Advanced Introduction to Deep Learning
Lecture 3: Supervised Learning with Linear Models

Han Zhao
han.zhao@cs.cmu.edu
Machine Learning Department, Carnegie Mellon University

July. 22nd, 2019

Lecture 3: Supervised Learning with Linear Models

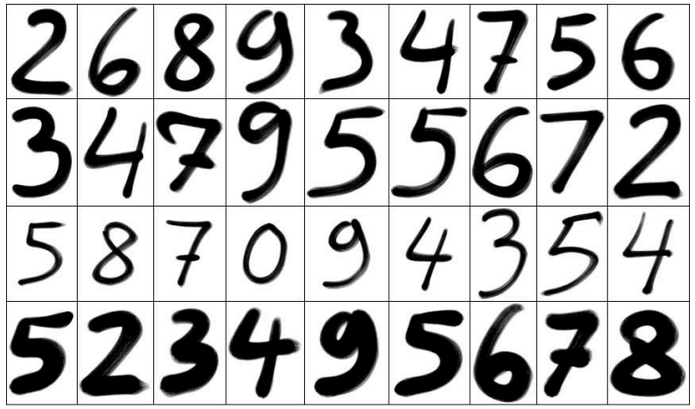
Overview:

- Supervised Learning
- Regression
  - Linear Regression
  - Ridge Regression
  - LASSO
- Classification

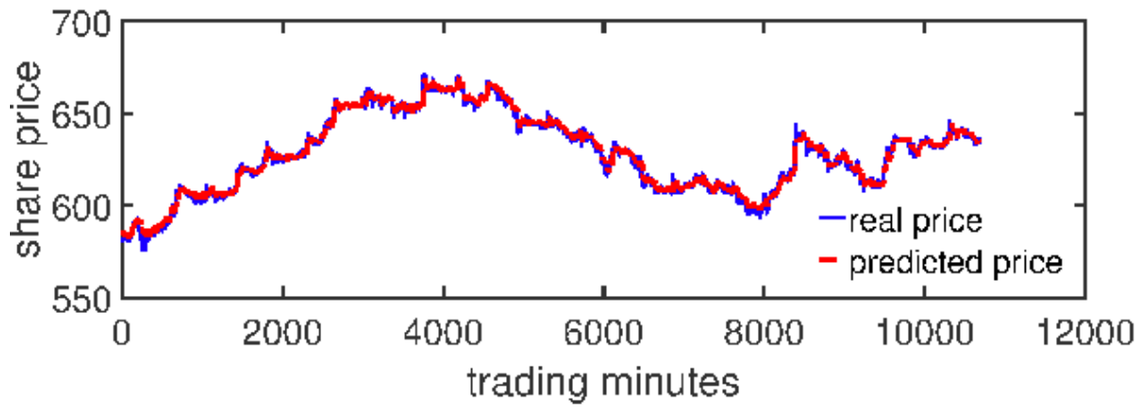
Supervised Learning

Roughly speaking, for supervised learning, there are two kinds of problems

- Classification: the output space is discrete and finite



- Regression: the output space is continuous and infinite



Supervised Learning

Formally, for classification problem, we have the following setting:

- Given input $\mathbf{x}$, the goal is to output a prediction $y = f(\mathbf{x})$
- Typically, we mainly consider the problem of binary classification, that is, $y \in \{0, 1\}$

From a black-box perspective, the whole system works as follows:

- A set of labeled data: $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$
- A parametric model with parameter $\theta \in \Theta$
- Train a classifier based on $\mathcal{D}$ to find the optimal (hopefully) model parameter θ^*
- Use $f_{\theta^*}(\mathbf{x})$ for prediction

Lecture 3: Supervised Learning with Linear Models

Overview:

- Supervised Learning
- Regression
  - Linear Regression
  - Ridge Regression
  - LASSO
- Classification

Supervised Learning — Regression

What is regression? a.k.a., curve fitting

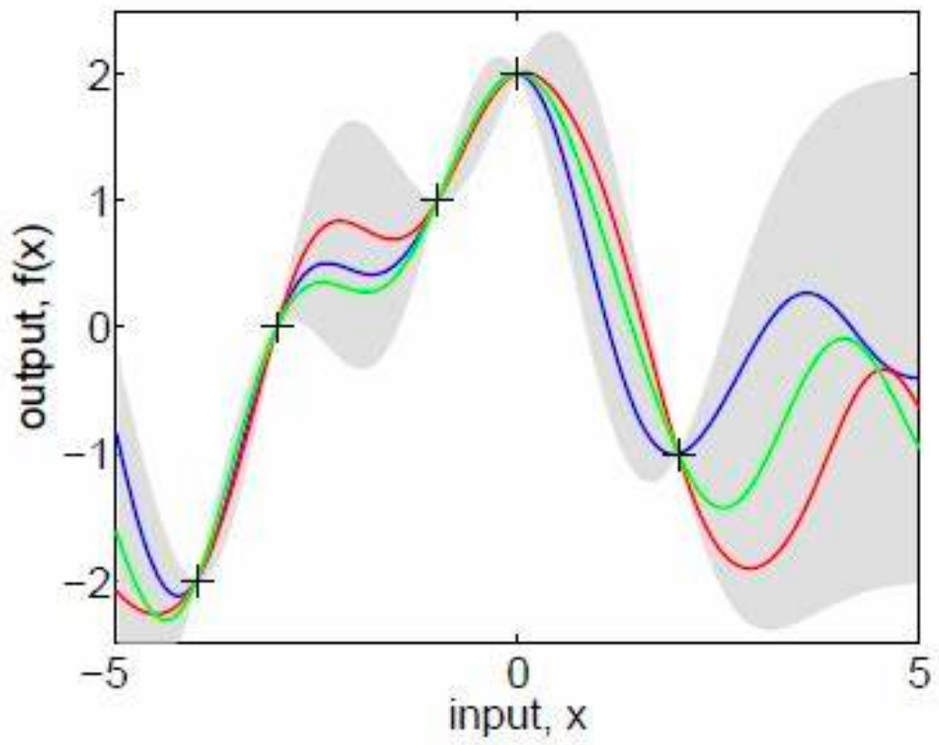
- Note that both the ground truth and the prediction are real numbers
- How to measure the quality of fitness?
  - Recall what we learned from Lecture 1: L_p loss
  - Typically, we use $p = 2$, and this is called mean-squared-error (MSE)

Specifically, the risk of a regression f is:

$$\mathrm{Err}(f) := \mathbb{E}[(Y - f(X))^2]$$

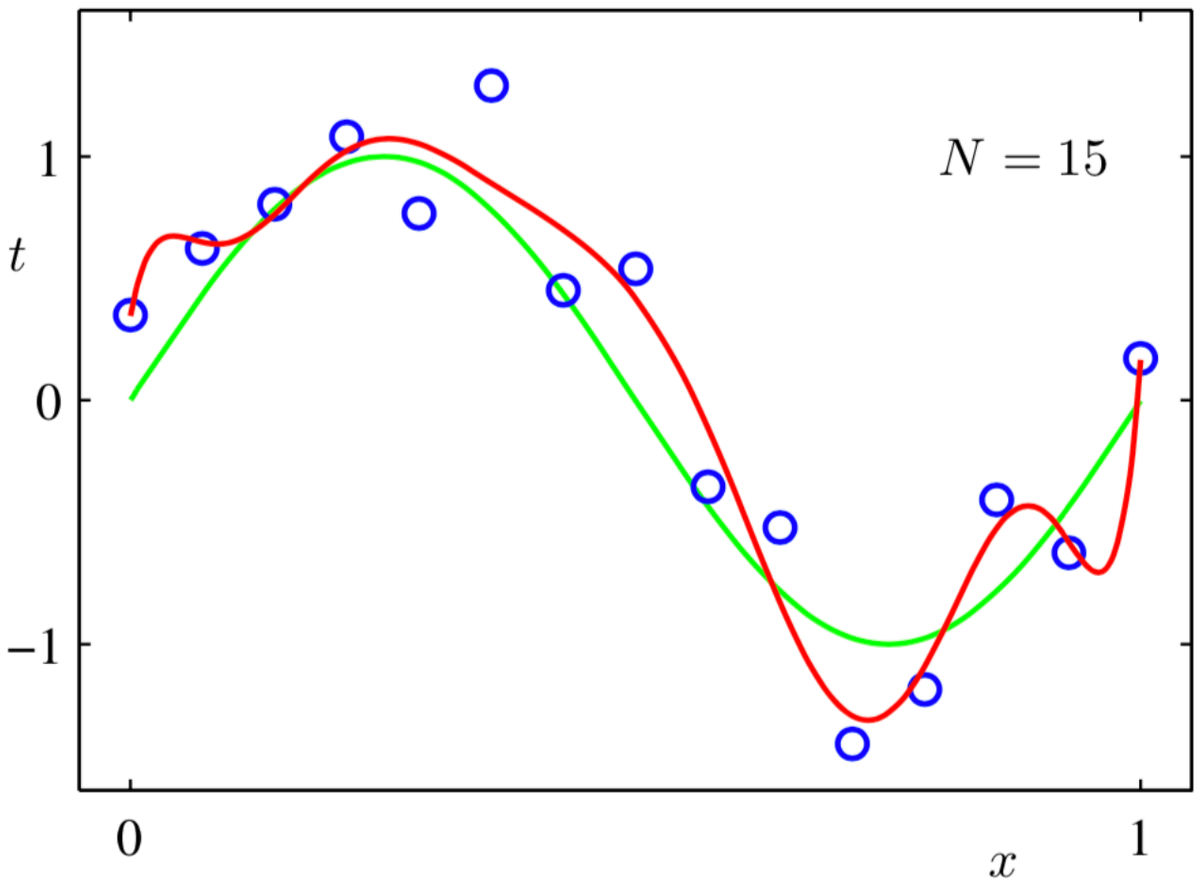
The empirical risk (training error) is:

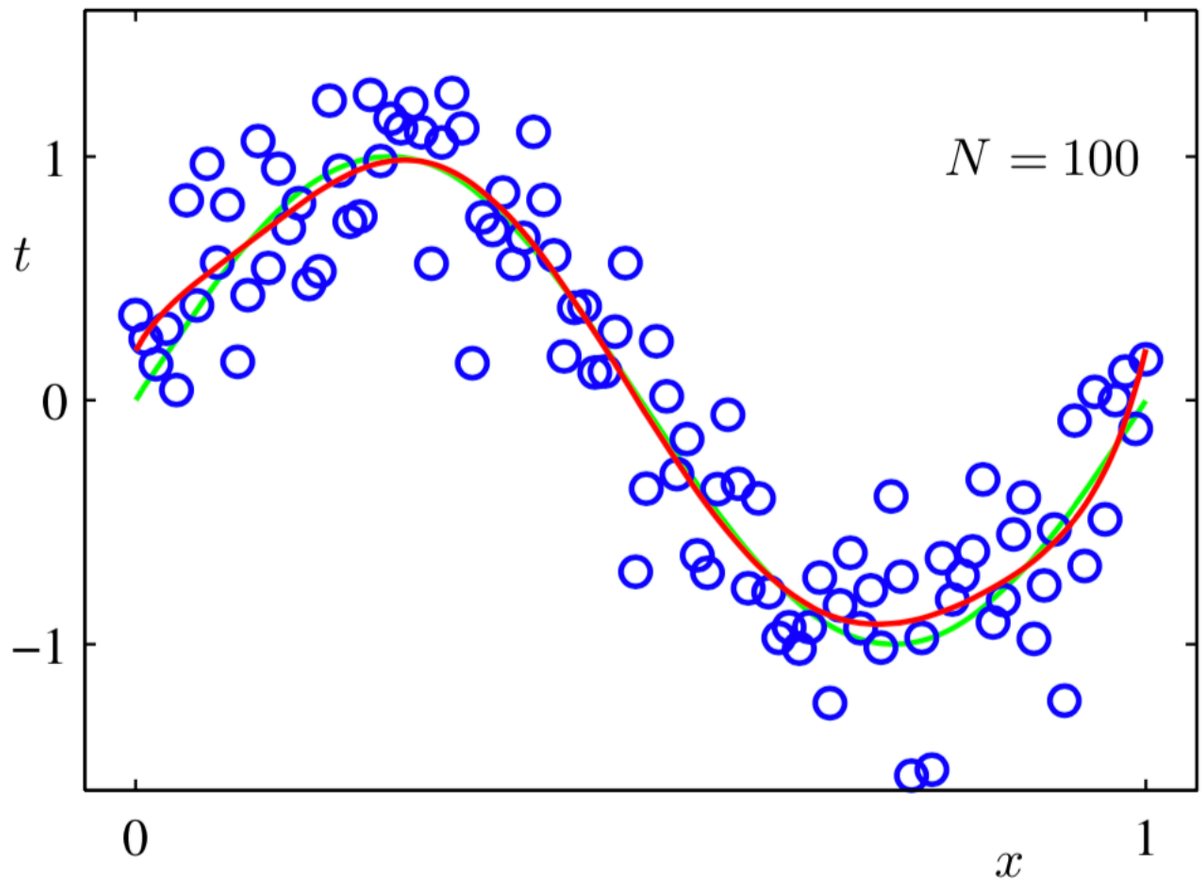
$$\widehat{\mathrm{Err}}(f) := \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2$$



Supervised Learning — Regression

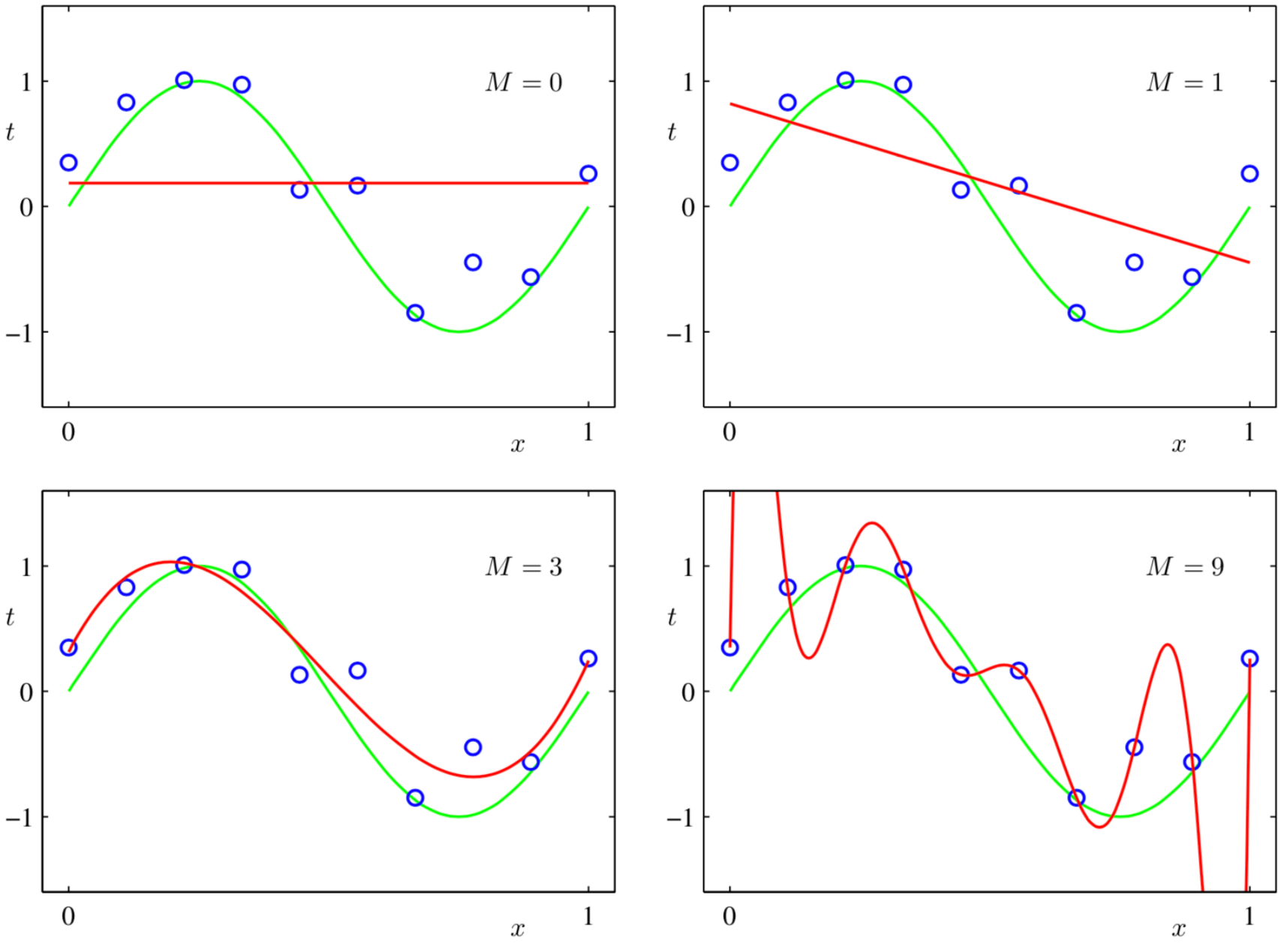
Of course, more training data leads to better curve fitting





Supervised Learning — Regression

And the richer the function class, the better the fit on training data, but not necessarily better fit on test data!



Supervised Learning — Regression

Smaller training error but larger test error? Overfitting

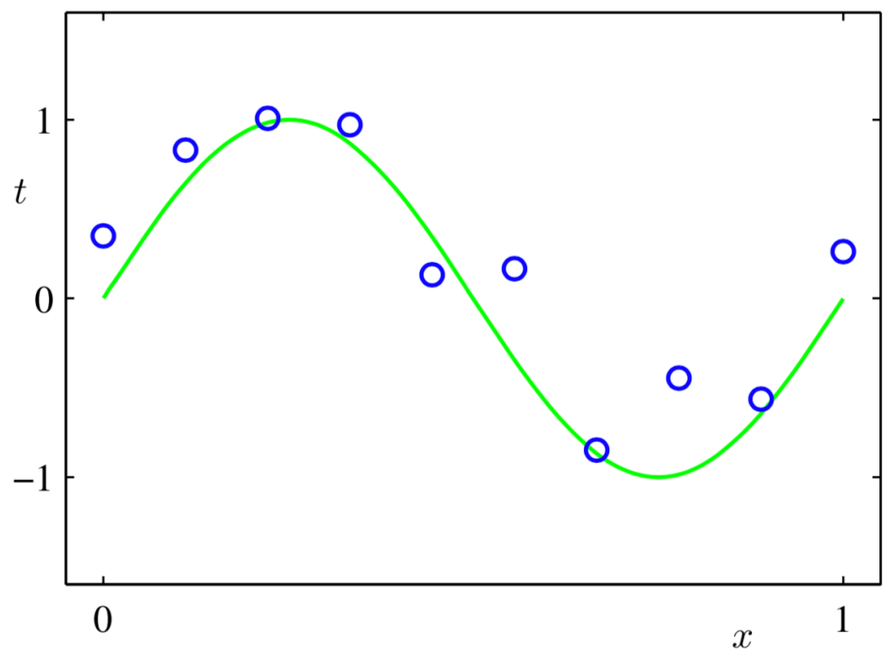
Let's consider a simple 1D polynomial regression to illustrate the phenomenon:

- Underlying function to generate the data: $y = \sin(2\pi x)$
- We don't know about this, so we try to fit the curve using our good friend: polynomial functions

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

- To minimize the training error, we solve:

$$\min_{\mathbf{w}} \quad \frac{1}{n} \sum_{i=1}^{n} (y_i - y(x_i, \mathbf{w}))^2$$



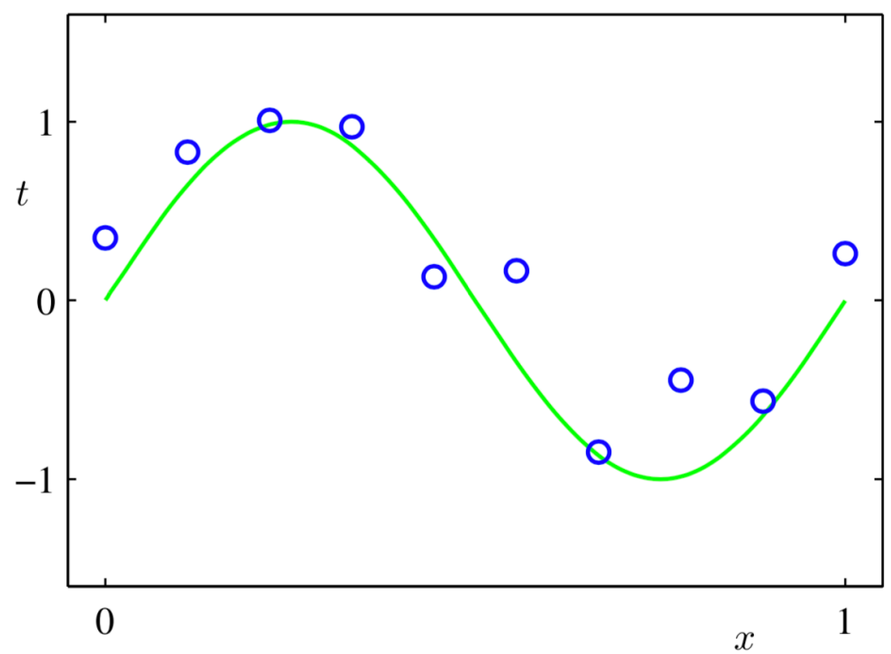
Supervised Learning — Regression

First, how to solve the 1D polynomial fitting problem?

$$\min_{\mathbf{w}} \quad \frac{1}{n}\sum_{i=1}^{n}(y_i - y(x_i, \mathbf{w}))^2$$

Any idea?

Hint: There is a simple solution that we learned in the past two lectures



Supervised Learning — Regression

First, how to solve the 1D polynomial fitting problem?

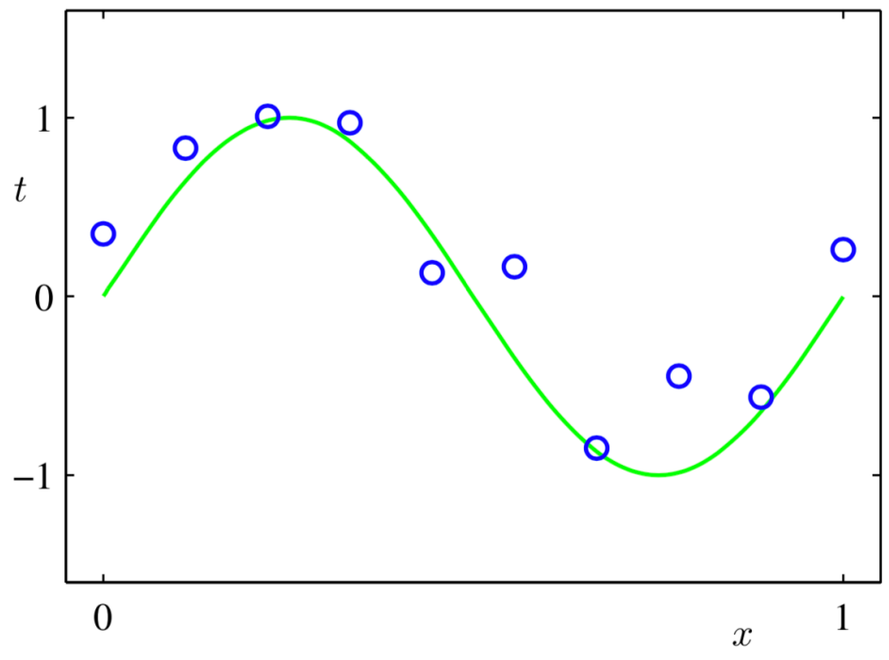
$$\min_{\mathbf{w}} \quad \frac{1}{n}\sum_{i=1}^{n}(y_i - y(x_i, \mathbf{w}))^2$$

Key idea: reduction to linear regression

$$X = \begin{bmatrix} 1 & x_1 & \cdots & x_1^M \\ 1 & x_2 & \cdots & x_2^M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^M \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{bmatrix}$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$\min_{\mathbf{w}} \quad \frac{1}{2n}||y - X\mathbf{w}||_2^2$$

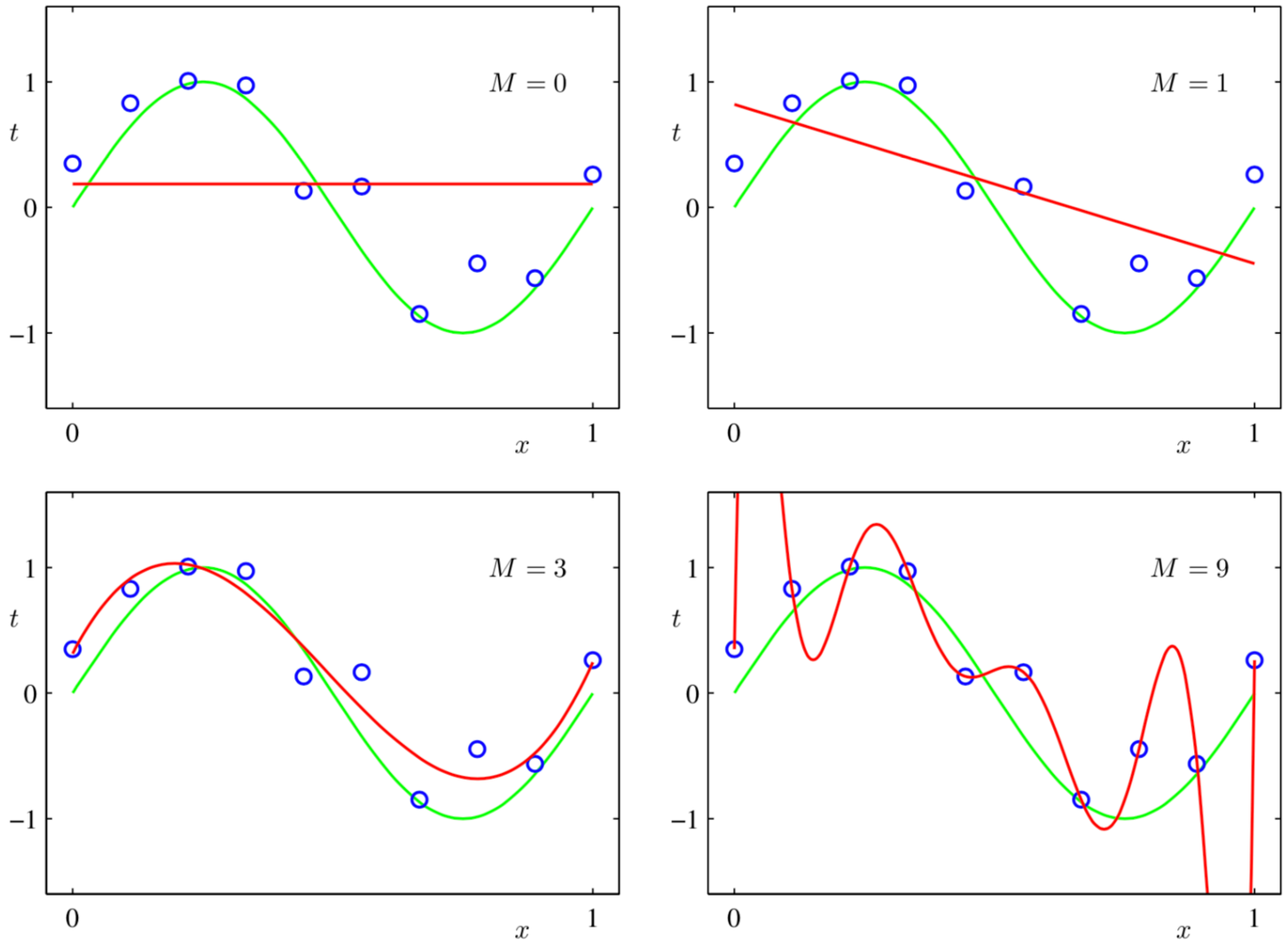


Supervised Learning — Regression

Overfitting: small training error but much larger test error

$$\min_{\mathbf{w}} \quad \frac{1}{n}\sum_{i=1}^{n}(y_i - y(x_i, \mathbf{w}))^2$$

