50.007 Machine Learning

Dimensionality Reduction

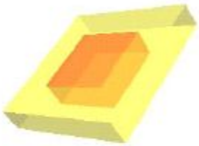
Roy Ka-Wei Lee
Assistant Professor, DAI/ISTD, SUTD

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

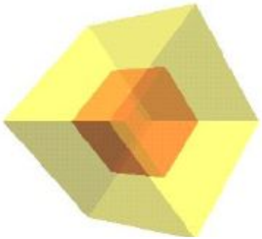
Curse of Dimensionality

- When dimensionality increases, data becomes increasingly sparse in the space that it occupies
- Definitions of distance and density between points become less meaningful in high dimensional space
- In very high-dimensional space, almost every point lie at the edge of the space, far away from the center

1-D, 50% data near edges



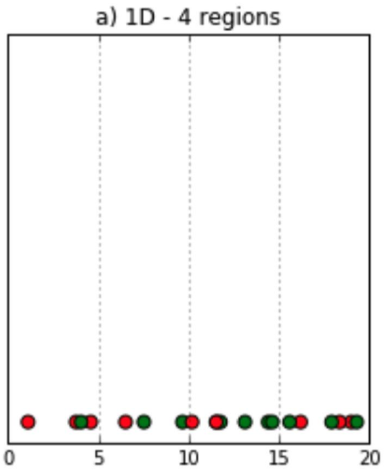
2-D, 74% data near edges



3-D, 87.5% data near edges

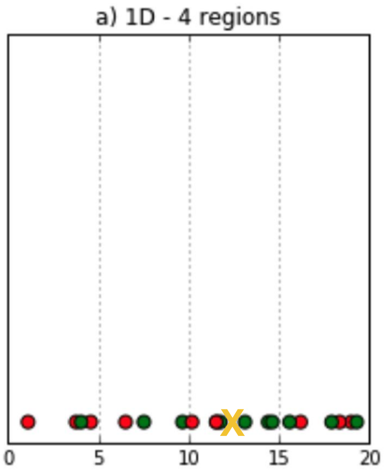
Curse of Dimensionality

- Example: We have 10 data points
    - Each point represent we should go out and catch Pokemon; Green for “go” and red for “nah”
- The data points are represent in one dimension space, i.e., there is only 1 feature in the dataset (e.g. outdoor temperature)



Curse of Dimensionality

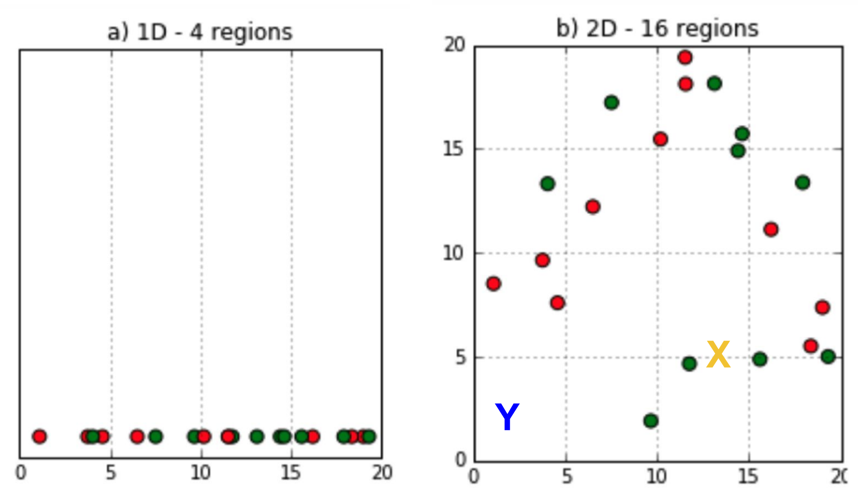
- Example: We have 10 data points
  - Each point represent we should go out and catch Pokemon; Green for “go” and red for “nah”
- The data points are represent in one dimension space, i.e., there is only 1 feature in the dataset (e.g. outdoor temperature)



- Guess X is likely to be green (base on the observations in the region)

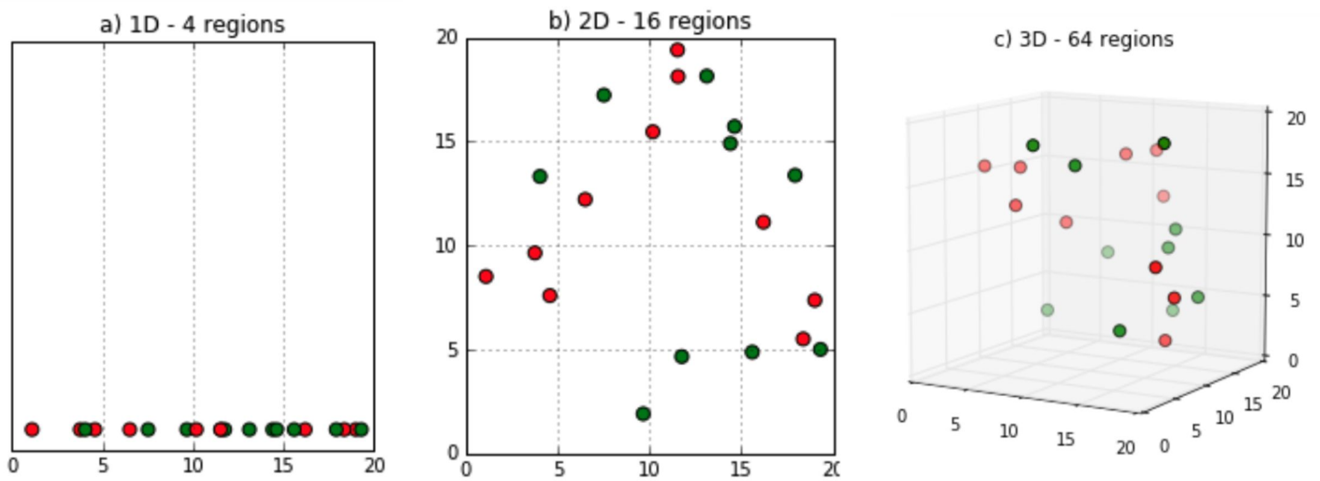
Curse of Dimensionality

- Now let's add more more dimension, i.e., one more feature (e.g. humidity)
- Now the dimension space becomes 10*10 = 100. However the data we still only have 10 data points!



Curse of Dimensionality

- Exponential growth in dimension space cause high sparsity in data set!
- This sparsity is problematic for any method that require statistical significance



Dimensionality Reduction

- Purposes:
  - Avoid curse of dimensionality
  - Reduce amount of time and memory required by machine learning algorithms
  - Allow data to be more easily visualized
  - May help to eliminate irrelevant features or reduce noise

- Simple Techniques
  - Feature Extraction
  - Feature Selection

Example

- Predict if you will over-eat during the long weekend

ID	Height	Weight	Age	Gender	Address	Postal Code
1	155	53.1	21	F	Blah Blah	123456
2	167	76.5	20	M	Blah Blah	654321
3	178	79.2	23	M	Blah Blah	246802
. .	. .	. .	. .	. .	. .	. .

Feature Selection

- Predict if you will over-eat during the long weekend

ID	Height	Weight	Age	Gender	Address	Postal Code
1	155	53.1	21	F	Blah Blah	123456
2	167	76.5	20	M	Blah Blah	654321
3	178	79.2	23	M	Blah Blah	246802
. .	. .	. .	. .	. .	. .	. .

Feature Selection: Select features that are more likely to be useful for this prediction

Feature Extraction

- Predict if you will over-eat during the long weekend

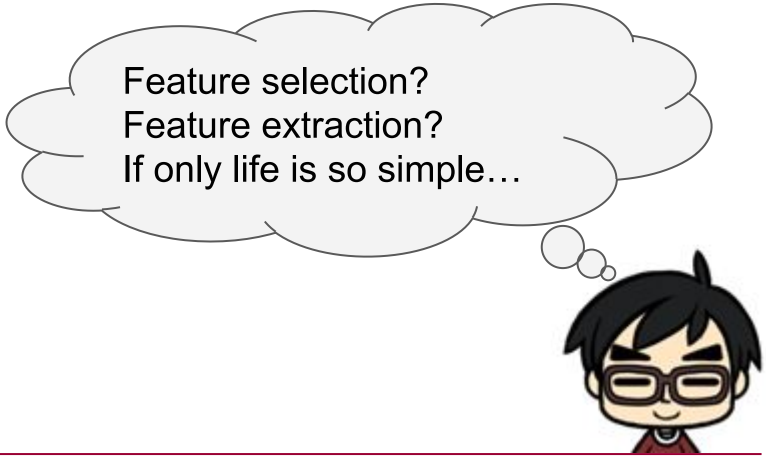
ID	Height	Weight	Age	Gender	Address	Postal Code
1	155	53.1	21	F	Blah Blah	123456
2	167	76.5	20	M	Blah Blah	654321
3	178	79.2	23	M	Blah Blah	246802
. .	. .	. .	. .	. .	. .	. .

Feature Extraction: Combine features to create new ones with the goal of reduce dimensionality: e.g. Combine “Height” and “Weight” to get “BMI”

Recall TF-IDF

- How we to do dimension reduction for the 5000 TF-IDF features in your project?

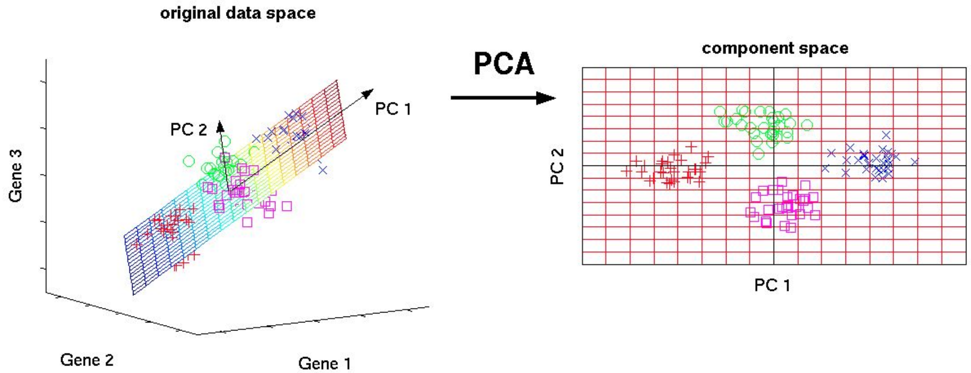
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	id	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	17185	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	17186	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	17187	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	17188	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	17189	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	17190	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	17191	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	17192	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	17193	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	17194	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	17195	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	17196	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	17197	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	17198	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	17199	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	17200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	17201	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	17202	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	17203	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	17204	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	17205	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	17206	0	0	0.29318822	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	17207	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	17208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	17209	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	17210	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	17211	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	17212	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	17213	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	17214	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN

Principal Component Analysis (PCA)

- **What is PCA**: Unsupervised technique for extracting variance structure from high dimensional datasets.



- PCA is an orthogonal projection or transformation of the data into a (possibly lower dimensional) subspace so that the variance of the projected data is maximized.

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

“Taking a photo of your data”

- Imaging taking a photo of your class

“Taking a photo of your data”

- Imaging taking a photo of your class

“Taking a photo of your data”

- Imaging taking a photo of your class

Inspired by video from Lui Serrano: https://www.youtube.com/watch?v=g-Hb26agBFg

“Taking a photo of your data”

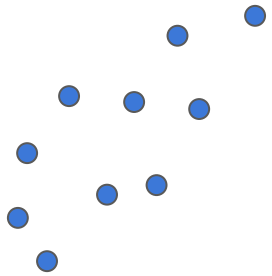
- Imaging taking a photo of your class

Inspired by video from Lui Serrano: https://www.youtube.com/watch?v=g-Hb26agBFg

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

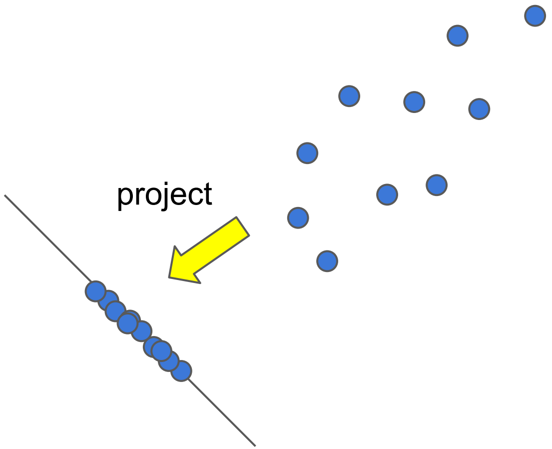
“Taking a photo of your data”

- Imaging taking a photo of data



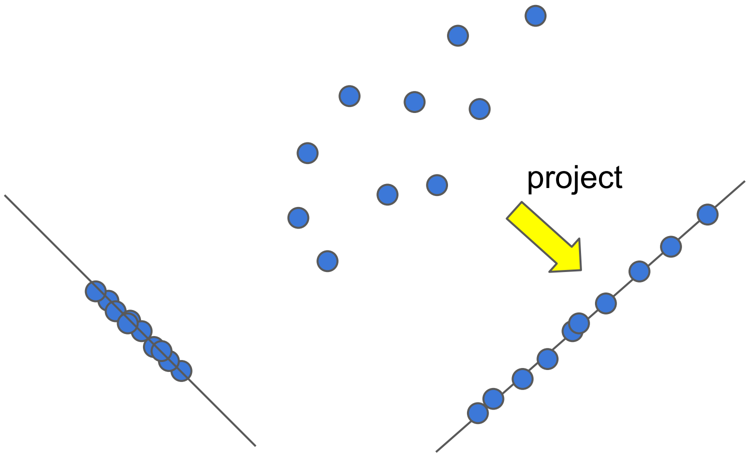
“Taking a photo of your data”

- Imaging taking a photo of data



“Taking a photo of your data”

