CPE 490 590: Machine Learning for Engineering Applications

04 Linear Regression

Rahul Bhadani
Electrical & Computer Engineering, The University of Alabama in Huntsville

Outline

Logistics

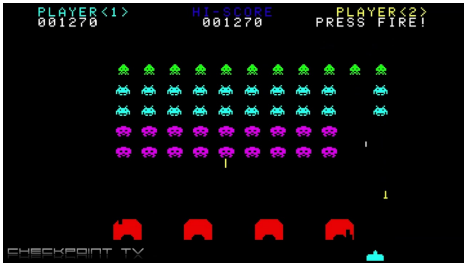
Announcement

- Quiz 1 Deadline: Jan 26, 2025, 11:59 PM

Machine Learning

What is Machine Learning?

Give a computer ability to learn without explicit programming using data.



Supervised vs Unsupervised Learning

Supervised Learning

- Input ($\mathbf{x}$) and output label y data available.
- We learn a mapping $f : \mathbf{x} \rightarrow y$.
- Give a learnt mapping f and a new $\mathbf{x}$, a computer can guess y.

Unsupervised Learning

- Only Input ($\mathbf{x}$) data available.
- Uncover patterns.
- Learn to distinguish $\mathbf{x}_1$ from $\mathbf{x}_2$.

Supervised Learning

What is Supervised Learning

Learn from examples

Supervised Learning from Examples

Input x		Output y	Application
Email	$\rightarrow$	Spam (? (0/1))	Spam Filtering
Audio	$\rightarrow$	Text Transcripts	Speech Recognition
English	$\rightarrow$	Bengali	Machine Translation
Ad, User Infor	$\rightarrow$	Clicked (0/1)	Online Advertising
Image, Radar Info	$\rightarrow$	Position of other Cards	Self-driving Cars
Image of Phone	$\rightarrow$	Defect (0/1)	Automated Quality Inspection

The Supervised Learning Setting

- Consider **Input**: $\mathcal{D} := \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ where $\mathbf{x}_i \in X$ is the feature vector (called **explanatory/predictor variable**), and $y_i \in Y$ is the known true label (called **response variable**).
- Supervised learning algorithms learn from **"right answers"**

Two Types of Supervised Learning

1. Regression

Input: continuous/categorical data points (or a vector of data points)

Output: continuous data points (floating point real numbers)

Example: Input: {road curvature, traffic density, precipitation}
Output: Driving speed to the car.

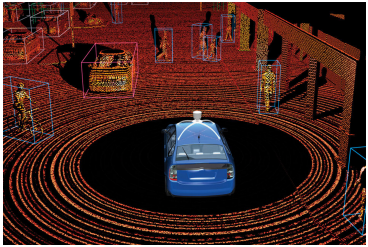
Two Types of Supervised Learning

2. Classification

Input: continuous/categorical data points (or a vector of data points)

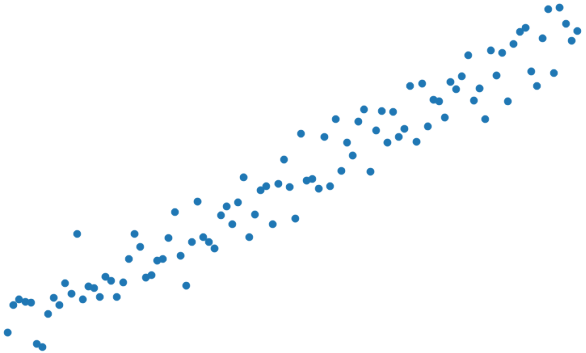
Output: categorical data points (also known as **class** or **category**) (yes/no, red/blue/green, defective/intact/partially defective)

Example: Input: {current speed of the car, LiDAR data}
Output: Moving Obstacle/Static Obstacle



