EECS 498-004: Introduction to Natural Language Processing

Instructor: Prof. Lu Wang
Computer Science and Engineering
University of Michigan
https://web.eecs.umich.edu/~wangluxy/

1

1

Two methods for getting short dense vectors

- Neural Language Model (word2vec)
- Singular Value Decomposition (SVD)

2

Two methods for getting short dense vectors

- Neural Language Model (word2vec)
- Singular Value Decomposition (SVD)

3

Sparse versus dense vectors

- Why dense vectors?
  - Short vectors may be easier to use as features in machine learning (less weights to tune)
  - Dense vectors may generalize better than storing explicit counts
  - They may do better at capturing synonymy:
    - *car* and *automobile* are synonyms; but are represented as distinct dimensions; this fails to capture similarity between a word with *car* as a neighbor and a word with *automobile* as a neighbor

4

Rank of a Matrix

- What is the rank of a matrix A?

5

Rank of a Matrix

- What is the rank of a matrix A?
- Number of linearly independent columns or rows of A

$$A = \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$$

6

Rank of a Matrix

- What is the rank of a matrix A?
- Number of linearly independent columns or rows of A

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$$

- Rank is 2
- We can rewrite A as two "basis" vectors: [1 2 1] [-2 -3 1]

7

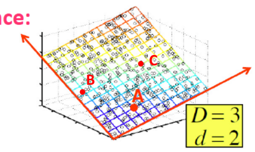
Rank as "Dimensionality"

Cloud of points 3D space:

- Think of point positions as a matrix:

$$\begin{array}{r c l} & \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix} & \begin{matrix} \mathbf{A} \\ \mathbf{B} \\ \mathbf{C} \end{matrix} \end{array}$$

1 row per point:



8

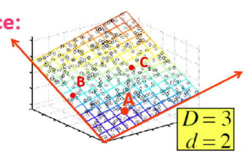
Rank as "Dimensionality"

Cloud of points 3D space:

- Think of point positions as a matrix:

$$\begin{array}{r c l} & \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix} & \begin{matrix} \mathbf{A} \\ \mathbf{B} \\ \mathbf{C} \end{matrix} \end{array}$$

1 row per point:



- Rewrite the coordinates in a more efficient way!
  - Old basis vectors: [1 0 0], [0 1 0], [0 0 1]
  - New basis vectors: [1 2 1], [-2 -3 1]

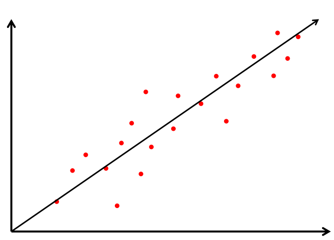
9

Intuition of Dimensionality Reduction

- Approximate an N-dimensional dataset using fewer dimensions
- By first rotating the axes into a new space
- In which the highest order dimension captures the most variance in the original dataset
- And the next dimension captures the next most variance, etc.

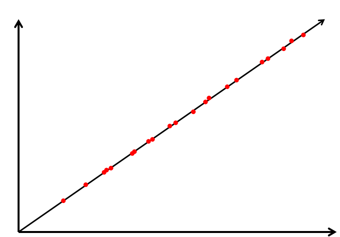
10

Sample Dimensionality Reduction



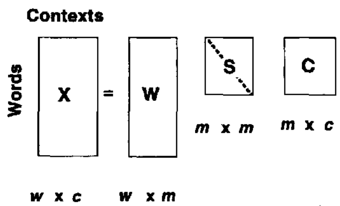
11

Sample Dimensionality Reduction



12

Singular Value Decomposition



(assuming the matrix has rank m, m<c)

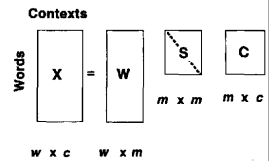
13

Singular Value Decomposition

*Any rectangular w x c matrix **X** equals the product of 3 matrices:*

W: rows corresponding to original but m columns represents a dimension in a new latent space, such that

