Lecture 2 - Statistical Learning and Linear Regression

DTU Management Engineering
Department of Management Engineering

Outline

- Statistical Learning
- Regression: Data and Notation
- Bivariate Regression Model
- Ordinary Least Squares
- Multivariate Regression Model
- Ridge Regression and the Lasso

What is Statistical Learning?

- Given a quantitative response variable Y and a collection of r predictor variables $X = (X_1, X_2, ..., X_r)$

What is Statistical Learning?

- Given a quantitative response variable Y and a collection of r predictor variables $X = (X_1, X_2, ..., X_r)$
- Assume there is a relationship between them:

$$Y = f(X) + \epsilon$$

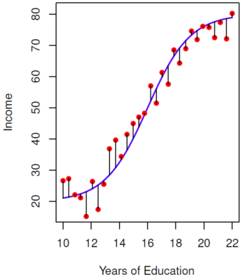
- f is a fixed but unknown function of the predictors $X_1, X_2, ..., X_r$
- ϵ is a random *error* term with zero mean (why do we need it?)

What is Statistical Learning?

- Given a quantitative response variable Y and a collection of r predictor variables $X = (X_1, X_2, ..., X_r)$
- Assume there is a relationship between them:

$$Y = f(X) + \epsilon$$

- f is a fixed but unknown function of the predictors $X_1, X_2, ..., X_r$
- ϵ is a random *error* term with zero mean (why do we need it?)



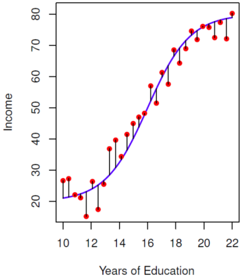
Picture from James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 17). New York: springer.

What is Statistical Learning?

- Given a quantitative response variable Y and a collection of r predictor variables $X = (X_1, X_2, ..., X_r)$
- Assume there is a relationship between them:

$$Y = f(X) + \epsilon$$

- f is a fixed but unknown function of the predictors $X_1, X_2, ..., X_r$
- ϵ is a random *error* term with zero mean (why do we need it?)



Picture from James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 17). New York: springer.

- *Statistical learning*: approaches to estimate f

Why estimate f?

- Two reasons: *prediction* and *inference*

Why estimate f?

- Two reasons: *prediction* and *inference*
- *Prediction*: inputs X available but not Y; we can predict Y using

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

where $\hat{f}$ represents our estimate of f and $\hat{Y}$ represents the resulting prediction for Y.

Why estimate f?

- Two reasons: *prediction* and *inference*
- *Prediction*: inputs X available but not Y; we can predict Y using

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

  where $\hat{f}$ represents our estimate of f and $\hat{Y}$ represents the resulting prediction for Y.

  Here, $\hat{f}$ is often treated as a *black box* – we are not concerned with the exact form of $\hat{f}$ as long as it yields accurate predictions.

Why estimate f?

- Two reasons: *prediction* and *inference*
- *Prediction*: inputs X available but not Y; we can predict Y using

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

where $\hat{f}$ represents our estimate of f and $\hat{Y}$ represents the resulting prediction for Y.

Here, $\hat{f}$ is often treated as a *black box* – we are not concerned with the exact form of $\hat{f}$ as long as it yields accurate predictions.

Accuracy of $\hat{Y}$ depends on two quantities:

  - Reducible error: $\hat{f}$ will in general not be a perfect estimate of f - however, this error is reducible

Why estimate f?

- Two reasons: *prediction* and *inference*
- *Prediction*: inputs X available but not Y; we can predict Y using

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

  where $\hat{f}$ represents our estimate of f and $\hat{Y}$ represents the resulting prediction for Y.

  Here, $\hat{f}$ is often treated as a *black box* – we are not concerned with the exact form of $\hat{f}$ as long as it yields accurate predictions.

  Accuracy of $\hat{Y}$ depends on two quantities:

  - Reducible error: $\hat{f}$ will in general not be a perfect estimate of f - however, this error is reducible
  - Irreducible error: even if we had the perfect f, i.e., $\hat{Y} = f(X)$, there will still be some error in $\hat{Y}$ due to ϵ

Why estimate f? (contd.)

- *Inference*:
    - Which predictors are associated with the response?
        - Identifying important predictors among a large set of variables.
    - What is the relationship between the response and each predictor?
    - What is the nature of the relationship (linear, non-linear, etc.)?

How do we estimate f? (contd.)

- *Parametric approaches*:
    - Make an assumption about the functional form of f
    - Reduce the problem of estimating f to that of estimating a set of parameters
    - Eg.: linear/non-linear regression, parametric logistic regression

How do we estimate f? (contd.)

- *Parametric approaches*:
  - Make an assumption about the functional form of f
  - Reduce the problem of estimating f to that of estimating a set of parameters
  - Eg.: linear/non-linear regression, parametric logistic regression
- *Non-parametric approaches*:
  - Do not make explicit assumptions about the functional form of f
  - Eg.: K-nearest neighbours, decision trees, support vector machines
- We will cover both during the course

Assessing model accuracy

- In a regression setting, we often use the *Mean Squared Error* (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\Big(y_i - \hat{f}(x_i)\Big)^2$$

where $\hat{f}(x_i)$ is the prediction for the ith observation in the *training set*.

Assessing model accuracy

- In a regression setting, we often use the *Mean Squared Error* (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2$$

  where $\hat{f}(x_i)$ is the prediction for the ith observation in the *training set*.

