COLUMBIA | ENGINEERING
EXECUTIVE EDUCATION

Week 16
Evaluating Data Models

Applied Data Science

Columbia University - Columbia Engineering

Course Agenda

- Week 10: Organizing and Analyzing Data with NumPy and Pandas
- Week 11: Cleaning and Visualizing Data with Pandas and Matplotlib
- Week 12: Statistical Distributions
- Week 13: Statistical Sampling
- Week 14: Hypothesis Testing
- Week 15: Regression Models in Python
- **Week 16: Evaluating Data Models**
- Week 17: Classification with K-Nearest Neighbors
- Week 18: Decision Tree Models
- Week 19: Clustering Models
- Week 20: Text Mining in Python -- Analyzing Sentiment
- Week 21: Text Mining in Python -- Topic Modeling

Data set 1: Rocks vs. Mines

- Independent variables: sonar soundings at different frequencies
- Dependent variable (target): Rock or Mine

Data Set 1: Rocks vs. Mines

In [1]:

```
import pandas as pd
from pandas import DataFrame
url="https://archive.ics.uci.edu/ml/machine-learning-databases/undocumented/connection
df = pd.read_csv(url,header=None)
df.describe()
```

Out[1]:

	0	1	2	3	4	5	6	7	8	9	...	50
count	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	...	208.000000
mean	0.029164	0.038437	0.043832	0.053892	0.075202	0.104570	0.121747	0.134799	0.178003	0.208259	...	0.016069
std	0.022991	0.032960	0.038428	0.046528	0.055552	0.059105	0.061788	0.085152	0.118387	0.134416	...	0.012008
min	0.001500	0.000600	0.001500	0.005800	0.006700	0.010200	0.003300	0.005500	0.007500	0.011300	...	0.000000
25%	0.013350	0.016450	0.018950	0.024375	0.038050	0.067025	0.080900	0.080425	0.097025	0.111275	...	0.008425
50%	0.022800	0.030800	0.034300	0.044050	0.062500	0.092150	0.106950	0.112100	0.152250	0.182400	...	0.013900
75%	0.035550	0.047950	0.057950	0.064500	0.100275	0.134125	0.154000	0.169600	0.233425	0.268700	...	0.020825
max	0.137100	0.233900	0.305900	0.426400	0.401000	0.382300	0.372900	0.459000	0.682800	0.710600	...	0.100400

8 rows × 60 columns

See all columns

In [2]:
```
pd.options.display.max_columns=70
df.describe()
```

Out[2]:

	0	1	2	3	4	5	6	7	8	9	10	11
count	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000
mean	0.029164	0.038437	0.043832	0.053892	0.075202	0.104570	0.121747	0.134799	0.178003	0.208259	0.236013	0.250221
std	0.022991	0.032960	0.038428	0.046528	0.055552	0.059105	0.061788	0.085152	0.118387	0.134416	0.132705	0.140072
min	0.001500	0.000600	0.001500	0.005800	0.006700	0.010200	0.003300	0.005500	0.007500	0.011300	0.028900	0.023600
25%	0.013350	0.016450	0.018950	0.024375	0.038050	0.067025	0.080900	0.080425	0.097025	0.111275	0.129250	0.133475
50%	0.022800	0.030800	0.034300	0.044050	0.062500	0.092150	0.106950	0.112100	0.152250	0.182400	0.224800	0.249050
75%	0.035550	0.047950	0.057950	0.064500	0.100275	0.134125	0.154000	0.169600	0.233425	0.268700	0.301650	0.331250
max	0.137100	0.233900	0.305900	0.426400	0.401000	0.382300	0.372900	0.459000	0.682800	0.710600	0.734200	0.706000

8 rows × 60 columns

Data Set 1: Rocks vs. Mines

```
In [2]: pd.options.display.max_columns=70
        df.describe()
```

Out[2]:

	0	1	2	3	4	5	6	7
count	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000
mean	0.029164	0.038437	0.043832	0.053892	0.075202	0.104570	0.121747	0.134799
std	0.022991	0.032960	0.038428	0.046528	0.055552	0.059105	0.061788	0.085152
min	0.001500	0.000600	0.001500	0.005800	0.006700	0.010200	0.003300	0.005500
25%	0.013350	0.016450	0.018950	0.024375	0.038050	0.067025	0.080900	0.080425
50%	0.022800	0.030800	0.034300	0.044050	0.062500	0.092150	0.106950	0.112100
75%	0.035550	0.047950	0.057950	0.064500	0.100275	0.134125	0.154000	0.169600
max	0.137100	0.233900	0.305900	0.426400	0.401000	0.382300	0.372900	0.459000

Examine the distribution of the data in column 4

```
<li>Quartile 1: from .0067 to .03805
<li>Quartile 2: from .03805 to .0625
<li>Quartile 3: from .0625 to .100275
<li>Quartile 4: from .100275 to .401

<h4>Quartile 4 is much larger than the other quartiles. This raises the
outliers</h4>
```

Examine the distribution of the data in column 4

- Quartile 1: from .0067 to .03805
- Quartile 2: from .03805 to .0625
- Quartile 3: from .0625 to .100275
- Quartile 4: from .100275 to .401

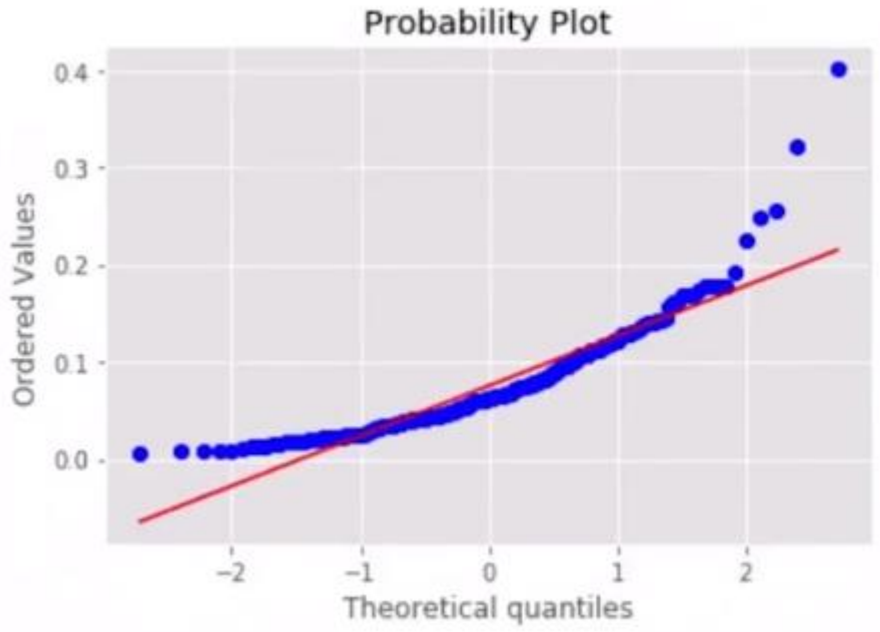
Quartile 4 is much larger than the other quartiles. This raises the possibility of outliers

A Quantile - Quantile (qq) plot can help identify outliers

- y-axis contains values
- x-axis is the cumulative normal density function plotted as a straight line (-3 to +3)
- y-axis is the values ordered from lowest to highest
- the closer the curve is to the line, the more it reflects a normal distribution

```
In [3]: import numpy as np
        import pylab
        import scipy.stats as stats
        import matplotlib
        import matplotlib.pyplot as plt
        matplotlib.style.use('ggplot')
        %matplotlib inline

        stats.probplot(df[4], dist="norm", plot=pylab)
        pylab.show()
```



Data Set 1: Rocks vs. Mines

Examine the dependent variable

```
In [4]: df[60].unique()
```

```
Out[4]: array(['R', 'M'], dtype=object)
```

