DSO530 Statistical Learning Methods

Lecture 2a: Review and Simple Linear Regression

Dr. Xin Tong
Department of Data Sciences and Operations
Marshall School of Business
University of Southern California

Review: random variables and probability

- Suppose X is a standard normal random variable, i.e., $X \sim N(0,1)$.
  - i). What is $P(X=0)$?
  - ii). Can we say that $P(X=0) > P(X=1)$?
  - iii). For some small $\epsilon > 0$, is it true

$$P\left(X \in (-\epsilon, \epsilon)\right) > P\left(X \in (1-\epsilon, 1+\epsilon)\right)?$$

- $Z \sim \mathit{Uniform}(0,1)$. How much is $P(0.3 \leq Z \leq 0.7)$?

- We cans visualize disjoint events A and B in a Venn Diagram. Is it clear how to visualize independent events?

- Are disjoint events independent?

Review: basics in Statistics and Lecture 1

- Can you name the difference between **supervised learning** and **unsupervised learning**?
- Define **population**, **sample**, **parameter**, **statistics** in the example: "A politician selects a random sample of 200 working U.S. women who are 16 to 24 year sold. Of the women in the sample, 5% are being paid minimum wage or less".
- How to interpret **confidence intervals** (CIs)?
- Other things being equal, is 99% CI wider than 95%? Why we don't talk about 100% CI?

Review: basic concepts in Statistics

- What is **Simpson's Paradox**?
- Check out the hospital example in the week one technical notes and google the Berkeley graduate admission case.
- Simpson's Paradox vividly illustrates why business analytics must not be viewed as a purely technical subject appropriate for mechanization or automation.
- For further reading (3 cases):

https://www.statslife.org.uk/the-statistics-dictionary/2012-simpson-s-paradox-a-cautionary-tale-in-advanced-analytics

On more thing: about hypothesis test

- Hypothesis test: **null hypothesis** H_0 v.s. **alternative hypothesis** H_a
- Null hypothesis is usually the status quo, old technology, ineffective new drugs. . . .
- What is a **p-value**? the probability of obtaining a result equal to or "more extreme" than what was actually observed, when the null hypothesis is true
- α is a user specified value called *level of significance*
- We should design a statistical test such that $P_{H_0}(\text{reject } H_0) \leq \alpha$
- Reject the null hypothesis if p-value is smaller than or equal to α

On more thing: about hypothesis test

- Hypothesis test: **null hypothesis** H_0 v.s. **alternative hypothesis** H_a
- Null hypothesis is usually the status quo, old technology, ineffective new drugs. . . .
- What is a **p-value**? the probability of obtaining a result equal to or "more extreme" than what was actually observed, when the null hypothesis is true
- α is a user specified value called *level of significance*
- We should design a statistical test such that $P_{H_0}(\text{reject } H_0) \leq \alpha$
- Reject the null hypothesis if p-value is smaller than or equal to α

Linear regression with a single variable

- The package **scikit-learn** is a widely used `Python` library for machine learning, built on top of `NumPy` and some other packages.

- There are five basic steps when you are implementing linear regression:
  - Import the packages and classes you need.
  - Provide data to work with and do appropriate transformations if necessary.
  - Create a regression model and fit it with existing data.
  - Check the results of model fitting to know whether the model is satisfactory.
  - Apply the model for predictions.

Import the packages and read in data

```
import numpy as np
from sklearn.linear_model import LinearRegression
```

Figure 1: import NumPy and LinearRegression

```
import pandas as pd
import matplotlib.pyplot as plt
```

```
height = pd.read_csv("data/galton.csv");height.head()
```

	Unnamed: 0	child	parent
0	1	66.435917	70.851069
1	2	65.943364	69.858889
2	3	64.278858	65.278141
3	4	63.851914	64.032631
4	5	63.192294	63.418992

Figure 2: Do you see any undesirable elements in this DataFrame?

Yes, it is about the column 0!

```
height = pd.read_csv("data/galton.csv", index_col=0); display(height.head())
#can also try print(height.head())
```

	child	parent
1	66.435917	70.851069
2	65.943364	69.858889
3	64.278858	65.278141
4	63.851914	64.032631
5	63.192294	63.418992

Figure 3: This time it looks right

```
height.info() # the .info() give you some information about the DataFrame height
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 928 entries, 1 to 928
Data columns (total 2 columns):
child     928 non-null float64
parent    928 non-null float64
dtypes: float64(2)
memory usage: 21.8 KB
```

Some summary statistics about the variables in the dataframe

```
height.describe() # columnwise summary statistics
```

	child	parent
count	928.000000	928.000000
mean	68.086288	68.299524
std	1.527244	1.857460
min	63.192294	63.418992
25%	67.027340	67.233314
50%	68.115752	68.336807
75%	69.077425	69.485036
max	72.185332	73.355968

Figure 4

- What is the unit of measurements?
- What are the medians of the child and parent heights?

