Introduction to Supervised Learning

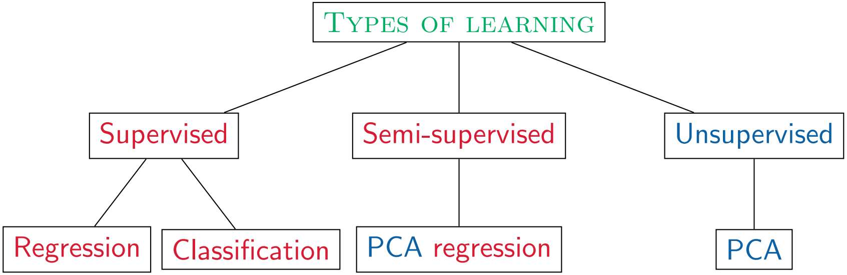
-Applied Multivariate Analysis & Statistical Learning-

ISLR: Chapters 1 to 3.5 (most should be essentially review from previous classes)

Lecturer: Darren Homrighausen, PhD

Preamble:

- Outline the framework for assessing the quality of a procedure
- Outline the notation for a linear regression model
- Review estimation & inference for linear regression models
- Examples of “classical”, “big data”, & “high dimensional”



Some comments:

Comparing predictions to Y gives a natural notion of prediction accuracy

Much more heuristic, unclear what a good solution would be. We'll return to this later in the semester.

Supervised Methods

The set-up

We observe n pairs of data $(X_1^\top, Y_1)^\top, \ldots, (X_n^\top, Y_n)^\top$

Let[1] $Z_i^\top = (X_i^\top, Y_i) \in \mathbb{R}^p \times \mathbb{R}$

We'll refer to the training data as $\mathcal{D} = \{Z_1, \ldots, Z_n\}$

- Y_i is the supervisor or response
  (NOT DEPENDENT VARIABLE)
- $X_i \in \mathbb{R}^p$ is the feature or covariate (vector)
  (or explanatory variables or predictors. NOT INDEPENDENT VARIABLES)

Example: Y_i is whether a threat is detected in an image and the X_{ij} is the value at the j^{th} pixel of an image (p might be $1024^2 = 1048576$)

[1]These transposes get tiredsome. We'll get a bit sloppy and drop them selectively in what follows.

The set-up

We use the training data $\mathcal{D}$ to train an algorithm, producing a function $\hat{f} : \mathbb{R}^p \to \mathbb{R}$

Goal: Given a new $X \in \mathbb{R}^p$, we want to form predictions

$$\hat{f}(X) = \hat{Y}$$

Such that $\hat{Y}$ is a good prediction of Y, the unobserved supervisor

Example: Classically, this is often done with maximum likelihood

- The likelihood ℓ

  (Ex: $\ell(\pi, Y) = \pi^Y(1-\pi)^{1-Y}$ is the Bernoulli likelihood for one observation)
- which is a function of a parameter θ and training data $\mathcal{D}$

$$\hat{\theta} = \arg\max_{\theta} \prod_{i=1}^{n} \ell(\theta, Z_i)$$

Risk, Bayes, bias, variance, and approximation

Loss functions and risk

If we want a $\hat{f}(X)$ which is a good prediction, what does good mean?

Define a loss function which

- Inputs both
  - $\underbrace{\hat{f}(X)}_{\text{Our prediction}}$
  - $\underbrace{Y}_{\text{Unknown, true value}}$
- Outputs a number $\ell(\hat{f}(X), Y)$ between 0 and ∞...

...such that smaller $\ell(\hat{f}(X), Y)$ indicate better performance

(There is an intimate connection between loss and likelihoods, hence same notation)

Risky (and lossy) business

Any distance function could serve for the loss function ℓ

As both $\hat{f}(X)$ and Y are random, the loss function is random

Hence, we define the risk to be the expectation of the loss

$$R(f) = \mathbb{E}\ell(f(X), Y)$$

(Hence, the risk is not random)

Definition: A good procedure f is one that has a small risk $R(f)$

Risky (and lossy) business

More Details: If we want a procedure with small risk it begs the question

$\rightarrow$ What procedure has the smallest risk?

The (unknown) function f_* with the smallest risk is known as the Bayes rule with respect to the loss function ℓ

$$f_* = \operatorname*{argmin}_f R(f) \qquad \text{and} \qquad \min_f R(f) = R(f_*)$$

(Warning: I will use argmin and min a lot in this class)

An example: Squared-error loss

An example: Squared-error loss

If the function $\ell(f(X), Y) = (f(X) - Y)^2$, then

$$f_*(X) = \mathbb{E}[Y|X]$$

This is known as the regression function; that is, the conditional expectation of Y given X.

(Emphasis: This is the Bayes rule with respect to the squared error loss function)

Example: In simple linear regression, the Bayes rule is modeled as

$$f_*(X) = \beta_0 + \beta_1 X$$

Giving rise to the model

$$Y = \beta_0 + \beta_1 X + \epsilon$$

where ϵ is some mean zero fluctuation

An example: Squared-error loss

Recap:

Given the training data $\mathcal{D}$, we want to predict some independent test data $Z = (X, Y)$

This means forming a $\hat{f}$, which is a function of both X and the training data $\mathcal{D}$, which provides predictions $\hat{Y} = \hat{f}(X)$.

The quality of this prediction is measured via the prediction risk

$$R(\hat{f}) = \mathbb{E}(Y - \hat{f}(X))^2$$

We know that the regression function, $f_*(X) = \mathbb{E}[Y|X]$, is the best possible prediction

However, as previously mentioned, it is *unknown*

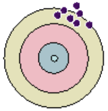
An example: Squared-error loss

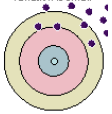
Note that squared prediction risk at any X can be written as

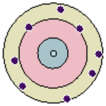
$$\mathbb{E}(\hat{f}(X) - Y)^2 = \mathrm{bias}^2(X) + \mathrm{var}(X) + \sigma^2$$

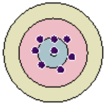
where

$$\begin{aligned}
\mathrm{bias}(X) &= \mathbb{E}\hat{f}(X) - f_*(X) \\
\mathrm{var}(X) &= \mathbb{V}\hat{f}(X) = \mathbb{E}(\hat{f}(X) - \mathbb{E}\hat{f}(X))^2 \\
\sigma^2 &= \mathbb{E}(Y - f_*(X))^2 = \mathbb{V}Y
\end{aligned}$$


Bias is large
variation is small


Bias is large


Bias is small
variation is large


Bias is small
variation is small

Bias-variance tradeoff

This can be heuristically thought of as

$$\text{Prediction risk} = \text{Bias}^2 + \text{Variance} + \text{Irreducible error}$$

There is a natural conservation between these quantities

Low bias $\rightarrow$ complex model $\rightarrow$ many parameters $\rightarrow$ high variance

The opposite also holds
(Think: $\hat{f} \equiv 0$.)

We'd like to 'balance' these quantities to get the best possible predictions

Bias-variance tradeoff

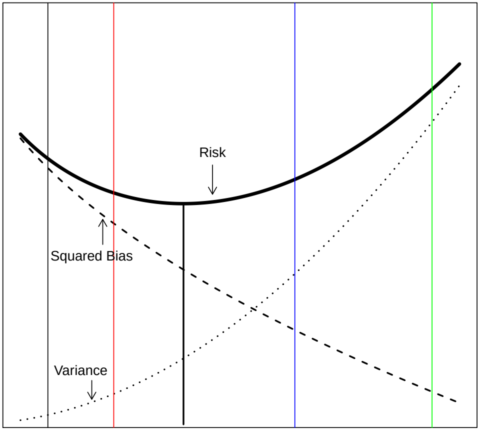


Figure: *

Model Complexity $\nearrow$

The main idea and the main problem

Main idea and problem

In a certain sense, we are done: minimize $R(f)$ over the types of f we are willing to consider
(i.e.: over all $f(X) = X^\top \beta$)

Problem: we never know the distribution of (X, Y)!

Not only is the Bayes rule unknown, but the risk itself is as well!

$$R(f) = \underbrace{\mathbb{E}}_{\text{unknown!}} \Big[\ell(f(X), Y) \Big]$$

Every (supervised) procedure we discuss provides a model/algorithm for estimating some aspect of the distribution of (X, Y) using $\mathcal{D}$

Training error and risk estimation

Since we want to minimize $R(f)$, which is an expectation, perhaps we can approximate it with an average

For any loss function $\ell(f(X), Y)$, we can form the training error

$$\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(X_i), Y_i)$$

In many applied statistical applications, this plug-in estimator of the risk is used

(Think: how many techniques rely on an unconstrained minimization of squared error, or maximum likelihood, or estimating equations, or ...)

This sometimes has disastrous results

Example

Let's look at the regression version: mean squared error (MSE)

$$\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n}(f(X_i) - Y_i)^2$$

Let's suppose $\mathcal{D}$ is drawn from

```
n = 30
X = (0:n)/n*2*pi
Y = sin(X) + rnorm(n,0,.25)
```

Example

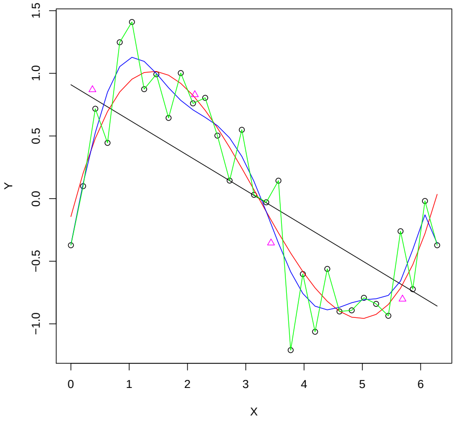
Now, let's fit some polynomials to this data.

We consider the following models:

- Model 1: $f(X_i) = \beta_0 + \beta_1 X_i$
- Model 2: $f(X_i) = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \beta_3 X_i^3$
- Model 3: $f(X_i) = \sum_{j=0}^{10} \beta_j X_i^j$
- Model 4: $f(X_i) = \sum_{j=0}^{n-1} \beta_j X_i^j$

Let's look at what happens...

Example



The $\hat{R}$'s are:

$\hat{R}(\text{Model 1}) = 10.98$

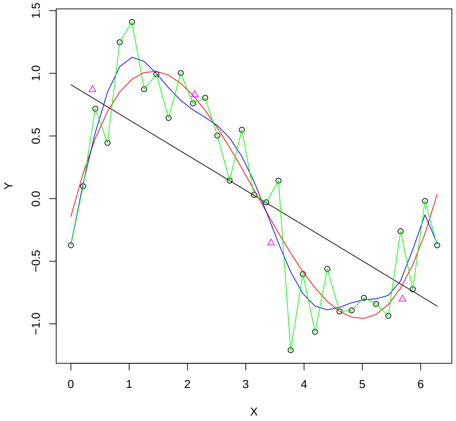
$\hat{R}(\text{Model 2}) = 2.86$

$\hat{R}(\text{Model 3}) = 2.28$

$\hat{R}(\text{Model 4}) = 0$

What about predicting new observations (Δ)?

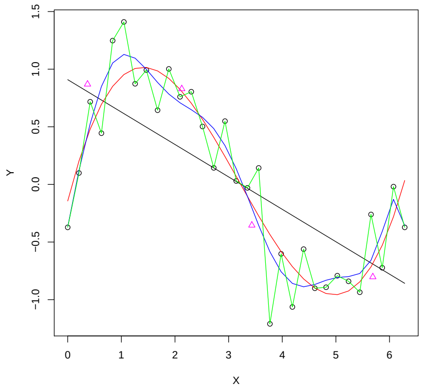
Example

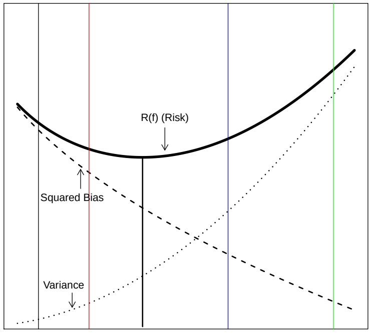


- Black model has low variance, high bias
- Green model has low bias, but high variance
- Red model and Blue model have intermediate bias and variance.

We want to balance these two quantities.

Bias vs. Variance





Model Complexity $\nearrow$

A linear model review

A linear model: Multiple regression

Recall: For regression, squared-error is the usual loss function

$\rightarrow$ The Bayes rule w.r.t. this loss function is $f_*(X) = \mathbb{E}Y|X$

Specify the model: $f_*(X) = \beta_0 + X^\top \beta = \beta_0 + \sum_{j=1}^p x_j \beta_j$

(This means that we think the relationship is approximately linear in X)

Then we recover the usual linear regression formulation

$$\mathbb{X} = \begin{bmatrix} x_1 & \cdots & x_p \end{bmatrix} = \begin{bmatrix} X_1^\top \\ X_2^\top \\ \vdots \\ X_n^\top \end{bmatrix} \in \mathbb{R}^{n \times p} \quad \text{and} \quad \mathbb{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} \in \mathbb{R}^n$$

Commonly, a column $x_0^\top = \underbrace{(1, \ldots, 1)}_{n \text{ times}}$ is included

(This encodes an intercept term, with intercept parameter β_0)

We could (should?) seek to find a β such that $\mathbb{Y} \approx \mathbb{X}\beta$

A linear model: Polynomial effects

Instead, we may believe

$$f_*(X) = \beta_0 + \sum_{j=1}^{p} x_j \beta_j + \sum_{j \leq j'}^{p} x_j x_{j'} \beta_{jj'}$$

Then the feature matrix is

$$\mathbb{X} = \left[\begin{array}{cccccccc} x_0 & x_1 & \cdots & x_p & x_1^2 & x_1 x_2 & \cdots & x_p^2 \end{array}\right]$$

(Here, interpret vector multiplication in the entrywise sense, as in R: `x * y`)

This corresponds to the "main and interaction effects" model

Example: Biometrics

Example

Suppose we have 4 subjects in an experiment

We record

- BMI
- minutes spent exercising in the last 7 days

We want to predict each subject's resting heart rate

The classic linear model would model the regression function as

$$f_*(X) = \beta_0 + \beta^\top X = \beta_0 + \beta_1 \mathrm{BMI} + \beta_2 \mathrm{exercise}$$

where

$$\begin{aligned} f_*(X) &= \mathbb{E}[\text{resting heart rate}|X] \\ X &= [\mathrm{BMI}, \mathrm{exercise}] \end{aligned}$$

(Note: we could write $f_*(X) = \beta^\top X$ and $X = [1, \mathrm{BMI}, \mathrm{exercise}]$ instead)

Example

Under this model, the feature matrix and supervisor vector look like

$$\mathbb{X} = \begin{bmatrix} x_1 & x_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 21 & 92 \\ 17 & 12 \\ 29 & 306 \\ 25 & 53 \end{bmatrix}}_{\text{BMI} \quad \text{exercise}} \in \mathbb{R}^{4\times 2}$$

and

$$\mathbb{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_4 \end{bmatrix} = \begin{bmatrix} 72 \\ 47 \\ 82 \\ 64 \end{bmatrix} \in \mathbb{R}^4$$

Example

Adding a quadratic polynomial transformation

$$
\begin{aligned}
f_*(X) &= \beta_0 + \sum_{j=1}^{p} x_j \beta_j + \sum_{j \leq j'}^{p} x_j x_{j'} \beta_{jj'} \\
&= \beta_0 + \beta_1 \mathrm{BMI} + \beta_2 \mathrm{exercise} + \beta_{11} \mathrm{BMI}^2 + \beta_{22} \mathrm{exercise}^2 \\
&\quad + \beta_{12} \mathrm{BMIexercise}
\end{aligned}
$$

Under this model, the feature matrix looks like

$$
\mathbb{X} = \left[\begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 \end{array} \right] = \underbrace{\left[\begin{array}{ccccc} 21 & 92 & 21^2 & 92^2 & 21 * 92 \\ 17 & 12 & 17^2 & 12^2 & 17 * 12 \\ 29 & 306 & 29^2 & 306^2 & 29 * 306 \\ 25 & 53 & 25^2 & 53^2 & 25 * 53 \end{array} \right]}_{\mathrm{BMI} \quad \mathrm{exercise} \quad \mathrm{BMI}^2 \quad \mathrm{exercise}^2 \quad \mathrm{BMI} * \mathrm{exercise}}
$$

($\mathbb{Y}$ is the same)

End example

A linear model: Estimating β

In either case, we have a feature matrix $\mathbb{X}$ and supervisor vector $\mathbb{Y}$

Now, we want to estimate a parameter vector β in the model

$$\mathbb{Y} = \mathbb{X}\beta + \epsilon$$

where $\mathbb{V}\epsilon = \sigma^2$

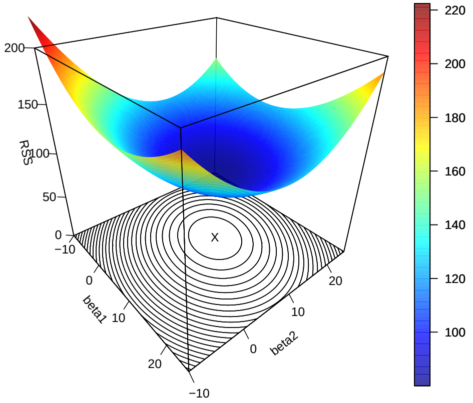
Classical least squares: Minimize the training error $\hat{R}(f)$ over all functions $f_\beta(X) = X^\top \beta$

$$\hat{\beta}_{LS} = \operatorname*{argmin}_\beta \hat{R}(f_\beta) = \operatorname*{argmin}_\beta \sum_{i=1}^n (Y_i - X_i^\top \beta)^2 = \operatorname*{argmin}_\beta ||\mathbb{Y} - \mathbb{X}\beta||_2^2$$

(Though we write this as equality, there is only a unique solution if $\mathrm{rank}(\mathbb{X}) = p$)

A linear model: Estimating β

What does the objective function look like?



A linear model: Properties of $\hat{\beta}_{LS}$

In this case,

$$\hat{f}(X) = X^\top \hat{\beta}_{LS} = X^\top \mathbb{X}^\dagger Y \underbrace{=}_{\text{rank}(\mathbb{X})=p} X^\top (\mathbb{X}^\top \mathbb{X})^{-1} \mathbb{X}^\top Y$$

($\mathbb{X}^\dagger$ is a pseudo inverse)

The fitted values are $\hat{\mathbb{Y}} = \mathbb{X}\hat{\beta}_{LS}$

(Contrary to $\hat{\beta}_{LS}$, the fitted values are always unique)

We can examine the first and second moment properties of $\hat{\beta}_{LS}$

$$\mathbb{E}\hat{\beta}_{LS} = \beta \qquad \text{(unbiased if } f_*(X) = X^\top \beta \text{ is correct model)}$$

$$\mathbb{V}\hat{\beta}_{LS} = \mathbb{X}^\dagger (\mathbb{V}Y)(\mathbb{X}^\dagger)^\top = \sigma^2 (\mathbb{X}^\top \mathbb{X})^{-1}$$

As $\hat{\beta}_{LS}$ is a fancy average, the central limit theorem (CLT) states

$$\hat{\beta}_{LS} \sim N(\beta, \sigma^2 (\mathbb{X}^\top \mathbb{X})^{-1})$$

A linear model: Inference using $\hat{\beta}_{LS}$

Using the CLT result:

$$\hat{\beta}_{LS} \sim N(\beta, \sigma^2(\mathbb{X}^\top \mathbb{X})^{-1})$$

We can test whether $\beta_j = \text{(some value)}$ via

$$t_j = \frac{\hat{\beta}_{LS,j} - \text{(some value)}}{\sqrt{\mathbb{V}\hat{\beta}_{LS,j}}}$$

where $\mathbb{V}\hat{\beta}_{LS,j}$ is the j^{th} diagonal element of $\sigma^2(\mathbb{X}^\top \mathbb{X})^{-1}$

Under the null hypothesis, $t_j \sim t_{n-p}$

So, large values of $|t_j|$ relative to quantiles of t_{n-p} provides some evidence that $\beta_j \neq \text{(some value)}$

End review

Turning these ideas into procedures

Each of these methods have parameters to choose:

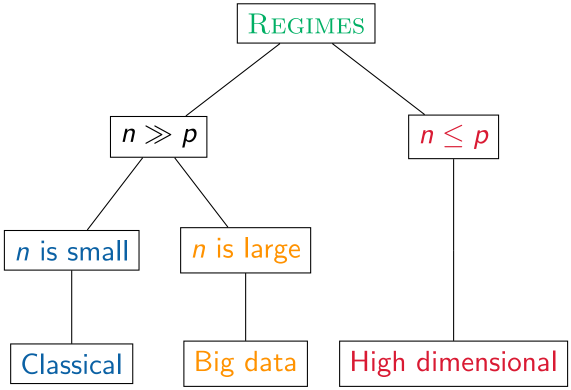
- p could be very large. Do we include all the features?
- If we include some polynomial (or other transformations) terms, should be include all of them?
- Are there other parameters that need to be set in an informed manner?

Additionally, we need to estimate the associated coefficient vector β or whatever

We would like the data to inform these parameters

Turning these ideas into procedures

Back to the three regimes of interest, assuming $\mathbb{X} \in \mathbb{R}^{n \times p}$



Classical regime

Back to $\hat{\beta}_{LS}$:

The Gauss-Markov theorem assures us that this is the best linear unbiased estimator of β

Also, it is the maximum likelihood estimator under the i.i.d. Gaussian model

(Hence, it is asymptotically efficient)

Does that necessarily make it is any good?

Classical regime

Write $\mathbb{X} = UDV^\top$ for the SVD of $\mathbb{X}$

Then $\mathbb{V}\hat{\beta}_{LS} = \sigma^2(\mathbb{X}^\top\mathbb{X})^{-1} = \sigma^2(VD\underbrace{U^\top U}_{=I}DV^\top)^{-1} = \sigma^2 VD^{-2}V^\top$

(Reminder: The d_j are the axes lengths of the ellipse of $\mathbb{X}$)

Suppose we are interested in estimating β

Then we want $\mathbb{E}||\hat{\beta}_{LS} - \beta||_2^2$ to be small

(That is, our estimator is close to the true parameter on average)

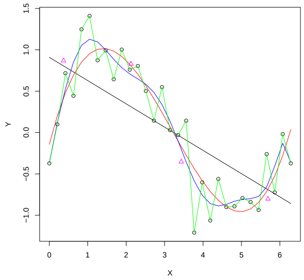
But,

$$\mathbb{E}||\hat{\beta}_{LS} - \beta||_2^2 = \mathrm{trace}(\mathbb{V}\hat{\beta}) = \sigma^2 \sum_{j=1}^{p} \frac{1}{d_j^2} \qquad (1)$$

(Can you show this? Hint: add and subtract $\mathbb{E}\hat{\beta}_{LS}$)

Important: Even in the classical regime, we can do arbitrarily badly if $d_p \approx 0$! (An example of this would be "multicollinearity")

Returning to polynomial example: Bias



Using a Taylor's series, for all x

$$\sin(x) = \sum_{q=0}^{\infty} \frac{(-1)^q x^{2q+1}}{(2q+1)!}$$

Higher order polynomial models will reduce the bias part

Returning to polynomial example: Variance

The least squares solution is given by solving $\min ||\mathbb{X}\beta - Y||_2^2$

$$\mathbb{X} = \begin{bmatrix} 1 & X_1 & \dots & X_1^{p-1} \\ & \vdots & & \\ 1 & X_n & \dots & X_n^{p-1} \end{bmatrix},$$

is the associated feature matrix

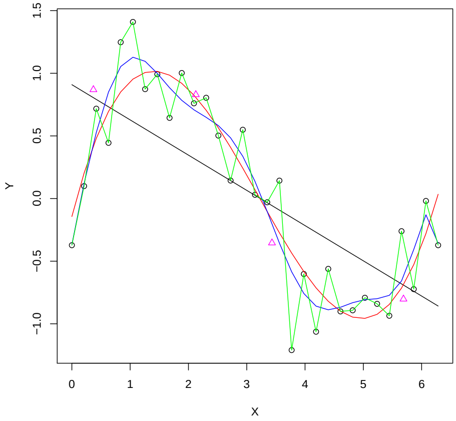
(This is known as the Vandermonde matrix in numerical analysis)

This matrix is well known for being numerically unstable due to $d_p \approx 0$

Hence

$$\sum_{j=1}^{p} \frac{1}{d_j^2} \text{ is huge!}$$

Returning to the polynomial example



Conclusion

Conclusion: Fitting the full least squares model, even in the classical regime, can lead to poor prediction/estimation performance

In the other regimes, we encounter even more sinister problems

Big data regime

Big data: Computational/storage complexity scales extremely quickly. This means that procedures that are feasible classically are not for large data sets

Example: Fit $\hat{\beta}_{LS}$ with $\mathbb{X} \in \mathbb{R}^{n \times p}$. Next fit $\hat{\beta}_{LS}$ with $\mathbb{X} \in \mathbb{R}^{3n \times 4p}$

The second case will take $\approx (3 * 4^2) = 48$ times longer to compute, as well as ≈ 12 times as much memory!
(In general, the computational complexity scales like np^2)

Conclusion

```
p = 300; n = 10000
Y = rnorm(n); X = matrix(rnorm(n*p),nrow=n,ncol=p)
start = proc.time()[3]
out   = lm(Y~.,data=data.frame(X))
end   = proc.time()[3]
smallTime = end - start

n = nMultiple*n; nMultiple = 3
p = pMultiple*p; pMultiple = 4
Y = rnorm(n); X = matrix(rnorm(n*p),nrow=n,ncol=p)
start = proc.time()[3]
out   = lm(Y~.,data=data.frame(X))
end   = proc.time()[3]
bigTime = end - start
> print(bigTime/smallTime)
 elapsed
38.61458
> print(nMultiple*pMultiple**2)
[1] 48
```

Treatment in practice

Depending on the data and the desired method, we could:

- Combine randomized projections together with in-memory procedures

  (Example: We can randomly subsample observations and then load into memory)
- Use (stochastic) gradient descent

  (We will return to this later)
- Leverage an iterative implementation for exact computation

  (Example: biglm in R)
- Break the computations down into small bits and distribute these to different cores/processors/nodes

  (This is like the map-reduce paradigm)

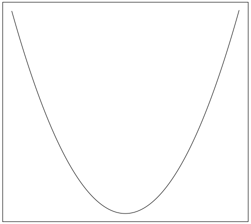
High dimensional regime

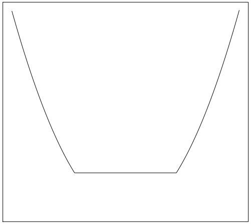
High dimensional: These problems tend to have many of the computational problems as Big data, as well as a rank problem:

Suppose $\mathbb{X} \in \mathbb{R}^{n \times p}$ and $p > n$

Then $\mathrm{rank}(\mathbb{X}) = n$ and the equation $\mathbb{X}\hat{\beta} = Y$:

- can be solved *exactly* (that is; the training error is 0)
- has an infinite number of solutions


$n > p$


$n < p$

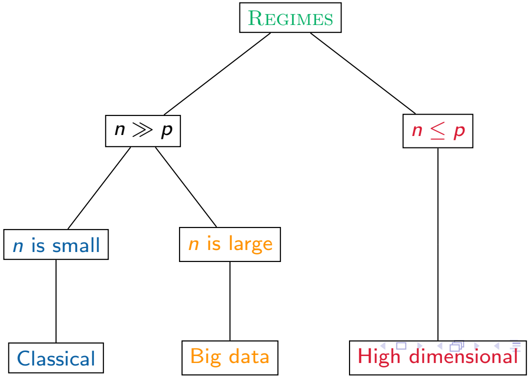
Postamble:

- Outline the framework for assessing the quality of a procedure

  (Loss and risk functions. A good procedure is one that has small risk. The risk is unknown in practice. The training error isn't a reliable estimator of the risk)
- Outline the notation for a linear regression model

  (Write $\mathbb{Y} = \mathbb{X}\beta + \epsilon$)
- Review estimation & inference for linear regression models

  (Least squares is same as minimizing training error with squared error loss)
- Examples of "classical", "big data", & "high dimensional"



Regimes
$n \gg p$
$n \leq p$
n is small
n is large
Classical
Big data
High dimensional