- m column vectors are orthogonal to each other
- Columns are ordered by the amount of variance in the dataset each new dimension accounts for



14

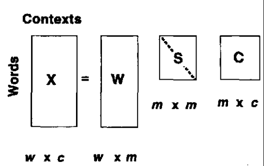
Singular Value Decomposition

*Any rectangular w x c matrix **X** equals the product of 3 matrices:*

W: rows corresponding to original but m columns represents a dimension in a new latent space, such that

- m column vectors are orthogonal to each other
- Columns are ordered by the amount of variance in the dataset each new dimension accounts for

S: diagonal *m* x *m* matrix of **singular values** expressing the importance of each dimension.



15

Singular Value Decomposition

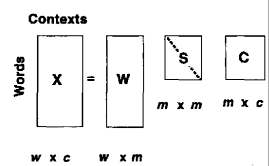
*Any rectangular w x c matrix **X** equals the product of 3 matrices:*

W: rows corresponding to original but m columns represents a dimension in a new latent space, such that

- m column vectors are orthogonal to each other
- Columns are ordered by the amount of variance in the dataset each new dimension accounts for

S: diagonal *m* x *m* matrix of **singular values** expressing the importance of each dimension.

C: columns corresponding to original but m rows corresponding to singular values



16

Singular Value Decomposition

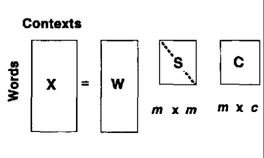
*Any rectangular w x c matrix **X** equals the product of 3 matrices:*

W: rows corresponding to original but m columns represents a dimension in a new latent space, such that

- m column vectors are orthogonal to each other
- Columns are ordered by the amount of variance in the dataset each new dimension accounts for

S: diagonal *m* x *m* matrix of **singular values** expressing the importance of each dimension.

C: columns corresponding to original but m rows corresponding to singular values

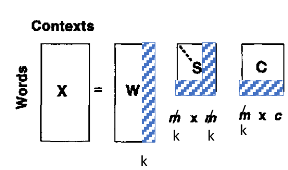


Existing tools from Python, MATLAB, R, etc, for SVD

17

SVD applied to word-document matrix

- If instead of keeping all m dimensions, we just keep the top k singular values. Let's say 300.
- Each row of W (keeping k columns of the original W):
  - A k-dimensional vector
  - Representing word w



18

SVD on Word-Document Matrix: Example

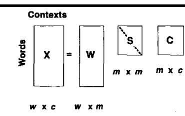
- The matrix X

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

19

Matrix **W**

	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09



Matrix **S**

	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

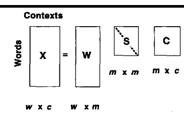
Matrix **C**

	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

20

Matrix **W**

	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09



Matrix **S**

	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

Matrix **C**

	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

21

Reduce dimension: The Matrix W

	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

	1	2	3	4	5
ship	−0.44	−0.30	0.00	0.00	0.00
boat	−0.13	−0.33	0.00	0.00	0.00
ocean	−0.48	−0.51	0.00	0.00	0.00
wood	−0.70	0.35	0.00	0.00	0.00
tree	−0.26	0.65	0.00	0.00	0.00

22

Reduce dimension: The Matrix S

1	2	3	4	5
2.16	0.00	0.00	0.00	0.00
0.00	1.59	0.00	0.00	0.00
0.00	0.00	1.28	0.00	0.00
0.00	0.00	0.00	1.00	0.00
0.00	0.00	0.00	0.00	0.39

1	2	3	4	5
2.16	0.00	0.00	0.00	0.00
0.00	1.59	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00

23

Reduce dimension: The Matrix C

d_1	d_2	d_3	d_4	d_5	d_6
−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
−0.29	−0.53	−0.19	0.63	0.22	0.41
0.28	−0.75	0.45	−0.20	0.12	−0.33
0.00	0.00	0.58	0.00	−0.58	0.58
−0.53	0.29	0.63	0.19	0.41	−0.22

d_1	d_2	d_3	d_4	d_5	d_6
−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
−0.29	−0.53	−0.19	0.63	0.22	0.41
0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00

