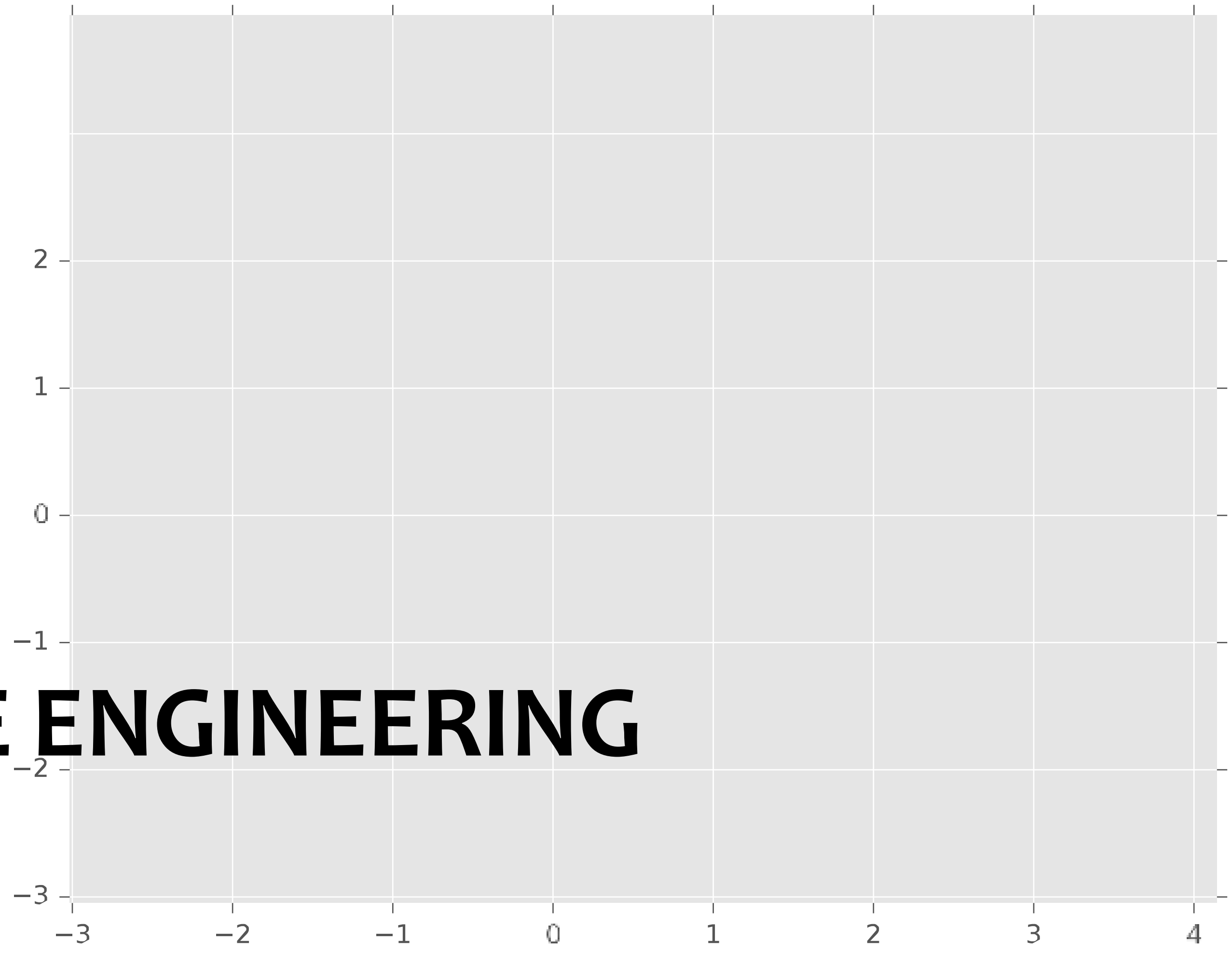
CS182: Introduction to Machine Learning – Feature Engineering Regularization

Yujiao Shi
SIST, ShanghaiTech
Spring, 2025

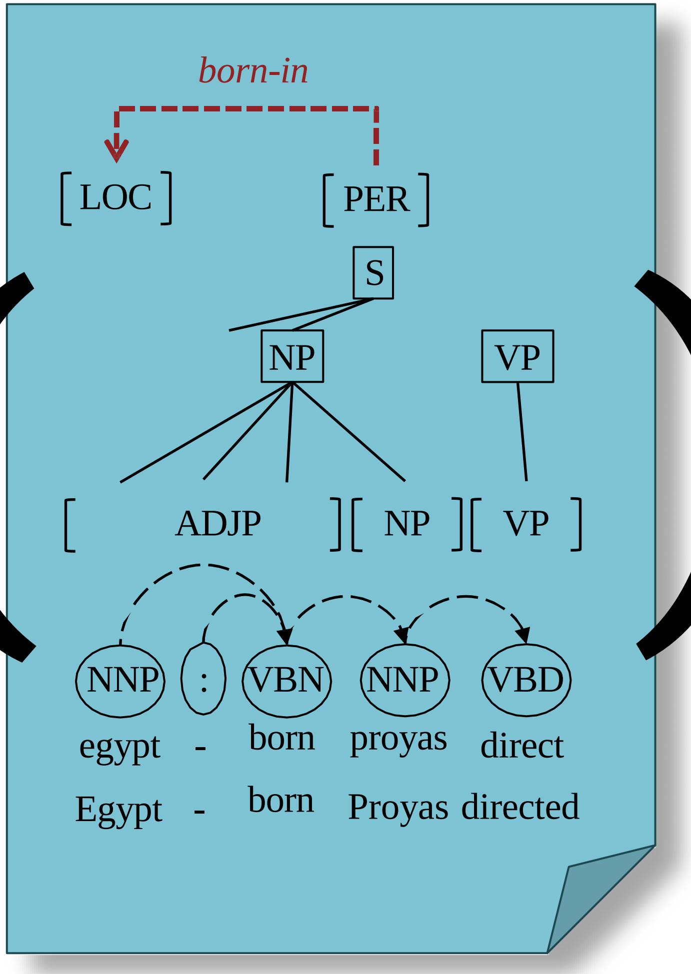
Slides Courtesy of Matt Gormley & Henry Chai, CMU



FEATURE ENGINEERING

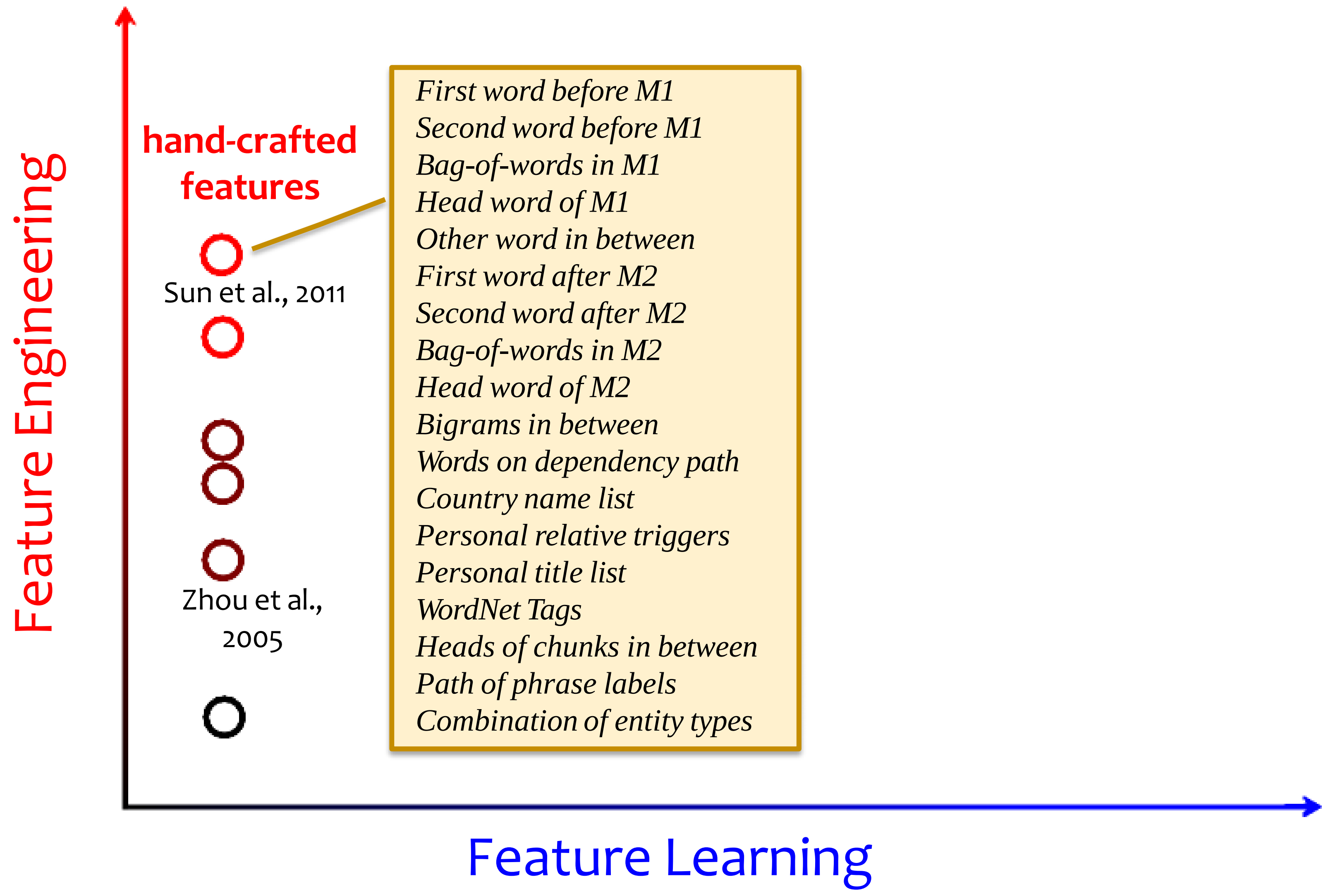
Handcrafted Features

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



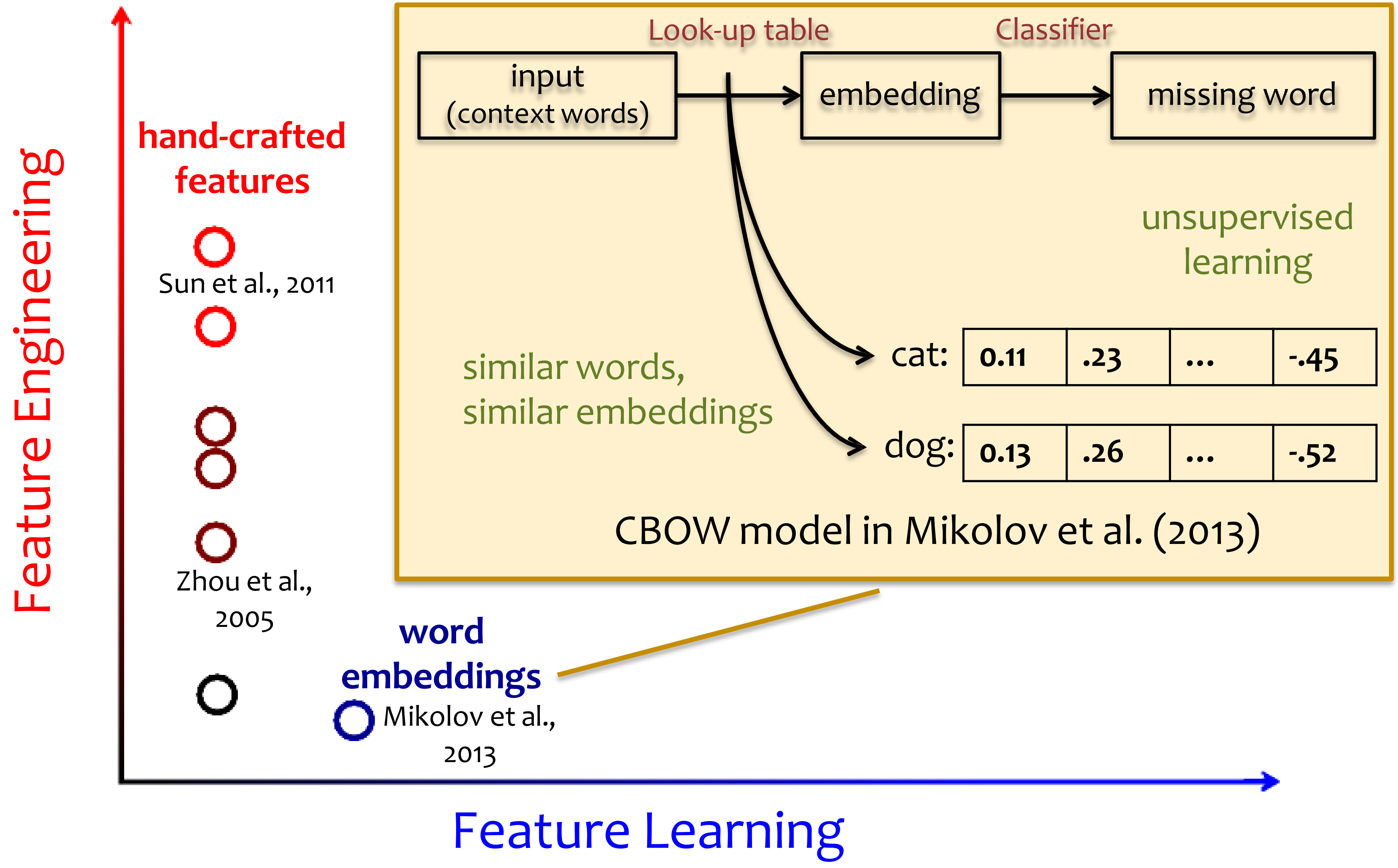
$$))$$

Where do features come from?

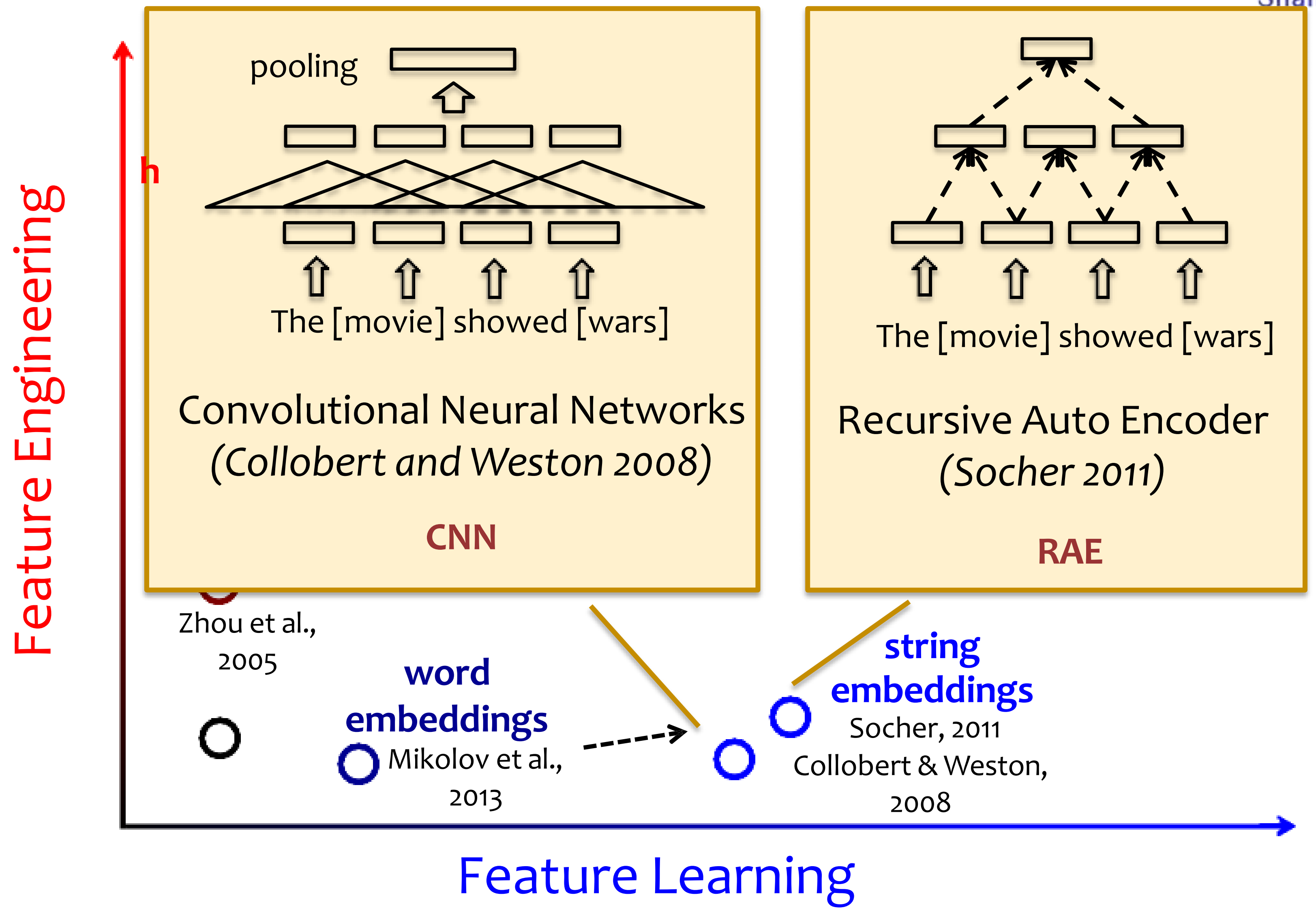


Where do features come from?

上海科技大学
ShanghaiTech University

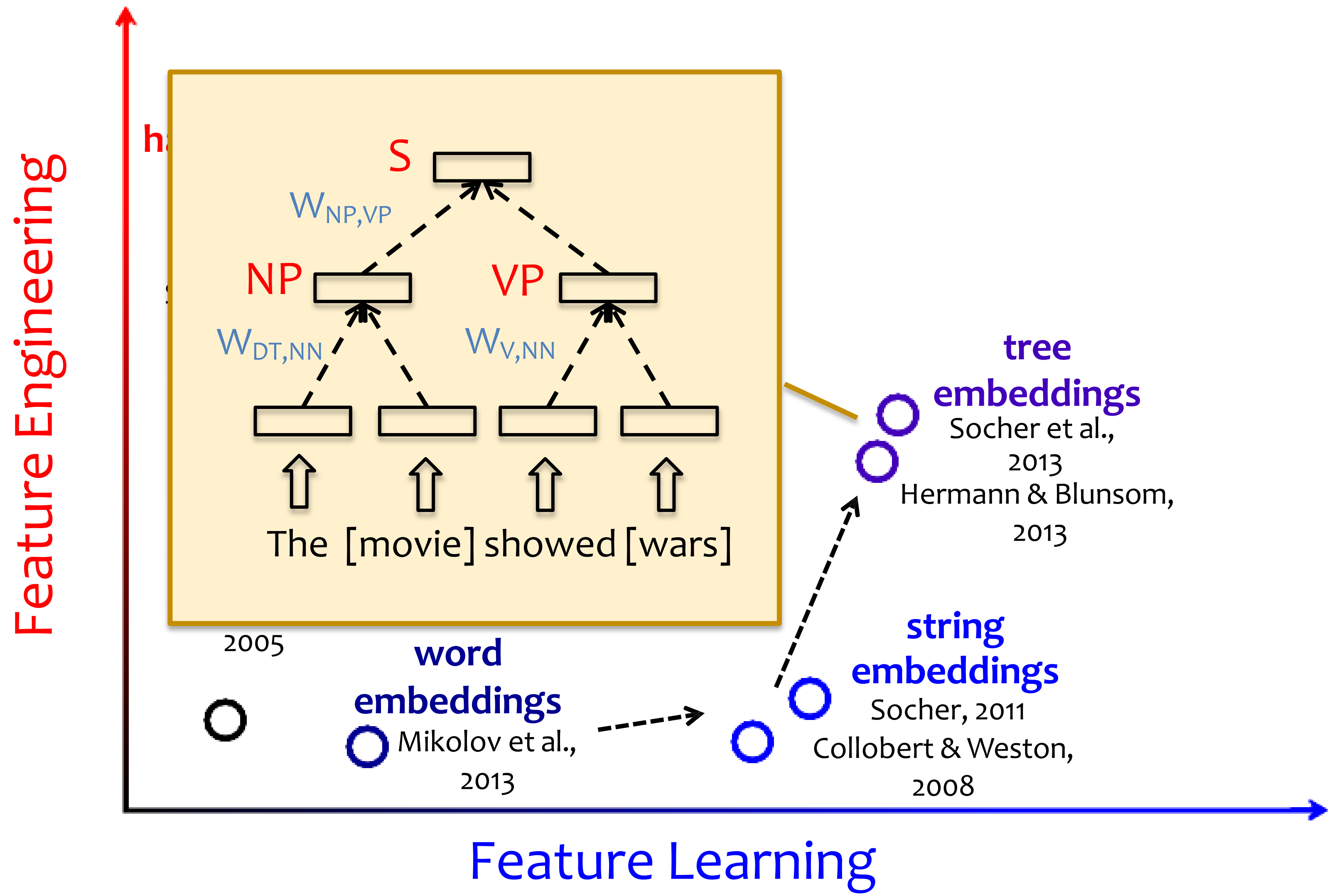


Where do features come from?



