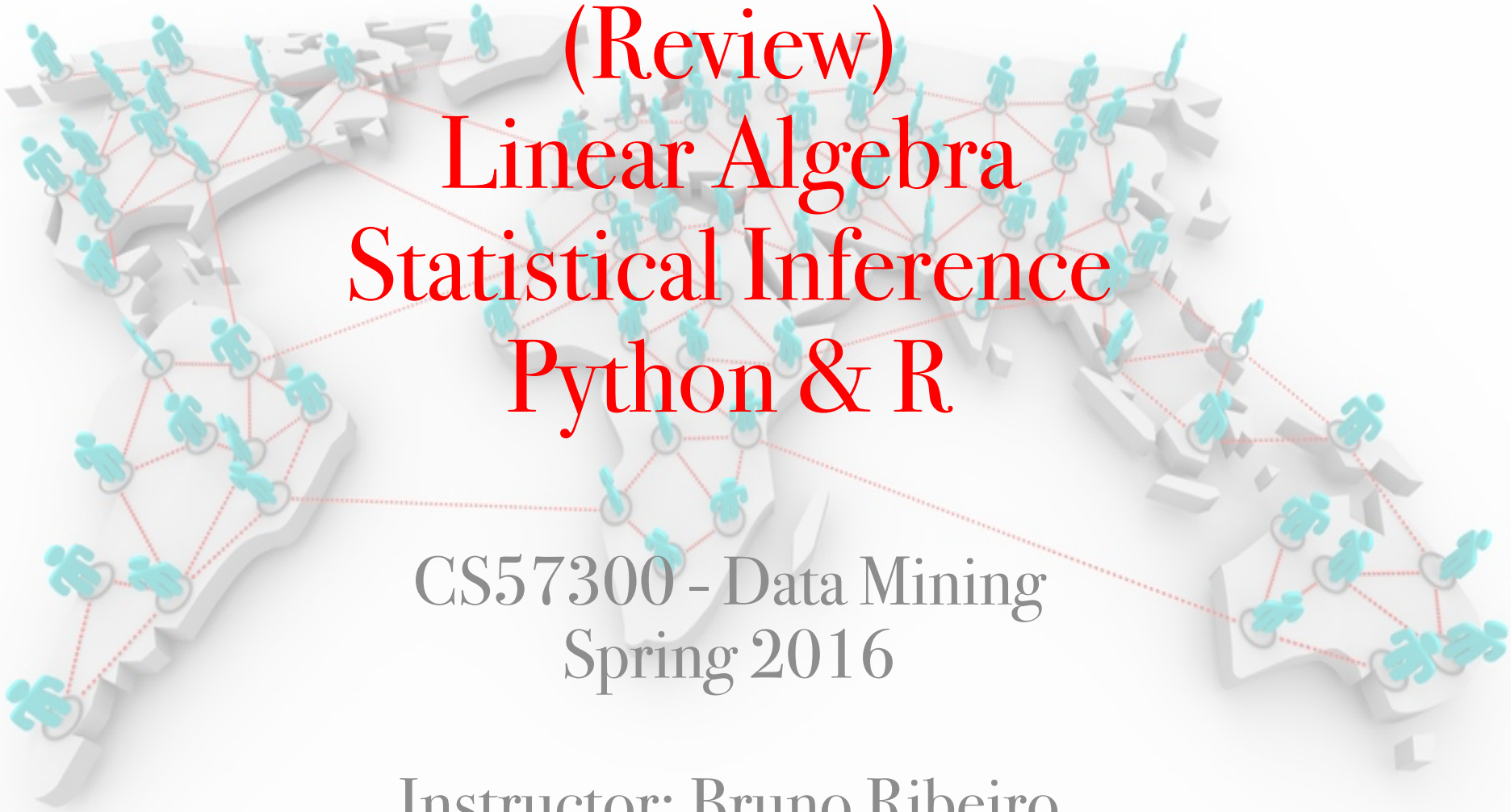


(Review) Linear Algebra Statistical Inference Python & R

CS57300 - Data Mining
Spring 2016

Instructor: Bruno Ribeiro

Goals today

- Review some basic concepts
  - Linear Algebra
  - Statistical Inference
- Introduce useful tools in Python and R
- Introduce the Scholar cluster

But before…

- 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?