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

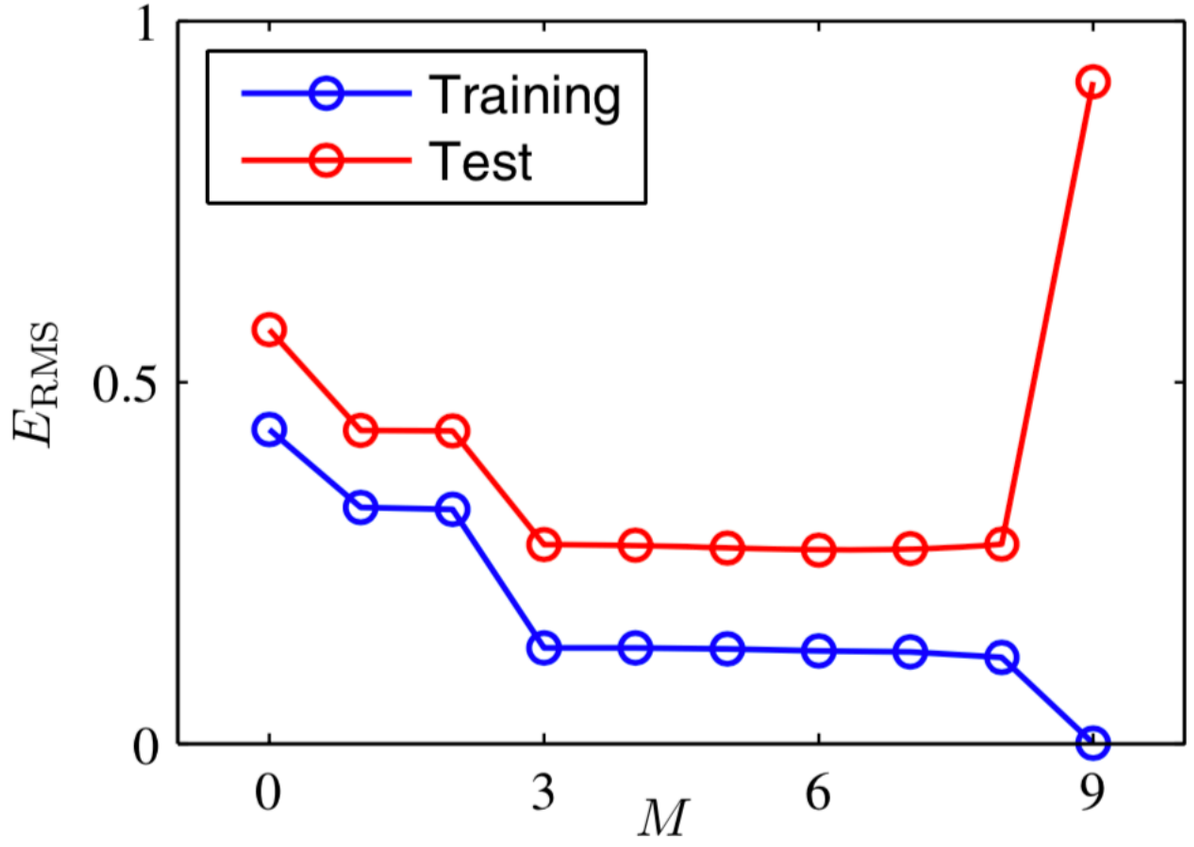


Supervised Learning — Regression

Overfitting: small training error but much larger test error

$$\min_{\mathbf{w}} \quad \frac{1}{n}\sum_{i=1}^{n}(y_i - y(x_i, \mathbf{w}))^2$$

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$



	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Supervised Learning — Regression

OK, now how about solving high-dimensional regression using polynomial fitting similarly?

Supervised Learning — Regression

OK, now how about solving high-dimensional regression using polynomial fitting similarly? (The curse of dimensionality)

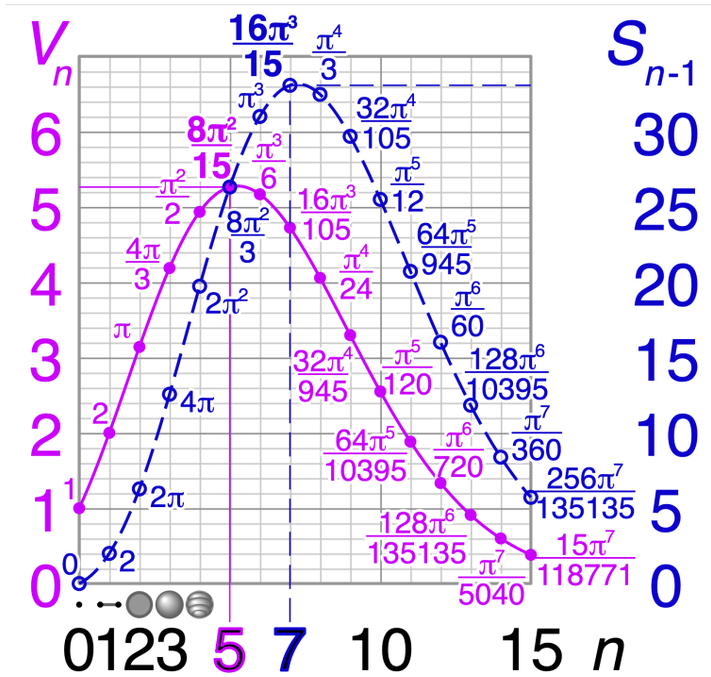
- Dimension explosion — how many parameters do we need to fit in this case?
- The geometry of high-dimensional space are **very different** from low-dimensional space

Supervised Learning — Regression

OK, now how about solving high-dimensional regression using polynomial fitting similarly? (The curse of dimensionality)

- Dimension explosion — how many parameters do we need to fit in this case?
- The geometry of high-dimensional space are **very different** from low-dimensional space
  - Volume are mostly in the surface

$$V_n(R) = \frac{\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}+1\right)} R^n$$



Supervised Learning — Regression

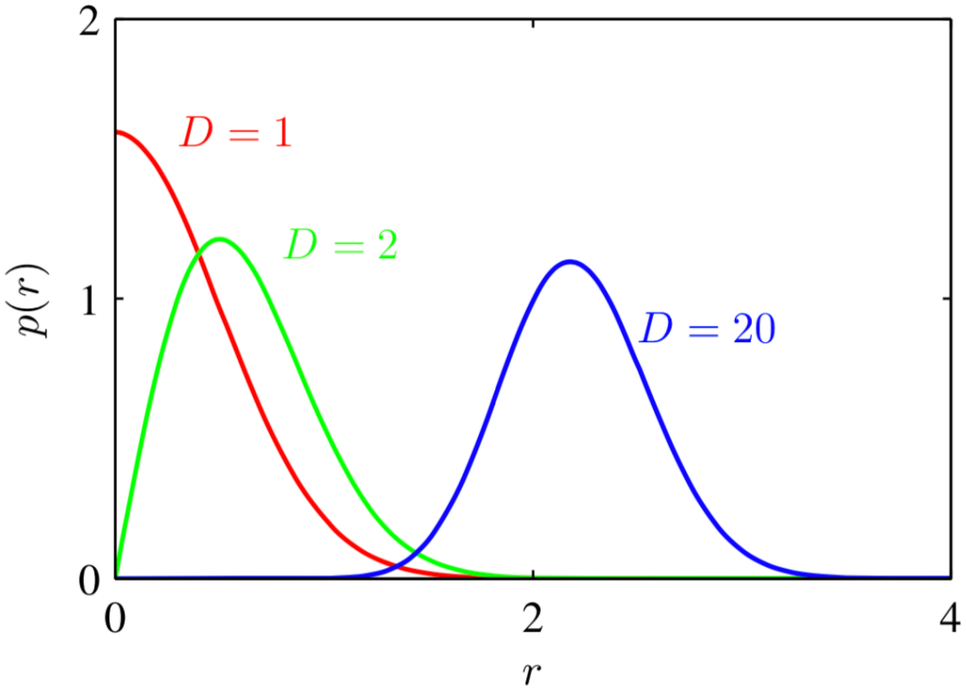
OK, now how about solving high-dimensional regression using polynomial fitting similarly? (The curse of dimensionality)

- Dimension explosion — how many parameters do we need to fit in this case?
- The geometry of high-dimensional space are **very different** from low-dimensional space
  - Vectors are near-orthogonal: randomly draw two points from unit-sphere or unit-ball, then with probability, these two vectors are orthogonal to each other

Supervised Learning — Regression

OK, now how about solving high-dimensional regression using polynomial fitting similarly? (The curse of dimensionality)

- Dimension explosion — how many parameters do we need to fit in this case?
- The geometry of high-dimensional space are **very different** from low-dimensional space
  - Probability mass of Gaussian are mostly in a thin annulus



Lecture 3: Supervised Learning with Linear Models

Overview:

- Supervised Learning
- Regression
  - Linear Regression
  - Ridge Regression
  - LASSO
- Classification

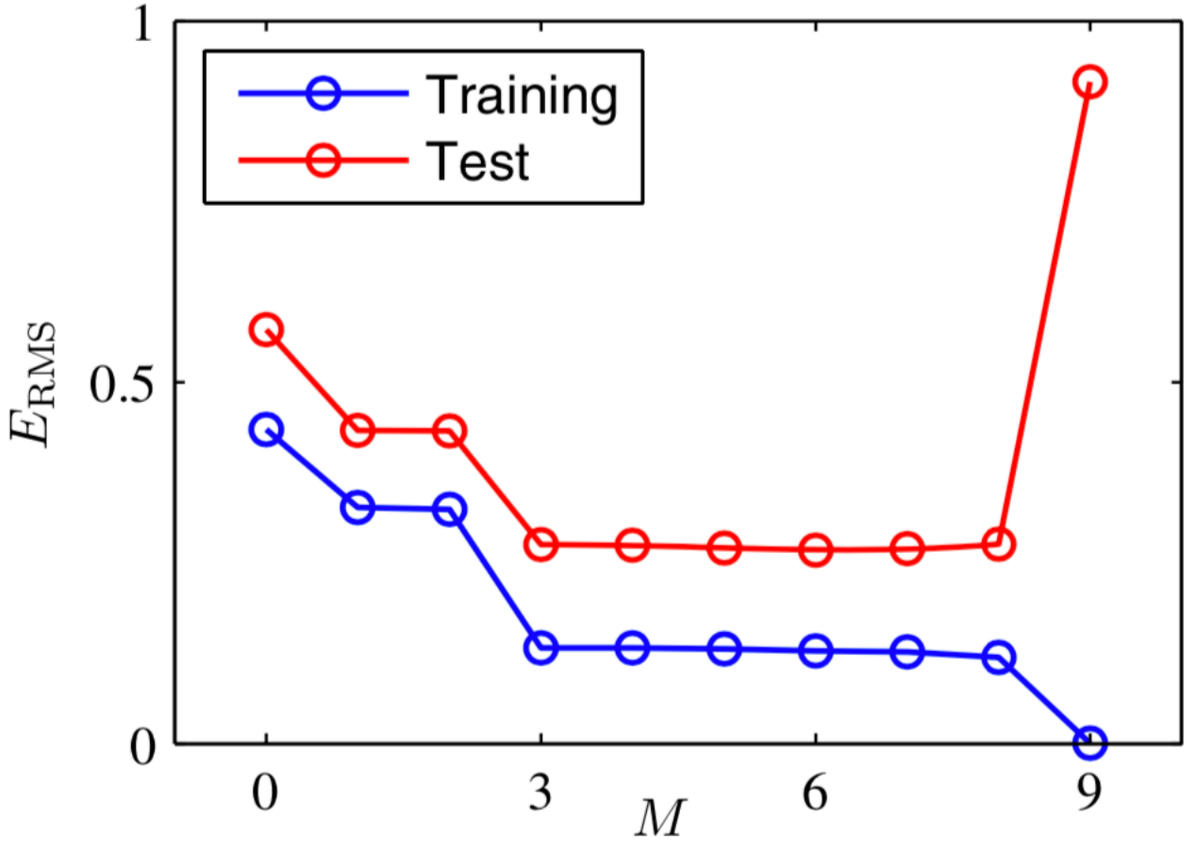
Supervised Learning — Regression

Let's go back to our overfitting problem in Linear Regression:

$$\min_{\mathbf{w}} \quad \frac{1}{n}\sum_{i=1}^{n}(y_i - y(x_i, \mathbf{w}))^2$$

How to mitigate this problem?

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$



	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Supervised Learning — Regression

Let's go back to our overfitting problem in Linear Regression:

$$\min_{\mathbf{w}} \quad \frac{1}{n}\sum_{i=1}^{n}(y_i - y(x_i, \mathbf{w}))^2$$

How to mitigate this problem?

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Since the weight are too large, let's make them smaller by penalizing large weights!

$$\min_{\mathbf{w}} \quad \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - y\left(x_i, \mathbf{w}\right)\right)^2 + \lambda||\mathbf{w}||_2^2$$

This is often called Ridge Regression or L2 regularization

	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Supervised Learning — Regression

Ridge Regression: two equivalent forms

Penalty form:

$$\min_{\mathbf{w}} \quad \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - y\left(x_i, \mathbf{w}\right)\right)^2 + \lambda||\mathbf{w}||_2^2$$

Constraint form:

$$\min_{\mathbf{w}} \quad \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - y\left(x_i, \mathbf{w}\right)\right)^2$$

$$\text{subject to} \quad ||\mathbf{w}||_2 \leq t$$

Claim: when the objective functions are convex, then these two forms are equivalent to each other

Supervised Learning — Regression

Due to the equivalence, we focus on the penalty form

Penalty form in matrix representation:

$$\min_{\mathbf{w}} \quad \frac{1}{2n}||X\mathbf{w} - y||_2^2 + \lambda||\mathbf{w}||_2^2$$

First, how to solve it? Recall your HW programming question yesterday :)

Supervised Learning — Regression

Due to the equivalence, we focus on the penalty form

Penalty form in matrix representation:

$$\min_{\mathbf{w}} \quad \frac{1}{2n}||X\mathbf{w} - y||_2^2 + \lambda||\mathbf{w}||_2^2$$

First, how to solve it?

Closed form solution:

$$\mathbf{w}^* = \frac{1}{n}(\frac{1}{n}X^TX + 2\lambda I)^{-1}X^Ty$$

Think: use the multivariate calculus rule in Lecture 1 to derive this solution

Supervised Learning — Regression

Due to the equivalence, we focus on the penalty form

Penalty form in matrix representation:

$$\min_{\mathbf{w}} \quad \frac{1}{2n}||X\mathbf{w} - y||_2^2 + \lambda||\mathbf{w}||_2^2$$

First, how to solve it?

Closed form solution:

$$\mathbf{w}^* = \frac{1}{n}(\frac{1}{n}X^TX + 2\lambda I)^{-1}X^Ty$$

Now compare with the Least-Square solution, what's the difference?

$$\mathbf{w}^* = \frac{1}{n}(\frac{1}{n}X^TX)^{-1}X^Ty$$

Supervised Learning — Regression

Now compare with the Least-Square solution, what's the difference?

Ridge Regression

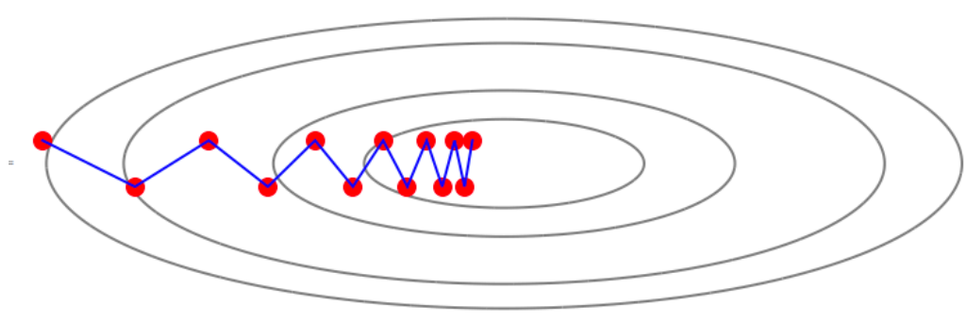
$$\mathbf{w}^* = \frac{1}{n}(\frac{1}{n}X^TX + 2\lambda I)^{-1}X^Ty$$

Ordinary Least Square

$$\mathbf{w}^* = \frac{1}{n}(\frac{1}{n}X^TX)^{-1}X^Ty$$

- Ridge regression is strongly convex, in the sense that the solution is unique (This is not the case for OLS, recall your observation from HW2)
- Ridge regression is more numerically stable (the condition number is better, so it's more friendly to gradient descent
- It can be shown that the error given by Ridge Regression is better in the population level $(\exists\lambda)$
- There is a tradeoff in the value of λ

(HW: derive the implement the GD version of Ridge Regression)



Supervised Learning — Regression

Now compare with the Least-Square solution, what's the difference?

