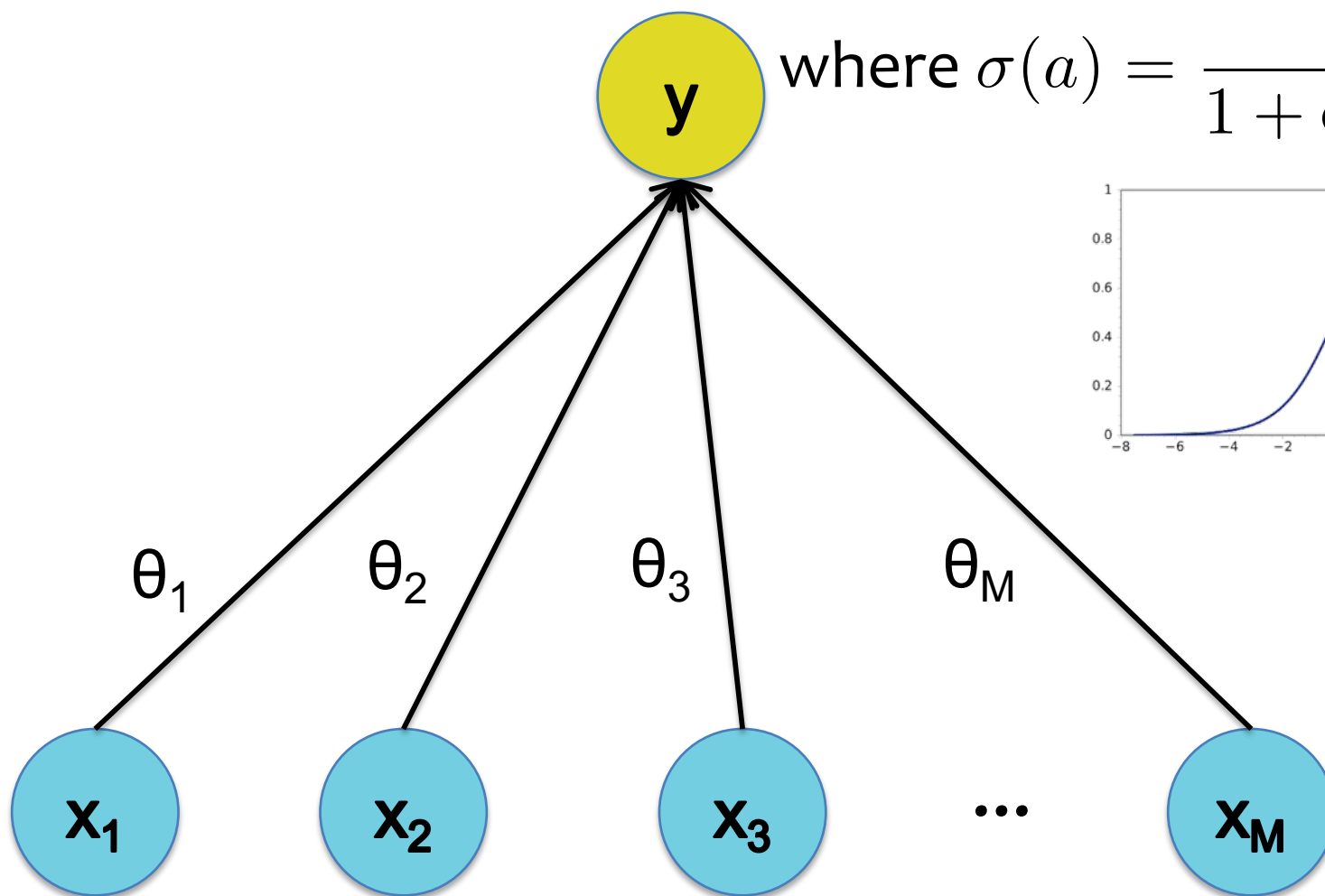
$$y = h_{\theta}(x) = \sigma(\theta^T x)$$

