# Week09

October 26, 2021

#### 0.0.1 Multivariate regression

Up until now we have explored simple linear regression (SLR). SLR supposed a the conditional distribution of a random variable Y has a Normal distribution where the mean is a function of a random variable X and the variance is constant.

But often there may be many variables that change the probability distribution of Y.

Multivariate Linear Regression (MLR) is a statistical model that relates more than one random variable X (called covariates) to Y.

We will first rewrite SLR using a centered $X$ random variable $X*$ to simplify maximum likelihood estimates of $\beta_0$, $\beta_1$. Next, we will rewrite these MLEs using matrix algebra. Finally, we generalize these MLE estimating equations to more than one variable $X$.

**Centering on X** Our typical SLR setup assumes a dataset of $x, y$ pairs generated from pairs of random variables and a specific probability distribution for $Y$. Suppose we collect a sample of $N$ data points $\mathcal{D} = [(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)]$ which are generated from $N$ pairs of random variables $(X, Y)$.

SLR supposes the following conditional probability distriution for $Y$

$$Y|x, \beta, \sigma^2 \sim \mathcal{N}\left(\beta_0 + \beta_1 x, \sigma^2\right) \tag{1}$$

or

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i \tag{2}$$
$$\epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right) \tag{3}$$

Lets change the above SLR model by creating a new variable $x*$by subtracting from each $x_i$ the sample average $\bar{x}$. This is called centering on x and $x_i^* = x_i - \bar{x}$ is called a centered covariate.

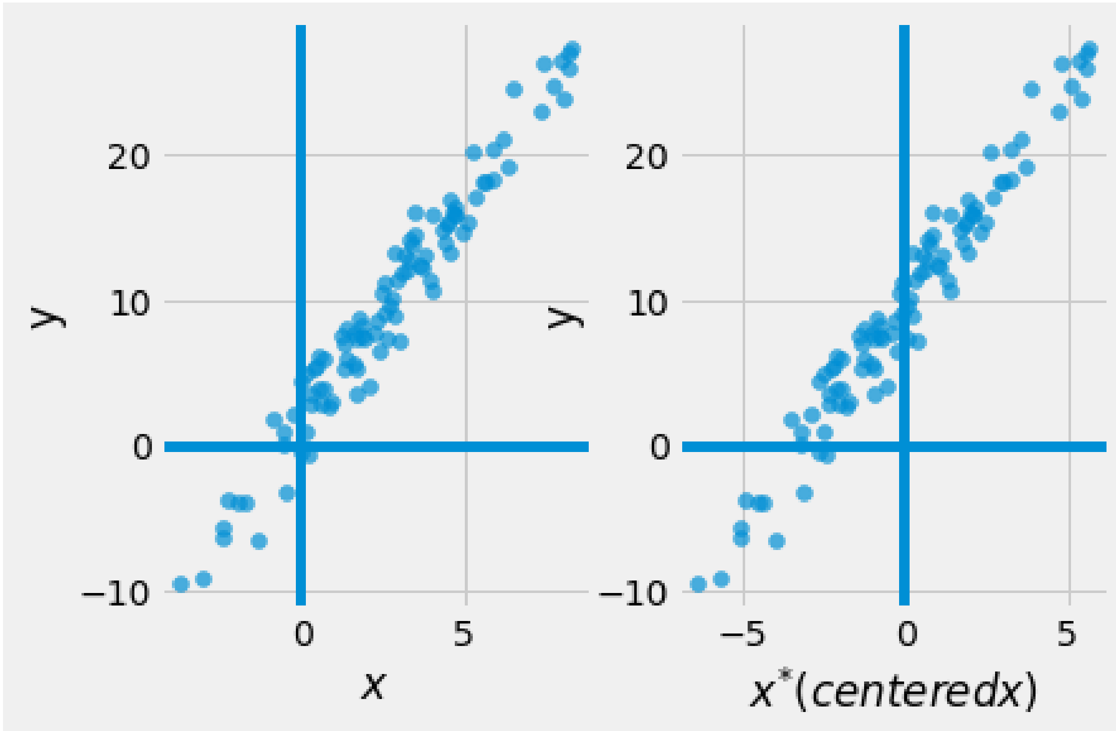
The SLR model relating $x^*$ and $Y$ is then

$$Y_i = \beta_0 + \beta_1 x_i^* + \epsilon_i \tag{4}$$
$$\epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right) \tag{5}$$

We can visualize the effect of centering below

[60]:
```
x     = np.random.normal(3,3,100) # generate 1000 points
xbar  = np.mean(x)
xstar = x-xbar

epsilons = np.random.normal(0,2,100)

y= 2+3*x + epsilons


fig,axs =plt.subplots(1,2)
ax=axs[0]
ax.scatter(x,y,alpha=0.7)
ax.set(xlabel=r"$x$",ylabel="y")

ax.axvline(0)
ax.axhline(0)


ax=axs[1]
ax.scatter(xstar,y,alpha=0.7)
ax.set(xlabel=r"$x^{*} (centered x)$",ylabel="y")


ax.axvline(0)
ax.axhline(0)
```

[60]:
```
<matplotlib.lines.Line2D at 0x1c902af70>
```




By centering, all of the (x,y) pairs in this particular case move to the left.

In our original SLR, we can compute the MLEs for $\beta_0$ and $\beta_1$ as

$$\hat{\beta_0} = \bar{y} - \beta_1 \bar{x} \tag{6}$$

$$\hat{\beta_1} = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N}(x_i - \bar{x})^2} = \frac{\sum_{i=1}^{N}(x_i - \bar{x})y_i}{\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{7}$$

Our new SLR model where we center $x$ simplifies the above estimates

$$\hat{\beta_0^*} = \bar{y} \tag{8}$$

$$\hat{\beta_1^*} = \frac{\sum_{i=1}^{N} x_i^* y_i}{\sum_{i=1}^{N}(x_i^*)^2} \tag{9}$$

Above we used the (very handy) fact that

$$
\begin{aligned}
\bar{x^*} &= \frac{\sum_{i=1}^{N}(x_i - \bar{x})}{N} && (10)\\
&= \frac{1}{N}\left(\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} \bar{x}\right) && (11)\\
&= \frac{1}{N}\left(\sum_{i=1}^{N} x_i - N\bar{x}\right) && (12)\\
&= \frac{1}{N}\left(N\bar{x} - N\bar{x}\right) && (13)\\
&= 0 && (14)
\end{aligned}
$$

**Are $\beta_1$ and $\beta_1^*$ equivalent?** Shifting all of our collected xvalues by the same constant does not change the relationship between Y and X. We can see that this is the case two ways.

Algebraic

$$
\begin{aligned}
Y_i &= \beta_0^* + \beta_1^* x_i^* + \epsilon_i && (15)\\
&= \beta_0^* + \beta_1^*(x_i - \bar{x}) + \epsilon_i && (16)\\
&= \beta_0^* - \beta_1^*\bar{x} + \beta_1^* x_i + \epsilon_i && (17)\\
\epsilon_i &\sim \mathcal{N}\left(0, \sigma^2\right) && (18)\\
& && (19)
\end{aligned}
$$

We see that our $\beta_1^*$ is attached to $x$ in the same way as our original SLR. This then means $\beta_1^* = \beta_1$

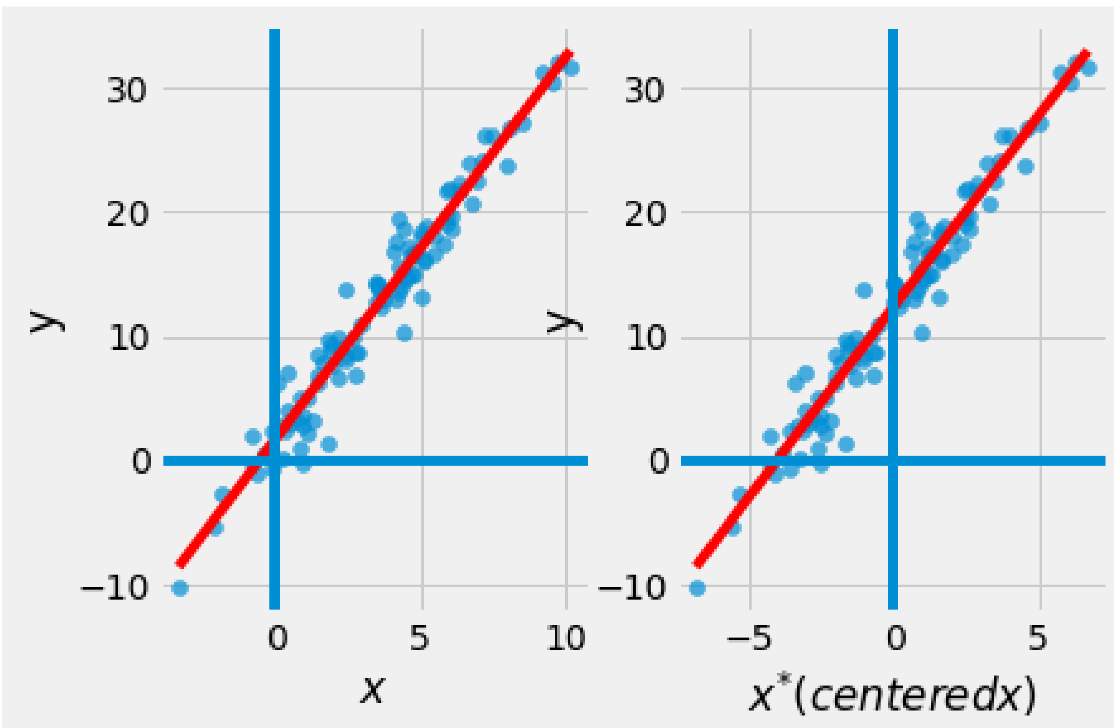
Visually

We can plot the points $(x, y)$ and $(x^*, y)$ and see that the estimated **slope** remains unchanged.

[61]:
```
x     = np.random.normal(3,3,100) # generate 1000 points
xbar  = np.mean(x)
xstar = x-xbar

epsilons = np.random.normal(0,2,100)

y= 2+3*x + epsilons

fig,axs =plt.subplots(1,2)
ax=axs[0]
ax.scatter(x,y,alpha=0.7)
ax.set(xlabel=r"$x$",ylabel="y")

minx,maxx = min(x),max(x)
b1,b0 = np.polyfit(x,y,1)
ax.plot([minx,maxx],[b0+b1*minx,b0+b1*maxx],color="red" )
```

```
ax.axvline(0)
ax.axhline(0)

ax=axs[1]
ax.scatter(xstar,y,alpha=0.7)
ax.set(xlabel=r"$x^{*} (centered x)$",ylabel="y")

minx,maxx = min(xstar),max(xstar)
b1,b0 = np.polyfit(xstar,y,1)
ax.plot([minx,maxx],[b0+b1*minx,b0+b1*maxx],color="red" )

ax.axvline(0)
ax.axhline(0)
```

[61]: <matplotlib.lines.Line2D at 0x1c917fbe0>




**Are $\beta_0$ and $\beta_0^*$ equivalent?** The intercept for the SLR and SLR with centered $x$ are **different**. From above, we saw that the SLR with centered x simplifed to ###### Algebraic

$$Y_i = \beta_0^* - \beta_1^* \bar{x} + \beta_1^* x_i + \epsilon_i \tag{20}$$
$$\epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right) \tag{21}$$
$$\tag{22}$$

The expression $\beta_0^* - \beta_1^* \bar{x}$ involve only constants. This expression is $\beta_0$. That is, $\beta_0 = \beta_0^* - \beta_1^* \bar{x}$

Visually

We see from above that the estimated regression line (red line) intersects the y-axis at different points depending on whether we center or do not center our data.

Though centering makes computing MLEs easier this process also changes our $\beta_0$ or y-intercept.

### 0.0.2 Matrix algebra to compute the MLEs for centered SLR

Let us look again at the MLE estimates for sentered SLR

$$
\begin{align}
\hat{\beta_0^*} &= \bar{y} \tag{23}\\
\hat{\beta_1^*} &= \frac{\sum_{i=1}^{N} x_i^* y_i}{\sum_{i=1}^{N} (x_i^*)^2} \tag{24}
\end{align}
$$

The MLE for $\hat{\beta_1^*}$ appears to be two inner products divided by one another. That is, we can compute $\hat{\beta_1^*}$ as

$$
\hat{\beta_1^*} = \frac{x^* \cdot y}{x^* \cdot x^*} \tag{25}
$$

where $x \cdot y = \sum_{i=1}^{N} x_i y_i$ and

$$
x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix} \tag{26}
$$

We can also recognize the MLE for $\hat{\beta_0^*}$ as an inner product

$$
\begin{align}
\hat{\beta_0^*} &= \bar{y} \tag{27}\\
&= \frac{1}{N} \sum_{i=1}^{N} y_i \tag{28}\\
&= \frac{1}{N} \sum_{i=1}^{N} 1 \times y_i \tag{29}\\
&= \frac{1}{N} \mathbb{1} \cdot y \tag{30}
\end{align}
$$

where

$$\mathbb{1} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{31}$$

Using matrix algebra, our MLEs become

$$\hat{\beta}_0^* = (\mathbb{1} \cdot \mathbb{1})^{-1}(\mathbb{1} \cdot y) \tag{32}$$
$$\hat{\beta}_1^* = (x^* \cdot x^*)^{-1}(x^* \cdot y) \tag{33}$$

**A data setup such that we arrive at the above MLEs** The above MLE estimates pop out of the following data setup. Let $X$ be a matrix with $N$ rows and 2 columns. The first column will be all ones and the second column will be filled, from top to bottom, with $x_1, x_2, \cdots, x_N$.

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} \tag{34}$$

The above MLE formulas may motivate us to try

$$(X'X)^{-1}X'y \tag{35}$$

$$X'y = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & \cdots & x_N \end{bmatrix} \times \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} \mathbb{1}'y \\ x'y \end{bmatrix} \tag{36}$$

The first multiplication looks like a step in the right direction.

$$X'X = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & \cdots & x_N \end{bmatrix} \times \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} = \begin{bmatrix} \mathbb{1}'\mathbb{1} & \mathbb{1}'x \\ \mathbb{1}'x & x'x \end{bmatrix} \tag{37}$$

If we centered our x covariate then $\bar{x}$ is zero and so

$$\mathbb{1}'x = \sum_{i=1}^{N} x_i = N\bar{x} = 0 \tag{38}$$

We can use the above to simplify $X'X$

$$X'X = \begin{bmatrix} \mathbb{1}'\mathbb{1} & 0 \\ 0 & x'x \end{bmatrix} \tag{39}$$

The matrix $X'X$ is now a diagonal matrix—a matrix with non-zero elements for the entries $(i, i)$. The inverse of a diagonal matrix is easy to compute

$$(X'X)^{-1} = \begin{bmatrix} 1\Big/\mathbb{1}'\mathbb{1} & 0 \\ 0 & 1\Big/x'x \end{bmatrix} \tag{40}$$

and we can now see our final product is

$$(X'X)^{-1}X'y = \begin{bmatrix} 1\Big/\mathbb{1}'\mathbb{1} & 0 \\ 0 & 1\Big/x'x \end{bmatrix} \begin{bmatrix} \mathbb{1}'y \\ x'y \end{bmatrix} = \begin{bmatrix} \mathbb{1}'y\Big/\mathbb{1}'\mathbb{1} \\ x'y\Big/x'x \end{bmatrix} \tag{41}$$

We can see that the above matrix multiplication returns a vector with two MLEs: $\hat{\beta}_0^*$ and $\hat{\beta}_1^*$.

**Variance for the MLEs follows a similar pattern.** We saw in simple linear regression that the variance for $\hat{\beta}_1$ was

$$\mathbb{V}(\hat{\beta}_1) = \sigma^2 \frac{1}{\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{42}$$

and the variance for $\hat{\beta}_0$

$$\mathbb{V}(\hat{\beta}_0) = \sigma^2 \left( \frac{1}{N} + \frac{1}{\sum_{i=1}^{N}(x_i - \bar{x})^2} \right) \tag{43}$$

There is a natural generaliztion for the variance of $\hat{\beta}_0$ and $\hat{\beta}_1$ like the above generaliztion for the MLEs.

$$\mathbb{V}(\hat{\beta}) = \sigma^2 (X'X)^{-1} \tag{44}$$

The above variance will return a **covariance matrix** where the variance of $be\hat{t}a_0$ is the first diagonal entry and the variance of $\hat{\beta}_1$ is the second diagonal entry.

### 0.0.3 Multivariate Linear Regression (finally) and an example

Suppose we collect data points

$$\mathcal{D} = [(x_1^1, x_1^2, x_1^3, y_1), (x_2^1, x_2^2, x_2^3, y_2), \cdots, (x_N^1, x_N^2, x_N^3, y_N)] \tag{45}$$

and propose that the conditional distribution of $Y_i$ is

$$Y_i | x_i^1, x_i^2, \cdots, x_i^3 \sim \mathcal{N}\left(\beta_0 + \beta_1 x_i^1 + \beta_2 x_i^2 + \beta_3 x_i^3, \sigma^2\right) \tag{46}$$

The above is **multivariate linear regression** model. We can compute the MLE for $\hat{\beta}$ as

$$\hat{\beta} = (X'X)^{-1}X'y \tag{47}$$

and we can compute the variance for our MLEs as

$$\mathbb{V}(\hat{\beta}) = \sigma^2(X'X)^{-1} \tag{48}$$

[62]:
```
from sklearn.datasets import load_diabetes
import numpy.linalg as la


diabData = load_diabetes()
X = diabData["data"]
y = diabData["target"]

# add intercept (column of ones)
X = np.column_stack([np.ones((len(X),1)),X]  )
```

[63]:
```
betaHat = la.solve( X.T.dot(X), X.T.dot(y))
betaHat

epsilons = y - X.dot(betaHat)
sigma2Hat = epsilons.T.dot(epsilons) / len(epsilons)

varBetaHat = sigma2Hat * la.inv(X.T.dot(X))
```

[64]:
```
import statsmodels.api as sm
model = sm.OLS(y,X)
res = model.fit().summary()
res
```

[64]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.518
```

```
Model:                            OLS   Adj. R-squared:                  0.507
Method:                 Least Squares   F-statistic:                     46.27
Date:                Tue, 26 Oct 2021   Prob (F-statistic):           3.83e-62
Time:                        00:27:29   Log-Likelihood:                -2386.0
No. Observations:                 442   AIC:                             4794.
Df Residuals:                     431   BIC:                             4839.
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        152.1335      2.576     59.061      0.000     147.071     157.196
x1           -10.0122     59.749     -0.168      0.867    -127.448     107.424
x2          -239.8191     61.222     -3.917      0.000    -360.151    -119.488
x3           519.8398     66.534      7.813      0.000     389.069     650.610
x4           324.3904     65.422      4.958      0.000     195.805     452.976
x5          -792.1842    416.684     -1.901      0.058   -1611.169      26.801
x6           476.7458    339.035      1.406      0.160    -189.621    1143.113
x7           101.0446    212.533      0.475      0.635    -316.685     518.774
x8           177.0642    161.476      1.097      0.273    -140.313     494.442
x9           751.2793    171.902      4.370      0.000     413.409    1089.150
x10           67.6254     65.984      1.025      0.306     -62.065     197.316
==============================================================================
Omnibus:                        1.506   Durbin-Watson:                   2.029
Prob(Omnibus):                  0.471   Jarque-Bera (JB):                1.404
Skew:                           0.017   Prob(JB):                        0.496
Kurtosis:                       2.726   Cond. No.                         227.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[65]:

```python
print(betaHat[1])
print(np.sqrt(varBetaHat[1,1]))
```

```
-10.012197817471035
59.00101633853383
```

[ ]:

```python

```