Announcements

Assignments

- HW4
  - Wed, 10/14, 11:59 pm

Survey

- Thanks!
- We’ll talk more Wednesday

Plan

Last Time

- Feature engineering
- Regularization with added penalty term

Today

- Wrap-up regularization
- Neural Networks
  - Perceptron
  - Multilayer perceptron
  - Building blocks
  - Objective
  - Optimization

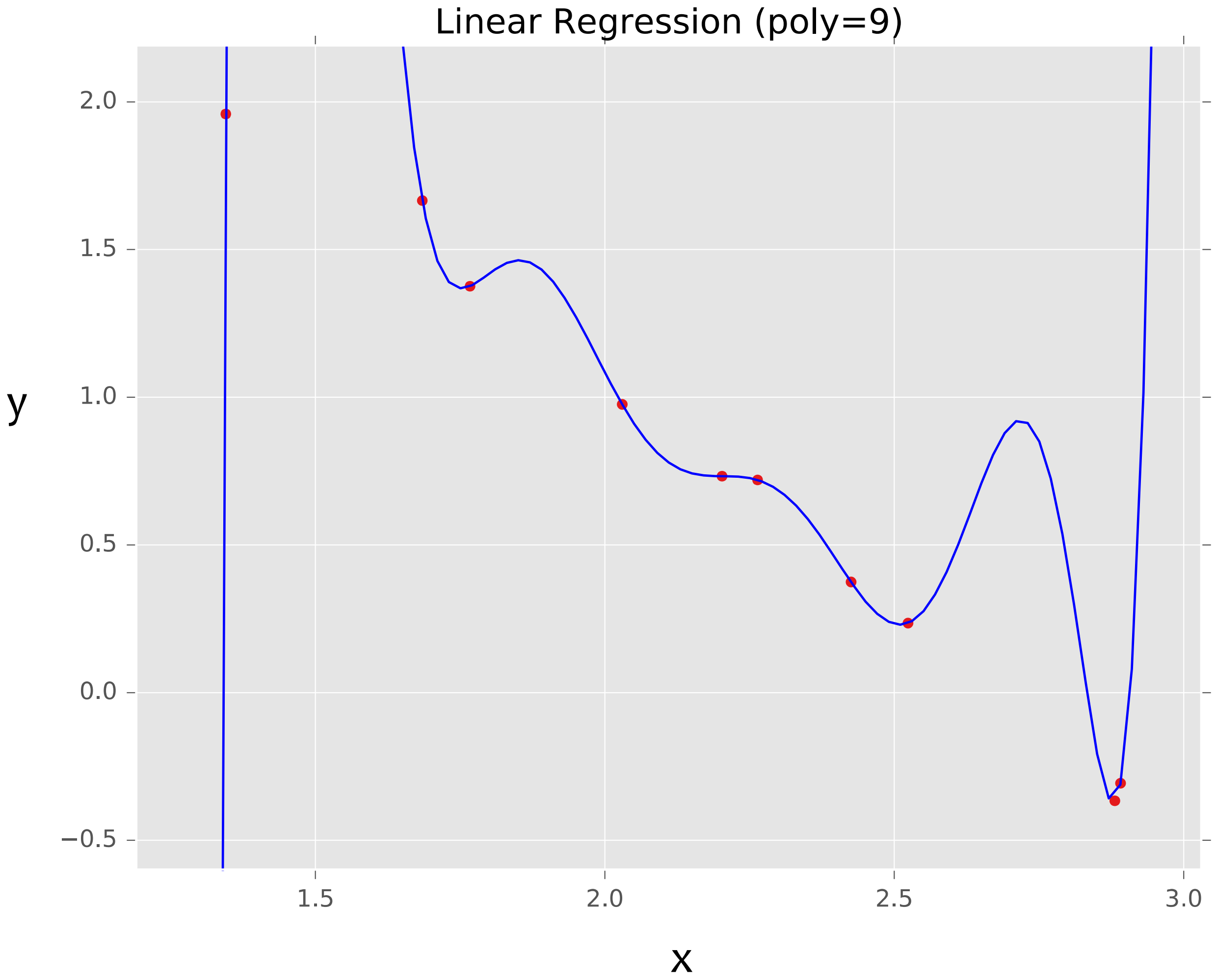
Wrap-up Regularization

Example: Linear Regression

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

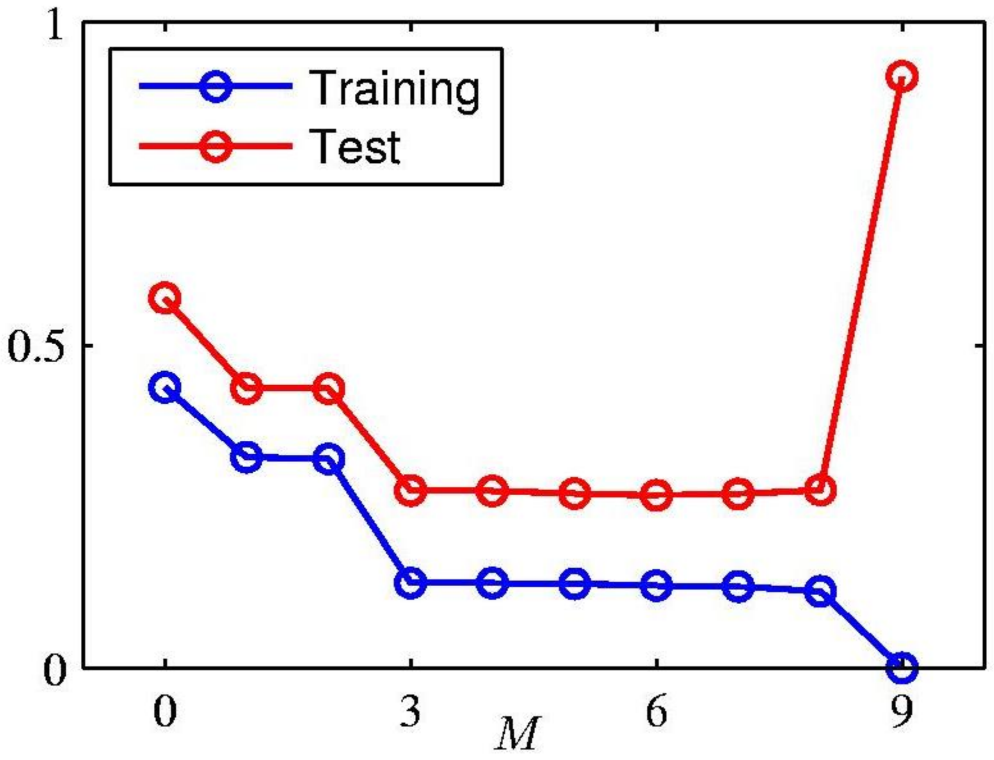
y	x	x^2	…	x^9
2.0	1.2	$(1.2)^2$	…	$(1.2)^9$
1.3	1.7	$(1.7)^2$	…	$(1.7)^9$
0.1	2.7	$(2.7)^2$	…	$(2.7)^9$
1.1	1.9	$(1.9)^2$	…	$(1.9)^9$

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



Slide credit: CMU MLD Matt Gormley

Over-fitting



Slide credit: CMU MLD 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 credit: CMU MLD William Cohen

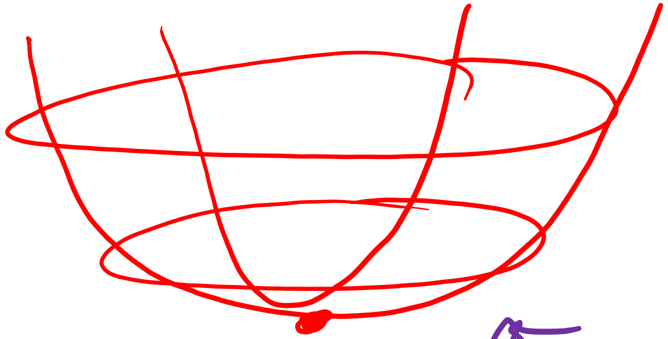
Regularization

θ

Given objective function: $J(\theta)$

Goal is to find: $\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}}\ J(\boldsymbol{\theta}) + \lambda r(\boldsymbol{\theta})$

$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$



Key idea: Define regularizer $r(\boldsymbol{\theta})$ s.t. we tradeoff between fitting the data and keeping the model simple

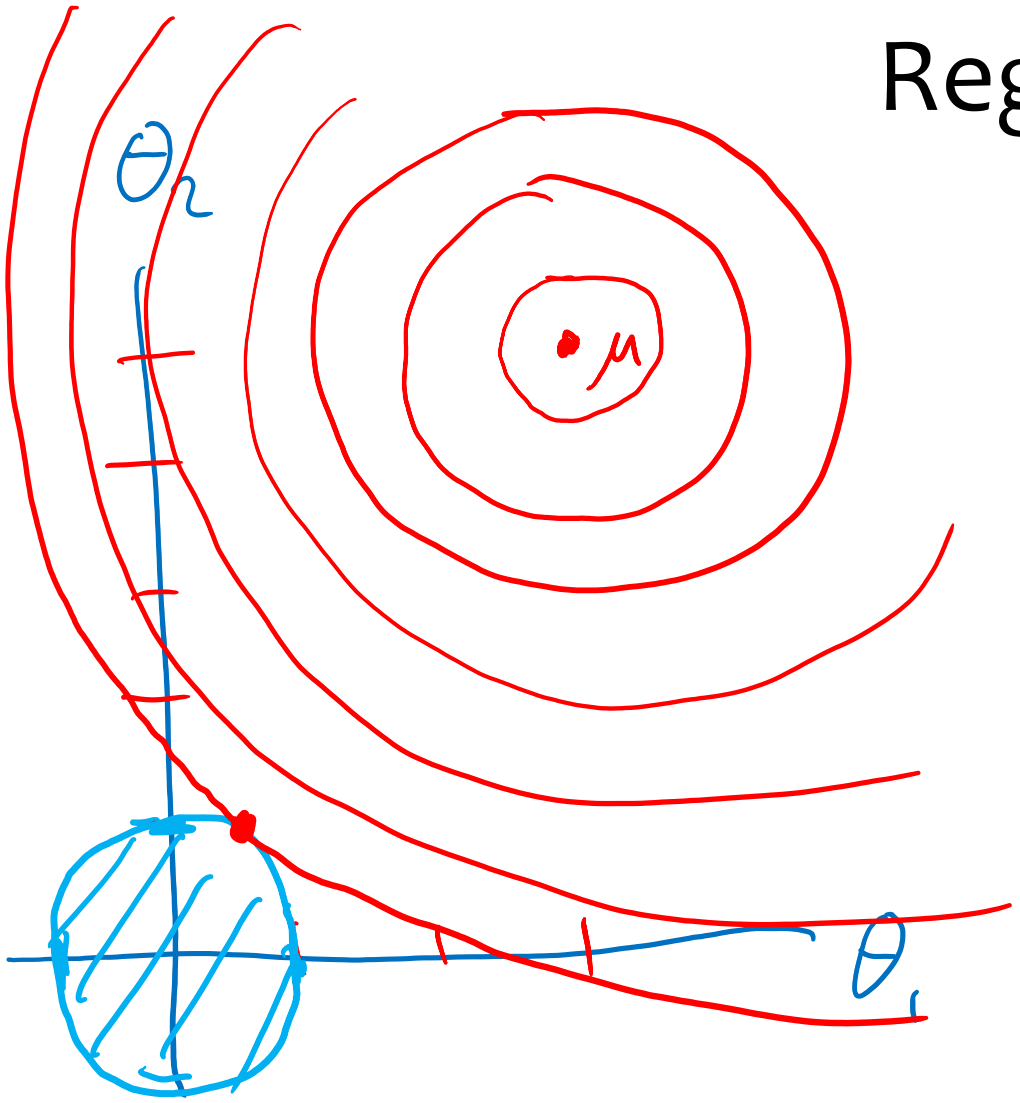
Choose form of $r(\boldsymbol{\theta})$:

– Example: q-norm (usually p-norm) $r(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_q = \left[\sum_{m=1}^{M} ||\theta_m||^q\right]^{(\frac{1}{q})}$

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

Slide credit: CMU MLD Matt Gormley

Regularization



$$J(\theta_1,\theta_2) = \|\vec{\theta} - \vec{\mu}\| \qquad \mu = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$$

$$\min_{\theta} \; J(\theta_1, \theta_2)$$

$$\text{s.t. } \|\theta\|_2 \le 1$$

Previous Piazza Poll

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)

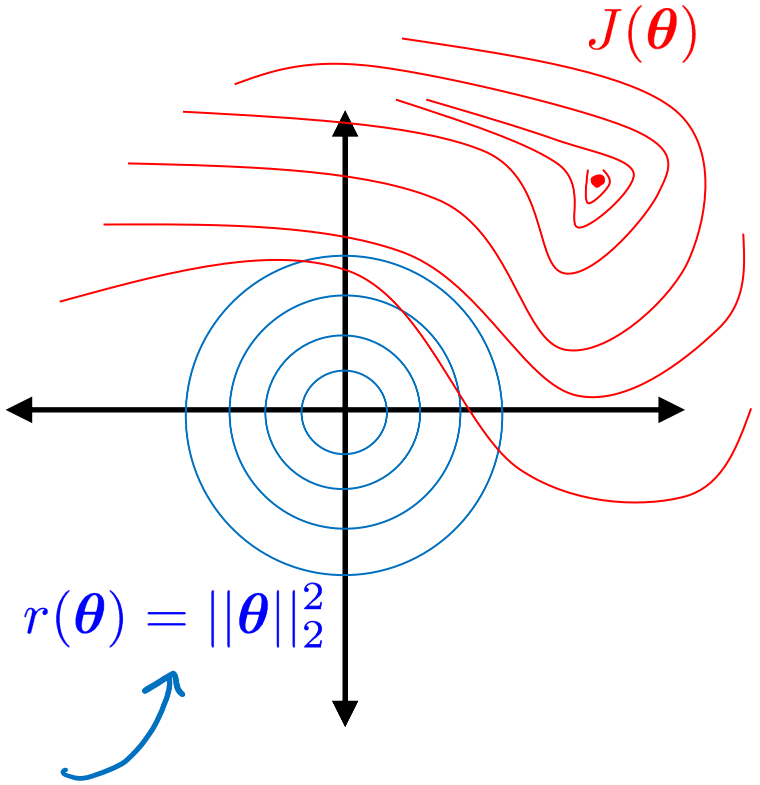
B. …move towards the minimum of J(θ)

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



Slide credit: CMU MLD Matt Gormley

Regularization Exercise

In-class Exercise

1. Plot train error vs. regularization weight (cartoon)
2. Plot test error vs . regularization weight (cartoon)

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

Slide credit: CMU MLD Matt Gormley

Piazza Poll 1

Question:

Suppose we are minimizing J'(θ) where

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

As we increase λ from zero, the **validation** error will…

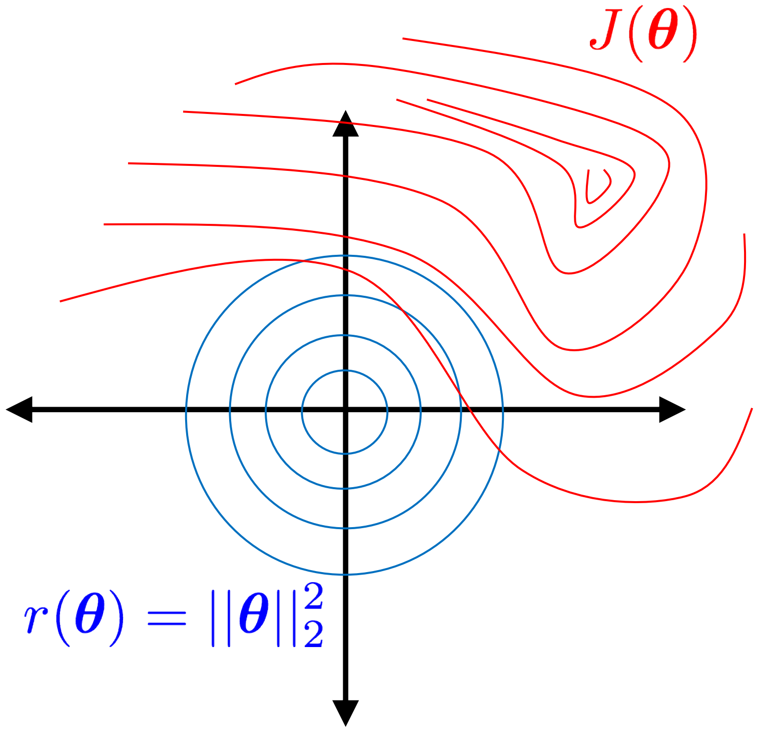
A. …increase

B. …decrease

C. …first increase, then decrease

D. …first decrease, then increase

E. …stay the same



Slide credit: CMU MLD Matt Gormley

Regularization

Don’t Regularize the Bias (Intercept) Parameter!