- Above is computed using the training data, i.e., *training MSE* - often not of much interest

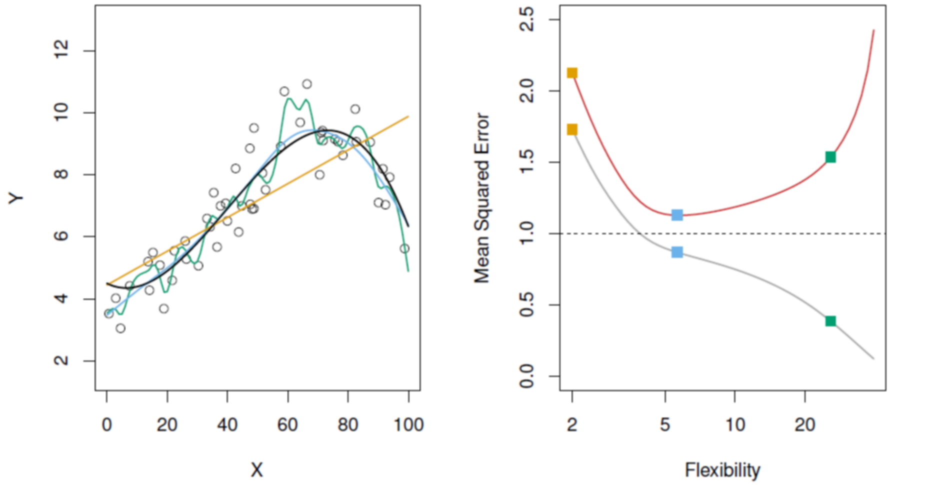
Assessing model accuracy

- In a regression setting, we often use the *Mean Squared Error* (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2$$

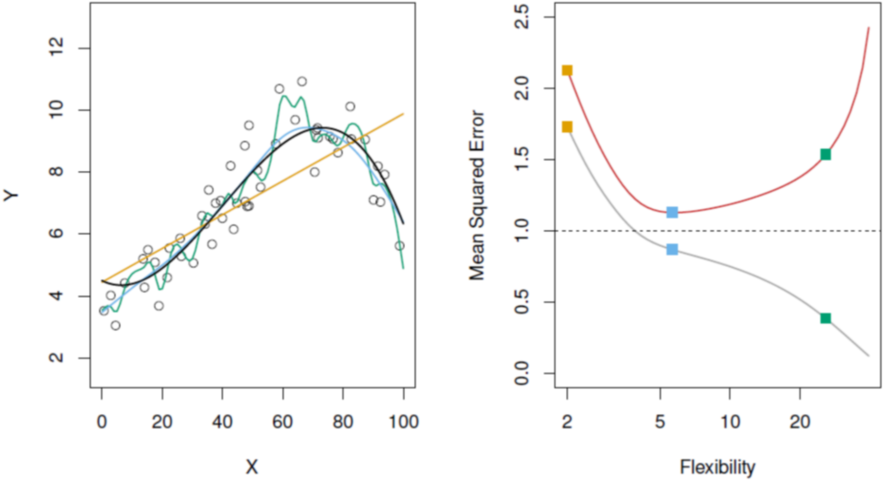
  where $\hat{f}(x_i)$ is the prediction for the ith observation in the *training set*.
- Above is computed using the training data, i.e., *training MSE* - often not of much interest
- Interested in the accuracy of the method when applied to previously unseen *test* data - *test MSE*

Assessing model accuracy - training vs test MSE



Picture from James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 31). New York: springer.

Assessing model accuracy - training vs test MSE



Picture from James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 31). New York: springer.

- Lower training MSE does not guarantee lower test MSE. Why?

The Bias-Variance decomposition

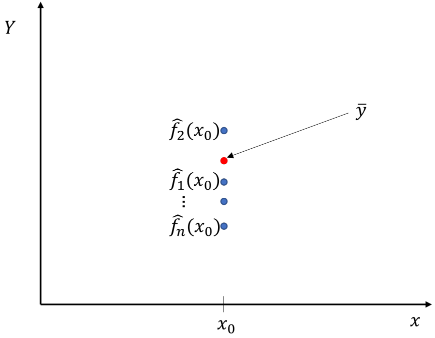
- Assume we have n different training sets $1, 2...n$ and we estimate f using each of these training sets to get: $\hat{f}_1, \hat{f}_2...\hat{f}_n$

The Bias-Variance decomposition

- Assume we have n different training sets $1, 2...n$ and we estimate f using each of these training sets to get: $\hat{f}_1, \hat{f}_2...\hat{f}_n$
- Now consider an arbitrary new test point x_0; our predictions are $\hat{f}_1(x_0), \hat{f}_2(x_0)...\hat{f}_n(x_0)$

The Bias-Variance decomposition

- Assume we have n different training sets $1, 2 \ldots n$ and we estimate f using each of these training sets to get: $\hat{f}_1, \hat{f}_2 \ldots \hat{f}_n$
- Now consider an arbitrary new test point x_0; our predictions are $\hat{f}_1(x_0), \hat{f}_2(x_0) \ldots \hat{f}_n(x_0)$



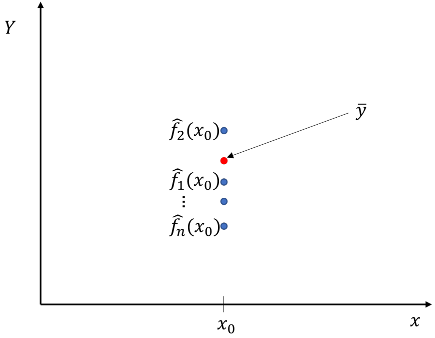
$$\bar{f}(x_0) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}_i(x_0)$$

$$Bias(\hat{f}(x_0)) = \left(\bar{y} - \bar{f}(x_0)\right)$$

$$Var(\hat{f}(x_0)) = \frac{1}{n}\sum_{i=1}^{n} \left(\hat{f}_i(x_0) - \bar{f}(x_0)\right)^2$$

