

# Copyright Notice

These slides are distributed under the Creative Commons License.

[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see <https://creativecommons.org/licenses/by-sa/2.0/legalcode>



deeplearning.ai

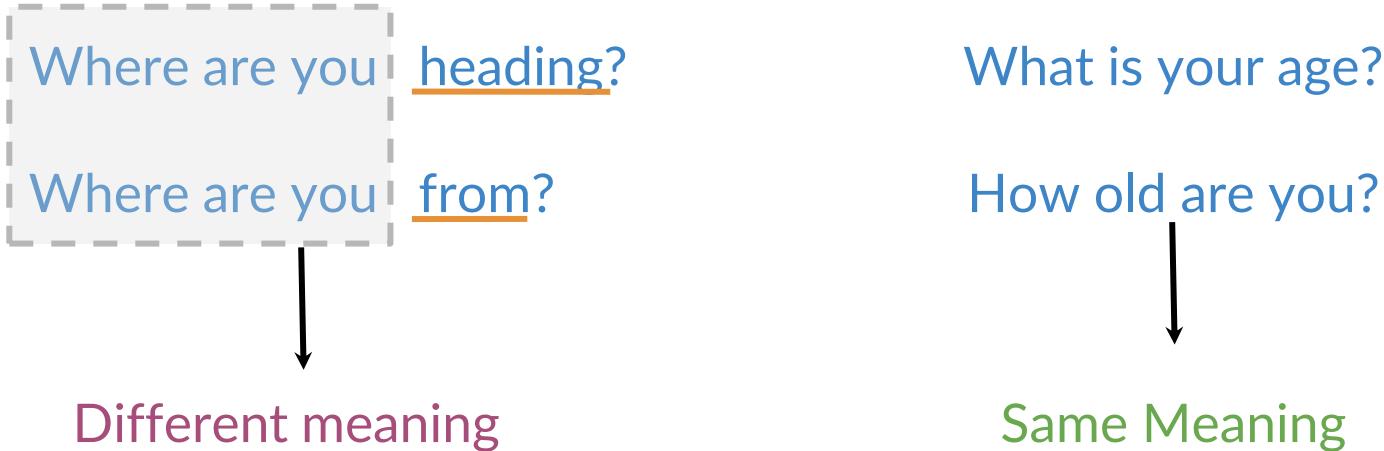
# Vector Space Models

---

# Outline

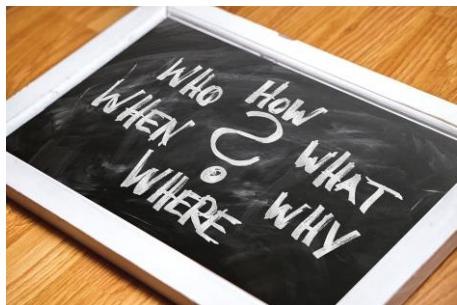
- Vector space models
- Advantages
- Applications

# Why learn vector space models?



# Vector space models applications

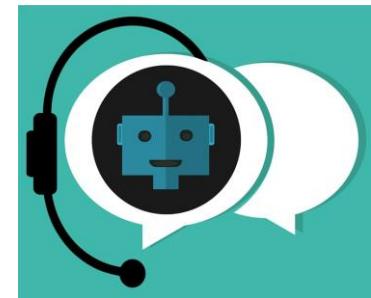
- You eat cereal from a bowl
- You buy something and someone else sells it



Information Extraction



Machine Translation



Chatbots

# Fundamental concept

“You shall know a word by the company it keeps”

Firth, 1957



(Firth, J. R. 1957:11)

# Summary

- Represent words and documents as **vectors**
- Representation that **captures** relative **meaning**



deeplearning.ai

Word by Word  
and Word by  
Doc.  

---

# Outline

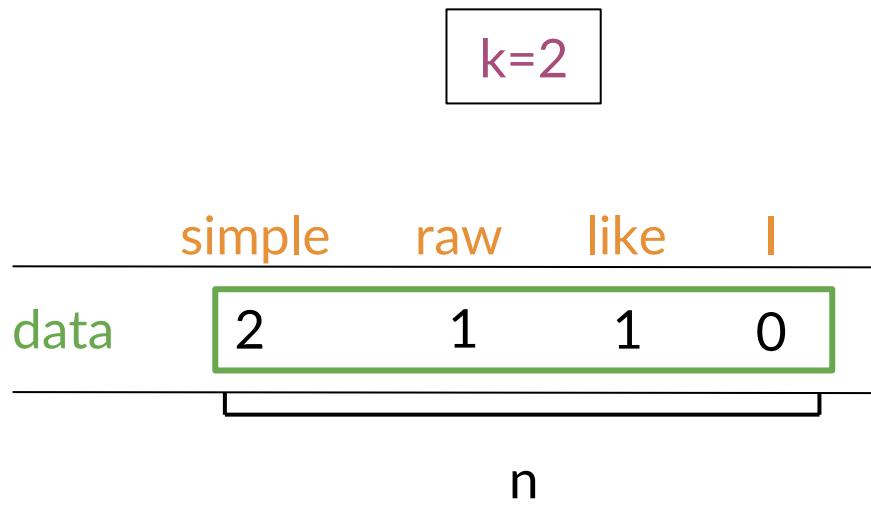
- Co-occurrence ————— Vector representation
- Relationships between words/documents

