Big Data Analysis

Lecture 6

2019/11/28

Linear Regression

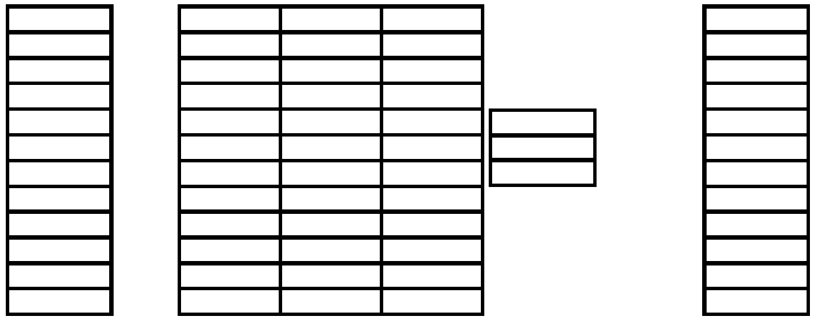
- Review of linear regression concepts
- Challenges when dealing with big data
- Solutions for these challenges
- Materials are borrowed and modified from 2 resources
  - CS 109 Data Science from Harvard, http://cs109.github.io/2015/
  - CS 229 Machine Learning by Andrew Ng, http://cs229.stanford.edu

Linear Regression

Linear Model

often called "OLS" (ordinary least squares), but that puts the focus on the procedure rather than the model.

$$\underbrace{y}_{n\times 1} = \underbrace{X}_{n\times k}\underbrace{\beta}_{k\times 1} + \underbrace{\epsilon}_{n\times 1}$$



Linear Regression

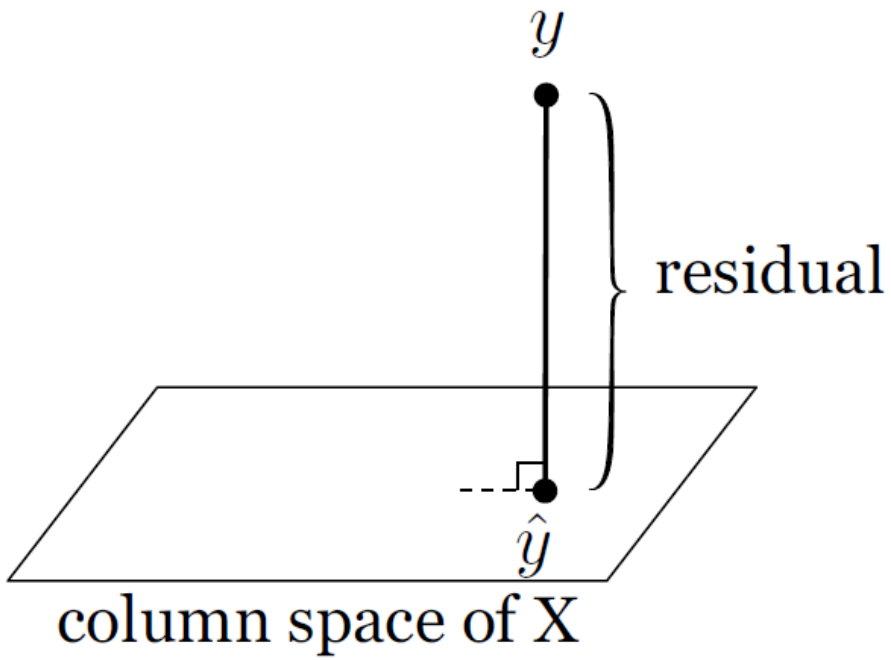
What's linear about it?

$$\underbrace{y}_{n\times 1} = \underbrace{X}_{n\times k}\underbrace{\beta}_{k\times 1} + \underbrace{\epsilon}_{n\times 1}$$

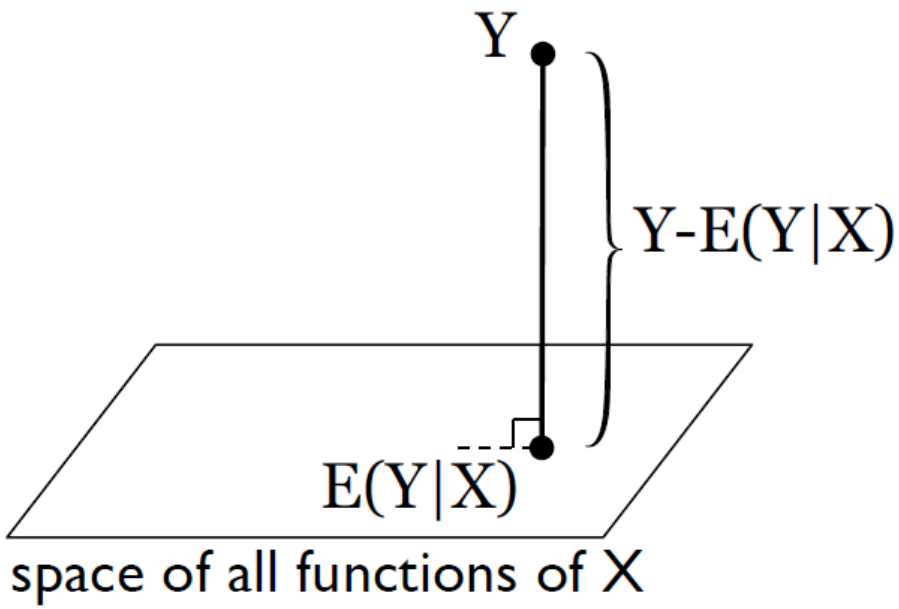
Linear refers to the fact that we're taking linear combinations of the predictors. Still linear if, e.g., use both x and its square and its cube as predictors.

