CS-UY 4563: Lecture 23
Semantic Embeddings, Clustering

NYU Tandon School of Engineering, Prof. Christopher Musco

COURSE LOGISTICS

If you have not already, **sign up for a time slot to present your project**. 5 Minute slots on Wed. 5/6 and Mon. 5/11.

- Prepare slides with images to showcase your project.
- Presentation followed by a few questions.
- What's the problem you are looking at? Why's it interesting and exciting?
- What data are you using? What features? What approaches have you tried, what stumbling blocks have you hit?

Presentation will count towards final project grade.

4 PAGE FINAL REPORT

This project is about exploring a data set. Report should be a guide through that exploration process.

- Explain the data set in detail. Show us what the data looks like with examples, scatter plots, histograms, etc.
- What's the question you sought to answer? What features did you focus on to answer that question and why?
- If you method worked, how well did it work? How did you evaluate it? What examples did it do well on? Where did it fail? How would that guide future work?
- If you had some setbacks, why? How did you try to address them? Don't include every failure but discuss any interesting ones.

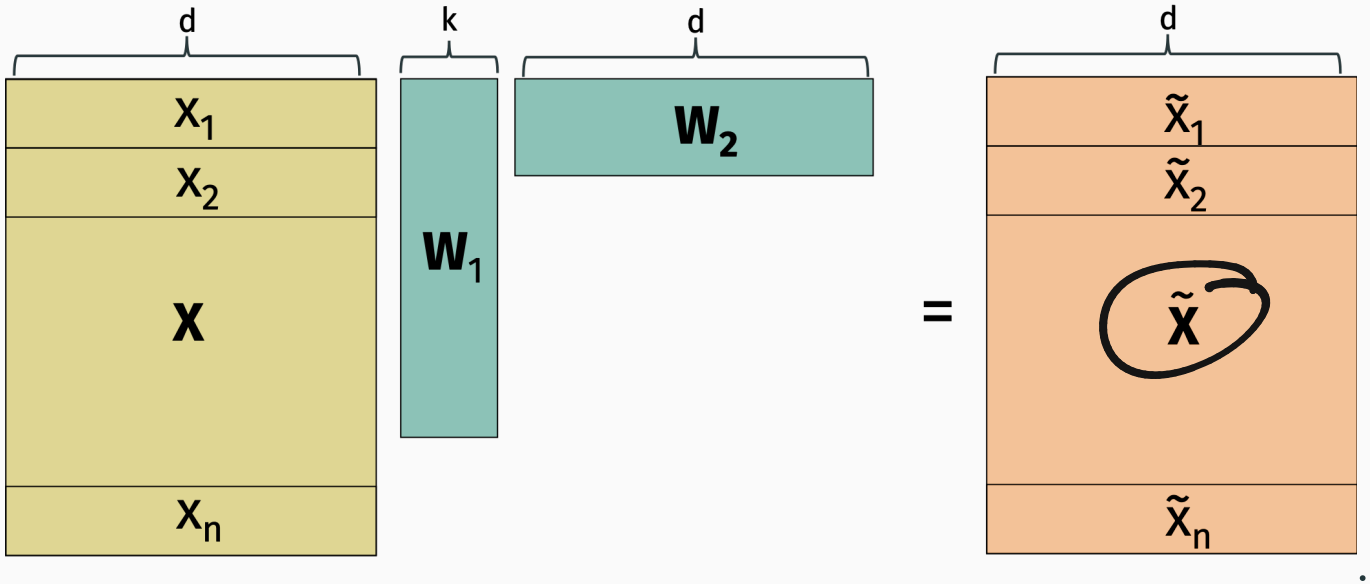
Think about what you might include if you were building a student lab for your project data.

4 PAGE FINAL REPORT

What I'm looking for:

- **Used appropriate tools from the class:** E.g. feature selection or regularization if you over fit, train-test split to evaluate models and set hyper-parameters, etc.
- **Good baseline exploration:** Most common class, *k*-nn, linear regression, etc.
- **Well justified choices:** Loss functions that make sense, final performance metrics that make sense, logical feature transformations.
- **Creativity:** Most ideas (for feature transformations, new models, etc.) will not pan out. But I want to see that you adapted your approach to what you saw in the data.

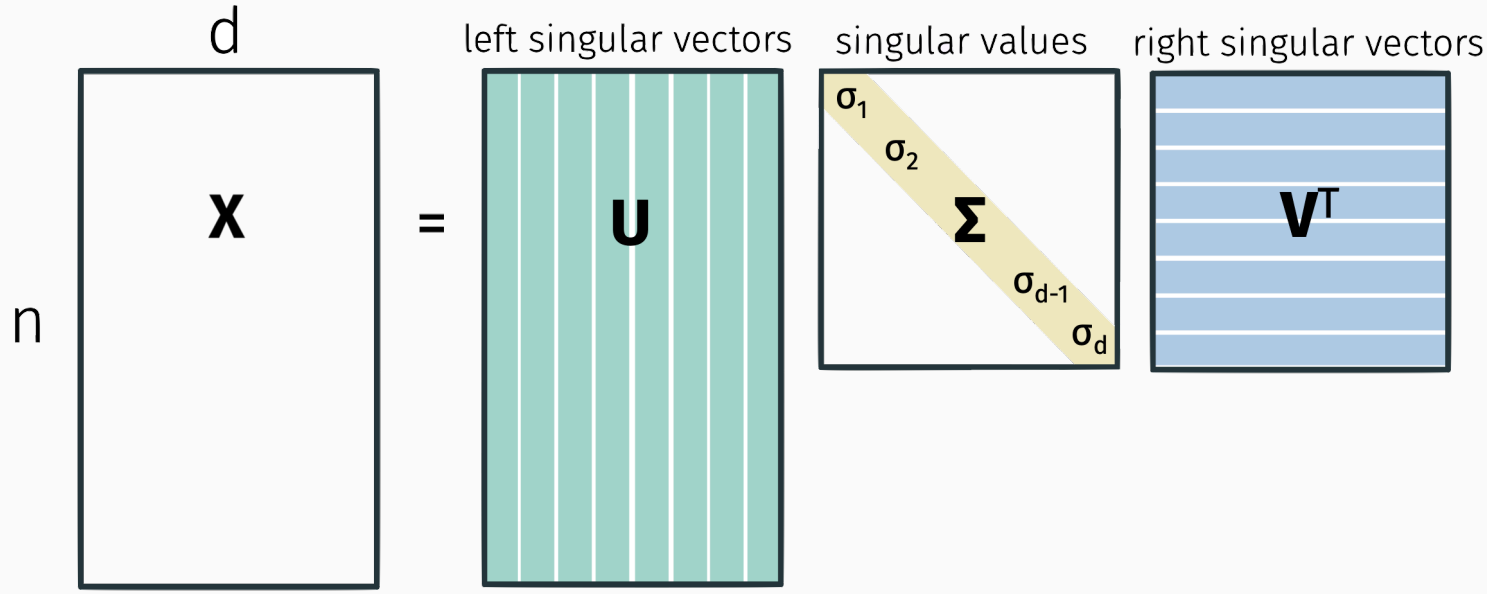
Recall: Given input $\mathbf{X} \in \mathbb{R}^{n \times d}$ and target rank k, find $\mathbf{W}_1$, $\mathbf{W}_2$ too minimize $\|X - \mathbf{X}\mathbf{W}_1\mathbf{W}_2\|_F^2$



This problem can be solved efficiently using the singular value decomposition of $\mathbf{X}$.

SINGULAR VALUE DECOMPOSITION

Any matrix $\mathbf{X}$ can be decomposed as:



Where $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}$, and $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_d \geq 0$. I.e. $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices.

CONNECTION TO EIGENDECOMPOSITION

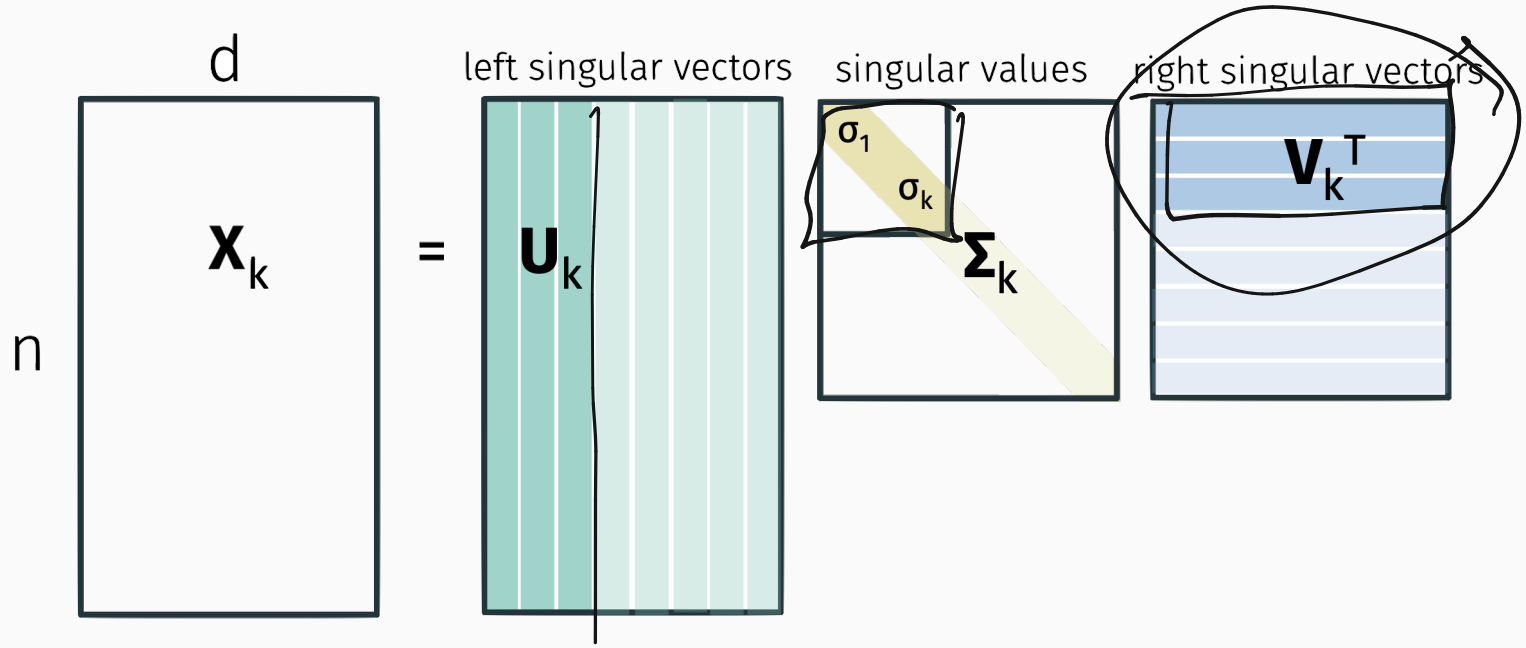
Recall that for a matrix $\mathbf{M} \in \mathbb{R}^{p \times p}$, $\mathbf{q}$ is an eigenvector of $\mathbf{M}$ if $\lambda \mathbf{q} = \mathbf{M}\mathbf{q}$ for any scalar λ.

