**Embeddings**

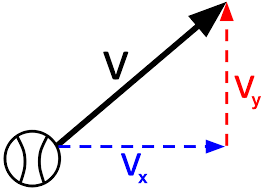
### **Intro:**

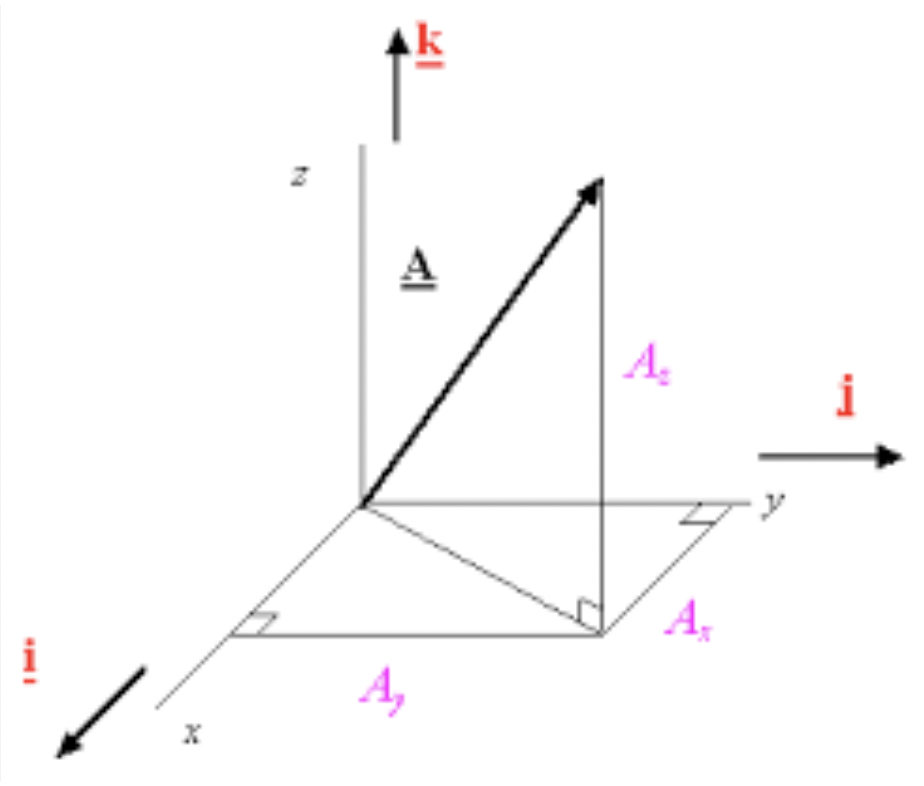
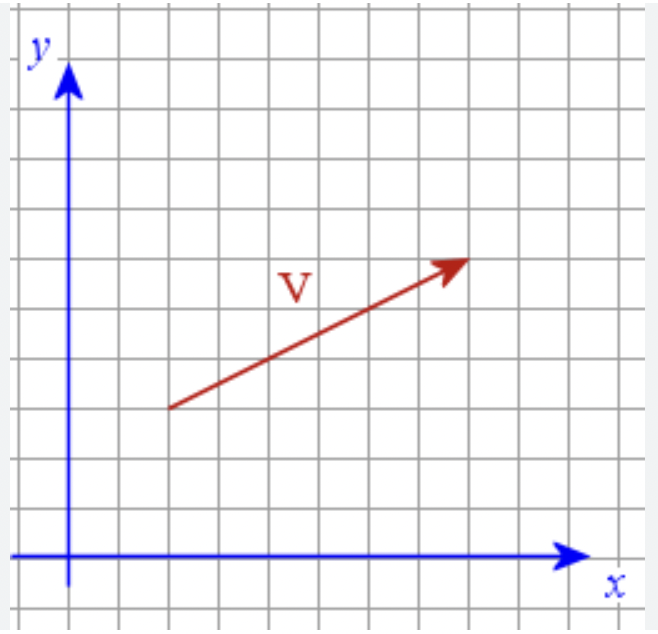
Vector embedding is a popular technique to represent information(Audio, video, text ) in a format that can be easily processed by algorithms, especially deep learning models.