- Imaging taking a photo of data

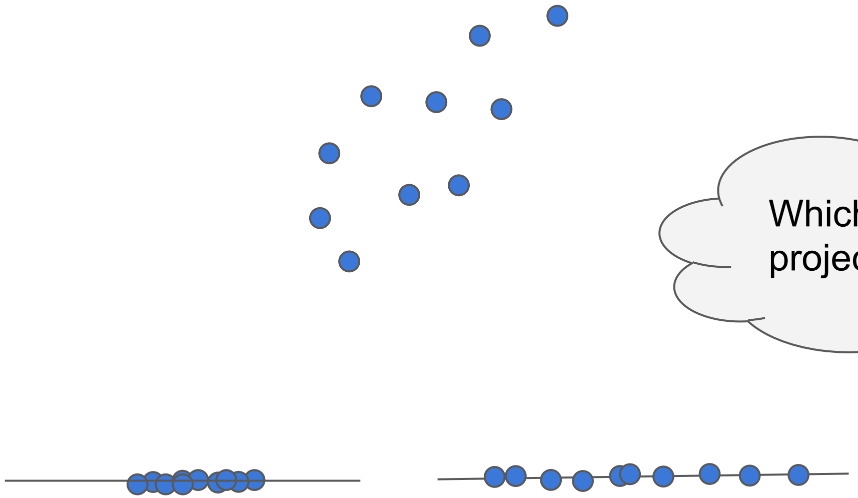


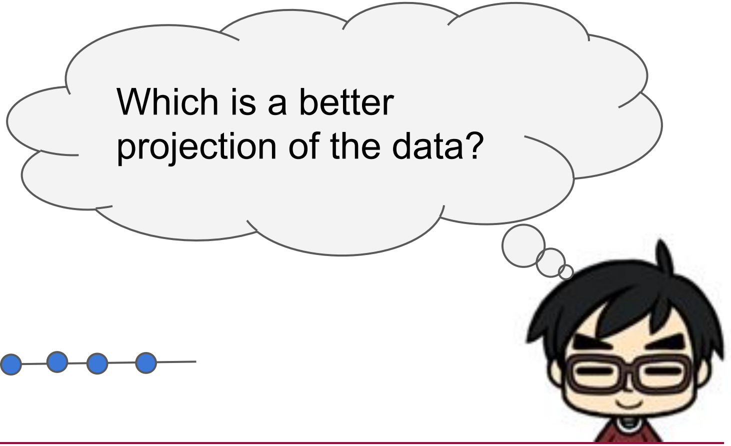
Inspired by video from Lui Serrano: https://www.youtube.com/watch?v=g-Hb26agBFg

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

“Taking a photo of your data”

- Imaging taking a photo of data

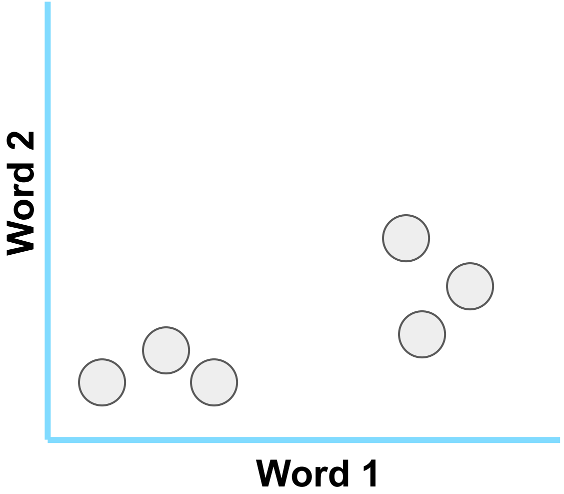




Step-by-Step PCA

- Step-by-step PCA using Singular Value Decomposition (SVD)
- Consider the below example with TF-IDF scores of 6 posts and 2 words

	Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
Word 1	0.2	0.4	0.1	0.7	0.5	0.6
Word 2	0.1	0.3	0.3	0.5	0.3	0.4

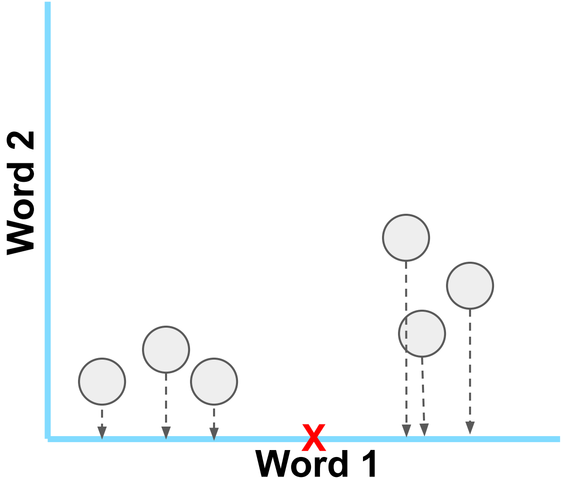


Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

Step-by-Step PCA

- Calculate the average score for Word 1

	Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
Word 1	0.2	0.4	0.1	0.7	0.5	0.6
Word 2	0.1	0.3	0.3	0.5	0.3	0.4

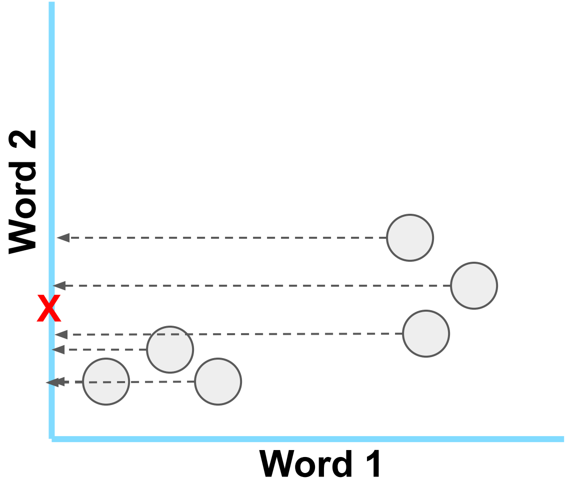


Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

Step-by-Step PCA

- Calculate the average score for Word 2

	Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
Word 1	0.2	0.4	0.1	0.7	0.5	0.6
Word 2	0.1	0.3	0.3	0.5	0.3	0.4

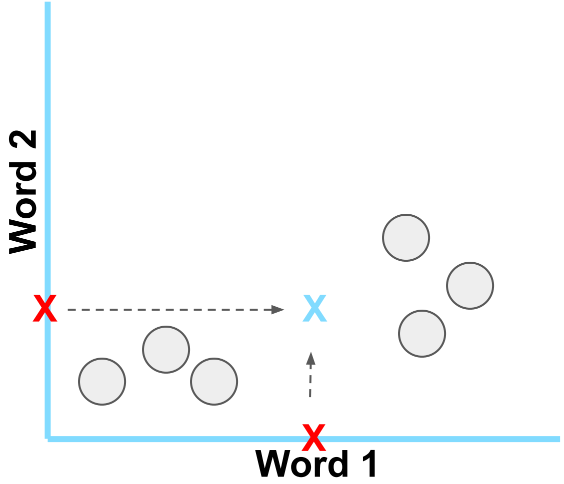


Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

Step-by-Step PCA

- Calculate the center of the data using the mean scores

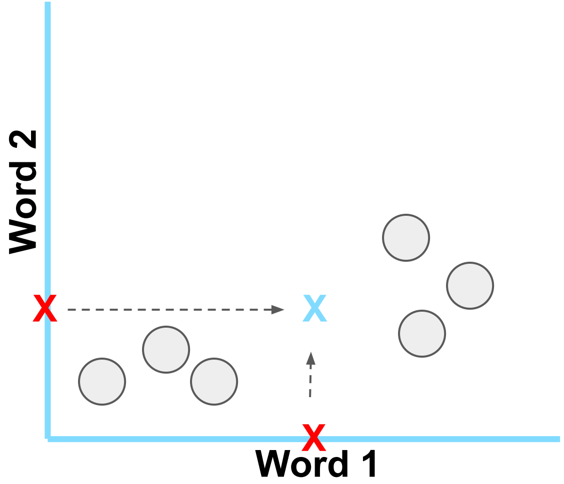
	Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
Word 1	0.2	0.4	0.1	0.7	0.5	0.6
Word 2	0.1	0.3	0.3	0.5	0.3	0.4



Step-by-Step PCA

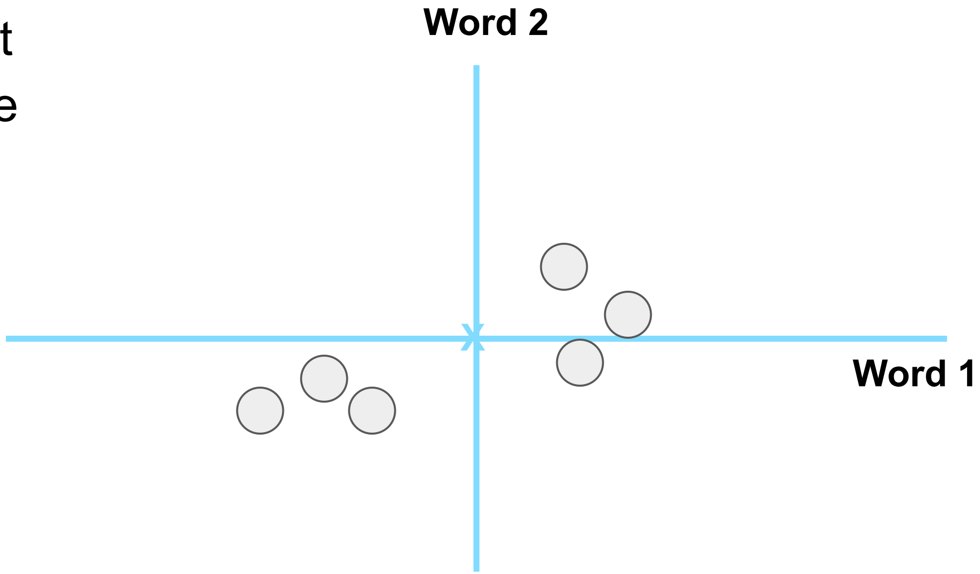
- From this point onwards, we will focus on the graph and put aside the original data (for discussion)

	Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
Word 1	0.2	0.4	0.1	0.7	0.5	0.6
Word 2	0.1	0.3	0.3	0.5	0.3	0.4



Step-by-Step PCA

- Shift the data so that the center is on the origin (0,0) in the graph
- **NOTE**: Shifting the data did not change how the data points are position relative to each other!

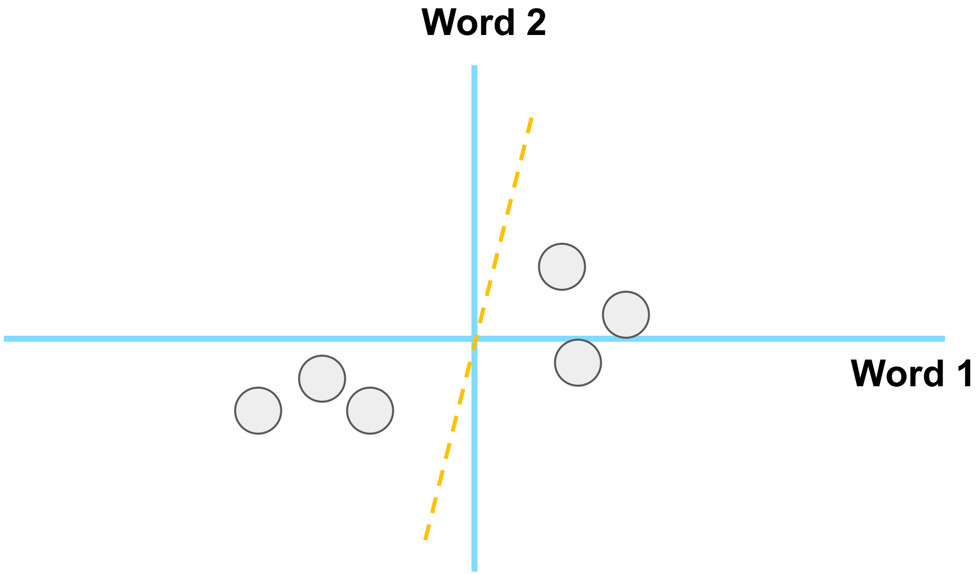


Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Step-by-Step PCA

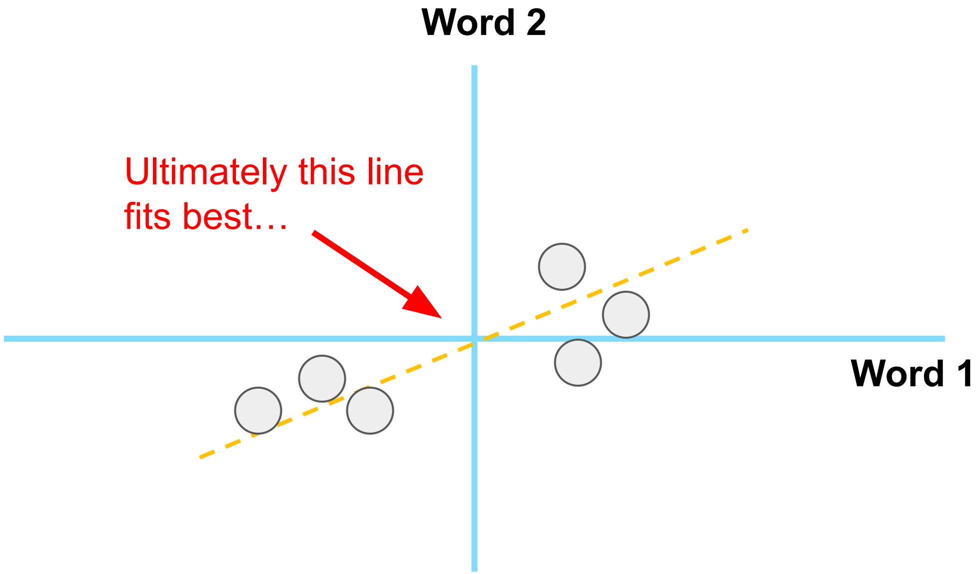
- Try to fit a line to the graph
- Start by drawing a random **line** that goes through the origin



Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

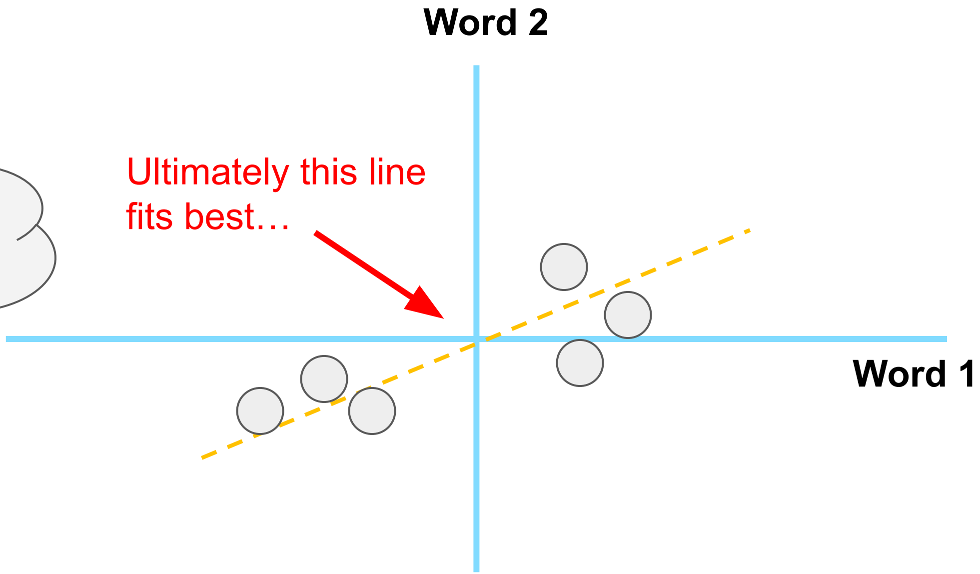
Step-by-Step PCA

- Then we rotate the line until it fits the data as well as it can, given that it has to go through the origin



Step-by-Step PCA

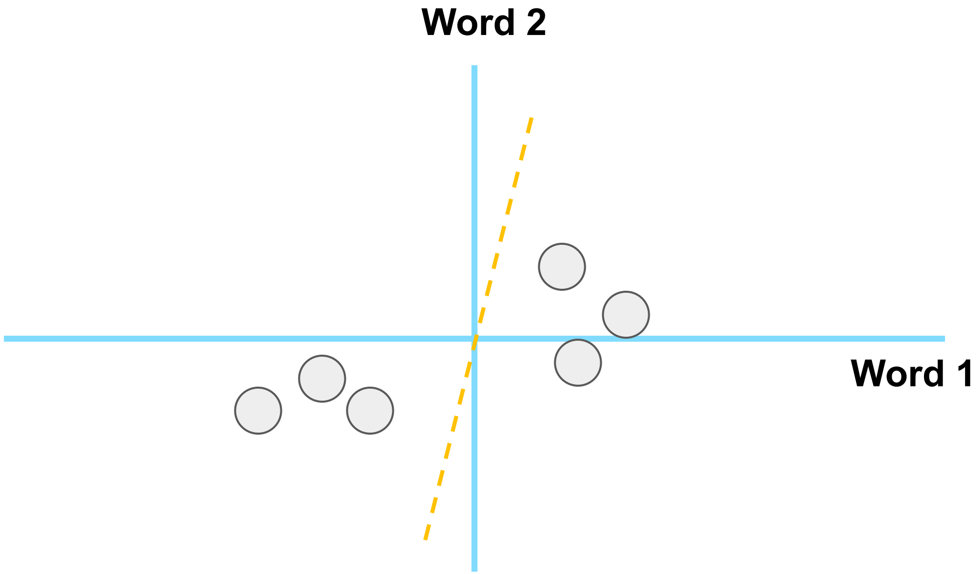
- Then we rotate the line until it fits the data as well as it can, given that it has to go through the origin



Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

How do we know best fit?

- Let's go back to the random line that goes through the origin

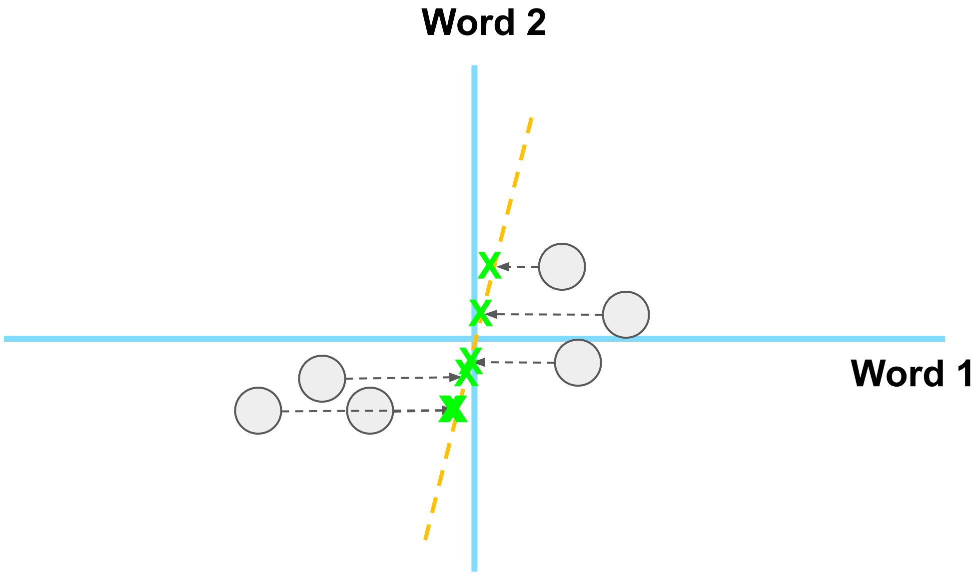


Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

How do we know best fit?

- PCA wil first project the data onto the line

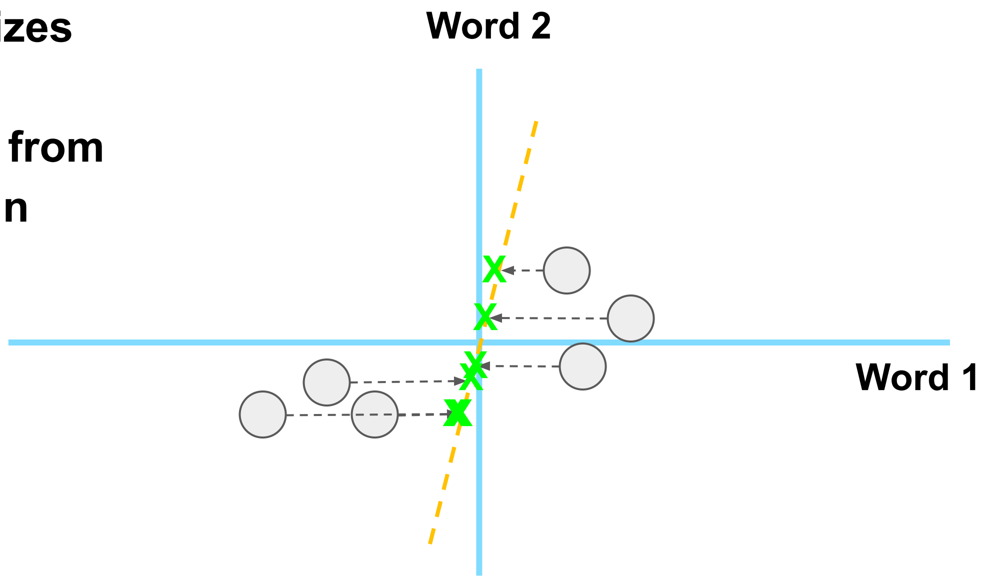


Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

How do we know best fit?

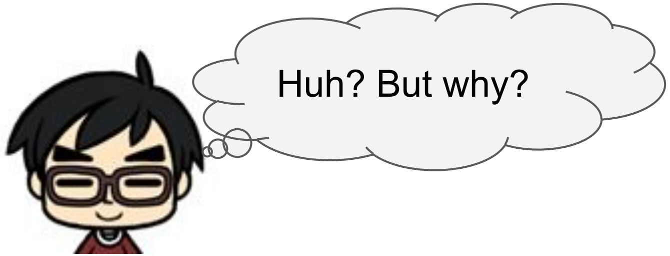
- Then it can either measure the **distance from the data to the line** and try to find the line that **minimizes** those distances
- Or find the measure the **distance from the projected points to the origin** and **maximize** those distances

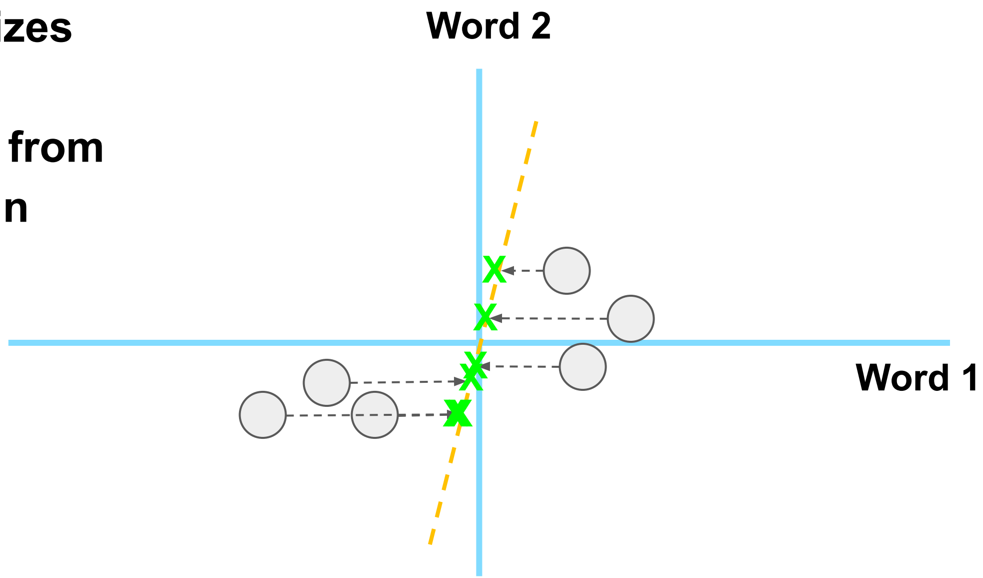


Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

How do we know best fit?

- Then it can either measure the **distance from the data to the line** and try to find the line that **minimizes** those distances
- Or find the measure the **distance from the projected points to the origin** and **maximize** those distances

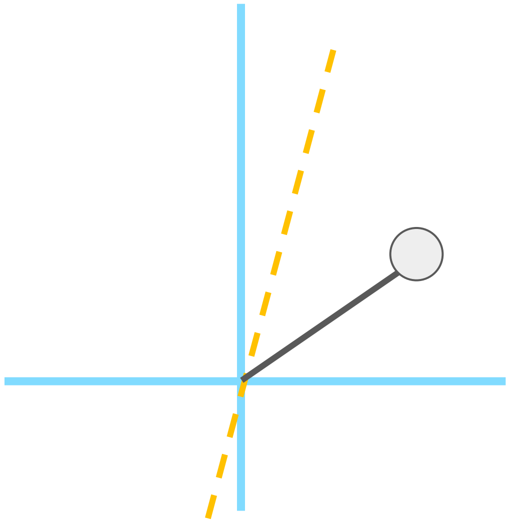




Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

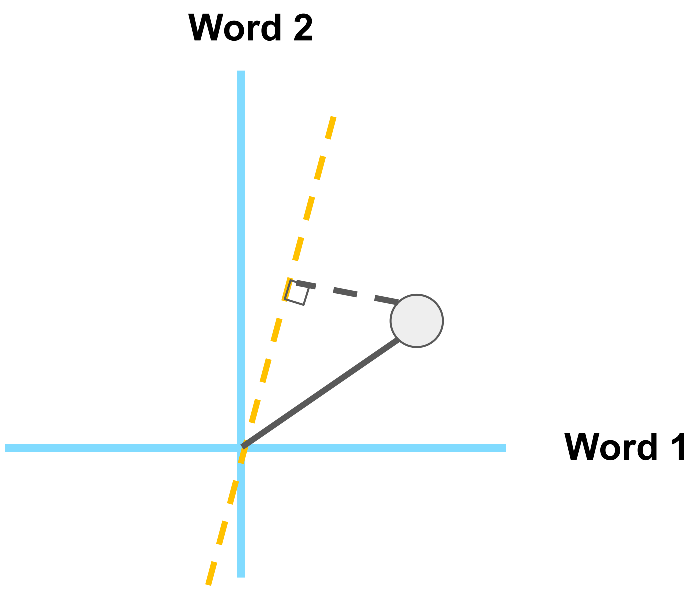
How do we know best fit?

- Let’s zoom in and consider a single data point
- This data point is fix (same distance to the origin no matter how the dotted line rotate)



How do we know best fit?

- When we project the data onto the orange dotted line, we get a right angle between the black dotted line and the orange dotted line

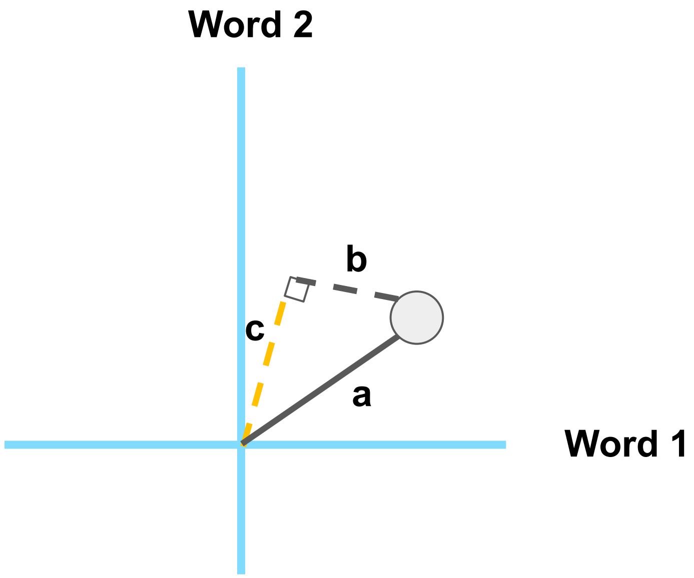


Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

How do we know best fit?

