Lecture 1 Part II Basic Data Science

Phayung Meesad, Ph.D.
King Mongkut's University of Technology
North Bangkok (KMUTNB)
Bangkok Thailand

Outlines

- Data
- Type of Data Attributes
- Data Exploration
- Data Visualization
- Python Libraries for Data Science

What is Data?

- Collection of data objects and their attributes
- An attribute is a property or characteristic of an object
  - Examples: eye color of a person, temperature, etc.
  - Attribute is also known as variable, field, characteristic, or feature
- A collection of attributes describe an object
  - Object is also known as record, point, case, sample, entity, or instance

Example of data

Attributes

Objects

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

Attributes Data Types

- Numeric
  - Continuous/Discrete
  - Binary/Real/Integer
  - Ratio/Interval
- Nominal/Ordinal
  - Binominal
  - Polynominal
- String/Text
- Date/Time

Record Data

- Data that consists of a collection of records, each of which consists of a fixed set of attributes

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

Data Matrix

- If data objects have the same fixed set of numeric attributes, then the data objects can be thought of as points in a multi-dimensional space, where each dimension represents a distinct attribute

- Such data set can be represented by an m by n matrix, where there are m rows, one for each object, and n columns, one for each attribute

Projection of x Load	Projection of y load	Distance	Load	Thickness
10.23	5.27	15.22	2.7	1.2
12.65	6.25	16.22	2.2	1.1

Document Data

- Each document becomes a `term' vector,
  - each term is a component (attribute) of the vector,
  - the value of each component is the number of times the corresponding term occurs in the document.

	team	coach	pla y	ball	score	game	wi n	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

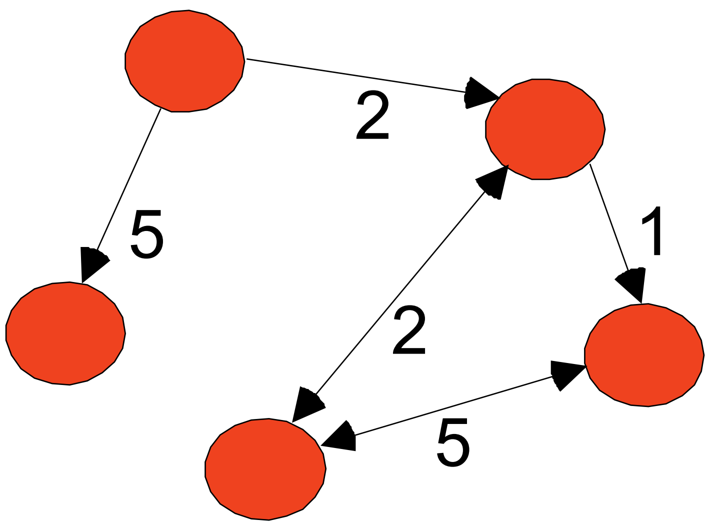
Transaction Data

- A special type of record data, where
  - each record (transaction) involves a set of items.

TID	*Items*
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Graph Data

- Examples: Generic graph and HTML Links



Data Mining

Graph Partitioning

Parallel Solution of Sparse Linear System of Equations

<a href="papers/papers.html#ffff">
N-Body Computation and Dense Linear System Solvers

Python for Data Science

- Python is an interpreted high-level general-purpose programming language.
- Python's design philosophy emphasizes code readability with its notable use of significant indentation.
- Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Tools for Data Science

python
NumPy
pandas
scikit learn
SciPy
matplotlib
plotly
Seaborn
TensorFlow
PySpark
Theano
Keras
R
Data Mining and Machine Learning
SymPy
rapidminer
Open for Innovation KNIME
WEKA The University of Waikato
orange DATA MINING FRUITFUL&FUN

Basic Mathematics

- Numeric Types
  - int
    - >>> type(1)
    - >>> 2
  - float
    - >>> type(1.)
    - >>> 2.
  - complex
    - >>> 1 + 2j
    - >>> 3 – 4j

- Operations

```
x = -10; y = 3  # assign variables
x + y           # addition
x - y           # subtraction
x * y           # multiplication
x / y           # division
x // y          # floor division
x % y           # modulo
x ** y          # power
pow(x,y)        # power
math.sqrt(x)    # square root (import math)
```

