



10-301/10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

(Linear Models) + Feature Engineering + Regularization

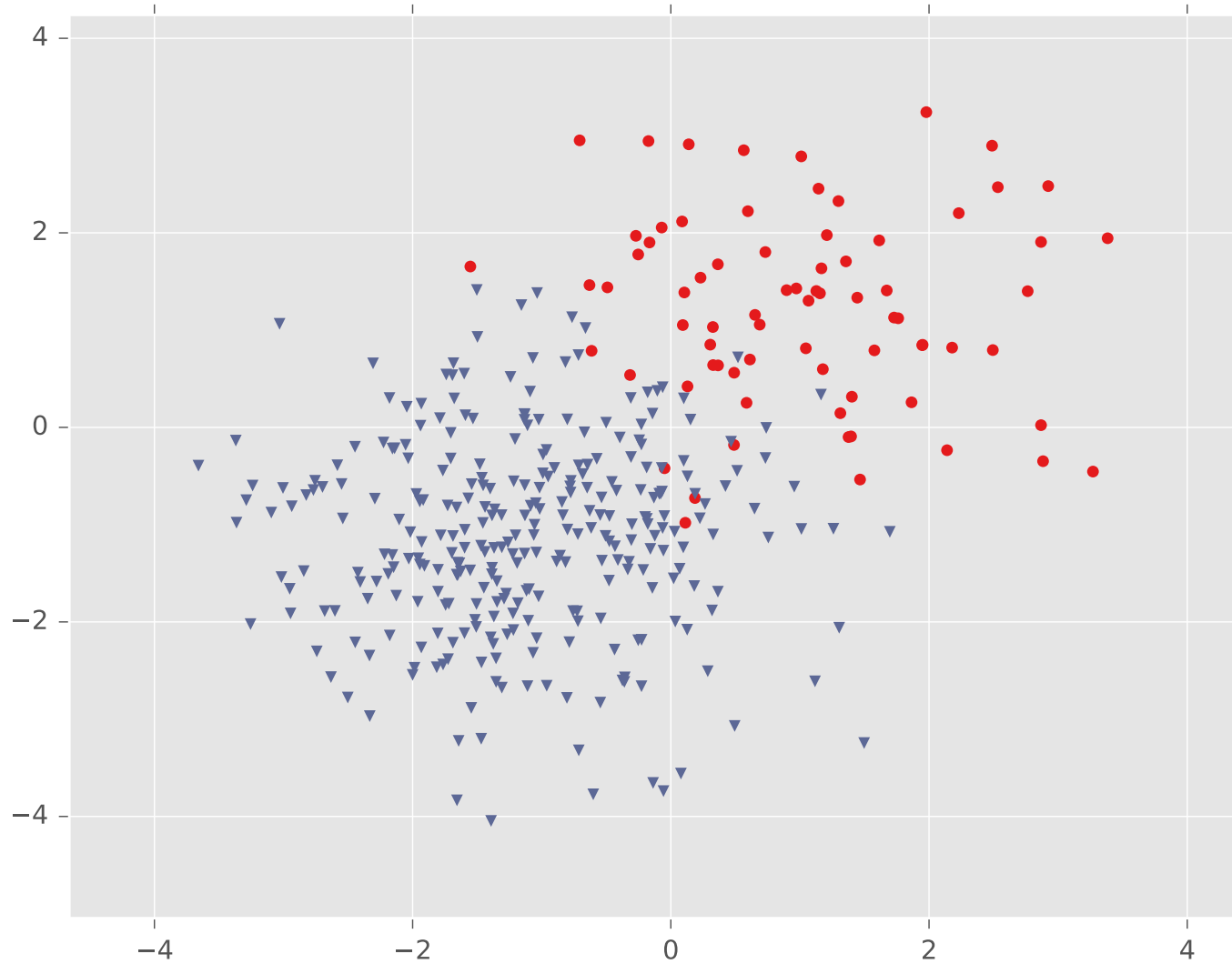
Matt Gormley
Lecture 10
Feb. 20, 2023

Reminders

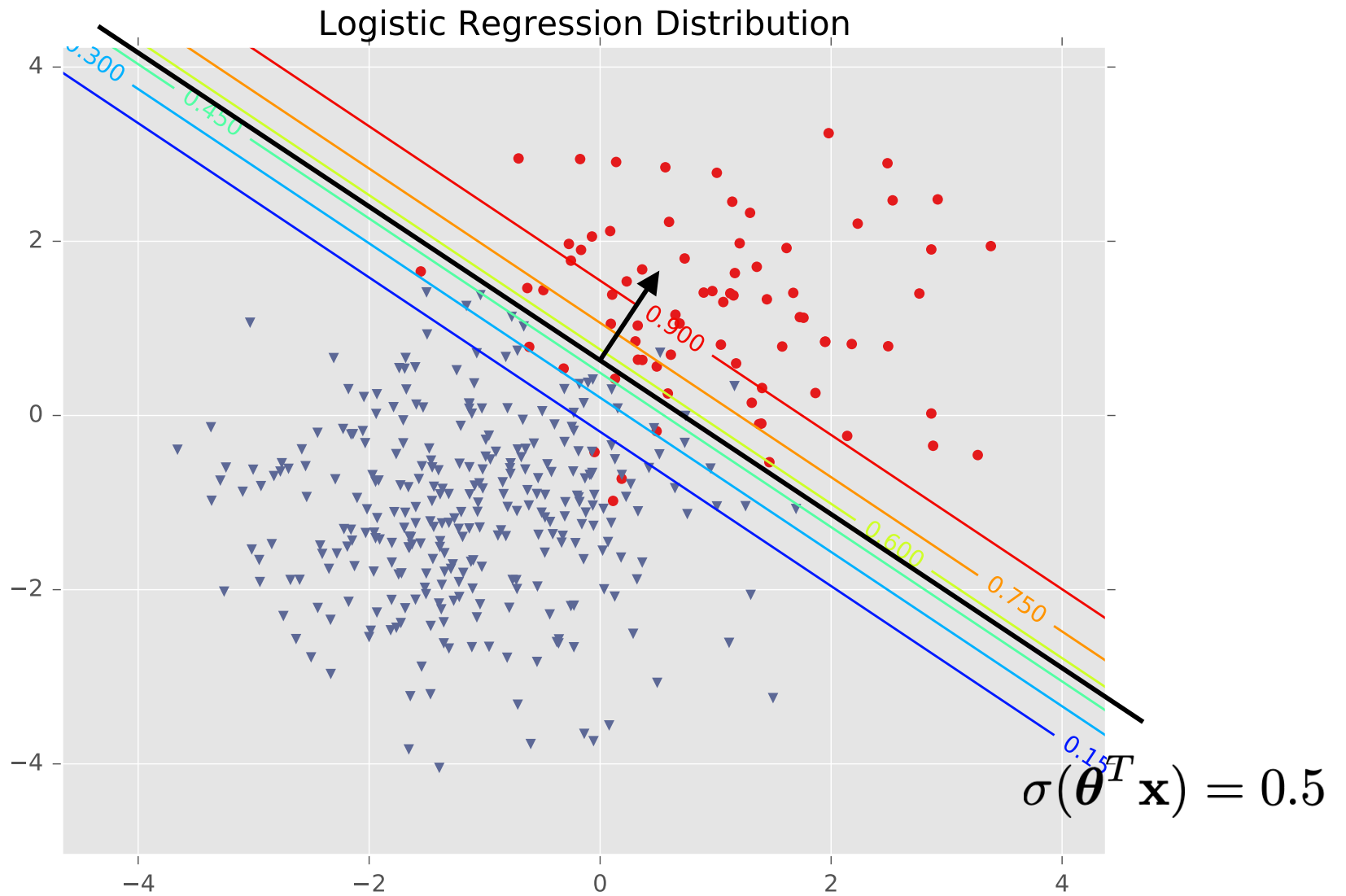
- **Homework 4: Logistic Regression**
 - **Out: Fri, Feb 17**
 - **Due: Sun, Feb. 26 at 11:59pm**

LOGISTIC REGRESSION ON GAUSSIAN DATA

Logistic Regression

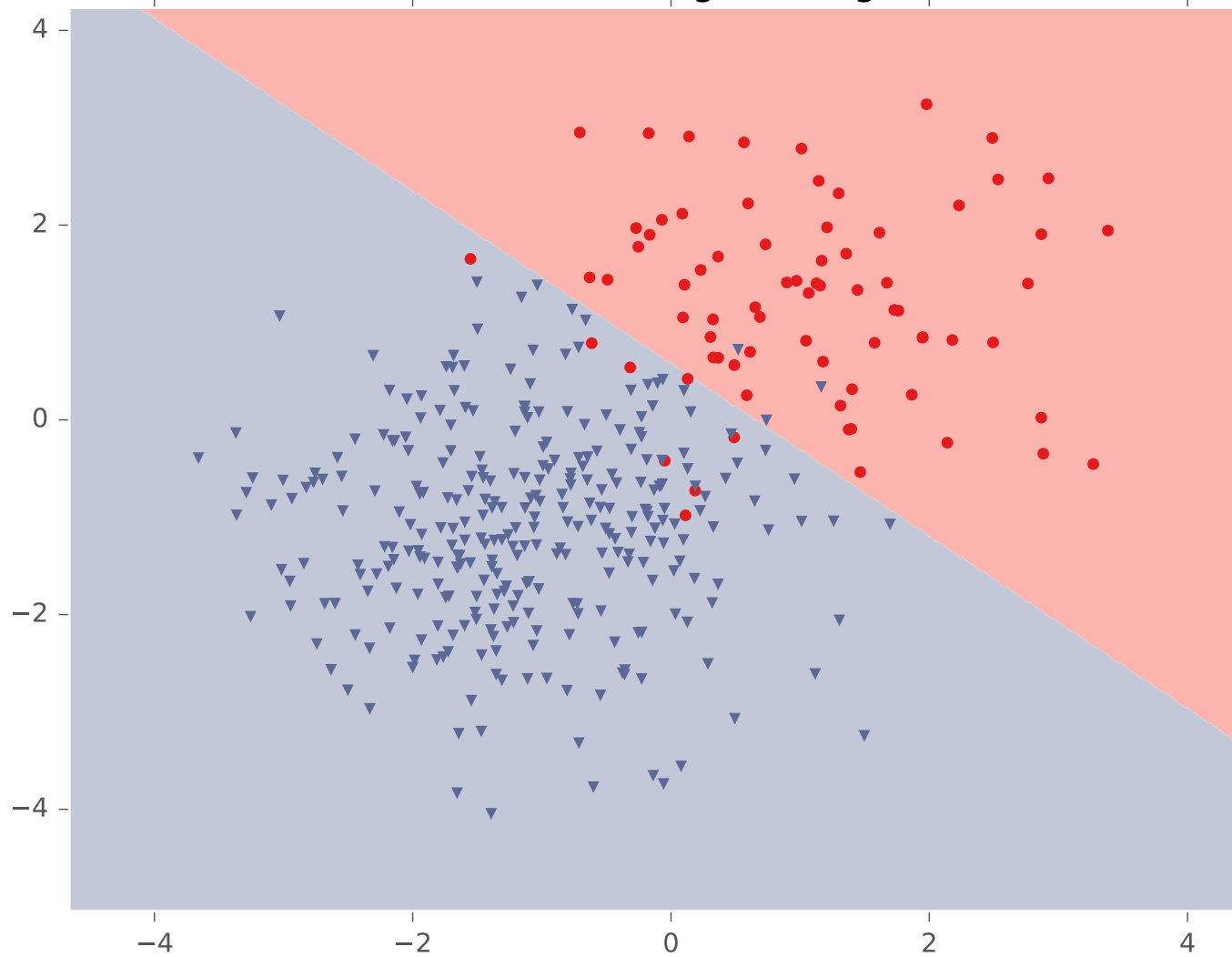


Logistic Regression



Logistic Regression

Classification with Logistic Regression



LEARNING LOGISTIC REGRESSION

Maximum Conditional Likelihood Estimation

Learning: finds the parameters that minimize some objective function.