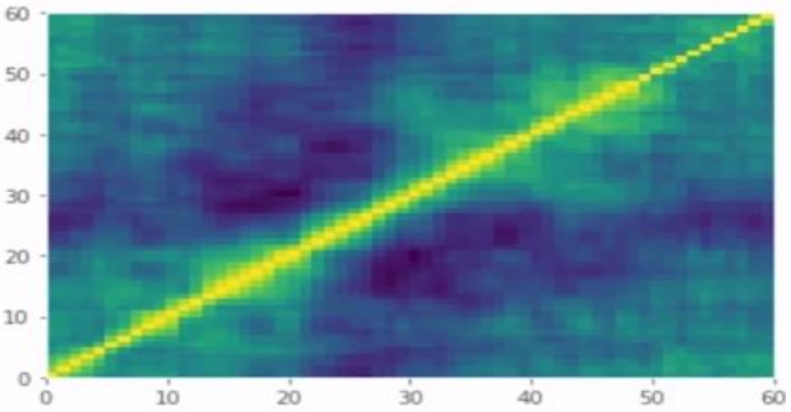
Examine correlations

```
In [ ]: df.corr()
```

```
In [ ]: import matplotlib.pyplot as plot
        plot.pcolor(df.corr())
        plot.show()
```

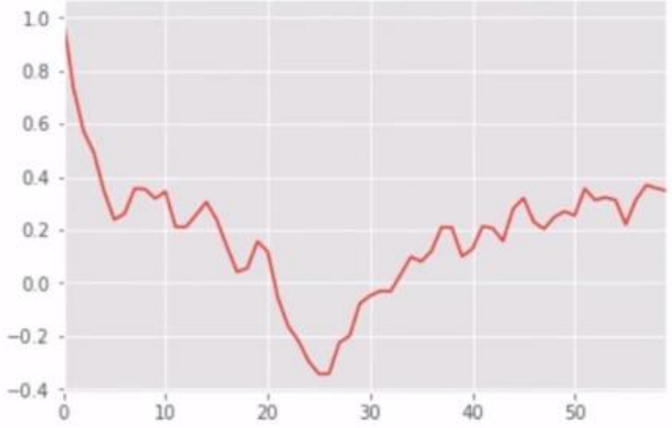
```
In [ ]: df.corr()[0].plot()
```

```
In [6]: import matplotlib.pyplot as plot
        plot.pcolor(df.corr())
        plot.show()
```




```
In [7]: df.corr()[0].plot()
```

```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x1169575f8>
```



Highly correlated items = not good!

Low correlated items = good

Correlations with target (dv) = good (high predictive power)

Data Set 2: Wine data

- Independent variables: Wine composition (alcohol content, sulphites, acidity, etc.)
- Dependent variable (target): Taste score (average of a panel of 3 wine tasters)

Data Set 2: Wine Data

- Independent variables: Wine composition (alcohol content, sulphites, acidity, etc.)
- Dependent variable (target): Taste score (average of a panel of 3 wine tasters)

```
In [8]: url = "http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/win
        import pandas as pd
        from pandas import DataFrame
        w_df = pd.read_csv(url,header=0,sep=';')
        w_df.describe()
```

Out[8]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.00000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000

```
In [9]: w_df['volatile acidity']

Out[9]: 0        0.700
        1        0.880
        2        0.760
        3        0.280
        4        0.700
        5        0.660
        6        0.600
        7        0.650
        8        0.580
        9        0.500
        10       0.580
        11       0.500
        12       0.615
        13       0.610
        14       0.620
        15       0.620
        16       0.280
        17       0.560
        18       0.590
        19       0.320
        20       0.220
```