- $\mathbf{U}$'s columns (the left singular vectors) are the orthonormal eigenvectors of $\mathbf{X}\mathbf{X}^T$.
- $\mathbf{V}$'s columns (the right singular vectors) are the orthonormal eigenvectors of $\mathbf{X}^T\mathbf{X}$.
- $\sigma_i^2 = \lambda_i(\mathbf{X}\mathbf{X}^T) = \lambda_i(\mathbf{X}^T\mathbf{X})$

Easy to check directly. This means you can use any (symmetric) eigensolver for computing the SVD.

SINGULAR VALUE DECOMPOSITION

```
U,S,V = scipy.sparse.linalg.svds(X, k).
```

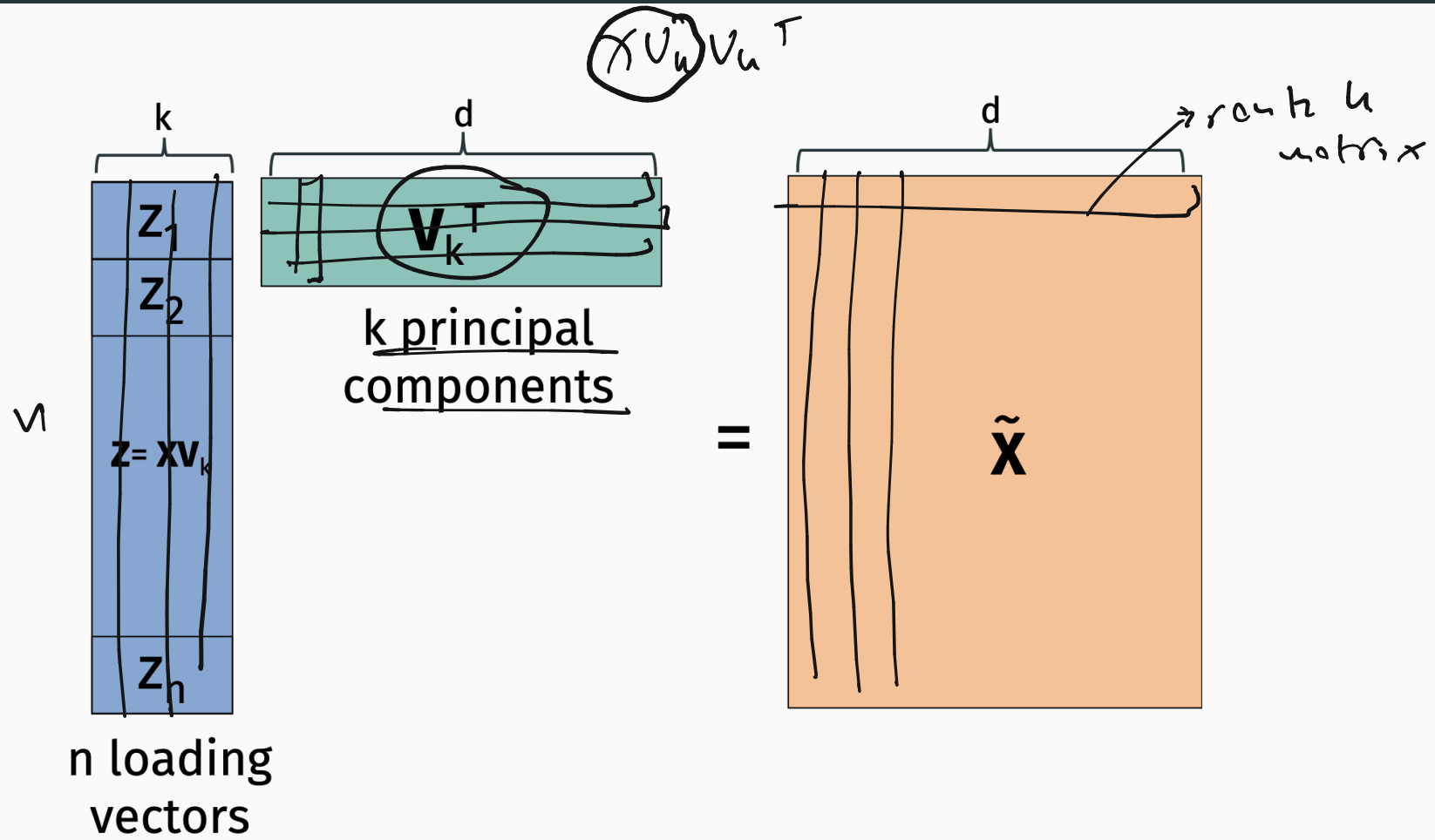


Eckart–Young–Mirsky Theorem: For any $k \leq d$, $\mathbf{X}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T = \mathbf{X}\mathbf{V}_k\mathbf{V}_k^T$ is the optimal k rank approximation to $\mathbf{X}$:

$$\mathbf{X}_k = \underset{\tilde{\mathbf{X}} \text{ with rank} \leq k}{\arg\min} \|\mathbf{X} - \tilde{\mathbf{X}}\|_F^2.$$

For PCA, set $\mathbf{W}_1 = \mathbf{V}_k$, $\mathbf{W}_2 = \mathbf{V}_k^T$.

PRINCIPAL COMPONENT ANALYSIS

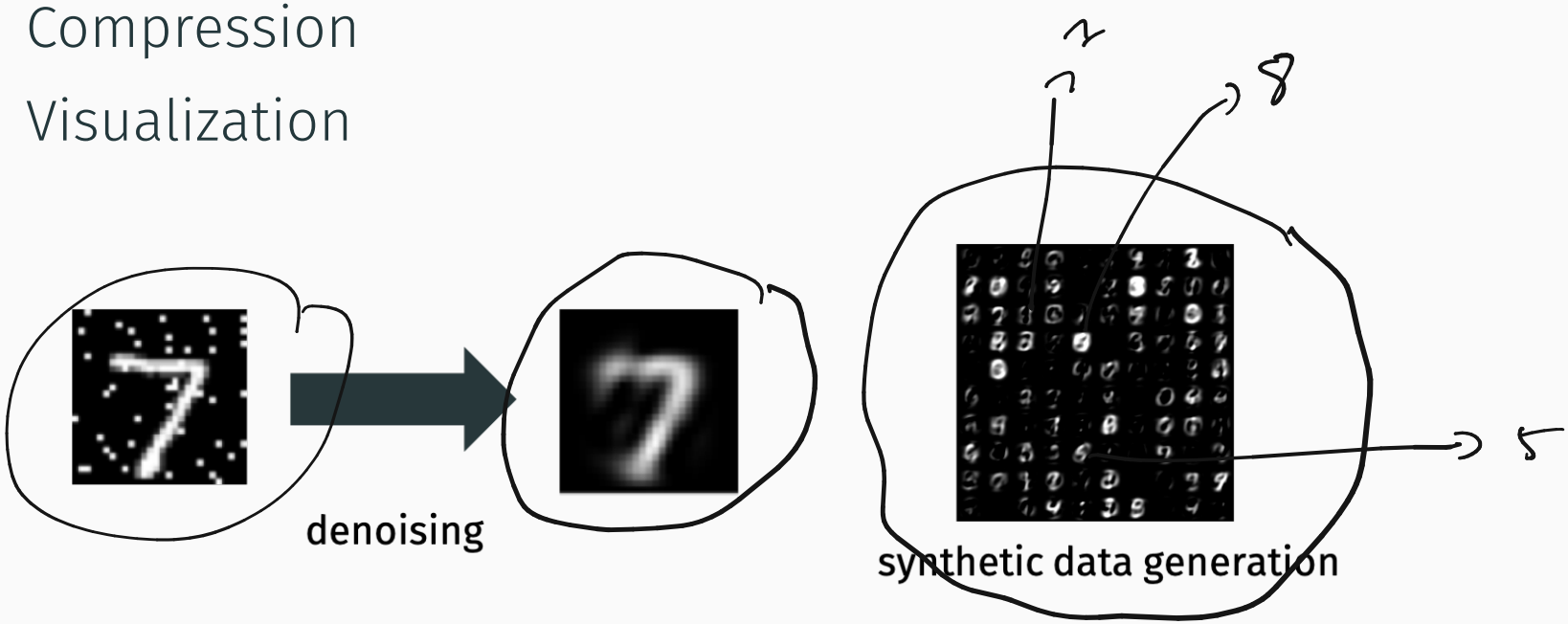


Usually $\vec{X}$'s columns (features) are mean centered and normalized to variance 1 before computing principal components.

