20211006 Singular Value Decomposition

Wednesday, October 6, 2021 4:12 PM

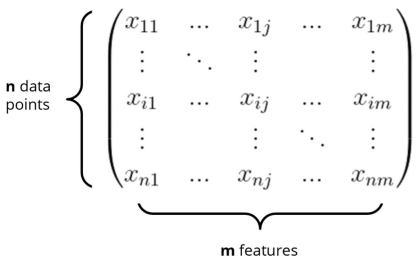
- HW1
  - A lot of back and forth about what packages you are allowed to use
    - Prof thought it was pretty clear
    - Some miscommunication has happened
      - Now some people used pandas for question 2 and now that is probably allowed
- Moving forward:
  - HWs are pretty much open to whatever packages you want
  - Otherwise, may put some piazza post specifying it
- You can implement all this stuff on your own
- Team lead role: check out piazza
- Posted monthly check-in poll
- HW grades are posted
- Midterm will have even more data, and will take even more time to code
  - Need to get used to running code overnight
  - Computation time needs to be factored into your planning
  - Kaggle competition, will take about 10 days
    - Start as soon as its posted

Singular Value Decomposition

Boston University CS 506 - Lance Galletti

- Another set of unsupervised techniques

Recall



Goal

Examine this matrix and uncover its linear algebraic properties to:

1. Approximate A with a smaller matrix B that is easier to store but contains similar information as A
2. Dimensionality Reduction / Feature Extraction
3. Anomaly Detection & Denoising

- Supposing everything is numeric in our dataset
- Goals are 3:
  1. Smaller matrix, so it takes less storage and all algorithms and improve the performance
     i. Maybe even denoise the dataset a little here
  2. Dimensionality reduction AND feature extraction
  3. anomaly detection and denoising

Linear Algebra Review

Definition: The vectors in a set $V = \{ \vec{v}_1, \ldots, \vec{v}_n \}$ are **linearly independent** if

$$a_1\vec{v}_1 + \ldots + a_n\vec{v}_n = \vec{o}$$

can only be satisfied by $a_i = 0$

Note: this means no vector in that set can be expressed as a **linear combination** of other vectors in the set.

- Turns out probability wasn't actually a prereq, but is linear algebra?
- Linearly independent:
  - Each element can't be represented by a linear combination of the other vectors in the set
  - The only special case is if all a is 0, but only allowed case

Linear Algebra Review

Definition:

The **determinant** of a square matrix A is a scalar value that encodes properties about the **linear mapping** described by A.

2x2:

$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ det(A) = ad - bc

---

- Determinant
  - Expresses so much about a matrix

Linear Algebra Review

Definition:

The **determinant** of a square matrix A is a scalar value that encodes

Definition:

The **determinant** of a square matrix A is a scalar value that encodes properties about the **linear mapping** described by A.

3x3:

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \quad det(A) = a \cdot det\begin{pmatrix} e & f \\ h & i \end{pmatrix} - b \cdot det\begin{pmatrix} d & f \\ g & i \end{pmatrix} + c \cdot det\begin{pmatrix} d & e \\ g & h \end{pmatrix}$$

- topic of this lab
- Do you see how to extend this to the nxn case?

Linear Algebra Review

Definition:

The **determinant** of a square matrix A is a scalar value that encodes properties about the **linear mapping** described by A.

n X n:

Can recursively compute it. How?

Linear Algebra Review

Property:

n vectors **{ $\vec{v}_1$, ..., $\vec{v}_n$ }** in an n-dimensional space are **linearly independent** iff the matrix **A**:

A = [$\vec{v}_1$, ..., $\vec{v}_n$] (n x n)

has non-zero determinant.

Q: Can **m > n** vectors in an **n**-dimensional space be linearly independent?

- No

Linear Algebra Review

Definition:

A **basis B** of a vector space (over a field **F**) is a **linearly independent** subset of **V** that **spans V**. **B spans V** if for every vector **v** in **V** it is possible to choose $\mathbf{v}_1$, ..., $\mathbf{v}_n$ in **F** and $\vec{\mathbf{b}}_1$, ..., $\vec{\mathbf{b}}_n$ in **B** such that:

$$\mathbf{v} = \mathbf{v}_1\vec{\mathbf{b}}_1 + \ldots + \mathbf{v}_n\vec{\mathbf{b}}_n$$

Ex: North & East in 2d-plane

Linear Algebra Review

Definition:

The **rank** of a matrix **A** is the dimension of the vector space spanned by its column space. This is equivalent to the maximal number of linearly independent columns / rows of **A**.

Definition:

A matrix **A** is **full-rank** iff **rank(A) = min(m, n)**

Note: Get the rank of a matrix through the **Gram-Schmidt process**

- We normally talk about columns not rows
- Data is normally far from full rank?
  - No, generally pretty close to full rank
    - Depends on the size of the matrix
- All columns could be linearly independent and it would still be full rank
- Gram-Smidt process
  - This is how you get the rank of the matrix
  - Take one out a time, eventually you run out of vectors

Approximation

In practice, matrices describing our dataset contain a lot of redundant information.

It would be great to capture all the information of our dataset in the least amount of space possible.

Approximation

To store an **n x m** matrix **A** requires storing **m · n** values.

However, if the rank of the matrix of **A** is **k**, **A** can be factored as

$$A = UV$$

where

U is **n x k**
V is **k x m**

which requires storing **k(m + n)** values.