The Bias-Variance decomposition

- Assume we have n different training sets $1, 2...n$ and we estimate f using each of these training sets to get: $\hat{f}_1, \hat{f}_2...\hat{f}_n$
- Now consider an arbitrary new test point x_0; our predictions are $\hat{f}_1(x_0), \hat{f}_2(x_0)...\hat{f}_n(x_0)$



$$\bar{f}(x_0) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}_i(x_0)$$

$$Bias(\hat{f}(x_0)) = \left(\bar{y} - \bar{f}(x_0)\right)$$

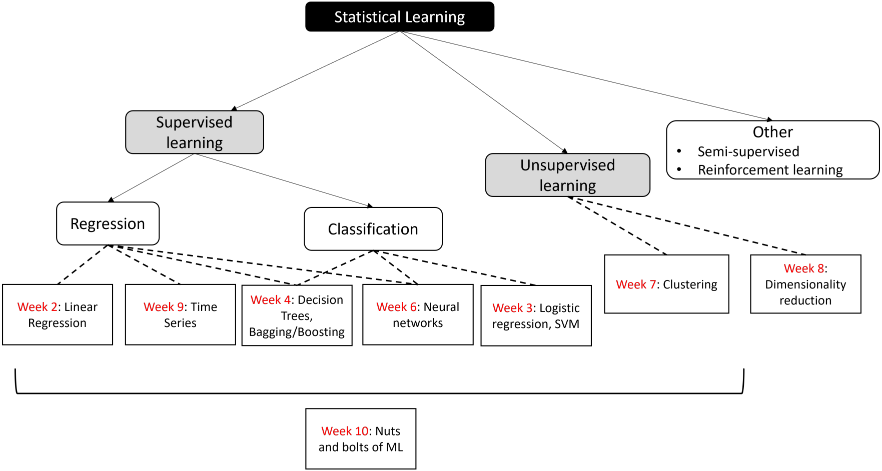
$$Var(\hat{f}(x_0)) = \frac{1}{n}\sum_{i=1}^{n} \left(\hat{f}_i(x_0) - \bar{f}(x_0)\right)^2$$

- Expected test MSE at $x_0 = \left[Bias(\hat{f}(x_0))\right]^2 + Var(\hat{f}(x_0)) + Var(\epsilon)$

The Bias-Variance tradeoff

- Ideally, we want to minimize both bias and variance to minimize the test MSE
- In reality, there is a tradeoff, reducing one increases the other and vice versa - our goal is to strike a balance
- Generally, more complex/flexible models (green curve in the example) have high variance and low bias (small changes in the training data can result in large changes in $\hat{f}$)
- Simpler models (orange line in the example) have a relatively higher bias but smaller variance

Overview of the coming weeks



Playtime!

- Open the “2. Intro_Stat_Learning.ipynb" notebook.
- Expected duration: 20 min

New York City taxi demand problem [1]

- (Almost) All taxi trips in NYC since 2009
- Original files have one trip per line:
  - Pick-up location, time
  - Drop-off location, time
- Research question: **predict taxi pick-ups across the city**
- Useful to optimize taxi service
  - Similar to many other demand problems (shared modes, public transport, energy, water, goods, communication...)

[1] Full dataset at http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

New York City taxi demand problem (cont'd)

- Let's further specify the research question: **predict taxi pick-ups for different zones in 15 min periods**
- We group by zone and time:
  - There are 29 zones[3]
  - Each zone has its own file
  - Each line is the sum of pick-ups/drop-offs in a 15 minute period

[3] To know what zones we have, check http://inigoreiriz.github.io/NYC_Cor/d3map_O/index.html

Regression and Notation

- Regression - predict response variable Y from a collection of r predictor variables $X_1, X_2, ..., X_i, ..., X_r$

- where, Y - dependent variable
  X_i - predictor, explanatory or independent variable(s)
- In the NYC taxi example,

$$Y = pickups$$
$$X_1, X_2, ..., X_i, ..., X_r = ??$$

- An observation j is a vector (with all predictor variables) denoted as $\mathbf{x}_j$

Regression and Notation

- Regression - predict response variable Y from a collection of r predictor variables $X_1, X_2, ..., X_i, ..., X_r$

- where, Y - dependent variable
  X_i - predictor, explanatory or independent variable(s)

- An observation j is a vector (with all predictor variables) denoted as $\mathbf{x}_j$
- In the NYC taxi example,

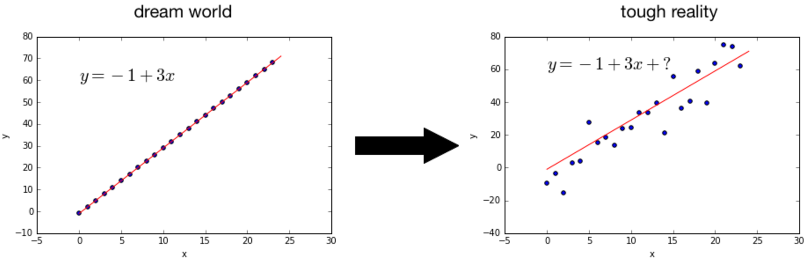
$$Y = pickup$$
$$X = hour$$

Bivariate Linear Regression Model

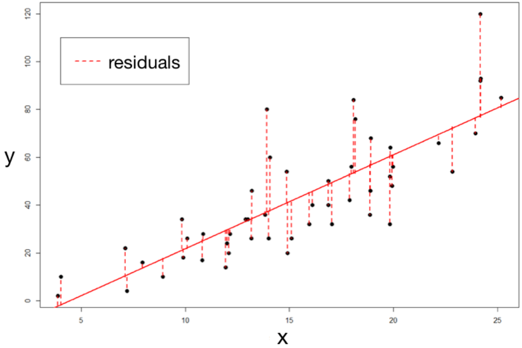
- A linear relationship between X and Y only means that there should be a function

