# Data Mining

CS57300
Purdue University

Jan 11, 2018

Bruno Ribeiro

# The Need for Formalism in Data Mining

- A woman is leading an environmental protest outside PMU today

- What is more likely?

  a) That she is an investment banker?

  b) That she is an investment banker studying Environmental Engineering at Purdue?

$$P[A] = \sum_{b \in B} P[A, b]$$

# Goals today

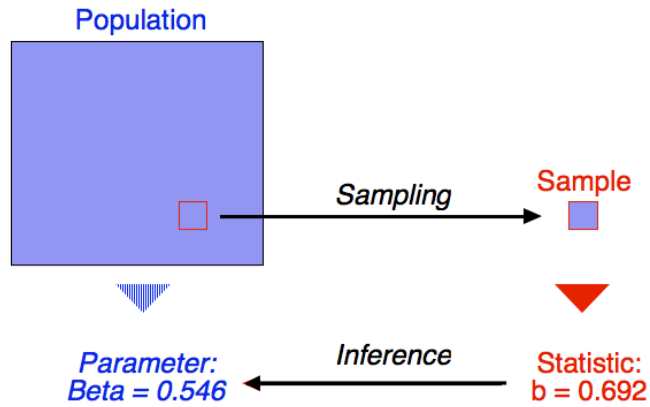
- Review some basic concepts
  - Statistical Inference
  - Linear Algebra

- Using Python in the Scholar cluster

# Probability and Statistical Estimation

# Populations and samples

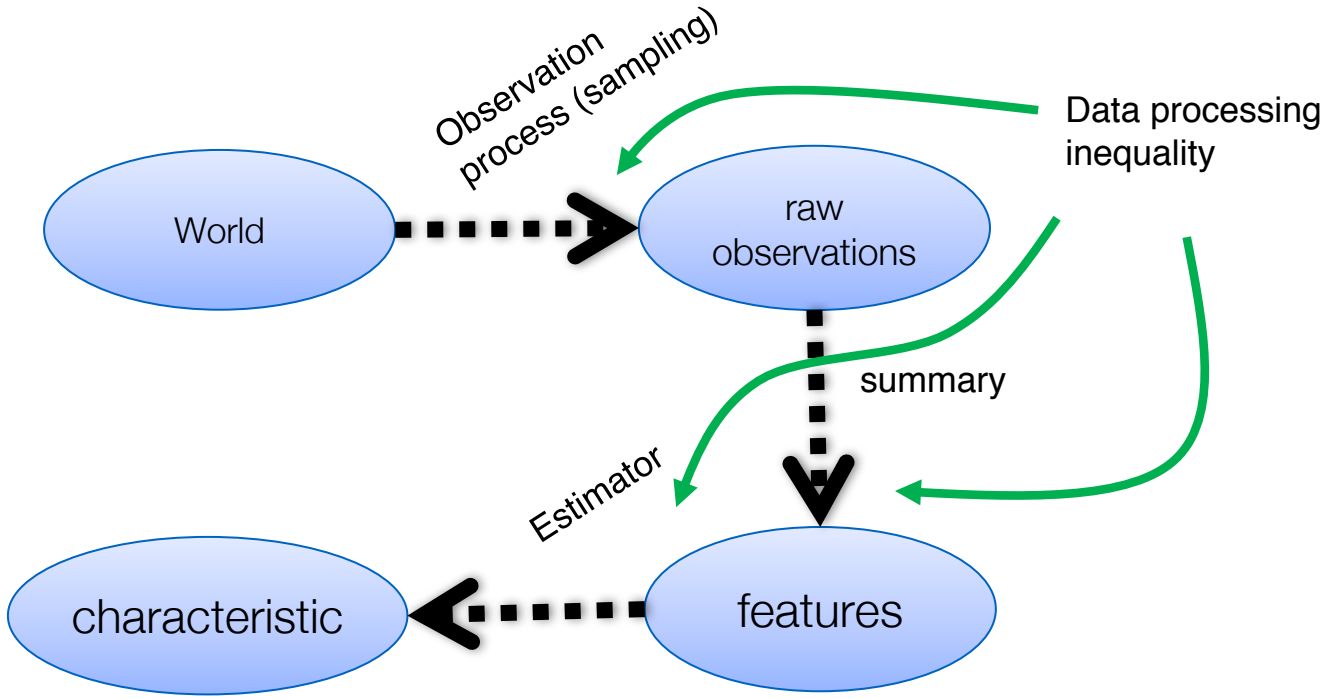
- In data mining we often work with a sample of data from the population of interest
- **Estimation** techniques allow inferences about population properties from sample data
- If we had the population we could **calculate** the properties of interest



# Estimating characteristics from sampling

## Data processing inequality:

*"No processing can increase the amount of statistical information already contained in the data"*



# Populations and samples

- Elementary units:
  - Entities (e.g., persons, objects, events) that meet a set of specified criteria
  - Example: All people who've purchased something at Wallmart
- Population:
  - Aggregate of elementary units (i.e, all items of interest)
- Sampling:
  - Sub-group of the population
  - Serves as a reference group for estimating characteristics about the population and drawing conclusions