Interpretations

Projection



Conditional Expectation



Normal Equation

Gauss-Markov Theorem

Consider a linear model

$$y = X\beta + \epsilon$$

where y is n by 1, $\mathbf{X}$ is an n by k matrix of covariates, β is a k by 1 vector of parameters, and the errors ϵ_j are uncorrelated with equal variance, $\epsilon_j \sim [0, \sigma^2]$. The errors do not need to be assumed to be Normally distributed.

Then it follows that...

$$\hat{\beta} \equiv (X'X)^{-1}X'y$$

is **BLUE** (the **B**est **L**inear **U**nbiased **E**stimator).

For Normal errors, this is also the MLE.

BIAS

Bias of an Estimator

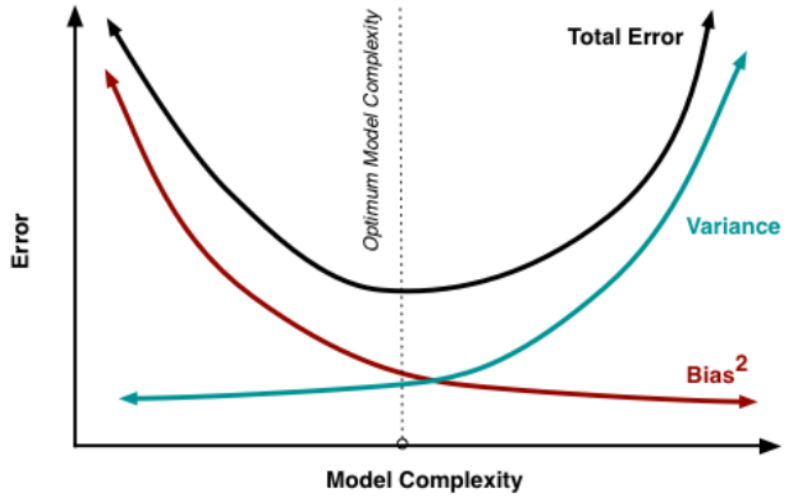
The *bias* of an estimator is how far off it is on average: $\text{bias}(\hat{\theta}) = E(\hat{\theta}) - \theta$

So why not just subtract off the bias?

BIAS – Variance Tradeoff

one form: $\mathrm{MSE}(\hat{\theta}) = \mathrm{Var}(\hat{\theta}) + \mathrm{bias}^2(\hat{\theta})$

often a little bit of bias can make it possible to have much lower MSE

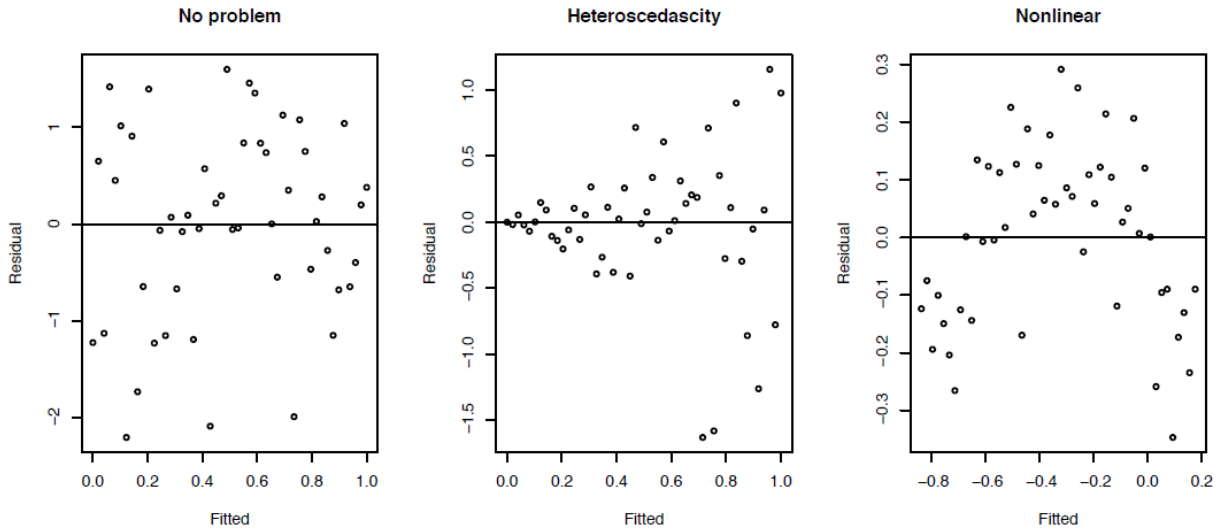


http://scott.fortmann-roe.com/docs/BiasVariance.html

Residuals

Residual Plots

Always plot the residuals! (Plot residuals vs. fitted values, and residuals vs. each predictor variable)