# Word by Word Design

Number of times they *occur together within a certain distance  $k$*

I like simple data  
I prefer simple raw data

$k=2$



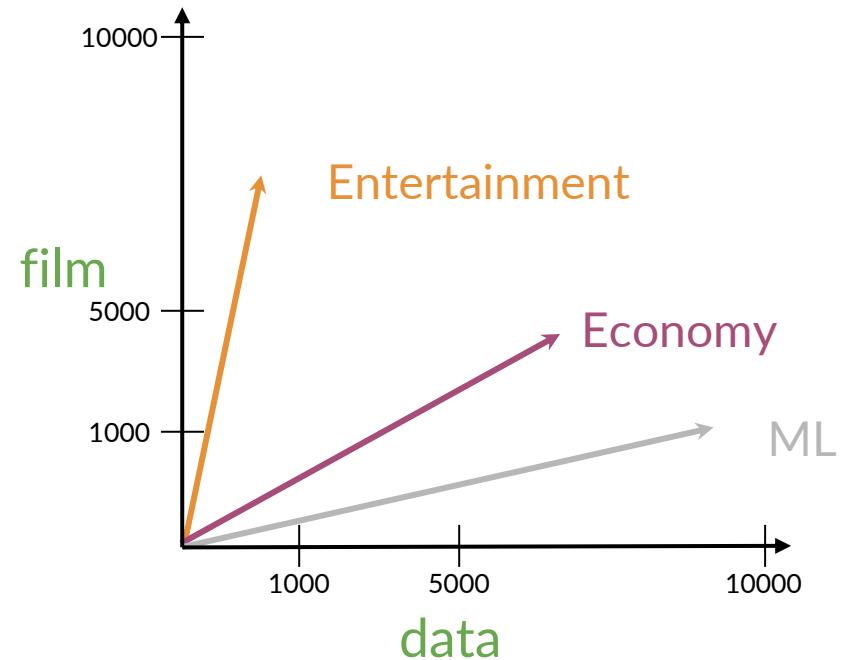
# Word by Document Design

Number of times a word *occurs within a certain category*

	Entertainment	Economy	Machine Learning
data	500	6620	9320
film	7000	4000	1000

*Handwritten annotations:*  
The word "Entertainment" has "ent" written above it.  
The word "Economy" has "econ" written above it.  
The word "Machine Learning" has "ML" written above it.

# Vector Space



	Entertainment	Economy	ML
data	500	6620	9320
film	7000	4000	1000

Measures of “similarity”:  
Angle  
Distance

# Summary

- W/W and W/D, *counts* of occurrence
- Vector Spaces → Similarity between words/documents



