Introduction to Regression

-Introduction to Data Science-

ISL 2.1, 2.2, 3.1, 3.2

Lecturer: Darren Homrighausen, PhD

Preamble:

- Outline the notation for a linear regression model
- Briefly review estimation, prediction, and inference for classical linear regression models
- Give examples of the “classical”, “big data”, and “high dimensional” types of problems

Notation recap

- We have data $\mathcal{D} = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$
  (The training data)
- $X \in \mathbb{R}^p$ is a vector of measurements for each subject
  (Example: $X_i = [1, \text{income}_i, \text{education}_i]^\top$)
- $x \in \mathbb{R}^n$ is a vector of subjects for each measurement
  (Example: $x_j = [\text{income}_1, \text{income}_2, \ldots, \text{income}_n]^\top$)
- X_{ij} is the j^{th} measurement on the i^{th} subject
  (Example: $X_{ij} = \text{income}_i$)

Notational landmine: representing the j^{th} entry of X

$\rightarrow$ A reasonable, but technically sloppy, solution: x_j

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

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

(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

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

Under this model, the feature matrix looks like

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

($\mathbb{Y}$ 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 $f_\beta(X) = X^\top \beta$

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

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

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

In this case,

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

($\mathbb{X}^\dagger$ 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:

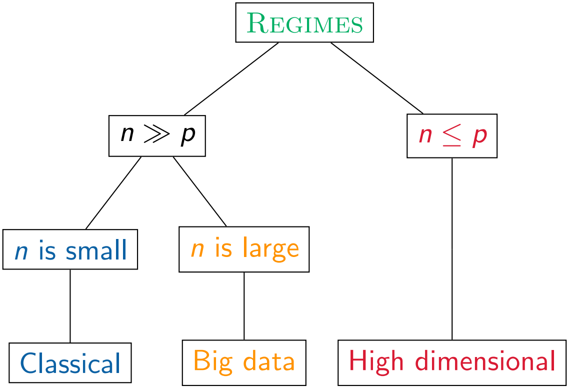
- p could be very large. Do we include all the features?
- If we include some polynomial (or other transformations) terms, should be include all of them?
- Are there other parameters that need to be set in an informed manner?

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)

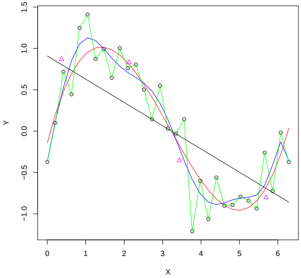
But,

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

(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 $\min ||\mathbb{X}\beta - Y||_2^2$

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

is the associated feature matrix

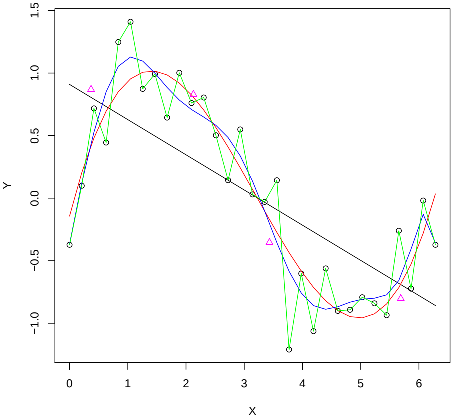
(This is known as the Vandermonde matrix in numerical analysis)

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)

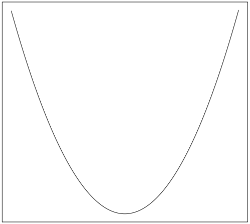
High dimensional regime

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

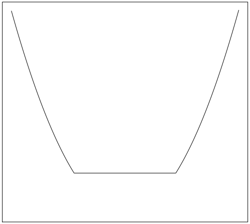
Suppose $\mathbb{X} \in \mathbb{R}^{n \times p}$ and $p > n$

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 notation for a linear regression model

  (Write $\mathbb{Y} = \mathbb{X}\beta + \epsilon$)
- Briefly review estimation, prediction, and inference for linear regression models

  (Least squares is same as minimizing training error with squared error loss)
- Define and give examples of the "classical", "big data", and "high dimensional" types of problems