Basic Mathematics

```
c = 3 - 4j
c.real
c.imag
abs(c)
int(c.real)
float(c.imag)
```

```
A = 3; B= 2.
type(A)
type(B)
C = A + B
D = A / B
E = A ** B
F = B / A
```

Boolean / String

- Boolean: bool

```
A = True
B = False
print(type(A + B))
print(A or B)
print(type(A | B))
```

- String: str

```
str1 = "     Hello "
str2 = "PyThon     "
print(str1 + str2)
str1.upper()
str2.lower()
str1.count('l')
str2 = str2.replace('T','t')
str3 = str1+ str2
str3.strip()
```

List

- Lists

names =['Somchai','Somying','Somjai','Somsak','Somsri']

height = [175, 160, 157, 182, 165]

weight = [71, 46, 44, 80, 60]

persons =[names, height, weight]

persons[0]

persons[1]

persons[2]

persons[0][0]

persons[1][0]

persons[2][0]

persons[0][-1]

persons[1][-3:-1]

persons[0][2:5]

START : END

Inclusive : Not inclusive

List Methods

mylist.index(item)

mylist.count(item)

mylist.append(item)

mylist.remove(item)

mylist.pop(-1)

del(mylist[start:stop+1])

mylist.sort()

mylist.sort(reverse=True)

mylist.reverse()

Numpy

1) Low-level: what does it do?
2) High-level: why do we need it?

- The core library
- Central object: The Numpy Array
- Be familiar with Python Lists
- One sentence summary: “Linear algebra and a bit of probability”

Numpy

- NumPy is often used along with packages like **SciPy** (Scientific Python) and **Matplotlib** (plotting library).
- This combination is widely used as a replacement for MatLab, a popular platform for technical computing.
- However, Python alternative to MatLab is now seen as a more modern and complete programming language.

Scalars, Vectors and Matrices

- A scalars is a single value; there is just a value without direction.
- A vectors contain two or more scalar values in an array. A vector contains both a magnitude and a direction. A vector can be a row vector or a column vector.
- A matrix contain at least a vector.

Scalar	Row vector	Column vector	Matrix
10	$\begin{bmatrix} 10 & 20 \end{bmatrix}$	$\begin{bmatrix} 10 \\ 20 \end{bmatrix}$	$\begin{bmatrix} 10 & 40 & 70 \\ 20 & 50 & 80 \\ 30 & 60 & 90 \end{bmatrix}$ Row x Column

Dot/Scalar/Inner Product

- The dot product or scalar product or Inner project is an algebraic operation that takes two equal-length sequences of numbers and returns a single number.
- The dot product of two vectors $\mathbf{a} = [a_1, a_2, \dots, an]$ and $\mathbf{b} = [b_1, b_2, \dots, bn]$ is defined as:

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T\mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

Matrix Multiplication

$$\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{bmatrix}$$

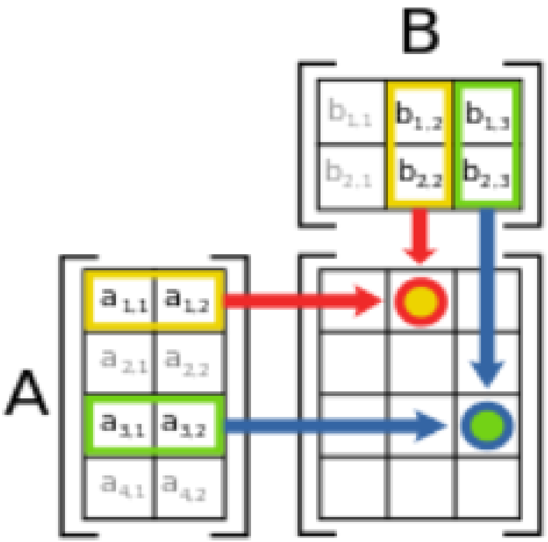
$$\mathbf{C} = \begin{bmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nn} \end{bmatrix}$$

$$\mathbf{C} = \mathbf{AB}$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj}$$

Matrix Multiplication

2D Matrix multiplication is possible when the number of columns in tensor **A** matches the number of rows in tensor **B**. In this case, the product of tensor **A** with size (x, y) and tensor **B** with size (y, z) results in a tensor of size (x, z)



$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \times \begin{bmatrix} m & n \\ p & q \\ r & s \end{bmatrix}$$

$$= \begin{bmatrix} (am + bp + cr) & (an + bq + cs) \\ (dm + ep + fr) & (dn + eq + fs) \end{bmatrix}$$