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

- **But**, unless the data perfectly aligns, this may not work in that exact way! :-(



Linear Regression Model (cont'd)



- The difference between the data and the model prediction is called **residual**

Linear Regression Model (cont'd)

- For each value of Y, the true expression should be:

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

- The error ϵ is a **random variable**. Usually, we assume $\epsilon \sim \mathcal{N}(0, \sigma^2)$
- As a consequence, Y is also a random variable
- Thus, another way to put it is

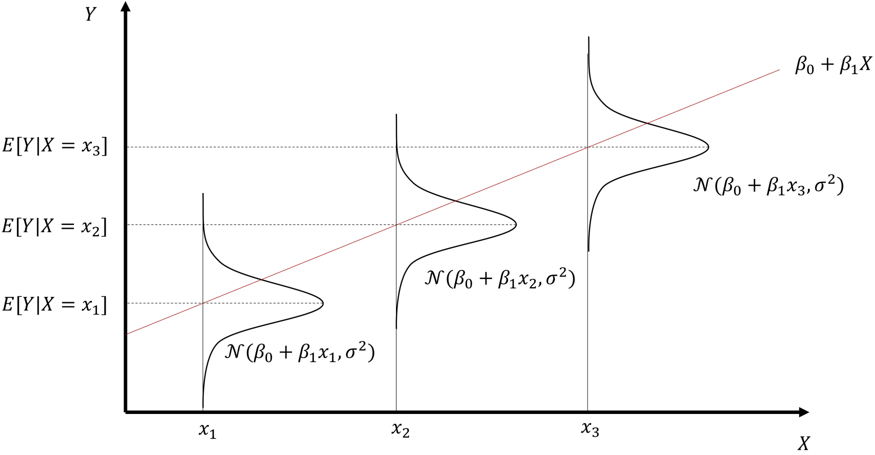
$$Y \sim \beta_0 + \beta_1 X + \mathcal{N}(0, \sigma^2)$$

[Response] **distributed as** [mean (depending on x)] + [error]

- More compactly

$$Y \sim \mathcal{N}(\beta_0 + \beta_1 X, \sigma^2)$$

Linear Regression Model (cont'd)

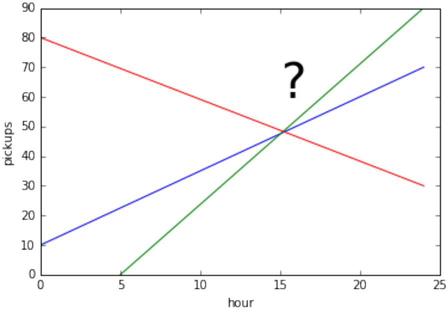


Playtime!

- Open “2 - regression.ipynb" notebook.
- Do the Part 1
- Expected duration: 30 min

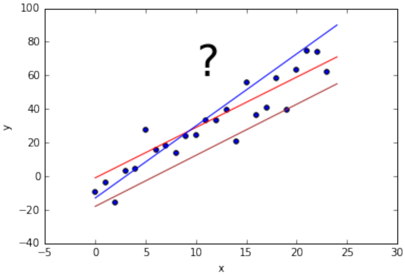
Bivariate Linear Regression Model

- We seem to believe that there is a (linear) relationship between the value of "hour" variable and "pickups"



Bivariate Linear Regression Model (cont'd)

- Recall that there will be some **error** in our model
- We want to find the model where this error is smallest
    - i.e. one that generates the smallest residuals
- Fine, but which one is it?



Least Squares estimator

- Model:

$$y_j = \beta_0 + \beta_1 x_j + \epsilon_j, \forall j = 1, 2, ..., n$$

- So,

$$\epsilon_j = y_j - (\beta_0 + \beta_1 x_j)$$

- Find β_0, β_1 that minimize the sum of squared errors:

$$S = \sum_{j=1}^{n} [y_j - (\beta_0 + \beta_1 x_j)]^2$$

- At the minimum of S, we have

$$\frac{\partial S}{\partial \beta_0} = 0$$

and

$$\frac{\partial S}{\partial \beta_1} = 0$$

Least Squares estimator

- The estimated values are denoted $\hat{\boldsymbol{\beta}} = [\hat{\beta}_0, \hat{\beta}_1]^T$
- Closed form expression for $\hat{\boldsymbol{\beta}}$ (derivation in Appendix):

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x}$$

$$\hat{\beta}_1 = \frac{\sum_{j=1}^{n} (y_j - \overline{y})(x_j - \overline{x})}{\sum_{j=1}^{n} (x_j - \overline{x})^2}$$

where, $\overline{x} = \frac{1}{n} \sum_{j=1}^{n} x_j, \overline{y} = \frac{1}{n} \sum_{j=1}^{n} y_j$

- It defines the **Least Squares estimator** for y, also represented as $\hat{y}$, with

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

Playtime!

- Part 2 of “4 - regression.ipynb" notebook.
- Expected duration: 30 min

Before we proceed..