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

Whitening Data

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

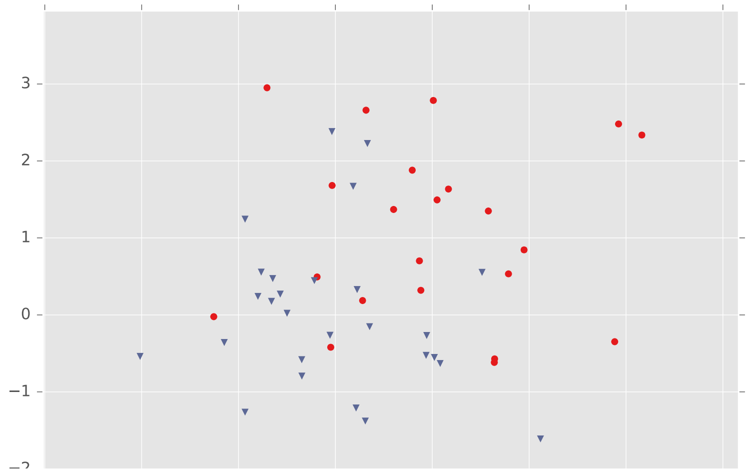
Slide credit: CMU MLD Matt Gormley

Logistic Regression with Nonlinear Features

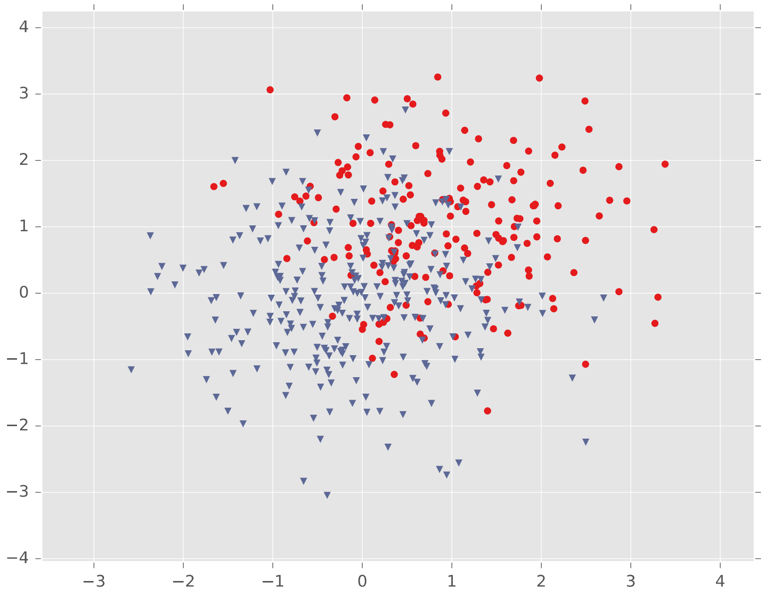
Jupyter notebook demo

Example: Logistic Regression

Training Data

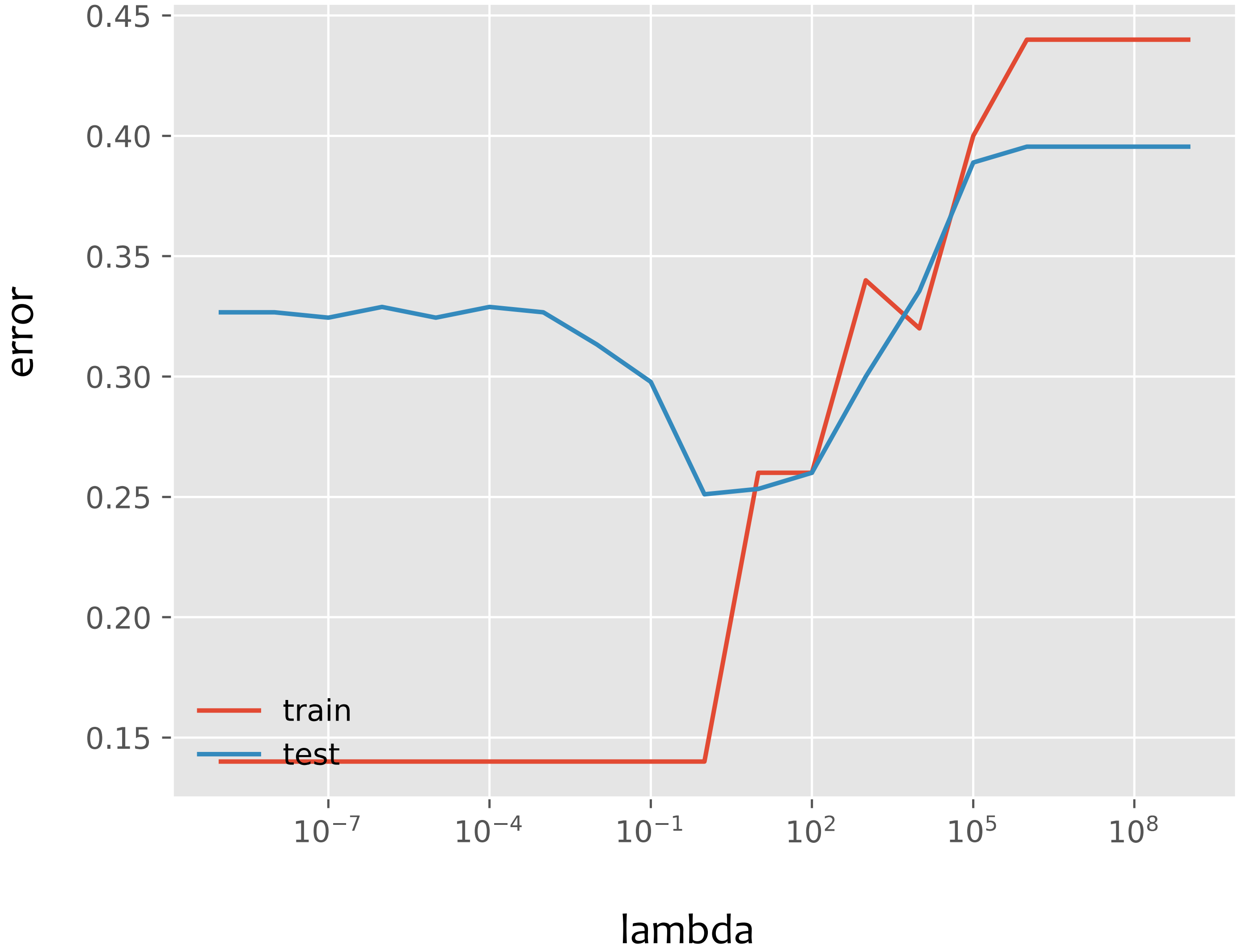


Test Data

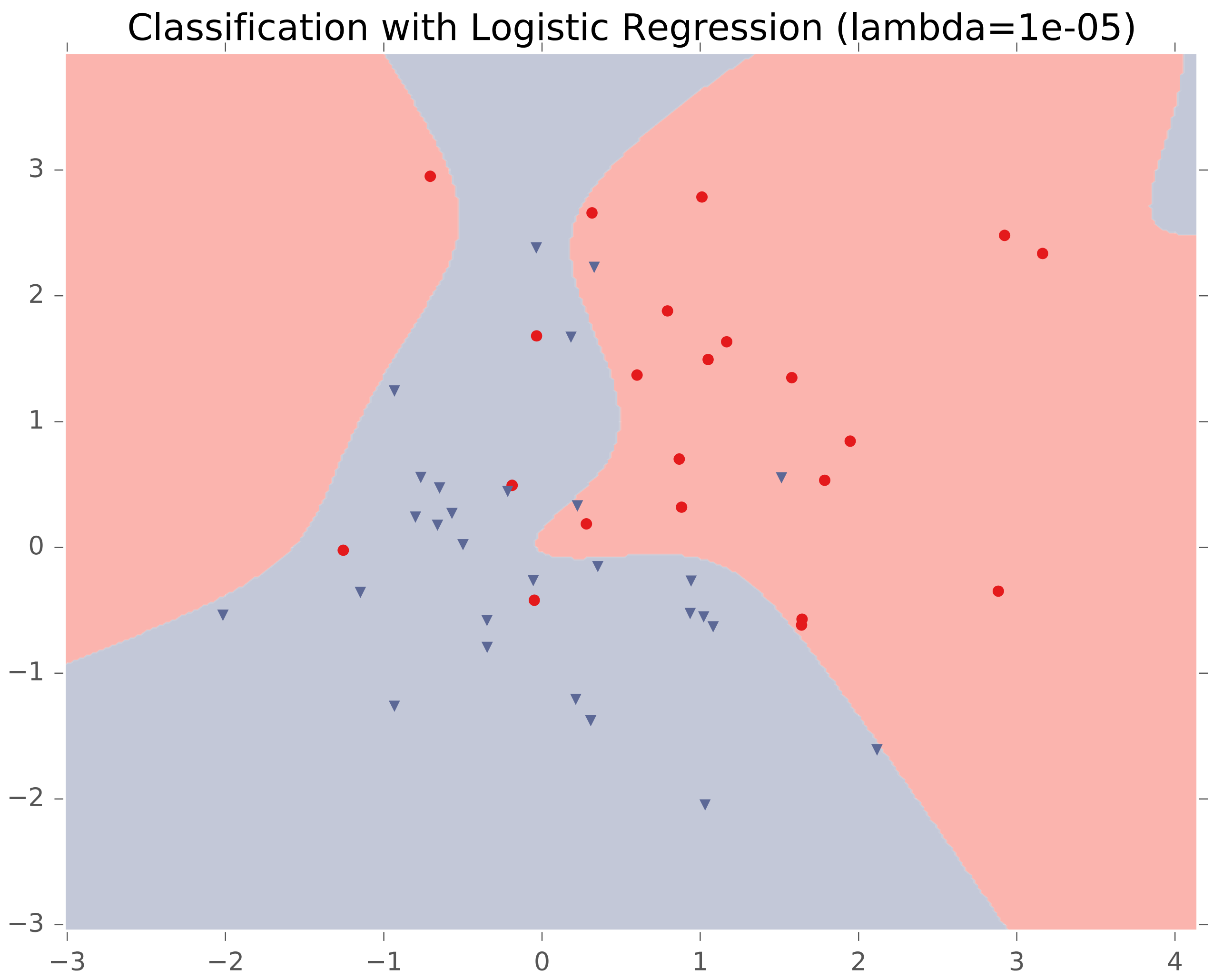


- For this example, we construct **nonlinear features** (i.e. feature engineering)
- Specifically, we add **polynomials up to order 9** of the two original features x_1 and x_2
- Thus our classifier is **linear** in the **high-dimensional feature space**, but the decision boundary is **nonlinear** when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