- If we label the sides of the right angle triangle…
- Then we can use the Pythagorean theorem to show how **b** and **c** are inversely related

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

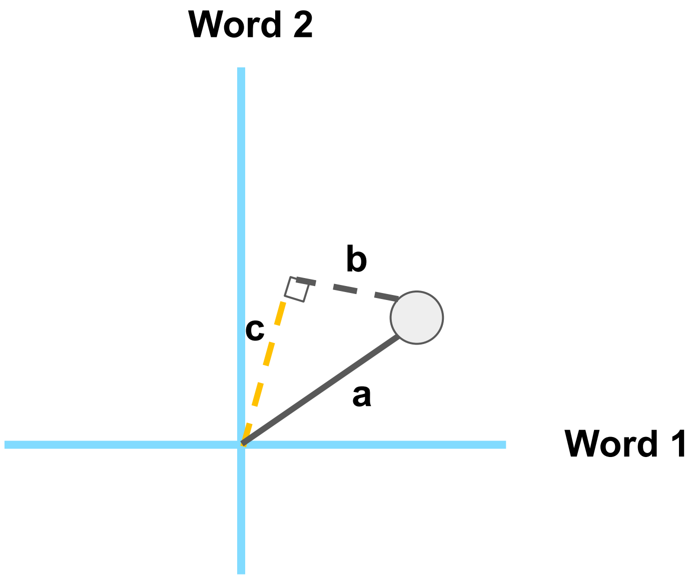


How do we know best fit?

- If we label the sides of the right angle triangle…
- Then we can use the Pythagorean theorem to show how **b** and **c** are inversely related

$$\mathbf{a^2 = b^2 + c^2}$$

When **c** increase, **b** decrease

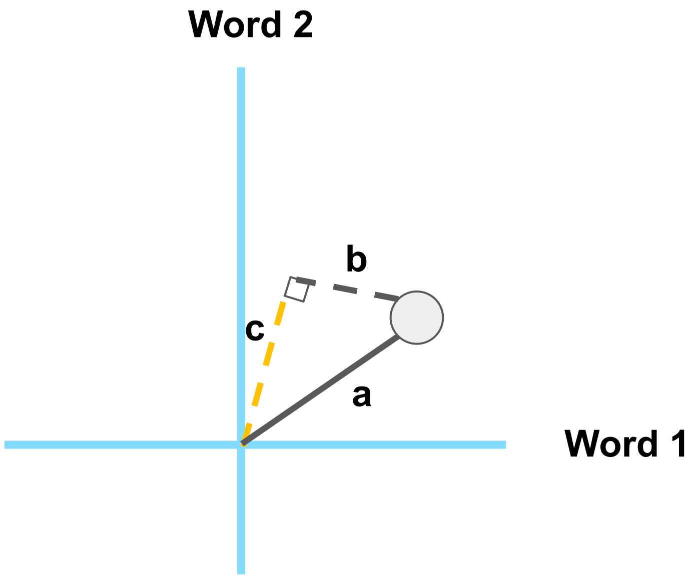


How do we know best fit?

- If we label the sides of the right angle triangle…
- Then we can use the Pythagorean theorem to show how **b** and **c** are inversely related

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

Thus, either maximize **c** or minimize **b**

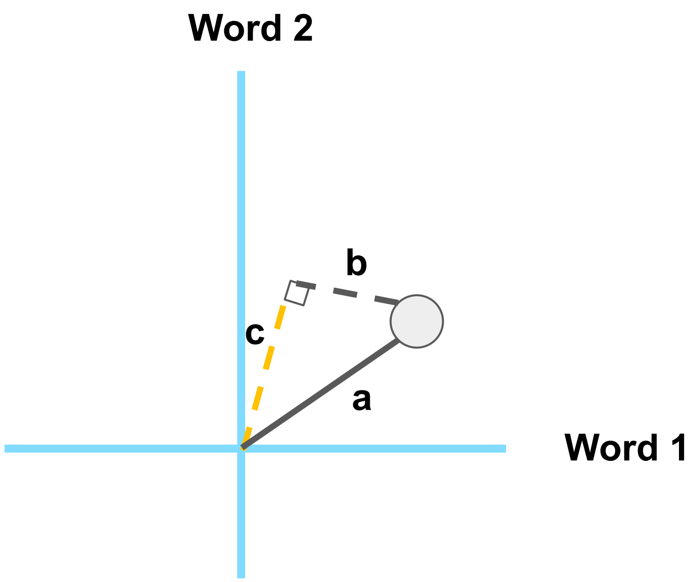


Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

How do we know best fit?

- In actual, PCA find the best line by **maximizing** the sum of square distances from the project points to the origin

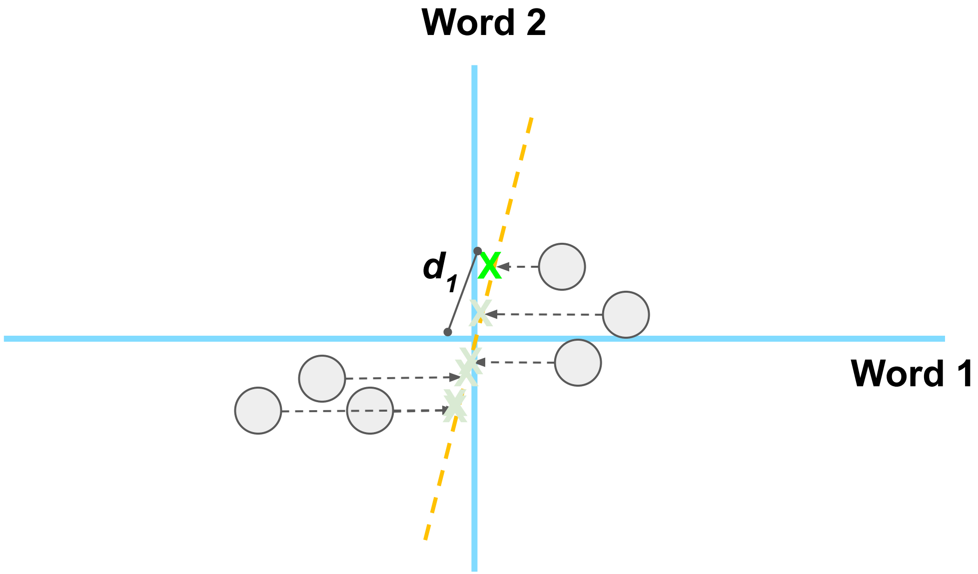
$$a^2 = b^2 + c^2$$



How do we know best fit?

- PCA measures the distance between the projection of the first point to the origin (***d_1***)

d_1

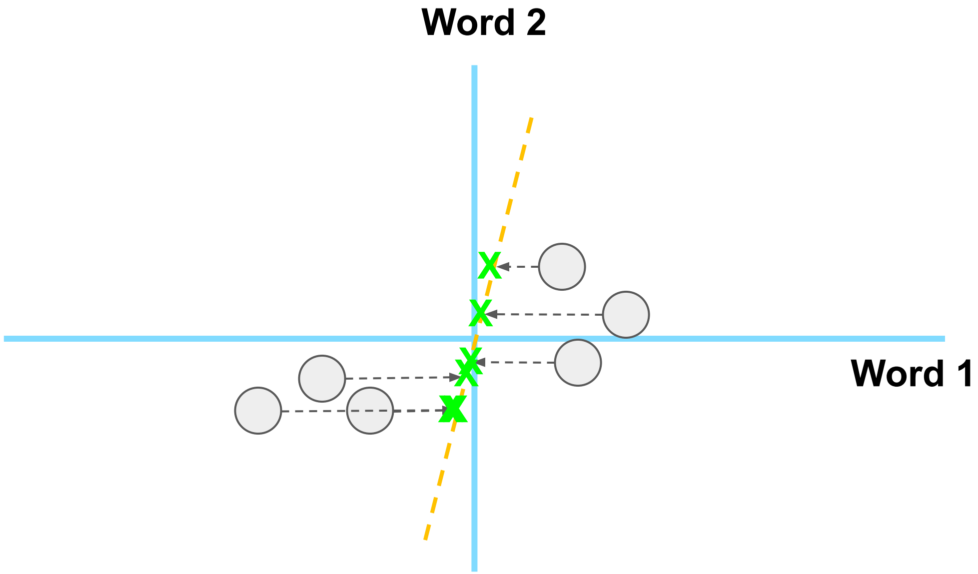


Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

How do we know best fit?

- PCA measures the distance between the projection of the first point to the origin (***d**$_1$*)
- Repeat this for all the other points

d_1 d_2 d_3 d_4 d_5 d_6



Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

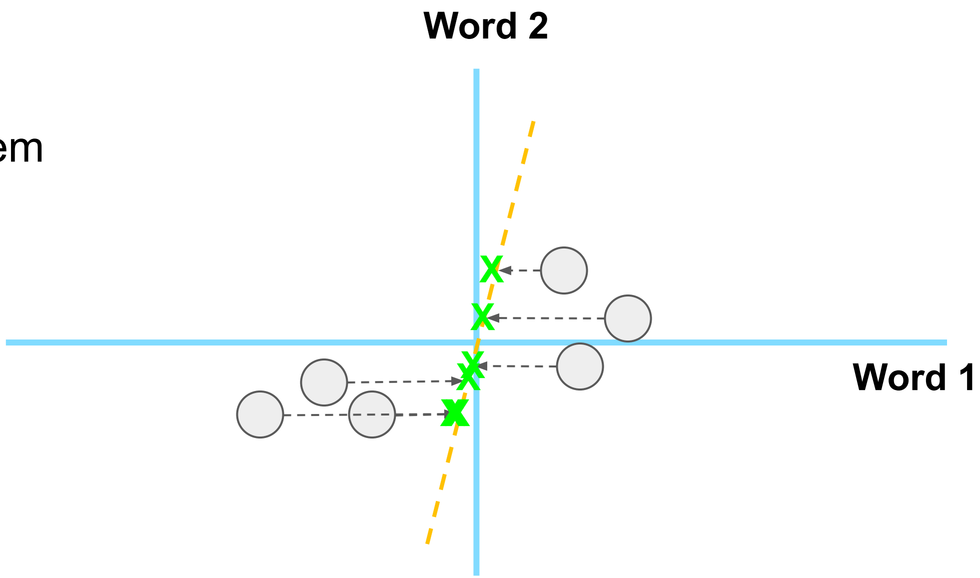
How do we know best fit?

- PCA measures the distance between the projection of the first point to the origin (d_1)
- Repeat this for all the other points
- Square the distances and sum them

$$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$$

= sum of square distances

= SS(distances)



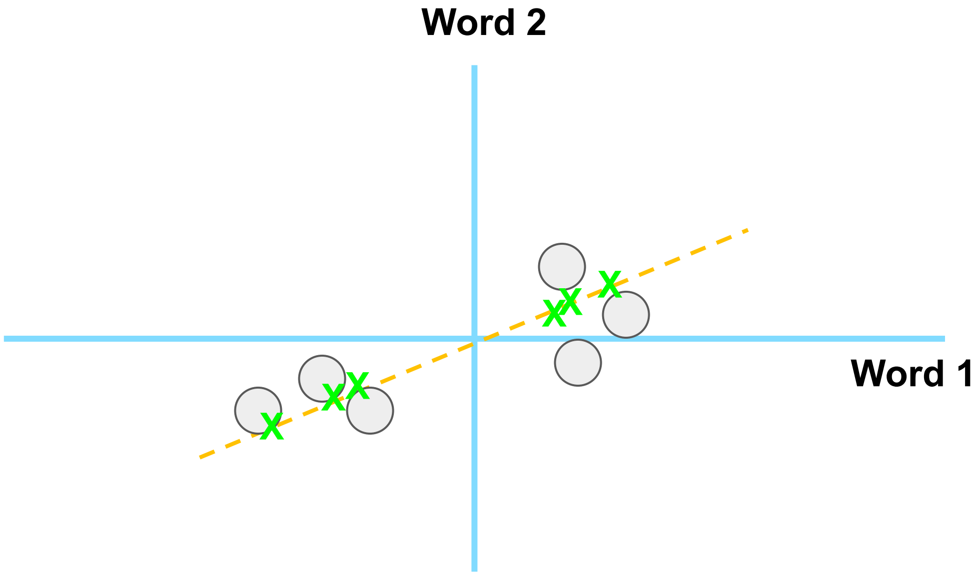
How do we know best fit?

- Rotate the line till we find the line with the largest SS(distances)

$$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$$

= sum of square distances

= SS(distances)



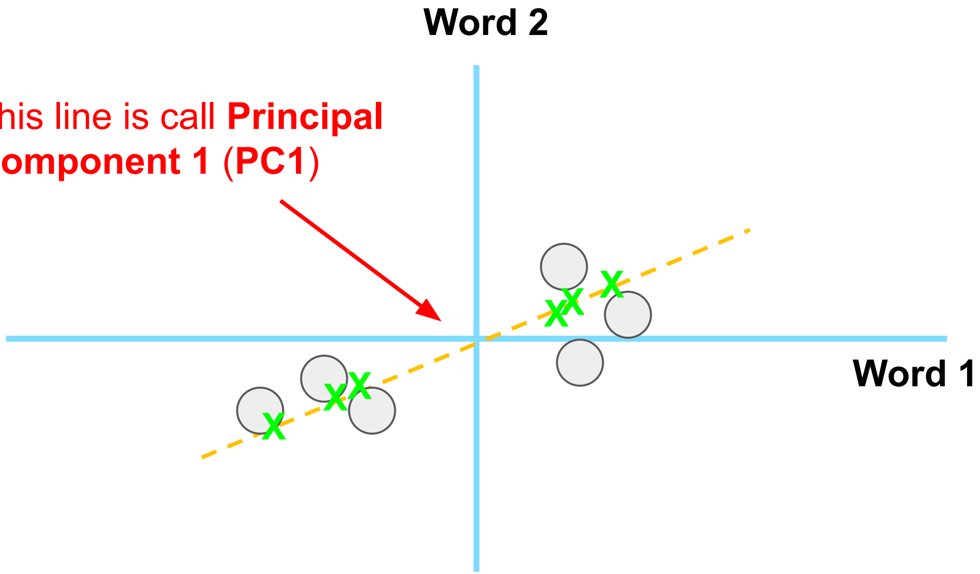
How do we know best fit?