deeplearning.ai

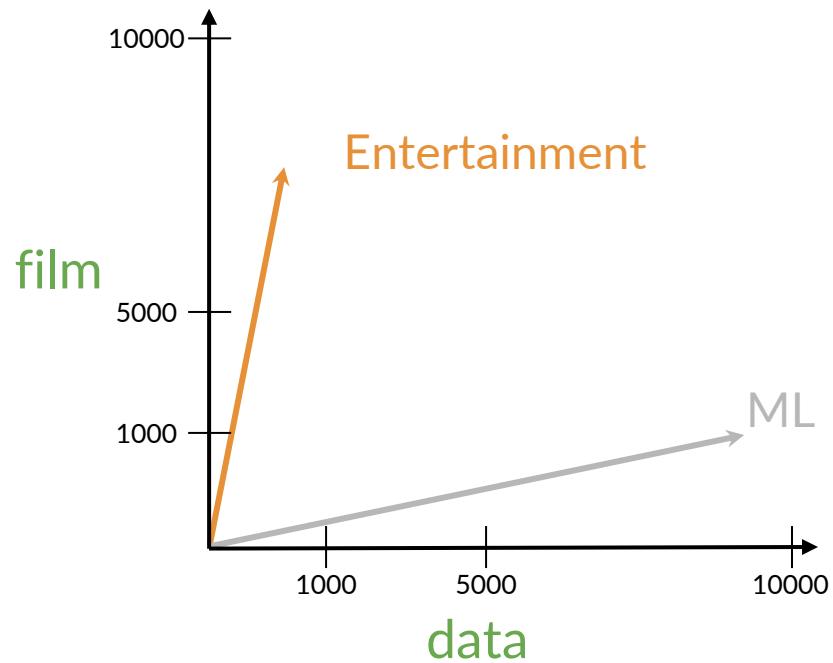
# Euclidean Distance

---

# Outline

- Euclidean distance
- N-dimension vector representations comparison

# Euclidean distance

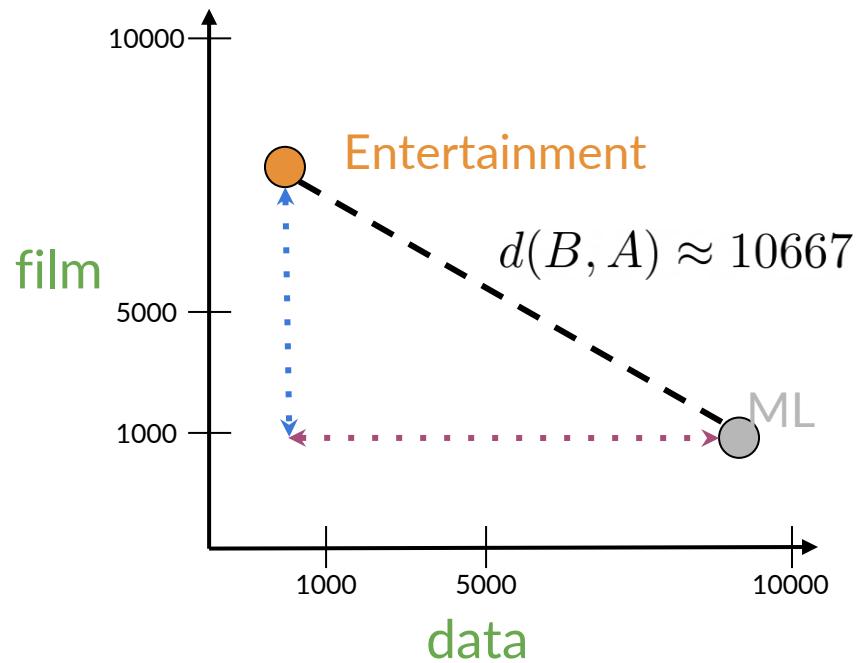


Corpus A: (500,7000)



Corpus B: (9320,1000)

# Euclidean distance



Corpus A: (500,7000)



Corpus B: (9320,1000)

fcr each dim

$$d(B, A) = \sqrt{(B_1 - A_1)^2 + (B_2 - A_2)^2}$$
$$c^2 = a^2 + b^2$$

$$d(B, A) = \sqrt{(-8820)^2 + (6000)^2}$$

# Euclidean distance for n-dimensional vectors

	data	$\vec{w}$	data	$\vec{v}$
AI	6	0	1	
drinks	0	4	6	
food	0	6	8	

$$\begin{aligned} & \text{dim1} \quad \text{dim2} \quad \text{dim3} \\ &= \sqrt{(1 - 0)^2 + (6 - 4)^2 + (8 - 6)^2} \\ &= \sqrt{1 + 4 + 4} = \sqrt{9} = 3 \end{aligned}$$

$$d(\vec{v}, \vec{w}) = \sqrt{\sum_{i=1}^n (v_i - w_i)^2} \longrightarrow \text{Norm of } (\vec{v} \cdot \vec{w})$$

# Euclidean distance in Python

```
# Create numpy vectors v and w
v = np.array([1, 6, 8])
w = np.array([0, 4, 6])

# Calculate the Euclidean distance d
d = np.linalg.norm(v-w)

# Print the result
print("The Euclidean distance between v and w is: ", d)
```