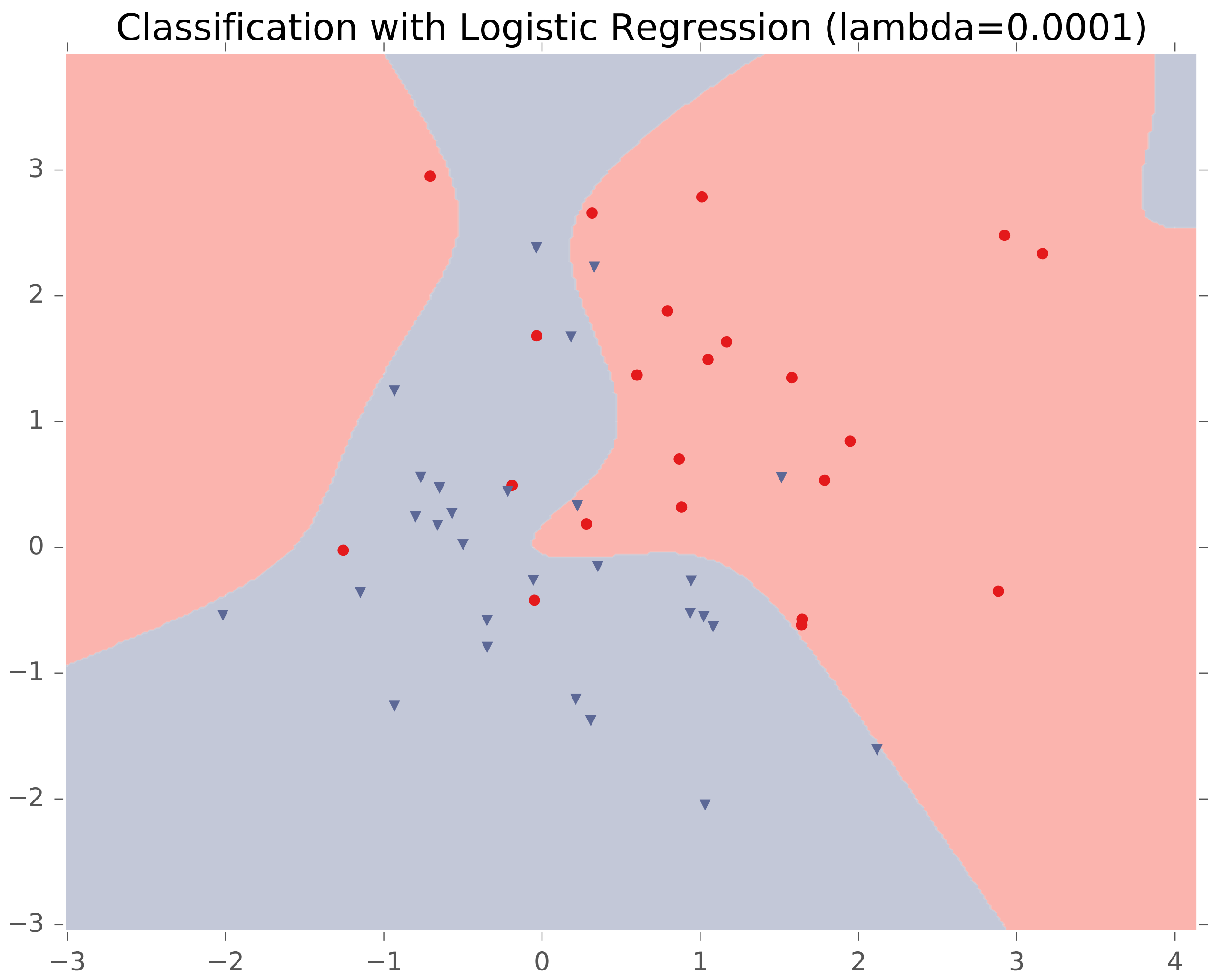
Example: Logistic Regression



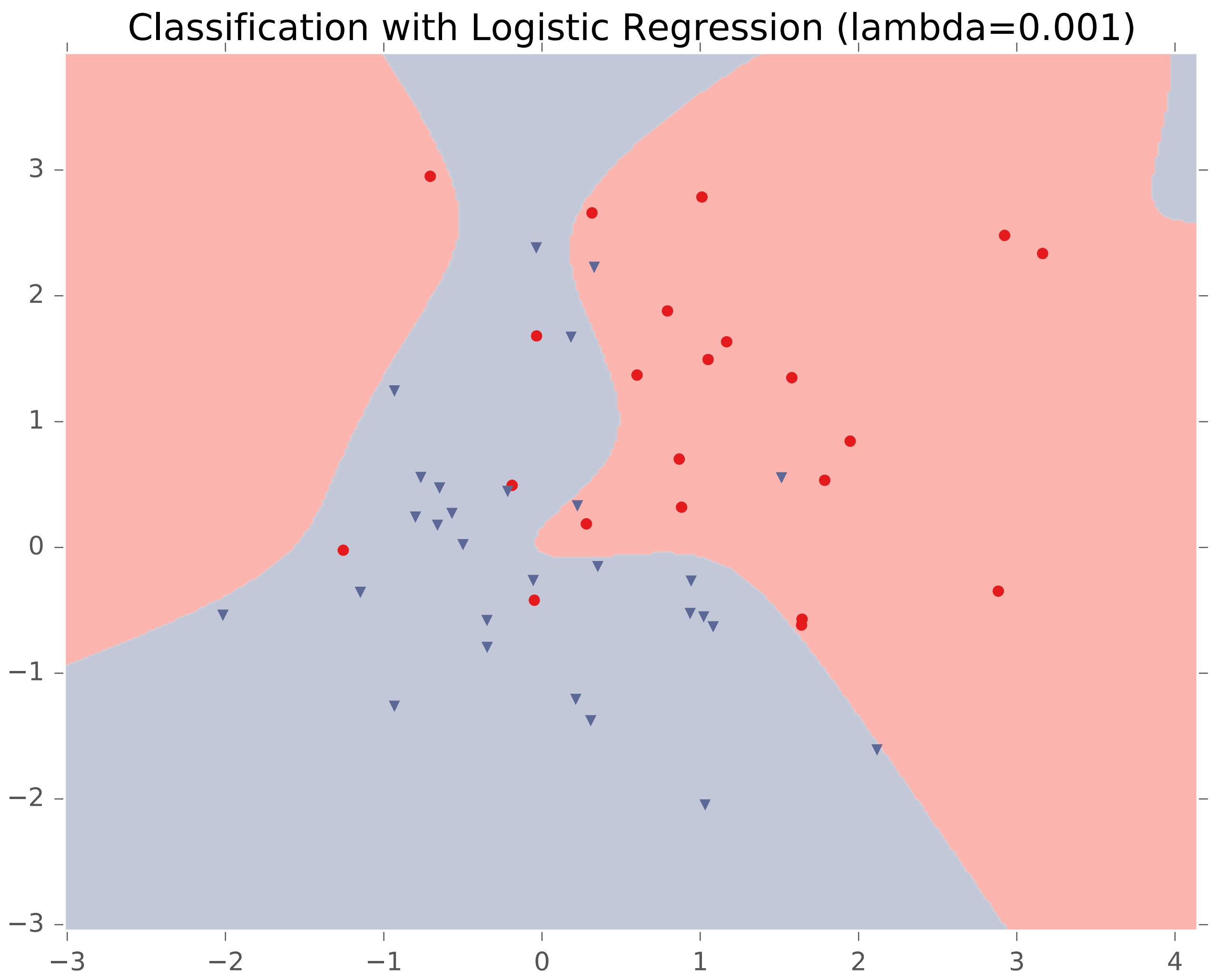
Example: Logistic Regression



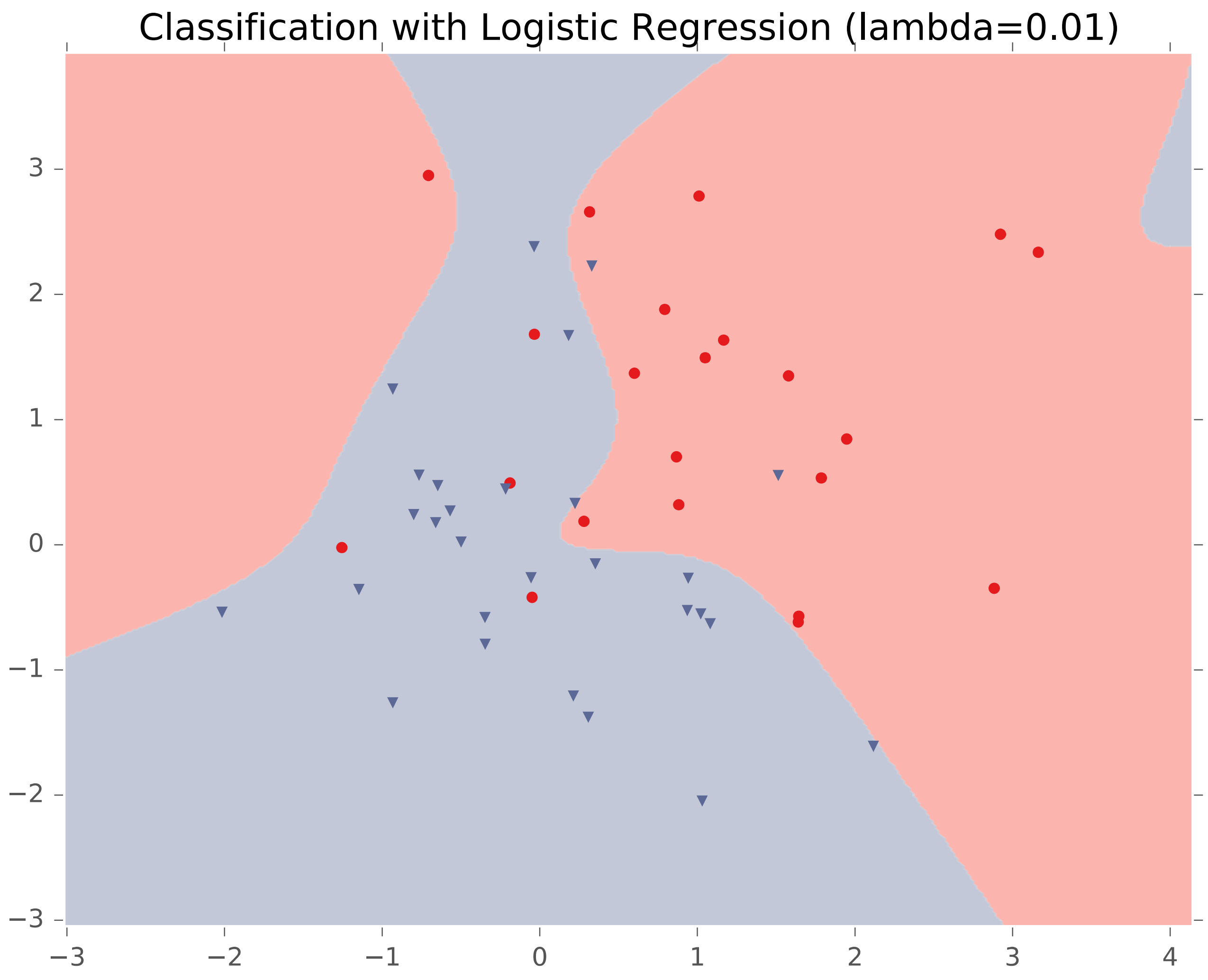
Example: Logistic Regression



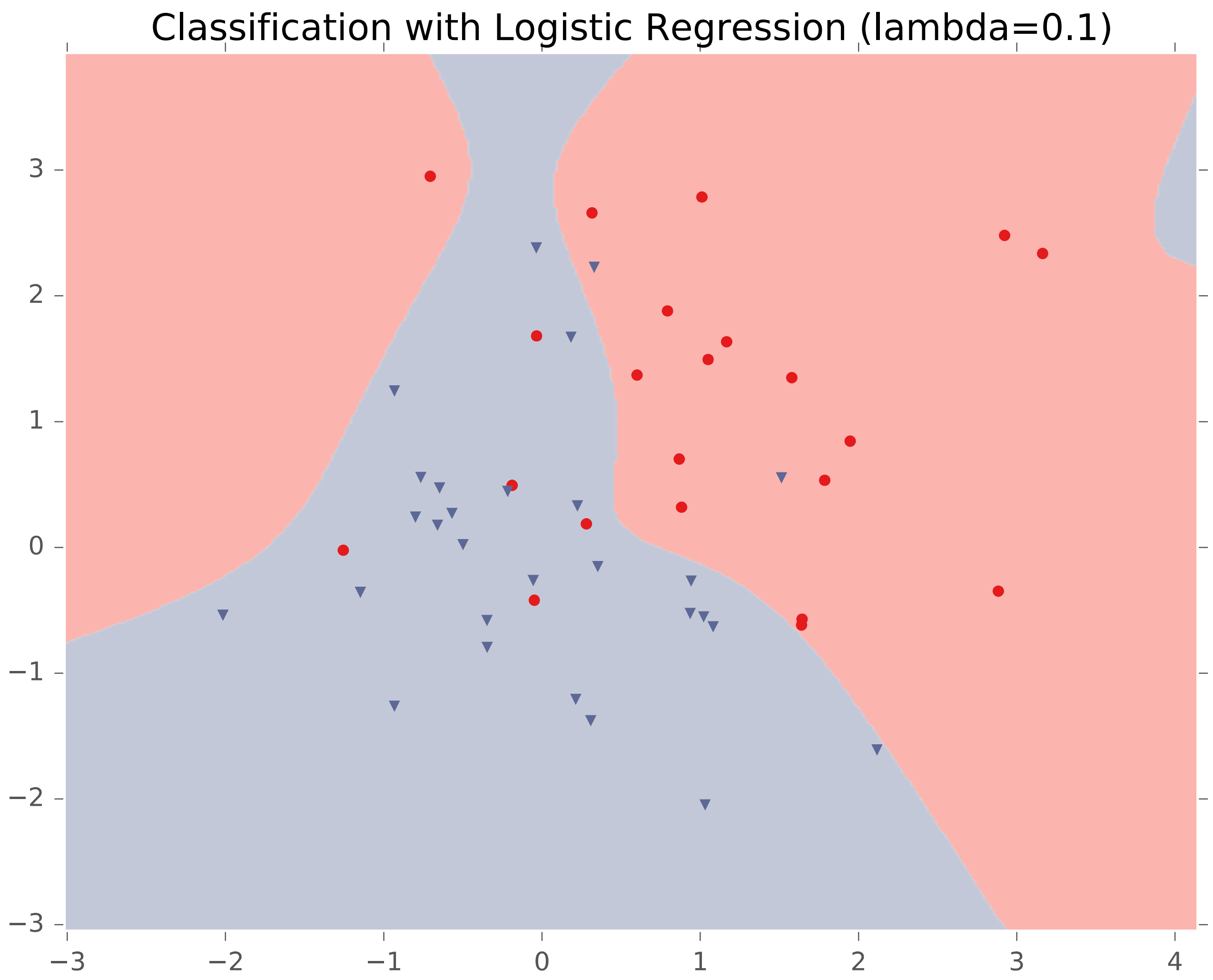
Example: Logistic Regression



Example: Logistic Regression

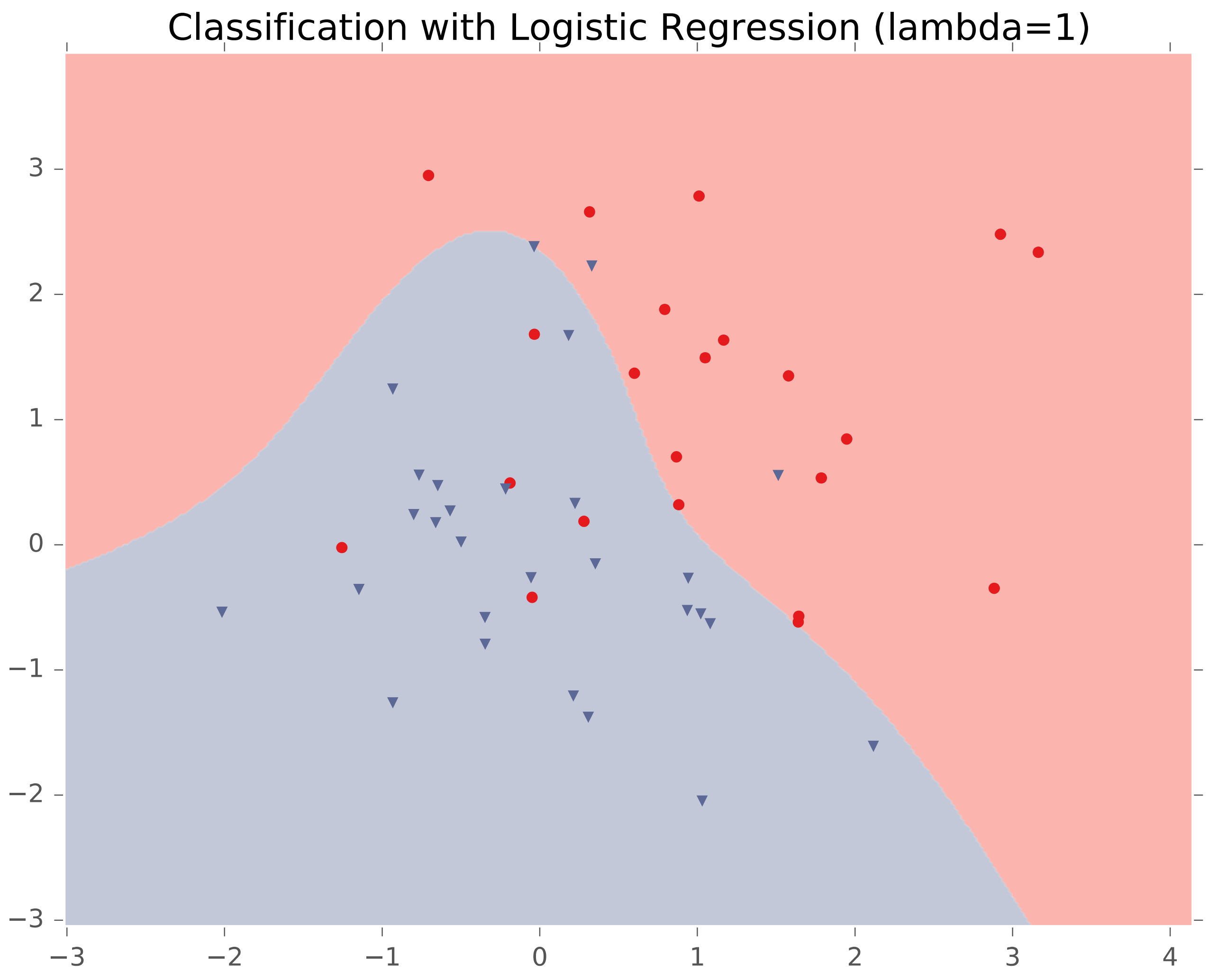


Example: Logistic Regression



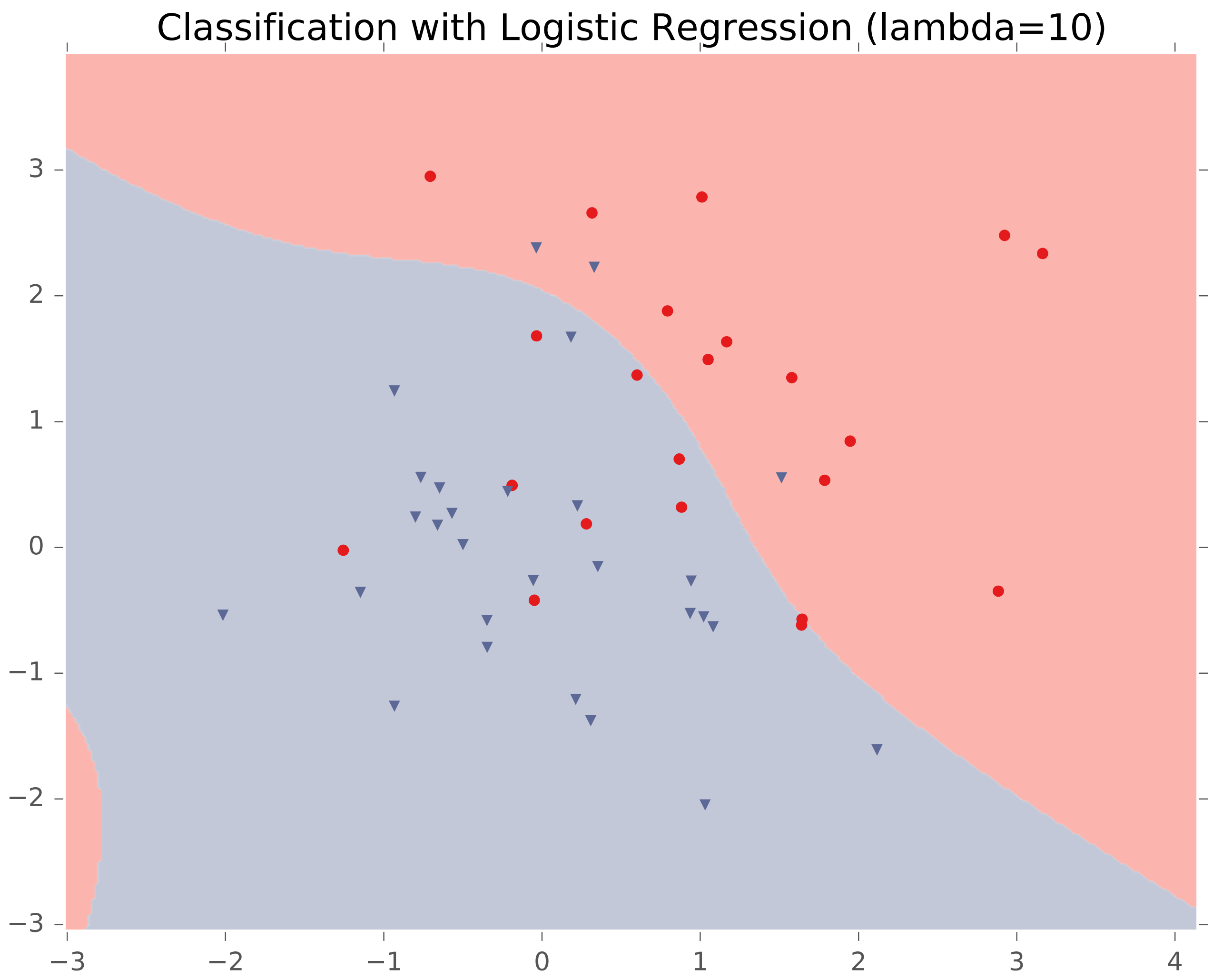
Slide credit: CMU MLD Matt Gormley

Example: Logistic Regression

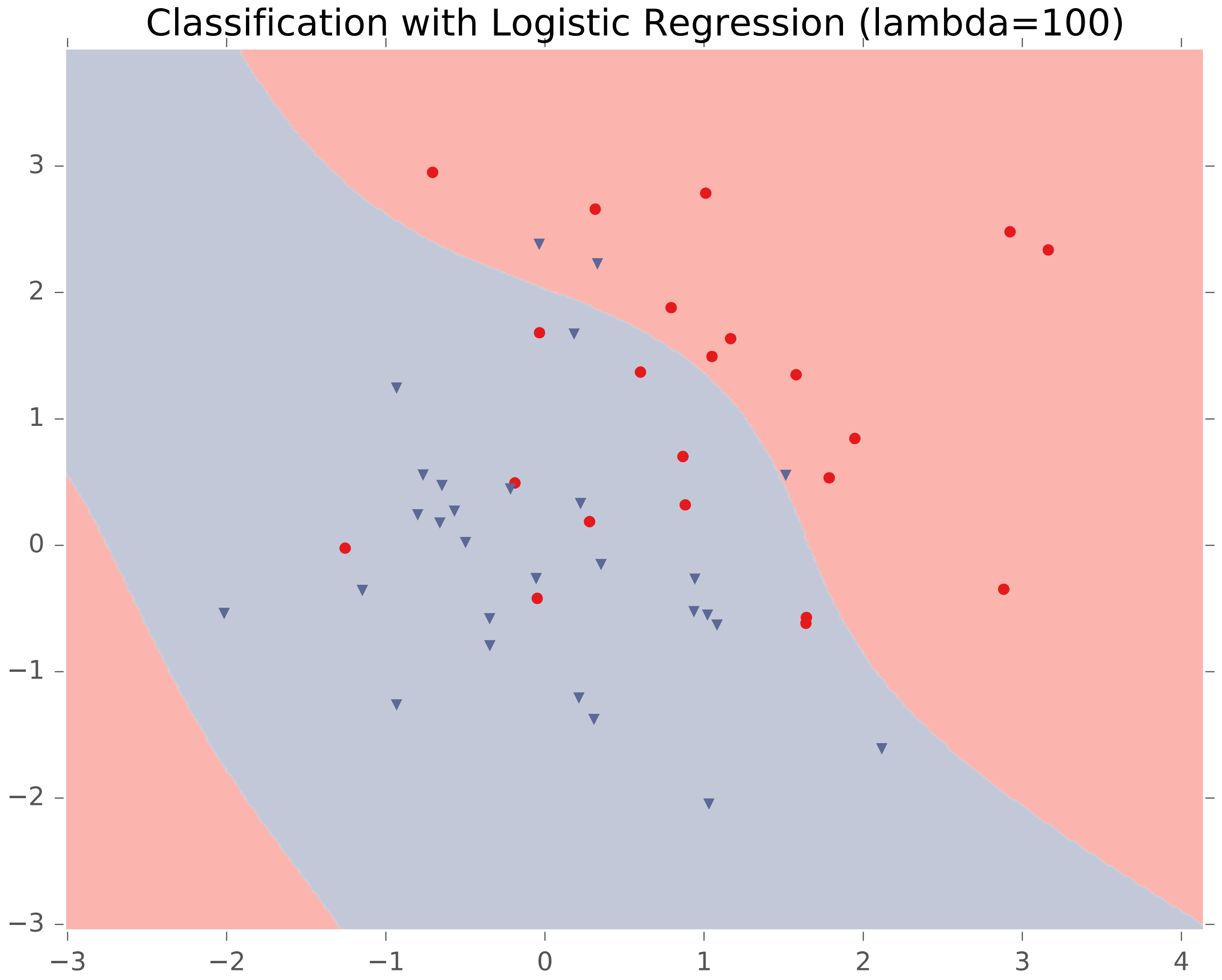


Slide credit: CMU MLD Matt Gormley

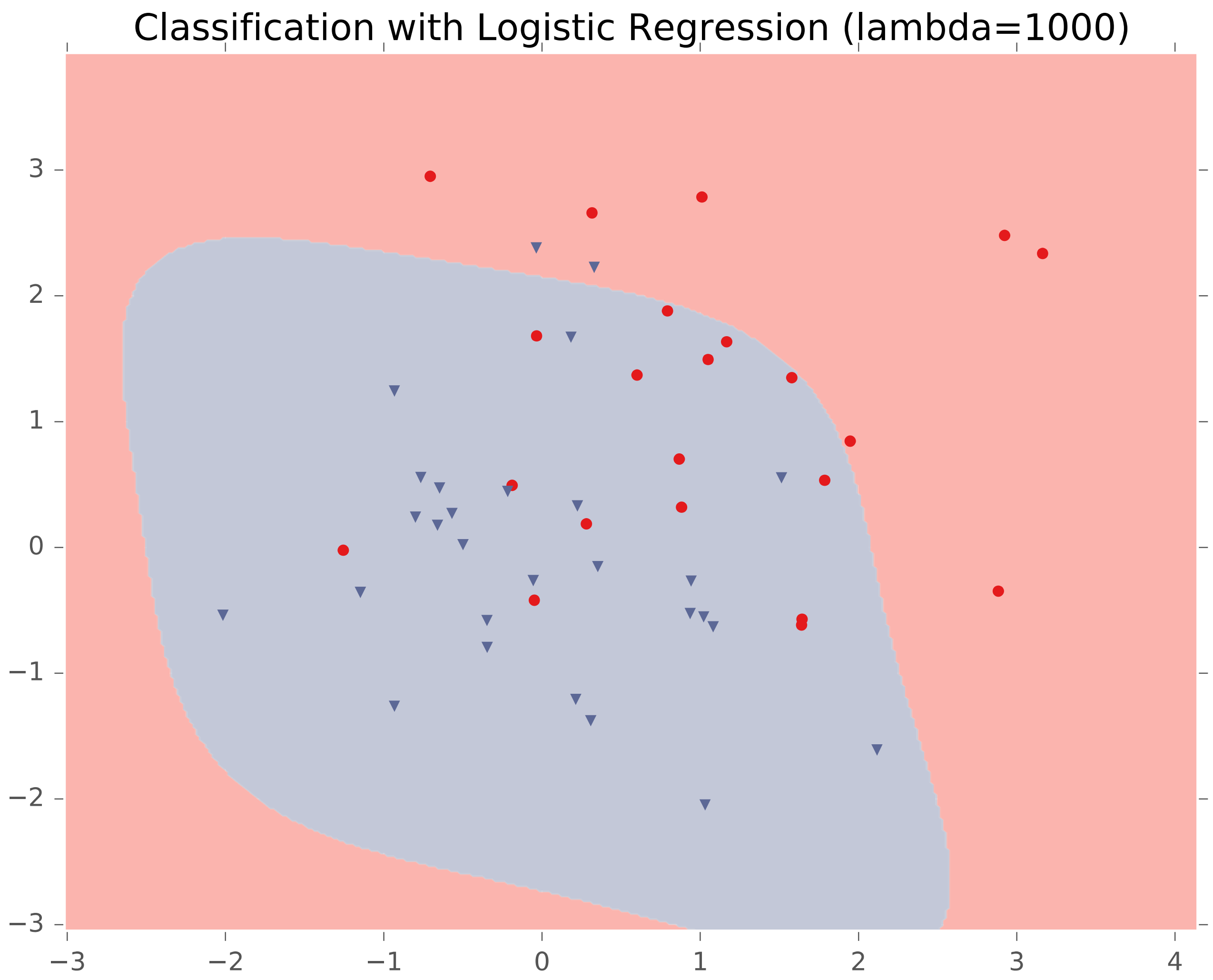
Example: Logistic Regression



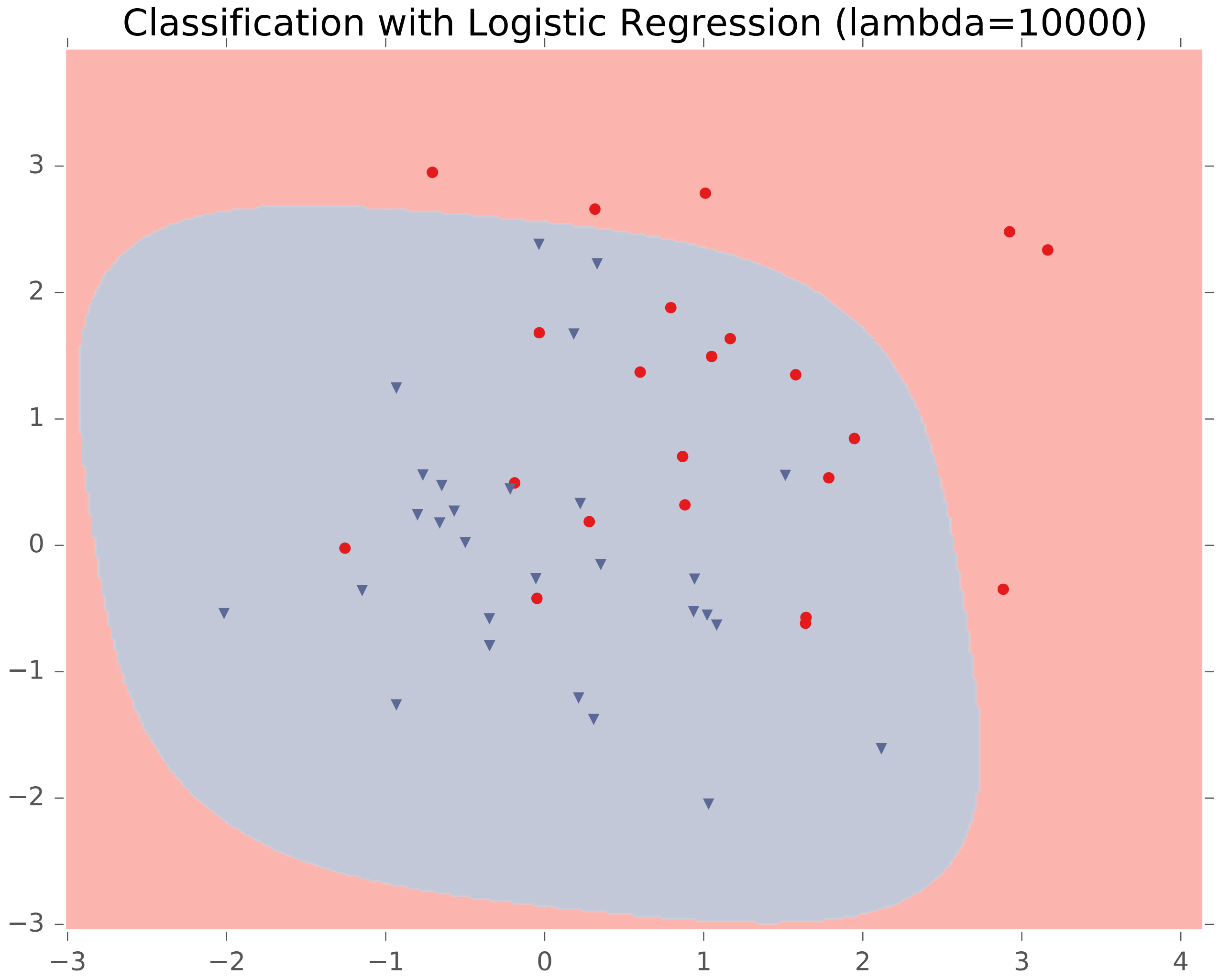