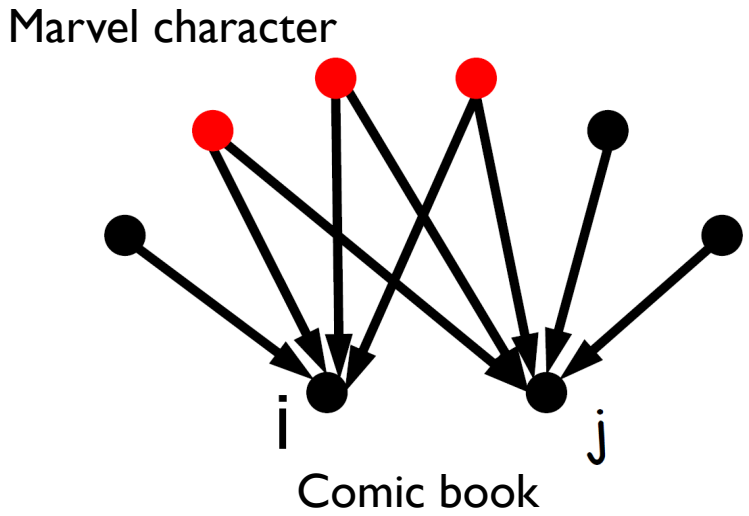
$$P[A] = \sum_{b \in B} P[A, b]$$

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

We will eventually use this to show
a drawback of
Principal Component Analysis (PCA)

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 $\mathbb{R}^2$

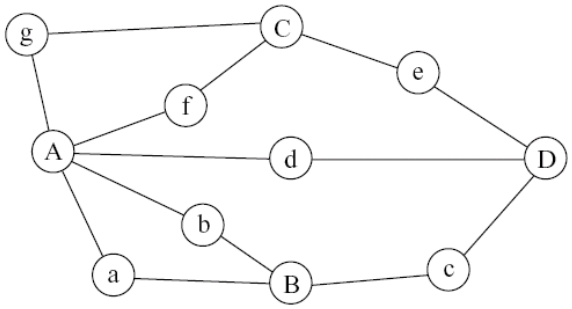
Eigenvalues and Eigenvectors

$A > 0$ is $n \times n$ full rank, λ 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

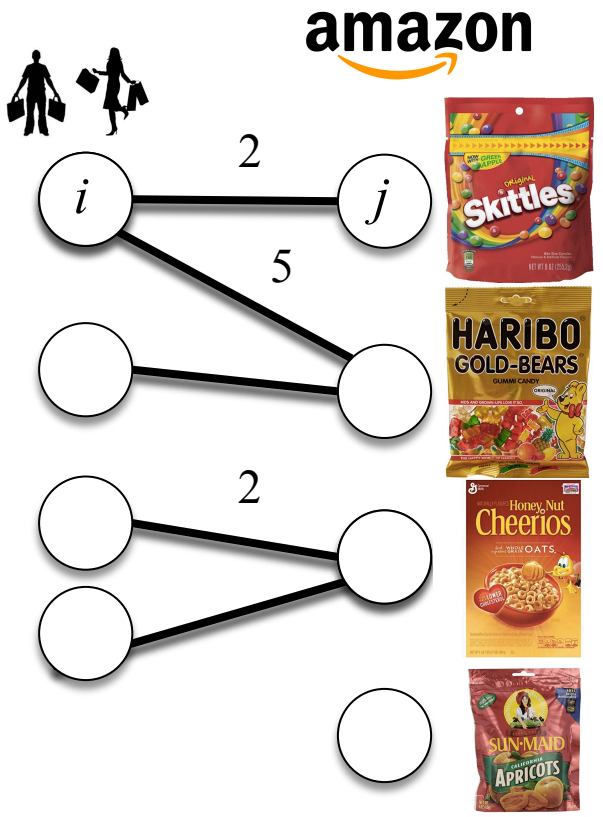
$$A = V \Lambda V^{-1}$$

- 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)

