Machine Learning Department
School of Computer Science
Carnegie Mellon University

Logistic Regression + Feature Engineering + Regularization

Matt Gormley & Henry Chai

Lecture 10

Feb. 12, 2025

Reminders

- Practice Problems 1
  - released on course website
- Exam 1: Mon, Feb. 17
  - Time: 7:00 – 9:00pm
  - Location: Your room/seat assignment will be announced on Piazza
- Homework 4: Logistic Regression
  - Out: Mon, Feb 17
  - Due: Wed, Feb. 26 at 11:59pm

EXAM 1 LOGISTICS

Exam 1

- **Time / Location**
  - **Time:** **Mon, Feb 17, at 7:00pm - 9:00pm**
  - **Location & Seats:** You have all been split across multiple rooms. Everyone has an assigned seat in one of these room.
  - Please watch Piazza carefully for announcements.
- **Logistics**
  - Covered material: Lecture 1 – Lecture 7
  - Format of questions:
    - Multiple choice
    - True / False (with justification)
    - Derivations
    - Short answers
    - Interpreting figures
    - Implementing algorithms on paper
  - No electronic devices
  - You are allowed to **bring** one 8½ x 11 sheet of notes (front and back)

Exam 1

- **How to Prepare**
  - Attend the Exam OHs on Friday ?
  - Review exam practice problems
  - Review this year's homework problems
  - Consider whether you have achieved the "learning objectives" for each lecture / section
  - Write your one-page cheat sheet (back and front)

Exam 1

- **Advice (for during the exam)**
  - Solve the easy problems first
    (e.g. multiple choice before derivations)
    - if a problem seems extremely complicated you're likely missing something
  - Don't leave any answer blank!
  - If you make an assumption, write it down
  - If you look at a question and don't know the answer:
    - we probably haven't told you the answer
    - but we've told you enough to work it out
    - imagine arguing for some answer and see if you like it

Topics for Exam 1

- Foundations
  - Probability, Linear Algebra, Geometry, Calculus
  - Optimization
- Important Concepts
  - Overfitting
  - Experimental Design
- Classification
  - Decision Tree
  - KNN
  - Perceptron
- Regression
  - KNN Regression
  - Decision Tree Regression
  - Linear Regression

LOGISTIC REGRESSION

Logistic Regression

1. Model

$y \sim \text{Bernoulli}(\phi)$

$\phi = \sigma(\vec{\theta}^T \vec{x})$ where $\sigma(v) = \frac{1}{1+\exp(-v)}$

$$p(y|\vec{x}, \theta) = \begin{cases} \sigma(\vec{\theta}^T \vec{x}) & \text{if } y=1 \\ 1-\sigma(\vec{\theta}^T \vec{x}) & \text{if } y=0 \end{cases}$$

2. Objective

$$\ell(\theta) = \log p(D|\vec{\theta}) = \log \prod_{i=1}^{N} p(y^{(i)}|\vec{x}^{(i)}, \vec{\theta})$$
$$= \sum_{i=1}^{N} \log p(y^{(i)}|\vec{x}^{(i)}, \vec{\theta})$$

$$J(\theta) = -\frac{1}{N}\ell(\vec{\theta})$$
$$= \frac{1}{N}\sum_{i=1}^{N} \underbrace{-\log p(y^{(i)}|\vec{x}^{(i)}, \vec{\theta})}_{J^{(i)}(\vec{\theta})}$$

Logistic Regression

3A. Derivatives

$$\frac{\partial J^{(i)}(\vec{\theta})}{\partial \theta_m} = \frac{\partial}{\partial \theta_m}\left(-\log p(y^{(i)} \mid \vec{x}^{(i)}, \vec{\theta})\right)$$

$$= \begin{cases} \frac{\partial}{\partial \theta_m}\left(-\log \sigma(\theta^T \vec{x}^{(i)})\right) & \text{if } y^{(i)} = 1 \\ \frac{\partial}{\partial \theta_m}\left(-\log\left(1 - \sigma(\theta^T \vec{x}^{(i)})\right)\right) & \text{if } y^{(i)} = 0 \end{cases}$$

$= \cdots$

$= \cdots$ } HW

$= \cdots$

$$= -\left(y^{(i)} - \sigma\left(\vec{\theta}^T \vec{x}^{(i)}\right)\right) x_m^{(i)}$$

scalar · scalar

3B. Gradients

$$\nabla J^{(i)}(\theta) = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} = -\left(y^{(i)} - \sigma\left(\vec{\theta}^T \vec{x}^{(i)}\right)\right) \vec{x}^{(i)}$$

$$\nabla J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \nabla J^{(i)}(\vec{\theta})$$

Logistic Regression

4. Optimization

Find $\vec{\theta}$ by GD
or by SGD

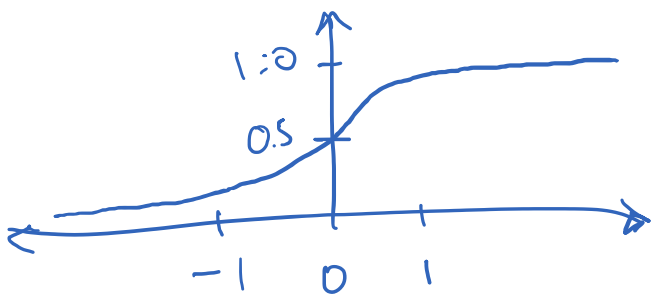
5. Prediction

Predict $\hat{y} = \underset{y \in \{0,1\}}{\operatorname{argmax}} \; p(y|\vec{x})$

