10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Binary Logistic Regression

+

Multinomial Logistic Regression

Matt Gormley

Lecture 10

Feb. 17, 2020

Reminders

- **Midterm Exam 1**
  - **Tue, Feb. 18, 7:00pm – 9:00pm**
- **Homework 4: Logistic Regression**
  - **Out: Wed, Feb. 19**
  - **Due: Fri, Feb. 28 at 11:59pm**
- **Today's In-Class Poll**
  - **http://p10.mlcourse.org**
- *Reading on Probabilistic Learning is reused later in the course for MLE/MAP*

MLE

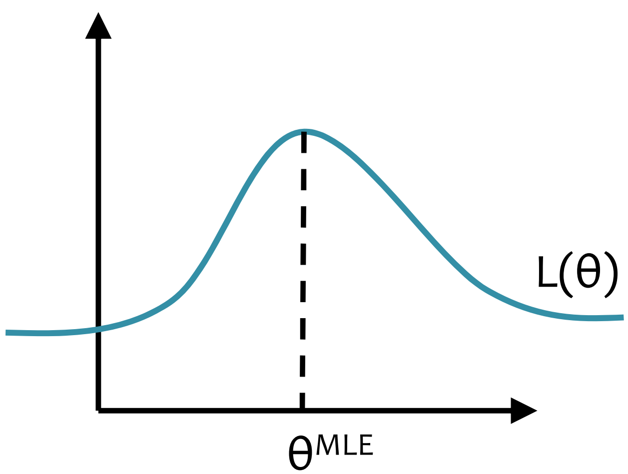
Suppose we have data $\mathcal{D} = \{x^{(i)}\}_{i=1}^{N}$

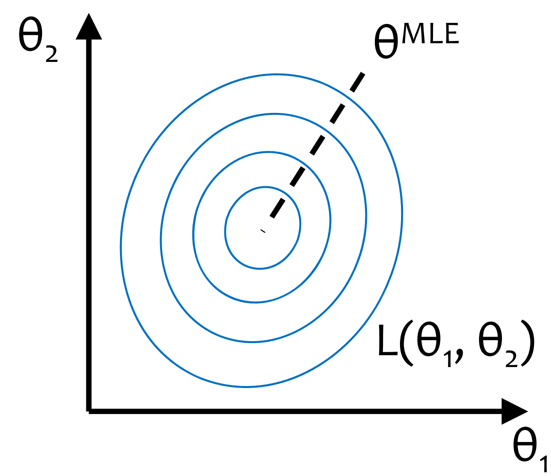
Principle of Maximum Likelihood Estimation:
Choose the parameters that maximize the likelihood of the data.

$$\boldsymbol{\theta}^{\text{MLE}} = \underset{\boldsymbol{\theta}}{\text{argmax}} \prod_{i=1}^{N} p(\mathbf{x}^{(i)}|\boldsymbol{\theta})$$

Maximum Likelihood Estimate (MLE)

$(x^{(i)}, y^{(i)})$





MLE

What does maximizing likelihood accomplish?

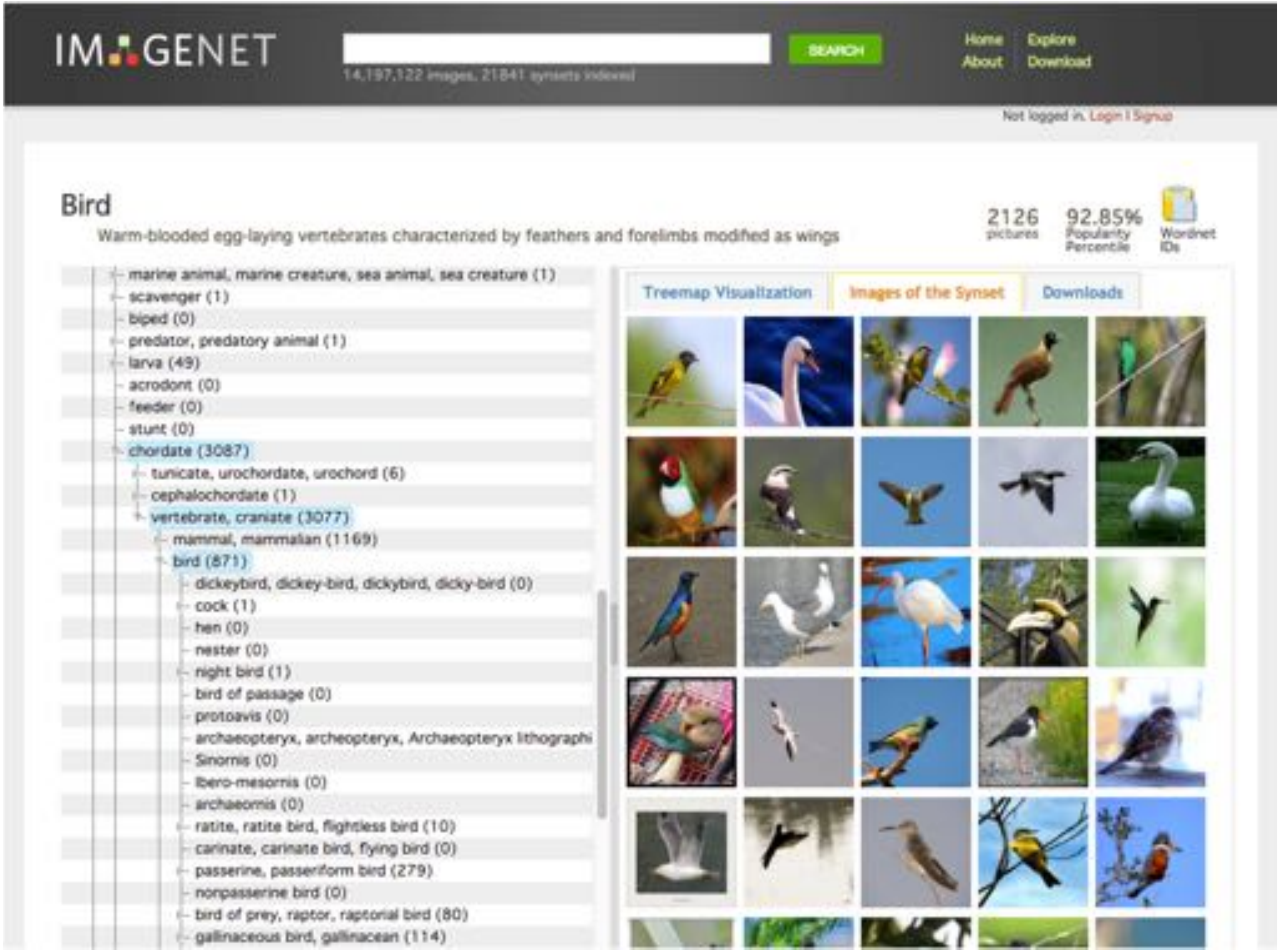
- There is only a finite amount of probability mass (i.e. sum-to-one constraint)
- MLE tries to allocate **as much** probability mass **as possible** to the things we have observed…

  …**at the expense** of the things we have **not** observed

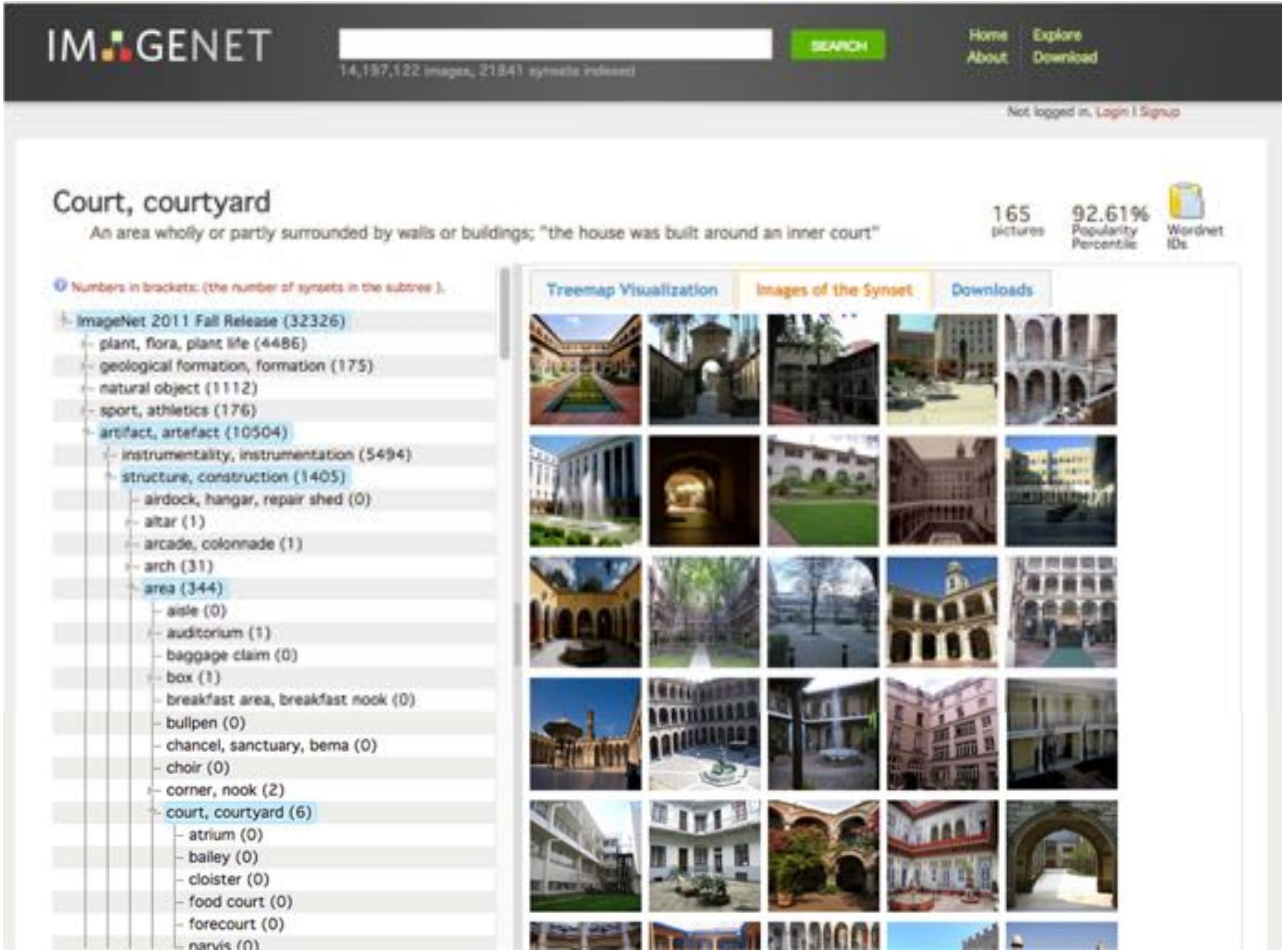
MOTIVATION:
LOGISTIC REGRESSION

Example: Image Classification

- ImageNet LSVRC-2010 contest:
  - **Dataset:** 1.2 million labeled images, 1000 classes
  - **Task:** Given a new image, label it with the correct class
  - **Multiclass** classification problem
- Examples from http://image-net.org/


IMAGENET
SEARCH
14,197,122 images, 21841 synsets indexed
Home
Explore
About
Download
Not logged in. Login | Signup
Bird
Warm-blooded egg-laying vertebrates characterized by feathers and forelimbs modified as wings
2126
pictures
92.85%
Popularity Percentile
Wordnet IDs
marine animal, marine creature, sea animal, sea creature (1)
scavenger (1)
biped (0)
predator, predatory animal (1)
larva (49)
acrodont (0)
feeder (0)
stunt (0)
chordate (3087)
tunicate, urochordate, urochord (6)
cephalochordate (1)
vertebrate, craniate (3077)
mammal, mammalian (1169)
bird (871)
dickeybird, dickey-bird, dickybird, dicky-bird (0)
cock (1)
hen (0)
nester (0)
night bird (1)
bird of passage (0)
protoavis (0)
archaeopteryx, archeopteryx, Archaeopteryx lithographi
Sinornis (0)
Ibero-mesornis (0)
archaeornis (0)
ratite, ratite bird, flightless bird (10)
carinate, carinate bird, flying bird (0)
passerine, passeriform bird (279)
nonpasserine bird (0)
bird of prey, raptor, raptorial bird (80)
gallinaceous bird, gallinacean (114)
Treemap Visualization
Images of the Synset
Downloads


IMAGENET
SEARCH
14,197,122 images, 21841 synsets indexed
Home
About
Explore
Download
Not logged in. Login | Signup
German iris, Iris kochii
Iris of northern Italy having deep blue-purple flowers; similar to but smaller than Iris germanica
469 pictures
49.6% Popularity Percentile
Wordnet IDs
halophyte (0)
succulent (39)
cultivar (0)
cultivated plant (0)
weed (54)
evergreen, evergreen plant (0)
deciduous plant (0)
vine (272)
creeper (0)
woody plant, ligneous plant (1868)
geophyte (0)
desert plant, xerophyte, xerophytic plant, xerophile, xerophilc
mesophyte, mesophytic plant (0)
aquatic plant, water plant, hydrophyte, hydrophytic plant (11
tuberous plant (0)
bulbous plant (179)
iridaceous plant (27)
iris, flag, fleur-de-lis, sword lily (19)
bearded iris (4)
Florentine iris, orris, Iris germanica florentina, Iris
German iris, Iris germanica (0)
German iris, Iris kochii (0)
Dalmatian iris, Iris pallida (0)
beardless iris (4)
bulbous iris (0)
dwarf iris, Iris cristata (0)
stinking iris, gladdon, gladdon iris, stinking gladwyn,
Persian iris, Iris persica (0)
yellow iris, yellow flag, yellow water flag, Iris pseuda
dwarf iris, vernal iris, Iris verna (0)
blue flag, Iris versicolor (0)
Treemap Visualization
Images of the Synset
Downloads


IMAGENET
SEARCH
14,197,122 images, 21841 synsets indexed
Home
About
Explore
Download
Not logged in. Login | Signup
Court, courtyard
An area wholly or partly surrounded by walls or buildings; "the house was built around an inner court"
165 pictures
92.61% Popularity Percentile
Wordnet IDs
Numbers in brackets: (the number of synsets in the subtree).
ImageNet 2011 Fall Release (32326)
plant, flora, plant life (4486)
geological formation, formation (175)
natural object (1112)
sport, athletics (176)
artifact, artefact (10504)
instrumentality, instrumentation (5494)
structure, construction (1405)
airdock, hangar, repair shed (0)
altar (1)
arcade, colonnade (1)
arch (31)
area (344)
aisle (0)
auditorium (1)
baggage claim (0)
box (1)
breakfast area, breakfast nook (0)
bullpen (0)
chancel, sanctuary, bema (0)
choir (0)
corner, nook (2)
court, courtyard (6)
atrium (0)
bailey (0)
cloister (0)
food court (0)
forecourt (0)
parvis (0)
Treemap Visualization
Images of the Synset
Downloads

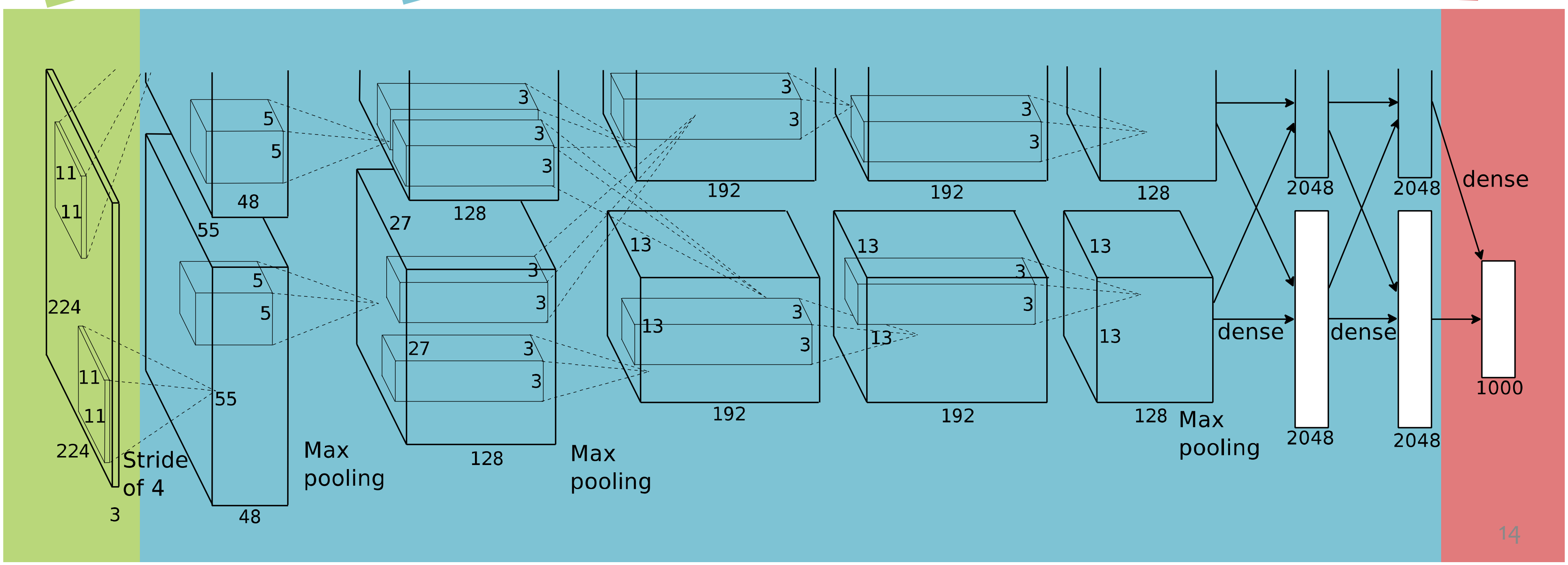
Example: Image Classification

CNN for Image Classification
(Krizhevsky, Sutskever & Hinton, 2011)
17.5% error on ImageNet LSVRC-2010 contest

Input image (pixels)

- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax



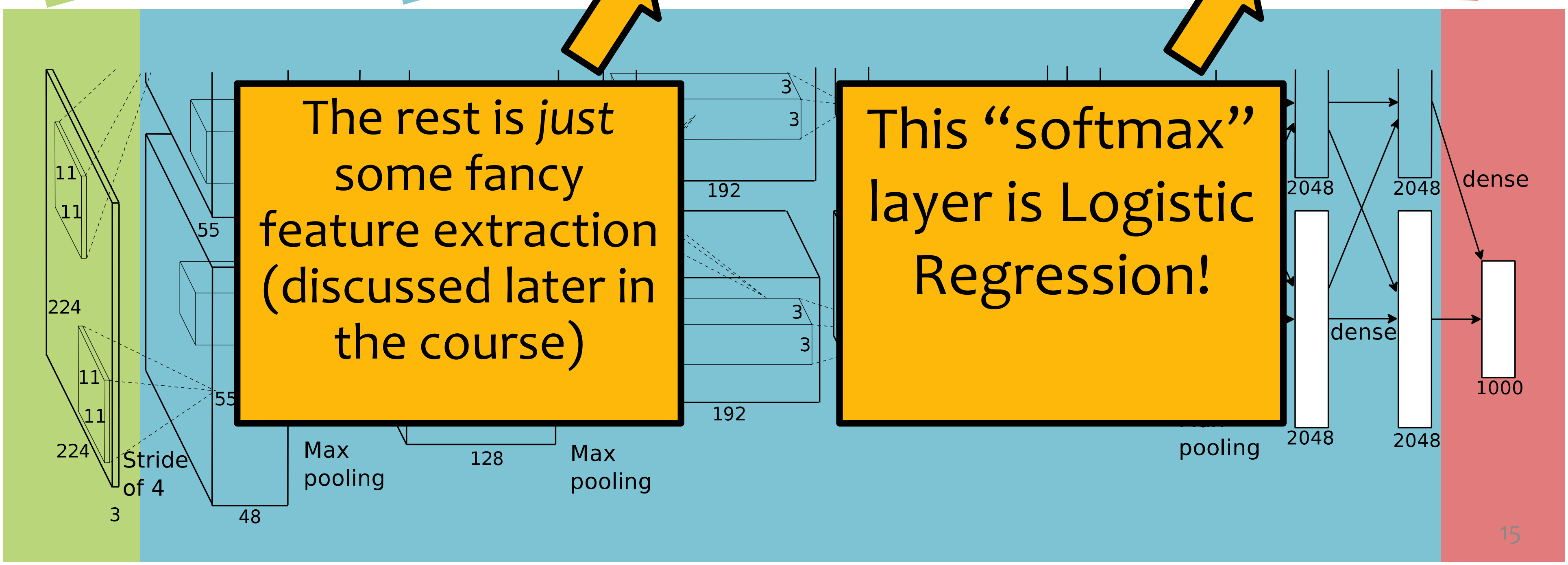
Example: Image Classification

CNN for Image Classification
(Krizhevsky, Sutskever & Hinton, 2011)
17.5% error on ImageNet LSVRC-2010 contest

Input image (pixels)

- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax



The rest is *just* some fancy feature extraction (discussed later in the course)

This "softmax" layer is Logistic Regression!

LOGISTIC REGRESSION

Logistic Regression

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

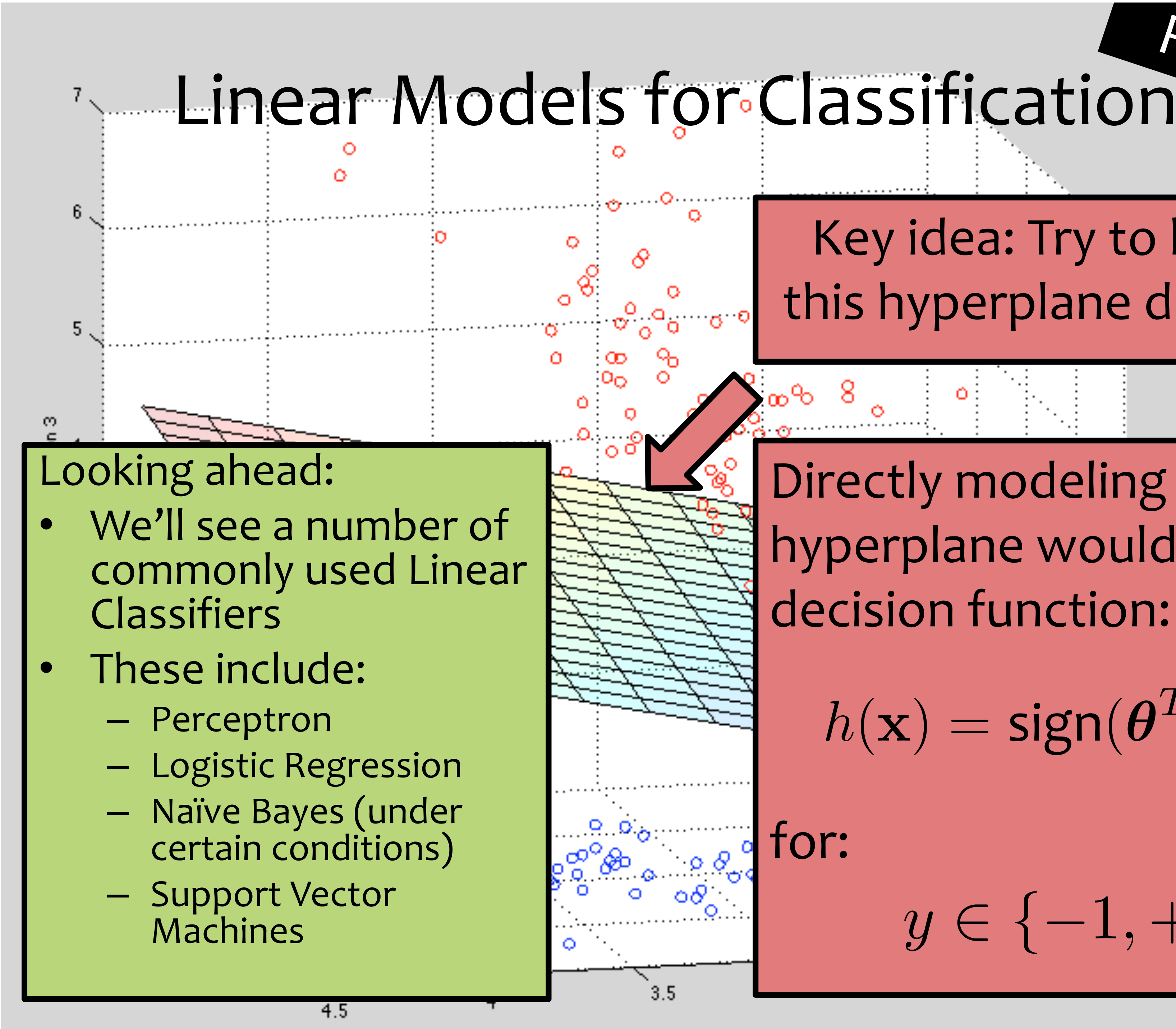
$$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N} \text{ where } \mathbf{x} \in \mathbb{R}^M \text{ and } y \in \{0, 1\}$$

We are back to classification.