# Sampling

- Sampling is the main technique employed for data selection
  - It is often used for both the preliminary investigation of the data and the final data analysis
- Reasons to sample
  - Obtaining the entire set of data of interest is too expensive or time consuming
  - Processing the entire set of data of interest is too expensive or time consuming
  - Note: Even if you use an entire dataset for analysis, you should be aware that most datasets are just samples of the entire population

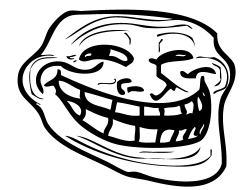
# Sampling …

- The key principle for effective sampling is the following:
  - Using a sample will work almost as well as using the entire dataset, if the sample is **representative** for the task you are interested
  - A sample is representative if it has approximately the same property (of interest) as the original data
  - Sample (observation) bias is a HUGE problem in data mining

# Warning: Careful with Sampling (Observation) Biases

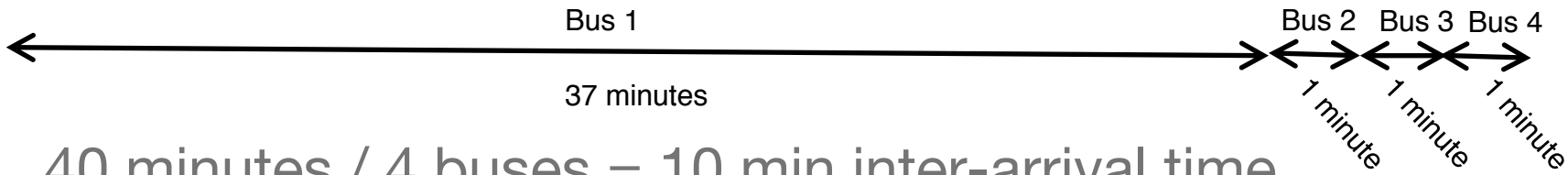
# Warning: Observation Biases are Prevalent


- Your experience with buses at peak hours:
  - Bus at 99% capacity at peak hours
  - You wait on average 17 minutes and 9 seconds for it to arrive
- Transportation admin:
  - *buses at peak hour are at 60% capacity*
  - *average bus inter-arrival time is 10 minutes*



?

# Inspection Paradox




- 40 minutes / 4 buses = 10 min inter-arrival time
- How long do you wait?
  - Assume you arrive uniformly during these 40 minutes
  - What is the probability you will arrive within the 37 minute interval?
    - P[Arrive at 37 min interval] = 37/40
    - What is the average waiting time if you arrive at the 37 min interval?
      - E[Wait | Arrive at 37 min interval] = 37/2 = 18.5

$$E[\text{Wait}] = \sum_i E[\text{Wait}|\text{Interval } i]P[\text{Interval } i] = \frac{37}{40} \times \frac{37}{2} + 3 \times \frac{1}{40} \times \frac{1}{2} = 17.15$$

End of Observation Bias Warning

# Statistical inference

- Infer properties of an unknown distribution with sample data generated from that distribution

- Parameter estimation
  - Infer the value of a population parameter based on a sample statistic (e.g., estimate the mean)

- Hypothesis testing
  - Infer the answer to a question about a population parameter based on a sample statistic (e.g., is the mean non-zero?)

# Common distributions

- Bernoulli
- Binomial
- Multinomial
- Poisson
- Normal

# Bernoulli

- Binary variable (0/1) that takes the value of 1 with probability p
  - E.g., Outcome of a fair coin toss is Bernoulli with *p=0.5*

$$P(x) = p^x(1-p)^{1-x}$$

$$E[X] = 1(p) + 0(1-p) = p$$

$$\begin{aligned} Var(X) &= E[X]^2 - (E[X])^2 \\ &= 1^2(p) + 0^2(1-p) - p^2 \\ &= p(1-p) \end{aligned}$$

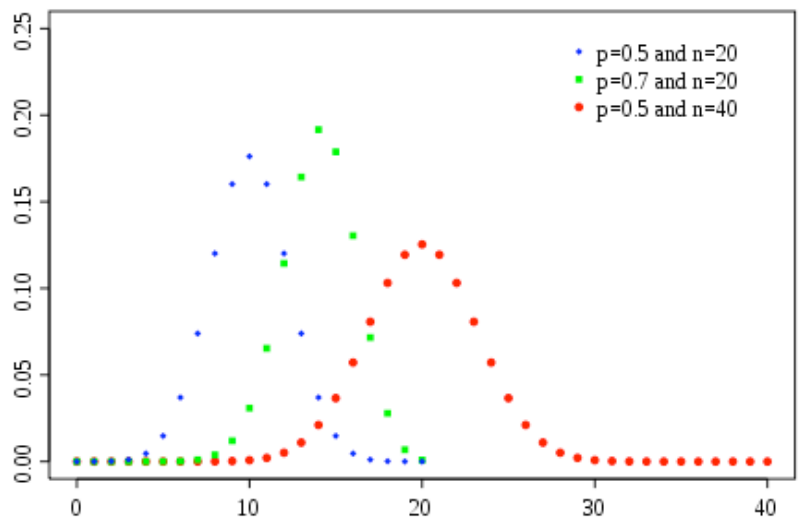
# Binomial

- Describes the number of successful outcomes in *n* independent Bernoulli(p) trials
  - E.g., Number of heads in a sequence of 10 tosses of a fair coin is Binomial with *n=10* and *p=0.5*