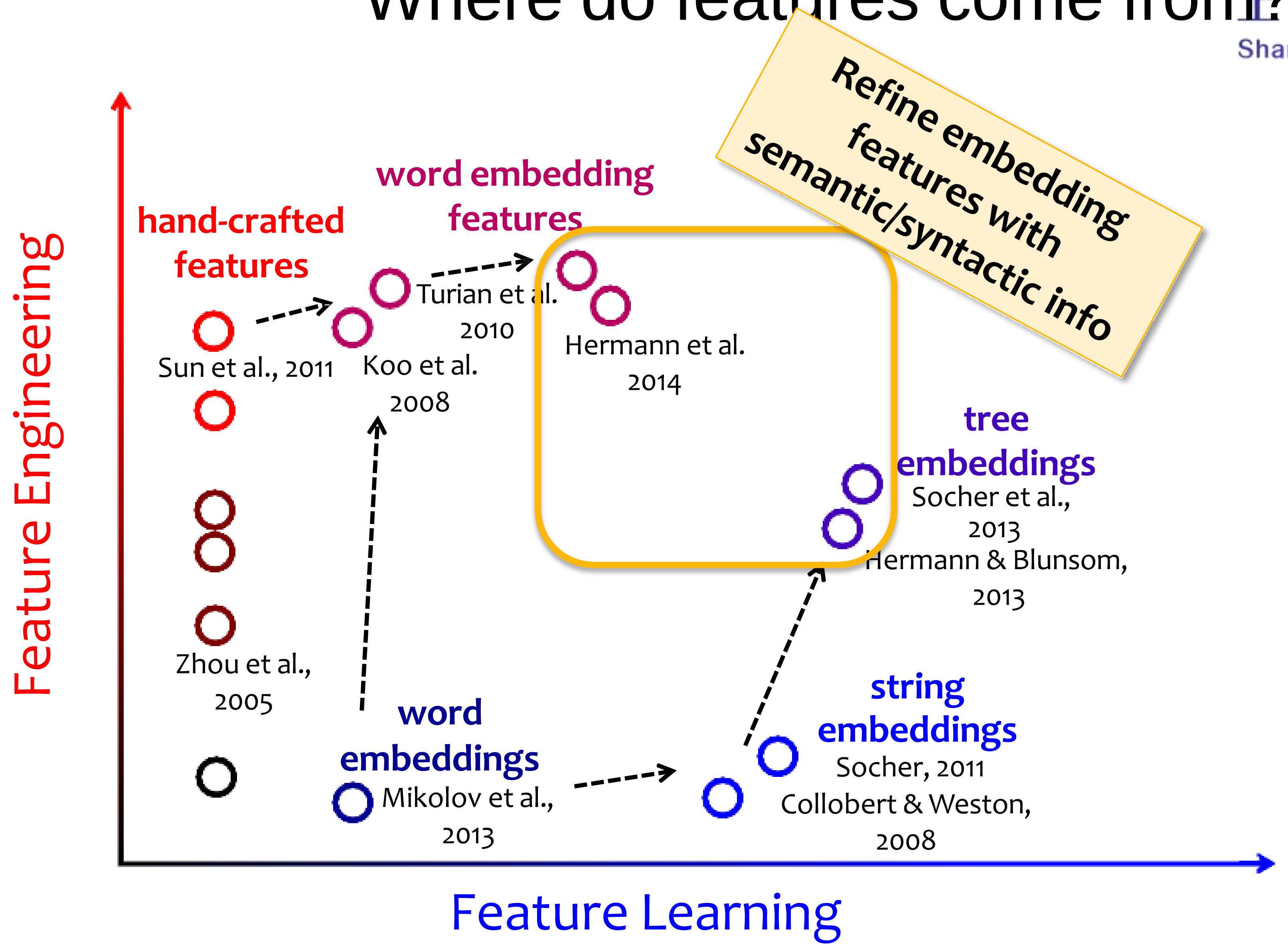
Where do features come from?

上海科技大学
ShanghaiTech University



Where do features come from?

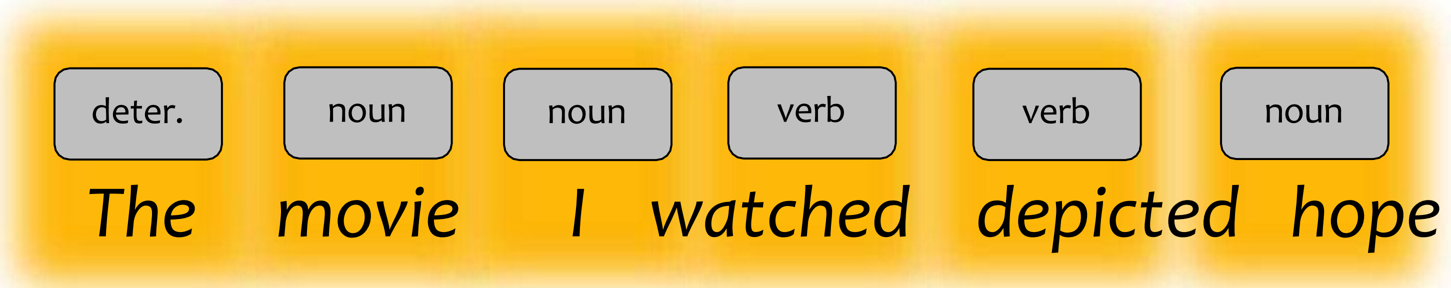
上海科技大学
ShanghaiTech University



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*

上海科技大学 ShanghaiTech University

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*

Feature Engineering for NLP

上海科技大学 ShanghaiTech University

Table from Manning (2011)

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*

Background: Word Embeddings

上海科技大学 ShanghaiTech University

One-hot vectors

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

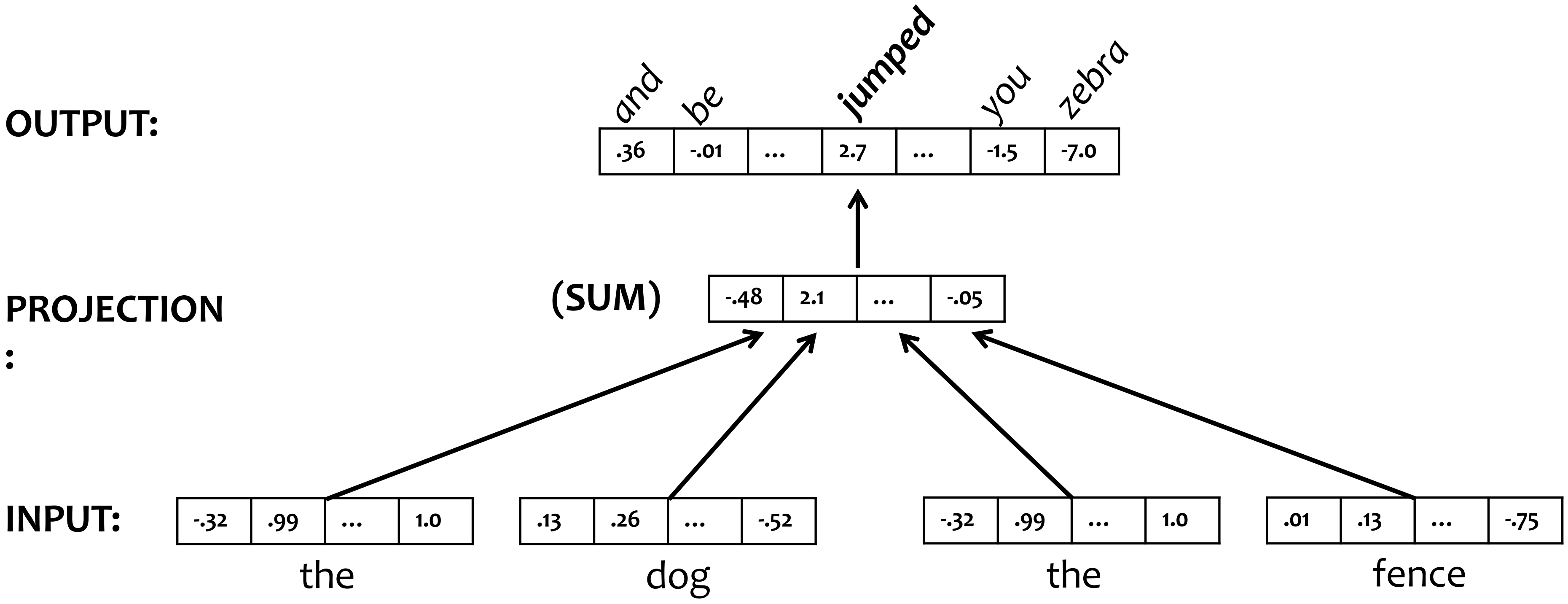
	a	and	be	cat	dog		you	zebra
cat:	0	0	0	1	0	…	0	0
dog:	0	0	0	0	1	…	0	0

- Word 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

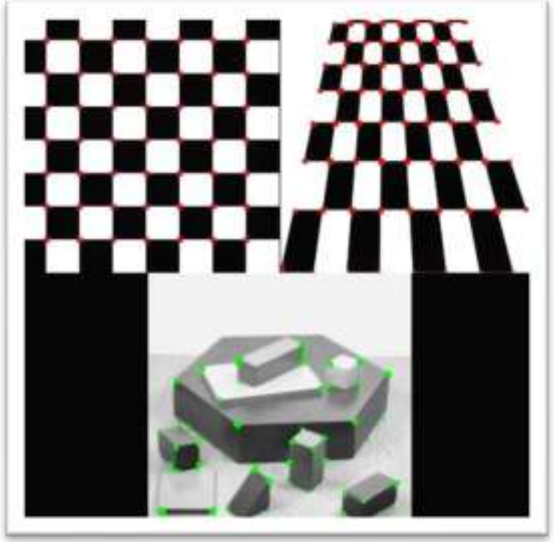
- It's common to use **neural-network** trained embeddings
  - Key idea: learn embeddings which are good at reconstructing the context of a word
  - Popular across HLT (speech, NLP)
- The Continuous Bag-of-words Model (CBOW) (Mikolov et al., 2013) maximizes the likelihood of a word given its context:



Feature Engineering for CV

Edge detection (Canny)

Corner Detection (Harris)



Figures from http://opencv.org

上海科技大学 ShanghaiTech University

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.

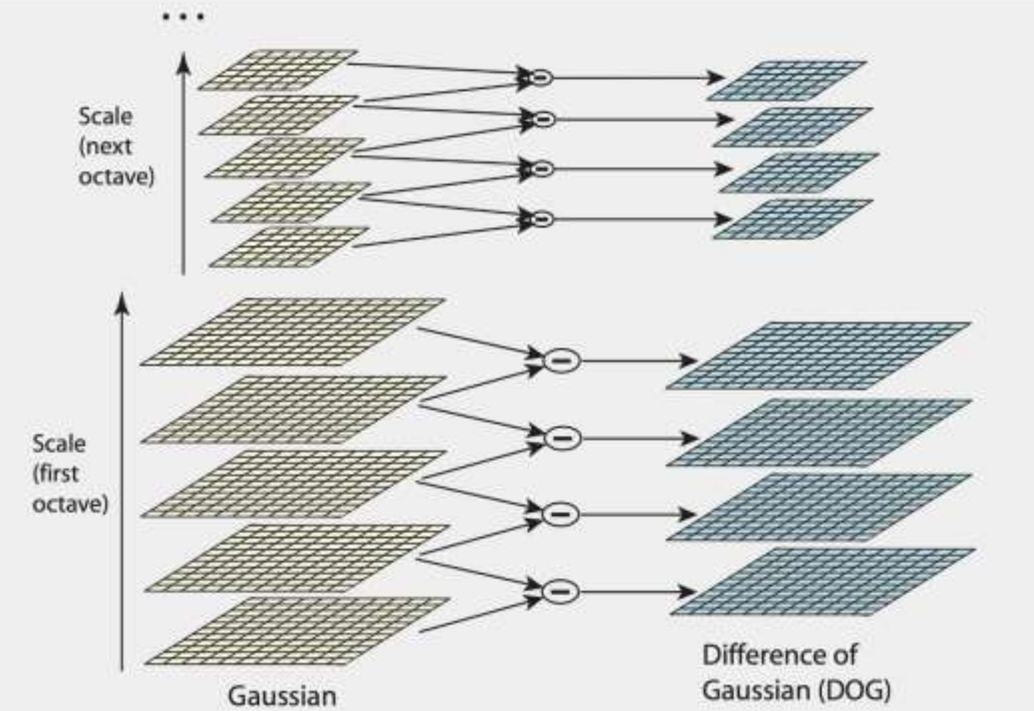


Figure 1: For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled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

Figure from Lowe (1999) and Lowe (2004)

NON-LINEAR FEATURES

Nonlinear Features

上海科技大学 ShanghaiTech University

- 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**

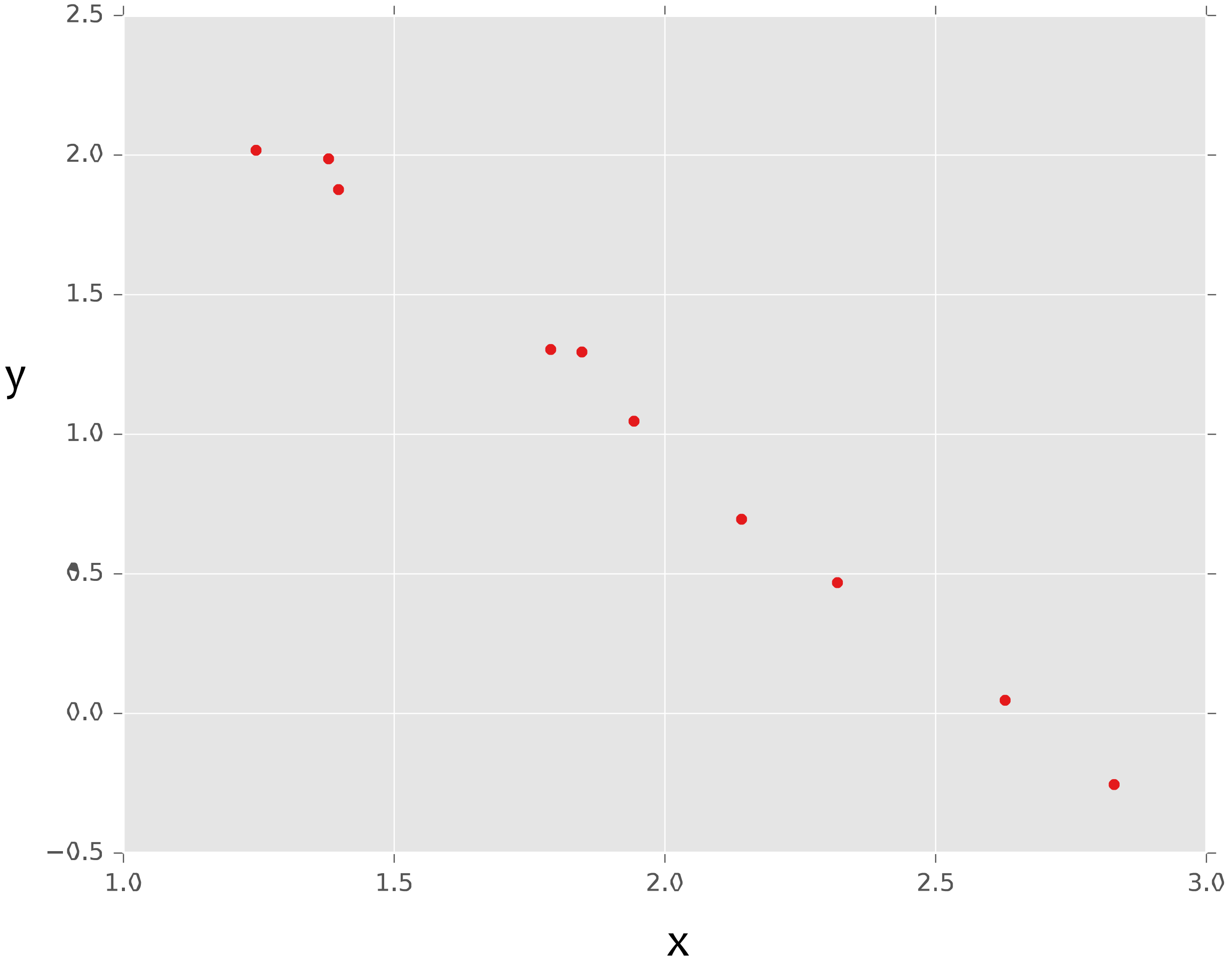
Examples:
- Perceptron
- Linear regression
- Logistic regression

