Dimension Reduction

CS 759/859 Natural Language Processing Lecture 6

Samuel Carton, University of New Hampshire

Last lecture

Key idea: Clustering text

Concepts

- Unsupervised learning
- Clustering
- K-means clustering
- Clustering metrics
    - Extrinsic
        - Mutual information
    - Intrinsic
        - Silhouette score
- Representing induvial clusters

Toolkits

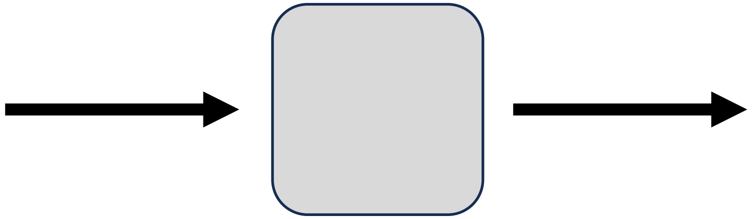
- Matplotlib for data visualization
- Scikit-Learn for model building and evaluation

Vectorizing text revisited

Preprocessed text

	preprocessed
0	my point is that you set up your view as the o...
1	by ' 8 grey level imag ' you mean 8 item of 1b...
2	first annual phig user group confer the first ...
3	i respond to jim 's other articl today , but i...
4	well , i am place a file at my ftp today that ...
5	i 'm also interest in info both public domain ...
6	they did the rollout alreadi ? ? ! ? i am go t...
7	georg william herbert sez : i like your optim ...
8	from the `` jpl univers '' april 23 , 1993 vlb...
9	] the `` corrupt over and over '' theori is pr...

1-gram vectorization



Sparse matrix

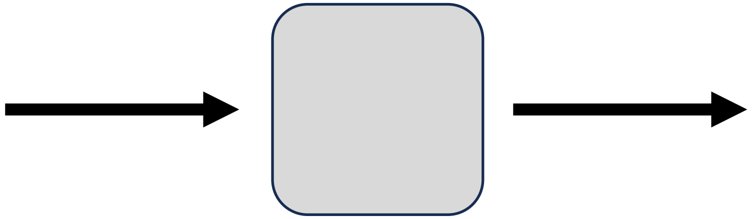
```
matrix([[0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.]])
```

Vectorizing text revisited

Preprocessed text

	preprocessed
0	my point is that you set up your view as the o...
1	by ' 8 grey level imag ' you mean 8 item of 1b...
2	first annual phig user group confer the first ...
3	i respond to jim 's other articl today , but i...
4	well , i am place a file at my ftp today that ...
5	i 'm also interest in info both public domain ...
6	they did the rollout alreadi ? ? ! ? i am go t...
7	georg william herbert sez : i like your optim ...
8	from the `` jpl univers '' april 23 , 1993 vlb...
9	] the `` corrupt over and over '' theori is pr...

1-gram vectorization



Sparse matrix

```
matrix([[0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.]])
```

Comprehension questions

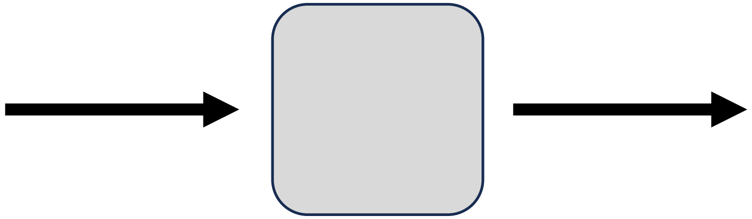
- How many rows does the sparse matrix have?
- How many columns?
- What does it mean if m[i][j] > 0?
- How will the matrix differ between CountVectorizer and TfidfVectorizer?

Vectorizing text revisited

Preprocessed text

	preprocessed
0	my point is that you set up your view as the o...
1	by ' 8 grey level imag ' you mean 8 item of 1b...
2	first annual phig user group confer the first ...
3	i respond to jim 's other articl today , but i...
4	well , i am place a file at my ftp today that ...
5	i 'm also interest in info both public domain ...
6	they did the rollout alreadi ? ? ! ? i am go t...
7	georg william herbert sez : i like your optim ...
8	from the `` jpl univers '' april 23 , 1993 vlb...
9	] the `` corrupt over and over '' theori is pr...

1-gram vectorization



Sparse matrix

```
matrix([[0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.]])
```

Comprehension questions

- How many rows does the sparse matrix have?
  - Number of texts
- How many columns?
  - Size of vocabulary
- What does it mean if m[i][j] > 0?
  - The i^{th} text contains the j^{th} word
- How will the matrix differ between CountVectorizer and TfidfVectorizer?
  - Counts vs TF-IDF scores, 0’s will be the same

Disadvantages of sparse matrices

In a given sparse text vector matrix, 99% of values are going to be 0

Example: 4 categories from the 20-newsgroups dataset

```
our_categories= [
    "alt.atheism",
    "talk.religion.misc",
    "comp.graphics",
    "sci.space"]
```

- 3387 texts
- Average text length: 244 tokens
- 5945 unique tokens in vocabulary

So…

- Sparse matrix is 3387x5945 = 20,135,715 elements
- But only 190,747 nonzero elements
- So space utilization of ~1%

```
matrix([[0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.]])
```

Bummer

Disadvantages of sparse matrices

Although why is it actually a bummer?