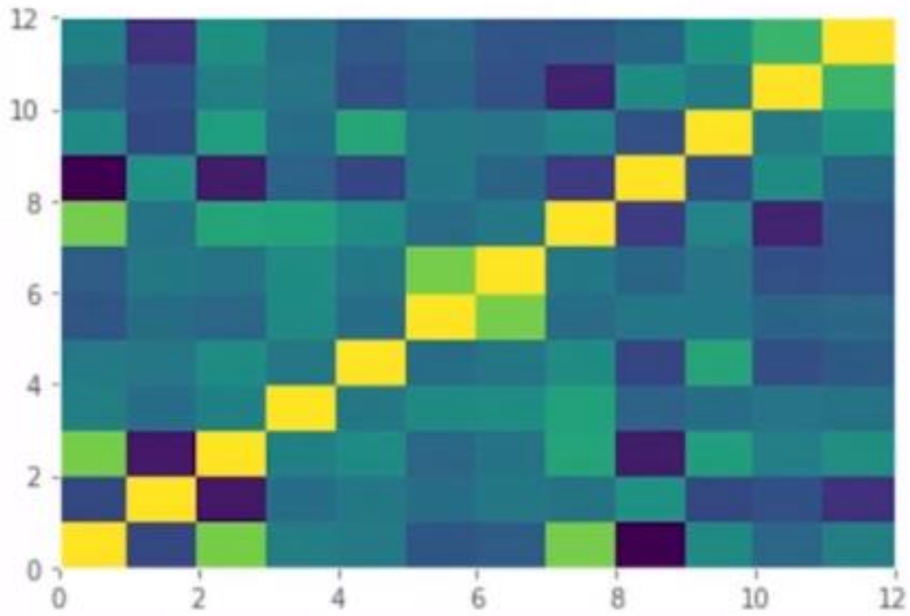
```
In [10]: w_df.corr()
```

Out[10]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400
free sulfur dioxide	-0.153794	-0.010504	-0.060978	0.187049	0.005562	1.000000	0.667666
total sulfur dioxide	-0.113181	0.076470	0.035533	0.203028	0.047400	0.667666	1.000000
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021946	0.071269

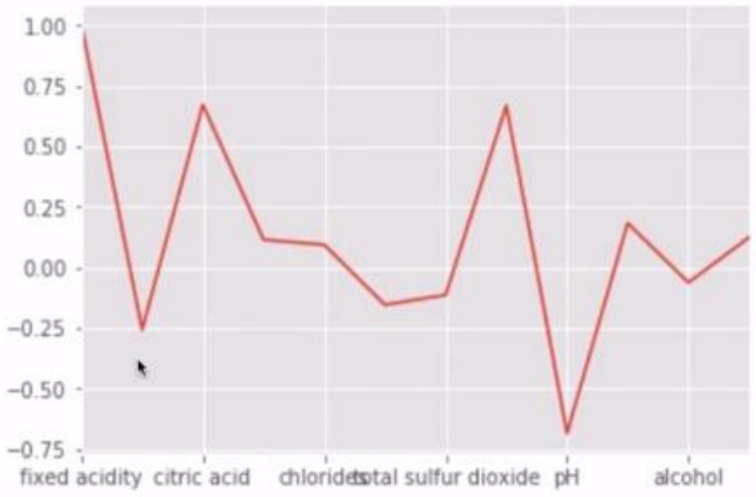
Data Set 2: Wine Data

```
In [11]: import matplotlib.pyplot as plot
         plot.pcolor(w_df.corr())
         plot.show()
```



Examining the correlation of one variable with the others

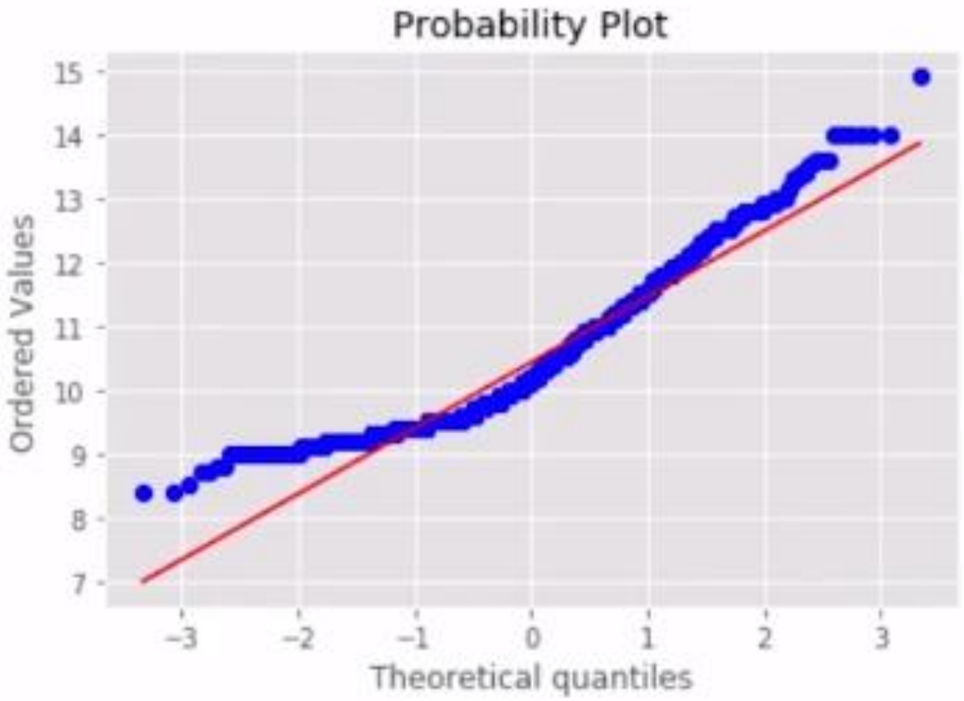
```
In [12]: w_df.corr()['fixed acidity'].plot()
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x116a9c390>
```