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

Output



Input



## Decision Functions

# Logistic Regression

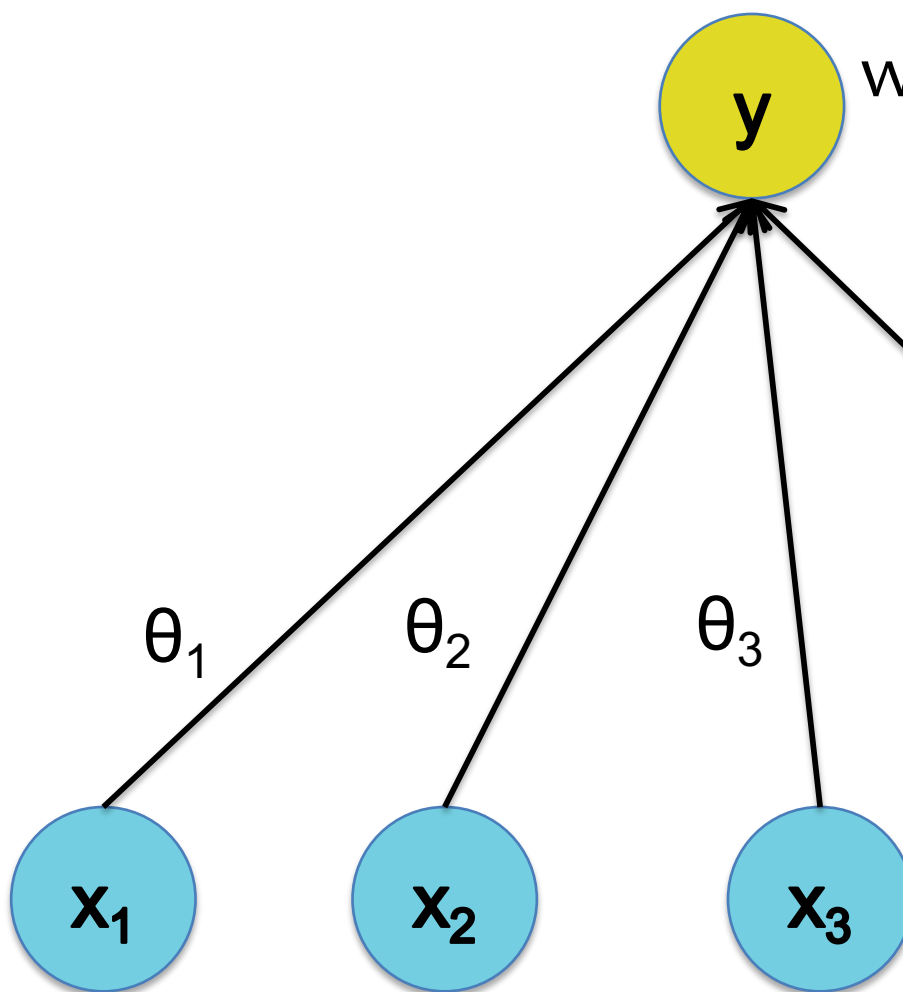
$$y = h_{\theta}(x) = \sigma(\theta^T x)$$

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

Output



Input



Face



Face



Not a face



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

Output

**y**

w

$\theta_1$

$\theta_2$

$\theta_3$

Input

**x<sub>1</sub>**

**x<sub>2</sub>**

**x<sub>3</sub>**

## In-Class Example

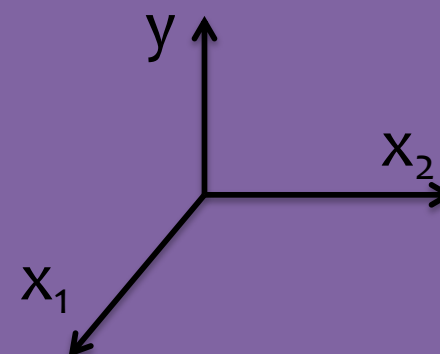
1



1



0





## Decision Functions

# Perceptron

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

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

Output

Input