- Rotate the line till we find the line with the largest SS(distances)

$$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$$

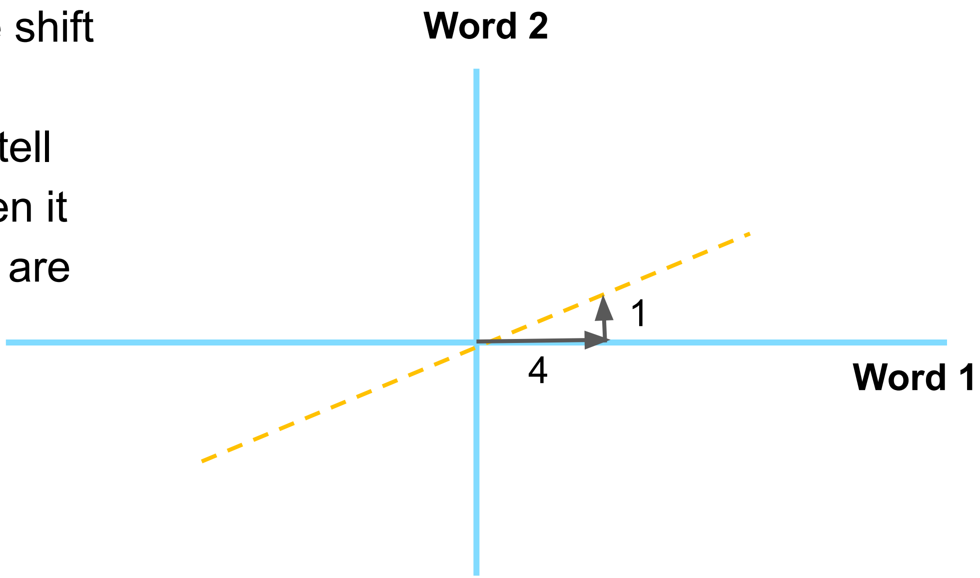
= sum of square distances

= SS(distances)



Describing PC1

- PC1 has a slope of 0.25
- In other words, for every 4 units we shift along the axis for Word 1, we shift 1 unit along Word 2
- The ratio of Word 1: Word 2 (4:1) tell you Word 1 is more important when it comes to describing how the data are spread out
- We call this ratio, the **linear combination** of Word 1 and 2

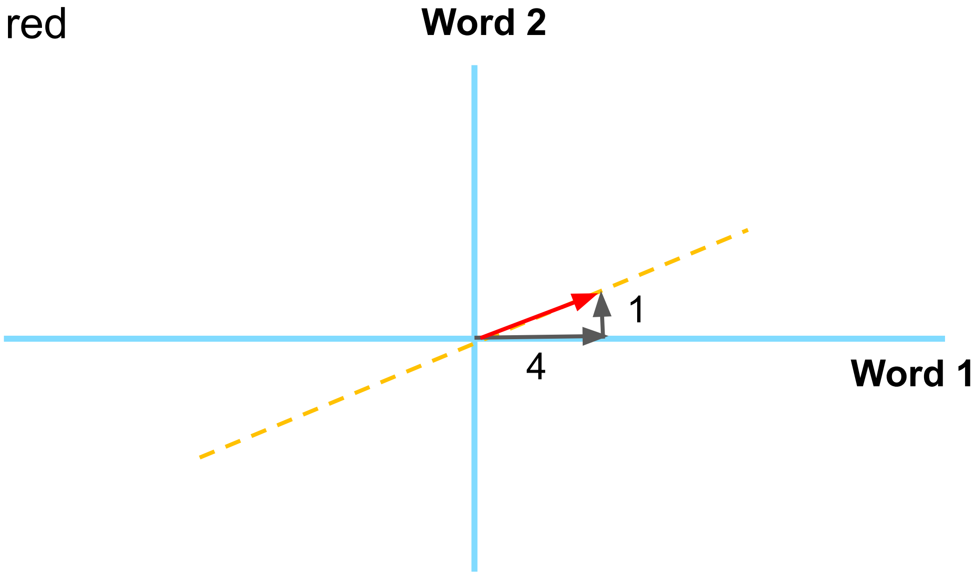


Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Describing PC1

- This linear combination of PC1 can be represented with the red line
- We can solve for the length of the red line using Pythagorean theorem



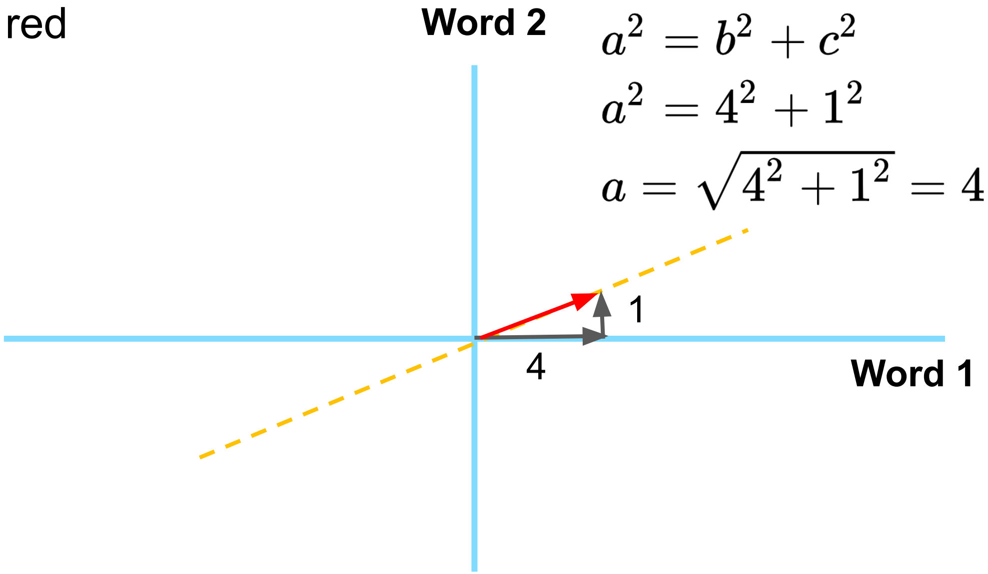
Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Describing PC1

- This linear combination of PC1 can be represented with the red line
- We can solve for the length of the red line using Pythagorean theorem
- Length of the red line is 4.12

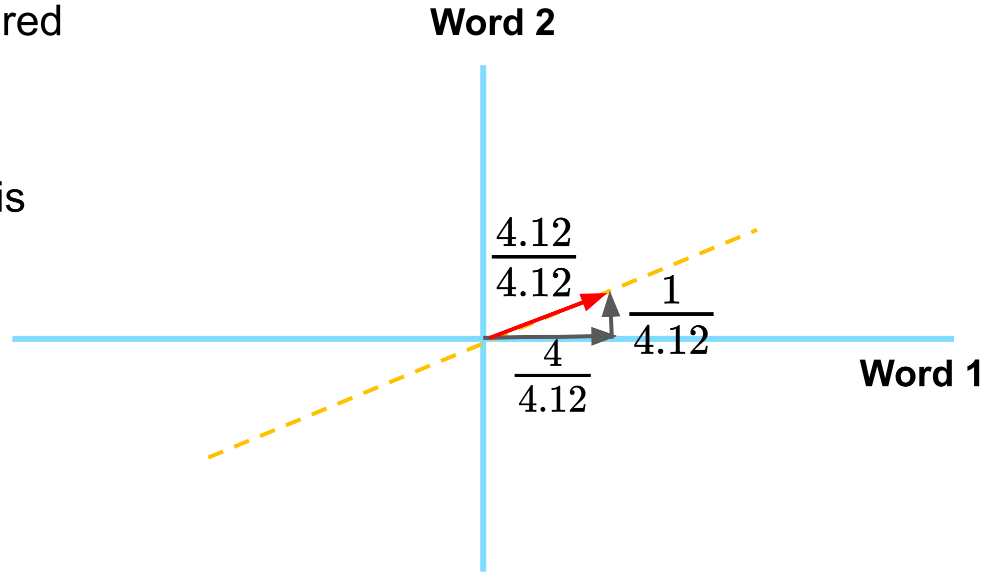
$$a^2 = b^2 + c^2$$
$$a^2 = 4^2 + 1^2$$
$$a = \sqrt{4^2 + 1^2} = 4.12$$



Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

Describing PC1

- This linear combination of PC1 can be represented with the red line
- We can solve for the length of the red line using Pythagorean theorem
- Length of the red line is 4.12
- When we do PCA with SVD, PC1 is scaled so that length of red line **=1**

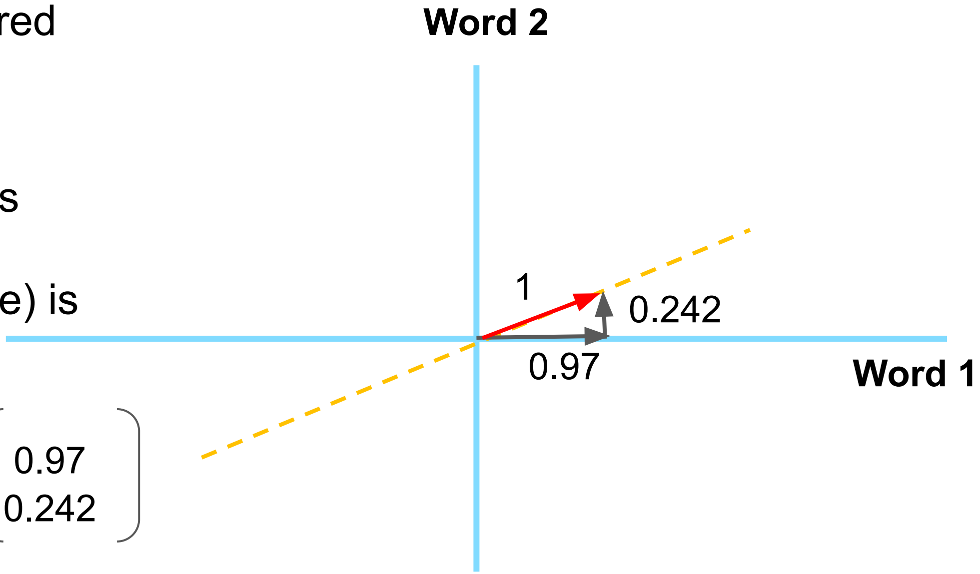


Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

Singular Vector/Eigenvector

- This linear combination of PC1 can be represented with the red line
- We can solve for the length of the red line using Pythagorean theorem
- Length of the red line is 4.12
- When we do PCA with SVD, PC1 is scaled so that length of red line **=1**
- This new 1 unit long vector (red line) is call the **Singular Vector** or the **Eigenvector** for **PC1**

$$\begin{pmatrix} \text{Word 1} \\ \text{Word 2} \end{pmatrix} \quad \begin{pmatrix} 0.97 \\ 0.242 \end{pmatrix}$$