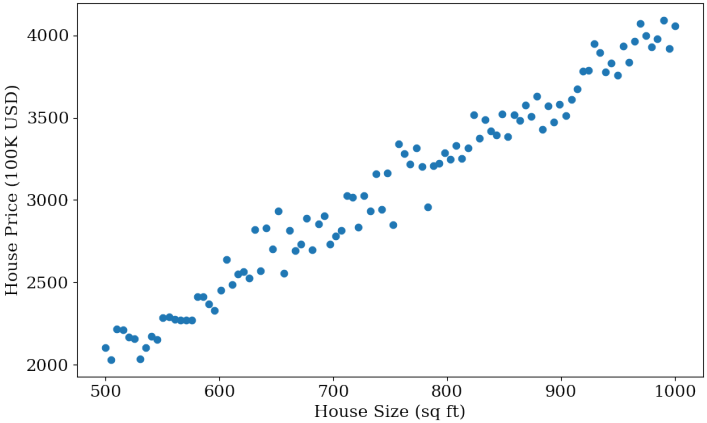
Linear Regression

Setting up the Linear Regression Problem



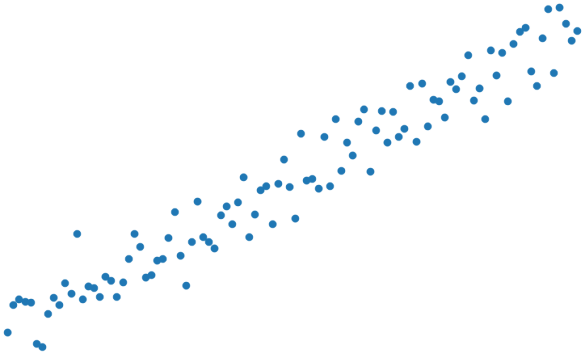
What kind of relationship we see here?

Setting up the Linear Regression Problem



Can we guess the price of house for a size of 650 sq ft?

Setting up the Linear Regression Problem



Can we only fit a straight line?

Setting up the Linear Regression Problem

Definition

Linear regression is a kind of supervised machine learning algorithm where a linear discriminant model $g(\mathbf{x})$ (a straight line) is used to fit on the given dataset.

We define a linear model as $g(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\mathbf{x} + w_0$.

Simple Linear Regression/SLR (One Predictor Variable)

For simplicity, we first consider a one-dimensional feature vector and one-dimensional label.

SLR in Python Using Scikit-Learn

Reading Data

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.model_selection import train_test_split, cross_val_score
from statistics import mean

# Read data
data = pd.read_csv("../Data/HousePrices/kc_house_data.csv")
data.dropna(inplace=True) #remove drop na (not a number)
data.head()
```

SLR in Python Using Scikit-Learn

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...

5 rows × 21 columns

SLR in Python Using Scikit-Learn

Get the independent (explanatory or predictor) and dependent (response) variable

```
y = data['price']
X = data[['sqft_living']]
```

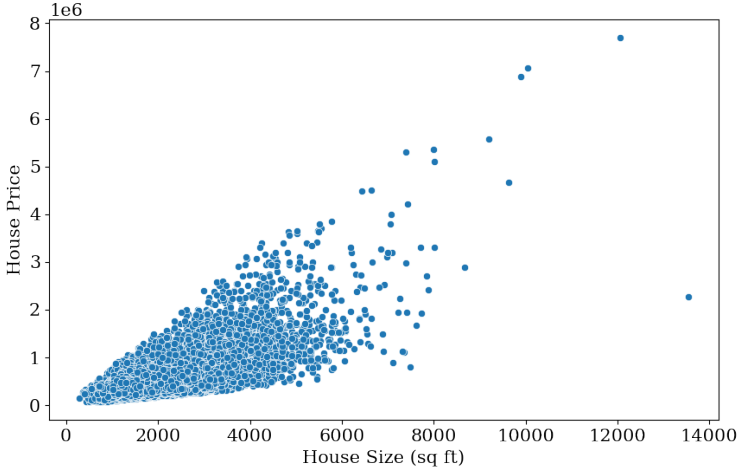
Ensure X is a 2D array because scikit-learn Linear Regressione expects 2D Array (n_samples, n_features)). Even if there is only one feature, it still needs to be shaped as a 2D array with one column.

SLR in Python Using Scikit-Learn

Plot between independent and dependent variable

```
import seaborn as sns
fig, ax = plt.subplots(figsize=(10, 6))
# We will use seaborn for plotting
sns.scatterplot(x='sqft_living', y='price', data=data, ax=ax)
fig.patch.set_alpha(0.0)
plt.xlabel('House Size (sq ft)')
plt.ylabel('House Price')
plt.show()
```

SLR in Python Using Scikit-Learn



SLR in Python Using Scikit-Learn

Dividing the data into training and testing set

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
```

We divide the dataset into training and test data. Test data contains 25% of the overall data points. Model is being trained using training dataset while model is evaluated using test dataset.

SLR in Python Using Scikit-Learn

Building the Linear Regression Model using One Feature Only

```
linearModel = LinearRegression()
linearModel.fit(X_train, y_train)

# Evaluating the Linear Regression model on the test dataset
# coefficient of determination of the prediction
print(linearModel.score(X_test, y_test))
```

For this training the evaluation score was 0.46 which is not very good, because relationship between dependent and independent variables are not strongly linear as we saw earlier in the scatter plot.

SLR in Python Using Scikit-Learn

Make a Prediction on trained model using a new data point

```
new_data_point = pd.DataFrame({'sqft_living': [5000]})

# Make the prediction
predicted_price = linearModel.predict(new_data_point)

print(f"Predicted house price with 5000 sqft living area: ${predicted_price[0]:,.2f}")
```

Output: Predicted house price with 5000 sqft living area: $1,372,166.50

Linear Model for SLR

$$g(x) = w_0 + w_1 x \tag{1}$$

We will denote $g(x)$ as $\hat{y}$ from now onwards as it is **predicted** value. w_1 is the slope, and w_0 is the y-intercept.

