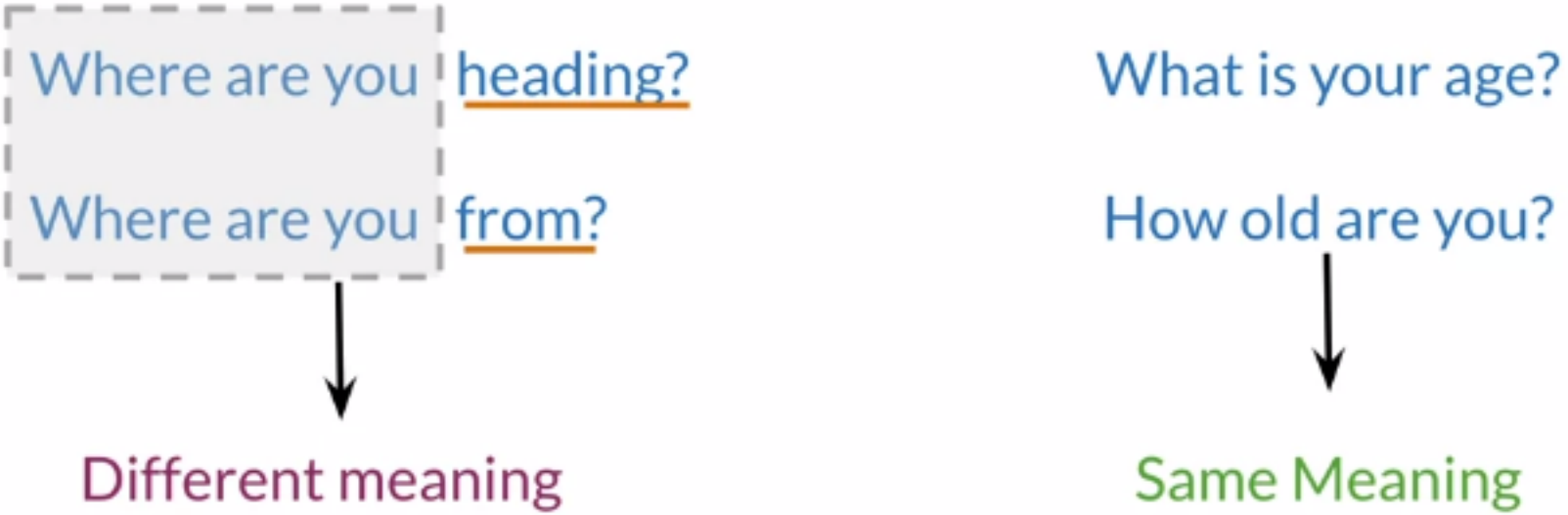
# Outline

- 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

# 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

# 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

| | simple | raw | like | I |
|---|---|---|---|---|
| data | 2 | 1 | 1 | 0 |

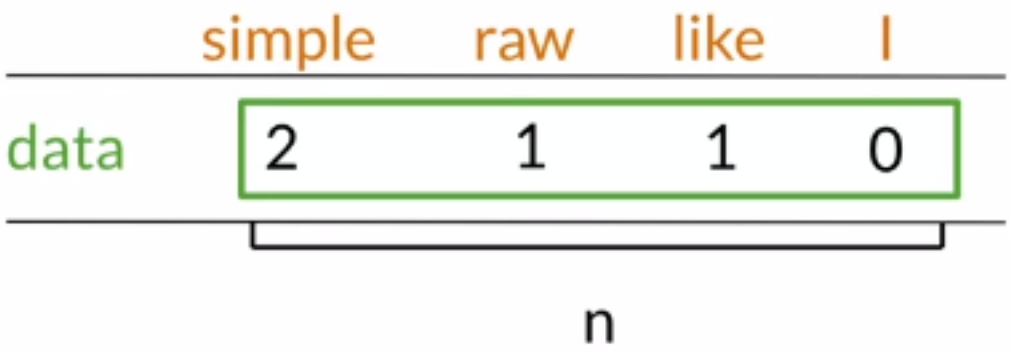
# Word by Word Design

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

I like simple data

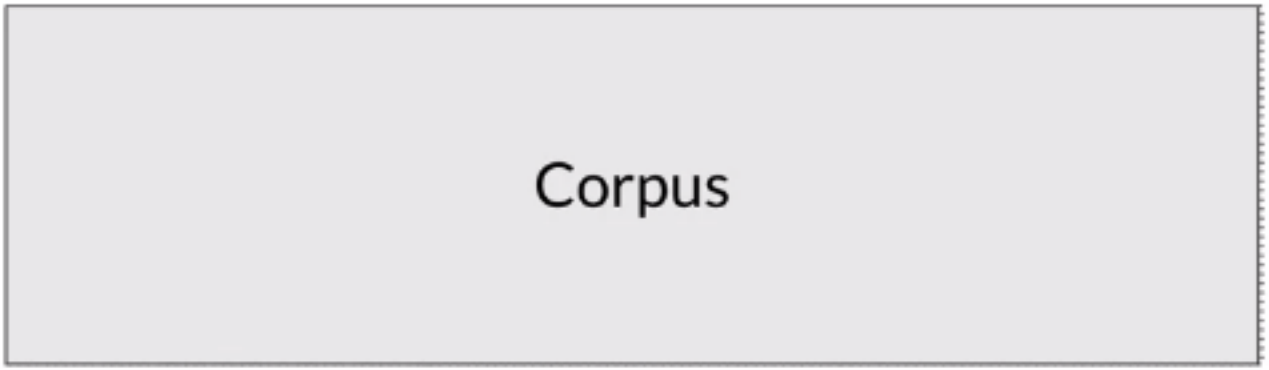
I prefer simple raw data

k=2



# Word by Document Design

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



# Word by Document Design

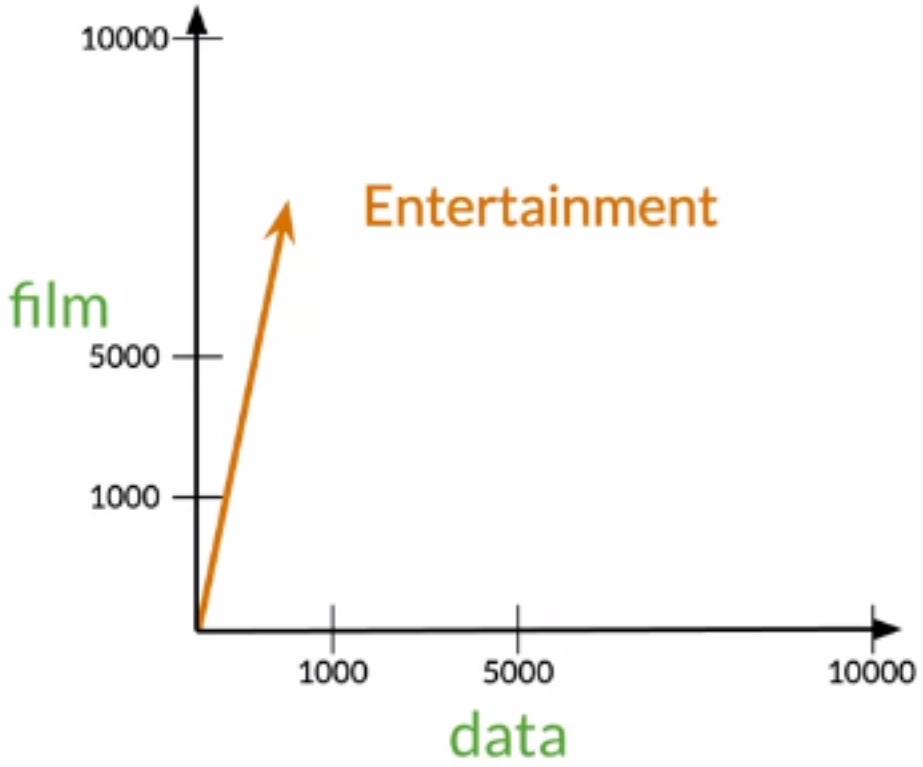
Number of times a word *occurs within a certain category*

# Word by Document Design

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

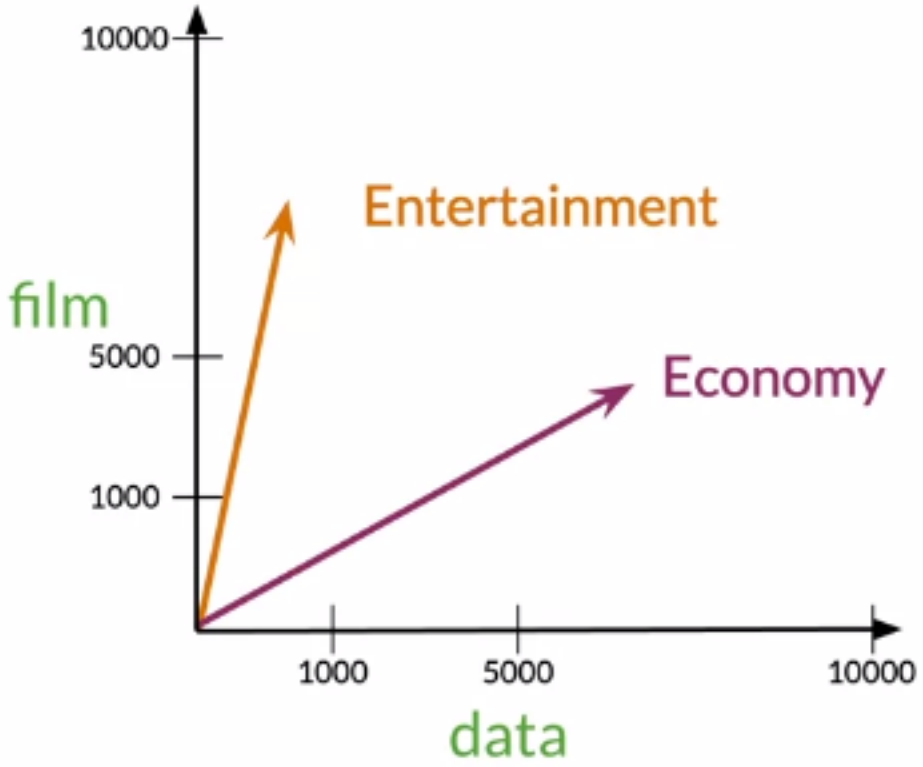

| | Entertainment | Economy | Machine Learning |
|---|---|---|---|
| data | 500 | 6620 | 9320 |
| film | 7000 | 4000 | 1000 |

# Vector Space



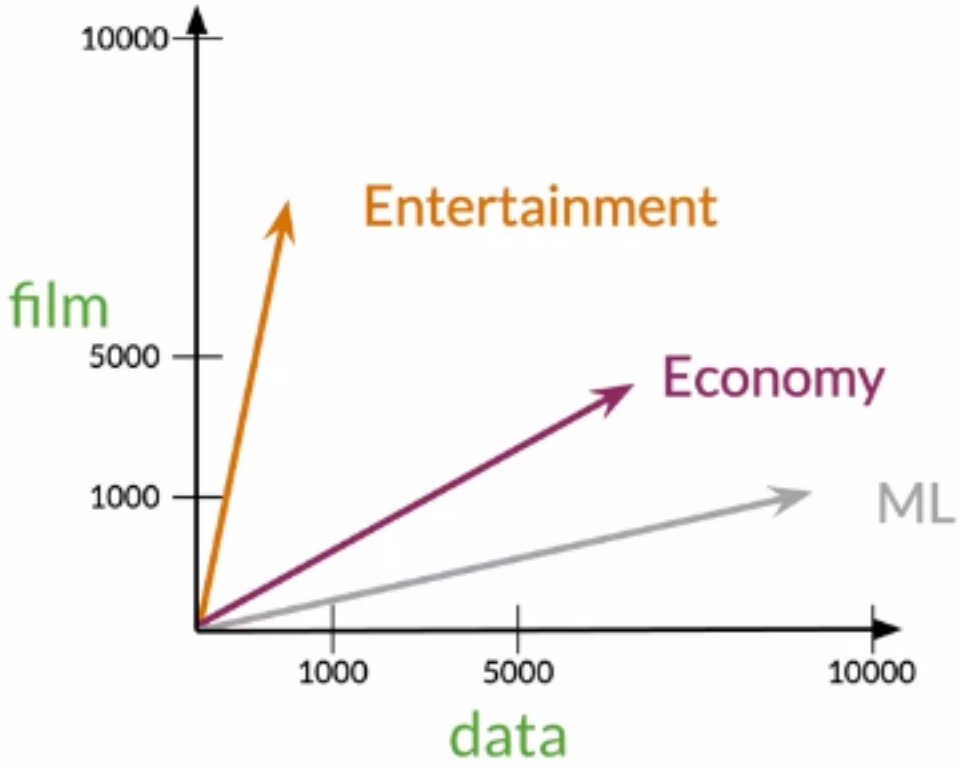

| | Entertainment | Economy | ML |
|---|---|---|---|
| data | 500 | 6620 | 9320 |
| film | 7000 | 4000 | 1000 |

