Unit 1
Simple Linear Regression

Prof. Phil Schniter

THE OHIO STATE UNIVERSITY

ECE 5307: Introduction to Machine Learning, Sp23

Learning objectives

- Understand how to load datasets in Python using pandas
- Visualize data using a scatter plot
- Understand sample mean, variance, and covariance
- Describe a linear model for data
  - Identify the target variable and predictor
- Compute the least-squares fit using the regression formula
- Compute the R^2 measure-of-fit
- Visually assess goodness-of-fit and identify causes of poor fit

Outline

Example: Predicting automobile mpg

- We will consider the task of predicting the fuel efficiency (in mpg) of a car from features like # cylinders, horsepower, weight, etc.
- Demo in Jupyter notebook: `demo01_auto_mpg.ipynb`
  - Learn from demo, and test your understanding in the lab
- Uses data from UCI library: `https://archive.ics.uci.edu/ml/`
- Data is loaded using Python's pandas library
  - Pandas routines have many options!
  - Learn from examples; Google is your friend!
  - Pandas `read_csv` command creates a dataframe object with 3 main components:
    - `df.values`: numerical values in Numpy format
    - `df.columns`: column labels
    - `df.index`: row labels

Result of pandas' read_csv

```
import pandas as pd
import numpy as np
```

```
In [3]: names = ['mpg', 'cylinders','displacement', 'horsepower',
                 'weight', 'acceleration', 'model year', 'origin', 'car name']
```

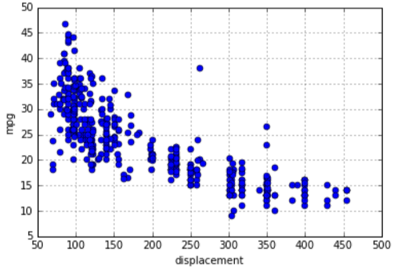
```
In [4]: df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/'+
                          'auto-mpg/auto-mpg.data',
                          header=None,delim_whitespace=True,names=names,na_values='?')
        df.head(6)
```

Out[4]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130.0	3504.0	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693.0	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436.0	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433.0	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449.0	10.5	70	1	ford torino
5	15.0	8	429.0	198.0	4341.0	10.0	70	1	ford galaxie 500

Visualizing the data

- When possible, visualize the data before working with it.
- Python has MATLAB-like plotting features in the matplotlib module.



Visualizing the Data

We load the `matplotlib` module to plot th
MATLAB.