- You can factor a matrix of rank k into two matrices
  - If k is small, how many values do you need to store?
  - n*k + k*m might be <<< n*m

Approximation

Goal:

Approximate **A** with $\mathbf{A^{(k)}}$ (low-rank matrix) such that

1. $\mathbf{d(A, A^{(k)})}$ is small
2. **k** is small compared to **m** & **n**

Frobenius Distance

$$d_F(A, B) = \|A - B\|_F = \sqrt{\sum_{i,j}(a_{ij} - b_{ij})^2}$$

i.e. the pairwise differences in values of A and B

Approximation

Definition:

When **k < rank(A)**, the **rank-k approximation** of **A** (in the least squares sense) is

$$A^{(k)} = \underset{\{B|rank(B)=k\}}{\arg\min} \; d_F(A, B)$$

- Has the same dimensions, but with lower rank so we can factor things out and store as smaller matrices
- Infinite number of matrices with rank k, and all sorts of possible numbers in that matrix

Approximation

Definition:

The **Singular Value Decomposition** of a rank-r matrix A has the form

$$\mathbf{A = U\Sigma V^T}$$

where

U is **n x r**
The columns of **U** are orthogonal & unit length ($\mathbf{U^TU = I}$)

V is **m x r**
The columns of **V** are orthogonal & unit length ($\mathbf{V^TV = I}$)

- V transpose is r x m

Approximation

Definition:

The **Singular Value Decomposition** of a rank-r matrix A has the form

$$\mathbf{A = U\Sigma V^T}$$

where

Approximation

Definition:

The **Singular Value Decomposition** of a rank-r matrix A has the form

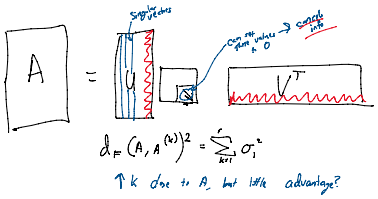
$$A = U\Sigma V^T$$

where

$$\Sigma = \begin{pmatrix} \sigma_1 & & 0 \\ & \sigma_2 & \\ & & \ddots \\ 0 & & \sigma_r \end{pmatrix}$$

with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$

σ_i is the square root of the eigenvalues of A^TA and are called **singular values**



- Diagonal matrix
- Singular values also have a particular size order

Approximation

Property:

$$d_F(A, A^{(k)})^2 = \sum_{i=k+1}^{r} \sigma_i^2$$

Note: the larger **k** is, the smaller the distance.

-

Approximation

Find $\mathbf{A}^{(k)}$ by decomposing $\mathbf{A}$:

$$A = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \Sigma_1 & \\ & \Sigma_2 \end{pmatrix} \begin{pmatrix} V_1 & V_2 \end{pmatrix}$$

$\mathbf{A}^{(k)} = \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^T$

Where

$\mathbf{U}_1$ is **n x k**
$\mathbf{\Sigma}_1$ is **k x k**
$\mathbf{V}_1$ is **m x k**

Approximation

U

1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	2	2
0	0	0	3	3
0	0	0	1	1

=

0.18	0
0.36	0
0.18	0
0.90	0
0	0.53
0	0.80
0	0.27

x

Σ

9.64	0
0	5.29

x

V^T

0.58	0.58	0.58	0	0
0	0	0	0.71	0.71

Rank 2

- Rank: of the dataset 2

Approximation

1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	2	2
0	0	0	3	3
0	0	0	1	1

~

0.18	0
0.36	0
0.18	0
0.90	0
0	0.53
0	0.80
0	0.27

x

9.64	0
0	0

x

0.58	0.58	0.58	0	0
0	0	0	0.71	0.71

Approximation

1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	2	2
0	0	0	3	3
0	0	0	1	1

~

0.18	0
0.36	0
0.18	0
0.90	0
0	0.53
0	0.80
0	0.27

x

9.64	0
0	0

x

0	0	0	0.71	0.71

- Rank two means we can represent the dataset with two features
- What these features mean and represent is not clear

Approximation

1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	2	2
0	0	0	3	3
0	0	0	1	1

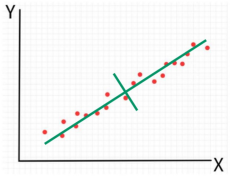
≈

1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Demo

Approximation

The **i^{th} singular vector** represents the direction of the i^{th} most variance.



Singular Values express the importance / significance of a singular vector

- the i-th singular vector (the i-th column)
  - Is in the direction of the i-th most variance
- x = y is the first singular vector, with the most variance
- second most variance, is in the perpendicular
- So we basically want to change the coordinate system used
- Matrix U is orthonormal, all columns are perpendicular to each other

Approximation

To find the right **k** you can:

1. Look at the singular value plot to find the elbow point
2. Look at the residual error of choosing different **k**

Demo

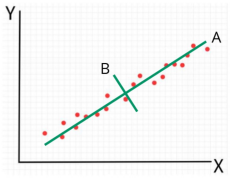
Principal Component Analysis

Idea: project the data onto a subspace generated from a subset of singular vectors / principal components.

We want to project onto the components that capture most of the variance / information in the data.

- The reasoning: we would like to project our data in a way to capture the most variance

Principal Component Analysis



Which principal component should we project on?

- Project data onto A

Demo

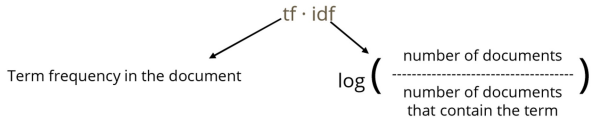
Are there datasets this will perform particularly poorly on?

(Demos all at end of notes)

Latent Semantic Analysis

Inputs are documents. Each word is a feature. We can represent each document by:

- The presence of the word (0 / 1)
- Count of the word (0, 1, ...)
- Frequency of the word (n_i / Σn_i)
- TfiDf



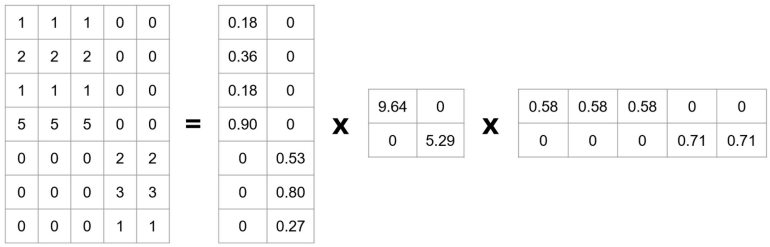
- Presence of word 0 or 1
  - Does not capture how much the word is used
  - "This document is not science" will use the word science once, even though the document is not about science
- Even better than count an frequency: log inverse document frequency * term frequency
  - Tells you frequently used in the document, factoring in how rarely used elsewhere
    - "the" is not representative of a unique topic if everyone is using it