And we can examine quintile plots as we did with the rocks and mines data

In [14]:
```
import numpy as np
import pylab
import scipy.stats as stats
%matplotlib inline

stats.probplot(w_df['alcohol'], dist="norm", plot=pylab)
pylab.show()
```



Training a Classifier on Rocks vs. Mines

COLUMBIA ENGINEERING
EXECUTIVE EDUCATION

```
In [ ]: import numpy
        import random
        from sklearn import datasets, linear_model
        from sklearn.metrics import roc_curve, auc
        import pylab as pl
```

```
In [ ]: import pandas as pd
        from pandas import DataFrame
        url="https://archive.ics.uci.edu/ml/machine-learning-databases/undocumented/connect
        df = pd.read_csv(url,header=None)
        df.describe()
```

Convert labels R and M to 0 and 1

```
In [17]: df[60]=np.where(df[60]=='R',0,1)
```

Divide the dataset into training and test samples

Separate out the x and y variable frames for the train and test samples

```
In [ ]: train
```

```
In [18]: from sklearn.model_selection import train_test_split
         train, test = train_test_split(df, test_size = 0.3)
         x_train = train.iloc[0:,0:60]
         y_train = train[60]
         x_test = test.iloc[0:,0:60]
         y_test = test[60]
         y_train
```

```
Out[18]: 6      0
         193    1
         179    1
```

What is Optimization?

- Mathematical models of real-world problems
  - **Goal**: Find "**best**" possible decision or solution

- Variables represent decisions
- Constraints model the problem dynamics
- Objective models performance metric (such as cost, profit etc)

Our Goals

- Understanding optimization formulations
  - strong emphasis on modeling

- Use of Solvers (R, Gurobi,. . .)
  - extremely powerful even for large scale problems

- Linear Optimization: linear constraints and objectives
- Integer Optimization: integer valued decision variables
- Convex Optimization: convex constraints and/or objectives
- Non-linear optimization: general non-linear constraints
- Optimization under uncertainty: uncertain constraints
  - stochastic optimization, robust optimization
  - online optimization

Example 1: Transportation Problem (minimum cost)

- Two factories F1, F2 with supplies 100, 200 units resp.

- Three shops S1, S2, S3 with demands 50, 100, 150 units resp.

- Shipping costs (in $ per unit)

	S1	S2	S3
F1	10	12	14
F2	11	12	13

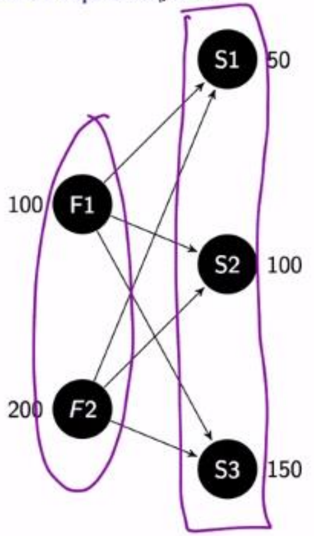
$F1 \longrightarrow S1$

\$10 per unit

- Problem: Minimize shipping cost

Example 1: Transportation Problem (minimum cost)

Example 1: Network Representation



Example 1: Constraints

- Supply: Can not ship more units from a factory than available
  - F1: $x_{11} + x_{12} + x_{13} \leq 100$
  - F2: $x_{21} + x_{22} + x_{23} \leq 200$
- Demand: Can not ship less units to a shop than needed
  - S1: $x_{11} + x_{21} \geq 50$
  - S2: $x_{12} + x_{22} \geq 100$
  - S3: $x_{13} + x_{23} \geq 150$
- Non-negativity: Can not ship negative amount of units
  - $x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23} \geq 0$

Example 1: Decisions and objective

- Decision x_{ij}: Number of units to ship from factory i to shop j
  (can be fractional)
- Objective: Minimize shipping cost
  - $\underset{x}{\text{minimize}}\ 10x_{11} + 12x_{12} + 14x_{13} + 11x_{21} + 12x_{22} + 13x_{23}$

Example 1: Linear program (LP)

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & 10x_{11} + 12x_{12} + 14x_{13} + 11x_{21} + 12x_{22} + 13x_{23} \\
\text{subject to} \quad & x_{11} + x_{12} + x_{13} \leq 100 \\
& x_{21} + x_{22} + x_{23} \leq 200 \\
& x_{11} + x_{21} \geq 50 \\
& x_{12} + x_{22} \geq 100 \\
& x_{13} + x_{23} \geq 150 \\
& x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23} \geq 0
\end{aligned}
\tag{1}
$$

Example 2: Cash Flows

- We have \$100 to invest

- Three investment options:
  a. for every \$1 invested now, we get \$0.1 one year from now, and \$1.3 three years from now
  b. for every \$1 invested now, we get \$0.2 one year from now, and \$1.1 two years from now
  c. for every \$1 invested a year from now, we get \$1.5 three years from now

- Money-market account: 2% per year (denote it by "d")

- Goal: Maximize cash at the end of year 3

Example 2: Cash Flows

Example 2: Decisions, other variables, and objective

- Decisions x_a, x_b, x_c, x_d: Dollars invested in each option
- Other variables y_1, y_2, y_3: Cash available at the end of each year
- Objective: Maximize cash at the end of year 3
  - $\underset{x,y}{\text{maximize}}\ y_3$

Example 2: Cash flow diagram

x_a : a) \$1 → \$0.1 → \$1.3

1 2 3

x_b : b) \$1 → \$0.2 → \$1.1

x_c : c) \$1 → \$1.5

x_d :

$$y_1 = 0.1x_a + 0.2x_b + 1.02x_d$$

$$x_c \leq y_1$$

$$y_2 = 1.1x_b + (y_1 - x_c)1.02$$

$$y_3 = 1.02y_2 + 1.3x_a + 1.5x_c$$

Example 2: Constraints

- Budget
  - $x_a + x_b + x_d = 100$
- Logical relationships (flow balance)
  - $y_1 = 0.1x_a + 0.2x_b + 1.02x_d$
  - $x_c \leq y_1$ (why?)
  - $y_2 = 1.1x_b + 1.02(y_1 - x_c)$
  - $y_3 = 1.02y_2 + 1.3x_a + 1.5x_c$
- Non-negativity
  - $x_a, x_b, x_c, x_d \geq 0$

Example 2: LP

$$
\begin{aligned}
\underset{x,y}{\text{maximize}} \quad & y_3 \\
\text{subject to} \quad & x_a + x_b + x_d = 100 \\
& y_1 = 0.1x_a + 0.2x_b + 1.02x_d \\
& x_c \leq y_1 \\
& y_2 = 1.1x_b + 1.02(y_1 - x_c) \\
& y_3 = 1.02y_2 + 1.3x_a + 1.5x_c \\
& x_a, x_b, x_c, x_d \geq 0
\end{aligned}
\tag{2}
$$

- Decision variables: $\mathbf{x} = (x_1, ..., x_n)$
- Objective function: $f(\mathbf{x})$ (linear in $\mathbf{x}$)

$$f(x) = c_0 + \sum_{j=1}^{n} c_j x_j$$

- Constraint i: $g_i(\mathbf{x}) \geq 0$ (linear in $\mathbf{x}$)