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

We minimize the *negative* log conditional likelihood:

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

Why?

1. We can't maximize likelihood (as in Naïve Bayes) because we don't have a joint model $p(\mathbf{x}, y)$
2. It worked well for Linear Regression (least squares is actually MCLE! more on this later...)

Maximum Conditional Likelihood Estimation

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

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)

Maximum Conditional Likelihood Estimation

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

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.

Linear Models

PERCEPTRON, LINEAR REGRESSION, AND LOGISTIC REGRESSION

Why is it not “Logistic Classification”?

Whiteboard

- Conceptual Change: 2D classification in 3D
- Why is it called *Logistic Regression* and not *Logistic Classification*?

Matching Game

Question:

Match the Algorithm to its Update Rule

1. SGD for Logistic Regression

$$h_{\theta}(\mathbf{x}) = p(y|x)$$

2. Least Mean Squares

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

3. Perceptron

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

$$4. \quad \theta_k \leftarrow \theta_k + (h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)})$$

$$5. \quad \theta_k \leftarrow \theta_k + \frac{1}{1 + \exp \lambda(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)})}$$

$$6. \quad \theta_k \leftarrow \theta_k + \lambda(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)})x_k^{(i)}$$

Answer:

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

I. None of the above

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...

- A. (1) compute the gradient of the log-likelihood for all examples (2) update all the parameters using the gradient
- B. (1) ask Matt for a description of SGD for Logistic Regression, (2) write it down, (3) report that answer
- 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
- 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
- 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
- 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

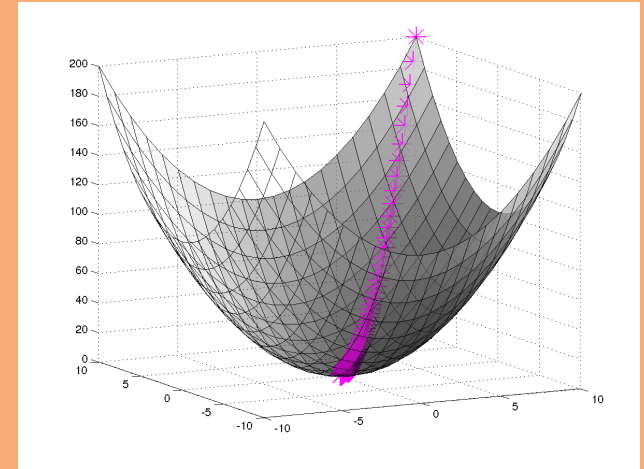
Gradient Descent

Algorithm 1 Gradient Descent

```

1: procedure GD( $\mathcal{D}$ ,  $\theta^{(0)}$ )
2:    $\theta \leftarrow \theta^{(0)}$ 
3:   while not converged do
4:      $\theta \leftarrow \theta - \gamma \nabla_{\theta} J(\theta)$ 
5:   return  $\theta$ 

```



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

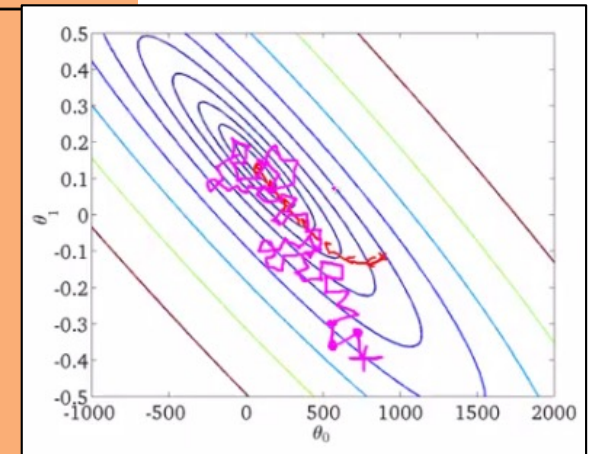
$$\nabla_{\theta} J(\theta) = \begin{bmatrix} \frac{d}{d\theta_1} J(\theta) \\ \frac{d}{d\theta_2} J(\theta) \\ \vdots \\ \frac{d}{d\theta_M} J(\theta) \end{bmatrix}$$

Recall...

Stochastic Gradient Descent (SGD)

Algorithm 1 Stochastic Gradient Descent (SGD)

```
1: procedure SGD( $\mathcal{D}, \theta^{(0)}$ )
2:    $\theta \leftarrow \theta^{(0)}$ 
3:   while not converged do
4:     for  $i \in \text{shuffle}(\{1, 2, \dots, N\})$  do
5:        $\theta \leftarrow \theta - \gamma \nabla_{\theta} J^{(i)}(\theta)$ 
6:   return  $\theta$ 
```



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

We need a per-example objective:

$$\text{Let } J(\boldsymbol{\theta}) = \sum_{i=1}^N J^{(i)}(\boldsymbol{\theta})$$

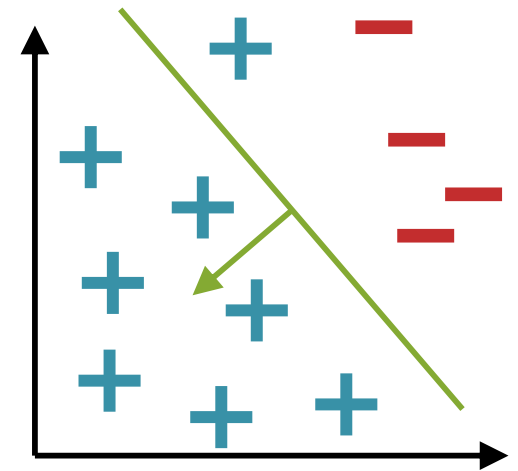
where $J^{(i)}(\boldsymbol{\theta}) = -\log p_{\boldsymbol{\theta}}(y^i | \mathbf{x}^i)$.

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:



BAYES OPTIMAL CLASSIFIER

Bayes Optimal Classifier

Suppose you knew the distribution $p^*(y | \mathbf{x})$ or had a good approximation to it.

Question:

How would you design a function $y = h(\mathbf{x})$ to predict a single label?

Answer:

You'd use the *Bayes optimal classifier!*

approximates $c(\mathbf{x})$

Probabilistic Learning

Today, we assume that our output is **sampled** from a conditional **probability distribution**:

$$\mathbf{x}^{(i)} \sim p^*(\cdot)$$


$$y^{(i)} \sim p^*(\cdot | \mathbf{x}^{(i)})$$

Our goal is to learn a probability distribution $p(y|\mathbf{x})$ that best approximates $p^*(y|\mathbf{x})$

Bayes Optimal Classifier

Suppose you have an **oracle** that knows the data generating distribution, $p^*(y|x)$.

Q: What is the optimal classifier in this setting?

A: The Bayes optimal classifier! This is the best classifier for the distribution p^* and the loss function.



Definition: The **reducible error** is the expected loss of a hypothesis $h(x)$ that could be reduced if knew a $p^*(y|x)$ and picked a the optimal $h(x)$ for that p^* .

Definition: The **irreducible error** is the expected loss of a hypothesis $h(x)$ that could **not** be reduced if knew a $p^*(y|x)$ and picked a the optimal $h(x)$ for that p^* .

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 randomly chosen examples

Mini-Batch SGD

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

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 $\forall s$

Logistic Regression Objectives

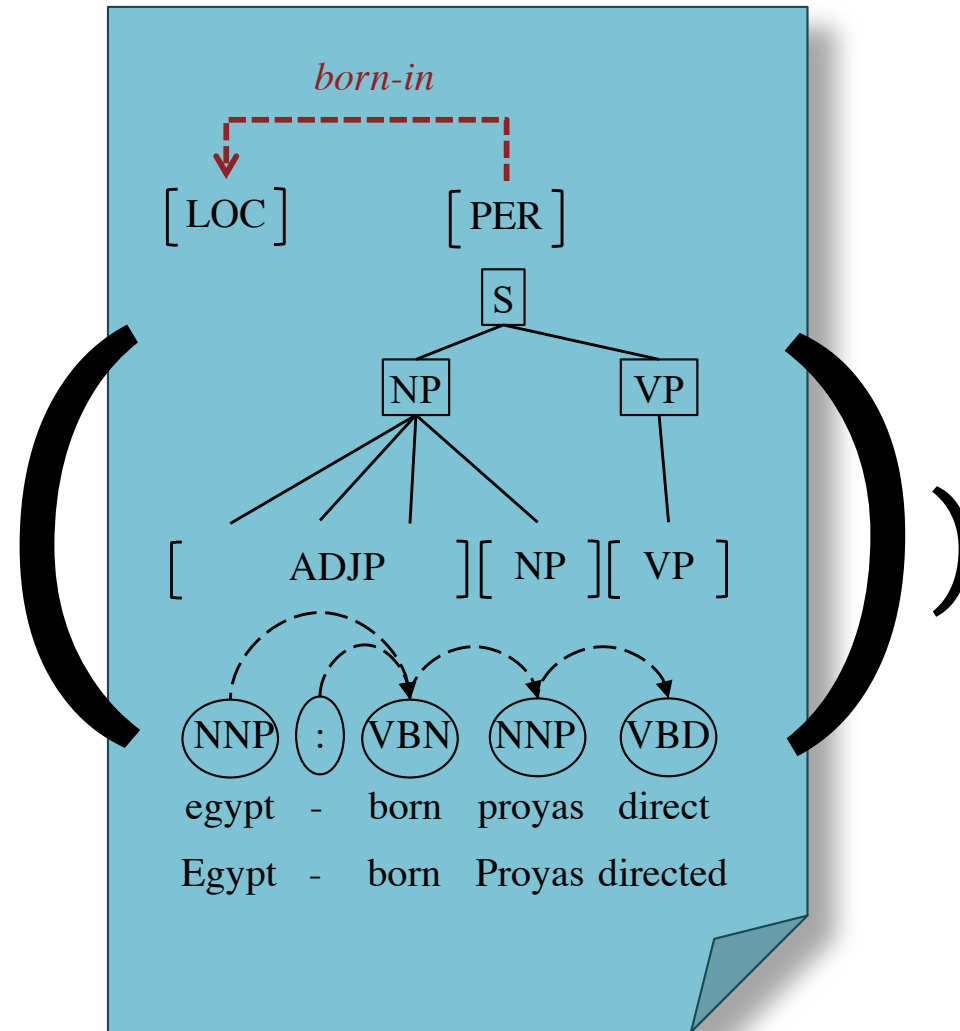
You should be able to...