# Vector Space



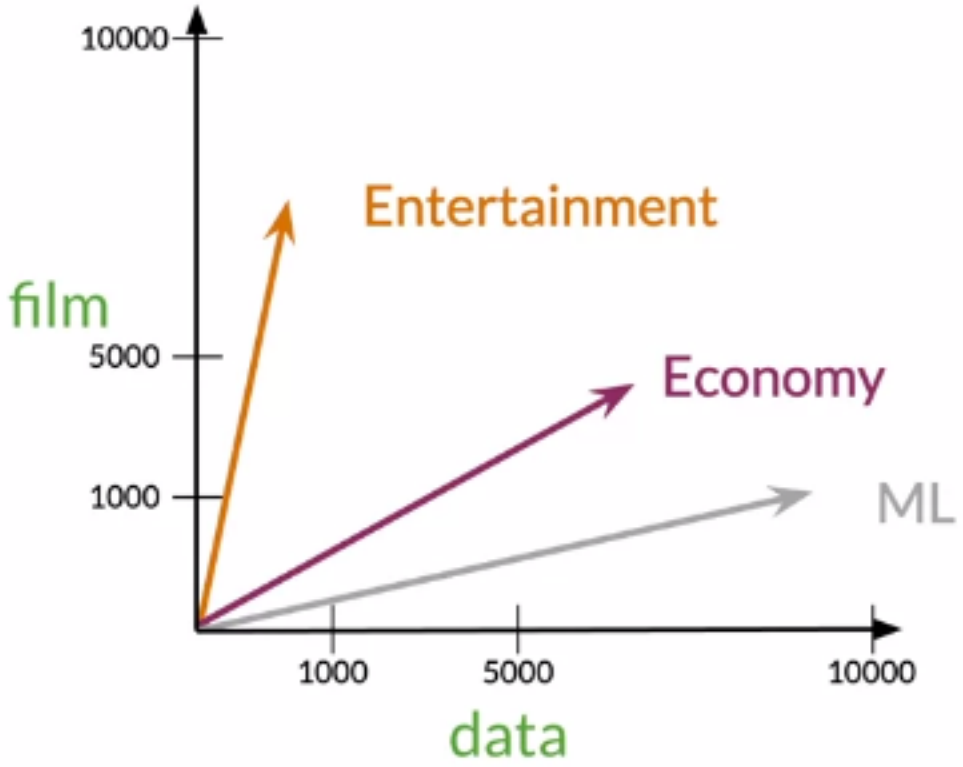

| | Entertainment | Economy | ML |
|---|---|---|---|
| data | 500 | 6620 | 9320 |
| film | 7000 | 4000 | 1000 |

# Vector Space




| | Entertainment | Economy | ML |
|---|---|---|---|
| data | 500 | 6620 | 9320 |
| film | 7000 | 4000 | 1000 |

# Vector Space




| | Entertainment | Economy | ML |
|---|---|---|---|
| data | 500 | 6620 | 9320 |
| film | 7000 | 4000 | 1000 |

Measures of "similarity:"
Angle
Distance

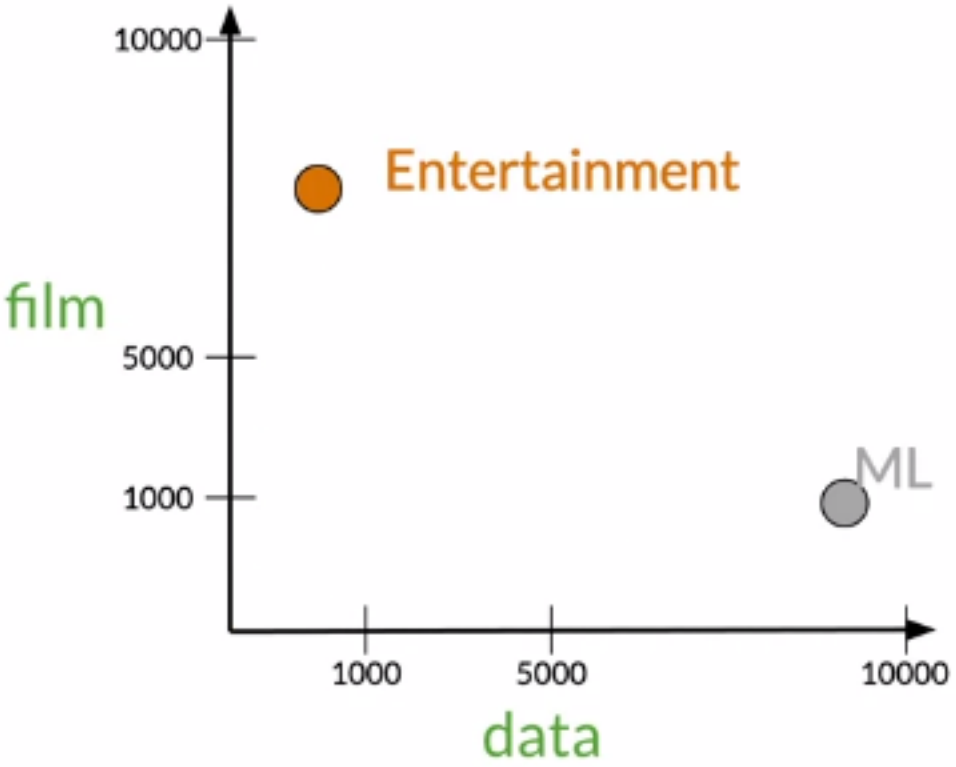
# Summary

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

# Outline

- Euclidean distance
- N-dimension vector representations comparison

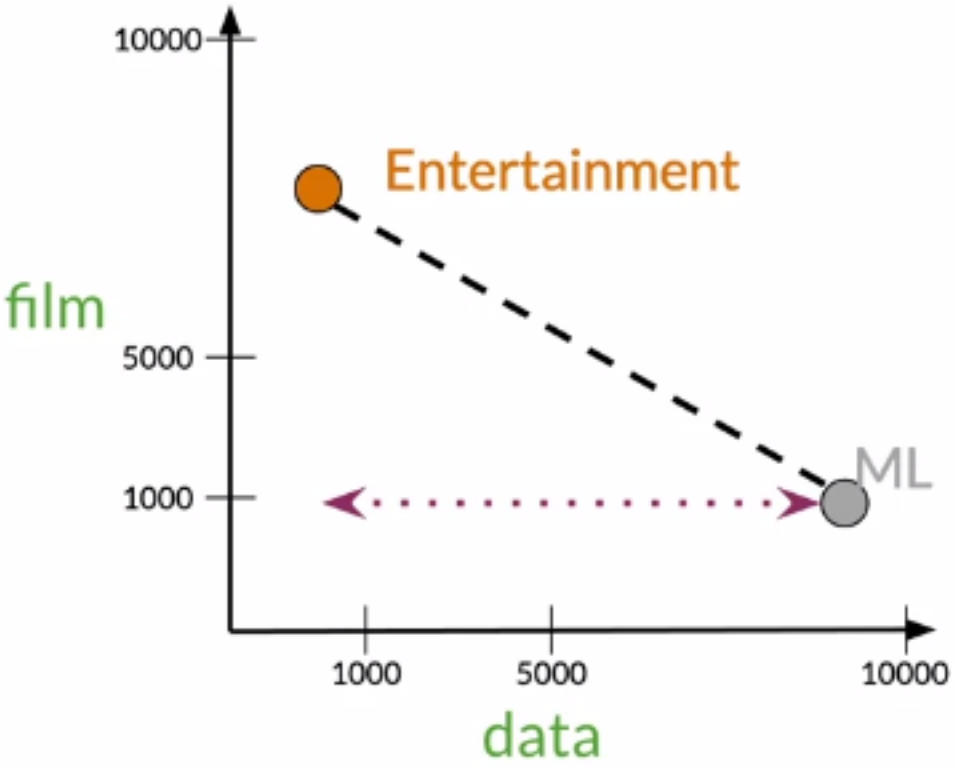
# Euclidean distance




Corpus A: (500,7000)

Corpus B: (9320,1000)

# Euclidean distance

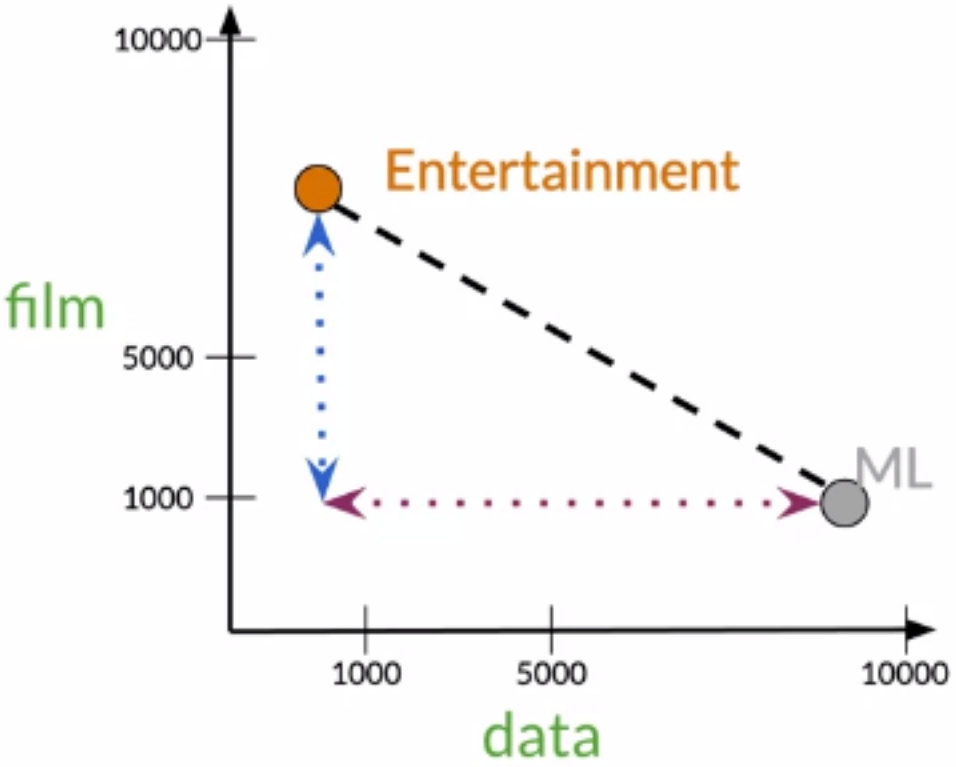



Corpus A: (500,7000)

Corpus B: (9320,1000)

