10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Feature Engineering

+

Regularization

Matt Gormley
Lecture 10
Oct. 1, 2018

Reminders

- **Homework 4: Logistic Regression**
  - **Out: Sun, Sep 30**
  - **Due: Mon, Oct 8 at 11:59pm**
- Reading on Probabilistic Learning is reused later in the course for MLE/MAP
- **Schedule changes:**
  - lecture on Friday (Oct. 5)
  - no lecture on Wednesday (Oct. 10)

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.

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]:
      for m in [1,…, M]:
          theta[k,m] = theta[k,m] + lambda * grad(x[i],
y[i], theta, k,m)
```

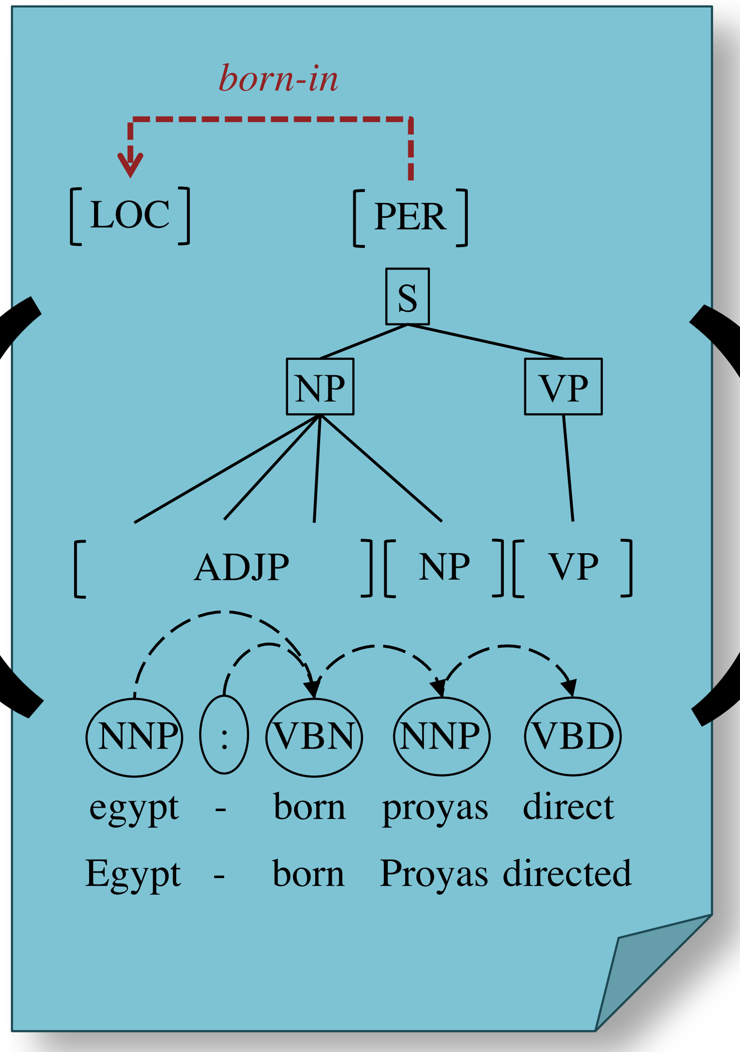
Assume: `grad(x[i], y[i], theta, k, m)` returns the partial derivative of the negative log-likelihood of the training example (x[i],y[i]) with respect to `theta[k,m]`. `lambda` is the learning rate. N = # of examples. K = # of output classes. M = # of features. `theta` is a K by M matrix.

FEATURE ENGINEERING

Handcrafted Features

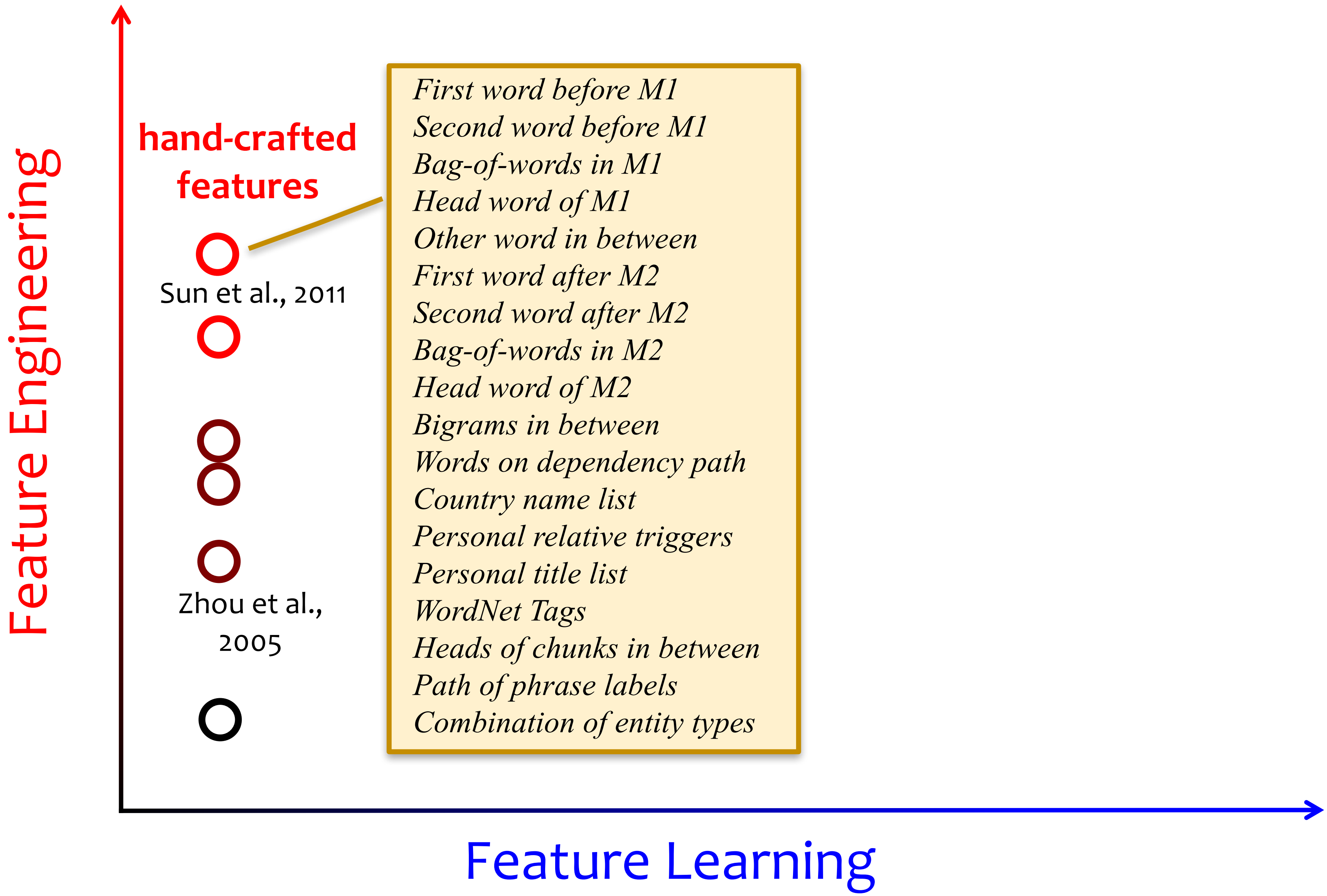
$p(y|x) \propto$

$\exp(\Theta_y \bullet f\Big($

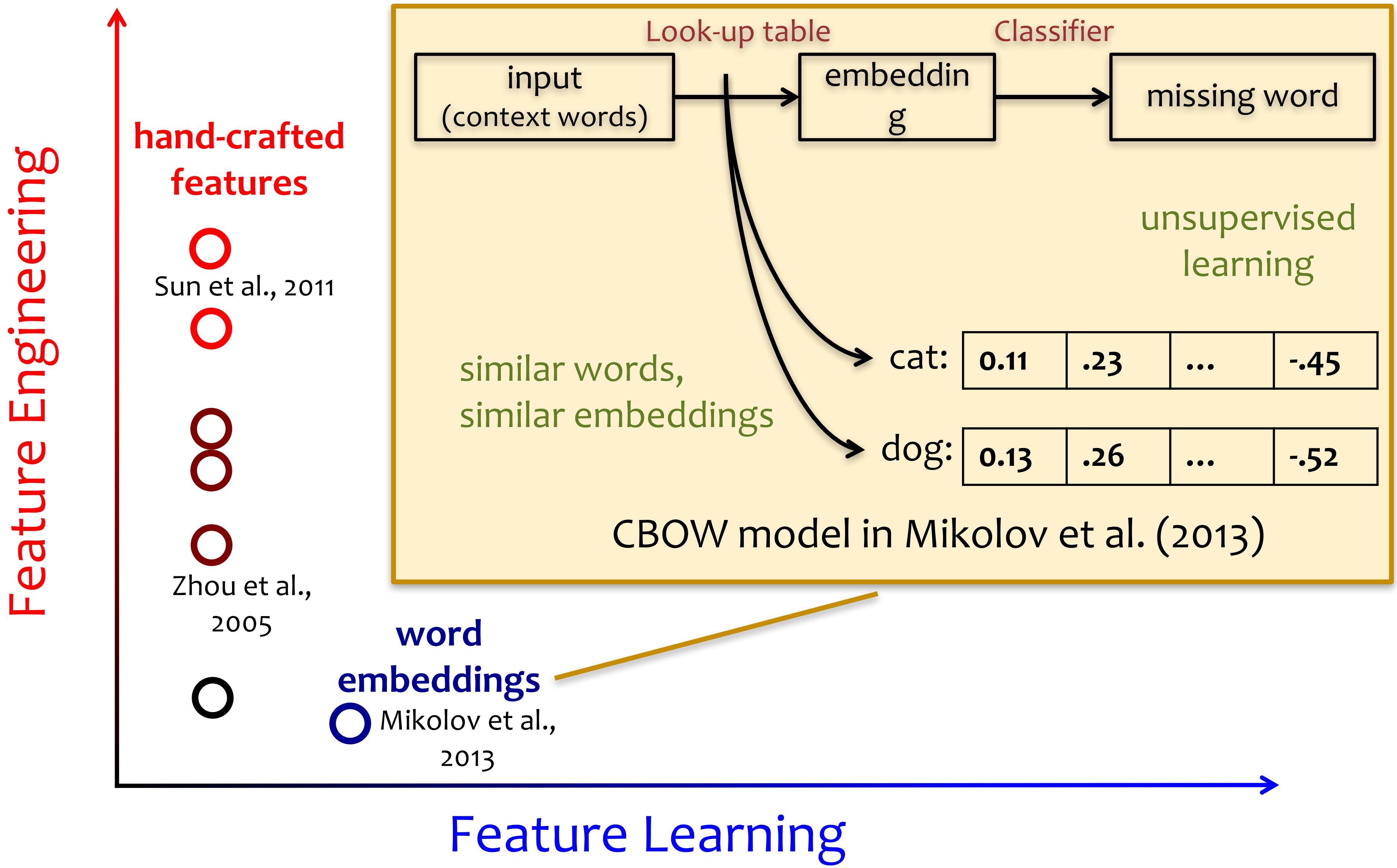


$\Big))$

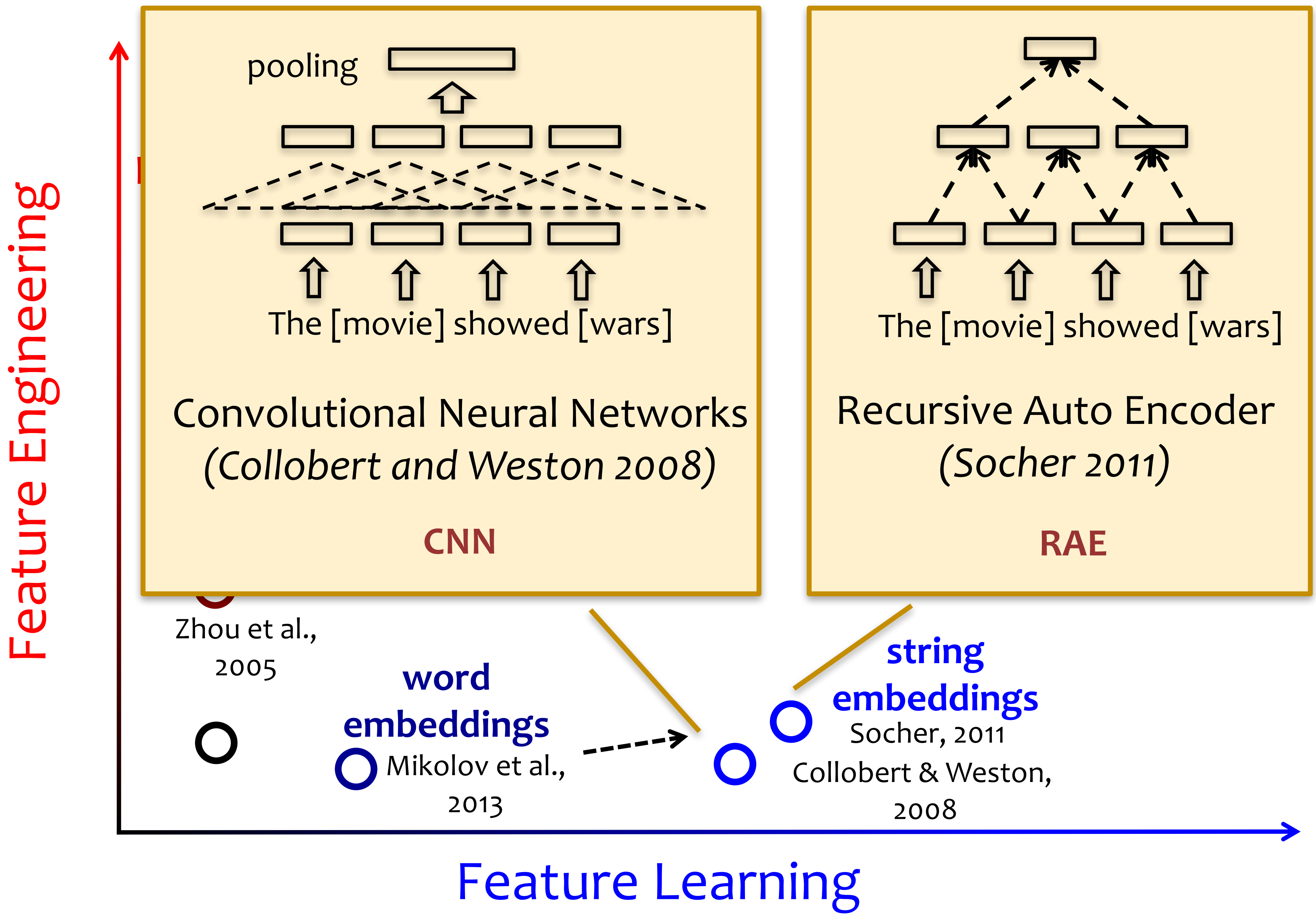
Where do features come from?



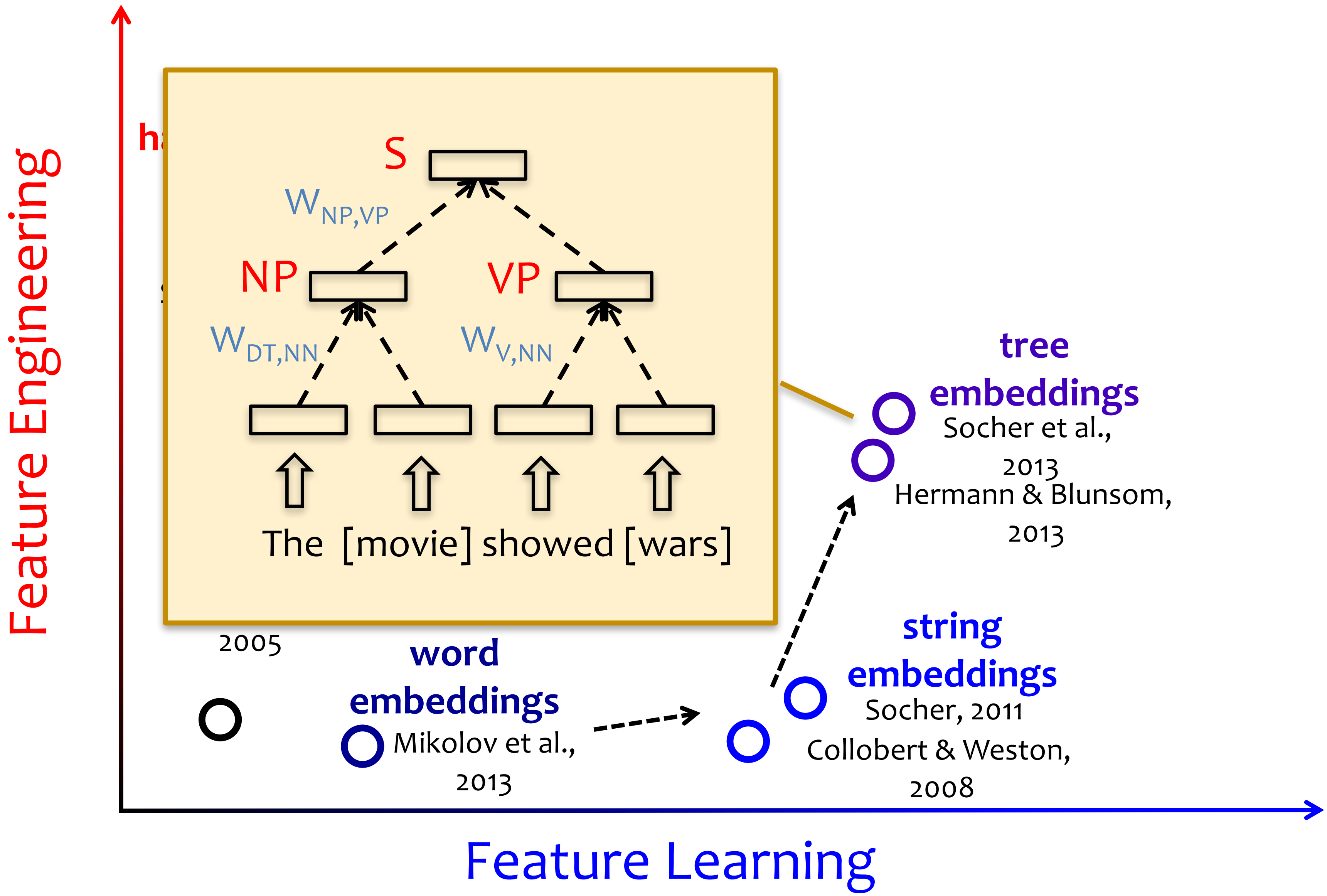
Where do features come from?



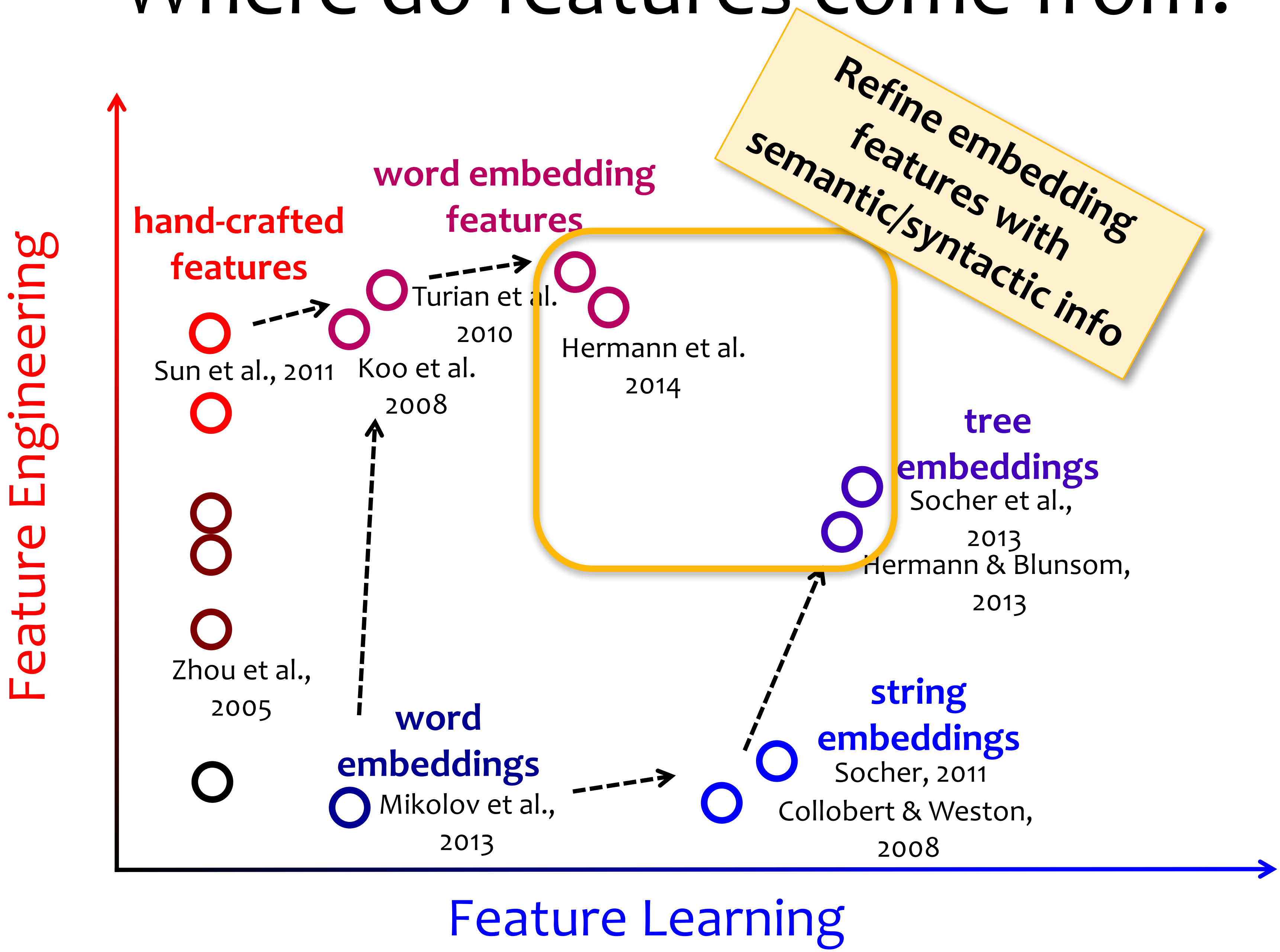
Where do features come from?