- 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 classification
- Prove that the decision boundary of binary logistic regression is linear

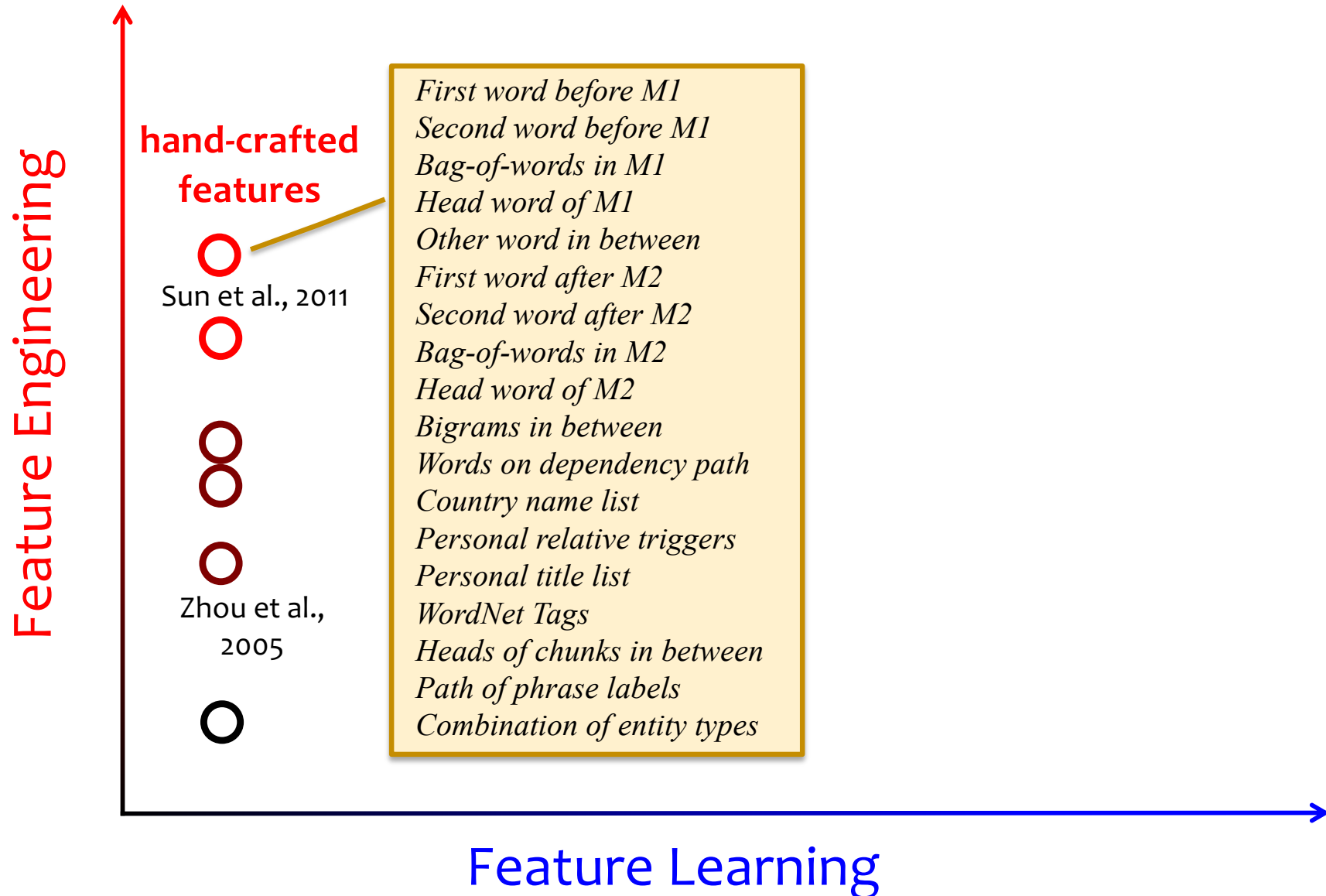
FEATURE ENGINEERING

Handcrafted Features

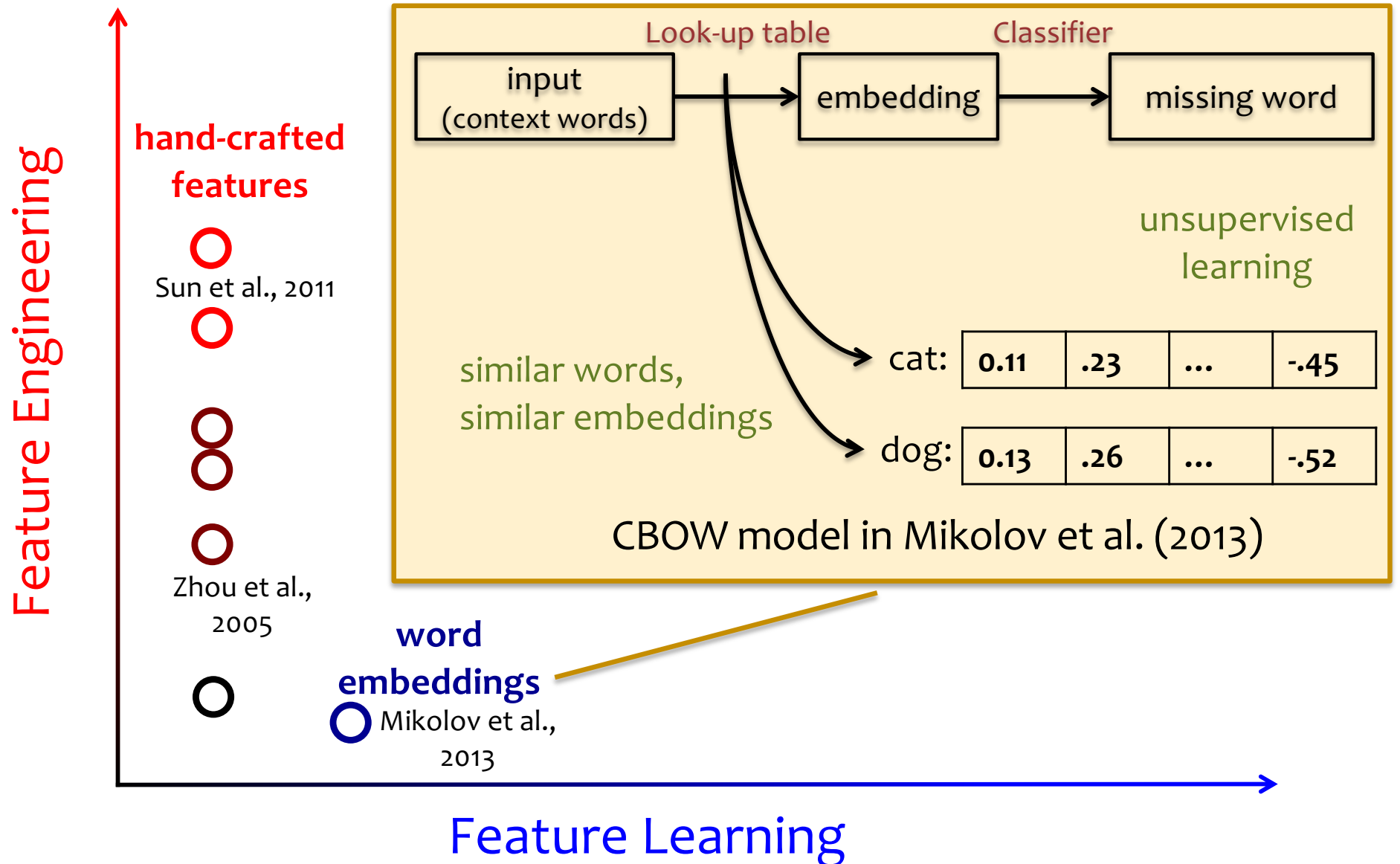
$$p(y|x) \propto \exp(\Theta_y \cdot f)$$



Where do features come from?

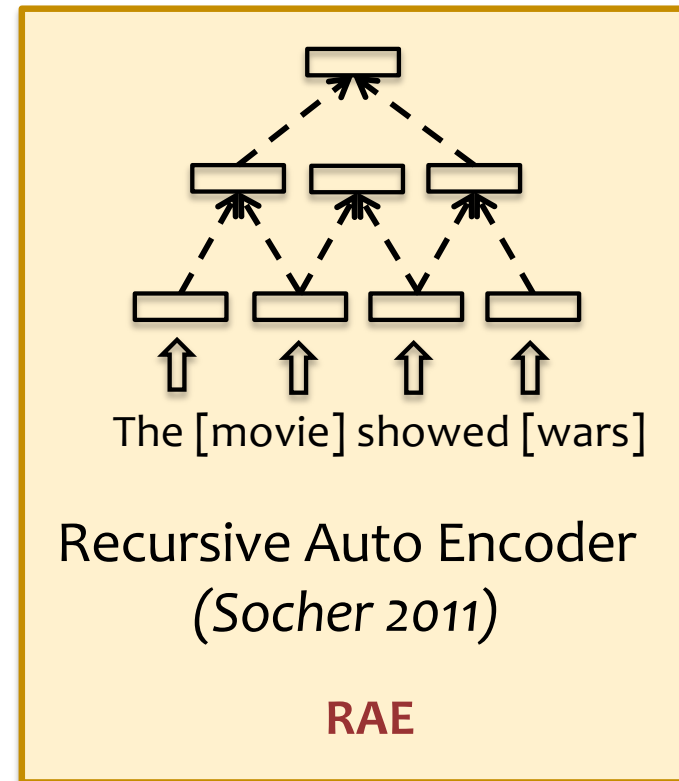
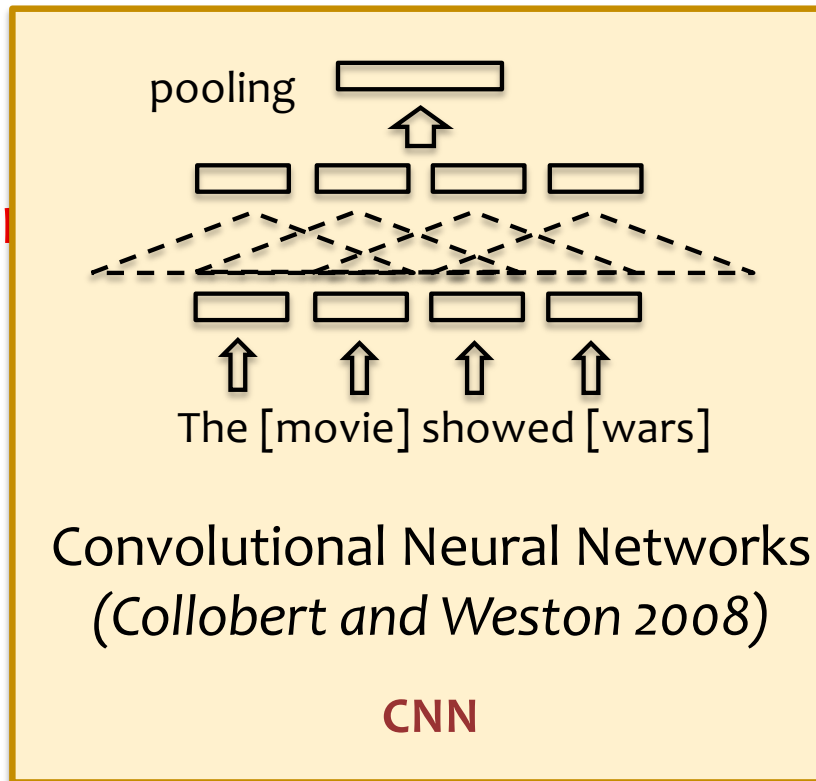


Where do features come from?



Where do features come from?

Feature Engineering



Zhou et al.,
2005



**word
embeddings**
Mikolov et al.,
2013

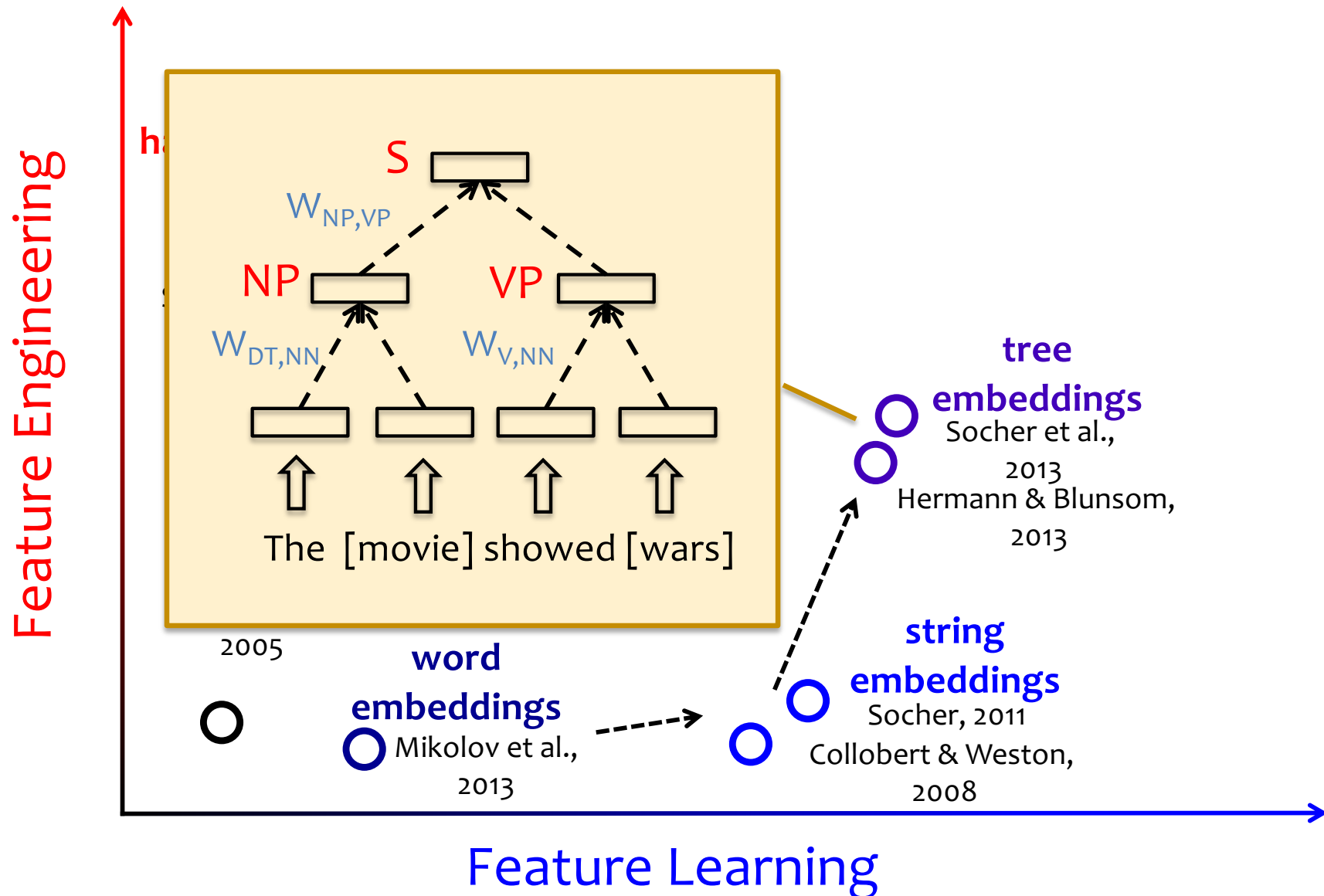


**string
embeddings**
Socher, 2011
Collobert & Weston,
2008

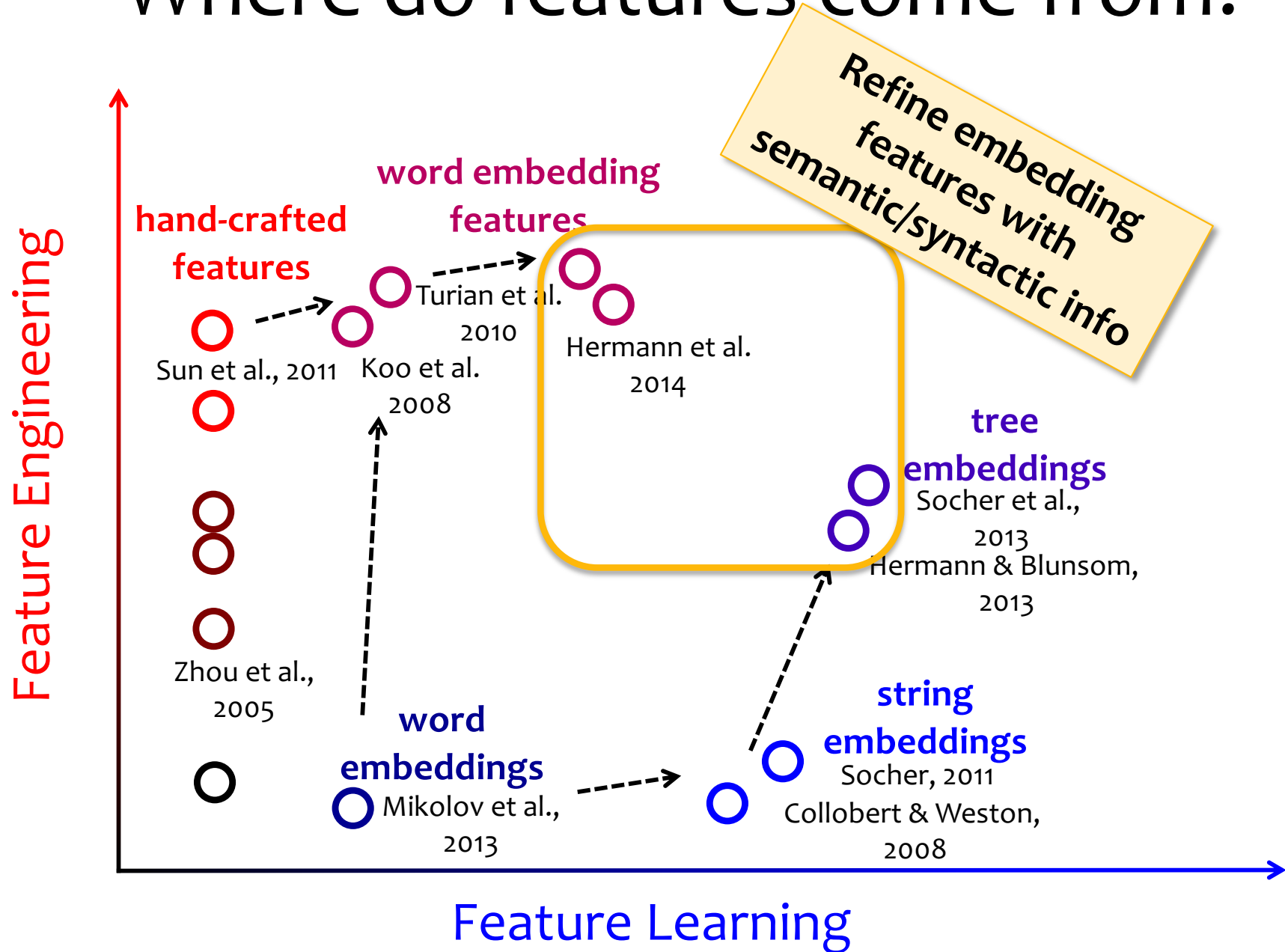


Feature Learning

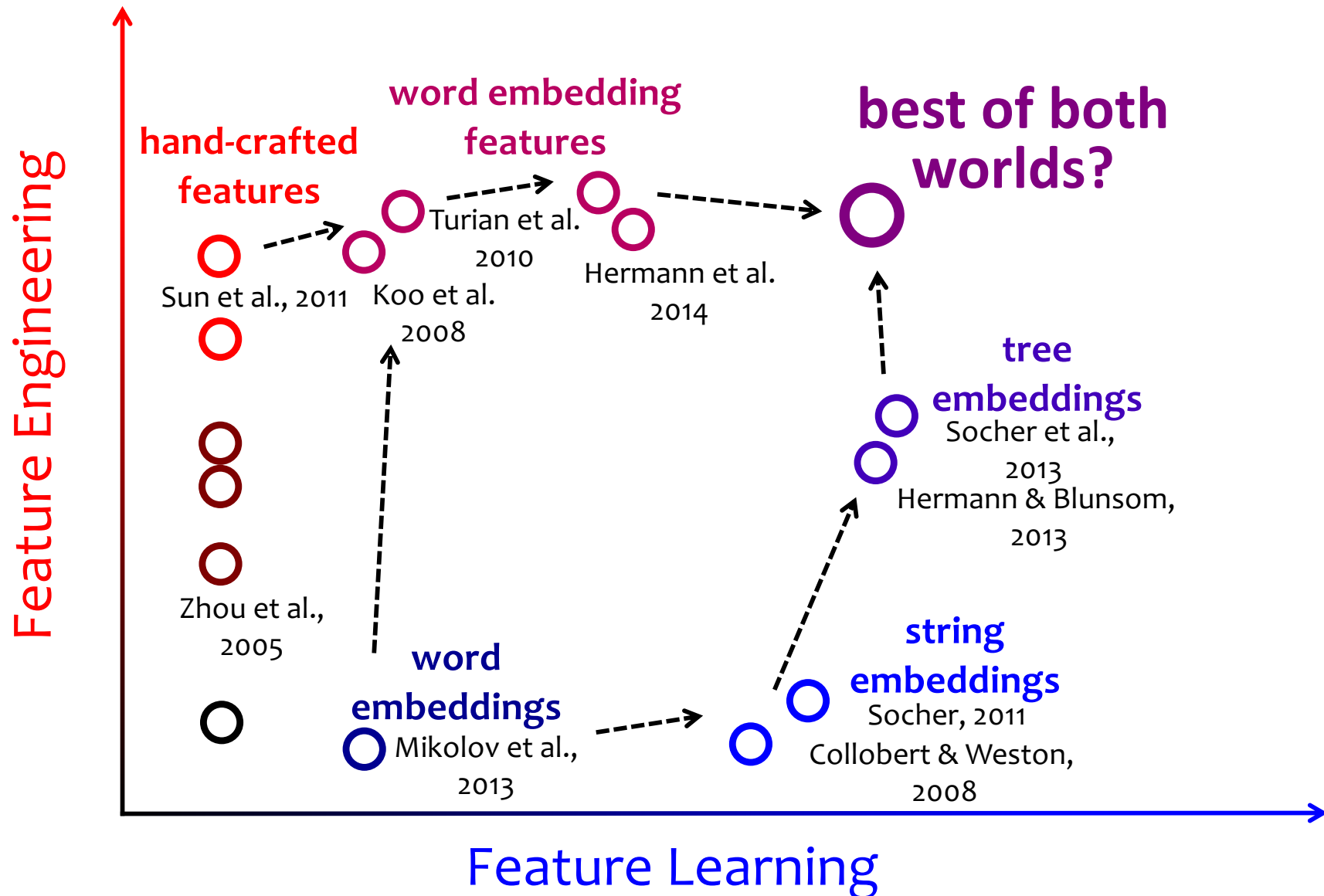
Where do features come from?



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(w_i)</code>	1	0	1	0	0	0
<code>endswith(w_i, "e")</code>	1	1	0	0	0	1
<code>endswith(w_i, "d")</code>	0	0	0	1	1	0
<code>endswith(w_i, "ed")</code>	0	0	0	1	1	0
<code>$w_i == \text{"aardvark"}$</code>	0	0	0	0	0	0
<code>$w_i == \text{"hope"}$</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:

	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$
...
$W_i == "I"$	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

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%

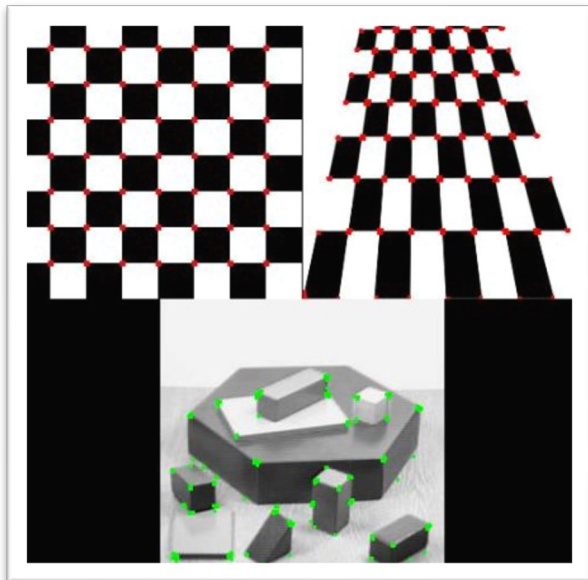
deter. noun noun verb verb noun
The movie I watched depicted hope

Feature Engineering for CV

Edge detection (Canny)

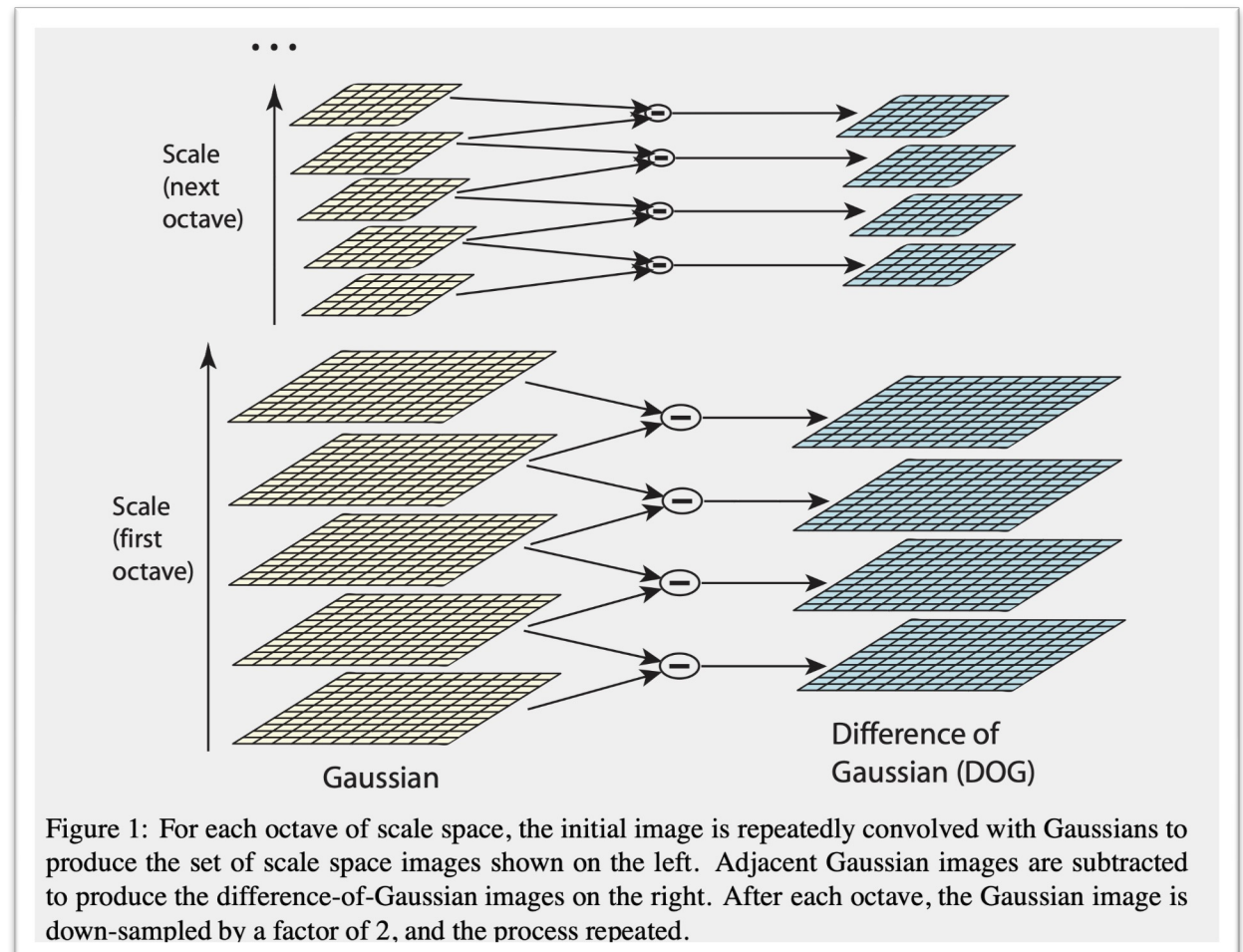


Corner Detection (Harris)



Feature Engineering for CV

Scale Invariant Feature Transform (SIFT)



Feature Engineering

Question:

Suppose you are building a linear regression model to predict the construction cost of all houses built in Pittsburgh in the 1930s – 1940s (in 30/40s dollars).

What features would you use?

Answer:

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: $\mathbf{x} \in \mathbb{R}^M$ where $M' > M$ (usually)
 - new input: $\mathbf{x}' \in \mathbb{R}^{M'}$
 - define $\mathbf{x}' = b(\mathbf{x}) = [b_1(\mathbf{x}), b_2(\mathbf{x}), \dots, b_{M'}(\mathbf{x})]$
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}

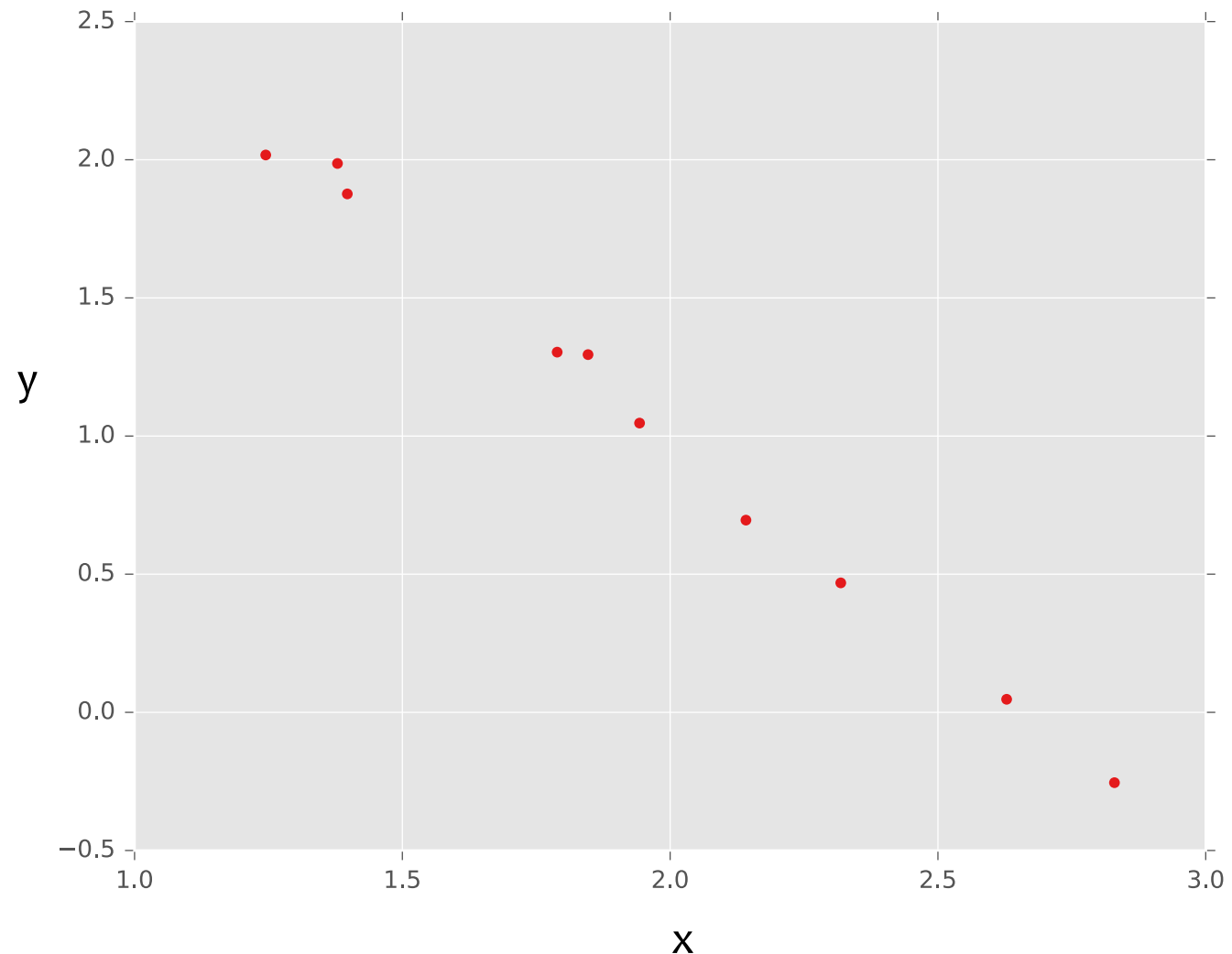
Examples:

- Perceptron
- Linear regression
- Logistic regression

Example: Linear Regression

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

i	y	x
1	2.0	1.2
2	1.3	1.7
...
10	1.1	1.9

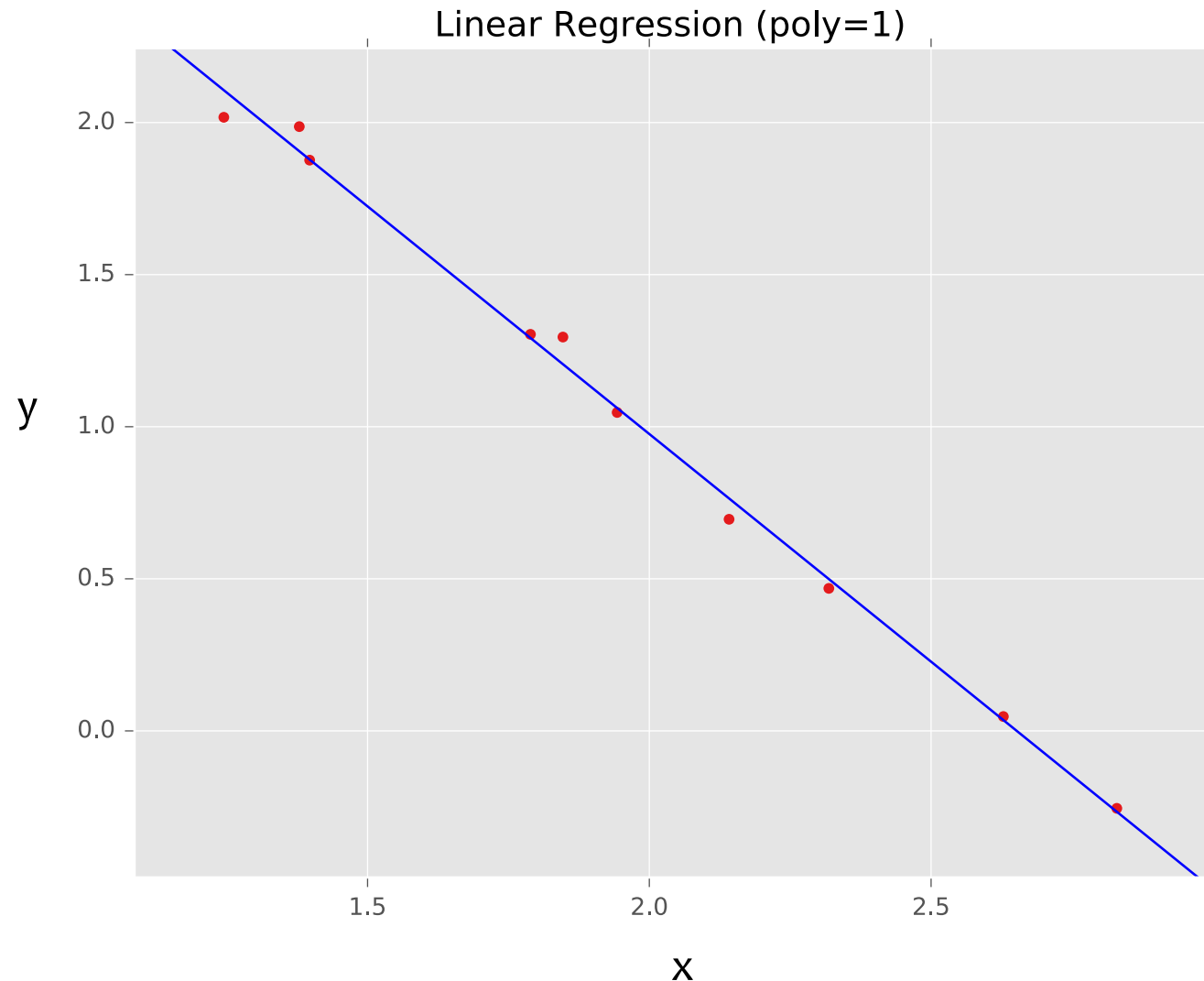


true “unknown”
target function is
linear with
negative slope
and gaussian
noise

Example: Linear Regression

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

i	y	x
1	2.0	1.2
2	1.3	1.7
...
10	1.1	1.9

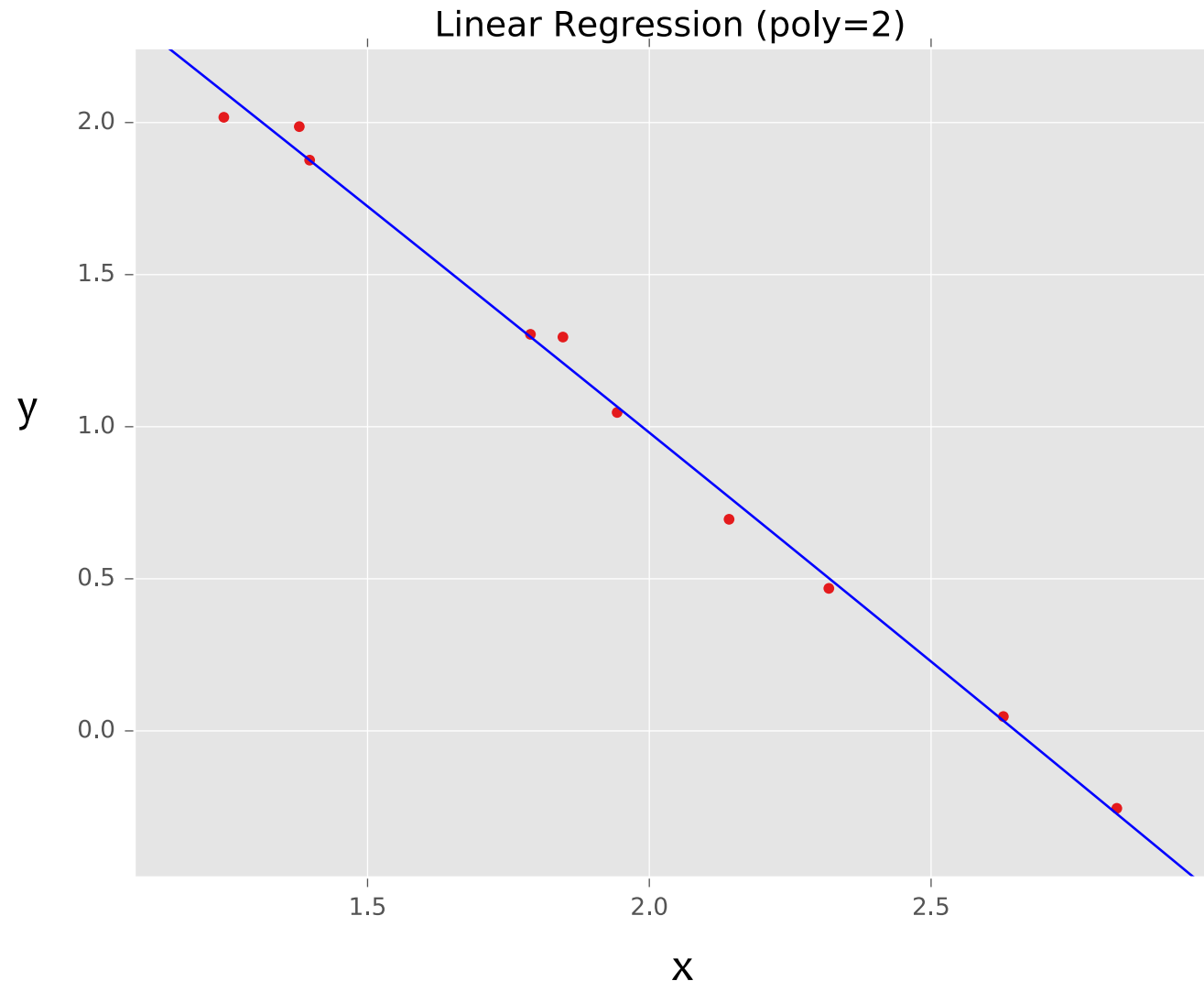


true “unknown”
target function is
linear with
negative slope
and gaussian
noise

Example: Linear Regression

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

i	y	x	x^2
1	2.0	1.2	$(1.2)^2$
2	1.3	1.7	$(1.7)^2$
...
10	1.1	1.9	$(1.9)^2$



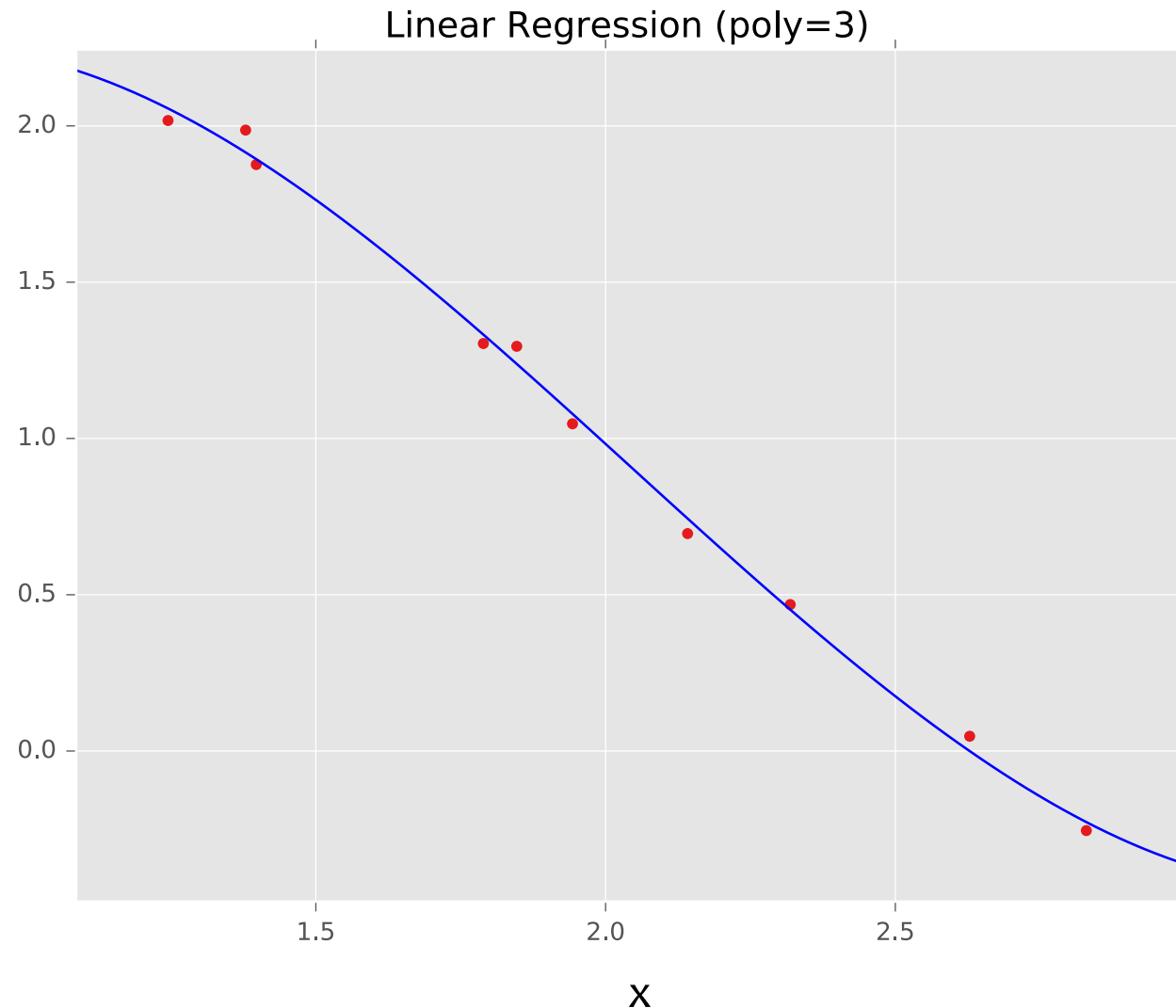
true “unknown”
target function is
linear with
negative slope
and gaussian
noise

Example: Linear Regression

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

i	y	x	x^2	x^3
1	2.0	1.2	$(1.2)^2$	$(1.2)^3$
2	1.3	1.7	$(1.7)^2$	$(1.7)^3$
...
10	1.1	1.9	$(1.9)^2$	$(1.9)^3$

y



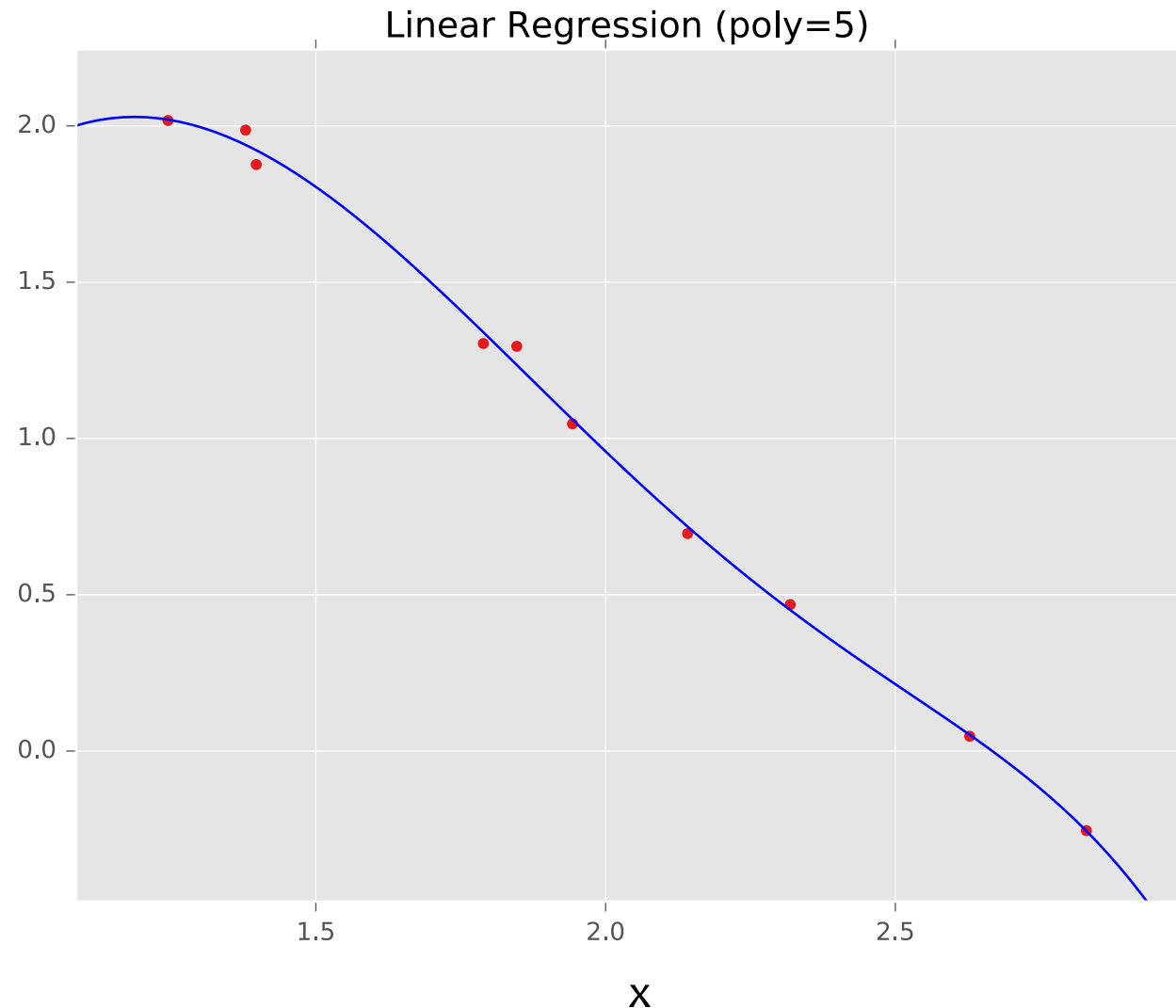
true “unknown”
target function is
linear with
negative slope
and gaussian
noise

Example: Linear Regression

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

i	y	x	...	x^5
1	2.0	1.2	...	$(1.2)^5$
2	1.3	1.7	...	$(1.7)^5$
...
10	1.1	1.9	...	$(1.9)^5$

y

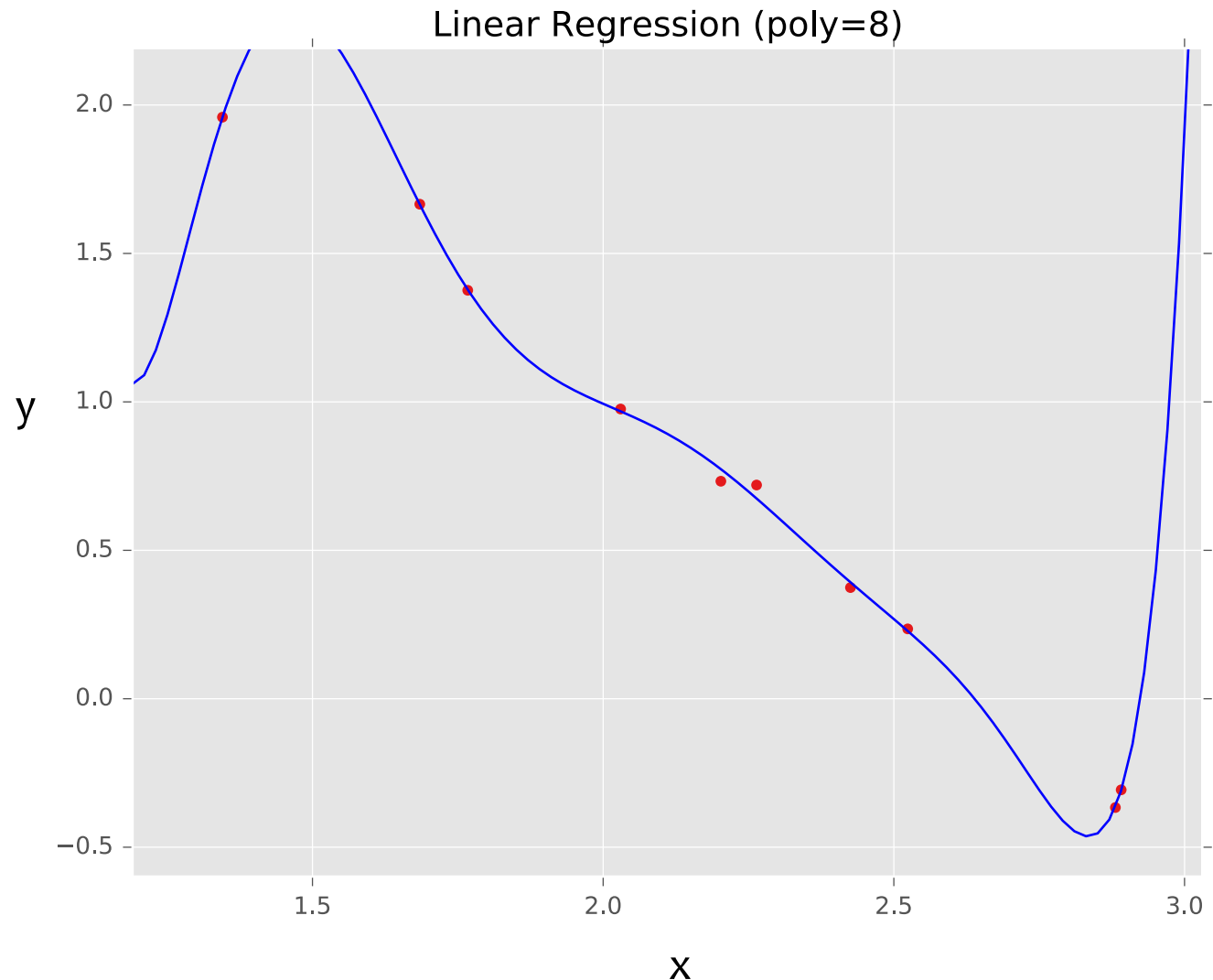


true “unknown”
target function is
linear with
negative slope
and gaussian
noise

Example: Linear Regression

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

i	y	x	...	x^8
1	2.0	1.2	...	$(1.2)^8$
2	1.3	1.7	...	$(1.7)^8$
...
10	1.1	1.9	...	$(1.9)^8$

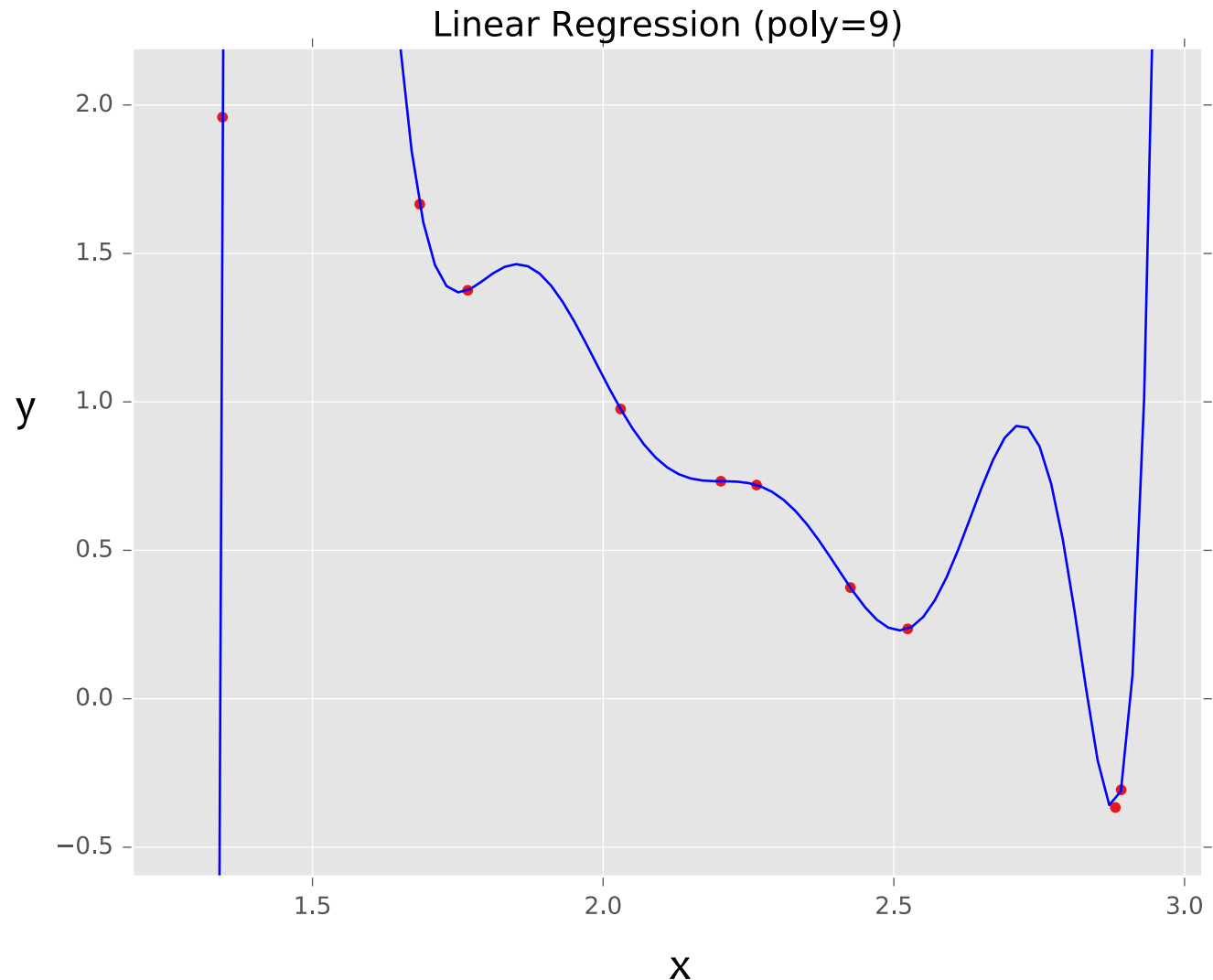


true “unknown”
target function is
linear with
negative slope
and gaussian
noise

Example: Linear Regression

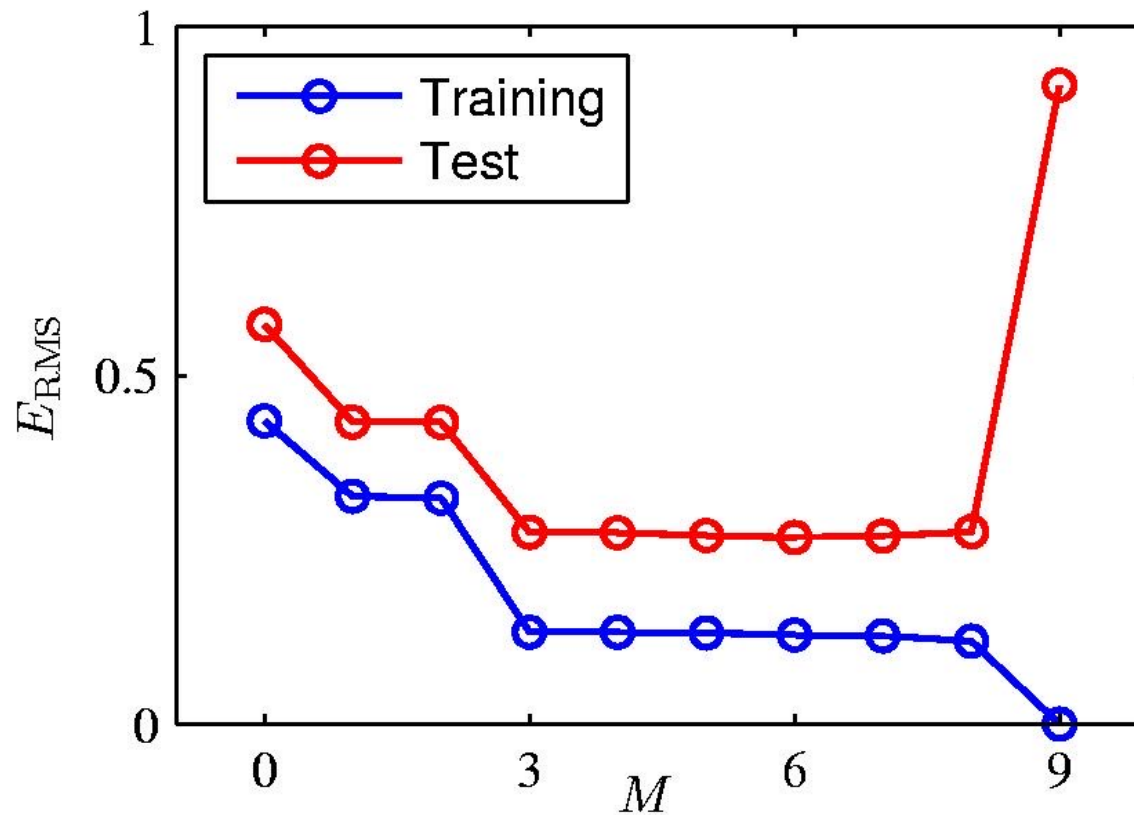
Goal: Learn $y = \mathbf{w}^T \mathbf{f}(\mathbf{x}) + b$
where $\mathbf{f}(\cdot)$ is a polynomial
basis function

i	y	x	...	x^9
1	2.0	1.2	...	$(1.2)^9$
2	1.3	1.7	...	$(1.7)^9$
...
10	1.1	1.9	...	$(1.9)^9$



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

Over-fitting



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

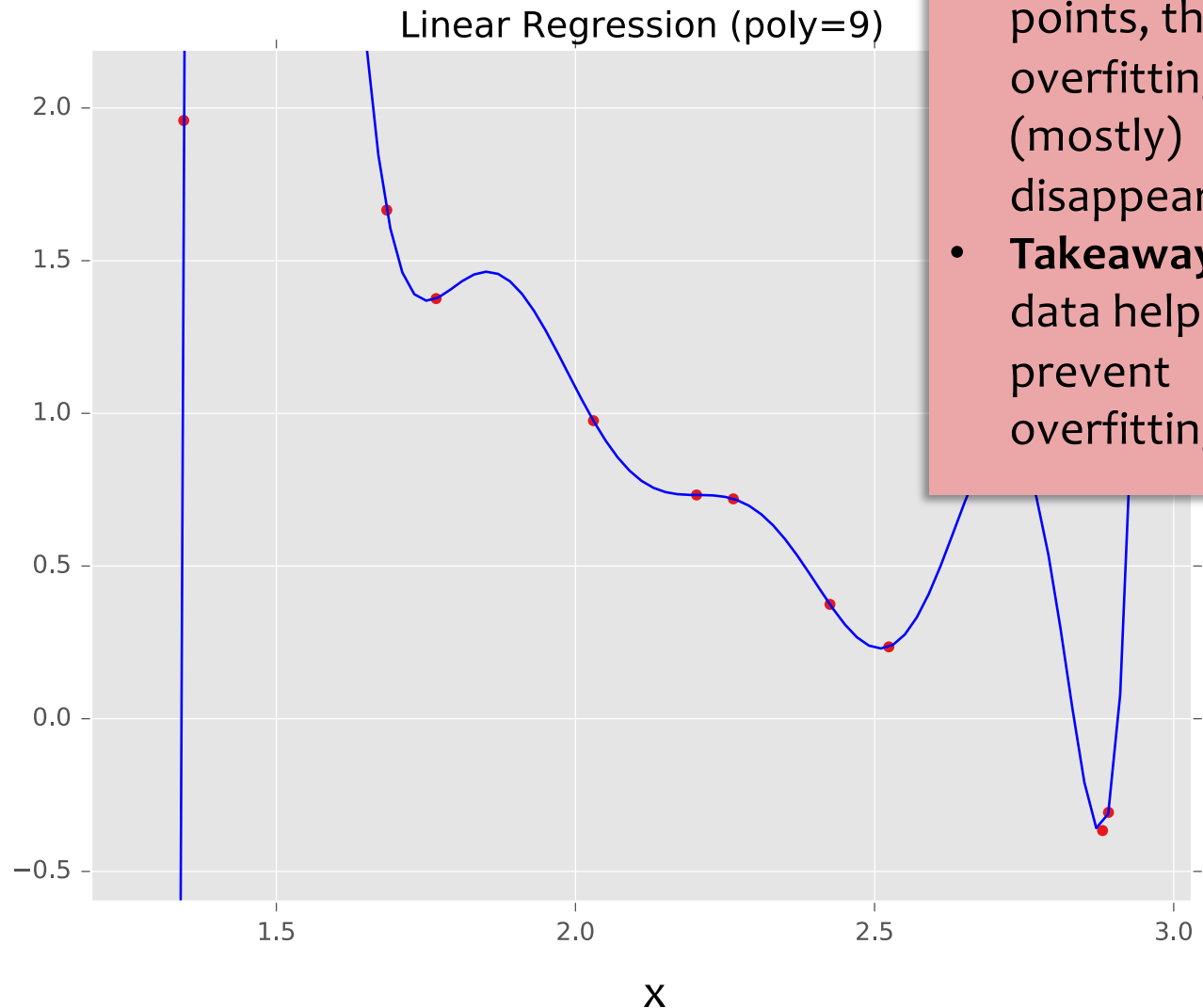
Polynomial Coefficients

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

Example: Linear Regression

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

i	y	x	...	x^9
1	2.0	1.2	...	$(1.2)^9$
2	1.3	1.7	...	$(1.7)^9$
...
10	1.1	1.9	...	$(1.9)^9$

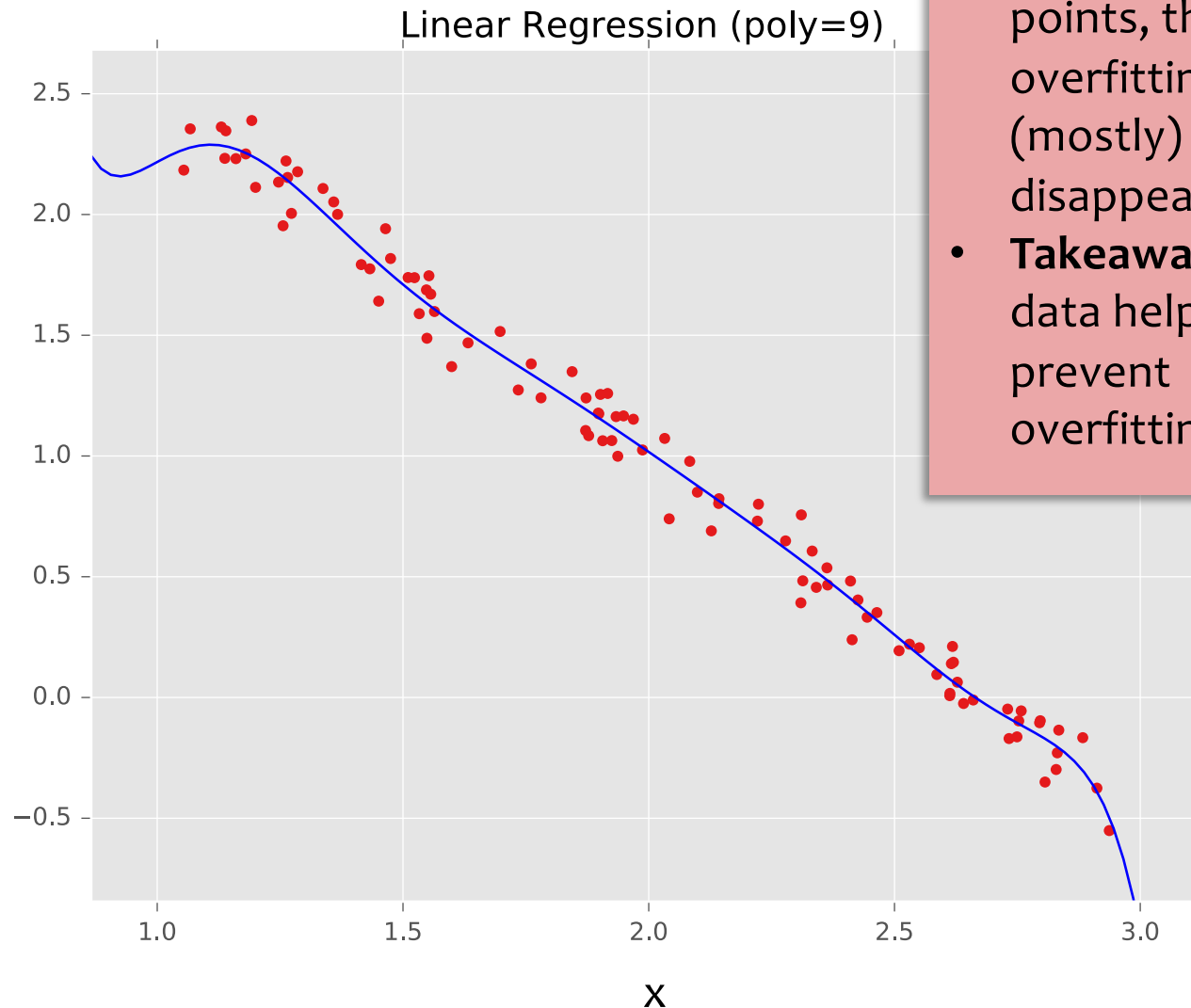


- With just $N = 10$ points we overfit!
- But with $N = 100$ points, the overfitting (mostly) disappears
- **Takeaway:** more data helps prevent overfitting

Example: Linear Regression

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

i	y	x	...	x^9
1	2.0	1.2	...	$(1.2)^9$
2	1.3	1.7	...	$(1.7)^9$
3	0.1	2.7	...	$(2.7)^9$
4	1.1	1.9	...	$(1.9)^9$
...
...
...
98
99
100	0.9	1.5	...	$(1.5)^9$



- With just $N = 10$ points we overfit!
- But with $N = 100$ points, the overfitting (mostly) disappears
- **Takeaway:** more data helps prevent overfitting

REGULARIZATION

Overfitting

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

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

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

Motivation: Regularization

- **Occam's Razor:** prefer the simplest hypothesis
- What does it mean for a hypothesis (or model) to be **simple**?
 1. small number of features (**model selection**)
 2. small number of “important” features (**shrinkage**)

Regularization

- **Given** objective function: $J(\theta)$
- **Goal** is to find: $\hat{\theta} = \underset{\theta}{\operatorname{argmin}} J(\theta) + \lambda r(\theta)$
- **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): $\|\theta\|_q = \left(\sum_{m=1}^M |\theta_m|^q \right)^{\frac{1}{q}}$

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 Examples

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

$$\begin{aligned} J_{\text{RR}}(\boldsymbol{\theta}) &= J(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_2^2 \\ &= \frac{1}{N} \sum_{i=1}^N \frac{1}{2} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2 + \lambda \sum_{m=1}^M \theta_m^2 \end{aligned}$$

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

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

Regularization Examples

Add an **L2 regularizer** to Logistic Regression

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

Add an **L1 regularizer** to Logistic Regression

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

Regularization

Question:

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

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

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

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