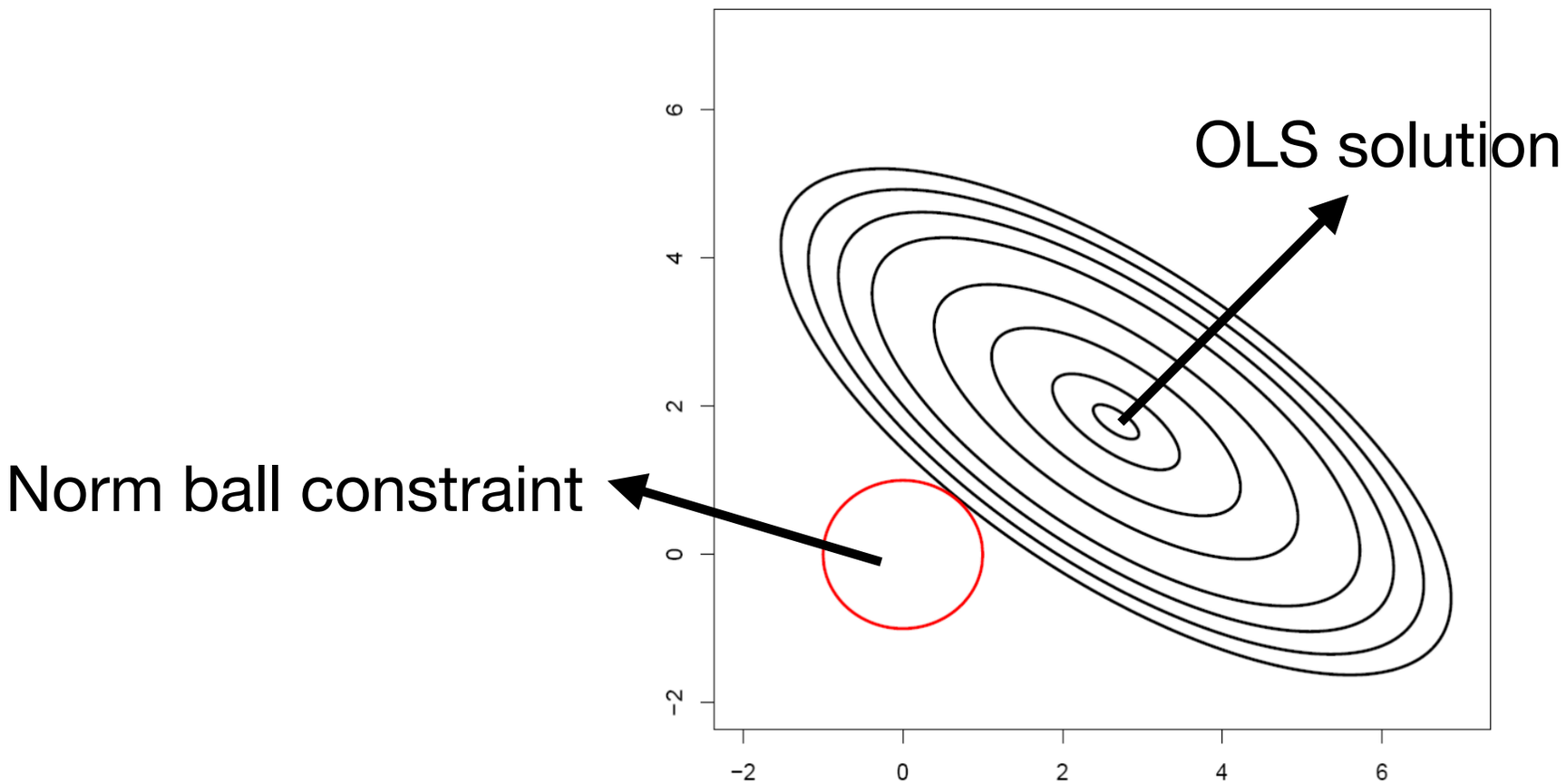
Ridge Regression

$$\mathbf{w}^* = \frac{1}{n}(\frac{1}{n}X^TX + 2\lambda I)^{-1}X^Ty$$

Ordinary Least Square

$$\mathbf{w}^* = \frac{1}{n}(\frac{1}{n}X^TX)^{-1}X^Ty$$

Geometrically, what's the effect of L2 penalty?



Supervised Learning — Regression

Bayesian interpretation of Ordinary Least Square and Ridge Regression

Ordinary Least Square:

Recall from Lecture 1 we learned the Bayes them:

$$p(\mathbf{w} \mid \mathcal{D}) \propto p(\mathbf{w}) \cdot p(\mathcal{D} \mid \mathbf{w})$$

Prior distribution

Posterior distribution

Likelihood function

Supervised Learning — Regression

Bayesian interpretation of Ordinary Least Square and Ridge Regression

Ordinary Least Square:

Recall from Lecture 1 we learned the Bayes them:

$$p(\mathbf{w} \mid \mathcal{D}) \propto p(\mathbf{w}) \cdot p(\mathcal{D} \mid \mathbf{w}) \iff \log p(\mathbf{w} \mid \mathcal{D}) = \log p(\mathbf{w}) + \log p(\mathcal{D} \mid \mathbf{w})$$

We first consider the principle of Maximum-likelihood Estimation (MLE):

- We would like to maximize the probability of seeing the data under our model assumption, hence:

$$\max_{\mathbf{w}} \quad \log p(\mathcal{D} \mid \mathbf{w})$$

- For OLS, the underlying assumption is:

$$y_i = \mathbf{w}^T \mathbf{x}_i + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, I)$$

- Now we can show that maximum MLE = minimum L2 loss

Supervised Learning — Regression

Bayesian interpretation of Ordinary Least Square and Ridge Regression

Ridge Regression:

Recall from Lecture 1 we learned the Bayes them:

$$p(\mathbf{w} \mid \mathcal{D}) \propto p(\mathbf{w}) \cdot p(\mathcal{D} \mid \mathbf{w}) \iff \log p(\mathbf{w} \mid \mathcal{D}) = \log p(\mathbf{w}) + \log p(\mathcal{D} \mid \mathbf{w})$$

Now we consider the principle of Maximum-A-Posteriori (MAP):

- We would like to maximize the posterior probability of model parameter given that we have seen the data

$$\max_{\mathbf{w}} \quad \log p(\mathbf{w}) + \log p(\mathcal{D} \mid \mathbf{w})$$

- For Ridge Regression, the underlying assumption is:

$$y_i = \mathbf{w}^T \mathbf{x}_i + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, I) \quad \mathbf{w} \sim \mathcal{N}(0, \sigma^2 I)$$

- maximum MAP = minimum L2 loss and L2 regularization

Lecture 3: Supervised Learning with Linear Models

Overview:

- Supervised Learning
- Regression
  - Linear Regression
  - Ridge Regression
  - LASSO
- Classification

Supervised Learning — Regression

Now how about using L1 regularization instead of L2?

LASSO:

$$\min_{\mathbf{w}} \quad \frac{1}{2n}\|X\mathbf{w} - y\|_2^2 + \lambda\|\mathbf{w}\|_1$$

- Note: L1 regularization also helps to shrink the model parameter

Question: How to solve it? Is there a closed form solution?

	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Supervised Learning — Regression

Now how about using L1 regularization instead of L2?

LASSO:

$$\min_{\mathbf{w}} \quad \frac{1}{2n}\|X\mathbf{w} - y\|_2^2 + \lambda\|\mathbf{w}\|_1$$

- Note: L1 regularization also helps to shrink the model parameter

Question: How to solve it? Is there a closed form solution?

No, unfortunately

Then how should we solve it?

Again, iterative method, but this time we consider the constrained formulation

Supervised Learning — Regression

Now how about using L1 regularization instead of L2?

LASSO:

$$\min_{\mathbf{w}} \quad \frac{1}{2n}\|X\mathbf{w} - y\|_2^2 + \lambda\|\mathbf{w}\|_1$$

- Note: L1 regularization also helps to shrink the model parameter

Question: How to solve it? Is there a closed form solution?

No, unfortunately

Then how should we solve it?

Again, iterative method, but this time we consider the constrained formulation

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - y\left(x_i, \mathbf{w}\right)\right)^2 \\ \text{subject to} \quad & \|\mathbf{w}\|_1 \leq t \end{aligned}$$

Supervised Learning — Regression

Now how about using L1 regularization instead of L2?

LASSO:

$$\min_{\mathbf{w}} \quad \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - y\left(x_i, \mathbf{w}\right)\right)^2$$

$$\text{subject to} \quad \|\mathbf{w}\|_1 \leq t$$

Algorithm: Projected Gradient Descent

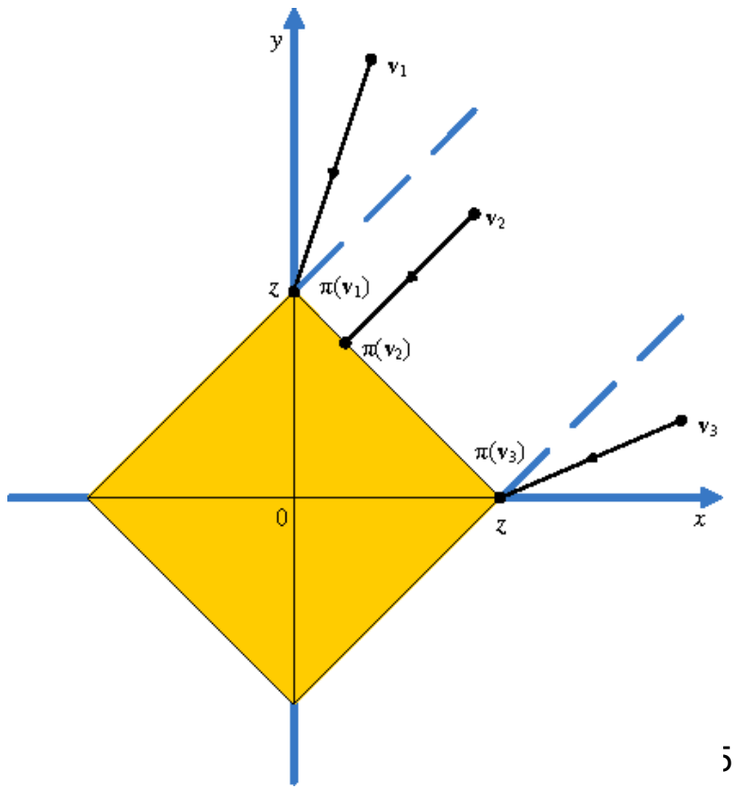
Main idea: gradient descent, but project the update vector back to the feasible region after each iteration

To project a vector back to L1 ball, we essentially solve the following 1D problem:

$$\min_{\mathbf{y}} \quad ||\mathbf{y} - \mathbf{x}||_2$$

$$\text{subject to} \quad ||\mathbf{y}||_1 \leq t$$

There exists an O(d log d) algorithm to do so



Supervised Learning — Regression

Now how about using L1 regularization instead of L2?

