10-301/10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

(Linear Models)
+ Feature Engineering
+ Regularization

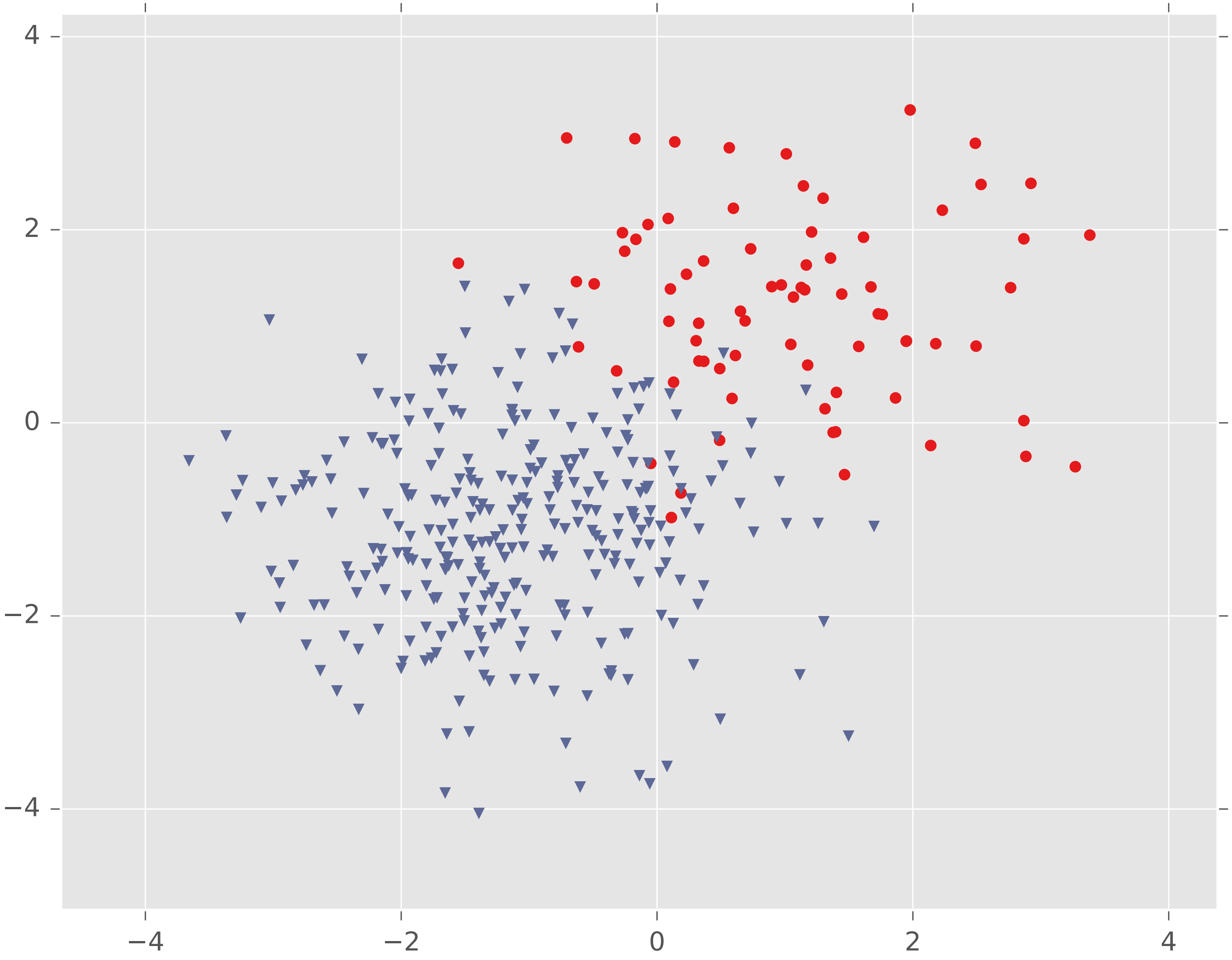
Matt Gormley
Lecture 10
Feb. 20, 2023

Reminders

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

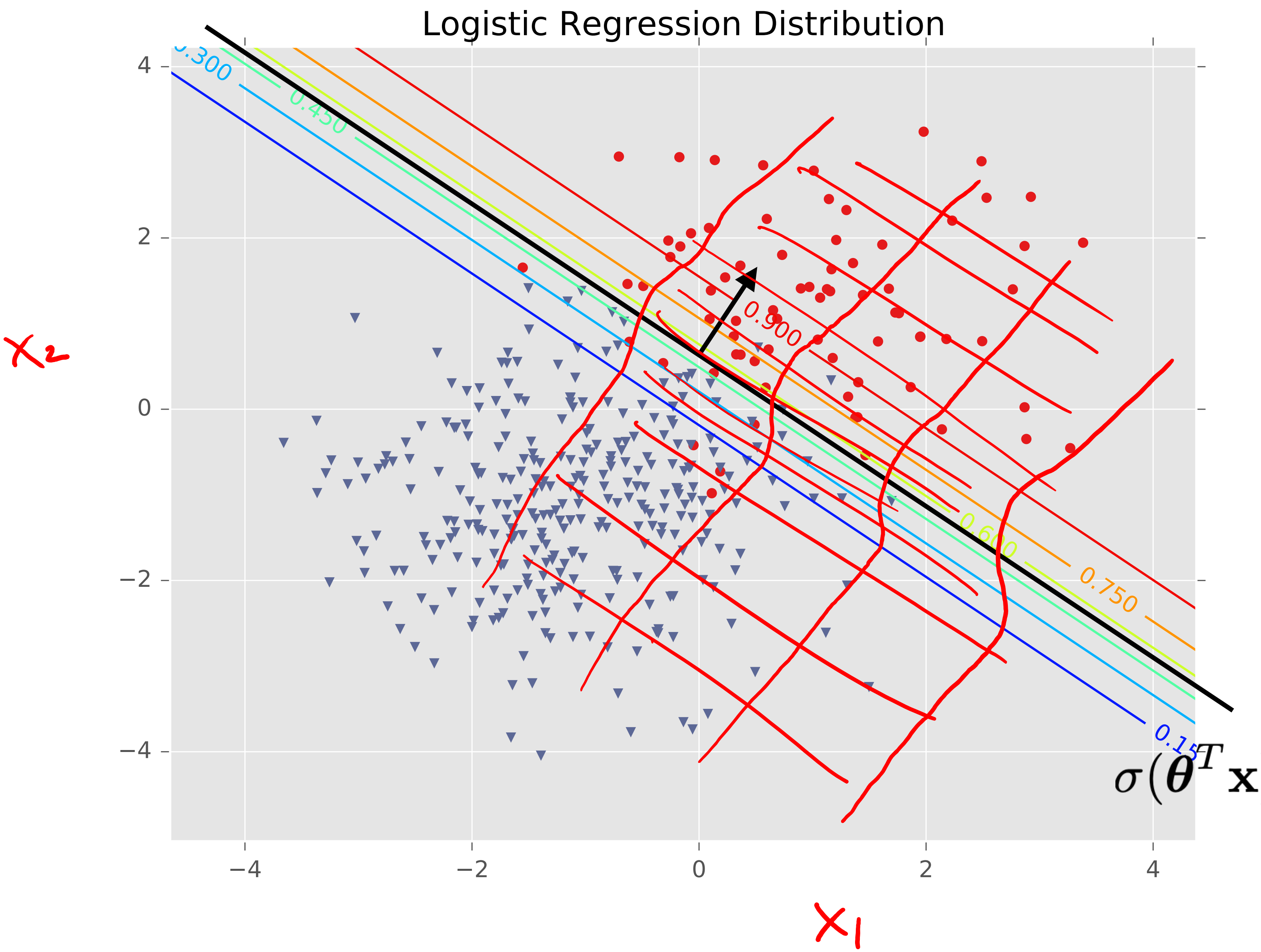
LOGISTIC REGRESSION ON GAUSSIAN DATA

Logistic Regression



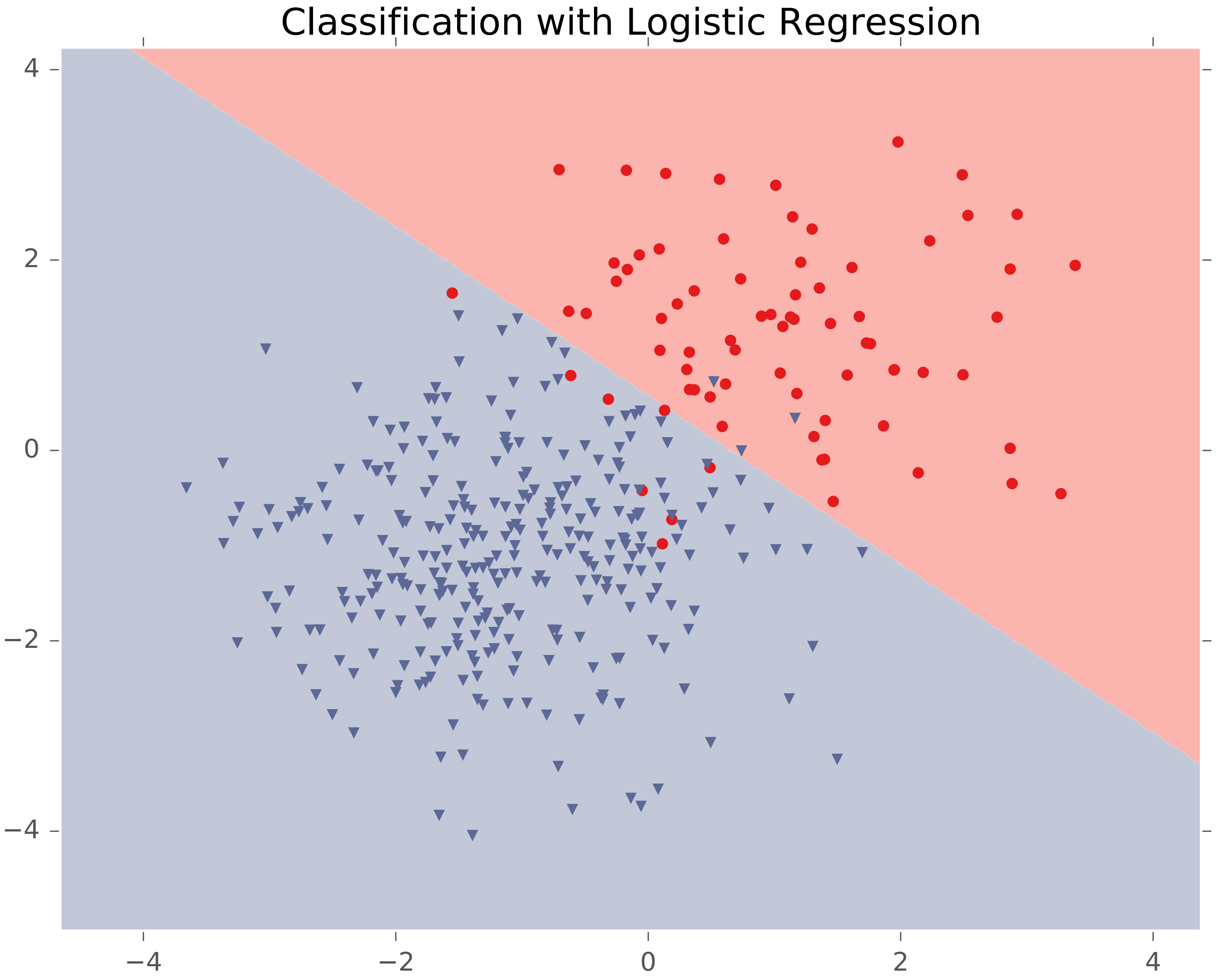
$p(y=1|\vec{x}) = \sigma(\theta^T\vec{x}) = \sigma(w^T x + b)$

Logistic Regression



$$\sigma(\boldsymbol{\theta}^T \mathbf{x}) = 0.5$$

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 actually MCLE! more on this later…)

Maximum **Conditional** Likelihood Estimation

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

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

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)~~

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

Linear Models

PERCEPTRON, LINEAR REGRESSION, AND LOGISTIC REGRESSION

poll.mlcourse.org

Matching Game

Question: Q1

Match the Algorithm to its Update Rule

1. SGD for Logistic Regression

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = p(y|x) = \sigma(\theta^T \vec{x})$$

2. Least Mean Squares

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

→ SGD for Linear Reg

3. Perceptron

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

4.

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

5.

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

6.

$$\underline{\theta_k \leftarrow \theta_k +} \lambda(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)})x_k^{(i)}$$

↑ learning rate

Answer:

A. 1=5, 2=4, 3=6

B. 1=5, 2=6, 3=4 70%

C. 1=6, 2=4, 3=4

D. 1=5, 2=6, 3=6

[✓ E.] 1=6, 2=6, 3=6 5%

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~~ → toxic

SGD for Logistic Regression

Question: Q2

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~~ → toxic

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

20% 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

60% 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

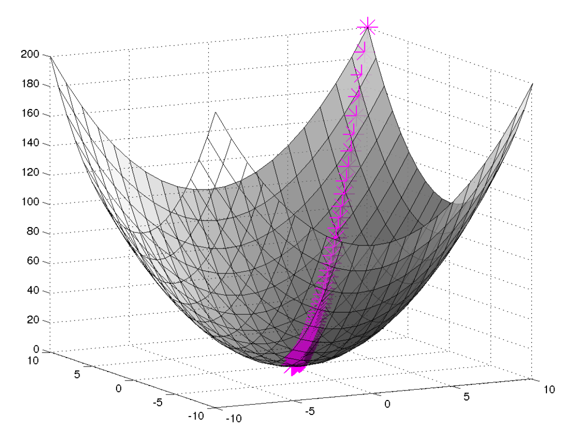
20% F. (1) randomly pick a parameter and an example, (2) compute the gradient of the log-likelihood for that example with respect to that parameter, (3) update that parameter using that gradient

Recall...