$$P(x) = \binom{n}{x} p^x (1-p)^{n-x}$$

$$E[X] = np$$

$$Var[X] = np(1-p)$$



# Multinomial

- Generalization of binomial to *k* possible outcomes; outcome *i* has probability $p_i$ of occurring
  - E.g., Number of {outs, singles, doubles, triples, homeruns} in a sequence of 10 times at bat is Multinomial
- Let $X_i$ denote the number of times the *i*-th outcome occurs in *n* trials:

$$P(x_1, ...x_k) = \binom{n}{x_1, ...x_k} p_1^{x_1} p_2^{x_2} ...p_k^{x_k}$$

$$E[X_i] = np_i$$

$$Var(X_i) = np_i(1 - p_i)$$

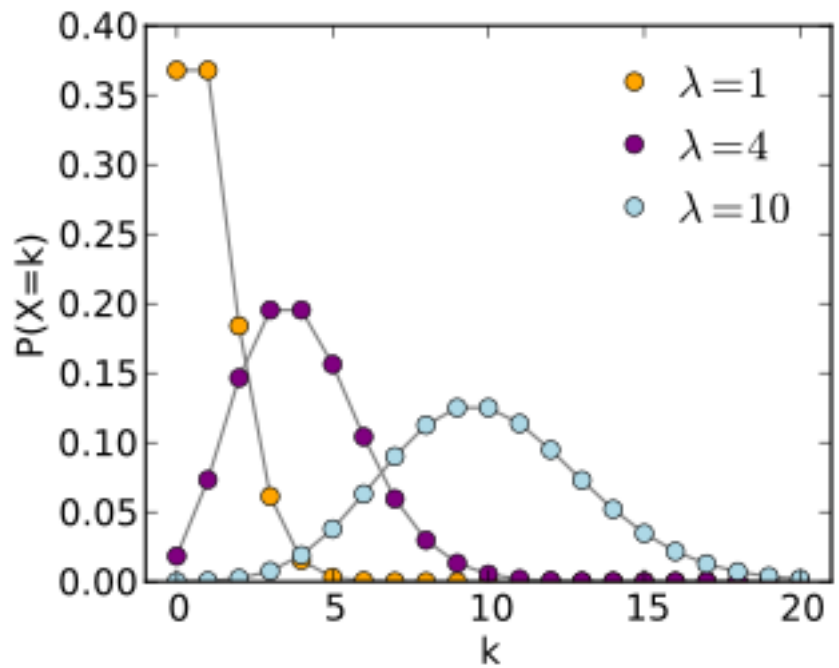
# Poisson

- Describes the number of successful outcomes occurring in a fixed interval of time (or space) if the “successes” occur *independently* with a known average rate
  - E.g., Number of emergency calls to a service center per hour, when the average rate per hour is $\lambda=10$

$$P(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

$$E[X] = \lambda$$

$$Var[X] = \lambda$$



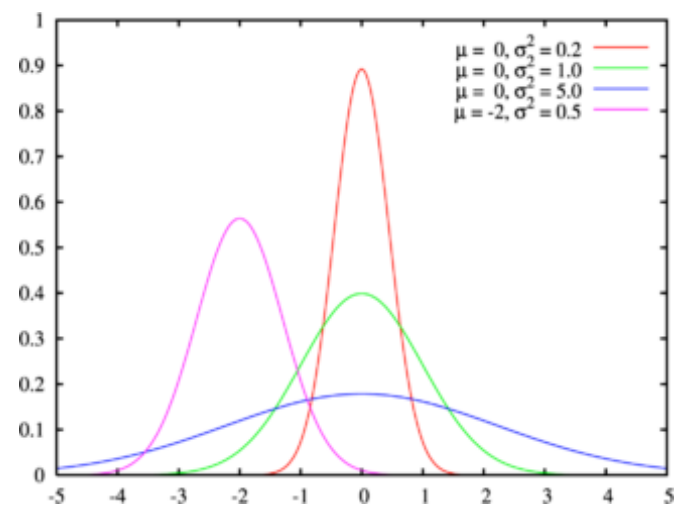
# Normal (Gaussian)

- Important distribution gives well-known bell shape
- Central limit theorem:
  - Distribution of the mean of n samples becomes normally distributed as n ↑, regardless of the distribution of the underlying population