The Euclidean distance between v and w is: 3

# Summary

- Straight line between points
- Norm of the difference between vectors



deeplearning.ai

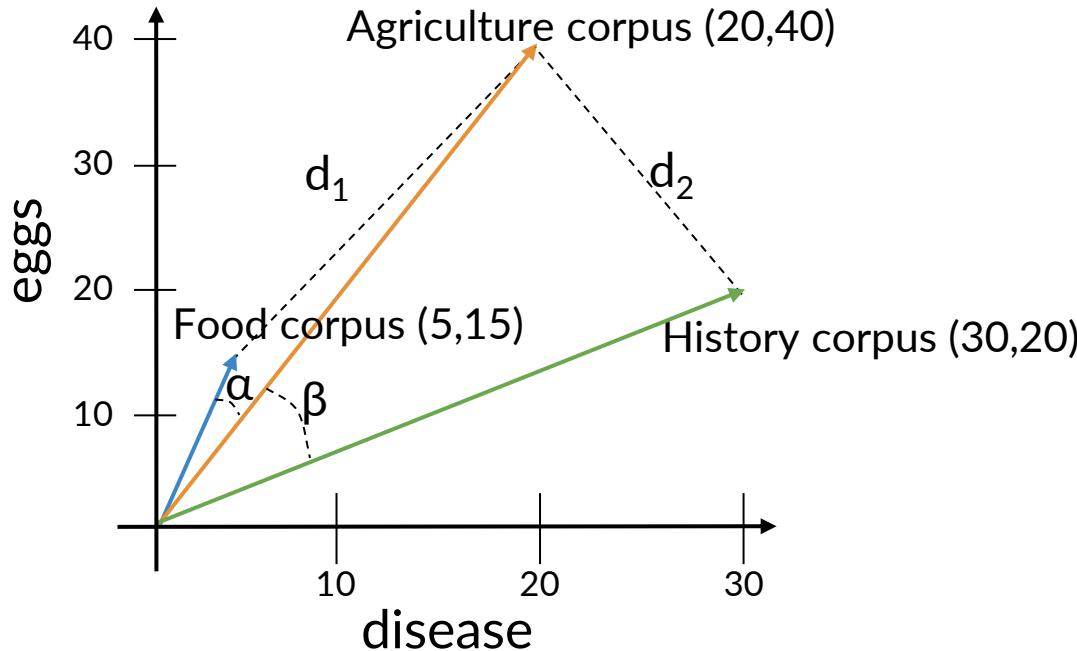
# Cosine Similarity: Intuition

---

# Outline

- Problems with Euclidean Distance
- Cosine similarity

# Euclidean distance vs Cosine similarity



angle larger ( $90^\circ$ )  
Cos closer to zero

Euclidean distance:  $d_2 < d_1$   
Angles comparison:  $\beta > \alpha$

The cosine of the angle  
between the vectors

# Summary

- Cosine similarity when corpora are different sizes



deeplearning.ai

# Cosine Similarity

---

# Outline

- How to get the cosine of the angle between two vectors
- Relation of this metric to similarity