Gradient Descent

Algorithm 1 Gradient Descent

```
procedure GD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
    while not converged do
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \underline{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})}$
    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).

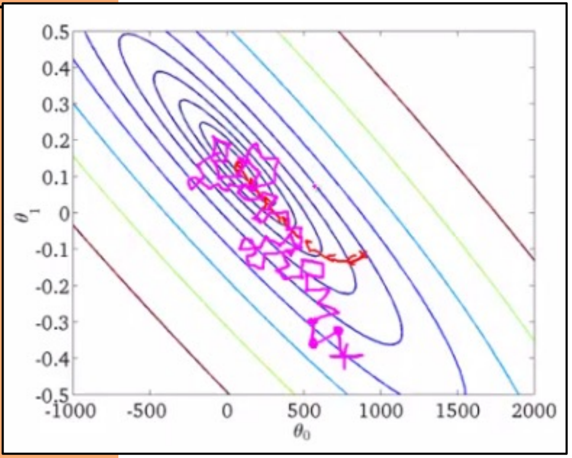
$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_M} J(\boldsymbol{\theta}) \end{bmatrix}$$

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
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \underline{\nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})}$
    return $\boldsymbol{\theta}$
```



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

We need a per-example objective:

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

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

Logistic Regression vs. Perceptron

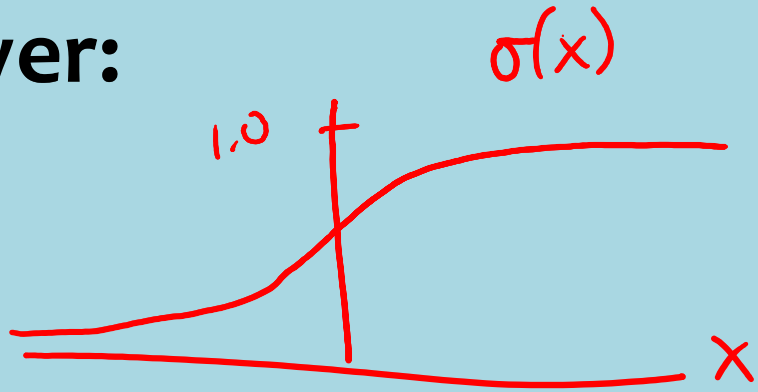
Question: Q3

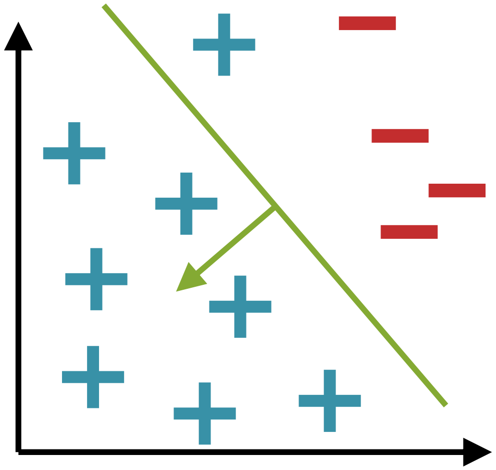
A = toxic B = True C = False

90%

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 | **x**) or had a good approximation to it.

Question:

How would you design a function y = h(**x**) to predict a single label?

Answer:

You'd use the *Bayes optimal classifier*!

ur

approximates c (**x**)

Probabilistic Learning

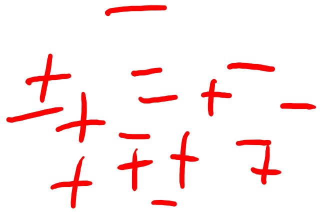
Today, we assume that our output is **sampled** from a conditional **probability distribution:**

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

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

Our goal is to learn a probability distribution p(y|**x**) that best approximates p^*(y|**x**)

Bayes Optimal Classifier



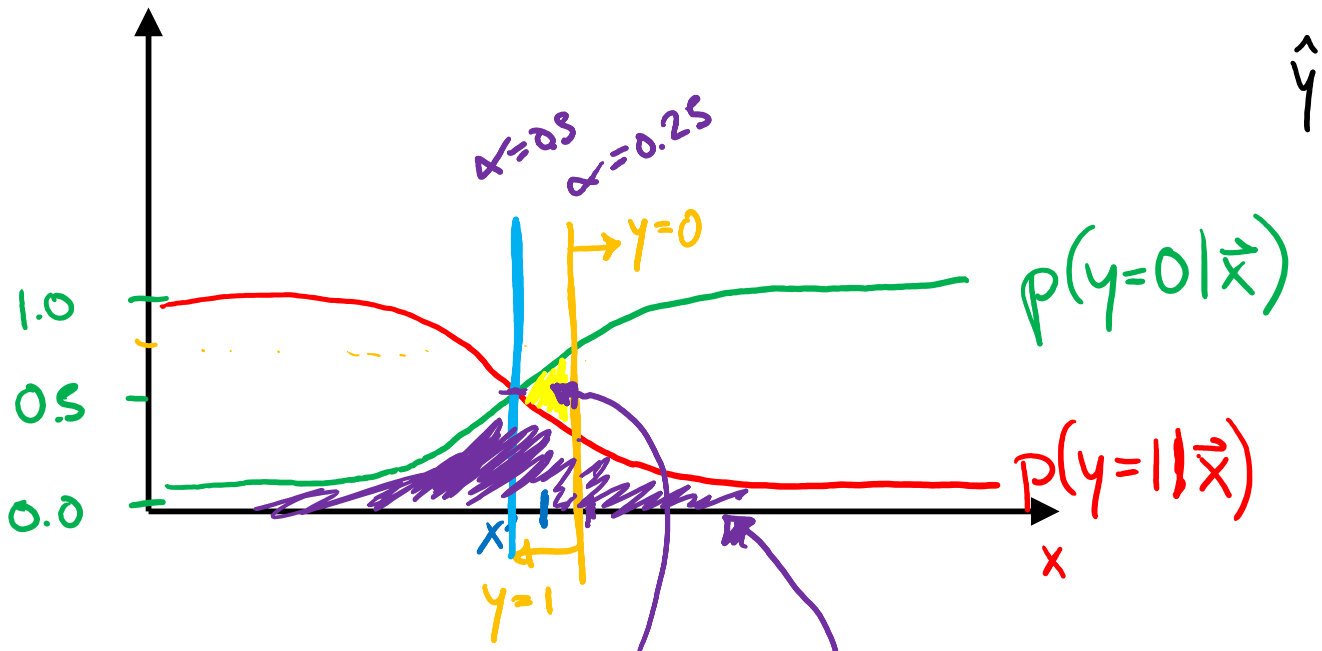
Suppose you have an **oracle** that knows the data generating distribution, p*(y|x).
Q: What is the optimal classifier in this setting?
A: The Bayes optimal classifier! This is the best classifier for the distribution p* and the loss function.

$$\ell(y,\hat{y}) = \mathbb{1}(y \neq \hat{y})$$

$$\hat{y} = h(\vec{x}) = \begin{cases} 1 & \text{if } p(y=1|\vec{x}) \geq \alpha \\ 0 & \text{otherwise} \end{cases}$$

$\alpha = 0.5$ for 0/1 Loss

$$\ell(y,\hat{y}) = \begin{cases} 1 \text{ million} & \text{if } y \neq \hat{y} \;\&\; y=1 \\ 1000 & \text{if } y \neq \hat{y} \;\&\; y=0 \\ 0 & \text{otherwise} \end{cases}$$



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

$$\boxed{\textbf{while} \text{ not converged: } \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \mathbf{g}}$$

Three variants of first-order optimization:

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

SGD: $\mathbf{g} = \boxed{\nabla J^{(i)}(\boldsymbol{\theta})}$ 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 $\forall s$

good use of GPU memory

$\{i_1, i_2, i_3, i_4\} = \{7, 23, 56, 100\}$

$S = 4$

Logistic Regression Objectives

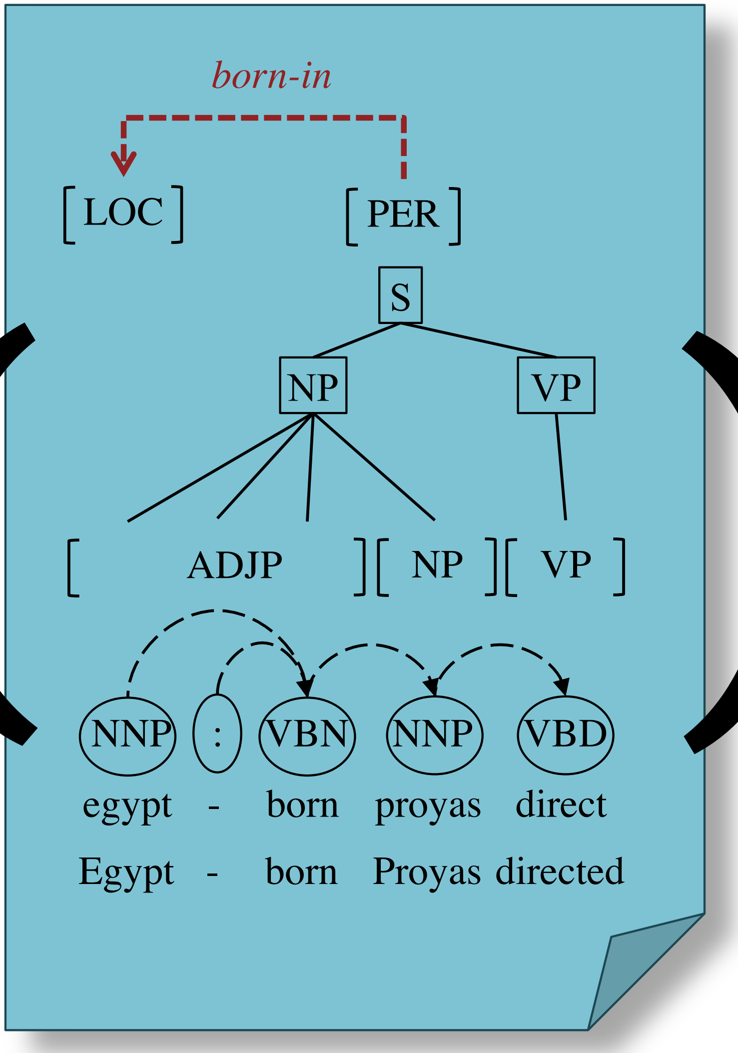
You should be able to…

- 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

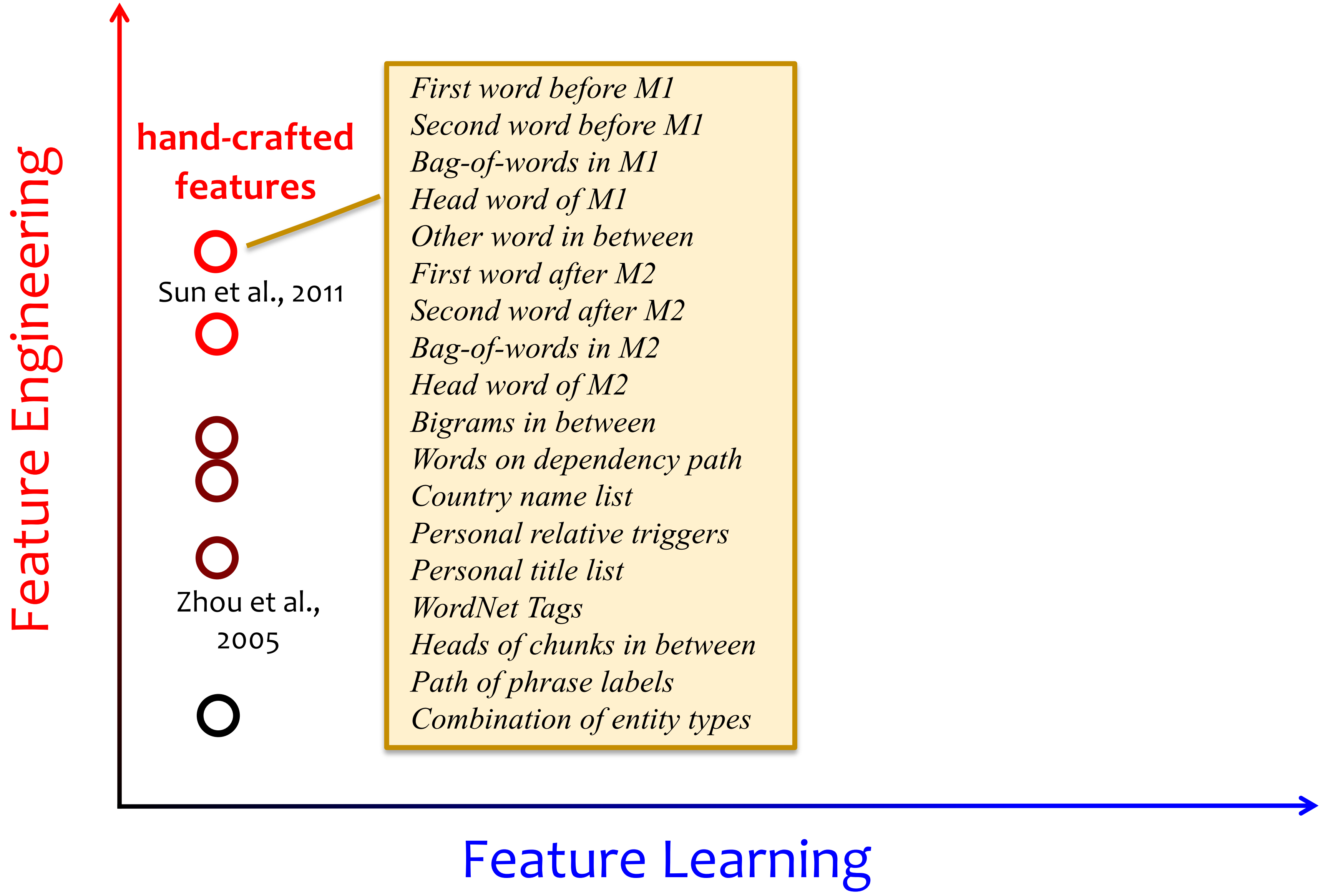
$$p(y|x) \propto \exp\left(\Theta_y \bullet f\left(\right.\right.$$



$$\left.\left.\right)\right)$$

$\vec{x}$

Where do features come from?