Latent Semantic Analysis

	data	information	retrival	brain	lung
CS-paper-1	1	1	1	0	0
CS-paper-2	2	2	2	0	0
CS-paper-3	1	1	1	0	0
CS-paper-4	5	5	5	0	0
Med-paper-1	0	0	0	2	2
Med-paper-2	0	0	0	3	3
Med-paper-3	0	0	0	1	1

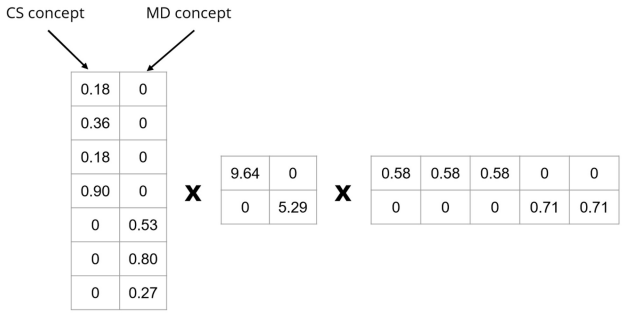
- CS papers vs medical papers
- Two linearly independent columns that represent our dataset
- Imagine you don't have access to which are CS and which are med papers:
  - Want to identify how many topics are in this dataset

Latent Semantic Analysis



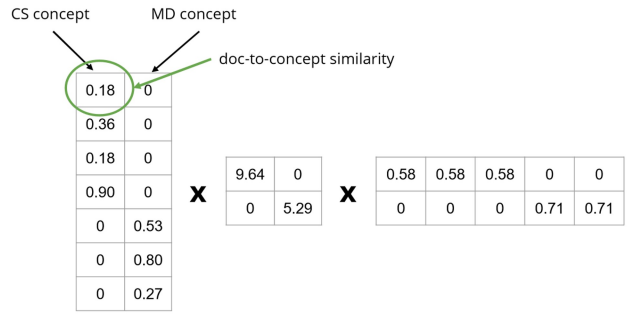
- Now we can associate meanings to each of these matrices

Latent Semantic Analysis



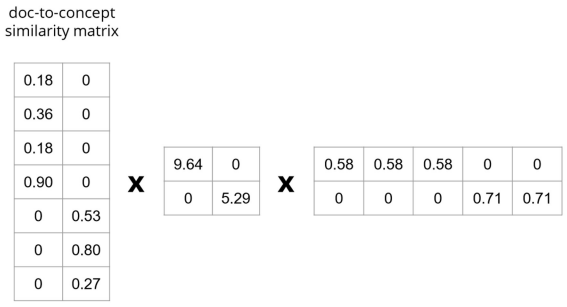
- 1st column: learned cs concept
- 2nd: MD concept

Latent Semantic Analysis

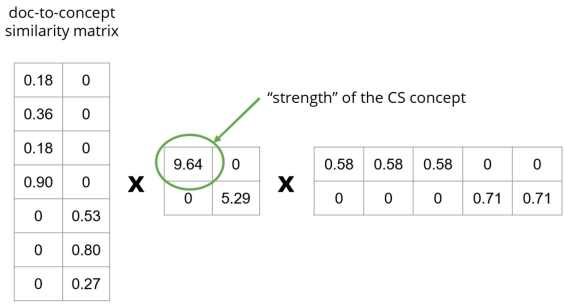


- How similar is a given document to a given concept?
  - One interpretation

Latent Semantic Analysis

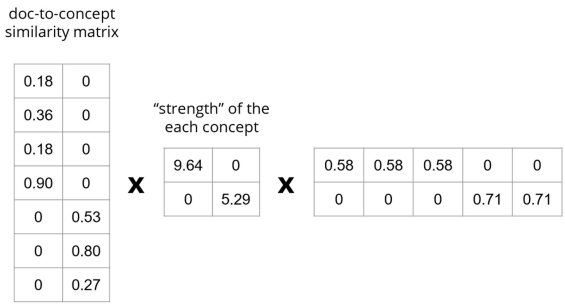


Latent Semantic Analysis

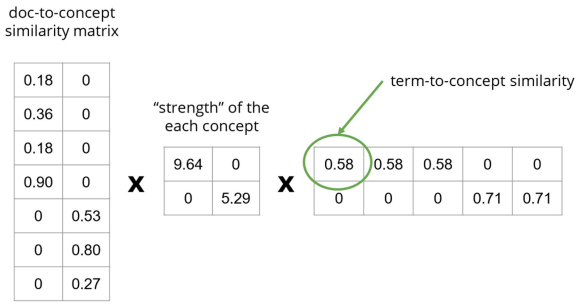


- Strength of the concept

Latent Semantic Analysis

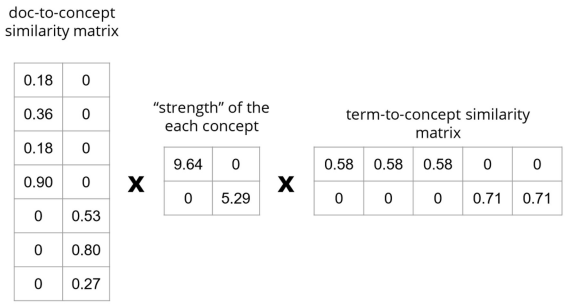


Latent Semantic Analysis



- How similar is a particular word to the concept
- How useful is a particular word to the concept

Latent Semantic Analysis



Demo

Anomaly Detection

Define $\mathbf{O} = \mathbf{A} - \mathbf{A}^{(k)}$

The largest rows of **O** could be considered anomalies

- To do anomaly detection, you need to ?

Demo

lsa.py is one of most important files for the midterm!!!!