Despite the name logistic **regression.**

Linear Models for Classification

Recall...



Key idea: Try to learn this hyperplane directly

Looking ahead:

- We'll see a number of commonly used Linear Classifiers
- These include:
  - Perceptron
  - Logistic Regression
  - Naïve Bayes (under certain conditions)
  - Support Vector Machines

Directly modeling the hyperplane would use a decision function:

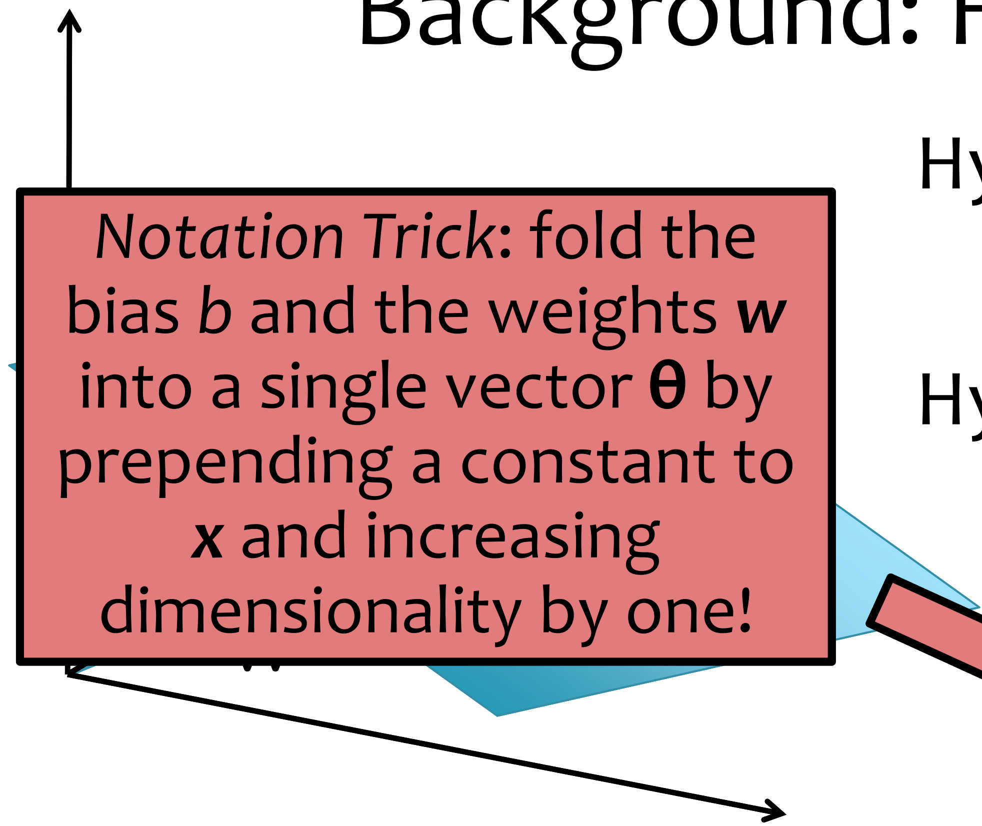
$$h(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$$

for:

$$y \in \{-1, +1\}$$

Recall...

Background: Hyperplanes



Notation Trick: fold the bias *b* and the weights ***w*** into a single vector **θ** by prepending a constant to ***x*** and increasing dimensionality by one!

Hyperplane (Definition 1):

$$\mathcal{H} = \{\mathbf{x} : \mathbf{w}^T \mathbf{x} = b\}$$

Hyperplane (Definition 2):

$$\mathcal{H} = \{\mathbf{x} : \boldsymbol{\theta}^T \mathbf{x} = 0 \text{ and } x_0 = 1\}$$

$$\boldsymbol{\theta} = [b, w_1, \ldots, w_M]^T$$

Half-spaces:

$$\mathcal{H}^+ = \{\mathbf{x} : \boldsymbol{\theta}^T \mathbf{x} > 0 \text{ and } x_0 = 1\}$$

$$\mathcal{H}^- = \{\mathbf{x} : \boldsymbol{\theta}^T \mathbf{x} < 0 \text{ and } x_0 = 1\}$$

Using gradient ascent for linear classifiers

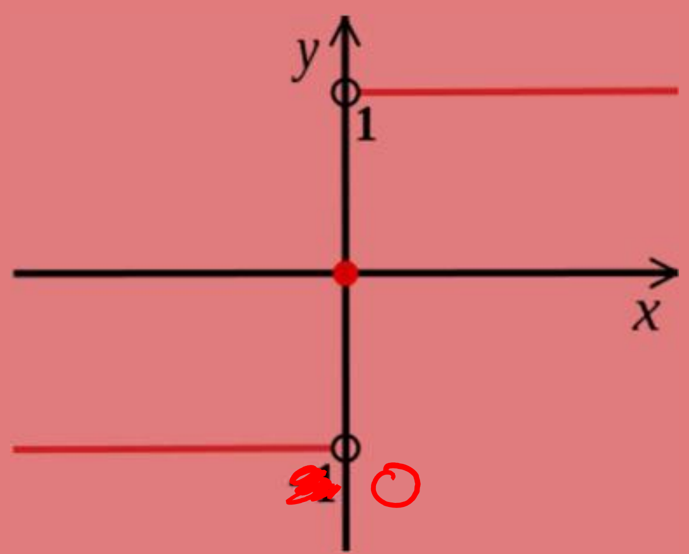
Key idea behind today's lecture:

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 linear classifiers

This decision function isn't differentiable:

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



sign(x)

Use a differentiable function instead:

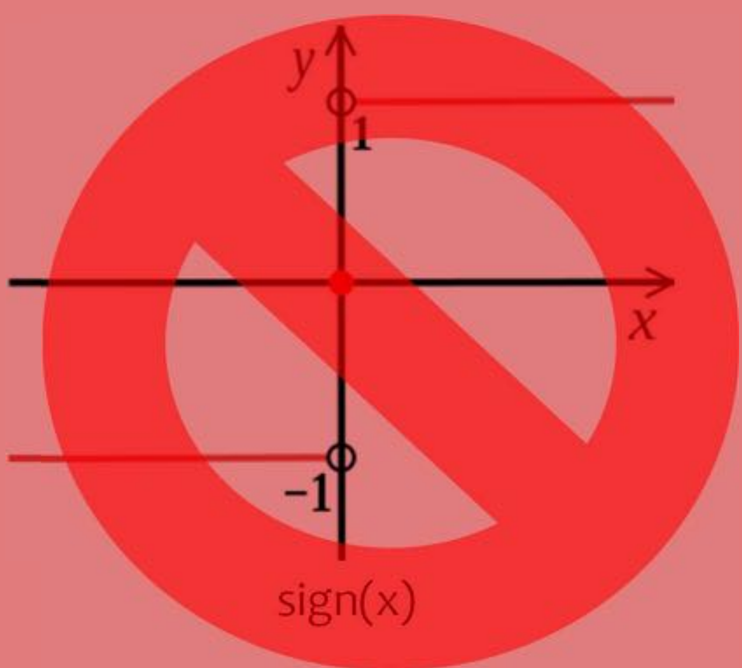
$$p_{\boldsymbol{\theta}}(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \mathbf{x})}$$

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

Using gradient ascent for linear classifiers

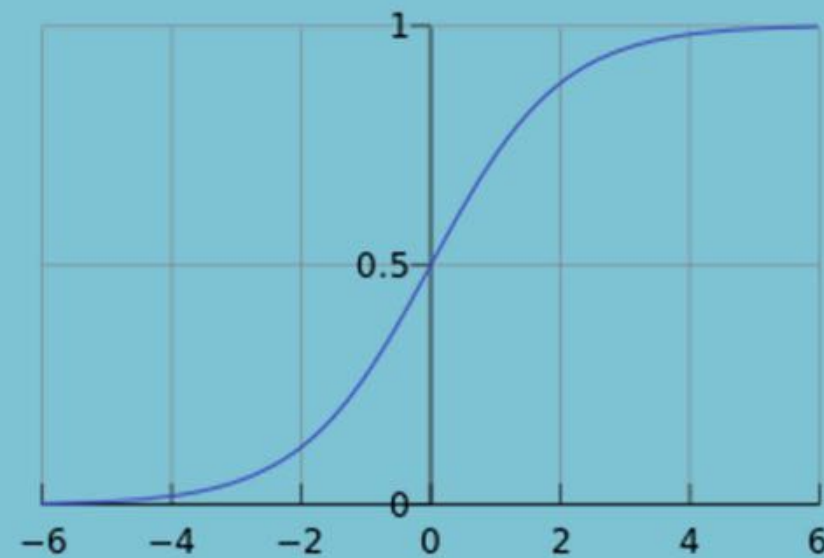
This decision function isn't differentiable:

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



Use a differentiable function instead:

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



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

Logistic Regression

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

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

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. $\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmin}}\, J(\boldsymbol{\theta})$

Prediction: Output is the most probable class.

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

Logistic Regression

Whiteboard

- Bernoulli interpretation
- Logistic Regression Model
- Decision boundary

Learning for Logistic Regression

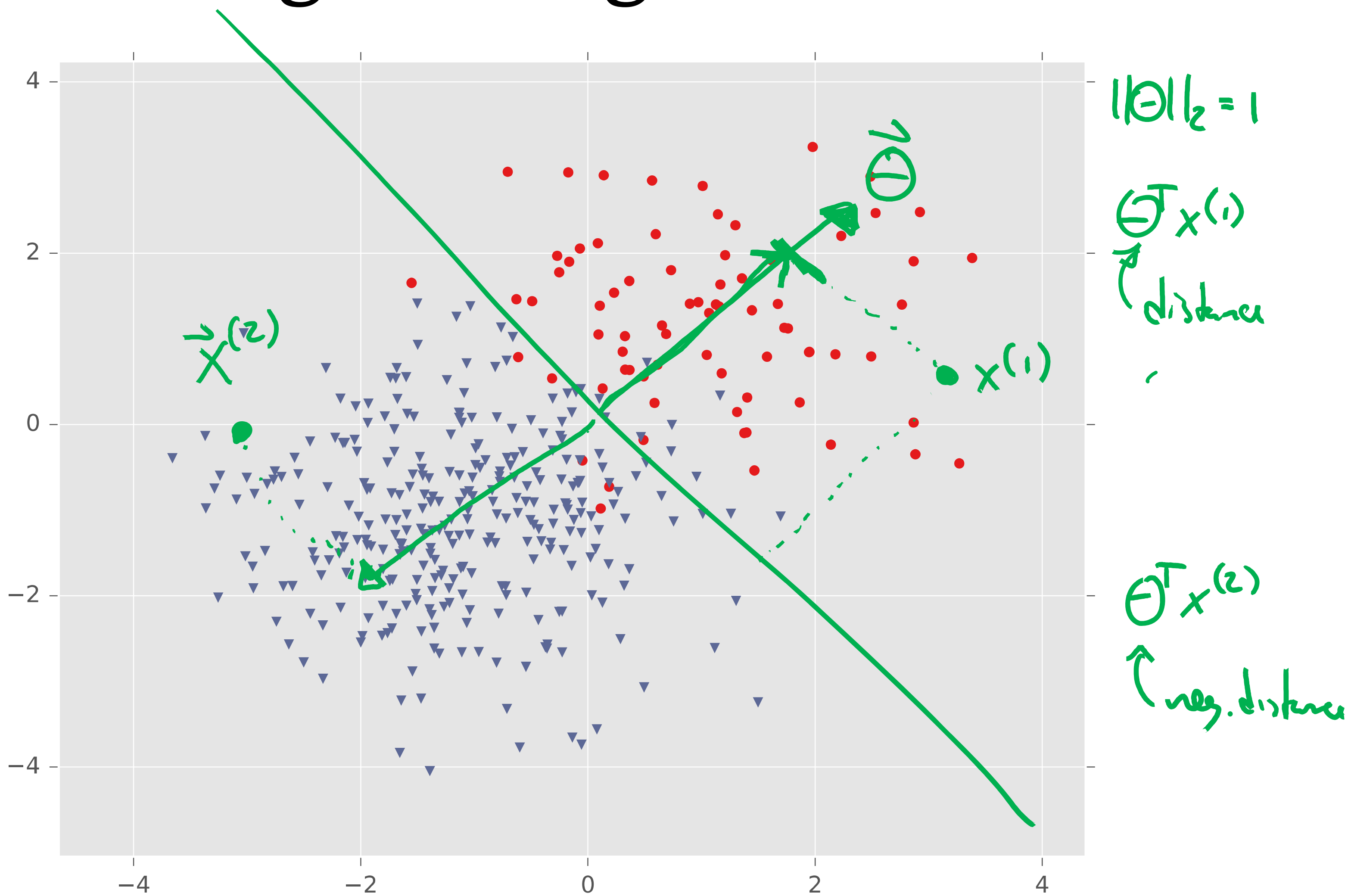
Whiteboard

– Partial derivative for Logistic Regression

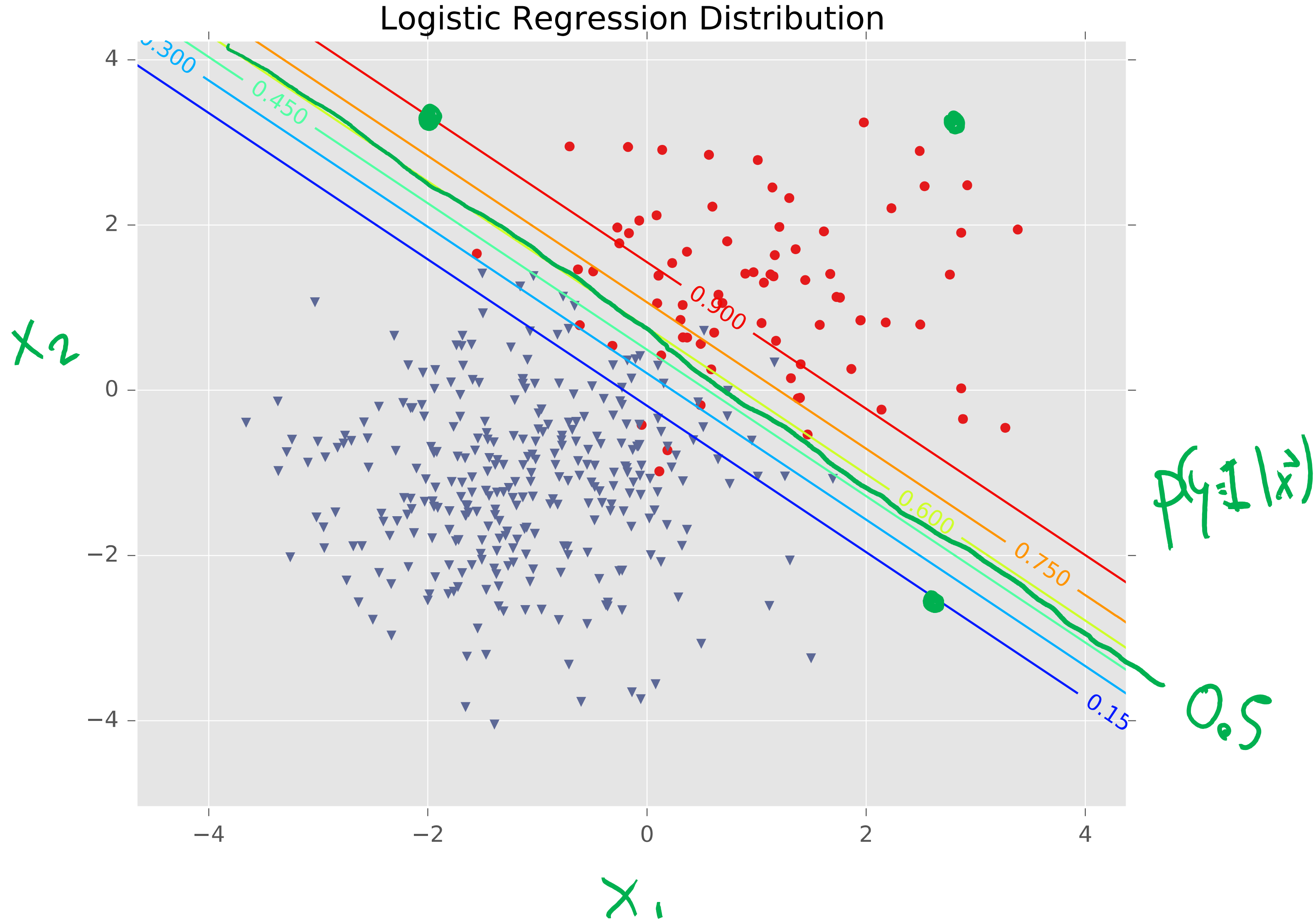
– Gradient for Logistic Regression

LOGISTIC REGRESSION ON GAUSSIAN DATA

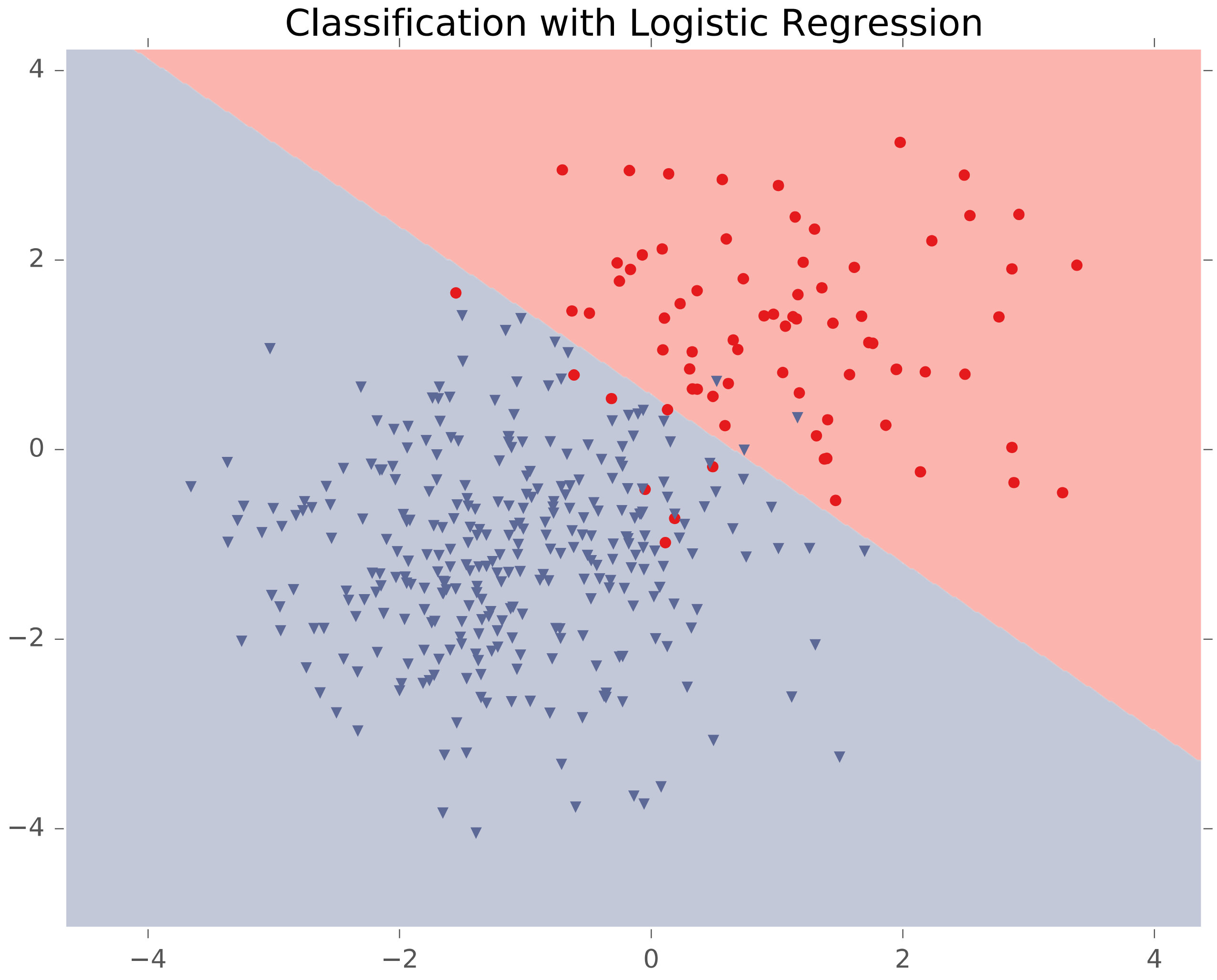
Logistic Regression



Logistic Regression



Logistic Regression



LEARNING LOGISTIC REGRESSION

Maximum **Conditional** Likelihood Estimation

Learning: finds the parameters that minimize some objective function.

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

We minimize the *negative* log conditional likelihood:

$$J(\boldsymbol{\theta}) = -\log \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(y^{(i)} | \mathbf{x}^{(i)})$$

Why?

1. We can't maximize likelihood (as in Naïve Bayes) because we don't have a joint model p(x,y)
2. It worked well for Linear Regression (least squares is MCLE)

Maximum **Conditional** Likelihood Estimation

Learning: Four approaches to solving $\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmin}}\, J(\boldsymbol{\theta})$

Approach 1: Gradient Descent
(take larger – more certain – steps opposite the gradient)

Approach 2: Stochastic Gradient Descent (SGD)
(take many small steps opposite the gradient)

Approach 3: Newton's Method
(use second derivatives to better follow curvature)

Approach 4: Closed Form???
(set derivatives equal to zero and solve for parameters)

Maximum **Conditional** Likelihood Estimation

Learning: Four approaches to solving $\theta^* = \operatorname{argmin}_{\theta} J(\theta)$

Approach 1: Gradient Descent
(take larger – more certain – steps opposite the gradient)

Approach 2: Stochastic Gradient Descent (SGD)
(take many small steps opposite the gradient)

Approach 3: Newton's Method
(use second derivatives to better follow curvature)

Approach 4: Closed Form???
(set derivatives equal to zero and solve for parameters)

Logistic Regression does not have a closed form solution for MLE parameters.

SGD for Logistic Regression

Question:

Which of the following is a correct description of SGD for Logistic Regression?

Answer:

At each step (i.e. iteration) of SGD for Logistic Regression we…

5 A. (1) compute the gradient of the log-likelihood for all examples (2) update all the parameters using the gradient

calamity B. ~~(1) ask Matt for a description of SGD for Logistic Regression, (2) write it down, (3) report that answer~~

6 C. (1) compute the gradient of the log-likelihood for all examples (2) randomly pick an example (3) update only the parameters for that example

18 D. (1) randomly pick a parameter, (2) compute the partial derivative of the log-likelihood with respect to that parameter, (3) update that parameter for all examples

56 E. (1) randomly pick an example, (2) compute the gradient of the log-likelihood for that example, (3) update all the parameters using that gradient

16 F. (1) randomly pick a parameter and an example, (2) compute the gradient of the log-likelihood for that example with respect to that parameter, (3) update that parameter using that gradient

Recall...

Gradient Descent

Algorithm 1 Gradient Descent

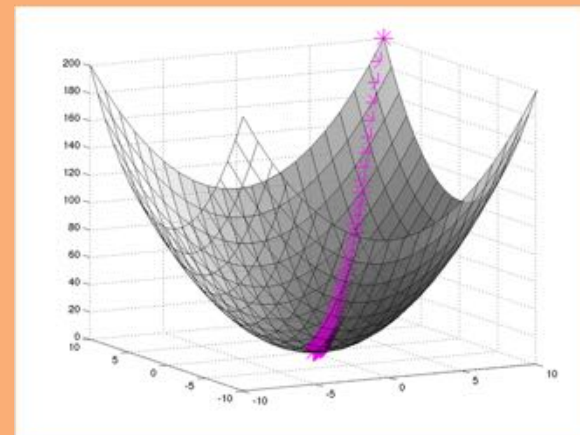
1: **procedure** $\text{GD}(\mathcal{D}, \boldsymbol{\theta}^{(0)})$

2: $\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$

3: $\quad$ **while** not converged **do**

4: $\quad\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

5: $\quad$ **return** $\boldsymbol{\theta}$



In order to apply GD to Logistic Regression all we need is the **gradient** of the objective function (i.e. vector of partial derivatives).