Feature Engineering

hand-crafted features

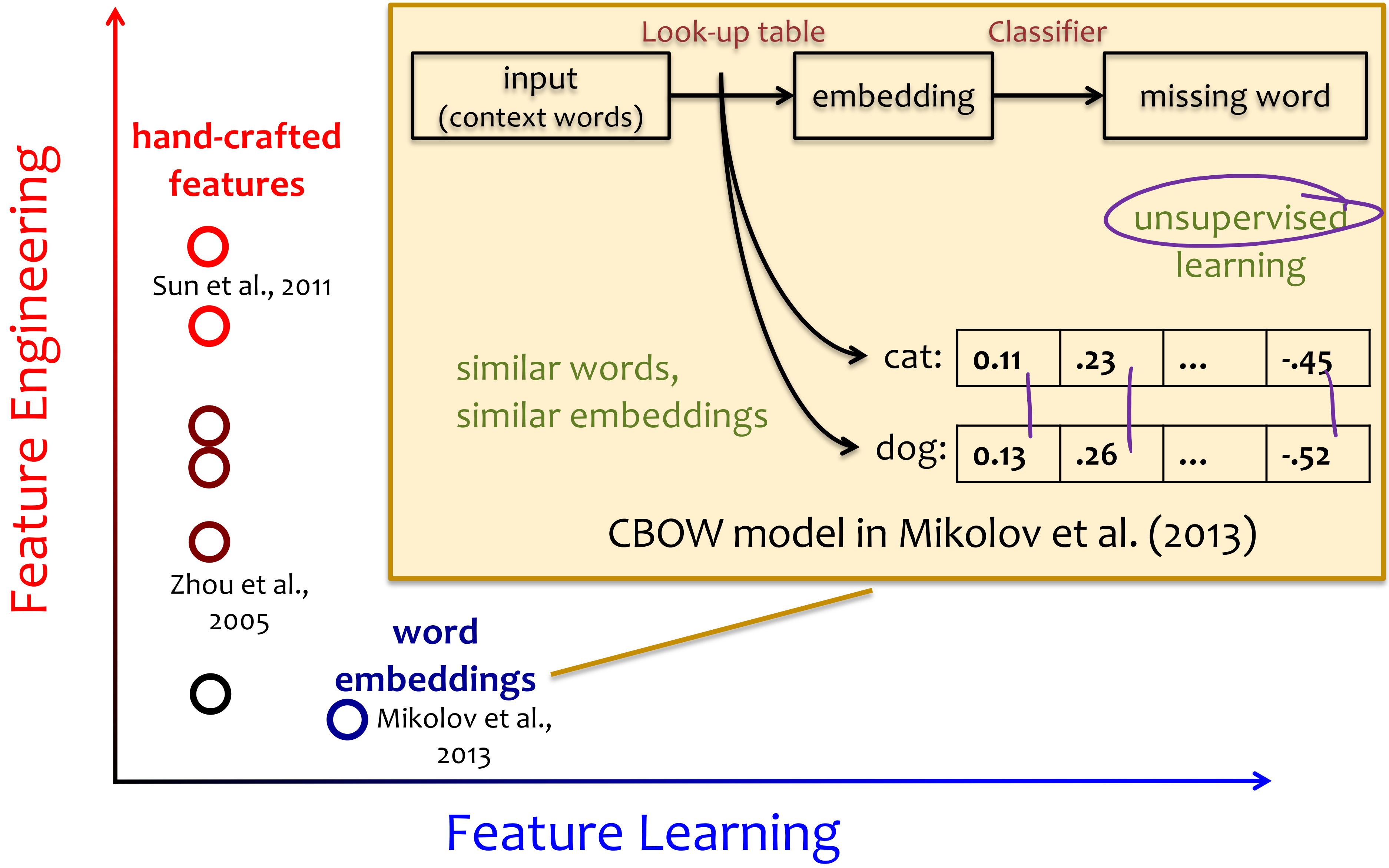
Sun et al., 2011

Zhou et al., 2005

First word before M1
Second word before M1
Bag-of-words in M1
Head word of M1
Other word in between
First word after M2
Second word after M2
Bag-of-words in M2
Head word of M2
Bigrams in between
Words on dependency path
Country name list
Personal relative triggers
Personal title list
WordNet Tags
Heads of chunks in between
Path of phrase labels
Combination of entity types

Feature Learning

Where do features come from?



Feature Engineering

hand-crafted features

Sun et al., 2011

Zhou et al., 2005

word embeddings

Mikolov et al., 2013

Feature Learning

Look-up table

Classifier

input
(context words)

embedding

missing word

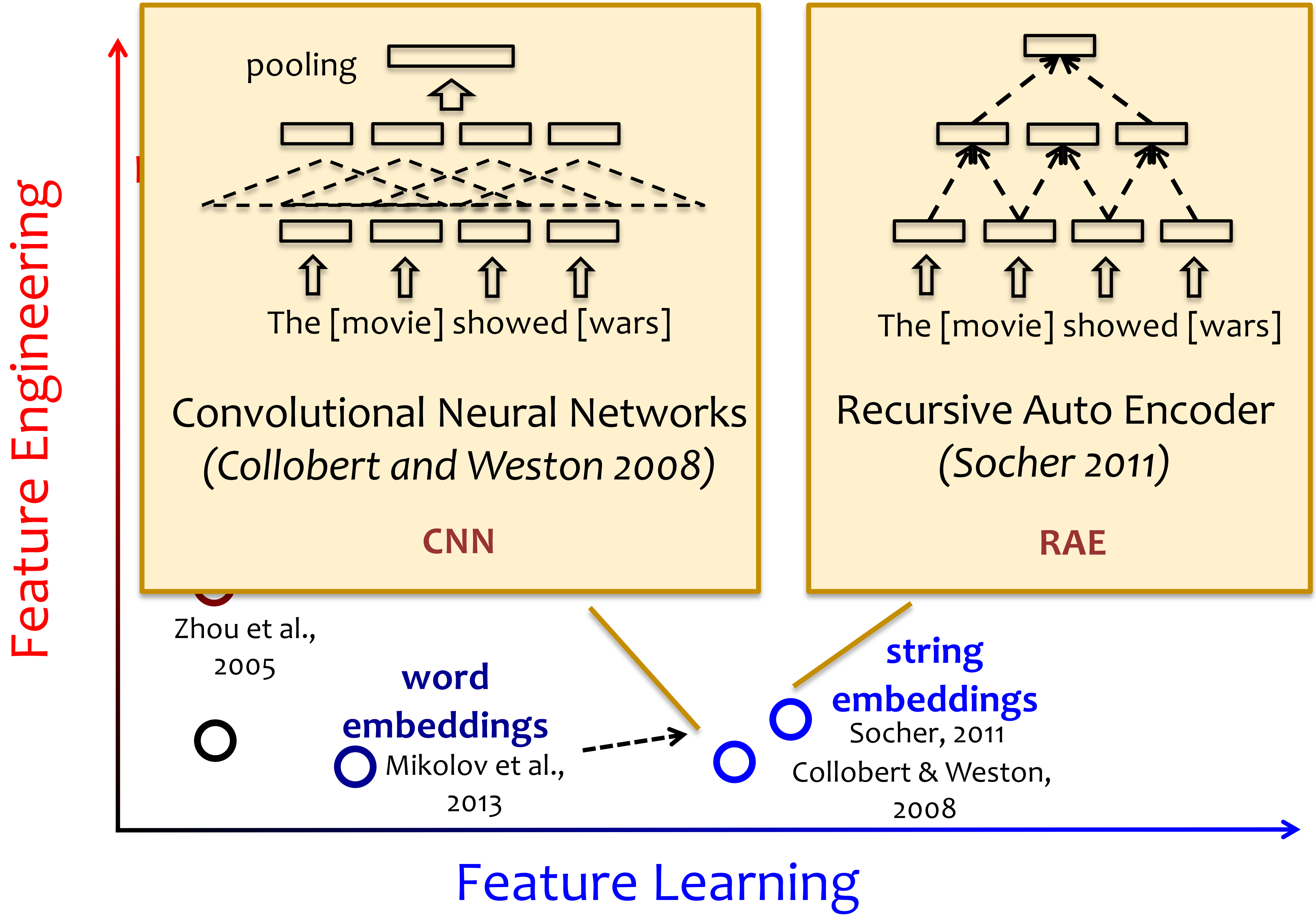
unsupervised learning

similar words,
similar embeddings

cat:	**0.11**	**.23**	**...**	**-.45**
dog:	**0.13**	**.26**	**...**	**-.52**

CBOW model in Mikolov et al. (2013)

Where do features come from?



Convolutional Neural Networks
(Collobert and Weston 2008)

CNN

Recursive Auto Encoder
(Socher 2011)