File - C:\Users\Wolfs\PycharmProjects\cs506lec20211006\lsa.py

```
import numpy as np
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import
TfidfVectorizer
from nltk.stem.snowball import SnowballStemmer
from nltk.tokenize import word_tokenize, sent_tokenize

# THIS IS PROBABLY ONE OF THE MOST IMPORTANT FILES FOR
 THE MIDTERM
# First preprocess your data. You should always do
this
# Reducing prefixes and suffixes to try to condenses
it farther
# Limitation: how you are handling stripping on prefix
/suffix and ignoring context
# Transform, center data, plotted singular values
# Already know there are 3 concepts, so want to do
kmeans with 3 clusters to see how well you can cluster
 based on
# this

categories = ['comp.os.ms-windows.misc', 'sci.space','
rec.sport.baseball']
news_data = fetch_20newsgroups(subset='train',
categories=categories)
vectorizer = TfidfVectorizer(stop_words='english',
min_df=4,max_df=0.8)

stemmed_data = [" ".join(SnowballStemmer("english",
ignore_stopwords=True).stem(word)
         for sent in sent_tokenize(message)
        for word in word_tokenize(sent))
        for message in news_data.data]

dtm = vectorizer.fit_transform(stemmed_data)
terms = vectorizer.get_feature_names()
centered_dtm = dtm - np.mean(dtm, axis=0)

```

File - C:\Users\Wolfs\PycharmProjects\cs506lec20211006\lsa.py

```
31 u, s, vt = np.linalg.svd(centered_dtm)
32 plt.xlim([0,50])
33 plt.plot(range(1,len(s)+1),s)
34 plt.show()
35
36 ag = []
37 max = len(u)
38 for k in range(1,25):
39     vectorsk = u[:,:k] @ np.diag(s[:k])
40     kmeans = KMeans(n_clusters=3, init='k-means++',
   max_iter=100, n_init=10, random_state=0)
41     kmeans.fit_predict(vectorsk)
42     labelsk = kmeans.labels_
43     ag.append(metrics.v_measure_score(labelsk,
   news_data.target)) # closer to 1 means better
   clustering
44
45 plt.plot(range(1,25),ag)
46 plt.ylabel('Agreement',size=20)
47 plt.xlabel('No of Prin Comps',size=20)
48 plt.show()
49
50
```



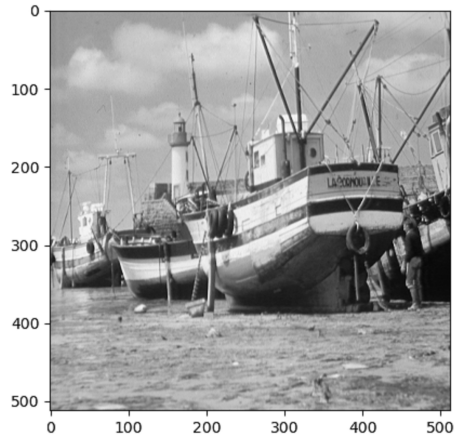
```
Traceback (most recent call last):
  File "C:\Users\Wolfs\PycharmProjects\cs506lec20211006\lsa.py", line 20, in <module>
    stemmed_data = [" ".join(SnowballStemmer("english", ignore_stopwords=True).stem(word)
  File "C:\Users\Wolfs\PycharmProjects\cs506lec20211006\lsa.py", line 21, in <listcomp>
    for sent in sent_tokenize(message)
  File "C:\Users\Wolfs\anaconda3\envs\cs506lec20211006\lib\site-packages\nltk\tokenize\__init__.py", line 106, in sent_tokenize
    tokenizer = load(f"tokenizers/punkt/{language}.pickle")
  File "C:\Users\Wolfs\anaconda3\envs\cs506lec20211006\lib\site-packages\nltk\data.py", line 750, in load
    opened_resource = _open(resource_url)
  File "C:\Users\Wolfs\anaconda3\envs\cs506lec20211006\lib\site-packages\nltk\data.py", line 876, in _open
    return find(path_, path + [""]).open()
  File "C:\Users\Wolfs\anaconda3\envs\cs506lec20211006\lib\site-packages\nltk\data.py", line 583, in find
    raise LookupError(resource_not_found)
LookupError:
**********************************************************************
  Resource punkt not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('punkt')

  For more information see: https://www.nltk.org/data.html

  Attempted to load tokenizers/punkt/english.pickle

  Searched in:
    - 'C:\\Users\\Wolfs/nltk_data'
    - 'C:\\Users\\Wolfs\\anaconda3\\envs\\cs506lec20211006\\nltk_data'
    - 'C:\\Users\\Wolfs\\anaconda3\\envs\\cs506lec20211006\\share\\nltk_data'
    - 'C:\\Users\\Wolfs\\anaconda3\\envs\\cs506lec20211006\\lib\\nltk_data'
    - 'C:\\Users\\Wolfs\\AppData\\Roaming\\nltk_data'
    - 'C:\\nltk_data'
    - 'D:\\nltk_data'
    - 'E:\\nltk_data'
    - ''
**********************************************************************
```

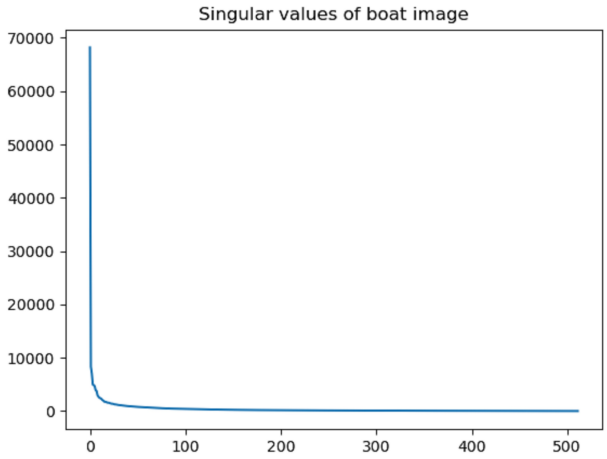
File - C:\Users\Wolfs\PycharmProjects\cs506lec20211006\svd-on-image.py