Elementwise Product

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \circ \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & a_{13}b_{13} \\ a_{21}b_{21} & a_{22}b_{22} & a_{23}b_{23} \\ a_{31}b_{31} & a_{32}b_{32} & a_{33}b_{33} \end{bmatrix}$$

Matrix Determinant

- The determinant of a product of matrices is the product of their determinants (the preceding property is a corollary of this one). The determinant of a matrix A is denoted det(A), det A, or |A|.
- The determinant is a scalar value that is a function of the entries of a square matrix. It allows characterizing some properties of the matrix and the linear map represented by the matrix.
- The determinant is nonzero if and only if the matrix is invertible, and the linear map represented by the matrix is an isomorphism.

In the case of a 2 × 2 matrix the determinant can be defined as

$$|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

Similarly, for a 3 × 3 matrix *A*, its determinant is

$$|A| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

$$= aei + bfg + cdh - ceg - bdi - afh.$$

Matrix Inverse

- To determine the inverse, we calculate a matrix of cofactors (adjugate matrix):

$$\mathbf{A}^{-1} = \frac{1}{|\mathbf{A}|}\mathbf{C}^{\mathrm{T}} = \frac{1}{|\mathbf{A}|}\begin{pmatrix} \mathbf{C}_{11} & \mathbf{C}_{21} & \cdots & \mathbf{C}_{n1} \\ \mathbf{C}_{12} & \mathbf{C}_{22} & \cdots & \mathbf{C}_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_{1n} & \mathbf{C}_{2n} & \cdots & \mathbf{C}_{nn} \end{pmatrix}$$

so that

$$\left(\mathbf{A}^{-1}\right)_{ij} = \frac{1}{|\mathbf{A}|}\left(\mathbf{C}^{\mathrm{T}}\right)_{ij} = \frac{1}{|\mathbf{A}|}\left(\mathbf{C}_{ji}\right)$$

where |**A**| is the determinant of **A**, **C** is the matrix of cofactors, and $\mathbf{C}^{\mathrm{T}}$ represents the matrix transpose.

Inversion of 2 × 2 matrices

The *cofactor equation* listed above yields the following result for 2 × 2 matrices. Inversion of these matrices can be done as follows:

$$\mathbf{A}^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{\det \mathbf{A}} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

A computationally efficient 3 × 3 matrix inversion is given by

$$\mathbf{A}^{-1} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix}^{\mathrm{T}} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} A & D & G \\ B & E & H \\ C & F & I \end{bmatrix}$$

(where the scalar *A* is not to be confused with the matrix **A**).

If the determinant is non-zero, the matrix is invertible, with the elements of the intermediary matrix on the right side above given by

$$\begin{alignedat}{6} A &= &(ei - fh), &\quad D &= -&(bi - ch), &\quad G &= &(bf - ce), \\ B &= -&(di - fg), &\quad E &= &(ai - cg), &\quad H &= -&(af - cd), \\ C &= &(dh - eg), &\quad F &= -&(ah - bg), &\quad I &= &(ae - bd). \end{alignedat}$$

The determinant of **A** can be computed by applying the rule of Sarrus as follows:

$$\det(\mathbf{A}) = aA + bB + cC.$$

Linear System

- A linear system is a mathematical model of a system based on the use of a linear operator.
- A linear system exhibits features and properties that are much simpler than the nonlinear case.
- A linear system is a collection of one or more linear equations involving the same set of variables.
- As a mathematical abstraction or idealization, linear systems find important applications in automatic control theory, signal processing, and telecommunications, as well as in AI and machine learning.

System of linear equations

A general system of *m* linear equations with *n* unknowns can be written as

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m, \end{aligned}$$

where $x_1, x_2, \ldots, x_n$ are the unknowns, $a_{11}, a_{12}, \ldots, a_{mn}$ are the coefficients of the system, and $b_1, b_2, \ldots, b_m$ are the constant terms.

Often the coefficients and unknowns are real or complex numbers, but integers and rational numbers are also seen, as are polynomials and elements of an abstract algebraic structure.