- Matrix notation - it'll be everywhere! ;-)

Model in Matrix Notation

- With n independent observations on y_j and the associated values of x_j, the complete model:

$$\begin{aligned} y_1 &= \beta_0 + \beta_1 x_1 + \epsilon_1 \\ y_2 &= \beta_0 + \beta_1 x_2 + \epsilon_2 \\ &\vdots \\ y_n &= \beta_0 + \beta_1 x_n + \epsilon_n \end{aligned}$$

- In matrix notation:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

Model in Matrix Notation

- With n independent observations on y_j and the associated values of x_j, the complete model:

$$
\begin{aligned}
y_1 &= \beta_0 + \beta_1 x_1 + \epsilon_1 \\
y_2 &= \beta_0 + \beta_1 x_2 + \epsilon_2 \\
&\vdots \\
y_n &= \beta_0 + \beta_1 x_n + \epsilon_n
\end{aligned}
$$

- In matrix notation:

$$
\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}
$$

$$
\mathbf{y} = \mathbf{X}\beta + \epsilon
$$

```
regr=linear_model.LinearRegression(fit_intercept=False)
x_= np.c_[np.ones(len(x)),np.array(x)]
regr.fit(x_, y)
print("sklearn linear regression:", regr.coef_)
```

Matrix Notation

- The least squares method minimizes S

$$S = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

- Closed form expression for **least squares estimates** $\hat{\beta}$ (proof in Appendix):

$$\boxed{\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}}$$

- $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$ -> $\hat{\mathbf{y}}$ are the **predictions** for data points $\mathbf{X}$
- $\hat{\boldsymbol{\epsilon}} = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}$ -> **residuals**.

Ordinary Least Squares

- The error terms are assumed to satisfy:
    1. $E(\epsilon_j) = 0$
    2. $Var(\epsilon_j) = \sigma^2$ (constant)
    3. $Cov(\epsilon_j, \epsilon_k) = 0, \ \forall j \neq k$
- **Ordinary Least Squares Model**:

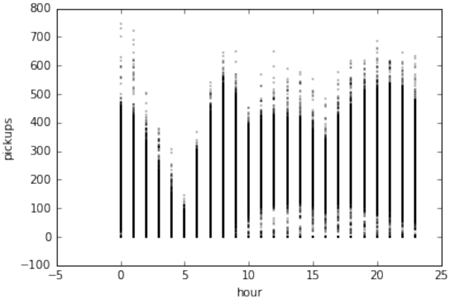
$$\boxed{\underset{(n\times 1)}{\boldsymbol{Y}} = \underset{(n\times 2)}{\mathbf{X}} \underset{(2\times 1)}{\boldsymbol{\beta}} + \underset{(n\times 1)}{\boldsymbol{\epsilon}}}$$

$$\boxed{E(\boldsymbol{\epsilon}) = \underset{(n\times 1)}{0} \quad and \ Cov(\boldsymbol{\epsilon}) = \sigma^2 \underset{(n\times n)}{\mathbf{I}}}$$

Back to our example

Back to our problem: **predicting taxi pick-ups in NYC**

- Our x is "hours" and our y is "pickup". The corresponding plot:



Back to our example

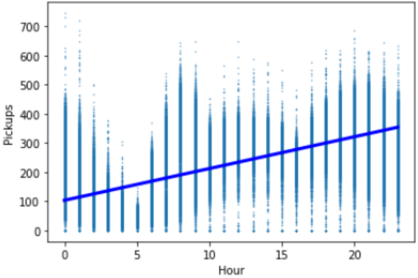
In [12]:
```
x= np.c_[np.ones(len(f)),f['hour']]
y= np.array(f['pickups'], ndmin=2).T
print (x.shape,y.shape)
```

```
(262848, 2) (262848, 1)
```

In [14]:
```
regr=linear_model.LinearRegression(fit_intercept=False)
regr.fit(x, y)
```

Out[14]:
```
LinearRegression(fit_intercept=False)
```

In [15]:
```
plt.scatter(f['hour'], y, s=0.1)
plt.plot(f['hour'], regr.predict(x),color='blue',linewidth=3)
plt.xlabel("Hour")
plt.ylabel("Pickups")
plt.show()
```



back to our example

- Is this model accurate enough?

```
In [39]: print("Average error (AE): %.2f" % np.mean(regr.predict(x) - y))
         print("Mean Absolute error (MAE): %.2f"% np.mean(abs(regr.predict(x) - y)))
         # The mean squared error
         print("Root Mean squared error: %.2f"
               % np.sqrt(np.mean((regr.predict(x) - y) ** 2)))

         Average error (AE): 0.00
         Mean Absolute error (MAE): 80.60
         Root Mean squared error: 99.83
```

Why is "average error" an (almost) irrelevant measure?

- Is there other data we can use to improve?
    - We need to get multivariate!

Playtime!

- Part 3 of “4 - regression.ipynb" notebook.
- Expected duration: 30 min

Multivariate Linear Regression Model

- Multivariate linear regression model (linear in the parameters β_i)

$$\boxed{Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_r X_r + \varepsilon}$$

- where,
  Y — number of taxi pick-ups in a period of 15 minutes (**dependent variable**)

  X_1 — hours
  X_2 — minutes
  $\ldots$
  (**predictor variables**)

- ε — **error term**

Formulation

- For n independent observations with r predictor variables

$$
\begin{aligned}
y_1 &= \beta_0 + \beta_1 x_{11} + \cdots + \beta_r x_{1r} + \varepsilon_1 \\
y_2 &= \beta_0 + \beta_1 x_{21} + \cdots + \beta_r x_{2r} + \varepsilon_2 \\
&\vdots \\
y_n &= \beta_0 + \beta_1 x_{n1} + \cdots + \beta_r x_{nr} + \varepsilon_n
\end{aligned}
$$

- Errors terms

$$
\begin{aligned}
E\left(\varepsilon_j\right) = 0,\ \operatorname{Var}\left(\varepsilon_j\right) = \sigma^2 (\text{const}) \\
\operatorname{Cov}\left(\varepsilon_j, \varepsilon_k\right) = 0, j \neq k, \forall j, k = 1, ..., n
\end{aligned}
$$