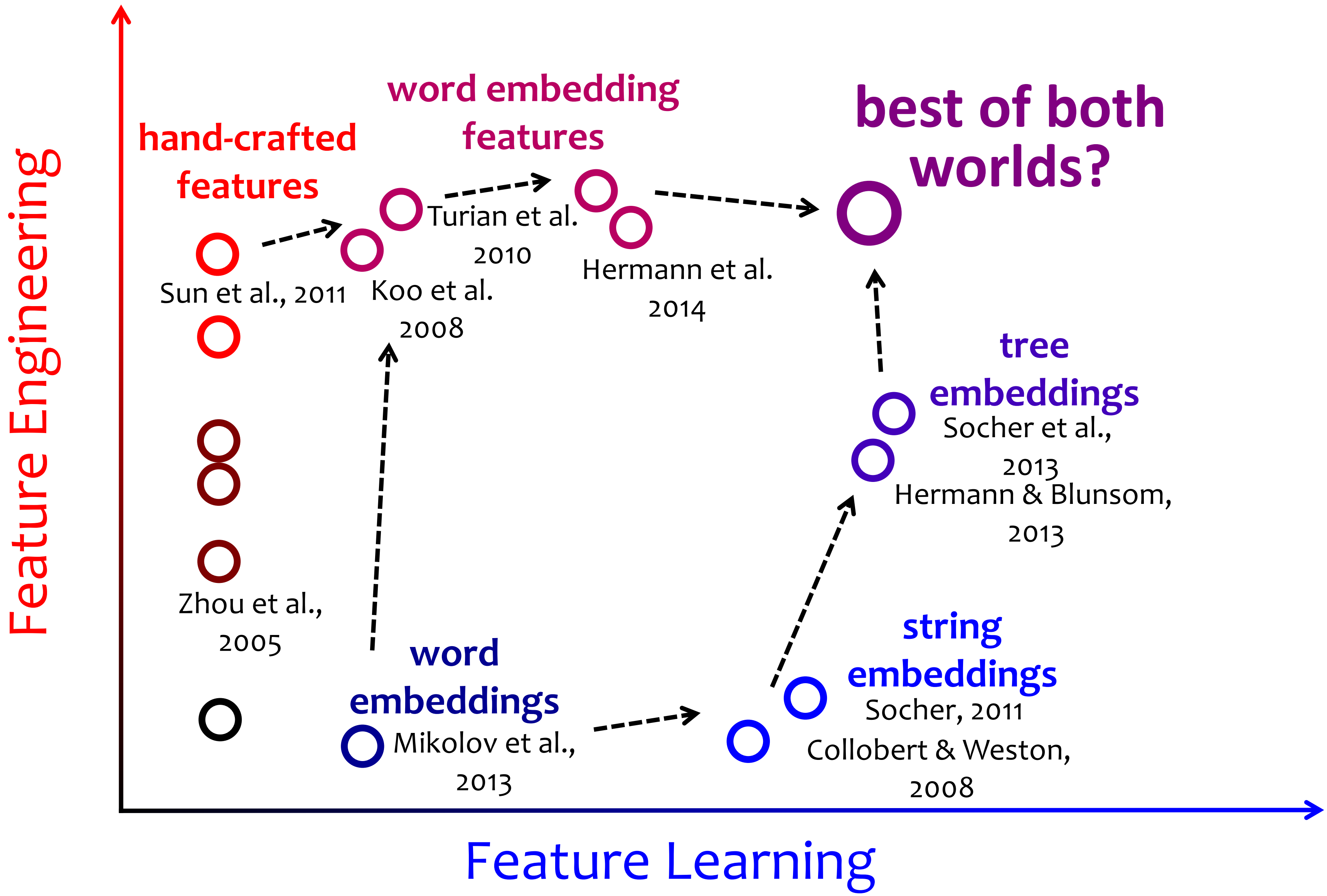
Where do features come from?



Where do features come from?



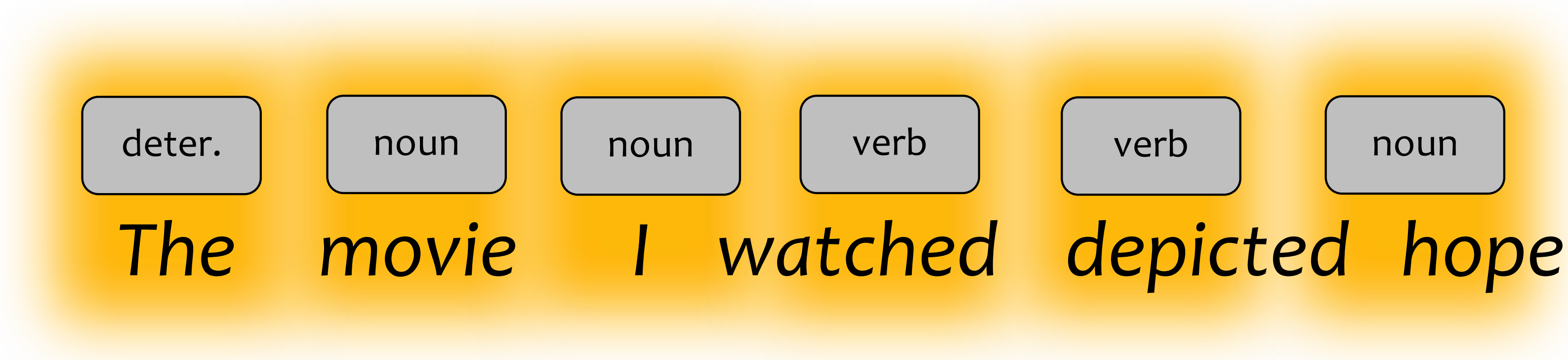
Where do features come from?



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
…	…	…	…	…	…	…

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 == "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
…	…	…	…	…	…	…

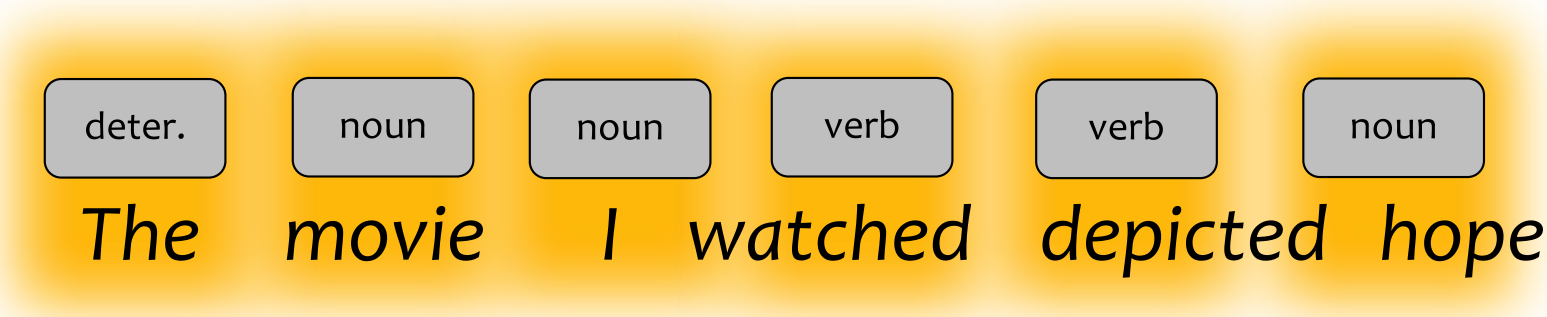
deter.	noun	noun	verb	verb	noun
The	*movie*	*I*	*watched*	*depicted*	*hope*

Table from Manning (2011)

Feature Engineering for NLP

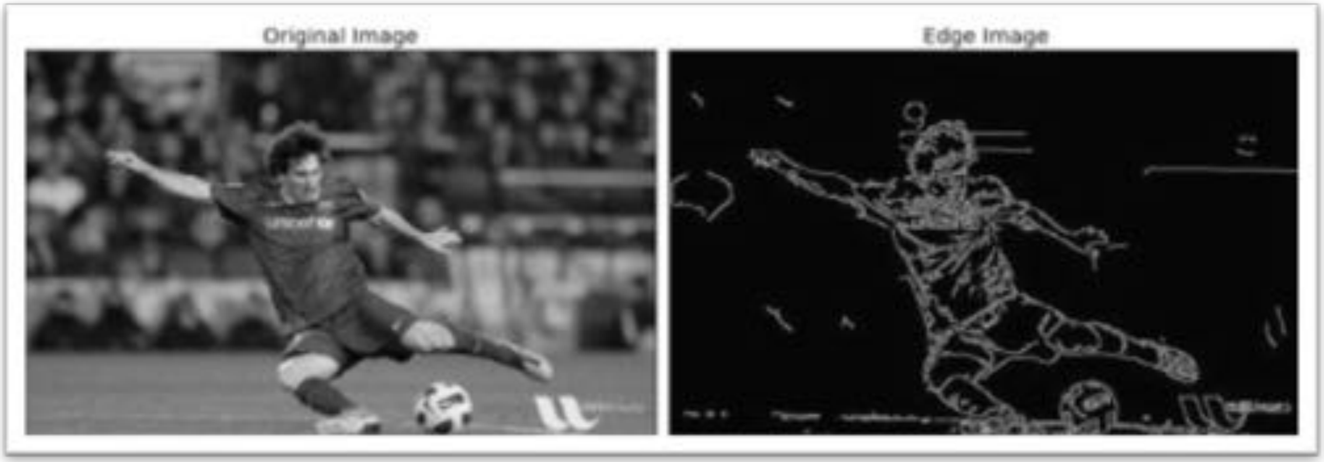
Table 3. Tagging accuracies with different feature templates and other changes on the *WSJ* 19-21 development set.

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%

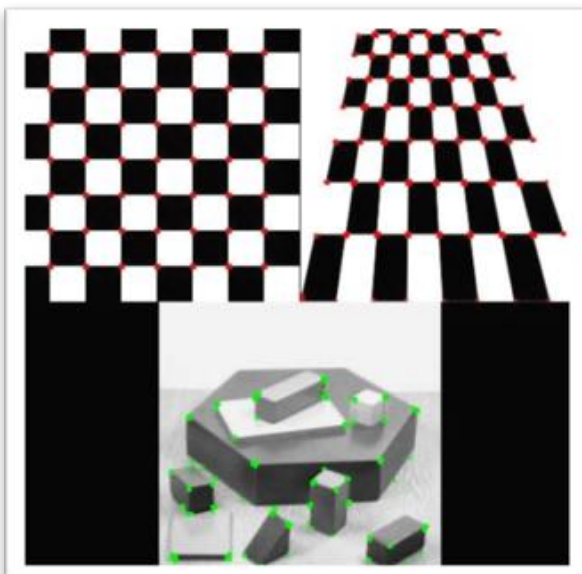


Feature Engineering for CV

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.

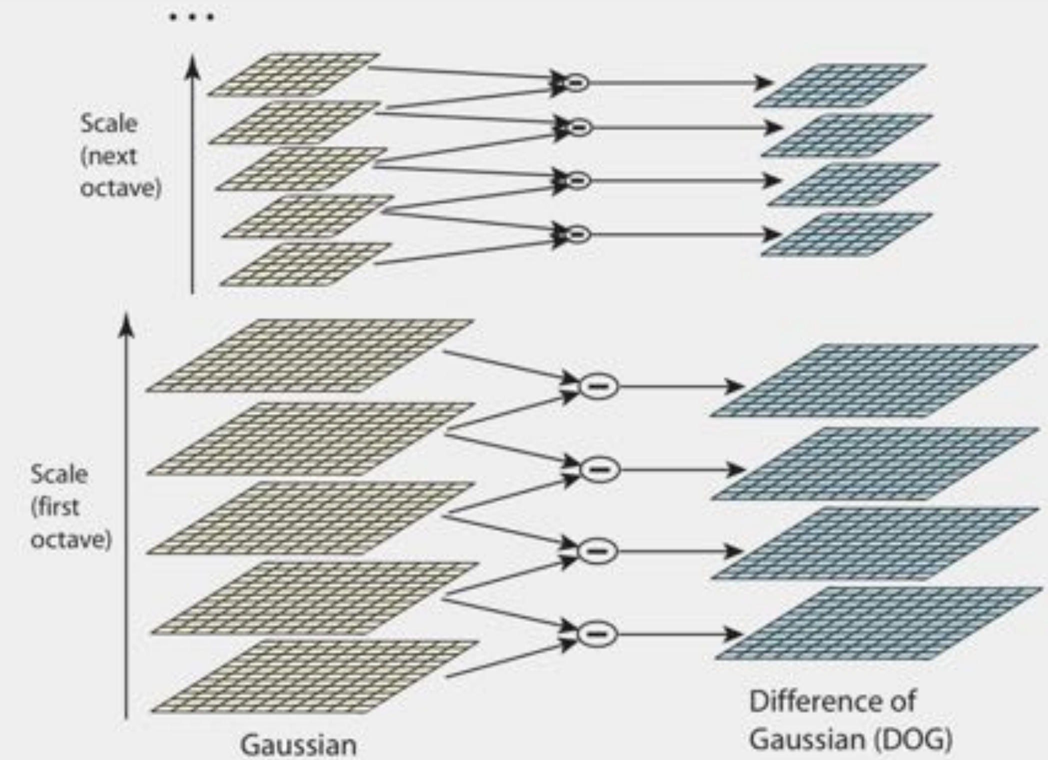


Figure 1: For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled 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: $\mathbf{x} \in \mathbb{R}^M$
  - new input: $\mathbf{x}' \in \mathbb{R}^{M'}$

    where $M' > M$ (usually)
  - define $\mathbf{x}' = b(\mathbf{x}) = [b_1(\mathbf{x}), b_2(\mathbf{x}), \ldots, b_{M'}(\mathbf{x})]$

    where $b_i : \mathbb{R}^M \rightarrow \mathbb{R}$ is any function
- **Examples:** (M = 1)

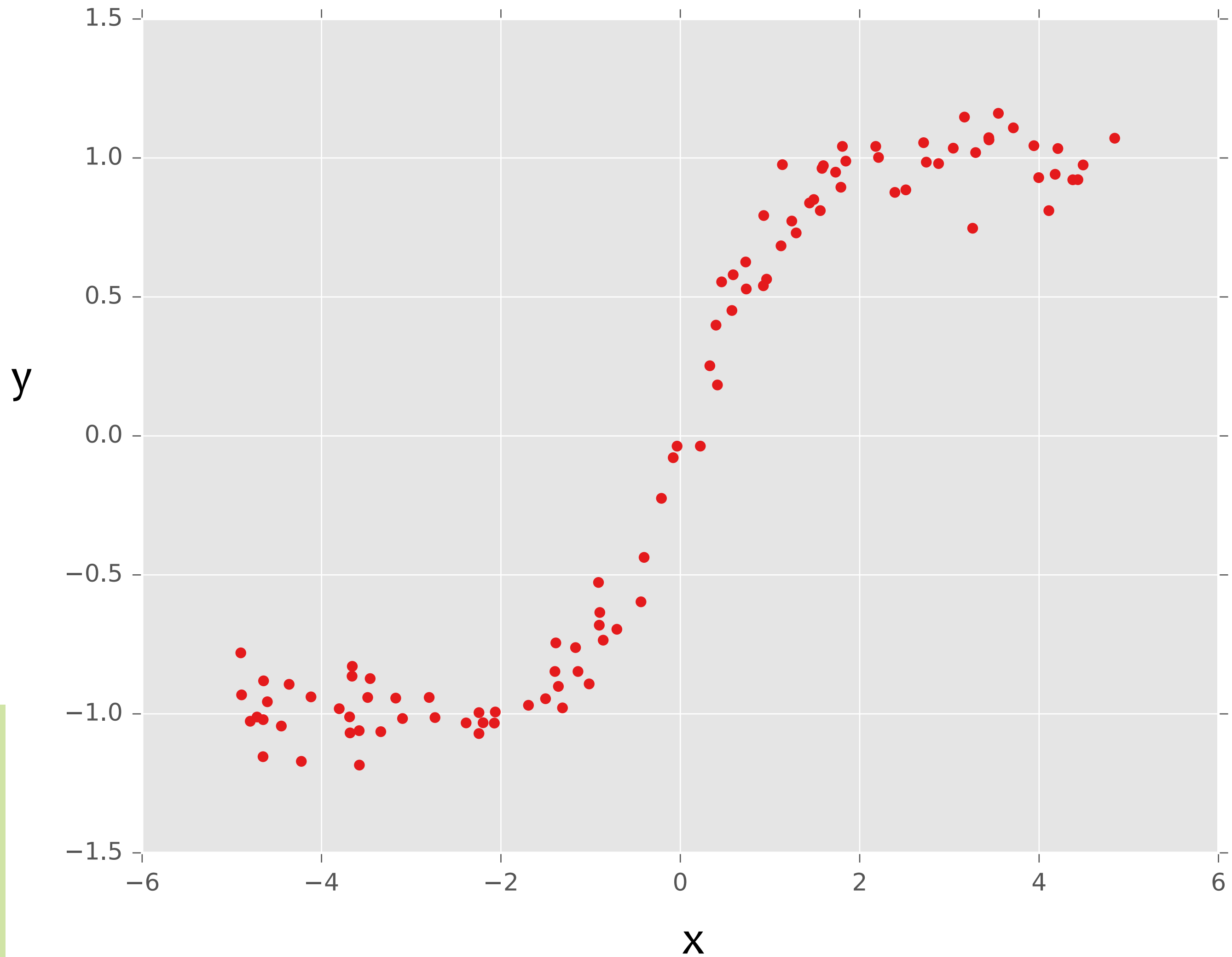
polynomial	$b_j(x) = x^j \quad \forall j \in \{1, \ldots, J\}$