# Previous definitions

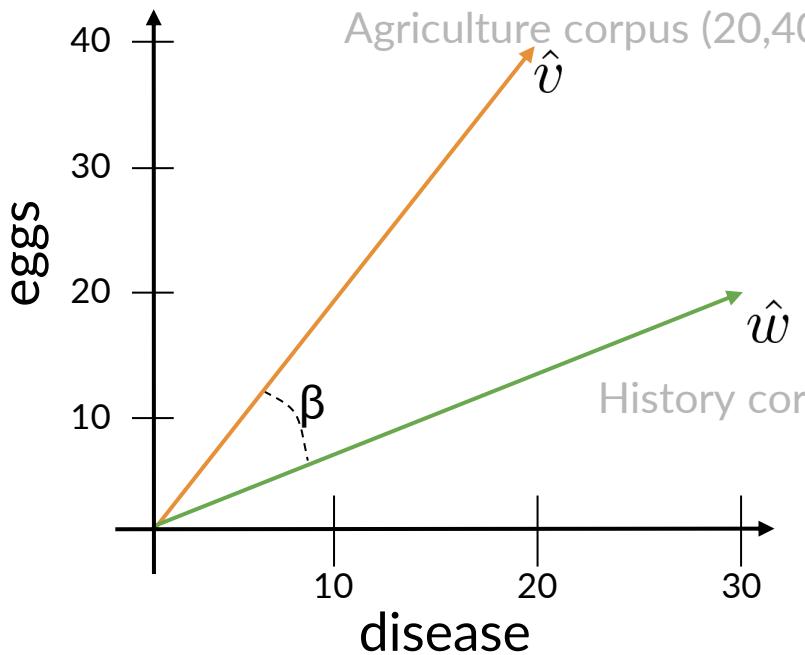
Vector norm

$$\|\vec{v}\| = \sqrt{\sum_{i=1}^n v_i^2}$$

Dot product

$$\vec{v} \cdot \vec{w} = \sum_{i=1}^n v_i \cdot w_i$$

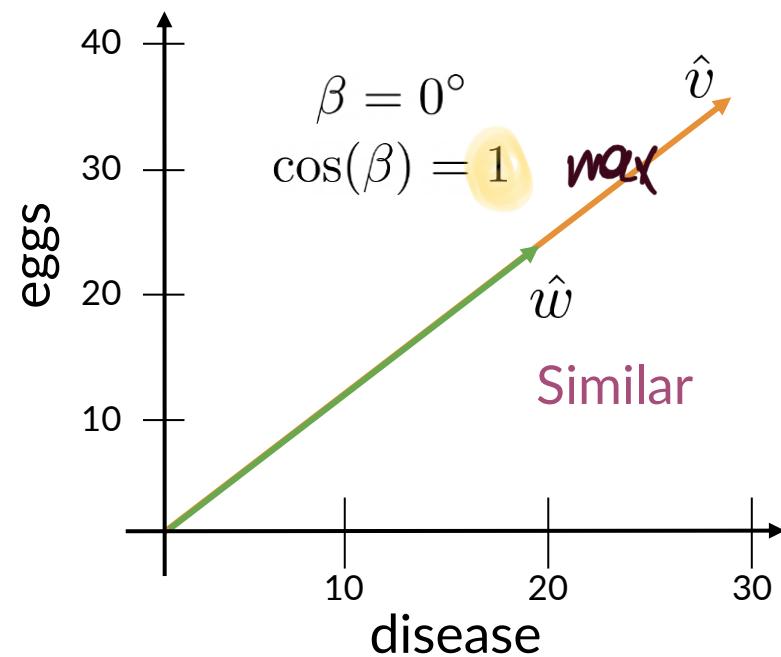
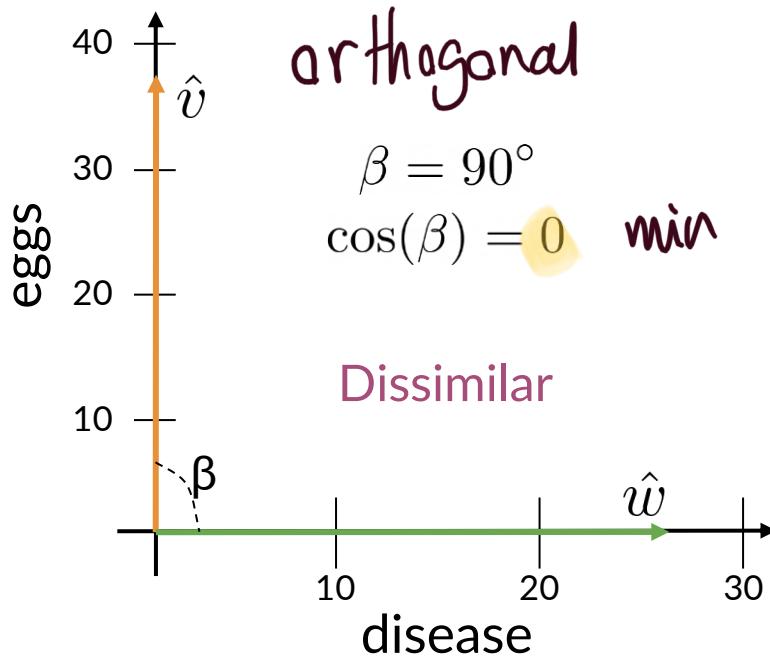
# Cosine Similarity



$$\hat{v} \cdot \hat{w} = \|\hat{v}\| \|\hat{w}\| \cos(\beta)$$
$$\cos(\beta) = \frac{\hat{v} \cdot \hat{w}}{\|\hat{v}\| \|\hat{w}\|}$$
$$= \frac{(20 \times 30) + (40 \times 20)}{\sqrt{20^2 + 40^2} \times \sqrt{30^2 + 20^2}}$$
$$= 0.87$$

Annotations: "dat" with arrows pointing to the x-coordinates of both vectors, and "narm" with arrows pointing to the y-coordinates of both vectors.