$i = 1, ... m$

$$g_i(x) = a_{i0} + \sum_{j=1}^{n} a_{ij} x_j \geq 0$$

$$g_i(x) = x_j \geq 0$$

- Mathematical model:

$$\boxed{\begin{array}{ll} \underset{\mathbf{x}}{\text{minimize}} & f(\mathbf{x}) \\ \text{subject to} & g_i(\mathbf{x}) \geq 0 \quad \forall i \end{array}} \qquad (3)$$

$g_i(x) \geq 0$

$-g_i(x) \geq 0$

- LP: **Both** $f(\mathbf{x})$ and $g_i(\mathbf{x})$ are linear functions of $\mathbf{x}$

Introduction: Linear vs. non-linear functions

Suppose $\mathbf{x} = (x_1, x_2)$

Linear functions of $\mathbf{x}$

- $x_1 + x_2$
- $2x_1 - 3x_2$
- $a_1x_1 + a_2x_2$, where $(a_1, a_2) \in \mathbb{R}^2$

Non-linear functions of $\mathbf{x}$

- $x_1 + x_2^2$
- $\sin(x_1) + e^{x_2}$
- $\log(x_1 + x_2)$
- Any function not of the form $a_1x_1 + a_2x_2$, where $(a_1, a_2) \in \mathbb{R}^2$

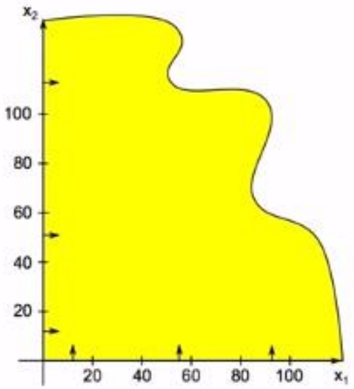
Graphical look at linear programming

Consider the following LP:

$$\begin{array}{rrcrcl} \underset{x_1, x_2}{\text{maximize}} & 3x_1 & + & 2x_2 & & \\ & x_1 & + & x_2 & \leq & 80 \\ & 2x_1 & + & x_2 & \leq & 100 \\ & x_1 & & & \leq & 40 \\ & & & x_1, x_2 & \geq & 0 \end{array}$$

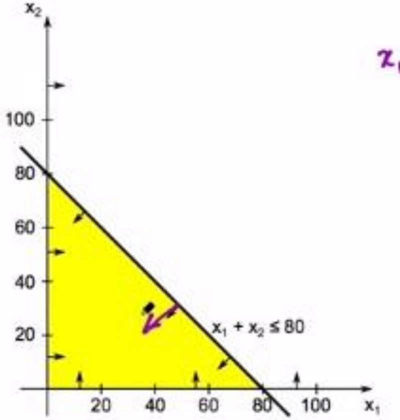
COLUMBIA ENGINEERING EXECUTIVE EDUCATION