radial basis function	$b_j(x) = \exp\left(\frac{-(x - \mu_j)^2}{2\sigma_j^2}\right)$
sigmoid	$b_j(x) = \frac{1}{1 + \exp(-\omega_j x)}$
log	$b_j(x) = \log(x)$

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

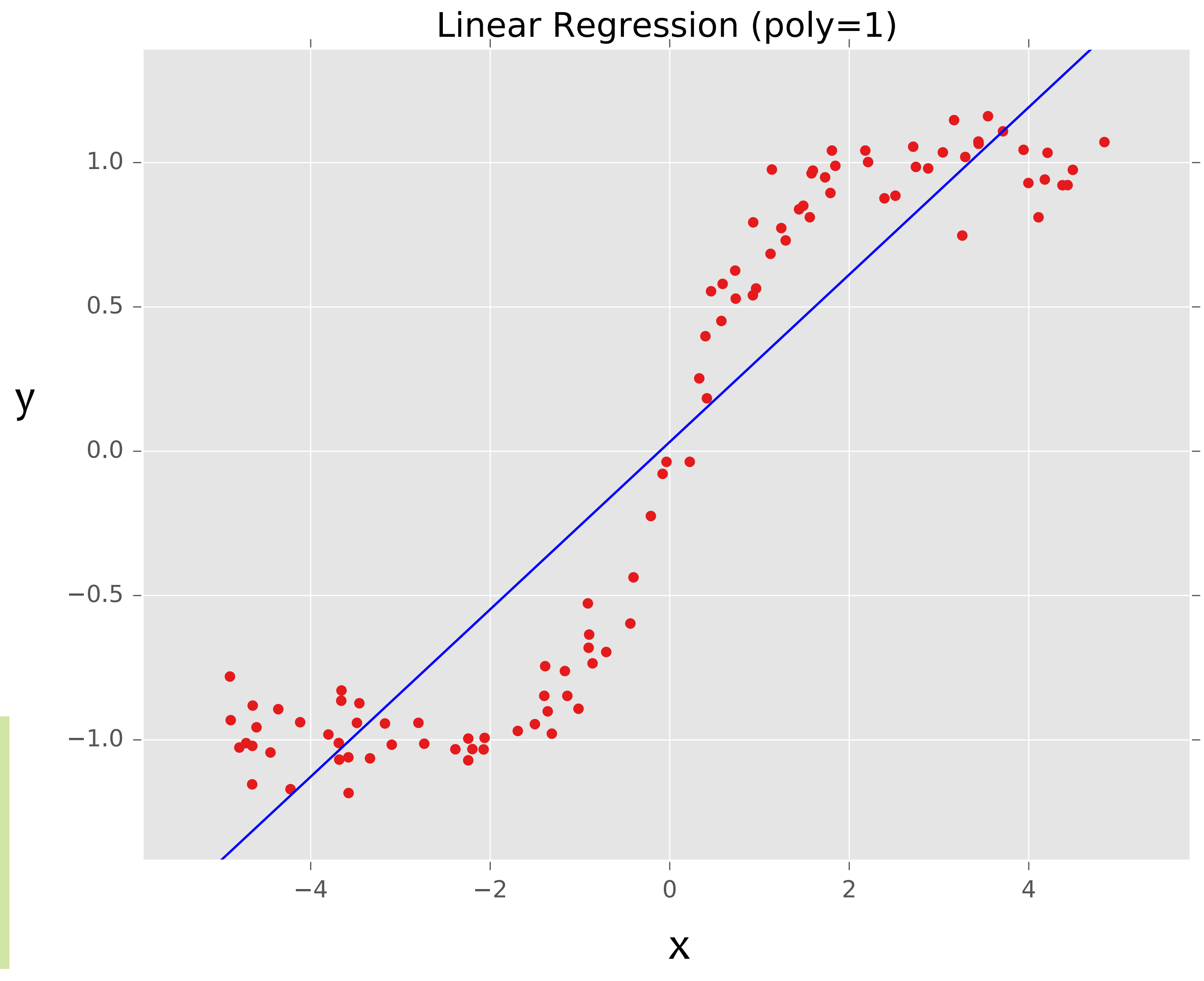
Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + b$ where f(.) is a polynomial basis function



true "unknown" target function is $y = \tanh(x) + \text{noise}$

Example: Linear Regression

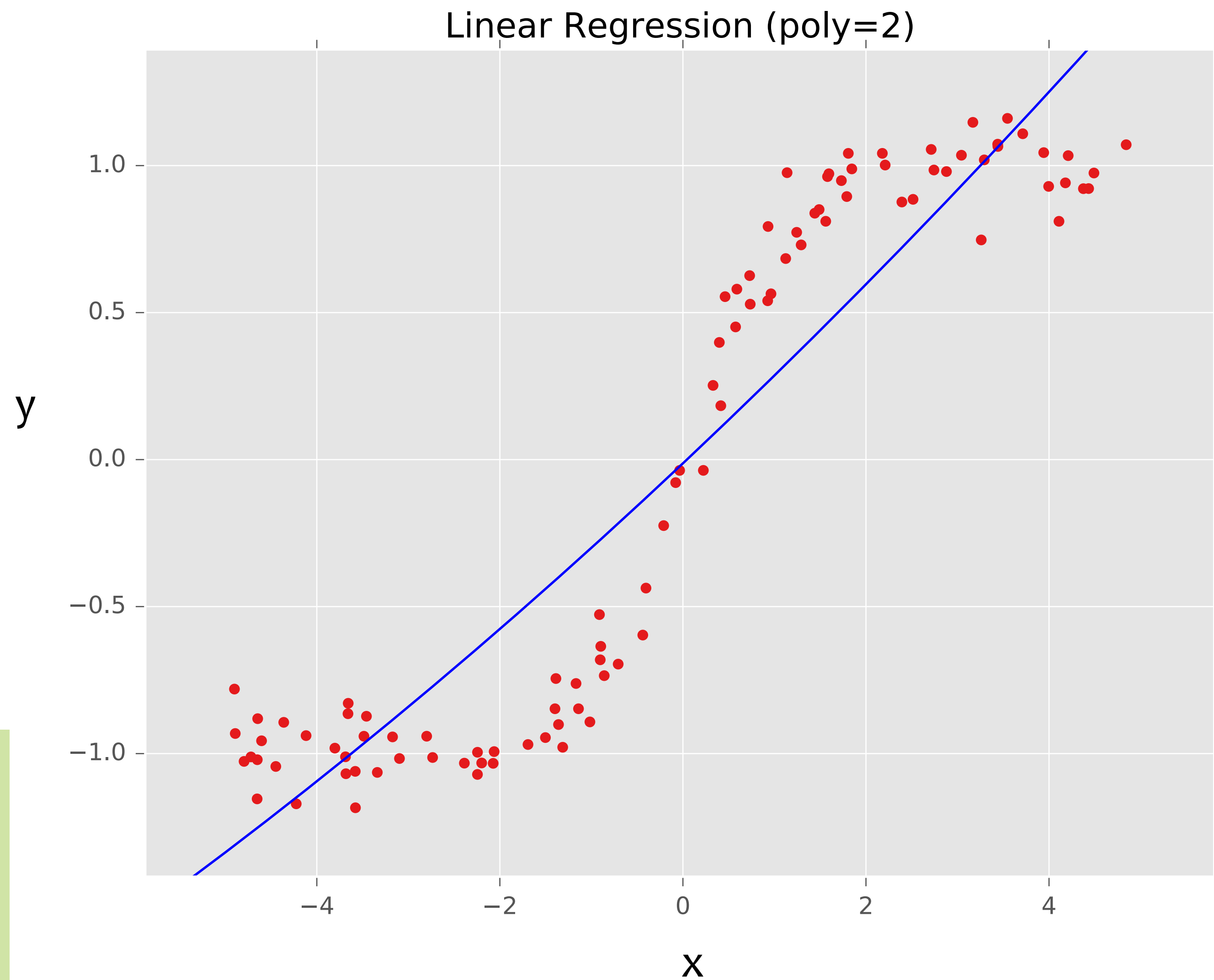
Goal: Learn y = $\mathbf{w}^T f(\mathbf{x}) + b$
where f(.) is a polynomial
basis function



true “unknown” target function is y = tanh(x) + noise

Example: Linear Regression

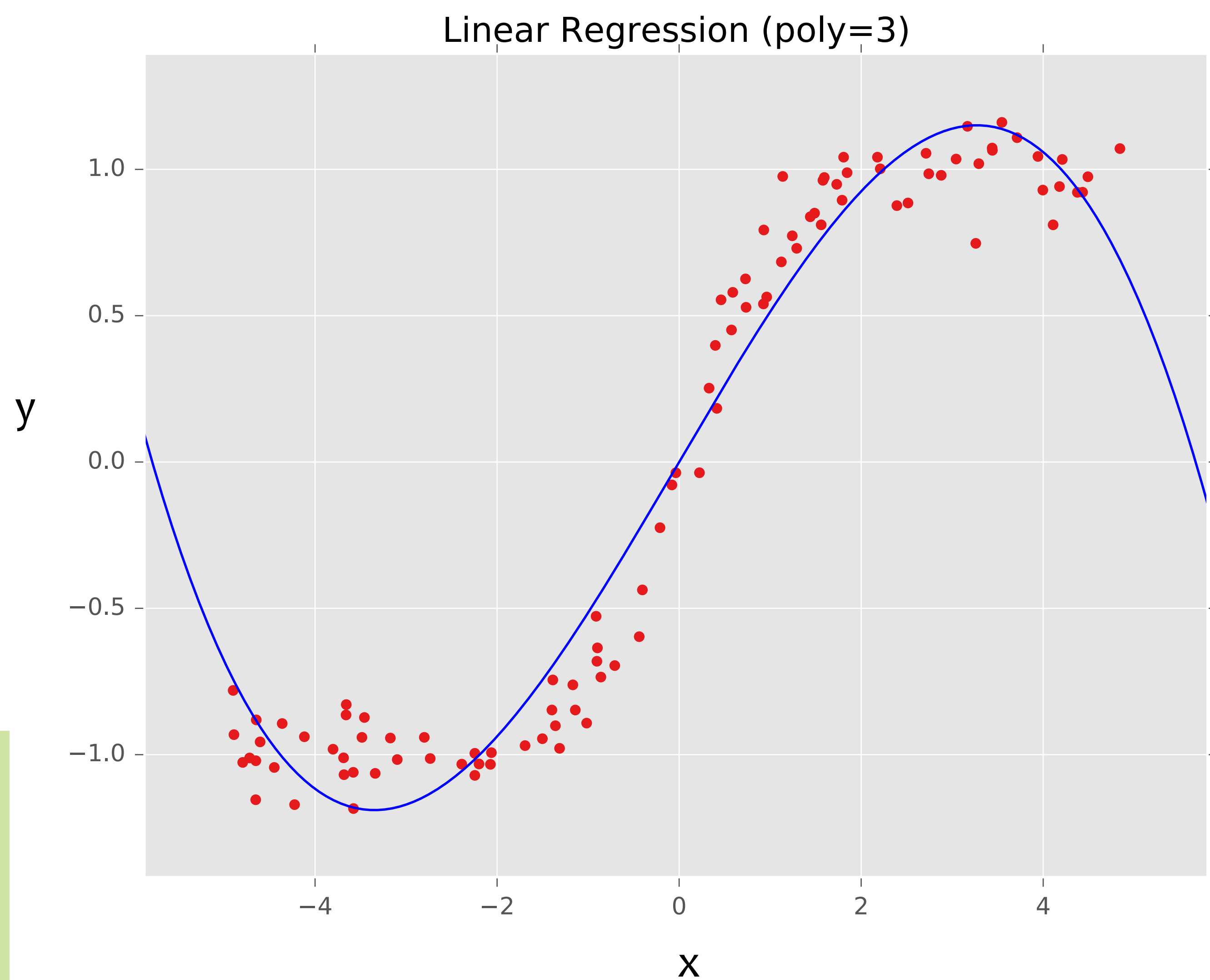
Goal: Learn y = $\mathbf{w}^T$ f($\mathbf{x}$) + b
where f(.) is a polynomial
basis function



true "unknown"
target function is
y = tanh(x) + noise

Example: Linear Regression

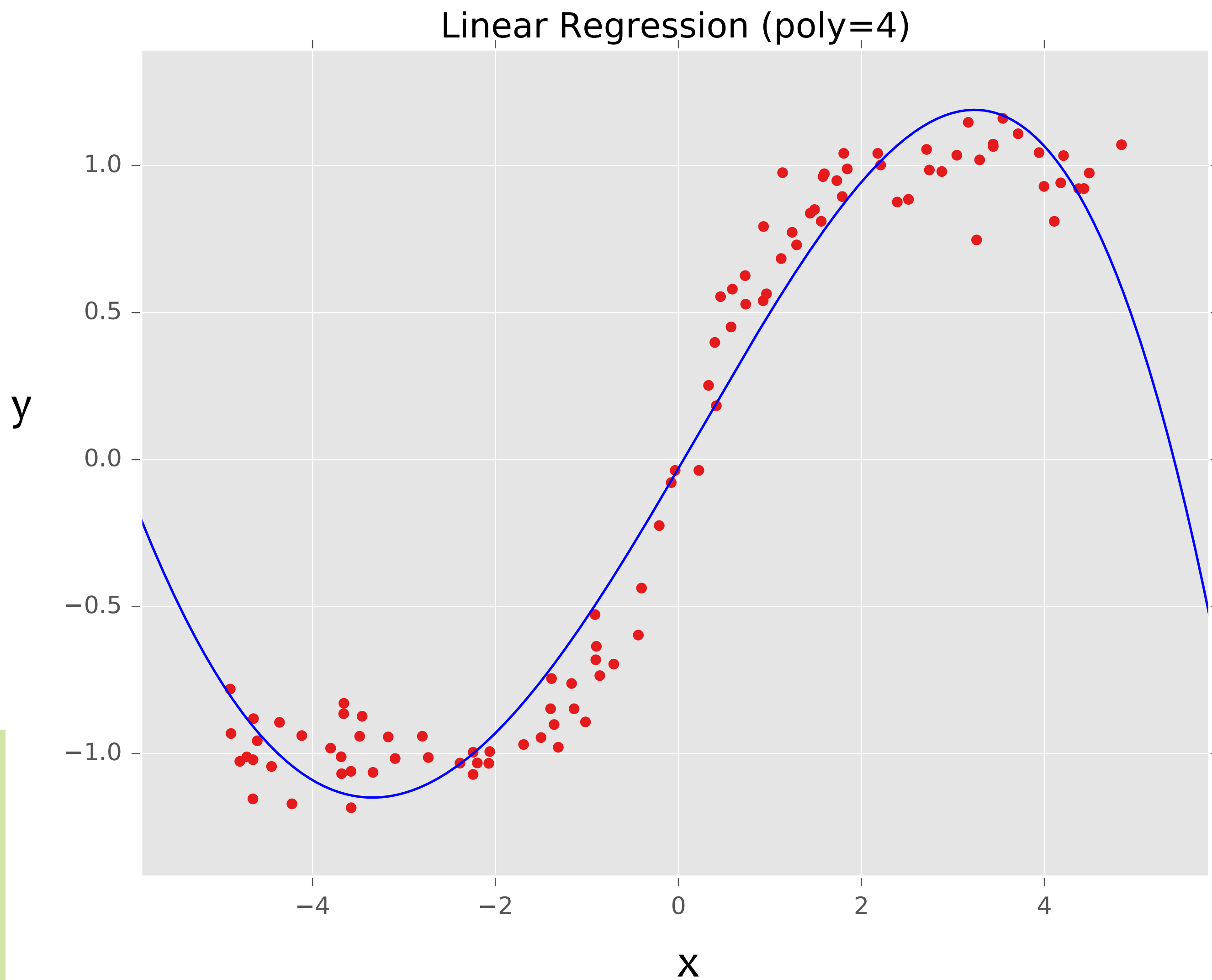
Goal: Learn y = $\mathbf{w}^T f(\mathbf{x}) + b$
where f(.) is a polynomial
basis function



true "unknown" target function is
y = tanh(x) + noise

Example: Linear Regression

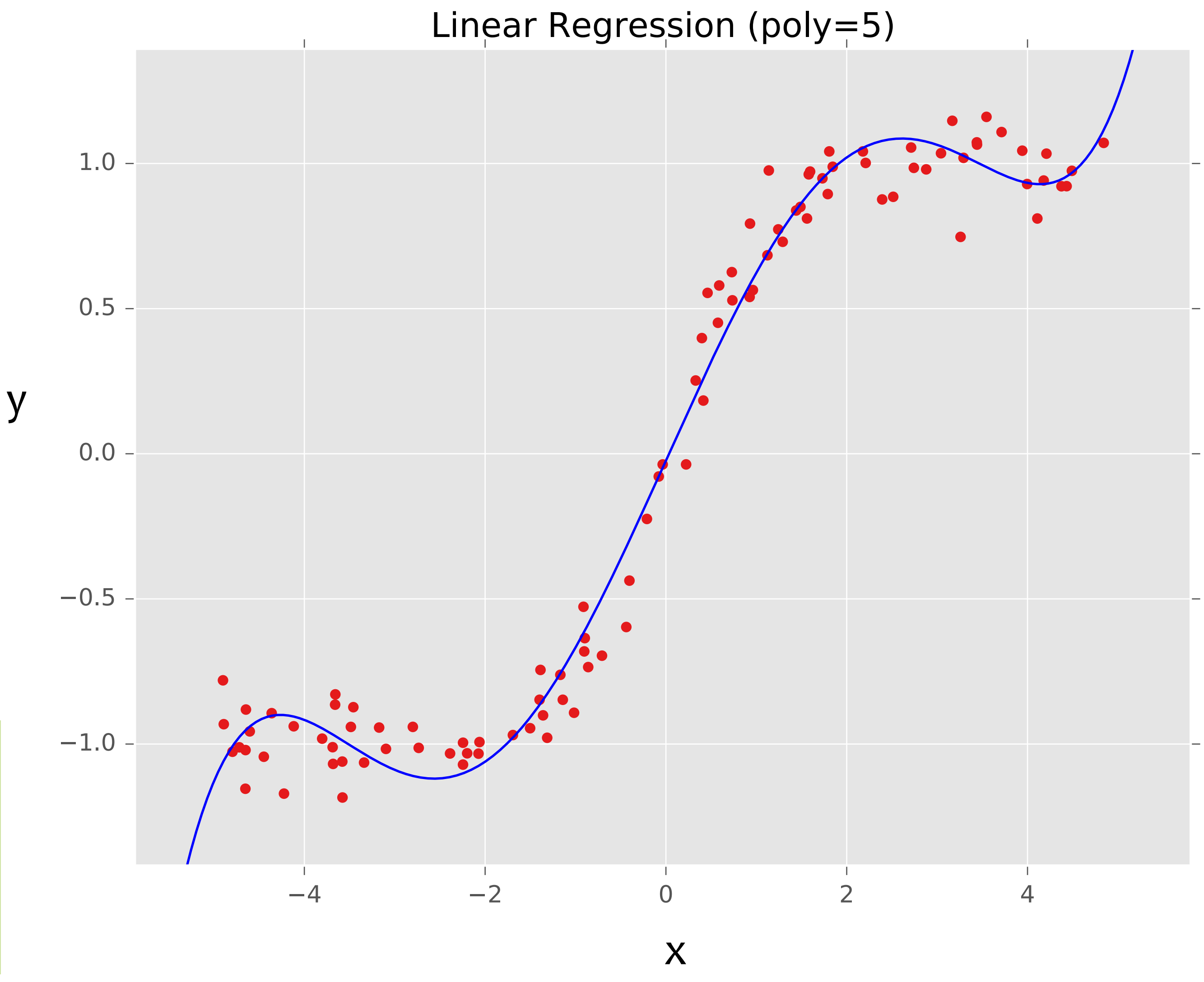
Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + b$
where f(.) is a polynomial
basis function



true "unknown" target function is
y = tanh(x) + noise

Example: Linear Regression

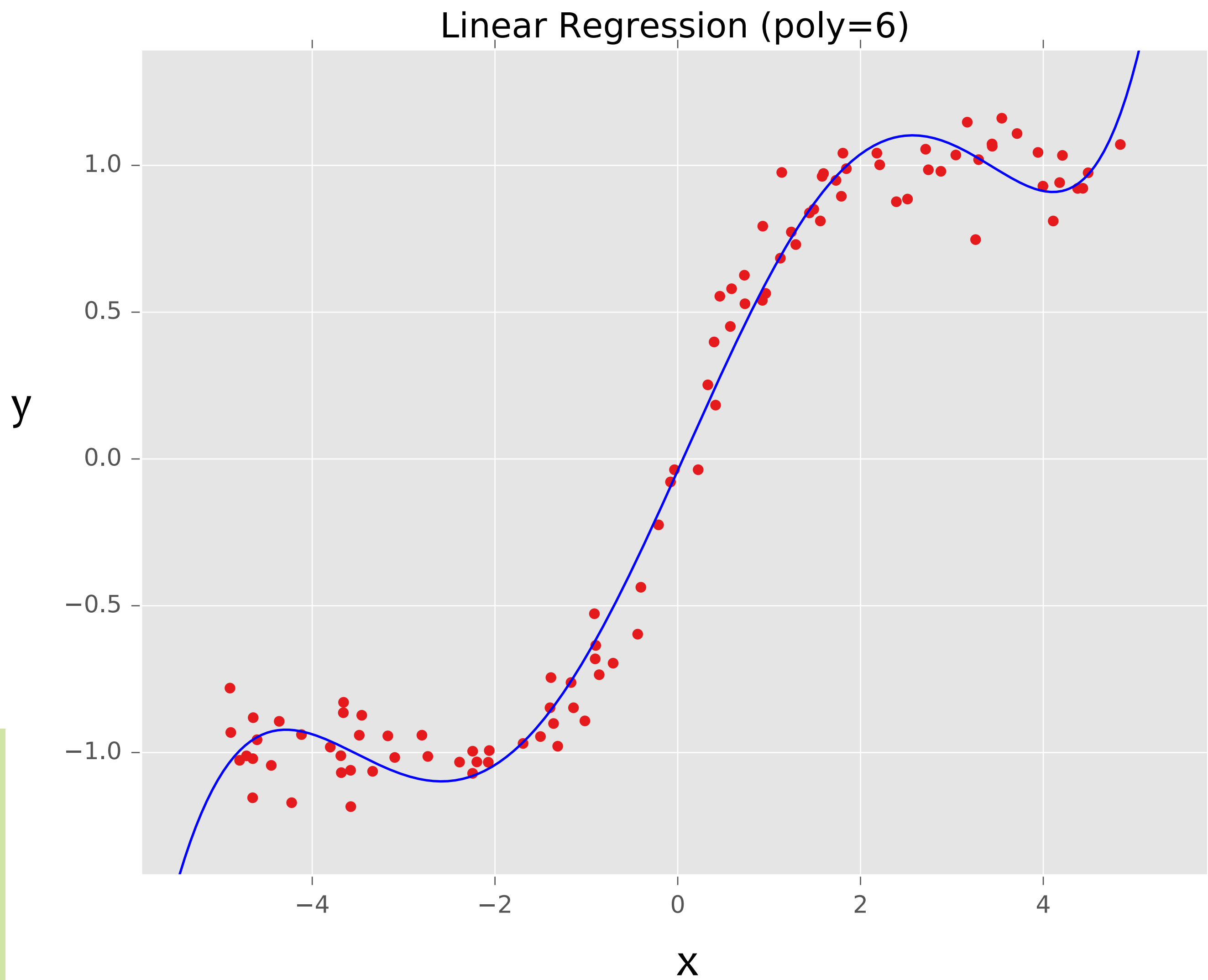
Goal: Learn y = $\mathbf{w}^T f(\mathbf{x}) + b$ where f(.) is a polynomial basis function



true "unknown" target function is y = tanh(x) + noise