Faraway, http://cran.r-project.org/doc/contrib/Faraway-PRA.pdf

R Square

"Explained" Variance

$$\text{var}(y) = \text{var}(X\hat{\beta}) + \text{var}(e)$$

$$R^2 = \frac{\text{var}(X\hat{\beta})}{\text{var}(y)} = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

R^2 measures goodness of fit, but
it does *not* validate the model.
Adding more predictors can only increase R^2.

Gradient Descent

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Gradient descent:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

$\}$ (simultaneously update for every $j = 0, \ldots, n$)

Gradient Descent

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update θ_j for $j = 0, \ldots, n$)

$\}$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

$\ldots$

Gradient Descent – General Rules

- Normalization
  - Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean (Do not apply to $x_0 = 1$).
  - Get x_i into range of $-1 \leq x_i \leq 1$
- Choose appropriate learning rate α
  - For sufficiently small α, $J(\theta)$ should decrease on every iteration.
  - But if α is too small, gradient descent can be slow to converge.
  - Do we need an adaptive one?

Gradient Descent vs Normal Equation

m training examples, n variables.

Gradient Descent	Normal Equation
• Need to choose α. • Needs many iterations. • Works well even when n is large.	• No need to choose α. • Don't need to iterate. • Need to compute $(X^T X)^{-1}$ • Slow if n is very large.

When does Normal Equation break down?

$$\theta = (X^T X)^{-1} X^T y$$

- What if $X^T X$ is non-invertible? (singular/ degenerate)
  - Redundant variables (linearly dependent).
  - Too many variables

Stochastic Gradient Descent

Batch gradient descent

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Repeat {

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

All samples are used.

(for every $j = 0, \ldots, n$)

}

Stochastic gradient descent

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{train}(\theta) = \frac{1}{m}\sum_{i=1}^{m} cost(\theta, (x^{(i)}, y^{(i)}))$$

1. Randomly shuffle dataset.
2. Repeat {

   for $i := 1, \ldots, m$ {

   $\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$

   (for $j = 0, \ldots, n$)

   }

}

Gradient Descent Methods

Mini-batch gradient descent

Batch gradient descent: Use all m examples in each iteration

Stochastic gradient descent: Use 1 example in each iteration

Mini-batch gradient descent: Use b examples in each iteration

Gradient Descent Convergence

Batch gradient descent:

Plot $J_{train}(\theta)$ as a function of the number of iterations of gradient descent.