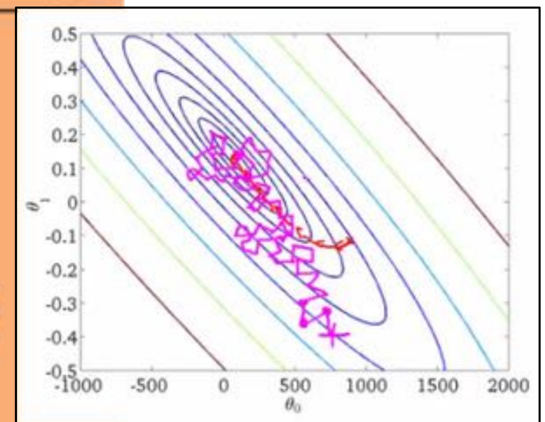
$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_M} J(\boldsymbol{\theta}) \end{bmatrix}$$

Stochastic Gradient Descent (SGD)

Recall…

Algorithm 1 Stochastic Gradient Descent (SGD)

procedure SGD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
    while not converged **do**
        for $i \in$ shuffle($\{1, 2, \ldots, N\}$) **do**
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$
    return $\boldsymbol{\theta}$



We can also apply SGD to solve the MCLE problem for Logistic Regression.

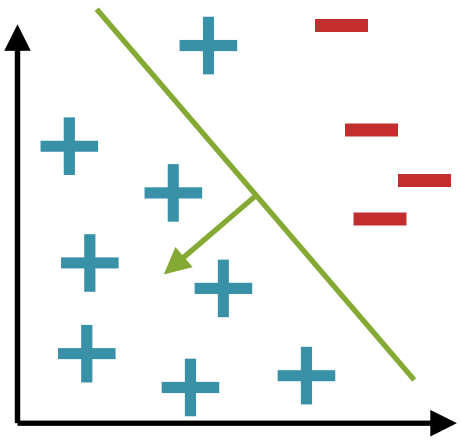
We need a per-example objective:

$$\text{Let } J(\boldsymbol{\theta}) = \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$
$$\text{where } J^{(i)}(\boldsymbol{\theta}) = -\log p_{\boldsymbol{\theta}}(y^i | \mathbf{x}^i).$$

Logistic Regression vs. Perceptron

Question:

True or False: Just like Perceptron**, one step** (i.e. iteration) **of SGD for Logistic Regression** will result in a change to the parameters **only** if the current example is **incorrectly** classified.



Answer:

A = calamity

B = T

C = F 90%

Matching Game

Goal: Match the Algorithm to its Update Rule

1. SGD for Logistic Regression $h_{\boldsymbol{\theta}}(\mathbf{x}) = p(y\|x)$	**4.** $\theta_k \leftarrow \theta_k + (h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)})$
2. Least Mean Squares $h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x}$	**5.** $\theta_k \leftarrow \theta_k + \frac{1}{1 + \exp \lambda(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)})}$
3. Perceptron $h_{\boldsymbol{\theta}}(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$	**6.** $\theta_k \leftarrow \theta_k + \lambda(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)})x_k^{(i)}$

A. 1=5, 2=4, 3=6
B. 1=5, 2=6, 3=4
C. 1=6, 2=4, 3=4
D. 1=5, 2=6, 3=6
E. 1=6, 2=6, 3=6
F. 1=6, 2=5, 3=5
G. 1=5, 2=5, 3=5
H. 1=4, 2=5, 3=6

OPTIMIZATION METHOD #4: MINI-BATCH SGD

Mini-Batch SGD

- **Gradient Descent:**
  Compute true gradient exactly from all N examples
- **Stochastic Gradient Descent (SGD):**
  Approximate true gradient by the gradient of one randomly chosen example
- **Mini-Batch SGD:**
  Approximate true gradient by the average gradient of ~~K~~ S randomly chosen examples

Mini-Batch SGD

while not converged: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \mathbf{g}$

Three variants of first-order optimization:

Gradient Descent: $\mathbf{g} = \nabla J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \nabla J^{(i)}(\boldsymbol{\theta})$

SGD: $\mathbf{g} = \nabla J^{(i)}(\boldsymbol{\theta})$ where i sampled uniformly

Mini-batch SGD: $\mathbf{g} = \frac{1}{S} \sum_{s=1}^{S} \nabla J^{(i_s)}(\boldsymbol{\theta})$ where i_s sampled uniformly $\forall s$

samples → N

Summary

1. Discriminative classifiers directly model the **conditional**, p(y|x)
2. Logistic regression is a **simple linear classifier**, that retains a **probabilistic** semantics
3. Parameters in LR are learned by **iterative optimization** (e.g. SGD)

Logistic Regression Objectives

You should be able to…

- Apply the principle of maximum likelihood estimation (MLE) to learn the parameters of a probabilistic model
- Given a discriminative probabilistic model, derive the conditional log-likelihood, its gradient, and the corresponding Bayes Classifier
- Explain the practical reasons why we work with the **log** of the likelihood
- Implement logistic regression for binary or multiclass classification
- Prove that the decision boundary of binary logistic regression is linear
- For linear regression, show that the parameters which minimize squared error are equivalent to those that maximize conditional likelihood

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:

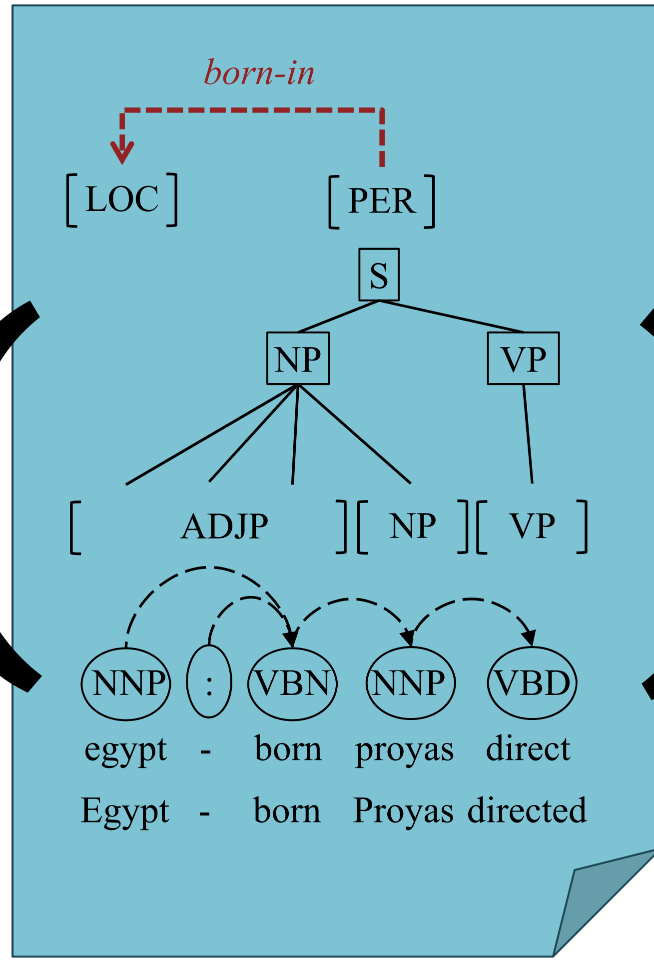
```
while not converged:
  for i in shuffle([1,…,N]):
    for k in [1,…,K]:
      theta[k] = theta[k] - lambda * grad(x[i], y[i], theta, k)
```

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

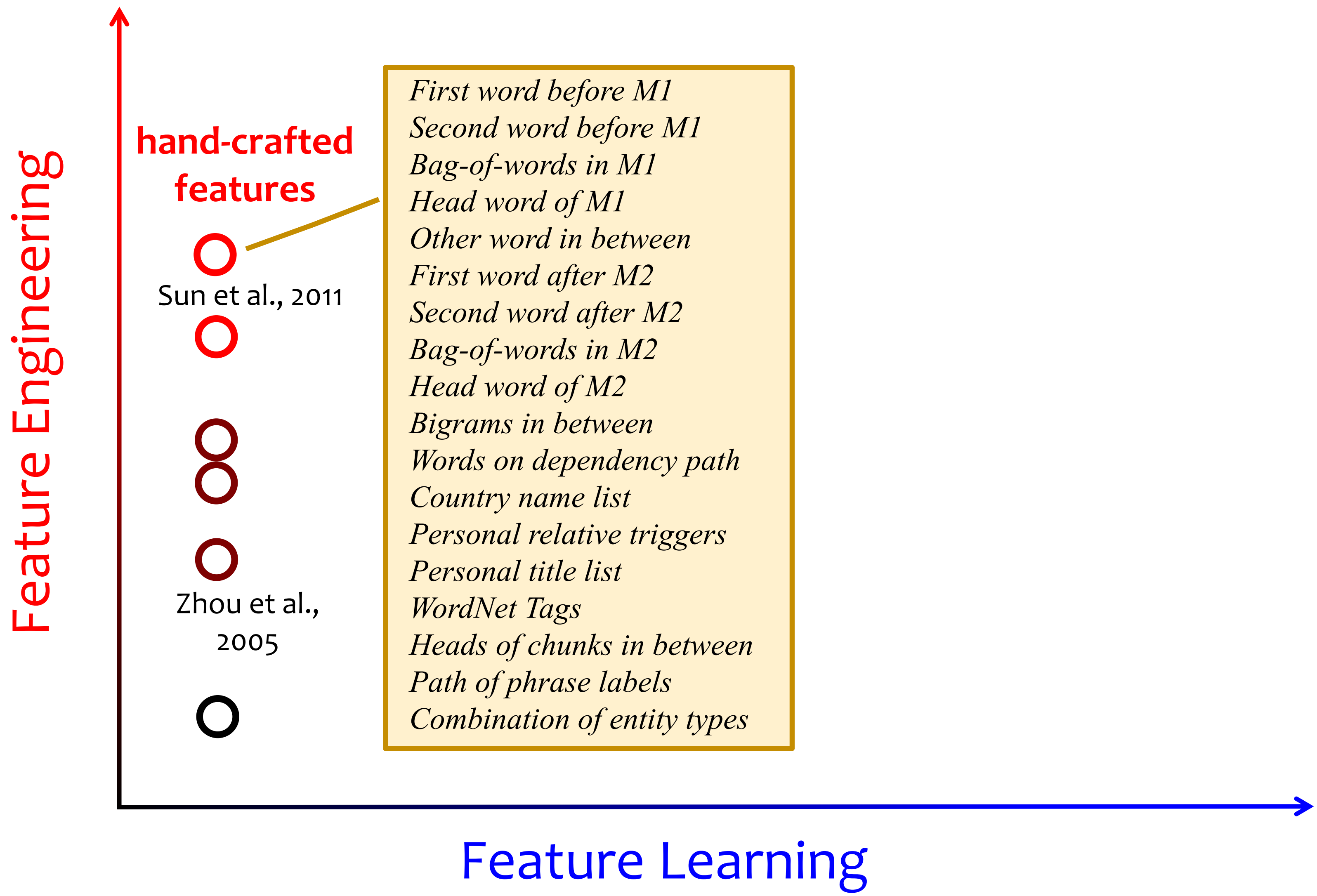
Handcrafted Features

$p(y|x) \propto \exp(\Theta_y \cdot f($

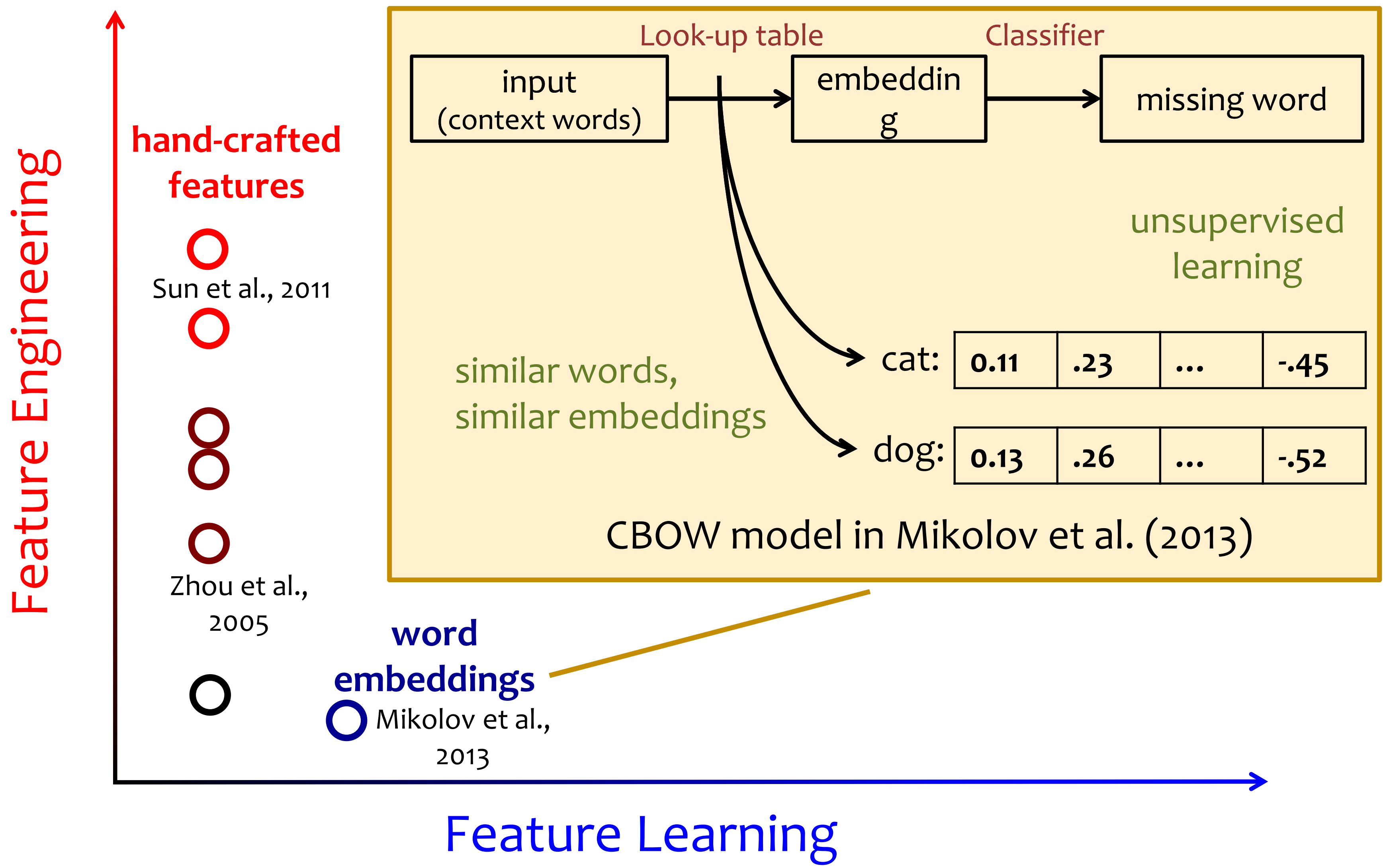


$))$

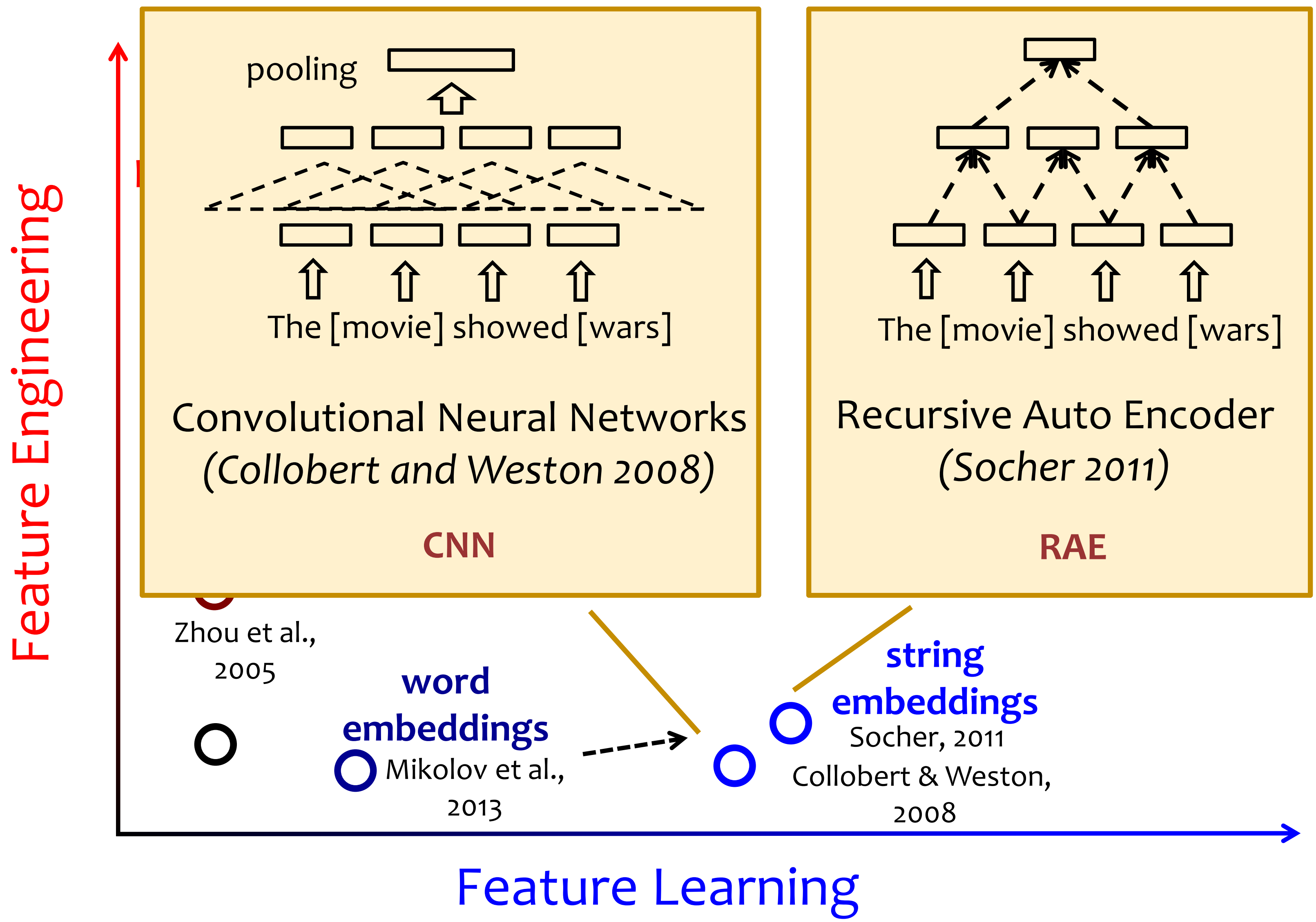
Where do features come from?



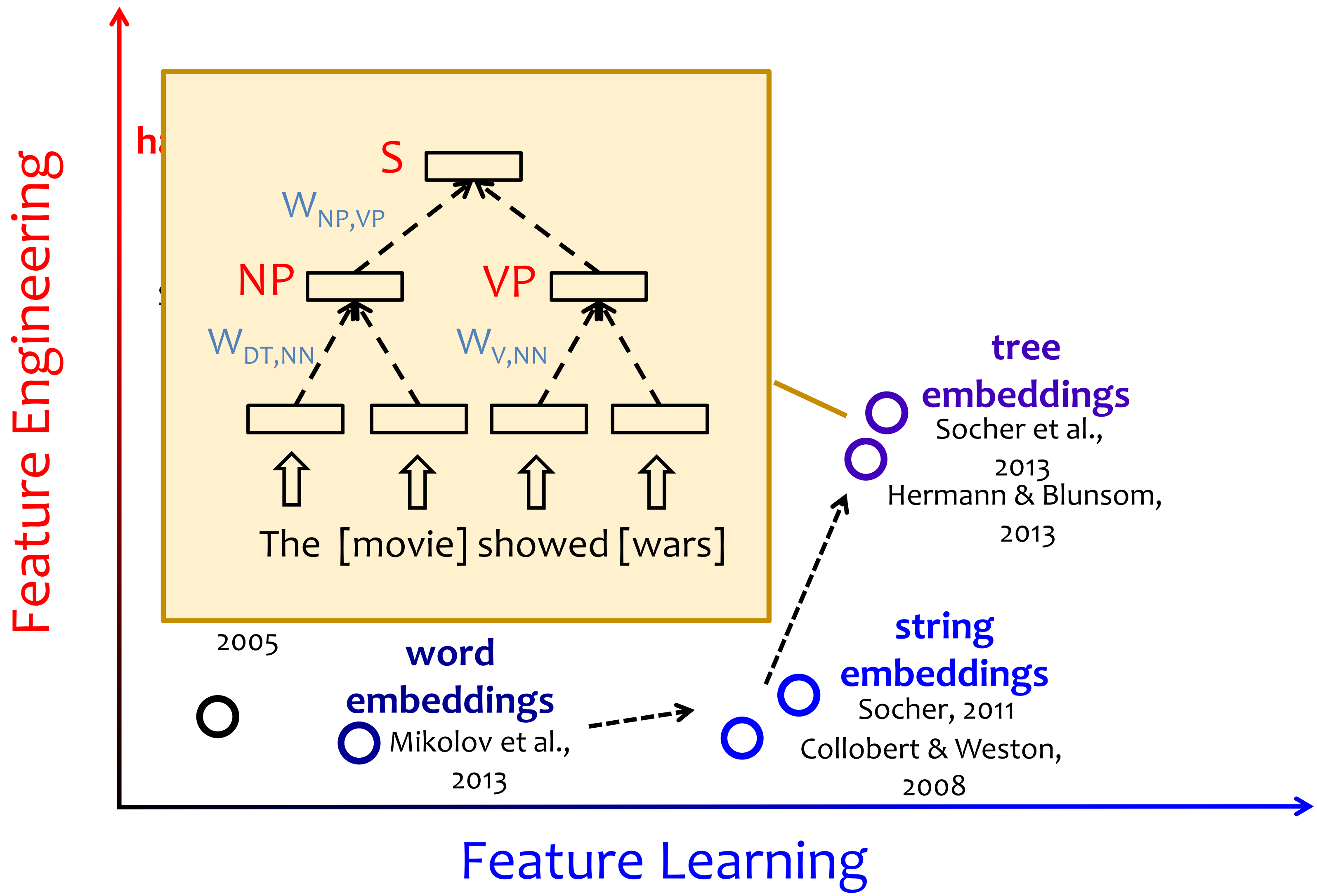
Where do features come from?



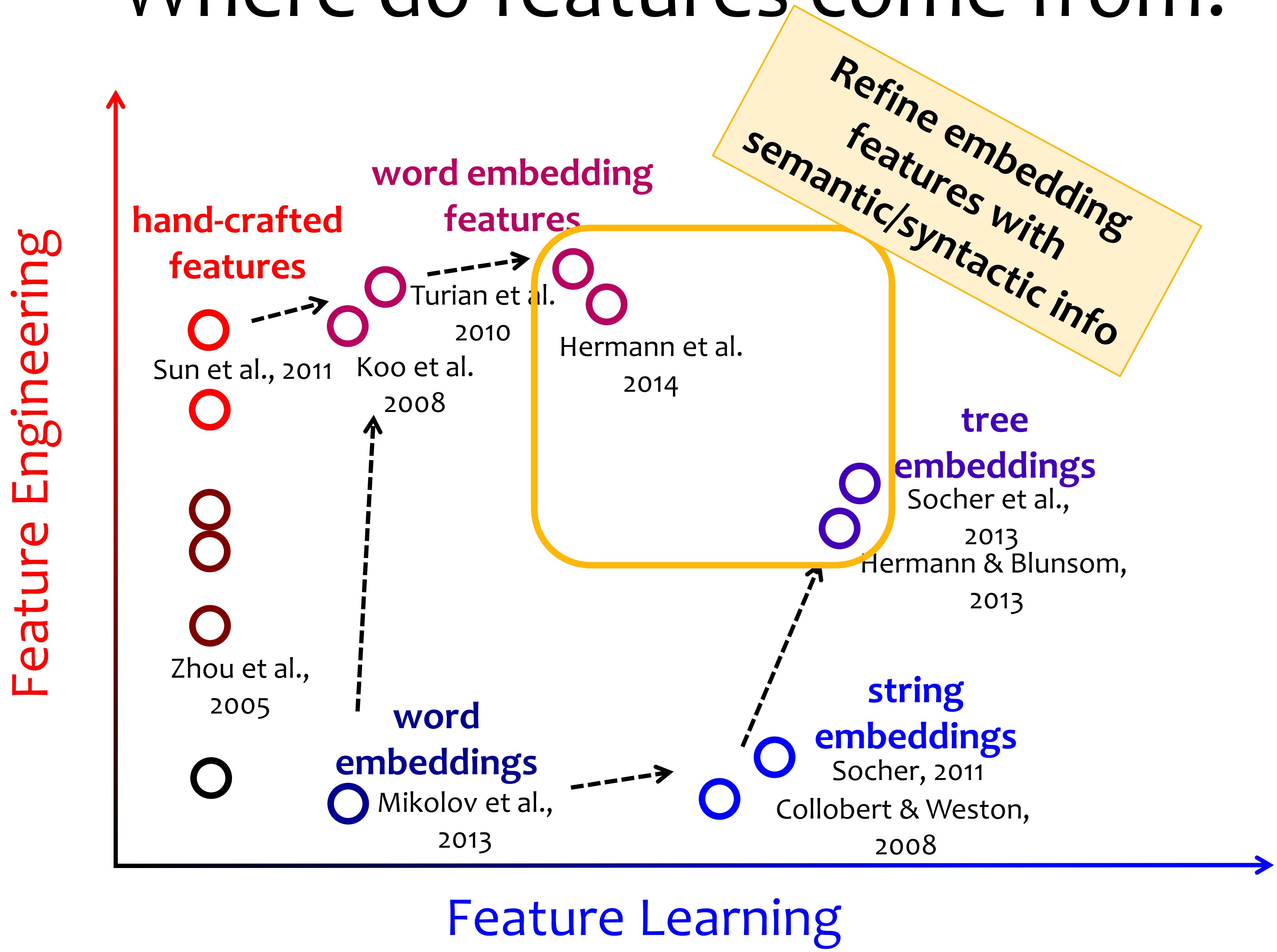
Where do features come from?



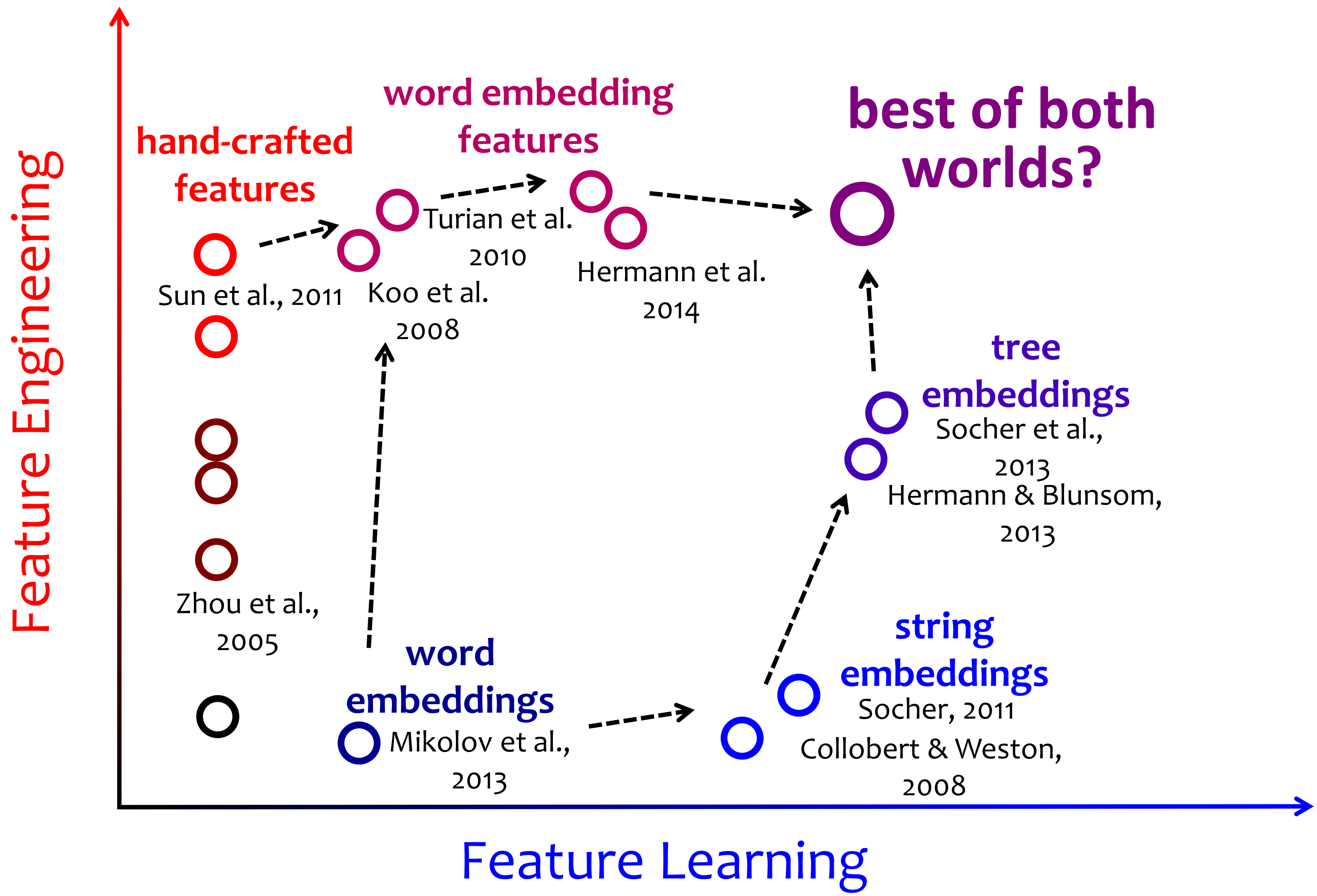
Where do features come from?



Where do features come from?



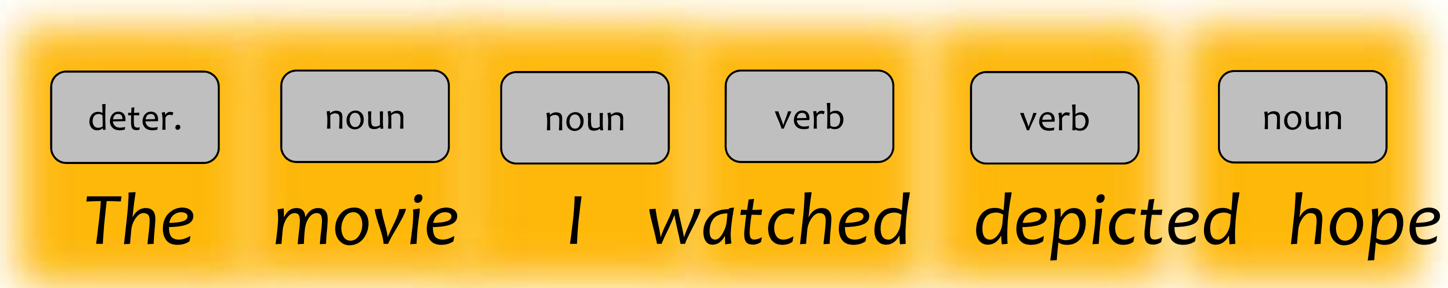
Where do features come from?



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?



Feature Engineering for NLP

Per-word Features:

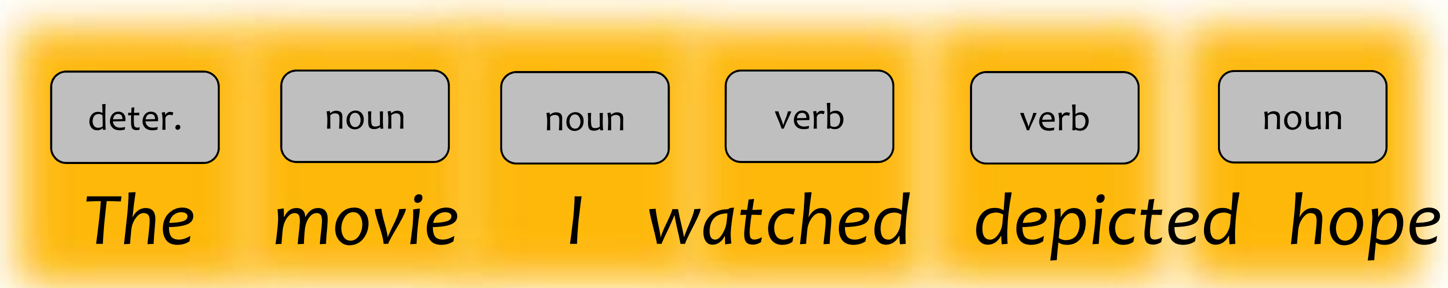
	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$
`is-capital(w_i)`	1	0	1	0	0	0
`endswith(w_i,"e")`	1	1	0	0	0	1
`endswith(w_i,"d")`	0	0	0	1	1	0
`endswith(w_i,"ed")`	0	0	0	1	1	0
`w_i == "aardvark"`	0	0	0	0	0	0
`w_i == "hope"`	0	0	0	0	0	1
…	…	…	…	…	…	…

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 == "watched"	0	0	0	1	0	0
w_{i+1} == "watched"	0	0	1	0	0	0
w_{i-1} == "watched"	0	0	0	0	1	0
w_{i+2} == "watched"	0	1	0	0	0	0
w_{i-2} == "watched"	0	0	0	0	0	1
…	…	…	…	…	…	…



Feature Engineering for NLP

Context Features:

	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$
…	…	…	…	…	…	…
`w_i == "I"`	0	0	1	0	0	0
`w_i+1 == "I"`	0	1	0	0	0	0
`w_i-1 == "I"`	0	0	0	1	0	0
`w_i+2 == "I"`	1	0	0	0	0	0
`w_i-2 == "I"`	0	0	0	0	1	0
…	…	…	…	…	…	…

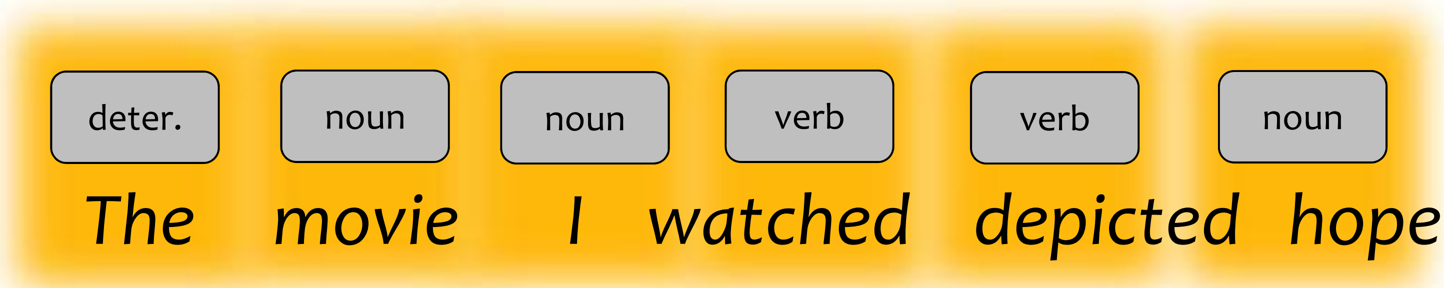
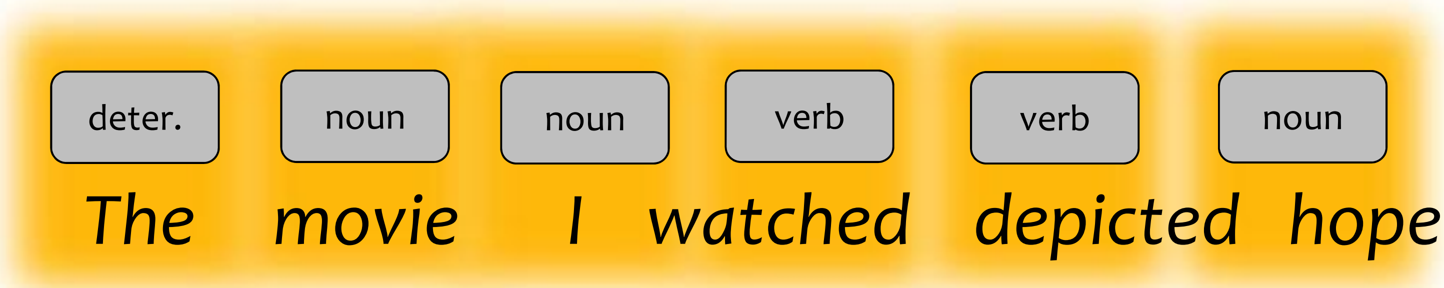


Table from Manning (2011)

Feature Engineering for NLP

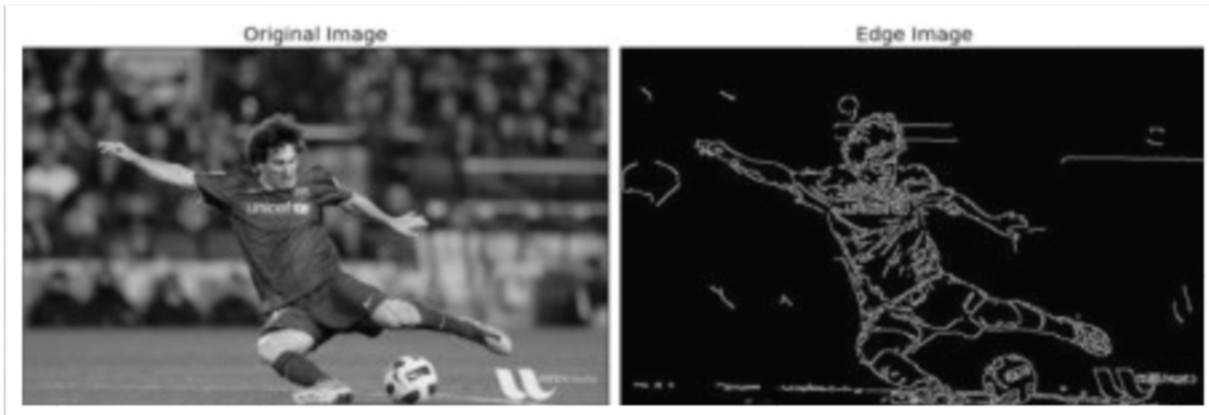
Table 3. Tagging accuracies with different feature templates and other changes on the WSJ 19-21 development set.

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%

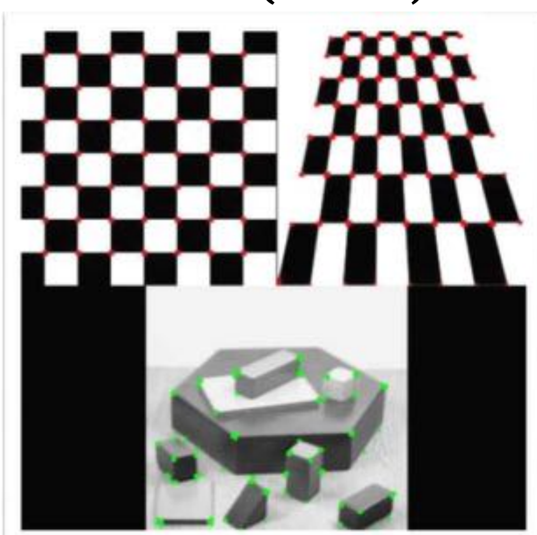


Feature Engineering for CV

Edge detection (Canny)



Corner Detection (Harris)



Figures from http://opencv.org

Feature Engineering for CV

Scale Invariant Feature Transform (SIFT)

Figure 3: Model images of planar objects are shown in the top row. Recognition results below show model outlines and image keys used for matching.

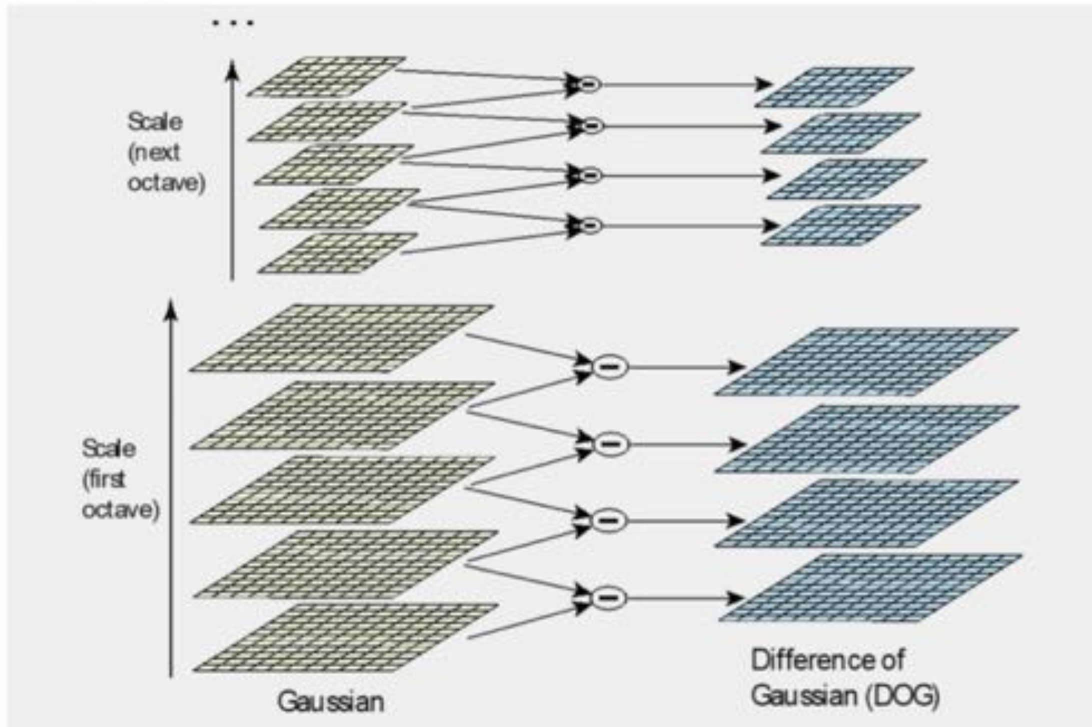


Figure 1: For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated.

Figure from Lowe (1999) and Lowe (2004)

NON-LINEAR FEATURES

Nonlinear Features

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

    where $b_i : \mathbb{R}^M \rightarrow \mathbb{R}$ is any function

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

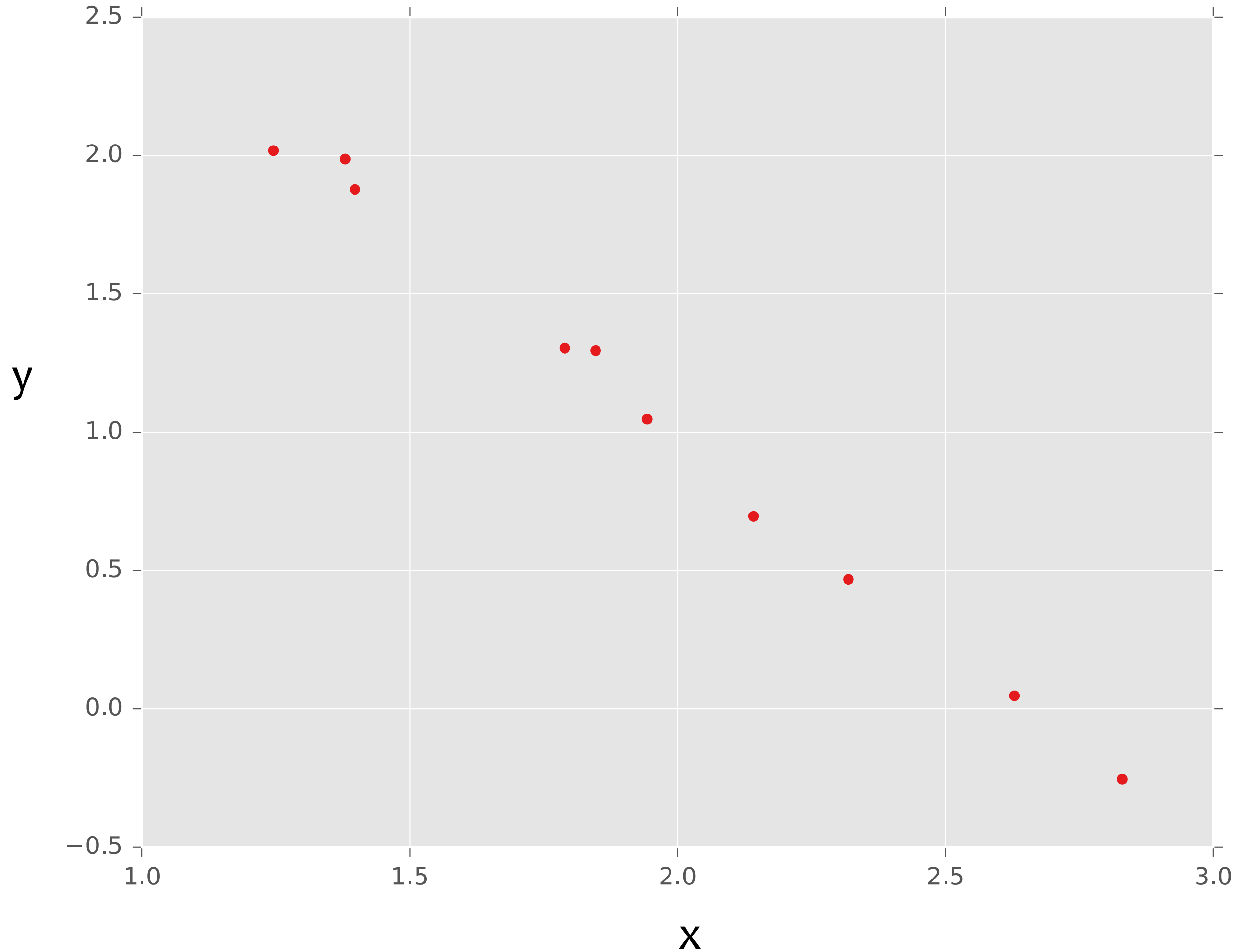
polynomial	$b_j(x) = x^j \quad \forall j \in \{1, \ldots, J\}$
radial basis function	$b_j(x) = \exp\left(\frac{-(x - \mu_j)^2}{2\sigma_j^2}\right)$
sigmoid	$b_j(x) = \frac{1}{1 + \exp(-\omega_j x)}$
log	$b_j(x) = \log(x)$

For a linear model: still a linear function of b(**x**) even though a nonlinear function of **x**

Examples:
- Perceptron
- Linear regression
- Logistic regression

Example: Linear Regression

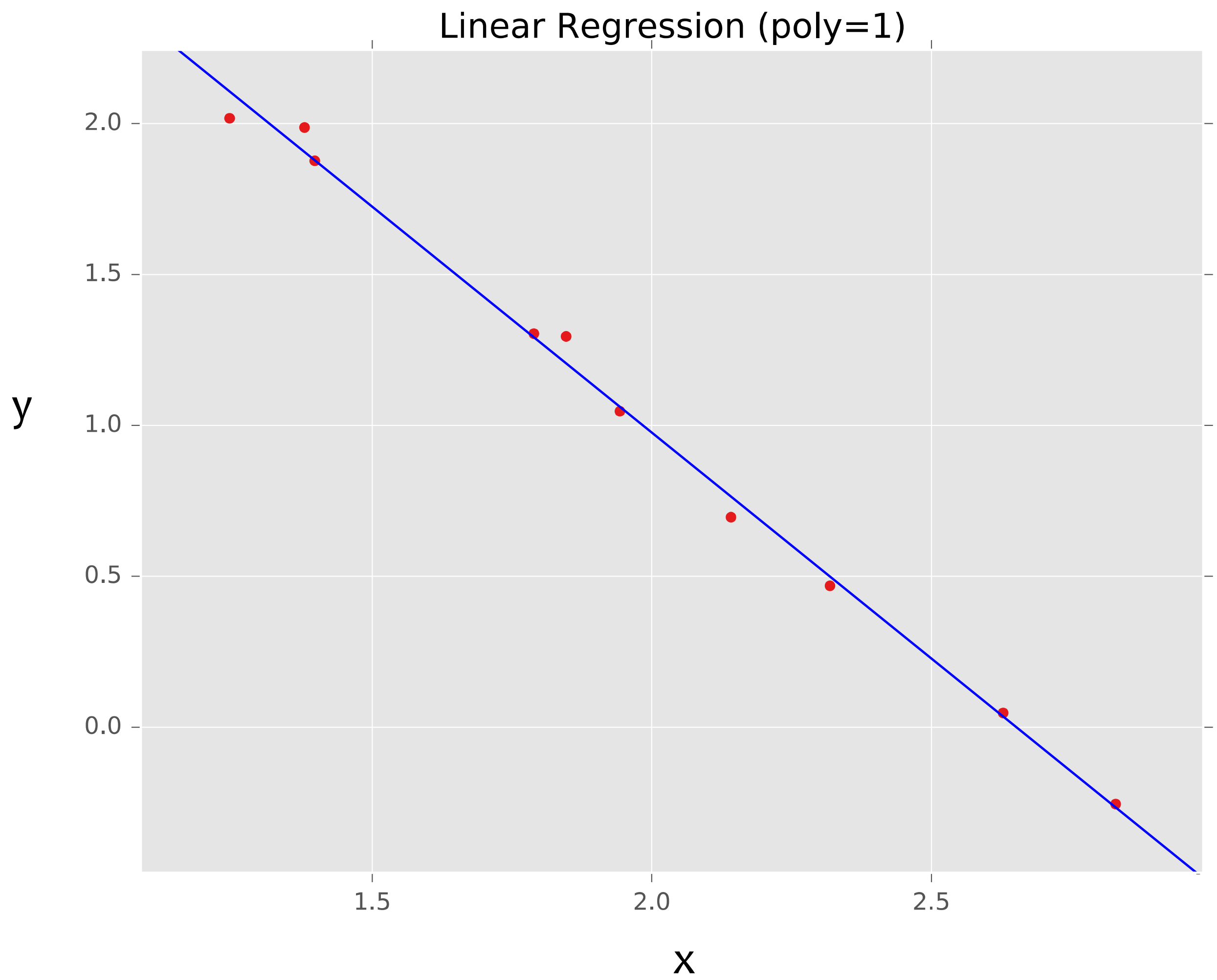
Goal: Learn y = $\mathbf{w}^T f(\mathbf{x}) + b$
where f(.) is a polynomial
basis function