RAE

Feature Engineering

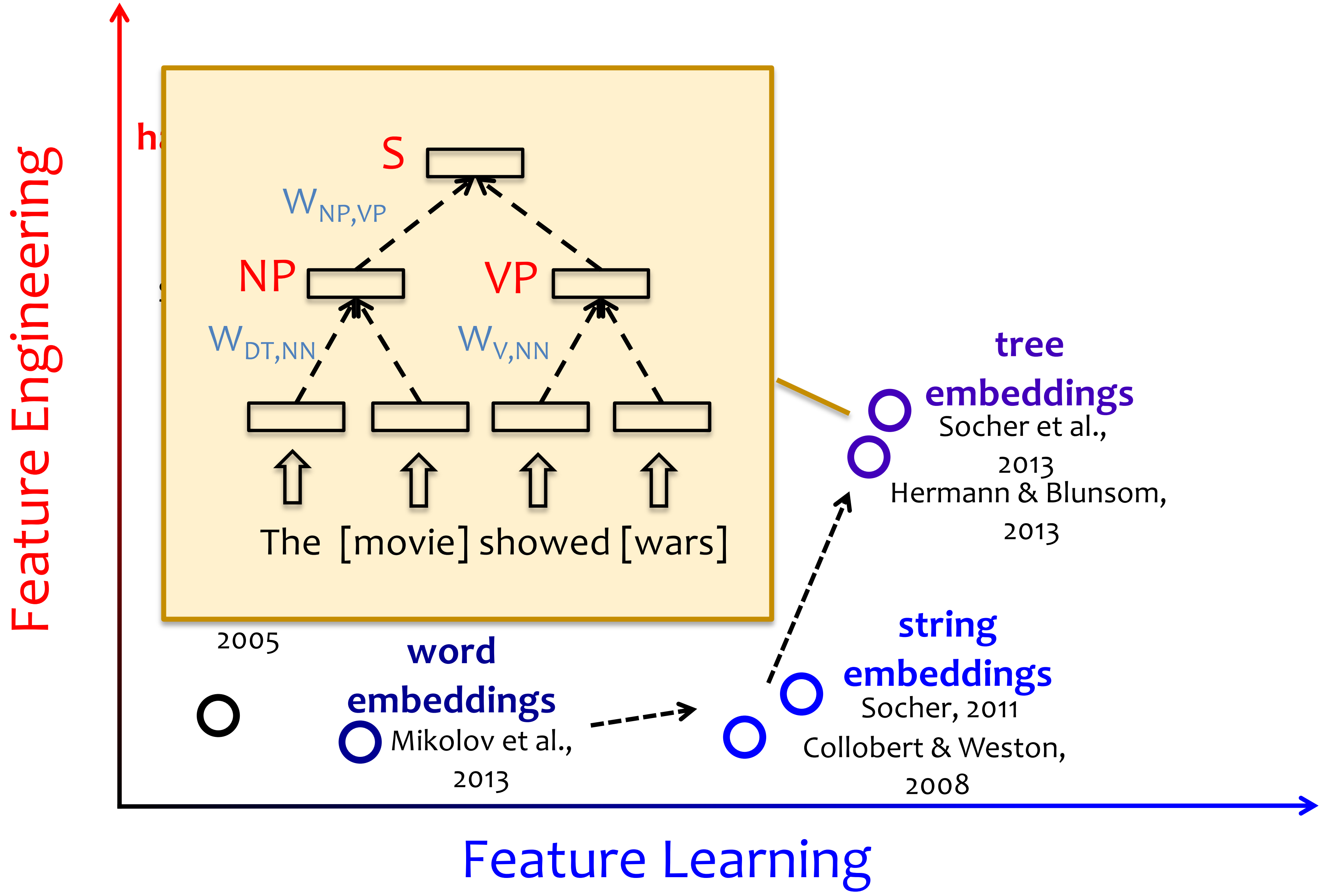
Zhou et al.,
2005

word embeddings
Mikolov et al.,
2013

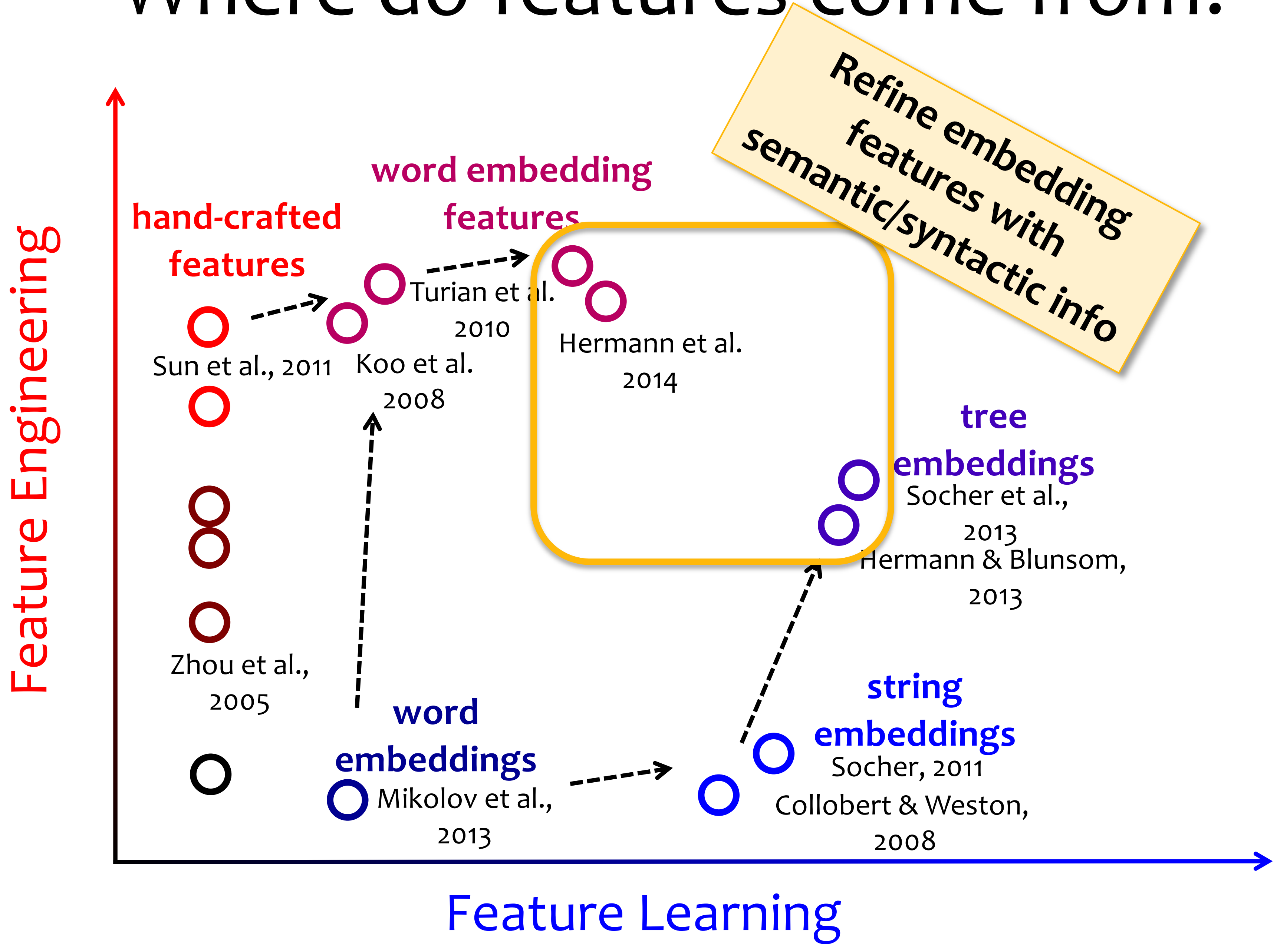
string embeddings
Socher, 2011
Collobert & Weston,
2008

Feature Learning

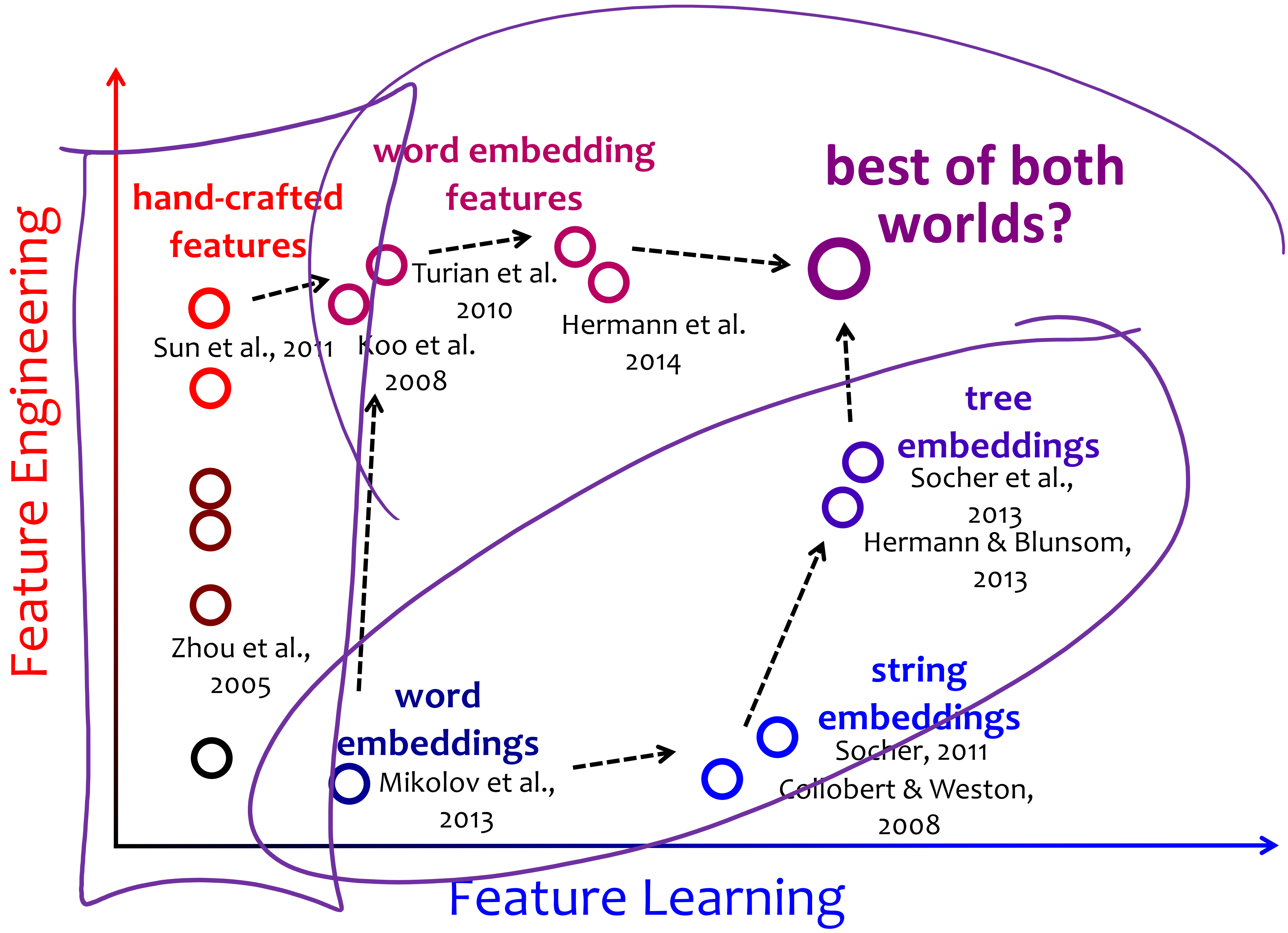
Where do features come from?



Where do features come from?



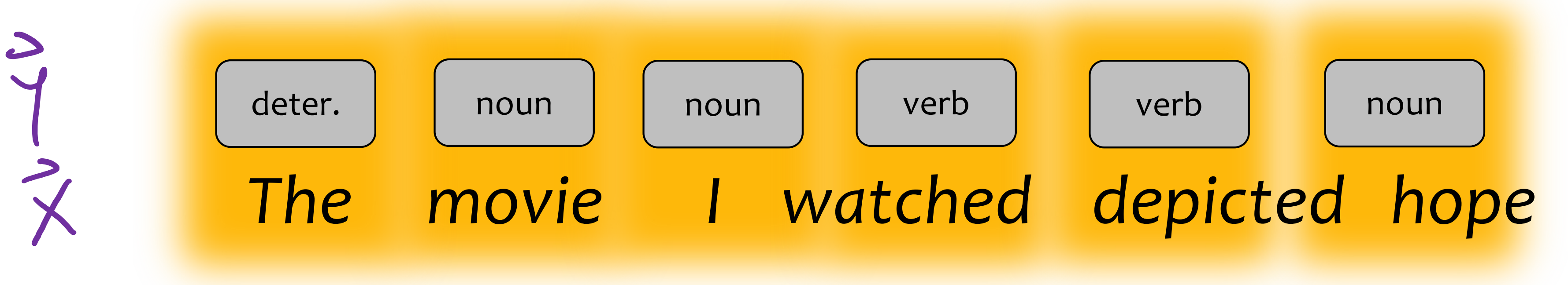
Where do features come from?



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?

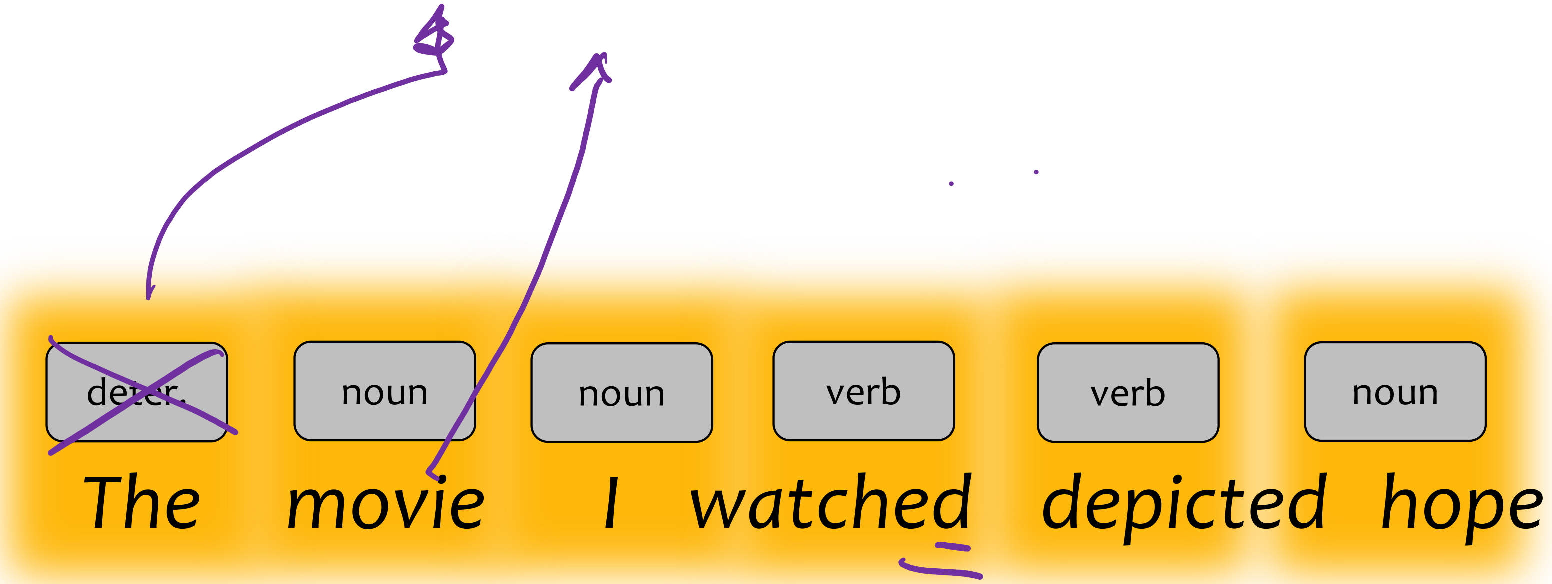


$\vec{y}$	deter.	noun	noun	verb	verb	noun
$\vec{x}$	*The*	*movie*	*I*	*watched*	*depicted*	*hope*

Feature Engineering for NLP

Per-word Features:

	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$
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
…	…	…	…	…	…	…