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

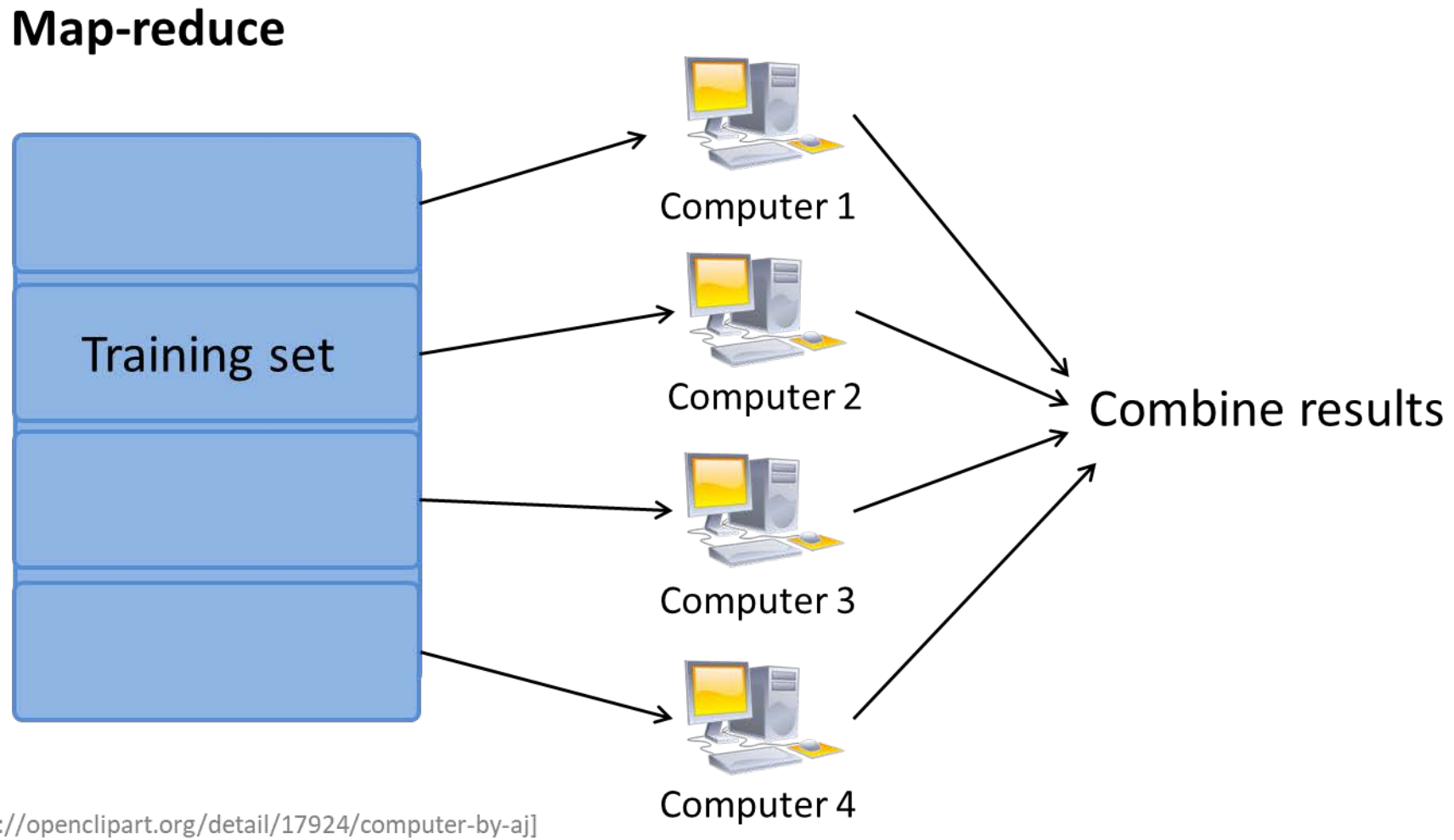
Stochastic gradient descent:

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$$

During learning, compute $cost(\theta, (x^{(i)}, y^{(i)}))$ before updating θ using $(x^{(i)}, y^{(i)})$.

Every 1000 iterations (say), plot $cost(\theta, (x^{(i)}, y^{(i)}))$ averaged over the last 1000 examples processed by algorithm.

MapReduce for Gradient Descent



[http://openclipart.org/detail/17924/computer-by-aj]

MapReduce for Gradient Descent

Map-reduce

Batch gradient descent: $\theta_j := \theta_j - \alpha \frac{1}{400} \sum_{i=1}^{400} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$

Machine 1: Use $(x^{(1)}, y^{(1)}), \ldots, (x^{(100)}, y^{(100)})$.

Machine 2: Use $(x^{(101)}, y^{(101)}), \ldots, (x^{(200)}, y^{(200)})$.

$$temp_j^{(2)} = \sum_{i=101}^{200} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

Machine 3: Use $(x^{(201)}, y^{(201)}), \ldots, (x^{(300)}, y^{(300)})$.

$$temp_j^{(3)} = \sum_{i=201}^{300} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

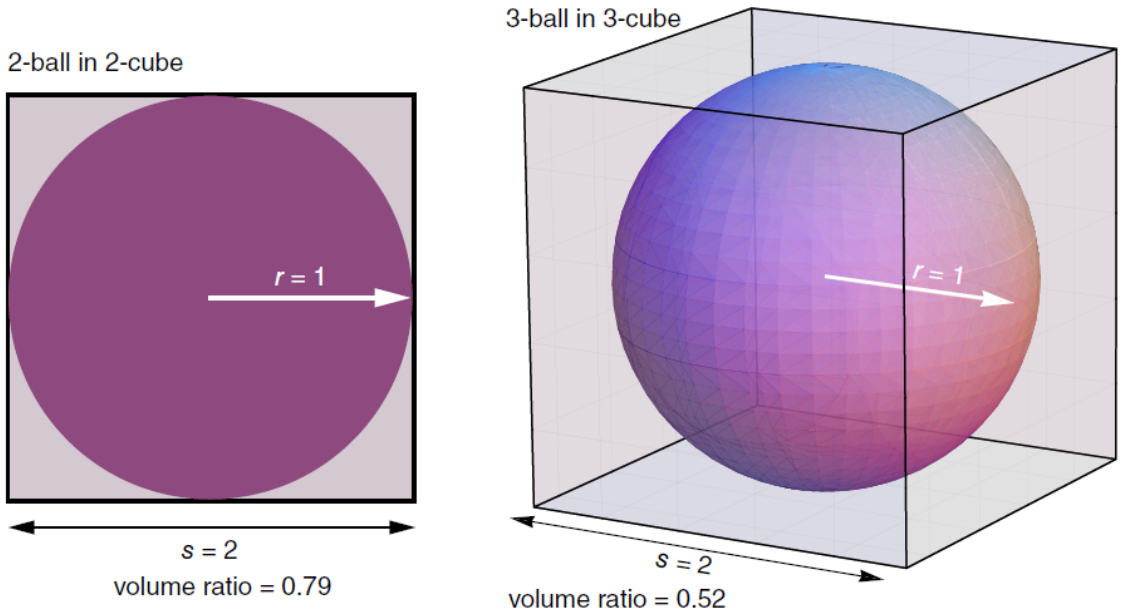
Machine 4: Use $(x^{(301)}, y^{(301)}), \ldots, (x^{(400)}, y^{(400)})$.