Example: Logistic Regression



Example: Logistic Regression

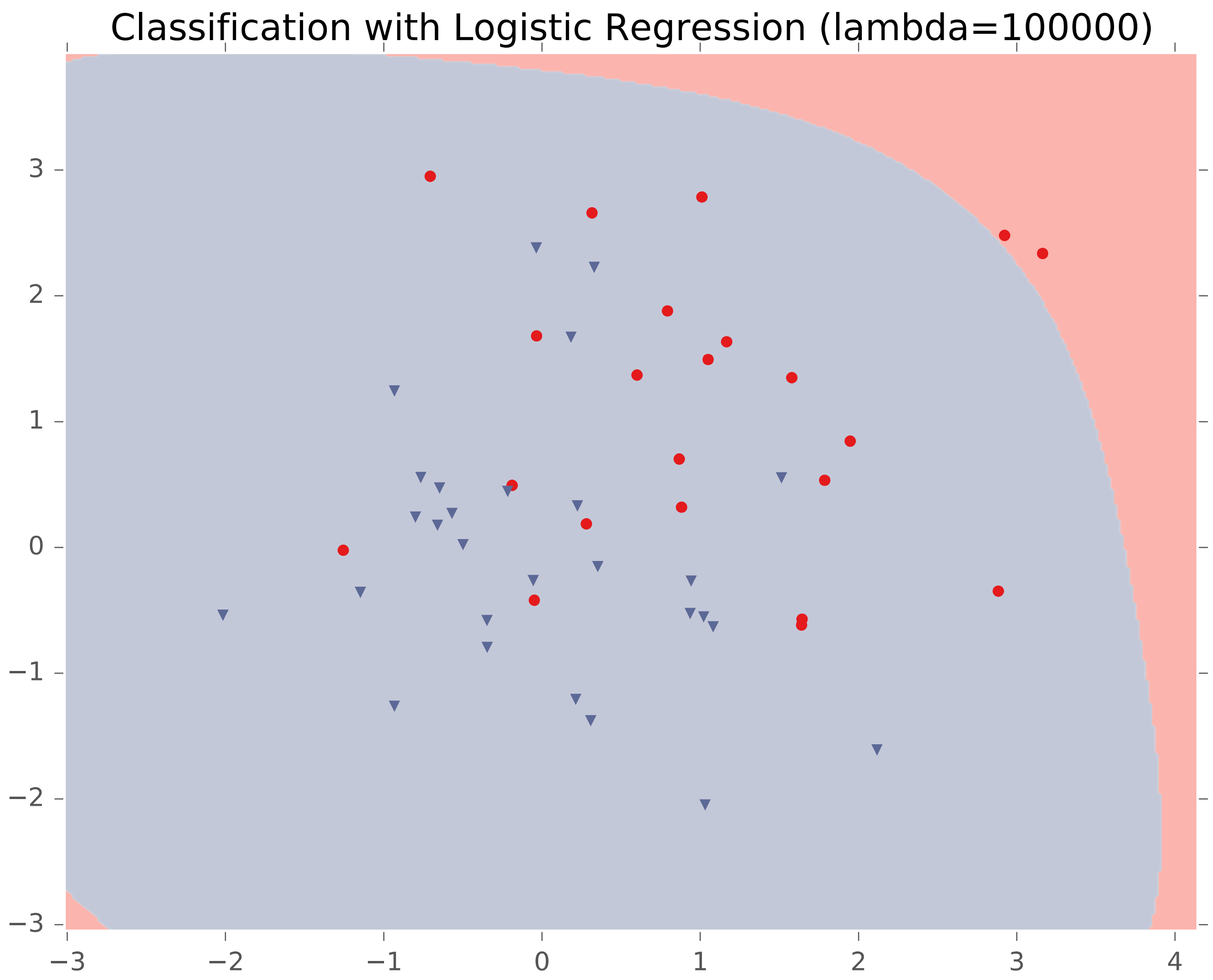


Example: Logistic Regression



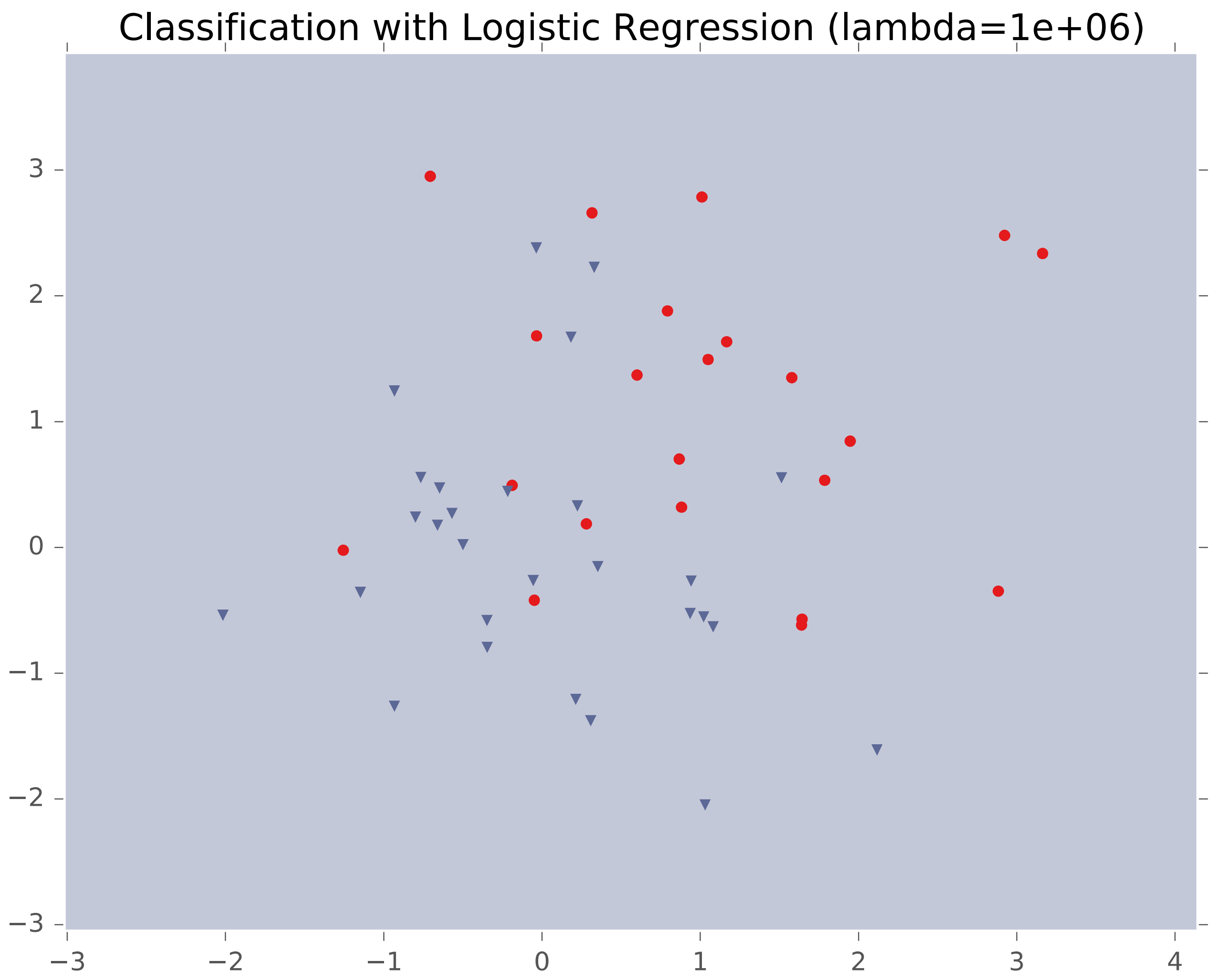
Slide credit: CMU MLD Matt Gormley

Example: Logistic Regression

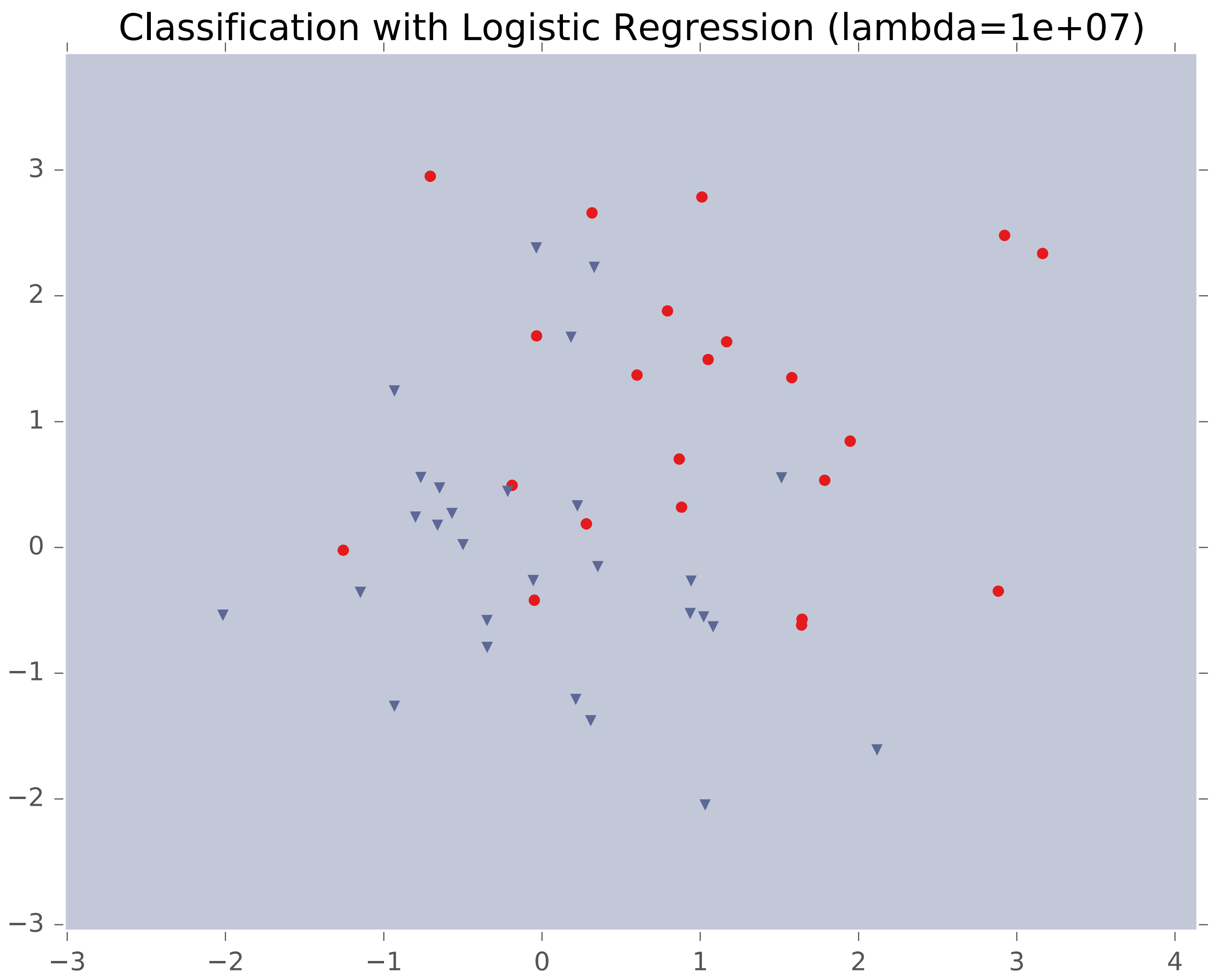


Slide credit: CMU MLD Matt Gormley

Example: Logistic Regression

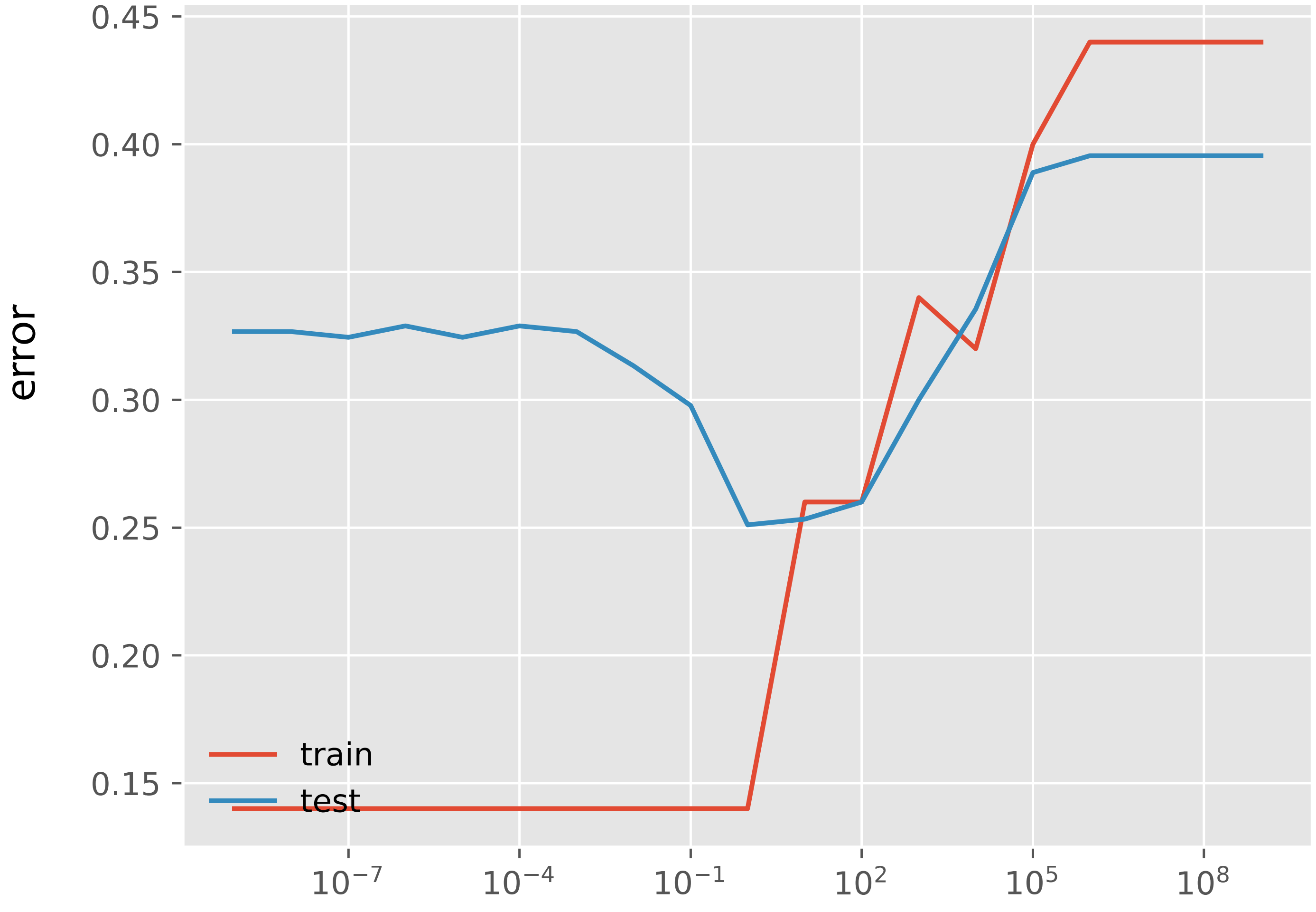


Example: Logistic Regression



Slide credit: CMU MLD Matt Gormley

Example: Logistic Regression



lambda

Slide credit: CMU MLD Matt Gormley

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 $r(\boldsymbol{\theta})$ s.t. we tradeoff between fitting the data and keeping the model simple

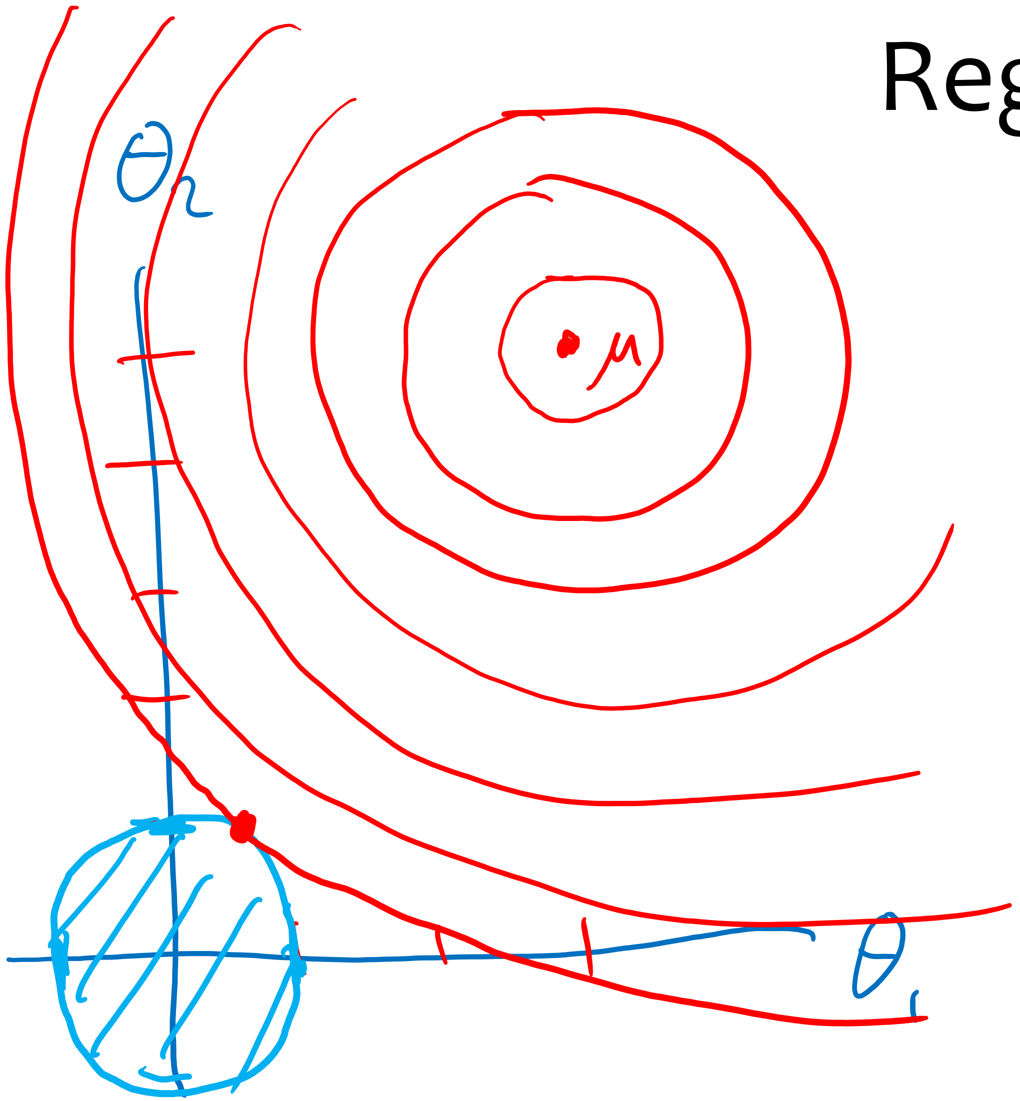
Choose form of $r(\boldsymbol{\theta})$:

– Example: q-norm (usually p-norm) $r(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_q = \left[\sum_{m=1}^{M} ||\theta_m||^q\right]^{\left(\frac{1}{q}\right)}$

q	$r(\boldsymbol{\theta})$	**yields parameters that are...**	**name**	**optimization notes**
0	$\lvert\lvert\boldsymbol{\theta}\rvert\rvert_0 = \sum \mathbb{1}(\theta_m \neq 0)$	zero values	L0 reg.	no good computational solutions
1	$\lvert\lvert\boldsymbol{\theta}\rvert\rvert_1 = \sum \lvert\theta_m\rvert$	zero values	L1 reg.	subdifferentiable
2	$(\lvert\lvert\boldsymbol{\theta}\rvert\rvert_2)^2 = \sum \theta_m^2$	small values	L2 reg.	differentiable

Slide credit: CMU MLD Matt Gormley

Regularization



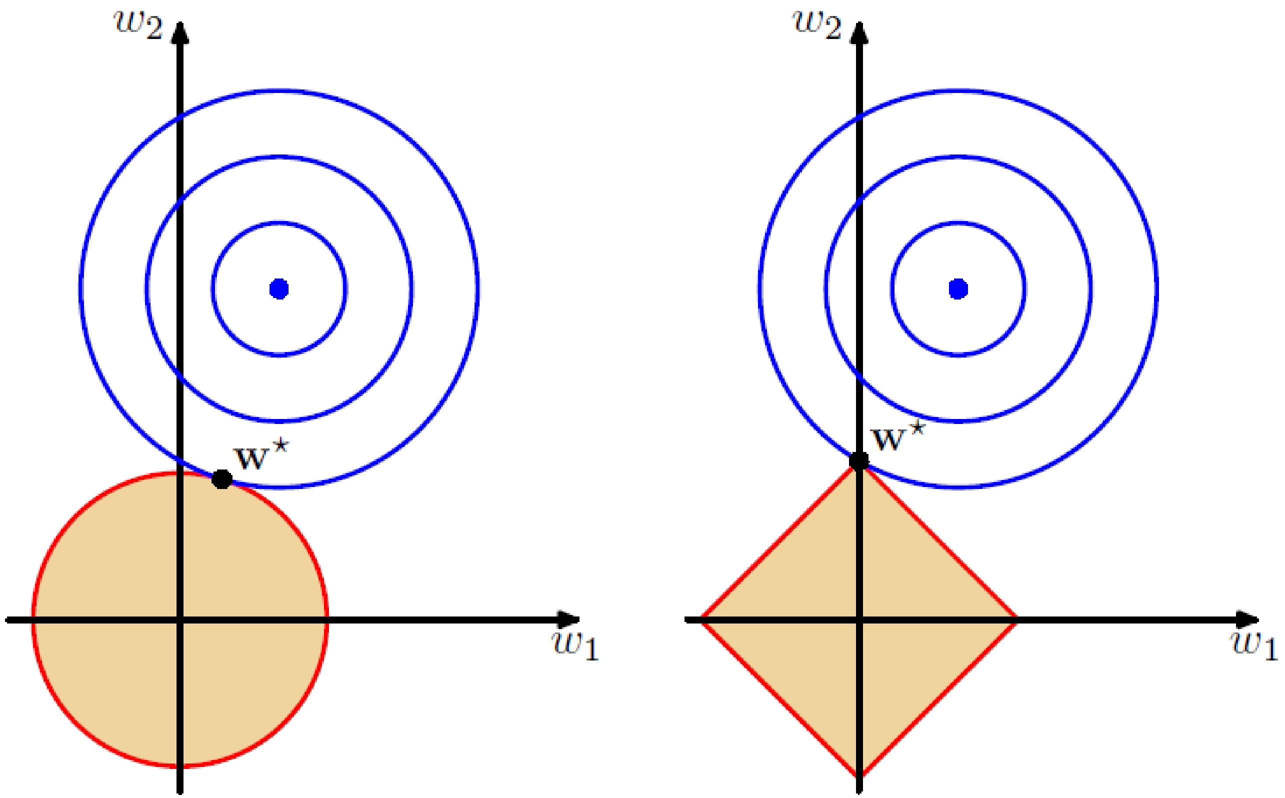
$$J(\theta_1, \theta_2) = \|\vec{\theta} - \vec{\mu}\| \qquad \mu = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$$

$$\min_{\theta} \; J(\theta_1, \theta_2)$$

$$\text{s.t. } \|\theta\|_2 \leq 1$$

L2 vs L1 Regularization

Combine original objective with penalty on parameters



Figures: Bishop, Ch 3.1.4

L2 vs L1: Housing Price Example

Predict housing price from several features

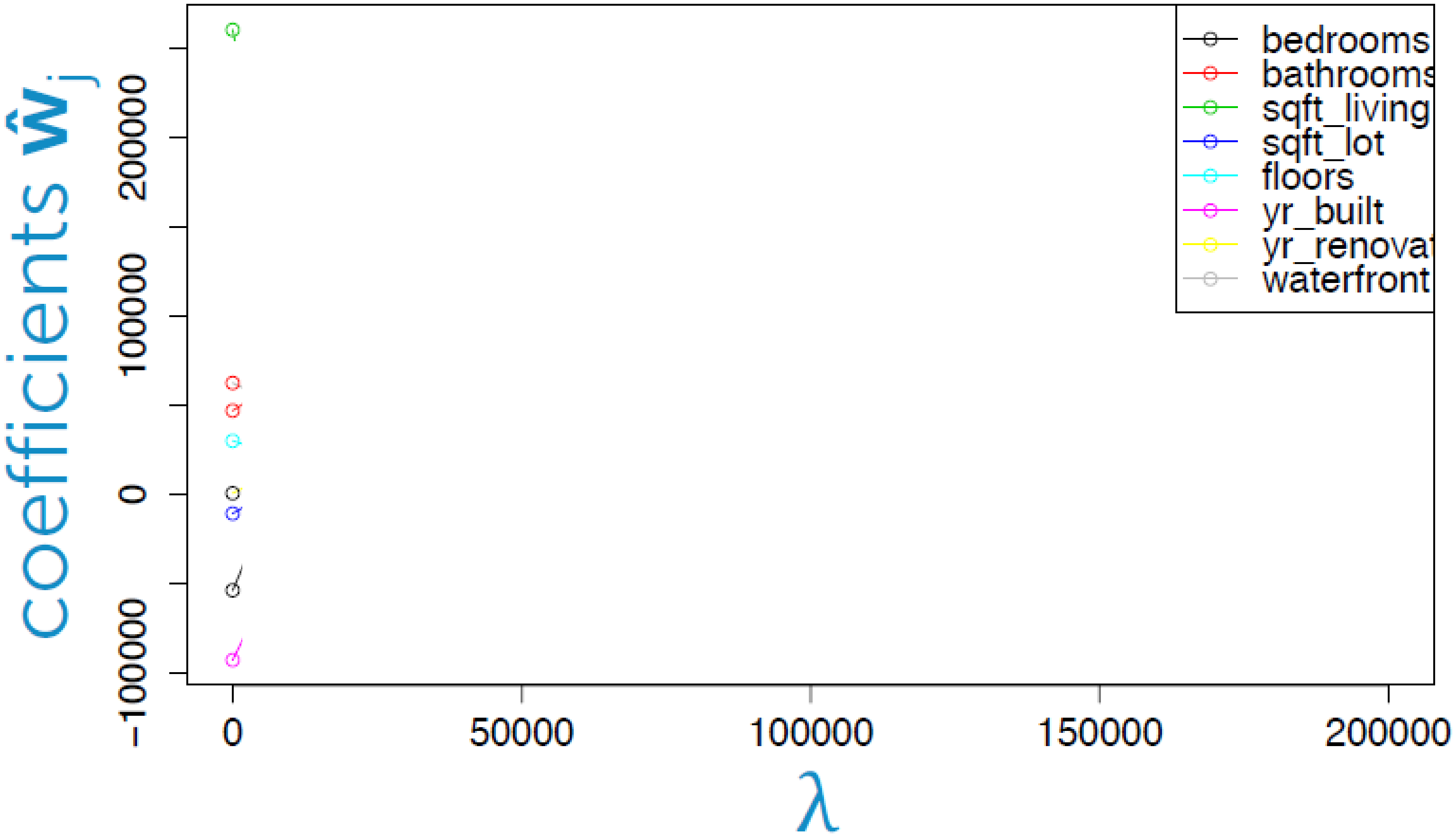


Figure: Emily Fox, University of Washington

L2 vs L1: Housing Price Example

Predict housing price from several features

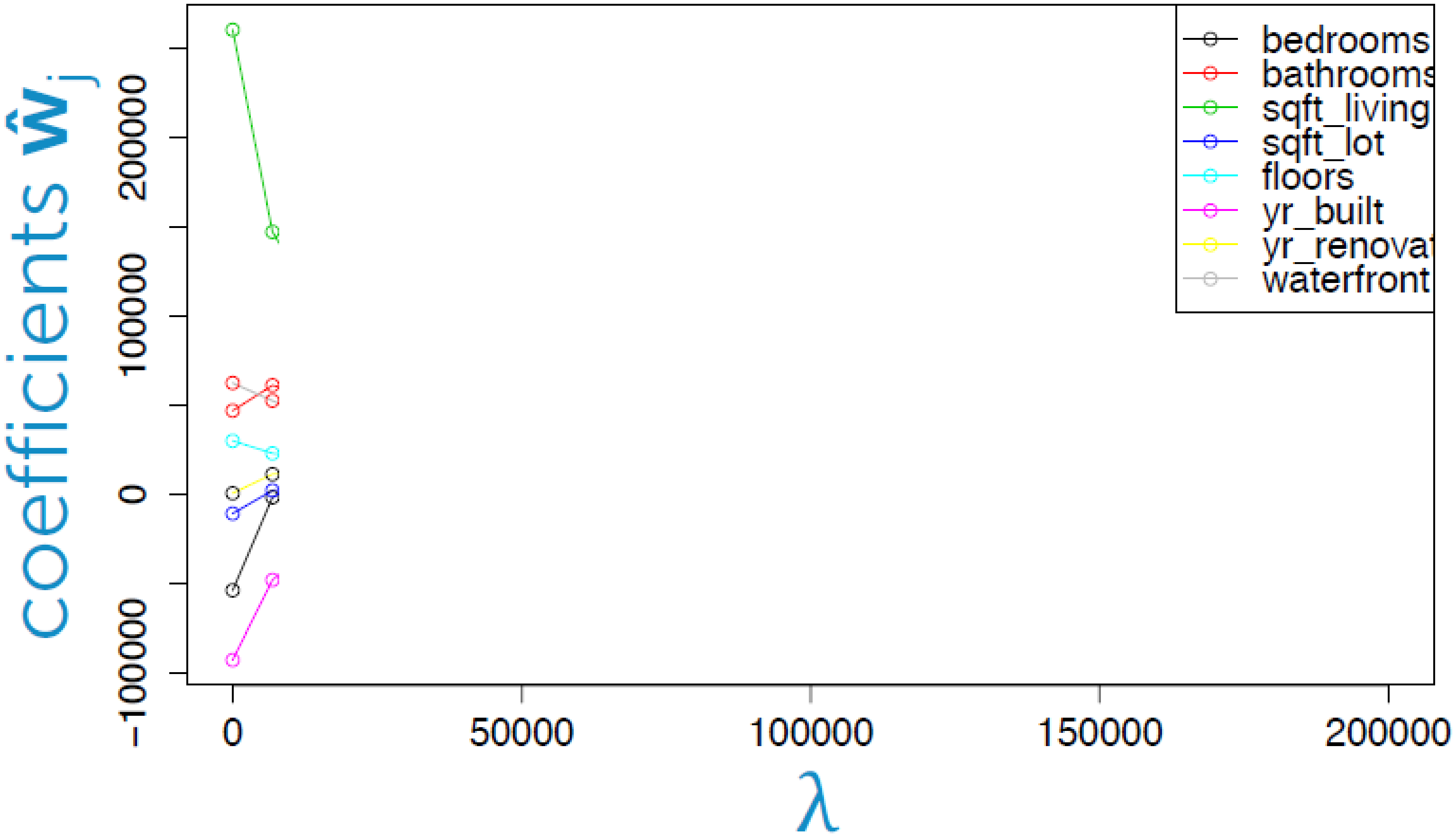


Figure: Emily Fox, University of Washington

L2 vs L1: Housing Price Example

Predict housing price from several features

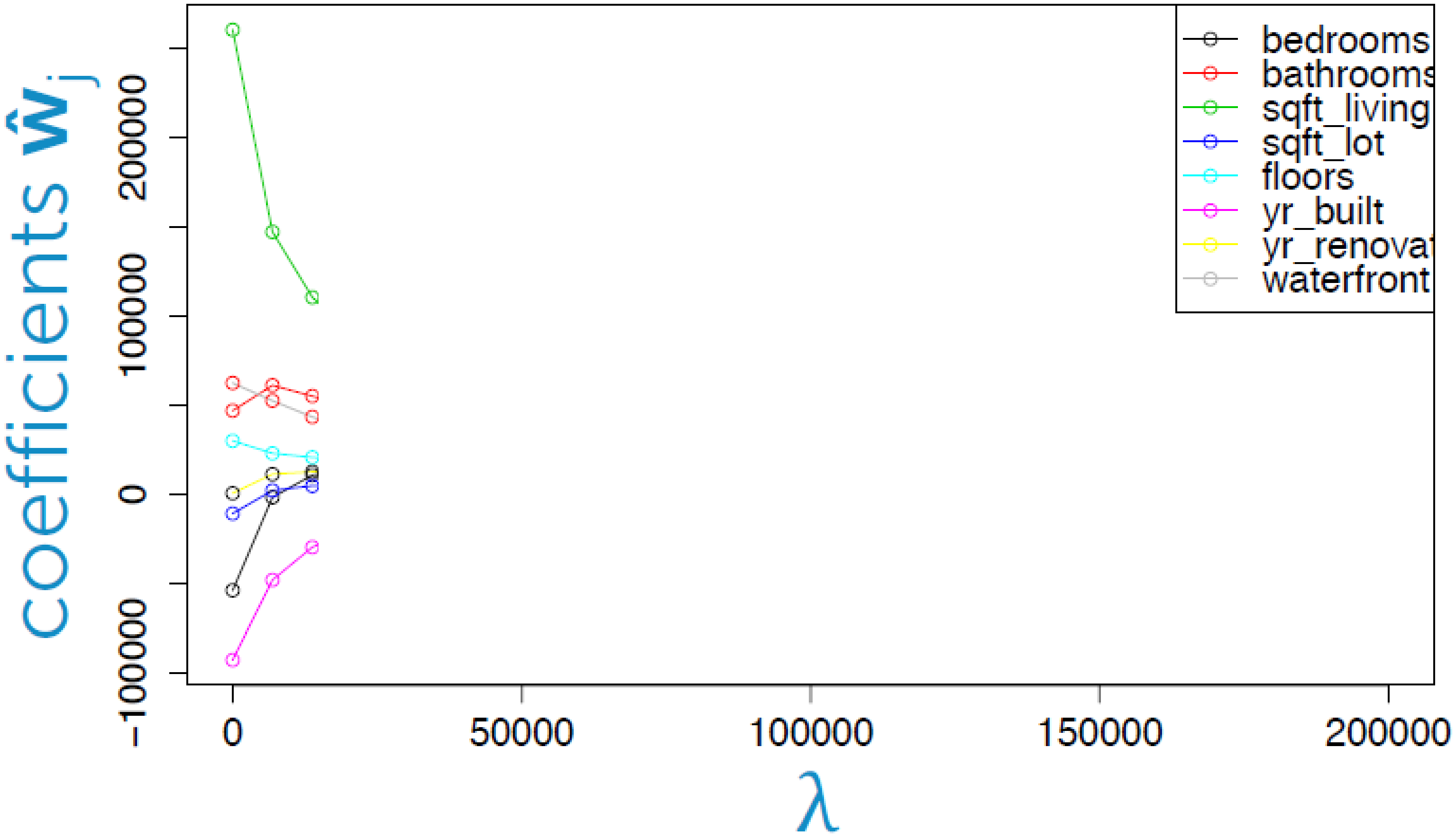


Figure: Emily Fox, University of Washington

L2 vs L1: Housing Price Example

Predict housing price from several features

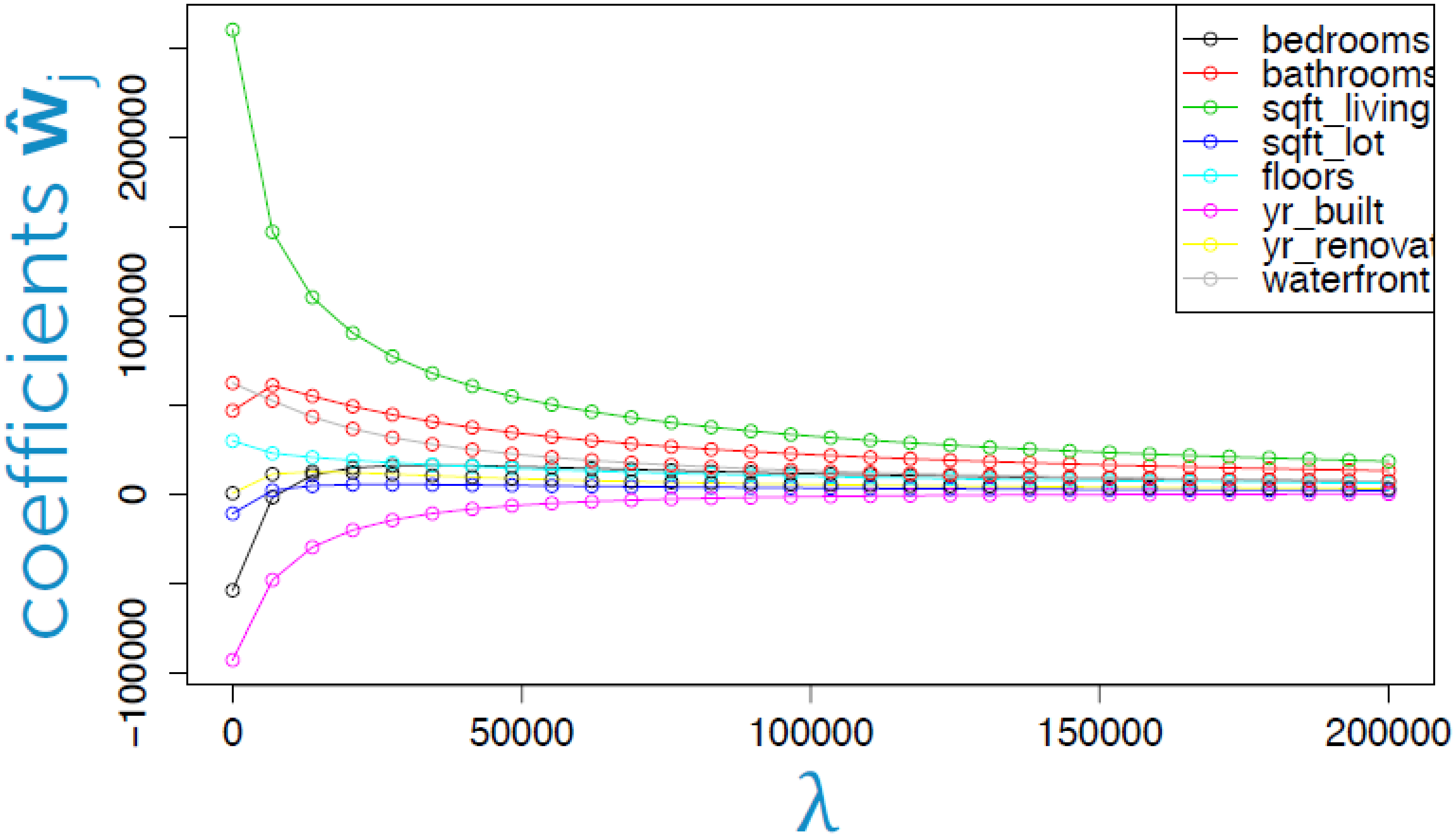


Figure: Emily Fox, University of Washington

L2 vs L1: Housing Price Example

Predict housing price from several features

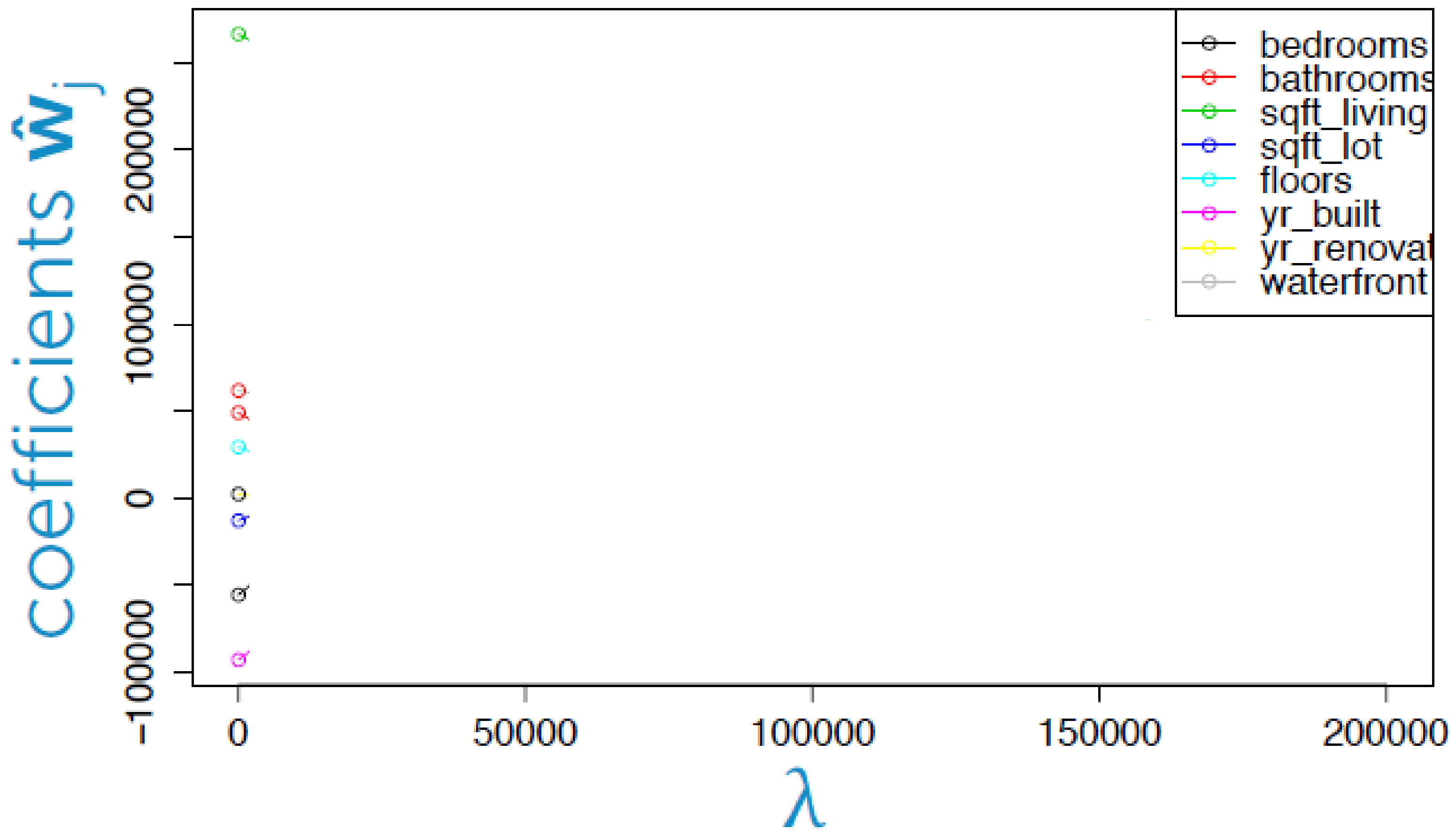


Figure: Emily Fox, University of Washington

L2 vs L1: Housing Price Example

Predict housing price from several features

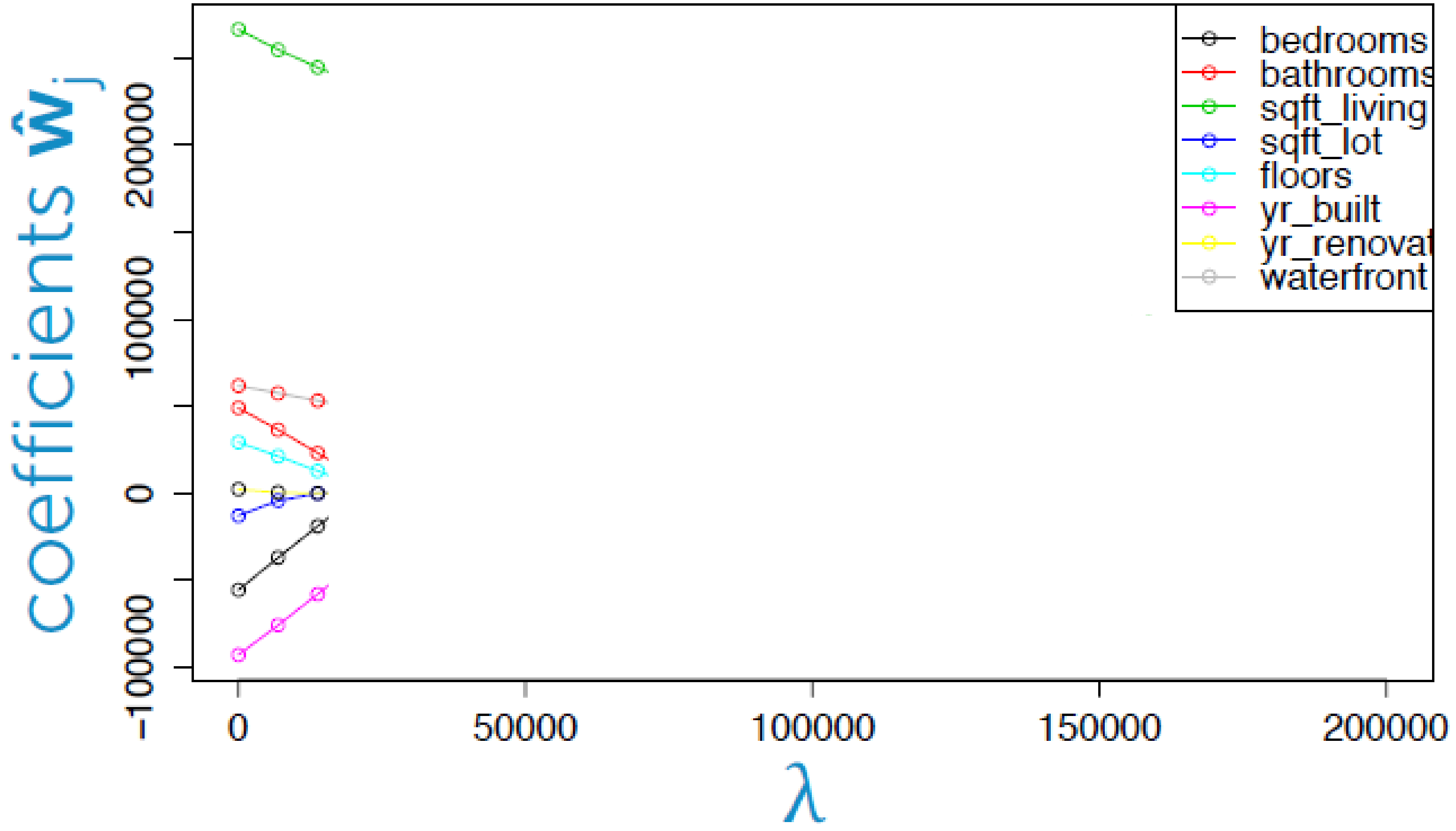


Figure: Emily Fox, University of Washington

L2 vs L1: Housing Price Example

Predict housing price from several features

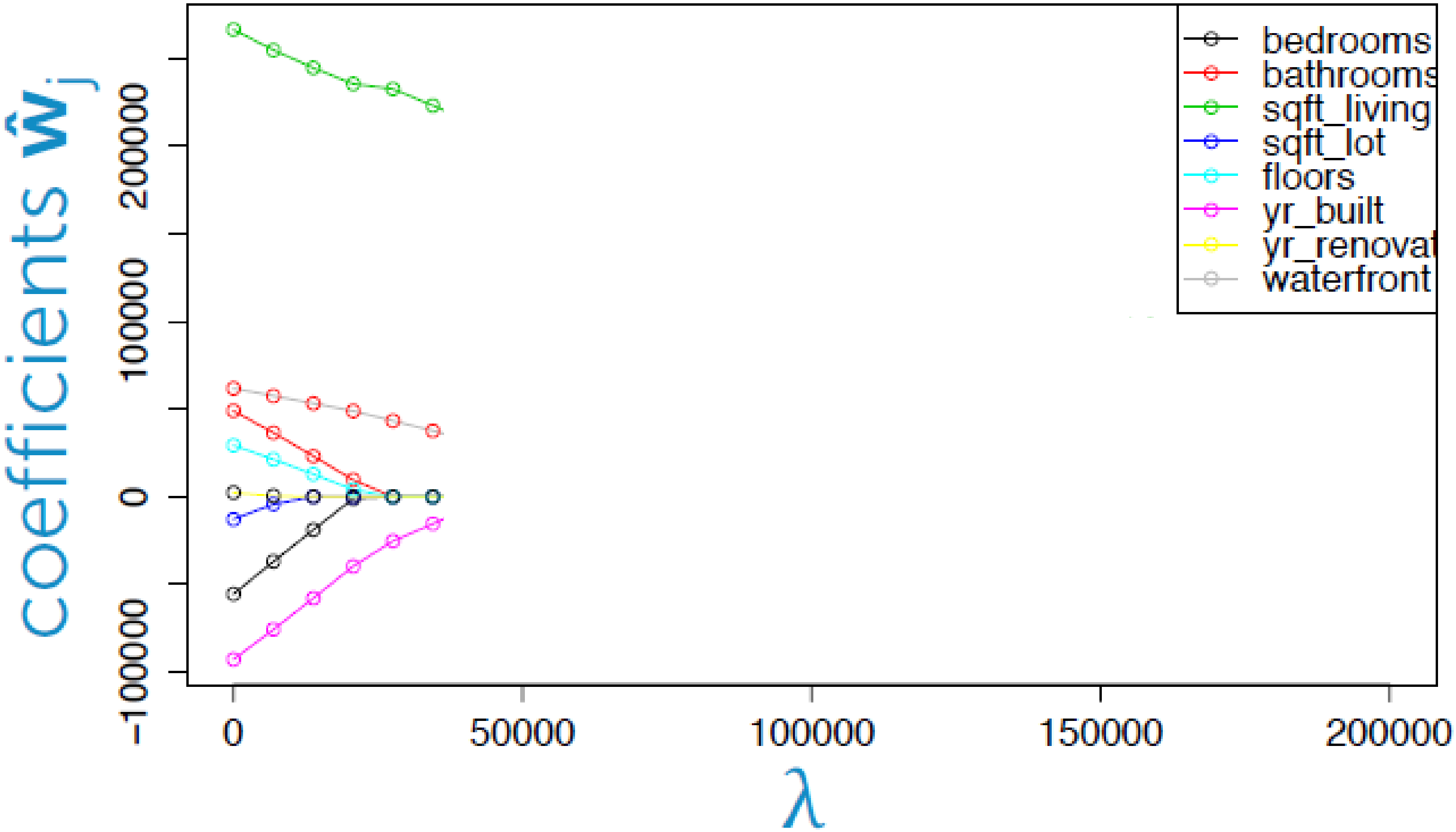


Figure: Emily Fox, University of Washington

L2 vs L1: Housing Price Example

Predict housing price from several features

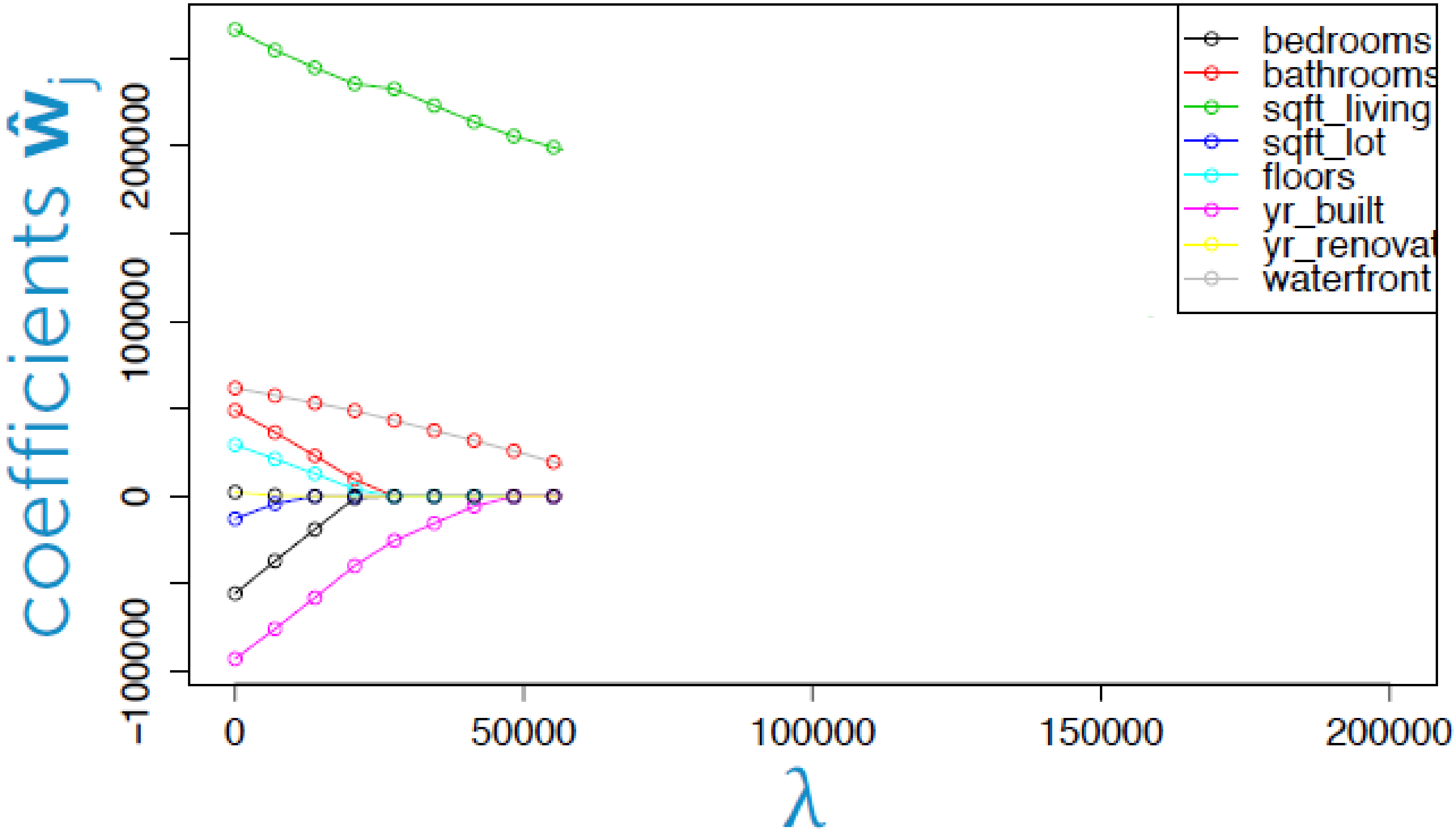


Figure: Emily Fox, University of Washington

L2 vs L1: Housing Price Example

Predict housing price from several features

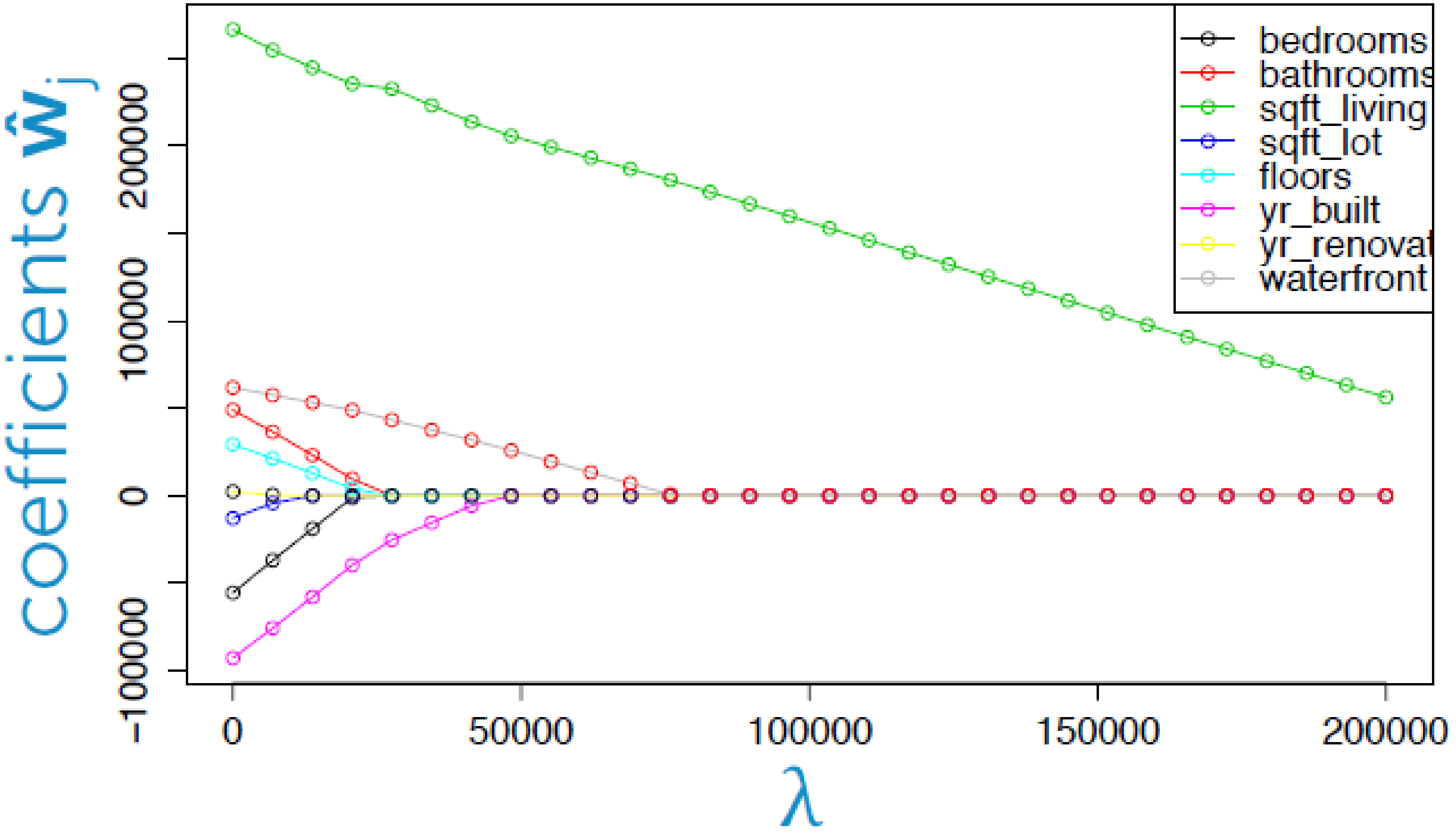


Figure: Emily Fox, University of Washington

Regularization as MAP

- L1 and L2 regularization can be interpreted as **maximum a-posteriori (MAP) estimation** of the parameters
- To be discussed later in the course…

Slide credit: CMU MLD Matt Gormley

Takeaways

1. **Nonlinear basis functions** allow **linear models** (e.g. Linear Regression, Logistic Regression) to capture **nonlinear** aspects of the original input
2. Nonlinear features **require no changes to the model** (i.e. just preprocessing)
3. **Regularization** helps to avoid **overfitting**
4. **(Regularization** and **MAP estimation** are equivalent for appropriately chosen priors)

Feature Engineering / Regularization Objectives

You should be able to…

- Engineer appropriate features for a new task
- Use feature selection techniques to identify and remove irrelevant features
- Identify when a model is overfitting