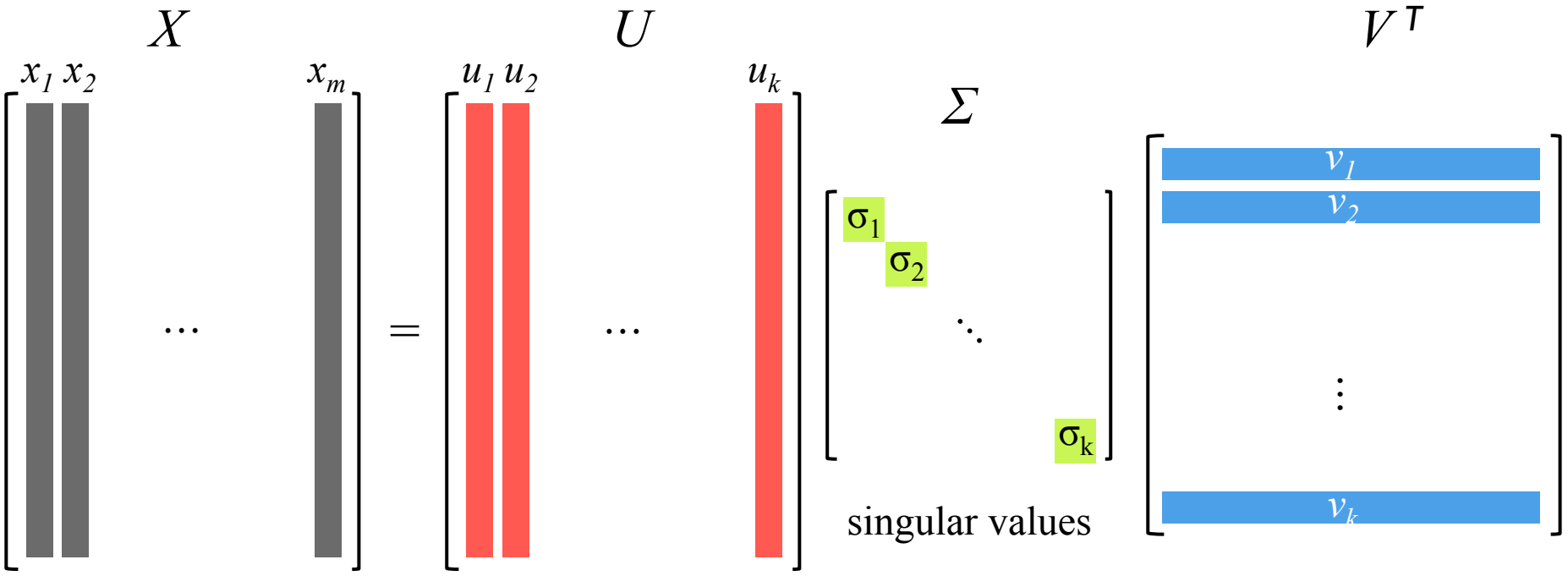
$$X = U\Sigma V^T$$

- 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



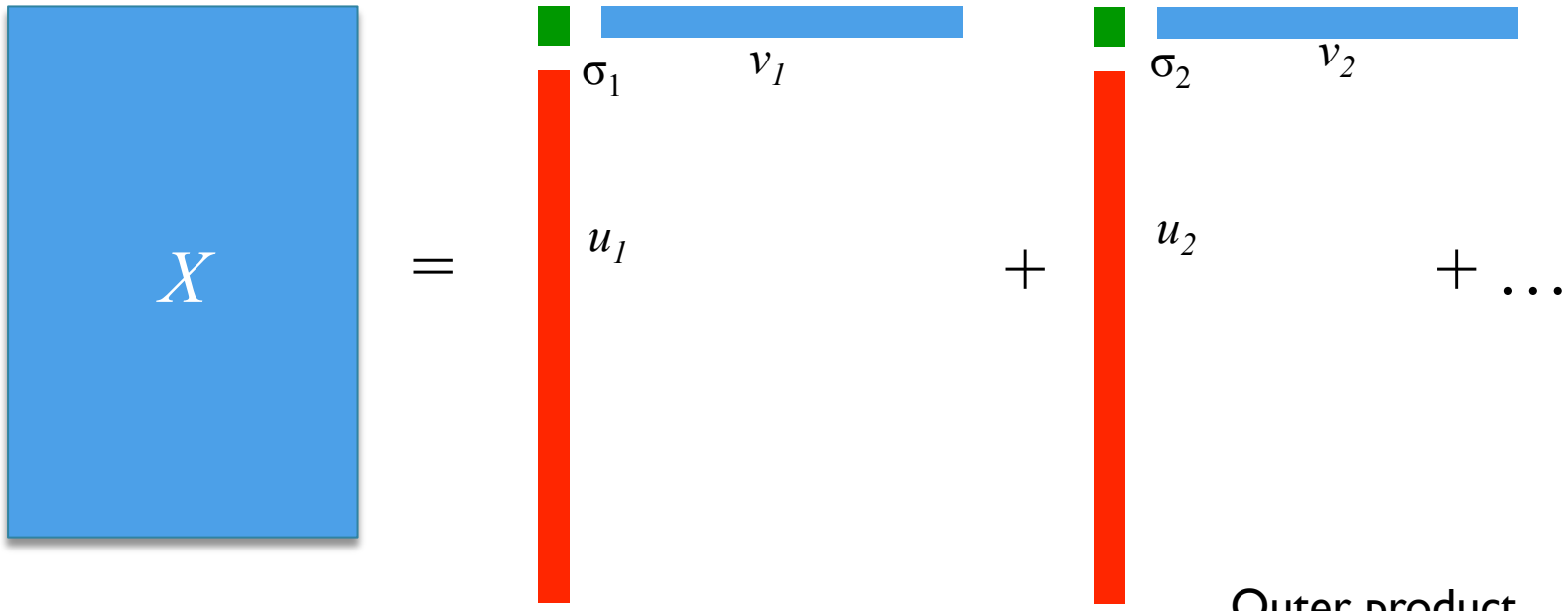
SVD Dimensions

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



SVD Definition

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

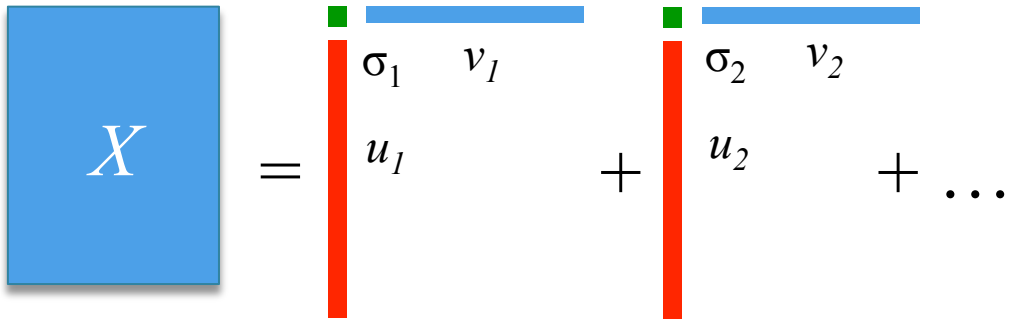


Outer product

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

SVD Properties (I)

- "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

SVD Properties (II)

- Theorem (Eckart-Young, 1936)
  - $\mathbf{U}\boldsymbol{\Sigma}_1\mathbf{V}^T$ is best rank 1 approximation of $\mathbf{X}$, that is
    $|\mathbf{X} - \mathbf{U}\boldsymbol{\Sigma}_1 \mathbf{V}^T|^2 \leq |\mathbf{X} - \mathbf{Y}|^2$ for every rank 1 matrix $\mathbf{Y}$
  - $\mathbf{U}\boldsymbol{\Sigma}_1\mathbf{V}^T + \mathbf{U}\boldsymbol{\Sigma}_2\mathbf{V}^T$ is the best rank 2 approximation of $\mathbf{X}$, that is
    $|\mathbf{X} - \mathbf{U}\boldsymbol{\Sigma}_1 \mathbf{V}^T - \mathbf{U}\boldsymbol{\Sigma}_2 \mathbf{V}^T|^2 \leq |\mathbf{X} - \mathbf{Y}|^2$
    for every rank ≤ 2 matrix $\mathbf{Y}$
  - also for 3 , 4, …, r

SVD Properties (III)

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

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

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

U and V are orthogonal

Singular Value Decomposition

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

$$\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix}$$

Columns of U are orthonormal

Columns of V are orthonormal

V* is the transpose if V is real-valued

SVD is significantly more generic:

- Applies to matrices of any shape, not just square matrices
- Applies to any matrix, not just invertible matrices
- $AA^T = (U\Sigma V^T)(U\Sigma V^T)^T = (U\Sigma V^T)(V\Sigma^T U^T)$
  $= U\Sigma\Sigma^T V^T = U\mathrm{diag}(\Sigma)^2 V^T$
- Moore–Penrose pseudoinverse is $A^+ = V\Sigma^+ U^T$

Understanding SVD singular vectors

- As U and V have orthogonal rows

$$X^T X = (U\Sigma V^T)^T (U\Sigma V^T) = V\Sigma^2 V^T$$

$$X X^T = (U\Sigma V^T)(U\Sigma V^T)^T = U\Sigma^2 U^T$$

Now you explain: What do V and U represent?