PCA APPLICATIONS

Like any autoencoder, PCA can be used for:

- Feature extraction
- Denoising and rectification
- Data generation
- Compression
- Visualization

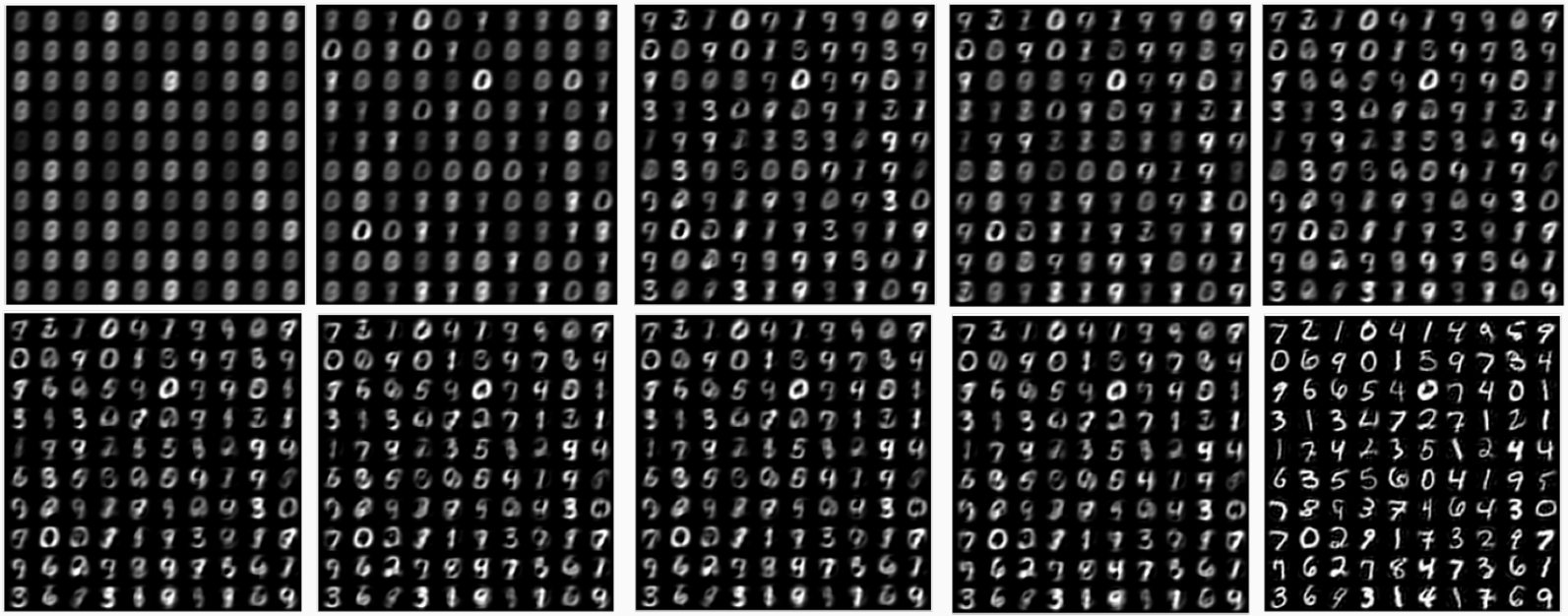


LOW-RANK APPROXIMATION

The larger we set k, the better approximation we get.

original data

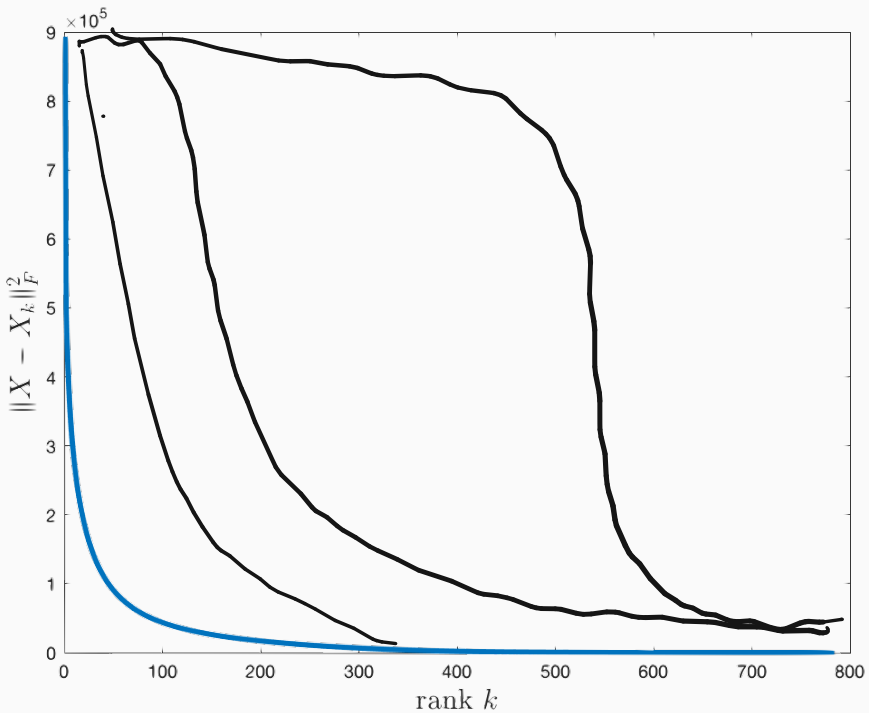
rank 1 approx. | rank 2 approx. | rank 3 approx. | rank 4 approx. | rank 5 approx.



rank 6 approx. | rank 7 approx. | rank 8 approx. | rank 9 approx. | rank 50 approx.

Error vs. k is dictated by $\mathbf{X}$'s singular values. This is often called the **spectrum** of $\mathbf{X}$.

$$\|\mathbf{X} - \mathbf{X}_k\|_F^2 = \sum_{i=k}^{d} \sigma_i^2.$$



LOW RANK INTUITION

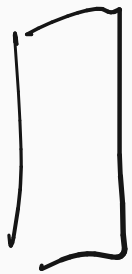
Which of these data sets has the better low-rank matrix:

1. House data:

$$\mathbf{x} = [\text{\# bedrooms}, \text{\# bathrooms}, \text{list price}, \text{sale price}, \text{property tax}]$$

2. Student data:

$$\mathbf{x} = [\text{gender}, \text{year}, \text{age}, \text{GPA}, \text{engineering major}]$$



Which of these data sets has the better low-rank matrix:

1. House data: 5 → 3

$$\tilde{\mathbf{x}} = [\#\text{ bedrooms}, \#\text{ bathrooms}, \text{list price}, \text{sale price}, \text{property tax}]$$

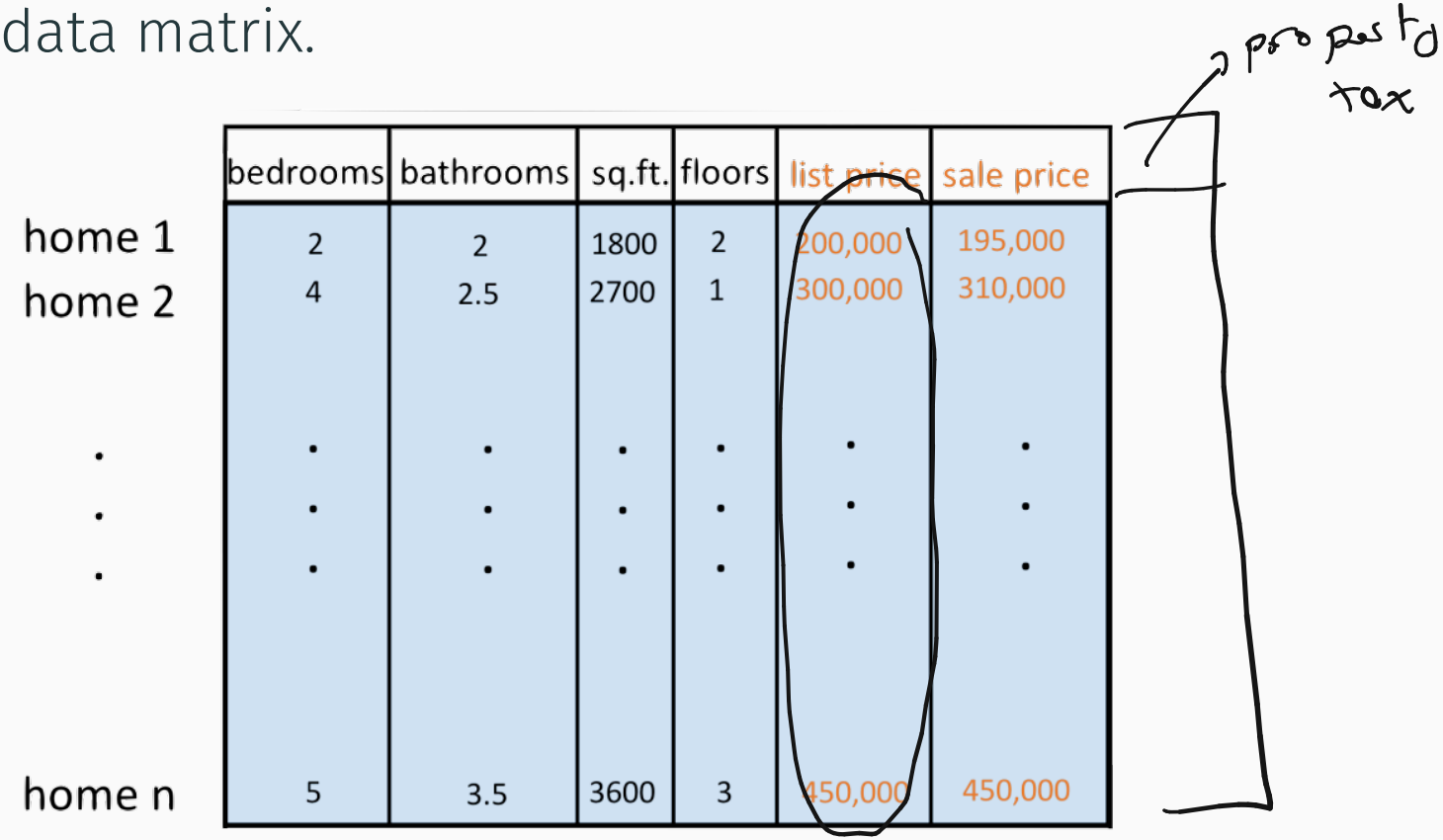
1 · list price .01 list price

2. Student data:

$$\mathbf{x} = [\text{gender}, \text{year}, \text{age}, \text{GPA}, \text{engineering major}]$$

COLUMN REDUNDANCY

Colinearity of data features leads to an approximately low-rank data matrix.



sale price $\approx 1.05 \cdot$ list price.

property tax $\approx .01 \cdot$ list price.