Data Model and Assumptions for SLR

As input data is $\mathcal{D} := \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, we imagine that the actual equation that models the physical process may be

$$y_i = (w_0 + w_1 x_i) + \epsilon_i \tag{2}$$

which in essence says that $Y = \textit{signal} + \textit{noise}$. ϵ_i is the random error or noise term.

For simplicity, we assume that the noise is zero-mean, constant variance noise.

- $\mathbb{E}[\epsilon_i] = 0 \; \forall \; i$.
- $\sigma^2[\epsilon_i] = \sigma^2$ (a constant) $\forall \; i$.
- $\sigma[\epsilon_i, \epsilon_j] = 0 \; \forall \; i \neq j$.

Validity of Model and Our Assumption

We see that

$$\begin{aligned}\mathbb{E}[y_i] &= \mathbb{E}[w_0 + w_1 x_i + \epsilon_i] = \mathbb{E}[w_0 + w_1 x_i] + \mathbb{E}[\epsilon_i] \\ &= w_0 + w_1 x_i + \mathbb{E}[\epsilon_i] \qquad\qquad = w_0 + w_1 x_i\end{aligned} \tag{3}$$

This was possible as $w_0 + w_1 x_i$ is nonrandom, and we consider zero mean Gaussian error.
$\mathbb{E}[y_i] = w_0 + w_1 x_i$ called mean response.
Also,

$$\sigma^2[y_i] = \sigma^2[w_0 + w_1 x_i + \epsilon_i] = \sigma^2[\epsilon_i] = \sigma^2 \tag{4}$$

as $w_0 + w_1 x_i$ is nonrandom.
We will still see $\sigma[y_i, y_j] = 0$ for all $i \neq j$.

Regression and Least Square

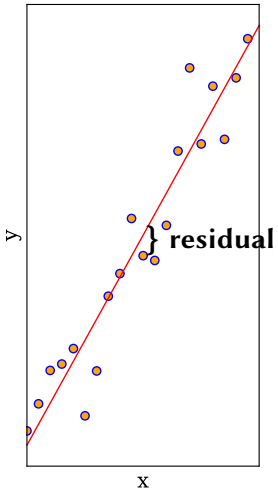
1. The problem of regression becomes finding w_0 and w_1.
2. We use **Least Square** method to estimate w_0, and w_1. We will denote estimated $\hat{w}_0$, and $\hat{w}_1$.
3. The fitted value would be $\hat{y}_i = \hat{w}_0 + \hat{w}_1 x_i$.

We want to minimize

$$Q = \sum (y_i - \hat{y}_i)^2.$$

4. Residuals are $e_i = y_i - \hat{y}_i$. We want residuals to be as low as possible.

Regression Line and Residuals



Here, points are data points, and the line is $\hat{w}_0 + \hat{w}_1 x_i$.

Solving for w_0 and w_1: Normal Equations

We find that the least square estimators solve the following system of equations:

$$\sum y_i = n\hat{w}_0 + \hat{w}_1 \sum x_i$$
$$\sum x_i y_i = \hat{w}_0 \sum x_i + \hat{w}_1 \sum x_i^2$$

They are called **normal equations** for the least-square estimators.

Gauss-Markov Theorem

The Gauss-Markov theorem says that, under certain conditions, the ordinary least squares (OLS) estimator of the coefficients of a linear regression model is the best linear unbiased estimator.

The Gauss-Markov theorem tells us that using normal equations, we can get the least square solutions for $\hat{w}_0$ and $\hat{w}_1$ such they are unbiased and have minimum variance among all unbiased linear estimators, i.e.

$$\mathbb{E}[\hat{w}_j] = w_j, \quad j = 0, 1 \tag{5}$$

and $\sigma^2[\hat{w}_0]$ and $\sigma^2[\hat{w}_1]$ are minimized.

The Least-Square Solution

The solution comes out to be the closed form:

$$\hat{w}_1 = \frac{\sum_i (x_i - \overline{x})(y_i - \overline{y})}{\sum_i (x_i - \overline{x})^2}$$

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

Example: Linear Regression on Toluca Dataset

We will consider the Toluca dataset: https://github.com/rahulbhadani/CPE490_590_Sp2025/blob/master/Data/Toluca/toluca.txt

We have

- x = lot size of refrigerator parts production.
- y = hours worked (labor)

at the Toluca manufacturing company.

We have a dataset, what do we do first?