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

Example: Linear Regression

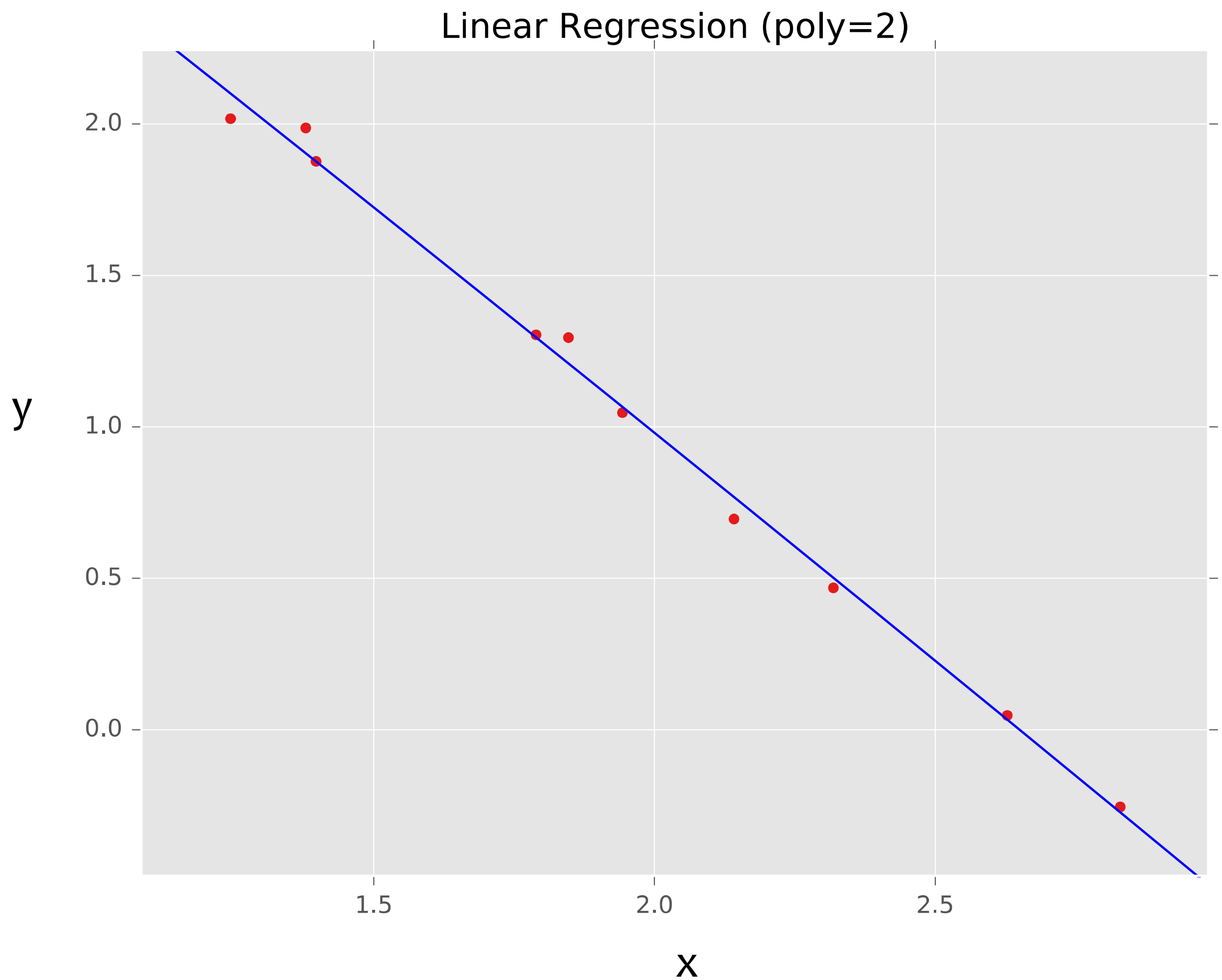
Goal: Learn y = $\mathbf{w}^T f(\mathbf{x}) + b$ where f(.) is a polynomial basis function



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

Example: Linear Regression

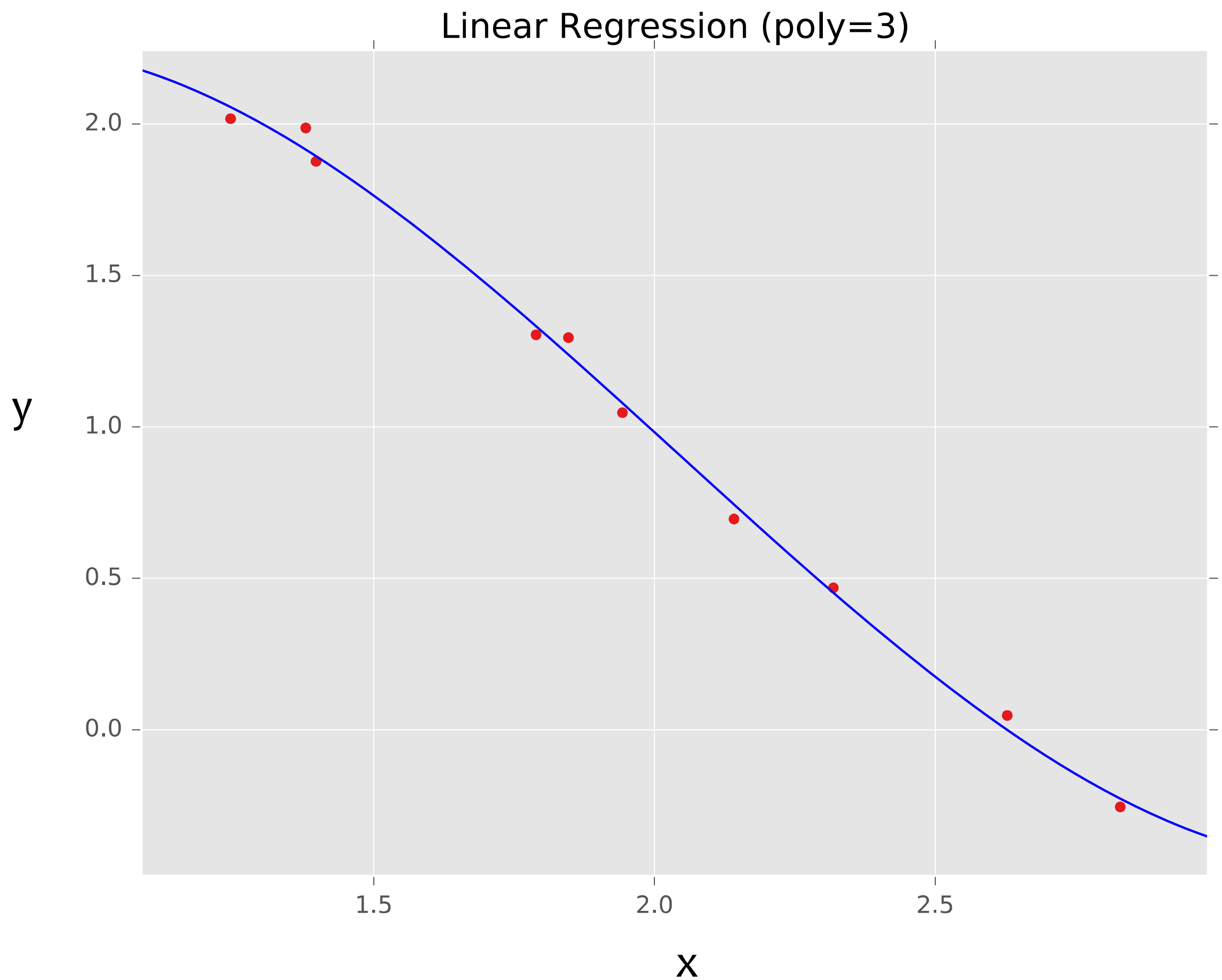
Goal: Learn y = $\mathbf{w}^T$ f($\mathbf{x}$) + b
where f(.) is a polynomial basis function



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

Example: Linear Regression

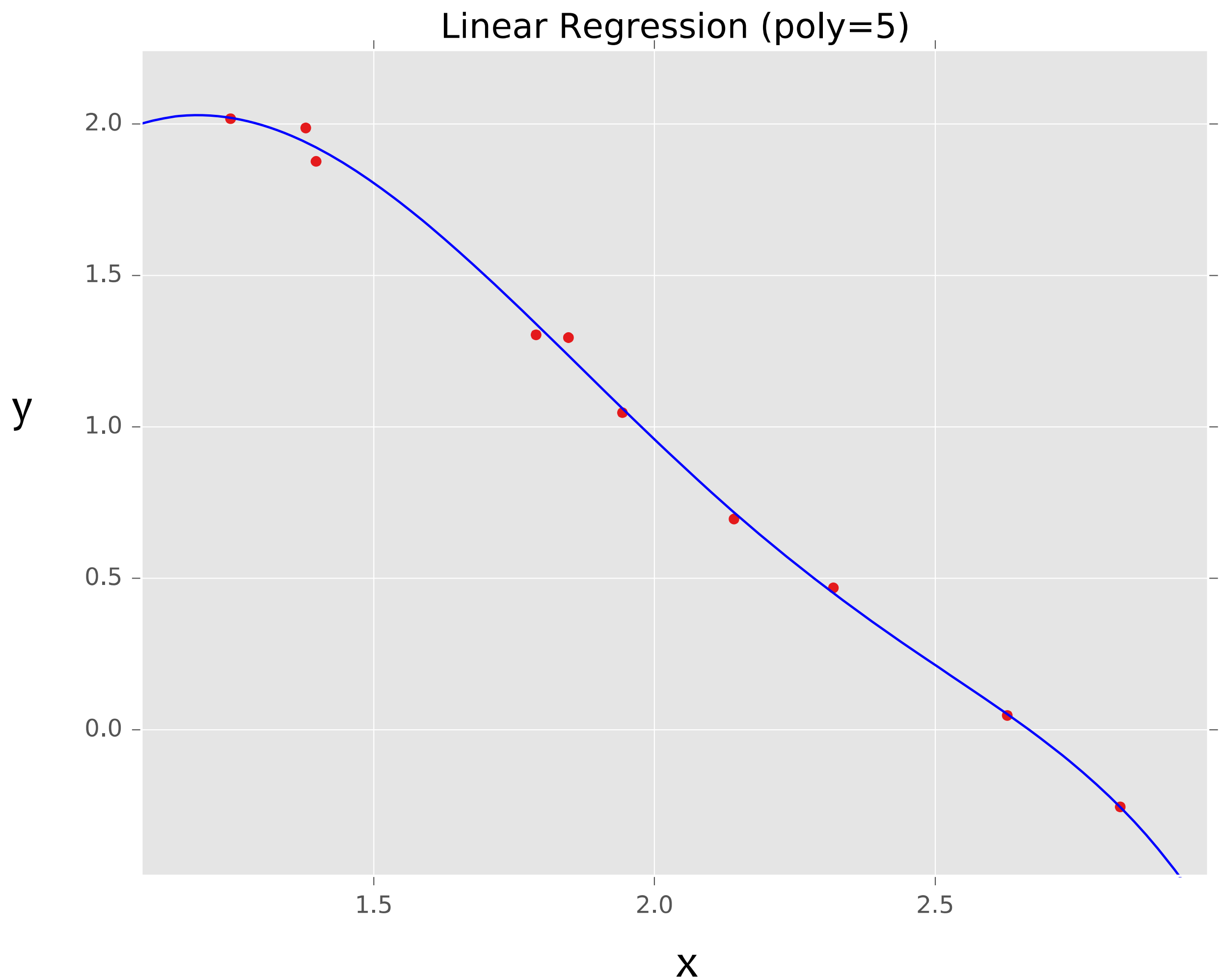
Goal: Learn y = $\mathbf{w}^T$ f($\mathbf{x}$) + b
where f(.) is a polynomial
basis function



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

Example: Linear Regression

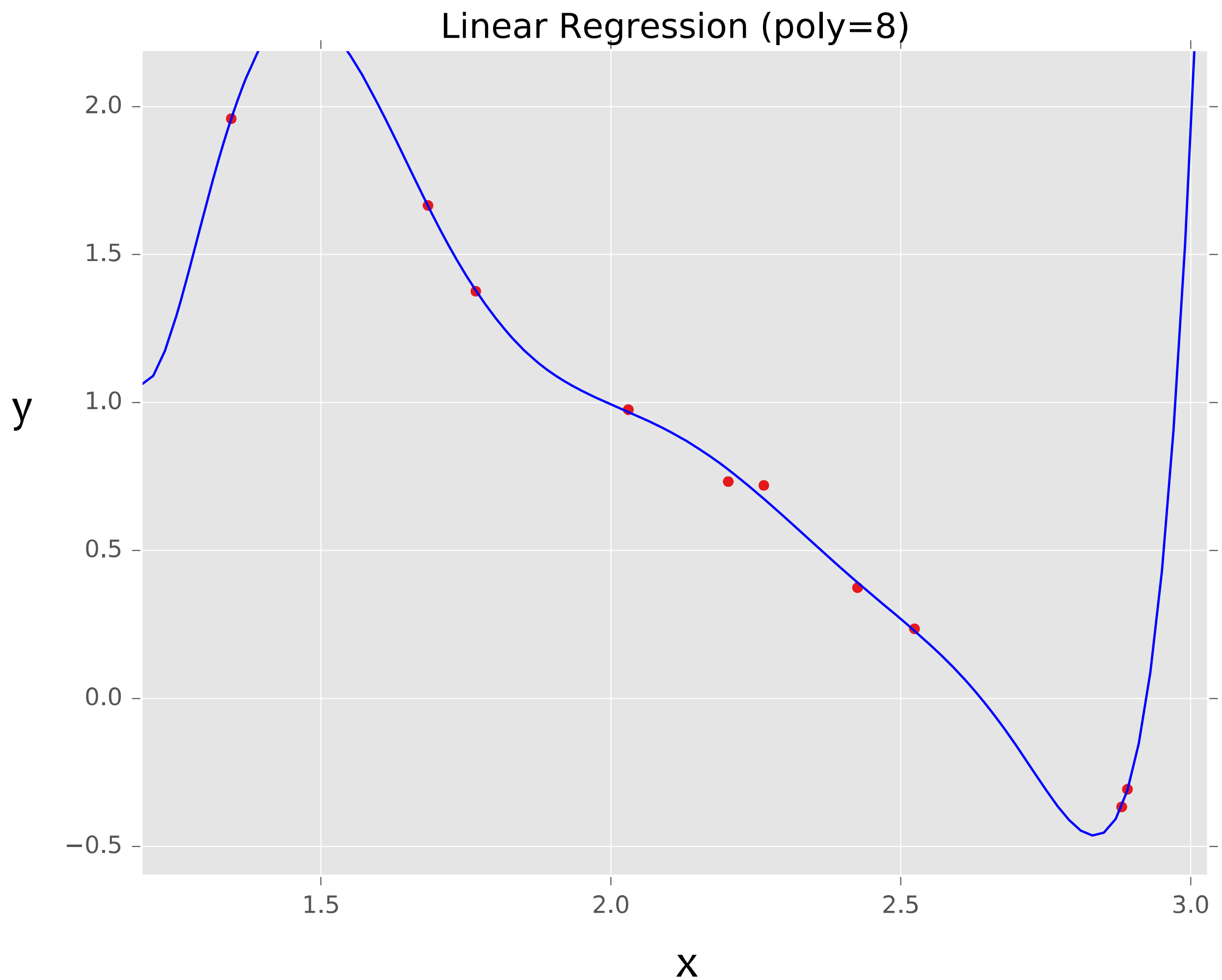
Goal: Learn y = $\mathbf{w}^T$ f($\mathbf{x}$) + b
where f(.) is a polynomial
basis function



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

Example: Linear Regression

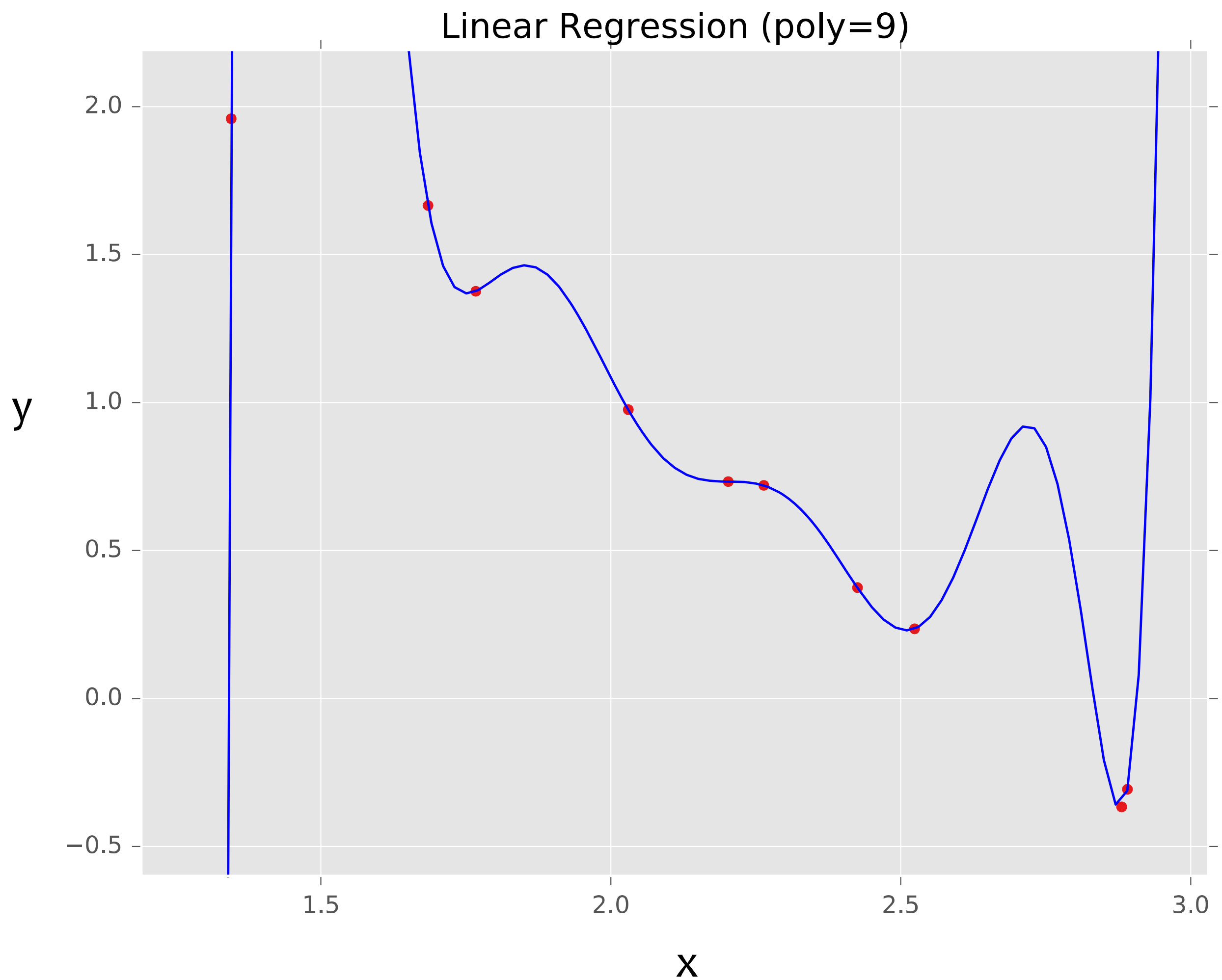
Goal: Learn y = $\mathbf{w}^T$ f($\mathbf{x}$) + b
where f(.) is a polynomial basis function