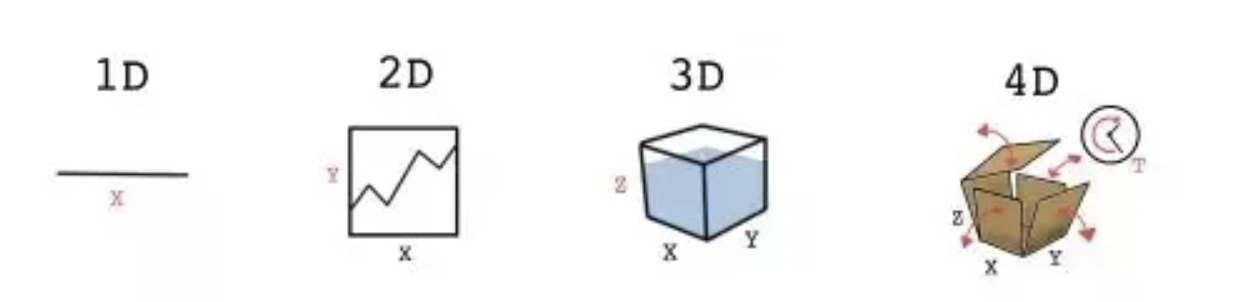
### **What is a Vector?**

Simply put, it is a numeric representation of any information (Text, Audio, Video, Image). Let’s follow an example to understand it better:

2-Dimensional Vector: **VELOCITY**







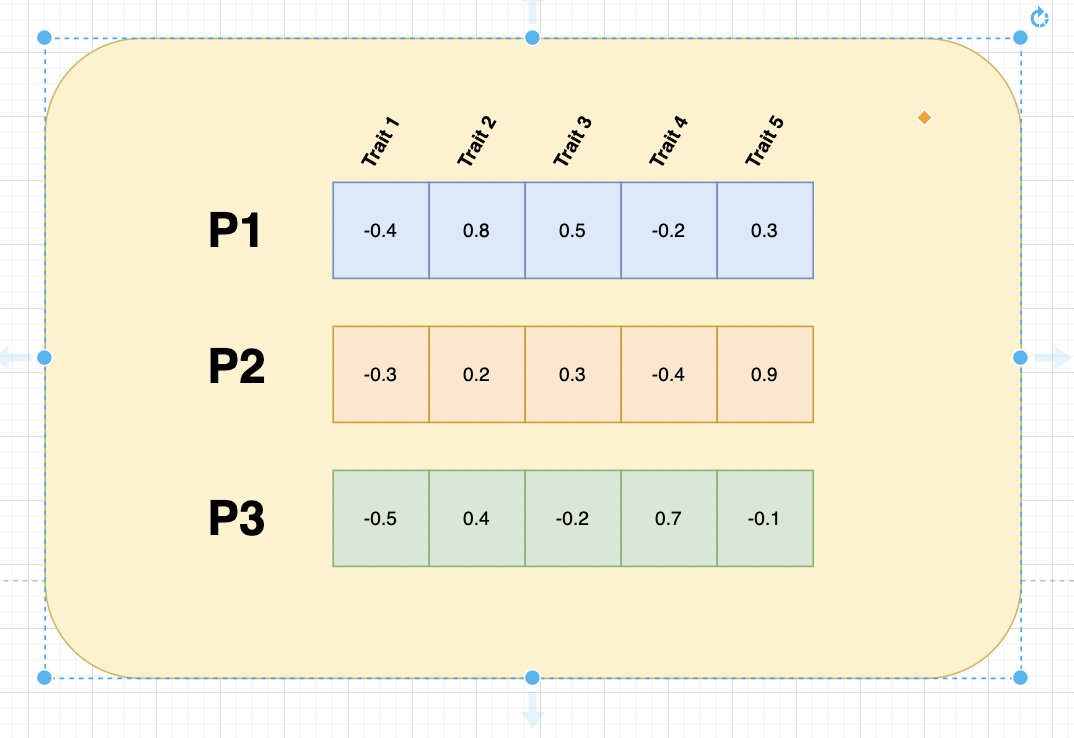
Let’s consider there are 5 traits that can represent a person’s personality:

****

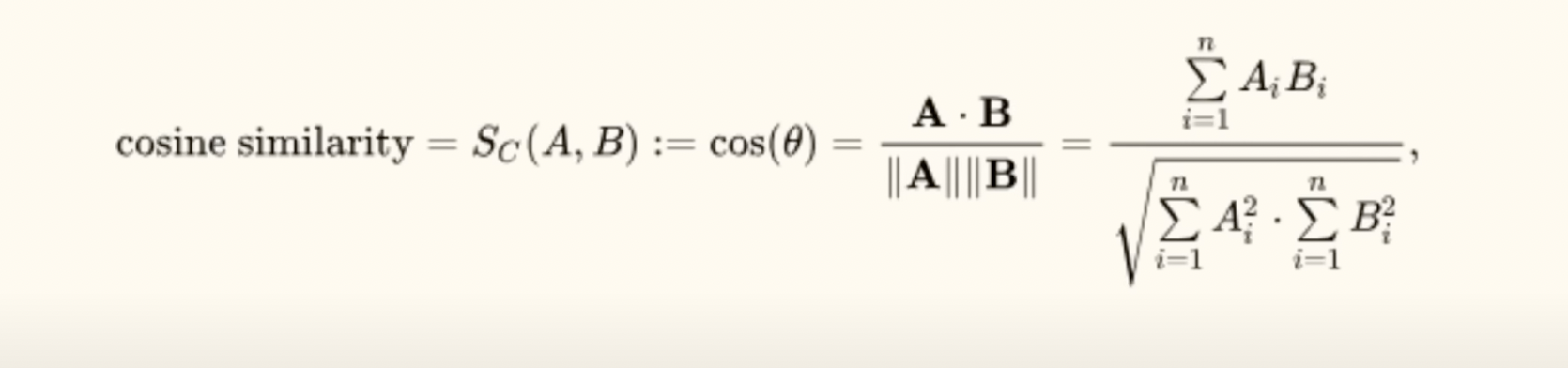
These are different 5 dimensions to show a person’s personality.

Let's assume the scale of each dimension is -1 to 1.

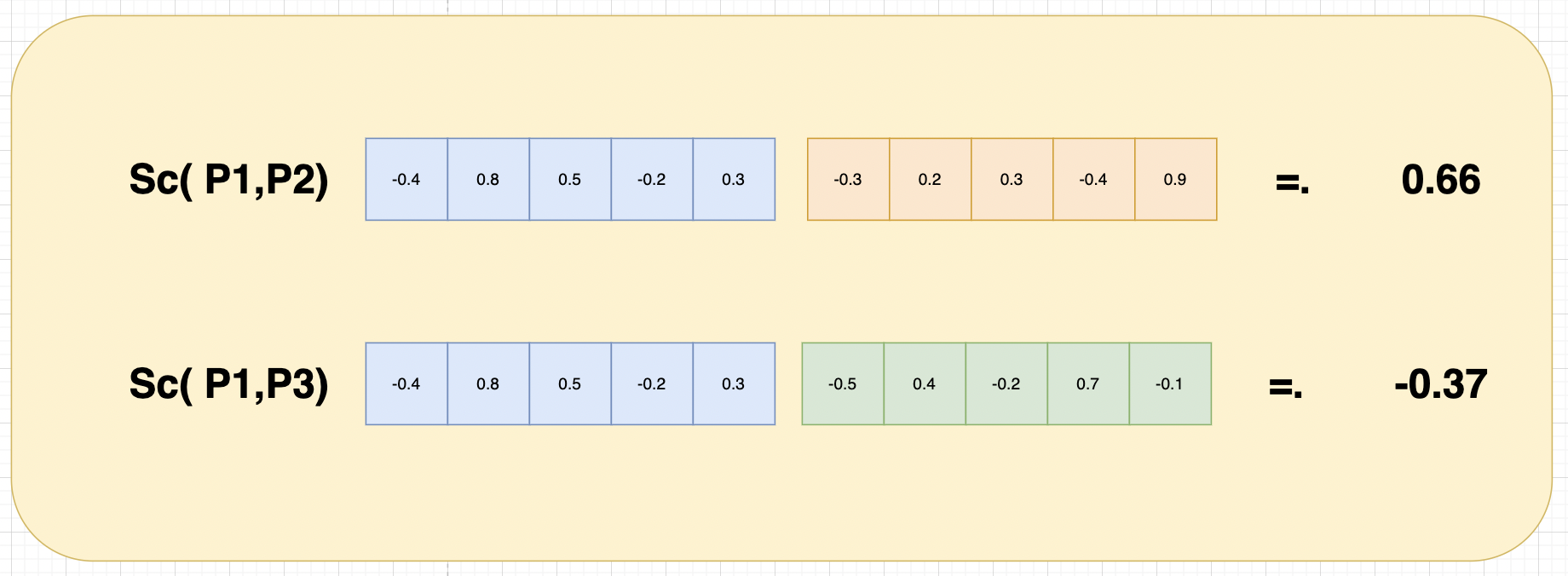
Now assume 3 persons P1, P2 and P3, and we need to figure out whether P1 is more similar to P2 or P3.