Example: Linear Regression on Toluca Dataset

We will consider the Toluca dataset: https://github.com/rahulbhadani/CPE490_590_Sp2025/blob/master/Data/Toluca/toluca.txt

We have

- x = lot size of refrigerator parts production.
- y = hours worked (labor)

at the Toluca manufacturing company.

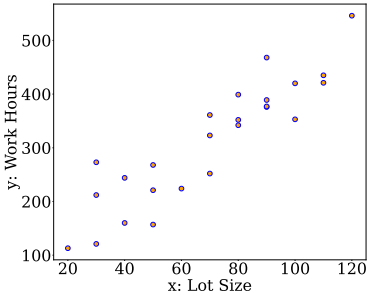
We have a dataset, what do we do first?

We plot the data first!

Example: Linear Regression on Toluca Dataset

Plot the data first

```
toluca = pd.read_csv('../Data/toluca.txt', sep='\t')
x = toluca['LotSize'].to_numpy()
y = toluca['WorkHours'].to_numpy()
plt.rcParams['font.family'] = 'Serif'
plt.rcParams['font.size'] = 15
fig, ax = plt.subplots(figsize=(17, 14))
plt.scatter(x, y, c='orange', edgecolors='blue', s=200)
plt.xlim([0, 123])
plt.xlabel('x: Lot Size')
plt.ylabel('y: Work Hours')
```

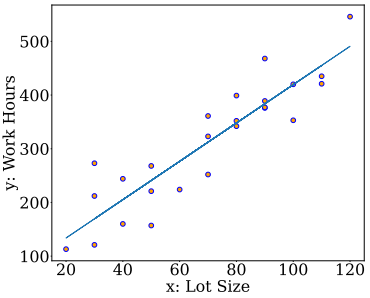


We see a linear relationship (sort of!).

Example: Linear Regression on Toluca Dataset

Fitting the data

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(x.reshape((-1, 1)), y)
print(reg.coef_) # w_1
print(reg.intercept_) #w_0
fig, ax = plt.subplots(figsize=(17, 14))
plt.scatter(x, y, c='orange', edgecolors='blue', s=200)
plt.xlim([0, 123])
plt.xlabel('x: Lot Size')
plt.ylabel('y: Work Hours')
yhat = reg.coef_*x + reg.intercept_
plt.plot(x, yhat)
```



Prediction on the Fitted Model

```
reg.predict([[50.0], [100.0]])
```

Output: array([240.8759596 , 419.38606061])

Consequences of the Least-Square Fit

We see the following mathematical consequences as a result of the least square fit:

1. $\sum e_i = 0$.
2. Sum of squared residuals (errors) $\sum e_i^2$ is minimum.
3. $\sum y_i = \sum \hat{y}_i$.
4. $\sum e_i x_i = 0$ (weighted sum of e_i's is zero).
5. $\sum e_i \hat{y}_i = 0$.
6. $\hat{y}(\overline{x}) = \overline{y}$.

Estimating Variance

We want to know how much uncertainty is there in our estimate.

Mean-Squared Error (MSE)

$$\text{MSE} = \frac{\text{SSE}}{\text{degree of freedom (df)}} = \frac{\sum(y_i - \hat{y})^2}{n-2}. \tag{6}$$

Here the degree of freedom is calculated as **df = no. of observations - no. of fitted components**.
Mathematically, we can show that $\mathbb{E}[\text{MSE}] = \sigma^2$.

Goodness of Fit using Coefficient of Determination R^2

Goodness of Fit, i.e. how well this linear regression model fits into the given dataset is given by

$$R^2 = 1 - \frac{\text{SSE}}{\text{SSTO}}$$

where $\text{SSTO} = \sum(y_i - \overline{y})^2$.
R^2 is the percentage of the total variation of the y_i explained by the regression model.

Properties of R^2

Some interesting properties of R^2:

1. $R^2 = 1$ when every point lies on the regression straight line.
2. $R^2 = 0$ when data are cloudy.

Some limitations of R^2:

1. $R^2 \to 1$ indicates a strong linear relationship but it might be a poor fit.
2. $R^2 \to 0$ indicates a weak linear relationship but it may be a good **nonlinear fit**.

Goodness of Fit on Toluca Dataset

```
# Calculate and print R^2
r_squared = reg.score(x.reshape((-1, 1)), y)
print("R^2 (Coefficient of Determination):", r_squared)
```

Let's pivot a little bit to discuss some concepts needed to build a foundation for additional concepts for linear regression.

Sample Statistics

Consider data points $y_1, y_2, \cdots, y_n$ represent a sample. Then,

1. Sample mean: $\overline{y} = \dfrac{\sum y_i}{n}$
2. Sample variance: $s^2 = \dfrac{\sum (y_i - \overline{y})^2}{n-1}$
3. Sample standard deviation: $\sqrt{s^2} = s$.
4. Degree of freedom: Number of independent elements required to determine some value. (See more: Chapter 2: Design of Experiments, Douglas Montogmery, Page 30, 9th Edition)
5. For a given normal distribution $Y \sim \mathcal{N}(\mu, \sigma^2)$, then we can create a standard normal random variable by writing

$$Z = \frac{Y - \mu}{\sigma}$$

.

Population vs. Sample Statistics

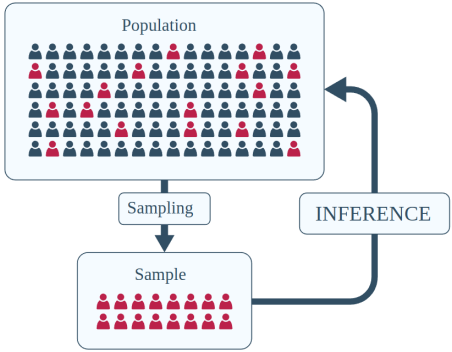
Population Statistics:

- Refers to measures computed using the entire population (ideal scenario).
- Denoted using Greek letters (e.g., μ, σ^2).

Sample Statistics:

- Refers to measures computed using a subset (sample) of the population (that's what we know from the data).
- Denoted using Latin letters (e.g., $\bar{x}$, s^2).

Making Inferences (or predictions)



We use data from an observed sample to make a conclusion about a population.

χ^2 Distribution

Let $Z_1, Z_2, \cdots, Z_\nu$ be the independent standard random variable. Then we define chi-square (χ^2) random variable as:

$$\chi^2(\nu) = Z_1^2 + Z_2^2 + \cdots + Z_\nu^2 \tag{7}$$

where ν is called the degree of freedom. We have the following results wrt χ^2 distribution:

- $\mathbb{E}[\chi^2(\nu)] = \nu$.
- We define $\chi^2(A, \nu)$ as $P(\chi^2(\nu) \leq \chi^2(A, \nu)) = A$.

Percentile in a Distribution

A percentile represents a value below which a certain percentage of observations in a distribution fall.
Example: The 90th percentile of a distribution means that 90% of the distribution lies to the left of this value, and 10% lies to the right.
Consider that X is a random variable, and we want to know a data point x, such that

$$P(X \leq x) = 0.90$$

which will gives a value of x such that 90 percent of the data points collected lies to left of x in a PDF graph.

Percentile in a χ^2 Distribution

If $X \sim \chi^2(\nu)$, the 90th percentile of the χ^2 distribution with $\nu = 5$ degrees of freedom is $x = \chi^2(.90; 5)$ that satisfies $P(X \leq x) = 0.90$.
Using either a table or programmatically, $x = 9.24$.