Sometimes these relationships are simple, other times more complex. But as long as there exists linear relationships between features, we will have a lower rank matrix.

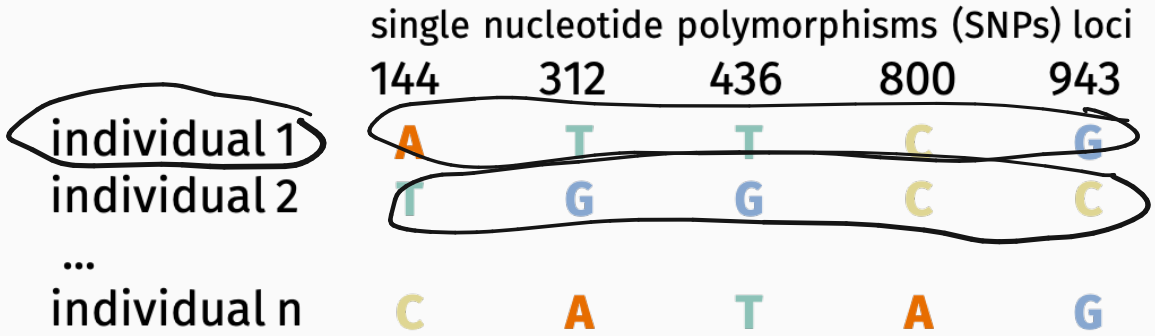
$$\text{yard size} \approx \text{lot size} - \frac{1}{2} \cdot \text{square footage}.$$

$$\begin{aligned}\text{cumulative GPA} \approx & \frac{1}{4} \cdot \text{year 1 GPA} + \frac{1}{4} \cdot \text{year 2 GPA} \\ & + \frac{1}{4} \cdot \text{year 3 GPA} + \frac{1}{4} \cdot \text{year 4 GPA}.\end{aligned}$$

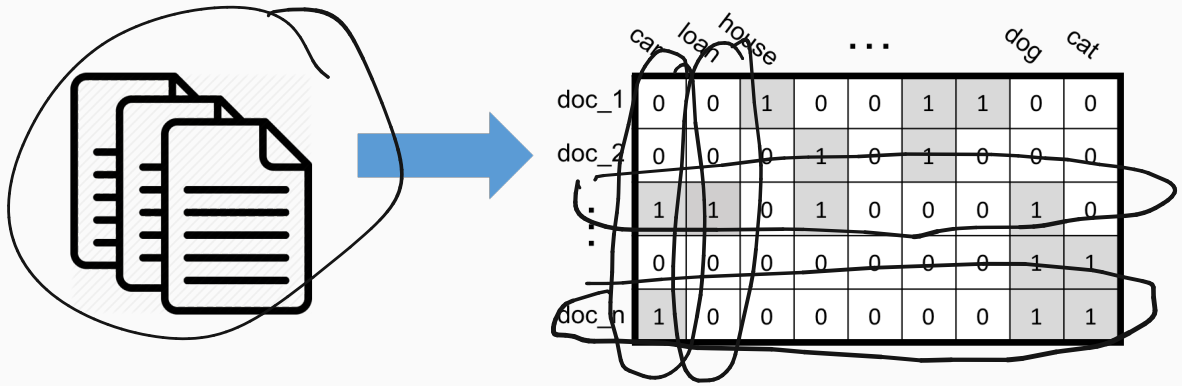
LOW-RANK INTUITION

Which of these data sets has a good low-rank matrix:

1. Genetic data:



2. "Term-document" matrix with bag-of-words data:

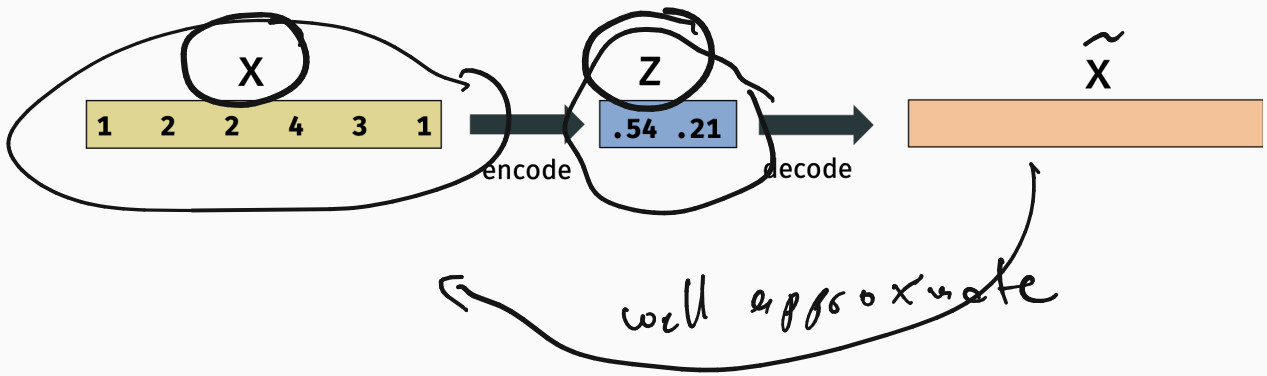


EXAMPLES OF LOW-RANK STRUCTURE

SNPs matrices tend to be very low-rank.

	single nucleotide polymorphisms (SNPs) loci				
	144	312	436	800	943
individual 1	A	T	T	C	G
individual 2	T	G	G	C	C
...					
individual n	C	A	T	A	G

Most of the information in $\vec{x}$ is explained by just a few **latent variable**.



SIMILARITY PRESERVATION

Very important note: Latent feature vectors preserve similarity and distance information in the original data.

Let $\vec{x}_1 \ldots, \vec{x}_n \in \mathbb{R}^d$ be our original data vectors. $\vec{z}_1 \ldots, \vec{z}_n \in \mathbb{R}^k$ be our loading vectors. $\tilde{x}_1 \ldots, \tilde{x}_n \in \mathbb{R}^d$ be our low-rank approximated data.

1. If we have a good low-rank approximation, we expect that $\vec{x}_i$ is close to $\tilde{x}_i$ for most i.
2. So for most i, j pairs, $\langle \vec{x}_i, \vec{x}_j \rangle \approx \langle \tilde{x}_i, \tilde{x}_j \rangle$ and $\|\vec{x}_i - \vec{x}_j\|_2 \approx \|\tilde{x}_i - \tilde{x}_j\|_2$.
3. $\langle \tilde{x}_i, \tilde{x}_j \rangle = \langle \mathbf{V}_k \vec{z}_i, \mathbf{V}_k \vec{z}_j \rangle$ since $\mathbf{V}_k$ is orthogonal. And $\|\tilde{x}_i - \tilde{x}_j\|_2 = \|\mathbf{V}_k \vec{z}_i - \mathbf{V}_k \vec{z}_j\|_2$.

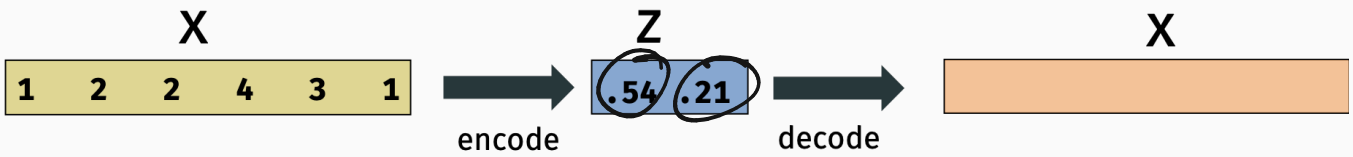
$= \|\vec{z}_i - \vec{z}_j\|_2$

$\vec{z}_i^T \mathbf{V}_k^T \mathbf{V}_k \vec{z}_j = \vec{z}_i^T \vec{z}_j$, where $\mathbf{V}_k^T \mathbf{V}_k = \mathbf{I}$

Conclusion: $\langle \vec{x}_i, \vec{x}_j \rangle \approx \langle \vec{z}_i, \vec{z}_j \rangle$ and $\|\vec{x}_i - \vec{x}_j\|_2 \approx \|\vec{z}_i - \vec{z}_j\|_2$.

EXAMPLES OF LOW-RANK STRUCTURE

"Genes Mirror Geography Within Europe" – Nature, 2008.



In data collected from European populations, latent variables capture information about geography.

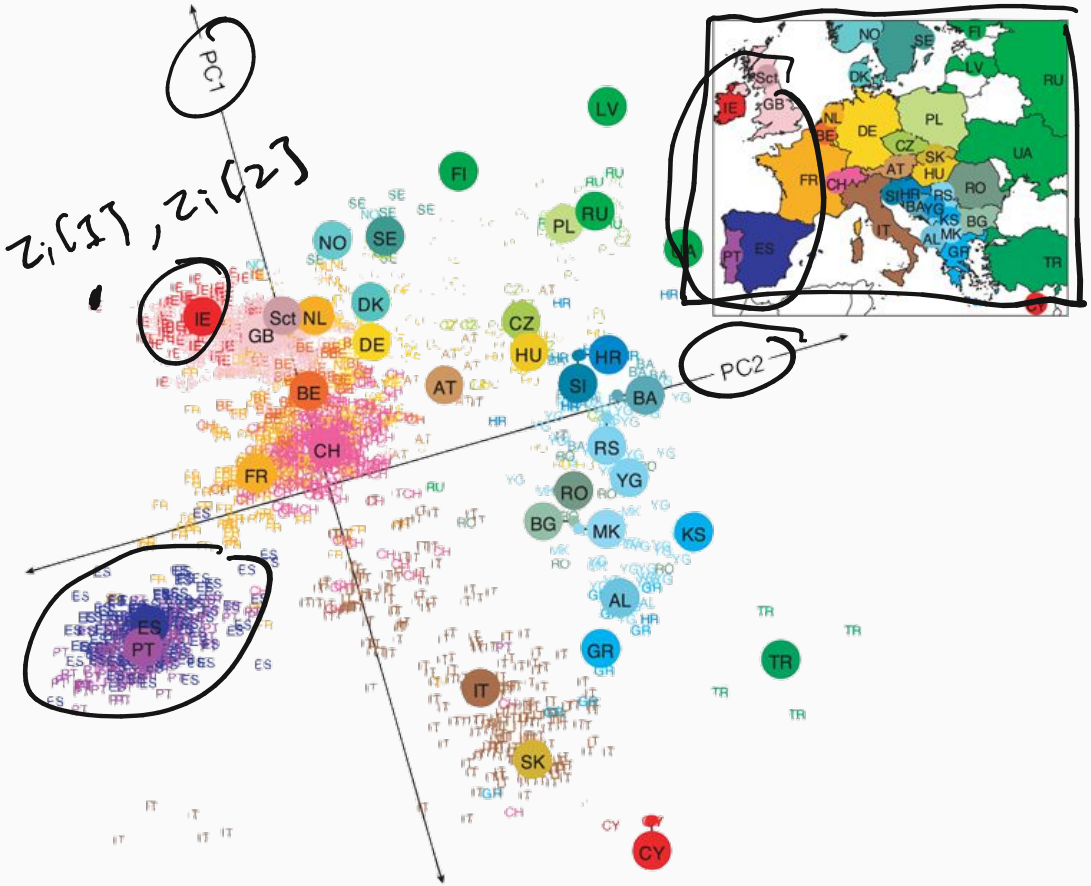
$\vec{z}[1] \approx$ relative north-south position of birth place

$\vec{z}[2] \approx$ relative east-west position of birth place

Individuals born in similar places tend to have similar genes.

PCA FOR DATA VISUALIZATION

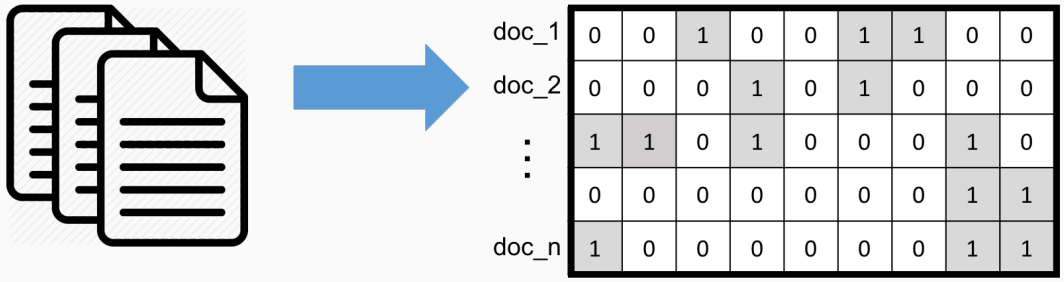
"Genes Mirror Geography Within Europe" – Nature, 2008.



Can be easily visualized using PCA! Plot each data example $\vec{x}$ using two loading variables in $\vec{z}$.

TERM DOCUMENT MATRIX

Word-document matrices tend to be low rank.

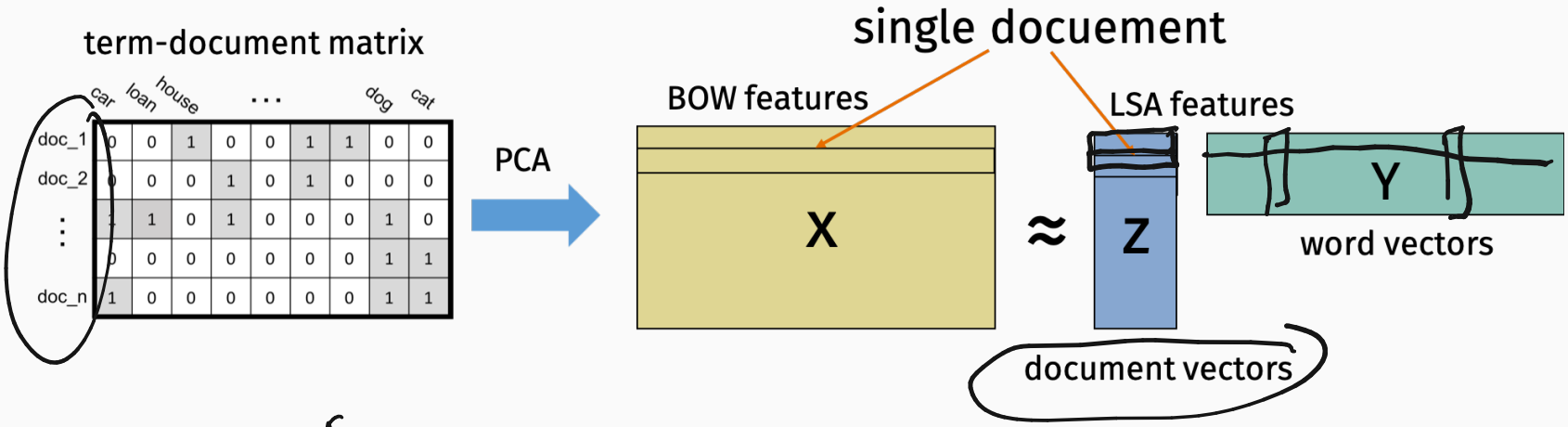


Documents tend to fall into a relatively small number of different categories, which use similar sets of words:

- **Financial news:** *markets, analysts, dow, rates, stocks*
- **US Politics:** *president, senate, pass, slams, twitter, media*
- **StackOverflow posts:** *python, help, convert, javascript*
- **Etc.**

LATENT SEMANTIC ANALYSIS

Latent semantic analysis = PCA applied to a word-document matrix (usually from a large corpus). One of the most fundamental techniques in **natural language processing** (NLP).



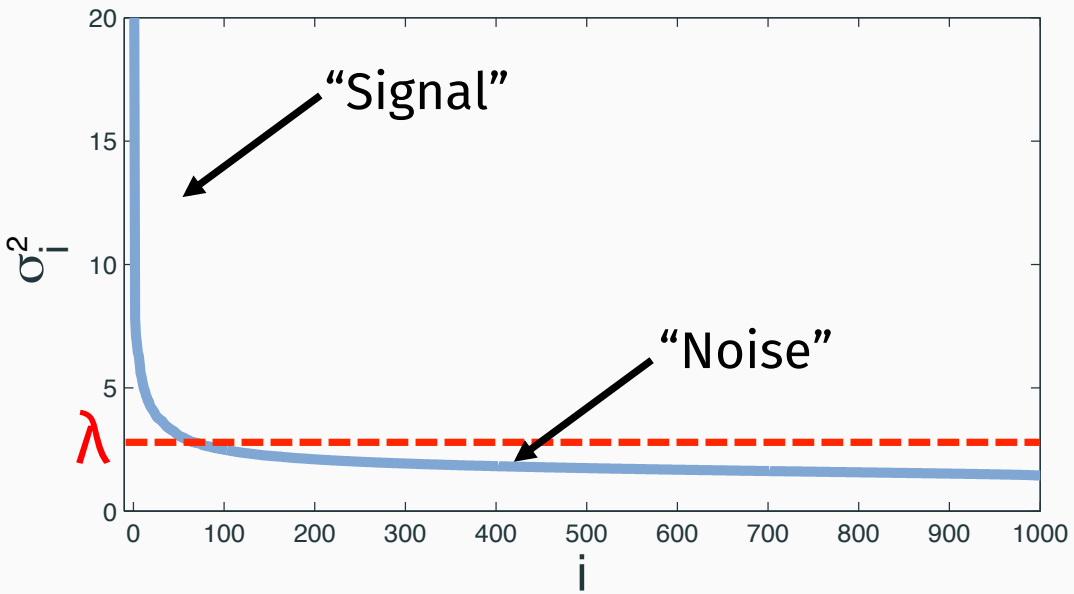
row of Z

Each ~~column of Z~~ corresponds to a latent "category" or "topic".

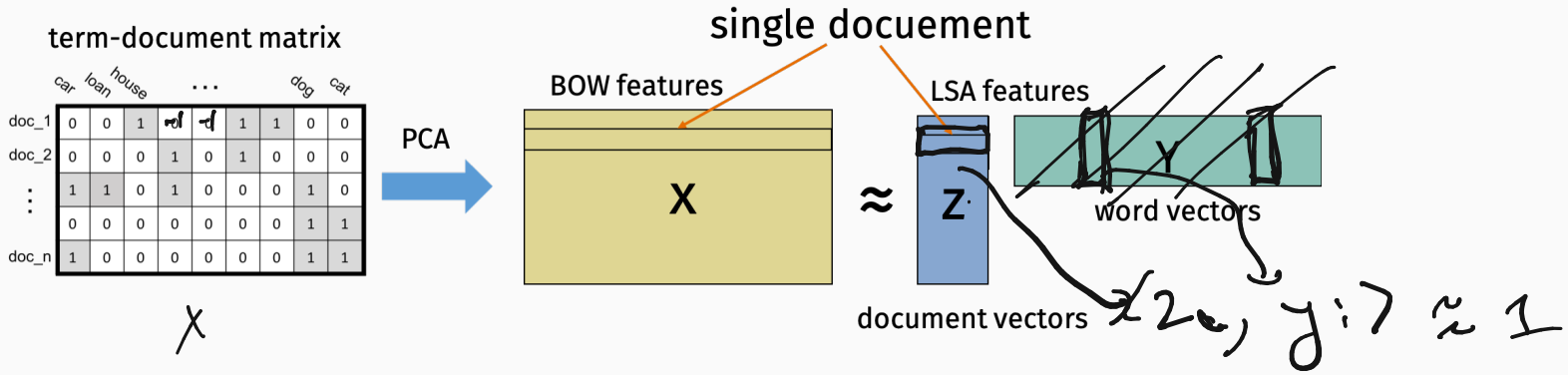
Similar documents have similar LSA document vectors. I.e. $\langle \vec{z}_i, \vec{z}_j \rangle$ is large. Provide a more compact "finger print" for documents than the long bag-of-words vectors. Useful for e.g search engines.