LASSO:

$$\min_{\mathbf{w}} \quad \frac{1}{2n}\|X\mathbf{w} - y\|_2^2 + \lambda\|\mathbf{w}\|_1$$

Or alternative algorithm: Subgradient Descent

Main idea: exactly as gradient descent, but replace gradient with subgradient

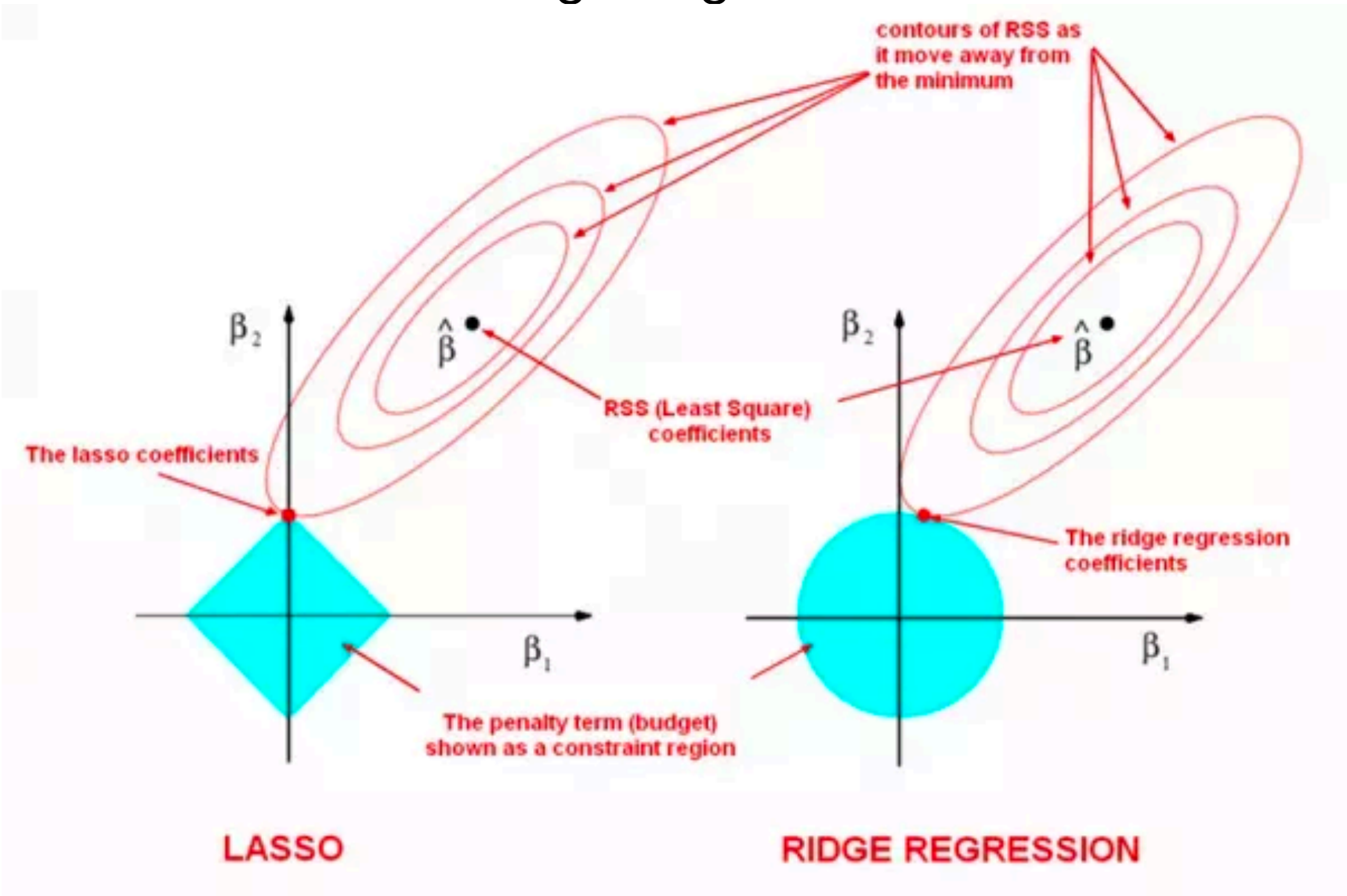
$$\nabla f(\mathbf{w}) = \frac{1}{n}X^T(X\mathbf{w} - y) + \lambda \cdot \mathrm{sgn}(\mathbf{w})$$

where

$$\mathrm{sgn}(t) := \mathbb{I}(t \geq 0) - \mathbb{I}(t < 0) \in \{+1, -1\}$$

Supervised Learning — Regression

Geometric intuition between Ridge Regression and LASSO



LASSO (L1 regularization) leads to sparse model parameters!

HW: show the Bayesian interpretation of LASSO as MAP with Laplace distribution

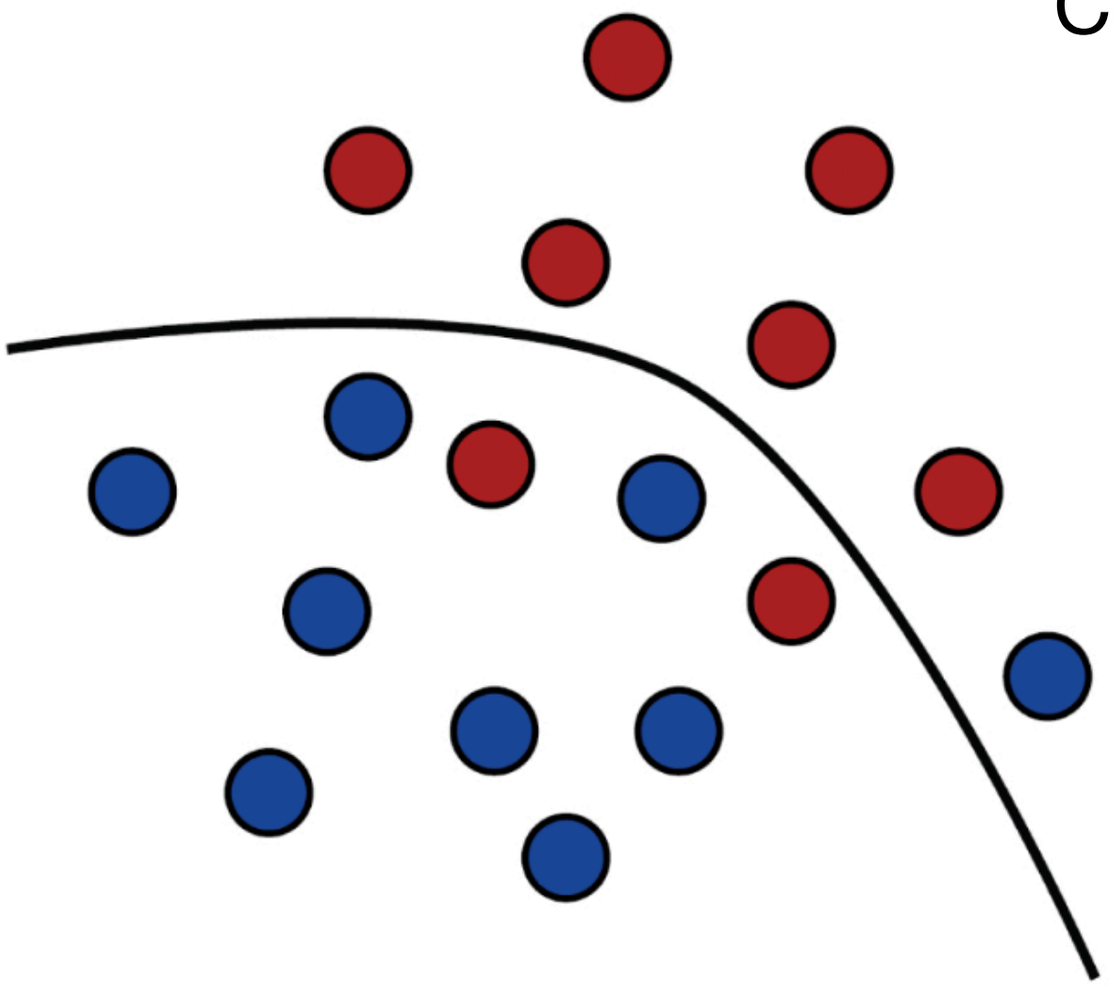
Lecture 3: Supervised Learning with Linear Models

Overview:

- Supervised Learning
- Regression
  - Linear Regression
  - Ridge Regression
  - LASSO
- Classification

Supervised Learning — Classification

Given a binary classifier, how should we measure its performance?



Confusion matrix:

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Supervised Learning — Classification

Confusion matrix:

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

- Accuracy: TN + TP / (TN + FP + FN + TP)
- Precision: TP / (TP + FP)
- Recall: TP / (FN + TP)
- F1-measure: 2 / (1 / Precision + 1 / Recall)

- True Positive Rate: TP / (FN + TP)
- True Negative Rate: TN / (TN + FP)
- Type-I Error (False Positive Rate): 1 - TPR
- Type-II Error (False Negative Rate): 1 - TNR

Supervised Learning — Classification

In most cases we focus on Accuracy as a metric for classification. But this is not always desired, why?