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

# Euclidean distance

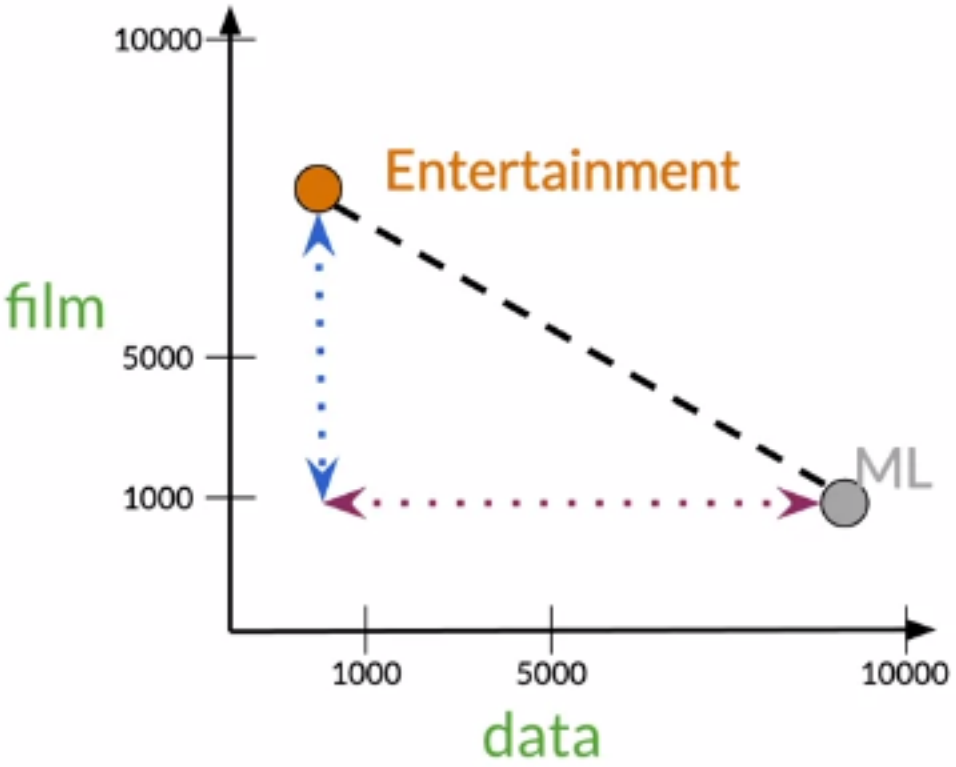



Corpus A: (500,7000)

Corpus B: (9320,1000)

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

# Euclidean distance




Corpus A: (500,7000)

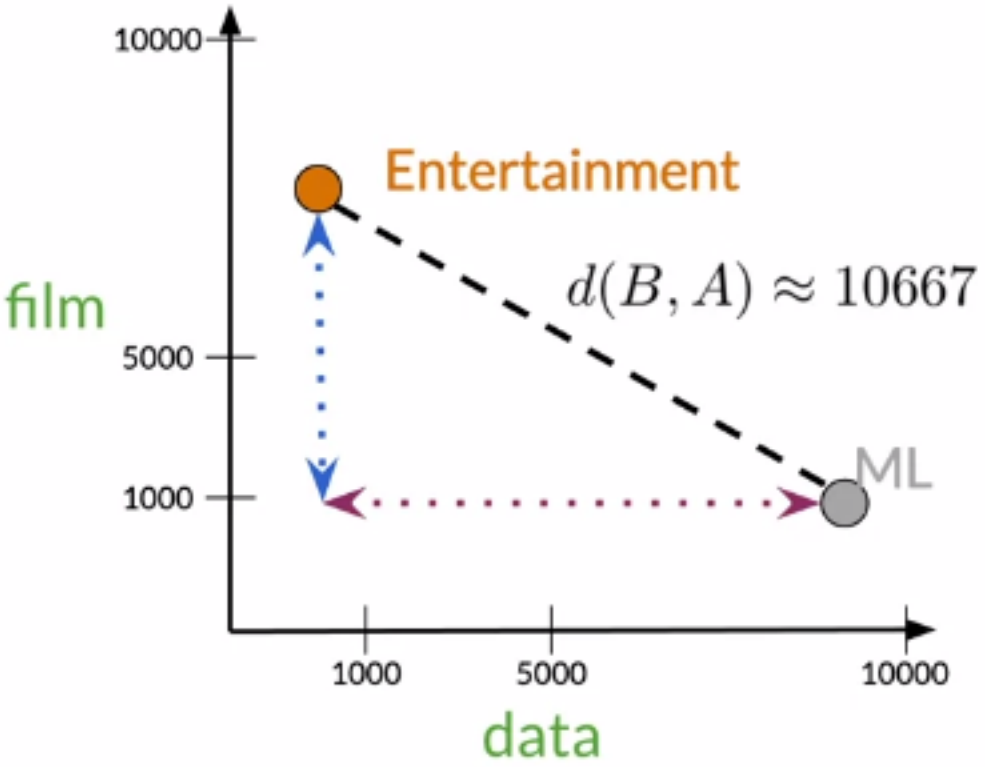
Corpus B: (9320,1000)

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

$$c^2 = a^2 + b^2$$

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

# Euclidean distance




Corpus A: (500,7000)

Corpus B: (9320,1000)

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

$$c^2 = a^2 + b^2$$

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

# Euclidean distance for n-dimensional vectors

$\vec{v}$

| | data | boba | ice-cream |
|---|---|---|---|
| AI | 6 | 0 | 1 |
| drinks | 0 | 4 | 6 |
| food | 0 | 6 | 8 |

# Euclidean distance for n-dimensional vectors

| | data | boba ($\vec{w}$) | ice-cream ($\vec{v}$) |
|---|---|---|---|
| AI | 6 | 0 | 1 |
| drinks | 0 | 4 | 6 |
| food | 0 | 6 | 8 |

# Euclidean distance for n-dimensional vectors

| | data | boba ($\vec{w}$) | ice-cream ($\vec{v}$) |
|---|---|---|---|
| AI | 6 | 0 | 1 |
| drinks | 0 | 4 | 6 |
| food | 0 | 6 | 8 |

$$= \sqrt{(1-0)^2 + (6-4)^2 + (8-6)^2}$$

$$= \sqrt{1+4+4} = \sqrt{9} = 3$$

# Euclidean distance for n-dimensional vectors

| | data | $\vec{w}$ boba | $\vec{v}$ ice-cream |
|---|---|---|---|
| AI | 6 | 0 | 1 |
| drinks | 0 | 4 | 6 |
| food | 0 | 6 | 8 |

$$= \sqrt{(1-0)^2 + (6-4)^2 + (8-6)^2}$$

$$= \sqrt{1+4+4} = \sqrt{9} = 3$$

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

# Euclidean distance for n-dimensional vectors

| | data | boba ($\vec{w}$) | ice-cream ($\vec{v}$) |
|---|---|---|---|
| AI | 6 | 0 | 1 |
| drinks | 0 | 4 | 6 |
| food | 0 | 6 | 8 |

$$= \sqrt{(1-0)^2 + (6-4)^2 + (8-6)^2}$$

$$= \sqrt{1+4+4} = \sqrt{9} = 3$$

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

# 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
```

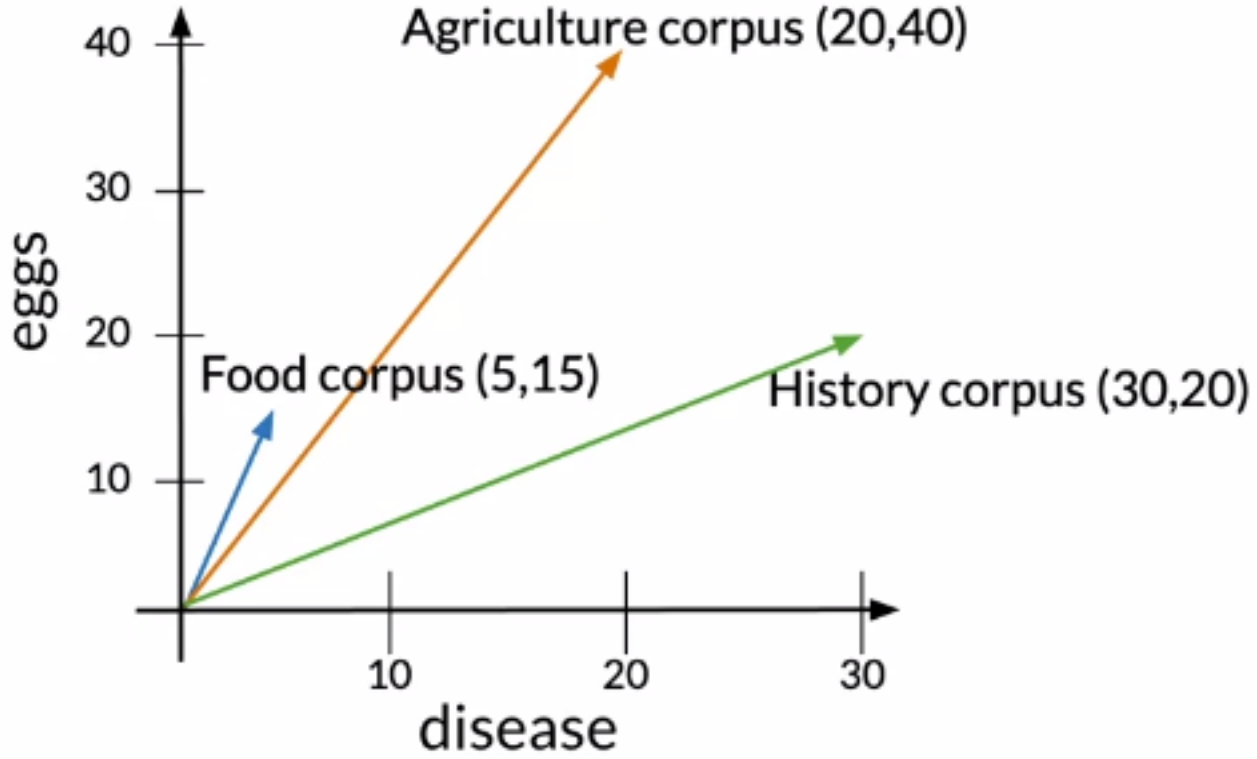
# Summary

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

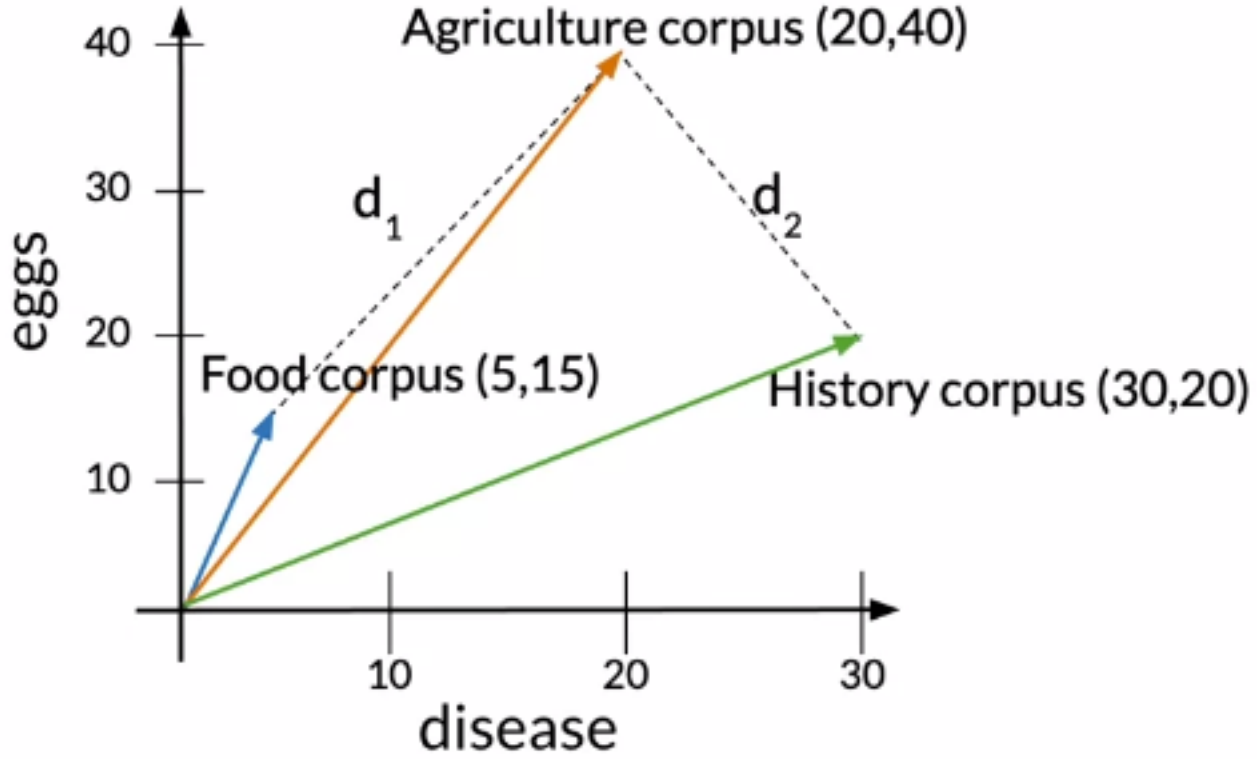
# Outline

- Problems with Euclidean Distance
- Cosine similarity

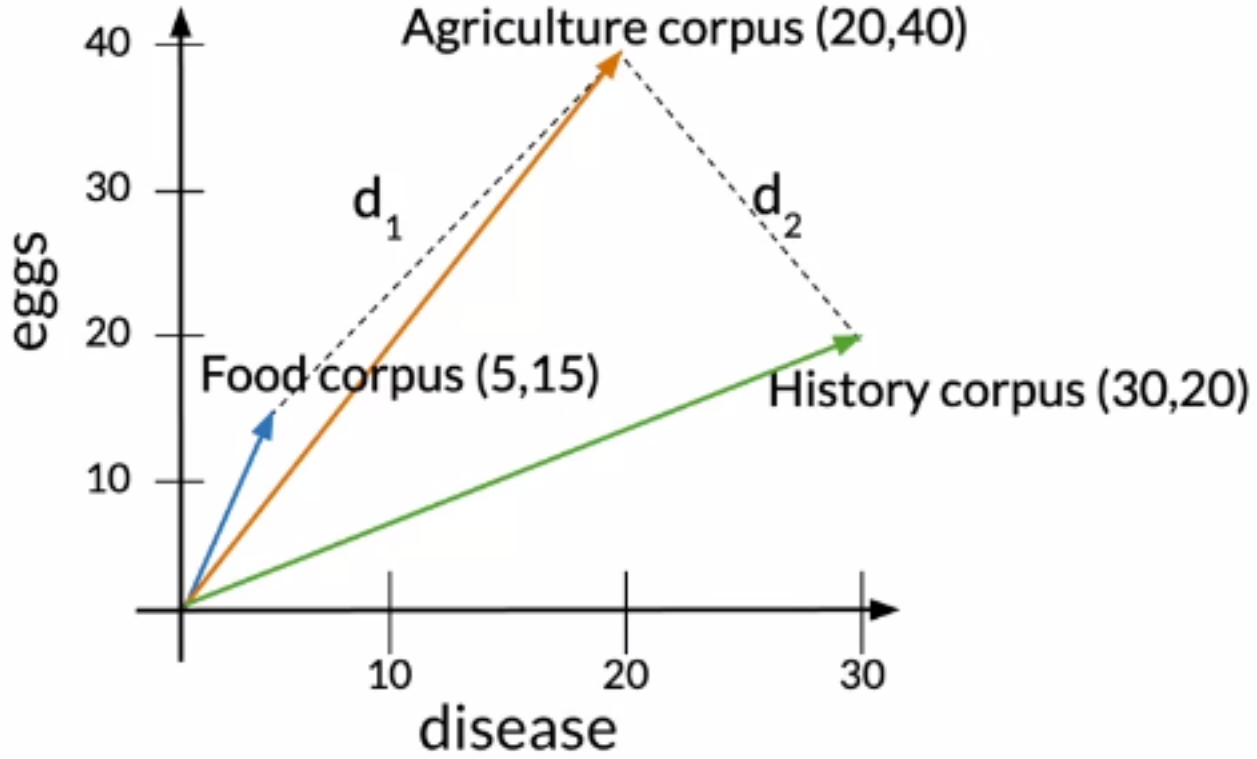
# Euclidean distance vs Cosine similarity



# Euclidean distance vs Cosine similarity

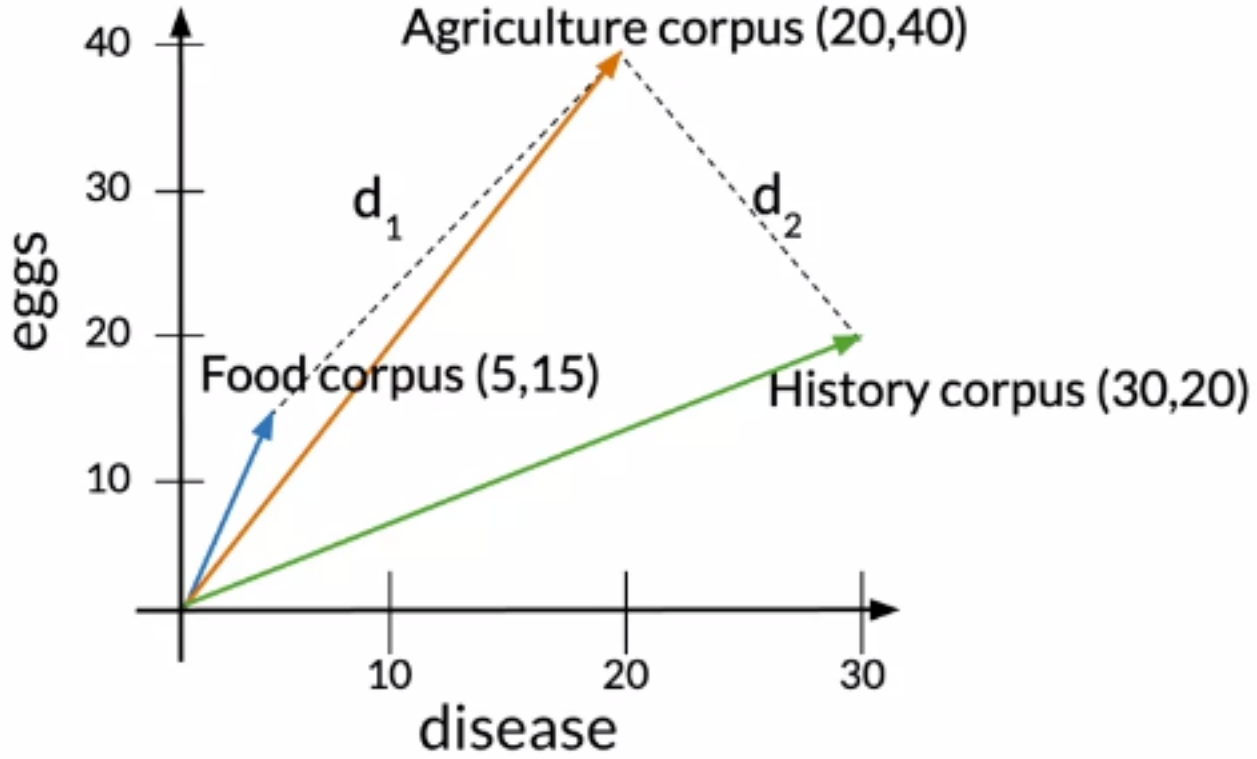


# Euclidean distance vs Cosine similarity




Euclidean distance: $d_2 < d_1$

# Euclidean distance vs Cosine similarity




Euclidean distance: $d_2 < d_1$

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

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

# Previous definitions

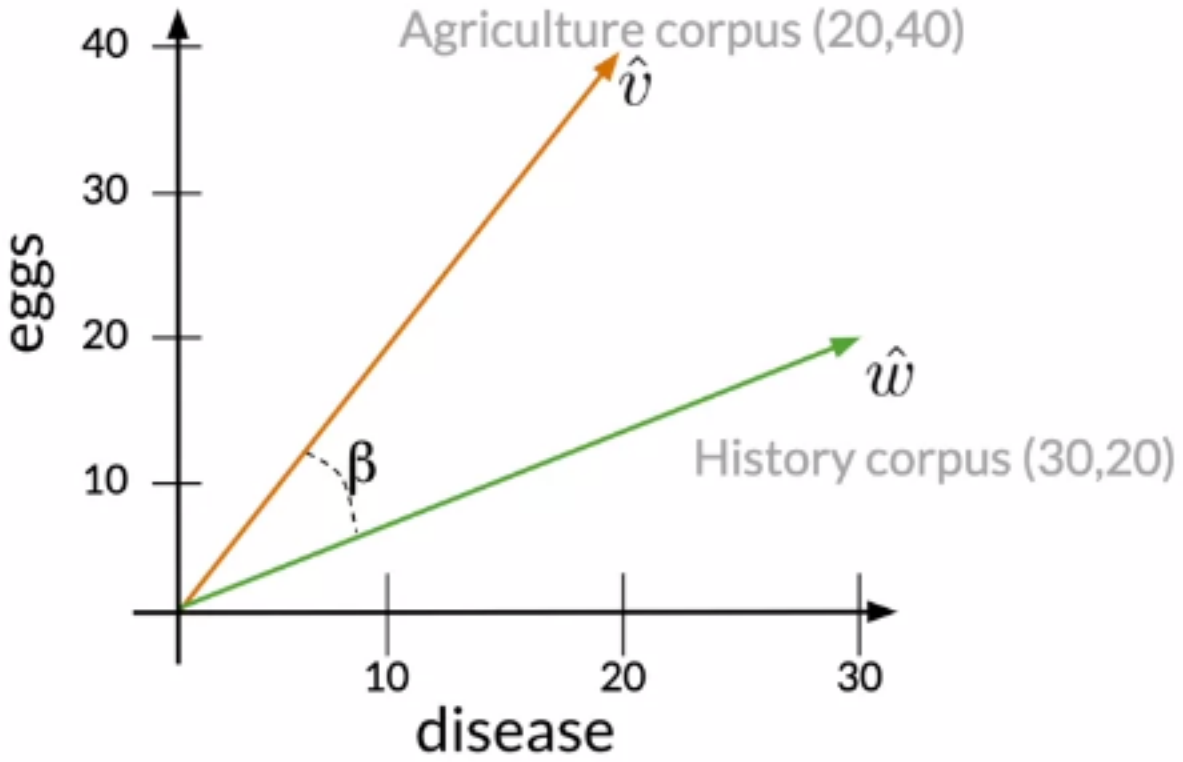
Vector norm

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

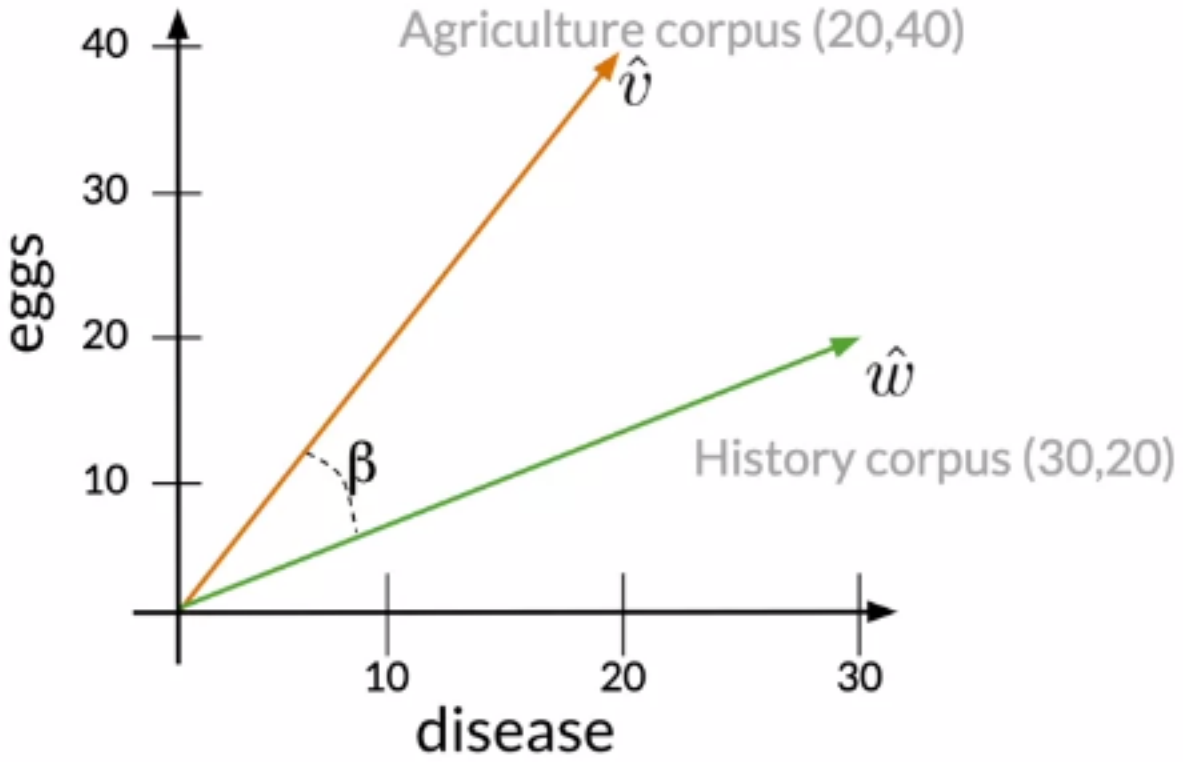
Dot product

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

# Cosine Similarity

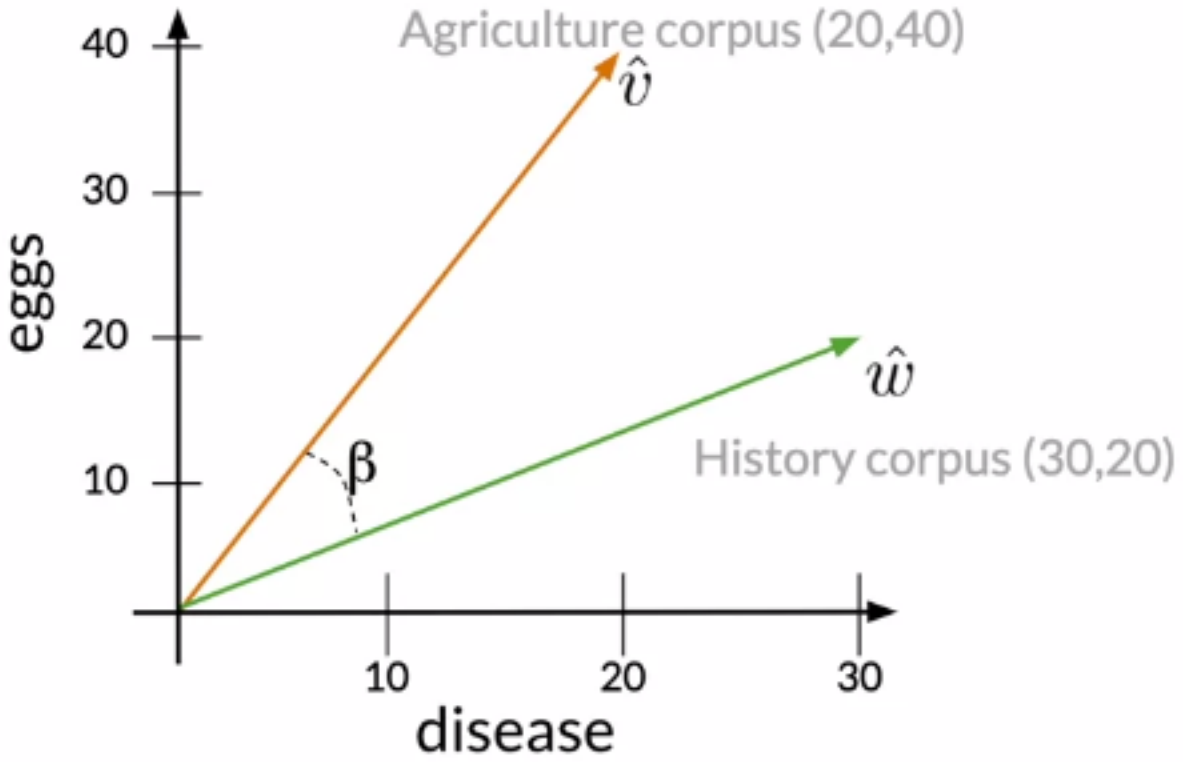


# Cosine Similarity




$$\hat{v} \cdot \hat{w} = \|\hat{v}\| \|\hat{w}\| \cos(\beta)$$

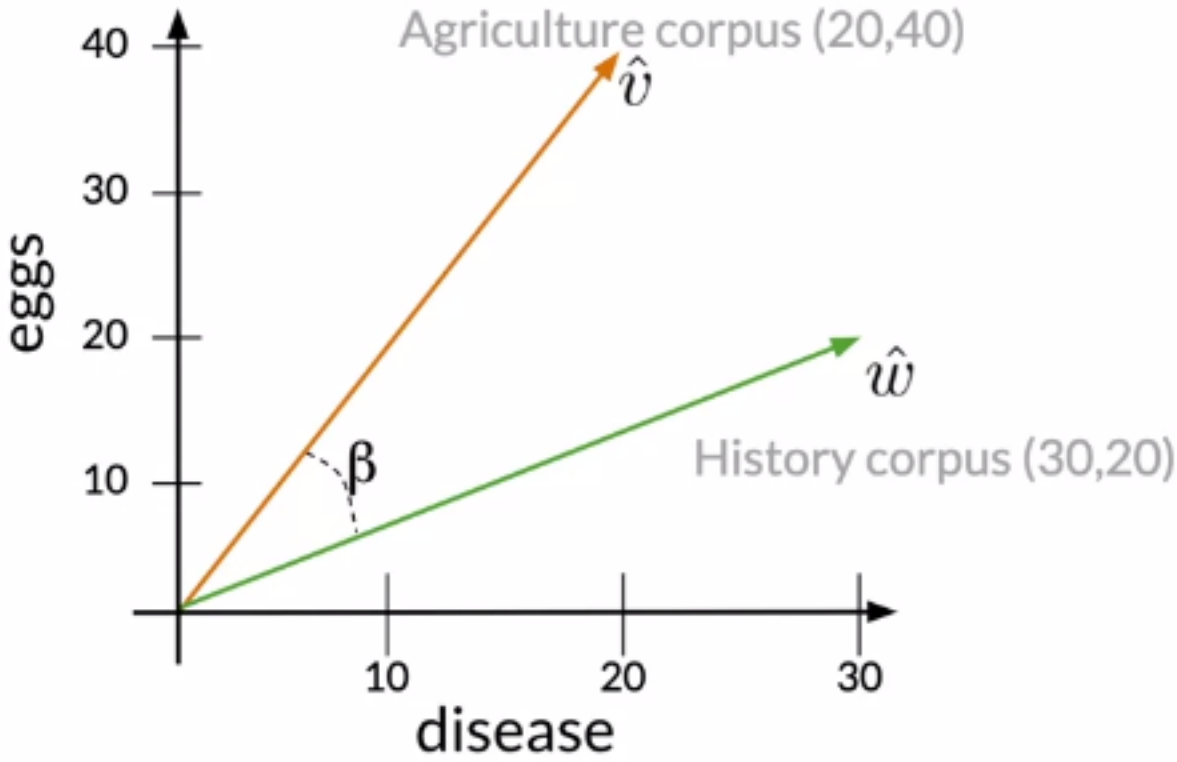
# Cosine Similarity




$$\hat{v} \cdot \hat{w} = \|\hat{v}\| \|\hat{w}\| \cos(\beta)$$

$$\cos(\beta) = \frac{\hat{v} \cdot \hat{w}}{\|\hat{v}\| \|\hat{w}\|}$$

# Cosine Similarity



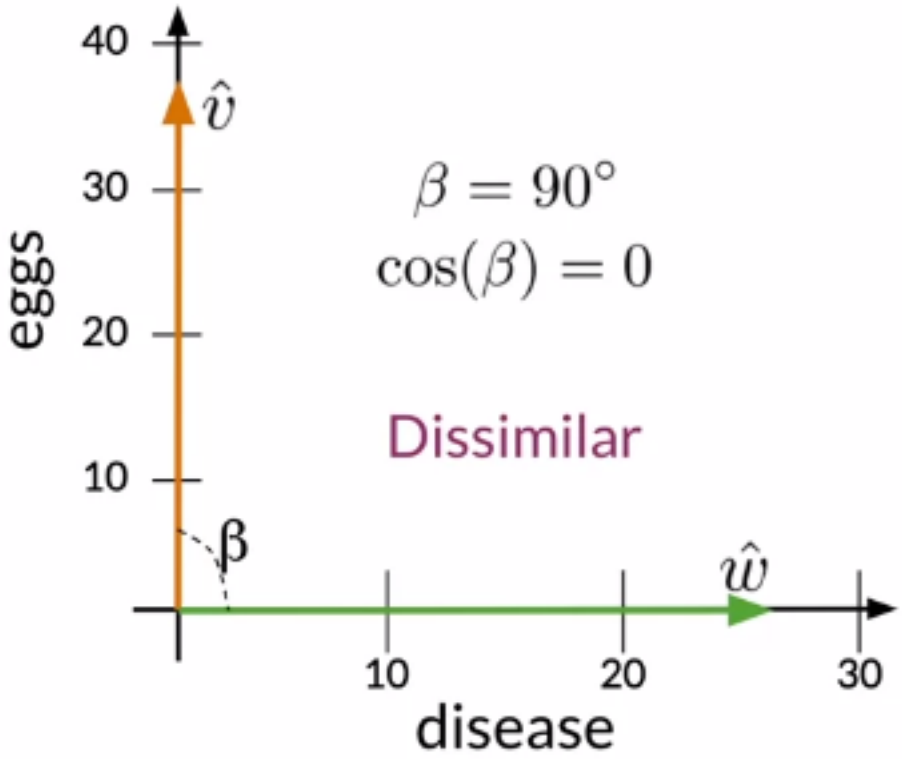

$$\hat{v} \cdot \hat{w} = \|\hat{v}\| \|\hat{w}\| \cos(\beta)$$

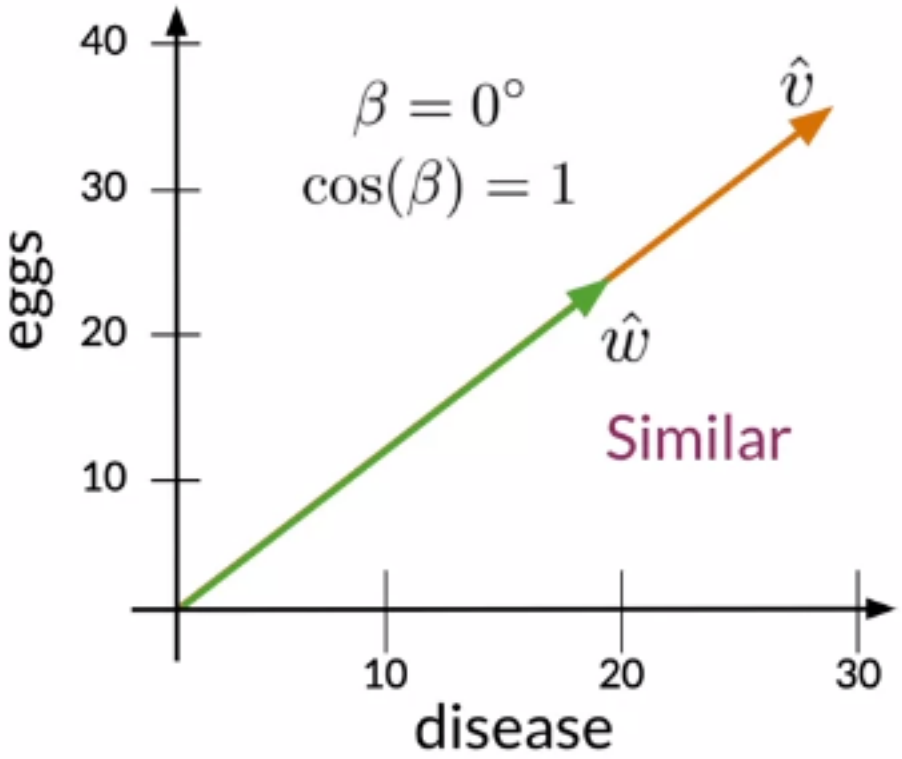
$$\cos(\beta) = \frac{\hat{v} \cdot \hat{w}}{\|\hat{v}\| \|\hat{w}\|}$$

$$= \frac{(20 \times 30) + (40 \times 20)}{\sqrt{20^2 + 40^2} \times \sqrt{30^2 + 20^2}}$$

$$= 0.87$$

# Cosine Similarity



# Summary

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

# Outline

- How to use vector representations

# Manipulating word vectors

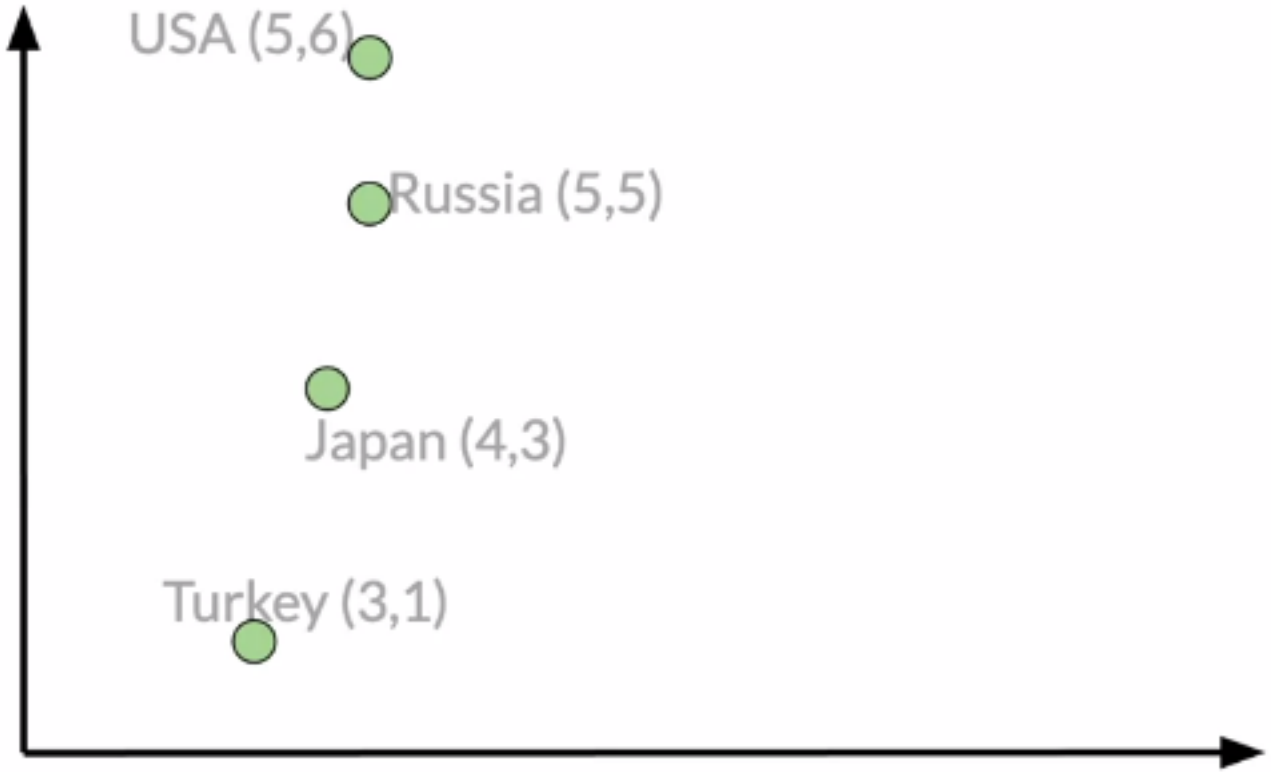
USA

Washington DC

Russia

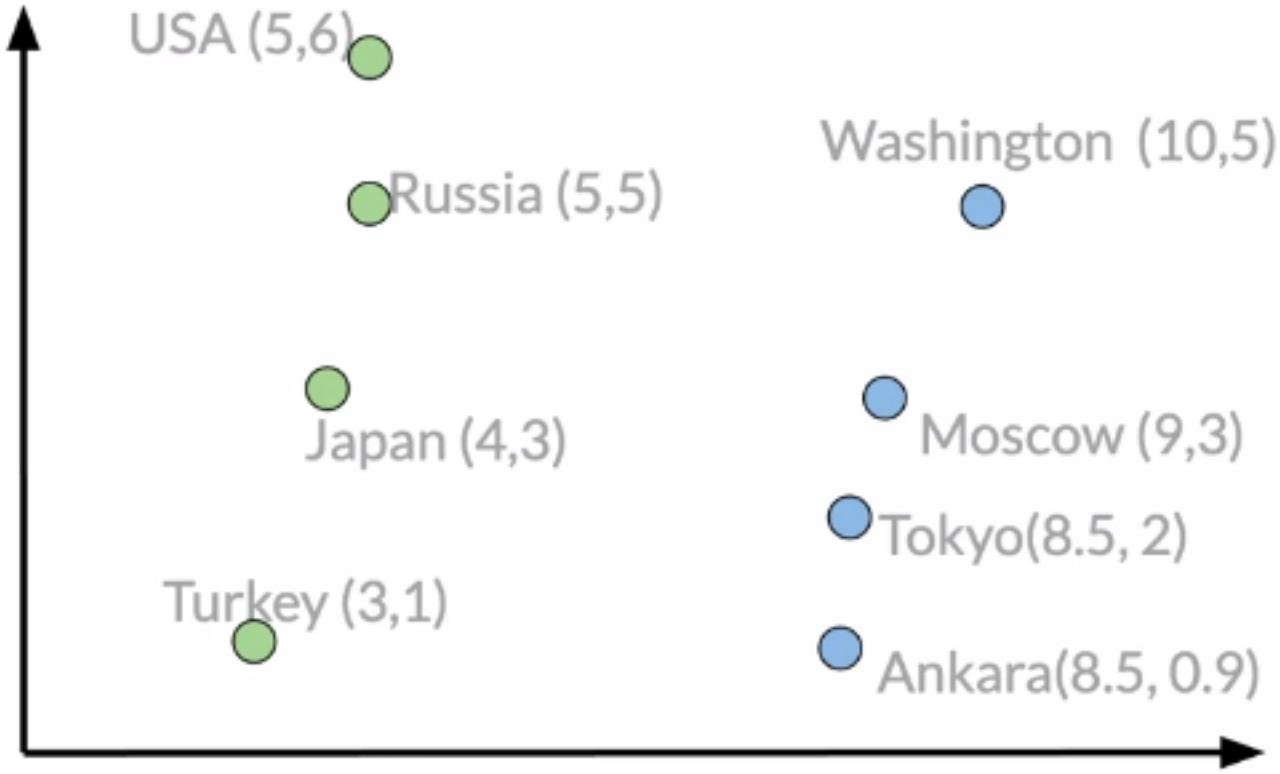
?

# Manipulating word vectors



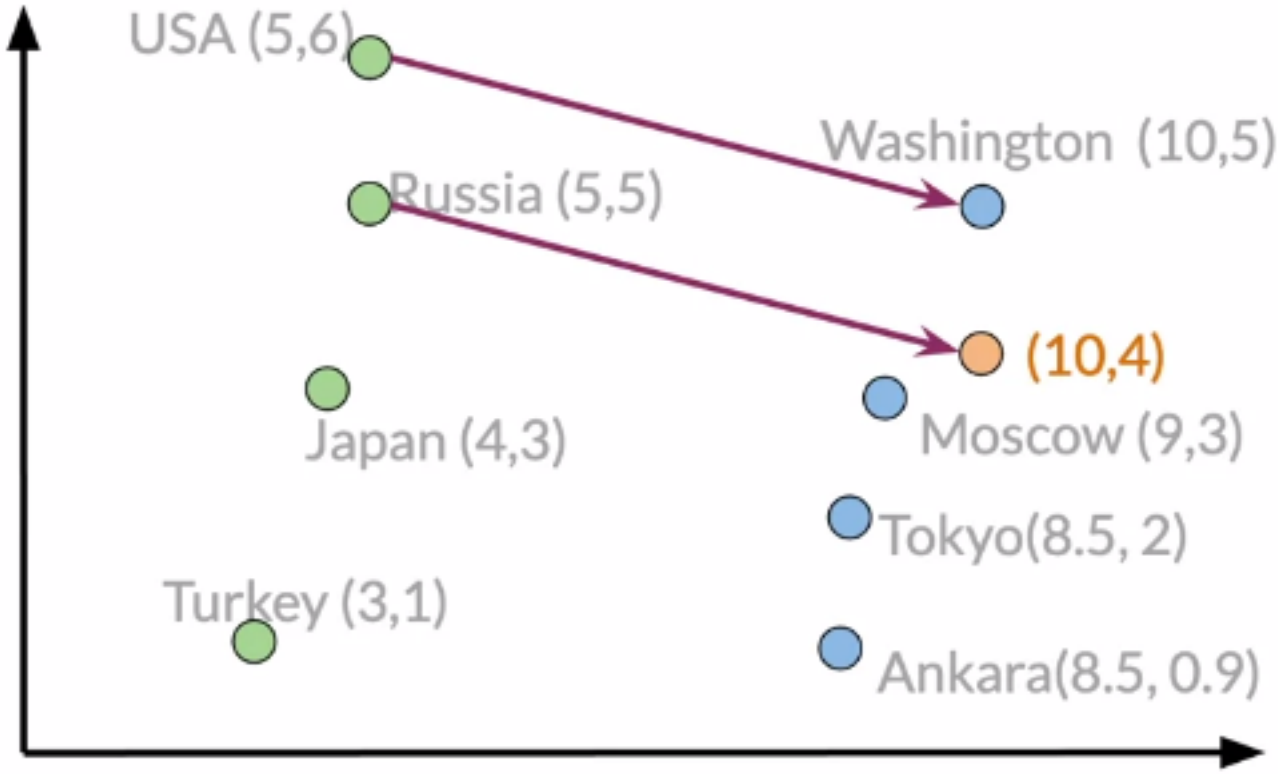

[Mikolov et al, 2013, Distributed Representations of Words and Phrases and their Compositionality]

# Manipulating word vectors




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

# Manipulating word vectors

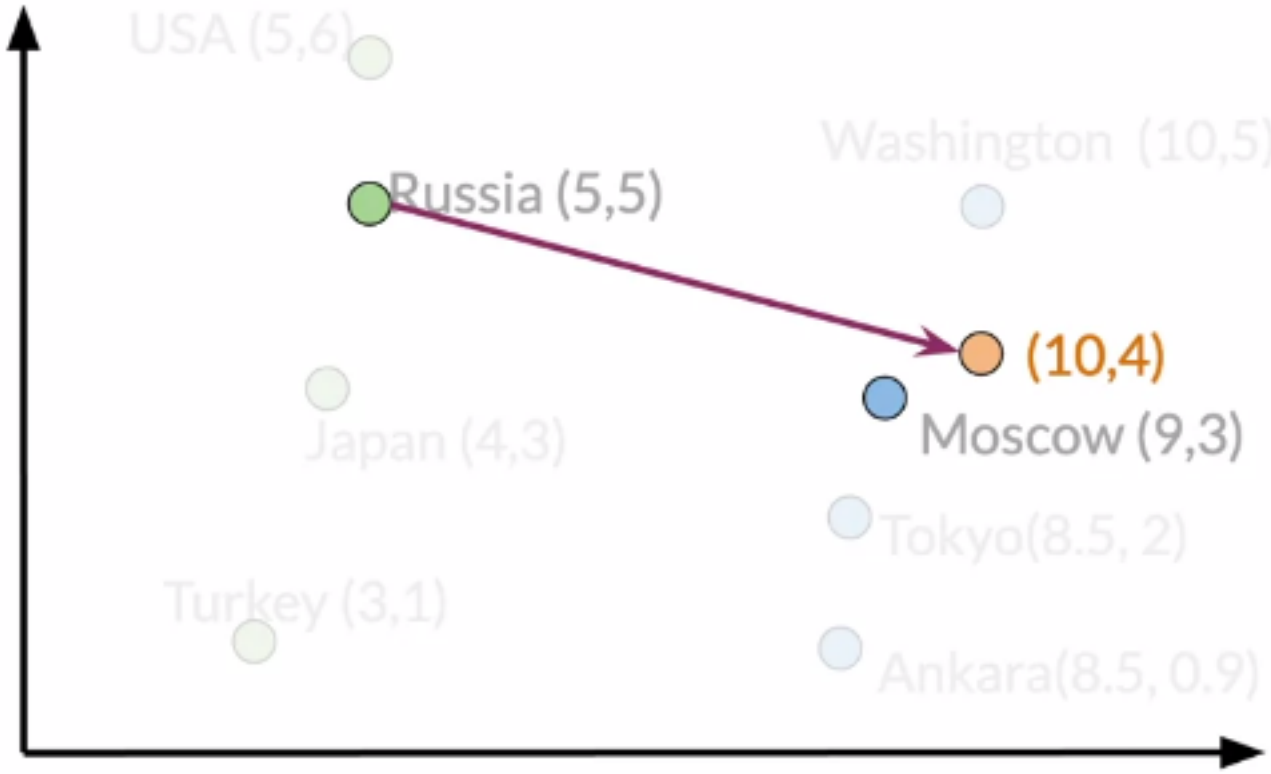



Washington - USA = $\begin{bmatrix} 5 & -1 \end{bmatrix}$

Russia + $\begin{bmatrix} 5 & -1 \end{bmatrix}$ = $\begin{bmatrix} 10 & 4 \end{bmatrix}$

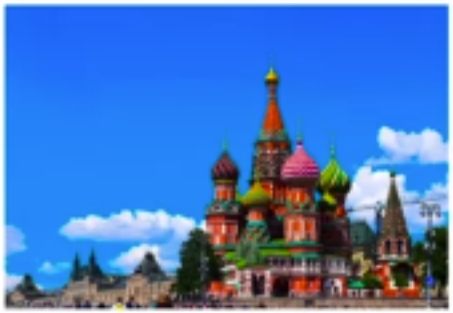
[Mikolov et al, 2013, Distributed Representations of Words and Phrases and their Compositionality]

# Manipulating word vectors




$$\text{Washington - USA} = \begin{bmatrix} 5 & -1 \end{bmatrix}$$

$$\text{Russia} + \begin{bmatrix} 5 & -1 \end{bmatrix} = \begin{bmatrix} 10 & 4 \end{bmatrix}$$



Moscow

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

# Summary

- Use known relationships to make predictions

# Outline

- Some motivation for visualization

# Outline

- Some motivation for visualization
- Principal Component Analysis

# Visualization of word vectors

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

# 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 |

oil & gas

town & city

# 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

$d > 2$

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

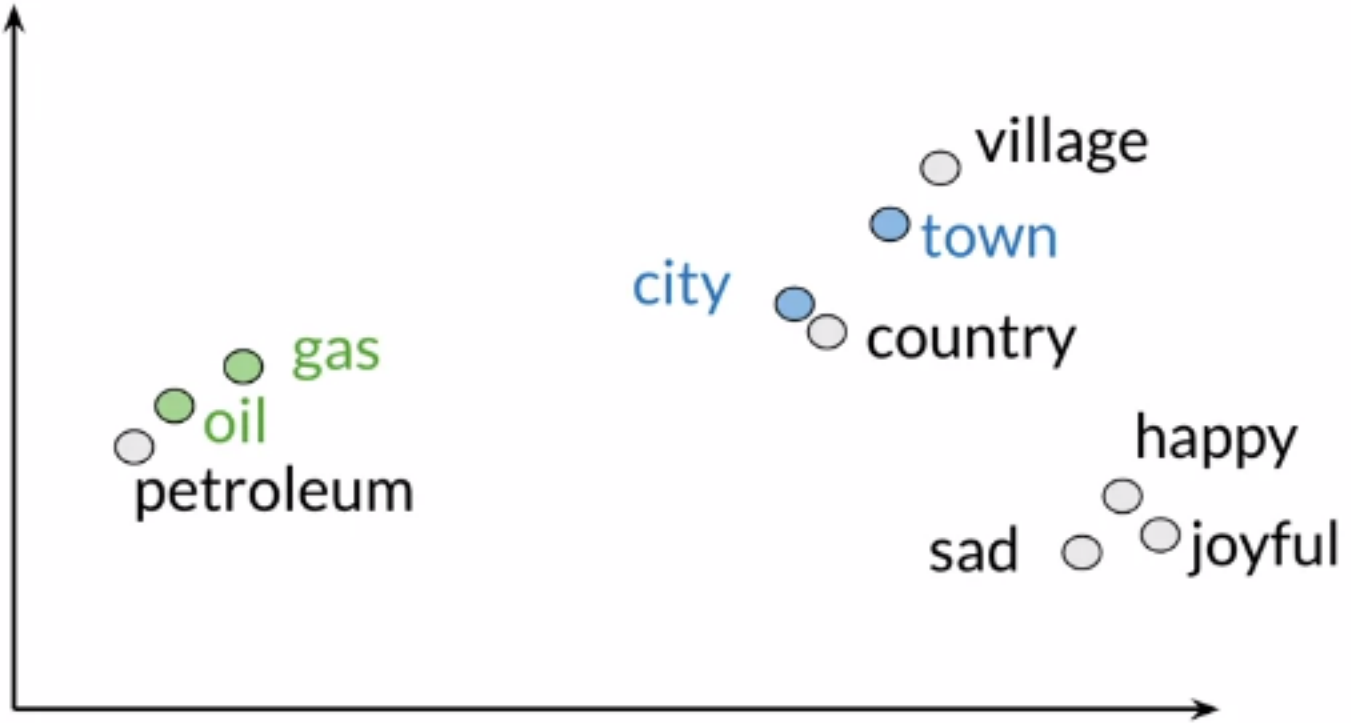
# Visualization of word vectors

| | $d > 2$ | | |
|---|---|---|---|
| oil | 0.20 | ... | 0.10 |
| gas | 2.10 | ... | 3.40 |
| city | 9.30 | ... | 52.1 |
| town | 6.20 | ... | 34.3 |

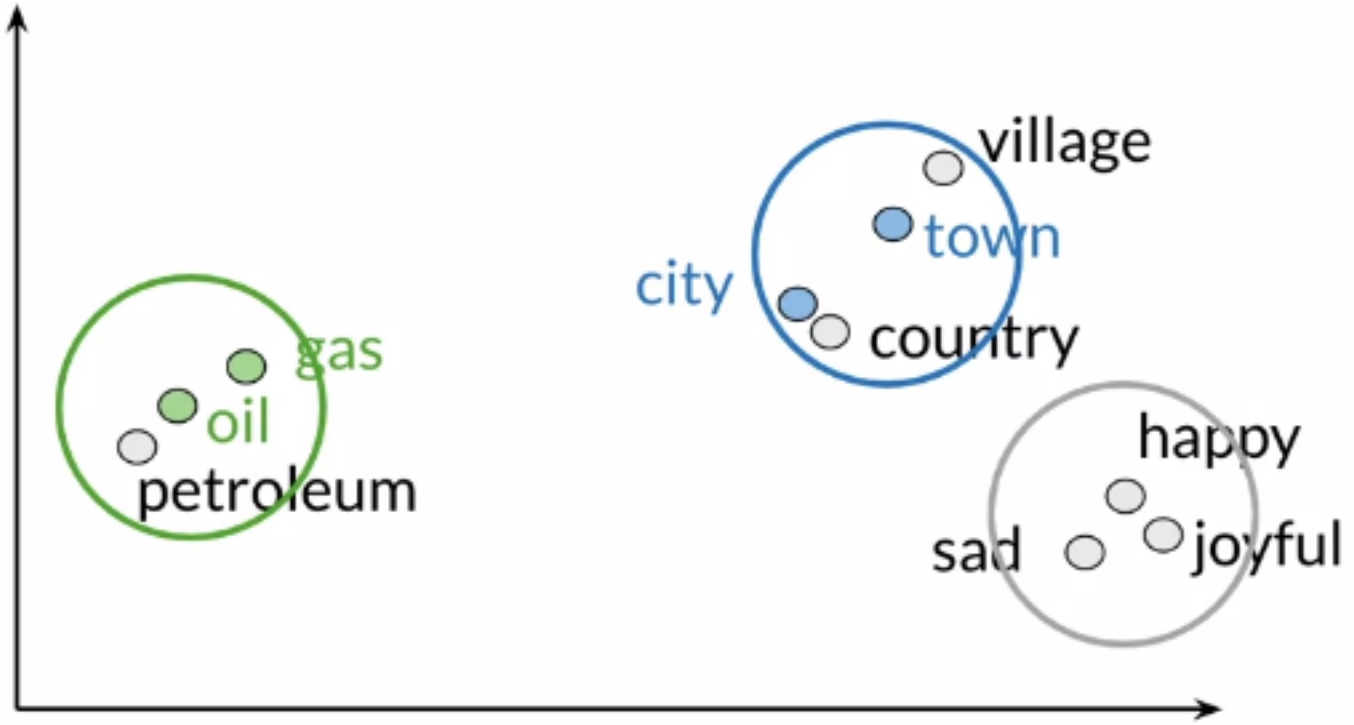
PCA →

| | $d = 2$ | |
|---|---|---|
| oil | 2.30 | 21.2 |
| gas | 1.56 | 19.3 |
| city | 13.4 | 34.1 |
| town | 15.6 | 29.8 |

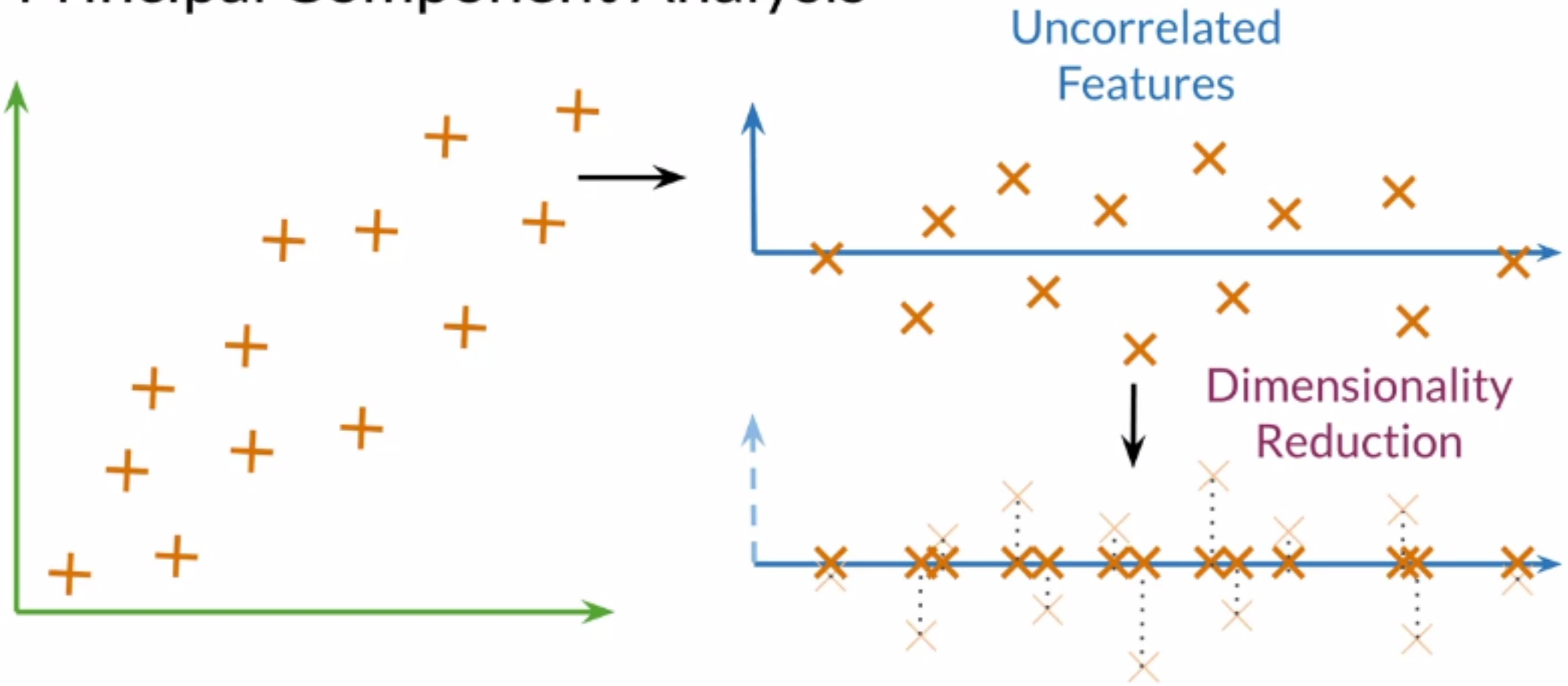
# Visualization of word vectors



# Visualization of word vectors



# Principal Component Analysis



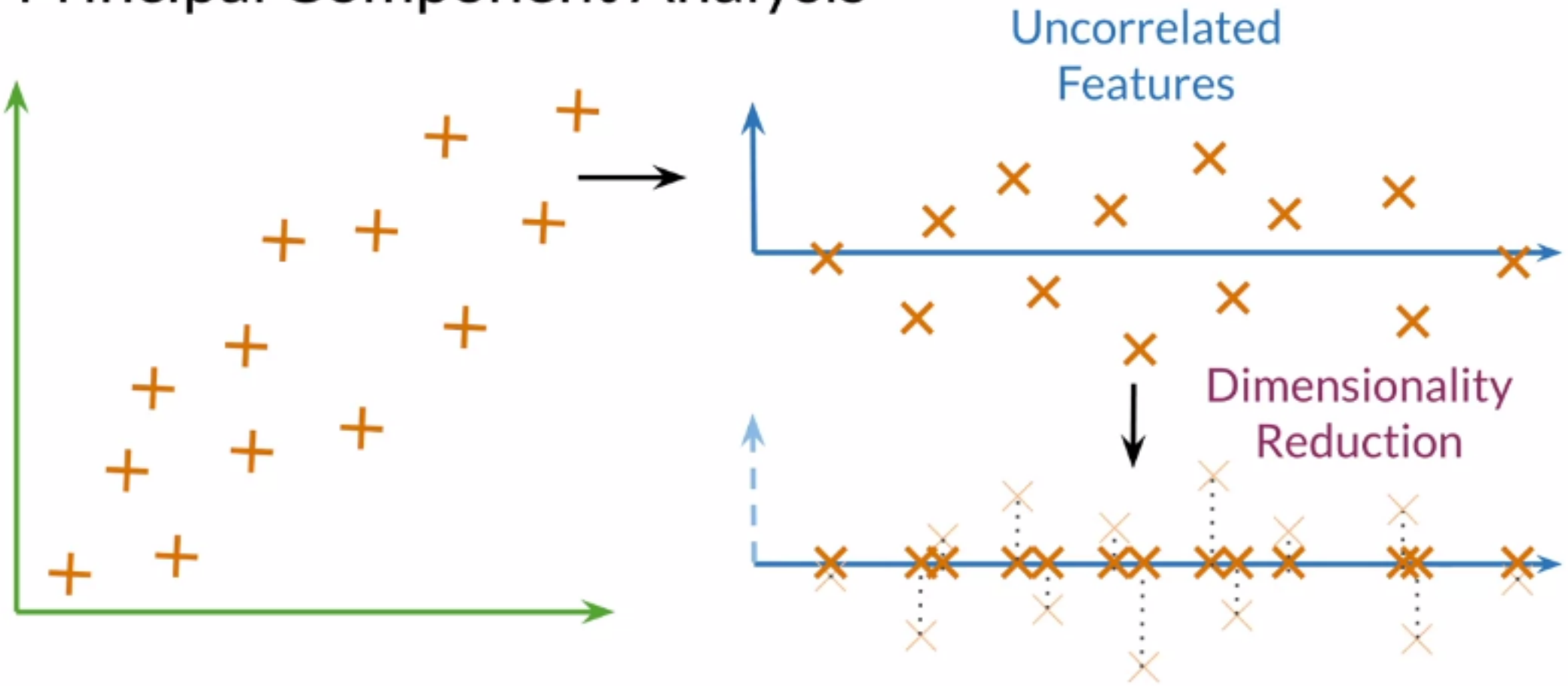
# Summary

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

# Outline

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

# Principal Component Analysis

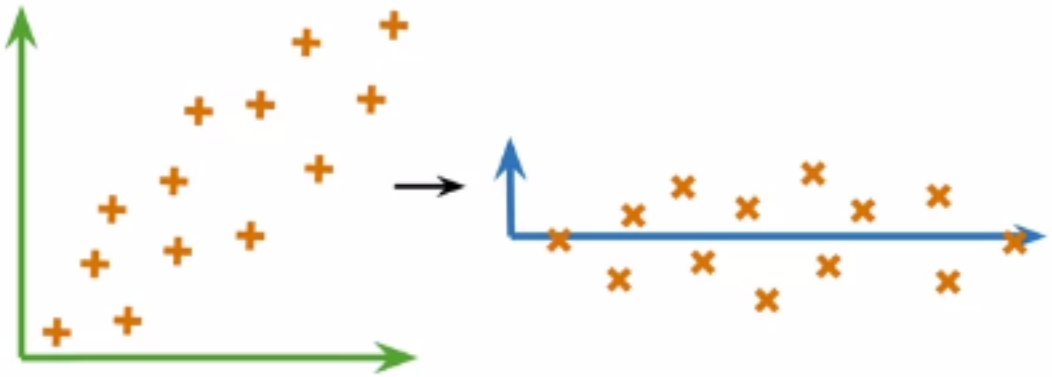


# PCA algorithm

Eigenvector: Uncorrelated features for your data

Eigenvalue: the amount of information retained by each feature

# PCA algorithm



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

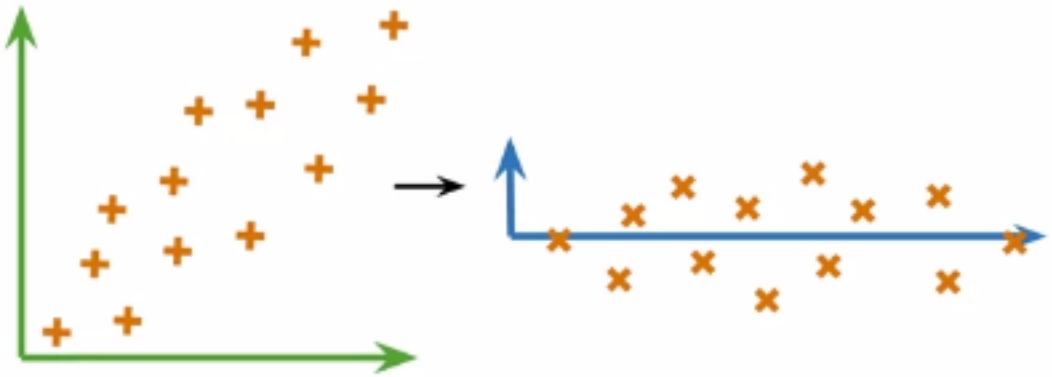
↓

Get Covariance Matrix $\Sigma$

↓

Perform SVD $\text{SVD}(\Sigma)$

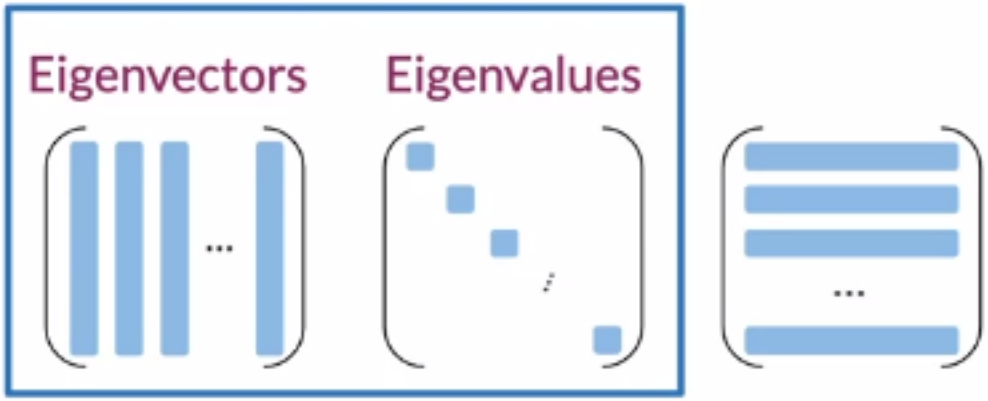
# PCA algorithm



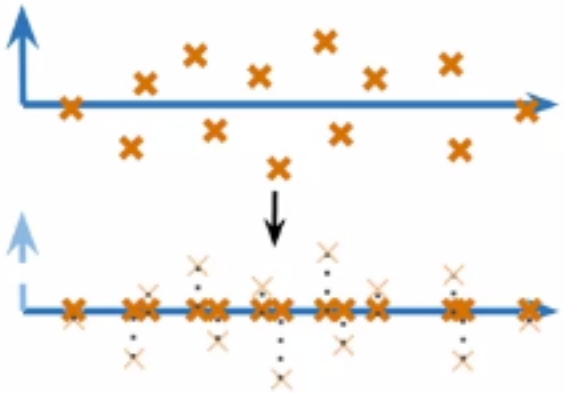
Mean Normalize Data $x_i = \frac{x_i - \mu_{x_i}}{\sigma_{x_i}}$

Get Covariance Matrix $\Sigma$

Perform SVD $\text{SVD}(\Sigma)$



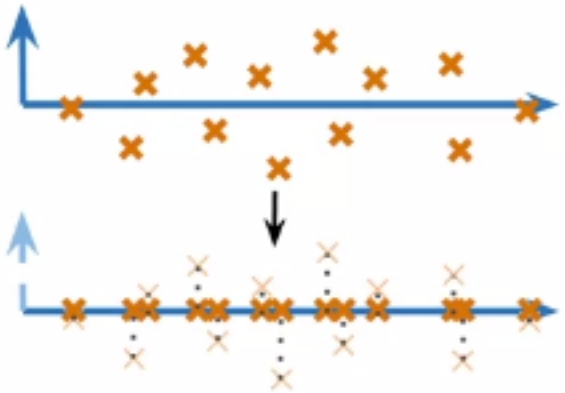
# PCA algorithm



Eigenvectors Eigenvalues

$$\begin{pmatrix} | & | & | & | \\ U_{\cdots} & & & \\ | & | & | & | \end{pmatrix} \quad \begin{pmatrix} \blacksquare & & & \\ & \blacksquare & & \\ & & S_{ii} & \\ & & & \blacksquare \end{pmatrix}$$

# PCA algorithm



Eigenvectors $U_{...}$ Eigenvalues $S_{.}$

Dot Product to Project Data $X' = XU[:, 0:2]$

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

# Summary

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