Matrix Notation

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1r} \\ 1 & x_{21} & \cdots & x_{2r} \\ 1 & x_{31} & \cdots & x_{3r} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nr} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_r \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

$$\boxed{\underset{(n\times 1)}{\boldsymbol{Y}} = \underset{(n\times(r+1))}{\mathbf{X}} \underset{((r+1)\times 1)}{\boldsymbol{\beta}} + \underset{(n\times 1)}{\boldsymbol{\varepsilon}}}$$

$$\boxed{\underset{(n\times 1)}{E(\boldsymbol{\varepsilon})} = \underset{(n\times 1)}{\mathbf{0}} \qquad \underset{(n\times n)}{\mathrm{Cov}(\boldsymbol{\varepsilon})} = \underset{(n\times n)}{\sigma^2\mathbf{I}}}$$

Multivariate Linear Regression

- The least squares method minimizes S

$$S = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

$$\boxed{\underset{((r+1)\times 1)}{\hat{\boldsymbol{\beta}}} = \left(\underset{((r+1)\times n)}{\mathbf{X}^T}\underset{(n\times(r+1))}{\mathbf{X}}\right)^{-1}\underset{((r+1)\times n)}{\mathbf{X}^T}\underset{(n\times 1)}{\mathbf{y}}}$$

- Derivation in the appendix

Multivariate NYC model

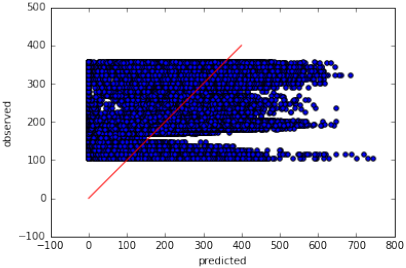
- Let's now try a model with two variables:
  - hours
  - minutes
- Result:

$$pickups = 102.6 + 10.9 * hours + 0.038 * minutes$$

  (previously, the best one was $pickups = 103.5 + 10.9 * hours$.
- Not much of a difference?

Multivariate NYC model

- A 45 degree plot (observed VS predicted):



- Horrible! How are the error statistics?

```
print("Mean Absolute error (MAE): %.2f"% np.mean(abs(regr.predict(x) - y)))
# The mean squared error
print("Root Mean squared error: %.2f"
      % np.sqrt(np.mean((regr.predict(x) - y) ** 2)))

Mean Absolute error (MAE): 80.60
Root Mean squared error: 99.83
```

- Our model is (still) pretty bad!... :-(

Multivariate NYC model

- Hmm, after all, how correlated are our variables?...

```
In [180]: f.corr()
Out[180]:
```

	hour	minute	pickups
hour	1.000000	0.000000	0.603024
minute	0.000000	1.000000	0.005164
pickups	0.603024	0.005164	1.000000

- "hours" seems to have a little correlation (0.6), but "minutes" is practically zero! ...what a surprise... |-O

Multivariate NYC model

- If we want to predict to the near future (for example, 15 minutes ahead), we actually have some information!
- For example, we should know the value for **now**, and 15 minutes ago. Why don't we use it?

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \epsilon_t$$

- These variables are often called **lagged**
- This is a simple type of **time series** linear model!

Multivariate NYC model

- After some data wrangling, we can get a new dataframe, with the extra information

```
In [184]: f_lagged.head()
```

Out[184]:

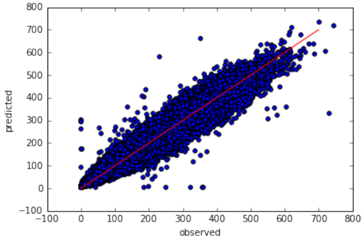
	date	hour	minute	pickups	pickups_lag1	pickups_lag2
2	2009-01-01	0	30	215	166	0
3	2009-01-01	0	45	223	215	166
4	2009-01-01	1	0	245	223	215
5	2009-01-01	1	15	182	245	223
6	2009-01-01	1	30	181	182	245

- The estimated model is

$$pickups = 5.3 + 0.97 * pickups\_lag1 + 0.005 * pickups\_lag2$$

Multivariate NYC model

- 45 degree plot of the new model:



- Much better! And the statistics are:

```
In [220]: print("Mean Absolute error (MAE): %.2f"% np
          # The mean squared error
          print("Root Mean squared error: %.2f"
                % np.sqrt(np.mean((regr.predict(x) -

          Mean Absolute error (MAE): 20.39
          Root Mean squared error: 26.85
```

- Much better too! Can you improve this further?

How to improve your model further: Ridge and LASSO

If the relationship between Y and X is approximately linear, then applying OLS will give you a good model, but notice that:

- if the dataset size n is much greater than the number of parameters p, i.e. $n >> p$ then your model coefficients (β) will have low variance. This is good!

How to improve your model further: Ridge and LASSO

If the relationship between Y and X is approximately linear, then applying OLS will give you a good model, but notice that:

- if the dataset size n is much greater than the number of parameters p, i.e. $n >> p$ then your model coefficients (β) will have low variance. This is good!
- if $n > p$, i.e. the number of datapoints is not much higher than p, your β will have very high variance. For example, if you remove/add a few datapoints, the coefficients will change a lot!

How to improve your model further: Ridge and LASSO

If the relationship between Y and X is approximately linear, then applying OLS will give you a good model, but notice that:

- if the dataset size n is much greater than the number of parameters p, i.e. $n >> p$ then your model coefficients (β) will have low variance. This is good!
- if $n > p$, i.e. the number of datapoints is not much higher than p, your β will have very high variance. For example, if you remove/add a few datapoints, the coefficients will change a lot!
- if $n < p$, you should not use OLS before applying dimensionality reduction techniques (you will learn later!)