Example: Linear Regression

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

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



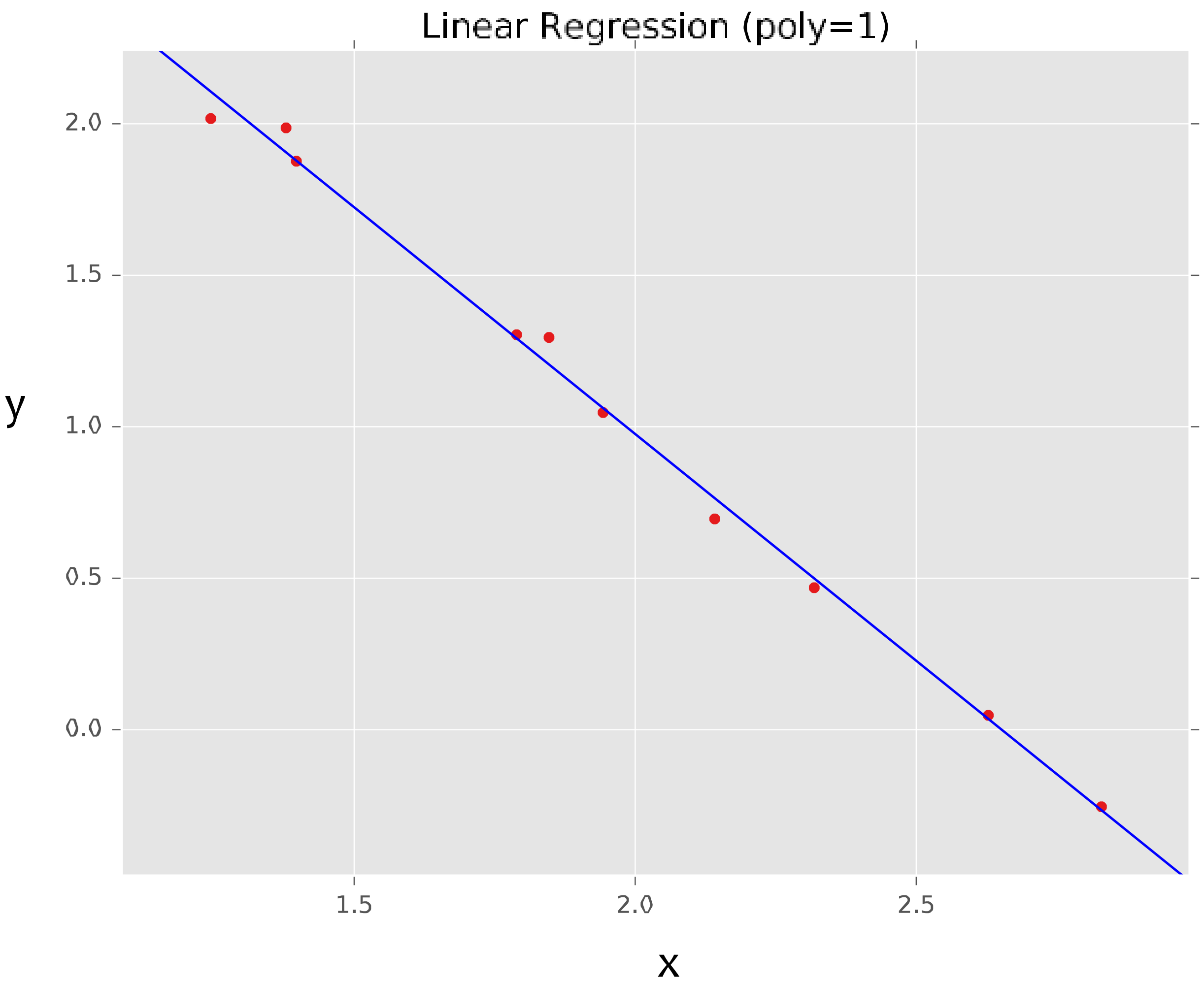
Example: Linear Regression

上海科技大学 ShanghaiTech University

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

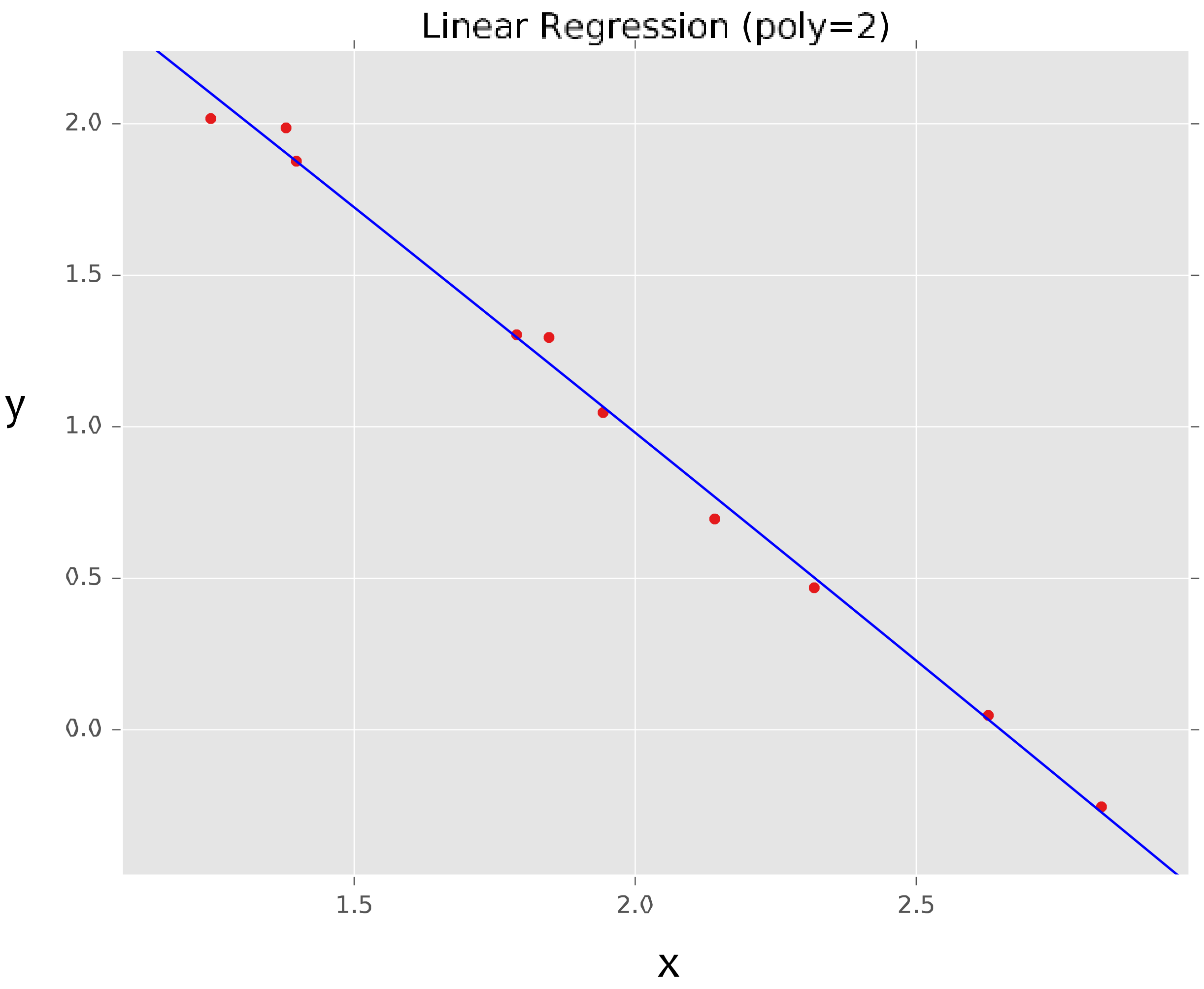


Example: Linear Regression

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



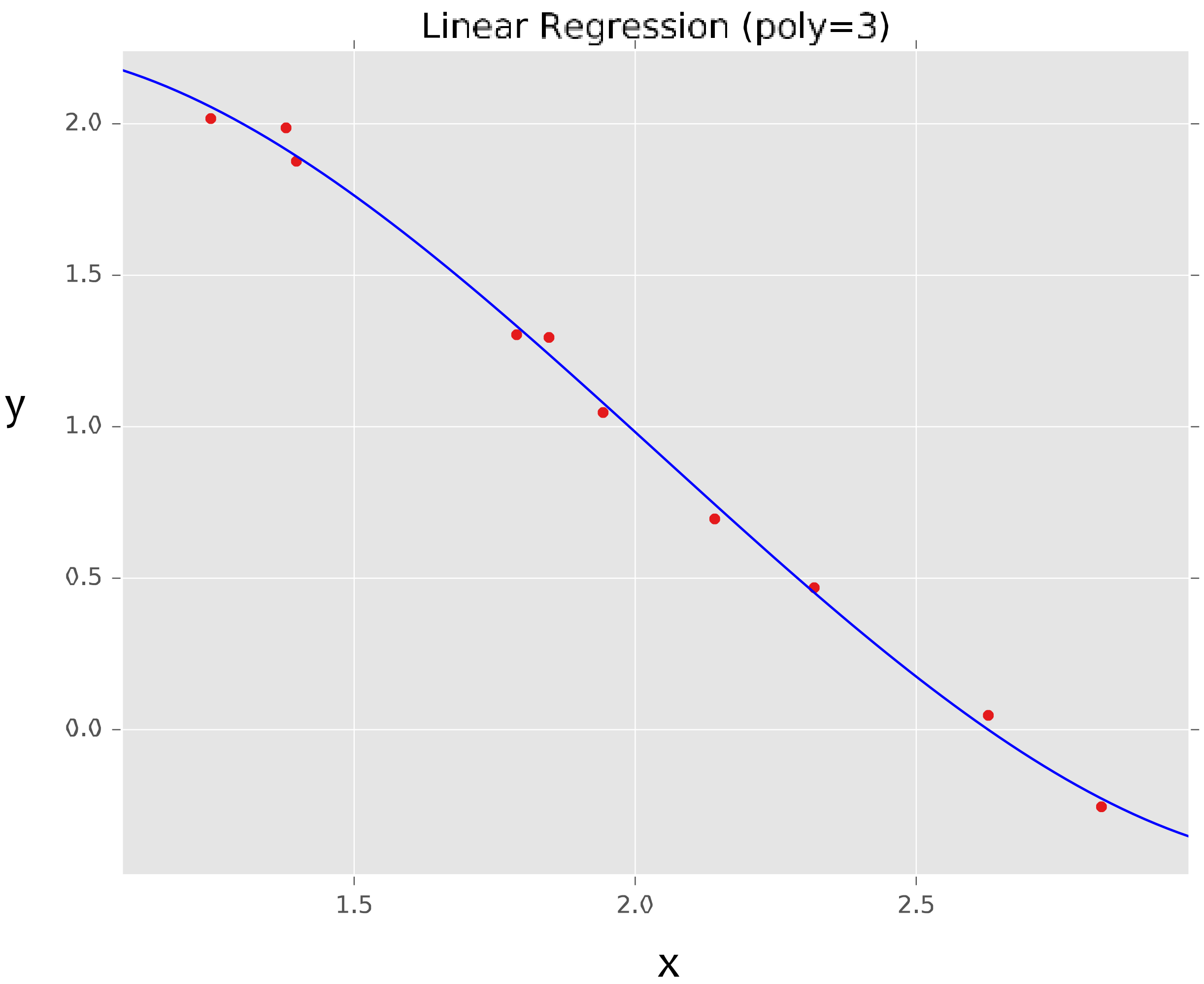
Example: Linear Regression

上海科技大学 ShanghaiTech University

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



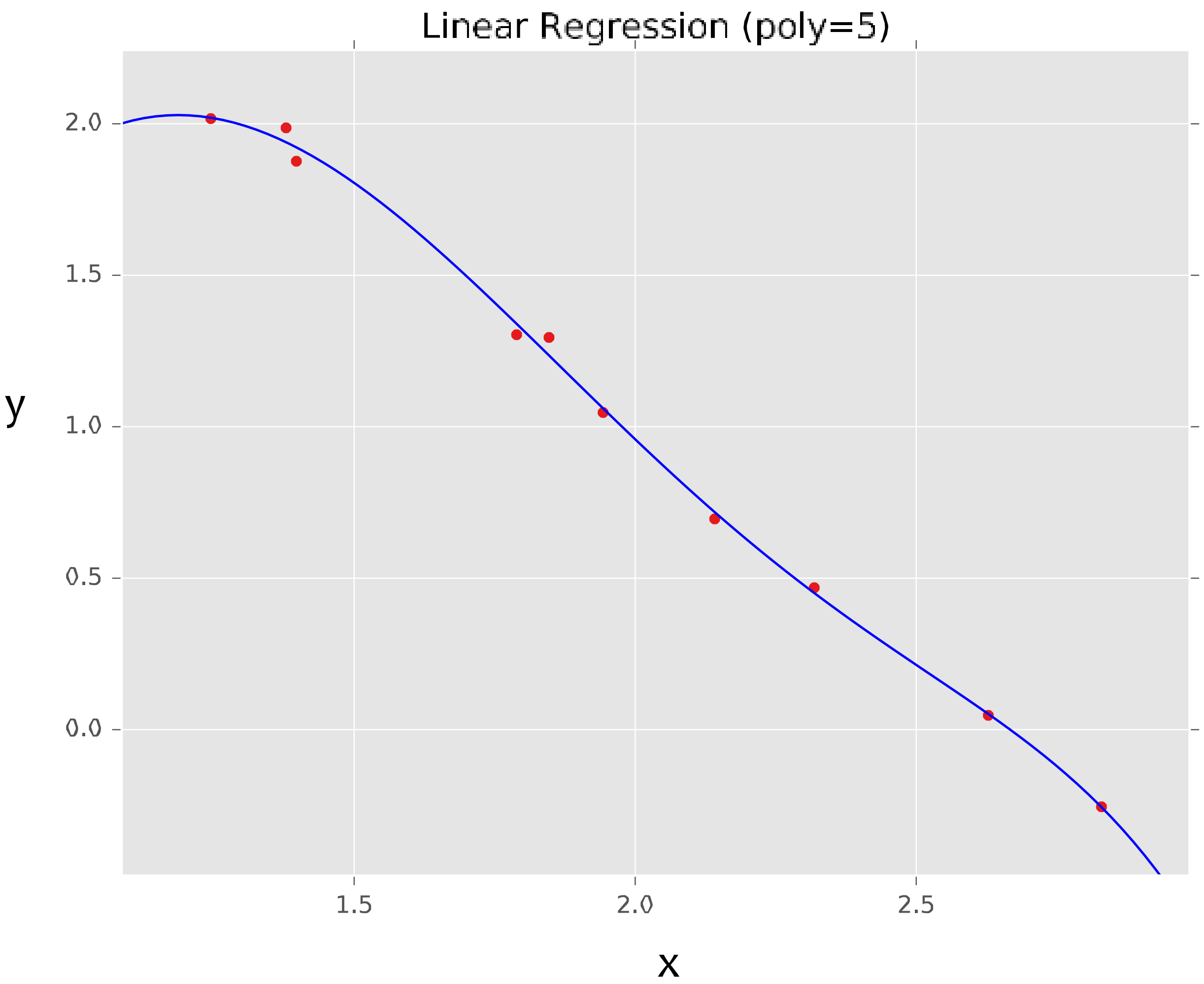
Example: Linear Regression

上海科技大学 ShanghaiTech University

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

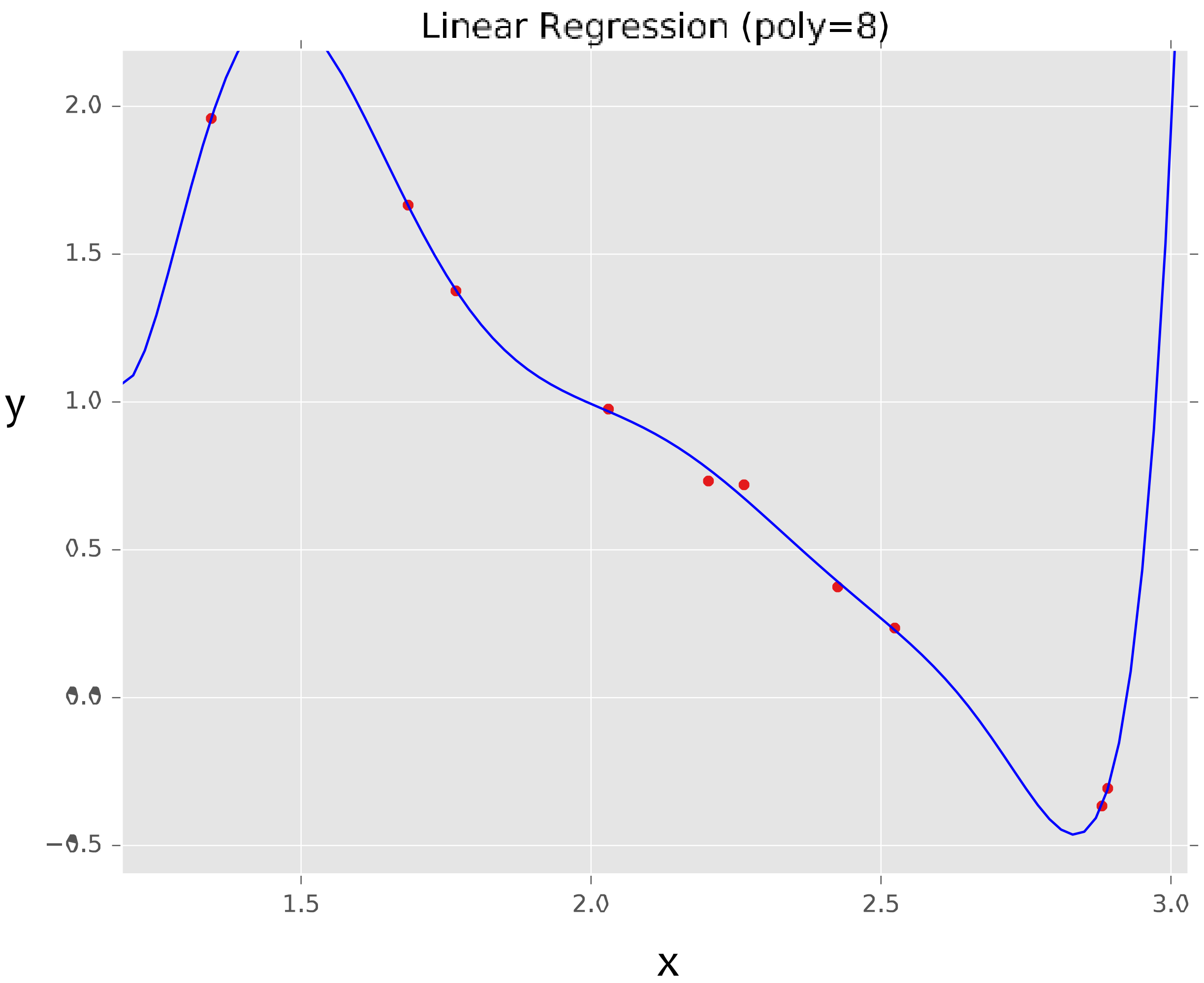


Example: Linear Regression

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

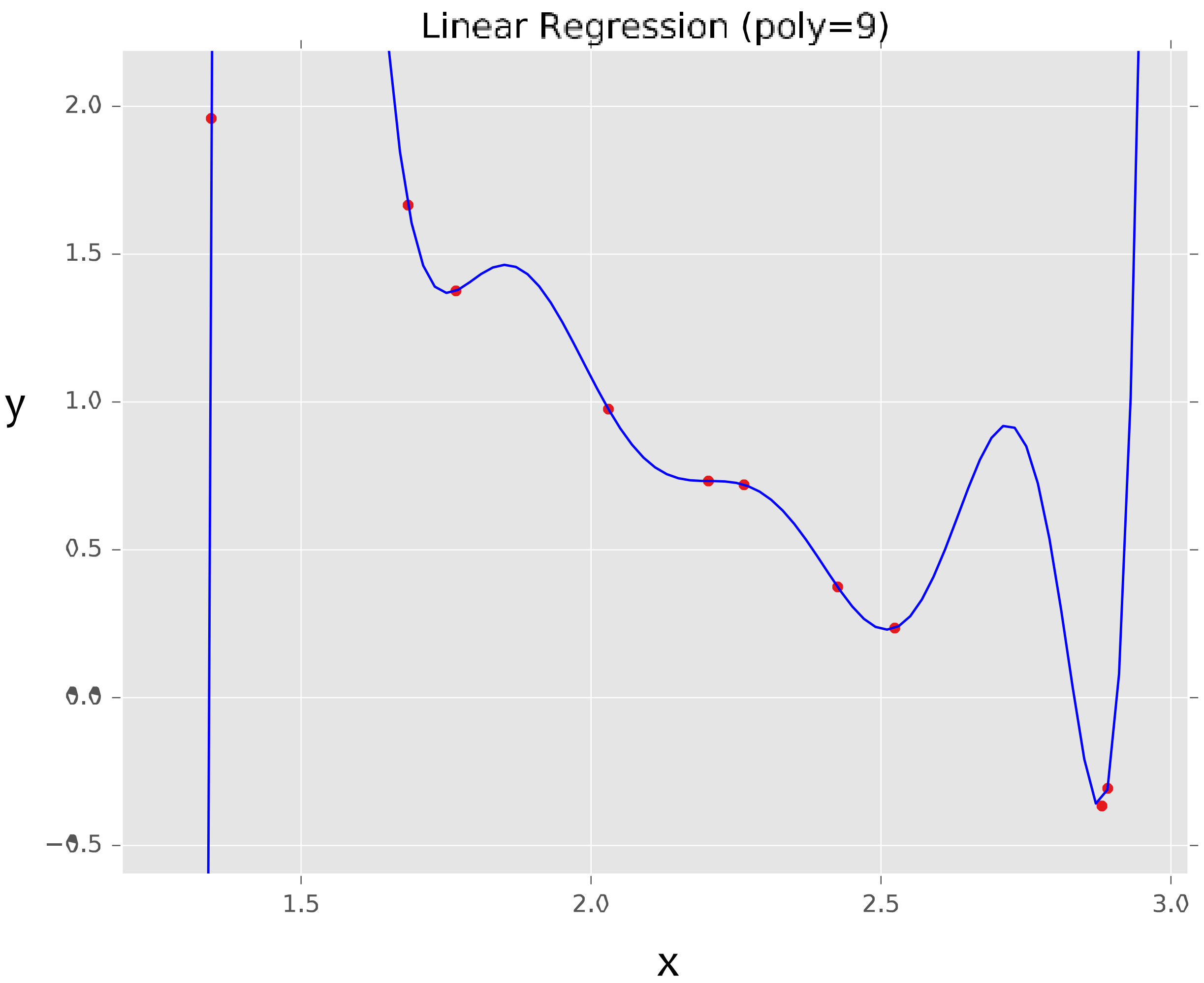


Example: Linear Regression

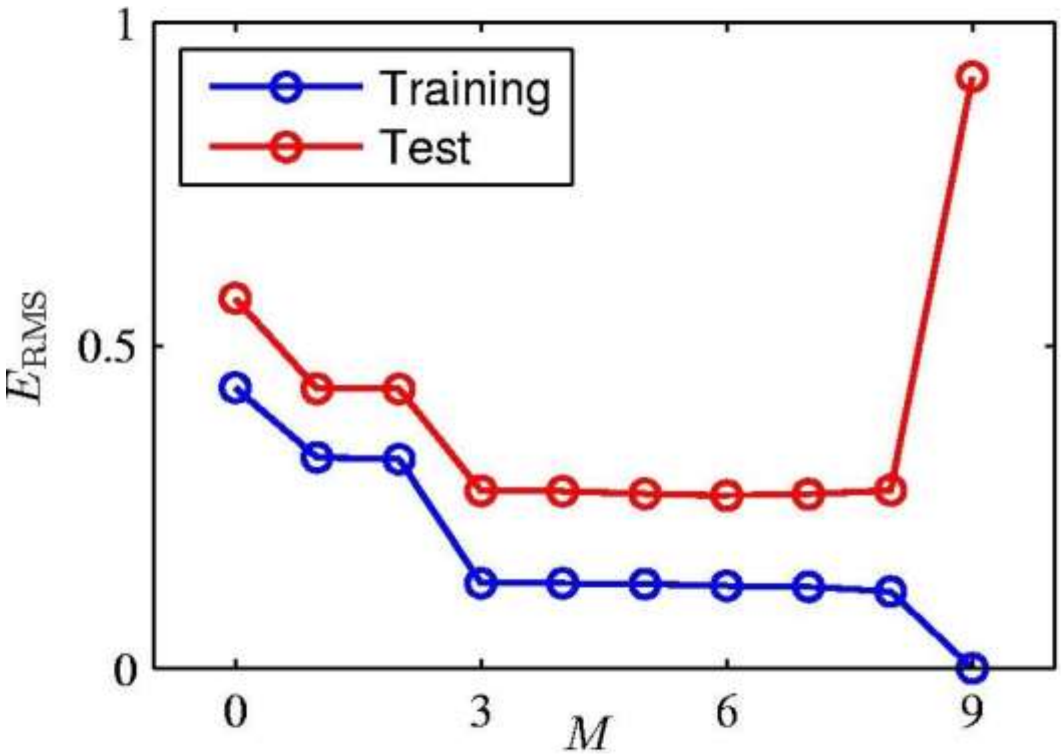
Goal: Learn y = $\mathbf{w}^T$ f($\mathbf{x}$) + b
where f(.) is a polynomial basis function