- Imbalanced data
- Instead, use Precision/Recall:
  - Precision: TP / (TP + FP)
  - Recall: TP / (FN + TP)

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

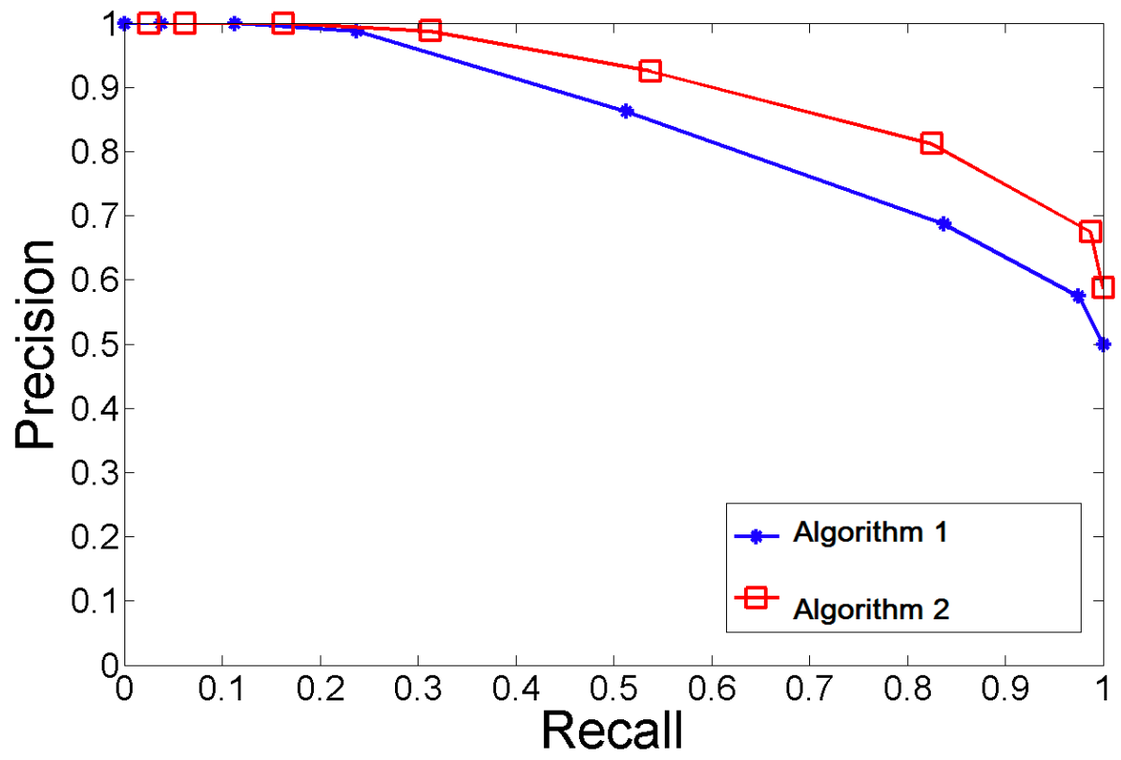
Consider the example of cancer detection

Supervised Learning — Classification

Still, there is a tradeoff between precision/recall

- Precision: TP / (TP + FP)
- Recall: TP / (FN + TP)

In this example, which Algorithm is better?



Supervised Learning — Classification

We've seen many (regularized) linear models for regression, how about classification?

Logistic Regression: for binary classification, we parametrize our model as

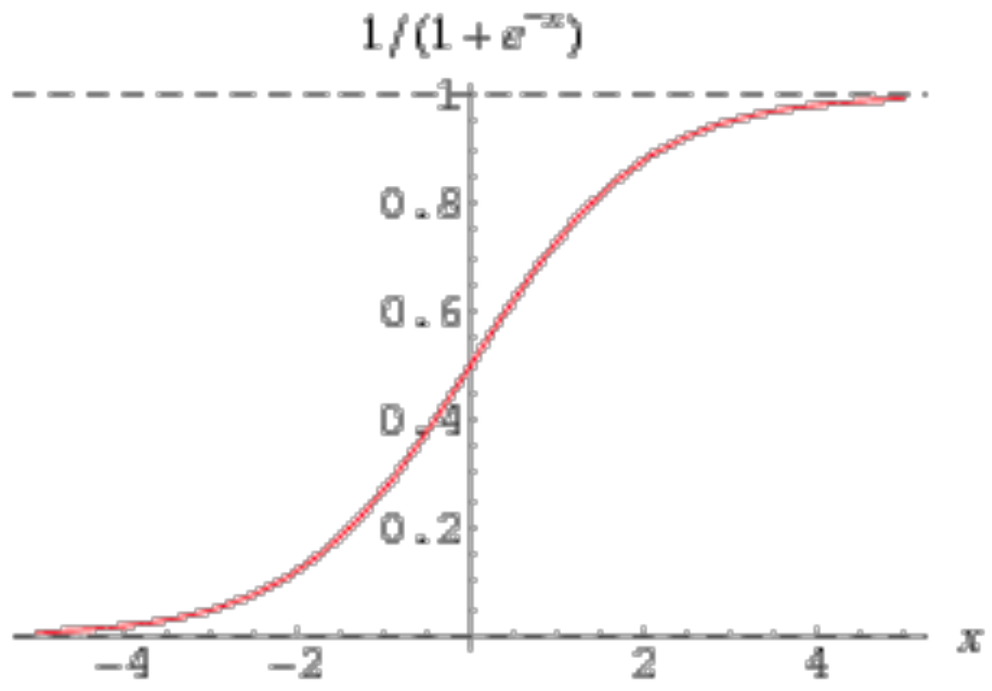
$$\Pr_{\mathbf{w}}(Y = 1 \mid X = \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

- sigmoid function:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

- So, LR =

$$\Pr_{\mathbf{w}}(Y = 1 \mid X = \mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})$$



Supervised Learning — Classification

Why is LR a linear classifier? $\Pr_{\mathbf{w}}(Y = 1 \mid X = \mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$

Hint: for output, we use:

- If $\sigma(\mathbf{w}^T\mathbf{x}) \geq 0.5 \implies \hat{Y} = 1$
- If $\sigma(\mathbf{w}^T\mathbf{x}) < 0.5 \implies \hat{Y} = 0$

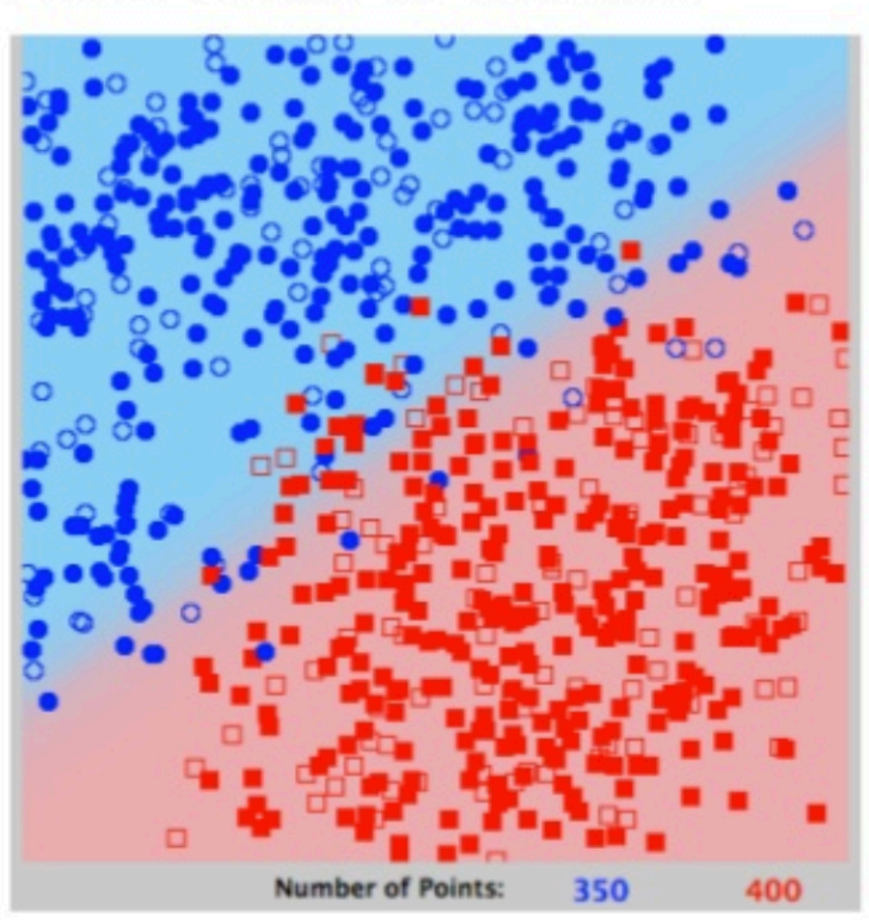
Supervised Learning — Classification

Why is LR a linear classifier? $\Pr_{\mathbf{w}}(Y = 1 \mid X = \mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$

Hint: for output, we use:

- If $\sigma(\mathbf{w}^T\mathbf{x}) \geq 0.5 \implies \hat{Y} = 1$
- If $\sigma(\mathbf{w}^T\mathbf{x}) < 0.5 \implies \hat{Y} = 0$

Decision surface is linear



Supervised Learning — Classification

How to train a LR model?

Again, let's use the MLE principle:

$$\max_{\mathbf{w}} \quad \sum_{i=1}^{n} \mathbb{I}(y_i = 1) \log \sigma(\mathbf{w}^T \mathbf{x}_i) + \mathbb{I}(y_i = 0) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))$$

On the other hand, the above MLE objective function is also equivalent to:

- Minimize KL-div between the empirical distribution and the predicted distribution
- Hence, it's also equivalent to the cross-entropy function between these two distributions

Supervised Learning — Classification

How to train a LR model?

Again, let's use the MLE principle:

$$\max_{\mathbf{w}} \quad \sum_{i=1}^{n} \mathbb{I}(y_i = 1) \log \sigma(\mathbf{w}^T \mathbf{x}_i) + \mathbb{I}(y_i = 0) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))$$

Similarly, we can use GD to optimize the above objective function

HW: Derive the gradient of model parameter in LR w.r.t. the above objective function

Lecture 3: Homework

1. Derive the gradient of model parameter in LR w.r.t. the above objective function Again, let's use the MLE principle:

$$\max_{\mathbf{w}} \quad \sum_{i=1}^{n} \mathbb{I}(y_i = 1) \log \sigma(\mathbf{w}^T \mathbf{x}_i) + \mathbb{I}(y_i = 0) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))$$

2. Let the eigenvalues of $X^T X$ be $\lambda_1, \ldots, \lambda_d$, what's the eigenvalues of $X^T X + \lambda I_d$?

3. A random variable has a Laplace distribution $L(\mu, b)$ if its probability density function is

$$p(x \mid \mu, b) = \frac{1}{2b} \exp(-\frac{|x - \mu|}{b})$$

Show that the objective function of LASSO is an application of the MAP principle with Laplace distribution as prior.