Example: Linear Regression

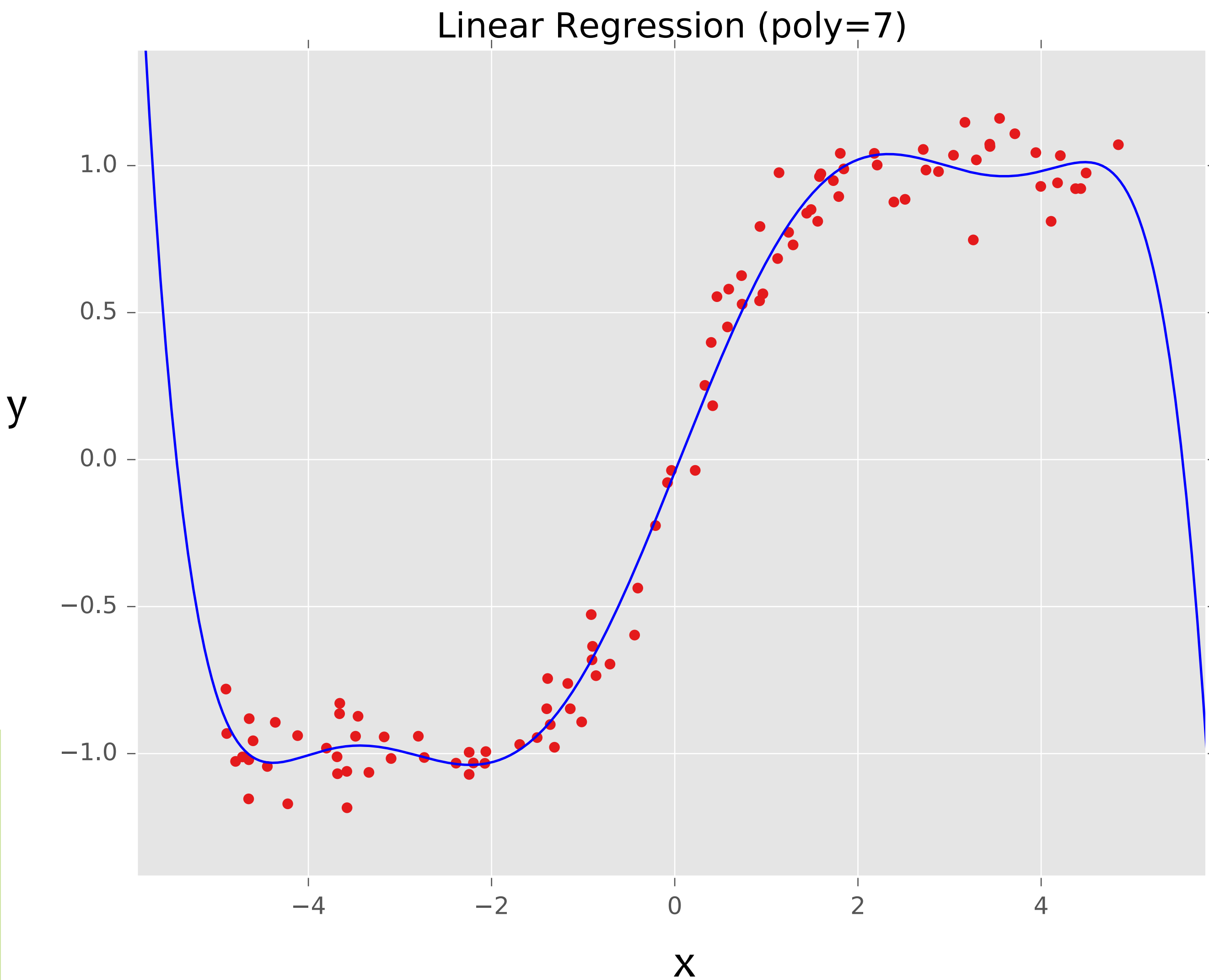
Goal: Learn y = $\mathbf{w}^T f(\mathbf{x}) + b$
where f(.) is a polynomial basis function



true "unknown" target function is
y = tanh(x) + noise

Example: Linear Regression

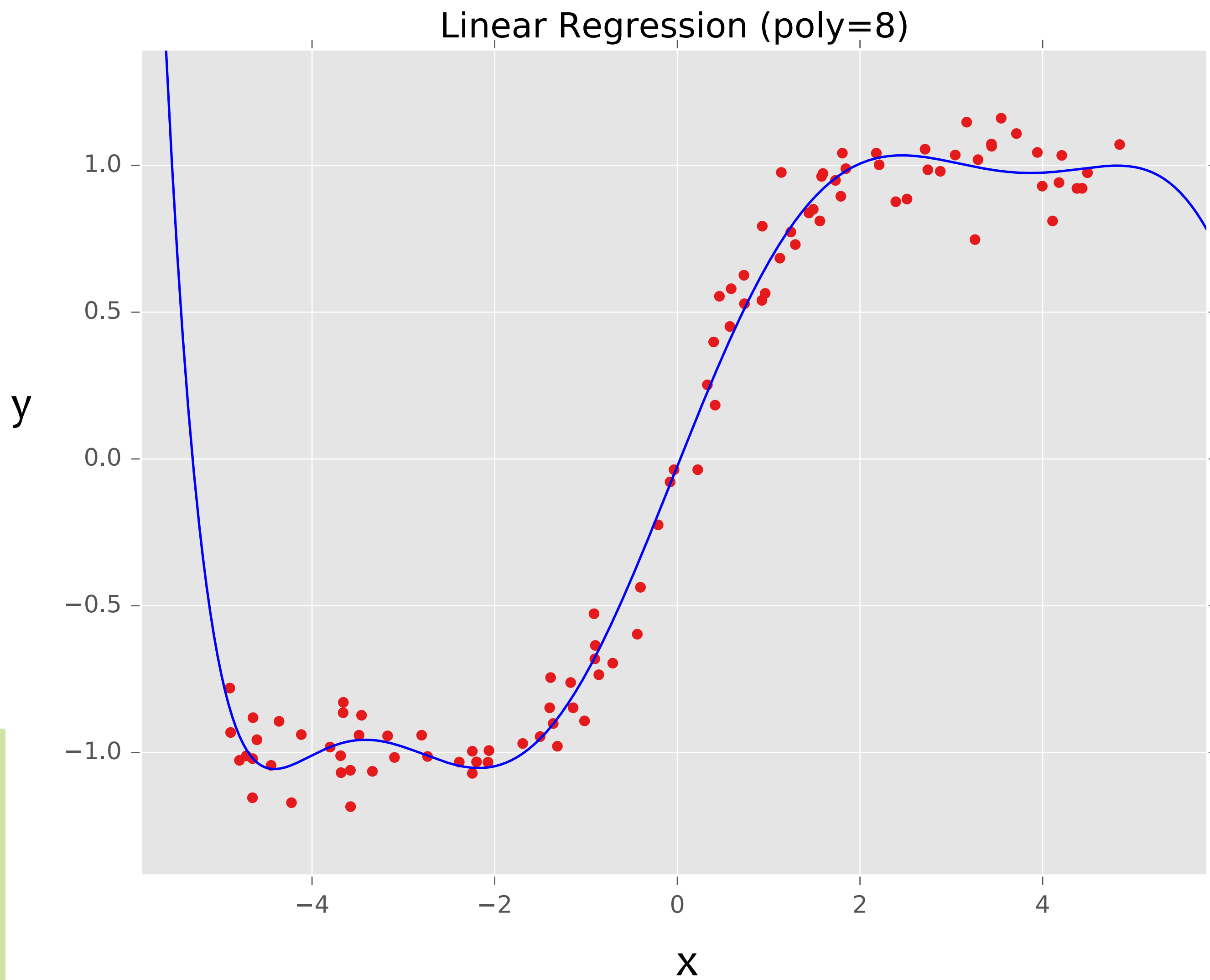
Goal: Learn y = $\mathbf{w}^T$ f($\mathbf{x}$) + b
where f(.) is a polynomial basis function



true "unknown" target function is
y = tanh(x) + noise

Example: Linear Regression

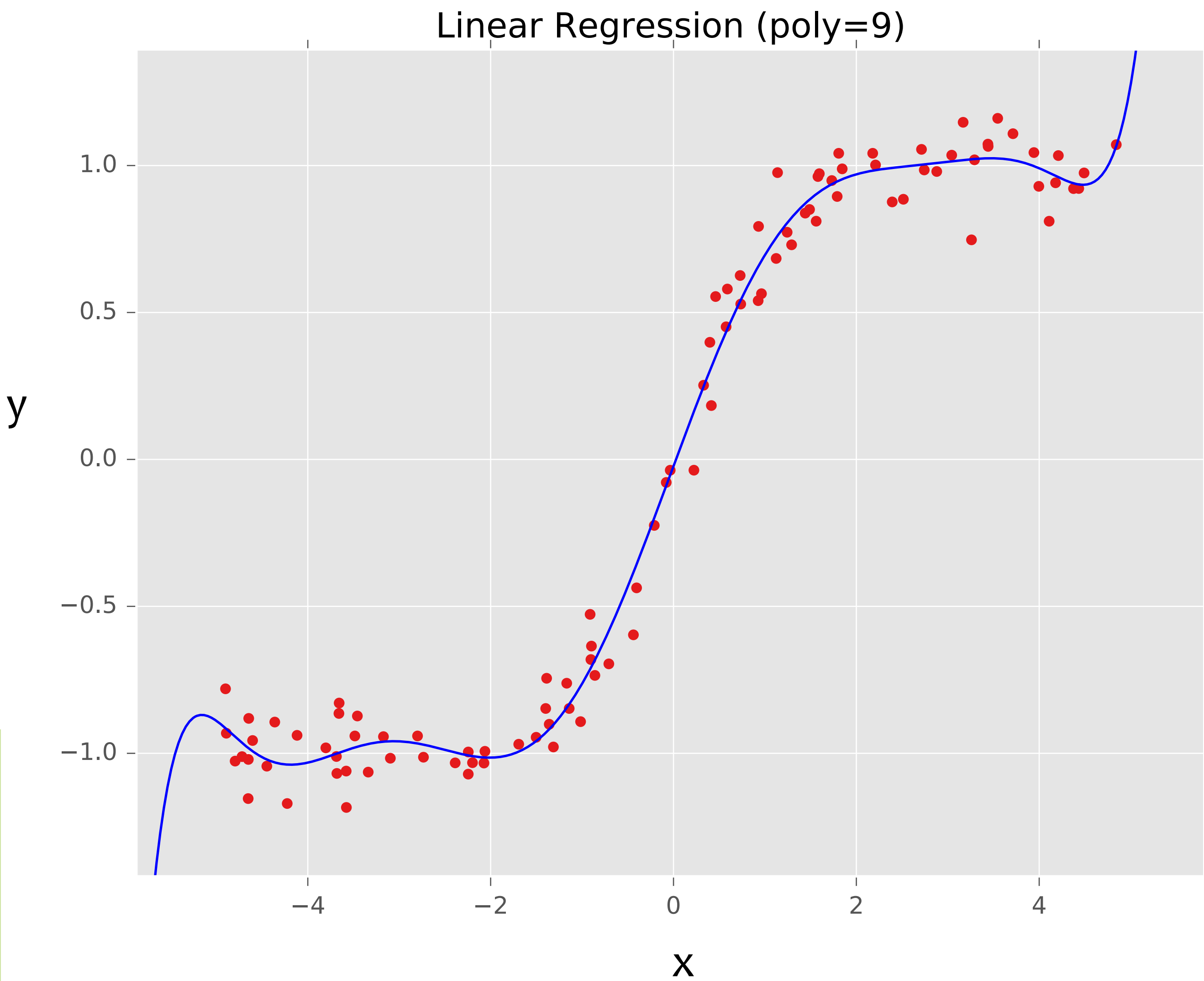
Goal: Learn y = $\mathbf{w}^T$ f($\mathbf{x}$) + b
where f(.) is a polynomial
basis function



true "unknown" target function is
y = tanh(x) + noise

Example: Linear Regression

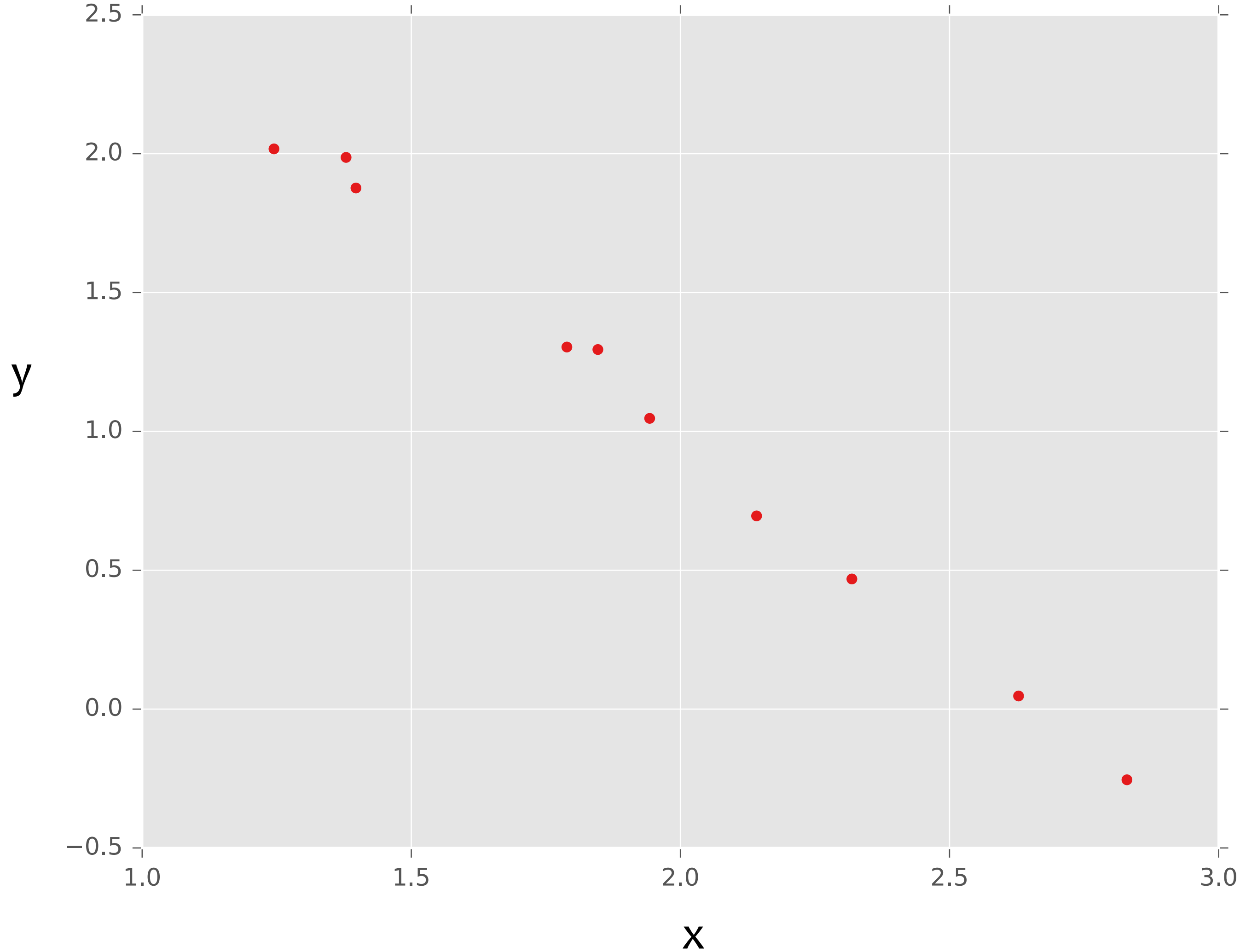
Goal: Learn y = $\mathbf{w}^T$ f($\mathbf{x}$) + b
where f(.) is a polynomial
basis function



true "unknown"
target function is
y = tanh(x) + noise

Example: Linear Regression

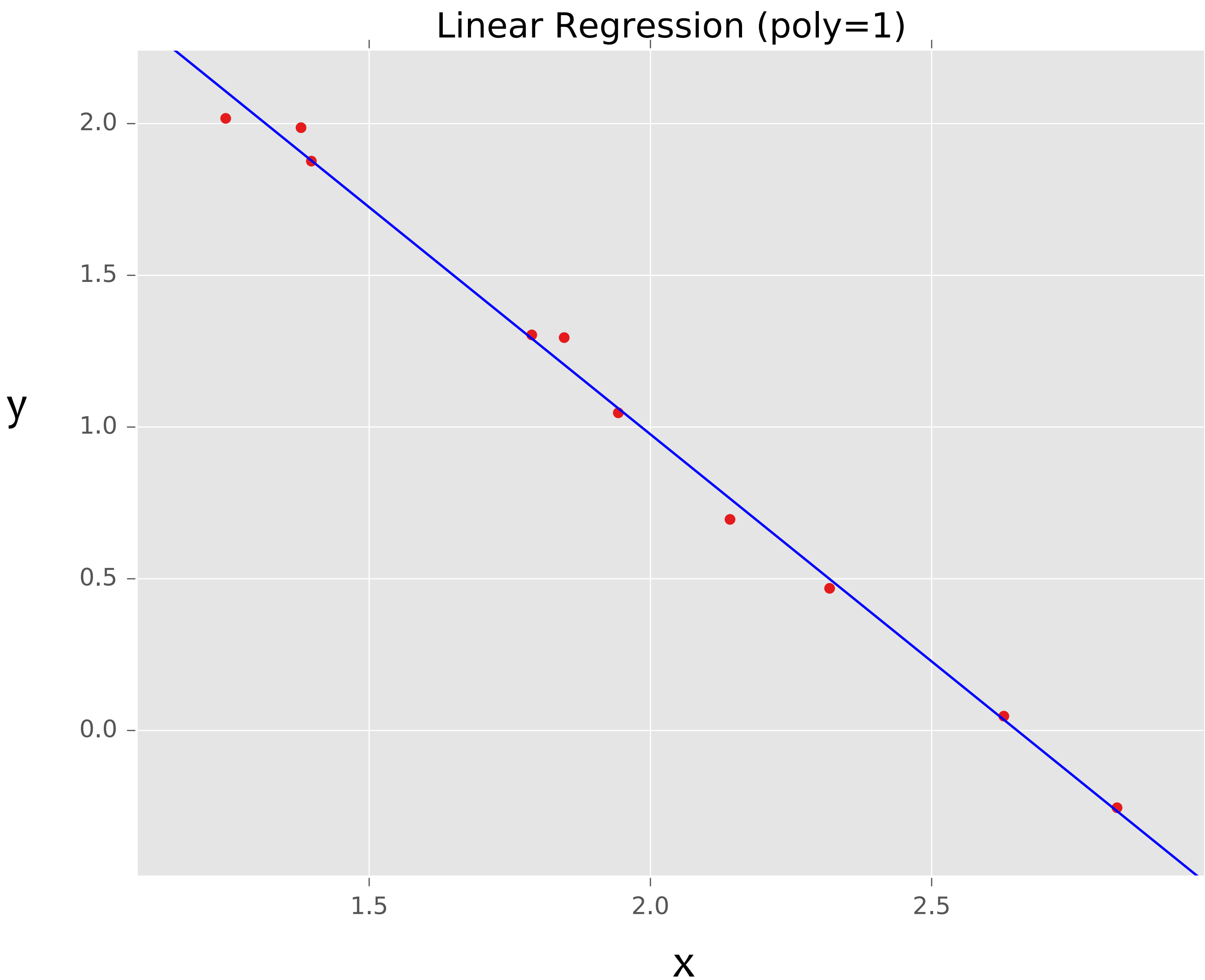
Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + b$ where f(.) is a polynomial basis function



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

Example: Linear Regression

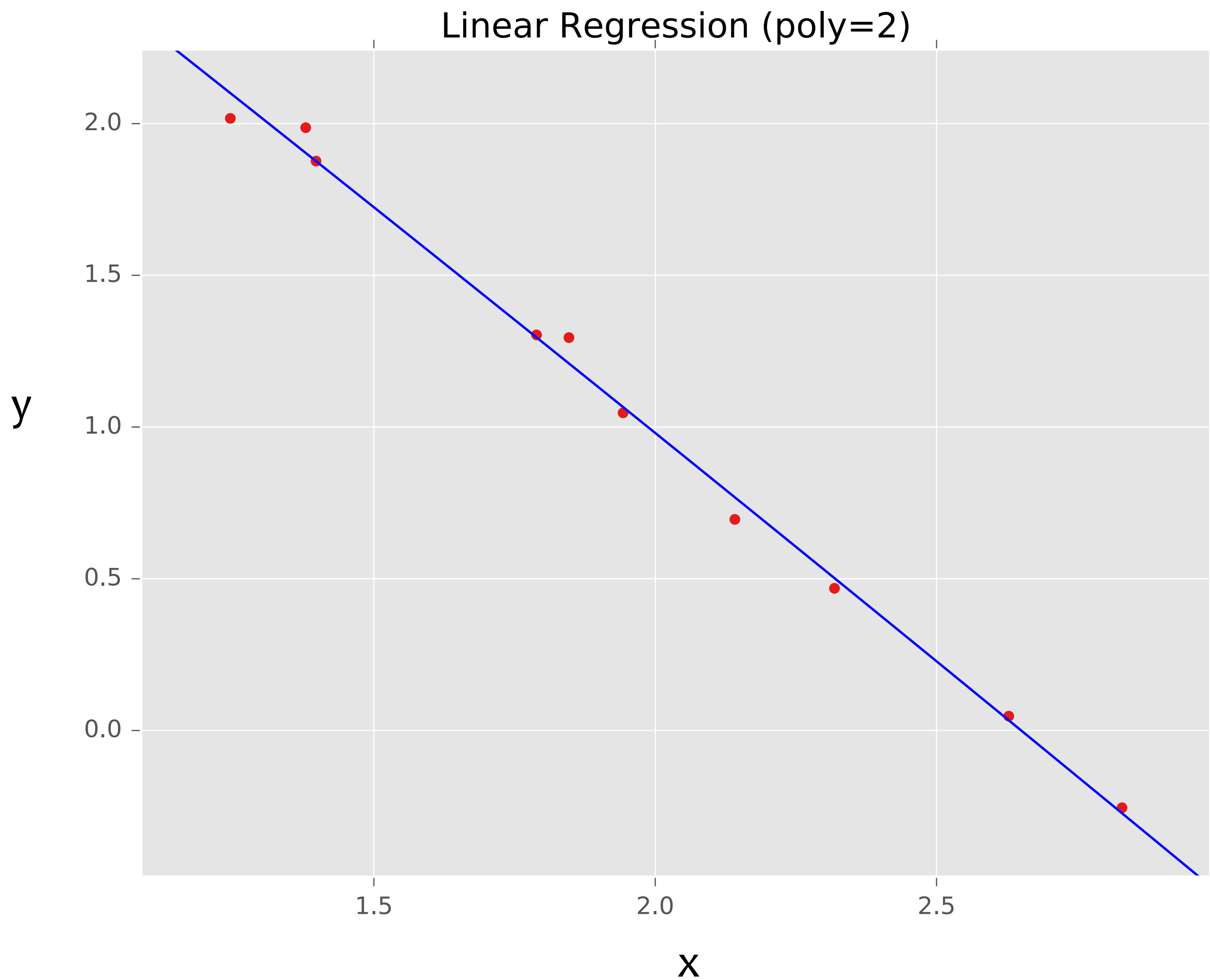
Goal: Learn y = $\mathbf{w}^T$ f($\mathbf{x}$) + b where f(.) is a polynomial basis function



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

Example: Linear Regression

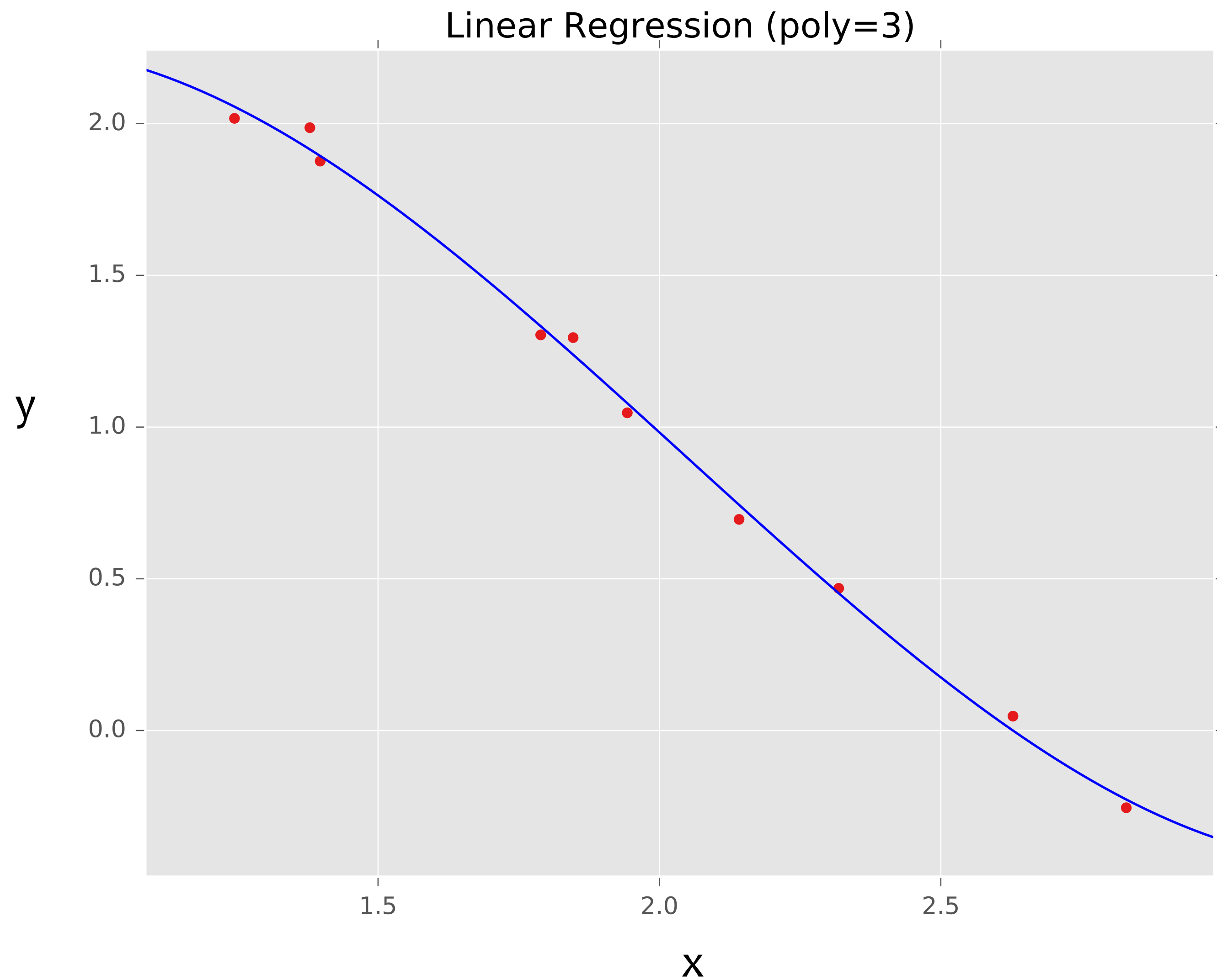
Goal: Learn y = $\mathbf{w}^T$ f($\mathbf{x}$) + b
where f(.) is a polynomial