Vector equation

One extremely helpful view is that each unknown is a weight for a column vector in a linear combination.

$$x_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix} + \cdots + x_n \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Matrix equation

The vector equation is equivalent to a matrix equation of the form

$$A\mathbf{x} = \mathbf{b}$$

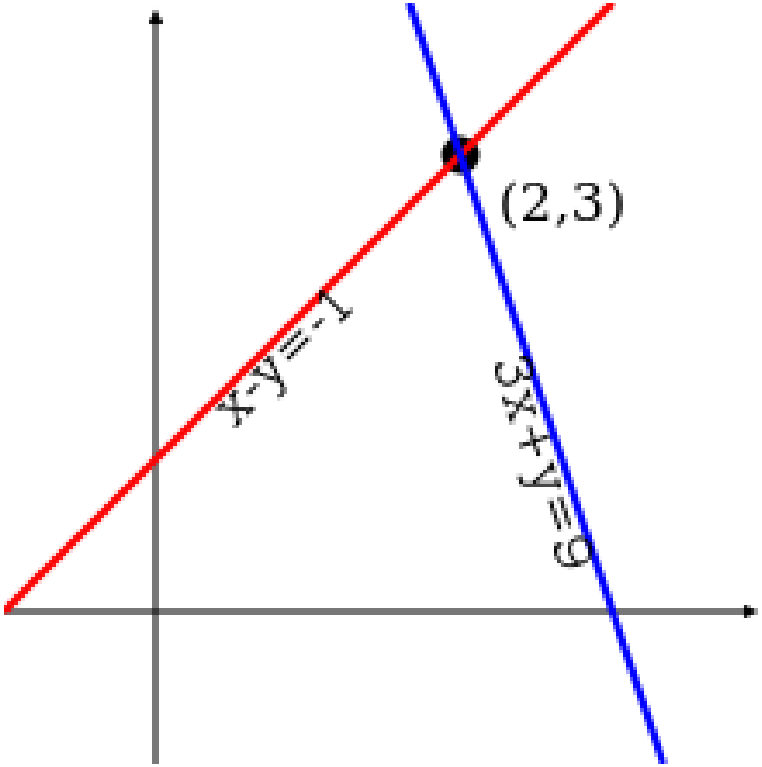
where A is an $m \times n$ matrix, $\mathbf{x}$ is a column vector with n entries, and $\mathbf{b}$ is a column vector with m entries.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

The number of vectors in a basis for the span is now expressed as the *rank* of the matrix.

A solution of a linear system

- A solution of a linear system is an assignment of values to the variables $x_1, x_2, \ldots, x_n$ such that each of the equations is satisfied.
- The set of all possible solutions is called the solution set.
- A linear system may behave in any one of three possible ways:
  1. The system has infinitely many solutions.
  2. The system has a single unique solution.
  3. The system has no solution.



Numpy usage

- Solve Linear equation: $\mathbf{A}x = \mathbf{b}$
- Calculate Matrix Inverse: $\mathbf{A}^{-1}$
- Calculate Matrix Determinant: $\det \mathbf{A} = |\mathbf{A}|$
- Random Numbers: Uniform, Gaussian

Applications

- Linear Regression
- Logistic Regression
- Artificial Neural Network
- Deep Learning
- Clustering
- Density Estimation
- Principal Components Analysis
- Matrix Factorization
- Support Vector Machine
- Hidden Markov Model
- Optimization

Numpy vs List

```
import numpy as np
names =['Somchai','Somying','Somjai','Somsak','Somsri']
height = [175., 160., 157., 182., 165.]
weight = [71., 46., 44., 80., 60.]
np_names = np.array(names)
np_height = np.array(height)
np_weight = np.array(weight)
np_persons = np.array([names, height, weight])
bmi = np_weight / (np_height/100) ** 2
overweight = bmi > 20
print(np_names[overweight])
```

```python
import numpy as np
L = [1, 2, 3]
A = np.array([1, 2, 3])
for e in L:
    print(e)
for e in A:
    print(e)
```

dtype

```
x = np.array([[1, 2, 3], [4, 5, 6]])

y = np.array([[1, 2, 3], [4, 5, 6]], np.int32)

z = x + y

x.dtype
y.dtype
z.dtype
```

Numpy Array Functions

- my_array.shape
- np.min(my_array)
- np.max(my_array)
- np.std(my_array)
- np.median(my_array)
- np.corrcoef(my_array)
- np.append(other_array)
- np.insert(my_array, 1, 5)
- np.delete(my_array, [1])