Graphical look: $x_1, x_2 \geq 0$

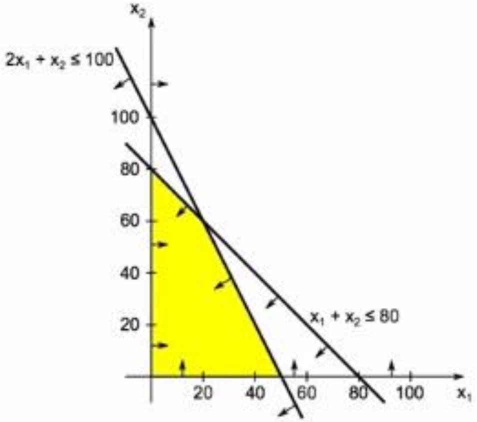


Graphical look: Add $x_1 + x_2 \leq 80$

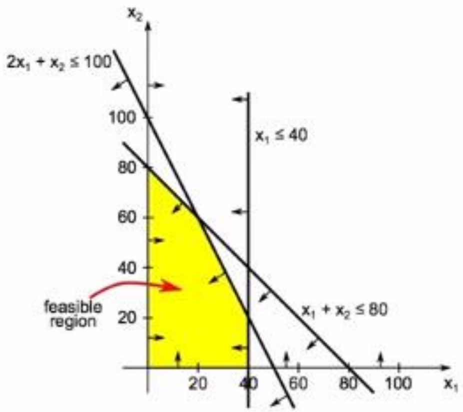
$x_1 + x_2 = 80$



Graphical look: Add $2x_1 + x_2 \leq 100$

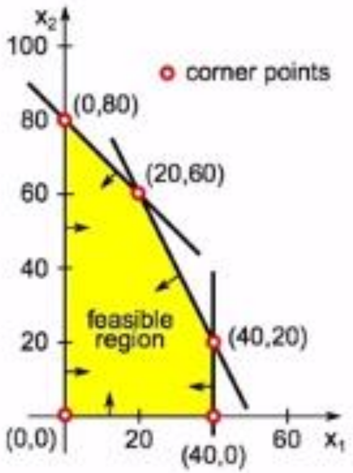


Graphical look: Add $x_1 \leq 40$

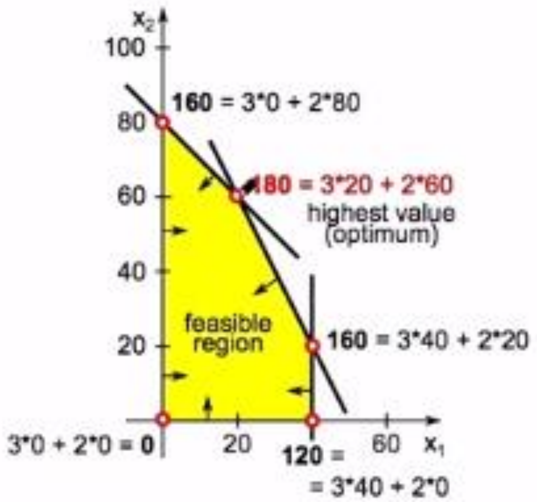


Graphical look: Corner/extreme points

If an LP has a finite optimum and the feasible region has at least one corner point, then there is a corner point optimal solution.



Graphical look: Obj. function value at the corner points



Linear Regression

- **Data**: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N) \in \mathbb{R}^n \times \mathbb{R}$
- **Goal**: Find the best **linear** model that explains the data, i.e., find $\boldsymbol{a} \in \mathbb{R}^n$, $b \in \mathbb{R}$ s.t.

$$y_i \approx \boldsymbol{a}^T \boldsymbol{x}_i + b, \quad 1 \le i \le N.$$

- Traditional approach:

$$\underline{\min_{\boldsymbol{a},b}} \left\{ \sum_{i=1}^{N} (\boldsymbol{a}^T \boldsymbol{x}_i + b - y_i)^2 \right\}.$$

$e_i^2 = (a^T x_i + b - y_i)^2$

Linear Regression

- **Data**: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N) \in \mathbb{R}^n \times \mathbb{R}$
- **Goal**: Find the best **linear** model that explains the data.
- **Alternate approach**:

$$\min_{\boldsymbol{a},b} \max_{i=1}^{N} |\boldsymbol{a}^T \boldsymbol{x}_i + b - y_i|.$$

$(a^T x_i + b - y_i)$

$$\min\ e$$
$$\boxed{e \geq e_i \quad \text{for all } i = 1 \ldots N}$$
$$e_i \geq (a^T x_i + b - y_i)$$
$$e_i \geq -(a^T x_i + b - y_i)$$

$(e_i \geq |a^T x_i + b - y_i|)$

Linear Regression

- **Data**: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N) \in \mathbb{R}^n \times \mathbb{R}$
- **Goal**: Find the best **linear** model that explains the data
- **Third approach**:

$$\min_{\boldsymbol{a}, b} \sum_{i=1}^{N} |\boldsymbol{a}^T \boldsymbol{x}_i + b - y_i|.$$

$$\rightarrow \min \sum_{i=1}^{N} e_i$$

$$\left.\begin{array}{l} e_i \geq (a^T x_i + b - y_i) \\ e_i \geq -(a^T x_i + b - y_i) \end{array}\right\} \quad e_i \geq |a^T x_i + b - y_i|$$

EMERITUS