In [15]:
```
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

First, let's extract some data columns and co

In [16]:
```
xstr = 'displacement'
x = np.array(df[xstr])
y = np.array(df['mpg'])
```

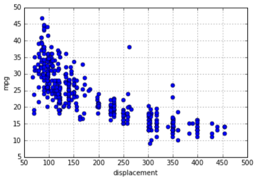
Now we can create a scatter plot:

In [17]:
```
plt.plot(x,y,'o')
plt.xlabel(xstr)
plt.ylabel('mpg')
plt.grid(True)
```

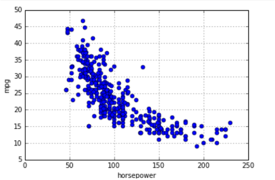
Postulating a model

- What relationships do you see?
- Is there a mathematical model relating the variables?
- How well can you predict mpg from these variables?

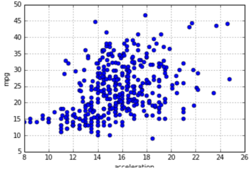
mpg



displacement



horsepower

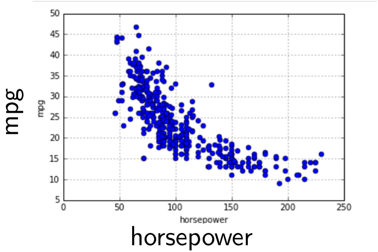


acceleration

Outline

Describing and visualizing data

- y: the variable we are trying to predict
  - Many names: target, label, regressand, response variable, dependent variable
- x: the variable we are using for prediction
  - Many names: feature, predictor, regressor, attribute, independent variable
  - For now we consider a single variable. Next unit we'll consider multiple
- Data: the set of pairs $\{(x_i, y_i)\}_{i=1}^n$
  - Each pair is called a sample
  - We will use n for the number of samples
- A scatter plot is used to visualize the data pairs

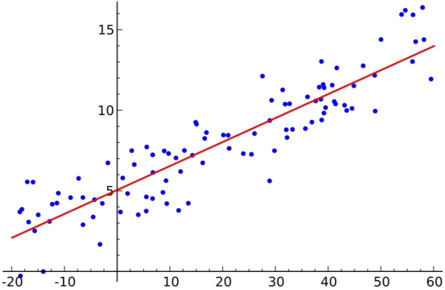


Single-variable linear model

- Assume a linear relationship:

$$y \approx \qquad\qquad \triangleq \widehat{y}$$

  - β_1: slope
  - β_0: intercept
  - $\widehat{y}$: prediction
- $\boldsymbol{\beta} = [\beta_0, \beta_1]^\mathsf{T}$ are the parameters of the model. Also called coefficients or weights
- Why this model?
  - easy to interpret & analyze
  - simple to optimize parameters (as we will see)
- When is this a good model?



The regression line $\widehat{y}(x)$ is shown in red

Outline

Linear model: The residual

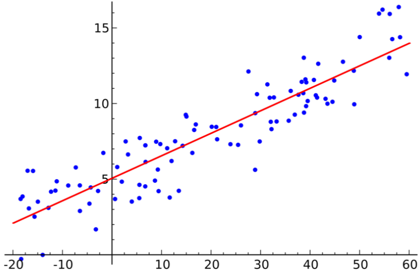
- Note: x does not *exactly* predict y:

$$y \approx$$

- Let's model this behavior explicitly using a residual, ϵ:

$$y =$$

- For the ith sample, we then have:
  - predicted value: $\widehat{y}_i = \beta_0 + \beta_1 x_i$
  - residual: $\epsilon_i = y_i - \widehat{y}_i$



The residual ϵ_i is the vertical deviation from y_i to the regression line

Least-squares fit

- How do we choose the model parameters $\boldsymbol{\beta} = [\beta_0, \beta_1]^\mathsf{T}$?
- Minimize the residual sum of squares (RSS):