# Cosine Similarity



# Summary

- Cosine  $\propto$  Similarity
- Cosine Similarity gives values between 0 and 1



deeplearning.ai

# Manipulating Words in Vector Spaces

---

# Outline

- How to use vector representations

# Manipulating word vectors



USA



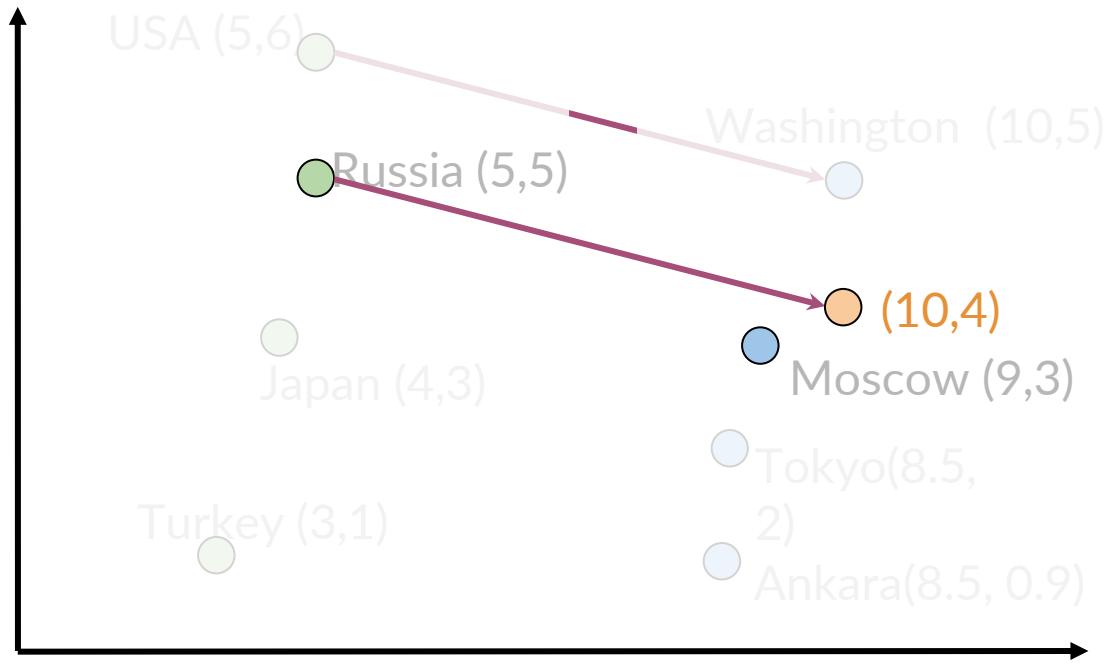
Washington  
DC



Russia



# Manipulating word vectors



$$\text{Washington} - \text{USA} = [5 \quad -1]$$

$$\text{Russia} + [5 \quad -1] = [10 \quad 4]$$



MOSCOW

[Mikolov et al, 2013, Distributed Representations of Words and Phrases and their Compositionality]

# Summary

- Use known relationships to make predictions



deeplearning.ai

# Visualization and PCA

dimension reduction

# Outline

- Some motivation for visualization
- Principal Component Analysis

higher dimension  
↓  
lower

# Visualization of word vectors

$d > 2$

oil	0.20	...	0.10
gas	2.10	...	3.40
city	9.30	...	52.1
town	6.20	...	34.3

How can you visualize if your representation captures these relationships?



oil & gas

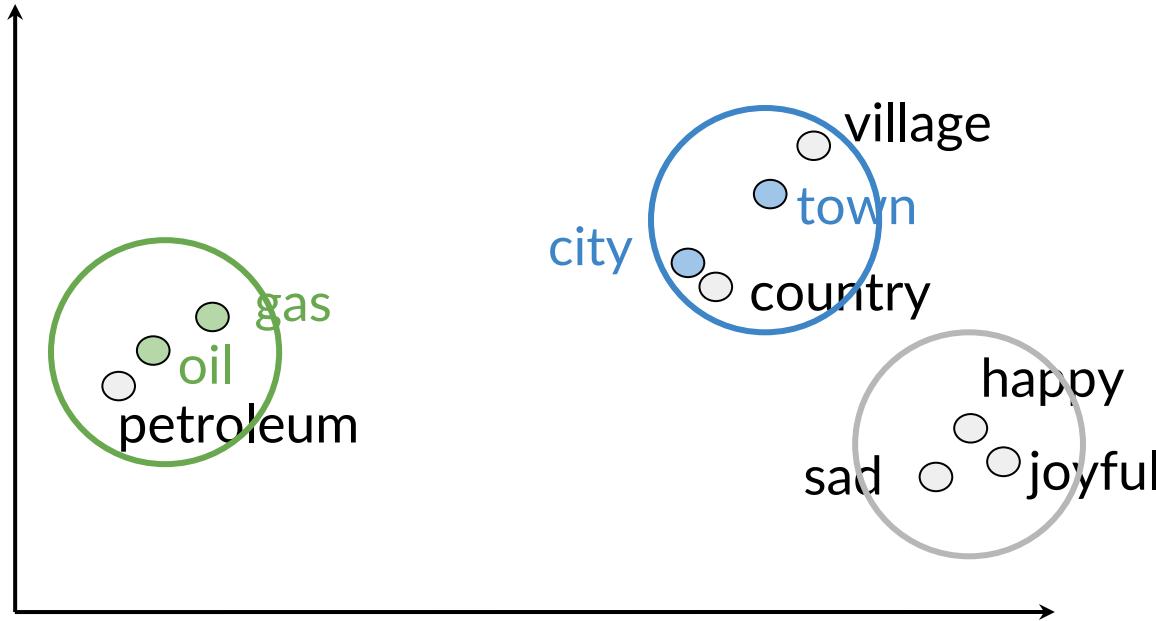


town & city

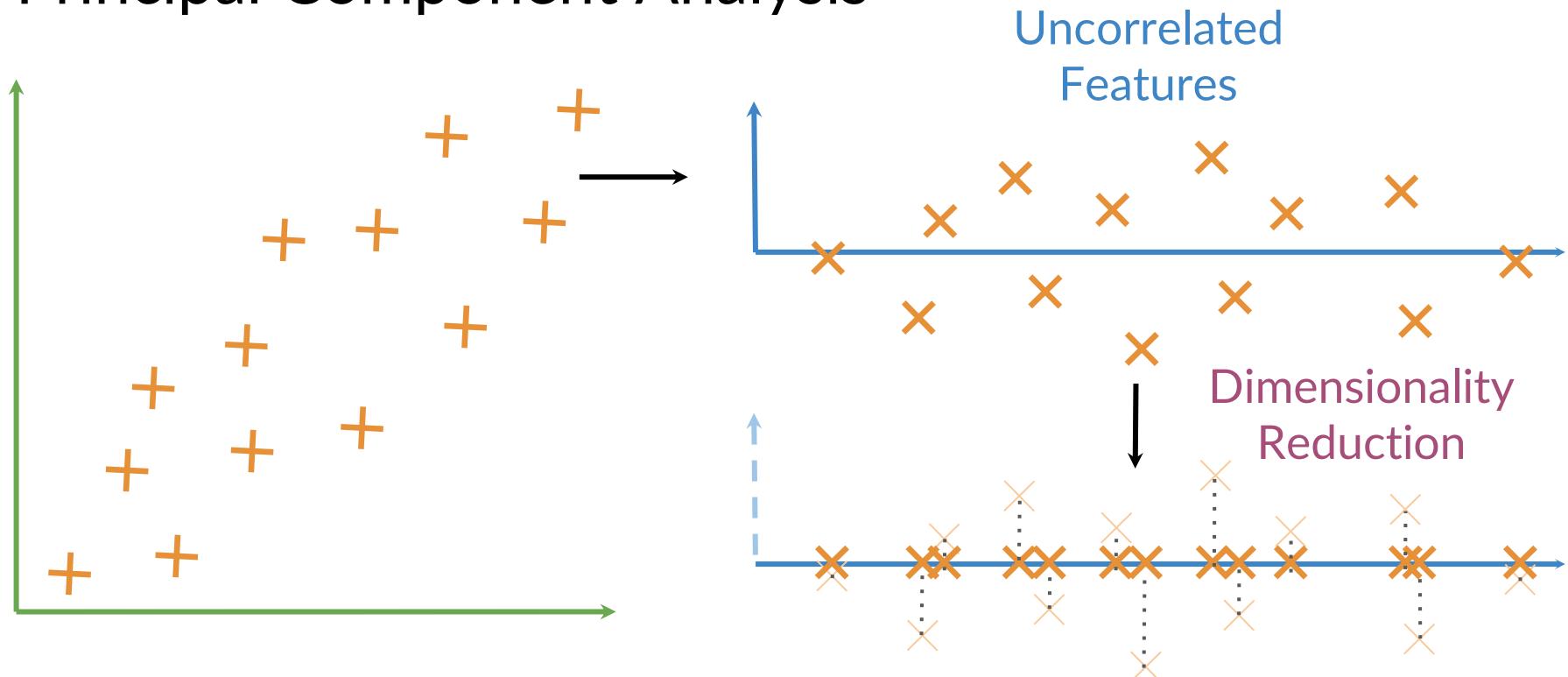
# Visualization of word vectors

$d > 2$				$d = 2$			
oil	0.20	...	0.10	oil	2.30	21.2	
gas	2.10	...	3.40	gas	1.56	19.3	
city	9.30	...	52.1	city	13.4	34.1	
town	6.20	...	34.3	town	15.6	29.8	

# Visualization of word vectors



# Principal Component Analysis



# Summary

- Original Space  Uncorrelated features  Dimension reduction
- Visualization to see words relationships in the vector space