Ridge regression

- Let us focus on the second situation: n is not much larger than p
- We want to use all possible data, but reduce variance of the model. i.e. make more “stable"

Ridge regression

- Let us focus on the second situation: n is not much larger than p
- We want to use all possible data, but reduce variance of the model. i.e. make more “stable"
- In Ridge regression, we add a penalty term to our objective function:

$$S = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2$$

Ridge regression

- Let us focus on the second situation: n is not much larger than p
- We want to use all possible data, but reduce variance of the model. i.e. make more “stable"
- In Ridge regression, we add a penalty term to our objective function:

$$S = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2$$

- The consequence is that higher values of β will be penalized...
- λ is a tuning parameter which constraints the effect of the “ridge". The higher it is, the smaller will β_j tend to be...

Ridge regression

- Let us focus on the second situation: n is not much larger than p
- We want to use all possible data, but reduce variance of the model. i.e. make more “stable"
- In Ridge regression, we add a penalty term to our objective function:

$$S = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2$$

- The consequence is that higher values of β will be penalized...
- λ is a tuning parameter which constraints the effect of the “ridge". The higher it is, the smaller will β_j tend to be...
- ...hence the model will become biased

Ridge regression

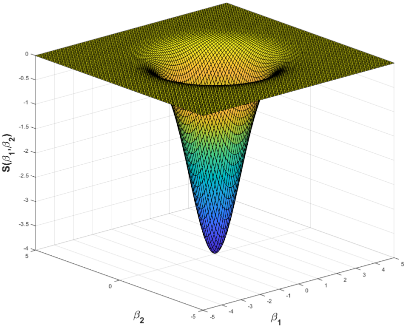
$$\text{Min} \;\; S = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$
$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 \leq t$$

Ridge regression

$$\text{Min} \;\; S = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 \leq t$$

- A visual intuition (with $p = 2$):

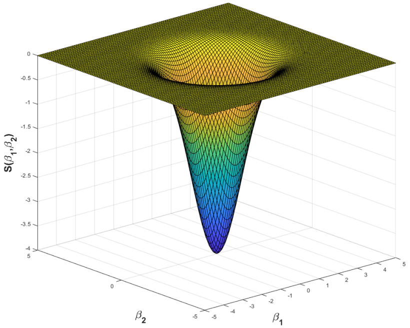


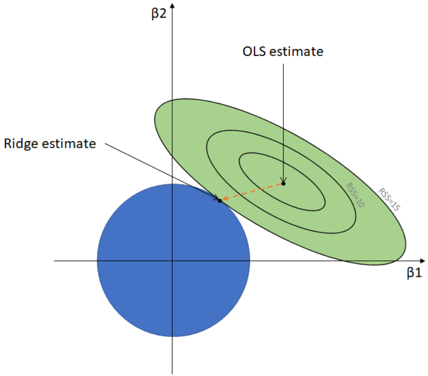
Ridge regression

$$\text{Min} \;\; S = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 \leq t$$

- A visual intuition (with $p = 2$):





Picture from https://towardsdatascience.com/ridge-regression-for-better-usage-2f19b3a202db

The Lasso

- One challenge of ridge regression is that the penalty term will never force any of the coefficients to be exactly zero, even if they are irrelevant
- An alternative is the *Lasso*
- Similar to ridge regression, but with a different penalty term that shrinks irrelevant coefficients exactly to zero.

The Lasso

- One challenge of ridge regression is that the penalty term will never force any of the coefficients to be exactly zero, even if they are irrelevant
- An alternative is the *Lasso*
- Similar to ridge regression, but with a different penalty term that shrinks irrelevant coefficients exactly to zero.

$$S = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} |\beta_j|$$

- Lasso uses an l_1 penalty instead of an l_2

The Lasso

- One challenge of ridge regression is that the penalty term will never force any of the coefficients to be exactly zero, even if they are irrelevant
- An alternative is the *Lasso*
- Similar to ridge regression, but with a different penalty term that shrinks irrelevant coefficients exactly to zero.

$$S = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} |\beta_j|$$

- Lasso uses an l_1 penalty instead of an l_2
- ...in practice, the lasso performs variable/feature selection!

The Lasso

$$\text{Min } \; S = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

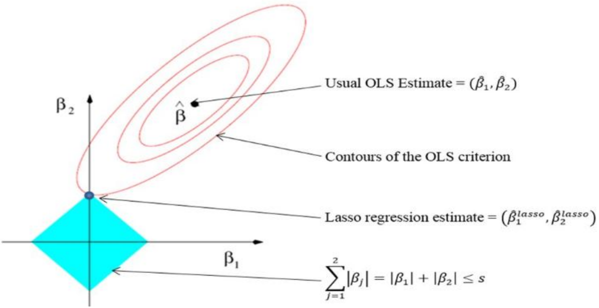
$$\text{subject to} \sum_{j=1}^{p} |\beta_j| \leq t$$

The Lasso

$$\text{Min} \;\; S = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

$$\text{subject to} \sum_{j=1}^{p} |\beta_j| \leq t$$

- A visual intuition (with $p = 2$):



Picture from https://rstatisticsblog.com/data-science-in-action/machine-learning/lasso-regression/

The Lasso vs Ridge regression

- lasso produces simpler and more interpretable models than ridge (and linear) regression!
- similar behavior to ridge regression: as λ grows, variance decreases and bias increases

The Lasso vs Ridge regression

- lasso produces simpler and more interpretable models than ridge (and linear) regression!
- similar behavior to ridge regression: as λ grows, variance decreases and bias increases
- Not very good at handling variables that show a correlation between them

The Lasso vs Ridge regression

- lasso produces simpler and more interpretable models than ridge (and linear) regression!
- similar behavior to ridge regression: as λ grows, variance decreases and bias increases
- Not very good at handling variables that show a correlation between them

Implementing Ridge regression and the Lasso

- You can try ridge regression using $sklearn.linear\_model.Ridge$
- You can try the Lasso using $sklearn.linear\_model.Lasso$
- Choose λ with cross-validation

Playtime!