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

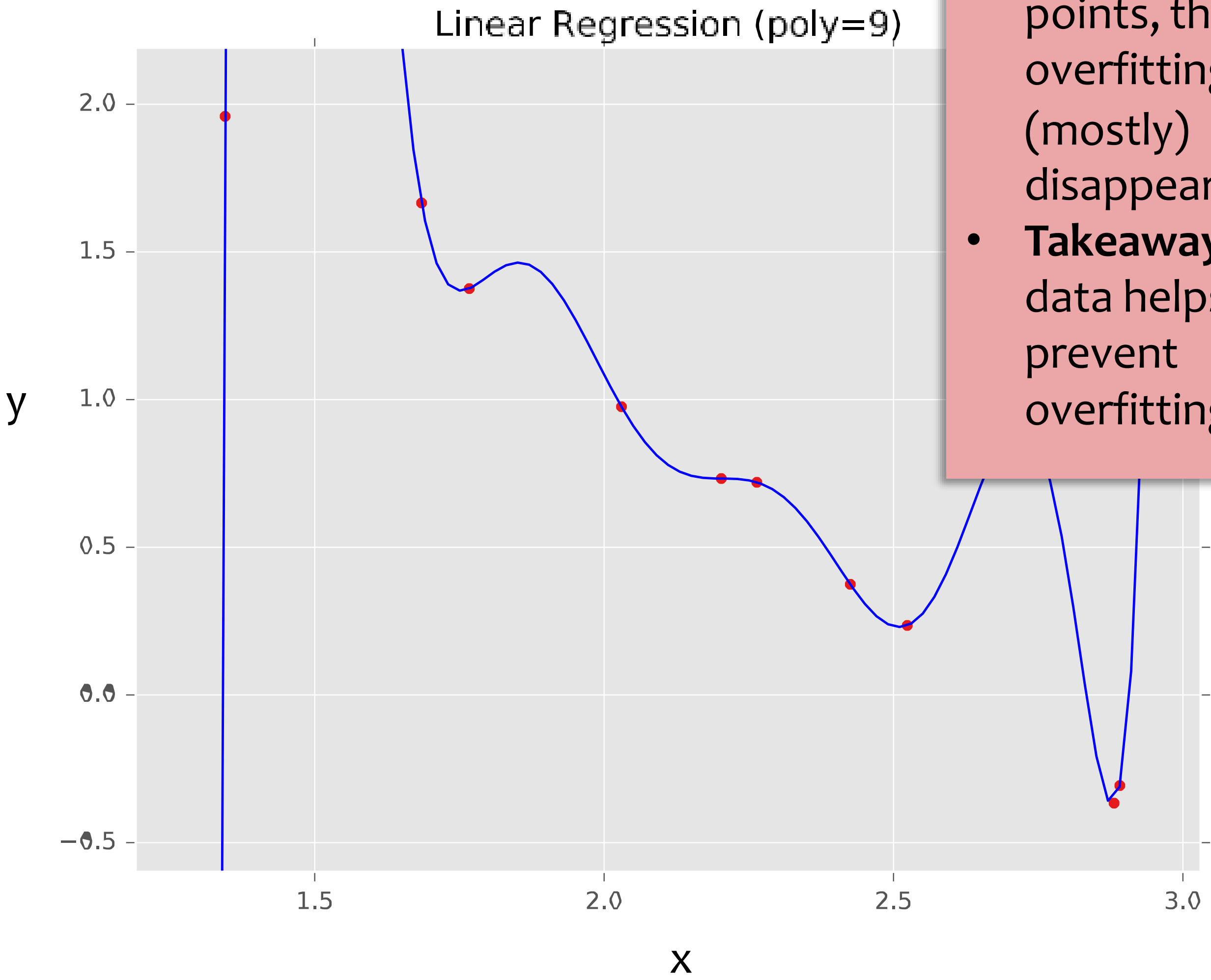
	$M = 0$	$M = 1$	$M = 3$	$M = 9$
θ_0	0.19	0.82	0.31	0.35
θ_1		-1.27	7.99	232.37
θ_2			-25.43	-5321.83
θ_3			17.37	48568.31
θ_4				-231639.30
θ_5				640042.26
θ_6				-1061800.52
θ_7				1042400.18
θ_8				-557682.99
θ_9				125201.43

Example: Linear Regression

上海科技大学
ShanghaiTech University

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	…	$(1.9)^9$

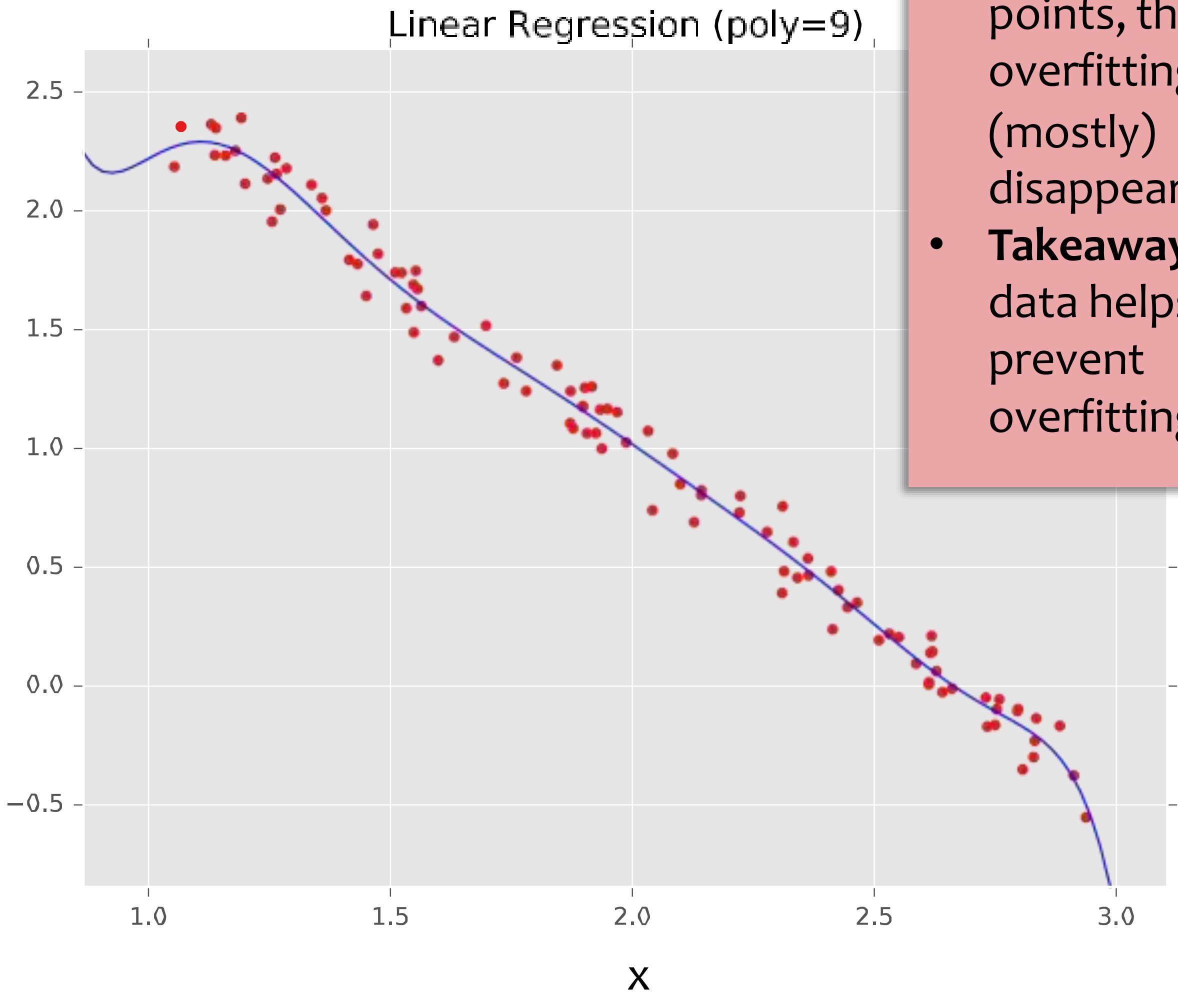


- With just N = 10 points we overfit!
- But with N = 100 points, the overfitting (mostly) disappears
- **Takeaway:** more data helps prevent overfitting

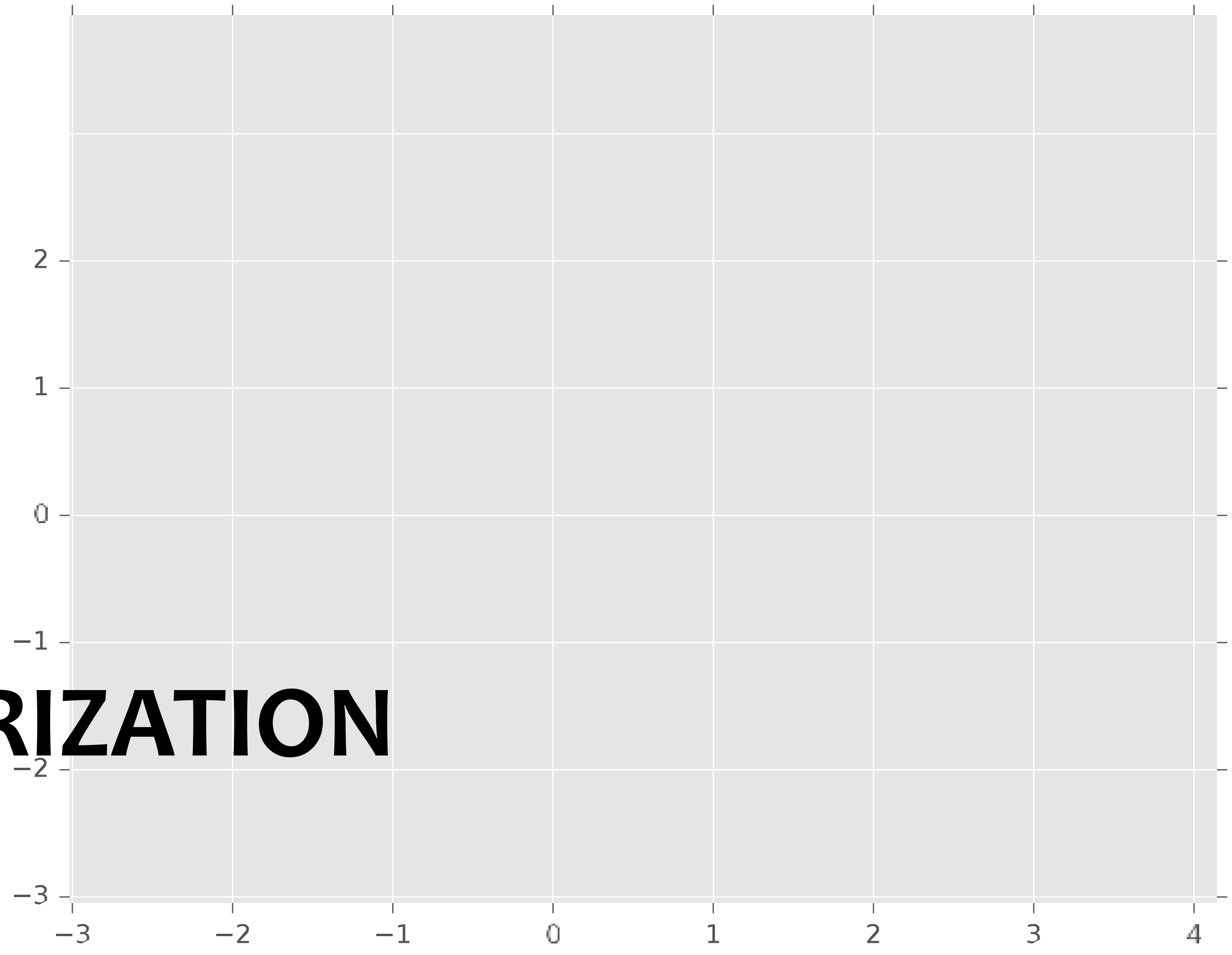
Example: Linear Regression

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

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

上海科技大学
ShanghaiTech University

- **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

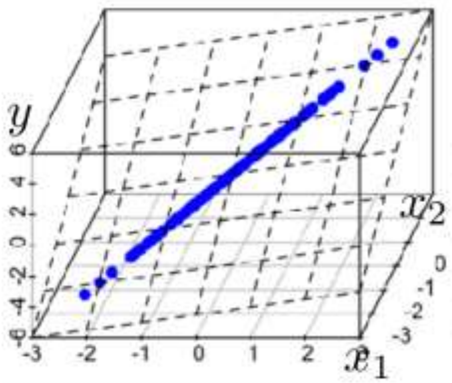
- **Given** objective function: $J(\theta)$
- **Goal** is to find: $\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta}) + \lambda r(\boldsymbol{\theta})$
- **Key idea:** Define regularizer $r(\theta)$ s.t. we tradeoff between fitting the data and keeping the model simple
- **Choose form of $r(\theta)$:**
  – Example: q-norm (usually p-norm): $\|\boldsymbol{\theta}\|_q = \left(\sum_{m=1}^{M} |\theta_m|^q\right)^{\frac{1}{q}}$

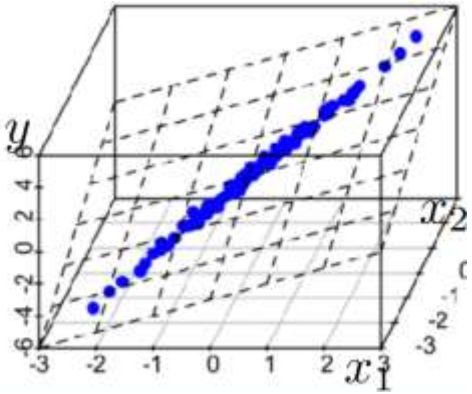
q	$r(\boldsymbol{\theta})$	yields parameters that are...	name	optimization notes
0	$\|\|\boldsymbol{\theta}\|\|_0 = \sum \mathbb{1}(\theta_m \neq 0)$	zero values	L0 reg.	no good computational solutions
1	$\|\|\boldsymbol{\theta}\|\|_1 = \sum \|\theta_m\|$	zero values	L1 reg.	subdifferentiable
2	$(\|\|\boldsymbol{\theta}\|\|_2)^2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

Regularization Examples

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

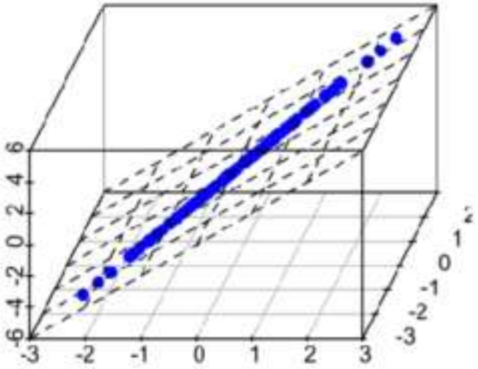
$$J_{\text{RR}}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \boxed{\lambda ||\boldsymbol{\theta}||_2^2}$$
$$= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)})^2 + \boxed{\lambda \sum_{m=1}^{M}\theta_m^2}$$





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

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



Question:

Suppose we are minimizing J'(θ) where

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

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

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

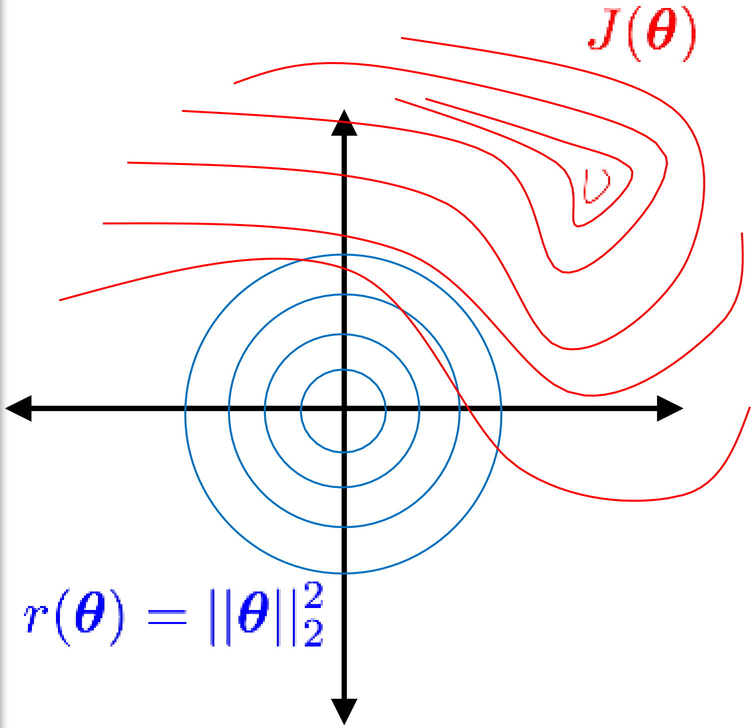
B. … move towards the minimum of J(θ)