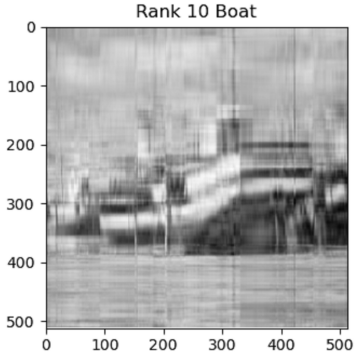
```python
import numpy as np
import matplotlib.cm as cm
import matplotlib.pyplot as plt

# Compression
# Image on screen is rarely what you see on screen

# How big the singular values are represents how important that feature is to the dataset
# you can see we have very high singular values before it shoots down, so you can cut off at particular point
# So he decided to cut off at rank 10 (but rank 40 seems more appropriate)
# With only 10 vectors, you can see boat still

boat = np.loadtxt('data/boat.dat')
plt.figure()
_ = plt.imshow(boat,cmap = cm.Greys_r)
plt.show()

u,s,vt=np.linalg.svd(boat,full_matrices=False)
_ = plt.plot(s)
plt.title('Singular values of boat image')
plt.show()

RANK = 10

for RANK in [10, 40]:
    # construct a rank-40 version of the boat
    scopy = s.copy()
    scopy[RANK:]=0
    boatApprox = u.dot(np.diag(scopy)).dot(vt)
    #
    plt.figure(figsize=(9,6))
    plt.subplot(1,2,1)
    plt.imshow(boatApprox,cmap = cm.Greys_r)
    plt.title('Rank '+str(RANK)+' Boat')
    plt.subplot(1,2,2)
    plt.imshow(boat,cmap = cm.Greys_r)
    plt.title('Rank 512 Boat')
```

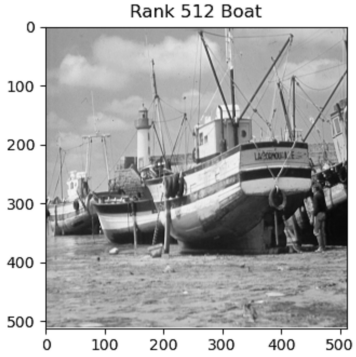
File - C:\Users\Wolfs\PycharmProjects\cs506lec20211006\svd-on-image.py

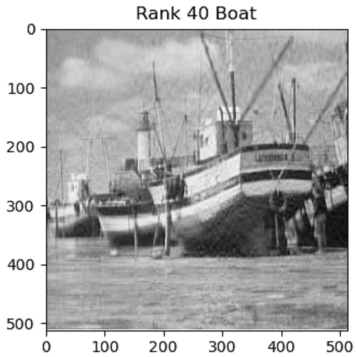
```
38      _ = plt.subplots_adjust(wspace=0.5)
39      plt.show()
40
```

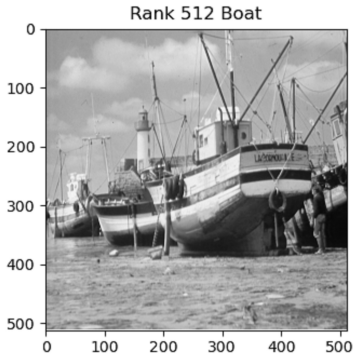




Singular values of boat image
70000
60000
50000
40000
30000
20000
10000
0
0
100
200
300
400
500


Rank 10 Boat
0
100
200
300
400
500
0
100
200
300
400
500


Rank 512 Boat
0
100
200
300
400
500
0
100
200
300
400
500


Rank 40 Boat
0
100
200
300
400
500
0
100
200
300
400
500


Rank 512 Boat
0
100
200
300
400
500
0
100
200
300
400
500

File - C:\Users\Wolfs\PycharmProjects\cs506lec20211006\relative-error.py

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# May want to center the matrix before doing stuff
# This is a trick you may want to do when pre-processing your dataset

with open('data/AbileneFlows/odnames','r') as f:
    odnames = [line.strip() for line in f]
dates = pd.date_range('9/1/2003',freq='10min',periods=1008)
Atraf = pd.read_table('data/AbileneFlows/X',sep='  ',header=None,names=odnames,engine='python')
Atraf.index = dates
print(Atraf.shape)

print("rank of our matrix is ", np.linalg.matrix_rank(Atraf))

u,s,vt = np.linalg.svd(Atraf)

fig = plt.figure(figsize=(6,4))
plt.plot(range(1,1+len(s)),s)
plt.xlabel(r'$k$',size=20)
plt.ylabel(r'$\sigma_k$',size=20)
_ = plt.title(r'Singular Values of $A$',size=20)
plt.show()

fig = plt.figure(figsize=(6,4))
Anorm = np.linalg.norm(Atraf)
plt.plot(range(1,21),s[0:20]/Anorm)
plt.xlim([0,20])
plt.xlabel(r'$k$',size=20)
_ = plt.ylabel(r'$\sigma_k$',size=20)
plt.show()