$$temp_j^{(4)} = \sum_{i=301}^{400} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

[Jeffrey Dean and Sanjay Ghemawat]

Curse of Dimensionality

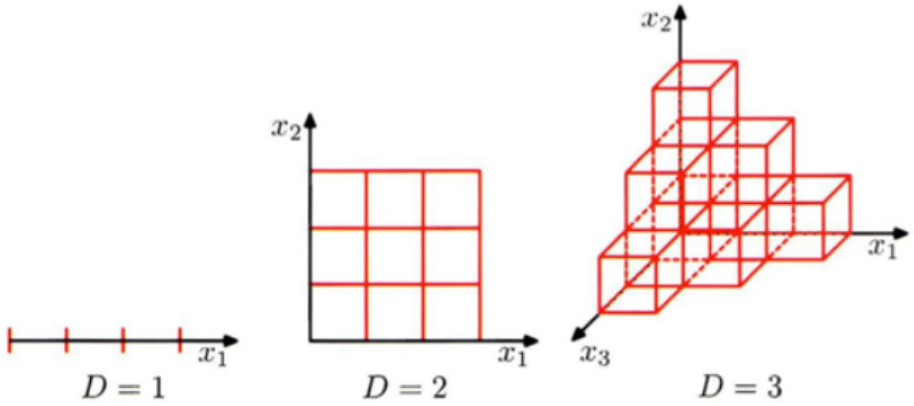
For a uniformly random point in a box with side length 2, what is the probability that the point is in the unit ball?



source: An Adventure in the nth Dimension, Brian Hayes, American Scientist 2011

Curse of Dimensionality

- When dimensionality increases, the volume of the space increases so fast that the available data becomes sparse
- Statistically sound result requires the sample size N to grow exponentially with d



Curse of Dimensionality

For a uniformly random point in the box in d dimensions with length 2 in each dimension, what is the probability that the random vector is in the unit ball in d dimensions?

d	probability
2	0.79
3	0.52
6	0.08
10	0.002
15	0.00001
100	$1.87 \cdot 10^{-70}$

In many high-dimensional settings, the vast majority of data will be near the boundaries, not in the center.

Blessing of Dimensionality

In statistics, "curse of dimensionality" is often used to refer to the difficulty of fitting a model when many possible predictors are available. But this expression bothers me, because more predictors is more data, and it should not be a "curse" to have more data....

With multilevel modeling, there is no curse of dimensionality. When many measurements are taken on each observation, these measurements can themselves be grouped. Having more measurements in a group gives us more data to estimate group-level parameters (such as the standard deviation of the group effects and also coefficients for group-level predictors, if available).

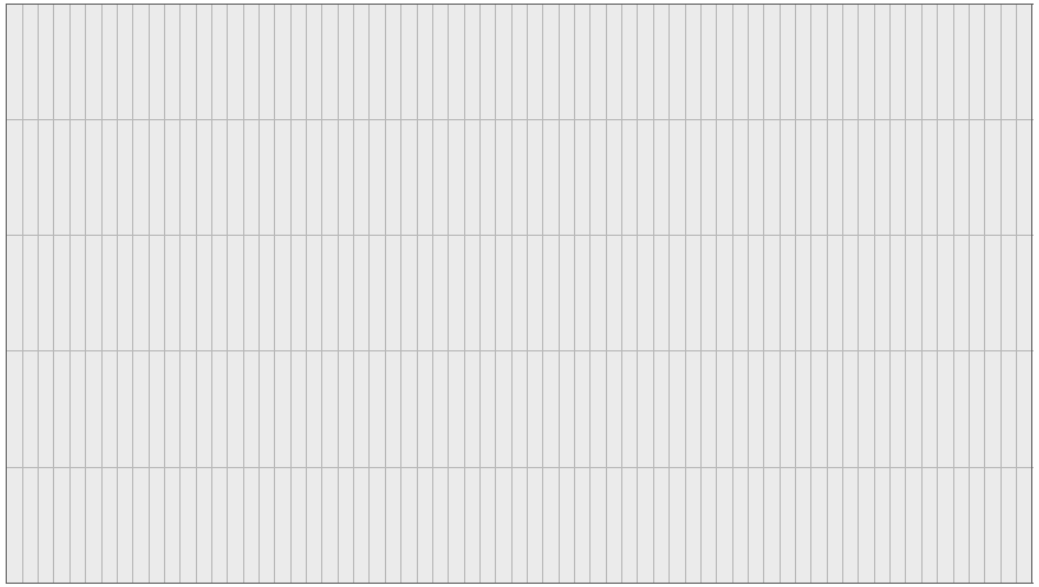
In all the realistic "curse of dimensionality" problems I've seen, the dimensions–the predictors–have a structure. The data don't sit in an abstract K-dimensional space; they are units with K measurements that have names, orderings, etc.