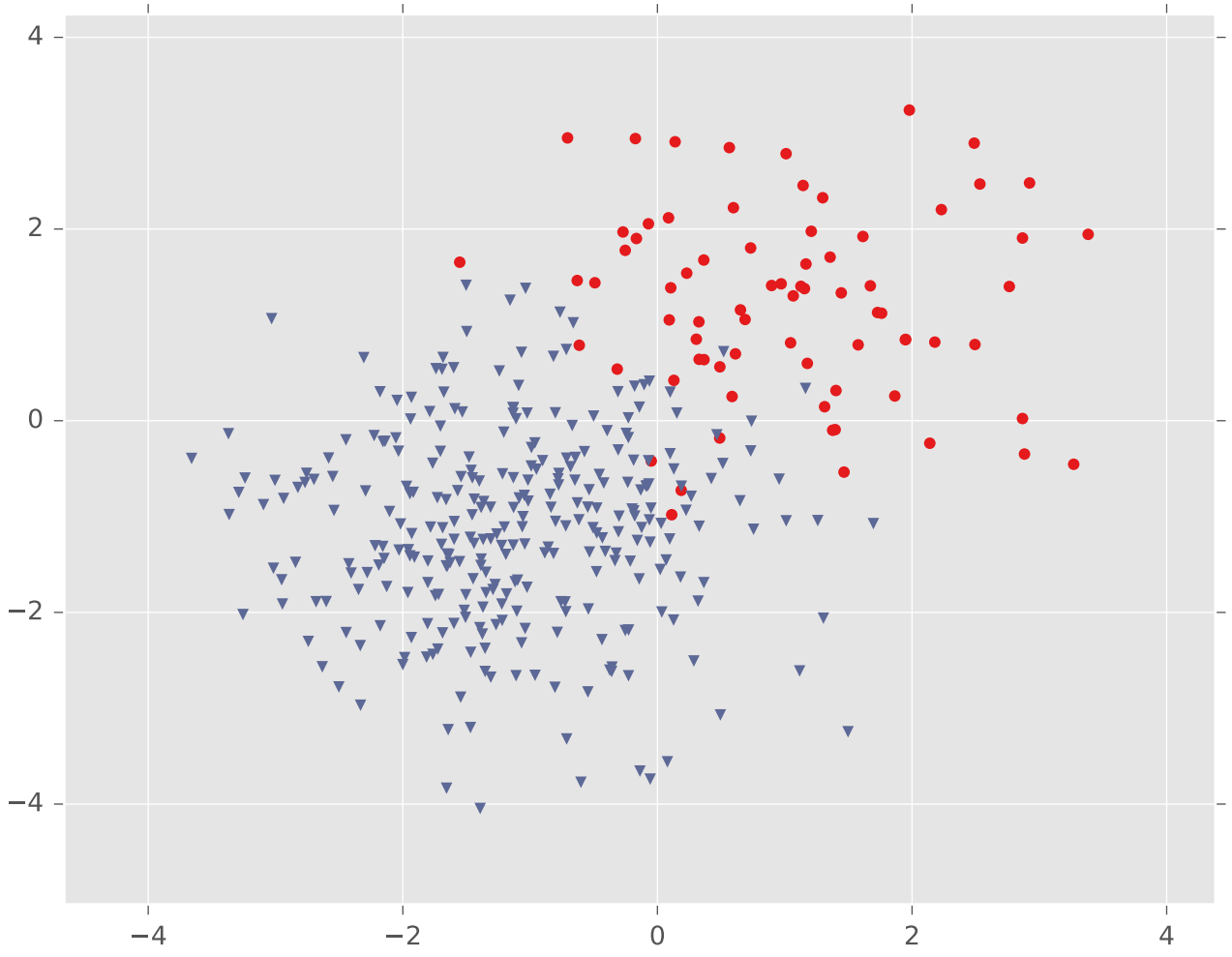
$= \cdots$

$= \cdots$

$= \text{"sign"}(\vec{\theta}^T \vec{x})$



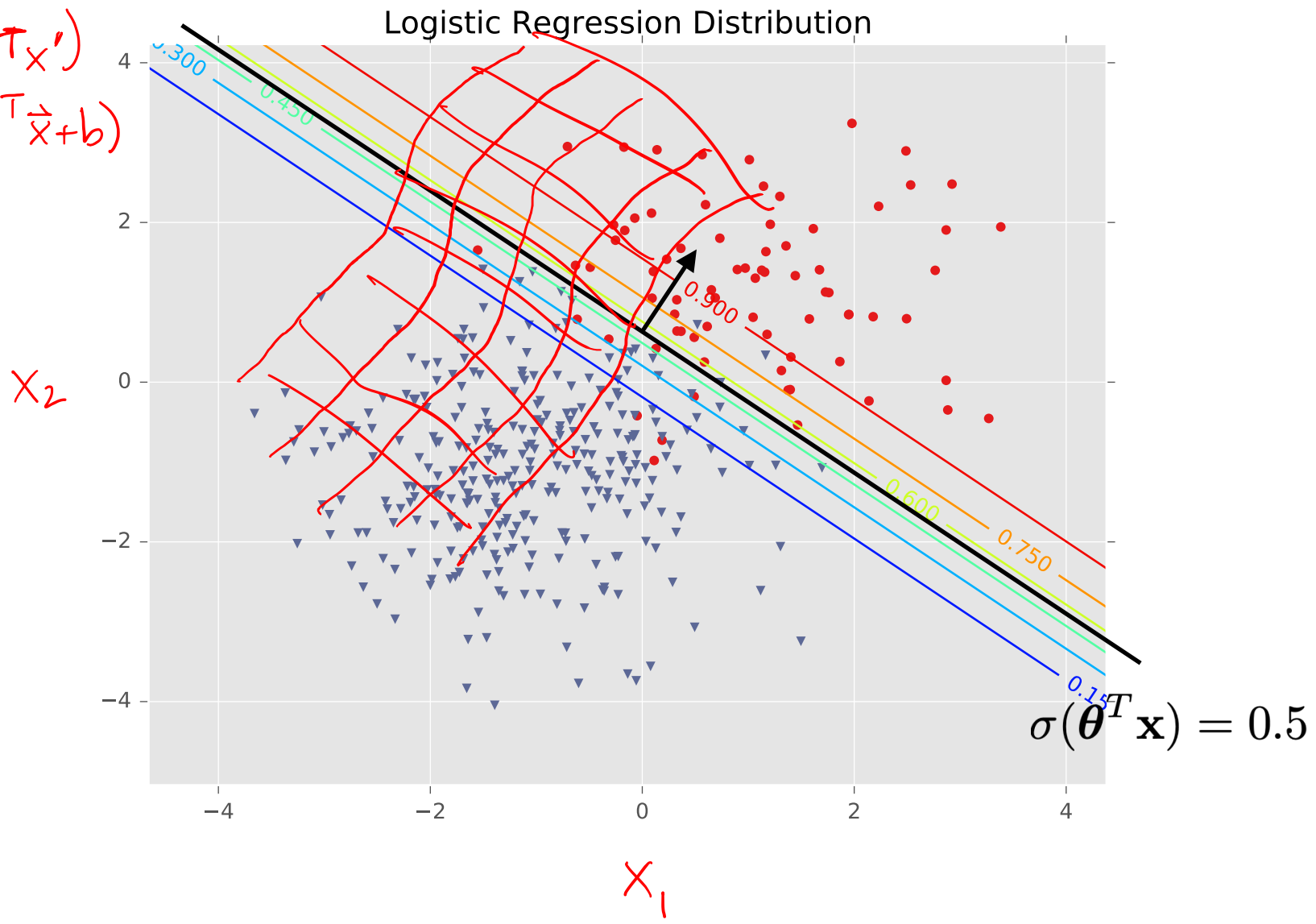
Logistic Regression



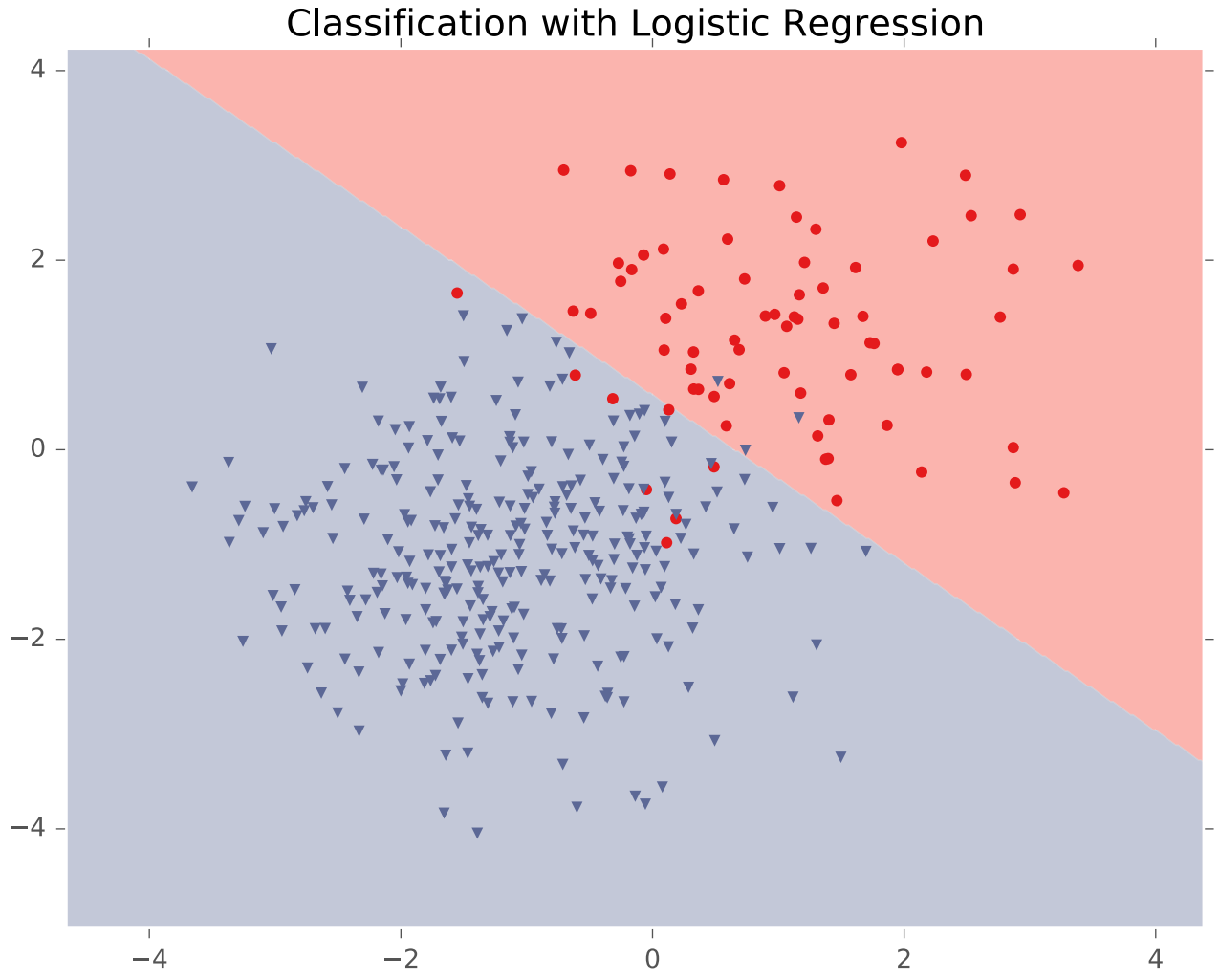
Logistic Regression

$p(y=1 \mid \vec{x}, \theta) = \sigma(\vec{\theta}^T x')$

$= \sigma(\vec{w}^T \vec{x} + b)$

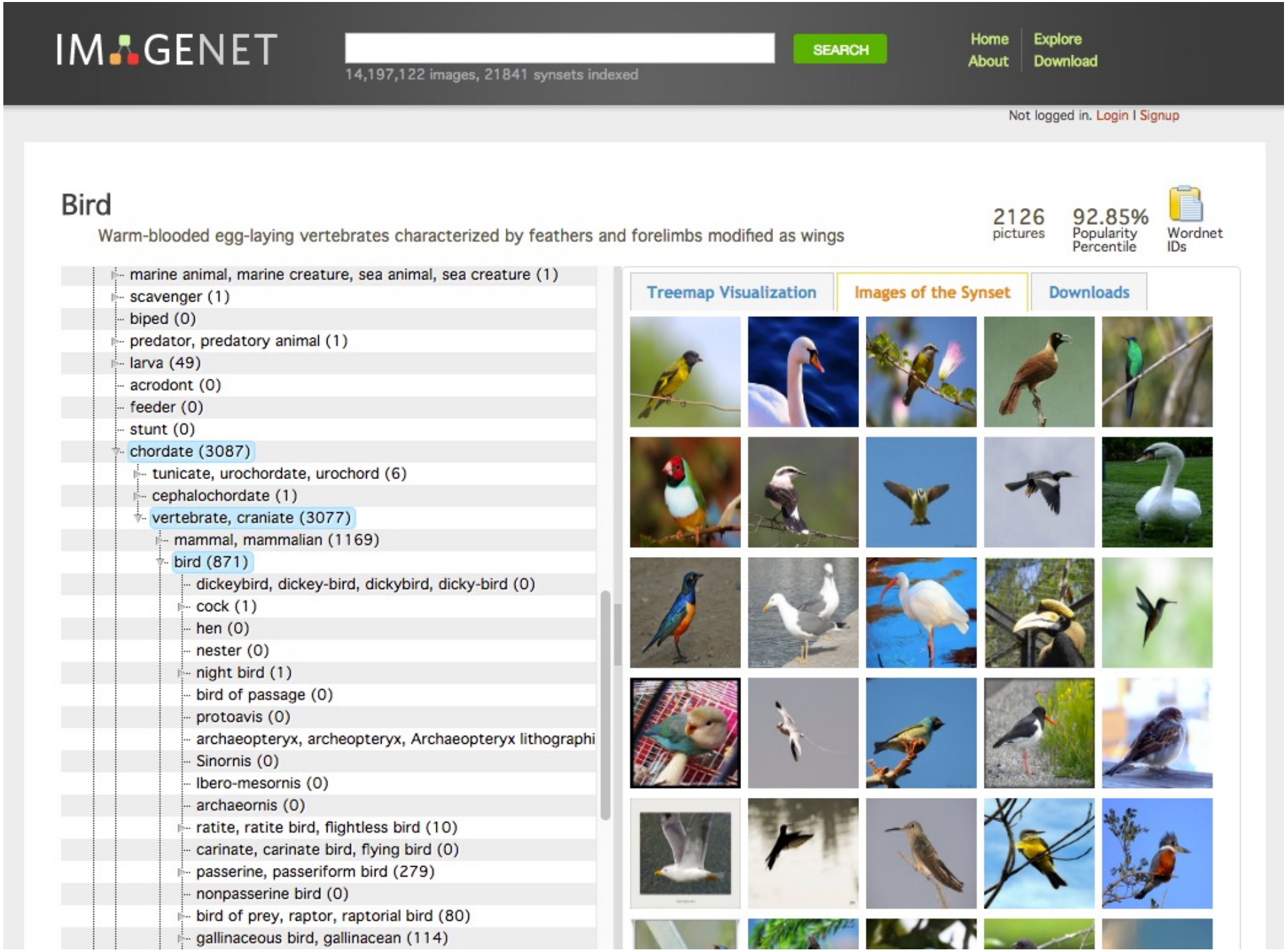


Logistic Regression



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 I 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 | Explore
About | Download

Not logged in. Login I Signup

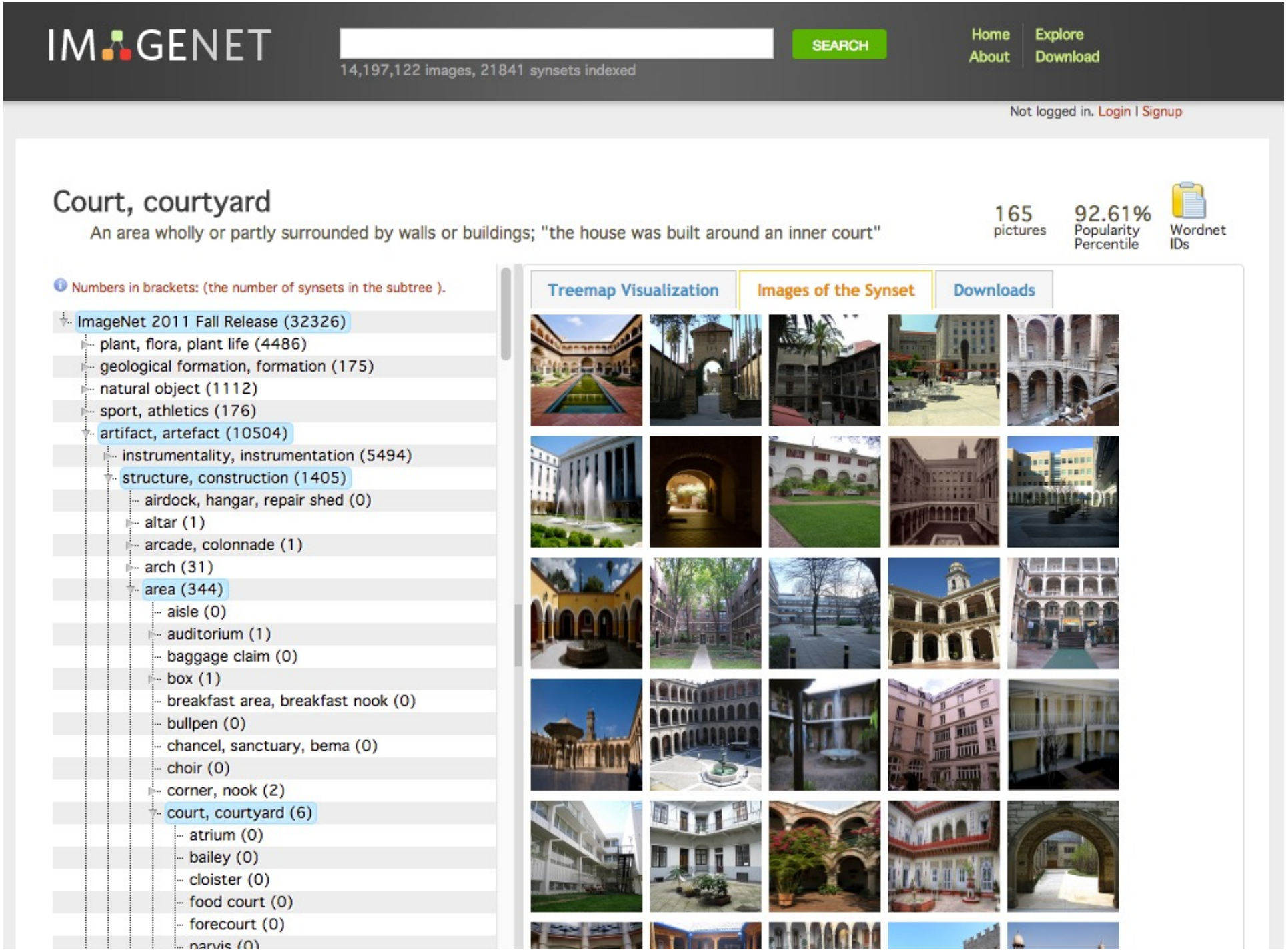
German iris, Iris kochii

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
Explore
About
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)
Treemap Visualization
Images of the Synset
Downloads

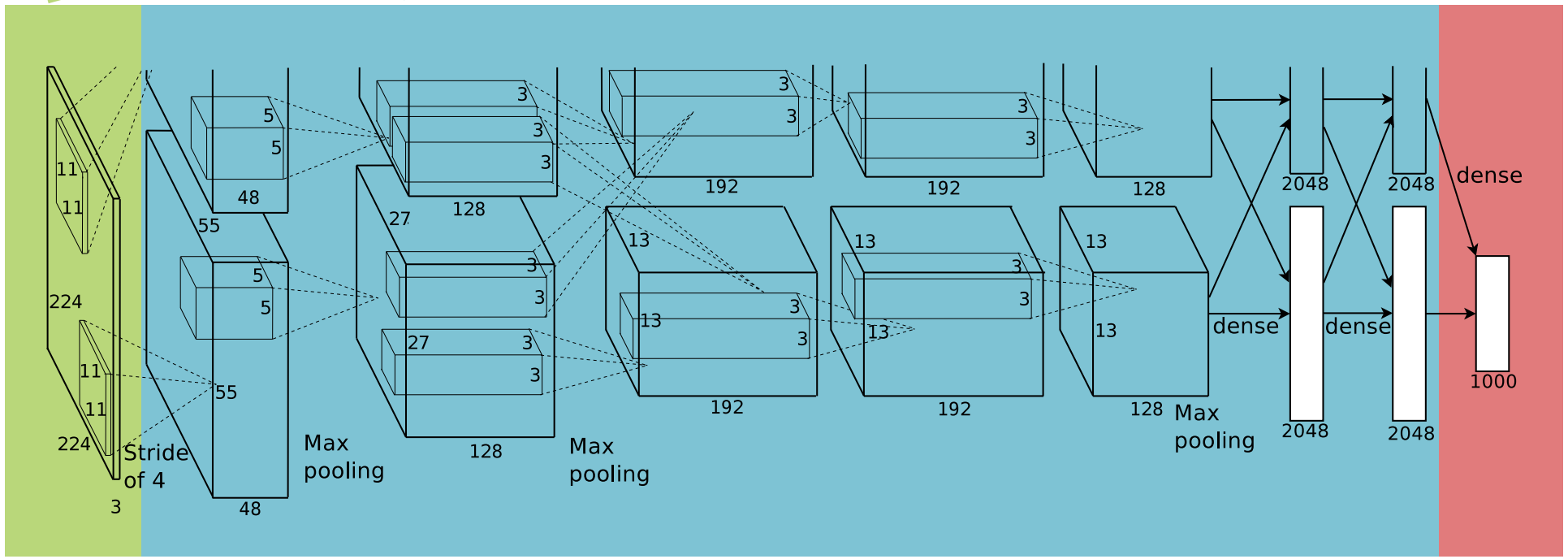
Example: Image Classification

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



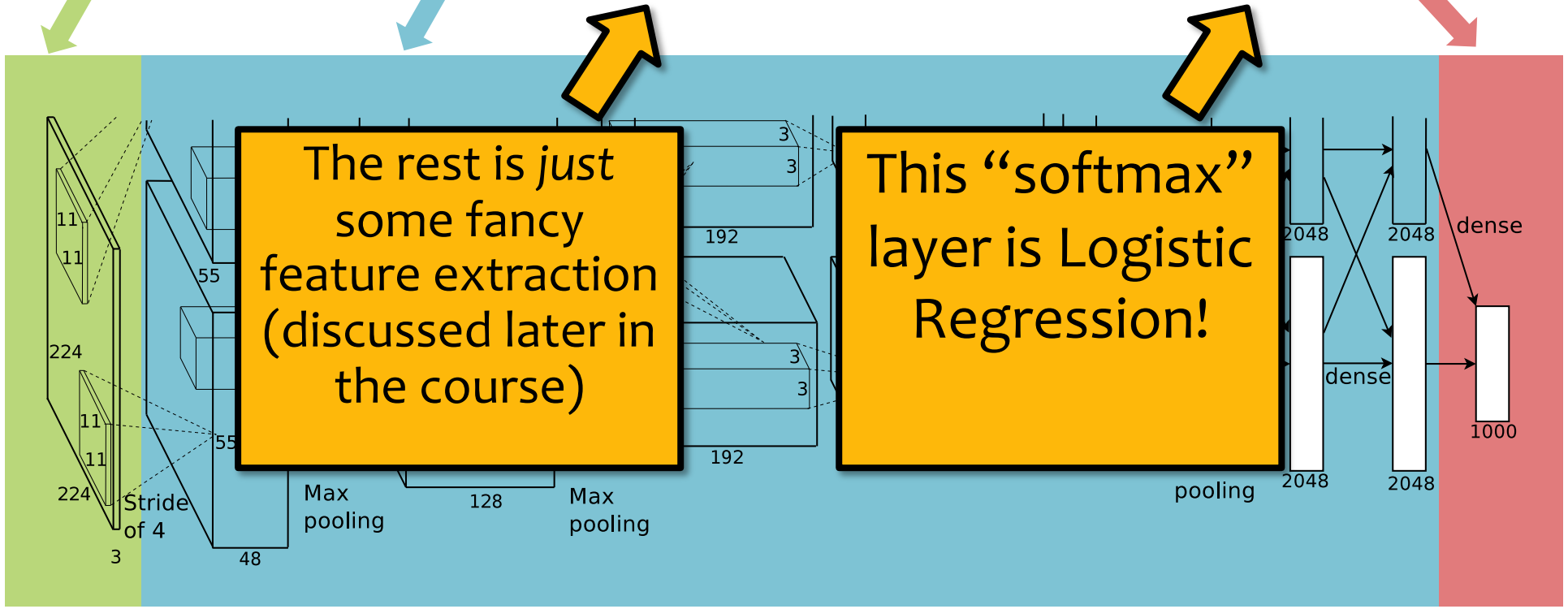
Example: Image Classification

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



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

Linear Models

PERCEPTRON, LINEAR REGRESSION, AND LOGISTIC REGRESSION

Matching Game

Poll Question:

Match the Algorithm to its Update Rule

1. SGD for Logistic Regression $h_{\boldsymbol{\theta}}(\mathbf{x}) = p(y = 1 \mid \mathbf{x})$	**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.** $\theta_k \leftarrow \theta_k + \dfrac{1}{1 + \exp \lambda(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)})}$
3. Perceptron $h_{\boldsymbol{\theta}}(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$	**6.** $\theta_k \leftarrow \theta_k + \lambda(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)}) x_k^{(i)}$

Answer:

18%	A. 1=5, 2=4, 3=6	7%	E. 1=6, 2=6, 3=6
67%	B. 1=5, 2=6, 3=4	0%	F. 1=6, 2=5, 3=5
0%	C. 1=6, 2=4, 3=4	0%	G. 1=5, 2=5, 3=5
5%	D. 1=5, 2=6, 3=6	5%	H. 1=4, 2=5, 3=6

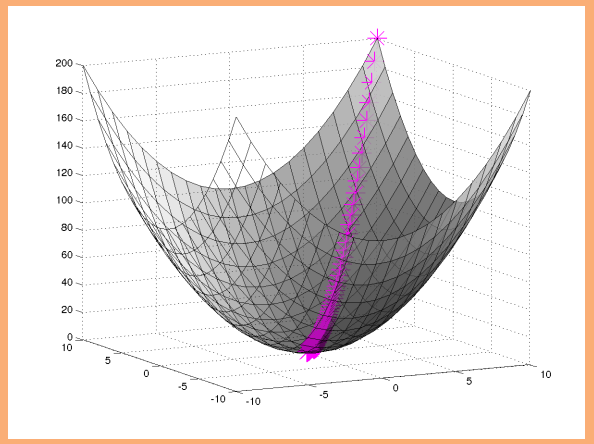
I. None of the above

Gradient Descent

Recall…

Algorithm 1 Gradient Descent

1: **procedure** 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).

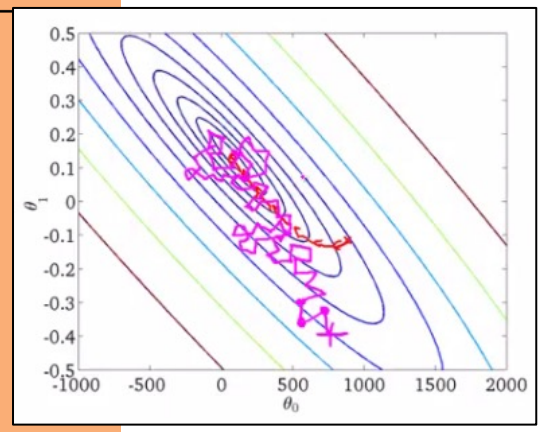
$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_M} J(\boldsymbol{\theta}) \end{bmatrix}$$

Stochastic Gradient Descent (SGD)

Recall…

Algorithm 1 Stochastic Gradient Descent (SGD)

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

A=toxic

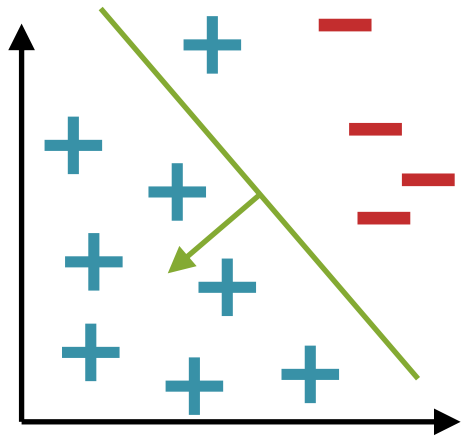
B

C

Poll 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

Functio

Previous
was gen
functior

Our goa
best app

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*!

Probabilistic Learning

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

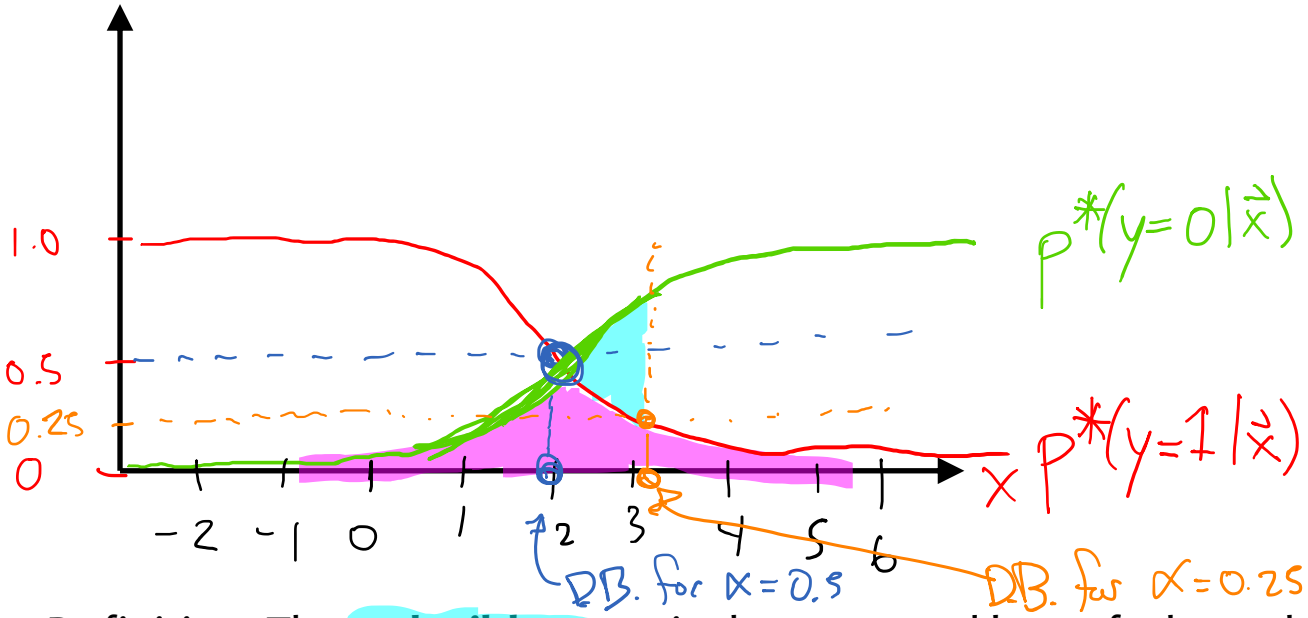
$$\mathbf{x}^{(i)} \sim p^*(\cdot)$$

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

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.



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

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

$\alpha = 0.5$ for 0/1 loss

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

choose $\alpha = 0.001$

Definition: The **reducible error** is the expected loss of a hypothesis h(x) that could be reduced if we knew p*(y|x) and picked 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 we knew p*(y|x) and picked 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 S randomly chosen examples

Mini-Batch SGD

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

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

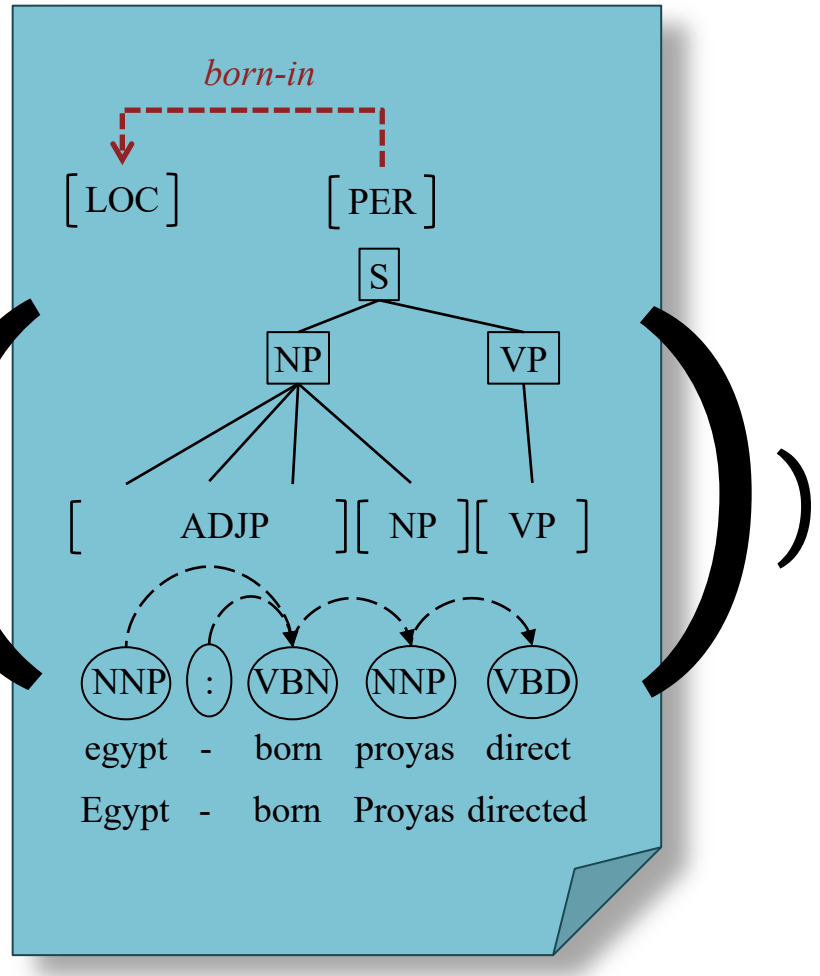
Mini-batch SGD: $\mathbf{g} = \frac{1}{S} \sum_{s=1}^{S} \nabla J^{(i_s)}(\boldsymbol{\theta})$ where i_s sampled uniformly $\forall s$