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

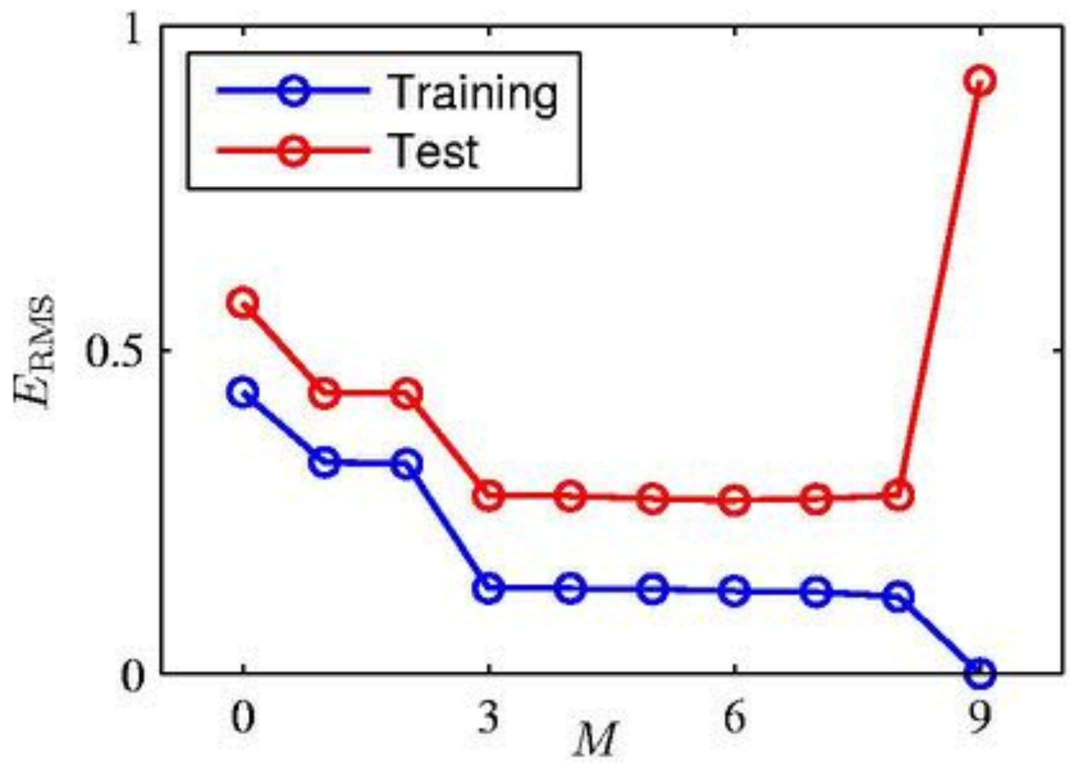
Example: Linear Regression

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



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

Over-fitting



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

Slide courtesy of William Cohen

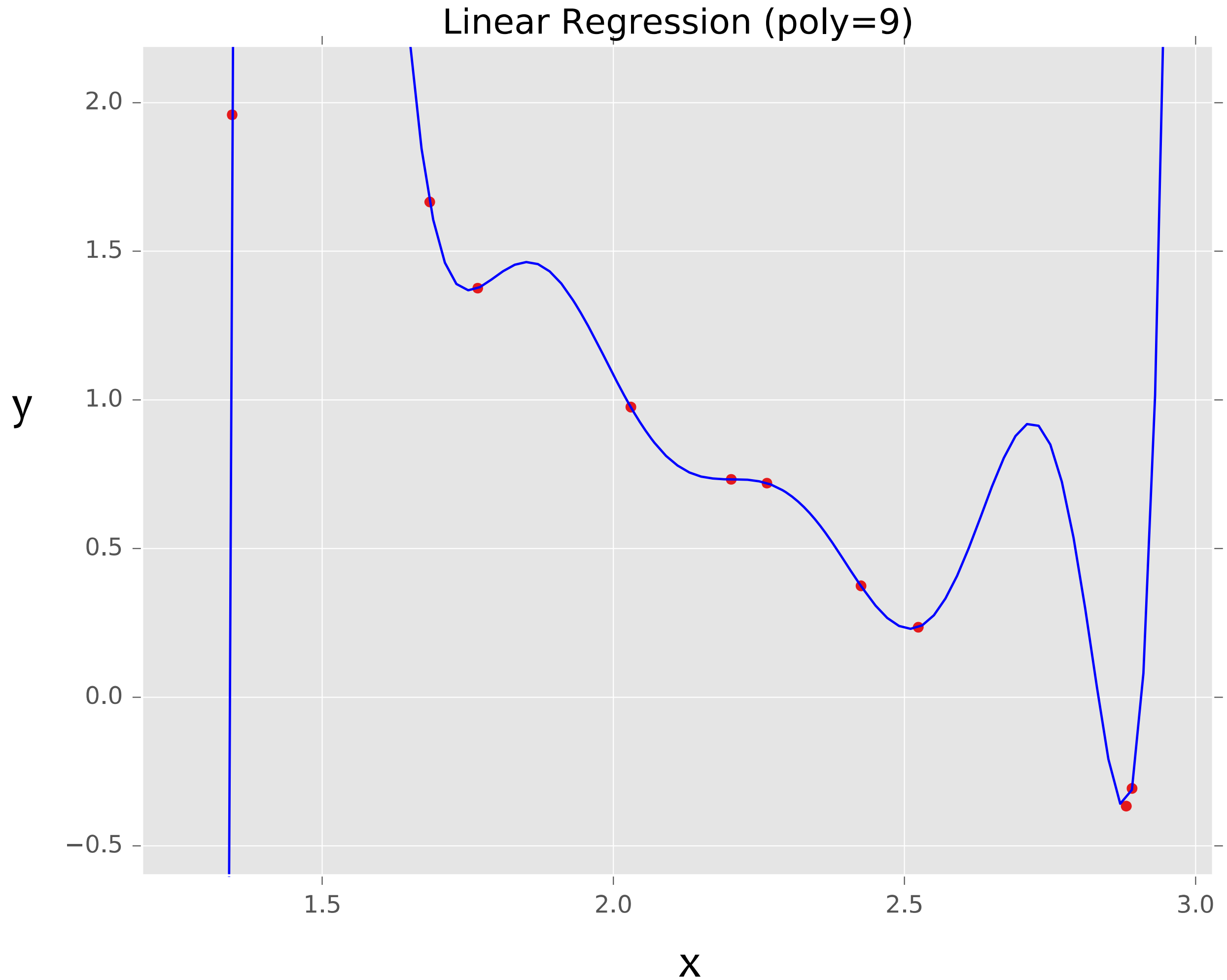
Polynomial Coefficients

	$M=0$	$M=1$	$M=3$	$M=9$
θ_0	0.19	0.82	0.31	0.35
θ_1		-1.27	7.99	232.37
θ_2			-25.43	-5321.83
θ_3			17.37	48568.31
θ_4				-231639.30
θ_5				640042.26
θ_6				-1061800.52
θ_7				1042400.18
θ_8				-557682.99
θ_9				125201.43

Slide courtesy of William Cohen

Example: Linear Regression

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

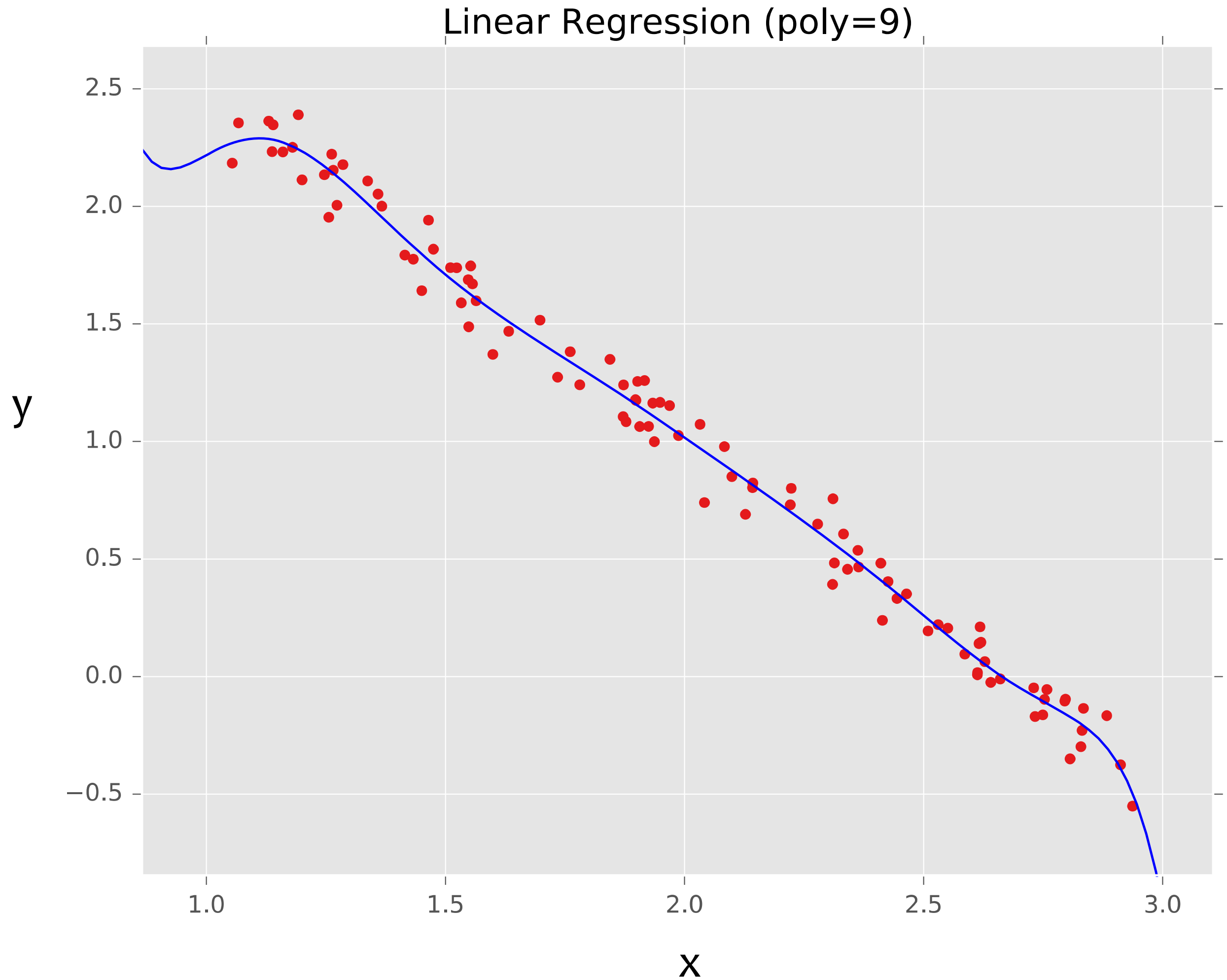


true "unknown" target function is linear with negative slope and gaussian noise

Example: Linear Regression

Goal: Learn y = $\mathbf{w}^T f(\mathbf{x}) + b$ where f(.) 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, … , 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(θ)
- **Goal** is to find: $\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}}\, J(\boldsymbol{\theta}) + \lambda r(\boldsymbol{\theta})$
- **Key idea:** Define regularizer r(θ) s.t. we tradeoff between fitting the data and keeping the model simple
- **Choose form of r(θ):**
  – Example: q-norm (usually p-norm) $r(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_q = \left[\sum_{m=1}^{M} ||\theta_m||^q\right]^{(\frac{1}{q})}$

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

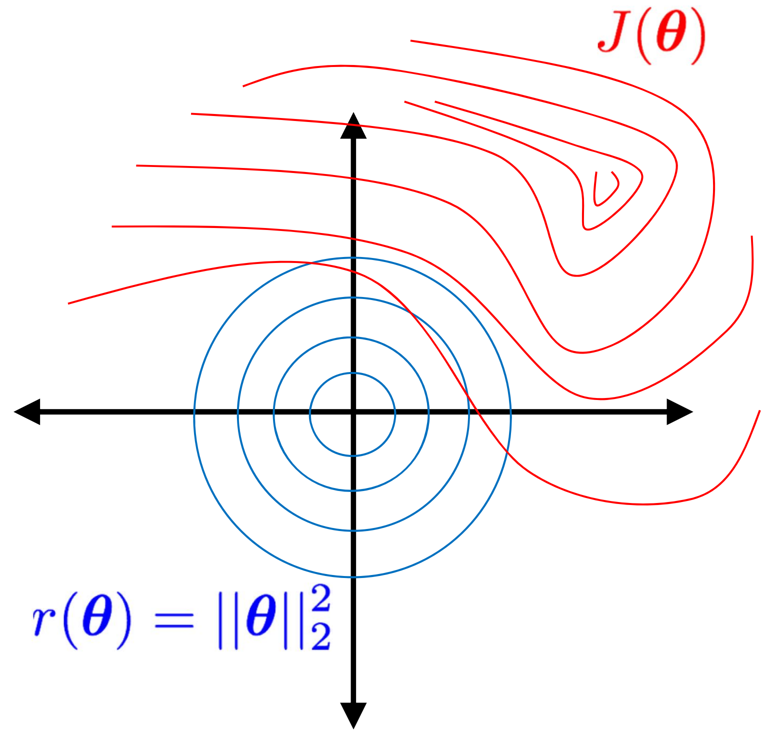
Question:

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 move…

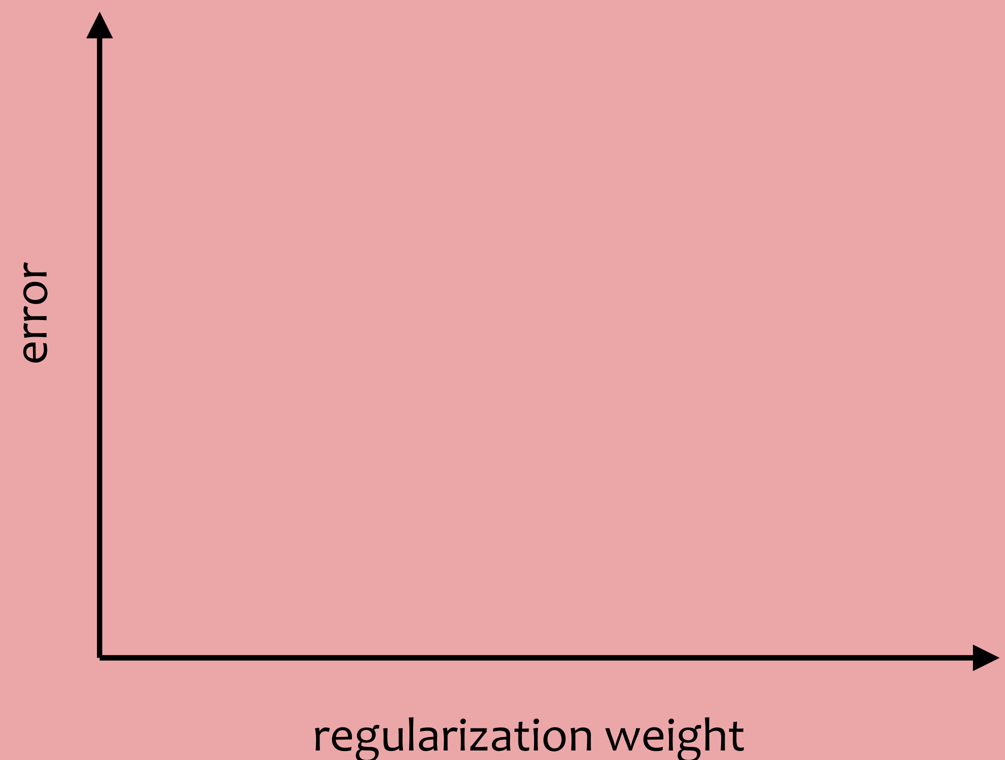
A. …towards the midpoint between J'(θ) and r(θ)
B. …towards the minimum of J(θ)
C. …towards the minimum of r(θ)
D. …towards a theta vector of positive infinities
E. …towards a theta vector of negative infinities



Regularization Exercise

In-class Exercise

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



Regularization

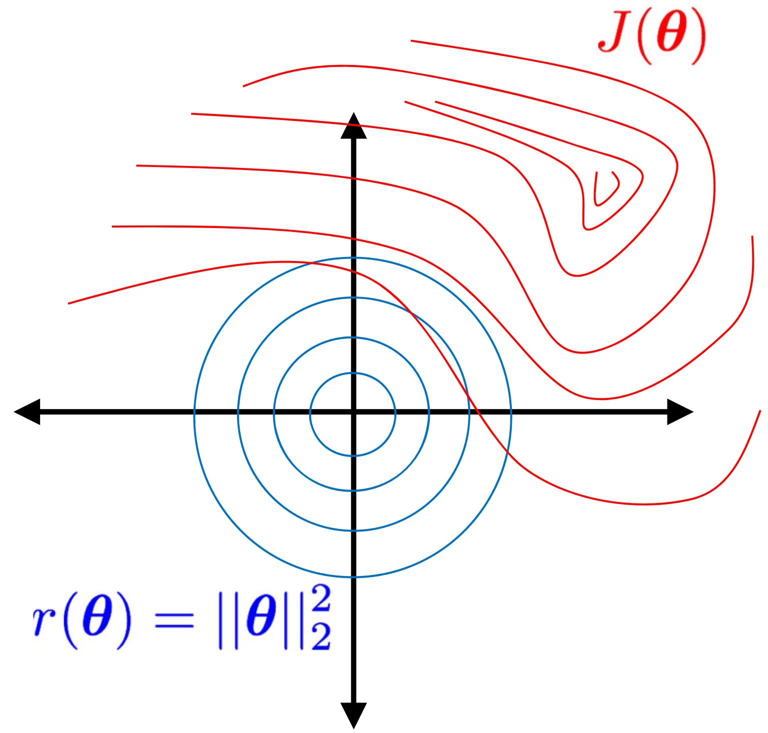
Question:

Suppose we are minimizing J'(θ) where

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

As we increase λ 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!

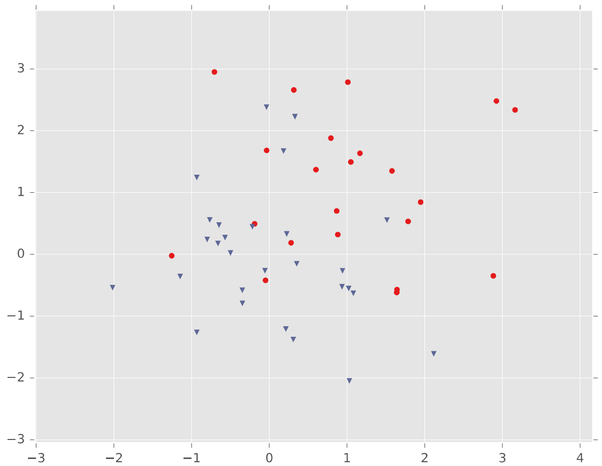
- In our models so far, the bias / intercept parameter is usually denoted by θ_0 -- that is, the parameter for which we fixed $x_0 = 1$
- Regularizers always avoid penalizing this bias / intercept parameter
- Why? Because otherwise the learning algorithms wouldn't be invariant to a shift in the y-values

Whitening Data

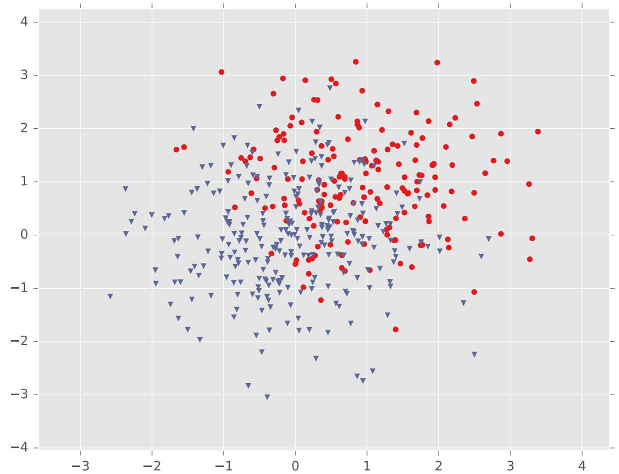
- It's common to *whiten* each feature by subtracting its mean and dividing by its variance
- For regularization, this helps all the features be penalized in the same units
(e.g. convert both centimeters and kilometers to z-scores)

Example: Logistic Regression

Training Data

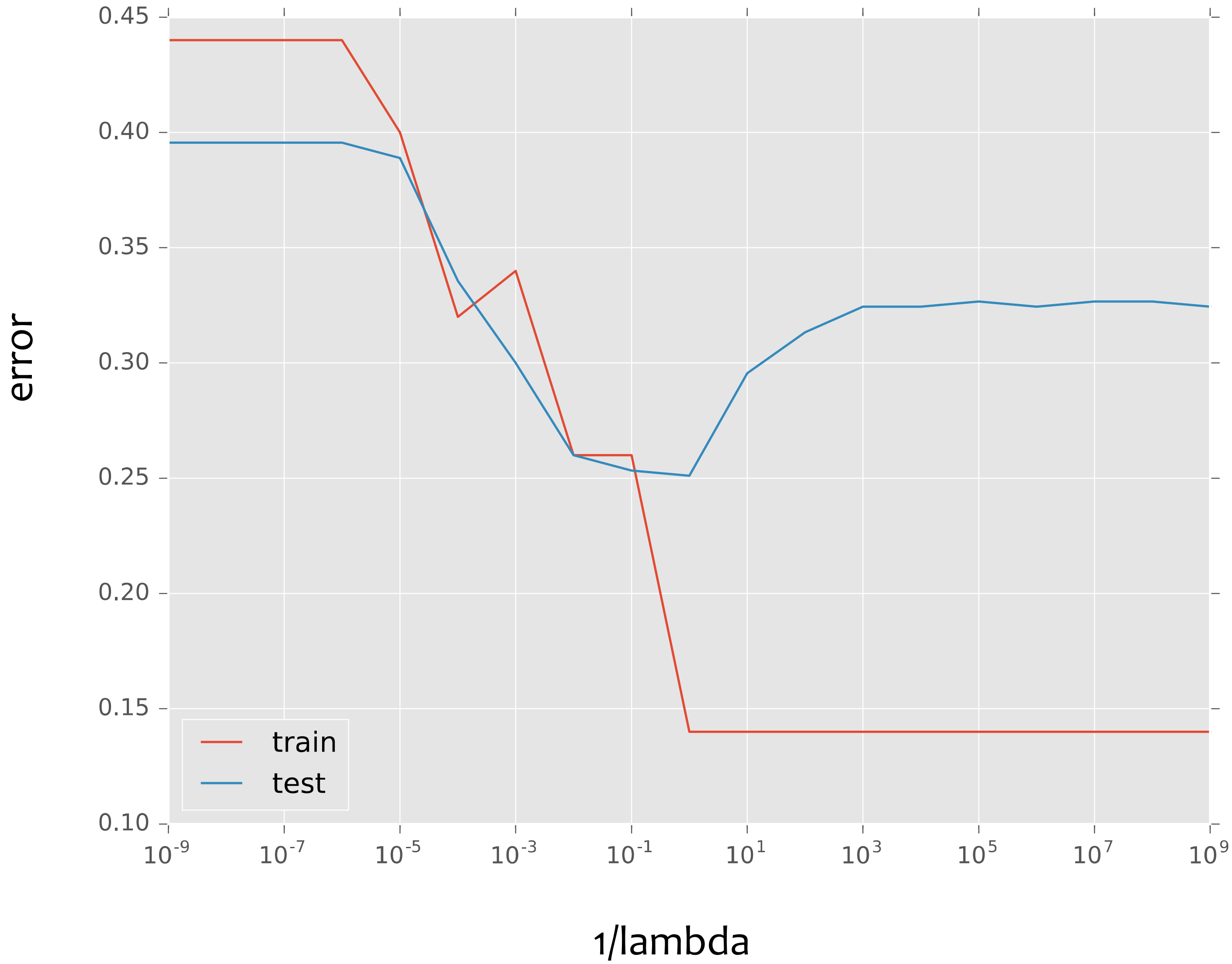


Test Data

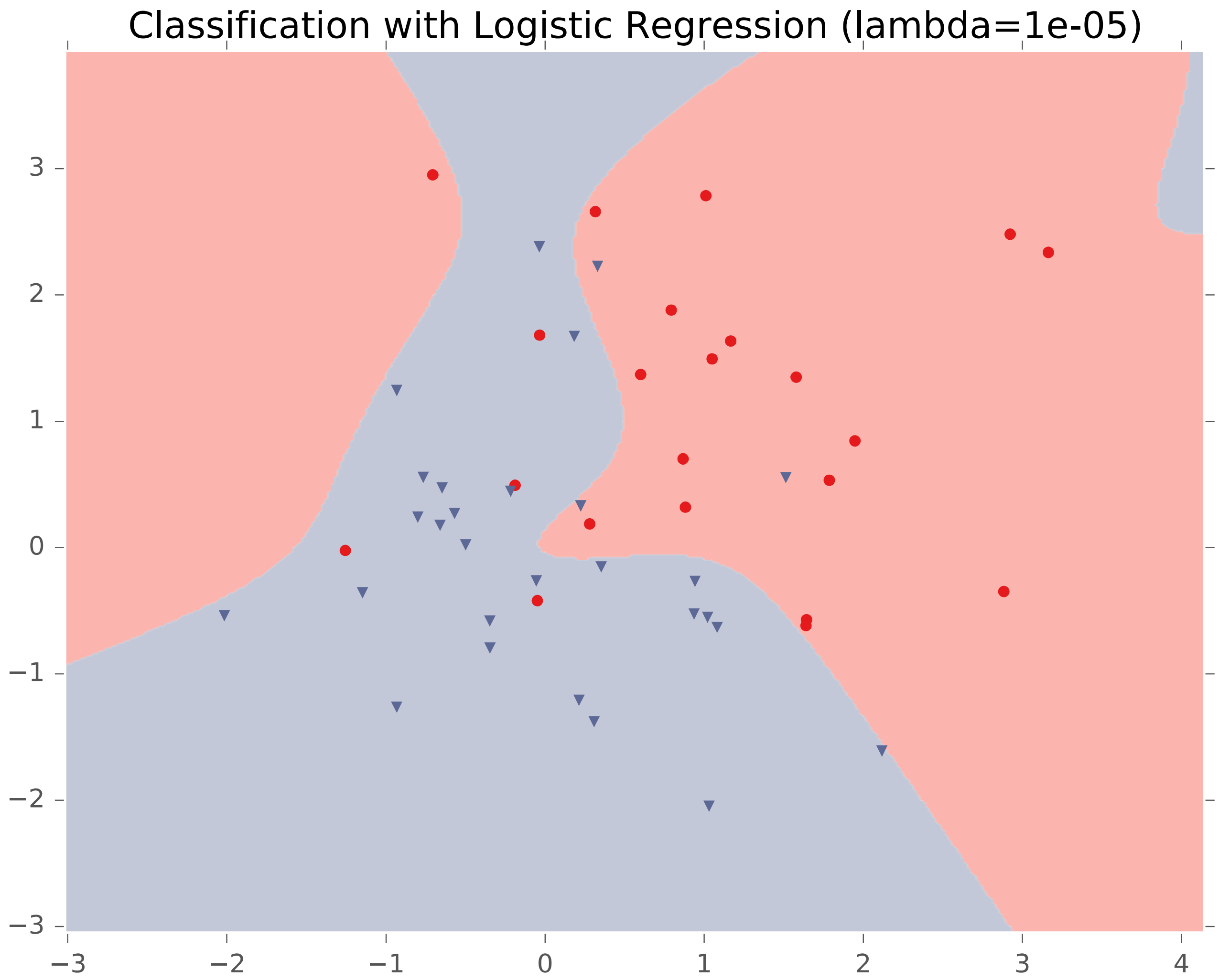


- For this example, we construct **nonlinear features** (i.e. feature engineering)
- Specifically, we add **polynomials up to order 9** of the two original features x_1 and x_2
- Thus our classifier is **linear** in the **high-dimensional feature space,** but the decision boundary is **nonlinear** when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