Andrew Gelman, http://andrewgelman.com/2004/10/27/the_blessing_of/

Wide Data

p measurements

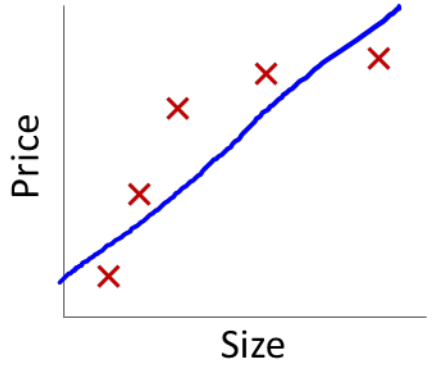
n people



Wide data are increasingly common in applications, e.g., neuroimaging, microarrays, MOOC data. But many traditional statistical methods assume n greater than p.

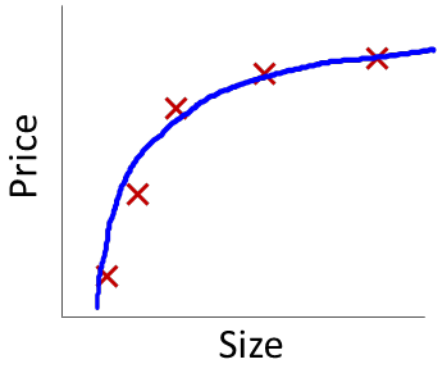
Regularization – Overfitting

Example: Linear regression (housing prices)



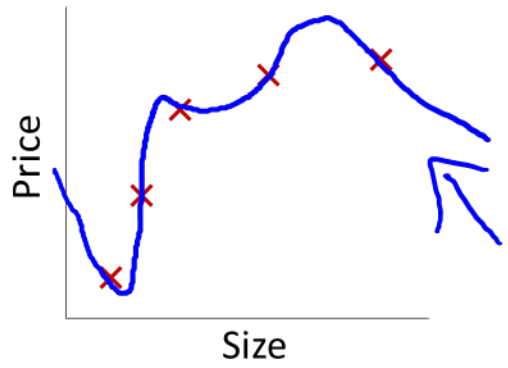
$\rightarrow \theta_0 + \theta_1 x$

"Underfit" "High bias"



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"



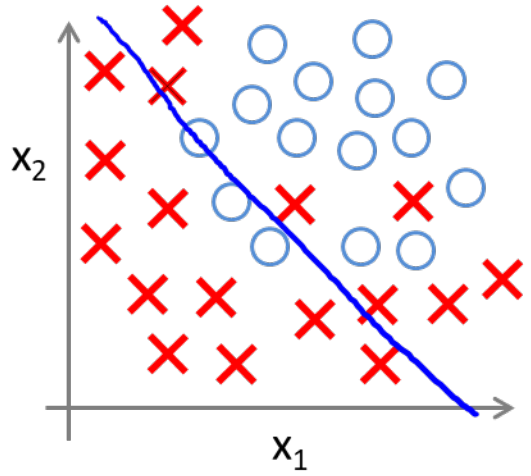
$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

"Overfit" "High variance"

Overfitting: If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$), but fail to generalize to new examples (predict prices on new examples).