basis function



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

Example: Linear Regression

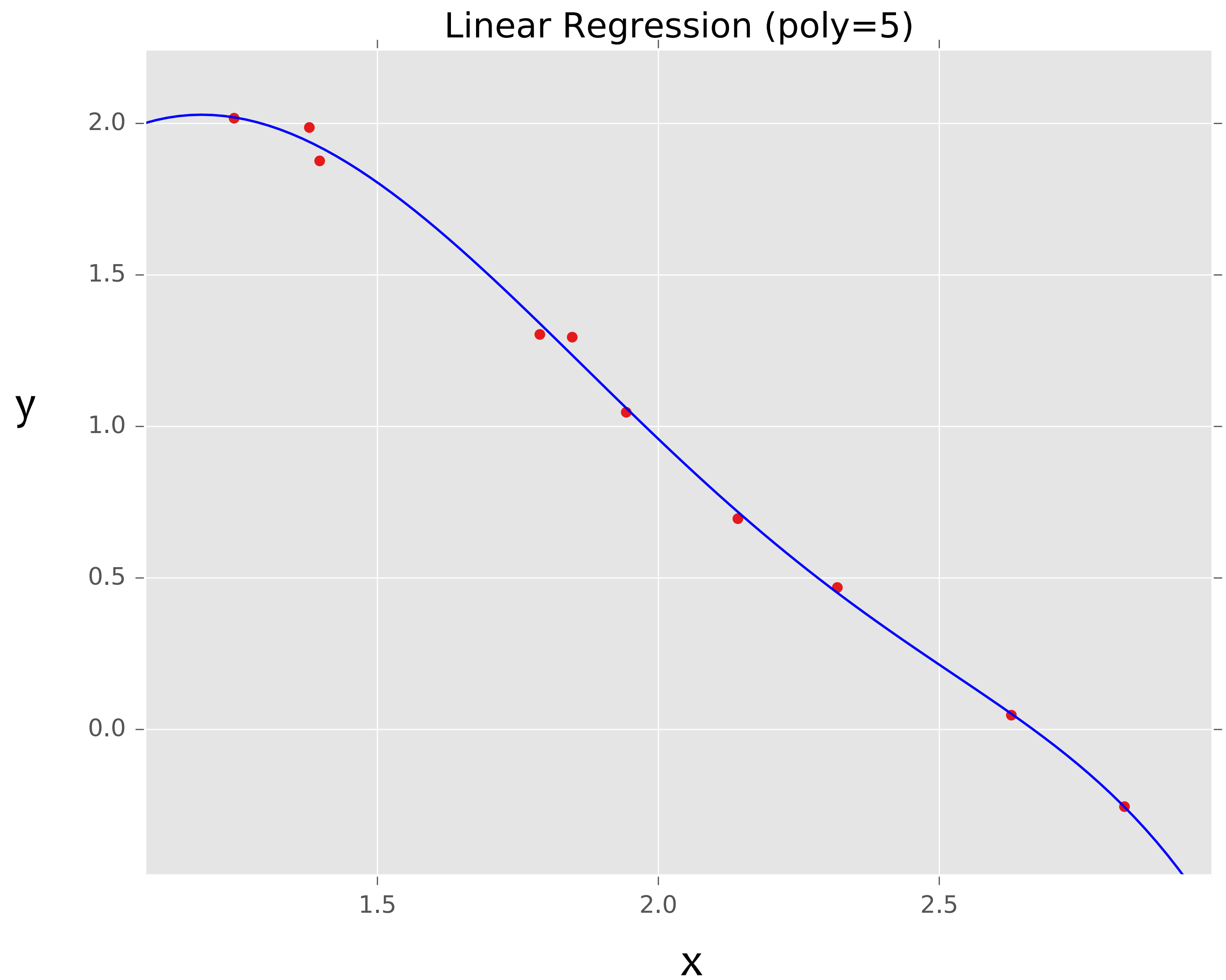
Goal: Learn y = $\mathbf{w}^T$ f($\mathbf{x}$) + b where f(.) is a polynomial basis function



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

Example: Linear Regression

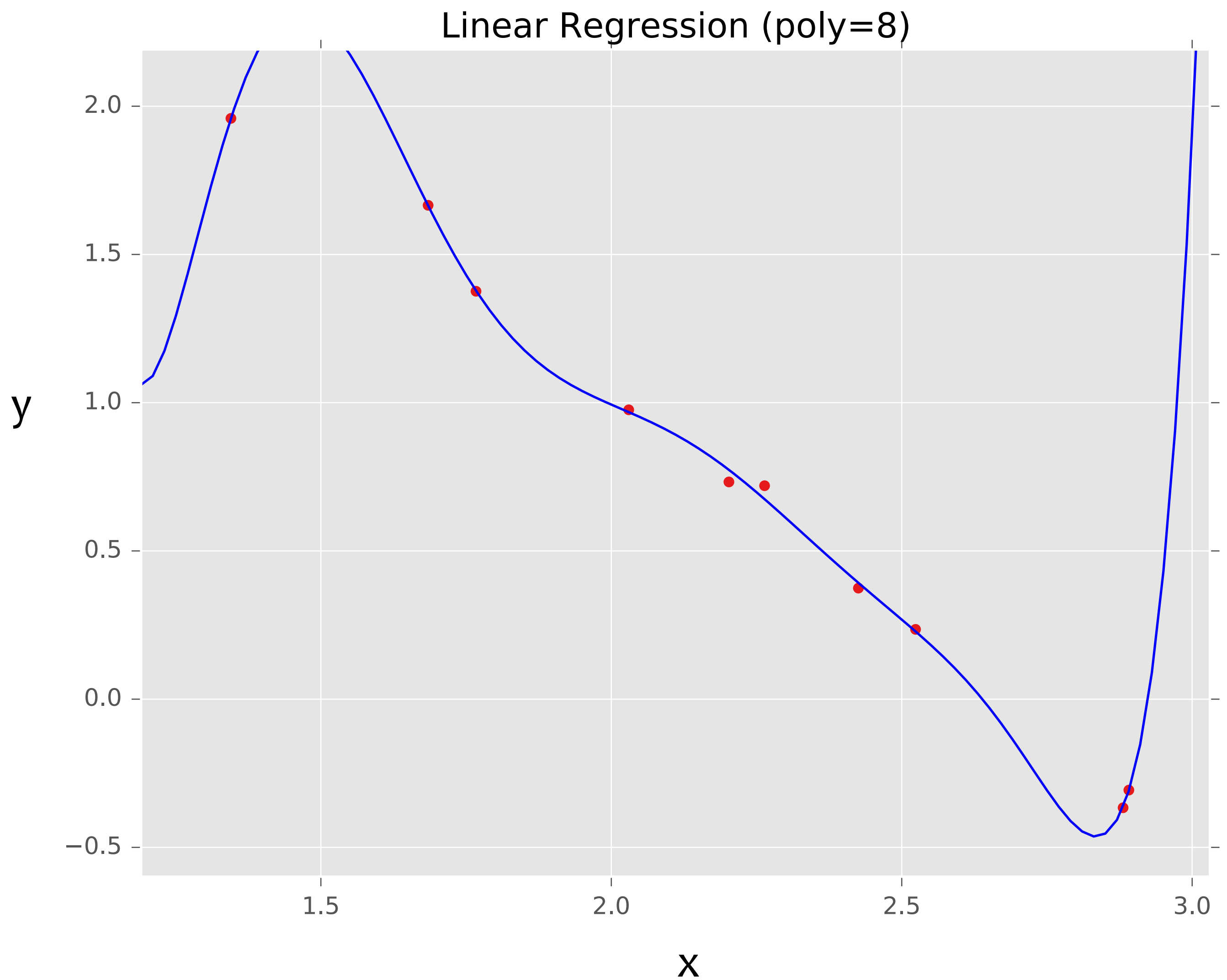
Goal: Learn y = $\mathbf{w}^T f(\mathbf{x}) + b$ where f(.) is a polynomial basis function



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

Example: Linear Regression

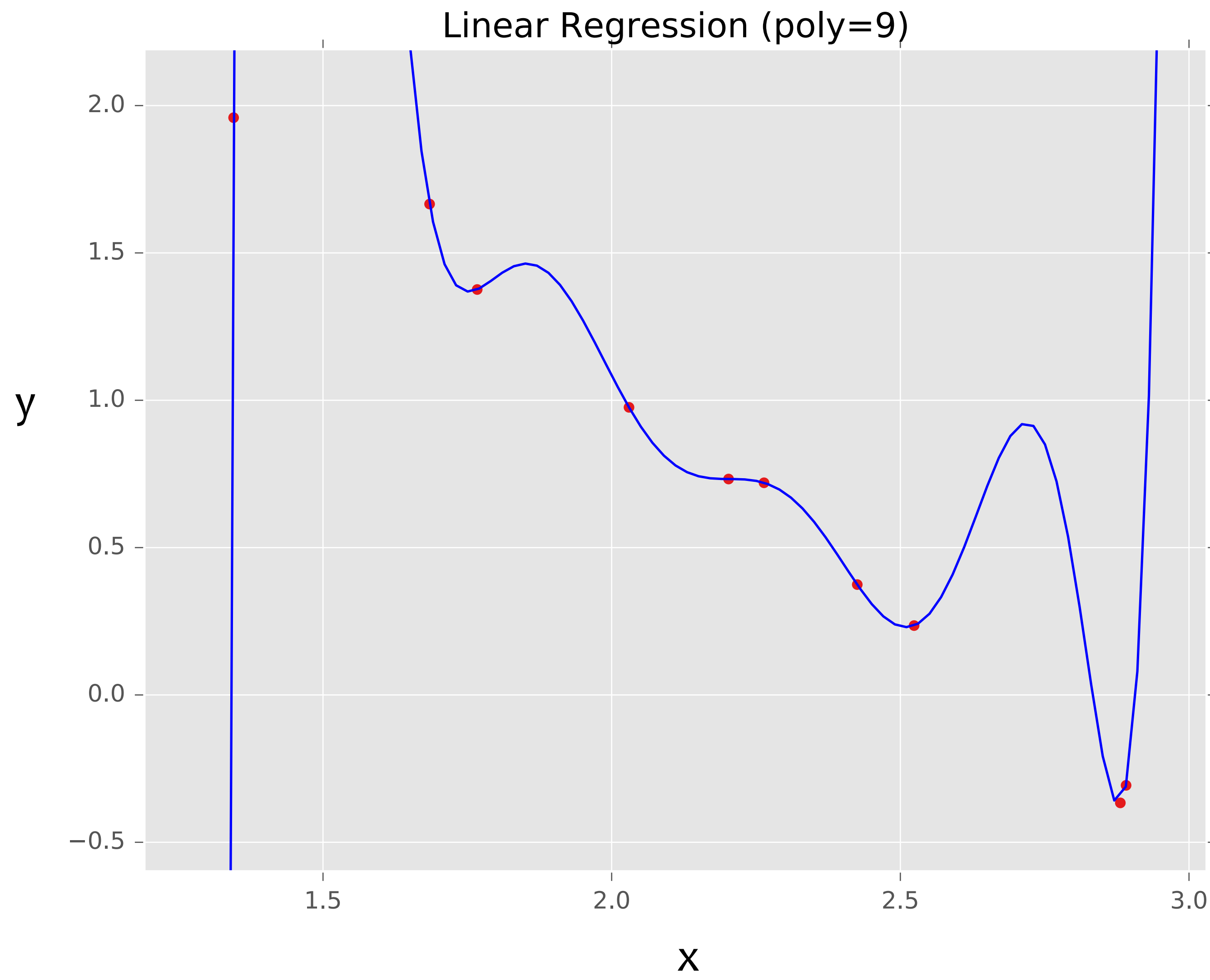
Goal: Learn y = $\mathbf{w}^T f(\mathbf{x}) + b$ where f(.) is a polynomial basis function



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

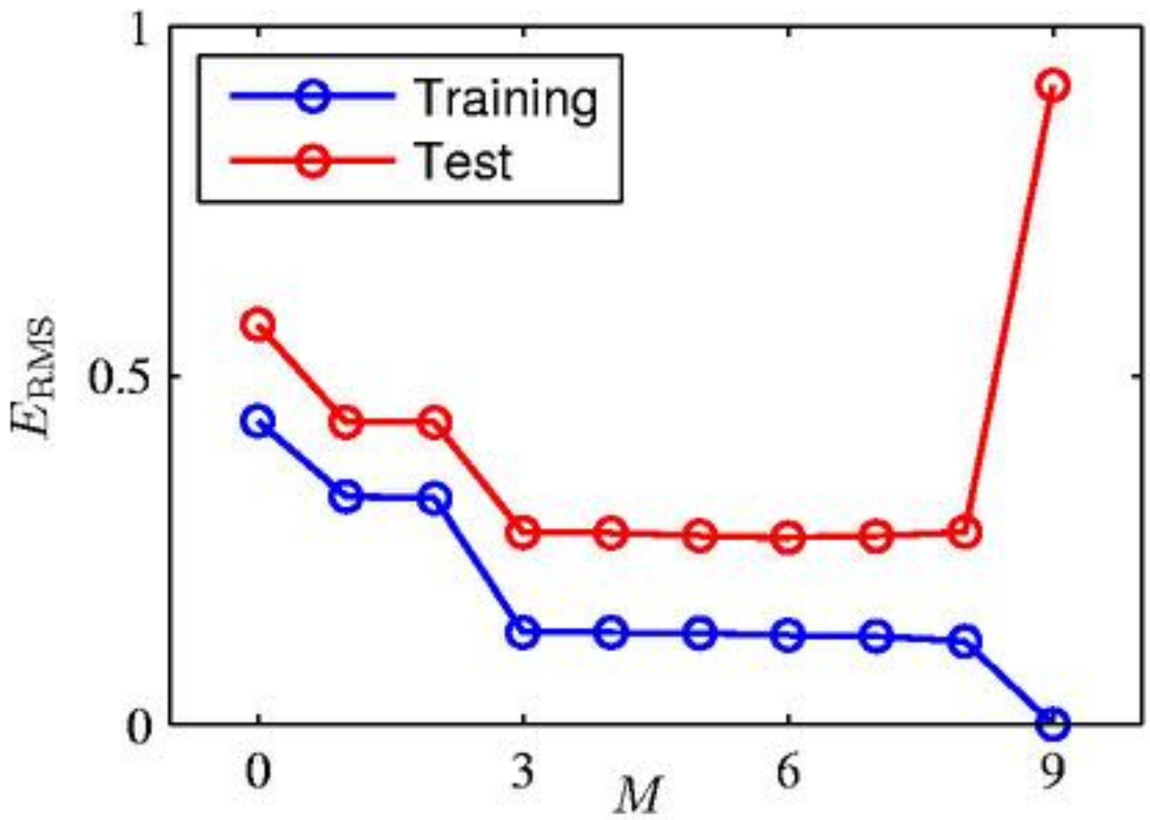
Example: Linear Regression

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

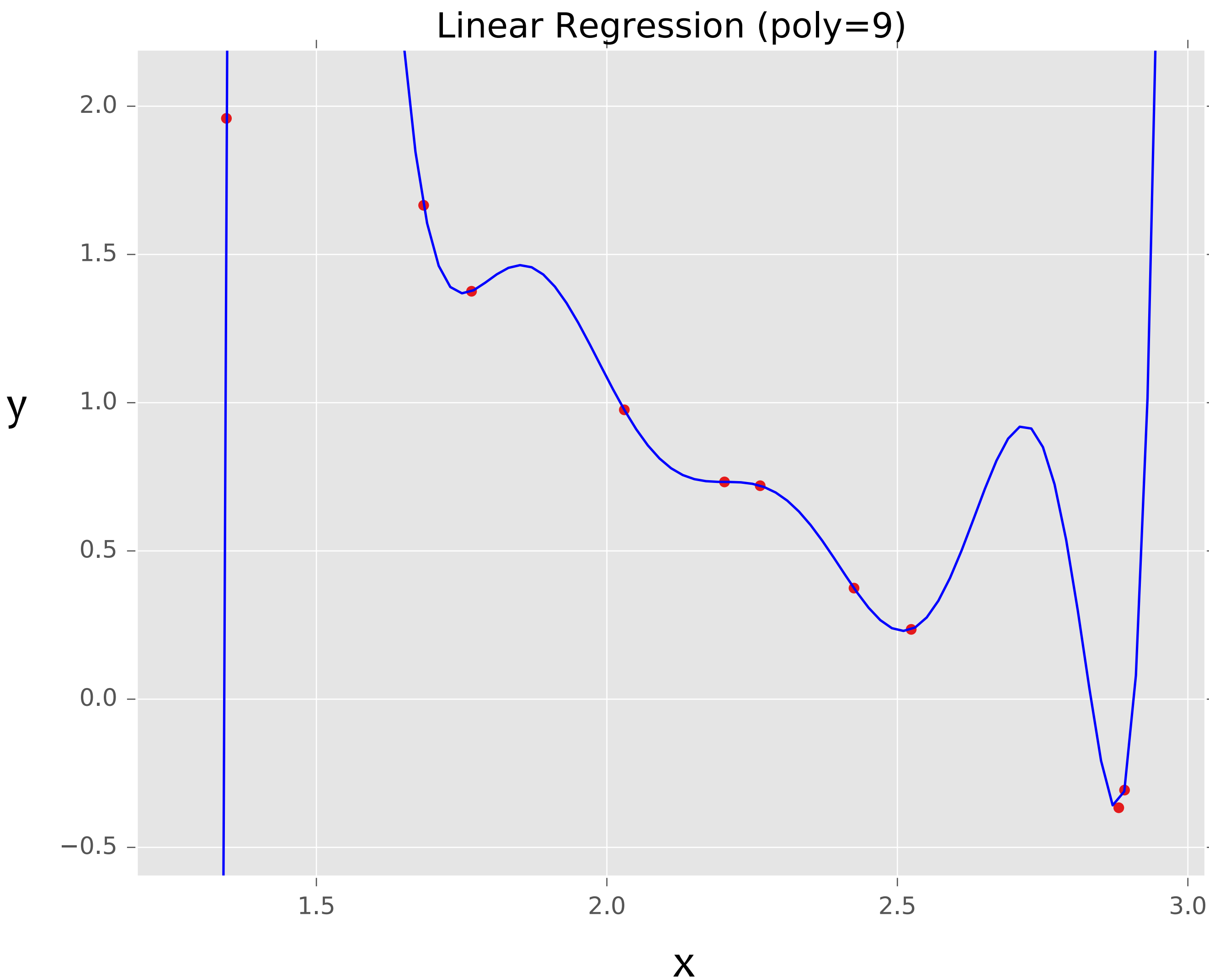
Polynomial Coefficients

	$M=0$	$M=1$	$M=3$	$M=9$
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

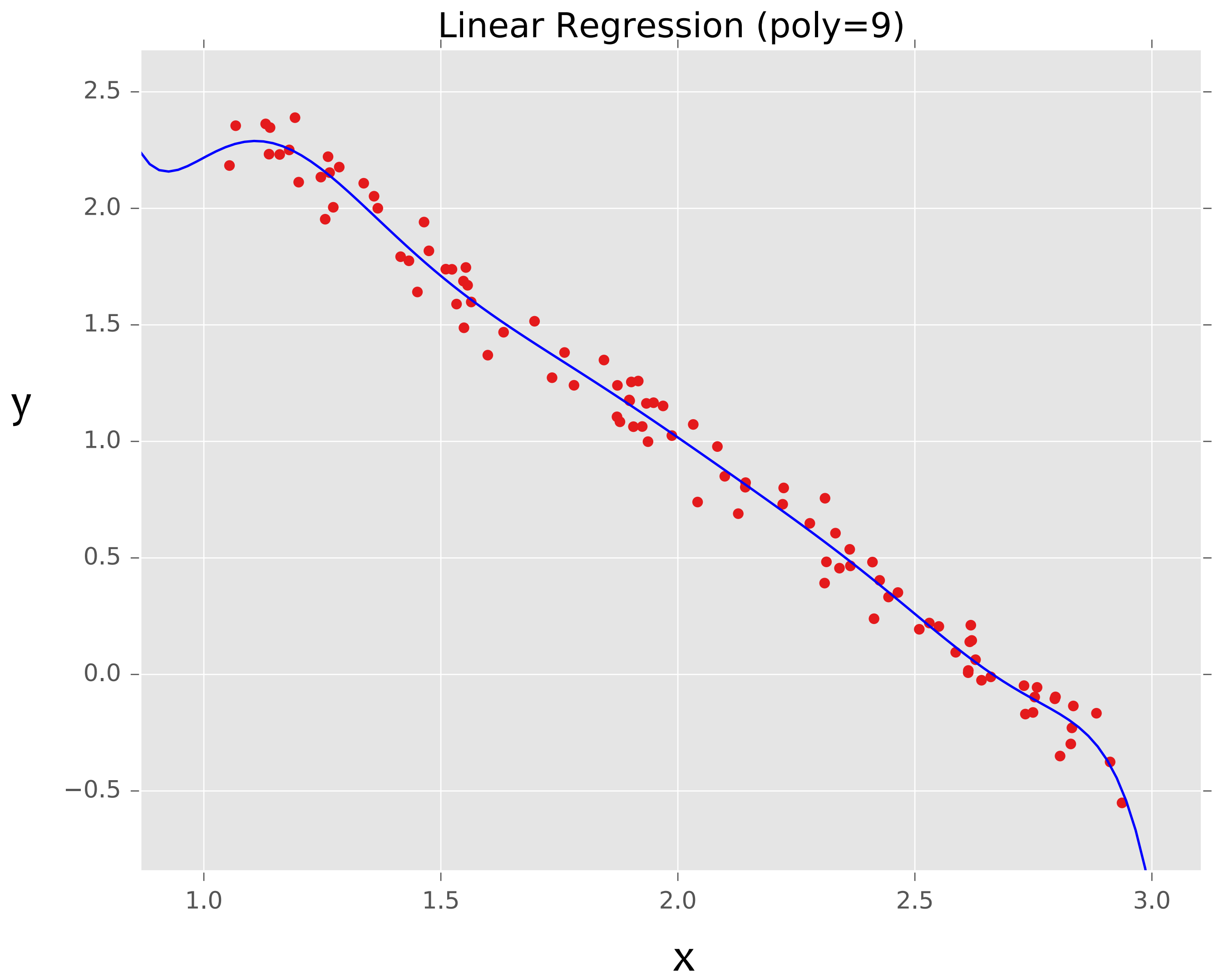


true “unknown” target function is linear with negative slope and gaussian noise

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:

- KNN (e.g. when k is small)
- Naïve Bayes (e.g. without a prior)
- Linear Regression (e.g. with basis function)
- 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

Chalkboard

– L2, L1, L0 Regularization
– Example: Linear Regression

Regularization

Don't Regularize the Bias (Intercept) Parameter!