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

$$E[X] = \mu$$

$$Var(X) = \sigma^2$$



# Multivariate RV

- A multivariate random variable ***X*** is a set $X_1$, $X_2$, ..., $X_p$ of random variables
- **Joint** density function: $P(\mathbf{x})=P(x_1, x_2, ..., x_p)$
- **Marginal** density function: the density of any subset of the complete set of variables, e.g.,:

$$P(x_1) = \sum_{x_2,x_3} p(x_1, x_2, x_3)$$

- **Conditional** density function: the density of a subset conditioned on particular values of the others, e.g.,:

$$P(x_1|x_2, x_3) = \frac{p(x_1, x_2, x_3)}{p(x_2, x_3)}$$

# Data Likelihood

- Consider a random variable X whose distribution has parameters **W**
- We will often use P(X | **W**) to denote the probability of X
- Consider $X_1$, …, $X_n$ random variables that are independent and identically distributed with distribution P(X | **W**)
- Dataset: The likelihood of observations $x_1$, …, $x_n$ is

$$P(X_1 = x_1, \ldots, X_n = x_n | \mathbf{W})$$
$$= \prod_{i=1}^{n} P(X_i = x_i | \mathbf{W})$$

# Parameter Estimation

- Maximum likelihood estimate (MLE)

$$\widehat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmax}}\, P[\text{Data}|\mathbf{W}]$$

- Maximum a posteriori probability estimate (MAP)

$$\widehat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmax}}\, P[\text{Data}|\mathbf{W}]P[\mathbf{W}]$$

# Parameter Estimation by Gradient Ascent

- Consider the j-th parameter of P(Data | **W**), $\mathbf{W}_j$
- We can maximize the likelihood by gradient ascent:

$$\mathbf{W}_{j,t+1} = \mathbf{W}_{j,t} + \eta \frac{\partial}{\partial \mathbf{W}_j} P(\text{Data}|\mathbf{W})$$

- The above is called (*full) batch* gradient ascent, as it uses the entire dataset.
  - For more complex models (or large datasets), we will often use only a subset of the data to compute an estimate of the gradient

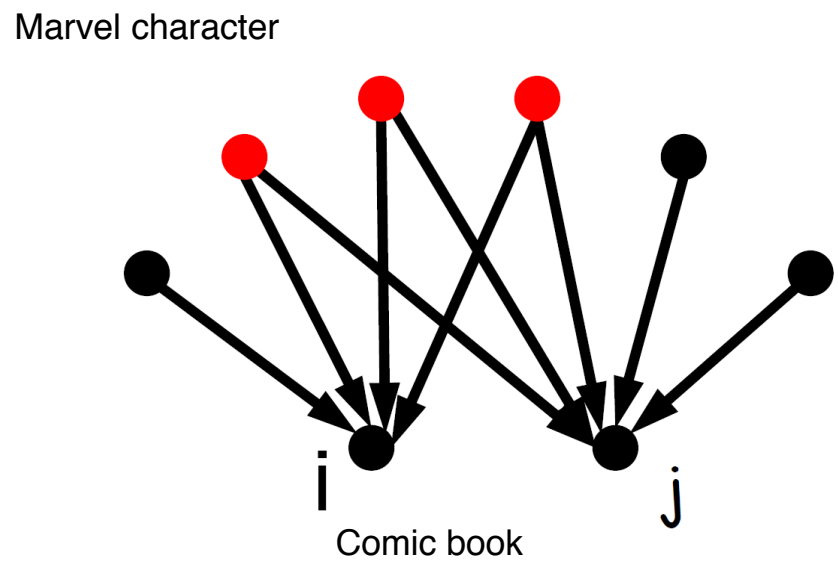
# Linear Algebra Review

# Why Linear Algebra?

- Why Algebra?
  - Computing is all about algebra
  - Way to mathematically describe data
- Why Linear?
  - Fast tools
  - Easy to understand
  - Many non-linear problems can be transformed or approximated as linear systems
- Combination works well in practice

# Example of Linear Algebra Application: Explain Relationships

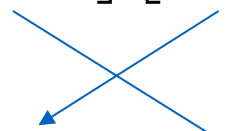
- Marvel character appears on comic book
- Described as an adjacency matrix (representing a graph)



# Important Properties

Matrix multiplication is *not commutative*

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$



$$\neq$$

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Trace: $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$

inner product $\langle x, y \rangle = x^{\mathsf{T}} y = \sum_{\forall i} x_i y_i$

2.1 $\langle x, x \rangle \geq 0$
2.2 $\|x\|^2 \equiv \langle x, x \rangle$
2.3 $\langle x, y \rangle = 0$ iff $x \perp y$
2.4 $\langle x, y \rangle = \|x\| \|y\| \cos\theta$, thus $y\langle x, y \rangle = \|x\| u$, where $u = y/\|y\|$

# Common Matrix Representations

Undirected graph:

$$A = A^{\mathsf{T}}$$

Bipartite graph: (undirected)

$$A = \begin{bmatrix} \mathbf{0} & B \\ B^{\mathsf{T}} & \mathbf{0} \end{bmatrix}$$

$k$ connected components:

$$A = \begin{bmatrix} A_1 & \cdots & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & A_k \end{bmatrix}$$

# Orthogonal vectors (linearly independent)

- Consider vectors $u_1, \ldots, u_k \in \mathbb{R}^k$

- Normalized if $u_i u_i^T = 1$ ,
  also defined as $\|u_i\| = 1, \quad i = 1, \ldots, k$

- Orthogonal if $u_i u_j^T = 0$ ,
  also defined as $u_i \perp u_j, \quad i \neq j$

- Orthonormal if both

  If $U$ is orthonormal then $UU^{-1} = UU^T = I$,
  where $I$ is the identity matrix