$S = 16, 32, 64, \ldots$

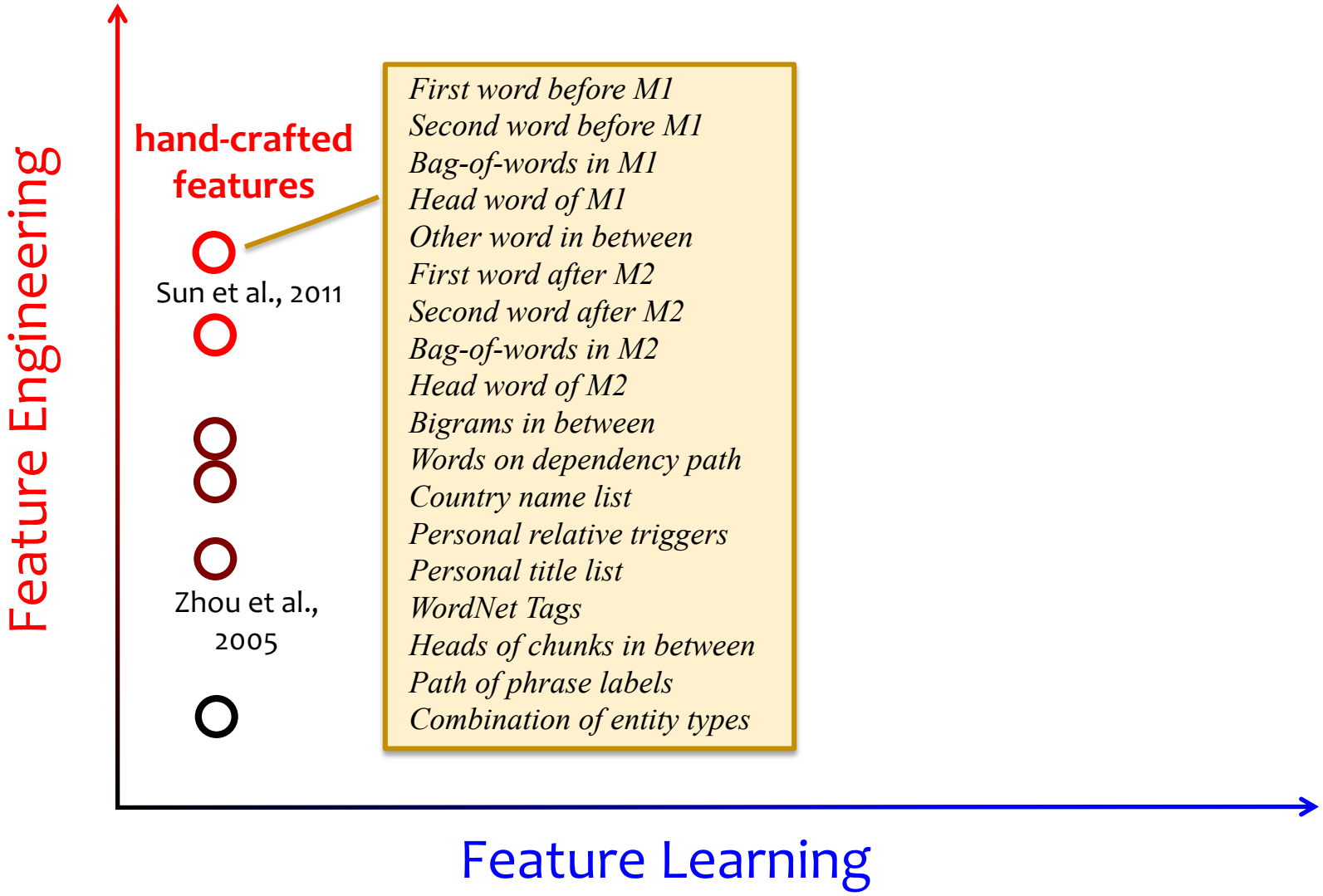
FEATURE ENGINEERING

Handcrafted Features

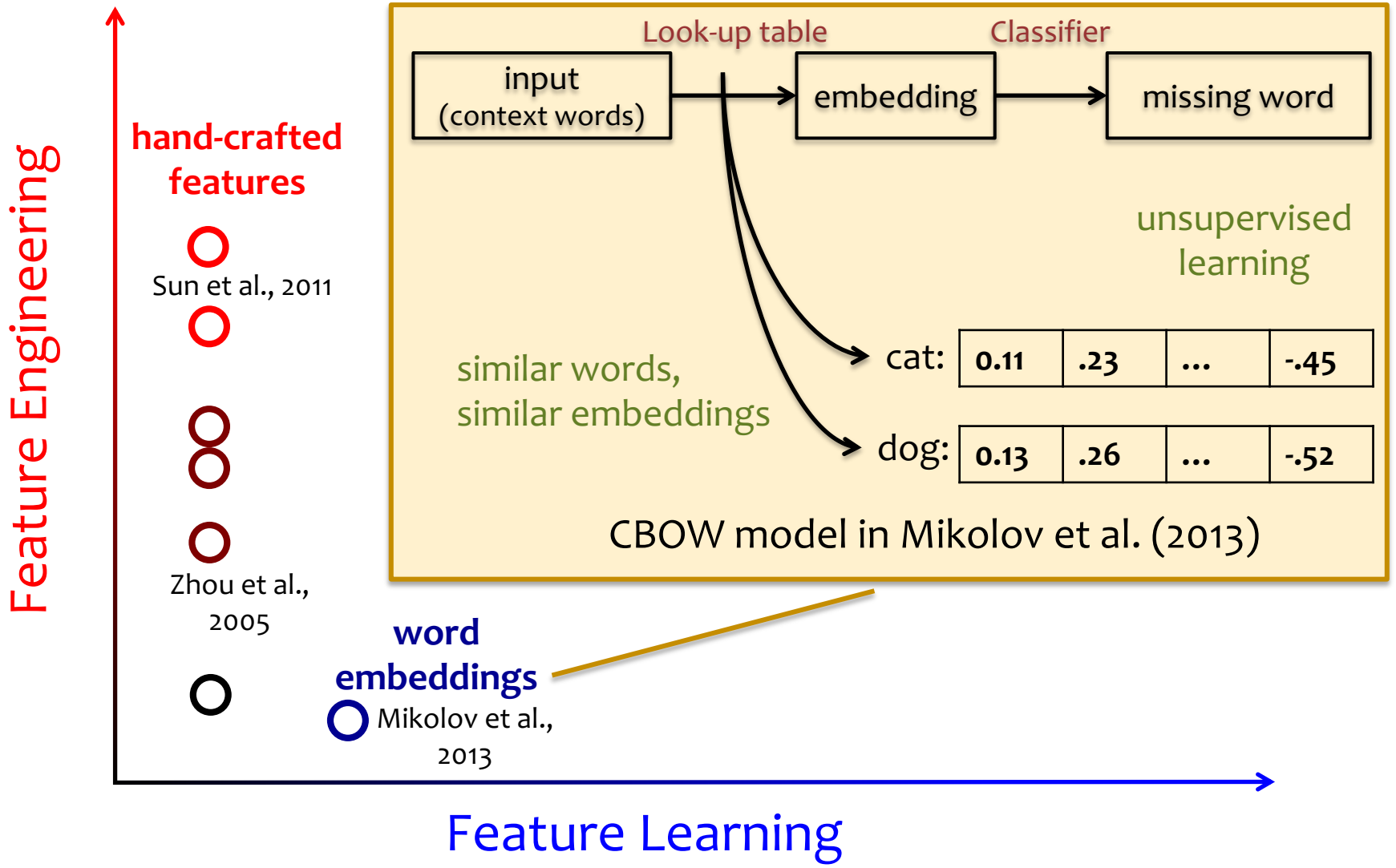
$$p(y|x) \propto \exp(\Theta_y \bullet f(\quad))$$



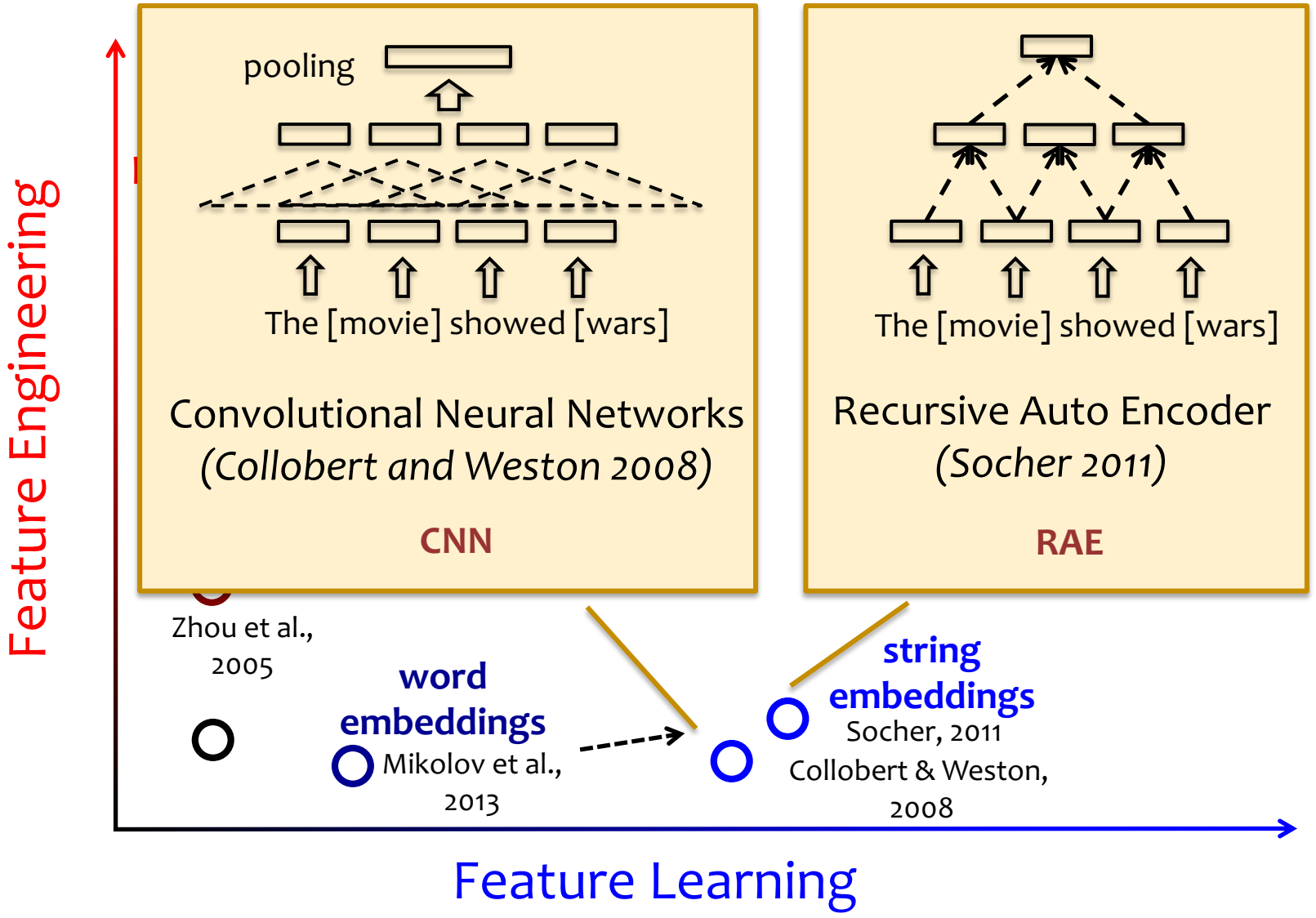
Where do features come from?



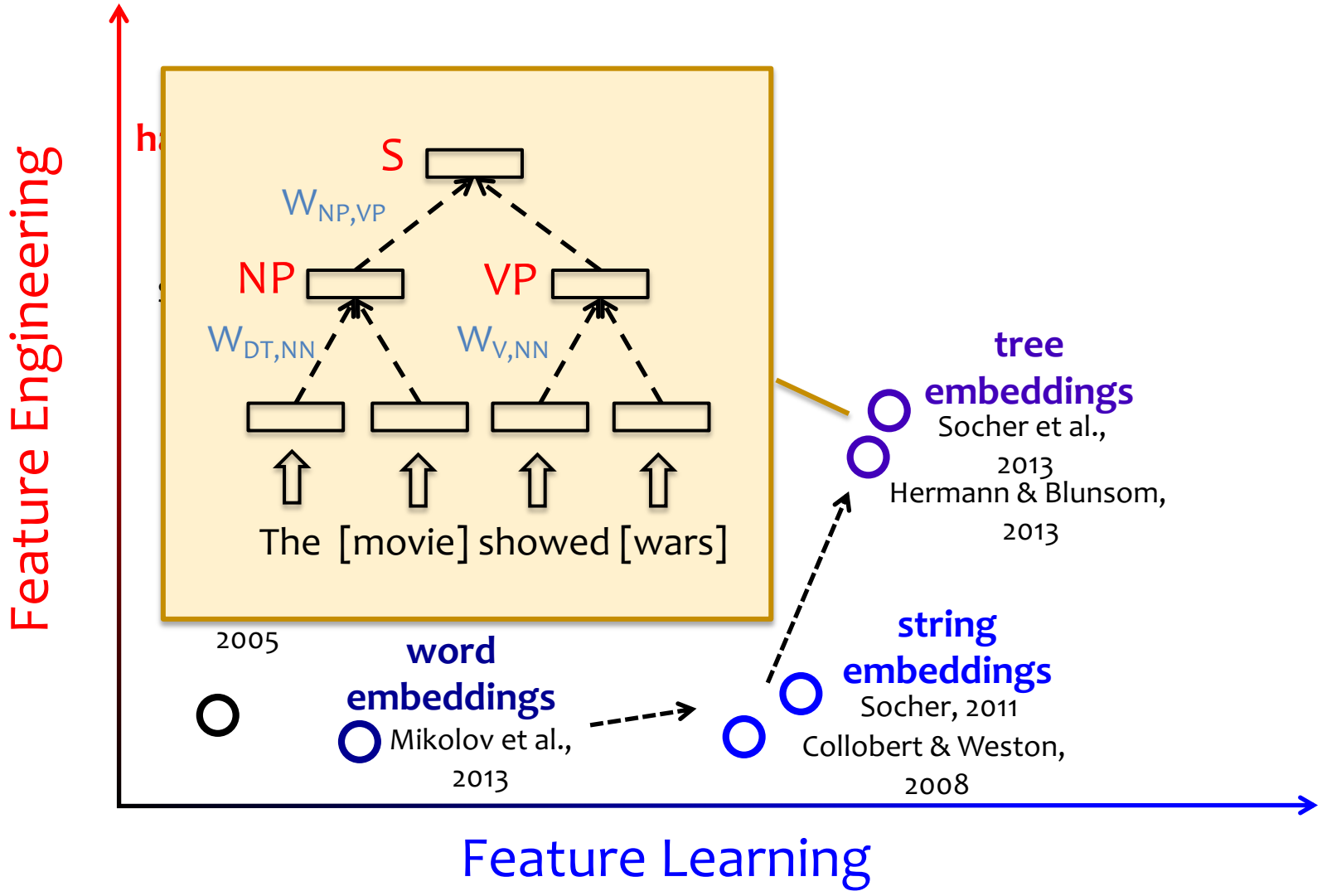
Where do features come from?



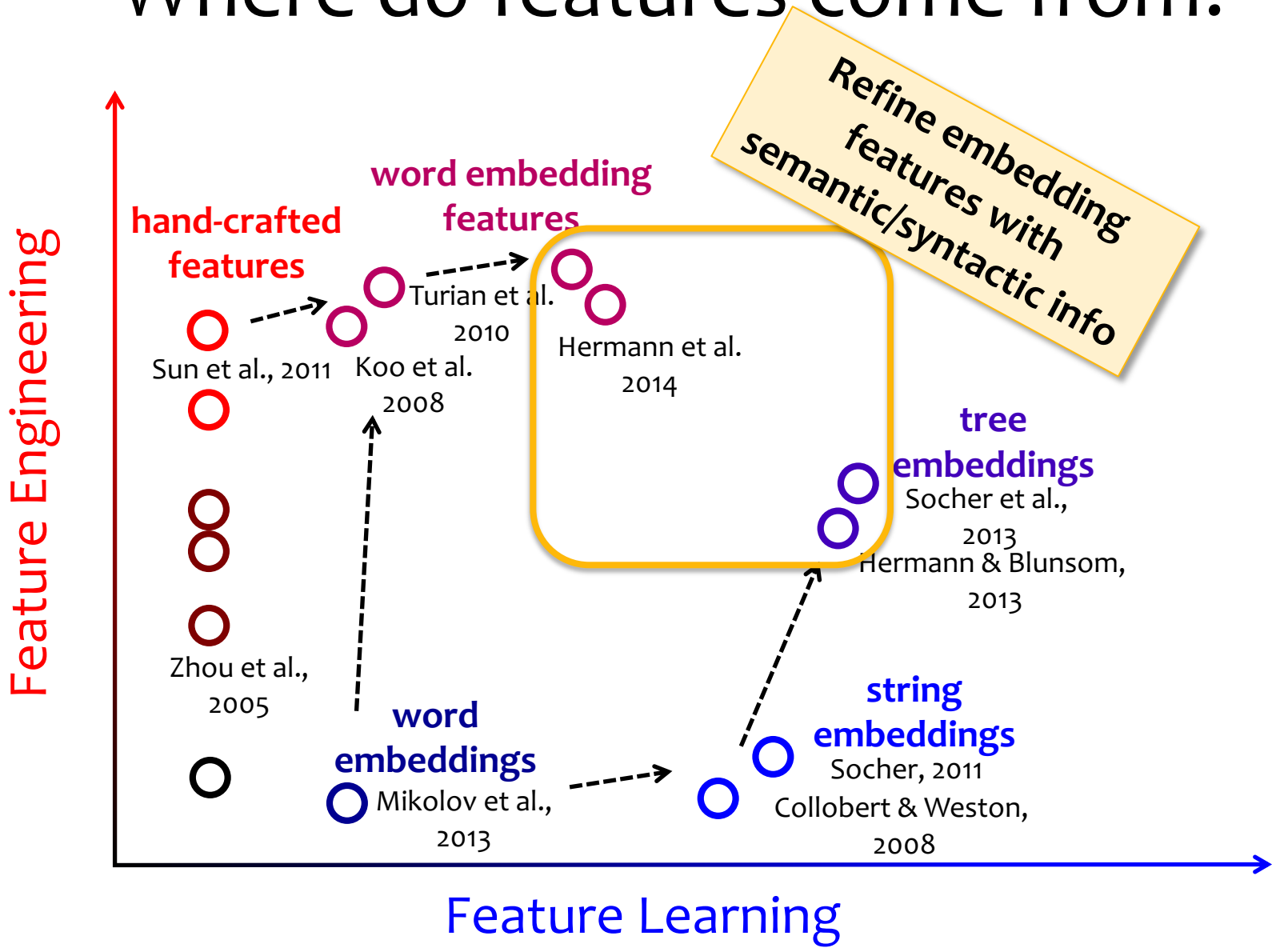
Where do features come from?