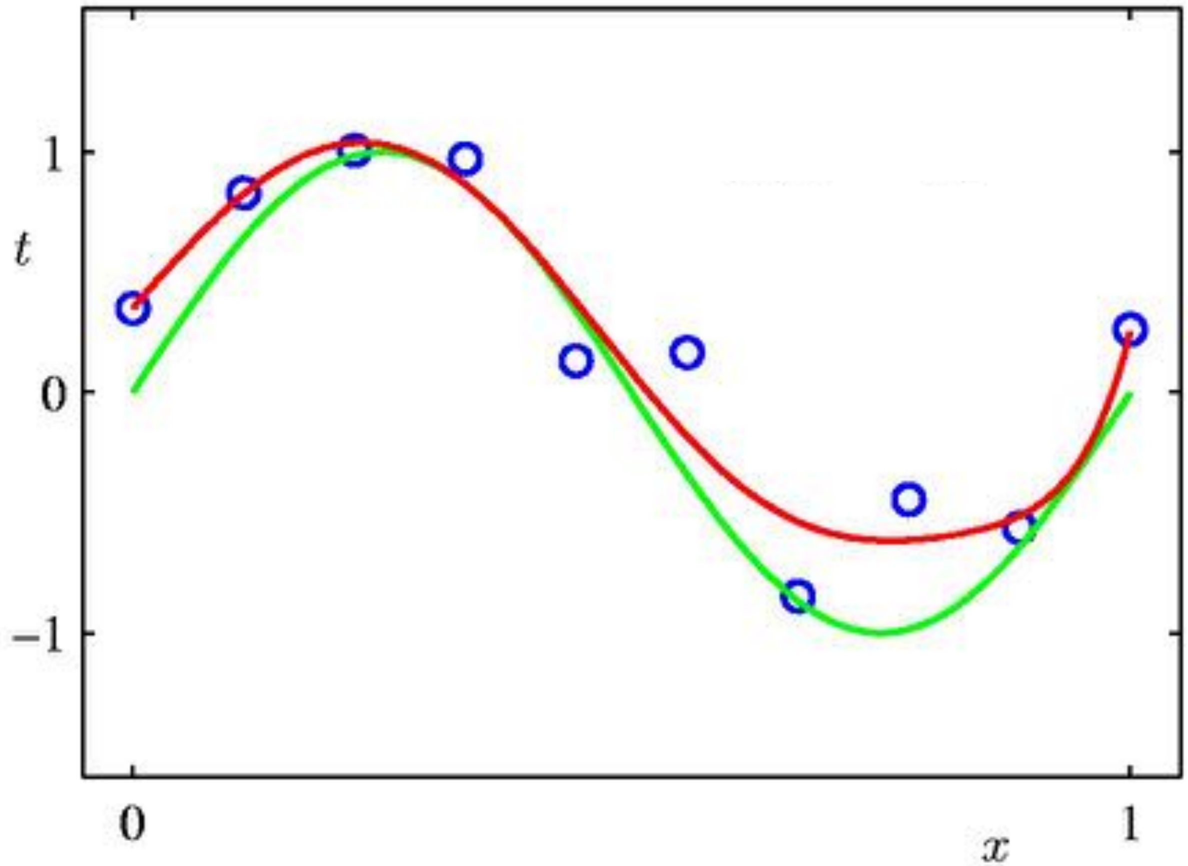
- In our models so far, the bias / intercept parameter is usually denoted by θ_0 -- that is, the parameter for which we fixed $x_0 = 1$
- Regularizers always avoid penalizing this bias / intercept parameter
- Why? Because otherwise the learning algorithms wouldn't be invariant to a shift in the y-values

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:

$$\ln\lambda = +.18$$



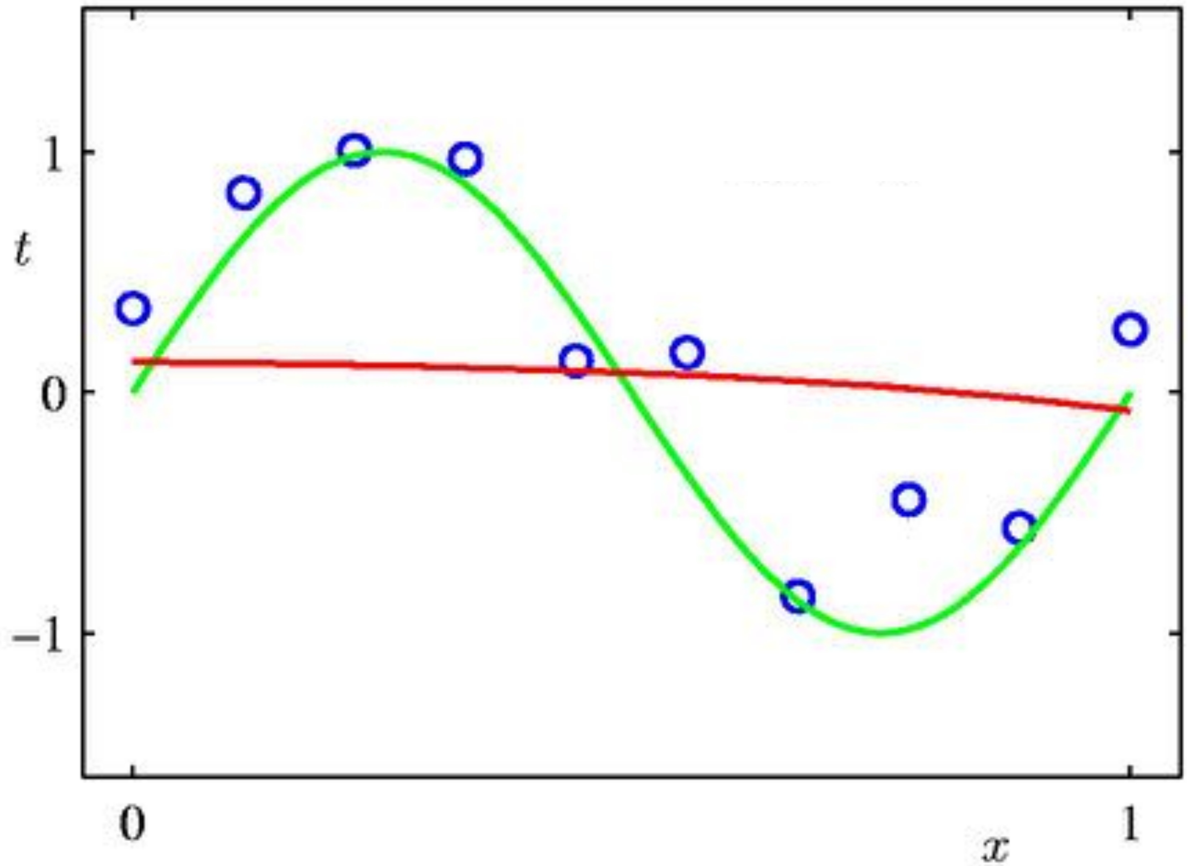
Slide courtesy of William Cohen

Polynomial Coefficients

	none	exp(18)	huge
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Slide courtesy of William Cohen

Over Regularization:

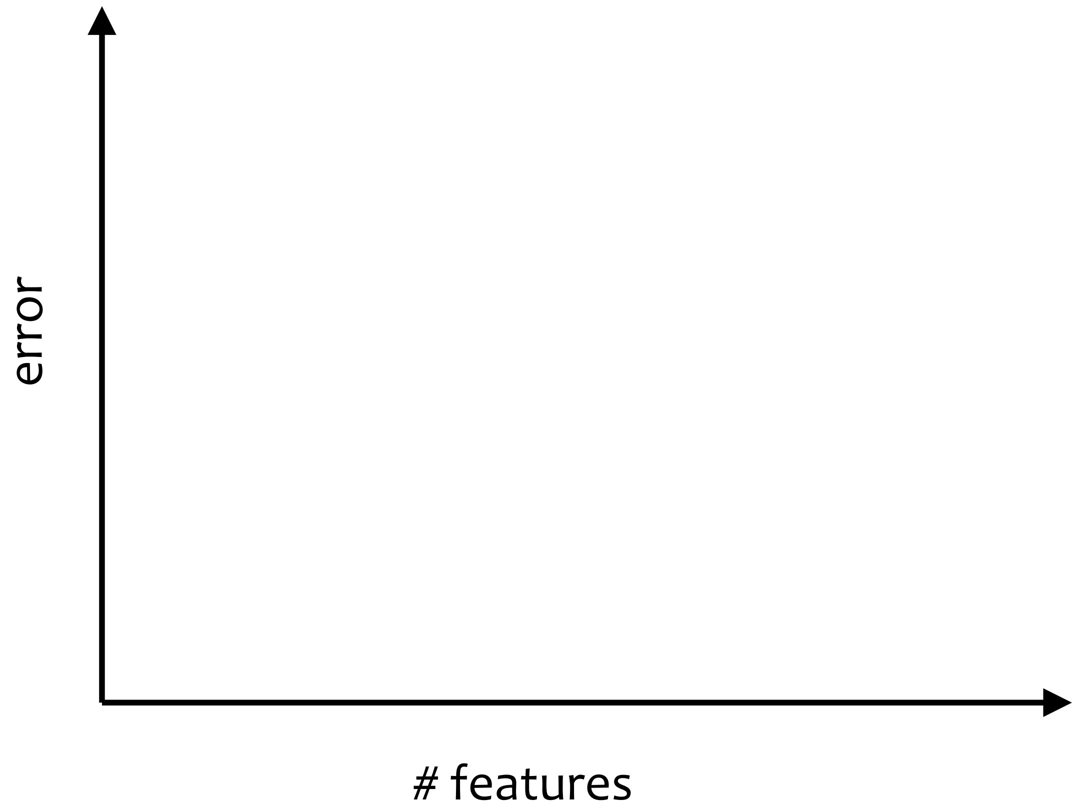