For this, we’ll use cosine similarity:



**After applying Cosine Similarity:**

****

That proves P1 is more similar to P2 in terms of personality.

Hence,

* Vectors are the numeric representation of data and related context.
* This is an array of floats which represents its numeric representation and context.
* Vectors are produced by sending data through an encoding model.

Data -> Encoder -> Vectors

Image/text/audio

**Few Usecase of Embeddings and Vector Database:**

* **Search** (where results are ranked by relevance to a query string)
* **Clustering** (where text strings are grouped by similarity)
* **Recommendations** (where items with related text strings are recommended)
* **Anomaly detection** (where outliers with little relatedness are identified)
* **Classification** (where text strings are classified by their most similar label)

### **Search using Embeddings:**

* Open AI Embedding Documentation: <https://platform.openai.com/docs/guides/embeddings>
* API References: <https://platform.openai.com/docs/api-reference/fine-tuning/object>

**NOTEBOOK->**

* Open AI Embeddings
* Gensim - Tool for WordToVec  
  Gensim is an open-source Natural Language Processing Library  
  <https://radimrehurek.com/gensim/models/word2vec.html>  
  It uses WordToVec to create vectors from words and vectors from documents.

**Embedding-gensim:** <https://colab.research.google.com/drive/1ETJSXIxlBlQx6TiZsH8rtGyUHpQcMY-6#scrollTo=ZTJj_pk1dwWq>

**Embeddings-OpenAI:**  
<https://drive.google.com/file/d/1_Lm6hIZpKIdctNGnPXw3FnKlLjuqLLVA/view?usp=drive_link>

**Vector DB- Simulation:**  
<https://drive.google.com/file/d/1bqC-bh328jT9126yfUP8G8Y4zFW9FyKv/view?usp=drive_link>

**Sample text:**  
text1 = "The decomposition of organic waste in landfills generates methane, a potent greenhouse gas."

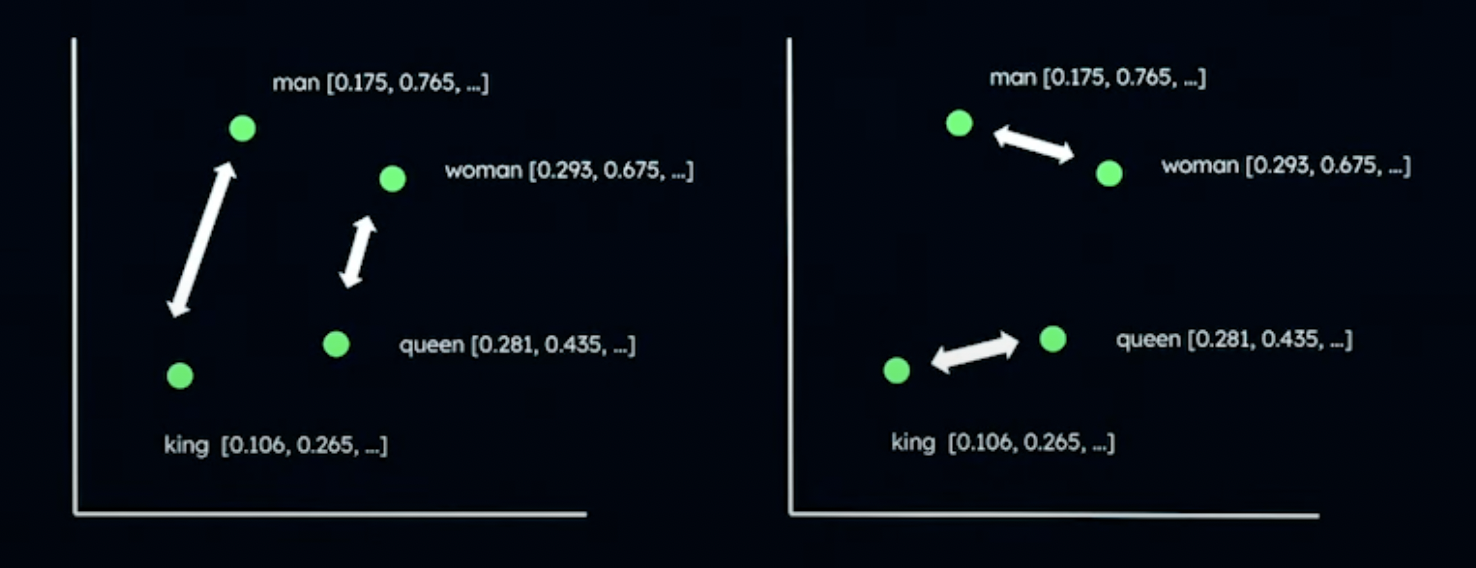
text2 = "The emission of greenhouse gases like carbon dioxide and methane is a key factor in climate change."