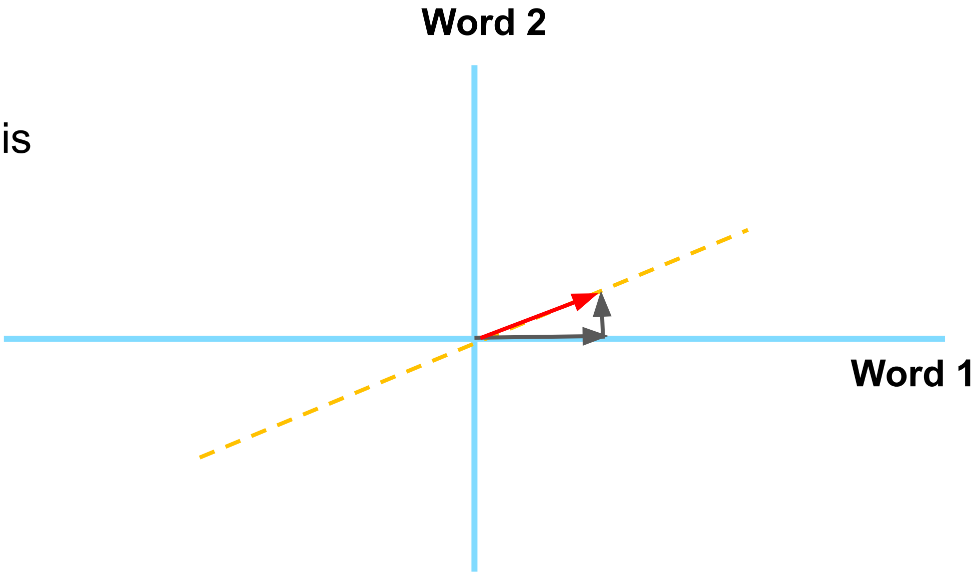
Singular Value/Eigenvalue

- PCA calls the SS(distances) for the best fit line the **Eigenvalue for PC1**

$$SS(\text{distance for PC1})$$

- The square root of the Eigenvalue is **Singular Value for PC1**

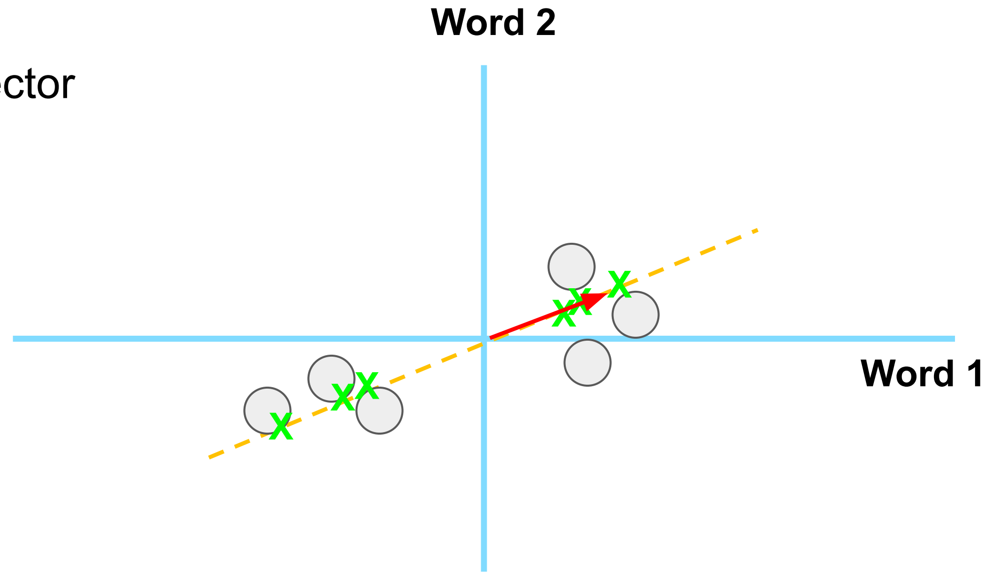
$$\sqrt{\text{Eigenvalue for PC1}}$$



Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

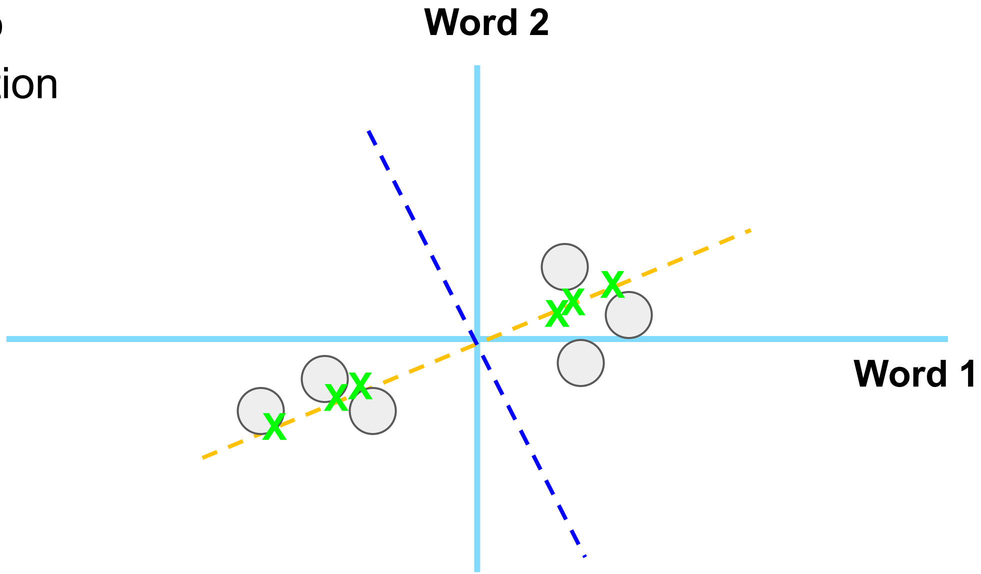
What we have done so far…

- So now we are back to looking at…
  - The data
  - The best fitting line
  - The Singular Vector/Eigenvector of PC1



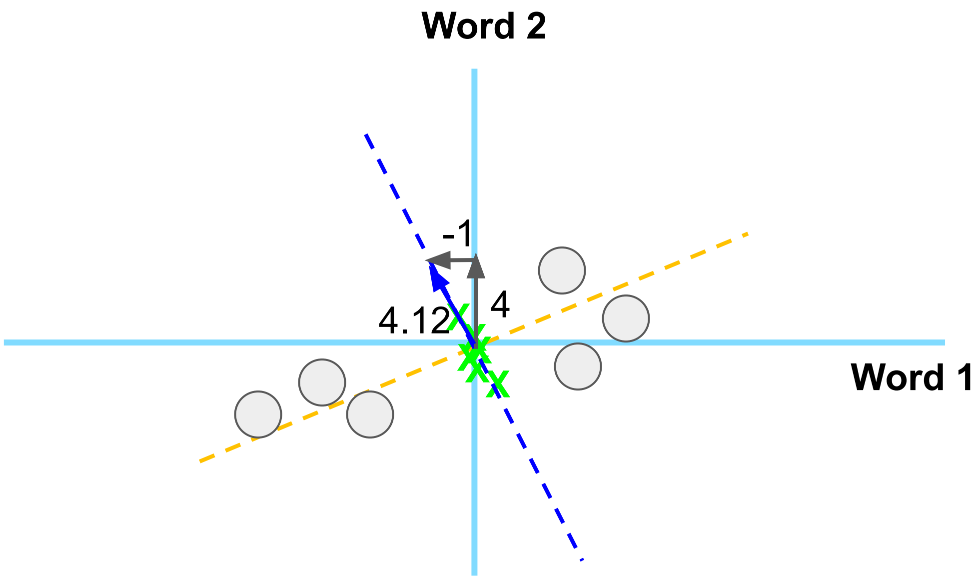
Finding PC2

- This is a 2D graph (only two word feature), **PC2** is simply the line through the origin that is **perpendicular** to **PC1**, without any further optimization



Finding PC2

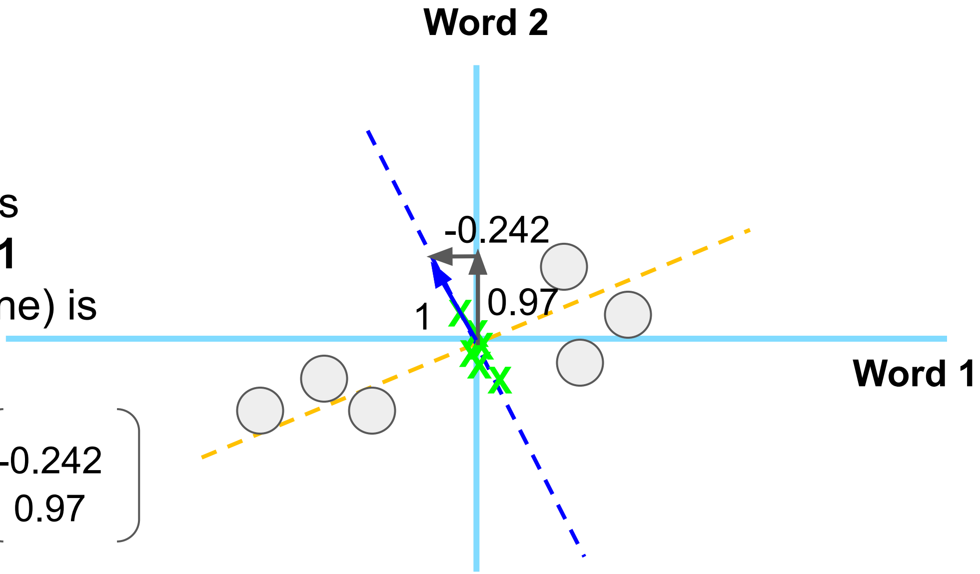
- Linear combination/ ratio of Word 1: Word 2 for PC2 is -1: 4
- This linear combination of PC2 is represented using the blue line
- Length of blue line is **4.12**



Finding PC2

- Linear combination/ ratio of Word 1: Word 2 for PC2 is -1: 4
- This linear combination of PC2 is represented using the blue line
- Length of blue line is **4.12**
- When we do PCA with SVD, PC2 is scaled so that length of blue line **=1**
- This new 1 unit long vector (blue line) is call the **Singular Vector** or the **Eigenvector** for **PC2**

$$\begin{pmatrix} \text{Word 1} \\ \text{Word 2} \end{pmatrix} \begin{pmatrix} -0.242 \\ 0.97 \end{pmatrix}$$



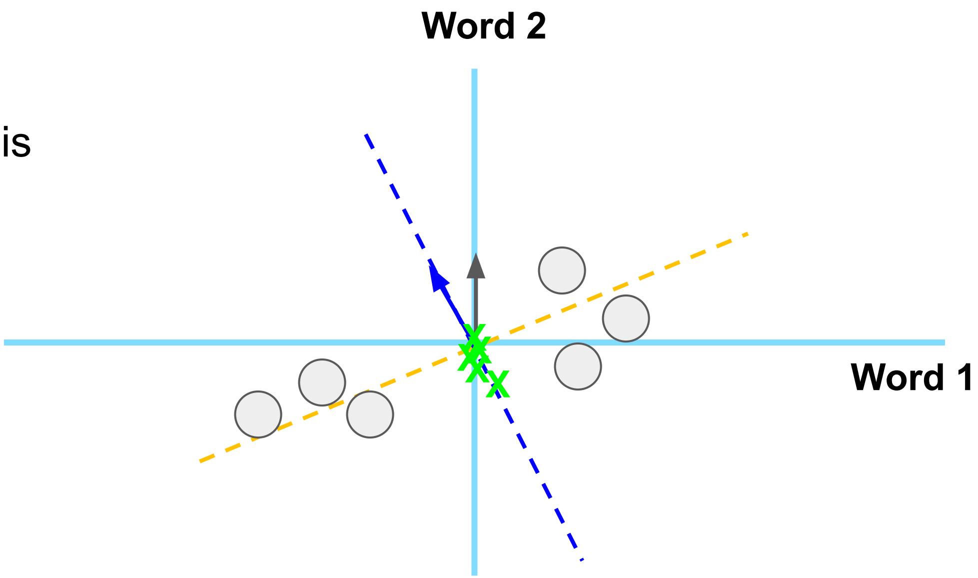
Finding PC2

- PCA calls the SS(distances) for the best fit line the **Eigenvalue for PC2**

$$SS(\text{distances for PC2})$$

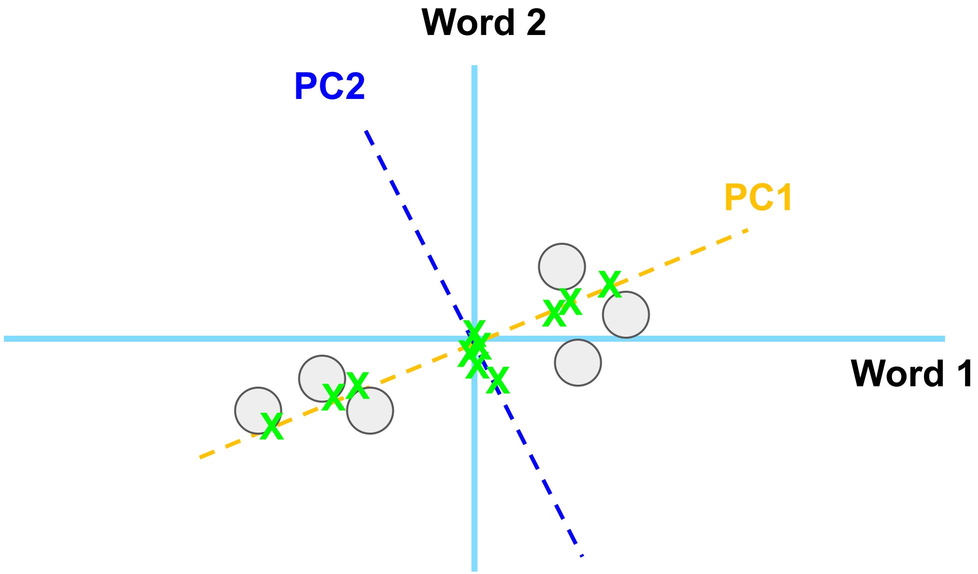
- The square root of the Eigenvalue is **Singular Value for PC2**

$$\sqrt{\text{Eigenvalue for PC2}}$$



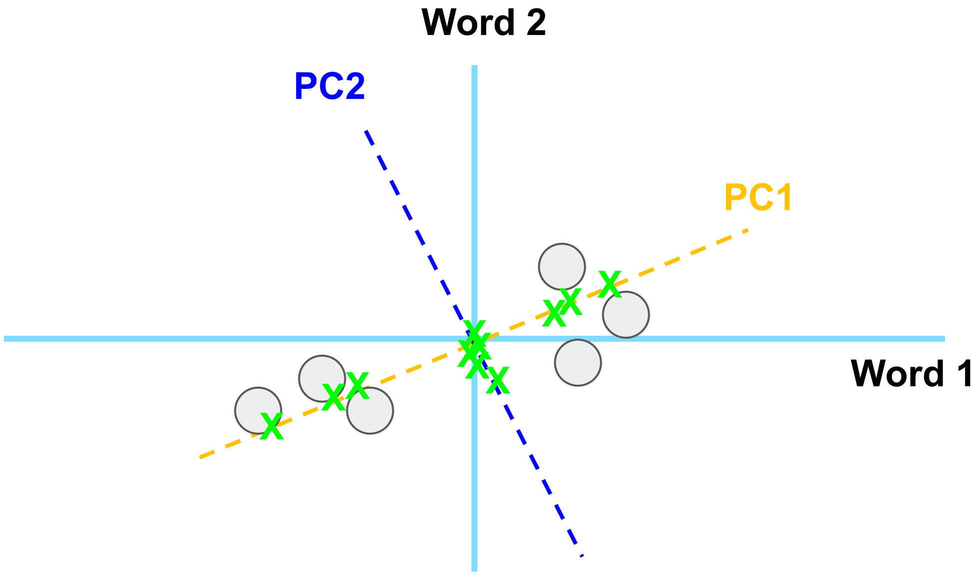
We have computed PC1 and PC2 - Yay!

- PC1 and PC2 can be used to represent the data points!



We have computed PC1 and PC2 - Yay!

- PC1 and PC2 can be used to represent the data points!