fig = plt.figure(figsize=(6,4))
Anorm = np.linalg.norm(Atraf)
err = np.cumsum(s[::-1]**2)
err = np.sqrt(err[::-1])
```

File - C:\Users\Wolfs\PycharmProjects\cs506lec20211006\relative-error.py

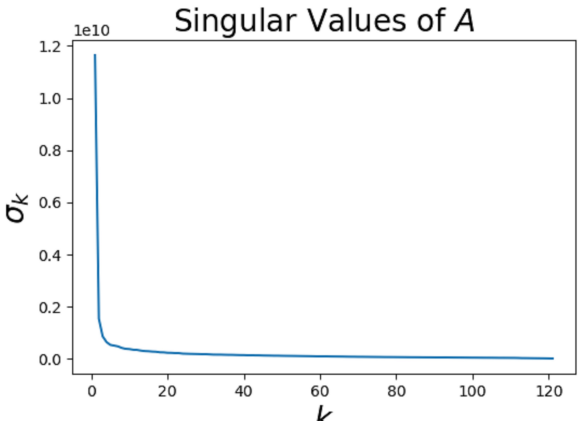
```
plt.plot(range(1,21),err[:20]/Anorm)
plt.xlim([0,20])
plt.xlabel(r'$k$',size=16)
plt.ylabel(r'relative F-norm error',size=16)
_ = plt.title(r'Relative Error of rank-$k$
approximation to $A$',size=16)
plt.show()
```

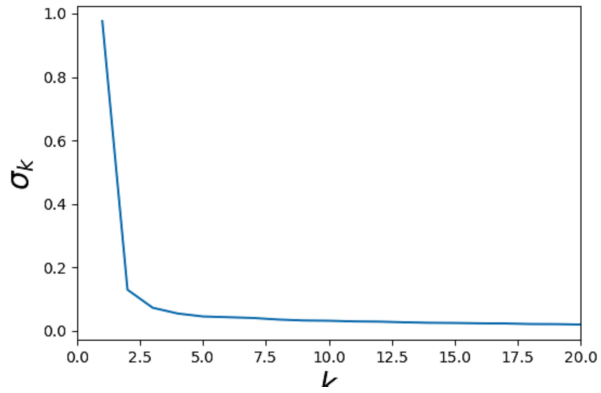
File - C:\Users\Wolfs\PycharmProjects\cs506lec20211006\relative-error.py

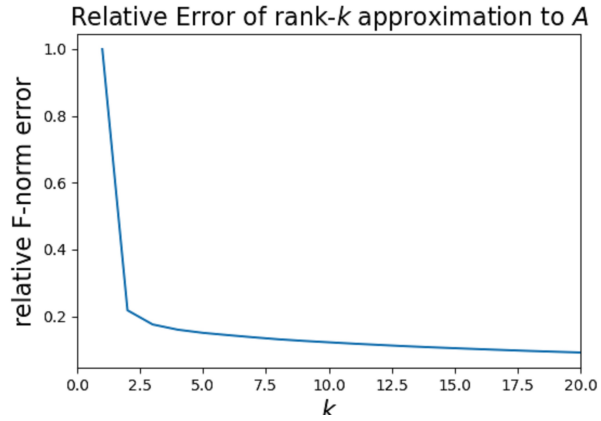
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# May want to center the matrix before doing stuff
# This is a trick you may want to do when pre-processing your dataset

with open('data/AbileneFlows/odnames','r') as f:
    odnames = [line.strip() for line in f]
dates = pd.date_range('9/1/2003',freq='10min',periods=1008)
Atraf = pd.read_table('data/AbileneFlows/X',sep='  ',header=None,names=odnames,engine='python')
Atraf.index = dates
print(Atraf.shape)

print("rank of our matrix is ", np.linalg.matrix_rank(Atraf))

u,s,vt = np.linalg.svd(Atraf)

fig = plt.figure(figsize=(6,4))
plt.plot(range(1,1+len(s)),s)
plt.xlabel(r'$k$',size=20)
plt.ylabel(r'$\sigma_k$',size=20)
_ = plt.title(r'Singular Values of $A$',size=20)
plt.show()

fig = plt.figure(figsize=(6,4))
Anorm = np.linalg.norm(Atraf)
plt.plot(range(1,21),s[0:20]/Anorm)
plt.xlim([0,20])
plt.xlabel(r'$k$',size=20)
_ = plt.ylabel(r'$\sigma_k$',size=20)
plt.show()

fig = plt.figure(figsize=(6,4))
Anorm = np.linalg.norm(Atraf)
err = np.cumsum(s[::-1]**2)
err = np.sqrt(err[::-1])
```

File - C:\Users\Wolfs\PycharmProjects\cs506lec20211006\relative-error.py

```
38 plt.plot(range(1,21),err[:20]/Anorm)
39 plt.xlim([0,20])
40 plt.xlabel(r'$k$',size=16)
41 plt.ylabel(r'relative F-norm error',size=16)
42 _ = plt.title(r'Relative Error of rank-$k$
   approximation to $A$',size=16)
43 plt.show()
44
45
46
```


1.0
0.8
0.6
0.4
0.2
0.0
σ_k
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
k

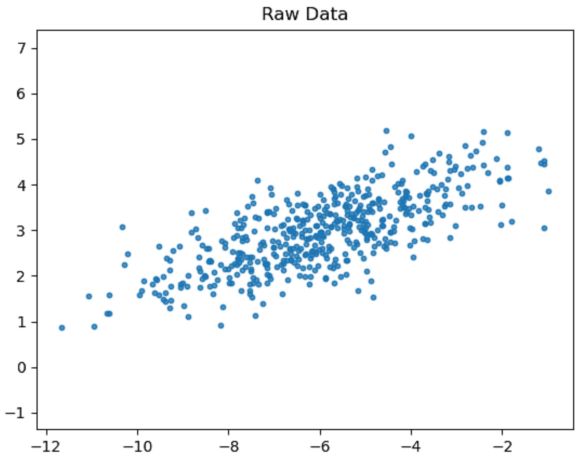

Relative Error of rank-k approximation to A
relative F-norm error
1.0
0.8
0.6
0.4
0.2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
k

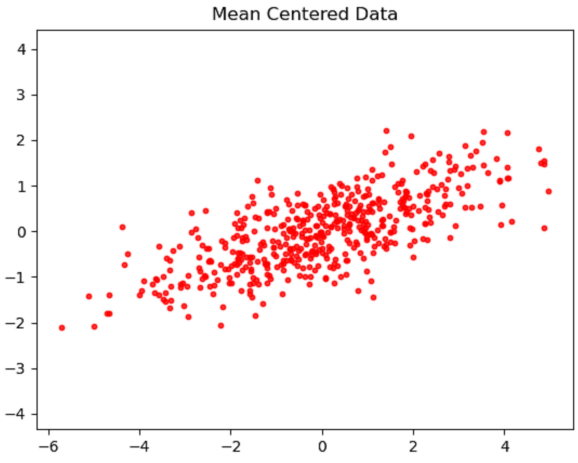
File - C:\Users\Wolfs\PycharmProjects\cs506lec20211006\svd-pca.py