$$\mathrm{RSS}(\beta_0, \beta_1) \triangleq$$

https://en.wikipedia.org/wiki/Residual_sum_of_squares

  - Also called the sum of squared errors (SSE) and sum of square residuals (SSR)
  - Note that ϵ_i and $\widehat{y}_i$ are implicitly functions of $[\beta_0, \beta_1]$
- The value of $\boldsymbol{\beta}$ that minimizes RSS is the least-squares fit.
  - Geometrically: minimizes sum of squared distances to the regression line.

The optimization approach to ML: A general recipe

General ML problem		Simple Linear Regression
Assume a model with some parameters	$\rightarrow$	Linear model: $\widehat{y} = \beta_0 + \beta_1 x$
Get training data	$\rightarrow$	Data: $\{(x_i, y_i)\}_{i=1}^n$
Choose a loss function	$\rightarrow$	$\mathrm{RSS}(\beta_0, \beta_1) \triangleq \sum_{i=1}^n (y_i - \widehat{y}_i)^2$
Find parameters that minimize loss	$\rightarrow$	Find (β_0, β_1) that minimizes $\mathrm{RSS}(\beta_0, \beta_1)$

Outline

- Motivating Example: Predicting Automobile MPG
- Linear Model
- Least-Squares Fit: Problem Definition
- Sample Mean, Variance, Standard Deviation, Covariance
- Least-Squares Fit: The Solution
- Assessing Goodness-of-Fit

Sample mean, variance, & standard deviation

If we summarize the data in the right way, we can easily design the optimal β!

- Sample mean:

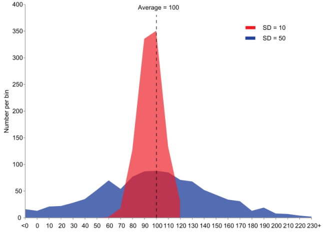
$$\overline{x} \triangleq \qquad \overline{y} \triangleq$$

- Sample variance:

$$s_x^2 \triangleq \qquad s_y^2 \triangleq$$

  - Note: some authors use $\frac{1}{n-1}$ to give an "unbiased estimate." We'll explain this later, in Unit 3.

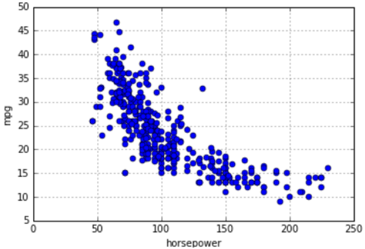
- Sample standard deviation (SD):
  - Simply the square-root of the sample variance



https://en.wikipedia.org/wiki/Variance

Visualizing sample mean & SD on scatter plot

- Sample means $\overline{x}$ and $\overline{y}$:
  - The center of mass in each axis
- Standard deviations s_x and s_y:
  - The "spread" in each axis about the mean
  - If the data was Gaussian distributed...
    - 68% of points < 1 SD from mean
    - 95% of points < 2 SDs from mean
    - 99.7% of points < 3 SDs from mean



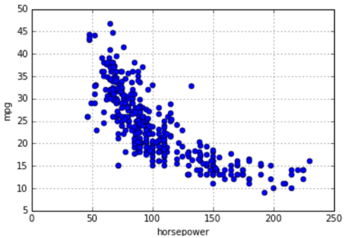
- What are your estimates of $\overline{x}, \overline{y}, s_x, s_y$ from the above scatter plot?

Computing the sample mean & SD in Python

- We can exactly compute $\overline{x}, \overline{y}, s_x, s_y$ using the Numpy package in Python