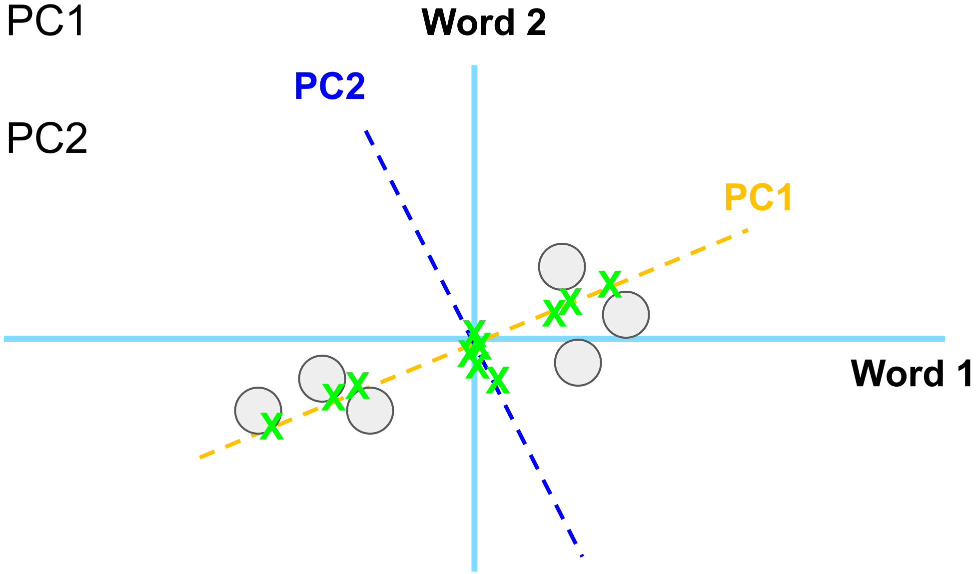
Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

Calculating Variation

- Recall the Eigenvalues?

SS(distances for PC1) = Eigenvalue for PC1

SS(distances for PC2) = Eigenvalue for PC2

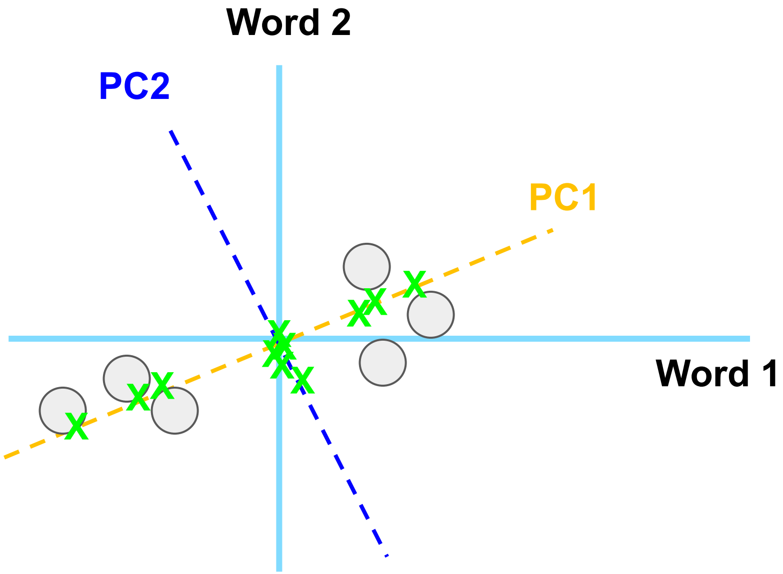


Calculating Variation

- We convert them into variation around the origin (0,0) by dividing by the sample size minus 1 (i.e., n-1)

$$\frac{SS(\text{distances for PC1})}{n-1} = \text{Variation for PC1}$$

$$\frac{SS(\text{distances for PC2})}{n-1} = \text{Variation for PC2}$$

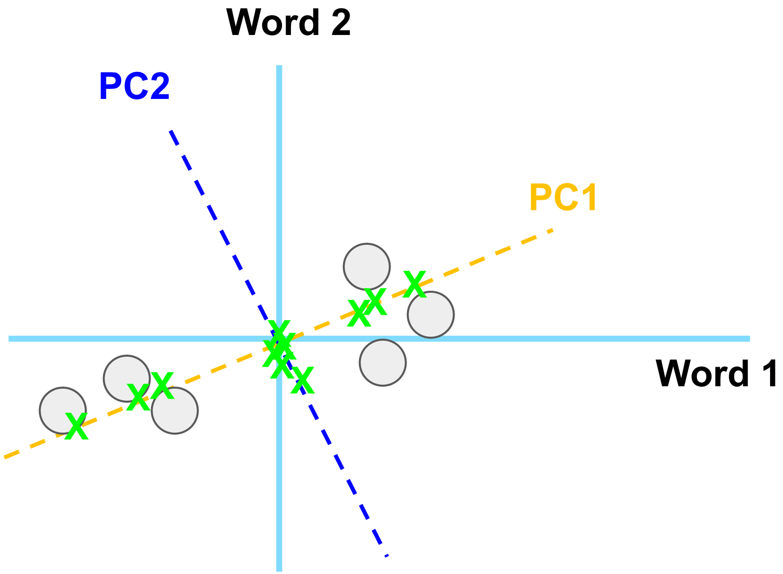


Calculating Variation

- For the sake of the example, imaging the variation for **PC1 = 15**, and variation for **PC2 = 3**.
- Total variation = **15 +3 = 18**

$$\frac{SS(\text{distances for PC1})}{n-1} = \text{Variation for PC1}$$

$$\frac{SS(\text{distances for PC2})}{n-1} = \text{Variation for PC2}$$

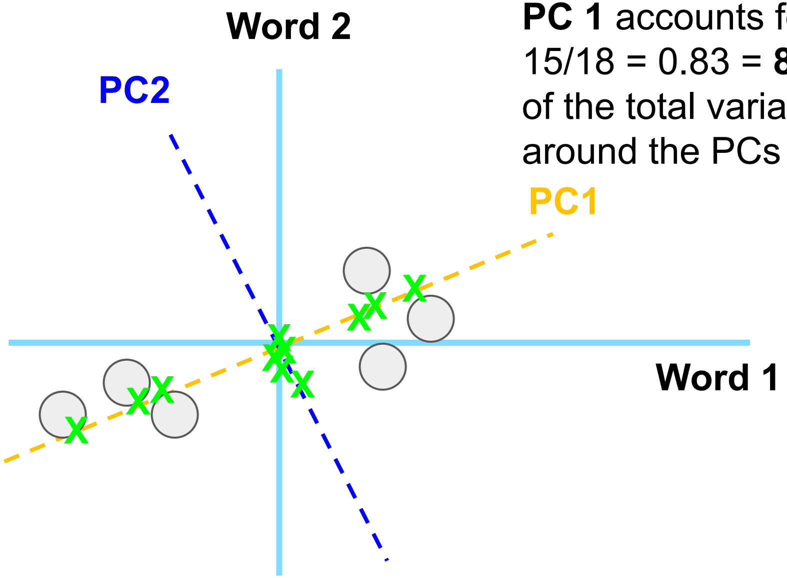


Calculating Variation

- For the sake of the example, imaging the variation for **PC1 = 15**, and variation for **PC2 = 3**.
- Total variation = **15 +3 = 18**

$$\frac{SS(\text{distances for PC1})}{n-1} = \text{Variation for PC1}$$

$$\frac{SS(\text{distances for PC2})}{n-1} = \text{Variation for PC2}$$



PC 1 accounts for 15/18 = 0.83 = **83%** of the total variation around the PCs

Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

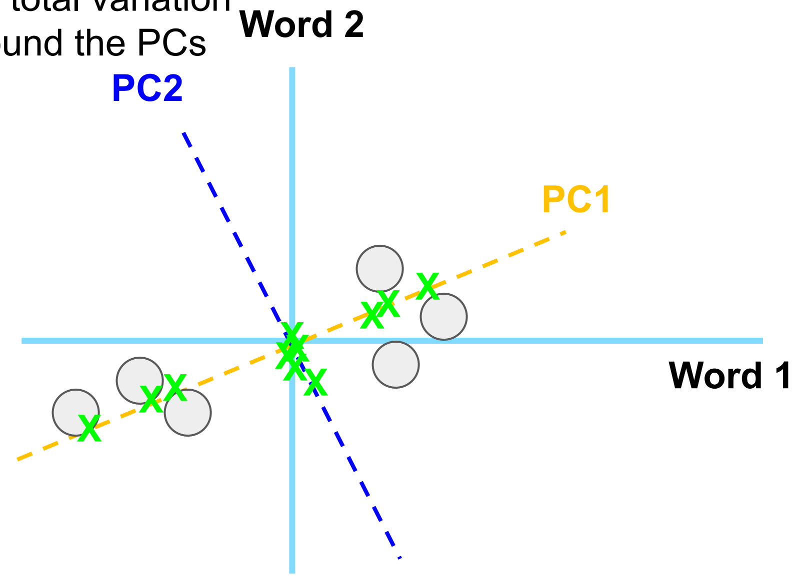
Calculating Variation

- For the sake of the example, imaging the variation for **PC1 = 15**, and variation for **PC2 = 3**.
- Total variation = **15 +3 = 18**

$$\frac{SS(\text{distances for PC1})}{n-1} = \text{Variation for PC1}$$

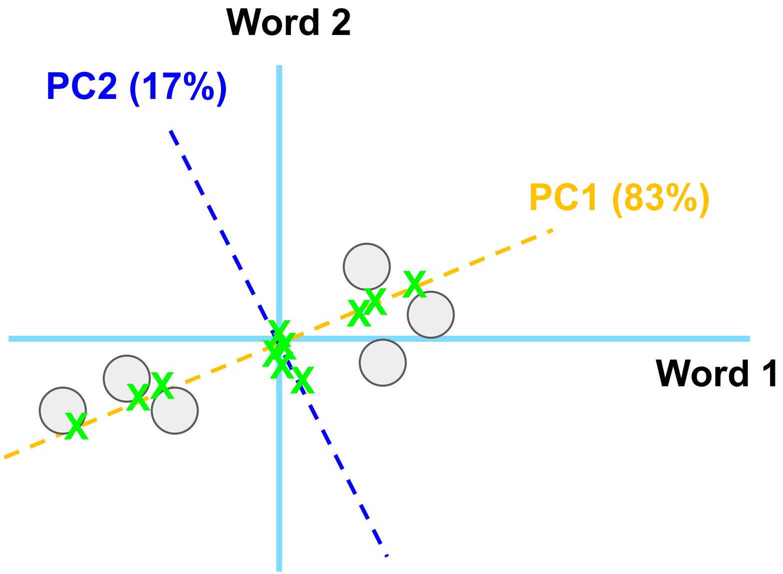
$$\frac{SS(\text{distances for PC2})}{n-1} = \text{Variation for PC2}$$

PC 2 accounts for 3/18 = 0.17 = **17%** of the total variation around the PCs



Calculating Variation

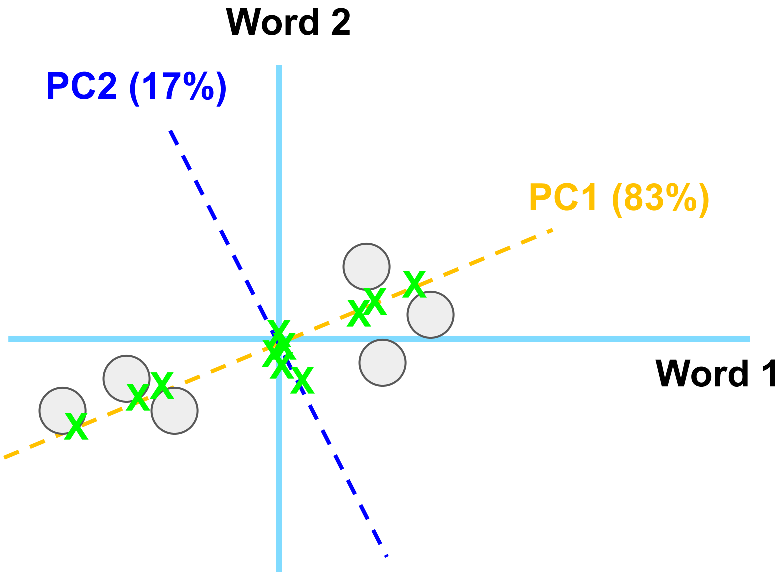
- The higher this variation, the better this PC represents the data point
  - PC1 seems more important :D



Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

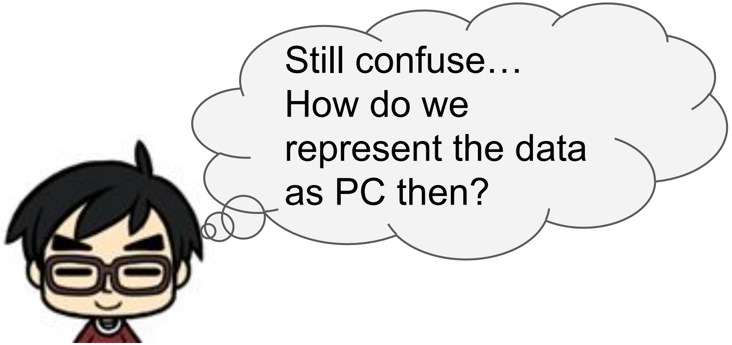
What we know so far…

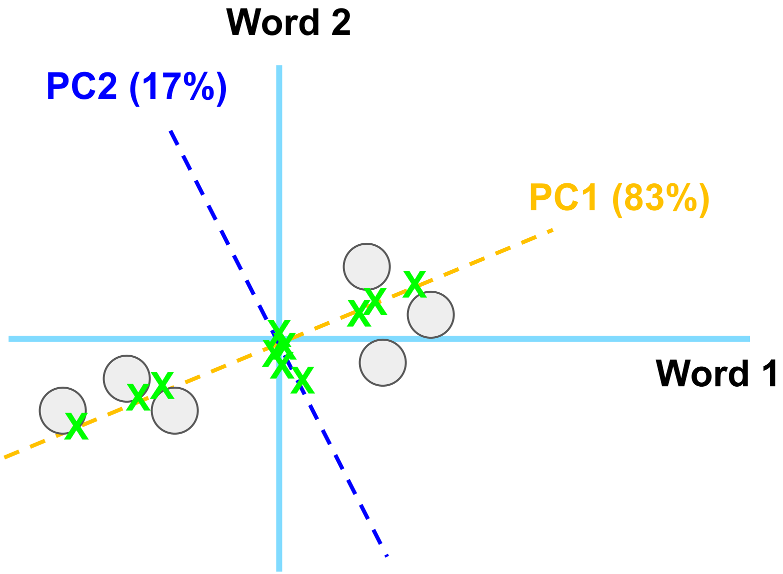
- How to find PC1 and PC2…
- How to tell which PC is more important…