LSA vectors often provide a more meaningful similarity metric than bag-of-words vectors. Capture high-level categorical information and eliminate document specific quirks.

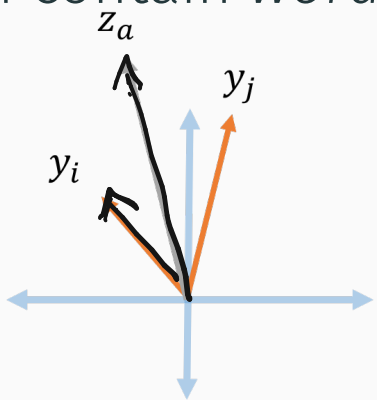
Spectrum of data matrix **X**.



WORD EMBEDDINGS

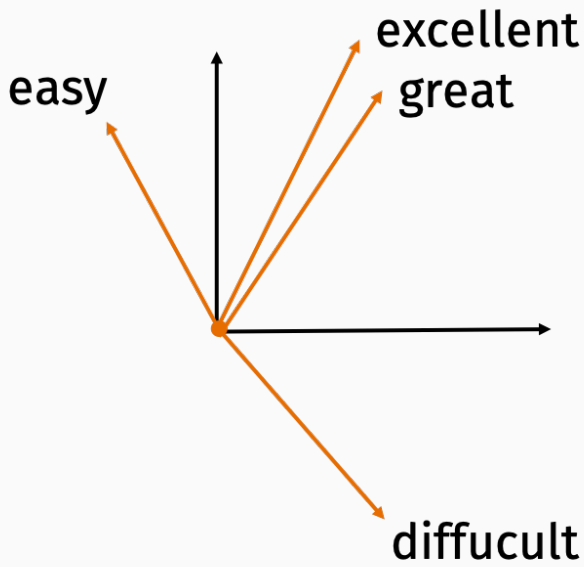


- $\langle \vec{y}_i, \vec{z}_a \rangle \approx 1$ when doc_i contains $word_a$.
- If doc_i and doc_j both contain $word_a$, $\langle \vec{y}_i, \vec{z}_a \rangle \approx \langle \vec{y}_j, \vec{z}_a \rangle \approx 1$.



If two words appear in the same document their, word vectors tend to point more in the same direction.

Result: Map words to numerical vectors in a semantically meaningful way. Similar words map to similar vectors. Dissimilar words to dissimilar vectors.



Extremely useful “side-effect” of LSA.

WORD EMBEDDINGS: MOTIVATING PROBLEM

Review 1: *Very small and handy for traveling or camping. Excellent quality, operation, and appearance.*

Review 2: *So far this thing is great. Well designed, compact, and easy to use. I'll never use another can opener.*

Review 3: *Not entirely sure this was worth* $20. *Mom couldn't figure out how to use it and it's fairly difficult to turn for someone with arthritis.*

Goal is to classify reviews as "positive" or "negative".

BAG-OF-WORDS FEATURES

Vocabulary: Small, handy, excellent, great, quality, compact, easy, difficult.

Review 1: *Very small and handy for traveling or camping. Excellent quality, operation, and appearance.*

[, , , , , , ,]

Review 2: *So far this thing is great. Well designed, compact, and easy to use. I'll never use another can opener.*

[, , , , , , ,]

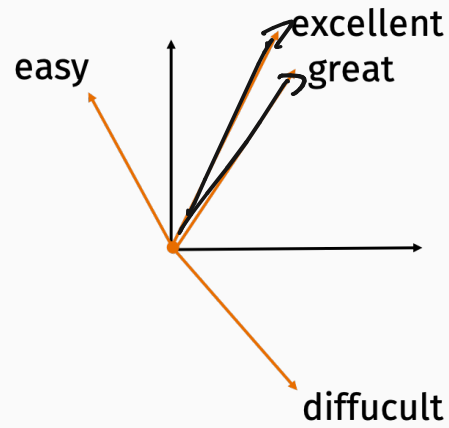
Review 3: *Not entirely sure this was worth* $20. *Mom couldn't figure out how to use it and it's fairly difficult to turn for someone with arthritis.*

[, , , , , , ,]

SEMANTIC EMBEDDINGS

Bag-of-words approach typically only works for large data sets.

The features do not capture the fact that “great” and “excellent” are near synonyms. Or that “difficult” and “easy” are antonyms.

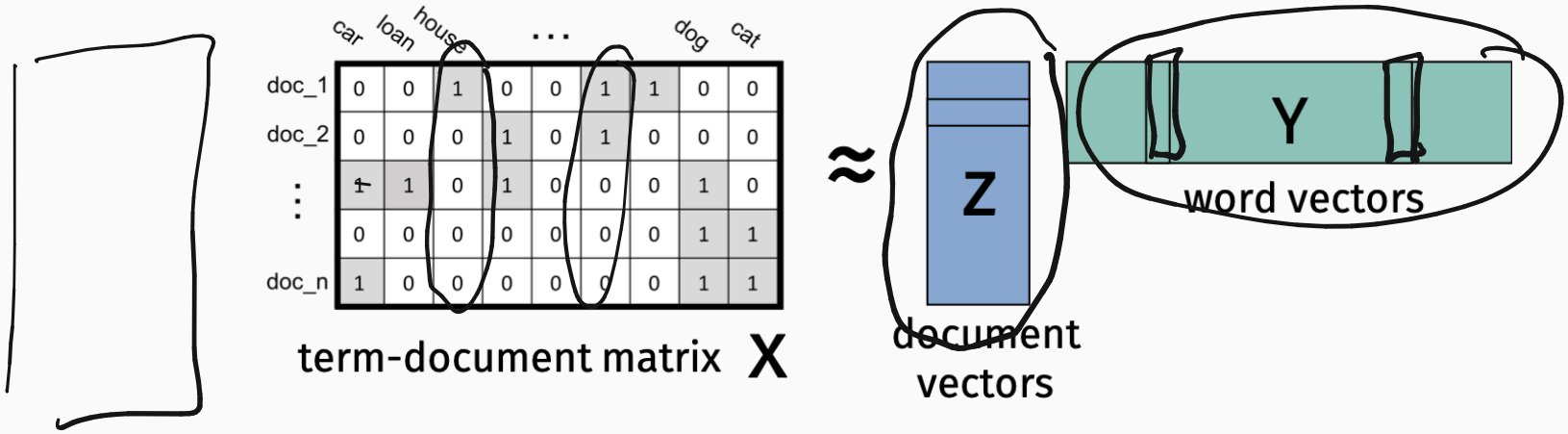


This can be addressed by first mapping words to semantically meaningful vectors. That mapping can be trained using a much large corpus of text than the data set you are working with (e.g. Wikipedia, Twitter, news data sets).

Current state of the art models: `GloVE`, `word2vec`.

- Based on same principal as LSA.
- `word2vec` was originally presented as a shallow neural network model, but it can be viewed as a matrix factorization method (Levy, Goldberg 2014).

Another view on word embeddings from LSA:



Choose Z to have orthogonal columns. E.g. $Z = U_k$ and $Y = \Sigma_k V_k^T$.

- $X \approx ZY$
- $X^T X \approx Y^T Z^T Z Y = Y^T Y$
- So for $word_i$ and $word_j$, $\langle \vec{y}_i, \vec{y}_j \rangle \approx [X^T X]_{i,j}$.

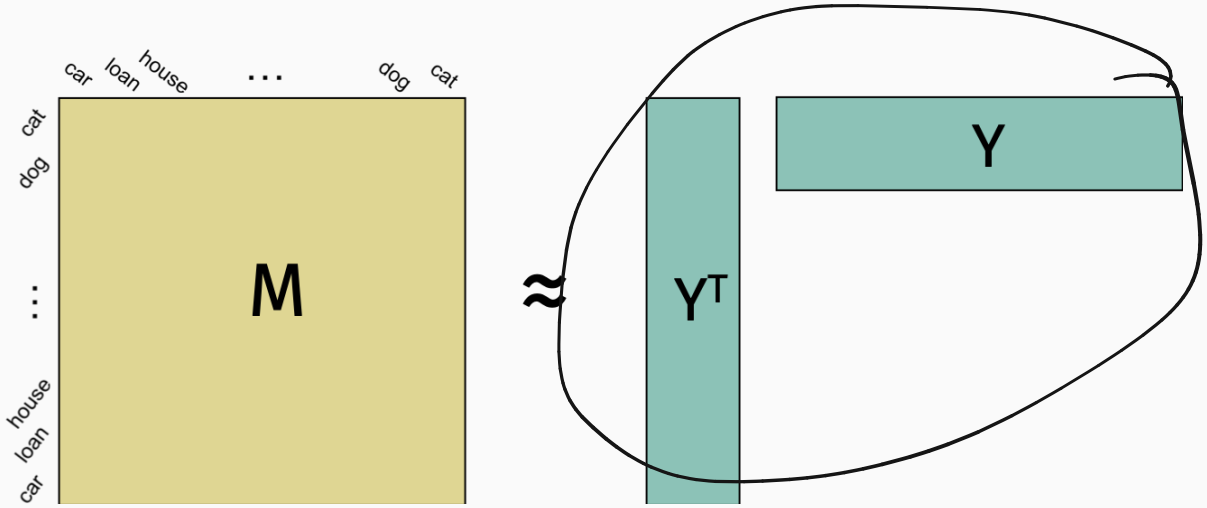
number of documents that both word i and word j appeared in

What does the i, j entry of $X^T X$ reprent?