```
In [27]:  xm = np.mean(x)
           ym = np.mean(y)
           sxx = np.mean((x-xm)**2)
           syy = np.mean((y-ym)**2)
```

```
xm        = 104.47,          ym=   23.45
sqrt(sxx)=  38.44,  sqrt(syy)=    7.80
```



Sample covariance

- The sample covariance is

$$s_{xy} \triangleq$$

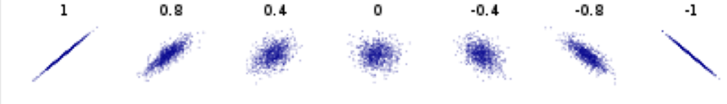
which indicates how "related" $\{x_i\}$ and $\{y_i\}$ are.

- The value s_{xy} is easier to interpret after normalization. This motivates the sample (Pearson) correlation coefficient:

$$\rho_{xy} \triangleq$$

  - The property $\rho_{xy} \in [-1, 1]$ is a consequence of the Cauchy-Schwarz inequality.

- Example scatterplots for datasets with various ρ_{xy}:



https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

Alternative expressions for sample variance & covariance

- Recall that the sample variance was defined as $s_x^2 \triangleq \frac{1}{n}\sum_{i=1}^n (x_i - \overline{x})^2$
- A very useful alternative formula can be found by expanding the square:

$$s_x^2 = \frac{1}{n}\sum_{i=1}^n x_i^2 - 2\overline{x}\frac{1}{n}\sum_{i=1}^n x_i + \overline{x}^2 =$$

- Similarly, for the sample covariance we had $s_{xy} \triangleq \frac{1}{n}\sum_{i=1}^n (x_i - \overline{x})(y_i - \overline{y})$
- A useful alternative expression for that is

$$s_{xy} =$$

Notation

- We will use the following notation in this class (we'll try to be consistent)
- Note: some books/authors use different notations

Statistic	Notation	Formula	Python
sample mean	$\overline{x}$	$\frac{1}{n}\sum_{i=1}^{n} x_i$	`xm`
sample variance	$s_x^2 = s_{xx}$	$\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2$	`sxx`
sample standard deviation	$s_x = \sqrt{s_{xx}}$	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2}$	`sx`
sample covariance	s_{xy}	$\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})$	`sxy`
sample correlation coefficient	ρ_{xy}	$\dfrac{s_{xy}}{s_x s_y}$	`rhoxy`

Outline

Minimizing RSS

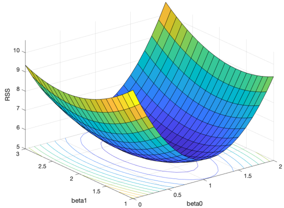
- To minimize $\mathrm{RSS}(\beta_0, \beta_1)$, we find the β_0 and β_1 that zero the gradient, i.e., the vector of partial derivatives:

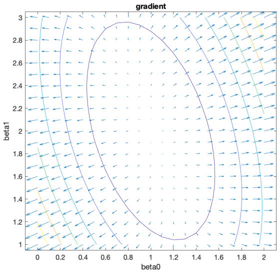
$$\left[\frac{\partial\,\mathrm{RSS}(\beta_0,\beta_1)}{\partial\beta_0}, \frac{\partial\,\mathrm{RSS}(\beta_0,\beta_1)}{\partial\beta_1}\right]^{\mathsf{T}} = \begin{bmatrix}0\\0\end{bmatrix}$$

  Because RSS is quadratic with a PSD Hessian, the zero-gradient point is guaranteed to be an RSS minimum (more on this later)

- After some manipulations (see next page), we obtain the optimal values

$$\beta_1 = \qquad\qquad \beta_0 =$$





Minimizing RSS: Derivation

The minimum RSS is achieved by values of (β_0, β_1) that zero the gradient, i.e.,

$$0 = \frac{\partial \, \mathrm{RSS}(\beta_0, \beta_1)}{\partial \beta_0} = \tag{1}$$

$$0 = \frac{\partial \, \mathrm{RSS}(\beta_0, \beta_1)}{\partial \beta_1} = \tag{2}$$

Starting with (1), we can multiply both sides by $-\frac{1}{2n}$ to give

$$0 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i) = \underbrace{\frac{1}{n}\sum_{i=1}^{n} y_i}_{\overline{y}} - \beta_0 - \beta_1 \underbrace{\frac{1}{n}\sum_{i=1}^{n} x_i}_{\overline{x}} \quad \Leftrightarrow \tag{3}$$

Doing the same with (2) gives

$$0 = \frac{1}{n}\sum_{i=1}^{n} x_i(y_i - \beta_0 - \beta_1 x_i) = \frac{1}{n}\sum_{i=1}^{n} x_i y_i - \overline{x}\beta_0 - \beta_1 \frac{1}{n}\sum_{i=1}^{n} x_i^2 \tag{4}$$

Minimizing RSS: Derivation (continued)

Plugging in (3) into (4) gives

$$0 = \underbrace{\frac{1}{n}\sum_{i=1}^{n} x_i y_i - \overline{x}\,\overline{y}} - \beta_1 \left(\underbrace{\frac{1}{n}\sum_{i=1}^{n} x_i^2 - \overline{x}^2} \right) \quad \Leftrightarrow$$

To find the minimum RSS, we plug the optimal value of β_0 into the RSS definition to get

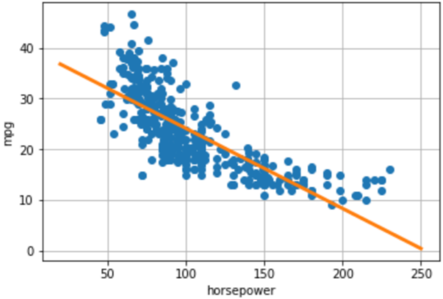
$$\begin{aligned} \mathrm{RSS}(\beta_0, \beta_1) &= \sum_i \Big(y_i - \beta_0 - \beta_1 x_i \Big)^2 = \sum_i \Big((y_i - \overline{y}) - \beta_1 (x_i - \overline{x}) \Big)^2 \\ &= \underbrace{\sum_i (y_i - \overline{y})^2} - 2\beta_1 \underbrace{\sum_i (y_i - \overline{y})(x_i - \overline{x})} + \beta_1^2 \underbrace{\sum_i (x_i - \overline{x})^2} \end{aligned} \qquad (5)$$

and then we plug the optimal value of β_1 into (5) to get

$$\begin{aligned} \mathrm{RSS}(\beta_0, \beta_1) &= n\left(s_{yy} - 2\frac{s_{xy}^2}{s_{xx}} + \frac{s_{xy}^2}{s_{xx}} \right) = n\left(s_{yy} - \frac{s_{xy}^2}{s_{xx}} \right) \\ &= n\left(1 - \frac{s_{xy}^2}{s_{xx} s_{yy}} \right) s_{yy} = \end{aligned}$$

Automobile demo

- Applied to our Python demo...