My "Help"

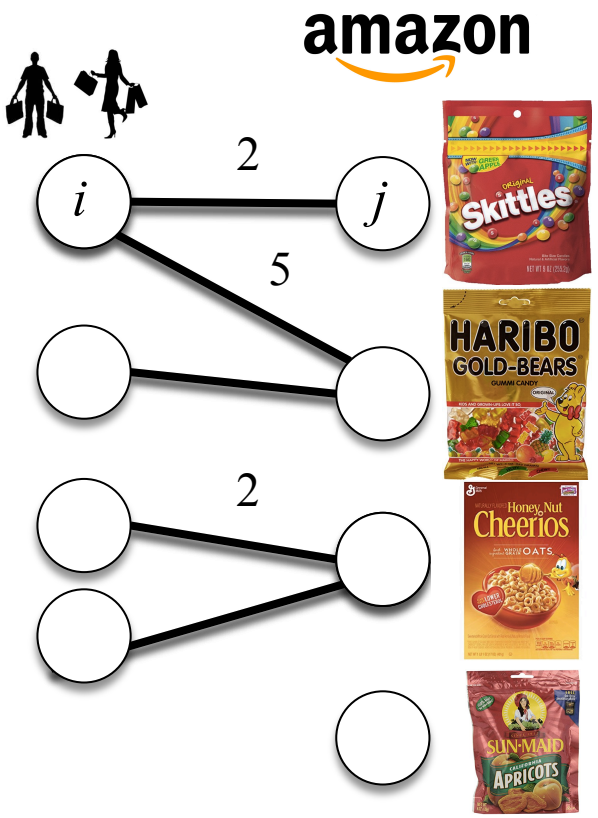
- If X(i,j) = user i buys product j

- What is X^TX?
  - Product-to-product similarity matrix
  - What does V represent?

$$X^T X = (U\Sigma V^T)^T (U\Sigma V^T) = V\Sigma^2 V^T$$

- What is XX^T?
  - User-to-user similarity matrix
  - What does U represent?

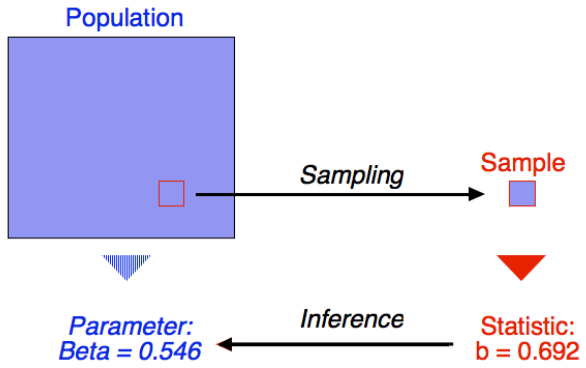
$$XX^T = (U\Sigma V^T)(U\Sigma V^T)^T = U\Sigma^2 U^T$$



Statistical Inference Review

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



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 of the sampling method that was used to gather the dataset

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
  - A sample is representative if it has approximately the same property (of interest) as the original data

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?)

Parameter Estimation

- Maximum likelihood estimate (MLE)

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, P[\text{Data}|\theta]$$
$$\text{s.t. } f(\theta) = 0$$

- Maximum a poseriori probability estimate (MAP)

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, P[\text{Data}|\theta]P[\theta]$$
$$\text{s.t. } f(\theta) = 0$$

Programming Languages

Python & R

Python example

```
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')
```

R example

```
# Read CSV into R
mydata = read.csv(file="input.csv", header=FALSE, sep=",")

#Generate a random matrix (elements exponentially distributed)
h = 100
k = 200
M = matrix(rexp(n=(h*k), rate=0.1),ncol=h,nrow=k)

print(max(M))
max_per_row = c()
for (i in 1:k) {
    max_per_row = c(max_per_row,max(M[i,]))
}
plot(max_per_row)
print(max(max_per_row)/min(max_per_row))
```

The Scholar Cluster

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)

- Last-resort if you can't install the necessary libraries on your computer

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 python
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)

Not ideal for someone learning Python and R

- Submission takes a while to run (few minutes)
  - How to iteratively debug code?
  - 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

- Scholar not supported by TAs or Instructor.

- Where to get help
  https://www.rcac.purdue.edu/compute/scholar/guide/