- Part 4 of “4 - regression.ipynb" notebook.
- Expected duration: 30 min

APPENDIX

On maximum likelihood estimation

- **Maximum Likelihood (ML)** assumes knowledge of the distribution

$$p(y|\mathbf{x}) = D(\theta)$$

  where θ are the paremeters of the distribution

- For example, in linear regression, we have

$$p(y|\mathbf{x}) = \mathcal{N}(\boldsymbol{\beta}^T\mathbf{x}, \sigma^2)$$

On maximum likelihood estimation

- **Maximum Likelihood (ML)** assumes knowledge of the distribution of the target variable

$$p(y|\mathbf{x}) = D(\theta)$$

  where θ are the paremeters of the distribution

- The goal of ML estimation is to estimate the parameters $\hat{\theta}$

- The ML estimator $\hat{\theta}$ is the one that **maximizes the probability of the data given the parameters**.

The likelihood principle

- The "probability" of observing y given $\mathbf{x}$ is given by the conditional density

$$p(y|\mathbf{x}) = D(\theta)$$

- If the N observations are **independent and identically distributed (i.i.d.)**, the probability of $y_1, y_2, ..., y_N$ (given $\mathbf{x}$) is

$$p(y_1, y_2, ..., y_N|\mathbf{x}) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n)$$

- The likelihood function for the sample is defined as:

$$L(\theta) = p(y_1, y_2, ..., y_N|\mathbf{x}) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n) = \prod_{n=1}^{N} L_n(\theta)$$

- where $L_n(\theta)$ is called the likelihood contribution for observation n.
- The **Maximum Likelihood Estimator** is the value $\hat{\theta}$ that maximizes $L(\theta)$

ML estimator

- Typically easier to estimate **log-likelihood**:

$$logL(\theta) = \sum_{n=1}^{N} logL_n(\theta)$$

- Typical calculus function maximization process
  1. differentiate in order of θ
  2. set to zero
  3. solve equation

Example with Linear Regression

- Model is:

$$y_n = \boldsymbol{\beta}^T \mathbf{x}_n + \epsilon_n, \text{ with } n = 1, 2, ..., N$$

- The "probability" of observing y_n given the model is:

$$p(y_n|\mathbf{x}_n) = \mathcal{N}(\boldsymbol{\beta}^T \mathbf{x}_n, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(y_n - \boldsymbol{\beta}^T \mathbf{x}_n)^2}{\sigma^2}}$$

- Since the observations are i.i.d., we have

$$L(\theta) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n) = \prod_{n=1}^{N} \mathcal{N}(\boldsymbol{\beta}^T \mathbf{x}_n, \sigma^2) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N \prod_{n=1}^{N} e^{-\frac{1}{2}\frac{(y_n - \boldsymbol{\beta}^T \mathbf{x}_n)^2}{\sigma^2}}$$

Example with Linear Regression

- The log becomes:

$$logL(\theta) = -\frac{N}{2}log(2\pi\sigma^2) - \frac{1}{2}\sum_{n=1}^{N}\frac{(y_n - \boldsymbol{\beta}^T\mathbf{x}_n)^2}{\sigma^2}$$

- We differentiate in order of β and σ

$$\frac{\partial logL(\theta)}{\beta} = \sum_{n=1}^{N}\frac{\mathbf{x}_n(y_n - \boldsymbol{\beta}^T\mathbf{x}_n)}{\sigma^2}$$

$$\frac{\partial logL(\theta)}{\sigma} = -\frac{T}{2\sigma^2} + \frac{1}{2}\sum_{n=1}^{N}\frac{(y_n - \boldsymbol{\beta}^T\mathbf{x}_n)^2}{\sigma^4}$$

Example with Linear Regression

- Now we equate to 0, and (after a bit of algebra) obtain the **same** solution for β as in OLS:

$$\hat{\boldsymbol{\beta}} = \left(\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^T \right)^{-1} \sum_{n=1}^{N} \mathbf{x}_n y_n = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

- And approximately the same for σ, letting $\hat{\epsilon_n} = (y_n - \hat{\beta}^T \mathbf{x}_n)$:

$$\frac{1}{2} \sum_{n=1}^{N} \frac{\hat{\epsilon_n}^2}{\sigma^4} = \frac{N}{2\sigma^2} \implies \hat{\sigma} = \sqrt{\frac{1}{N} \sum_{n=1}^{N} \hat{\epsilon_n}^2}$$

Derivation of OLS Estimates

- Residual for the jth observation for an estimate $\boldsymbol{\beta}$:

$$y_j - \beta_0 - \beta_1 x_{j1} - \cdots - \beta_r x_{jr}$$

- OLS minimizes the **residual sum of squares**

$$\begin{aligned} S &\equiv \sum_{j=1}^{n} \left(y_j - \beta_0 - \beta_1 x_{j1} - \cdots - \beta_r x_{jr} \right)^2 \\ &= \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta} \right)^T \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta} \right) \end{aligned}$$

Derivation of OLS Estimates (cont.)

- Rewrite S

$$
\begin{aligned}
S &= (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) = \left(\mathbf{y}^T - \boldsymbol{\beta}^T\mathbf{X}^T\right)(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \\
&= \mathbf{y}^T\mathbf{y} - \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} \\
&= \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}
\end{aligned}
$$

- First order condition $\frac{\partial S}{\partial \boldsymbol{\beta}} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = 0$

$$\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^T\mathbf{y}$$

$$\boxed{\underset{((r+1)\times 1)}{\hat{\boldsymbol{\beta}}} = \left(\underset{((r+1)\times n)}{\mathbf{X}^T}\underset{(n\times(r+1))}{\mathbf{X}}\right)^{-1}\underset{((r+1)\times n)}{\mathbf{X}^T}\underset{(n\times 1)}{\mathbf{y}}}$$