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



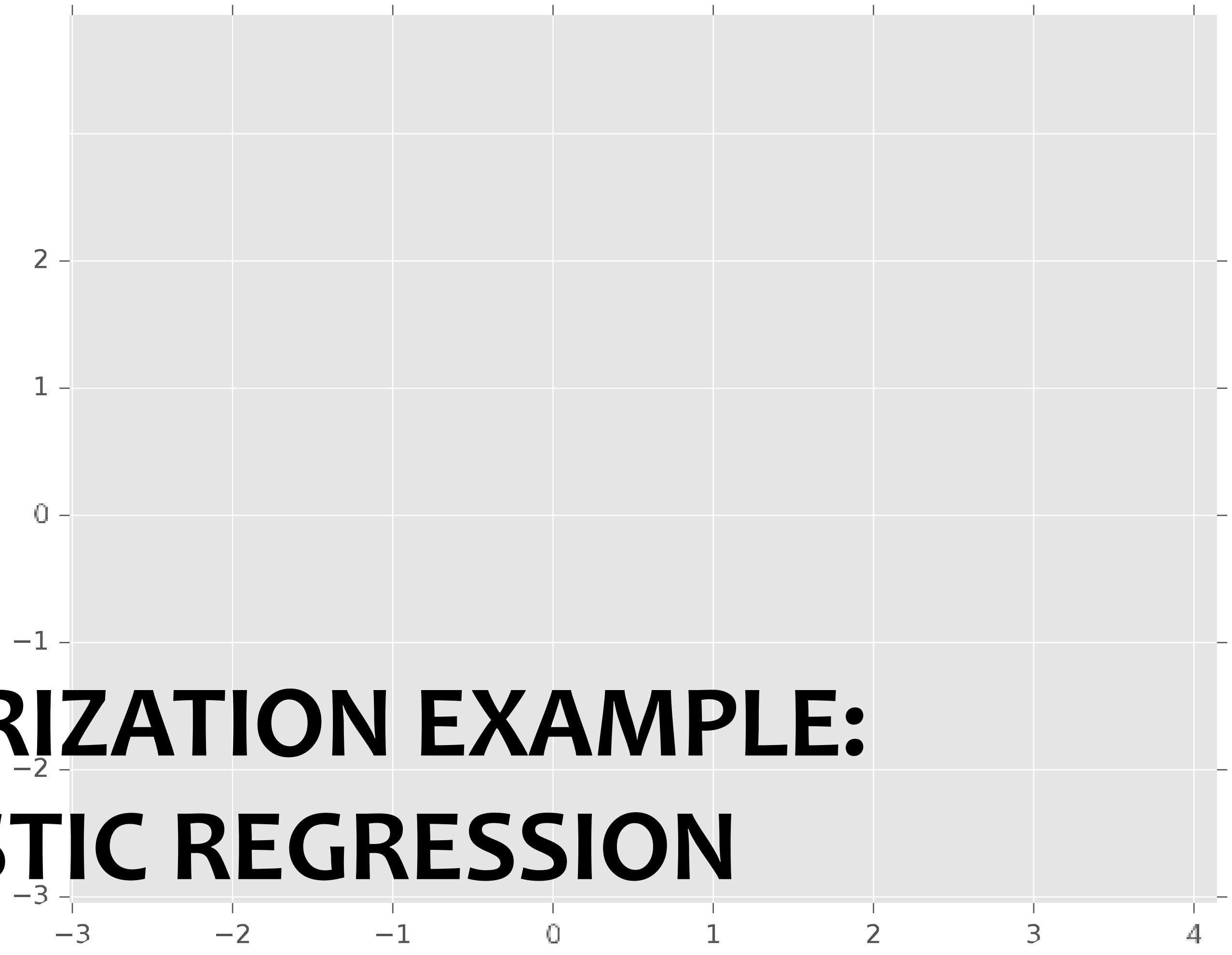
Don't Regularize the Bias (Intercept) Parameter!

- 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

Standardizing Data

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

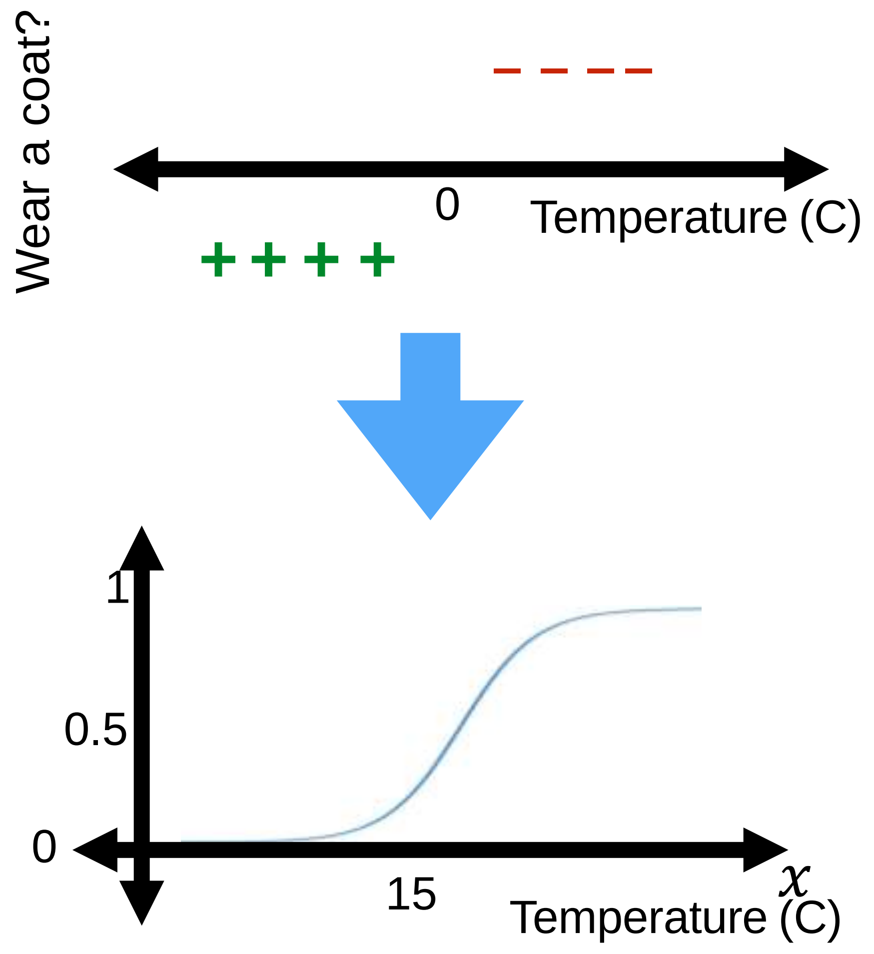
REGULARIZATION EXAMPLE: LOGISTIC REGRESSION



Gradient Descent for Logistic Regression

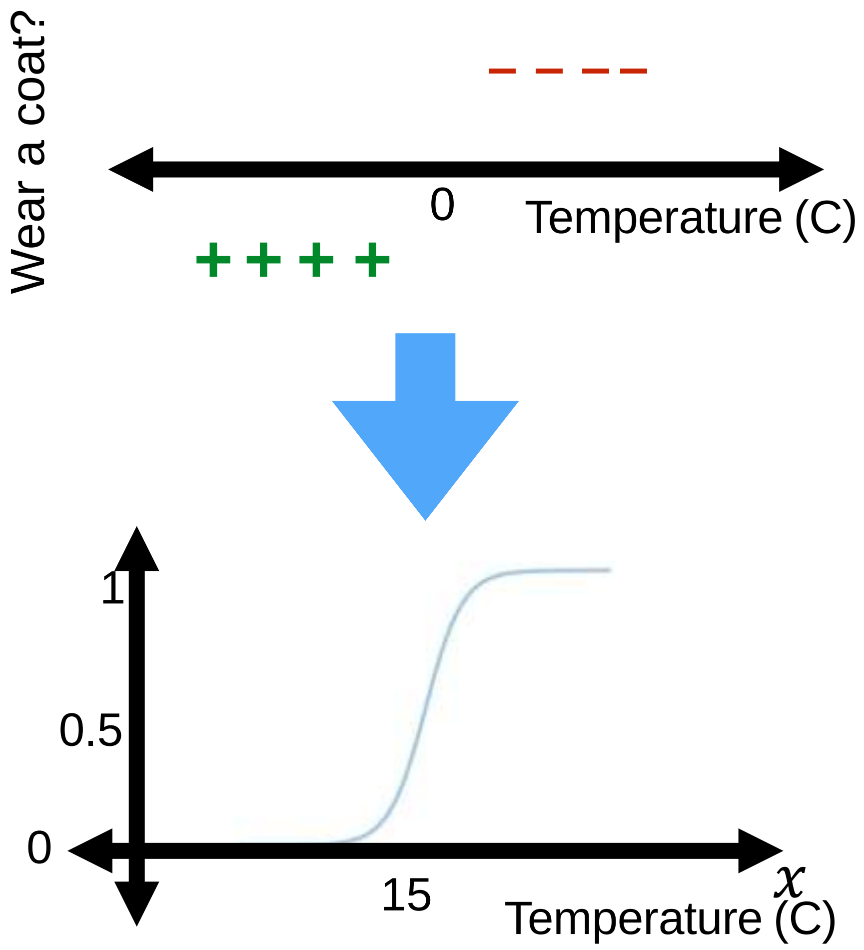
上海科技大学
ShanghaiTech University

- Loss $J(w, b)$ is differentiable and convex
- Run Gradient Descent



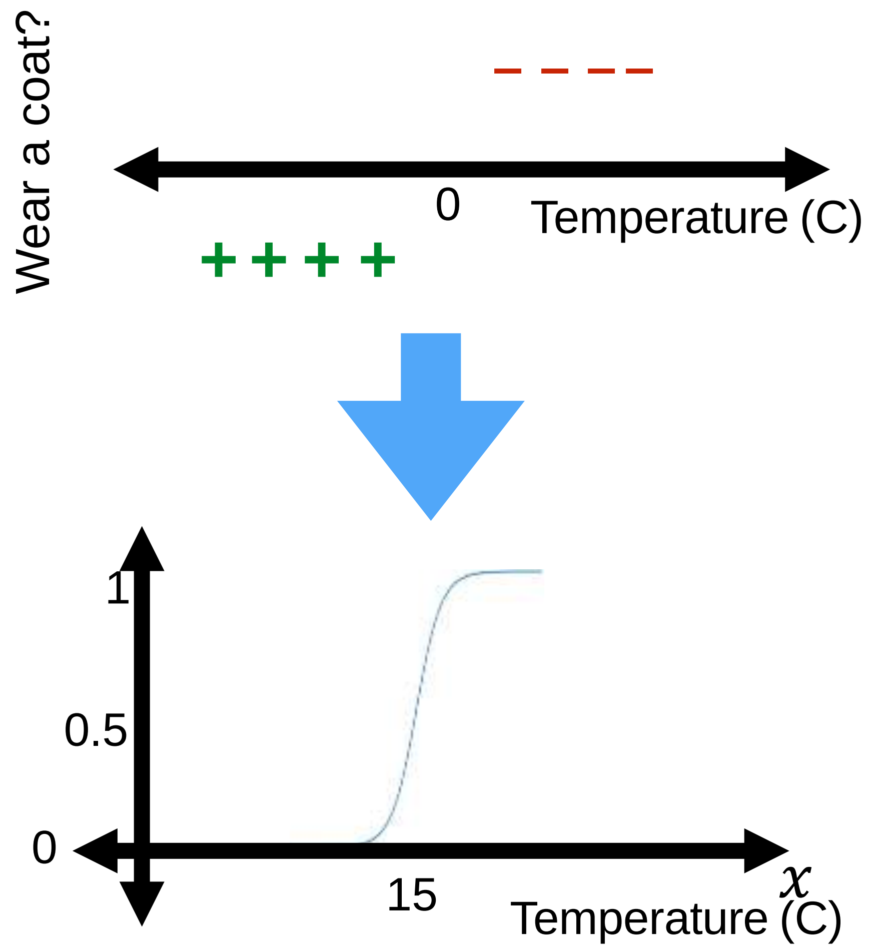
Gradient Descent for Logistic Regression

- Loss $J(w, b)$ is differentiable and convex
- Run Gradient Descent



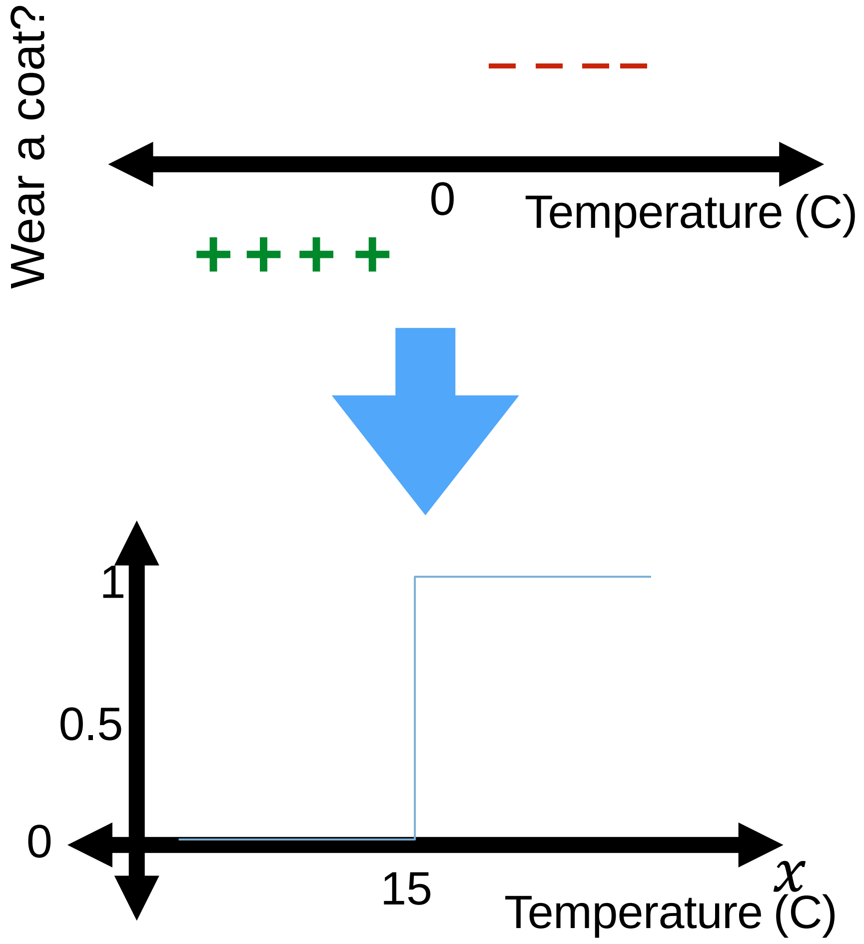
Gradient Descent for Logistic Regression

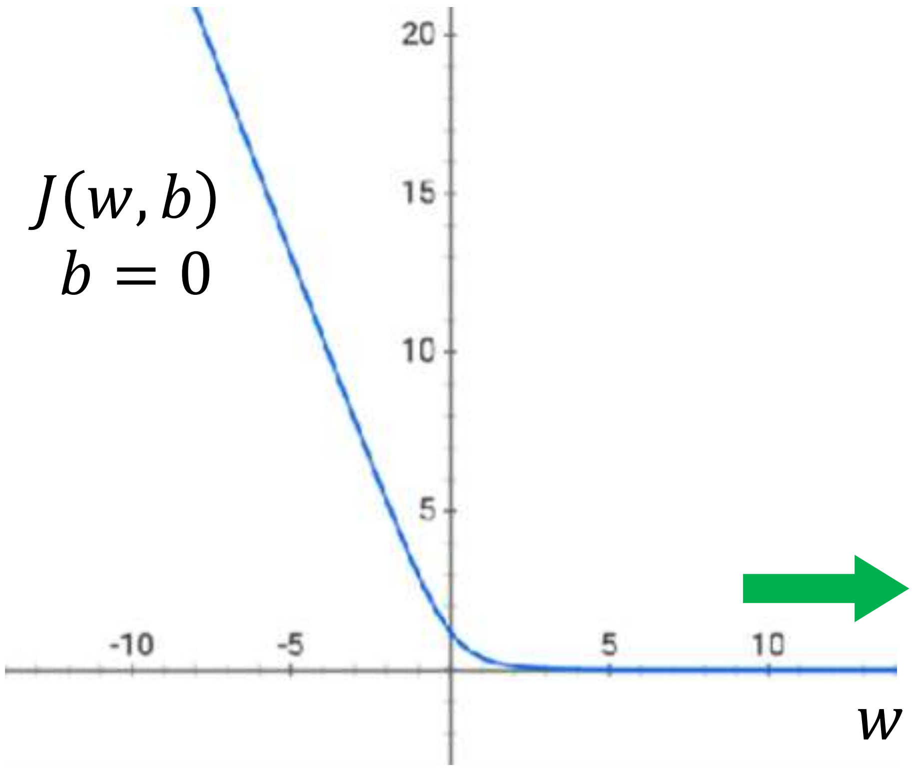
- Loss $J(w, b)$ is differentiable and convex
- Run Gradient Descent



Gradient Descent for Logistic Regression

- Loss $J(w, b)$ is differentiable and convex
- Run Gradient Descent

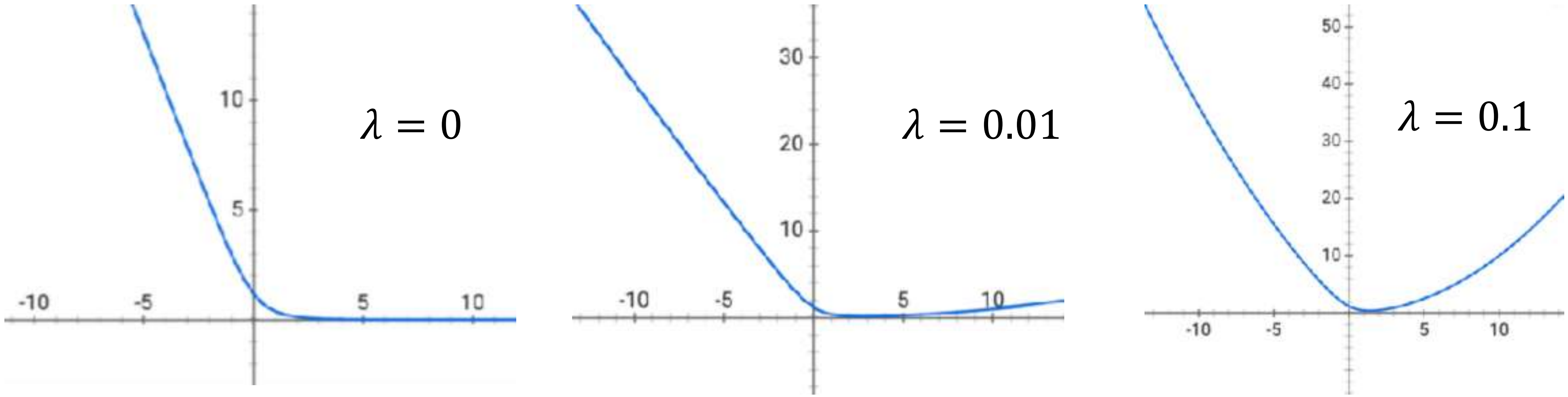




Logistic Regression Loss Revisited

$$J(w,b) = \frac{1}{n}\sum_{i=1}^{n} L_{nll}\left(\sigma(w^T x^{(i)} + b), y^{(i)}\right) + \lambda\|w\|^2$$

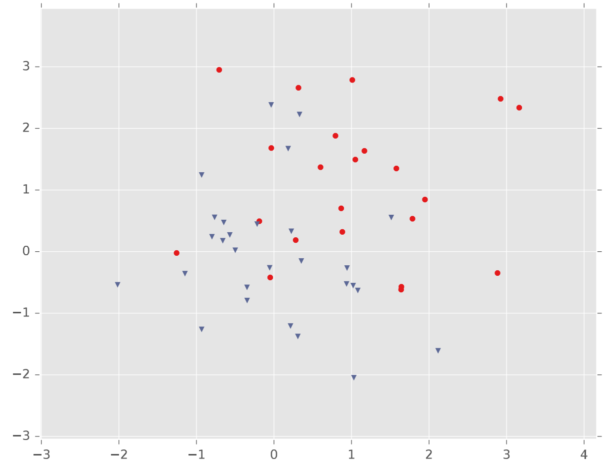
- A “regularizer” or “penalty” $\lambda\|w\|^2$
- Penalizes being overly certain
- Objective is still differentiable & convex (gradient descent)



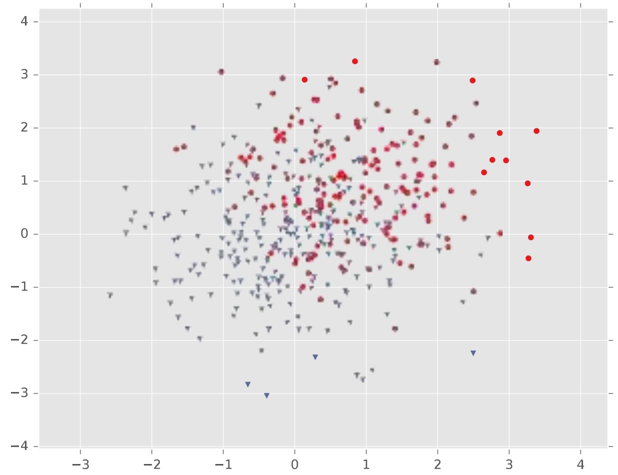
- How to choose hyperparameters? One option: consider a handful of possible values and compare via cross validation.

Example: Logistic Regression

Training Data

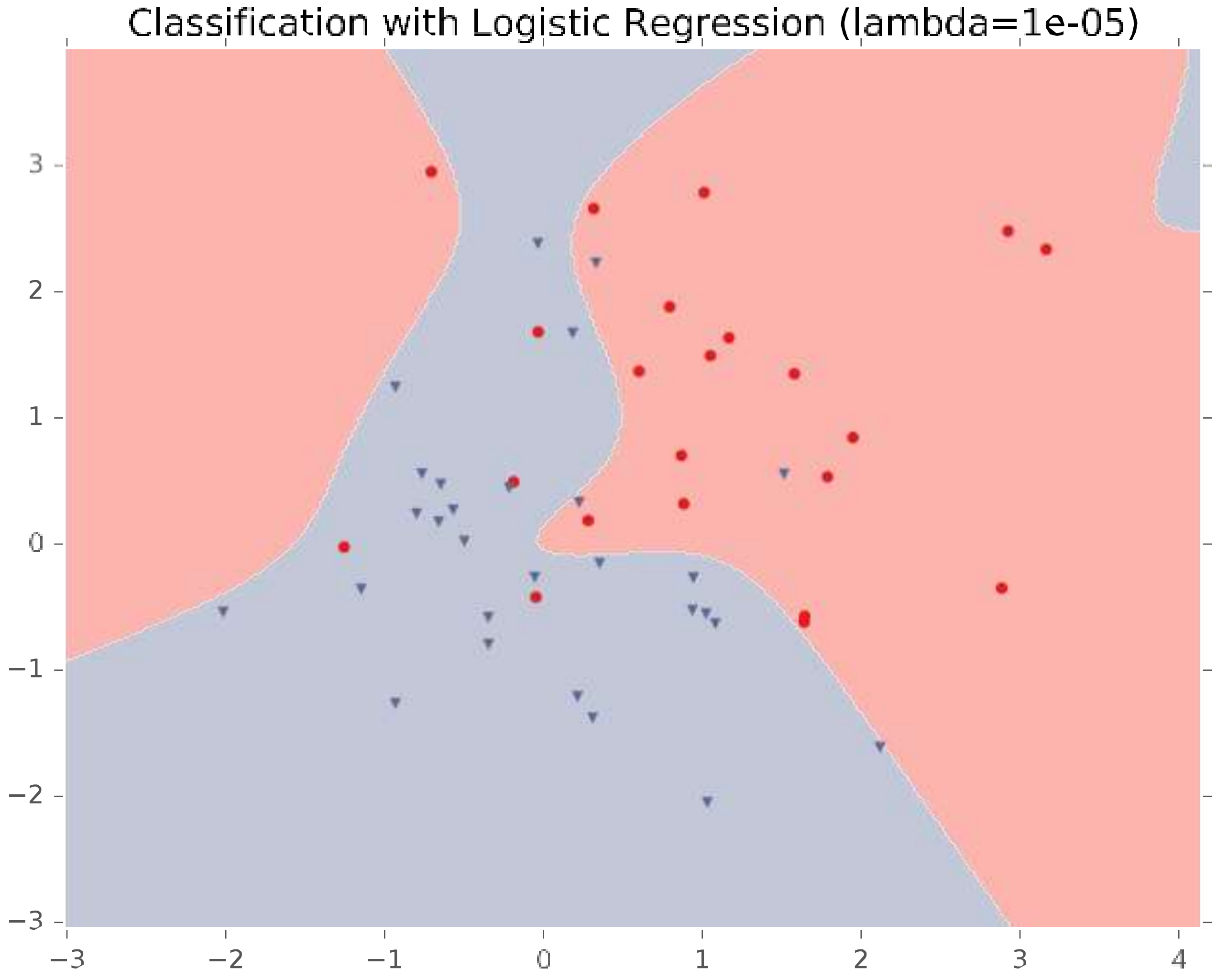


Test Data

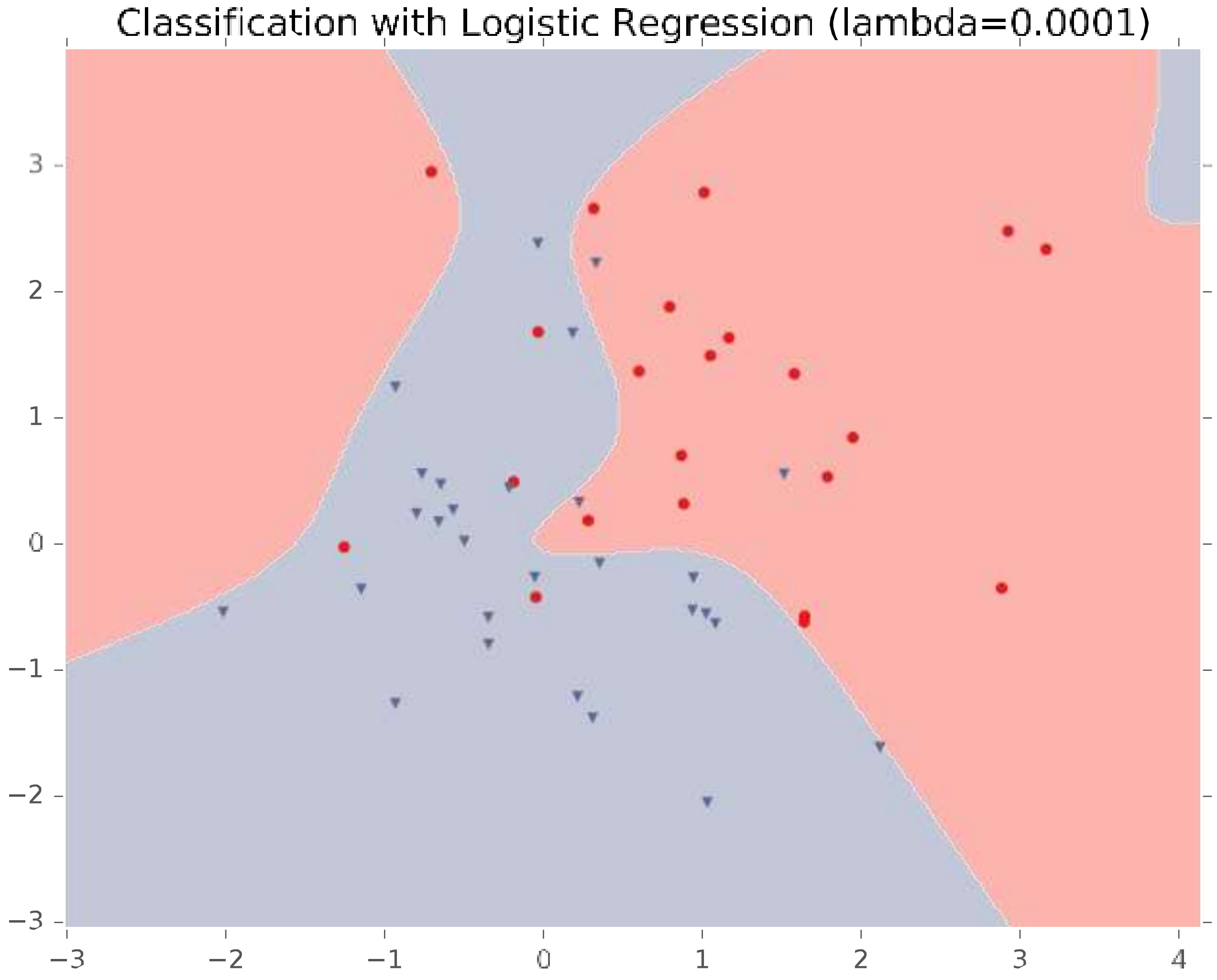


- For this example, we construct **nonlinear features** (i.e. feature engineering)
- Specifically, we add **polynomials up to order 9** of the two original features x_1 and x_2
- Thus our classifier is **linear** in the **high-dimensional feature space**, but the decision boundary is **nonlinear** when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