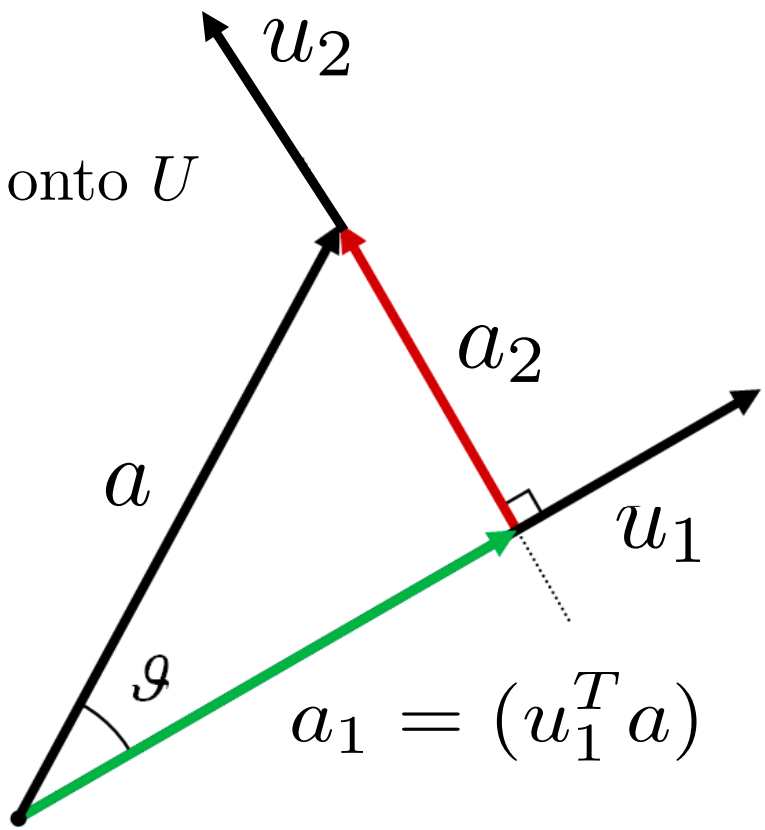
# Orthogonal Projections

Note that $I = UU^T = \sum_{i=1}^{k} u_i u_i^T$

Thus $a = Ia = U(U^T a), \quad a \in \mathbb{R}^k$

The vector $(U^T a)$ is the projection of $a$ onto $U$

Thus, $a = \sum_{i=1}^{k} (u_i^T a) u_i$



# Can we have non-orthogonal basis?

- Yes

- For instance:

  Let $U = [u_1, u_2]$ be an orthonormal basis and let

  $v_1 = 2u_1 + u_2$

  $v_2 = u_1 + 2u_2$

  The vectors $v_1$ and $v_1$ are not orthogonal:

  $v_1 v_2^T = 4\,,$

  but still form a basis for $\mathbb{R}^2$

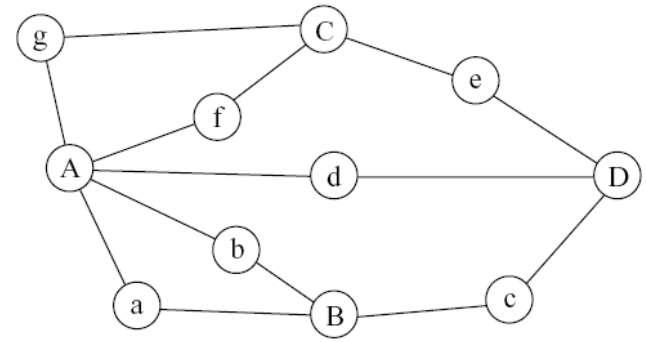
# Eigenvalues and Eigenvectors

$A > 0$ is $n \times n$ full rank, $\lambda$ eigenvalue

$$\det(A - \lambda I) = 0$$

and $x$ eigenvector (right eigenvector)

$$Ax = \lambda x,$$




we assume $\|x\| = 1$.

$$A\begin{bmatrix} x_1, & \cdots, & x_n \end{bmatrix} = \begin{bmatrix} x_1, & \cdots, & x_n \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix},$$