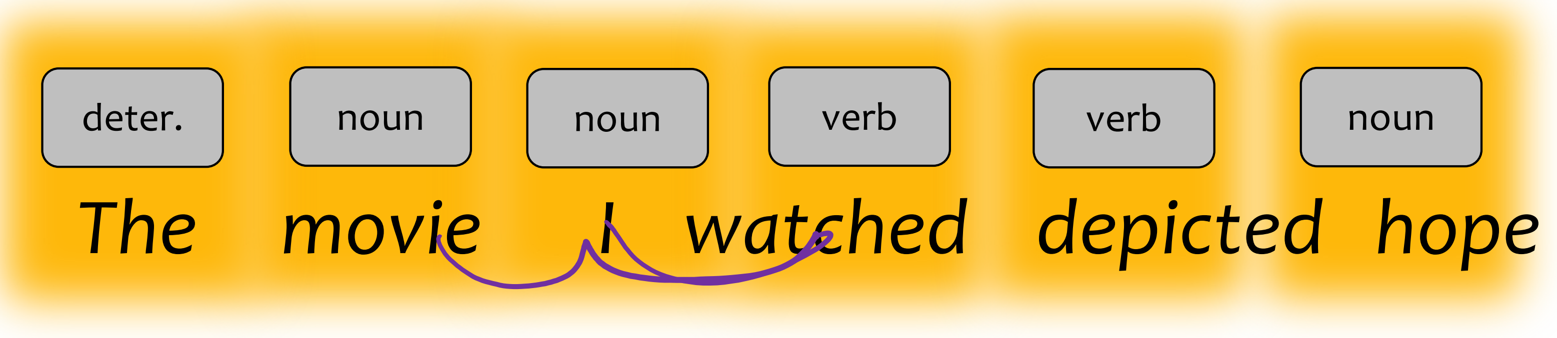


~~deter.~~	noun	noun	verb	verb	noun
The	*movie*	*I*	*watched*	*depicted*	*hope*

Feature Engineering for NLP

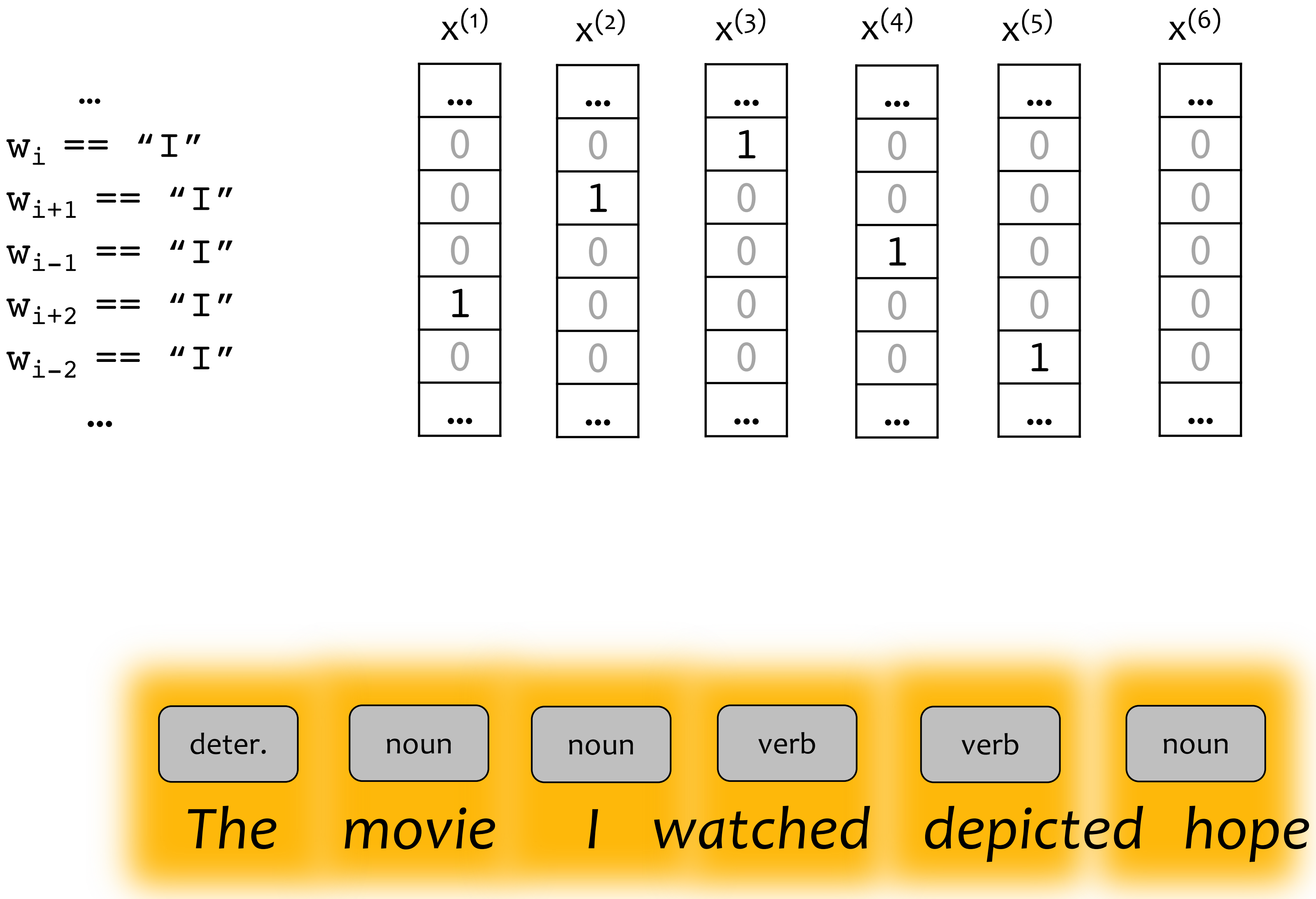
Context Features:

	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$
…	…	…	…	…	…	…
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:



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

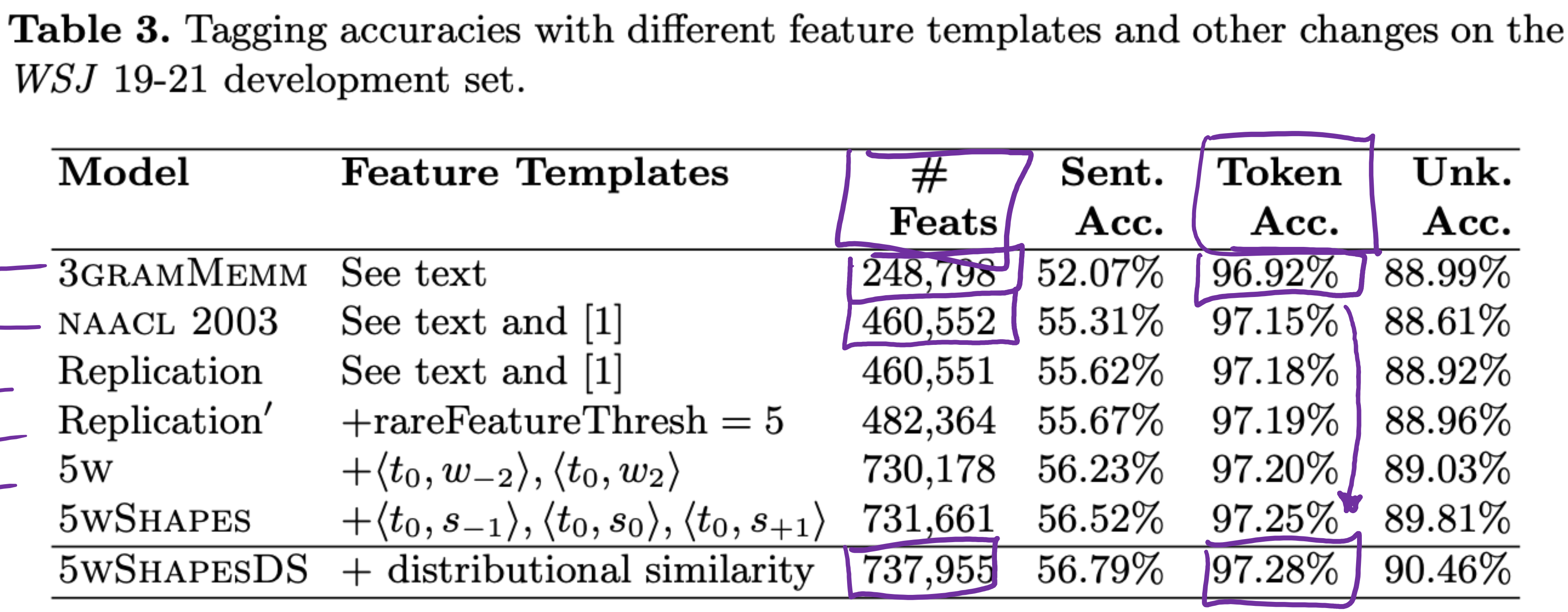


Table 3. Tagging accuracies with different feature templates and other changes on the *WSJ* 19-21 development set.

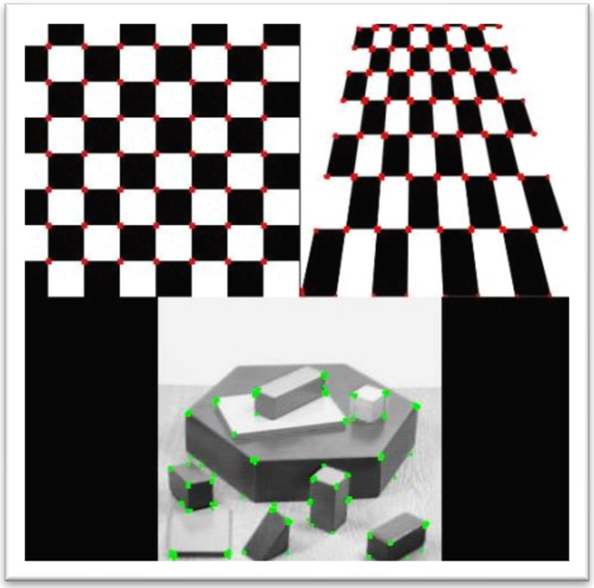
Model	Feature Templates	# Feats	Sent. Acc.	Token Acc.	Unk. Acc.
3GramMemm	See text	248,798	52.07%	96.92%	88.99%
NAACL 2003	See text and [1]	460,552	55.31%	97.15%	88.61%
Replication	See text and [1]	460,551	55.62%	97.18%	88.92%
Replication′	+rareFeatureThresh = 5	482,364	55.67%	97.19%	88.96%
5w	$+\langle t_0, w_{-2}\rangle, \langle t_0, w_2\rangle$	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%

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

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 op row. Recognition results below show model outlines and mage keys used for matching.

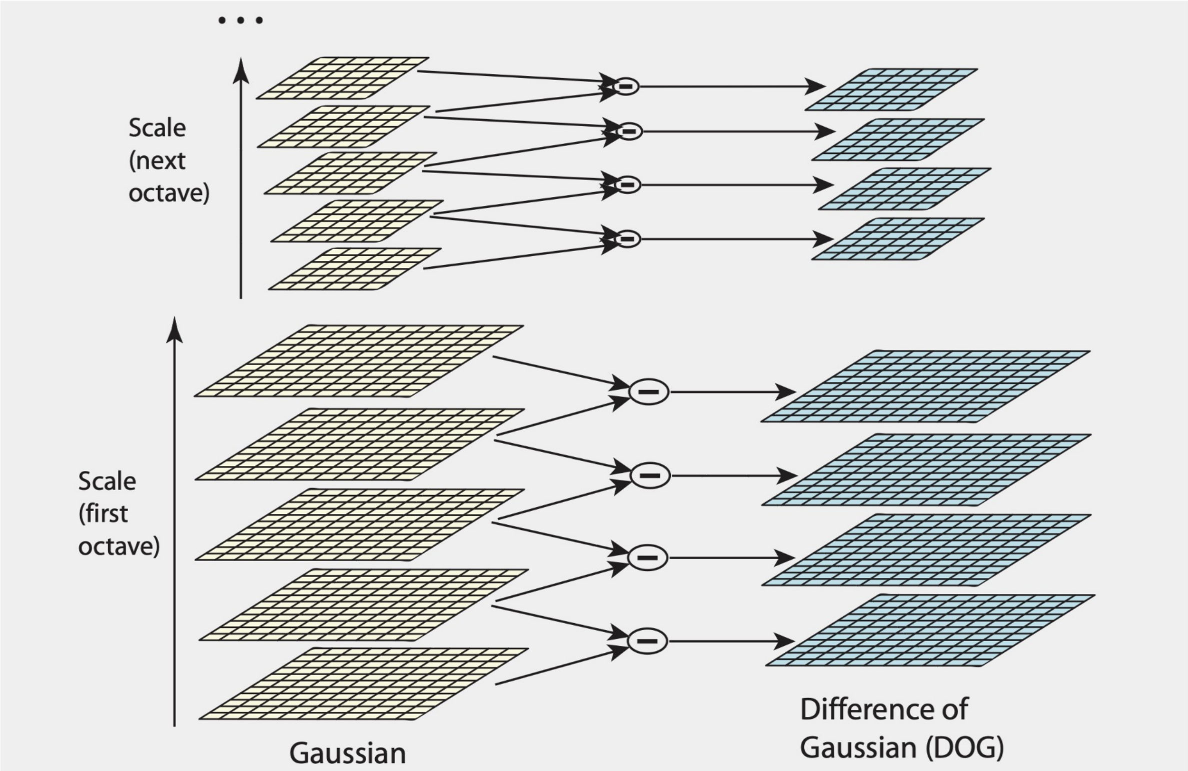


Figure 1: For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled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

Figure from Lowe (1999) and Lowe (2004)

Feature Engineering

$\vec{\theta}^T \vec{x}$

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:

① soil type

② asbestos

③ steel?