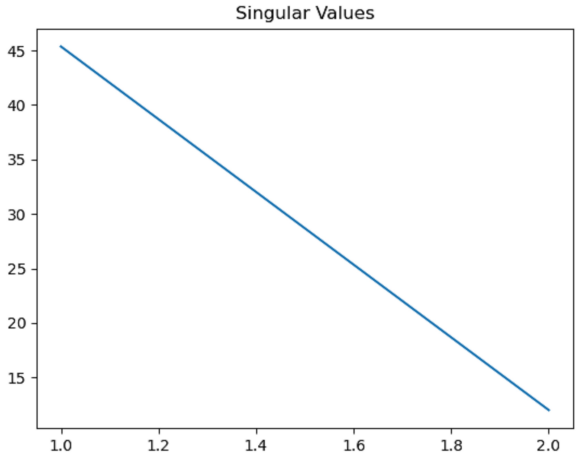
```python
import numpy as np
import matplotlib.pyplot as plt

# You want ot normalize your data first, some features may have scales that are pretty incompatible
# Mean center your data
# We are using numpy as our linear algebra library
# Singular values: 2 values (we only had two dimensions)
# The points in blue in the last plot are the rank 1 approximation of the points in red projected onto the
# green line
# The feature from this column (the first, the strongest) may more accurately represent the dataset
# So then we end up with the dataset with the blue points

n_samples = 500
C = np.array([[0.1, 0.6], [2., .6]])
X = np.random.randn(n_samples, 2) @ C + np.array([-6, 3])
plt.axis('equal')
plt.scatter(X[:, 0], X[:, 1], s=10, alpha=0.8)
plt.title("Raw Data")
plt.show()

Xc = X - np.mean(X,axis=0)
plt.axis('equal')
plt.scatter(Xc[:, 0], Xc[:, 1], s=10, alpha=0.8, color='r')
plt.title("Mean Centered Data")
plt.show()

u, s, vt = np.linalg.svd(Xc,full_matrices=False)
plt.plot(range(1,len(s)+1),s)
plt.title("Singular Values")
plt.show()

scopy = s.copy()
scopy[1] = 0.
reducedX = u @ np.diag(scopy) @ vt
```

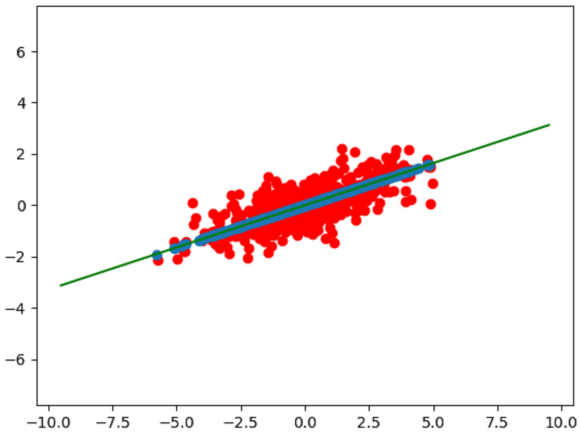
File - C:\Users\Wolfs\PycharmProjects\cs506lec20211006\svd-pca.py

```
35 plt.axis('equal')
36 plt.scatter(Xc[:,0],Xc[:,1], color='r')
37 plt.scatter(reducedX[:,0], reducedX[:,1])
38 endpoints = np.array([[-10],[10]]) @ vt[[0],:]
39 _ = plt.plot(endpoints[:,0], endpoints[:,1], 'g-')
40 plt.show()
41
42
```


Mean Centered Data
4
3
2
1
0
−1
−2
−3
−4
−6
−4
−2
0
2
4


Singular Values
45
40
35
30
25
20
15
1.0
1.2
1.4
1.6
1.8
2.0

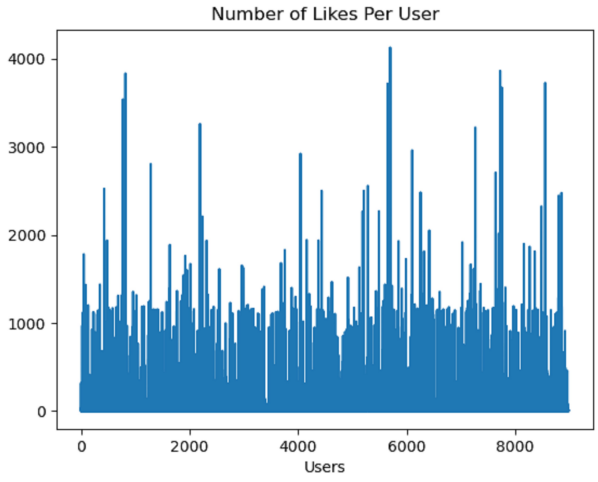

6
4
2
0
−2
−4
−6
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

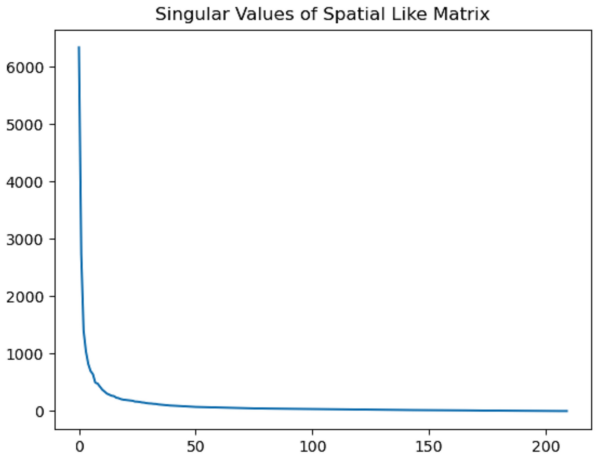
File - C:\Users\Wolfs\PycharmProjects\cs506lec20211006\svd-anomally-det.py

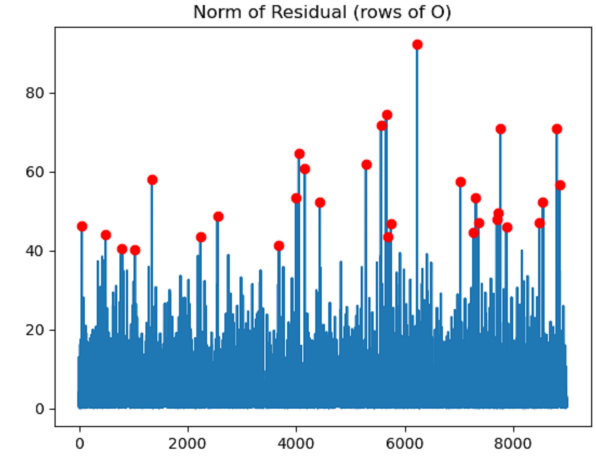
```python
import numpy as np
import matplotlib.pyplot as plt

# The most anomalous users are not what is expected
# Singular value decomposition can pinpoint anomalous users really quickly, but could be wrong

data = np.loadtxt('data/spatial_data.txt')
# This data consists of the number of 'Likes'
# during a six month period, for each of 9000
# users across the 210 content categories that
# Facebook assigns to pages.

FBSpatial = data[:,1:]
FBSnorm = np.linalg.norm(FBSpatial,axis=1,ord=1)
plt.plot(FBSnorm)
plt.title('Number of Likes Per User')
_ = plt.xlabel('Users')
plt.show()

u,s,vt = np.linalg.svd(FBSpatial,full_matrices=False)
plt.plot(s)
_ = plt.title('Singular Values of Spatial Like Matrix')
plt.show()

scopy = s.copy()
scopy[25:] = 0.
N = u @ np.diag(scopy) @ vt
O = FBSpatial - N
Onorm = np.linalg.norm(O,axis=1)
anomSet = np.argsort(Onorm)[-30:]
plt.plot(Onorm)
plt.plot(anomSet,Onorm[anomSet],'ro')
_ = plt.title('Norm of Residual (rows of O)')
plt.show()

anomSet = np.argsort(Onorm)[-30:]
plt.plot(FBSnorm)
plt.plot(anomSet,FBSnorm[anomSet],'ro')
_ = plt.title('Top 30 Anomalous Users - Total Number
```

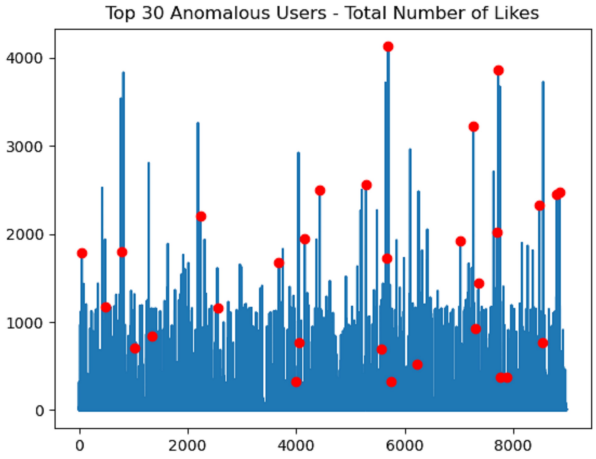
File - C:\Users\Wolfs\PycharmProjects\cs506lec20211006\svd-anomally-det.py

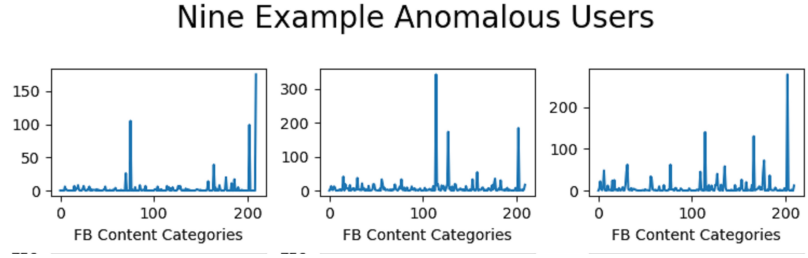
```
39 of Likes')
40 plt.show()
41
42 # anomalous users
43 plt.figure(figsize=(9,6))
44 for i in range(1,10):
45     ax = plt.subplot(3,3,i)
46     plt.plot(FBSpatial[anomSet[i-1],:])
47     plt.xlabel('FB Content Categories')
48 plt.subplots_adjust(wspace=0.25,hspace=0.45)
49 _ = plt.suptitle('Nine Example Anomalous Users',size=
   20)
50 plt.show()
51
52 # normal users
53 set = np.argsort(Onorm)[0:7000]
54 # that have high overall volume
55 max = np.argsort(FBSnorm[set])[::-1]
56 plt.figure(figsize=(9,6))
57 for i in range(1,10):
58     ax = plt.subplot(3,3,i)
59     plt.plot(FBSpatial[set[max[i-1]],:])
60     plt.xlabel('FB Content Categories')
61 plt.subplots_adjust(wspace=0.25,hspace=0.45)
62 _ = plt.suptitle('Nine Example Normal Users',size=20)
63 plt.show()
64
65
```

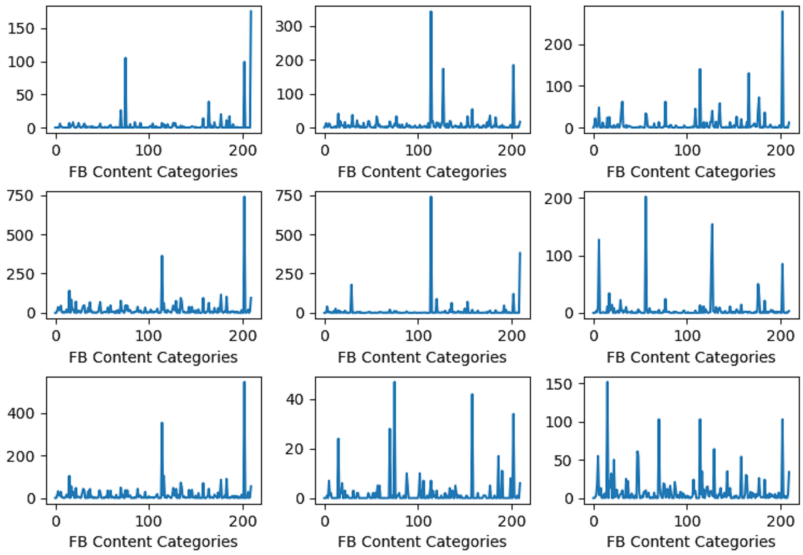

Singular Values of Spatial Like Matrix
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200


Norm of Residual (rows of O)
80
60
40
20
0
0
2000
4000
6000
8000


Top 30 Anomalous Users - Total Number of Likes
4000
3000
2000
1000
0
0
2000
4000
6000
8000


Nine Example Anomalous Users
150
100
50
0
0
100
200
FB Content Categories
300
200
100
0
0
100
200
FB Content Categories
200
100
0
0
100
200
FB Content Categories

Nine Example Anomalous Users



Nine Example Normal Users