```
from scipy.stats import chi2
# Given percentile and degrees of freedom
percentile = 0.90
degrees_of_freedom = 5
# Calculate the 90th percentile of the chi-squared distribution
chi2_value = chi2.ppf(percentile, degrees_of_freedom)
print(f"The 90th percentile is: {chi2_value:.2f}")
```

PDF of χ^2 Distribution

The probability density functions for χ^2 distribution is

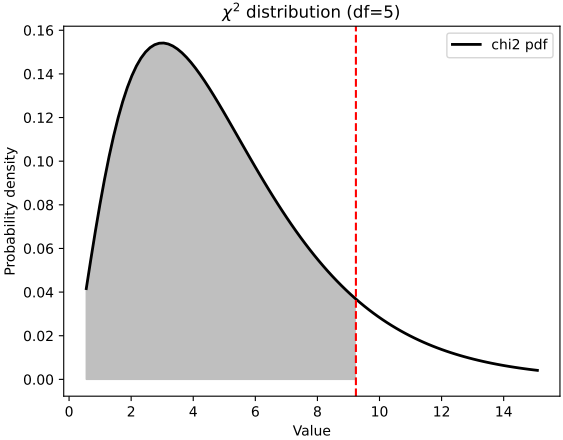
$$f(x) = \frac{1}{2^{\nu/2}\Gamma(\nu/2)} x^{(\nu/2)-1} e^{-x/2}, \quad x > 0 \tag{8}$$

If $y_1, y_2, \cdots, y_n$ are random samples drawn from $\mathcal{N}(\mu, \sigma^2)$, then

$$\frac{\sum(y_i - \overline{y})^2}{\sigma^2} \sim \chi^2(n-1) \tag{9}$$

Note: Some books write χ^2_ν instead of $\chi^2(\nu)$.

χ^2 Distribution



t Distribution

Consider, Z, a standard normal random variable, and χ^2_ν be another random variable so that they are independent, then t distribution, also known as student's t distribution, is written as

$$T_\nu = \frac{Z}{\left(\chi^2(\nu)/\nu\right)^{1/2}}.$$

t Distribution

The PDF of t distribution of ν degree of freedom is

$$f(t) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma(\nu/2)} \frac{1}{\left((t^2/\nu)+1\right)^{(\nu+1)/2}}, \quad -\infty < t < \infty. \tag{10}$$

If $y_1, y_2, \cdots, y_n$ are random samples drawn from $\mathcal{N}(\mu, \sigma^2)$, then

$$t = \frac{\overline{y} - \mu}{s/\sqrt{n}} \tag{11}$$

follows the t distribution with $n-1$ degrees of freedom. Here, s is the sample standard deviation.

F Distribution

If we have two random variable: χ^2_u and χ^2_v with u and v degrees of freedom respectively, then

$$F_{u,v} = \frac{\chi^2_u / u}{\chi^2_v / v} \tag{12}$$

follows F distribution with u numerator degree of freedom, and v denominator degree of freedom.

F Distribution

Consider two independent normal populations of the common variance σ^2. If $y_1, y_2, \cdots, y_n$ be random samples of n observations drawn from the first population, and $x_1, x_2, \cdots, x_m$ samples of m observations drawn from the second population, and if their sample variances are s_y^2 and s_x^2 respectively, then

$$\frac{s_y^2}{s_x^2} \sim F_{n-1,m-1}. \tag{13}$$

Confidence Interval

Definition

A confidence interval (CI) is the interval that a population parameter falls between a set of values, i.e. what may be the range of the estimated value. This is important, as estimated values are just estimates and there is some uncertainty about it. CI is measuring that uncertainty.

Confidence Interval

Assume that θ is the parameter, consider two statistics θ_L and θ_U such that

$$P(\theta_L \leq \theta \leq \theta_U) = 1 - \alpha \tag{14}$$

is true then the interval $\theta_L \leq \theta \leq \theta_U$ is called a $100(1-\alpha)$ percent confidence interval for the parameter θ. $1-\alpha$ is called confidence coefficient.
You can understand CI this way: in repeated random samplings, a large number of such intervals are constructed, and $100(1-\alpha)$ percent of them will contain the true value of θ.

Let's come back to SLR.

Confidence Interval on SLR Coefficients

As coefficients (called weights in some literature) $\hat{w}_0$ and $\hat{w}_1$ are estimates, they have some uncertainty.
We first look at the sampling distribution of $\hat{w}_1$.

The sampling distribution of $\hat{w}_1$ means that different values of $\hat{w}_1$ would be obtained with repeated sampling when the levels of the predictor variable x are held constant from sample to sample.

– Applied Linear Regression Models, *Kutner, Nachtsheim, Neter*

Expectations and Variance of $\hat{w}_1$

Expectations and variance of $\hat{w}_1$ are

$$\begin{aligned} \mathbb{E}[\hat{w}_1] &= w_1 \\ \sigma^2[\hat{w}_1] &= \frac{\sigma^2}{\sum(x_i - \overline{x})^2} \end{aligned} \tag{15}$$

where σ^2 is the assumed error variance.

Sample Variance for $\hat{w}_1$

Of course, we won't know the σ^2 because that's a mathematical assumption, so we replace it with MSE defined earlier and hence we have