24

Reduce dimension: The Matrix W

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

	1	2	3	4	5
ship	−0.44	−0.30	0.00	0.00	0.00
boat	−0.13	−0.33	0.00	0.00	0.00
ocean	−0.48	−0.51	0.00	0.00	0.00
wood	−0.70	0.35	0.00	0.00	0.00
tree	−0.26	0.65	0.00	0.00	0.00

25

Reduce dimension: The Matrix W

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

	1	2
ship	−0.44	−0.30
boat	−0.13	−0.33
ocean	−0.48	−0.51
wood	−0.70	0.35
tree	−0.26	0.65

26

Reduce dimension: The Matrix W

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

	1	2
ship	−0.44	−0.30
boat	−0.13	−0.33
ocean	−0.48	−0.51
wood	−0.70	0.35
tree	−0.26	0.65

Similarity between ***ship*** and ***boat*** ***vs*** ***ship*** and ***wood*** ?

27

Reduce dimension: The Matrix W

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

	1	2
ship	−0.44	−0.30
boat	−0.13	−0.33
ocean	−0.48	−0.51
wood	−0.70	0.35
tree	−0.26	0.65

28

More details

- 300 dimensions are commonly used
- The cells are commonly weighted by a product of two weights (TF-IDF)
  - Local weight: term frequency (or log version)
  - Global weight: idf

29

Let's return to PPMI word-word matrices

- Can we apply SVD to them?

30

SVD applied to term-term matrix

$$\underset{|V|\times|V|}{\begin{bmatrix} \\ X \\ \\ \end{bmatrix}} = \underset{|V|\times|V|}{\begin{bmatrix} \\ W \\ \\ \end{bmatrix}} \underset{|V|\times|V|}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_V \end{bmatrix}} \underset{|V|\times|V|}{\begin{bmatrix} \\ C \\ \\ \end{bmatrix}}$$

(assuming the matrix has rank |V|, may not be true)

31

SVD applied to term-term matrix

$$\underset{|V|\times|V|}{\begin{bmatrix} \\ X \\ \\ \end{bmatrix}} = \underset{|V|\times|V|}{\begin{bmatrix} \\ W \\ \\ \end{bmatrix}} \underset{|V|\times|V|}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_V \end{bmatrix}} \underset{|V|\times|V|}{\begin{bmatrix} \\ C \\ \\ \end{bmatrix}}$$

(assuming the matrix has rank |V|, may not be true)

32

Truncated SVD on term-term matrix

$$\underset{|V|\times|V|}{\begin{bmatrix} \\ X \\ \\ \end{bmatrix}} = \underset{|V|\times k}{\begin{bmatrix} \\ W \\ \\ \end{bmatrix}} \underset{k\times k}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_k \end{bmatrix}} \underset{k\times|V|}{\begin{bmatrix} C \end{bmatrix}}$$

33

Truncated SVD produces embeddings

- Each row of W matrix is a k-dimensional representation of each word *w*
- k might range from 50 to 1000
- Generally we keep the top k dimensions, but some experiments suggest that getting rid of the top 1 dimension or even the top 50 dimensions is helpful (Lapesa and Evert 2014).

embedding for word i

$$\underset{|V|\times k}{\begin{bmatrix} \\ W \\ \\ \end{bmatrix}}$$

34

Embeddings versus sparse vectors

- Dense SVD embeddings sometimes work better than sparse PPMI matrices at tasks like word similarity
  - Denoising: low-order dimensions may represent unimportant information
  - Truncation may help the models generalize better to unseen data.
  - Having a smaller number of dimensions may make it easier for classifiers to properly weight the dimensions for the task.
  - Dense models may do better at capturing higher order co-occurrence.

35