④ # bedrooms

⑤ # baths

⑥ sq ft. (livable)

⑦ grade

⑧ $(\text{sqft})^2$ ⑨ log(sqft)

⑩ exp(sqft)

NON-LINEAR FEATURES

Nonlinear Features

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

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

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

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

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

log $b_j(x) = \underline{\log(x)}$

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

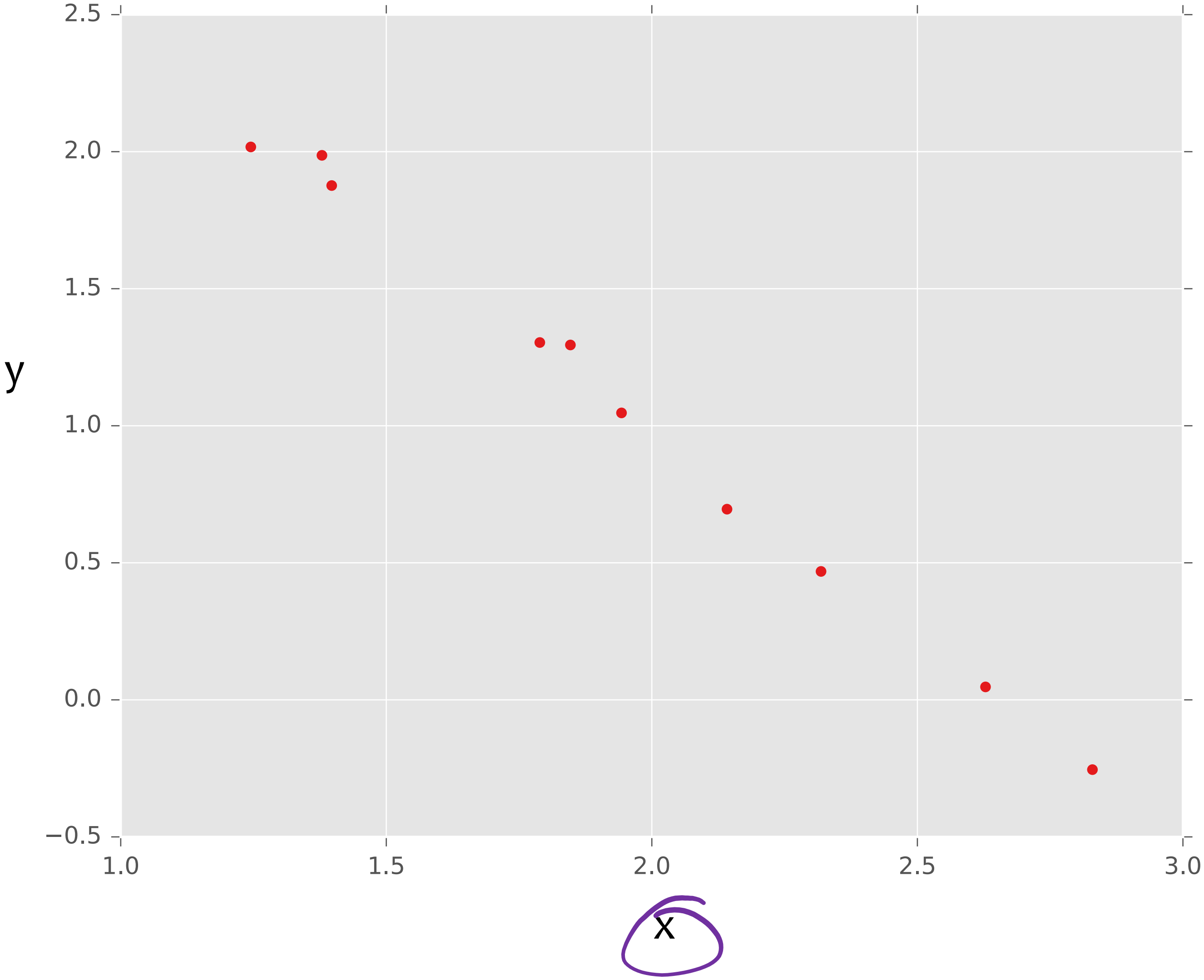
Examples:
- Perceptron
- Linear regression
- Logistic regression

Example: Linear Regression

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

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

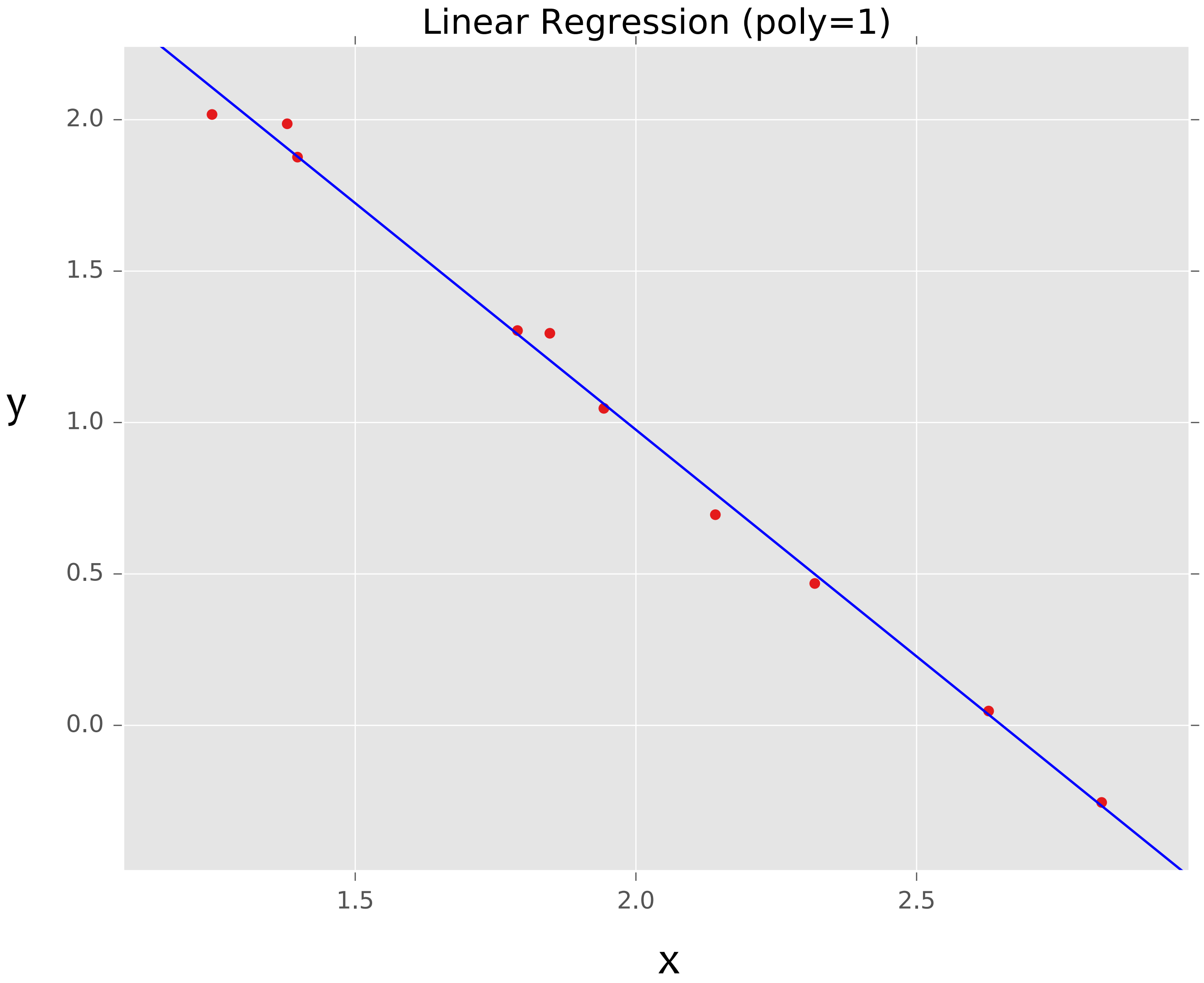


Example: Linear Regression

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

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

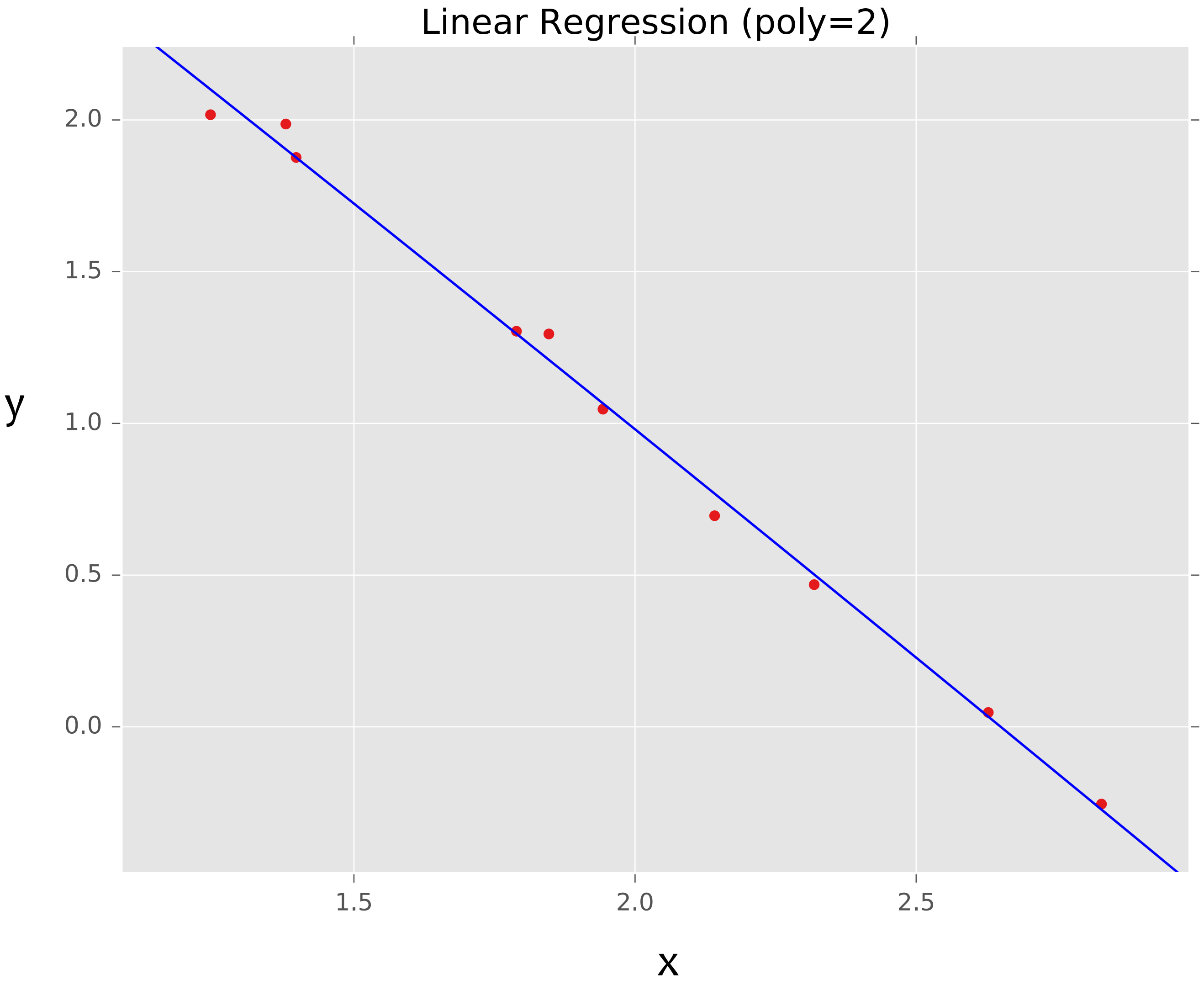
true "unknown" target function is linear with negative slope and gaussian noise



Example: Linear Regression

Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + b$ where f(.) 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	$(1.9)^2$



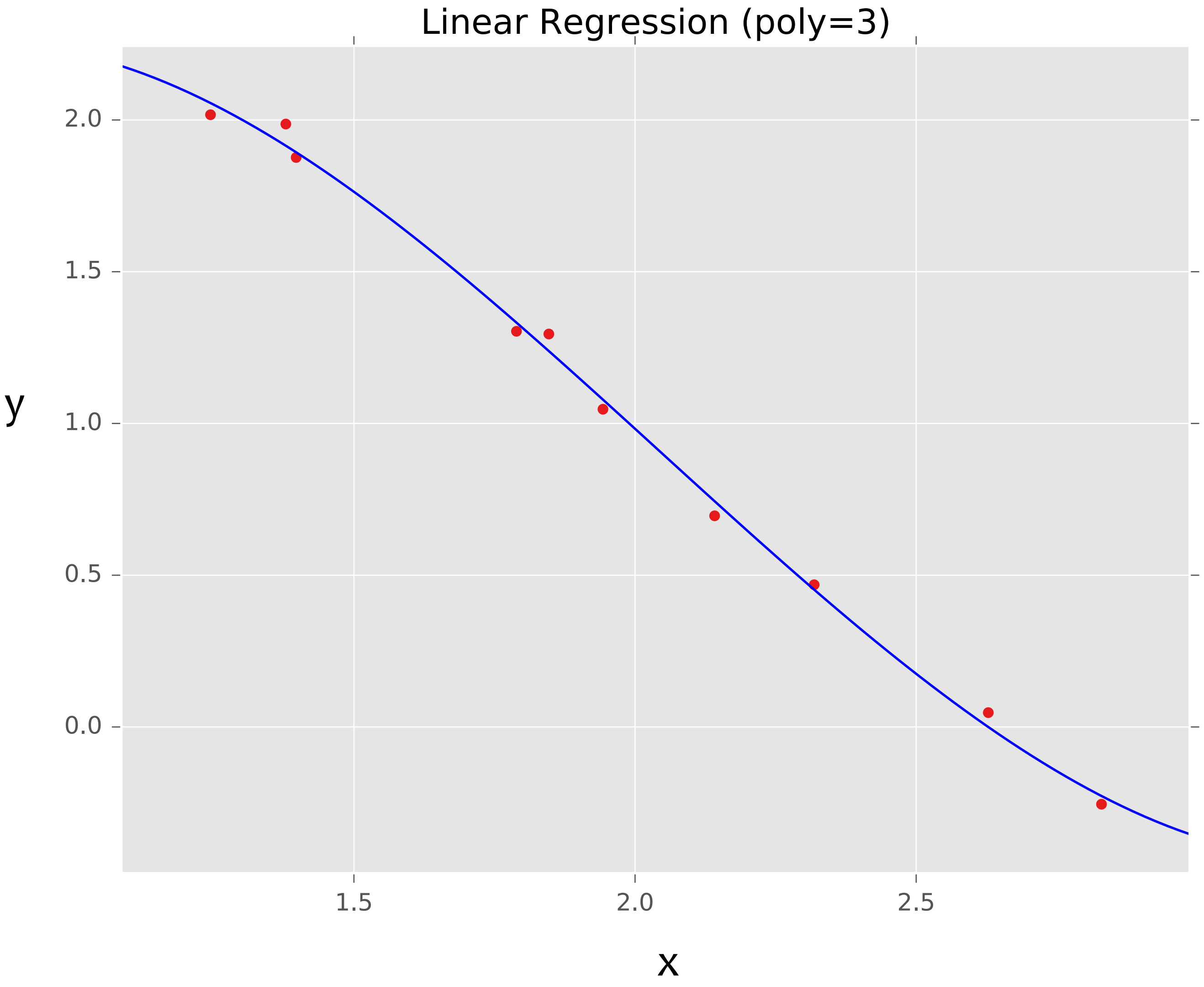
true "unknown" target function is linear with negative slope and gaussian noise

Example: Linear Regression

Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + b$
where f(.) 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	$(1.9)^2$	$(1.9)^3$

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

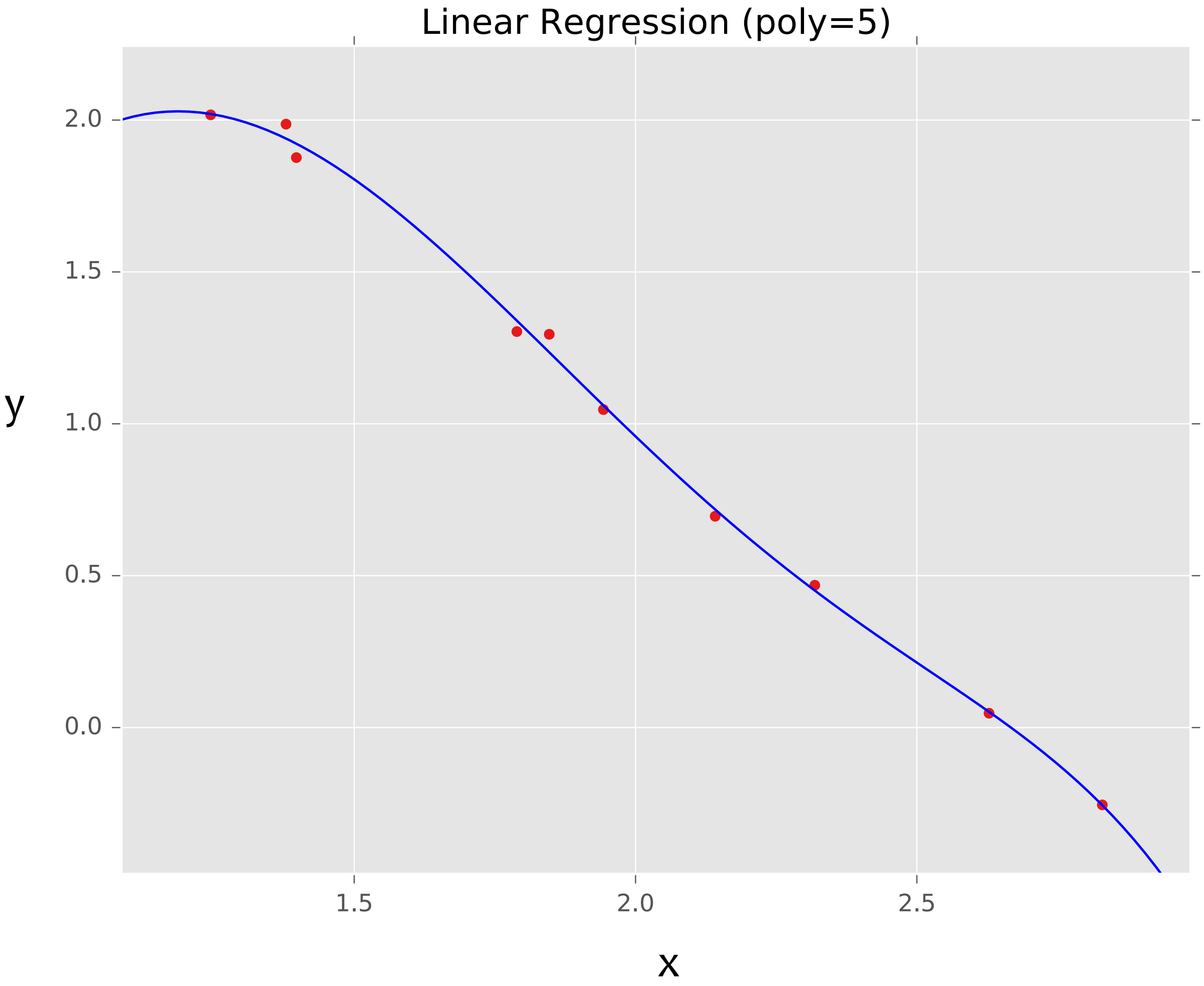


Example: Linear Regression

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

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	…	$(1.9)^5$

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

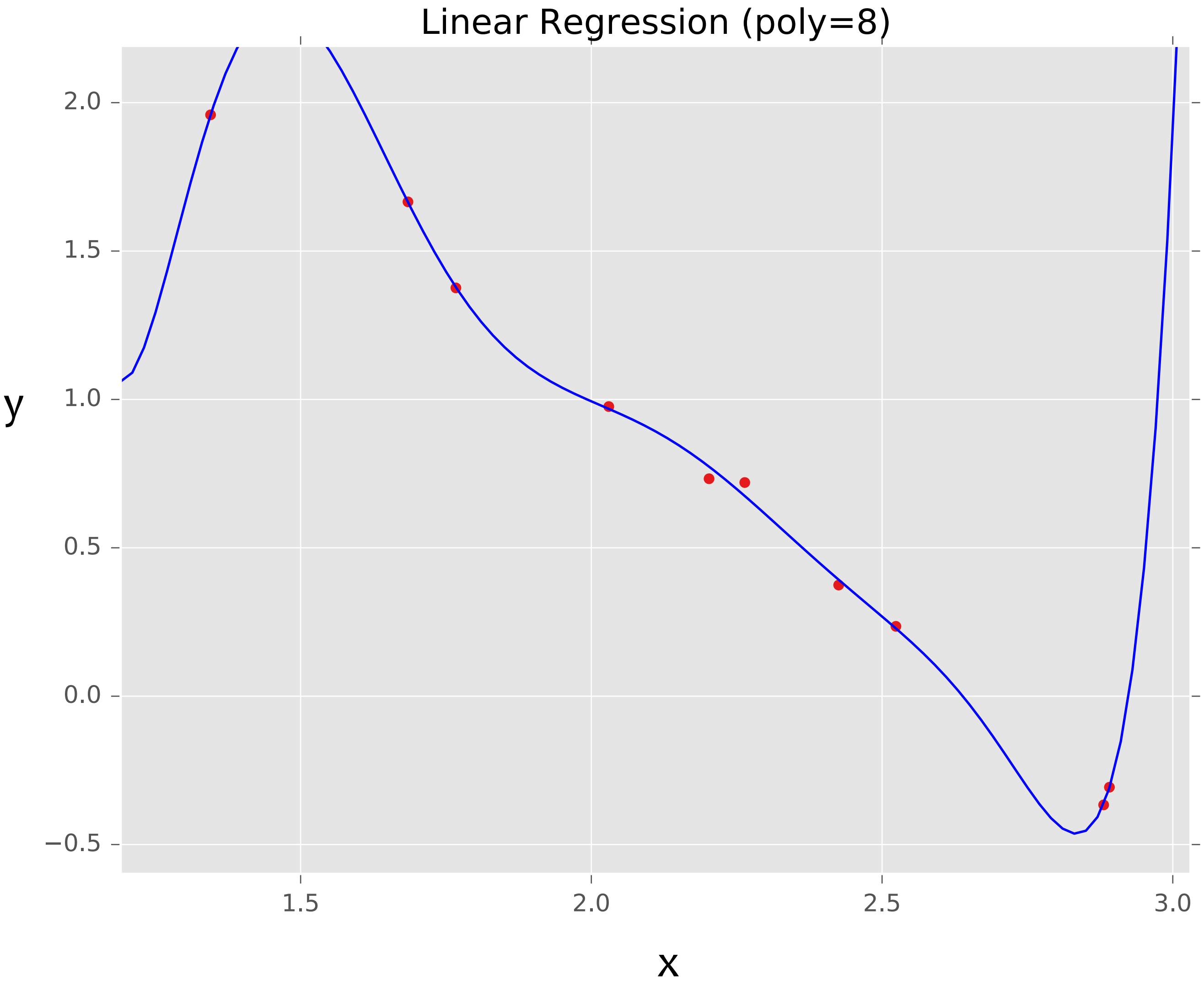


Example: Linear Regression

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

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	…	$(1.9)^8$

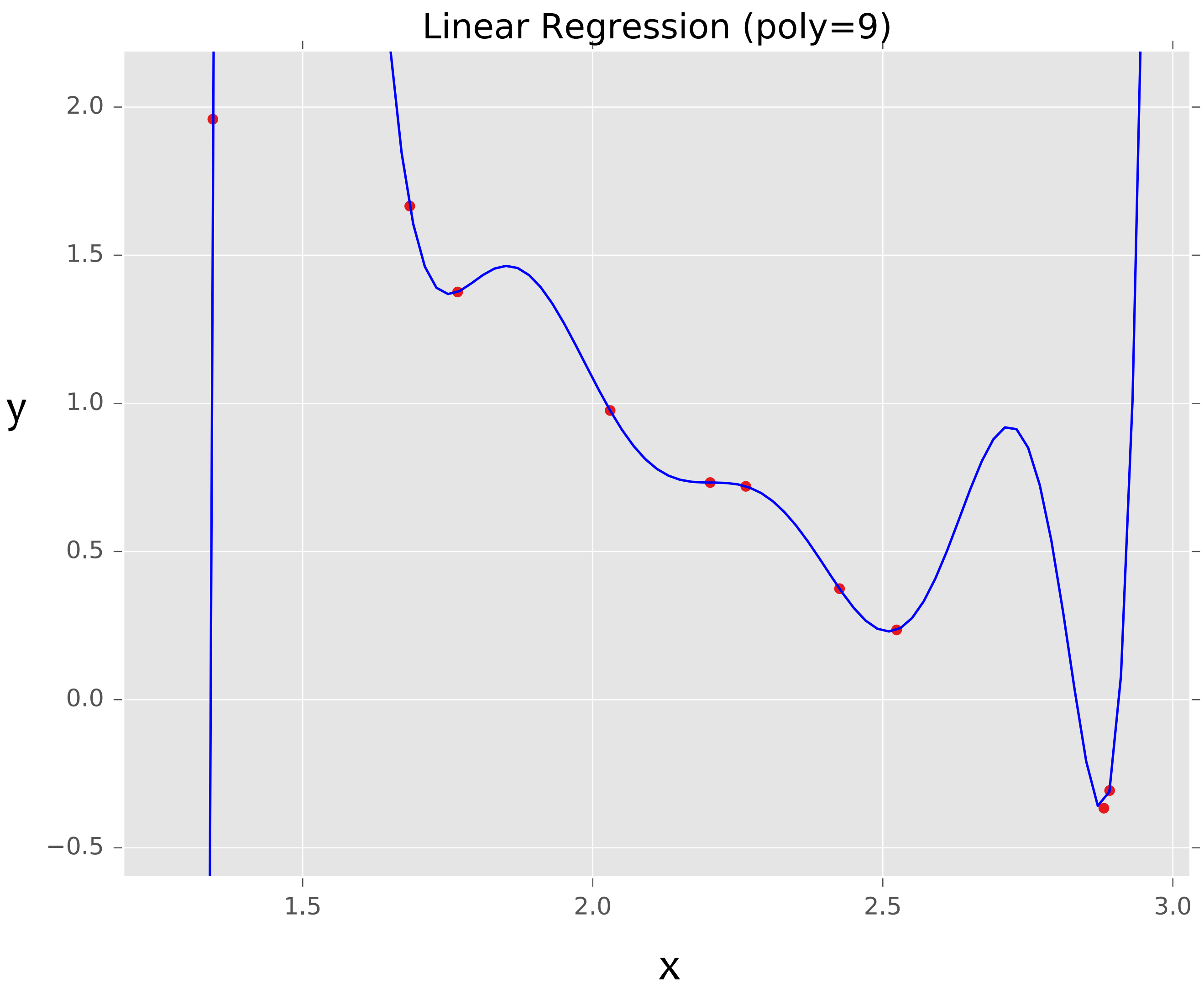
true "unknown" target function is linear with negative slope and gaussian noise



Example: Linear Regression

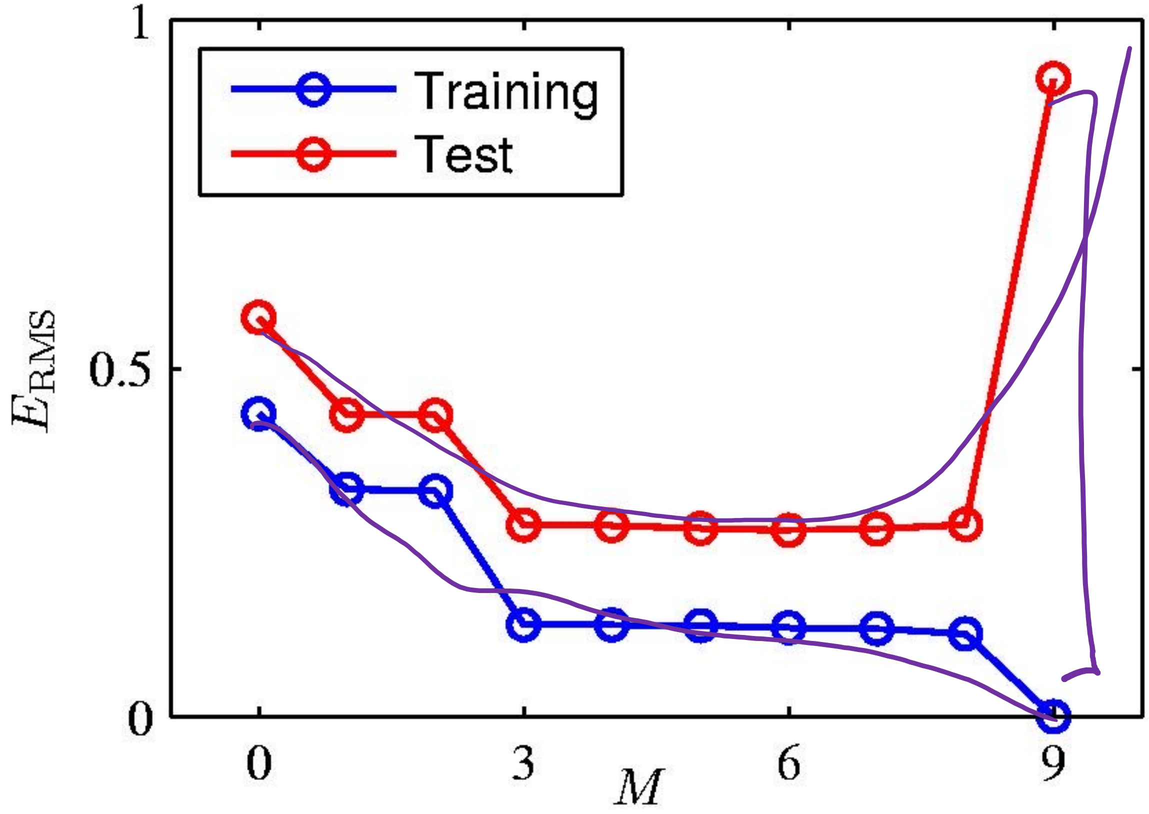
Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + b$ where f(.) is a polynomial basis function

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

Slide courtesy of William Cohen

Polynomial Coefficients

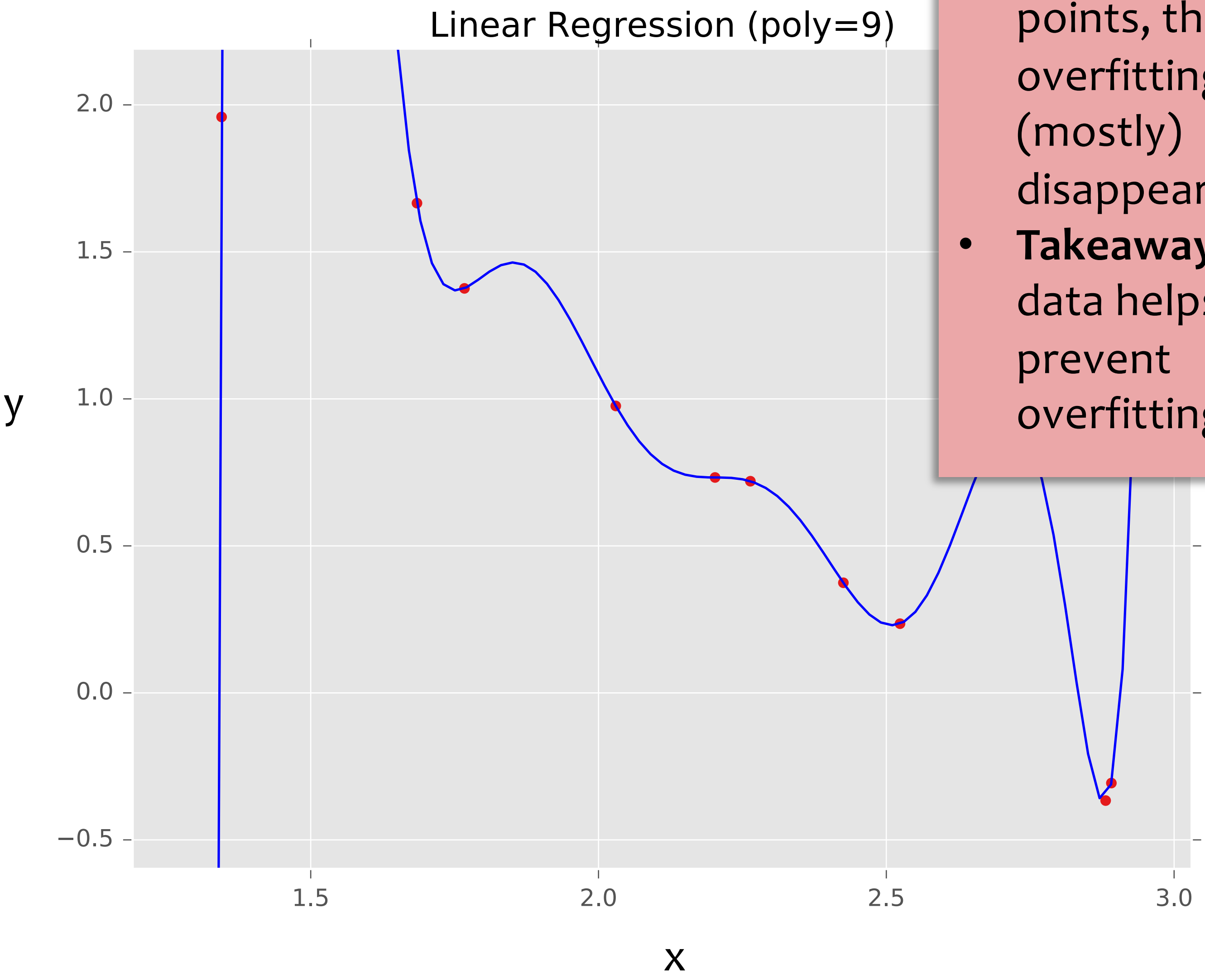
		b	w_1x+b	$w_1x+w_2x^2+w_3x^3+b$	$w_1x+\dots w_9x^9+b$
		$M=0$	$M=1$	$M=3$	$M=9$
b	θ_0	0.19	0.82	0.31	0.35
w_1	θ_1		-1.27	7.99	232.37
$\vdots$	θ_2			-25.43	-5321.83
	θ_3			17.37	48568.31
	θ_4				-231639.30
	θ_5				640042.26
	θ_6				-1061800.52
	θ_7				1042400.18
	θ_8				-557682.99
w_9	θ_9				125201.43

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

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	…	$(1.9)^9$

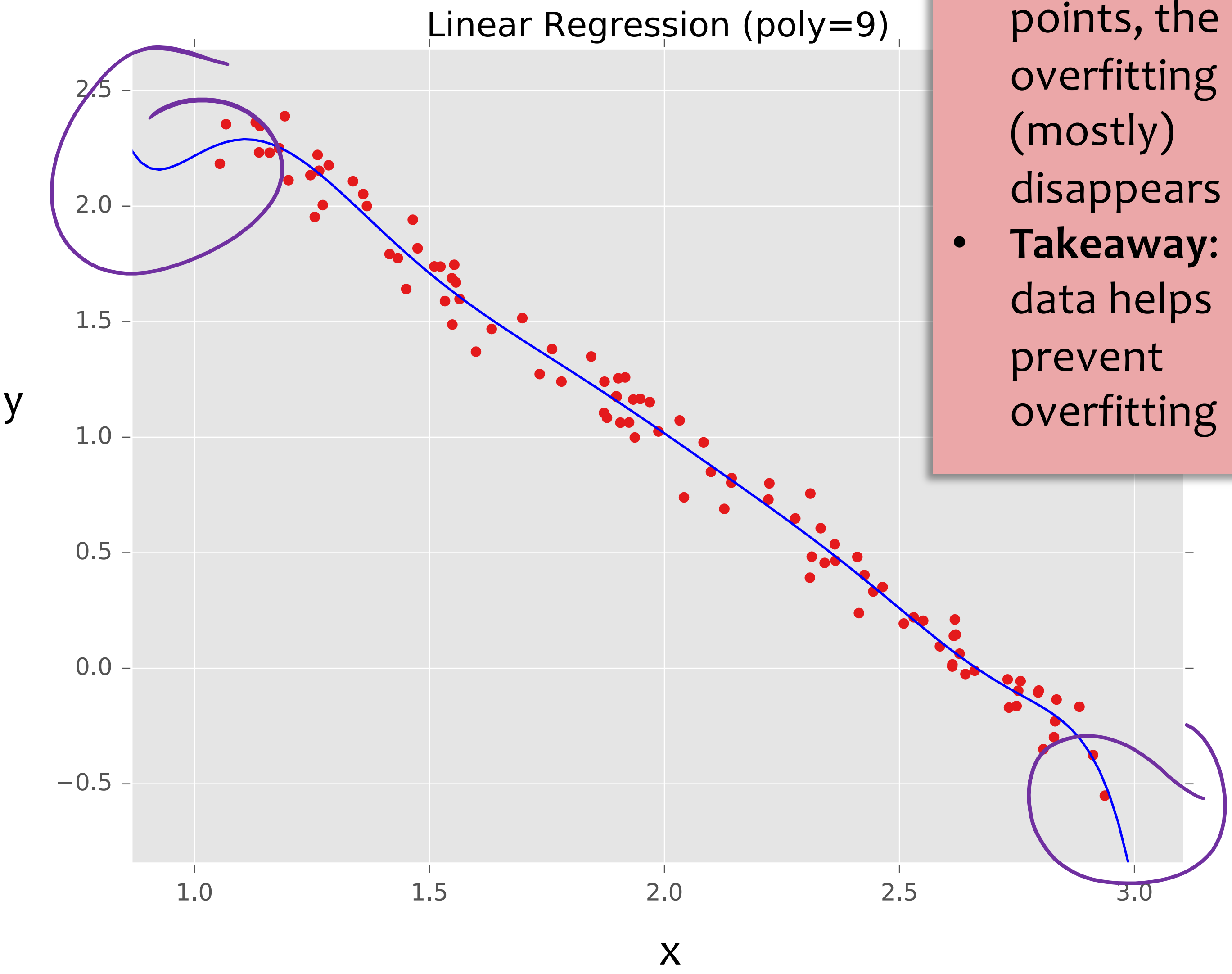


- With just N = 10 points we overfit!
- But with N = 100 points, the overfitting (mostly) disappears
- **Takeaway:** more data helps prevent overfitting

Example: Linear Regression

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

$\vec{\theta}' = \begin{bmatrix}\theta_2 \\ \theta_4\end{bmatrix}$ $\vec{x}' = \begin{bmatrix}x_2 \\ x_4\end{bmatrix}$ $\vec{\theta}'^T \vec{x}'$ simpler

- 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**)

$\vec{\theta} = \begin{bmatrix}\theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4\end{bmatrix} = \begin{bmatrix}0.001 \\ 103 \\ 0.0002 \\ -70\end{bmatrix}$ $\vec{x} = \begin{bmatrix}x_1 \\ x_2 \\ x_3 \\ x_4\end{bmatrix} \in [0,1]^4$ $\vec{\theta}^T \vec{x} \approx \vec{\theta}'^T \vec{x}'$

Regularization

fit the data

keep model model

- **Given** objective function: J(θ)
- **Goal** is to find: $\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \underbrace{J(\boldsymbol{\theta}) + \lambda r(\boldsymbol{\theta})}_{J'(\theta)}$

hyperparameter

- **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): $\|\boldsymbol{\theta}\|_q = \left(\sum_{m=1}^{M} |\theta_m|^q\right)^{\frac{1}{q}}$

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

Regularization Examples

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

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

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

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

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

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

Regularization Examples

Add an **L2 regularizer** to Logistic Regression

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

$$= \underbrace{\frac{1}{N}\sum_{i=1}^{N} -\log p(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta})}_{} + \boxed{\lambda \sum_{m=1}^{M} \theta_m^2}$$

Add an **L1 regularizer** to Logistic Regression

$$J'(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \boxed{\lambda||\boldsymbol{\theta}||_1}$$

$$= \underbrace{\frac{1}{N}\sum_{i=1}^{N} -\log p(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta})}_{} + \boxed{\lambda \sum_{m=1}^{M} |\theta_m|}$$

Regularization

Question:

Suppose we are minimizing J’(θ) where

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

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

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

B. …move towards the minimum of J(θ)

[✓ C.] …move towards the minimum of r(θ)

D. …move towards a theta vector of positive infinities

E. …move towards a theta vector of negative infinities

F. …stay the same