Slide courtesy of William Cohen

Regularization Exercise

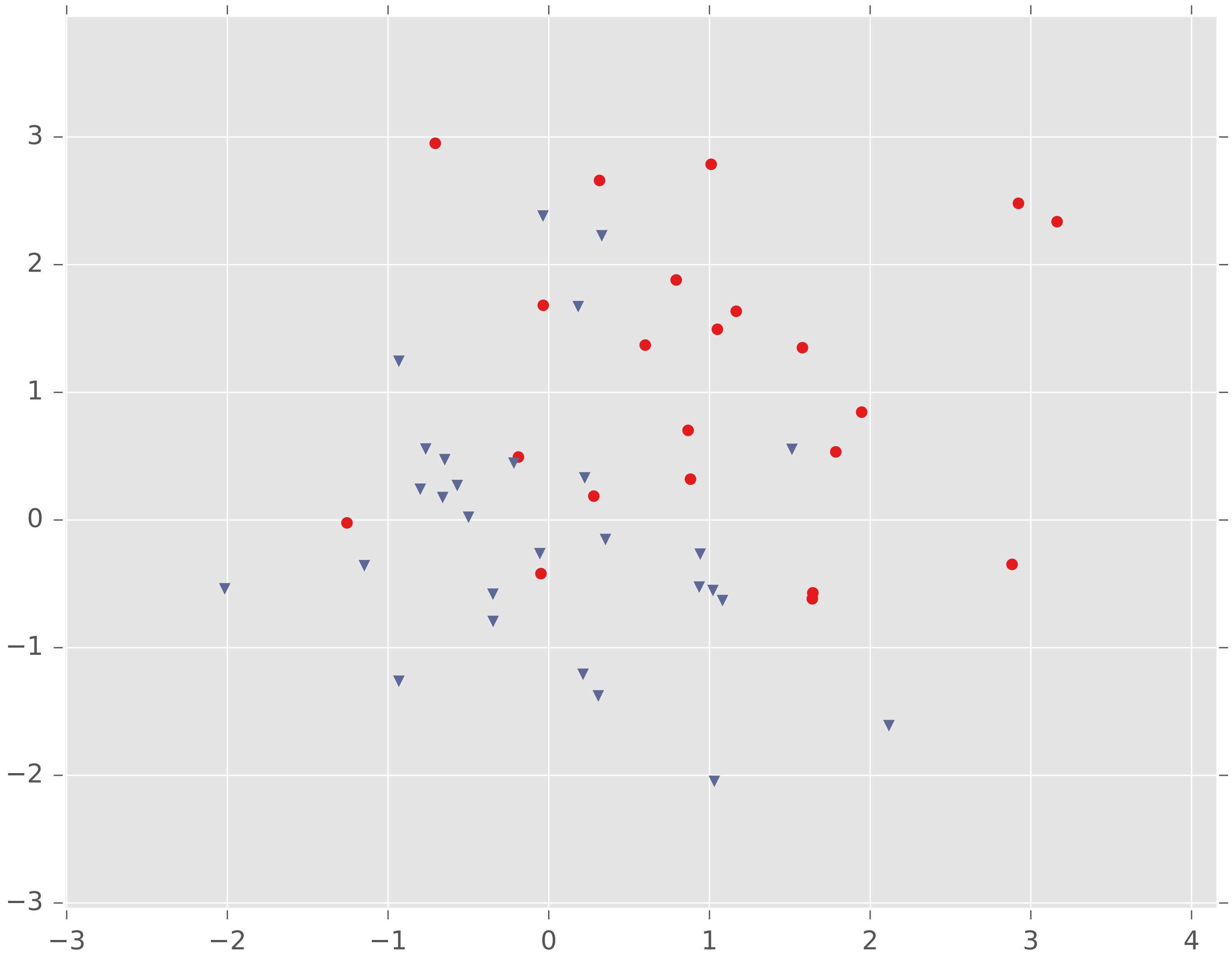
In-class Exercise

1. Plot train error vs. # features (cartoon)
2. Plot test error vs. # features (cartoon)



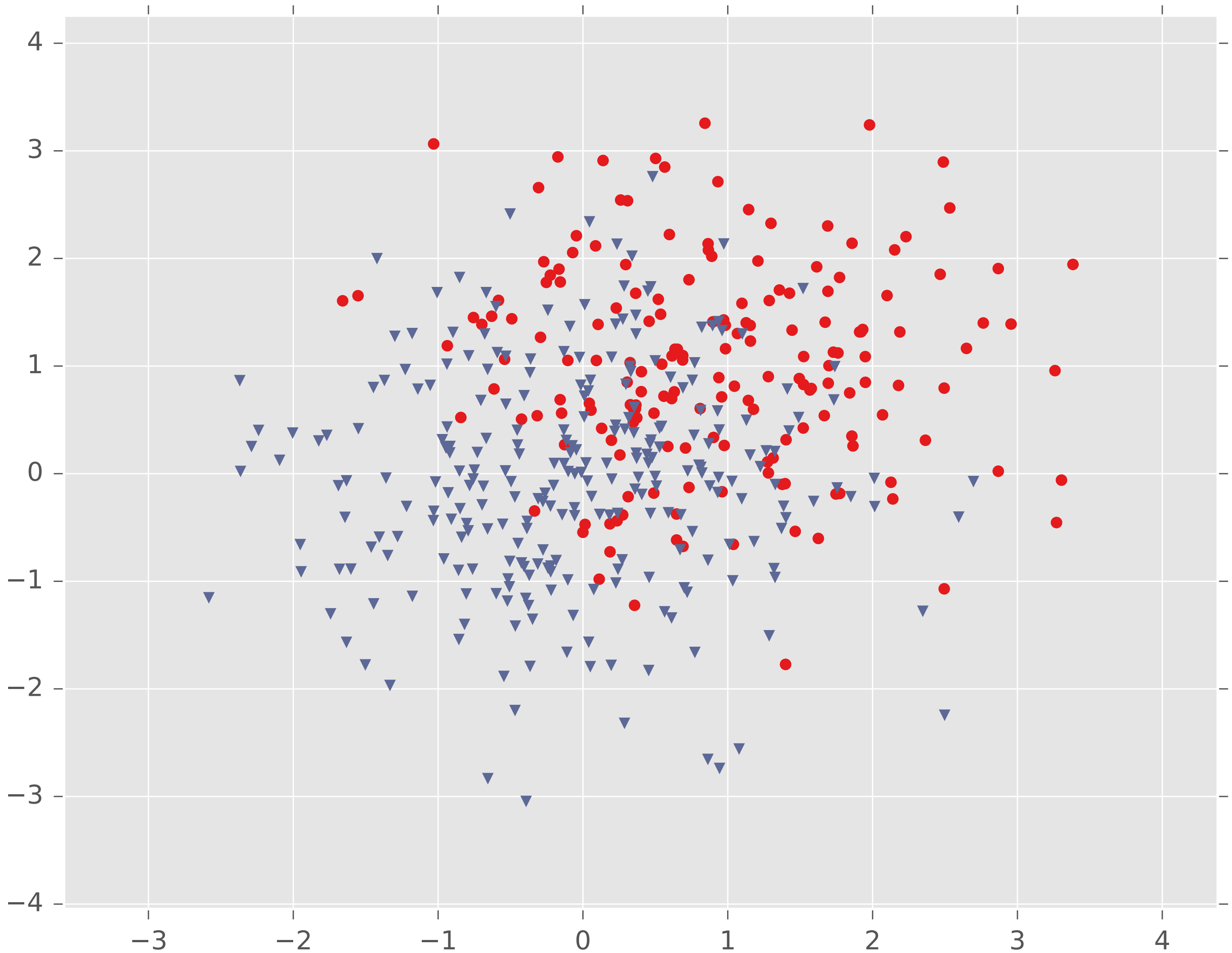
Example: Logistic Regression

Training Data

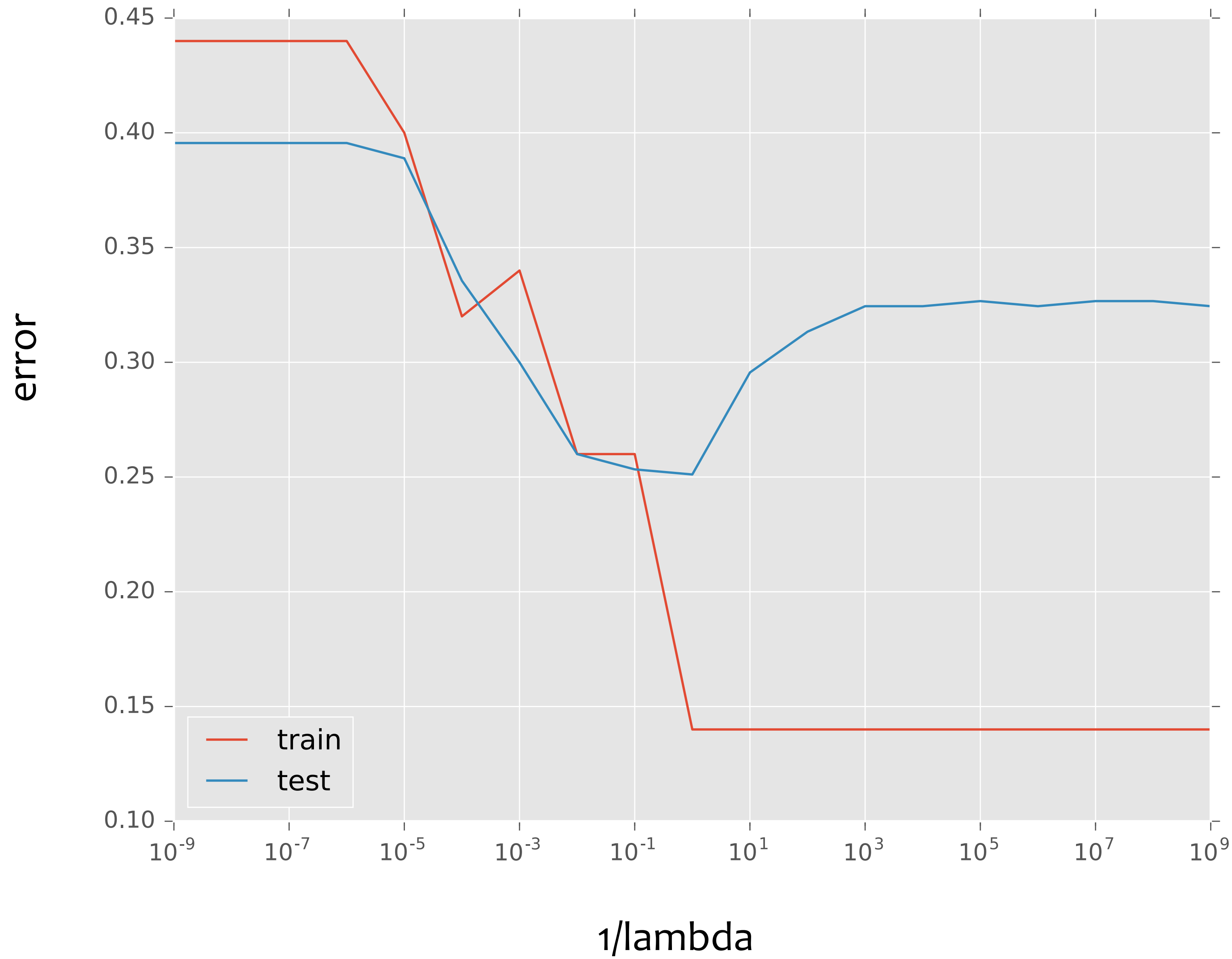


Example: Logistic Regression

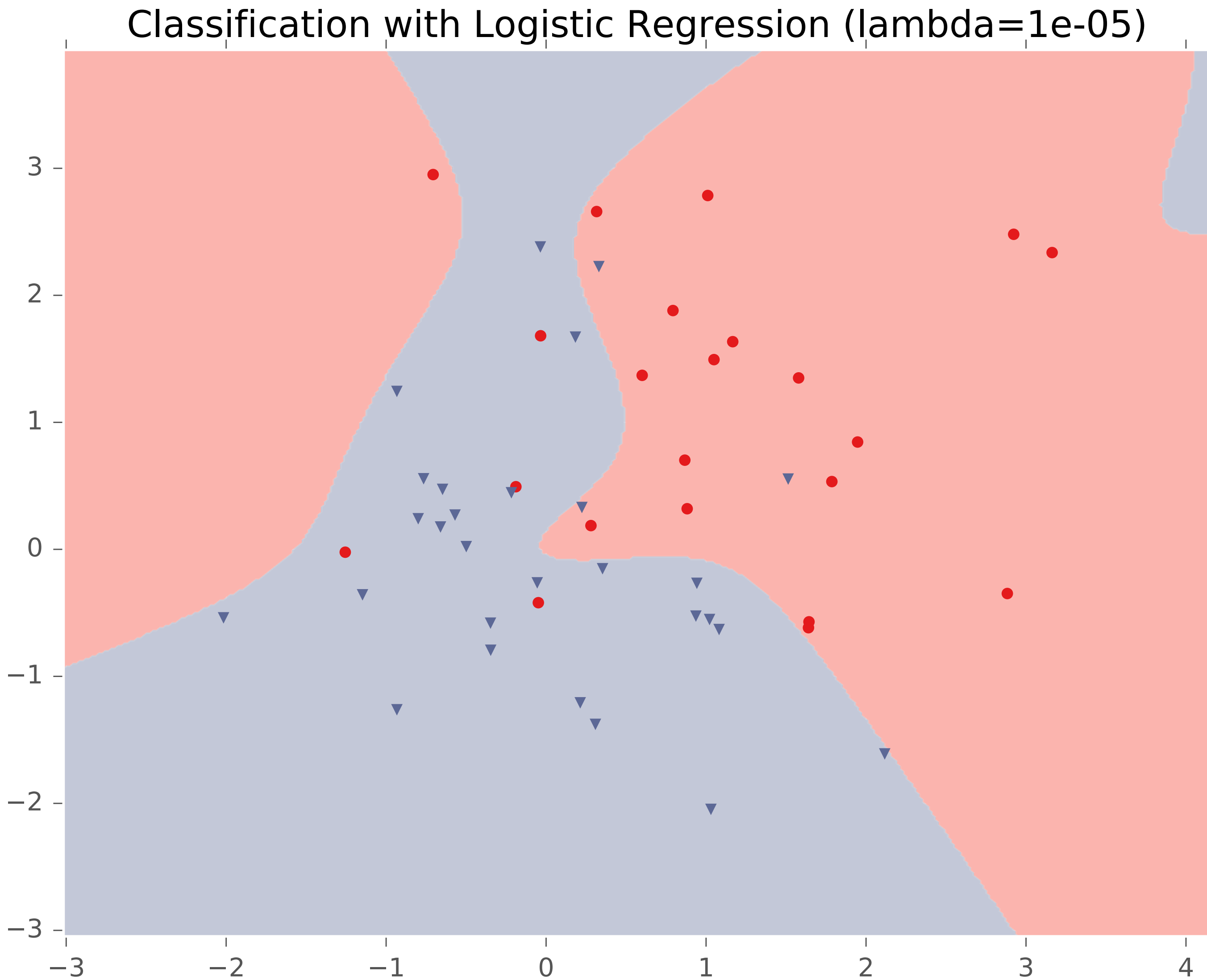
Test Data



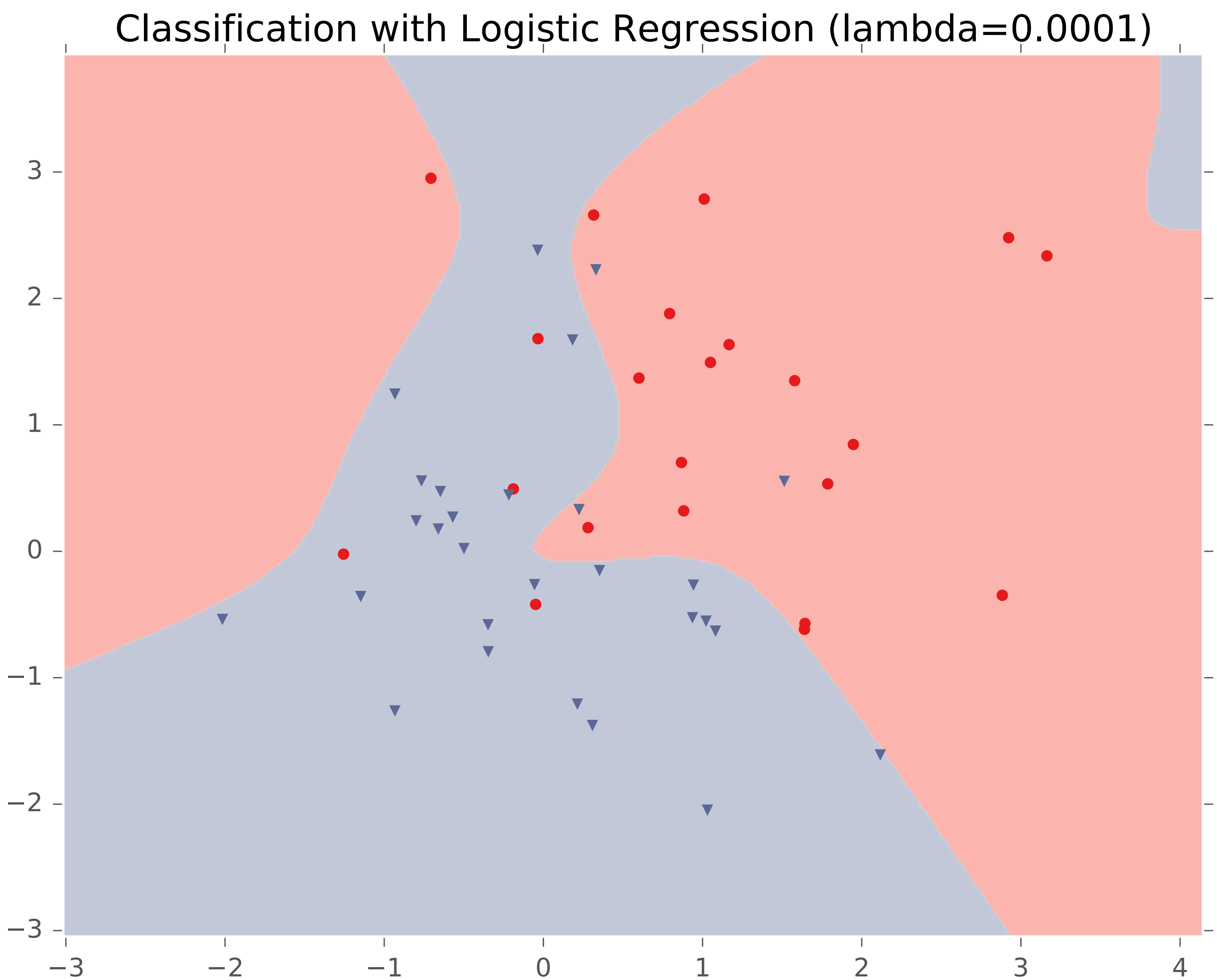
Example: Logistic Regression



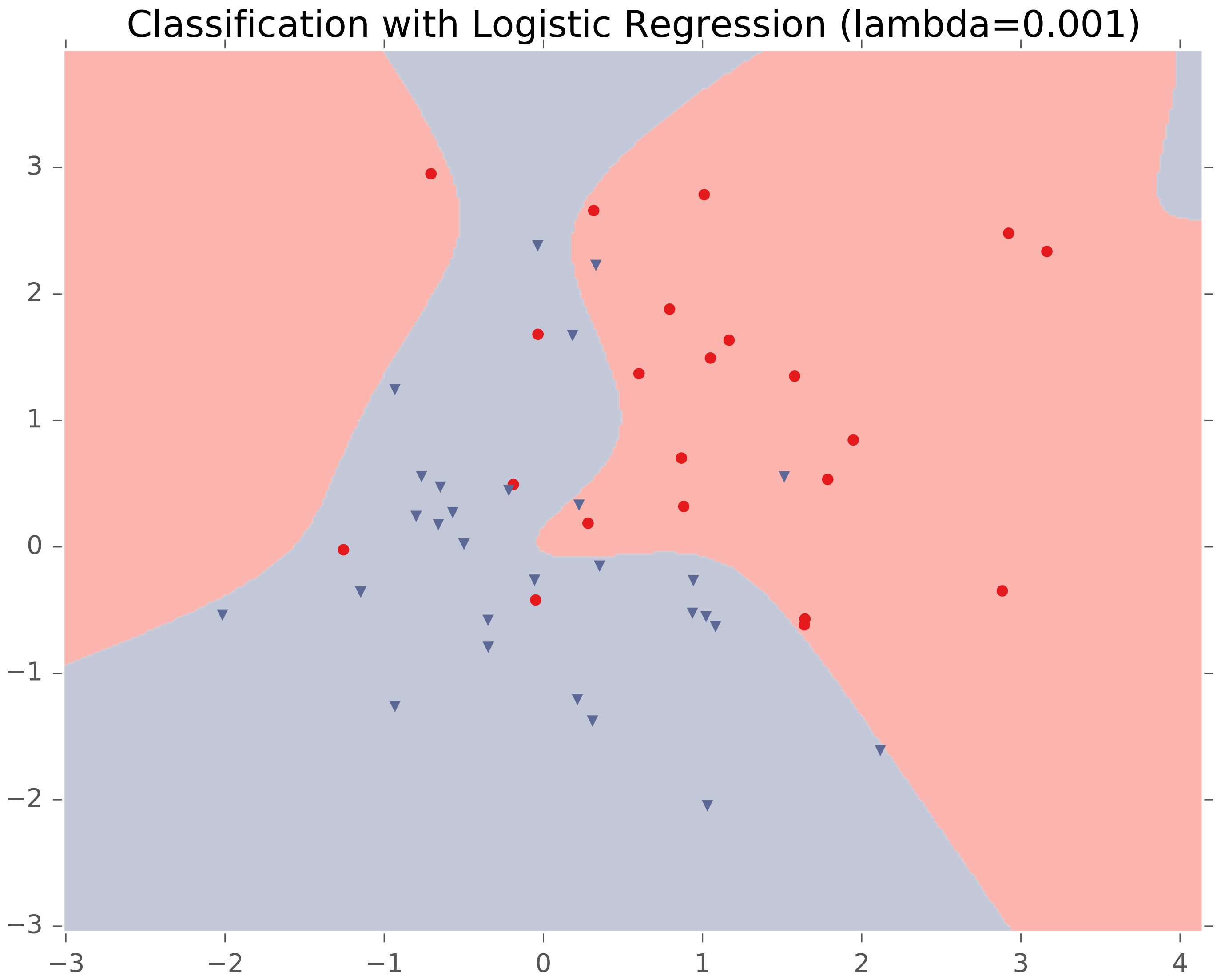
Example: Logistic Regression



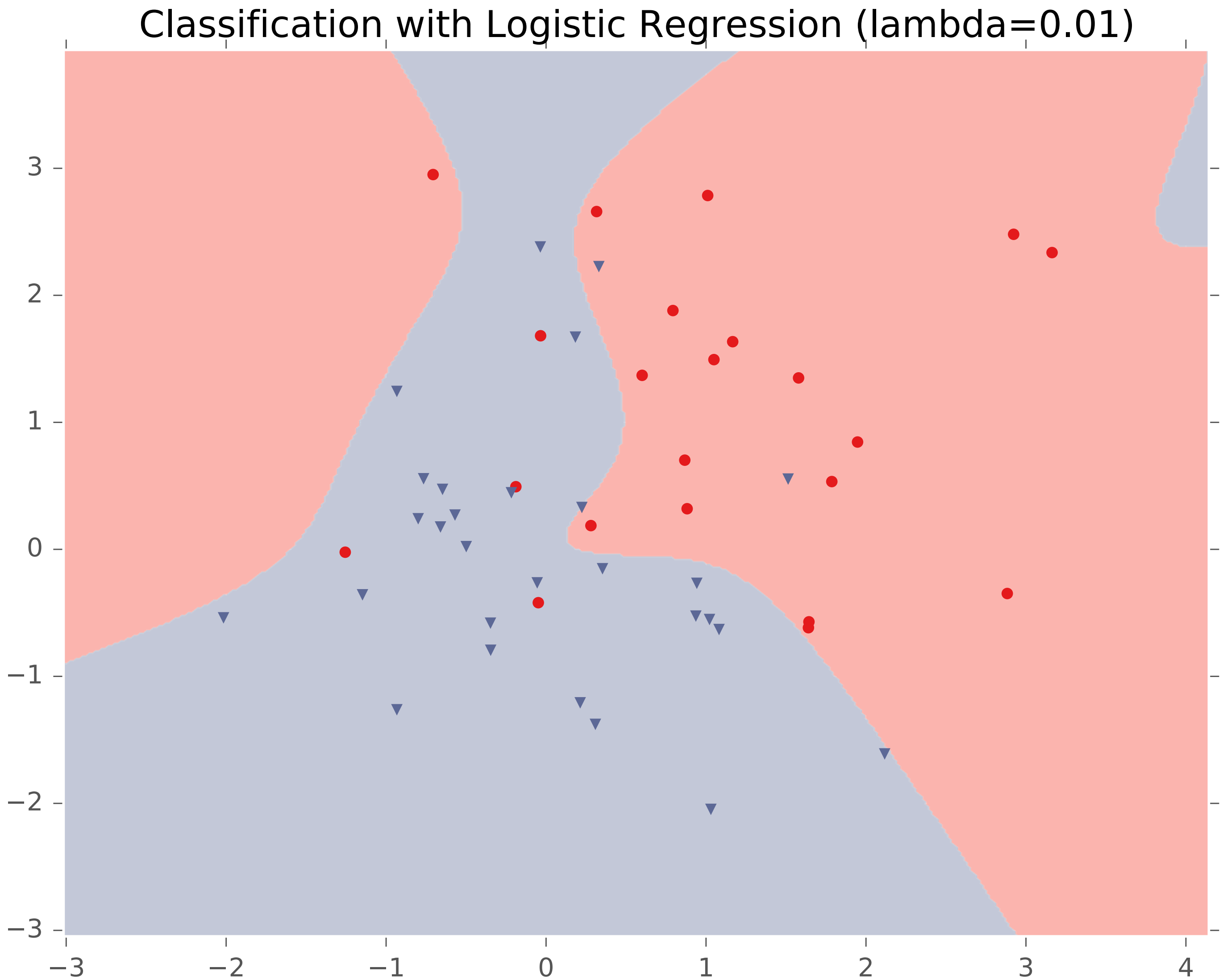
Example: Logistic Regression



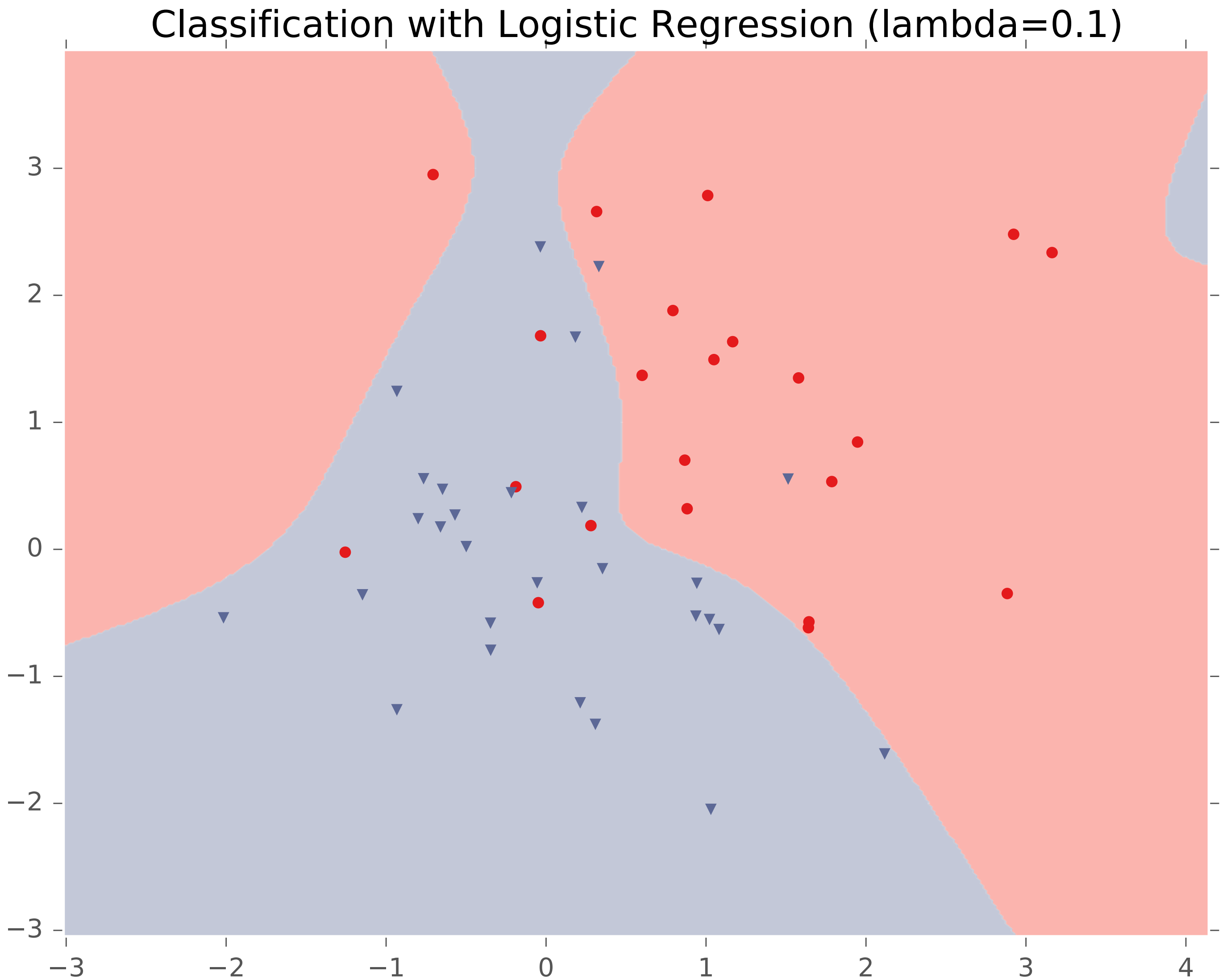
Example: Logistic Regression