text3 = "Changes in climate patterns can be attributed to human activities such as burning fossil fuels and deforestation. "

text4 = "The recipe for a delicious apple pie."

text5 = "How to train a pet dog to do tricks."

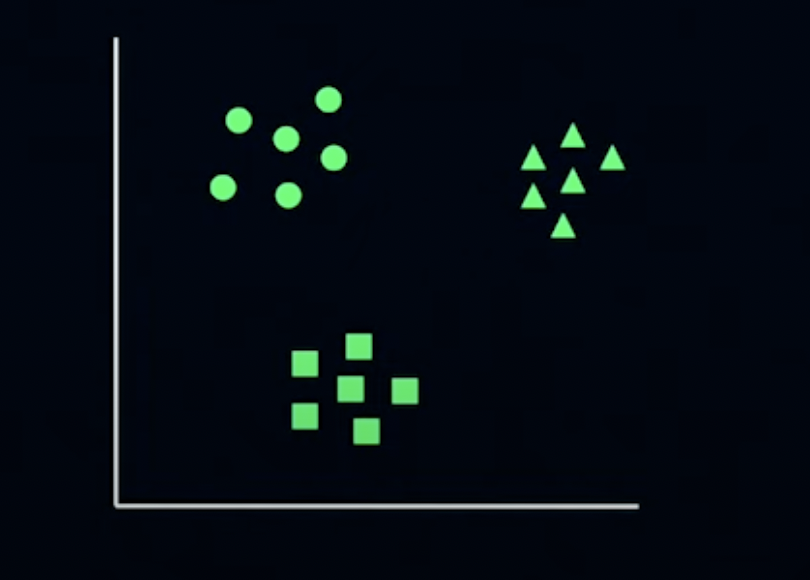
**Similar vectors plotted in space will be near one another.**



* Scenario I: Using one embedding model with a distance function may result in a man and a king being more semantically similar whereas a Woman and a queen are more similar to each other.
* Scenario 2: Using another embedding model or distance function may result in the king and queen being more semantically similar whereas man and woman are more to each other.

**Distance Functions:**

Data transformed into vectors creates clusters of semantically similar data:



K-nearest algorithm is one of the most popular algo to search similar vectors:

Where K represents the number of neighbours you are looking for.

When we use K-nearest neighbour, we have to define the similarity function as well. There are a number of similarity functions but the following are the most popular ones:

* **Euclidean** - Distance between ends of the vectors.  
  Use case: It is specifically used for dense data where specific values matter, (e.g. image similarity, where it is very important that this pixel should be equal to this pixel.)
* **Cosine** - based on the angles of between 2 vectors.  
  Use case: for sparse data where orientation is more important (e.g. text concepts.)
* **Dot product** - based on the angles between two vectors as well as the magnitude.

Use case: For sparse data where both the orientation and intensity matter.