$$s^2[\hat{w}_1] = \frac{\text{MSE}}{\sum(x_i - \overline{x})^2} \tag{16}$$

The t Distribution

In that case, we can follow one statistical theorem that says

> When a statistic is standardized but the denominator is an estimated standard deviation (or variance) rather than the true standard deviation (or variance), then the statistic follows **t distribution**.

Thus,

$$T = \frac{\hat{w}_1 - w_1}{s[\hat{w}_1]} \tag{17}$$

follows a t distribution with $n - 2$ degrees of freedom, i.e. $T \sim t_{n-2}$.

Confidence Interval for $\hat{w}_1$

We can write $(1-\alpha)$ CI for $\hat{w}_1$ as

$$
\begin{aligned}
1-\alpha &= P\Big(t(\alpha/2, n-2) < T < t(1-\alpha/2, n-2)\Big) \\
&= P\left(t(\alpha/2, n-2) < \frac{\hat{w}_1 - w_1}{s[\hat{w}_1]} < t(1-\alpha/2, n-2)\right)
\end{aligned} \tag{18}
$$

Final CI Expression for $\hat{w}_1$

Hence, we can write the following CI for $\hat{w}_1$:

$$\hat{w}_1 \pm t\left(1 - \frac{\alpha}{2}, n - 2\right) s[\hat{w}_1] \tag{19}$$

this is one number

Variance of $\hat{w}_0$

Similarly, $\hat{w}_0$ follows the Normal distribution $\mathcal{N}(w_0, \sigma^2[\hat{w}_0])$ where

$$\sigma^2[\hat{w}_0] = \sigma^2 \times \left(\frac{1}{n} + \frac{\overline{x}^2}{\sum(x_i - \overline{x})^2}\right) \tag{20}$$

Of course, we won't know σ^2 and hence, it is replaced by the MSE and we have the sample variance for $\hat{w}_0$ as

$$s^2[\hat{w}_0] = \text{MSE} \times \left(\frac{1}{n} + \frac{\overline{x}^2}{\sum(x_i - \overline{x})^2}\right) \tag{21}$$

Confidence Interval for $\hat{w}_0$

Similar to $\hat{w}_1$, the CI for $\hat{w}_0$ is

$$\hat{w}_0 \pm t\left(1 - \frac{\alpha}{2}, n - 2\right) s\left[\hat{w}_0\right] \tag{22}$$

Making Inference and CI on a New Predictor

Assume that we want to make a prediction at a new predictor x_h.

- The least square estimator will give $\hat{y}_h = \hat{w}_0 + \hat{w}_1 x_h$.

We find that the expectations and the sample variance of $\hat{y}_h$ are

$$\mathbb{E}[\hat{y}_h] = w_0 + w_1 x_h$$
$$s^2[\hat{y}_h] = \text{MSE} \times \left(\frac{1}{n} + \frac{(x_h - \overline{x})^2}{\sum (x_i - \overline{x})^2} \right) \tag{23}$$

Confidence Interval for $\hat{y}_h$

Fortunately, we won't need to know the exact w_0 and w_1 to construct the CI for y_h. It will just be

$$\hat{y}_h \pm t(1 - \frac{\alpha}{2}, n - 2) s[\hat{y}_h] \tag{24}$$

Additional Considerations for x_h

Note: The above CI is based on the assumption that x_h comes from the same trial as the original data. If x_h is obtained from another experiment, that is independent of the data used to develop the regression model, we may need to account for additional prediction error ϵ_h, which still has the same variance σ^2. In such a case, the sample variance will be

$$\text{MSE} \times \left(1 + \frac{1}{n} + \frac{(x_h - \overline{x})^2}{\sum(x_i - \overline{x})^2}\right).$$

Refer to Section 2.4 and 2.5 of **Applied Linear Regression Models, *Kutner, Nachtsheim, and Neter*, 4th Ed.**

Confidence Bands

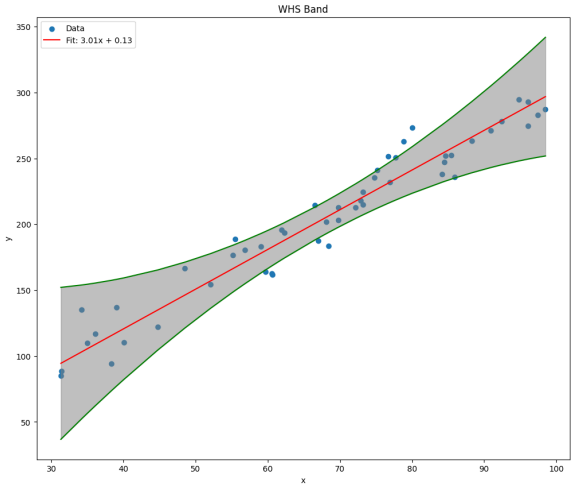
We may also want a confidence band for the entire regression line $\mathbb{E}[Y] = w_0 + w_1 X$.
It is given by Working & Hotelling (1929) and Scheffe (1953) as:

$$\hat{y}_h \pm W_\alpha s[\hat{y}] \tag{25}$$

for any value of y_h where $W_\alpha = \sqrt{2F(1-\alpha; 2, n-2)}$; F is the F distribution.

These bands, called WHS bands are in the shape of parabolas.

Confidence Band: An Example



Multiple Linear Regression using Matrix Approach

First-order Model with Two Predictor Variables

$$y_i = w_0 + w_1 x_{i1} + w_2 x_{i2} + \epsilon_i \tag{26}$$

In this case, our general assumption model, as seen in the single predictor variable will hold true, such as $\mathbb{E}[y_i] = w_0 + w_1 x_{i1} + w_2 x_{i2}$.

In addition, the following value holds, as you see by taking the partial derivatives of the model:

$$\frac{\partial \mathbb{E}[Y]}{\partial X_1} = w_1 \qquad \frac{\partial \mathbb{E}[Y]}{\partial X_2} = w_2 \tag{27}$$

First-order Model with more than Two Predictor Variables