- Add a regularizer to an existing objective in order to combat overfitting
- Convert linearly inseparable dataset to a linearly separable dataset in higher dimensions
- Describe feature engineering in common application areas

“Dollar fifty in late charges at the public library”

https://www.youtube.com/watch?v=e1DnltskkWk

Breakout rooms: Why is a CMU course better than a library card?

Introduction to Machine Learning

Neural Networks

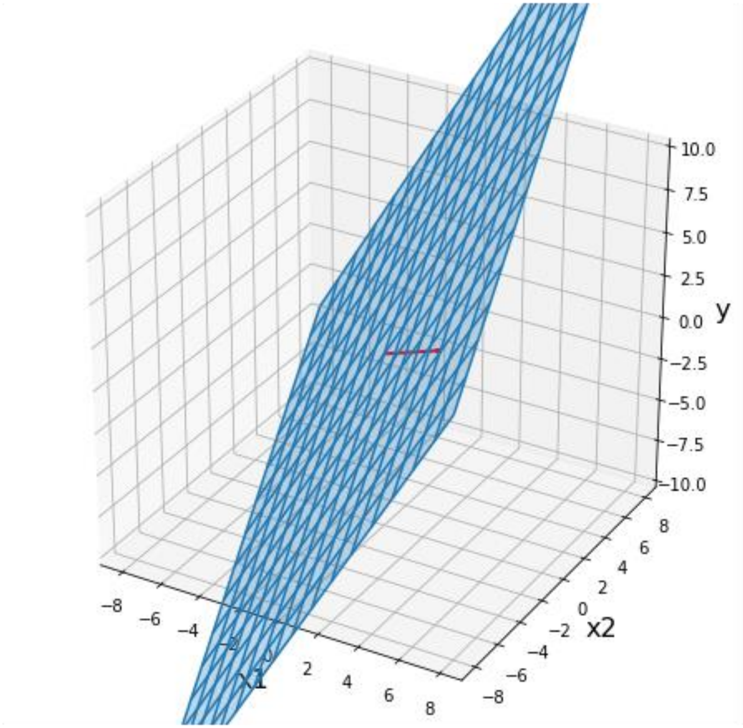
Instructor: Pat Virtue

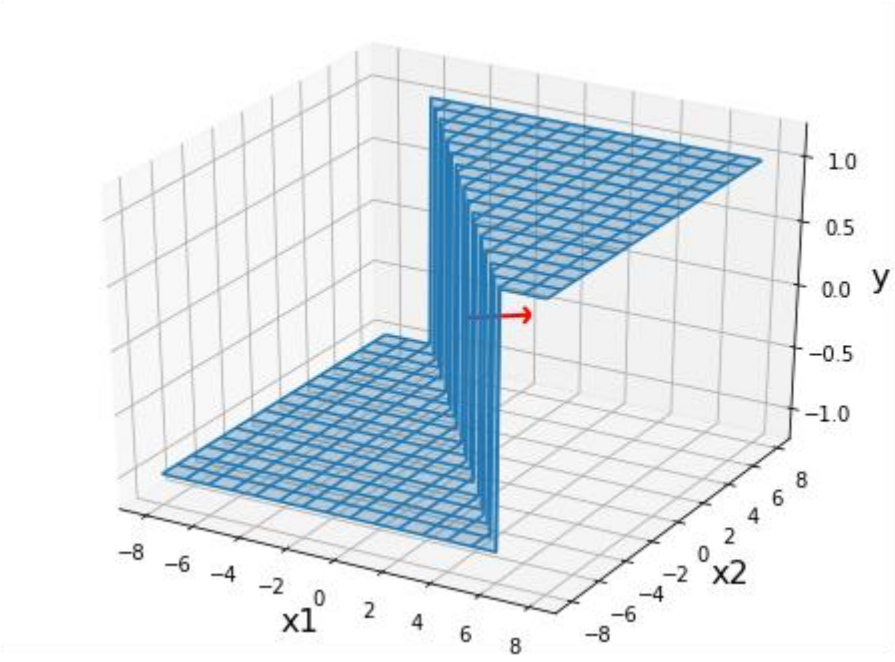
Perceptron

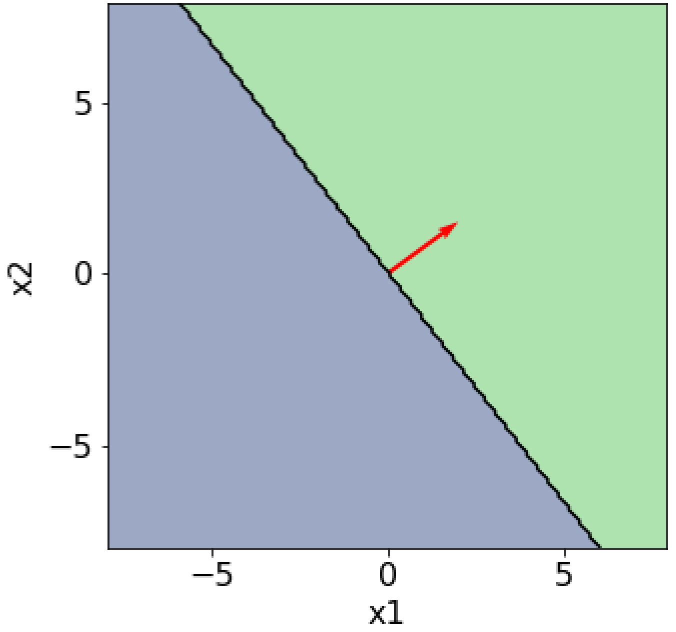
$$sign(\boldsymbol{z}) = \begin{cases} 1, & if\ z \geq 0 \\ -1, & if\ z < 0 \end{cases}$$

Classification: Hard threshold on linear model

$$h(\boldsymbol{x}) = sign\left(\boldsymbol{w}^T \boldsymbol{x} + b\right)$$







Piazza Poll 2

$$h(\boldsymbol{x}) = sign(\boldsymbol{w}^T\boldsymbol{x} + b)$$

Which of the following perceptron parameters will perfectly classify this data?

$$sign(\boldsymbol{z}) = \begin{cases} 1, & if\ z \geq 0 \\ -1, & if\ z < 0 \end{cases}$$

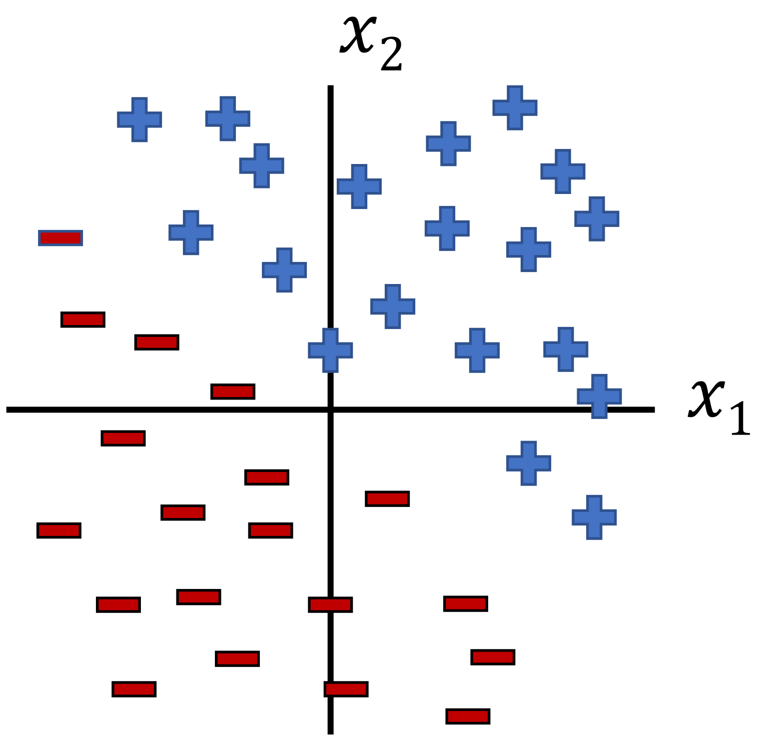
A. $w = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b = 0$

B. $w = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, b = 0$

C. $w = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, b = 0$

D. $w = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, b = 0$

E. None of the above



Piazza Poll 3

$$h(\boldsymbol{x}) = sign(\boldsymbol{w}^T\boldsymbol{x} + b)$$

Which of the following perceptron parameters will perfectly classify this data?

$$sign(\boldsymbol{z}) = \begin{cases} 1, & if\ z \geq 0 \\ -1, & if\ z < 0 \end{cases}$$

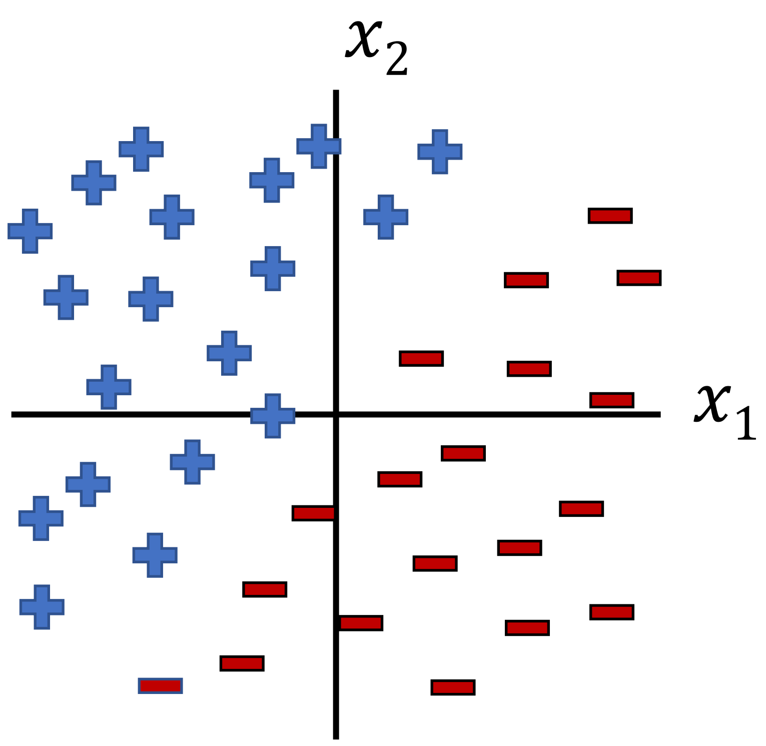
A. $w = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b = 0$

B. $w = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, b = 0$

C. $w = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, b = 0$

D. $w = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, b = 0$

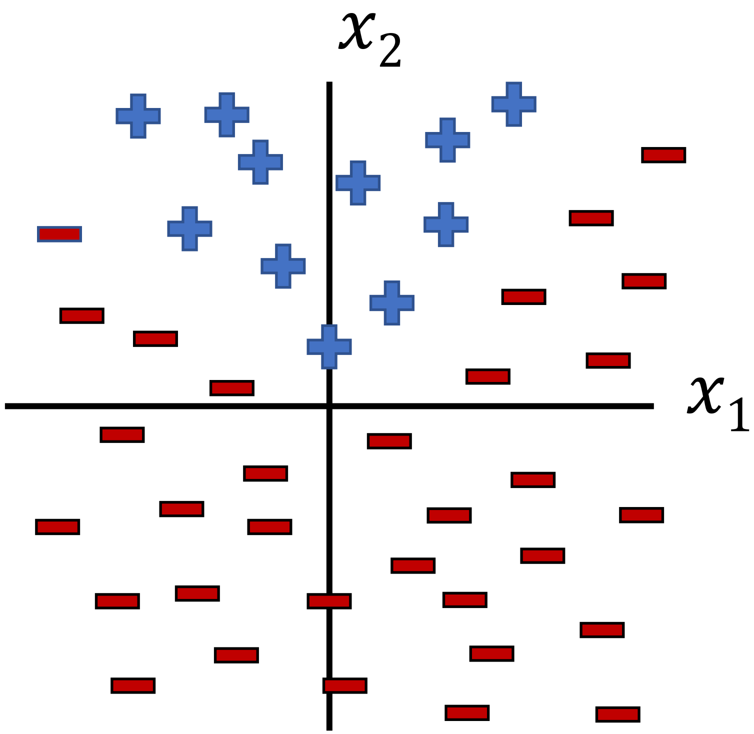
E. None of the above



Classification Design Challenge

$h_A(\boldsymbol{x}) = sign(\boldsymbol{w}_A^T \boldsymbol{x} + b_A)$
$h_B(\boldsymbol{x}) = sign(\boldsymbol{w}_B^T \boldsymbol{x} + b_B)$
$h_C(\boldsymbol{x}) = sign(\boldsymbol{w}_C^T \boldsymbol{x} + b_C)$

How could you configure three specific perceptrons to classify this data?



Piazza Poll 4

$$h_C(\boldsymbol{z}) = sign\left(\boldsymbol{w}_C^T \boldsymbol{z} + b_C\right)$$

Which of the following parameters of $h_C(\boldsymbol{z})$ will perfectly classify this data?