$\langle \vec{y}_i, \vec{y}_j \rangle$ is larger if $word_i$ and $word_j$ appear in more documents together (high value in **word-word co-occurrence matrix**. Similarity of word embedding vectors mirrors similarity of word context.

General word embedding recipe:

1. Choose similarity metric $k(word_i, word_j)$ which can be computed for any pair of words.
2. Construct symmetric similarity matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ with $\mathbf{M}_{i,j} = k(word_i, word_j)$.
3. Find symmetric low rank factorization $\mathbf{M} \approx \mathbf{Y}^T\mathbf{Y}$ where $\mathbf{Y} \in \mathbb{R}^{k \times n}$.
4. Columns of $\mathbf{Y}$ are word embedding vectors.



How do current state-of-the-art methods differ from LSA?

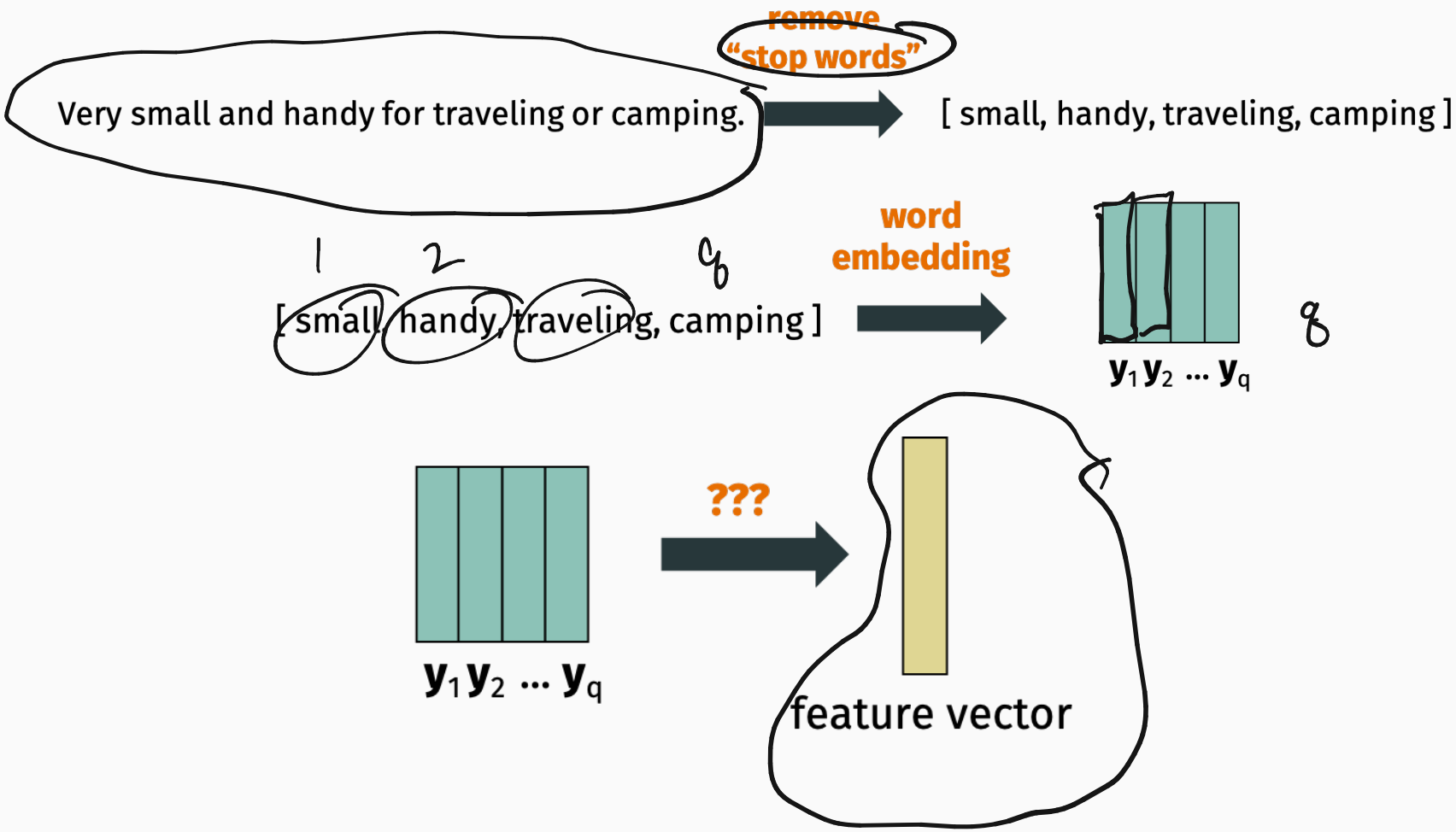
- Similarity based on co-occurrence in smaller chunks of words. E.g. in sentences or in any consecutive sequences of 10 words.
- Usually transformed in non-linear way. E.g. $k(word_i, word_j) = \frac{p(i,j)}{p(i)p(j)}$ where $p(i,j)$ is the frequency both i, j appeared together, and $p(i)$, $p(j)$ is the frequency either one appeared.

EASIEST WAY TO USE WORD EMBEDDINGS

If you want to use word embeddings for your project, the easiest approach is to download pre-trained word vectors:

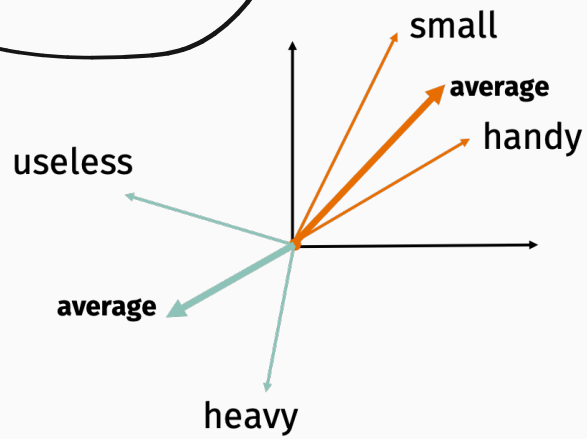
- Original gloVe website: `https://nlp.stanford.edu/projects/glove/`.
- Compilation of many sources: `https://github.com/3Top/word2vec-api`

How to go from word embeddings to features for a whole sentence or chunk of text?

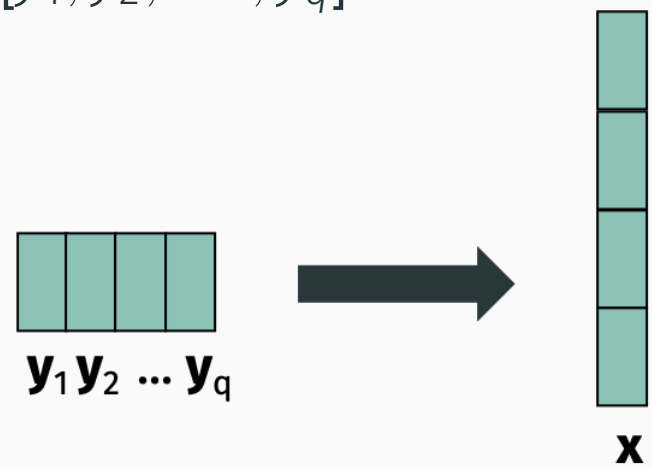


A few simple options:

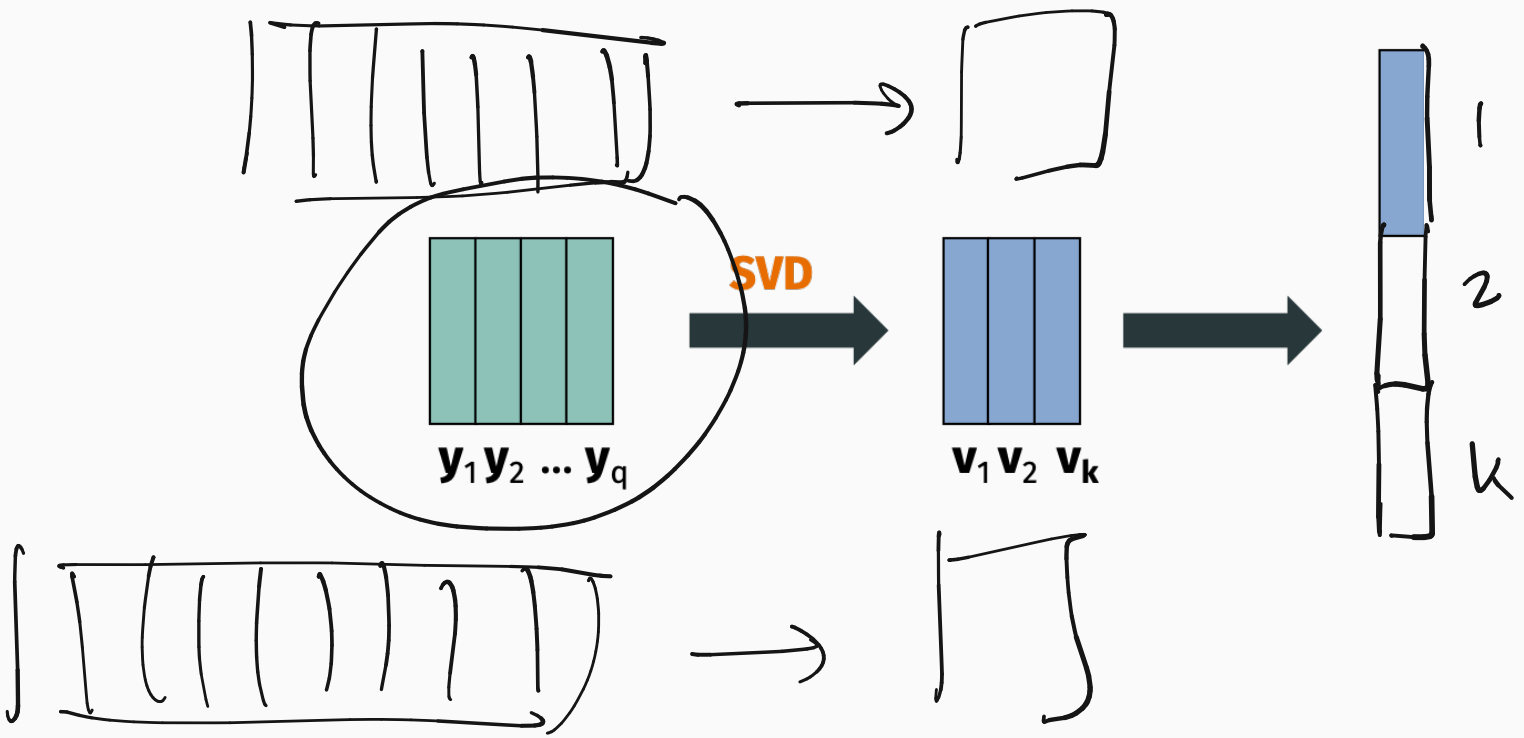
Feature vector $\vec{x} = \frac{1}{q} \sum_{i=1}^{q} \vec{y}_q$.