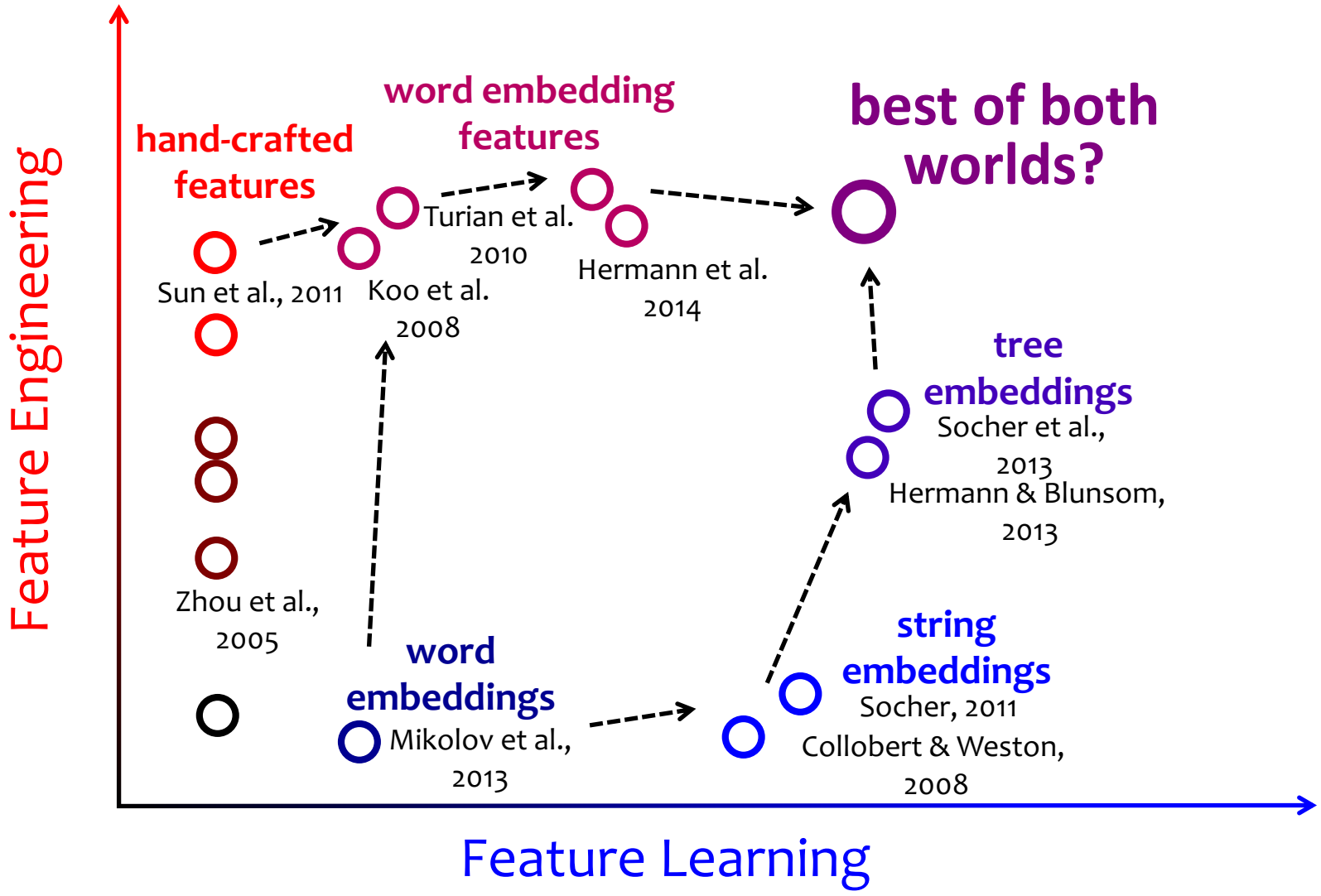
Where do features come from?



Where do features come from?



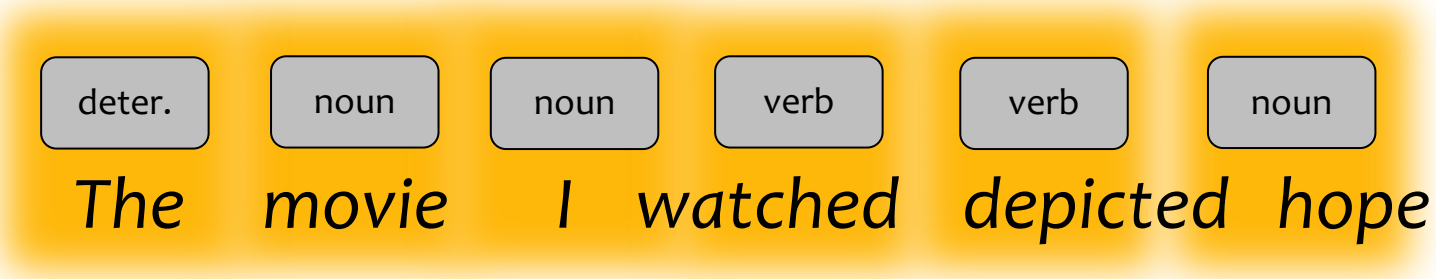
Where do features come from?



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.

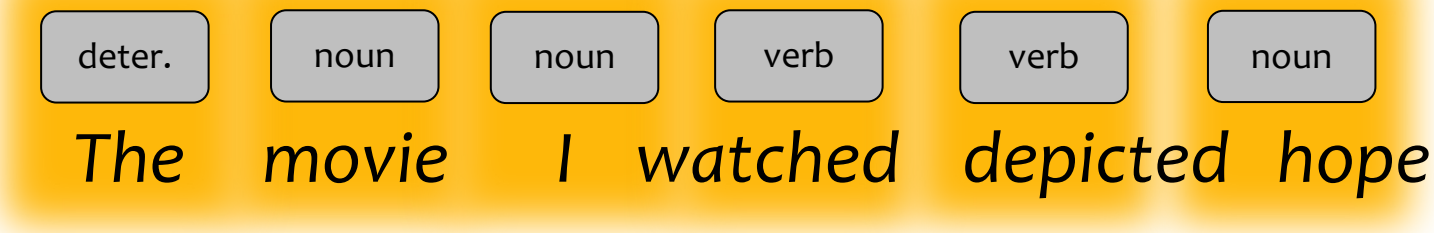
What features should you use?



Feature Engineering for NLP

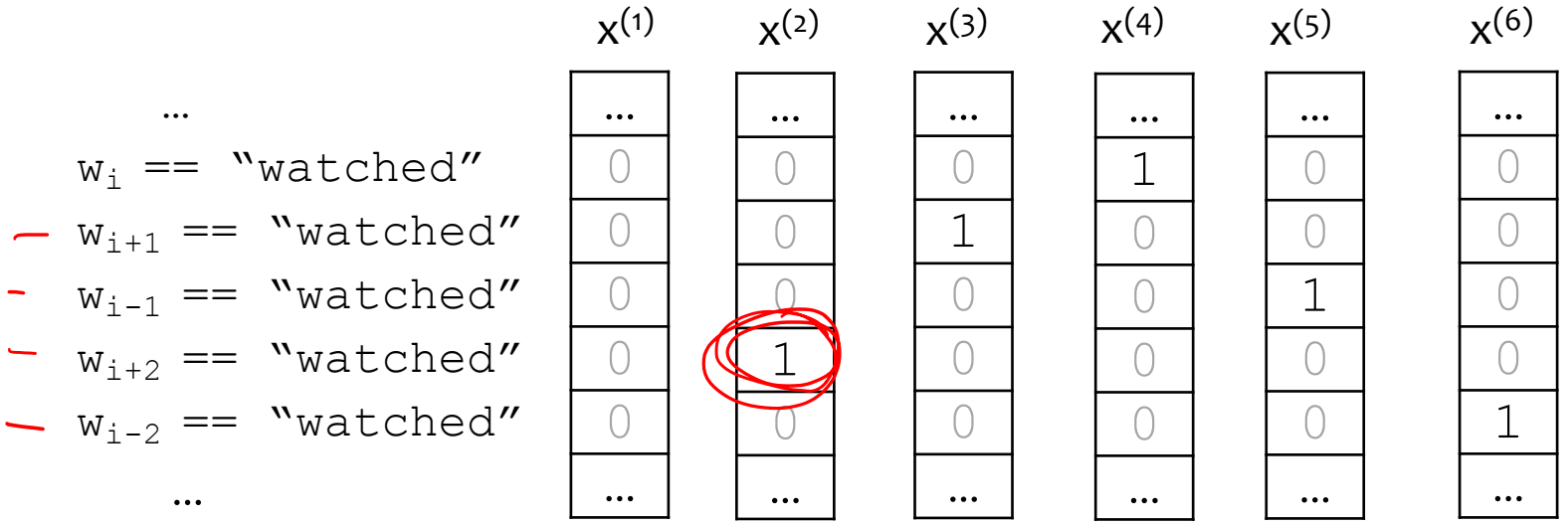
Per-word Features:

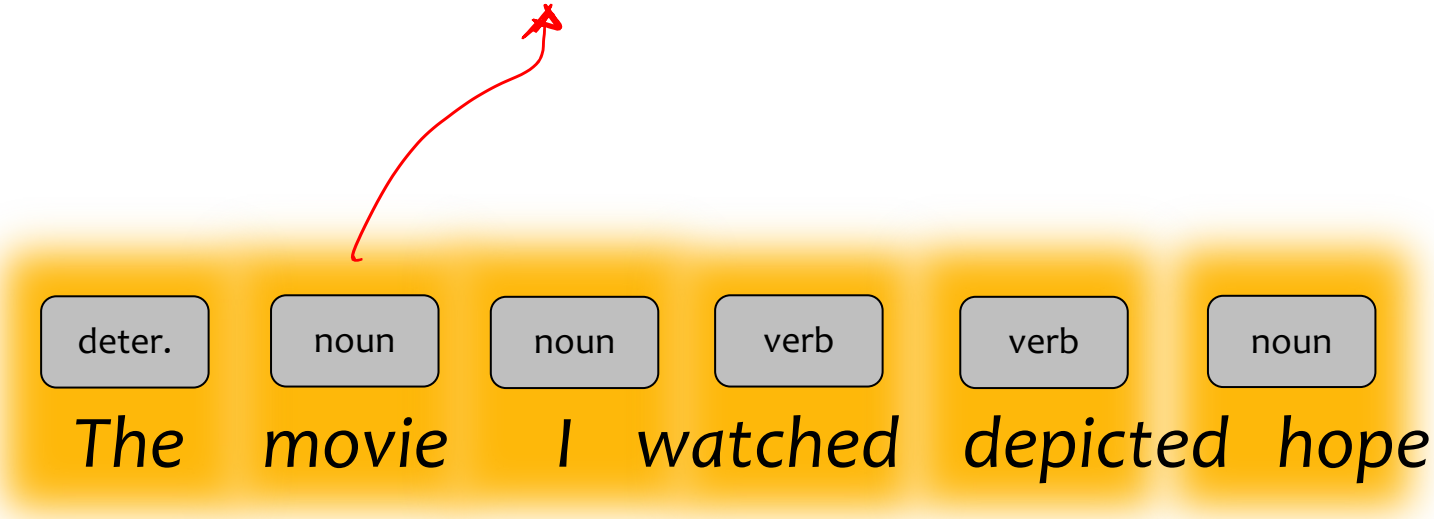
	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$
is-capital(w_i)	1	0	1	0	0	0
endswith(w_i,"e")	1	1	0	0	0	1
endswith(w_i,"d")	0	0	0	1	1	0
endswith(w_i,"ed")	0	0	0	1	1	0
w_i == "aardvark"	0	0	0	0	0	0
w_i == "hope"	0	0	0	0	0	1
…	…	…	…	…	…	…



Feature Engineering for NLP

Context Features:





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
…	…	…	…	…	…	…

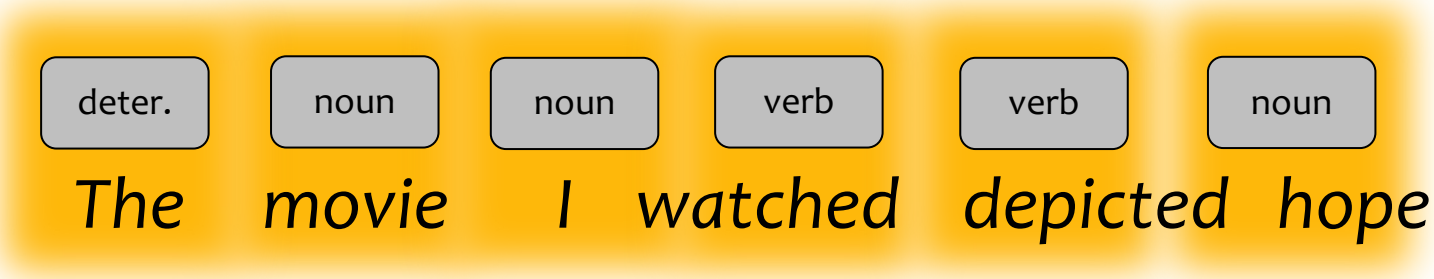
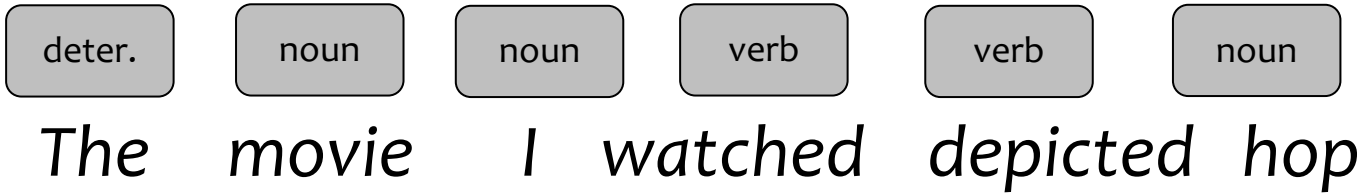


Table from Manning (2011)