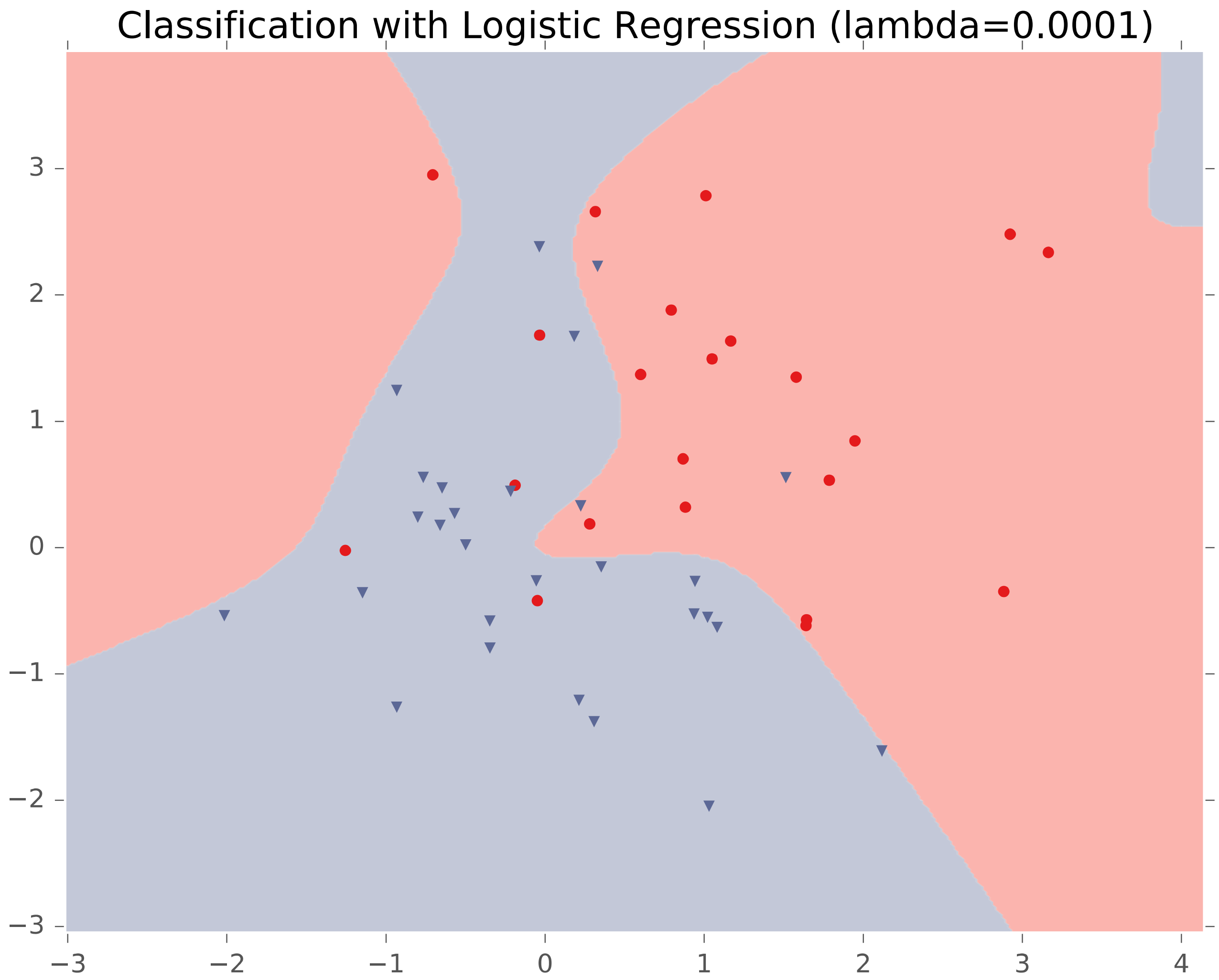
Example: Logistic Regression



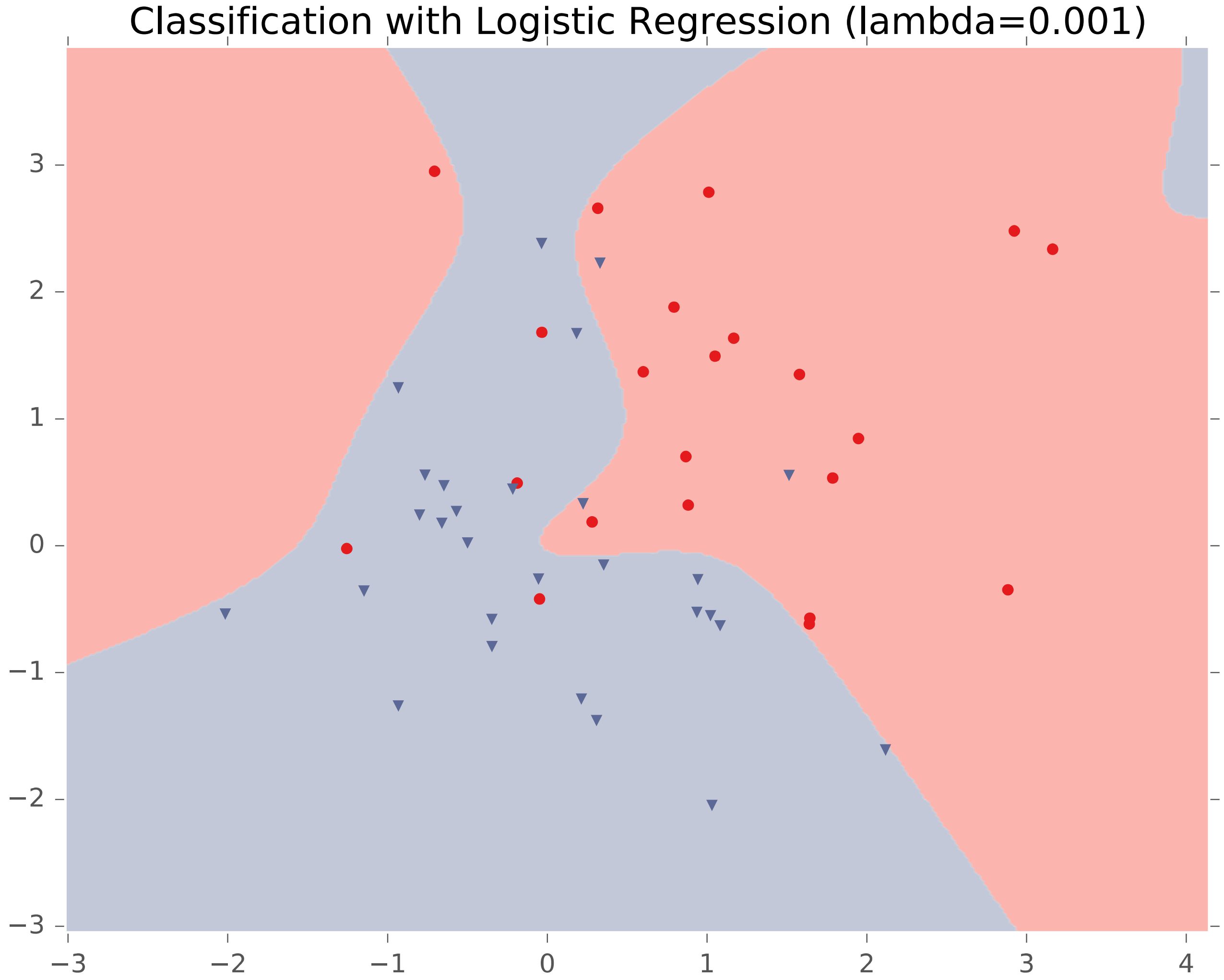
Example: Logistic Regression



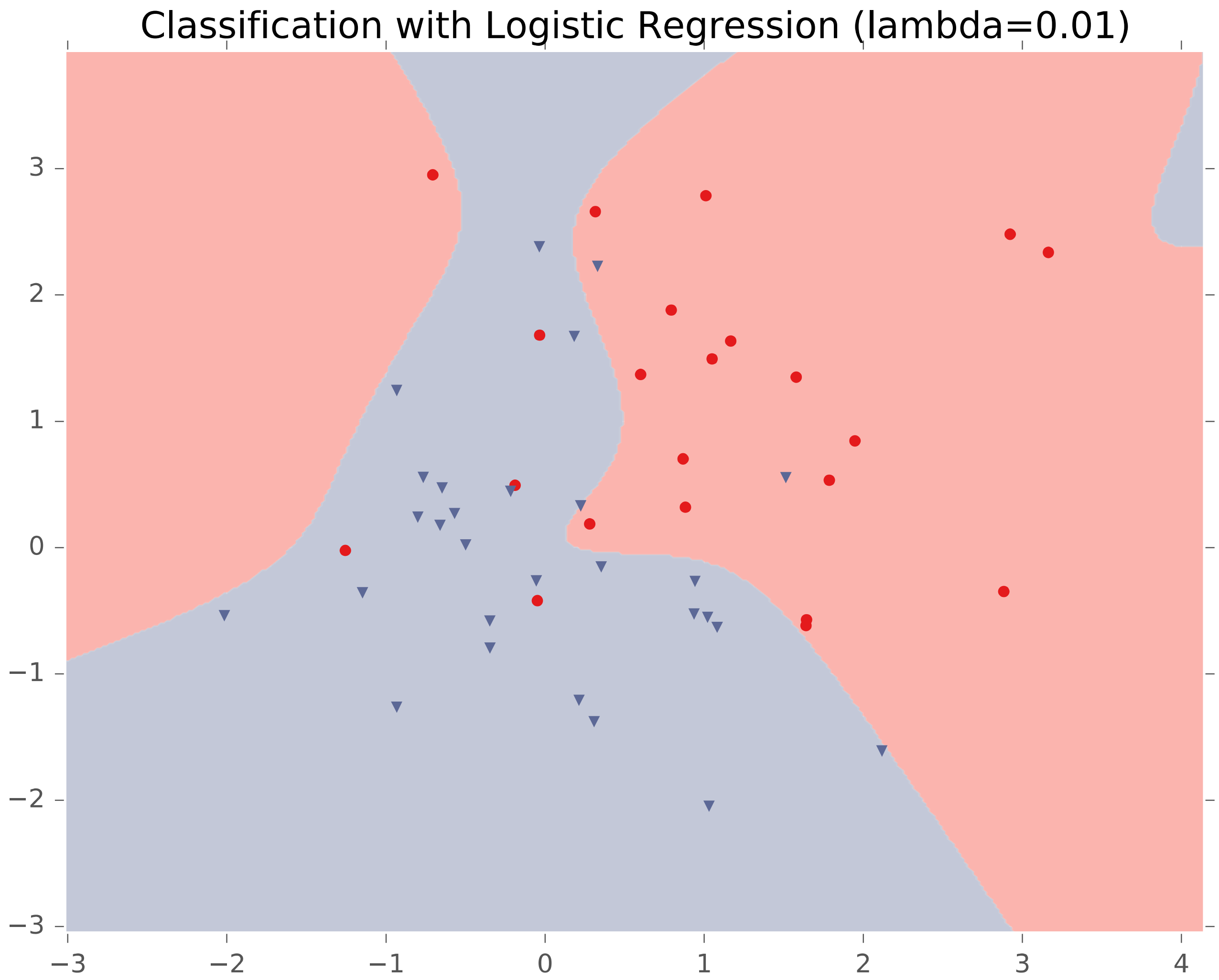
Example: Logistic Regression



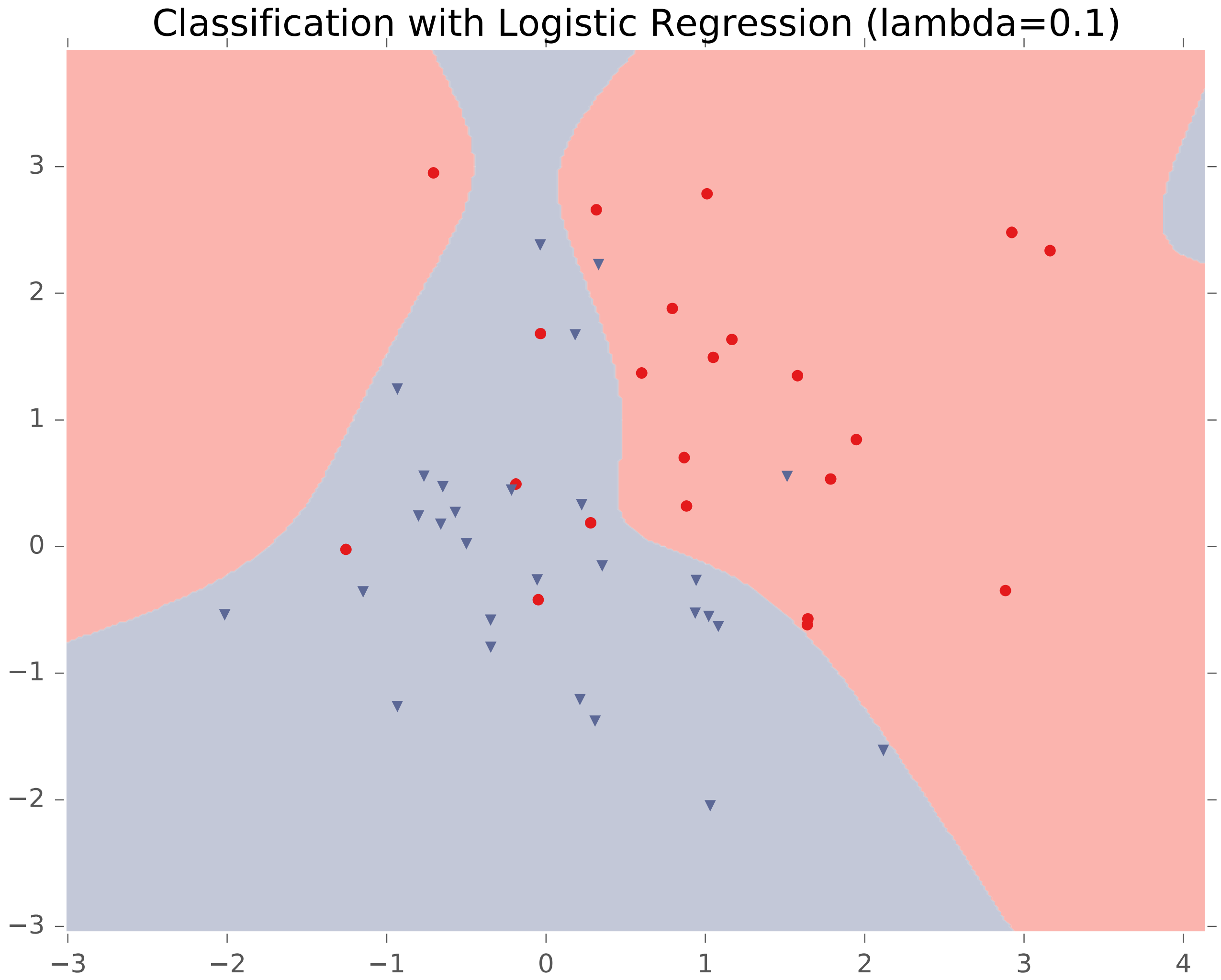
Example: Logistic Regression



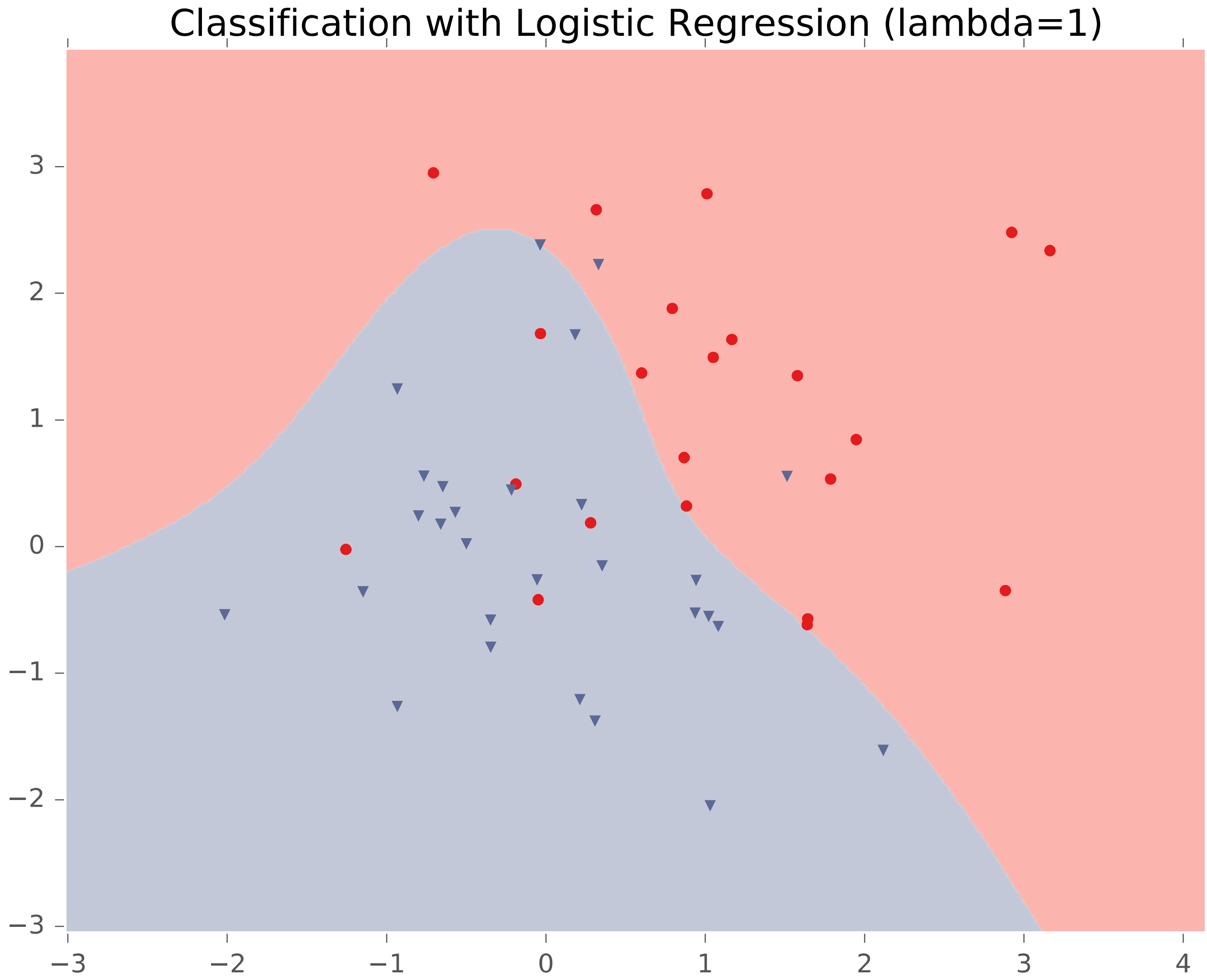
Example: Logistic Regression



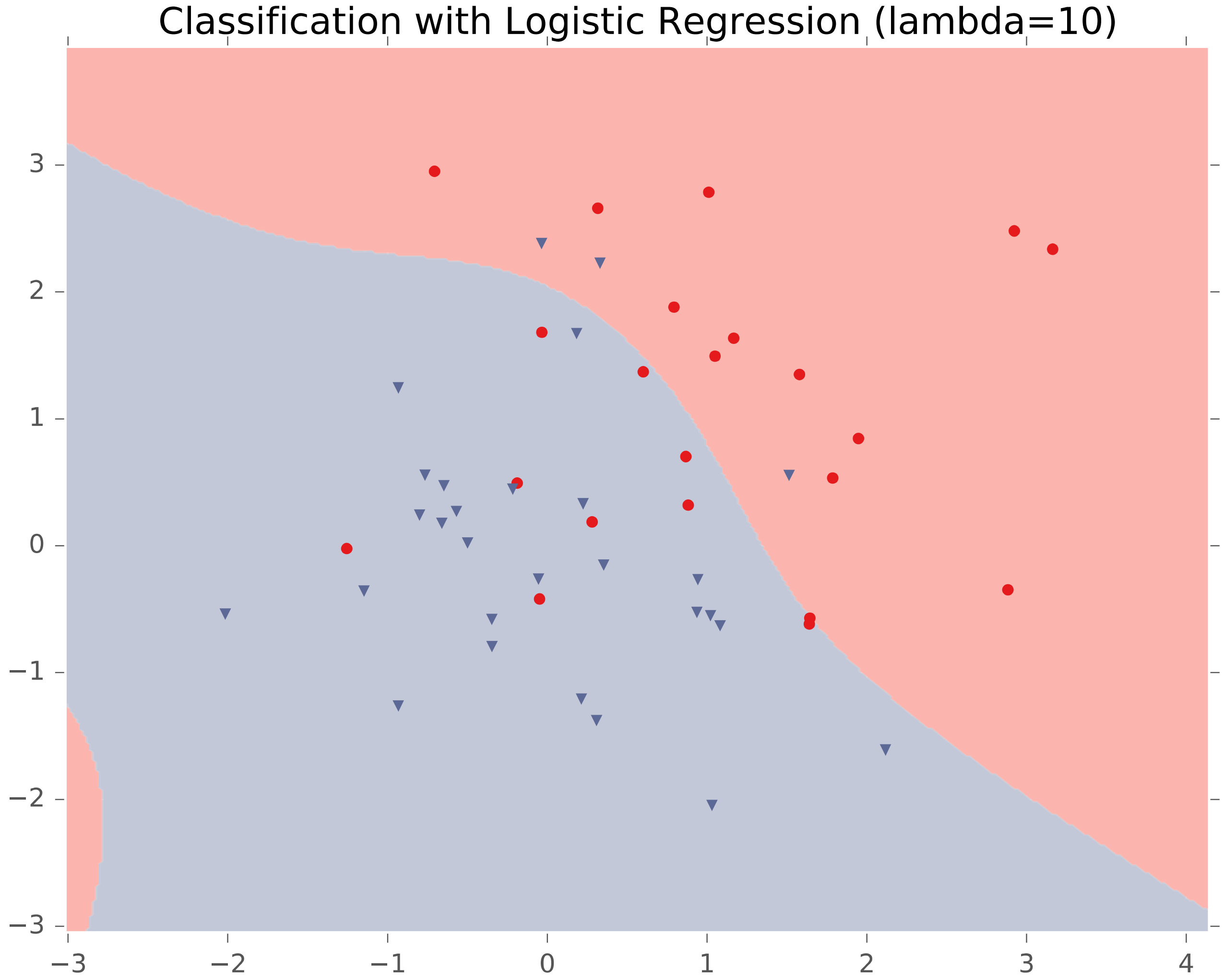
Example: Logistic Regression



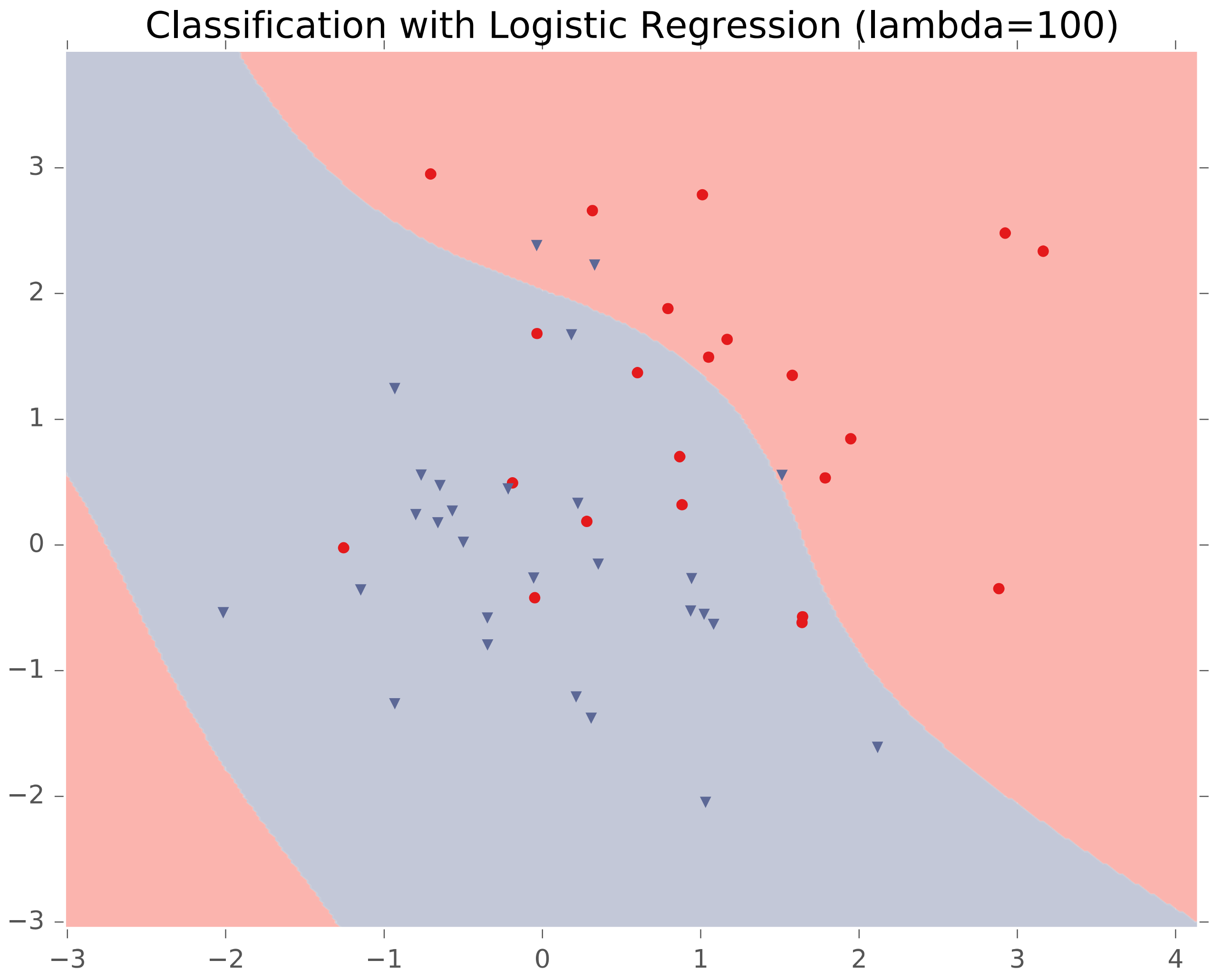
Example: Logistic Regression