Regularization – Overfitting

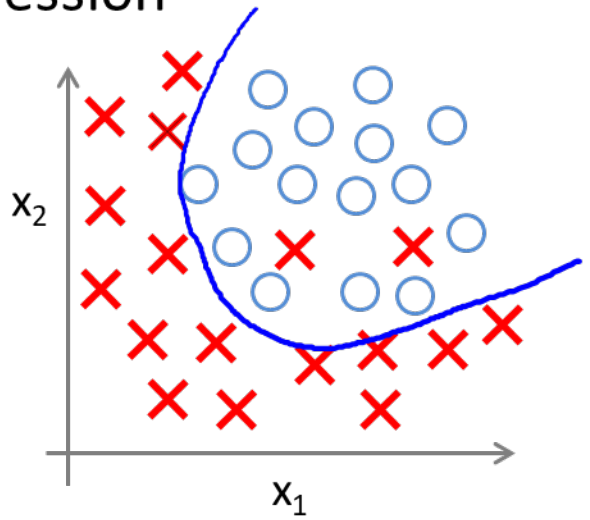
Example: Logistic regression



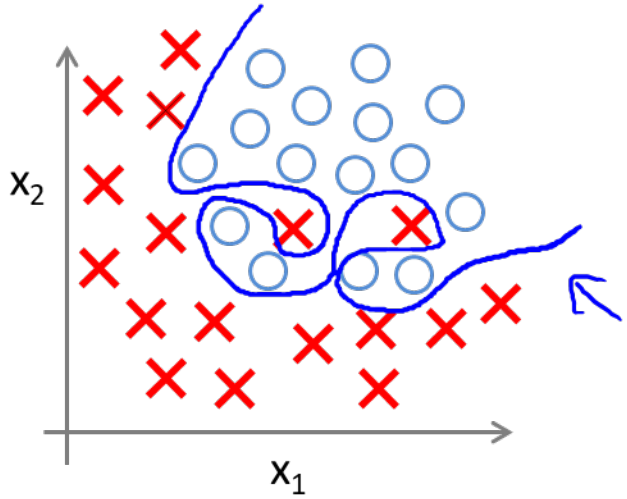
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(g = sigmoid function)

"Underfit"



$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$



$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$

"Overfit"

Addressing Overfitting

Options:

1. Reduce number of features
   - Manually select which features to keep.
   - Model selection algorithm (later in course).
2. Regularization
   - Keep all the features, but reduce magnitude/values of parameters θ_j
   - Works well when we have a lot of features, each of which contributes a bit to predicting sample outputs.

Ridge Regression and Shrinkage

$$\min_{\theta} J(\theta) \qquad J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?

- Algorithm works fine; setting λ to be very large can't hurt it
- Algortihm fails to eliminate overfitting.
- Algorithm results in underfitting. (Fails to fit even training data well).
- Gradient descent will fail to converge.

Gradient Descent for Ridge Regression

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

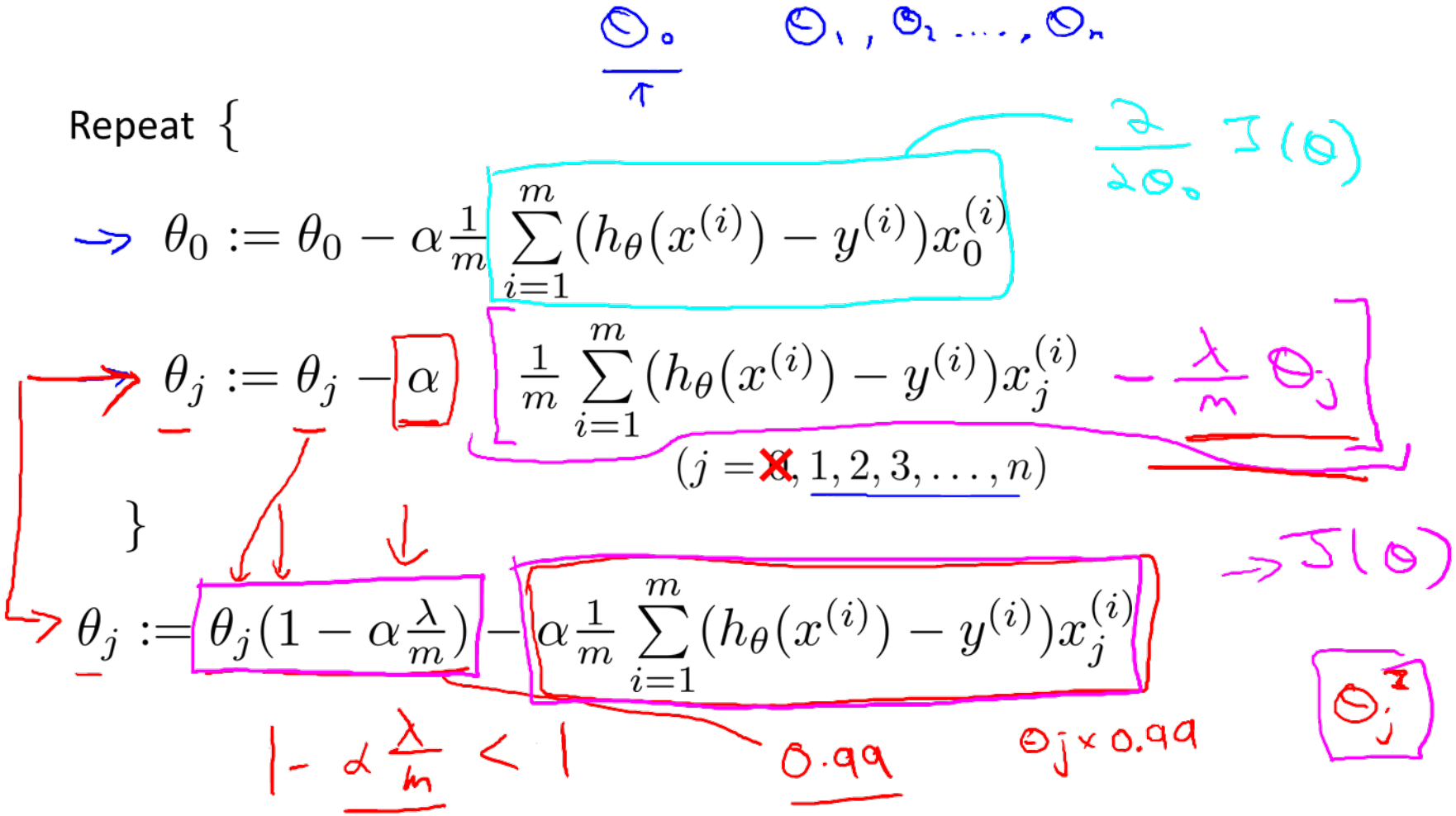
$$\theta_j := \theta_j - \alpha \quad \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$(j = \cancel{0}, 1, 2, 3, \ldots, n)$$