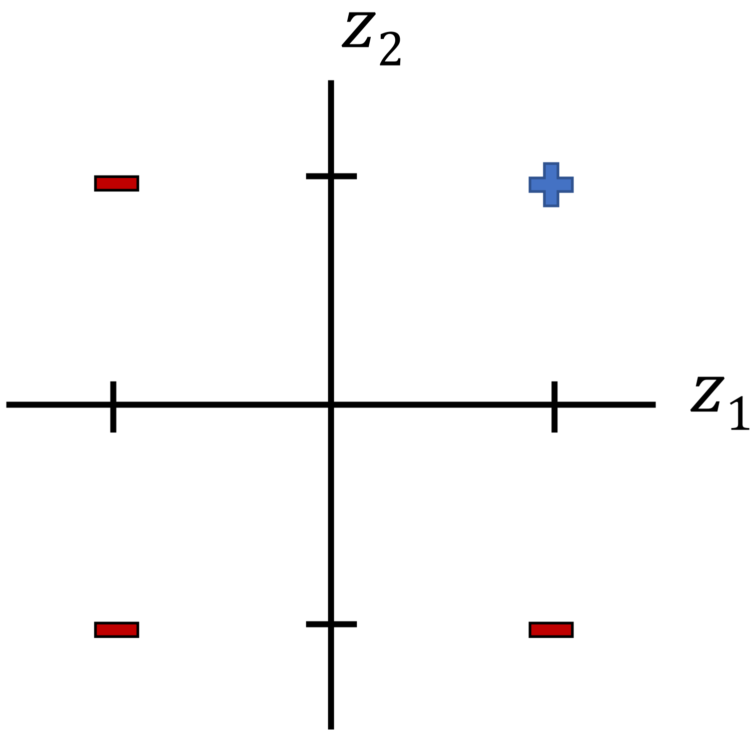
$$sign(\boldsymbol{x}) = \begin{cases} 1, & if\ x \geq 0 \\ -1, & if\ x < 0 \end{cases}$$

A. $w_C = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b_C = 0$

B. $w_C = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b_C = 1$

C. $w_C = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b_C = -1$

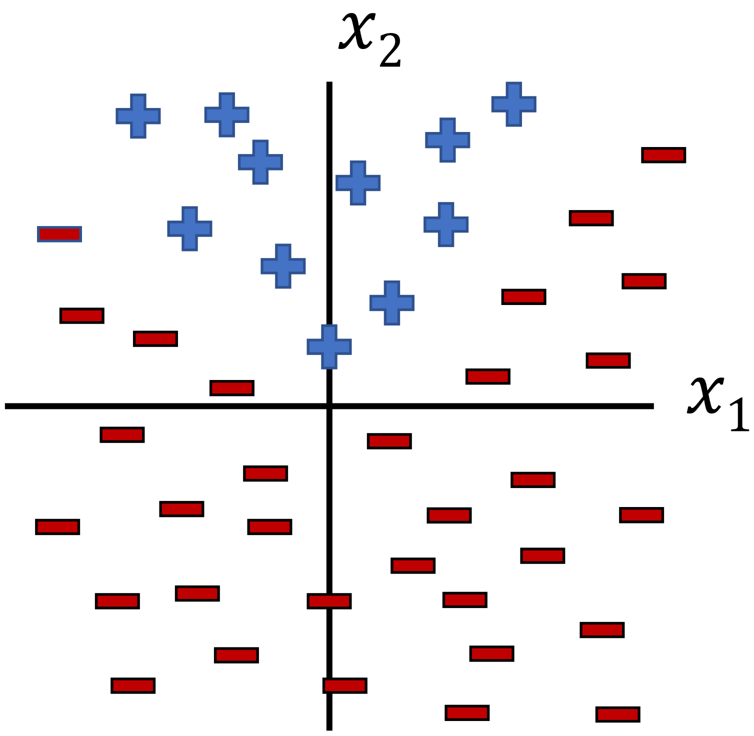
D. None of the above



Classification Design Challenge

$$h_A(\boldsymbol{x}) = sign\left(\boldsymbol{w}_A^T\boldsymbol{x} + b_A\right)$$
$$h_B(\boldsymbol{x}) = sign\left(\boldsymbol{w}_B^T\boldsymbol{x} + b_B\right)$$
$$h_C(\boldsymbol{x}) = sign\left(\boldsymbol{w}_C^T\boldsymbol{x} + b_C\right)$$

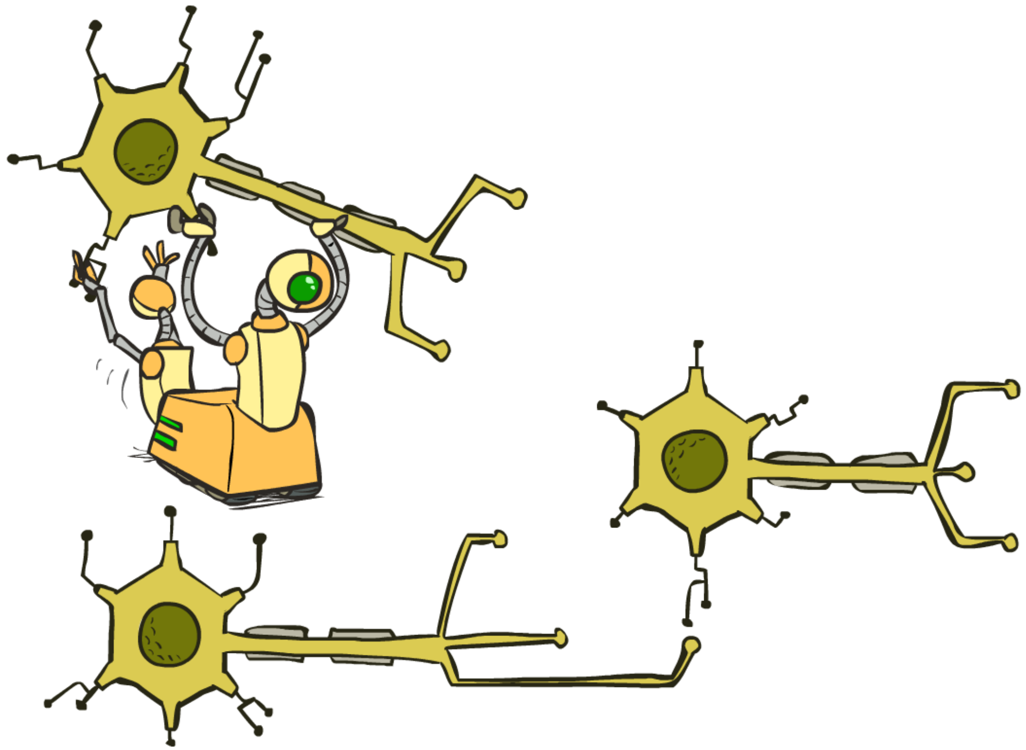
How could you configure three specific perceptrons to classify this data?



Multilayer Perceptrons

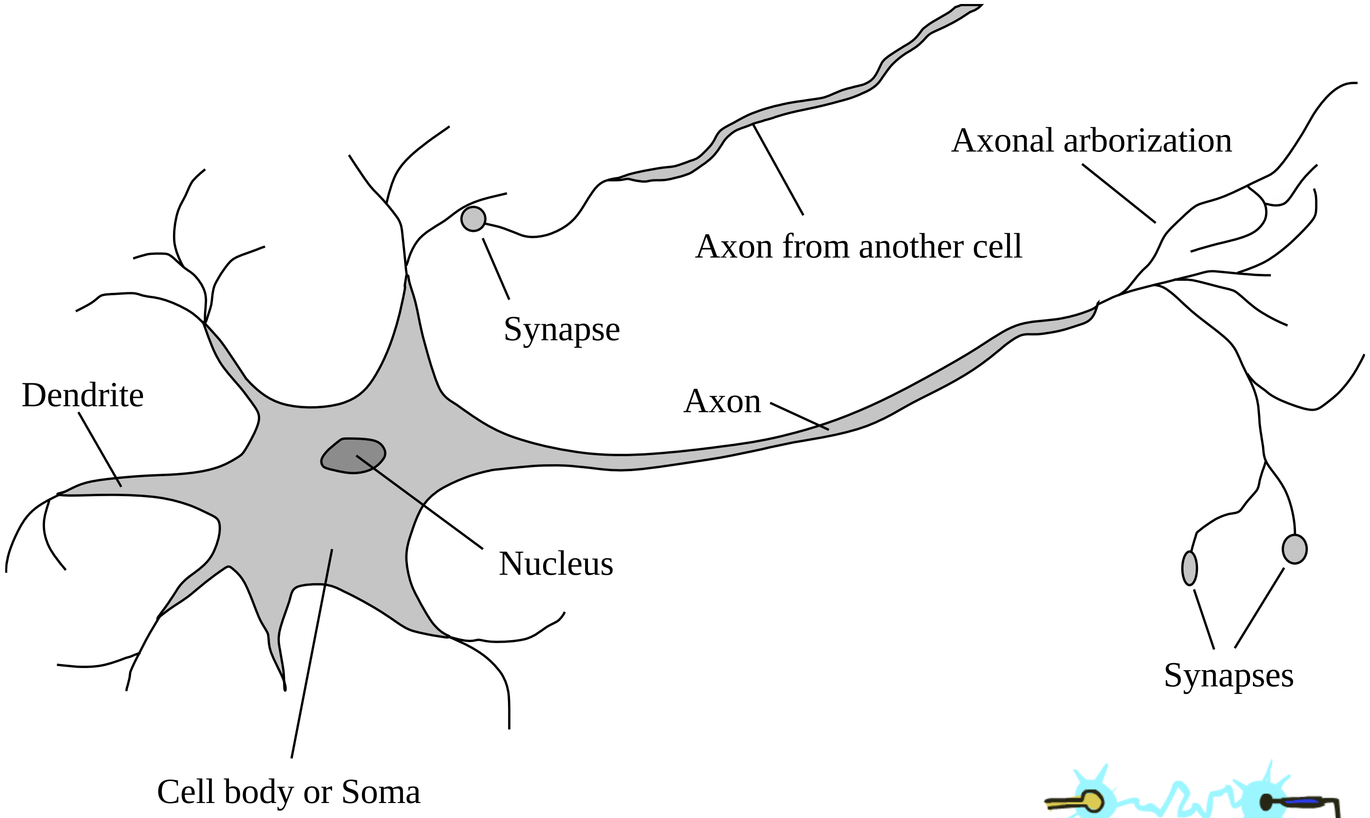
A ***multilayer perceptron*** is a feedforward neural network with at least one ***hidden layer*** (nodes that are neither inputs nor outputs)

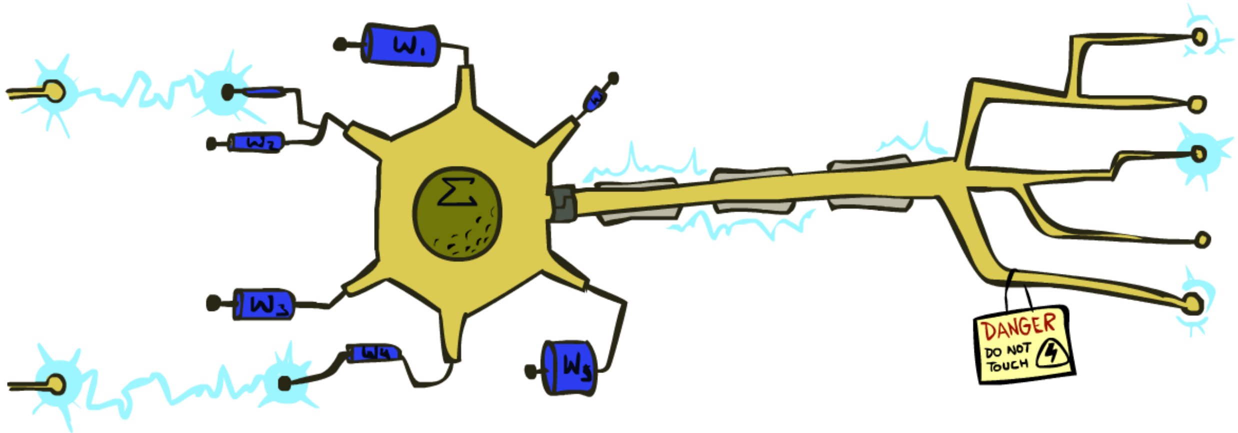
MLPs with enough hidden nodes can represent any function



Slide from UC Berkeley AI

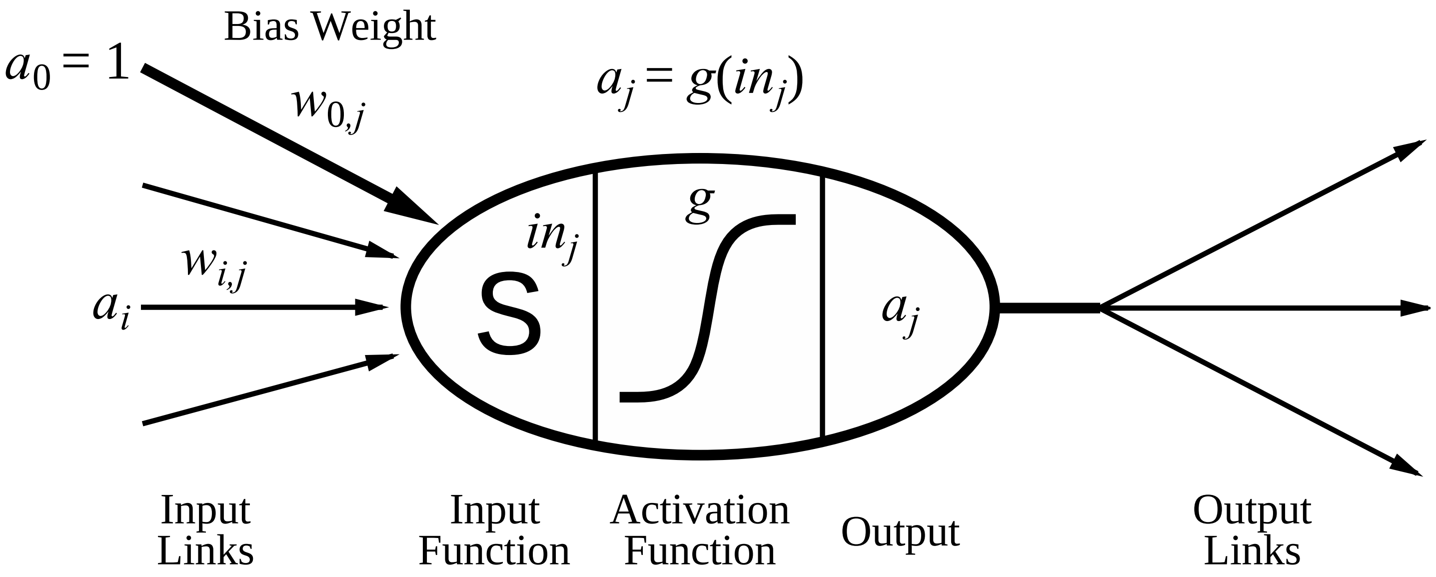
Very Loose Inspiration: Human Neurons





Slide from UC Berkeley AI

Simple Model of a Neuron (McCulloch & Pitts, 1943)



Inputs a_i come from the output of node i to this node j (or from "outside")

Each input link has a ***weight*** $w_{i,j}$

There is an additional fixed input a_0 with ***bias*** weight $w_{0,j}$

The total input is $in_j = \Sigma_i \, w_{i,j} \, a_i$

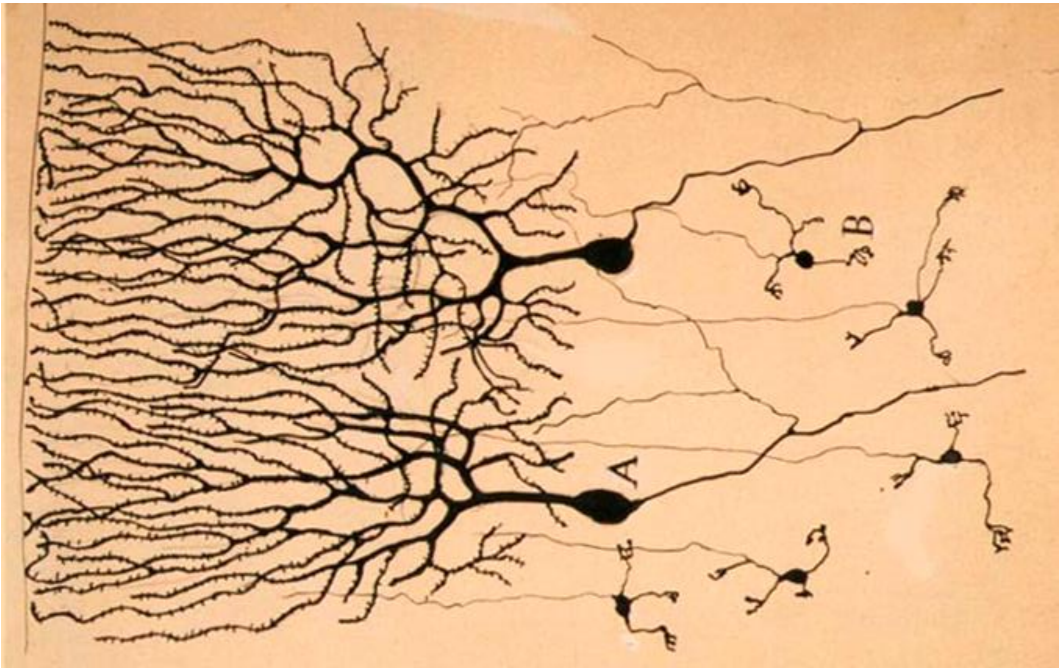
The output is $a_j = g(in_j) = g(\Sigma_i \, w_{i,j} \, a_i) = g(\mathbf{w.a})$

Slide from UC Berkeley AI

Neural Networks

Inspired by actual human brain

Input Signal



Output Signal

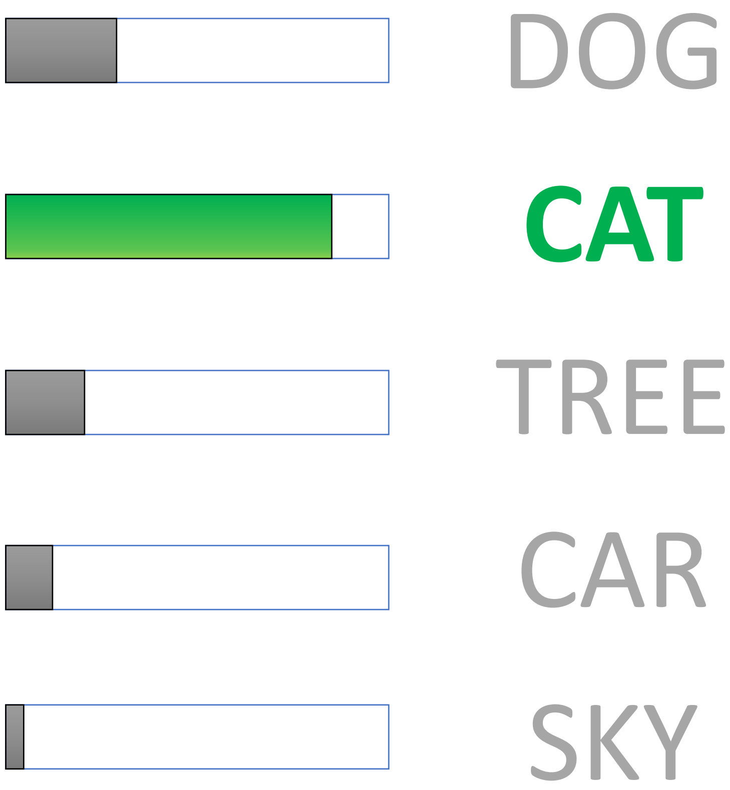


Image: https://en.wikipedia.org/wiki/Neuron