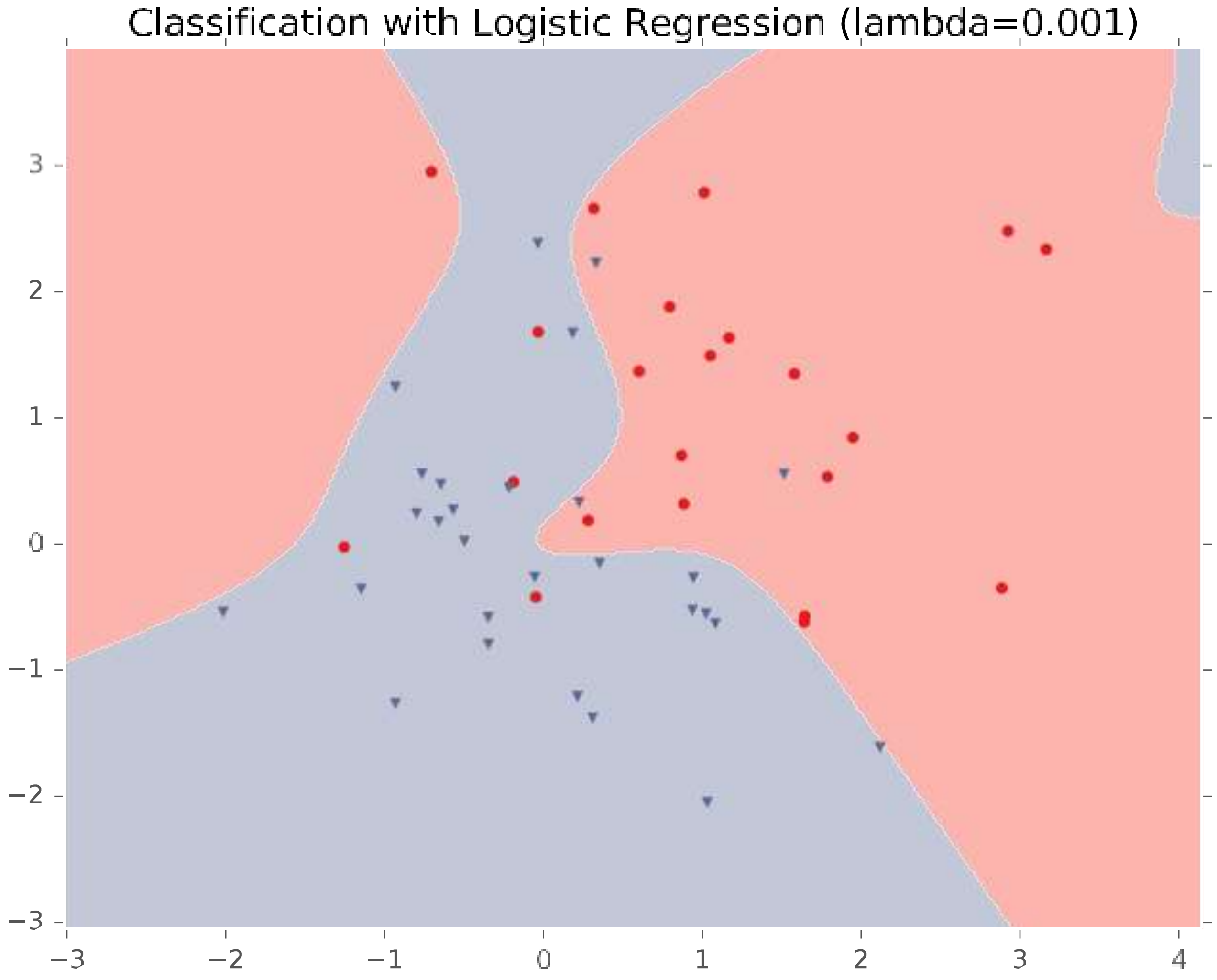
Example: Logistic Regression



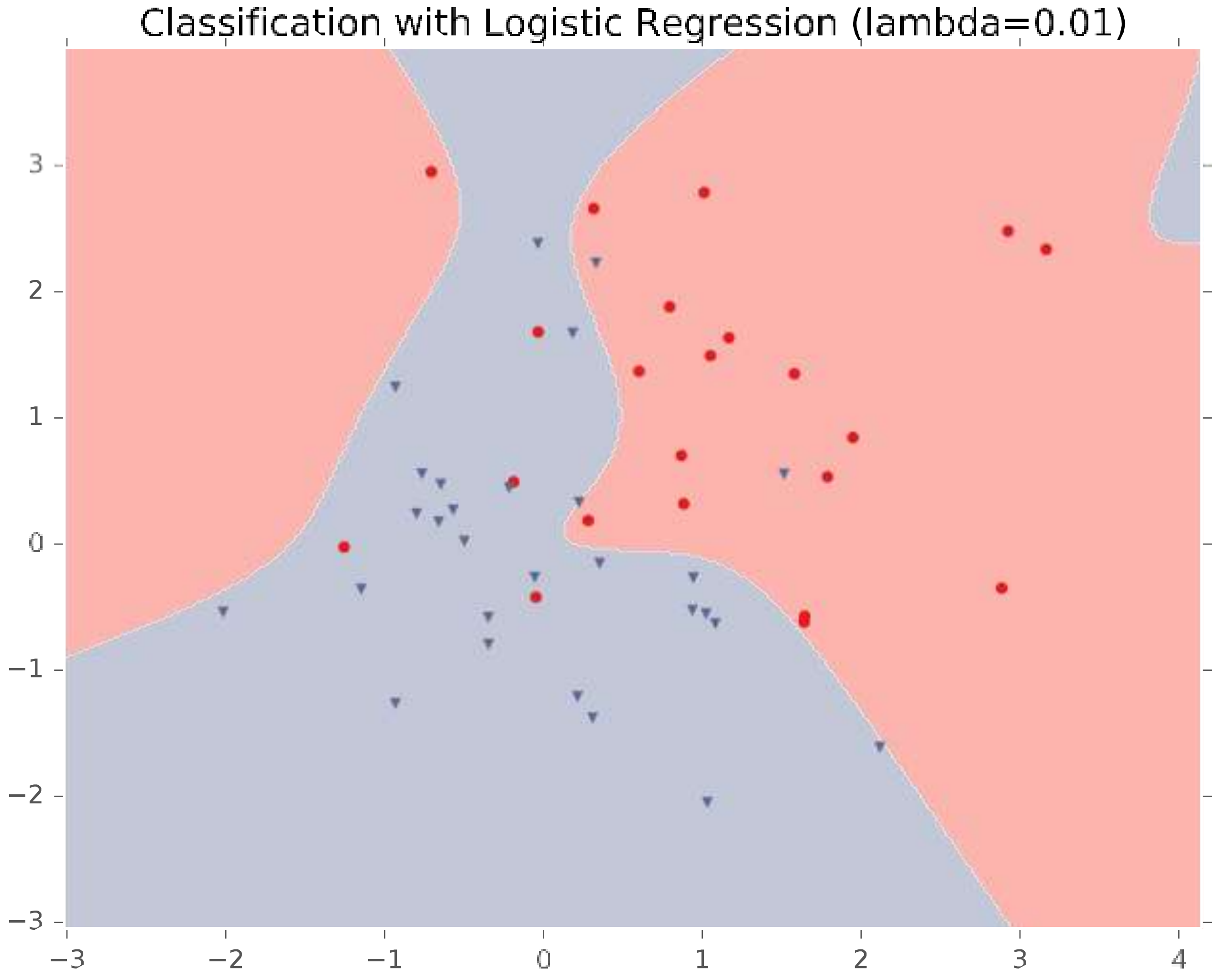
Example: Logistic Regression



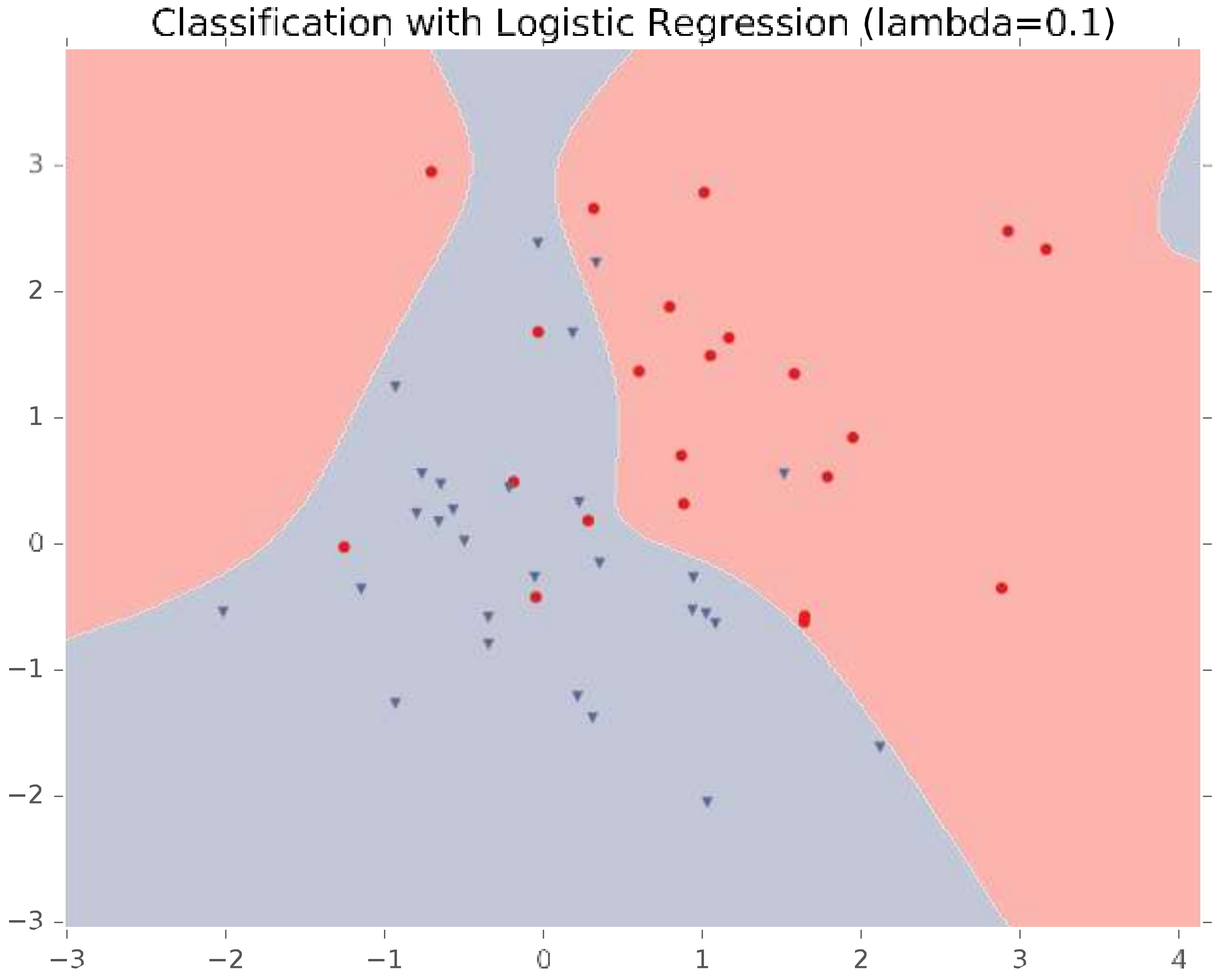
Example: Logistic Regression



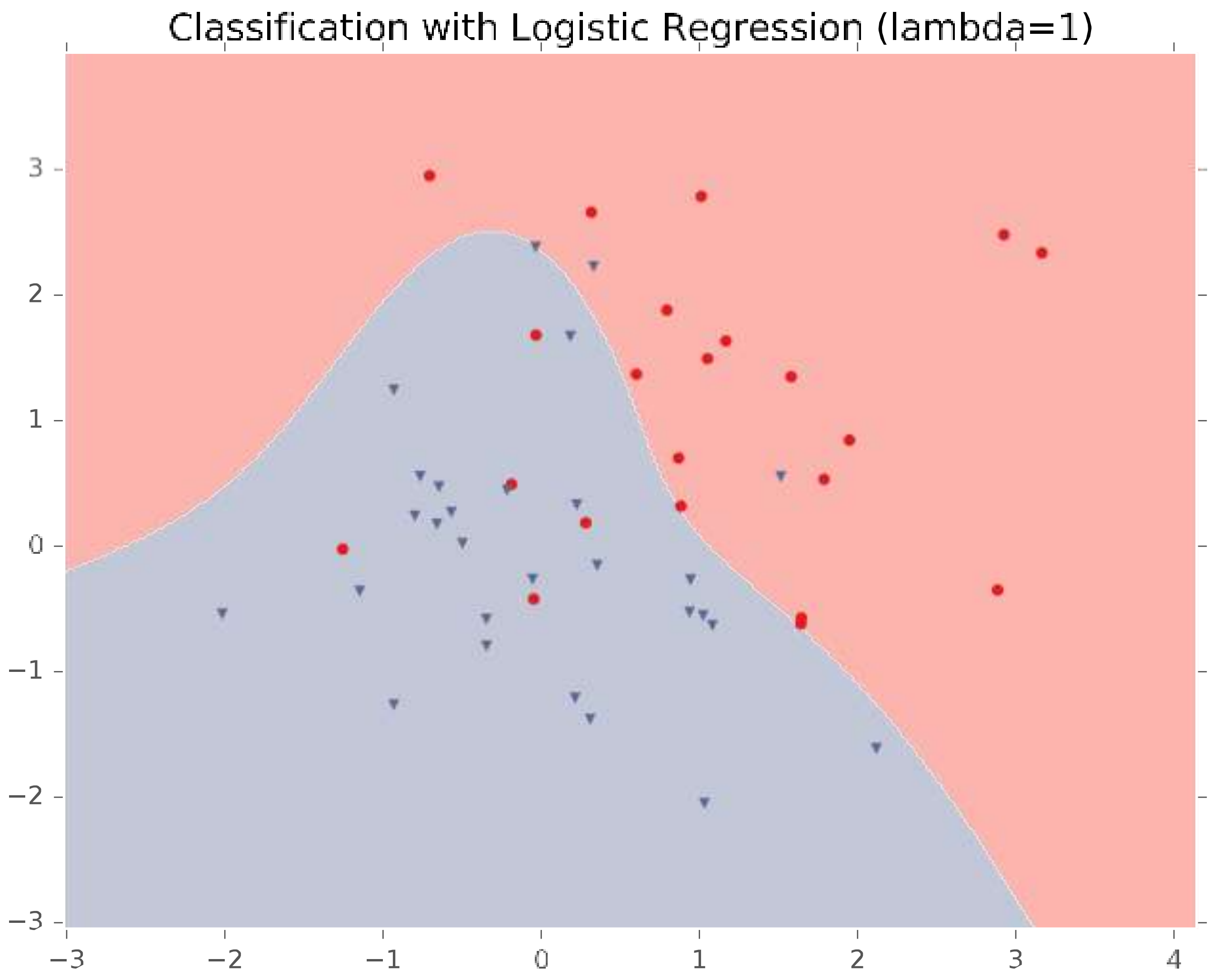
Example: Logistic Regression



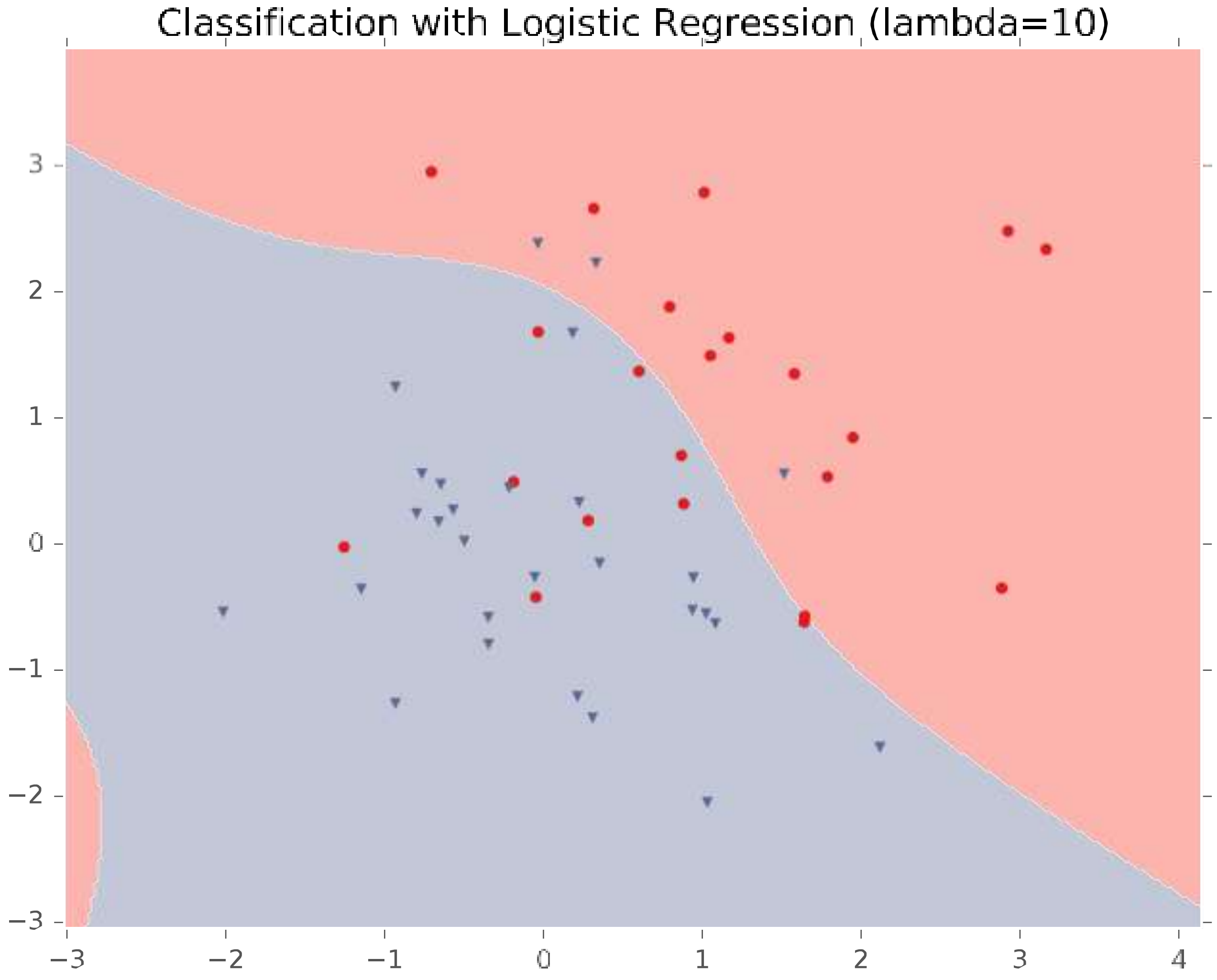
Example: Logistic Regression



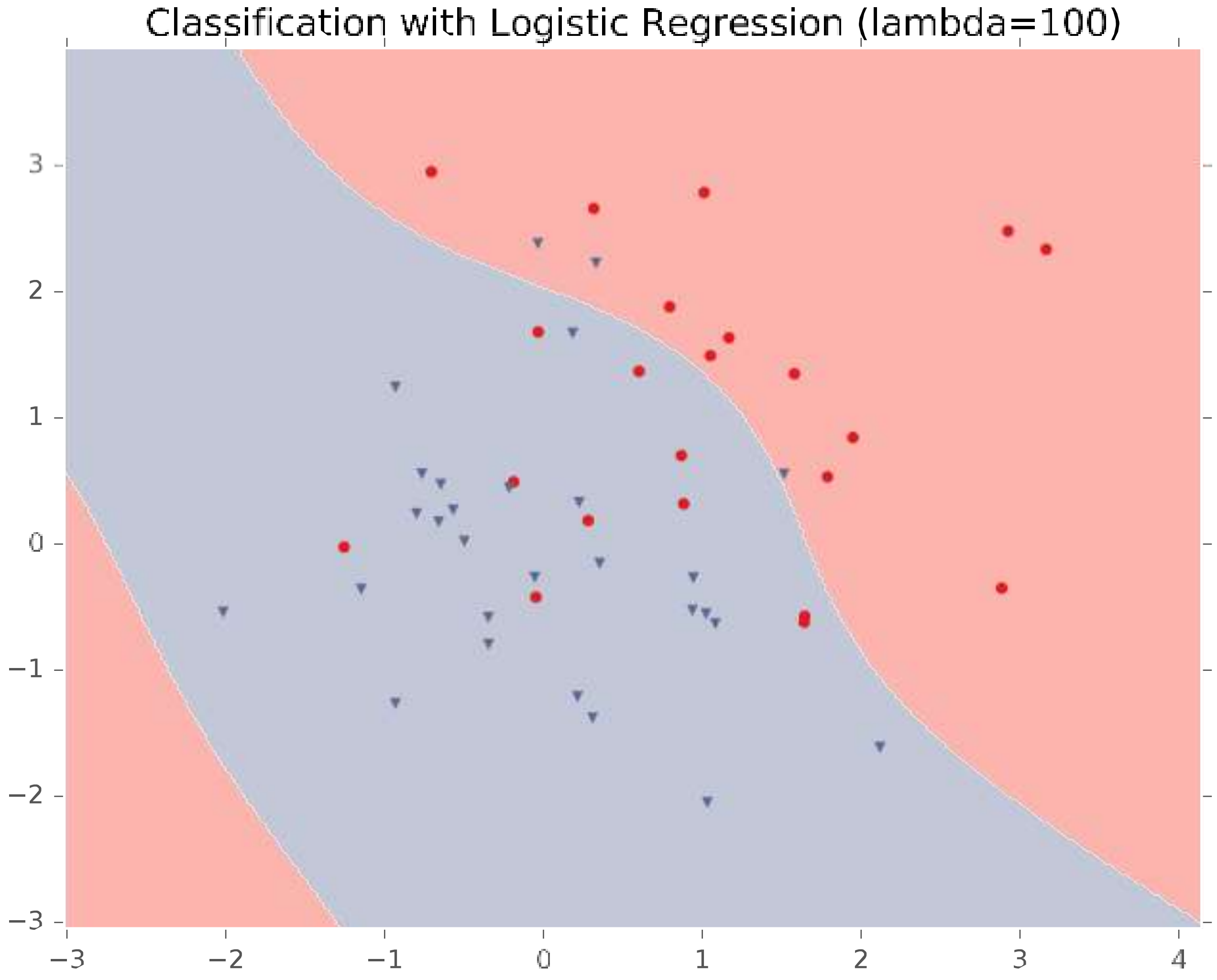
Example: Logistic Regression



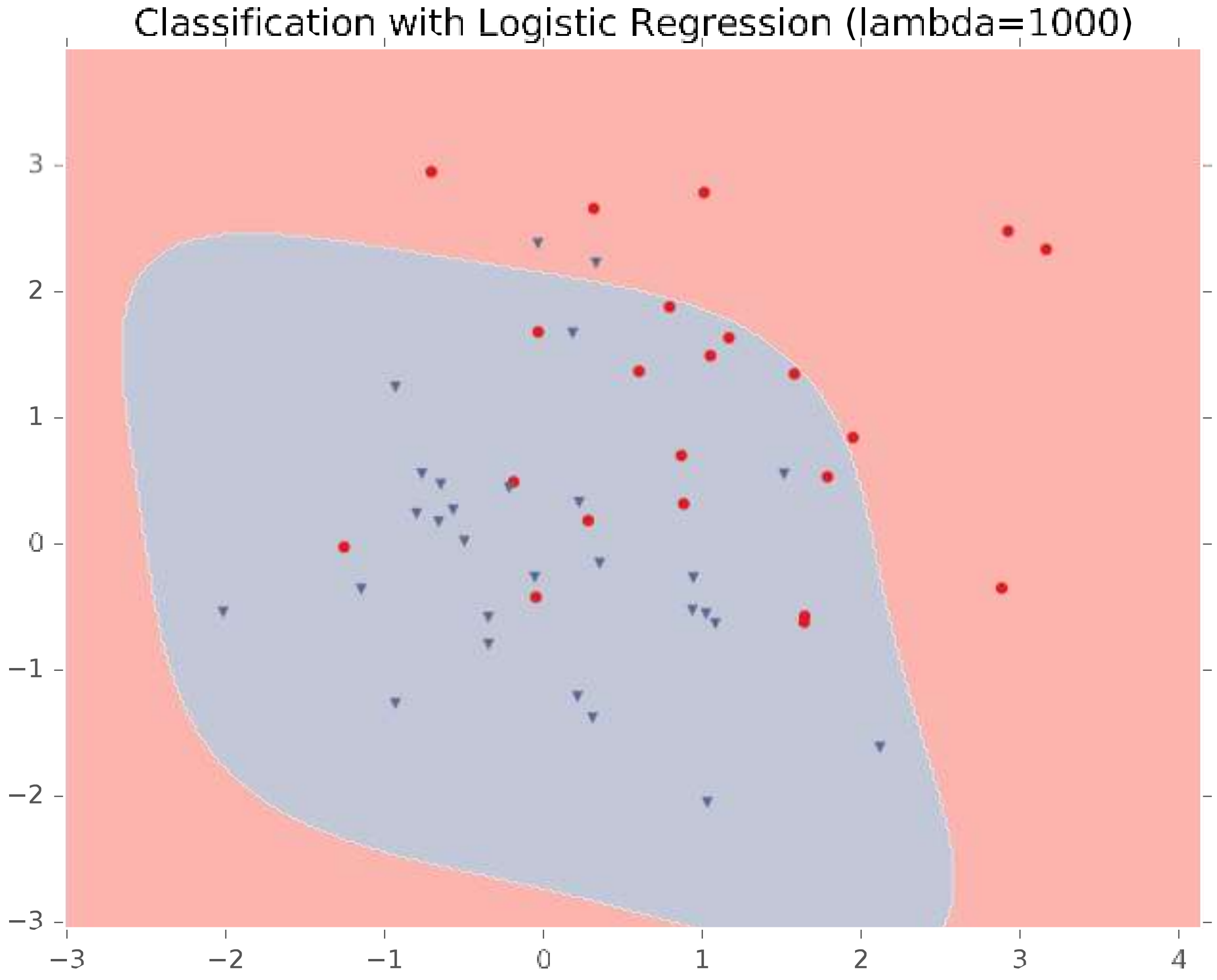
Example: Logistic Regression



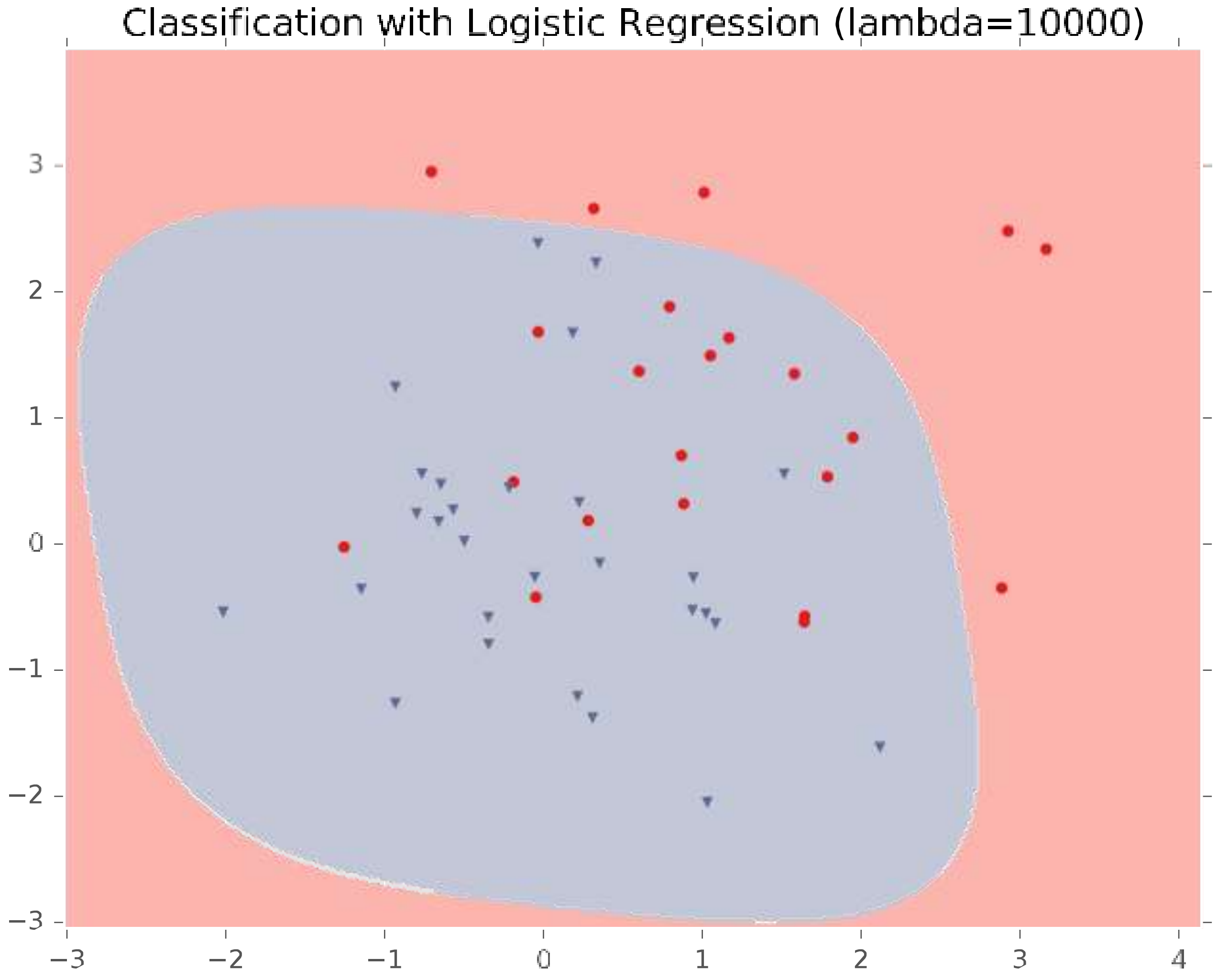
Example: Logistic Regression



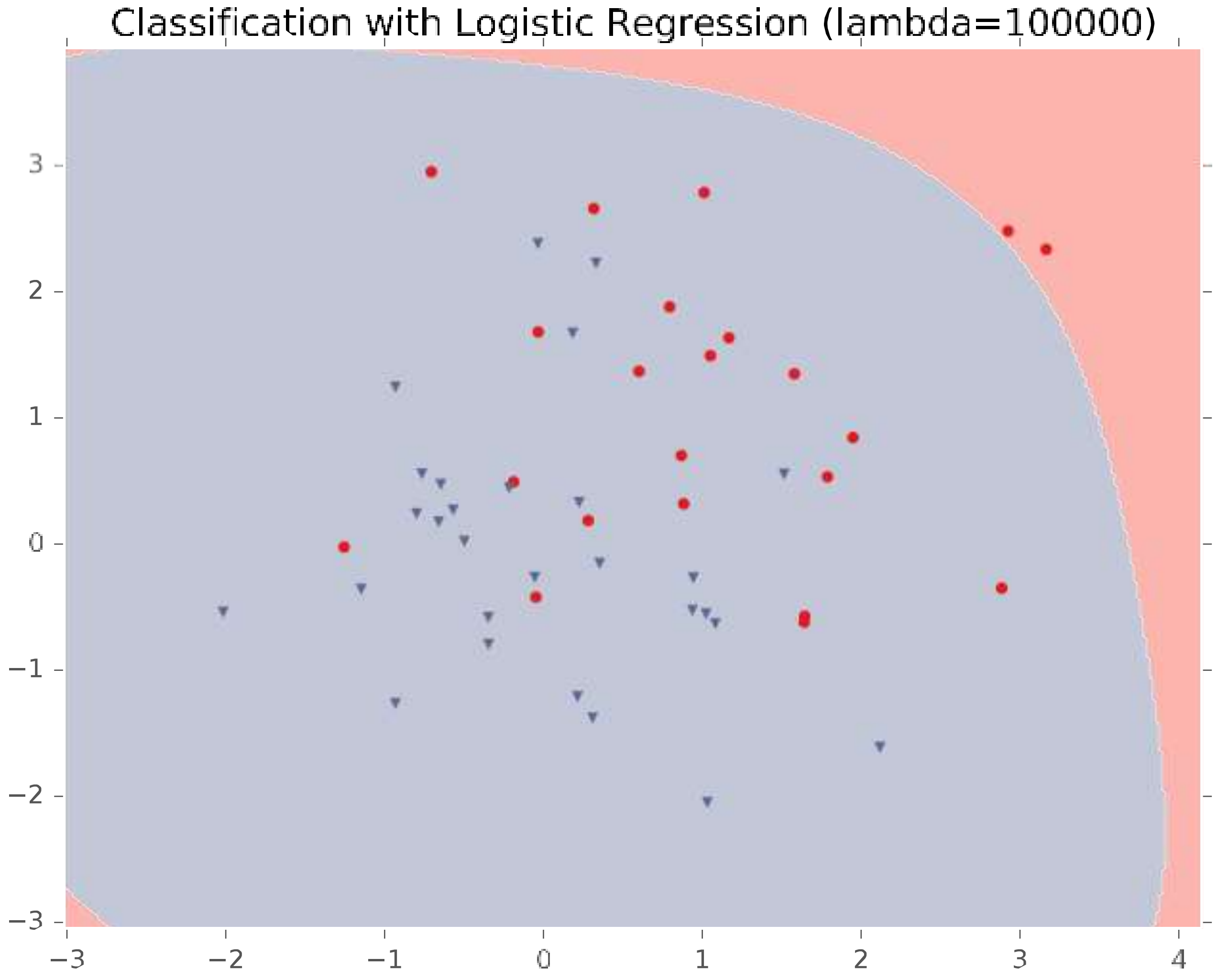
Example: Logistic Regression



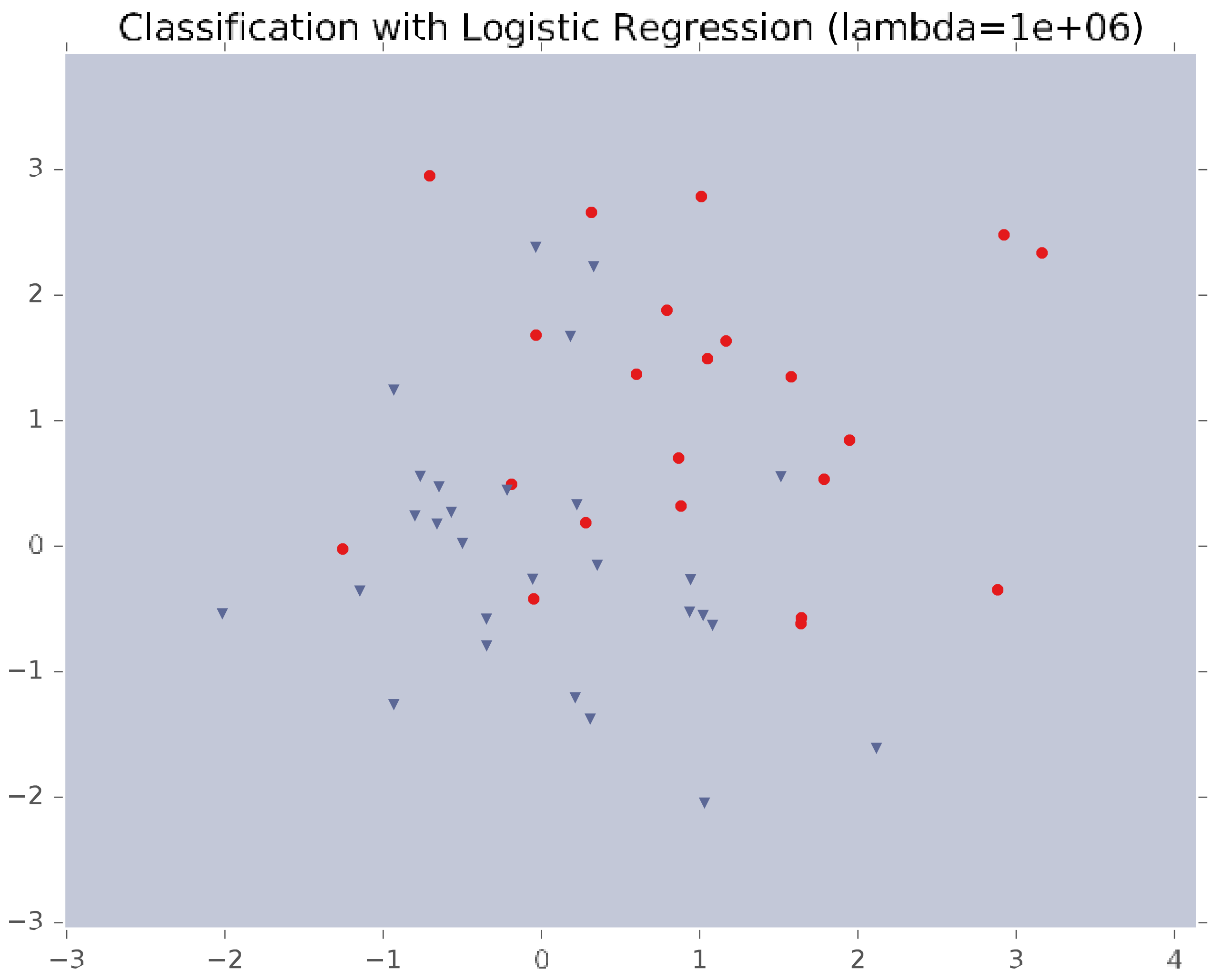
Example: Logistic Regression



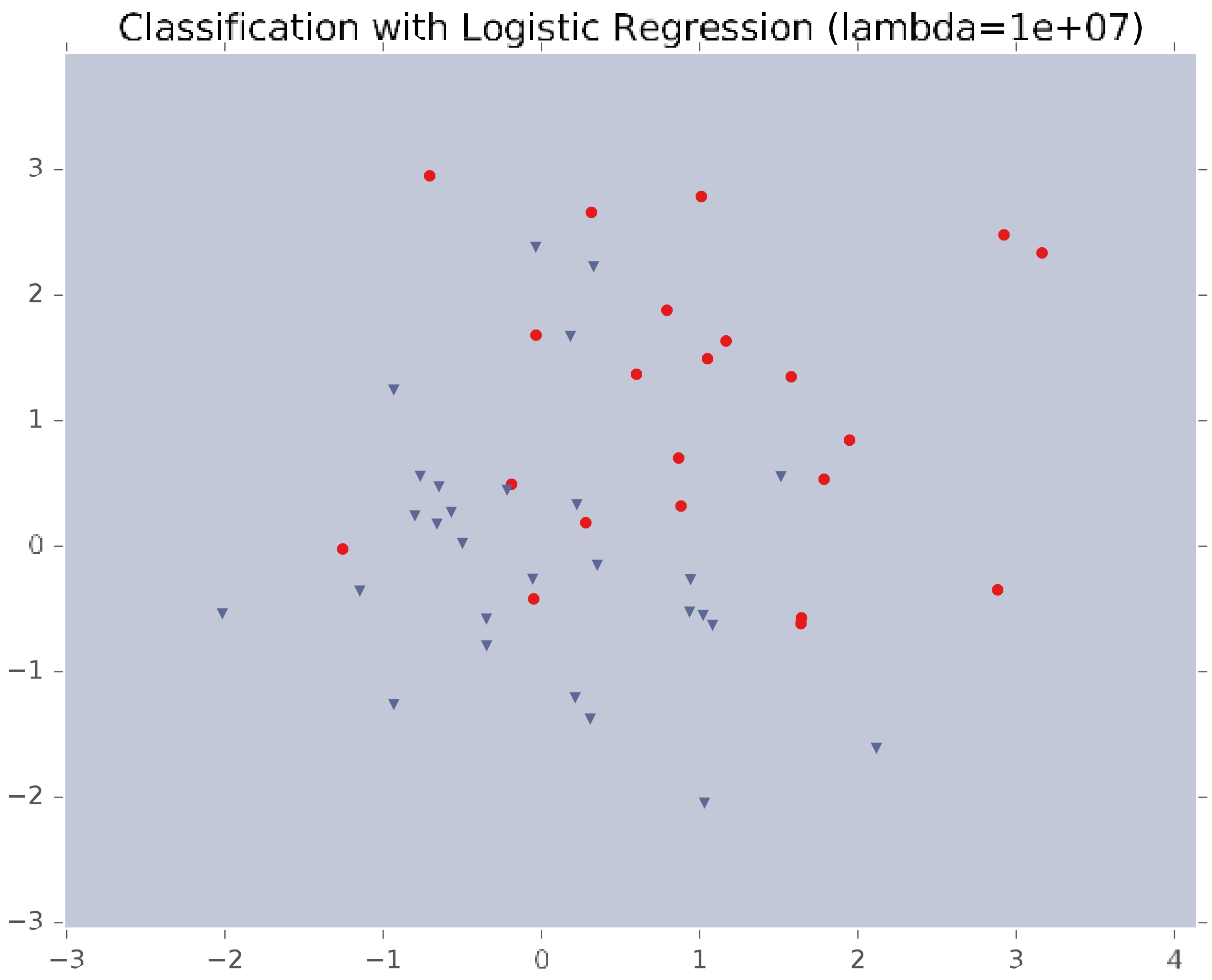
Example: Logistic Regression



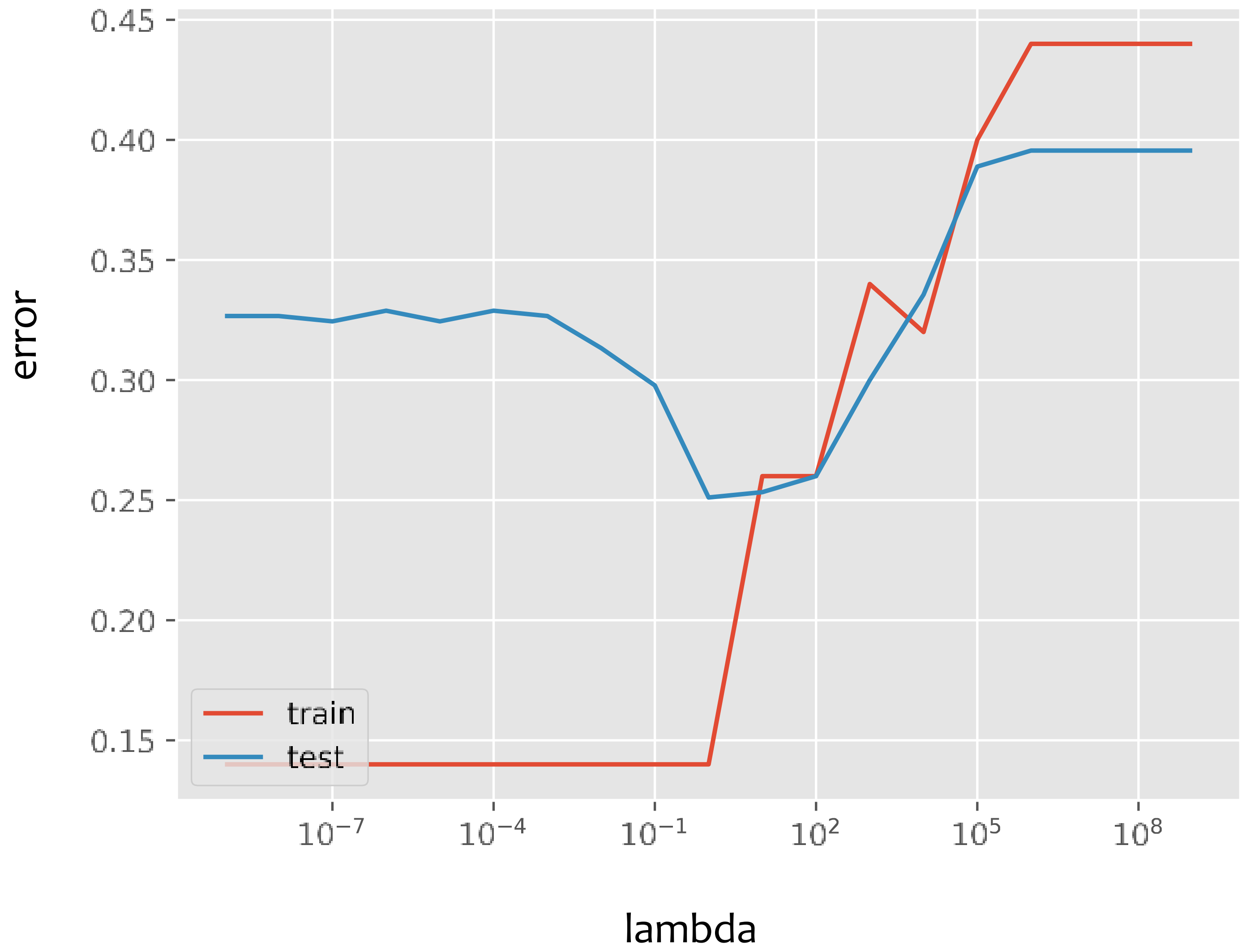
Example: Logistic Regression

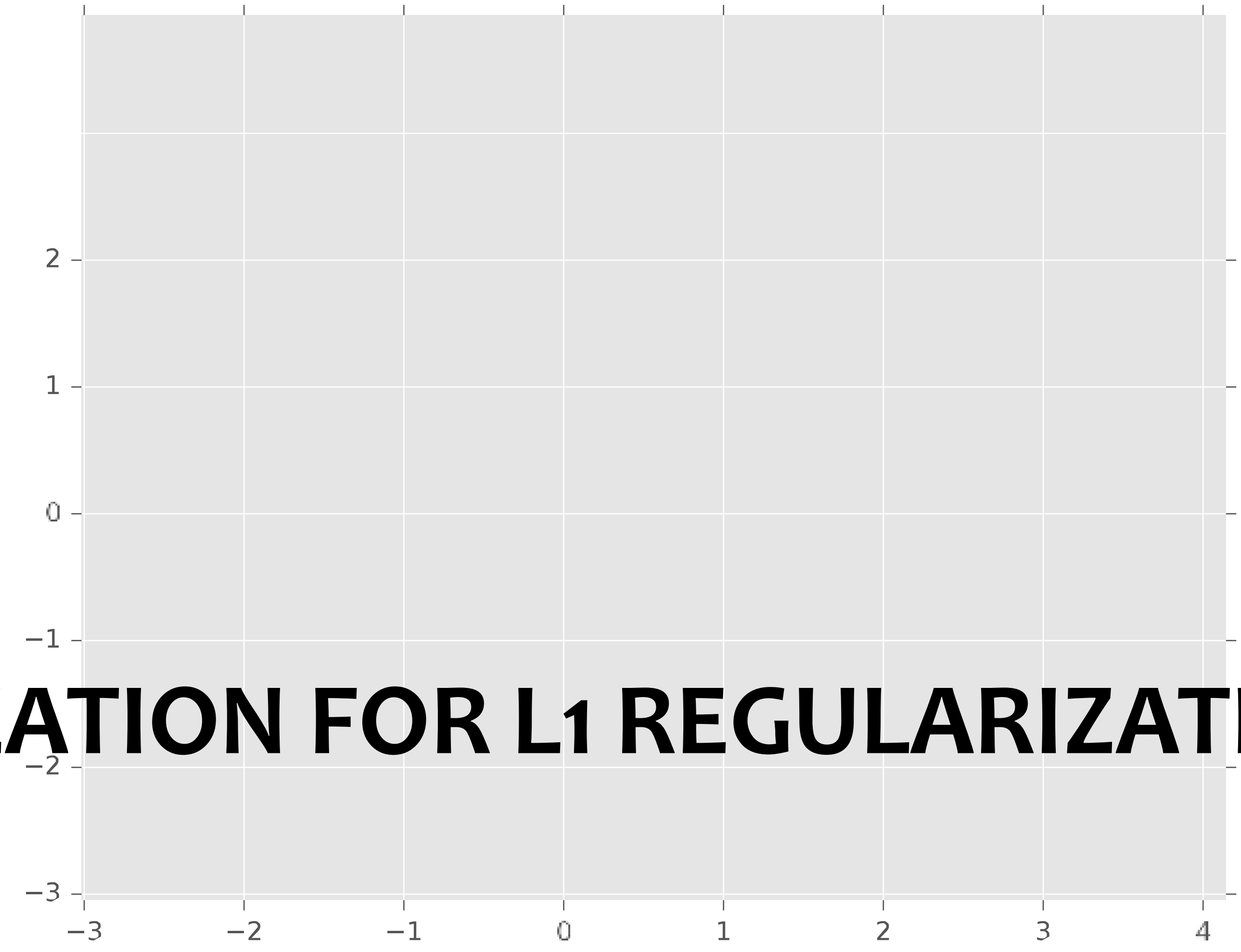


Example: Logistic Regression



Example: Logistic Regression





OPTIMIZATION FOR L1 REGULARIZATION

Can we apply SGD to the LASSO learning problem?

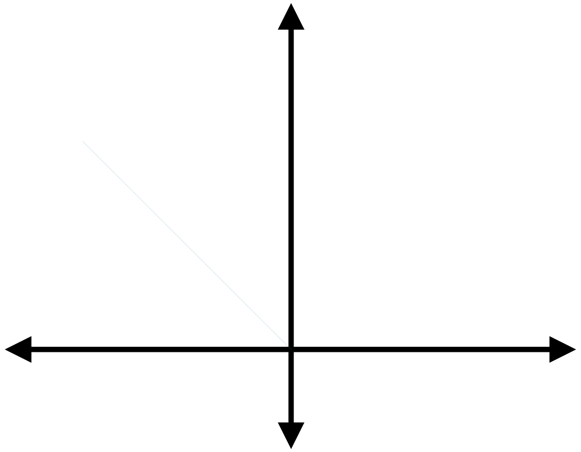
$$\underset{\boldsymbol{\theta}}{\text{argmin}} \quad J_{\text{LASSO}}(\boldsymbol{\theta})$$

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

Optimization for L1 Regularization