```
xm = np.mean(x)
ym = np.mean(y)
sxx = np.mean((x-xm)**2)
syy = np.mean((y-ym)**2)
syx = np.mean((y-ym)*(x-xm))
beta1 = syx/sxx
beta0 = ym - beta1*xm
Rsq = syx**2/sxx/syy
```

`beta0=39.94, beta1=-0.16`

`Rsq=0.61`

$$\widehat{\text{mpg}} = \beta_0 + \beta_1 \times \text{horsepower}$$

- Another good simple-linear-regression demo can be found at
  https://stattrek.com/regression/regression-example.aspx?Tutorial=AP

Outline

R^2 Goodness-of-fit

- Question: How to judge whether a predictor is doing "well" on a dataset?
- Answer: Use a *normalized* version of RSS:
  - RSS includes contributions from n training samples.
    By considering RSS/n, we remove the dependence on n.
  - RSS/n depends on s_y^2, the variance-of-y (i.e., if s_y^2 doubles then RSS/n doubles).
    By considering $\frac{\mathrm{RSS}/n}{s_y^2}$, we remove the dependence on s_y^2.
- More commonly, we report

known as the "coefficient of determination"

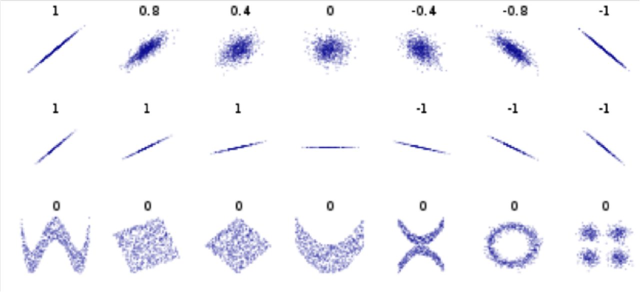
https://en.wikipedia.org/wiki/Coefficient_of_determination

  - $R^2 = 1$ implies that the predictor is perfect (i.e., $\widehat{y}_i = y_i$)
  - $R^2 = 0$ implies the predictor is no better than the trivial one (i.e., $\widehat{y}_i = \overline{y}$)
  - $R^2 < 0$ implies worse than trivial!
- With RSS-minimizing β_0 and β_1, we know

$$\mathrm{RSS}(\beta_0, \beta_1) = n\big(1 - \rho_{xy}^2\big) s_y^2 \quad \Rightarrow$$

Visualizing ρ_{xy}^2 from data $\{(x_i, y_i)\}_{i=1}^n$

- If $\rho_{xy}^2 \approx$, then LS linear model gives a very good fit
- If $\rho_{xy}^2 \approx$, then LS linear model gives a very poor fit
- Since LS coef $\beta_1 = \frac{\rho_{xy} s_y}{s_x}$, we have that $\text{sgn}(\beta_1) =$



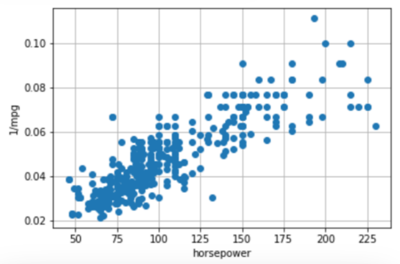
$\leftarrow$ data with varying ρ_{xy} (and fixed $s_x = s_y$)

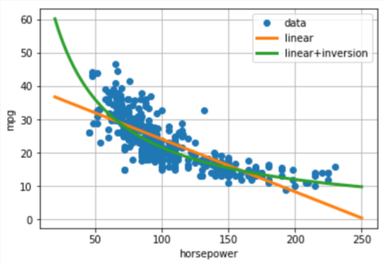
$\leftarrow$ data with $\rho_{xy} \in \{-1, 0, 1\}$ (but varying s_y)

$\leftarrow$ data with $\rho_{xy} = 0$: linear prediction doesn't work

A better model for the automobile example?

- What if we predicted the inverse:
  $\frac{1}{\text{mpg}} \approx \beta_0 + \beta_1 \times \text{horsepower}$
- This uses a *nonlinear data transformation* followed by linear regression
- Will explore this idea later in the course





Learning objectives

- Understand how to load datasets in Python using pandas
- Visualize data using a scatter plot
- Understand sample mean, variance, and covariance
- Describe a linear model for data
  - Identify the target variable and predictor
- Compute the least-squares fit using the regression formula
- Compute the R^2 measure-of-fit
- Visually assess goodness-of-fit and identify causes of poor fit