Consider a regression model with $p-1$ predictor variable:

$$\begin{aligned} y_i &= w_0 + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_{p-1} x_{i(p-1)} \epsilon_i \\ &= w_0 + \sum_{k=1}^{p-1} w_k x_{ik} + \epsilon_i \end{aligned} \tag{28}$$

If we set $x_{i0} = 1$, then we can write:

$$y_i = \sum_{k=0}^{p-1} w_k x_{ik} + \epsilon_i \tag{29}$$

Meaning of Linearity in the Regression Model

The linear model implies that the model is linear in its parameters, i.e. it can be expressed as

$$y_i = \sum c_{i0} w_i$$

The model such as

$$y_i = w_0 \exp(w_1 x_i) + \epsilon_i$$

is not linear because we cannot express it as $\sum c_{i0} w_i$.

Matrix form of the Linear Regression Model

$$\mathbf{y}_{n\times 1} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{x}_{n\times p} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1(p-1)} \\ 1 & x_{21} & x_{22} & \cdots & x_{2(p-1)} \\ \vdots & \vdots & \vdots & \vdots & \\ 1 & x_{n1} & x_{n2} & \cdots & x_{n(p-1)} \end{bmatrix} \tag{30}$$

$$\mathbf{w}_{p\times 1} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{p-1} \end{bmatrix} \qquad \boldsymbol{\epsilon}_{n\times 1} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix} \tag{31}$$

Linear Regression Model in Matrix Form

In matrix form, we can write it as

$$\mathbf{y}_{n\times 1} = \mathbf{x}_{n\times p}\mathbf{w}_{p\times 1} + \boldsymbol{\epsilon}_{n\times 1} \tag{32}$$

Expectation and Variance-Covariance Matrix

Here, $\mathbb{E}[\boldsymbol{\epsilon}] = 0$, and variance-covariance matrix is

$$\sigma^2[\boldsymbol{\epsilon}] = \begin{bmatrix} \sigma^2 & 0 & 0 & \cdots & 0 \\ 0 & \sigma^2 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \\ 0 & 0 & \cdots & 0 & \sigma^2 \end{bmatrix} = \sigma^2 \mathbf{I} \tag{33}$$

In this case, instead of random variable Y, and X, we have random vector $\mathbf{Y}$, and $\mathbf{X}$, and $\mathbb{E}[\mathbf{Y}] = \mathbf{X}\mathbf{w}$.

Least Square Estimator/Cost Function for MLR

Cost Function

$$J = \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \sum_{k=0}^{p-1} w_k x_{ik}\right)^2$$
$$= \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \mathbf{w}^\top \mathbf{x}_i\right)^2 = \frac{1}{2n}(\mathbf{y} - \mathbf{x}\mathbf{w})^\top(\mathbf{y} - \mathbf{x}\mathbf{w}) \quad (34)$$

Normal Equation for MLR

Normal Equation

$$\mathbf{x}^\top \mathbf{x}\hat{\mathbf{w}} = \mathbf{x}^\top \mathbf{y} \tag{35}$$

How to solve this for $\mathbf{w}$?

Deriving the Solution

Normal Equation

$$\frac{\partial J}{\partial \mathbf{w}} = -\frac{1}{2n}\mathbf{x}^\top(\mathbf{y} - \mathbf{x}\mathbf{w}) = 0 \Rightarrow -\frac{1}{2n}(\mathbf{x}^\top\mathbf{y} - \mathbf{x}^\top\mathbf{x}\mathbf{w}) = 0 \tag{36}$$

Final Solution and Inverse Condition

Normal Equation

$$\hat{\mathbf{w}} = \frac{1}{n}(\mathbf{x}^\top \mathbf{x})^{-1}\mathbf{x}^\top \mathbf{y}$$

provided $(\mathbf{x}^\top \mathbf{x})^{-1}$ exists.

What's the condition for an inverse to exist?

Estimated Coefficients/Weight in Python

```
def linear_regression(x, y):
    return np.dot( np.dot(np.linalg.inv(np.dot(x.T, x)), x.T), y)
```

Fitted Value/How do we make the prediction?

Fitted Value

$$\hat{y} = ?$$

Fitted Value/How do we make the prediction?

Fitted Value

$$\hat{\mathbf{y}} = \mathbf{x}\hat{\mathbf{w}} \tag{37}$$

Discussion on Numerical Implementation

Training/Testing

- Train:= $\hat{w} = \frac{1}{n}(\mathbf{x}_{tr}^{\top}\mathbf{x})^{-1}\mathbf{x}_{tr}^{\top}\mathbf{y}_{tr}$.
- Test:=$\hat{y} = \hat{w}\mathbf{x}_{test}$.
- $\mathbf{x}^{\top}\mathbf{x}$ is not guaranteed to be invertible for any arbitrary dataset.
- If p or n or both are large, it would be computationally expensive to implement this model.
- If p is very large, computing the inverse of a large matrix will be computationally challenging.

Code Availability and Python Notebook

Code used for this chapter is available at https://github.com/rahulbhadani/CPE490_590_Sp2025/blob/master/Code/Chapter_04_Linear_Regression.ipynb

The End