- Consider the absolute value function:

$$r(\boldsymbol{\theta}) = \lambda \sum_{k=1}^{K} |\theta_k|$$



- The L1 penalty is subdifferentiable (i.e. not differentiable at 0)

Def: A vector $g \in \mathbb{R}^M$ is called a **subgradient** of a function $f(\mathbf{x}) : \mathbb{R}^M \to \mathbb{R}$ at the point $\mathbf{x}$ if, for all $\mathbf{x}' \in \mathbb{R}^M$, we have:

$$f(\mathbf{x}') \geq f(\mathbf{x}) + \mathbf{g}^T(\mathbf{x}' - \mathbf{x})$$

Optimization for L1 Regularization

- The L1 penalty is subdifferentiable (i.e. not differentiable at 0)
- An array of optimization algorithms exist to handle this issue:
  - Subgradient descent
  - Stochastic subgradient descent
  - Coordinate Descent
  - Othant-Wise Limited memory Quasi-Newton (OWL-QN) (Andrew & Gao, 2007) and provably convergent variants
  - Block coordinate Descent (Tseng & Yun, 2009)
  - Sparse Reconstruction by Separable Approximation (SpaRSA) (Wright et al., 2009)
  - Fast Iterative Shrinkage Thresholding Algorithm (FISTA) (Beck & Teboulle, 2009)

Basically the same as GD and SGD, but you use one of the subgradients when necessary

Takeaways

上海科技大学 ShanghaiTech University

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

上海科技大学 ShanghaiTech University

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