$\}$

$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Gradient Descent for Ridge Regression



Normal Equation – Not Invertible?

Suppose $m \leq n$,

(#examples) (#features)

$$\theta = (X^T X)^{-1} X^T y$$

If $\lambda > 0$,

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots \\ & & & & 1 \end{bmatrix} \right)^{-1} X^T y$$

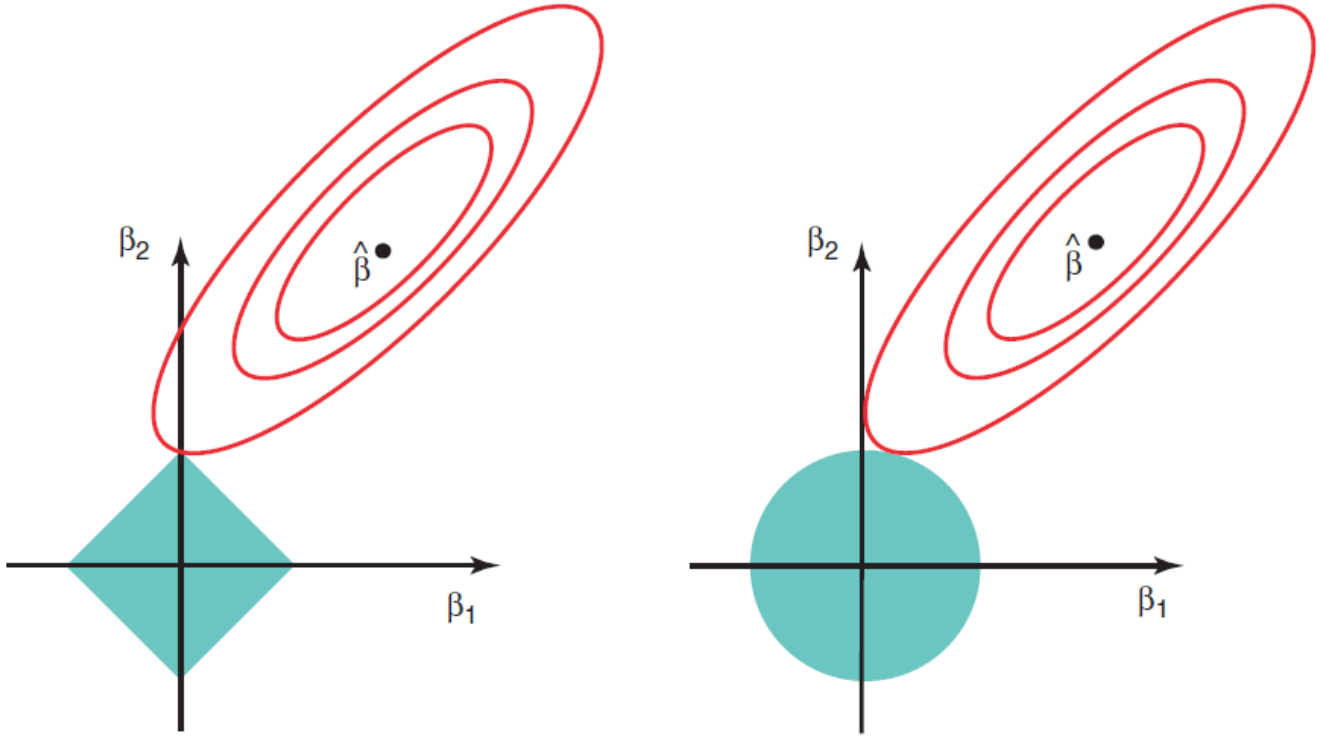
LASSO and Sparsity

In a linear regression model, in place of minimizing the sum of squared residuals, LASSO says to minimize

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j| \;=\mathrm{RSS}+\lambda\sum_{j=1}^{p}|\beta_j|.$$

This helps induce *sparsity*, reducing the number of variables one has to deal with.

LASSO vs. Ridge Constraints



source: Introduction to Statistical Learning, James, Witten, Hastie, Tibshirani, http://www-bcf.usc.edu/~gareth/ISL/