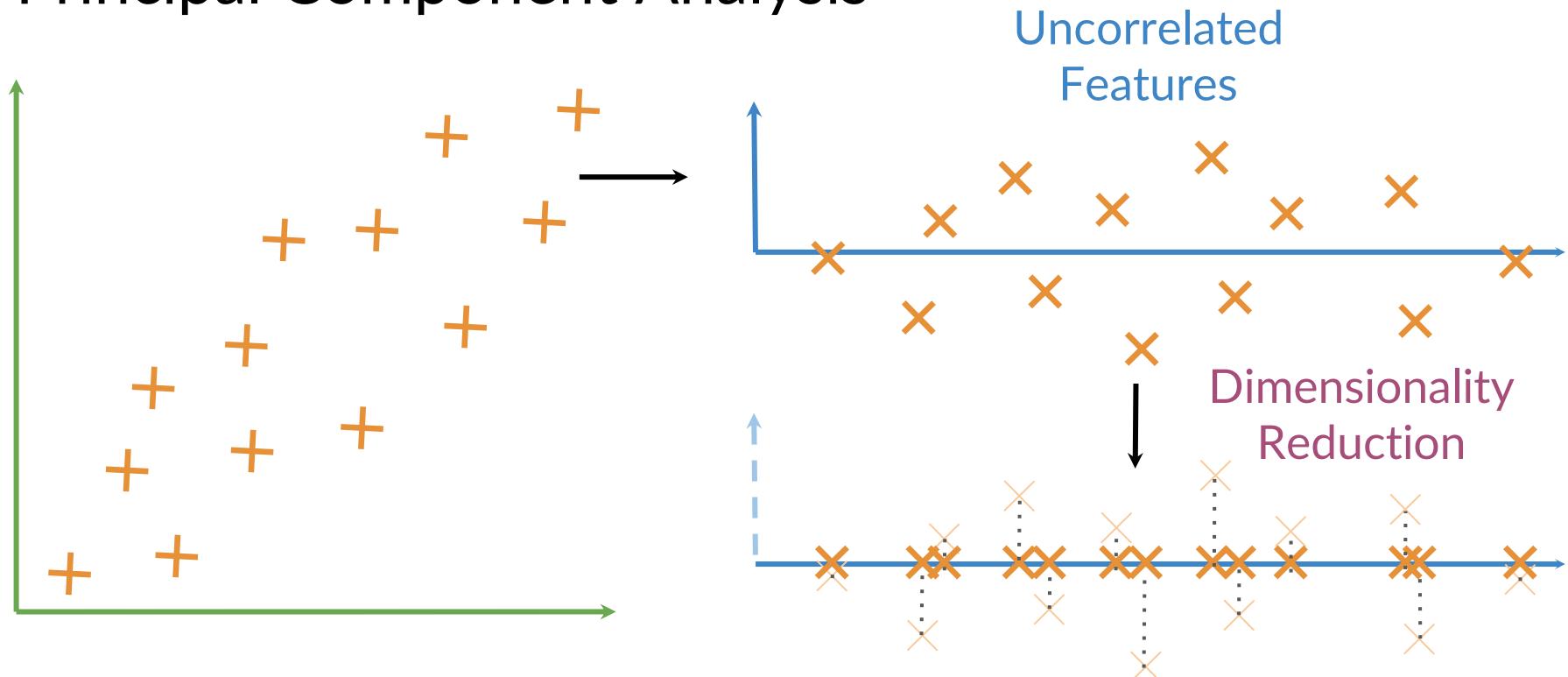
deeplearning.ai

# PCA Algorithm

# Outline

- How to get uncorrelated features
- How to reduce dimensions while retaining as much information as possible

# Principal Component Analysis

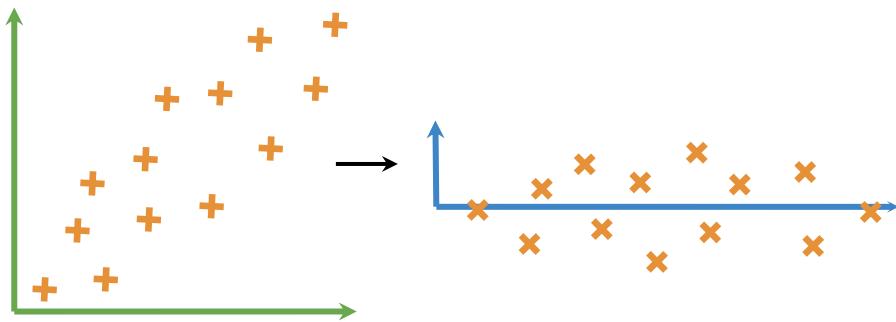


# PCA algorithm

Eigenvector: Uncorrelated features for your data

Eigenvalue: the amount of information retained by each feature

# PCA algorithm

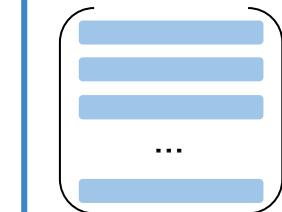
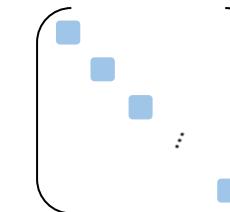
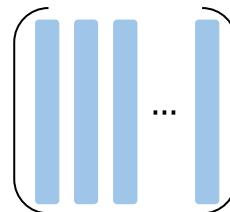


Mean Normalize Data  $x_i = \frac{x_i - \mu_{x_i}}{\sigma_{x_i}}$

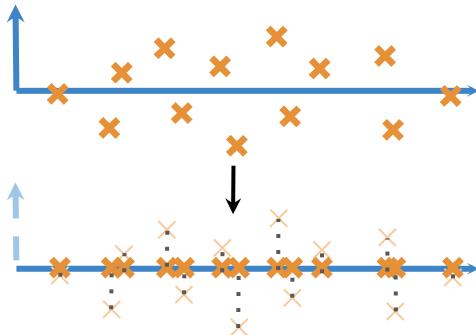
Get Covariance Matrix  $\Sigma$

Perform SVD  $SVD(\Sigma)$

Eigenvectors      Eigenvalues

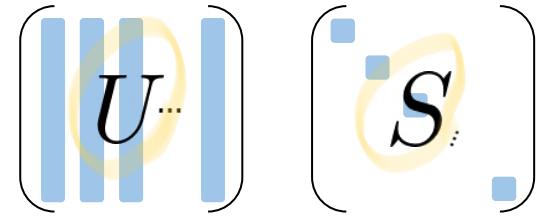


# PCA algorithm



Eigenvectors

Eigenvalues



Dot Product to  
Project Data

$$X' = XU[:, 0 : 2]$$

Percentage of  
Retained Variance

$$\frac{\sum_{i=0}^1 S_{ii}}{\sum_{j=0}^d S_{jj}}$$

final  
reduced  
dim

# Summary

- Eigenvectors give the direction of uncorrelated features
- Eigenvalues are the variance of the new features
- Dot product gives the projection on uncorrelated features