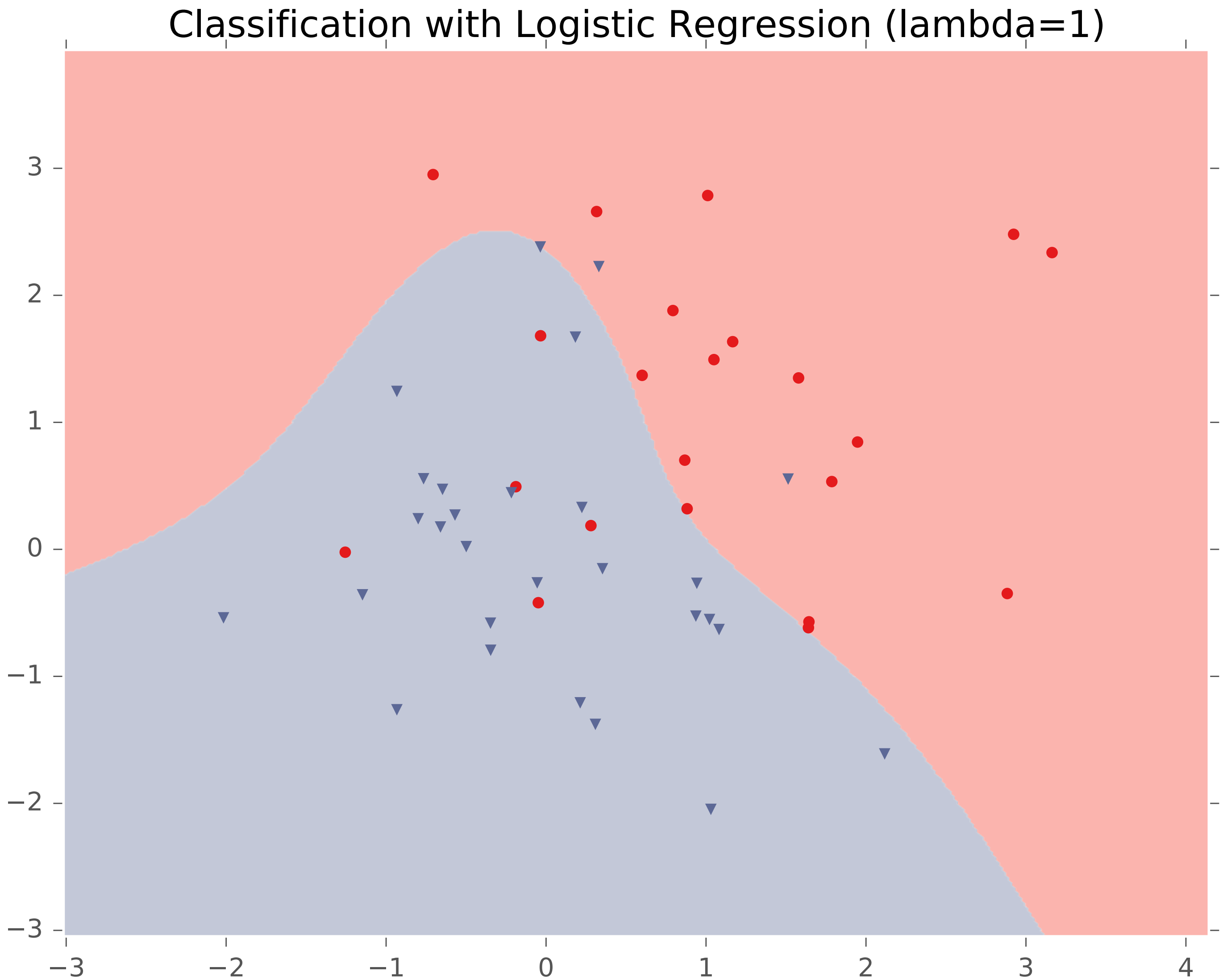
Example: Logistic Regression



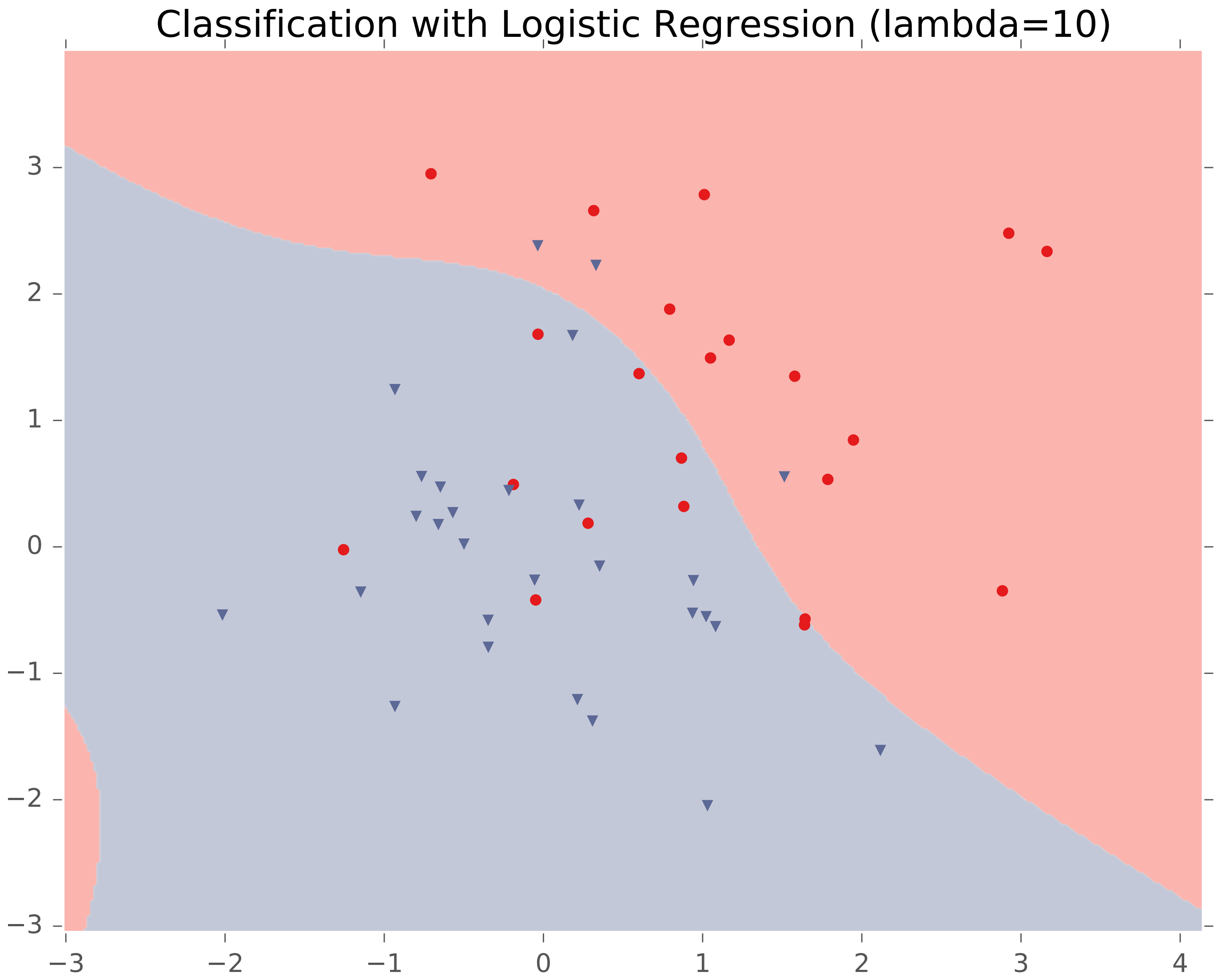
Example: Logistic Regression



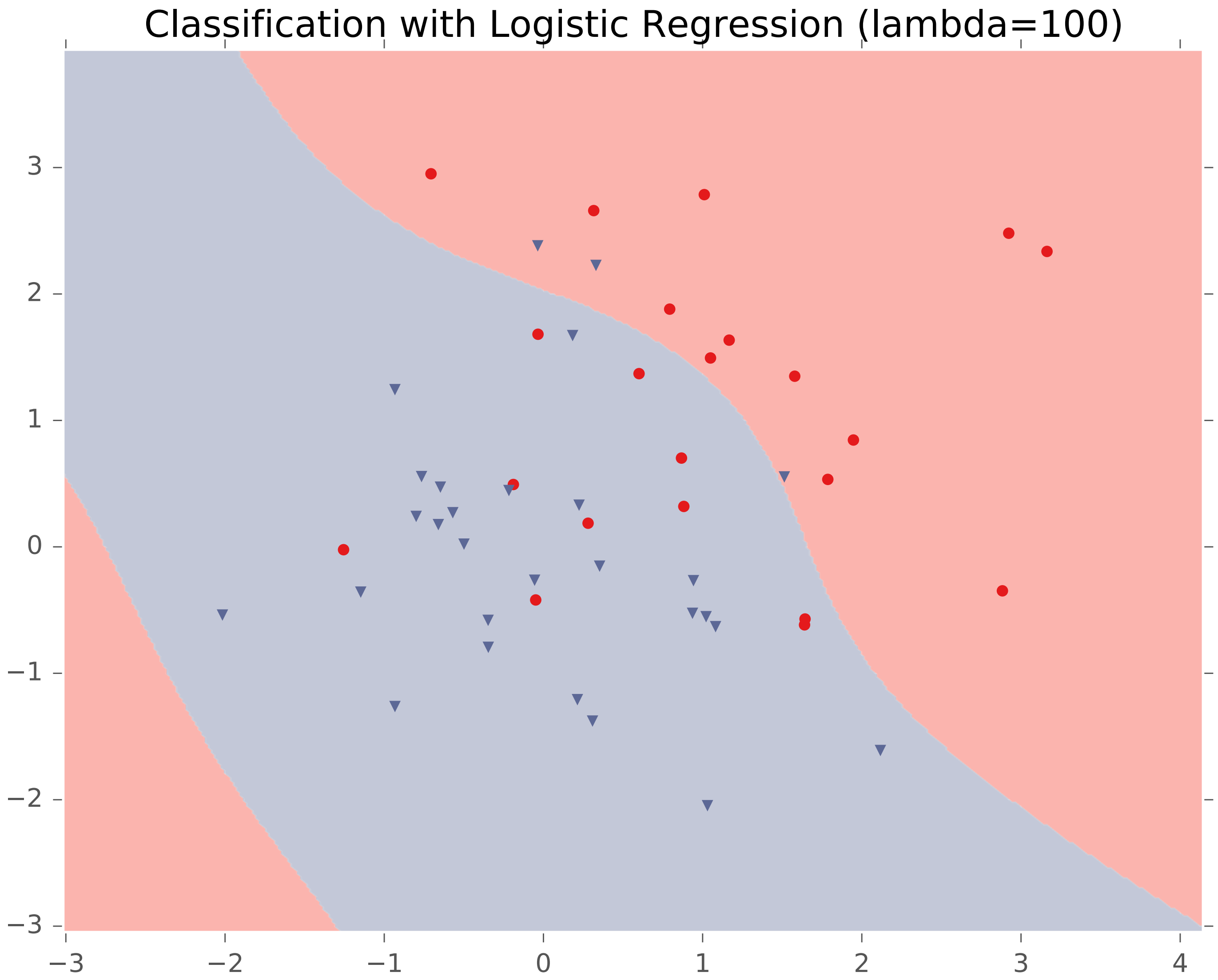
Example: Logistic Regression



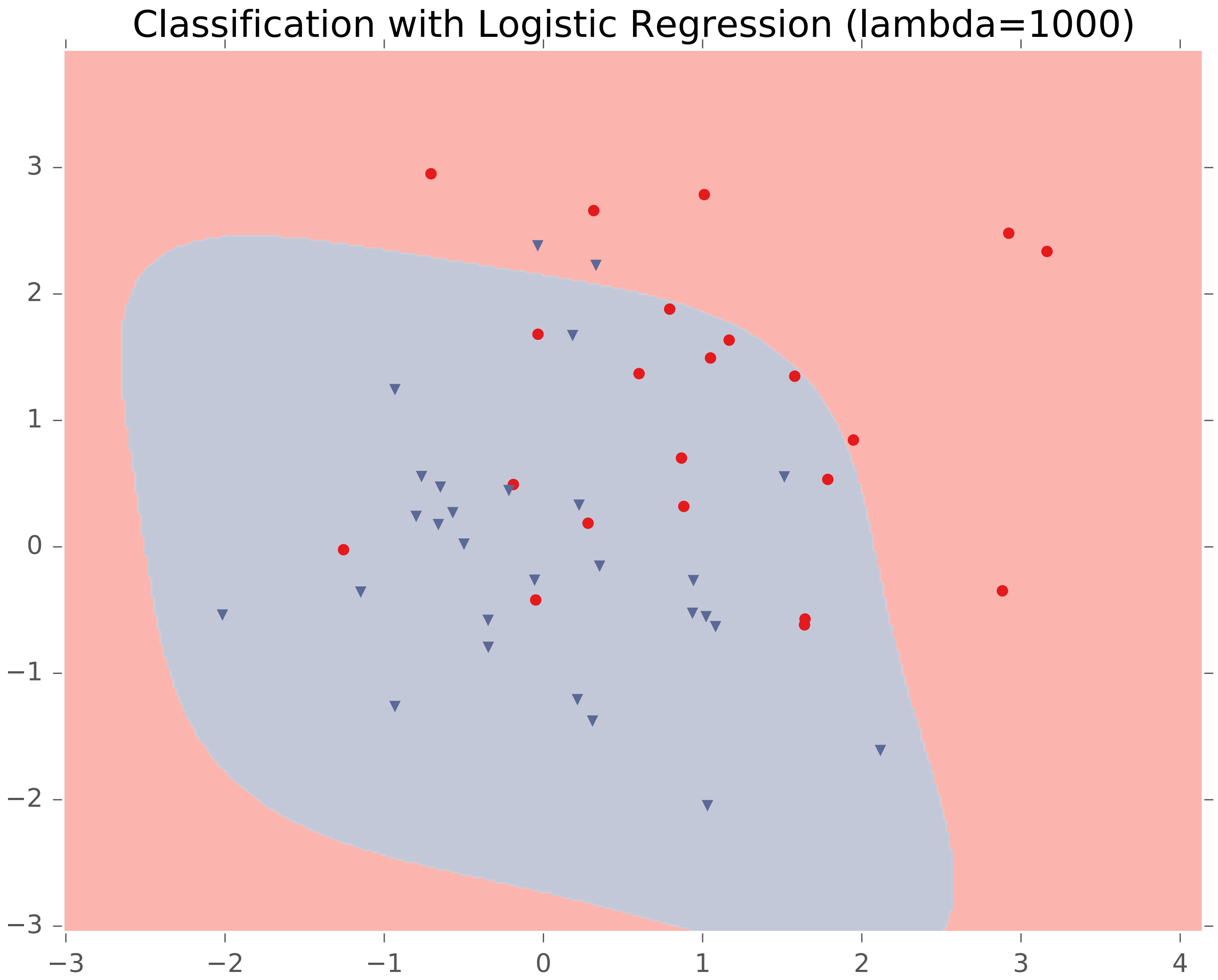
Example: Logistic Regression



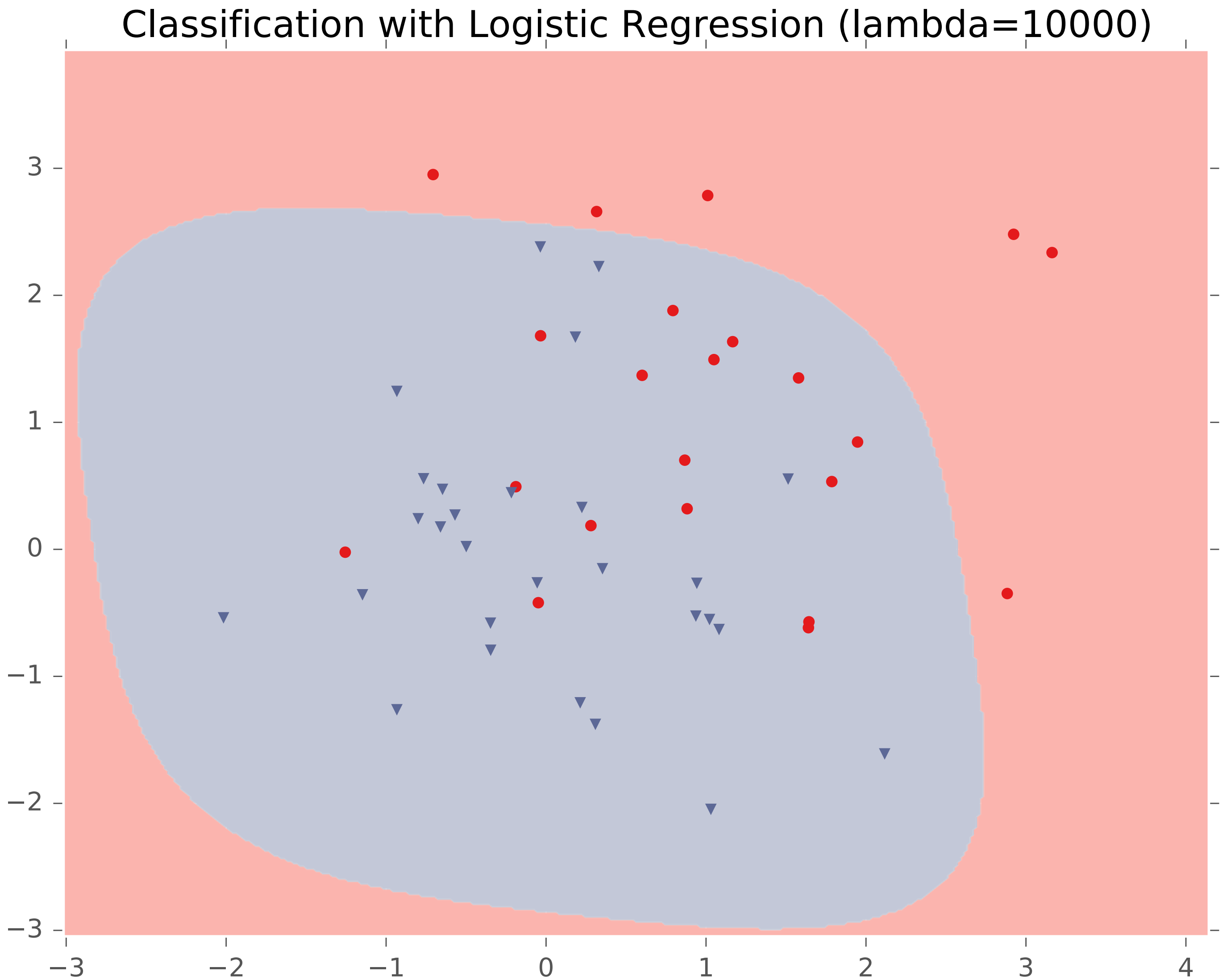
Example: Logistic Regression



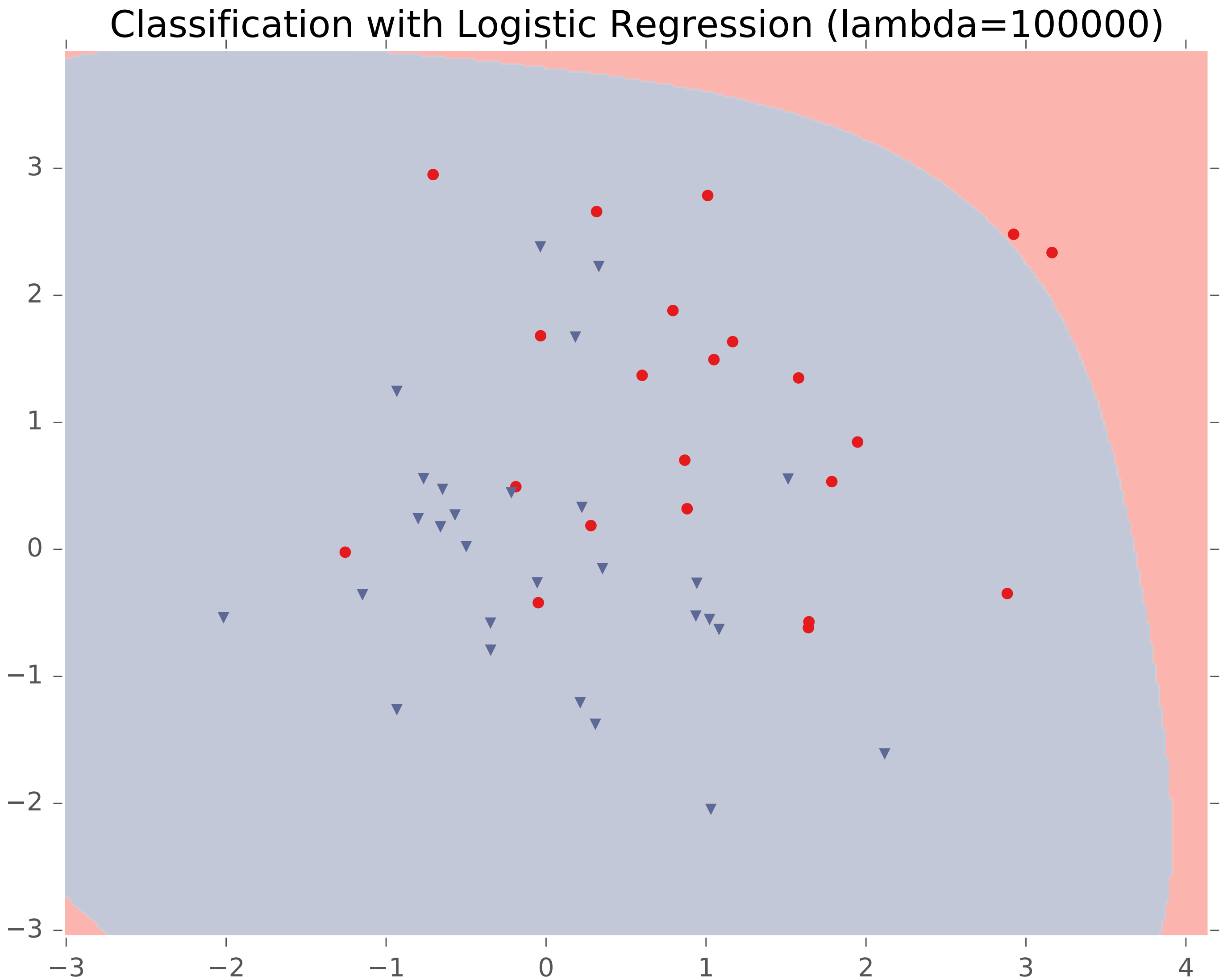
Example: Logistic Regression



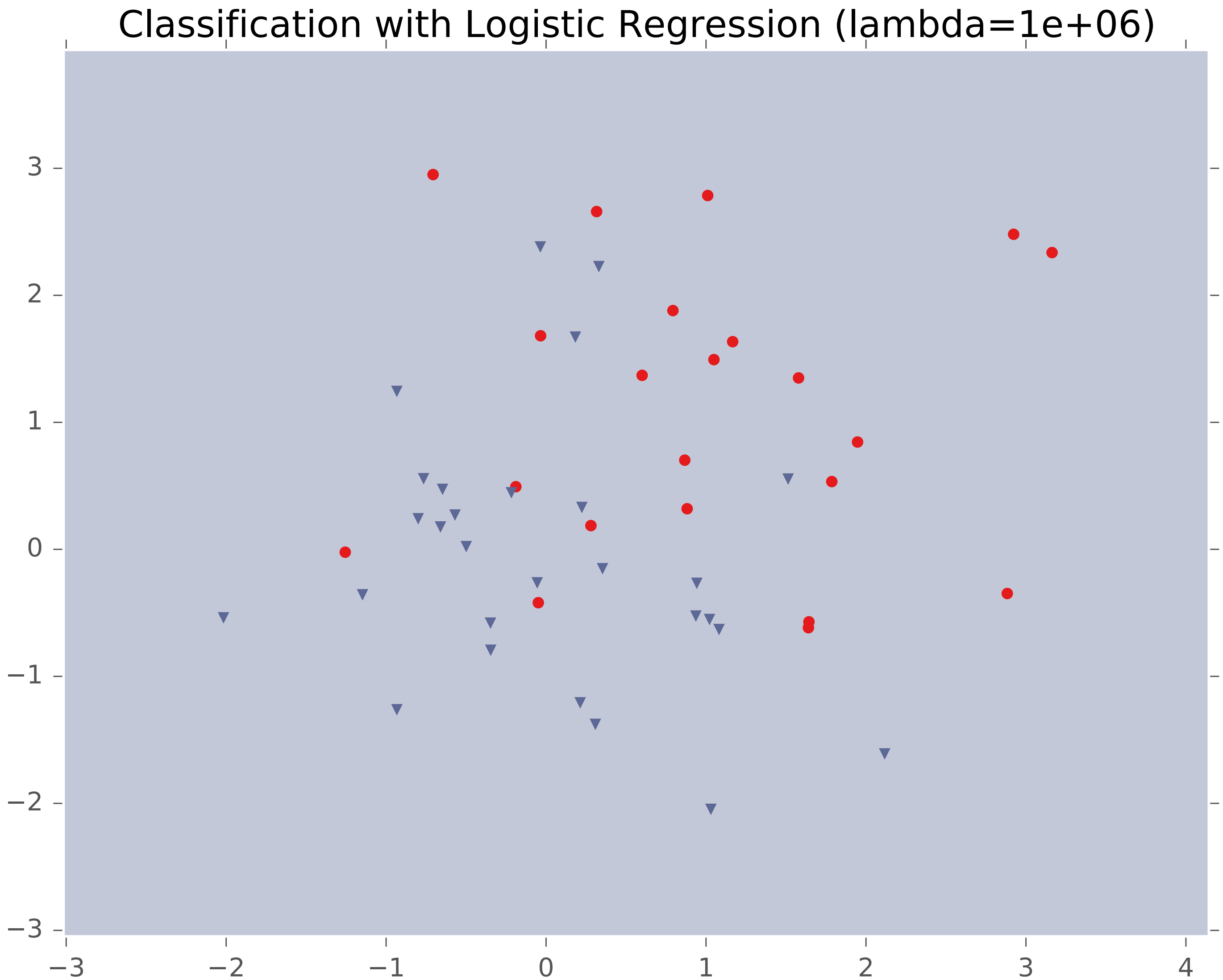
Example: Logistic Regression



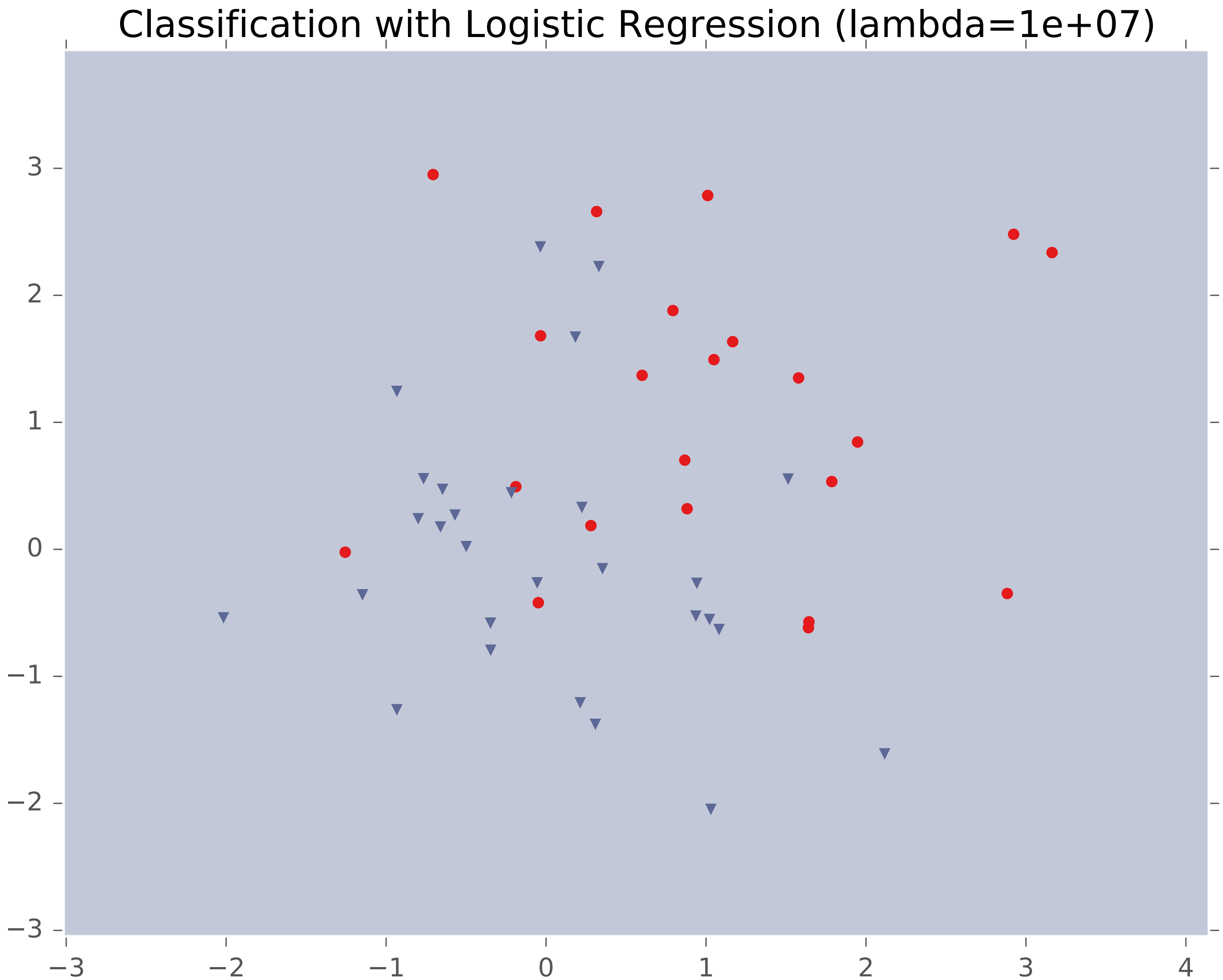
Example: Logistic Regression



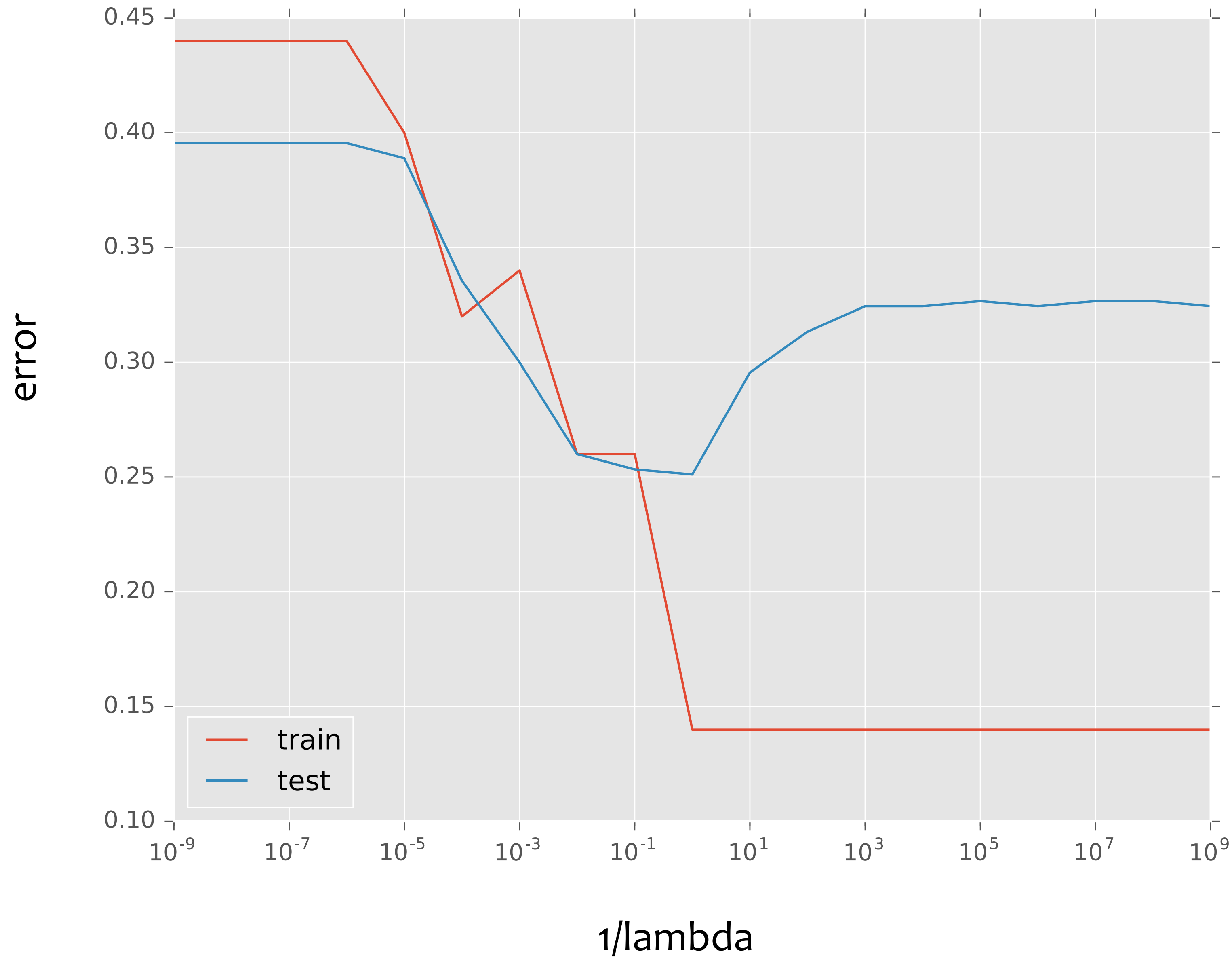
Example: Logistic Regression



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