One big issue: **compute**

- Training and inference on ML models usually scales at least linearly with length/width of data
- So it's unfortunate to have to work with giant matrices that have a bunch of wasted space

```
matrix([[0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.]])
```

Sparse matrices in Python

The SciPy language has implementations of **sparse matrices** which store only nonzero values

- https://docs.scipy.org/doc/scipy/reference/sparse.html
- A few different options for how exactly values are stored (CSR, CSC, etc)
- Scit-learn vectorizers output CSR matrices, which each row is stored in a compressed form

Still kind of a pain to work with though

```python
vectorizer = TfidfVectorizer(
    max_df=0.5,
    min_df=5,
    stop_words="english",
)

ng_X = vectorizer.fit_transform(ng_df['preprocessed'])
```

```python
ng_X.shape
```

```
(3387, 5945)
```

```python
ng_X.todense()
```

```
matrix([[0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        ...,
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.]])
```

```python
ng_X
```

```
<3387x5945 sparse matrix of type '<class 'numpy.float64'>'
	with 190747 stored elements in Compressed Sparse Row format>
```

Sparse matrices in Python

Two key ways to use Scipy sparse matrices:

1. Convert them to dense numpy arrays:
   - Be aware that this may be huge and overwhelm your RAM

```
1   ng_X.todense()
```

```
matrix([[0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        ...,
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., ..., 0., 0., 0., 0., 0.]])
```

2. Identify the row and column indices of nonzero values:

```
1   nonzero_indices = ng_X.nonzero()
2   nonzero_indices
```

```
(array([   0,    0,    0,    0,    0, ..., 3386, 3386, 3386, 3386, 3386], dtype=int32),
 array([2546, 1930, 2054, 4349, 1701, ..., 2491, 3675, 5747, 2546, 4156], dtype=int32))
```

```
1   ng_X[nonzero_indices[0][0], nonzero_indices[1][0]]
```

```
0.07893546173278095
```

Disadvantages of sparse matrices

Another big issue: **sparsity**

Consider “He is an idiot” vs. “They are morons”

- Pretty similar!

Rest of the vocabulary

	he	they	is	are	an	idiot	moron	…
“He is an idiot”	1	0	1	0	1	1	0	0…
“They are morons”	0	1	0	1	0	0	1	0…

What happens when we try to calculate the cosine similarity between these two vectors?

Reminder: $$\text{cosine similarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}},$$

Solution: dimension reduction

It would be nice if we had a way to take these big sparse matrices and squeeze them down to a **denser** representation

- Without losing too much information (compare to lossy vs. lossless compression)

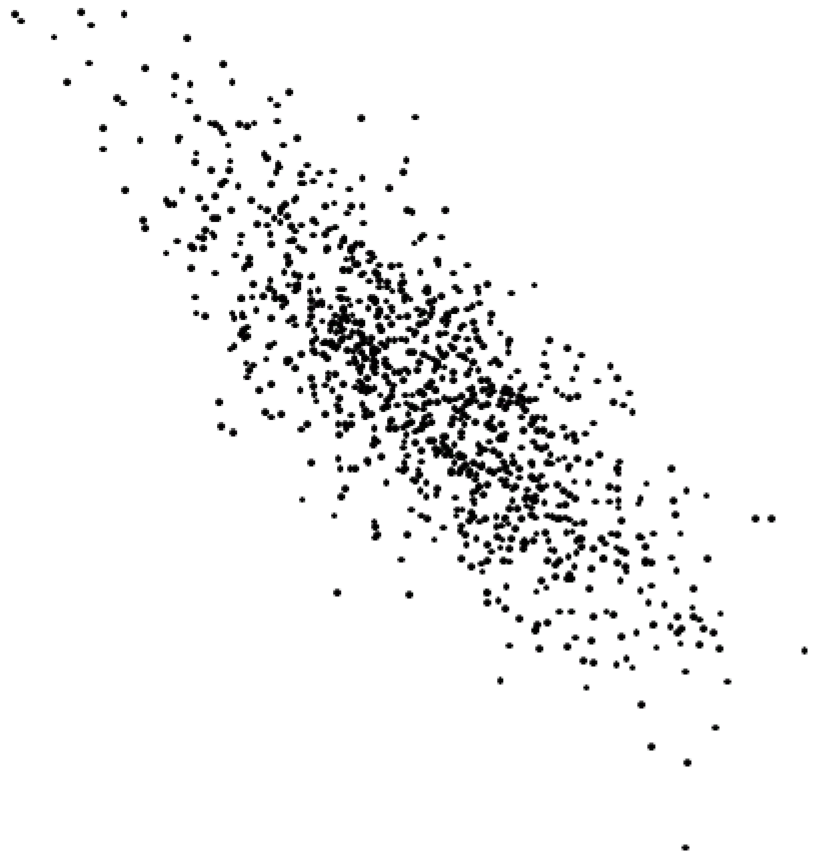
Go from 3387x5945 to 3387x**100** or 3387x**50**

If we could do that then:

1. They'd be easier to work with computationally
2. We *might* begin to be able to overcome some of our data sparsity issues
    - Spoiler alert: this is what word embeddings are

Key idea: **dimension reduction**

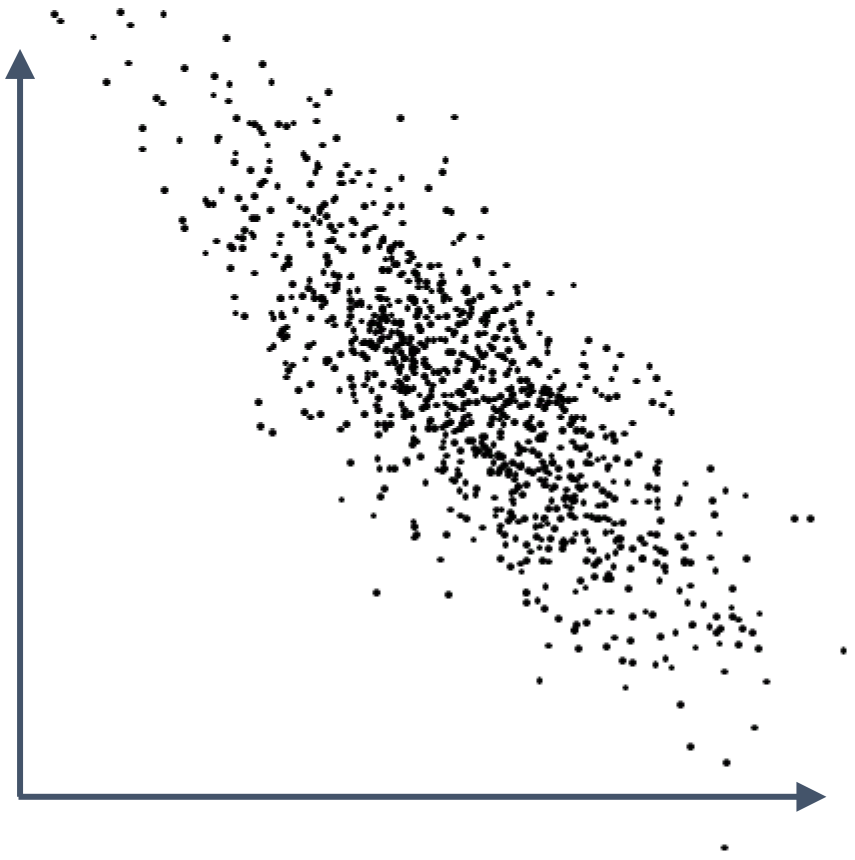
How to represent data?



When we have a bunch of datapoints, and each datapoint is a vector of N-elements, we can think of them as points in an N-dimensional space.

Next few slides partially borrowed from
https://cbmm.mit.edu/learning-hub/tutorials/computational-tutorial/dimensionality-reduction-i

How to represent data?

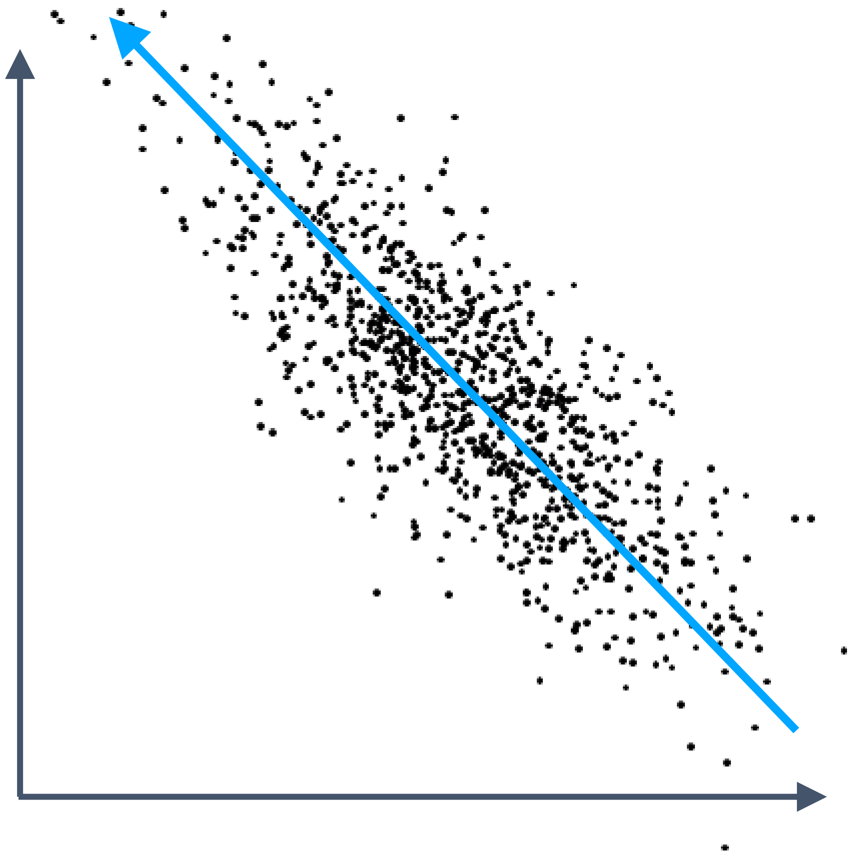


The dataset on the could represent a bunch of 2-element vectors, where there first element is the x value and the second is the y value:

D=

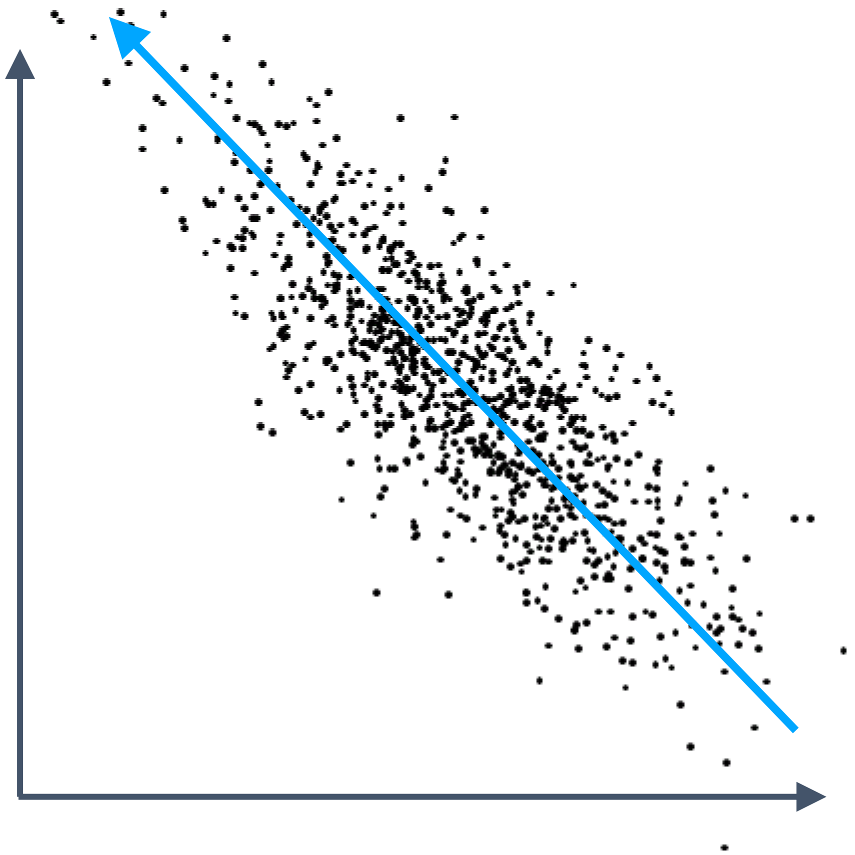
X	Y
.45	.47
.22	.76
.64	.48
.51	.59
.17	.91
.88	.20
...	

How to represent data?



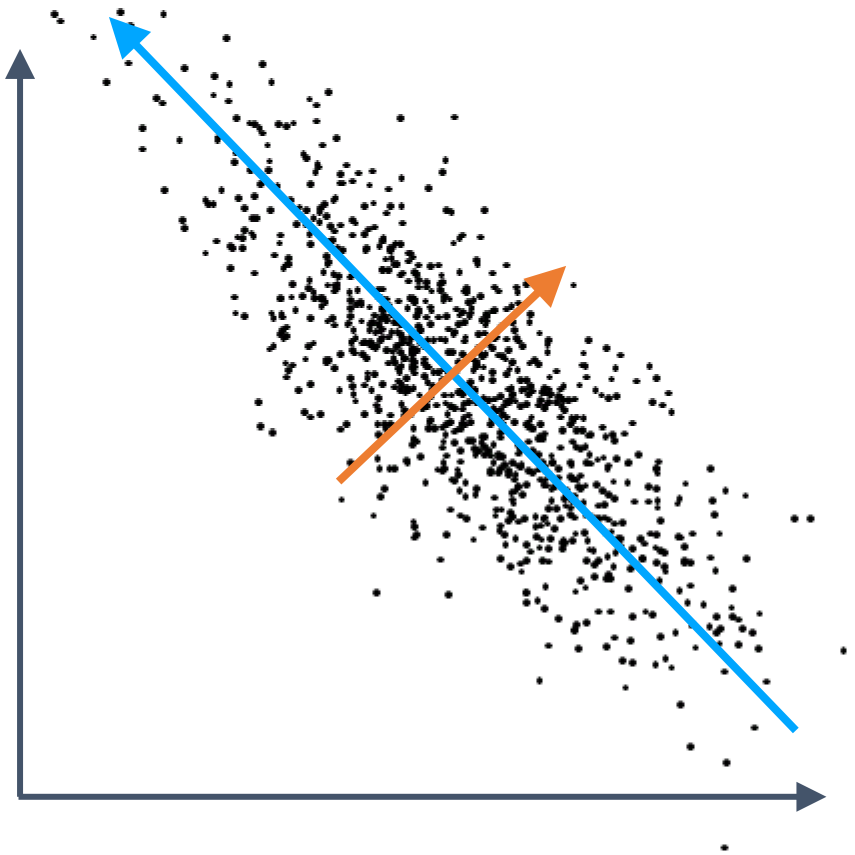
But one thing you can note is that most of the **variance** here is actually just along one line

How to represent data?



But one thing you can note is that most of the **variance** here is actually just along one line

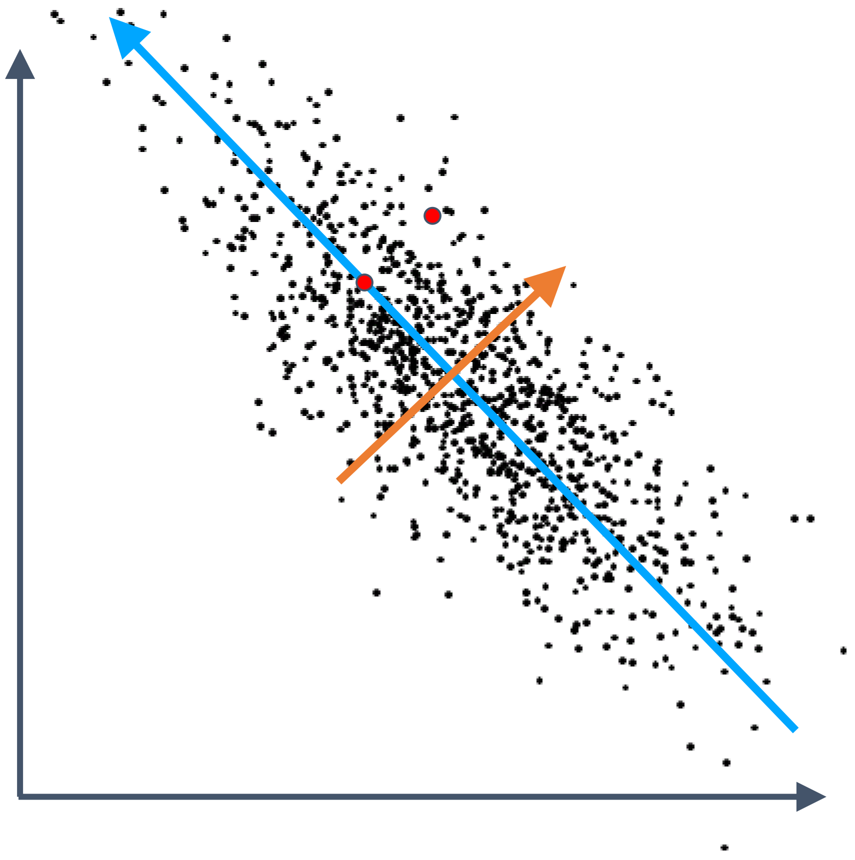
How to represent data?



But one thing you can note is that most of the **variance** here is actually just along one line

And relatively little is along the orthogonal direction

How to represent data?

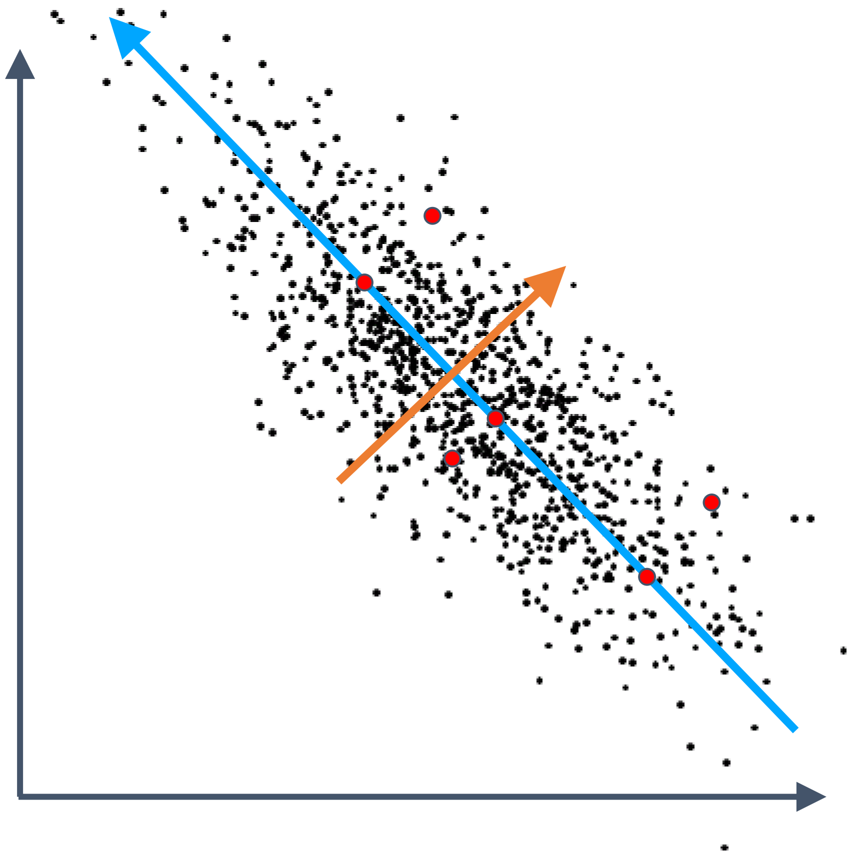


But one thing you can note is that most of the **variance** here is actually just along one line

And relatively little is along the orthogonal direction.

Which means that if you are trying to describe the location of a point with only **one** number, you get the most mileage out of describing how far along the blue line it is

How to represent data?



So if we had to approximate D using only one dimension, we could use the blue line as the new **basis** for our data

D=

X	Y
.45	.47
.22	.76
.64	.48
.51	.59
.17	.91
.88	.20
…	

≈

D*=

V
.44
.80
.34
.37
.99
.15
…

Dimension reduction

Basic idea: take a data matrix of size M×N and compress it to M×D, where D << M, while still retaining most of the useful information in the matrix

- For text, go from M *sparse* vectors of dimensionality V=size of the vocabulary, to M *dense* vectors of size D, where D is significantly smaller (100, 200, etc.)
- While still being useful for classification, clustering, etc.

In many approaches, do this by identifying new **basis vectors** of high variance, and then represent each datapoint in terms of only the most important ones

Two most popular approaches:

- Principal component analysis (PCA)
- Singular value decomposition (SVD)

Matrix multiplication

When you multiply two matrixes $A^{m \times n} * B^{n \times p}$, you get $C^{m \times p}$ by calculating the dot product of every row of A with every column of B

Only matrixes with matching inner dimensions (e.g. m×n versus n×p) can be multiplied

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix}$$

$$\mathbf{AB} = \mathbf{C} = \begin{pmatrix} a_{11}b_{11} + \cdots + a_{1n}b_{n1} & a_{11}b_{12} + \cdots + a_{1n}b_{n2} & \cdots & a_{11}b_{1p} + \cdots + a_{1n}b_{np} \\ a_{21}b_{11} + \cdots + a_{2n}b_{n1} & a_{21}b_{12} + \cdots + a_{2n}b_{n2} & \cdots & a_{21}b_{1p} + \cdots + a_{2n}b_{np} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{11} + \cdots + a_{mn}b_{n1} & a_{m1}b_{12} + \cdots + a_{mn}b_{n2} & \cdots & a_{m1}b_{1p} + \cdots + a_{mn}b_{np} \end{pmatrix}$$

Matrix transpose

Transposing a matrix, denoted by M^T, switches its dimensions.

Very common operation in linear algebra

Also the only way you can multiply a matrix by itself unless it is square

Examples:

$$\begin{bmatrix} 1 & 2 \end{bmatrix}^T = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

Covariance matrix

The **covariance matrix** quantifies the joint variability between two random variables *X* and *Y*

$$cov(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$$

When X is a data matrix of n samples × m features, centered on 0, then the covariance matrix can be calculated as:

$$\mathbf{C} = \frac{\mathbf{X}^\top \mathbf{X}}{n-1}$$

So for a count or TF-IDF matrix, we end up with an m × m matrix where C[i][j] represents… what?

PCA and SVD

Principle component analysis (PCA) and singular value decomposition (SVD) are both **matrix factorization** techniques, which calculate how to express the covariance matrix as a product of lower-dimensionality matrices.

PCA performs an **eigendecomposition** of the covariance matrix C, which learns to represent it as $C = W\Lambda W^{-1}$, where W is a m×m matrix of **eigenvectors**, and Λ a diagonal m×m matrix of **eigenvalues**

SVD performs a decomposition of C into a product of two unitary matrices (U, V*) and a rectangular diagonal matrix of singular values (Σ): $C = U\Sigma V^*$

In both cases, we can get back a lower-dimension approximation of our original **X** by performing the product with subsets of the factor matrices:

Example for PCA: $\mathbf{X}_k = \mathbf{X}\mathbf{W}_k$

PCA vs. SVD in practice

In practice, PCA is calculated “under the hood” using SVD (truncated SVD for text)

- SVD singular values are easily convertible to PCA eigenvalues

So mostly you will be using SVD for dimension reduction in practice

And you can look at the singular values to get a sense of how much of the variance of the original data are explained in the D dimensions you’ve chosen to retain.

Conventional clustering

```python
our_categories= [
    "alt.atheism",
    "talk.religion.misc",
    "comp.graphics",
    "sci.space",

]

ng_dataset = fetch_20newsgroups(remove=('headers', 'footers', 'quotes'),
                                categories=our_categories,
                                subset="all",
                                shuffle=True,
                                random_state=42,
                                )
```

```python
ng_df = pd.DataFrame({'text':ng_dataset.data, 'group':ng_dataset.target})
ng_df
```

	text	group
0	My point is that you set up your views as the ...	0
1	\nBy '8 grey level images' you mean 8 items of...	1
2	FIRST ANNUAL PHIGS USER GROUP CONFERENCE\n\n ...	1
3	I responded to Jim's other articles today, but...	3
4	\nWell, I am placing a file at my ftp today th...	1
...	...	...
3382	I am working on a program to display 3d wirefr...	1
3383	\n Did the Russian spacecraft(s) on the ill-f...	2
3384	\n\nOh gee, a billion dollars! That'd be just...	2
3385	I am looking for software to run on my brand n...	1
3386	Within the next several months I'll be looking...	1

3387 rows × 2 columns

Conventional clustering

```python
model = KMeans(n_clusters=4,
               max_iter=100,
               n_init=5).fit(ng_X)

ng_df['cluster'] = model.predict(ng_X)
ng_df
```

	text	group	preprocessed	cluster
0	My point is that you set up your views as the ...	0	my point is that you set up your view as the o...	1
1	\nBy '8 grey level images' you mean 8 items of...	1	by ' 8 grey level imag ' you mean 8 item of 1b...	2
2	FIRST ANNUAL PHIGS USER GROUP CONFERENCE\n\n ...	1	first annual phig user group confer the first ...	0
3	I responded to Jim's other articles today, but...	3	i respond to jim 's other articl today , but i...	0
4	\nWell, I am placing a file at my ftp today th...	1	well , i am place a file at my ftp today that ...	2
...	...	...	...	...
3382	I am working on a program to display 3d wirefr...	1	i am work on a program to display 3d wirefram ...	2
3383	\n Did the Russian spacecraft(s) on the ill-f...	2	did the russian spacecraft (s) on the ill-fa...	2
3384	\n\nOh gee, a billion dollars! That'd be just...	2	oh gee , a billion dollar ! that 'd be just ab...	0
3385	I am looking for software to run on my brand n...	1	i am look for softwar to run on my brand new t...	2
3386	Within the next several months I'll be looking...	1	within the next sever month i 'll be look for ...	2

3387 rows × 4 columns

Conventional clustering

```python
from sklearn.metrics import normalized_mutual_info_score, adjusted_rand_score, silhouette_score
```

```python
def evaluate_clustering(X, true_labels, cluster_assignments):
    print(f"Normalized mutual information between true labels and cluster assignments: \
    {normalized_mutual_info_score(true_labels, cluster_assignments):.3f}")
    print(f"Adjusted Rand score between true labels and cluster assignments: \
    {adjusted_rand_score(true_labels, cluster_assignments):.3f}")
    print(f"Average silhouette score of cluster assignments: \
    {silhouette_score(X, cluster_assignments, sample_size=2000):.3f}")

```

```python
evaluate_clustering(ng_X, ng_df['group'], ng_df['cluster'])
```

```
Normalized mutual information between true labels and cluster assignments: 0.347
Adjusted Rand score between true labels and cluster assignments: 0.191
Average silhouette score of cluster assignments: 0.009
```

SVD for clustering

```
from sklearn.decomposition import TruncatedSVD
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import Normalizer
```

```
pipeline = make_pipeline(TruncatedSVD(n_components=100), Normalizer(copy=False))

pipeline.fit(ng_X)
```

```
Pipeline(steps=[('truncatedsvd', TruncatedSVD(n_components=100)),
                ('normalizer', Normalizer(copy=False))])
```

SVD for clustering

```
print(pipeline[0].explained_variance_ratio_)
```

```
[0.00584104 0.0095496  0.00653745 0.00486296 0.00444869 0.00418118
 0.00406312 0.00366628 0.00364483 0.00347826 0.0031842  0.00308523
 0.00297885 0.00293872 0.00281783 0.00278395 0.00266414 0.00263875
 0.00255509 0.00251268 0.00250039 0.00247195 0.00244391 0.00239149
 0.00237247 0.00234858 0.0023194  0.00229551 0.00225731 0.0022259
 0.0022006  0.00215427 0.0021362  0.00212559 0.00208279 0.00206926
 0.00206116 0.00205015 0.00202851 0.00200864 0.00197045 0.00197029
 0.00196684 0.00192687 0.00191495 0.00189608 0.0018912  0.0018752
 0.00184478 0.00183305 0.0018195  0.00181102 0.00178057 0.00177471
 0.00176546 0.00175936 0.00174562 0.00173275 0.00170926 0.00170442
 0.00169418 0.00167836 0.0016533  0.00164725 0.00164349 0.00163075
 0.00161503 0.00159829 0.00158777 0.00157845 0.00157376 0.00155967
 0.00155458 0.00154128 0.00153191 0.00152127 0.00152078 0.00151378
 0.00149234 0.0014815  0.00147753 0.0014712  0.00145651 0.00144281
 0.00143771 0.00143188 0.00142375 0.00141679 0.00140436 0.00139205
 0.00138605 0.00136832 0.00136518 0.00135738 0.00134907 0.00133329
 0.00132955 0.00130884 0.00130752 0.00129157]
```

```
print(pipeline[0].explained_variance_ratio_.sum())
```

```
0.219038407432514
```

SVD for clustering

```python
r_ng_X = pipeline.transform(ng_X)
r_model = KMeans(n_clusters=5, random_state=0).fit(r_ng_X)
ng_df['r_cluster'] = r_model.predict(r_ng_X)
ng_df
```

	text	group	preprocessed	cluster	r_cluster
0	My point is that you set up your views as the ...	0	my point is that you set up your view as the o...	1	2
1	\nBy '8 grey level images' you mean 8 items of...	1	by ' 8 grey level imag ' you mean 8 item of 1b...	2	0
2	FIRST ANNUAL PHIGS USER GROUP CONFERENCE\n\n ...	1	first annual phig user group confer the first ...	0	0
3	I responded to Jim's other articles today, but...	3	i respond to jim 's other articl today , but i...	0	2
4	\nWell, I am placing a file at my ftp today th...	1	well , i am place a file at my ftp today that ...	2	0
...	...	...	...	...	...
3382	I am working on a program to display 3d wirefr...	1	i am work on a program to display 3d wirefram ...	2	0
3383	\n Did the Russian spacecraft(s) on the ill-f...	2	did the russian spacecraft (s) on the ill-fa...	2	0
3384	\n\nOh gee, a billion dollars! That'd be just...	2	oh gee , a billion dollar ! that 'd be just ab...	0	3
3385	I am looking for software to run on my brand n...	1	i am look for softwar to run on my brand new t...	2	0
3386	Within the next several months I'll be looking...	1	within the next sever month i 'll be look for ...	2	0

3387 rows × 5 columns

SVD for clustering

```
print('Evaluation of dimension-reduced clustering:')
evaluate_clustering(r_ng_X, ng_df['group'], ng_df['r_cluster'])

print('\nEvaluation of original clustering:')
evaluate_clustering(ng_X, ng_df['group'], ng_df['cluster'])
```

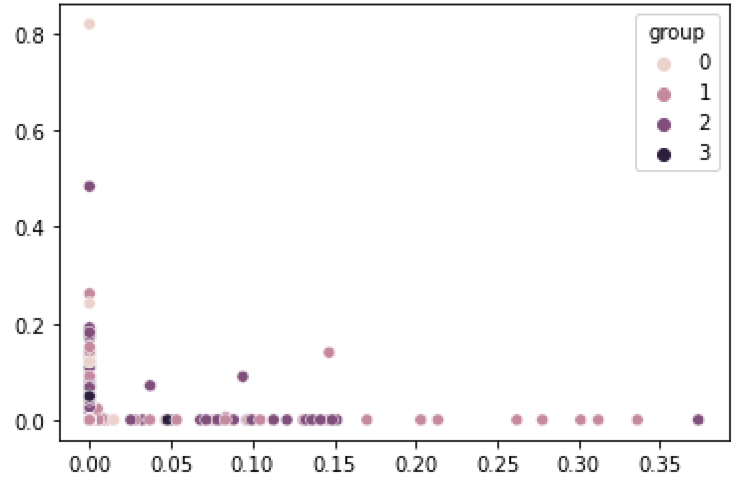
```
Evaluation of dimension-reduced clustering:
Normalized mutual information between true labels and cluster assignments: 0.363
Adjusted Rand score between true labels and cluster assignments: 0.304
Average silhouette score of cluster assignments: 0.030

Evaluation of original clustering:
Normalized mutual information between true labels and cluster assignments: 0.347
Adjusted Rand score between true labels and cluster assignments: 0.191
Average silhouette score of cluster assignments: 0.008
```