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%



Background: Word Embeddings

One-hot vectors

- Standard representation of a word in NLP: 1-hot vector (aka. a string)
- Vectors representing related words share nothing in common

	a	and	be	cat	dog	…	you	zebra
cat:	0	0	0	1	0	…	0	0
dog:	0	0	0	0	1	…	0	0

Word embeddings

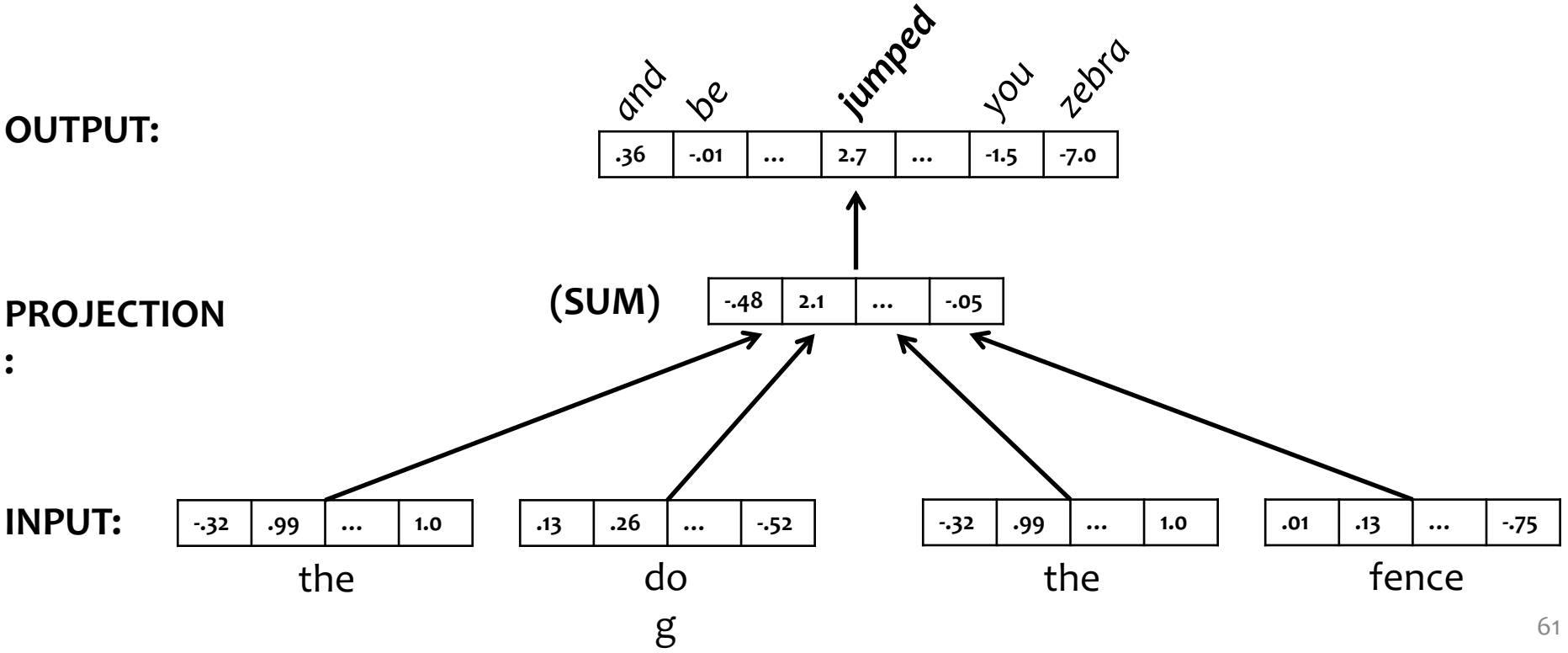
M=1024

- Word embedding: real-valued vector representation of a word in *M* dimensions
- Related words have similar vectors
- Long history in NLP: Term-doc frequency matrices, Reduce dimensionality with {LSA, NNMF, CCA, PCA}, Brown clusters, Vector space models, Random projections, Neural networks / deep learning

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

Background: Word Embeddings

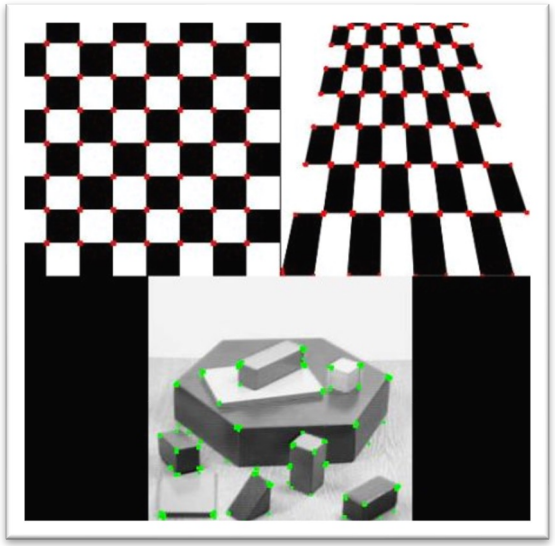
- It's common to use **neural-network** trained embeddings
  - Key idea: learn embeddings which are good at reconstructing the context of a word
  - Popular across HLT (speech, NLP)
- The Continuous Bag-of-words Model (CBOW) (Mikolov et al., 2013) maximizes the likelihood of a word given its context:



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.

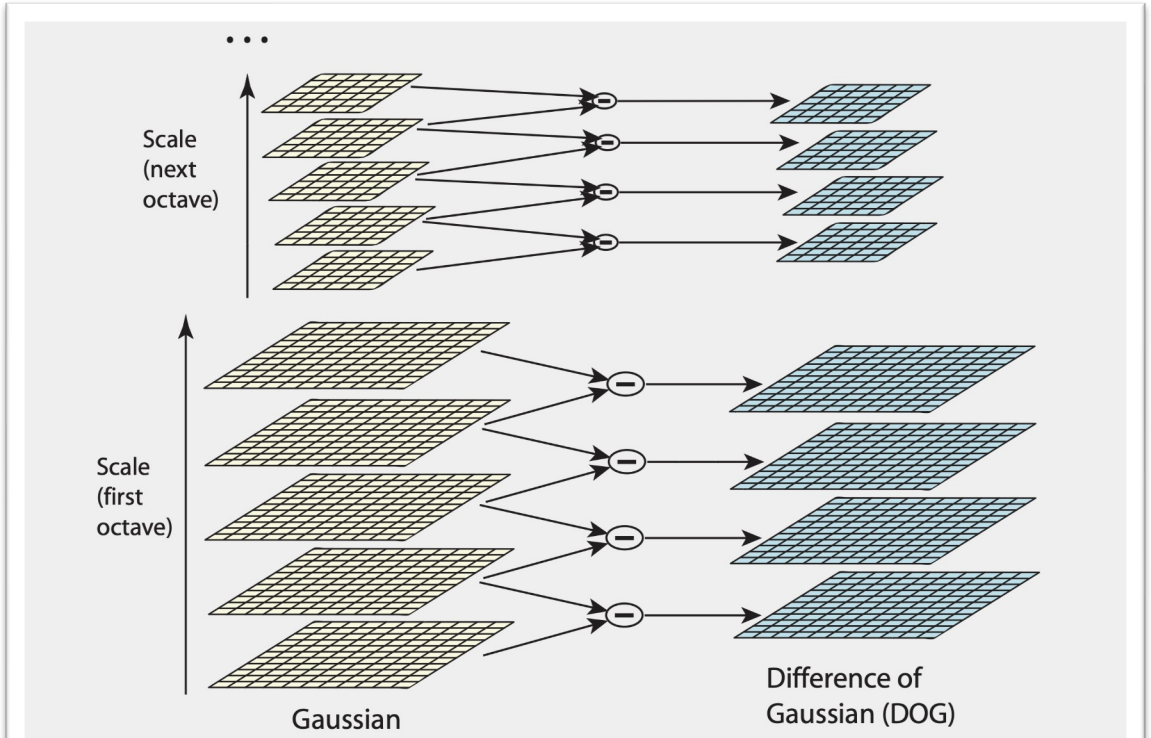


Figure 1: For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled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

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

Examples:
- Perceptron
- Linear regression
- Logistic regression

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

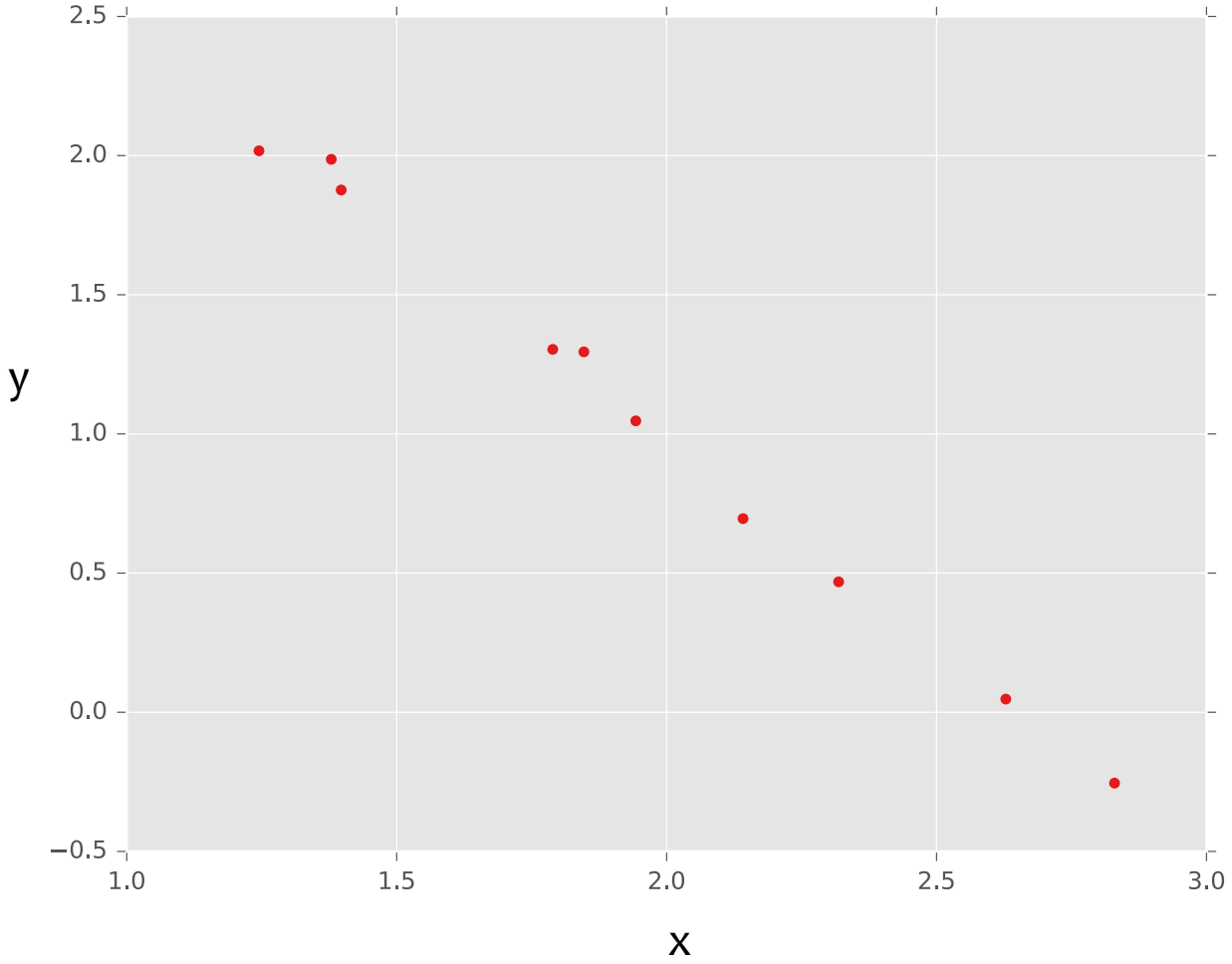
log $b_j(x) = \log(x)$

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

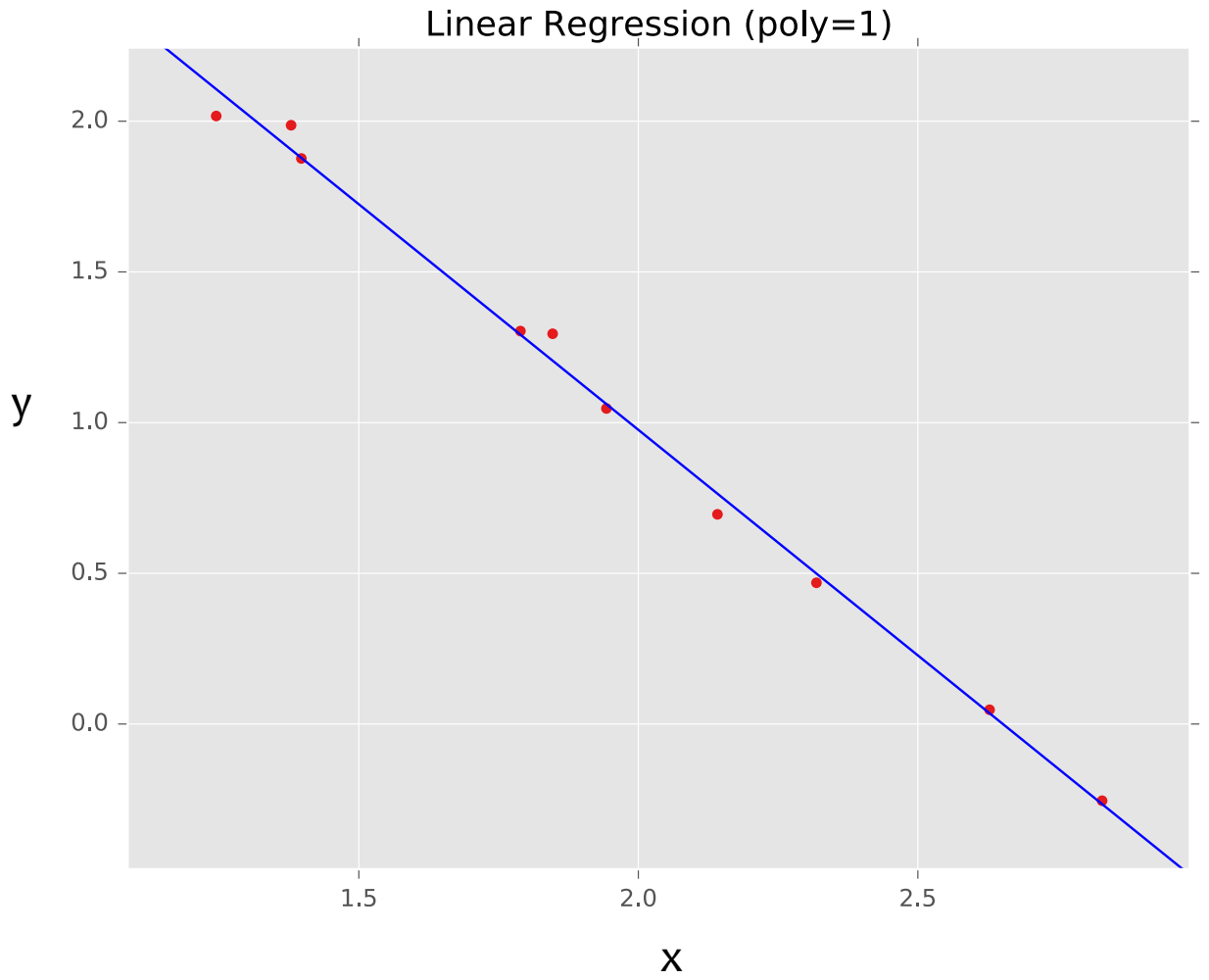


Example: Linear Regression

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

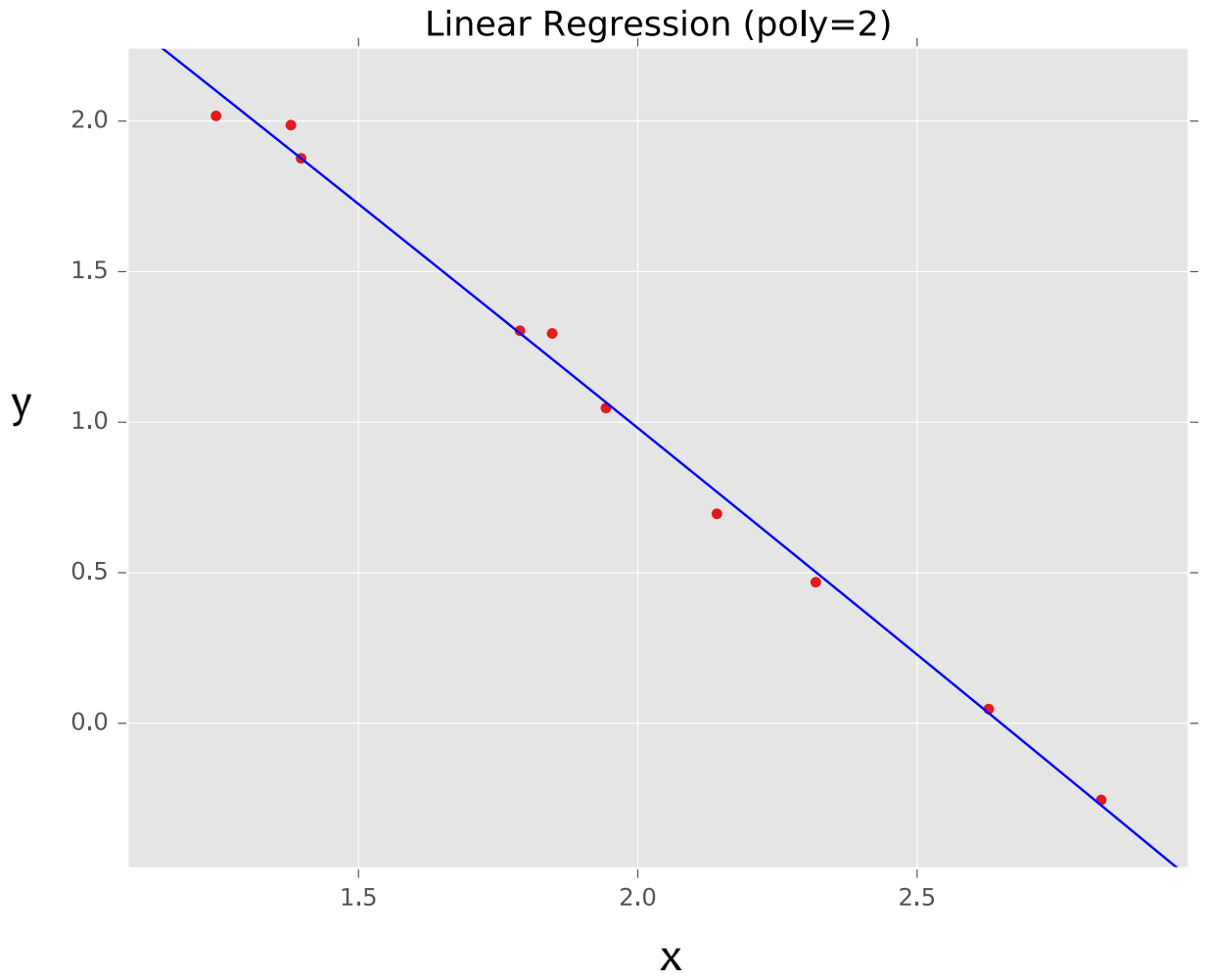


Example: Linear Regression

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

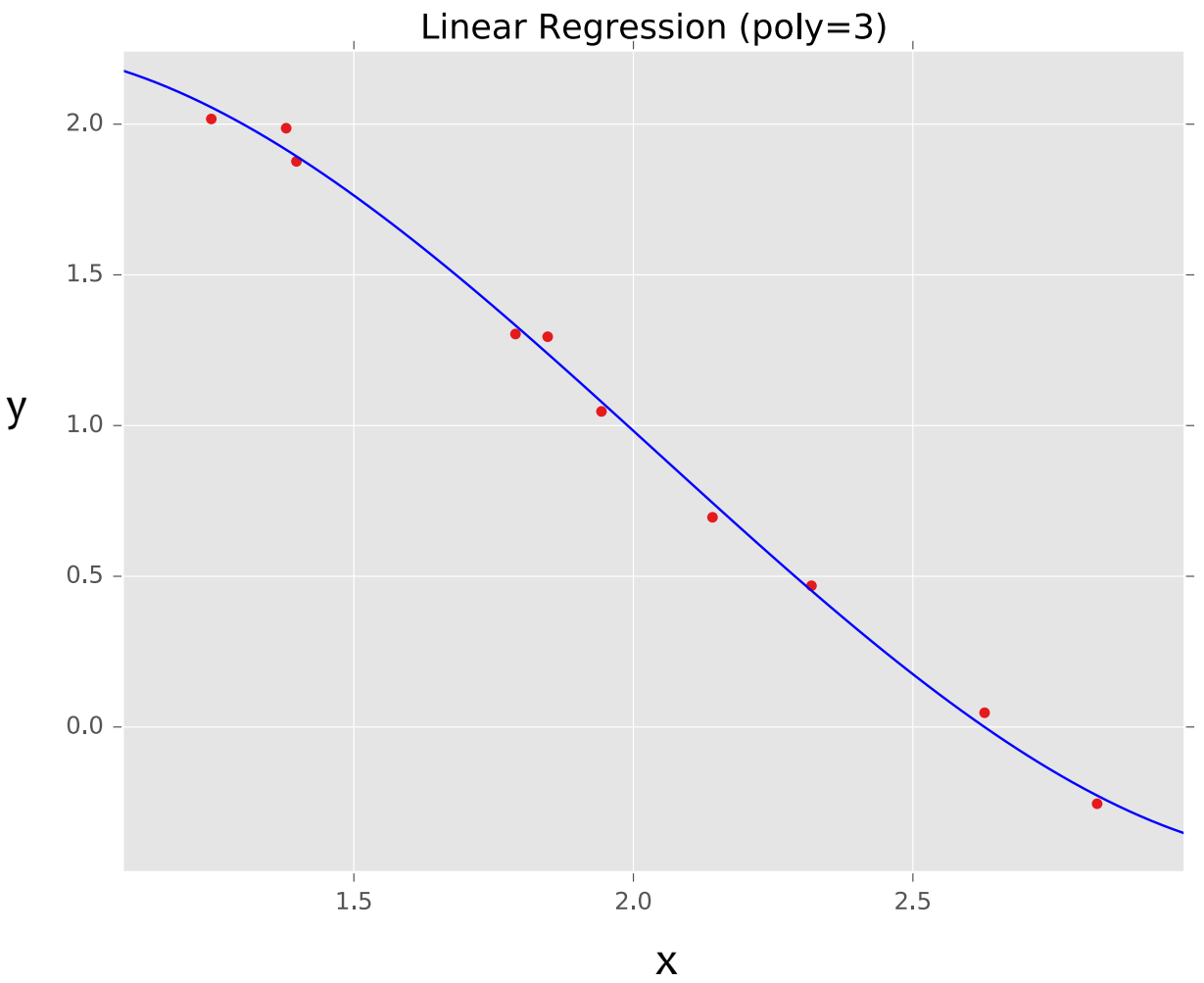


Example: Linear Regression

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

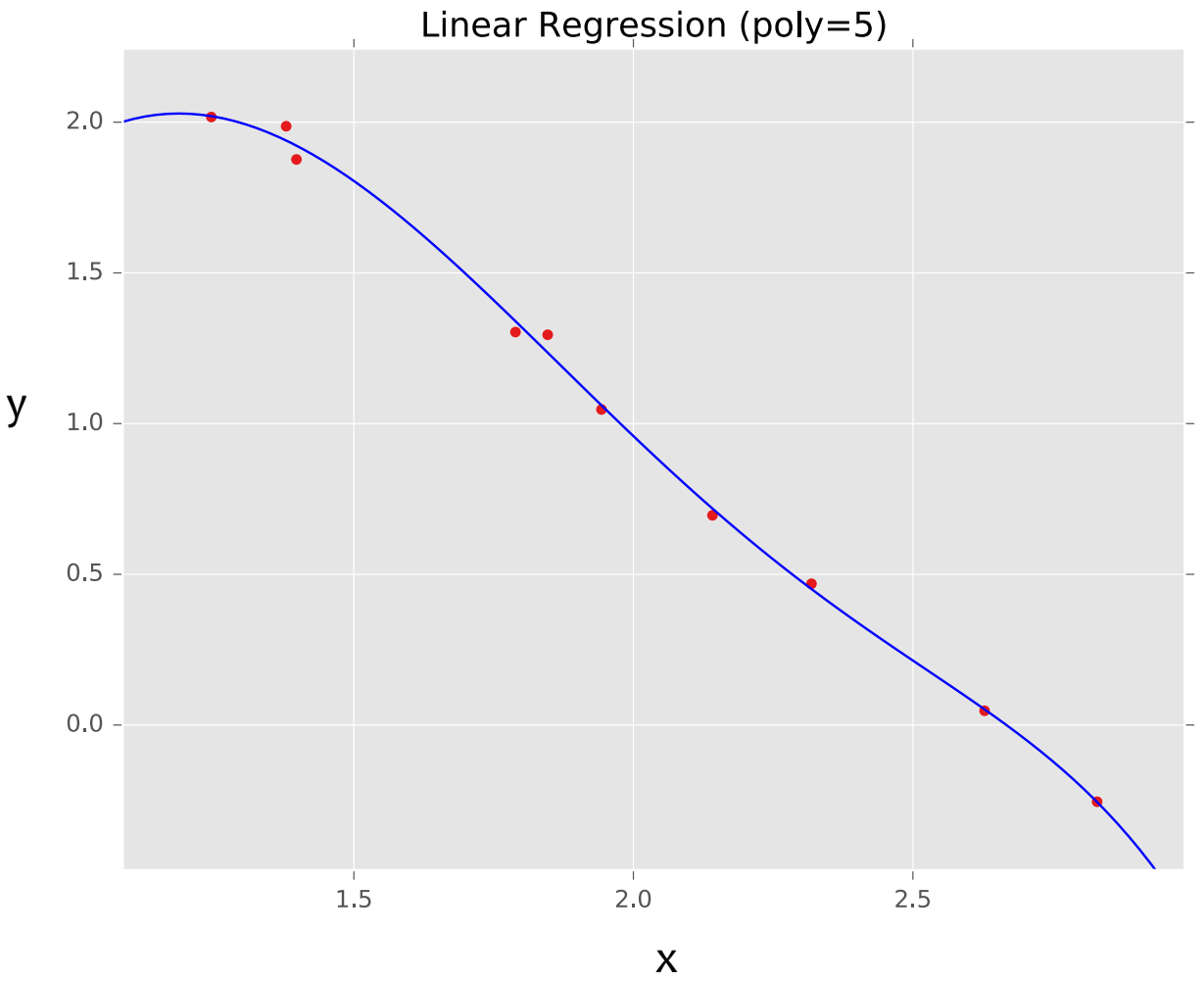


Example: Linear Regression

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

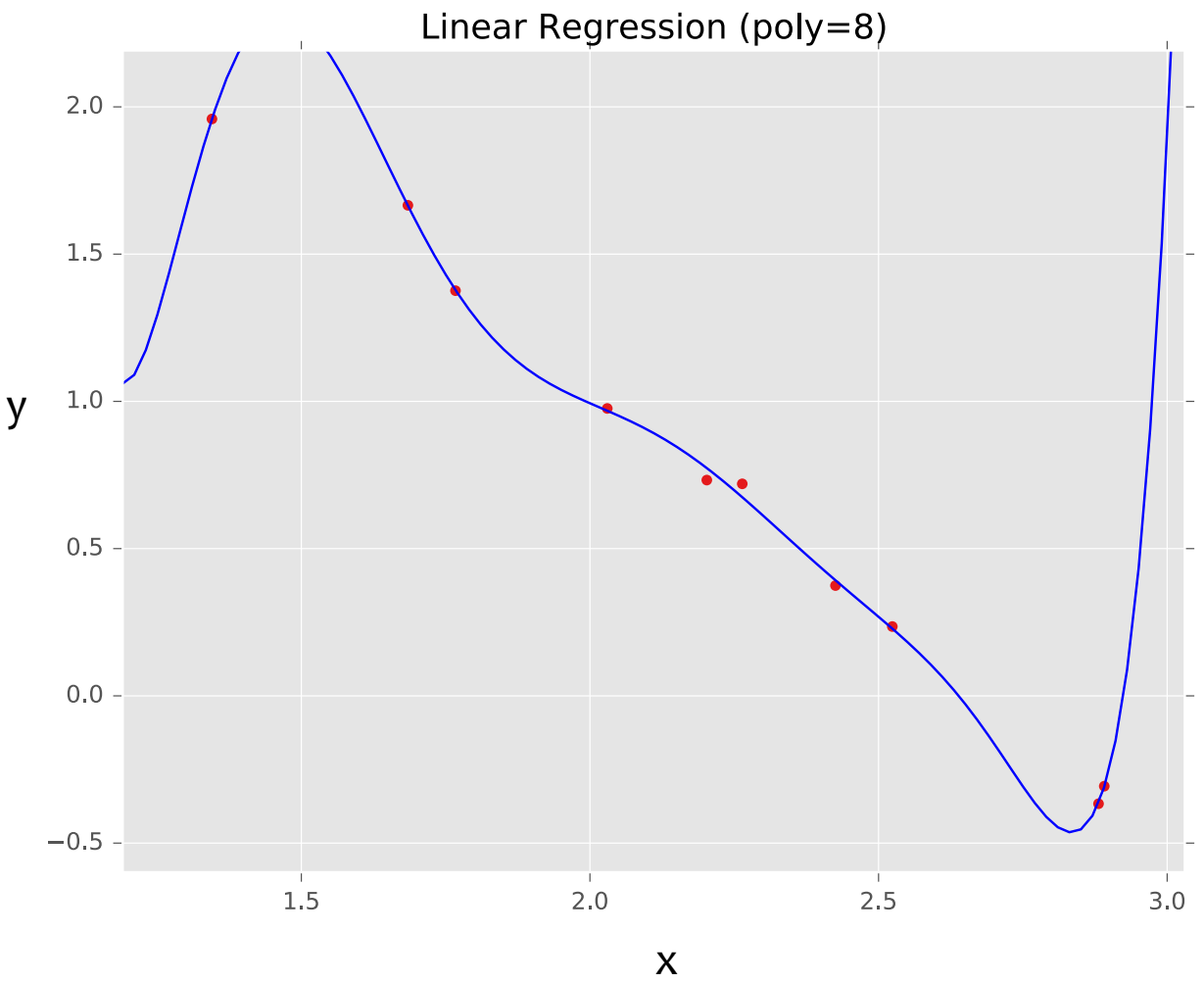


Example: Linear Regression

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

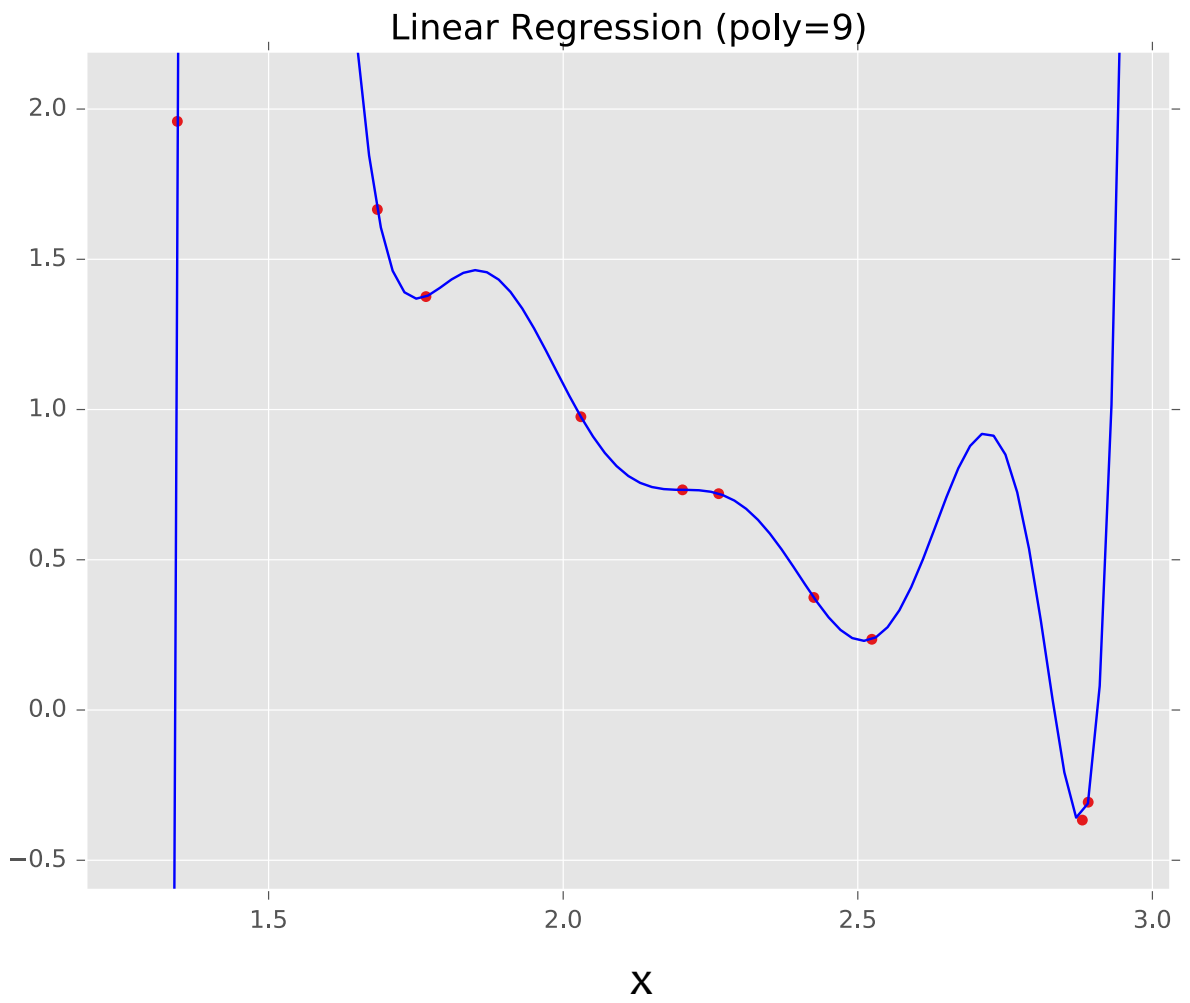


Example: Linear Regression

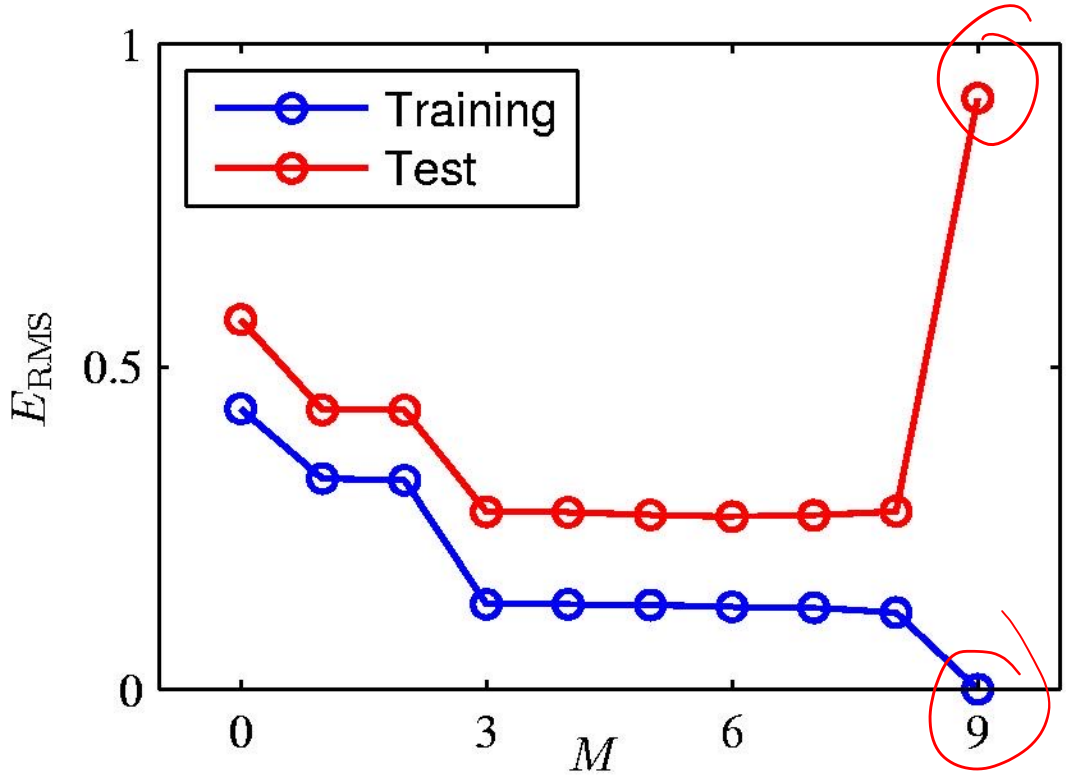
Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + b$ where f(.) is a polynomial basis function

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

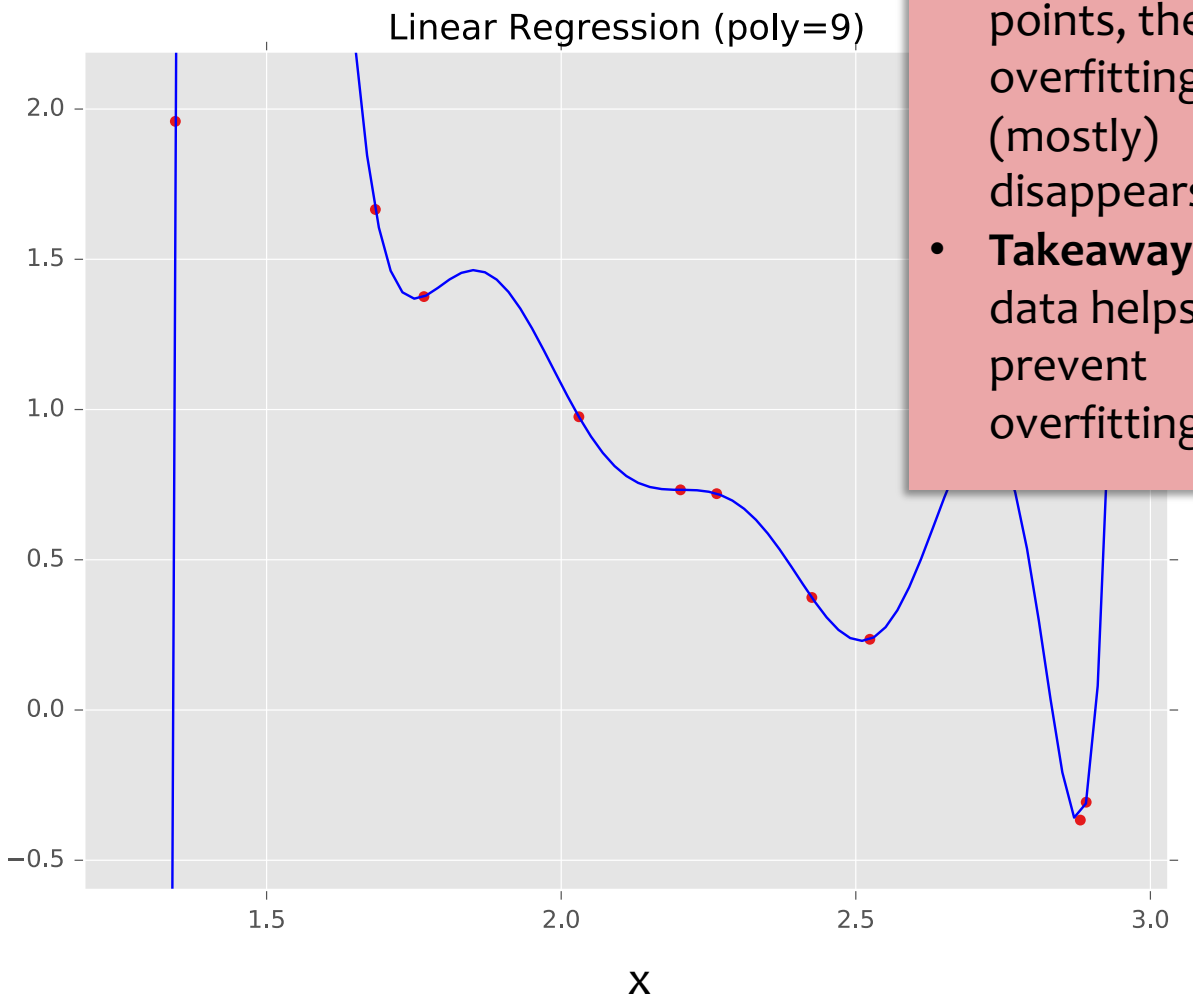
	$M=0$	$M=1$	$M=3$	$M=9$
θ_0	0.19	0.82	0.31	0.35
θ_1		-1.27	7.99	232.37
θ_2			-25.43	-5321.83
θ_3			17.37	48568.31
θ_4				-231639.30
θ_5				640042.26
θ_6				-1061800.52
θ_7				1042400.18
θ_8				-557682.99
θ_9				125201.43

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

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	…	$(1.9)^9$

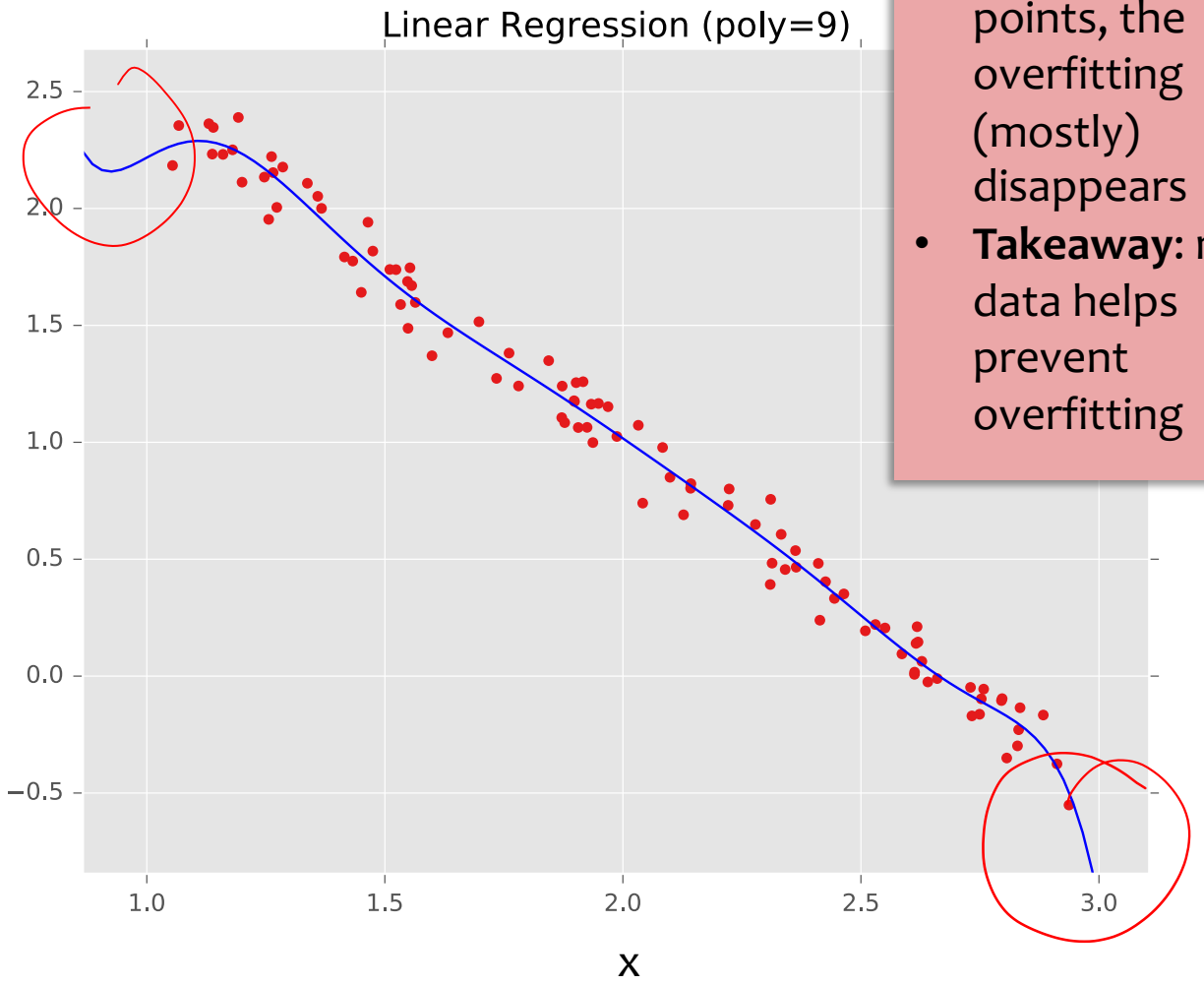


- With just N = 10 points we overfit!
- But with N = 100 points, the overfitting (mostly) disappears
- **Takeaway:** more data helps prevent overfitting

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

- 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**)

$$\vec{\theta} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 107 \\ -25 \\ 0 \\ 0 \end{bmatrix} \qquad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$

$$\vec{\theta}^T \vec{x} = \theta_4 x_4 + \theta_5 x_5$$

Regularization

- **Given** objective function: J(θ)
- **Goal** is to find:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta}) + \lambda r(\boldsymbol{\theta})$$

- **Key idea:** Define regularizer r(θ) s.t. we tradeoff between fitting the data and keeping the model simple
- **Choose form of regularizer:**
  - Common choice: p-norm: $r(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_p = \left[\sum_{m=1}^{M} |\theta_m|^p\right]^{(\frac{1}{p})}$

p	$r(\boldsymbol{\theta})$	**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