Visual inspection

```
plt.figure(); plt.scatter(height['parent'], height['child']); plt.show()
```

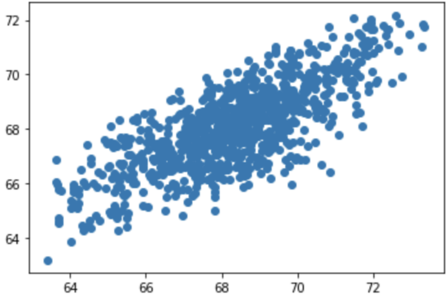

Figure 5: Scatter plot

- What information can be tell by looking at this scatter plot?

The most naive prediction

- Take parent's height as the predicted value for child's height: $\hat{y} = x$

```
plt.figure(); plt.scatter(height['parent'], height['child'])
plt.xlabel('parent height'); plt.ylabel('child height'); plt.title('child vs. parent heights')
lineStart = height['parent'].min(); lineEnd = height['parent'].max()
plt.plot([lineStart, lineEnd], [lineStart, lineEnd], color = 'r'); plt.show()
```

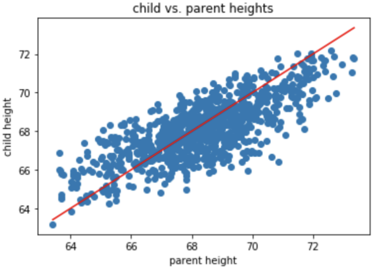

Figure 6: The red line is the 45 degree angle line

- Does the prediction look good? By what standard?

Predict child's heights using parent's heights

- Probably a better idea: $\widehat{y} = \underbrace{\widehat{\beta}_0}_{\text{intercept}} + \underbrace{\widehat{\beta}_1}_{\text{slope}} x$

- The **best** fit line of this form. (what does "best" mean?)

- Why do people believe in a simple prediction formula of this kind?
  - simplicity means better interpretability potential
  - in the old days, the solution can be calculated by hand
  - OK prediction performance on some problems

- Before we continue the Python implementation of the simple linear regression, we take a detour and talk about the simple linear regression **model**

- But what is a model? More specifically, what is a simple linear regression model?

Linear regression model

- Defining the model (according to George Box)
  - All models are wrong; some models are useful...
  - Just as the ability to devise simple but evocative models is the signature of the great scientist
  - so overelaboration and overparameterization is often the mark of mediocrity (an insight when the available sample sizes were NOT huge)
- The simplest of all is a (simple) *linear* regression model; that is, the *response* or *target* y satisfies

$$y = \beta_0 + \beta_1 x + \varepsilon$$

in which

- β_0 is the *intercept* term
- β_1 is the *slope*
- β_0 and β_1 are *coefficients* or *parameters* of the linear model
- ε is a noise term, which is usually assumed independent of x and mean zero. It is often assumed to be normally distributed in theoretical analysis. Often, we assume the standard deviation of ε (σ) is unknown, which is another parameter of the simple linear regression model.

Linear regression (child heights vs. parent heights)

Suppose you have used `Python` to find the *best* linear fit:

$$\begin{aligned}\hat{y} &= \hat{\beta}_0 \qquad + \qquad \hat{\beta}_1 x \\ &= \hat{\beta}_0 + \hat{\beta}_1 \overline{x} + \hat{\beta}_1 \cdot (x - \overline{x}),\end{aligned}$$

where $\overline{x}$ is the mean parent height

Why did we decompose the equation this way?

Let me first tell you that the slope estimate $\hat{\beta}_1 \in (0, 1)$ in our example

- $\hat{\beta}_0 + \hat{\beta}_1 \overline{x}$ is average child height
- $\hat{\beta}_1 \cdot (x - \overline{x})$ is *regression to mean*, so taller parents' children shrink toward average

How do $\hat{\beta}_0$ and $\hat{\beta}_1$ differ from β_0 and β_1?

Fitting the regression model

- **Data**: Have n pairs (x_i, y_i), where x_i is the height of parent i and y_i is height of child i
- **Predictions**: $\hat{y}_i$ is our model's prediction of the height of child i
- **Loss function**: First, define a way to measure *error* or *residual* $e_i = y_i - \hat{y}_i$

$$Loss(y, \hat{y}) = (y - \hat{y})^2$$

- Why does this loss function make sense? It is the only loss function that makes sense?

Fitting the regression model

- **Data**: Have a lot of pairs (x_i, y_i), where x_i is the height of parent i and y_i is height of child i, and $i = 1, \cdots, n$
- **Predictions**: $\hat{y}_i = \beta_0 + \beta_1 x_i$ is model prediction of child height i
- **Loss on data**: Estimate parameters β_0 and β_1 by solving least squares problem (suppose the minimizers are $\hat{\beta}_0$ and $\hat{\beta}_1$)

$$\underset{\beta_0, \beta_1}{\text{minimize}} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \left(= \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2 \right)$$

- $RSS = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + \cdots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2$
- residual standard error (RSE)

$$RSE = \sqrt{\frac{1}{n - p - 1} RSS}$$

($p = 1$ in simple linear regression)

R^2 (a.k.a. coefficient of determination)

- To calculate R^2, we use the formula

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

where

- $RSS = \sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$ is residual sum of squares. $e_i = y_i - \widehat{y}_i$ is the ith residual.
- $TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2$ is total sum of squares
- $TSS - RSS = \sum_{i=1}^{n}(\widehat{y}_i - \bar{y})^2$
- This R^2 definition works for both simple linear regression and multiple linear regression
- R^2 is the percent of the variation in the response explained by the regression model; a common measure for how good a linear fit is.
- $0 \leq R^2 \leq 1$ Is a bigger R^2 better?

Note: Another set of notation is also common in textbooks. Total sum of squares (TSS) is often written as SST, and residual sum of squares (RSS) is also called error sum of squares (SSE). Most confusingly, in some other textbooks, $SSR = SST - SSE$ is sum of squares due to regression. So in this alternative universe of notations, $R^2 = SSR/SST$.

R^2 (a.k.a. coefficient of determination)

- To calculate R^2, we use the formula

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

where

- $RSS = \sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$ is residual sum of squares. $e_i = y_i - \widehat{y}_i$ is the ith residual.
- $TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2$ is total sum of squares
- $TSS - RSS = \sum_{i=1}^{n}(\widehat{y}_i - \bar{y})^2$
- This R^2 definition works for both simple linear regression and multiple linear regression
- R^2 is the percent of the variation in the response explained by the regression model; a common measure for how good a linear fit is.
- $0 \leq R^2 \leq 1$ Is a bigger R^2 better?

Note: Another set of notation is also common in textbooks. Total sum of squares (TSS) is often written as SST, and residual sum of squares (RSS) is also called error sum of squares (SSE). Most confusingly, in some other textbooks, $SSR = SST - SSE$ is sum of squares due to regression. So in this alternative universe of notations, $R^2 = SSR/SST$.

Correlation and R^2

- (Sample) correlation r between X and Y:

$$r(X, Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

- Measures the **linear** dependency between two numerical variables.

- For simple linear regression, $R^2 = r^2$. This relation does not extend to multiple linear regression. Why?

- In different application fields, *good* R^2 values are vastly different.

Back to Python implementation on the height data

```
X = height.drop('child', axis =1).values
y = height['child'].values
## DataFrame.values returns the NumPy representation of the Data Frame, and the axis label will be removed.
## Equivalently, one can use .to_numpy().
print(type(X)); print(type(y)) ## try to remove the .values and see the types
```

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

Figure 7: Data preparation

```
linear_model = LinearRegression(); linear_model.fit(X, y)
## if X were created by X = height['parent'] it is necessary to transform X
## by.reshape(-1,1) before calling the fit function, for the instructor's way,
## X is already two dimensional.
r_sq = linear_model.score(X, y); print('coefficient of determination:', r_sq)
```

```
coefficient of determination: 0.5712707984937204
```

```
print('intercept:', linear_model.intercept_); print('slope:', linear_model.coef_)
## You can notice that .intercept_ is a scalar, while .coef_ is an array.
```

```
intercept: 25.641176413281514
slope: [0.62145545]
```

Figure 8: Fitting a simple linear regression model

Making Predictions

```
: y_pred = linear_model.predict(X) ## making prediction on the training X.
```

```
: x_new = np.arange(50, 60).reshape((-1, 1))
  ## Making prediction on some new points
  ## Here the .reshape(-1,1) function is necessary to make a 1-D array two dimensional
```

```
: y_new = linear_model.predict(x_new); print(y_new)

  [56.71394872 57.33540416 57.95685961 58.57831506 59.1997705  59.82122595
   60.4426814  61.06413684 61.68559229 62.30704773]
```

Figure 9: Prediction on training and new x

But how do we find the minimizer in the loss function?

- We have use the Python `LinearRegression` as a blackbox to find the $\hat{\beta}_0$ and $\hat{\beta}_0$ that minimize the RSS.

- People in the precomputer age (or in exam settings) use an exact formula

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \text{ and } \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

  - remark: every least squares regression line passes $(\bar{x}, \bar{y})$

- Modern packages use optimization techniques. Not covered in DSO 530.