What we know so far…

- How to find PC1 and PC2…
- How to tell which PC is more important…





Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

Projecting Data to PCA

	Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
Word 1	0.2	0.4	0.1	0.7	0.5	0.6
Word 2	0.1	0.3	0.3	0.5	0.3	0.4

	Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
PC1						
PC2						

Projecting Data to PCA

	Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
Word 1	0.2	0.4	0.1	0.7	0.5	0.6
Word 2	0.1	0.3	0.3	0.5	0.3	0.4

	Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
PC1	0.218					
PC2						

- Recall your Eigenvector/Singular Vector for **PC1**

$$\begin{pmatrix} \text{Word 1} \\ \text{Word 2} \end{pmatrix} \quad \begin{pmatrix} 0.97 \\ 0.242 \end{pmatrix}$$

- Compute PC1 score for Post 1:
  (0.97 * 0.2) + (0.242 * 0.1) = **0.218**

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Projecting Data to PCA

	Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
Word 1	0.2	0.4	0.1	0.7	0.5	0.6
Word 2	0.1	0.3	0.3	0.5	0.3	0.4

	Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
PC1	0.218	0.460	0.169	0.8	0.557	0.678
PC2						

- Recall your Eigenvector/Singular Vector for **PC1**

$$\begin{pmatrix} \text{Word 1} \\ \text{Word 2} \end{pmatrix} \begin{pmatrix} 0.97 \\ 0.242 \end{pmatrix}$$

- Repeat compute PC1 score for all posts…

Projecting Data to PCA

	Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
Word 1	0.2	0.4	0.1	0.7	0.5	0.6
Word 2	0.1	0.3	0.3	0.5	0.3	0.4

	Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
PC1	0.218	0.460	0.169	0.8	0.557	0.678
PC2	0.048					

- Recall your Eigenvector/Singular Vector for **PC2**

$$\begin{pmatrix} \text{Word 1} \\ \text{Word 2} \end{pmatrix} \begin{pmatrix} -0.242 \\ 0.97 \end{pmatrix}$$

- Compute PC2 score for Post 1:
  (-0.242 * 0.2) + (0.97 * 0.1) = **0.048**

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Projecting Data to PCA

	Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
Word 1	0.2	0.4	0.1	0.7	0.5	0.6
Word 2	0.1	0.3	0.3	0.5	0.3	0.4

	Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
PC1	0.218	0.460	0.169	0.8	0.557	0.678
PC2	0.048	0.194	0.266	0.315	0.17	0.242

- Recall your Eigenvector/Singular Vector for **PC2**

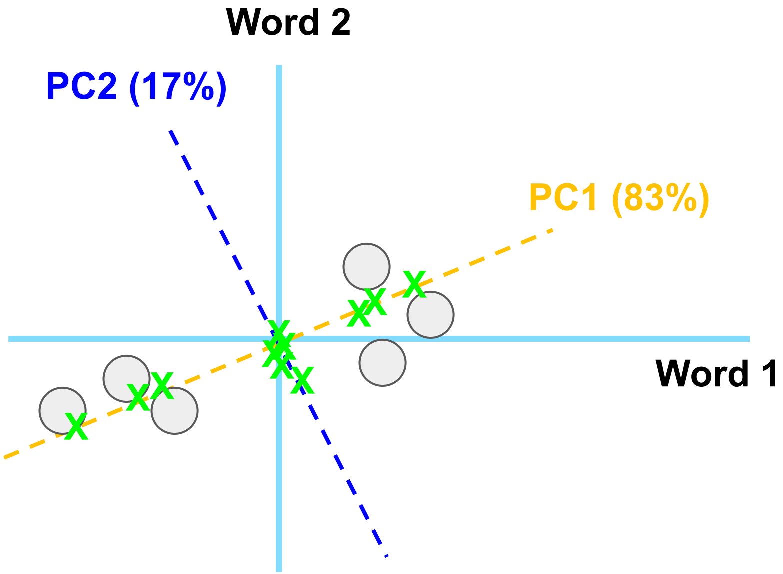
$$\begin{pmatrix} \text{Word 1} \\ \text{Word 2} \end{pmatrix} \begin{pmatrix} -0.242 \\ 0.97 \end{pmatrix}$$

- Repeat compute PC2 score for all posts…

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

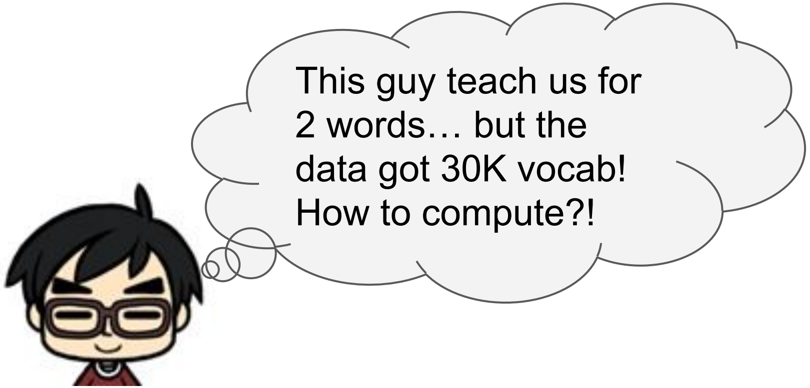
What we know so far…

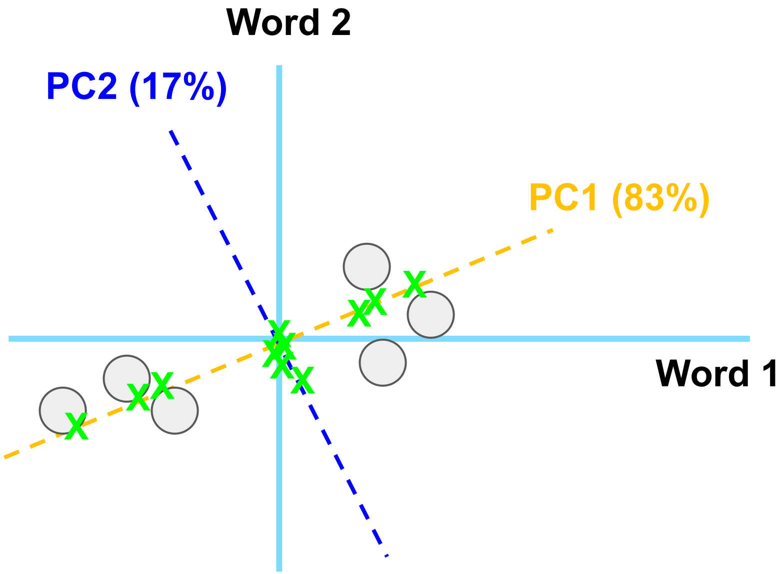
- How to find PC1 and PC2…
- How to tell which PC is more important…
- How to project and represent data into the PCs…



What we know so far…

- How to find PC1 and PC2…
- How to tell which PC is more important…
- How to project and represent data into the PCs…

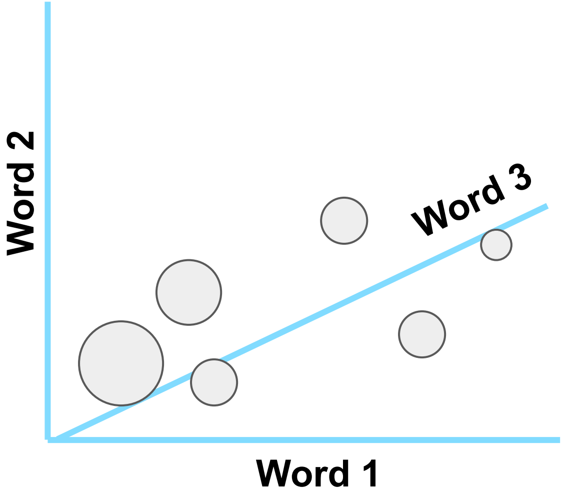




Computing PCA with more Variables

- Let's add another word…

	Post 1	Post 2	Post 3	Post 4	Post 5	Post 6
Word 1	0.2	0.4	0.1	0.7	0.5	0.6
Word 2	0.1	0.3	0.3	0.5	0.3	0.4
Word 3	0.1	0.2	0.1	0.2	0.4	0.3



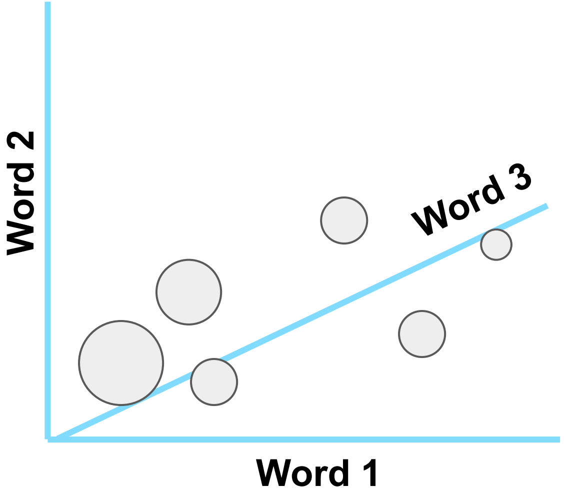
Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

Computing PCA with more Variables

- PCA with three variables is pretty much the same as two variables
    1. Find **PC1** by drawing the line and project the data onto the line
    2. Rotate the **PC1** line till we get best fit
    3. Find **PC2** by drawing a line perpendicular to **PC1**
    4. Find **PC3** by drawing a line perpendicular to **PC1** and **PC2**

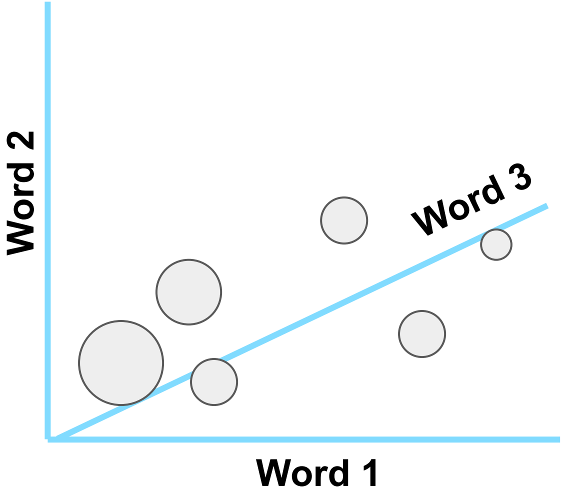
Singular Vector or the **Eigenvector** for the PCs now have 3 values

$$\begin{pmatrix}\text{Word 1}\\ \text{Word 2}\\ \text{Word 3}\end{pmatrix}\begin{pmatrix}0.62\\ 0.15\\ 0.77\end{pmatrix}$$



How to reduce dimension with PCA

- In theory, we can have maximum ***n*** number of PCs, where ***n*** number of variables (i.e., words in the dataset)
- In practice, we just need to use first ***x*** PCs that account for a **high % variation of the dataset**, and simply drop the rest!



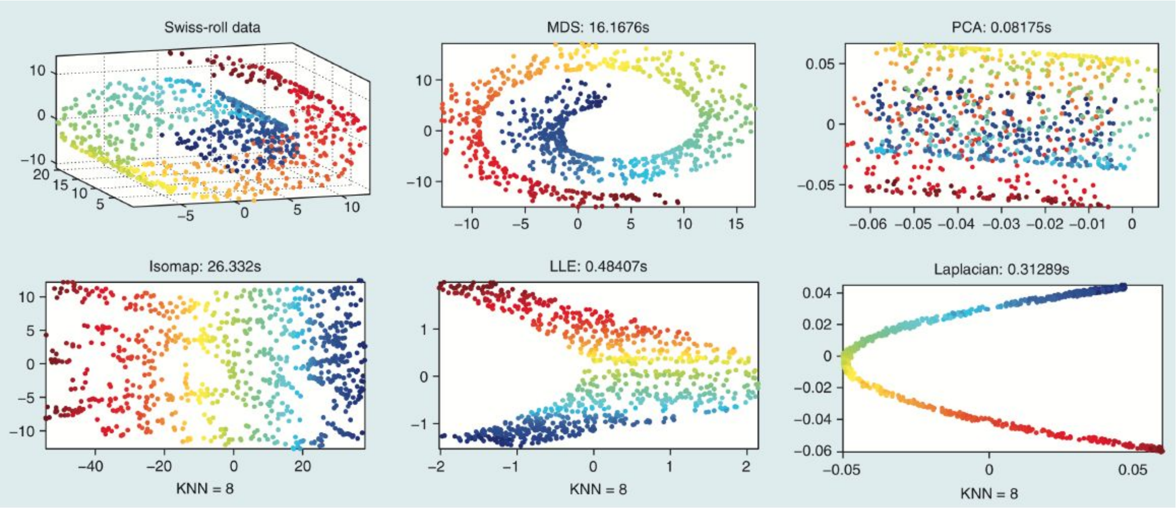
Inspired by video from StatQuest: https://www.youtube.com/watch?v=FgakZw6K1QQ

Additional Material

- For students interested to know more about the math and PCA algorithms:
    1. https://www.cs.cmu.edu/~mgormley/courses/10701-f16/slides/lecture14-pca.pdf

Other Dimension Reduction Techniques

- Autoencoders
- Multidimensional Scaling (MDS)
- Isomap
- Locally Linear Embedding (LLE)
- Laplacian Eigenmaps
- t-SNE



SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN