**WEEK 3**

Vector Space Models

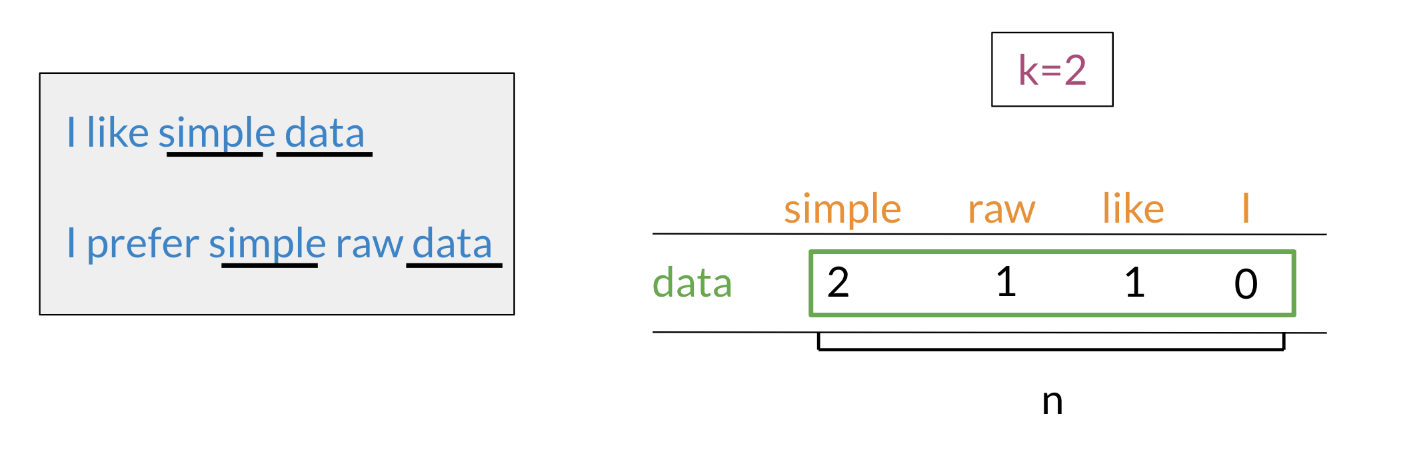
The famous quote by Firth says, **"You shall know a word by the company it keeps".** When learning these vectors, you usually make use of the neighboring words to extract meaning and information about the center word. If you were to cluster these vectors together, as you will see later in this specialization, you will see that adjectives, nouns, verbs, etc. tend to be near one another. Another cool fact, is that synonyms and antonyms are also very close to one another. This is because you can easily interchange them in a sentence and they tend to have similar neighboring words!

There are numerous instances we may decide to employ a vector spaced model, for instance: Information Filtering, Information Retrieval, Machine Translation, Chatbots And many more! In general, Vector space models allow us to represent words and documents as vectors.

Word by Word and Word by Doc

**Word by word**

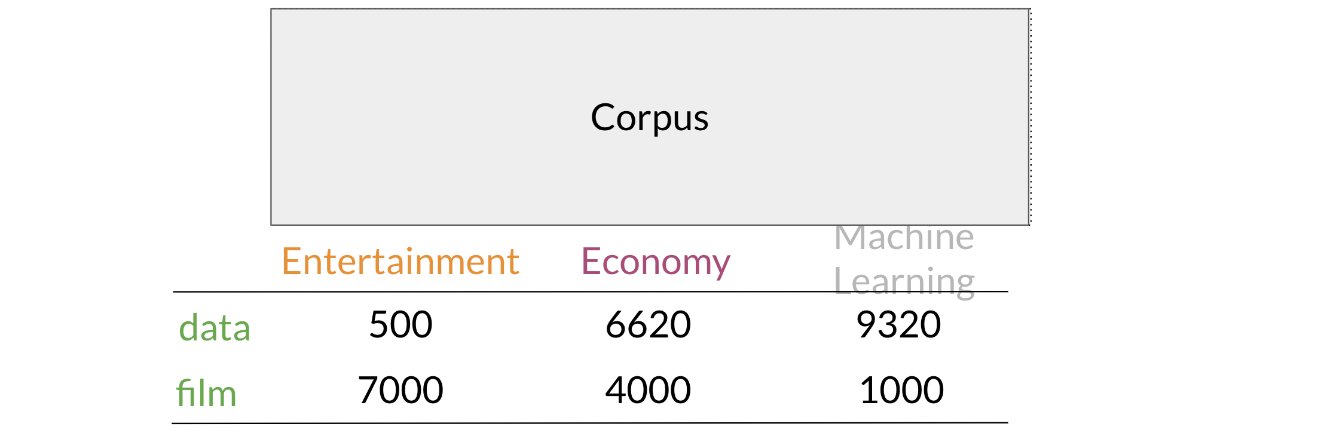
We will start by exploring the word by word design. Assume that you are trying to come up with a vector that will represent a certain word. One possible design would be to create a matrix where each row and column corresponds to a word in your vocabulary. Then you can iterate over a document and see the number of times each word shows up next each other word. You can keep track of the number in the matrix. In the video I spoke about a parameter K*K*. You can think of K*K* as the bandwidth that decides whether two words are next to each other or not.

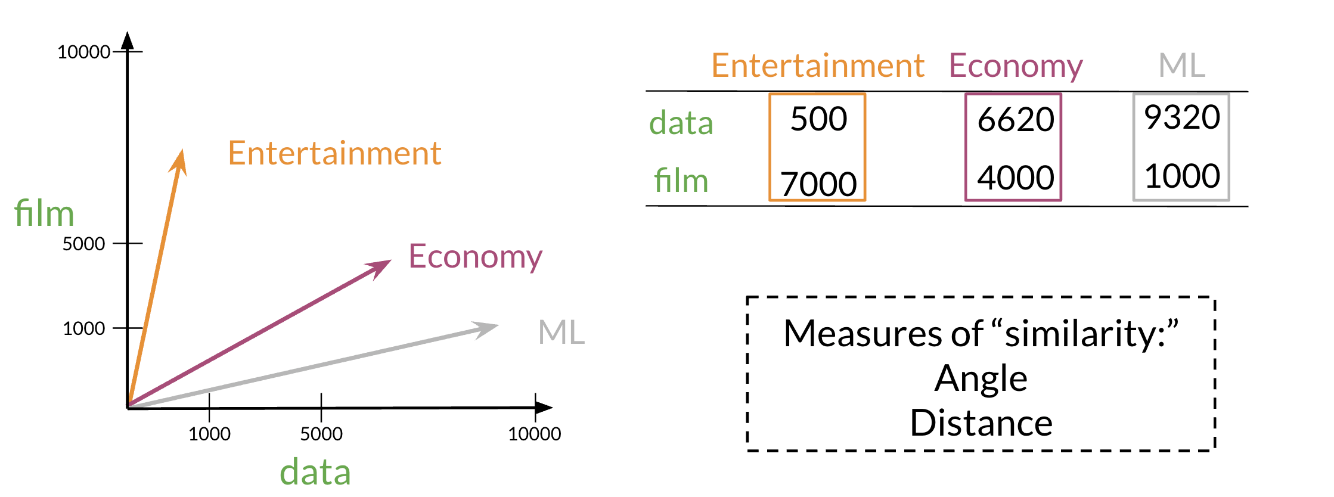


n the example above, you can see how we are keeping track of the number of times words occur together within a certain distance k*k*. At the end, you can represent the word data, as a vector v = [2,1,1,0]*v*=[2,1,1,0].

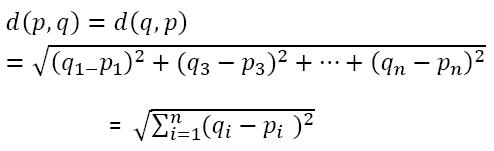
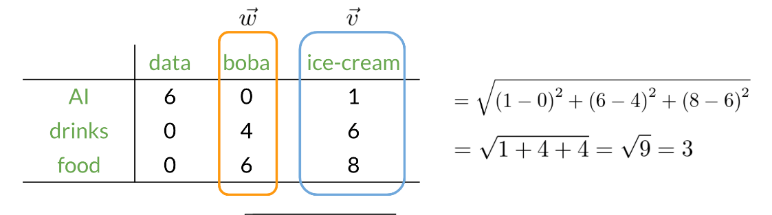
**Word by Document Design**

You can now apply the same concept and map words to documents. The rows could correspond to words and the columns to documents. The numbers in the matrix correspond to the number of times each word showed up in the document.

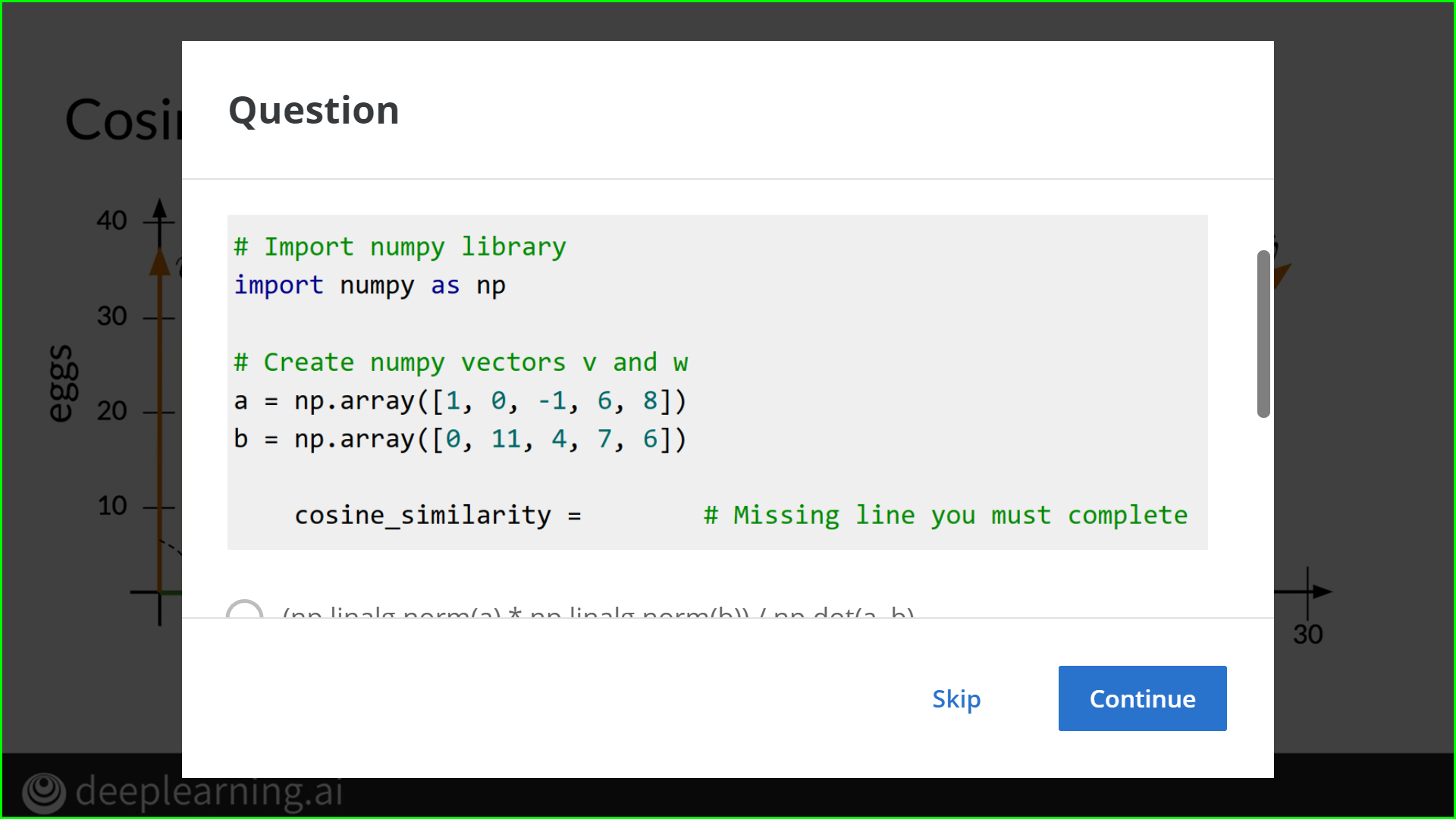


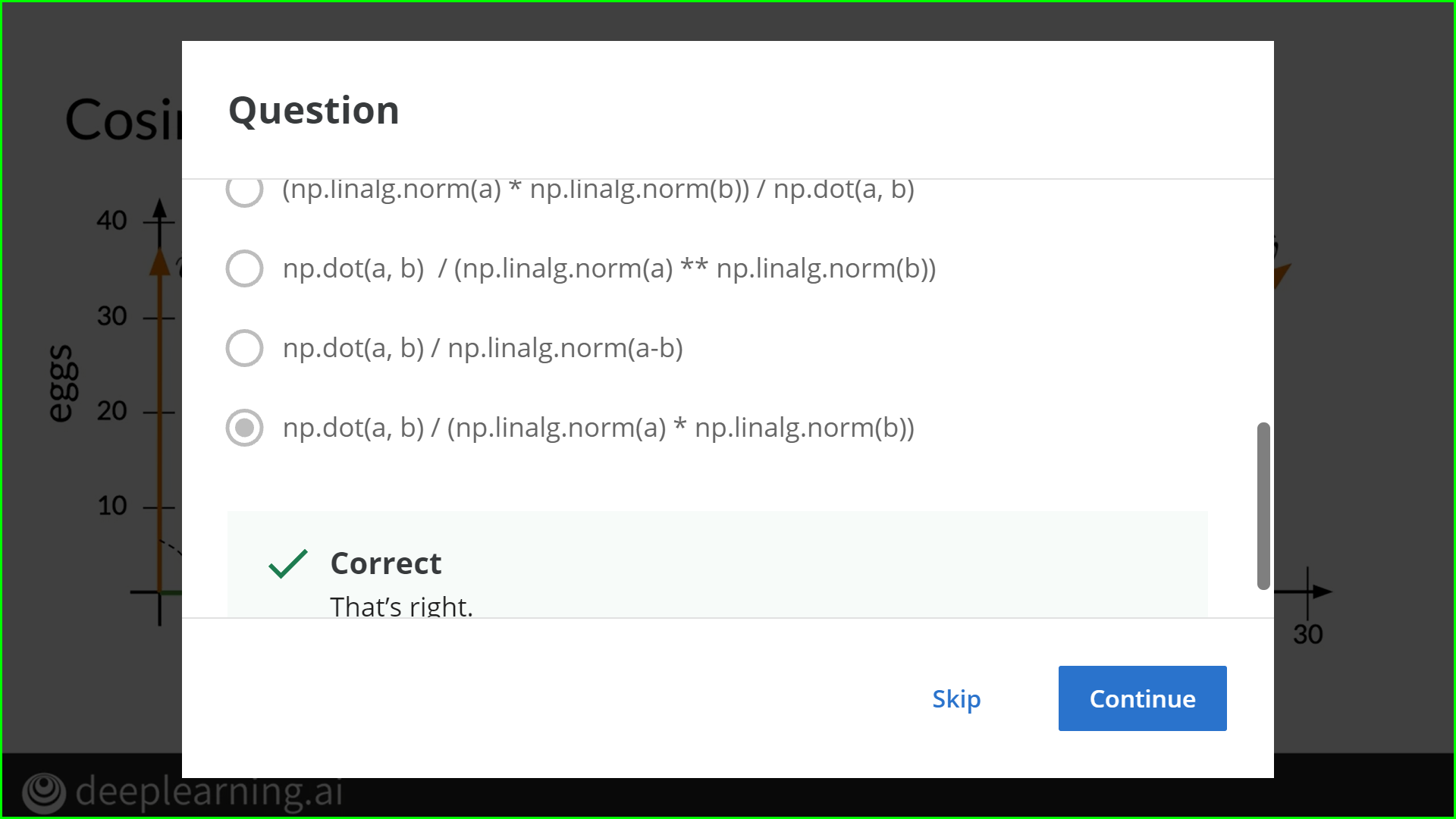
You can represent the entertainment category, as a vector v = [500, 7000]*v*=[500,7000]. You can then also compare categories as follows by doing a simple plot. 

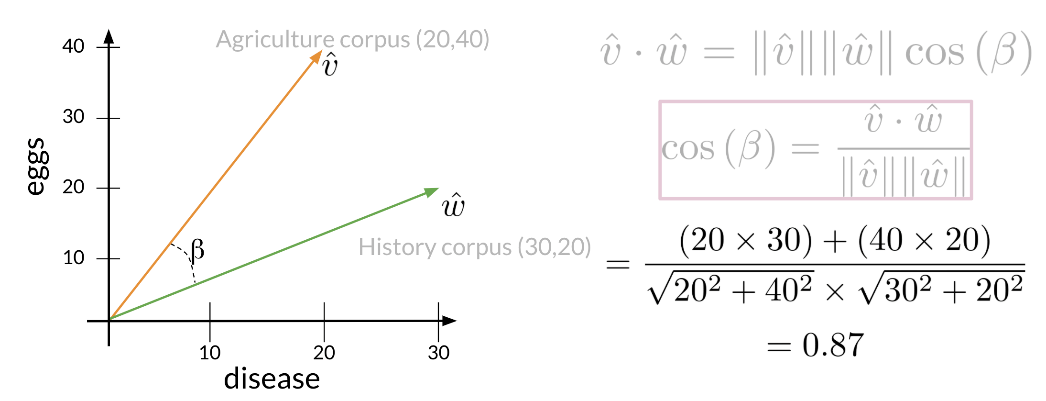
Euclidian Distance



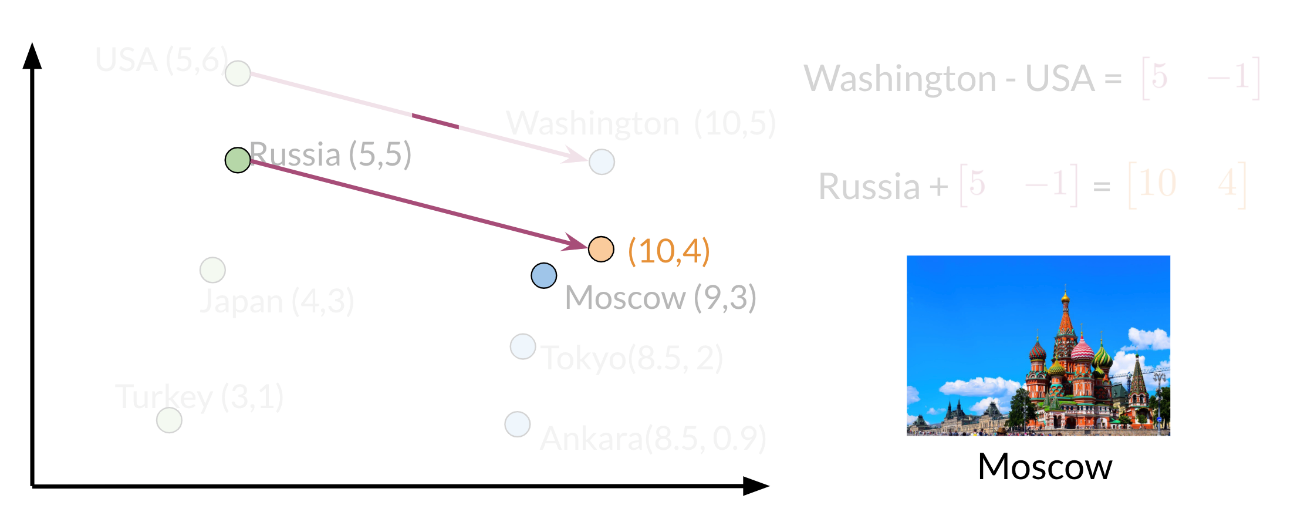
# Cosine Similarity





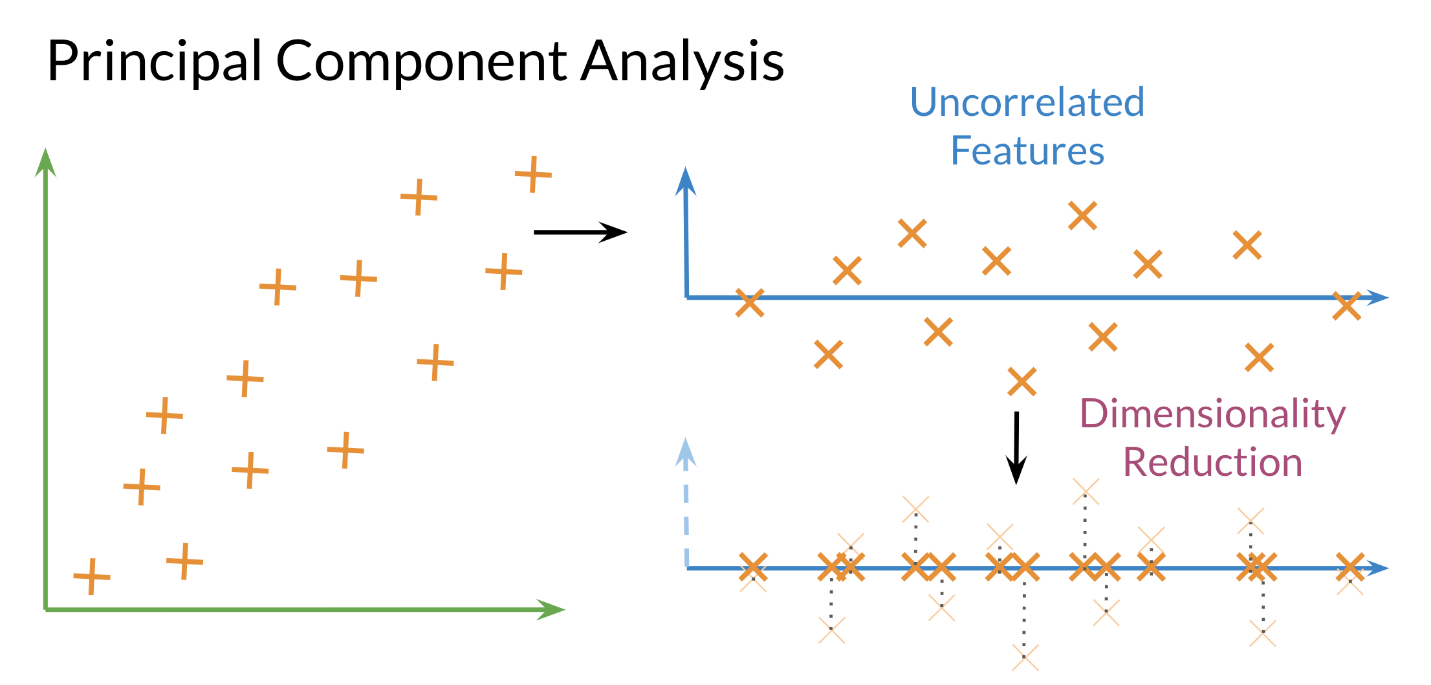


# Manipulating Words in Vector Spaces



# PCA

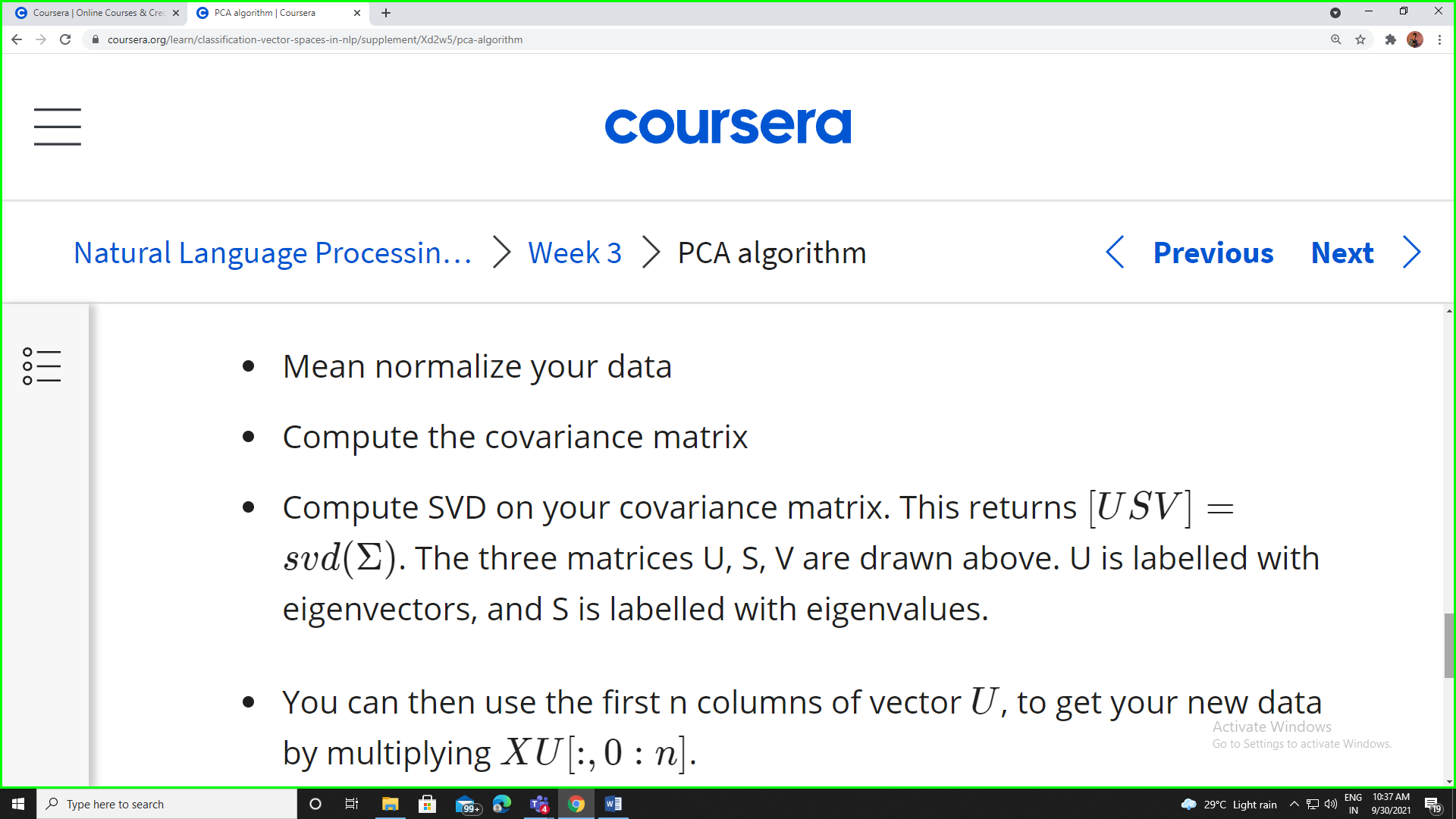
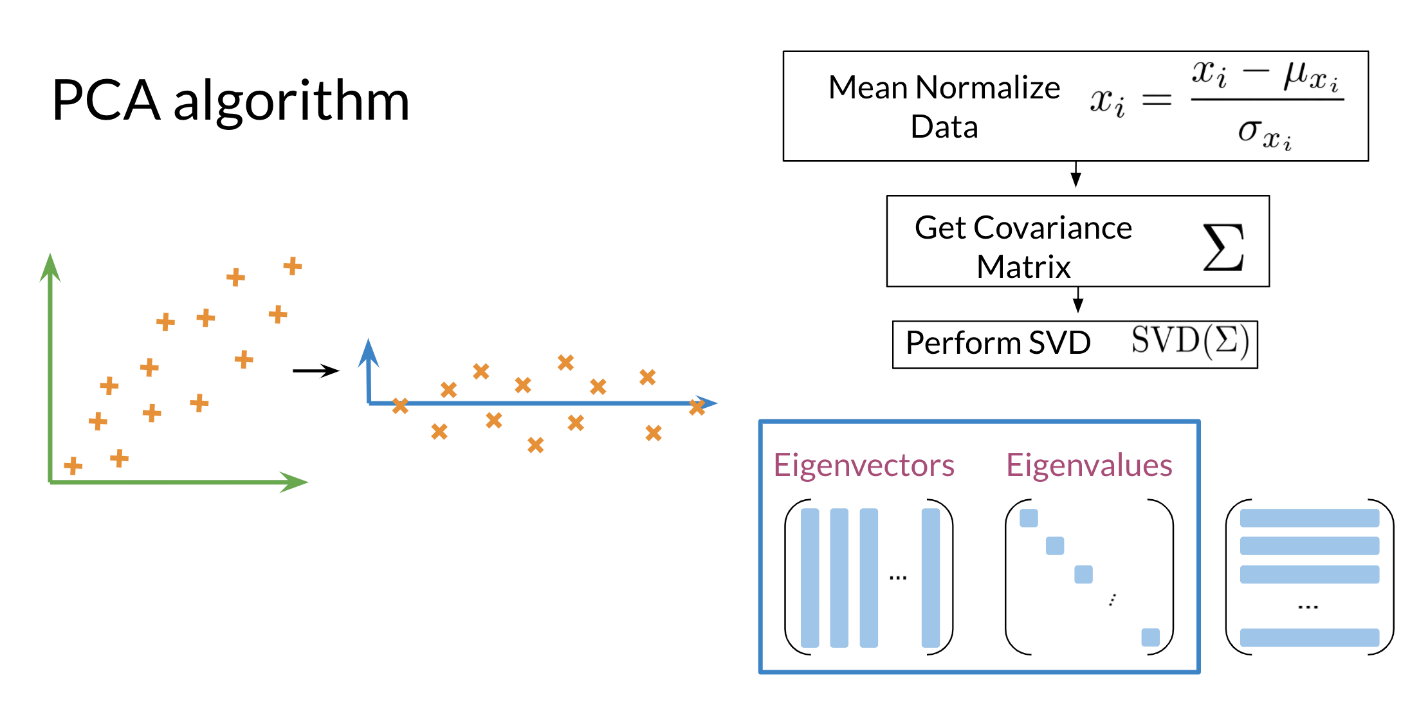
PCA is commonly used to reduce the dimension of your data. Intuitively the model collapses the data across principal components. You can think of the first principal component (in a 2D dataset) as the line where there is the most amount of variance. You can then collapse the data points on that line. Hence you went from 2D to 1D. You can generalize this intuition to several dimensions.



**Eigenvector**: the resulting vectors, also known as the uncorrelated features of your data

**Eigenvalue:** the amount of information retained by each new feature. You can think of it as the variance in the eigenvector.

Also each **eigenvalue** has a corresponding eigenvector. The eigenvalue tells you how much variance there is in the eigenvector. Here are the steps required to compute PCA:



When should I use PCA?

Do you want to reduce the number of variables, but aren’t able to identify variables to completely remove from consideration? Do you want to ensure your variables are independent of one another? Are you comfortable making your independent variables less interpretable?