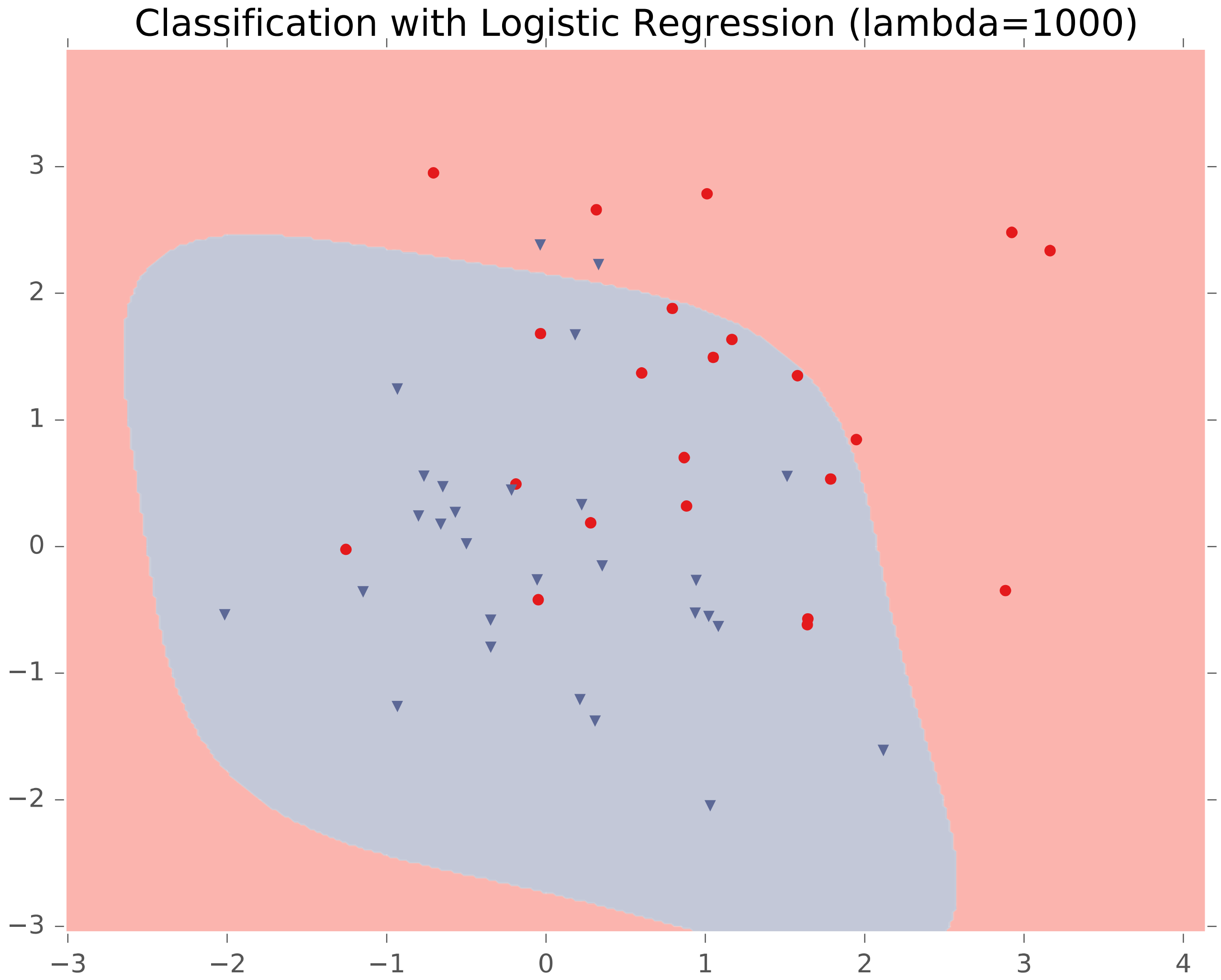
Example: Logistic Regression



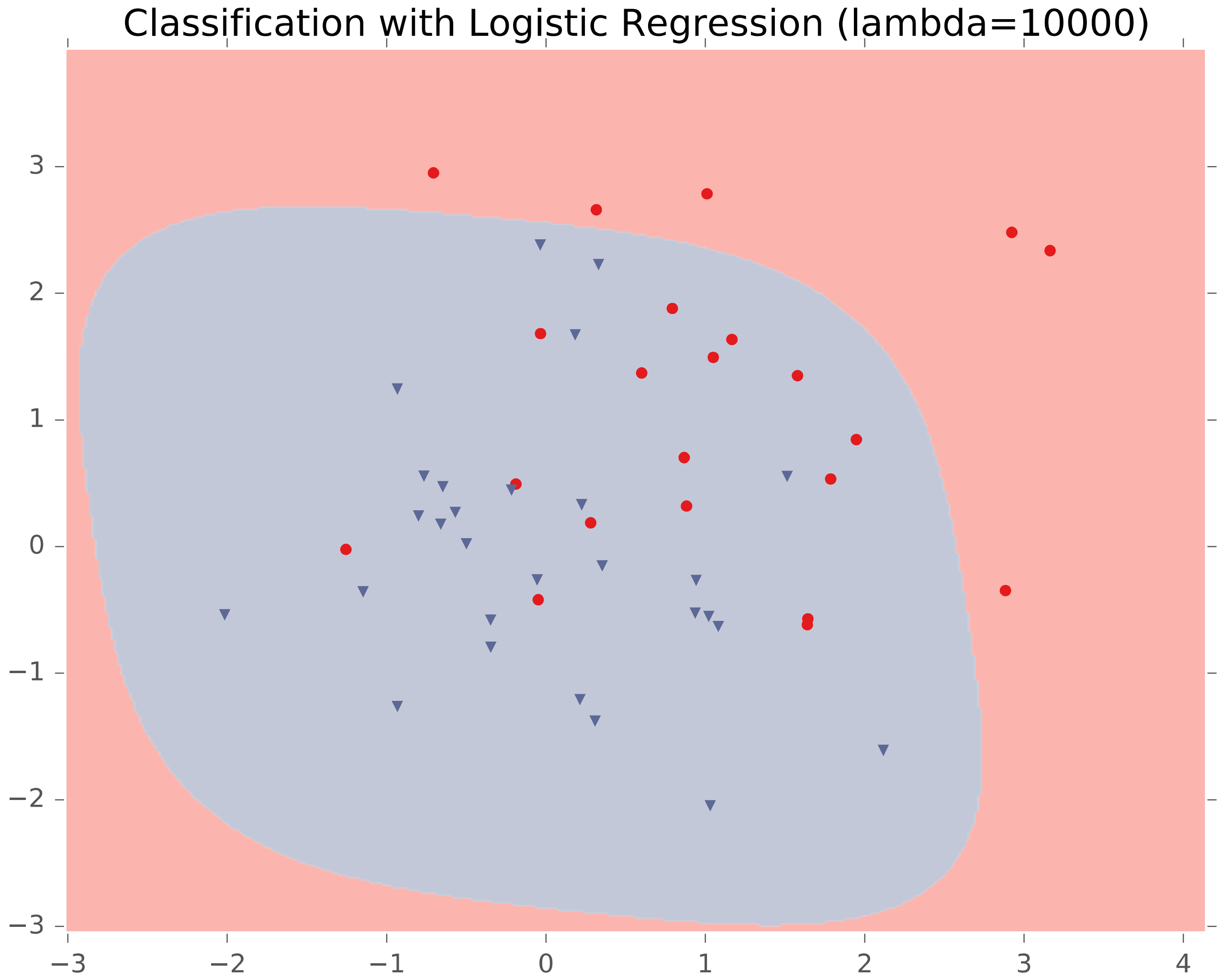
Example: Logistic Regression



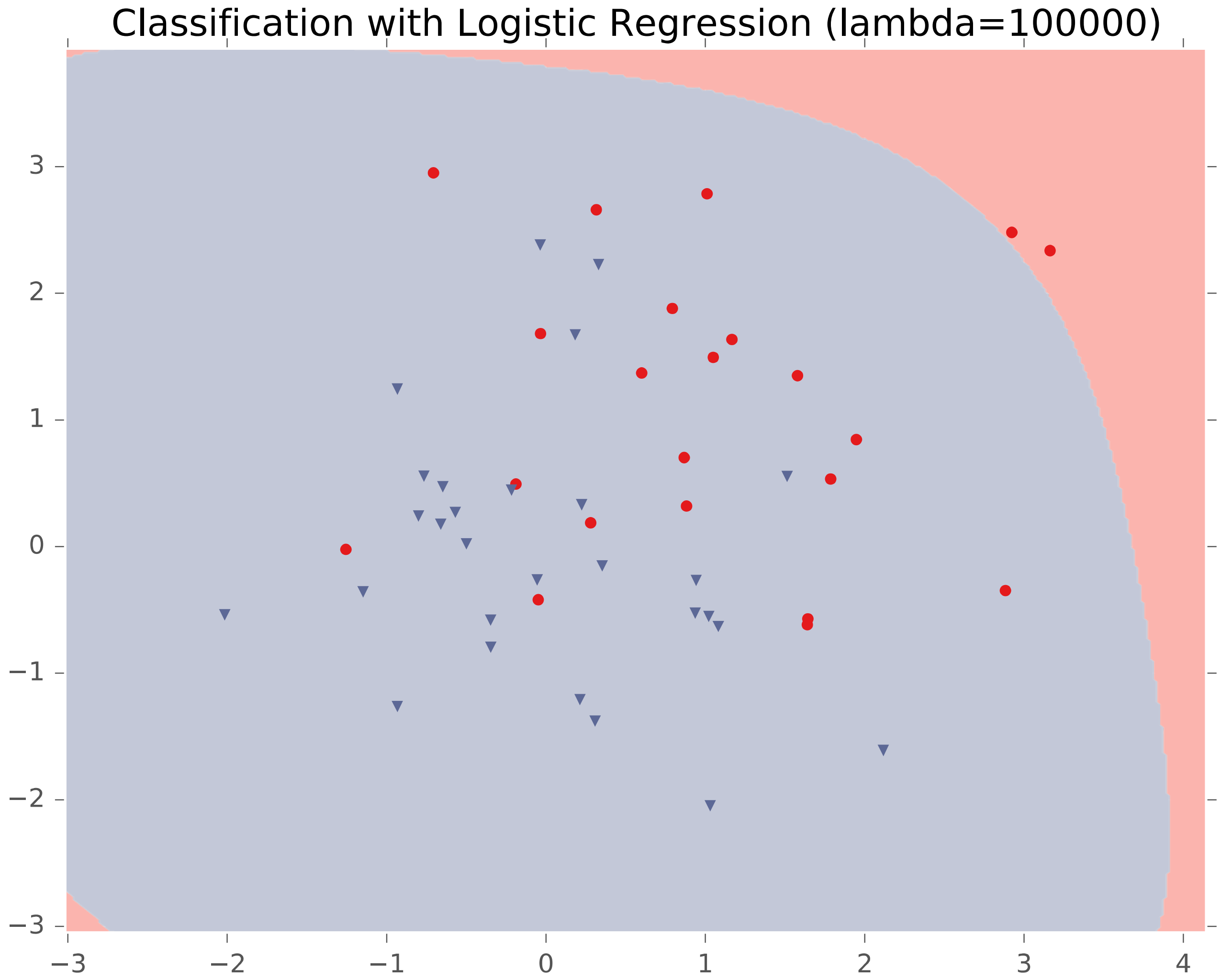
Example: Logistic Regression



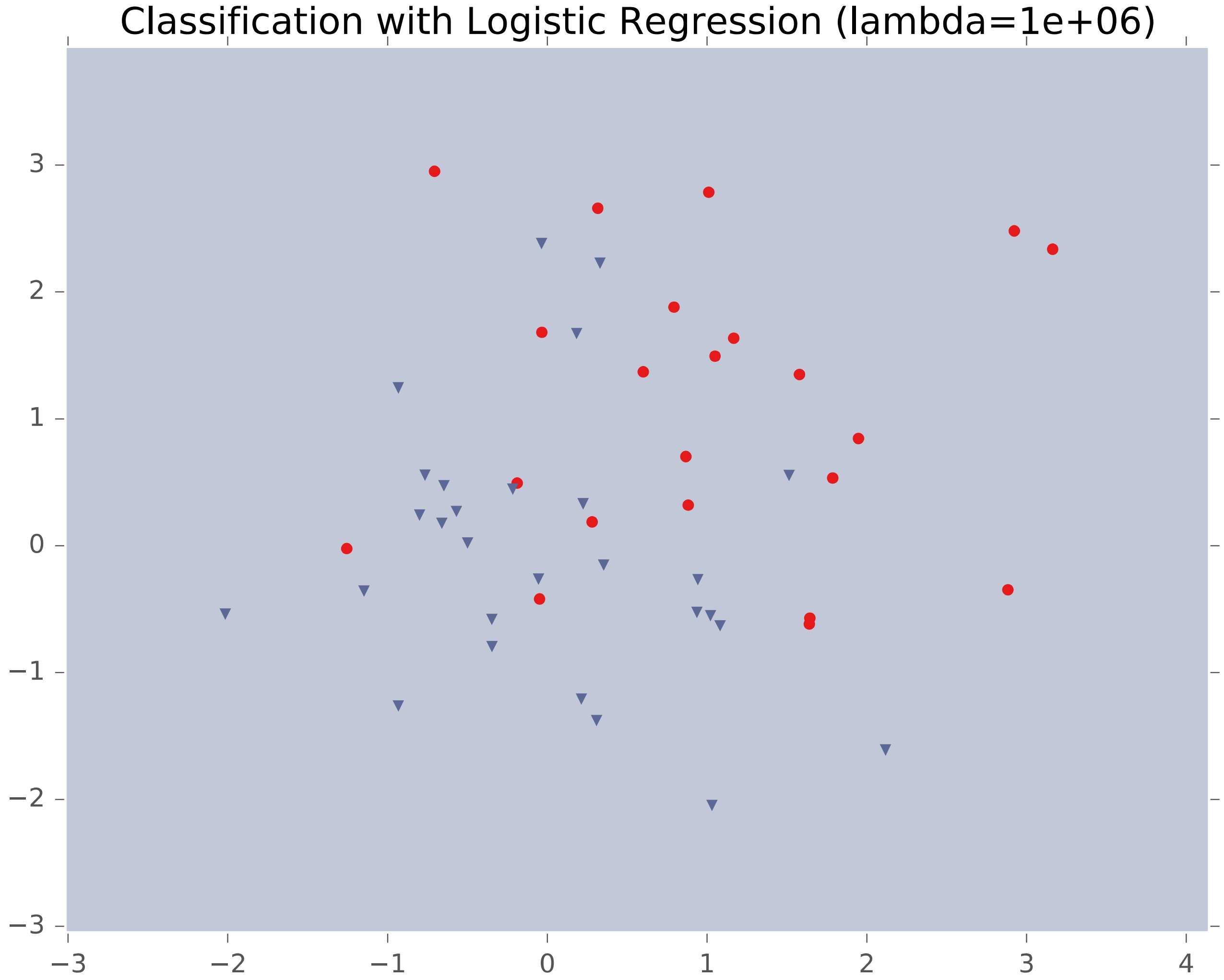
Example: Logistic Regression



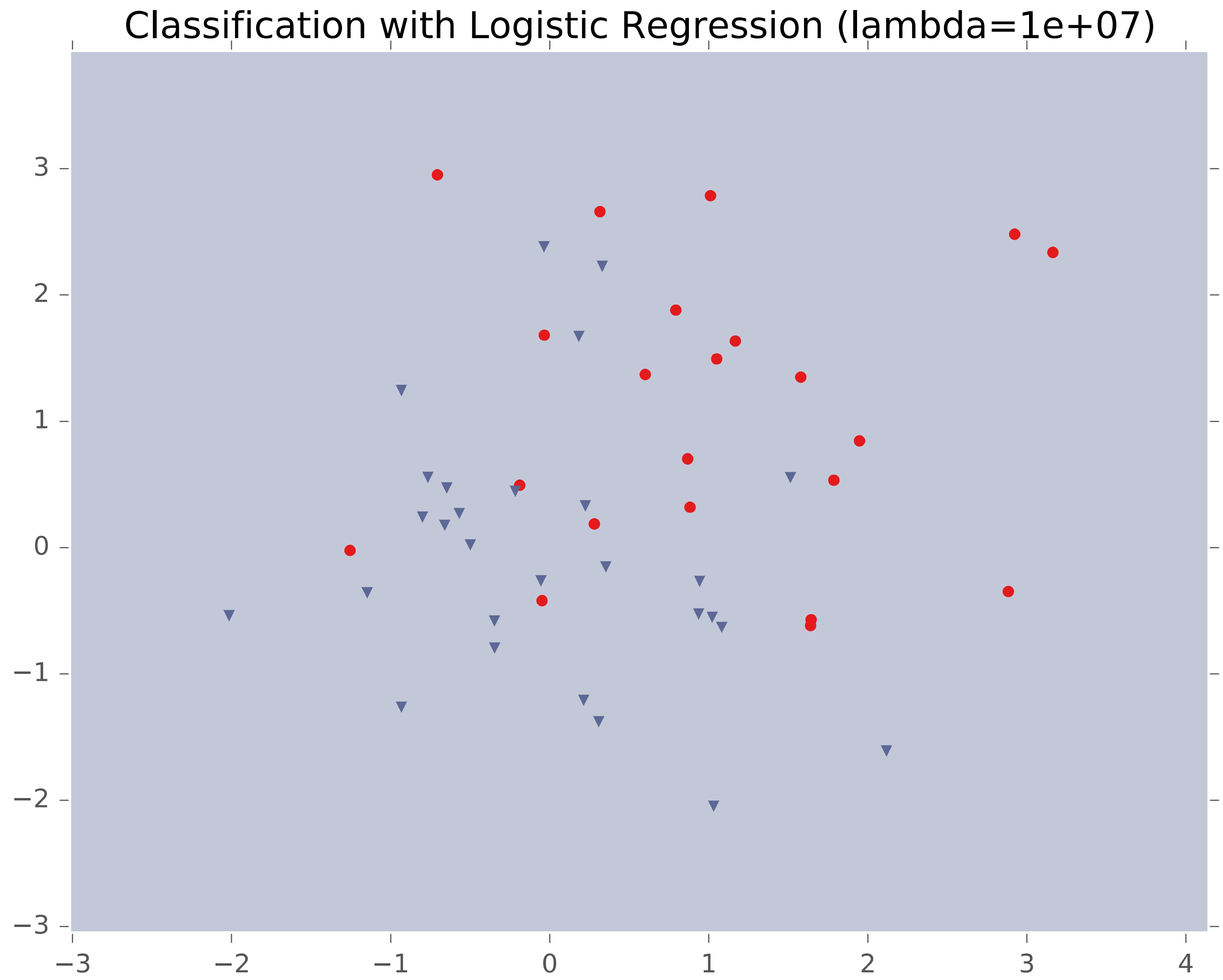
Example: Logistic Regression



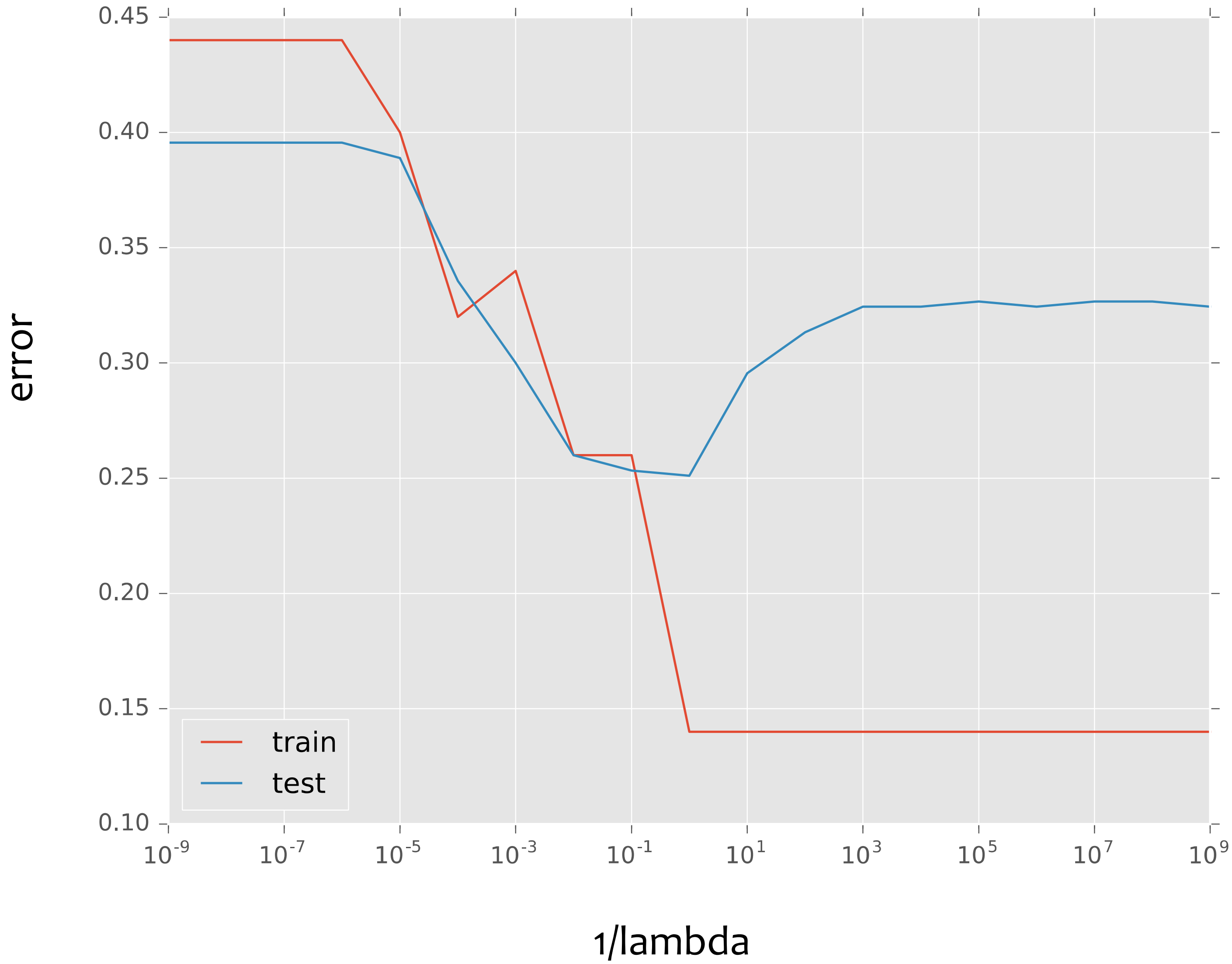
Example: Logistic Regression



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 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