SVD for visualizing clusters

```
# Each dimension in the original term-document matrix refers to the presence
# or absence of one word from the vocabulary
# So trying to plot the data in terms of any two of these dimensions is not very
# helpful
dense_ng_X = np.array(ng_X.todense())
sns.scatterplot(x=dense_ng_X[:,0], y=dense_ng_X[:,1], hue=ng_df['group'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cf8d2a580>
```

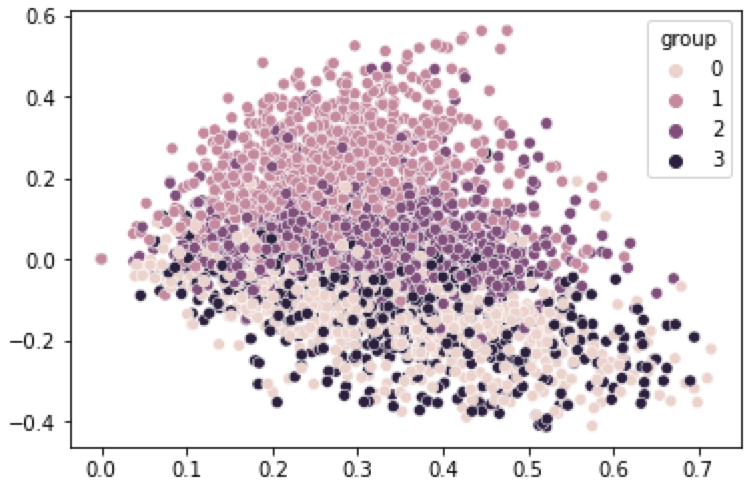


SVD for visualizing clusters

```
# If we instead plot the data in terms of the first two principle
# components, we start seeing some real structure

sns.scatterplot(x=r_ng_X[:,0], y=r_ng_X[:,1], hue=ng_df['group'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cfb2b9280>
```

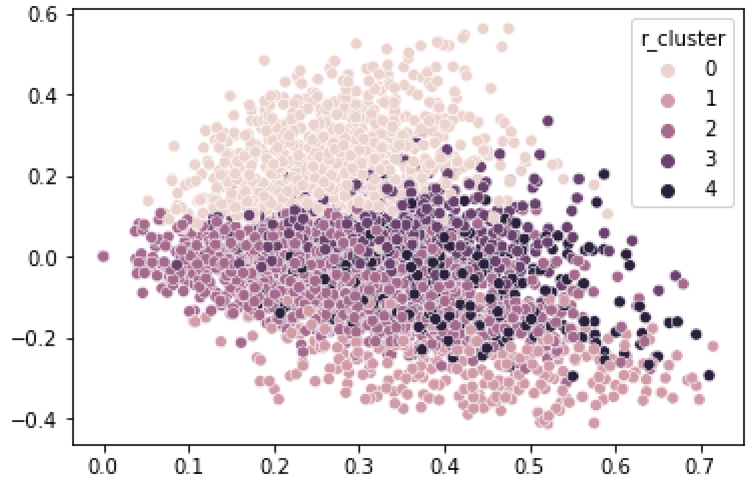


SVD for visualizing clusters

```python
# And that structure gets captured by running K-means clustering on that
# dimension-reduced data

sns.scatterplot(x=r_ng_X[:,0], y=r_ng_X[:,1], hue=ng_df['r_cluster'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cf8c10ca0>
```

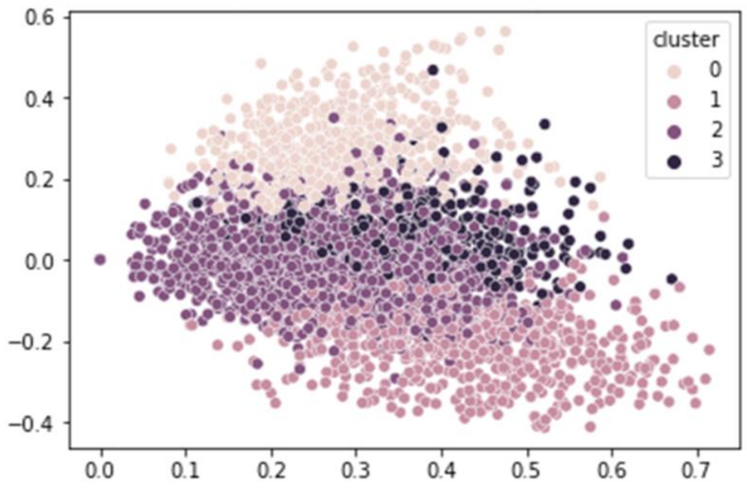


SVD for visualizing clusters

```python
# And even doing clustering on the original sparse matrics captured much
# of that same structure, it's just that it was hard to visualize

sns.scatterplot(x=r_ng_X[:,0], y=r_ng_X[:,1], hue=ng_df['cluster'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1cf8be39a0>
```



Concluding thoughts

In NLP, learning **dense representations** of text is absolutely critical.

You can only get so far with sparse bag-of-words or TF-IDF representations.

Deep learning, aka **representation learning**

- Learns “targeted” dense representations optimized for specific tasks

Dimension reduction: “general-purpose” representations optimized for mathematical properties

- SVD on term-document matrix is called **Latent Semantic Analysis** (1988)

Still useful for prediction, clustering, visualization, etc.

Much of this lecture borrowed from: https://towardsdatascience.com/pca-and-svd-explained-with-numpy-5d13b0d2a4d8