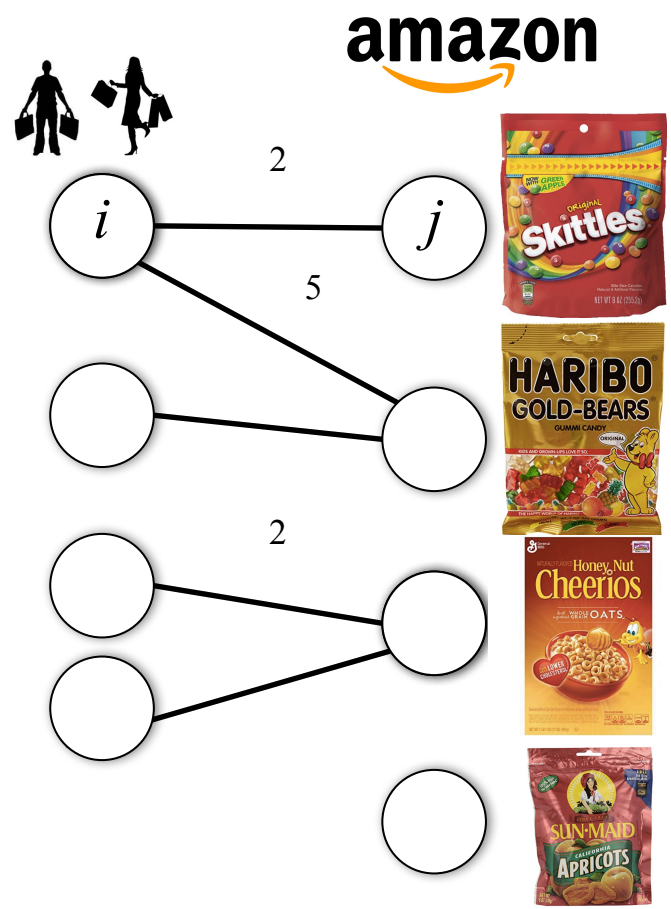
where $x_i$ is the $i$-th eigenvector of $A$ ordered s.t. $\lambda_1 \geq \cdots \geq \lambda_n$
If $A$ has $n$ linearly independent eigenvectors

$$A = V \Lambda V^{-1}$$

- If $A$ is symmetric positive semidefinite $V^{-1} = V^{T}$
- Square is $A^2 = V \Lambda^2 V^{-1}$
- Inverse is $A^{-1} = V \Lambda^{-1} V^{-1}$, where $\Lambda^{-1} = \begin{bmatrix} 1/\lambda_1 & & \\ & \ddots & \\ & & 1/\lambda_n \end{bmatrix}$

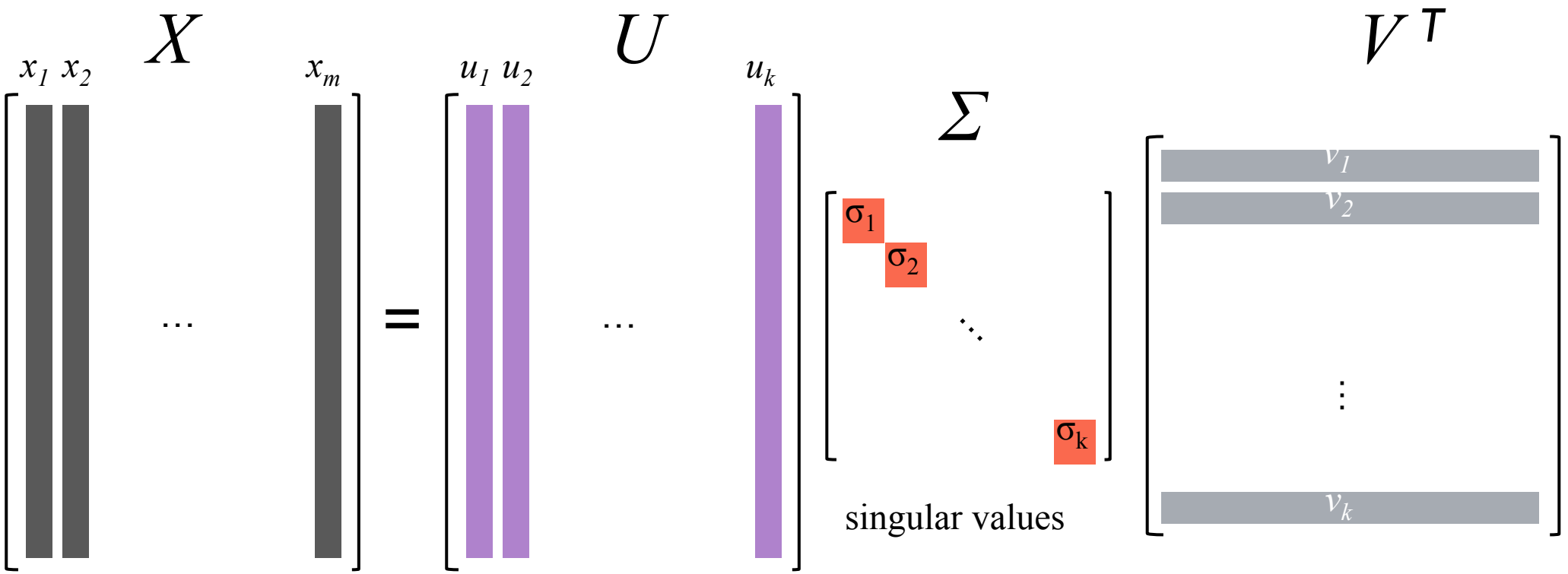
# Singular Value Decomposition (SVD)

- X(i,j) = value of user i for property j
  - X(Alice, cholesterol) = 10
  - X(i,j) = number of times i buys j
  - X(i,j) = how much i pays j
  - X(i,j) = 1 if i and j are friends, 0 otherwise
  - X(i,j) = temperature of sensor j at time i



# SVD Illustrated

$$X = U\Sigma V^T$$




Data

Left singular vectors

Columns of $U$ are orthonormal

Right singular vectors

Columns of $V$ are orthonormal

# SVD Properties (I)

$$X = U\Sigma V^T$$

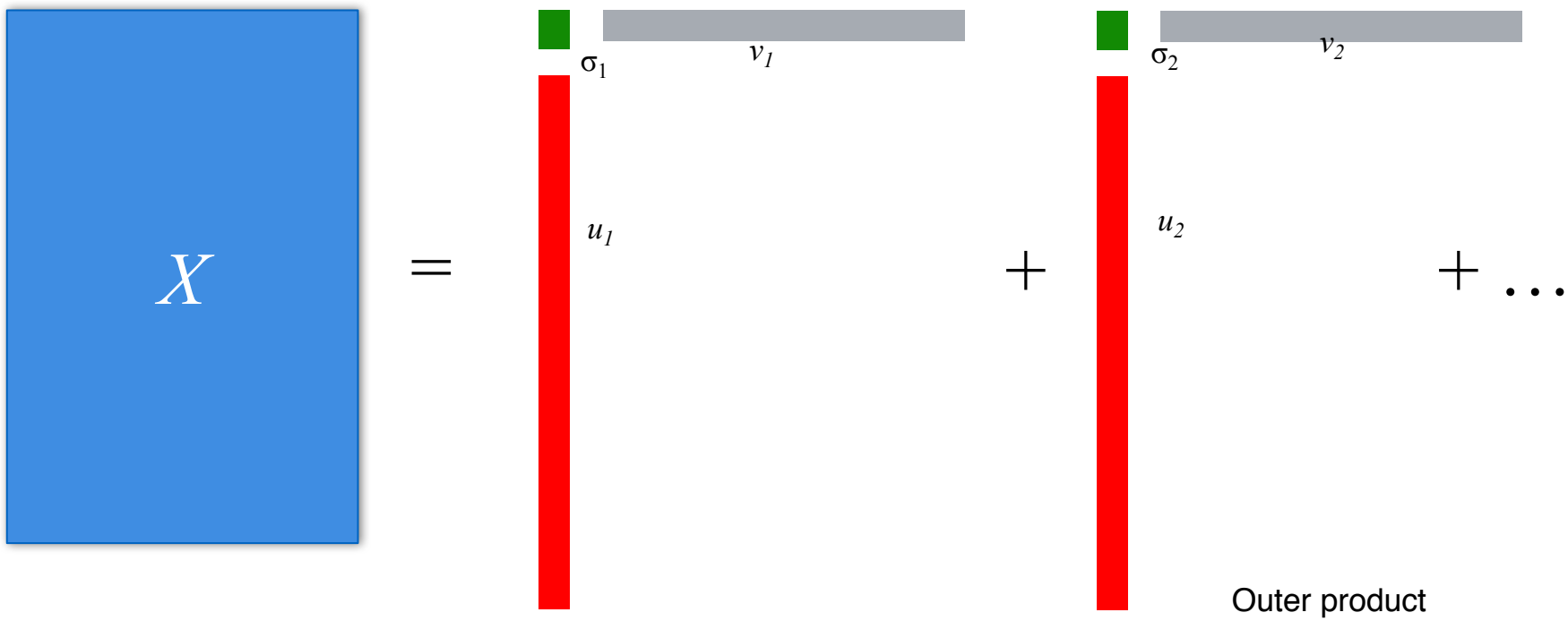
$$U^T U = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

$$V^T V = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

U and V are orthonormal
(orthogonal & unit norm)

# SVD Definition

- SVD gives best rank-k approximation of $X$ in $L_2$ and Frobenius norms

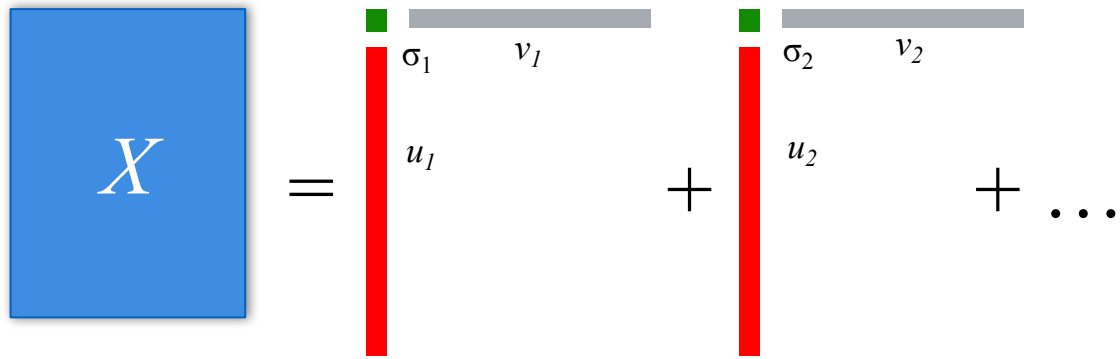



Outer product

$$X = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i \otimes v_i$$

# SVD Properties (II)

- "Almost unique" decomposition




- There are two sources of ambiguity
  - Orientation of singular vectors
    - Permute rows of left singular vector and corresponding rows of left singular vector
  - If I is identity matrix: $I = UIU^T$ , for all orthonormal U
    - "Hypersphere ambiguity"
    - Related to rotational ambiguity of PCA (we will see this later)

# SVD Properties (III)

- Theorem (Eckart-Young, 1936)
  - $U\Sigma_1 V^T$ is best rank 1 approximation of $X$, that is
    $|X - U\Sigma_1 V^T|^2 \leq |X - Y|^2$ for every rank 1 matrix $Y$
  - $U\Sigma_1 V^T + U\Sigma_2 V^T$ is the best rank 2 approximation of $X$, that is
    $|X - U\Sigma_1 V^T - U\Sigma_2 V^T|^2 \leq |X - Y|^2$
    for every rank ≤ 2 matrix $Y$
  - also for 3 , 4, …, r

# Singular Value Decomposition

- SVD factorization $A = U\Sigma V^\star$ is more general than eigenvalue / eigenvector factorization $A = V\Lambda V^{-1}$.

SVD also gives Moore-Penrose pseudoinverse (inverse of non-square matrices):

Moore–Penrose pseudoinverse is $A^+ = V\Sigma^+ U^T$

where $\Sigma^+$ is the reciprocal of each non-zero diagonal element of $\Sigma$.

# Python + Scholar Cluster

- Detailed tutorial:
  - https://www.cs.purdue.edu/homes/ribeirob/courses/Spring2018/notes/python/python-how-to.html
    - Links to Python resources
    - Detailed instructions on the use of the Scholar cluster

# Python example in the cluster

```
import numpy as np
from matplotlib import use
# Avoid using the xterminal to create plots (needed if plotting
#    on Scholar)
use("Agg")
import matplotlib.pyplot as plt
import scipy as sp
import math

p = 0.8
MAX_DEGREE = 101
ECCDF = 1.0
x = []
y = []
for d in xrange(MAX_DEGREE):
#Be careful with machine precision
    x.append(d)
    y.append(ECCDF)
    ECCDF = ECCDF - (1-p)*p**d
plt.xlim([1,max(x)])
plt.xlabel("node degree", fontsize=18)
plt.ylabel("ECCDF", fontsize=18)
plt.loglog(x,y,"ro")
plt.savefig('ECCDF_plot.pdf')
```

# Scholar Cluster

- High Performance Computing Cluster

```
                                Current Number of Cores
Queue                 Total     Queue      Run      Free      Max Walltime
===============     ========================================    ==============
debug                    32        32        0        32          0:30:00
scholar                  32        16       32         0        168:00:00
scholar-b                16         0        0        16        168:00:00
standby               9,024    39,737    1,366     1,956          4:00:00
standby-c               288       160        0       256          4:00:00
standby-g               192         0        0        71          4:00:00

-bash-4.1$
```

- Jobs must be submitted with qsub
  (do not use main terminal to run tasks or you will be banished)

- But you can also use your own computer
  (but Scholar learning curve pays-off later)

# Qsub Example File

```
#!/bin/bash -l
# Example submission file (myjob.sub)
# choose queue to use (e.g. standby or scholar-b)
#PBS -q standby
# FILENAME: myjob.sub
module load devel
module load gcc
module load anaconda/4.4.1-py35
module load r

cd $PBS_O_WORKDIR

unset DISPLAY



python matlibplot_example.py
```

# Scholar Cluster Job Submission

- qsub myjob.sub
- qstat -u <your Purdue username>

```
carter-adm.rcac.purdue.edu:
                                                                  Req'd    Req'd       Elap
Job ID                  Username    Queue    Jobname          SessID  NDS   TSK   Memory   Time    S   Time
----------------------- ----------- -------- ---------------- ------ ----- ------ ------ --------- - ---------
8389124.carter-adm.rca  ribeirob    standby  myjob.sub           --    --       1    --   00:30:00 Q       --
```

- qstat -u <your Purdue username>

```
carter-adm.rcac.purdue.edu:
                                                                  Req'd    Req'd       Elap
Job ID                  Username    Queue    Jobname          SessID  NDS   TSK   Memory   Time    S   Time
----------------------- ----------- -------- ---------------- ------ ----- ------ ------ --------- - ---------
8389124.carter-adm.rca  ribeirob    standby  myjob.sub           --    --       1    --   00:30:00 R  00:00:02
```

- After job finishes (submission directory has output files):
  - myjob.sub.o<jobID> (std output)
  - myjob.sub.e<jobID> (error)
  - ECCDF_plot.pdf (your plot)

# Unsupported: Jupyter Hub

- https://www.rcac.purdue.edu/compute/scholar/

- In the Jupyter Hub you will be unable to load some libraries (e.g., pytorch).

# Job Submission System is Not for Debugging

- Submission takes a while to run (few minutes)
  - How to iteratively debug code?
    1. qsub -I -q standby -l walltime=01:00:00
       # asks for 1 hour iterative terminal to debug problems in your code
       # remember to load the modules you need
    2. Or you can quickly run your code in the terminal to see if it is working.
- Further instructions at:
  https://www.cs.purdue.edu/homes/ribeirob/courses/Spring2018/notes/python/python-how-to.html
- Where to get further help
  https://www.rcac.purdue.edu/compute/scholar/guide/