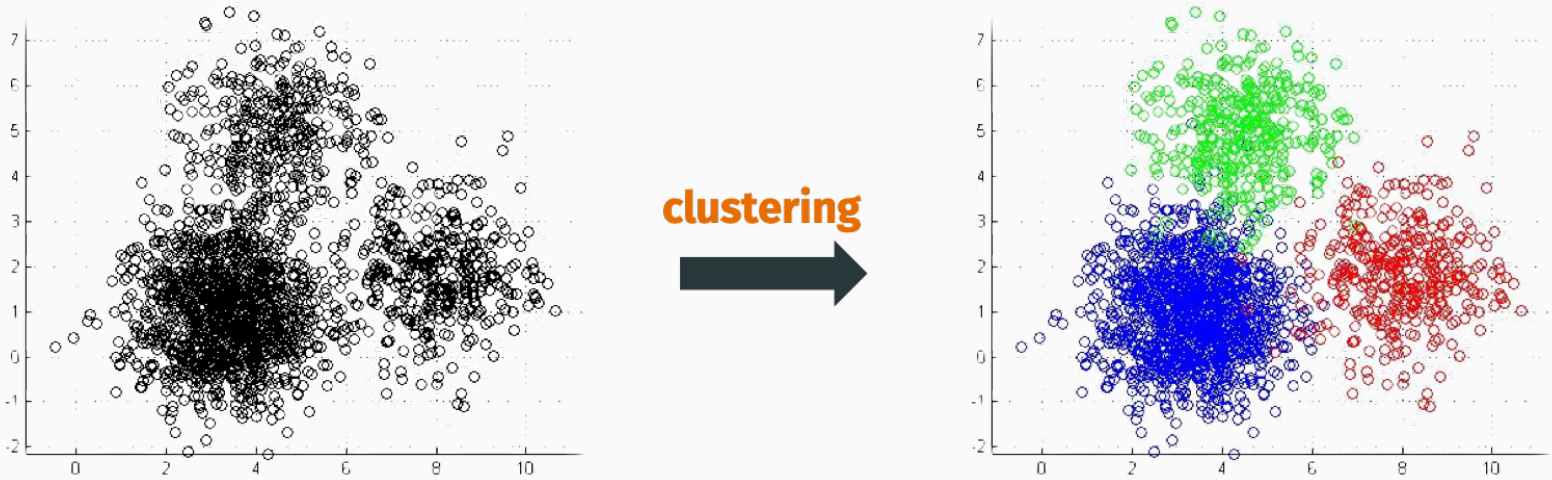
Feature vector $\vec{x} = [\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_q]$.



Better option than concatenation: To avoid issues with inconsistent sentence length, word ordering, etc. try concatenating a fixed number of top principal components of the matrix of word vectors:



Another important unsupervised learning task:

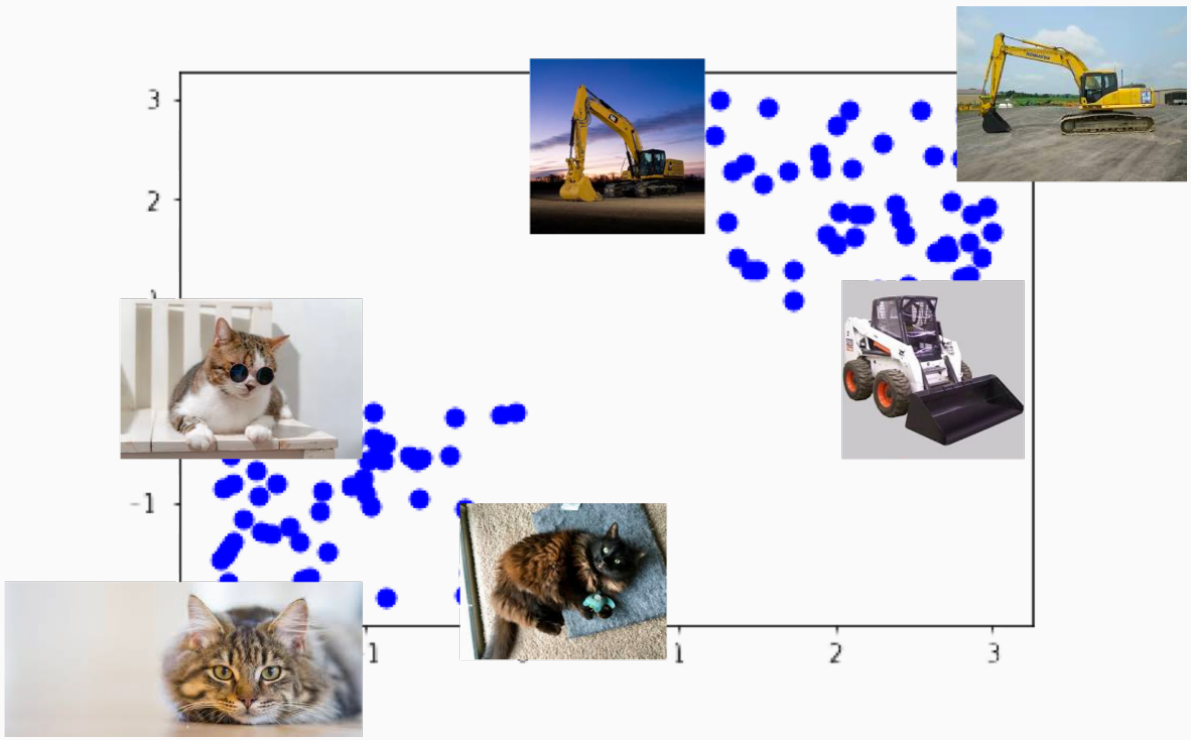


Separate unlabeled data into natural clusters.

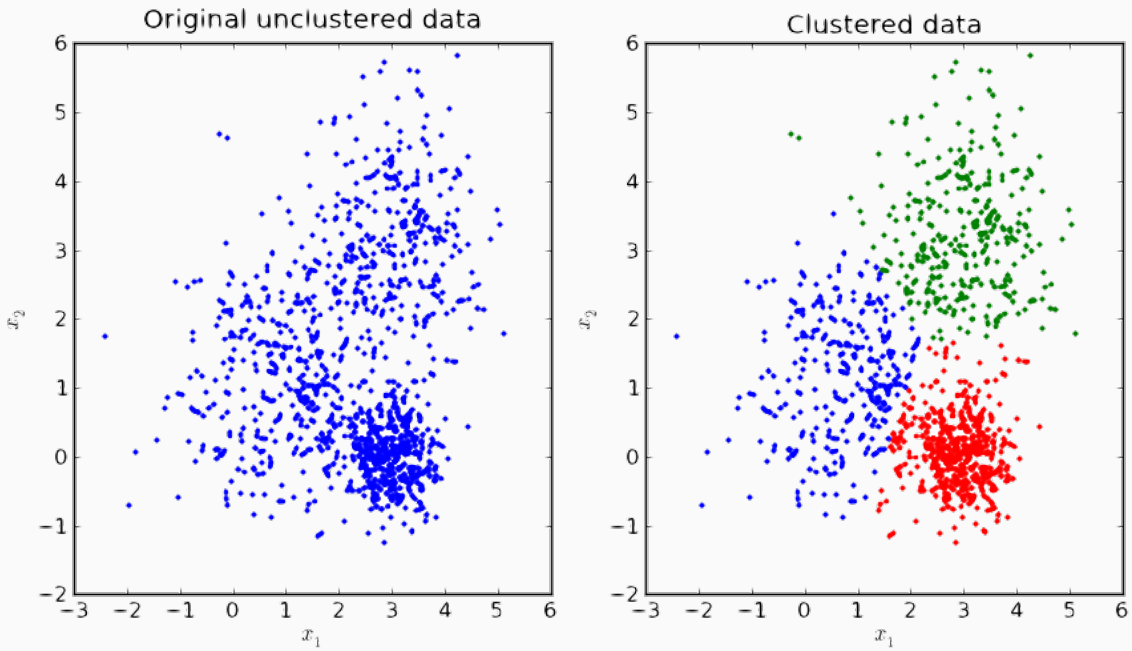
- Exploratory data analysis.
- Categorizing and grouping data.
- Visualizing data.

Example application:

Images of Cats.



Find sub-classes in your data which you may not have known about. Helps you decide how to adjust features or improve data set for a supervised application.



k-center clustering:

- Choose centers $\vec{c}_1, \ldots, \vec{c}_k \in \mathbb{R}^d$.
- Assign data point $\vec{x}$ to cluster i if $\vec{c}_i$ is the closest center to $\vec{x}$.